CY7C63310，CY7C638xx

enCoRe™II

低速USB外设控制器

特征

- USB 2.0-USB-IF认证（TID# 40000085）
- enCoRe™ II USB - ‘enhanced Component Reduction’
  - 无 晶振，支持外部时钟. 该内部振荡器消除了对外部晶体的需求或谐振器.
  - 两个内部3.3 V稳压器和一个内部USB上拉电阻
  - 无需外部设备，即可实现真实世界界面的可配置I / O组件
- 符合USB规范
  - 符合USB规格2.0版
  - 符合USB HID规范，版本1.1
  - 支持一个低速USB设备地址
  - 支持一个控制端点和两个数据端点
  - 带专用3.3 V稳压器的集成USB收发器USB信号和D-上拉.
- 增强型8位微控制器
  - 哈佛建筑
  - M8C CPU的速度高达24 MHz或由外部提供时钟信号
- 内部存储器
  - 最多256个字节的RAM
  - 高达8 KB的闪存包括EEROM仿真
- 接口可以自动配置为PS / 2或USB
  - 没有用于在PS / 2和PS / 2之间切换的外部组件USB模式
  - 无需管理通用I / O（GPIO）引脚双模式能力
- 低功耗
  - 通常在6 MHz时为10 mA
  - 10µA的 睡眠
- 在系统可重编程性中：
  - 允许轻松更新固件
- GPIO端口
  - 最多20个GPIO引脚

  所有GPIO引脚上均为2 mA源电流. 可配置8或指定引脚上的50 mA /引脚电流接收器.
  - 每个GPIO端口都支持高阻抗输入，可配置的上拉，开漏输出，CMOS / TTL输入，和CMOS输出

  all所有I / O引脚上的可屏蔽中断
- 用于USB PHY的专用3.3 V稳压器. 有助于发信号和D-线上拉
- 125 mA 3.3 V稳压器为外部3.3 V器件供电
- 3.3个VI / O引脚
  - 具有3.3 V逻辑电平的4个I / O引脚
  - 每个3.3 V引脚均支持高阻抗输入，内部上拉，开漏输出或传统的CMOS输出
- SPI串行通信
  - 主或从属操作
  - 在主 设备中 最多可配置4 Mbit /秒传输模式
  - 支持光学传感器的半双工单数据线模式
- 2通道8位或1通道16位捕捉定时器寄存器. 捕捉定时器寄存器存储上升沿和下降沿时间.
  - 两个寄存器分别用于两个输入引脚
  - 单独的寄存器用于上升沿和下降沿捕捉
  - 简化无线应用的射频输入接口
- 暂停模式下的内部低功耗唤醒定时器：
  - 周期性唤醒，无需外部元件
- 具有中断的12位可编程间隔定时器
- 基于赛普拉斯PSoC®工具的高级开发工具
- 看门狗定时器（WDT）
- 具有用户可配置阈值的低电压检测电压
- 工作电压从4.0 V到5.5 V DC
- 工作温度从0-70°C
- 提供16引脚和18引脚PDIP; 16,18和24引脚SOIC; 24引脚QSOP封装和32引脚QFN封装
- 行业标准的程序员支持

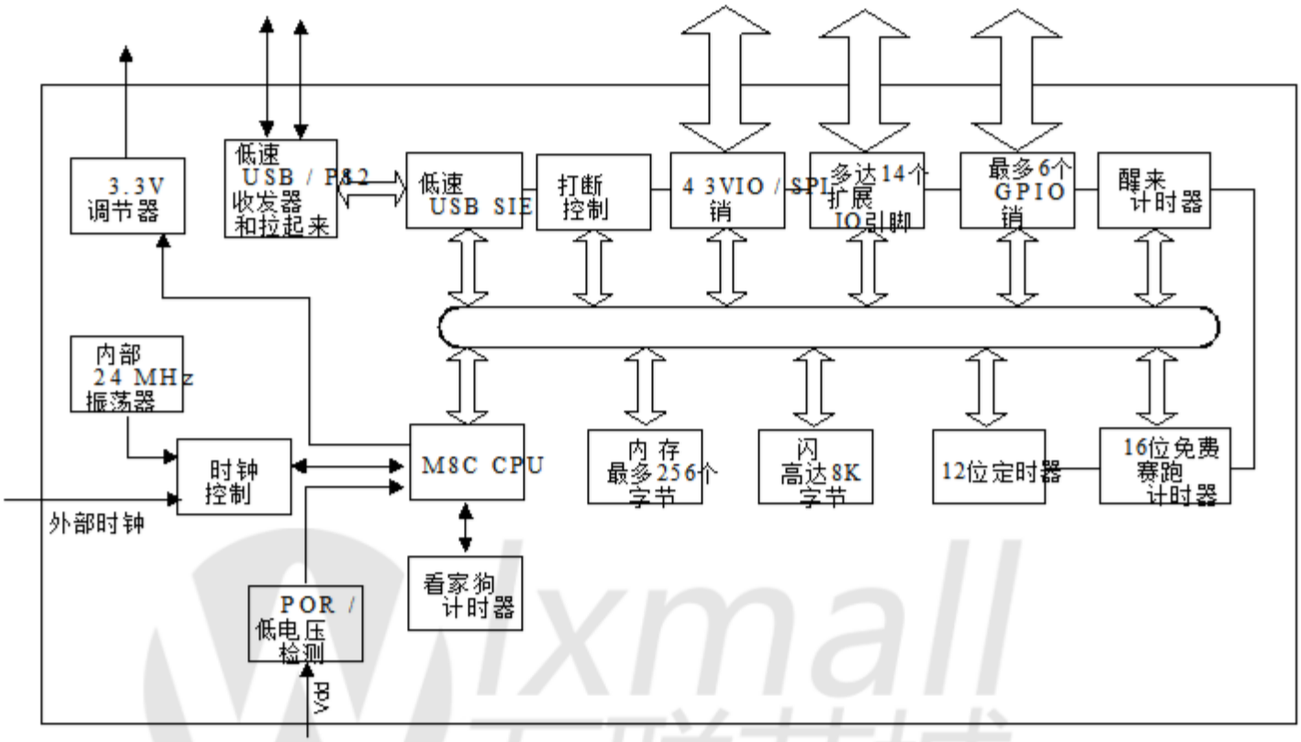
0.1应用

CY7C63310 / CY7C638xx适用于以下应用
应用：

- PC HID设备
  - 鼠标（光机械，光学，轨迹球）
- 游戏
  - 操纵杆
  - 游戏手柄
- 通用
  - 条形码扫描仪
  - POS 终端
  - 消费电子产品
  - 玩具
  - 遥控器
  - 安全加密狗

内容

逻辑框图



3.介绍

赛普拉斯已经在低速下重新发挥其领导地位
USB市场与一个新的创新微控制器系列.
Introducing enCoRe II USB - 'enhanced Component Reduction.'
赛普拉斯利用其在USB解决方案方面的设计专长
推进其低速USB微控制器系列，其中
使外围开发人员能够用a设计新产品
最少数量的组件. enCoRe II USB
技术建立在enCoRe系列之上. enCoRe家族拥有
一个集成的振荡器，消除外部晶体或
谐振器，降低总体成本.还集成到这个芯片中
低速USB中常见的其他外部组件
应用，如上拉电阻，唤醒电路和a
3.3V稳压器.集成这些组件可以减少
整体系统成本.

enCoRe II是一款8位闪存可编程微控制器
带有一个集成的低速USB接口.指令集
虽然，它专门针对USB和PS / 2操作进行了优化
微控制器可以用于各种其他嵌入式设备
应用.

enCoRe II具有多达20个GPIO引脚来支持USB，
PS / 2和其他应用程序. IO引脚分为四组
端口（端口0到3）.端口0和端口1上的引脚可能都是
单独配置，而端口2和3上的引脚是单独配置的
仅配置为一个组.每个GPIO端口都支持高电平
阻抗输入，可配置的上拉，开漏输出，
CMOS / TTL输入和CMOS输出，最多五个引脚
支持高达50 mA接收器的可编程驱动强度
当前. GPIO端口1具有四个可在电压下连接的引脚
3.3V的电平.此外，每个IO引脚都可用于生成
一个GPIO中断给微控制器.每个GPIO端口都有它的
拥有GPIO中断向量;另外，GPIO端口0有三个
具有独立中断向量的专用引脚（P0.2 -
P0.4）.

enCoRe II具有内部振荡器.随着存在
的USB流量，内部振荡器可能被设置为精确调谐
到USB时序要求（24 MHz±1.5％）.可选地，一个
外部12 MHz或24 MHz时钟用于提供更高的频率
USB操作的精密参考.时钟发生器
提供仍然在内部的12 MHz和24 MHz时钟
微控制器. enCoRe II还有一个12位程序 -
mable间隔定时器和一个16位自由运行定时器
捕捉定时器寄存器此外，enCoRe II还包括a
看门狗定时器和向量中断控制器.

enCoRe II拥有多达8 KB的用户代码和闪存
用于堆栈空间和用户变量的最多256字节的RAM.

上电复位电路在上电时检测逻辑
到设备，将逻辑重置为已知状态，然后开始
在闪存地址0x0000处执行指令.当权力
低于可编程的跳闸电压，它会产生复位或
可被配置为产生中断.有一个低
电压检测电路，用于检测V CC何时 降至a以下
可编程跳闸电压.它可以配置为生成LVD
中断以通知处理器低电压事件.
POR和LVD共享相同的中断.没有单独的
每个中断.看门狗定时器可用于确保
固件永远不会陷入无限循环.

微控制器支持22个可屏蔽中断
矢量中断控制器.中断源包括一个USB总线
复位，LVR / POR，可编程间隔定时器，1.024毫秒
自由运行定时器的输出，三个USB端点，两个
捕捉定时器，四个GPIO端口，三个端口0引脚，两个SPI，a
16位自由运行定时器包装，一个内部睡眠定时器和一个总线
活动中断.睡眠定时器在导致周期性中断时
启用. USB事务后USB端点中断
完成在公共汽车上.捕获定时器在新的时候中断
由于选定的GPIO边沿事件，定时器值被保存.一个
共有七个GPIO中断支持TTL或CMOS
阈值.在边缘敏感的GPIO上提供更多的灵活性
引脚，中断极性被编程为上升或下降.

自由运行的16位定时器提供两个中断源：
1.024 ms输出和自由运行计数器回绕中断.
可编程间隔定时器提供最多1个 微秒
分辨率，并在每次到期时提供中断.这些
定时器用于测量一个事件的持续时间
通过在开始和在读取期望的定时器来进行固件控制
一个事件的结束，然后计算之间的差异
两个值.两个8位捕捉定时器寄存器保存a
GPIO时，可编程8位范围的自由运行定时器
边沿出现在两个捕捉引脚（P0.5，P0.6）上.这两个8位
捕捉可能会被集成到一个单一的16位捕捉.

enCoRe II包含一个集成的USB串行接口
引擎（SIE），可让芯片轻松连接至USB
主办.硬件支持三个USB设备地址
端点.

USB D +和D-引脚可选择用作PS / 2 SCLK和
SDATA信号，以便产品能够响应
USB或PS / 2操作模式. PS / 2操作是
内部5 K支持 Ω上拉电阻在P1.0（D +）和
P1.1（D-），以及一个中断来表示PS / 2活动的开始.在
USB模式，集成1.5 K D上的Ω上拉电阻可能是
受固件控制.没有外部组件
双USB和PS / 2系统所必需的，并且没有GPIO引脚
需要致力于切换模式.

enCoRe II通过使用D +支持系统编程
和D-引脚作为串行编程模式接口.该
编程协议不是USB.

4.公约

在这个数据手册中，寄存器中的位位置用阴影表示
表明enCoRe II系列的哪些成员实施了该项目
位.

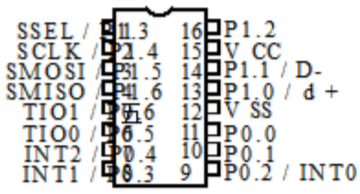
	可在所有enCoRe II家庭成员中使用
	仅CY7C638（1/2/3）3

5.引脚

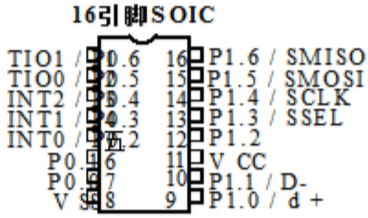
图5-1.引脚图

顶视图

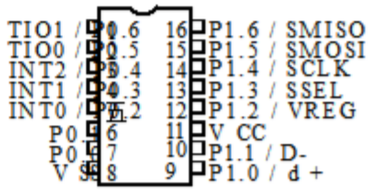
CY7C63801，CY7C63310
16引脚PDIP



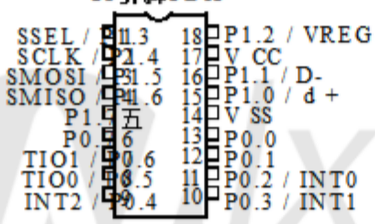
CY7C63801，CY7C63310
16引脚SOIC



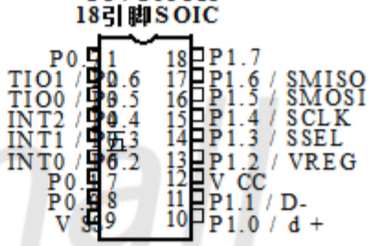
CY7C63803
16引脚SOIC



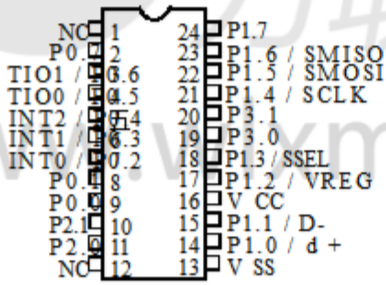
CY7C63813
18引脚PDIP



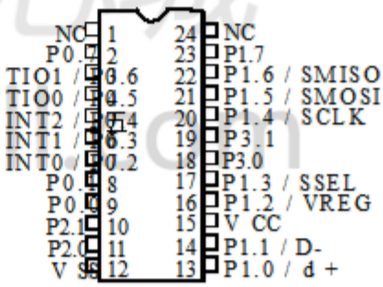
CY7C63813
18引脚SOIC



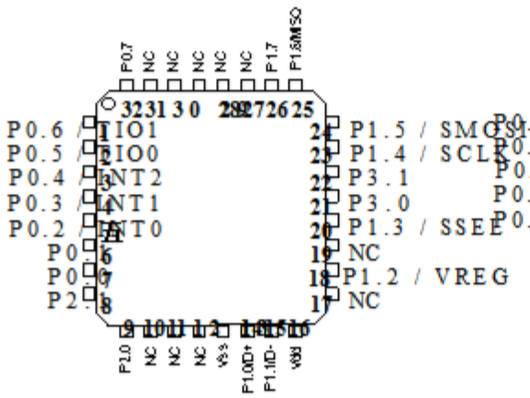
CY7C63823
24引脚QSOP



CY7C63823
24引脚SOIC



CY7C63833　32引脚QFN



CY7C63833 32引脚锯开QFN

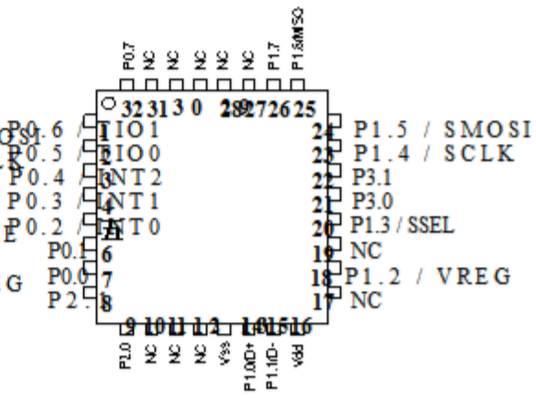
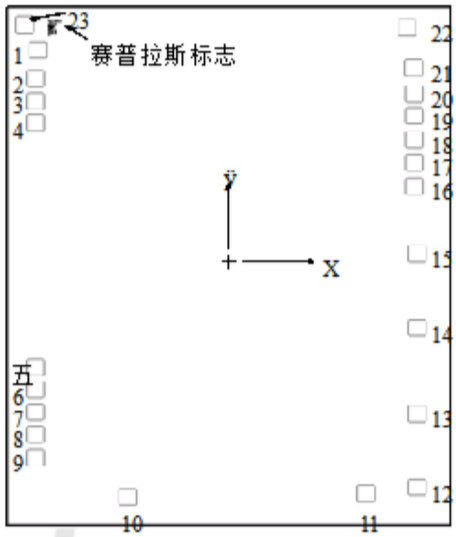


图5-2. CY7C63823模具形式



传说

模台阶=1792.98µm×2272.998µm
芯片尺寸=1727µm×2187µm
焊盘开口= 70微米×70微米
模具厚度= 14密耳

表5-1. Die Pad摘要

垫号	填充名称	X（微米）	Y（微米）
1	P0.7	-742.730	911.990
2	P0.6	-755.060	792.200
3	P0.5	-755.060	699.300
4	P0.4	-755.060	606.400
五	P0.3	-755.060	-430.080
6	P0.2	-755.060	-522.980
7	P0.1	-755.060	-618.830
8	P0.0 CLKIN	-755.060	-714.020
9	P2.1	-755.060	-810.220
10	P2.0	-393.580	-977.930
11	VSS	537.500	-964.700
12	PI.0 D +	736.110	-936.680
13	P1.1 D-	736.110	-625.130
14	V D D	736.110	-260.670
15	P1.2 VREG	736.110	53.800
16	P1.3	723.510	336.780
17	P3.0	723.510	438.690
18	P3.1	723.510	532.880
19	P1.4	723.510	635.310
20	P1.5 SMOSI	723.510	728.220
21	P1.6 SMISO	723.510	839.290
22	P1.7	696.630	1008.480
23	保留的	-795.400	1023.270

表5-2.引脚说明

32 QFN	24 QSOP	24 SOIC	18 SIOC	18 PDIP	16 SOIC	16 PDIP	名称	描述
21	19	18	-	-	-	-	P3.0	**GPIO端口3** .配置为一个组（字节）.
22	20	19	-	-	-	-	P3.1	
9 1	1	11	-	-	-	-	P2.0	**GPIO端口2** .配置为一个组（字节）.
8 1	0	10	-	-	-	-	P2.1	
14	14	13	10	15	9	13	P1.0 / d +	GPIO端口1位0 / USB D + [1如果此引脚用作a 通用输出，它吸取电流.这个引脚 必须配置为输入以减少电流 画.
15	15	14	11	16	10	14	P1.1 / D-	**GPIO端口1位1 / USB D- [1]** 如果此引脚用作a 通用输出，它吸取电流.这个引脚 必须配置为输入以减少电流 画.
18	17	16	13	18	12	16	P1.2 / VREG	GPIO端口1位2.单独配置. 3.3V如果稳压器启用.（3.3 V稳压器不是 可在CY7C63310和CY7C63801中获得.）1- µF min，2Vreg输出需要µF最大电容.
20	18	17	14	1	13	1	P1.3 / SSEL	GPIO端口1位3.单独配置. 备用功能是SPI总线TTL的SSEL信号 电压阈值.虽然Vreg不可用 与CY7C63310一样，3.3 VI / O仍然可用.
23	21	20	15	2	14	2	P1.4 / SCLK	GPIO端口1位4.单独配置. 备用功能是SPI总线TTL的SCLK信号 电压阈值.虽然Vreg不可用 与CY7C63310一样，3.3 VI / O仍然可用.
24	22	21	16	3	15	3	P1.5 / SMOSI	GPIO端口1位5.单独配置. 备用功能是SPI总线TTL的SMOSI信号 电压阈值.虽然Vreg不可用 与CY7C63310一样，3.3 VI / O仍然可用.
25	23	22	17	4	16	4	P1.6 / SMISO	GPIO端口1位6.单独配置. 备用功能是SPI总线TTL的SMISO信号 电压阈值.虽然Vreg不可用 与CY7C63310一样，3.3 VI / O仍然可用.
26	24	23	18	五	-	-	P1.7	**GPIO端口1位7** .单独配置. TTL电压阈值.
7	9	9	8	13	7	11	P0.0	**GPIO端口0位0** .单独配置. 在CY7C638xx和CY7C63310上，外部时钟 输入时配置为时钟输入.
6	8	8	7	12	6	10	P0.1	**GPIO端口0位1** .单独配置. 在CY7C638xx和CY7C63310上，时钟输出为 配置为时钟输出.
57		7	6	11	五	9	P0.2 / INT0	GPIO端口0位2.单独配置. 可选的上升沿中断INT0.
46		6	五	10	4	8	P0.3 / INT1	GPIO端口0位3.单独配置. 可选的上升沿中断INT1.
3	五	五	4	9	3	7	P0.4 / INT2	GPIO端口0位4.单独配置. 可选的上升沿中断INT2.

注意
1.当用作GPIO和I sb模式时，P1.0（D +）和P1.1（D-）引脚必须处于I / O模式.

表5-2.引脚说明 （继续）

32 QFN	24 QSOP	24 SOIC	18 SIOC	18 PDIP	16 SOIC	16 PDIP	名称	描述
2	4	4	3	8	2	6	P0.5 / TIO0	GPIO端口0位5.单独配置 备用功能定时器捕捉输入或定时器 输出TIO0
1	3	3	2	7	1	五	P0.6 / TIO1	GPIO端口0位6.单独配置 备用功能定时器捕捉输入或定时器 输出TIO1
32	2	2	1	6	-	-	P0.7	**GPIO端口0位7** .单独配置 不在16引脚PDIP或SOIC封装中
10	1	1	-	-	-	-	NC	**没有连接**
11	12	24	-	-	-	-	NC	**没有连接**
12	-	-	-	-	-	-	NC	**没有连接**
17	-	-	-	-	-	-	NC	**没有连接**
19	-	-	-	-	-	-	NC	**没有连接**
27	-	-	-	-	-	-	NC	**没有连接**
28	-	-	-	-	-	-	NC	**没有连接**
29	-	-	-	-	-	-	NC	**没有连接**
三十	-	-	-	-	-	-	NC	**没有连接**
31	-	-	-	-	-	-	NC	**没有连接**
16	16	15	12	17	11	15	VCC	**供应**
13	13	12	9	14	8	12	V SS	**地面**

6. CPU架构

该系列微控制器基于高性能，
8位哈佛架构微处理器.五个寄存器控制
CPU核心的主要操作.这些寄存器是
受各种指令影响，但不能直接访问
通过用户的注册空间.

表6-1. CPU寄存器和寄存器名称

CPU寄存器	注册名称
旗	CPU_F
程序计数器	CPU_PC
累加器	CPU_A
堆栈指针	CPU_SP
指数	CPU_X

16位程序计数器寄存器（CPU_PC）允许直接访问
寻址全部8 KB的程序存储空间.

累加器寄存器（CPU_A）是通用的
寄存器，它保存指定任何指令的结果
的源寻址模式.

索引寄存器（CPU_X）保存使用的偏移值
在索引寻址模式下.通常，这用于
寻址数据存储空间内的数据块.

堆栈指针寄存器（CPU_SP）保存地址
数据存储空间中当前堆栈的顶端.它受到影响
通过PUSH，POP，LCALL，CALL，RETI和RET指令，
管理软件堆栈.它也受到SWAP的影响
和ADD指令.

标志寄存器（CPU_F）有三个状态位：零标志位
[1];进位标志位[2];监督状态位[3].全球
中断允许位[0]全局启用或禁用中断.
用户不能操纵监控状态位[3].
这些标志受算术，逻辑和移位操作的影响.
每个标志被改变的方式取决于
正在执行的指令，如AND，OR，XOR和
其他.请参阅第13页上的表 8-1.

7. CPU寄存器

enCoRe II器件中的CPU寄存器位于两个存储区中，每个存储区有256个寄存器. CPU标志寄存器中的位[4] / XI / O位必须置位/清零以在两个寄存器组之间进行选择，请参阅第9页上的表7-1

7.1标志注册

标志寄存器只能用逻辑指令设置或复位.

表7-1. CPU标志寄存器（CPU_F）[R / W]

位#	7	6	五	4	3	2	1	0
领域	保留的			XIO	超	携带	零	全球IE
读/写	-	-	-	R / W	[R	RW	RW	RW
默认	00000010							

位[7：5]:保留的

位4： XIO

由用户设置以在寄存器组之间进行选择

0 = Bank 0

1 =银行1

位3： 超

指示CPU是执行用户代码还是执行代码.（此代码不能直接由用户访问.）

0 =用户代码

1 =主管代码

位2： 携带

由CPU设置以指示在前一个逻辑/算术运算中是否有进位.

0 =没有携带

1 =携带

位1： 零

由CPU设置以指示在前一个逻辑/算术运算中是否存在零结果.

0 =不等于零

1 =等于零

位0： 全球IE

确定是否启用或禁用所有中断

0 =禁用

1 =启用

注意 CPU_F寄存器只能用明确的寄存器地址0xF7读取. OR F，expr和AND F，expr指令必须用于设置和清除CPU_F位.

表7-2. CPU累加器寄存器（CPU_A）

位#	7	6	五	4	3	2	1	0
领域	CPU累加器[7：0]							
读/写	--------							
默认	00000000							

位[7：0]:CPU累加器[7：0]

8位数据值保存使用源寻址模式的任何逻辑/算术指令的结果

表7-3. CPU X寄存器（CPU_X）

位#	7	6	五	4	3	2	1	0
领域	X [7：0]							
读/写	--------							
默认	00000000							

位[7：0]:X [7：0]

8位数据值包含任何使用索引寻址模式的指令的索引.

表7-4. CPU堆栈指针寄存器（CPU_SP）

位#	7	6	五	4	3	2	1	0
领域	堆栈指针[7：0]							
读/写	--------							
默认	00000000							

位[7：0]:堆栈指针[7：0]

8位数据值包含一个指向栈顶的指针.

表7-5. CPU程序计数器高位寄存器（CPU_PCH）

位#	7	6	五	4	3	2	1	0
领域	程序计数器[15：8]							
读/写	--------							
默认	00000000							

位[7：0]:程序计数器[15：8]

8位数据值保存程序计数器的高字节.

表7-6. CPU程序计数器低寄存器（CPU_PCL）

位#	7	6	五	4	3	2	1	0
领域	程序计数器[7：0]							
读/写	--------							
默认	00000000							

位[7：0]:程序计数器[7：0]

8位数据值保存程序计数器的低字节.

7.2寻址模式

7.2.1来源即时

放置使用该寻址模式的指令的结果
在A寄存器，F寄存器，SP寄存器或X寄存器中，
它被指定为指令操作码的一部分.操作数1
是作为教学来源的直接价值.
算术指令需要两个来源;第二个来源是
操作码中指定的A或X寄存器.使用说明
这种寻址模式的长度是两个字节.

表7-7.来源即时

操作码	操作数1
指令	立即价值

例子

加	一个	7	7的即时值加上 累加器和结果放置在 累加器.
MOV	X	8	8的直接值被移动到X. 寄存器.
和	F	9	9的直接值与逻辑与 F寄存器和结果放在F中 寄存器.

7.2.2 来源直接

放置使用该寻址模式的指令的结果
在指定为A部分的A寄存器或X寄存器中
的指令操作码.操作数1是指向的地址
RAM存储空间或寄存器空间中的位置
是指令的来源.算术指令要求
两个来源;第二个来源是A寄存器或X寄存器
在操作码中指定.使用这种寻址模式的指令
是两个字节的长度.

表7-8.源代码直接

操作码	操作数1
指令	来源地址

例子

加	一个	[7]	RAM存储器中的值位于 地址7被添加到累加器， 结果放在Accumu- lator.
MOV	X	REG [8]	地址寄存器空间中的值 8被移动到X寄存器.

7.2.3 来源索引

放置使用该寻址模式的指令的结果
在指定为A部分的A寄存器或X寄存器中
的指令操作码.操作数1被添加到X寄存器
形成指向RAM存储器中的位置的地址
空间或作为指令来源的寄存器空间.
算术指令需要两个来源;第二个来源是
操作码中指定的A寄存器或X寄存器.说明
使用这种寻址模式的长度是两个字节.

表7-9.来源索引

操作码	操作数1
指令	来源索引

例子

加	一个	[X + 7]	内存位置处的值 地址X + 7添加了 累加器，并放置结果 在累加器中.
MOV	X	REG [X + 8]	寄存器空间中的值为 地址X + 8被移动到X 寄存器.

7.2.4 目的地直接

放置使用该寻址模式的指令的结果
在RAM存储器空间或寄存器空间内.操作数1
是指向结果位置的地址.来源
因为该指令是A寄存器或X寄存器，
被指定为指令操作码的一部分.算术指令
tions需要两个来源;第二个来源是位置
操作数1指定.使用此寻址模式的指令
是两个字节的长度.

表7-10.目的地直接

操作码	操作数1
指令	目的地地址

例子

加	[7]	一个	内存位置处的值 地址7与Accumu- lator，结果被放置在 地址7处的内存位置 累加器不变.
MOV	REG [8]	一个	累加器被移动到 在地址8注册空间位置. 累加器不变.

7.2.5 目的地索引

放置使用该寻址模式的指令的结果
在RAM存储器空间或寄存器空间内.操作数1
被添加到X寄存器中形成指向该地址的地址
结果的位置.该指令的来源是A
寄存器.算术指令需要两个来源;第二
源是由X添加的操作数1指定的位置
寄存器.使用这种寻址模式的指令是两个字节
在长度.

表7-11.目的地索引

操作码	操作数1
指令	目的地索引

例

ADD	[X + 7]	一个	价值在;内存位置 地址X + 7与Accumu- lator，结果被放置在 内存位置在地址x + 7.该 累加器不变.

7.2.6 目标直接源即时

放置使用该寻址模式的指令的结果
在RAM存储器空间或寄存器空间内.操作数1
是结果的地址.指令的来源是
操作数2，它是一个立即值.算术指令
需要两个来源;第二个来源是指定的位置
通过操作数1.使用这种寻址模式的指令有三种
字节长度.

表7-12.目的地直接来源直接

操作码	操作数1	操作数2
指令	目的地地址	立即价值

例子

ADD	[7]	五	地址内存位置的值 7被添加到5的立即值，并且 结果放在内存位置 地址7.
MOV	REG [8]	6	6的直接值被移入 在地址8注册空间位置.

7.2.7 目标索引源即时

放置使用该寻址模式的指令的结果
在RAM存储器空间或寄存器空间内.操作数1
被添加到X寄存器以形成结果的地址.该
指令的来源是操作数2，它是一个直接的
值.算术指令需要两个来源;第二
源是由X添加的操作数1指定的位置
寄存器.使用这种寻址模式的指令是三个字节
在长度.

表7-13.目的地索引源即时

操作码	操作数1	操作数2
指令	目的地索引	即刻的价值

例子

加	[X+7]	五	内存位置处的值 地址X + 7添加了 立即值5，结果 放在内存位置 地址X + 7.
MOV	REG [X + 8]	6	6的直接值被移动 进入寄存器空间的位置 在地址X + 8.

7.2.8 目标直接源直接

放置使用该寻址模式的指令的结果
在RAM内存中.操作数1是结果的地址.
操作数2是指向RAM中某个位置的地址
内存是指令的来源.这个寻址
模式仅在MOV指令中有效.使用说明
这种寻址模式的长度是三个字节.

表7-14.目的地直接来源直接

操作码	操作数1	操作数2
指令	目的地地址	来源地址

例

MOV	[7]	[8]	地址8处的内存位置中的值 被移动到地址7的存储位置.

7.2.9 源间接递增

放置使用该寻址模式的指令的结果
在累加器中.操作数1是指向一个地址的地址
在内存空间中的位置，其中包含一个地址
（间接地址）作为指令的来源.该
间接地址作为指令的一部分递增
执行.此寻址模式仅在MVI上有效
指令.使用这种寻址模式的指令是两个
字节长度.请参考*PSoC Designer：Assembly Language User Guide*了解MVI指令的更多细节.

表7-15.源间接后期增量

操作码	操作数1
指令	源地址地址

例

MVI	一个	[8]	地址内存位置的值 8是间接地址.内存位置 由间接地址指向的移动 进入累加器.间接地址是 然后递增.

7.2.10 目的地间接递增

放置使用该寻址模式的指令的结果
在内存空间内.操作数1是指向一个地址的地址
在内存空间中的位置，其中包含一个地址
（间接地址）作为指令的目的地.该
间接地址作为指令的一部分递增
执行.指令的来源是累加器.这个
寻址模式仅在MVI指令中有效.该
使用这种寻址模式的指令的长度是两个字节.

表7-16.目的地间接递增

操作码	操作数1
指令	目的地址地址

例

MVI	[8]	一个	内存位置处的值 地址8是一个间接地址.该 累加器被移入内存 间接指向的位置 地址.那么间接地址就是 递增.

笔记
2.在中断向量表恢复执行之前，中断例程需要13个周期.
3.对于闪存空间中跨越256字节边界的指令，指令所需的周期数增加1.

8.指令集汇总

表8-1以数字形式总结了指令集，并作为快速参考.如果需要更多信息，指令集汇总表在“PSoC Designer汇编语言用户指南”中详细介绍（可在赛普拉斯网站http://www.cypress.com）.

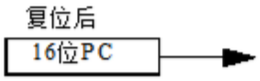
表8-1.指令集汇总按操作码顺序数字排序[2,3]

Opcode Hex	Cycles	Bytes	指令格式	旗	Opcode Hex	Cycles	Bytes	指令格式	旗	Opcode Hex	Cycles	Bytes	指令格式	旗
00	15	1	个SSC	-	2D	8	2	或[X + expr]，A	ž	5A	五	2	MOV [expr]，X	-
01	4	2	ADD A，expr	C，Z	2E	9	3	或[expr]，expr	ž	5B	4	1	MOV A，X	ž
02	6	2	ADD A，[expr]	C，Z	2F	10	3	或[X + expr]，expr	ž	5C	4	1	MOV X，A	-
03	7	2	ADD A，[X + expr]	C，Z	三十	9	1	HALT	-	5D	6	2	MOV A，reg [expr]	ž
04	7	2	ADD [expr]，A	C，Z	31	4	2	XOR A，expr	ž	5E	7	2	MOV A，reg [X + expr]	ž
05	8	2	ADD [X + expr]，A	C，Z	32	6	2	XOR A，[expr]	ž	5F	10	3	MOV [expr]，[expr]	-
06	9	3	ADD [expr]，expr	C，Z	33	7	2	XOR A，[X + expr]	ž	60	五	2	MOV reg [expr]，A	-
07	10	3	ADD [X + expr]，expr	C，Z	34	7	2	XOR [expr]，A	ž	61	6	2	MOV reg [X + expr]，A	
08	4	1	PUSH A	-	35	8	2	XOR [X + expr]，A	ž	62	8	3	MOV reg [expr]，expr	-
09	4	2	ADC A，expr	C，Z	36	9	3	XOR [expr]，expr	ž	63	9	3	MOV reg [X + expr]，expr	
0A	6	2	ADC A，[expr]	C，Z	37	10	3	XOR [X + expr]，expr	ž	64	4	1	ASL A	C，Z
0B	7	2	ADC A，[X + expr]	C，Z	38	五	2	ADD SP，expr	-	65	7	2	ASL [expr]	C，Z
0C	7	2	ADC [expr]，A	C，Z	39	五	2	CMP A，expr		66	8	2	ASL [X + expr]	C，Z
0D	8	2	ADC [X + expr]，A	C，Z	3A	7	2	CMP A，[expr]		67	4	1	ASR A	C，Z
0E	9	3	ADC [expr]，expr	C，Z	3B	8	2	CMP A，[X + expr]	如果（A = B），Z = 1 如果（A < B），C = 1	68	7	2	ASR [expr]	C，Z
0F	10	3	ADC [X + expr]，expr	C，Z	3C	8	3	CMP [expr]，expr		69	8	2	ASR [X + expr]	C，Z
10	4	1	PUSH X	-	3D	9	3	CMP [X + expr]，expr		6A	4	1	RLC A	C，Z
11	4	2	SUB A，expr	C，Z	3E	10	2	MVI A，[[expr] ++	ž	6B	7	2	RLC [expr]	C，Z
12	6	2	SUB A，[expr]	C，Z	3F	10	2	MVI [[expr] ++]，A	-	6C	8	2	RLC [X + expr]	C，Z
13	7	2	SUB A，[X + expr]	C，Z	40	4	1	NOP	-	6D	4	1	RRC A	C，Z
14	7	2	SUB [expr]，A	C，Z	41	9	3	和reg [expr]，expr	ž	6E	7	2	RRC [expr]	C，Z
15	8	2	SUB [X + expr]，A	C，Z	42	10	3	AND reg [X + expr]，expr	ž	6F	8	2	RRC [X + expr]	C，Z
16	9	3	SUB [expr]，expr	C，Z	43	9	3	OR reg [expr]，expr	ž	70	4	2	AND F，expr	C，Z
17	10	3	SUB [X + expr]，expr	C，Z	44	10	3	或reg [X + expr]，expr	ž	71	4	2	或F，表达	C，Z
18	五	1	POP A	ž	45	9	3	XOR reg [expr]，expr	ž	72	4	2	XOR F，expr	C，Z
19	4	2	SBB A，expr	C，Z	46	10	3	XOR reg [X + expr]，expr	ž	73	4	1	CPL A	ž
1A	6	2	SBB A，[expr]	C，Z	47	8	3	TST [expr]，expr	ž	74	4	1	INC A	C，Z
1B	7	2	SBB A，[X + expr]	C，Z	48	9	3	TST [X + expr]，expr	ž	75	4	1	INC X	C，Z
1C	7	2	SBB [expr]，A	C，Z	49	9	3	TST reg [expr]，expr	ž	76	7	2	INC [expr]	C，Z
1D	8	2	SBB [X + expr]，A	C，Z	4A	10	3	TST reg [X + expr]，expr	ž	77	8	2	INC [X + expr]	C，Z
1E	9	3	SBB [expr]，expr	C，Z	4B	五	1	SWAP A，X	ž	78	4	1	DEC A	C，Z
1F	10	3	SBB [X + expr]，expr	C，Z	4C	7	2	SWAP A，[expr]	ž	79	4	1	DEC X	C，Z
20	五	1	POP X	-	4D	7	2	SWAP X，[expr]	-	7A	7	2	DEC [expr]	C，Z
21	4	2	AND A，expr	ž	4E	五	1	SWAP A，SP	ž	7B	8	2	DEC [X + expr]	C，Z
22	6	2	AND A，[expr]	ž	4F	4	1	MOV X，SP	-	7C	13	3	LCALL	-
23	7	2	AND A，[X + expr]	ž	50	4	2	MOV A，expr	ž	7D	7	3	LJMP	-
24	7	2	AND [expr]，A	ž	51	五	2	MOV A，[expr]	ž	7E	10	1	RETI	C，Z
25	8	2	AND [X + expr]，A	ž	52	6	2	MOV A，[X + expr]	ž	7F	8	1	RET	-
26	9	3	AND [expr]，expr	ž	53	五	2	MOV [expr]，A	-	8X	五	2	JMP	-
27	10	3	和[X + expr]，expr	ž	54	6	2	MOV [X + expr]，A	-	9x	11	2	打电话	-
28	11	1	ROMX	ž	55	8	3	MOV [expr]，expr	-	斧头	五	2	JZ	-
29	4	2	或A，表达	ž	56	9	3	MOV [X + expr]，expr	-	BX	五	2	JNZ	-
2A	6	2	OR A，[expr]	ž	57	4	2	MOV X，expr	-	CX	五	2	JC	-
2B	7	2	或A，[X + expr]	ž	58	6	2	MOV X，[expr]	-	DX	五	2	JNC	-
2C	7	2	OR [expr]，A	ž	59	7	2	MOV X，[X + expr]	-	防爆	7	2	JACC	-
										Fx	13	2	指数	ž

9.记忆组织

9.1闪存程序存储器组织

图9-1.程序存储器空间与中断向里表

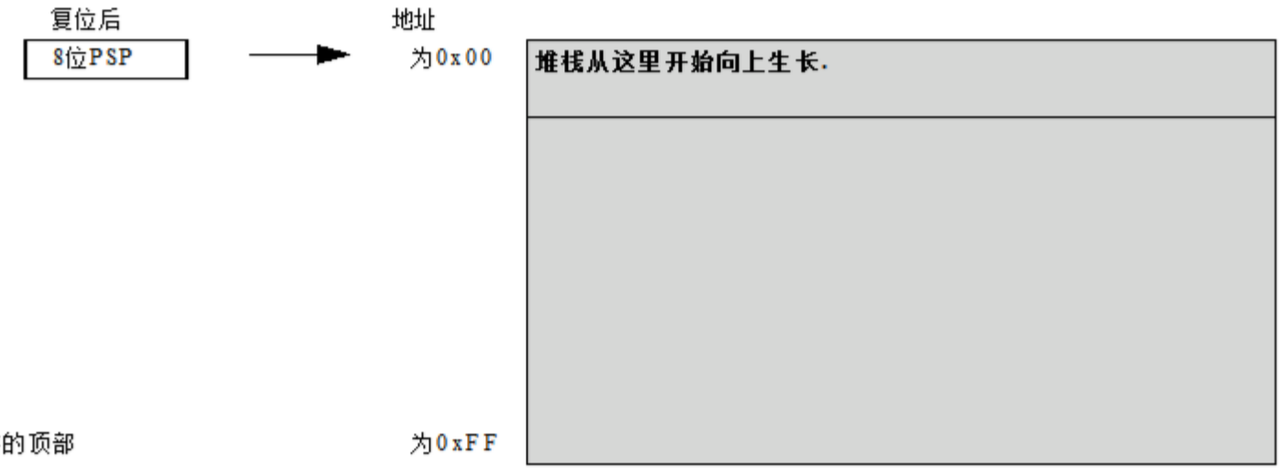


地址	
为0x0000	复位后程序从这里开始执行
0x0004	POR / LVD
×0008	INT0
0x000C	SPI发送器空了
0×0010	SPI接收器已满
0x0014	GPIO端口0
0x0018处	GPIO端口1
0x001C	INT1
0×0020	EP0
0x0024	EP1
0×0028	EP2
0x002C	USB重置
0x0030	USB活动
0x0034	1毫秒间隔计时器
0x0038	可编程间隔定时器
0x003C	定时器捕获0
即0x0040	定时器捕捉1
0x0044	16位自由运行计时器包装
0x0048	INT2
0x004C	PS2数据低
0x0050	GPIO端口2
0x0054	GPIO端口3
0x0058	保留的
0x005C	保留的
0x0060	保留的
0x0064	睡眠定时器
0x0068	**程序存储器从这里开始（如果下面的中断没有被使用，程序内存可以开始降低）**
0x0BFF	**在这里结束3 KB（CY7C63310）**
0x0FFF	**4 KB到此为止（CY7C63801）**
0x1FFF的	**8 KB到此结束（CY7C638x3）**

9.2数据存储器组织

CY7C63310 / 638xx微控制器提供高达256字节的数据RAM.

图9-2.数据存储器组织



9.3 Flash

本节介绍了enCoRe II的闪存模块.大部分
用户可见的闪光功能包括编程和
安全性在M8C Supervisory Read Only中实施
内存（SROM）. enCoRe II闪光灯的续航能力为1000
周期和10年的数据保留能力.

9.3.1闪存编程和安全

所有的Flash编程都由SROM中的代码执行.该
控制闪存编程的寄存器仅对其可见
当M8C CPU执行完SROM时.这使它
不可能通过忽略，写入或擦除闪存
安全机制在SROM中实施.

客户固件只能通过SROM对闪存进行编程
调用.数据或代码图像来源于任何界面
与适当的支持固件.这种类型的编程
requires a 'boot-loader', which is a piece of firmware resident on
闪光.出于安全原因，此引导加载程序不得
在固件重写期间被覆盖.

闪光灯提供四个额外的辅助行，用于保存
闪存块保护标志，启动时间校准值，
配置表和任何设备值.例程
访问这些辅助行记录在本节中
第15页的SROM部分.辅助行不受影响
通过设备擦除功能.

9.3.2在系统编程中

大多数包含enCoRe II部件的设计都有USB
连接器连接到设备上的USB D +和D-引脚.
这些设计需要能够编程或重新编程零件
仅通过USB D +和D-引脚.

enCoRe II设备支持这种类型的系统
通过使用D +和D-引脚作为串行编程
编程模式界面.这允许一个外部控制器
以使enCoRe II部分能够进入串行编程
模式，然后使用测试队列发出闪存访问
在SROM中起作用.编程协议不是USB.

9.4 SROM

SROM包含启动部件的代码，校准电路，
并执行闪光操作（第15页上的表9-1列出了
SROM功能）. SROM的功能可以通过访问
正常的用户代码或从闪存操作. SROM存在
在与用户代码分离的存储空间中. SROM
通过执行监控系统来访问功能
调用指令（SSC），其操作码为00h.之前
执行SSC M8C的累加器必须加载
所需的SROM功能代码来自第15页上的表9-1.
如果用户代码调用未定义的函数，将导致HALT.
SROM函数通过调用执行代码;结果是，
这些功能需要堆栈空间.除重置外，
所有的SROM功能在SRAM中都有一个参数块
必须在执行SSC之前进行配置.表9-2
16列出了所有可能的参数块变量.的意思
关于特定SROM功能的每个参数是
将在本节稍后介绍.

表9-1. SROM功能代码

功能代码	函数名称	堆栈空间
00H	SWBootReset	0
01H	读出数据块	7
02H	WriteBlock	10
03H	EraseBlock	9
05H	删除所有	11
06H	TableRead	3
07H	校验	3

用于所有功能的两个重要变量是KEY1
和KEY2.这些变量用于帮助区分
有效的SSC和无意的SSC之间. KEY1必须始终
具有3 Ah的值，而KEY2必须具有与之相同的值
SROM函数开始执行时的堆栈指针.
这是SSC操作码所在的堆栈指针值
执行，加上三个.如果其中任何一个键不符合
预期值，M8C暂停（除了
SWBootReset函数）.以下代码提供了正确的答案
KEY1和KEY2中的值.代码从停止开始，强制执行
程序直接跳转到设置代码而不会遇到它.

停
SSCOP：mov [KEY1]，3ah
mov X，SP
mov A，X
添加A，3
mov [KEY2]，A

表9-2. SROM功能参数

变量名	SRAM地址
Key1 / Counter / Return Code	0，F8H
KEY2 / TMP	0，F9H
块标识	0，FAH
指针	0，FBH
时钟	0，FCH
模式	0，以及FDh
延迟	0，FEH
PCL	0，FFH

9.4.1返回码

SROM还具有返回码和锁定功能.

返回代码有助于确定成功或失败
一个特定的功能.返回码存储在KEY1的位置
在参数块中. CheckSum和TableRead函数
没有返回码，因为KEY1的位置在
参数块用于返回其他数据.

表9-3. SROM返回码

返回代码	描述
00H	成功
01H	由于保护级别，不允许功能 在块上.
02H	软件重置，无需硬件重置.
03H	致命错误，SROM停止.

如果目标块为，则读取，写入和擦除操作可能会失败
读或写保护.块保护级别在设置期间设置
设备编程.

EraseAll函数除了保留数据之外还会覆盖数据
整个用户在擦除状态下闪烁. EraseAll函数循环
通过产品中闪存宏的数量，执行
按照以下顺序：擦除，批量编程全零，擦除.后
所有闪存宏中的所有用户空间都被擦除，第二个
循环擦除，然后用零对每个保护块编程.

9.5 SROM功能描述

9.5.1 SWBootReset函数

SROM函数SWBootReset是一个函数
负责将器件从复位状态转换到
运行用户代码. SWBootReset函数被执行
每当用M8C累加器值输入SROM时
00h：SRAM参数块不被用作输入
功能.这是在硬件重置后通过设计发生的，
因为M8C的累加器被重置为00h或当用户
代码执行累加器值为的SSC指令
00H.当SSC时SWBootReset函数不执行
指令是使用错误的键值和非零值执行的
功能代码.一个enCoRe II设备执行HALT
指令是否给KEY1或KEY2指定了错误的值.

SWBootReset功能验证校准的完整性
数据通过16位校验和，然后释放M8C
运行用户代码.

9.5.2 ReadBlock函数

ReadBlock函数用于读取64个连续字节
从闪光灯：一个块.

该功能首先检查保护位并确定是否
期望的BLOCKID是可读的.如果读保护打开，
ReadBlock函数退出设置累加器和KEY2
回到00h. KEY1的值为01h，表示读取失败.
如果未启用读取保护，则该函数将从中读取64个字节
闪存使用ROMX指令并将结果存储在
SRAM使用MVI指令.第一个64字节是
存储在SRAM所指示的地址的值中
POINTER参数.当ReadBlock完成时
成功地，累加器，KEY1和KEY2都有一个值
00h.

表9-4. ReadBlock参数

名称	地址	描述
KEY1	0，F8H	3 A H
KEY2	0，F9H	堆栈指针值，当SSC时 执行.
BLOCKID	0，FAH	闪存块编号
指针	0，FBH	SRAM中的64个地址中的第一个 必须存储返回的数据.

9.5.3 WriteBlock函数

WriteBlock功能用于将数据存储在闪存中.数据一次从SRAM移动64个字节到使用它的闪存功能. WriteBlock功能首先检查保护位并确定所需的BLOCKID是否可写.如果写保护打开，WriteBlock功能退出设置累加器和KEY2回到00h. KEY1的值为01h，表示写入失败. WriteBlock的配置功能很简单.闪存块的BLOCKID，数据存储在哪里，必须确定并存储在SRAM地址FAh.

要存储的64个字节中第一个的SRAM地址必须使用中的POINTER变量指示闪烁参数块（SRAM地址FBh）.最后，CLOCK和DELAY值必须正确设置. CLOCK值确定 - 挖掘用于存储数据的写入脉冲的长度在闪光灯中. CLOCK和DELAY值依赖于CPU速度并且必须正确设置.

表9-5. WriteBlock参数

名称	地址	描述
KEY1	0，F8H	3AH
KEY2	0，F9H	堆栈指针值，当SSC时执行.
BLOCKID	0，FAH	8KB闪存块号（00h-7Fh） 4KB闪存块号（00h-3Fh） 3KB闪存块号（00h-2Fh）
指针	0，FBH	SRAM中的64个地址中的第一个，其中要存储在闪存中的数据是位于调用WriteBlock之前.
时钟	0，FCH	时钟分频器用于设置写入脉冲宽度.
延迟	0，FEH	对于12 MHz的CPU速度设置为56H.

9.5.4擦除块功能

擦除块功能用于擦除64块闪存中的连续字节. EraseBlock函数首先检查保护位并确定所需的BLOCKID是否为写.如果写保护打开，则使用EraseBlock功能退出将累加器和KEY2设置回00h. KEY1有值为01h，表示写入失败. EraseBlock功能仅作为编程的第一步才有用.当一个块是擦除，块中的数据不是百分之百不可读.如果目标是消除块中的数据，那么最好的方法是执行一个EraseBlock，然后执行Write-全部为零的块.

要设置EraseBlock功能的参数块，正确的键值必须存储在KEY1和KEY2中.该块要删除的数字必须存储在BLOCKID变量中CLOCK和DELAY的值必须基于当前的CPU速度.

表9-6.擦除块参数

名称	地址	描述
KEY1	0，F8H	3AH
KEY2	0，F9H	堆栈指针值，当SSC时执行.
BLOCKID	0，FAH	闪存块号（00h-7Fh）
时钟	0，FCH	时钟分频器用于设置擦除脉冲宽度.
延迟	0，FEH	对于12 MHz的CPU速度设置为56H

9.5.5 ProtectBlock函数

enCoRe II器件逐块提供闪光保护基础.表9-7列出了可用的保护模式.在这个表格中，ER和EW表示执行外部读取和处理的能力写道.对于内部写入，使用IW.内部阅读是允许通过ROMX指令.通过阅读的能力SR指示SROM ReadBlock功能的方式.该根据表9-7保护级别存储在两个位中.这些位被打包到64个字节的保护中块.结果，每个保护块字节存储四个闪存块的保护级别.这些位被打包成一个字节，最低编号块的保护级别存储在最低编号位表9-7.

保护块的第一个地址包含保护级别0到3;第二个地址是块4通过7.第64个字节存储块的保护级别252至255.

表9-7.保护模式

模式	设置	描述	营销
00B	SR ER EW IW	无保护	无保护
01B	SR ER EW IW	读取保护	工厂升级
10B	SR ER EW IW	禁用外部写	现场升级
11B	SR ER EW IW	禁用内部写	全面保护

76		54		321			0
块n + 3		Block n + 2		块n + 1		Block n	

只有EraseAll才能降低保护级别将零置于保护块的所有位置.设置保护级别，使用ProtectBlock功能.这个函数从SRAM获取数据，从地址80h开始，将它与保护块中的当前值进行逻辑或运算.结果然后将OR运算存储在保护块中.该EraseBlock函数不会更改a的保护级别块.因为保护数据的SRAM位置是固定的并且每个闪存宏只有一个保护块ProtectBlock函数需要的变量非常少调用该函数之前要设置的参数块.该除了键以外，还必须设置参数块值CLOCK和DELAY值.

表9-8. ProtectBlock参数

名称	地址	描述
KEY1	0，F8H	3AH
KEY2	0，F9H	SSC时的堆栈指针值 执行.
时钟	0，FCH	时钟分频器用来设置写入脉冲 宽度.
延迟	0，FEH	对于12 MHz的CPU速度设置为56h.

9.5.6 EraseAll函数

EraseAll函数执行一系列的步骤来破坏
闪存宏中的用户数据并重置保护块
每个闪存宏全部为零（不受保护的状态）.该
EraseAll函数不会影响上面的三个隐藏块
每个闪存宏中的保护块.这四个中的第一个
隐藏块用于存储八个保护表
千字节的用户数据.

EraseAll函数从擦除用户空间开始
具有最高地址范围的闪存宏.批量程序
然后在相同的闪光宏上执行全零，以销毁
所有以前内容的痕迹.大宗计划得到遵守
通过第二次擦除，使闪存宏处于准备状态
写作.然后擦除，编程，擦除序列
在地址空间中的下一个最低闪存宏上执行
如果存在.擦除用户空间后，保护块
对于地址范围最高的闪存宏被擦除.
擦除保护块后，将写入零
保护表的每一位.下一个最低的闪光宏
地址空间的保护块被擦除并填充
与零.

EraseAll函数的最终结果是，所有的用户数据
闪光灯被破坏并且闪光灯处于未编程状态，
准备接受各种写入命令之一.该
所有用户数据的保护位也被重置为零状态

除了键之外，必须设置的参数块值，
是CLOCK和DELAY值.

表9-9.擦除所有参数

名称	地址	描述
KEY1	0，F8H	3AH
KEY2	0，F9H	SSC时的堆栈指针值 执行.
时钟	0，FCH	时钟分频器用来设置写入脉冲 宽度.
延迟	0，FEH	对于12 MHz的CPU速度设置为56h

9.5.7 表读功能

TableRead函数使用户可以访问特定的部分
制造期间存储在闪存中的数据.它也返回一个
芯片版本号（不要与Silicon ID混淆）.

表9-10.表读取参数

名称	地址	描述
KEY1	0，F8H	3AH
KEY2	0，F9H	SSC时的堆栈指针值 执行.
BLOCKID	0，FAH	要读取的表号.

enCoRe II的表空间只是一个64字节的行被破坏
分成8个八字节的表格.表格编号为零
通过七. CY7C638xx中的所有用户和隐藏块
部分由64个字节组成.

内部表格（表格0）包含Silicon ID并返回
版本ID. Silicon ID在SRAM中返回，而
修订版本和系列ID在CPU_A和CPU_X中返回
寄存器. Silicon ID是表格中的一个值
编程闪存并由Cypress Semicon-
产品工程.版本ID被硬编码到
SROM也冗余放置在SROM表1中
将在本节后面进行更详细的讨论.

SROM表1保存的是系列/芯片ID和版本ID值
该设备并返回一个1字节的内部修订计数器.该
内部修订计数器的值从零开始，并且是
当其他版本号之一不增加时递增 -
mented.当其他修订版本之一被重置为零
数字递增.返回内部修订计数
在CPU_A寄存器中. CPU_X寄存器始终设置为FFh
当读取表1时. CPU_A和CPU_X总是寄存器
当读取表2-7时返回值FFh. BLOCKID
值在参数块中指示哪个表必须是
返回给用户.只有三个最低有效位
TableRead函数使用BLOCKID参数
enCoRe II设备.高五位被忽略.当...的时候
函数被调用，它将表中的字节传输到SRAM
地址F8h-FFh.

TableRead函数使用M8C的A和X寄存器
返回芯片的版本ID.版本ID是一个16位值
硬编码到唯一识别芯片的SROM中
设计.

相应Table调用的返回值列表为
如第18页上的表 9-11 所示

表9-11.表读取的返回值

表号	返回值	
	一个	X
0	版本ID	家庭ID
1	内部修订计数器0xFF	
2-7	为0xFF	为0xFF

图9-3. SROM表

	F8H	F9H	FAH	FBH	FCH	以及FDH	FEH	FFH
Table0	Silicon ID [15-8]	Silicon ID [7-0]						
表格1	家庭/模具ID	调整ID						
表2								
表3								
表4								
表5								
表6								
表7								

enCoRe II器件的硅ID存储在该部分的SROM表中，如图9-3所示

可以使用SROM Table读取从零件中读出硅ID（表0）.这在下面的伪代码中演示.
如第15页的SROM部分所述，SROM变量通过SRAM中的FFh占用地址F8h.每一个
变量及其定义在第15页的章节SROM中给出.

```
区域SSCParmBlkA（RAM，ABS）

    组织   F8h //变量从地址F8h开始定义

SSC_KEY1:                    ; F8H   监督关键
SSC_RETURNCODE:    blk 1; F8H      结果代码
SSC_KEY2:          blk 1    ;F9H   监督堆栈密钥
SSC_BLOCKID:       blk 1; FAH      块ID
SSC_POINTER:       blk 1; FBH      指针指向数据缓冲区
SSC_CLOCK:         blk 1; FCH      时钟
SSC_MODE:          blk 1;以及FDh   ClockW ClockE乘法器
SSC_DELAY:         blk 1; FEH      闪光宏序列延迟计数
SSC_WRITE_ResultCode: blk 1; FFH      临时结果代码

_主要:
            MOV   A，0
            MOV   [SSC_BLOCKID]，A //从表0中读取 - 硅ID存储在表0中
//调用SROM操作读取SROM表
            MOV   X，SP        ;将SP复制到X中
            MOV   A，X         ;存储在X中的临时文件
            加    A，3                ;创建3个字节的堆栈帧（2 +推A）
            MOV   [SSC_KEY2]，A       ;为监控代码保存堆栈帧

    ;加载用于闪存操作的监控代码
            MOV   [SSC_KEY1]，3Ah   ; flash_OPER_KEY - 3Ah

            MOV   A，6         ;用特定的操作加载A. 06h是表格9-1的表格代码
            SSC                ; SSC调用管理ROM

//在SSC命令结束时，芯片ID存储在SRAM的F8（MSB）和F9（LSB）中

.terminate:
    jmp.终止
```

9.5.8 校验和功能

校验和功能通过a计算16位校验和
用户可指定的块数，在一个闪存宏中
（银行）从零开始. BLOCKID参数是
用于传递块数来计算校验和
过度. BLOCKID值1仅计算校验和
块0，而BLOCKID值为0则计算校验和
全部256个用户块. KEY1和KEY中返回16位校验和
KEY2.参数KEY1保存的低八位
校验和和参数KEY2保存高8位
校验和.

校验和算法执行以下序列
三个指令在块数64以上
校验和.

romx
添加[KEY1]，A
adc [KEY2]，0

表9-1.校验和参数

名称	地址	描述
KEY1	0，F8H	3A H
KEY2	0，F9H	SSC时的堆栈指针值 执行.
BLOCKID	0，FAH	要计算的闪存块数量 校验和.

10.时钟

enCoRe II有两个内部振荡器，内部24 MHz
振荡器和32 kHz低功耗振荡器.

内部24 MHz振荡器的设计可能如此
在整个温度范围内调整为24 MHz的输出频率
电压变化.随着USB流量的出现，内部
24 MHz振荡器可能被设置为精确调谐到USB时序
要求（24 MHz±1.5%）.没有USB流量，内部
24 MHz振荡器精度为24 MHz±5%（0°-70°C之间）.
不需要外部组件来达到这个水平
准确性.

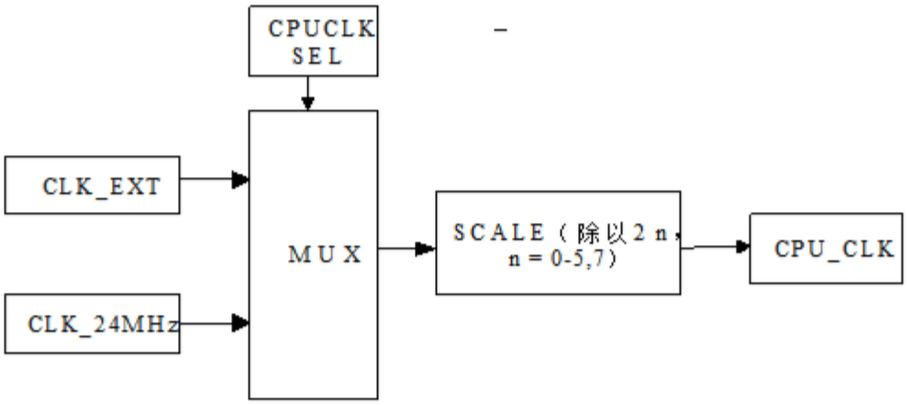
标称32 kHz的内部低速振荡器提供a
enCoRe II在暂停模式下的慢时钟源，特别 -
larly产生一个周期性的唤醒中断，也提供
在上电和断电期间连续逻辑的时钟
主时钟停止时的事件.另外，这个振荡器
也可以用作间隔定时器时钟的时钟源
（ITMRCLK）和捕捉定时器时钟（TCAPCLK）. 32 kHz
低功耗振荡器可以在低功耗模式下工作或可以工作
在正常模式下提供更准确的时钟.内置的
32 kHz低功耗振荡器精度范围（在
0°-70°C）遵循：

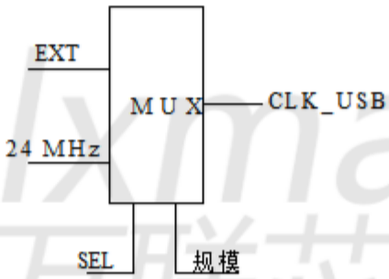
- 5 V正常模式：-8%至+ 16%
- 5 V LP模式：+ 12%至+ 48%

当使用32 kHz振荡器时，PITMRL / H寄存器必须
直到连续两次读数匹配结果为止
认为有效.以下固件示例假定为
开发人员对PIT的低位字节感兴趣.

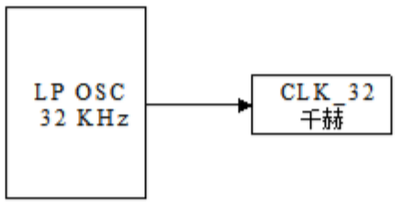
```
Read_PIT_counter:
mov A，reg [PITMRL]
mov [57h]，A
mov A，reg [PITMRL]
mov [58h]，A
mov [59h]，A
mov A，reg [PITMRL]
mov [60h]，A
;;;开始比较
mov A，[60h]
mov X，[59h]
子A，[59h]
jz完成了
mov A，[59h]
mov X，[58h]
子A，[58h]
jz完成了
mov X，[57h]
;;;正确的数据在内存位置57h
完成：
mov [57h]，X
RET
```

图10-1.时钟框图





SEL	规模	OUT
0X		12 MHz
0X		12 MHz
1	1	EXT / 2
11		EXT



10.1时钟架构描述

enCoRe II时钟选择电路允许选择
CPU，USB，间隔定时器和CPU的独立时钟
捕获定时器.

CPU时钟CPUCLK来自外部时钟或外部时钟
内部24 MHz振荡器.所选的时钟源是
可选地除以2 *n*，其中n是0-5,7（参见第10页上的表10-4）
24）.

USBCLK，USB SIE的功能必须是12 MHz
正确地，由内部24 MHz振荡器或一个
外部12 MHz / 24 MHz时钟.可选的二分频允许
使用24 MHz的信号源.

间隔定时器时钟（ITMRCLK）来自外部
时钟，内部24 MHz振荡器，内部32 kHz低电平
电源振荡器，或来自定时器捕捉时钟（TCAPCLK）.一个
可编程预分频器的1,2,3,4然后将选中的分频
资源.

定时器捕捉时钟（TCAPCLK）来自外部
时钟，内部24 MHz振荡器或内部32 kHz低电平
电源振荡器.

CLKOUT引脚（P0.1）由多个源之一驱动.这个
用于测试，也用于某些应用.该
驱动CLKOUT的来源如下：

- 可选的EFTB滤波器之后的CLKIN
- 内部24 MHz振荡器
- 内部32 kHz低功耗振荡器
- 可编程分频器后的CPUCLK

表10-1. IOSC修剪（IOSCTR）[0x34] [R / W]

位#	7	6	五	4	3	2	1	0
领域	foffset [2：0]			增益[4：0]				
读/写	R / W	R / W	R / W	R / W	R / W	R / W	R / W	R / W
默认	0	0	0	d	d	d	d	d

IOSC校准寄存器校准内部振荡器.重置值未定义，但在启动期间，SROM写入一个
在制造测试期间确定的校准值.在正常使用期间，此值不需要更改.这是
the meaning of 'D' in the Default field.

位[7：5]:foffset [2：0]

该值用于调整内部振荡器的频率.这些位在工厂校准中不使用，并且为零.
设置这些位中的每一位都会使振荡器频率产生适当的精确偏移.

foffset位0 = 7.5 kHz

foffset位1 = 15 kHz

foffset位2 = 30 kHz

位[4：0]:增益[4：0]

偏移量输入的有效频率变化通过增益输入进行控制.增益设置的较低值增加
偏移量输入的增益.该值设置内部振荡器的每个偏移步长的大小.名义收益变化
（kHz / offsetStep），典型条件（24 MHz操作）：

增益位0 = -1.5 kHz

增益位1 = -3.0 kHz

增益位2 = -6 kHz

增益位3 = -12 kHz

增益位4 = -24 kHz

表10-2. LPOSC修剪（LPOSCTR）[0x36] [R / W]

位#	7	6	五	4	3	2	1	0
领域	32_kHz低功率	保留的	32 kHz偏置调整[1：0]		32 kHz频率调整[3：0]			
读/写	R / W	-	R / W	R / W	R / W	R / W	R / W	R / W
默认	0	d	d	d	DD		d	d

该寄存器用于校准32 kHz低速振荡器.重置值未定义，但在启动期间SROM写入
在制造测试期间确定的校准值.在正常使用期间，此值不需要更改.这个
is the meaning of 'D' in the Default field. If the 32 kHz Low power bit is written, care must be taken to not disturb the
32 kHz Bias Trim和32 kHz Freq Trim字段与其出厂校准值进行比较.

位7： 32 kHz低功耗

0 = 32 kHz低速振荡器在正常模式下工作

1 = 32 kHz低速振荡器在低功耗模式下工作.振荡器继续正常工作，但是
精度降低.

位6： 保留的

位[5：4]：32 kHz偏置调整[1：0]

这些位控制低功耗振荡器的偏置电流.

0 0 =中偏

0 1 =高偏差

1 0 =保留

1 1 =保留

注意 不要用保留的10b值编程32 kHz偏置调整[1：0]字段，因为振荡器根本不会振荡
这个设置的角落条件.

位[3：0]：32 kHz频率调整[3：0]

这些位用于调整低功耗振荡器的频率.

表10-3. CPU / USB时钟配置 （ CPUCLKCR）[0x30] [R / W]

位#	7	6	五	4	3	2	1	0
领域	保留的	USB CLK禁用	USB CLK选择		保留的			CPUCLK选择
读/写	-	R / W	R / W	-	-	-	-	R / W
默认	0	0	0	0	0	0	0	0

位7： 保留的

位6： USB CLK / 2禁用

该位仅在源为外部时钟时影响USBCLK. USBCLK信号源为内部24 MHz时
振荡器，除以2总是启用

0 = USBCLK信号源被二分频.这是在使用内部24 MHz振荡器或何时使用的正确设置
外部信号源使用24 MHz时钟

1 = USBCLK是未分开的.仅在12 MHz外部时钟时使用此设置

位5： USB CLK选择

该位控制USB SIE的时钟源.

0 =内部24 MHz振荡器.随着USB流量的出现，内部24 MHz振荡器被调整以符合USB
要求1.5%的容差（参见第25页上的表10-5）

**1 =外部时钟 - 内部振荡器未调整为USB通信.正确的USB SIE操作需要12 MHz或24 MHz
时钟精确到<1.5%.**

位[4：1]：保留的

位0： CPU CLK选择

0 =内部24 MHz振荡器.

1 =外部时钟 - CLKIN（P0.0）引脚的外部时钟.

注意 CPU速度选择使用OSC_CR0寄存器（第24页的表10-4）进行配置.

表10-4. OSC控制0（OSC_CR0）[0x1E0] [R / W]

位#	7	6	五	4	3	2	1	0
领域	保留的		没有嗡嗡声	睡眠定时器[1：0]		CPU速度[2：0]		
读/写	-	-	R / W	R / W	R / W	R / W	R / W	R / W
默认	0	0	0	0	0	0	0	0

位[7：6]：保留的

位5： 没有嗡嗡声

在休眠期间（休眠位置于CPU_SCR寄存器 - 第28页的表11-1中），LVD和POR检测电路将转为定期检测V CC 引脚上的任何POR和LVD事件 （ECO_TR中的休眠占空比位用于控制占空比 - 第33页的表13-3）.为了便于检测POR和LVD事件，使用No Buzz位来强制执行LVD和POR检测电路在休眠期间连续使能.这导致对LVD或POR的响应速度更快睡眠期间的事件以略高于平均睡眠电流为代价.

0 =按照休眠占空比中的配置，LVD和POR检测电路周期性地打开.

1 =睡眠占空比值被覆盖. LVD和POR检测电路始终处于使能状态.

注意 周期性休眠占空比使能与休眠时间间隔无关，如下面的休眠[1：0]位所示.

位[4：3]：睡眠定时器[1：0]

注意 休眠间隔是近似的.

位[2：0]：CPU速度[2：0]

enCoRe II可以在一系列CPU时钟速度下运行. CPU速度位的复位值为零;结果，默认CPU速度是内部24 MHz或3 MHz的八分之一

无论CPU速度位的设置如何，如果实际CPU速度大于12 MHz，则为24 MHz操作要求应用.这种情况的一个示例是配置为使用提供20 MHz频率的外部时钟的器件.如果CPU速度寄存器的值为0b011，则CPU时钟为20 MHz.因此，该设备的电源电压要求与部件工作在24 MHz时相同.在CPU速度之前，工作电压要求不会放宽在12 MHz或更低.

CPU速度 [2：0]	内部时CPU 选择振荡器	外部时钟
000	3 MHz（默认）	时钟输入/ 8
001	6 MHz	时钟输入/ 4
010	12 MHz	时钟输入/ 2
011	24 MHz	时钟输入/ 1
100	1.5 MHz	时钟输入/ 16
101	750 kHz	时钟输入/ 32
110	187千赫	时钟输入/ 128
111	保留的	保留的

注意 正确的USB操作要求CPU时钟速度至少为1.5 MHz或不低于USB时钟/ 8.如果两个时钟具有相同的信号源，那么CPU时钟分频器不能设置为8以上的分频.如果两个时钟不同来源，在两个时钟的完整规格范围内，USB时钟/ CPU时钟的最大比例不得超过8源.

注意 该寄存器存在于第二组I / O空间中.这需要设置CPU标志寄存器中的XIO位.

表10-5. USB时钟时钟配置（OSCLCKCR）[0x39] [R / W]

位#	7	6	五	4	3	2	1	0
领域	保留的						只有微调	USB时钟禁用
读/写	-	-	-	-	-	-	R / W	R / W
默认	0	0	0	0	0	0	0	0

该寄存器用于使用接收到的低速USB数据包作为定时参考来调整内部24 MHz振荡器.该当内部24 MHz振荡器提供USB时钟时，USB Osclock电路处于活动状态.

位[7：2]:保留的

位1： 只有微调

0 =精细和课程调整

1 =禁止振荡器锁定执行其重新调谐的粗调部分.振荡器锁必须被允许执行粗调以调谐振荡器以进行正确的USB SIE操作.振荡器正确调谐后，该位置位以减少可能导致调整的内部振荡器频率的变化.

位0： USB Osclock禁用

0 =启用.由于USB流里的存在，内部24 MHz振荡器精确调谐至24 MHz±1.5%

1 =禁用.内部24 MHz振荡器未根据USB数据包进行调整.这个设置在内部时很有用振荡器不是源自USBSIE时钟.

表10-6.定时器时钟配置（TMRCLKCR）[0x31] [R / W]

位#	7	6	五	4	3	2	1	0
领域	TCAPCLK分频器		TCAPCLK选择		ITMRCLK分频器		ITMRCLK选择	
读/写	R / W	R / W	R / W	R / W	R / W	R / W	R / W	R / W
默认	1	0	0	0	1	1	1	1

位[7：6]:TCAPCLK分频器[1：0]

TCAPCLK分频器控制TCAPCLK除数.

0 0 =分频器值2

0 1 =分频器值4

1 0 =分频器值6

1 1 =分频器值8

位[5：4]:TCAPCLK选择

TCAPCLK选择字段控制TCAPCLK的来源.

0 0 =内部24 MHz振荡器

0 1 = CLKIN（P0.0）输入时的外部时钟 - 外部时钟.

1 0 =内部32 kHz低功耗振荡器

1 1 = TCAPCLK禁用

注意 1024 μs间隔定时器基于TCAPCLK以4 MHz运行的假设. TCAPCLK频率的变化导致1024的相应变化 μs间隔定时器频率.

位[3：2]:ITMRCLK分频器

ITMRCLK分频器控制ITMRCLK除数.

0 0 =分频器值为1

0 1 =分频器值2

1 0 =分频器值3

1 1 =分频器值4

位[1：0]:ITMRCLK选择

0 0 =内部24 MHz振荡器

0 1 = CLKIN（P0.0）输入时的外部时钟 - 外部时钟.

1 0 =内部32 kHz低功耗振荡器

1 1 = TCAPCLK

10.1.1间隔定时器时钟（ITMRCLK）

间隔定时器时钟（TITMRCLK）来源于
外部时钟，内部24 MHz振荡器，内部32 kHz
低功耗振荡器或定时器捕获时钟.一个
然后1,2,3或4的可编程预分频器将选中的分频器分频
资源. 12位可编程间隔定时器非常简单
具有可编程重载值的递减计数器.它提供了一个
1 µs默认分辨率.当减计数器达到零时，
下一个时钟用于重新加载.重载值被读取和
计数器正在运行时写入，但计数器不能
当12位重载值仅为无意中重新加载时
部分存储，即在12位值的两次写入之间.
可编程间隔定时器产生一个中断
CPU在每次重新加载.

要设置的参数在设备编辑器视图中显示
当enCoRe II定时器用户模块是PSoC Designer时
放置.参数是PITIMER_Source和
PITIMER_Divider. PITIMER_Source是定时器的时钟
并且PITMER_Divider是时钟除以的值.

间隔寄存器（PITMR）保存加载的值
终端计数中的PIT计数器. PIT柜台下跌
计数器.

可编程间隔定时器分辨率是可配置的.对于
例：

TCAPCLK除以CPU时钟的x（例如，TCAPCLK
除以2个24 MHz CPU时钟产生12 MHz的频率.）

ITMRCLK除以TCAPCLK的x（例如，ITMRCLK
TCAPCLK的3分频为4 MHz，因此分辨率为0.25 微秒）.

10.1.2定时器捕捉时钟（TCAPCLK）

定时器捕捉时钟来自外部时钟，
内部24 MHz振荡器或内部32 kHz低功耗
振荡器.一个可编程的2，4，6或8分频器然后分频
选定的来源.

图10-2.可编程间隔定时器框图

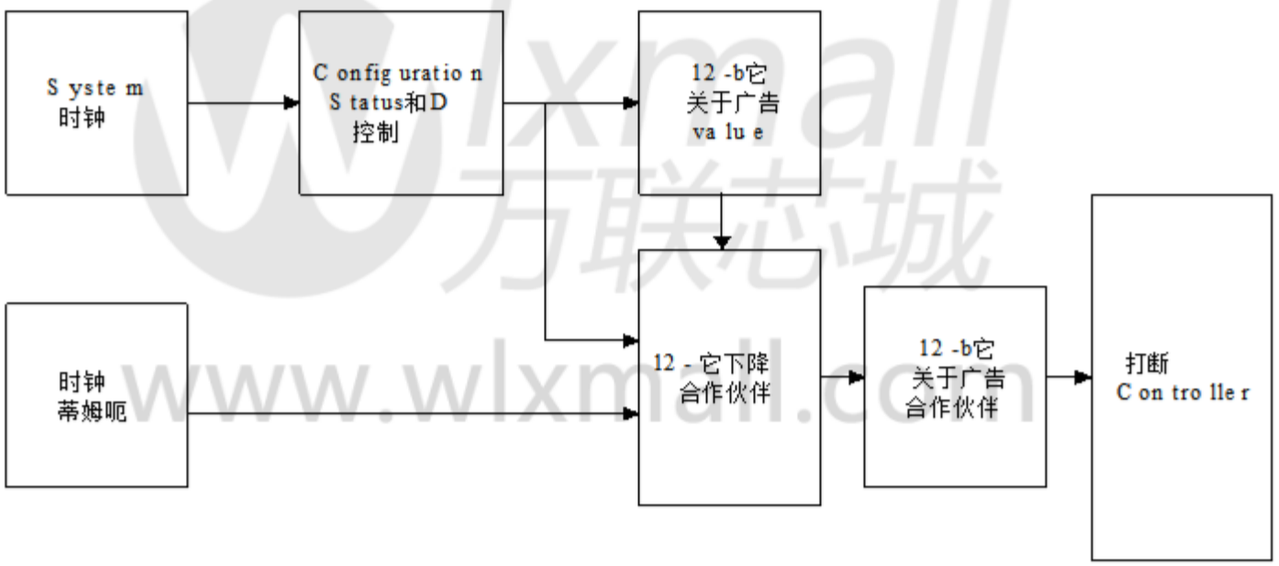


图10-3.定时器捕获框图

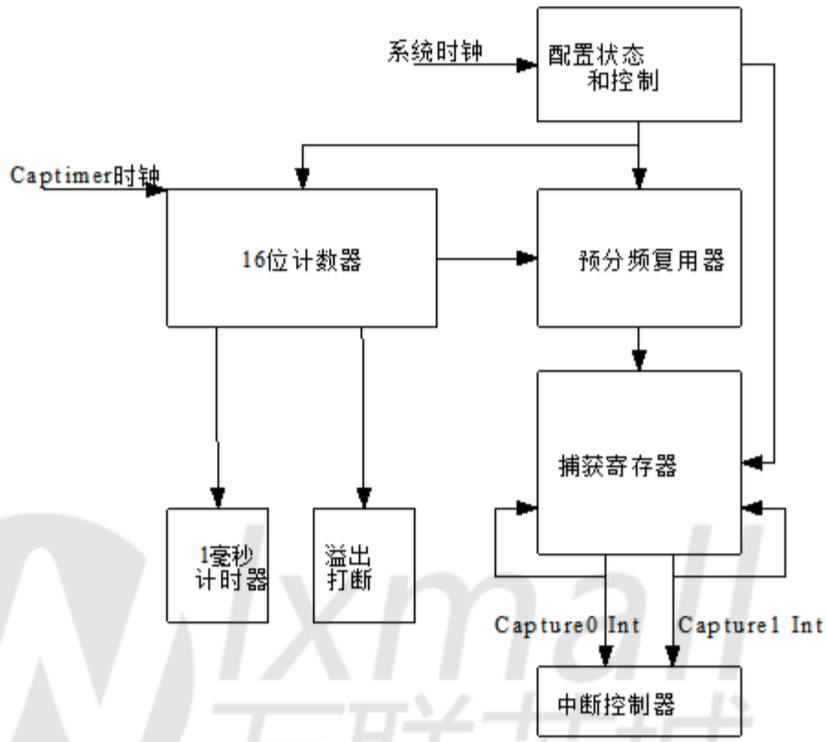


表10-1.时钟IO配置（CLKIOCR）[0x32] [R / W]

位#	7	6	五	4	3	2	1	0
领域	保留的						CLKOUT选择	
读/写	-	-	-	---			R / W	R / W
默认	0	0	0000				0	0

位[7：2]:保留的

位[1：0]:CLKOUT选择

0 0 =内部24 MHz振荡器

0 1 =外部时钟 - CLKIN的外部时 钟（P0.0）

1 0 =内部32 kHz低功耗振荡器

1 1 = CPUCLK

10.2睡眠模式下的CPU时钟

当CPU进入睡眠模式时，CPUCLK选择（第23页的表10-3的位[0]）强制为内部振荡器，而振荡器停止.当CPU退出睡眠模式时，它将在内部振荡器上运行.内部振荡器恢复时间为内部32 kHz低功耗振荡器的三个时钟周期.

如果系统要求CPU在休眠模式唤醒后耗尽外部时钟，则固件必须切换时钟源为CPU.

11.重置

微控制器支持两种类型的复位：上电复位（POR）和看门狗复位（WDR）.当重新启动时，全部寄存器恢复到默认状态，所有中断都被禁止.

系统状态和控制寄存器（CPU_SCR）中记录了复位的发生.该寄存器中的位记录分别发生POR和WDR复位.固件询问这些位以确定复位的原因.

复位后，微控制器从闪存地址0x0000恢复执行.内部时钟模式在复位后有效，直到被用户固件更改.

注意 在POR时，CPU时钟默认为3 MHz（内部24 MHz振荡器除8模式），以保证在低V CC这可能在供应阶段出现.

表11-1.系统状态和控制寄存器（CPU_SCR）[0xFF] [R / W]

位#	7	6	五	4	3	2	1	0
领域	GIES	保留的	WDRS	PORS	睡觉	保留的		停止
读/写	[R	-	R / C [4]	R / C [4]	R/W	-	-	R/W
默认	0	0	0	1	00		0	0

CPU_SCR寄存器的位用于传送状态和enCoRe II设备各种功能的事件控制.

位7： GIES

全局中断使能状态位是只读状态位，不鼓励使用. GIES位是一个遗留位，用于提供读取CPU_F寄存器的GIE位的功能.但是，CPU_F寄存器现在可读.什么时候该位置1，表示CPU_F寄存器中的GIE位也被置1，这反过来表示微处理器服务中断.

0 =全局中断禁用

1 =启用全局中断

位6： 保留的

位5： WDRS

WDRS位由CPU设置以指示发生了WDR事件.用户可以读取该位来确定类型重置已发生.用户可以清除但不设置此位.

0 =没有WDR

1 =发生了WDR事件

位4： PORS

PORS位由CPU设置以指示发生了POR事件.用户可以读取该位来确定类型重置已发生.用户可以清除但不设置此位

0 =没有POR

1 =发生了POR事件.（注意WDR事件在该位清零之前不会发生）

位3： 睡觉

由用户设置以启用CPU睡眠状态. CPU处于休眠模式，直到有任何中断挂起.睡眠位被覆盖
详细信息请参阅第29页的睡眠模式部分.

0 =正常操作

1 =睡眠

位[2：1]：保留的

位0： 停

该位由用户设置以暂停CPU.在复位（WDR，POR或外部复位）发生之前，CPU保持停止状态.如果应用程序想要停止代码执行直到重置，则首选方法是使用HALT指令而不是写入到这一点.

0 =正常CPU操作

1 = CPU暂停（不推荐）

注意
4. C =清除.该位仅由用户清除，不能由固件设置.

11.1上电复位

每次开启设备电源时都会发生POR.
当电源电压上升时典型值为2.6V时，POR被释放
电源转换，通常在50 mV的滞后期间
打开瞬态电源.系统状态和控制的位4
寄存器（CPU_SCR）被设置为记录该事件（寄存器
内容由POR设置为00010000）.在POR之后，
对于V CC，微处理器停止约20 ms
在执行第一条指令之前稳定供应
地址0x00在闪光灯中. 如果V CC电压下降到
POR向下供电跳变点，POR重新置位. V CC
电源必须在200 ms内从0至4V线性上升.

注意 PORS状态位被设置为POR，并且只能被清零
用户.它不能由固件设置.

11.2看门狗定时器复位

用户可以选择启用WDT. WDT已启用
通过清除PORS位. PORS位清零后，WDT
不能被禁用.唯一的例外是如果是POR事件
发生，从而禁用WDT.

睡眠定时器用于生成睡眠时间段和睡眠时间
看门狗时间段.睡眠定时器使用内部32 kHz
低功耗振荡器系统时钟产生睡眠时间
期.用户可以使用程序编程睡眠时间段
OSC_CR0寄存器的睡眠定时器位（第10页的表10-4）
24）.当睡眠时间过去时（睡眠定时器溢出），一个
产生中断睡眠定时器中断向量.

看门狗定时器周期自动设置为3
睡眠定时器溢出的计数.这代表两个之间
和三个睡眠间隔，具体取决于睡眠中的计数
定时器在以前的WDT清除.当这个计时器达到三时，
生成WDR.

用户既可以清除WDT，也可以清除WDT和睡眠
计时器.当用户写入复位WDT寄存器时
（RES_WDT），WDT被清除.如果写入的数据是
十六进制值0x38，睡眠定时器也同时被清除.

表11-2.复位看门狗定时器（RESWDT）[0xE3] [W]

位#	7	6	五	4	3	2	1	0
领域	复位看门狗定时器[7：0]							
读/写	w^	w^	w^	w^	WW		w^	w^
默认	0	0	0	0	00		0	0

任何对该寄存器的写操作都会清除看门狗定时器;写入0x38也会清除睡眠定时器.

位[7：0]:复位看门狗定时器[7：0]

12.睡眠模式

CPU只能通过固件进入睡眠状态.这是
通过在系统状态中设置睡眠位来完成
控制寄存器（CPU_SCR）.这阻止了CPU
执行指令，并且CPU一直保持睡着状态，直到执行指令
中断等待，或有重置事件（电源
复位或看门狗定时器复位）.

低电压检测电路（LVD）完全下降
功能降低的状态，以及LVD的延迟
增加.实际的等待时间是交易对功率
通过更改ECO_TR的睡眠占空比字段进行消耗
寄存器.

内部32 kHz低速振荡器保持运行.
在进入挂起模式之前，固件可以选择
将32 kHz低速振荡器配置为低电平工作
功耗模式有助于降低总体功耗
（使用第23页的表10-2中的第7位）.这有助于节省
大约5 μA;然而，权衡是32 kHz低
高速振荡器不太准确.

所有中断保持活动状态.只有发生中断
从睡眠中醒来.系统状态中的停止位和
控制寄存器（CPU_SCR）必须清零
恢复睡眠. CPU的全局中断使能位
标志寄存器（CPU_F）没有任何作用.任何
未被屏蔽的中断唤醒系统.因此，任何
中断不打算醒来必须通过禁用
中断屏蔽寄存器.

当CPU进入睡眠模式时，CPUCLK选择（位1，
表10-3在23页）强制为内部振荡器.该
内部振荡器的恢复时间是三个时钟周期的
内部32 kHz低功耗振荡器.内部24 MHz
振荡器在退出睡眠模式后立即重新启动.如果
使用外部时钟时，固件将切换时钟源
中央处理器.

退出睡眠模式后，时钟稳定并延迟
时间已经过期，紧接着睡眠之后的指示
指令在中断服务程序之前执行（如果
启用）.

休眠中断允许微控制器唤醒
定期和调查系统组件，同时保持非常
低平均功耗.睡眠中断也可能
用于在非休眠模式下提供周期性中断.

12.1睡眠顺序

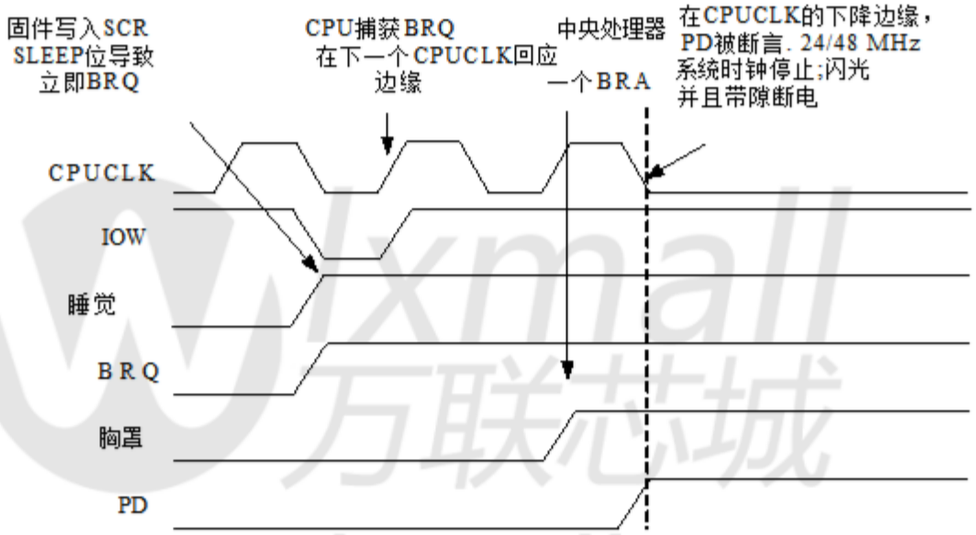
SLEEP位是睡眠逻辑电路的输入.这个电路旨在将设备排序和排除硬件睡眠状态.使设备进入睡眠状态的硬件序列如图12-1所示，定义如下.

1.固件将CPU_SCR0寄存器中的SLEEP位置1.该总线请求（BRQ）信号立即发送给CPU断言.这是系统请求暂停CPU在指令边界进行操作. CPU采样BRQ在CPUCLK的积极边缘.

2.由于寄存器写入的特定时序，CPU发出一个总线请求确认（BRA）上的以下肯定CPU时钟的边缘.睡眠逻辑等待下列内容CPU时钟的下降沿，然后断言一个系统宽电源关闭（PD）信号.在第30页的图12-1中CPU停止工作，系统宽泛的掉电信号断言.

3.系统广泛的PD（掉电）信号控制几个主要电路块：闪存模块，内部24 MHz振荡器，EFTB滤波器和带隙电压参考.这些电路转变为零功率状态.芯片上唯一的操作电路是低功耗振荡器，带隙刷新电路和电源电压.监控.（POR / LVD）电路.

图12-1.睡眠时间



12.2唤醒序列

一旦睡着了，唯一能唤醒系统的事件就是一次打断. CPU标志寄存器的全局中断使能为不需要设置.任何未被屏蔽的中断都会唤醒系统向上.之后CPU实际进行中断是可选的唤醒顺序.唤醒序列是同步的为了排序启动延迟的目的，将其设置为32kHz时钟，让闪存模块有足够的时间开机在CPU断言第一次读访问之前.另一个原因延迟是允许振荡器，Bandgap和LVD / POR在系统实际使用之前，电路需要时间来解决问题.如第31页的图12-2所示，唤醒序列为如下：

1.唤醒中断发生并通过neg-32 kHz时钟的边沿.

2.在32 kHz时钟的下一个上升沿，系统宽泛的PD信号被否定.闪存模块内部振荡器，EFTB和带隙电路全部上电至a正常运行状态.

3.在32 kHz时钟的下一个上升沿，电流精确的POR和LVD的值已经确定并且是采样.

4.在32 kHz时钟的下一个负沿（大约在后15μS标称），BRQ信号被睡眠逻辑取消电路.在以下CPUCLK上，BRA被CPU取消并继续执行指令.请注意，在图12-2中第31页固定的功能块，例如闪存，内部振荡器，EFTB和带隙，都有大约15μSec的启动.唤醒时间（中断到CPU操作）的范围从75μS至105μS.

12.3睡眠模式下的低功耗

在实现可能的最低功耗时暂停或睡眠时，必须遵守以下条件除了关于睡眠定时器的注意事项：

1.所有GPIO必须设置为输出并驱动为低电平.

2.在USB和非USB操作期间清除P11CR [0]，P10CR [0]蒸发散

3.清除USB启用USBCR [7] - 在USB模式下操作 - 蒸发散

4.设置P10CR [1] - 在非USB模式操作期间

5.确保32 kHz振荡器时钟未被选为时钟源到ITMRCLK，TCAPCLK.甚至没有作为时钟输出源，到P01_CLKOUT或P12_VREG引脚.

所有其他功能块将自动进入省电模式暂停.

以下步骤是用户可配置的，并有助于减少平均挂起模式功耗.

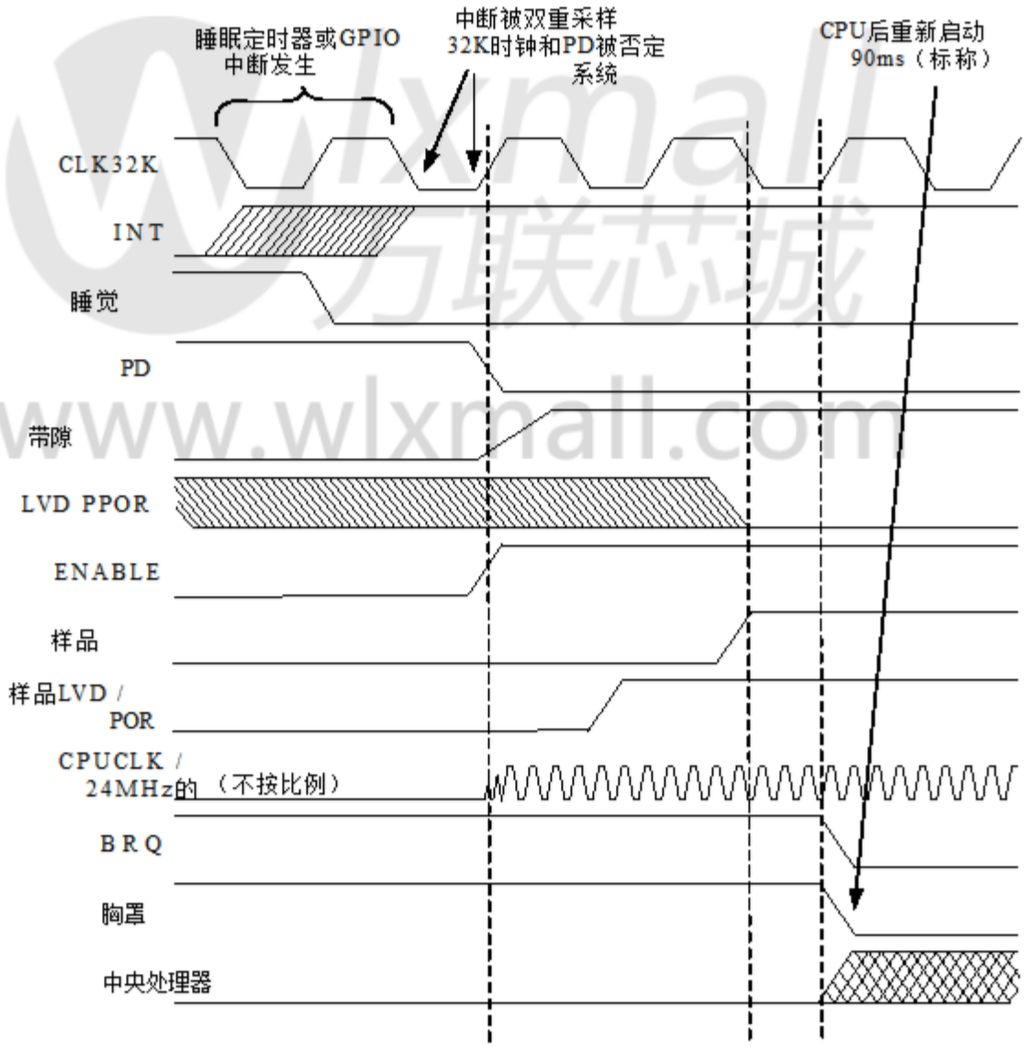
1.配置电源监控器，vals，控制寄存器位为1，EB [7：6]（电源系统睡眠占空比PSSDC [1：0]）.

2.将低功耗振荡器配置为低功耗模式，控制寄存器位是LOPSCTR [7].

用于外部时钟睡眠时的低功耗考虑因素被用作CPUCLK源，必须保持时钟源低以避免无意的泄漏电流.如果时钟被保持高，那么通过M8C可能会有泄漏.为了避免电流消耗确保ITMRCLK，TCPCLK和USBCLK不是由低功耗32 kHz振荡器或24 MHz提供的无晶振荡器.不要选择24 MHz或32 kHz振荡器时钟到P01_CLKOUT / P12_VREG引脚.

注意 在自供电设计的情况下，特别是电池电源，可能无法满足USB暂停电流规格因为USB引脚正在预期终止.

图12-2.唤醒时机



13.低电压检测控制

表13-1.低电压控制寄存器（LVDCR）[0x1E3] [R / W]

位#	7	6	五	4	3	2	1	0
领域	保留的		PORLEV [1：0]		保留的	VM [2：0]		
读/写	-	-	R / W	R / W	- R /	W	R / W	R / W
默认	0	0	0	0	00		0	0

该寄存器控制上电复位/低电压检测模块的配置.

注意 该寄存器存在于IO空间的第二组中.这需要设置CPU标志寄存器中的XIO位.

位[7：6]:保留的

位[5：4]:PORLEV [1：0]

该字段控制精度上电复位（PPOR）检测器产生复位的电平.

0 0 = 2.7V范围（跳闸接近2.6V）

0 1 = 3V范围（跳闸2.9V附近）

1 0 = 5V范围，> 4.75V（跳闸接近4.65V）.在12 MHz以上运行CPU时必须使用此设置.

1 1 = PPOR不会产生复位，但是从电压监控器比较寄存器（表13-2）中读取的值给出内部PPOR比较器状态，跳闸点设置为3V范围设置.

位3： 保留的

位[2：0]:VM [2：0]

VM [2：0]	LVD Trip Point（V）Min	LVD Trip 点（V）典型	LVD Trip 点（V）最大值
000	保留的	保留的	保留的
001	保留的	保留的	保留的
010	保留的	保留的	保留的
0 1 1	保留的	保留的	保留的
100	4.439	4.48	4.528
101	4.597	4.64	4.689
1 1 0	4.680	4.73	4.774
111	4.766	4.82	4.862

表13-2.电压监视器比较寄存器（VLTCMP）[0x1E4] [R]

位#	7	6	五	4	3	2	1	0
领域	保留的						LVD	PPOR
读/写	-	-	-	-	-		[R	[R
默认	0	0	0	0	00		0	0

该只读寄存器允许读取低电压检测和精密上电复位的当前状态，
ators

位[7：2]:保留的

位1： LVD

该位置位表示低电压检测比较器已跳闸，表明电源电压已降至低于此值
由VM [2：0]设置的跳闸点（见表13-1）
0 =没有低电压检测事件
1 =低压检测器跳闸

位0： PPOR

该位置位表示精密上电复位比较器跳闸，表明电源电压低于该值
由PORLEV [1：0]设置的跳闸点
0 =没有精密上电复位事件
1 =发生精密上电复位事件

注意 该寄存器存在于第二组I / O空间中.这需要设置CPU标志寄存器中的XIO位.

13.0.1 ECO修整寄存器

表13-3. ECO（ECO_TR）[0x1EB] [R / W]

位#	7	6	五	4	3	2	1	0
领域	睡眠占空比[1：0]		保留的					
读/写	R / W	R / W	-	-	-	-	-	-
默认	0	0	0	0	00		0	0

该寄存器控制LVD和POR检测的“开”时间与“关”时间的比率（32 kHz时钟周期数）
电路.

位[7：6]:睡眠占空比[1：0]
0 0 =内部32 kHz低速振荡器的1/128个周期
0 1 =内部32 kHz低速振荡器的1/512个周期
1 0 =内部32 kHz低速振荡器的1/32周期
1 1 =内部32 kHz低速振荡器的1/8个周期

注意 该寄存器存在于第二组I / O空间中.这需要设置CPU标志寄存器中的XIO位.

14.通用I / O（GPIO）端口

14.1端口数据寄存器

表14-1. P0数据寄存器（P0DATA）[0x00] [R / W]

位#	7	6	五	4	3	2	1	0
领域	P0.7	P0.6 / TIO1	P0.5 / TIO0	P0.4 / INT2	P0.3 / INT1	P0.2 / INT0	P0.1 / CLKOUT	P0.0 / CLKIN
读/写	R/W	R/W	R/W	R/W	R/W	R / W	R/W	R/W
默认	0	0	0	0	00		0	0

该寄存器包含端口0的数据.写入该寄存器可设置要在输出使能引脚上输出的位值.读
从该寄存器返回端口0引脚的当前状态.

位7： P0.7数据

P0.7仅在CY7C638xx中存在

位[6：5]:P0.6-P0.5数据/ TIO1和TIO0

除了用作P0.6-P0.5 GPIO外，这些引脚还可用作捕获定时器输入的备用功能
定时器输出引脚（TIO1和TIO0）.要配置P0.5和P0.6引脚，请参考P0.5 / TIO0-P0.6 / TIO1配置寄存器
（第38页的表14-4）.

在所有的enCoRe II部件中都使用引脚作为P0.6-P0.5 GPIO和备用功能.

位[4：2]:P0.4-P0.2数据/ INT2 - INT0

除用作P0.4-P0.2 GPIO外，这些引脚还用作中断引脚（INT0-INT2）的复用功能.
要配置P0.4-P0.2引脚，请参考P0.2 / INT0-P0.4 / INT2配置寄存器（第14页的表14-3）.

在所有的enCoRe II部件中都使用引脚作为P0.4-P0.2 GPIO和备用功能.

位1： P0.1 / CLKOUT

除用作P0.1 GPIO外，该引脚还用作CLK OUT引脚的备用功能.要配置P0.1引脚，
请参考P0.1 / CLKOUT配置寄存器（第14页的表14-2）.

位0： P0.0 / CLKIN

除用作P0.0 GPIO外，该引脚还用作CLKIN引脚的备用功能.要配置P0.0引脚，
请参考P0.0 / CLKIN配置寄存器（第14页的表14-1）.

表14-2. P1数据寄存器（P1DATA）[0x01] [R / W]

位#	7	6	五	4	3	2	1	0
领域	P1.7	P1.6 / SMISO	P1.5 / SMOSI	P1.4 / SCLK	P1.3 / SSEL	P1.2 / VREG	P1.1 / D-	P1.0 / d +
读/写	R / W	R / W	R / W	R / W	R / W	R / W	R / W	R / W
默认	0	0	0	0	0	0	0	0

该寄存器包含端口1的数据.写入该寄存器会设置要在输出使能引脚上输出的位值.读
从该寄存器返回端口1引脚的当前状态.

位7： P1.7数据

P1.7只存在于CY7C638xx中.

位[6：3]:P1.6-P1.3数据/ SPI引脚（SMISO，SMOSI，SCLK，SSEL）

除了用作P1.6-P1.3 GPIO外，这些引脚还用作SPI接口引脚的复用功能.至
配置P1.6-P1.3引脚，请参考P1.3-P1.6配置寄存器（第40页的表14-9）.

在所有的enCoRe II部件中都使用引脚作为P1.6-P1.3 GPIO和备用功能.

位2： P1.2 / VREG

在CY7C638x3上，该引脚用作P1.2 GPIO或VREG输出.如果VREG输出使能（位0
第58页的表19-1被置位），引脚上放置一个3.3V电源，该引脚的GPIO功能被禁止.

CY7C63310和CY7C63801变体中不存在VREG功能. A 1 μF分钟，最大电容为2μF
需要VREG输出.

位[1：0]:P1.1-P1.0 / D-和D +

当USB模式被禁止时（第59页表21-1中的位7清除），P1.1和P1.0位用来控制
P1.0和P1.1引脚.当USB模式使能时，P1.1和P1.0引脚分别用作D-和D +引脚.
如果USB强制状态位（表19-1中的位0）置1，则D-和D +引脚的状态通过写入D-和D +位来控制.

表14-3. P2数据寄存器（P2DATA）[0x02] [R / W]

位#	7	6	五	4	3	2	1	0
领域	保留的						P2.1-P2.0	
读/写	-	-	-	-	-	-	R / W	R / W
默认	0	0	0	0	0	0	0	0

该寄存器包含端口2的数据.写入该寄存器可设置输出使能引脚上的位值.读
从该寄存器返回端口2引脚的当前状态.

位[7：2]:保留数据[7：2]

位[1：0]:P2数据[1：0]

P2.1-P2.0仅存在于CY7C638（2/3）3中.

表14-4. P3数据寄存器（P3DATA）[0x03] [R / W]

位#	7	6	五	4	3	2	1	0
领域	保留的						P3.1-P3.0	
读/写	-	-	-	-	-	-	R / W	R / W
默认	0	0	0	0	0	0	0	0

该寄存器包含端口3的数据.写入该寄存器可设置要在输出使能引脚上输出的位值.读
从该寄存器返回端口3引脚的当前状态.

位[7：2]:保留数据[7：2]

位[1：0]:P3数据[1：0]

CY7C638（2/3）3中只存在P3.1-P3.0.

14.2 GPIO端口配置

所有GPIO配置寄存器都具有通用配置
控制.以下是GPIO的位定义
配置寄存器.

14.2.1 Int启用

置位时，INT使能位允许GPIO产生
中断.无论是否产生中断，都可能发生
该引脚被配置为输入或输出.所有的中断都是边缘的
敏感;但是对于多个共享的任何中断
源（即端口2，3和4）所有输入必须置为无效
在发生新的中断之前.

清零时，相应的中断在引脚上禁用.

可以将GPIO配置为输出，启用中断
在引脚上，然后通过驱动appro-
引脚状态.这在测试中很有用，可能有价值
应用.

14.2.2

置位时，相应的中断在下降时有效
边缘.

清除时，相应的中断在上升时有效
边缘.

14.2.3 TTL

设置时，输入具有TTL阈值.清楚时，输入
标准CMOS门限.

14.2.4

设置时，输出可以下降到50 mA.

清除后，输出可以下沉至8 mA.

只有P1.7-P1.3具有50 mA的灌电流驱动能力.其他引脚
具有8 mA吸收驱动能力.

14.2.5

置位时，引脚上的输出由端口数据决定
寄存器.如果端口数据寄存器中的相应位置位，
该引脚处于高阻抗状态.如果在相应位中
端口数据寄存器清零，引脚被驱动为低电平.

清零时，输出被驱动为低电平或高电平.

14.2.6

当设置该引脚具有7 K上拉到V CC（或VREG的端口
启用V3.3）.

清除时，禁止上拉.

14.2.7

置位时，引脚的输出驱动器被使能.

清除时，引脚的输出驱动器被禁用.

对于具有共享功能的引脚，有一些特殊情况.

14.2.8 VREG/ SPI

P1.2（VREG），P1.3（SSEL），P1.4（SCLK），P1.5（SMOSI）和
P1.6（SMISO）引脚用于其专用功能或用于
GPIO.

要启用GPIO引脚，请清除相应的VREG
输出或SPI使用位. SPI功能控制输出
当它们的GPIO使能位使能它的专用功能引脚时
清楚了.在CY7C63801上VREG输出不可用
和CY7C63310.

14.2.9 3.3V

P1.3（SSEL），P1.4（SCLK），P1.5（SMOSI）和P1.6（SMISO）
引脚具有来自电压调节器的备用电压源.
如果3.3V驱动位置1，则从电压驱动高电平
调节器而不是V CC.

设置3.3 V驱动器位不会启用电压调节器.
这必须通过设置VREG使能位来明确完成
VREGCR寄存器（第58页的表19-1）.

图14-1. GPIO的框图

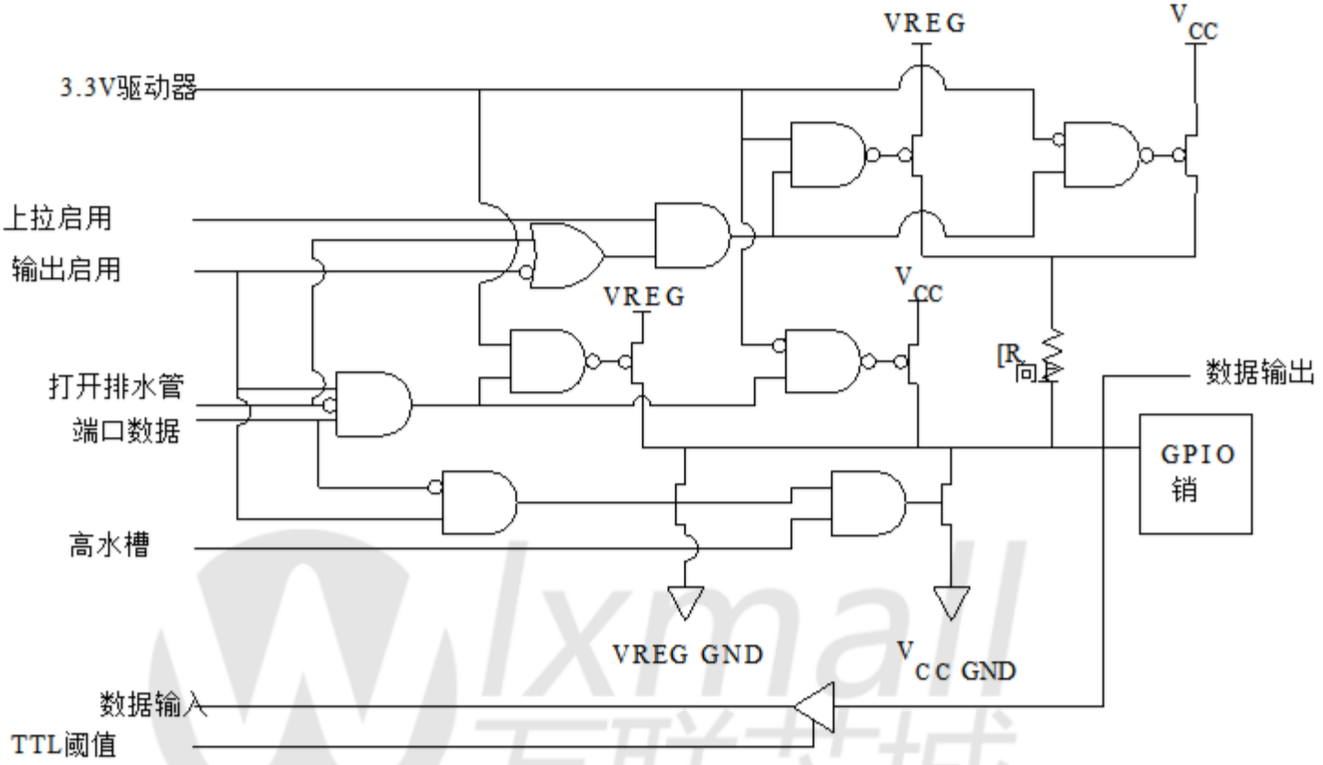


表14-1. P0.0 / CLKIN配置（P00CR）[0x05] [R / W]

位#	7	6	五	4	3	2	1	0
领域	保留的	Int启用	廉政法案低	TTL Thresh	保留的	打开排水管	上拉启用	输出启用
读/写	-	R / W	R / W	R / W	- R /	W	R / W	R / W
默认	0	0	0	0	00		0	0

该引脚在P0.0 GPIO使用和CLKIN引脚之间共享一个外部时钟.当外部时钟输入启用时（位于寄存器CPUCLKCR的表10-3（第23页）中的位[0]）时，该寄存器的设置将被忽略.

在所有的enCoRe II部件中都可以使用该引脚作为P0.0 GPIO.

表14-2. P0.1 / CLKOUT配置（P01CR）[0x06] R / W]

位#	7	6	五	4	3	2	1	0
领域	CLK输出	Int启用	廉政法案低	TTL Thresh	保留的	打开排水管	上拉启用	输出启用
读/写	R / W	R / W	R / W	R / W	- R /	W	R / W	R / W
默认	0	0	0	0	00		0	0

该引脚在P0.1 GPIO使用和CLKOUT引脚之间共享. CLK输出置位时，内部选定的时钟为发送到P0.1CLKOUT引脚.

在所有的enCoRe II部件中都可以使用该引脚作为P0.1 GPIO.

位7： CLK输出

0 =时钟输出被禁止.

1 =由CLK Select域选择的时钟（CLKIOCR寄存器的位[1：0]（第27页的表10-1））被驱动到该引脚.

表14-3. P0.2 / INT0-P0.4 / INT2配置（P02CR-P04CR）[0x07-0x09] [R / W]

位#	7	6	五	4	3	2	1	0
领域	保留的		廉政法案低	TTL Thresh	保留的	打开排水管	上拉启用	输出启用
读/写	-	-	R / W	R / W	- R /	W	R / W	R / W
默认	0	0	0	0	00		0	0

这些寄存器分别控制引脚P0.2-P0.4的操作.这些引脚由P0.2-P0.4 GPIO和GPIO共用
INT0-INT2.这些寄存器存在于所有的enCoRe II部分. INT0-INT2中断与所有其他GPIO不同
中断.这些引脚直接连接到中断控制器，以提供三个独立的边沿敏感中断
中断向量.当Int act Low清零时，这些中断发生在上升沿，当Int act Low置位时，发生在下降沿.
这些引脚在中断控制寄存器中作为中断源使用（第56页的表17-7和第54页的表17-5）.

要将这些引脚用作中断输入，请通过清除相应的输出使能将它们配置为输入.如果INT0-INT2
引脚被配置为中断使能的输出，固件可以通过写适当的值来产生中断
P0数据寄存器中的P0.2，P0.3和P0.4数据位.

无论引脚是用作中断还是GPIO引脚，都可以使用Int Enable，Int act Low，TTL Threshold，Open Drain和
上拉使能位控制引脚的行为.

P0.2 / INT0-P0.4 / INT2引脚分别通过P02CR（0x07），P03CR（0x08）和P04CR（0x09）
tively.

注意 改变Int Act低位的状态可能会导致无意的中断产生.配置这些时
中断源，最好遵循以下过程：

1.禁用中断源

2.配置中断源

3.清除来自源的任何挂起的中断

4.启用中断源

表14-4. P0.5 / TIO0 - P0.6 / TIO1配置（P05CR-P06CR）[0x0A-0x0B] [R / W]

位#	7	6	五	4	3	2	1	0
领域	TIO输出	Int启用	廉政法案低	TTL Thresh	保留的	打开排水管	上拉启用	输出启用
读/写	-	R / W	R / W	R / W	- R /	W	R / W	R / W
默认	0	0	0	0	00		0	0

这些寄存器分别控制引脚P0.5至P0.6的操作.这些寄存器存在于所有的enCoRe II部分.

P0.5和P0.6分别与TIO0和TIO1共享.要将这些引脚用作捕捉定时器输入，请将它们配置为输入
通过清除相应的输出使能.要将TIO0和TIO1用作定时器输出，请设置TIOx输出和输出使能
位.如果这些引脚配置为输出，并且TIO输出位清零，则固件可以通过控制TIO0和TIO1输入
将该值写入P0数据寄存器中的P0.5和P0.6数据位.

无论是否将任一引脚用作TIO或GPIO引脚，都可以使用Int Enable，Int act Low，TTL Threshold，Open Drain和
上拉使能控制引脚的行为.

TIO0（P0.5）使能时，从自由运行定时器输出一个正脉冲.这是内部使用的相同信号
生成1024 μs定时器中断.该信号不受中断使能状态的门控.该脉冲在一个周期内有效
的捕捉定时器时钟.

TIO1（P0.6）使能时，从可编程间隔定时器输出一个正脉冲.这与使用的信号相同
在内部产生可编程定时器间隔中断.该信号不受中断使能状态的门控.脉搏
在间隔定时器时钟的一个周期内有效.

P0.5 / TIO0和P0.6 / TIO1引脚分别通过P05CR（0x0A）和P06CR（0x0B）进行配置.

表14-5. P0.7配置（P07CR）[0x0C] [R / W]

位#	7	6	五	4	3	2	1	0
领域	保留的	Int启用	廉政法案低	TTL Thresh	保留的	打开排水管	上拉启用	输出启用
读/写	-	R / W	R / W	R / W	-	R / W	R / W	R / W
默认	0	0	0	0	0	0	0	0

该寄存器控制引脚P0.7的操作. P0.7引脚仅存在于CY7C638（1/2/3）3中.

表14-6. P1.0 / D +配置（P10CR）[0x0D] [R / W]

位#	7	6	五	4	3	2	1	0
领域	保留的	Int启用	廉政法案低	保留的			PS / 2上拉启用	输出启用
读/写	R / W	R / W	R / W	-	-		R / W	R / W
默认	0	0	0	0	00		0	0

当USB接口未使能时，该寄存器控制P1.0（D +）引脚的操作，允许该引脚用作
一个PS2接口或一个GPIO.有关启用USB的信息，请参见第59页的表21-1.当USB被启用时，没有
该寄存器中的控制对P1.0引脚有任何影响.

注意 P1.0是仅漏极开路输出.它可以主动驱动信号较低，但不能主动驱动信号较高.

位1： PS / 2上拉使能
0 =禁用5K欧姆上拉电阻
1 =使能P1.0和P1.1的5K欧姆上拉电阻.启用使用P1.0（D +）和P1.1（D-）引脚作为PS2样式
接口.

表14-7. P1.1 / D-配置（P11CR）[0x0E] [R / W]

位#	7	6	五	4	3	2	1	0
领域	保留的	Int启用	廉政法案低	保留的		打开排水管	保留的	输出启用
读/写	-	R / W	R / W	-	- R /	W	-	R / W
默认	0	0	0	0	00		0	0

当USB接口未使能时，该寄存器控制P1.1（D-）引脚的操作，允许该引脚用作
一个PS2接口或一个GPIO.有关启用USB的信息，请参见第59页的表21-1.当USB启用时，没有任何一个
该寄存器中的控制对P1.1引脚有任何影响.当USB被禁用时，该引脚上的5K欧姆上拉电阻可能是
由P10CR寄存器的PS / 2上拉使能位使能（表14-6）

注意 该引脚不具有2 mA的采购能力. 该引脚只能在V OL3 吸收5 mA（参见章节直流特性部分
第69页）

表14-8. P1.2配置（P12CR）[0x0F] [R / W]

位#	7	6	五	4	3	2	1	0
领域	CLK输出	Int启用	廉政法案低	TTL阈值	保留的	打开排水管	上拉启用	输出启用
读/写	R / W	R / W	R / W	R / W	- R /	W	R / W	R / W
默认	0	0	0	0	00		0	0

该寄存器控制P1.2的操作.

位7： CLK输出
0 =内部选择的时钟不发送到P1.2引脚
1 =当CLK输出置位时，内部选择的时钟被发送到P1.2引脚
注意:表10-1，？$ paratext> ,?（第27页）用于在enCoRe II设备中选择外部或内部时钟

表14-9. P1.3配置（P13CR）[0x10] [R / W]

位#	7	6	五	4	3	2	1	0
领域	保留的	Int启用	廉政法案低	3.3V驱动器	高水槽	打开排水管	上拉启用	输出启用
读/写	-	R / W	R / W	R / W	R / W	R / W	R / W	R / W
默认	0	0	0	0	00		0	0

该寄存器控制P1.3引脚的操作.该寄存器存在于所有的enCoRe II部分.

P1.3 GPIO的阈值始终设置为TTL.

当SPI硬件使能或禁止时，该引脚由输出使能位和相应的位控制
P1数据寄存器.

不管该引脚是用作SPI还是GPIO引脚，Int Enable，Int act Low，3.3V驱动器，高吸收，漏极开路和
上拉使能控制引脚的行为.

表14-10. P1.4-P1.6配置（P14CR-P16CR）[0x11-0x13] [读/写]

位#	7	6	五	4	3	2	1	0
领域	SPI使用	Int启用	廉政法案低	3.3V驱动器	高水槽	打开排水管	上拉启用	输出启用
读/写	R / W	R / W	R / W	R / W	R / W	R / W	R / W	R / W
默认	0	0	0	0	00		0	0

这些寄存器分别控制引脚P1.4-P1.6的操作.这些寄存器存在于所有的enCoRe II部分.

位7： SPI使用

0 =禁用SPI备用功能.该引脚用作GPIO

1 =启用SPI功能. SPI电路控制引脚的输出

P1.4-P1.6 GPIO的阈值始终设置为TTL.

当SPI硬件使能时，被配置为SPI使用的引脚的输出使能和输出状态由控制
SPI电路.当SPI硬件被禁止或某个引脚的SPI使用位清零时，该引脚由输出使能控制
位和P1数据寄存器中的相应位.

无论是否使用任何引脚作为SPI或GPIO引脚，Int Enable，Int act Low，3.3V Drive，High Sink，Open Drain，
和上拉使能控制引脚的行为.

通信模式01或10（SPI主控或SPI从器件，请参阅第42页上的表15-2）

当配置为SPI（SPI使用= 1且通信模式[1：0] = SPI主模式或SPI从模式）时，输入和输出方向
引脚P1.5和P1.6由SPI逻辑自动设置.但是，引脚P1.4的输入和输出方向不是自动的
组;它必须由固件明确设置.对于SPI主模式，引脚P1.4必须配置为输出;对于SPI从模式，
引脚P1.4必须配置为输入.

表14-11. P1.7配置（P17CR）[0x14] [R / W]

位#	7	6	五	4	3	2	1	0
领域	保留的	Int启用	廉政法案低	保留的	高水槽	打开排水管	上拉启用	输出启用
读/写	-	R / W	R / W	-	R / W	R / W	R / W	R / W
默认	0	0	0	0	0	0	1	0

该寄存器控制引脚P1.7的操作.该寄存器仅存在于CY7C638（1/2/3）3中. P1.7 GPIO的门槛是
始终设置为TTL.

表14-12. P2配置（P2CR）[0x15] [R / W]

位#	7	6	五	4	3	2	1	0
领域	保留的	Int启用	廉政法案低	TTL Thresh	保留的	打开排水管	上拉启用	输出启用
读/写	-	R / W	R / W	R / W	-	R / W	R / W	R / W
默认	0	0	0	0	0	0	0	0

该寄存器仅存在于CY7C638（2/3）3中.该寄存器控制引脚P2.0-P2.1的操作.

表14-13. P3配置（P3CR）[0x16] [R / W]

位#	7	6	五	4	3	2	1	0
领域	保留的	Int启用	廉政法案低	TTL Thresh	保留的	打开排水管	上拉启用	输出启用
读/写	-	R / W	R / W	R / W	-	R / W	R / W	R / W
默认	0	0	0	0	0	0	1	0

该寄存器存在于CY7C638（2/3）3中.该寄存器控制引脚P3.0-P3.1的操作.

15.串行外设接口（SPI）

SPI主/从接口内核逻辑运行在SPI时钟域，因此其功能独立于系统时钟速度.
SPI是一个四引脚串行接口，由一个时钟，一个使能和两个数据引脚组成.

15.1 SPI数据寄存器

表15-1. SPI数据寄存器（SPIDATA）[0x3C] [R / W]

位#	7	6	五	4	3	2	1	0
领域	SPIDATA [7：0]							
读/写	R / W	R / W	R / W	R / W	R / W	R / W	R / W	R / W
默认	0	0	0	0	00		0	0

读取时，该寄存器返回接收缓冲区的内容.写入时，它加载发送保持寄存器.

位[7：0]:SPI数据[7：0]

当发生中断向固件指示接收数据字节可用或发送器保持寄存器为空时，
固件有7个SPI时钟来管理缓冲区：清空接收缓冲区或重新填充发送保持寄存器.未能
满足这个时间要求会导致不正确的数据传输.

15.2 SPI配置寄存器

表15-2. SPI配置寄存器（SPICR）[0x3D] [R / W]

位#	7	6	五	4	3	2	1	0
领域	交换	LSB第一	通信模式		CPOL	CPHA	SCLK选择	
读/写	R/W	R/W	R/W	R/W	R/W	R / W	R/W	R/W
默认	0	0	0	0	00		0	0

位7： 交换

0 =禁用交换功能.

1 = SPI模块交换使用SMOSI和SMISO.这对于实现类似于SPI的单线通信非常有用.

位6： LSB第一

0 = SPI首先发送并接收MSB（最高有效位）.

1 = SPI首先发送并接收LSB（最低有效位）.

位[5：4]:通信模式[1：0]

0 0：禁止所有SPI通信.

0 1：SPI主模式

1 0：SPI从模式

1 1：保留

位3： CPOL

该位控制SPI时钟（SCLK）空闲极性.

0 = SCLK空闲低电平

1 = SCLK空闲

位2： CPHA

时钟相位位控制数据采样时钟的相位.第43页上的表15-4显示了
LSB First，CPOL和CPHA的各种组合.

位[1：0]:SCLK选择

该字段选择主SCLK的速度.当处于主模式时，SCLK通过分频基本CPUCLK产生.

通信模式01b或10b（SPI主机或SPI从机）的注意事项
当配置为SPI时，（SPI使用= 1表40），P1.3，P1.5和P1.6的输入/输出方向被设置
由SPI逻辑自动完成.但是，引脚P1.4的输入/输出方向不会自动设置;它必须被明确地设置
固件.对于SPI主模式，引脚P1.4必须配置为输出;对于SPI从模式，引脚P1.4必须配置为
一个输入.

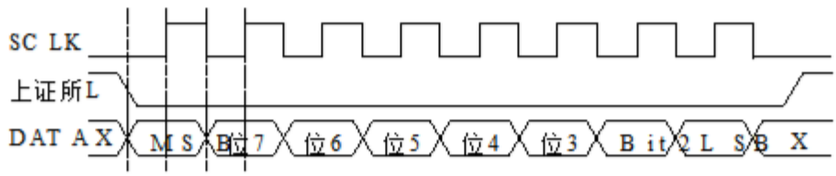
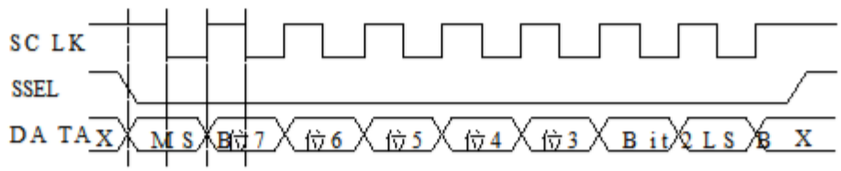
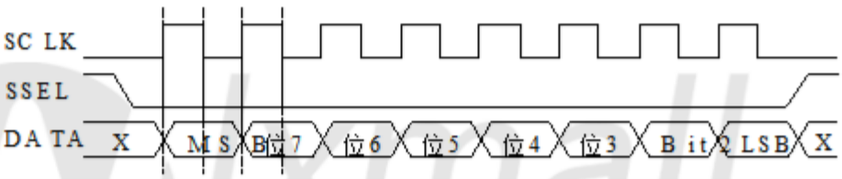
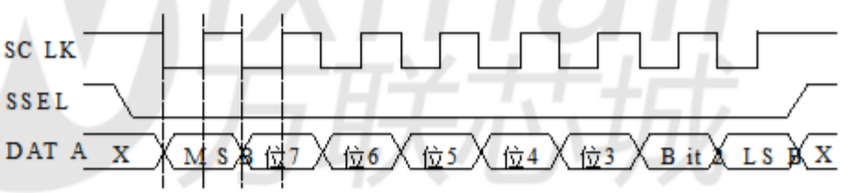
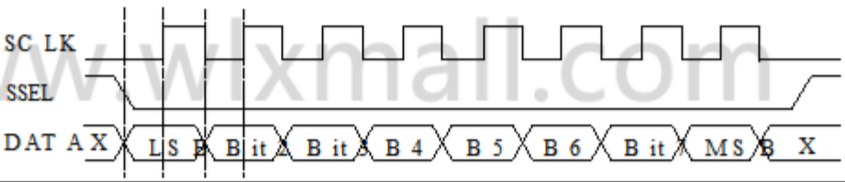
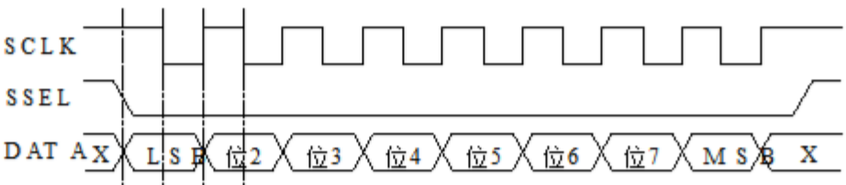
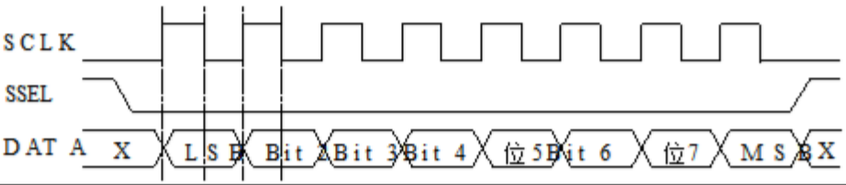
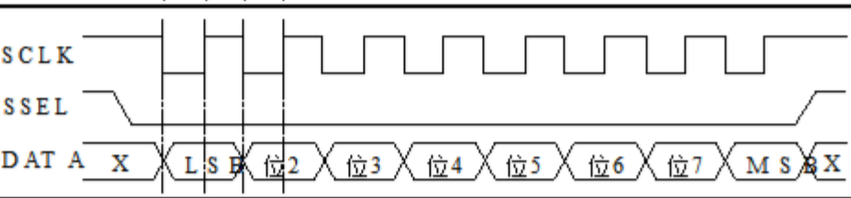
表15-3. SPI SCLK频率

SCLK选择	CPUCLK除数	当CPUCLK =时，SCLK频率	
		12 MHz	24 MHz
00	6	2 MHz	4 MHz
01	12	1 MHz	2 MHz
10	48	250千赫	500 kHz
11	96	125 kHz	250千赫

15.3 SPI接口引脚

SPI接口使用P1.3-P1.6引脚.这些引脚使用P1.3和P1.4-P1.6配置进行配置.

表15-4. SPI模式时序与LSB首先，CPOL和CPHA

LSB第—	CPHA	CPOL	图
00		0	
00		1	
01		0	
01		1	
10		0	
10		1	
11		0	
11		1	

16.定时器寄存器

enCoRe II的所有定时器功能都由一个定时器模块提供.定时器模块与CPU时钟异步.

16.1注册

16.1.1自由运行计数器

16位自由运行计数器由定时器捕捉时钟（TCAPCLK）提供时钟.它是用软件读取的，用作通用目的时间基准.当低位字节被读取时，高位字节被注册.读取高位字节读取该寄存器，允许CPU以原子方式读取16位值（一次加载所有位）.自由运行定时器在1024处产生中断 µs速率 当由一个4 MHz的时钟源.当自由运行计数器溢出每16.384毫秒发生时，它也会产生一个中断一个4 MHz的信号源）.这允许用软件延长定时器的长度.

图16-1. 16位自由运行计数器框图

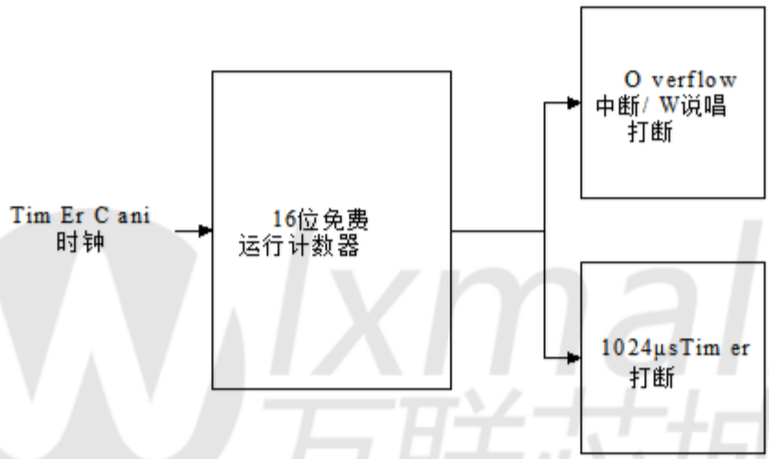


表16-1.自由运行定时器低位字节（FRTMRL）[0x20] [R / W]

位#	7	6	五	4	3	2	1	0
领域	自由运行计时器[7：0]							
读/写	R / W	R / W	R / W	R / W	R / W	R / W	R / W	R / W
默认	0	0	0	0	00		0	0

位[7：0]:自由运行计时器[7：0]

该寄存器保存16位自由运行定时器的低位字节.读这个寄存器会导致高位字节移入一个保持寄存器，允许同时自动读取所有16位数据.

对于读取，实际读取发生在读取低位时的周期中.对于写入，写入的实际时间是周期当高阶被写入时.

读取自由运行定时器时，必须先读取低位字节，然后再读取高位字节.写时，低顺序字节必须先写入高位字节.

表16-2.自由运行定时器高位字节（FRTMRH）[0x21] [R / W]

位#	7	6	五	4	3	2	1	0
领域	自由运行计时器[15：8]							
读/写	R / W	R / W	R / W	R / W	R / W	R / W	R / W	R / W
默认	0	0	0	0	00		0	0

位[7：0]:自由运行计时器[15：8]

读取自由运行定时器时，必须先读取低位字节，然后再读取高位字节.写时，低顺序字节必须先写入高位字节.

表16-3.定时器捕捉0上升（TIO0R）[0x22] [R / W]

位#	7	6	五	4	3	2	1	0
领域	捕获0上升[7：0]							
读/写	R/W	R/W	R/W	R/W	R/W	R / W	R/W	R/W
默认	0	0	0	0	00		0	0

位[7：0]:捕获0上升[7：0]

当TIO0输入发生最后一个上升沿时，该寄存器保存自由运行定时器的值.捕获0时
处于8位模式时，此处存储的位由定时器配置寄存器中的预分频比[2：0]位选择.什么时候
捕捉0处于16位模式，该寄存器保存16位定时器的低8位.

表16-4.定时器捕获1上升（TIO1R）[0x23] [R / W]

位#	7	6	五	4	3	2	1	0
领域	捕获1上升[7：0]							
读/写	R/W	R/W	R/W	R/W	R/W	R / W	R/W	R/W
默认	0	0	0	0	00		0	0

位[7：0]:捕获1上升[7：0]

当8位模式下TIO1输入发生最后一个上升沿时，该寄存器保存自由运行定时器的值.
此处存储的位由定时器配置寄存器中的Prescale [2：0]位选择.当捕获0处于
16位模式该寄存器保存最后一个捕捉0上升沿的16位定时器的高8位.

表16-5.定时器捕捉0下降（TIO0F）[0x24] [R / W]

位#	7	6	五	4	3	2	1	0
领域	捕捉0下降[7：0]							
读/写	R/W	R/W	R/W	R/W	R/W	R / W	R/W	R/W
默认	0	0	0	0	00		0	0

位[7：0]:捕捉0下降[7：0]

当TIO0输入发生最后一个下降沿时，该寄存器保存自由运行定时器的值.捕获0时
处于8位模式时，此处存储的位由定时器配置寄存器中的预分频比[2：0]位选择.什么时候
捕捉0处于16位模式，该寄存器保存16位定时器的低8位.

表16-6.定时器捕捉1下降（TIO1F）[0x25] [R / W]

位#	7	6	五	4	3	2	1	0
领域	捕获1下降[7：0]							
读/写	R/W	R/W	R/W	R/W	R/W	R / W	R/W	R/W
默认	0	0	0	0	00		0	0

位[7：0]: Capture 1Falling [7：0]
当8位模式下TIO1输入发生最后一个下降沿时，该寄存器保存自由运行定时器的值.
此处存储的位由定时器配置寄存器中的Prescale [2：0]位选择.当捕捉0处于
该寄存器保存16位定时器从上一个捕捉0下降沿的高8位.

表16-7.可编程间隔定时器低电平（PITMRL）[0x26] [R]

位#	7	6	五	4	3	2	1	0
领域	程序间隔定时器[7：0]							
读/写	[R	[R	[R	[R	RR		[R	[R
默认	0	0	0	0	00		0	0

位[7：0]:程序间隔定时器[7：0]

该寄存器保存12位可编程间隔定时器的低位字节.读取该寄存器会导致高位字节
被移入保持寄存器，允许同时自动读取全部12位.

表16-8.可编程间隔定时器高电平（PITMRH）[0x27] [R]

位#	7	6	五	4	3	2	1	0
领域	保留的				程序间隔定时器[11：8]			
读/写	-	-	-	-	RR		[R	[R
默认	0	0	0	0	00		0	0

位[7：4]:保留的

位[3：0]: Prog内部定时器[11：8]

该寄存器保存12位可编程间隔定时器的高位半字节.读这个寄存器返回高位
在最后一次读取低位字节的瞬间，轻点12位定时器.

表16-9.可编程时间间隔重载低（PIRL）[0x28] [R / W]

位#	7	6	五	4	3	2	1	0
领域	程序间隔[7：0]							
读/写	R/W	R/W	R/W	R/W	R/W	R / W	R/W	R/W
默认	0	0	0	0	00		0	0

位[7：0]:程序间隔[7：0]

该寄存器保存定时器的低8位.写入12位重加载寄存器时，先写入低位字节，然后写入高位
蚕食.

表16-10.可编程时间间隔重载（PIRH）[0x29] [R / W]

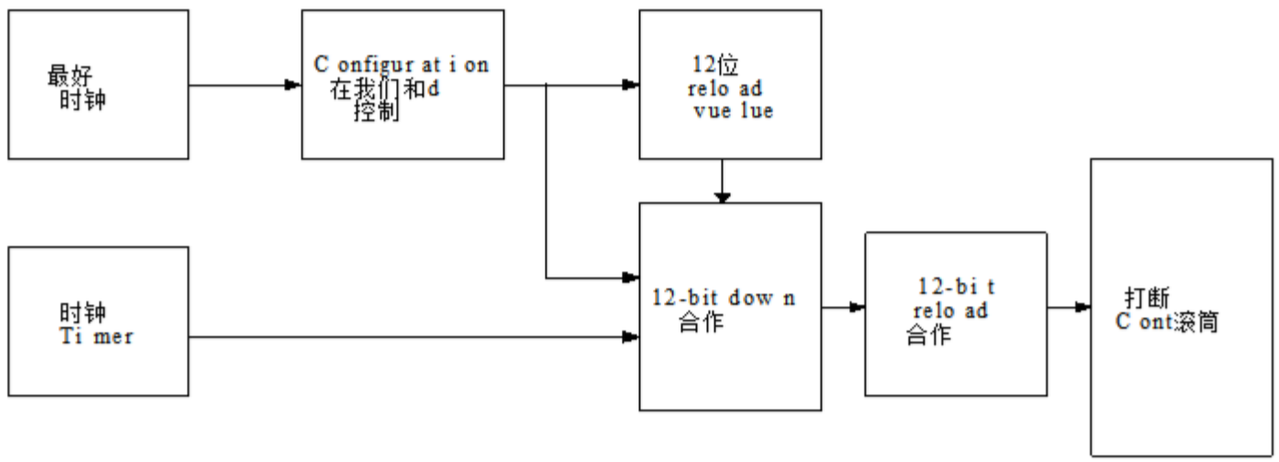
位#	7	6	五	4	3	2	1	0
领域	保留的				Prog Interval [11：8]			
读/写	-	-	-	-	R/W	R / W	R/W	R/W
默认	0	0	0	0	00		0	0

位[7：4]:保留的

位[3：0]: Prog Interval [11：8]

该寄存器保存定时器的高4位.在写入12位重加载寄存器时，首先写入低字节，然后写入高字节
蚕食.

图16-2.可编程间隔定时器框图



16.1.2定时器捕捉

赛普拉斯enCoRe II具有两个8位捕获.每个捕捉在上升和下降时间都有单独的寄存器.两个八位捕获可以配置为单个16位捕捉.配置时，捕捉1寄存器保存16位定时器的高位字节捕获价值.四个捕获寄存器中的每一个都可以通过编程在加载时产生中断.

表16-1.定时器配置（TMRCR）[0x2A] [R / W]

位#	7	6	五	4	3	2	1	0
领域	第一边缘举行	8位捕捉预分频[2：0]			Cap0 16bit启用	保留的		
读/写	R / W	R / W	R / W	R / W	R / W	-	-	-
默认	0	0	0	0	00		0	0

位7： 第一边缘举行

第一个边沿保持功能适用于所有四个捕捉定时器.

0 =最近沿的时间保存在捕捉定时器数据寄存器中.如果自阅读以来出现多个边缘捕获计时器，读取最近一次的时间.

1 =在捕捉定时器数据寄存器中保持首次出现边沿的时间，直到读取数据.随后在读取捕捉定时器数据寄存器之前边沿被忽略.

位[6：4]:8位捕捉预分频[2：0]

该位控制位模式下16位自由运行定时器的哪些8位被捕获.

0 0 0 =捕捉定时器[7：0]

0 0 1 =捕捉定时器[8：1]

0 1 0 =捕捉定时器[9：2]

0 1 1 =捕捉定时器[10：3]

1 0 0 =捕捉定时器[11：4]

1 0 1 =捕捉定时器[12：5]

1 1 0 =捕捉定时器[13：6]

1 1 1 =捕捉定时器[14：7]

位3： Cap0 16位使能

0 =禁止捕获0 16位模式

1 =使能捕获0 16位模式.捕捉1被禁止，捕捉1的上升和下降寄存器被用作扩展到捕获0寄存器 - 将它们扩展到16位

位[2：0]:保留的

表16-2.捕捉中断使能（TCAPINTE）[0x2B] [R / W]

位#	7	6	五	4	3	2	1	0
领域	保留的				Cap1秋季启用	Cap1上升启用	Cap0秋季启用	Cap0上升启用
读/写	----				R / W	R / W	R / W	R / W
默认	0	0	0	0	00		0	0

位[7：4]:保留的

位3： 启用Cap1 Fall
0 =禁止捕获1下降沿中断
1 =使能捕捉1下降沿中断

位2： Cap1上升启用
0 =禁止捕捉1上升沿中断
1 =使能捕捉1上升沿中断

位1： Cap0下降使能
0 =禁止捕获0下降沿中断
1 =使能捕捉0下降沿中断

位0： Cap0上升启用
0 =禁止捕获0上升沿中断
1 =使能捕捉0上升沿中断

表16-3.捕捉中断状态（TCAPINTS）[0x2C] [R / W]

位#	7	6	五	4	3	2	1	0
领域	保留的				TIO1秋季活性	TIO1上升活跃	TIO0秋季活性	TIO0上升活跃
读/写	----				R / W	R / W	R / W	R / W
默认	0	0	0	0	00		0	0

位[7：4]:保留的

位3： TIO1处于活动状态
0 =没有事件
1 = TIO1上出现下降沿

位2： TIO1上升活跃
0 =没有事件
1 = TIO1上出现上升沿

位1： TIO0开始工作
0 =没有事件
1 = TIO0上出现下降沿

位0： TIO0上升活跃
0 =没有事件
1 = TIO0发生上升沿

注意 The interrupt status bits must be cleared by firmware to enable subsequent interrupts. This is achieved by writing a '1' to 相应的中断状态位.

图16-3.定时器功能序列图

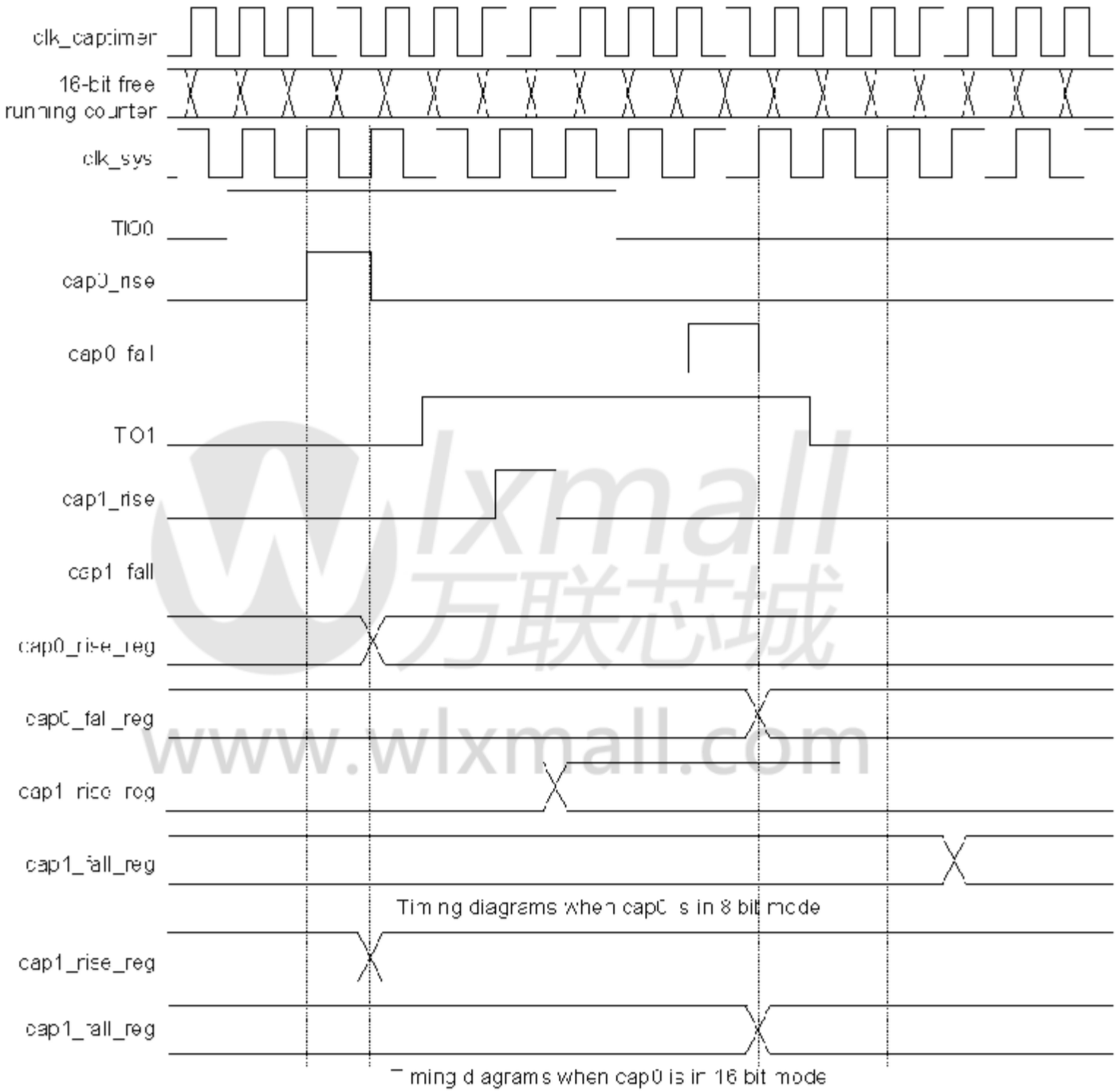


图16-4. 16位自由运行计数器加载时序图

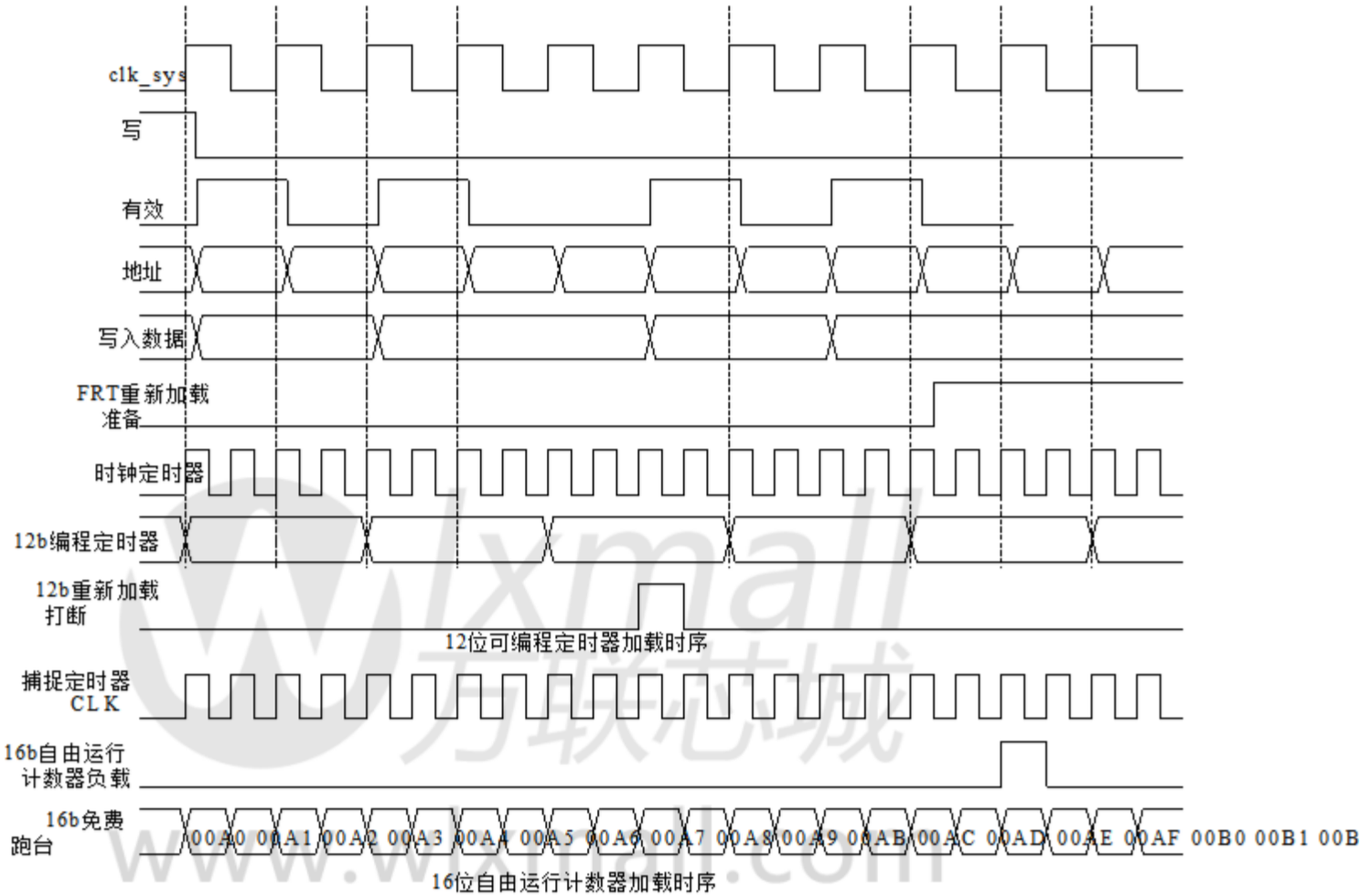
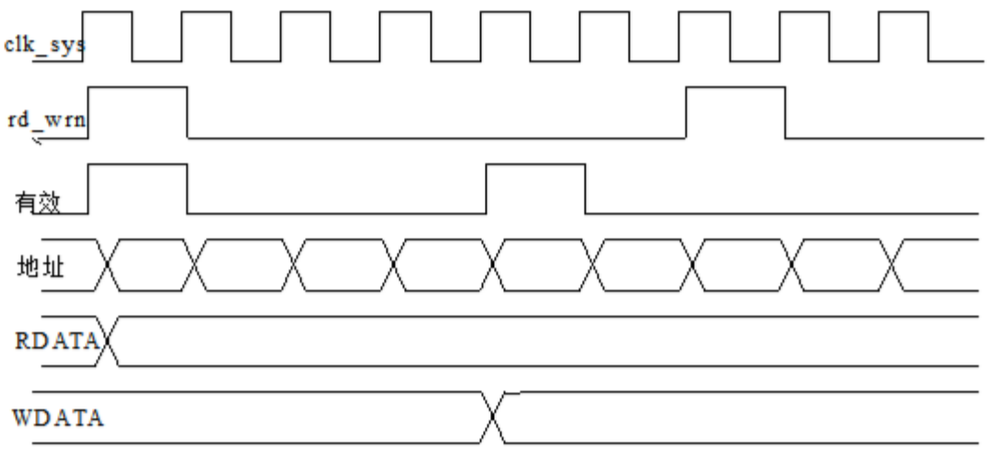


图16-5.存储器映射寄存器读/写时序图



17.中断控制器

中断控制器及其相关的寄存器允许
用户的代码来响应几乎每一个中断
功能块在enCoRe II设备中.寄存器
与中断控制器相关联允许禁止中断
全球或个人.寄存器还提供了一种机制
用户可以通过它清除所有挂起和发布的中断，或者
清除单个发布或挂起的中断.

下表列出了所有中断和优先级
可用于enCoRe II设备.

表17-1.中断号码，优先权，向里

打断 优先	打断 地址	名称
0	0000H	重启
1	0004H	POR / LVD
2	0008H	INT0
3	000CH	SPI发送器空了
4	0010H	SPI接收器已满
五	0014H	GPIO端口0
6	位于0018	GPIO端口1
7	001Ch	INT1
8	0020H	EP0
9	0024H	EP1
10	0028H	EP2
11	002Ch	USB重置
12	0030H	USB活动
13	0034H	1毫秒间隔计时器
14	0038H	可编程间隔定时器
15	003CH	定时器捕获0
16	0040H	定时器捕捉1
17	0044h	16位自由运行计时器包装
18	0048h	INT2
19	004Ch	PS2数据低
20	0050H	GPIO端口2
21	0054h	GPIO端口3
22	0058h	保留的
23	005Ch	保留的
24	0060H	保留的
25	0064H	睡眠定时器

17.1建筑描述

发生中断时发布中断.这个
results in the flip-flop in Figure 17-1 on page 52 clocking in a '1'.
中断保持发布状态，直到中断或直到中断为止
通过写入适当的INT_CLRx寄存器来清除.

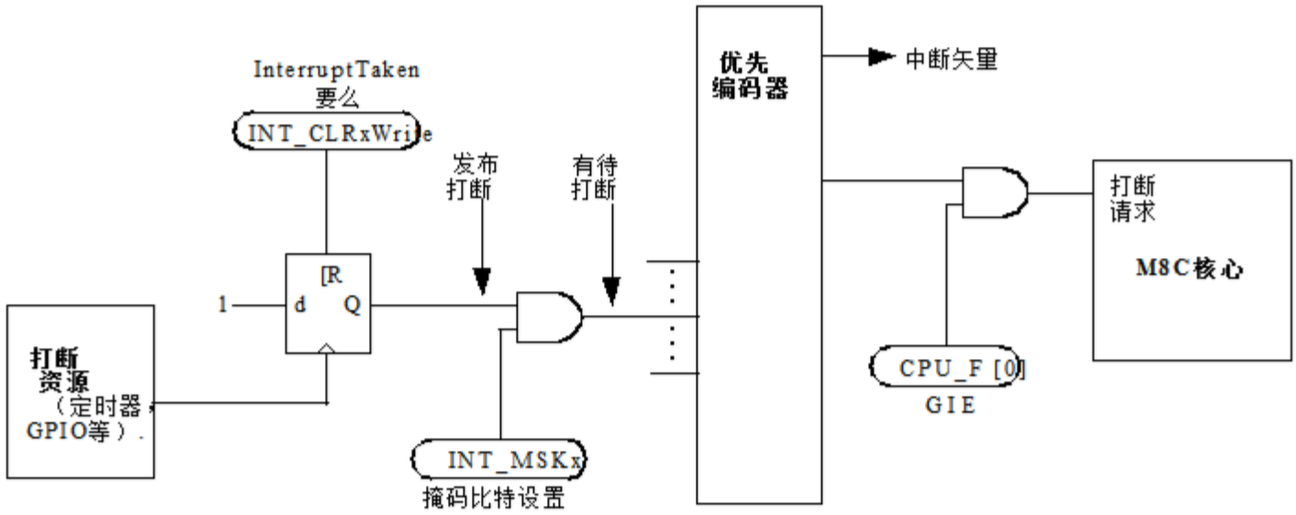
发布的中断不是挂起的，除非它通过设置启用
其中断屏蔽位（在相应的INT_MSKx寄存器中）.所有
待处理中断由优先编码器处理
确定M8C采取的最高优先级中断
如果全局中断使能位设置在CPU_F寄存器中.

通过清除其中断屏蔽位来禁止中断（在
INT_MSKx寄存器）不会清除发布的中断，也不会
它防止发布中断.它阻止了发布
中断不成为挂起.

嵌套中断通过重新启用中断来完成
在中断服务程序中.为此，请在IE中设置IE位
国旗注册.

enCoRe II中断控制器的框图如图所示
图17-1在52页.

图17-1.中断控制器框图



17.2中断处理

中断处理期间发生的事件序列
如下：

1.中断变为活动状态，因为：
一个.发生中断情况（例如，计时器到期）.
湾之前发布的中断通过更新启用
一个中断屏蔽寄存器.
C.中断处于挂起状态，GIE在CPU中设置为0到1
国旗注册.

1.当前执行指令结束.

2.发出内部中断，共13个周期.中
这次会发生以下操作：MSB和LSB
程序计数器和标志寄存器（CPU_PC和CPU_F）
通过自动CALL存储到程序堆栈中
指令（13个周期）在中断期间产生
确认过程.
一个. PCH，PCL和标志寄存器（CPU_F）存储在上面
程序堆栈（按此顺序）通过自动CALL
指令（13个周期）在中断期间产生
确认过程
湾 CPU_F寄存器然后被清除.因为这清除了
GIE位为0时，其他中断暂时被禁止.
C. PCH（PC [15：8]）清零.
d.中断向量是从中断控制器读取的
其值放入PCL（PC [7：0]）.这设置程序
计数器指向中断中的适当地址
表（例如，用于POR / LVD中断的0004h）.

1.将执行向量编程到中断表.通常，a
LJMP指令在中断表中发送执行到
用户的中断服务程序（ISR）.

2. ISR执行.请注意，中断被禁用，因为
GIE = 0.在ISR中，通过设置重新启用中断
GIE = 1（必须小心避免堆栈溢出）.

3. ISR以恢复指令的RETI指令结束
程序计数器和标志寄存器（CPU_PC和CPU_F）.
恢复的标志寄存器重新启用中断，因为
GIE = 1.

4.执行从下一条指令开始，继续执行
在中断之前发生.但是，如果还有更多
待处理的中断将处理随后的中断
在下一个正常程序指令之前.

17.3中断触发条件

大多数中断的触发条件请参见第51页的表17-1
已在相关章节中进行了解释.然而，
USB Active（中断地址0030h）
和PS2数据低（中断地址004Ch）中断
触发的解释如下.

1. USB激活中断：当D +/-线路处于a时触发
非空闲状态，即K状态或SE0状态.
2. PS2数据低中断：当SDATA变低时触发
当SDATA焊盘至少处于输入模式6-7时
32 kHz周期.
3. GPIO中断是边沿触发的.

17.4中断延迟

断言使能中断和发生中断之间的时间
其ISR的开始由以下等式计算.

延迟=当前指令完成的时间+内部时间
中断程序执行+ LJMP指令的时间
中断表来执行.

例如，如果执行5个周期的JMP指令，
中断变为活动状态，CPU时钟周期总数
ISR开始之前如下：

（JMP完成1至5个周期）+（中断例程13个周期）
+（LJMP的7个周期）= 21到25个周期.

在前面的例子中，在24 MHz时需要25个时钟周期
1.042 微秒.

17.5中断寄存器

中断清除寄存器（INT_CLRx）用于使各个中断源能够清除发送的中断.

当INT_CLRx寄存器被读取时，任何被设置的位表示已经为该硬件资源发布了中断.因此，
读取这些寄存器可以让用户确定所有发布的中断.

表17-1.中断清除0（INT_CLR0）[0xDA] [R / W]

位#	7	6	五	4	3	2	1	0
领域	GPIO端口1	睡眠定时器	INT1	GPIO端口0	SPI接收	SPI传输	INT0	POR / LVD
读/写	R/W	R/W	R/W	R/W	R/W	R / W	R/W	R/W
默认	0	0	0	0	00		0	0

在阅读本注册时，
0 =相应硬件没有发布中断
1 =存在相应硬件的已发布中断

Writing a '0' to the bits clears the posted interrupts for the corresponding hardware. Writing a '1' to the bits AND to the ENSWINT（INT_MSK3寄存器的位7）发送相应的硬件中断.

表17-2.中断清除1（INT_CLR1）[0xDB] [R / W]

位#	7	6	五	4	3	2	1	0
领域	TCAP0	程序间隔计时器	1毫秒定时器	USB活动	USB重置	USB EP2	USB EP1	USB EP0
读/写	R/W	R/W	R/W	R/W	R/W	R / W	R/W	R/W
默认	00000000							

在阅读本注册时，
0 =相应硬件没有发布中断.
1 =存在相应硬件的已发布中断.

Writing a '0' to the bits clears the posted interrupts for the corresponding hardware. Writing a '1' to the bits and to the ENSWINT（INT_MSK3寄存器的位7）发送相应的硬件中断.

表17-3.中断清除2（INT_CLR2）[0xDC] [R / W]

位#	7	6	五	4	3	2	1	0
领域	保留的	保留的	GPIO端口3	GPIO端口2	PS / 2数据低	INT2	16位计数器包	TCAP1
读/写	R/W	R/W	R/W	R/W	R/W	R / W	R/W	R/W
默认	0	0	0	0	00		0	0

在阅读本注册时，

0 =相应硬件没有发布中断.

1 =存在相应硬件的已发布中断.

Writing a '0' to the bits clears the posted interrupts for the corresponding hardware. Writing a '1' to the bits AND to the ENSWINT（INT_MSK3寄存器的位7）发送相应的硬件中断.

17.5.1中断屏蔽寄存器

中断屏蔽寄存器（INT_MSKx）使能个人
中断源创建挂起中断的能力.

有四个中断屏蔽寄存器（INT_MSK0，
INT_MSK1，INT_MSK2和INT_MSK3）
一般来说就是INT_MSKx.如果清零，则INT_MSKx中的每一位
寄存器防止发布的中断成为挂起
中断（输入到优先级编码器）.但是，中断可以
即使其掩码位为零，也仍会发布.所有INT_MSKx位都是
独立于所有其他INT_MSKx位.

如果INT_MSKx位置1，则与中断源关联
该屏蔽位可能会产生一个中断，成为挂起
打断.

INT_MSK3 [7]中的使能软件中断（ENSWINT）
决定写入一个单独的位值的方式
INT_CLRx寄存器被解释.当它被清除时，写入1
到INT_CLRx寄存器不起作用.但是，写0到一个
INT_CLRx寄存器，当ENSWINT被清除时，会导致
相应的中断清除.如果ENSWINT位置位，则任何
写入INT_CLRx寄存器的0被忽略.但是，1s
写入INT_CLRx寄存器，当设置ENSWINT时，会导致
一个中断来发布相应的中断.

软件中断可以帮助调试中断服务
通过消除创建系统级互动的需要
有时仅需要创建一个硬件
打断.

表17-4.中断屏蔽3（INT_MSK3）[0xDE] [R / W]

位#	7	6	五	4	3	2	1	0
领域	ENSWINT	保留的						
读/写	R / W	-	-	-	-	-	-	-
默认	0	0	0	0	00		0	0

位7： 启用软件中断（ENSWINT）

0 =禁用.写入0到INT_CLRx寄存器，当ENSWINT被清除时，使相应的中断清零

1 =启用.将1写入INT_CLRx寄存器（当ENSWINT置位时）会导致相应的中断发布.

位[6：0]:保留的

表17-5.中断屏蔽2（INT_MSK2）[0xDF] [R / W]

位#	7	6	五	4	3	2	1	0
领域	保留的	保留的	GPIO端口3 Int启用	GPIO端口2 Int启用	PS / 2数据低 Int启用	INT2 Int启用	16位计数器 自动换行启用	TCAP1 Int启用
读/写	-	R / W	R / W	R / W	R / W	R / W	R / W	R / W
默认	0	0	0	0	00		0	0

位7： 保留的

位6： GPIO端口4中断使能

0 =屏蔽GPIO端口4中断

1 =取消屏蔽GPIO端口4中断

位5： GPIO端口3中断使能

0 =屏蔽GPIO端口3中断

1 =取消屏蔽GPIO端口3中断

位4： GPIO端口2中断使能

0 =屏蔽GPIO端口2中断

1 =取消屏蔽GPIO端口2中断

位3： PS / 2数据低电平中断使能

0 =屏蔽PS / 2数据低中断

1 =取消屏蔽PS / 2数据低电平中断

位2： INT2中断使能

0 =屏蔽INT2中断

1 =取消屏蔽INT2中断

位1： 16位计数器封装中断使能

0 =掩码16位计数器绕回中断

1 =取消屏蔽16位计数器绕回中断

位0： TCAP1中断使能

0 =屏蔽TCAP1中断

1 =取消屏蔽TCAP1中断

表17-6.中断屏蔽1（INT_MSK1）[0xE1] [R / W]

位#	7	6	五	4	3	2	1	0
领域	TCAP0 Int启用	程序间隔计时器 Int启用	1毫秒定时器 Int启用	USB活动 Int启用	USB重置 Int启用	USB EP2 Int启用	USB EP1 Int启用	USB EP0 Int启用
读/写	R / W	R / W	R / W	R / W	R / W	R / W	R / W	R / W
默认	0	0	0	0	00		0	0

位7： TCAP0中断使能

0 =屏蔽TCAP0中断

1 =取消屏蔽TCAP0中断

位6： 编程间隔定时器中断使能

0 =掩码程序间隔定时器中断

1 =取消屏蔽程序间隔定时器中断

位5： 1毫秒定时器中断使能

0 =屏蔽1毫秒中断

1 =取消屏蔽1毫秒中断

位4： USB主动中断使能

0 =屏蔽USB活动中断

1 =取消屏蔽USB活动中断

位3： USB复位中断使能

0 =屏蔽USB复位中断

1 =解除USB复位中断

位2： USB EP2中断使能

0 =屏蔽EP2中断

1 =取消屏蔽EP2中断

位1： USB EP1中断使能

0 =屏蔽EP1中断

1 =取消屏蔽EP1中断

位0： USB EP0中断使能

0 =屏蔽EP0中断

1 =取消屏蔽EP0中断

表17-7.中断屏蔽0（INT_MSK0）[0xE0] [R / W]

位#	7	6	五	4	3	2	1	0
领域	GPIO端口1 Int启用	睡眠定时器 Int启用	INT1 Int启用	GPIO端口0 Int启用	SPI接收 Int启用	SPI传输 Int启用	INT0 Int启用	POR / LVD Int启用
读/写	R / W	R / W	R / W	R / W	R / W	R / W	R / W	R / W
默认	0	0	0	0	00		0	0

位7： GPIO端口1中断使能

0 =屏蔽GPIO端口1中断

1 =取消屏蔽GPIO端口1中断

位6： 睡眠定时器中断使能

0 =屏蔽睡眠定时器中断

1 =取消屏蔽睡眠定时器中断

位5： INT1中断使能

0 =屏蔽INT1中断

1 =取消屏蔽INT1中断

位4： GPIO端口0中断使能

0 =屏蔽GPIO端口0中断

1 =取消屏蔽GPIO端口0中断

位3： SPI接收中断使能

0 =屏蔽SPI接收中断

1 =取消屏蔽SPI接收中断

位2： SPI发送中断使能

0 =屏蔽SPI发送中断

1 =取消屏蔽SPI发送中断

位1： INT0中断使能

0 =屏蔽INT0中断

1 =取消屏蔽INT0中断

位0： POR / LVD中断使能

0 =屏蔽POR / LVD中断

1 =取消屏蔽POR / LVD中断

表17-8.中断向里清除寄存器（INT_VC）[0xE2] [R / W]

位#	7	6	五	4	3	2	1	0
领域	待处理中断[7：0]							
读/写	R / W	R / W	R / W	R / W	R / W	R / W	R / W	R / W
默认	0	0	0	0	00		0	0

中断向里清除寄存器（INT_VC）在读取时保持最高优先级待处理中断的中断向里
当写入清除所有挂起的中断.

位[7：0]:待处理中断[7：0]

8位数据值保存最高优先级待处理中断的中断向里.写入这个寄存器清除所有未决的
中断.

18.调节器输出

18.1 VREG控制

表18-1. VREG控制寄存器（VREGCR）[0x73] [R / W]

位#	7	6	五	4	3	2	1	0
领域	保留的						活着	VREG启用
读/写	-	-	-	-	-	-	R / W	R / W
默认	0	0	0	0	00		0	0

位[7：2]：保留的

位1： 活着

保持活动状态，设置时，允许电压调节器在电压调节器禁用时提供高达20µA的电流.
P12CR [0]，P12CR [7]必须清零.

0 =禁用

1 =启用

位0： VREG启用

该位打开3.3 V电压调节器. 当V CC高于4.35 V时，电压调节器仅在规格内运行 .
当V CC低于4.35V时不能使能该模块 - 尽管如果在4.35 V以下使能，则不会发生损坏或不规则.

0 =禁止VREG / P1.2引脚上的3.3 V电压调节器输出.

1 =使能VREG / P1.2引脚上的3.3 V电压调节器输出. P1.2的GPIO功能被禁用.

注意 在引脚P1.3-P1.6上使用备用驱动器需要将VREG使能位设置为使能稳压器并提供
交替电压.

19. USB / PS2收发器

尽管USB收发器具有帮助连接到PS / 2的功能，但这些功能不受这些寄存器控制.该
寄存器只控制USB接口功能. PS / 2接口选项由D +和D- GPIO配置控制
寄存器（请参见第35页上的表14-2）.

19.1 USB收发器配置

表19-1. USB收发器配置寄存器（USBXCR）[0x74] [R / W]

位#	7	6	五	4	3	2	1	0
领域	USB上拉启用	保留的						USB强制状态
读/写	R / W	-	-	-	-	-	-	R / W
默认	0	0	0	0	00		0	0

位7： USB上拉使能

0 =禁用D-上拉电阻，

1 =使能D-上的上拉电阻. 如果PHY的内部电压调节器未启用或内部上 拉，则该上拉为V CC
当VREG使能时产生3.3 V电压.

位[6：1]：保留的

位0： USB强制状态

该位允许当USB使能时，USB IO引脚D-和D +的状态被强制为一个状态.

0 =禁用USB强制状态

1 =启用USB强制状态.当USBIO处于输入状态时，允许D-和D +引脚分别由P1.1和P1.0控制
USB模式.有关更多信息，请参阅第35页的表14-2.

注意 USB收发器具有专用的3.3 V稳压器，用于USB信号传输，并提供1.5 KD上拉.
与其他3.3 V稳压器不同，该稳压器不能被固件控制或访问.当设备暂停时，
该调节器随带隙（向调节器提供参考电压）和D-线路一起被禁用
通过另一个6.5 K电阻上拉至5 V.在暂停后苏醒过程中，带隙和调节器是
以任何顺序开机.在极端罕见的情况下，当总线复位条件唤醒后，
年龄调节器和带隙以该特定顺序打开时，D-电极上可能出现毛刺或低脉冲，
线.主人可能会误解为分离条件.这种情况虽然很少见，但可以通过保持带隙电路来避免，
cuitry enabled during sleep. This is achieved by setting the 'No Buzz' bit, bit[5] in the OSC_CR0 register. This is an issue only
如果设备在总线重置条件下进入休眠状态.

20. USB串行接口引擎（SIE）

SIE允许微控制器与USB通信
低速数据速率（1.5 Mbps）的主机. SIE简化了
微控制器和USB之间的接口通过合并 -
评级硬件可处理以下USB总线活动
独立于微控制器.

- 转换编码的接收数据并格式化数据
  在公共汽车上传输.
- CRC检查和生成. 如果出现错误，则标记微控制器
  在传输中存在.
- 地址检查. 忽略未发往的交易
  装置.
- 发送适当的ACK / NAK / STALL握手.
- 令牌类型标识（SETUP，IN或OUT）. 设置
  接收到有效令牌后的适当令牌位.
- 将有效的接收数据放在适当的端点FIFO中.
- 发送并更新数据切换位（Data1 / 0）.
- 位填充和拆分.

需要固件才能处理USB接口的其余部分
以下任务：

- 通过解码USB设备请求来协调枚举.
- 填充并清空FIFO.
- 暂停和恢复协调.
- 验证并选择数据切换值.

21. USB设备

21.1 USB设备地址

表21-1. USB设备地址（USBCR）[0x40] [R / W]

位#	7	6	五	4	3	2	1	0
领域	USB启用	设备地址[6：0]						
读/写	R / W	R / W	R / W	R / W	R / W	R / W	R / W	R / W
默认	0	0	0	0	00		0	0

位7： USB启用

在串行接口引擎（SIE）响应指定地址处的USB通信之前，该位必须由固件启用
在设备地址[6：0]中.该位清零时，USB收发器进入掉电状态.用户的固件必须清除它
在进入睡眠模式以节省电力之前.

0 =禁用USB设备地址并使USB收发器进入掉电状态.

1 =启用USB设备地址并将USB收发器置于正常工作模式.

位[6：0]:设备地址[6：0]

这些位必须在USB枚举过程中（即SetAddress）由固件设置为分配的非零地址
由USB主机.

21.2端点0,1和2计数

表21-2.端点0,1和2计数（EP0CNT-EP2CNT）[0x41，0x43，0x45] [R / W]

位#	7	6	五	4	3	2	1	0
领域	数据切换	数据有效	保留的		字节计数[3：0]			
读/写	R / W	R / W	R / W	R / W	R / W	R / W	R / W	R / W
默认	0	0	0	0	00		0	0

位7： 数据切换

该位选择数据包的切换状态.对于IN事务，固件必须设置该位来选择传输的数据
切换.对于OUT或SETUP事务，硬件将此位设置为接收到的数据切换位的状态.

0 = DATA0

1 = DATA1

位6： 数据有效

该位仅用于OUT和SETUP标记.如果发生CRC，bitstuff或PID错误，则该位被清除为0.这一点
不会更新某些端点模式设置.

0 =数据无效.如果启用，即使收到无效数据，也会发生端点中断.

1 =数据有效

位[5：4]:保留的

位[3：0]:字节计数位[3：0]

字节计数位指示事务中数据字节的数量：对于IN事务，固件会将计数与数字一起加载
的字节从端点FIFO发送到主机.有效值为0到8（含）.对于OUT或SETUP交易，
计数由硬件更新为接收到的数据字节数，再加上2个CRC字节.有效值为2-10（含）.

对于端点0计数寄存器，当计数从SETUP或OUT事务更新时，计数寄存器锁定并且不能
由CPU编写.读寄存器可以解锁它.这可以防止固件覆盖状态更新.

21.3端点0模式

由于固件和SIE都允许写入端点0模式和计数寄存器，因此SIE提供了互锁
防止意外覆盖数据的机制.

当SIE写入这些寄存器时，它们被锁定，并且处理器在读取它们之前不能写入它们.写入
无论写入的值是多少，该寄存器都会清除高四位.

表21-3.端点0模式（EP0MODE）[0x44] [R / W]

位#	7	6	五	4	3	2	1	0
领域	收到设置	IN收到	OUT收到	确认Trans	模式[3：0]			
读/写	R / C [5]	R / C [5]	R / C [5]	R / C [5]	R/W	R / W	R/W	R/W
默认	0	0	0	0	00		0	0

位7： 收到设置

当收到一个有效的SETUP数据包时，该位由硬件置位.它从数据包开始阶段强制为高电平
SETUP事务直到控制写入传输的数据阶段结束，并且在此间隔期间无法清除.
该位设置为1时，CPU不能写入EP0 FIFO.这可以防止固件覆盖传入的SETUP
在固件有机会读取SETUP数据之前进行交易.

该位由任何非锁定写入寄存器清除.

0 =没有收到SETUP
1 =接收到SETUP

位6： IN收到

This bit when set indicates a valid IN packet has been received. This bit is updated to '1' after the host acknowledges an IN data
包.清除时，表示未收到IN或主机未通过发送ACK来确认IN数据
握手.

该位由任何非锁定写入寄存器清除.

0 =没有收到
1 =接收到IN

位5： OUT收到

This bit when set indicates a valid OUT packet has been received and ACKed. This bit is updated to '1' after the last received
数据包在一个OUT事务中.清除时，表示没有收到OUT.

该位由任何非锁定写入寄存器清除.

0 =没有收到OUT
1 =收到OUT

位4： 确认交易

The ACK'd transaction bit is set when the SIE engages in a transaction to the register's endpoint, which completes with a ACK
包.

该位由任何非锁定写入寄存器清除.

1 =事务以ACK完成.
0 =事务没有完成ACK.

位[3：0]:模式[3：0]

端点模式确定SIE如何响应主机发送到端点的USB流量.模式控制
USB SIE如何响应流量，以及USB SIE如何将主机数据包的端点模式更改为
端点.

注意
5. C =清除.该位仅由用户清除，不能由固件设置.

21.4端点1和2模式

表21-4.端点1和2模式（EP1MODE - EP2MODE）[0x45，0x46] [读/写]

位#	7	6	五	4	3	2	1	0
领域	摊子	保留的	NAK Int启用	ACK'd 交易	模式[3：0]			
读/写	R/W	R/W	R/W	R/C（注4）	R/W	R / W	R/W	R/W
默认	0	0	0	0	00		0	0

位7： 摊子

当该位置位时，如果模式位设置为ACK-OUT，SIE会暂停一个OUT数据包，并且如果模式位设置为ACK-IN.对于所有其他模式，该位必须清除

位6： 保留的

位5： NAK Int启用

当该位置位时，即使传输以NAK结束，也会产生端点中断.与enCoRe不同，除非设置了此位，否则enCoRe II系列成员不会在这些条件下生成端点中断.

0 = Disable interrupt on NAK'd transactions
1 = Enable interrupt on NAK'd transaction

位4： 确认交易

The ACK'd transaction bit is set when the SIE engages in a transaction to the register's endpoint that completes with an ACK 包.

该位由寄存器的任何写入清除.

0 =事务没有完成ACK
1 =事务以ACK完成

位[3：0]:模式[3：0]

端点模式决定SIE如何响应主机发送到端点的USB流量.该模式控制如何USB SIE响应流量，以及USB SIE如何通过主机数据包更改该端点的模式端点.

注意 当SIE写入EP1MODE或EP2MODE寄存器时，它会阻止固件写入EP2MODE或EP1MODE寄存器（如果两个写入发生在相同的时钟周期）.这是因为设计只使用一个common 'update' signal for both EP1MODE and EP2MODE registers. As a result, when SIE writes to say EP1MODE register, 更新信号被置位，这会防止固件写入EP2MODE寄存器. SIE写入端点模式寄存器比固件写入更高的优先级.这种模式寄存器写入块的情况会使端点处于不正确的模式.固件如果读取的值不正确，则必须在固件写入后立即读取EP1 / 2MODE寄存器并重新写入.

表21-5.端点0数据（EP0DATA）[0x50-0x57] [R / W]

位#	7	6	五	4	3	2	1	0
领域	端点0数据缓冲区[7：0]							
读/写	R/W	R/W	R/W	R/W	R/W	R / W	R/W	R/W
默认	未知	未知	未知	未知	未知	未知	未知	未知

端点0缓冲区由位于地址0x50至0x57的8个字节组成.

表21-6.端点1数据（EP1DATA）[0x58-0x5F] [R / W]

位#	7	6	五	4	3	2	1	0
领域	端点1数据缓冲区[7：0]							
读/写	R/W	R/W	R/W	R/W	R/W	R / W	R/W	R/W
默认	未知	未知	未知	未知	未知	未知	未知	未知

端点1缓冲区由位于地址0x58至0x5F的8个字节组成.

表21-7.端点2数据（EP2DATA）[0x60-0x67] [读/写]

位#	7	6	五	4	3	2	1	0
领域	端点2数据缓冲区[7：0]							
读/写	R/W	R / W	R/W	R / W	R/W	R / W	R/W	R / W
默认	未知	未知	未知	未知	未知	未知	未知	未知
端点2缓冲区由位于地址0x60至0x67的8个字节组成.								

这三个数据缓冲区用于保存IN和OUT事务的数据.每个数据缓冲区长度为8个字节.

端点数据寄存器的复位值是未知的.

与过去的enCoRe零件不同，USB数据缓冲区只能在处理器的IO空间中访问.

22. USB模式表

模式	编码	建立	在	OUT	注释
DISABLE	0000	忽视	忽视	忽视	忽略到此端点的所有USB流量.由Data和控制端点.
NAK IN / OUT	0001	接受	NAK	NAK	NAK IN和OUT令牌.仅限控制端点.
状态输出	0010	接受	失速	检查	STALL IN和ACK零字节OUT.控制端点只要.
STALL IN / OUT	0011	接受	失速	失速	STALL IN和OUT令牌.仅限控制端点.
仅限状态	0110	接受	TX0字节	失速	STALL OUT并发送IN令牌的零字节数据. CON-仅限终端.
ACK OUT - STATUS 在	1011	接受	TX0字节	ACK	确认OUT令牌或发送IN令牌的零字节数据.仅限控制端点.
ACK进入状态 OUT	1111	接受	TX计数	检查	响应IN数据或状态OUT.控制端点on-LY.
NAK OUT	1000	忽视	忽视	NAK	发送NAK握手到OUT令牌.数据端点在线 -LY.
ACK OUT（STALL = 0）	1001	忽视	忽视	ACK	此模式由SIE更改为is-ACK握手到OUT的信号.数据端点只要.
ACK OUT（STALL = 1）	1001	忽视	忽视	失速	停止OUT传输.
NAK IN	1100	忽视	NAK	忽视	发送IN令牌的NAK握手.仅限数据端点.
ACK IN（STALL = 0）	1101	忽视	TX计数	忽视	之后，该模式由SIE更改为模式1100接收到IN数据的ACK握手.数据端点只要.
ACK IN（STALL = 1）	1101	忽视	失速	忽视	停止IN传输.仅限数据端点.
保留的	0101	忽视	忽视	忽视	SIE不支持这些模式.固件必须不在控制和数据端点中使用此模式.
保留的	0111	忽视	忽视	忽视	
保留的	1010	忽视	忽视	忽视	
保留的	0100	忽视	忽视	忽视	
保留的	1110	忽视	忽视	忽视	

22.1模式列

The 'Mode' column contains the mnemonic names given to the 终点的模式.终点的模式已确定 by the 4-bit binaries in the 'Encoding' column as discussed in the 以下部分.状态IN和状态OUT代表控制权转移的状态IN或OUT阶段.

22.2编码列

The contents of the 'Encoding' column represent the Mode 端点模式寄存器的位[3：0]（第60页上的表21-3）和第61页上的表21-4）.端点模式决定了方式SIE响应主机发送给不同的令牌端点.例如，如果端点0的模式位[3：0] Mode Register are set to '0001', which is NAK IN/OUT mode, the SIE发送ACK握手以响应SETUP令牌和NAK任何IN或OUT令牌.

22.3 SETUP，IN和OUT列

Depending on the mode specified in the 'Encoding' column, the 'SETUP', 'IN', and 'OUT' columns contain the SIE's responses when
端点分别接收SETUP，IN和OUT令牌.

A 'Check' in the Out column means that upon receiving an OUT token the SIE checks to see whether the OUT is of zero length and
具有1的数据切换（Data1 / 0）.如果这些条件成立，则SIE以ACK进行响应.如果任何这些条件不符合，
SIE回应STALL或忽略.

IN列中的“TX计数”条目表示SIE发送的字节数位[3：0]中指定的字节数
端点计数寄存器（表21-2）响应任何IN令牌.

23.不同交通状况的模式细节

控制端点																
SIE	公交赛事				SIE	EP0模式寄存器				EP0计数寄存器				EP0	打断	注释
模式	代币	计数	DVAL	D0 /	回应S		一世A模式			DTOGDVAL			COUNT	FIFO		
禁用																
0000	X	X	X	X												忽略所有
STALL_IN_OUT																
0011	建立	> 10	X	X										破烂		忽视
0011	建立	<= 10	无效	X										破烂		忽视
0011	建立	<= 10	有效	X	ACK	1			1	0001	更新	1	更新	数据	是	ACK设置
0011	在	X	X	X	失速											停在IN
0011	OUT	> 10	X	X												忽视
0011	OUT	<= 10	无效	X												忽视
0011	OUT	<= 10	有效	X	失速											停止OUT
NAK_IN_OUT																
0001	设置	> 10	X	X										破烂		忽视
0001	设置	<= 10	无效	X										破烂		忽视
0001	设置	<= 10	有效	X	ACK	1			1	0001	更新	1	更新	数据	是	ACK设置
0001	在	X	X	X	NAK											NAK IN
0001	OUT	> 10	X	X												忽视
0001	OUT	<= 10	无效	X												忽视
0001	OUT	<= 10	有效	X	NAK											NAK OUT
ACK_IN_STATUS_OUT																
1111	建立	> 10	X	X										破烂		忽视
1111	建立	<= 10	无效	X										破烂		忽视
1111	建立	<= 10	有效	X	ACK	1			1	0001	更新	1	更新	数据	是	ACK设置
1111	在	X	X	X	TX											主机未确认
1111	在	X	X	X	TX		1		1	0001					是	主持人确认
1111	OUT	> 10	X	X												忽视
1111	OUT	<= 10	无效	X												忽视
1111	OUT	<= 10，<> 2	有效	X	失速					0011					是	状态不佳
1111	OUT	2	有效	0	失速					0011					是	状态不佳
1111	OUT	2	有效	1	ACK			1	1	0010	1	1	2		是	良好状态
STATUS_OUT																
0010	设置	> 10	X	X										破烂		忽视
0010	设置	<= 10	无效	X										破烂		忽视
0010	设置	<= 10	有效	X	ACK	1			1	0001	更新	1	更新	数据	是	ACK设置
0010	在	X	X	X	失速					0011					是	停在IN
0010	OUT	> 10	X	X												忽视
0010	OUT	<= 10	无效	X												忽视

23.不同交通情况的模式详情（续）

控制端点																
SIE	公交赛事				SIE	EP0模式寄存器					EP0计数寄存器			EP0	打断	注释
模式	代币	计数	DVAL	D0 /	回应S					一世A模式	DTOGDVAL		COUNT	FIFO		
0010	OUT	<= 10，<> 2	有效	X	失速					0011					是	状态不佳
0010	OUT	2	有效	0	失速					0011					是	状态不佳
0010	OUT	2	有效	1	ACK			1	1		1	1	2		是	良好的状态
ACK_OUT_STATUS_IN																
1011	建立	>10	X	X										破烂		忽视
1011	建立	<= 10	无效	X										破烂		忽视
1011	建立	<= 10	有效	X	ACK	1			1	0001	更新	1	更新	数据	是	ACK设置
1011	在	X	X	X	TX 0											主机未确认
1011	在	X	X	X	TX 0		1		1	0011					是	主持人确认
1011	OUT	>10	X	X										破烂		忽视
1011	OUT	<= 10	无效	X										破烂		忽视
1011	OUT	<= 10	有效	X	ACK			1	1	0001	更新	1	更新	数据	是	好OUT
STATUS_IN																
0110	建立	>10	X	X										破烂		忽视
0110	建立	<= 10	无效	X										破烂		忽视
0110	建立	<= 10	有效	X	ACK	1			1	0001	更新	1	更新	数据	是	ACK设置
0110	在	X	X	X	TX 0											主机未确认
0110	在	X	X	X	TX 0		1		1	0011					是	主持人确认
0110	OUT	> 10	X	X												忽视
0110	OUT	<= 10	无效	X												忽视
0110	OUT	<= 10	有效	X	失速					0011					是	停止OUT
数据输出端点																
ACK OUT（STALL位= 0）																
1001	在	X	X	X												忽视
1001	OUT	> MAX	X	X										破烂		忽视
1001	OUT	<= MAX	无效	无效										破烂		忽视
1001	OUT	<= MAX	有效	有效	ACK				1	1000	更新	1	更新	数据	是	ACK OUT
ACK OUT（STALL位= 1）																
1001	在	X	X	X												忽视
1001	OUT	> MAX	X	X												忽视
1001	OUT	<= MAX	无效	无效												忽视
1001	OUT	<= MAX	有效	有效	失速											停止OUT
NAK OUT																
1000	在	X	X	X												忽视
1000	OUT	> MAX	X	X												忽视
1000	OUT	<= MAX	无效	无效												忽视
1000	OUT	<= MAX	有效	有效	NAK										如果启用	NAK OUT
终端中的数据																
ACK IN（STALL位= 0）																
1101	OUT	X	X	X												忽视
1101	在	X	X	X												主机未确认
1101	在	X	X	X	TX				1	1100					是	主持人确认
ACK IN（STALL位= 1）																
1101	OUT	X	X	X												忽视

23.不同交通情况的模式详情（续）

控制端点																
SIE	公交赛事				SIE	EP0模式寄存器					EP0计数寄存器			EP0	打断	注释
模式	代币	计数	DVAL	D0/1	回应S						DTOGDVAL		COUNT	FIFO		
1101	在	X	X	X	失速											停在IN
NAK IN																
1100	OUT	X	X	X												忽视
1100	在	X	X	X	NAK										如果启用	NAK IN

24.注册摘要

CPU标志寄存器中的XIO位必须设置为访问0xFF以上所有寄存器的扩展寄存器空间.

地址	名称	7	6	五	4	3	2	1	0	R/W	默认
00	P0DATA	P0.7	P0.6 / TIO1	P0.5 / TIO0	P0P4/3IN	INTP0.2 / IN	P0.1 / CLKOUT	P0.0 / CLK		BBBBBBBB	00000000
01	P1DATA	P1.7	P1.6 / SPI所以	P1.5 / SPISI	P1.4 / SCLK	P1.3 / SSEL	P1.2	P1VREG	P1.0 / d +	BBBBBBBB	00000000
02	P2DATA	RES						P2.1-P2.0		BBBBBBBB	00000000
03	P3DATA	RES						P3.1-P3.0		BBBBBBBB	00000000
04	P4DATA	RES				RES				---- BBBB	00000000
05	P00CR	保留的	诠释启用	完整低	TTL门限	保留	打开排水管	拉起启用	产里启用	-bbbbbbb	00000000
06	P01CR	CLK产里	诠释启用	完整低	TTL门限	保留	打开排水管	拉起启用	产里启用	BBBBBBBB	00000000
07-09	P02CR-P04CR	保留保留		完整低	TTL门限	保留	打开排水管	拉起启用	产里启用	--bbbbbb	00000000
0A-0B	P05CR-P06CR	TIO产里	诠释启用	完整低	TTL门限	保留	打开排水管	拉起启用	产里启用	BBBBBBBB	00000000
0℃	P07CR	保留的	诠释启用	完整低	TTL门限	保留	打开排水管	拉起启用	产里启用	-bbbbbbb	00000000
0D	P10CR	保留的	诠释启用	完整低	保留的			PS / 2拉起启用	产里启用	-BB --- BB	00000000
0E	P11CR	保留的	诠释启用	完整低	保留的		打开排水管	保留的	产里启用	-BB - BB	00000000
0F	P12CR	CLK产里	诠释启用	完整低	TTL门限	保留	打开排水管	拉起启用	产里启用	BBBBBBBB	00000000
10	P13CR	保留的	诠释启用	完整低	3.3 V驱动	高接收器	打开排水管	拉起启用	产里启用	-bbbbbbb	00000000
11-13	P14CR-P16CR	SPI使用	诠释启用	完整低	3.3 V驱动	高接收器	打开排水管	拉起启用	产里启用	BBBBBBBB	00000000
14	P17CR	保留的	诠释启用	完整低	TTL门槛	高水槽	打开排水管	拉起启用	产里启用	-bbbbbbb	00000000
15	P2CR	保留的	诠释启用	完整低	TTL门限	保留	打开排水管	拉起启用	产里启用	-bbbbbbb	00000000
16	P3CR	保留的	诠释启用	完整低	TTL门限	保留	打开排水管	拉起启用	产里启用	-bbbbbbb	00000000
20	FRTMRL	自由运行计时器[7：0]								BBBBBBBB	00000000
21	FRTMRH	自由运行计时器[15：8]								BBBBBBBB	00000000
22	TCAP0R	捕获0上升[7：0]								BBBBBBBB	00000000
23	TCAP1R	捕获1上升[7：0]								BBBBBBBB	00000000
24	TCAP0F	捕捉0下降[7：0]								BBBBBBBB	00000000
25	TCAP1F	捕获1下降[7：0]								BBBBBBBB	00000000
26	PITMRL	程序间隔定时器[7：0]								BBBBBBBB	00000000
27	PITMRH	保留的				程序间隔定时器[11：8]				---- BBBB	00000000
28	PIRL	程序间隔[7：0]								BBBBBBBB	00000000
29	PIRH	保留的				Prog Interval [11：8]				---- BBBB	00000000

24.注册摘要 （续）

CPU标志寄存器中的XIO位必须设置为访问0xFF以上所有寄存器的扩展寄存器空间.

地址	名称	7	6	五	4	3	2	1	0	R/W	默认
2A	TMRCR	第一边缘保持	8位捕捉预分频			Cap0 16bit启用	保留的			BBBBB --	00000000
2B	TCAPINTE	保留的				Cap1秋季活性	Cap1上升活性	Cap0秋季活性	Cap0上升活性	---- BBBB	00000000
2C	TCAPINTS	保留的				Cap1秋季活性	Cap1上升活性	Cap0秋季活性	Cap0上升活性	---- BBBB	00000000
三十	CPUCLKCR	保留	USB CLK/2禁用	USB CLK选择	保留的				中央处理器CLK选择	-BB ---- b	00010000
31	ITMRCLKCR	TCAPCLK分频器		TCAPCLK选择		ITMRCLK分频器		ITMRCLK选择		BBBBBBBB	10001111
32	CLKIOCR	保留的			保留的			CLKOUT选择		--- BBBBB	00000000
34	IOSCTR	foffset [2：0]			增益[4：0]					BBBBBBBB	000ddddd
36	LPOSCTR	32千赫低功率	保留32 kHz偏置调整[1：0]			32 kHz频率调整[3：0]				B-BBBBBB	DDDDDDDD
39	OSCLCKCR	保留的						微调只要	USB Osclock禁用	------ BB	00000000
3C	SPIDATA	SPIDATA [7：0]								BBBBBBBB	00000000
3D	SPICR	交换	LSB第一	通信模式		CPOL	CPHA	SCLK选择		BBBBBBBB	00000000
40	USBCR	USB启用	设备地址[6：0]							BBBBBBBB	00000000
41	EP0CNT	数据切换	数据有效	保留的		字节计数[3：0]				BBBBBBBB	00000000
42	EP1CNT	数据切换	数据有效	保留的		字节计数[3：0]				BBBBBBBB	00000000
43	EP2CNT	数据切换	数据有效	保留的		字节计数[3：0]				BBBBBBBB	00000000
44	EP0MODE	建立rcvd	IN rcv'd	OUT rcv'd	ACK'd trans	模式[3：0]				ccccbbbb	00000000
45	EP1MODE	摊子	保留的	NAK Int启用	ack'd trans	模式[3：0]				B-bcbbbb	00000000
46	EP2MODE	摊子	保留的	NAK Int启用	ack'd trans	模式[3：0]				B-bcbbbb	00000000
50-57	EP0DATA	端点0数据缓冲区[7：0]								BBBBBBBB	????????
58-5F	EP1DATA	端点1数据缓冲区[7：0]								BBBBBBBB	????????
60-67	EP2DATA	端点2数据缓冲区[7：0]								BBBBBBBB	????????
73	VREGCR	保留的						活着	VREG启用	------ BB	00000000
74	USBXCR	USB拉起启用	保留的						USB Force州	b ------ b	00000000
DA	INT_CLR0	GPIO端口1	睡觉计时器	INT1	GPIO端口0	SPI接收	SPI传输	INT0	POR / LVD	BBBBBBBB	00000000
DB	INT_CLR1	TCAP0	PROG间隔计时器	1毫秒计时器	USB主动	USB重置	USB EP2	USB EP1	USB EP0	BBBBBBBB	00000000
DC	INT_CLR2	保留保留	GPIO端口3		GPIO端口2	PS / 2数据低	INT2	16位计数器包	TCAP1	-bbbbbbb	00000000
DE	INT_MSKEN	SWINT	保留的							b -------	00000000
DF	INT_MSK	保留保留	GPIO端口3 Int启用		GPIO端口2 Int启用	PS / 2数据低 Int启用	INT2 Int启用	16位计数器包 Int启用	TCAP1 Int启用	-bbbbbbb	00000000
E1	INT_MSK	TCAP0 Int启用	PROG间隔计时器 Int启用	1毫秒计时器 Int启用	USB活动 Int启用	USB重置 Int启用	USB EP2 Int启用	USB EP1 Int启用	USB EP0 Int启用	BBBBBBBB	00000000

24.注册摘要 （续）

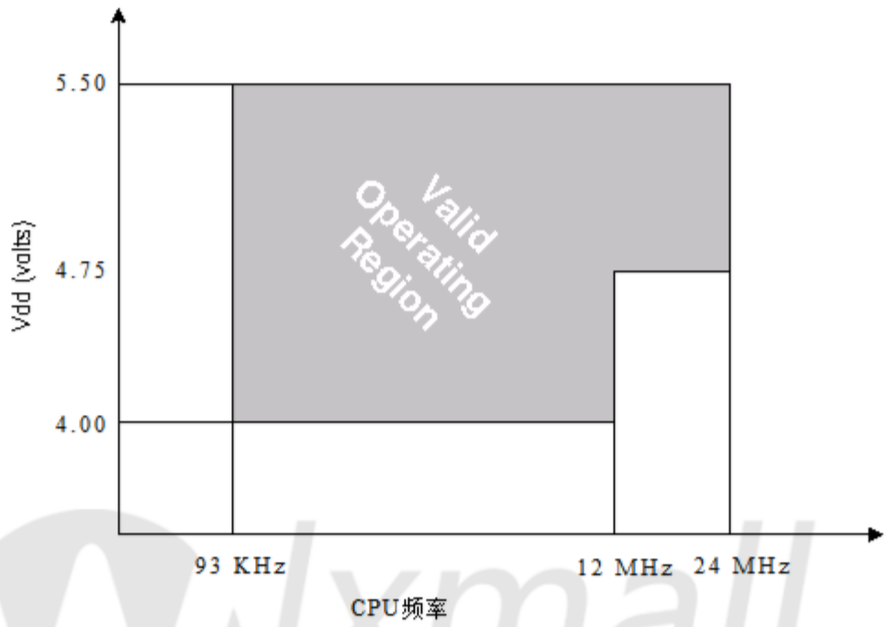
CPU标志寄存器中的XIO位必须设置为访问0xFF以上所有寄存器的扩展寄存器空间.

地址	名称	7	6	五	4	3	2	1	0	R/W	默认
E0	INT_MSK0	GPIO端口1 Int启用	睡觉计时器 Int启用	INT1 Int启用	GPIO端口0 Int启用	SPI接收 Int启用	SPI传输 Int启用	INT0 Int启用	POR / LVD Int启用	BBBBBBBB	00000000
E2	INT_VC	待处理中断[7：0]								BBBBBBBB	00000000
E3	RESWDT	复位看门狗定时器[7：0]								WWWWWWWW	00000000
-	CPU_A	临时寄存器T1 [7：0]								--------	00000000
-	CPU_X	X [7：0]								--------	00000000
-	CPU_PCL	程序计数器[7：0]								--------	00000000
-	CPU_PCH	程序计数器[15：8]								--------	00000000
-	CPU_SP	堆栈指针[7：0]								--------	00000000
-	CPU_F	保留的			XOI	超	携带	零	全球IE	--- brwww	00000010
FF	CPU_SCR	GIES	保留的	WDRS	PORS	睡觉	保留的	保留的	停止	R-CCB - B	00010000
1E0	OSC_CR0	保留的		没有嗡嗡声睡眠定时器[1：0]			CPU速度[2：0]			--bbbbbb	00000000
1E3	LVDCR	保留的		PORLEV [1：		0保留的	VM [2：0]			--bb-BBBB	00000000
1EB	ECO_TR	睡眠占空比[1：0]		保留的						BB ------	00000000
1E4	VLTCMP	保留的						LVD	PPOR	------ RR	00000000

传说
在R / W列中，
b =读取和写入
r =占领
w =占写
c =读取/清除
? =未知
d =校准值.在正常使用期间不得更改.

25.电压与CPU频率特性

图25-1.电压与CPU频率特性



以24 MHz运行CPU需要最小的电压4.75 V.这适用于12 MHz以上的任何CPU速度，因此使用12-24MHz之间的外部时钟也必须遵守这一点需求.供电时以24MHz运行CPU电压低于4.75 V可能会导致不良行为，必须应避免.

许多enCoRe II应用程序使用USB Vbus 5 V作为电源设备的来源.根据USB规范，Vbus上的电压可能低于4.75 V（如果USB端口为低电平）电源端口的电压可以在4.4 V和5.25 V之间）.甚至对于外部供电的5 V应用，开发人员必须考虑在通电和断电时电压小于4.75V一段时间.固件必须正确执行防止不需要的行为.

使用24 MHz需要使用高POR跳变点大约4.55-4.65 V（寄存器LVDCR 0x1E3，PORLEV [1：0] = 10b）.这个设置足以保护设备免受因CPU在低电压下工作而出现的问题速度高于12 MHz.这必须在设置CPU之前设置速度大于12 MHz.对于功率较慢的设备可能会导致将POR阈值更改为高电平在设备的一次或多次重置中，随着功率的增加，芯片的默认POR设定点约为2.6 V高POR设定点.

如果多次复位对于低功率斜坡是不合需要的，那么固件必须执行以下操作：

- 设置低电压检测电路（寄存器LVDCR 0x1E3，VM [2：0]）为POR（VM [2：0] =）以上的其中一个设定点110b～4.73V或111b～4.82V）.
- 监视LVD，直到电压高于跳闸点（寄存器VLTCMP 0x1E4，位1清零）.
- 去除指示以确保电压高于设定值指出可能的嘈杂用品.
- 将POR设置为高设定点.
- 将CPU速度移至24 MHz.

如果电源电压低于4.75 V并且应用可以允许以12 MHz的CPU速度运行，然后应用固件也可以执行以下操作来最小化由于电压瞬变而导致复位事件的机会：

- 将LVD设置为所需的高电平设置（～4.73 V或～4.82V）.
- 启用LVD中断.
- 在LVD ISR中，将CPU速度降至12 MHz，然后移动POR降低到一个较低的阈值.
- 固件可以监视VLTCMP在正常情况下清除应用程序主循环.
- 去除指示以确保电压高于设定值点.
- 将POR切换到高设定点.
- 将CPU切换到24 MHz.

注意
6.在主模式下，SCLK引脚上的主时钟边沿之前，第一位可用0.5 SPICLK周期.

26.绝对最大额定值

超过最大额定值可能会缩短使用寿命
设备.用户指南未经测试.

存储温度..... -40°C至+90°C

施加功率时的环境温度... -0°C至+70°C

V CC 上的电源电压 相对于V SS .....- 0.5 V至+7.0 V

直流输入电压..... -0.5 V至+ V CC + 0.5 V

用于输出的直流电压
高阻态..... -0.5 V至+ V CC + 0.5 V

最大总吸收电流到端口0
和端口1引脚..... ..... 70 mA

进入GPIO引脚的最大总输出电流为30 mA

最大的片上功耗
在任何GPIO引脚..... ..... 50 mW

功耗 .... 300 mW

静电放电电压..... 2200 V

闩锁电流..... ..... 200毫安

27.直流特性

参数	描述	条件	敏	典型	马克斯	单元
	一般					
V CC1	工作电压	没有USB活动，CPU速度<12_MHz	4	-	5.5	V
V CC2	工作电压	USB活动，CPU速度<12 MHz	4.35	-	5.25	V
V CC3	工作电压	Flash编程	4	-	5.5	V
V CC4	工作电压	没有USB活动，CPU速度是 在12 MHz和24 MHz之间	4.75	-	5.5	V
T FP	操作温度	Flash编程	0	-	70	C
我 CC1	V CC 工作电源电流	V CC = 5.25 V，无GPIO负载， 24 MHz	-		40	嘛
我 CC2	V CC 工作电源电流	V CC = 5.0 V，无GPIO负载，6 MHz	-	10		嘛
我 SB1	待机电流	内部和外部振荡器， 带隙，闪存，CPU时钟，定时器 时钟，USB时钟全部禁用	-		10	µA
低电压检测						
V LVD	低电压检测跳闸电压 （8个可编程的跳闸点）	-	2.681	-	4.872	V
3.3V稳压器						
我 VREG	最大调节器输出电流	4.35 V ≤V CC ≤5.5 V	-	-	125	嘛
我 KA	保持活力	当调节器被禁用时 “保持活力”启用	-		20	µA
V KA	保持电压	在VREGCR中 保留位设置	2.35	-	3.8	V
V REG1	V REG 输出电压	V CC ≥ 4.35V，0 <温度<40°C， 25 mA ≤I VREG ≤125 mA（3.3V±8%） T = 0至70°C	3.0	-	3.6	V
V REG2	V REG 输出电压	V CC ≥ 4.35V，0 <温度<40°C， 1 mA ≤I VREG ≤25 mA（3.3V±4%） T = 0至40°C	3.15	-	3.45	V
C LOAD	Vreg引脚上的容性负载	-	1	-	2	µF
L N REG	线路调节	-	-	-	1	%/V
L D REG	负载调节	-	-	-	0.04	%/嘛
USB接口						
V ON	静态输出高	15 K±5%欧姆至V SS	2.8	-	3.6	V
V OFF	静态输出低	R UP 已启用	-	-	0.3	V

笔记
7.仅适用于CY7C638xx P1.3，P1.4，P1.5，P1.6，P1.7.
8.除GPIO模式下的引脚P1.0和P1.1外.
9.除引脚P1.0和P1.1外.

27.直流特性（续）

参数	描述	条件	敏	典型	马克斯	单元
	一般					
V DI	差分输入灵敏度	-	0.2	-	-	V
V CM	差分输入共模范围	-	0.8	-	2.5	V
V SE	单端接收机门限	-	0.8	-	2	V
C IN	收发器电容	-	-	-	20	pF的
我 IO	Hi-Z状态数据线泄漏	0V <V IN <3.3V	-10	-	10	µA
PS / 2接口						
V OLP	静态输出低	SDATA或SCLK引脚	-	-	0.4	V
R PS2	内部PS / 2上拉电阻	SDATA，SCLK引脚，PS / 2已启用	3	-	7	kΩ
通用IO接口						
R UP	上拉电阻		4	-	12	kΩ
V ICR	输入阈值电压低，CMOS 模式[8]	低到高边缘	40 %	-	65 %	V CC
V ICF	输入阈值电压低，CMOS 模式[8]	高到低边缘	30 %	-	55 %	V CC
V HC	输入滞后电压，CMOS 模式[8]	高到低边缘	3%	-	10 %	V CC
V ILTTL	输入低电压，TTL模式[9]	I / O引脚电源= 4.0-5.5 V		-	0.8	V
V IHTTL	输入高电压，TTL模式[9]	I / O引脚电源= 4.0-5.5 V	2.0	-		V
V OL1	输出低电压，高驱动[7]	I OL1 = 50 mA	-	-	0.8	V
V OL2	输出低电压，高驱动[7]	我 OL1 = 25毫安	-	-	0.4	V
V OL3	输出低电压，低驱动[8]	我 OL2 = 8毫安	-	-	0.4	V
V OH	输出高电压[8]	我 OH = 2毫安	V CC -0.5	-	-	V
C LOAD	最大负载电容[9]	-		-	50	pF的

28. AC特性

参数	描述	条件	敏	典型	马克斯	单元
时钟						
T ECLKDC	外部时钟占空比	-	45	-	55	%
T ECLK1	外部时钟频率	外部时钟是的来源 CPUCLK	0.187	-	24	兆赫
T ECLK2	外部时钟频率	外部时钟不是的来源 CPUCLK	0-		24	兆赫
F IMO1	内部主振荡器频率	没有USB存在	22.8	-	25.2	兆赫
F IMO2	内部主振荡器频率	USB存在	23.64	-	24.3	兆赫
F ILO1	内部低功耗振荡器	正常模式	29.44	-	37.12	千赫
F ILO2	内部低功耗振荡器	低功耗模式	35.84	-	47.36	千赫
3.3 V调节器						
V ORIP	输出纹波电压	CLOAD = 1时为10 Hz至100 MHzµF	-	-	200	mV pp

注意
10.在主模式下，SCLK引脚上的主时钟边沿之前，第一位可用0.5 SPICLK周期.

28.交流特性（续）

参数	描述	条件	敏	典型	马克斯	单元
USB驱动程序						
T R1	转型上升时间	C LOAD = 200 pF	75	-	-	NS
T R2	转型上升时间	C LOAD = 600 pF		-	300	NS
T F1	转型下降时间	C LOAD = 200 pF	75	-		NS
T F2	转型下降时间	C LOAD = 600 pF		-	300	NS
T R	上升/下降时间匹配	-	80	-	125	%
V CRS	输出信号交叉电压	-	1.3	-	2.0	V
USB数据定时						
T DRATE	低速数据速率	平均比特率（1.5 Mbps±1.5%）	1.4775	-	1.5225	Mbps的
T DJR1	接收器数据抖动容限	到下一个转变	-75	-	75	NS
T DJR2	接收器数据抖动容限	配对转换	-45	-	45	NS
T DEOP	差分到EOP转换偏斜	-	-40	-	100	NS
T EOPR1	接收器的EOP宽度	拒绝为EOP		-	330	NS
T EOPR2	接收器的EOP宽度	接受为EOP	675	-		NS
T EOPT	来源EOP宽度	-	1.25	-	1.5	微秒
T UDJ1	差分驱动器抖动	到下一个转变	-95	-	95	NS
T UDJ2	差分驱动器抖动	配对转换	-95	-	95	NS
T LST	差异期间SE0的宽度.过渡	-	-	-	210	NS
非USB模式驱动器特性						
T FPS2	SDATA / SCK转换下降时间	-	50	-	300	NS
GPIO时序						
T R_GPIO	输出上升时间[8]	测量10到90% 具有50 pF负载的Vdd / Vreg	-		50	NS
T F_GPIO	输出下降时间[8]	测量10到90% 具有50 pF负载的Vdd / Vreg	-		15	NS
SPI时序						
T SMCK	SPI主时钟速率	F CPUCLK / 6	-	-	2	兆赫
T SSCK	SPI从属时钟速率	-	-	-	2.2	兆赫
T SCKH	SPI时钟高时间	CPOL = 0时为高电平，CPOL = 1时为低电平	-	-	-	NS
T SCKL	SPI时钟低电平时间	CPOL = 0时为低电平，CPOL = 1时为高电平	-	-	-	NS
T MDO	主数据输出时间[10]	SCK到数据有效	-25	-	50	NS
T MDO1	主数据输出时间， 第一位CPHA = 0	领先SCK边缘之前的时间	100	-	-	NS
T MSU	主输入数据建立时间	-	50	-	-	NS
T MHD	主输入数据保持时间	-	50	-	-	NS
T SSU	从机输入数据建立时间	-	50	-	-	NS
T SHD	从机输入数据保持时间	-	50	-	-	NS
T SDO	从站数据输出时间	SCK到数据有效		-	100	NS
T SDO1	从站数据输出时间， 第一位CPHA = 0	SS低电平到数据有效之后的时间	-	-	100	NS
T SSS	从机选择设置时间	在第一次SCK边缘之前	150	-	-	NS
T SSH	从机选择保持时间	在最后的SCK边沿之后	150	-	-	NS

1

图28-1.时钟计时

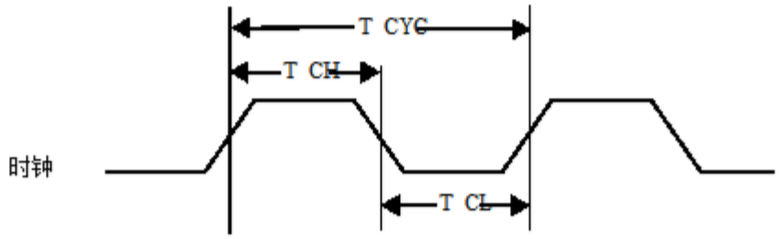


图28-2. GPIO时序图

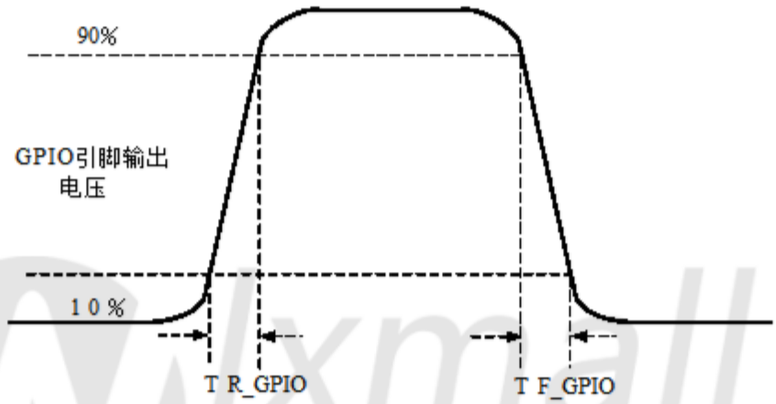


图28-3. USB数据信号定时

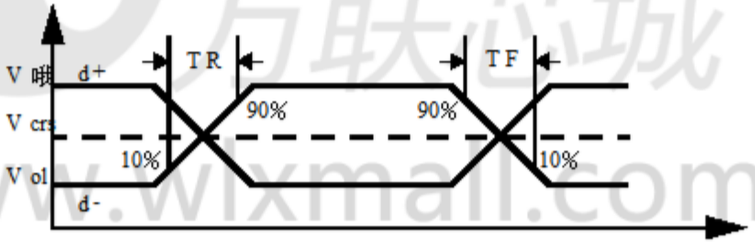


图28-4.接收器抖动容差

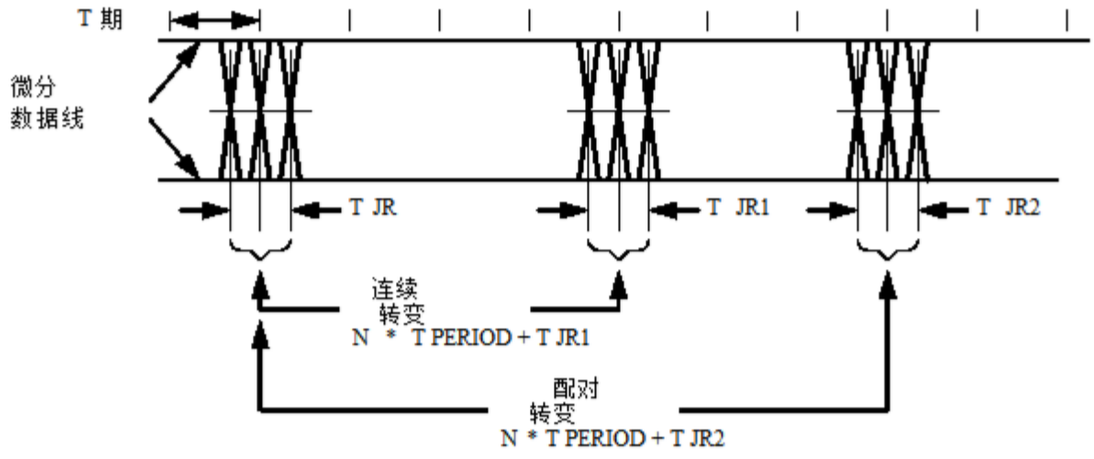


图28-5.差分到EOP转换偏移和EOP宽度

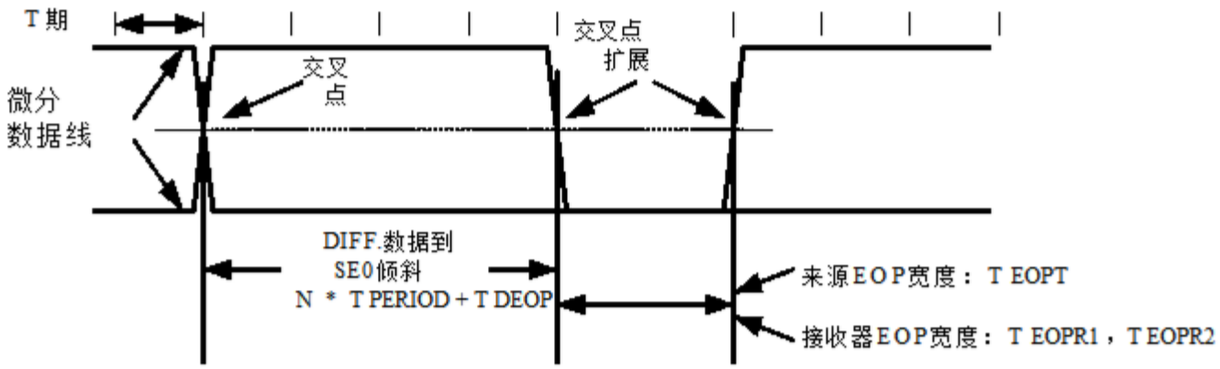


图28-6.差分数据抖动

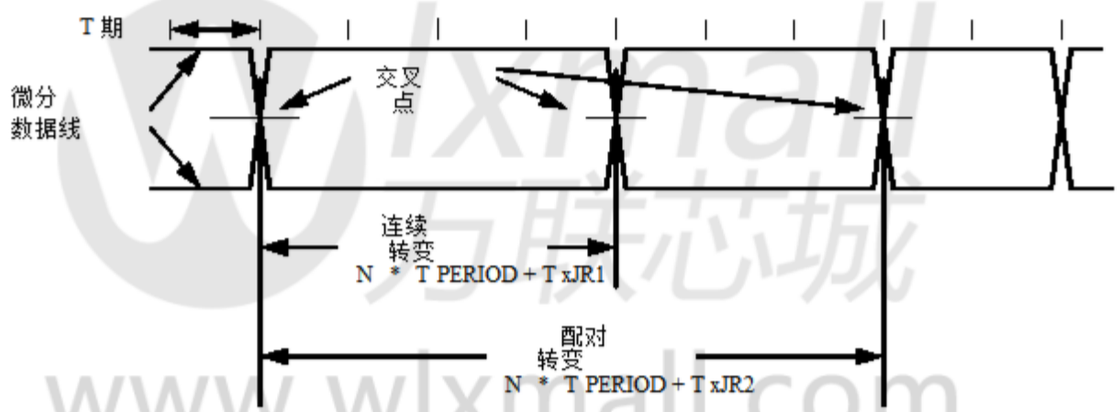


图28-7. SPI主时钟，CPHA = 1

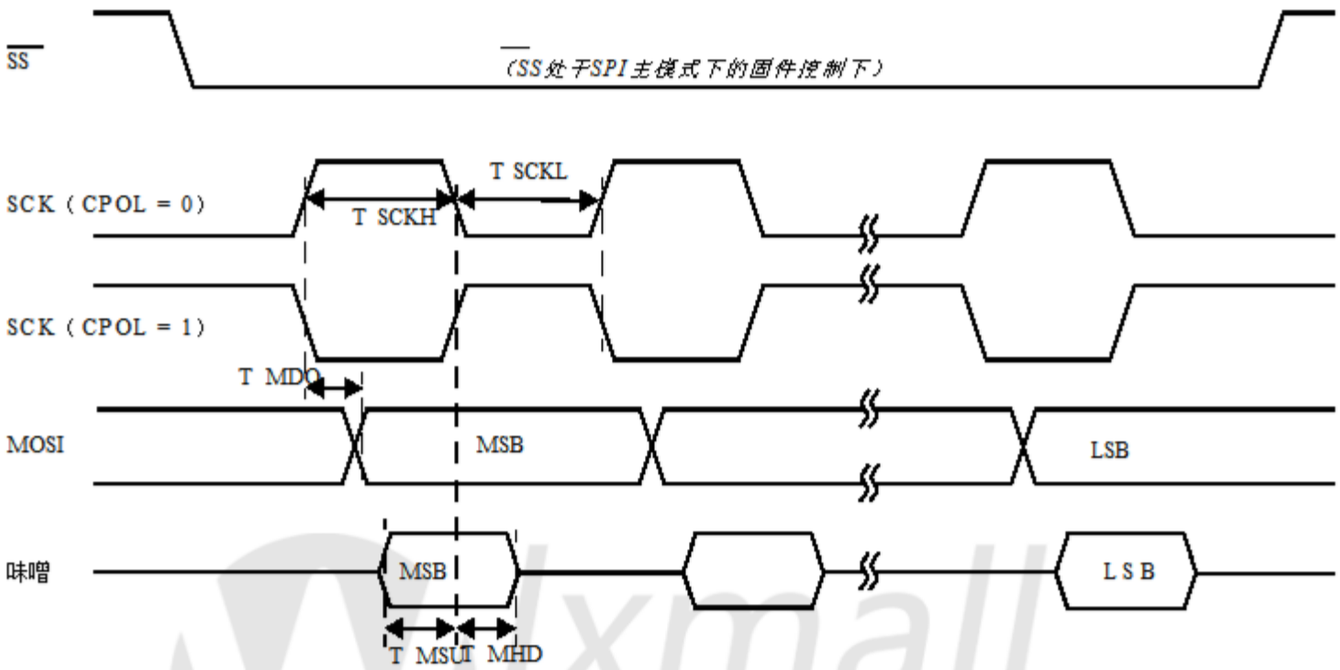


图28-8. SPI从时序，CPHA = 1

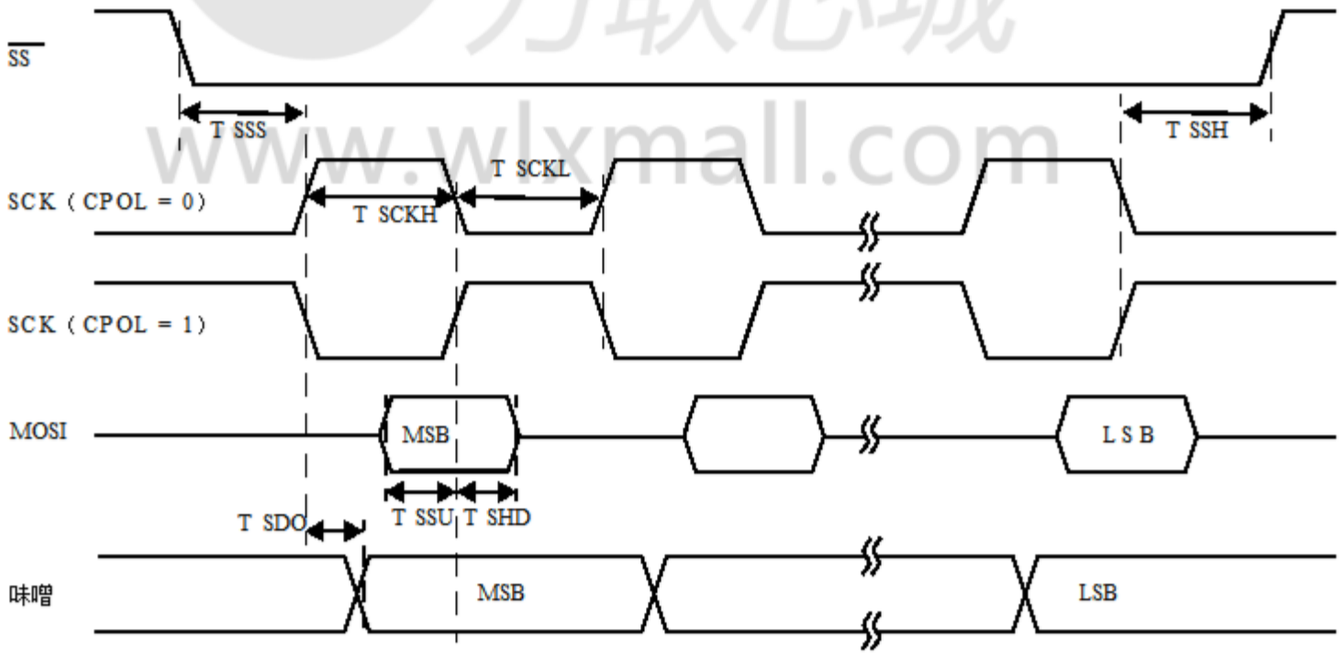


图28-9. SPI主时钟，CPHA = 0

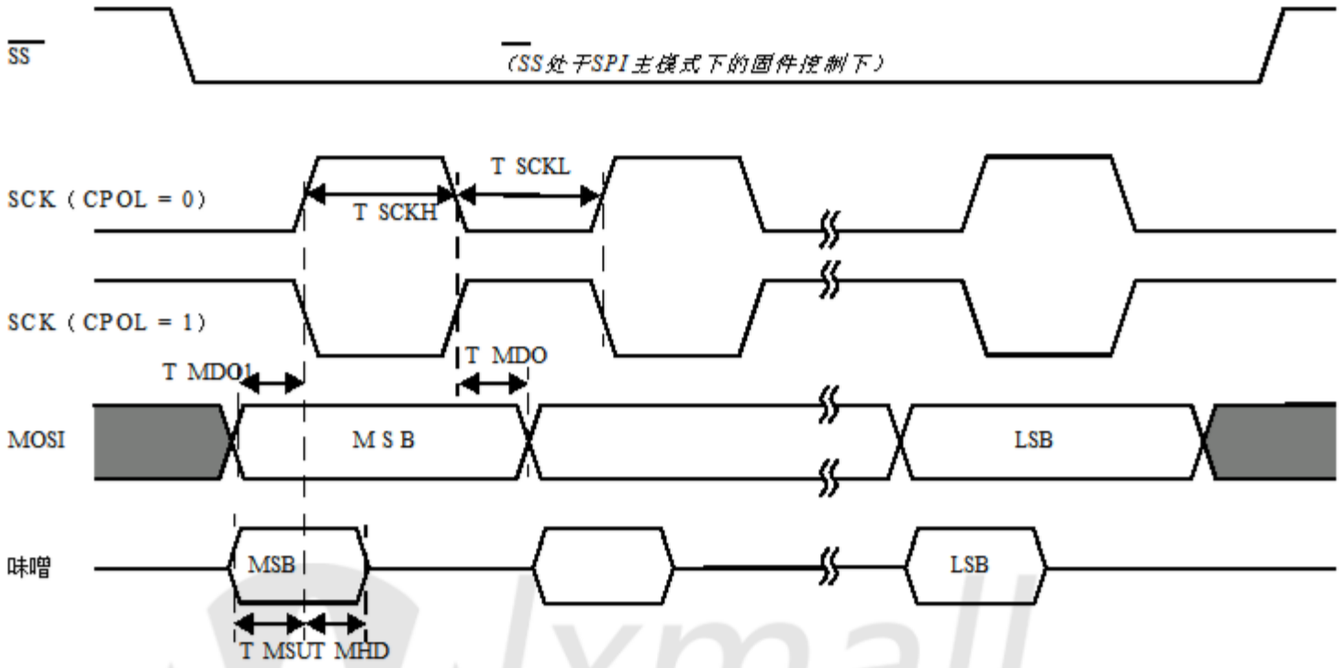
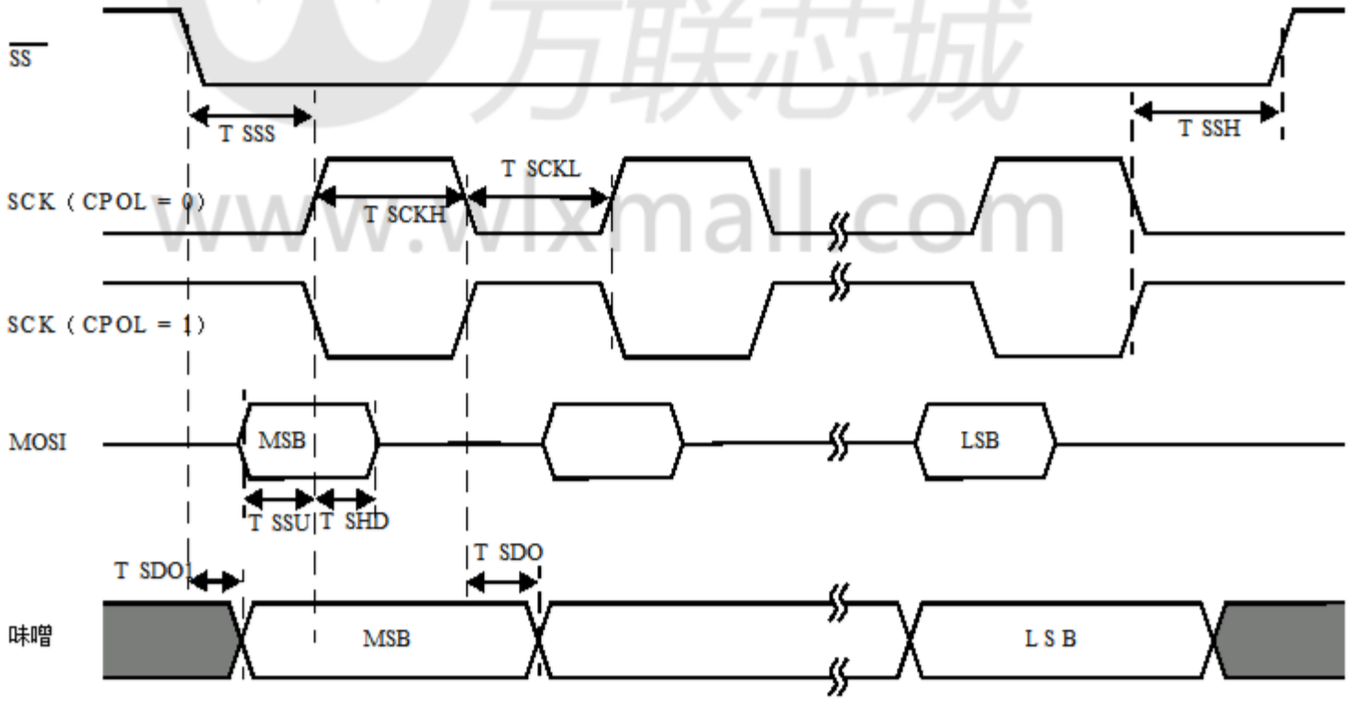


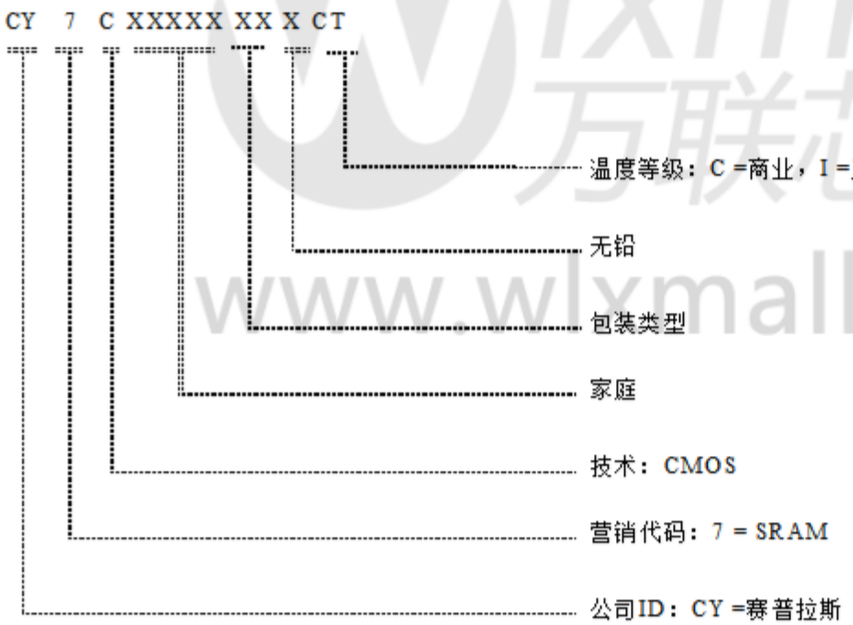
图28-10. SPI从时序，CPHA = 0



29.订购信息

订购代码	FLASH大小	RAM大小	包装类型
CY7C63310-SXC	3K	128	16-SOIC
CY7C63801-SXC	4K	256	16-SOIC
CY7C63803-SXC	8K	256	16-SOIC
CY7C63803-SXCT	8K	256	16-SOIC，卷带式
CY7C63813-PXC	8K	256	18引脚PDIP
CY7C63813-SXC	8K	256	18-SOIC
CY7C63823-QXC	8K	256	24 QSOP
CY7C63823-SXC	8K	256	24-SOIC
CY7C63823-SXCT	8K	256	24-SOIC，卷带式
CY7C63803-LQXC	8K	256	24-QFN锯
CY7C63823-XC	8K	256	模具形式
CY7C63823-3XW14C	8K	256	晶圆形式
CY7C63833-LTXC	8K	256	32-QFN锯
CY7C63833-LTXCT	8K	256	32-QFN锯，卷带和卷轴

29.1订购代码定义



30.包装处理

一些IC封装在焊接到PCB上之前需要烘烤以去除可能在离开后吸收的水分
工厂.包装上的标签上有关于实际烘烤温度和最小烘烤时间的详细信息以去除这种湿气.
最大烘烤时间是零件暴露于烘烤温度的总时间.超过这个曝光时间可能
降低器件的可靠性.

参数	描述	敏	典型	马克斯	单元
T BAKETEMP	烘烤温度	-	125	见包装标签	C
T烘烤	烘烤时间	见包装标签	-	72	小时

31.封装图

图31-1. 16引脚（300-Mil）模制DIP P1

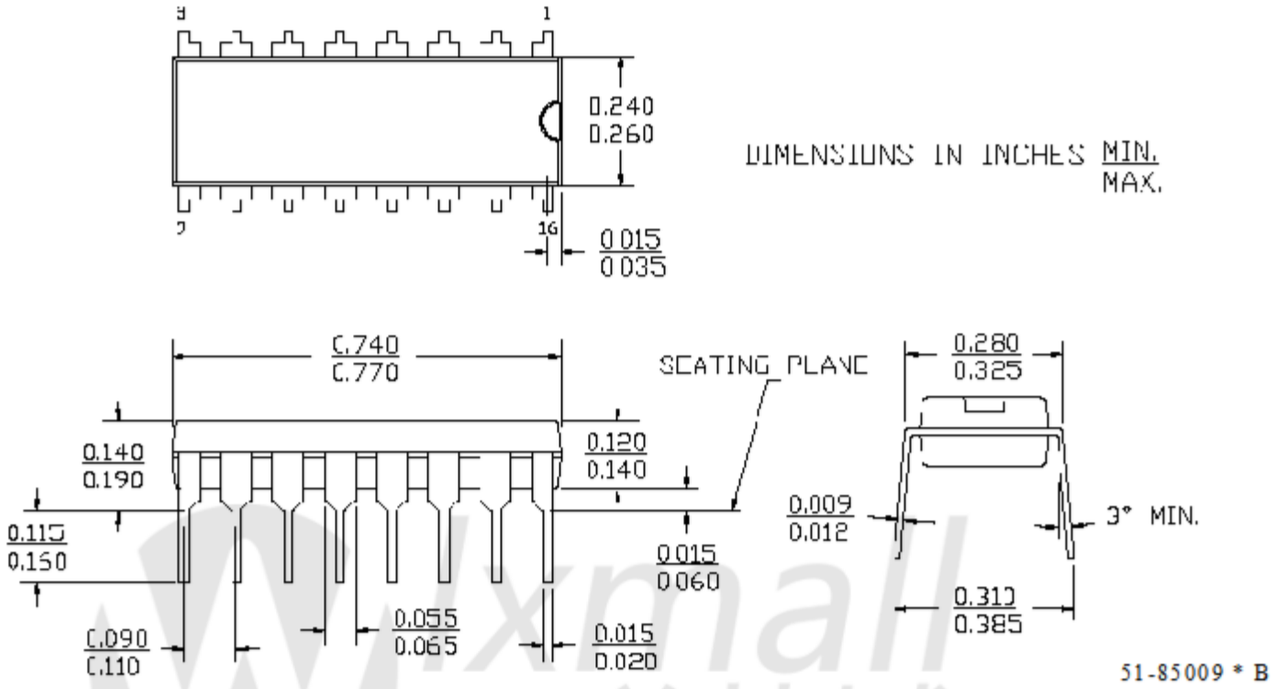


图31-2. 16引脚（150-Mil）SOIC S16.15

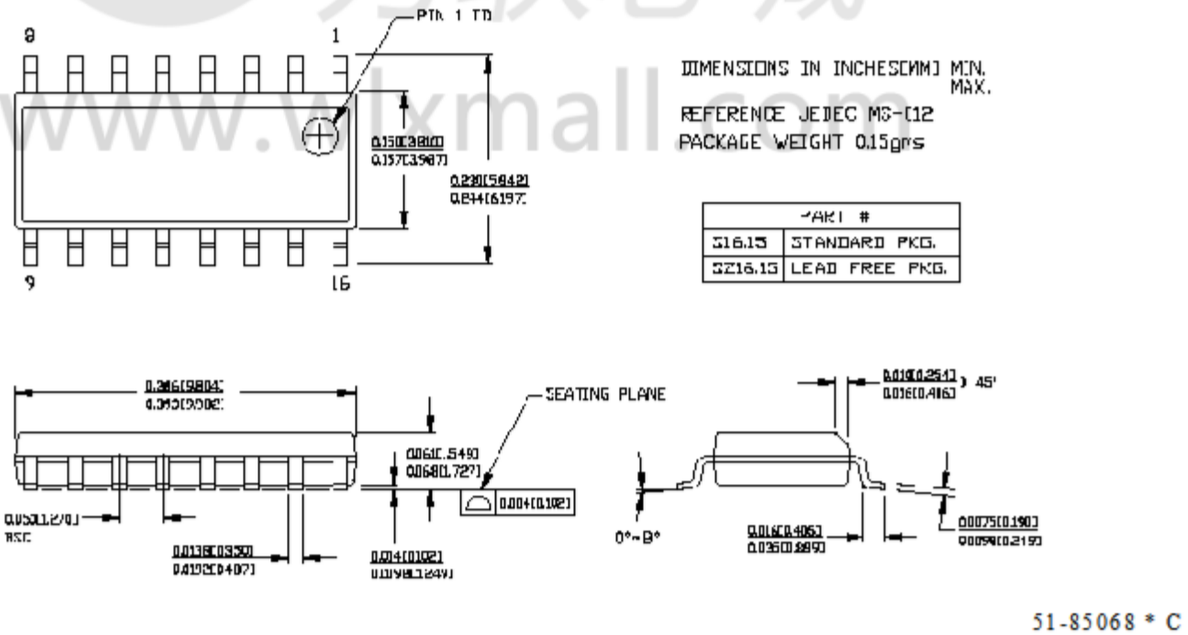


图31-3. 18引脚（300-Mil）模塑DIP P3

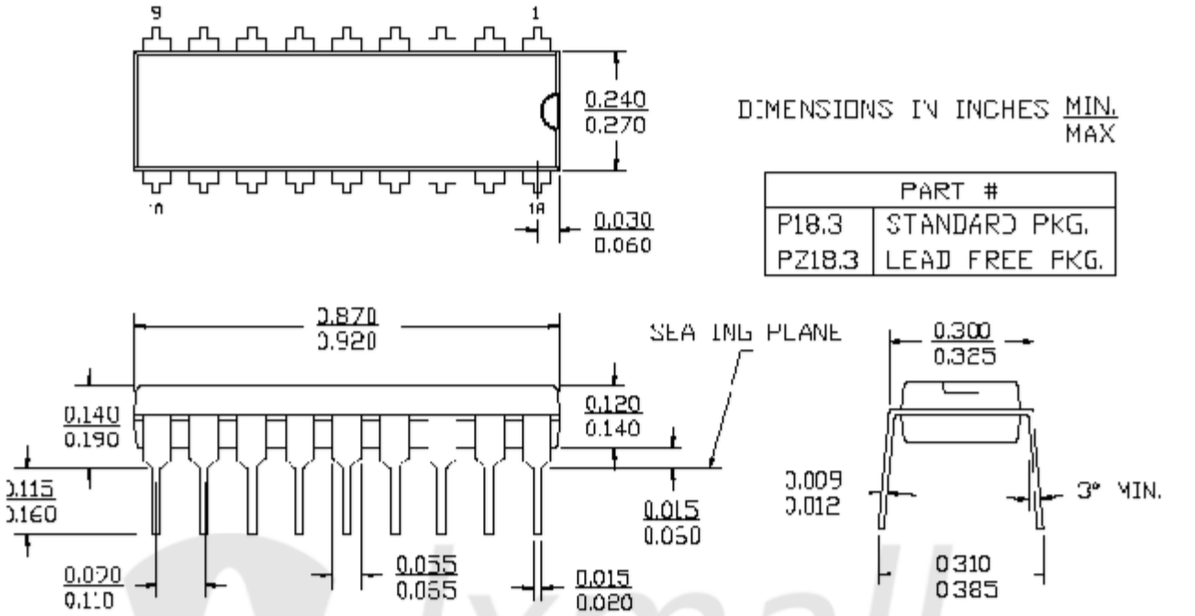


图31-4. 18引脚（300-Mil）模制SOIC S3

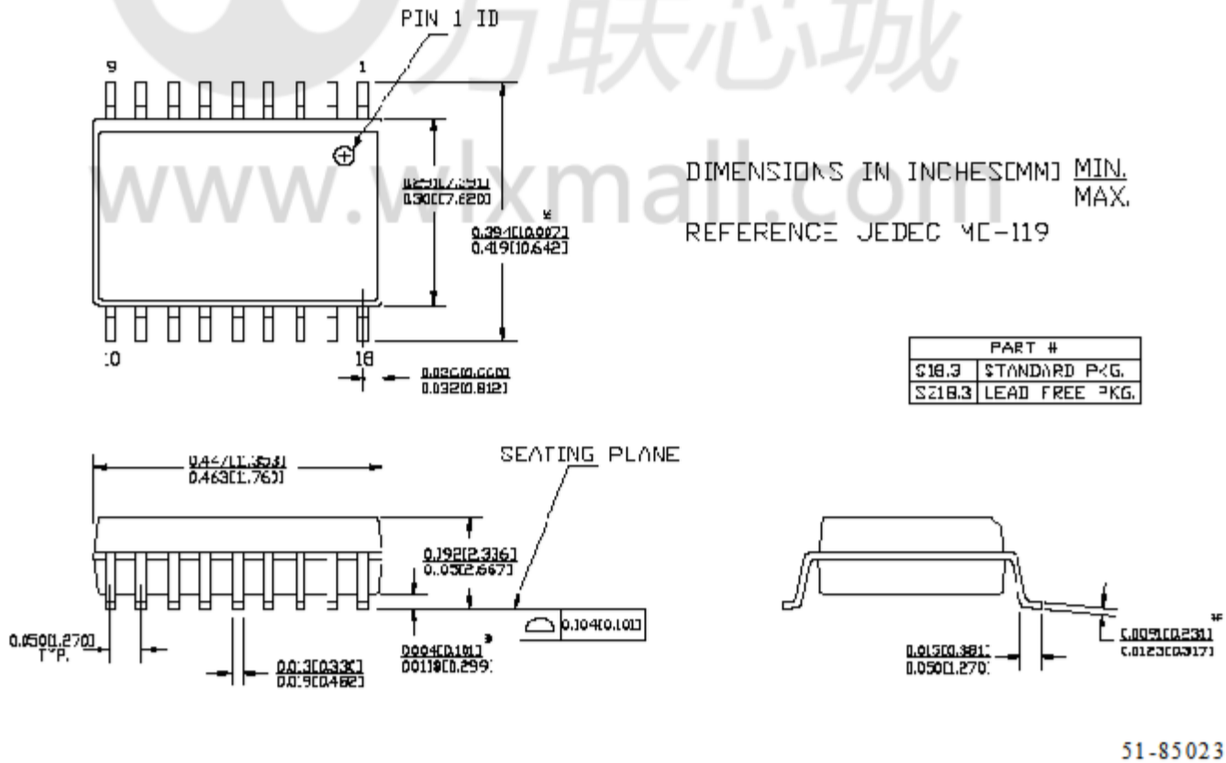


图31-5. 24引脚（300-Mil）SOIC S13

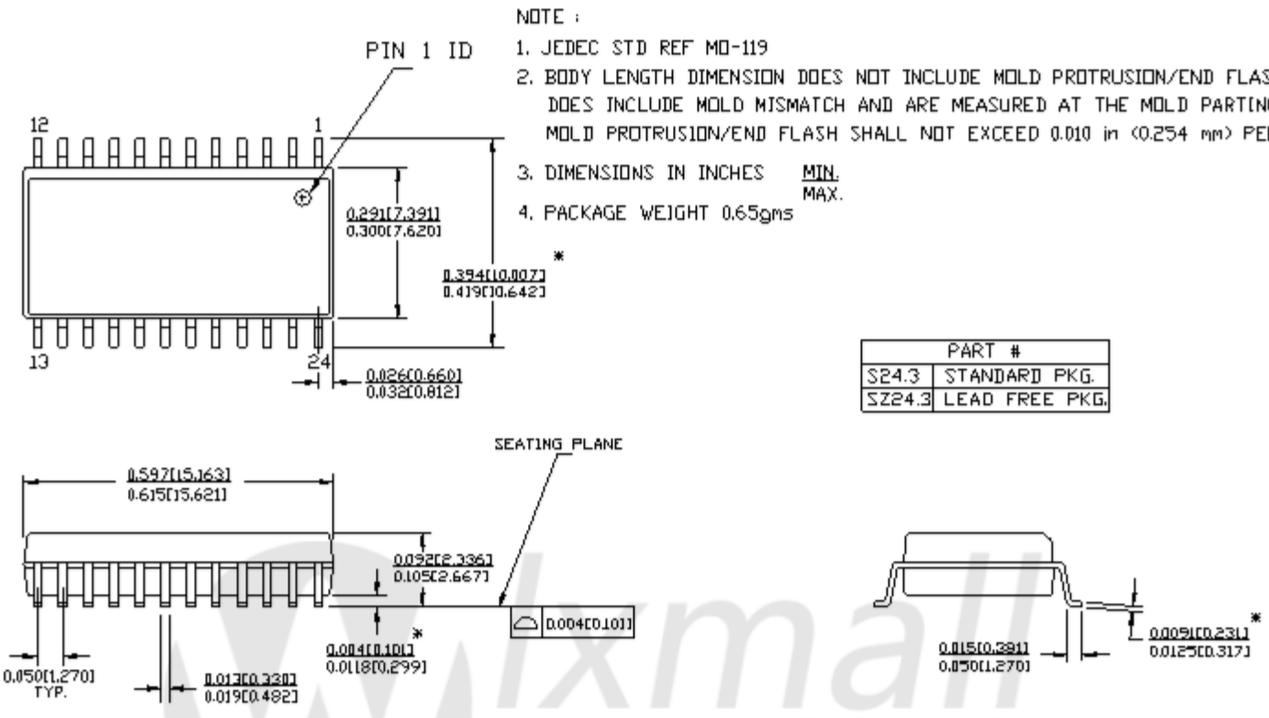


图31-6. 24引脚QSOP O241

小号

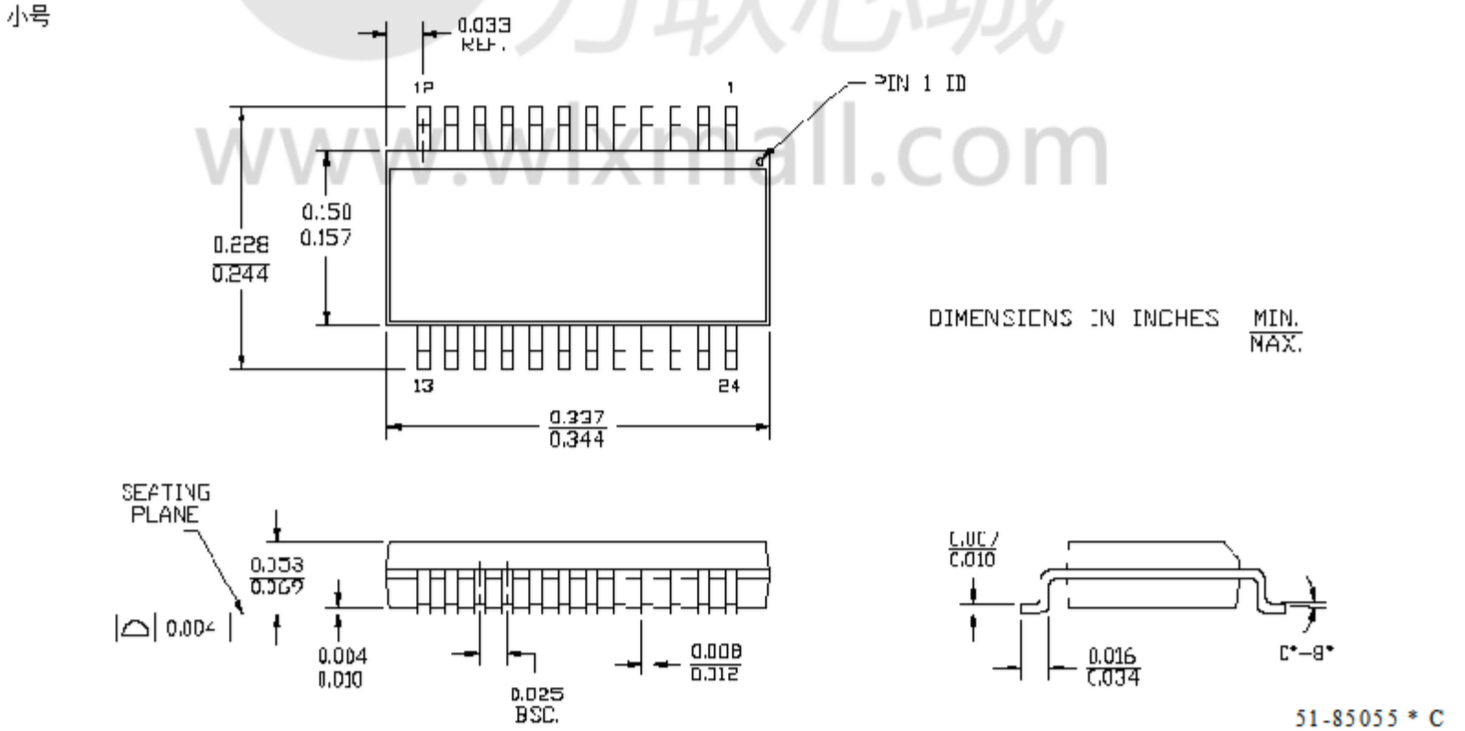
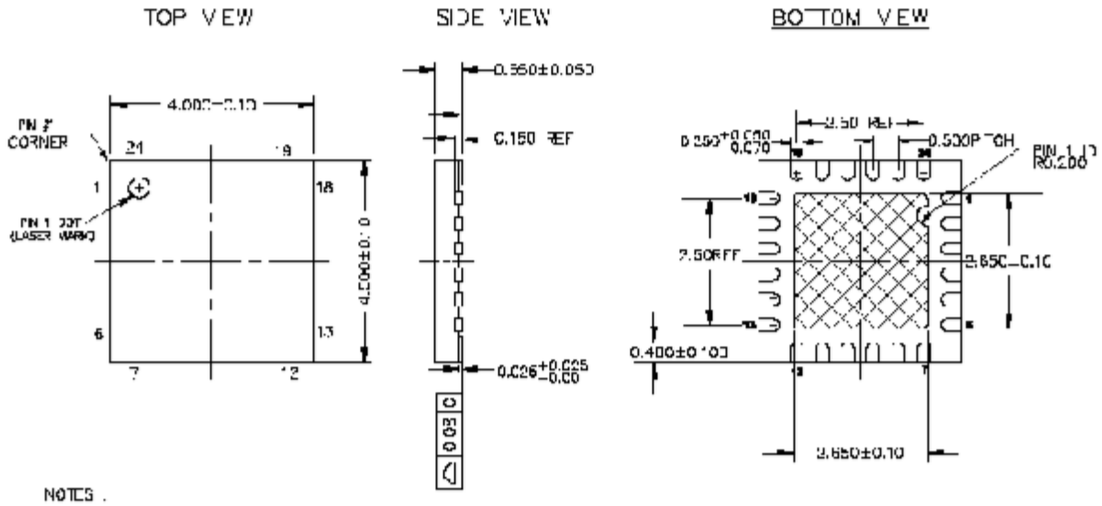


图31-7. 24引脚QFN 4X4X0.55毫米LQ24 A 2.65X2.65X EPAD（锯）

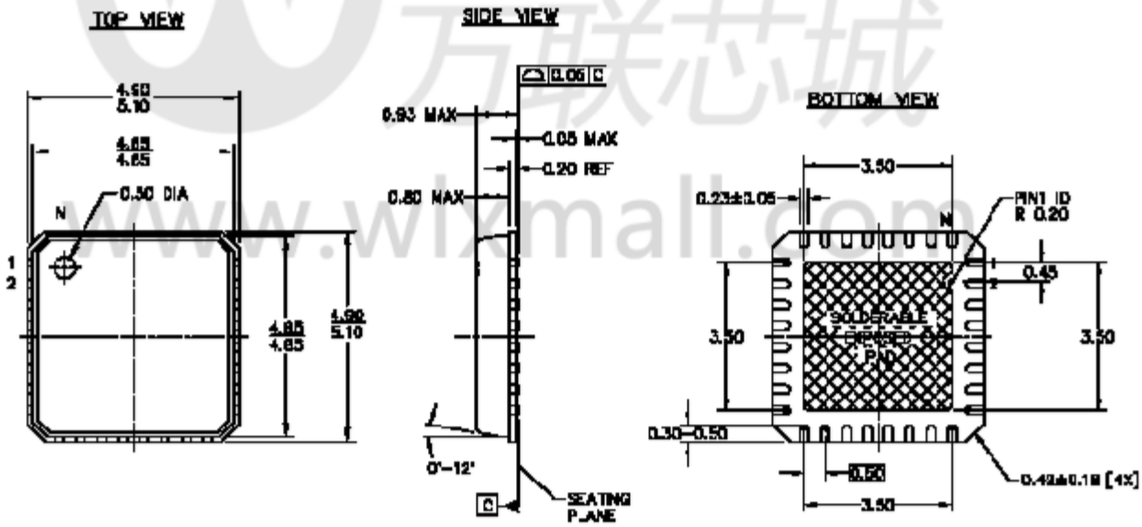


NOTES .

1. HATCH S SOLDERABLE EXPOSED METAL.
2. REFERENCE JEDEC # MO-248
3. ON PACKAGE WEIGHT 0.024 grams
4. ALL DIMENSIONS ARE IN MILLIMETERS

001-13937 * C

图31-8. 32引脚QFN封装



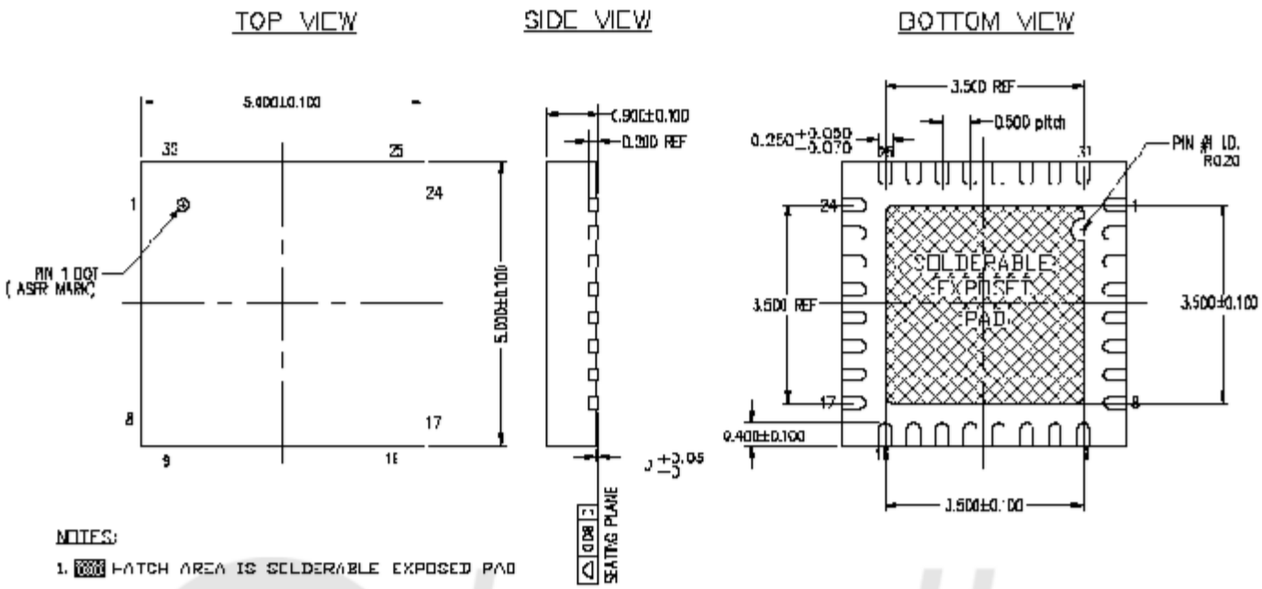
NOTES:

1. HATCH AREA IS SOLDERABLE EXPOSED PAD.
2. REFERENCE JEDEC#: MO-220
3. PACKAGE WEIGHT: 0.054g
4. ALL DIMENSIONS ARE IN MM [MIN/MAX]
5. PACKAGE CODE

PART #	DESCRIPTION
LF32	STANDARD
LY32	PB-FREE

51-85188 * D

图31-9. 32引脚的锯开QFN封装



32.缩略语

缩写	描述
CMOS	互补金属氧化物半导体
$\overline{CE}$	芯片启用
I / O	输入输出
$\overline{OE}$	输出使能
SRAM	静态随机存取存储器
TSOP	薄小外形包装
$\overline{我们}$	写使能

33.文件公约

计量单位

符号	测量单位
NS	纳秒
V v o l t s	
µA	微安培
嘛	毫安
pF的	皮克法拉
C	摄氏度
W w a t t s	

34.文档历史页面

文档标题：CY7C63310，CY7C638xx enCoRe™II低速USB外设控制器 文件号：38-08035				
调整	ECN号码	原稿.的更改	Submis-锡永日期	变更描述
**	131323	XGR	03年12月11日	新的数据表
*一个	221881	孔敬大学	请参阅ECN	新增寄存器说明和包装信息，从预先更改 信息初步
*B	271232	BON	请参阅ECN	重新格式化.用最新的信息更新
*C	299179	BON	请参阅ECN	第7页上的表5-2中更正了24-PDIP引脚排字错误添加了表10-1 第22页.更新了第17页上的表9-5，第23页上的表10-3，第13页上的表13-1 第32页的表17-1，第53页的表17-1，第53页的表17-3，第17页的表17-5 54.和第15页的表15-2.向GPIO部分添加了各种更新 （通用I / O（GPIO）端口）（第34页）更正了页面上的表15-4 43.修正了第74页上的图28-7和第74页上的图28-8.添加了 16引脚PDIP封装图（第77页的封装图部分）
* d	322053	TVR BON	请参阅ECN	第4页介绍：删除了最后一段中的低压复位.有 没有LVR，只有LVD（低电压检测）.解释更多关于LVD和POR. 将捕捉引脚从P0.0，P0.1更改为P0.5，P0.6. 第8页的表6-1：已更改的表标题（已删除助记符并已创建 作为注册名称）. 第17页的表9-5：包含不同闪存大小的# 行. 第22页的时钟体系结构说明：更改了CPUCLK的可选项 选项从n = 0-5,7,8到n = 0-5,7. 第20页的时钟：将ITMRCLK分区更改为1,2,3,4.更新了 来源ITMRCLK，TCAPCLKs.提到P17永久启用TTL. 更正FRT，PIT数据写入顺序.更新了INTCLR，INTMSK寄存器 登记表也. 直流特性（第69页）：将LVR更改为LVD，包括最大最小值 基于char数据的可编程跳闸点.更新了50ma sink接脚 638xx，63903.提到的保持电压对应于保持活跃状态 20uA的电流.包含有关VOL，VOH在P1.0，P1.1和TMDO上的注意事项 规范. 第70页的交流特性：T MDO1，T SDO1 更改了描述栏 阶段为0. 第5页的引脚分布：从CY7C63310和CY7C63801中删除了VREG 删除了SCLK和SDATA.创建一个单独的引脚图 CY7C63813. 添加了GPIO框图（第37页的图14-1） 第24页上的表10-4：将睡眠定时器时钟单元从32 kHz计数改为 到Hz. 第59页的表21-1：向寄存器添加了更多描述.
* E	341277	BHA	请参阅ECN	更正 直流特性中的 V IH TTL值（第69页）. 更新了V IL TTL值. 为D + / D-引脚添加脚注说明表. 向低电压检测表添加了典型值. 更正了16引脚PDIP封装的引脚标签. 更正了轻微的错别字.
*F	408017	T Y J	请参阅ECN	第5页的表5-2：更正了24引脚QSOP封装的引脚分配 - GPIO端口3 新分配：引脚19分配给P3.0，引脚20分配给P3.1 第55页的表17-6：INT_MASK1更改为0xE1 第56页的表17-7：INT_MASK0更改为0xE0 寄存器汇总（第65页）：寄存器汇总，地址E0分配给 INT_MASK0和分配给INT_MASK1的地址E1

34.文件历史记录页（续）

文档标题：CY7C63310，CY7C638xx enCoRe™II低速USB外设控制器 文件号：38-08035				
调整	ECN号码	原稿.的更改	Submis-锡永日期	变更描述
*G	424790	TYJ	请参阅ECN	小的文本更改使文档更具可读性 删除了CY7C639xx 从订购信息中删除CY7C639xx（第76页） 在第7页上的表5-2中增加了关于电流消耗的文字，用于P0.0和P0.1 更正了第16页上的图9-2以表示单个堆栈 在第7页的P1.3-P1.6 iTable 5-2上添加了有关3.3V IO可用性的评论 在Flash上 增加了有关闪存耐用性和数据保留的信息 第15页 增加了框图和时序图 添加了CY7C638xx裸片图，焊盘分配表和订购 信息 键盘参考删除 删除了CY7C63923-XC裸片图，删除了对639xx部件的引用 标题中更新了部件号
*H	491711	TYJ	请参阅ECN	小的文字更改 添加了32-QFN部分 删除了638xx晶粒图和晶粒垫分配 删除了GPIO端口4的配置细节 将P0.0和P0.1的GPIO特性分别修正为P1.0和P1.1
*一世	504691	TYJ	请参阅ECN	小的文字更改 去除了所有对外部晶体振荡器和GPIO4的残余参考 记录USB收发器的专用3.3V稳压器 记录带隙/电压调节器在唤醒时的行为 记录电压调节器线路/负载调节 记录USB活动和PS2数据低中断触发条件. 包括GPIO电容和时序图 讨论清除捕获中断状态的方法 睡眠和唤醒顺序记录. 讨论了EP1MODE / EP2MODE寄存器问题.

34.文件历史记录页（续）

文档标题：CY7C63310，CY7C638xx enCoRe™II低速USB外设控制器 文件号：38-08035				
调整	ECN号码	原稿.的更改	Submis-锡永日期	变更描述
*∫	2147747	VGT / AESA	二零零八年五月	在第1章上输入的TID号码.还更改了句子“高电流驱动器 在GPIO引脚上“变为”在所有GPIO引脚上的2mA电流源“. 第69页上的第26.0点直流特性更改了最小值.和最大.电压 Vcc3（第3行）分别为4.0和5.5. 点19.0，标题修改为“调节器输出”，而不是“USB调节器” 输出”. 在点17.3下添加了一个点＃3. 在点＃25.0中将存储温度改变为“-40℃至90℃”（绝对温度） 最大额定值在第69页）. 在第4页结束后添加了模具表单. 在第3行中，在第35页的表14-2的“位2：P1.2 / VREG”下，进行了更改 是“CY7C63310 / CY7C638xx”而不是“CY7C63813”. 在第1行中，输入了第23页的表10-3中的“位6：USB CLK / 2禁用” 单词“时钟”而不是“晶体振荡器”. 输入单词“保留”，并将其相应的字段留空 第32页的表13-1“位[2：0]：VM [2：0]”下的子表. 在“位[7：6]：睡眠占空比[1：0]”下进行了以下更改： 0 0 =内部32 kHz低速振荡器的1/128个周期. 0 1 =内部32 kHz低速振荡器的1/512个周期. 1 0 =内部32 kHz低速振荡器的1/32周期. 1 1 =内部32 kHz低速振荡器的1/8个周期. 在第53页上的表17-2中，在第4行中，删除了“57”，并将单词“AND”改为 小写. 增加了32引脚锯开QFN引脚图，封装图和订购 信息. 在第10部分中删除了对32 kHz振荡器的3V的参考.时钟. 增加了关于SROM表读取的信息 - 第9.6节. 更新了第12.3节睡眠模式下的低功耗 - 包含集P10CR [1] - 在非USB模式操作期间. 增加了第25节 - 电压与CPU频率字符. 更新了P1DATA寄存器信息. Vreg可以独立于USB进行操作 连接. AC字符部分包括IMO和ILO特征. 已更新至数据表模板* E.
*K	2620679	CMCC / PYRS	08年12月12日	增加包处理信息
*L	2964259	AJHA	10年6月29日	订购信息表中添加了部件号CY7C63803-LQXC 添加包图（spec 001-13937） 从订购信息表中删除非活动部件. CY7C63310-PXC CY7C63801-PXC CY7C63833-LFXC 更新了软件包图.
*M	3074654	NXZ	10年10月29日	增加了订购代码定义，首字母缩略词和文档约定.
*N	3193555	KKCN	11年3月14日	在订购信息表中添加了部分CY7C63823-3XW14C.

35.销售，解决方案和法律信息

全球销售和设计支持

赛普拉斯拥有遍布全球的办事处，解决方案中心，制造商代表和分销商网络.找办公室
请在赛普拉斯地点与我们联系.

制品

汽车	cypress.com/go/automotive
时钟和缓冲器	cypress.com/go/clocks
接口	cypress.com/go/interface
照明和电源控制	cypress.com/go/powerpsoc
	cypress.com/go/plc
记忆	cypress.com/go/memory
光学和图像传感	cypress.com/go/image
的PSoC	cypress.com/go/psoc
触摸感应	cypress.com/go/touch
USB控制器	cypress.com/go/USB
无线/射频	cypress.com/go/wireless

PSoC解决方案

psoc.cypress.com/solutions
PSoC 1 | PSoC 3 | PSoC 5