

PMS151
1KW OTP IO
类型单片机

数据手册

第0.01版

2020年1月01日

PMS151

1KW OTP IO型单片机

目 录

目录

修订历史:	4
1. 功能	5
1.1. 特性	5
1.2. 系统特性	5
1.3. CPU 特性	5
1.4. 封装信息	5
2. 系统概述和方框图	6
3. 引脚功能说明	7
4. 器件电气特性	11
4.1. 直流交流电气特性	11
4.2. 绝对最大值范围	12
4.3. IHRC 频率与VDD 关系曲线图（校准到16MHz）	12
4.4. ILRC 频率与VDD 关系曲线图	13
4.5. IHRC 频率与温度关系曲线图（校准到16MHz）	13
4.6. ILRC 频率与温度关系曲线图	14
4.7. 工作电流 vs. VDD 与系统时钟 = IHRC/n 关系曲线图	14
4.8. 工作电流 vs. VDD 与系统时钟 = ILRC/n 关系曲线图	15
4.9. IO 引脚上拉阻抗曲线图	15
4.10. IO 引脚输出的驱动电流(I _{OH})与灌电流(I _{OL})曲线图	16
4.11. IO 引脚输入高/低阈值电压(V _{IH} /V _{IL})曲线图	17
4.12. 掉电电流(I _{PD})和省电电流(I _{PS})	18
5. 功能概述	19
5.1. 程序存储器 – OTP	19
5.2. 启动程序	19
5.3. 数据存储器 – SRAM	20
5.4. 振荡器和时钟	20
5.5. 16 位计数器 (Timer16)	24
5.6. 看门狗计数器	25
5.7. 中断	26
5.8. 省电和掉电	28
5.8.1. 省电模式 (“stopexe”)	28
5.8.2. 掉电模式 (“stopsys”)	29
5.8.3. 唤醒	29
5.9. T-type Key Scan 唤醒器	30
5.10. IO 引脚	31
5.11. 复位	32
5.12. 8 位IR Carrier 频率生成器 (IRTCMC / IRTMB)	33
6. IO 寄存器	35
6.1. ACC 状态标志寄存器 (<i>flag</i>), IO 地址 =0'h00	35
6.2. 堆栈指针寄存器 (<i>sp</i>), IO 地址 =0x02	35
6.3. 时钟模式寄存器 (<i>clkmd</i>), IO 地址 =0x03	35
6.4. 中断允许寄存器 (<i>inten</i>), IO 地址 =0x04	35
6.5. 中断请求寄存器 (<i>intrq</i>), IO 地址 =0x05	36
6.6. Timer16 控制寄存器 (<i>t16m</i>), IO 地址 =0x06	36
6.7. 杂项寄存器 (<i>misc</i>), IO 地址 = 0x1b	37
6.8. 中断边缘选择寄存器 (<i>integs</i>), IO 地址 =0x0c	37
6.9. 端口A 数字输入使能寄存器 (<i>padier</i>), IO 地址 =0x0d	37

PMS151

1KW OTP IO型单片机

6.10. 端口B 数字输入使能寄存器 (<i>pbdier</i>), IO 地址 =0x0e.....	38
6.11. 端口A 数据寄存器 (<i>pa</i>), IO 地址 =0x10.....	38
6.12. 端口A 控制寄存器 (<i>pac</i>), IO 地址 =0x11.....	38
6.13. 端口A 上拉控制寄存器 (<i>paph</i>), IO 地址 =0x12.....	38
6.14. 端口B 数据寄存器 (<i>pb</i>), IO 地址 =0x14.....	38
6.15. 端口B 控制寄存器 (<i>pbc</i>), IO 地址 =0x15.....	39
6.16. 端口B 上拉控制寄存器 (<i>pbph</i>), IO 地址 =0x16.....	39
6.17. 端口A T-scan 选择寄存器 (<i>pats</i>), IO 地址 =0x13.....	39
6.18. 端口B T-scan 选择寄存器 (<i>pbts</i>), IO 地址 =0x17.....	39
6.19. T-scan 控制寄存器 (<i>tscnc</i>), IO 地址 =0x1E.....	39
6.20. IR Timer 控制寄存器 (<i>irtmc</i>), IO 地址 =0x1C.....	40
6.21. IRTimer 上限寄存器 (<i>irtmb</i>), IO 地址 =0x1D.....	40
7. 指令.....	41
7.1. 数据传输类指令.....	42
7.2. 算数运算类指令.....	45
7.3. 移位运算类指令.....	47
7.4. 逻辑运算类指令.....	48
7.5. 位运算类指令.....	50
7.6. 条件运算类指令.....	50
7.7. 系统控制类指令.....	51
7.8. 指令执行周期综述.....	53
7.9. 指令影响标志综述.....	53
7.10. 位定义.....	53
8. 程序选项.....	54
9. 特别注意事项.....	54
9.1. 使用IC.....	54
9.1.1. IO 引脚的使用和设定.....	54
9.1.2. 中断.....	55
9.1.3. 系统时钟选择.....	55
9.1.4. 掉电模式、唤醒和看门狗.....	55
9.1.5. TIMER 溢出.....	55
9.1.6. IHRC.....	56
9.1.7. PMS151 的烧录方法.....	56
9.1.8. PMS151 应用参考原理图.....	56
9.2 使用 ICE.....	57

PMS151

1KW OTP IO型单片机

修订历史:

修订	日期	描述
0.00	2019/06/11	初版
0.01	2019/07/01	1. 删去部分敏感字眼及图示 2. 删去重要声明部分 3. 删去原 9.1 节：警告 4. 修改第 9.1.6 节 5. 修改第 9.1.7 节 6. 修改第 9.1.8 节

PMS151

1KW OTP IO型单片机

1. 功能

1.1. 特性

- ◆ 通用系列
- ◆ 请勿使用于 AC 阻容降压供电，强电源纹波，或高 EFT 要求之应用
- ◆ 工作温度范围：-20°C ~ 70°C

1.2. 系统特性

- ◆ 1KW OTP 程序存储器
- ◆ 32 字节数据存储器
- ◆ 一个硬件 16 位计数器
- ◆ 一组 Key Scan 硬件唤醒器
- ◆ 13 个 IO 引脚并带有上拉电阻选项
- ◆ 时钟源：内部高频 RC 振荡器，内部低频 RC 振荡器

1.3. CPU 特性

- ◆ 单一处理单元工作模式
- ◆ 提供 79 个有效指令
- ◆ 大部分都是 1T（单周期）指令
- ◆ 可程序设定的堆栈指针和堆栈深度（使用 2 bytes SRAM 作为一层堆栈）
- ◆ 数据存取支持直接和间接寻址模式，用数据存储器即可当作间接寻址模式的数据指针(index pointer)
- ◆ IO 地址以及存储地址空间互相独立

1.4. 封装信息

- ◆ PMS151-S08B: SOP8 (150mil)
- ◆ PMS151-S16D: SOP16D (150mil)

PMS151

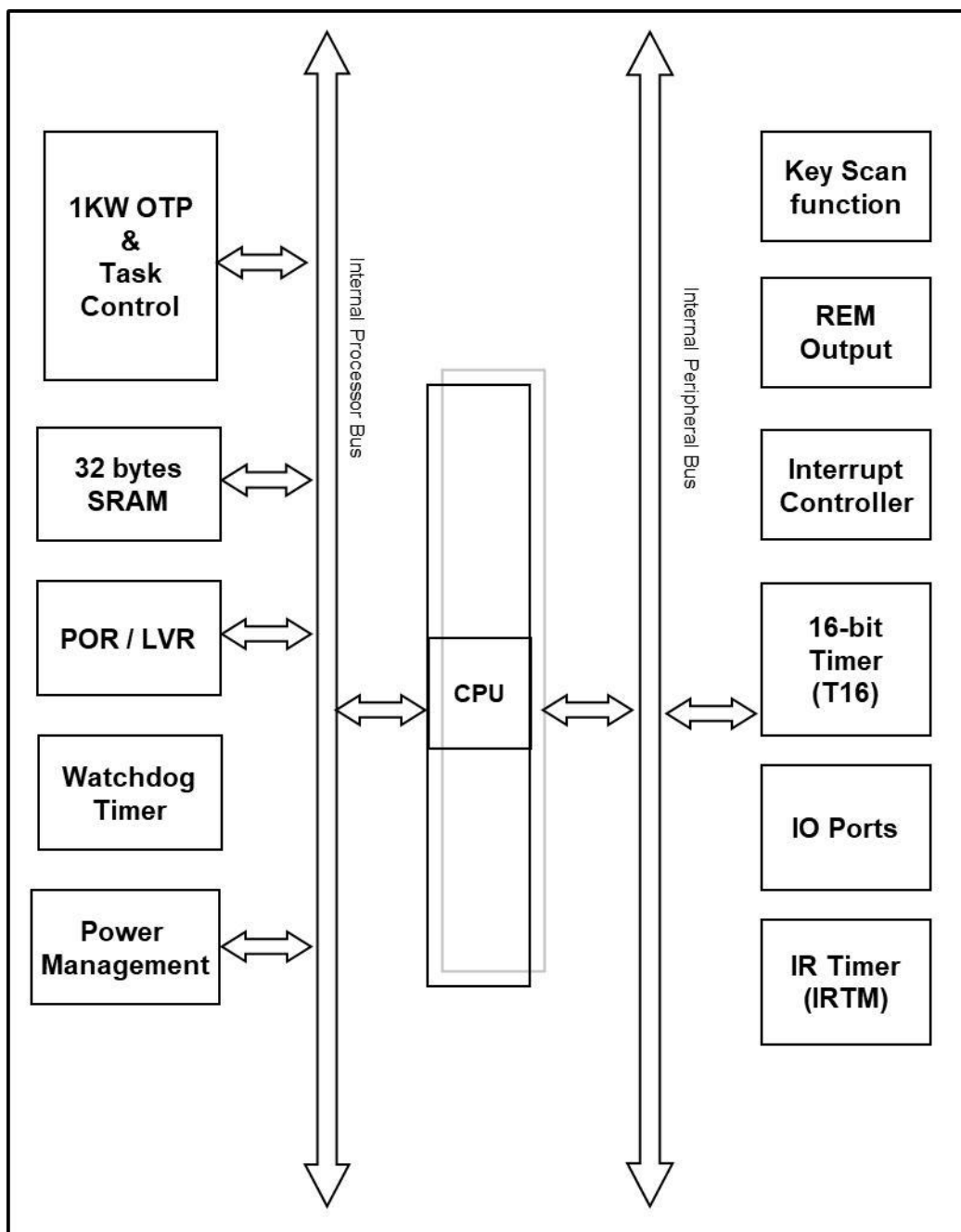
1KW OTP IO型单片机

2. 系统概述和方框图

PMS151 系列是一款完全静态的，以 OTP 为程序基础的 CMOS 8-bit 微处理器。它运用 RISC 的架构并且所有的指令架构的执行周期都是一个指令周期，只有少部分指令需要两个指令周期。

PMS151 是 1KW OTP 程序存储器以及 32 字节数据存储器，一个 16 位的硬件计数器。

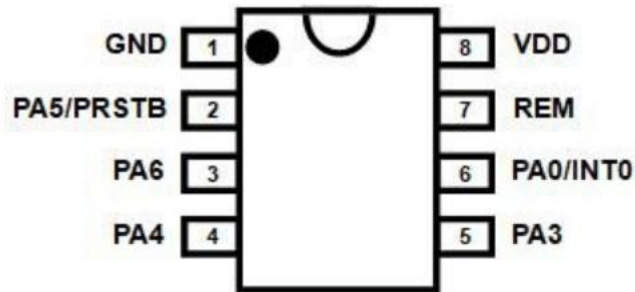
PMS151 有一个 REM 输出用于 IRTX 输出。IR Carrier 频率生成器也支持 8 位 IR 定时器。此外，它同时支持 T-type 和 Matrix key scan 唤醒功能。



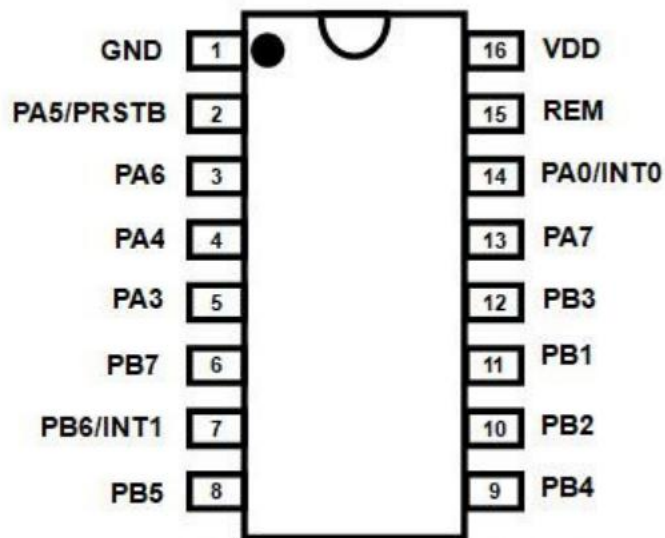
PMS151

1KW OTP IO型单片机

3. 引脚功能说明



PMS151-S08(SOP8-150mil)



PMS151-S16D(SOP16D-150mil)

PMS151

1KW OTP IO型单片机

引脚名称	引脚 & 缓存器类型	描述
PA7	IO ST / CMOS	此引脚可以用作： 端口 A 位 7。可编程设定为输入或输出，弱上拉电阻模式。
PA6	IO ST / CMOS	此引脚可以用作： 端口 A 位 6。可编程设定为输入或输出，弱上拉电阻模式。
PA5 / PRSTB	IO ST / CMOS	此引脚可以用作： (1) 端口 A 位 5。此引脚可以设定为输入或开漏输出(open drain)，弱上拉电阻模式。 (2) 外部复位引脚。
PA4	IO ST / CMOS	此引脚可以用作： 端口 A 位 4。可编程设定为输入或输出，弱上拉电阻模式。
PA3	IO ST / CMOS	此引脚可以用作： 端口 A 位 3。可编程设定为输入或输出，弱上拉电阻模式。
PA0 / INT0	IO ST / CMOS	此引脚可以用作： (1) 端口 A 位 0。可编程设定为输入或输出，弱上拉电阻模式。 (2) 外部中断源 0。上升沿和下降沿都可触发中断。
REM	Output	IRTX 输出为 IR LED 提供电流。
PB1	IO ST / CMOS	此引脚可以用作： 端口 B 位 1。可编程设定为输入或输出，弱上拉电阻模式。
PB2	IO ST / CMOS	此引脚可以用作： 端口 B 位 2。可编程设定为输入或输出，弱上拉电阻模式。
PB3	IO ST / CMOS	此引脚可以用作： 端口 B 位 3。可编程设定为输入或输出，弱上拉电阻模式。
PB4	IO ST / CMOS	此引脚可以用作： 端口 B 位 4。可编程设定为输入或输出，弱上拉电阻模式。
PB5	IO ST / CMOS	此引脚可以用作： 端口 B 位 5。可编程设定为输入或输出，弱上拉电阻模式。

PMS151

1KW OTP IO型单片机

引脚名称	引脚 & 缓存器类型	描述
PB6 / INT1	IO ST / CMOS	此引脚可以用作： (1) 端口 B 位 6。可编程设定为输入或输出，弱上拉电阻模式。 (2) 外部中断源 1。 <u>上升沿和下降沿都可触发中断</u> 。
PB7	IO ST / CMOS	此引脚可以用作： 端口 B 位 7。可编程设定为输入或输出，弱上拉电阻模式。
PA7	IO ST / CMOS	此引脚可以用作： 端口 A 位 7。可编程设定为输入或输出，弱上拉电阻模式。
VDD		正电源
GND		地
注意： IO：输入/输出；ST：施密特触发器输入；CMOS：CMOS 电压基准位		

PMS151

1KW OTP IO型单片机

4. 器件电气特性

4.1. 直流交流电气特性

符号	描述	最小值	典型值	最大值	单位	条件
V_{DD}	工作电压	1.8*		3.6	V	* 受限于 LVR 的精度
V_{LVR}	LVR 电压		1.6 1.7 1.8	1.8 1.9 2.0	V	$T_a = 25^{\circ}\text{C}$ $0^{\circ}\text{C} < T_a < 70^{\circ}\text{C}$ $-20^{\circ}\text{C} < T_a < 0^{\circ}\text{C}$
I_{OP}	工作电流		0.5 5		mA uA	$f_{SYS}=IHRC/16=1MIPS@3V$ $f_{SYS}=ILRC=36KHz@3V$
I_{PD}	掉电电流 (通过 stopsys 命令设置)		1		uA	$V_{DD}=3V$
I_{PS}	省电电流 (通过 stopexe 命令设置 *停用 IHRC)		1		uA	$V_{DD}=3V$
V_{IL}	输入低电压	0		$0.1 V_{DD}$	V	
V_{IH}	输入高电压	$0.8 V_{DD}$ $0.7 V_{DD}$		V_{DD} V_{DD}	V	PA5 其 他 IO 口
I_{OL}	IO 输出灌电流 REM Others		220 20		mA	$V_{DD}=3V, V_{OL}=0.3V$
I_{OH}	IO 输出驱动电流 PA5/REM Others		0 10		mA	$V_{DD}=3V, V_{OH}=2.7V$
V_{IN}	输入电压	-0.3		$V_{DD}+0.3$	V	
$I_{INJ} (PIN)$	引脚输入电流		1		uA	$V_{DD} + 0.3 \geq V_{IN} \geq -0.3$
R_{PH}	上拉电阻		120		K Ω	$V_{DD}=3V$
f_{IHRC}	校准后 IHRC 频率 *	15.76*	16*	16.24*	MHz	$V_{DD}=1.8V\sim 3.6V$ $-20^{\circ}\text{C} < T_a < 70^{\circ}\text{C}$
f_{ILRC}	ILRC 频率 *		36		KHz	$V_{DD}=3V$
t_{INT}	中断脉冲宽度	30			ns	$V_{DD}=3V$
t_{WDT}	看门狗超时溢出时间		4k		ILRC clock period	misc[1:0]=00 (默认)
			8k			misc[1:0]=01
			32k			misc[1:0]=10
			128k			misc[1:0]=11
t_{SBP}	系统开机时间		40		ms	@ $V_{DD}=3V$
t_{RST}	外部复位脉冲宽度	120			us	@ $V_{DD}=3V$

* 这些参数是设计参考值，并不是每个芯片测试。

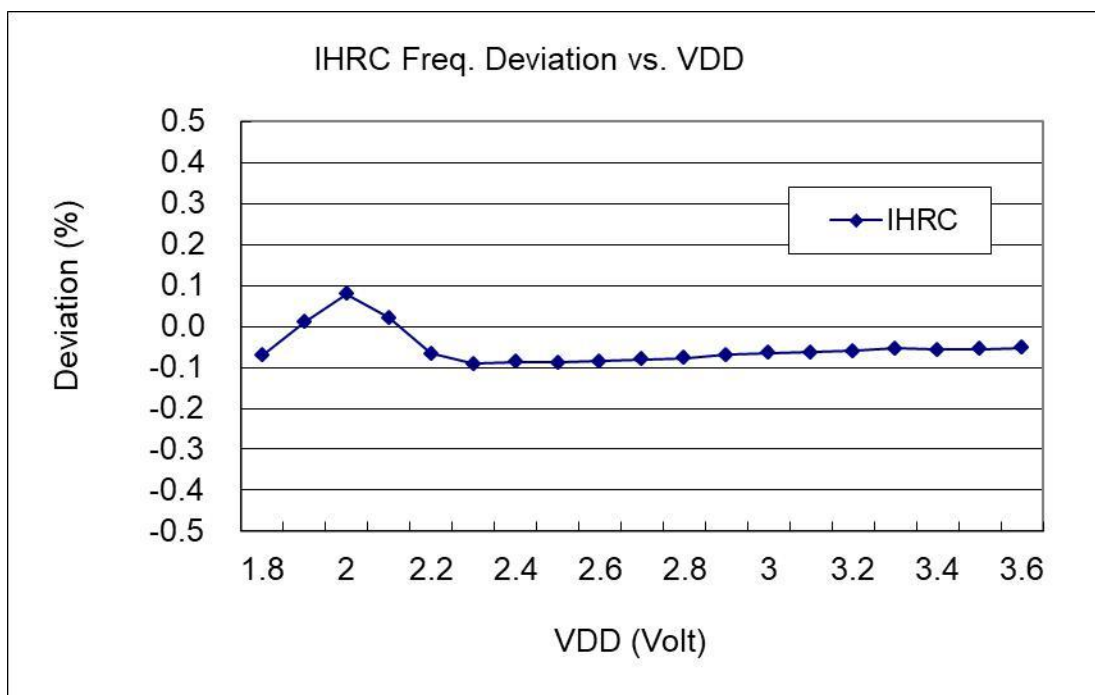
PMS151

1KW OTP IO型单片机

4.2. 绝对最大值范围

- 电源电压..... V ~ 3.6V
- 输入电压..... V ~ $V_{DD} + 0.3V$
- 工作温度 -20°C ~ 70°C
- 存储温度 -50°C ~ 125°C
- 节点温度..... 150°C

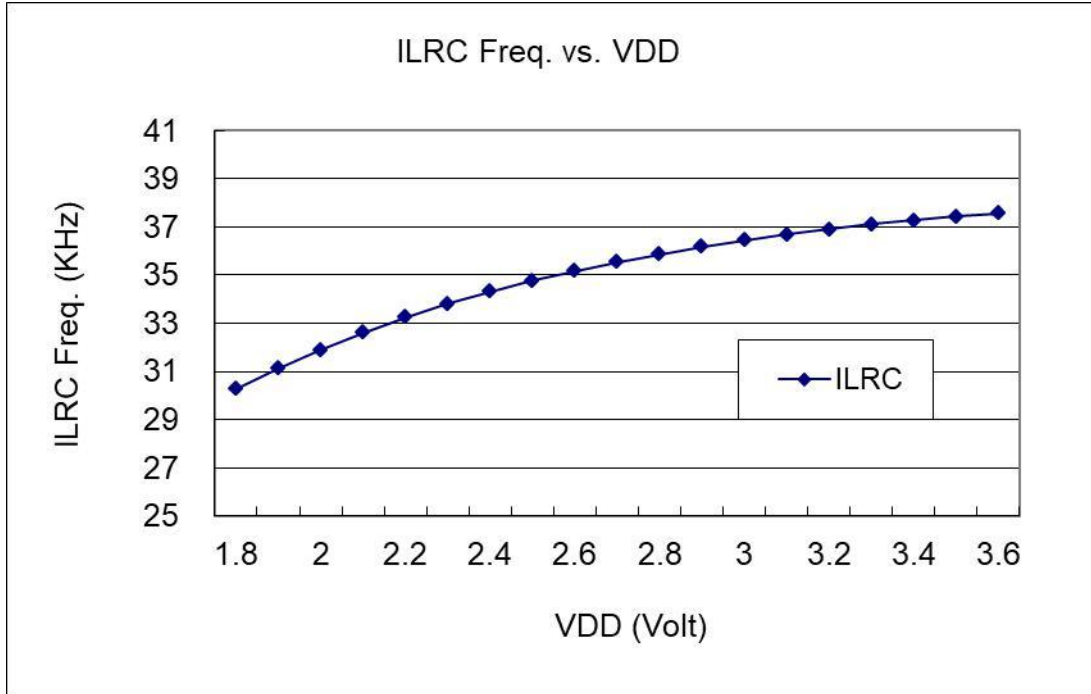
4.3. IHRC 频率与 VDD 关系曲线图（校准到 16MHz）



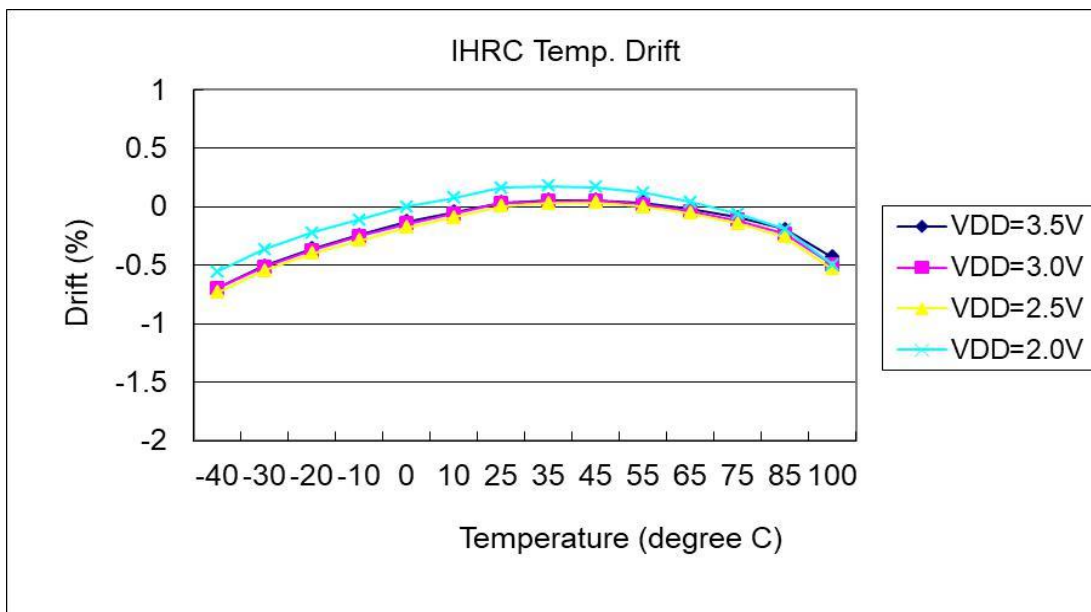
PMS151

1KW OTP IO型单片机

4.4. ILRC 频率与 VDD 关系曲线图



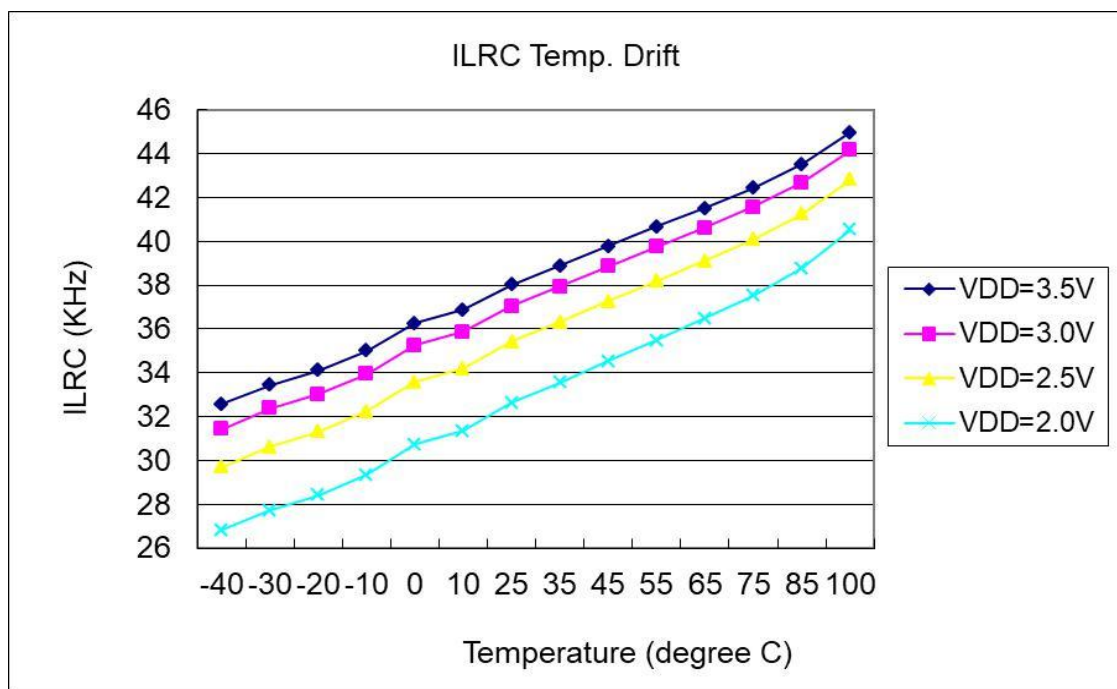
4.5. IHRC 频率与温度关系曲线图（校准到 16MHz）



PMS151

1KW OTP IO型单片机

4.6. ILRC 频率与温度关系曲线图

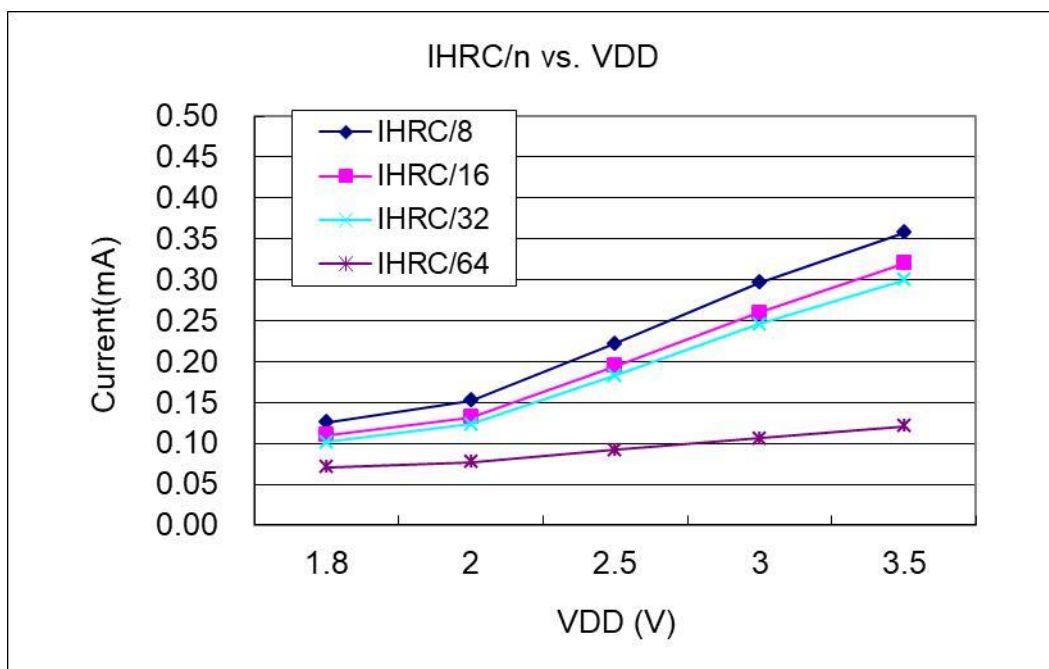


4.7. 工作电流 vs. VDD 与系统时钟 = IHRC/n 关系曲线图

➤ 测量条件:

pa0 间隔(1s)高低电平翻转。

启用: Band-gap, LVR, IHRC。停用: t16 定时器, 中断, ILRC, 且 IO 引脚不悬空。



PMS151

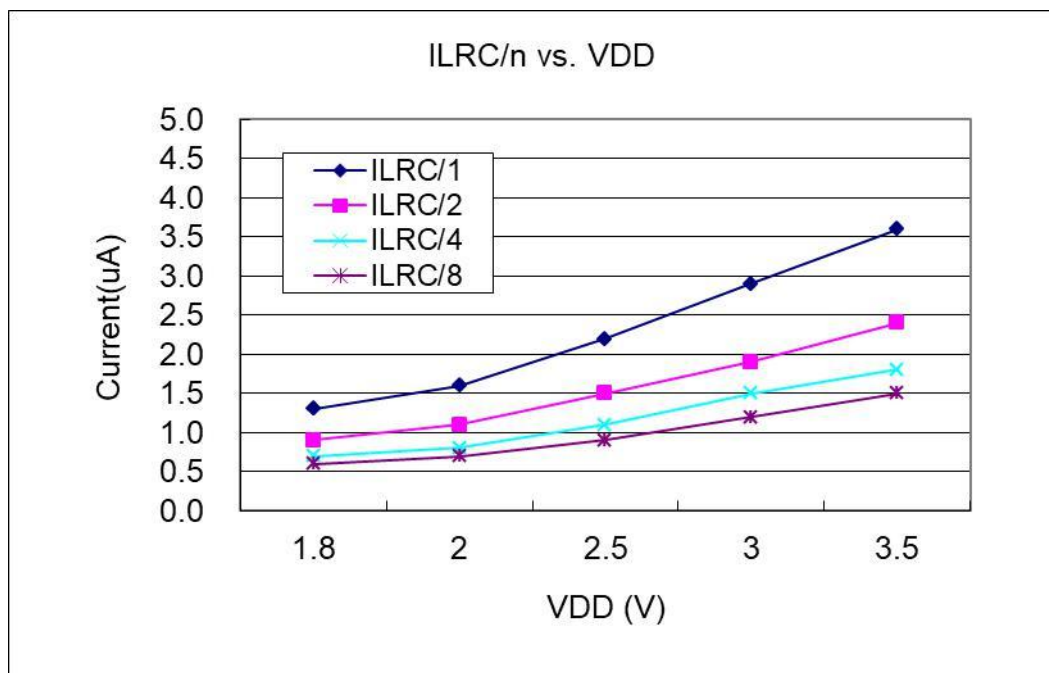
1KW OTP IO型单片机

4.8. 工作电流 vs. VDD 与系统时钟 = ILRC/n 关系曲线图

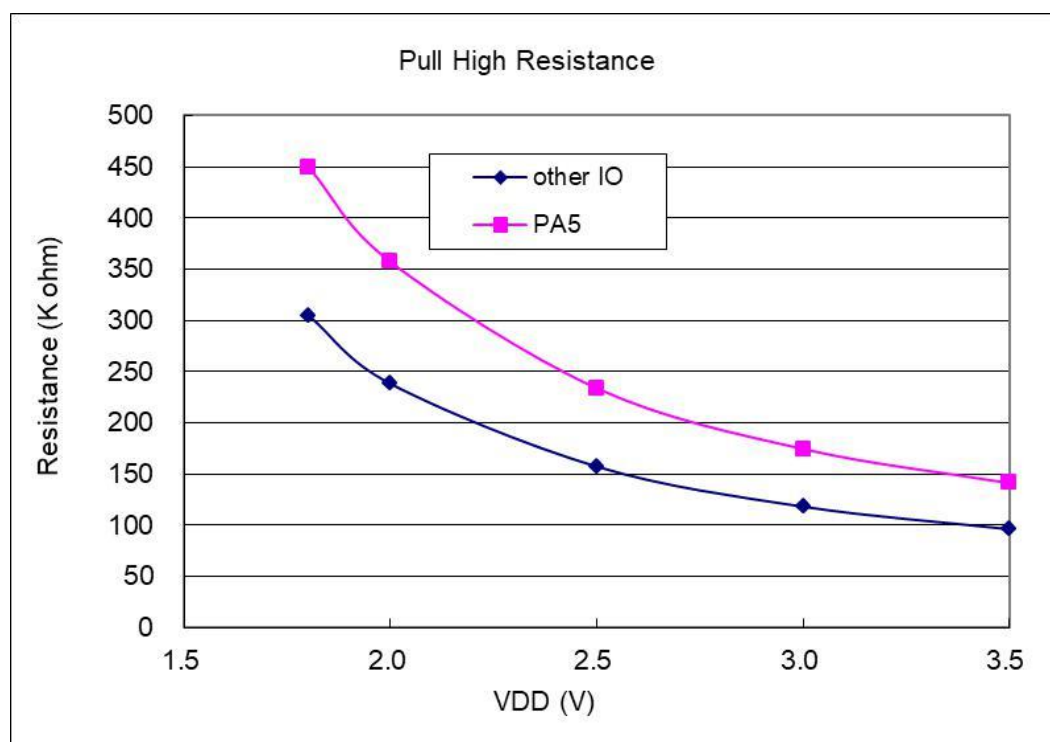
➤ 测量条件:

pa0 间隔(1s)高低电平翻转。

启用: Band-gap, LVR, ILRC。 停用: t16 定时器, 中断, IHRC, 且 IO 引脚不悬空。



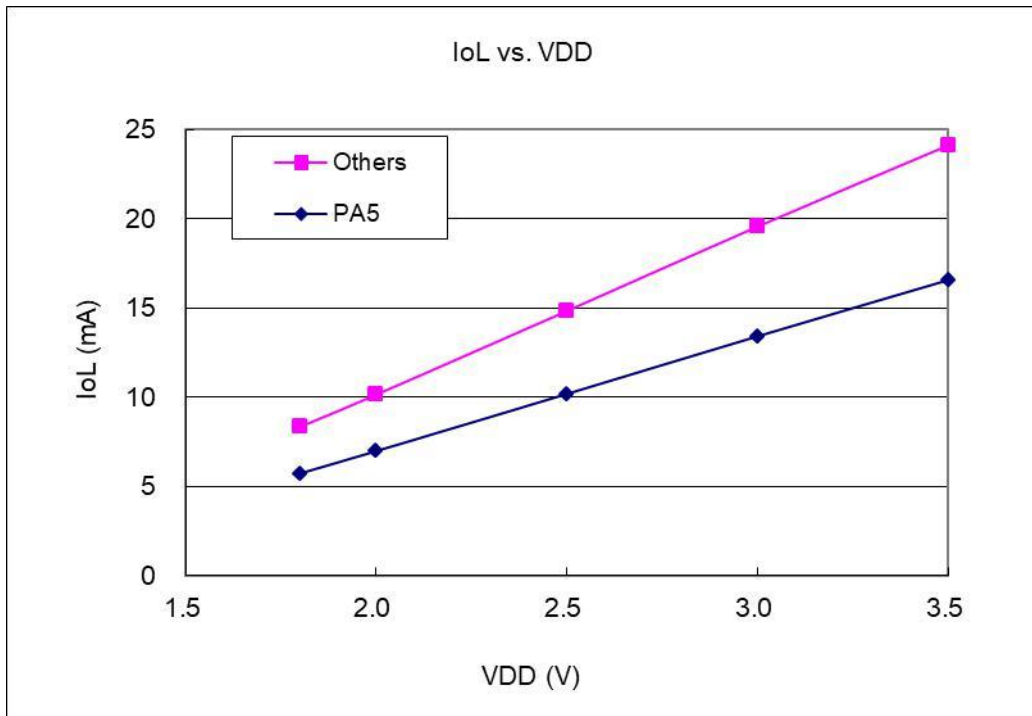
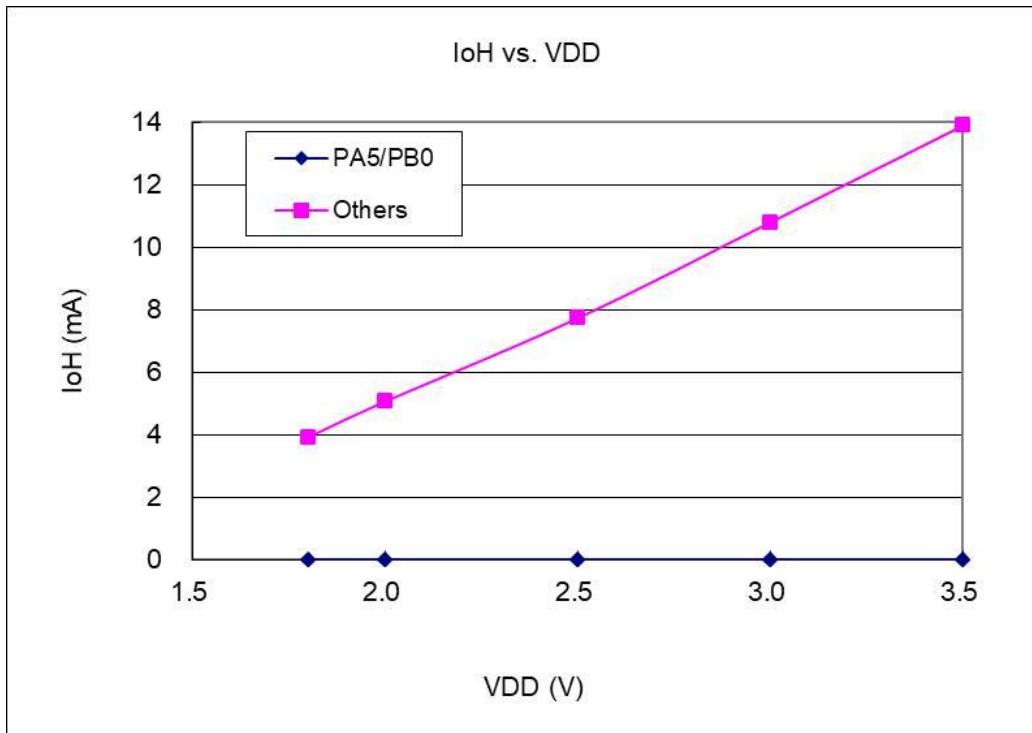
4.9. IO 引脚上拉阻抗曲线图



PMS151

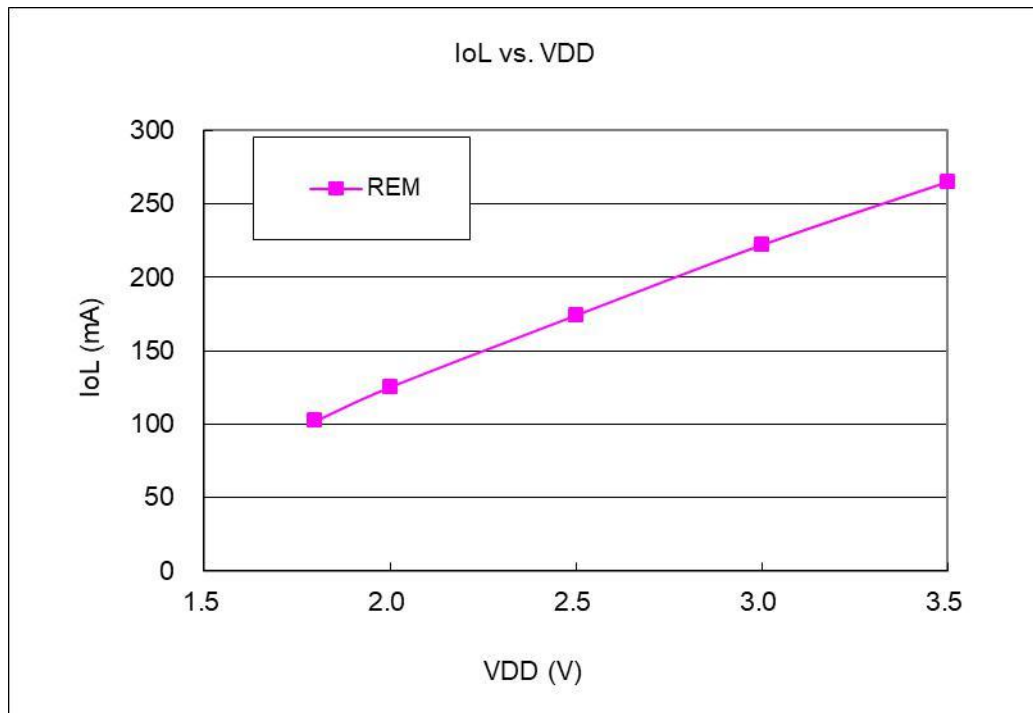
1KW OTP IO型单片机

4.10. IO 引脚输出的驱动电流(I_{OH})与灌电流(I_{OL})曲线图

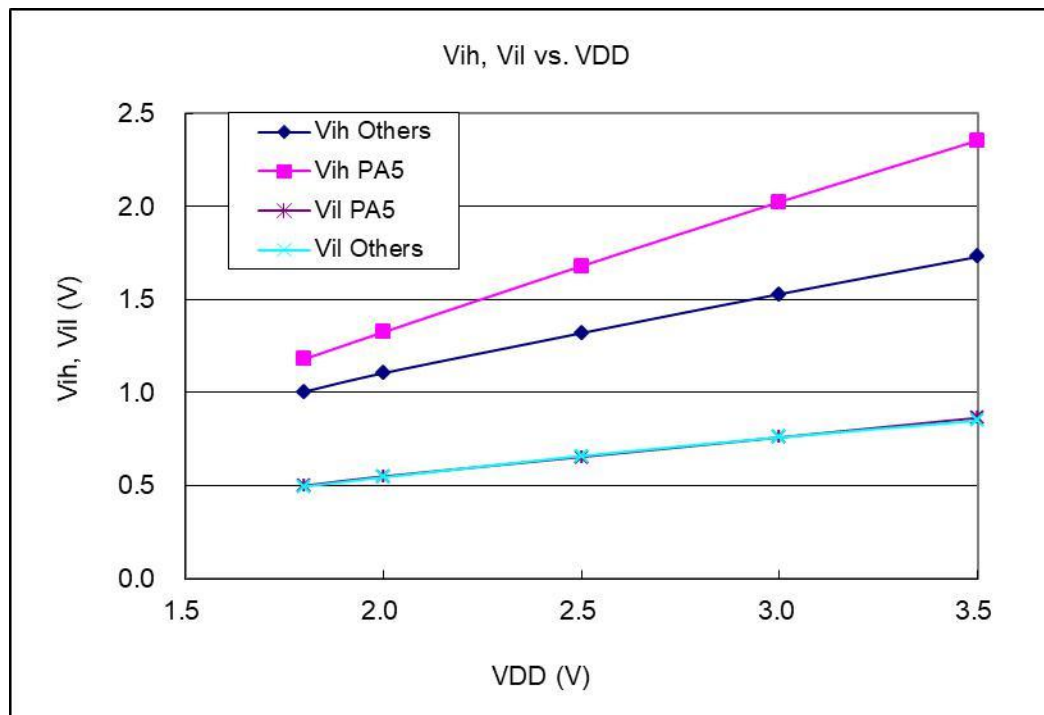


PMS151

1KW OTP IO型单片机



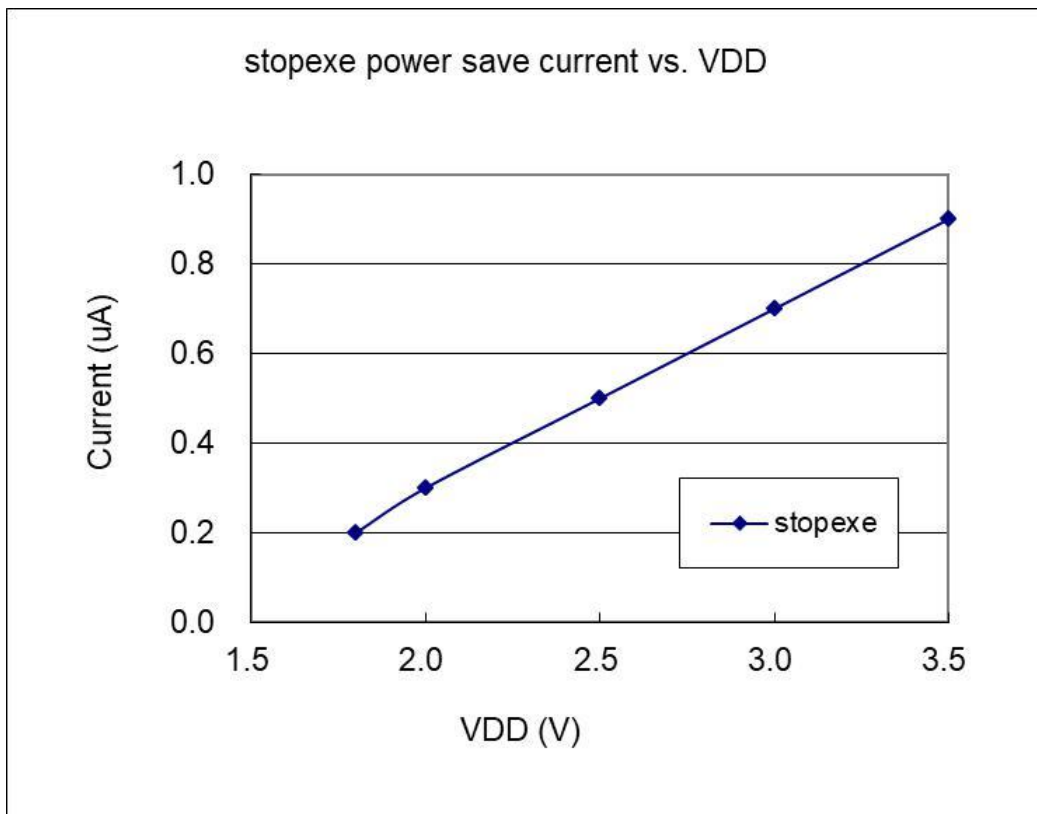
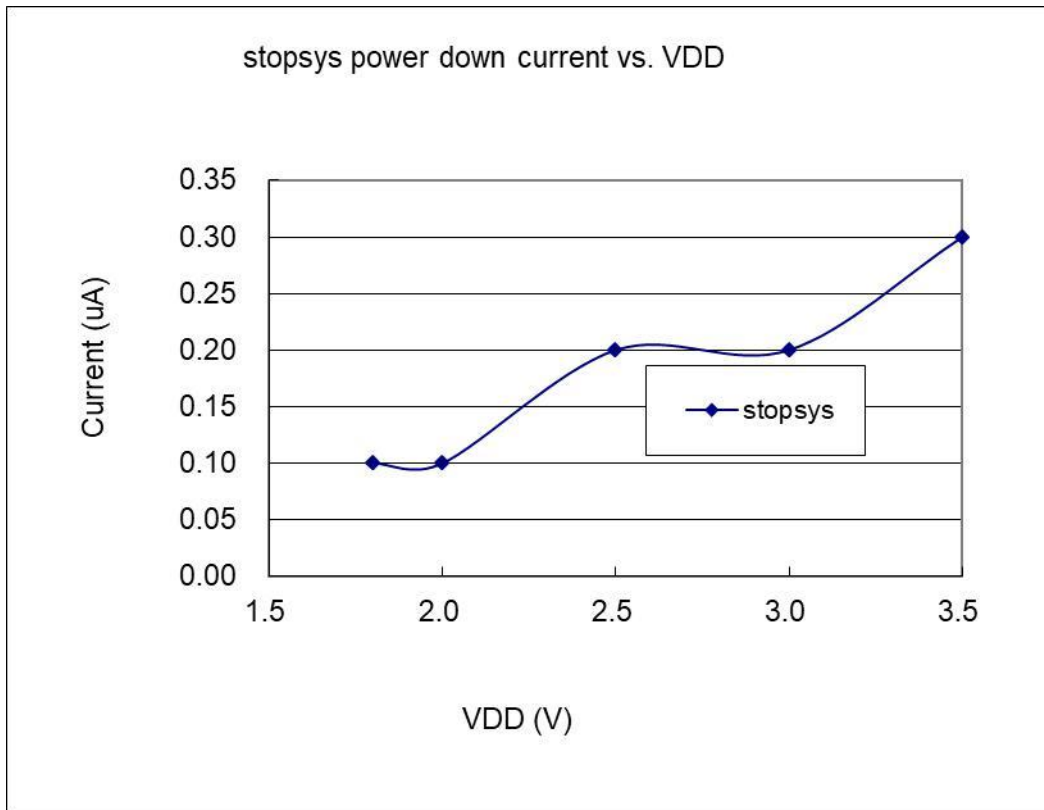
4.11. IO 引脚输入高/低阈值电压(V_{IH}/V_{IL})曲线图



PMS151

1KW OTP IO型单片机

4.12. 掉电电流(I_{PD})和省电电流(I_{PS})



PMS151

1KW OTP IO型单片机

5. 功能概述

5.1. 程序存储器 – OTP

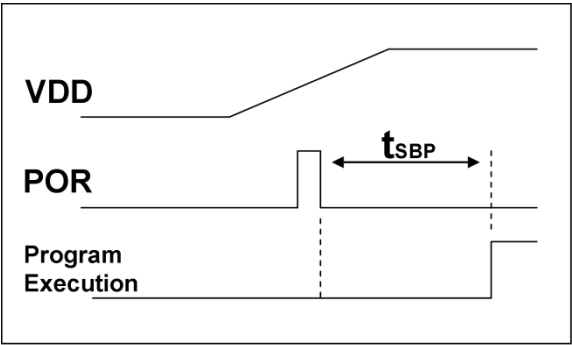
OTP（一次性可编程）程序存储器用来存放要执行的程序指令。OTP 程序存储器可以储存数据，包含：数据，表格和中断入口。复位之后，FPP0 的初始地址为 0x000 保留给系统使用，中断入口是 0x010。PMS151 的 OTP 程序存储器容量为 1KW 如表 1 所示。OTP 存储器从地址“0x3F0 ~0x3FF”供系统使用，从 0x001 到0x00F 和从 0x011 到 0x3EF 地址空间是用户的程序空间。

地址	功能
0x000	用于 FPP0 复位, goto 主程序
0x001	用户程序区
•	•
•	•
0x00F	用户程序区
0x010	中断入口地址
0x011	用户程序区
•	•
0x3EF	用户程序区
0x3F0	系统使用
•	•
0x3FF	系统使用

表 1: 程序存储器结构

5.2. 启动程序

开机时，POR（上电复位）是用于复位PMS151，开机时间约 1000 ILRC 时钟周期。用户在使用时，必须确保上电后电源电压稳定，开机时序如图 1 所示，其中 t_{SBP} 是开机时间。



Boot up from Power-On Reset

图 1: 开机时序图

PMS151

1KW OTP IO型单片机

5.3. 数据存储器 – SRAM

数据存储可以是字节或位操作。除了存储数据外，数据存储器还可以担任间接存取方式的数据指针，以及堆栈存储器。

堆栈定义在数据存储器里面，堆栈指针定义在堆栈指针寄存器，用户可在使用时自行定义堆栈深度，堆栈存储器对堆栈的排列是非常灵活的，用户可以动态调整堆栈。

对于间接存储指令而言，数据存储器可以用作数据指针来当作数据地址。所有的数据存储器都可以当作资料指针，这对于间接存储指令是相当灵活和有效的。由于数据宽度是 8 位，PMS151 的所有 32 字节的数据存储器都可以利用间接存取指令做存取。

5.4. 振荡器和时钟

PMS151 有两个振荡器电路：内部高频 RC 振荡器(IHRC) 和内部低频振荡器(ILRC)，这两个振荡器可以分别通过寄存器 `clkmd.4` 和 `clkmd.2` 来启用或停用。使用者可以选择不同的振荡器作为系统时钟源，同时可以通过设置 `clkmd` 寄存器来满足不同的应用要求。

在快开机模式，IHRC 默认为开启，在慢开机模式，IHRC 默认为关闭；但程序执行时，`.ADJUST_IC` 命令会依据使用者的 `SYSCLK` 选择，再调整 IHRC 模块的开关。

振荡器模块	启用/停用
IHRC	<code>clkmd.4</code>
ILRC	<code>clkmd.2</code>

表 2：振荡器模块

5.4.1. 内部高频 RC 振荡器和内部低频 RC 振荡器

开机后，IHRC 和 ILRC 振荡器是自动启用的。IHRC 频率能通过 `ihrcr` 寄存器校准，通常校准到 16 MHz。详细请参考 IHRC 与 V_{DD} 及温度关系曲线图。

ILRC 的频率会因生产工艺，使用的电源电压和温度的差异而产生漂移，请参考直流电气特性规格数据，建议不要应用在要求精准时序的产品上。

5.4.2. IHRC 校准

在芯片生产制造时，每颗芯片的 IHRC 频率都有可能稍微不同，PMS151 提供 IHRC 频率校准来消除这些差异，校准功能可以被用户的程序选择并编译，同时这个命令会自动嵌入用户的程序里面。

校准命令如下所示：

`.ADJUST_IC SYSCLK=IHRC/(p1), IHRC=(p2)MHz, VDD=(p3)`

p1= 8, 16, 32; 用以提供不同的系统时钟。

p2=14 ~ 18; 用以校准芯片到不同的频率，16MHz 是通用的选择。

p3=2.3 ~ 5.5; 用以在不同的工作电压下校准频率。

PMS151

1KW OTP IO型单片机

5.4.3. IHRC 频率校准和系统时钟

在用户编译程序时，IHRC 频率校准和系统时钟的选项如表 3 所示：

SYSClk	CLKMD	IHRCR	描述
○ Set IHRC / 8	= 34h (IHRC / 8)	有校准	IHRC 校准到 16MHz, CLK=2MHz (IHRC/8)
○ Set IHRC / 16	= 14h (IHRC / 16)	有校准	IHRC 校准到 16MHz, CLK=1MHz (IHRC/16)
○ Set IHRC / 32	= 74h (IHRC / 32)	有校准	IHRC 校准到 16MHz, CLK=0.5MHz (IHRC/32)
○ Set ILRC	= E4h (ILRC / 1)	有校准	IHRC 校准到 16MHz, CLK=ILRC
○ Disable	不改变	没改变	IHRC 不校准, CLK 不改变

表 3: IHRC 频率校准选项

通常，.ADJUST_IC 是开机后第一条指令，以便系统开机后能设定系统频，IHRC 频率校准仅在烧录 OTP 程序代码的时候执行一次，烧录之后不会重复执行了。如果用户选择了不同的频率校准选项，PMS151 的系统状态在开机后也会不同。以下所示为不同的选项开机后，PMS151 执行此命令后的状态：

(1) .ADJUST_IC SYSClk=IHRC/8, IHRC=16MHz, V_{DD}=2.5V

开机后，CLKMD = 0x34:

- ◆ IHRC 频率在 V_{DD}=2.5V 时校准到 16MHz，并且 IHRC 模块是启用的
- ◆ 系统时钟= IHRC/8 = 2MHz
- ◆ 看门狗计数器停用，ILRC 启用，PA5 引脚是输入模式

(2) .ADJUST_IC SYSClk=IHRC/16, IHRC=16MHz, V_{DD}=2.3V

开机后，CLKMD = 0x14:

- ◆ IHRC 频率在 V_{DD}=2.3V 时校准到 16MHz，并且 IHRC 模块是启用的
- ◆ 系统时钟= IHRC/16 = 1MHz
- ◆ 看门狗计数器停用，ILRC 启用，PA5 引脚是输入模式

(3) .ADJUST_IC SYSClk=IHRC/32, IHRC=16MHz, V_{DD}=5V

开机后，CLKMD = 0x74:

- ◆ IHRC 频率在 V_{DD}=5V 时校准到 16MHz，并且 IHRC 模块是启用的
- ◆ 系统时钟= IHRC/32 = 500kHz
- ◆ 看门狗计数器停用，ILRC 启用，PA5 引脚是输入模式

(4) .ADJUST_IC SYSClk=ILRC, IHRC=16MHz, V_{DD}=5V

开机后，CLKMD = 0XE4:

- ◆ IHRC 频率在 V_{DD}=5V 时校准到 16MHz，并且 IHRC 模块是停用的
- ◆ 系统时钟 = ILRC
- ◆ 看门狗计数器停用，ILRC 启用，PA5 引脚是输入模式

(5) .ADJUST_IC DISABLE

开机后，CLKMD 寄存器没有改变（没任何动作）:

- ◆ IHRC 没有校准并且 IHRC 模块是停用的（只有在快速开机时才启用 IHRC）。
- ◆ 系统频率= ILRC 或 IHRC/64
- ◆ 看门狗计数器启用，ILRC 启用，PA5 引脚是输入模式。

PMS151

1KW OTP IO型单片机

5.4.4. 系统时钟

系统时钟来自 IHRC 或者 ILRC，PMS151 的时钟系统的硬件框图，如图 2 所示：

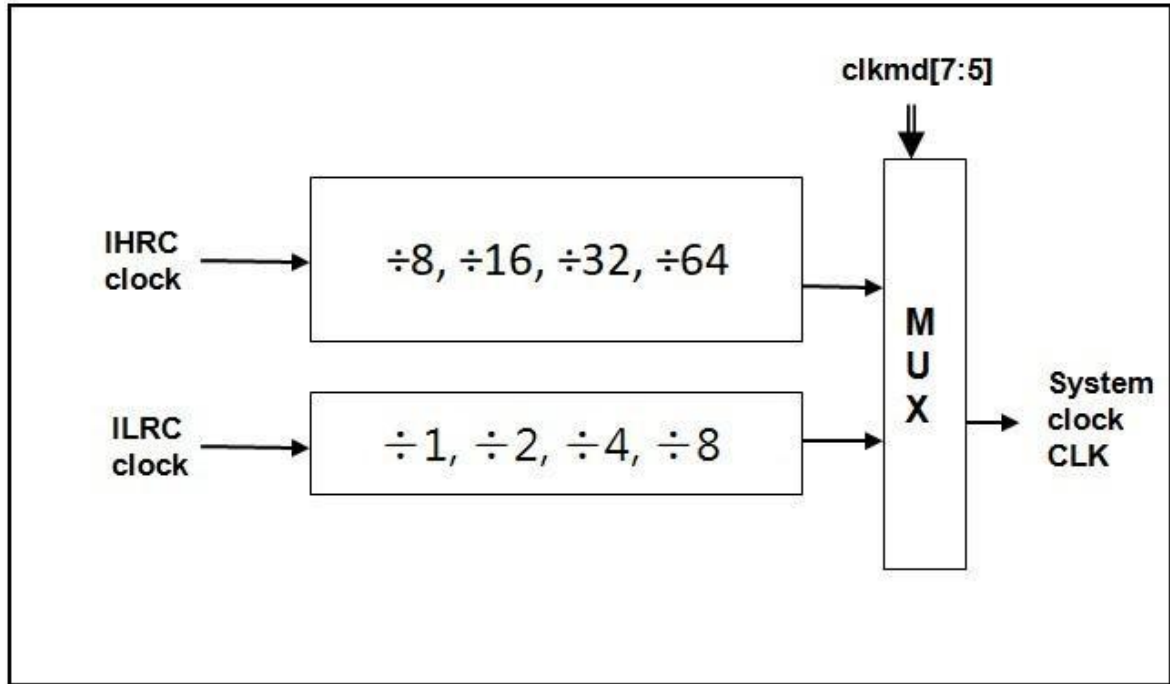


图 2：系统时钟选项

使用者可以在不同的需求下选择不同的系统时钟。

PMS151

1KW OTP IO型单片机

5.4.5. 系统时钟切换

IHRC 校准后，用户可能要求切换系统时钟到新的频率或者可能会随时切换系统时钟来优化系统性能及功耗。基本上，PMS151 的系统时钟能够随时通过设定寄存器 **clkmd** 在 IHRC 和 ILRC 之间切换。在设定寄存器 **clkmd** 之后，系统时钟立即转换成新的频率。**请注意，在下命令给 **clkmd** 寄存器时，不能同时关闭原来的时钟模块，下面这些例子显示更多时钟切换需知道的信息，请参阅 IDE 工具“求助”->“使用手册”->“IC 介绍”->“缓存器介绍”->CLKMD”。**

例 1: 系统时钟从 ILRC 切换到 IHRC/8

```
... // 系统时钟是LRC
CLKMD = 0x34; // 切换到IHRC/8, ILRC 不能在这里停用
CLKMD.2 = 0; // ILRC 可以在这里停用
...
```

例 2: 系统时钟从 IHRC/8 切换到 ILRC

```
... // 系统时钟是HRC/8
CLKMD = 0xF4; // 切换到ILRC, IHRC 不能在这里停用
CLKMD.4 = 0; // IHRC 可以在这里停用
...
```

例 3: 系统时钟从 IHRC/8 切换到IHRC/16

```
... // 系统时钟是HRC/8, ILRC 在这里是启用的
CLKMD = 0X14; // 切换到HRC/16
...
```

例 4: 如果同时切换系统时钟关闭原来的振荡器，系统会当机

```
... // 系统时钟是LRC
CLKMD = 0x30; // 不能从ILRC 切换到HRC/8 同时关闭LRC 振荡器
```

PMS151

1KW OTP IO型单片机

5.5. 16 位计数器 (Timer16)

PMS151 内置一个 16 位硬件计数器 (Timer16)，计数器时钟可来自于系统时钟 (CLK)，内部高频振荡时钟 (IHRC)，内部低频振荡时钟 (ILRC)，PA4 和 PA0。在送到 16 位计数器之前，1 个可软件编程的预分频器提供 $\div 1$ 、 $\div 4$ 、 $\div 16$ 、 $\div 64$ 选择，让计数范围更大。16 位计数器只能向上计数，计数器初始值可以使用 *stt16* 指令来设定，而计数器的数值也可以利用 *ldt16* 指令存储到 SRAM 数据存储器。

16 位计数器的中断请求可以通过 16 位计数器的位[15:8]来选择，中断类型可以上升沿触发或下降沿触发，定义在寄存器 *intgs.4*。Timer16 模块框图如图 3 所示。

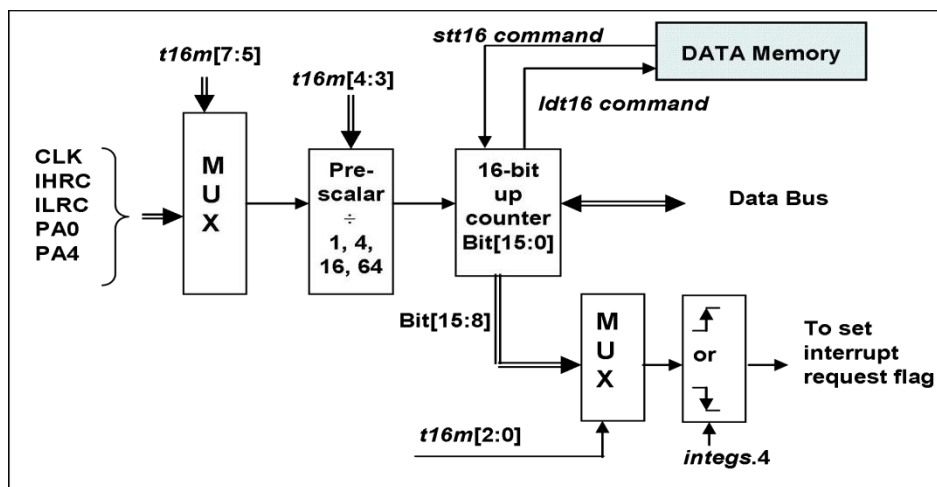


图 3: Timer16 模块框图

当使用 Timer16 时，Timer16 的语法定义在 .inc 文件中。有三个参数来定义 Timer16 的使用。第一个参数是用来定义 Timer16 的时钟源，第二个参数是用来定义预分频器，最后一个参数是定义中断源。详细如下：

```
T16M  IO_RW  0x06
$ 7~5:  STOP, SYSCLK, X, PA4_F, IHRC, X, ILRC, PA0_F           //第一个参数
$ 4~3:  /1, /4, /16, /64                                         //第二个参数
$ 2~0:  BIT8, BIT9, BIT10, BIT11, BIT12, BIT13, BIT14, BIT15   //第三个参数
```

使用者可以依照系统的要求来定义 T16M 参数，例子如下，更多例子请参考 IDE 软件“帮助→使用手册→IC 介绍→缓存器介绍→T16M”：

```
$ T16M SYSCLK, /64, BIT15;
// 选择(SYSCLK/64)当 Timer16 时钟源，每 2^16 个时钟周期产生一次 INTRQ.2=1
// 系统时钟 System Clock = IHRC / 8 = 2 MHz
// SYSCLK/64 = 2 MHz/64 = 31kHz(32us)，约每 2 S 产生一次 INTRQ.2=1

$ T16M PA0, /1, BIT8;
// 选择 PA0 当 Timer16 时钟源，每 2^9 个时钟周期产生一次 INTRQ.2=1
// 每接收 512 个 PA0 时钟周期产生一次 INTRQ.2=1

$ T16M STOP;
// 停止 Timer16 计数
```

PMS151

1KW OTP IO型单片机

5.6. 看门狗计数器

看门狗是一个计数器，其时钟源来自内部低频振荡器 (ILRC)。利用 *misc* 寄存器的选择，可以设定四种不同的看门狗超时时间，它是：

- ◆ 当 *misc*[1:0]=00（默认）时：4k ILRC 时钟周期
- ◆ 当 *misc*[1:0]=01 时：8k ILRC 时钟周期
- ◆ 当 *misc*[1:0]=10 时：32k ILRC 时钟周期
- ◆ 当 *misc*[1:0]=11 时：128k ILRC 时钟周期

ILRC 的频率有可能因为工厂制造的变化，电源电压和工作温度而漂移很多，使用者必须预留安全操作范围。由于在系统重启或者唤醒之后，看门狗计数周期会比预计要短，为防止看门狗计数溢出导致复位，建议在系统重启或唤醒之后使用立即 *wdreset* 指令清零看门狗计数。

当看门狗超时溢出时，PMS151 将复位并重新运行程序。看门狗时序图如图 4 所示。

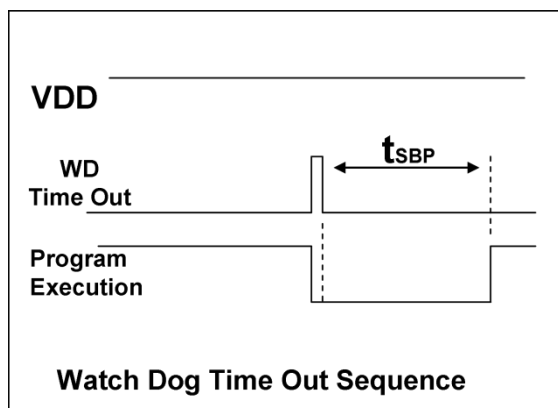


图 4：看门狗超时溢出时序图

PMS151

1KW OTP IO型单片机

5.7. 中断

PMS151 有 4 个中断源：

- ◆ 外部中断源 PA0
- ◆ 外部中断源 PB6（仿真器不支持）
- ◆ Timer16 中断
- ◆ ILRC/8

每个中断请求源都有自己的中断控制位来启用或停用。中断功能的硬件框图如图 5 所示。所有的中断请求标志位是由硬件置位并且并通过软件写寄存器 *intrq* 清零。中断请求标志设置点可以是上升沿或下降沿或两者兼而有之，这取决于对寄存器 *intregs* 的设置。所有的中断请求源最后都需由 *engint* 指令控制（启用全局中断）使中断运行，以及使用 *disgint* 指令（停用全局中断）停用它。

中断堆栈与数据存储器共享，其地址由堆栈寄存器 *sp* 指定。由于程序计数器是 16 位宽度，堆栈寄存器 *sp* 位 0 应保持 0。此外，用户可以使用 *pushaf* 指令存储 *ACC* 和标志寄存器的值到堆栈，以及使用 *popaf* 指令将值从堆栈恢复到 *ACC* 和标志寄存器中。由于堆栈与数据存储器共享，在 Mini-C 模式，堆栈位置与深度由编译程序安排。在汇编模式或自行定义堆栈深度时，用户应仔细安排位置，以防地址冲突。

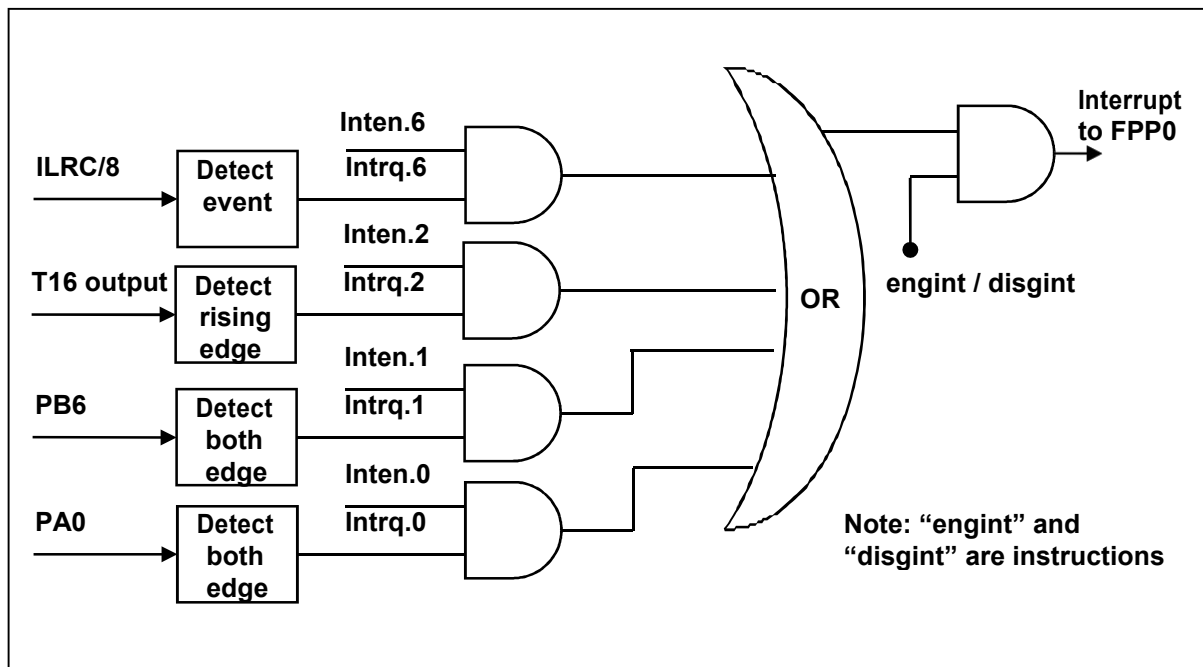


图 5：中断控制器硬件框图

一旦发生中断，其具体工作流程将是：

- ◆ 程序计数器将自动存储到 *sp* 寄存器指定的堆栈存储器。
- ◆ 新的 *sp* 将被更新为 *sp+2*。
- ◆ 全局中断将被自动停用。
- ◆ 将从地址 0x010 获取下一条指令。

在中断服务程序中，可以通过读寄存器 *intrq* 知道中断发生源。

PMS151

1KW OTP IO型单片机

注意：即使 INTEN 为 0，INTRQ 还是会被中断发生源触发。

中断服务程序完成后，发出 **reti** 指令返回既有的程序，其具体工作流程将是：

- ◆ 从 **sp** 寄存器指定的堆栈存储器自动恢复程序计数器。
- ◆ 新的 **sp** 将被更新为 **sp-2**。
- ◆ 全局中断将自动启用。
- ◆ 下一条指令将是中断前原来的指令。

使用者必须预留足够的堆栈存储器以存中断向量，一级中断需要两个字节，两级中断需要 4 个字节。下面的示例程序演示了如何处理中断，请注意，处理中断和 **pushaf** 是需要四个字节堆栈存储器。

```
void      FPPA0  (void)
{
    ...
    $ INTEN PA0;           // INTEN =1; 当PA0 准位改变 产生中断请求
    INTRQ = 0;             // 清除INTRQ
    ENGINT                 // 启用全局中断
    ...
    DISGINT                // 停用全局中断
    ...
}

void Interrupt (void)      // 中断程序
{
    PUSHAF                 // 存储ALU 和FLAG 寄存器

    // 如果INTEN.PA0 在程序会动态开和关 则表达式可以判断INTEN.PA0 是否为1。
    // 例如 If (INTEN.PA0 && INTRQ.PA0)  {...}

    // 如果INTEN.PA0 一直在使能状态，就可以省略判断INTEN.PA0，以加速中断执行。

    If (INTRQ.PA0)
    {
        // PA0 的中断程序
        INTRQ.PA0 = 0; // 只须清除对应的位(PA0)
        ...
    }
    ...
    // X: INTRQ = 0;      // 不建议在中断程序最后，才使用INTRQ = 0 一次全部清除
                          // 因为它可能会把刚发生而尚未处理的中断，意外清除掉
    POPAF                 // 恢复ALU 和FLAG 寄存器
}
```

PMS151

1KW OTP IO型单片机

5.8. 省电和掉电

PMS151 有三个由硬件定义的操作模式，分别为：正常工作模式，电源省电模式和掉电模式。正常工作模式是所有功能都正常运行的状态，省电模式(**stopexe**)是在降低工作电流而且 CPU 保持在随时可以继续工作的状态，掉电模式(**stopsys**)是用来深度的节省电力。因此，省电模式适合在偶尔需要唤醒的系统工作，掉电模式是在非常低消耗功率且很少需要唤醒的系统中使用。表 4 显示省电模式(**stopexe**)和掉电模式(**stopsys**)之间在振荡器模块的差异（没改变就是维持原状态）。

STOPSYS 和 STOPEXE 模式下在振荡器的差异		
	IHRC	ILRC
STOPSYS	停止	停止
STOPEXE	没改变	没改变

表 4：省电模式和掉电模式在振荡器模块的差异

5.8.1. 省电模式 (“**stopexe**”)

用 **stopexe** 指令进入省电模式，只有系统时钟被停用，其余所有的振荡器模块都仍继续工作。所以只有 CPU 是停止执行指令，然而，对 Timer16 计数器而言，如果它的时钟源不是系统时钟，那 Timer16 仍然会保持计数。**stopexe** 的省电模式下，唤醒源可以是 IO 的切换，或者 Timer16 计数到设定值时（假如 Timer16 的时钟源是 IHRC 或者 ILRC）。假如系统唤醒是因输入引脚切换，那可以视为系统继续正常运行。省电模式的详细信息如下所示：

- IHRC 振荡器模块：没改变，如果被启用，则仍然保持运行状态。
- ILRC 振荡器模块：必须保持启用，唤醒时需要靠 ILRC 启动。
- 系统时钟：停用，因此 CPU 停止运行。
- OTP 存储器关闭。
- Timer16：停止计数，如果选择系统时钟或相应的振荡器模块被停止，否则，仍然保持计数。
- 唤醒源：IO 唤醒（PxDIER 位是 1）或者 Timer16。

再举例使用 Timer16 唤醒省电模式 “**stopexe**”：

```
$ T16M  ILRC, /1, BIT8           // Timer16 setting
...
WORD    count    =    0;
STT16   count;
stopexe;
nop;
...
```

Timer16 的初始值为 0，在 Timer16 计数了 256 个 ILRC 时钟后，系统将被唤醒。

PMS151

1KW OTP IO型单片机

5.8.2. 掉电模式(“stopsys”)

掉电模式是深度省电的状态，所有的振荡器模块都会被关闭。通过使用“**stopsys**”指令，芯片会直接进入掉电模式。下面显示发出 **stopsys** 命令后，PMS151 内部详细的状态：

- 所有的振荡器模块被关闭
- OTP 存储器被关闭
- SRAM 和寄存器内容保持不变
- 唤醒源：数字输入使能（PxDIER 位是 1）的 IO 口发生切换

输入引脚的唤醒可以被视为正常运行的延续，为了降低功耗，进入掉电模式之前，所有的 I/O 引脚应仔细检查，避免悬空而漏电。断电参考示例程序如下所示：

```
CLKMD = 0xF4;      // 系统时钟从IHRC 变为ILRC, 关闭看门狗时钟
CLKMD.4 = 0;       // IHRC 停用
...
while (1)
{
    STOPSYS;        // 进入断电模式
    if (...) break; // 假如发生唤醒且检查OK, 就返回正常工作
                   // 否则, 停留在断电模式
}
CLKMD = 0x34;      // 系统时钟从ILRC 变为IHRC/8
```

5.8.3. 唤醒

进入掉电或省电模式后，PMS151 可以通过切换 IO 引脚恢复正常工作；而 Timer16 的唤醒只适用于省电模式。表 5 显示 **stopsys** 掉电模式和 **stopexe** 省电模式在唤醒源的差异。

掉电模式(stopsys)和省电模式(stopexe)在唤醒源的差异		
	IO 引脚切换	T16 唤醒
STOPSYS	是	否
STOPEXE	是	是

表 5：掉电模式和省电模式在唤醒源的差异

当使用 IO 引脚来唤醒 PMS151, **padier** 寄存器应对每一个相应的引脚正确设置“使能唤醒功能”。从唤醒事件发生后开始计数，正常的唤醒时间大约是 1K ILRC 时钟周期。可通过 **misc** 寄存器选择快速唤醒以减少唤醒时间，快速唤醒的时间约为 IO 引脚切换的 16 ILRC 时钟。

PMS151

1KW OTP IO型单片机

休眠模式	唤醒模式	切换IO 引脚的唤醒时间(twup)
STOPEXE suspend or STOPSYS suspend	快速唤醒	$16 * T_{ILRC}$, 这里 T_{ILRC} 是指 ILRC 时钟周期
STOPEXE suspend or STOPSYS suspend	正常唤醒	$1K * T_{ILRC}$, 这里 T_{ILRC} 是指 ILRC 时钟周期

表 6: 休眠模式/唤醒模式/IO 唤醒时间

5.9. T-type Key Scan 唤醒器

PMS151 支持低功耗硬件 T-type 扫描从 STOPEXE 唤醒 T-type 矩阵按键配置。通过设置 PATS 和 PBTS 寄存器，可以将每个 IO 选择为 T-scan 端口。但是，T-scan 端口必须是输入设置和上拉启用端口。一旦将一个引脚作为 T-scan 端口启用后，只接受一个低电平唤醒系统。

扫描速率由 tscnc [1:0] 寄存器选择。扫描速率越低，功耗越低，但对用户输入的敏感度越低。在进入 STOPEXE 之前，通过 tscnc.7 寄存器打开硬件 T-scan，并在 STOPEXE 唤醒后关闭它。

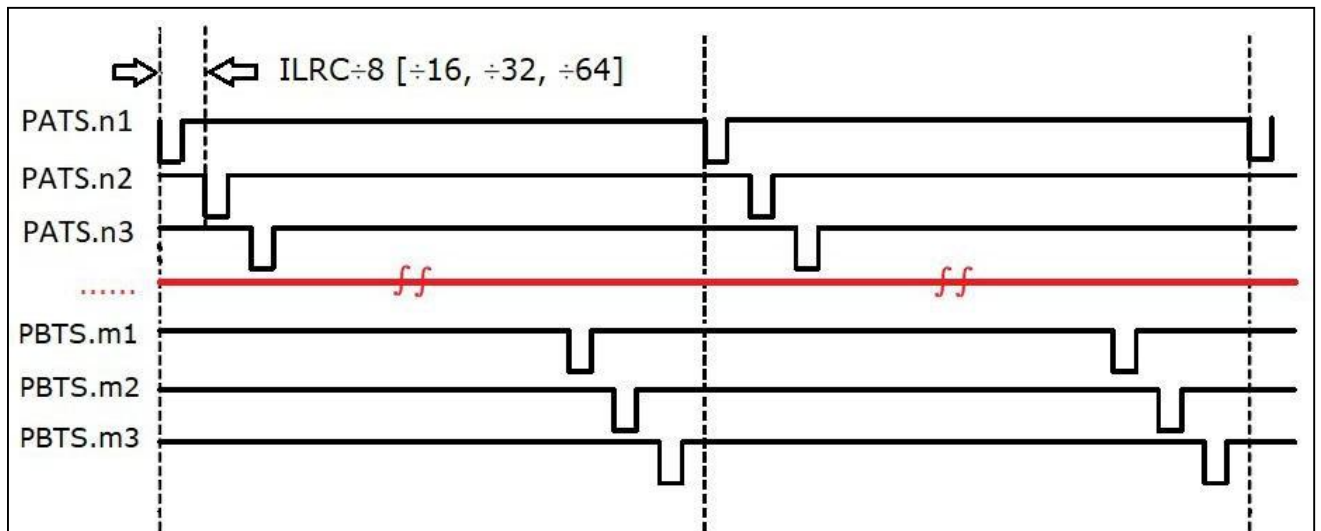


图 6: T-type Key Scan 波形

程序区块可以分成：初始阶段，软件判断 Key Scan，进入 STOPEXE 做硬件 Key Scan。

初始阶段：

```

PATS = 0x11;           // PA0, PA4 当作 Key Scan 的 IO 引脚
PBTS = 0x42;           // PB1, PB5 当作 Key Scan 的 IO 引脚
PAPH = xxx | 0x11;     // 设定对应的 Pull High
PBPH = xxx | 0x42;
PAC =  xxx & ~0x11;    // 设定对应的输入模式
PBC =  xxx & ~0x42;
$ TSCNC ILRC/8;        // 设定 Scan 频率为 ILRC/8
  
```

PMS151

1KW OTP IO型单片机

软件判断 Key Scan:

// PA0 / PA4 / PB1 / PB5 在程序使用过程中，允许 In/Out 切换做Key Scan
// 但在进入 STOPEXE 前，需要设为 Input 状态

进入 STOPEXE 做硬件Key Scan

```
$ CLKMD      En_ILRC, En_IHRC, ILRC/1;    // 进入 ILRC
CLKMD.En_IHRC = 0;                        // 关闭 IHRC
TSCNC.Enable = 1;                          // 开始 Key Scan
stopexe;                                    // Sleep
TSCNC.Enable = 0;                          // IC 唤醒，不须再做硬件Key Scan 了
$ CLKMD      En_ILRC, En_IHRC, IHRC/8;    // 进入高速的 IHRC/8
```

5.10. IO 引脚

除了 PA5，PMS151 所有 IO 引脚都可以设定成输入或输出，透过数据寄存器(*pa, pb*)，控制寄存器(*pac, pbc*)和弱上拉电阻(*paph, pbph*)设定，每一 IO 引脚都可以独立配置成不同的功能；所有这些引脚设置有施密特触发输入缓冲器和 CMOS 输出驱动电位水平。当这些引脚为输出低电位时，弱上拉电阻会自动关闭。如果要读取端口上的电位状态，一定要先设置成输入模式；在输出模式下，读取到的数据是数据寄存器的值。表 7 为端口 PA0 位的设定配置表。图 7 显示了 IO 缓冲区硬件图。

<i>pa.0</i>	<i>pac.0</i>	<i>paph.0</i>	描述
X	0	0	输入，没有弱上拉电阻
X	0	1	输入，有弱上拉电阻
0	1	X	输出低电位，没有弱上拉电阻
1	1	X	输出高电位，没有弱上拉电阻

表 7：PA0 设定配置表

PMS151

1KW OTP IO型单片机

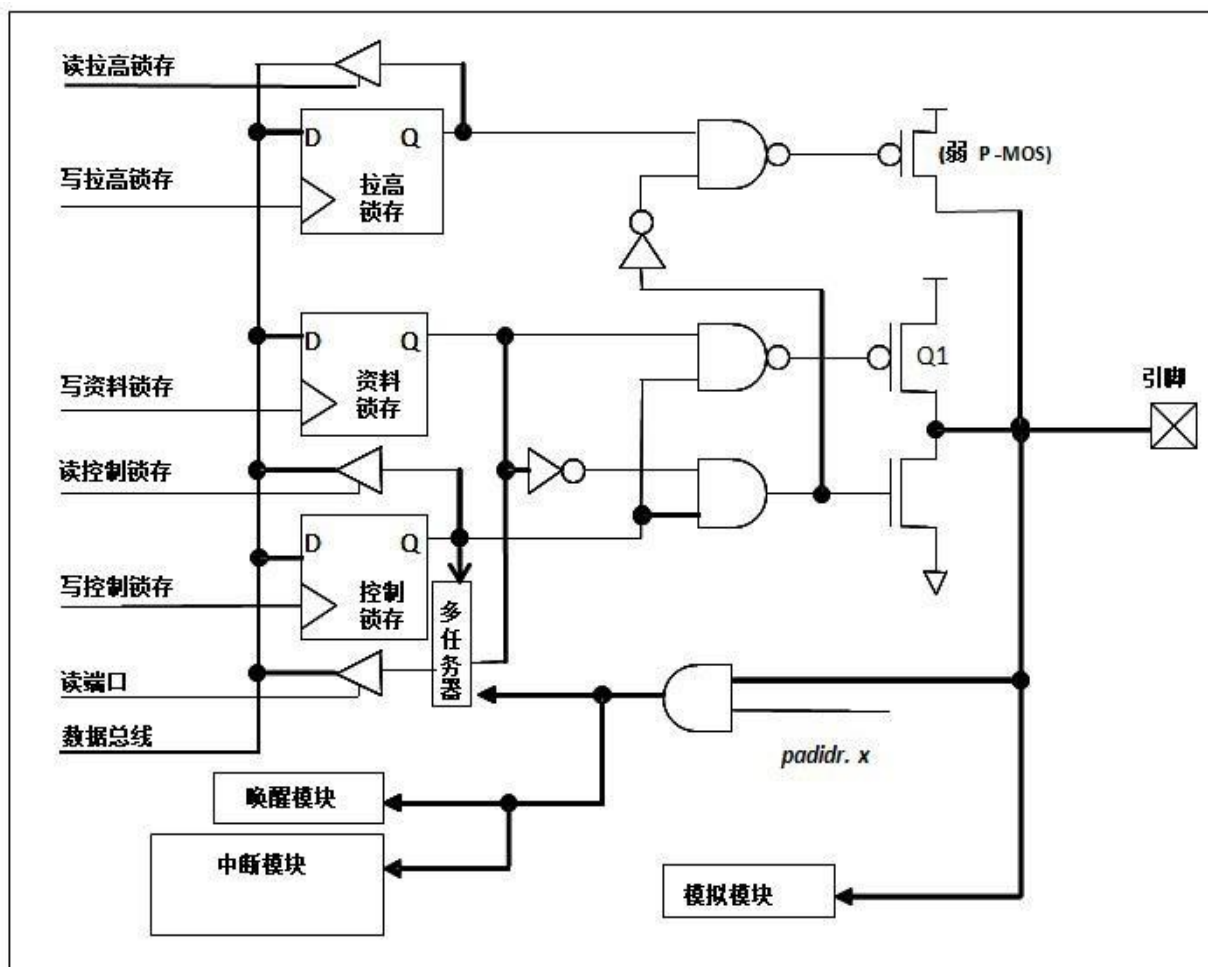


图 7: IO 引脚缓冲区硬件图

除了 PA5 外，所有的 IO 引脚具有相同的结构：PA5 的输出只能是漏极开路模式（没有 Q1）。对于被选择为模拟功能的引脚，必须在寄存器 **padier & pbdier** 相应位设置为低，以防止漏电流。当 PMS151 在掉电或省电模式，每一个引脚都可以切换其状态来唤醒系统。对于需用来唤醒系统的引脚，必须设置为输入模式以及寄存器 **padier & pbdier** 相应位为高。同样的原因，当 PA0 用作外部中断引脚时，**padier.0** 应设置为高。

5.11. 复位

引起 PMS151 复位的原因很多，一旦复位发生，PMS151 的所有寄存器将被设置为默认值，系统会重新启动，程序计数器会跳跃地址 0x0。当发生上电复位或 LVR 复位，数据存储器的值是在不确定的状态，然而，若是复位是因为 PRSTB 引脚或 WDT 超时溢位，数据存储器的值将被保留。

PMS151

1KW OTP IO型单片机

5.12. 8 位 IR Carrier 频率生成器 (IRTMC / IRTMB)

PMS151 内置 1 个 8 位 IR carrier 频率生成器 (IRTMB)。图 8 为 IRTMC 硬件框图，计数器的时钟源来自内部高频 RC 振荡器时钟 (IHRC)，有 $\div 1$ ， $\div 2$ ， $\div 16$ 和 $\div 64$ 的四种选择，由寄存器 IRTMC 的位[5:4]来设定。

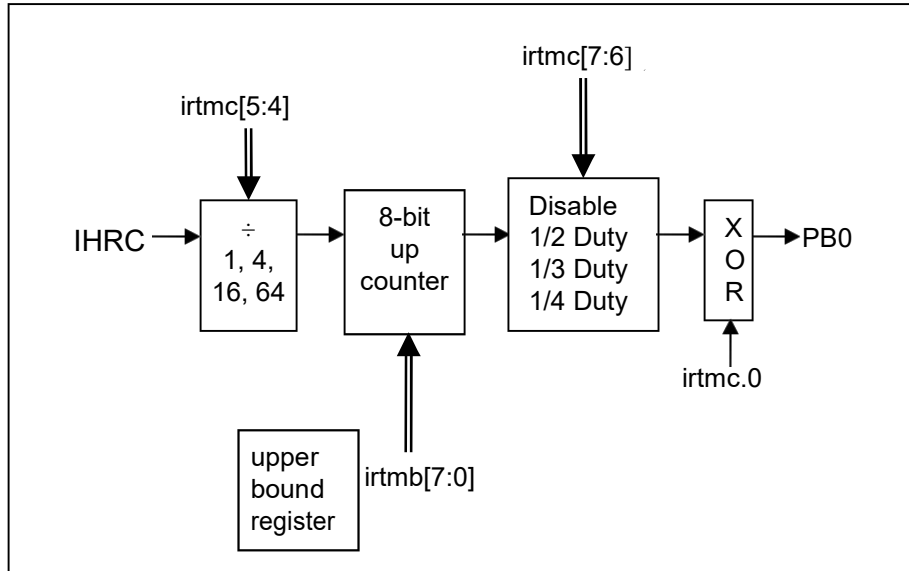


图 8: IRTMC 硬件框图

使用者可透过 IRTMC[7:6] 为 IR carrier 选择三种占空比，共有 1/2 duty，1/3 duty，1/4 占空比可以选择。此外，如果与 IRTMC.0 一起，则还有 2/3，3/4 这两个占空比可透过反转 IR carrier 获得。IR carrier 频率由 IRTM 生成，1/2，1/3，1/4 占空比周期对应于 $IRTMB \times 2$ ， $IRTMB \times 3$ ， $IRTMB \times 4$ 。

IRTMB 为 8 位计数器，如要输出 IR = 40KHz，需 16MHz 除以 400 才能达到，所以计数 200 次的时间，等于 IR 的半周期。

```
IRTMB    =    200 -1;           //    设定 199 等于计数 200 次
$ IRTMC   1/2, IHRC/1;          //    1/2 duty, 16MHz
                                     //    IR carrier = 40KHz
```

如要输出 IR = 38KHz，需 16MHz 除以 421 才能达到，所以计数 210 次的时间，等于 IR 的半周期。

```
IRTMB    =    210 -1;           //    设定 209 等于计数 210 次
$ IRTMC   1/2, IHRC/1;          //    1/2 duty, 16MHz
                                     //    IR carrier = 38.095KHz
```

PMS151

1KW OTP IO型单片机

要将 IR carrier 调整到 REM 输出，首先 pbc.0 设置的选择为 REM 脚位的输出模式。请参考下表，在选择 IR carrier 频率后，其余用户要做的是控制 pb.0。

<i>pbc.0</i>	<i>pb.0</i>	IRTMC[7:6]	描述
0	X	X	关闭 REM 电流输出
1	1	X	关闭 REM 电流输出
1	0	00	REM 输出低电平
1	0	others	REM 输出 IR carrier

表 8: REM 设定配置表

仿真器模拟时，对 IRTMC / IRTMB 的所有读写都会延迟，无法及时仿真，请以 IC 来做实际验证。

IRTMC	IO_RW	0x1C (0x3F)	
\$ 7 ~ 6 :	Disable, 1/2, 1/3, 1/4		// duty 选择
\$ 5 ~ 4 :	IHRC/1, IHRC/2, IHRC/16, IHRC/64		// 时钟源
\$ 0 :	X, Inverse		// 输出反向
IRTMB	IO_WO	0x1D (0x3F)	

PMS151

1KW OTP IO型单片机

6. IO 寄存器

6.1. ACC 状态标志寄存器 (*flag*), IO 地址 =0'h00

位	初始值	读/写	描述
7 - 4	-	-	保留。这 4 个位读是“1”。
3	-	读/写	OV（溢出标志）。溢出时置 1。
2	-	读/写	AC（辅助进位标志）。两个条件下，此位设置为 1：(1)是进行低半字节加法运算产生进位，(2)减法运算时，低半字节向高半字节借位。
1	-	读/写	C（进位标志）。有两个条件下，此位设置为 1：(1)加法运算产生进位，(2)减法运算有借位。进位标志还受带进位标志的 <i>shift</i> 指令影响。
0	-	读/写	Z（零）。此位将被设置为 1，当算术或逻辑运算的结果是 0；否则将被清零。

6.2. 堆栈指针寄存器 (*sp*), IO 地址 =0x02

位	初始值	读/写	描述
7 - 0	-	读/写	堆栈指针寄存器。读出当前堆栈指针，或写入以改变堆栈指针。请注意 0 位必须维持为 0 因程序计数器是 16 位。

6.3. 时钟模式寄存器 (*clkmd*), IO 地址 =0x03

位	初始值	读/写	描述
7 - 5	111	读/写	系统时钟 (CLK)选择： 000: IHRC/16 001: IHRC/8 010: ILRC/8（仿真器不支持） 011: IHRC/32 100: IHRC/64 101: ILRC/4 110: ILRC/2（仿真器不支持） 111: ILRC/1
4	0	读/写	内部高频 RC 振荡器功能。0/1：停用/启用
3	0	读/写	保留
2	1	读/写	内部低频 RC 振荡器功能。0/1：停用/启用 当内部低频 RC 振荡器功能停用时，看门狗功能同时被关闭。
1	1	读/写	看门狗功能。0/1：停用/启用
0	0	读/写	引脚 PA5/PRSTB 功能。0/1：PA5 / PRSTB

6.4. 中断允许寄存器 (*inten*), IO 地址 =0x04

位	初始值	读/写	描述
7	0	-	保留
6	0	读/写	使能 ILRC/8 中断。0/1：停用/启用
5	0	-	保留
4	0	-	保留
3	0	-	保留
2	0	读/写	使能 Timer16 溢出中断。0/1：停用/启用
1	0	读/写	使能 PB6 中断。0/1：停用/启用
0	0	读/写	使能 PA0 中断。0/1：停用/启用

PMS151

1KW OTP IO型单片机

6.5. 中断请求寄存器 (*intrq*), IO 地址 =0x05

位	初始值	读/写	描述
7	-	-	保留
6	-	读/写	ILRC/8 的中断请求, 此位是由硬件置位并由软件清零。0/1: 不要求/请求
5	-	-	保留
4	-	-	保留
3	-	-	保留
2	-	读/写	Timer16 的中断请求, 此位是由硬件置位并由软件清零。0/1: 不要求/请求
1	-	读/写	引脚 PB6 的中断请求, 此位是由硬件置位并由软件清零。0/1: 不要求/请求
0	-	读/写	引脚 PA0 的中断请求, 此位是由硬件置位并由软件清零。0/1: 不要求/请求

6.6. Timer16 控制寄存器 (*t16m*), IO 地址 =0x06

位	初始值	读/写	描述
7 - 5	000	读/写	Timer16 时钟选择: 000: Timer16 停用 001: CLK (系统时钟) 010: 保留 011: PA4 下降沿 (从外部引脚) 100: IHRC 101: 保留 110: ILRC 111: PA0 下降沿 (从外部引脚)
4 - 3	00	读/写	Timer16 时钟分频: 00: ÷1 01: ÷4 10: ÷16 11: ÷64
2 - 0	000	读/写	中断源选择。当所选择的状态位变化时, 中断事件发生。 0: bit 8 of Timer16 1: bit 9 of Timer16 2: bit 10 of Timer16 3: bit 11 of Timer16 4: bit 12 of Timer16 5: bit 13 of Timer16 6: bit 14 of Timer16 7: bit 15 of Timer16

PMS151

1KW OTP IO型单片机

6.7. 杂项寄存器 (*misc*), IO 地址 = 0x1b

位	初始值	读/写	描述
7 - 6	-	-	保留（写 0）。
5	0	只写	快唤醒功能。 0: 正常唤醒 唤醒时间大约 1KILRC 时钟（不适用快速开机） 1: 快速唤醒 唤醒时间大约 16 个 ILRC 时钟+ 振荡稳定时间 如果从 STOPEXE 唤醒，振荡稳定时间为 0 如果从 STOPSYS 唤醒，振荡稳定时间为 IHRC 或者 ILRC 的上电振荡稳定时间
4 - 2	-	-	保留（写 0）。
1 - 0	00	只写	看门狗时钟超时时间设定： 00: 4k ILRC 时钟周期 01: 8k ILRC 时钟周期 10: 32k ILRC 时钟周期 11: 128k ILRC 时钟周期

6.8. 中断边缘选择寄存器 (*intregs*), IO 地址 =0x0c

位	初始值	读/写	描述
7 - 5	-	-	保留。写 0。
4	0	只写	Timer16 中断边缘选择： 0: 上升缘请求中断。 1: 下降缘请求中断。
3 - 2	00	只写	PB6 中断边缘选择。 00: 上升缘和下降缘都请求中断 01: 上升缘请求中断 10: 下降缘请求中断 11: 保留
1 - 0	00	只写	PA0 中断边缘选择。 00: 上升缘和下降缘都请求中断 01: 上升缘请求中断 10: 下降缘请求中断 11: 保留

6.9. 端口 A 数字输入使能寄存器 (*padier*), IO 地址 =0x0d

位	初始值	读/写	描述
7 - 3	11111	只写	使能 PA7~PA3 数字输入和唤醒事件。1/0: 启用/ 停用 如果 PA7~PA3 位设 0 可停用唤醒。
2 - 1	-	-	保留。
0	1	只写	使能 PA0 数字输入和唤醒事件以及中断请求。1/0: 启用/ 停用 如果这个位设为 0, PA0 则不能用来唤醒系统, 并且停用中断请求。

PMS151

1KW OTP IO型单片机

6.10. 端口 B 数字输入使能寄存器 (*pbdier*), IO 地址 =0x0e

位	初始值	读/写	描述
7	1	只写	使能 PB7 数字输入和唤醒事件。1/0: 启用/ 停用 如果这个位设为 0, PB7 则不能用来唤醒系统。
6	1	只写	使能 PB6 数字输入和唤醒事件以及中断请求。1/0: 启用/ 停用 如果这个位设为 0, PB6 则不能用来唤醒系统, 并且停用中断请求。
5	1	只写	使能 PB5 数字输入和唤醒事件。1/0: 启用/ 停用 如果这个位设为 0, PB5 则不能用来唤醒系统。
4	1	只写	使能 PB4 数字输入和唤醒事件。1/0: 启用/ 停用 如果这个位设为 0, PB4 则不能用来唤醒系统。
3	1	只写	使能 PB3 数字输入和唤醒事件。1/0: 启用/ 停用 如果这个位设为 0, PB3 则不能用来唤醒系统。
2	1	只写	使能 PB2 数字输入和唤醒事件。1/0: 启用/ 停用 如果这个位设为 0, PB2 则不能用来唤醒系统。
1	1	只写	使能 PB1 数字输入和唤醒事件。1/0: 启用/ 停用 如果这个位设为 0, PB1 则不能用来唤醒系统。
0	-	只写	保留

6.11. 端口 A 数据寄存器 (*pa*), IO 地址 =0x10

位	初始值	读/写	描述
7 - 3	5'h00	读/写	数据寄存器的端口 A。
2 - 1	-	-	保留
0	0	读/写	数据寄存器的端口 A。

6.12. 端口 A 控制寄存器 (*pac*), IO 地址 =0x11

位	初始值	读/写	描述
7 - 3	5'h00	读/写	端口 A 控制寄存器。这些寄存器是用来定义端口 A 每个相应的引脚的输入模式或输出模式。 0/1: 输入/输出
2 - 1	-	-	保留
0	0	读/写	端口 A 控制寄存器。这些寄存器是用来定义端口 A 每个相应的引脚的输入模式或输出模式。 0/1: 输入/输出

6.13. 端口 A 上拉控制寄存器 (*paph*), IO 地址 =0x12

位	初始值	读/写	描述
7 - 0	5'h00	写	端口 A 上拉控制寄存器。这些寄存器是用来控制上拉高端口 A 每个相应的引脚。 0/1: 停用/启用
2 - 1	-	-	保留
0	0	写	口 A 上拉控制寄存器。这些寄存器是用来控制上拉高端口 A 每个相应的引脚。 0/1: 停用/启用

6.14. 端口 B 数据寄存器 (*pb*), IO 地址 =0x14

位	初始值	读/写	描述
7 - 1	7'h00	读/写	数据寄存器的端口 B。
0	-	-	保留

PMS151

1KW OTP IO型单片机

6.15. 端口 B 控制寄存器 (*pbc*), IO 地址 =0x15

位	初始值	读/写	描述
7 - 1	7'h00	读/写	端口 B 控制寄存器。这些寄存器是用来定义端口 B 每个相应的引脚的输入模式或输出模式。 0/1: 输入/输出
0	-	-	保留

6.16. 端口 B 上拉控制寄存器 (*pbph*), IO 地址 =0x16

位	初始值	读/写	描述
7 - 1	7'h00	读/写	端口 B 上拉控制寄存器。这些寄存器是用来控制上拉高端口 B 每个相应的引脚。 0/1: 停用/启用
0	-	-	保留

6.17. 端口 A T-scan 选择寄存器 (*pats*), IO 地址 =0x13

位	初始值	读/写	描述
7 - 3	5'h00	读/写	在 STOPEXE 中选择PA7~PA3 作为 T-scan 或者正常 IO 唤醒端口。 0/1: 正常/T-scan
2 - 1	-	-	保留
0	0	读/写	在 STOPEXE 中选择PA0 作为 T-scan 或者正常 IO 唤醒端口。 0/1: 正常/T-scan

6.18. 端口 B T-scan 选择寄存器 (*pbts*), IO 地址 =0x17

位	初始值	读/写	描述
7 - 1	7'h00	读/写	在 STOPEXE 中选择PB7~PB1 作为 T-scan 或者正常 IO 唤醒端口。 0/1: 正常/T-scan
0	0	读/写	保留

6.19. T-scan 控制寄存器 (*tscnc*), IO 地址 =0x1E

位	初始值	读/写	描述
7	0	读/写	在 STOPEXE 中启用硬件 T-scan。0/1: 停用/启用
6 - 2	-	-	保留
1 - 0	2'h0	读/写	选择 T-scan 频率。 00: ILRC/8 01: ILRC/16 10: ILRC/32 11: ILRC/64

PMS151

1KW OTP IO型单片机

6.20. IR Timer 控制寄存器 (*irtmc*), IO 地址 =0x1C

位	初始值	读/写	描述
7 - 6	2'h0	读/写	IRTX Carrier 占空比周期选择。 00: IRTX Carrier 总是 0 01: IRTX Carrier 为 1/2 占空比, IRTX Carrier 周期 = irtmb * 2 10: IRTX Carrier 为 1/3 占空比, IRTX Carrier 周期 = irtmb * 3 11: IRTX Carrier 为 1/4 占空比, IRTX Carrier 周期 = irtmb * 4
5 - 4	2'h0	读/写	IR timer 频率选择。 00: IHRC/1 01: IHRC/2 10: IHRC/16 11: IHRC/64
3 - 1	-	-	保留。
0	0	读/写	启用以反转 IRTX 载波输出的极性。 0/1: 停用/启用

6.21. IRTimer 上限寄存器 (*irtmb*), IO 地址 =0x1D

位	初始值	读/写	描述
7 - 0	-	只写	IRTX Carrier 计数上限寄存器。

PMS151

1KW OTP IO型单片机

7. 指令

符号	描述
ACC	累加器 (Accumulator 的缩写)
a	累加器 (Accumulator 在程序里的代表符号)
sp	堆栈指针
flag	ACC 标志寄存器
l	立即数据
&	逻辑与
	逻辑或
←	移动
^	异或
+	加
-	减
~	按位取反 (逻辑补数, 1 补数)
⌋	负数 (2 补码)
OV	溢出 (2 补数系统的运算结果超出范围)
Z	零 (如果零运算单元操作的结果是 0, 这位设置为 1)
C	进位 (Carry)
AC	辅助进位标志 (Auxiliary Carry)
word	只允许寻址在address 0~0x1F (0~31) 的位置
M.n	只允许寻址在address 0~0xF (0~15) 的位置

PMS151

1KW OTP IO型单片机

7.1. 数据传输类指令

<i>mov</i> a, l	移动即时数据到累加器 例如: <i>mov</i> a, 0x0f; 结果: $a \leftarrow 0fh$ 受影响标志位: Z: 『不变』, C: 『不变』, AC: 『不变』, OV: 『不变』
<i>mov</i> M, a	移动数据由累加器到存储器 例如: <i>mov</i> MEM, a; 结果: $MEM \leftarrow a$ 受影响标志位: Z: 『不变』, C: 『不变』, AC: 『不变』, OV: 『不变』
<i>mov</i> a, M	移动数据由存储器到累加器 例如: <i>mov</i> a, MEM; 结果: $a \leftarrow MEM$; 当 MEM 为零时, 标志位 Z 会被置位。 受影响标志位: Z: 『受影响』, C: 『不变』, AC: 『不变』, OV: 『不变』
<i>mov</i> a, IO	移动数据由 IO 到累加器 例如: <i>mov</i> a, pa; 结果: $a \leftarrow pa$; 当 pa 为零时, 标志位 Z 会被置位。 受影响标志位: Z: 『受影响』, C: 『不变』, AC: 『不变』, OV: 『不变』
<i>mov</i> IO, a	移动数据由累加器到IO 例如: <i>mov</i> pb, a; 结果: $pb \leftarrow a$ 受影响标志位: Z: 『不变』, C: 『不变』, AC: 『不变』, OV: 『不变』
<i>ldt16</i> word	将 Timer16 的 16 位计算值复制到 RAM 例如: <i>ldt16</i> word; 结果: $word \leftarrow 16\text{-bit timer}$ 受影响标志位: Z: 『不变』, C: 『不变』, AC: 『不变』, OV: 『不变』 应用范例: <pre>word T16val; // 定义一个 RAM word ... clear lb@T16val; // 清零 T16val (LSB) clear hb@T16val; // 清零 T16val (MSB) stt16 T16val; // 设定 Timer16 的起始值为 0 ... set1 t16m.5; // 启用 Timer16 ... set0 t16m.5; // 停用 Timer16 ldt16 T16val; // 将 Timer16 的 16 位计算值复制到 RAM T16val</pre>

PMS151

1KW OTP IO型单片机

<i>stt16</i> word	将放在word 的 16 位 RAM 复制到Timer16 例如: <i>stt16</i> word; 结果: 16-bit timer $\leftarrow$ word 受影响标志位: Z: 『不变』, C: 『不变』, AC: 『不变』, OV: 『不变』 应用范例: <pre> word T16val; // 定义一个 RAM word ... mov a, 0x34; mov lb@T16val, a; // 将 0x34 搬到 T16val (LSB) mov a, 0x12; mov hb@T16val, a; // 将 0x12 搬到 T16val (MSB) stt16 T16val; // Timer16 初始化 0x1234 ... </pre>
<i>idxm</i> a, index	使用索引作为RAM 的地址并将 RAM 的数据读取并载入到累加器。它需要 2T 时间执行这一指令 例如: <i>idxm</i> a, index; 结果: $a \leftarrow [\text{index}]$, index 是用 word 定义。 受影响的标志位: Z: 『不变』, C: 『不变』, AC: 『不变』, OV: 『不变』 应用范例: <pre> word RAMIndex; // 定义一个 RAM 指针 ... mov a, 0x5B; // 指定指针地址 (LSB) mov lb@RAMIndex, a; // 将指针存到 RAM (LSB) mov a, 0x00; // 指定指针地址为 0x00 (MSB), 在 PMS151 要为 0 mov hb@RAMIndex, a; // 将指针存到 RAM (MSB) ... idxm a, RAMIndex; // 将 RAM 地址为 0x5B 的数据读取并载入累加器 </pre>
<i>ldxm</i> index, a	使用索引作为RAM 的地址并将累加器的数据读取并载入到 RAM。它需要 2T 时间执行这一指令 例如: <i>ldxm</i> index, a; 结果: $[\text{index}] \leftarrow a$; index 是以 word 定义。 受影响的标志位: Z: 『不变』, C: 『不变』, AC: 『不变』, OV: 『不变』 应用范例: <pre> word RAMIndex; // 定义一个 RAM 指针 ... mov a, 0x5B; // 指定指针地址 (LSB) mov lb@RAMIndex, a; // 将指针存到 RAM (LSB) mov a, 0x00; // 指定指针地址为 0x00 (MSB), 在 PMS151 要为 0 mov hb@RAMIndex, a; // 将指针存到 RAM (MSB) ... mov a, 0xA5; ldxm RAMIndex, a; // 将累加器数据读取并载入地址为 0x5B 的 RAM </pre>

PMS151

1KW OTP IO型单片机

<i>xch</i> M	累加器与 RAM 之间交换数据 例如： <i>xch</i> MEM ; 结果： MEM ← a , a ← MEM 受影响的标志位： Z: 『不变』, C: 『不变』, AC: 『不变』, OV: 『不变』
<i>pushaf</i>	将累加器和算术逻辑状态寄存器的数据存到堆栈指针指定的堆栈存储器 例如： <i>pushaf</i>; 结果： [sp] ← {flag, ACC}; sp ← sp + 2 ; 受影响的标志位： Z: 『不变』, C: 『不变』, AC: 『不变』, OV: 『不变』 应用范例： ----- .romadr 0x10 ; // 中断服务程序入口地址 <i>pushaf</i> ; // 将累加器和算术逻辑状态寄存器的资料存到堆栈存储器 ... // 中断服务程序 ... // 中断服务程序 <i>popaf</i> ; // 将堆栈存储器的资料回存到累加器和算术逻辑状态寄存器 <i>reti</i> ; -----
<i>popaf</i>	将堆栈指针指定的堆栈存储器的数据回传到累加器和算术逻辑状态寄存器 例如： <i>popaf</i>; 结果： sp ← sp - 2 ; {Flag, ACC} ← [sp] ; 受影响的标志位： Z: 『受影响』, C: 『受影响』, AC: 『受影响』, OV: 『受影响』

PMS151

1KW OTP IO型单片机

7.2. 算术运算类指令

<i>add</i> a, I	将立即数据与累加器相加，然后把结果放入累加器 例如： <i>add</i> a, 0x0f; 结果： $a \leftarrow a + 0fh$ 受影响的标志位： Z:『受影响』, C:『受影响』, AC:『受影响』, OV:『受影响』
<i>add</i> a, M	将 RAM 与累加器相加，然后把结果放入累加器 例如： <i>add</i> a, MEM; 结果： $a \leftarrow a + MEM$ 受影响的标志位： Z:『受影响』, C:『受影响』, AC:『受影响』, OV:『受影响』
<i>add</i> M, a	将 RAM 与累加器相加，然后把结果放入 RAM 例如： <i>add</i> MEM, a; 结果： $MEM \leftarrow a + MEM$ 受影响的标志位： Z:『受影响』, C:『受影响』, AC:『受影响』, OV:『受影响』
<i>addc</i> a, M	将 RAM、累加器以及进位相加，然后把结果放入累加器 例如： <i>addc</i> a, MEM; 结果： $a \leftarrow a + MEM + C$ 受影响的标志位： Z:『受影响』, C:『受影响』, AC:『受影响』, OV:『受影响』
<i>addc</i> M, a	将 RAM、累加器以及进位相加，然后把结果放入 RAM 例如： <i>addc</i> MEM, a; 结果： $MEM \leftarrow a + MEM + C$ 受影响的标志位： Z:『受影响』, C:『受影响』, AC:『受影响』, OV:『受影响』
<i>addc</i> a	将累加器与进位相加，然后把结果放入累加器 例如： <i>addc</i> a; 结果： $a \leftarrow a + C$ 受影响的标志位： Z:『受影响』, C:『受影响』, AC:『受影响』, OV:『受影响』
<i>addc</i> M	将 RAM 与进位相加，然后把结果放入 RAM 例如： <i>addc</i> MEM; 结果： $MEM \leftarrow MEM + C$ 受影响的标志位： Z:『受影响』, C:『受影响』, AC:『受影响』, OV:『受影响』
<i>sub</i> a, I	累加器减立即数据，然后把结果放入累加器 例如： <i>sub</i> a, 0x0f; 结果： $a \leftarrow a - 0fh$ ($a + [2's \text{ complement of } 0fh]$) 受影响的标志位： Z:『受影响』, C:『受影响』, AC:『受影响』, OV:『受影响』
<i>sub</i> a, M	累加器减 RAM，然后把结果放入累加器 例如： <i>sub</i> a, MEM; 结果： $a \leftarrow a - MEM$ ($a + [2's \text{ complement of } M]$) 受影响的标志位： Z:『受影响』, C:『受影响』, AC:『受影响』, OV:『受影响』
<i>sub</i> M, a	RAM 减累加器，然后把结果放入 RAM 例如： <i>sub</i> MEM, a; 结果： $MEM \leftarrow MEM - a$ ($MEM + [2's \text{ complement of } a]$) 受影响的标志位： Z:『受影响』, C:『受影响』, AC:『受影响』, OV:『受影响』
<i>subc</i> a, M	累加器减 RAM，再减进位，然后把结果放入累加器 例如： <i>subc</i> a, MEM; 结果： $a \leftarrow a - MEM - C$ 受影响的标志位： Z:『受影响』, C:『受影响』, AC:『受影响』, OV:『受影响』
<i>subc</i> M, a	RAM 减累加器，再减进位，然后把结果放入 RAM 例如： <i>subc</i> MEM, a; 结果： $MEM \leftarrow MEM - a - C$ 受影响的标志位： Z:『受影响』, C:『受影响』, AC:『受影响』, OV:『受影响』

PMS151

1KW OTP IO型单片机

<i>subc</i> a	累加器减进位，然后把结果放入累加器 例如: <i>subc</i> a; 结果: $a \leftarrow a - C$ 受影响的标志位: Z:『受影响』, C:『受影响』, AC:『受影响』, OV:『受影响』
<i>subc</i> M	RAM 减进位，然后把结果放入 RAM 例如: <i>subc</i> MEM; 结果: $MEM \leftarrow MEM - C$ 受影响的标志位: Z:『受影响』, C:『受影响』, AC:『受影响』, OV:『受影响』
<i>inc</i> M	RAM 加 1 例如: <i>inc</i> MEM ; 结果: $MEM \leftarrow MEM + 1$ 受影响的标志位: Z:『受影响』, C:『受影响』, AC:『受影响』, OV:『受影响』
<i>dec</i> M	RAM 减 1 例如: <i>dec</i> MEM; 结果: $MEM \leftarrow MEM - 1$ 受影响的标志位: Z:『受影响』, C:『受影响』, AC:『受影响』, OV:『受影响』
<i>clear</i> M	清除 RAM 为 0 例如: <i>clear</i> MEM ; 结果: $MEM \leftarrow 0$ 受影响的标志位: Z:『不变』, C:『不变』, AC:『不变』, OV:『不变』

PMS151

1KW OTP IO型单片机

7.3. 移位运算类指令

<i>sr a</i>	累加器的位右移，位 7 移入值为 0 例如： <i>sr a</i> ; 结果： $a(0,b7,b6,b5,b4,b3,b2,b1) \leftarrow a(b7,b6,b5,b4,b3,b2,b1,b0)$, $C \leftarrow a(b0)$ 受影响的标志位： Z:『不变』, C:『受影响』, AC:『不变』, OV:『不变』
<i>src a</i>	累加器的位右移，位 7 移入进位标志位 例如： <i>src a</i> ; 结果： $a(c,b7,b6,b5,b4,b3,b2,b1) \leftarrow a(b7,b6,b5,b4,b3,b2,b1,b0)$, $C \leftarrow a(b0)$ 受影响的标志位： Z:『不变』, C:『受影响』, AC:『不变』, OV:『不变』
<i>sr M</i>	RAM 的位右移，位 7 移入值为 0 例如： <i>sr MEM</i> ; 结果： $MEM(0,b7,b6,b5,b4,b3,b2,b1) \leftarrow MEM(b7,b6,b5,b4,b3,b2,b1,b0)$, $C \leftarrow MEM(b0)$ 受影响的标志位： Z:『不变』, C:『受影响』, AC:『不变』, OV:『不变』
<i>src M</i>	RAM 的位右移，位 7 移入进位标志位 例如： <i>src MEM</i> ; 结果： $MEM(c,b7,b6,b5,b4,b3,b2,b1) \leftarrow MEM(b7,b6,b5,b4,b3,b2,b1,b0)$, $C \leftarrow MEM(b0)$ 受影响的标志位： Z:『不变』, C:『受影响』, AC:『不变』, OV:『不变』
<i>sl a</i>	累加器的位左移，位 0 移入值为 0 例如： <i>sl a</i> ; 结果： $a(b6,b5,b4,b3,b2,b1,b0,0) \leftarrow a(b7,b6,b5,b4,b3,b2,b1,b0)$, $C \leftarrow a(b7)$ 受影响的标志位： Z:『不变』, C:『受影响』, AC:『不变』, OV:『不变』
<i>slc a</i>	累加器的位左移，位 0 移入进位标志位 例如： <i>slc a</i> ; 结果： $a(b6,b5,b4,b3,b2,b1,b0,c) \leftarrow a(b7,b6,b5,b4,b3,b2,b1,b0)$, $C \leftarrow a(b7)$ 受影响的标志位： Z:『不变』, C:『受影响』, AC:『不变』, OV:『不变』
<i>sl M</i>	RAM 的位左移，位 0 移入值为 0 例如： <i>sl MEM</i> ; 结果： $MEM(b6,b5,b4,b3,b2,b1,b0,0) \leftarrow MEM(b7,b6,b5,b4,b3,b2,b1,b0)$, $C \leftarrow MEM(b7)$ 受影响的标志位： Z:『不变』, C:『受影响』, AC:『不变』, OV:『不变』
<i>slc M</i>	RAM 的位左移，位 0 移入进位标志位 例如： <i>slc MEM</i> ; 结果： $MEM(b6,b5,b4,b3,b2,b1,b0,C) \leftarrow MEM(b7,b6,b5,b4,b3,b2,b1,b0)$, $C \leftarrow MEM(b7)$ 受影响的标志位： Z:『不变』, C:『受影响』, AC:『不变』, OV:『不变』
<i>swap a</i>	累加器的高 4 位与低 4 位互换 例如： <i>swap a</i> ; 结果： $a(b3,b2,b1,b0,b7,b6,b5,b4) \leftarrow a(b7,b6,b5,b4,b3,b2,b1,b0)$ 受影响的标志位： Z:『不变』, C:『不变』, AC:『不变』, OV:『不变』

PMS151

1KW OTP IO型单片机

7.4. 逻辑运算类指令

<i>and</i> a, l	累加器和立即数据执行逻辑 AND，然后把结果保存到累加器 例如： <i>and</i> a, 0x0f ; 结果： $a \leftarrow a \& 0fh$ 受影响的标志位： Z: 『受影响』, C: 『不变』, AC: 『不变』, OV: 『不变』
<i>and</i> a, M	累加器和 RAM 执行逻辑 AND，然后把结果保存到累加器 例如： <i>and</i> a, RAM10 ; 结果： $a \leftarrow a \& RAM10$ 受影响的标志位： Z: 『受影响』, C: 『不变』, AC: 『不变』, OV: 『不变』
<i>and</i> M, a	累加器和 RAM 执行逻辑 AND，然后把结果保存到RAM 例如： <i>and</i> MEM, a ; 结果： $MEM \leftarrow a \& MEM$ 受影响的标志位： Z: 『受影响』, C: 『不变』, AC: 『不变』, OV: 『不变』
<i>or</i> a, l	累加器和立即数据执行逻辑 OR，然后把结果保存到累加器 例如： <i>or</i> a, 0x0f ; 结果： $a \leftarrow a 0fh$ 受影响的标志位： Z: 『受影响』, C: 『不变』, AC: 『不变』, OV: 『不变』
<i>or</i> a, M	累加器和 RAM 执行逻辑 OR，然后把结果保存到累加器 例如： <i>or</i> a, MEM ; 结果： $a \leftarrow a MEM$ 受影响的标志位： Z: 『受影响』, C: 『不变』, AC: 『不变』, OV: 『不变』
<i>or</i> M, a	累加器和 RAM 执行逻辑 OR，然后把结果保存到 RAM 例如： <i>or</i> MEM, a ; 结果： $MEM \leftarrow a MEM$ 受影响的标志位： Z: 『受影响』, C: 『不变』, AC: 『不变』, OV: 『不变』
<i>xor</i> a, l	累加器和立即数据执行逻辑 XOR，然后把结果保存到累加器 例如： <i>xor</i> a, 0x0f ; 结果： $a \leftarrow a \wedge 0fh$ 受影响的标志位： Z: 『受影响』, C: 『不变』, AC: 『不变』, OV: 『不变』
<i>xor</i> IO, a	累加器和 IO 寄存器执行逻辑 XOR，然把结果存到 IO 寄存器 例如： <i>xor</i> pa, a ; 结果： $pa \leftarrow a \wedge pa$; // pa 是 port A 资料寄存器 受影响的标志位： Z: 『不变』, C: 『不变』, AC: 『不变』, OV: 『不变』
<i>xor</i> a, M	累加器和 RAM 执行逻辑 XOR，然后把结果保存到累加器 例如： <i>xor</i> a, MEM ; 结果： $a \leftarrow a \wedge RAM10$ 受影响的标志位： Z: 『受影响』, C: 『不变』, AC: 『不变』, OV: 『不变』
<i>xor</i> M, a	累加器和 RAM 执行逻辑 XOR，然后把结果保存到 RAM 例如： <i>xor</i> MEM, a ; 结果： $MEM \leftarrow a \wedge MEM$ 受影响的标志位： Z: 『受影响』, C: 『不变』, AC: 『不变』, OV: 『不变』

PMS151

1KW OTP IO型单片机

<i>not</i> a	累加器执行 1 补码运算，结果放在累加器 例如： <i>not</i> a； 结果： $a \leftarrow \sim a$ 受影响的标志位： Z:『受影响』， C:『不变』， AC:『不变』， OV:『不变』 应用范例： <pre> mov a, 0x38 ; // ACC=0X38 not a ; // ACC=0XC7 </pre>
<i>not</i> M	RAM 执行 1 补码运算，结果放在 RAM 例如： <i>not</i> MEM； 结果： $MEM \leftarrow \sim MEM$ 受影响的标志位： Z:『受影响』， C:『不变』， AC:『不变』， OV:『不变』 应用范例： <pre> mov a, 0x38 ; mov mem, a ; // mem = 0x38 not mem ; // mem = 0xC7 </pre>
<i>neg</i> a	累加器执行 2 补码运算，结果放在累加器 例如： <i>neg</i> a； 结果： $a \leftarrow a$ 的 2 补码 受影响的标志位： Z:『受影响』， C:『不变』， AC:『不变』， OV:『不变』 应用范例： <pre> mov a, 0x38 ; // ACC=0X38 neg a ; // ACC=0XC8 </pre>
<i>neg</i> M	RAM 执行 2 补码运算，结果放在 RAM 例如： <i>neg</i> MEM； 结果： $MEM \leftarrow MEM$ 的 2 补码 受影响的标志位： Z:『受影响』， C:『不变』， AC:『不变』， OV:『不变』 应用范例： <pre> mov a, 0x38 ; mov mem, a ; // mem = 0x38 not mem ; // mem = 0xC8 </pre>

PMS151

1KW OTP IO型单片机

7.5. 位运算类指令

<i>set0</i> IO.n	IO 口的位 N 拉低电位 例 如： <i>set0</i> pa.5； 结果： PA5=0 受影响的标志位： Z: 『不变』， C: 『不变』， AC: 『不变』， OV: 『不变』
<i>set1</i> IO.n	IO 口的位 N 拉高电位 例 如： <i>set1</i> pa.5； 结果： PA5=1 受影响的标志位： Z: 『不变』， C: 『不变』， AC: 『不变』， OV: 『不变』
<i>set0</i> M.n	RAM 的位 N 设为 0 例如： <i>set0</i> MEM.5； 结果： MEM 位 5 为 0 受影响的标志位： Z: 『不变』， C: 『不变』， AC: 『不变』， OV: 『不变』
<i>set1</i> M.n	RAM 的位 N 设为 1 例如： <i>set1</i> MEM.5； 结果： MEM 位 5 为 1 受影响的标志位： Z: 『不变』， C: 『不变』， AC: 『不变』， OV: 『不变』

7.6. 条件运算类指令

<i>ceqsn</i> a, l	比较累加器与立即数据，如果是相同的，即跳过下一指令。标志位的改变与 $(a \leftarrow a - l)$ 相同 例如： <i>ceqsn</i> a, 0x55； <i>inc</i> MEM； <i>goto</i> error； 结果： 假如 $a=0x55$, then “goto error”；否则, “inc MEM” 受影响的标志位： Z: 『受影响』， C: 『受影响』， AC: 『受影响』， OV: 『受影响』
<i>ceqsn</i> a, M	比较累加器与 RAM，如果是相同的，即跳过下一指令。标志位改变与 $(a \leftarrow a - M)$ 相同 例如： <i>ceqsn</i> a, MEM； 结果： 假如 $a=MEM$ ，跳过下一个指令 受影响的标志位： Z: 『受影响』， C: 『受影响』， AC: 『受影响』， OV: 『受影响』
<i>t0sn</i> IO.n	如果 IO 的指定位是 0，跳过下一个指令 例如： <i>t0sn</i> pa.5； 结果： 如果 PA5 是 0，跳过下一个指令 受影响的标志位： Z: 『不变』， C: 『不变』， AC: 『不变』， OV: 『不变』
<i>t1sn</i> IO.n	如果 IO 的指定位是 1，跳过下一个指令 例如： <i>t1sn</i> pa.5； 结果： 如果 PA5 是 1，跳过下一个指令 受影响的标志位： Z: 『不变』， C: 『不变』， AC: 『不变』， OV: 『不变』
<i>t0sn</i> M.n	如果 RAM 的指定位是 0，跳过下一个指令 例如： <i>t0sn</i> MEM.5； 结果： 如果 MEM 的位 5 是 0，跳过下一个指令 受影响的标志位： Z: 『不变』， C: 『不变』， AC: 『不变』， OV: 『不变』
<i>t1sn</i> M.n	如果 RAM 的指定位是 1，跳过下一个指令

PMS151

1KW OTP IO型单片机

	例如: <code>t1sn MEM.5</code> ; 结果: 如果 MEM 的位 5 是 1, 跳过下一个指令 受影响的标志位: Z:『不变』, C:『不变』, AC:『不变』, OV:『不变』
<code>izsn a</code>	累加器加 1, 若累加器新值是 0, 跳过下一个指令 例如: <code>izsn a</code>; 结果: $a \leftarrow a + 1$, 若 $a=0$, 跳过下一个指令 受影响的标志位: Z:『受影响』, C:『受影响』, AC:『受影响』, OV:『受影响』
<code>dzsn a</code>	累加器减 1, 若累加器新值是 0, 跳过下一个指令 例如: <code>dzsn a</code>; 结果: $a \leftarrow a - 1$, 若 $a=0$, 跳过下一个指令 受影响的标志位: Z:『受影响』, C:『受影响』, AC:『受影响』, OV:『受影响』
<code>izsn M</code>	RAM 加 1, 若 RAM 新值是 0, 跳过下一个指令 例如: <code>izsn MEM</code>; 结果: $MEM \leftarrow MEM + 1$, 若 $MEM=0$, 跳过下一个指令 受影响的标志位: Z:『受影响』, C:『受影响』, AC:『受影响』, OV:『受影响』
<code>dzsn M</code>	RAM 减 1, 若 RAM 新值是 0, 跳过下一个指令 例如: <code>dzsn MEM</code>; 结果: $MEM \leftarrow MEM - 1$, 若 $MEM=0$, 跳过下一个指令 受影响的标志位: Z:『受影响』, C:『受影响』, AC:『受影响』, OV:『受影响』

7.7. 系统控制类指令

<code>call label</code>	函数调用, 地址可以是全部空间的任一地址 例如: <code>call function1</code>; 结果: $[sp] \leftarrow pc + 1$ $pc \leftarrow function1$ $sp \leftarrow sp + 2$ 受影响的标志位: Z:『不变』, C:『不变』, AC:『不变』, OV:『不变』
<code>goto label</code>	转到指定的地址, 地址可以是全部空间的任一地址 例如: <code>goto error</code>; 结果: 跳到 <code>error</code> 并继续执行程序 受影响的标志位: Z:『不变』, C:『不变』, AC:『不变』, OV:『不变』
<code>ret l</code>	将立即数据复制到累加器, 然后返回 例如: <code>ret 0x55</code>; 结果: $A \leftarrow 55h$ <code>ret</code> ; 受影响的标志位: Z:『不变』, C:『不变』, AC:『不变』, OV:『不变』
<code>ret</code>	从函数调用中返回原程序 例如: <code>ret</code>; 结果: $sp \leftarrow sp - 2$ $pc \leftarrow [sp]$ 受影响的标志位: Z:『不变』, C:『不变』, AC:『不变』, OV:『不变』
<code>reti</code>	从中断服务程序返回到原程序。在这指令执行之后, 全部中断将自动启用。 例如: <code>reti</code>; 受影响的标志位: Z:『不变』, C:『不变』, AC:『不变』, OV:『不变』
<code>nop</code>	没有任何动作 例如: <code>nop</code>;

PMS151

1KW OTP IO型单片机

	结果： 没任何改变 受影响的标志位： Z:『不变』, C:『不变』, AC:『不变』, OV:『不变』
<i>pcadd a</i>	目前的程序计数器加累加器是下一个程序计数器 例如： <i>pcadd a</i>; 结果： $pc \leftarrow pc + a$ 受影响的标志位： Z:『不变』, C:『不变』, AC:『不变』, OV:『不变』 应用范例： <pre> ... mov a, 0x02 ; pcadd a ; // PC <- PC+2 goto err1 ; goto correct ; // 跳到这里 goto err2 ; goto err3 ; ... correct: // 跳到这里 ... </pre>
<i>engint</i>	允许全部中断 例如： <i>engint</i>; 结果： 中断要求可送到 FPP0，以便进行中断服务 受影响的标志位： Z:『不变』, C:『不变』, AC:『不变』, OV:『不变』
<i>disgint</i>	停止全部中断 例如： <i>disgint</i>; 结果： 送到 FPP0 的中断要求全部被挡住，无法进行中断服务 受影响的标志位： Z:『不变』, C:『不变』, AC:『不变』, OV:『不变』
<i>stopsys</i>	系统停止 例如： <i>stopsys</i>; 结果： 停止系统时钟和关闭系统 受影响的标志位： Z:『不变』, C:『不变』, AC:『不变』, OV:『不变』
<i>stopexe</i>	CPU 停止。所有震荡器模块仍然继续工作并输出：但是系统时钟是被停用以节省功耗 例如： <i>stopexe</i>; 结果： 停住系统时钟，但是仍保持震荡器模块工作 受影响的标志位： Z:『不变』, C:『不变』, AC:『不变』, OV:『不变』
<i>reset</i>	复位整个单片机，其运行将与硬件复位相同 例如： <i>reset</i>; 结果： 复位整个单片机 受影响的标志位： Z:『不变』, C:『不变』, AC:『不变』, OV:『不变』
<i>wdreset</i>	复位看门狗 例如： <i>wdreset</i>; 结果： 复位看门狗 受影响的标志位： Z:『不变』, C:『不变』, AC:『不变』, OV:『不变』

PMS151

1KW OTP IO型单片机

7.8. 指令执行周期综述

2T		<i>goto, call, idxm, pcadd, ret, reti,</i>
2T	条件满足	<i>ceqsn, t0sn, t1sn, dzsn, izsn</i>
1T	条件不满足	
1T		Others

7.9. 指令影响标志综述

指令	Z	C	AC	OV	指令	Z	C	AC	OV	指令	Z	C	AC	OV
<i>mov a, l</i>	-	-	-	-	<i>mov M, a</i>	-	-	-	-	<i>mov a, M</i>	Y	-	-	-
<i>mov a, IO</i>	Y	-	-	-	<i>mov IO, a</i>	-	-	-	-	<i>ldt16 word</i>	-	-	-	-
<i>stt16 word</i>	-	-	-	-	<i>idxm a, index</i>	-	-	-	-	<i>idxm index, a</i>	-	-	-	-
<i>xch M</i>	-	-	-	-	<i>pushaf</i>	-	-	-	-	<i>popaf</i>	Y	Y	Y	Y
<i>add a, l</i>	Y	Y	Y	Y	<i>add a, M</i>	Y	Y	Y	Y	<i>add M, a</i>	Y	Y	Y	Y
<i>addc a, M</i>	Y	Y	Y	Y	<i>addc M, a</i>	Y	Y	Y	Y	<i>addc a</i>	Y	Y	Y	Y
<i>addc M</i>	Y	Y	Y	Y	<i>sub a, l</i>	Y	Y	Y	Y	<i>sub a, M</i>	Y	Y	Y	Y
<i>sub M, a</i>	Y	Y	Y	Y	<i>subc a, M</i>	Y	Y	Y	Y	<i>subc M, a</i>	Y	Y	Y	Y
<i>subc a</i>	Y	Y	Y	Y	<i>subc M</i>	Y	Y	Y	Y	<i>inc M</i>	Y	Y	Y	Y
<i>dec M</i>	Y	Y	Y	Y	<i>clear M</i>	-	-	-	-	<i>sr a</i>	-	Y	-	-
<i>src a</i>	-	Y	-	-	<i>sr M</i>	-	Y	-	-	<i>src M</i>	-	Y	-	-
<i>sl a</i>	-	Y	-	-	<i>slc a</i>	-	Y	-	-	<i>sl M</i>	-	Y	-	-
<i>slc M</i>	-	Y	-	-	<i>swap a</i>	-	-	-	-	<i>and a, l</i>	Y	-	-	-
<i>and a, M</i>	Y	-	-	-	<i>and M, a</i>	Y	-	-	-	<i>or a, l</i>	Y	-	-	-
<i>or a, M</i>	Y	-	-	-	<i>or M, a</i>	Y	-	-	-	<i>xor a, l</i>	Y	-	-	-
<i>xor IO, a</i>	-	-	-	-	<i>xor a, M</i>	Y	-	-	-	<i>xor M, a</i>	Y	-	-	-
<i>not a</i>	Y	-	-	-	<i>not M</i>	Y	-	-	-	<i>neg a</i>	Y	-	-	-
<i>neg M</i>	Y	-	-	-	<i>set0 IO.n</i>	-	-	-	-	<i>set1 IO.n</i>	-	-	-	-
<i>set0 M.n</i>	-	-	-	-	<i>set1 M.n</i>	-	-	-	-	<i>ceqsn a, l</i>	Y	Y	Y	Y
<i>ceqsn a, M</i>	Y	Y	Y	Y	<i>t0sn IO.n</i>	-	-	-	-	<i>t1sn IO.n</i>	-	-	-	-
<i>t0sn M.n</i>	-	-	-	-	<i>t1sn M.n</i>	-	-	-	-	<i>izsn a</i>	Y	Y	Y	Y
<i>dzsn a</i>	Y	Y	Y	Y	<i>izsn M</i>	Y	Y	Y	Y	<i>dzsn M</i>	Y	Y	Y	Y
<i>call label</i>	-	-	-	-	<i>goto label</i>	-	-	-	-	<i>ret l</i>	-	-	-	-
<i>ret</i>	-	-	-	-	<i>reti</i>	-	-	-	-	<i>nop</i>	-	-	-	-
<i>pcadd a</i>	-	-	-	-	<i>engint</i>	-	-	-	-	<i>disgint</i>	-	-	-	-
<i>stopsys</i>	-	-	-	-	<i>stopexe</i>	-	-	-	-	<i>reset</i>	-	-	-	-
<i>wdreset</i>	-	-	-	-										

7.10. 位定义

位寻址只能定义在 RAM 区地址的 0x00 to 0x0F。

PMS151

1KW OTP IO型单片机

8. 程序选项

选项	选择	描述
Security	启用	OTP 内容加密 7 / 8 words
	停用	OTP 内容不加密，程序可以被读取
Boot-up_Time	Slow	慢开机
	Fast	快开机

9. 特别注意事项

此章节提醒用户在使用 PMS151 系列 IC 时避免常犯的一些错误。

9.1. 使用 IC

9.1.1. IO 引脚的使用和设定

(1) IO 作为数字输入时

- ◆ IO 作为数字输入时，Vih 与 Vil 的准位，会随着电压与温度变化，请遵守 Vih 的最小值，Vil 的最大值规范
- ◆ 内部上拉电阻值将随着电压、温度与引脚电压而变动，并非为固定值

(2) IO 作为数字输入和打开唤醒功能

- ◆ 设置 IO 为输入
- ◆ 用 PADIER 和 PBDIER 寄存器，将对应的位设为 1

(3) PA5 设置为输出引脚

- ◆ PA5 只能做 Open Drain 输出，输出高需要外加上拉电阻

(4) PA5 设置为 PRSTB 输入引脚

- ◆ 设定 PA5 作输入
- ◆ 设定 CLKMD.0=1 来启用 PA5 作为 PRSTB 输入引脚

(5) PA5 作为输入并通过长导线连接至按键或者开关

- ◆ 必需在 PA5 与长导线中间串接 $>33\Omega$
- ◆ 应尽量避免使用 PA5 作为输入

PMS151

1KW OTP IO型单片机

9.1.2. 中断

(1) 使用中断功能的一般步骤如下：

步骤 1：设定 INTEN 寄存器，开启需要的中断的控制位

步骤 2：清除 INTRQ 寄存器

步骤 3：主程序中，使用 ENGINT 指令允许 CPU 的中断功能

步骤 4：等待中断。中断发生后，跳入中断子程序

步骤 5：当中断子程序执行完毕，返回主程序

*在主程序中，可使用DISGINT 指令关闭所有中断

* 跳入中断子程序处理时，可使用 PUSHAF 指令来保存 ALU 和 FLAG 寄存器数据，并在 RETI 之前，使用 POPAF 指令复原，步骤如下：

```
void Interrupt (void)    // 中断发生后，跳入中断子程序
{
    // 自动进入 DISGINT 的状态，CPU 不会再接受中断
    PUSHAF;
    ...
    POPAF;
} // 系统自动填入 RETI，直到执行 RETI 完毕才自动恢复到 ENGINT 的状态。
```

(2) INTEN, INTRQ 没有初始值，所以要使用中断前，一定要根据需要设定数值。

9.1.3. 系统时钟选择

利用 CLKMD 寄存器可切换系统时钟源。**请注意，不可在切换系统时钟源的同时把原时钟源关闭。**例如：从 A 时钟源切换到 B 时钟源时，应该先用 CLKMD 寄存器切换系统时钟源，然后再通过 CLKMD 寄存器关闭 A 时钟振荡源。

- ◆ 例一：系统时钟从 ILRC 切换到 IHRC/8

```
CLKMD = 0x34;           // 切到 IHRC，但 ILRC 不要停用
CLKMD.2 = 0;            // 此时才可关闭 ILRC
```
- ◆ 错误的写法：ILRC 切换到 IHRC/8，同时关闭 ILRC

```
CLKMD = 0x30;           // MCU 会死机
```

9.1.4. 掉电模式、唤醒和看门狗

当 ILRC 关闭时，看门狗也会失效。

9.1.5. TIMER 溢出

当设定 \$ INTEGS BIT_R 时（这是 IC 默认值），且设定 T16M 计数器 BIT8 产生中断，若 T16 计数从 0 开始，则第一次中断是在计数到 0x100 时发生（BIT8 从 0 到 1），第二次中断在计数到 0x300 时发生（BIT8 从 0 到 1）。所以设定 BIT8 是计数 512 次才中断。请注意，如果在中断中重新给 T16M 计数器设值，则下一次中断也将在 BIT8 从 0 变 1 时发生。

如果设定 \$ INTEGS BIT_F（BIT 从 1 到 0 触发）而且设定 T16M 计数器 BIT8 产生中断，则 T16 计数改为每次数到 0x200/0x400/0x600/...时发生中断。两种设定 INTEGS 的方法各有好处，也请注意其中差异。

PMS151

1KW OTP IO型单片机

9.1.6. IHRC

- (1) IHRC 频率会在使用烧录器烧录程序时校准。
- (2) 校准 IHRC 时，不管是封装片机台烧录还是 COB 在线烧录，EMC 的干扰都会对 IHRC 的精度有影响。如果在封装前校准了 IHRC，那么在封装后 IHRC 的实际频率可能会出现偏差并超出规格范围。通常封装后频率会变慢一点。
- (3) 频率偏离较大的情况一般发生在 COB 烧录或者 QTP 时。本公司对这种情况不承担责任。
- (4) 客户可根据自己的经验做一些调整，例如，可以将 IHRC 频率预先设置高 0.5%到 1%以让最终的实际频率更符合原来的期望。

9.1.7. PMS151 的烧录方法

请使用 5S-P-003 进行烧录。3S-P-002 或之前的 writer 版本皆已不支持烧录 PMS151。详细信息请依照5S-P-003-UM 的说明，连接jumper 即可。

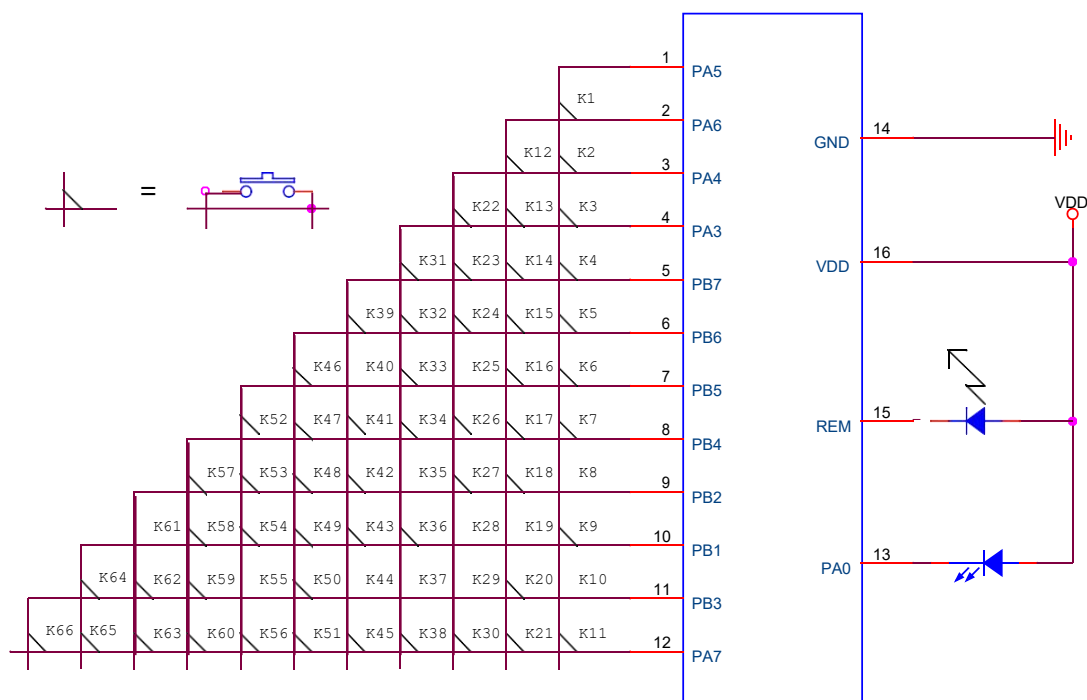
◆ 合封（MCP）或在板烧录（On-Board Writing）时的有关电压和电流的注意事项

- (1) PA5（V_{PP}）可能高于 8.5V。
- (2) V_{DD}可能高于 7V，而最大供给电流最高可达约 20mA。
- (3) 其他烧录引脚（GND 除外）的电位与 V_{DD} 相同。

请用户自行确认在使用本产品于合封或在板烧录时，周边电路及元件不会被上述电压破坏，也不会限制上述电压。

9.1.8. PMS151 应用参考原理图

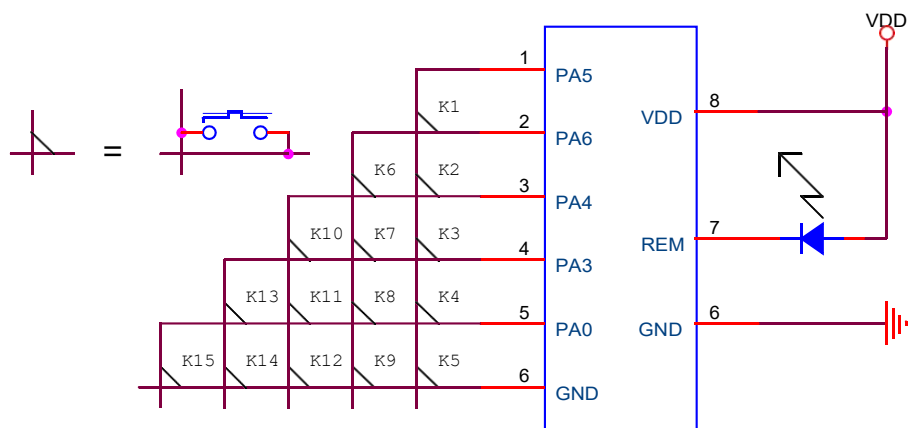
- (1)66 键带 LED 指示灯原理图，仅供参考。



PMS151

1KW OTP IO型单片机

(2)15 键原理图，仅供参考。



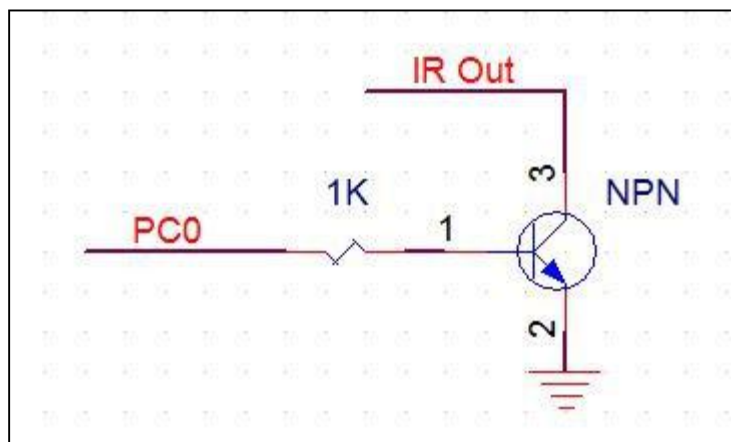
◆ 画 PCB 板注意事项:

- (1) 避免电源线、地线走线过长。
- (2) 建议发射电路的电源线和 IC 的电源线分开走线，或将线加粗。

9.2 使用 ICE

(1) 请使用 5S-I-S01/2(B)仿真器。仿真时请注意以下几点:

- ◆ 仿真器不支持系统时钟为 ILRC/2 和 ILRC/8 和 IHRC/64 的仿真
- ◆ 仿真器不支持PB6 当中断源
- ◆ 仿真器对所有 IRTMC / IRTMB / TSCNC / PAPH / PBPH / PATS / PBTS / PADIER / PBDIER / STOPEXE 功能都进行改写替换的动作，以期望能仿真 Key Scan 与 IR 功能，但也因此，会造成执行时序的偏慢
- ◆ IRTMB / IRTMC 的 REM 输出(IR Out)，可以用仿真器的 PC0 接 NPN 反向输出模拟出来



- ◆ 仿真时的快速唤醒时间和 IC 不一样；仿真器：128 个系统时钟； IC：32 ILRC
- ◆ 看门狗溢出时间仿真和 IC 不一样

PMS151

1KW OTP IO型单片机

看门狗溢出设置	5S-I-S01/2(B)	PMS15 1
misc[1:0]=00	2k * T _{ILRC}	4k * T _{ILRC}
misc[1:0]=01	4k * T _{ILRC}	8k * T _{ILRC}
misc[1:0]=10	16k * T _{ILRC}	32k * T _{ILRC}
misc[1:0]=11	256 * T _{ILRC}	128k * T _{ILRC}

ICE cycle	IO = A	A = IO	IO. bit = 0/1	If (IO. bit)
PADIER	6	-	-	-
PBDIER	7	-	-	-
PAPH	6	1	6	1/2
PBPH	6	1	6	1/2
PATS	1	1	1	1/2
PBTS	6	1	1	1/2
IRTCMC	61	1	61	1/2
IRTMB	61	-	61	-
TSCNC	8	1	1	1/2