华大半导体有限公司
HUADA SEMICONDUCTOR CO.,LTD.

HC16L 系列

16-bit 超低功耗 MCU

用户手册

日　　期：　**2015-8**

版　　本：　**1.30**

华大半导体有限公司

声明

- 华大半导体有限公司（以下简称华大）保有在不事先通知的情况下而修改这份文档的权利。华大认为提供的信息是准确可信的。本文档信息于2014年3月开始使用。在实际进行生产设计时，请参阅各产品最新的数据手册等相关资料以获取本公司产品的最新规格。
- 华大对本手册拥有包括版权等知识产权，受法律保护。未经本公司事先书面许可，任何单位及个人不得以任何方式或理由对本手册进行复制、修改、抄录、传播等。本文件所登载内容的错误，本公司概不负责。
- 华大对于因使用本文件中列明的本公司产品而引起的，对第三方的专利、版权以及其它知识产权的侵权行为概不负责。本文件登载的内容不应视为华大对其他公司或个人所拥有的专利、版权以及其它知识产权做出任何明示或默示的许可及授权。
- 本文件中的电路、软件以及相关信息仅用以说明半导体产品的运作和应用示例。用户如在设备设计中应用本文件中的电路、软件以及相关信息，应自行负责。对于用户或其他人因使用了上述电路、软件以及相关信息而引起的任何损失，华大概不负责。
- 另外，华大的产品不建议应用于生命相关的设备和系统。在使用该器件中因为设备或系统运转失灵而导致的损失，华大不承担任何责任。
- 虽然本公司致力于提高半导体产品的质量及可靠性，但用户应知晓并同意，我们仍然无法完全消除出现产品缺陷的可能。为了最大限度地减少因本公司半导体产品故障而引起的对人身、财产造成损害（包括死亡）的危险，用户务必在其设计中采用必要的安全措施，如冗余度、防火和防故障等安全设计。

导读

■ 本手册的功能

本手册主要介绍了本系列产品的功能特性，使用方法以及相关规格参数。
使用本系列产品前，请系统阅读本手册。

■ 标记

1. 寄存器说明中的标记如下所示:

- 类型:各位的读/写属性

  R：只读

  W：只写

  R/W：读/写

  -：未定义

- 复位值:复位后寄存器的初始值

  0：初始值”0”

  1：初始值”1”

  X：初始值不定

2. 地址等数值的标记如下所示:

- 16进制数值：”0x”作为前缀标记在数值前面(例:0x0000)；

  “H”或者”h” 作为后缀标记在数值后面(例:000h)

- 2进制数值： ”B”或者”b”作为后缀标记在数值后面(例:000B)

■ 术语

双字: 以 32位为单位进行访问
字： 以 16 位为单位进行访问
半字： 以 8 位为单位进行访问
字节： 以 8 位为单位进行访问

目录

1 简介

HC16Lxx 是一款旨在延长便携式测量系统的电池使用寿命的超低功耗MCU。集成了12位高精度SARADC（逐次逼近型ADC），160段LCD驱动外设，硬件随机数发生器以及硬件DES加解密电路。

HC16Lxx内核采用哈佛结构以及增强型流水线架构，在相同时钟频率下的处理能力为标准8xC251处理能力的3倍，并且比标准型80C51处理能力快40倍。配合Keil μVision4调试开发软件，支持C语言和汇编语言，并且与标准8xC251/80C51完全兼容。

超低功耗MCU的典型应用：

- 各类水表，燃气表，热能表等工业仪表
- 血糖监测仪，血压监护仪和心电记录监护仪等健康器材
- 火警探头，智能门锁，无线传感器，无线监控等智能传感器应用
- 各种对于电池供电和功耗苛求的便携式设备等

超低功耗MCU的特征：

- 增强型 80251，16 位 CPU 平台
- HC16Lxx 具有低功耗性能：
  - 0.9μA @ 3V 低功耗模式 1：主时钟关闭，外部 32.768 kHz 晶振时钟关闭，包括上电复位有效，寄存器，RAM 和 CPU 数据保存状态。
  - 1.2μA@3V 低功耗模式 2：除了包括低功耗模式 1 的配置外，RTC 工作并且外部 32.768 kHz 晶振工作。
  - 1.4μA @ 3V 低功耗模式 3：除了包括低功耗模式 1 的配置外，基于电荷泵型的 LCD 工作。
  - 1.6μA @ 3V 低功耗模式 4：除了包括低功耗模式 2 的配置外，基于电荷泵型的 LCD 工作。
  - 1.7μA @ 3V 低功耗模式 5：除了包括低功耗模式 4 的配置外，WDT 使用外部 32.768 kHz 晶振工作。
  - 2.0μA @ 3V 低功耗模式 6：除了包括低功耗模式 4 的配置外，WDT 使用内部专用的 RC OSC 工作。
  - 40μA/MHz@3V 空闲模式：CPU 停止工作，外设模块运行，主时钟运行。
  - 260μA/MHz@3V 工作模式：CPU 和外设模块运行，程序在 Flash 内部运行。
  - 芯片从低功耗模式下的唤醒时间是 3μs。

  上述特性为室温下典型值，具体的电气特性，功耗特性请查阅电气特性一章。

- 具有高安全特性
  - 128 位 JTAG 密码授权保护，使用密码认证机制可以防止外部工具及人员的非法访问。
  - 集成了 DES 协处理器。
  - 集成了硬件 64 位真随机数模块（RNG）。
  - 集成了硬件 16 位循环冗余校验码（CRC）。
  - 每颗芯片具备 8 字节的唯一设备标识号，启动主程序之前，需进行 ID 解密，从而保证客户代码的安全。
  - 可通过 smart card 调用 RNG/DES，通过多重认证机制来加强系统的高安全性。
  - 执行主程序前，通过硬件 DMA 与 CRC 计算功能检验 Flash 内部程序正确后，才启动主程序，保证程序的完整性，增加系统可靠性。
  - 将 MCU 中的部分代码或算法进行 DES 加密放在 Flash 中，运行前解密到 RAM 中，在 RAM 中运行高安全性程序，以保护密钥以及特殊程序，增加程序的安全性。

  通过以上措施，可以增加程序的安全性和系统的可靠性。

2 功能模块

2.1 功能模块图简介

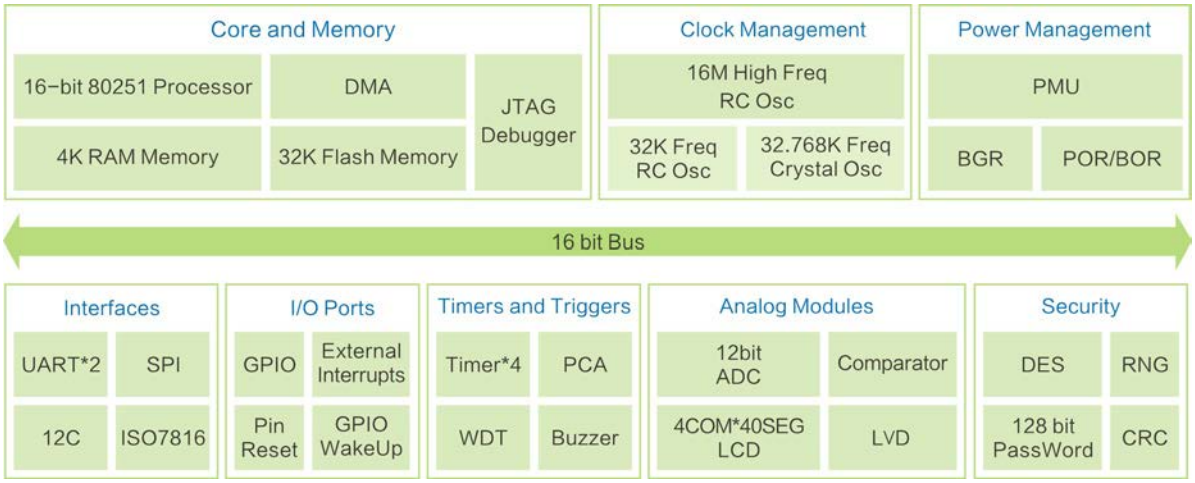


Figure 2-1 HC16Lxx 功能模块简介图

2.2 应用电路

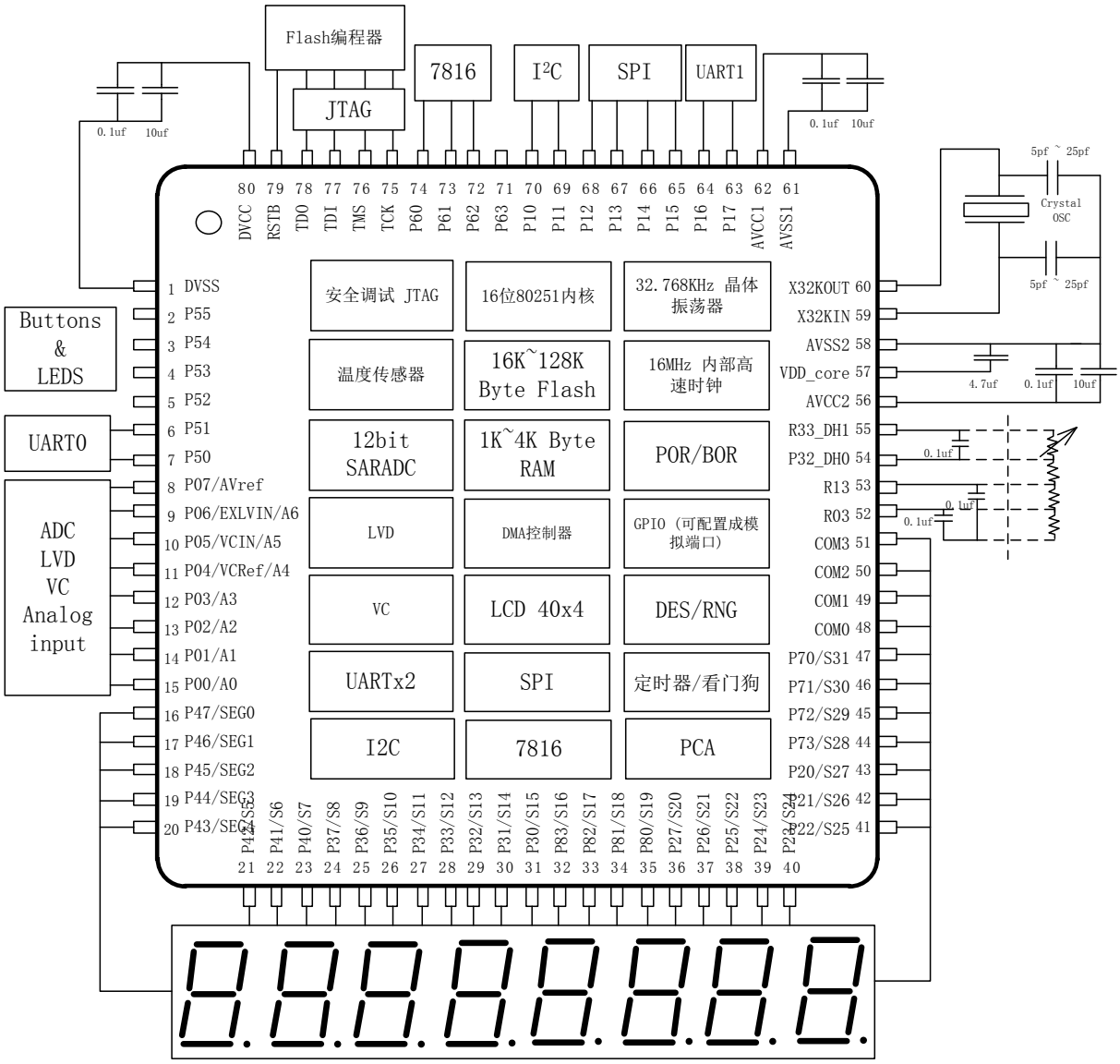


Figure 2-2 HC16Lxx 应用电路

3 功能模块介绍

3.1 增强型流水线 80251 内核

- 嵌入式 80251 是标准 80C51 的加强版，集成了 16 位和 32 位处理性能，与标准 80C51 指令兼容，当编译 C 程序时，能够减少代码容量。
- CPU 针对 8 位和 16 位 MCU 的应用需求增强了处理能力。
- 内核采用哈佛结构以及增强型流水线架构，在相同时钟频率下的处理能力为标准 8xC251 处理能力的三倍，并且比标准型 80C51 处理能力快 40 多倍。

3.2 Flash

- 集成 Flash 控制器，无需外部高压输入，内置操作定时器支持编程，擦除。
- 存储空间大小为 16K － 128K 字节
- 支持在程序运行中编程（IAP）和在线编程（ICP）功能。
- 支持直接存储器存取（DMA）传输模式。
- 每 512 个字节为一个最小擦除页。
- 常温下擦写次数 ≥ 20000 次
- 擦除时间 ≥ 20ms，编程时间 ≥ 20us。
- 常温下数据保存 ≥ 100 年。

3.3 RAM 存储器

- 在各种低功耗模式下，RAM 的数据可保留。
- 存储空间大小为 1K – 8K 字节
- 支持在 RAM 中执行程序。
- 支持 DMA 传输模式。

3.4 通用 IO 端口（GPIO）

- 每个 GPIO 可用于数字功能模块以及模拟模块输入输出复用。
- 每个端口有独立的控制寄存器位。
- 支持边沿、电平触发中断；支持从低功耗模式下唤醒 MCU。
- 支持 Push−pull CMOS 推挽输出，Open−Drain 开漏输出。
- 支持 pull−up 上拉电阻，具备滤波输入。
- 输出驱动能力可控；最大支持 16mA 的驱动能力。
- 支持外部异步中断。

3.5 控制器

3.6 DMA 控制器

通过 DMA 控制器，存储器和外部设备之间直接进行数据传送，不需要 CPU 的参与。

1 路 DMA 通道，支持下列操作：

- Flash → SRAM
- SRAM→ Flash
- Flash → Flash
- SRAM→ SRAM
- SRAM→ CRC
- Flash → CRC

3.7 时钟控制器

- 一个 16M 高精度内部时钟，从低功耗下唤醒时间为 3uS，精度为±2. 5%
- 一个 32. 768KHz 的外部晶振，主要提供 RTC 实时时钟频率
- 一个 32KHz 内部时钟，当外部 X32K 晶振失效时备用，增加抗干扰能力
- 比较器 VC 和低压检测 LVD 共用内部时钟，进行时域滤波，增加抗干扰能力

3.8 中断控制器

- 带有四个中断优先级，能够进行实时控制和中断处理
- 16 个中断入口向量地址
- 一个非可屏蔽中断 NMI，即外部 32K 晶振失效中断
- 一个软件中断，为了 JTAG 调试使用

3.9 复位控制器

- 上电掉电复位
- 低电压检测 LVD 复位
- 外部 RSTB 管脚复位
- 看门狗复位
- 可编程计数器阵列(PCA)看门狗复位
- JTAG 软件复位
- 每个复位都有复位标志寄存器

3.10 定时器/计数器

- 4 个通用 16 位定时器/计数器
- 1 个 20 位可编程计数看门狗（WDT），内建专用超低功耗 RC OSC 提供 WDT 计数
- 2 个专用 8 位定时器给 UART 使用产生可变波特率
- 1 个专用 8 位定时器给 I^2C 使用产生可变波特率
- 1 个专用 8 位定时器给 SPI 使用产生可变波特率
- 1 个专用 8 位定时器给 Buzzer 使用产生驱动频率

3.11 可编程计数器阵列（PCA）

- 5 个 16 位的捕获/比较模块
- 每个模块都可以独立编程，以提供输入捕获，输出比较或脉冲宽度调制功能
- 一个模块可以用作一个额外的 WDT

3.12 看门狗（WDT）

- 一个 20 位可配置定时器，提供 MCU 异常情况复位
- 时钟来源有 2 个，分别是外部 32. 768KHz 晶振和内部 32K 的 IRC 时钟
- 最大溢出时间为 32 秒

3.13 实时时钟（RTC）

- 24/12 小时时间模式，支持 BCD 数据的寄存器
- 32. 768KHz 时钟输入，中断周期可配置为年/月/日/小时/分钟/秒
- 最小可调精度为 1/16 秒
- 具有硬件自动修正闰年的日历功能
- 支持年/月/日/小时/分钟/秒 软件+1，−1 操作
- 支持 2 个可配置频率、宽度、相位的脉冲输出
- 用于指示时间和日期的 RTC 日历记录器在 MCU 受外部因素影响而复位时不会清除保留值

3.14 通信模块

3.15 UART

- 2 个通用 UART 通讯接口
- 支持所有 4 种 8xC251 标准的工作模式
- 具有帧位错误检测
- 多处理器通信
- 自动地址识别
- 独立 2 个 8 位计数器，支持可编程波特率

3.16 SPI

- 1 个通用 SPI 通讯接口
- 支持主从模式，支持全双工半双工模式
- 2 级 FIFO
- 独立 1 个 8 位计数器，支持可编程波特率

3.17 I^2C 接口

- 总线采用串行同步时钟，串行 8 位双向数据传输最大速度可以到 400Kps
- 支持 26 种传输状态
- 为双线、双向串行总线，支持主从收发 4 种传输模式
- 支持 7 位地址自动识别
- 独立 1 个 8 位计数器，支持可编程波特率

3.18 ISO7816 主控器

- 标准智能卡通讯接口；符合 ISO/IEC 7816-3 标准
- 支持 T=0 和 T=1 的传输模式
- 收发自动转换
- 支持重试次数设置
- 支持额外的 ETU 设置
- 支持 ETU 计算器计数
- 支持硬件 LRC/CRC 计算
- 起始位采用 16 次采样判决算法
- 每一位数据三次采样并遵从择多判决算法
- 支持 ISO7816 接口软复位
- 支持接收数据未取走情况下，被新数据覆盖（overrun）标志
- 支持 ISO7816 接口状态（busy/idle）标志

3.19 DES 协处理器

- 64 位标准数据加解密（DES）算法，即对称分组密码算法，明文、密钥、密文均为 64 位
- 安全的硬件架构，能够抵御旁路攻击中的简单功耗攻击(SPA)和差分功耗分析攻击（DPA）

3.20 随机数产生器（RNG）

- 支持 2 种随机数产生，分别是伪随机数，和真随机数
- 一个周期可产生 64 位随机数
- 可以对 Key 密钥和时钟扰乱等抗 DPA 技术提供支持

3.21 循环冗余码校验（CRC）

- 符合 ISO/IEC13239 中多项式 F(x) $=X^{16}+ X^{12}+ X^{5}+1$ 的要求

3.22 蜂鸣器

- 1 个专用 8 位定时器作为驱动
- 灌电流最大支持 16mA(具体参数参考章节“电气特性”中的端口特性)，不需要额外的三极管进行蜂鸣器工作

3.23 A/D 转换器

- 一次采样 ≥ 17 个时钟周期；转换速率高达 200 ksps
- 单调的不失码 12 位转换器
- 可调采样和保持周期，可选采样转化速率
- 可选片上精确参考电压（1. 5v 或 2. 5v）或外部参考电压
- 7 路外部管脚输入通道以及 1 路内部温度传感器通道
- 内建采样单位增益放大器

3.24 模拟电压比较器（VC）

- 芯片管脚电压监测
- 4 通道输入
- 可产生异步中断
- 可配置的硬件迟滞电路
- 软件可配置防抖时间（16us – 64ms）
- 比较器输出可作为计数器的输入使用
- 低功耗模式下可工作、可快速唤醒

3.25 低电压检测器（LVD）

- 可检测芯片电源或芯片管脚电压
- 可配置成中断或复位
- 可选 16 阶电压比较输入（2. 0v – 3. 6v）
- 可配置硬件迟滞电路
- 软件可配置防抖时间（16us – 64ms）

3.26 温度传感器

- 使用 PTAT 架构，精度达到 1℃
- 可做为内部 ADC 采样的一路通道

3.27 上电掉电复位模块（POR）

当芯片接收到电源以及丢失电源时，或当芯片电源低于某一个阈值电压，内部会产生一个POR信号，当芯片高于某一个阈值电压时，释放POR信号。

POR信号会把芯片的寄存器，控制信号全部复位。

3.28 LCD 液晶驱动模块

- 最大支持 40SEG x 4COM 段码显示
- 支持电阻分压以及电荷泵产生参考电压
- 可调帧频率
- 支持 4 种 LCD 显示模式（静态，2 段码，3 段码，4 段码）

3.29 嵌入式调试系统（JTAG）

- 可配合 Keil μVision4 调试开发软件
- 支持 2 个硬断点以及多个软断点
- 支持标准 Rlink，PC 与芯片通过标准 USB-JTAG 连接

3.30 128 位密码

128 位 JTAG 密码授权，保护客户程序不会被不法侵入，在 JTAG 前面有一个密码保护模块，当输入授权的 128 位密码之后，JTAG 端口才被开放出来，使用 Password 认证机制可以防止外部工具及人员的非法访问。

4 HC16L 超低功耗 MCU 系列

家族	平台	项目编号	配置表	Parter number	PKG	Flash	IAP/ISP	RAM	I/O	双电源	双时钟	RTC	CRC	12bit ADC	VC	捕捉模块	PWM	Timer	低功耗计数单元	SPI	I2C	7816	UART	IrDA	Beep	LCD
HC16LC1 系列	80251	3801T412	标准GPIO型-0	HC16LC10J6UA-QFN48	QFN48	32K	Y	4K	34	N/A	Y	Y	Y	7ch	Y	1	N/A	4	N/A	1	1	1	2	N/A	1	N/A
	80251	3801T412		HC16LC10F6UA-QFN32	QFN32	32K	Y	4K	18	N/A	Y	Y	Y	3ch	Y	1	N/A	4	N/A	1	1	0	1	N/A	1	N/A
	80251	3801T412		HC16LC10J4UA-QFN48	QFN48	16K	Y	2K	34	N/A	Y	Y	N/A	7ch	Y	1	N/A	2	N/A	1	1	0	2	N/A	1	N/A
	80251	3801T412		HC16LC10F4UA-QFN32	QFN32	16K	Y	2K	18	N/A	Y	Y	N/A	3ch	Y	1	N/A	2	N/A	1	1	0	1	N/A	1	N/A
	80251	3801T412	LCD型-6	HC16LC16M6TA-LQFP80	LQFP80 12*12	32K	Y	4K	58	N/A	Y	Y	Y	7ch	Y	1	N/A	4	N/A	1	1	1	2	N/A	1	40*4
	80251	3801T412		HC16LC16M6TA-LQ80	LQFP80 10*10	32K	Y	4K	58	N/A	Y	Y	Y	7ch	Y	1	N/A	4	N/A	1	1	1	2	N/A	1	40*4
	80251	3801T412		HC16LC16K6TA-LQ64	LQFP64 7*7	32K	Y	4K	58	N/A	Y	Y	Y	7ch	Y	1	N/A	4	N/A	1	1	1	2	N/A	1	40*4
	80251	3801T412		HC16LC16K6TA-LQFP64	LQFP64 10*10	32K	Y	4K	42	N/A	Y	Y	Y	7ch	Y	1	N/A	4	N/A	1	1	1	2	N/A	1	32*4

5 管脚定义

5.1 HC16LC16M6TA (80 管脚)

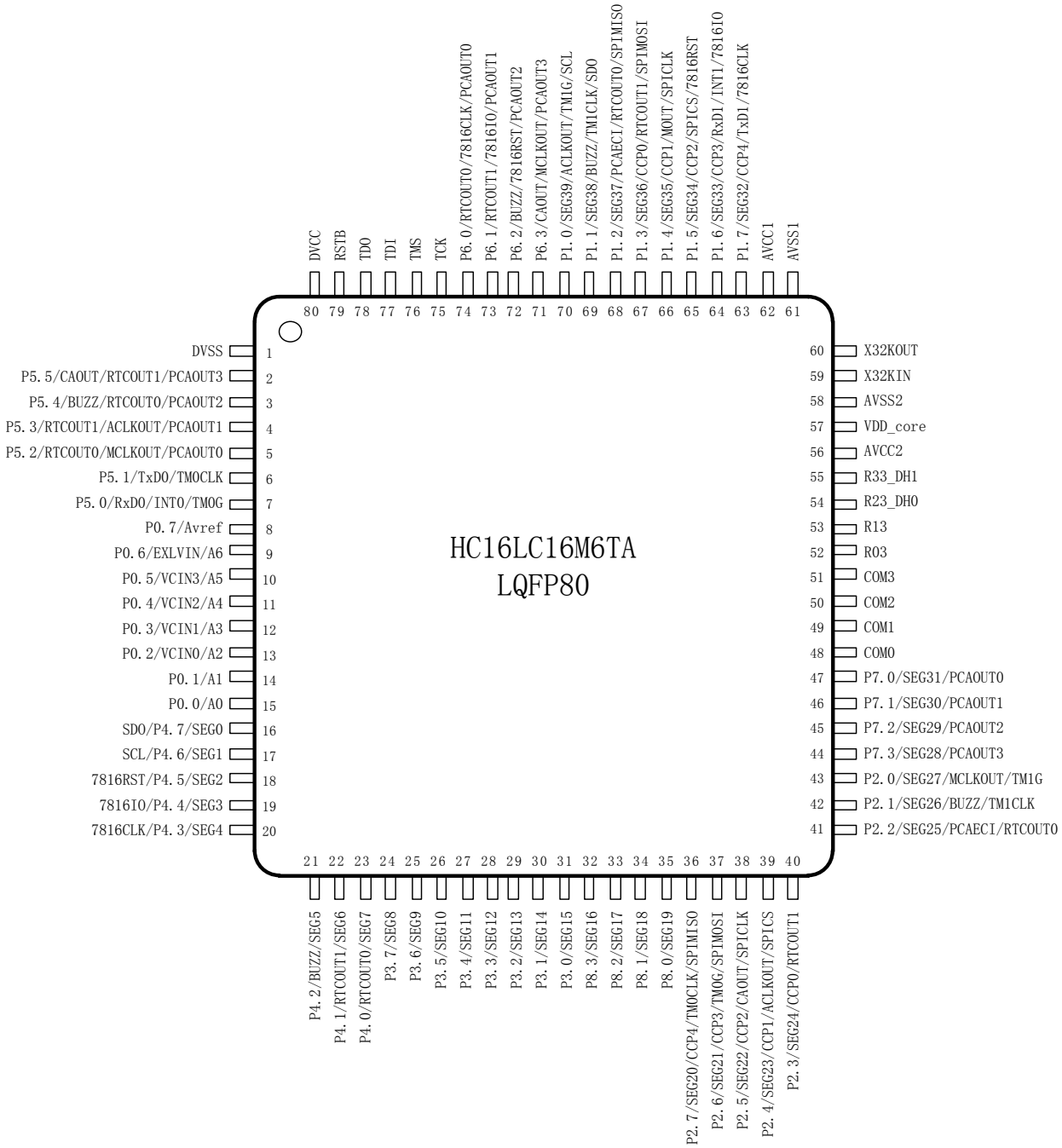


Figure 5-1 HC16LC16M6TA 80-pin 管脚定义

5.2 HC16LC16K6TA (64 管脚)

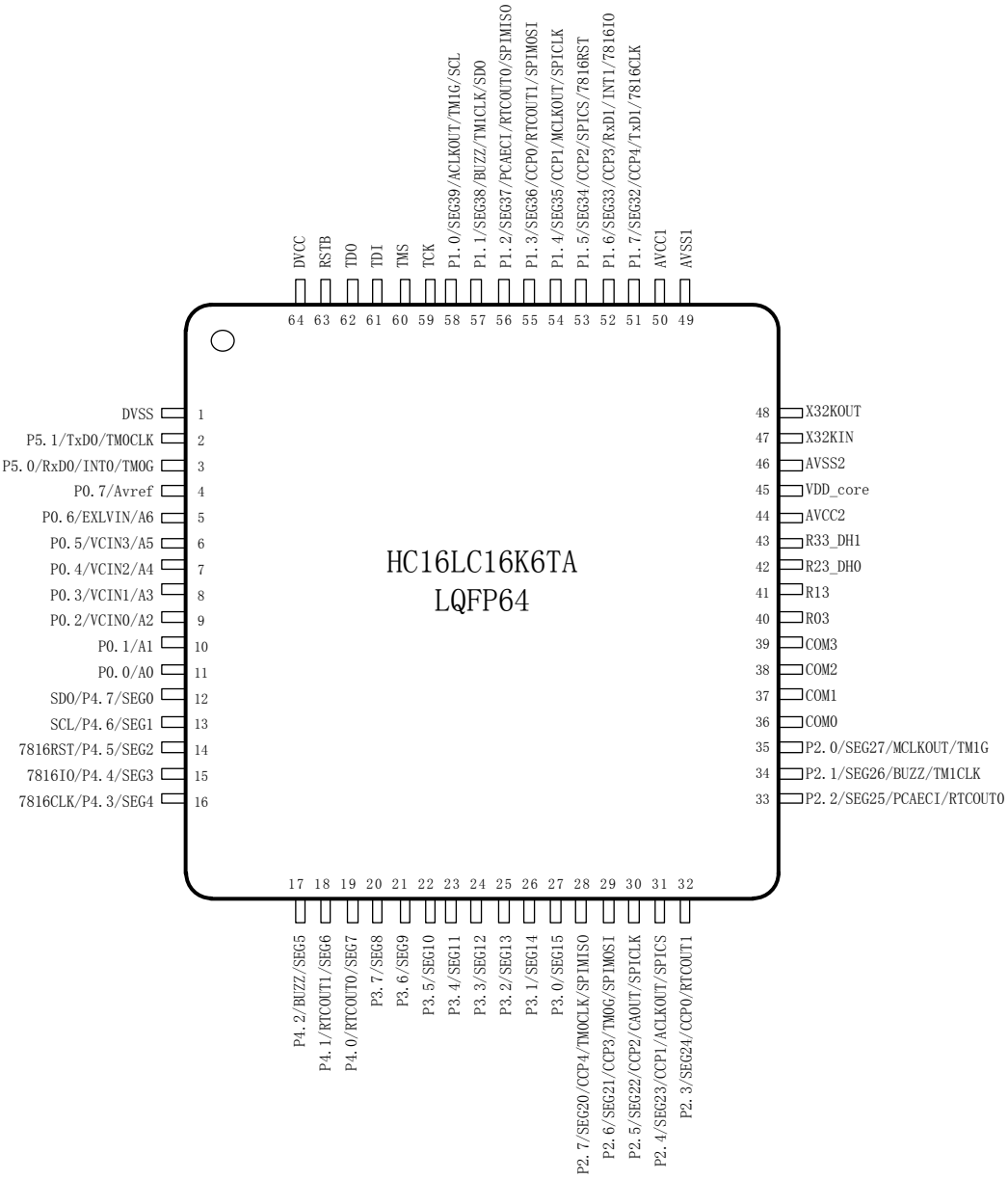


Figure 5-2 HC16LC10F6UA 64-pin 管脚定义

注意事项：当使用 64pin 芯片时，需要把未使用的 GPIO 通过端口寄存器设置为内部上拉模式或输出低电平模式，以防止漏电

5.3 HC16LC10J6UA/HC16LC10J4UA (48 管脚)

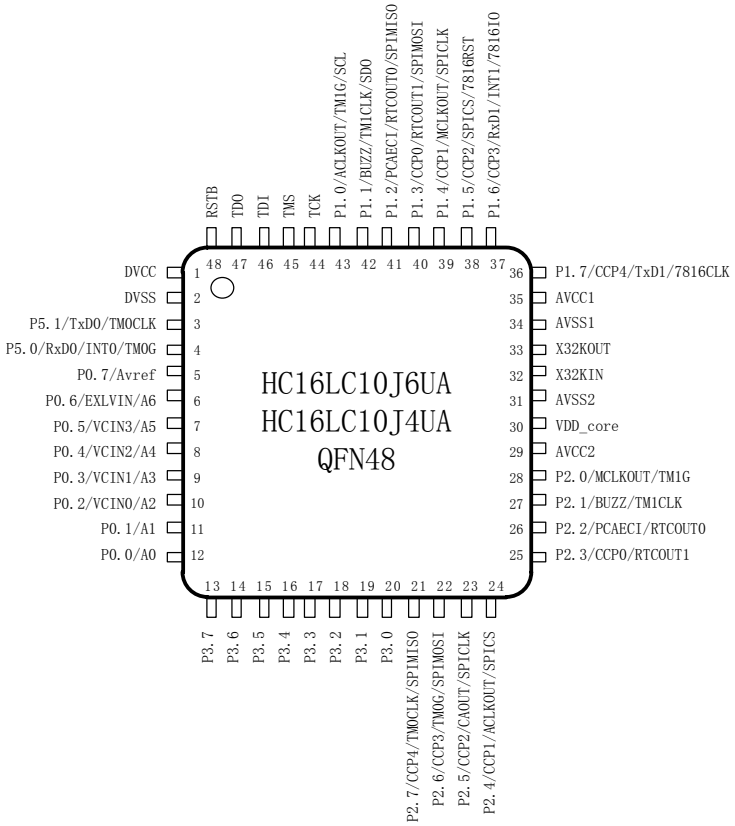


Figure 5-3 HC16LC10J6UA/HC16LC10J4UA 48-pin 管脚定义

注意事项：当使用 48pin 芯片时，需要把未使用的 GPIO 通过端口寄存器设置为内部上拉模式或输出低电平模式，以防止漏电

5.4 HC16LC10F6UA/HC16LC10F4UA (32 管脚)

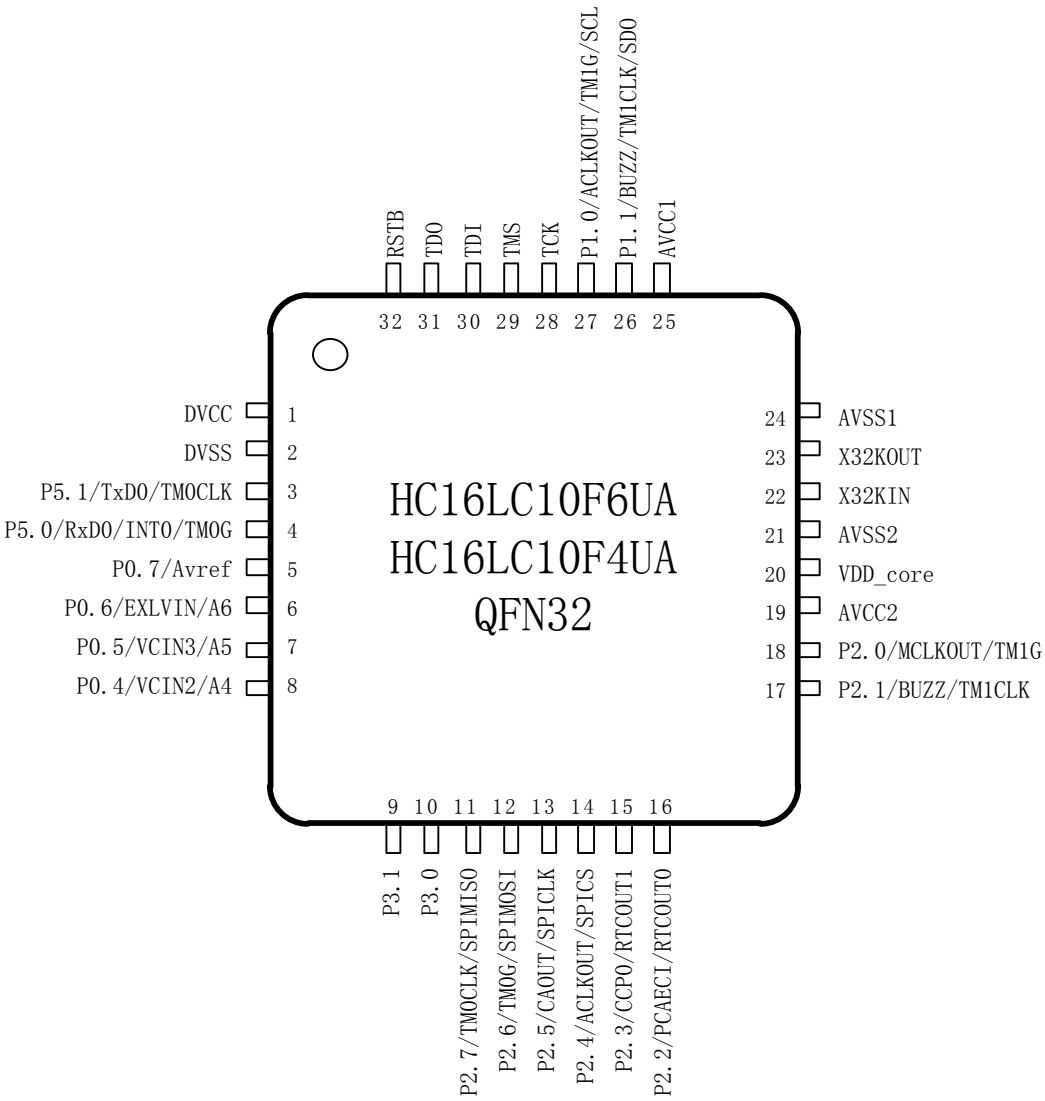


Figure 5-4 HC16LC10F6UA/HC16LC10F4UA 32-pin 管脚定义

注意事项：当使用 32pin 芯片时，需要把未使用的 GPIO 通过端口寄存器设置为内部上拉模式或输出低电平模式，以防止漏电

5.5 HC16LC16M6T/HC16LC16K6TA 管脚定义

管脚号	管脚号	管脚名称	管脚类型	描述
LQFP80	LQFP64			
1	1	DVSS	GND	数字电源接地端
2	/	P55	I/O	通用数字输入/输出
		CAOUT	O	电压比较器输出
		RTCOUT1	O	RTC脉冲输出1
		PCAOUT3	O	PCA3比较输出
3	/	P54	I/O	通用数字输入/输出
		BUZZ	O	Buzz输出频率
		RTCOUT0	O	RTC脉冲输出0
		PCAOUT2	O	PCA2比较输出
4	/	P53	I/O	通用数字输入/输出
		RTCOUT1	O	RTC 脉冲输出1
		ACLKOUT	O	外部32K时钟输出
		PCAOUT1	O	PCA1比较输出
5	/	P52	I/O	通用数字输入/输出引脚
		RTCOUT0	O	RTC脉冲输出0
		MOUT	O	主时钟分频输出
		PCAOUT0	O	PCA0比较输出
6	2	P51	I/O	通用数字输入/输出引脚
		TxD0	O	UART TxD输出

管脚号 LQFP80	管脚号 LQFP64	管脚名称	管脚类型	描述
		TM0CLK	I	定时器0时钟输入
7	3	P50	I/O	通用数字输入/输出引脚
		RxD0	I	UART RxD输入
		TM0G	I	定时器0门控输入
		INT0	I	快速端口中断0
8	4	P07	I/O	通用数字输入/输出引脚
		AVref	O	ADC参考电压
9	5	P06	I/O	通用数字输入/输出引脚
		EXLVIN	I	LVD外部输入电压监测引脚
		A6	I	ADC输入通道6
10	6	P05	I/O	通用数字输入/输出引脚
		VC_IN3	I	电压比较器的电压输入信号3
		A5	I	ADC输入通道5
11	7	P04	I/O	通用数字输入/输出引脚
		VC_IN2	I	电压比较器的电压输入信号2
		A4	I	ADC输入通道4
12	8	P03	I/O	通用数字输入/输出引脚
		VC_IN1	I	电压比较器的电压输入信号1
		A3	I	ADC输入通道3
13	9	P02	I/O	通用数字输入/输出引脚

管脚号 LQFP80	管脚号 LQFP64	管脚名称	管脚类型	描述
		VC_IN0	I	电压比较器的电压输入信号0
		A2	I	ADC输入通道2
14	10	P01	I/O	通用数字输入/输出引脚
		A1	I	ADC输入通道1
15	11	P00	I/O	通用数字输入/输出引脚
		A0	I	ADC输入通道0
16	12	SEG0	O	LCD segment0 输出
		P47	I/O	通用数字输入/输出引脚
		SDO	I/O	I^2C串行数据信号线
17	13	SEG1	O	LCD segment1 输出
		P46	I/O	通用数字输入/输出引脚
		SCL	I/O	I^2C串行时钟信号线
18	14	SEG2	O	LCD segment2 输出
		P45	I/O	通用数字输入/输出引脚
		7816RST	O	ISO7816 主机复位输出
19	15	SEG3	O	LCD segment3 输出
		P44	I/O	通用数字输入/输出引脚
		7816IO	I/O	ISO7816 主机数据信号线
20	16	SEG4	O	LCD segment4 输出
		P43	I/O	通用数字输入/输出引脚

管脚号 LQFP80	管脚号 LQFP64	管脚名称	管脚类型	描述
		7816CLK	O	ISO7816 主机时钟输出信号线
21	17	SEG5	O	LCD segment5 输出
		P42	I/O	通用数字输入/输出引脚
		BUZZ	O	Buzz 输出频率
22	18	SEG6	O	LCD segment6 输出
		P41	I/O	通用数字输入/输出引脚
		RTCOUT1	O	RTC脉冲输出1
23	19	SEG7	O	LCD segment7 输出
		P40	I/O	通用数字输入/输出引脚
		RTCOUT0	O	RTC脉冲输出0
24	20	P37	I/O	通用数字输入/输出引脚
		SEG8	O	LCD segment8 输出
25	21	P36	I/O	通用数字输入/输出引脚
		SEG9	O	LCD segment9 输出
26	22	P35	I/O	通用数字输入/输出引脚
		SEG10	O	LCD segment10 输出
27	23	P34	I/O	通用数字输入/输出引脚
		SEG11	O	LCD segment11 输出
28	24	P33	I/O	通用数字输入/输出引脚
		SEG12	O	LCD segment12 输出

管脚号 LQFP80	管脚号 LQFP64	管脚名称	管脚类型	描述
29	25	P32	I/O	通用数字输入/输出引脚
		SEG13	O	LCD segment13 输出
30	26	P31	I/O	通用数字输入/输出引脚
		SEG14	O	LCD segment14 输出
31	27	P30	I/O	通用数字输入/输出引脚
		SEG15	O	LCD segment15 输出
32	/	P83	I/O	通用数字输入/输出引脚
		SEG16	O	LCD segment16 输出
33	/	P82	I/O	通用数字输入/输出引脚
		SEG17	O	LCD segment17 输出
34	/	P81	I/O	通用数字输入/输出引脚
		SEG18	O	LCD segment18 输出
35	/	P80	I/O	通用数字输入/输出引脚
		SEG19	O	LCD segment19 输出
36	28	P27	I/O	通用数字输入/输出引脚
		SEG20	O	LCD segment20 输出
		CCP4	I/O	PCA4端口,PCACAPT4捕获输入, PACCOMP4比较输出
		TM0CLK	I	定时器0时钟输入
		SPIMISO	I/O	SPI主数据输入, 从数据输出
37	29	P26	I/O	通用数字输入/输出引脚

管脚号 LQFP80	管脚号 LQFP64	管脚名称	管脚类型	描述
		SEG21	O	LCD segment21 输出
		CCP3	I/O	PCA3端口，PACCAPT3捕获输入，PCACOMP3比较输出
		TM0G	I	定时器0门控输入
		SPIMOSI	I/O	SPI主数据输出，从数据输入
38	30	P25	I/O	通用数字输入/输出引脚
		SEG22	O	LCD segment22输出
		CCP2	I/O	PCA2端口，PCACAPT2捕获输入，PCACOMP2比较输出
		CAOUT	O	电压比较器输出
		SPICLK	I/O	SPI时钟从输入，主输出
39	31	P24	I/O	通用数字输入/输出引脚
		SEG23	O	LCD segment23 输出
		CCP1	I/O	PCA1端口，PCACAPT1捕获输入，PCACOMP1比较输出
		ACLKOUT	O	外部32K时钟输出
		SPICS	I/O	SPI的CS信号输出
40	32	P23	I/O	通用数字输入/输出引脚
		SEG24	O	LCD segment24 输出
		CCP0	I/O	PCA0端口，PCACAPT0捕获输入，PCACOMP0比较输出
		RTCOUT1	O	RTC脉冲输出1
41	33	P22	I/O	通用数字输入/输出引脚
		SEG25	O	LCD segment25 输出

管脚号 LQFP80	管脚号 LQFP64	管脚名称	管脚类型	描述
		RTCOUT0	O	RTC脉冲输出0
		PCAECI	I	PCA时钟输入
42	34	P21	I/O	通用数字输入/输出引脚
		SEG26	O	LCD segment 26 输出
		BUZZ	O	Buzz频率输出
		TM1CLK	I	定时器1的时钟输入
43	35	P20	I/O	通用数字输入/输出引脚
		SEG27	O	LCD segment 27 输出
		MOUT	O	主时钟分频输出
		TM1G	I	定时器1门控输入
44	/	P73	I/O	通用数字输入/输出引脚
		SEG28	O	LCD segment28 输出
		PCAOUT3	O	PCA3比较输出
45	/	P72	I/O	通用数字输入/输出引脚
		SEG29	O	LCD segment29 输出
		PCAOUT2	O	PCA2比较输出
46	/	P71	I/O	通用数字输入/输出引脚
		SEG30	O	LCD segment30 输出
		PCAOUT1	O	PCA1比较输出
47	/	P70	I/O	通用数字输入/输出引脚

管脚号 LQFP80	管脚号 LQFP64	管脚名称	管脚类型	描述
		SEG31	O	LCD segment31 输出
		PCAOUT0	O	PCA0比较输出
48	36	COM0	O	LCD Common0 输出，COM0—3 用于LCD底板驱动
49	37	COM1	O	LCD Common1 输出，COM0—3 用于LCD底板驱动
50	38	COM2	O	LCD Common2 输出，COM0—3 用于LCD底板驱动
51	39	COM3	O	LCD Common3 输出，COM0—3 用于LCD底板驱动
52	40	R03	I/O	LCD 偏置电压1
53	41	R13	I/O	LCD 偏置电压2
54	42	R23_DH0	I/O	LCD 偏置电压3
55	43	R33_DH1	I/O	LCD 驱动电压
56	44	AVCC2	Power	模拟电源2
57	45	VDD_core	Power	LDO内核供电输出 （仅限内部电路使用，连接1uF~4.7uF的电容到地）
58	46	AVSS2	AGND	模拟接地端
59	47	X32KIN	I	32KHz晶体振荡器的输入。可以连接标准时钟或石英晶体
60	48	X32KOUT	O	32KHz晶体振荡器的输出。可以连接标准时钟或石英晶体
61	49	AVSS1	AGND	模拟接地端
62	50	AVCC1	Power	模拟电源1
63	51	P17	I/O	通用数字输入/输出引脚
		SEG32	O	LCD segment 32 输出
		CCP4	I/O	PCA4端口,PCACAPT4捕获输入,PCACOMP4比较输出

管脚号 LQFP80	管脚号 LQFP64	管脚名称	管脚类型	描述
		TxD1	O	UART TxD1 输出
		7816CLK	O	ISO7816主机时钟输出
64	52	P16	I/O	通用数字输入/输出引脚
		SEG33	O	LCD segment 33 输出
		CCP3	I/O	PCA3端口,PCACAPT3捕获输入,PCACOMP3比较输出
		RxD1	I	UART RxD 输入
		INT1	I	快速端口中断1
		7816IO	I/O	ISO7816 主机数据信号线
65	53	P15	I/O	通用数字输入/输出引脚
		SEG34	O	LCD segment34 输出
		CCP2	I/O	PCA2端口,PCACAPT2捕获输入,PCACOMP2比较输出
		7816RST	O	ISO7816 主机复位输出
		SPICS	I/O	SPI CS 信号输出
66	54	P14	I/O	通用数字输入/输出引脚
		SEG35	O	LCD segment35 输出
		CCP1	I/O	PCA1端口,PCACAPT1捕获输入,PCACOMP1比较输出
		MOUT	O	主时钟分频输出
		SPICLK	I/O	SPI时钟从输入，主输出
67	55	P13	I/O	通用数字输入/输出引脚
		SEG36	O	LCD segment 36 输出

管脚号 LQFP80	管脚号 LQFP64	管脚名称	管脚类型	描述
		CCP0	I/O	PCA0端口，PCACAPT0捕获输入，PCACOMP0比较输出
		RTCOUT1	O	RTC脉冲输出1
		SPIMOSI	I/O	SPI主数据输出，从数据输入
68	56	P12	I/O	通用数字输入/输出引脚
		SEG37	O	LCD segment 37 输出
		RTCOUT0	O	RTC 脉冲输出0
		PCAECI	I	PCA时钟输入
		SPIMISO	I/O	SPI主数据输入，从数据输出
69	57	P11	I/O	通用数字输入/输出引脚
		SEG38	O	LCD segment 38 输出
		BUZZ	O	Buzz频率输出
		TM1CLK	O	定时器1的时钟输入
		SDO	I/O	I^2C串行数据信号线
70	58	P10	I/O	通用数字输入/输出引脚
		SEG39	O	LCD segment 39 输出
		ACLKOUT	O	外部32K时钟输出
		TM1G	I	定时器1门控输入
		SCL	I/O	I^2C串行时钟信号线
71	/	P63	I/O	通用数字输入/输出引脚
		CAOUT	O	电压比较器输出

管脚号 LQFP80	管脚号 LQFP64	管脚名称	管脚类型	描述
		PCAOUT3	O	PCA3比较输出
		MOUT	O	主时钟分频输出
72	/	P62	I/O	通用数字输入/输出引脚
		7816RST	O	ISO7816主机复位输出
		PCAOUT2	O	PCA2比较输出
		BUZZ	O	Buzz频率输出
73	/	P61	I/O	通用数字输入/输出引脚
		7816IO	I/O	ISO7816主机数据信号线
		PCAOUT1	O	PCA1比较输出
		RTCOUT1	O	RTC脉冲输出1
74	/	P60	I/O	通用数字输入/输出引脚
		7816CLK	O	ISO7816主机时钟输出
		PCAOUT0	O	PCA0比较输出
		RTCOUT0	O	RTC脉冲输出0
75	59	TCK	I	JTAG测试输入时钟
76	60	TMS	I	JTAG测试模式选择输入
77	61	TDI	I	JTAG测试数据输入
78	62	TDO	O	JTAG测试数据输出
79	63	RSTB	I	外部复位引脚，低电平有效，芯片复位
80	64	DVCC	Power	数字电源，提供给所有的数字部分。

5.6 HC16LC10J6UA/HC16LC10F6UA/HC16LC10J4UA/HC16LC10F4UA 管脚定义

管脚号 QFN48	管脚号 QFN32	管脚名称	管脚类型	描述
1	1	DVCC	Power	数字电源，提供给所有的数字部分
2	2	DVSS	GND	数字地线
3	3	P51	I/O	通用数字输入/输出引脚
		TxD0	O	UART TxD输出
		TM0CLK	I	定时器0时钟输入
4	4	P50	I/O	通用数字输入/输出引脚
		RxD0	I	UART RxD输入
		INT0	I	快速端口中断0
		TM0G	I	定时器0门控输入
5	5	P07	I/O	通用数字输入/输出引脚
		Avref	O	ADC参考电压
6	6	P06	I/O	通用数字输入/输出引脚
		EXLVIN	I	LVD外部输入电压监测引脚
		A6	I	ADC输入通道6
7	7	P05	I/O	通用数字输入/输出引脚
		VC_IN3	I	电压比较器的电压输入信号3
		A5	I	ADC输入通道5
8	8	P04	I/O	通用数字输入/输出引脚

管脚号 QFN48	管脚号 QFN32	管脚名称	管脚类型	描述
		VC_IN2	I	电压比较器的电压输入信号2
		A4	I	ADC输入通道4
9	/	P03	I/O	通用数字输入/输出引脚
		VC_IN1	I	电压比较器的电压输入信号1
		A3	I	ADC输入通道3
10	/	P02	I/O	通用数字输入/输出引脚
		VC_IN0	I	电压比较器的电压输入信号0
		A2	I	ADC输入通道2
11	/	P01	I/O	通用数字输入/输出引脚
		A1	I	ADC输入通道1
12	/	P00	I/O	通用数字输入/输出引脚
		A0	I	ADC输入通道0
13	/	P37	I/O	通用数字输入/输出引脚
14	/	P36	I/O	通用数字输入/输出引脚
15	/	P35	I/O	通用数字输入/输出引脚
16	/	P34	I/O	通用数字输入/输出引脚
17	/	P33	I/O	通用数字输入/输出引脚
18	/	P32	I/O	通用数字输入/输出引脚
19	9	P31	I/O	通用数字输入/输出引脚
20	10	P30	I/O	通用数字输入/输出引脚

管脚号 QFN48	管脚号 QFN32	管脚名称	管脚类型	描述
21	11	P27	I/O	通用数字输入/输出引脚
		CCP4	I/O	PCA4端口,PCACAPT4捕获输入, PCACOMP4比较输出
		TM0CLK	I	定时器0时钟输入
		SPIMISO	I/O	SPI主数据输入,从数据输出
22	12	P26	I/O	通用数字输入/输出引脚
		CCP3	I/O	PCA3端口,PCACAPT3捕获输入,PCACOMP3比较输出
		TM0G	I	定时器0门控输入
		SPIMOSI	I/O	SPI主数据输出,从数据输入
23	13	P25	I/O	通用数字输入/输出引脚
		CCP2	I/O	PCA2端口,PCACAPT2捕获输入,PCACOMP2比较输出
		CAOUT	O	电压比较器输出
		SPICLK	I/O	SPI时钟从输入, 主输出
24	14	P24	I/O	通用数字输入/输出引脚
		CCP1	I/O	PCA1端口,PCACAPT1捕获输入,PCACOMP1比较输出
		ACLKOUT	O	外部32K时钟输出
		SPICS	I/O	SPI的CS信号输出
25	15	P23	I/O	通用数字输入/输出引脚
		CCP0	I/O	PCA0端口,PCACAPT0捕获输入,PCACOMP0比较输出
		RTCOUT1	O	RTC脉冲输出1
26	16	P22	I/O	通用数字输入/输出引脚

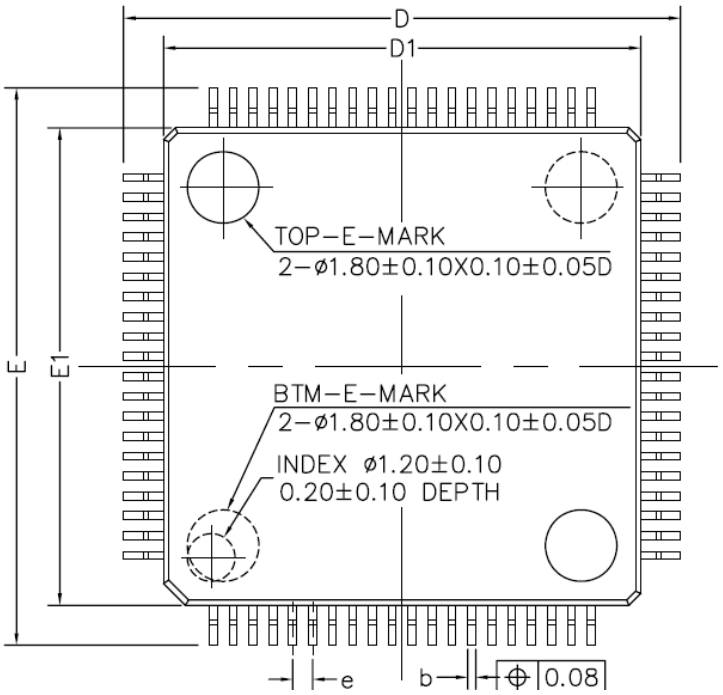
管脚号 QFN48	管脚号 QFN32	管脚名称	管脚类型	描述
		RTCOUT0	O	RTC脉冲输出0
		PCAECI	I	PCA时钟输入
27	17	P21	I/O	通用数字输入/输出引脚
		BUZZ	O	Buzz频率输出
		TM1CLK	I	定时器1的时钟输入
28	18	P20	I/O	通用数字输入/输出引脚
		MOUT	O	主时钟分频输出
		TM1G	I	定时器1门控输入
29	19	AVCC2	Power	模拟电源
30	20	VDD_core	Power	LDO内核供电输出 （仅限内部电路使用，连接1uF~4.7uF的电容到地）
31	21	AVSS2	AGND	模拟接地端
32	22	X32KIN	I	32KHz晶体振荡器的输入。可以连接标准时钟或石英晶体。
33	23	X32KOUT	O	32KHz晶体振荡器的输出。可以连接标准时钟或石英晶体。
34	24	AVSS1	AGND	模拟接地端
35	25	AVCC1	Power	模拟电源
36	/	P17	I/O	通用数字输入/输出引脚
		CCP4	I/O	PCA4端口,PCACAPT4捕获输入,PCACOMP4比较输出
		TxD1	O	UART TxD1 输出
		7816CLK	O	ISO7816 主机时钟输出信号线
37	/	P16	I/O	通用数字输入/输出引脚

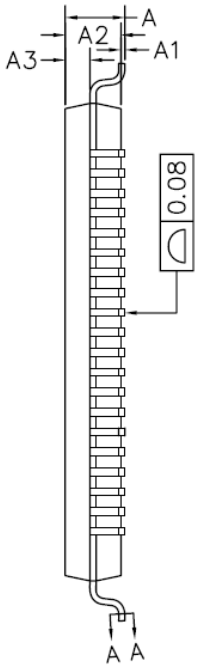
管脚号 QFN48	管脚号 QFN32	管脚名称	管脚类型	描述
		CCP3	I/O	PCA3端口,PCACAPT3捕获输入,PCACOMP3比较输出
		RxD1	I	UART RxD输入
		INT1	I	快速端口中断1
		7816IO	I/O	ISO7816主机数据信号线
38	/	P15	I/O	通用数字输入/输出引脚
		CCP2	I/O	PCA2端口,PCACAPT2捕获输入,PCACOMP2比较输出
		7816RST	O	ISO7816 主机复位输出
		SPICS	I/O	SPI CS信号输出
39	/	P14	I/O	通用数字输入/输出引脚
		CCP1	I/O	PCA1端口,PCACAPT1捕获输入,PCACOM1比较输出
		MOUT	O	主时钟分频输出
		SPICLK	I/O	SPI时钟从输入，主输出
40	/	P13	I/O	通用数字输入/输出引脚
		CCP0	I/O	PCA0端口,PCACAPT0捕获输入,PCACOMP0比较输出
		RTCOUT1	O	RTC脉冲输出1
		SPIMOSI	I/O	SPI主机数据输出，从机数据输入
41	/	P12	I/O	通用数字输入/输出引脚
		RTCOUT0	O	RTC脉冲输出0
		PCAECI	I	PCA时钟输入
		SPIMISO	I/O	SPI主机数据输入，从机数据输出

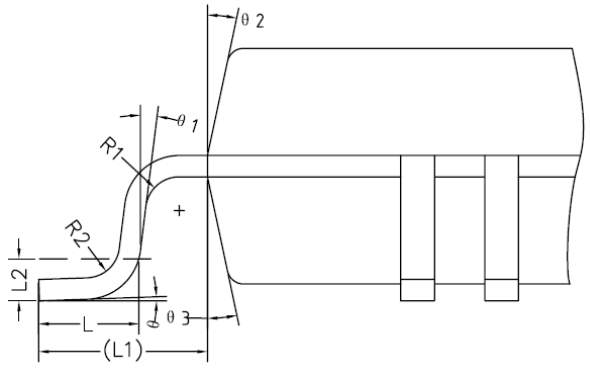
管脚号 QFN48	管脚号 QFN32	管脚名称	管脚类型	描述
42	26	P11	I/O	通用数字输入/输出引脚
		BUZZ	O	Buzz频率输出
		TM1CLK	I	定时器1的时钟输入
		SDO	I/O	I^2C串行数据信号线
43	27	P10	I/O	通用数字输入/输出引脚
		ACLKOUT	O	外部32K时钟输出
		TM1G	I	定时器1门控输入
		SCL	I/O	I^2C串行时钟信号线
44	28	TCK	I	JTAG测试输入时钟
45	29	TMS	I	JTAG测试模式选择输入
46	30	TDI	I	JTAG测试数据输入
47	31	TDO	O	JTAG测试数据输出
48	32	RSTB	I	外部复位引脚，低电平有效，芯片复位

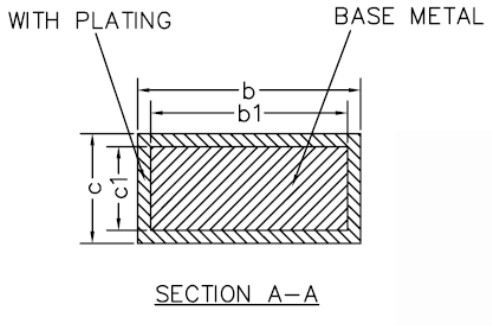
6 封装描述

6.1 LQFP80 12x12 封装







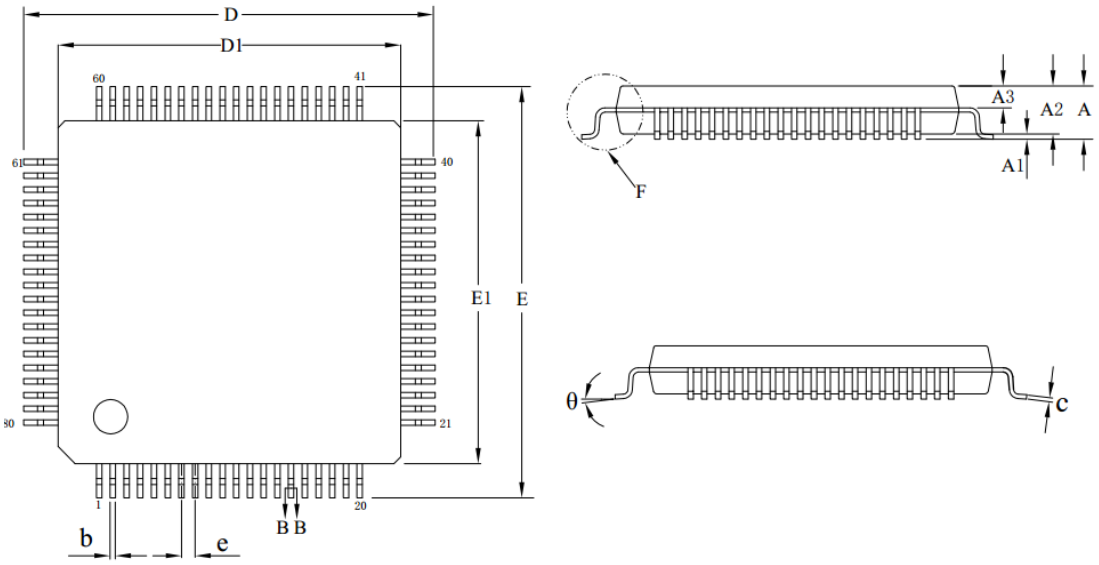


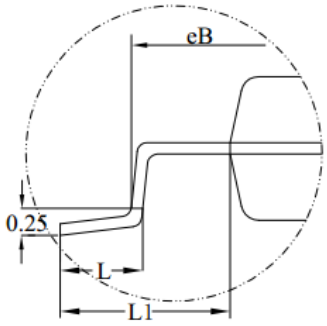
SECTION A-A

(UNITS OF MEASURE=MILLIMETER)

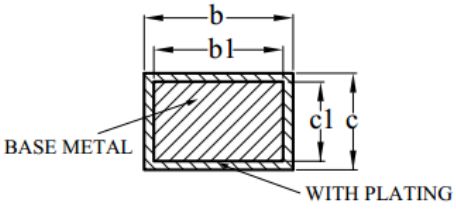
SYMBOL	MIN	NOM	MAX
A	–	–	1.60
A1	0.05	–	0.15
A2	1.35	1.40	1.45
A3	0.59	0.64	0.69
b	0.18	–	0.27
b1	0.17	0.20	0.23
c	0.13	–	0.18
c1	0.12	0.127	0.134
D	13.80	14.00	14.20
D1	11.90	12.00	12.10
E	13.80	14.00	14.20
E1	11.90	12.00	12.10
e	0.40	0.50	0.60
L	0.45	0.60	0.75
L1	1.00REF		
L2	0.25BSC		
R1	0.08	–	–
R2	0.08	–	0.20
θ	0°	3.5°	7°
θ 1	0°	–	–
θ 2	11°	12°	13°
θ 3	11°	12°	13°

6.2 LQFP80 10x10 封装





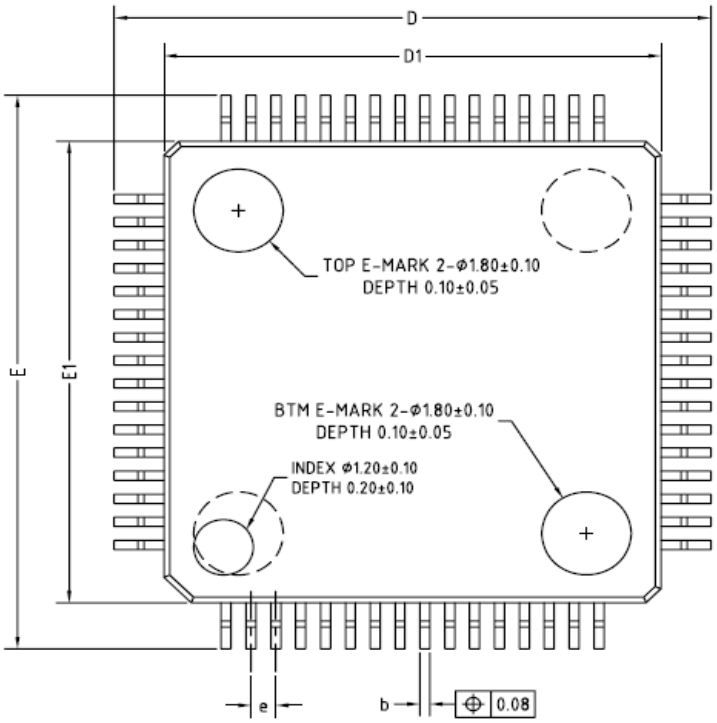
DETAIL: F

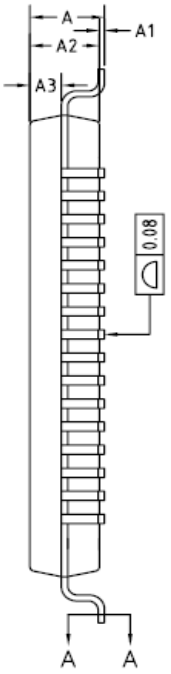


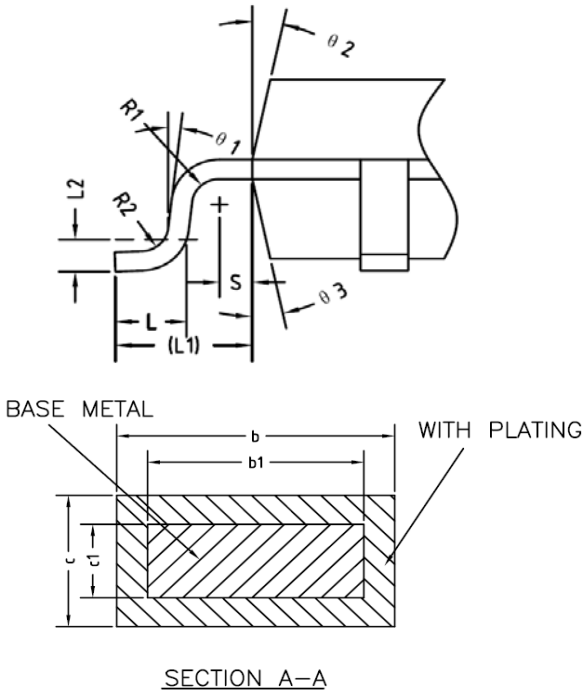
SECTION B-B

SYMBOL	MILLIMETER		
	MIN	NOM	MAX
A	—	—	1.60
A1	0.05	—	0.15
A2	1.35	1.40	1.45
A3	0.59	0.64	0.69
b	0.14	—	0.22
b1	0.13	0.16	0.19
c	0.13	—	0.17
c1	0.12	0.13	0.14
D	11.80	12.00	12.20
D1	9.90	10.00	10.10
E	11.80	12.00	12.20
E1	9.90	10.00	10.10
eB	11.25	—	11.45
e	0.40BSC		
L	0.45	—	0.75
L1	1.00REF		
θ	0	—	7°

6.3 LQFP64 10x10 封装





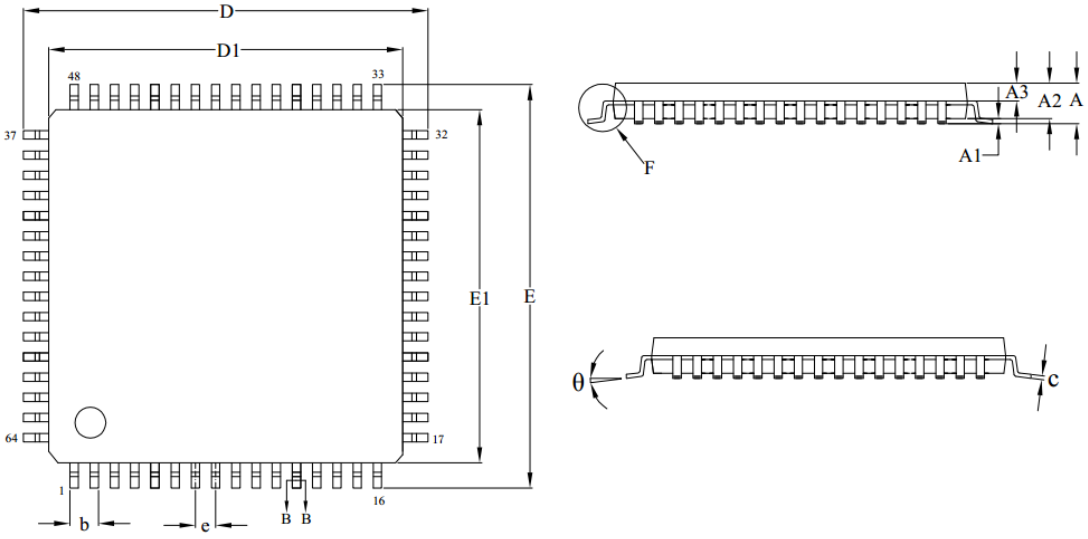


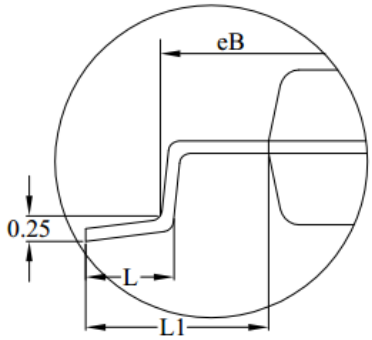
SECTION A–A

(UNITS OF MEASURE=MILLIMETER)

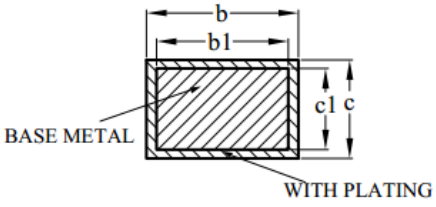
SYMBOL	MIN	NOM	MAX
A	–	–	1.60
A1	0.05	–	0.15
A2	1.35	1.40	1.45
A3	0.59	0.64	0.69
b	0.18	–	0.27
b1	0.17	0.20	0.23
c	0.13	–	0.18
c1	0.12	0.127	0.134
D	11.80	12.00	12.20
D1	9.90	10.00	10.10
E	11.80	12.00	12.20
E1	9.90	10.00	10.10
e		0.50BSC	
L	0.45	0.60	0.75
L1		1.00REF	
L2		0.25BSC	
R1	0.08	–	–
R2	0.08	–	0.20
S	0.20	–	–
θ	0°	3.5°	7°
θ1	0°	–	–
θ2	11°	12°	13°
θ3	11°	12°	13°

6.4 LQFP64 7x7 封装





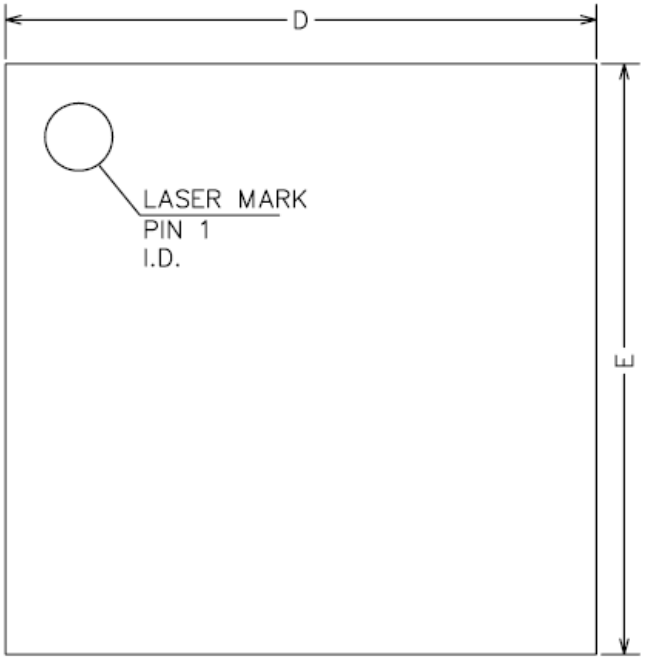
DETAIL: F



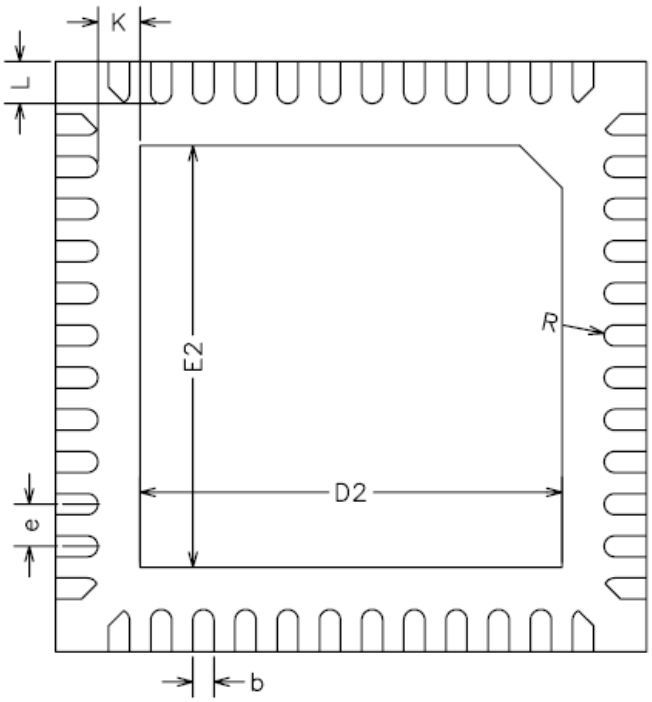
SECTION B-B

SYMBOL	MILLIMETER		
	MIN	NOM	MAX
A	—	—	1.60
A1	0.05	—	0.15
A2	1.35	1.40	1.45
A3	0.59	0.64	0.69
b	0.16	—	0.24
b1	0.15	0.18	0.21
c	0.13	—	0.17
c1	0.12	0.13	0.14
D	8.80	9.00	9.20
D1	6.90	7.00	7.10
E	8.80	9.00	9.20
E1	6.90	7.00	7.10
eB	8.10	—	8.25
e	0.40BSC		
L	0.40	—	0.65
L1	1.00REF		
θ	0	—	7°

6.5 QFN48 7x7 封装

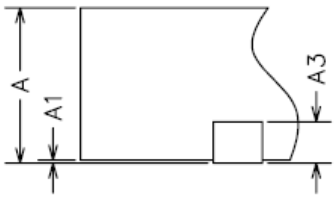


TOP VIEW



BOTTOM VIEW

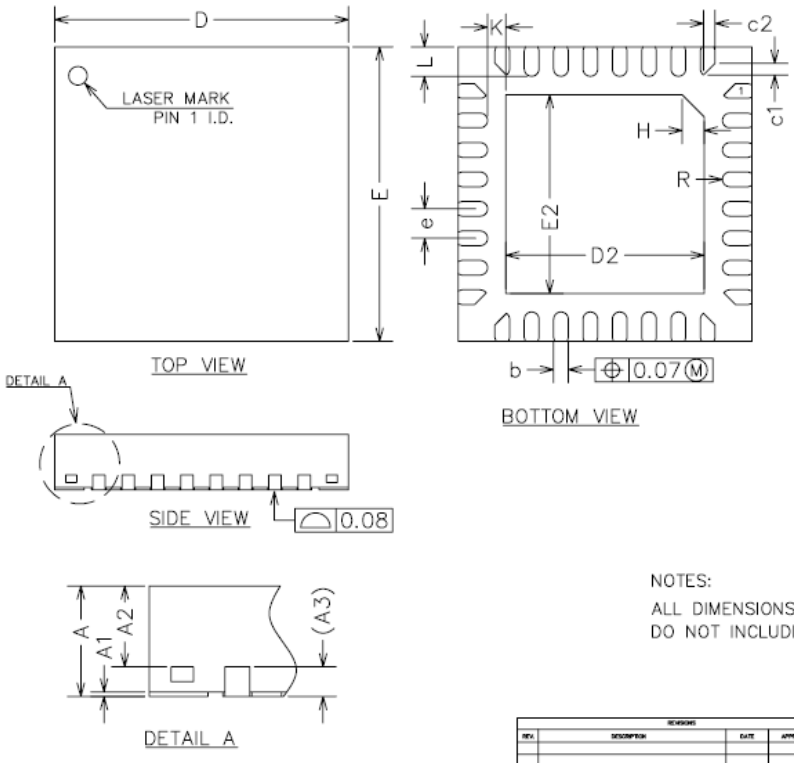
SIDE VIEW



COMMON DIMENSIONS
(UNITS OF MEASURE=MILLIMETER)

SYMBOL	MIN	NOM	MAX
A	0.70	0.75	0.80
A1	0	0.02	0.05
A3		0.20REF	
b	0.18	0.25	0.30
D	6.90	7.00	7.10
E	6.90	7.00	7.10
D2	4.50	4.65	4.80
E2	4.50	4.65	4.80
e	0.40	0.50	0.60
K	0.20	–	–
L	0.45	0.50	0.55
R	0.09	–	–

6.6 QFN32 4x4 封装



COMMON DIMENSIONS
(UNITS OF MEASURE=MILLIMETER)

SYMBOL	MIN	NOM	MAX
A	0.70	0.75	0.80
A1	0	0.02	0.05
A2	0.50	0.55	0.60
A3		0.20REF	
b	0.15	0.20	0.25
D	3.90	4.00	4.10
E	3.90	4.00	4.10
D2	2.60	2.70	2.80
E2	2.60	2.70	2.80
e	0.30	0.40	0.50
H		0.30REF	
K		0.25REF	
L	0.35	0.40	0.45
R	0.09	–	–
c1	–	0.16	–
c2	–	0.16	–

7 CPU 内核及内存

7.1 内核简介

- 嵌入式 80251 是标准 80C51 的加强版，具有 16 位和 32 位处理性能，与标准 80C51 指令完全兼容。

- HC16Lxx 采用哈佛结构，提供了独立的数据总线去访问程序存储器和数据存储器。

- 内核采用哈佛结构以及增强型流水线架构，在相同时钟频率下的处理能力为标准 8xC251 处理能力的三倍，并且比标准型 80C51 处理能力快 40 多倍。

- HC16Lxx 当编译 C 程序时，能够大幅减少代码容量大小。相比于 C51 的 C-compiler, 80251 的 C-compiler 可以把程序容量缩小 3 倍以上。

7.2 CPU 模块结构框图

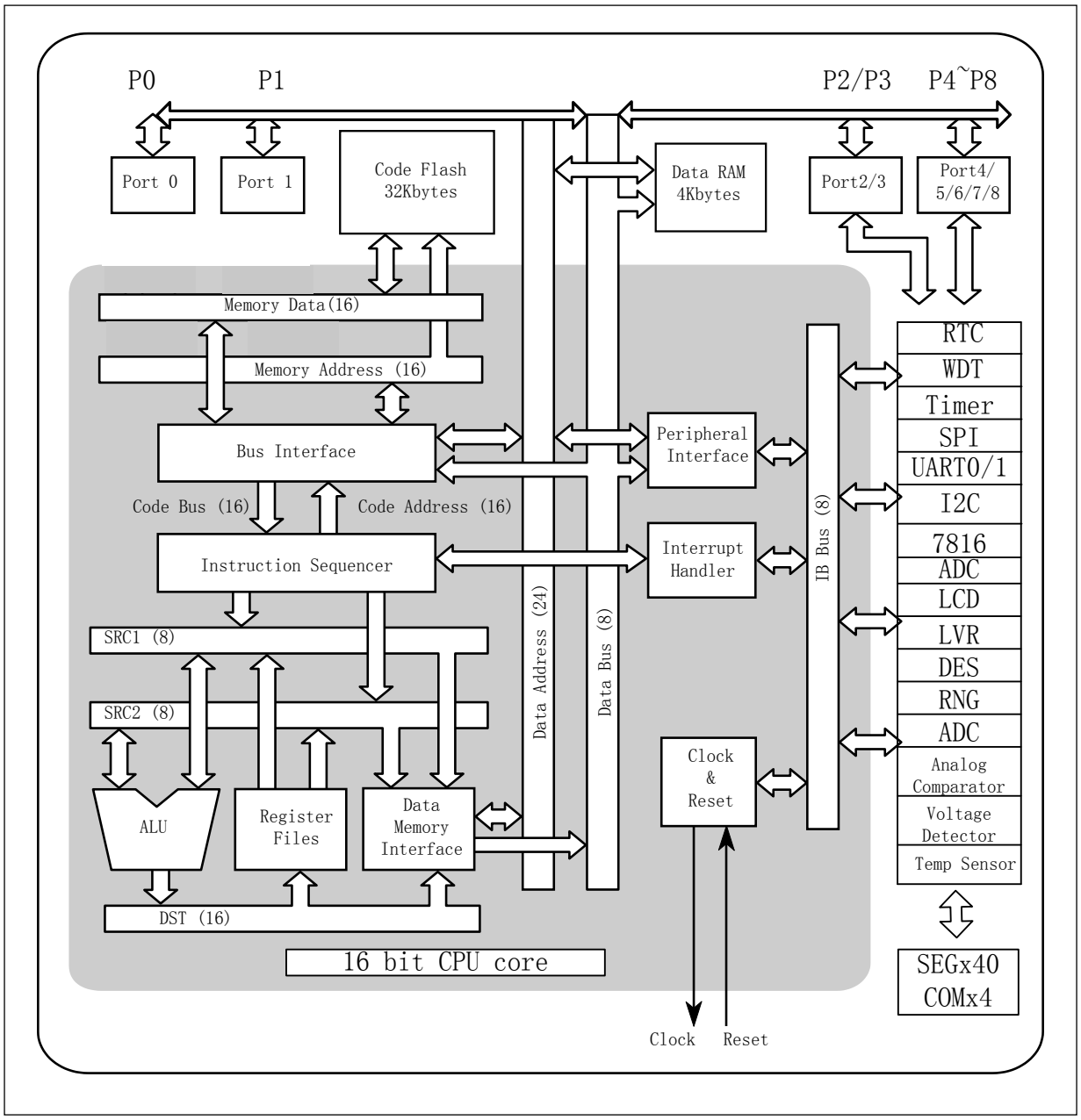


Figure 7-1 HC16Lxx CPU 模块

7.3 内存排列

HC16Lxx 有 3 块地址区间：

内存空间， 特殊功能寄存器（SFR）空间以及寄存器文件(Register File)空间。

7.4 内存空间架构

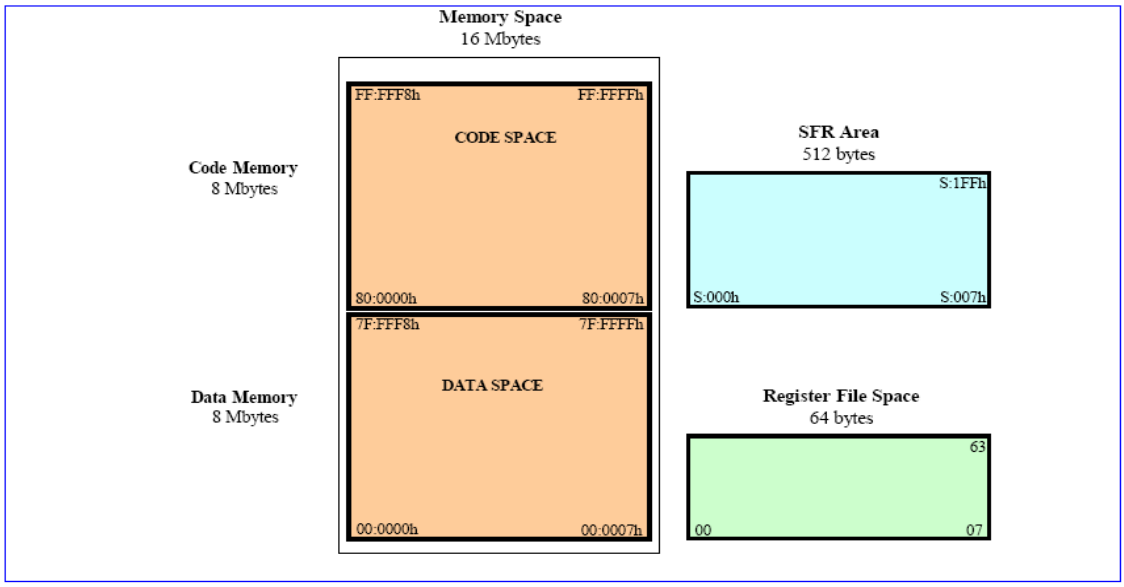


Figure 7-2 80251 内存空间分布图

在标准80251中，支持全部16M 字节的地址寻址。

[00:0000h－7F:FFFFh] 为8M字节的数据区， [80:0000h － FF:FFFFh] 为8M字节的程序区。

7.5 程序区

每次系统复位时，程序读取FF:0000h的指令。HC16Lxx一个周期可以取2字节指令（16位指令）。此外，HC16Lxx的程序接口提供写入能力，例如，将代码下载到代码存储器。下图为程序区读写指令：

程序存储器读取	程序存储器写入
数据传送指令	
MOV Rm, @DRk	MOV @DRk, Rm
MOV WRj, @DRk	MOV @DRk, WRj
MOV Rm, @DRk+dis24	MOV @DRk+dis24, Rm
MOV WRj, @DRk+dis24	MOV @DRk+dis24, WRj
MOVC A, @A+DPTR	
MOVC A, @A+PC	
算数运算	
ADD Rm, @DRk	
SUB Rm, @DRk	
CMP Rm, @DRk	
逻辑运算	
ANL Rm, @DRk	
ORL Rm, @DRk	
XRL Rm, @DRk	

Figure 7-3 80251 程序区读写指令

7.6 数据区

00:0000h - 01:FFFFh区域可以支持多种访问寻址模式，其它区域（02:0000h-7F:FFFFh)只支持24位的间接指令和24位替代指令（或通过改变DXPL的值获取MOVX指令）访问寻址模式。

通用数据存储器从00:0020h开始。

7.7 特殊功能寄存器（SFR）空间

SFR空间最多可容纳512个8位特殊功能寄存器，地址范围从S:000h到S:1FFh。8xC251架构中，前缀“S”和SFR地址一起使用来区分它们的内存空间地址(00:0000h - 00:01FFh)。

区域S:000h - S:07Fh和S:100h - S:1FFh未使用。

7.8 寄存器文件空间

寄存器文件都有独立的地址空间。位置0-7表示的4个寄存器组之一，每组有8个寄存器。在内存空间中，这32个字节需要占据位置00:0000h - 00:001Fh。寄存器文件8-63的不占用内存空间。

7.9 与 80C51 架构的兼容性

80C51架构的地址映射到8xC251架构的地址,所以80C51微控制器的代码可以在8xC251微控制器上运行。

内部数据区（地址00h-7Fh）可以直接或间接地进行寻址，内部数据区（地址80h-0FFh）只能间接寻址。代码存储器（64K字节）有一个独立的内存空间。只有用MOVC指令才能寻址在代码存储器中的数据。同样，64K字节的外部数据存储器可以只用MOVX指令访问。

寄存器文件（寄存器R0-R7）包含四个可切换寄存器组，每个寄存器组有八个寄存器。四组寄存器需要32字节，占用数据存储器00h-1Fh的位置。

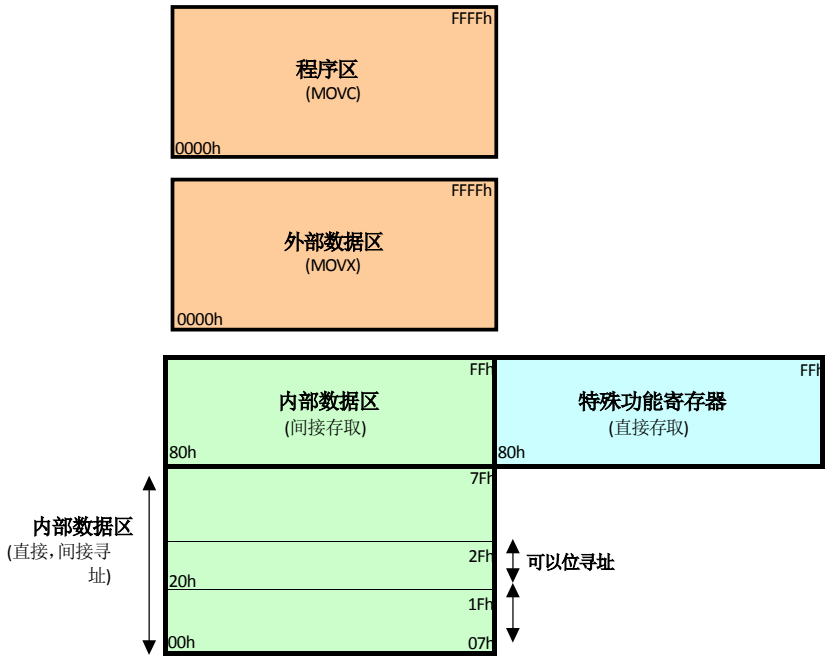


Figure 7-4 80C51地址空间

Figure 7-5 显示了在80C51架构的地址空间如何映射到8xC251架构的地址空间。80C51微控制器的64 K字节的程序存储器映射到8xC251微控制器的内存地址FF:，数据访问代码存储器（MOVC）将被重定向到这个区域。汇编程序还映射中断向量区域FF，这种映射是对用户透明的。

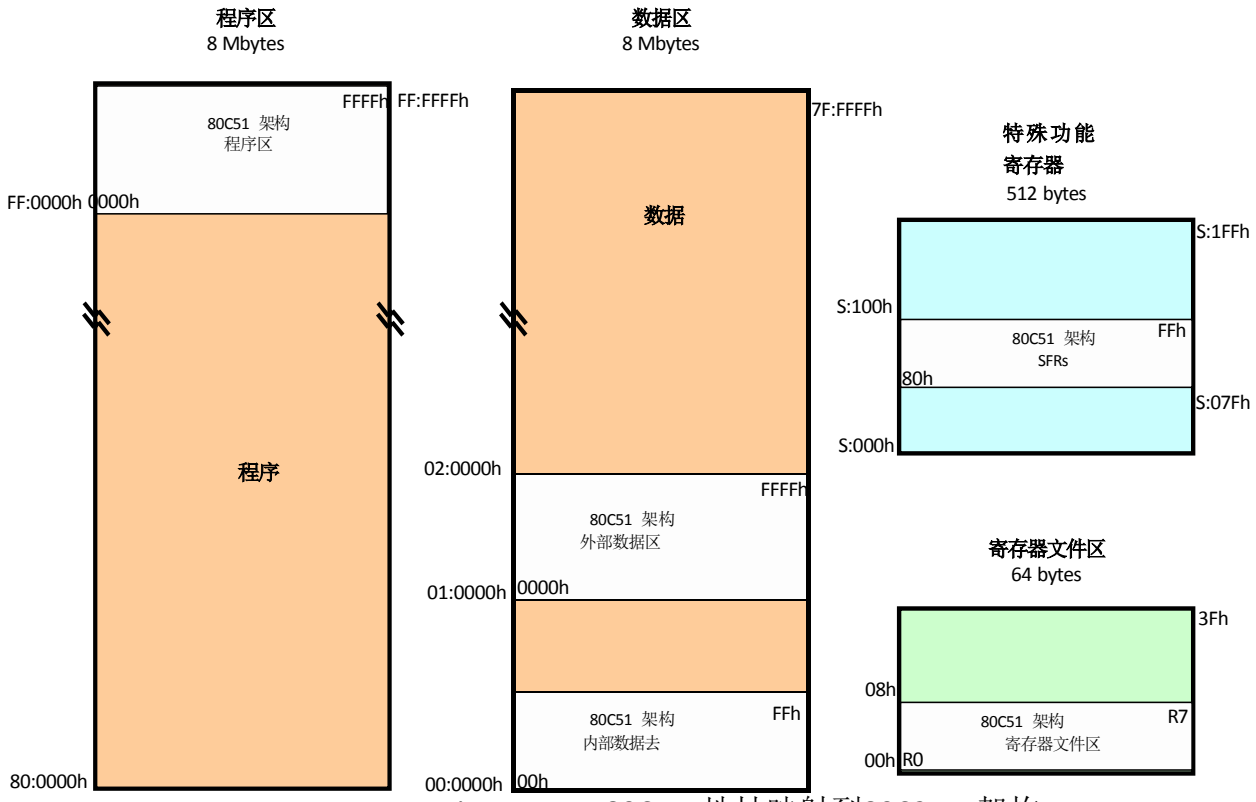


Figure 7-5 80C51 地址映射到80C251 架构

内存类型	80C51 架构			8xC251 架构
	大小	地址	数据寻址	地址
程序区	64K字节	[0000h-FFFFh]	用 MOVC 间 接 寻	[FF:0000h- FF:FFFFh]
外部数据区	64K字节	[0000h-FFFFh]	用MOVX间接寻址	[01:0000h-01:FFFFh]
内部数据区	128字节	[00h -7Fh]	直接/间接寻址	[00:0000h-00:007Fh]
	128字节	[80h -FFh]	间接寻址	[00:0080h-00:00FFh]
特殊功能寄存器	128字节	[S:80h -S:FFh]	直接寻址	[S:080h -S:0FFh]
寄存器文件	8 字节	[R0 -R7]	寄存器	[00:0000h-00:001Fh]

Figure 7-6 地址映射

80C51微控制器的64K字节的外部数据存储器，用于映射到所指定的存储器区域的数据指针DPX的16-23位，如DPXL。DPXL可以访问地址57的寄存器，同时也可在 S:084h作为特殊功能寄存器。DPXL的复位值是01h，映射外部存储器到区域01。可以通过给DPXL写不同的值来更改此映射。

80C51微控制器的256字节的片上数据存储器（00h－FFh）映射到地址 [00:0000h－00:00FFh]，以确保运行时的兼容性。在80C51架构，低128字节 （00h－7Fh），可直接和间接寻址，高128字节只能间接寻址访问。在8xC251架构中，所有区域00：通过直接、间接、变址寻址访问。

80C51微控制器的128字节的SFR空间映射到8xC251架构的512字节的SFR空间起始地址为S:080h。这可以直接寻址80C51微控制器的SFR（包括位寻址）。在新的架构下，SFR地址是不变的。

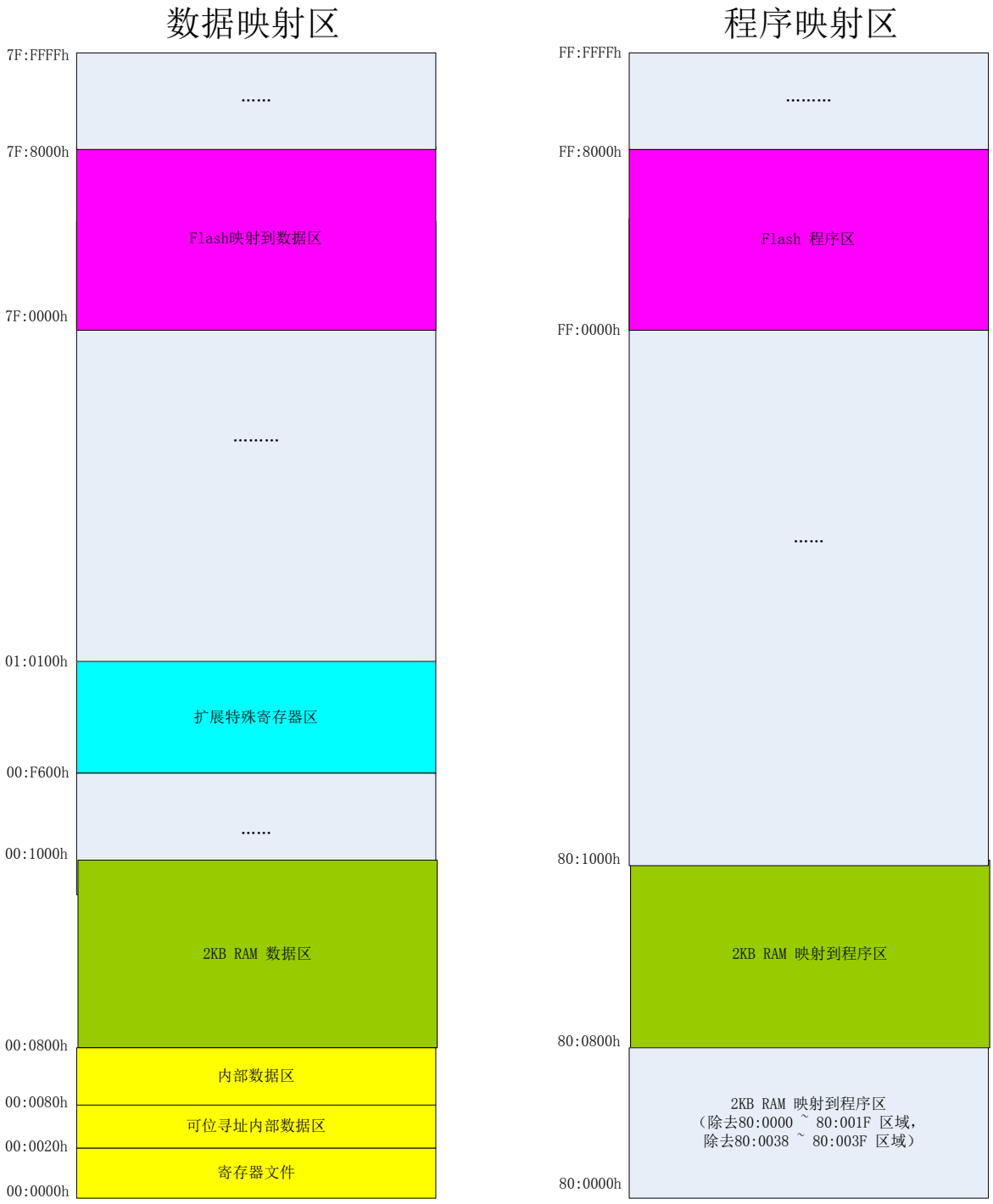


Figure 7-7 HC16Lxx 地址映射

7.10 寄存器文件组

HC16Lxx寄存器文件组由40个字节的位置：0-31和56-63组成。这些位置可以通过位，字节，字，双字进行访问。

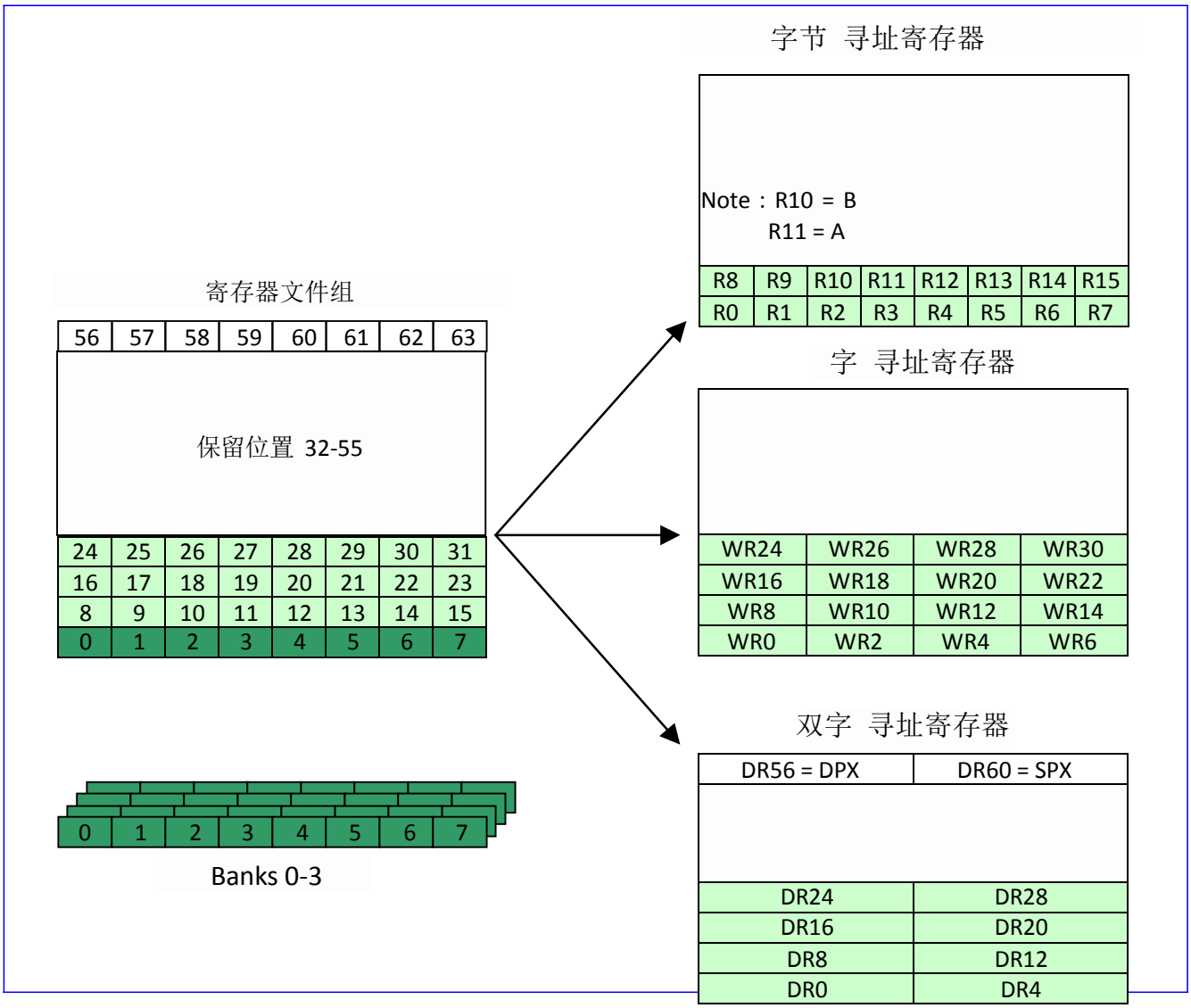


Figure 7-8 寄存器文件组

寄存器文件组的位置0-7实际由四个寄存器组组成，每一个寄存器组由八个寄存器组成。四组作为始终可以访问的内存地址空间[00:0000h-00：001Fh]（请参阅表7-11：寄存器组选择）。当一条指令使用R0到R7位时，在同一时间，只能使用一组寄存器。改变状态字PSW中的工作寄存器组选择位（RS1/RS0）可以改变当前工作寄存器组。在默认情况下（系统复位后）这些寄存器位于地址00h – 07h，即寄存器组0。间接寻址模式使用R0和R1作为索引寄存器。

- 寄存器文件组位置8-31和56-63可以通过寄存器文件地址去访问。可以像CPU中的寄存器一样访问这些寄存器文件组。
- 标准的8xC251中寄存器文件组的位置32-55是保留的，不能被访问。但在HC16Lxx中，这部分是内部数据区可以被使用。

在8xC251标准中，寄存器组实现了第一个32字节的片上随机存储器。在HC16Lxx中，寄存器组在内核中是以触发器实现的，以便加快使用这些寄存器中指令的执行时间。存储器访问的地址范围00:0000h – 00:001Fh 被重定向到触发器。外部数据存储器不能访问低于00:001Fh的地址。

寄存器地址	寄存器组	RS1	RS0	数据区地址范围
R0 - R7	Bank 0	0	0	[00:0000h - 00:0007h]
R0 - R7	Bank 1	0	1	[00:0008h - 00:000Fh]
R0 - R7	Bank 2	1	0	[00:0010h - 00:0017h]
R0 - R7	Bank 3	1	1	[00:0018h - 00:001Fh]

Figure 7-9 寄存器组选择

7.11 字节，字和双字寄存器

根据寄存器在寄存器文件组中的位置，可通过字节，字，或双字寻址，在图7-8的右侧所示。寄存器被命名为最低的字节位置的编号。比如：

- R4是由位置4的字节寄存器组成的。
- WR4是由寄存器4和5的字寄存器组成的。
- DR4是由寄存器4到7组成的双字寄存器。

位置R0-R15可以以字节，字或双字进行寻址。位置16-31只能以字或双字寻址。位置56-63仅能以双字寻址。寄存器只能按照 图 7-8 定义的名称进行寻址。

7.12 专用寄存器

HC16Lxx有四个专用寄存器：

- R10是B累加器
- R11是A累加器
- DR56是扩展数据指针，DPX
- DR60是扩展堆栈指针SPX

这些寄存器位于寄存器文件组中,R10，R11，和一些DR56中的字节和DR60也可作为SFR寻址。在寄存器文件组中的专用寄存器和它们相关的SFR在图 7-10中解释。

寄存器文件						SFRs	
名称			指令助记符	Reg.	位置	指令助记符	地址
扩展堆栈指针(SPX)	-		-	DR60	60	-	-
	-		-		61	-	-
	扩展堆栈指针，高		SPH		62	SPH	S:0BEh
	扩展堆栈指针，低		SP		63	SP	S:081h
扩展数据指针(DPX)	-		-	DR56	56	-	-
	扩展数据指针，低		DPXL		57	DPXL	S:084h
	数据指针DPTR	扩展数据指针，高	DPH		58	DPH	S:083h
		扩展数据指针，低	DPL		59	DPL	S:082h
8位累加器 (A 寄存器)			A	R11	11	ACC	S:0E0h
B 寄存器			B	R10	10	B	S:0F0h

Figure 7-10 专用寄存器，寄存器文件和相应的特殊功能寄存器

- A累加器和B累加器

8位累加器（ACC）是字节的寄存器R11，也可以在SFR空间中S：0E0h作为ACC进行寻址（图 7-11）。B累加器用于乘除，是R10，也可在SFR空间S:0F0h进行寻址。作为寄存器寻址ACC和B累加器，比寻址它们作为SFR快得多。在80C51架构中，使用累加器作为主寄存器进行数据移动和计算。而在8xC251架构中，任何的寄存器R0-R15可以进行数据移动和计算。PSW和PSW1寄存器的位反映了累加器的状态。对其他的寄存器，没有相同的状态指示。

- 扩展数据指针（DPX）

双字寄存器DR56是扩展的数据指针，DPX（Figure 7-11）。DPX低三个字节（DPL，DPH，DPXL）可作为SFR寻址。DPL和DPH组成的16位的数据指针DPTR。80C51架构使用DPTR作为数据指针，8xC251架构使用字或双字寄存器作为数据指针。DPXL，字节寻址位置为57，在80C51架构指定了内存区域（00:-FF:)映射到64K字节外部数据存储器空间。DPXL的复位值是01h。

- 扩展堆栈指针（SPX）

双字寄存器DR60是堆栈指针，SPX（Figure 7-11）。在80C51架构中，位置63是8位堆栈指针SP。位置62是堆栈指针的高，SPH。两个字节允许堆栈延伸到存储器区域的顶部00:。SP和SPH可作为SFR进行寻址。

指令PUSH和POP可直接寻址堆栈指针。子程序调用（ACALL，LCALL，紧急呼叫）和返回（ERET，RET，RETI）也可使用堆栈指针。为了保留堆栈，不要将DR60作为一个通用寄存器。

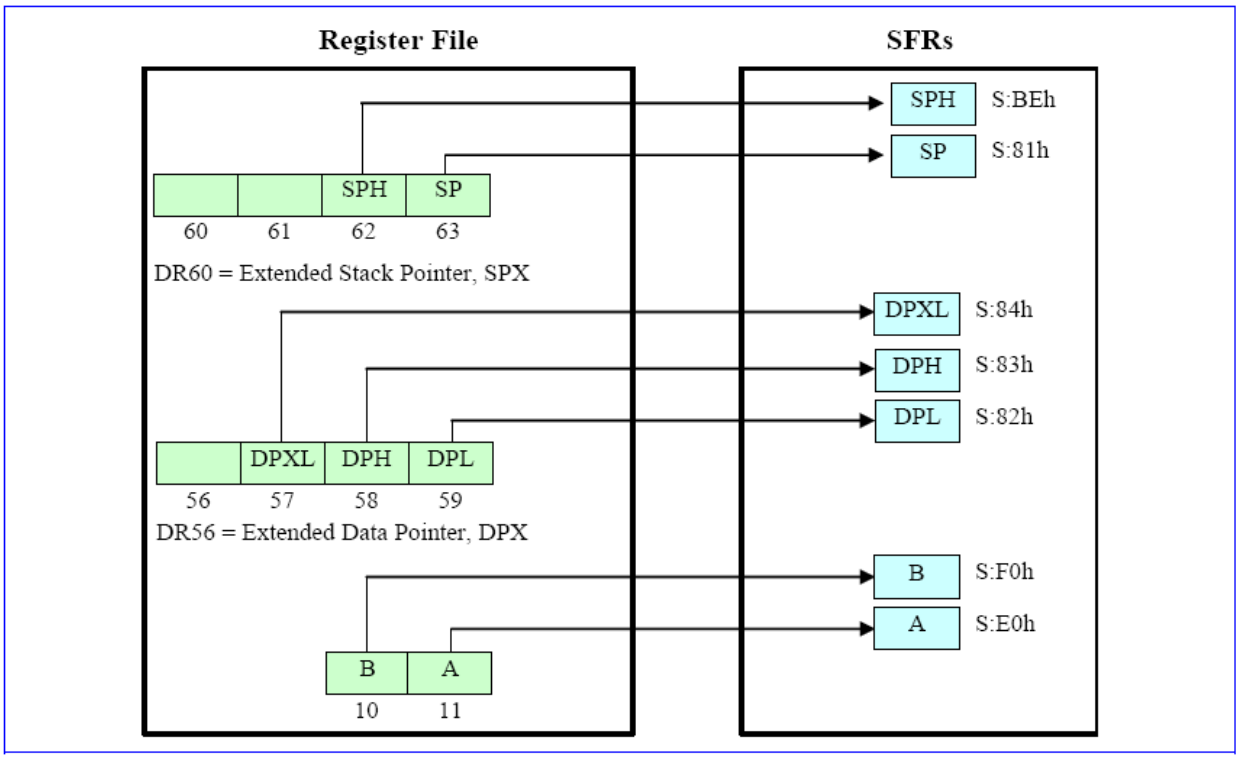


Figure 7-11 专用寄存器在寄存器文件组位置以及相对应的SFR地址

7.13 特殊功能寄存器（SFR）

特殊功能寄存器（SFR）定义相关外设以及CPU内核的操作。下表显示了含有SFR的助记符和复位值的地址空间。用SFR地址前面的S:来区分它们的内存空间中的地址。

SFR空间中的未占用的位置（Figure 7-12 中空白位置）即没有这些寄存器。如果指令试图写在这种位置，则该指令执行后什么都没写入。如果这种位置被读取，则返回一个不确定的值。

8xC251体系结构定义了512个SFR位置（S:000h - S:1FFh），但是Intel所定义的8xC251指令集规定只允许访问SFR地址S:080h - S:0FFh。即没有指令能够访问SFR的位置S:000h - S:07Fh和S:100h - S:1FFh。

F8	AIPL	CH	CCAP0H	CCAP1H	CCAP2H	CCAP3H	CCAP4H		FF
F0	B		P4_DIR	P5_DIR	P6_DIR	P7_DIR	P8DIR	AIPH	F7
E8	AIE	CL	CCAP0L	CCAP1L	CCAP2L	CCAP3L	CCAP4L		EF
E0	ACC		P4	P5	P6	P7	P8		E7
D8	CCON	CMOD	CCAPM0	CCAPM1	CCAPM2	CCAPM3	CCAPM4	CCAPO	DF
D0	PSW	PSW1	I2C_RUN	I2C_TM	I2CCON	I2CDAT	I2CADR	I2CSTA	D7
C8	U0_TMR	U0_TM	U1_TMR	U1_TM	VC3			X32K_CTL	CF
C0	AIF			TCON2	TL2	TH2	TL3	TH3	C7
B8	IPL0	SADEN	SADDR1	SADEN1	VC1	VC2	SPH	LVDC2	BF
B0	P3	Flash_PL		PERI_CLK0	PERI_CLK1	PERI_CLK2	LVDC1	IPH0	B7
A8	IE0	SADDR	SCON1	SBUF1	P0_DIR	P1_DIR	P2_DIR	P3_DIR	AF
A0	P2	MPAGE	Buzz_CTL	Buzz_CNT	Reset_flag	WDTCON	WDTRST		A7
98	SCON	SBUF	BGRC	ADC1	ADC2	ADC3	ADCH	ADCL	9F
90	P1	Flash_CTL	CLKC	CLKC1	CLKC2	LPM		MMCON	97
88	TCON	TMOD	TL0	TL1	TH0	TH1	CCMCON	CCMVAL	8F
80	P0	SP	DPL	DPH	DPXL			PCON	87

Figure 7-12 HC16Lxx 特殊寄存器

注意事项：
空白区域代表没有寄存器，为避免程序读到无效数值和浪费功耗，请不要对空白处进行读写操作。

指令助记符	地址	描述	复位值
ACC	S:0E0h	累加器	00h
B	S:0F0h	B累加器	00h
DPH	S:083h	数据指针高位	00h
DPL	S:082h	数据指针低位	00h
DPXL	S:084h	扩展数据指针低位	01h
MPAGE	S:0A1h	存储器分页寄存器	00h
PCON	S:087h	电源控制寄存器	00h
PSW	S:0D0h	状态寄存器	00h
PSW1	S:0D1h	状态寄存器1	00h
SP	S:081h	堆栈指针低位寄存器	07h
SPH	S:0BEh	堆栈指针高位寄存器	00h

Figure 7-13 内核特殊功能寄存器

指令助记符	地址	描述	复位值
AIE	S:0E8h	附加中断使能寄存器	00h
AIF	S:0C0h	附加中断标志寄存器	00h
AIPH	S:0F7h	附加中断优先级高位寄存器	00h
AIPL	S:0F8h	附加中断优先级低位寄存器	00h
IE0	S:0A8h	中断使能寄存器	00h
IPH0	S:0B7h	中断优先级高位寄存器	00h
IPL0	S:0B8h	中断优先级低位寄存器	00h

Figure 7-14 中断相关的特殊寄存器

指令助记符	地址	描述	复位值
P0	S:080h	端口P0寄存器	xxh
P0_DIR	S:0ACh	端口P0配置输入输出寄存器	FFh
P1	S:090h	端口P1寄存器	xxh
P1_DIR	S:0ADh	端口P1配置输入输出寄存器	FFh
P2	S:0A0h	端口P2寄存器	xxh
P2_DIR	S:0AEh	端口P2配置输入输出寄存器	FFh
P3	S:0B0h	端口P3寄存器	xxh
P3_DIR	S:0AFh	端口P3配置输入输出寄存器	FFh
P4	S:0E2h	端口P4寄存器	xxh
P4_DIR	S:0F2h	端口P4配置输入输出寄存器	FFh
P5	S:0E3h	端口P5寄存器	xxh
P5_DIR	S:0F3h	端口P5配置输入输出寄存器	3Fh
P6	S:0E4h	端口P6寄存器	xxh
P6_DIR	S:0F4h	端口P6配置输入输出寄存器	0Fh
P7	S:0E5h	端口P7寄存器	xxh
P7_DIR	S:0F5h	端口P7配置输入输出寄存器	0Fh
P8	S:0E6h	端口P8寄存器	xxh
P8_DIR	S:0F6h	端口P8配置输入输出寄存器	0Fh

Figure 7-15 端口相关的特殊寄存器

指令助记符	地址	描述	复位值
SADDR	S:0A9h	串口0从机独立地址寄存器	00h
SADEN	S:0B9h	串口0从机地址掩膜寄存器	00h
SBUF	S:099h	串口0数据缓冲寄存器	00h
SCON	S:098h	串口0控制寄存器	00h
SADDR1	S:0BAh	串口1从机独立地址寄存器	00h
SADEN1	S:0BBh	串口1从机地址掩膜寄存器	00h
SBUF1	S:0ABh	串口1数据缓冲寄存器	00h
SCON1	S:0AAh	串口1控制寄存器	00h
U0_TMR	S:0C8h	串口0计数器启动寄存器	00h
U0_TM	S:0C9h	串口0计数器自动重装载寄存器	00h
U1_TMR	S:0CAh	串口1计数器启动寄存器	00h
U1_TM	S:0CBh	串口1计数器自动重装载寄存器	00h

Figure 7-16 UART相关的特殊寄存器

指令助记符	地址	描述	复位值
I2C_TMR	S:0D2h	I^2C 定时器启动寄存器	00h
I2C_TM	S:0D3h	I^2C 定时器寄存器	00h
I2CCON	S:0D4h	I^2C 控制寄存器	00h
I2CDAT	S:0D5h	I^2C 数据寄存器	00h
I2CADR	S:0D6h	I^2C 地址寄存器	00h
I2CSTA	S:0D7h	I^2C 状态寄存器	F8h

Figure 7-17 I^2C相关的特殊寄存器

Memonic	地址	描述	复位值
TCON	S:088h	定时器0/1控制寄存器	00h
TH0	S:08Ch	定时器0高8位寄存器	00h
TH1	S:08Dh	定时器1高8位寄存器	00h
TL0	S:08Ah	定时器0低8位寄存器	00h
TL1	S:08Bh	定时器1低8位寄存器	00h
TMOD	S:089h	定时器模式选择寄存器	00h
TCON2	S:0C3h	定时器2/3控制寄存器	00h
TH2	S:0C5h	定时器2高8位寄存器	00h
TL2	S:0C4h	定时器2低8位寄存器	00h
TH3	S:0C7h	定时器3高8位寄存器	00h
TL3	S:0C6h	定时器3低8位寄存器	00h
WDTCON	S:0A5h	看门狗控制寄存器	07h
WDTRST	S:0A6h	看门狗复位寄存器	00h

Figure 7-18 定时器相关的特殊寄存器

指令助记符	地址	描述	复位值
CCAP0H	S:0FAh	PCA 模块0 比较/捕获模式高8位寄存器	00h
CCAP0L	S:0EAh	PCA 模块0 比较/捕获模式低8位寄存器	00h
CCAP1H	S:0FBh	PCA 模块1 比较/捕获模式高8位寄存器	00h
CCAP1L	S:0EBh	PCA 模块1 比较/捕获模式低8位寄存器	00h
CCAP2H	S:0FCh	PCA 模块2 比较/捕获模式高8位寄存器	00h
CCAP2L	S:0ECh	PCA 模块2 比较/捕获模式低8位寄存器	00h
CCAP3H	S:0FDh	PCA 模块3 比较/捕获模式高8位寄存器	00h
CCAP3L	S:0EDh	PCA 模块3 比较/捕获模式低8位寄存器	00h
CCAP4H	S:0FEh	PCA 模块4 比较/捕获模式高8位寄存器	00h
CCAP4L	S:0EEh	PCA 模块4 比较/捕获模式低8位寄存器	00h
CCAPM0	S:0DAh	PCA 模块0 比较/捕获模式控制寄存器	00h
CCAPM1	S:0DBh	PCA 模块1 比较/捕获模式控制寄存器	00h
CCAPM2	S:0DCh	PCA 模块2 比较/捕获模式控制寄存器	00h
CCAPM3	S:0DDh	PCA 模块3 比较/捕获模式控制寄存器	00h
CCAPM4	S:0DEh	PCA 模块4 比较/捕获模式控制寄存器	00h
CCAPO	S:0DFh	PCA 定时/计数器PWM模式和高速输出模式控制	00h
CCON	S:0D8h	PCA 计数/定时器控制寄存器	00h
CH	S:0F9h	PCA 计数/定时器高8位寄存器	00h
CL	S:0E9h	PCA 计数/定时器低8位寄存器	00h
CMOD	S:0D9h	PCA 计数/定时器模式寄存器	00h

Figure 7-19 可编程计数器阵列相关的特殊寄存器

指令助记符	地址	描述	复位值
CCMCON	S:08Eh	JTAG调试控制寄存器	00h
CCMVAL	S:08Fh	JTAG调试数据寄存器	00h
MMCON	S:097h	JTAG调试模式下周边模块控制寄存器	07h

Figure 7-20 调试相关的特殊寄存器

指令助记符	地址	描述	复位值
Flash_CTL	S: 091h	Flash 控制寄存器	00h
CLKC	S: 092h	时钟控制寄存器	03h
CLKC1	S: 093h	主时钟控制寄存器	60h
CLKC2	S: 094h	时钟稳定控制寄存器	57h
LPM	S: 095h	低电源模式控制寄存器	00h
BGRC	S: 09Ah	BGR控制寄存器	A0h
ADC1	S: 09Bh	ADC控制寄存器1	00h
ADC2	S: 09Ch	ADC控制寄存器2	00h
ADC3	S: 09Dh	ADC控制寄存器3	18h
ADCH	S: 09Eh	ADC采样结果高位寄存器	00h
ADCL	S: 09Fh	ADC采样结果低位寄存器	00h
Buzz_CTL	S: 0A2h	蜂鸣器控制寄存器	00h
Buzz_CNT	S: 0A3h	蜂鸣器计数控制寄存器	00h
Reset_Flag	S: 0A4h	复位信号标志寄存器	01h
Flash_PL	S:0B1h	Flash页加解锁控制器	00h
PERI_CLK0	S: 0B3h	周围功能模块时钟控制寄存器0	FFh
PERI_CLK1	S: 0B4h	周围功能模块时钟控制寄存器1	FFh
PERI_CLK2	S: 0B5h	周围功能模块时钟控制寄存器2	FFh
LVDC1	S: 0B6h	低电压检测寄存器1	00h
VC1	S: 0BCh	模拟电压比较器寄存器1	00h
VC2	S: 0BDh	模拟电压比较器寄存器2	00h
LVDC2	S: 0BFh	低电压检测寄存器2	00h
VC3	S: 0CCh	模拟电压比较器寄存器3	00h
X32K_CTL	S: 0CFh	X32K时钟控制寄存器	A0h

Figure 7-21 模拟相关的特殊寄存器

8 HC16Lxx 编程

本章介绍寻址方式和综述指令集。

指令集分为数据指令、位指令和控制指令。

8.1 源模式或二进制模式操作码

源模式（Source mode）和二进制模式(Binary mode)，是指8xC251架构分配操作码指令集的两种方式。通过HC16Lxx的选择配置，可以选定某种模式，从而产生更高效的代码。HC16Lxx架构有两种类型的指令：

- 源自80C51的指令
- 源自8xC251的特有指令

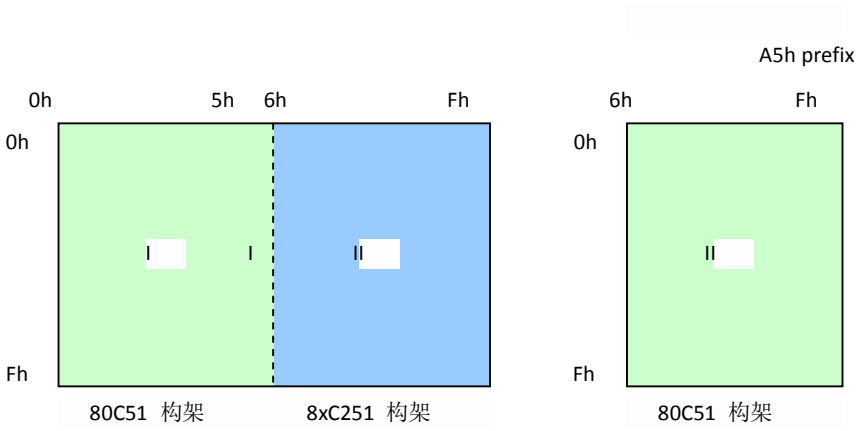


Figure 8-1 源模式操作码映射 (HC16Lxx 支持)

图 8-1 是源模式的操作码布局。区域II和区域III有开关区域，区域II的操作码要求脱离前缀A5h，在区域III（8xC251架构）指令操作码则不用脱离前缀。

为了说明二进制模式和源模式的操作码之间的差异，下表显示了三个样品指令操作码分配。

指令	操作码	
	二进制模式	源模式
INC A	08h	08h
ADD A, R4	2Ch	A5h 2Ch
ADD R4, R4	A5h 2Ch 44h	2Ch 44h

Figure 8-2 二进制模式和源模式操作码示例

二进制模式和源模式区别（HC16Lxx只支持源模式）：

已有80C51构架的代码，无需修改，选择二进制模式在HC16Lxx微控制器上运行，无需重新编译源代码。

一个含有前缀的操作码的指令，需要多个字节存储代码，并且额外的读取需要额外的字节，执行时间增加一个时钟周期。因此使用较少的前缀操作码将产生更高效的代码。

如果程序只使用80C51指令，二进制模式的代码更有效，因为它不使用前缀。另一方面，如果程序使用80C51指令中没有的新指令，源模式可能产生更高效的代码。

如果用C语言编写程序，通常源代码模式能产生更高效的代码（代码大小和速度）。

8.2 8xC251 架构编程特性

8xC251微控制器的指令集除与80C51微控制器的指令集保持兼容之外，还为用户提供了新的指令去开拓此架构的特性。许多新的指令针对8位，16位或32位进行操作（相对于8位和16位操作数，32位操作数的寻址模式较少），因此使用高级程序语言（如C语言）编程8xC251微控制器更方便、更高效。

指令集分成数据指令、位指令和控制指令。数据指令处理8位，16位和32位数据；位指令处理操作位，控制指令管理程序流程。

8.3 数据类型

下表列出了指令集需要用到的数据类型。可以在存储器的任何字节开始处存储字或双字；不强制要求两个字节或四个字节处边界对齐。字和双字以大端格式存储在存储器和寄存器文件中。

数据类型		位宽
位	Bit	1
字节	Byte	8
字	Word	16
双字	Dword (double word)	32

Figure 8-3 数据类型

HC16Lxx微控制器在内存和寄存器文件组中存储字（2字节）和双字（4字节）用大端格式。在内存存储区中，字或双字的最高字节（MSB）被存储在指令中最低地址，剩余的字节被存储在高地址，最低字节（LSB）存储在最高地址。字和双字可以以任何字节地址为开端存储在内存中。在寄存器文件组中，最高位MSB存储在指令指定的寄存器中的最低位字节。在Figure 8-4 中的代码示意了字和双字的大端格式存储。

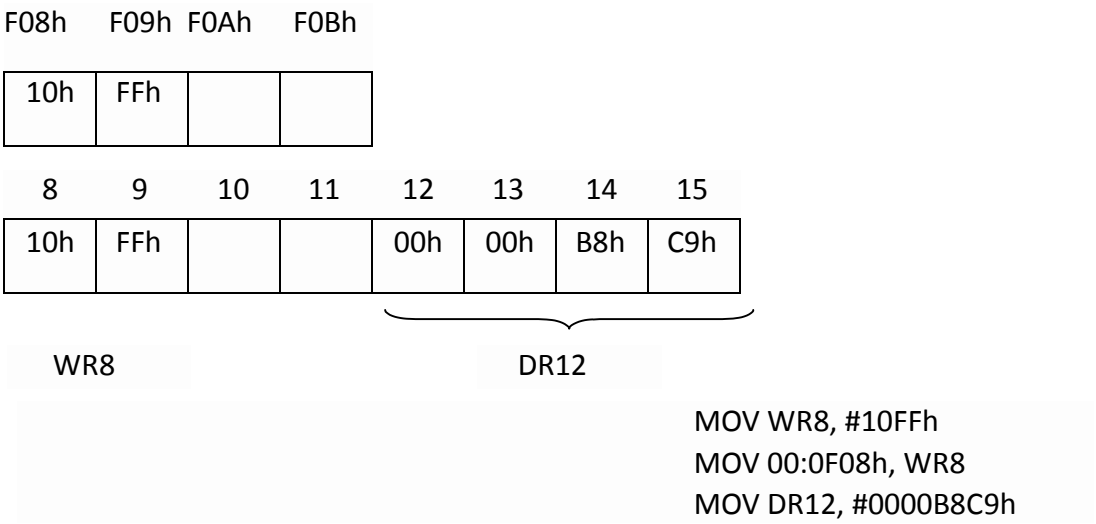


Figure 8-4 字和双字以大端模式存储

8.4 寄存器符号

在寄存器的寻址指令中，特定的指数表示该指令中的寄存器。例如，指令ADD A，Rn用“Rn”表示任何一个R0，R1，...，R7，n的范围是0-7。指令ADD Rm，data使用“Rm”去表示R0，R1，….,R15，m的范围就是从0到15。当一个指令包含相同类型的两个寄存器（比如MOV Rmd；Rms）第一个符号“d”表示“目标寄存器”，第二个符号“s”表示“源寄存器”。

寄存器类型	寄存器	目标寄存器	源寄存器	寄存器范围
字节	Ri	-		R0, R1
	Rn	-		R0-R7
	Rm	Rmd	Rms	R0-R15
字	WRj	WRjd	WRjs	WR0, WR2, WR4,..., WR30
双字	DRk	DRkd	DRks	DR0, DR4, DR8,..., DR28, DR56, DR60

Figure 8-5 字节寄存器，字寄存器和双字寄存器的符号

8.5 地址符号

在8xC251架构中，内存地址包括一个区域号码（00:，01:，...，FF:）。SFR地址有一个前缀“S:”（S:000h - S:1FFh）。有必要区分内存地址和SFR地址，因为内存位置00:0000h - 00:01FFh和SFR位置S:000h-S:1FFh都可以在一条指令中直接寻址。

在80C51架构使用80h - FFh作为内存和SFR地址，因为内存位置只能通过间接寻址，SFR可以直接寻址。为了实现兼容性，HC16Lxx控制器的软件工具能够识别用于80C51架构指令的符号，代码不需要改变。对8xC251架构中新的指令来说，需要内存区域的前缀（00:，01:，...，FF:)和SFR的前缀（S:）。同样，8xC251架构的软件工具允许寻址内存00h-FFh时使用00:，允许在80C51架构中的指令中使用前缀S:去寻址SFR。

8.6 寻址模式

8xC251架构支持以下寻址模式：

寻址模式	描述
寄存器寻址	指令指定的寄存器，它包含的操作数
立即数寻址	指令包含了操作数
直接寻址	直接寻址，指令包含了操作数地址
间接寻址	指令指定的操作数寄存器，它包含地址
索引寻址	该指令指定一寄存器和偏移量。操作数地址是基址和偏移量地址的和
相对寻址	指令包含从下一个指令到目标指领地址的偏移（比如，跳跃地址）
位寻址	该指令包含的位地址

Figure 8-6 HC16Lxx 寻址模式

8.7 状态寄存器（PSW）

状态寄存器（PSW）和状态寄存器1（PSW1）包含四种类型的位：

- CY，AC，OV，N，Z　- 表示操作结果的标志，由硬件设置
- P　- 累加器中的奇偶性。
- RS0和RS1　- 通过软件编程来为R0-R7来选择有效的寄存器组。
- F0和UD　- 通用标志。

PSW (S:D0h)								
状态寄存器								
位	7	6	5	4	3	2	1	0
符号	CY	AC	F0	RS1	RS0	OV	UD	P
复位值	0000 0000b							

位	符号	描述
7	CY	进位标志位 当执行加法指令(ADD, ADDC) MSB溢出时，或执行减法指令（SUB，SUBB）或比较指令(CMP)MSB有借位时，进位标志会被置“1”。 当循环移位和移位指令，逻辑位指令，位移指令，乘法指令，十进制调节指令，都可能置“1”。
6	AC	辅助进位标志位，又称为半进位标志位。 主要为了对于BCD码（二进制编码的十进制数）进行加减法时， 在加法时bit3有进位，减法时bit3有借位，辅助进位标志位置“1”，否则辅助进位标志为“0”。
5	F0	用户标志位0 通用标志位，允许用户自定义使用。
4	RS1	寄存器组选择位
3	RS0	
2	OV	溢出标志位 执行加减法运算时，如果发生运算溢出，则溢出标志被置1，否则被清零。在对于符号数进行加减运算后，软件可以通过检查溢出标志的状态得知计算结果是否在正确的范围之内。如果进行的是无符号数运算，溢出标志无效。但如果是有符号数加减法，出现大于+127或者小于-128的结果将使溢出标志为被置“1”。当执行一个乘法溢出一个字节或执行一个除数为“0”的除法也能使溢出标志为被置1。
1	UD	保留位 通用标志位，允许用户自定义使用。
0	P	奇偶校验位 每次累加器ACC被指令更改时，实际上每个机器周期之后奇偶校验位都会重新计算。 当ACC内“1”为奇数个时，奇偶校验位置1，当ACC内“1”为偶数个时，奇偶校验位置0。 原则是ACC和P中“1”的个数始终为偶数。

RS1	RS0	寄存器组	地址
0	0	0	00h-07h
0	1	1	08h-0FH
1	0	2	10h-17h
1	1	3	18h-1Fh

Figure 8-7 状态寄存器PSW

PSW1 (S:D1h)								
状态寄存器1								
位	7	6	5	4	3	2	1	0
符号	CY	AC	N	RS1	RS0	OV	Z	---
复位值	0000 0000b							

位	符号	描述
7	CY	进位标志位 同PSW进位标志位
6	AC	辅助进位标志位 同PSW辅助进位标志位
5	N	负数标志位 当逻辑或算法运算结果为负数时（比如 bit15=1），负数标志位置1。否则负数标志位为0。
4	RS1	寄存器组选择位
3	RS0	同PSW寄存器组选择位
2	OV	溢出标志位 同PSW溢出标志位
1	Z	零标志位 当逻辑或算法运算结果为“0”时，零标志位置1。否则清0
0	---	

Figure 8-8 状态寄存器PSW1

PSW和PSW1是读/写寄存器，但奇偶校验位不能软件写。单独的位可以通过位指令进行寻址。在条件跳转指令中隐含使用了PSW和PSW1位。PSW寄存器与80C51微控制器中的PSW寄存器是相同的。PSW1寄存器只存在于80251微控制器。在PSW1中的位CY，AC，RS0，RS1，和OV与PSW相应的位相同。

下表列出了影响CY，AC，OV，N和Z位的指令。

指令类型	指令	影响标志位[1]				
		CY	OV	AC [2]	N	Z
算数运算	ADD, ADDC, SUB, CMP	X	X	X	X	X
	INC, DEC				X	X
	MUL, DIV [3]	0	X		X	X
	DA	X			X	X
逻辑运算	ANL, ORL, XRL, CLR A, CPL A, RL, RR, SWAP				X	X
	RLC, RRC, SRL, SLL, SRA [4]	X			X	X
程序控制	CJNE	X			X	X
	DJNZ				X	X

Figure 8-9 在PSW和PSW1的标志上对指令的影响

1. X：标志受指令影响；0：标志被指令清除
2. AC：标志由8位操作数影响
3. 如果除数是零， OV标志被设置，其他位无意义。
4. SRL、SLL和SRA，最后一位移出的是存储在CY的位。

8.8 数据指令

数据指令由为8位、16位、和32位的算术指令、逻辑指令和数据传输组成。本节描述的数据的寻址模式和数据指令集。

8.9 数据寻址模式

寄存器R0-R7，WR0-WR6，DR0，和DR2是指当前由PSW和PSW1寄存器选择的寄存器组（见状态寄存器）。在所有组内的寄存器（活跃的和非活跃的）可以在内存（范围00h - 1Fh）中寻址。80C51架构的指令寻址外部存储器，是通过在字节DPXL中的扩展数据指针寄存器DPX（DR56）所指定的内存区域寻址。复位后，DPXL包含01h，将外部存储映射到区域01:。通过写DR56或DPXL SFR来指定不同的区域。

8.10 寄存器寻址

- 8xC251 架构

在寄存器寻址模式中，数据指令中的操作数（s）是在字节寄存器（R0 - R15），字寄存器(WR0，WR2，...，WR30)，或双字寄存器 (DR0，DR4，...DR28，DR56，DR60)中。

- 80C51 架构

指令寻址寄存器只有R0 - R7。

8.11 立即寻址

- 8xC251 架构

在立即寻址模式中，指令包含了数据操作数本身。字节操作使用8位立即数据（#data）；字操作使用16位的立即数据（＃data16）。双字操作数使用16位立即数据的低字，高位字为8个零（＃0data16表示），或其他高位字（＃1data16表示）。移动指令把16位立即数据写入一个双字寄存器（DRK），把数据要么放到高位字，低16位保持不变（MOVH），或使用符号扩展（MOVS）或零扩展（MOVZ）指令放到低16位。递增和递减指令包含立即数（＃short=1、2或4），其指定了增加或减少的量。

- 80C51 架构

指令仅仅使用8位立即数据（#data）。

8.12 直接寻址

- 8xC251 架构

在直接寻址模式中，指令包含了数据的操作数地址。片上RAM的8位直接寻址模式（dir8 = 00:0000h - 00:007Fh），字节和字都可以进行寻址，寻址SFR（dir8 = S:080h - S:0FFh）只能用字节。16位的直接模式可以在内存中寻址以字节或字（dir16 = 00:0000h - 00:FFFFh）。

- 80C51 架构

8位直接模式寻址片上RAM的128个字节（dir8 = 00h - 7Fh）只能通过字节，寻址SFR（dir8 = 80h - FFh）只能用字节。

寻址模式	操作数地址范围	汇编语言参考	注释
寄存器	00:0000h - 00:001Fh	R0 - R7 (由PSW选择的组)	
立即数	指令中的操作数	#data = #00h - #FFh	
直接	00:0000h - 00:007Fh	dir8 = 00h - 7Fh	内部RAM
	SFRs	dir8 = 80h - FFh 或 SFR 存储	SFR地址
间接	00:0000h - 00:00FFh	@R0, @R1	寻址内部RAM
	00:0000h - 7F:FFFFh	@DPTR, @R0, @R1	寻址外部数据存储器(MOVX)
	80:0000h - FF:FFFFh	@A+DPTR, @A+PC	寻址代码存储器(MOVC)

Figure 8-10 80C51架构中的数据指令的寻址模式

8.13 间接寻址

算术和逻辑指令使用间接寻址，源操作数通常是一个字节，目标寄存器是累加器或一个字节的寄存器（R0 – R15）。源地址是一个字节，字，或双字。两个架构通过不同的寄存器间接寻址：

- 8xC251 架构
  - 通过字和双字的寄存器间接寻址内存。
  - 字寄存器（@WRj，j=0，2，4，…，30）。
  - WRj的16位地址可以访问的位置为00:0000h – 00:FFFFh。
  - 双字寄存器(@DRk，k = 0，4，8，...，28，56和60)。

    24个低有效位LSB可以访问整个16兆字节的地址空间。在DRK上面8位必须为0。

    如果DR60作为一般数据指针，要注意DR60是扩展堆栈指针寄存器SPX。
- 80C51 架构
  - 指令使用间接寻址访问片上RAM，代码内存，和外部RAM。指令寻址外部存储器，由外部数据指针寄存器的DPX中的DPXL字节指定的内存区域。
  - 字节寄存器(@Ri，i = 0，1)。

    寄存器R0和R1间接寻址的片上存储器的00h – FFh的位置。当MOVX指令使用间接模式时，16位地址的最高有效位填充到MPAGE寄存器（S:0A1h）中。然后，它允许MOVX @ Ri指令访问64KB的外部数据存储器。
  - 16位的数据指针(@DPTR)。

    MOVX指令使用这些间接模式，使用扩展数据指针（DPX）访问外部数据RAM。
  - 16位数据指针(@A+DPTR)。

    MOVC指令使用这些间接模式去访问当前64K代码存储器。
  - 16位程序计数(@A+PC)。

    MOVC指令使用间接模式去访问当前64K代码存储器。

8.14 索引寻址

某些移动指令使用索引寻址，从源地址到目的地址移动字节或字。十六位索引寻址（@ WRJ+ dis16）间接访问内存的最低64K字节，基址可以是任何字寄存器WRj。该指令包含一个16位有符号偏移量加上基地址。用于计算的操作数地址只能用最低16位的总和。如果基址和一个正的偏移量的和超过FFFFh，则计算出的地址绕到区域00附近（例如F000h + 2005h变为1005h）。同样的，如果基址和一个负的偏移的总和小于零，其计算的地址就会绕到区域00上面附近（例如，2005h + F000h变为1005h）。24位索引寻址（@DRK + dis24）间接访问整个16M字节的地址空间。基址必须是DR0，DR4，DR24，DR28，DR56或DR60。该指令包含一个16位有符号偏移量与基址相加。

寻址模式	操作数地址范围	汇编语言参考	注释
寄存器寻址	00:0000h - 00:001Fh	R0 - R15， WR0 - WR30， DR0-DR28， DR56，DR60	R0-R7，WR0-WR6，DR0-DR4在由PSW和PSW1选择的寄存器组中
2位立即数寻址	(N.A) 指令操作数	#short = 1, 2 or 4	仅仅在增加和减少指令下使用
8位立即数寻址	(N.A) 指令操作数	#data8 = #00h - #FFh	
16位立即数寻址	(N.A) 指令操作数	#data16 = #0000h - #FFFFh	
直接寻址 8位地址空间	00:0000h - 00:007Fh	dir8 = 00:0000h - 00:0007Fh	内部 RAM
	SFRs	dir8 = S:080h - S:0FFh 或特殊功能寄存器（SFR）	SFR 地址
直接寻址 16位地址空间	00:0000h - 00:FFFFh	dir16 = 00:0000h - 00:FFFFh	
间接寻址 16位地址空间	00:0000h 00:FFFFh	@WR0 - @WR30	
间接寻址 24位地址空间	00:0000h - FF:FFFFh	@DR0 - @DR28, @DR56, @DR60	
索引寻址 16位地址空间	00:0000h - 0:FFFFh	@WRj+dis16 = @WR0+0h through @WR30+FFFFh	偏移量是有符号的，地址跳到00附近
索引寻址 24位地址空间	00:0000h - FF:FFFFh	@DRk+dis24= @DR0+0h through @DR28+FFFFh, @DR56+(0-FFFFh), @DR60+(0-FFFFh)	偏移量是有符号的

Figure 8-11 HC16Lxx架构数据指令的寻址模式

地址80:0000h以下是一个24位的读写数据存储器，通过间接寻址或索引寻址访问；地址80:0000h以上是一个24位的读写程序寄存器，通过间接或位移寻址访问。

如果以字的方式访问数据内存边界（7F:FFFFh），第二个字节将访问下一个地址（80:0000h），程序存储器控制信号将被激活，而地址不会跳到区域00:0000。

8.15 算术运算指令

算术运算指令集在8xC251架构中被扩展了。ADD和SUB指令以字节来操作和字被访问,有下面几种方式：

- 作为累加器的内容，一个字节的寄存器（Rn），或一个字寄存器（WRj）
- 在指令本身内（立即数据）
- 在内存中通过直接或间接寻址

ADDC和SUBB指令与80C51微控制器的指令相同。

CMP（比较）指令计算两个字节或字的差异，然后在PSW和PSW1寄存器写入标志CY，OV，AC，N和Z。这个差别并没有存储。操作数可以在多种模式下寻址。最频繁使用CMP是进行数据比较，或寻址前一个条件跳转指令。80C51微控制器的INC（增量）和DEC（递减）指令由指令补充，可以寻址字节，字，和双字寄存器，并且增加，或减少它们为1，2，或4（由＃short表示）。这些指令主要用于基于寄存器的地址指针和循环计数器。

8xC251架构为无符号的8位和16位数据提供了MUL（乘）和DIV（除）的指令。通过转换过程，有符号的乘和除留给用户去转换处理。执行以下操作：

- 8位乘法　　8位 × 8位 → 16位
- 16位乘法　　16位 × 16位 → 32位
- 8位除法　　8位 ÷ 8位 → 16位（8位商数，8位余数）
- 16位除法　　16位 ÷ 16位 → 32位（16位商，16位余数）

这些指令以一对字节寄存器（RMD，RMS），字寄存器（WRjd，WRjs），或累加器和B寄存器（A，B）工作。

对于8位的乘法，结果是存储在包含第一个操作数的字寄存器中。例如，MUL R2，R9的乘法结果被存储在WR2中。对于16位的乘法，结果被存储在双字寄存器，它包含第一个操作数寄存器。例如，指令MUL WR7, WR16 的乘法结果被存储在DR4中。

对于8位除法，操作数是字节寄存器。结果被存储在字寄存器，它包含第一个操作数寄存器。商存储在低字节，余数被存储在高字节。16位的除法，第一个操作数是一个字寄存器，结果存储在双字寄存器。如果第二个操作数（除数）是零，溢出标志位（OV）被设置，PSW和PSW1中的其他位是毫无意义的。

8.16 逻辑指令

8xC251的架构提供了一组执行逻辑操作的指令集。指令ANL，ORL，XRL（逻辑与，逻辑或，逻辑异或）用字，字节通过几种寻址模式寻址。一个字节寄存器，字寄存器，或累加器在逻辑上可以与一个寄存器连接，直接或间接寻址立即数据或数据，这些指令影响Z和N的标志。

除了CLR（清除），CPL（补），交换（Swap），和四个旋转的累加器的操作指令，8xC251微控制器有三个转移命令用于字节和字寄存器：

- SLL（逻辑左移）向左移动寄存器1位，并用0取代LSB
- SRL（逻辑右移）向右移动寄存器1位，并用0取代MSB
- SRA（右移运算）向右移动寄存器1位，MSB不变

8.17 数据传送指令

数据传送指令将数据从一个寄存器或内存位置复制到另一个位置。这些指令包括移动和交换，推送和取指令。

MOV（移动）是最常用的指令，8xC251架构扩大了其寻址模式。MOV可以在任何两个寄存器，寄存器和地址空间中的任何位置之间传输一个字节，字或双字。

MOVX（移动到外部）指令将一个字节从外部存储器移动到累加器或从累加器移动到内存。外部存储器所在的区域由DPXL指定，其复位值是01h。

MOVC（移动代码）指令将一个字节从代码存储移动到累加器。

MOVS(符号扩展移动）和MOVZ（零扩展移动）移动一个8位寄存器里的内容到一个16位寄存器的低字节。高字节填充的符号位（MOVS）或零（MOVZ）。

MOVH（移动到高位字）指令将16位立即数据转换成一个双字寄存器的高位字。

XCH（交换）指令将累加器中的内容与寄存器或内存位置的内容进行交换。

XCHD（交换数据）指令将累加器中的低半字节与片上RAM低半字节进行交换。 XCHD对BCD（二进制编码十进制）操作是有用的。

PUSH指令将一个字节，字或双字推到堆栈上，使用即时，直接，寄存器寻址模式。

POP指令从堆栈中取出一个字节或字到寄存器或内存。

8.18 位指令

一个位指令在内存或SFR上寻址一个特定的位。

有四类位指令：

- SETB（设置位），CLR（清除位），CPL（补位）

  这些指令可以设置，清除或补充地址位。

- ANL（与逻辑），ANL/（与逻辑补），ORL（或逻辑），ORL/（或逻辑补）

  这些指令允许对任何地址位“与”和“或”，或者用CY标志补充。

- MOV（移动）指令

  将任何寻址位转移到进位（CY），反之亦然。

- 位条件跳转指令

如果该位有一个指定的状态，则执行一个跳转，位条件跳转指令被归类为控制指令。

在片上RAM和SFR，位可以被单独的寻址。8xC251架构的位指令比80C51架构的位指令寻址范围更广。

两种架构地址位的指令在某些方面有差异。在80C51架构中，一个位（表示为bit51）可以在某一个寄存器中指定一个位置，或者可以在00h – 7Fh范围中指定一个位地址。8xC251架构并没有这样的位地址，一个位可以通过其名称或位置进行寻址，而不是通过位地址。

Figure 8-13 通过使用两个示例位寻址，来说明两种架构的位寻址：

- RAM_REG_b2是RAM_REG中的第二位，位置为2Fh。RAM_REG_b2和RAM_REG是定义在用户代码中的。
- TF0是TCON的第5位，它是一个位于88h位置的SFR。

构架	位地址位置	
	片内RAM	SFRs
8xC251	20h - 7Fh	所有定义的 SFRs
80C51	20h - 2Fh	SFRs以8h，0h结束的位置 (e.g. 88h, A0h,…)

Figure 8-12可寻址位的位置

“Bit”表示用MCS 251架构指令进行寻址，“bit51”表示用MCS 51架构指令进行寻址。

位置	地址模式	80C51 构架	8xC251 构架
片内 RAM	寄存器名称	RAM_REG.2	RAM_REG.2
	寄存器地址	2Fh.2	2Fh.2
	位名称	RAM_REG_b2	RAM_REG_b2
	位地址	7Ah	N.A.
SFR	寄存器名称	TCON.5	TCON.5
	寄存器地址	88.5h	88.5h
	位名称	TF0	TF0
	位地址	8Dh	N.A.

Figure 8-13 位寻址示例

构架	变量	位地址	内存/SFR 地址
8xC251 (位)	内存	NA	20h.0 - 7Fh.7
	SFR	NA	所有定义的SFRs
80C51 (位)	内存	00h - 7Fh	20h.0-2Fh.7
	SFR	80h - FFh	XXh.0 - XXh.7 XX= 80, 88, 90, 98, A0, A8, B0, B8, C0, C8, D0, D8, E0, E8, F0 or F8

Figure 8-14 位指令的寻址模式

8.19 控制指令

控制指令包括引用、返回、有条件和无条件的跳跃。代替执行队列中的下一条指令，处理器执行目标指令。HC16Lxx有一个24位程序计数器（PC），可以让目标指令到达16兆字节地址空间的任何地方，但上8兆被认为是代码存储器（80:0000h – FF:FFFFh）。本节讨论的，一些控制指令的目标地址限制到2k字节或64k字节地址范围，通过最低的11或16位的程序计数器改变。

8.20 控制指令的寻址模式

- 相对寻址：

控制指令提供的目标地址，是一个与下一条指令地址的8位带符号相对偏移量。

- 直接寻址：

控制指令提供的目标地址，可以有11位（ADDR11），16位（ADDR16）或24位（addr24）。

  - addr11

    只有较低的11位的PC被改变;比如，目标地址必须在2k字节的模块中，其包含下一条指令的第一个字节。

  - addr16

    只有较低的16位的PC被改变，即目标地址必须在64K字节的区域，包括将下一个指令的第一个字节。

  - addr24

    目标地址可以在16M字节地址空间的任何地方。

- 间接寻址：

对控制指令来讲，有两种类型的间接寻址指令：

  - LCALL@ WRJ和LJMP @ WRJ指令

    目标地址在64 K字节区域。WRJ中的低16位地址放在PC中的低16位。PC下一条指令的地址的高8位保持不变的。

  - JMP @ A + DPTR指令

    PC累加器和DPTR的总和被放置在PC的低16位，PC的高8位从一条指令的地址保持不变的。JMP@ A+ DPTR指令与8xC251标准的行为是不同的，该指令总是指向区域FF。

描述	地址位	地址范围
8位相对地址	8	-128 to 127 从下一条指令的第一个字节
11位直接地址 (addr11)	1	当前2 K字节
16位直接地址 (addr16)	1	当前 64 K字节
24位直接地址 (addr24) [1]	2	00:0000h - FF:FFFFh
间接地址 (@WRj) [1]	1	当前64 K字节
间接地址 (@A+DPTR)	1	当前64 K字节

Figure 8-15控制指令的寻址模式

注意事项：

1. 这些模式不能应用在MCS 51构架中

8.21 条件跳转

8xC251架构支持位条件跳转、条件比较跳转和基于累加器的值跳转。

位条件跳转是基于一个位的状态。在一个比较的条件跳转中，跳转是基于两个操作数的比较。所有的条件跳转都是相对的。指令集包含三种位条件跳转：

- JB（以位跳转）　　如果该位被设置，则跳转
- JNB（不以位跳转）　　如果该位被清除，则跳转
- JBC（跳转位，随后清除）　　如果该位被设置，则跳转，然后将其清除

比较指令跳转之前，会检查PSW和PSW1寄存器和它们的标志。每个标志的状态都由最后一个影响标志的指令决定。

状态标志用于测试以下操作数的6个关系：

- 等于（=），不等于（≠）
- 大于（>），小于（<）
- 大于等于（≥），小于等于（≤）

对于每一个关系，都有两个指令，一个有符号操作数和一个无符号操作数（见下表）。

类型	关系					
	=	≠	>	<	≥	≤
无符号	JE	JNE	JG	JL	JGE	JLE
有符号			JSG	JSL	JSGE	JSLE

Figure 8-26　比较条件跳转指令

8.22 无条件跳转

有五个无条件跳转。NOP和SJMP是相对于程序计数器跳转寻址。AJMP，LJMP和EJMP是直接或间接寻址。

- NOP （无操作） 无条件跳转到下一个指令
- SJMP（短跳转） 跳转到任何指令的下一条指令（-128 － 127）
- AJMP（绝对跳转） 改变PC最低11位以跳转到当前2K字节区域
- LJMP（长跳） 改变PC最低16位以跳转到当前任何64K字节区域。
- EJMP（扩展跳转） 改变PC所有24位以在16MB区域任意跳转

8.23 调用和返回

8xC251架构提供相对的、直接的、间接的调用和返回。

- ACALL（绝对调用）

  将下一条指令低16位的地址压入堆栈，然后更改PC低11位地址到此指令指定的十一位地址。此调用地址可以在下一个指令的2K字节内存块地址内。
- LCALL（长调用）

  将下一条指令的低16位地址压入堆栈，然后把PC的低16位地址更改成指令指定的的16位地址。该调用到的地址可以在下一条指令的64K字节内存块地址内。
- RET（返回）

  将顶部的两个字节从堆栈中推出，返回到子程序调用的指令地址。返回地址必须是相同的64K字节区域。
- ERET（异常返回）

  从堆栈中推出顶部的前三个字节，返回到子程序调用后的地址。返回的地址可以是16M字节的地址空间的任意位置。
- RETI（中断返回）

  提供从中断服务程序的返回。
- HC16Lxx只支持中断压栈4个字节。

8.24 对齐及非对齐内存访问

HC16Lxx规定大端模式、16位对齐模式或非对齐模式访问8位内存。字的存储顺序同样为大端模式，MSB字节处于低地址空间，LSB字节处于高地址空间。CPU没有限制对于字的对齐方式， 可以以偶地址访问（Address[0]=0），也可以以奇地址访问（Address[0]=1）。HC16Lxx内建了转换桥电路以支持对齐/非对齐模式的访问。

标准嵌入式Flash不支持单个时钟写入2个地址，因此HC16Lxx不支持以非对齐字形式写Flash，只支持以非对齐字形式的读Flash。

HC16Lxx支持非对齐RAM的读写。

访问类型	ADDR[23:0]	CPU写入的值 [15:8]	[7:0]	内存内容 (以字节为地址) @2N	@2N+1	@2N+2
对齐16位	2N	MSB	LSB	MSB	LSB	
非对齐16位	2N+1	MSB	LSB		LSB	MSB
8位奇地址	2N	MSB	x	MSB		
8位偶地址	2N	x	LSB		LSB	

Figure 8-37 对齐/非对齐word写

访问类型	ADDR[23:0]	内存内容 (以字节为地址) @2N	@2N+1	@2N+2	CPU读取的值 [15:8]	[7:0]
对齐16位	2N	MSB	LSB		MSB	LSB
非对齐16位	2N+1		LSB	MSB	MSB	LSB

Figure 8-48 对齐/非对齐word读

案列：
写16位对齐word:
mov 20h,WR0 with WR0=1234h => rambyte[1:0]=11 ;
ramdataout[15:0]=1234h ; @20h=12h and @21h=34h

写16位非对齐word:
mov 21h,WR0 with WR0=1234h => rambyte[1:0]=11 ;
ramdataout[15:0]=3412h ; @21h=12h and @22h=34h

写8位byte：
mov 20h, R0 with R0=12h => rambyte[1:0]=10 ;
ramdataout[15:0]=12xxh ; @20h=12h and @21h=unchanged

mov 21h, R0 with R0=34h => rambyte[1:0]=01 ;
ramdataout[15:0]=xx34h ; @20h=unchanged and @21h=34h

读16位对齐word：
mov WR0,20h with @20h=12h and @21h=34h => ramdatain[15:0]=1234h ;
WR0=1234h

读16位非对齐word：
mov WR0,21h with @21h=12h and @22h=34h => ramdatain[15:0]=3412h ;
WR0=1234h

9 指令集参考

本章包含8xC251架构的参考说明。它包括操作码和指令的长度和执行时间。

指令执行时间是从代码存储器中执行代码或往数据存储器中读或写的时间，这两个过程都非常快，都可以在一个时钟周期完成。当访问速度较慢的存储器时，需要插入等待状态来增加执行的时间。

9.1 寻址模式首字母缩写词

寻址模式	描述	8xC251	80C51
dir8	8位直接地址。可以是一个存储器地址(00:0000h – 00:00FFh)或SFR地址(S:00h – S:0FFh).	V	V
dir16	16位的内存地址(00:0000h-00:FFFFh)， 用于直接寻址。	V	

Figure 9-1 直接寻址符号

寻址模式	描述	8xC251	80C51
#data	8位常数，在一个指令中立即寻址	V	V
#data 16	16位常数，在一个指令中立即寻址	V	
#0data16 **#1data16**	32位常数，在一个指令中立即寻址 高字节是0（＃0data16）或者1（＃1data16）	V	
#short	1, 2或4位常数，在一个指令中立即寻址	V	

Figure 9-2 立即寻址符号

寻址模式	描述	8xC251	80C51
bit51	直接寻址位，可访问内部数据RAM或特殊功能寄存器（SFR）的地址(00h - FFh) 00h - 7Fh是128位，在内部RAM的字节地址是20h - 2Fh。 80h – FFh是128位，在16个SFR的字节地址用0h或8h用结束，如S:80h, S:88h, S:90h, ..., S:F0h, S:F8h		V
bit	直接寻址位，可以访问内存00:0020h-00:007Fh的地址或已定义SFR中的地址。	V	

Figure 9-3 位寻址符号

寻址模式	描述	8xC251	80C51
rel	一个有符号（2的补码）的8位相对地址 目的地是相对下一个指令的第一个字节的-128到127字节。	V	V
addr11	一个11位的目的地址。目的地是和下一个指令的第一个字节相同的2K字节的存储区。		V
addr16	一个16位的目的地址。目的地是和下一个指令的第一个字节相同64K字节的任何区域。		V
addr24	一个24位的目的地址。目的地可以在16M字节的地址空间的任何地方。	V	

Figure 9-4　控制指令符号

寻址模式	描述	8xC251	80C51
@Ri	通过字节寄存器R0和R1间接寻址内部数据地址(00h-FFh).		V
Rn **n**	当前所选的寄存器组的寄存器R0-R7 字节寄存器指数：n=0-7		V
Rm **Rmd** **Rms** **m, md, ms**	字节寄存器文件中的寄存器R0-R15 目标寄存器 源寄存器 字节寄存器指数：m，md，ms=0-15	V	
WRj **WRjd** **WRjs** **@WRj** **@WRj+dis16** **j, js, jd**	字寄存器文件中的字寄存器 WR0, WR2,…,WR30 目标寄存器 源寄存器 通过WR0-WR30间接寻址存储器地址(00:0000h-00:FFFFh) 通过字寄存器(WR0-WR30)+位移值（0-64K字节）间接寻址存储器地址(00:0000h-00:FFFFh) 字寄存器指数：j, js, jd=0-30	V	
DRk **DRkd** **DRks** **@DRk** **@DRk+dis24** **k, kd, ks**	双字寄存器，如DR0,DR4,...,DR28,DR56,DR60 目标寄存器 源寄存器 通过双字寄存器DR0-DR28,DR56,DR60间接寻址寄存器地址(00:0000h-FF:FFFFh) 通过双字寄存器 (DR0-DR28,DR56,DR60)+位移值（0-64K字节）间接寻址存储器位置(00:0000h-FF:FFFFh) 双字寄存器指数：k，kd，ks = 0, 4, 8, ..., 28, 56, 60	V	

Figure 9-5　寄存器操作数的表示法

9.2 代码大小和执行时间汇总

9.3 算术指令

指令助记符	字节		标准 8xC251 机器周期		HC16Lxx 机器周期	
	二进制模式	源模式	二进制模式	源模式	二进制模式	源模式
ADD A, Rn	1	2	2	4	1	1
ADD A, dir8	2	2	2	2	3	3
ADD A, @Ri	1	2	4	6	3	3
ADD A, #data	2	2	2	2	1	1
ADDC A, Rn	1	2	2	4	1	1
ADDC A, dir8	2	2	2	2	3	3
ADDC A, @Ri	1	2	4	6	3	3
ADDC A, #data	2	2	2	2	1	1
SUBB A, Rn	1	2	2	4	1	1
SUBB A, dir8	2	2	2	2	3	3
SUBB A, @Ri	1	2	4	6	3	3
SUBB A, #data	2	2	2	2	1	1
ADD Rmd, Rms	3	2	4	4	1	1
ADD WRjd, WRjs	3	2	6	4	1	1
ADD DRkd, DRks	3	2	10	4	2	2
ADD Rm, #data	4	3	6	6	1	1
ADD WRj, #data16	5	4	8	8	1	1
ADD DRk, #0data16	5	4	12	8	2	2
ADD Rm, dir8	4	3	6	6	3	3
ADD WRj, dir8	4	3	8	6	3	3
ADD Rm, dir16	5	4	6	8	3	3
ADD WRj, dir16	5	4	8	8	3	3
ADD Rm, @WRj	4	3	6	6	3	3
ADD Rm, @DRk	4	3	8	6	3	3
SUB Rmd, Rms	3	2	4	4	1	1
SUB WRjd, WRjs	3	2	6	4	1	1
SUB DRkd, DRks	3	2	10	4	2	2
SUB Rm, #data	4	3	6	6	1	1
SUB WRj, #data16	5	4	8	8	1	1
SUB DRk, #0data16	5	4	12	8	2	2
SUB Rm, dir8	4	3	6	6	3	3
SUB WRj, dir8	4	3	8	6	3	3
SUB Rm, dir16	5	4	6	8	3	3
SUB WRj, dir16	5	4	8	8	3	3
SUB Rm, @WRj	4	3	6	6	3	3
SUB Rm, @DRk	4	3	8	6	3	3

Figure 9-6 加法和减法指令汇总

指令助记符	字节		标准 8xC251 机器周期		HC16Lxx 机器周期	
	二进制模式	源模式	二进制模式		二进制模式	源模式
INC A	1	1	2	2	1	1
INC Rn	1	2	2	4	1	1
INC dir8	2	2	4	4	4	4
INC @Ri	1	2	6	8	4	4
DEC A	1	1	2	2	1	1
DEC Rn	1	2	2	4	1	1
DEC dir8	2	2	4	4	4	4
DEC @Ri	1	2	6	8	4	4
INC DPTR	1	1	2	2	1	1
INC Rm, #short	3	2	4	2	1	1
INC WRj, #short	3	2	4	2	1	1
INC DRk, #short	3	2	8	6	2	2
DEC Rm, #short	3	2	4	2	1	1
DEC WRj, #short	3	2	4	2	1	1
DEC DRk, #short	3	2	10	8	2	2

Figure 9-7 递增和递减指令汇总

指令助记符	字节		标准 8xC251 机器周期		HC16Lxx 机器周期	
	二进制模式	源模式	二进制模式	源模式	二进制模式	源模式
MUL AB	1	1	10	10	2	2
DIV AB	1	1	20	20	10	10
DA A	1	1	2	2	1	1
MUL Rmd, Rms	3	2	12	10	2	2
MUL WRjd, WRjs	3	2	24	22	8	8
DIV Rmd, Rms	3	2	22	20	10	10
DIV WRjd, WRjs	3	2	42	40	19	19

Figure 9-8 乘法，除法，十进制调整的指令汇总

指令助记符	字节		标准 8xC251 机器周期		HC16Lxx 机器周期	
	二进制模式	源模式	二进制模式	源模式	二进制模式	源模式
CMP Rmd, Rms	3	2	4	2	1	1
CMP WRjd, WRjs	3	2	6	4	1	1
CMP DRkd, DRks	3	2	10	8	2	2
CMP Rm, #data	4	3	6	4	1	1
CMP WRj, #data16	5	4	8	6	1	1
CMP DRk, #0data16	5	4	12	10	2	2
CMP DRk, #1data16	5	4	12	10	2	2
CMP Rm, dir8	4	3	6	4	3	3
CMP WRj, dir8	4	3	8	6	3	3
CMP Rm, dir16	5	4	6	4	3	3
CMP WRj, dir16	5	4	8	6	3	3
CMP Rm, @WRj	4	3	6	4	3	3
CMP Rm, @DRk	4	3	8	6	3	3

Figure 9-9 比较指令汇总

9.4 逻辑指令

指令助记符	字节		标准 8xC251 机器周期		HC16Lxx 机器周期	
	二进制模式	源模式	二进制模式	源模式	二进制模式	源模式
ANL A, Rn	1	2	2	4	1	1
ANL A, dir8	2	2	2	2	3	3
ANL A, @Ri	1	2	4	6	3	3
ANL A, #data	2	2	2	2	1	1
ANL dir8, A	2	2	4	4	4	4
ANL dir8, #data	3	3	6	6	4	4
ORL A, Rn	1	2	2	4	1	1
ORL A, dir8	2	2	2	2	3	3
ORL A, @Ri	1	2	4	6	3	3
ORL A, #data	2	2	2	2	1	1
ORL dir8, A	2	2	4	4	4	4
ORL dir8, #data	3	3	6	6	4	4
XRL A, Rn	1	2	2	4	1	1
XRL A, dir8	2	2	2	2	3	3
XRL A, @Ri	1	2	4	6	3	3
XRL A, #data	2	2	2	2	1	1
XRL dir8, A	2	2	4	4	4	4
XRL dir8, #data	3	3	6	6	4	4
CLR A	1	1	2	2	1	1
CPL A	1	1	2	2	1	1
RL A	1	1	2	2	1	1
RLC A	1	1	2	2	1	1
RR A	1	1	2	2	1	1
RRC A	1	1	2	2	1	1
SWAP A	1	1	4	4	1	1
ANL Rmd, Rms	3	2	4	2	1	1
ANL WRjd, WRjs	3	2	6	4	1	1
ANL Rm, #data	4	3	6	4	1	1
ANL WRj, #data16	5	4	8	6	1	1
ANL Rm, dir8	4	3	6	4	3	3
ANL WRj, dir8	4	3	8	6	3	3
ANL Rm, dir16	5	4	6	4	3	3
ANL WRj, dir16	5	4	8	6	3	3
ANL Rm, @WRj	4	3	6	4	3	3
ANL Rm, @DRk	4	3	8	6	3	3
ORL Rmd, Rms	3	2	4	2	1	1
ORL WRjd, WRjs	3	2	6	4	1	1
ORL Rm, #data	4	3	6	4	1	1
ORL WRj, #data16	5	4	8	6	1	1
ORL Rm, dir8	4	3	6	4	3	3
ORL WRj, dir8	4	3	8	6	3	3

ORL Rm, dir16	5	4	6	4	3	3
ORL WRj, dir16	5	4	8	6	3	3
ORL Rm, @WRj	4	3	6	4	3	3
ORL Rm, @DRk	4	3	8	6	3	3
XRL Rmd, Rms	3	2	4	2	1	1
XRL WRjd, WRjs	3	2	6	4	1	1
XRL Rm, #data	4	3	6	4	1	1
XRL WRj, #data16	5	4	8	6	1	1
XRL Rm, dir8	4	3	6	4	3	3
XRL WRj, dir8	4	3	8	6	3	3
XRL Rm, dir16	5	4	6	4	3	3
XRL WRj, dir16	5	4	8	6	3	3
XRL Rm, @WRj	4	3	6	4	3	3
XRL Rm, @DRk	4	3	8	6	3	3
SLL Rm	3	2	4	2	1	1
SLL WRj	3	2	4	2	1	1
SRA Rm	3	2	4	2	1	1
SRA WRj	3	2	4	2	1	1
SRL Rm	3	2	4	2	1	1
SRL WRj	3	2	4	2	1	1

Figure 9-10 逻辑指令汇总

9.5 数据转移

指令助记符	字节		标准 8xC251 机器周期		HC16Lxx 机器周期	
	二进制模式	源模式	二进制模式	源模式	二进制模式	源模式
MOVC A, @A+DPTR	1	1	12	12	2	2
MOVC A, @A+PC	1	1	12	12	2	2
MOVX A, @Ri	1	1	8	10	3	3
MOVX A,@DPTR	1	1	6	6	3	3
MOVX @Ri, A	1	1	8	8	2	2
MOVX @DPTR, A	1	1	8	8	2	2
MOVH DRk, #data16	5	4	6	4	1	1
MOVS WRj, Rm	3	2	4	2	1	1
MOVZ WRj, Rm	3	2	4	2	1	1

Figure 9-11 移动指令汇总 (1/3)

指令助记符	字节		标准 8xC251 机器周期		HC16Lxx 机器周期	
	二进制模式	源模式	二进制模式	源模式	二进制模式	源模式
MOV A, Rn	1	2	2	4	1	1
MOV A, dir8	2	2	2	2	3	3
MOV A, @Ri	1	2	4	6	3	3
MOV A, #data	2	2	2	2	1	1
MOV Rn, A	1	2	2	4	1	1
MOV Rn, dir8	2	3	2	4	3	3
MOV Rn, #data	2	3	2	4	1	1
MOV dir8, A	2	2	4	4	2	2
MOV dir8, Rn	2	3	4	6	2	2
MOV dir8, dir8	3	3	6	6	4	4
MOV dir8, @Ri	2	3	6	8	4	4
MOV dir8, #data	3	3	6	6	2	2
MOV @Ri, A	1	2	6	8	2	2
MOV @Ri, dir8	2	3	6	8	4	4
MOV @Ri, #data	2	3	6	8	2	2
MOV DPTR, #data16	3	3	4	4	1	1

Figure 9-12移动指令汇总 (2/3)

指令助记符	字节		标准 8xC251 机器周期		HC16Lxx 机器周期	
	二进制模式	源模式	二进制模式	源模式	二进制模式	源模式
MOV Rmd, Rms	3	2	4	2	1	1
MOV WRjd, WRjs	3	2	4	2	1	1
MOV DRkd, DRks	3	2	6	4	2	2
MOV Rm, #data	4	3	6	4	1	1
MOV WRj, #data16	5	4	6	4	1	1
MOV DRk, #0data16	5	4	10	8	2	2
MOV DRk, #1data16	5	4	10	8	2	2
MOV Rm, dir8	4	3	6	4	3	3
MOV WRj, dir8	4	3	8	6	3	3
MOV DRk, dir8	4	3	12	10	5	5
MOV Rm, dir16	5	4	6	4	3	3
MOV WRj, dir16	5	4	8	6	3	3
MOV DRk, dir16	5	4	12	10	5	5
MOV Rm, @WRj	4	3	6	4	3	3
MOV Rm, @DRk	4	3	8	6	3	3
MOV WRjd,@WRjs	4	3	8	6	3	3
MOV WRj,@DRk	4	3	10	8	3	3
MOV dir8, Rm	4	3	8	6	2	2
MOV dir8, WRj	4	3	10	8	2	2
MOV dir8, DRk	4	3	14	12	4	4
MOV dir16, Rm	5	4	8	6	2	2
MOV dir16, WRj	5	4	10	8	2	2
MOV dir16, DRk	5	4	14	12	4	4
MOV @WRj, Rm	4	3	8	6	2	2
MOV @DRk, Rm	4	3	10	8	2	2
MOV @WRjd, WRjs	4	3	10	8	2	2
MOV @DRk, WRj	4	3	12	10	2	2
MOV Rm, @WRj+dis16	5	4	12	10	3	3
MOV WRj, @WRj+dis16	5	4	14	12	3	3
MOV Rm, @DRk+dis24	5	4	14	12	3	3
MOV WRj, @DRk+dis24	5	4	16	14	3	3
MOV @WRj+dis16, Rm	5	4	12	10	2	2
MOV @WRj+dis16, WRj	5	4	14	12	2	2
MOV @DRk+dis24, Rm	5	4	14	12	2	2
MOV @DRk+dis24, WRj	5	4	16	14	2	2

Figure 9-13 移动指令汇总 (3/3)

指令助记符	字节		标准 8xC251 机器周期		HC16Lxx 机器周期	
	二进制模式	源模式	二进制模式	源模式	二进制模式	源模式
XCH A, Rn	1	2	6	8	1	1
XCH A, dir8	2	2	6	6	4	4
XCH A, @Ri	1	2	8	10	3	3
XCHD a, @Ri	1	2	8	10	4	4
PUSH dir8	2	2	4	4	4	4
POP dir8	2	2	6	6	4	4
PUSH #data	4	3	8	6	3	3
PUSH #data16	5	4	10	10	3	3
PUSH Rm	3	2	8	6	3	3
PUSH WRj	3	2	10	8	3	3
PUSH DRk	3	2	18	16	5	5
POP Rm	3	2	6	4	3	3
POP WRj	3	2	10	8	3	3
POP DRk	3	2	18	16	4	4

Figure 9-14 交换、推入、取出指令汇总

9.6 程序分支

指令助记符	字节		标准 8xC251 机器周期		HC16Lxx 机器周期	
	二进制模式	源模式	二进制模式	源模式 (*)	二进制模式 (*)	源模式 (*)
JC rel	2	2	2/8	2/8	2/5	2/5
JNC rel	2	2	2/8	2/8	2/5	2/5
JB bit51, rel	3	3	4/10	4/10	4/7	4/7
JNB bit51, rel	3	3	4/10	4/10	4/7	4/7
JBC bit51, rel	3	3	8/14	8/14	4/7	4/7
JZ rel	2	2	4/10	4/10	2/5	2/5
JNZ rel	2	2	4/10	4/10	2/5	2/5
CJNE A, dir8, rel	3	3	4/10	4/10	5/8	5/8
CJNE A, #data, rel	3	3	4/10	4/10	3/6	3/6
CJNE Rn, #data, rel	3	4	4/10	6/12	3/6	3/6
CJNE @Ri, #data, rel	3	4	6/12	8/14	5/8	5/8
DJNZ Rn, rel	2	3	4/10	6/12	3/6	3/6
DJNZ dir8, rel	3	3	6/12	6/12	5/8	5/8
JE rel	3	2	4/10	2/8	2/5	2/5
JNE rel	3	2	4/10	2/8	2/5	2/5
JG rel	3	2	4/10	2/8	2/5	2/5
JLE rel	3	2	4/10	2/8	2/5	2/5
JSL rel	3	2	4/10	2/8	2/5	2/5
JSLE rel	3	2	4/10	2/8	2/5	2/5
JSG rel	3	2	4/10	2/8	2/5	2/5
JSGE rel	3	2	4/10	2/8	2/5	2/5
JB bit, rel	5	4	8/14	6/12	4/7	4/7
JNB bit, rel	5	4	8/14	6/12	4/7	4/7
JBC bit, rel	5	4	14/20	12/18	4/7	4/7

Figure 9-15 条件转移指令汇总 (续)

指令助记符	字节		标准 8xC251 机器周期		HC16Lxx 机器周期	
	二进制模式	源模式	二进制模式	源模式	二进制模式	源模式
AJMP addr11	2	2	6	6	4	4
LJMP addr16	3	3	10	10	4	4
SJMP rel	2	2	8	8	4	4
JMP @A+DPTR	1	1	10	10	4	4
NOP	1	1	2	2	1	1
EJMP addr24	5	4	12	10	5	5
EJMP @DRk	3	2	14	12	5	5
LJMP @WRj	3	2	12	10	5	5

Figure 9-16 无条件跳转指令汇总

指令助记符	字节		标准 8xC251 机器周期		HC16Lxx 机器周期	
	二进制模式	源模式	二进制模式	源模式	二进制模式	源模式
ACALL addr11	2	2	18	18	4	4
LCALL addr16	3	3	18	18	4	4
RET	1	1	14	14	6	6
RETI	1	1	14	14	5/7([14])	5/7([15])
ECALL @DRk	3	2	28	26	5	5
ECALL addr24	5	4	28	26	5	5
LCALL @WRj	3	2	20	18	4	4
ERET	3	2	18	16	7	7
TRAP	2	1	24	22	6	6

Figure 9-17 调用和返回指令汇总

9.7 布尔操作

指令助记符	字节		标准 8xC251 机器周期		HC16Lxx 机器周期	
	二进制模式	源模式	二进制模式	源模式	二进制模式	源模式
CLR CY	1	1	2	2	1	1
CLR bit51	2	2	4	4	4	4
SETB CY	1	1	2	2	1	1
SETB bit51	2	2	4	4	4	4
CPL CY	1	1	2	2	1	1
CPL bit51	2	2	4	4	4	4
ANL CY, bit51	2	2	2	2	3	3
ANL CY, /bit51	2	2	2	2	3	3
ORL CY, bit51	2	2	2	2	3	3
ORL CY, /bit51	2	2	2	2	3	3
MOV CY, bit51	2	2	2	2	3	3
MOV bit51, CY	2	2	4	4	4	4
CLR bit	4	3	8	6	4	4
SETB bit	4	3	8	6	4	4
CPL bit	4	3	8	6	4	4
ANL CY, bit	4	3	6	4	3	3
ANL CY, /bit	4	3	6	4	3	3
ORL CY, bit	4	3	6	4	3	3
ORL CY, /bit	4	3	6	4	3	3
MOV CY, bit	4	3	6	4	3	3
MOV bit, CY	4	3	8	6	4	4

Figure 9-18 位指令汇总

9.8 操作码映射

Bin.	0	1	2	3	4	5	6-7	8-F
Src.	0	1	2	3	4	5	A5x6-A5x7	A5x8-A5xF
0	NOP	AJMP addr11	LJMP addr16	RR A	INC A	INC dir8	INC @Ri	INC Rn
1	JBC bit51, rel	ACALL addr11	LCALL addr16	RRC A	DEC A	DEC dir8	DEC @Ri	DEC Rn
2	JB bit51, rel	AJMP addr11	RET	RL A	ADD A, #data	ADD A, dir8	ADD A, @Ri	ADD A, Rn
3	JNB bit51, rel	ACALL addr11	RETI	RLC A	ADDC A, #data	ADDC A, dir8	ADDC A, @Ri	ADDC A, Rn
4	JC rel	AJMP addr11	ORL dir8, A	ORL dir8, #data	ORL A, #data	ORL A, dir8	ORL A, @Ri	ORL A, Rn
5	JNC rel	ACALL addr11	ANL dir8, A	ANL dir8, #data	ANL A, #data	ANL A, dir8	ANL A, @Ri	ANL A, Rn
6	JZ rel	AJMP addr11	XRL dir8, A	XRL dir8, #data	XRL A, #data	XRL A, dir8	XRL A, @Ri	XRL A, Rn
7	JNZ rel	ACALL addr11	ORL CY, bit51	JMP @A+DPTR	MOV A, #data	MOV dir8, #data	MOV @Ri, #data	MOV Rn, #data
8	SJMP rel	AJMP addr11	ANL CY, bit51	MOVC A, @A+PC	DIV AB	MOV dir8, dir8	MOV dir8, @Ri	MOV dir8, Rn
9	MOV DPTR, #data16	ACALL addr11	MOV bit51, CY	MOVC A, @A+DPTR	SUBB A, #data	SUBB A, dir8	SUBB A, @Ri	SUBB A, Rn
A	ORL CY, /bit51	AJMP addr11	MOV CY, bit51	INC DPTR	MUL AB	ESC	MOV @Ri, dir8	MOV Rn, dir8
B	ANL CY, /bit51	ACALL addr11	CPL bit51	CPL CY	CJNE A,#data,rel	CJNE A,dir8,rel	CJNE @Ri,#data, rel	CJNE Rn,#data, rel
C	PUSH dir8	AJMP addr11	CLR bit51	CLR CY	SWAP A	XCH A, dir8	XCH A, @Ri	XCH A, Rn
D	POP dir8	ACALL addr11	SETB bit51	SETB CY	DA A	DJNZ dir8, rel	XCHD A, @Ri	DJNZ Rn, rel
E	MOVX A, @DPTR	AJMP addr11	MOVX A, @Ri		CLR A	MOV A, dir8	MOV A, @Ri	MOV A, Rn
F	MOVX @DPTR, A	ACALL addr11	MOVX @Ri, A		CPL A	MOV dir8, A	MOV @Ri, A	MOV Rn, A

Figure 9-19　80C51构架微控制器指令

Bin.	A5x8	A5x9	A5xA	A5xB	A5xC	A5xD	A5xE	A5xF
Src.	x8	x9	xA	xB	xC	xD	xE	xF
0	JSLE rel	MOV Rm, @WRj+dis	MOVZ WRj, Rm	INC R,#short (1) MOV reg,ind			SRA reg	
1	JSG rel	MOV @WRj+dis,Rm	MOVS WRj, Rm	DEC R,#short (1) MOV ind,reg			SRL reg	
2	JLE rel	MOV Rm,@DRk+dis			ADD Rm, Rm	ADD WRj, WRj	ADD reg, op2 (2)	ADD DRk, DRk
3	JG rel	MOV @DRk+dis, Rm					SLL reg	
4	JSL rel	MOV WRj,@WRj+dis			ORL Rm, Rm	ORL WRj, WRj	ORL reg, op2 (2)	
5	JSGE rel	MOV @WRj+dis, WRj			ANL Rm, Rm	ANL WRj, WRj	ANL reg, op2 (2)	
6	JE rel	MOV WRj,@DRk+dis			XRL Rm, Rm	XRL WRj, WRj	XRL reg, op2 (2)	
7	JNE rel	MOV @DRk+dis, WRj	MOV op1,reg (2)		MOV Rm, Rm	MOV WRj, WRj	MOV reg, op2 (2)	MOV DRk, DRk
8		LJMP @WRj EJMP @DRk	EJMP addr24		DIV Rm, Rm	DIV WRj, WRj		
9		LCALL @WRj ECALL @DRk	ECALL addr24		SUB Rm, Rm	SUB WRj, WRj	SUB reg, op2 (2)	SUB DRk, DRk
A		Bit Instructions (3)	ERET		MUL Rm, Rm	MUL WRj, WRj		
B		TRAP			CMP Rm, Rm	CMP WRj, WRj	CMP reg, op2 (2)	CMP DRk, DRk
C			PUSH op1					
D			POP op1 (4)					
E								
F								

Figure 9-20 8xC251构架微控制器指令

- R = Rm/WRj/DRk
- op1/op2 在Figure 9-21中描述
- 参考 Figure 9-23、Figure 9-24和Figure 9-25

9.9 指令编码

指令	字节0		字节1		字节2		字节3
Oper Rmd, Rms	x	C	md	ms			
Oper WRjd, WRjs	x	D	jd/2	js/2			
Oper DRkd, DRks	x	F	kd/4	ks/4			
Oper Rm, #data	x	E	m	0	#data		
Oper WRj, #data16	x	E	j/2	4	#data (high)		#data (low)
Oper DRk, #0data16	x	E	k/4	8	#data (high)		#data (low)

MOVH DRk, #data16
MOV DRk, #1data16
CMP DRk, #1data16
	7
	7
	B
		A
		E
		E
			k/4	C	#data (high)		#data (low)

Oper Rm, dir8	x	E	m	1	dir8 addr		
Oper WRj, dir8	x	E	j/2	5	dir8 addr		
Oper DRk, dir8	x	E	k/4	D	dir8 addr		
Oper Rm, dir16	x	E	m	3	dir16 addr (high)		dir16 addr (low)
Oper WRj, dir16	x	E	j/2	7	dir16 addr (high)		dir16 addr (low)
Oper DRk, dir16 (1)	x	E	k/4	F	dir16 addr (high)		dir16 addr (low)
Oper Rm, @WRj	x	E	j/2	9	m	0	
Oper Rm, @DRk	x	E	k/4	B	m	0	

Figure 9-21　数据指令的编码

以下操作数仅适用于MOV指令。

x	操作	注意事项
2	ADD reg, op2	支持所有寻址模式 (1)
9	SUB reg, op2	
B	CMP reg, op2	
4	ORL reg, op2 (2)	
5	ANL reg, op2 (2)	
6	XRL reg, op2 (2)	
7	MOV reg, op2	
8	DIV reg, op2	只支持两种模式 op2 =Rmd, Rms或WRjd, WRjs
A	MUL reg, op2	

Figure 9-22　数据指令的高半0字节的编码

1)　DRK，dir16只对MOV指令有效。

2)　对于ORL，ANL和XRL来说，寄存器或op2都不可以是DRK。

8xC251架构中所有位指令都有操作码A9，它会作为一个转义字节（类似A5）。第1个字节的高四位定义了位指令。

	指令	字节0 (X)		字节1			字节2	字节3 *
1	Bit instr (dir8)	A	9	xxxx	0	bit	dir8 addr	rel addr

Figure 9-23 位指令编码

（*）只能用于跳转位指令（JBC/JB/JNB）

xxxx	位指令
1	JBC bit
2	JB bit
3	JNB bit
7	ORL CY, bit
8	ANL CY, bit
9	MOV bit, CY
A	MOV CY, bit
B	CPL bit
C	CLR bit
D	SETB bit
E	ORL CY, /bit
F	ANL CY, /bit

Figure 9-24 位指令的高四位字节1的编码

指令	字节0 (x)		字节1		字节2	字节3
PUSH #data	C	A	0	2	#data	
PUSH #data16	C	A	0	6	#data16 (high)	#data16 (low)
PUSH Rm	C	A	m	8		
PUSH WRj	C	A	j/2	9		
PUSH DRk	C	A	k/4	B		
POP Rm	D	A	m	8		
POP WRj	D	A	j/2	9		
POP DRk	D	A	k/4	B		

Figure 9-25 PUSH / POP指令编码

指令	字节0		字节1		字节2	字节3
EJMP addr24	8	A	addr[23:16]		addr[15:8]	addr[7:0]
ECALL addr24	9	A	addr[23:16]		addr[15:8]	addr[7:0]
LJMP @WRj	8	9	j/2	4		
LCALL @WRj	9	9	j/2	4		
EJMP @DRk	8	9	k/4	8		
ECALL @DRk	9	9	k/4	8		
ERET	A	A				
JE rel	6	8	rel			
JNE rel	7	8	rel			
JLE rel	2	8	rel			
JG rel	3	8	rel			
JSL rel	4	8	rel			
JSGE rel	5	8	rel			
JSLE rel	0	8	rel			
JSG rel	1	8	rel			
TRAP	B	9				

Figure 9-26 控制指令编码

指令	字节0		字节1		字节2		字节3
MOV Rm, @WRj+dis16	0	9	m	j/2	dis[15:8]		dis[7:0]
MOV WRjd, @WRjs+dis16	4	9	jd/2	js/2	dis[15:8]		dis[7:0]
MOV Rm, @DRk+dis24	2	9	m	k/4	dis[15:8]		dis[7:0]
MOV WRj, @DRk+dis24	6	9	j/2	k/4	dis[15:8]		dis[7:0]
MOV @WRj+dis16, Rm	1	9	m	j/2	dis[15:8]		dis[7:0]
MOV @WRjd+dis16, WRjs	5	9	js/2	jd/2	dis[15:8]		dis[7:0]
MOV @DRk+dis24, Rm	3	9	m	k/4	dis[15:8]		dis[7:0]
MOV @DRk+dis24, WRj	7	9	j/2	k/4	dis[15:8]		dis[7:0]
MOVS WRj, Rm	1	A	j/2	m			
MOVZ WRj, Rm	0	A	j/2	m			
MOV WRjd, @WRjs	0	B	js/2	8	jd/2	0	
MOV WRj, @DRk	0	B	k/4	A	j/2	0	
MOV @WRjd, WRjs	1	B	jd/2	8	js/2	0	
MOV @DRk, WRj	1	B	k/4	A	j/2	0	
MOV dir8, Rm	7	A	m	1	dir8 addr		
MOV dir8, WRj	7	A	j/2	5	dir8 addr		
MOV dir8, DRk	7	A	k/4	D	dir8 addr		
MOV dir16, Rm	7	A	m	3	dir16 addr (high)		dir16 addr (low)
MOV dir16, WRj	7	A	j/2	7	dir16 addr (high)		dir16 addr (low)
MOV dir16, DRk	7	A	k/4	F	dir16 addr (high)		dir16 addr (low)
MOV @WRj, Rm	7	A	j/2	9	m	0	
MOV @DRk, Rm	7	A	k/4	B	m	0	

Figure 9-27　位移/扩展/移动指令的编码

	指令	字节0 (x)		字节1		
1	INC Rm, #short	0	B	m	00	vv
2	INC WRj, #short	0	B	j/2	01	vv
3	INC DRk, #short	0	B	k/4	11	vv
4	DEC Rm, #short	1	B	m	00	vv
5	DEC WRj, #short	1	B	j/2	01	vv
6	DEC DRk, #short	1	B	k/4	11	vv

Figure 9-28增加减少指令的编码

vv	#short
00	1
01	2
10	4

Figure 9-29　字节1的增加/减少指令的编码

	指令	Byte 0		Byte 1	
1	SRA Rm	0	E	m	0
2	SRA WRj	0	E	j/2	4
3	SRL Rm	1	E	m	0
4	SRL WRj	1	E	j/2	4
5	SLL Rm	3	E	m	0
6	SLL WRj	3	E	j/2	4

Figure 9-30 转移指令的编码

9.10 性能比较

二进制模式		源模式	
操作数	速度改善	指令数	速度改善
1	6.00	2	8.00
1	5.00	1	7.00
9	4.50	8	6.00
5	4.00	2	5.20
5	3.00	5	5.00
4	2.67	1	4.67
2	2.50	1	4.50
2	2.33	28	4.00
1	2.00	1	3.67
1	1.56	2	3.33
2	1.50	1	3.20
2	1.43	6	3.00
3	1.38	1	2.75
1	1.33	1	2.73
2	1.27	7	2.67
1	1.08	1	2.40
1	1.00	1	2.29
2	0.67	1	2.11
		52	2.00
		2	1.64
		4	1.50
		8	1.43
		21	1.33
总数	加权平均	总数	加权平均
2	**2.1 X**	**157**	**2.8 X**

Figure 9-31 标准C251和HC16Lxx指令集速度的比较

在源模式下使用指令整体速度提高约 2.8 倍。但是，实际速度的提高依赖于应用程序。

10 Flash 控制器

10.1 Flash 控制器简介

10.2 概述

HC16Lxx 内置了1块16位宽度的超低功耗Flash存储器，具有如下特点：

- 存储空间大小为32K~128K字节(512字节/页)
- 支持字，字节和双字编程，以及整片擦除或者页擦除
- 由内建LDO产生工作电压，不需要外部任何的VPP 高压输入

注意事项：在对Flash存储器进行编程或擦除操作之前，为了防止电压突降造成Flash数据丢失，需要使能片内LVD 低电压复位。

10.3 Flash 特性

参数	测试环境	最小	标准	最大	单位
可擦除编程的次数	常温25度下	20K			次
数据保存时间	温度为85度下	10			年
	常温25度下	100			年
编程时间	常温25度下	20		40	μs
页擦除时间	常温25度下	20		40	ms
整片擦除时间	常温25度下	20		40	ms

Figure 10-1 Flash 特性表

10.4 Flash 页地址定义

Flash 页	SFR	程序区	数据区	页面大小
Page63	EF_pagelock7[7]	FF:7E00 ~ FF:7FFF	7F:7E00 ~ 7F:7FFF	512 字节
Page62	EF_pagelock7[6]	FF:7C00 ~ FF:7DFF	7F:7C00 ~ 7F:7DFF	512 字节
Page61	EF_pagelock7[5]	FF:7A00 ~ FF:7BFF	7F:7A00 ~ 7F:7BFF	512 字节
Page60	EF_pagelock7[4]	FF:7800 ~ FF:79FF	7F:7800 ~ 7F:79FF	512 字节
Page59	EF_pagelock7[3]	FF:7600 ~ FF:77FF	7F:7600 ~ 7F:77FF	512 字节
Page58	EF_pagelock7[2]	FF:7400 ~ FF:75FF	7F:7400 ~ 7F:75FF	512 字节
Page57	EF_pagelock7[1]	FF:7200 ~ FF:73FF	7F:7200 ~ 7F:73FF	512 字节
Page56	EF_pagelock7[0]	FF:7000 ~ FF:71FF	7F:7000 ~ 7F:71FF	512 字节
Page55	EF_pagelock6[7]	FF:6E00 ~ FF:6FFF	7F:6E00 ~ 7F:6FFF	512 字节
Page54	EF_pagelock6[6]	FF:6C00 ~ FF:6DFF	7F:6C00 ~ 7F:6DFF	512 字节
Page53	EF_pagelock6[5]	FF:6A00 ~ FF:6BFF	7F:6A00 ~ 7F:6BFF	512 字节
Page52	EF_pagelock6[4]	FF:6800 ~ FF:69FF	7F:6800 ~ 7F:69FF	512 字节
Page51	EF_pagelock6[3]	FF:6600 ~ FF:67FF	7F:6600 ~ 7F:67FF	512 字节
Page50	EF_pagelock6[2]	FF:6400 ~ FF:65FF	7F:6400 ~ 7F:65FF	512 字节
Page49	EF_pagelock6[1]	FF:6200 ~ FF:63FF	7F:6200 ~ 7F:63FF	512 字节
Page48	EF_pagelock6[0]	FF:6000 ~ FF:61FF	7F:6000 ~ 7F:61FF	512 字节
Page47~Page40	EF_pagelock5[7:0]	FF:5000~FF:5FFF		
Page39~Page32	EF_pagelock4[7:0]	FF:4000~FF:4FFF		
Page31~Page24	EF_pagelock3[7:0]	FF:3000~FF:3FFF		
Page23~Page16	EF_pagelock2[7:0]	FF:2000~FF:2FFF		
Page15	EF_pagelock1[7]	FF:1E00 ~ FF:1FFF	7F:1E00 ~ 7F:1FFF	512 字节
Page14	EF_pagelock1[6]	FF:1C00 ~ FF:1DFF	7F:1C00 ~ 7F:1DFF	512 字节
Page13	EF_pagelock1[5]	FF:1A00 ~ FF:1BFF	7F:1A00 ~ 7F:1BFF	512 字节
Page12	EF_pagelock1[4]	FF:1800 ~ FF:19FF	7F:1800 ~ 7F:19FF	512 字节
Page11	EF_pagelock1[3]	FF:1600 ~ FF:17FF	7F:1600 ~ 7F:17FF	512 字节
Page10	EF_pagelock1[2]	FF:1400 ~ FF:15FF	7F:1400 ~ 7F:15FF	512 字节
Page9	EF_pagelock1[1]	FF:1200 ~ FF:13FF	7F:1200 ~ 7F:13FF	512 字节
Page8	EF_pagelock1[0]	FF:1000 ~ FF:11FF	7F:1000 ~ 7F:11FF	512 字节
Page7	EF_pagelock0[7]	FF:0E00 ~ FF:0FFF	7F:0E00 ~ 7F:0FFF	512 字节
Page6	EF_pagelock0[6]	FF:0C00 ~ FF:0DFF	7F:0C00 ~ 7F:0DFF	512 字节
Page5	EF_pagelock0[5]	FF:0A00 ~ FF:0BFF	7F:0A00 ~ 7F:0BFF	512 字节
Page4	EF_pagelock0[4]	FF:0800 ~ FF:09FF	7F:0800 ~ 7F:09FF	512 字节
Page3	EF_pagelock0[3]	FF:0600 ~ FF:07FF	7F:0600 ~ 7F:07FF	512 字节
Page2	EF_pagelock0[2]	FF:0400 ~ FF:05FF	7F:0400 ~ 7F:05FF	512 字节
Page1	EF_pagelock0[1]	FF:0200 ~ FF:03FF	7F:0200 ~ 7F:03FF	512 字节
Page0	EF_pagelock0[0]	FF:0000 ~ FF:01FF	7F:0000 ~ 7F:01FF	512 字节

Figure 10-2 Flash 页定义

10.5 Flash 操作

10.6 RAM 中运行程序进行整片页擦除

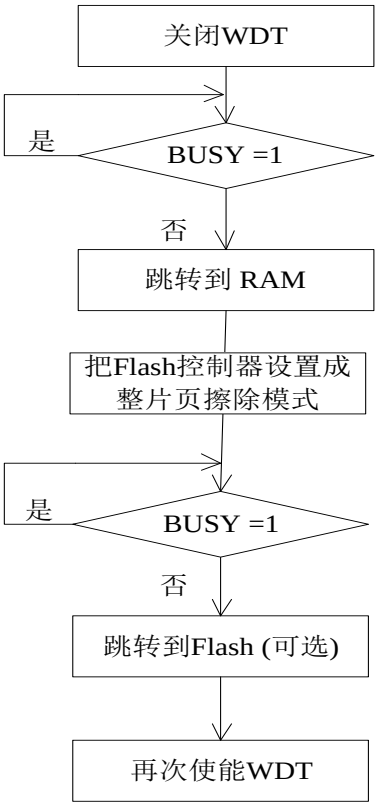


Figure 10-3 RAM 中运行程序进行 Flash 整片页擦除

10.7 Flash 中运行程序进行页擦除

HC16Lxx 不支持在 Flash 中运行程序进行当前页页擦除（这将导致 Flash_CTL.error_flag 产生），但支持非当前页的页擦除。

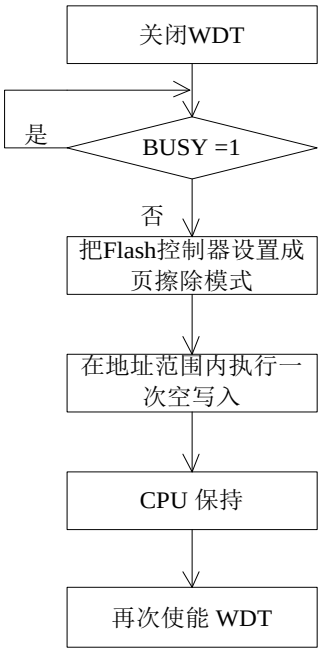


Figure 10-4 Flash 中运行程序进行 Flash 页擦除

10.8 RAM 中运行程序进行页擦除

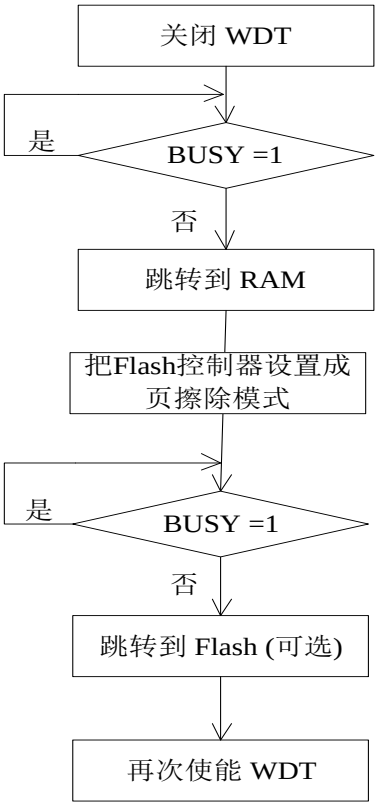


Figure 10-5 RAM 中运行程序进行 Flash 页擦除

10.9 Flash 中运行程序进行编程

HC16Lxx 不允许当前页编程（这将导致 Flash_CTL.error_flag 产生），但支持非当前页非加锁的页进行编程。

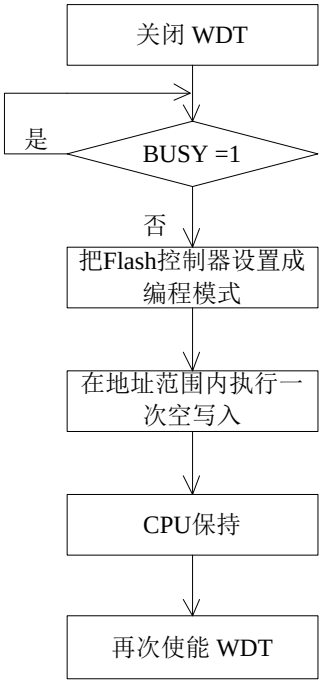


Figure 10-6 Flash 中运行程序进行 Flash 编程

10.10 RAM 中运行程序进行编程

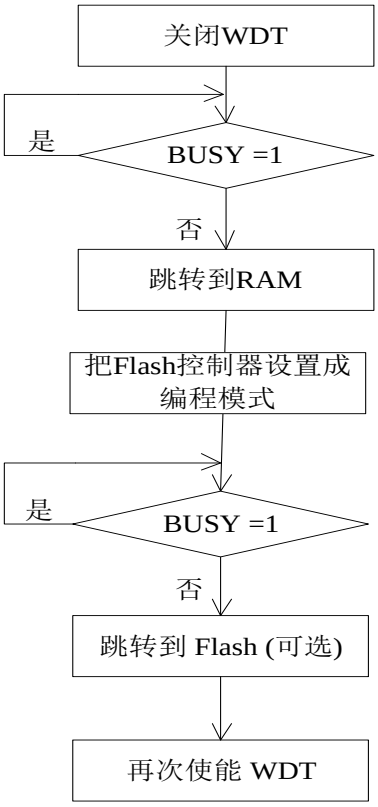


Figure 10-7 RAM 中运行程序进行 Flash 编程

10.11 Flash 控制寄存器

寄存器缩写	寄存器名称	复位值 4M 时序	16M 推荐设定值	8M 推荐设定值	4M 推荐设定值	2M 推荐设定值
Tnvs	NVSTR 建立时间寄存器	8’d20	8’ d80	8’ d40	8’d20	8’ d10
Tpgs	NVSTR 与编程信号建立时间寄存器	8’d40	8’ d160	8’ d80	8’d40	8’ d20
Tprog	Flash 编程时间寄存器	8’d40	8’ d240	8’ d120	8’d60	8’ d30
Tnvh	NVSTR 保持时间寄存器	8’d20	8’ d80	8’ d40	8’d20	8’ d10
Trcv	NVSTR 恢复时间寄存器	8’d4	8’ d16	8’ d8	8’d4	8’ d2
Terase	Flash 擦除时间寄存器	8’d20	8’ d120	8’ d60	8’d30	8’ d15
Tme	整片擦除时间寄存器	8’d20	8’ d120	8’ d60	8’d30	8’ d15
Tnvh1	整片擦除中 NVSTR 保持时间寄存器	8’d25	8’ d100	8’ d50	8’d25	8’ d13
EF_pagelock0	Flash 页保护寄存器 0	8’ h00				
EF_pagelock1	Flash 页保护寄存器 1	8’ h00				
EF_pagelock2	Flash 页保护寄存器 2	8’ h00				
EF_pagelock3	Flash 页保护寄存器 3	8’ h00				

EF_pagelock4	Flash 页保护寄存器 4	8’h00	
EF_pagelock5	Flash 页保护寄存器 5	8’h00	
EF_pagelock6	Flash 页保护寄存器 6	8’h00	
EF_pagelock7	Flash 页保护寄存器 7	8’h00	
Flash_CTL	Flash 控制寄存器	8’h00	
Flash_PL	Flash 页加解锁控制器	8’h00	

Figure 10-8 Flash 控制寄存器

参数	符号	最小	最大	单位
NVSTR 建立时间	Tnvs	5	-	μs
NVSTR 保持时间	Tnvh	5	-	μs
NVSTR 保持时间(整片擦除)	Tnvh1	100	-	μs
NVSTR 与编程信号建立时间	Tpgs	10	-	μs
编程时间	Tprog	20	40	μs
恢复时间	Trcv	1	-	μs
擦除时间	Terase	20	40	ms
整片擦除时间	Tme	20	40	ms
累计编程 HV 周期	Thv	-	8	ms

Figure 10-9 Flash 重要时序

10.12 NVSTR 建立时间寄存器（Tnvs）

Tnvs								
NVSTR 建立时间寄存器								
地址空间： 0x00F800								
位	7	6	5	4	3	2	1	0
符号	--	--	--	--	--	--	--	--
类型	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	0	1	0	1	0	0
位	符号	描述						
[7:0]		NVSTR建立时间 Tnvs = 20*250ns ~ 5us @4MHz						

Figure 10-10 NVSTR 建立时间寄存器

10.13 NVSTR 与编程信号建立时间寄存器(Tpgs)

Tpgs								
NVSTR与编程信号建立时间寄存器								
地址空间： 0x00F802								
位	7	6	5	4	3	2	1	0
符号	--	--	--	--	--	--	--	--
类型	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	1	0	1	0	0	0
位	符号	描述						
[7:0]		NVSTR与编程信号建立时间 Tpgs = 40*250ns ~ 10uS @4MHz						

Figure 10-11 NVSTR 与编程信号建立时间寄存器

10.14 Flash 编程时间寄存器(Tprog)

Tprog								
Flash编程时间寄存器								
地址空间： 0x00F804								
位	7	6	5	4	3	2	1	0
符号	--	--	--	--	--	--	--	--
类型	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	1	0	1	0	0	0
位	符号	描述						
[7:0]		Flash编程时间 Tprog = 40 *2 *250ns ~ 20uS @4MHz						

Figure 10-12 Flash 编程时间寄存器

10.15 NVSTR 保持时间寄存器(Tnvh)

Tnvh								
NVSTR保持时间寄存器								
地址空间： 0x00F806								
位	7	6	5	4	3	2	1	0
符号	--	--	--	--	--	--	--	--
类型	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	0	1	0	1	0	0
位	符号	描述						
[7:0]		NVSTR保持时间 Tnvh = 20*250ns ~ 5us @4MHz						

Figure 10-13 NVSTR 保持时间寄存器

10.16 NVSTR 恢复时间寄存器(Trcv)

Trcv								
NVSTR恢复时间寄存器								
地址空间： 0x00F808								
位	7	6	5	4	3	2	1	0
符号	--	--	--	--	--	--	--	--
类型	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	0	0	0	1	0	0
位	符号	描述						
[7:0]		NVSTR恢复时间 Trcv = 4*250ns ~ 1us @4MHz						

Figure 10-14 NVSTR 恢复时间寄存器

10.17 Flash 擦除时间寄存器(Terase)

Terase								
Flash擦除时间寄存器								
地址空间： 0x00F80A								
位	7	6	5	4	3	2	1	0
符号	--	--	--	--	--	--	--	--
类型	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	0	1	0	1	0	0
位	符号	描述						
[7:0]		Flash 擦除时间 Terase = 40 *4096 * 250ns ~ 40ms						

Figure 10-15 Flash 擦除时间寄存器

10.18 整片擦除时间寄存器(Tme)

Tme								
整片擦除时间寄存器								
地址空间： 0x00F80C								
位	7	6	5	4	3	2	1	0
符号	--	--	--	--	--	--	--	--
类型	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	0	1	0	1	0	0
位	符号	描述						
[7:0]		整片擦除时间 mass erase Tme = 40 *4096 * 250ns ~ 40ms						

Figure 10-16 整片擦除时间寄存器

10.19 整片擦除中 NVSTR 保持时间寄存器(Tnvh1)

Tnvh1								
整片擦除中NVSTR保持时间寄存器								
地址空间： 0x00F80E								
位	7	6	5	4	3	2	1	0
符号	--	--	--	--	--	--	--	--
类型	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	0	1	1	0	0	1
位	符号	描述						
[7:0]		整芯片擦除中NVSTR保持时间 Tme = 25 *16 * 250ns ~ 100us						

Figure 10-17 整片擦除中 NVSTR 保持时间寄存器

10.20 Flash 页保护寄存器 x(EF_pagelockx,x=0~7)

寄存器	复位值	类型	地址空间	描述
EF_pagelock0	8'h00	R/W	0x00F810	Page7~Page0 锁存位
EF_pagelock1	8'h00	R/W	0x00F812	Page15~Page8 锁存位
EF_pagelock2	8'h00	R/W	0x00F814	Page23~Page16 锁存位
EF_pagelock3	8'h00	R/W	0x00F816	Page31~Page24 锁存位
EF_pagelock4	8'h00	R/W	0x00F818	Page39~Page32 锁存位
EF_pagelock5	8'h00	R/W	0x00F81A	Page47~Page40 锁存位
EF_pagelock6	8'h00	R/W	0x00F81C	Page55~Page48 锁存位
EF_pagelock7	8'h00	R/W	0x00F81E	Page63~Page56 锁存位

Figure 10-18 EF_pagelockx寄存器构成

EF_pagelockx								
Flash 页保护寄存器x								
位	7	6	5	4	3	2	1	0
符号	--	--	--	--	--	--	--	--
类型	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	0	0	0	0	0	0
位	符号	描述						
[7:0]		Pagex 锁存位						

Figure 10-19 EF_pagelockx寄存器

为了保证Flash的安全性，特别是在上电掉电或者电压异常变化时对Flash进行擦写操作，Flash的数据就会出现丢失或改写，系统就会出现异常错误，HC16Lxx内建了Flash保护电路。一共有64位页保护寄存器EF_pagelock0 ~ EF_pagelock7。

0：锁存住，只能读，不能修改，即不能进行擦除和编程。

1：非锁存，可以进行读写操作，即可以进行擦除和编程。

注意事项:

- 寄存器的各位分别设定到 Flash 的各页上。位配置和 Flash 页一一对应。

  例如，EF_pagelock0 的第 7 位对应 Flash Page7; EF_pagelock4 的第 0 位对应 Flash Page32。

加减锁原理：

（1）当上电复位中或上电复位之后，页锁存寄存器为“0”，表示所有的Flash处于保护状态，只有读才能被执行。

（2）HC16Lxx会去查看Flash 锁存区的数据，（Figure 10-20定义的地址，位于用户程序区）随后配置Flash锁存器，EF_pagelock0 ~ EF_pagelock7被配置后，CPU开始运行用户程序。

（3）用户在使用中进行页的加减锁过程：写0x5A到Flash_PL，写0xA5到Flash_PL，写需要定义的值到EF_pagelockx进行加减锁，进行配置。

（4）如果需要操作的页不属于同一个EF_pagelockx，那么重复执行（3）序列，写需要定义的值到EF_pagelockx进行加减锁，进行配置。

（5）只有上电复位之后，Flash锁存器被配置，其他复位源进来后，Flash锁存器不会被配置。即用户使用加减锁配置之后，非上电复位的其他复位之后，这些配置还保存着。但上电复位之后，这些加减锁配置就会丢失，芯片执行（2），系统重新上电读取页锁存区地址的数据进行EF_pagelockx的配置。

Flash_Protect0		=24’hFF0086 (Flash 地址)
Flash_Protect1		=24’hFF0087 (Flash 地址)
Flash_Protect2		=24’hFF0088 (Flash 地址)
Flash_Protect3		=24’hFF0089 (Flash 地址)
Flash_Protect4		=24’hFF008A (Flash 地址)
Flash_Protect5		=24’hFF008B (Flash 地址)
Flash_Protect6		=24’hFF008C (Flash 地址)
Flash_Protect7		=24’hFF008D (Flash 地址)

Figure 10-20 Flash页锁存区地址

10.21 Flash 控制寄存器(Flash_CTL)

Flash_CTL								
Flash 控制寄存器								
地址空间： S:0x91H								
位	7	6	5	4	3	2	1	0
符号	error_flag	--	--	ef_busy	--	--	ef_fun[1]	ef_fun[0]
类型	R	R	R	R	R	R	R/W	R/W
复位值	0	0	0	0	0	0	0	0

位	符号	描述
[7]	error_flag	Flash出错标志，硬件置“1”，软件清“0” （1）当对锁存住的page进行编程或擦除时，error_flag被硬件置“1”。 （2）对于程序执行时的当前page进行编程或页擦除，无论当前page是否锁存，error_flag被硬件置“1”。 （3）对于程序在Flash中执行时，进行整片页擦除，error_flag被硬件置“1”。
[6]	--	保留位
[5]	--	保留位
[4]	ef_busy	Flash忙指示位 0：Flash处于空闲状态，PC指令运行，CPU正常运行 1：Flash处于忙状态，PC固定，CPU被暂停运行 这一位由硬件自动控制，指示Flash处于编程或擦除状态。
[3]	--	保留位
[2]	--	保留位
[1:0]	ef_fun[1:0]	Flash 控制选择位 00：读 01：编程写 10：页擦除 11：整片页擦除 关于页擦除与整片擦除，Flash控制器接收到一个命令之后，自动查找相应的页面进行页擦除，指令写的数据属于冗余数据，没有任何意义。比如： ef_fun = 01；写FF:0123地址，数据为0xABCDh，那么这个数据是有效的，完成之后，FF:0123的数据就会变成0xABCDh。 ef_fun=10，11时，写FF:0123地址，数据为0xABCDh，那么这数据是无意义的，Flash控制器只会选择page0的擦除或是整片页擦除。

Figure 10-21 Flash控制寄存器

10.22 Flash 页加解锁控制器(Flash_PL)

Flash_PL								
Flash 页加解锁控制器								
地址空间： S:0xB1H								
位	7	6	5	4	3	2	1	0
符号	--	--	--	--	--	seq_error	seq2	seq1
类型	R	R	R	R	R	R/W	R/W	R/W
复位值	0	0	0	0	0	0	0	0

位	符号	描述
[7]	--	保留位
[6]	--	保留位
[5]	--	保留位
[4]	--	保留位
[3]	--	保留位
[2]	seq_error	解Flash锁存序列错误标志： 1：一旦0x5A ➔ 0xA5的序列出错，硬件自动置“1”。 0：只有reset复位才能清“0”。 （1）一旦0x5A ➔ 0xA5的加解锁序列出错，那么整个加解锁功能就会关闭，无论以后是否加解锁序列正确，都将关闭加解锁功能，只有芯片复位后，才能继续执行加解锁功能。 （2）每次0x5A ➔ 0xA5的正确解锁序列之后，只能进行一次加解锁功能操作，即每次加解锁系列之后，只能操作一次EF_pagelockx，最多进行这个EF_pagelockx的内部8个页面的加解锁。如果要对不同的EF_pagelockx进行操作，每次操作前必须重复0x5A ➔ 0xA5序列。
[1]	Seq2	序列标志2： 1：0x5A ➔ 0xA5 加解锁序列正确 0：0x5A ➔ 0xA5 加解锁的第二步还未执行，或者执行错误。 需配合seq_error来一起观察。
[0]	Seq1	序列标志1： 1：0x5A正确写入Flash_PL寄存器中 0：0x5A还未写入Flash_PL寄存器中 需要配合seq_error来一起观察。

Figure 10-22 Flash页加解锁控制器

11 DMA 控制器

11.1 DMA 简介

直接存储器存取(Direct Memory Access ： DMA)用来提供在外设和存储器之间或者存储器和存储器之间的高速数据传输。无须CPU干预，数据可以通过DMA快速地移动，节省了CPU的负载以及降低系统功耗。

HC16Lxx DMA控制器特点：

- DMA控制器是一个可以被访问的内存地址映射的外设。
- 源地址和目标地址支持3种寻址模式，包括增量地址模式，减量地址模式和固定地址模式。
- 支持3种时序模式的读写操作，包括并行模式，串行模式和串行等待模式。
- 最大支持64K字的传输长度。HC16Lxx Flash为16K字。
- 支持16位数据宽度。

有三个计时模式，分别是并行模式，串行模式和串行等待模式。

- 并行模式：它是一个同步快速的方式，可以应用于两种不同的存储区域之间的数据传输。写和读发生在同一时间。
- 串行模式：它是时间共享的方式，数据在一个周期中被获取，并在下一个周期中被写入，并且可以在相同的内存区域内进行数据传输。
- 串行等待模式：除了在写入期间时序不同，其他与串行模式相同，DMA不会继续读下个字，直到当前地址的Flash写入完成，因为Flash的写操作需要较长时间（20us ~ 40us）。

注意事项：启动DMA操作时，系统数据总线被DMA控制器所控制，在这个阶段CPU停止工作，直至DMA完成数据总线上的操作。这期间的外设仍可继续工作，但中断亦被锁存住，直到DMA完成操作，中断才能被CPU响应。

11.2 DMA 工作模式

11.3 DMA 并行操作的读写时序

RAM 到 CRC，Flash 到 RAM，Flash 到 CRC，可选择并行操作

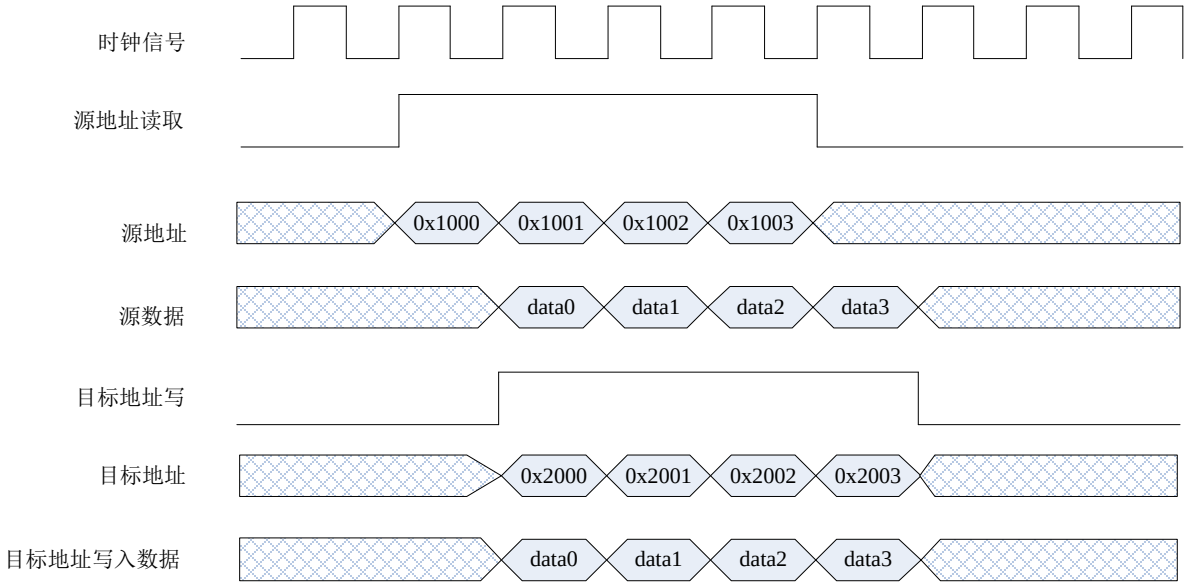


Figure 11-1 DMA 并行操作时序图

11.4 DMA 串行操作的读写时序

RAM 到RAM的数据搬运可选择串行操作

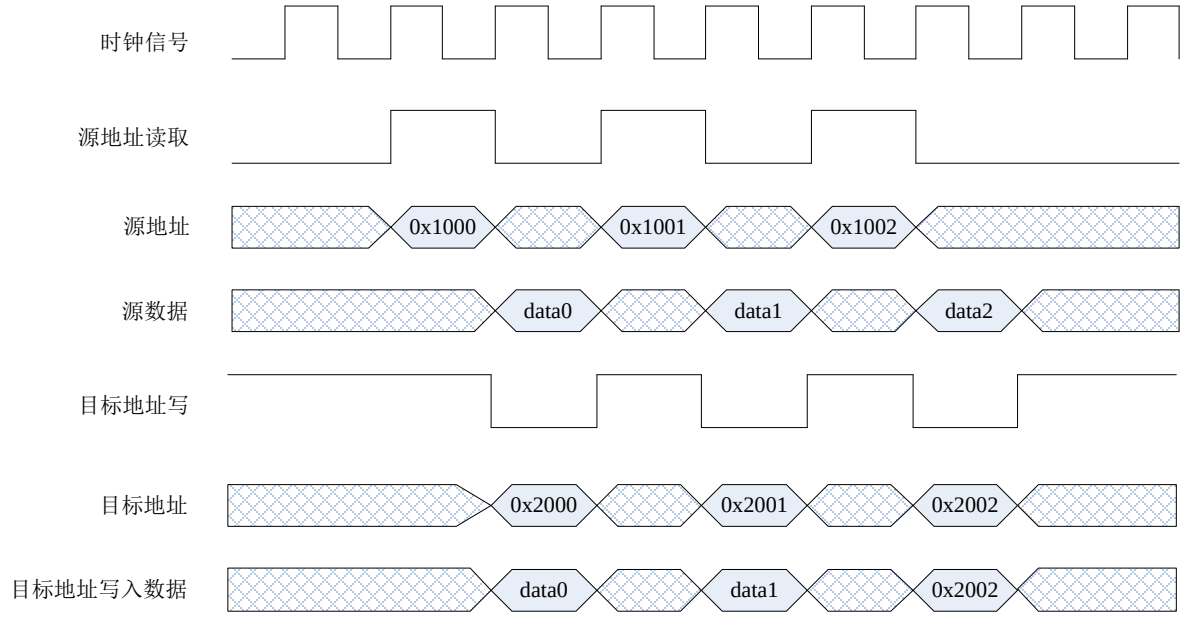


Figure 11-2 DMA 串行操作时序图

11.5 DMA 串行等待模式

RAM 到Flash， Flash 到 Flash 可以选择串行等待模式

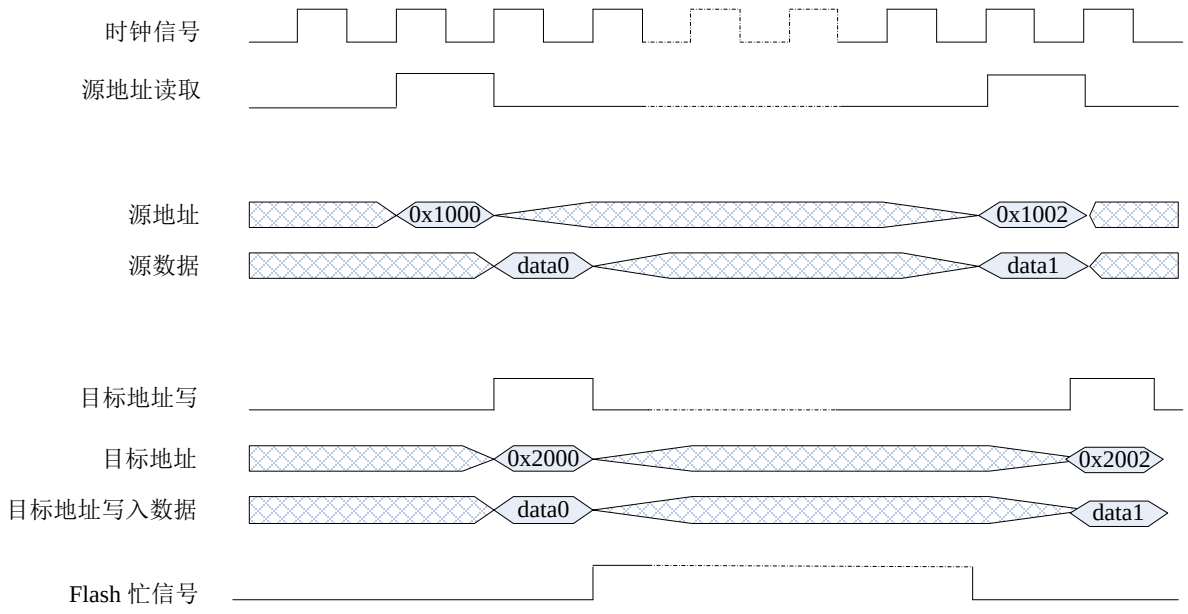


Figure 11-3 DMA 串行等待Flash时序模式图

11.6 用户软件配置

（1） 在DMA_CTL中配置传输模式，地址模式

（2） 配置源地址，目标地址，传输范围

（3） 把DMA_EN值设置成“1”，使能DMA传输

（4） 读到DMA_EN的值为0，表示DMA传输结束

11.7 DMA 控制器相关寄存器

11.8 DMA 控制寄存器(DMA_CTL)

DMA_CTL								
DMA控制寄存器								
地址空间： 0x00F900								
位	7	6	5	4	3	2	1	0
符号	--	DMA_Mode		DST_Addr_Mode		SRC_Addr_Mode		DMA_EN
类型	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	0	0	0	0	0	0

位	符号	描述
[7]	--	保留位
[6:5]	DMA_ Mode	00:并行操作模式，针对于 RAM ➔ CRC Flash ➔ RAM Flash ➔ CRC 01:串行等待操作模式，针对于 RAM ➔ Flash Flash ➔ Flash 10: 串行模式，针对于 RAM ➔ RAM 11: 保留位
[4:3]	DST_Add r_Mode	目标地址模式选择 00：自动地址加 1 模式 01：自动地址减 1 模式 10：地址固定模式 11：保留位
[2:1]	SRC_Add r_Mode	源地址模式选择 00：自动地址加 1 模式 01：自动地址减 1 模式 10：地址固定模式 11：保留位
[0]	DMA_EN	DMA 使能信号 0：DMA 无效 1：DMA 使能 这个 DMA 控制信号，由用户软件置“1”，当 DMA 完成相应的操作后，硬件自动清“0”。

Figure 11-4 DMA_CTL 寄存器

11.9 DMA 源地址低 8 位寄存器(DMA_SRC_ADDR0)

DMA_SRC_ADDR0								
DMA源地址低8位寄存器								
地址空间： 0x00F902								
位	7	6	5	4	3	2	1	0
符号	DMA_SRC_ADDR0							
类型	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	0	0	0	0	0	0
位	符号	描述						
[7:0]	DMA_SRC_ADDR0	DMA 源地址低 8 位 DMA_SRC_ADDR[7:0]						

Figure 11-5 DMA_SRC_ADDR0 寄存器

11.10 DMA 源地址中 8 位寄存器(DMA_SRC_ADDR1)

DMA_SRC_ADDR1								
DMA源地址中8位寄存器								
地址空间： 0x00F904								
位	7	6	5	4	3	2	1	0
符号	DMA_SRC_ADDR1							
类型	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	0	0	0	0	0	0
位	符号	描述						
[7:0]	DMA_SRC_ADDR1	DMA 源地址中 8 位 DMA_SRC_ADDR[15:8]						

Figure 11-6 DMA_SRC_ADDR1 寄存器

11.11 DMA 源地址高 8 位寄存器(DMA_SRC_ADDR2)

DMA_SRC_ADDR2								
DMA源地址高8位寄存器								
地址空间： 0x00F906								
位	7	6	5	4	3	2	1	0
符号	DMA_SRC_ADDR2							
类型	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	0	0	0	0	0	0
位	符号	描述						
[7:0]	DMA_SRC_ADDR2	DMA 源地址高 8 位 DMA_SRC_ADDR[23:16]						

Figure 11-7 DMA_SRC_ADDR2 寄存器

11.12 DMA 目标地址低 8 位寄存器(DMA_DST_ADDR0)

DMA_DST_ADDR0								
DMA目标地址低8位寄存器								
地址空间： 0x00F908								
位	7	6	5	4	3	2	1	0
符号	DMA_DST_ADDR0							
类型	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	0	0	0	0	0	0
位	符号	描述						
[7:0]	DMA_DST_ADDR0	DMA 目标地址低 8 位 DMA_DST_ADDR[7:0]						

Figure 11-8 DMA_DST_ADDR0 寄存器

11.13 DMA 目标地址中 8 位寄存器(DMA_DST_ADDR1)

DMA_DST_ADDR1								
DMA目标地址中8位寄存器								
地址空间： 0x00F90A								
位	7	6	5	4	3	2	1	0
符号	DMA_DST_ADDR1							
类型	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	0	0	0	0	0	0
位	符号	描述						
[7:0]	DMA_DST_ADDR1	DMA 目标地址中 8 位 DMA_DST_ADDR[15:8]						

Figure 11-9 DMA_DST_ADDR1 寄存器

11.14 DMA 目标地址高 8 位寄存器(DMA_DST_ADDR2)

DMA_DST_ADDR2								
DMA目标地址高8位寄存器								
地址空间： 0x00F90C								
位	7	6	5	4	3	2	1	0
符号	DMA_DST_ADDR2							
类型	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	0	0	0	0	0	0
位	符号	描述						
[7:0]	DMA_DST_ADDR2	DMA 目标地址高 8 位 DMA_DST_ADDR[23:16]						

Figure 11-10 DMA_DST_ADDR2 寄存器

11.15 DMA 传输地址范围低 8 位寄存器(DMA_TRANS_CNT0)

DMA_TRANS_CNT0								
DMA传输地址范围低8位寄存器								
地址空间： 0x00F90E								
位	7	6	5	4	3	2	1	0
符号	DMA_TRANS_CNT0							
类型	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	0	0	0	0	0	0
位	符号	描述						
[7:0]	DMA_TRANS_CNT0	DMA 传输地址范围低 8 位 DMA_TRANS_CNT [7:0]						

Figure 11-11 DMA_TRANS_CNT0 寄存器

11.16 DMA 传输地址范围高 8 位寄存器(DMA_TRANS_CNT1)

DMA_TRANS_CNT1								
DMA传输地址范围高8位寄存器								
地址空间： 0x00F910								
位	7	6	5	4	3	2	1	0
符号	DMA_TRANS_CNT1							
类型	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	0	0	0	0	0	0
位	符号	描述						
[7:0]	DMA_TRANS_CNT1	DMA 传输地址范围高 8 位 DMA_TRANS_CNT [15:8]						

Figure 11-12 DMA_TRANS_CNT1 寄存器

12 工作模式

HC16Lxx工作电压为1.8v - 3.8v，由内建LDO产生内部1.8v工作电压。

HC16Lxx有3种主要工作模式：

- 工作模式（Active mode）

  CPU以及外围模块都工作，时钟来自主频(内部16MHz时钟)或副频（外部32K晶振）

- 空闲模式(Idle mode)

  CPU停止工作，外围模块工作状态

- 低功耗模式(Low power mode)

  CPU以及大部分外围模块停止工作，仅RTC/WDT/LCD 3个外围模块工作

	工作模式	空闲模式	低功耗模式
16M 内部时钟	v	v	–
32K 外部时钟	v	v	v
看门狗专用时钟	v	v	v
CPU 时钟来源	16M/32K	–	–
实时时钟	v	v	v
ADC/TS	v	v	–
UART0/1	v	v	–
电压比较	v	v	v
定时器 0/1/2/3	v	v	–
SPI	v	v	–
ISO7816 master	v	v	–
I^2C	v	v	–
POR/LVR	v	v	v
低电压检测模块	v	v	v
Reset pad	v	v	v
外部端口	v	v	v
可编程计数器阵列	v	v	–
看门狗	v	v	v
随机数产生模块	v	v	–
CRC	v	v	–
DES 协处理器	v	v	–
LCD	v	v	v
JTAG	v	–	–

Figure 12-1 周边功能模块以及工作模式

12.1 空闲模式

HC16Lxx提供2种低功耗的模式，Idle（空闲模式）和LPM（低功耗模式）。用户可以通过设置PCON[0]进入Idle（空闲模式），以及LPM[1:0]进入LPM（低功耗模式）。

空闲模式时，CPU内核停止工作，但允许CPU通过中断被唤醒。在这种模式下，程序停止执行，程序计数器PC，状态寄存器，RAM，SFR的内容都被保留，输出端口保持输出状态不变。

用户通过设置PCON寄存器IDL位，可以进入空闲模式，设置IDL位的指令是在最后一条指令执行。离开空闲模式和中断处理之前，最多有两个时钟周期，为了防止意外，IDLE指令之后需放置两条NOP指令。

退出空闲模式有两种方法：

- 中断

  启动中断使能产生中断，硬件清除PCON中的IDL位，CPU恢复时钟，将继续执行中断服务程序。

- 复位

  每个复位信号都可以退出空闲模式，同时硬件清除PCON中的IDL位。CPU恢复时钟，PC指针向量会被复位初始化到FF:0000h。

12.2 低功耗模式

主时钟停止工作，程序停止执行，程序计数器PC，状态寄存器，RAM， SFR的内容都被保留，输出端口保持输出状态不变。

在LPM模式下，LDO的驱动能力会降低。

退出低功耗模式有两种方法：

- 中断

  启动中断使能，产生中断，硬件清除LPM[1]位，CPU恢复时钟，继续执行中断服务程序。端口中断，RTC中断，LVD中断和VC中断都可以把MCU从低功耗模式唤醒。

- 复位

  各个复位信号（Reset端口，上电掉电复位POR，PCA复位，WDT复位，LVD复位）都可以退出低功耗模式，同时硬件清除LPM[1]。CPU恢复时钟，PC指针向量会被复位初始化到FF:0000h。

为了防止意外的行为，进入LPM指令之后放置两条NOP指令。

12.3 工作模式寄存器

12.4 电源控制寄存器（PCON）

电源控制寄存器								
地址空间： S:87h								
位	7	6	5	4	3	2	1	0
符号	SMOD01	SMOD00	--	--	SMOD11	SMOD10	--	IDL
类型	R/W	R/W	保留位	保留位	R/W	R/W	保留位	R/W
复位值	0	0	0	0	0	0	0	0

位	符号	描述
7	SMOD01	UART0 双倍波特率设置 1：双倍波特率 0：正常波特率
6	SMOD00	UART0 中 SCON[7]功能选择 1：SCON[7]作为 FE 位 （frame error） 0：SCON[7]作为 SM0 位 （具体查看 UART 一章）
5:4	--	保留位
3	SMOD11	UART1 双倍波特率设置 1：双倍波特率 0：正常波特率
2	SMOD10	UART1 中 SCON1[7]功能选择 1：SCON1[7]作为 FE 位（frame error） 0：SCON1[7]作为 SM0 位 （具体查看 UART 一章）
1	--	保留位
0	IDL	Idle 空闲模式设置位 1：CPU 进入 Idle 模式，CPU 停止工作 0：当外部一个中断或复位发生时，被硬件自动清零，退出空闲模式。

Figure 12-2 电源控制寄存器

12.5 低功耗模式控制寄存器（LPM）

低功耗模式控制寄存器								
地址空间： S:95h								
位	7	6	5	4	3	2	1	0
符号	--	--	--	--	--	--	LPM	LPMEN
类型	--	--	--	--	--	--	R/W	R/W
复位值	0	0	0	0	0	0	0	0

位	符号	描述
7:2	--	保留位
1	LPM	低功耗进入位 1：进入低功耗模式，主时钟关闭，LDO 驱动降低。 0：正常模式。
0	LPMEN	低功耗进入使能位 1：允许进入低功耗模式。 0：不允许进入低功耗模式。 用户必须必须先把 LPM[0]置 1，然后再把 LPM[1]置 1，MCU 才被允许进入低功耗模式。

Figure 12-3 低功耗模式控制寄存器

13 时钟控制器

13.1 介绍

时钟控制模块主要控制时钟以及晶振。HC16Lxx支持2个不同的时钟，外部32.768KHz晶振和内部16M晶振。时钟控制器可以配置不同的时钟源，系统分频时钟，启动或禁用外围设备时钟。

两种时钟源可以当系统时钟:内部高速RC晶振，外部32.768KHz时钟晶振。

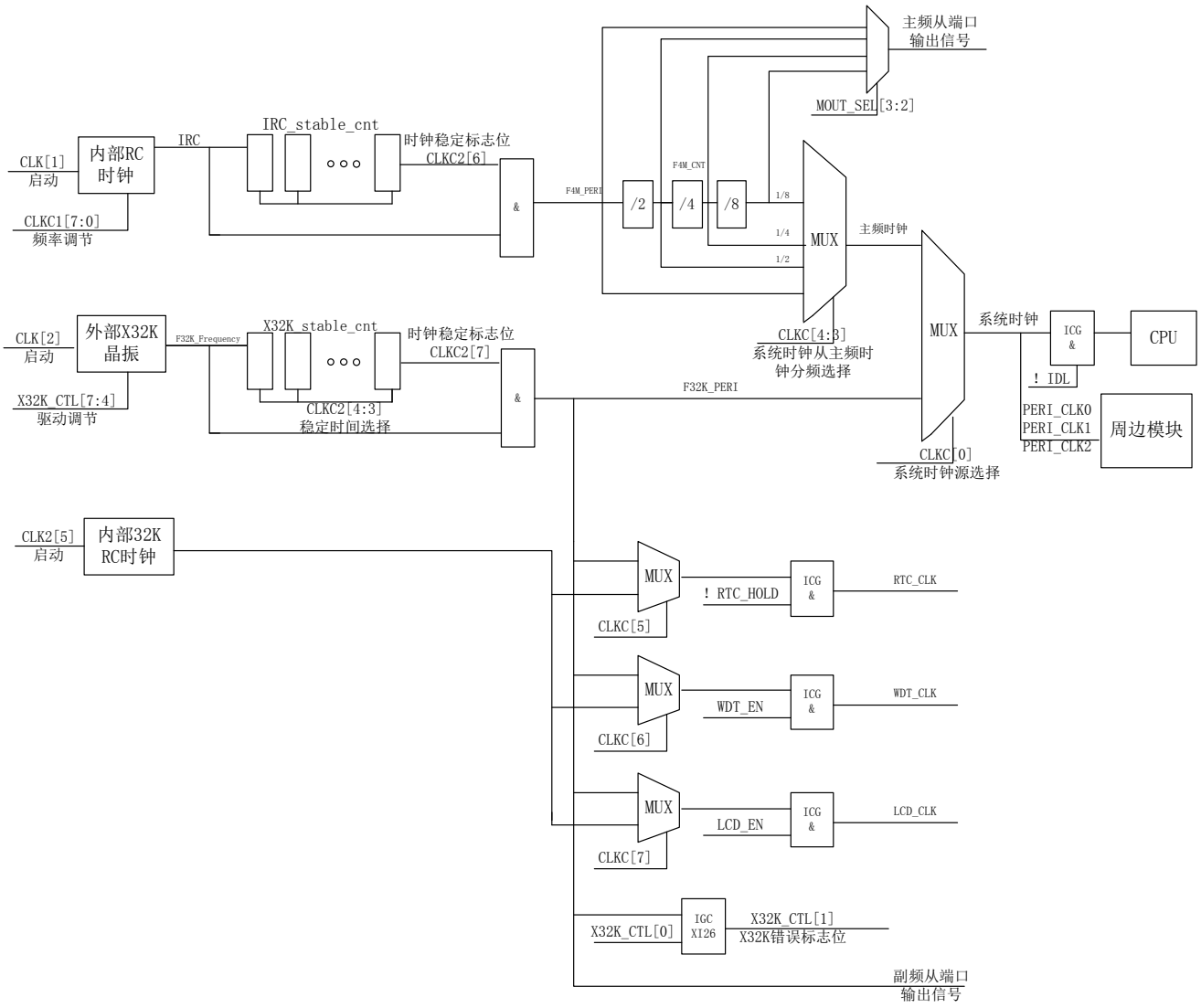


Figure 13-1 HC16Lxx 时钟分配图

13.2 内部高速时钟

主时钟信号是从内部RC OSC产生的，这个频率可以通过CLKC1来调整，出厂后会给出精确 2M, 4M, 8M, 16M的校准配置。系统时钟可以通过调整CLKC[4:3]来选择内部高速时钟的1分频，2分频，4分频，8分频。内部高速时钟在芯片上电或复位后即开始工作。这个高速时钟可以当成唤醒低功耗模式的时钟。唤醒时间是3us，系统可以快速响应外部中断。

13.3 内部 32K 时钟

此内部时钟主要为WDT提供稳定时钟信号。通常，当外部X32K时钟处于异常状态，如高温高湿、酸碱腐蚀，外部X32K无法工作时，为了防止程序失效，WDT需要一个在各种状态下可以稳定工作，低功耗，但不要求精准的时钟来源。这就是内部32K时钟的主要作用。

同时用户可以为了节省成本，省去外部精准X32K晶振，而使用这个内部32K时钟进行RTC计时，WDT计数，LCD显示等功能。

13.4 外部晶振时钟

外部晶振时钟外接一个32.768kHz的低功耗晶振，具有超高精度以及低功耗的特征，低功耗模式下的工作模块都是以此时钟源当时钟信号。

13.5 系统时钟切换

共有两种时钟源可以当系统的时钟：

- 内部RC OSC （主频）
- 外部32.768KHz OSC （副频）

时钟源的选择由CLKC[0]配置。在双时钟模式下，系统时钟由主频时钟产生，如有需要，系统时钟可由副频时钟产生。当主副时钟进行切换时，必须按照流程来实现，否则会出现异常。不能同时进行副频启动，主副频切换和主频禁用这三种操作。

主频 ➔ 副频，CLKC[2:0]的配置为：

1. 011B （主频工作，系统时钟选择主频时钟）
2. 111B （主频工作，副频启动，系统时钟选择主频时钟）
3. 等待副频稳定
4. 110B （主频工作，副频工作，系统切换）
5. 100B （主频禁用，副频工作，系统时钟选择副频时钟）

副频 ➔ 主频，CLKC[2:0]的配置为：

1. 100B （主频禁用，副频工作，系统时钟选择副频时钟）
2. 110B （主频工作，副频启动，系统时钟选择副频时钟）
3. 等待主频稳定
4. 111B （主频工作，副频工作，系统切换）
5. 011B （主频工作，副频禁用，系统时钟选择主频时钟）

13.6 蜂鸣器

HC16Lxx提供可调频率输出。

蜂鸣器的频率可从主频时钟而来，亦可选择副频时钟而来。

$$Buzz\_CLK = \frac{CLK\_source}{2 * (256-CNT)}$$

具体蜂鸣器时钟分频如下：

时钟源频率	32K	2M	4M	8M	16M
蜂鸣器时钟源（CLK_source）	8K	4K	8K	16K	32K
	蜂鸣器输出时钟频率（Buzz_CLK）				
CNT = FFh	4K	2K	4K	8K	16K
CNT = FEh	2K	1K	2K	4K	8K
CNT = FDh	1.33K	667	1.33K	2.66K	5.33K
CNT = FCh	1K	512	1K	2K	4K
CNT = FBh	800	400	800	1.6K	3.2K
CNT = FAh	667	333	667	1.33k	2.66k
CNT = F9h	571	286	571	1.14k	2.28k
CNT = F8h	500	250	500	1k	2k
… …	… …	… …	… …	… …	… …
CNT = 00h	15.6	7.8	15.6	31	62

Figure 13-2 蜂鸣器时钟分频表

13.7 时钟控制寄存器

13.8 时钟控制寄存器（CLKC）

CLKC								
时钟控制寄存器								
地址空间： S:92h								
位	7	6	5	4	3	2	1	0
符号	LCDCS	WDTCS	RTCCS	SYSCS1	SYSCS0	X32KEN	IRCEN	CLKSS
类型	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	0	0	0	0	1	1
位	符号	描述						
7	LCDCS	LCD 时钟选择 1：LCD 时钟来源为内部 32K RC OSC 0：LCD 时钟来源为外部 32K 晶振						
6	WDTCS	WDT 时钟选择 1：WDT 时钟来源为内部 32K RC OSC 0：WDT 时钟来源为外部 32K 晶振						
5	RTCCS	RTC 时钟选择 1：RTC 时钟来源为内部 32K RC OSC 0：RTC 时钟来源为外部 32K 晶振						
4:3	SYSCS [1:0]	系统时钟从主频时钟分频选择 00：系统时钟为主频时钟 01：系统时钟为主频时钟的 1/2 分频 10：系统时钟为主频时钟的 1/4 分频 11：系统时钟为主频时钟的 1/8 分频						
2	X32KEN	外部 32.768kHz 晶振使能位 1：外部 32.768kHz 晶振时钟使能 0：外部 32.768kHz 晶振时钟关闭 注意事项： 当 CPU 系统主频运行在 X32K 的副频时，程序无法关闭 X32KEN，硬件会自动防止这种情况以防芯片丢失时钟而失效						
1	IRCEN	内部 RC OSC 时钟使能位 1：内部 RC OSC 时钟使能 0：内部 RC OSC 时钟关闭 注意事项： 当 CPU 系统主频运行在内部主频下，程序无法关闭 IRCEN，硬件会自动防止这种情况以防芯片丢失时钟而失效						
0	CLKSS	系统时钟时钟源选择 1：系统时钟源来源为主频高速 RC 时钟 0：系统时钟源来源为副频低速 X32K 时钟						

Figure 13-3 时钟控制寄存器（CLKC）

13.9 主时钟控制寄存器（CLKC1）

CLKC1								
主时钟控制寄存器								
地址空间： S:93h								
位	7	6	5	4	3	2	1	0
符号	--		--					
类型	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	1	1	0	0	0	0	0

位	符号	描述
7：6	--	主频时钟频率选择 00：2M 01：4M 10：8M 11：16M
5:0	--	主频时钟频率调整 111111：最高频率 … … 000000：最低频率 注意事项： 出厂时，频率保留在校准寄存器中，用户只要读 CAL0,CAL1 寄存器中的值，写入 CLKC1 即可配置精准主频时钟。 用户亦可通过写入不同的值选择不同的频率，范围从 1.5MHz－22 MHz。

Figure 13-4 主时钟控制寄存器（CLKC1）

13.10 时钟稳定控制寄存器（CLKC2）

CLKC2								
时钟稳定控制寄存器								
地址空间： S:94h								
位	7	6	5	4	3	2	1	0
符号	X32K_ Stable	IRC_ Stable	IRC32K _EN	X32K_stable _sel[1:0]		－－		
类型	R	R	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	1	0	1	0	0	0	1

位	符号	描述
7	X32K_ Stable	X32K 稳定信号标志位 1：X32K 时钟已稳定 0：X32K 未使能，或 X32K 使能后未稳定
6	IRC_ Stable	内部主频稳定信号标志位 1：内部主频 RC OSC 时钟已稳定 0：内部主频未使能，或内部主频使能后未稳定
5	IRC32K_ EN	内部慢速 32K RC OSC 使能 1：内部慢速 32K RC OSC 使能 0：内部慢速 32K RC OSC 停止
4:3	X32K_st able_se l[1:0]	外部 X32K 稳定时间选择 00：外部稳定时间为 256 个外部 X32K 时钟，约 7ms 01：外部稳定时间为 1024 个外部 X32K 时钟，约 30ms 10：外部稳定时间为 4096 个外部 X32K 时钟，约 125ms（默认） 11：外部稳定时间为 16384 个外部 X32K 时钟，约 500ms
2：0	－－	保留值

Figure 13-5 时钟稳定控制寄存器（CLKC2）

13.11 X32K 时钟控制寄存器（X32K_CTL）

X32K_CTL								
X32K时钟控制寄存器								
地址空间： S:CFh								
位	7	6	5	4	3	2	1	0
符号	X32K_BiasSel				MOUT_SEL		X32K_fault_flag	X32K_fault_enable
类型	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	1	0	1	0	0	0	0	0

位	符号	描述
7:4	X32K_BiasSel	外部 X32K 晶振驱动选择 0000：最小晶振驱动能力 ~ 1100：最大晶振驱动能力 默认值为 1010,推荐值为 1010。 注意事项： 1111：抗干扰能力最强，但功耗最高 0000：抗干扰能力最弱，但功耗最低
3:2	MOUT_SEL	主频从端口输出频率选择 00：端口输出频率关闭 01：端口输出频率为主频 10：端口输出频率为 1/2 主频 11：端口输出频率为 1/4 主频
1	X32K_fault_flag	外部 X32K 故障标志位 1：外部 X32K 处于故障状态，无法正常工作 0：外部 X32K 正常工作状态中 注意事项： 这个信号会触发 NMI 中断。一旦故障出现，被硬件置“1”，只有通过软件才能清除。即使 X32K 恢复正常工作，这个标志位还会被保留。如果软件清除这个标志位后，X32K 的故障依然存在，这个标志位即会马上被硬件置“1”。
0	X32K_fault_enable	外部 X32K 故障报警 1：启动外部 X32K 故障报警 0：关闭外部 X32K 故障报警 在启动 X32K 故障报警时，必须启动内部主频时钟以便检查 X32K 的输出波形，当主频工作 4096 个时钟内，X32K 还没有频率输出时，就会触发 X32K 故障报警。 比如主频时钟为 4MHz，4096 个时钟就是 1ms，如果 X32K 在 1ms 即 32 个 cycle 没有正常工作，就会触发 X32K 故障报警

Figure 13-6 X32K 时钟控制寄存器（X32K_CTL）

13.12 蜂鸣器控制寄存器（Buzz_CTL）

Buzz_CTL								
蜂鸣器控制寄存器								
地址空间： S:A2h								
位	7	6	5	4	3	2	1	0
符号	--	--	--	--	--	--	BUZ_CS	BUZ_RUN
类型	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	0	0	0	0	0	0

位	符号	描述
7:2	--	保留位
1	BUZ_CS	蜂鸣器时钟来源选择 1：时钟来源为外部 X32K 副频 0：时钟来源为内部主频的 512 分频 比如： 主频选择 2M，蜂鸣器来源为 4K 主频选择 4M，蜂鸣器来源为 8K 主频选择 8M，蜂鸣器来源为 16K 主频选择 16M，蜂鸣器来源为 32K
0	BUZ_RUN	蜂鸣器启动 1：蜂鸣器使能 0：蜂鸣器禁用

Figure 13-7 蜂鸣器控制寄存器（Buzz_CTL）

13.13 蜂鸣器计数控制寄存器（Buzz_CNT）

Buzz_CNT								
蜂鸣器计数控制寄存器								
地址空间： S:A3h								
位	7	6	5	4	3	2	1	0
符号	--							
类型	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	0	0	0	0	0	0

位	符号	描述
7:0	--	蜂鸣器计数器初值

Figure 13-8 蜂鸣器计数控制寄存器（Buzz_CNT）

13.14 校准寄存器 0（Cal0）

Cal0								
校准寄存器0								
地址空间： 0x10230h								
位	15	14	13	12	11	10	9	8
符号	Trim_2M							
类型	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	0	0	0	0	0	0
位	7	6	5	4	3	2	1	0
符号	Trim_4M							
类型	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	0	0	0	0	0	0

位	符号	描述
15:8	Trim_2M	内部 2M RC OSC 调整值
7:0	Trim_4M	内部 4M RC OSC 调整值

Figure 13-9 校准寄存器 0（Cal0）

13.15 校准寄存器 1（Cal1）

Cal1								
校准寄存器1								
地址空间： 0x10232h								
位	15	14	13	12	11	10	9	8
符号	Trim_8M							
类型	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	0	0	0	0	0	0
位	7	6	5	4	3	2	1	0
符号	Trim_16M							
类型	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	0	0	0	0	0	0

位	符号	描述
15:8	Trim_8M	内部 8M RC OSC 调整值
7:0	Trim_16M	内部 16M RC OSC 调整值

Figure 13-20 校准寄存器 1(Cal1)

13.16 校准寄存器 2（Cal2）

Cal2								
校准寄存器2								
地址空间： 0x10234h								
位	－－	－－	－－	－－	11	10	9	8
符号	--							
类型	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	0	0	0	0	0	0
位	7	6	5	4	3	2	1	0
符号	--							
类型	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	0	0	0	0	0	0

位	符号	描述
15:12	--	
11:0	--	温度传感器常温下 ADC 使用 2.5v 内部参考电压采样数据。

Figure 13-31 校准寄存器 2（Cal2）

13.17 校准寄存器 3（Cal3）

Cal3								
校准寄存器3								
地址空间： 0x10236h								
位	———	———	———	———	———	———	———	———
符号	--							
类型	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	0	0	0	0	0	0
位	———	———	5	4	3	———	———	———
符号	--		ADC_Trim[2:0]			--		
类型	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	0	0	0	0	0	0

位	符号	描述
15:6	———	保留值
5:3	ADC_Trim [2:0]	ADC 内部 2.5v 参考电压微调，从 CAL3 寄存器读取出厂测试结果，写入 ADC3 以得到最优化值。 111：最高参考电压 ~ 000：最低参考电压
2:0	———	保留值

Figure 13-42 校准寄存器 3（Cal3）

13.18 校准寄存器 4（Cal4）

Cal4								
校准寄存器4								
地址空间： 0x10238h								
位	15	14	13	12	11	10	9	8
符号	--							
类型	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	0	0	0	0	0	0
位	7	6	5	4	3	2	1	0
符号	--							
类型	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	0	0	0	0	0	0
位	符号	描述						
15:8	--	唯一 ID 识别号[15:8]						
7:0	--	唯一 ID 识别号[7:0]						

Figure 13-53 校准寄存器 4（Cal4）

13.19 校准寄存器 5（Cal5）

Cal5								
校准寄存器5								
地址空间： 0x1023Ah								
位	15	14	13	12	11	10	9	8
符号	--							
类型	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	0	0	0	0	0	0
位	7	6	5	4	3	2	1	0
符号	--							
类型	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	0	0	0	0	0	0
位	符号	描述						
15:8	--	唯一 ID 识别号[31:24]						
7:0	--	唯一 ID 识别号[23:16]						

Figure 13-64 校准寄存器 5（Cal5）

13.20 校准寄存器 6（Cal6）

Cal6								
校准寄存器6								
地址空间： 0x1023Ch								
位	15	14	13	12	11	10	9	8
符号	--							
类型	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	0	0	0	0	0	0
位	7	6	5	4	3	2	1	0
符号	--							
类型	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	0	0	0	0	0	0

位	符号	描述
15:8	--	唯一 ID 识别号[47:40]
7:0	--	唯一 ID 识别号[39：32]

Figure 13-15 校准寄存器 6（Cal6）

13.21 校准寄存器 7（Cal7）

Cal7								
校准寄存器7								
地址空间： 0x1023Eh								
位	15	14	13	12	11	10	9	8
符号	--							
类型	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	0	0	0	0	0	0
位	7	6	5	4	3	2	1	0
符号	--							
类型	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	0	0	0	0	0	0

位	符号	描述
15:8	--	唯一 ID 识别号[63:56]
7:0	--	唯一 ID 识别号[55:48]

Figure 13-16 校准寄存器 7（Cal7）

13.22 周围功能模块时钟控制寄存器 0（PERI_CLK0）

PERI_CLK0								
周围功能模块时钟控制寄存器0								
地址空间： S:B3h								
位	7	6	5	4	3	2	1	0
符号	--	--	--	VC_CLKEN	LVD_CLKEN	BGR_CLKEN	ADC_CLKEN	LCD_CLKEN
类型	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	1	1	1	1	1	1	1	1

位	符号	描述
7:5	--	保留位
4	VC_CLKEN	电压比较器时钟使能 1：电压比较器有时钟输入 0：电压比较器无时钟输入
3	LVD_CLKEN	低电压侦测器时钟使能 1：低电压侦测器有时钟输入 0：低电压侦测器无时钟输入
2	BGR_CLKEN	BGR 时钟使能，BGR 产生内部参考电压，供给 ADC，VC，LVD 1：BGR 有时钟输入 0：BGR 无时钟输入
1	ADC_CLKEN	ADC 时钟使能 1：ADC 有时钟输入 0：ADC 无时钟输入
0	LCD_CLKEN	LCD 时钟使能 1：LCD 有时钟输入 0：LCD 无时钟输入

Figure 13-17 周围功能模块时钟控制寄存器 0（PERI_CLK0）

注意事项：

当系统不需要某个模块工作时，可以关闭进入这个功能模块的时钟，以节省系统功耗。

13.23 周围功能模块时钟控制寄存器 1（PERI_CLK1）

PERI_CLK1								
周围功能模块时钟控制寄存器1								
地址空间： S:B4h								
位	7	6	5	4	3	2	1	0
符号	--	SPI_ CLKEN	DES_ CLKEN	RNG_ CLKEN	CRC_ CLKEN	DMA_ CLKEN	RTC_ CLKEN	Flash_ CLKEN
类型	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	1	1	1	1	1	1	1	1

位	符号	描述
7	--	保留位
6	SPI_CLKEN	SPI 时钟使能 1：SPI 有时钟输入 0：SPI 无时钟输入
5	DES_CLKEN	DES 时钟使能 1：DES 有时钟输入 0：DES 无时钟输入
4	RNG_CLKEN	RNG 时钟使能 1：RNG 有时钟输入 0：RNG 无时钟输入
3	CRC_CLKEN	CRC 时钟使能 1：CRC 有时钟输入 0：CRC 无时钟输入
2	DMA_CLKEN	DMA 时钟使能 1：DMA 有时钟输入 0：DMA 无时钟输入
1	RTC_CLKEN	实时时钟时钟使能 1：实时时钟有时钟输入 0：实时时钟无时钟输入
0	Flash_CLKEN	Flash 时钟使能 1：Flash 有时钟输入 0：Flash 无时钟输入 注意事项： 在配置 Flash 控制寄存器时才需要 Flash 时钟使能，其余在读模式下，Flash 时钟使能可以关闭。

Figure 13-18 周围功能模块时钟控制寄存器 1（PERI_CLK1）

13.24 周围功能模块时钟控制寄存器 2（PERI_CLK2）

PERI_CLK2								
周围功能模块时钟控制寄存器2								
地址空间： S:B5h								
位	7	6	5	4	3	2	1	0
符号	7816_ CLKEN	I2C_ CLKEN	UART0_ CLKEN	UART1_ CLKEN	Timer_ CLKEN	PCA_ CLKEN	WDT_ CLKEN	PORT_ CLKEN
类型	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	1	1	1	1	1	1	1	1

位	符号	描述
7	7816_CLKEN	7816 时钟使能 1：7816 有时钟输入 0：7816 无时钟输入
6	I2C_CLKEN	I2C 时钟使能 1：I2C 有时钟输入 0：I2C 无时钟输入
5	UART0_CLKEN	UART0 时钟使能 1：UART0 有时钟输入 0：UART0 无时钟输入
4	UART1_CLKEN	UART1 时钟使能 1：UART1 有时钟输入 0：UART1 无时钟输入
3	Timer_CLKEN	定时器时钟使能 1：定时器有时钟输入 0：定时器无时钟输入 注意事项： Timer_CLKEN 控制了 4 个计数器模块（定时器 0 - 定时器 3）
2	PCA_CLKEN	PCA 时钟使能 1：PCA 有时钟输入 0：PCA 无时钟输入
1	WDT_CLKEN	WDT 时钟使能 1：WDT 有时钟输入 0：WDT 无时钟输入
0	PORT_CLKEN	端口时钟使能 1：端口有时钟输入 0：端口无时钟输入

Figure 13-19 周围功能模块时钟控制寄存器 2（PERI_CLK2）

14 复位控制器

14.1 概要

HC16Lxx 具有 6 个复位信号来源，每个复位信号可以让 CPU 重新运行，将绝大多数寄存器置为复位值，程序计数器 PC 指向初始地址 FF：0000h。

（1） 上电掉电复位POR

（2） 外部复位管脚（Reset PAD）复位，低有效

（3） CPU软件复位（主要针对JTAG功能）

（4） PCA复位

（5） WDT复位

（6） LVD低电压复位

不同复位标志位对应芯片复位的来源，由硬件置“1”，软件清除。除了POR的复位标志位，其他复位标志位都可以被上电掉电复位清除。

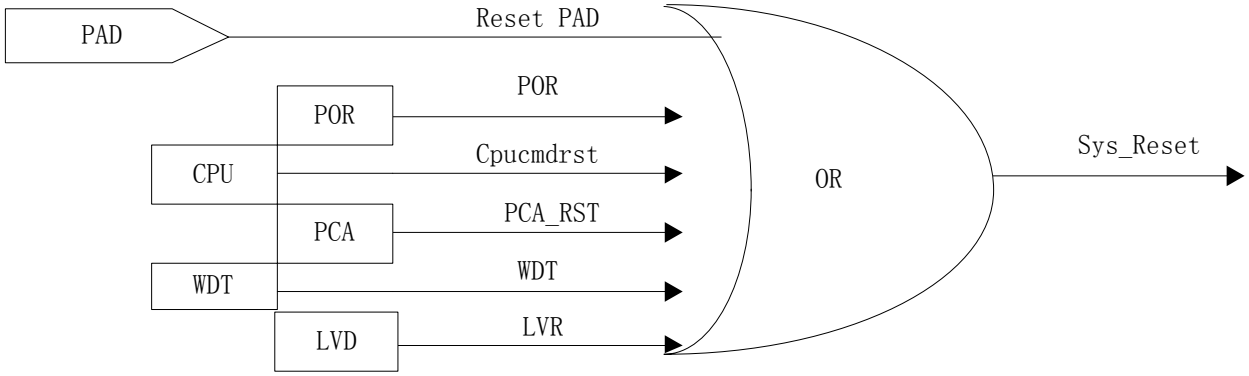


Figure 14-1 复位源构成

14.2 复位操作

14.3 上电掉电复位 POR

当芯片电源电压低于阈值电压，内部会产生一个 POR 信号，当芯片高于阈值电压时，释放 POR 信号。POR 信号会把芯片的 SFR，控制信号全部复位。

14.4 外部复位管脚复位

把外部复位管脚拉到地电位就会产生一个系统复位。这个复位管脚内部带有上拉电阻，因此外部无需连接任何器件。复位管脚在内部集成了一个毛刺过滤电路，典型情况下，小于 10uS 的拉低信号会被作为毛刺而过滤，因此用户在使用复位管脚产生复位信号时，建议保持大于 20uS 的低电位以保证芯片可以接收到复位信号。

14.5 CPU 软件复位

主要为了 JTAG 仿真调试时使用。

14.6 WDT 软件复位

看门狗复位，请参看 WDT 一章说明。

14.7 PCA 复位

PCA 复位，充当一个额外的看门狗复位，请参考 PCA 一章说明。

14.8 LVD 低电压复位

LVD 可以侦测电源电压，也可侦测端口输入电压。

LVD 可以产生中断，也可配置成复位。请参考 LVD 一章说明。

14.9 复位信号标志寄存器(Reset_Flag)

Reset Flag								
复位信号标志寄存器								
地址空间： S:A4h								
位	7	6	5	4	3	2	1	0
符号	--	LVDRST_Flag	SWRST_Flag	WDTRST_Flag	PCARST_Flag	RSTPAD_Flag	--	POR_Flag
类型	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	0	0	0	0	0	1

位	符号	描述
7	--	保留位
6	LVDRST_Flag	LVD 复位标志位 由 LVD 硬件置“1” 由 POR 以及软件清“0”
5	SWRST_Flag	软件复位标志位 硬件接受到 JTAG 复位命令后硬件置“1” 由 POR 以及软件清“0”
4	WDTRST_Flag	看门狗复位标志位 由看门狗硬件置“1” 由 POR 以及软件清“0”
3	PCARST_Flag	PCA 复位标志位 由 PCA 硬件置“1” 由 POR 以及软件清“0”
2	RSTPAD_Flag	RSTB 端口复位标志位 由 PCA 硬件置“1” 由 POR 以及软件清“0”
1	--	保留位
0	POR_Flag	POR 复位标志位 由 POR 硬件置“1” 由软件清“0”

Figure 14-2 复位标志位

15 中断控制器

15.1 中断控制器概要

HC16Lxx除与Intel 8xC251具有相同的中断源外，还具有更多额外的中断源。中断处理器在每个时钟周期的上升沿进行采样，并在每条指令的完成周期以及Idle空闲模式下的每个周期处理中断。

HC16Lxx支持16个中断源，每个中断源有4个优先级。每个中断源可以在一个SFR中有一个或多个中断标志。当一个周边外设满足中断条件时，相应的中断标志被置“1”。

每个中断源都可以使用中断允许寄存器的使能信号来允许或禁止中断产生，全局中断使能信号EA位(IE. 7)必须被置“1”，以保证每个独立的中断允许位有效。如果EA位被清“0”，所有中断都被禁止，无论相关中断使能位是否使能。

15.2 中断控制器构成

中断控制框图如下：

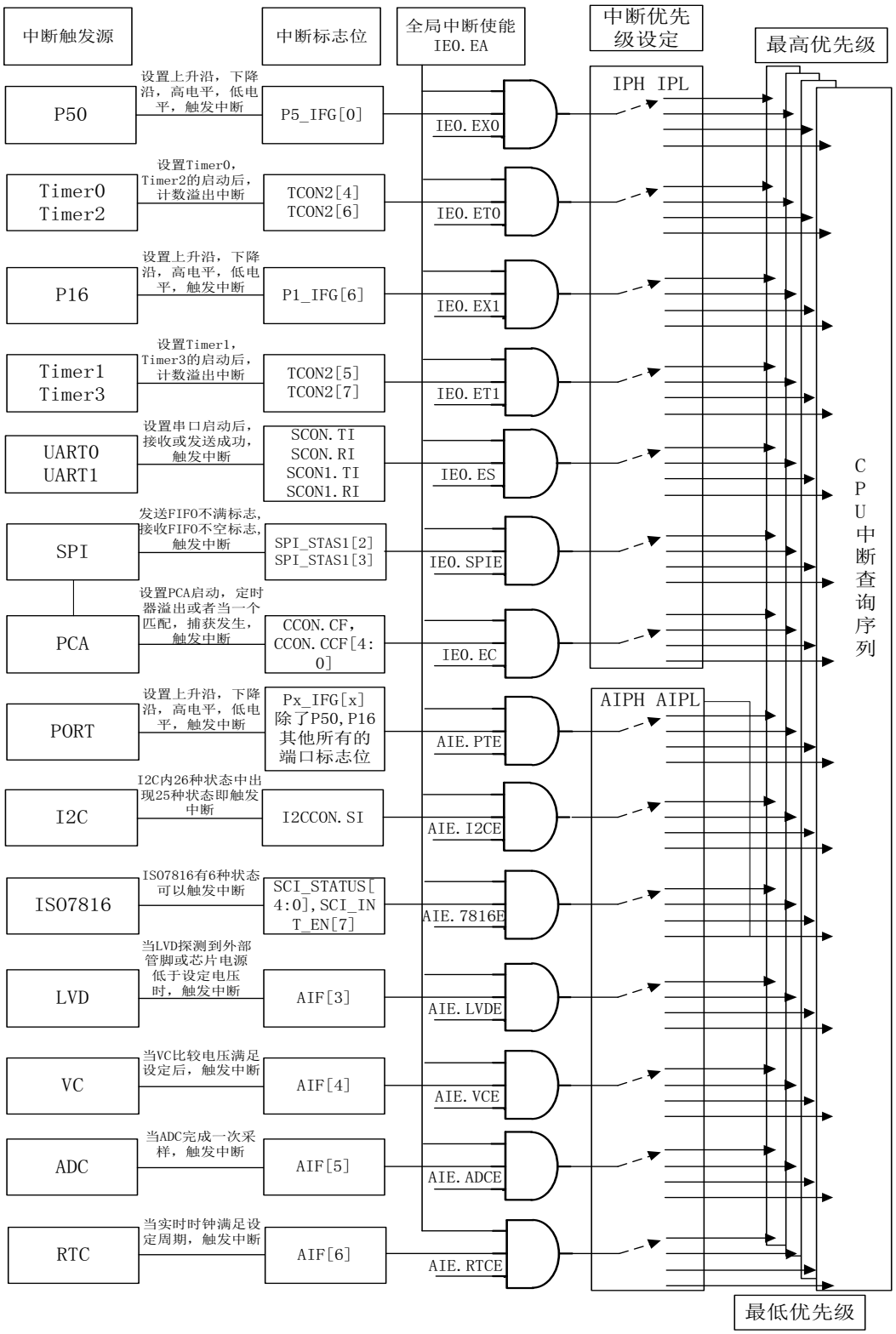


Figure 15-1中断框图

以下列出中断向量表：

中断分类	中断名称	中断处理优先级	中断向量地址	可唤醒空闲模式 IDL	可唤醒低功耗模式 LPM
S/W	TRAP(JTAG调试中断)	1	FF:007Bh	-	-
NMI	NMI	2	FF:003Bh	V	-
内部中断	快速端口中断0(P50)	3	FF:0003h	V	V
	定时器中断0(Timer0/2)	4	FF:000Bh	V	-
	快速端口中断1(P16)	5	FF:0013h	V	V
	定时器中断1(Timer1/3)	6	FF:001Bh	V	-
	UART中断(UART0/1)	7	FF:0023h	V	-
	SPI中断	8	FF:002Bh	V	-
	PCA中断	9	FF:0033h	V	-
附加中断	端口中断	10	FF:0043h	V	V
	I2C中断	11	FF:004Bh	V	-
	ISO7816中断	12	FF:0053h	V	-
	LVD中断	13	FF:005Bh	V	V
	VC中断	14	FF:0063h	V	V
	ADC中断	15	FF:006Bh	V	-
	RTC中断	16	FF:0073h	V	V

Figure 15-2 中断向量表

15.3 中断寄存器

15.4 定时器控制寄存器(TCON)

TCON								
定时器控制寄存器								
地址空间： S:88h								
位	7	6	5	4	3	2	1	0
符号	--	TR1	--	TR0	IE1_	IT1	IE0_	IT0
类型	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	0	0	0	0	0	0

位	符号	描述
7	--	保留位
6	TR1	定时器 1 运行控制位 1：启动定时器 1 0：关闭定时器 1
5	--	保留位
4	TR0	定时器 0 运行控制位 1：启动定时器 0 0：关闭定时器 0
3	IE1_	保留位
2	IT1	必须置为 0
1	IE0_	保留位
0	IT0	必须置为 0

Figure 15-3 TCON寄存器

15.5 中断使能寄存器(IE0)

IE0								
中断使能寄存器								
地址空间： S:A8h								
位	7	6	5	4	3	2	1	0
符号	EA	EC	SPIE	ES	ET1	EX1	ET0	EX0
类型	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	0	0	0	0	0	0

位	符号	描述
7	EA	全局中断使能 （除 TRAP 以及 NMI 中断，即 TRAP 以及 NMI 的使能与 EA 无关） 1：CPU 响应中断 0：CPU 不响应中断
6	EC	PCA 中断使能信号 1：PCA 中断使能 0：PCA 中断禁止
5	SPIE	SPI 中断使能信号 1：SPI 中断使能 0：SPI 中断禁止
4	ES	串口中断使能信号 1：串口中断使能 0：串口中断禁止
3	ET1	定时器 1、定时器 3 中断使能信号 1：定时器 1, 定时器 3 中断使能 0：定时器 1, 定时器 3 中断禁止
2	EX1	外部快速中断 1（P16） 1：外部快速中断 1 使能 0：外部快速中断 1 禁止
1	ET0	定时器 0、定时器 2 中断使能信号 1：定时器 0, 定时器 2 中断使能 0：定时器 0, 定时器 2 中断禁止
0	EX0	外部快速中断 0（P50） 1：外部快速中断 0 使能 0：外部快速中断 0 禁止

Figure 15-4 IE0寄存器

15.6 附加中断使能寄存器(AIE)

AIE								
附加中断使能寄存器								
地址空间： S:E8h								
位	7	6	5	4	3	2	1	0
符号	--	RTCE	ADCE	VCE	LVDE	7816E	I2CE	PTE
类型	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	0	0	0	0	0	0

位	符号	描述
7	--	保留位
6	RTCE	RTC 中断使能信号 1：RTC 中断使能 0：RTC 中断禁止
5	ADCE	ADC 中断使能信号 1：ADC 中断使能 0：ADC 中断禁止
4	VCE	VC 中断使能信号 1：VC 中断使能 0：VC 中断禁止
3	LVDE	LVD 中断使能信号 1：LVD 中断使能 0：LVD 中断禁止
2	7816E	7816 中断使能信号 1：7816 中断使能 0：7816 中断禁止
1	I2CE	I^2C 中断使能信号 1：I^2C 中断使能 0：I^2C 中断禁止
0	PTE	端口中断使能信号 1：端口中断使能 0：端口中断禁止

Figure 15-5 AIE寄存器

15.7 附加中断标志寄存器(AIF)

AIF								
附加中断标志寄存器								
地址空间： S:C0h								
位	7	6	5	4	3	2	1	0
符号	--	RTC_ Flag	ADC_ Flag	VC_ Flag	LVD_ Flag	7816_ Flag	I2C_Flag	Port_ Flag
类型	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	0	0	0	0	0	0

位	符号	描述
7	--	保留位
6	RTC_Flag	RTC 中断标志信号 1：RTC 中断触发 0：RTC 中断未触发 硬件置“1”，软件在中断子程序退出之前清“0”
5	ADC_Flag	ADC 中断标志信号 1：ADC 中断触发 0：ADC 中断未触发 硬件置“1”，软件在中断子程序退出之前清“0”
4	VC_Flag	VC 中断标志信号 1：VC 中断触发 0：VC 中断未触发 硬件置“1”，软件在中断子程序退出之前清“0”
3	LVD_Flag	LVD 中断标志信号 1：LVD 中断触发 0：LVD 中断未触发 硬件置“1”，软件在中断子程序退出之前清“0”
2	7816_Flag	7816 中断标志位，是 7816 各中断标志位组合 对 7816 内的各中断标志位全部清“0”，此标志位自动清“0”
1	I2C_Flag	I2C 中断标志位，为 I2CCON. SI 映射 对 I2CCON. SI 清“0”，此标志位自动清“0”
0	Port_Flag	端口中断标志位，为除 P50/P16 之外所有 Px_IFG[n]标志位组合 对端口中断中所有的中断标志位清“0”，此标志位自动清“0”

Figure 15-6 AIF寄存器

15.8 中断优先级寄存器(IPH0，IPL0)

IPH0								
中断优先级高位寄存器								
地址空间： S:B7h								
位	7	6	5	4	3	2	1	0
符号	--	IPHC	IPHT2	IPHS	IPHT1	IPHX1	IPHT0	IPHX0
类型	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	0	0	0	0	0	0
IPL0								
中断优先级低位寄存器								
地址空间： S:B8h，								
位	7	6	5	4	3	2	1	0
符号	--	IPLC	IPLT2	IPLS	IPHL1	IPLX1	IPLT0	IPLX0
类型	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	0	0	0	0	0	0

位	符号	描述
7	--	保留位
6	IPHC IPLC	PCA 中断优先级设置
5	IPHT2 IPLT2	SPI 中断优先级设置
4	IPHS IPLS	UART 中断优先级设置

PCA 中断优先级设置

IPHC	IPLC	优先级	
0	0	0	最低优先级
0	1	1	
1	0	2	
1	1	3	最高优先级

SPI 中断优先级设置

IPHT2	IPLT2	优先级	
0	0	0	最低优先级
0	1	1	
1	0	2	
1	1	3	最高优先级

UART 中断优先级设置

IPHS	IPLS	优先级	
0	0	0	最低优先级
0	1	1	
1	0	2	
1	1	3	最高优先级

<table>
<tr><td>3</td><td>IPHT1
IPLT1</td><td>定时器 1 & 定时器 3 中断优先级设置<table><tr><td>IPHT1</td><td>IPLT1</td><td>优先级</td><td></td></tr><tr><td>0</td><td>0</td><td>0</td><td>最低优先级</td></tr><tr><td>0</td><td>1</td><td>1</td><td></td></tr><tr><td>1</td><td>0</td><td>2</td><td></td></tr><tr><td>1</td><td>1</td><td>3</td><td>最高优先级</td></tr></table></td></tr>
<tr><td>2</td><td>IPHX1
IPLX1</td><td>快速中断 1 优先级设置<table><tr><td>IPHX1</td><td>IPLX1</td><td>优先级</td><td></td></tr><tr><td>0</td><td>0</td><td>0</td><td>最低优先级</td></tr><tr><td>0</td><td>1</td><td>1</td><td></td></tr><tr><td>1</td><td>0</td><td>2</td><td></td></tr><tr><td>1</td><td>1</td><td>3</td><td>最高优先级</td></tr></table></td></tr>
<tr><td>1</td><td>IPHT0
IPLT0</td><td>定时器 0 & 定时器 2 中断优先级设置<table><tr><td>IPHT0</td><td>IPLT0</td><td>优先级</td><td></td></tr><tr><td>0</td><td>0</td><td>0</td><td>最低优先级</td></tr><tr><td>0</td><td>1</td><td>1</td><td></td></tr><tr><td>1</td><td>0</td><td>2</td><td></td></tr><tr><td>1</td><td>1</td><td>3</td><td>最高优先级</td></tr></table></td></tr>
<tr><td>0</td><td>IPHX0
IPLX0</td><td>快速中断 0 优先级设置<table><tr><td>IPHX0</td><td>IPLX0</td><td>优先级</td><td></td></tr><tr><td>0</td><td>0</td><td>0</td><td>最低优先级</td></tr><tr><td>0</td><td>1</td><td>1</td><td></td></tr><tr><td>1</td><td>0</td><td>2</td><td></td></tr><tr><td>1</td><td>1</td><td>3</td><td>最高优先级</td></tr></table></td></tr>
</table>

Figure 15-7 IPH0，IPL0 寄存器

15.9 附加中断优先级寄存器(AIPH，AIPL)

AIPH								
附加中断优先级高位寄存器								
地址空间： S:F7h								
位	7	6	5	4	3	2	1	0
符号	--	AIPH6	AIPH5	AIPH4	AIPH3	AIPH2	AIPH1	AIPH0
类型	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	0	0	0	0	0	0
AIPL								
附加中断优先级低位寄存器								
地址空间： S:F8h，								
位	7	6	5	4	3	2	1	0
符号	--	AIPL6	AIPL5	AIPL4	AIPL3	AIPL2	AIPL1	AIPL0
类型	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	0	0	0	0	0	0

位	符号	描述
7	--	保留位
6	AIPH6 AIPL6	RTC 中断优先级设置
5	AIPH5 AIPL5	ADC 中断优先级设置
4	AIPH4 AIPL4	VC 中断优先级设置

RTC 中断优先级设置

AIPH6	AIPL6	优先级	
0	0	0	最低优先级
0	1	1	
1	0	2	
1	1	3	最高优先级

ADC 中断优先级设置

AIPH5	AIPL5	优先级	
0	0	0	最低优先级
0	1	1	
1	0	2	
1	1	3	最高优先级

VC 中断优先级设置

AIPH4	AIPL4	优先级	
0	0	0	最低优先级
0	1	1	
1	0	2	
1	1	3	最高优先级

<table>
<tr><td>3</td><td>AIPH3
AIPL3</td><td>LVD 中断优先级设置<table><tr><td>AIPH3</td><td>AIPL3</td><td>优先级</td><td></td></tr><tr><td>0</td><td>0</td><td>0</td><td>最低优先级</td></tr><tr><td>0</td><td>1</td><td>1</td><td></td></tr><tr><td>1</td><td>0</td><td>2</td><td></td></tr><tr><td>1</td><td>1</td><td>3</td><td>最高优先级</td></tr></table></td></tr>
<tr><td>2</td><td>AIPH2
AIPL2</td><td>7816 优先级设置<table><tr><td>AIPH2</td><td>AIPL2</td><td>优先级</td><td></td></tr><tr><td>0</td><td>0</td><td>0</td><td>最低优先级</td></tr><tr><td>0</td><td>1</td><td>1</td><td></td></tr><tr><td>1</td><td>0</td><td>2</td><td></td></tr><tr><td>1</td><td>1</td><td>3</td><td>最高优先级</td></tr></table></td></tr>
<tr><td>1</td><td>AIPH1
AIPL1</td><td>I2C 中断优先级设置<table><tr><td>AIPH1</td><td>AIPL1</td><td>优先级</td><td></td></tr><tr><td>0</td><td>0</td><td>0</td><td>最低优先级</td></tr><tr><td>0</td><td>1</td><td>1</td><td></td></tr><tr><td>1</td><td>0</td><td>2</td><td></td></tr><tr><td>1</td><td>1</td><td>3</td><td>最高优先级</td></tr></table></td></tr>
<tr><td>0</td><td>AIPH0
AIPL0</td><td>端口优先级设置<table><tr><td>AIPH0</td><td>AIPL0</td><td>优先级</td><td></td></tr><tr><td>0</td><td>0</td><td>0</td><td>最低优先级</td></tr><tr><td>0</td><td>1</td><td>1</td><td></td></tr><tr><td>1</td><td>0</td><td>2</td><td></td></tr><tr><td>1</td><td>1</td><td>3</td><td>最高优先级</td></tr></table></td></tr>
</table>

Figure 15-2 AIPH，AIPL 寄存器

16 通用 IO 端口（GPIO）

16.1 通用 IO 端口简介

HC16Lxx 具有58个数字通用输入输出端口，P07~P00, P17~P10, P27~P20, P37~P30, P47~P40, P55~P50, P63~P60, P73~P70, P83~P80。其中LCD 的segment 输出与 P1x，P2x，P3x，P4x，P7x，P8x分别共用一个管脚，ADC，VC，LVD的输入与 P0x 共用一个管脚。复用信息请参考第五章的管脚定义。P1x端口的驱动能力比其他端口的驱动能力要大，具体的驱动电流请参照本手册的电气特性一章。

每个端口可以配置成以下几种输入/输出方式：内部拉高输入，浮动输入，推挽输出，开漏输出，双倍驱动能力输出。其中各功能模块（如SPI，UART，I^2C，7816等）的输入输出配置需要由用户自己选择，HC16Lxx不做硬件强制配置。端口复位后为浮动输入，以防芯片被异常复位之后，外部器件产生异常动作。但为了避免浮动输入而产生的漏电，用户要在芯片启动之后对通用输入/输出端口进行配置。

每个端口可以使用寄存器单独配置相应功能，每个端口可以被单独读取或写入。

当端口被配置成模拟端口之后，数字功能将被隔离，不能输出数字“1”和“0”，如果对端口进行读取，其结果为“0”。

所有的端口都具有响应中断的能力，并且每个端口可以被独立设置响应中断的方式。P50独享外部快速中断0， P16独享外部中断1， 剩余的56个端口共用一个中断源，检查相应的Px_IFG[n]的中断标志位即可查询出中断触发端口。每个端口中断可以配置成高电平触发，低电平触发，上升沿触发，下降沿触发。

所有的端口内部数据寄存器Px[n]以及端口输入输出寄存器Px_DIR[n]都支持位操作。其他的端口配置寄存器则不支持位操作，只支持字节操作。

16.2 读取-修改-回写 功能

下列指令是被用来读取端口内部寄存器，而不是读取外部引脚状态。这些指令会读取指定寄存器的值、并对读取值进行修改、最后将修改后的值写回该寄存器的指令，称为“读取–修改–回写”指令。

指令	描述	与端口操作例程
ANL	逻辑与	ANL P0, A
ORL	逻辑或	ORL P0, A
XRL	逻辑异或	XRL P0, A
JBC	判断可以位寻址区域内指定位是否为1，为1则跳转到指定位置，并同时清除该位（置0）	JBC P10, rel
CPL	补充位	CPL P11
INC	增加	INC P2
DEC	减少	DEC P2
DJNZ	地址字节或寄存器减1，若不为0则跳转	DJNZ P2, rel
MOV bit, CY	将CY bit的值赋给另一个bit	MOV P37, CY
CLR bit	清除位	CLR P36
SETB bit	设置位	SET P35

Figure 16-1 读取–修改–回写 指令

在上表中的最后三条“读 – 修改 – 写回”指令，不能以位的方式操作，只能字节为单位进行读取–修改–写回；其余的指令只能用来读取存放外部引脚状态的端口寄存器，而不是用来读取内部寄存器。（比如： MOV A, P0）

16.3 端口寄存器

16.4 端口输出高/低电平寄存器 Px (x=0,1,2,3,4,5,6,7,8)

bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	地址	初始值	属性	支持端口
P0								0x080	xxxxxxxx	R/W	P07--P00
P1								0x090			P17--P10
P2								0x0A0			P27--P20
P3								0x0B0			P37--P30
P4								0x0E2			P47--P40
保留位	P5							0x0E3			P55--P50
保留位		P6						0x0E4			P63--P60
保留位		P7						0x0E5			P73--P70
保留位		P8						0x0E6			P83--P80

端口Px寄存器		
位	符号	描述
7	Px7	端口 Px7 输出数据 1：输出为高电平，如果配置成开漏输出，则需外部上拉电阻来拉高。 0：输出为低电平。
6	Px6	端口 Px6 输出数据 1：输出为高电平，如果配置成开漏输出，则需外部上拉电阻来拉高。 0：输出为低电平。
5	Px5	端口 Px5 输出数据 1：输出为高电平，如果配置成开漏输出，则需外部上拉电阻来拉高。 0：输出为低电平。
4	Px4	端口 Px4 输出数据 1：输出为高电平，如果配置成开漏输出，则需外部上拉电阻来拉高。 0：输出为低电平。
3	Px3	端口 Px3 输出数据 1：输出为高电平，如果配置成开漏输出，则需外部上拉电阻来拉高。 0：输出为低电平。
2	Px2	端口 Px2 输出数据 1：输出为高电平，如果配置成开漏输出，则需外部上拉电阻来拉高。 0：输出为低电平。
1	Px1	端口 Px1 输出数据 1：输出为高电平，如果配置成开漏输出，则需外部上拉电阻来拉高。 0：输出为低电平。
0	Px0	端口 Px0 输出数据 1：输出为高电平，如果配置成开漏输出，则需外部上拉电阻来拉高。 0：输出为低电平。

Figure 16-2 Px寄存器

在上电复位或掉电复位后，此寄存器状态为随机态；其他方式复位之后（复位过程中芯片未掉电），此寄存器内的数值保持之前的写入值不变

16.5 端口输入/输出选择寄存器 Px_DIR （x=0,1,2,3,4,5,6,7,8）

bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	地址	初始值	属性	支持端口
P0_DIR								0x0AC	11111111	R/W	P07--P00
P1_DIR								0x0AD	11111111		P17--P10
P2_DIR								0x0AE	11111111		P27--P20
P3_DIR								0x0AF	11111111		P37--P30
P4_DIR								0x0F2	11111111		P47--P40
保留位	P5_DIR							0x0F3	xx111111		P55--P50
保留位				P6_DIR				0x0F4	xxxx1111		P63--P60
保留位				P7_DIR				0x0F5	xxxx1111		P73--P70
保留位				P8_DIR				0x0F6	xxxx1111		P83--P80

Px_DIR		
端口Px配置输入输出寄存器		
位	符号	描述
7	PxDIR7	端口 Px7 输入输出配置寄存器 1：配置成输入 0：配置成输出
6	PxDIR6	端口 Px6 输入输出配置寄存器 1：配置成输入 0：配置成输出
5	PxDIR5	端口 Px5 输入输出配置寄存器 1：配置成输入 0：配置成输出
4	PxDIR4	端口 Px4 输入输出配置寄存器 1：配置成输入 0：配置成输出
3	PxDIR3	端口 Px3 输入输出配置寄存器 1：配置成输入 0：配置成输出
2	PxDIR2	端口 Px2 输入输出配置寄存器 1：配置成输入 0：配置成输出
1	PxDIR1	端口 Px1 输入输出配置寄存器 1：配置成输入 0：配置成输出
0	PxDIR0	端口 Px0 输入输出配置寄存器 1：配置成输入 0：配置成输出

Figure 16-3 Px_DIR寄存器

16.6 端口中断源选择寄存器 Px_INT_SEL (x=0,1,2,3,4,5,6,7,8)

bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	地址	初始值	属性	支持端口
P0_INT_SEL								0x0FF04	00000000	R/W	P07--P00
P1_INT_SEL								0x0FF24	00000000		P17--P10
P2_INT_SEL								0x0FF44	00000000		P27--P20
P3_INT_SEL								0x0FF64	00000000		P37--P30
P4_INT_SEL								0x0FF84	00000000		P47--P40
保留位	P5_INT_SEL							0x0FFA4	xx000000		P55--P50
保留位				P6_INT_SEL				0x10004	xxxx0000		P63--P60
保留位				P7_INT_SEL				0x10024	xxxx0000		P73--P70
保留位				P8_INT_SEL				0x10044	xxxx0000		P83--P80

Px_INT_SEL		
端口Px 中断源选择寄存器		
位	符号	描述
7	Px7_INT_SEL	中断源选择信号，端口 Px7 可以配置成电平触发或沿触发 0：沿触发 1：电平触发
6	Px6_INT_SEL	中断源选择信号，端口 Px6 可以配置成电平触发或沿触发 0：沿触发 1：电平触发
5	Px5_INT_SEL	中断源选择信号，端口 Px5 可以配置成电平触发或沿触发 0：沿触发 1：电平触发
4	Px4_INT_SEL	中断源选择信号，端口 Px4 可以配置成电平触发或沿触发 0：沿触发 1：电平触发
3	Px3_INT_SEL	中断源选择信号，端口 Px3 可以配置成电平触发或沿触发 0：沿触发 1：电平触发
2	Px2_INT_SEL	中断源选择信号，端口 Px2 可以配置成电平触发或沿触发 0：沿触发 1：电平触发
1	Px1_INT_SEL	中断源选择信号，端口 Px1 可以配置成电平触发或沿触发 0：沿触发 1：电平触发
0	Px0_INT_SEL	中断源选择信号，端口 Px0 可以配置成电平触发或沿触发 0：沿触发 1：电平触发

Figure 16-4 Px_INT_SEL 寄存器

16.7 端口中断方式选择寄存器 Px_Edge_SEL (x=0,1,2,3,4,5,6,7,8)

bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	地址	初始值	属性	支持端口
P0_Edge_SEL								0x0FF06	00000000	R/W	P07--P00
P1_Edge_SEL								0x0FF26	00000000		P17--P10
P2_Edge_SEL								0x0FF46	00000000		P27--P20
P3_Edge_SEL								0x0FF66	00000000		P37--P30
P4_Edge_SEL								0x0FF86	00000000		P47--P40
保留位	P5_Edge_SEL							0x0FFA6	xx000000		P55--P50
保留位				P6_Edge_SEL				0x10006	xxxx0000		P63--P60
保留位				P7_Edge_SEL				0x10026	xxxx0000		P73--P70
保留位				P8_Edge_SEL				0x10046	xxxx0000		P83--P80

Px_Edge_SEL		
端口Px 中断方式选择寄存器		
位	符号	描述
7	Px7_Edge_SEL	中断方式选择信号，端口 Px7 可以配置成电平触发或沿触发 0：下降沿触发（当 Px7_INT_SEL=0）或低电平触发（当 Px7_INT_SEL=1） 1：上降沿触发（当 Px7_INT_SEL=0）或高电平触发（当 Px7_INT_SEL=1）
6	Px6_Edge_SEL	中断方式选择信号，端口 Px6 可以配置成电平触发或沿触发 0：下降沿触发（当 Px6_INT_SEL=0）或低电平触发（当 Px6_INT_SEL=1） 1：上降沿触发（当 Px6_INT_SEL=0）或高电平触发（当 Px6_INT_SEL=1）
5	Px5_Edge_SEL	中断方式选择信号，端口 Px5 可以配置成电平触发或沿触发 0：下降沿触发（当 Px5_INT_SEL=0）或低电平触发（当 Px5_INT_SEL=1） 1：上降沿触发（当 Px5_INT_SEL=0）或高电平触发（当 Px5_INT_SEL=1）
4	Px4_Edge_SEL	中断方式选择信号，端口 Px4 可以配置成电平触发或沿触发 0：下降沿触发（当 Px4_INT_SEL=0）或低电平触发（当 Px4_INT_SEL=1） 1：上降沿触发（当 Px4_INT_SEL=0）或高电平触发（当 Px4_INT_SEL=1）
3	Px3_Edge_SEL	中断方式选择信号，端口 Px3 可以配置成电平触发或沿触发 0：下降沿触发（当 Px3_INT_SEL=0）或低电平触发（当 Px3_INT_SEL=1） 1：上降沿触发（当 Px3_INT_SEL=0）或高电平触发（当 Px3_INT_SEL=1）
2	Px2_Edge_SEL	中断方式选择信号，端口 Px2 可以配置成电平触发或沿触发 0：下降沿触发（当 Px2_INT_SEL=0）或低电平触发（当 Px2_INT_SEL=1） 1：上降沿触发（当 Px2_INT_SEL=0）或高电平触发（当 Px2_INT_SEL=1）
1	Px1_Edge_SEL	中断方式选择信号，端口 Px1 可以配置成电平触发或沿触发 0：下降沿触发（当 Px1_INT_SEL=0）或低电平触发（当 Px1_INT_SEL=1） 1：上降沿触发（当 Px1_INT_SEL=0）或高电平触发（当 Px1_INT_SEL=1）
0	Px0_Edge_SEL	中断方式选择信号，端口 Px0 可以配置成电平触发或沿触发 0：下降沿触发（当 Px0_INT_SEL=0）或低电平触发（当 Px0_INT_SEL=1） 1：上降沿触发（当 Px0_INT_SEL=0）或高电平触发（当 Px0_INT_SEL=1）

Figure 16-5 Px_Edge_SEL 寄存器

16.8 端口中断使能寄存器 Px_IE (x=0,1,2,3,4,5,6,7,8)

bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	地址	初始值	属性	支持端口
P0_IE								0x0FF08	00000000	R/W	P07--P00
P1_IE								0x0FF28	00000000		P17--P10
P2_IE								0x0FF48	00000000		P27--P20
P3_IE								0x0FF68	00000000		P37--P30
P4_IE								0x0FF88	00000000		P47--P40
保留位	P5_IE							0x0FFA8	xx000000		P55--P50
保留位				P6_IE				0x10008	xxxx0000		P63--P60
保留位				P7_IE				0x10028	xxxx0000		P73--P70
保留位				P8_IE				0x10048	xxxx0000		P83--P80

Px_IE		
端口Px 中断使能寄存器		
位	符号	描述
7	Px7_IE	端口 Px7 中断使能信号 0：中断禁用 1：中断使能
6	Px6_IE	端口 Px6 中断使能信号 0：中断禁用 1：中断使能
5	Px5_IE	端口 Px5 中断使能信号 0：中断禁用 1：中断使能
4	Px4_IE	端口 Px4 中断使能信号 0：中断禁用 1：中断使能
3	Px3_IE	端口 Px3 中断使能信号 0：中断禁用 1：中断使能
2	Px2_IE	端口 Px2 中断使能信号 0：中断禁用 1：中断使能
1	Px1_IE	端口 Px1 中断使能信号 0：中断禁用 1：中断使能
0	Px0_IE	端口 Px0 中断使能信号 0：中断禁用 1：中断使能

Figure 16-6 Px_IE 寄存器

16.9 端口中断标志位寄存器 Px_IFG (x=0,1,2,3,4,5,6,7,8)

bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	地址	初始值	属性	支持端口
P0_IFG								0x0FF0A	00000000	R/W	P07--P00
P1_IFG								0x0FF2A	00000000		P17--P10
P2_IFG								0x0FF4A	00000000		P27--P20
P3_IFG								0x0FF6A	00000000		P37--P30
P4_IFG								0x0FF8A	00000000		P47--P40
保留位	P5_IFG							0x0FFAA	xx000000		P55--P50
保留位				P6_IFG				0x1000A	xxxx0000		P63--P60
保留位				P7_IFG				0x1002A	xxxx0000		P73--P70
保留位				P8_IFG				0x1004A	xxxx0000		P83--P80

Px_IFG		
端口Px 中断标志位寄存器		
位	符号	描述
7	Px7_IFG	端口 Px7 中断触发标志位，硬件置“1”，软件清“0” 0：无中断触发 1：中断触发标志位
6	Px6_IFG	端口 Px6 中断触发标志位，硬件置“1”，软件清“0” 0：无中断触发 1：中断触发标志位
5	Px5_IFG	端口 Px5 中断触发标志位，硬件置“1”，软件清“0” 0：无中断触发 1：中断触发标志位
4	Px4_IFG	端口 Px4 中断触发标志位，硬件置“1”，软件清“0” 0：无中断触发 1：中断触发标志位
3	Px3_IFG	端口 Px3 中断触发标志位，硬件置“1”，软件清“0” 0：无中断触发 1：中断触发标志位
2	Px2_IFG	端口 Px2 中断触发标志位，硬件置“1”，软件清“0” 0：无中断触发 1：中断触发标志位
1	Px1_IFG	端口 Px1 中断触发标志位，硬件置“1”，软件清“0” 0：无中断触发 1：中断触发标志位
0	Px0_IFG	端口 Px0 中断触发标志位，硬件置“1”，软件清“0” 0：无中断触发 1：中断触发标志位

Figure 16-7 Px_IFG 寄存器

16.10 端口输入上拉使能寄存器 Px_PE (x=0,1,2,3,4,5,6,7,8)

bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	地址	初始值	属性	支持端口
P0_PE								0x0FF0C	00000000	R/W	P07--P00
P1_PE								0x0FF2C	00000000		P17--P10
P2_PE								0x0FF4C	00000000		P27--P20
P3_PE								0x0FF6C	00000000		P37--P30
P4_PE								0x0FF8C	00000000		P47--P40
保留位	P5_PE							0x0FFAC	xx000000		P55--P50
保留位				P6_PE				0x1000C	xxxx0000		P63--P60
保留位				P7_PE				0x1002C	xxxx0000		P73--P70
保留位				P8_PE				0x1004C	xxxx0000		P83--P80

Px_PE		
端口Px 输入上拉使能寄存器		
注意：若端口配置成模拟端口，Px_PE设置失效，无上拉功能。		
位	符号	描述
7	Px7_PE	端口 Px7 输入上拉使能信号 0：无上拉功能 1：有内部上拉功能
6	Px6_PE	端口 Px6 输入上拉使能信号 0：无上拉功能 1：有内部上拉功能
5	Px5_PE	端口 Px5 输入上拉使能信号 0：无上拉功能 1：有内部上拉功能
4	Px4_PE	端口 Px4 输入上拉使能信号 0：无上拉功能 1：有内部上拉功能
3	Px3_PE	端口 Px3 输入上拉使能信号 0：无上拉功能 1：有内部上拉功能
2	Px2_PE	端口 Px2 输入上拉使能信号 0：无上拉功能 1：有内部上拉功能
1	Px1_PE	端口 Px1 输入上拉使能信号 0：无上拉功能 1：有内部上拉功能
0	Px0_PE	端口 Px0 输入上拉使能信号 0：无上拉功能 1：有内部上拉功能

Figure 16-8 Px_PE 寄存器

16.11 端口增强输出电流使能寄存器 Px_DS (x=0,1,2,3,4,5,6,7,8)

bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	地址	初始值	属性	支持端口
P0_DS								0x0FF0E	00000000	R/W	P07--P00
P1_DS								0x0FF2E	00000000		P17--P10
P2_DS								0x0FF4E	00000000		P27--P20
P3_DS								0x0FF6E	00000000		P37--P30
P4_DS								0x0FF8E	00000000		P47--P40
保留位	P5_DS							0x0FFAE	xx000000		P55--P50
保留位				P6_DS				0x1000E	xxxx0000		P63--P60
保留位				P7_DS				0x1002E	xxxx0000		P73--P70
保留位				P8_DS				0x1004E	xxxx0000		P83--P80

Px_DS		
端口Px 增强输出电流使能寄存器		
注意：端口配置成模拟端口，Px_DS设置失效，无增强驱动电流能力功能。		
位	符号	描述
7	Px7_DS	端口 Px7 增强输出电流使能信号 0：普通型电流输出，正常 Sink/Drive 电流能力。 1：增强型电流输出，增加 Sink/Drive 电流能力。
6	Px6_DS	端口 Px6 增强输出电流使能信号 0：普通型电流输出，正常 Sink/Drive 电流能力。 1：增强型电流输出，增加 Sink/Drive 电流能力。
5	Px5_DS	端口 Px5 增强输出电流使能信号 0：普通型电流输出，正常 Sink/Drive 电流能力。 1：增强型电流输出，增加 Sink/Drive 电流能力。
4	Px4_DS	端口 Px4 增强输出电流使能信号 0：普通型电流输出，正常 Sink/Drive 电流能力。 1：增强型电流输出，增加 Sink/Drive 电流能力。
3	Px3_DS	端口 Px3 增强输出电流使能信号 0：普通型电流输出，正常 Sink/Drive 电流能力。 1：增强型电流输出，增加 Sink/Drive 电流能力。
2	Px2_DS	端口 Px2 增强输出电流使能信号 0：普通型电流输出，正常 Sink/Drive 电流能力。 1：增强型电流输出，增加 Sink/Drive 电流能力。
1	Px1_DS	端口 Px1 增强输出电流使能信号 0：普通型电流输出，正常 Sink/Drive 电流能力。 1：增强型电流输出，增加 Sink/Drive 电流能力。
0	Px0_DS	端口 Px0 增强输出电流使能信号 0：普通型电流输出，正常 Sink/Drive 电流能力。 1：增强型电流输出，增加 Sink/Drive 电流能力。

Figure 16-9 Px_DS 寄存器

16.12 端口开漏输出功能寄存器 Px_OpenDrain (x=0,1,2,3,4,5,6,7,8)

bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	地址	初始值	属性	支持端口
P0_OpenDrain								0x0FF16	00000000	R/W	P07--P00
P1_OpenDrain								0x0FF36	00000000		P17--P10
P2_OpenDrain								0x0FF56	00000000		P27--P20
P3_OpenDrain								0x0FF76	00000000		P37--P30
P4_OpenDrain								0x0FF96	00000000		P47--P40
保留位	P5_OpenDrain							0x0FFB6	xx000000		P55--P50
保留位		P6_OpenDrain						0x10016	xxxx0000		P63--P60
保留位		P7_OpenDrain						0x10036	xxxx0000		P73--P70
保留位		P8_OpenDrain						0x10056	xxxx0000		P83--P80

Px_OpenDrain		
端口Px 开漏输出功能寄存器		
位	符号	描述
7	Px7_OD	端口 Px7 开漏/推挽输出使能 0：输出为推挽(CMOS or Push-Pull)输出 1：输出为开漏(Open Drain)输出
6	Px6_OD	端口 Px6 开漏/推挽输出使能 0：输出为推挽(CMOS or Push-Pull)输出 1：输出为开漏(Open Drain)输出
5	Px5_OD	端口 Px5 开漏/推挽输出使能 0：输出为推挽(CMOS or Push-Pull)输出 1：输出为开漏(Open Drain)输出
4	Px4_OD	端口 Px4 开漏/推挽输出使能 0：输出为推挽(CMOS or Push-Pull)输出 1：输出为开漏(Open Drain)输出
3	Px3_OD	端口 Px3 开漏/推挽输出使能 0：输出为推挽(CMOS or Push-Pull)输出 1：输出为开漏(Open Drain)输出
2	Px2_OD	端口 Px2 开漏/推挽输出使能 0：输出为推挽(CMOS or Push-Pull)输出 1：输出为开漏(Open Drain)输出
1	Px1_OD	端口 Px1 开漏/推挽输出使能 0：输出为推挽(CMOS or Push-Pull)输出 1：输出为开漏(Open Drain)输出
0	Px0_OD	端口 Px0 开漏/推挽输出使能 0：输出为推挽(CMOS or Push-Pull)输出 1：输出为开漏(Open Drain)输出

Figure 16-10 Px_OD 寄存器

16.12.1 端口 P0 功能配置寄存器 0 P0_SEL0

P0_SEL0								
P0 数字端口/模拟端口配置寄存器0								
地址空间： 0x00FF10								
位	7	6	5	4	3	2	1	0
符号	P07_SEL0	P06_SEL0	P05_SEL0	P04_SEL0	P03_SEL0	P02_SEL0	P01_SEL0	P00_SEL0
类型	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	0	0	0	0	0	0

设置P0x_SEL0不会自动设置端口的输入输出

位	符号	描述
7	P07_SEL0	端口 P07 功能配置寄存器 0 0：通用输入输出引脚 1：模拟输入输出引脚，ADC 参考电压输入
6	P06_SEL0	端口 P06 功能配置寄存器 0 0：通用输入输出引脚 1：模拟输入输出引脚，ADC 通道 6 输入，LVD 低电压侦测输入
5	P05_SEL0	端口 P05 功能配置寄存器 0 0：通用输入输出引脚 1：模拟输入输出引脚，ADC 通道 5 输入，VC 通道 3 输入
4	P04_SEL0	端口 P04 功能配置寄存器 0 0：通用输入输出引脚 1：模拟输入输出引脚，ADC 通道 4 输入，VC 通道 2 输入
3	P03_SEL0	端口 P03 功能配置寄存器 0 0：通用输入输出引脚 1：模拟输入输出引脚，ADC 通道 3 输入，VC 通道 1 输入
2	P02_SEL0	端口 P02 功能配置寄存器 0 0：通用输入输出引脚 1：模拟输入输出引脚，ADC 通道 2 输入，VC 通道 0 输入
1	P01_SEL0	端口 P01 功能配置寄存器 0 0：通用输入输出引脚 1：模拟输入输出引脚，ADC 通道 1 输入
0	P00_SEL0	端口 P00 功能配置寄存器 0 0：通用输入输出引脚 1：模拟输入输出引脚，ADC 通道 0 输入

Figure 16-11 P0_SEL0 寄存器

16.13 端口 P1 功能配置寄存器 0 P1_SEL0

P1_SEL0								
P1 数字端口/模拟端口配置寄存器0								
地址空间： 0x00FF30								
位	7	6	5	4	3	2	1	0
符号	P17_SEL0	P16_SEL0	P15_SEL0	P14_SEL0	P13_SEL0	P12_SEL0	P11_SEL0	P10_SEL0
类型	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	0	0	0	0	0	0
设置P1x_SEL0不会自动设置端口的输入输出								

位	符号	描述
7	P17_SEL0	端口 P17 功能配置寄存器 0 0：功能由 P1_SEL1，P1_SEL2 决定 1：LCD Segment 32
6	P16_SEL0	端口 P16 功能配置寄存器 0 0：功能由 P1_SEL1，P1_SEL2 决定 1：LCD Segment 33
5	P15_SEL0	端口 P15 功能配置寄存器 0 0：功能由 P1_SEL1，P1_SEL2 决定 1：LCD Segment 34
4	P14_SEL0	端口 P14 功能配置寄存器 0 0：功能由 P1_SEL1，P1_SEL2 决定 1：LCD Segment 35
3	P13_SEL0	端口 P13 功能配置寄存器 0 0：功能由 P1_SEL1，P1_SEL2 决定 1：LCD Segment 36
2	P12_SEL0	端口 P12 功能配置寄存器 0 0：功能由 P1_SEL1，P1_SEL2 决定 1：LCD Segment 37
1	P11_SEL0	端口 P11 功能配置寄存器 0 0：功能由 P1_SEL1，P1_SEL2 决定 1：LCD Segment 38
0	P10_SEL0	端口 P10 功能配置寄存器 0 0：功能由 P1_SEL1，P1_SEL2 决定 1：LCD Segment 39

Figure 16-12 P1_SEL0 寄存器

16.14 端口 P1 功能配置寄存器 1 P1_SEL1

P1_SEL1								
P1 数字端口功能配置寄存器1								
地址空间： 0x00FF32								
位	7	6	5	4	3	2	1	0
符号	P17_SEL1	P16_SEL1	P15_SEL1	P14_SEL1	P13_SEL1	P12_SEL1	P11_SEL1	P10_SEL1
类型	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	0	0	0	0	0	0
设置P1x_SEL2,P1x_SEL1不会自动设置端口的输入输出								

位	符号	描述
7	P17_SEL1	请看 P17_SEL2 寄存器描述
6	P16_SEL1	请看 P16_SEL2 寄存器描述
5	P15_SEL1	请看 P15_SEL2 寄存器描述
4	P14_SEL1	请看 P14_SEL2 寄存器描述
3	P13_SEL1	请看 P13_SEL2 寄存器描述
2	P12_SEL1	请看 P12_SEL2 寄存器描述
1	P11_SEL1	请看 P11_SEL2 寄存器描述
0	P10_SEL1	请看 P10_SEL2 寄存器描述

Figure 16-13 P1_SEL1 寄存器

16.15 端口 P1 功能配置寄存器 2 P1_SEL2

P1_SEL1								
P1 数字端口功能配置寄存器2								
地址空间： 0x00FF34								
位	7	6	5	4	3	2	1	0
符号	P17_SEL2	P16_SEL2	P15_SEL2	P14_SEL2	P13_SEL2	P12_SEL2	P11_SEL2	P10_SEL2
类型	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	0	0	0	0	0	0

位	符号	描述
7	P17_SEL2	P17_SEL2,P17_SEL1: 00: P17 被配置成 GPIO 01: P17 被配置成 UART1 TXD 端口 10: P17 被配置成 ISO 7816 CLK 端口 11: P17 被配置成 PCA4 端口，包括 PCA4 捕获输入,PCA4 的比较输出
6	P16_SEL2	P16_SEL2,P16_SEL1: 00: P16 被配置成 GPIO 01: P16 被配置成 UART1 RXD 端口 10: P16 被配置成 ISO 7816 I/O 端口 11: P16 被配置成 PCA3 端口，包括 PCA3 捕获输入,PCA3 的比较输出
5	P15_SEL2	P15_SEL2,P15_SEL1: 00: P15 被配置成 GPIO 01: P15 被配置成 SPI CS 端口 10: P15 被配置成 ISO 7816 reset 端口 11: P15 被配置成 PCA2 端口，包括 PCA2 捕获输入,PCA2 的比较输出
4	P14_SEL2	P14_SEL2,P14_SEL1: 00: P14 被配置成 GPIO 01: P14 被配置成 SPI CLK 端口 10: P14 被配置成主频时钟输出端口 11: P14 被配置成 PCA1 端口，包括 PCA1 捕获输入,PCA1 的比较输出
3	P13_SEL2	P13_SEL2,P13_SEL1: 00: P13 被配置成 GPIO 01: P13 被配置成 SPI MOSI 端口 10: P13 被配置成 RTC 脉冲输出 1 端口 11: P13 被配置成 PCA0 端口，包括 PCA0 捕获输入,PCA0 的比较输出
2	P12_SEL2	P12_SEL2,P12_SEL1: 00: P12 被配置成 GPIO 01: P12 被配置成 SPI MISO 端口 10: P12 被配置成 RTC 脉冲输出 0 端口 11: P12 被配置成 PCA 时钟输入
1	P11_SEL2	P11_SEL2,P11_SEL1: 00: P11 被配置成 GPIO

		01: P11 被配置成 I2C SDO 端口 10: P11 被配置成 Buzzer 蜂鸣器输出端口 11: P11 被配置成计数器 Timer1 的时钟输入
0	P10_SEL2	P10_SEL2,P10_SEL1: 00: P10 被配置成 GPIO 01: P10 被配置成 I^2C SCL 端口 10: P10 被配置成副频时钟 X32K 输出端口 11: P10 被配置成计数器 Timer1 的门控输入

Figure 16-14 P1_SEL2 寄存器

16.16 端口 P2 功能配置寄存器 0 P2_SEL0

P2_SEL0								
P2 数字端口/模拟端口配置寄存器0								
地址空间： 0x00FF50								
位	7	6	5	4	3	2	1	0
符号	P27_SEL0	P26_SEL0	P25_SEL0	P24_SEL0	P23_SEL0	P22_SEL0	P21_SEL0	P20_SEL0
类型	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	0	0	0	0	0	0
设置P2x_SEL0不会自动设置端口的输入输出								

位	符号	描述
7	P27_SEL0	端口 P27 功能配置寄存器 0 0：功能由 P2_SEL1，P2_SEL2 决定 1：LCD Segment 20
6	P26_SEL0	端口 P26 功能配置寄存器 0 0：功能由 P2_SEL1，P2_SEL2 决定 1：LCD Segment 21
5	P25_SEL0	端口 P25 功能配置寄存器 0 0：功能由 P2_SEL1，P2_SEL2 决定 1：LCD Segment 22
4	P24_SEL0	端口 P24 功能配置寄存器 0 0：功能由 P2_SEL1，P2_SEL2 决定 1：LCD Segment 23
3	P23_SEL0	端口 P23 功能配置寄存器 0 0：功能由 P2_SEL1，P2_SEL2 决定 1：LCD Segment 24
2	P22_SEL0	端口 P22 功能配置寄存器 0 0：功能由 P2_SEL1，P2_SEL2 决定 1：LCD Segment 25
1	P21_SEL0	端口 P21 功能配置寄存器 0 0：功能由 P2_SEL1，P2_SEL2 决定 1：LCD Segment 26
0	P20_SEL0	端口 P20 功能配置寄存器 0 0：功能由 P2_SEL1，P2_SEL2 决定 1：LCD Segment 27

Figure 16-15 P2_SEL0 寄存器

16.17 端口 P2 功能配置寄存器 1 P2_SEL1

P2_SEL1								
P2 数字端口功能配置寄存器1								
地址空间： 0x00FF52								
位	7	6	5	4	3	2	1	0
符号	P27_SEL1	P26_SEL1	P25_SEL1	P24_SEL1	P23_SEL1	P22_SEL1	P21_SEL1	P20_SEL1
类型	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	0	0	0	0	0	0
设置P2x_SEL1,P2x_SEL0不会自动设置端口的输入输出								

位	符号	描述
7	P27_SEL1	请参考 P27_SEL2 定义
6	P26_SEL1	请参考 P26_SEL2 定义
5	P25_SEL1	请参考 P25_SEL2 定义
4	P24_SEL1	请参考 P24_SEL2 定义
3	P23_SEL1	请参考 P23_SEL2 定义
2	P22_SEL1	请参考 P22_SEL2 定义
1	P21_SEL1	请参考 P21_SEL2 定义
0	P20_SEL1	请参考 P20_SEL2 定义

Figure 16-16 P2_SEL1 寄存器

16.18 端口 P2 功能配置寄存器 2 P2_SEL2

P2_SEL1								
P2 数字端口功能配置寄存器2								
地址空间： 0x00FF54								
位	7	6	5	4	3	2	1	0
符号	P27_SEL2	P26_SEL2	P25_SEL2	P24_SEL2	P23_SEL2	P22_SEL2	P21_SEL2	P20_SEL2
类型	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	0	0	0	0	0	0

位	符号	描述
7	P27_SEL2	P27_SEL2,P27_SEL1: 00: P27 被配置成 GPIO 01: P27 被配置成 SPI MISO 端口 10: P27 被配置成计数器 Timer0 时钟输入端口 11: P27 被配置成 PCA4 端口，包括 PCA4 捕获输入,PCA4 的比较输出
6	P26_SEL2	P26_SEL2,P26_SEL1: 00: P26 被配置成 GPIO 01: P26 被配置成 SPI MOSI 端口 10: P26 被配置成计数器 Timer0 门控输入端口 11: P26 被配置成 PCA3 端口，包括 PCA3 捕获输入,PCA3 的比较输出
5	P25_SEL2	P25_SEL2,P25_SEL1: 00: P25 被配置成 GPIO 01: P25 被配置成 SPI CLK 端口 10: P25 被配置成电压比较器结果输出端口 11: P25 被配置成 PCA2 端口，包括 PCA2 捕获输入,PCA2 的比较输出
4	P24_SEL2	P24_SEL2,P24_SEL1: 00: P24 被配置成 GPIO 01: P24 被配置成 SPI CS 端口 10: P24 被配置成副频时钟输出端口 11: P24 被配置成 PCA1 端口，包括 PCA1 捕获输入,PCA1 的比较输出
3	P23_SEL2	P23_SEL2,P23_SEL1: 00: P23 被配置成 GPIO 01: P23 被配置成 GPIO 10: P23 被配置成 RTC 脉冲输出 1 端口 11: P23 被配置成 PCA0 端口，包括 PCA0 捕获输入,PCA0 的比较输出
2	P22_SEL2	P22_SEL2,P22_SEL1: 00: P22 被配置成 GPIO 01: P22 被配置成 GPIO 10: P22 被配置成 RTC 脉冲输出 0 端口 11: P22 被配置成 PCA 时钟输入
1	P21_SEL2	P21_SEL2,P21_SEL1: 00: P21 被配置成 GPIO

		01: P21 被配置成 GPIO 10: P21 被配置成 Buzzer 蜂鸣器输出端口 11: P21 被配置成计数器 Timer1 的时钟输入
0	P20_SEL2	P20_SEL2,P20_SEL1: 00: P20 被配置成 GPIO 01: P20 被配置成 GPIO 10: P20 被配置成主频时钟输出端口 11: P20 被配置成计数器 Timer1 的门控输入

Figure 16-17 P2_SEL2 寄存器

16.19 端口 P3 功能配置寄存器 0 P3_SEL0

P3_SEL0								
P3 数字端口/模拟端口配置寄存器0								
地址空间： 0x00FF70								
位	7	6	5	4	3	2	1	0
符号	P37_SEL0	P36_SEL0	P35_SEL0	P34_SEL0	P33_SEL0	P32_SEL0	P31_SEL0	P30_SEL0
类型	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	0	0	0	0	0	0

设置P3x_SEL0不会自动设置端口的输入输出

位	符号	描述
7	P37_SEL0	端口 P37 功能配置寄存器 0 0：通用输入输出引脚 1：LCD Segment 8
6	P36_SEL0	端口 P36 功能配置寄存器 0 0：通用输入输出引脚 1：LCD Segment 9
5	P35_SEL0	端口 P35 功能配置寄存器 0 0：通用输入输出引脚 1：LCD Segment 10
4	P34_SEL0	端口 P34 功能配置寄存器 0 0：通用输入输出引脚 1：LCD Segment 11
3	P33_SEL0	端口 P33 功能配置寄存器 0 0：通用输入输出引脚 1：LCD Segment 12
2	P32_SEL0	端口 P32 功能配置寄存器 0 0：通用输入输出引脚 1：LCD Segment 13
1	P31_SEL0	端口 P31 功能配置寄存器 0 0：通用输入输出引脚 1：LCD Segment 14
0	P30_SEL0	端口 P30 功能配置寄存器 0 0：通用输入输出引脚 1：LCD Segment 15

Figure 16-18 P3_SEL0 寄存器

16.20 端口 P4 功能配置寄存器 0 P4_SEL0

P4_SEL0								
P4 数字端口/模拟端口配置寄存器0								
地址空间： 0x00FF90								
位	7	6	5	4	3	2	1	0
符号	P47_SEL0	P46_SEL0	P45_SEL0	P44_SEL0	P43_SEL0	P42_SEL0	P41_SEL0	P40_SEL0
类型	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	0	0	0	0	0	0
设置P4x_SEL0不会自动设置端口的输入输出								

位	符号	描述
7	P47_SEL0	端口 P47 功能配置寄存器 0 0：功能由 P4_SEL1 决定 1：LCD Segment 0
6	P46_SEL0	端口 P46 功能配置寄存器 0 0：功能由 P4_SEL1 决定 1：LCD Segment 1
5	P45_SEL0	端口 P45 功能配置寄存器 0 0：功能由 P4_SEL1 决定 1：LCD Segment 2
4	P44_SEL0	端口 P44 功能配置寄存器 0 0：功能由 P4_SEL1 决定 1：LCD Segment 3
3	P43_SEL0	端口 P43 功能配置寄存器 0 0：功能由 P4_SEL1 决定 1：LCD Segment 4
2	P42_SEL0	端口 P42 功能配置寄存器 0 0：功能由 P4_SEL1 决定 1：LCD Segment 5
1	P41_SEL0	端口 P41 功能配置寄存器 0 0：功能由 P4_SEL1 决定 1：LCD Segment 6
0	P40_SEL0	端口 P40 功能配置寄存器 0 0：功能由 P4_SEL1 决定 1：LCD Segment 7

Figure 16-19 P4_SEL0 寄存器

16.21 端口 P4 功能配置寄存器 1 P4_SEL1

P4_SEL1								
P4 数字端口功能配置寄存器1								
地址空间： 0x00FF92								
位	7	6	5	4	3	2	1	0
符号	P47_SEL1	P46_SEL1	P45_SEL1	P44_SEL1	P43_SEL1	P42_SEL1	P41_SEL1	P40_SEL1
类型	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	0	0	0	0	0	0
设置P4x_SEL1不会自动设置端口的输入输出								

位	符号	描述
7	P47_SEL1	端口 P47 功能配置寄存器 1 0:端口 P47 被设置成 GPIO 1:端口 P47 被设置成 I^2C 的 SDO 信号
6	P46_SEL1	端口 P46 功能配置寄存器 1 0:端口 P46 被设置成 GPIO 1:端口 P46 被设置成 I^2C 的 SCL 信号
5	P45_SEL1	端口 P45 功能配置寄存器 1 0:端口 P45 被设置成 GPIO 1:端口 P45 被设置成 ISO7816 Reset 复位信号
4	P44_SEL1	端口 P44 功能配置寄存器 1 0:端口 P44 被设置成 GPIO 1:端口 P44 被设置成 ISO7816 I/O 信号
3	P43_SEL1	端口 P43 功能配置寄存器 1 0:端口 P43 被设置成 GPIO 1:端口 P43 被设置成 ISO7816 CLK 信号
2	P42_SEL1	端口 P42 功能配置寄存器 1 0:端口 P42 被设置成 GPIO 1:端口 P42 被设置成蜂鸣器信号
1	P41_SEL1	端口 P41 功能配置寄存器 1 0:端口 P41 被设置成 GPIO 1:端口 P41 被设置成 RTC 脉冲输出信号 1
0	P40_SEL1	端口 P40 功能配置寄存器 1 0:端口 P40 被设置成 GPIO 1:端口 P40 被设置成 RTC 脉冲输出信号 0

Figure 16-20 P4_SEL1 寄存器

16.22 端口 P5 功能配置寄存器 0 P5_SEL0

P5_SEL0								
P5 数字端口/模拟端口配置寄存器0								
地址空间： 0x00FFB0								
位	7	6	5	4	3	2	1	0
符号	--	--	P55_SEL0	P54_SEL0	P53_SEL0	P52_SEL0	P51_SEL0	P50_SEL0
类型	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	0	0	0	0	0	0
设置P5x_SEL0不会自动设置端口的输入输出								

位	符号	描述
7	--	保留位
6	--	保留位
5	P55_SEL0	请参考 P55_SEL1 描述
4	P54_SEL0	请参考 P54_SEL1 描述
3	P53_SEL0	请参考 P53_SEL1 描述
2	P52_SEL0	请参考 P52_SEL1 描述
1	P51_SEL0	请参考 P51_SEL1 描述
0	P50_SEL0	请参考 P50_SEL1 描述

Figure 16-21 P5_SEL0 寄存器

16.23 端口 P5 功能配置寄存器 1 P5_SEL1

P5_SEL1								
P5 数字端口功能配置寄存器1								
地址空间： 0x00FFB2								
位	7	6	5	4	3	2	1	0
符号	--	--	P55_SEL1	P54_SEL1	P53_SEL1	P52_SEL1	P51_SEL1	P50_SEL1
类型	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	0	0	0	0	0	0

位	符号	描述
[7:6]	--	保留位
5	P55_SEL1	端口 P55 设置寄存器 P55_SEL1, P55_SEL0 00：端口 P55 被设置成 GPIO 01：端口 P55 被设置成电压比较器结果信号输出 10：端口 P55 被设置成 RTC 脉冲输出 1 11：端口 P55 被设置成 PCA3 比较输出
4	P54_SEL1	端口 P54 设置寄存器 P54_SEL1, P54_SEL0 00：端口 P54 被设置成 GPIO 01：端口 P54 被设置成蜂鸣器 Buzzer 输出 10：端口 P54 被设置成 RTC 脉冲输出 0 11：端口 P54 被设置成 PCA2 比较输出
3	P53_SEL1	端口 P53 设置寄存器 P53_SEL1, P53_SEL0 00：端口 P53 被设置成 GPIO 01：端口 P53 被设置成 RTC 脉冲输出 1 10：端口 P53 被设置成副频输出 11：端口 P53 被设置成 PCA1 比较输出
2	P52_SEL1	端口 P52 设置寄存器 P52_SEL1, P52_SEL0 00：端口 P52 被设置成 GPIO 01：端口 P52 被设置成 RTC 脉冲输出 0 10：端口 P52 被设置成主频时钟输出 11：端口 P52 被设置成 PCA0 比较输出
1	P51_SEL1	端口 P51 设置寄存器 P51_SEL1, P51_SEL0 00：端口 P51 被设置成 GPIO 01：端口 P51 被设置成 GPIO 10：端口 P51 被设置成 UART0 TXD 端口 11：端口 P51 被设置成计数器 Timer0 时钟输入信号
0	P50_SEL1	端口 P50 设置寄存器 P50_SEL1, P50_SEL0 00：端口 P50 被设置成 GPIO 01：端口 P50 被设置成 GPIO 10：端口 P50 被设置成 UART0 RXD 端口 11：端口 P50 被设置成计数器 Timer0 门控输出信号

Figure 16-22 P5_SEL1 寄存器

16.24 端口 P6 功能配置寄存器 0 P6_SEL0

P6_SEL0								
P6 数字端口配置寄存器0								
地址空间： 0x010010								
位	7	6	5	4	3	2	1	0
符号	--	--	--	--	P63_SEL0	P62_SEL0	P61_SEL0	P60_SEL0
类型	--	--	--	--	R/W	R/W	R/W	R/W
复位值	--	--	--	--	0	0	0	0
设置P6x_SEL0不会自动设置端口的输入输出								
位	符号	描述						
7：4	--	保留位						
3	P63_SEL0	请参考 P63_SEL1 描述						
2	P62_SEL0	请参考 P62_SEL1 描述						
1	P61_SEL0	请参考 P61_SEL1 描述						
0	P60_SEL0	请参考 P60_SEL1 描述						

Figure 16-23 P6_SEL0 寄存器

16.25 端口 P6 功能配置寄存器 1 P6_SEL1

P6_SEL1								
P6 数字端口功能配置寄存器1								
地址空间： 0x010012								
位	7	6	5	4	3	2	1	0
符号	--	--	--	--	P63_SEL1	P62_SEL1	P61_SEL1	P60_SEL1
类型	--	--	--	--	R/W	R/W	R/W	R/W
复位值	--	--	--	--	0	0	0	0

设置P6x_SEL1不会自动设置端口的输入输出

位	符号	描述
7：4	--	保留位
3	P63_SEL0	端口 P63 设置寄存器 P63_SEL1,P63_SEL0 00：端口 P63 被设置为 GPIO 01：端口 P63 被设置为电压比较器结果输出端口 10：端口 P63 被设置为主频时钟输出端口 11：端口 P63 被设置为 PCA3 比较输出端口
2	P62_SEL0	端口 P62 设置寄存器 P62_SEL1,P62_SEL0 00：端口 P62 被设置为 GPIO 01：端口 P62 被设置为蜂鸣器 Buzzer 输出端口 10：端口 P62 被设置为 ISO7816 Reset 复位输出端口 11：端口 P62 被设置为 PCA2 比较输出端口
1	P61_SEL0	端口 P61 设置寄存器 P61_SEL1,P61_SEL0 00：端口 P61 被设置为 GPIO 01：端口 P61 被设置为 RTC 脉冲输出 1 端口 10：端口 P61 被设置为 ISO7816 I/O 端口 11：端口 P61 被设置为 PCA1 比较输出端口
0	P60_SEL0	端口 P60 设置寄存器 P60_SEL1,P60_SEL0 00：端口 P60 被设置为 GPIO 01：端口 P60 被设置为 RTC 脉冲输出 0 端口 10：端口 P60 被设置为 ISO7816 CLK 时钟端口 11：端口 P60 被设置为 PCA0 比较输出端口

Figure 16-24 P6_SEL1 寄存器

16.26 端口 P7 功能配置寄存器 0 P7_SEL0

P7_SEL0								
P7 数字端口/模拟端口配置寄存器0								
地址空间： 0x010030								
位	7	6	5	4	3	2	1	0
符号	--	--	--	--	P73_SEL0	P72_SEL0	P71_SEL0	P70_SEL0
类型	--	--	--	--	R/W	R/W	R/W	R/W
复位值	--	--	--	--	0	0	0	0
设置P7x_SEL0不会自动设置端口的输入输出								
位	符号	描述						
7:4	--	保留位						
3	P73_SEL0	端口 P73 功能配置寄存器 0 0：功能由 P7_SEL1 决定 1：LCD Segment 28						
2	P72_SEL0	端口 P72 功能配置寄存器 0 0：功能由 P7_SEL1 决定 1：LCD Segment 29						
1	P71_SEL0	端口 P71 功能配置寄存器 0 0：功能由 P7_SEL1 决定 1：LCD Segment 30						
0	P70_SEL0	端口 P70 功能配置寄存器 0 0：功能由 P7_SEL1 决定 1：LCD Segment 31						

Figure 16-25 P7_SEL0 寄存器

16.27 端口 P7 功能配置寄存器 1 P7_SEL1

P7_SEL1								
P7 数字端口功能配置寄存器1								
地址空间： 0x010032								
位	--	--	--	--	3	2	1	0
符号	--	--	--	--	P73_SEL1	P72_SEL1	P71_SEL1	P70_SEL1
类型	--	--	--	--	R/W	R/W	R/W	R/W
复位值	--	--	--	--	0	0	0	0

设置P7x_SEL0不会自动设置端口的输入输出

位	符号	描述
7:4	--	保留位
3	P73_SEL1	端口 P73 功能配置寄存器 1 0：端口 P73 被设置为 GPIO 1：端口 P73 被设置为 PCA 比较输出 3
2	P72_SEL1	端口 P72 功能配置寄存器 1 0：端口 P72 被设置为 GPIO 1：端口 P72 被设置为 PCA 比较输出 2
1	P71_SEL1	端口 P71 功能配置寄存器 1 0：端口 P71 被设置为 GPIO 1：端口 P71 被设置为 PCA 比较输出 1
0	P70_SEL1	端口 P70 功能配置寄存器 1 0：端口 P70 被设置为 GPIO 1：端口 P70 被设置为 PCA 比较输出 0

Figure 16-26 P7_SEL1 寄存器

16.28 端口 P8 功能配置寄存器 0 P8_SEL0

P8_SEL0								
P8 数字端口/模拟端口配置寄存器0								
地址空间： 0x010050								
位	7	6	5	4	3	2	1	0
符号	--	--	--	--	P83_SEL0	P82_SEL0	P81_SEL0	P80_SEL0
类型	--	--	--	--	R/W	R/W	R/W	R/W
复位值	--	--	--	--	0	0	0	0
设置P8x_SEL0不会自动设置端口的输入输出								
位	符号	描述						
3	P83_SEL0	端口 P87 功能配置寄存器 0 0：通用输入输出引脚 1：LCD Segment 16						
2	P82_SEL0	端口 P87 功能配置寄存器 0 0：通用输入输出引脚 1：LCD Segment 17						
1	P81_SEL0	端口 P87 功能配置寄存器 0 0：通用输入输出引脚 1：LCD Segment 18						
0	P80_SEL0	端口 P87 功能配置寄存器 0 0：通用输入输出引脚 1：LCD Segment 19						

Figure 16-27 P8_SEL0 寄存器

16.29 功能模块共享引脚优先级设定

定时器 0 门控输入	寄存器 VC3 的设置	寄存器 P5_SEL1[1]/ P5_SEL0[0]设置值	寄存器 P2_SEL2[6]/ P2_SEL1[6]/ P2_SEL0[6]设置值
信号来源于 VC 比较器输出	VC3[1]=1; VC3[0]=0	xx	xx
信号来源于 VC 比较器输出取反后的值	VC3[1]=1; VC3[0]=1	xx	xx
信号来源于 P50 管脚	VC3[1]=0; VC3[0]=x	!=11	!=100
信号来源于 P26 管脚		!=11	=100
信号来源于 P50 管脚		=11	!=100
信号来源于 P50 管脚		=11	=100

定时器 1 门控输入	寄存器 VC3 的设置	寄存器 P1_SEL2[0]/ P1_SEL1[0]/P1_SEL0[0] 设置值	寄存器 P2_SEL2[0]/ P2_SEL1[0]/P2_SEL0[0] 设置值
信号来源于 VC 比较器输出	VC3[1]=1/VC3[0]=0	xx	xx
信号来源于 VC 比较器输出取反后的值	VC3[1]=1/ VC3[0]=1	xx	xx
信号来源于 P10 管脚	VC3[1]=0/VC3[0]=x	!=110	!=110
信号来源于 P20 管脚		!=110	=110
信号来源于 P10 管脚		=110	!=110
信号来源于 P10 管脚		=110	=110

定时器 0 的时钟信号来源	寄存器 P5_SEL1[1]/ P5_SEL0[1]设置值	寄存器 P2_SEL2[7]/P2_SEL1[7]/ P2_SEL0[7]设置值
信号来源于 P51 管脚	!=11	!100
信号来源于 P27 管脚	!=11	=100
信号来源于 P51 管脚	=11	!=100
信号来源于 P51 管脚	=11	=100

定时器 1 的时钟信号来源	寄存器 P1_SEL2[1]/ P1_SEL1[1]/P1_SEL0[1]设置值	寄存器 P2_SEL2[1]/ P2_SEL1[1]/P2_SEL0[1]设置值
信号来源于 P11 管脚	!=110	!=110
信号来源于 P21 管脚	!=110	=110
信号来源于 P11 管脚	=110	!=110
信号来源于 P11 管脚	=110	=110

PCA 时钟信号的来源	寄存器 P1_SEL2[2]/ P1_SEL1[2]/P1_SEL0[2]设置值	寄存器 P2_SEL2[2]/ P2_SEL1[2]/P2_SEL0[2]设置值
信号来源于 P12 管脚	!=110	!=110
信号来源于 P22 管脚	!=110	=110
信号来源于 P12 管脚	=110	!=110
信号来源于 P12 管脚	=110	=110

PCACAPT0 输入信号的来源	寄存器 P1_SEL2[3]/ P1_SEL1[3]/P1_SEL0[3]设置值	寄存器 P2_SEL2[3]/ P2_SEL1[3]/P2_SEL0[3]设置值
信号来源于 P13 管脚	!=110	!=110
信号来源于 P23 管脚	!=110	=110
信号来源于 P13 管脚	=110	!=110
信号来源于 P13 管脚	=110	=110

PCACAPT1 输入信号的来源	寄存器 P1_SEL2[4]/ P1_SEL1[4]/P1_SEL0[4]设置值	寄存器 P2_SEL2[4]/ P2_SEL1[4]/P2_SEL0[4]设置值
信号来源于 P14 管脚	!=110	!=110
信号来源于 P24 管脚	!=110	=110
信号来源于 P14 管脚	=110	!=110
信号来源于 P14 管脚	=110	=110

PCACAPT2 输入信号的来源	寄存器 P1_SEL2[5]/ P1_SEL1[5]/P1_SEL0[5]设置值	寄存器 P2_SEL2[5]/ P2_SEL1[5]/P2_SEL0[5]设置值
信号来源于 P15 管脚	!=110	!=110
信号来源于 P25 管脚	!=110	=110
信号来源于 P15 管脚	=110	!=110
信号来源于 P15 管脚	=110	=110

PCACAPT3 输入信号的来源	寄存器 P1_SEL2[6]/ P1_SEL1[6]/P1_SEL0[6]设置值	寄存器 P2_SEL2[6]/ P2_SEL1[6]P2_SEL0[6]设置值
信号来源于 P16 管脚	!=110	!=110
信号来源于 P26 管脚	!=110	=110
信号来源于 P16 管脚	=110	!=110
信号来源于 P16 管脚	=110	=110

PCACAPT4 输入信号的来源	寄存器 P1_SEL2[7]/ P1_SEL1[7]/P1_SEL0[7]设置值	寄存器 P2_SEL2[7]/ P2_SEL1[7]/P2_SEL0[7]设置值
信号来源于 P17 管脚	!=110	!=110
信号来源于 P27 管脚	!=110	=110
信号来源于 P17 管脚	110	!=110
信号来源于 P17 管脚	=110	=110

SPI 接口的 CS 信号来源	寄存器 P1_SEL2[5]/ P1_SEL1[5]P1_SEL0[5]设置值	寄存器 P2_SEL2[4]/ P2_SEL1[4]/P2_SEL0[4]设置值
信号来源于 P15 管脚	!=010	!=010
信号来源于 P24 管脚	!=010	=010
信号来源于 P15 管脚	=010	!=010
信号来源于 P15 管脚	=010	=010

SPI 接口的 CLK 信号来源	寄存器 P1_SEL2[4]/ P1_SEL1[4]/P1_SEL0[4]设置值	寄存器 P2_SEL2[5]/ P2_SEL1[5]/P2_SEL0[5]设置值
信号来源于 P14 管脚	!=010	!=010
信号来源于 P25 管脚	!=010	=010
信号来源于 P14 管脚	=010	!=010
信号来源于 P14 管脚	=010	=010

SPI 接口的“MOSI” 信号来源	寄存器 P1_SEL2[3]/ P1_SEL1[3]/P1_SEL0[3]设置值	寄存器 P2_SEL2[6]/ P2_SEL1[6]/P2_SEL0[6]设置值
信号来源于 P13 管脚	!=010	!=010
信号来源于 P26 管脚	!=010	=010
信号来源于 P13 管脚	=010	!=010
信号来源于 P13 管脚	=010	=010

SPI 接口的“MISO” 信号来源	寄存器 P1_SEL2[2]/ P1_SEL1[2]/P1_SEL0[2]设置值	寄存器 P2_SEL2[7]/ P2_SEL1[7]/P2_SEL0[7]设置值
信号来源于 P12 管脚	!=010	!=010
信号来源于 P27 管脚	!=010	=010
信号来源于 P12 管脚	=010	!=010
信号来源于 P12 管脚	=010	=010

I^2C 接口的 SCL 信号来源	寄存器 P1_SEL2[0]/ P1_SEL1[0]/P1_SEL0[0]设置值	寄存器 P4_SEL1[6]/ P4_SEL0[6]设置值
信号来源于 P10 管脚	!=010	!=10
信号来源于 P46 管脚	!=010	=10
信号来源于 P10 管脚	=010	!=10
信号来源于 P10 管脚	=010	=10

I^2C 接口的 SDO 信号来源	寄存器 P1_SEL2[1]/ P1_SEL1[1]/P1_SEL0[1]设置值	寄存器 P4_SEL1[7]/ P4_SEL0[7]设置值
信号来源于 P11 管脚	!=010	!=10
信号来源于 P47 管脚	!=010	=10
信号来源于 P11 管脚	=010	!=10
信号来源于 P11 管脚	=010	=10

ISO7816 数据输入信号来源	寄存器 P6_SEL1[1]/ P6_SEL0[1]设置值	寄存器 P1_SEL2[6]/ P1_SEL1[6]/P1_SEL0[6] 设置值	寄存器 P4_SEL1[4]/ P4_SEL0[4]设置值
信号来源于 P61 管脚	=10	=100	=10
信号来源于 P61 管脚	=10	=100	!=10
信号来源于 P61 管脚	=10	!=100	=10
信号来源于 P61 管脚	=10	!=100	!=10
信号来源于 P16 管脚	!=10	=100	=10
信号来源于 P16 管脚	!=10	=100	!=10
信号来源于 P44 管脚	!=10	!=100	=10
信号来源于 P61 管脚	!=10	!=100	!=10

Figure 16-28 HC16Lxx 端口配置优先级

17 定时器/计数器

HC16Lxx 提供了4个专用的16位定时/计数器。定时器0、 定时器1 支持标准的8xC251的4种工作模式（详见Figure17-1 定时器0/1的工作模式），与标准80C51兼容。另外定时器2、定时器3支持标准8xC251的2种工作模式（详见Figure17-2 定时器2/3的工作模式）。每个定时器/计数器都有独立的控制启动信号，以及外部输入时钟，门控信号。

定时/计数器0使用端口TM0CLK,TM0G来进行计数功能，其中TM0CLK被用于计数器的输入时钟信号，TM0G被用于高电平计数使能信号。同样定时/计数器1使用TM1CLK,TM1G来进行计数功能。

- 定时模式：

  当使用中断定时器功能时，可以通过编程使定时器在特定的时间间隔发生溢出中断并置位溢出标志位。

  为了兼容标准80C51和8xC251的12个时钟周期，寄存器[THx:TLx]保留12个时钟周期累加一次。

- 计数模式：

  计数模式用于测定某个事件发生的次数，而不是测量事件之间的时间间隔。在计数模式中，寄存器[THx:TLx]在每个相应的输入时钟的下降沿累加一次。

  为了兼容标准80C51和8xC251的12个时钟周期，输入时钟信号12个时钟周期采样一次。因此外部输入时钟频率不能超过系统时钟的1/12。

17.1 定时/计数器

每个计数器/定时器都是一个16位的寄存器，在被访问时以两个字节的形式出现：一个低字节（TL0或TL1）和一个高字节（TH0或TH1）。计数器/定时器控制寄存器（TCON）用于允许定时器0和定时器1以及指示它们的状态。

通过将IE寄存器中的ET0位置‘1’来允许定时器0,定时器2中断；通过将ET1位置‘1’来允许定时器1，定时器3中断。

定时器0/1支持4种工作方式，通过设置计数器/定时器方式寄存器（TMOD）中的方式选择位M11/M01和M10/M00来选择工作方式，每个定时器都可以被独立配置。下面对每种工作方式进行详细说明。

模式	工作模式
模式0	13位定时/计数器
模式1	16位定时/计数器
模式2	溢出自动加载 定时/计数器
模式3	定时器0：2个8位定时/计数器 定时器1：停止工作

Figure 17-1 定时器0/1的工作模式

定时器2/3支持2种工作方式（Figure 17-2 中的模式1和模式2），通过设置计数器/定时器方式寄存器（TCON2）中的方式选择位TMM2, TMM3来选择工作方式，每个定时器都可以被独立配置。下面对每种工作方式进行详细说明。

模式	工作模式
模式1	16位定时/计数器
模式2	溢出自动加载 定时/计数器

Figure 17-2 定时器2/3的工作模式

17.2 模式 0（13 位计数器/定时器 ）

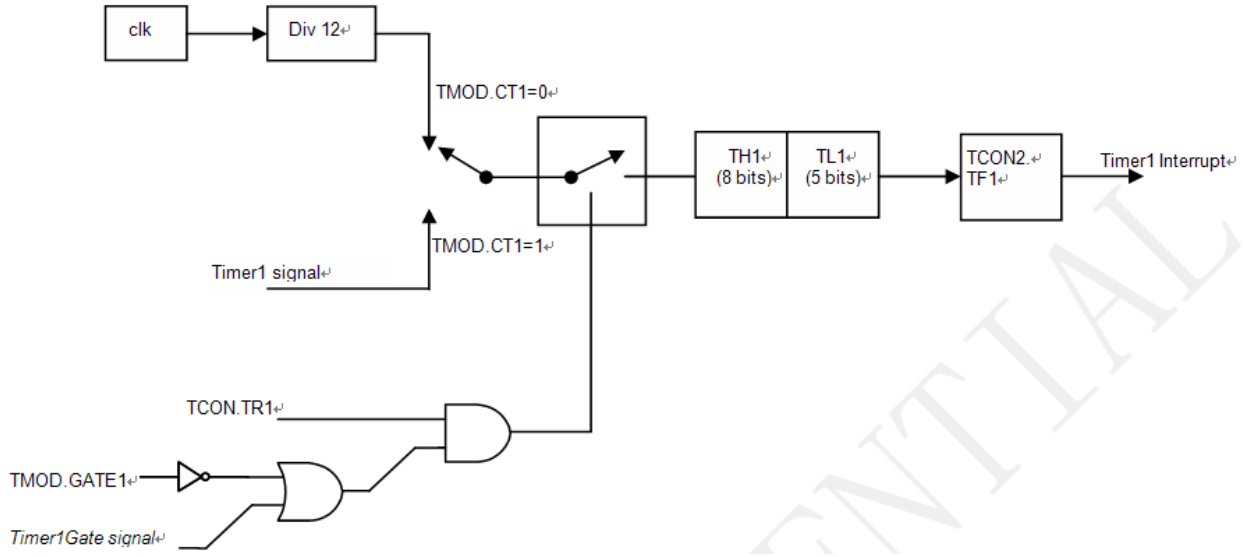


Figure 17-3 13位计数器/定时器工作模式

TH0寄存器保持13位计数器/定时器的8个MSB，TL0在TL0[4:0]位置保持5个LSB。TL0的高3位（TL0[7:5]）是不确定的，在读计数值时应屏蔽掉或忽略这3位。作为13位定时器寄存器，当计数到0x1FFF（13’b1_1111_1111_1111）后再计一次将发生溢出，使计数值回到0x0000，此时定时器溢出标志TF0（TCON2[4]）被置位并产生一个中断（如果该中断被允许）。

注意事项：

置位TR0不会强制定时器复位。但在启动定时器之前，需将定时器寄存器装入所需要的初值。

17.3 模式 1（16 位计数器/定时器）

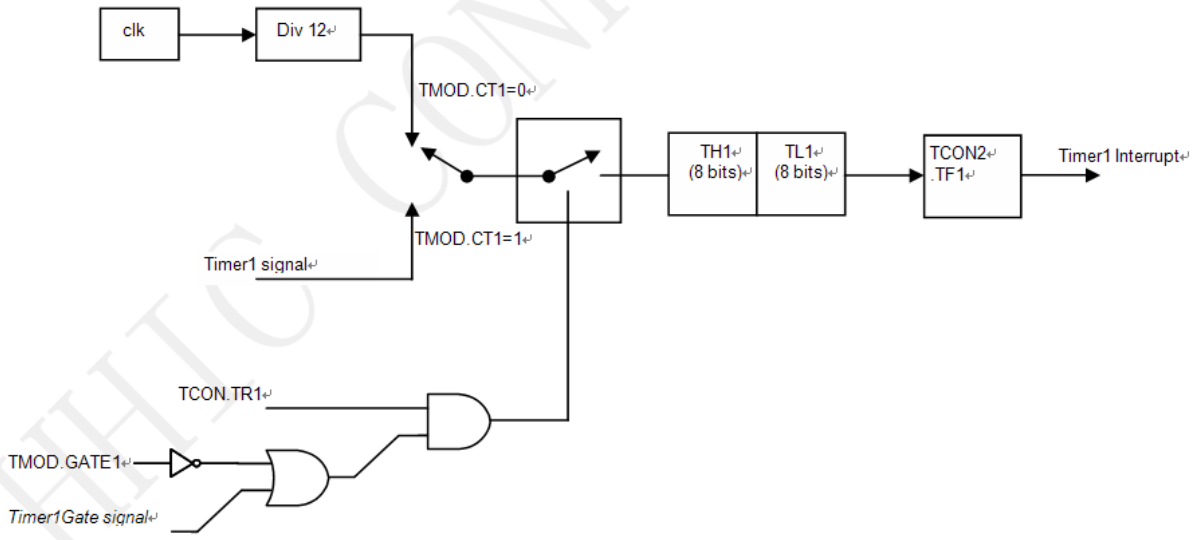


Figure 17-4 16位计数器/定时器工作模式

模式1的操作与模式0完全一样，所不同的是计数器/定时器使用全部16位。

可以用与模式0相同的方法控制计数器/定时器在模式1下工作。

17.4 模式 2（溢出自动加载 定时/计数器）

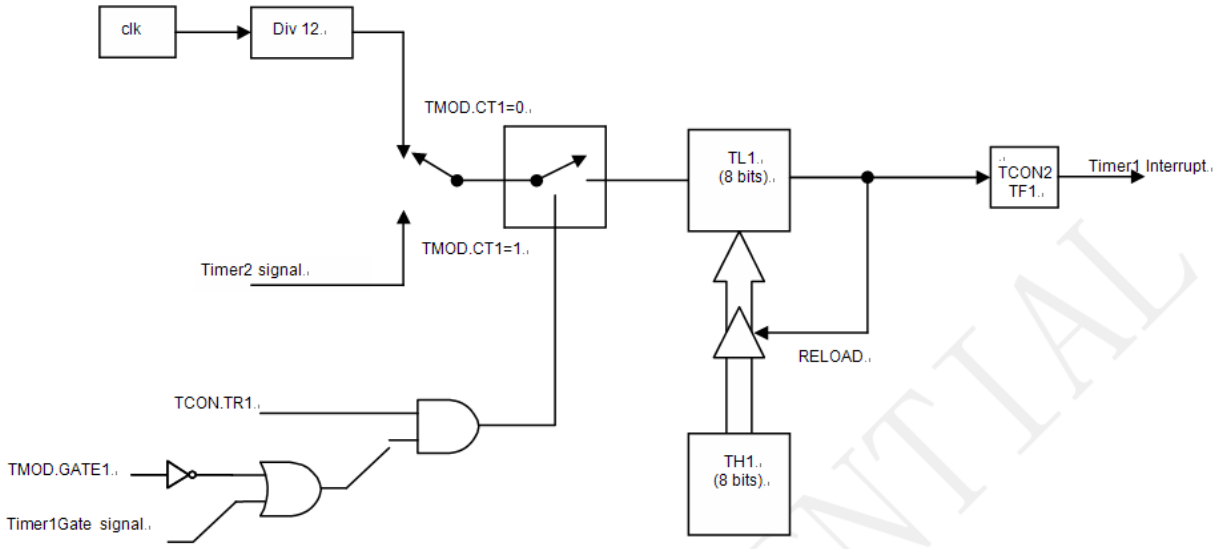


Figure 17-5 溢出自动加载 定时/计数器 工作模式

模式2将定时器0/1配置为具有自动重新装入计数初值能力的8位计数器/定时器。TL0保持计数值，而TH0保持重载值。当TL0中的计数值发生溢出（从0xFF到0x00）时，定时器溢出标志TCON2.TM0_OVF被置位，TH0中的重载值被重新装入到TL0。如果中断被允许，在TF0被置位时将产生一个中断。TH0中的重载值保持不变。为了保证第一次计数正确，必须在允许定时器之前将TL0初始化为所希望的计数初值。

17.5 模式 3（2 个 8 位定时/计数器）

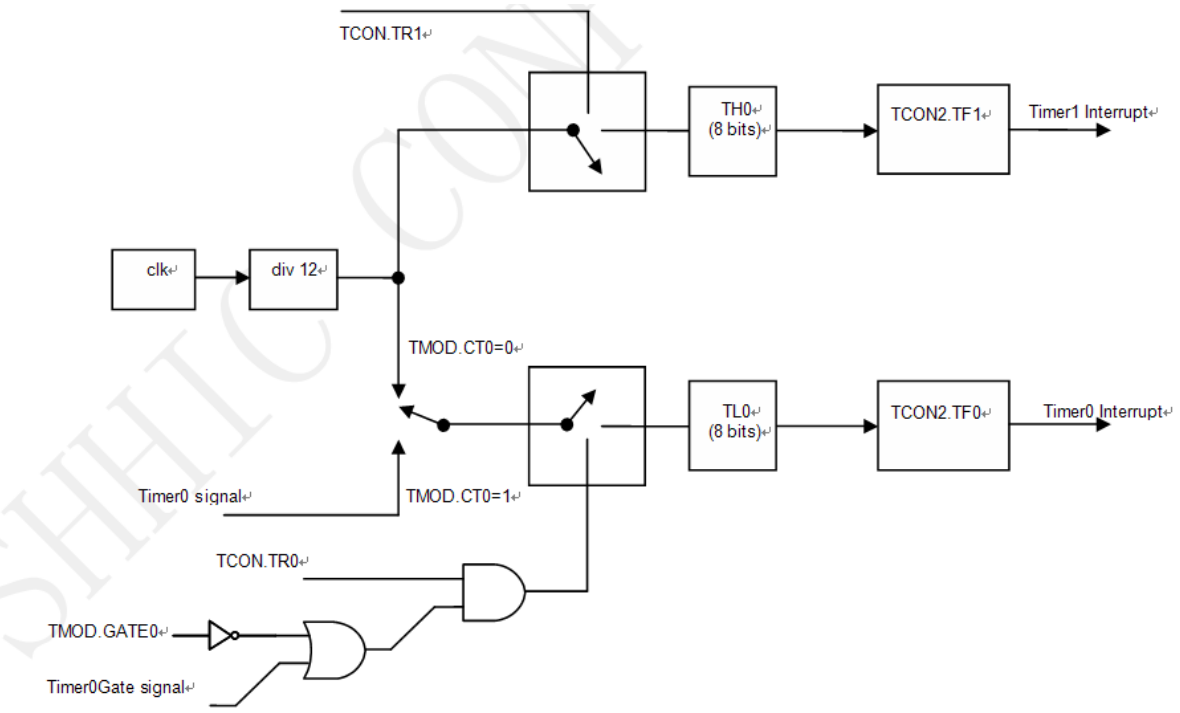


Figure 17-6 2个8位定时/计数器工作模式

在模式3时，定时器0被配置两个独立的8位定时器/计数器，计数值分别在TL0和TH0中。在TL0中的计数器/定时器使用TCON和TMOD中定时器0的控制/状态位：TR0、CT0、GATE0和TM0_OVF。TL0既可以使用系统时钟也可以使用一个外部输入信号作为时基。TH0寄存器只能作为定时器使用，由系统时钟1/12分频时钟提供时钟。TH0使用定时器1的运行控制位TR1，并在发生溢出时将定时器1的溢出标志位TM1_OVF置1，共享定时器1的中断。

17.6 定时/计数器寄存器

17.7 定时器 0/1 控制寄存器（TCON）

TCON								
定时器0/1控制寄存器								
地址空间： S:88h								
位	7	6	5	4	3	2	1	0
符号	--	TR1	--	TR0	IE1_	IT1	IE0_	IT0
类型	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	0	0	0	0	0	0

位	符号	描述
7	--	保留位
6	TR1	计数器 1 运行控制位 1：启动计数器 1 0：关闭计数器 1
5	--	保留位
4	TR0	计数器 0 运行控制位 1：启动计数器 0 0：关闭计数器 0
3	IE1_	HC16Lxx 中无需此标志位，标志位在 P1_IFG[6]。
2	IT1	标准 80C251/80C51 中，选择下降沿或低电压触发 HC16Lxx 必须选择 0
1	IE0_	HC16Lxx 中无需此标志位，标志位在 P5_IFG[0]。
0	IT0	标准 80C251/80C51 中，选择下降沿或低电压触发 HC16Lxx 必须选择 0

Figure 17-7 定时器0/1控制寄存器（TCON）

17.8 定时器 2/3 控制寄存器（TCON2）

TCON2								
定时器2/3控制寄存器								
地址空间： S:C3h								
位	7	6	5	4	3	2	1	0
符号	TM3_OVF	TM2_OVF	TM1_OVF	TM0_OVF	TMM3	TR3	TMM2	TR2
类型	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	0	0	0	0	0	0

位	符号	描述
7	TM3_OVF	Timer3 计数溢出标志位
6	TM2_OVF	Timer2 计数溢出标志位
5	TM1_OVF	Timer1 计数溢出标志位
4	TM0_OVF	Timer0 计数溢出标志位
3	TMM3	Timer3 工作模式选择 0：16 位计数器工作模式 1：8位自动重加载工作模式
2	TR3	Timer3 启动控制位 1：Timer3启动计数功能 0：Timer3停止计数功能
1	TMM2	Timer2 工作模式选择 0：16 位计数器工作模式 1：8位自动重加载工作模式
0	TR2	Timer2 启动控制位 1：Timer2启动计数功能 0：Timer2停止计数功能

Figure 17-8 定时器2/3控制寄存器（TCON2）

17.9 定时器模式选择寄存器（TMOD）

TMOD								
定时器模式选择寄存器								
地址空间： S:89h								
位	7	6	5	4	3	2	1	0
符号	GATE1	CT1	M11	M01	GATE0	CT0	M10	M00
类型	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	0	0	0	0	0	0

位	符号	描述
7	GATE1	定时器1门控位 0：当TR1=1时，定时器1工作，与外部门控输入（P10或P20）无关 1：只有当TR1=1，并且外部门控输入(P10或P20)为1时，定时器1工作
6	CT1	定时器 1 功能选择 0：定时器功能：定时器 1 由系统时钟/12 来进行计数 1：计数器功能：计数器 1 由外部输入（P11 或 P21）下降沿进行计数
5 4	M11 M01	M11 M01 计数器/定时器 1 工作方式 0 0 模式 0：13 位计数器/定时器 0 1 模式 1：16 位计数器/定时器 1 0 模式 2：溢出自动加载 8 位计数器/定时器 1 1 模式 3：定时器 1 停止运行
3	GATE0	定时器0门控位 0：当TR0=1时，定时器1工作，与外部门控输入（P50或P26）无关 1：只有当TR0=1，并且外部门控输入(P50或P26)为1时，定时器1工作
2	CT0	定时器 0 功能选择 0：定时器功能：定时器 1 由主频/12 来进行计数 1：计数器功能：计数器 1 由外部输入（P51 或 P27）下降沿进行计数
1 0	M10 M00	M10 M00 计数器/定时器 0 工作方式 0 0 模式 0：13 位计数器/定时器 0 1 模式 1：16 位计数器/定时器 1 0 模式 2：溢出自动加载 8 位计数器/定时器 1 1 模式 3： 2 个 8 位计数器/定时器

Figure 17-9定时器模式选择寄存器（TMOD）

17.10 定时器 0 高 8 位寄存器（TH0）

TH0								
定时器0高8位寄存器								
地址空间： S:8Ch								
位	7	6	5	4	3	2	1	0
符号	TH0[7:0]							
类型	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	0	0	0	0	0	0
位	符号	描述						
7:0	TH0[7:0]	定时器0 高8位寄存器						

Figure 17-10 定时器0高8位寄存器（TH0）

17.11 定时器 0 低 8 位寄存器（TL0）

TL0								
定时器0低8位寄存器								
地址空间： S:8Ah								
位	7	6	5	4	3	2	1	0
符号	TL0[7:0]							
类型	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	0	0	0	0	0	0
位	符号	描述						
7:0	TL0[7:0]	定时器0 低8位寄存器						

Figure 17-11 定时器0低8位寄存器（TL0）

17.12 定时器 1 高 8 位寄存器（TH1）

TH1								
定时器1高8位寄存器								
地址空间： S:8Dh								
位	7	6	5	4	3	2	1	0
符号	TH1[7:0]							
类型	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	0	0	0	0	0	0
位	符号	描述						
7:0	TH1[7:0]	定时器1 高8位寄存器						

Figure 17-12 定时器1高8位寄存器（TH1）

17.13 定时器 1 低 8 位寄存器（TL1）

TL1								
定时器1低8位寄存器								
地址空间： S:8Bh								
位	7	6	5	4	3	2	1	0
符号	TL1[7:0]							
类型	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	0	0	0	0	0	0
位	符号	描述						
7:0	TL1[7:0]	定时器1 低8位寄存器						

Figure 17-13 定时器1低8位寄存器（TL1）

17.14 定时器 2 高 8 位寄存器（TH2）

TH2								
定时器2高8位寄存器								
地址空间： S:C5h								
位	7	6	5	4	3	2	1	0
符号	TH2[7:0]							
类型	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	0	0	0	0	0	0
位	符号	描述						
7:0	TH2[7:0]	定时器2 高8位寄存器						

Figure 17-14 定时器2高8位寄存器（TH2）

17.15 定时器 2 低 8 位寄存器（TL2）

TL2								
定时器2低8位寄存器								
地址空间： S:C4h								
位	7	6	5	4	3	2	1	0
符号	TL2[7:0]							
类型	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	0	0	0	0	0	0
位	符号	描述						
7:0	TL2[7:0]	定时器2 低8位寄存器						

Figure 17-15 定时器2低8位寄存器（TL2）

17.16 定时器 3 高 8 位寄存器（TH3）

TH3								
定时器3高8位寄存器								
地址空间： S:C7h								
位	7	6	5	4	3	2	1	0
符号	TH3[7:0]							
类型	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	0	0	0	0	0	0
位	符号	描述						
7:0	TH3[7:0]	定时器3 高8位寄存器						

Figure 17-16 定时器3高8位寄存器（TH3）

17.17 定时器 3 低 8 位寄存器（TL3）

TL3								
定时器3低8位寄存器								
地址空间： S:C6h								
位	7	6	5	4	3	2	1	0
符号	TL3[7:0]							
类型	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	0	0	0	0	0	0
位	符号	描述						
7:0	TL3[7:0]	定时器3 低8位寄存器						

Figure 17-17 定时器3低8位寄存器（TL3）

18 可编程计数阵列(PCA)

18.1 PCA 简介

PCA(可编程计数器阵列Programmable Counter Array)支持最多 5 个 16 位的捕获/比较模块和一个16位的定时/计数器。该定时/计数器可用作捕获/比较模块的一个通用的时钟计数/事件计数器。PCA的每个模块都可以进行独立编程，以提供输入捕捉，输出比较或脉冲宽度调制。另外模块4有额外的看门狗定时器模式。与标准80C51的定时/计数器相比，他需要较少的CPU干预。

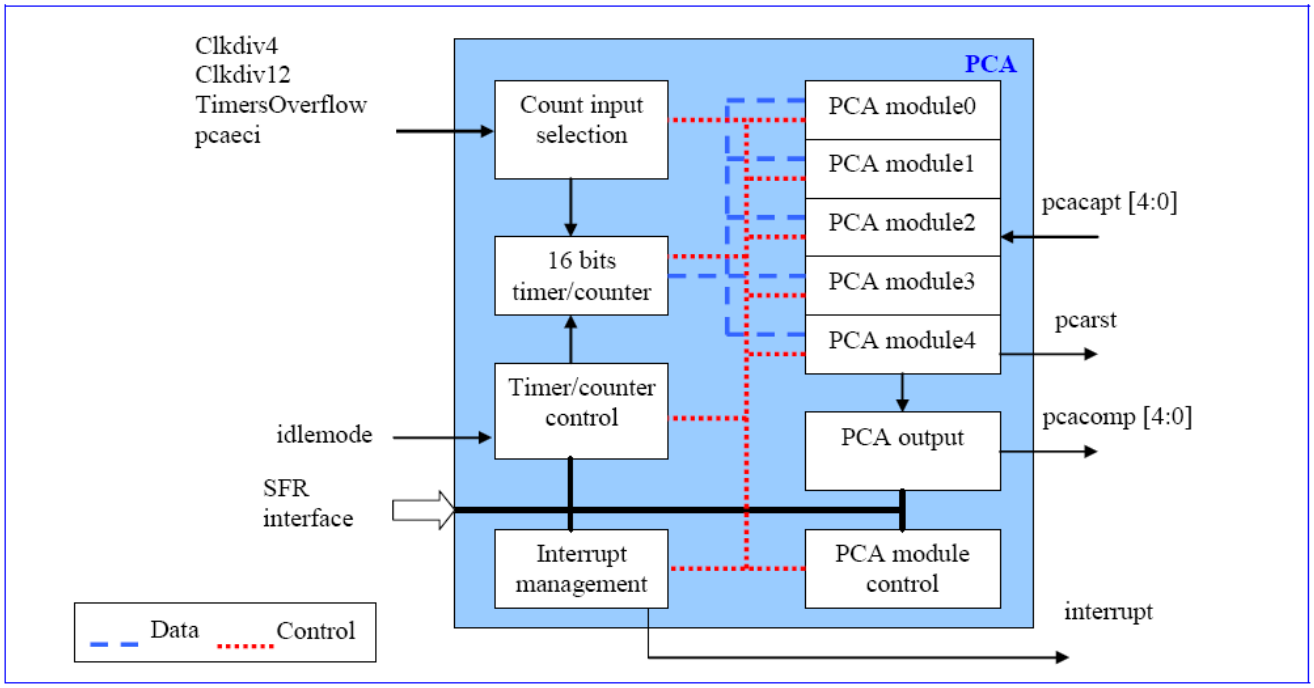


Figure 18-1 PCA原理框图

18.2 PCA 定时/计数器

CH/CL 这2组特殊功能寄存器可用作为一个16位定时器/计数器。CL（低字节）寄存器会根据选择的输入信号递增，当CL（低字节）寄存器溢出时，高位 CH（高字节）寄存器递增。如果CMOD.ECF位被置“1”，当CH溢出时硬件自动设置PCA溢出标志（CCON.CF）并产生PCA中断请求。通过CMOD.CPS1位和CMOD.CPS0位选择以下四个信号中的一个输入到定时器/计数器。

- 系统时钟的12分频。
- 系统时钟的4分频。
- 定时器0的溢出。每次定时器0计数溢出后，CL就递增，这样提供了PCA的可变编程频率输入。
- ECI。CPU每过4个时钟周期就对PCAECI进行采样，当每次采样结果从高变低时，CL自动加1，因此最高的ECI输入频率不能高于系统时钟的1/8,以满足采样需求。

设置运行控制器（CCON.CR）启动PCA定时/计数器。当CMOD.CIDL置“1”后，PCA定时器/计数器可以继续运行在空闲模式下。CPU可以随时读取CH,CL的数值,但当计数启动后(CCON.CR=1)时，为了防止计数错误，CH,CL是禁止写入的。

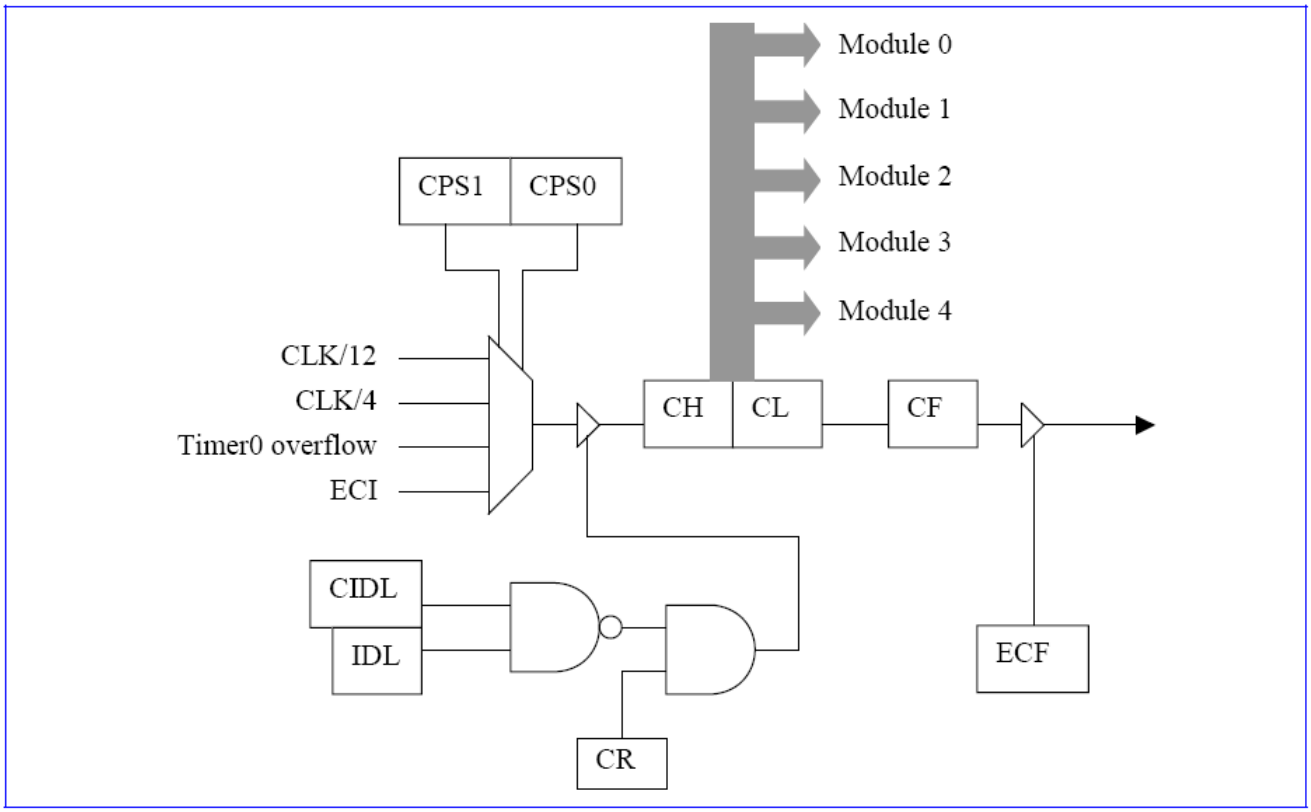


Figure 18-2 PCA定时/计数模式

18.3 PCA 工作模式配置

每个模块都可被配置为独立工作，有五种工作方式：边沿触发捕捉、软件定时器、高速输出、频率输出(看门狗定时器)、8位脉宽调制器。每个模块在系统控制器中都有属于自己的特殊功能寄存器（SFR），这些寄存器用于配置模块的工作方式和与模块交换数据。

每组比较/捕获模块是由一组比较/捕获寄存器组(CCAPxH,CCAPxL)，以及1个16位比较器，和各种逻辑门控制组成。寄存器组用来存储时间或次数，针对外部触发捕获条件，或内部触发比较条件。在PWM模式下，低字节寄存器用来控制输出波形的占空比。

每个模块都可以独立编程的操作在以下三种模式：

•16位边沿触发捕获模式:上升沿，下降沿或任意沿触发。

•16位比较模式：16位软件定时器，16位高速输出，16位看门狗定时器（模块4）或8位脉冲宽度调制。

•未启动。

设置比较/捕获模块模式寄存器（CCAPMx）确定相应的工作模式。对比较/捕获模块进行编程时，他们是基于共同的时间计数。定时器/计数器的打开和关闭通过CCON.CR位即可控制。在一个比较/捕获模块运用捕获，软件定时器或高速输出功能时，设置模块的比较/捕获标志（CCON.CCFx）后，如果已经在CCAPMx寄存器设置了相应的使能位,那么就会产生PCA中断请求。

18.4 16 位边沿触发捕获模式

PCA捕获模式提供了测量PCA的周期，脉冲宽度，占空比和相位差的功能,一共有5个通道。

引脚上出现的电平跳变会触发PCA捕捉PCA计数器/定时器的值并将其载入到对应模块的16位捕捉/比较寄存器（CCAPxH/CCAPxL）中。CCAPMX.CAPPx 以及 CCAPMx.CAPNx位用于选择触发捕捉的电平变化类型：低电平到高电平（正沿）、高电平到低电平（负沿）或任何变化（正沿或负沿）。

当捕捉发生时，CCON中的捕捉/比较标志（CCFn）被置为逻辑‘1’并产生一个中断请求（如果CCF中断被允许）。当CPU转向中断服务程序时，CCFn位不能被硬件自动清除，需要用软件写0清除这个标志位。

捕获的分辨率等于定时器/计数器的时钟。输入信号必须在高电平或低电平期间至少保持2个时钟周期，以保证输入信号能够被硬件识别。

CPU可以在任何时候读取或写入CCAPxH和CCAPxL寄存器。

捕获设置：

当需要在外部上升沿进行捕获，CCPMX.CAPPx =“1”以及 CCAPMx.CAPNx =“0”

当需要在外部下降沿进行捕获，CCPMX.CAPPx =“0”以及 CCAPMx.CAPNx =“1”

当需要在外部上升、下降沿进行捕获，CCPMX.CAPPx =“1”以及 CCAPMx.CAPNx =“1”

注意事项：同一模块新的捕获值会覆盖现有捕获的值。为了保持捕获的值，在中断服务程序中将它保存在RAM里面，这个操作必须在下一次事件出现之前完成，否则就会丢失前面一次捕获采样值。

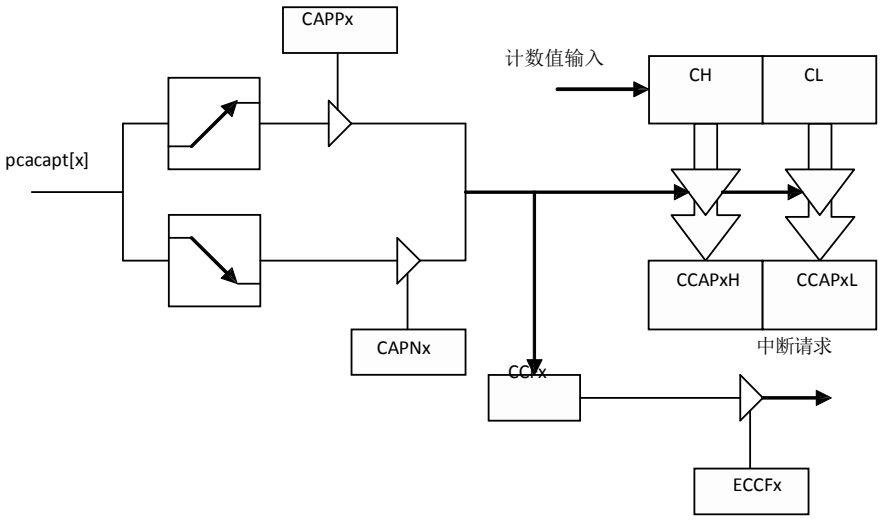


Figure 18-3 PCA 16位捕获模式

18.5 16 位比较模式

比较功能具有四种模式：16位软件定时器模式，高速输出模式，WDT模式和PWM模式。通过设置CCAPMx.ECOMx位选择该模块的比较功能。

在前三个功能中，比较/捕获模块会比较16位PCA定时器/计数器的值与预先加载到该模块的CCAPxH/ CCAPxL寄存器中的16位值。在PWM模式下，PCA模块将PCA定时器/计数器低字节寄存器（CL）与一个在CCAPxL模块寄存器8位的值进行比较。每4个时钟周期比较一次，即与最快的PCA定时器/计数器的时钟速率相匹配。若要正确使用在比较模式下的模块，请遵守以下的程序流程：

1. 选择PCA模块的操作模式
2. 选择PCA定时器/计数器的输入信号。
3. 设定比较值到模块的比较/捕获寄存器。
4. 设置PCA定时器/计数器运行控制位。
5. 匹配后产生中断，清除模块的比较/捕获标志。

18.6 16 位软件计数模式

要设定一个比较/捕获模块到16位软件定时器模式下，需要使能CCAPMx.ECOMx和CCAPMx.MATx位。一旦在PCA定时器/计数器和比较/捕获的寄存器（CCAPxH/ CCAPxL）之间发生了匹配，会自动设置模块的比较/捕获标志（CCON.CCFx）。如果设置了相应的中断使能位（CCAPMx.ECCFx），就将产生一个中断请求。中断处理时，由于硬件不清除比较/捕获标志，需要用软件清除标志。而在中断服务程序中，一个新的16位比较值可以被写入比较/捕获的寄存器（CCAPxH/ CCAPxL）中。

注意事项：在更新这些寄存器时，为了防止无效的匹配发生，应该先写CCAPxL，后写CCAPxH。一旦对CCAPxL进行写操作就会清除ECOMx位，而对CCAPxH进行写操作就会使能ECOMx位，重新启用比较功能。即当向PCA的捕捉/比较寄存器写入一个16位数值时，应先写低字节。

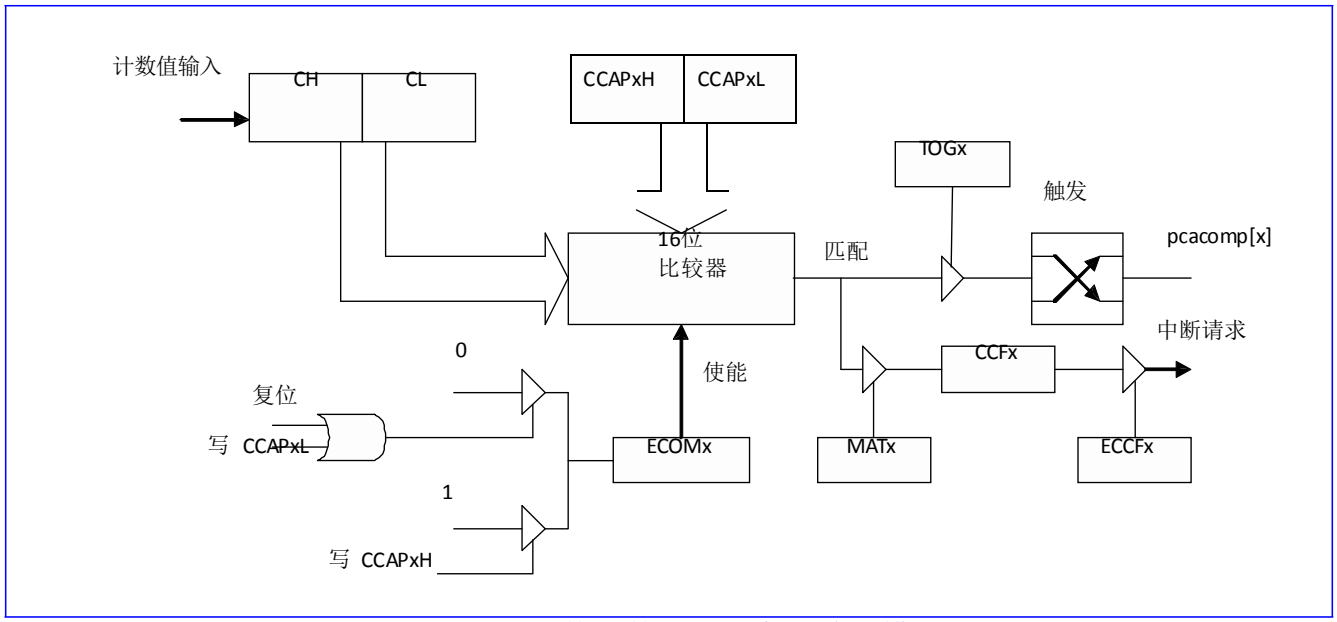


Figure 18-4　PCA 软件计数器以及高速输出模式

18.7 高速输出模式

在高速输出模式，每当PCA计数器内的值与模块的16位捕捉/比较寄存器（CCAPxH/ CCAPxL）的值发生匹配时，模块的PCACOMP[x]信号(即CCPx引脚)上的逻辑电平将发生变化。由于这个高速输出不会被中断响应而影响输出频率，会比切换IO输出拥有更高的精度。

要设定一个比较/捕获模块到高速输出模式，需要使能CCAPMx.ECOMx，CCAPMx.MATx和CCAPMx.TOGx位。 PCA定时器/计数器和比较/捕获的寄存器（CCAPxH/ CCAPxL）之间的匹配发生时可以选择进行如下两种操作:

- 切换PCACOMP[x]信号，并设置模块的比较/捕获标志（CCON.CCFx）位。
  通过软件设置或清除的方式，用户可以选择匹配后切换PCACOMP[x]信号从低到高或高到低。由于硬件无法清除比较/捕获标志，需要用软件清除这个标志位。
- 通过设置相应的中断使能位（CCAPMx.ECCFx）产生一个中断请求。
  如果在中断程序中没有改变比较/捕获寄存器，PCA将重新根据比较值计数，如相匹配则发生下一次操作。在中断服务程序中，一个新的16位比较值可以被写入比较/捕获的寄存器（CCAPxH/ CCAPxL）。

注意事项：更新寄存器时，为了防止无效的匹配发生，应该先写CCAPxL，后写CCAPxH。一旦对CCAPxL进行写操作就会清除ECOMx位，而对CCAPxH进行写操作就会使能ECOMx位，重新启用比较功能。

18.8 PCA 模块 4 的 WDT 功能

HC16Lxx除了一个WDT硬件模块，PCA的模块4还提供一个可编程频率的16位WDT。当PCA定时器/计数器的计数值与模块4中比较/捕获寄存器（CCAP4H/CCAP4L）存储的值相匹配时，将产生复位信号。

PCA的WDT复位信号作为一个独立的复位信号，与外部复位（RST），硬件看门狗复位（WDTRST）LVD低电压复位和POR上电掉电复位相结合，可以被自由组合或单独使用。模块4是唯一具有WDT模式的PCA模块。当不设置为WDT时，它可以在其它模式中独立使用。

当把 PCA模块4当WDT使用时，需要使能CCAPM4.ECOM4，CCAPM4.MAT4以及 CMOD.WDTE位。另外通过设置CMOD.CPS0 和CMOD.CPS1来选择不同的输入计数频率。

在比较/捕获寄存器（CCAP4H/CCAP4L）中输入一个16位的比较值，在 PCA定时器/计数器（CH/ CL）中输入一个16位的初始值或使用复位值（0000h）。这两个值之间的差额乘以PCA输入脉冲频率确定了WDT匹配发生的运行时间。设置定时器/计数器运行控制位（CCON.CR）启动PCA WDT。

每次匹配时，PCA的WDT都会产生复位信号。要防止产生复位，有以下三种方法：

•定期的改变比较值CCAP4H/CCAP4L，匹配不会发生。

•定期更改PCA定时器/计数器值（CH/ CL），匹配不会发生。

•在匹配前清除CMOD.WDTE位来禁用模块复位输出信号，然后再重新启用它。

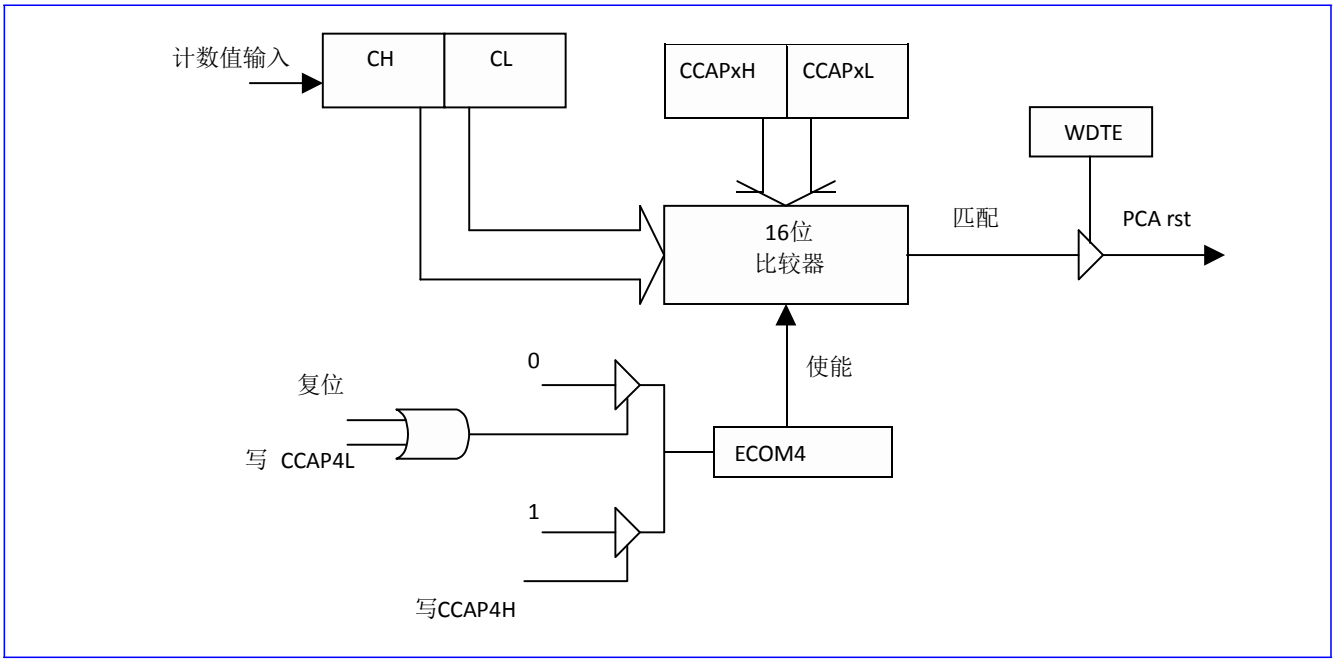


Figure 18-5 PCA WDT功能

18.9 PCA 8 位脉宽调制功能

脉宽调制是一种使用程序来控制波形占空比，周期，相位的技术。5个PCA模块都可以被独立地用于在对应的PCACOMP[x]引脚产生脉宽调制（PWM）输出，脉冲宽度为8位分辨率，而输出的频率取决于PCA计数器/定时器的时间基准。使用模块的捕捉/比较寄存器CCAPxL可以改变PWM输出信号的占空比。

当PCA计数器/定时器的低字节（CL）与CCAPxL中的值相等时，PCACOMP[x]引脚上的输出被置“1”；当CL中的计数值溢出时（从0xFF到0x00），PCACOMP[x]输出被复位“0”。同时保存在PCCAPxH中的值被自动装入到CCAPxL，不需软件干预。

在这种模式下，PCA定时器/计数器（CL）的低字节中的值是不间断地与比较/捕获寄存器（CCAPxL）中的值相比较。当CL < CCAPxL时，输出波形为低；从两者匹配（CL= CCAPxL）直到CL溢出(从 FFH 到 00H)，输出波形为高；溢出时，CCAPxH的值自动装载到CCAPxL内，一个新的周期开始。

CCAPxL的值决定当前波形的占空比，CCAPxH的值确定下一个波形的占空比。改变CCAPxL中的值可更改脉冲宽度调制。正如图所示，在CCAPxL中8位数值的范围是从0（100％占空比）到255（0.4％占空比）。要改变CCAPxL值而不产生毛刺，需要在高字节寄存器（CCAPxH）写入一个新值。当CL超过0xFF翻转到0x00，这个值由硬件自动加载到CCAPxL。

要设定一个比较/捕获模块到PWM模式下，需要使能CCAPMx.ECOMx和CCAPMx.PWMx位。另外通过配置CMOD.CSP0 和CMOD.CSP1可以选择输入计数信号频率。在CCAPxL和CCAPxH输入一个8位的值指定第一个以及第二个PWM波形的占空比。最后设置定时器/计数器运行控制位（CCON.CR）启动PCA定时器/计数器。

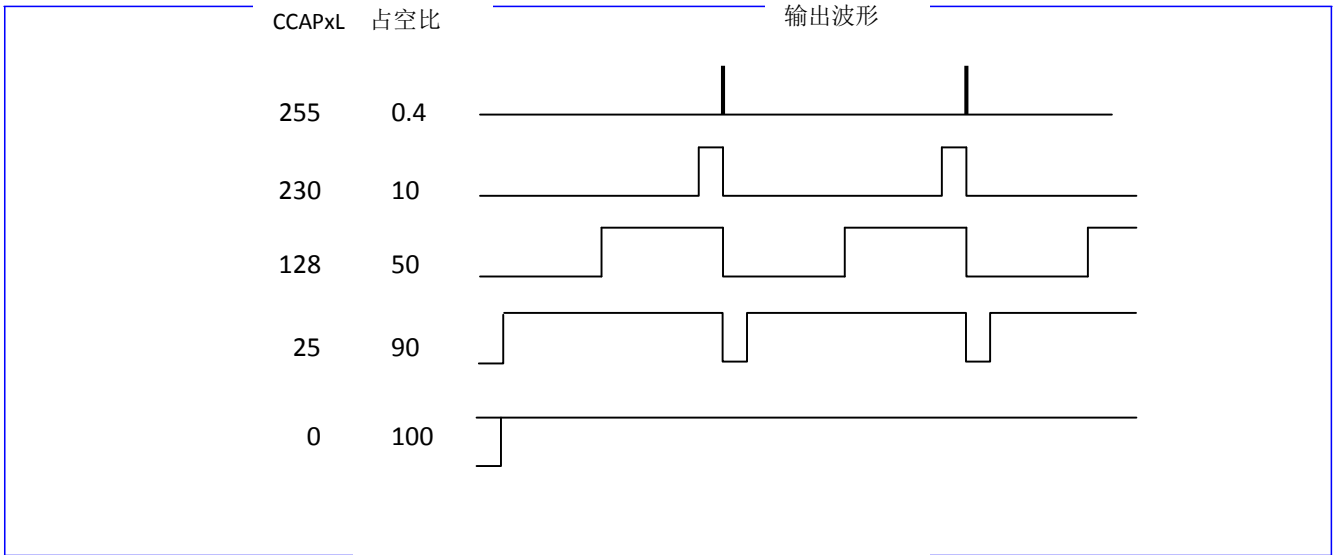


Figure 18-6 PWM 脉冲调制波形

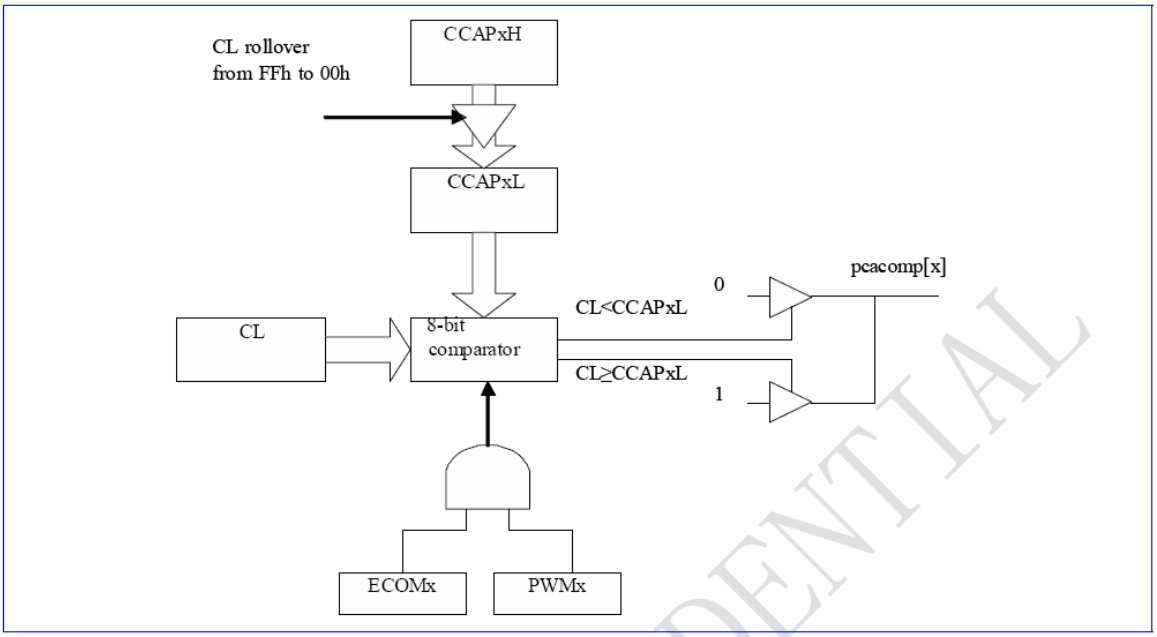


Figure 18-7　PCA 8位 PWM 功能模块

ECOM	CAPP	CAPN	MAT	TOG	PWM	ECCF	工作方式
X	1	0	0	0	0	X	用CCPn的正沿触发捕捉
X	0	1	0	0	0	X	用CCPn的负沿触发捕捉
X	1	1	0	0	0	X	用CCPn的跳变触发捕捉
1	0	0	1	0	0	X	软件定时器
1	0	0	1	1	0	X	高速输出
1	0	0	X	1	1	X	频率输出
1	0	0	X	0	1	X	8位脉冲宽度调制器

Figure 18-8　PCA 比较/捕获功能模块设置

18.10 PCA 寄存器

寄存器缩写	寄存器地址	描述	复位值
CCON	S:0D8h	PCA 定时/计数器控制	00h
CMOD	S:0D9h	PCA 定时/计数器模式	00h
CH	S:0F9h	PCA 定时/计数器高8位	00h
CL	S:0E9h	PCA 定时/计数器低8位	00h
CCAPM0	S:0DAh	PCA 模块 0 比较/捕获模式控制	00h
CCAPM1	S:0DBh	PCA 模块 1 比较/捕获模式控制	00h
CCAPM2	S:0DCh	PCA 模块 2 比较/捕获模式控制	00h
CCAPM3	S:0DDh	PCA 模块 3 比较/捕获模式控制	00h
CCAPM4	S:0DEh	PCA 模块 4 比较/捕获模式控制	00h
CCAP0H	S:0FAh	PCA 模块 0 比较/捕获模式高8位	00h
CCAP0L	S:0EAh	PCA 模块 0 比较/捕获模式低8位	00h
CCAP1H	S:0FBh	PCA 模块 1 比较/捕获模式高8位	00h
CCAP1L	S:0EBh	PCA 模块 1 比较/捕获模式低8位	00h
CCAP2H	S:0FCh	PCA 模块 2 比较/捕获模式高8位	00h
CCAP2L	S:0ECh	PCA 模块 2 比较/捕获模式低8位	00h
CCAP3H	S:0FDh	PCA 模块 3 比较/捕获模式高8位	00h
CCAP3L	S:0EDh	PCA 模块 3 比较/捕获模式低8位	00h
CCAP4H	S:0FEh	PCA 模块 4 比较/捕获模式高8位	00h
CCAP4L	S:0EEh	PCA 模块 4 比较/捕获模式低8位	00h
CCAPO	S:0DFh	PCA 定时/计数器输出(PWM 模式和高速输出模式)	00h

Figure 18-9 PCA 寄存器

18.11 PCA 计数/定时器控制寄存器(CCON）

CCON								
PCA 计数/定时器控制寄存器								
地址空间： S:0D8h								
位	7	6	5	4	3	2	1	0
符号	CF	CR	--	CCF4	CCF3	CCF2	CCF1	CCF0
类型	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	0	0	0	0	0	0

位	符号	描述
7	CF	PCA计数器阵列溢出标志位。 1：当PCA计数器溢出时，CF由硬件置位。如果CMOD寄存器的ECF位为“1”，则CF标志可用来产生中断。 0：CF可以通过硬件或软件置位，但只可通过软件清零。
6	CR	PCA计数器阵列运行控制位。 1：启动PCA计数器阵列计数。 0：关闭PCA计数器阵列计数。
5	--	保留位
4	CCF4	PCA计数器阵列模块4 比较/捕获标志位。 1：当出现匹配或捕获时，该位由硬件置位。 0：该位必须通过软件清零。 当CCAPMx.ECCFx置位时，这个标志位就会产生一个PCA中断。
3	CCF3	PCA计数器阵列模块3 比较/捕获标志位。 1：当出现匹配或捕获时，该位由硬件置位。 0：该位必须通过软件清零。 当CCAPMx.ECCFx置位时，这个标志位就会产生一个PCA中断。
2	CCF2	PCA计数器阵列模块2 比较/捕获标志位。 1：当出现匹配或捕获时，该位由硬件置位。 0：该位必须通过软件清零。 当CCAPMx.ECCFx置位时，这个标志位就会产生一个PCA中断。
1	CCF1	PCA计数器阵列模块1 比较/捕获标志位。 1：当出现匹配或捕获时，该位由硬件置位。 0：该位必须通过软件清零。 当CCAPMx.ECCFx置位时，这个标志位就会产生一个PCA中断。
0	CCF0	PCA计数器阵列模块0 比较/捕获标志位。 1：当出现匹配或捕获时，该位由硬件置位。 0：该位必须通过软件清零。 当CCAPMx.ECCFx置位时，这个标志位就会产生一个PCA中断。

Figure 18-10 CCON寄存器

18.12 PCA 计数/定时器模式寄存器(CMOD)

CMOD								
PCA 计数/定时器模式寄存器								
地址空间： S:0D9h								
位	7	6	5	4	3	2	1	0
符号	CIDL	WDTE	UF2	UF1	UF0	CPS1	CPS0	ECF
类型	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	0	0	0	0	0	0

位	符号	描述
7	CIDL	空闲模式下是否停止PCA计数 1：在空闲模式IDLE下，PCA计数器停止工作。 0：在空闲模式IDLE下，PCA计数器继续工作。
6	WDTE	PCA WDT功能使能信号 1： PCA模块4 启动 WDT功能 0： PCA模块4 禁止 WDT功能
5：3	UF2:0	保留位，用户可自定义。
2:1	CPS1:0	PCA计数器阵列模块时钟输入选择寄存器。
0	ECF	PCA计数器阵列计数/定时器中断使能信号。 1：PCA计数/定时器中断使能信号 0：PCA计数/定时器中断禁用信号

CPS1	CPS0	模式	输入
0	0	0	系统主频的12分频
0	1	1	系统主频4分频
1	0	2	定时器0的溢出
1	1	3	外部ECI输入引脚（最高频率 =系统时钟8分频）

Figure 18-11 CCON寄存器

18.13 PCA 计数/定时器低 8 位寄存器(CL)

CL								
PCA 计数/定时器低8位寄存器								
地址空间： S:0E9h								
位	7	6	5	4	3	2	1	0
符号	CL[7:0]							
类型	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	0	0	0	0	0	0
位	符号	描述						
7:0	CL	PCA 定时/计数器 低8位寄存器						

Figure 18-12 CL寄存器

18.14 PCA 计数/定时器高 8 位寄存器(CH)

CH								
PCA 计数/定时器高8位寄存器								
地址空间： S:0F9h								
位	7	6	5	4	3	2	1	0
符号	CH[7:0]							
类型	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	0	0	0	0	0	0
位	符号	描述						
7:0	CH	PCA 定时/计数器 高8位寄存器						

Figure 18-13 CH寄存器

18.15 PCA 比较/捕获模式控制寄存器(CCAPM0~4)

CCAPM0~CCAPM4

PCA 比较/捕获模式控制寄存器

地址空间：
CCAPM0 (S:DAh)
CCAPM1 (S:DBh)
CCAPM2 (S:DCh)
CCAPM3 (S:DDh)
CCAPM4 (S:DEh)

位	7	6	5	4	3	2	1	0
符号	--	ECOMx	CAPPx	CAPNx	MATx	TOGx	PWMx	ECCFx
类型	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	0	0	0	0	0	0

位	符号	描述
7	--	保留位
6	ECOMx	允许比较器功能控制位 1：允许比较器功能 0：禁止比较器功能 当PCA被用于软件计数器,高速输出,PWM模式和WDT模式时，要置位ECOMx
5	CAPPx	正沿捕获控制位 1：允许上升沿捕获 0：禁止上升沿捕获
4	CAPNx	负沿捕获控制位 1：允许下升沿捕获 0：禁止下升沿捕获
3	MATx	允许匹配控制位 1：当MATx=1时，PCA计数值与模块的比较/捕获寄存器的值一旦匹配。这将置位CCON寄存器的中断标志位CCF0。 0：禁止匹配功能
2	TOGx	翻转控制位 1：当TOGx=1时，工作在PCA高速输出模式，PCA计数器的值与模块的比较/捕获寄存器的值一旦匹配就会使CCPx引脚翻转。 0：禁止翻转功能
1	PWMx	脉宽调制控制位 1：当PWMx=1时，允许CCPx引脚当作PWM输出。 0：禁止PWM脉宽调制功能
0	ECCFx	PCA使能中断。 1：使能寄存器CCON中的比较/捕获标志位CCFx，用来产生中断。 0：禁止产生PCA比较/捕获功能中断

Figure 18-14 PCA比较/捕获模式控制寄存器CCAPMx

18.16 PCA 比较/捕获模式高 8 位寄存(CCAP0H~CCAP4H)

CCAP0H~CCAP4H								
PCA 比较/捕获模式高8位寄存器								
地址空间： CCAP0H (S:FAh) CCAP1H (S:FBh) CCAP2H (S:FCh) CCAP3H (S:FDh) CCAP4H (S:FEh)								
位	7	6	5	4	3	2	1	0
符号	CCAP0H~CCAP4H							
类型	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	0	0	0	0	0	0
位	符号	描述						
7：0	CCAPxH	比较/捕获模式高8位寄存器 当PCA模块用于比较/捕获模式时，他们用于保存各个模块16位捕捉计数值；当PCA用于PWM模式时，他们用来控制输出占空比。						

Figure 18-15 PCA比较/捕获模式高8位寄存器CCAPxH

18.17 PCA 比较/捕获模式低 8 位寄存器(CCAP0L~CCAP4L)

CCAP0L~CCAP4L								
PCA 比较/捕获模式低8位寄存器								
地址空间： CCAP0L(S:EAh) CCAP1L(S:EBh) CCAP2L(S:ECh) CCAP3L(S:EDh) CCAP4L(S:EEh)								
位	7	6	5	4	3	2	1	0
符号	CCAP0L~CCAP4L							
类型	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	0	0	0	0	0	0
位	符号	描述						
7：0	CCAPxL	比较/捕获模式低8位寄存器 当PCA模块用于比较/捕获模式时，他们用于保存各个模块16位捕捉计数值；当PCA用于PWM模式时，他们用来控制输出占空比。						

Figure 18-16 PCA比较/捕获模式低8位寄存器CCAPxL

18.18 PCA 计数/定时器 PWM 模式和高速输出模式控制寄存器(CCAPO)

CCAPO								
PCA 定时/计数器PWM模式和高速输出模式控制寄存器								
地址空间： S:DFh								
位	7	6	5	4	3	2	1	0
符号	--	--	--	CCAPO4	CCAPO3	CCAPO2	CCAPO1	CCAPO0
类型	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	0	0	0	0	0	0
注意：CCAPO可以由硬件，软件设置，万一硬件软件同时设置这个寄存器，只有软件设置才会被实现。								

位	符号	描述
7：5	--	保留位
4	CCAPO4	比较/捕获模块4 输出值
3	CCAPO3	比较/捕获模块3 输出值
2	CCAPO2	比较/捕获模块2 输出值
1	CCAPO1	比较/捕获模块1 输出值
0	CCAPO0	比较/捕获模块0 输出值

Figure 18-17 PCA 定时/计数器在PWM模式和高速输出模式控制寄存器CCAPO

19 看门狗（WDT）

看门狗模块以内部32K 时钟或外部X32K时钟为时钟信号进行计时，在计时溢出时自动产生复位信号，对整个系统进行复位。为了防止系统在异常情况下，程序失效，导致系统异常工作，WDT复位可以帮助系统自动恢复。

注意事项：

为了提供一个比标准8xC251更加灵活的WDT模块，HC16Lxx增加了一个控制寄存器（WDTCON），该寄存器可配置WDT计时溢出的时间。

19.1 概要

- WDT模块是一个20比特的计时器。
- 看门狗复位寄存器（WDTRST）使能、复位WDT模块。
- 看门狗控制寄存器（WDTCON）配置计时溢出的时间周期。

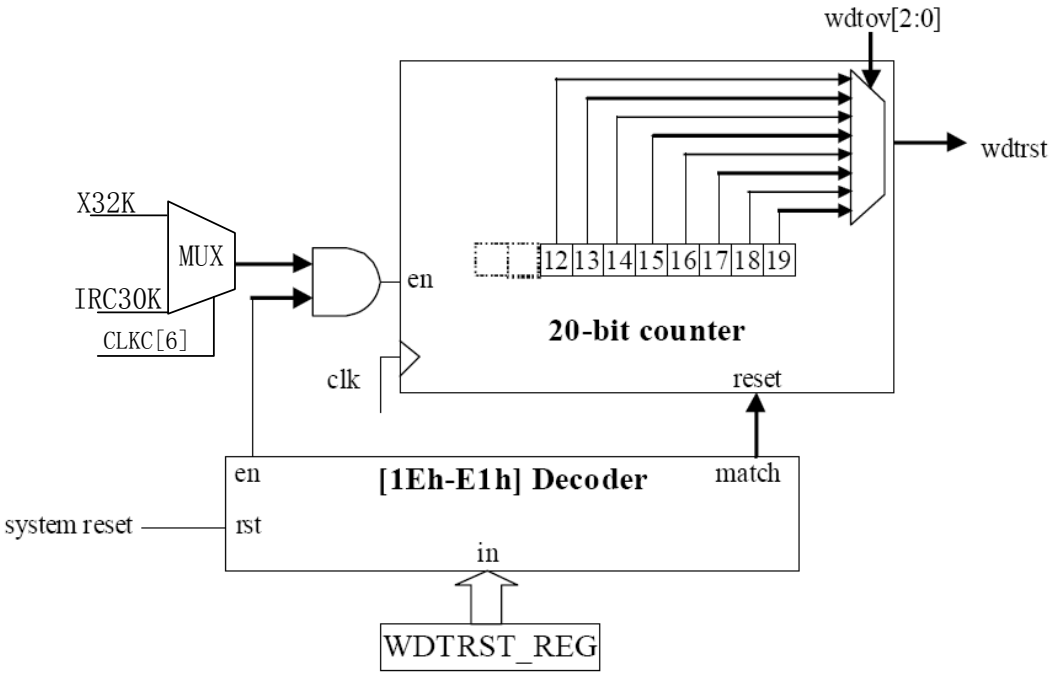


Figure 19-1 看门狗功能模块

注意事项：

- 复位WDT模块，WDT模块会处于停止状态，需要用户重新启动。
- 通过配置WDTCON寄存器，配置计时溢出时间。
- 写0x1E、0xE1序列到WDTRST寄存器，清零并启动WDT计时器。
- 一旦WDT计时器启动计时，WDTCON将无法写入。
- WDTRST寄存器是只写寄存器，软件读取该寄存器将返回0FFh。。
- 可通过WDTCON[5:2]读取WDT计时器当前的计时值。
- WDT计时时钟源可以选择外部32.768KHz时钟输入或者内部RC OSC时钟输出，通过寄存器CLKC[6]选择。

19.2 看门狗应用步骤

使用看门狗的步骤如下：

1. 配置WDTCON寄存器，选择WDT计时溢出时间
2. 写1Eh－E1h序列到WDTRST寄存器，WDT计时器被清零并启动计时，初始计时值为00h
3. 在WDT计时溢出之前，写1Eh － E1h序列到WDTRST寄存器，清零并重启WDT计时器

通过不断重复步骤中的第3项，可以使WDT不产生复位信号。

19.3 看门狗寄存器

19.4 看门狗复位寄存器（WDTRST）

WDTRST								
看门狗复位寄存器								
地址空间： SFR：A6h								
位	7	6	5	4	3	2	1	0
符号	WDTRST[7:0]							
类型	W	W	W	W	W	W	W	W
复位值	0	0	0	0	0	0	0	0
位	符号	描述						
7:0	WDTRST	WDT计时器控制。 写入1Eh - E1h序列将清零并启动WDT计时						

Figure 19-2 看门狗复位寄存器（WDTRST）

19.5 看门狗控制寄存器（WDTCON）

WDTCON								
看门狗控制寄存器								
地址空间： SFR：A5h								
位	7	6	5	4	3	2	1	0
符号	WDT3	WDT2	WDT1	WDT0	WDTR	WOV2	WOV1	WOV0
类型	R	R	R	R	R	R/W	R/W	R/W
复位值	0	0	0	0	0	0	0	0

位	符号	描述
7：4	WDT[3:0]	20位WDT计时值最低4位LSB
3	WDTR	WDT运行标志位 1：WDT计时 0：WDT停止
2:0	WOV[2:0]	WDT计时溢出控制位

看门狗溢出时间选择			看门狗计数器位数	溢出周期	溢出时间@X32K
WOV2	WOV1	WOV0			
0	0	0	13-bit	8 K	250ms
0	0	1	14-bit	16 K	500ms
0	1	0	15-bit	32 K	1s
0	1	1	16-bit	64 K	2s
1	0	0	17-bit	128 K	4s
1	0	1	18-bit	256 K	8s
1	1	0	19-bit	512 K	16s
1	1	1	20-bit	1024 K	32s

Figure 19-3 看门狗控制寄存器（WDTCON）

20 实时时钟（RTC）

实时时钟（RTC）用于记录实时时间并具有万年历的功能。可以通过选择外部32.768KHz的晶振来进行精确计时。具体如下功能：

- 24/12 小时时间模式；支持 BCD 数据的寄存器；
- 32.768KHz 时钟输入；中断周期可配置为年/月/日/小时/分钟/秒；
- 最小可调精度为 1/16 秒；可调整的年/月/日/小时/分钟/秒；
- 具有硬件自动修正闰年的日历功能；
- 支持年/月/日/小时/分钟/秒的软件+1，-1 操作，这样使用内部温度传感器或外部精准温度传感器可以进行实时时钟精确度补偿；
- 支持 2 个可配置频率，宽度，相位的脉冲输出；
- 用于指示时间和日期的日历记录器在 MCU 受外部因素影响而复位时不会清除保留值；
- 支持两个可选择的时钟输入(外部 32.768KHz 晶振时钟和内部 RC32K 时钟)

20.1 功能描述

20.2 计时

RTC是一个万年历计时模块，提供软件读取和设置，计时单位包括年（闰年）、月、日、时、分、秒、1/16秒。通过配置相应寄存器来完成计时设置和计时读取。

所有软件写入和读取的日期时间值都为BCD码，无须十六进制转换为十进制。

任何无效的日期时间将无法写入，比如1月32日，25时59分，23时65分70秒等。有些情况下会发生计时溢出，该溢出的计时单位将在下一秒复位。比如先设置1月31号，再把月份调整为2月，这样日计时单位溢出，在下一秒将复位，变为2月1日。

HOLD寄存器初值为1，RTC模块处于暂停状态；

HOLD寄存器为0，RTC模块开始计时。

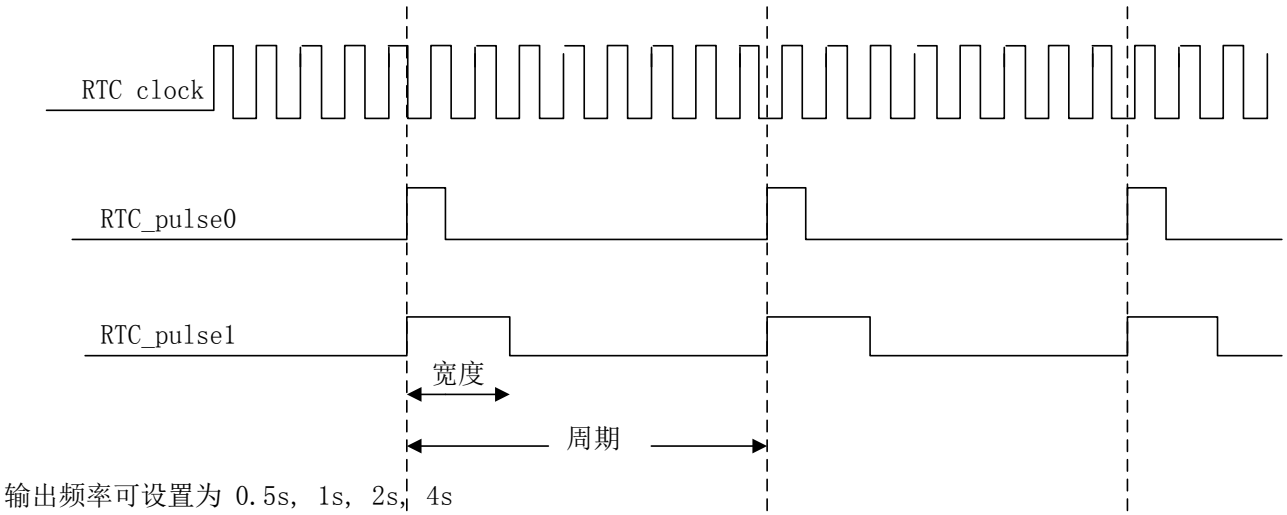
RTC_RESET寄存器为0，RTC复位到原始时期：2013年1月1日00时00分00秒

20.3 中断

在RTC计时过程中，可产生一个RTC中断。对中断产生间隔寄存器操作可配置中断的频率。最快频率为1/16毫秒，最慢频率为一年。

20.4 脉冲输出

在低功耗工作模式下仍然提供两通道脉冲输出，每个脉冲的频率，宽度，以及两个脉冲间的相位差都可通过配置CONF0，CONF1以及PHASE寄存器完成。



输出频率可设置为 0.5s，1s，2s，4s
输出高脉冲宽度为244us(8个X32K周期)，488us(16个X32K周期)，976us(32个X32K周期)，1952us(64个X32K周期)
输出脉冲频率与RTC中断无关
例如：RTC 中断设置在0.5秒，RTC 脉冲可以设置在2秒

Figure 20-1 RTC 脉冲相位关系图

20.5 时钟源

普通计时模式下，RTC模块采用外部32.768KHz晶振输入，其他模式下还可选择使用内部32KHz时钟源。

20.6 软件操作

RTC是一个多时钟域模块，寄存器读写是在系统CPU时钟域工作，而计时器是在RTC时钟域（32.768KHz或者内部RC32KHz）工作。他们属于异步时钟域，为了防止亚稳态出现，硬件电路必须同步。因此软件写入寄存器值要2个RTC时钟（60us左右）才能同步到RTC时钟域。

注意事项：

- HOLD=1时，RTC暂停，年/月/日/时/分/秒等寄存器的值不会装载到RTC计时器中。
- 软件对同一个寄存器操作的间隔不能小于100us（3个32,768时钟周期），否则只有最后一次操作有效。
- RTC_RESET寄存器=0时，RTC模块复位；RTC_RESET寄存器=1时，RTC模块复位撤离。实际上，复位信号在该寄存器操作的60us（2个32，768时钟周期）之后才会真正撤离，因此其他RTC模块寄存器操作必须在RTC_RESET寄存器置1之后60us进行，否则RTC模块仍旧处于RESET状态，寄存器操作无效。

20.7 寄存器定义

寄存器名称	地址	位宽（位）	R/W	默认值
SIXTEENOFSEC	0x00FD00	4	R/W	4’h0
SECOND	0x00FD02	7	R/W	7’h00
MINUTE	0x00FD04	7	R/W	7’h00
HOUR	0x00FD06	6	R/W	6’h00
DAY	0x00FD08	6	R/W	6’h01
MONTH	0x00FD0A	5	R/W	5’h01
YEAR_0	0x00FD0C	8	R/W	8’h13
YEAR_1	0x00FD0E	8	R/W	8’h20
SIXTEENOFSEC_DIS	0x00FD10	4	R	4’h0
SECOND_DIS	0x00FD12	7	R	7’h00
MINUTE_DIS	0x00FD14	7	R	7’h00
HOUR_DIS	0x00FD16	6	R	6’h00
DAY_DIS	0x00FD18	6	R	6’h01
MONTH_DIS	0x00FD1A	5	R	5’h01
YEAR_0_DIS	0x00FD1C	8	R	8’h13
YEAR_1_DIS	0x00FD1E	8	R	8’h20
ADJUST_INC	0x00FD20	7	R/W	7’h00
ADJUST_DEC	0x00FD22	7	R/W	7’h00
FORMAT	0x00FD24	1	R/W	1’h1
INTERVAL	0x00FD26	8	R/W	8’h00
EIGHTOFONE_0_DIS	0x00FD28	8	R	8’h0
EIGHTOFONE_1_DIS	0x00FD2A	3	R	3’h0
CONF0	0x00FD2C	4	R/W	4’h0
CONF1	0x00FD2E	4	R/W	4’h0
PHASE	0x00FD30	6	R/W	6’h00
HOLD	0x00FD32	1	R/W	1’h1
RTC_RESET	0x00FD34	1	R/W	1’h1

Figure 20-2 RTC 模块寄存器

20.8 1/16 秒设置寄存器（SIXTEENOFSEC）

SIXTEENOFSEC[3:0]								
1/16秒设置寄存器（RTC可设置最小精度）								
地址空间： 0x00FD00								
位	7	6	5	4	3	2	1	0
符号	--				SIXTEENOFSEC[3:0]			
类型	--				R/W			
复位值	--				4’h0			

位	符号	描述
[7:4]	--	保留位
[3:0]	SIXTEENOFSEC[3:0]	RTC可设置最小精度寄存器1/16秒 0x0 - 0xf：用户可配置最小时间为1/16秒 - 15/16秒

Figure 20-3 SIXTEENOFSEC 寄存器

20.9 秒设置寄存器（SECOND）

SECOND[6:0]								
秒设置寄存器								
地址空间： 0x00FD02								
位	7	6	5	4	3	2	1	0
符号	--	SECOND[6:0]						
类型	--	R/W						
复位值	--	7’h00						

位	符号	描述
[7]	--	保留位
[6:0]	SECOND[6:0]	RTC 秒设置寄存器 BCD码：0x00 - 0x59，秒 注意事项： HC16Lxx提供防呆措施，超过标准时间的设定，比如设置0x4A，0x65，下一秒就会被恢复成默认值0x00。

Figure 20-4 SECOND 寄存器

20.10 分钟设置寄存器（MINUTE）

MINUTE[6:0]								
分钟设置寄存器								
地址空间： 0x00FD04								
位	7	6	5	4	3	2	1	0
符号	--	MINUTE[6:0]						
类型	--	R/W						
复位值	--	7’h00						

位	符号	描述
[7]	--	保留位
[6:0]	MINUTE[6:0]	RTC 分钟设置寄存器 BCD 码：0x00 - 0x59，分钟 注意事项： HC16Lxx 提供防呆措施，超过标准时间的设定，比如设置 0x4A，0x65，下一秒就会被恢复成默认值 0x00。

Figure 20-5 MINUTE 寄存器

20.11 小时设置寄存器（HOUR）

HOUR[5:0]								
小时设置寄存器								
地址空间： 0x00FD06								
位	7	6	5	4	3	2	1	0
符号	--		HOUR[5:0]					
类型	--		R/W					
复位值	--		7’h00					

位	符号	描述
[7:6]	--	保留位
[5:0]	HOUR[5:0]	RTC 小时设置寄存器 BCD 码：24 小时制：0x00 - 0x23， 12 小时制：0x00 - 0x11，小时 注意事项： HOUR[5] 在 12 小时制表示为 AM/PM，软件不可配置，由硬件自动进位。 HC16Lxx 提供防呆措施，超过标准时间的设定，比如设置 0x3A，0x25，下一秒就会被恢复成默认值 0x00。

Figure 20-6 HOUR 寄存器

20.12 日期设置寄存器（DAY）

DAY[5:0]								
日期设置寄存器								
地址空间： 0x00FD08								
位	7	6	5	4	3	2	1	0
符号	--		DAY[5:0]					
类型	--		R/W					
复位值	--		7’h01					

位	符号	描述
[7:6]	--	保留位
[5:0]	DAY[5:0]	RTC 日期设置寄存器 BCD 码：0x00 - 0x31，日 注意事项： HC16Lxx 提供防呆措施，超过标准时间的设定，比如设置 0x33，0x2A，下一秒就会被恢复成默认值 0x01。

Figure 20-7 DAY 寄存器

20.13 月份设置寄存器（MONTH）

MONTH[4:0]								
月份设置寄存器								
地址空间： 0x00FD0A								
位	7	6	5	4	3	2	1	0
符号	--			MONTH[4:0]				
类型	--			R/W				
复位值	--			5’h01				

位	符号	描述
[7:5]	--	保留位
[4:0]	MONTH[4:0]	RTC 月份设置寄存器 BCD 码：0x01 - 0x12，月 注意事项： HC16Lxx 提供防呆措施，超过标准时间的设定，比如设置 0x13，0x2A，下一秒就会被恢复成默认值 0x01。

Figure 20-8 MONTH 寄存器

20.14 年设置寄存器（YEAR_0）

YEAR_0[7:0]								
年设置寄存器（十位和个位）								
地址空间： 0x00FD0C								
位	7	6	5	4	3	2	1	0
符号	YEAR_0[7:0]							
类型	R/W							
复位值	8’h13							

位	符号	描述
[7:0]	YEAR_0[7:0]	RTC 年设置寄存器（十位和个位） BCD码： 0x00 - 0x99，年的十位与个位 注意事项： HC16Lxx提供防呆措施，超过标准时间的设定，比如设置0x13，0x2A，下一秒就会被恢复成默认值0x00。

Figure 20-9 YEAR_0 寄存器

20.15 年设置寄存器（YEAR_1）

YEAR_1[7:0]								
年设置寄存器（千位和百位）								
地址空间： 0x00FD0E								
位	7	6	5	4	3	2	1	0
符号	YEAR_1[7:0]							
类型	R/W							
复位值	8’h20							

位	符号	描述
[7:0]	YEAR_1[7:0]	RTC 年设置寄存器（千位和百位） BCD 码：0x00 - 0x99，年的千位与百位 注意事项： HC16Lxx 提供防呆措施，超过标准时间的设定，比如设置 0x1A，0xB2，下一秒就会被恢复成默认值 0x00。

Figure 20-10 YEAR_1 寄存器

20.16 1/16 秒时间显示寄存器（SIXTEENOFSEC_DIS）

SIXTEENOFSEC_DIS[3:0]								
1/16 秒时间显示寄存器								
地址空间： 0x00FD10								
位	7	6	5	4	3	2	1	0
符号	--				SIXTEENOFSEC_DIS[3:0]			
类型	--				R			
复位值	--				4’h0			

位	符号	描述
[7:4]	--	保留位
[3:0]	SIXTEENOFSEC_DIS[3:0]	1/16 秒计时值 BCD 码，0x0 - 0xf 1/16 秒计数值，读取用于时间显示。 SIXTEENOFSEC 为设定值， SIXTEENOFSEC_DIS 为计数值，读取此寄存器，即可读取 RTC 的 1/16 秒值。

Figure 20-11 SIXTEENOFSEC_DIS 寄存器

20.17 秒时间显示寄存器（SECOND_DIS）

SECOND_DIS[6:0]								
秒时间显示寄存器								
地址空间： 0x00FD12								
位	7	6	5	4	3	2	1	0
符号	--	SECOND_DIS[6:0]						
类型	--	R						
复位值	--	7’h00						

位	符号	描述
[7]	--	保留位
[6:0]	SECOND_DIS[6:0]	秒 BCD 码，0x00 - 0x59 读取用于时间显示。 SECOND 为设定值， SECOND_DIS 为计数值。

Figure 20-12 SECOND_DIS 寄存器

20.18 分时间显示寄存器（MINUTE_DIS）

MINUTE_DIS[6:0]								
分时间显示寄存器								
地址空间： 0x00FD14								
位	7	6	5	4	3	2	1	0
符号	--	MINUTE_DIS[6:0]						
类型	--	R						
复位值	--	7’h00						

位	符号	描述
[7]	--	保留位
[6:0]	MINUTE_DIS[6:0]	分 BCD 码，0x00 - 0x59，读取用于时间显示。 MINUTE 为设定值， MINUTE_DIS 为计数值。

Figure 20-13 MINUTE_DIS 寄存器

20.19 小时时间显示寄存器（HOUR_DIS）

HOUR_DIS[5:0]								
小时时间显示寄存器								
地址空间： 0x00FD16								
位	7	6	5	4	3	2	1	0
符号	--		HOUR_DIS[5:0]					
类型	--		R					
复位值	--		6’h00					

位	符号	描述
[7:6]	--	保留位
[5:0]	HOUR_DIS[5:0]	小时 BCD 码，24 小时制：0x00 - 0x23，12 小时制：0x00 - 0x11，读取用于时间显示。 HOUR 为设定值， HOUR_DIS 为计数值。

Figure 20-14 HOUR_DIS 寄存器

20.20 日时间显示寄存器（DAY_DIS）

DAY_DIS[5:0]								
日时间显示寄存器								
地址空间： 0x00FD18								
位	7	6	5	4	3	2	1	0
符号	--		DAY_DIS[5:0]					
类型	--		R					
复位值	--		6’h00					

位	符号	描述
[7:6]	--	保留位
[5:0]	DAY_DIS[5:0]	日 BCD 码，0x01 - 0x31，读取用于时间显示。 DAY 为设定值， DAY_DIS 为计数值。

Figure 20-15 DAY_DIS 寄存器

20.21 月时间显示寄存器（MONTH_DIS）

MONTH_DIS[4:0]								
月时间显示寄存器								
地址空间： 0x00FD1A								
位	7	6	5	4	3	2	1	0
符号	--			MONTH_DIS[4:0]				
类型	--			R				
复位值	--			5’h01				

位	符号	描述
[7:5]	--	保留位
[4:0]	MONTH_DIS[4:0]	月 BCD 码，0x01 - 0x12，读取用于时间显示。 MONTH 为设定值， MONTH_DIS 为计数值。

Figure 20-16 MONTH_DIS 寄存器

20.22 年时间显示寄存器（十位和个位）（YEAR_0_DIS）

YEAR_0_DIS[7:0]								
年时间显示寄存器（十位和个位）								
地址空间： 0x00FD1C								
位	7	6	5	4	3	2	1	0
符号	YEAR_0_DIS[7:0]							
类型	R/W							
复位值	8’h13							

位	符号	描述
[7:0]	YEAR_0_DIS[7:0]	年(十位与个位)BCD 码，0x00 - 0x99，读取用于时间显示。 YEAR_0 为设定值， YEAR_0_DIS 为计数值。

Figure 20-17 YEAR0_DIS 寄存器

20.23 年时间显示寄存器（千位和百位）（YEAR_1_DIS）

YEAR_1_DIS[7:0]								
年时间显示寄存器（千位和百位）								
地址空间： 0x00FD1E								
位	7	6	5	4	3	2	1	0
符号	YEAR_1_DIS[7:0]							
类型	R/W							
复位值	8’h20							

位	符号	描述
[7:0]	YEAR_1_DIS[7:0]	年（千位与百位）BCD 码，0x00 - 0x99 读取用于时间显示。 YEAR_1 为设定值， YEAR_1_DIS 为计数值。

Figure 20-18 YEAR1_DIS 寄存器

20.24 时间加 1 寄存器（ADJUST_INC）

ADJUST_INC[6:0]								
时间加 1 寄存器（年月日时分秒 1/16 秒）								
地址空间： 0x00FD20								
位	7	6	5	4	3	2	1	0
符号	--	SIXTEEN_INC	SECOND_INC	YEAR_INC	MONTH_INC	DAY_INC	HOUR_INC	MINTUE_INC
类型	--	W	W	W	W	W	W	W
复位值	--	--	--	--	--	--	--	--

位	符号	描述
[7]	--	保留位
[6]	SIXTEEN_INC	置 1，1/16 秒计时值加 1，该位自动清零
[5]	SECOND_INC	置 1，秒计时值加 1，该位自动清零
[4]	YEAR_INC	置 1，年计时值加 1，该位自动清零
[3]	MONTH_INC	置 1，月计时值加 1，该位自动清零
[2]	DAY_INC	置 1，日计时值加 1，该位自动清零
[1]	HOUR_INC	置 1，小时计时值加 1，该位自动清零
[0]	MINUTE_INC	置 1，分钟计时值加 1，该位自动清零

Figure 20-19 ADJUST_INC 寄存器

20.25 时间减 1 寄存器(ADJUST_DEC)

ADJUST_DEC[6:0]								
时间减 1 寄存器(年月日时分秒 1/16 秒)								
地址空间： 0x00FD22								
位	7	6	5	4	3	2	1	0
符号	--	SIXTEEN_DEC	SECOND_DEC	YEAR_DEC	MONTH_DEC	DAY_DEC	HOUR_DEC	MINTUE_DEC
类型	--	W	W	W	W	W	W	W
复位值	--	--	--	--	--	--	--	--

位	符号	描述
[7]	--	保留位
[6]	SIXTEEN_DEC	置 1，1/16 秒计时值减 1，该位自动清零
[5]	SECOND_DEC	置 1，秒计时值减 1，该位自动清零
[4]	YEAR_DEC	置 1，年计时值减 1，该位自动清零
[3]	MONTH_DEC	置 1，月计时值减 1，该位自动清零
[2]	DAY_DEC	置 1，日计时值减 1，该位自动清零
[1]	HOUR_DEC	置 1，小时计时值减 1，该位自动清零
[0]	MINUTE_DEC	置 1，分钟计时值减 1，该位自动清零

Figure 20-20 ADJUST_DEC 寄存器

20.26 时间制式选择寄存器(FORMAT)

FORMAT[0]								
时间制式选择寄存器(24、12 小时制选择)								
地址空间： 0x00FD24								
位	7	6	5	4	3	2	1	0
符号	--							EN24
类型	--							R/W
复位值	--							1’b1

位	符号	描述
[7:1]	Reserved	
[0]	EN24	1: 24 小时制; 0: 12 小时制; 注意事项: HC16Lxx 不支持 12 小时与 24 小时 2 种时间制式的自动切换，用户需要使用软件来进行时间制式的转换。

Figure 20-21 时间制式选择寄存器

20.27 中断产生间隔设置寄存器（INTERVAL）

20.28 INTERVAL[7:0]								
RTC 中断产生间隔设置寄存器								
地址空间： 0x00FD26								
位	7	6	5	4	3	2	1	0
符号	INTERVAL[7:0]							
类型	R/W							
复位值	8’h00							
位	符号	描述						
[7:0]	INTERVAL[7:0]	8’b00000000: ：无中断 8’b00000001-00001111 ：1/16 - 15/16 秒设置中断设置 8’b0001xxxx ：天中断设置 8’b0010xxxx ：每月中断设置 8’b0011xxxx ：每年中断设置 8’b01000001-01111011 ：1 - 59 秒中断设置 8’b10000001-10111011 ：1 - 59 分中断设置 8’b11000001-11010111 ：1 - 23 小时设置 8’b11100000-11110000 ：1 - 16ms 中断设置						

Figure 20-22 RTC 中断产生间隔设置寄存器

20.29 1/16 秒 11 位计数器低位寄存器（EIGHTOFONE_0_DIS）

EIGHTOFONE_0_DIS[7:0]								
1/16 秒 11 位计数器低位寄存器								
地址空间： 0x00FD28								
位	7	6	5	4	3	2	1	0
符号	EIGHTOFONE_0_DIS[7:0]							
类型	R							
复位值	8’h00							
位	符号		描述					
[7:0]	EIGHTOFONE_0_DIS[7:0]		宽度为 11 位的 1/32768 秒计时器的低 8 位，用于观察使用，11 位计数器溢出即为 1/16 秒。					

Figure 20-23 1/16 秒 11 位计数器低位寄存器

20.30 1/16 秒 11 位计数器低位寄存器（EIGHTOFONE_1_DIS）

EIGHTOFONE_1_DIS[2:0]								
1/16 秒 11 位计数器低位寄存器								
地址空间： 0x00FD2A								
位	7	6	5	4	3	2	1	0
符号	--					EIGHTOFONE_1_DIS[2:0]		
类型	--					R		
复位值	--					3’h0		
位	符号		描述					
7:3	--		保留位					
2:0	EIGHTOFONE_1_DIS[2:0]		宽度为 11 位的 1/32768 秒计时器的高 3 位，用于观察，11 位计数器溢出即为 1/16 秒。					

Figure 20-24 1/16 秒 11 位计数器高位寄存器

20.31 脉冲 0 频率宽度设置寄存器（CONF0）

CONF0[3:0]								
脉冲 0 频率宽度设置寄存器								
地址空间： 0x00FD2C								
位	7	6	5	4	3	2	1	0
符号	--				CONF0[3:0]			
类型	--				R/W			
复位值	--				4’h0			

位	符号	描述
7:4	--	保留位
3:2	CONF0[3:2]	脉冲 0 频率： 00：0.5 秒； 01：1.0 秒； 10：2.0 秒； 11：4.0 秒；
1:0	CONF0[1:0]	脉冲 0 宽度：（单位：RTC 时钟周期） 00：8 cycles = 8 * 30.52uS，约 1/4mS 01：16 cycles = 16 * 30.52uS，约 1/2mS 10：32 cycles = 32 * 30.52uS，约 1mS 11：64 cycles = 64 * 30.52uS，约 2mS

Figure 20-25 RTC PULSE0 频率宽度设置寄存器

20.32 脉冲 1 频率宽度设置寄存器（CONF1）

CONF1[3:0]								
脉冲 1 频率宽度设置寄存器								
地址空间： 0x00FD2E								
位	7	6	5	4	3	2	1	0
符号	--				CONF1[3:0]			
类型	--				R/W			
复位值	--				4’h0			

位	符号	描述
7:4	--	保留位
3:2	CONF1[3:2]	脉冲 1 频率： 00：0.5 秒； 01：1.0 秒； 10：2.0 秒； 11：4.0 秒；
1:0	CONF1[1:0]	脉冲 1 宽度：（单位：RTC 时钟周期） 00：8 cycles = 8 * 30.52uS，约 1/4mS 01：16 cycles = 16 * 30.52uS，约 1/2mS 10：32 cycles = 32 * 30.52uS，约 1mS 11：64 cycles = 64 * 30.52uS，约 2mS

Figure 20-26 RTC PULSE1 频率宽度设置寄存器

20.33 脉冲输出相位关系设置寄存器（PHASE）

PHASE[5:0]								
脉冲输出相位关系设置寄存器								
地址空间： 0x00FD30								
位	7	6	5	4	3	2	1	0
符号	--		PHASE[5:0]					
类型	--		R/W					
复位值	--		6’h00					

位	符号	描述
7:6	--	保留位
5:0	PHASE[5:0]	脉冲 0 和脉冲 1 相位差设置： 000000： 0/16 秒（同向） 000001： 相差 1/16 秒 000010： 相差 2/16 秒 …… 010000： 相差 16/16 秒 …… 111111： 相差 63/16 秒

Figure 20-27 RTC 脉冲输出相位关系设置寄存器

20.34 计数暂停寄存器（HOLD）

HOLD[0]								
计数暂停寄存器								
地址空间： 0x00FD32								
位	7	6	5	4	3	2	1	0
符号	--							HOLD
类型	--							R/W
复位值	--							1’b1

位	符号	描述
7:1	--	保留位
0	HOLD	1：RTC 暂停计时 0：RTC 开始计时

Figure 20-28 RTC 暂停寄存器

20.35 复位寄存器（RTC_RESET）

RTC_RESET[0]								
复位寄存器								
地址空间： 0x00FD34								
位	7	6	5	4	3	2	1	0
符号	--							RTC_RESET
类型	--							R/W
复位值	--							1’b1

位	符号	描述
7:1	--	保留位
0	RTC_RESET	RTC 复位信号，低电平有效 1：RTC 复位无效 0：RTC 复位有效

Figure 20-29 RTC 复位寄存器

21 DES 协处理器

21.1 DES 算法简述

DES(Data Encryption Standard)算法是对称分组密码算法，它的明文、密钥、密文均为64位。不过，64位密钥实际只用到其中的56位，其余8位在算法中被用作校验位。算法共由16轮相似的操作组成，而每一轮均有一组子密钥参与运算，每一轮的子密钥都是由初始的64位（实际上是56位）密钥生成的。

- 执行DES算法标准的加密流程和解密流程，其执行结果完全符合“FIPS PUB 46-3”对算法原理的描述。
- 执行Triple DES标准的加密流程和解密流程，其执行结果完全符合“FIPS PUB 46-3”对算法原理的描述。
- 提供ECB操作模式和CBC操作模式。
- 提供一个安全的硬件实现架构，能够有效抵御侧信道攻击(Side Channel Attack)中的差异功耗分析（DPA）的攻击。

21.2 操作说明

本模块可支持DES和Triple DES的操作。

本模块提供了3个DES密钥寄存器，可以同时存放3组密钥。进行DES运算时，可选择3组密钥中的1组来进行运算。进行Triple DES运算时，则可选择双密钥(密钥1、密钥2，此时密钥1将被使用2次)还是三密钥（密钥1、密钥2、密钥3）来进行运算。

本模块提供ECB和CBC两种操作模式。

注意事项：

- 除非进行复位或者写入操作，否则本模块的KEYx_REG中的值不会改变、本模块DATA_REG中的值在非运算期间也不会改变。这就意味着，如果下次运算还是采用相同的密钥来进行，则无需重新写入密钥；如果下次运算的被操作数据就是本次运算的结果，那么也无需重新写入数据。
- IV寄存器的值在CBC运算过程中会改变，所以，每次进行CBC操作时的第一次运算都需要对IV寄存器进行写入。
- 在运算期间（CNTRL_REG.DES_Start=1），对本模块任何SFR的写操作都将被忽略，对本模块除CNTRL_REG外的任何SFR的读操作均将得到全0。

运行时间说明:

DES: 本模块从启动一次运算（CNTRL_REG.DES_Start写入1）到该次运算结束（CNTRL_REG.DES_Start恢复到0）一共16个PERI_CLK1周期。

Triple DES: 本模块从启动一次运算（CNTRL_REG.DES_Start写入1）到该次运算结束（CNTRL_REG.DES_Start恢复到0）一共48个PERI_CLK1周期。

21.3 ECB 操作模式（DES）

1.将64位随机数写入RAND_REG中（在有抗DPA要求的应用场合下，每次本模块复位后的第一次操作必须写入新的64位随机数，其余场合可以选择省略本步骤）

2.根据应用需求将密钥写入相应的KEYx_REG中，除非复位或者重新写入，否则KEYx_REG中的值将被保持。如果多次运算的密钥相同，则在第一次写入后即可忽略本步骤。

3.将被操作的数据写入DATA_REG中，如果是加密操作，写入64位明文，如果是解密操作，写入64位密文。

4.设置CNTRL_REG中的各位，包括

a. 设置CNTRL_REG.DES_Encrypt，选择加密/解密

b. 设置CNTRL_REG.Key_Sel，选择用哪个密钥寄存器中的密钥进行运算

c. 设置CNTRL_REG.Des_Mode=0

d. 设置CNTRL_REG.DES_OPMode=2’b00

e. 对CNTRL_REG.DES_Start写1，启动模块进行运算

以上的步骤可同时进行。

5. 判断模块运算是否结束。

不断读取CNTRL_REG.DES_Start，如果其值变为0，则表示运算结束。

6. 读取DATA_REG，获得64位运算结果。

如果要继续进行新的运算，回到步骤1。

21.4 CBC 操作模式（DES）

1. 将64位随机数写入RAND_REG中（在有抗DPA要求的应用场合下，每次本模块复位后的第一次操作必须写入新的64位随机数，其余场合可以选择省略本步骤）

2. 根据应用需求将密钥写入相应的KEYx_REG中，除非复位或者重新写入，否则KEYx_REG中的值将被保持。如果多次运算的密钥相同，则在第一次写入后即可忽略本步骤。

3. 如果当前是本数据块中的第一个64位数据运算，则将IV写入IV_REG中，后续数据运算将跳过本步骤

4. 将被操作的数据写入DATA_REG中，如果是加密操作，写入64位明文，如果是解密操作，写入64位密文。

5. 设置CNTRL_REG中的各位，包括

   a. 设置CNTRL_REG.DES_Encrypt，选择加密/解密

   b. 设置CNTRL_REG.Key_Sel，选择用哪个密钥寄存器中的密钥进行运算

   c. 设置CNTRL_REG.Des_Mode=0

   d. 设置CNTRL_REG.DES_OPMode=2’b01

   e. 对CNTRL_REG.DES_Start写1，启动模块进行运算

以上的步骤可同时进行。

6. 判断模块运算是否结束。

不断读取CNTRL_REG.DES_Start，如果其值变为0，则表示运算结束。

读取DATA_REG，获得64位运算结果。

如果本数据块还有数据未运算完，则回到步骤4，重复运算直到所有数据运算完成。在本数据块运算过程中，CNTRL_REG.DES_Encrypt和CNTRL_REG.Key_Sel应保持不变。

21.5 ECB 操作模式（Triple DES）

1. 将64位随机数写入RAND_REG中（在有抗DPA要求的应用场合下，每次本模块复位后的第一次操作必须写入新的64位随机数，其余场合可以选择省略本步骤）

2. 根据应用需求将密钥写入相应的KEYx_REG中，除非复位或者重新写入，否则KEYx_REG中的值将被保持。如果多次运算的密钥相同，则在第一次写入后即可忽略本步骤。

3. 将被操作的数据写入DATA_REG中，如果是加密操作，写入64位明文，如果是解密操作，写入64位密文。

4. 设置CNTRL_REG中的各位，包括

   a. 设置CNTRL_REG. DES_Encrypt，选择加密/解密

   b. 设置CNTRL_REG.Key_Sel，如果选择双密钥的方式，则设置为2’b00（此时KEY1_REG和KEY2_REG中的密钥会被使用，且KEY1_REG中的密钥将被使用2次）；如果选择三密钥的方式，则设置为2’b01（此时三个密钥寄存器中的密钥都将被使用）

   c. 设置CNTRL_REG.Des_Mode=1

   d. 设置CNTRL_REG. DES_OPMode=2’b00

   e. 对CNTRL_REG.DES_Start写1，启动模块进行运算

以上的步骤可同时进行。

5. 判断模块运算是否结束。

不断读取CNTRL_REG.DES_Start，如果其值变为0，则表示运算结束。

6. 读取DATA_REG，获得64位运算结果。

如果要继续进行新的运算，回到步骤1。

21.6 CBC 操作模式（Triple DES）

1. 将64位随机数写入RAND_REG中（在有抗DPA要求的应用场合下，每次本模块复位后的第一次操作必须写入新的64位随机数，其余场合可以选择省略本步骤）

2. 根据应用需求将密钥写入相应的KEYx_REG中，除非复位或者重新写入，否则KEYx_REG中的值将被保持。如果多次运算的密钥相同，则在第一次写入后即可忽略本步骤。

3. 如果当前是本数据块中的第一个64位数据运算，则将IV写入IV_REG中，后续数据运算将跳过本步骤

4. 将被操作的数据写入DATA_REG中，如果是加密操作，写入64位明文，如果是解密操作，写入64位密文。

5. 设置CNTRL_REG中的各位，包括

   a. 设置CNTRL_REG.DES_Encrypt，选择加密/解密

   b. 设置CNTRL_REG.Key_Sel，如果选择双密钥的方式，则设置为2’b00（此时KEY1_REG和KEY2_REG中的密钥会被使用，且KEY1_REG中的密钥将被使用2次）；如果选择三密钥的方式，则设置为2’b01（此时三个密钥寄存器中的密钥都将被使用）

   c. 设置CNTRL_REG.Des_Mode=1

   d. 设置CNTRL_REG.DES_OPMode=2’b01

   e. 对CNTRL_REG.DES_Start写1，启动模块进行运算

   以上的步骤可同时进行。

6. 判断模块运算是否结束。

   不断读取CNTRL_REG.DES_Start，如果其值变为0，则表示运算结束。

7. 读取DATA_REG，获得64位运算结果。

如果本数据块还有数据未运算完，则回到步骤4，重复运算直到所有数据运算完成。在本数据块运算过程中，CNTRL_REG.DES_Encrypt和CNTRL_REG.Key_Sel应保持相同的配置。

21.7 DES 寄存器组

名 称	地址	说 明	复位值
控制寄存器 (CNTRL_REG)	0x00FB00	包括启动、加解密选择、密钥选择	0
RFU		保留今后使用	-
随机数寄存器 (RAND_REG)	0x00FB20~ 0x00FB2E	存放本模块安全运算所需的 64 位随机数	某一固定值
IV 寄存器 (IV_REG)	0x00FB30~ 0x00FB3E	存放 CBC 模式下的 IV 数据	0
数据寄存器 (DATA_REG)	0x00FB40~ 0x00FB4E	存放 DES 算法的 64 位明文/密文	0
密钥寄存器 1 (KEY1_REG)	0x00FB50~ 0x00FB5E	存放 DES 算法的 64 位密钥 1	0
密钥寄存器 2 (KEY2_REG)	0x00FB60~ 0x00FB6E	存放 DES 算法的 64 位密钥 2	0
密钥寄存器 3 (KEY3_REG)	0x00FB70~ 0x00FB7E	存放 DES 算法的 64 位密钥 3	0

Figure 21-1 DES SFR 列表

21.8 DES 控制寄存器（CNTRL_REG）

CNTRL_REG								
DES控制寄存器								
地址空间： 0x00FB00								
位	7	6	5	4	3	2	1	0
符号	--	DES_OPMode		DES_Mode	Key_Sel		DES_Encrypt	DES_Start
类型	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	0	0	0	0	0	0

位	符号	描述
7	--	保留位
6:5	DES_OPMode	DES操作模式： 2’b00: ECB 2’b01: CBC 2’b10: RFU 2’b11: RFU
4	DES_Mode	DES工作模式 0：DES 1：Triple DES
3:2	Key_Sel	密钥选择
1	DES_Encrypt	加密加密设置 0：加密运算 1：解密运算
0	DES_Start	DES启动设置 1：DES启动 0：DES禁用 说明： 本寄存器中的启动（DES_Start）位的操作方法是：软件对本位写入1后，本模块将启动运行，本次运行结束后本模块硬件会自动将本位清0，软件查询到本位为0即表示本次运行完成。

	DES	TripleDES
2’b00	选择密钥1	选择双密钥（密钥1和密钥2）
2’b01	选择密钥2	选择三密钥（密钥1、密钥2、密钥3）
2’b10	选择密钥3	RFU
2’b11	RFU	RFU

Figure 21-2 DES 控制寄存器寄存器

21.9 DES 随机数寄存器（RAND_REG）

RAND_REG								
DES随机数寄存器								
地址空间： RAND_REG0：　0x00FB20 RAND_REG1：　0x00FB22 RAND_REG2：　0x00FB24 RAND_REG3：　0x00FB26 RAND_REG4：　0x00FB28 RAND_REG5：　0x00FB2A RAND_REG6：　0x00FB2C RAND_REG7：　0x00FB2E								
位	7	6	5	4	3	2	1	0
符号	RAND_REGn (n=0~7)							
类型	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	0	0	0	0	0	0

位	符号	描述
7:0	RAND_REGn	随机数寄存器地址0~7 随机数寄存器由8个8位的寄存器组成64位随机数。本模块在复位后的第一次使用时需要写入64位真随机数以达到抵抗差异功耗分析的安全需求。

Figure 21-3 DES 随机数寄存器

21.10 DES IV 寄存器（IV_REG）

IV_REG								
DES IV 寄存器								
地址空间： IV_REG0： 0x00FB30 IV_REG1： 0x00FB32 IV_REG2： 0x00FB34 IV_REG3： 0x00FB36 IV_REG4： 0x00FB38 IV_REG5： 0x00FB3A IV_REG6： 0x00FB3C IV_REG7： 0x00FB3E								
位	7	6	5	4	3	2	1	0
符号	IV_REGn (n=0~7)							
类型	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	0	0	0	0	0	0

位	符号	描述
7:0	IV_REGn	IV寄存器由8个的寄存器组成一个64位的寄存器。在CBC模式时，本寄存器用于存放IV数据，该IV数据在整个CBC过程中的第一次运算时使用。 IV的作用主要是用于产生密文的第一个块，以使最终生成的密文产生差异（明文相同的情况下），使密码攻击变得更为困难。

Figure 21-4 DES IV 寄存器

21.11 DES 数据寄存器（DATA_REG）

DATA_REG								
DES 数据寄存器								
地址空间： DATA_REG0：0x00FB40 DATA_REG1：0x00FB42 DATA_REG2：0x00FB44 DATA_REG3：0x00FB46 DATA_REG4：0x00FB48 DATA_REG5：0x00FB4A DATA_REG6：0x00FB4C DATA_REG7：0x00FB4E								
位	7	6	5	4	3	2	1	0
符号	DATA_REGn （n=0~7）							
类型	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	0	0	0	0	0	0

位	符号	描述
7:0	DATA_REGn	数据寄存器由 8 个 8 位的寄存器组成 64 位数据，用于在模块运算前存放需要被加密的明文或者需要被解密的密文，并且，运算完成后存放加密后的密文或者解密后的明文。

Figure 21-5 DES 数据寄存器

8 个 8 位寄存器连接在一起组成一个 64 位的数据，读写操作时需要分别对每个寄存器进行操作。数据寄存器对应的操作顺序如下：

数据举例：
64'hA0_A1_A2_A3_A4_A5_A6_A7

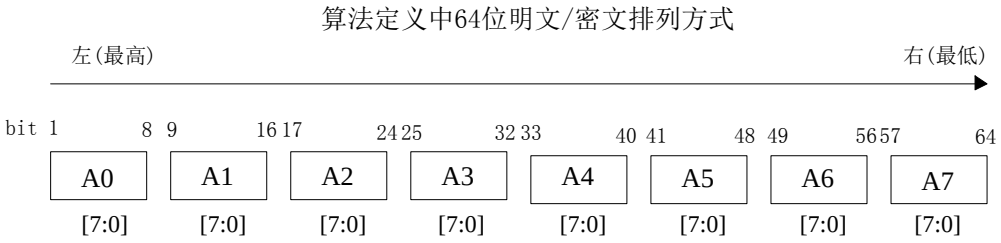


Figure 21-6 DES 数据寄存器操作顺序

21.12 密钥寄存器（KEY1_REG）

KEY1_REG								
DES KEY1寄存器								
地址空间： KEY1_REG0：0x00FB50 KEY1_REG1：0x00FB52 KEY1_REG2：0x00FB54 KEY1_REG3：0x00FB56 KEY1_REG4：0x00FB58 KEY1_REG5：0x00FB5A KEY1_REG6：0x00FB5C KEY1_REG7：0x00FB5E								
位	7	6	5	4	3	2	1	0
符号	KEY1_REGn （n=0~7）							
类型	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	0	0	0	0	0	0

位	符号	描述
7:0	KEY1_REGn	密钥寄存器 1 由 8 个 8 位的寄存器组成 64 位数据，用于存放 DES 运算的 64 位密钥 1。

Figure 21-7 DES KEY1 寄存器

写操作时需要分别对 8 个 8 位寄存器进行操作。对应的操作顺序如下：

密钥举例：
64'hB0_B1_B2_B3_B4_B5_B6_B7

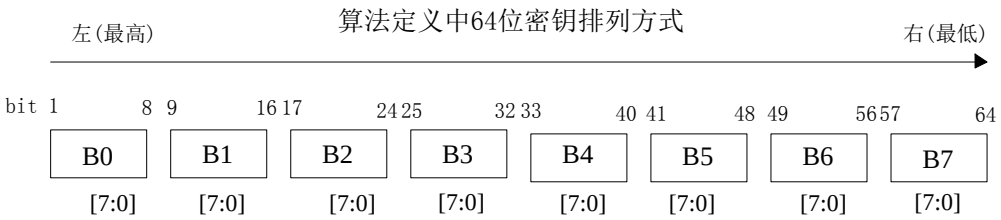


Figure 21-8 DES 密钥寄存器操作顺序

21.13 密钥寄存器（KEY2_REG）

KEY2_REG								
DES KEY2寄存器								
地址空间： KEY2_REG0：0x00FB60 KEY2_REG1：0x00FB62 KEY2_REG2：0x00FB64 KEY2_REG3：0x00FB66 KEY2_REG4：0x00FB68 KEY2_REG5：0x00FB6A KEY2_REG6：0x00FB6C KEY2_REG7：0x00FB6E								
位	7	6	5	4	3	2	1	0
符号	KEY2_REGn （n=0~7）							
类型	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	0	0	0	0	0	0

位	符号	描述
7:0	KEY2_REGn	密钥寄存器 2 由 8 个 8 位的寄存器组成 64 位数据，用于存放 DES 运算的 64 位密钥 2。

Figure 21-9 DES KEY2 寄存器

21.14 密钥寄存器（KEY3_REG）

KEY3_REG								
DES KEY3寄存器								
地址空间： KEY3_REG0：0x00FB70 KEY3_REG1：0x00FB72 KEY3_REG2：0x00FB74 KEY3_REG3：0x00FB76 KEY3_REG4：0x00FB78 KEY3_REG5：0x00FB7A KEY3_REG6：0x00FB7C KEY3_REG7：0x00FB7E								
位	7	6	5	4	3	2	1	0
符号	KEY3_REGn （n=0~7）							
类型	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	0	0	0	0	0	0

位	符号	描述
7:0	KEY3_REGn	密钥寄存器 3 由 8 个 8 位的寄存器组成 64 位数据，用于存放 DES 运算的 64 位密钥 3。

Figure 21-1 DES KEY3 寄存器

22 随机数产生器(RNG)

22.1 简介

HC16Lxx提供一个真随机数发生器，用来产生随机数。

22.2 RNG 操作流程

22.3 生成 64Bits 随机数的操作流程

要生成64Bits随机数，需要经过以下操作：

- 打开随机源电路
- 配置随机数的产生模式
- 生成随机数
- 读取随机数
- 关闭随机数电路

各个操作的软件操作流程如下：

- 打开随机源电路

1. 对随机数控制寄存器的B0（RNGCtrlReg.RNGcir_EN）写入“1”，启动随机源电路，开始输出串行随机数。

- 配置随机数的生成模式

1. 选择是否重新装载初始值：如果将随机数模式寄存器的B0（RNGModeReg. TRNG_Load）设置为“1”，则产生的新随机数的初始值是从随机源获得的，与上一次的随机数无关，产生的新随机数是“真随机数”；如果B0设置为“0”，则初始值是上一次的随机数，产生的新随机数是“伪随机数”。至少在第一次产生随机数时，需要重新装载初始值。
2. 选择64位PRNG(PRNG64)的反馈方式：如果将随机数模式寄存器的B1（RNGModeReg. TRNG_FDBK）设置为“1”，则在进行随机数的数学后处理时，先把PRNG64的反馈信号与随机源的当前值进行“异或”运算后，再输入到PRNG64中；如果将B1设置为“0”，则直接将反馈信号输入PRNG64中。
3. 选择PRNG64的移位次数：通过设置随机数模式寄存器的B4－B2（RNGModeReg. TRNG_cnt），确定移位次数。如果设置为3’b000，则选择不进行数学后处理，直接输出随机源的采样数据；设置为其余值则进行数学后处理，只是移位次数不同。

- 生成随机数

1. 生成随机数：软件将“1”写入随机数控制寄存器的B1（RNGCtrlReg.TRNG_RUN），硬件根据随机数生成配置进行操作，在操作完成后，硬件自动将B1清为“0”。

- 读取随机数

1. 读取随机数：软件在查询到随机数控制寄存器的B1（RNGCtrlReg. TRNG_RUN）变为“0”后，通过读取随机数数据寄存器0(RNGData0Reg)和随机数数据寄存器1(RNGData1Reg)，得到64Bits随机数。

- 关闭随机源电路

1. 对随机数控制寄存器的B0写入“0”（RNGCtrlReg.RNGcir_EN），关闭随机源电路。

22.4 推荐的生成 64Bits 随机数的流程

1. 打开随机源电路

1.1 对随机数控制寄存器的B0（RNGCtrlReg.RNGcir_EN）写入“1”，启动随机源电路，开始输出串行随机数。

2. 每次芯片上电后第一次生成随机数的步骤（请保证随机数模块时钟及随机源此时已打开）。

2.1 选择重新装载初始值：将随机数模式寄存器的B0（RNGModeReg. TRNG_Load）设置为“1”，使新产生的随机数的初始值从随机源获得。

2.2 选择PRNG64的直接反馈的方式：将随机数模式寄存器的B1（RNGModeReg. TRNG_FDBK）设置为“1”，将反馈信号与随机源异或后输入PRNG中。

2.3 选择PRNG64的移位次数：设置随机数模式寄存器的B4－B2（RNGModeReg. TRNG_cnt）为”110”，选择移位256次。

2.4 生成随机数：软件将“1”写入随机数控制寄存器的B1（RNGCtrlReg. TRNG_RUN），硬件根据随机数生成配置进行操作，在操作完成后，硬件自动将B1清为“0”。

2.5 选择不重新装载初始值：将随机数模式寄存器的B0（RNGModeReg. TRNG_Load）设置为“0”。

2.6 选择PRNG64的直接反馈的方式：将随机数模式寄存器的B1（RNGModeReg. TRNG_FDBK）设置为“0”，将反馈信号直接输入PRNG中。

2.7 选择PRNG64的移位次数：设置随机数模式寄存器的B4－B2（RNGModeReg. TRNG_cnt）为”100”，选择移位64次。

2.8 生成随机数：软件将“1”写入随机数控制寄存器的B1（RNGCtrlReg. TRNG_run），硬件根据随机数生成配置进行操作，在操作完成后，硬件自动将B1清为“0”。

3. 读取随机数：软件在查询到随机数控制寄存器的B1（RNGCtrlReg. TRNG_RUN）变为“0”后，通过读取随机数数据寄存器0(RNGData0Reg)和随机数数据寄存器1(RNGData1Reg)，

得到64Bits随机数。

4. 此后生成随机数的步骤（请保证随机源此时已打开）：

4.1 选择不重新装载初始值：将随机数模式寄存器的B0（RNGModeReg. TRNG_Load）设置为“0”。

4.2 选择PRNG64的直接反馈的方式：将随机数模式寄存器的B1（RNGModeReg. TRNG_FDBK）设置为“1”，将反馈信号与随机源异或后输入PRNG中。

4.3 选择PRNG64的移位次数：设置随机数模式寄存器的B4－B2（RNGModeReg. TRNG_cnt）为”110”，选择移位256次。

4.4 生成随机数：软件将“1”写入随机数控制寄存器的B1（RNGCtrlReg.TRNG_RUN），硬件根据随机数生成配置进行操作，在操作完成后，硬件自动将B1清为“0”。

4.5 选择PRNG64的直接反馈的方式：将随机数模式寄存器的B1（RNGModeReg. TRNG_FDBK）设置为“0”，将反馈信号直接输入PRNG中。

4.6 选择PRNG64的移位次数：设置随机数模式寄存器的B4－B2（RNGModeReg. TRNG_cnt）为”100”，选择移位64次。

4.7 生成随机数：软件将“1”写入随机数控制寄存器的B1（RNGCtrlReg. TRNG_RUN），硬件根据随机数生成配置进行操作，在操作完成后，硬件自动将B1清为“0”。

5. 读取随机数：软件在查询到随机数控制寄存器的B1（RNGCtrlReg. TRNG_RUN）变为“0”后，通过读取随机数数据寄存器0(RNGData0Reg)和随机数数据寄存器1(RNGData1Reg)，得到64Bits随机数。

如果需要继续生成新的随机数，那么回到步骤4，直到满足要求。

6. 完成随机数的生成后，推荐选择关闭随机源电路，节省功耗：对随机数控制寄存器的B0（RNGCtrlReg.RNGcir_En）写入“0”，关闭随机源电路。

22.5 随机数寄存器组

名 称	地址	说明	复位值
RNGCtrlReg	0x00FC00	随机数控制寄存器	3'b000
RNGModeReg	0x00FC08	随机数模式寄存器	5'b10000
RNGData0Regn	0x00FC18 0x00FC1A 0x00FC1C 0x00FC1E	随机数数据寄存器 0	—
RNGData1Regn	0x00FC20 0x00FC22 0x00FC24 0x00FC26	随机数数据寄存器 1	—
RFU		—	—

Figure 22-1 RNG模块寄存器列表

22.6 随机数控制寄存器（RNGCtrlReg）

RNGCtrlReg								
随机数控制寄存器								
地址空间： RNGCtrlReg： 0x00FC00								
位	7	6	5	4	3	2	1	0
符号	--	--	--	--	--	PRNG_RUN	TRNG_RUN	RNGcir_EN
类型	--	--	--	--	--	R/W	R/W	R/W
复位值	--	--	--	--	--	0	0	0

位	符号	描述
7:3	--	保留位
2	PRNG_RUN	0：停止随机序列运转 1：启动随机序列运转
1	TRNG_RUN	软件写入“1”后，开始产生新的 64bits 随机数；运行完毕后，硬件自动清除。 0：随机数电路停止运行 1：随机数电路正在运行
0	RNGcir_EN	随机源电路使能位 0：关闭随机源电路（模拟电路） 1：打开随机源电路（模拟电路）

Figure 22-2 RNG 随机数控制寄存器

22.7 随机数模式寄存器（RNGModeReg）

RNGModeReg								
随机数模式寄存器								
地址空间： RNGCtrlReg： 0x00FC08								
位	7	6	5	4	3	2	1	0
符号	--	--	--	TRNG_cnt			TRNG_FDBK	TRNG_Load
类型	--	--	--	R/W	R/W	R/W	R/W	R/W
复位值	--	--	--	1	0	0	0	0

位	符号	描述
7:5	--	保留位
4:2	TRNG_cnt	64bits PRNG 的反馈移位次数 3’b000：移位 0 次（即输出随机源的采样值） 3’b001：移位 8 次 3’b010：移位 16 次 3’b011：移位 32 次 3’b100：移位 64 次 3’b101：移位 128 次 3’b110：移位 256 次 3’b111：RFU（目前为移位 256 次）
1	TRNG_FDBK	在移位操作时，64bits PRNG 的反馈信号是否与随机源进行异或操作 0：不进行异或操作 1：进行异或操作
0	TRNG_Load	在产生新的随机数时，64bits PRNG 是否从随机源获得新的初始值 0：不装载新的初始值 （产生伪随机数） 1：装载新的初始值（产生真随机数）

Figure 22-3 RNG 随机数模式寄存器

22.8 随机数数据寄存器 0（RNGData0Regn）

RNGData0Regn								
随机数数据寄存器 0								
地址空间： RNGData0Reg0： 0x00FC18 RNGData0Reg1： 0x00FC1A RNGData0Reg2： 0x00FC1C RNGData0Reg3： 0x00FC1E								
位	7	6	5	4	3	2	1	0
符号	RNGData0Regn （n=0,1,2,3）							
类型	R	R	R	R	R	R	R	R
复位值	--	--	--	--	--	--	--	--

位	符号	描述
7:0	RNGData0Regn	软件对本寄存器读取将得到低 32 位的随机数

Figure 22-4 RNG 随机数数据寄存器 0

22.9 随机数数据寄存器 1（RNGData1Regn）

RNGData1Regn								
随机数数据寄存器 1								
地址空间： RNGData1Reg0： 0x00FC20 RNGData1Reg1： 0x00FC22 RNGData1Reg2： 0x00FC24 RNGData1Reg3： 0x00FC26								
位	7	6	5	4	3	2	1	0
符号	RNGData1Regn （n=0,1,2,3）							
类型	R	R	R	R	R	R	R	R
复位值	--	--	--	--	--	--	--	--

位	符号	描述
7:0	RNGData1Regn	软件对本寄存器读取将得到高 32 位的随机数

Figure 22-5 RNG 随机数数据寄存器 1

23 循环冗余码校验（CRC）

23.1 CRC 概述

CRC(循环冗余码校验)是一种错误检测方式。本模块采用的CRC算法遵从ISO/IEC13239的定义，采用16位长度的CRC，计算多项式为$x^{16}+x^{12}+x^{5}+1$，16位初值为“FFFF”。

关于该算法的进一步细节请参见ISO/IEC13239。

以下示意了CRC算法的数据传输模型。

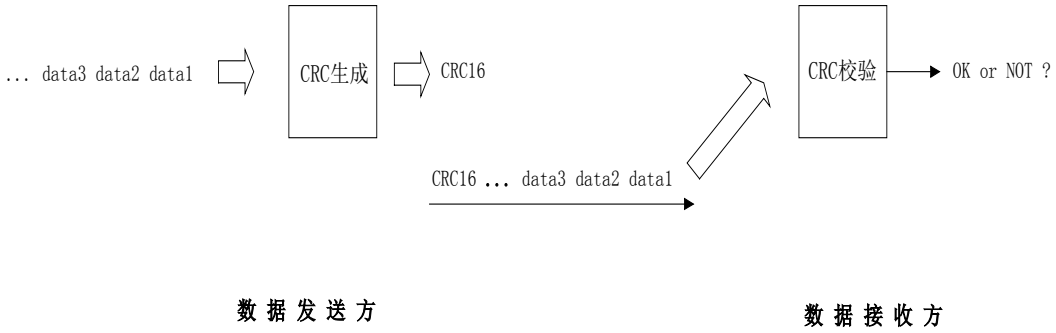


Figure 23-1 CRC 数据传输模型

23.2 CRC 操作

23.3 CRC 编码

CRC编码就是对一串数据进行运算，产生16位的CRC编码结果。

操作流程如下：

将CRC结果寄存器0[15:0]初始化为16’hFFFF。

此操作以通过直接写入0xFFFF到CRC结果寄存器0来实现，也可以通过复位来实现。

将需要运算的数据依次写入CRC数据寄存器，每次写操作对应输入1个数据（8位，16位或32位）。

如果选择32位的数据输入方式，可采用STM指令来加快数据输入，方法是选择本寄存器的一个地址（比如8’h80），然后通过STM指令一次传送若干个数据至CRC数据寄存器（注意STM指令中地址的变化，不要超出CRC数据寄存器规定的地址范围）

在将所有需要运算的数据写入CRC数据寄存器后，读取CRC结果寄存器0[15:0]即可获得16位CRC编码（如果编码以字节表示，应该低8位在先，高8位在后）

23.4 CRC 校验

CRC校验是对一串数据以及16位CRC编码进行判断，检验其是否正确。

CRC校验的操作与上述CRC编码的操作极为类似。区别在于将所有需要运算的数据写入CRC数据寄存器后，需要写入16位的CRC编码（注意如果编码拆成字节应该低8位在先，高8位在后）到CRC数据寄存器。最后，读取CRC结果寄存器1[0]，如果为1则表示校验成功，为0则表示校验失败。

23.5 CRC 寄存器

23.6 CRC 结果寄存器 0（CrcResultReg0）

CrcResultReg0		
CRC 结果寄存器 0		
地址空间： 0x00FA04		
位	15:0	
符号	CrcResultReg0	
类型	R/W	
复位值	16’hFFFF	
位	符号	描述
15:0	CrcResultReg0	每次 CRC 计算结果的更新和保存运算后，读取本寄存器将得到 16 位的 CRC 编码结果。 根据标准规定，运算完成后，16 位的 CRC 编码值是运算寄存器取反后的结果，因此，对本寄存器[15:0]的读取将得到本寄存器当前[15:0]的取反值。

Figure 23-2 CRC 结果寄存器 0

23.7 CRC 结果寄存器 1（CrcResultReg1）

CrcResultReg1		
CRC 结果寄存器 1		
地址空间： 0x00FA06		
位	15:1	0
符号	--	CrcFlag
类型	--	R
复位值	--	0
位	符号	描述
0	CrcFlag	该位为只读，对其写操作时不会对该位产生影响。 进行 CRC 校验运算时，在所有数据和 16 位 CRC 编码输入 CRC 数据寄存器后读取本位，如果为 1 则表明校验成功。 0：当前校验错误 1：当前校验正确

Figure 23-3 CRC 结果寄存器 1

23.8 CRC 数据寄存器（CrcDataReg）

CrcDataReg		
CRC 数据寄存器		
地址空间： 0x00FA80 ~ 0x00FAFE		
位	15:0	
符号	CrcDataReg	
类型	R/W	
复位值	16’h0	
位	符号	描述
15:0	CrcDataReg	本寄存器的地址是一个范围（8’h80-8’hFF），对该范围内的任何一个地址进行操作都会认为是对本寄存器进行操作。支持 8/16/32 位操作

Figure 23-4 CRC 数据寄存器

24 UART

24.1 功能描述

HC16Lxx提供了两个串口通讯接口，支持标准8xC251的各种工作模式，但在管脚功能上有一些细微的区别。HC16Lxx具有独立的串行端口管脚。原始8xC251的模式0（同步模式，半双工）设计中串口的一个管脚被用于双向发送和接收数据。在HC16Lxx的设计中，RXD、TXD独立使用，需要用户设置端口的输入输出特性。每个串口都有独立的自动重装载定时器为可变波特率提供时钟。

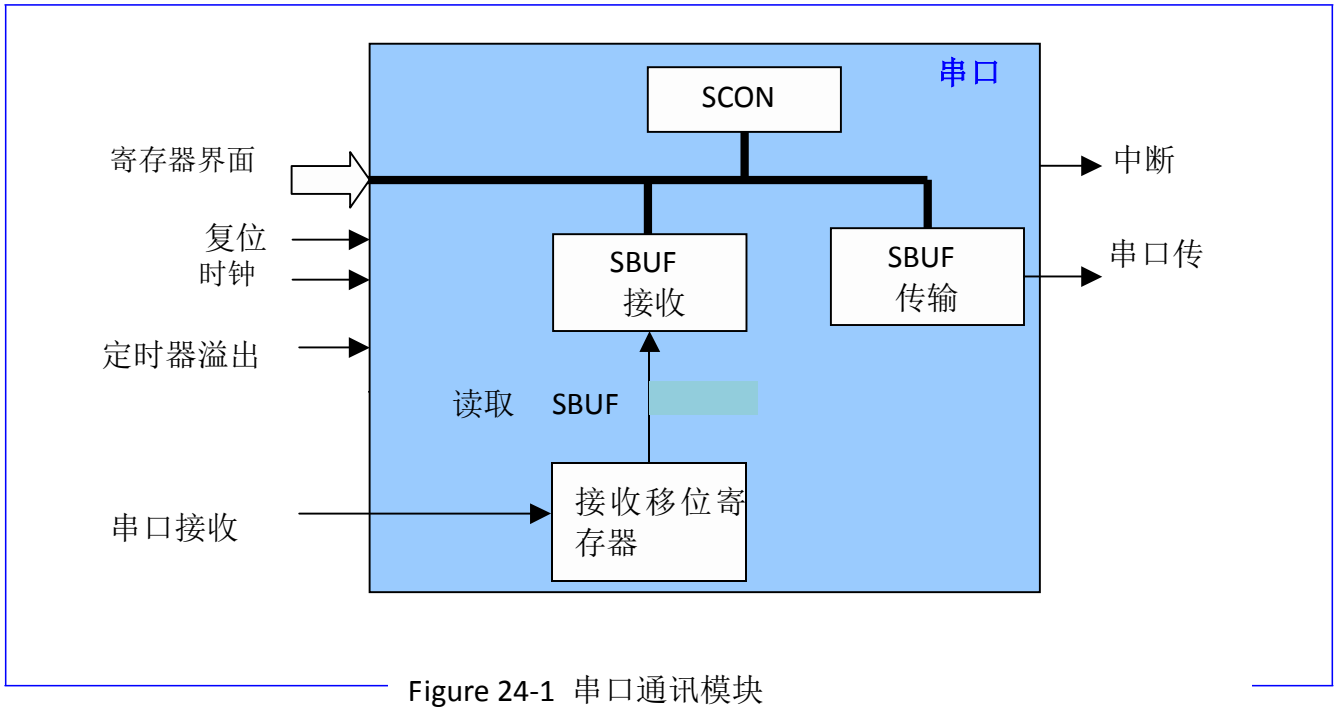


Figure 24-1 串口通讯模块

24.2 工作模式

24.3 模式 0（同步模式，半双工）

当工作在模式0下，串行通讯接口工作在同步移位寄存器模式，其波特率为固定的CPU时钟的1/12。串行口数据由RXD输入或输出（UART0为P50输入输出数据，UART1为P16输入输出数据），同步移位脉冲时钟由TXD输出（UART0从P51输出，UART1从P17输出）。在模式0中，数据的发送/接收是8位的（没有起始位，没有结束位）。数据的发送频率被锁定在微控制器的时钟频率的1/12。若将串口控制寄存器（SCON/SCON1）的SM0与SM1清零，串行通讯接口可工作在模式0下。

24.4 发送数据

发送数据时，清除串口控制寄存器REN位，并将数据写入串口数据缓冲寄存器（SBUF）。数据将被从RXD管脚移出（最低位先出，最高位后出）。串行口数据由RXD输出（UART0为P50输出数据，UART1为P16输出数据），同步移位脉冲时钟由TXD输出（UART0从P51输出，UART1从P17输出）。

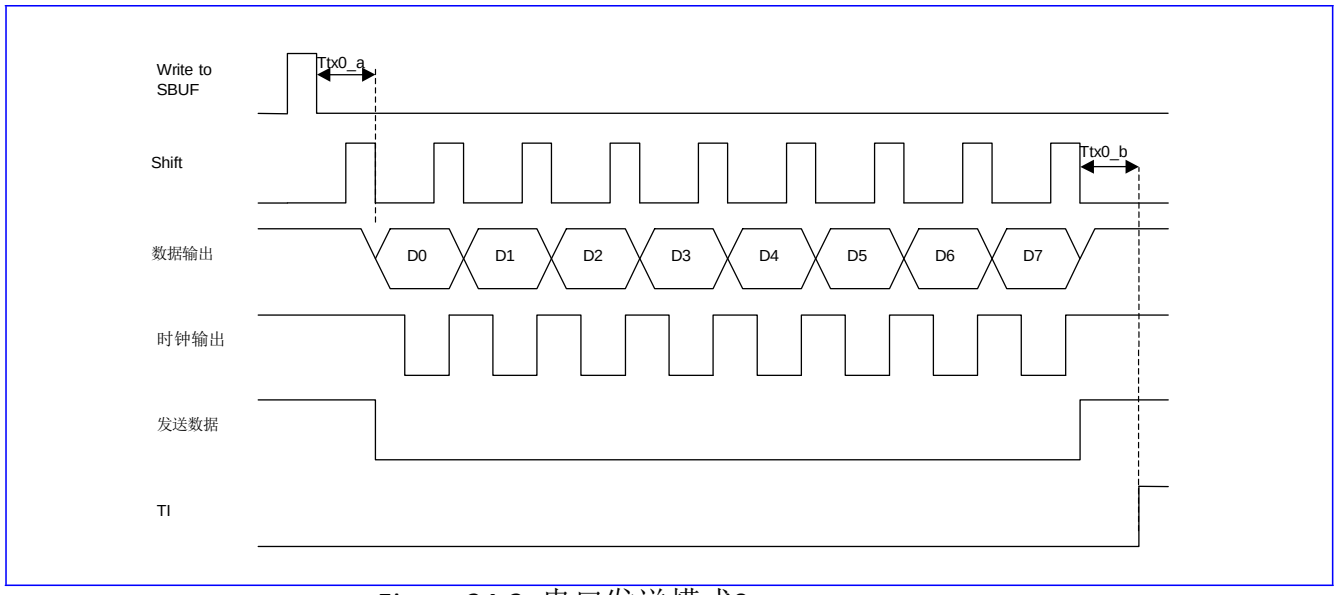


Figure 24-2 串口发送模式0

24.5 接收数据

接收信号时将串口控制寄存器REN位置1，并将RI位清零。开始接收数据，当接收到数据时，数据从串口数据缓冲寄存器读出。串行口数据由RXD输入（UART0为P50输入数据，UART1为P16输入数据），同步移位脉冲时钟由TXD输出（UART0从P51输出，UART1从P17输出）。

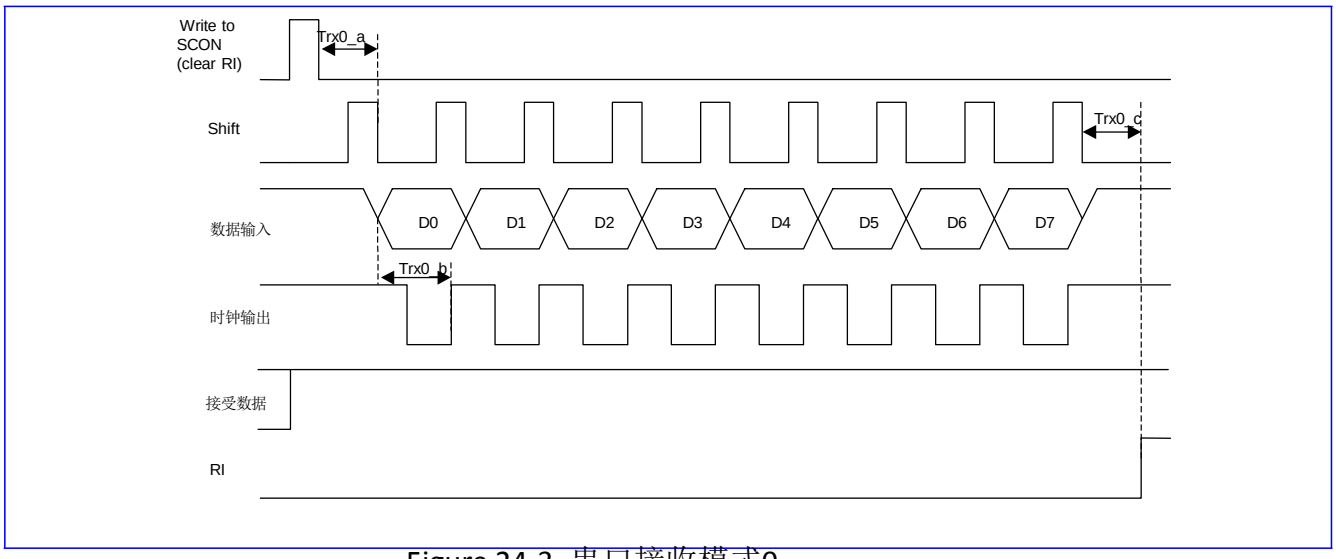


Figure 24-3 串口接收模式0

24.6 模式 1（异步模式，全双工）

在模式1下，数据通过TXD信号发送并且通过RXD信号接收，该数据由10位组成:从起始位“0”开始，紧接着是8个数据位（最低位先，最高位后）。最后是结束位“1”。模式1下的波特率由定时器1和定时器2控制，并且是可编程的。若将串口控制寄存器SM0清零，SM1置“1”，则可工作在模式1下。

24.7 发送数据

发送数据时，将串口控制寄存器REN位清零。并将数据写入串口数据缓冲寄存器中，数据会从TXD管脚移出（最低位先，最高位后）。 UART0为P51输出数据，UART1为P17输出数据。

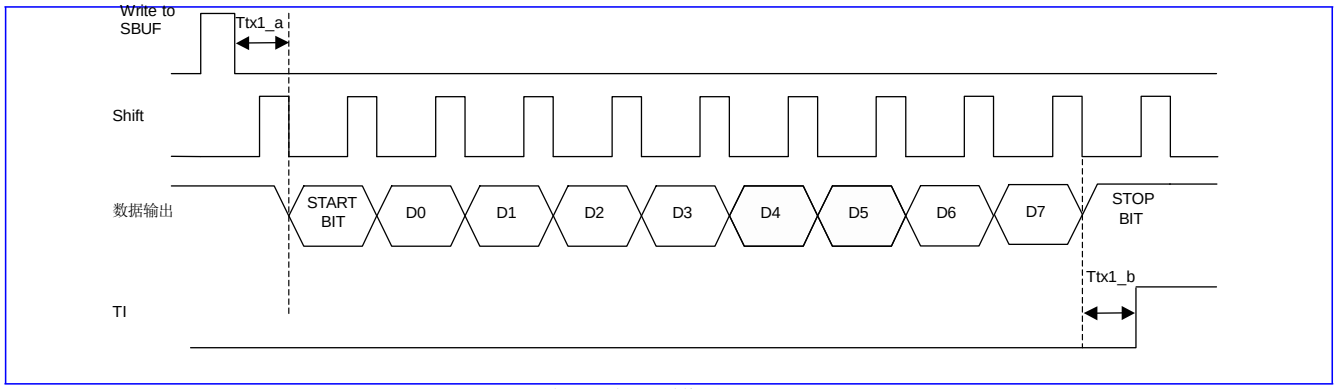


Figure 24-4 串口发送模式1

24.8 接收数据

接收数据时，将串口控制寄存器REN位置1，并且将RI位清零，开始接收数据，当接收到数据值时可以从串口数据缓冲寄存器读出。UART0为P50输入数据，UART1为P16输入数据。

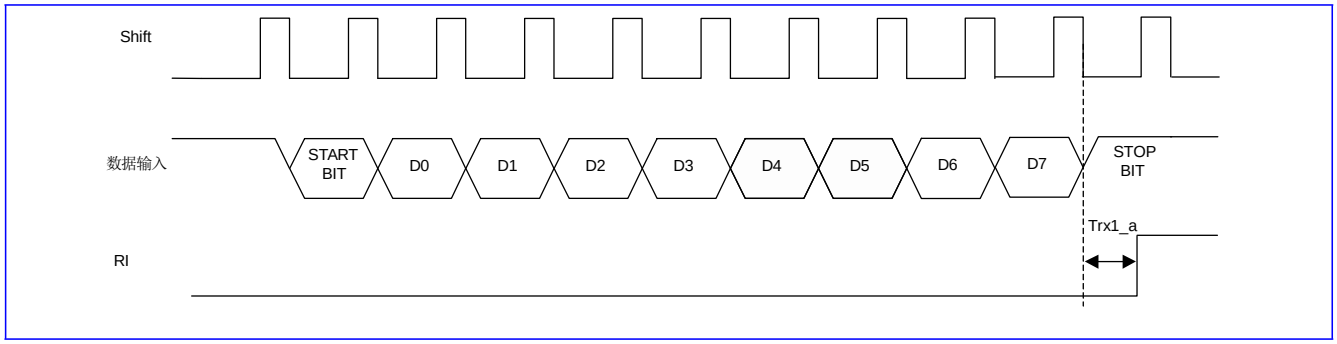


Figure 24-5 串行接收模式1

24.9 模式 2（异步模式，全双工）

当工作在模式2下，数据通过TXD信号发送并通过RXD信号接收，该数据由11位组成，1个起始位，8个数据位，1个TB8位（在串口控制寄存器信号中）和结束位。额外的TB8位是在多进程通信环境下使用的。当不需要用到多进程通信任务时，此位也可作为parity位使用。模式2下可独立产生波特率。并且可被用于其他用途。若将串口控制寄存器SM0置“1”，将SM1清零可选择工作在模式2下。

24.10 发送数据

发送数据时，将串口控制寄存器REN位清零，并且把数据写入串口数据缓冲寄存器中，数据会从TXD管脚移出（最低位先出，最高位后出）。UART0为P51输出数据，UART1为P17输出数据。

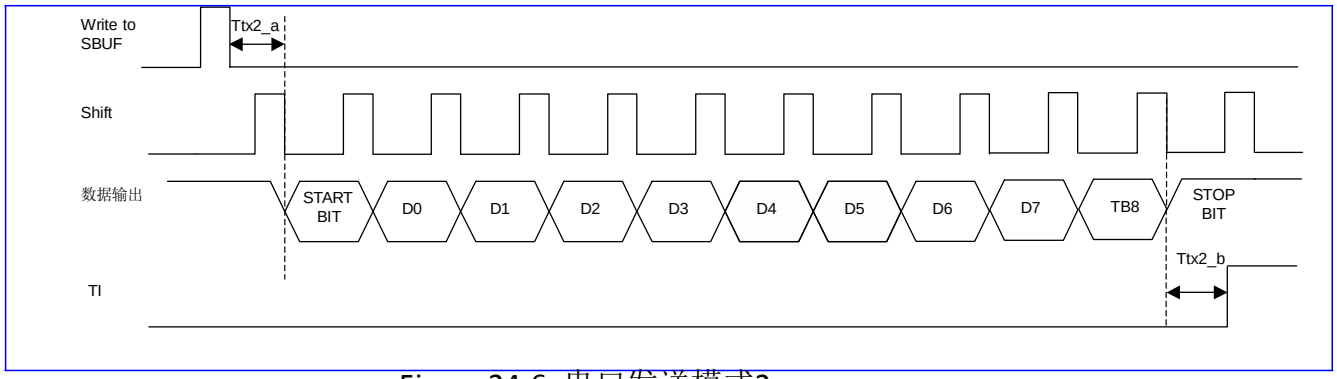


Figure 24-6 串口发送模式2

24.11 接收数据

接收数据时，将串口控制寄存器REN位置“1”，将RI位置0开启接收功能。当接收到数据时，可从串口数据缓冲寄存器读出该值。UART0为P50输入数据，UART1为P16输入数据。

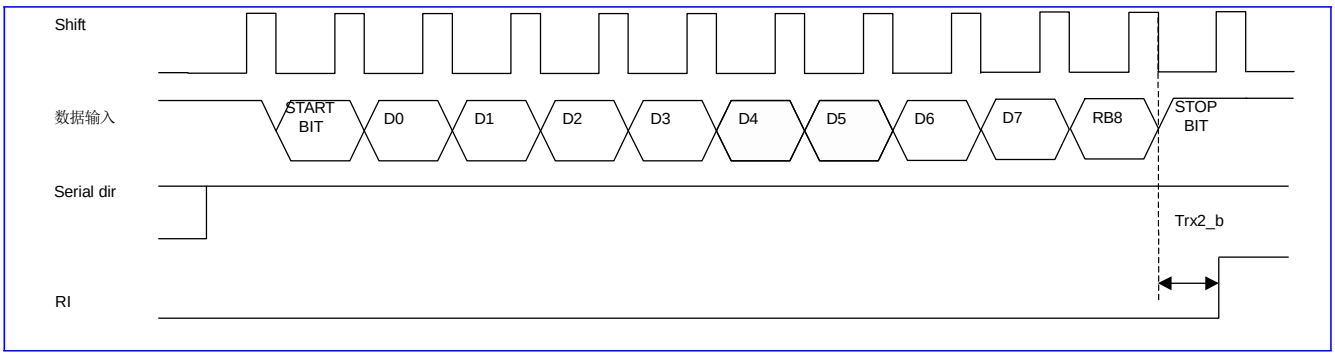


Figure 24-7 串口接收模式2

24.12 模式 3（异步模式，全双工）

模式3的操作与模式2的相同，唯一的区别是UART0定时器，UART1定时器控制波特率。模式3有着和模式2相同的时序图（如上图），但是移位脉冲源是不同的，若将串口控制寄存器的SM0和SM1置“1” 可选择工作在模式3。

24.13 帧同步位错误检测

帧同步位错误检测可在模式1,2,3下使用，将电源控制寄存器（PCON）SMOD00位置“1”可启动此功能。此功能启动后，接收端会检测每个到来的数据帧中的有效停止位，若未发现有效停止位，则将SCON.FE置1.

软件会在每次接收数据后检测SCON.FE位用来检查是否有数据传输错误。硬件置位后，只有通过软件或者复位键将SCON.FE位清零。如未软件清零，那后续而来的有效停止的数据帧不能将FE位清零。

24.14 多进程通信

模式2和模式3提供了一个串口控制寄存器TB8位用于多进程通信。将串口控制寄存器SCON/SCON1.SM2位置“1”可开启该功能。开启后，串口可将数据帧（SCON/SCON1.TB8清零）和地址帧（SCON/SCON1.TB8置“1”）区分。这使得微控制器可在多从机共享同一串口线的环境下作为主，从机使用。

当多进程通信功能开启后，接收端将忽略第9位（SCON/SCON1.TB8）为“0”的数据帧。若第九位被置“1”，接收端会检测帧与地址是否匹配。如果接收到的地址与从地址匹配，串口会将SCON/SCON1.RB8与SCON/SCON1.RI置“1”，并产生一个中断信号。

指定地址的从机系统将SCON/SCON1.SM2位清零，并准备接收数据。因为其他从机在等待地址帧，故其他从机不会受到这些数据的影响。（SCON/SCON1.TB8=0为数据帧）。

注意事项：

必须将中断使能寄存器（IE）的ES位置1才能保证SCON/SCON1.RI位能产生中断信号。

24.15 自动地址识别

当多机通信功能开启后，自动地址识别功能也将启动（将SCON/SCON1.SM2位置“1”）。该功能在硬件中实现，并可通过允许串口检测每个到来的地址帧，用以增强多进程通信功能（SCON/SCON1.TB8=1为地址帧）。只有当串口识别到它自身的地址时，才允许接收端将SCON/SCON1.RI位置“1”，并产生中断信号。这保证了CPU不会被其它的设备的地址帧所干扰。如果有需要可以在模式1下开启自动地址识别功能，在此模式下，停止位会代替TB8位，当接收端地址匹配到设备地址，并且被有效停止信号终止后，SCON/SCON1.RI位才会被置“1”。

注意事项：

- 多进程通信功能和自动地址识别功能不能在模式0下使用。将SCON/SCON1.SM2位在模式0置“1”没有作用。
- 设备必须能被指定地址和广播地址识别才支持自动地址识别功能。

24.16 指定地址

每一个设备都可以在串口从机独立地址寄存器（SADDR）中被指定一个独立地址。串口从机地址掩膜寄存器（SADEN）是包含了无关位（定义为0）来形成设备指定地址的掩膜字节，这些无关位增加了同时寻址一个或多个从机的地址的灵活性。

注意事项：

寻址一个设备（通过其独立地址）时，SADEN掩膜字节必须是FFh。

24.17 广播地址

广播地址是由SADDR和SADEN寄存器的逻辑或组成的。无关位的使用使得定义广播地址时更加的灵活，然而大多数应用中，广播地址是FFh。

24.18 定址一个从机串口

从机串口通常响应它的指定地址和广播地址。通过以下设置：

SADDR ： 01101001

SADEN ： 11111011

指定定址 ：01101x01

广播地址： 11111x11

主机可以和从机通过4个地址通信（两个指定地址和两个广播地址）

01101001 和01101101（指定地址）

11111011 和11111111（广播地址）

24.19 复位地址

复位时，SADDR 和SADEN寄存器被初始化为00h，另外，指定地址和广播地址为xxxxxxxx（所有无关位）。这保证了串口可以向下兼容不支持自动地址识别的80C51构架。

24.20 波特率编程

24.21 模式 0

当串口工作在模式0时，波特率被锁定在时钟频率的1/12。不需要定时器或计数器启动（只需要SCON/SCON1寄存器）

24.22 模式 1 和模式 3

模式1和模式3的波特率由U0_TM/U1_TM产生。

U0_TM是为UART0产生波特率的自动载入计数器。

U0_TMR是启动U0_TM计数器的启动信号。

U1_TM是为UART1产生波特率的自动载入计数器。

U1_TMR是启动U1_TM计数器的启动信号。

$$\text{波特率} = \frac{(\text{PCON.SMOD0x+1})*\text{clk}}{(256-\text{U0\_TM}) * 32}$$

给定一个波特率，U0_TM的重新载入值是

$$\text{U0\_TM} = 256 - \frac{(\text{PCON.SMOD0x+1})*\text{clk}}{\text{波特率} * 32}$$

假如U0_TM/U1_TM不是整数值那么波特率或系统频率就必须做相应变化。（CLKC1为微调主频的寄存器，CLK为系统时钟）

24.23 模式 2

在串口模式2下波特率被锁定到（PCON.SMODx1+1)*clk/64。

24.24 UART 中断

如要使用UART中断，必须置位中断使能寄存器的串口中断使能信号（IE0.ES），以及全局中断使能（IE0.EA）。

24.25 UART 串口寄存器

寄存器名称	地址	描述	默认值
SCON	S:098h	UART0 串口控制寄存器	00h
SBUF *1	S:099h	UART0 串口数据缓冲寄存器	00h
SADDR	S:0A9h	UART0 串口从机独立地址寄存器	00h
SADEN	S:0B9h	UART0 串口从机地址掩膜寄存器	00h
SCON1	S:0AAh	UART1 串口控制寄存器	00h
SBUF1 *2	S:0ABh	UART1 串口数据缓冲寄存器	00h
SADDR1	S:0BAh	UART1 串口从机独立地址寄存器	00h
SADEN1	S:0BBh	UART1 串口从机地址掩膜寄存器	00h
U0_TM	S:0C9h	UART0 串口计数器自动重装载寄存器	00h
U0_TMR	S:0C8h	UART0 串口计数器启动寄存器	00h
U1_TM	S:0CBh	UART1 串口计数器自动重装载寄存器	00h
U1_TMR	S:0CAh	UART1 串口计数器启动寄存器	00h

注意事项：

*1：写 SBUF 会写入传输寄存器，读取 SBUF 会读取接收缓冲寄存器

*2：写 SBUF1 会写入传输寄存器，读取 SBUF1 会读取接收缓冲寄存器

24.26 串口 0 控制寄存器（SCON）

SCON								
串口 0 控制寄存器								
地址空间： S:098h								
位	7	6	5	4	3	2	1	0
符号	FE SM0	SM1	SM2	REN	TB8	RB8	TI	RI
类型	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	0	0	0	0	0	0

位	符号	描述
7	FE SM0	FE：错误帧标志位： 选择这个功能，必须把 PCON.SMOD00 置“1” FE 由硬件置“1”，表示硬件检测到一个非有效停止位，由软件清零，FE 不能被一个有效帧清零。 SM0：串口模式配置 0 位 选择这个功能，必须把 PCON.SMOD00 清“0” 软件操作 SM1 和 SM0 来控制串口操作模式
6	SM1	SM1：串口模式配置 1 位
5	SM2	串口模式配置 2 位：软件配置多机通讯以及自动地址匹配模式 1：启动多机通讯以及地址自动匹配 0：关闭多机通讯以及地址自动匹配 在模式 2 和模式 3 中： 如果 SM2=1，并且 REN=1，则接收机处于地址帧监测模式，可以使用接收到的第 9 位 RB8 来进行地址筛选。RB8=1 为地址帧，通讯数据可以进入 SBUF，置位 RI，进入中断服务程序中进行地址比较；RB8=0 为数据帧，接收机忽略这些数据帧并保持 RI=0 如果 SM2=0，并且 REN=1，则接收机不使用地址监测模式，无论收到的 RB8 为 0 或 1，都直接接收并且进入 SUBF，置位 RI，RB8 在这种模式下为校验位。 在模式 0 和模式 1 中：是非多机通讯方式，这 2 种模式下，SM2 要设置为“0”。
4	REN	接收使能控制位 1：接收使能 0：接收关闭
3	TB8	8 位传输模式

SM0	SM1	模式	描述	波特率
0	0	0	位移寄存器	CLK/12
0	1	1	8 位串口传输	可变波特率
1	0	2	9 位串口传输	CLK/32,CLK/64
1	1	3	9 位串口传输	可变波特率

		在传输模式 2 和 3 下，TB8 为要发送的第 9 位数据，按需要由软件配置 0 或 1。在传输模式 1 和 0 下，该位无效。
2	RB8	在传输模式 2 和 3 下，由硬件置“1”或清零来表示接收的第 9 位数据，SM2 必须置“1”。 在传输模式 1 下，由由硬件置“1”或清零来表示有效的停止位。 不能用于模式 0。
1	TI	发送中断标志位 1：在发送完一个帧的最后一位数据后，由硬件置“1”。 0：由软件清零
0	RI	接收中断标志位 1：当接收到一个帧的最后一位数据后，由硬件置“1”。 0：由软件清零

Figure 24-8 串口0控制寄存器

24.27 串口 0 从机独立地址寄存器（SADDR）

SADDR								
串口 0 从机独立地址寄存器								
地址空间： S:0A9h								
位	7	6	5	4	3	2	1	0
符号	SADDR							
类型	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	0	0	0	0	0	0

位	符号	描述
7：0	SADDR	串口 0 从机独立地址

Figure 24-9 串口0从机地址寄存器

24.28 串口 0 从机地址掩膜寄存器（SADEN）

SADEN								
串口 0 从机地址掩膜寄存器								
地址空间： S:B9h								
位	7	6	5	4	3	2	1	0
符号	SADEN							
类型	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	0	0	0	0	0	0

位	符号	描述
7：0	SADEN	串口 0 从机地址掩膜寄存器

Figure 24-10 串口0从机地址掩膜寄存器

24.29 串口 0 数据缓冲寄存器(SBUF)

SBUF								
串口 0 数据缓冲寄存器								
地址空间： S:99h								
位	7	6	5	4	3	2	1	0
符号	SBUF							
类型	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	0	0	0	0	0	0

位	符号	描述
7：0	SBUF	串口 0 数据缓冲寄存器

Figure 24-11 串口0数据缓冲寄存器

24.30 串口 0 计数器启动寄存器(U0_TMR)

U0_TMR								
串口 0 计数器启动寄存器								
地址空间： S:C8h								
位	7	6	5	4	3	2	1	0
符号	--	--	--	--	--	--	--	U0_TMR
类型	--	--	--	--	--	--	--	R/W
复位值	--	--	--	--	--	--	--	0

位	符号	描述
7：1	--	保留位
0	U0_TMR	串口 0 计数器启动寄存器 1：启动串口 0 计数器配置可变波特率 0：禁用串口 0 计数器

Figure 24-12 串口0计数器启动寄存器

24.31 串口 0 计数器自动重装载寄存器(U0_TM)

U0_TM								
串口 0 计数器自动重装载寄存器								
地址空间： S:C9h								
位	7	6	5	4	3	2	1	0
符号	U0_TM							
类型	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	0	0	0	0	0	0
位	符号	描述						
7：0	U0_TM	串口 0 计数器自动重装载寄存器						

Figure 24-13 串口0计数器自动重装载寄存器

24.32 串口 1 控制寄存器(SCON1)

SCON1								
串口 1 控制寄存器								
地址空间： S:0AAh								
位	7	6	5	4	3	2	1	0
符号	FE SM0	SM1	SM2	REN	TB8	RB8	TI	RI
类型	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	0	0	0	0	0	0

位	符号	描述
7	FE SM0	FE：错误帧标志位： 选择这个功能，必须把 PCON.SMOD10 置“1” FE 由硬件置“1”表示硬件检测到一个非有效停止位，由软件清零，FE 不能被一个有效帧清零。 SM0：串口模式配置 0 位 选择这个功能，必须把 PCON.SMOD10 清“0” 软件操作 SM1 和 SM0 来控制串口操作模式
6	SM1	SM1：串口模式配置 1 位
5	SM2	串口模式配置 2 位：软件配置多机通讯以及自动地址匹配模式 1：启动多机通讯以及地址自动匹配 0：关闭多机通讯以及地址自动匹配 在模式 2 和模式 3 中： 如果 SM2=1，并且 REN=1，则接收机处于地址帧监测模式，可以使用接收到的第 9 位 RB8 来进行地址筛选。RB8=1 为地址帧，通讯数据可以进入 SBUF，置位 RI，进入中断服务程序中进行地址比较；RB8=0 为数据帧，接收机忽略这些数据帧并保持 RI=0 如果 SM2=0，并且 REN=1，则接收机不使用地址监测模式，无论收到的 RB8 为 0 或 1，都直接接收并且进入 SUBF，置位 RI，RB8 在这种模式下为校验位。 在模式 0 和模式 1 中：是非多机通讯方式，这 2 种模式下，SM2 要设置为“0”。
4	REN	接收使能控制位 1：接收使能 0：接收关闭
3	TB8	8 位传输模式

SM0	SM1	模式	描述	波特率
0	0	0	位移寄存器	CLK/12
0	1	1	8 位串口传输	可变波特率
1	0	2	9 位串口传输	CLK/32,CLK/64
1	1	3	9 位串口传输	可变波特率

		在传输模式 2 和 3 下，TB8 为要发送的第 9 位数据，按需要由软件配置 0 或 1。在传输模式 1 和 0 下，该位无效。
2	RB8	在传输模式 2 和 3 下，由硬件置“1”或清零来表示接收的第 9 位数据，SM2 必须置“1”。 在传输模式 1 下，由由硬件置“1”或清零来表示有效的停止位。 不能用于模式 0。
1	TI	发送中断标志位 1：在发送完一个帧的最后一位数据后，由硬件置“1”。 0：由软件清零
0	RI	接收中断标志位 1：当接收到一个帧的最后一位数据后，有硬件置“1”。 0：由软件清零

Figure 24-14 串口1控制寄存器

24.33 串口 1 从机独立地址寄存器(SADDR1)

SADDR1								
串口 1 从机独立地址寄存器								
地址空间： S:0BAh								
位	7	6	5	4	3	2	1	0
符号	SADDR1							
类型	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	0	0	0	0	0	0
位	符号	描述						
7：0	SADDR1	串口 1 从机独立地址						

Figure 24-15 串口1从机地址寄存器

24.34 串口 1 从机地址掩膜寄存器(SADEN1)

SADEN1								
串口 1 从机地址掩膜寄存器								
地址空间： S:BBh								
位	7	6	5	4	3	2	1	0
符号	SADEN1							
类型	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	0	0	0	0	0	0
位	符号	描述						
7：0	SADEN1	串口 1 从机地址掩膜寄存器						

Figure 24-16 串口1从机地址掩膜寄存器

24.35 串口 1 数据缓冲寄存器(SBUF1)

SBUF1								
串口 1 数据缓冲寄存器								
地址空间： S:ABh								
位	7	6	5	4	3	2	1	0
符号	SBUF1							
类型	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	0	0	0	0	0	0

位	符号	描述
7：0	SBUF1	串口 1 数据缓冲寄存器

Figure 24-17 串口1数据缓冲寄存器

24.36 串口 1 计数器启动寄存器(U1_TMR)

U1_TMR								
串口 1 计数器启动寄存器								
地址空间： S:CAh								
位	7	6	5	4	3	2	1	0
符号	--	--	--	--	--	--	--	U0_TMR
类型	--	--	--	--	--	--	--	R/W
复位值	--	--	--	--	--	--	--	0

位	符号	描述
7：1	--	保留位
0	U1_TMR	串口 1 计数器启动寄存器 1：启动串口 1 计数器配置可变波特率 0：禁用串口 1 计数器

Figure 24-18 串口1计数器启动寄存器

24.37 串口 1 计数器自动重装载寄存器(U1_TM)

U1_TM								
串口 1 计数器自动重装载寄存器								
地址空间： S:CBh								
位	7	6	5	4	3	2	1	0
符号	U1_TM							
类型	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	0	0	0	0	0	0
位	符号	描述						
7：0	U1_TM	串口计数器自动重装载寄存器						

Figure 24-19 串口1计数器自动重装载寄存器

25 SPI

25.1 功能描述

SPI采用主从模式（Master Slave）架构，支持多从模式应用，时钟由主设备控制，在时钟移位脉冲下，数据按位传输。HC16Lxx中的SPI可配置为主模式或者从模式。

在主模式下具有如下功能：

- 波特率可配置，最高波特率为系统时钟的二分频；
- 每一帧的位长度可配置；
- 数据传输格式（先发高位/低位）可配置；
- 时钟相位、极性可配置；
- 具有相互独立的接收 FIFO 和发送 FIFO，均为 8 位宽、2 字节深。

在从模式下具有如下功能：

- 最高波特率为系统时钟的八分频；
- 数据传输格式（先发高位/低位）可配置；
- 时钟相位、极性可配置；
- 具有相互独立的接收 FIFO 和发送 FIFO，均为 8 位宽、2 字节深；
- 状态查询；
- 命令错误报警；
- 溢出报警。

25.2 SPI 架构框图

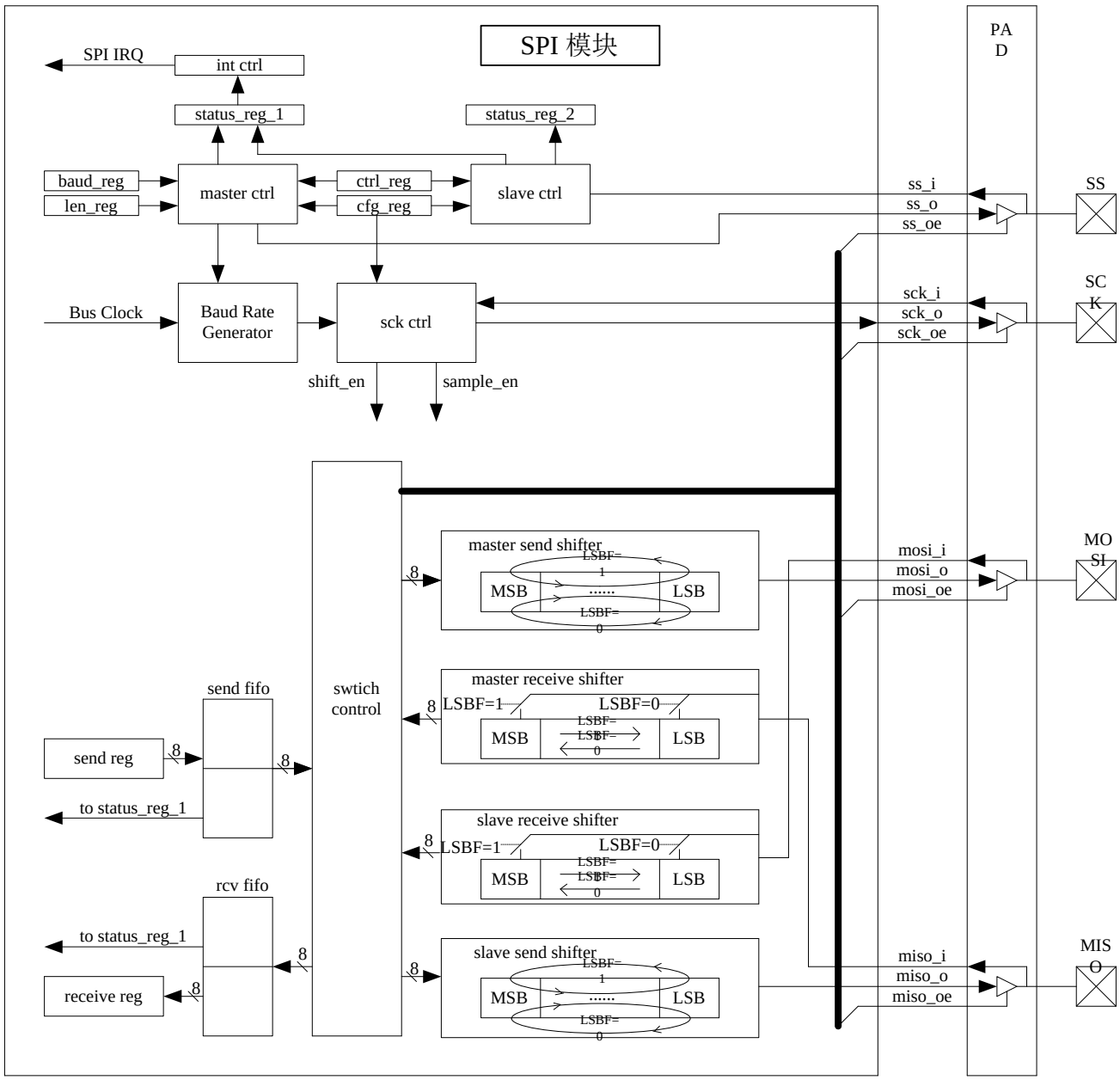


Figure 25-1 SPI 架构框图

25.3 工作模式

SPI 可配置为主模式或从模式，下面分别对其说明。

25.4 主模式

在主模式下，SPI 有三种工作方式：

- 发送方式
- 接收方式
- 全双工方式

发送方式：

SPI 使能后，如果对发送数据寄存器进行写操作，并且发送 FIFO 不满，写入的数据将进入发送 FIFO。设置控制寄存器（SPI_CTRL）的 Opr_mode[1:0]控制位启动发送后，如果发送 FIFO 不空，数据将会由发送 FIFO 装载到发送移位寄存器，在 SCK 同步时钟的控制下，移位寄存器进行左移或右移（由配置寄存器的 LSBF 决定），待发送数据按位顺序出现在串行数据输出线上。发送的一帧（一个数据）的位长度由配置寄存器的 Frame_len[2:0]决定，一帧发送完毕后，如果发送 FIFO 不空，继续按照上述过程发送下一个数据，直到发送 FIFO 变空后，发送停止。后续如果继续向发送数据寄存器写入数据，会再次启动数据的发送。待所有要发送的数据发送完毕后，将控制寄存器的 Opr_mode[1:0]设置为 2’b00。以 CPOL=0、CPHA=00、LSBF=0、每帧由 8 位组成、连续发送两个字节为例，时序图如下：

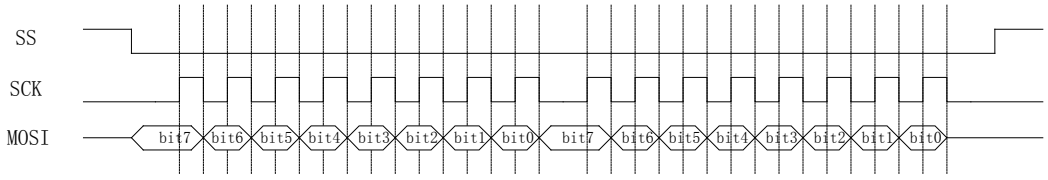


Figure 25-2 SPI 主模式时发送方式时序图

接收方式：

SPI 使能后，根据接收数据长度寄存器（SPI_LEN）及 Frame_len 的设置，发送同步时钟 SCK 信号，在 SCK 的采样沿对串行数据输入线进行采样，采到的数据按位顺序进入接收移位寄存器。当一帧接收完毕后，如果接收 FIFO 不满，接收到的数据由接收移位寄存器载入接收 FIFO，并开始下一帧的接收。当接收 FIFO 的状态指示为不空时，软件可以通过读接收寄存器来获取接收 FIFO 中的数据。当一帧接收完毕后，如果接收 FIFO 已满，SCK 暂停，接收到的数据将暂存在接收移位寄存器中，等待用户读取接收数据寄存器以腾出接收 FIFO 的空间，待接收 FIFO 有空间接收新数据时，继续数据的接收过程。按照上述过程，所有待接收的数据已接收完毕后，接收结束标志置位。以 CPOL=0、CPHA=00、LSBF=0、每帧由 8 位组成、连续接收两个字节为例，时序图如下：

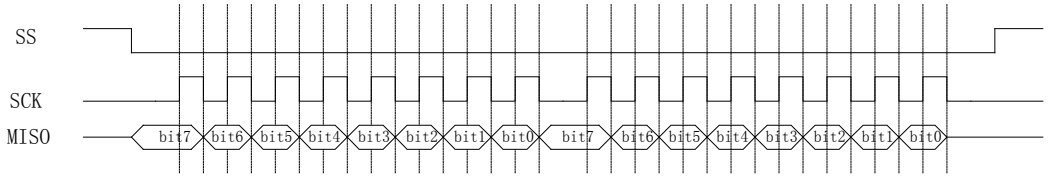


Figure 25-3 SPI 主模式时接收方式时序图

全双工方式：

在全双工方式下，因从设备无法判断主设备已发送结束还是发送的数据为 00 或 FF，所以在全双工方式下，发送的数据长度应大于等于接收的数据长度，即只要存在 SCK，从设备便可采样其输入数据线上的数据。全双工的工作过程与发送及接收过程类似，只是要在一帧的数据发送及接收均完成后，才会开始下一帧的传输。待所有要发送及接收的数据均完成后，SCK 停止，传输过程结束。以 CPOL=0、CPHA=00、LSBF=0、每帧由 8 位组成、连续发送并接收两个字节为例，时序图如下：

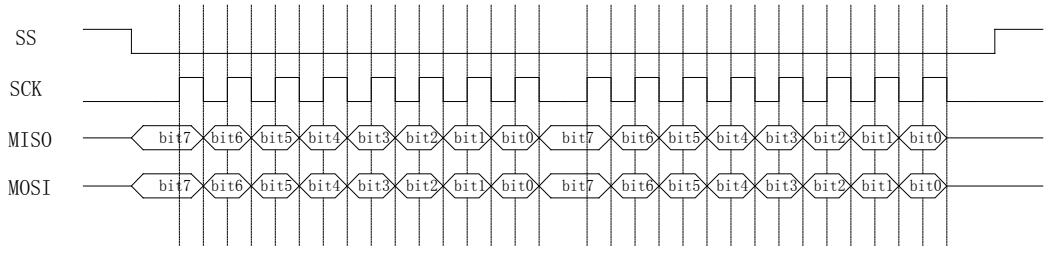


Figure 25-4 SPI 主模式时全双工方式时序图

25.5 从模式

主设备在发送应用命令或收送数据以前必须对从设备的状态进行不断的查询，直至从设备报告状态信息显示其准备好进行接下来的通信任务。

在主设备和本模块进行应用数据传输之前，主设备应首先发送状态查询命令（按照状态查询命令寄存器（SPI_QUERY_CMD）的设置），从设备收到该命令后，硬件将状态寄存器 2（SPI_STATUS2）的数据通过 MISO 引脚循环传出。主设备收到从设备传来的状态数据后进行判断，如果从设备没有准备好（SPI_STATUS2.rx_ready、SPI_STATUS2.tx_ready 均为 0），主设备应持续发送状态查询用的同步时钟；如果从设备已准备好，主设备将片选信号拉高。以 CPOL=0、CPHA=00、LSBF=0、状态查询命令为 55H 为例，状态查询时序图如下：

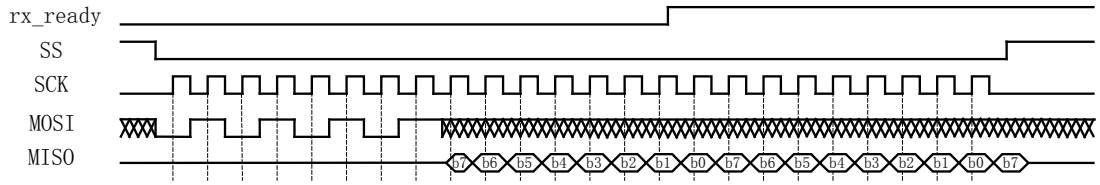


Figure 25-5 SPI 从模式时报告状态时序图

在从模式下，如果从设备已经准备好，那么 SPI 有三种工作方式：

- 接收方式
- 发送方式
- 全双工方式

接收方式：

如果从设备已准备好接收数据，软件将 SPI_CTRL.Opr_mode[1:0]设置为 2’b10，随后硬件将状态寄存器 2（SPI_STATUS2）的 rx_ready 置位。主设备通过状态查询命令获知该信息后，将片选信号拉高，一段时间后再将片选信号拉低，SPI 进行数据接收。以 CPOL=0、CPHA=00, LSBF=0 为例，数据接收时序图如下：

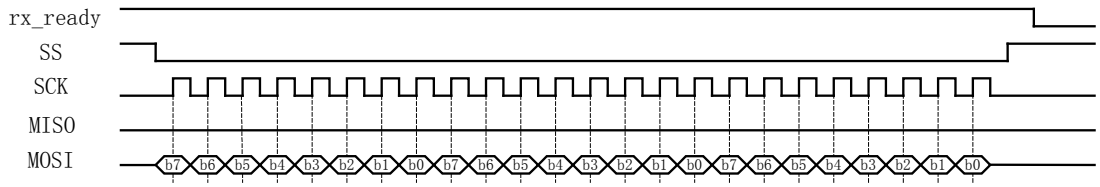


Figure 25-6 SPI 从模式时数据接收状态时序图

发送方式：

如果从设备已准备好发送数据，软件将 SPI_CTRL.Opr_mode[1:0]设置为 2’b01，随后硬件将 SPI_STATUS2. tx_ready 置位。主设备通过状态查询命令获知该信息后，将片选信号拉高，一段时间后再将片选信号拉低，SPI 进行数据发送。以 CPOL=0、CPHA=00、LSBF=0 为例，数据发送时序图如下：

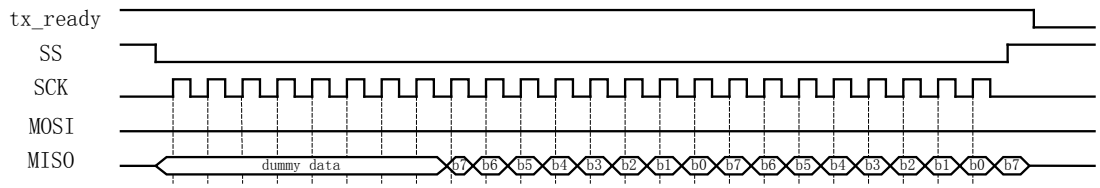


Figure 25-7 SPI 从模式时数据发送状态时序图

在数据发送时，主设备提供的同步时钟 SCK 的个数 = 待接收字节数 x 8 + 8，其中前 8 个为假时钟，从设备从第 9 个 SCK 发送数据，主设备应放弃假时钟采到的数据。

全双工方式：

如果从设备已准备好全双工通讯，软件将 SPI_CTRL.Opr_mode[1:0]设置为 2'b11，随后硬件将 tx_ready、rx_ready 置位。主设备通过状态查询命令获知该信息后，将片选信号拉高，一段时间后再将片选信号拉低，SPI 进行全双工通讯。以 CPOL=0、CPHA=00、LSBF=0 为例，全双工时序图如下：

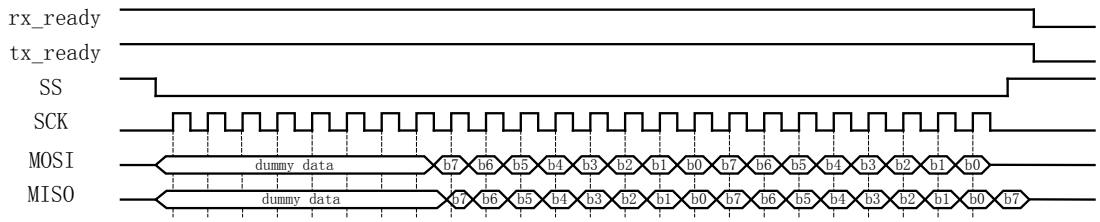


Figure 25-8 SPI 从模式时全双工状态时序图

在全双工通讯时，主设备提供的同步时钟 SCK 的个数 = 待发送或接收的最大字节数 x 8 + 8，其中前 8 个为假时钟，从设备从第 9 个 SCK 发送数据，主设备与从设备应放弃假时钟采到的数据。

25.6 时钟相位与极性

为了兼容不同外围接口芯片，通过设置配置寄存器（SPI_CONFIG）的时钟相位控制位(CPHA[1:0])和时钟极性控制位（CPOL），SCK 的相位和极性有 8 种组合方式：

方式	CPHA[1:0]		CPOL
0	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
6	1	1	0
7	1	1	1

Figure 25-9 SPI 时钟相位和极性的组合方式

时钟相位控制位 CPHA[1:0]用于设定接收和发送的时钟触发沿，时钟极性控制位 CPOL 用于设定 SCK 处于空闲状态时的电平。相互通讯的主、从设备的时钟相位及极性的设置应该相同。

- CPHA[1:0] = 00　在 SCK 的奇数沿接收数据，在偶数沿发送数据
- CPHA[1:0] = 01　在 SCK 的偶数沿接收数据，在奇数沿发送数据
- CPHA[1:0] = 10　在 SCK 的奇数沿接收数据，在奇数沿发送数据
- CPHA[1:0] = 11　在 SCK 的偶数沿接收数据，在偶数沿发送数据
- CPOL = 0　SCK 在空闲状态为低电平
- CPOL = 1　SCK 在空闲状态为高电平

25.7 波特率发生器

在主模式下，波特率发生器用于控制 SCK 的时钟输出及其频率，其频率由系统时钟及波特率寄存器的设置决定。波特率的计算公式为：

$$\text{Baud Rate} = \frac{\text{BusClock}}{2*\text{SPI\_Baud}}$$

上式中，Baud Rate 为 SCK 的频率，BusClock 为系统时钟频率，SPI_Baud 为波特率寄存器的设置值。

25.8 SPI 中断

在 SPI 内部有 2 个与中断相关的状态标志：

- SDR_not_full　　发送 FIFO 不满标志，硬件置位，硬件清 0
- RDR_not_empty　　接收 FIFO 不空标志，硬件置位，硬件清 0

根据 SPI 的工作方式，中断源有所不同：

- 发送方式　　中断源为 SDR_not_full
- 接收方式　　中断源为 RDR_not_empty
- 全双工方式　　中断源为 SDR_not_full||RDR_not_empty

中断请求的撤销：

- 中断源变为 0 后，SPI 的中断请求信号自动撤销
- 向控制寄存器的 Opr_mode[1:0]写入 2'b00 也会使 SPI 的中断请求信号撤销

25.9 主模式软件操作流程

此处提供了中断与查询两种方式的操作流程，但中断方式的数据传输速度要比查询方式慢得多，在传输数据量较大时，将会频繁调用中断服务子程序。

注意事项：

本软件流程是基于硬件片选信号的，如果使用 GPIO 代替 SS 作为片选信号，则应据应用要求，在传输前将 GPIO 置低，在传输后将 GPIO 置高。

由于硬件片选信号一直有效，所以如果所接从设备的数目多于 1，均应使用 GPIO 作为从设备的片选信号。

25.10 初始化设置

1. 将相关 GPIO 配置为 SPI 模式
2. 打开 SPI 时钟使能
3. 根据需要设置中断使能控制位
4. 根据需要设置配置寄存器
5. 根据需要设置波特率寄存器

25.11 发送流程

25.12 查询方式

1. 将待发送数据写入到发送数据寄存器，可连续写入两次
2. 将 Opr_mode[1:0]写入 2'b01
3. 等待状态寄存器的 SDR_not_full 位为 1 时，将下一个数据写入到发送数据寄存器
4. 如果继续发送数据，重复步骤 3，否则跳至下一步
5. 等待状态寄存器的 Send_end 位为 1 时，将 Opr_mode[1:0]写入 2'b00

25.13 中断方式

1. 将待发送数据写入到发送数据寄存器，可连续写入两次
2. 将 Opr_mode[1:0]写入 2'b01
3. 等待中断
4. 中断服务程序：如果仍有数据要发送，将下一个数据写入到发送数据寄存器
5. 退出中断
6. 如果继续发送数据，重复步骤 3-5，否则跳至下一步
7. 等待状态寄存器的 Send_end 位为 1 时，将 Opr_mode[1:0]写入 2'b00

注意事项：

1. 如果待发送的数据长度大于 1，那么在第 1 步时应将发送 FIFO 写满
2. 主设备端软件应在下一次中断到来前，完成写发送数据寄存器操作（使发送 FIFO 处于满状态），否则下一次中断不能被响应

25.14 接收流程

25.15 查询方式

1. 向接收数据长度寄存器写入待接收的数据长度
2. 将 Opr_mode[1:0]写入 2’b10
3. 等待状态寄存器的 RDR_not_empty 位为 1 时，读接收数据寄存器
4. 如果数据未接收完成，重复步骤 3，否则跳至下一步
5. 将 Opr_mode[1:0]写入 2’b00
6. 如果要继续接收，重复步骤 1-5 或者 2-5

25.16 中断方式

1. 向接收数据长度寄存器写入待接收的数据长度
2. 将 Opr_mode[1:0]写入 2’b10
3. 等待中断
4. 中断服务程序：读一次接收数据寄存器
5. 退出中断
6. 如果数据未接收完成，重复步骤 3-5，否则跳至下一步
7. 将 Opr_mode[1:0]写入 2’b00
8. 如果要继续接收，重复步骤 1-6 或者 2-6

注意事项：

主设备端软件应在下一次中断到来前，完成读接收数据寄存器操作（使接收 FIFO 状态为空），否则下一次中断不能被响应。

25.17 全双工流程

25.18 查询方式

1. 将待发送数据写入到发送数据寄存器，可连续写入两次
2. 向接收数据长度寄存器写入待接收的数据长度
3. 将 Opr_mode[1:0]写入 2'b11
4. 等待状态寄存器的 SDR_not_full 位为 1 时，将下一个数据写入到发送数据寄存器
5. 等待状态寄存器的 RDR_not_empty 位为 1 时，读一次接收数据寄存器
6. 如果此次发送或接收过程未完成，重复步骤 4-5，否则跳至下一步
7. 等待状态寄存器的 Send_end 及 Recv_end 位都为 1 时，将 Opr_mode[1:0]写入 2'b00
8. 如果要继续全双工通讯，重复步骤 1-7

25.19 中断方式

1. 将待发送数据写入到发送数据寄存器，可连续写入两次
2. 向接收数据长度寄存器写入待接收的数据长度
3. 将 Opr_mode[1:0]写入 2'b11
4. 等待中断
5. 中断服务程序：
   a) 读状态寄存器判断中断的来源：如果 SDR_not_full 位为 1，将下一个数据写入到发送数据寄存器；如果 RDR_not_empty 位为 1，读一次接收数据寄存器
   b) 退出中断；
6. 如果发送或接收过程未完成，重复步骤 4-5，否则跳至下一步
7. 等待状态寄存器的 Send_end 及 Recv_end 位都为 1 时，将 Opr_mode[1:0]写入 2'b00
8. 如果要继续全双工通讯，重复步骤 1-7

注意事项

1. 在上述流程中，发送的数据长度应大于等于接收的数据长度
2. 如果待发送的数据长度大于 1，那么在第 1 步时应将发送 FIFO 写满
3. 主设备端软件应在下一次中断到来前，保证发送 FIFO 为满、接收 FIFO 为空，否则下一次中断不能被响应

25.20 从模式软件操作流程

25.21 主从 SPI 通讯流程

在从模式下，典型的通信模式如下。主从双方的通讯过程可以存在 4 个时相：状态查询 ——> 命令传输——> 状态查询 ——> 获取结果，其流程如下：

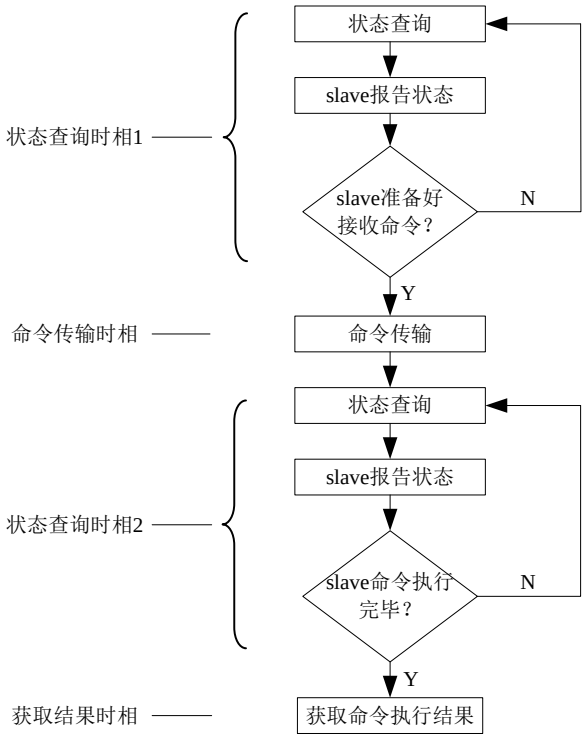


Figure 25-10 SPI 主从 SPI 通讯流程

如果主设备要发送应用命令或发送数据，必须等到从设备反馈的状态显示其已准备好接收（rx_ready 置位）。如果主设备要接收数据，必须等到从设备反馈的状态显示其已准备好发送数据（tx_ready 置位）。

在主设备发送查询命令或应用命令之后必须对从设备的状态进行查询，其需要从从设备反馈的状态信息中了解之前发送的命令有没有出错（query_cmd_err，app_cmd_err）或有没有发生接收溢出（rcv_ov），如果出错，应重传该命令；

建议在主设备发送完数据帧以后对从设备的状态进行查询，可以从从设备反馈的状态信息中了解从设备有没发生接收溢出（rcv_ov），如果发生了接收溢出，应重传该数据。

建议在主设备接收完数据帧以后对从设备的状态进行查询，其可以从从设备反馈的状态信息中了解从设备有没发生发送溢出（send_ov），如果发生了发送溢出，应丢弃该笔数据，并重新接送。

25.22 对主设备 SPI 的要求

25.23 时序要求

- SS 的高电平时间 T_interval 最少为 1/2 个 SCK 周期；
- SS 下降沿到 SCK 的第一个边沿的时间 T_lead 最少为 1/2 个 SCK 周期；
- SCK 的最后一个边沿到 SS 上升沿的时间 T_trail 最少为 1/2 个 SCK 周期；
- 状态查询命令的最后一位，在 SCK 第 16 个边沿后，MOSI 信号的保持时间 T_hold 最少为两个系统时钟周期。

上述参数的说明见下图：

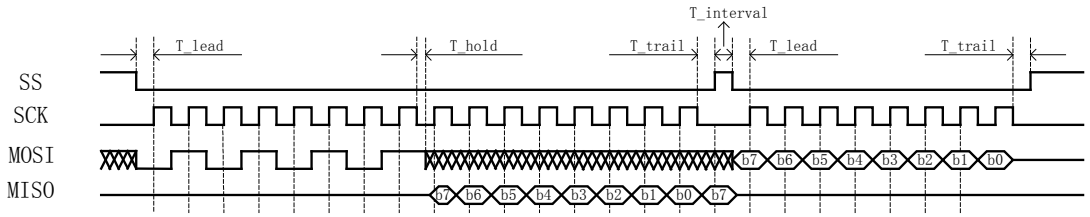


Figure 25-11 对主设备 SPI 的时序要求

25.24 应用要求

- SCK 的最高频率为从设备系统时钟的 8 分频；
- 数据传输以为字节为单位；
- 主设备要等待从设备进行初始化设置完成后，才能发起通讯或状态查询；
- 主设备发送数据的第一个字节不能与状态查询命令寄存器的设置相同；
- 主设备在接收数据时，发出的同步时钟 SCK 的个数 = 待接收字节数 x 8 + 8。其中，前 8 个 SCK 为假时钟，主设备应将接收到的第一个字节丢弃，后面的同步时钟锁存的数据才是有效数据；
- 在全双工通讯时，发出的同步时钟 SCK 的个数 = 待发送或接收的最大字节数 x 8 + 8。其中，前 8 个 SCK 为假时钟，mosi、miso 两数据线上的数据均无意义，主设备软件应将接收到的第一个字节丢弃，在 8 个假时钟后，主设备再开始发送和接收数据；
- 在全双工通讯时，对于发送、接收的数据长度不等的情况，由软件按约定进行处理。
- 主设备发送状态查询包以前，必须把 SS 拉高；主设备在完成状态查询后，一定要把 SS 拉高

25.25 初始化设置

1. 将相关 GPIO 配置为 SPI 模式；
2. 打开 SPI 时钟使能；
3. 根据需要，设置中断使能控制位；
4. 根据需要，设置配置寄存器；
5. 根据应用，设置状态查询命令寄存器。

25.26 状态查询流程

1. 状态查询的过程由硬件实现，软件在初始化设置后，如果准备好接收数据，则：
2. 将控制寄存器的 FIFO_clr 写 1；
3. 将控制寄存器的 Opr_mode[1:0]写入 2’b10；
4. 如果准备好发送数据，则：
5. 将控制寄存器的 FIFO_clr 写 1；
6. 将待发送数据写入到发送数据寄存器，可连续写入两次；
7. 将 Opr_mode[1:0]写入 2’b01。

25.27 接收流程

25.28 查询方式

1. 清 FIFO（可选）；
2. 将 Opr_mode[1:0]写入 2’b10；
3. 等待状态寄存器的 RDR_not_empty 位为 1 时，读接收数据寄存器；
4. 如果数据未接收完成，重复步骤 3，否则跳至下一步；
5. 判断接收到的数据是否符合命令格式约定，如果命令正确，跳至下一步；如果命令不正确，将状态寄存器 2 的 app_cmd_err 位置 1，重新接收；
6. 将 Opr_mode[1:0]写入 2’b00。

25.29 中断方式

1. 清 FIFO（可选）；
2. 将 Opr_mode[1:0]写入 2’b10；
3. 等待中断；
4. 中断服务程序：读一次接收数据寄存器；
5. 退出中断；
6. 如果数据未接收完成，重复步骤 3~5，否则跳至下一步；
7. 将 Opr_mode[1:0]写入 2’b00。

注意事项：

接收数据时，从设备端软件应在 64 个 CPU 时钟周期内完成读接收数据寄存器操作，否则波特率便不能达到系统时钟的 8 分频。

25.30 发送流程

25.31 查询方式

1. 将 Opr_mode[1:0]写入 2’b01；
2. 等待状态寄存器的 SDR_not_full 位为 1 时，将下一个数据写入到发送数据寄存器；
3. 如果继续发送数据，重复步骤 1，否则跳至下一步；
4. 等待状态寄存器的 Send_end 位为 1 时，将 Opr_mode[1:0]写入 2’b00。

25.32 中断方式

1. 将 Opr_mode[1:0]写入 2’b01；
2. 等待中断；
3. 中断服务程序：如果仍有数据要发送，将下一个数据写入到发送数据寄存器；
4. 退出中断；
5. 如果继续发送数据，重复步骤 1~3，否则跳至下一步；
6. 等待状态寄存器的 Send_end 位为 1 时，将 Opr_mode[1:0]写入 2’b00。

注意事项：

发送数据时，从设备端软件应在 64 个 CPU 时钟周期内完成写发送 FIFO 操作，否则波特率便不能达到系统时钟的 8 分频。

25.33 全双工流程

25.34 查询方式

1. 清 FIFO；
2. 将前两个待发送数据写入发送数据寄存器；
3. 将 Opr_mode[1:0]写入 2’b11；
4. 等待状态寄存器的 RDR_not_empty 位为 1 时，读一次接收数据寄存器，将读到的数据丢弃；
5. 等待状态寄存器的 SDR_not_full 位为 1 时，将下一个数据写入到发送数据寄存器；
6. 等待状态寄存器的 RDR_not_empty 位为 1 时，读一次接收数据寄存器；
7. 如果仍有数据要发送，重复步骤 5~6，否则跳至下一步；
8. 等待状态寄存器的 SDR_not_full 位为 1 时，写任意一个数据到发送数据寄存器；
9. 等待状态寄存器的 RDR_not_empty 位为 1 时，读一次接收数据寄存器；
10. 等待状态寄存器的 RDR_not_empty 位为 1 时，读一次接收数据寄存器；
11. 将 Opr_mode[1:0]写入 2’b00。

25.35 中断方式

1. 清 FIFO；
2. 将前两个待发送数据写入发送数据寄存器；
3. 将 Opr_mode[1:0]写入 2’b11；
4. 等待中断；
5. 中断服务程序：
6. 将下一个数据写入到发送数据寄存器，如果待发送的数据已发送完毕，则写入任意数据；
7. 读一次接收数据寄存器（如果是第一次读，将读到的数据丢弃）；
8. 退出中断；
9. 如果发送或接收过程未完成，重复步骤 1~2，否则跳至下一步；
10. 将 Opr_mode[1:0]写入 2’b00。

注意事项：

1. 以全双工方式进行通讯开始之前，从设备端软件应连续向发送数据寄存器写入两次（即第 2 步操作），除非待发送的数据字节数为 1；如果不这样操作，在发第 2 个字节时，会因发送 FIFO 为空而造成 send_ov 标志位会置位，以至后续通讯无法正常进行；
2. 将待发送数据全部写入到发送数据寄存器后，从设备端软件仍需再写入一个任意数据到发送数据寄存器（查询方式的第 8 步和中断服务程序的第 1 步），如果不这样操作，在将待发送数据全部发送完毕后，会因发送 FIFO 为空而使状态机进入溢出状态，造成最后一个数据接收不到。这也是第 1 步存在的原因；
3. 在每传输完一个字节后，硬件进行装载下一个待发送的数据以及存储接收到的数据，从设备端软件应保证此时发送 FIFO 不空、接收 FIFO 不满，否则会使硬件进入溢出状态，造成后续通讯无法正常进行；
4. 在全双工方式下，最快波特率取决于从设备软件的简洁程度，如果从设备端软件无法在 64 个 CPU 时钟周期内完成读接收 FIFO、写发送 FIFO，那么波特率便不能达到系统时钟的 8 分频；
5. 对于发送的数据长度大于接收的数据长度情况，为了防止因接收 FIFO 满溢出而造成通讯终止，从设备软件应按发送的数据长度接收数据，将接收到的多余数据丢弃；

6. 对于发送的数据长度小于接收的数据长度情况，为了防止因发送 FIFO 空溢出而造成通讯终止，从设备软件应按接收的数据长度发送数据。发送的额外数据为任意数据，主设备端软件会根据双方约定而进行丢弃处理。

25.36 SPI 寄存器组

SPI 共有 9 个寄存器，对于这些寄存器的保留位进行读操作，返回 0；对保留位进行写操作，无影响。寄存器列表如下：

地址	寄存器	名称	访问形式	复位值
0x010100	控制寄存器	SPI_CTRL	R/W	0x00
0x010108	配置寄存器	SPI_CONFIG	R/W	0x80
0x010110	接收数据长度寄存器	SPI_LEN	R/W	0x00
0x010118	波特率寄存器	SPI_BAUD	R/W	0x01
0x010120	状态寄存器 1	SPI_STATUS1	R	0x04
0x010128	状态寄存器 2	SPI_STATUS2	R/W	0x00
0x010130	发送数据寄存器	SPI_SDR	W	0x00
0x010138	接收数据寄存器	SPI_RDR	R	0x00
0x010140	状态查询命令寄存器	SPI_QUERY_CMD	R/W	0x55

Figure 25-12 SPI 寄存器组

注意事项：

下列寄存器定义的支持模式表示

- M　　仅在主 模式下有效
- S　　仅在从模式下有效
- S/M　　在从模式和主模式下都有效

25.37 控制寄存器（SPI_CTRL）

SPI_CTRL								
控制寄存器								
地址空间： 0x010100								
位	7	6	5	4	3	2	1	0
符号	--	--	--	--	--	FIFO_clr	Opr_mode[1:0]	
类型	--	--	--	--	--	R/W	R/W	R/W
复位值	--	--	--	--	--	0	0	0
支持模式						M/S	M/S	M/S

位	符号	描述
7：3	--	保留位
2	FIFO_clr	清 FIFO 控制位，将该位置为 1 后，发送 FIFO 和接收 FIFO 将被清 0，FIFO 指针回到 0，一个 CPU 时钟周期后，该位自动变为 0。 设置该控制位是基于如下考虑： 在从设备接收数据前，首先执行清 FIFO 操作，然后进行数据接收，接收到需要长度的数据后，便可以进行命令解析和执行。如此操作后，即便主设备多发数据，也不会造成错误。
1:0	Opr_mode	工作方式控制位 00 - 传输禁止 01 - 发送 10 - 接收 11 - 全双工

Figure 25-13 SPI控制寄存器

注意事项：

1. 在从模式下，将 Opr_mode[1:0]设置为 00 后，如果主设备发来同步时钟，除状态查询命令外，从设备不会进行其它的数据传输。
2. 在主模式下，将 Opr_mode[1:0]设置为 01 后，如果发送 FIFO 变空，则 SCK 会暂停。
3. 在主模式下，将 Opr_mode [1:0]设置为 11 后，如果发送 FIFO 为空，或者接收 FIFO 已满，SCK 均会暂停，等待软件的操作。所以，在通讯过程中，软件应交替写发送数据寄存器、读接收数据寄存器。

25.38 配置寄存器（SPI_CONFIG）

SPI_CONFIG								
配置寄存器								
地址空间： 0x010108								
位	7	6	5	4	3	2	1	0
符号	Master	Frame_len[2:0]			CPHA[1:0]		CPOL	LSBF
类型	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	1	0	0	0	0	0	0	0
支持模式	M/S	M			M/S		M/S	M/S

位	符号	描述
7	Master	主从模式切换控制位 1：SPI 工作在主模式 0：SPI 工作在从模式
6:4	Frame_len [2:0]	帧长度控制位，设置“1”帧中的位长度。 011 - 4 bit 100 - 5 bit 101 - 6 bit 110 - 7 bit others - 8 bit 这三位仅在主模式下有效，在从模式下为复位状态，帧长度固定为 8 位。
3:2	CPHA[1:0]	SPI 触发时钟沿控制位 00： 在 SCK 的奇数沿接收数据，在偶数沿发送数据 01： 在SCK的偶数沿接收数据，在奇数沿发送数据 10： 在SCK的奇数沿接收数据，在奇数沿发送数据 11： 在SCK的偶数沿接收数据，在偶数沿发送数据
1	CPOL	SPI 同步时钟极性控制位 1： 在空闲状态，SCK 为高电平 0： 在空闲状态，SCK 为低电平
0	LSBF	LSB 先传使能控制位 1： 低位先传 0： 高位先传 这一位的设置不影响数据寄存器中 MSB 及 LSB 的位置，bit0 始终为最低位。

Figure 25-14 SPI配置寄存器

注意事项

1. 在发送或接收过程中（Opr_mode[1:0]不等于 0x00），对该寄存器进行写操作无效。
2. 该寄存器的复位由上电复位控制，SPI 使能控制位（spi_en）为 0 时，该寄存器不会复位，以使得再次使能 SPI 时，减少软件的重复设置。
3. 无论工作在主模式还是从模式下，都应该根据对方支持的通信方式对 CPHA[1:0]进行合理的配置，配置方式只有以下 4 种：
    a) 如果对方配置成奇沿接收偶沿发送，CPHA[1:0]应该配置成奇沿接收偶沿发送；
    b) 如果对方配置成偶沿接收奇沿发送，CPHA[1:0]应该配置成偶沿接收奇沿发送；
    c) 如果对方配置成偶沿接收偶沿发送，CPHA[1:0]应该配置成奇沿接收奇沿发送；
    d) 如果对方配置成奇沿接收奇沿发送，CPHA[1:0]应该配置成偶沿接收偶沿发送。

25.39 接收数据长度寄存器（SPI_LEN）

SPI_LEN								
接收数据长度寄存器								
地址空间： 0x010110								
位	7	6	5	4	3	2	1	0
符号	SPI_LEN[7:0]							
类型	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	0	0	0	0	0	0
支持模式	M							

位	符号	描述
7:0	SPI_LEN	写入该寄存器的数据范围为 0~255。 0x00： 待接收的数据长度 = 256； 其它值：待接收的数据长度 = 写入值。 数据长度的倍率基数为帧，而一帧中的位长度由配置寄存器的 Frame_len[2:0]决定。

Figure 25-15 SPI接收数据长度寄存器

注意事项：

1. 在发送或接收过程中（Opr_mode[1:0]不等于 0x00），对该寄存进行写操作无效；
2. 该寄存器的复位由上电复位控制，SPI 使能控制位（spi_en）为 0 时，该寄存器不会复位，以使得再次使能 SPI 时，减少软件的重复设置；
3. 该寄存器仅在主模式下有效，在从模式下，该寄存器始终处于复位状态，SPI 接收的数据长度由应用软件及同步时钟决定。

25.40 波特率寄存器（SPI_BUAD）

SPI_BUAD								
波特率寄存器								
地址空间： 0x010118								
位	7	6	5	4	3	2	1	0
符号	SPI_BUAD[7:0]							
类型	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	0	0	0	0	0	0
支持模式	M							

位	符号	描述
7:0	SPI_BUAD	$\text{Baud Rate} = \frac{\text{BusClock}}{2*\text{SPI\_Baud\_Reg}}$ 设置范围为：1 – 255。

Figure 25-16 SPI波特率寄存器

注意事项：

1. 在发送或接收过程中（Opr_mode[1:0]不等于 0x00），对该寄存进行写操作无效；
2. 如果对该寄存写 0，硬件会自动将写入值变为 1；
3. 该寄存器的复位由上电复位控制，SPI 使能控制位（spi_en）为 0 时，该寄存器不会复位，以使得再次使能 SPI 时，减少软件的重复设置；
4. 该寄存器仅在主模式下有效，在从模式下，该寄存器始终处于复位状态，波特率由主机决定。

25.41 状态寄存器 1（SPI_STATUS1）

SPI_STATUS1								
状态寄存器 1								
地址空间： 0x010120								
位	7	6	5	4	3	2	1	0
符号	--	--	--	--	RDR_not_empty	SDR_not_full	Recv_end	Send_end
类型	--	--	--	--	0	1	0	0
复位值	--	--	--	--	R	R	R	R
支持模式	M/S							

位	符号	描述
7:4	--	保留位
3	RDR_not_empty	接收FIFO不空标志 1： 接收FIFO不空 0： 接收FIFO已空
2	SDR_not_full	发送FIFO不满标志 1： 发送FIFO不满 0： 发送FIFO已满
1	Recv_end	接收结束标志位 1： 接收结束 0： 接收未结束 在主模式下，当按接收数据长度寄存器的设置将数据全部接收完毕后，该位置位；该位置位后，如果向Opr_mode[1:0]写入了2’b00，该位将自动清0。 在从模式下，该位不使用，为复位状态。
0	Send_end	发送结束标志位 1： 发送结束 0： 发送未结束 当写入发送数据寄存器中的数据全部发送完毕后，该位置位；该位置位后，如果又向发送寄存器中写入了数据，或者向Opr_mode[1:0]写入了2’b00，该位将自动清0。

Figure 25-17 SPI状态寄存器1

25.42 状态寄存器 2（SPI_STATUS2）

SPI_STATUS2								
状态寄存器 2								
地址空间： 0x010128								
位	7	6	5	4	3	2	1	0
符号	--	--	send_ov	rcv_ov	app_cmd_err	query_cmd_err	tx_ready	rx_ready
类型	--	--	0	0	0	0	0	0
复位值	--	--	R/W	R/W	R/W	R	R	R
支持模式	S							

位	符号	描述
7:6	--	保留位
5	send_ov	从设备发送溢出标志 1： 从设备发生发送溢出 0： 从设备未发生发送溢出 在数据发送阶段，从设备在装载数据到发送移位寄存器前，硬件会对发送FIFO的状态进行判断，如果发送FIFO状态为空，则该位置位；该位置位后，从设备停止数据的发送，需软件进行干预才能重新开始数据传输，详见注意事项2。
4	rcv_ov	从设备接收溢出标志 1： 从设备发生接收溢出 0： 从设备未发生接收溢出 在命令接收阶段，从设备每收到一个数据后，硬件会对接收FIFO的状态进行判断，如果接收FIFO已满，则该位置位；该位置位后，从设备停止数据的接收，需软件进行干预才能重新开始数据传输，详见注意事项2。
3	app_cmd_err	应用命令错误标志 1： 应用命令错误 0： 应用命令正确 在命令接收阶段，从设备收到命令后，如果从设备端软件经解析判断为命令错误，则应将该位置位；在主设备重发命令阶段，从设备收到第一个字节后，如果不是状态查询命令，该位自动清0。
2	query_cmd_err	查询命令错误标志

		1： 查询命令错误 0： 查询命令正确 在状态查询阶段，如果从设备收到的查询命令不正确，该位由硬件自动置位，片选信号拉高后，该位自动清0。
1	tx_ready	发送准备好标志 1： 从设备已准备好发送数据 0： 从设备尚未准备好发送数据 如果控制寄存器的Opr_mode[1:0]为2’b01或者2’b11，该位置位，否则为0。
0	rx_ready	接收准备好标志 1： 从设备已准备好接收数据 0： 从设备尚未准备好接收数据 如果控制寄存器的Opr_mode[1:0]为2’b10或者2’b11，该位置位，否则为0。

Figure 25-18 SPI状态寄存器2

注意事项：

1. 该寄存器仅在从模式下有效，在主模式下，该寄存器为复位状态。
2. 当发生发送溢出或接收溢出时，需将 SPI 的使能控制信号 spi_en 拉低，使 SPI 内部状态机及状态寄存器复位，再对 SPI 重新配置，才能正常工作。

25.43 发送数据寄存器（SPI_SDR）

SPI_SDR								
发送数据寄存器								
地址空间： 0x010130								
位	7	6	5	4	3	2	1	0
符号	SPI_SDR[7:0]							
类型	W	W	W	W	W	W	W	W
复位值	0	0	0	0	0	0	0	0
支持模式	M/S							

位	符号	描述
7:0	SPI_SDR	状态寄存器的 SDR_not_full 位为 1 时，可以向该寄存器写入数据，写入的数据会进入发送 FIFO。发送 FIFO 的宽度为 8，深度为 2。 主模式下，该寄存器中的有效位长度由配置寄存器的 Frame_len [2:0]决定。 从模式下，该寄存器的有效位长度为 8。

Figure 25-19 SPI发送数据寄存器

注意事项：

1. 当帧长度不为 8 位时，待发送数据写入时，bit0 为最低位；
2. 当状态寄存器的 SDR_not_full 位为 0 时，向该寄存器写入的数据不会进入发送 FIFO。

25.44 接收数据寄存器（SPI_RDR）

SPI_RDR								
接收数据寄存器								
地址空间： 0x010138								
位	7	6	5	4	3	2	1	0
符号	SPI_RDR[7:0]							
类型	R	R	R	R	R	R	R	R
复位值	0	0	0	0	0	0	0	0
支持模式	M/S							

位	符号	描述
7:0	SPI_RDR	该寄存器中的有效位长度由配置寄存器的 Frame_len [2:0]决定。当配置寄存器的 RDR_not_empty 位为 1 时，可以通过读该寄存器获取接收 FIFO 中的数据。接收 FIFO 的宽度为 8，深度为 2。 如果接收 FIFO 已满，又有新的数据接收到，那么接收到的数据仍将留在移位寄存器中，而不会传输至接收数据寄存器中。待接收 FIFO 中的数据被读出以腾出空间后，移位寄存器中的数据再进入接收 FIFO。

Figure 25-20 SPI接收数据寄存器

注意事项：

1. 当帧长度不为 8 位时，bit0 为最低位；
2. 当接收 FIFO 为空时，读出值不定。

25.45 状态查询命令寄存器（SPI_QUERY_CMD）

SPI_QUERY_CMD								
状态查询命令寄存器								
地址空间： 0x010140								
位	7	6	5	4	3	2	1	0
符号	SPI_QUERY_CMD [7:0]							
类型	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	1	0	1	0	1	0	1
支持模式	S							

位	符号	描述
7:0	SPI_QUERY_CMD	根据应用情况，写入该寄存器的值将作为状态查询命令，主设备须按照此寄存器的设置发送状态查询命令。 该寄存器仅在从模式下有效，在主模式下，该寄存器为复位状态。

Figure 25-21 SPI状态查询命令寄存器

26 I²C

26.1 I²C 简介

I²C 总线控制器能满足 I²C 总线的各种规格并支持所有与 I²C 总线通信的传输模式。

I²C 是双线双向的串行总线，它使用连接设备的"SCL"(串行时钟总线)和"SDA"(串行数据总线)来传送信息。数据在主机与从机之间通过 SCL 时钟线控制在 SDA 数据线上实现一字节一字节的同步传输，每个字节为 8 位长度，一个 SCL 时钟脉冲传输一个数据位，数据由最高位 MSB 开始传输，每个传输字节后跟随一个应答位，每个位在 SCL 为高时采样。因此，SDA 线只有在 SCL 为低时才可以改变，在 SCL 为高时 SDA 保持稳定。当 SCL 为高时，SDA 线上的跳变视为命令中断（START 或 STOP)

I²C 逻辑能自主地处理字节的传输。它能保持跟踪串行传送，而且还有一个状态寄存器（I2CSTAT)能反映 I²C 总线控制器和 I²C 总线的状态。

器件的片上 I²C 逻辑提供符合 I²C 总线标准模式规格的串连接口。I²C 端口自动处理字节传输，将 I2CCON 的 ENS1 位设置为 1，即可使能 I²C 模块。 I²C 硬件接口通过 SDA (P11 / P47 串行数据线) 与 SCL (P10 / P46 串行时钟线) 两个引脚连到 I²C 总线。引脚 P11 / P47 与 P10 / P46 用于 I²C 操作需要上拉电阻，因为这两个引脚需要设置为开漏脚。在 I/O 引脚作为 I²C 端口使用时, 用户必须预先设置引脚功能。

根据读写操作位（R/W）状态的不同，I²C 总线上存在以下两种类型的数据传输：

- □ 主发送器向从接收器发送数据。主机发送的第一个字节是从机地址，接下来是数据字节。从机每接收一个字节就返回一个应答位；
- □ 从发送器向主接收器发送数据。主机发送的第一个字节是从机地址，然后从机返回一个应答位。接下来从机向主机发送数据字节。主机每接收一个字节都会返回一个应答位，最后一个字节除外。接收完最后一个字节后，主机返回一个“非应答位”。主机产生所有的串行时钟脉冲、起始条件以及停止条件。每一帧都以一个停止条件或一个重复起始条件来结束。由于重复的起始条件也是下一帧的开始，所以将不会释放 I²C 总线。

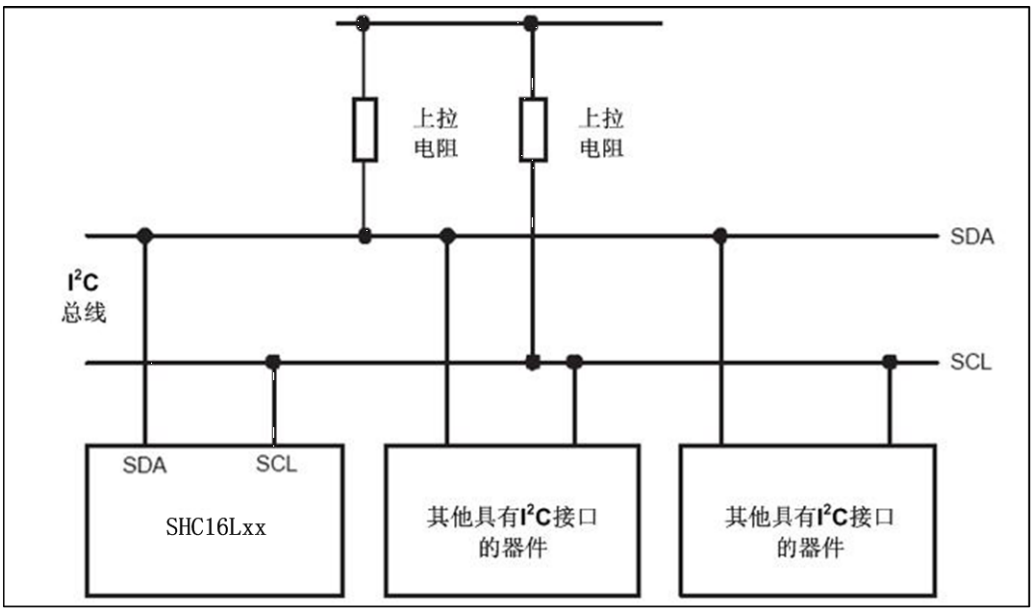


Figure 26-1 I²C 总线配置

26.2 I²C 协议描述

通常标准 I²C 传输协议包含四个部分

(1) 起始条件或重复起始条件

(2) 从机地址传输和 R/W 位传输

(3) 数据传输

(4) 停止条件

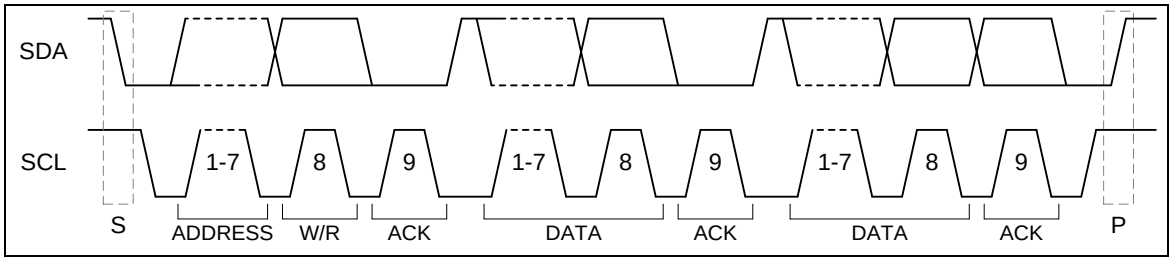


Figure 26-2 I²C 传输协议

26.3 I²C 总线上数据传输

主机发出且由从机接收 7 位地址(一个字节)

传输方向未改变

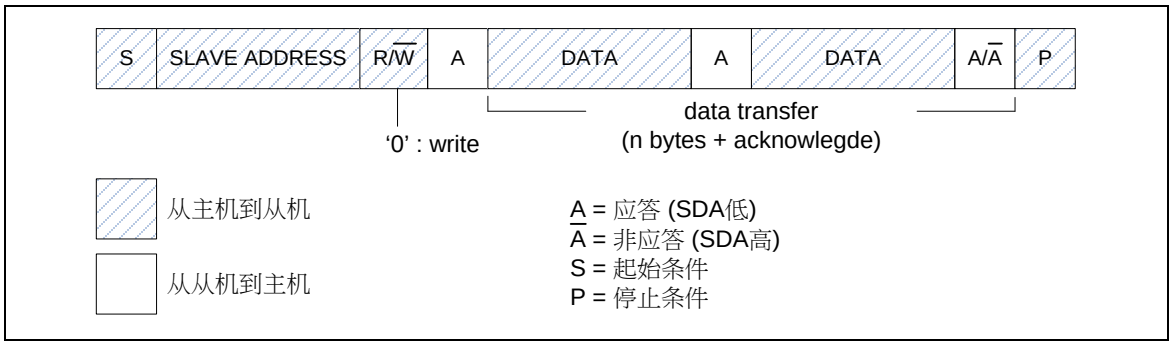


Figure 26-3 主机向从机传输数据

第一个字节后主机紧接着由从机读取数据

传输方向改变

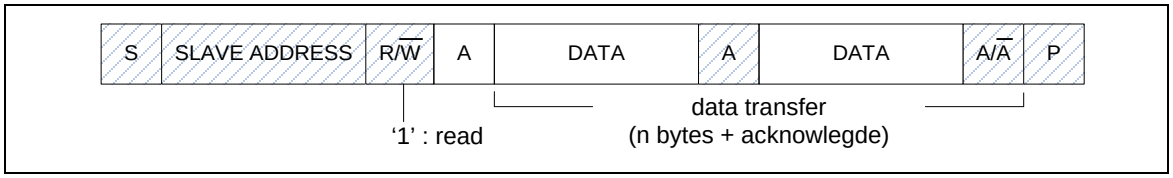


Figure 26-4 主机由从机读取地址

26.4 起始条件或重复起始条件

当总线处于空闲状态下，说明没有主机对总线发起传输请求(SCL 和 SDA 线同时为高)，主机可以通过发送一个起始(START)条件来发起传输请求。

起始条件：通常表示为 S-bit，当 SCL 线为高时，SDA 线上信号由高至低，标示总线上产生起始条件，新的传输开始.

重复起始条件 (Sr) ：即在两个起始(START)条件之间没有停止(STOP)条件. 主机采用这种方法与另一个从机或相同的从机以不同传输方向进行通信 (例如：从写入设备到从设备读出) 而不释放总线.

停止条件：主机向总线发出停止条件结束数据传送。停止条件，通常用 P-bit 表示，当 SCL 线为高时，SDA 线上出现由低到高的信号，被定义为停止条件.

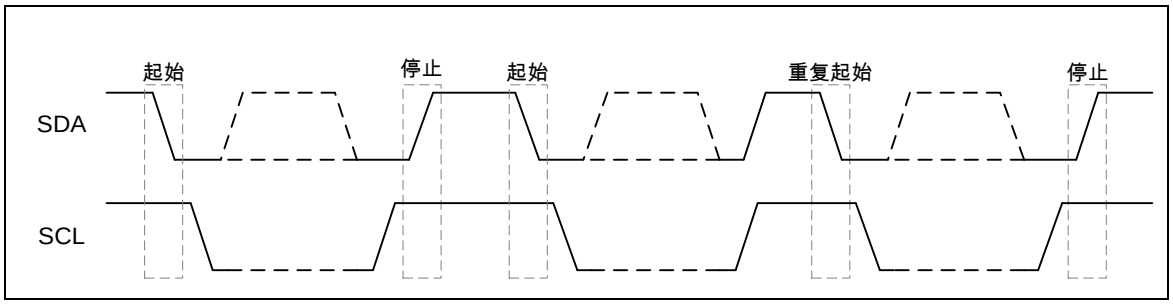


Figure 26-5 START 和 STOP 条件

26.5 从机地址传输

起始(START)条件满足时，主机立即传输数据的第一位。这是一个跟随有一个 RW 位的 7 位调用地址，RW 位控制从机的信号传输方向。系统中没有两个从机有相同的地址，只有被主机寻址的从机会通过在第 9 个 SCL 时钟周期将 SDA 置低电平作为应答。

26.6 数据传输

当从机地址被成功识别，就可以根据 RW 所决定的方向，开始一字节一字节的数据传输，每个传输字节最后带一个第 9 时钟周期上的应答信号，如果从机上产生非应答(NACK)，主机可以产生停止条件来退出数据传输，或者产生重复起始条件开始新一轮的数据传输。

当主机作为接收器件时，产生非应答(NACK) ,从机释放 SDA 线，使主机产生停止条件或重复起始条件.

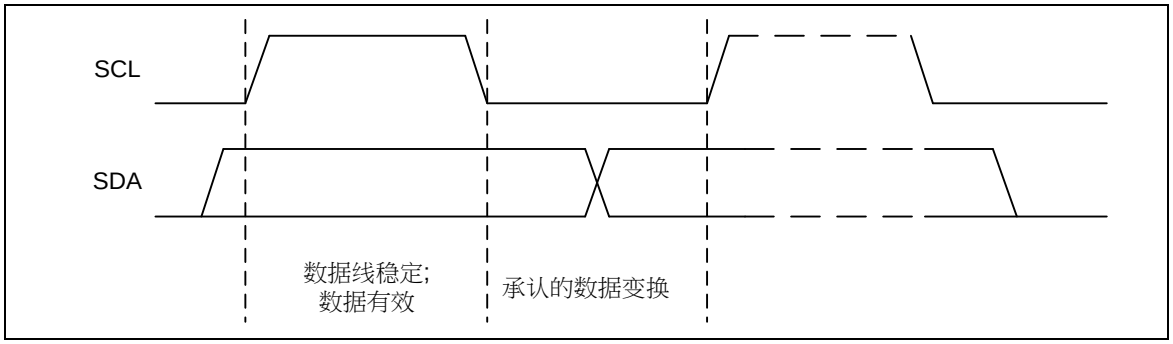


Figure 26-6 I²C 总线上位传输

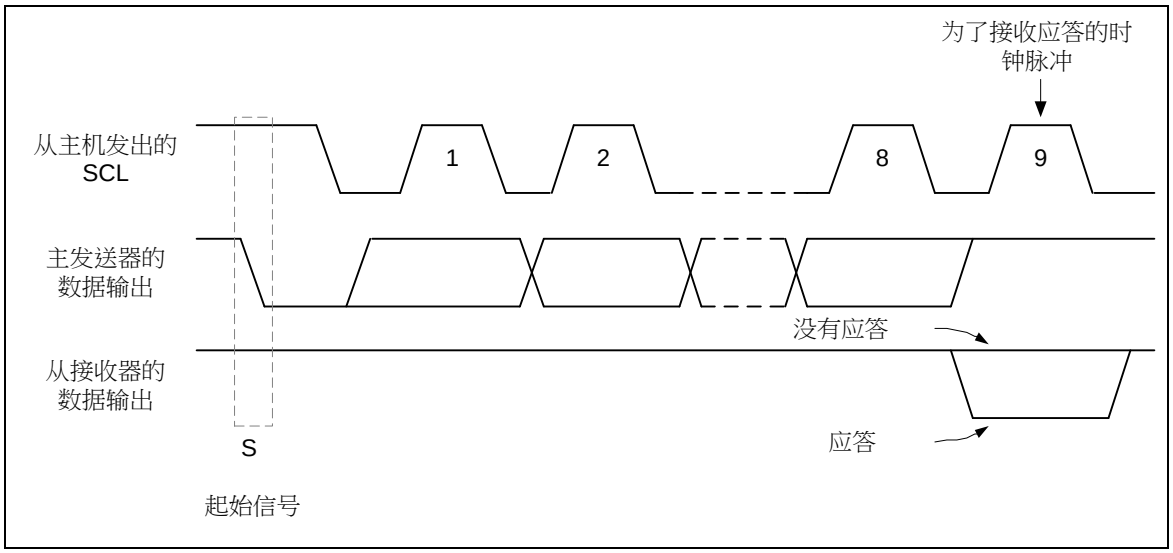


Figure 26-7 I²C 总线上应答信号

26.7 I²C 主要功能描述

I²C 总线使用双线在连接到总线"SCL"(串行时钟线)和"SDA"(串行数据线)的设备间传送信息。由于只有无方向端口，I²C 组件需要使用到引脚的漏端开路缓冲器。每个连接到总线的设备都能使用软件通过特定地址寻址。I²C 标准是一个具有冲突检测机制和仲裁机制的多主机总线。它能防止两个或者多个主机在同时开始传输数据时发生数据冲突。滤波逻辑可以过滤数据总线上的毛刺来保护数据的完整性。

26.8 I²C 操作模式

I²C 组件可实现 8 位的双向数据传输，传输速率在标准模式下可达到 100Kbits/s 而在高速模式下可达 400Kbits/s，并且可以在以下四种模式下操作。

a) 主发送模式：当“SCL”输出串行时钟信号时“SDA”输出串行数据。

b) 主接收模式:当“SCL”输出串行时钟信号时串行数据通过 “SDA”接收。

c) 从接收模式:串行数据和串行时钟分别通过“SDA”和“SCL”接收。

d) 从发送模式:当串行时钟从“SCL”口输入时串行数据通过“SDA”口发送。

26.9 仲裁与同步逻辑

在主发送模式中，仲裁逻辑检查每个发送的逻辑 1 是否真正出现在总线上。如果总线的另一个器件撤消了一个逻辑 1 并将 SDA 线拉低，仲裁丢失，I²C 模块立刻由主发送器变为从接收器。I²C 模块将继续输出时钟脉冲（在 SCL 上），直至发送完当前的串行字节。

仲裁也可能在主接收模式中丢失。这种情况只在 I²C 模块正在向总线返回一个“非应答:（逻辑 1）”时出现。当总线的另一个器件将信号拉低时仲裁丢失。由于它只在串行字节结束时出现，因此 I²C 模块不会再产生时钟脉冲

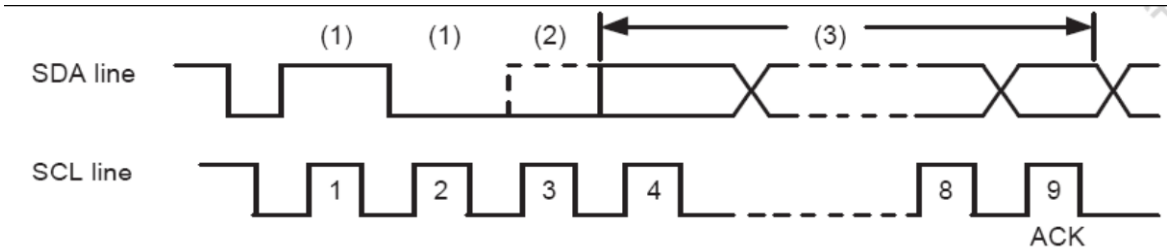


Figure 26-8 I²C 总线仲裁示例

（1）另一器件发送串行数据；

（2）另一器件通过拉低 SDA 先撤消了该 I²C 主机发送的一个逻辑 1（虚线）。仲裁丢失，I²C 进入从接收模式；

（3）此时 I²C 处于从接收模式，但仍产生时钟脉冲，直至发送完当前字节。I²C 将不为下个字节的传输产生时钟脉冲。一旦赢得仲裁，SDA 上的数据传输由新的主机来启动。

同步逻辑使得串行时钟发生器与另一个器件 SCL 线上的时钟脉冲同步。如果 2 个或更多主器件产生时钟脉冲，则高电平周期取决于产生最短高电平时间的器件；低电平周期取决于产生最长低电平时间的器件。

26.10 串行时钟发生器

当 I²C 模块在主发送或主接收模式时，这个可编程时钟脉冲发生器可产生 SCL 时钟脉冲。当 I²C 模块处于从机模式时时钟发生器关闭。时钟发生器的功能由"I2CCON"寄存器的"CR0","CR1"和"CR2"位控制。下边表格展示了主机模式下"clko"可能达到的速率。

表格中的"bclk"输入与 I²C 专用定时器溢出输出口连接。这意味着 I²C 的波特率可由定时器控制。

CR2	CR1	CR0	I²C 波特率
0	0	0	1/256 系统时钟
0	0	1	1/224 系统时钟
0	1	0	1/198 系统时钟
0	1	1	1/170 系统时钟
1	0	0	1/960 系统时钟
1	0	1	1/150 系统时钟
1	1	0	1/90 系统时钟
1	1	1	“bclk”输入除以 8
1	1	1	定时器初值(CNT)=FF 1/64 系统时钟（I²C 最快波特率）
1	1	1	CNT=0xFE 1/88 系统时钟
1	1	1	CNT=0xFD 1/120 系统时钟
1	1	1	CNT=0xFC 1/160 系统时钟
1	1	1	CNT=0xFB 1/200 系统时钟
1	1	1	CNT=0xFA 1/192 系统时钟
1	1	1	CNT=0xF9 1/224 系统时钟
1	1	1	0x00<CNT 设定< 0xFA，满足“bclk”输入除以 8 的条件，即：波特率= 系统时钟/(256-CNT)*32
1		1	CNT=0x00 1/8192 系统时钟（I2C 最慢波特率）

Figure 26-9 I²C 波特率设置

26.11 输入滤波器

输入信号与时钟信号（clk）同步，低于 3 个时钟周期的尖锋脉冲信号会被滤除。每个滤波器由 3 个触发器组成。第一个触发器用来直接锁存输入信号，并将数据载入由另外两个构成的移位寄存器中。当第二和第三个触发器的状态是"11"或"00"时，内部的滤除信号会各自被置 1 或置 0。

26.12 地址比较器

I²C 比较器将自己的从机地址与接收到的 7 位从机地址做比较。它可使用"I2CADR"寄存器对自己的从机地址进行编程。并且会根据"I2CADR"寄存器的"GC"位与首次接收到的 8 位字节与通用调用地址（0x00）相比较。如果任何一者相同，"I2CCON"寄存器的"SI"位会被置 1 并产生一个中断请求。

26.13 中断产生器

I²C 模块的所有四种模式都被使用时，则有 26 种可能的总线状态。当 I²C 进入 26 种状态中的 25 种状态时，"I2CCON"寄存器的"SI"标志位会被硬件置 1。"SI"位唯一不会被置 1 的状态是:F8h(状态代码)，这表明没有有效的相关状态信息。"SI"标志位必须通过软件清零。为了清除"SI"位，必须把 0 写入此位。若在"SI"里写 1 不会改变"SI"的值。

I²C 中断变量由 AIE.I2CE 使能。在被写入中断控制器前，"SI"信号是 I²C 的标志位。为了确定中断的实际中断源，中断服务程序在清除"SI"标志位之前，会对 I²C 状态寄存器进行查询。

26.14 I²C 模式状态

26.15 I²C 主发送模式

必须将I2CCON.ENS1置“1”来使能I²C模块。如果I2CCON.AA位复位，当另一个器件正变成总线主机时，I²C模块将不会应答其自身的从机地址或通用调用地址，I²C接口就不能进入从机模式。I2CCON寄存器的STA，STO和SI位必须复位。

此时，可通过置位I2CCON.STA位进入主发送模式。一旦总线空闲，I²C逻辑会马上测试I²C总线并产生一个起始条件。当发送起始条件时，串行中断标志（SI）置位，状态寄存器（I2CSTAT）中的状态代码为0x08。中断服务程序利用该状态代码进入相应的状态服务程序，将从机地址和数据方向位（SLA+W）装入I2CDAT。I2CCON的SI位必须在串行传输继续之前复位。

当发送完从机地址和方向位且接收到一个应答位时，串行中断标志（SI）再次置位，I2STAT中可能是一系列不同的状态代码。主机模式下为0x18，0x20或0x38，从机模式（I2CCON.AA=逻辑1）为0x68，0x78或0xB0。每个状态代码对应的操作在下表中详细介绍。在发送完重复起始条件（状态0x10）后，I²C模块通过将SLA+R装入I2CDAT切换到主接收模式。

在下面的表格中，“SLA”代表了从机地址，"R"代表了与从机地址一起发送的R/W位=1。“W”代表了与从机地址一起发送的R/W位=0。

状态代码	I²C 总线和硬件状态	应用软件响应					I²C 硬件执行的下一个动作
		读/写 I2CDAT	写 I2CCON				
			STA	STO	SI	AA	
08H	已发送起始条件	装入 SLA+W	X	0	0	X	将发送 SLA+W ，接收应答位(ACK)
10H	已发送重复起始条件	装入 SLA+W	X	0	0	X	同上
		装入 SLA+R	X	0	0	X	将发送 SLA +R ， I²C 自动切换到主接收模式
18H	已发送 SLA+W 已接收 ACK	装入数据字节	0	0	0	X	将发送数据字节，将接收 ACK
		无 I2CDAT 动作	1	0	0	X	将发送重复起始条件
		无 I2CDAT 动作	0	1	0	X	将发送停止条件，STO 标志位复位
		无 I2CDAT 动作	1	1	0	X	将先发送停止条件，随后发送起始条件，STO 标志位复位
20H	已发送 SLA +W 已接收非 ACK	装入数据字节	0	0	0	X	将发送数据字节，将接收 ACK
		无 I2CDAT 动作	1	0	0	X	将发送重复起始条件
		无 I2CDAT 动作	0	1	0	X	将发送停止条件，STO 标志位复位
		无 I2CDAT 动作	1	1	0	X	将先发送停止条件，随后发送起始条件，STO 标志位复位
28H	已发送 I2CDAT 中的数据; 已接收 ACK	装入数据字节	0	0	0	X	将发送数据字节，将接收 ACK
		无 I2CDAT 动作	1	0	0	X	将发送重复起始条件
		无 I2CDAT 动作	0	1	0	X	将发送停止条件，STO 标志位复位
		无 I2CDAT 动作	1	1	0	X	将先发送停止条件，随后发送起始条件，STO 标志位复位
30H	已发送 I2CDAT 中的数据;	装入数据字节	0	0	0	X	将发送数据字节，将接收 ACK
		无 I2CDAT 动作	1	0	0	X	将发送重复起始条件

		无 I2CDAT 动作	0	1	0	X	将发送停止条件，STO 标志位复位
		无 I2CDAT 动作	1	1	0	X	将先发送停止条件，随后发送起始条件，STO 标志位复位
38H	在 SLA+R/W 或写数据字节时丢失仲裁	无 I2CDAT 动作	0	0	0	X	I^2C 总线被释放。 进入不可寻址从模式。
		无 I2CDAT 动作	1	0	0	X	当 I^2C 总线空闲时发送起始条件

Figure 26-2 I^2C 主发送模式表

26.16 I²C 主接收模式

在主接收模式中，主机所接收的数据字节来自从发送器。按主发送模式中的方法初始化传输。当发送完起始条件后，中断服务程序必须把7位从机地址和数据方向位（SLA+R）装入I2CDAT。必须先清除I2CCON中的SI位，再继续执行串行传输。

当发送完从机地址和数据方向位且接收到一个应答位时，串行中断标志SI再次置位，这时，I2CSTAT中可能是一系列不同的状态代码。主机模式下为0x40，0x48或0x38，从机模式（I2CCON.AA=1）为0x68，0x78或0xB0。每个状态代码对应的操作详见下表。在发送完重复起始条件（状态0x10）后，I²C模块通过将SLA+W装入I2CDAT切换到主发送模式。

状态代码	I2C 总线和硬件状态	应用软件响应					I2C 硬件执行的下一个动作
		读/写 I2CDAT	写 I2CCON				
			STA	STO	SI	AA	
08H	已发送起始条件	装入 SLA+R	X	0	0	X	将发送 SLA+R；接收 ACK；
10H	已发送重复起始条件	装入 SLA+R	X	0	0	X	同上
		装入 SLA+W	X	0	0	X	将发送 SLA +W ； I2C 模块将切换进入主发送模式；
38H	在非 ACK 中丢失仲裁	无 I2CDAT 动作	0	0	0	X	I2C 总线将被释放；进入从模式；
		无 I2CDAT 动作	1	0	0	X	当总线空闲时发送起始条件
40H	已发送 SLA+R 已接收 ACK	无 I2CDAT 动作	0	0	0	0	将接收数据字节，将返回非 ACK
		无 I2CDAT 动作	0	0	0	1	将接收数据字节，将返回 ACK
48H	已发送 SLA+R 已接收非 ACK	无 I2CDAT 动作	1	0	0	X	将发送重复的起始条件
		无 I2CDAT 动作	0	1	0	X	将发送停止条件； STO 标志位将复位
		无 I2CDAT 动作	1	1	0	X	将先发送停止条件，随后发送起始条件，STO 标志位复位
50H	已接收数据字节 ACK 已返回	读取数据字节	0	0	0	0	将接收数据字节，将返回非 ACK
		读取数据字节	0	0	0	1	将接收数据字节，将返回 ACK
58H	已接收数据字节非 ACK 已返回	读取数据字节	1	0	0	X	将发送重复的起始条件
		读取数据字节	0	1	0	X	将发送停止条件；STO 标志位将复位
		读取数据字节	1	1	0	X	将先发送停止条件，随后发送起始条件，STO 标志位复位

Figure 26-3 I²C 主接收模式表

26.17 I²C 从接收模式

在从接收模式中，从机接收的数据字节来自主发送器。高7位是主机寻址时I²C模块响应的地址。如果LSB（I2CADR.GC）被置位，I²C模块将响应通用调用地址（0x00）；否则忽略通用调用地址。

I²C总线速率的设置不影响从机模式中的I²C模块。必须置位I2CCON.ENS1来使能I²C模块。I2CCON.AA位必须置位以使能I²C模块来应答其自身从机地址或通用调用地址。I2CCON的STA，STO和SI位必须复位。

当I2CADR和I2CCON完成初始化后，I²C模块会一直等待，直至被从机地址寻址，之后是数据方向位寻址。为了工作在从接收模式中，数据方向位必须为“0”（W）。接收完其自身的从机地址和W位后，串行中断标志（SI）置位，可从I2CSTAT中读出一个有效的状态代码。该状态代码用作状态服务程序的向量。每个状态代码的对应操作见下表。如果I²C模块在主机模式中仲裁丢失，也可进入从接收模式（请参考状态0x68和0x78的描述）。

如果I2CCON.AA位在传输过程中复位，则在接收完下一个数据字节后I²C模块将向SDA返回一个非应答（逻辑1）。当AA复位时，I²C模块不响应其自身的从机地址或通用调用地址。但是，I²C总线仍被监控，而且，地址识别可随时通过置位I2CCON.AA来恢复。这就意味着I2CCON.AA位可临时将I²C模块从I²C总线上分离出来。

状态代码	I²C 总线和硬件状态	应用软件响应					I²C 硬件执行的下一个动作
		读/写 I2CDAT	写 I2CCON				
			STA	STO	SI	AA	
60H	已接收自身的 SLA+W； 已接收 ACK	无 I2CDAT 动作	X	0	0	0	将接收数据字节，将返回非 ACK
		无 I2CDAT 动作	X	0	0	1	将接收数据字节，将返回 ACK
68H	主控时在 SLA+R/W 丢失仲裁； 已接收自身的 SLA+W ； 已返回 ACK；	无 I2CDAT 动作	X	0	0	0	将接收数据字节，将返回非 ACK
		无 I2CDAT 动作	X	0	0	1	将接收数据字节，将返回 ACK
70H	已接收通用调用地址（0x00）； 已返回 ACK；	无 I2CDAT 动作	X	0	0	0	将接收数据字节，将返回非 ACK
		无 I2CDAT 动作	X	0	0	1	将接收数据字节，将返回 ACK
78H	主控时在 SLA+R/W 中丢失仲裁；已接收通用调用地址； 已返回 ACK；	无 I2CDAT 动作	X	0	0	0	将接收数据字节，将返回非 ACK
		无 I2CDAT 动作	X	0	0	1	将接收数据字节，将返回 ACK
80H	前一次寻址使用自身从地址； 已接收数据字节； 已返回 ACK；	无 I2CDAT 动作	X	0	0	0	将接收数据字节，将返回非 ACK
		无 I2CDAT 动作	X	0	0	1	将接收数据字节，将返回 ACK
88H	前一次寻址使用自身从地址；	读取数据字节	0	0	0	0	切换到不可寻址从模式； 不识别自身从地址或通用地

	已接收数据字节； 已返回非 ACK;						址；
		读取数据字节	0	0	0	1	切换到不可寻址从模式； 识别自身从地址或通用地址；
		读取数据字节	1	0	0	0	切换到不可寻址从模式； 识别自身从地址或通用地址；
		读取数据字节	1	0	0	1	切换到不可寻址从模式； 识别自身从地址或通用地址； 当总线空闲后发送起始条件；
90H	前一次寻址使用通用调用地址；已接收数据； 已返回 ACK；	读取数据字节	X	0	0	0	将接收数据字节，将返回非 ACK
		读取数据字节	X	0	0	1	将接收数据字节，将返回 ACK
98H	前一次寻址使用通用调用地址；已接收数据； 已返回非 ACK;	读取数据字节	0	0	0	0	切换到不可寻址从模式； 不识别自身从地址或通用地址；
		读取数据字节	0	0	0	1	切换到不可寻址从模式； 识别自身从地址或通用地址；
		读取数据字节	1	0	0	0	切换到不可寻址从模式； 不识别自身从地址或通用地址； 当总线空闲后发送起始条件；
		读取数据字节	1	0	0	1	切换到不可寻址从模式； 识别自身从地址或通用地址； 当总线空闲后发送起始条件；
A0H	当使用从接收/从发送模式中静态寻址时，接收到停止条件或重复起始条件	无 I2CDAT 动作	0	0	0	0	切换到不可寻址从模式； 不识别自身从地址或通用地址；
		无 I2CDAT 动作	0	0	0	1	切换到不可寻址从模式； 识别自身从地址或通用地址；
		无 I2CDAT 动作	1	0	0	0	切换到不可寻址从模式； 不识别自身从地址或通用地址； 当总线空闲后发送起始条件；
		无 I2CDAT 动作	1	0	0	1	切换到不可寻址从模式； 识别自身从地址或通用地址； 当总线空闲后发送起始条件；

Figure 26-4 I^2C 从接收模式表

26.18 I²C 从发送模式

在从发送模式中，向主接收器发送数据字节。数据传输按照从接收模式中的情况初始化。当初始化I2CADR和I2CCON后，I²C模块会一直等待，直至被自身的从机地址寻址。之后是数据方向位，该数据方向位必须为“1”（R），以便I²C模块工作在从发送模式下。接收完其自身的从机地址和R位后，串行中断标志（SI）置位，并且可从I2CSTAT中读取一个有效的状态代码。该状态代码用作状态服务程序的向量，每个状态代码的对应操作见下表所示。如果I²C模块在主机模式下时仲裁丢失，则可进入从发送模式（见状态0xB0）。

如果I2CCON.AA位在传输过程中复位，则I²C模块将发送最后一个字节并进入状态0xC0或0xC8。I²C模块切换到非寻址的从机模式，如果继续传输，它将忽略主接收器。因此主接收器接收所有1作为串行数据。当I2CCON.AA复位时，I²C模块不响应其自身的从机地址或通用调用地址。但是，I²C总线仍被监控，而且，地址识别可随时通过置位I2CCON.AA来恢复。这就意味着I2CCON.AA位可用来暂时将I²C模块从I²C总线上分离出来。

状态代码	I²C 总线和硬件状态	应用软件响应					I²C 硬件执行的下一个动作
		读/写 I2CDAT	写 I2CCON				
			STA	STO	SI	AA	
A8H	已接收自身的 SLA+R ；已返回 ACK	装入数据字节	X	0	0	0	将发送最后一个数据字节；将接收 ACK；
		装入数据字节	X	0	0	1	将发送一个数据字节；将接收 ACK
B0H	当主控时在 SLA+R/W 中丢失仲裁；已接收自身 SLA+R；已返回 ACK；	装入数据字节	X	0	0	0	将发送最后一个数据字节；将接收 ACK；
		装入数据字节	X	0	0	1	将发送一个数据字节；将接收 ACK
B8H	已发送数据；已接收 ACK；	装入数据字节	X	0	0	0	将发送最后一个数据字节；将接收 ACK；
		装入数据字节	X	0	0	1	将发送一个数据字节；将接收 ACK；
C0H	已发送数据字节；已接收非 ACK；	无 I2CDAT 动作	0	0	0	0	切换到不可寻址从模式；不识别自身从地址或通用地址；
		无 I2CDAT 动作	0	0	0	1	切换到不可寻址从模式；识别自身从地址或通用地址；
		无 I2CDAT 动作	1	0	0	0	切换到不可寻址从模式；不识别自身从地址或通用地址；当总线空闲后发送起始条件；
		无 I2CDAT 动作	1	0	0	1	切换到不可寻址从模式；识别自身从地址或通用地址；当总线空闲后发送起始条件；
C8H	装入的数据字节已被发送；已接收 ACK；	无 I2CDAT 动作	0	0	0	0	切换到不可寻址从模式；不识别自身从地址或通用地址；
		无 I2CDAT 动作	0	0	0	1	切换到不可寻址从模式；识别自身从地址或通用地址；
		无 I2CDAT 动作	1	0	0	0	切换到不可寻址从模式；

							不识别自身从地址或通用地址； 当总线空闲后发送起始条件；
		无 I2CDAT 动作	1	0	0	1	切换到不可寻址从模式； 识别自身从地址或通用地址； 当总线空闲后发送起始条件；

Figure 26-5 I²C 从发送模式表

26.19 I²C 其他杂项状态

I2CSTAT= 0xF8：

这个状态码表示没有任何可用的相关信息，因为串行中断标志I2CCON.SI还没有置位。这种情况在其它状态和I²C模块还未开始执行串行传输之间出现。

I2CSTAT= 0x00：

该状态代码表示在I²C串行传输过程中出现了总线错误。当格式帧的非法位置上出现了起始或停止条件时总线错误产生。这些非法位置是指在串行传输过程中的地址字节、数据字节或应答位。当外部干扰影响到内部I²C模块信号时也会产生总线错误。总线错误出现时I2CCON.SI置位。

要从总线错误中恢复，I2CCON.STO标志必须置位，I2CCON.SI必须被清除。这使得I²C模块进入“非寻址的”从机模式（已定义的状态）并清除I2CCON.STO标志（I2CCON中的其它位不受影响）。SDA和SCL线被释放（不发送停止条件）。

状态代码	I²C 总线和硬件状态	应用软件响应					I²C 硬件执行的下一个动作
		读/写 I2CDAT	写 I2CCON				
			STA	STO	SI	AA	
F8H	无可用的相关状态信息；SI=0；	无 I2CDAT 动作	无 I2CDAT 动作				等待或执行当前传输
00H	由于非法的起始或停止条件的出现，在主机或被选中的从机将出现总线错误； 当外部干扰使 I2C 进入未定义的状态时也会出 0x00 状态	无 I2CDAT 动作	0	1	0	X	只有在主机或被寻址的从机模式中，内部硬件受影响。一般情况下，总线被释放，I2C 模块切换到非寻址的从机模式。STO 复位。

Figure 26-6 I²C 其他杂项状态

26.20 操作模式范例

26.21 初始化程序

将 I^2C 接口初始化用作从机和/或主机的例子。

a）将自身的从机地址装入 I2CADR，使能通用调用识别 I2CADR.GC（如果需要的话）；

b）使能 I^2C 中断 AIE.I2CE；

c）向寄存器 I2CCON 写入 0x44 来置位 ENS1 和 AA 位，使能从机功能。对于主机功能，可向寄存器 I2CCON 写入 0x40。

26.22 启动主机发送功能

通过建立缓冲区、指针和数据计数然后发启起始条件便可执行主发送操作。

a）初始化主机数据计数器；

b）建立数据将被发送到的从机地址，并且添加写位；

c）向 I2CCON 写入 0x20 来置位 STA 位；

d）在主发送缓冲区内建立要发送的数据；

e）初始化主机数据计数器来匹配正在发送的信息长度；

f）退出。

26.23 启动主机接收功能

通过建立缓冲区、指针和数据计数然后发启起始条件便可执行主接收操作。

a）初始化主机数据计数器；

b）建立数据将被发送到的从机地址，并且添加读位；

c）向 I2CCON 写入 0x20 来置位 STA 位；

d）在主接收缓冲区内建立要发送的数据；

e）初始化主机数据计数器来匹配正在发送的信息长度；

f）退出。

26.24 I^2C 中断程序

确定 I^2C 的状态和处理该状态的状态程序。

a）从 I2CSTAT 中读出 I^2C 的状态；

b）使用状态值跳转到 26 个可能状态程序中的一个。

26.25 无指定模式的状态

（1）状态：0x00 总线错误。

进入非寻址的从机模式并释放总线。

a）向 I2CCON 写入 0x14 来置位 STO 和 AA 位；

b）向 I2CCON 写入 !0x08 来清除 SI 标志；

c）退出。

（2）状态：0x08

已发送起始条件。即将发送从机地址＋R/W 位和接收 ACK 位。

a）向 I2CDAT 写入从机地址和 R/W 位；

b）向 I2CCON 写入 0x04 来置位 AA 位；

c）向 I2CCON 写入 !0x08 来清除 SI 标志；

d）建立主发送模式数据缓冲区；

e）建立主接收模式数据缓冲区；

f）初始化主机数据计数器；

g）退出。

（3）状态：0x10

已发送重复起始条件。即将发送从机地址＋R/W 位和接收 ACK 位。

a）向 I2CDAT 写入从机地址和 R/W 位；

b）向 I2CCON 写入 0x04 来置位 AA 位；

c）向 I2CCON 写入 !0x08 来清除 SI 标志；

d）建立主发送模式数据缓冲区；

e）建立主接收模式数据缓冲区；

f）初始化主机数据计数器；

g）退出。

注意事项:

状态 0x08 和 0x10 适用于主发送模式和主接收模式。R/W 位决定了下一个状态是在主发送模式中还是在主接收模式中。

26.26 主发送器状态

（1）状态：0x18

之前状态为 0x08 或 0x10 表示已发送从机地址和写操作位，并接收了应答。即将发送第一个数据字节和接收 ACK 位。

a）将主发送缓冲区的第一个数据字节装入 I2CDAT；

b）向 I2CCON 写入 0x04 来置位 AA 位；

c）向 I2CCON 写入 !0x08 来清除 SI 标志；

d）主发送缓冲区指针加 1；

e）退出。

（2）状态：0x20

已发送从机地址和写操作位并接收了非应答。即将发送停止条件。

a）向 I2CCON 写入 0x14 来置位 STO 和 AA 位；

b）向 I2CCON 写入 !0x08 来清除 SI 标志；

c）退出。

（3）状态：0x28

已发送数据并接收了 ACK。如果发送的数据是最后一个数据字节则发送一个停止条件，否则发送下一个数据字节。

a）主机数据计数器减 1，如果发送的不是最后一个数据字节就跳至第 e）步；

b）向 I2CCON 写入 0x14 来置位 STO 和 AA 位；

c）向 I2CCON 写入 !0x08 来清除 SI 标志；

d）退出；

e）将主发送缓冲区的下一个数据字节装入 I2CDAT；

f）向 I2CCON 写入 0x04 来置位 AA 位；

g）向 I2CCON 写入 !0x08 来清除 SI 标志；

h）主机发送缓冲区指针加 1；

i）退出。

（4）状态：0x30

已发送数据并接收到非应答。即将发送停止条件；

a）向 I2CCON 写入 0x14 来置位 STO 和 AA 位；

b）向 I2CCON 写入 !0x08 来清除 SI 标志；

c）退出。

（5）状态：0x38

仲裁已在发送从机地址和写操作位或数据的过程中丢失。总线已被释放且进入非寻址的从 机模式。当总线再次空闲时将发送一个新的起始条件。

a）向 I2CCON 写入 0x24 来置位 STA 和 AA 位；

b）向 I2CCON 写入 !0x08 来清除 SI 标志；

c）退出。

26.27 主接收器状态

（1）状态：0x40

前面的状态是 0x08 或 0x10 表示已发送从机地址和读操作位，并接收到 ACK。将接收数据和返回 ACK。

a）向 I2CCON 写入 0x04 来置位 AA 位；

b）向 I2CCON 写入 !0x08 来清除 SI 标志；

c）退出。

（2）状态：0x48

已发送从机地址和读操作位，并接收到非应答。将发送停止条件。

a）向 I2CCON 写入 0x14 来置位 STO 和 AA 位；

b）向 I2CCON 写入 !0x08 来清除 SI 标志；

c）退出。

（3）状态：0x50

已接收到数据，并返回 ACK。将从 I2CDAT 读取数据,并将接收其它的数据。如果这是最后一个数据字节，则返回非应答，否则返回 ACK。

a）读取 I2CDAT 中的数据字节，存放到主机接收缓冲区；

b）主机数据计数器减 1，如果不是最后一个数据字节就跳到第 e）步；

c）向 I2CCON 写入 !0x0C 来清除 SI 标志和 AA 位；

d）退出；

e）向 I2CCON 写入 0x04 来置位 AA 位；

f）向 I2CCON 写入 !0x08 来清除 SI 标志；

g）主机接收缓冲区指针加 1；

h）退出。

（4）状态：0x58

已接收到数据，已返回非应答。将从 I2CDAT 中读取数据和发送停止条件。

a）读取 I2CDAT 中的数据字节，存放到主机接收缓冲区；

b）I2CCON 写入 0x14 来置位 STO 和 AA 位；

c）向 I2CCON 写入 !0x08 来清除 SI 标志；

d）退出。

26.28 从接收器状态

（1）状态：0x60

已接收到自身从机地址和写操作位，已返回 ACK。将接收数据和返回 ACK。

a）向 I2CCON 写入 0x04 来置位 AA 位；

b）向 I2CCON 写入 !0x08 来清除 SI 标志；

c）建立从接收模式数据缓冲区；

d）初始化从机数据计数器；

e）退出。

（2）状态：0x68

用作总线主机时仲裁已在传输从机地址和 R/W 位时丢失。已接收到自身从机地址和写操作位，并已返回 ACK。将接收数据和返回 ACK。当总线再次空闲后置位 STA 来重启主机模式。

a）向 I2CCON 写入 0x24 来置位 STA 和 AA 位；

b）向 I2CCON 写入 !0x08 来清除 SI 标志；

c）建立从接收模式数据缓冲区；

d）初始化从机数据计数器；

e）退出。

（3）状态：0x70

已接收到通用调用地址和返回 ACK。将接收数据和返回 ACK。

a）向 I2CCON 写入 0x04 来置位 AA 位；

b）向 I2CCON 写入 !0x08 来清除 SI 标志；

c）建立从接收模式数据缓冲区；

d）初始化从机数据计数器；

e）退出。

（4）状态：0x78

用作总线主机时仲裁已在传输从机地址和 R/W 位时丢失。已接收到通用调用地址和返回 ACK。 将接收数据和返回 ACK。当总线再次空闲后置位 STA 来重启主机模式。

a）向 I2CCON 写入 0x24 来置位 STA 和 AA 位；

b）向 I2CCON 写入 !0x08 来清除 SI 标志；

c）建立从接收模式数据缓冲区；

d）初始化从机数据计数器；

e）退出。

（5）状态：0x80

之前寻址自身从机地址。已接收到数据并返回 ACK。将读取其它数据。

a）读取 I2CDAT 的数据字节，存放到从机接收缓冲区。

b）从机数据计数器减 1，如果不是最后一个数据字节就跳到第 e）步；

c）向 I2CCON 写入 !0x0C 来清除 SI 标志和 AA 位；

d）退出；

e）向 I2CCON 写入 0x04 来置位 AA 位；

f）向 I2CCON 写入 !0x08 来清除 SI 标志；

g）从机接收缓冲区指针加 1；

h）退出。

（6）状态：0x88

之前寻址自身从机地址。已接收到数据并返回非应答。不会保存接收到的数据。进入非寻址的从机模式。

a）向 I2CCON 写入 0x04 来置位 AA 位；

b）向 I2CCON 写入 !0x08 来清除 SI 标志；

c）退出。

（7）状态：0x90

之前寻址通用调用地址，已接收到数据并返回 ACK，将保存接收到的数据。只接收第一个数据字节并返回 ACK。接收其它数据字节后返回非应答。

a）读取 I2CDAT 的数据字节，并放入从机接收缓冲区；

b）向 I2CCON 写入 !0x0C 来清除 SI 标志和 AA 位；

c）退出。

（8）状态：0x98

之前寻址通用调用地址，已接收到数据并返回非应答，不会保存接收到的数据。进入非寻址的从机模式。

a）向 I2CCON 写入 0x04 来置位 AA 位；

b）向 I2CCON 写入 !0x08 来清除 SI 标志；

c）退出。

（9）状态：0xA0

已接收停止条件或重复起始条件，但仍作为从机寻址。不保存接收到的数据。进入非寻址的从机模式。

a）向 I2CCON 写入 0x04 来置位 AA 位；

b）向 I2CCON 写入 !0x08 来清除 SI 标志；

c）退出。

26.29 从发送器状态

（1）状态：0xA8

已接收自身从机地址和读操作位并返回 ACK。将发送数据和接收 ACK 位。

a）将从机发送缓冲区的第一个数据字节装入 I2CDAT；

b）向 I2CCON 写入 0x04 来置位 AA 位；

c）向 I2CCON 写入 !0x08 来清除 SI 标志；

d）建立从发送模式数据缓冲区；

e）从机发送缓冲区指针加 1；

f）退出。

（2）状态：0xB0

用作总线主机时，在传输从机地址和 R/W 位时丢失仲裁。已接收自身从机地址和读操作位并返回 ACK。将发送数据和接收 ACK 位。当总线再次空闲后置位 STA 来重启主机模式。

a）将从机发送缓冲区的第一个数据字节装入 I2CDAT；

b）向 I2CCON 写入 0x24 来置位 STA 和 AA 位；

c）向 I2CCON 写入 !0x08 来清除 SI 标志；

d）建立从发送模式数据缓冲区；

e）从机发送缓冲区指针加 1；

f）退出。

（3）状态：0xB8

已发送数据并接收到 ACK。将发送数据和接收 ACK 位。

a）将从机发送缓冲区的数据字节装入 I2CDAT；

b）向 I2CCON 写入 0x04 来置位 AA 位；

c）向 I2CCON 写入 !0x08 来清除 SI 标志；

d）从机发送缓冲区指针加 1；

e）退出。

（4）状态：0xC0

已发送数据并接收到非应答。进入非寻址的从机模式。

a）向 I2CCON 写入 0x04 来置位 AA 位；

b）向 I2CCON 写入 !0x08 来清除 SI 标志；

c）退出。

（5）状态：0xC8

已发送最后一个数据字节并接收到 ACK。进入非寻址的从机模式。

a）向 I2CCON 写入 0x04 来置位 AA 位；

b）向 I2CCON 写入 !0x08 来清除 SI 标志；

c）退出。

26.30 I²C 寄存器组

微处理器与I²C组件通过以下四种特殊功能寄存器相联系：“I2CCON”(控制寄存器)，“I2CSTAT”(状态寄存器)，“I2CDAT”(数据寄存器)和“I2CADR”(地址寄存器)。

“I2CCON”寄存器包含有I²C全局使能信号位“ENS1”以及波特率设置位(“CR0”，“CR1”和“CR2”)。它也提供了能初始化发送起始或者停止条件给I²C总线的信号位(“STA”，“STO”位)。在接收到自身从机地址或者通用地址寻址或在主机/从机模式下接到数据之后，“AA”信号位控制I²C发送中的ACK位。最后 ，“I2CCON”提供了中断请求标志位“SI”。

“I2CSTAT”寄存器反映了I²C组件的主FSM状态，此寄存器的三个最低有效位一直保持为0。

“I2CDAT”寄存器包含有一个被I²C总线发送的字节或一个从I²C总线接收到的字节。“I2CDAT”寄存器并没有被屏蔽或者是双缓冲，因此MCU在I²C中断发生时会读此寄存器。

“I2CADR”寄存器存储着I²C器件自身从机地址信息，“GC”位可使能通用寻址的识别。

26.31 I²C 数据寄存器(I2CDAT)

I2CDAT								
I²C 数据寄存器								
地址空间： S：0D5h								
位	7	6	5	4	3	2	1	0
符号	I2CDAT[7:0]							
类型	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	0	0	0	0	0	0

位	符号	描述
7:0	I2CDAT[7:0]	I2CDAT：该 8 位寄存器用来存放要发送的一个字节或刚接收到的一个字节的串行数据。I2CDAT 的数据总是从右向左移动；最先发送的是最高位（位 7），接收完一个字节后，接收到的数据的第一位放置到 I2CDAT 的最高位。数据移出时，总线上的数据同时移入；I2CDAT 通常保存的是总线上的最后一个字节。因此，在仲裁丢失时，主发送器到从接收器的转变和 I2CDAT 中数据的更新同时进行。 该寄存器没有映射寄存器或者双缓存寄存器，因此用户应当在 I²C 中断发生后，在中断服务程序中读取 I2CDAT。

Figure 26-7 I²C数据寄存器

26.32 I²C 地址寄存器(I2CADR)

I2CADR								
I²C 地址寄存器								
地址空间： S：0D6h								
位	7	6	5	4	3	2	1	0
符号	I2CADR[7:1]							GC
类型	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	0	0	0	0	0	0

位	符号	描述
7:1	I2CDAT[7:1]	I²C 自身从地址（7 位） 包含从机模式下 I²C 接口操作的 7 位从地址，不在主机模式下使用。
0	GC	最低位 GC 决定从机是否响应通用调用地址 1：从机响应通用调用地址，通用调用地址（0x00）被识别。 0：从机不响应通用调用地址，通用调用地址（0x00）不被识别。

Figure 26-8 I²C地址寄存器

26.33 I²C 状态寄存器(I2CSTAT)

I2CSTAT								
I²C 状态寄存器								
地址空间： S：0D7h								
位	7	6	5	4	3	2	1	0
符号	I2C_Status_Code[4:0]					--	--	--
类型	R	R	R	R	R	R	R	R
复位值	1	1	1	1	1	0	0	0

位	符号	描述
7:3	I2C_Status_Code[4:0]	作为一个整字节时，状态寄存器代表一个状态代码。共有 26 种可能存在的状态代码。当代码为 0xF8 时，无可用的相关信息，SI 位不会置位。所有其他 25 种状态代码都对应一个已定义的 I²C 状态。当进入其中一种状态时，SI 位将置位。
2:0	--	保留位 最低 3 位总是为 0。

Figure 26-9 I²C状态寄存器

26.34 I2C 控制寄存器(I2CCON)

I2CCON								
I²C 控制寄存器								
地址空间： S：0D4h								
位	7	6	5	4	3	2	1	0
符号	CR2	ENS1	STA	STO	SI	AA	CR1	CR0
类型	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	0	0	0	0	0	0

位	符号	描述
7	CR2	波特率选择位 2
6	ENS1	I²C 接口使能。 当 ENS1 置“1”时，使能 I²C 接口。 当 ENS1 清零时，I²C 接口功能被禁止。 当 ENS1 为 0 时，SDA 和 SCL 输入信号被忽略，I²C 模块在“不可寻址的”从机状态中，且 STO 位被强制为“0”。 ENS1 不应用于暂时释放 I²C 总线，当 ENS1 复位时，I²C 总线状态丢失。应使用 AA 标志位代替。
5	STA	起始标志位。 当 STA=1 时，I²C 接口进入主模式并发送一个起始条件，如果已经处于主模式，则发送一个重复起始条件。 当 STA=1 并且 I²C 接口还没进入主模式时，I²C 接口进入主模式，检测总线并在总线空闲时产生一个起始条件。如果总线忙，则等待一个停止条件（释放总线）并在延迟半个内部时钟发生器周期后发送一个起始条件。当 I²C 接口已经处于主模式中并发送或接收了数据时，I²C 接口会发送一个重复的起始条件。STA 可在任何时候置位，当 I²C 接口处于可寻址的从模式时，STA 也可以置位。 当 STA=0 时，不会产生起始或重复起始条件。 当 STA 和 STO 都置位时，如果 I²C 接口处于主模式，I²C 接口将向总线发送一个停止条件，然后发送一个起始条件。如果 I²C 接口处于从模式，则产生一个内部停止条件，但不发送到总线上。
4	STO	停止标志位。 在主模式中，当 STO 为 1 时，会使 I²C 接口发送一个停止条件或在从模式中从错误状态中恢复。当总线检测到停止条件时，STO 自动清零。 在从模式中，置位 STO 位可从错误状态中恢复。这种情况下不向总线发送停止条件。硬件的表现就好像是接收到一个停止条件并切换到不可寻址的从接收模式。 STO 标志由硬件自动清零。
3	SI	I²C 中断标志。 当 I²C 状态改变时 SI 置位。但进入状态 F8 不会使 SI 置位。 当 SI 置位时，SCL 线上的低电平串行时钟周期会被扩展，且串行传输被中止。当 SCL 为高时，SI 标志的状态不受影响。

		SI 必须通过软件复位。写“0”来复位 SI，写“1”不起作用。
2	AA	AA：应答标志位。 当 AA 置位时，在 SCL 线的应答时钟脉冲内，出现下面的任意条件之一将产生一个应答信号（SDA 线为低电平）： (1) 接收到自身从地址寄存器中的地址； (2) 当 I2CADR 中的通用调用位（GC）置位时，接收到通用调用地址； (3) 当 I²C 接口处于可寻址的从接收模式时，接收到一个数据字节。 当 AA 为零时，SCL 线的应答时钟脉冲内出现下列情况将返回一个非应答信号（SDA 线为高电平）： (1) 当 I²C 接口处于主接收模式时，接收到一个数据字节； (2) 当 I²C 接口处于可寻址的从接收模式时，接收到一个数据字节。
1	CR1	波特率选择位 1
0	CR0	波特率选择位 0

Figure 26-10 I²C控制寄存器

26.35 I²C 定时器启动寄存器(I2C_TMR)

I2C_TMR								
I²C 定时器启动寄存器								
地址空间： S：0D2h								
位	7	6	5	4	3	2	1	0
符号	--	--	--	--	--	--	--	I2C_TMR
类型	--	--	--	--	--	--	--	R/W
复位值	--	--	--	--	--	--	--	0

位	符号	描述
7:1	--	保留位
0	I2C_TMR	启动 I²C 可变波特率计时器使能信号 1：I²C 专用计时器开始计数 0：I²C 专用计时器停止计数

Figure 26-19 I²C定时器启动寄存器

26.36 I²C 定时器寄存器(I2C_TM)

I2C_TM								
I²C 定时器寄存器								
地址空间： S：0D3h								
位	7	6	5	4	3	2	1	0
符号	I2C_TM							
类型	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	0	0	0	0	0	0

位	符号	描述
7:0	I2C_TM	I²C 可变波特率计时器初值

Figure 26-11 I²C 定时器寄存器

27 ISO7816 主控器

27.1 简介

ISO7816 为异步半双工串行通讯端口，符合 ISO/IEC 7816-3 标准，功能和特性如下：

- 支持 T=0 和 T=1 的传输模式；
- 收发自动转换；
- 支持重试次数设置；
- 支持额外的基本时间单位(ETU,elementary time unit)设置；
- 支持 ETU 计数器计数；
- 支持硬件 LRC/CRC(纵向冗余校验/循环冗余校验)计算；
- 起始位采用 16 次采样判决算法；
- 每一个 Bit 数据三次采样并遵从择多判决算法；
- 支持 ISO7816 接口软复位；
- 支持接收数据未取走情况下，被新数据覆盖（overrun）标志
- 支持 ISO7816 接口状态（忙 busy/空闲 idle）标志

ISO7816 接口有以下中断源：

- 发送/接收数据完成（包括成功和失败）；
- ETU 计数器 中断

27.2 ETU 计数器的使用

ISO7816 提供 ETU 计数器的功能，软件可以根据需要设置 ETU 计数器，产生中断后做相应的处理。

使用 ETU 计数器的情况有以下几种：

1) 计算字符超时等待时间(CWT)

2) 计算块保护等待时间(BGT)

3) 计算额外的保护时间（额外 ETU）

- CWT 应用:软件按需要设置 ETU 计数器为 CWT 的值,如果接收两个字节数据之间间隔超过 CWT，则会产生 ETU 计数器中断，软件可以在 ETU 计数器中断服务函数中做相应的超时处理。
- BGT 应用：T=1 模式下，软件按需要设置 ETU 计数器为 BGT 的值，在接收到一个包数据后，必须等 ETU 计数器中断产生后，才可以发送数据包，用以满足块保护时间的要求。
- 额外的保护时间应用：ISO7816 提供 0/1/2/4 四种额外 ETU 的可选配置，但如果有其他需要，可以通过设置 ETU 计数器并等待中断的方法，满足特殊的额外 ETU 需求。

27.3 ISO7816 通信速率设置

ISO7816 模块通信速率由 SBDRH/SBDRL 寄存器控制。该寄存器的值为一个 ETU 所对应的外部时钟周期数。具体寄存器设置与 7816 模块通信速率的对应关系如下表所示：

Fi/Di	{SBDRH[5:0],SBDRL[7:0]}	波特率 (3.57MHz)	波特率 (5MHz)
372	0x0174	9.6bps	13.4kbps
64	0x0040	57.6kbps	80.7kbps
32	0x001F	115.2kbps	161.3kbps
16	0x0010	230.4kbps	320.6kbps

Figure 27-1 ISO7816 通信速率设置

27.4 ISO7816 字符等待时间（CWT）

字符等待时间 CWT 表示 ISO7816 相同传输方向上连续两个字符间起始沿的最大延迟。

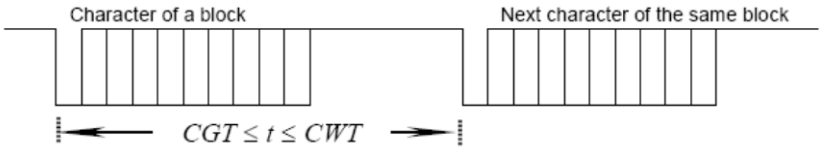


Figure 27-2 7816 字符等待时间

在 T = 0 模式下，字符等待时间 $CWT = WI * 960 * \frac{F}{f}$ (WI=0~255，默认为 10)，该默认值为 9600 ETU；

在 T =1 模式下，字符等待时间 $CWT = 11etu + 2^{CWI}$ (CWI=0~15，默认为 13)，该默认值为 8203 ETU；

27.5 ISO7816 块保护时间（BGT）

在 T=1 模式下，块保护时间 BGT 表示，两个反向字符第一个沿之间的最小延迟。如下图所示：

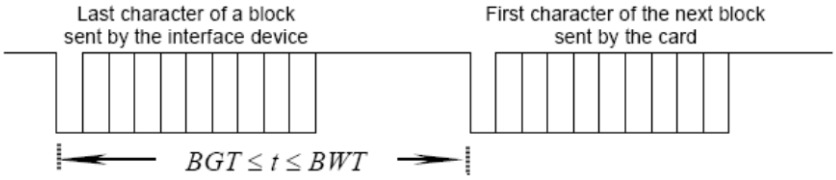


Figure 27-3 ISO7816 块保护时间

27.6 起始位采样

起始位采样采用 16 次采样，采样图形如下：

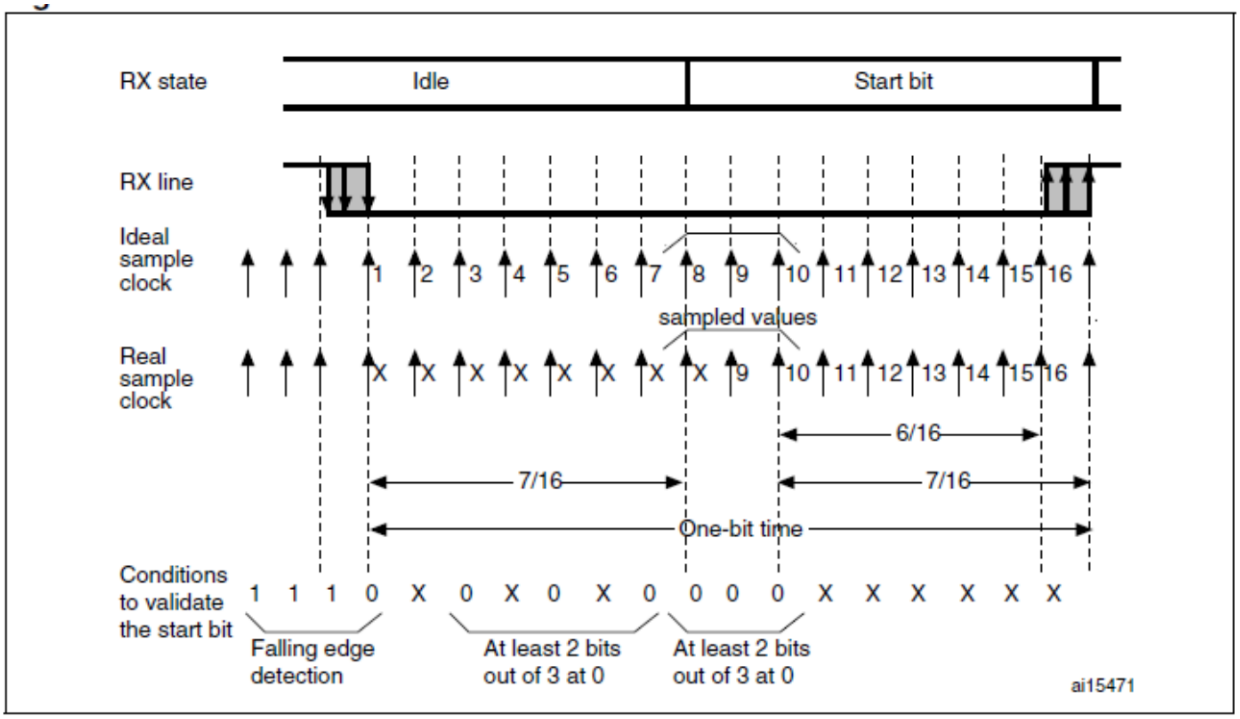


Figure 27-4 采样判决方式

采样判决方式说明如下：

- 两组采样点中，全部6个采样点均为0（第3、5、7采样点处为0，第8、9、10采样点处为0），则起始位采样有效，不置起SCI_STATUS.RX_NE标志。
- 对两组采样点中，两组里的3个采样点至少有2个为0（第3、5、7采样点和第8、9、10采样点），起始位采样有效，并置起SCI_STATUS.RX_NE标志（第一种6个采样点全为0的情况除外）。
- 如果以上采样条件不满足，退出起始位采样，并使7816接口回到等待状态。

27.7 软件操作流程

27.8 初始化配置

1） 清除 SCI_STATUS 寄存器里的中断标志；

2） 设置 SCI_MODE 寄存器；

3） 配置中断优先级，通过 AIE.7816E 使能 7816 中断；

4） 设置 SCI_CON 寄存器，使能 ISO7816 模块，及配置其他功能；

5） 配置 SBDRH/SBDRL 寄存器；

6） 配置 ETU_CNTH/ETU_CNTL 寄存器；

27.9 发送（中断方式）

1） 初始化配置；

2） 将待发送数据写入 SCI_DATA 寄存器中；

3） 当数据发送成功或失败（超过重试设置），则会产生中断，则软件应首先清除中断标志，并读取中断标志寄存器确认中断清除成功；

4） 如继续发送，则重复步骤 2-3；

5） 数据发送过程结束后，硬件自动把 7816 模块设成硬件接收模式，不需要软件设置；

27.10 发送（轮询方式）

1） 初始化配置；

2） 将待发送数据写入 SCI_DATA 寄存器中；

3） 软件查询 SCI_STATUS 寄存器标志位，若数据传输成功则硬件会置 TX_FIN 标志，若失败则置 TX_ERR 标志；程序查询到两标志之一置起后，需把相应标志清 0，并读取 SCI_STATUS 标志寄存器确认清除成功；

4） 如继续发送，重复步骤 2-3；

5） 数据发送过程结束后，硬件自动把 7816 模块设成硬件接收模式，不需要软件设置；

27.11 接收（中断方式，无 FIFO）

1) 初始化配置；
2) 当接收了一个字节数据成功或失败之后，会产生中断请求，若数据接收成功则硬件会置 SCI_STATUS.RX_FIN 标志，若失败则置 SCI_STATUS.RX_ERR 标志；软件应首先清除中断标志，并读取中断标志寄存器确认中断清除成功；
3) 若接收成功，则软件应从 SCI_DATA 中读取收到的数据；若接收失败，则进行出错处理；
4) 继续接收重复步骤 2-3；

27.12 接收（轮询方式，无 FIFO）

1) 初始化配置；
2) 软件查询 SCI_STATUS 标志，当接收了一个字节数据成功或失败之后，会产生中断请求，若数据接收成功则硬件会置 SCI_STATUS.RX_FIN 标志，若失败则置 SCI_STATUS.RX_ERR 标志；软件应首先清除中断标志，并读取中断标志寄存器确认中断清除成功；
3) 若接收成功，则软件应从 SCI_DATA 中读取收到的数据；若接收失败，则进行出错处理；
4) 继续接收重复步骤 2-3；

27.13 寄存器定义

ISO7816 寄存器共 12 个，列表如下：

寄存器名称	位宽	读写	地址	寄存器描述
SCI_CON	8	R/W	0x10180’h	ISO7816 控制寄存器
SCI_MODE	8	R/W	0x10182’h	ISO7816 模式配置寄存器
SCI_STATUS	8	R/W	0x10184’h	ISO7816 状态寄存器
SCI_DATA	8	R/W	0x1018A’h	ISO7816 收发数据寄存器
EDC_DATA_H	8	R/W	0x1018C’h	ISO7816 校验数据寄存器 1
EDC_DATA_L	8	R/W	0x1018E’h	ISO7816 校验数据寄存器 2
SBDRH	8	R/W	0x10190’h	ISO7816 波特率配置寄存器 1
SBDRL	8	R/W	0x10192’h	ISO7816 波特率配置寄存器 2
ETU_CNTH	8	R/W	0x10194’h	ISO7816 ETU 计数寄存器 1
ETU_CNTL	8	R/W	0x10196’h	ISO7816 ETU 计数寄存器 2
SCI_CLK_RST	8	R/W	0x10198’h	ISO7816 时钟复位寄存器
SCI_INT_EN	8	R/W	0x1019A’h	ISO7816 主机中断使能寄存器

Figure 27-5 ISO7816 寄存器组

27.14 ISO7816 控制寄存器(SCI_CON)

SCI_CON								
ISO7816 控制寄存器								
地址空间： 0x10180’h								
位	7	6	5	4	3	2	1	0
符号	TX_FIFO_CLR	RX_FIFO_CLR	ETU_CNT_EN	EDC_EN	RETRY_EN	PARITY_CHECK_EN	F1_EN	SCI_EN
类型	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	1	1	0	0	0	1

位	符号	描述
7	TX_FIFO_CLR	软件置位后，清空 ISO7816 发送缓存中的数据，发送 FIFO 中的所有状态标志位恢复默认值，之后 TX_FIFO_CLR 位自动清零；
6	RX_FIFO_CLR	软件置位后，清空 ISO7816 接收缓存中的数据，接收 FIFO 中的所有状态标志位恢复默认值，之后 RX_FIFO_CLR 位自动清零；
5	ETU_CNT_EN	ETU 计数器使能信号 1：ETU 计数器功能使能； 0：ETU 计数器功能关闭； 注意事项：在 ETU 功能打开后，以收发数据的下降沿为计时开始点，以 ETU 为单位计数，若达到设置的 ETU 计数值，则产生标志位。该功能用以辅助 7816 协议的超时处理，可以代替定时器计时。
4	EDC_EN	1：EDC(error detection code)功能有效； 0：EDC(error detection code)功能无效； 注意事项：仅在 T=1 模式下有效。在该功能打开后，在接收到校验正确的数据，或者发送数据时，硬件会自动计算 EDC 的值并保存在 EDC_DATA_H/EDC_DATA_L 寄存器中。
3	RETRY_EN	自动重发功能使能 1：自动重收发功能有效； 0：自动重收发功能无效；
2	PARITY_CHECK_EN	校验检查使能 1：校验位检查功能有效； 0：校验位检查功能无效；
1	F1_EN	ISO7816 强制输出高电平功能控制信号 1：强制输出高电平功能有效； 0：强制输出高电平功能无效。 注意事项：打开此功能后，接收数据时，在奇偶校验错时，请求重发的一个 ETU 的低电平之前和之后，ISO7816 硬件强制输出一个时钟周期的高电平；发送数据时，在起始位开始前和数据位之后，ISO7816 硬件强制输出一个时钟周期的高电平。在该功能不打开的情况下，IO PAD 在输入状态时，只能靠上拉保持高电平，因此从电平转换的上升时间或下降时间有可能超出规范（具体数据依赖于系统设计方案）；打开强制输出高电平功能，可以在电平转换时先将 IO PAD 电平强制

		拉高，之后切换为上拉控制，这样可以改善电平转换时间。
0	SCI_EN	ISO7816 模块使能控制信号； 1：ISO7816 硬件使能； 0：ISO7816 硬件关闭。 注意事项：SCI_EN 清零后，所有状态位恢复默认值。

Figure 27-6 ISO7816控制寄存器

注意事项：

1. 如出现 FIFO 异常，需要使用 TX/RX_FIFO_CLR 功能清空 FIFO，则软件在置位跟接收或发送对应的 TX/RX_FIFO_CLR 控制位后，需要查询等待该位自动清零后，FIFO 方可正常使用。

2.

RETRY_EN	PARITY_CHECK_EN	功能
0	0	不做校验检查，不重试，不会产生错误标志
0	1	做校验检查，不重试，如校验出错则产生错误标志
1	0	-
1	1	做校验检查，出错重试，如重试次数超出设置，则产生错误标志

- 在 PARITY_CHECK_EN=1 且 RETRY_EN=1 时，在接收数据时，若接收到奇偶校验错误的字节（且未超过重试次数设置），将自动在 10.5-11.5 ETU 处拉低 IO 一个 ETU 的时间，用来请求重发，且重试次数加 1；若超过重试次数设置，则硬件不再请求重发，并产生接收错误标志/中断。
- 在 PARITY_CHECK_EN=1 且 RETRY_EN=1 时，在发送数据时，若在 11ETU 处采样到对方的重发请求（且未超过重试次数设置），则硬件自动重发之前发送的最后一个字节数据，且重试次数加 1；若超过重试次数设置，则硬件不再重发，并产生发送错误标志/中断。
- 在 PARITY_CHECK_EN=1 且 RETRY_EN=0 时，在接收数据时，若接收到奇偶校验错误的字节，硬件不请求重发，并产生接收错误标志/中断。
- 在 PARITY_CHECK_EN=1 且 RETRY_EN=0 时，在发送数据时，若在 11ETU 处采样到对方的重发请求，硬件不重发，并产生发送错误标志/中断。

- 在 PARITY_CHECK_EN=0 且 RETRY_EN=0 时，在接收数据时，不校验接收的数据，硬件不请求重发，不产生接收错误标志/中断。
- 在 PARITY_CHECK_EN=0 且 RETRY_EN=0 时，在发送数据时，若在 11ETU 处采样到对方的重发请求，硬件不重发，不产生发送错误标志/中断。

27.15 ISO7816 模式配置寄存器(SCI_MODE)

SCI_MODE								
ISO7816 模式配置寄存器								
地址空间： 0x10182’h								
位	7	6	5	4	3	2	1	0
符号	RETRY_TIMES			EDC_MODE	--	CDCVT	PARITY	T_MODE
类型	R/W	R/W	R/W	R/W	--	R/W	R/W	R/W
复位值	0	0	0	0	--	0	0	0

位	符号	描述
7：5	RETRY_TIMES[2：0]	自动重收发次数设定，在 PARITY_CHECK_EN=1 且 RETRY_EN=1 时有效 000：允许无限次自动重收发，出错标志不会置位； 001：允许一次自动重收发，如仍有奇偶校验错则停止收发，出错标志被置位； 010：允许二次自动重收发，如仍有奇偶校验错则停止收发，出错标志被置位； …… 111：允许七次自动重收发，如仍有奇偶校验错则停止收发，出错标志被置位；
4	EDC_MODE	EDC 校验模式: 0：LRC 校验； 1：CRC 校验。 注意事项：仅 T=1 模式下有效。LRC 校验为 1 字节，存在 EDC_DATA_L 寄存器中；CRC 校验为 2 字节，高字节存在 EDC_DATA_H 中，低字节存在 EDC_DATA_L 中。 CRC16 校验算法参照 ISO/IEC 13239，算法多项式为 x^16+x^12+x^5+1，初始值为全 FFFF，输出取反。
3	--	保留位
2	CDCVT	0：正向传输； 1：反向传输。
1	PARITY	奇偶校验 0：偶校验； 1：奇校验。
0	T_MODE	传输模式 0：T=0 模式； 1：T=1 模式。 注意事项：T=1 模式下，PARITY_CHECK 和 RETRY 功能不可用，收发数据字符间隔默认为 11ETU（T=0 为 12ETU）。

Figure 27-7 ISO7816模式配置寄存器

27.16 ISO7816 状态寄存器(SCI_STATUS)

SCI_STATUS								
ISO7816 状态寄存器								
地址空间： 0x10184’h								
位	7	6	5	4	3	2	1	0
符号	RX_NE	RX_OVR	SCI_STAT	ETU_CNT_FIN	TX_ERR	TX_FIN	RX_ERR	RX_FIN
类型	R	R	R	R/W	R/W	R/W	R/W	R/W
复位值	0	0	0	0	0	0	0	0

位	符号	描述
7	RX_NE	接收起始位噪声标志: 接收数据开始后，起始位采样有效，但 6 个采样点并非全部为 0，则会置位，可做查询使用。在下一次接收数据的起始位后，如果起始位无效，或者 6 个采样点全部为 0（无噪声），则该标志自动清零。
6	RX_OVR	接收数据时，若没有及时取走 buffer 中的数据，在正确接到新的数据时，在接收起始位下降沿开始 11.5 个 ETU 时自动置位，表明有数据被覆盖；置位 RX_FIFO_CLR 可将此位清零。 0：ISO7816 接收 buffer 没有被覆盖； 1：ISO7816 接收 buffer 被覆盖了。
5	SCI_STAT	ISO7816 工作状态： 1：忙(busy) 0：空闲(idle) 在接收数据时，从接收起始位下降沿 12 个 ETU 内一直为 1，12 个 ETU 后自动清零，从接收新的字节开始，再次被自动置位。 在发送数据时，从发送起始位下降沿 12 个 ETU 内一直为 1，12 个 ETU 后自动清零，直到发送新的字节开始，再次被自动置位。
4	ETU_CNT_FIN	ETU 计数器中断标志： 以收发数据的下降沿为计时开始点，以 ETU 为单位计数，若达到设置的 ETU 计数值，则置位该标志位。置位后如果有新的数据收发，则标志位被清零并重新开始计数；也可以由软件清零，软件写 1 无效。
3	TX_ERR	ISO7816 数据发送失败的中断标志位： T=0 模式下，因奇偶校验错误，发送重试次数超过设置的次数，则此标志在最后一次发送起始位下降沿后 11.5 ETU 时被置位，须由软件清零，软件写 1 无效。 T=1 模式下，该标志位无效。
2	TX_FIN	ISO7816 数据发送成功的中断标志位： 在成功发送一个字节数据，起始位下降沿后 11.5 ETU（T=0）/10.5 ETU（T=1）时被置位，须由软件清零，软件写 1 无效。

1	RX_ERR	ISO7816 数据接收失败的中断标志位： T=0 模式下，因奇偶校验错误，接收重试次数超过设置的次数，则此标志在最后一次接收起始位下降沿后 11.5 ETU 时被置位，须由软件清零，软件写 1 无效。 T=1 模式下，该标志位无效
0	RX_FIN	ISO7816 数据接收成功的中断标志位： 在接收成功一个字节数据，起始位下降沿后 11.5 ETU（T=0）或 10.5 ETU（T=1）时被置位，须由软件清零，软件写 1 无效。

Figure 27-8 ISO7816状态寄存器

27.17 ISO7816 收发数据寄存器(SCI_DATA)

SCI_DATA								
ISO7816 收发数据寄存器								
地址空间： 0x1018A’h								
位	7	6	5	4	3	2	1	0
符号	D7	D6	D5	D4	D3	D2	D1	D0
类型	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	0	0	0	0	0	0

位	符号	描述
7：0	SCI_DATA[7:0]	数据缓存寄存器； 它存放待发送的数据字节，以及接收到的数据；二者地址映射在一起。 读取：已接收到的数据字节； 写入：待发送的数据字节。

Figure 27-9 ISO7816收发数据寄存器

27.18 ISO7816 校验数据寄存器 1(EDC_DATA_H)

EDC_DATA_H								
ISO7816 校验数据寄存器 1								
地址空间： 0x1018C’h								
位	7	6	5	4	3	2	1	0
符号	D7	D6	D5	D4	D3	D2	D1	D0
类型	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	0	0	0	0	0	0

位	符号	描述
7：0	EDC_DATA_H [7:0]	校验数据高字节: T=1 模式下，存储接收到或待发送的高字节 CRC 校验值。

Figure 27-10 ISO7816校验数据寄存器1

27.19 ISO7816 校验数据寄存器 2(EDC_DATA_L)

EDC_DATA_L								
ISO7816 校验数据寄存器 2								
地址空间： 0x1018E’h								
位	7	6	5	4	3	2	1	0
符号	D7	D6	D5	D4	D3	D2	D1	D0
类型	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	0	0	0	0	0	0

位	符号	描述
7：0	EDC_DATA_L [7:0]	校验数据低字节: T=1 模式下，存储接收到或待发送的 LRC 或低字节 CRC 校验值。

Figure 27-4 ISO7816校验数据寄存器2

操作流程如下：

1. 发送时写入 SCI_DATA 的数据，硬件会自动进行计算，并将计算结果随时更新至 EDC_DATA_H/EDC_DATA_L 寄存器中。软件可以将整个块数据(除 EDC 字节外)全部写入 SCI_DATA 寄存器，再读取 EDC_DATA_H/EDC_DATA_L 寄存器，将运算结果写入 SCI_DATA 寄存器，此时 EDC_DATA_H/EDC_DATA_L 寄存器将自动恢复默认值 0；
2. 接收时所有接收到的数据，硬件会自动进行计算，并将计算结果随时更新至 EDC_DATA_H/EDC_DATA_L 寄存器中。如果所有数据均接收正确，在整个块数据接收完成后，EDC_DATA_H/EDC_DATA_L 寄存器最终计算结果应该为 0。软件可以在完成一个块数据接收后查询 EDC_DATA_H/EDC_DATA_L 来校验数据，也可以在接收过程中查询 EDC_DATA_H/EDC_DATA_L 的值并与软件计算结果进行对比。

27.20 波特率配置寄存器 1(SBDRH)

SBDRH								
ISO7816 波特率配置寄存器 1								
地址空间： 0x10190’h								
位	7	6	5	4	3	2	1	0
符号	EXTRA_ETU_SEL		SBDR_13	SBDR_12	SBDR_11	SBDR_10	SBDR_9	SBDR_8
类型	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	0	0	0	0	0	1

位	符号	描述
7：6	EXTRA_ETU_SEL[1:0]	ISO7816 发送模式增加的额外 ETU: 无额外 ETU 情况下，T=0 默认为 12ETU，T=1 默认为 11ETU。 00：无额外 ETU； 01：增加一个额外 ETU； 10：增加二个额外 ETU；11：增加四个额外 ETU。
5:0	SBDR_13~SBDR_8	波特率配置寄存器高 6 位

Figure 27-5 ISO7816波特率配置寄存器高位

27.21 波特率配置寄存器 0(SBDRL)

SBDRL								
ISO7816 波特率配置寄存器 0								
地址空间： 0x10192’h								
位	7	6	5	4	3	2	1	0
符号	SBDR_7	SBDR_6	SBDR_5	SBDR_4	SBDR_3	SBDR_2	SBDR_1	SBDR_0
类型	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	1	1	1	0	1	0	0

位	符号	描述
7：0	SBDR_7~SBDR_0	波特率配置寄存器低 8 位 SBDR（{SBDRH[5：0],SBDRL[7:0]}）：ISO7816 波特率配置寄存器，SBDR = Fi/Di，默认值 0x174，表示 1 个 ETU 单位为 372 CLK。若 ISO7816 输入时钟不是 3.57MHz，需要根据频率比例计算相应的 SBDR 值。

Figure 27-6 ISO7816波特率配置寄存器低位

27.22 ETU 计数寄存器 1(ETU_CNTH)

ETU_CNTH								
ETU 计数寄存器高位								
地址空间： 0x10194’h								
位	7	6	5	4	3	2	1	0
符号	ETU_CNT_15	ETU_CNT_14	ETU_CNT_13	ETU_CNT_12	ETU_CNT_11	ETU_CNT_10	ETU_CNT_9	ETU_CNT_8
类型	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	1	0	0	1	0	1

位	符号	描述
7：0	ETU_CNTH[7：0]	ETU 计数寄存器高 8 位

Figure 27-7 ISO7816 ETU计数寄存器高位

27.23 ETU 计数寄存器 0(ETU_CNTL)

ETU_CNTL								
ETU 计数寄存器低位								
地址空间： 0x10196’h								
位	7	6	5	4	3	2	1	0
符号	ETU_CNT_7	ETU_CNT_6	ETU_CNT_5	ETU_CNT_4	ETU_CNT_3	ETU_CNT_2	ETU_CNT_1	ETU_CNT_0
类型	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	1	0	0	0	0	0	0	0

位	符号	描述
7：0	ETU_CNTL[7：0]	ETU 计数寄存器低 8 位 {ETU_CNTH[7:0], ETU_CNTL[7:0]}: ISO7816 ETU 计数寄存器，计算单位为 1 个 ETU 时间，默认值 0x2580，表示 9600 ETU。

Figure 27-8 ISO7816 ETU计数寄存器低位

27.24 ISO7816 时钟复位寄存器(SCI_CLK_RST)

SCI_CLK_RST								
7816 时钟复位寄存器								
地址空间： 0x10198’h								
位	7	6	5	4	3	2	1	0
符号	--	--	--	--	7816_CLK_SEL1	7816_CLK_SEL0	7816_CLK_EN	7816_RESET
类型	--	--	--	--	R/W	R/W	R/W	R/W
复位值	--	--	--	--	0	0	0	0

位	符号	描述
7:4	--	保留位
3：2	7816_CLK_SEL[1:0]	7816 主机输出时钟频率选择 00：系统时钟 01：1/2 系统时钟 10：1/4 系统时钟 11：1/8 系统时钟
1	7816_CLK_EN	7816 主机输出时钟驱动使能 1：7816 输出时钟使能 0：7816 输出时钟关闭
0	7816_RESET	7816 主机输出复位信号 1：7816 主机输出复位信号为高电平，从机工作 0：7816 主机输出复位信号为低电平，从机复位

Figure 27-9 ISO7816 时钟复位寄存器

27.25 ISO7816 主机中断使能寄存器(SCI_INT_EN)

SCI_INT_EN								
7816 主机中断使能寄存器								
地址空间： 0x1019A’h								
位	7	6	5	4	3	2	1	0
符号	--	--	--	ISO7816_etu_fin_int_en	ISO7816_tx_err_int_en	ISO7816_tx_fin_int_en	ISO7816_rx_err_int_en	ISO7816_rx_fin_int_en
类型	--	--	--	R/W	R/W	R/W	R/W	R/W
复位值	0	--	0	0	0	0	0	0

位	符号	描述
7：5	--	保留位
4	ISO7816_etu_fin_int_en	与 SCI_STATUS[4]相对应，ETU 计数器中断使能信号 1：MCU 接受 7816 ETU 计数器中断。 0：MCU 不接受 7816 ETU 计数器中断。
3	ISO7816_tx_err_int_en	与 SCI_STATUS[3]相对应，SCI7816 数据发送失败的中断使能信号。 1：MCU 接受 7816 数据发送失败的中断。 0：MCU 不接受 7816 数据发送失败的中断。
2	ISO7816_tx_fin_int_en	与 SCI_STATUS[2]相对应，SCI7816 数据发送成功的中断使能信号。 1：MCU 接受 7816 数据发送成功的中断。 0：MCU 不接受 7816 数据发送成功的中断。
1	ISO7816_rx_err_int_en	与 SCI_STATUS[1]相对应，SCI7816 数据接收失败的中断使能信号。 1：MCU 接受 7816 数据接收失败的中断。 0：MCU 不接受 7816 数据接收失败的中断。
0	ISO7816_rx_fin_int_en	与 SCI_STATUS[0]相对应，SCI7816 数据接收成功的中断使能信号。 1：MCU 接受 7816 数据接收成功的中断。 0：MCU 不接受 7816 数据接收成功的中断。

Figure 27-10 ISO7816主机中断使能寄存器

28 A/D 转换器

28.1 A/D 转换器简介

HC16Lxx 内部集成了一个高精度，高转换速度的 12 位逐次逼近型 A/D 转换器（SARADC）。该转换器具有如下特性：

- 转换速度可达 200KPS
- 单调无失码的 12 位转化
- 采样保持时间可设定
- 内建 2 个参考电压 （1.5V 和 2.5V）
- ADC 参考电压来源可设定（外部输入，内部 1.5V 或 2.5V）
- 支持 8 路采样源（7 个外部引脚以及一个内部温度传感器）
- 转换时钟可设定
- 支持单次采样和连续采样
- ADC 采样结果保存于 1 个 16 位或 2 个 8 位寄存器中，采样结果支持左对齐和右对齐
- 信号输入内建放大器，可采样外部微弱信号

28.2 SARADC 框架图

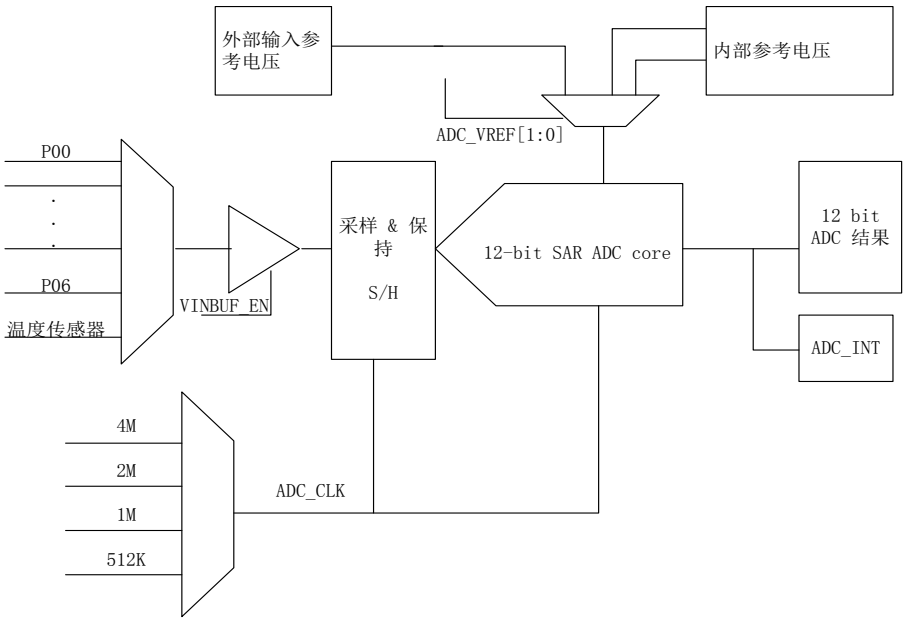


Figure 28-1 SARADC 框架图

28.3 A/D 转换器操作

28.4 单次采样模式

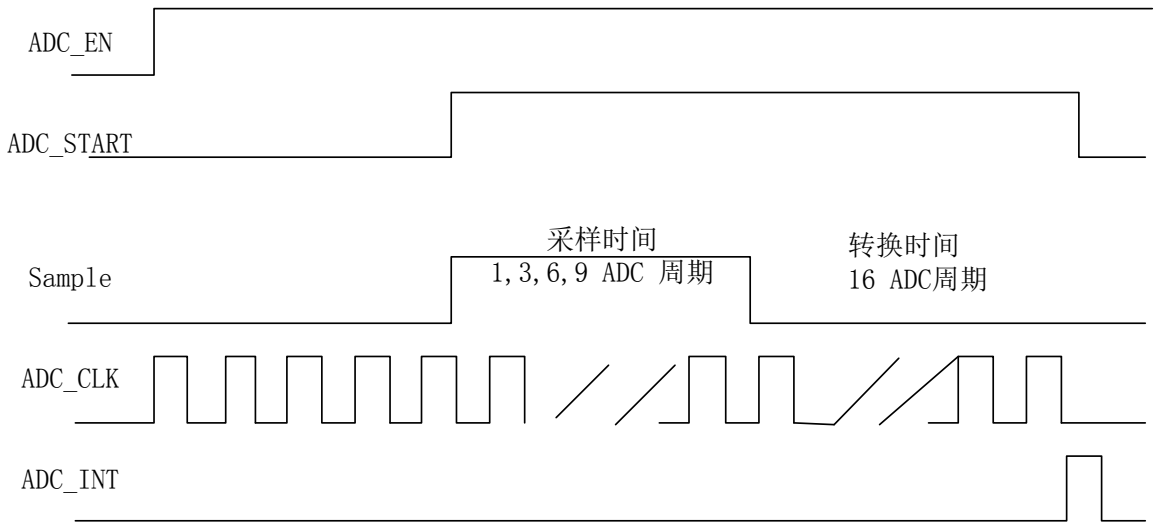


Figure 28-2 ADC单次采样过程

操作流程：

1) 使能 BGR.EN
2) 使能 ADC.EN
3) 等待 30us 的稳定时间，建立 ADC 工作点时间。
4) 在等待过程中配置 ADC 采样条件：如参考电压选择，采样输入，采样时钟，采样方式等。
5) 置位 ADC_START
6) 等待 17~26 个 ADC 周期，ADC 自动得到 12 位 ADC 采样结果，产生 ADC 中断。
7) 注意 ADC 结果寄存器中的数据会被后一次的转化结果覆盖，需要及时取出。
8) 单次采样中，ADC_START 被硬件清零。
9) 若需要提高采样精度，可进行过采样，重复执行(5)~(7)。
10) 关闭 ADC
11) 关闭 BGR

28.5 连续采样模式

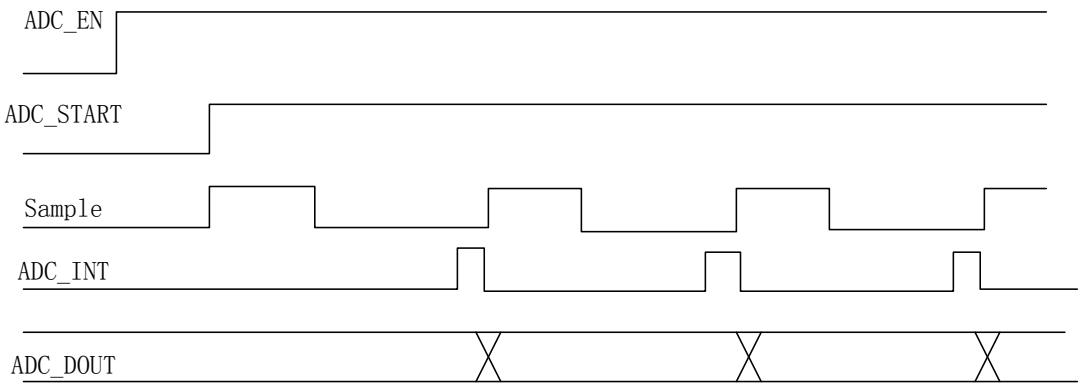


Figure 28-3 ADC 连续采样过程

操作流程

1) 使能 BGR.EN
2) 使能 ADC.EN
3) 等待 30us 的稳定时间，建立 ADC 工作点时间。
4) 在等待过程中配置 ADC 采样条件：如参考电压选择，采样输入，采样时钟，采样方式等。
5) 置位 ADC_START
6) 等待 17~26 个 ADC 周期，ADC 自动得到 12 位 ADC 采样结果，产生 ADC 中断。
7) 注意 ADC 结果寄存器中的数据会被后一次的转化结果覆盖，需要及时取出。
8) 多次采样中，ADC_START 不会被硬件清零。重复执行(6)~(7)。
9) 关闭 ADC
10) 关闭 BGR

28.6 温度传感器

HC16Lxx 温度传感器使用了硅晶体管的随温度变化的基本属性，产生一个与绝对温度成正比的电流。

传感器温度梯度，毫伏/摄氏度详见产品规范描述。通过选择内部基准电压，具有 12 位分辨率的 ADC 采样内部温度传感器（输入 VINBUF_EN 需要置“1”），温度可根据下面的公式来进行计算。

Temp = T25C +(ADC_Result_Temp - ADC_Result_25C)* (Vref/4096) / 梯度

例子 1：

A 温度下 ADC 测量值为 7E5，折合十进制为 2021；

25 度下 ADC 测量值为 76C，折合十进制为 1900；

温度梯度为 2.8mV/℃；

Vref = 2.5v；ADC 每个 LSB 最小精度为 Vref/4096 = 0.61mV

Temp = 25℃ + （2021-1900）* 0.61mV / 2.8 mV/℃

= 25℃ + 26.5℃ = 51.5 ℃

例子 2：

A 温度下 ADC 测量值为 72D，折合十进制为 1837；

25 度下 ADC 测量值为 76C，折合十进制为 1900；

温度梯度为 2.8mV/℃；

Vref = 2.5v；ADC 每个 LSB 最小精度为 Vref/4096 = 0.61mV

Temp = 25℃ + （1837-1900）* 0.61mV / 2.8 mV/℃

= 25℃ - 13.7℃ = 11.3 ℃

28.7 ADC 寄存器

28.8 ADC 控制寄存器 1

ADC1				
ADC 控制寄存器 1				
地址空间： S:09Bh				
位	7:6	5:3	2:1	0
符号	ADC_VREF[1:0]	ADC_CHL_SEL[2:0]	ADC_CLK_SEL[1:0]	ADC_EN
类型	R/W	R/W	R/W	R/W
复位值	00	000	00	0

位	符号	描述
7:6	ADC_VREF[1:0]	ADC 参考电压选择寄存器 1X： 选择外部参考电压 01：内部 1.5v 参考电压 00：内部 2.5v 参考电压
5:3	ADC_CHL_SEL[2:0]	ADC 采样通道选择 111：选择内部温度传感器 110：选择外部 P06 输入 101：选择外部 P05 输入 100：选择外部 P04 输入 011：选择外部 P03 输入 010：选择外部 P02 输入 001：选择外部 P01 输入 000：选择外部 P00 输入
2:1	ADC_CLK_SEL[1:0]	ADC 采样转换时钟选择 00：系统时钟 01：系统时钟 2 分频 10：系统时钟 4 分频 11：系统时钟 8 分频
0	ADC_EN	ADC 使能信号 1：启动 ADC 0：关闭 ADC

Figure 28-4 ADC控制寄存器1

28.9 ADC 控制寄存器 2

ADC2							
ADC 控制寄存器 2							
地址空间： S:09Ch							
位	7:6	5	4	3	2	1	0
符号	ADC_Sample_SEL [1:0]	VINBUF_EN	--	ADC_Mode	ADC_Align	--	ADC_START
类型	R/W	R/W	--	R/W	R/W	--	R/W
复位值	00	0	--	0	0	--	0

位	符号	描述
7:6	ADC_Sample_SEL [1:0]	ADC 采样时间选择 00：用 3 个 ADC 时钟进行采样 01：用 6 个 ADC 时钟进行采样 10：用 1 个 ADC 时钟进行采样 11：用 9 个 ADC 时钟进行采样
5	VINBUF_EN	ADC 输入信号放大器使能信号 0：输入信号放大器禁止，输入直接连接外部输入。 1：输入信号放大器使能，外部输入通过该放大器接入电路，用于采样外部微弱信号。 注意：若采样内部温度传感器，VINBUF_EN 应设置为“1”，使能内部信号放大器。
4	--	保留位
3	ADC_Mode	ADC 采样模式 1：单次采样 0：连续采样
2	ADC_Align	ADC 采样结果对齐 1：右对齐。结果为 xxxxMSB~LSB 0：左对齐。结果为 MSB~LSBxxxx
1	--	保留位
0	ADC_START	ADC 采样启动信号 1：启动 ADC 采样 0：停止 ADC 采样 在单次采样模式下，ADC_START 会在一次采样结束后被硬件清零。启动第二次采样，需对 ADC_START 软件置位。

Figure 28-5 ADC控制寄存器2

28.10 ADC 控制寄存器 3

ADC3					
ADC 控制寄存器 3					
地址空间： S:09Dh					
位	7	6	5:3	2:1	0
符号	VC2PT	-	ADC_Trim[2:0]	ADC_Current[1:0]	TSEN
类型	R/W	-	R/W	R/W	R/W
复位值	0	0	011	00	0

位	符号	描述
7	VC2PT	模拟电压比较器结果从端口输出 1：反向结果输出 0：正向结果输出
6	--	保留位
5:3	ADC_Trim[2:0]	ADC 内部 2.5v 参考电压微调，从校准寄存器可读取最优化的结果。 111：最高参考电压 ~ 000：最低参考电压
2:1	ADC_Current[1:0]	ADC 工作电流配置 11：最低 ADC 工作电流(推荐) ~ 00：最高 ADC 工作电流(默认)
0	TSEN	内部温度传感器启动 1：启动内部温度传感器 0：禁用内部温度传感器

Figure 28-6 ADC控制寄存器3

28.11 ADCH 采样结果高位寄存器

ADCH		
ADC 采样结果高位寄存器		
地址空间： S:09Eh		
位	7:0	
符号	ADCH[7:0]	
类型	R	
复位值	00000000	

位	符号	描述
7:0	ADCH[7:0]	ADC 采样结果高位寄存器

Figure 28-7 ADC采样结果高位寄存器

28.12 ADCL 采样结果低位寄存器

ADCL		
ADC 采样结果低位寄存器		
地址空间： S:09Fh		
位	7:0	
符号	ADCL[7:0]	
类型	R	
复位值	00000000	
位	符号	描述
7:0	ADCL[7:0]	ADC 采样结果低位寄存器 右对齐: {ADCH[7:0],ADCL[7:0]} = {4’h0, ADC[11:0]} 左对齐: {ADCH[7:0],ADCL[7:0]} = {ADC[11:0], 4’h0}

Figure 28-8 ADC采样结果低位寄存器

28.13 ADC 采样结果 12 位寄存器

ADC_Result		
ADC 采样结果 12 位寄存器		
地址空间： 0x:100E4		
符号	ADC_Result[15:0]	
类型	R	
复位值	16’h0000	
位	符号	描述
15:0	ADC_Result[15:0]	ADC 采样结果寄存器 右对齐: ADC_Result[15:0] = {4’h0, ADC[11:0]} 左对齐: ADC_Result[15:0] = {ADC[11:0], 4’h0}

Figure 28-9 ADC采样结果12位寄存器

28.14 BGR 控制寄存器

BGRC								
BGR 控制寄存器								
地址空间： S:09Ah								
位	7	6	5	4	3	2	1	0
符号	OPT2	OPT1	OPT0	BGR_Auto	--	--	--	BGR_EN
类型	R/W	R/W	R/W	R/W	--	--	--	R/W
复位值	1	0	1	0	--	--	--	0

位	符号	描述
7:5	OPT[2:0]	BGR 工作选项
4	BGR_Auto	BGR 低功耗自动关闭设置 1： BGR 在进入低功耗模式时，自动关闭；工作模式时自动开启。 0： BGR 开关由 BGR_EN 控制.
3:1	--	保留位
0	BGR_EN	BGR 使能位 1： BGR 使能 0： BGR 关闭

Figure 28-10 BGR寄存器

29 模拟电压比较器(VC)

29.1 模拟电压比较器简介

电压比较器的功能：比较两个输入模拟电压的大小(用输出电压的高或低电平，表示两个输入电压的大小关系)：

当”＋”输入端电压高于”－”输入端时，电压比较器输出为高电平；

当”＋”输入端电压低于”－”输入端时，电压比较器输出为低电平；

输入端可以是外部输入，也可以是内部 BGR 输出的参考电压

特征：

- 支持电压监测功能
- 支持外部 4 个输入以及内部 BGR 输出
- 可选中断产生来源，高电平和上升沿下降沿。
- 支持迟滞电路以增强抗干扰能力
- 支持响应时间以增强抗干扰能力
- 支持提供给定时器 0(Timer0)，定时器 1(Timer1)的捕获输入
- 支持低功耗模式下工作，电压比较器中断可以唤醒低功耗模式

29.2 模拟电压比较器框图

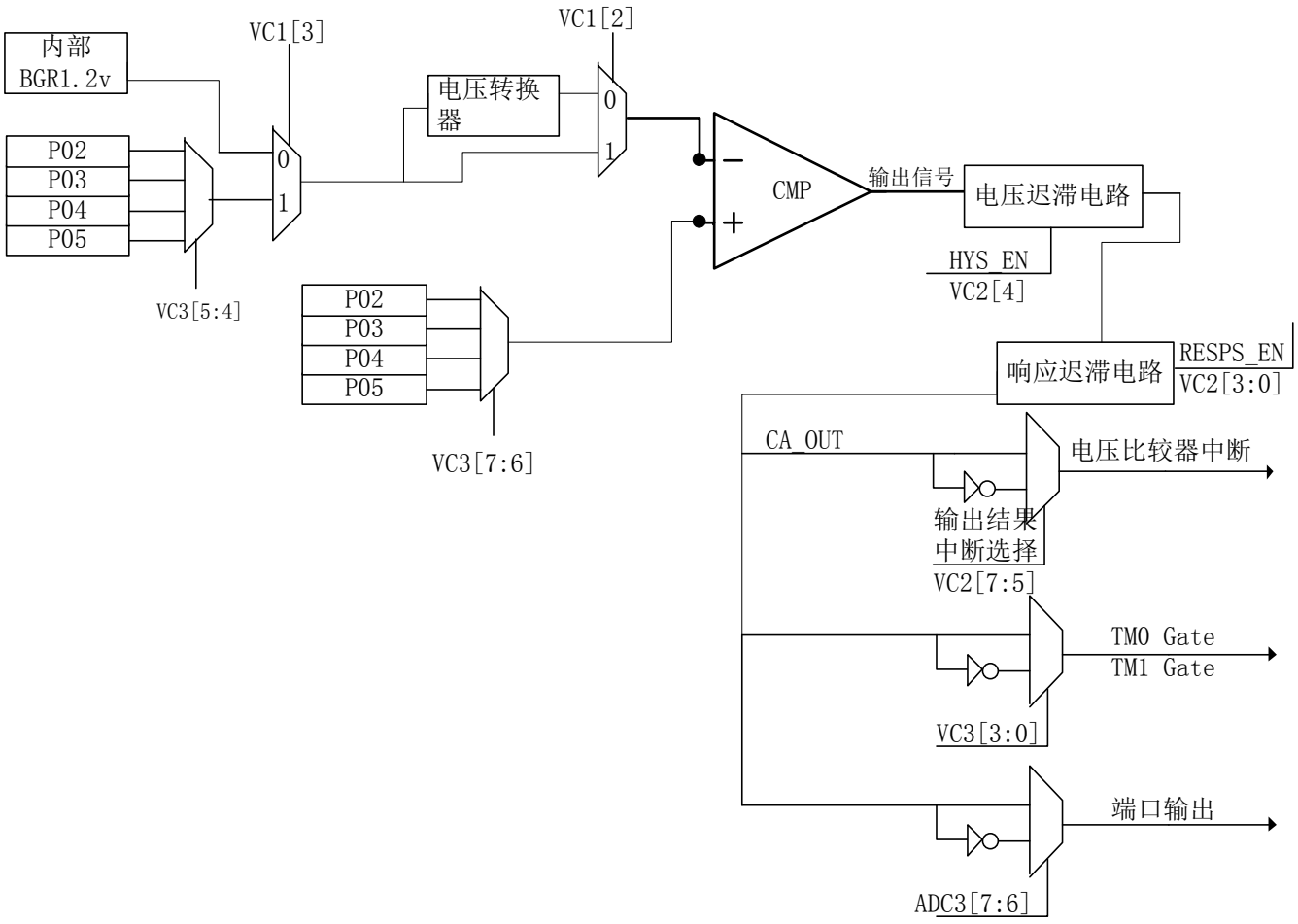


Figure 29-1 模拟电压比较器框图

29.3 模拟电压比较器操作

29.4 建立时间

当使用电压比较器来监测外部电压变化时，需要内部 BGR 输出的参考电压做为比较基准电压。BGR 的启动需要 30uS 的时间，电压比较器需要等待 30us 的稳定时间，建立电压比较器的工作点。

29.5 响应时间

当电压比较器的输入电压发生变换后，到输出结果产生翻转的这段时间被称为响应时间，小于响应时间的电压变化将被忽略。电压比较器固有响应时间小于 500ns，也可通过配置寄存器 VC2[0]使能滤波电路。当滤波电路开启后，通过配置寄存器 VC2[3:1]选择不同的滤波时间，以获得不同响应时间。

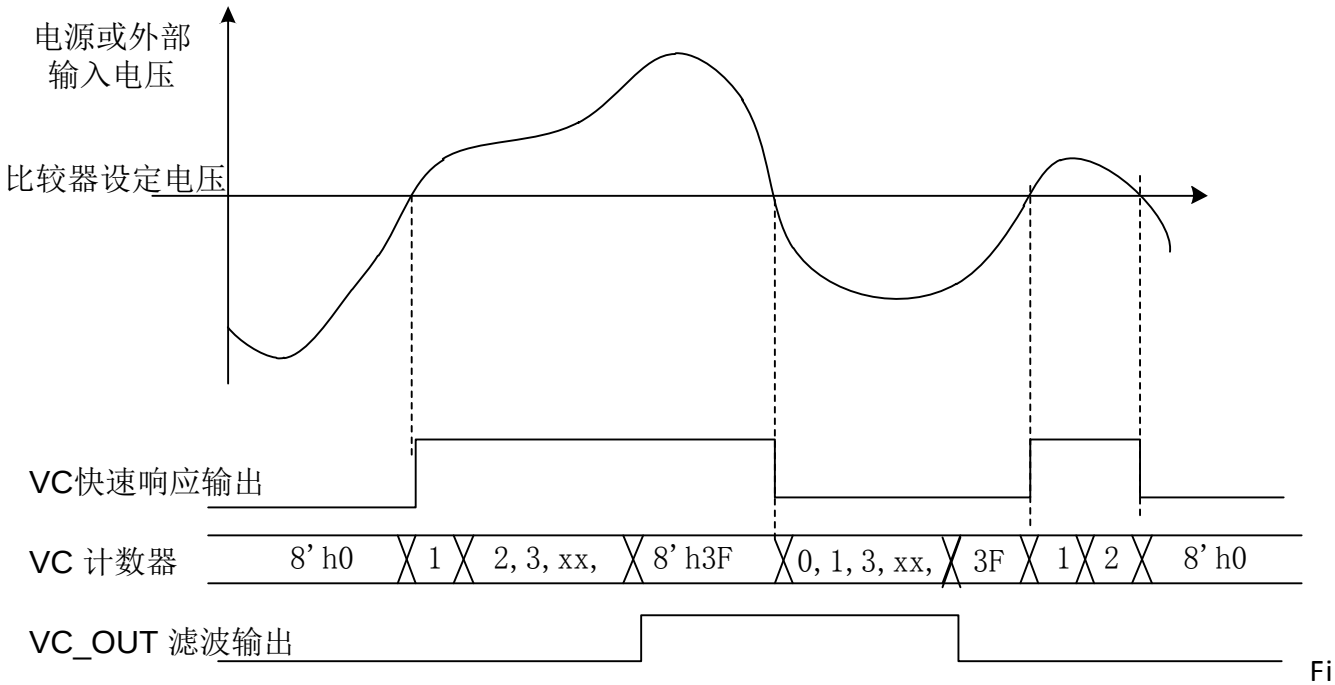


Figure 29-2 滤波响应时间

29.6 迟滞选择

通过配置寄存器 VC2[4]可选择关闭或开启迟滞电路。若迟滞电路关闭，当正端输入电压高于负端输入电压，结果则为“1”，当正端输入电压低于负端输入电压，结果则为“0”；若迟滞电路开启，当正端输入电压高于负端正迟滞电压时，结果为“1”，当正端输入电压低于负端负迟滞电压时，结果为“0”。

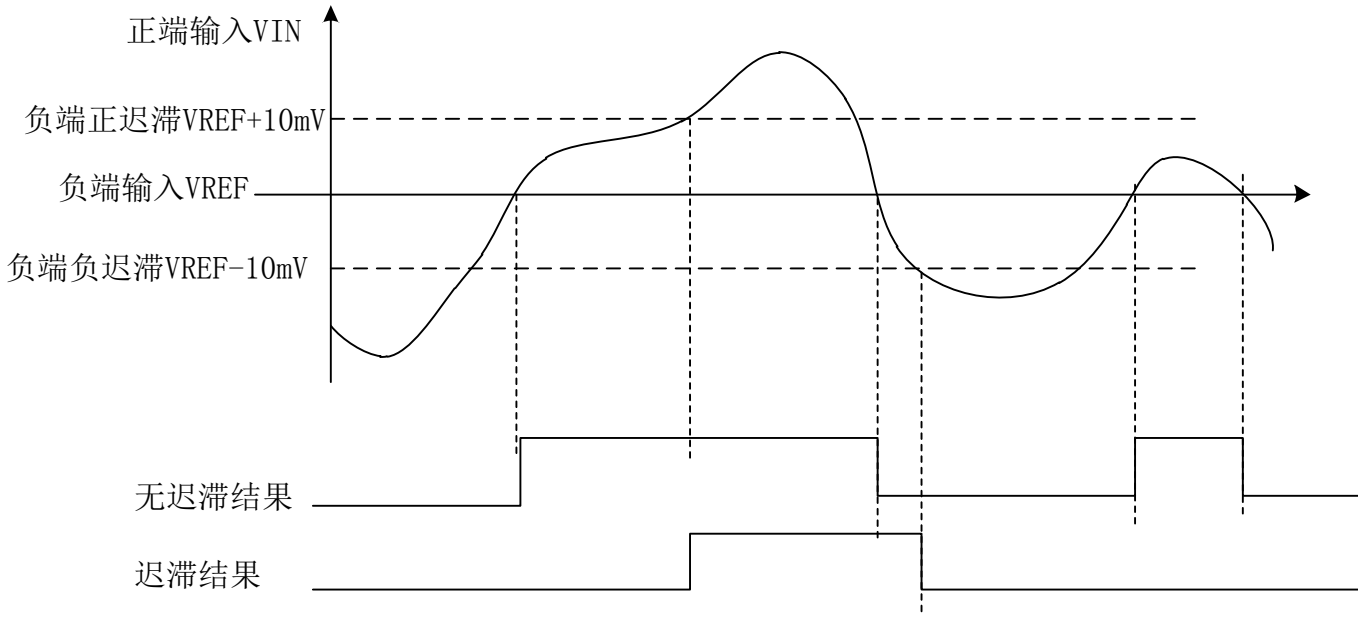


Figure 29-3 模拟比较器迟滞

29.7 模拟电压比较器寄存器

29.8 模拟电压比较器寄存器 1(VC1)

VC1				
模拟电压比较器寄存器 1				
地址空间： S:0BCh				
位	7:4	3:2	1	0
符号	VCVOL_SEL[3:0]	VC_VREF[1:0]	VC_OUT	VC_EN
类型	R/W	R/W	R	R/W
复位值	0000	00	0	0

位	符号	描述
7:4	VCVOL_SEL[3:0]	模拟比较器输入监测电压选择 0000: 2.83v +/-100mV 0001: 2.94v +/-100mV 0010: 3.05v +/-100mV 0011: 3.16v +/-100mV 0100: 3.27v +/-100mV 0101: 3.38v +/-100mV 0110: 3.49v +/-100mV 0111: 3.60v +/-100mV 1000: 3.71v +/-100mV 1001: 3.82v +/-100mV 1010: 3.93v +/-100mV 1011: 4.04v +/-100mV 1100: 4.15v +/-100mV 1101: 2.50v +/-100mV 1110: 2.61v +/-100mV 1111: 2.72v +/-100mV 注意：外部电压不得高于芯片工作电压 VCC+0.3v
3:2	VC_VREF[1:0]	模拟比较器基准（负端）电压选择 00：内部 BGR 经过电压转换器转换后的电压 01：内部 BGR（ 1.2V） 电压直接作为比较器基准电压 10：外部基准经过电压转换器转换后的电压 11：外部基准直接作为比较器基准电压 注意：在 10 模式下，外部基准电压为 1.2V 时，转换器转换后电压为 VC1[7:4]中选择的值，若外部基准不为 1.2V，则转换后电压值不能确定，不建议如此使用。
1	VC_OUT	模拟比较器输出结果
0	VC_EN	模拟比较器使能位 1： 模拟比较器使能 0： 模拟比较器关闭

Figure 29-4 VC1寄存器

29.9 模拟电压比较器寄存器 2(VC2)

VC2						
模拟电压比较器寄存器 2						
地址空间： S:0BDh						
位	7	6	5	4	3:1	0
符号	VCINT_ high	VCINT_ rising	VCINT_ falling	VC_HYS_ EN	VC_ response[3:1]	VC_Filter_EN
类型	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	0	0	000	0

位	符号	描述
7	VCINT_ high	电压比较器中断边沿检测/高电平检测选择 1：高电平检测产生中断。 0：边沿检测产生中断。
6	VCINT_rising	电压比较器中断上升沿选择 1：当电压比较器输出从低电平转到高电平，产生中断。 0：当电压比较器输出从低电平转到高电平，不产生中断
5	VCINT_falling	电压比较器中断下降沿选择 1：当电压比较器输出从高电平转到低电平，产生中断 0：当电压比较器输出从高电平转到低电平，不产生中断 注意：这 3 种中断可以随意设置，组合使用。 比如：置位 VCINT_Rising，置位 VCINT_Falling，那么电压比较器输出的上升沿和下降沿都可以产生中断
4	VC_HYS_EN	电压比较器迟滞开关 1：打开迟滞开关，大约 20mV 0：关闭迟滞开关 注意事项:若 VC1[2]=1’b1,无论 VC_HYS_EN 的值是多少，电压比较器迟滞关闭。
3:1	VC_response [3:1]	电压比较器滤波时间选择 000：滤波时间 16us 001：滤波时间 32us 010：滤波时间 64us 011：滤波时间 256us 100：滤波时间 1ms 101：滤波时间 4ms 110：滤波时间 16ms 111：滤波时间 64ms 注意：所有时间均为标准状况下的值
0	VC_Filter_EN	电压比较器滤波开关 1：滤波开关打开 0：滤波开关关闭

Figure 29-5 VC2寄存器

29.10 模拟电压比较器寄存器 3(VC3)

VC3						
模拟电压比较器寄存器 3						
地址空间： S:0CCh						
位	7:6	5:4	3	2	1	0
符号	VCIN_POS_SEL [1:0]	VCIN_NEG_SEL [1:0]	VC_TM1G	VCINV_TM1G	VC_TM0G	VCINV_TM0G
类型	R/W	R/W	R/W	R/W	R/W	R/W
复 位值	00	00	0	0	0	0

位	符号	描述
7：6	VCIN_POS_SEL	VCIN_POS_SEL：外部输入电压源选择 00：外部输入电压源为 P02 01：外部输入电压源为 P03 10：外部输入电压源为 P04 11：外部输入电压源为 P05
5：4	VCIN_NEG_SEL	VCIN_NEG_SEL：外部基准参考电压源选择 00：外部基准参考电压源为 P02 01：外部基准参考电压源为 P03 10：外部基准参考电压源为 P04 11：外部基准参考电压源为 P05 注意：当电压比较器的正端输入源和负端输入源都连接到同一个外部引脚时，电压比较器的输出结果是不保证的。
3	VC_TM1G	电压比较器输出到定时器 1 使能 1：电压比较器输出结果连接到定时器 1 的门控输入端 0：电压比较器输出结果不连接到定时器 1 的门控输入端
2	VCINV_TM1G	电压比较器取反结果输出到定时器 1 使能 1：电压比较器取反结果连接到定时器 1 的门控输入端 0：电压比较器结果直接连接到定时器 1 的门控输入端
1	VC_TM0G	电压比较器输出到定时器 0 使能 1：VC 输出结果连接到定时器 0 的门控输入端 0：VC 输出结果不连接到定时器 0 的门控输入端
0	VCINV_TM0G	电压比较器取反结果输出到定时器 0 使能 1：电压比较器取反结果连接到定时器 0 的门控输入端 0：电压比较器结果直接连接到定时器 0 的门控输入端

Figure 29-6 VC3寄存器

30 低电压检测器(LVD)

30.1 低电压检测器简介

低电压检测电路（LVD）用于检测 HC16Lxx 的工作电压以及外部 P06 输入电压。

特性：

- LVD 检测芯片工作电压，当芯片电压降低到某一个设定值时，产生中断或复位。
- LVD 检测芯片引脚输入电压，当输入电压降低到某一个设定值时，产生中断或复位。
- 可设定 16 阶检测电压。
- 当 LVD 检测到低电压时，触发复位或中断信号；当 LVD 检测到电压恢复后，释放复位或中断信号；

30.2 低电压检测器框架图

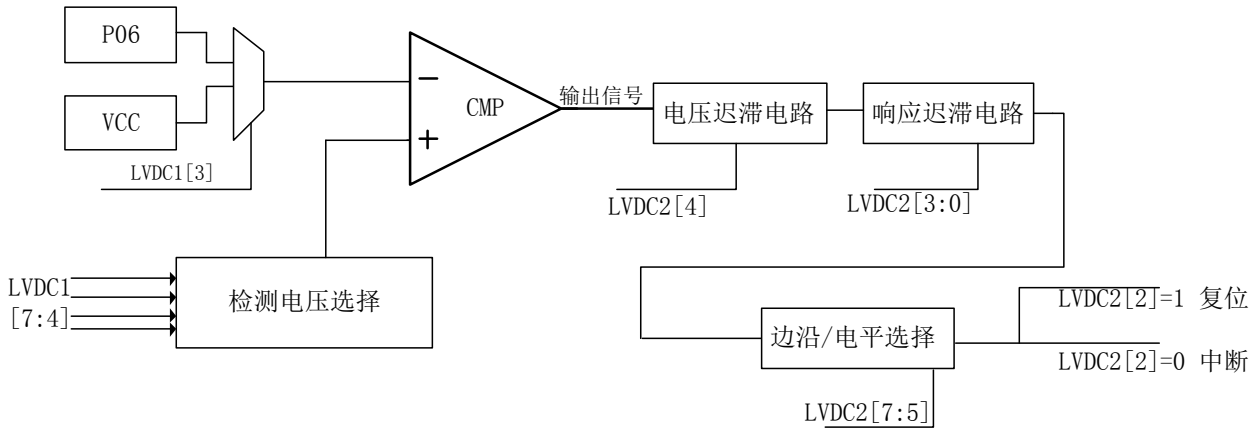


Figure 30-1 低电压检测器框架图

30.3 低电压检测器操作

30.4 建立时间

当使用低电压检测器来监测外部电压或芯片工作电压变化时，需要内部 BGR 输出的参考电压做为比较基准电压。BGR 的启动需要 30uS 的时间，低电压检测器需要等待 30us 的稳定时间，建立低电压检测器的工作点。

30.5 响应时间

当低电压检测器的输入电压发生变换后，到输出结果产生翻转的这段时间被称为响应时间，小于响应时间的电压变化将被忽略。低电压检测器固有响应时间小于<1us，也可通过配置寄存器 LVDC2[0]使能滤波电路。当滤波电路开启后，通过配置寄存器 LVDC2[3:1]选择不同的滤波时间，以获得不同响应时间。

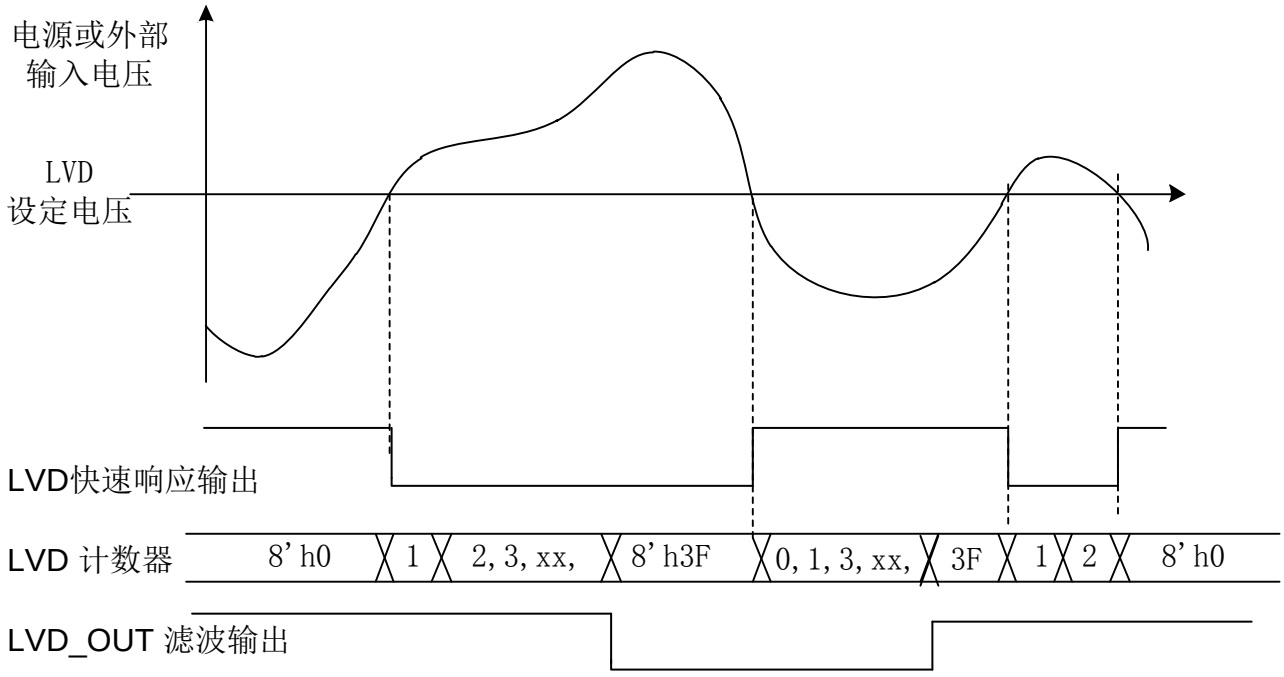


Figure 30-2 LVD 滤波响应时间

30.6 迟滞选择

通过配置寄存器 LVDC2[4]可选择关闭或开启迟滞电路。若迟滞电路关闭，当输入电压高于 LVD 设定电压，结果则为“1”，当输入电压低于 LVD 设定电压，结果则为“0”；若迟滞电路开启，当输入电压高于设定电压+10mV 时，结果为“1”，当输入电压低于设定电压-10mV 时，结果为“0”。

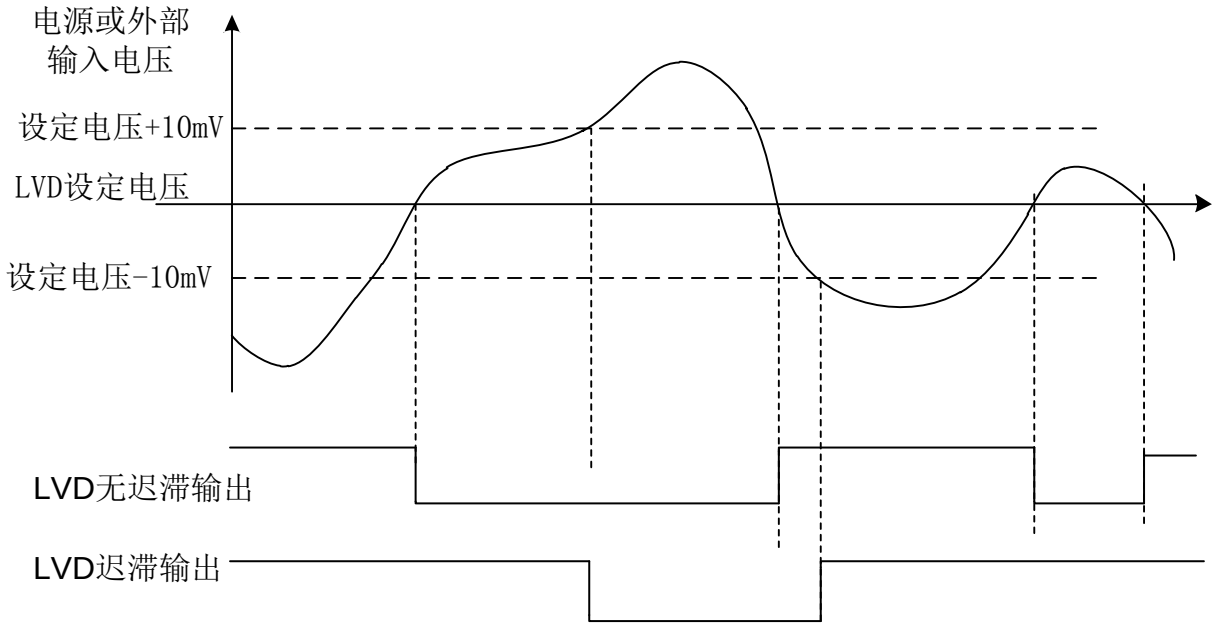


Figure 30-3 LVD 迟滞响应

30.7 低电压检测器寄存器

30.8 低电压检测寄存器 1(LVDC1)

LVDC1					
低电压检测寄存器 1					
地址空间： S:0B6h					
位	7:4	3	2	1	0
符号	LVDVOL_SEL[3:0]	LVD_ Source	LVD_ Reset	LVD_ Flag	LVD_EN
类型	R/W	R/W	R/W	R/W	R/W
复位值	0000	0	0	0	0

位	符号	描述
7:4	LVDVOL_SEL[3:0]	LVD 检测电压选择 1111： 3.60v±100mV 1110： 3.49v±100mV 1101： 3.38v±100mV 1100： 3.27v±100mV 1011： 3.16v±100mV 1010： 3.05v±100mV 1001： 2.94v±100mV 1000： 2.83v±100mV 0111： 2.72v±100mV 0110： 2.61v±100mV 0101： 2.50v±100mV 0100： 2.40v±100mV 0011： 2.30v±100mV 0010： 2.20v±100mV 0001： 2.10v±100mV 0000： 2.00v±100mV
3	LVD_ Source	LVD 检测信号来源选择 1：检测外部引脚 P06 电压 0：检测芯片工作电压 VCC
2	LVD_Reset	LVD 复位/中断选择 1：当 LVD 检测到低于设置电压，产生复位信号。 0：当 LVD 检测到低于设置电压，产生中断信号。
1	LVD_Flag	LVD 标志位 1：当 LVD 检测到低于设置电压 0：当 LVD 检测到高于设置电压
0	LVD_EN	LVD 使能信号 1：LVD 检测器使能 0：LVD 检测器禁用

Figure 30-4 LVDC1寄存器

30.9 低电压检测寄存器 2(LVDC2)

LVDC2						
低电压检测寄存器 2						
地址空间： S:0BFh						
位	7	6	5	4	3:1	0
符号	LVDINT_high	LVDINT_rising	LVDINT_falling	LVD_HYS_EN	LVD_ response[3:1]	LVD_Filter_EN
类型	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	0	0	000	0

位	符号	描述
7	LVDINT_ high	低电压检测器中断/复位边沿检测/高电平检测选择 1：高电平检测产生中断/复位。 0：边沿检测产生中断/复位。
6	LVDINT_rising	低电压检测器中断/复位上升沿选择 1：当低电压检测器输出从低电平转到高电平，产生中断/复位。 0：当低电压检测器输出从低电平转到高电平，不产生中断/复位。
5	LVDINT_falling	低电压检测器中断/复位下降沿选择 1：当低电压检测器输出从高电平转到低电平，产生中断/复位。 0：当低电压检测器输出从高电平转到低电平，不产生中断/复位。
4	LVD_HYS_EN	低电压检测器迟滞开关 1：打开迟滞开关，大约 20mV 0：关闭迟滞开关
3:1	LVD_ response[3:1]	低电压检测器滤波时间选择 000：滤波时间 16us 001：滤波时间 32us 010：滤波时间 64us 011：滤波时间 256us 100：滤波时间 1ms 101：滤波时间 4ms 110：滤波时间 16ms 111：滤波时间 64ms 注意：所有时间均为标准状况下的值
0	LVD_Filter_EN	低电压检测器滤波开关 1：滤波开关打开 0：滤波开关关闭

Figure 30-5 LVDC2寄存器

31 LCD 液晶驱动模块

31.1 LCD 简介

HC16Lxx 内部集成了一个 LCD 驱动电路，可以直接驱动外部 LCD 面板而无需额外的 LCD 驱动芯片。LCD 支持自动显示无需 CPU 的干预，可以工作在低功耗模式下。

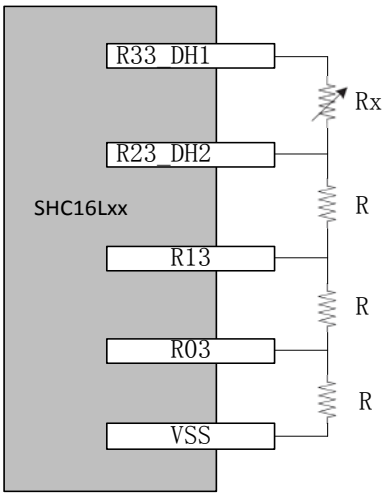
支持 4 种显示功能：

- 1x40
- 2x40
- 3x40
- 4x40

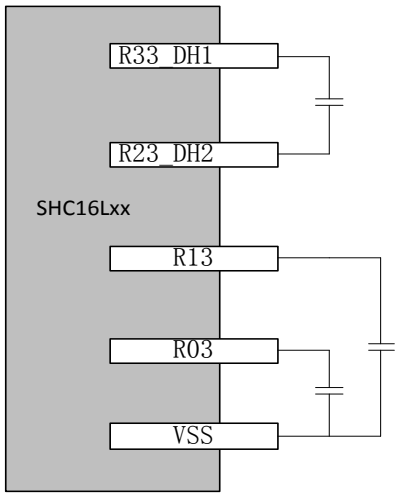
特征：

- 支持电阻分压型 LCD 和电荷泵型 LCD;
- 电阻分压型 LCD 可以根据外部电阻比例关系进行对比度的调节
- 电荷泵型 LCD 最高电压跟随 VCC 变化
- 当选择电阻分压型 LCD 驱动，内部电荷泵自动关闭
- 时钟来源分为 2 种，一种是外部精准 X32K，一种为内部 RC 32K 时钟。
- 自动 LCD 显示内容刷新而无需 CPU 干预。
- 支持 4 种显示模式
  - 静态显示
  - 2 段码，1/2 或 1/3 偏压
  - 3 段码，1/2 或 1/3 偏压
  - 4 段码，1/2 或 1/3 偏压

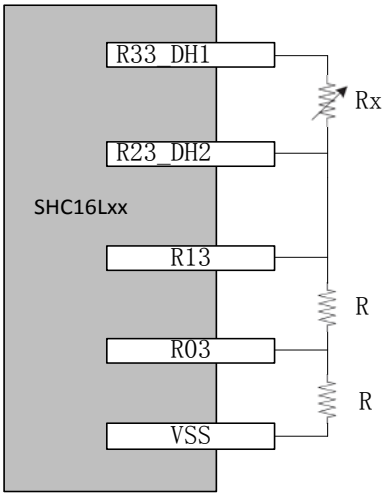
31.2 LCD 应用电路图(电阻分压型和电荷泵型)



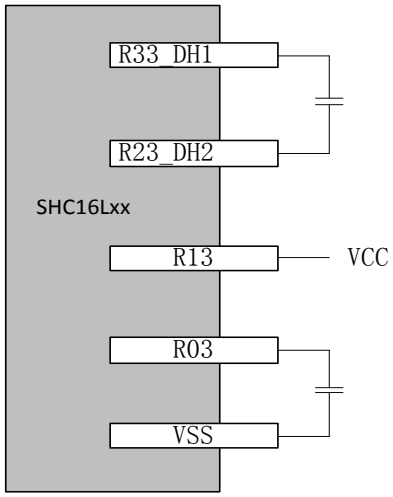
1/3 电阻分压型



1/3 电荷泵型



1/2 电阻分压型



1/2 电荷泵型

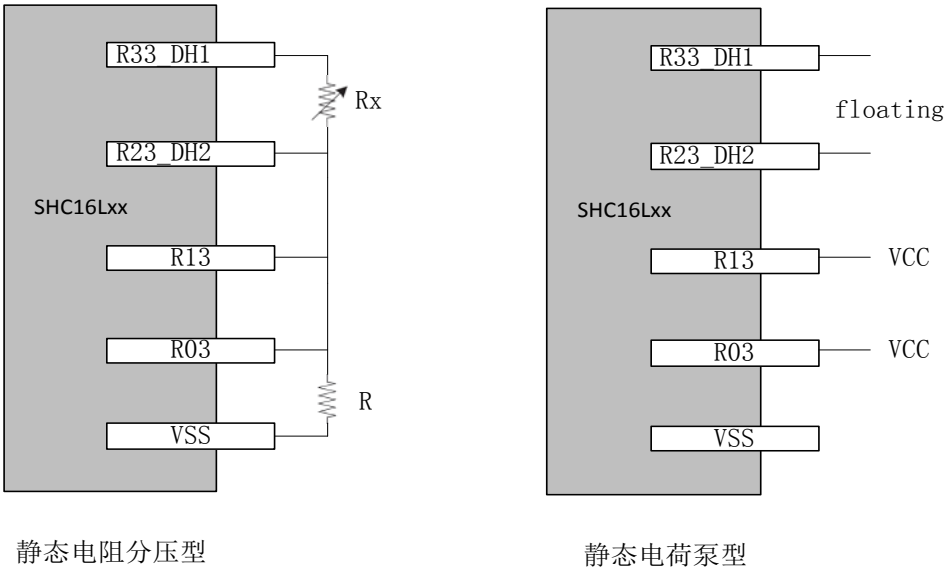


Figure 31-1 LCD 典型应用图

31.3 LCD RAM 映射图

LCD RAM 数据	寄存器地址	COM0 对应值	COM1 对应值	COM2 对应值	COM3 对应值
SEG0	0x00FE00[3:0]	SEG0[0]	SEG0[1]	SEG0[2]	SEG0[3]
SEG1	0x00FE00[7:4]	SEGn[0]	SEGn[1]	SEGn[2]	SEGn[3]
SEG2	0x00FE02[3:0]				
SEG3	0x00FE02[7:4]				
SEG4	0x00FE04[3:0]				
SEG5	0x00FE04[7:4]				
SEG6	0x00FE06[3:0]				
SEG7	0x00FE06[7:4]				
SEG8	0x00FE08[3:0]				
SEG9	0x00FE08[7:4]				
SEG10	0x00FE0A[3:0]				
SEG11	0x00FE0A[7:4]				
SEG12	0x00FE0C[3:0]				
SEG13	0x00FE0C[7:4]				
SEG14	0x00FE0E[3:0]				
SEG15	0x00FE0E[7:4]				
SEG16	0x00FE10[3:0]				
SEG17	0x00FE10[7:4]				
SEG18	0x00FE12[3:0]				
SEG19	0x00FE12[7:4]				
SEG20	0x00FE14[3:0]				
SEG21	0x00FE14[7:4]				
SEG22	0x00FE16[3:0]				
SEG23	0x00FE16[7:4]				
SEG24	0x00FE18[3:0]				
SEG25	0x00FE18[7:4]				
SEG26	0x00FE1A[3:0]				
SEG27	0x00FE1A[7:4]				
SEG28	0x00FE1C[3:0]				
SEG29	0x00FE1C[7:4]				
SEG30	0x00FE1E[3:0]				
SEG31	0x00FE1E[7:4]				
SEG32	0x00FE1F[3:0]				
SEG33	0x00FE1F[7:4]				
SEG34	0x00FE20[3:0]				
SEG35	0x00FE20[7:4]				
SEG36	0x00FE21[3:0]				
SEG37	0x00FE21[7:4]				
SEG38	0x00FE22[3:0]				
SEG39	0x00FE22[7:4]	SEG39[0]	SEG39[1]	SEG39[2]	SEG39[3]

Figure 31-2 LCD RAM 映射图

31.4 LCD 寄存器

31.5 控制寄存器 0（LCDRC0）

LCDRC0								
控制寄存器 0								
地址空间： 0x00FE40h								
位	7	6	5	4	3	2	1	0
符号	LCD_Mode		LCD_Bias	LCD_Frquency		LCD_Frame		LCD_EN
类型	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	0	0	0	0	0	0

位	符号	描述
7:6	LCD_Mode	LCD 显示模式 00：静态显示 01：2 段码显示 10：3 段码显示 11：4 段码显示
5	LCD_Bias	LCD 偏压模式: 1： 1/2 偏压显示模式 0： 1/3 偏压显示模式
4:3	LCD_Frquency	LCD 电荷泵时钟频率选择 11：16K 10：8K 01：4K 00：2K
2:1	LCD_Frame	LCD 帧刷新频率选择 11：512 10：256 01：128（推荐设置：128/4 = 32Hz，人眼不能分辨闪烁且功耗最低） 00：64（默认）
0	LCD_EN	LCD 使能开关 1：LCD 电路打开 0：LCD 电路关闭

Figure 31-8 LCD 控制寄存器 0

31.6 控制寄存器 1（LCDRC1）

LCDRC1								
控制寄存器 1								
地址空间： 0x00FE42h								
位	7	6	5	4	3	2	1	0
符号	--	LCD_BUF		--	--	--	--	LCD_Chareg epump_EN
类型	--	R/W	R/W	--	--	--	--	R/W
复位值	--	0	0	--	--	--	--	0

位	符号	描述
7	--	保留位
6:5	LCD_BUF	LCD 驱动能力选择 11：最强驱动能力 … … 00：最小驱动能力（推荐值）
4:1	--	保留位
0	LCD_Chareg epump_EN	LCD 电荷泵使能位 1： LCD 电荷泵启动，LCD 使用电荷泵产生 LCD 显示偏压 0： LCD 电荷泵关闭，LCD 使用外部电阻分压来得到 LCD 显示偏压

Figure 31-9 LCD 控制寄存器 1

31.7 显示内存寄存器(LCDRAM0 - LCDRAM19)

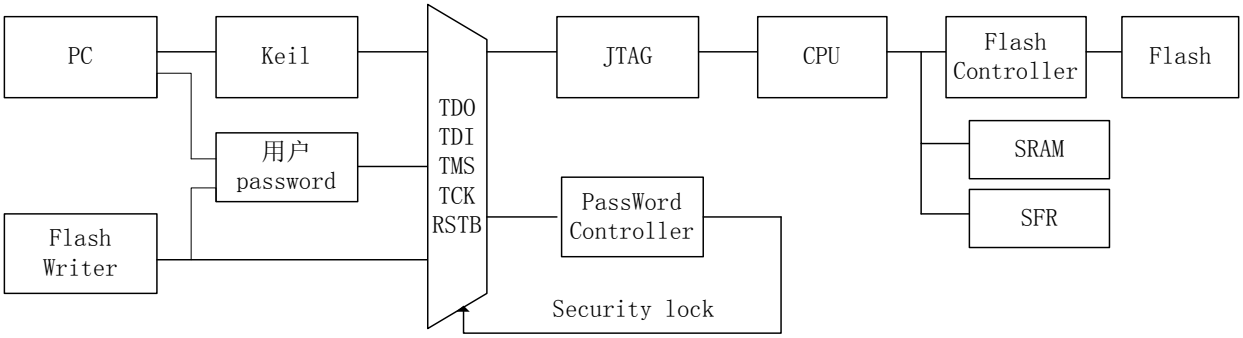
LCDRAM0 – LCDRAM19								
显示内存寄存器								
地址空间： 0x00FE00+（2*n）h LCDRAM0：0x00FE00 LCDRAM1：0x00FE02 … LCDRAM15：0x00FE1E LCDRAM16：0x00FE20 … LCDRAM19：0x00FE26								
位	7	6	5	4	3	2	1	0
符号	LCDRAM0 – LCDRAM19							
类型	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	0	0	0	0	0	0

位	符号	描述
7：4	LCDRAMx[7:4]	SEG 高位显示
3:0	LCDRAMx[3:0]	SEG 低位显示

Figure 31-10 LCD 显示内存寄存器

32 嵌入式调试系统（JTAG）

32.1 HC16Lxx 特有的密码保护电路



JTAG调试器端口与flash烧录端口与密码解码端口共用
(a) 默认为密码解码端口（上电复位后恢复到默认状态，其他复位，此端口特性不变）
(b) JTAG调试端口（只有当128位密码正确输入之后才导通此通道）
(C) Flash烧录端口（与JTAG同一通道，使用ICP功能进行flash的在线，脱机烧录）

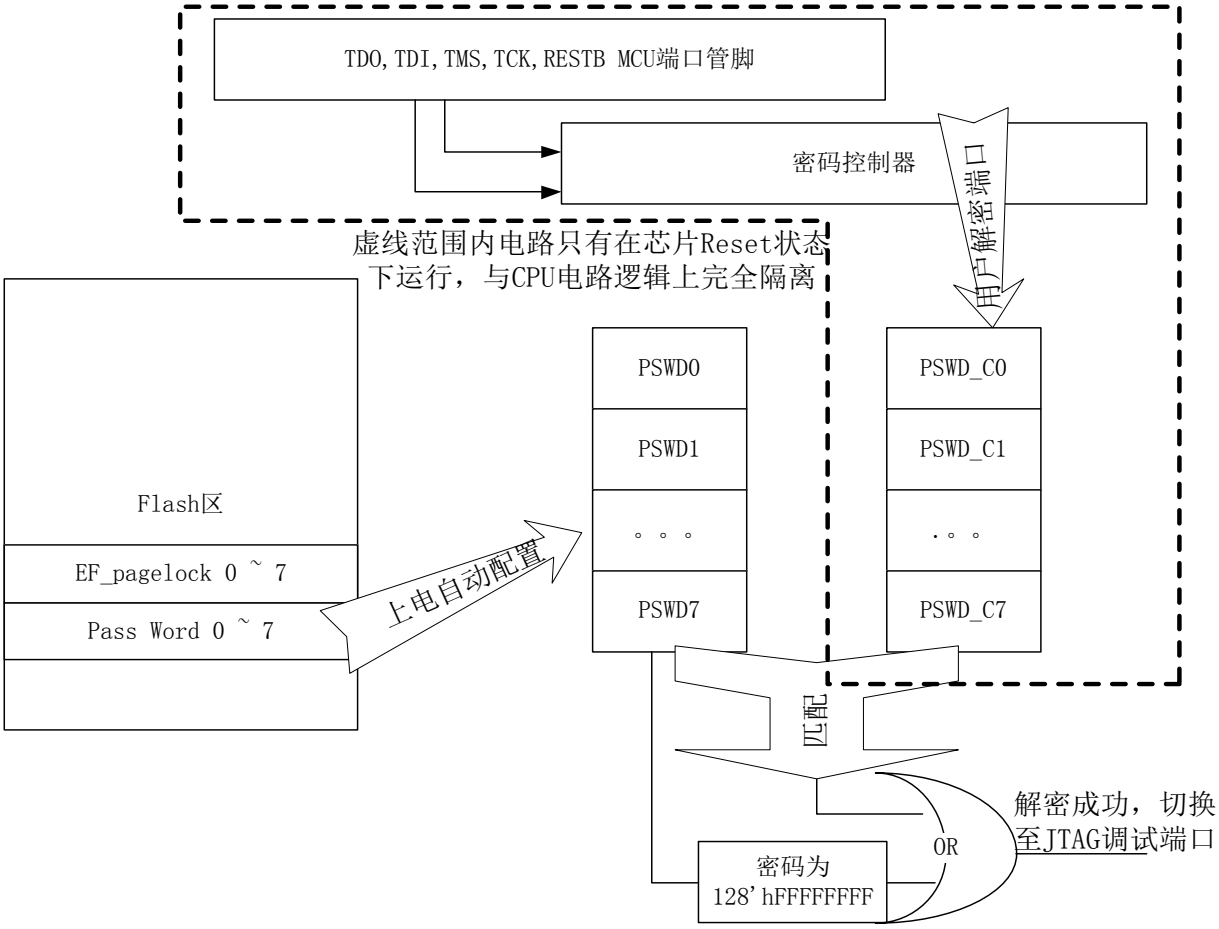


Figure 32-1 HC16Lxx 加解密原型

通用 MCU 的 JTAG 是非加密的。但 HC16Lxx 为了保护用户密码及系统安全，在 JTAG 端口上设计了

128 位密码保护。上电后，端口默认配置成密码输入端口，只有在上位机软件的配合下输入正确的 128 位密码，芯片才自动切换到 JTAG 调试端口。可以使用 R-Link 进行在线调试。因为 Flash 烧录器模拟解密波形以及 JTAG 波形，同样适用 JTAG 调试端口及协议。

密码保护电路与 CPU 电路完全隔离，因此当外界密码输入不匹配时，芯片不会做出任何反应。输入密码与授权密码一致时，JTAG 可以连接调试；而输入密码与授权密码不一致时，JTAG 无法连接调试。

注意事项：

1. 当用户没有设置密码时，即没有在 Flash 区的 FF:0090 – FF:009F 写入密码时，同 Flash 擦除后是相同的值，那密码即为 128 位“1”， HC16Lxx 视为用户没有密码保护。上电复位之后，JTAG 调试端口自动打开。
2. 密码保护电路与 CPU 电路完全独立，CPU 无法进行密码的读取或确认。因此用户输入授权密码调试完成之后，需要重新写入错误密码或上电复位，才能关闭原先破解后的 JTAG 端口。

UserPassWord0		=24’hFF0090 (Flash 地址)
UserPassWord1		=24’hFF0091 (Flash 地址)
UserPassWord2		=24’hFF0092 (Flash 地址)
UserPassWord3		=24’hFF0093 (Flash 地址)
UserPassWord4		=24’hFF0094 (Flash 地址)
UserPassWord5		=24’hFF0095 (Flash 地址)
UserPassWord6		=24’hFF0096 (Flash 地址)
UserPassWord7		=24’hFF0097 (Flash 地址)
UserPassWord8		=24’hFF0098 (Flash 地址)
UserPassWord9		=24’hFF0099 (Flash 地址)
UserPassWord10		=24’hFF009A (Flash 地址)
UserPassWord11		=24’hFF009B (Flash 地址)
UserPassWord12		=24’hFF009C (Flash 地址)
UserPassWord13		=24’hFF009D (Flash 地址)
UserPassWord14		=24’hFF009E (Flash 地址)
UserPassWord15		=24’hFF009F (Flash 地址)

Figure 32-2 用户密码保护区

32.2 传统 JTAG 调试电路

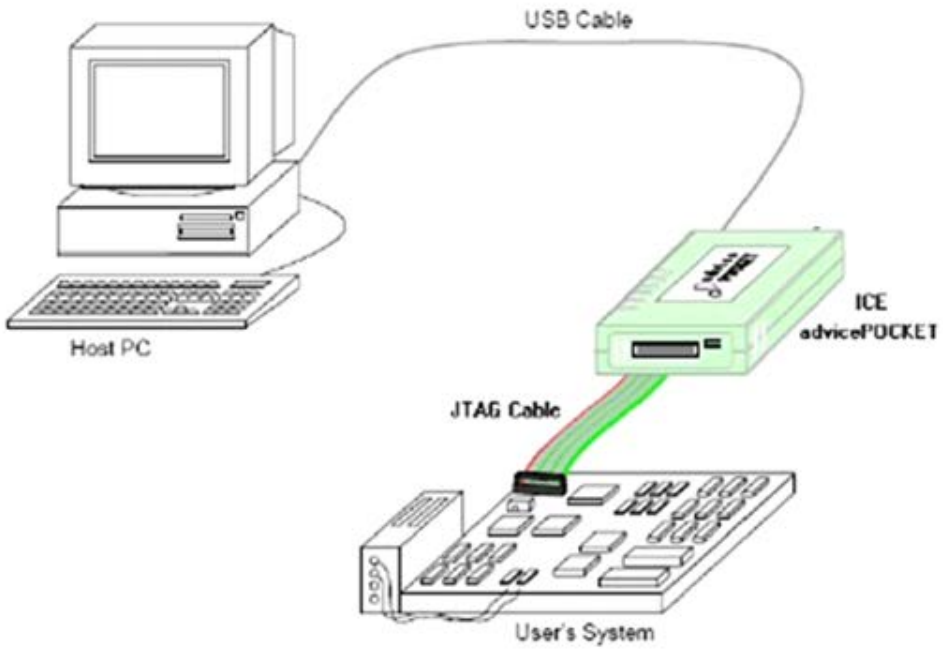


Figure 32-3 HC16Lxx JTAG调试系统

HC16Lxx提供了内建实时调试接口BIRD (Built-in Real-time Debugger),提供全方位的调试功能接口。

BIRD由一些软件硬件配合使用：

- 带有BIRD调试电路的MCU ： HC16Lxx
- Rlink_BIRD ： USB-JTAG适配器，连接PC和HC16Lxx
- Debug software ： IDE开发环境为C251版本的Keil

33 电气特性

33.1 测试条件

测试基于室温（25℃） 和 VCC=3.0V。

33.2 芯片推荐工作条件

符号	参数	条件	最小	标准	最大	单位
Viopin	加载在IO管脚的电压		-0.3		VCC+0.3	V
VCC	工作电压		1.80	3.0	3.80*1	V
Tstg	保存温度		-40		125	℃
Top	工作温度		-40		85	℃
Fcpu	CPU工作频率		32K		20M	Hz
V_{ESDHBM}	静电防护（人体放电模式）	室温25℃			2	kV
V_{ESDCDM}	静电防护（外部设备放电模式）	室温25℃			1	kV

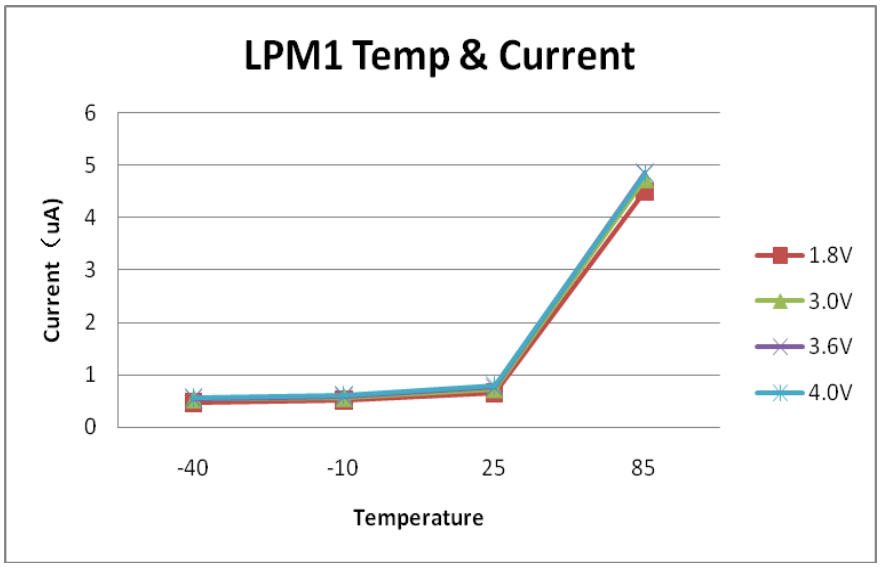
注意事项: 1.VCC的可加载电压范围是0V~3.8V。

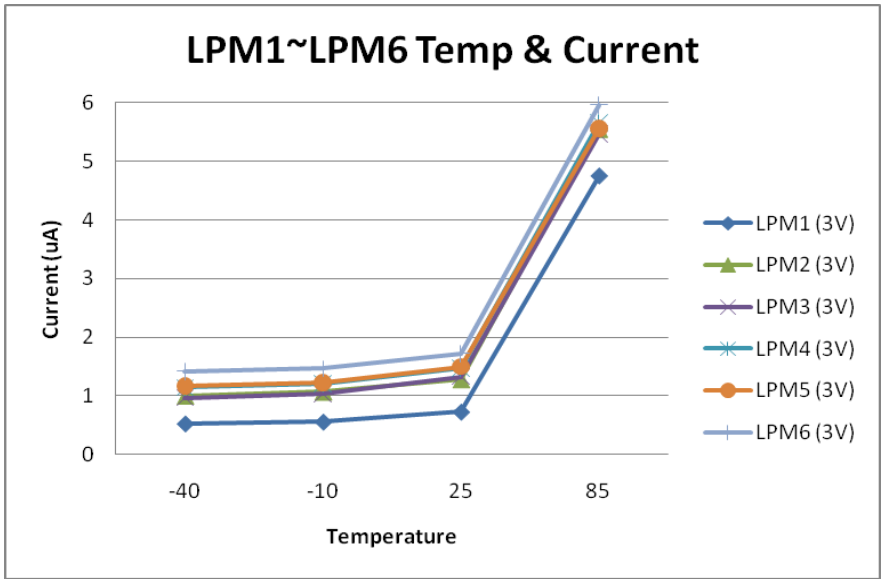
33.3 工作电流特性

符号	参数	条件	VCC	最小	标准	最大	单位
I(AM1)	工作模式1	LDO输出电压为1.8V -40°C 至 85°C F(MCLK) = 4MHz F(ACLK)= 32.768KHz F(system) = F(MCLK) While(1)loop 在Flash内执行指令，外设时钟全部关闭	1.8v		800	900	uA
			3.0v		800	900	uA
			3.8v		800	900	uA
I(AM2)	工作模式2	LDO输出电压为1.8V -40°C 至 85°C F(MCLK) = 4MHz F(ACLK)= 32.768KHz F(system) = F(MCLK) While(1)loop 在Flash内执行指令，外设时钟全部打开	1.8v		1000	1200	uA
			3.0v		1000	1200	uA
			3.8v		1000	1200	uA
I(IM1)	空闲模式1	LDO输出电压为1.8V	1.8v		130	150	uA

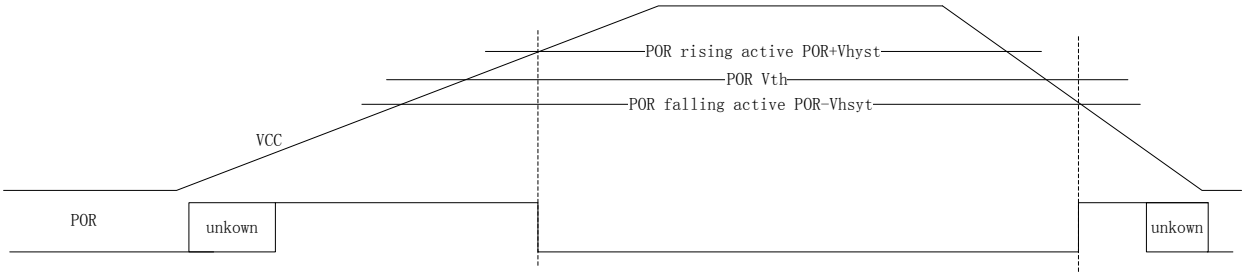
		-40°C 至 85°C F(MCLK) = 4MHz F(ACLK)= 32.768K Hz F(system) = F(MCLK) 空闲模式下定时器计数 在Flash内执行指令，除定时器外的其它外设时钟全部关闭	3.0v		130	150	uA
			3.8v		130	150	uA
I(AM3)	工作模式3	LDO输出电压为1.8V -40°C 至 85°C F(MCLK)关闭 F(ACLK)= 32.768KHz F(system) = F(ACLK) While(1)loop 在Flash内执行指令	1.8v		8	10	uA
			3.0v		8	10	uA
			3.8v		8	10	uA
I(IM2)	空闲模式2	LDO输出电压为1.8V -40°C 至 85°C F(MCLK) 关闭 F(ACLK)= 32.768K Hz F(system) = F(ACLK) 空闲模式下定时器计数 在Flash内执行指令，除定时器的时钟外，其它外设时钟关闭	1.8v		3	5	uA
			3.0v		3	5	uA
			3.8v		3	5	uA
I(AM4)	工作模式4	LDO输出电压为1.8V -40°C 至 85°C F(MCLK) = 4MHz F(ACLK)= 32.768KHz F(system) = F(MCLK) While(1)loop 在SRAM内执行指令，外设时钟全部关闭	1.8v		560	600	uA
			3.0v		560	600	uA
			3.8v		560	600	uA
I(IM3)	空闲模式3	LDO输出电压为1.8V -40°C 至 85°C F(MCLK) = 4MHz F(ACLK)= 32.768KHz F(system) = F(MCLK) 空闲模式下计数器保持计	1.8v		130	150	uA
			3.0v		130	150	uA
			3.8v		130	150	uA

		数，执行SRAM内指令					
I(LPM1)	低功耗模式1	LDO输出电压为1.8V -40°C 至 85°C 上电复位有效，寄存器，RAM和CPU保持	1.8v		0.9	5.0	uA
			3.0v		0.9	5.0	uA
			3.8v		0.9	5.0	uA
I(LPM2)	低功耗模式2	LDO输出电压为1.8V -40°C 至 85°C 除了包括低功耗模式1配置外，打开RTC与外部32.768KHz晶振时钟源	1.8v		1.2	5.0	uA
			3.0v		1.2	5.0	uA
			3.8v		1.2	5.0	uA
I(LPM3)	低功耗模式3	LDO输出电压为1.8V -40°C 至 85°C 除了包括低功耗模式1配置外，打开LCD显示	1.8v		1.4	6.0	uA
			3.0v		1.4	6.0	uA
			3.8v		1.4	6.0	uA
I(LPM4)	低功耗模式4	LDO输出电压为1.8V -40°C 至 85°C 除了包括低功耗模式2配置外，打开LCD显示	1.8v		1.6	6.0	uA
			3.0v		1.6	6.0	uA
			3.8v		1.6	6.0	uA
I(LPM5)	低功耗模式5	LDO输出电压为1.8V -40°C 至 85°C 除了包括低功耗模式4配置外，使能看门狗（时钟源为32.768KHz晶振）	1.8v		1.7	6.0	uA
			3.0v		1.7	6.0	uA
			3.8v		1.7	6.0	uA
I(LPM6)	低功耗模式6	LDO输出电压为1.8V -40°C 至 85°C 除了包括低功耗模式4配置外，使能看门狗（时钟源为内部32KHz专用时钟）	1.8v		2.0	6.5	uA
			3.0v		2.0	6.5	uA
			3.8v		2.0	6.5	uA


LPM1 Temp & Current
Current（uA）
6
5
4
3
2
1
0
-40
-10
25
85
Temperature
1.8V
3.0V
3.6V
4.0V


LPM1~LPM6 Temp & Current
Current(uA)
6
5
4
3
2
1
0
-40
-10
25
85
Temperature
LPM1 (3V)
LPM2 (3V)
LPM3 (3V)
LPM4 (3V)
LPM5 (3V)
LPM6 (3V)

33.4 POR 上电复位、Reset 端口复位

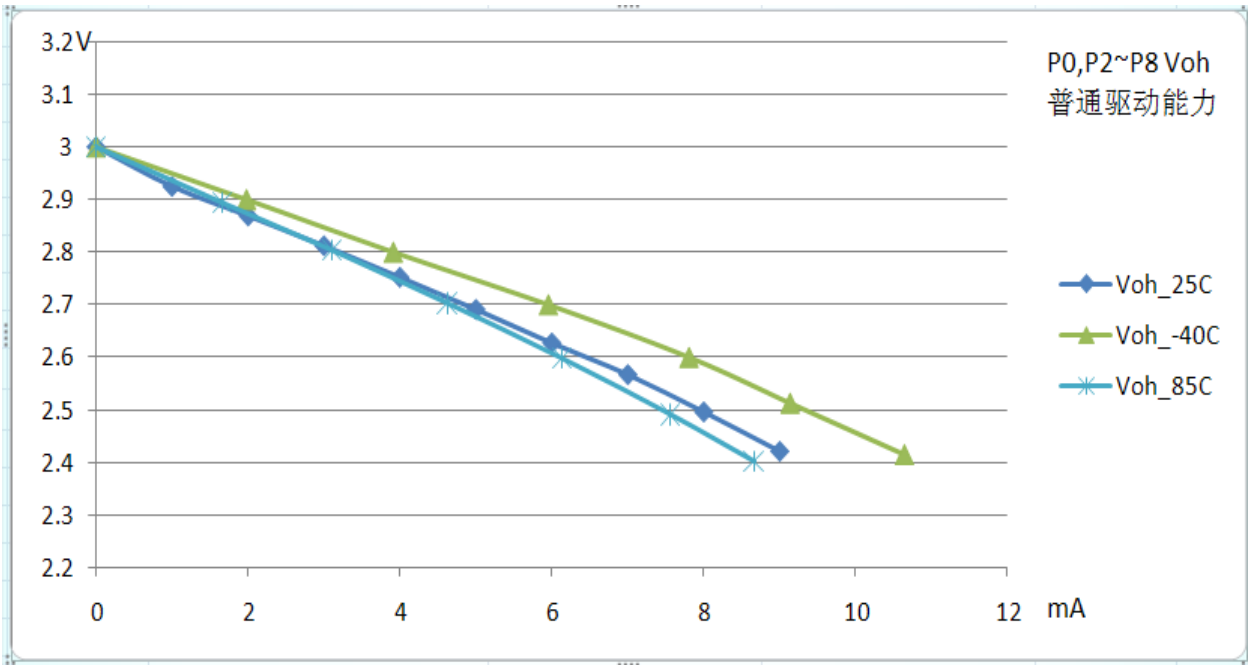


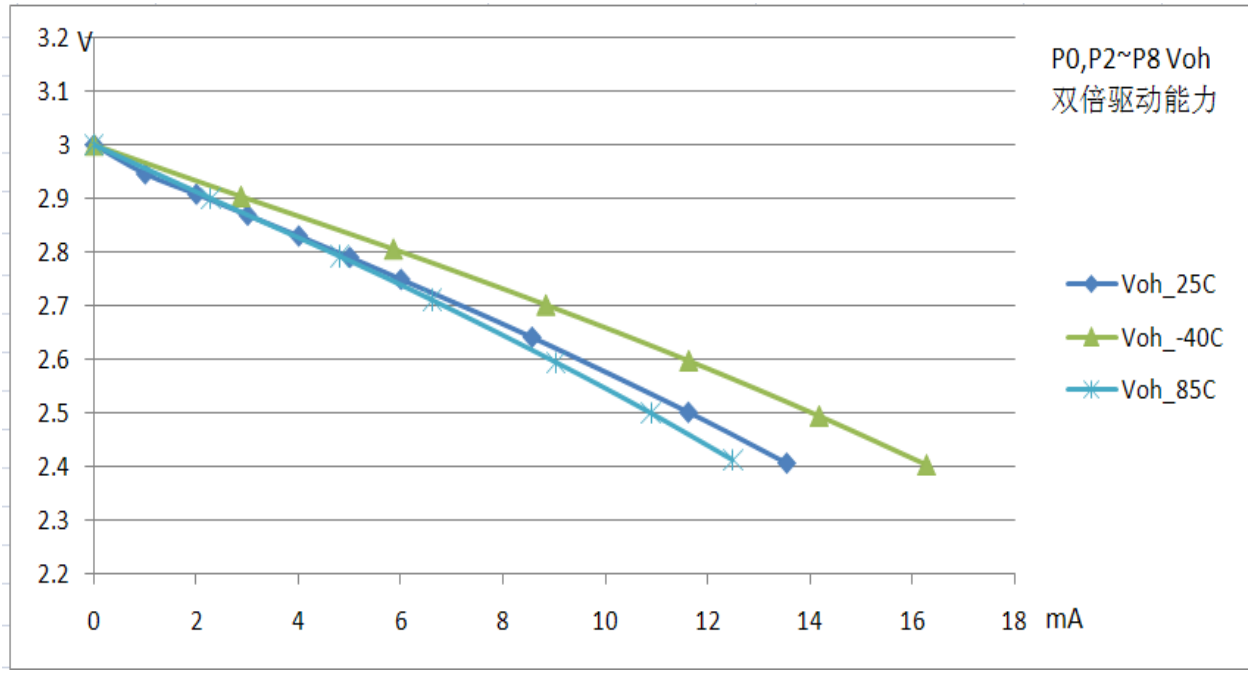
符号	参数	条件	最小	标准	最大	单位
$V_{POR+Vhyst}$	POR 释放电压（上电过程） dVCC/dt > 3V/s	T_{AMB} = -40℃	1.60	1.70	1.80	V
		T_{AMB} = 25℃	1.30	1.40	1.50	V
		T_{AMB} = 85℃	1.00	1.10	1.20	V
$V_{POR-Vhyst}$	POR产生电压（断电过程）	T_{AMB} = -40℃	1.60	1.70	1.80	V
		T_{AMB} = 25℃	1.30	1.40	1.50	V
		T_{AMB} = 85℃	1.00	1.10	1.20	V
T_{reset}	可被芯片接收的Reset端口脉冲宽度	T_{AMB} = 25℃	20			uS

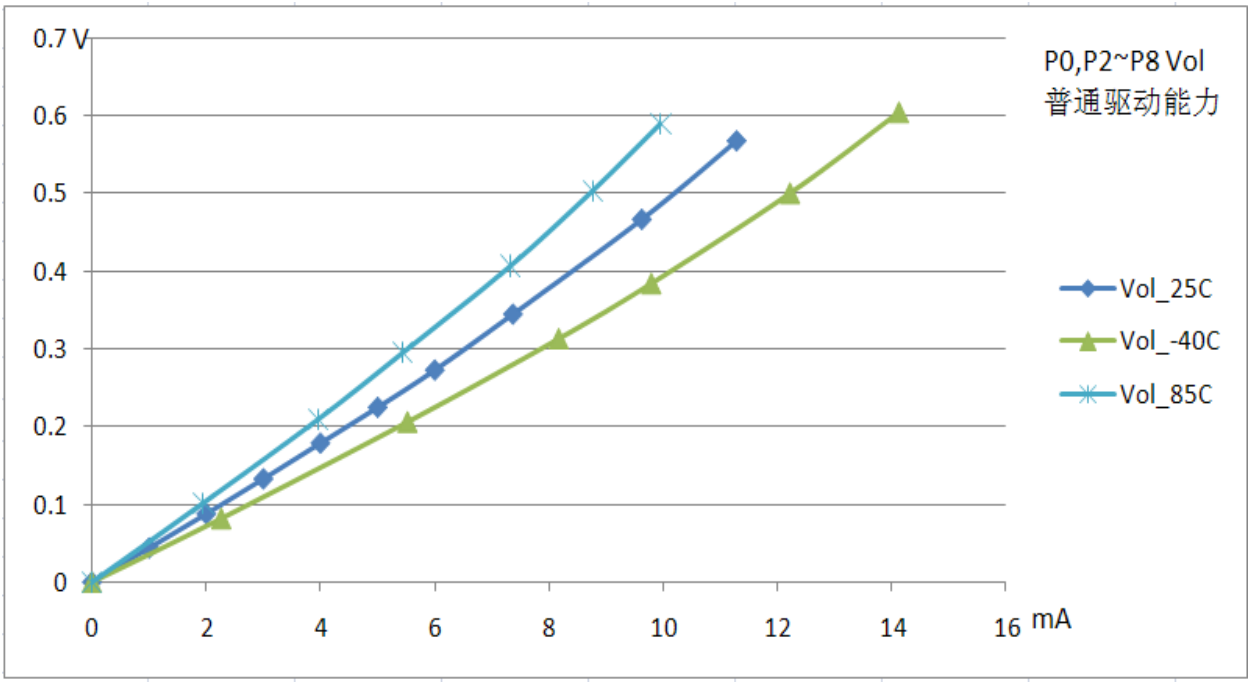
33.5 端口特性 -- P0,P2,P3,P4,P5,P6,P7,P8

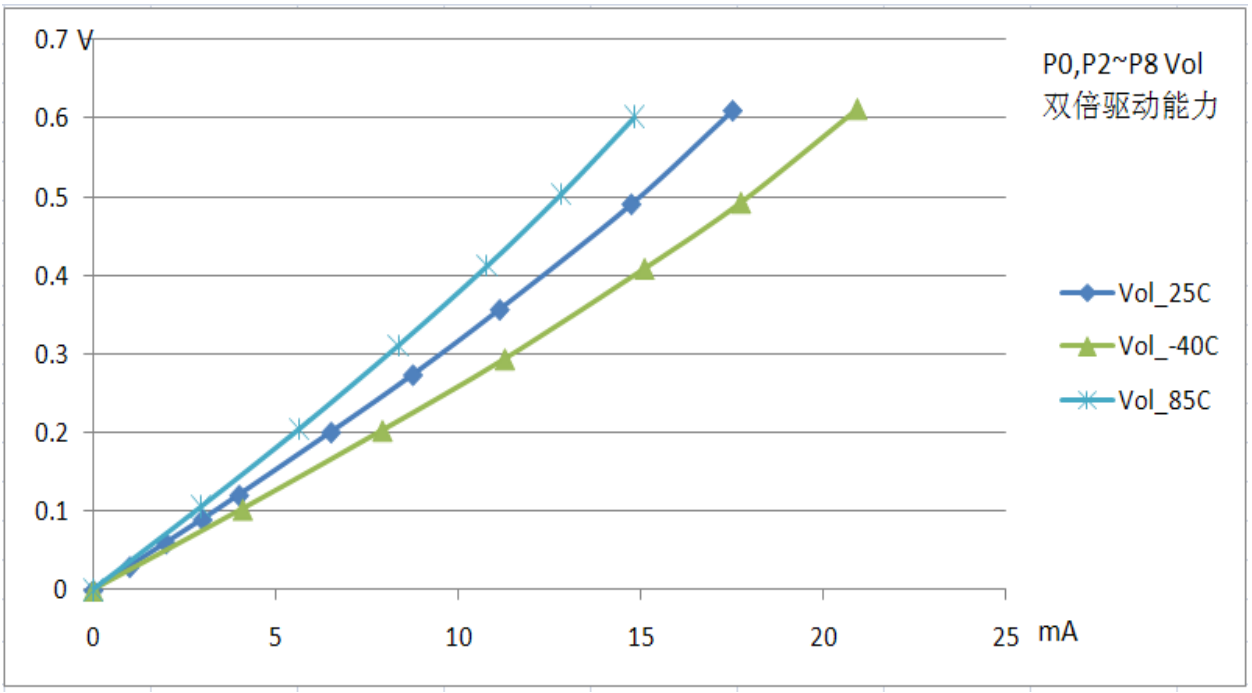
符号	参数	测试条件	最小	最大	单位
VOH	高电平输出电压（驱动电流）	IOH(max) = -4 mA, VCC = 3.0 V *1	VCC-0.25	VCC	V
		IOH(max) = -6 mA, VCC = 3.0 V *2	VCC--0.6	VCC	V
VOL	低电平输出电压（灌电流）	IOL(max) = 4 mA, VCC = 3.0 V *1		VSS+0.25	V
		IOL(max) = 6 mA, VCC = 3.0 V *2		VSS+0.6	V
VOHD	高电平输出电压（双倍驱动电流）	IOH(max) = -8 mA, VCC = 3.0 V *1	VCC-0.25	VCC	V
		IOH(max) = -12 mA, VCC = 3.0 V *2	VCC--0.6	VCC	V
VOLD	低电平输出电压（双倍驱动电流）	IOL(max) = 8 mA, VCC = 3.0 V *1		VSS+0.25	V
		IOL(max) = 12 mA, VCC = 3.0 V *2		VSS+0.6	V

注意事项：
1. 芯片全部输出管脚的IOH(max)和IOL(max)不应超过40mA，以保证芯片电压下降最大值不超出规范
2. 芯片全部输出管脚的IOH(max)和IOL(max)不应超过±100mA，以保证芯片电压下降最大值不超出规范


3.2V
3.1
3
2.9
2.8
2.7
2.6
2.5
2.4
2.3
2.2
0
2
4
6
8
10
12
mA
P0,P2~P8 Voh
普通驱动能力
Voh_25C
Voh_-40C
Voh_85C


3.2 V
3.1
3
2.9
2.8
2.7
2.6
2.5
2.4
2.3
2.2
0
2
4
6
8
10
12
14
16
18
mA
P0,P2~P8 Voh
双倍驱动能力
Voh_25C
Voh_-40C
Voh_85C

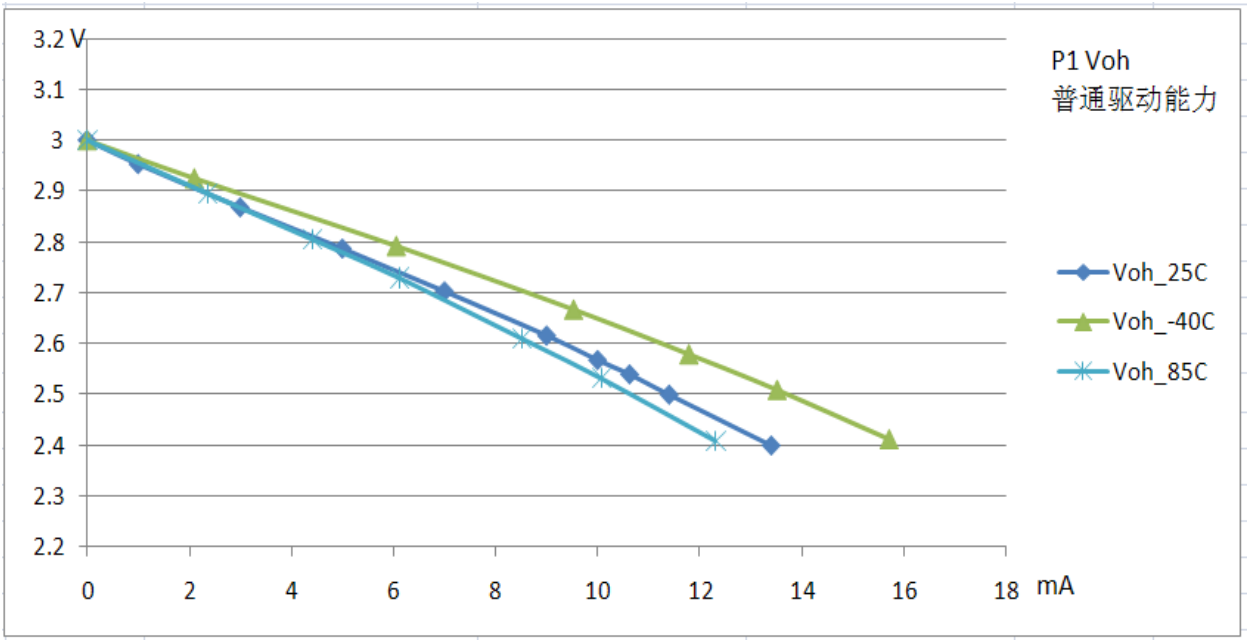

0.7 V
0.6
0.5
0.4
0.3
0.2
0.1
0
0
2
4
6
8
10
12
14
16
mA
P0,P2~P8 Vol
普通驱动能力
Vol_25C
Vol_-40C
Vol_85C

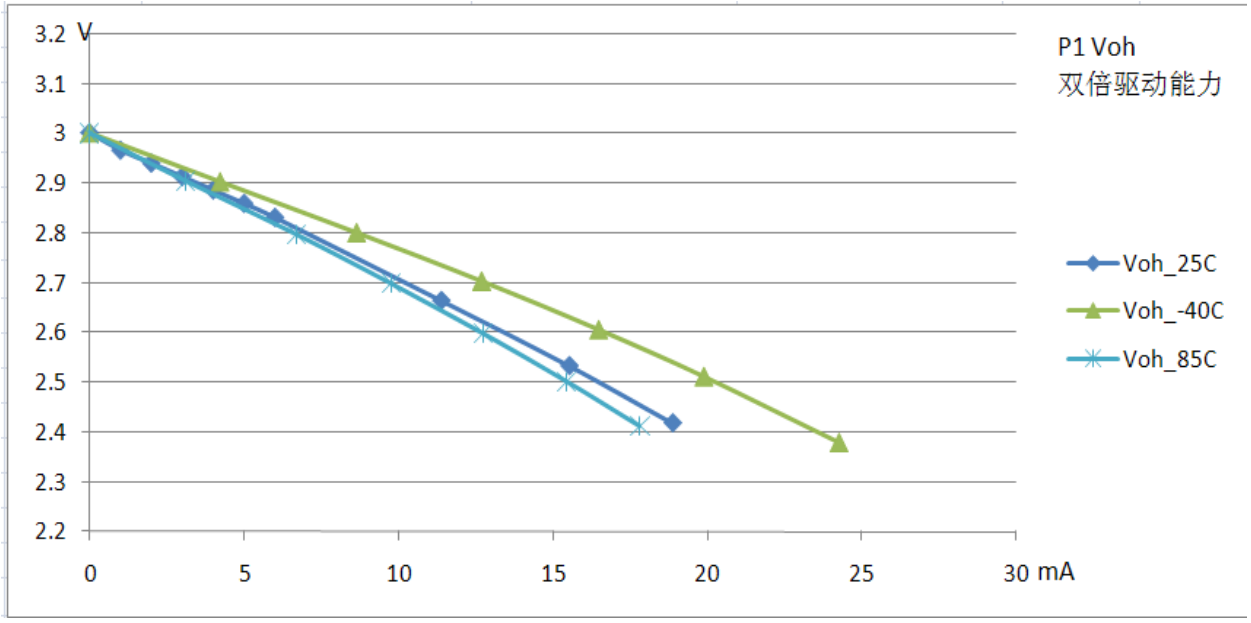

0.7 V
0.6
0.5
0.4
0.3
0.2
0.1
0
0
5
10
15
20
25
mA
P0,P2~P8 Vol
双倍驱动能力
Vol_25C
Vol_-40C
Vol_85C

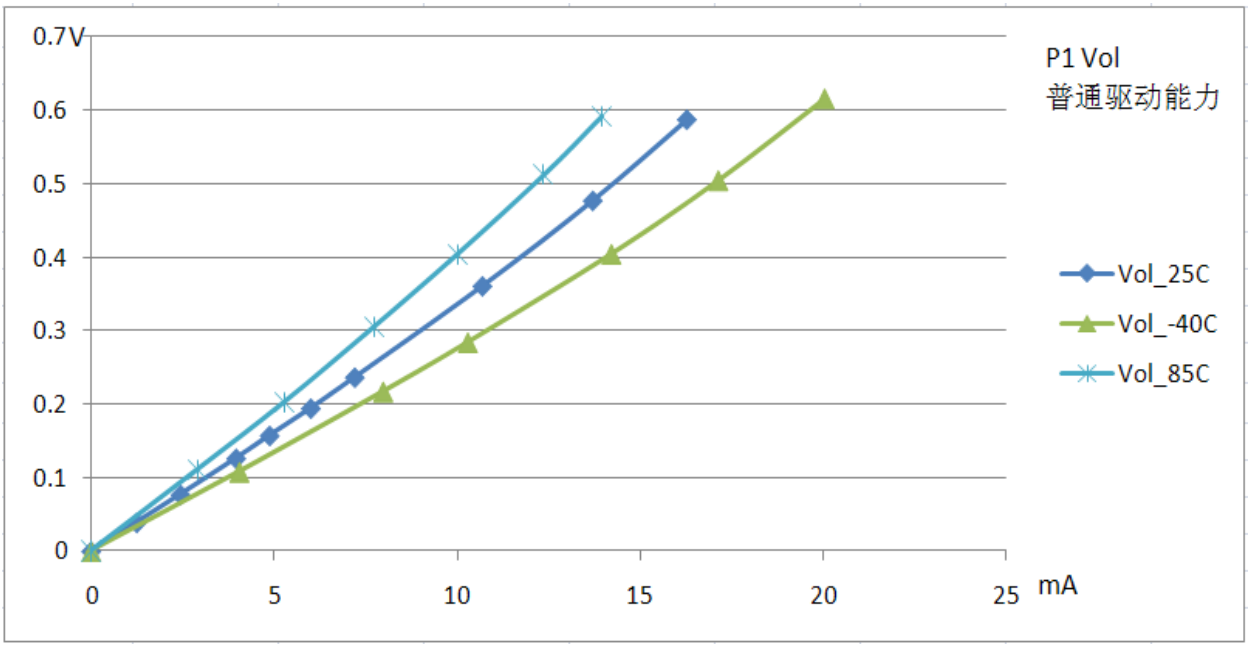
33.6 端口特性-- P1

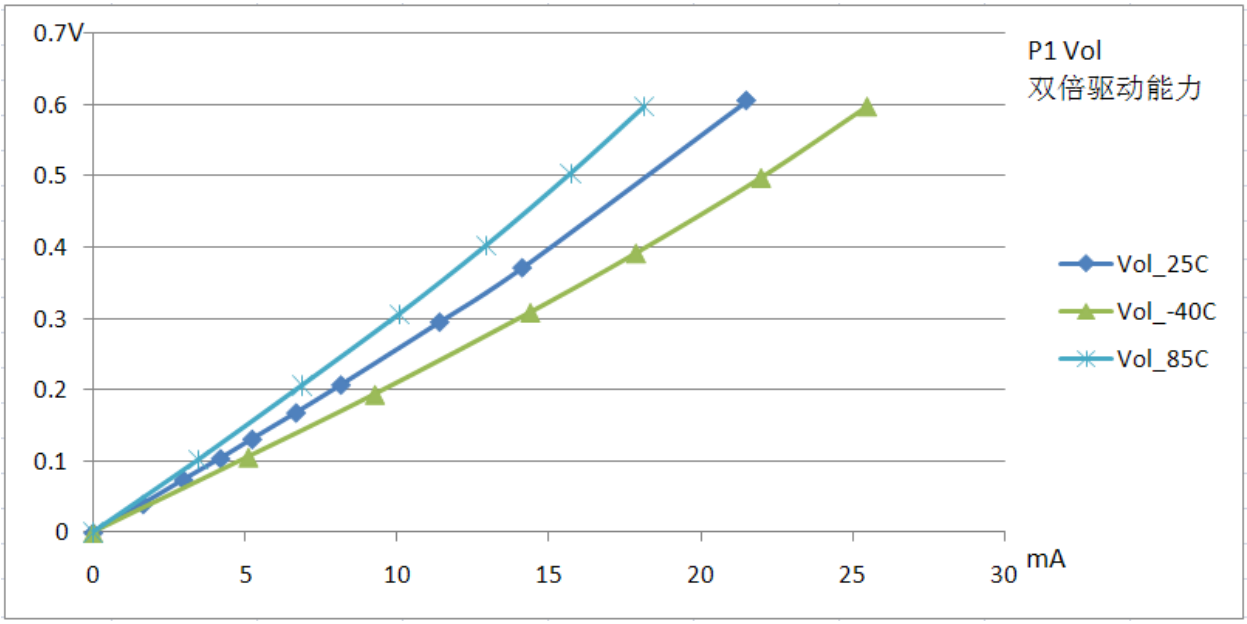
符号	参数	条件	最小	最大	单位
VOH	高电平输出电压（驱动电流）	IOH(max) = -8 mA, VCC = 3.0 V *1	VCC-0.25	VCC	V
		IOH(max) = -12 mA, VCC = 3.0 V *2	VCC--0.6	VCC	V
VOL	低电平输出电压（灌电流）	IOL(max) = 8 mA, VCC = 3.0 V *1		VSS+0.25	V
		IOL(max) = 12 mA, VCC = 3.0 V *2		VSS+0.6	V
VOHD	高电平输出电压（双倍驱动电流）	IOH(max) = -12 mA, VCC = 3.0 V *1	VCC-0.25	VCC	V
		IOH(max) = -16 mA, VCC = 3.0 V *2	VCC--0.6	VCC	V
VOLD	低电平输出电压（双倍驱动电流）	IOL(max) = 12 mA, VCC = 3.0 V *1		VSS+0.25	V
		IOL(max) = 16 mA, VCC = 3.0 V *2		VSS+0.6	V

注意事项：
1. 芯片全部输出管脚的IOH(max)和IOL(max)不应超过40mA，以保证芯片电压下降最大值不超出规范
2. 芯片全部输出管脚的IOH(max)和IOL(max)不应超过±100mA，以保证芯片电压下降最大值不超出规范




3.2 V
3.1
3
2.9
2.8
2.7
2.6
2.5
2.4
2.3
2.2
0
5
10
15
20
25
30 mA
P1 Voh
双倍驱动能力
Voh_25C
Voh_-40C
Voh_85C


0.7V
0.6
0.5
0.4
0.3
0.2
0.1
0
0
5
10
15
20
25
mA
P1 Vol
普通驱动能力
Vol_25C
Vol_-40C
Vol_85C



33.7 施密特触发器输入特性-- P0,P1,P2,P3,P4,P5,RESET, JTAG

符号	参数	VCC	最小	标准	最大	单位
VIT+	上升沿触发门限电压	1.8v			1.4	V
		3.0v			2.4	V
		3.6v			2.8	V
VIT--	下降沿触发门限电压	1.8v	0.4			V
		3.0v	0.6			V
		3.6v	0.8			V
Vhys	输入电压门限范围 (VIT+ -- VIT--)	1.8v		1.0		V
		3.0v		1.8		V
		3.6v		2.0		V
Rpullhigh	上拉电阻阻值 VIN=VSS	1.8v		40		Kohm
		3.0v		40		Kohm
		3.6v		40		Kohm
CI	输入电容 VIN=VSS 或 VCC			5		pf

33.8 外部输入采样要求 Timer Gate, Timer clock

符号	参数	条件	VCC	最小	最大	单位
t(int)	外部中断时序	外部信号触发中断间隔时间*1	1.8v	50		ns
			3.0v	40		ns
			3.6v	30		ns
t(cap)	计数器捕捉所需时间	计数器0/1捕捉脉冲的宽度 Fsystem = 4MHz	1.8v	6		us
			3.0v	6		us
			3.6v	6		us
t(clk)	计数器时钟特性	计数器外部时钟输入频率 Fsystem = 4MHz	1.8v		4/24	MHz
			3.0v		4/24	MHz
			3.6v		4/24	MHz
t(pca)	PCA时钟特性	PCA外部输入时钟频率 Fsystem = 4MHz	1.8v		4/12	MHz
			3.0v		4/12	MHz
			3.6v		4/12	MHz

注意事项：

1. 外部信号每隔一次 t(int) 的最小时间可以触发一次中断。不同的工作电压条件下，此最小时间不相同。

33.9 端口漏电特性 P0,P1,P2,P3,P4,P5

符号	参数	条件	VCC	最大	单位
Ilkg(Px.y)	漏电电流	V(Px.y) *1*2	1.8 V/3.6 V	+/-50	nA

注意事项：

1.被测管脚需要连接到VSS或VCC，除非有其它备注

2.GPIO必须为输入模式

33.10 Flash

符号	参数	条件	最小	标准	最大	单位
ECflash	Flash擦除次数	LDO输出电压1.5V，测试温度为常温25℃	20K	100K		次
RETflash	数据保持时间	测试温度为85℃	10			年
Tw_prog	双字（double word）编程时间		20		40	μs
Tp_erase	页擦除时间		20		40	ms
Tm_erase	整片擦除时间		20		40	ms

33.11 外部 32.768KHz 晶振

符号	参数	条件	最小	标准	最大	单位
fLFXO	时钟频率			32.768		kHz
ESRLFXO	等效串联电阻值(ESR)			30	120	kOhm
CLFXOL	晶振外部负载电容选择范围		5		25	pF
DCLFXO	占空比		48	50	53.5	%
ILFXO	经过启动时间后，晶振所消耗的电流	ESR= 30 kOhm, CL=6 pF		250		nA
tLFXO	晶振启动时间	ESR=30 kOhm, CL=10 pF, 占空比达到40%--60%之间		400		ms

33.12 内部 16MHz 时钟

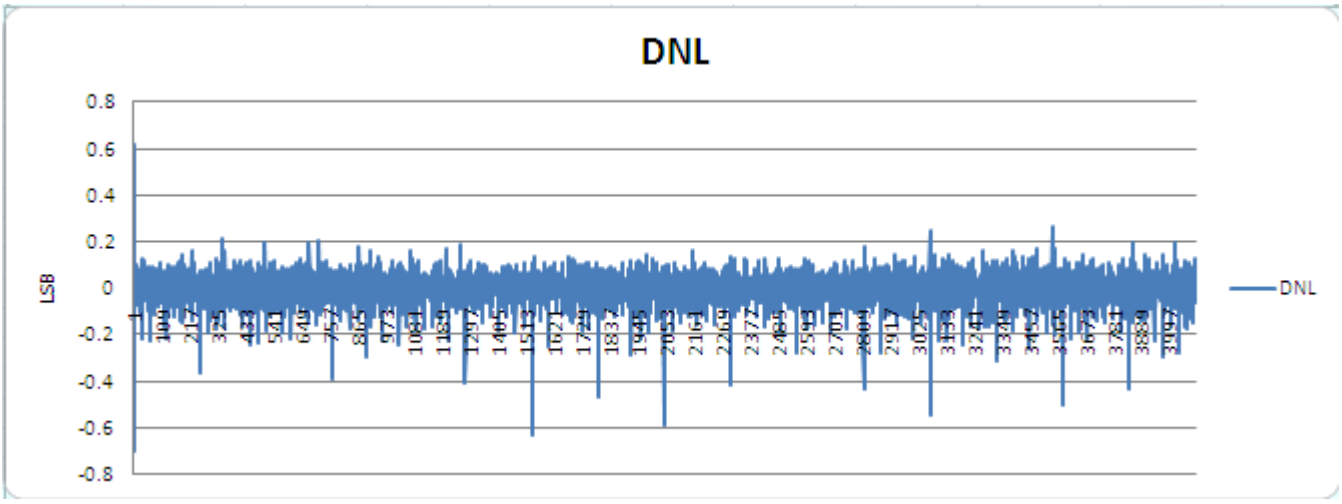
符号	参数	条件	最小	标准	最大	单位
Fmain	内部RC振荡器频率		1.5	2.0/ 4.0/ 8.0/	22.0	MHz
Fstart	振荡器启动时间（不包括软件校准）	Fmain = 2MHz		6.0		μs
		Fmain = 4MHz		4.0		μs
		Fmain = 8MHz		3.0		μs
		Fmain = 16MHz		2.5		μs
DCmain	占空比		48	50	52	%
IFmain	电流消耗	Fmain = 8MHz		80		μA
Dev2M	2MHz工作频率时的偏差	Fmain = 2MHz VCC = 2.0v ~ 3.8v, -40°C to 85°C	-2.5%	2MHz	+2.5%	MHz
		Fmain = 2MHz VCC = 1.8v ~ 3.8v, -40°C to 85°C	-5%	2MHz	+5%	MHz
Dev4M	4MHz工作频率时的偏差	Fmain = 4MHz VCC = 2.0v ~ 3.8v, -40°C to 85°C	-2.5%	4MHz	+2.5%	MHz
		Fmain = 4MHz VCC = 1.8v ~ 3.8v, -40°C to 85°C	-5%	4MHz	+5%	MHz
Dev8M	8MHz工作频率时的偏差	Fmain = 8MHz VCC = 2.0v ~ 3.8v, -40°C to 85°C	-2.5%	8MHz	+2.5%	MHz
		Fmain = 8MHz VCC = 1.8v ~ 3.8v, -40°C to 85°C	-5%	8MHz	+5%	MHz
Dev16M	16MHz工作频率时的偏差	Fmain = 16MHz VCC = 2.0v ~ 3.8v, -40°C to 85°C	-2.5%	16MHz	+2.5%	MHz
		Fmain = 16MHz VCC = 1.8v ~ 3.8v, -40°C to 85°C	-5%	16MHz	+5%	MHz

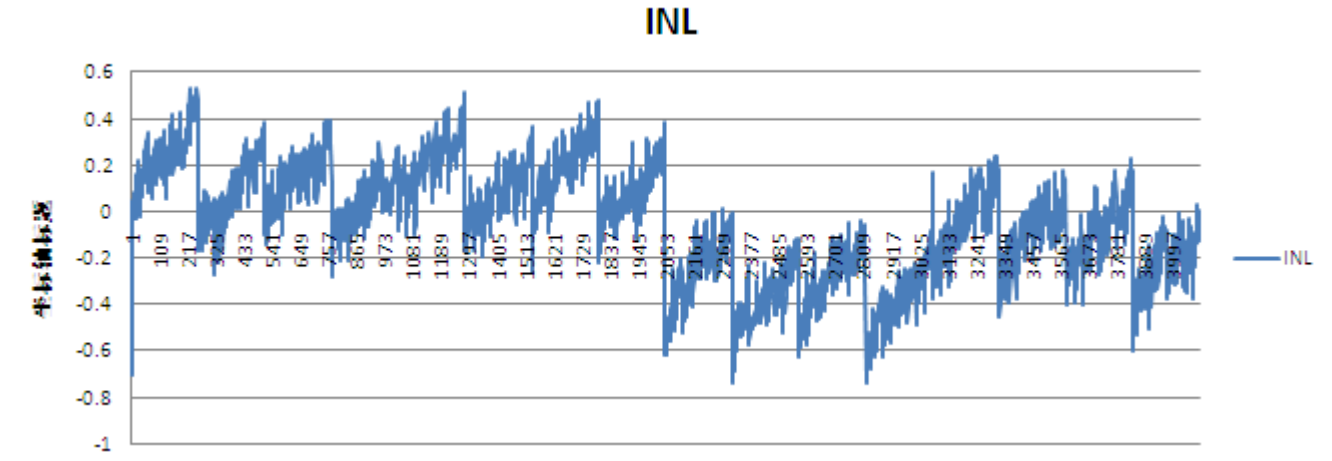
33.13 内部 32KHz 时钟

符号	参数	条件	最小	标准	最大	单位
DCmain	占空比		48	50	52	%
IFmain	电流消耗	Fmain = 8MHz		0.2		μA
Dev2M	当主时钟为2MHz时，内部32K时钟的偏差值	Fmain = 2MHz VCC = 2.0v ~ 3.8v, -40°C to 85°C	-20%	32.0	+20%	KHz

33.14 模数转换器(ADC)

符号	参数	条件	最小	标准	最大	单位
Vadcin	外部参考电压的输入范围	25°C	0		VCC	V
Vadccm	模拟输入端电压范围	25°C	0		VCC	V
I(ADC)	工作电流（使能参考电压）	200KSamples/s, 12bit, VCC=3.6v		3.5		mA
		200KSamples/s, 12bit, VCC=3.0v		3.0		mA
		200KSamples/s, 12bit, VCC=2.2v		2.5		mA
C(ADCI)	ADC 输入电容	VCC=3.0v		10		Pf
Fadcclk	ADC 时钟频率	Fsystem = 4MHz	512K	4M	8M	Hz
Tadcstart	内部参考电压建立时间与 ADC 建立时间		25	30		µS
Tadcconv	转换周期		17		25	cycles
SNRadc	信噪比	200K 次/秒采样率，12 位精度，参考电压为内部 2.5V		60		dB
SNDRadc	信号-噪声失调比（SNDR）	200K 次/秒采样率，12 位精度，参考电压为内部 2.5V		60		dB
SFDRadc	无杂波动态范围（SFDR）	200K 次/秒采样率，12 位精度，参考电压为内部 2.5V		60		dB
VADCoffset	偏置电压			10		mV
DNLadc	差分非线性失真（DNL）	内部 1.5V 做参考电压		2		LSB
		内部 2.5V 做参考电压		2		LSB
		VCC 做参考电压		2		LSB
INLadc	积分非线性失真（INL）	内部 1.5V 做参考电压		4		LSB
		内部 2.5V 做参考电压		4		LSB
		VCC 做参考电压		4		LSB
Eo	偏置误差				+/- 4	LSB
Eg	增益误差	Unbuffered external reference			+/- 4	LSB
Et	绝对误差				+/-5	LSB
Mcadc	丢失码	无丢失码		0		code
Vref25	内部 2.5V 参考电压	25°C，出厂已校准	2.475	2.5	2.525	V
Vref15	内部 1.5V 参考电压	25°C，未校准	1.4	1.5	1.6	V


DNL
LSB
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
DNL


INL
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
INL

33.15 内置温度传感器

符号	参数	条件	VCC(V)	最小	标准	最大	单位
Isensor	温度传感器工作电流	在 25°C 环境下	3.0		30		μA
TCsensor				3.1	3.2	3.3	mV/°C
Voffset	传感器偏置电压			-100		100	mV
Vsensor	传感器输出电压	在 85°C 环境下	2.6	1200	1250	1300	mV
			3.0	1210	1260	1310	mV
			3.6	1220	1270	1320	mV
		在 25°C 环境下	2.6	1020	1070	1120	mV
			3.0	1030	1080	1130	mV
			3.6	1040	1090	1140	mV
		在-40°C 环境下	2.6	800	850	900	mV
			3.0	810	860	910	mV
			3.6	820	870	920	mV
Tsensor	传感器采样时间			80	90		μS

注意事项1：下面的公式可以用来计算温度传感器的输出电压

VSensor,typ = TCSensor (273 + T [°C]) + VOffset,sensor [mV] 或

VSensor,typ = TCSensor T [°C] + VSensor(TA = 0C) [mV]

华大会把温度传感器在常温下(25摄氏度+/-3摄氏度)测试值写入CAL2寄存器的[11：0]位，ADC使用校准后2.5V内部参考电压采样数据。

33.16 模拟电压比较器（VC）

符号	参数	条件	最小	标准	最大	单位
Vvcmpin	输入电压范围		0		VCC	V
Vvcmpcm	外部基准电压输入范围		0		VCC	V
Ivcmp1	VC操作电流1	与内部BGR产生的基准电压做比较		30		μA
Ivcmp2	VC操作电流2	与外部电压源的基准电压做比较(内部BGR关闭)		5		μA
Tvcstart	参考电压建立时间			30		μS
Vvcmoff	VC偏置电压	单端		10		mV
		差分		10		mV
Vvcmhyst	迟滞电压			20		mV
Tfilter	比较器滤波时间	VC_response = 000		16		μS
		VC_response = 001		32		μS
		VC_response = 010		64		μS
		VC_response = 011		256		μS
		VC_response = 100		1		mS
		VC_response = 101		4		mS
		VC_response = 110		16		mS
		VC_response = 111		64		mS
Vcomp	比较电压	VC1[7:4]=1101	2.40	2.50	2.60	V
		VC1[7:4]=1110	2.51	2.61	2.71	V
		VC1[7:4]=1111	2.62	2.72	2.82	V
		VC1[7:4]=0000	2.73	2.83	2.93	V
		VC1[7:4]=0001	2.84	2.94	3.04	V
		VC1[7:4]=0010	2.95	3.05	3.15	V
		VC1[7:4]=0011	3.06	3.16	3.26	V
		VC1[7:4]=0100	3.17	3.27	3.37	V
		VC1[7:4]=0101	3.28	3.38	3.48	V
		VC1[7:4]=0110	3.39	3.49	3.59	V
		VC1[7:4]=0111	3.50	3.60	3.70	V
		VC1[7:4]=1000	3.61	3.71	3.81	V
		VC1[7:4]=1001	3.72	3.82	3.92	V
		VC1[7:4]=1010	3.83	3.93	4.03	V
		VC1[7:4]=0100	3.94	4.04	4.14	V
		VC1[7:4]=1100	4.05	4.15	4.25	V

33.17 低电压监测（LVD）

符号	参数	条件	最小	标准	最大	单位
Ilvd	LVD操作电流			4		μA
Tvcstart	参考电压建立时间			30		μS
Vlvdoff	LVD偏置电压			10		mV
Vlvdhyst	LVD迟滞电压			20		mV
Tfilter	滤波器响应时间	LVD_response = 000	-10%	16	+10%	μS
		LVD_response = 001	-10%	32	+10%	μS
		LVD_response = 010	-10%	64	+10%	μS
		LVD_response = 011	-10%	256	+10%	μS
		LVD_response = 100	-10%	1	+10%	mS
		LVD_response = 101	-10%	4	+10%	mS
		LVD_response = 110	-10%	16	+10%	mS
		LVD_response = 111	-10%	64	+10%	mS
Vlvd	监测电压门限值	LVDC1[7:4]=0000	1.90	2.00	2.10	V
		LVDC1[7:4]=0001	2.005	2.105	2.205	V
		LVDC1[7:4]=0010	2.11	2.210	2.31	V
		LVDC1[7:4]=0011	2.215	2.315	2.415	V
		LVDC1[7:4]=0100	2.32	2.420	2.52	V
		LVDC1[7:4]=0101	2.425	2.525	2.625	V
		LVDC1[7:4]=0110	2.53	2.630	2.73	V
		LVDC1[7:4]=0111	2.635	2.735	2.835	V
		LVDC1[7:4]=1000	2.74	2.840	2.94	V
		LVDC1[7:4]=1001	2.845	2.945	3.045	V
		LVDC1[7:4]=1010	2.95	3.050	3.15	V
		LVDC1[7:4]=0100	3.055	3.155	3.255	V
		LVDC1[7:4]=1100	3.16	3.260	3.36	V
		LVDC1[7:4]=1101	3.265	3.365	3.465	V
		LVDC1[7:4]=1110	3.37	3.470	3.57	V
		LVDC1[7:4]=1111	3.475	3.575	3.675	V

34 修改记录

版本	日期	记录人	更新主题
0.80	2013-8-12		初稿
1.00	2014-3-15		修改芯片电气参数
1.10	2014-5-12		修改内部 RC OSC 的偏差参数 原来只有：VCC = 2.0v ~ 3.8v, -40° C to 85° C 误差<+/-2.5% 另外增加：VCC = 1.8v ~ 3.8v, -40° C to 85° C 误差<+/- 5.0%
1.20	2014-10-20		增加整个超低功耗系列描述
1.30	2015-08-31		1. 统一电气特性与正文中不对应的参数 2. 删除冗余描述 3. 调整文档格式与结构