芯海科技
CHIPSEA

CSU8RF3224 用户手册

带 12-bit ADC 的 8 位 RISC FLASH MCU

REV 1.0

通讯地址：深圳市南山区蛇口南海大道 1079 号花园城数码大厦 A 座 9 楼
邮政编码：518067
公司电话：+(86 755)86169257
传　　真：+(86 755)86169057
公司网站：www.chipsea.com
微 信 号：芯海科技
微信二维码：

版本历史

历史版本	修改内容	版本日期
REV 1.0	初始版本	2013-10-09

目 录

1 产品概述

1.1 功能描述

CSU8RF3224 是一个带 12-bit ADC 的 8 位 CMOS 单芯片 FLASH MCU，内置 2K×16 位 FLASH 程序存储器。

1.2 主要特性

高性能的 RISC CPU

- 8 位单片机 MCU
- 内置 2K×16 位程序存储器
- 128 字节数据存储器（SRAM）
- 96 字节的 E2PROM，用于数据存储
- 只有 43 条单字指令
- 8 级 PC 存储堆栈
- 8 级 PUSH 和 POP 堆栈

振荡器

- 内带 16MHz 振荡器，精度为±1%
- 外部 32768Hz 晶振（RTC）
  4MHz~16MHz 晶振
  ERC 8M@5V DC

外设特性

- 17 位双向 I/O 口,1 位输入口
- 2 路蜂鸣器输出
- 2 路 PWM 输出
- 5 个内部中断，2 个外部中断
- 8 个具有唤醒功能的输入口
- 高耐久性的 E2PROM：
  读写次数：至少 100,000 次
  保持时间：至少 10 年
- 5 路 12-bitADC
  — 内部 1.40V、VDD、外部输入三种参考电压选择
  — 带数字比较器
- 一个内置模拟比较器或运算放大器
- 低电压检测（LVD）引脚，内部提供 2.4V、3.6V 电压检测
- 6 个开漏输出口

专用微控制器的特性

- 上电复位（POR）
- 上电复位和硬件复位延迟定时器（40ms）
- 内带低电压复位（LVR）
- 定时器 0
  — 8 位可编程预分频的 8 位的定时计数器
- 定时/计数器 2
  — 8 位可编程预分频的 8 位的分频器
- 定时/计数器 3
  — 8 位可编程预分频的 8 位的分频器
- 扩展型看门狗定时器（32K WDT）
  — 可编程的时间范围

CMOS 技术

- 工作电压范围
  — VDD 2.3V~5.5V
- 工作温度范围
  — -40~85°C

低功耗特性

- MCU 工作电流
  — 正常模式 0.9mA@4MHz,3V
  — 正常模式 8uA@32KHz,3V
  — 休眠模式下的电流小于 1μA

封装

- SOP20/DIP20

应用范围

- 小家电
- 玩具
- 消费类

1.3 PIN 配置

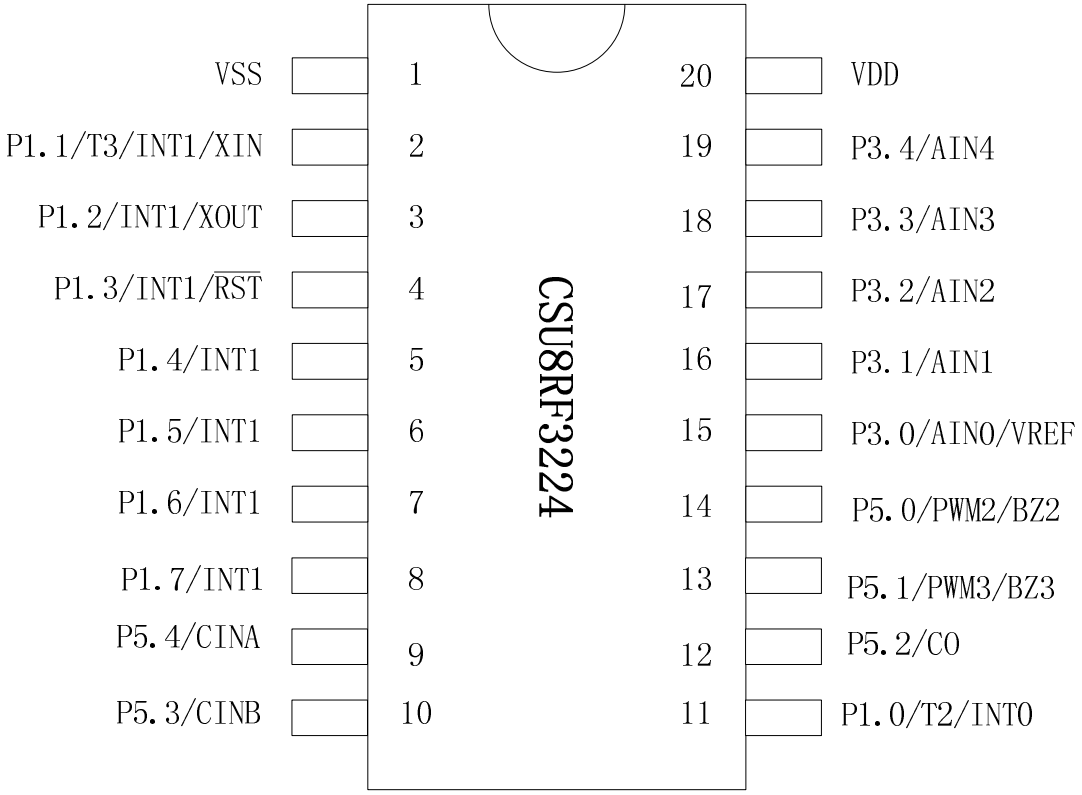


图1　　PIN 配置图

图2　　引脚说明表

管脚名称	输入/输出	管脚序号	描述
VSS	P	1	地
P1.1 /T3/INT1/XIN	I/O	2	IO，具有开漏输出功能；定时/计数器 3 外部输入；外部中断 1 输入，具有唤醒功能；外置晶振输入；
P1.2/INT1/XOUT	I/O	3	IO；外部中断 1 输入，具有唤醒功能；外置晶振输出；
P1.3/INT1/$\overline{RST}$	I	4	普通输入口；外部中断 1 输入，具有唤醒功能；复位输入；
P1.4/INT1	I/O	5	IO；外部中断 1 输入，具有唤醒功能；
P1.5/INT1	I/O	6	IO；外部中断 1 输入，具有唤醒功能；
P1.6/INT1	I/O	7	IO；外部中断 1 输入，具有唤醒功能；
P1.7/INT1	I/O	8	IO；外部中断 1 输入，具有唤醒功能；
P5.4/CINA	I/O	9	IO，具有开漏输出功能；比较器的输入端/运放的正端输入
P5.3/CINB	I/O	10	IO，具有开漏输出功能；比较器的输入端/运放的负端输入
P1.0/T2/INT0	I/O	11	IO；定时/计数器 2 外部输入；外部中断 0 输入，具有唤醒功能；
P5.2/CO	I/O	12	IO，具有开漏输出功能；比较器/运放的输出
P5.1 /PWM3/BZ3	I/O	13	IO，具有开漏输出功能；PWM3 输出；蜂鸣器 3 输出；
P5.0 /PWM2/BZ2	I/O	14	IO，具有开漏输出功能；PWM2 输出；蜂鸣器 2 输出；
P3.0/AIN0/VREF	I/O	15	IO；ADC 输入 0；ADC 参考电压输入
P3.1/AIN1	I/O	16	IO；ADC 输入 1

P3.2/AIN2	I/O	17	IO；ADC 输入 2
P3.3/AIN3	I/O	18	IO；ADC 输入 3
P3.4/AIN4	I/O	19	IO；ADC 输入 4
VDD	P	20	电源

2 标准功能

2.1 CPU核

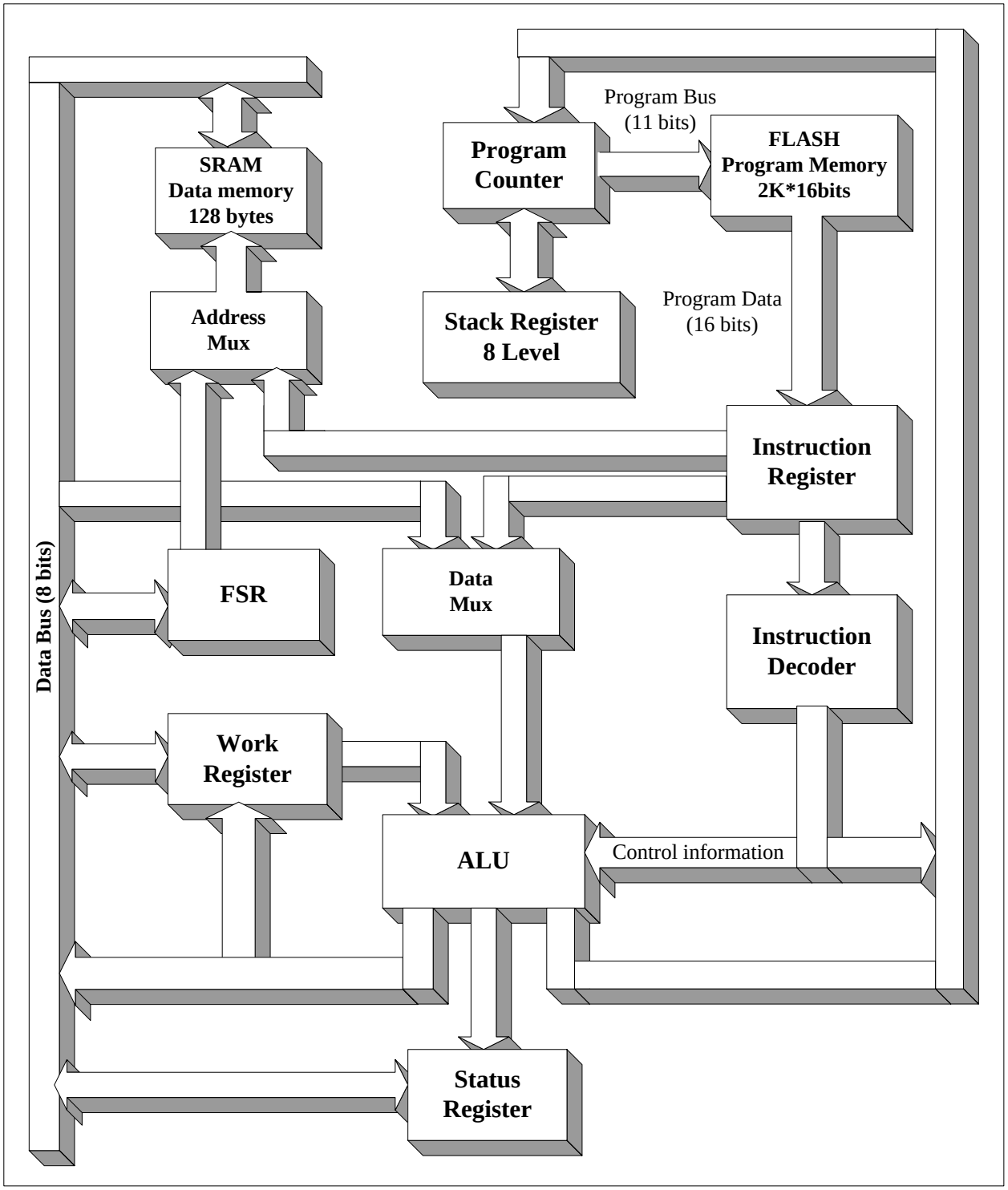


图3　　CSU8RF3224 CPU 核的功能模块图

据 CPU 核的功能模块图，它主要包含 7 个主要寄存器及 2 个存储器单元。

表 1　MCU 架构说明

模块名称	描述
程序计数器	此寄存器在 CPU 的工作周期间起到很重要的作用，它记录 CPU 每个周期处理程序存储器中指令的指针。在一个 CPU 周期中，程序计数器将程序存储器地址（11bits），指令指针推送到程序存储器，然后自动加 1 以进行下一次周期。
堆栈寄存器	堆栈寄存器是用来记录程序返回的指令指针。当程序调用函数，程序计数器会将指令指针推送到堆栈寄存器。在函数执行结束之后，堆栈寄存器会将指令指针送回到程序计数器以继续原来的程序处理。
指令寄存器	程序计数器将指令指针（程序存储器地址）推送到程序存储器，程序存储器将程序存储器的数据（16bits）推送到指令寄存器 。 CSU8RF3224 的指令是 16bits，包括 3 种信息：直接地址，立即数及控制信息。 直接地址（8bits）：数据存储器的地址。CPU 能利用此地址来对数据存储器进行操作。 立即数（8bits）：CPU 通过 ALU 利用此数据对工作寄存器进行操作。 控制信息：它记录着 ALU 的操作信息。
指令译码器	指令寄存器将控制信息推送到指令译码器以进行译码，然后译码器将译码后的信息发送到相关的寄存器。
算术逻辑单元	算术逻辑单元不仅能完成 8 位二进制的加，减，加 1，减 1 等算术计算，还能对 8 位变量进行逻辑的与，或，异或，循环移位，求补，清零等逻辑运算。
工作寄存器	工作寄存器是用来缓存数据存储器中某些存储地址的数据。
状态寄存器	当 CPU 利用 ALU 处理寄存器数据时，如下的状态会随着如下顺序变化：PD，TO，DC，C 及 Z。
文件选择寄存器	在 CSU8RF3224 的指令集中，FSR 是用于间接数据处理（即实现间接寻址）。用户可以利用 FSR 来存放数据存储器中的某个寄存器地址，然后通过 IND 寄存器对这个寄存器进行处理。
程序存储器	CSU8RF3224 内带 2K×16 位的 FLASH 作为程序存储器。由于指令的操作码（OPCODE）是 16bits，用户最多只能编程 2K 的指令。程序存储器的地址总线是 11bits，数据总线是 16bits。
数据存储器	CSU8RF3224 内带 128bytes 的 SRAM 作为数据存储器。此数据存储器的地址总线是 7bits，数据总线是 8bits。

2.1.1 存储器

（1）程序存储器

程序存储器主要用于指令的存储，该程序存储器是 2K*16bit 的程序 FLASH，对于程序员来说，该存储器只读，不可以写入。系统的 reset 地址为 0x000，中断入口地址为 0x004，需要注意的一点就是所有的中断共用同一个中断入口地址。数据 E2PROM 的地址范围为 0x800~0x82F。

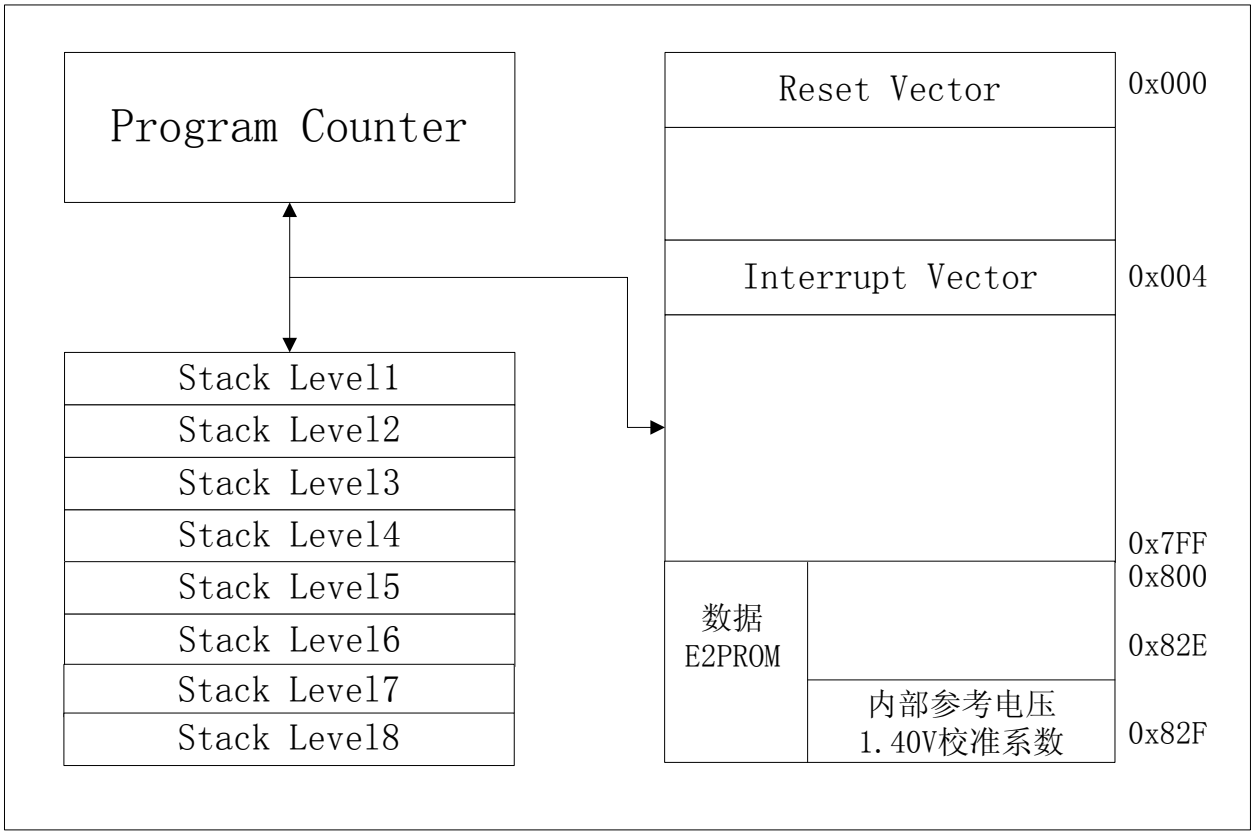


图4　　程序存储器

（2）数据存储器

数据存储器主要用于程序运行过程中，全局以及中间变量的存储。该存储器分为三个部分。地址的 00H 至 07H 是系统特殊功能寄存器，例如间接地址，间接地址指针，状态寄存器，工作寄存器，中断标志位，中断控制寄存器。地址的 08H 至 7FH 外设特殊功能寄存器，例如 IO 端口，定时器，系统特殊功能寄存器和外设特殊功能寄存器是用寄存器实现，而通用数据存储器是 RAM 实现，可以读出也可以写入。

表 2　数据存储器地址分配

数据存储器	起始地址	结束地址
系统特殊功能寄存器	0x00	0x07
外设特殊功能寄存器	0x08	0x7F
通用数据存储器	0x80	0xFF

通过 IND0 以及 FSR0 这两个寄存器可以对数据存储器以及特殊功能寄存器进行间接访问。当从间接地址寄存器(IND0)读入数据时，MCU 实际上是以 FSR0 中的值作为地址去访问数据存储器得到数据。当向间接寄存器(IND0)写入数据时，MCU 实际上是以 FSR0 中的值作为地址去访问数据存储器将值存入该地址。其访问方式见。

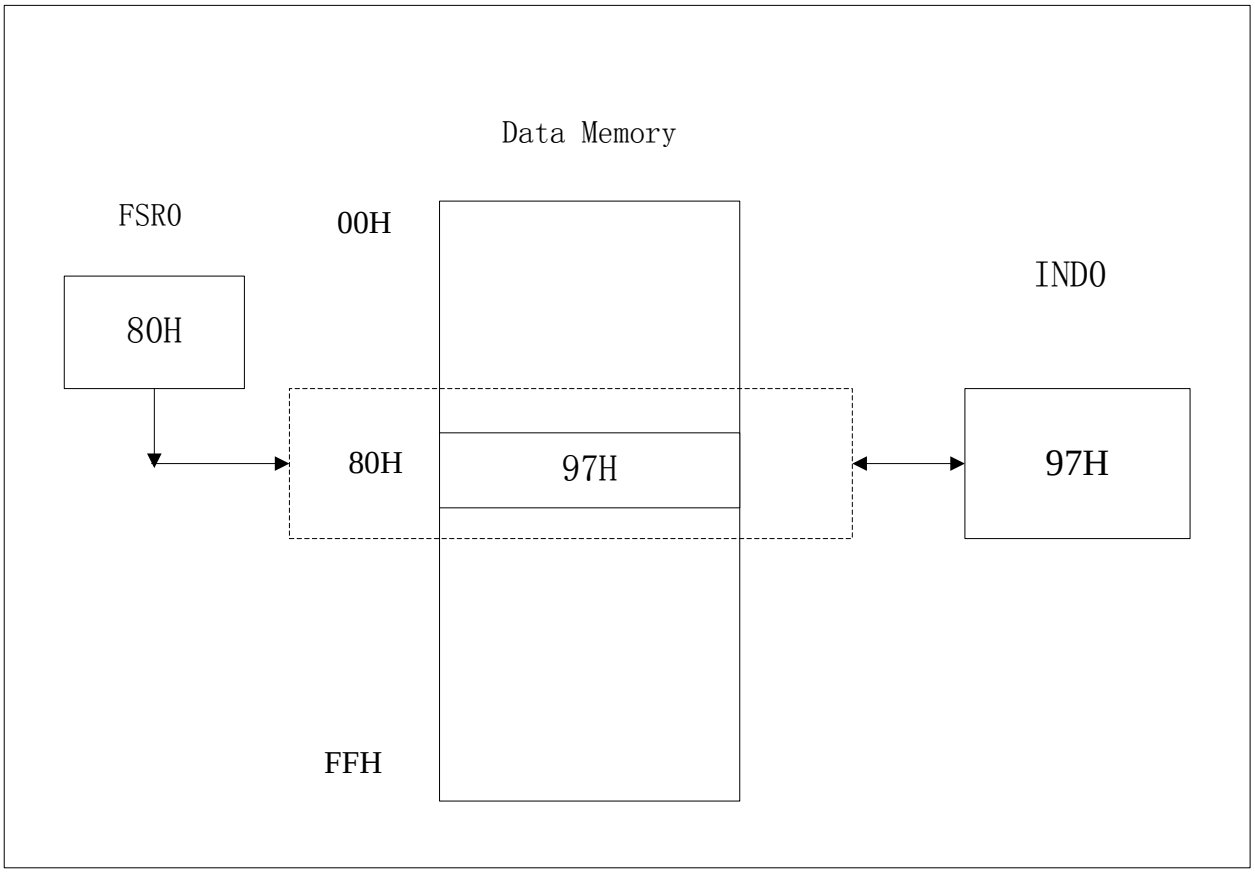


图5　　间接地址访问

2.1.2 状态寄存器

状态寄存器包含 ALU 的算术状态及复位状态。状态寄存器类似于其它寄存器，可以作为任何指令的目标寄存器。如果状态寄存器是某条指令的目标寄存器，而且影响到 Z，DC 或 C 位，那么对这三个位的写是不使能。这些位是由器件逻辑进行置位或清零。TO 及 PD 位是不可写的。

状态寄存器（地址为 04h）

特性	R-0	R-0	U-0	R-0	R-0	R/W-0	R/W-0	R/W-0
STATUS	LVD36	LVD24		PD	TO	DC	C	Z
	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0

Bit 7　LVD36：3.6V LVD 工作电压标志位，只有当代码选项 LVD_SEL 为 2’b01 和 2’b10 有效
　　1：表明芯片工作电压低于 3.6V
　　0：表明芯片工作电压高于 3.6V
Bit 6　LVD24：2.4V LVD 工作电压标志位，只有当代码选项 LVD_SEL 为 2’b01 有效
　　1：表明芯片工作电压低于 2.4V
　　0：表明芯片工作电压高于 2.4V
Bit 4　PD：掉电标志位。通过对此位写 0 清零，sleep 后置此位
　　1：执行 SLEEP 指令后
　　0：上电复位后或硬件复位或 CLRWDT 指令之后
Bit 3　TO：看门狗定时溢出标志。通过对此位写 0 清零，看门狗定时溢出设置此位
　　1：看门狗定时溢出发生
　　0：上电复位后或硬件复位或 CLRWDT 指令后或 SLEEP 指令后
Bit 2　DC：半字节进位标志/借位标志
　　用于借位时，极性相反
　　1：结果的第 4 位出现进位溢出
　　0：结果的第 4 位不出现进位溢出
Bit 1　C：　进位标志/借位标志
　　用于借位时，极性相反
　　1：结果的最高位（MSB）出现进位溢出
　　0：结果的最高位（MSB）不出现进位溢出
Bit 0　Z：零标志
　　1：算术或逻辑操作是结果为 0
　　0：算术或逻辑操作是结果不为 0

特性（Property）：
R = 可读位　　W = 可写位　　U = 无效位
-n = 上电复位后的值　　‘1’ = 位已设置　　‘0’ = 位已清零　　X = 不确定位

2.1.3 SFR

特殊功能寄存器（SFR）包含系统专用寄存器和辅助专用寄存器。

系统专用寄存器用于完成 CPU 核的功能，由间接地址，间接地址指针，状态寄存器，工作寄存器，中断标志及中断控制寄存器。

辅助专用寄存器是为辅助功能而设计，比如 I/O 口，定时器，信号的条件控制寄存器。

表 3　寄存器列表

地址	名称	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	上电复位值
00h	IND0	以 FSR0 中内容作为地址的数据存储器中的数据								xxxxxxxx
02h	FSR0	间接数据存储器的地址指针 0								00000000
04h	STATUS	LVD36	LVD24		PD	TO	DC	C	Z	xxu00000
05h	WORK	工作寄存器								00000000
06h	INTF		TM2IF		TM0IF	SRADIF		E1IF	E0IF	u0u00u00
07h	INTE	GIE	TM2IE		TM0IE	SRADIE		E1IE	E0IE	00u00u00
0Ah	EADRH					PAR[11:8]				uuuu0000
0Bh	EADRL	PAR[7:0]								00000000
0Ch	EDATH	EDATH[7:0]								00000000
0Dh	WDTCON	WDTEN					WTS[2:0]			0uuuu000
0Eh	WDTIN	WDTIN[7:0]								11111111
0Fh	TM0CON	T0EN	T0RATE[2:0]				T0RSTB	T0SEL[1:0]		0000u100
10h	TM0IN	TM0IN[7:0]								11111111
11h	TM0CNT	TM0CNT[7:0]								00000000
16h	MCK	CST	CST_IN	CST_WDT	EO_SLP				CLKSEL	0010uuu0
17h	TM2CON	T2EN	T2RATE[2:0]			T2CKS	T2RSTB	T2OUT	PWM2OUT	00000100
18h	TM2IN	TM2IN[7:0]								11111111
19h	TM2CNT	TM2CNT[7:0]								00000000
1ah	TM2R	TM2R[7:0]								00000000
1bh	TM3CON	T3EN	T3RATE[2:0]			T3CKS	T3RSTB	T3OUT	PWM3OUT	00000100
1ch	TM3IN	TM3IN[7:0]								11111111
1dh	TM3CNT	TM3CNT[7:0]								00000000
1eh	TM3R	TM3R[7:0]								00000000
20h	PT1	PT1[7:0]								xxxxxxxx
21h	PT1EN	PT1EN[7:4]					PT1EN[2:0]			0000u000
22h	PT1PU	PT1PU[7:0]								00000000
23h	PT1CON0	PT11OD	PT1W1[3:0]				E1M	E0M[1:0]		00000000
28h	PT3				PT3[4:0]					uuuxxxxx
29h	PT3EN				PT3EN[4:0]					uuu00000
2ah	PT3PU				PT3PU[4:0]					uuu00000
2bh	PT3CON				PT3CON[4:0]					uuu00000
30h	PT5				PT5[4:0]					uuuxxxxx
31h	PT5EN				PT5EN[4:0]					uuu00000
32h	PT5PU				PT5PU[4:0]					uuu00000
33h	PT5CON				PT5OD[4:0]					uuu00000
38h	PT1CON1						PT1W2[2:0]			uuuuu000
3ch	INTF2				TM3IF					uuu0uuuu
3dh	INTE2				TM3IE					uuu0uuuu
3eh	INTF3	CMPIF								0uuuuuuu
3fh	INTE3	CMPIE								0uuuuuuu
50h	SRADCON0			SRADACKS[1:0]				SRADCKS[1:0]		uu00uu00
51h	SRADCON1	SRADEN	SRADS	OFTEN	CALIF	ENOV	OFFEX	VREFS[1:0]		00000000
52h	SRADCON2	CHS[3:0]								0000uuuu
54h	SRADL	SRAD[7:0]								00000000

55h	SRADH					SRAD[11:8]						uuuu0000
56h	SROFTL	SROFT[7:0]										00000000
57h	SROFTH					SROFT[11:8]						uuuu0000
6ah	CMPCON	CMPEN	COS[2:0]							CMPOUT		0000uuu0

注：进行读操作时，无效位读出为 0

特性（Property）：
R = 可读位　　W = 可写位　　U = 无效位
-n = 上电复位后的值　　‘1’ = 位已设置　　‘0’ = 位已清零　　X = 不确定位

2.2 时钟系统

2.2.1 概述

芯片的时钟系统包括内置 16MHz 的 RC 振荡时钟（IHRC）、外置高速晶振、内置低速 32KHz 的 WDT 时钟、外置低速的晶振时钟、外部 RC 时钟、外部时钟源。除去 WDT 时钟外，以上时钟都可以做为系统时钟源 Fosc。Fcpu 是 CPU 时钟频率。

普通模式（高速时钟）：Fcpu=Fosc/N，N=4、8、16、32

低速模式（低速时钟）：Fcpu=Fosc/N，N=4、8、16、32

2.2.2 时钟框图

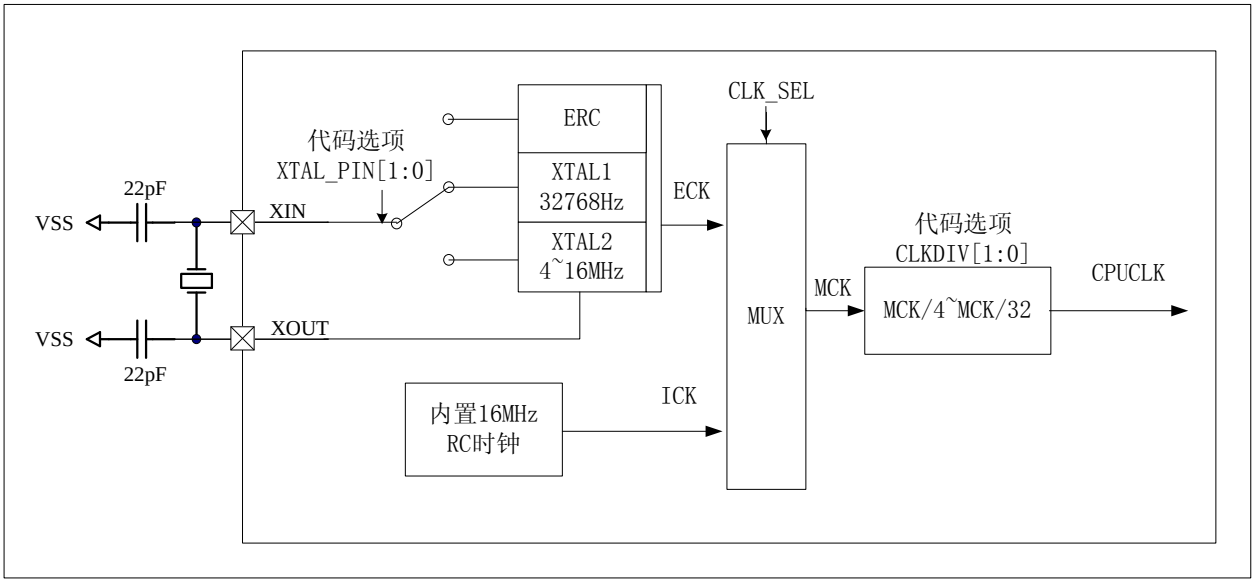


图6　　CSU8RF3224 振荡器状态框图 A

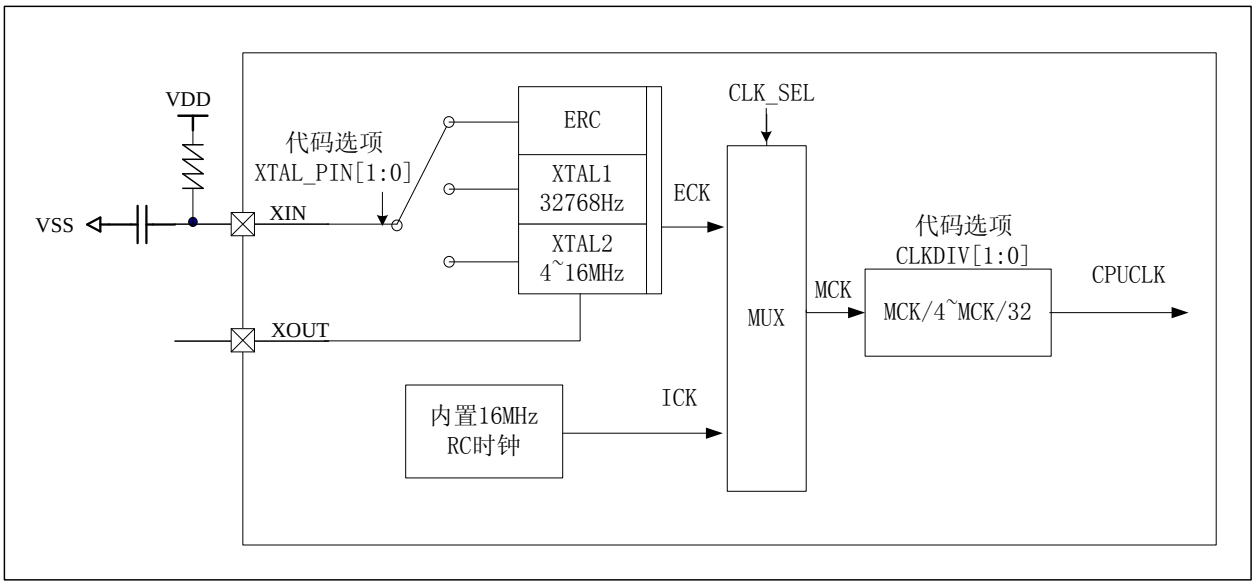


图7　　CSU8RF3224 振荡器状态框图 B

2.2.3 寄存器

表 4　CSU8RF3224 时钟系统寄存器列表

地址	名称	Bit7	Bits6	Bit5	Bits4	Bit3	Bits2	Bit1	Bit0	上电 复位值
16H	MCK	CST	CST_IN	CST_WDT	EO_SLP				CLKSEL	0010uuu0

表 5　MCK 寄存器各位功能表

位地址	标识符	功能
7	CST	外部晶振启动开关 1：外部晶振关闭 0：外部晶振打开
6	CST_IN	内部晶振启动开关 1：内部晶振关闭 0：内部晶振打开
5	CST_WDT	内部 WDT 晶振启动开关 1：内部 WDT 晶振关闭 0：内部 WDT 晶振打开
4	EO_SLP	外部低速晶振控制位 1：如果选择的是外部低速晶振（32768Hz），在 sleep 模式下不关闭外部晶振 0：sleep 模式下关闭外部晶振
0	CLKSEL	时钟源选择位 CLK_SEL：CPU 时钟 0：内部晶振系统时钟 1：外部晶振系统时钟

对 MCK 寄存器进行写操作时，建议使用 bcf 或 bsf 指令。

注意：把 CPU 时钟由内部晶振切换到外部晶振，并把内部晶振关闭时应按照以下顺序执行

```
bcf mck,7        ;打开外部晶振
call delay       ;低速 32768Hz 推荐延迟 2S；高速 16MHz 推荐延迟 15mS
bsf mck,0        ;切换到外部晶振
nop
nop
bsf mck,6        ;关闭内部晶振
…
```

2.2.4 内部高速RC时钟

内部高速 RC 时钟（16MHz），通过寄存器 CST_IN 使能开关。当使用内部高速 RC 时钟做为系统的主时钟时，外部晶振引脚 PT1.1、PT1.2 可以通过代码选项配置做为普通的 GPIO 口。

2.2.5 内部低速wdt时钟

内部低速 wdt 时钟（32kHz），通过寄存器 CST_WDT 使能开关。内部 wdt 时钟不能做为系统主时钟，只能做为 WDT 使用和定时器 0 使用。

2.2.6 外部高速晶振时钟

外部高速晶振时钟，通过代码选项配置为外部高速时钟，同时通过寄存器 CST 使能开关。此时，PT1.1、PT1.2 口做为晶振引脚。

2.2.7 外部低速晶振时钟

外部低速晶振时钟，通过代码选项配置为外部低速时钟，同时通过寄存器 CST 使能开关。此时，PT1.1、PT1.2 口做为晶振引脚。

2.2.8 外部RC振荡器

外部 RC 振荡器，通过代码选项配置为外部 RC 振荡器，同时通过寄存器 CST 使能开关。此时，PT1.1 口做为 RC 输入引脚，PT1.2 做为普通的 GPIO 口。外置 RC 振荡器的频率最高可以到 8MHz，最低可以到几 KHz，甚至更低。

2.2.9 外部时钟源

外部时钟源，通过代码选项配置为外部高速时钟或外部低速时钟或外部 RC 振荡器，同时通过寄存器 CST 使能开关。外部时钟源通过 PT1.1 口灌入时钟。当外部时钟源频率较快时，代码选项建议选择外部高速时钟或外部 RC 时钟；当外部时钟源频率较低时（与外部低速时钟频率相当），代码选项可以选择外部低速时钟或外部 RC 时钟。

2.3 复位系统

CSU8RF3224 有以下方式复位：

1） 上电复位
2） $\overline{RST}$ 硬件复位（正常操作）
3） $\overline{RST}$ 硬件复位（从 Sleep 模式）
4） WDT 复位（正常操作）
5） WDT 复位（从 Sleep 模式）
6） 低电压复位（LVR）

上述任意一种复位发生时，所有系统寄存器恢复默认状态（WDT 复位 TO、PD 标志位除外），程序停止运行，同时程序计数器 PC 清零。复位结束后，系统从向量 000H 重新开始。各种复位情况下的 TO，PD 标志位如下表所示。

表 6　复位信号和状态寄存器关系

条件	TO	PD
上电复位	0	0
$\overline{RST}$ 硬件复位（正常操作）	0	0
$\overline{RST}$ 硬件复位（从 Sleep 模式）	0	0
WDT 复位（正常操作）	1	不变
WDT 复位（从 Sleep 模式）	1	不变
低电压复位	0	0

下图给出了复位电路原理图。

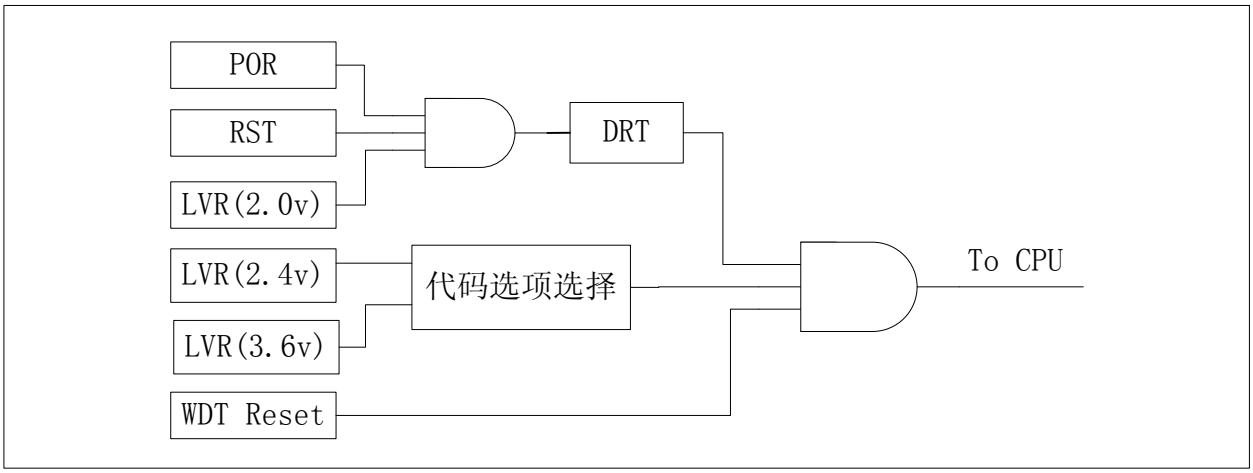


图8　　复位电路原理图

任何一种复位情况都需要一定的响应时间，系统提供完善的复位流程以保证复位动作的顺利进行。对于不同类型的振荡器起振的时间不同，所以完成复位的时间也有所不同。RC 振荡器起振时间最短，外置低速晶振起振时间最长。所以在有外部晶振电路应用的情况下，用户应在上电复位后，预留一定的时间再从内部 RC 时钟切换到外部晶振电路。用户在终端使用过程中，应注意考虑主机对上电复位的要求。

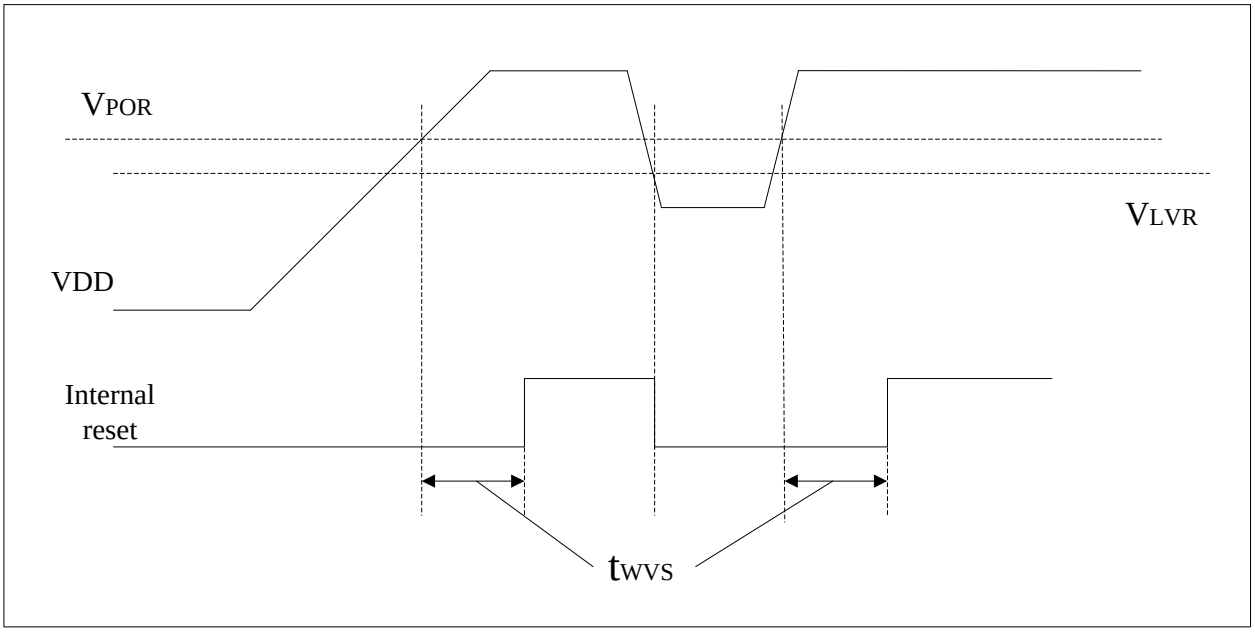


图9　　上电复位电路示例及上电过程

参数	最小值	典型值	最大值
VPOR	1.8V	2.0V	2.2V
VLVR	1.8V	2.0V	2.2V
t_{WVS} (测试条件：VDD=5V，T=25℃)	32ms	40ms	48ms

VPOR：上电复位
VLVR：低电压复位
t_{WVS}：　等待电压稳定时间

2.3.1　上电复位

系统上电呈现逐渐上升的曲线形式，需要一定时间才能达到正常的工作电压（对于不同的指令周期所需工作电压是不同的，指令周期越快相应所需的工作电压就越高，见 5.2 直流特性）。要求用户系统的上电速度要大于 0.15V/mS，尤其是要注意指令周期是 4MHz 时，因为他要求的工作电压最高。

2.3.2　看门狗复位

看门狗复位是一种系统的保护设置。在正常状态下，程序将看门狗定时器清零。如出错，系统处于未知状态，此时利用看门狗复位。看门狗复位后，系统重新进入正常状态。

2.3.3　掉电复位

掉电复位针对外部引起的系统电压跌落情况，例如受到干扰或者负载变化。系统掉电可能会引起系统工作状态不正常或者程序执行错误。

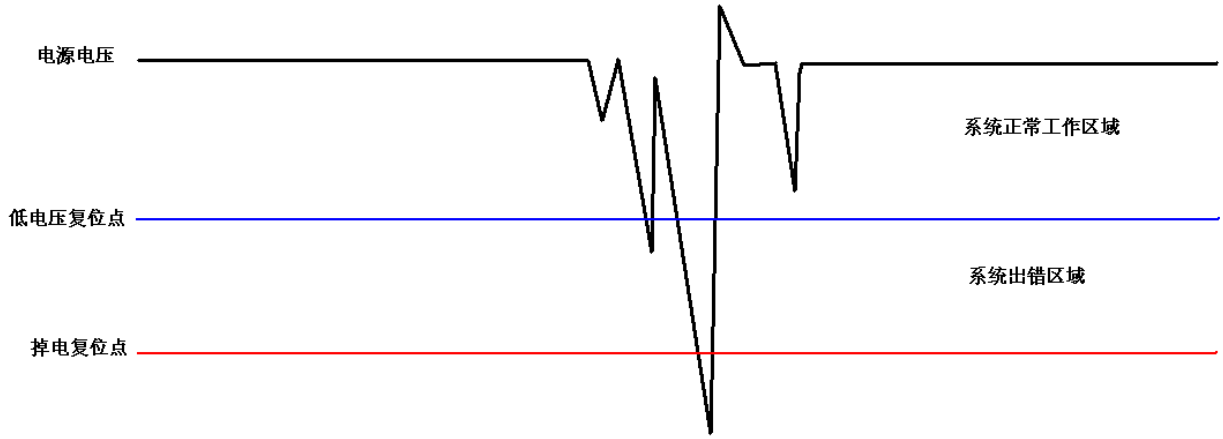


图10　系统掉电复位示意图

电压跌落可能会进入系统死区。进入系统死区，即电源电压不能满足系统的最小工作电压要求。系统掉电复位示意图如上图所示。芯片的掉电复位点在 2.0V，芯片的低电压复位点可以通过代码选项设置成 2.4V 或者不设置低电压复位点。

为避免进入系统死区，建议利用低电压复位（LVR）功能，尤其是指令周期是高速应用的情况。

不同指令周期的系统出错区域不同，取决于指令周期工作电压范围，见 5.2。掉电复位性能的改善可以通过如下几点实现：

1) 低电压复位（LVR）
2) 看门狗复位
3) 降低系统指令周期
4) 采用外部复位电路（稳压二极管复位电路；电压偏移复位电路；外部 IC 复位）

2.3.4 外部硬件复位

外部复位由代码选项 RESET_PIN 控制，见 3.9。通过设置该代码选项，可使能外部硬件复位功能。外部硬件复位引脚为施密特触发结构，低电平有效。硬件复位引脚为高电平时，系统正常工作；硬件复位引脚为低电平时，系统复位。

在芯片代码选项使能外部硬件复位功能后，需要注意的是：在系统上电完成后，外部复位需要输入高电平，否则，系统会一直复位，直到外部硬件复位结束。

外部硬件复位可以在上电过程中使用系统复位。良好的外部复位电路可以保护系统避免进入系统死区。

2.4 中断

CSU8RF3224 有 7 个中断源，只有 1 个中断入口地址 004H。与中断相关的 SFR：中断使能控制寄存器 INTE 和中断标志位寄存器 INTF。这 7 个中断源都各自有一个中断使能，和一个总使能位 GIE，并且它们的标志位硬件置位，软件清 0。

当响应中断时，会把当前的 PC 值入栈保护，并把 PC 置为 004H，同时把总使能位 GIE 清 0。执行完中断服务程序，并用 RETFIE 返回到之前的主程序，并把 GIE 置 1。

所有的中断都可以唤醒 sleep 睡眠模式和 halt 停止模式。

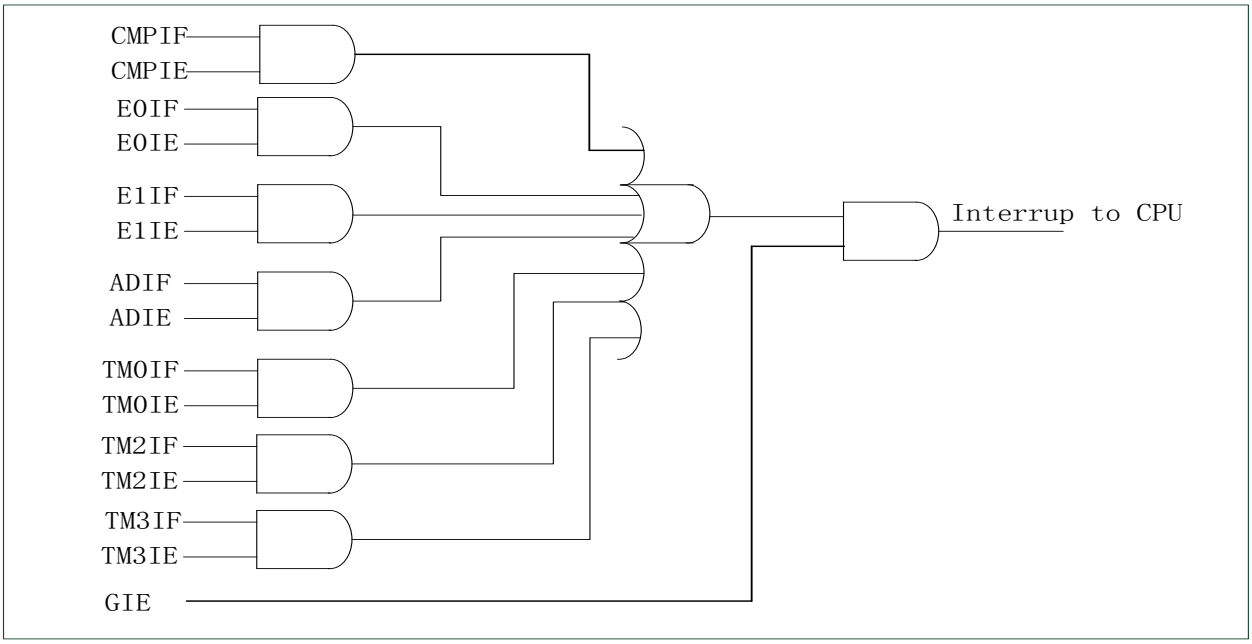


图11　中断逻辑

2.4.1 中断使能寄存器

INTE 寄存器（地址为 07h）

特性	R/W-0	R/W-0	U-0	R/W-0	R/W-0	U-0	R/W-0	R/W-0
INTE	GIE	TM2IE		TM0IE	SRADIE		E1IE	E0IE
	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0

Bit 7 GIE：全局中断使能标志
1 = 使能所有非屏蔽中断
0 = 不使能所有中断

Bit 6 TM2IE：8-Bit 定时/计数器 2 中断使能标志
1 = 使能定时/计数器 2 中断
0 = 不使能定时/计数器 2 中断

Bit 4 TM0IE：8-Bit 定时 0 器中断使能标志
1 = 使能定时器 0 中断
0 = 不使能定时器 0 中断

Bit 3 SRADIE：AD 中断使能标志
1 = 使能 AD 中断
0 = 不使能 AD 中断

Bit 1 E1IE：外部中断 1 使能标志
1 =使能外部中断 1
0 = 不使能外部中断 1

Bit 0 E0IE：外部中断 0 使能标志
1 = 使能外部中断 0
0 = 不使能外部中断 0

INTE2 寄存器（地址为 3dh）

特性	U-0	U-0	U-0	R/W-0	U-0	U-0	U-0	U-0
INTE2				TM3IE				
	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0

Bit 4 TM3IE：8-Bit 定时/计数器 3 中断使能标志
1 = 使能定时/计数器 3 中断
0 = 不使能定时/计数器 3 中断

INTE3 寄存器（地址为 3fh）

特性	R/W-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
INTE2	CMPIE							
	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0

Bit 7 CMPIE：比较器中断使能标志
1 = 使能比较器中断
0 = 不使能比较器中断

特性（Property）：
R = 可读位　　W = 可写位　　U = 无效位
-n = 上电复位后的值　　‘1’ = 位已设置　　‘0’ = 位已清零　　X = 不确定位

2.4.2 中断标志寄存器

中断标志位都是硬件置 1，软件清 0。**比较器中断标志位，就算其中断使能不为 1 时，也可能硬件置位；但其他的中断标志位只有在其对应的中断使能位为 1 的情况下，才有可能硬件置 1。**

INTF 寄存器（地址为 06h）

特性	U-0	R/W-0	U-0	R/W -0	R/W -0	U-0	R/W -0	R/W -0
INTF		TM2IF		TM0IF	SRADIF		E1IF	E0IF
	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0

Bit 6 TM2IF：8-Bit 定时/计数器 2 中断标志，软件清零，硬件置高
1 = 发生定时中断，必须软件清 0
0 = 没发生定时中断

Bit 4 TM0IF：8-Bit 定时器 0 中断标志，软件清零，硬件置高
1 = 发生定时中断，必须软件清 0
0 = 没发生定时中断

Bit 3 SRADIF：AD 中断中断标志，软件清零，硬件置高
1 = 发生 AD 中断，必须软件清 0
0 = 没发生 AD 中断

Bit 1 E1IF：外部中断 1 中断标志，软件清零，硬件置高
1 =外部中断 1 发生中断，必须软件清 0
0 =外部中断 1 没发生中断

Bit 0 E0IF：外部中断 0 中断标志，软件清零，硬件置高
1 = 外部中断 0 发生中断，必须软件清 0
0 =外部中断 0 没发生中断

INTF2 寄存器（地址为 3ch）

特性	U-0	U-0	U-0	R/W -0	U-0	U-0	U-0	U-0
INTF2				TM3IF				
	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0

Bit 4 TM3IF：8-Bit 定时/计数器 3 中断标志，软件清零，硬件置高
1 = 发生定时中断，必须软件清 0
0 = 没发生定时中断

INTF3 寄存器（地址为 3eh）

特性	R/W -0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
INTF2	CMPIF							
	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0

Bit 7 CMPIF：比较器中断标志，软件清零，硬件置高
1 = 发生比较器中断，必须软件清 0
0 = 没发生比较器中断

特性（Property）：
R = 可读位　　W = 可写位　　U = 无效位
-n = 上电复位后的值　　‘1’ = 位已设置　　‘0’ = 位已清零　　X = 不确定位

2.4.3 外部中断 0

PT1.0 为外部中断 0 的输入端。触发方式由 PT1CON0 寄存器中的 E0M[1:0]寄存器决定。INTE 寄存器中的 E0IE 为外部中断 0 的使能位，INTF 寄存器中的 E0IF 为中断标志位，硬件置 1，软件清 0。可唤醒 sleep 或 halt 模式。只有 E0IE 使能的情况下，PT1.0 被触发，中断标志位 E0IF 才会置 1。

PT1CON0 寄存器（地址为 23h）

特性	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
PT1CON0						E1M	E0M[1:0]	
	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0

Bit 2 E1M：外部中断 1 触发模式
1 = 外部中断 1 为下降沿触发
0 = 外部中断 1 在状态改变时触发

Bit 1-0 E0M[1:0]：外部中断 0 触发模式
11 = 外部中断 0 在状态改变时触发
10 = 外部中断 0 在状态改变时触发
01 = 外部中断 0 为上升沿触发
00 = 外部中断 0 为下降沿触发

2.4.4 外部中断 1

PT1.1、PT1.2、PT1.3、PT1.4、PT1.5、PT1.6 和 PT1.7 都可作为外部中断 1 的输入端。触发方式由 PT1CON0 寄存器中的 E1M 寄存器决定。INTE 寄存器中的 E1IE 为外部中断 0 的使能位，INTF 寄存器中的 E1IF 为中断标志位，硬件置 1，软件清 0。只有 E1IE 使能的情况下，外部中断 1 被触发，中断标志位 E1IF 才会置 1。

PT1CON0 寄存器（地址为 23h）

特性	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
PT1CON0		PT1W[3:0]						
	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0

Bit 6 PT1W[3]:PT1.5 外部中断 1 使能
0 = 禁止 PT1.5 外部中断 1
1 = 使能 PT1.5 外部中断 1

Bit 5 PT1W[2]:PT1.4 外部中断 1 使能
0 = 禁止 PT1.4 外部中断 1
1 = 使能 PT1.4 外部中断 1

Bit 4 PT1W[1]:PT1.3 外部中断 1 使能
0 = 禁止 PT1.3 外部中断 1
1 = 使能 PT1.3 外部中断 1

Bit 3 PT1W[0]:PT1.1 外部中断 1 使能
0 = 禁止 PT1.1 外部中断 1
1 = 使能 PT1.1 外部中断 1

PT1CON0 寄存器（地址为 23h）

特性	U-0	U-0	U-0	U-0	U-0	R/W-0	R/W-0	R/W-0
PT1CON1						PT1W2[2:0]		
	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0

Bit 2 PT1W2[2]:PT1.7 外部中断 1 使能
0 = 禁止 PT1.7 外部中断 1
1 = 使能 PT1.7 外部中断 1
Bit 1 PT1W2[1]:PT1.6 外部中断 1 使能
0 = 禁止 PT1.6 外部中断 1
1 = 使能 PT1.6 外部中断 1
Bit 0 PT1W2[0]:PT1.2 外部中断 1 使能
0 = 禁止 PT1.2 外部中断 1
1 = 使能 PT1.2 外部中断 1

特性（Property）：
R = 可读位 W = 可写位 U = 无效位
-n = 上电复位后的值 ‘1’ = 位已设置 ‘0’ = 位已清零 X = 不确定位

2.4.5 AD中断溢出

INTE 寄存器中的 SRADIE 为 ADC 中断的使能位，INTF 寄存器中的 SRADIF 为中断标志位，软件清 0。当 ADC 转换完成时，只有 SRADIE 使能的情况下，SRADIF 才能硬件置 1。

2.4.6 定时器 0 溢出中断

INTE 寄存器中的 TM0IE 为定时器 0 中断的使能位，INTF 寄存器中的 TM0IF 为中断标志位，软件清 0。当定时器 0 溢出时，只有 TM0IE 使能的情况下，TM0IF 才能硬件置 1。

2.4.7 定时/计数器 2 溢出中断

INTE 寄存器中的 TM2IE 为定时/计数器 2 中断的使能位，INTF 寄存器中的 TM2IF 为中断标志位，软件清 0。当定时/计数器 2 溢出时，只有 TM2IE 使能的情况下，TM2IF 才能硬件置 1。

2.4.8 定时/计数器 3 溢出中断

INTE2 寄存器中的 TM3IE 为定时/计数器 3 中断的使能位，INTF2 寄存器中的 TM3IF 为中断标志位，软件清 0。当定时/计数器 3 溢出时，只有 TM3IE 使能的情况下，TM3IF 才能硬件置 1。

2.4.9 比较器中断

INTE3 寄存器中的 CMPIE 为比较器中断的使能位，INTF3 寄存器中的 CMPIF 为中断标志位，软件清 0。当比较器结果改变时，不管 CMPIE 是否使能，CMPIF 也会硬件置 1。

2.4.10 PUSH和POP处理

CSU8RF3224 有 8 级的 PUSH 和 POP 堆栈。有中断请求被响应后，程序跳转到 004h 执行子程序。响应中断之前必须保存 WORK 和 STATUS 中的的标志位(只保存 C，DC，Z)。芯片提供 PUSH 和 POP 指令进行入栈保存和出栈恢复，从而避免中断中断结束后程序运行错误。子程序中也可以使用 PUSH 和 POP 指令对 WORK 和 STATUS(C，DC，Z)进行保存和恢复。

```
        …
        org 004H
        goto int_server
        …
int_server:
        push
        btfsc intf,e0if    ;判断外部中断 0 标志
        goto ex0_int
        btfsc intf,e1if    ;判断外部中断 1 标志
        goto ex1_int
        btfsc intf,tm0if   ;判断定时器 0 中断标志
        goto tm0_int
        btfsc intf,tm2if   ;判断定时/计数器 2 中断标志
        goto tm2_int
        btfsc intf,tm3if   ;判断定时/计数器 3 中断标志
        goto tm3_int
        …
ex0_int:
        bcf intf, e1if     ;清除 e1if
        …
        pop
        retfie
        …
```

2.5 定时器 0

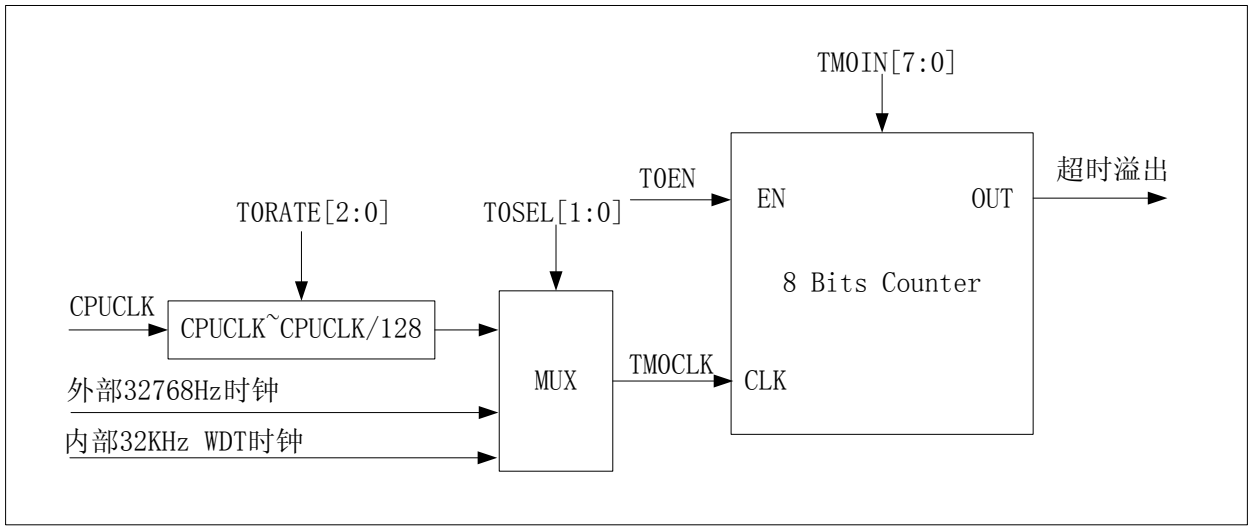


图12 定时器 0 功能框图

定时器 0 模块的输入为 CPUCLK。在定时器 0 模块集成了一个分频器，分频的时钟 TM0CLK 作为 8 bits 计数器的输入时钟。当用户设置了定时器 0 模块的使能标志，8 bits 计数器将启动，将会从 000H 递增到 TM0IN。用户需要设置 TM0IN（定时器 0 模块中断信号选择器）以选择定时超时中断信号。当定时超时发生时，中断标志位会自设置，程序计数器会跳转到 004H 以执行中断服务程序。

表 7 定时器 0 寄存器列表

地址	名称	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	上电复位值
06H	INTF				TM0IF					u0u00u00
07H	INTE	GIE			TM0IE					00u00u00
0FH	TM0CON	T0EN	T0RATE[2:0]				T0RSTB	T0SEL[1:0]		0000u100
10H	TM0IN	TM0IN[7:0]								11111111
11H	TM0CNT	TM0CNT[7:0]								00000000

表 8　TM0CON 寄存器各位功能表

<table>
<tr><th>位地址</th><th>标识符</th><th>功能</th></tr>
<tr><td>7</td><td>T0EN</td><td>定时器 0 使能位
1：使能定时器 0
0：禁止定时器 0</td></tr>
<tr><td>6:4</td><td>T0RATE[2:0]</td><td>定时器 0 时钟选择
<table>
<tr><td>T0RATE [2:0]</td><td>TM0CLK</td></tr>
<tr><td>000</td><td>CPUCLK</td></tr>
<tr><td>001</td><td>CPUCLK /2</td></tr>
<tr><td>010</td><td>CPUCLK /4</td></tr>
<tr><td>011</td><td>CPUCLK /8</td></tr>
<tr><td>100</td><td>CPUCLK /16</td></tr>
<tr><td>101</td><td>CPUCLK /32</td></tr>
<tr><td>110</td><td>CPUCLK /64</td></tr>
<tr><td>111</td><td>CPUCLK /128</td></tr>
</table></td></tr>
<tr><td>2</td><td>T0RSTB</td><td>定时器 0 复位
1：禁止定时器 0 复位
0：使能定时器 0 复位
当将该位为 0 时，定时器 0 复位后，T0RSTB 会自动置 1</td></tr>
<tr><td>1:0</td><td>T0SEL[1:0]</td><td>时钟源选择
<table>
<tr><td>T0SEL[1:0]</td><td>定时器 0 时钟源</td></tr>
<tr><td>00</td><td>TM0CLK</td></tr>
<tr><td>01</td><td>TM0CLK</td></tr>
<tr><td>10</td><td>外部 32768Hz 晶振时钟，
仅当外部接 32768Hz 晶振，且晶振打开时有效</td></tr>
<tr><td>11</td><td>内部 32K WDT 时钟，
仅当内部 WDT 晶振打开时有效</td></tr>
</table></td></tr>
</table>

表 9　TM0IN 寄存器各位功能表

位地址	标识符	功能
7 ： 0	TM0IN[7:0]	定时器 0 溢出值（溢出值：1~255）

表 10　TM0CNT 寄存器各位功能表

位地址	标识符	功能
7 ： 0	TM0CNT[7:0]	定时器 0 计数寄存器，**只读**

操作：

1) 设置 TM0CLK，为定时器 0 模块选择输入。
2) 设置 TM0IN，选择定时器 0 溢出值。（溢出值：1~255）
3) 设置寄存器标志位：TM0IE 与 GIE，使能定时器 0 中断。
4) 清零寄存器标志位：T0RSTB，复位定时器 0 模块的计数器。
5) 设置寄存器标志位：TM0EN，使能定时器 0 模块的 8 bits 计数器。
6) 当定时超时发生时，程序计数器会跳转到 004H。

定时器 0 溢出时间计算方法：

定时器 0 溢出时间=（TM0IN+1）/TM0CLK.

2.6 I/O PORT

表 11 I/O 口寄存器表

地址	名称	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	上电复位值
20h	PT1	PT1[7:0]								xxxxxxxx
21h	PT1EN	PT1EN[7:4]					PT1EN[2:0]			0000u000
22h	PT1PU	PT1PU[7:0]								00000000
23h	PT1CON	PT11OD	PT1W1[3:0]				E1M	E0M[1:0]		00000000
28h	PT3				PT3[4:0]					uuuxxxxx
29h	PT3EN				PT3EN[4:0]					uuu00000
2ah	PT3PU				PT3PU[4:0]					uuu00000
2bh	PT3CON				PT3CON[4:0]					uuu00000
30h	PT5				PT5[4:0]					uuuxxxxx
31h	PT5EN				PT5EN[4:0]					uuu00000
32h	PT5PU				PT5PU[4:0]					uuu00000
33h	PT5CON				PT5OD[4:0]					uuu00000
38h	PT5CON						PT1W2[2:0]			uuuuu000

微控制器中的通用 I/O 口（GPIO）用于通用的输入与输出功能。用户可以通过 GPIO 接收数据信号或将数据传送给其它的数字设备。CSU8RF3224 的部分 GPIO 可以被定义为其它的特殊功能。在本节，只说明 GPIO 的通用 I/O 口功能，特殊功能将会在接下来的章节中说明。

2.6.1 PT1 口

PT1 寄存器（地址为 20h）

特性	R/W-X	R/W-X	R/W-X	R/W-X	R/W-X	R/W-X	R/W-X	R/W-X
PT1	PT1[7:0]							
	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0

Bit 7-0 PT1[7:0]：GPIO1 口数据标志
PT1[7] = GPIO1 bit 7 数据标志位
PT1[6] = GPIO1 bit 6 数据标志位
PT1[5] = GPIO1 bit 5 数据标志位
PT1[4] = GPIO1 bit 4 数据标志位
PT1[3] = GPIO1 bit 3 数据标志位
PT1[2] = GPIO1 bit 2 数据标志位
PT1[1] = GPIO1 bit 1 数据标志位
PT1[0] = GPIO1 bit 0 数据标志位

PT1EN 寄存器（地址为 21h）

特性	R/W-0	R/W-0	R/W-0	R/W-0	U-0	R/W-0	R/W-0	R/W-0
PT1EN	PT1EN[7:4]					PT1EN[2:0]		
	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0

Bit 7-0 PT1EN[7:0]：GPIO1 口输入/输出控制标志

PT1EN[7] = GPIO1 bit 7 的 I/O 控制标志位；0 = 定义为输入口，1 = 定义为输出口
PT1EN[6] = GPIO1 bit 6 的 I/O 控制标志位；0 = 定义为输入口，1 = 定义为输出口
PT1EN[5] = GPIO1 bit 5 的 I/O 控制标志位；0 = 定义为输入口，1 = 定义为输出口
PT1EN[4] = GPIO1 bit 4 的 I/O 控制标志位；0 = 定义为输入口，1 = 定义为输出口
PT1EN[2] = GPIO1 bit 2 的 I/O 控制标志位；0 = 定义为输入口，1 = 定义为输出口
PT1EN[1] = GPIO1 bit 1 的 I/O 控制标志位；0 = 定义为输入口，1 = 定义为输出口
PT1EN[0] = GPIO1 bit 0 的 I/O 控制标志位；0 = 定义为输入口，1 = 定义为输出口

PT1PU 寄存器（地址为 22h）

特性	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
PT1PU	PT1PU[7:0]							
	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0

Bit 7-0 PT1PU[7:0]：GPIO1 口上拉电阻使能标志

PT1PU[7] = GPIO1 bit 7 控制标志位；0 = 断开上拉电阻，1 = 使用上拉电阻
PT1PU[6] = GPIO1 bit 6 控制标志位；0 = 断开上拉电阻，1 = 使用上拉电阻
PT1PU[5] = GPIO1 bit 5 控制标志位；0 = 断开上拉电阻，1 = 使用上拉电阻
PT1PU[4] = GPIO1 bit 4 控制标志位；0 = 断开上拉电阻，1 = 使用上拉电阻
PT1PU[3] = GPIO1 bit 3 控制标志位；0 = 断开上拉电阻，1 = 使用上拉电阻
PT1PU[2] = GPIO1 bit 2 控制标志位；0 = 断开上拉电阻，1 = 使用上拉电阻
PT1PU[1] = GPIO1 bit 1 控制标志位；0 = 断开上拉电阻，1 = 使用上拉电阻
PT1PU[0] = GPIO1 bit 0 控制标志位；0 = 断开上拉电阻，1 = 使用上拉电阻

特性（Property）：
R = 可读位　　W = 可写位　　U = 无效位
-n = 上电复位后的值　‘1’ = 位已设置　‘0’ = 位已清零　　X = 不确定位

PT1CON0 寄存器（地址为 23h）

特性	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
PT1CON0	PT11OD	PT1W[3:0]				E1M	E0M[1:0]	
	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0

Bit 7 PT11OD：PT1.1 漏极开路使能位
0 = 禁止 PT1.1 漏极开路
1 = 使能 PT1.1 漏极开路
Bit 6 PT1W[3]:PT1.5 外部中断 1 使能
0 = 禁止 PT1.5 外部中断 1
1 = 使能 PT1.5 外部中断 1
Bit 5 PT1W[2]:PT1.4 外部中断 1 使能
0 = 禁止 PT1.4 外部中断 1
1 = 使能 PT1.4 外部中断 1
Bit 4 PT1W[1]:PT1.3 外部中断 1 使能
0 = 禁止 PT1.3 外部中断 1
1 = 使能 PT1.3 外部中断 1
Bit 3 PT1W[0]:PT1.1 外部中断 1 使能
0 = 禁止 PT1.1 外部中断 1
1 = 使能 PT1.1 外部中断 1
Bit 2 E1M：外部中断 1 触发模式
1 = 外部中断 1 为下降沿触发
0 = 外部中断 1 在状态改变时触发
Bit 1-0 E0M[1:0]：外部中断 0 触发模式
11 = 外部中断 0 在状态改变时触发
10 = 外部中断 0 在状态改变时触发
01 = 外部中断 0 为上升沿触发
00 = 外部中断 0 为下降沿触发

PT1CON1 寄存器（地址为 38h）

特性	U-0	U-0	U-0	U-0	U-0	R/W-0	R/W-0	R/W-0
PT1CON1						PT1W2[2:0]		
	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0

Bit 2 PT1W2[2]:PT1.7 外部中断 1 使能
0 = 禁止 PT1.7 外部中断 1
1 = 使能 PT1.7 外部中断 1
Bit 1 PT1W2[1]:PT1.6 外部中断 1 使能
0 = 禁止 PT1.6 外部中断 1
1 = 使能 PT1.6 外部中断 1
Bit 0 PT1W2[0]:PT1.2 外部中断 1 使能
0 = 禁止 PT1.2 外部中断 1
1 = 使能 PT1.2 外部中断 1

特性（Property）：
R = 可读位 W = 可写位 U = 无效位
-n = 上电复位后的值 ‘1’ = 位已设置 ‘0’ = 位已清零 X = 不确定位

2.6.2 PT3 口

PT3 寄存器（地址为 28h）

特性	U-0	U-0	U-0	R/W-X	R/W-X	R/W-X	R/W-X	R/W-X
PT3				PT3[4:0]				
	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0

Bit 4-0 PT3[4:0]：GPIO3 口数据标志位
PT3[4] = GPIO3 bit 4 的数据标志位
PT3[3] = GPIO3 bit 3 的数据标志位
PT3[2] = GPIO3 bit 2 的数据标志位
PT3[1] = GPIO3 bit 1 的数据标志位
PT3[0] = GPIO3 bit 0 的数据标志位

PT3EN 寄存器（地址为 29h）

特性	U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
PT3EN				PT3EN[4:0]				
	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0

Bit 4-0 PT3EN[4:0]：GPIO3 口输入/输出控制标志
PT3EN[4] = GPIO3 bit 4 的 I/O 控制标志位；0 = 定义为输入口，1 = 定义为输出口
PT3EN[3] = GPIO3 bit 3 的 I/O 控制标志位；0 = 定义为输入口，1 = 定义为输出口
PT3EN[2] = GPIO3 bit 2 的 I/O 控制标志位；0 = 定义为输入口，1 = 定义为输出口
PT3EN[1] = GPIO3 bit 1 的 I/O 控制标志位；0 = 定义为输入口，1 = 定义为输出口
PT3EN[0] = GPIO3 bit 0 的 I/O 控制标志位；0 = 定义为输入口，1 = 定义为输出口

PT3PU 寄存器（地址为 2ah）

特性	U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
PT3PU				PT3PU[4:0]				
	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0

Bit 4-0 PT3PU[4:0]：GPIO3 口上拉电阻使能标志
PT3PU[4] = GPIO3 bit 4 控制标志位；0 = 断开上拉电阻，1 = 使用上拉电阻
PT3PU[3] = GPIO3 bit 3 控制标志位；0 = 断开上拉电阻，1 = 使用上拉电阻
PT3PU[2] = GPIO3 bit 2 控制标志位；0 = 断开上拉电阻，1 = 使用上拉电阻
PT3PU[1] = GPIO3 bit 1 控制标志位；0 = 断开上拉电阻，1 = 使用上拉电阻
PT3PU[0] = GPIO3 bit 0 控制标志位；0 = 断开上拉电阻，1 = 使用上拉电阻

特性（Property）：
R = 可读位　　W = 可写位　　U = 无效位
-n = 上电复位后的值　‘1’ = 位已设置　‘0’ = 位已清零　　X = 不确定位

PT3CON 寄存器（地址为 2bh）

特性	U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
PT3CON				PT3CON[4:0]				
	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0

Bit 4-0 PT3CON[4:0]：GPIO3 口模拟/数字端口使能标志

PT3CON[4] = GPIO3bit 4 的 I/O 控制标志位；0 = 定义为数字口，1 = 定义为模拟口
PT3CON[3] = GPIO3bit 3 的 I/O 控制标志位；0 = 定义为数字口，1 = 定义为模拟口
PT3CON[2] = GPIO3bit 2 的 I/O 控制标志位；0 = 定义为数字口，1 = 定义为模拟口
PT3CON[1] = GPIO3bit 1 的 I/O 控制标志位；0 = 定义为数字口，1 = 定义为模拟口
PT3CON[0] = GPIO3bit 0 的 I/O 控制标志位；0 = 定义为数字口，1 = 定义为模拟口

2.6.3 PT5 口

PT5 寄存器（地址为 30h）

特性	U-0	U-0	U-0	R/W-X	R/W-X	R/W-X	R/W-X	R/W-X
PT5				PT5[4:0]				
	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0

Bit 4-0 PT5[4:0]：GPIO5 口数据标志位

PT5[4] = GPIO5 bit 4 的数据标志位
PT5[3] = GPIO5 bit 3 的数据标志位
PT5[2] = GPIO5 bit 2 的数据标志位
PT5[1] = GPIO5 bit 1 的数据标志位
PT5[0] = GPIO5 bit 0 的数据标志位

PT5EN 寄存器（地址为 31h）

特性	U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
PT5EN				PT5EN[4:0]				
	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0

Bit 4-0 PT5EN[4:0]：GPIO5 口输入/输出控制标志

PT5EN[4] = GPIO5 bit 4 的 I/O 控制标志位；0 = 定义为输入口，1 = 定义为输出口
PT5EN[3] = GPIO5 bit 3 的 I/O 控制标志位；0 = 定义为输入口，1 = 定义为输出口
PT5EN[2] = GPIO5 bit 2 的 I/O 控制标志位；0 = 定义为输入口，1 = 定义为输出口
PT5EN[1] = GPIO5 bit 1 的 I/O 控制标志位；0 = 定义为输入口，1 = 定义为输出口
PT5EN[0] = GPIO5 bit 0 的 I/O 控制标志位；0 = 定义为输入口，1 = 定义为输出口

特性（Property）：
R = 可读位　　W = 可写位　　U = 无效位
-n = 上电复位后的值　‘1’ = 位已设置　‘0’ = 位已清零　　X = 不确定位

PT5PU 寄存器（地址为 32h）

特性	U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
PT5PU				PT5PU[4:0]				
	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0

Bit 4-0 PT5PU[4:0]：GPIO5 口上拉电阻使能标志

PT5PU[4] = GPIO5 bit 4 控制标志位；0 = 断开上拉电阻，1 = 使用上拉电阻
PT5PU[3] = GPIO5 bit 3 控制标志位；0 = 断开上拉电阻，1 = 使用上拉电阻
PT5PU[2] = GPIO5 bit 2 控制标志位；0 = 断开上拉电阻，1 = 使用上拉电阻
PT5PU[1] = GPIO5 bit 1 控制标志位；0 = 断开上拉电阻，1 = 使用上拉电阻
PT5PU[0] = GPIO5 bit 0 控制标志位；0 = 断开上拉电阻，1 = 使用上拉电阻

PT5CON 寄存器（地址为 33h）

特性	U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
PT5CON				PT5OD[4:0]				
	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0

Bit 4-0 PT5CON[4:0]：GPIO5 口控制标志

PT5CON[4] = GPIO5 bit 4 控制标志位；0 = 禁止开漏输出，1 = 使能开漏输出
PT5CON[3] = GPIO5 bit 3 控制标志位；0 = 禁止开漏输出，1 = 使能开漏输出
PT5CON[2] = GPIO5 bit 2 控制标志位；0 = 禁止开漏输出，1 = 使能开漏输出
PT5CON[1] = GPIO5 bit 1 控制标志位；0 = 禁止开漏输出，1 = 使能开漏输出
PT5CON[0] = GPIO5 bit 0 控制标志位；0 = 禁止开漏输出，1 = 使能开漏输出

特性（Property）：
R = 可读位　　W = 可写位　　U = 无效位
-n = 上电复位后的值　‘1’ = 位已设置　‘0’ = 位已清零　　X = 不确定位

3 增强功能

3.1 Halt和Sleep模式

CSU8RF3224 支持低功耗工作模式。为了使其处于待机状态， 可以让 CPU 停止工作使 CSU8RF3224 进行停止或睡眠模式，减低功耗。这两种模式描述如下：

停止模式

CPU 执行停止指令后，程序计数器停止计数直到出现中断指令。为了避免由中断返回（Interrupt Return）引起的程序错误，建议在停止指令之后加一 NOP 指令以保证程序返回时能正常运行。

睡眠模式

CPU 执行睡眠指令后，所有的振荡器停止工作(EO_SLP 为 0 时)直到出现一个外部中断指令复位 CPU。为了避免由中断返回（Interrupt Return）引起的程序错误，建议停止指令之后加一 NOP 指令以保证程序的正常运行。在睡眠模式下的功耗大约有 1uA。

为了保证 CPU 在睡眠模式下的功耗最小，在执行睡眠指令之前，需要把 IO 口的上拉电阻断开，并且保证所有的输入口是接到 VDD 或 VSS 电平。

注：
芯片如果处于 sleep 状态，这时候降低电压，配置 2.4V 低电压复位不会起作用，低于 2.0V 掉电复位点才会复位。如果 sleep 唤醒后，此时还处于低电压复位点以下，则会立即复位。

Halt 示范程序：

```
…
movlw 01h
movwf pt1up    ;断开 pt1 除 bit0(pt1[0])外的其他接口的上拉电阻
movlw feh
movwf pt1en    ;pt1 口除 bit0(pt1[0])做输入口外，其他接口作为输出口(pt1.3 除外)
clrf pt1       ;将 pt1[4:1]输出为低
clrf pt3up     ;断开 pt3 上拉电阻
clrf pt3en     ;pt3 口用作输入口
clrf pt3con    ;pt3 口用作数字口
clrf pt3       ;将 pt3 输出为低
clrf pt5up     ;断开 pt5 上拉电阻
clrf pt5en     ;pt5 口用作输入口
clrf pt5       ;将 pt5 输出为低
clrf intf      ;清除中断标志位
movlw 81h
movwf inte     ;使能外部中断 0
halt           ;进入停止模式
nop            ; 保证 CPU 重启后程序能正常工作
…
```

Sleep 示范程序:

```
…
movlw 01h
movwf pt1up    ;断开 pt1 除 bit0(pt1[0])外的其他接口的上拉电阻
movlw feh
movwf pt1en    ;pt1 口除 bit0(pt1[0])做输入口外，其他接口作为输出口(pt1.3 除外)
clrf pt1       ; 将 pt1[4:1]输出为低
clrf pt3up     ;断开 pt3 上拉电阻
clrf pt3en     ;pt3 口用作输入口
clrf pt3con    ;pt3 口用作数字口
clrf pt3       ;将 pt3 输出为低
clrf pt5up     ;断开 pt5 上拉电阻
clrf pt5en     ;pt5 口用作输入口
clrf pt5       ;将 pt5 输出为低
clrf intf      ;清除中断标志位
movlw 81h
movwf inte     ;使能外部中断 0
sleep          ;进入睡眠模式
nop            ; 保证 CPU 重启后程序能正常工作
…
```

3.2 看门狗(WDT)

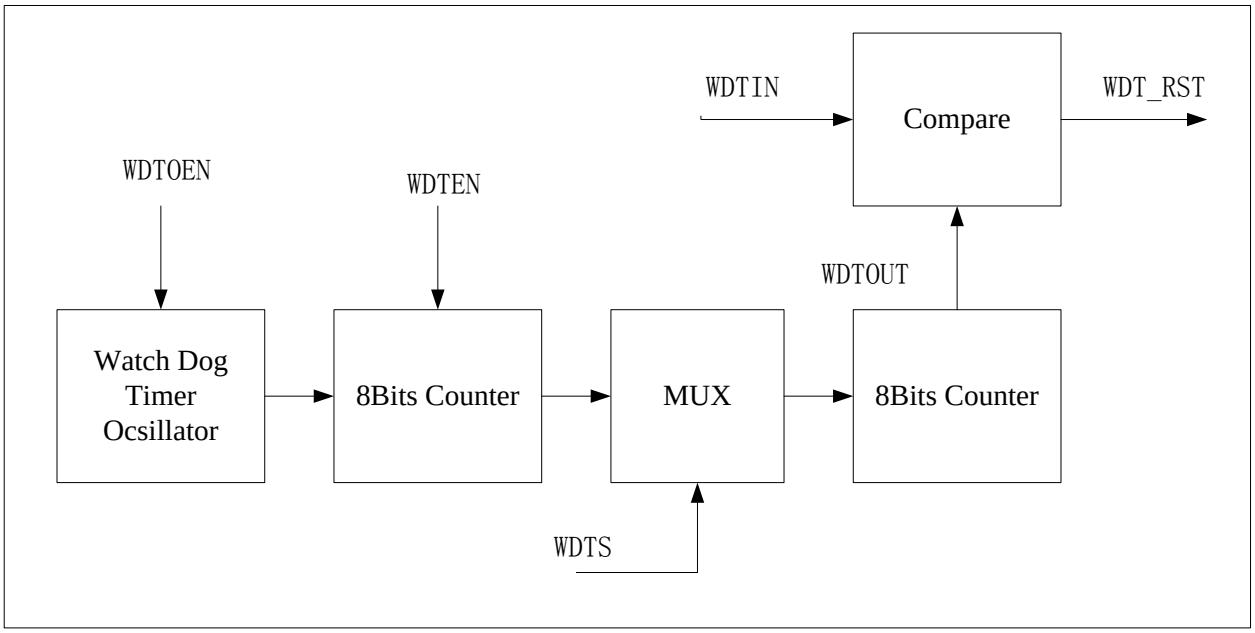


图13　看门狗定时器功能框图

看门狗定时器（WDT）用于防止程序由于某些不确定因素而失去控制。当 WDT 启动时，WDT 计时超时后将使 CPU 复位。在运行的程序一般在 WDT 复位 CPU 之前先复位 WDT。当出现某些故障时，程序会被 WDT 复位到正常状态下，但程序不会复位 WDT。

当用户把 CST_WDT 清 0 时，则内部的看门狗定时器振荡器（32KHz）将会启动，产生的时钟被送到“8 bits 计数器 1”。当用户置位 WDTEN 时，“8 bits 计数器 1”开始计数，“8 bits 计数器 1”的输出是内部信号 WDTA[7:0]，被发送到一个受寄存器标志位 WDTS[2:0]控制的多路选择器，选择器的输出作为“8 bits 计数器 2”的时钟输入。当“8 bits 计数器 2”计数值与 WDTIN 数值相等时溢出，溢出时它会发送 WDTOUT 信号复位 CPU 及置位 TO 标志位。用户可以使用指令 CLRWDT 复位 WDT。

表 12　看门狗定时器寄存器表

地址	名称	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	上电复位值
04H	STATUS					TO				00u00000
0DH	WDTCON	WDTEN					WDTS[2:0]			0uuuu000
0EH	WDTIN	WDTIN[7:0]								11111111

操作：

1. 设置 WDTS[3:0]，选择 WDT 时钟频率。
2. 设置 WDTIN，选择不同的溢出时间值
2. 置位寄存器标志位：WDTEN，使能 WDT。
3. 把 CST_WDT 清 0，打开 WDT 的晶振。
4. 在程序中执行 CLRWDT 指令复位 WDT。

WDT 溢出时间计算公式：

$$溢出时间=\frac{2^{(8-WDTS[2:0])}}{32k}*(WDTIN[7:0]+1)$$

WDTS[2:0]范围为 0~7，WDTIN[7:0]范围为 0~255。

WDTS[2:0]	计数器时钟	时间（当 WDTIN==FFH）
000	WDTA [0]	2048ms
001	WDTA [1]	1024ms
010	WDTA [2]	512ms
011	WDTA [3]	256ms
100	WDTA [4]	128ms
101	WDTA [5]	64ms
110	WDTA [6]	32ms
111	WDTA [7]	16ms

3.3 定时/计数器 2

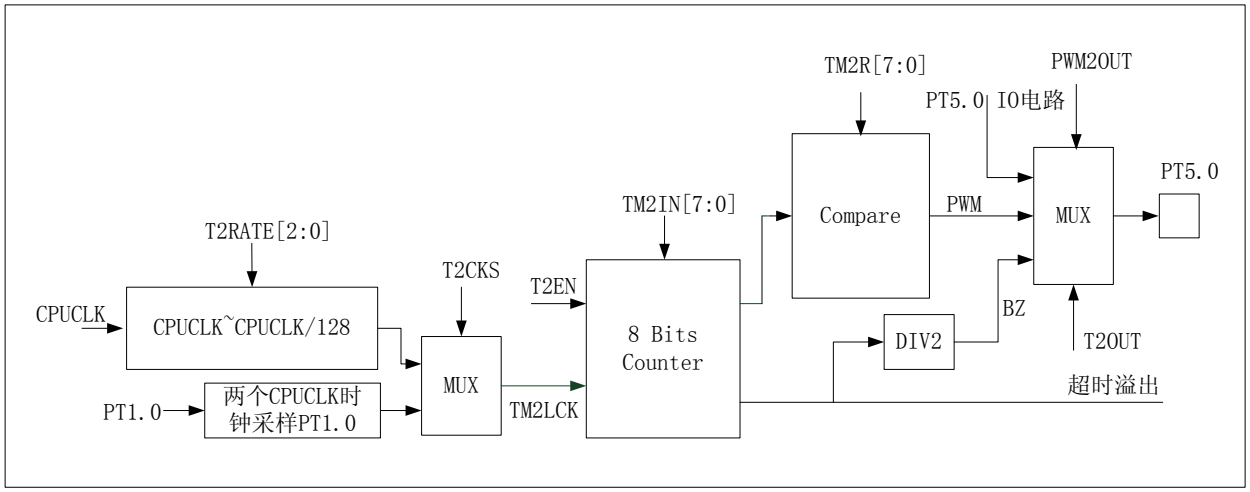


图14　定时/计数器 2 模块的功能框图

定时/计数器 2 模块的输入是 TM2CLK。当用户设置了定时/计数器 2 模块的使能标志，8 bits 计数器将启动，从 00h 递增到 TM2IN。用户需要设置 TM2IN（定时器模块中断信号选择器）以选择定时超时中断信号。当定时超时发生时，BZ 输出信号发生跳变。

主要功能：
1） 8 位可编程定时器；
2） 外部事件计数；
3） 蜂鸣器输出；
4） PWM2 输出；

3.3.1 寄存器描述

表 13　定时器寄存器列表

地址	名称	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	上电复位值
06h	INTF		TM2IF							u0u00u00
07h	INTE	GIE	TM2IE							00u00u00
17h	TM2CON	T2EN	T2RATE[2:0]			T2CKS	T2RSTB	T2OUT	PWM2OUT	00000100
18h	TM2IN	TM2IN[7:0]								11111111
19h	TM2CNT	TM2CNT[7:0]								00000000
1ah	TM2R	TM2R[7:0]								00000000

表 14　TM2CON 寄存器各位功能表

<table>
<tr><th>位地址</th><th>标识符</th><th>功能</th></tr>
<tr><td>7</td><td>T2EN</td><td>定时/计数器 2 使能位
1：使能定时器 2
0：禁止定时器 2</td></tr>
<tr><td>6:4</td><td>T2RATE[2:0]</td><td>定时/计数器 1 时钟选择
<table>
<tr><td>T2RATE [2:0]</td><td>TM2CLK</td></tr>
<tr><td>000</td><td>CPUCLK</td></tr>
<tr><td>001</td><td>CPUCLK /2</td></tr>
<tr><td>010</td><td>CPUCLK /4</td></tr>
<tr><td>011</td><td>CPUCLK /8</td></tr>
<tr><td>100</td><td>CPUCLK /16</td></tr>
<tr><td>101</td><td>CPUCLK /32</td></tr>
<tr><td>110</td><td>CPUCLK /64</td></tr>
<tr><td>111</td><td>CPUCLK /128</td></tr>
</table></td></tr>
<tr><td>3</td><td>T2CKS</td><td>定时/计数器 2 时钟源选择位
1：PT1.0 作为时钟
0：CPUCLK 的分频时钟</td></tr>
<tr><td>2</td><td>T2RSTB</td><td>定时/计数器 2 复位
1：禁止定时/计数器 2 复位
0：使能定时/计数器 2 复位
当将该位为 0 时，定时器 2 复位后，T2RSTB 会自动置 1</td></tr>
<tr><td>1</td><td>T2OUT</td><td rowspan="2">PT5.0 口输出控制
<table>
<tr><td>T2OUT</td><td>PWM2OUT</td><td>PT5.0 输出控制，仅当 PT5.0 配置为输出有效</td></tr>
<tr><td>0</td><td>0</td><td>IO 输出</td></tr>
<tr><td>0</td><td>1</td><td>PWM2 输出</td></tr>
<tr><td>1</td><td>0</td><td>蜂鸣器输出</td></tr>
<tr><td>1</td><td>1</td><td>PWM2 输出</td></tr>
</table></td></tr>
<tr><td>0</td><td>PWM2OUT</td></tr>
</table>

表 15　TM2IN 寄存器各位功能表

位地址	标识符	功能
7 ： 0	TM2IN[7:0]	定时/计数器溢出值

表 16　TM2CNT 寄存器各位功能表

位地址	标识符	功能
7 ： 0	TM2CNT[7:0]	定时/计数器 2 计数寄存器，**只读**

表 17　TM2R 寄存器各位功能表

位地址	标识符	功能
7 ： 0	TM2R[7:0]	定时/计数器 2 的 PWM 高电平占空比控制寄存器

操作：

1) 设置 TM2CLK，为定时器模块选择输入。
2) 设置 TM2IN，选择定时器溢出值。
3) 设置寄存器标志位：TM2IE 与 GIE，使能定时器中断。
4) 清零寄存器标志位：T2RSTB，复位定时器模块的计数器。
5) 设置寄存器标志位：T2EN，使能定时器模块的 8 bits 计数器。
6) 当定时超时发生时，BZ 输出信号发生跳变，可作为蜂鸣器输出；程序计数器会跳转到 004H。

定时器 2 溢出时间计算方法：

定时器 2 溢出时间=（TM2IN+1）/TM2CLK. （TM2IN 不为 0）

3.3.2 蜂鸣器

操作：

1) 把 PT5.0 配置为输出口。
2) 设置 TM2CLK，为定时器模块选择输入。
3) 设置 TM2IN，选择定时器溢出值。
4) 清零寄存器标志位：T2RSTB，复位定时器模块的计数器。
5) 设置寄存器标志位：T2EN，使能定时器模块的 8 bits 计数器。
6) 当定时超时发生时，BZ 输出信号发生跳变，可作为蜂鸣器输出。

蜂鸣器周期计算方法：

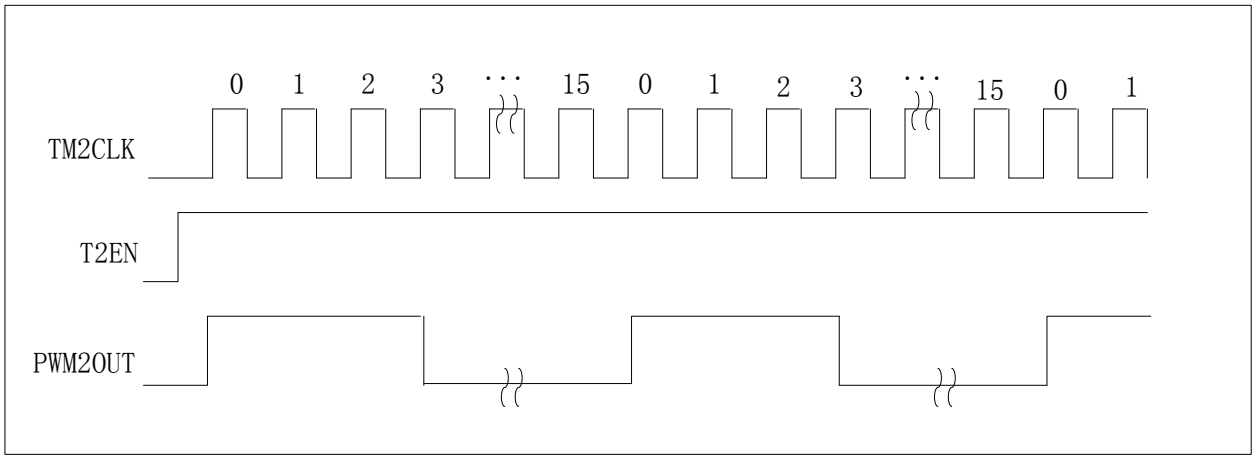
蜂鸣器周期=（TM2IN+1）*2/TM2CLK. （TM2IN 不为 0）

3.3.3 PWM

操作：

1) 把 PT5.0 配置为输出口。
2) 设置 TM2CLK，为定时/计数器 2 模块选择输入。
3) 设置 TM2IN 来配置 PWM2 的周期。
4) 设置 TM2R 来配置 PWM2 的高电平的的脉宽。
5) 使能 PWM2OUT 输出，配置 PT5.0 为输出端口，之后把 T2EN 置 1 启动定时器。
6) PWM 从 PT5.0 输出。

周期为 TM2IN+1，高电平脉宽为 TM2R。如 TM2IN=0x0F，TM2R=0x03 的 PWM2 波形输出如下：



3.4 定时/计数器 3

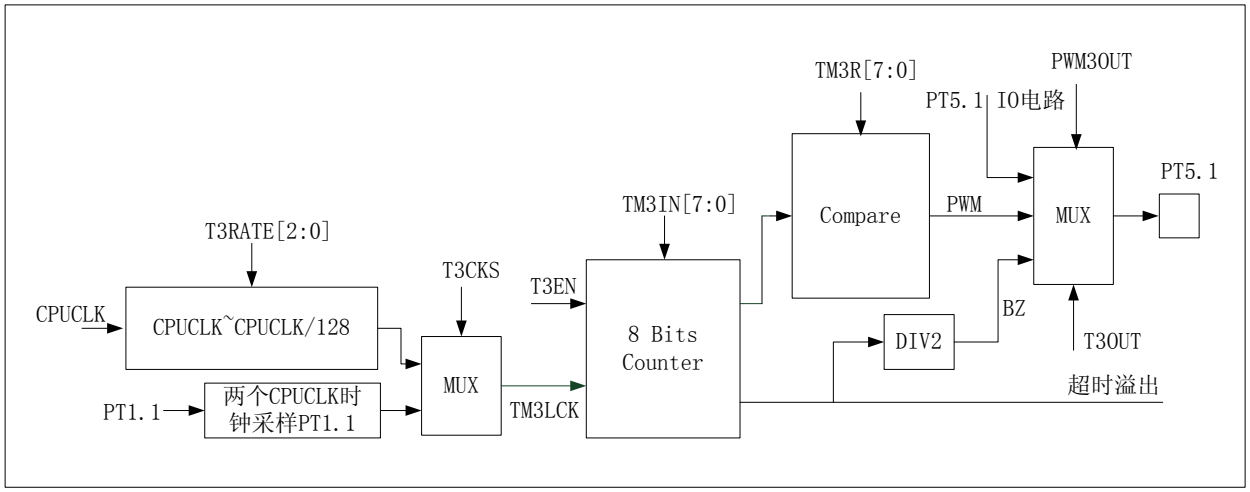


图15　定时/计数器 3 模块的功能框图

定时/计数器 3 模块的输入是 TM3CLK。当用户设置了定时/计数器 3 模块的使能标志，8 bits 计数器将启动，从 00h 递增到 TM3IN。用户需要设置 TM3IN（定时器模块中断信号选择器）以选择定时超时中断信号。当定时超时发生时，BZ 输出信号发生跳变。

主要功能：
1） 8 位可编程定时器；
2） 外部事件计数；
3） 蜂鸣器输出；
4） PWM 输出；

3.4.1 寄存器描述

表 18　定时器寄存器列表

地址	名称	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	上电复位值
3ch	INTF2				TM3IF					uuu0uuuu
3dh	INTE2				TM3IE					uuu0uuuu
1bh	TM3CON	T3EN	T3RATE[2:0]			T3CKS	T3RSTB	T3OUT	PWM3OUT	00000100
1ch	TM3IN	TM3IN[7:0]								11111111
1dh	TM3CNT	TM3CNT[7:0]								00000000
1eh	TM3R	TM3R[7:0]								00000000

表 19　TM3CON 寄存器各位功能表

<table>
<tr><th>位地址</th><th>标识符</th><th>功能</th></tr>
<tr><td>7</td><td>T3EN</td><td>定时/计数器 3 使能位
1：使能定时器 3
0：禁止定时器 3</td></tr>
<tr><td>6:4</td><td>T3RATE[2:0]</td><td>定时/计数器 3 时钟选择
<table>
<tr><td>T3RATE [2:0]</td><td>TM3CLK</td></tr>
<tr><td>000</td><td>CPUCLK</td></tr>
<tr><td>001</td><td>CPUCLK /2</td></tr>
<tr><td>010</td><td>CPUCLK /4</td></tr>
<tr><td>011</td><td>CPUCLK /8</td></tr>
<tr><td>100</td><td>CPUCLK /16</td></tr>
<tr><td>101</td><td>CPUCLK /32</td></tr>
<tr><td>110</td><td>CPUCLK /64</td></tr>
<tr><td>111</td><td>CPUCLK /128</td></tr>
</table></td></tr>
<tr><td>3</td><td>T3CKS</td><td>定时/计数器 3 时钟源选择位
1：PT1.1 作为时钟
0：CPUCLK 的分频时钟</td></tr>
<tr><td>2</td><td>T3RSTB</td><td>定时/计数器 3 复位
1：禁止定时/计数器 3 复位
0：使能定时/计数器 3 复位
当将该位为 0 时，定时器 3 复位后，T3RSBT 会自动置 1</td></tr>
<tr><td>1</td><td>T3OUT</td><td rowspan="2">PT5.1 口输出控制
<table>
<tr><td>T3OUT</td><td>PWM3OUT</td><td>PT5.1 输出控制，仅当 PT5.1 配置为输出有效</td></tr>
<tr><td>0</td><td>0</td><td>IO 输出</td></tr>
<tr><td>0</td><td>1</td><td>PWM3 输出</td></tr>
<tr><td>1</td><td>0</td><td>蜂鸣器输出</td></tr>
<tr><td>1</td><td>1</td><td>PWM3 输出</td></tr>
</table></td></tr>
<tr><td>0</td><td>PWM3OUT</td></tr>
</table>

表 20　TM3IN 寄存器各位功能表

位地址	标识符	功能
7 ： 0	TM3IN[7:0]	定时/计数器溢出值

表 21　TM3CNT 寄存器各位功能表

位地址	标识符	功能
7 ： 0	TM3CNT[7:0]	定时/计数器 3 计数寄存器，**只读**

表 22　TM3R 寄存器各位功能表

位地址	标识符	功能
7 ： 0	TM3R[7:0]	定时/计数器 3 的 PWM 高电平占空比控制寄存器

操作：

1) 设置 TM3CLK，为定时器模块选择输入。
2) 设置 TM3IN，选择定时器溢出值。
3) 设置寄存器标志位：TM3IE 与 GIE，使能定时器中断。
4) 清零寄存器标志位：T3RSTB，复位定时器模块的计数器。
5) 设置寄存器标志位：T3EN，使能定时器模块的 8 bits 计数器。
6) 当定时超时发生时，BZ 输出信号发生跳变，可作为蜂鸣器输出；程序计数器会跳转到 004H。

定时器 3 溢出时间计算方法：

定时器 3 溢出时间=（TM3IN+1）/TM3CLK. （TM3IN 不为 0）

3.4.2 蜂鸣器

操作：

1) 把 PT5.1 配置为输出口。
2) 设置 TM3CLK，为定时器模块选择输入。
3) 设置 TM3IN，选择定时器溢出值。
4) 清零寄存器标志位：T3RSTB，复位定时器模块的计数器。
5) 设置寄存器标志位：T3EN，使能定时器模块的 8 bits 计数器。
6) 当定时超时发生时，BZ 输出信号发生跳变，可作为蜂鸣器输出

蜂鸣器周期计算方法：

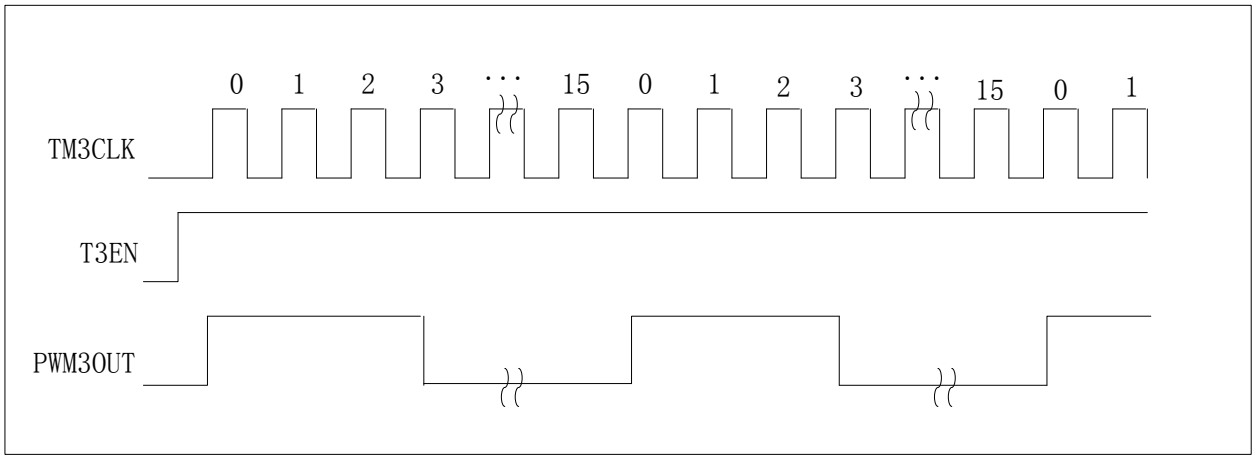
蜂鸣器周期=（TM3IN+1）*2/TM3CLK. （TM3IN 不为 0）

3.4.3 PWM

操作：

1) 把 PT5.1 配置为输出口。
2) 设置 TM3CLK，为定时/计数器 3 模块选择输入。
3) 设置 TM3IN 来配置 PWM3 的周期。
4) 设置 TM3R 来配置 PWM3 的高电平的的脉宽。
5) 使能 PWM3OUT 输出，配置 PT5.1 为输出端口，之后把 T3EN 置 1 启动定时器。
6) PWM3 从 PT5.1 输出。

周期为 TM3IN+1，高电平脉宽为 TM3R。如 TM3IN=0x0F，TM3R=0x03 的 PWM3 波形输出如下：



3.5 模数转换器（ADC）

CSU8RF3224 模数转换模块共用 5 条外部通道（AIN0~AIN4）和 3 条特殊通道(AIN5：内部 1/8VDD；AIN6：内部参考电压；AIN7：GND)，可以将模拟信号转换成 12 位数字信号。进行 AD 转换时，首先要选择输入通道(AIN0~AIN5)，然后把 SRADEN 置 1 使能 ADC，之后把 SRADS 置 1，启动 AD 转换。转换结束后，系统自动将 SRADS 清 0，并将转换结果存入寄存器 SRADL 和 SRADH 中。

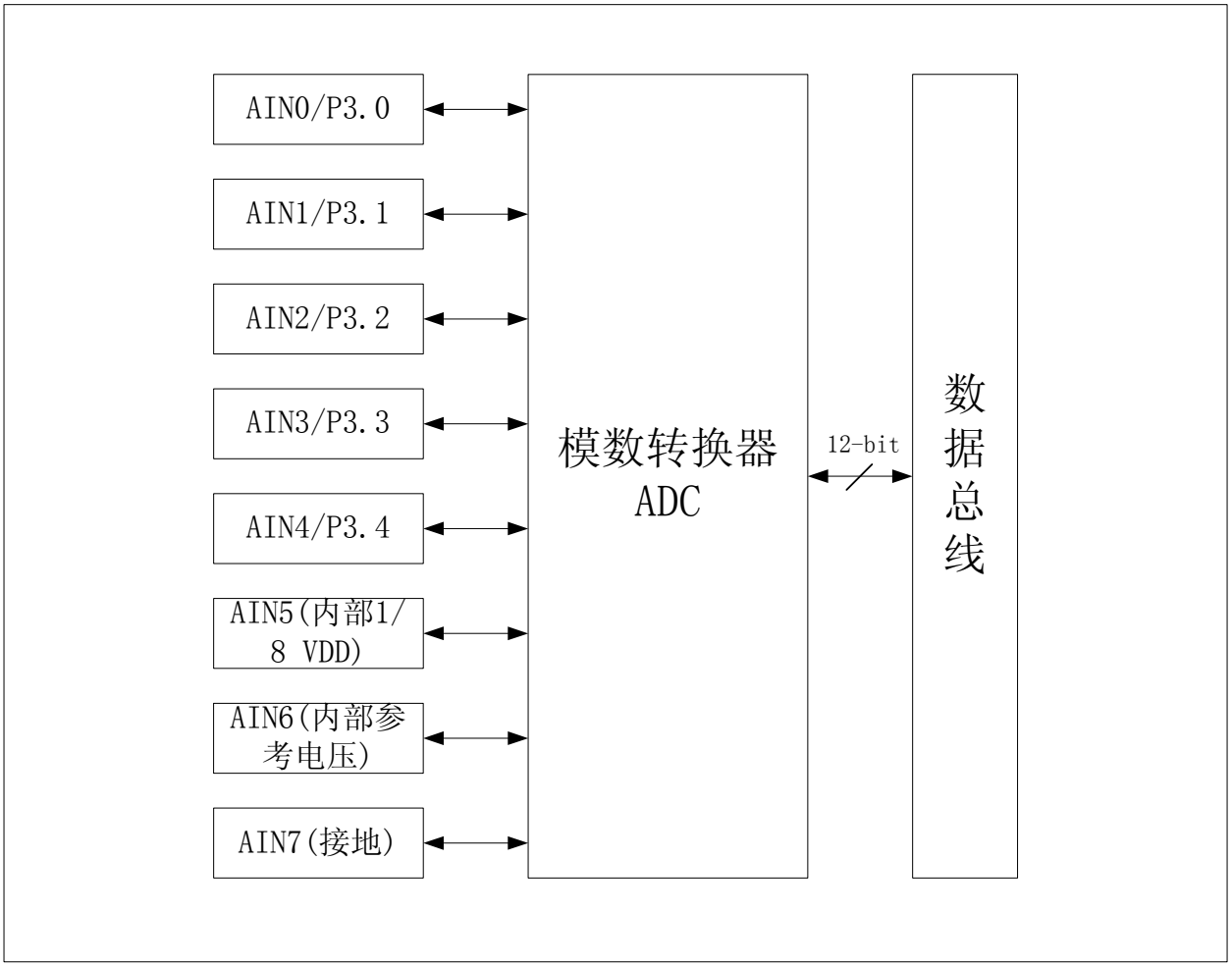


图16　模数转换器 ADC 功能框图

3.5.1 寄存器描述

表 23　ADC 寄存器列表

地址	名称	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	上电复位值
06h	INTF					ADIF				u0u00u00
07h	INTE	GIE				ADIE				00u00u00
50h	SRADCON0			SRADACKS[1:0]				SRADCKS[1:0]		uu00uu00
51h	SRADCON1	SRADEN	SRADS	OFTEN	CALIF	ENOV	OFFEX	VREFS[1:0]		00000000
52h	SRADCON2	CHS[3:0]								0000uuuu
54h	SRADL	SRAD[7:0]								00000000
55h	SRADH					SRAD[11:8]				uuuu0000
56h	SROFTL	SROFT[7:0]								00000000
57h	SROFTH					SROFT[11:8]				uuuu0000

表 24　SRADCON0 寄存器各位功能表

<table>
<tr><th>位地址</th><th>标识符</th><th>功能</th></tr>
<tr><td>5：4</td><td>SRADACKS[1:0]</td><td>ADC 输入信号获取时间
<table>
<tr><td>SRADACKS[1:0]</td><td>ADC 输入信号获取时间</td></tr>
<tr><td>00</td><td>16 个 ADC 时钟</td></tr>
<tr><td>01</td><td>8 个 ADC 时钟</td></tr>
<tr><td>10</td><td>4 个 ADC 时钟</td></tr>
<tr><td>11</td><td>2 个 ADC 时钟</td></tr>
</table></td></tr>
<tr><td>1：0</td><td>SRADCKS[1:0]</td><td>ADC 时钟
<table>
<tr><td>SRADCKS[1:0]</td><td>ADC 采样时钟</td></tr>
<tr><td>00</td><td>CPUCLK</td></tr>
<tr><td>01</td><td>CPUCLK/2</td></tr>
<tr><td>10</td><td>CPUCLK/4</td></tr>
<tr><td>11</td><td>CPUCLK/8</td></tr>
</table></td></tr>
</table>

表 25　SRADCON1 寄存器各位功能表

<table>
<tr><th>位地址</th><th>标识符</th><th>功能</th></tr>
<tr><td>7</td><td>SRADEN</td><td>ADC 使能位
1：使能
0：禁止</td></tr>
<tr><td>6</td><td>SRADS</td><td>ADC 启动位/状态控制位
1：开始，转换过程中
0：停止，转换结束
当置位后，启动 ADC 转换，转换完成会自动清 0</td></tr>
<tr><td>5</td><td>OFTEN</td><td>转换结果选择控制位
1：转换结果放在 SROFT 寄存器中
0：转换结果放在 SRAD 寄存器中</td></tr>
<tr><td>4</td><td>CALIF</td><td>校正控制位(OFTEN 为 0 时有效)
1：使能校正，即 AD 转换的结果是减去了 SROFT 失调电压值
0：禁止校正，即 AD 转换结果是没有减去 SROFT 失调电压值</td></tr>
<tr><td>3</td><td>ENOV</td><td>使能比较器溢出模式(CALIF 为 1 时有效)
1：使能，上溢或下溢直接是减去后的结果
0：禁止，下溢为 000h，上溢为 fffh</td></tr>
<tr><td>2</td><td>OFFEX</td><td>OFFSET 交换
1：比较器两端信号交换
0：比较器两端信号不交换（正端为信号，负端为参考电压）</td></tr>
<tr><td>1:0</td><td>VREFS[1:0]</td><td>ADC 参考电源选择
注：不同参考电压切换，建议延迟 10uS 再做 AD 转换
<table>
<tr><td>VREFS[1:0]</td><td>AD 参考电压</td></tr>
<tr><td>00</td><td>VDD</td></tr>
<tr><td>01</td><td>PT3.0 外部参考电源输入</td></tr>
<tr><td>10</td><td>内部参考电压 1.40V</td></tr>
<tr><td>11</td><td>内部参考电压 1.40V，PT3.0 做为内部参考电压输出端，可外接电容作为内置参考电压滤波使用，以提高精度。</td></tr>
</table></td></tr>
</table>

表 26　SRADCON2 寄存器各位功能表

<table>
<tr><th>位地址</th><th>标识符</th><th>功能</th></tr>
<tr><td>7：4</td><td>CHS[3:0]</td><td>ADC 输入通道选择位
<table>
<tr><td>CHS[3:0]</td><td>输入通道</td></tr>
<tr><td>0000</td><td>AIN0 输入</td></tr>
<tr><td>0001</td><td>AIN1 输入</td></tr>
<tr><td>0010</td><td>AIN2 输入</td></tr>
<tr><td>0011</td><td>AIN3 输入</td></tr>
<tr><td>0100</td><td>AIN4 输入</td></tr>
<tr><td>0101</td><td>AIN5 输入，内部 1/8VDD</td></tr>
<tr><td>0110</td><td>AIN6 输入，内部参考电压</td></tr>
<tr><td>0111</td><td>AIN7 输入，内部接地</td></tr>
<tr><td>其它</td><td>保留</td></tr>
</table></td></tr>
</table>

表 27　SRADL 寄存器各位功能表

位地址	标识符	功能
7：0	SRAD[7:0]	ADC 数据的低 8 位，只可读

表 28　SRADH 寄存器各位功能表

位地址	标识符	功能
3：0	SRAD[11:8]	ADC 数据的高 4 位，只可读

表 29　SROFTL 寄存器各位功能表

位地址	标识符	功能
7：0	SROFT[7:0]	校正值数据的低 8 位

表 30　SROFTH 寄存器各位功能表

位地址	标识符	功能
3：0	SROFT[11:8]	校正值数据的高 4 位

表 31　输入电压和 SRAD 输出数据的关系

输入电压	SRAD[11:0]											
	11	10	9	8	7	6	5	4	3	2	1	0
0/4096*VREF	0	0	0	0	0	0	0	0	0	0	0	0
1/4096*VREF	0	0	0	0	0	0	0	0	0	0	0	1
…												
…												
4094/4096*VREF	1	1	1	1	1	1	1	1	1	1	1	0
4095/4096*VREF	1	1	1	1	1	1	1	1	1	1	1	1

3.5.2 转换时间

12 位 AD 转换时间=（1/ADC 时钟频率）× （12+ADC 输入信号获取时间+CALIF）

表 32 转换时间说明表(1)

CLKDIV (2)	CALIF	SRADCKS	SRADACKS	AD 转换时间(3)
/4	0	01	00	1/（(16MHz / 4) / 2) × (12 + 0 + 16) = 14us
			01	1/（(16MHz / 4) / 2) × (12 + 0 + 8) = 10us
		10	00	1/（(16MHz / 4) / 4) × (12 + 0 + 16) = 28us
			01	1/（(16MHz / 4) / 4) × (12 + 0 + 8) = 20us
			10	1/（(16MHz / 4) / 4) × (12 + 0 + 4) = 16us
		11	00	1/（(16MHz / 4) / 8) × (12 + 0 + 16) = 56us
			01	1/（(16MHz / 4) / 8) × (12 + 0 + 8) = 40us
			10	1/（(16MHz / 4) / 8) × (12 + 0 + 4) = 32us
			11	1/（(16MHz / 4) / 8) × (12 + 0 + 2) = 28us
	1	01	00	1/（(16MHz / 4) / 2) × (12 + 1 + 16) = 14.5us
			01	1/（(16MHz / 4) / 2) × (12 + 1 + 8) = 10.5us
		10	00	1/（(16MHz / 4) / 4) × (12 + 1 + 16) = 29us
			01	1/（(16MHz / 4) / 4) × (12 + 1 + 8) = 21us
			10	1/（(16MHz / 4) / 4) × (12 + 1 + 4) = 17us
		11	00	1/（(16MHz / 4) / 8) × (12 + 1 + 16) = 58us
			01	1/（(16MHz / 4) / 8) × (12 + 1 + 8) = 42us
			10	1/（(16MHz / 4) / 8) × (12 + 1 + 4) = 34us
			11	1/（(16MHz / 4) / 8) × (12 + 1 + 2) = 30us
/8	0	00	00	1/（(16MHz / 8) / 1) × (12 + 0 + 16) = 14us
			01	1/（(16MHz / 8) / 1) × (12 + 0 + 8) = 10us
		01	00	1/（(16MHz / 8) / 2) × (12 + 0 + 16) = 28us
			01	1/（(16MHz / 8) / 2) × (12 + 0 + 8) = 20us
			10	1/（(16MHz / 8) / 2) × (12 + 0 + 4) = 16us
		10	00	1/（(16MHz / 8) / 4) × (12 + 0 + 16) = 56us
			01	1/（(16MHz / 8) / 4) × (12 + 0 + 8) = 40us
			10	1/（(16MHz / 8) / 4) × (12 + 0 + 4) = 32us
			11	1/（(16MHz / 8) / 4) × (12 + 0 + 2) = 24us
		11	00	1/（(16MHz / 8) / 8) × (12 + 0 + 16) = 112us
			01	1/（(16MHz / 8) / 8) × (12 + 0 + 8) = 80us
			10	1/（(16MHz / 8) / 8) × (12 + 0 + 4) = 64us
			11	1/（(16MHz / 8) / 8) × (12 + 0 + 2) = 48us
	1	00	00	1/（(16MHz / 8) / 1) × (12 + 1 + 16) = 14.5us
			01	1/（(16MHz / 8) / 1) × (12 + 1 + 8) = 10.5us
		01	00	1/（(16MHz / 8) / 2) × (12 + 1 + 16) = 29us
			01	1/（(16MHz / 8) / 2) × (12 + 1 + 8) = 21us
			10	1/（(16MHz / 8) / 2) × (12 + 1 + 4) = 17us
		10	00	1/（(16MHz / 8) / 4) × (12 + 1 + 16) = 58us
			01	1/（(16MHz / 8) / 4) × (12 + 1 + 8) = 42us
			10	1/（(16MHz / 8) / 4) × (12 + 1 + 4) = 34us
			11	1/（(16MHz / 8) / 4) × (12 + 1 + 2) = 30us

		11	00	1/（(16MHz / 8) / 8) × (12 + 1 + 16) = 116us
			01	1/（(16MHz / 8) / 8) × (12 + 1 + 8) = 84us
			10	1/（(16MHz / 8) / 8) × (12 + 1 + 4) = 68us
			11	1/（(16MHz / 8) / 8) × (12 + 1 + 2) = 60us
/16	0	00	00	1/（(16MHz / 16) / 1) × (12 + 0 + 16) = 28us
			01	1/（(16MHz / 16) / 1) × (12 + 0 + 8) = 20us
			10	1/（(16MHz / 16) / 1) × (12 + 0 + 4) = 16us
		01	00	1/（(16MHz / 16) / 2) × (12 + 0 + 16) = 56us
			01	1/（(16MHz / 16) / 2) × (12 + 0 + 8) = 40us
			10	1/（(16MHz / 16) / 2) × (12 + 0 + 4) = 32us
			11	1/（(16MHz / 16) / 2) × (12 + 0 + 2) = 28us
		10	00	1/（(16MHz / 16) / 4) × (12 + 0 + 16) = 112us
			01	1/（(16MHz / 16) / 4) × (12 + 0 + 8) = 80us
			10	1/（(16MHz / 16) / 4) × (12 + 0 + 4) = 64us
			11	1/（(16MHz / 16) / 4) × (12 + 0 + 2) = 48us
		11	00	1/（(16MHz / 16) / 8) × (12 + 0 + 16) = 224us
			01	1/（(16MHz / 16) / 8) × (12 + 0 + 8) = 160us
			10	1/（(16MHz / 16) / 8) × (12 + 0 + 4) = 128us
			11	1/（(16MHz / 16) / 8) × (12 + 0 + 2) = 96us
	1	00	00	1/（(16MHz / 16) / 1) × (12 + 1 + 16) = 29us
			01	1/（(16MHz / 16) / 1) × (12 + 1 + 8) = 21us
			10	1/（(16MHz / 16) / 1) × (12 + 1 + 4) = 17us
		01	00	1/（(16MHz / 16) / 2) × (12 + 1 + 16) = 58us
			01	1/（(16MHz / 16) / 2) × (12 + 1 + 8) = 42us
			10	1/（(16MHz / 16) / 2) × (12 + 1 + 4) = 34us
			11	1/（(16MHz / 16) / 2) × (12 + 1 + 2) = 15us
		10	00	1/（(16MHz / 16) / 4) × (12 + 1 + 16) = 116us
			01	1/（(16MHz / 16) / 4) × (12 + 1 + 8) = 84us
			10	1/（(16MHz / 16) / 4) × (12 + 1 + 4) = 68us
			11	1/（(16MHz / 16) / 4) × (12 + 1 + 2) = 60us
		11	00	1/（(16MHz / 16) / 8) × (12 + 1 + 16) = 232us
			01	1/（(16MHz / 16) / 8) × (12 + 1 + 8) = 168us
			10	1/（(16MHz / 16) / 8) × (12 + 1 + 4) = 136us
			11	1/（(16MHz / 16) / 8) × (12 + 1 + 2) = 120us
/32	0	00	00	1/（(16MHz / 32) / 1) × (12 + 0 + 16) = 56us
			01	1/（(16MHz / 32) / 1) × (12 + 0 + 8) = 40us
			10	1/（(16MHz / 32) / 1) × (12 + 0 + 4) = 32us
			11	1/（(16MHz / 32) / 1) × (12 + 0 + 2) = 28us
		01	00	1/（(16MHz / 32) / 2) × (12 + 0 + 16) = 112us
			01	1/（(16MHz / 32) / 2) × (12 + 0 + 8) = 80us
			10	1/（(16MHz / 32) / 2) × (12 + 0 + 4) = 64us
			11	1/（(16MHz / 32) / 2) × (12 + 0 + 2) = 56us
		10	00	1/（(16MHz / 32) / 4) × (12 + 0 + 16) = 224us
			01	1/（(16MHz / 32) / 4) × (12 + 0 + 8) = 160us
			10	1/（(16MHz / 32) / 4) × (12 + 0 + 4) = 128us
			11	1/（(16MHz / 32) / 4) × (12 + 0 + 2) = 96us
		11	00	1/（(16MHz / 32) / 8) × (12 + 0 + 16) = 448us
			01	1/（(16MHz / 32) / 8) × (12 + 0 + 8) = 320us
			10	1/（(16MHz / 32) / 8) × (12 + 0 + 4) = 256us

			11	1/（(16MHz / 32) / 8) × (12 + 0 + 2) = 192us
	1	00	00	1/（(16MHz / 32) / 1) × (12 + 1 + 16) = 58us
			01	1/（(16MHz / 32) / 1) × (12 + 1 + 8) = 42us
			10	1/（(16MHz / 32) / 1) × (12 + 1 + 4) = 34us
			11	1/（(16MHz / 32) / 1) × (12 + 1 + 2) = 30us
		01	00	1/（(16MHz / 32) / 2) × (12 + 1 + 16) = 116us
			01	1/（(16MHz / 32) / 2) × (12 + 1 + 8) = 84us
			10	1/（(16MHz / 32) / 2) × (12 + 1 + 4) = 68us
			11	1/（(16MHz / 32) / 2) × (12 + 1 + 2) = 60us
		10	00	1/（(16MHz / 32) / 4) × (12 + 1 + 16) = 232us
			01	1/（(16MHz / 32) / 4) × (12 + 1 + 8) = 168us
			10	1/（(16MHz / 32) / 4) × (12 + 1 + 4) = 136us
			11	1/（(16MHz / 32) / 4) × (12 + 1 + 2) = 120us
		11	00	1/（(16MHz / 32) / 8) × (12 + 1 + 16) = 464us
			01	1/（(16MHz / 32) / 8) × (12 + 1 + 8) = 336us
			10	1/（(16MHz / 32) / 8) × (12 + 1 + 4) = 272us
			11	1/（(16MHz / 32) / 8) × (12 + 1 + 2) = 240us

(1) fosc=16MHz
(2) 代码选项
(3) AD 转换时间随 fosc 频率的改变而改变。

3.5.3 使用内部参考电压 1.40V的 校准方法

内部 1.4V 作参考时,由于 1.4V 本身的浮动范围在 1.3-1.45V 之间,所以需要对 1.4V 的参考进行校正。在烧录芯片的时候,已经在 E2PROM 的 082fH(不同型号 MCU 的位置有差别,注意参考用户手册)的地址里烧录了 1.4V 的校正值,所以在上电的时候,需要把这个校正值读出来,在采到 AD 后,把这个校正值与采到的 AD 值相乘然后除以 8000H,得到的最后结果才是实际 AD 值。

如果有多个通道在使用时,当检测其中任一通道时,其它模拟口要设置为数字口。

```
movlf  macro  d1,f1
              movlw  d1
              movwf  f1
              endm
movff  macro  f1,f2
              movfw  f1
              movwf  f2
              endm
;*******************************************************************************
 E2PROM_ROUT:                        ;上电先从E2PROM的82fH中读出1.4V参考的校准值
       movlf    2fH,eadrl            ;要读的EEPORM地址放入EADRL EADRH中
       movlf    08H,eadrh
       movp
       movwf    M_D_TEMP12           ;低位W中的数据放在M_D_TEMP12中
       movff    edath,M_D_TEMP11     ;高位EDATH的数据放在M_D_TEMP11中
       return
 AD_init:                            ;CPUCLK = 16/4 =4MHZ
       movlw  00000001B              ;B1:B0  01          ADC CLOCK = CPUCLK/2=2M
       movwf  SRADCON0               ;B5 B4 = 0  0       ADC输入信号获取时间 16个ADC时钟
       movlw  00000010B              ;B[5] 0             结果放在SRAD中
       movwf  SRADCON1               ;B[1:0] 10          内部参考电压1.4V
```

```
        movlw   00000000B               ;00H                AIN0输入
        movwf   SRADCON2
        movlw   00000000B
        movwf   SRADL                   ;[ADL7:0]           ADC数据的低8位
        movwf   SRADH                   ;[ADH11:8]          ADC数据的高4位
        return

        …
ADC_CONVERT:
        movlf   00001000B               ;把其它的模拟口设为数字口，打开当前的模拟口
        movwf   PT3CON
        movlf   30h,SRADCON2            ;转换第3通道的信号
        bsf     SRADCON1,ADEN
        call    delay40US
        bsf     SRADCON1,SRADS
        nop
        btfsc   SRADCON1,SRADS
        goto    $-1
        movfw   SRADL
        movwf   AD3_VALU_l,1
        movfw   SRADH
        movwf   AD3_VALU_h,1

        movff   AD3_VALU_h,M_D_TEMP3    ;采到的AD值作为被乘数
        movff   AD3_VALU_l,M_D_TEMP4
        movff   M_D_TEMP11,M_D_TEMP9
        movff   M_D_TEMP12,M_D_TEMP10   ;E2PROM里014AH读出的值为乘数
        call    F_Mul2_2                    ;调用乘法运算
        movlf   80H,M_D_TEMP9
        movlf   00H,M_D_TEMP10              ;AD值与校准系数得到的积作被除数/8000H
        call     F_Div4_2                   ;调用除法运算
        movff   M_D_TEMP3,AD3_VALU_h        ;把除法结果重新放在AD保存寄存器里
        movff   M_D_TEMP4,AD3_VALU_l
        …
```

3.5.4 AD失调电压校正

不同芯片由于离散性的原因，AD 的失调电压可能有正有负。

校正失调电压的方法：

在 AD 转换过程中通过不断变换 SRADCON1 寄存器中的 OFFEX 的值。如第一次 AD 转换 OFFEX 置 0，第二次 AD 转换 OFFEX 置 1，然后将第一次和第二次测试的 AD 值求平均值。两次转换得到的平均值就是去掉失调电压的正确结果。

```
…
clrf sradcon1      ;VDD 为参考电压,often=0,calif=0;enov=0,offex=0,vrefs=00
movlw 20h
movwf sradcon2     ;chs[3:0]=0010，选择通道 2
bsf sradcon1,7     ;使能 ADC 模块
call delay_10us
…
bsf sradcon1,6     ;srads=1,开始转换
btfsc sradcon1,6   ;检测转换是否完成
goto $-1
movlw sradl
movwf adtmp1_1
movlw sradh
movwf adtmph_1
…
bsf sradcon1,2     ;offex=1
bsf sradcon1,6     ;srads=1,开始转换
btfsc sradcon1,6   ;检测转换是否完成
goto $-1
movlw sradl
movwf adtmp1_2
movlw sradh
movwf adtmph_2
aver adtmph_1,adtmpl_1,adtmph_2,adtmpl_2 ;求两次 AD 值平均值，并保存在
                                          ;adtmph_1,adtmpl_1
…
```

3.5.5 数字比较器

ADC 模块可作为一个数字比较器。被测信号的输入频率应小于转换频率的 1/2。比较器的速率是和 AD 转换频率相关的。

操作：

1） 通过 ADC 通道选择控制位 chs[3:0]选择比较器负端的信号输入，之后把 OFTEN 置 1，CALIF 清 0，ENOV 置 0，把 SRADEN 置 1 使能 ADC，SRADS 置 1 启动转换，转换完成可把转换结果写入 SROFT 寄存器。
   也可以直接把负端信号的 AD 值直接写到 SROFT 寄存器中，即人为指定负端电压值。
2） 通过 ADC 通道选择控制位 chs[3:0]选择比较器正端的信号输入，之后把 OFTEN 置 0，CALIF 清 1，ENOV 置 1，把 SRADEN 置 1 使能 ADC，SRADS 置 1 启动转换。
3） AD 数据的最高位 SRAD[11]则是比较器的结果，为 0 时表示正端电压大于负端电压，为 1 时表示正端电压小于负端电压。SRAD[11:0]为差值，带符号位的补码。

比较通道 0 和通道 1 的电压值，通道 0 接比较器正端，通道 1 接比较器负端。

```
…
clrf sradcon1      ;VDD 为参考电压,often=0,calif=0;enov=0,offex=0,vrefs=00
bsf sradcon1,5     ;often=1,结果保存在 sroft 寄存器中
movlw 00h
movwf sradcon2     ;chs[3:0]=0000，选择通道 0 作为比较器负端
bsf sradcon1,7     ;使能 ADC 模块
call delay_10us
bsf sradcon1,6     ;srads=1,开始转换
btfsc sradcon1,6   ;检测转换是否完成
goto $-1
…
movlw 10h
movwf sradcon2     ;chs[3:0]=0001，选择通道 1 作为比较器正端
bcf sradcon1,5     ;often=0
bsf sradcon1,4     ;calif=1
bsf sradcon1,3     ;enov=1
bsf sradcon1,6     ;srads=1,开始转换
btfsc sradcon1,6   ;检测转换是否完成
goto $-1
btfsc sradh,3
goto le_cmp        ;正端电压小于负端电压
goto gt_cmp        ;正端大于等于负端电压
…
```

比较 1V 电压和通道 1 的电压，通道 1 接比较器正端，1V 接比较器负端，假设采用 5V 的 VDD 作为参考电压，那么 1V 的 AD 值为 0x333。

```
…
clrf sradcon1      ;VDD 为参考电压,often=0,calif=0;enov=0,offex=0,vrefs=00
movlw 10h
movwf sradcon2     ;chs[3:0]=0001，选择通道 1 作为比较器正端
bsf sradcon1,4     ;calif=1
bsf sradcon1,3     ;enov=1
movlw 03h
movwf srofth
movlw 33h
movwf sroftl       ;sroft 寄存器存入 333h，即 1V 作为比较器负端
bsf sradcon1,7     ;使能 ADC 模块
call delay_10us
bsf sradcon1,6     ;srads=1,开始转换
btfsc sradcon1,6   ;检测转换是否完成
goto $-1
btfsc sradh,3
goto le_cmp        ;正端电压小于负端电压
goto gt_cmp        ;正端大于等于负端电压
…
```

3.5.6 内部测量VDD的电压

用户可以通过使用内部参考电压或者外部参考电压输入（外部参考电压固定且不随 VDD 电压变化）两种方法来测试芯片内部 VDD 的电压。

使用外部参考电压，使用条件较多，需额外提供参考源。

使用内部参考电压不需要额外的硬件条件。但是，使用内部参考电压会由于本身内部参考电压值的不准而影响精度。可以通过内部参考电压校正来提高测试的精度。

外接 3V 作为参考电压，测 VDD 电压。选择通道 5，测出 1/8VDD 的 AD 值，之后乘以 8 得出 VDD 的 AD 值，再乘以参考电压则为 VDD 电压。

```
…
clrf sradcon1      ;often=0,calif=0;enov=0,offex=0,vrefs=00
bsf sradcon1,0     ;vrefs=01,选择外部参考电压，接 3V
movlw 50h
movwf sradcon2     ;chs[3:0]=0101，选择通道 5 ,1/8VDD
bsf sradcon1,7     ;使能 ADC 模块
call delay_10us
bsf sradcon1,6     ;srads=1,开始转换
btfsc sradcon1,6   ;检测转换是否完成
goto $-1
movlw sradl
movwf adtmp1
movlw sradh
movwf adtmph
bcf status,c
rlf adtmp1
rlf adtmph         ;AD 值乘以 2
rlf adtmp1
rlf adtmph         ;AD 值乘以 4
rlf adtmp1
rlf adtmph         ;AD 值乘以 8,小数点在 adtmph 的 bit3 和 bit4 之间
…
```

3.6 比较器/运算放大器

CSU8RF3224 有一个模拟比较器/运算放大器，带两个模拟输入端 CINA 和 CINB，也可以使用 ADC 的参考电压 VREF/2 作为比较器的一个输入，CO 脚可做为比较器/运算放大器的输出。

如果在输入输出之间外间一个反馈电阻，则比较器可做运算放大器使用。做运算放大器使用时，需配置 CMPCON 寄存器的 COS[2:0]位为 3'b100。

地址	名称	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	上电复位值
3eh	INTF3	CMPIF								0uuuuuuu
3fh	INTE3	CMPIE								0uuuuuuu
6ah	CMPCON	CMPEN	COS[2:0]						CMPOUT	0000uuu0

标识符	功能
CMPEN	比较器/运算放大器使能位 1：使能 0：禁止
COS[2:0]	比较器/运算放大器选择

COS[2:0]	功能描述
000	比较器/运算放大器未使用，PT5.2、PT5.3、PT5.4 做为普通 IO
001	用作比较器，PT5.3 接比较器负端，PT5.4 接比较器正端，PT5.2 做为普通 IO
010	用作比较器，PT5.3 接比较器正端，PT5.4 接比较器负端，PT5.2 做为普通 IO
011	用作比较器，PT5.3 接比较器正端，VREF/2 接比较器负端，PT5.4、PT5.2 做为普通 IO
100	用作运算放大器，PT5.2 做为运算放大器输出口，PT5.4 接运放正端，PT5.3 接运放负端，
101	用作比较器，PT5.3 接比较器负端，PT5.4 接比较器正端，PT5.2 做为比较器输出
110	用作比较器，PT5.3 接比较器正端，PT5.4 接比较器负端，PT5.2 做为比较器输出
111	用作比较器，PT5.3 接比较器正端，VREF/2 接比较器负端， PT5.2 做为比较器输出，PT5.4 做为普通 IO

CMPOUT	比较器的比较结果

表 33　比较器/运算放大器的寄存器各位功能表

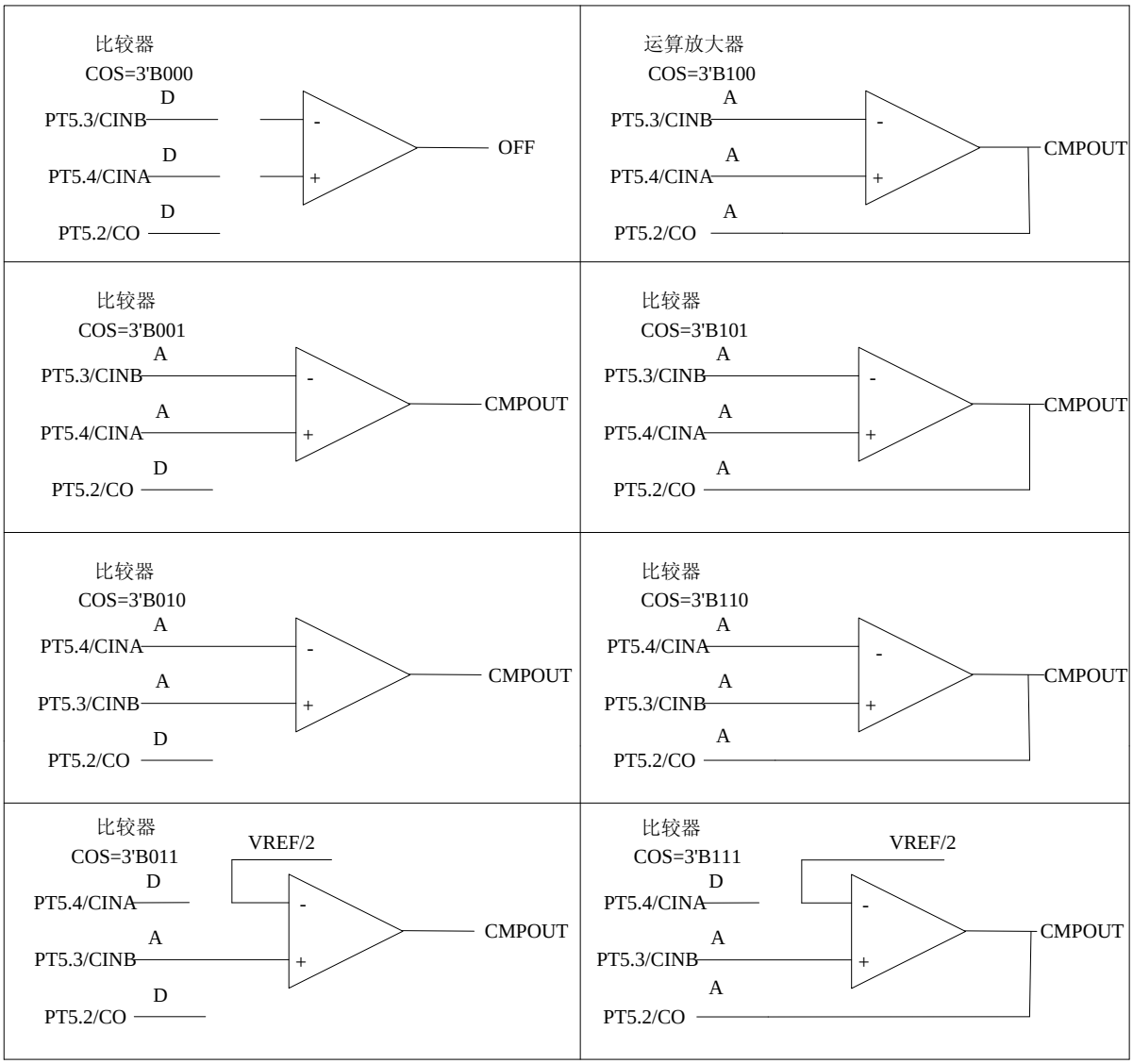
PT5.2	COS[2]	COS[1]	COS[0]	cmp_en	pt5en[2]	rst_n_pad	pt5od[2]
x(GPIO)	0	x	x	x	x	x	x
Z	1	0	0	0	x	x	x
Ao	1	0	0	1	x	x	x
Z	1	0	1	x	0	x	x
Z	1	0	1	x	x	0	x
0	1	0	1	0	1	1	x
Co（1）	1	0	1	1	1	1	1
Co	1	0	1	1	1	1	0
Z	1	1	0	x	0	x	x
Z	1	1	0	x	x	0	x
0	1	1	0	0	1	1	x

Co（1）	1	1	0	1	1	1	1
Co	1	1	0	1	1	1	0
Z	1	1	1	x	0	x	x
Z	1	1	1	x	x	0	x
0	1	1	1	0	1	1	x
Co（1）	1	1	1	1	1	1	1
Co	1	1	1	1	1	1	0

注：（1）此处的比较器输出是开漏输出，需要外部上拉电阻。

Ao 表示运算放大器模拟输出；Co 表示比较器输出；Z 表示高阻

下图为各种情况下比较器及运算放大器的使用示意图



3.6.1 比较器参考电压

用户可以使用 ADC 的参考电压 VREF 的二分之一作为比较器的一个输入，参考电压值可以为 VDD、内部 1.40v 参考电压（使用内部 1.40V 作为参考电压时必须将寄存器 SRADEN 置 1）、0V（使用 0V 作为参考电压时必须将寄存器 SRADEN 置 0）和 PT3.0 输入的外部参考电压，通过 SRADCON1 寄存器的 VREFS[1:0]位进行选择。

表 34　使用 ADC 参考电压的二分之一作为比较器的负端输入

VREFS[1]	VREFS[0]	SRADEN	比较器负端电压
0	0	X	VDD/2
0	1	X	外部参考电压/2
1	X	1	内部 1.40V/2
1	X	0	0V

3.6.2 比较器中断

1、 必须是比较器/运算放大器使能的情况下（CMPEN=1），比较器中断使能才能有效。
2、 在比较器/运算放大器使能的情况下（CMPEN=1），比较器输出端的状态变化都会使比较器结果变化标志位置 1。如果将比较器中断使能，比较器输出端的状态变化会使芯片进入中断，同时置标志位。

3.7 数据E2PROM

CSU8RF3224 具有 96 bytes 的 E2PROM，和用户程序存储器共同编址，具有掉电不丢失数据的特性。数据 E2PROM 的地址范围为 800H~82FH。

数据 E2PROM 地址为 82FH 的位置烧录时存放着内部参考电压 1.40V 的校准系数，若不需要用到内部参考电压 1.40V 时，82FH 可用作数据 E2PROM。

表 35 数据 E2PROM 寄存器列表

地址	名称	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	上电复位值
05H	WORK	工作寄存器								00000000
0AH	EADRH					PAR[11:8]				uuuu0000
0BH	EADRL	PAR[7:0]								00000000
0CH	EDATH	EDATH[7:0]								00000000

EADRH/EADRL 提供写或者读操作的数据地址；
EDATH/WORK 提供写或读操作所用的数据。
E2PROM 的读或写操作都是基于一个字（16 bits）的。

执行写操作时，在地址和数据寄存器输入相应的值，之后执行 TBLP 指令(指令后面参数固定为 0，如 tblp 0)，便可在相应的 E2PROM 地址写入相应的数据。写操作要求指令周期为 1MHz~4MHz，或代码选项选 32 个时钟周期且指令周期 200KHz~800KHz。执行一次写操作大概需要 3ms 的时间。执行 TBLP 指令的时候，应关闭所有中断和 WDT。

执行读操作时，在地址寄存器输入相应的值，之后执行 MOVP 指令，便可在相应的 E2PROM 地址的数据读入到 EDATH/WORK 寄存器中。执行一次读操作大概需要 3 个指令周期。

```
movlw 08H
movwf EADRH  ;给高字节地址赋值
movlw 00H
movwf EADRL  ;给低字节地址赋值
movlw aaH
movwf EDATH  ;给高字节数据赋值
movlw 55H    ;给低字节数据赋值
tblp 0       ;执行写操作
nop
…
movp         ;执行读操作
nop
…
```

3.8 烧录模块

烧写器的接口：

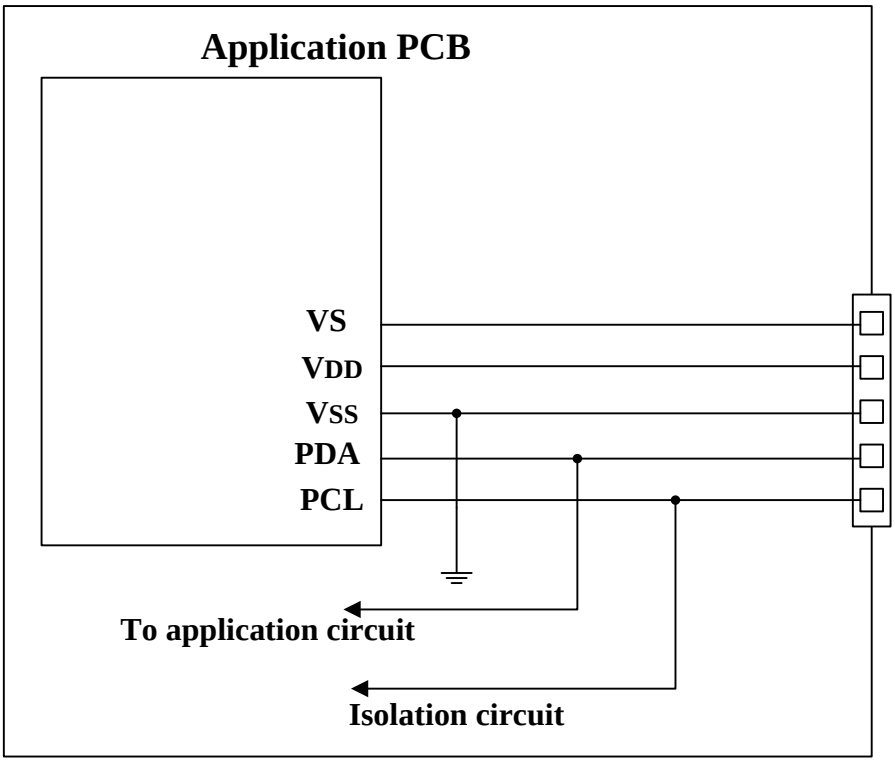


图17　烧写器接口图

表 36　烧录接口说明

端口名称	说明
VDD	输入
VSS	输入
PDA	PT1[2]输入/输出
PCL	PT1[1]输入
VS	PT3[0]AD 校正信号线，程序中若使用 AD 校正功能，则烧录时需连接此线，若无使用 1.4V 校正功能，则可以不连接，若连接，对烧录亦无影响。 （上述描述仅针对离线烧录器，使用仿真器烧录时无论是否使用 AD 校正功能，均不需要连接）

3.9 代码选项

<table>
<tr><th>标识符</th><th>功能</th></tr>
<tr><td>CLKDIV</td><td>指令周期选择
<table><tr><td>指令周期</td></tr><tr><td>指令周期=4 个时钟周期</td></tr><tr><td>指令周期=8 个时钟周期</td></tr><tr><td>指令周期=16 个时钟周期</td></tr><tr><td>指令周期=32 个时钟周期</td></tr></table></td></tr>
<tr><td>LVD_SEL</td><td>LVD 配置
<table><tr><td>功能</td></tr><tr><td>2.0V 上电/掉电复位。低电压检测关闭。</td></tr><tr><td>2.0V 上电/掉电复位。
STATUS 的 LVD24 作为 2.4V 的低电压检测器。
STATUS 的 LVD36 作为 3.6V 的低电压检测器。</td></tr><tr><td>2.4V 上电/掉电复位。
STATUS 的 LVD36 作为 3.6V 的低电压检测器。</td></tr><tr><td>3.6V 上电/掉电复位。</td></tr></table></td></tr>
<tr><td>RESET_PIN</td><td>复位引脚选择
PT1.3 作为复位引脚
PT1.3 作为普通输入口</td></tr>
<tr><td>XTAL_PIN</td><td>晶振引脚选择
<table><tr><td>晶振引脚</td></tr><tr><td>PT1.1 和 PT1.2 作为普通 IO 口</td></tr><tr><td>PT1.1 作为外部 RC 时钟输入；或者
PT1.1 通过外部时钟源直接灌入时钟；
PT1.2 还是普通 IO 口；</td></tr><tr><td>PT1.1 和 PT1.2 接外部晶振为 32768Hz</td></tr><tr><td>PT1.1 和 PT1.2 接外部晶振
4M~16MHz；还可通过 PT1.1 灌时钟，
PT1.2 悬空</td></tr></table></td></tr>
</table>

4 MCU指令集

表 37 表 MCU 指令集

指令	操作	指令周期	标志位
ADDLW k	[W]←[W]+k	1	C,DC,Z
ADDPCW	[PC] ←[PC]+1+[W]	2	~
ADDWF f,d	[Destination] ←[f]+[W]	1	C,DC,Z
ADDWFC f,d	[Destination] ←[f]+[W]+C	1	C,DC,Z
ANDLW k	[W]←[W] AND k	1	Z
ANDWF f,d	[Destination] ← [W] AND [f]	1	Z
BCF f,b	[f<b>]←0	1	~
BSF f,b	[f<b>]←1	1	~
BTFSC f,b	Jump if[f<b>]=0	1/2	~
BTFSS f,b	Jump if[f<b>]=1	1/2	~
CALL k	Push PC+1 and Goto K	2	~
CLRF f	[f]←0	1	Z
CLRWDT	Clear watch dog timer	1	~
COMF f,d	[f]←NOT([f])	1	Z
DAW	Decimal Adjust W	1	C,DC
DECF f,d	[Destination] ←[f] -1	1	Z
DECFSZ f,d	[Destination] ←[f] -1,jump if the result is zero	1/2	~
GOTO k	PC←k	2	~
HALT	CPU Stop	1	~
INCF f,d	[Destination] ←[f]+1	1	Z
INCFSZ f,d	[Destination] ←[f]+1,jump if the result is zero	1/2	~
IORLW k	[W]←[W] OR k	1	Z
IORWF f,d	[Destination] ← [W] OR [f]	1	Z
MOVFW f	[W]←[f]	1	~
MOVLW k	[W]←k	1	~
MOVP	Read e2prom	3	~
MOVWF f	[f]←[W]	1	~
NOP	No operation	1	~
POP	Pop W and Status	2	~
PUSH	Push W and Status	2	~
RETFIE	Pop PC and GIE =1	2	~
RETLW k	RETURN and W=k	2	~
RETURN	POP PC	2	~
RLF f,d	[Destination<n+1>] ←[f<n>]	1	C,Z
RRF f,d	[Destination<n-1>] ←[f<n>]	1	C,Z
SLEEP	STOP OSC	1	PD
SUBLW k	[W] ← k – [W]	1	C,DC,Z
SUBWF f,d	[Destinnation] ← [f]– [W]	1	C,DC,Z
SUBWFC f,d	[Destinnation] ← [f]– [W]-1+C	1	C,DC,Z
SWAPF f,d	swap f	1	~
TBLP k	Write e2prom	-	~
XORLW k	[W]←[W] XOR k	1	Z
XORWF f,d	[Destination] ← [W] XOR [f]	1	Z

参数说明：

f: 数据存储器地址(00H ~FFH)

W: 工作寄存器

k: 立即数

d: 目标地址选择: d=0 结果保存在工作寄存器, d=1: 结果保存在数据存储器 f 单元

b: 位选择(0~7)

[f]: f 地址的内容

PC: 程序计数器

C: 进位标志

DC: 半加进位标志

Z: 结果为零标志

PD: 睡眠标志位

TO: 看门狗溢出标志

WDT: 看门狗计数器

表 38 MCU 指令集描述

1

ADDLW	加立即数到工作寄存器
指令格式	ADDLW K (0<=K<=FFH)
操作	(W)<—(W)+K
标志位	C，DC，Z
描述	工作寄存器的内容加上立即数 K 结果保存到工作寄存器中
周期	1
例子 ADDLW 08H	在指令执行之前： W=08H 在指令执行之后： W=10H

2

ADDPCW	将 W 的内容加到 PC 中
指令格式	ADDPCW
操作	(PC)<—(PC)+1+(W) 当(W)<=7FH (PC)<—(PC)+1+(W)-100H 其余
标志位	没有
描述	将地址 PC+1+W 加载到 PC 中
周期	2
例子 1 ADDPCW	在指令执行之前： W=7FH，PC=0212H 指令执行之后： PC=0292H
例子 2 ADDPCW	在指令执行之前： W=80H，PC=0212H 指令执行之后： PC=0193H
例子 3 ADDPCW	在指令执行之前： W=FEH，PC=0212H 指令执行之后： PC=0211H

3

ADDWF	加工作寄存器到 f
指令格式	ADDWF f,d 0<=f<=FFH d=0,1
操作	[目标地址]<—(f)+(W)
标志位	C，CD，Z
描述	将 f 的内容和工作寄存器的内容加到一起。 如果 d 是 0，结果保存到工作寄存器中。 如果 d 是 1，结果保存到 f 中。
周期	1
例子 1 ADDWF f 0	指令执行之前： f=C2H W=17H 在指令执行之后 f=C2H W=D9H
例子 2 ADDWF f 1	指令执行之前 f=C2H W=17H 指令执行之后 f=D9H W=17H

4

ADDWFC	将 W f 和进位位相加
指令格式	ADDWFC f，d 0<=f<=FFH d=0,1
操作	(目标地址)<—(f)+(W)+C
标志位	C，DC，Z
描述	将工作寄存器的内容和 f 的内容以及进位位相加 当 d 为 0 时结果保存到工作寄存器 当 d 为 1 时结果保存到 f 中
周期	1
例子 ADDWFC f，1	指令执行之前 C=1 f=02H W=4DH 指令执行之后 C=0 f=50H W=4DH

5

ANDLW	工作寄存器与立即数相与
指令格式	ANDLW K 0<=K<=FFH
操作	(W)<—(W) AND K
标志位	Z
描述	将工作寄存器的内容与 8bit 的立即数相与，结果保存到工作寄存器中。
周期	1
例子 ANDLW 5FH	在指令执行之前 W=A3H 在指令执行之后 W=03H

6

ANDWF	将工作寄存器和 f 的内容相与
指令格式	ANDWF f，d 0<=f<=FFH d=0,1
操作	(目标地址)<—(W) AND (f)
标志位	Z
描述	将工作寄存器的内容和 f 的内容相与 如果 d 为 0 结果保存到工作寄存器中 如果 d 为 1 结果保存到 f 中
周期	1
例子 1 ANDWF f，0	在指令执行之前 W=0FH f=88H 在指令执行之后 W=08H f=88H
例子 2 ANDWF f，1	在指令执行之前 W=0FH f=88H 在指令执行之后 W=0FH f=08H

7

BCF	清除 f 的某一位
指令格式	BCF f，b 0<=f<=FFH 0<=b<=7
操作	(f[b])<—0
标志位	无
描述	F 的第 b 位置为 0
周期	1
例子 BCF FLAG 2	指令执行之前： FLAG=8DH 指令执行之后： FLAG=89H

8

BSF	F 的 b 位置 1
指令格式	BSF f，b 0<=f<=FFH 0<=b<=7
操作	(f[b])<—1
标志位	无
描述	将 f 的 b 位置 1
周期	1
例子 BSF FLAG 2	在指令执行之前 FLAG=89H 在指令执行之后 FLAG=8DH

9

BTFSC	如果 bit 测试为 0 则跳转
指令格式	BTFSC f，b 0<=f<=FFH 0<=b<=7
操作	Skip if (f[b])=0
标志位	无
描述	如果 f 的 bit 位是 0，下一条取到的指令将被丢到，然后执行一条空指令组成一个两周期的指令。
周期	无跳转则为 1 个指令周期，否则 2 个指令周期

例子
NODE BTFSC FLAG 2
OP1:
OP2:
	在程序执行以前
	PC=address(NODE)
	指令执行之后
	If(FLAG[2])=0
	PC=address(OP2)
	If(FLAG[2])=1
	PC=address(OP1)

10

BTFSS	如果 bit 测试为 1，则跳转
指令格式	BTFSS f，b 0<=f<=FFH 0<=b<=7
操作	Skip if (f[b])=1
标志位	无
描述	如果 f 的 bit 位是 1，下一条取到的指令将被丢到，然后执行一条空指令组成一个两周期的指令。
周期	无跳转则为 1 个指令周期，否则 2 个指令周期

例子
NODE BTFSS FLAG 2
OP1:
OP2:
	在程序执行以前
	PC=address(NODE)
	指令执行之后
	If(FLAG[2])=0
	PC=address(OP1)
	If(FLAG[2])=1
	PC=address(OP2)

11

CALL	子程序调用
指令格式	CALL K 0<=K<=3FFH
操作	(top stack)<—PC+1 PC<—K
标志位	无
描述	子程序调用，先将 PC+1 压入堆栈，然后把立即数地址下载到 PC 中。
周期	2

12

CLRF	清除 f
指令格式	CLRF f 0<=f<=FFH
操作	(f)<—0
标志位	Z
描述	将 f 的内容清零
周期	1
例子 CLRF WORK	在指令执行之前 WORK=5AH 在指令执行之后 WORK=00H

*注。当 clrf 清除 status 寄存器时，标志位 Z 不会置高

13

CLRWDT	清除看门狗定时器
指令格式	CLRWDT
操作	看门狗计数器清零
标志位	无
描述	清除看门狗定时器
周期	1
例子 CLRWDT	指令执行之后 WDT=0

14

COMF	f 取反
指令格式	COMF f，d 0<=f<=FFH d=0,1
操作	(目的地址)<—NOT(f)
标志位	Z
描述	将 f 的内容取反， 当 d 为 0 时，结果保存到工作寄存器中， 当 d 为 1 时，结果保存到 f 中。
周期	1
例子 COMF f，0	在指令执行之前 W=88H，f=23H 在指令执行之后 W=DCH，f=23H
例子 2 COMF f，1	在指令执行之前 W=88H，f=23H 在指令执行之后 W=88H，f=DCH

15

DAW	十进制调整 W 寄存器
指令格式	DAW
操作	十进制调整 W 寄存器
标志位	C,DC
描述	一般与加法一起使用。 如果低半字节的值大于 9 或 DC 为 1 时，低半字节加 6； 如果高半字节的值大于 9 或 C 为 1 时，高半字节加 6
周期	1
例子 若 W=25H； ADDLW 39H DAW	在 DAW 指令执行之前 W=25H+39H =64=5EH 在指令执行之后 W=（64）BCD 25H + 39H ---------- 5EH + 06H ---------- 64H

16

DECF	f 减 1
指令格式	DECF f，d 0<=f<=FFH d=0,1
操作	(目的地址)<—(f)-1
标志位	Z
描述	F 的内容减 1 当 d 为 0 时，结果保存到工作寄存器中 当 d 为 1 时，结果保存到 f 中。
周期	1
例子 DECF f，0	在指令执行之前 W=88H f=23H 在指令执行之后 W=22H f=23H
例子 2 DECF f，1	在指令执行之前 W=88H f=23H 在指令执行之后 W=88H f=22H

17

DECFSZ	f 减 1 如果为 0 则跳转
指令格式	DECFSZ f，d 0<=f<=FFH d=0,1
操作	(目的地址)<—(f)-1,如果结果为 0 跳转
标志位	无
描述	f 的内容减 1。 如果 d 为 0，结果保存到工作寄存器中。 如果 d 为 1，结果保存到 f 中 如果结果为 0，下一条已经取到的指令将被丢掉，然后插入一条 NOP 指令组成一个两个周期的指令。
周期	无跳转则为 1 个指令周期，否则 2 个指令周期

例子
Node DECFSZ FLAG，1
OP1：
OP2：
	在指令执行之前
	PC=address(Node)
	在指令执行之后
	(FLAFG)=(FLAG)-1
	If(FLAG)=0
	PC=address(OP2)
	If(FLAG)!=0
	PC=address(OP1)

18

GOTO	无条件跳转
指令格式	GOTO K 0<=K<=3FFH
操作	PC<—K
标志位	无
描述	立即地址载入 PC
周期	2

19

HALT	停止 CPU 时钟
指令格式	HALT
操作	CPU 停止
标志位	无
描述	CPU 时钟停止，晶振仍然工作，CPU 能够通过内部或者外部中断重启。
周期	1

20

INCF	f 加 1
指令格式	INCF f，d 0<=f<=FFH d=0,1
操作	(目的地址)<—(f)+1
标志位	Z
描述	f 加 1 如果 d 为 0，结果保存到工作寄存器中 如果 d 为 1，结果保存到 f 中。
周期	1
例子 INCF f，0	在指令执行之前 W=88H f=23H 在指令执行之后 W=24H f=23H
例子 2 INCF f，1	在指令执行之前 W=88H f=23H 在指令执行之后 W=88H f=24H

21

INCFSZ	f 加 1，如果结果为 0 跳转
指令格式	INCFSZ f，d 0<=f<=FFH d=0,1
操作	(目的地址)<—(f)+1 如果结果为 0 就跳转
标志位	无
描述	f 的内容加 1。 如果 d 为 0，结果保存到工作寄存器中。 如果 d 为 1，结果保存到 f 中 如果结果为 0，下一条已经取到的指令将被丢掉，然后插入一条 NOP 指令组成一个两个周期的指令。
周期	无跳转则为 1 个指令周期，否则 2 个指令周期

例子
Node INCFSZ FLAG，1
OP1：
OP2：
	在指令执行之前
	PC=address(Node)
	在指令执行之后
	(FLAFG)=(FLAG)+1
	If(FLAG)=0
	PC=address(OP2)
	If(FLAG)!=0
	PC=address(OP1)

22

IORLW	工作寄存器与立即数或
指令格式	IORLW K 0<=K<=FFH
操作	(W)<—(W)|K
标志位	Z
描述	立即数与工作寄存器的内容或。结果保存到工作寄存器中。
周期	1
例子 IORLW 85H	在指令执行之前 W=69H 在指令执行之后 W=EDH

23

IORWF	f 与工作寄存器或
指令格式	IORWF f，d 0<=f<=FFH d=0,1
操作	(目的地址)<—(W)|(f)
标志位	Z
描述	f 和工作寄存器或 当 d 为 0 时，结果保存到工作寄存器中 当 d 为 1 时，结果保存到 f 中
周期	1
例子 IORWF f,1	在指令执行前 W=88H f=23H 在指令执行后 W=88H f=ABH

24

MOVFW	传送到工作寄存器
指令格式	MOVFW f 0<=f<=FFH
操作	(W)<—(f)
标志位	无
描述	将数据从 f 传送到工作寄存器
周期	1
例子 MOVFW f	在指令执行之前 W=88H f=23H 在指令执行之后 W=23H f=23H

25

MOVLW	将立即数传送到工作寄存器中
指令格式	MOVLW K 0<=K<=FFH
操作	(W)<—K
标志位	无
描述	将 8bit 的立即数传送到工作寄存器中
周期	1
例子 MOVLW 23H	在指令执行之前 W=88H 在指令执行之后 W=23H

26

MOVP	读 E2PROM 数据
指令格式	MOVP
操作	把 E2PROM 数据读到 EDATH/WORK 中
标志位	无
描述	把地址为 EADRH/EADRL 的 E2PROM 数据读到 EDATH/WORK 中
周期	2
例子 MOVP	在指令执行之前 EADRH=04H，EADRL=00H 地址为 0400H 的 E2PROM 数据位 1234H 在指令执行之后 EDATH=12H，W=34H

27

MOVWF	将工作寄存器的值传送到 f 中
指令格式	MOVWF f 0<=f<=FFH
操作	(f)<—(W)
标志位	无
描述	将工作寄存器的值传送到 f 中
周期	1
例子 MOVWF f	在指令执行之前 W=88H f=23H 在指令执行之后 W=88H f=88H

28

NOP	无操作
指令格式	NOP
操作	无操作
标志位	无
描述	无操作
周期	1

29

PUSH	把 work 和 status 寄存器入栈保护
指令格式	PUSH
操作	(top stack)<—work/status
标志位	无
描述	把 work 和 status 寄存器的值做入栈处理，支持 8 级堆栈，不同于 PC 堆栈； 其中状态寄存器不包括 LVD36，LVD24，PD 和 TO。
周期	2

30

POP	把 work 和 status 寄存器出栈处理
指令格式	POP
操作	(Top Stack)=>work/status Pop Stack
标志位	无
描述	把当前栈顶的值做出栈处理，分别更新 work 和 status 寄存器，支持 8 级堆栈，不同于 PC 堆栈；其中状态寄存器不包括 LVD36，LVD24，PD 和 TO。
周期	2

31

RETFIE	从中断返回
指令格式	RETFIE
操作	(Top Stack)=>PC Pop Stack 1=>GIE
标志位	无
描述	PC 从堆栈顶部得到，然后出栈，设置全局中断使能位为 1
周期	2

32

RETLW	返回，并将立即数送到工作寄存器中
指令格式	RETLW K 0<=K<=FFH
操作	(W)<—K (Top Stack)=>PC Pop Stack
标志位	无
描述	将 8bit 的立即数送到工作寄存器中，PC 值从栈顶得到，然后出栈
周期	2

33

RETURN	从子程序返回
指令格式	RETURN
操作	(Top Stack)=>PC Pop Stack
标志位	无
描述	PC 值从栈顶得到，然后出栈
周期	2

34

RLF	带进位左移
指令格式	RLF f，d 0<=f<=FFH d=0,1
操作	(目标地址[n+1])<—(f[n]) (目标地址[0])<—C C<—(f[7])
标志位	C，Z
描述	F 带进位位左移一位 如果 d 为 0，结果保存到工作寄存器 如果 d 为 1，结果保存到 f 中
周期	1
例子 RLF f，1	在指令执行之前 C=0 W=88H f=E6H 在指令执行之后 C=1 W=88H f=CCH

35

RRF	带进位右移
指令格式	RRF f，d 0<=f<=FFH d=0,1
操作	(目标地址[n-1])<—(f[n]) (目标地址[7])<—C C<—(f[7])
标志位	C
描述	F 带进位位右移一位 如果 d 为 0，结果保存到工作寄存器 如果 d 为 1，结果保存到 f 中
周期	1
例子 RRF f，0	在指令执行之前 C=0 W=88H f=95H 在指令执行之后 C=1 W=4AH f=95H

36

SLEEP	晶振停止
指令格式	SLEEP
操作	CPU 晶振停止
标志位	PD
描述	CPU 晶振停止。CPU 通过外部中断源重启
周期	1

37

SUBLW	立即数减工作寄存器的值
指令格式	SUBLW K 0<=K<=FFH
操作	(W)<—K-(W)
标志位	C，DC，Z
描述	8bit 的立即数减去工作寄存器的值，结果保存到工作寄存器中
周期	1
例子 SUBLW 02H	在指令执行之前 W=01H 在指令执行之后 W=01H C=1(代表没有借位) Z=0(代表结果非零)
例子 2 SUBLW 02H	在指令执行之前 W=02H 在指令执行之后 W=00H C=1(代表没有借位) Z=1(代表结果为零)
例子 2 SUBLW 02H	在指令执行之前 W=03H 在指令执行之后 W=FFH C=0(代表有借位) Z=0(代表结果非零)

38

SUBWF	f 的值减工作寄存器的值
指令格式	SUBWF f，d 0<=f<=FFH d=0,1
操作	(目标地址)<—(f)-(W)
标志位	C，DC，Z
描述	f 的值减去工作寄存器的值。 如果 d 为 0，结果保存到工作寄存器 如果 d 为 1，结果保存到 f 中
周期	1
例子 SUBWF f，1	在指令执行之前 f=33H W=01H 在指令执行之后 f=32H C=1 Z=0
例子 2 SUBWF f，1	在指令执行之前 f=01H W=01H 在指令执行之后 f=00H C=1 Z=1
例子 3 SUBWF f，1	在指令执行之前 f=04H W=05H 在指令执行之后 f=FFH C=0 Z=0

39

SUBWFC	带借位的减法
指令格式	SUBWFC f，d 0<=f<=FFH d=0,1
操作	(目标地址)<—(f)-(W)-1+C
标志位	C，DC，Z
描述	f 的值减去工作寄存器的值 如果 d 为 0，结果保存到工作寄存器 如果 d 为 1，结果保存到 f 中
周期	1
例子 SUBWFC f，1	在指令执行之前 W=01H f=33H C=1 在指令执行之后 f=32H C=1 Z=0
例子 2 SUBWFC f，1	在指令执行之前 W=01H f=02H C=0 在指令执行之后 f=00H C=1 Z=1
例子 3 SUBWFC f，1	在指令执行之前 W=05H f=04H C=0 在指令执行之后 f=FEH C=0 Z=0

40

SWAPF	交换寄存器的值
指令格式	SWAPF f，d 0<=f<=FFH d=0,1
操作	(des[3:0])<—f[7:4] (des[7:4])<—f[3:0]
标志位	无
描述	把 f 寄存器的高 4 位数据给目标寄存器的低 4 位； 把 f 寄存器的低位数据给目标寄存器的高 4 位 d 为 1 时，f 寄存器为目标寄存器；否则，w 寄存器为目标寄存器
周期	1
例子 SWAPF f,1	在指令执行之前 f=ACH 在指令执行之后 f=CAH

41

TBLP	写 E2PROM
指令格式	TBLP k （k 取 0）
操作	写 E2PROM
标志位	无
描述	把 EDATH/WORK 的数据写到地址为 EADRH/EADRL 的 E2PROM 里
周期	约为 3ms 时间
例子 TBLP 0	在指令执行之前 EDATH=BAH，W=ACH，EADRH=04H，EADRL=00H 在指令执行之后 把 BAACH 写到地址为 0400H 的 E2PROM 里

42

XORLW	工作寄存器的值与立即数异或
指令格式	XORLW K 0<=K<=FFH
操作	(W)<—(W)^K
标志位	Z
描述	8bit 的立即数与工作寄存器的值异或，结果保存在工作寄存器中
周期	1
例子 XORLW 5FH	在指令执行之前 W=ACH 在指令执行之后 W=F3H

43

XORWF	f 的值与工作寄存器的值异或
指令格式	XORWF f，d 0<=f<=FFH d=0,1
操作	(目标地址)<—(W)^(f)
标志位	Z
描述	F 的值与工作寄存器的值异或， 当 d 为 0 时，结果保存到工作寄存器中 当 d 为 1 时，结果保存到 f 中
周期	1
例子 XORWF f，1	在指令执行之前 W=ACH f=5FH 在指令执行之后 f=F3H

5 电气特性

5.1 极限值

参数	范围	单位
电源 VDD	-0.3~6.0	V
引脚输入电压	-0.3~VDD+0.3	V
工作温度	-40~+125	ºC
存贮温度	-55~+150	ºC
焊接温度，时间	220ºC，10 秒	

5.2 直流特性（VDD = 5V，T_A = 25ºC，如无其他说明则都是此条件）

符号	参数	测试条件	最小值	典型值	最大值	单位
VDD	工作电压	25 ºC	2.3	5	5.5	V
		-40 ºC ~+85 ºC	2.5	5	5.5	V
FXT	外部晶振频率				16	MHz
EXT_CK	外部灌入时钟	25 ºC，代码选项 XTAL_PIN 选 ERC			16	MHz
		25 ºC，代码选项 XTAL_PIN 选外部低速晶振			50	KHz
		25 ºC，代码选项 XTAL_PIN 选外部高速晶振			4	MHz
Vpor	系统电源电压上升速率		0.15			V/ms
Tcpu	指令周期	VDD：2.3V~5.5V	2000			ns
		VDD：2.4V~5.5V	500			
		VDD：3.6V~5.5V	250			
VIH	数字输入高电平	PT1，PT3，PT5	0.75VDD			V
	复位输入高电平		0.8VDD			
VIL	数字输入低电平	PT1，PT3，PT5			0.3VDD	V
	复位输入低电平				0.2VDD	
IPU	上拉电流	PT1,PT3,PT5；Vin = 0;		50		uA
IOH	高电平输出电流	VOH=0.9VDD；VDD=5V		9.5		mA
		VOH=0.9VDD；VDD=3V		4.5		mA
IOL	低电平输出电流	VOL=0.1VDD；VDD=5V		13.5		mA
		VOL=0.1VDD；VDD=3V		6.8		mA
LVD	复位电压/低电压检测电压	2.0V 上电/掉电复位点；25 度	1.9	2.0	2.3	V
		2.0V 上电/掉电复位点；-40~85 度	1.8	2.0	2.5	
		2.4V 上电/掉电复位点；25 度	2.3	2.4	2.7	
		2.4V 上电/掉电复位点；-40~85 度	2.0	2.4	3.0	
		3.6V 上电/掉电复位点；25 度	3.3	3.6	4.0	
		3.6V 上电/掉电复位点；-40~85 度	3.0	3.6	4.5	
IRC	内置 RC 时	25℃，5V	15.84	16.0	16.16	MHz

	钟	-40℃~85℃, 2.3V~5.5V	15.3	16.0	16.5	
WDT	内置看门狗时钟	25℃，5V	29	32	35	KHz
		-40℃~85℃, 2.3V~5.5V	26	32	38	KHz
Tint0,1	中断触发脉宽	25℃，5V	Tcpu			ns
IDD1	sleep 模式电流	VDD=3V, 关掉 WDT		0.7		uA
		VDD=3V, 打开 WDT		2.7		uA
		VDD=5V, 关掉 WDT		0.9		uA
		VDD=5V, 打开 WDT		3.6		uA
IDD2	工作电流	内部振荡器关闭 (fcpu=fosc/4) fosc = 32768Hz,3V		8		uA
		内部振荡器关闭 (fcpu=fosc/4) fosc = 32768Hz,5V		13		
		内部晶振模式，F=16MHz，VDD=3V，fcpu=fosc/4		2.7		mA
		内部晶振模式，F=16MHz，VDD=3V，fcpu=fosc/8		1.5		
		内部晶振模式，F=16MHz，VDD=3V，fcpu=fosc/16		0.9		
		内部晶振模式，F=16MHz，VDD=3V，fcpu=fosc/32		0.6		
		内部晶振模式，F=16MHz，VDD=5V，fcpu=fosc/4		4.2		
		内部晶振模式，F=16MHz，VDD=5V，fcpu=fosc/8		2.4		
		内部晶振模式，F=16MHz，VDD=5V，fcpu=fosc/16		1.5		
		内部晶振模式，F=16MHz，VDD=5V，fcpu=fosc/32		1.0		
IDD3	工作电流	ERC 模式，VDD=5V，fcpu=fosc/2				mA
		C=47P R=1KΩ F=8.4MHZ		7.03		
		C=47P R=10KΩ F=3.7MHZ		2.31		
		C=47P R=100KΩ F=530KHZ		0.33		
		C=47P R=1MΩ F=60KHZ		0.05		
		C=100P R=1KΩ F=8.3MHZ		8.26		
		C=100P R=10KΩ F=3.3MHZ		2.18		
		C=100P R=100KΩ F=350KHZ		0.29		
		C=100P R=1MΩ F=50KHZ		0.05		
		ERC 模式，VDD=3V，fcpu=fosc/2				
		C=47P R=1KΩ F=7.4MHZ		3.84		
		C=47P R=10KΩ F=2.7MHZ		0.95		
		C=47P R=100KΩ F=300KHZ		0.13		
		C=47P R=1MΩ F=40KHZ		0.02		
		C=100P R=1KΩ F=7.3MHZ		4.34		
		C=100P R=10KΩ F=1.6MHZ		0.84		
		C=100P R=100KΩ F=210KHZ		0.12		
		C=100P R=1MΩ F=30KHZ		0.02		

5.3 ADC特性（VDD = 5V，T_A = 25ºC，如无其他说明则都是此条件）

符号	参数	测试条件	最小值	典型值	最大值	单位
VDD	ADC 工作电压范围	25 ℃	2.3	5	5.5	V
		-40 ℃ ~+85 ℃	2.5	5	5.5	V
AIN0~ AIN5 input voltage	模拟输入范围	VREF 受寄存器 VREFS[1:0]控制	0		VREF	V
Vref input range	外部参考电压输入范围	VREFS[1:0]=01	0		VDD	V
ADC current consumption	ADC 功耗	VDD=5V(VDD 作为参考电压)		0.38		mA
		VDD=3V(VDD 作为参考电压)		0.36		mA
ADC Conversion Cycle Time	ADC 转换周期	注意模拟信号的输出阻抗对于 ADC 转换周期的限制，典型值 10uS 要求输出阻抗不超过 10K	1.75	10		uS
INL	积分非线性	SRADACKS[1:0]=01； SRADCKS[1:0]=01；		±4	±8	LSB
No missing code	无失码	VREFS[1:0]=01，外部参考电压	10	11	12	Bits
		VREFS[1:0]=00，VDD 做为参考电压	9	10	11	Bits
		VREFS[1:0]=10，内部参考电压	7	8	9	Bits
IVREF	内部参考电压			1.40		V
IVREF temp drift	内部参考电压温漂			50		ppm
Offset	ADC 失调电压			7.5		mV

5.4 比较器的直流特性

符号	参数	测试条件	最小值	典型值	最大值	单位
VDD	工作电压	25 ºC	2.3	5	5.5	V
		-40 ºC ~+85 ºC	2.5	5	5.5	V
Comp Vos	比较器/运算放大器的失调电压			±1	±3	mV
Comp Vos drift	比较器/运算放大器的失调电压温漂			5*		uV
Comp IVR	比较器输入电压范围	VDD=5V，GND=0V	0		5	V
COMP OVS	比较器/运算放大器输出电压范围	VDD=5V， GND=0V，Rload=10KΩ	0		0.3	V
			4.7		5	V
Icomp	比较器工作电流			150		uA
Iopa	运算放大器工作电流			350		uA
Large signal voltage gain	运算放大器大信号增益	Rload=2KΩ		40*		V/mV
PSRR	比较器/运算放大器电源电压抑制比			60*		dB
CMRR	比较器/运算放大器共模抑制比			60*		dB
Comp/opa vdd range	比较器/运算放大器工作电压范围		2.3V		5.5	V
Comp Reponse	比较器响应时间			1(1)		uS
CMP LSB	比较器最小分辨率			1		mV
Comp VCM	比较器、运算放大器共模电压输入范围		0		5	V
OPA output current	运算放大器输出电流	Isource（v+=1，v-=0，VDD=5V，CO=2V）	20*			mA
		Isink（v+=0，v-=1，VDD=5V，CO=2V）	20*			mA
		Isink（v+=0，v-=1，VDD=5V，CO=200mV）	3*			mA
OPA THD	运放的谐波失真 Vsin=10KHz	VDD=5V，Vopp=5V		60*		dB
		VDD=5V，Vopp=4V		80*		dB
		VDD=5V，Vopp=3V		95*		dB
		VDD=2.3，Vopp=2.3V		55*		dB
		VDD=2.3V，Vopp=1V		75*		dB
Max CAP	允许直接接到运算放大器输出端的最大到地的负载电容				100	pF

(1) 响应时间测量是在比较器一端接(VDD-1.5)/2，另外一端从 VSS 跳变到 VDD-1.5。

* 表示理论设计值，未经过实际测试。

5.5 RC时钟频率特性

下图位实际芯片的测试数据，不同芯片会略微有所差异，仅供参考。

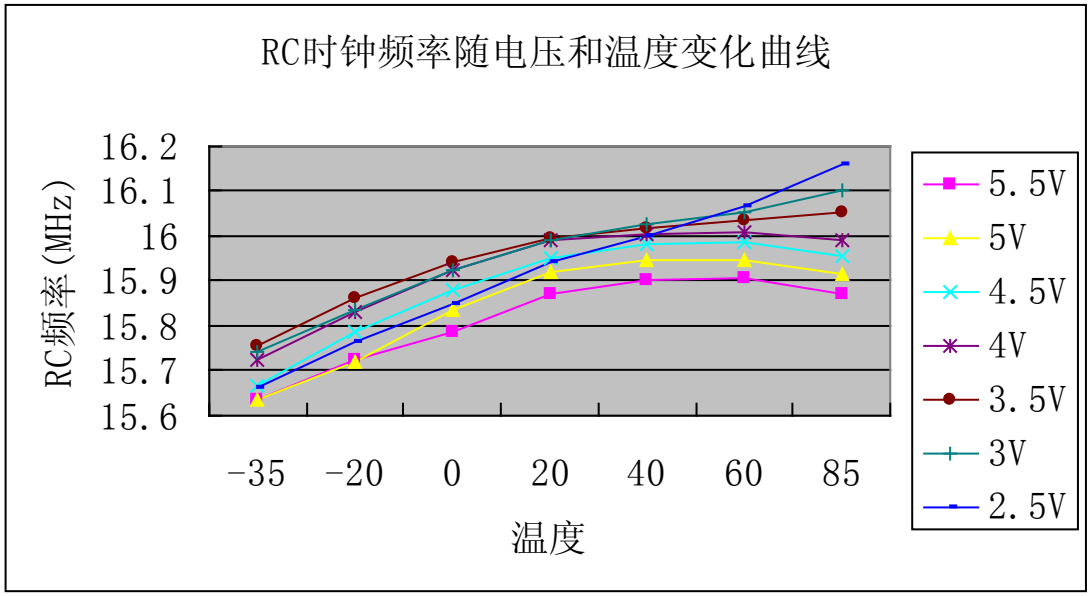


图18 RC 时钟频率的电压和温度特性

5.6 WDT时钟频率特性

下图位实际芯片的测试数据，不同芯片会略微有所差异，仅供参考。

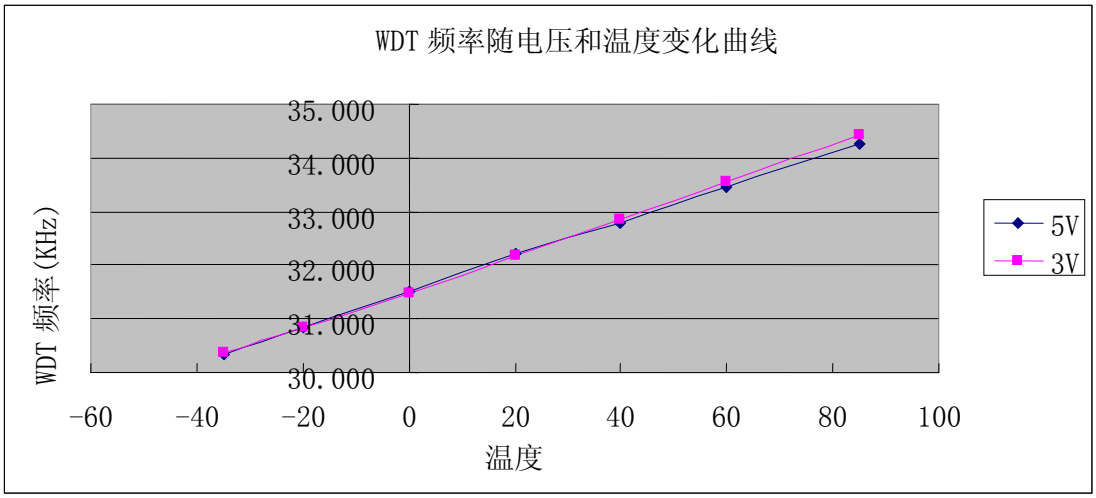


图19 WDT 频率的电压和温度特性

5.7 ERC频率特性

下表是实际芯片的测试数据，不同芯片会略微有所差异，仅供参考。

表 39 不同阻容的 ERC 频率

电容（pF）	电阻（Kohm）	电压（V）	晶振频率（kHz）	电流（uA） Fcpu=Fosc/2
47	1	5	8380	7026
		3	7431	3836
	10	5	3653	2311

		3	2262	953
	100	5	553	331
		3	307	129
	1000	5	67	49
		3	40	23
	10000	5	9	15
		3	5	10
100	1	5	8584	8266
		3	7273	4336
	10	5	2703	2178
		3	1575	835
	100	5	357	289
		3	214	115
	1000	5	51	50
		3	31	23
	10000	5	8	18
		3	4	11

下图是实际芯片的测试数据，不同芯片会略微有所差异，仅供参考。

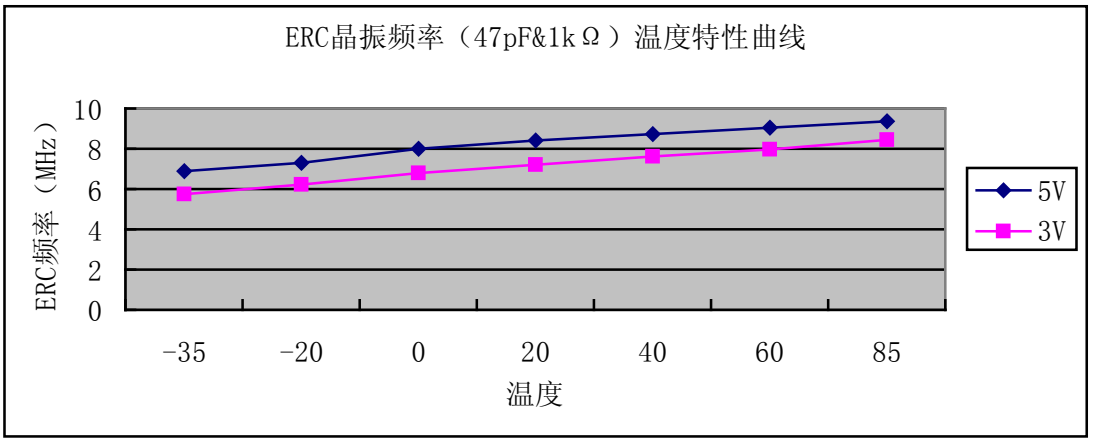
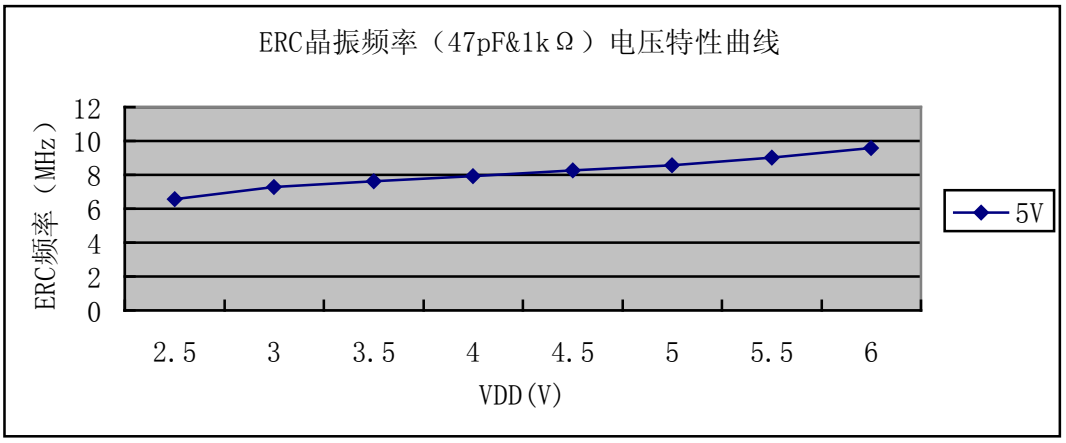


图20　ERC@(47pF&1kΩ)温度特性



5.8　2.0V掉电复位温度特性

下图位实际芯片的测试数据，不同芯片会略微有所差异，仅供参考。

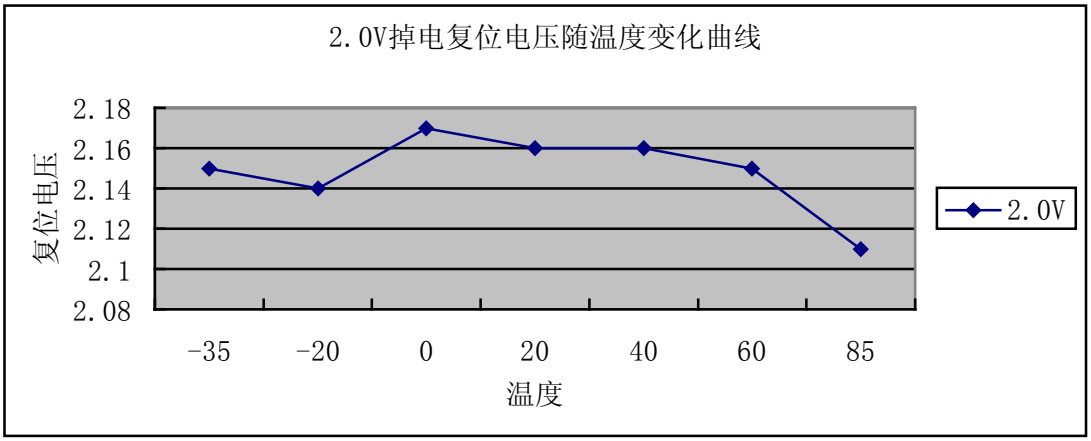


图21　2.0V 掉电复位温度特性

5.9　2.4V低电压复位温度特性

下图位实际芯片的测试数据，不同芯片会略微有所差异，仅供参考。

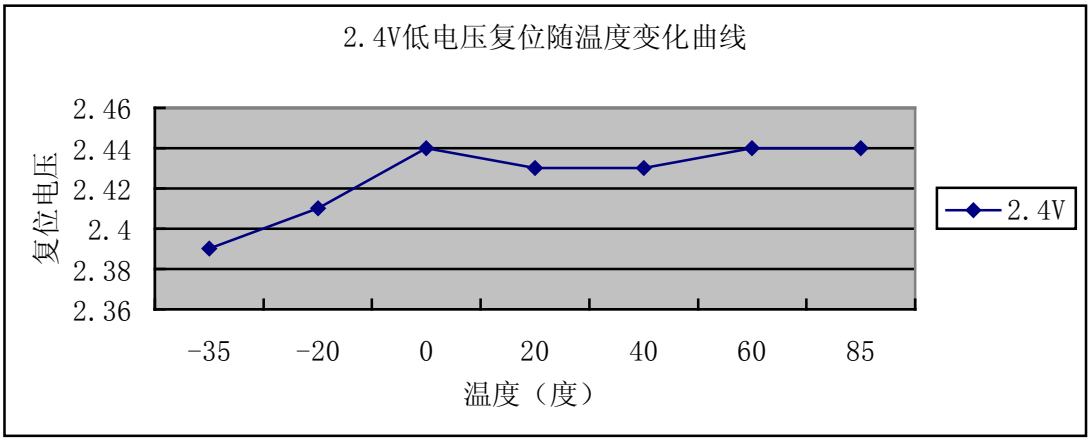


图22　2.4V 低电压复位温度特性

5.10　1.40V内部参考电压温度特性

下图位实际芯片的测试数据，不同芯片会略微有所差异，仅供参考。

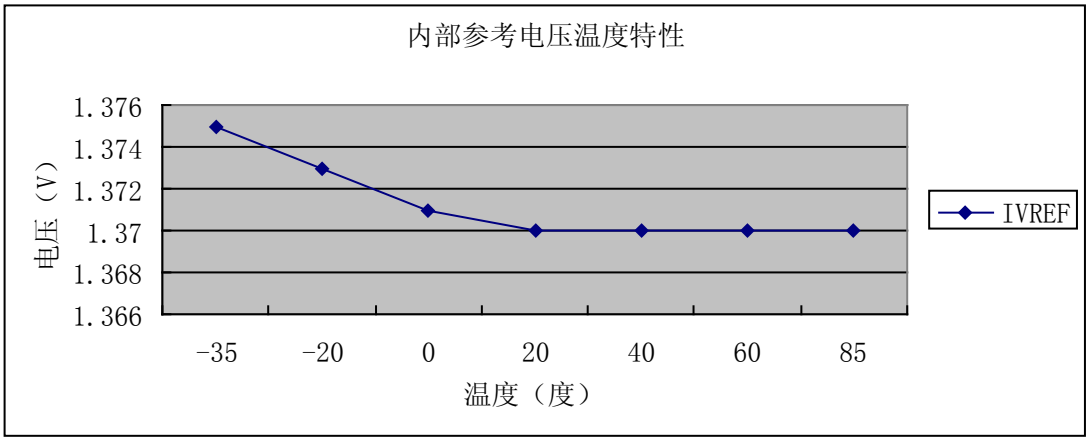
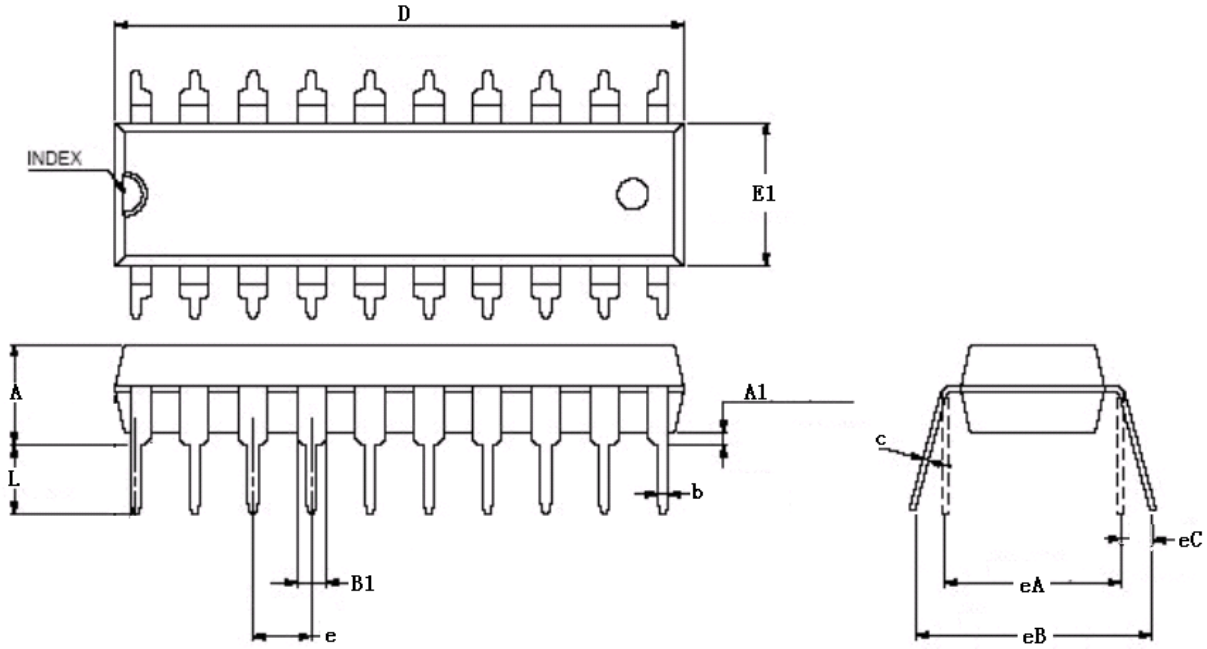


图23　内置参考电压温度特性

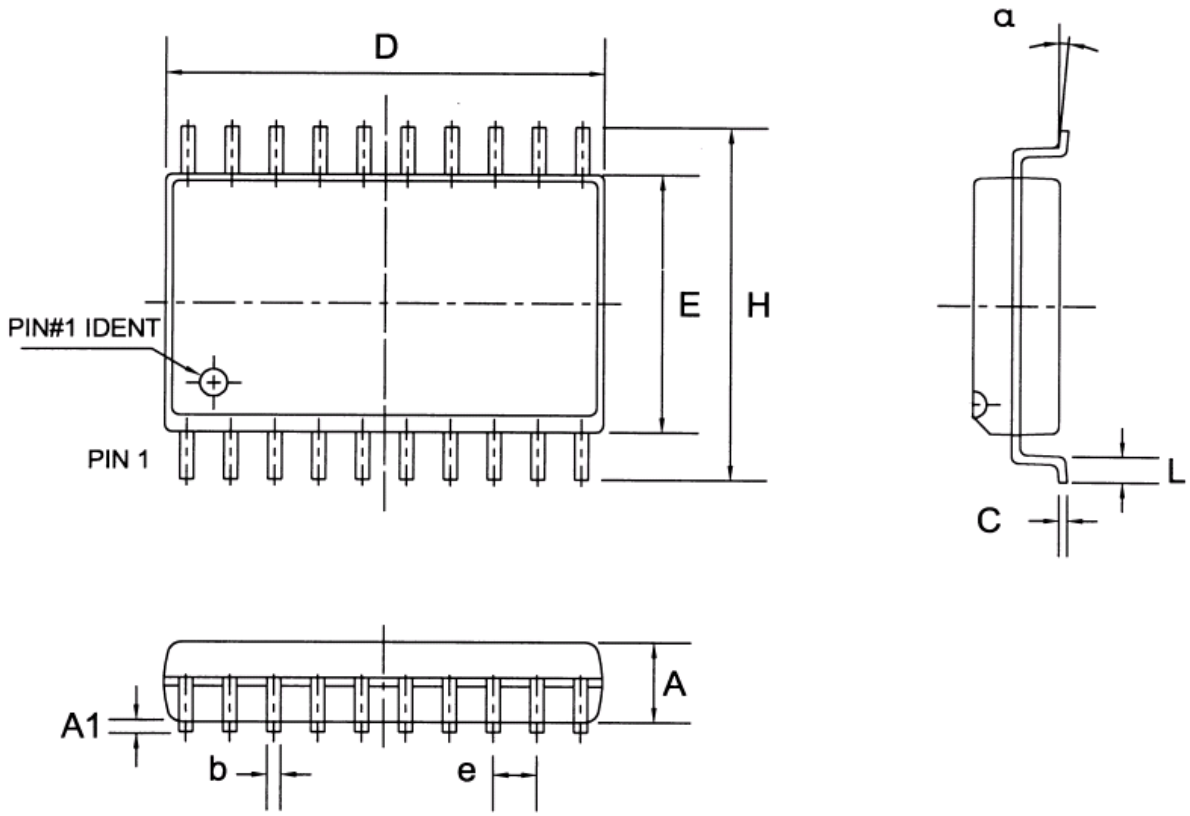
6 封装图

6.1 DIP-20pin



SYMBOLS	MIN	NOR	MAX
	(mm)		
A	3.6	3.8	4.0
A1	0.51	-	-
b	0.44	-	0.53
B1	1.52BSC		
c	0.25	-	0.31
D	26.03	26.23	26.43
E1	6.35	6.55	6.75
e	2.54BSC		
eA	7.62BSC		
eB	7.62	-	9.30
eC	0	-	0.84
L	3.0	-	-

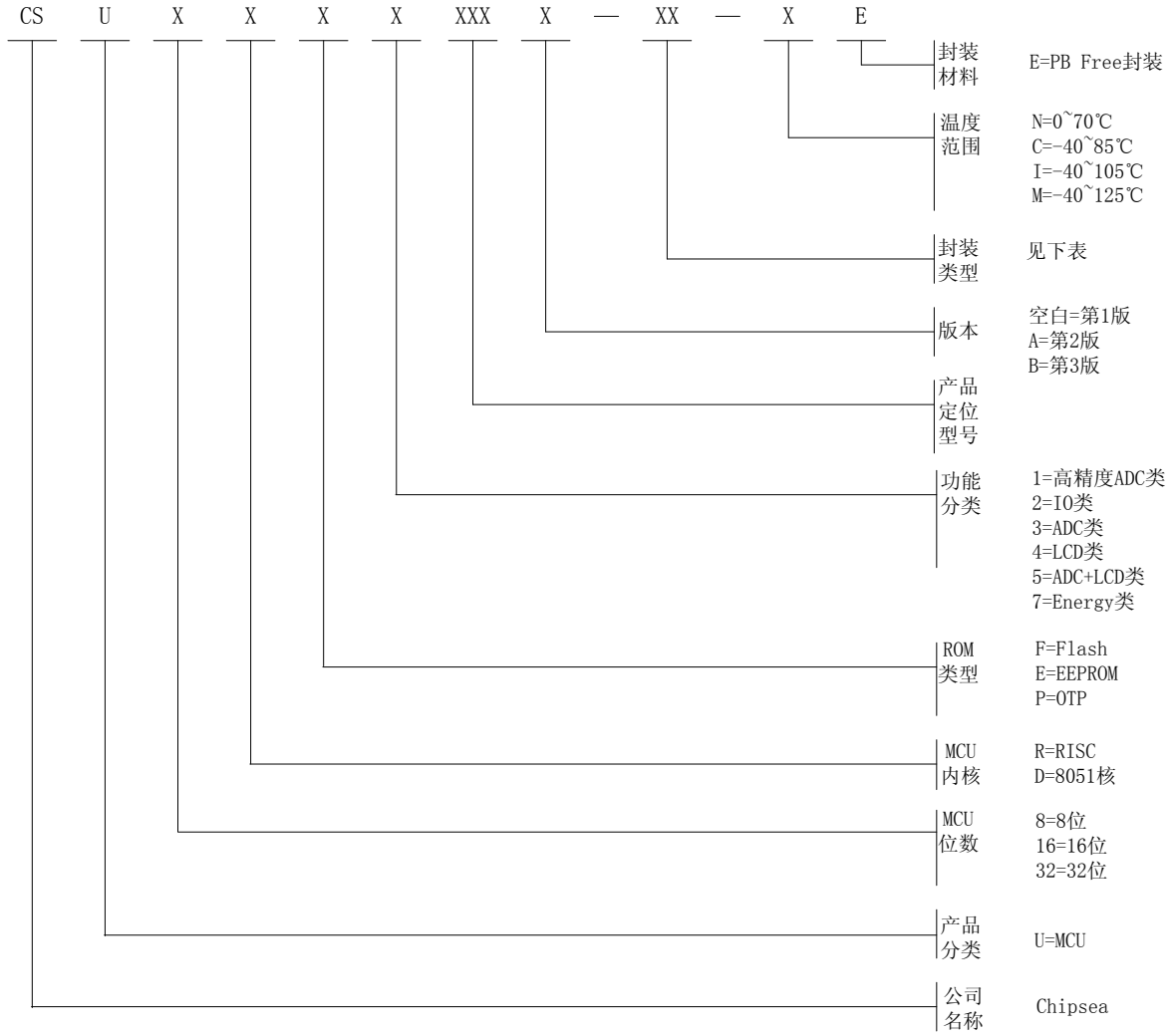
6.2 SOP-20pin



SYMBOLS	MIN	NOR	MAX
	(mm)		
A	2.25	2.30	2.35
A1	0.1	-	0.3
b	0.35	-	0.44
C	0.26	-	0.31
D	12.6	12.8	13.0
E	7.3	7.5	7.7
e		1.27BSC	
α	0°	-	8°
H	10.1	10.3	10.5
L	0.7	-	1.0

7 单片机产品命名规则

7.1 产品型号说明

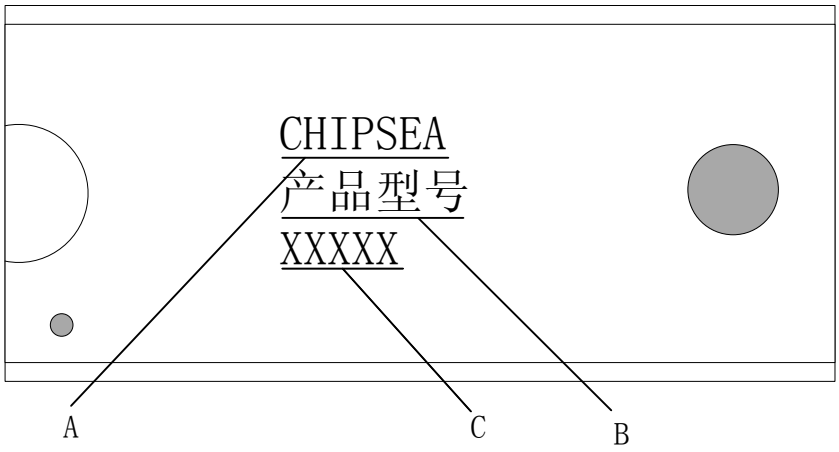


标示符	封装类型
BD	Bonding
DI	DIP
SD	SDIP
SO	SOP
SS	SSOP
TS	TSSOP
QF	QFP
LQ	LQFP
TQ	TQFP
QN	QFN

7.2 命名举例说明

名称	内核	ROM 类型	功能分类	产品定位型号	芯片版本	封装形式	工作温度范围	封装材料
CSU8RF3224-SO-CE	8 位 Risc MCU	Flash	ADC	224	第 1 版	SOP	-40~85 ℃	无铅封装(PB-Free 封装)

7.3 产品印字说明



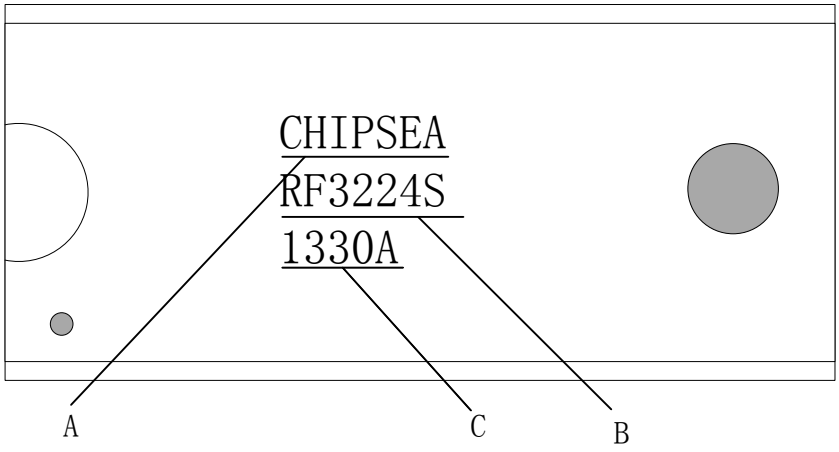
芯片正面印字一般有 3 行：

第一行为公司名称，为 CHIPSEA。

第二行为产品型号。对于一些小尺寸封装，会对产品型号进行缩减。

第三行为日期码。从左端起算，前两位为公历年号后两位；第三第四位为本年度日历周数，不足两位时左端补 0；最后一位为产品随机号。

例如，CSU8RF3224 的印字如下：



注：”-SO”会缩减为”S”，”-DI”会缩减为”D”，如 CSU8RF3224-SO-CE 的产品型号印字为 RF3224S。