PADAUK 應廣科技

RapiDragon（迅龙）

PFS123

8 位 MTP 型单片机带 12 位电阻式 ADC

数据手册

第 0.01 版

2023 年 5 月 16 日

Copyright ©2023 by PADAUK Technology Co., Ltd., all rights reserved.

6F-6, No.1, Sec. 3, Gongdao 5th Rd., Hsinchu City 30069, Taiwan, R.O.C.

TEL: 886-3-572-8688 www.padauk.com.tw

重要声明

应广科技保留权利在任何时候变更或终止产品，建议客户在使用或下单前与应广科技或代理商联系以取得最新、最正确的产品信息。

应广科技不担保本产品适用于保障生命安全或紧急安全的应用，应广科技不为此类应用产品承担任何责任。关键应用产品包括，但不仅限于，可能涉及的潜在风险的死亡，人身伤害，火灾或严重财产损失。

应广科技為服务客户所提供之任何编程软件，皆为服务与参考性质，不具备任何软件漏洞责任，应广科技不承担任何责任来自于因客户的产品设计所造成的任何损失。在应广科技所保障的规格范围内，客户应设计和验证他们的产品。为了尽量减少风险，客户设计产品时，应保留适当的产品工作范围安全保障。

提供本文档的中文简体版是为了便于了解，请勿忽视文中英文的部份，因为其中提供有关产品性能以及产品使用的有用信息，应广科技暨代理商对于文中可能存在的差错不承担任何责任，建议参考本文件英文版。

©Copyright 2023, PADAUK Technology Co. Ltd

目录

修订历史 7
使用警告 7
1. 单片机特性 8
1.1. 特性 8
1.2. 系统特性 8
1.3. CPU 特性 8
1.4. 订购/封装信息 8
2. 系统概述和方框图 9
3. 引脚功能说明 10
4. 器件电气特性 17
4.1. 直流交流电气特性 17
4.2. 绝对最大值 19
4.3. ILRC 频率与 VDD 的关系曲线图 19
4.4. IHRC 频率与 VDD 的关系曲线图（校准到 16MHz） 20
4.5. ILRC 频率与温度的关系曲线图 20
4.6. IHRC 频率与温度的关系曲线图（校准到 16MHz） 21
4.7. 工作电流与 VDD、系统时钟 CLK = ILRC/n 关系曲线图 21
4.8. 工作电流与 VDD、系统时钟 CLK = IHRC/n 关系曲线图 22
4.9. 工作电流与 VDD、系统时钟 CLK = 4MHz EOSC / n 关系曲线图 22
4.10. 工作电流与 VDD、系统时钟 CLK = 32KHz EOSC / n 关系曲线图 23
4.11. 工作电流与 VDD、系统时钟 CLK = 1MHz EOSC / n 关系曲线图 23
4.12. IO 引脚输出驱动电流(I_{OH}) 和灌电流 (I_{OL})曲线图 24
4.13. IO 引脚输入高/低阈值电压(V_{IH}/V_{IL})曲线图 26
4.14. IO 引脚上拉/下拉阻抗曲线图 26
4.15. 掉电模式消耗电流(I_{PD})与省电模式消耗电流(I_{PS})曲线图 28
5. 功能概述 29
5.1. 程序存储器 - MTP 29
5.2. 开机流程 29
5.2.1.复位时序图 30
5.3. 数据存储器 - SRAM 31
5.4. 振荡器和时钟 31
5.4.1. 内部高频振荡器和低频振荡器 31
5.4.2. 芯片校准 31

©Copyright 2023, PADAUK Technology Co. Ltd

5.4.3. IHRC 频率校准与系统时钟 32
5.4.4. 外部晶体振荡器 33
5.4.5. 系统时钟和 LVR 基准位 35
5.4.6. 系统时钟切换 36
5.5. 比较器 37
5.5.1 内部参考电压 ($V_{internal\ R}$) 38
5.5.2 使用比较器 40
5.5.3 使用比较器和 bandgap 1.20V 41
5.6 VDD/2 LCD Bias 电压生成器 42
5.7 16 位计数器(Timer16) 43
5.8 8 位 PWM 计数器 (Timer2, Timer3) 45
5.8.1 使用 Timer2 产生周期波形 46
5.8.2 使用 Timer2 产生 8 位 PWM 波形 48
5.8.3 使用 Timer2 产生 6 位 PWM 波形 49
5.9 11 位 PWM 计数器 50
5.9.1 PWM 波形 50
5.9.2 硬件框图 51
5.9.3 11 位 PWM 生成器计算公式 52
5.9.4 带互补死区的 PWM 波形范例 53
5.10 看门狗 55
5.11 中断 56
5.12 省电与掉电 58
5.12.1 省电模式(“*stopexe*”) 58
5.12.2 掉电模式 (“*stopsys*”) 59
5.12.3 唤醒 60
5.13 IO 引脚 61
5.14 复位和 LVR 62
5.14.1 复位 62
5.14.2 LVR 复位 62
5.15 模拟-数字转换器(ADC) 模块 62
5.15.1 AD 转换的输入要求 63
5.15.2 选择参考高电压 64
5.15.3 ADC 时钟选择 64
5.15.4 配置模拟引脚 64
5.15.5 使用 ADC 64
5.15.6 如何计算 ADC 输入电压 V_{IN} 65
6. IO 寄存器 66

6.1. ACC 状态标志寄存器(*flag*), IO 地址 = 0x00 66
6.2. 堆栈指针寄存器(*sp*), IO 地址 = 0x02 66
6.3. 时钟模式寄存器(*clkmd*), IO 地址 = 0x03 66
6.4. 中断允许寄存器(*inten*), IO 地址 = 0x04 67
6.5. 中断请求寄存器(*intrq*), IO 地址 = 0x05 67
6.6. Timer16 控制寄存器 (*t16m*), IO 地址 = 0x06 68
6.7. 外部晶体振荡器控制寄存器(*eoscr*), IO 地址 = 0x0a 68
6.8. 中断边缘选择寄存器(*integs*), IO 地址 = 0x0c 69
6.9. 端口 A 数字输入使能寄存器(*padier*), IO 地址 = 0x0d 69
6.10. 端口 B 数字输入使能寄存器(*pbdier*), IO 地址 = 0x0e 70
6.11. 端口 C 数字输入使能寄存器(*pcdier*), IO 地址= 0x0f 70
6.12. 端口 A 数据寄存器(*pa*), IO 地址 = 0x10 70
6.13. 端口 A 控制寄存器(*pac*), IO 地址 = 0x11 70
6.14. 端口 A 上拉控制寄存器(*paph*), IO 地址 = 0x12 70
6.15. 端口 B 数据寄存器(*pb*), IO 地址 = 0x13 70
6.16. 端口 B 控制寄存器(*pbc*), IO 地址 = 0x14 71
6.17. 端口 B 上拉控制寄存器(*pbph*), IO 地址 = 0x15 71
6.18. 端口 C 数据寄存器((*pc*), IO 地址= 0x16 71
6.19. 端口 C 控制寄存器 (*pcc*), IO 地址= 0x17 71
6.20. 端口 C 上拉控制寄存器(*pcph*), IO 地址= 0x18 71
6.21. 端口 B 下拉控制寄存器 (*pbpl*), IO 地址= 0x19 71
6.22. 端口 C 下拉控制寄存器(*pcpl*), IO 地址= 0x1a 71
6.23. ADC 控制寄存器(*adcc*), IO 地址 = 0x20 72
6.24. ADC 模式寄存器(*adcm*), IO 地址 = 0x21 72
6.25. ADC 高位结果寄存器(*adcrh*), IO 地址= 0x22 72
6.26. ADC 低位结果寄存器 (*adcrl*), IO 地址 = 0x23 73
6.27. ADC 调节控制寄存器(*adcrgc*), IO 地址 = 0x24 73
6.28. 杂项寄存器(*misc*), IO 地址 = 0x26 73
6.29. 比较器控制寄存器(*gpcc*), IO 地址 = 0x2b 74
6.30. 比较器选择寄存器(*gpcs*), IO 地址 = 0x2c 74
6.31. Timer2 控制寄存器(*tm2c*), IO 地址 = 0x30 75
6.32. Timer2 计数寄存器(*tm2ct*), IO 地址 = 0x31 75
6.33. Timer2 分频寄存器(*tm2s*), IO 地址 = 0x32 75
6.34. Timer2 上限寄存器(*tm2b*), IO 地址 = 0x33 76
6.35. Timer3 控制寄存器(*tm3c*), IO 地址 = 0x34 76
6.36. Timer3 计数寄存器(*tm3ct*), IO 地址 = 0x35 76
6.37. Timer3 分频寄存器 (*tm3s*), IO address = 0x36 77
6.38. Timer3 上限寄存器(*tm3b*), IO 地址 = 0x37 77

©Copyright 2023, PADAUK Technology Co. Ltd

6.39. PWMG0 控制寄存器(*pwmg0c*), IO 地址 = 0x40 77
6.40. PWMG 时钟寄存器(*pwmgclk*), IO 地址 = 0x41 78
6.41. PWMG0 占空比高位寄存器(*pwmg0dth*), IO 地址 = 0x42 78
6.42. PWMG0 占空比低位寄存器((*pwmg0dtl*), IO 地址 = 0x43 78
6.43. PWMG 计数上限高位寄存器(*pwmgcubh*), IO 地址 = 0x44 78
6.44. PWMG 计数上限低位寄存器(*pwmgcubl*), IO 地址= 0x45 78
6.45. PWMG1 控制寄存器 (*pwmg1c*), IO 地址= 0x46 79
6.46. PWMG1 占空比高位寄存器 (*pwmg1dth*), IO 地址= 0x48 79
6.47. PWMG1 占空比低位寄存器(*pwmg1dtl*), IO 地址= 0x49 79
6.48. PWMG2 控制寄存器(*pwmg2c*), IO 地址 = 0x4C 80
6.49. PWMG2 占空比高位寄存器(*pwmg2dth*), IO 地址= 0x4E 80
6.50. PWMG2 占空比低位寄存器(*pwmg2dtl*), IO 地址= 0x4F 80

7. 指令 81

7.1. 数据传输类指令 82
7.2. 算术运算类指令 86
7.3. 移位运算类指令 88
7.4. 逻辑运算类指令 89
7.5. 位运算类指令 92
7.6. 条件运算类指令 93
7.7. 系统控制类指令 94
7.8. 指令执行周期综述 96
7.9. 指令影响标志综述 96
7.10. 位定义 97

8. 程序选项(Code Options) 97

9. 特别注意事项 99

9.1. 使用 IC 99
9.1.1. IO 引脚的使用和设定 99
9.1.2. 中断 100
9.1.3. 系统时钟切换 100
9.1.4. 看门狗 100
9.1.5. TIMER 溢出 101
9.1.6. IHRC 101
9.1.7. LVR 101
9.1.8. 烧录方法 102
9.2. 使用 ICE 104

©Copyright 2023, PADAUK Technology Co. Ltd

修订历史

修订	日期	描述
0.00	2020/08/26	初版
0.01	2023/05/16	1. 更新”重要声明” 2. 修改工作温度范围 3. 更新 I_{OP} 与 I_{PD} 特性（第 4.1 节） 4. 修改 1.2、1.3、5.8、5.11、5.12.1、5.14.1、6.7、6.9、6.29、7.1、7.9、9.1.1、9.1.4、9.1.7、9.2 5. 修改图 3、图 11、图 15、表 5 6. 其他已知细节错误修正

使用警告

在使用 IC 前，请务必认真阅读 PMB180 相关的 APN（应用注意事项）。

请至官网下载查看与之关联的最新 APN 资讯：

http://www.padauk.com.tw/cn/product/show.aspx?num=145&kw=PFS123

（下列图示仅供参考，依官网为主。）

Feature Documents Software &Tools Application Note

内容	說明	中文下载	英文下载
APN001	ADC仿真信号源输出阻抗应用须知		
APN002	过压保护应用须知		
APN003	IO输出引脚连接长导线时的应用须知		
APN004	半自动烧录机台使用需知		
APN005	过电压输入对ADC的影响使用须知		
APN007	设置LVR时的使用须知		
APN011	半自动烧录机台提高烧录稳定性		
APN013	晶振使用须知		
APN019	E-PAD 产品的PCB布局指南		

©Copyright 2023, PADAUK Technology Co. Ltd

1. 单片机特性

1.1. 特性

- 通用系列
- 不建议使用于 AC 阻容降压供电或有高 EFT 要求的应用。应广不对使用于此类应用而不达安规要求负责
- 工作温度范围：-20°C ~ 85°C

1.2. 系统特性

- 3KW MTP 程序空间（可编程 1000 次以上）
- 256 Bytes 数据空间
- 一个硬件 16 位定时器
- 两个 8 位带 PWM 功能的定时器
- 一组可设三路 11 位 SuLED (Super LED) PWM 生成器和计数器
- 一个硬件比较器
- Bandgap 电路提供 1.20V 参考电压
- 高达 14 通道 12 位精度电阻式 ADC（R-ADC*）（其中一个通道来自内部 bandgap 电压）
  *注：电阻式 ADC 在线性特性（INL&DNL）、热稳定度及抗电源波动能力上，皆略逊于 PMS13x 系列内建的电容式 ADC，请选型时注意。
- ADC 参考高电压：外部输入，内部 VDD
- 最大 18 IO 引脚带可选择的上拉电阻，其中 4 个 IO 引脚同时带下拉电阻
- 每个 IO 引脚都可设定为唤醒功能
- 内设 VDD/2 LCD bias 电压生成器以提供最大 5x12 点阵 LCD 显示
- 时钟源：IHRC、ILRC 和 EOSC (XTAL)
- 每个能唤醒的 IO 均支持两种可选的唤醒速度：正常和快速
- 8 级可选择的 LVR 复位电压从 1.8V 到 4.0V
- 两个可选择的外部中断引脚

1.3. CPU 特性

- 8 位高性能精简指令集 CPU
- 90 个高效指令
- 绝大部分指令都是单周期(1T)指令
- 可程序设定的堆栈指针和堆栈深度
- 数据存取支持直接和间接寻址模式，用数据存储器即可當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. 订购/封装信息

- PFS123-U06: SOT23-6 (60mil)
- PFS123-S08: SOP8 (150mil)
- PFS123-M10: MSOP10 (118mil)
- PFS123-S14: SOP14 (150mil)
- PFS123-S16: SOP16 (150mil)
- PFS123-D16: DIP16 (300mil)
- PFS123-S20: SOP20 (300mil)
- PFS123-H20: HTSOP20 (150mil)
- PFS123-1J16A: QFN3*3-16P (0.5pitch)

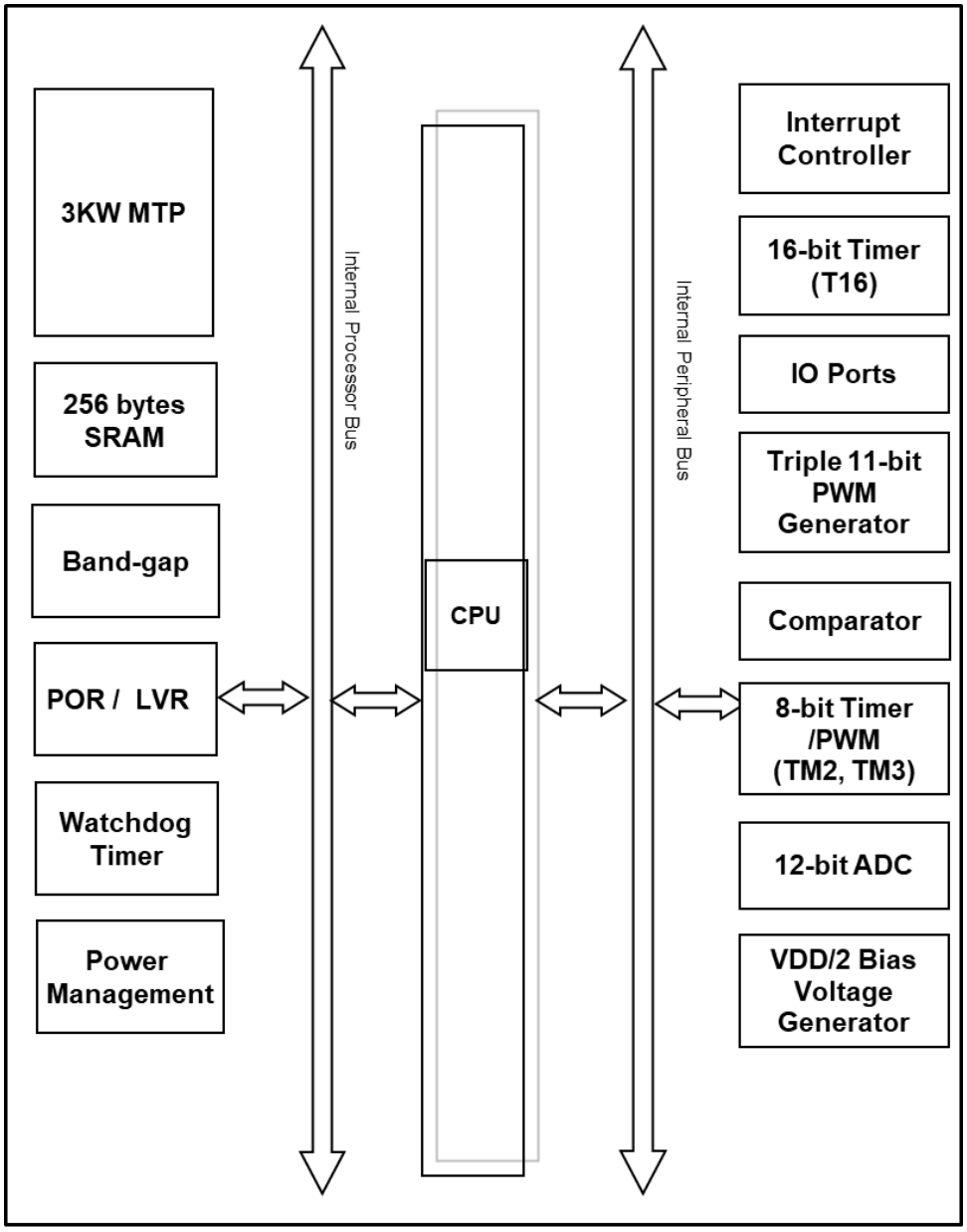
- 封装尺寸信息请参考官网文件：”封装信息“

©Copyright 2023, PADAUK Technology Co. Ltd

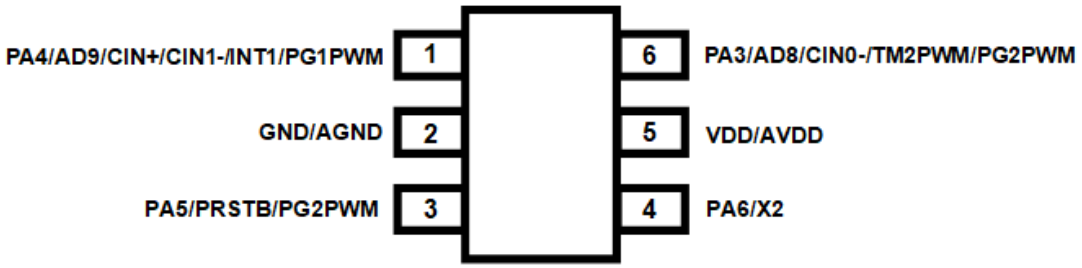
2. 系统概述和方框图

PFS123 是一款基于 CMOS 带 8 位 ADC 的 MTP 的 8 位微控制器。它运用 RISC 的架构基础使大部分的指令执行时间都是一个指令周期，只有少部分间接地址访问的指令是需要两个指令周期。

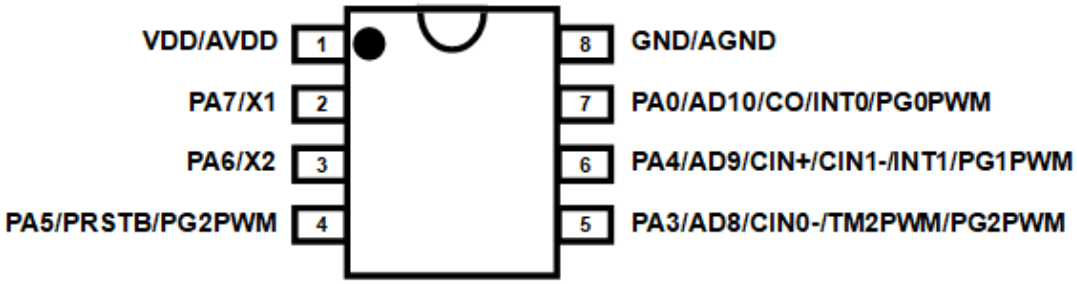
PFS123 内置高达 3KW MTP 程序存储器和 256 bytes 数据存储器，内设一个高达 14 通道带多路参考电压可选的 12 位电阻式 ADC 转换器。PFS123 同时提供 4 个硬件计数器：一个是 16 位定时器，两个 8 位计数器带 PWM 生成器和一组新的三路 11 位计数器带 SuLED PWM 生成器(PWMG0、PWMG1 & PWMG2)。PFS123 也支持一个硬件比较器和 VDD/2 LCD bias 电压生成器作为 LCD 显示应用。



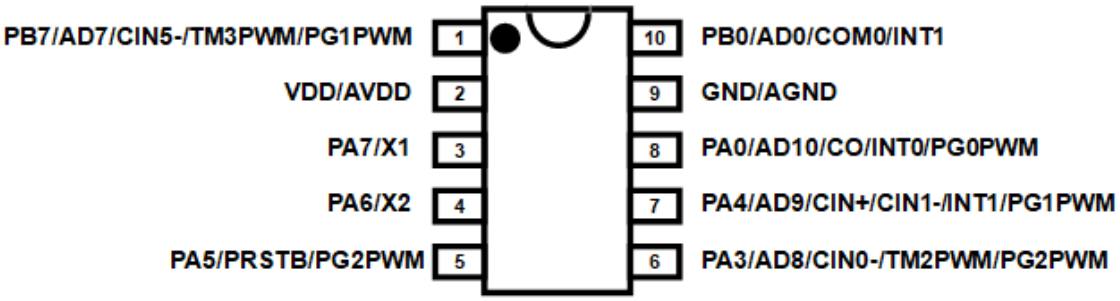
3. 引脚功能说明



PFS123-U06 (SOT23-6 60mil)

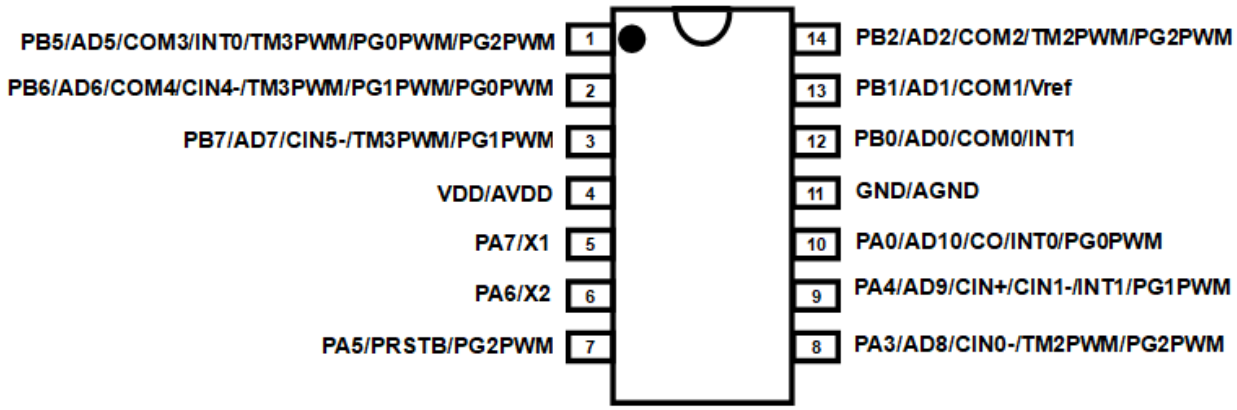


PFS123-S08 (SOP8-150mil)

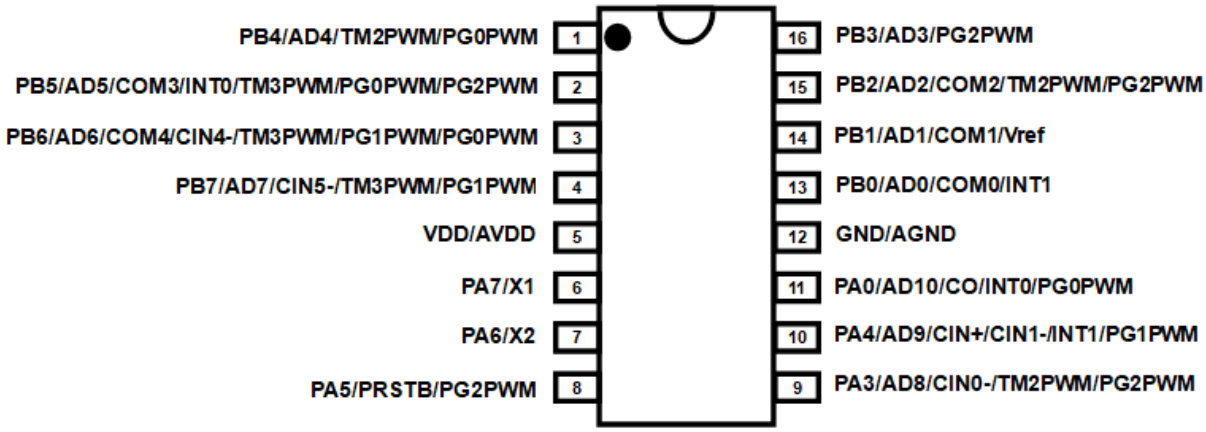


PFS123-M10 (MSOP10-118mil)

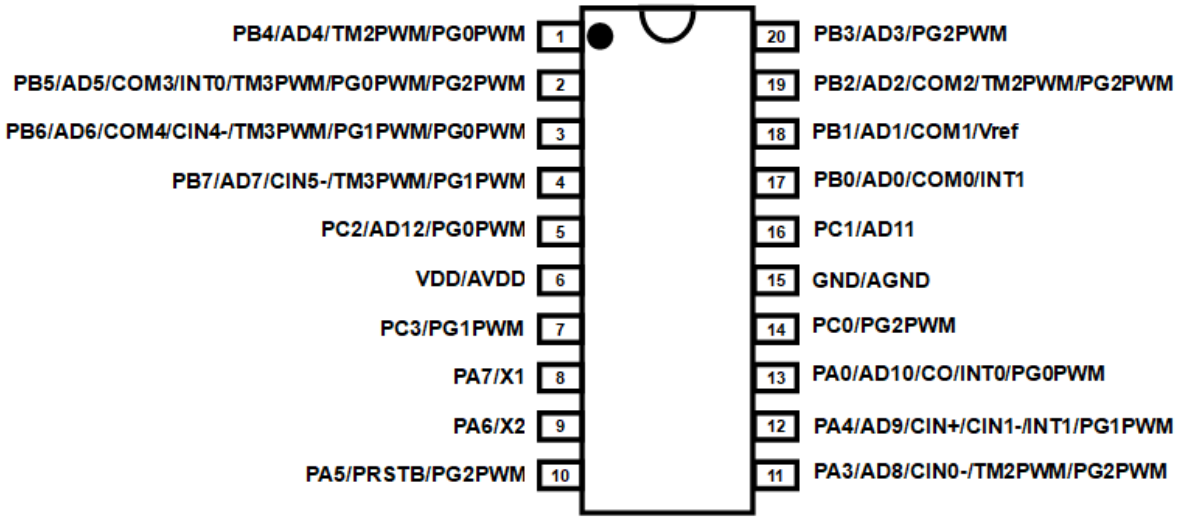
©Copyright 2023, PADAUK Technology Co. Ltd



PFS123-S14 (SOP14-150mil)

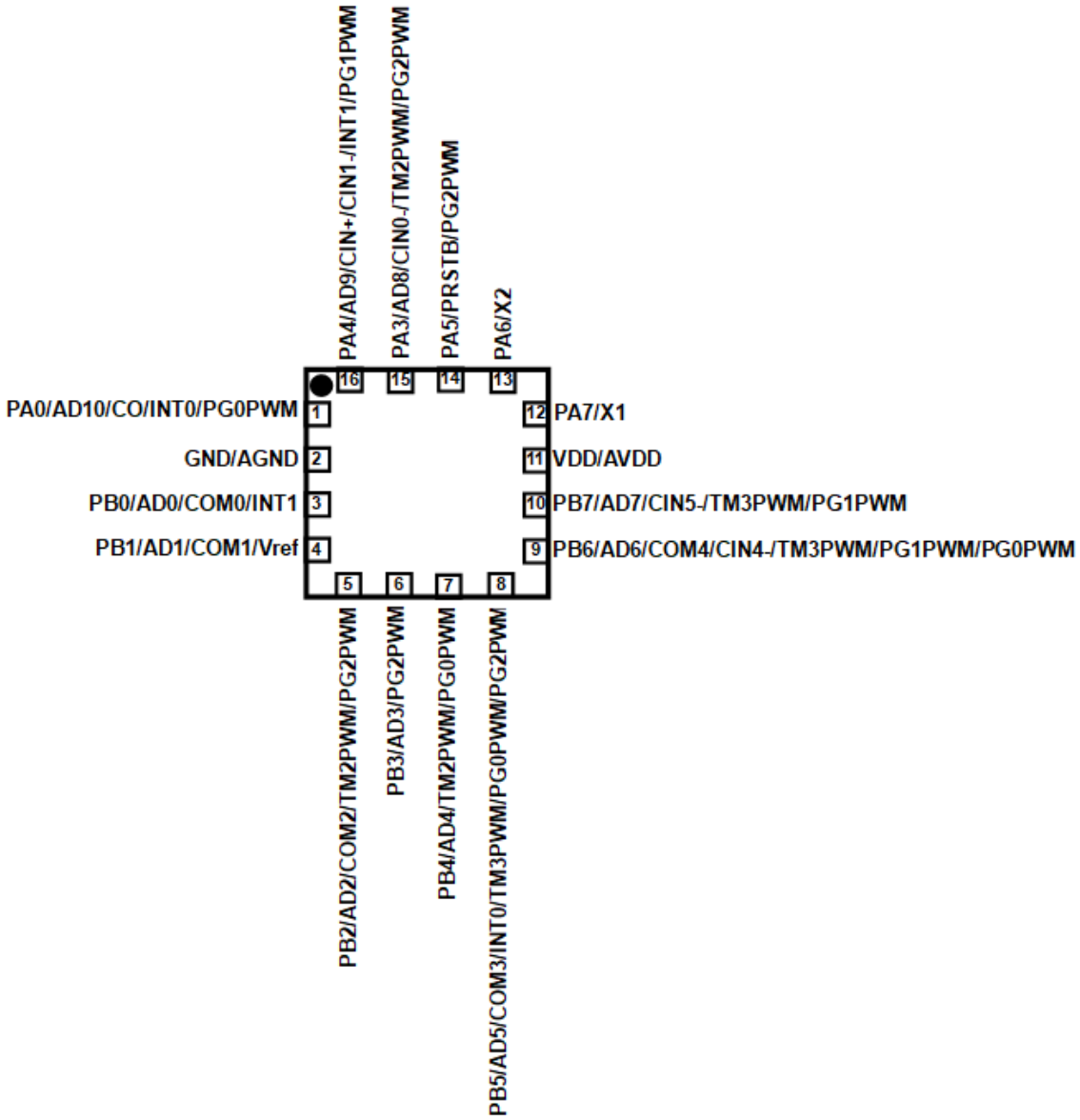


PFS123-S16 (SOP16-150mil)
PFS123-D16 (DIP16-300mil)



PFS123-S20 (SOP20-300mil)
PFS123-H20 (HTSOP20-150mil)

©Copyright 2023, PADAUK Technology Co. Ltd



PFS123-1J16A (QFN3*3-16P-0.5pitch)

引脚名称	引脚&缓冲器类型	功能描述
PA7 / X1	IO ST / CMOS	此引脚可用做： (1) 端口 A 位 7，并可编程设定为输入或输出，弱上拉电阻模式。 (2) 当使用外部晶体振荡器时，做为 X1 引脚。 当用做晶体振荡器的功能时，为减少漏电流，请用 *padier* 寄存器位 7 关闭其数字输入功能，这个引脚可以设定在睡眠中唤醒系统的功能；但当寄存器 *padier* 位 7 为”0”时，唤醒功能是被关闭的。
PA6 / X2	IO ST / CMOS	此引脚可用做： (1) 端口 A 位 6，并可编程设定为输入或输出，弱上拉电阻模式。 (2) 当使用外部晶体振荡器时，做为 X2 引脚。 当用做晶体振荡器的功能时，为减少漏电流，请用 *padier* 寄存器位 6 关闭其数字输入功能，这个引脚可以设定在睡眠中唤醒系统的功能；当寄存器 *padier* 位 6 为”0”时，唤醒功能是被关闭的。
PA5 / PRSTB / PG2PWM	IO (OD) ST / CMOS	此引脚可用做： (1) 端口 A 位 5，并可编程设定为输入或开漏输出(open drain)，弱上拉电阻模式。 (2) 硬件复位脚。 (3) 11 位计数器 PWMG2 的输出。 这个引脚可以设定在睡眠中唤醒系统的功能；但是，当寄存器 *padier* 位 5 为”0”时，唤醒功能是被关闭的。当此引脚设定成输入时，对于需要高抗干扰能力的系统，请串接 33Ω 电阻。
PA4 / AD9 / CIN+ / CIN1- / INT1 / PG1PWM	IO ST / CMOS / Analog	此引脚可用做： (1) 端口 A 位 4，并可编程设定为输入或输出，弱上拉电阻模式。 (2) ADC 模拟输入通道 9。 (3) 比较器的正输入源。 (4) 比较器的负输入源 1。 (5) 外部中断源 1。上升沿和下降沿均可触发中断。 (6) 11 位计数器 PWMG1 的输出。 当用做模拟输入功能时，为减少漏电流，请用 *padier* 寄存器位 4 关闭其数字输入功能。这个引脚可以设定在睡眠中唤醒系统的功能；当寄存器 *padier* 位 4 为”0”时，唤醒功能是被关闭的。
PA3 / AD8 / CIN0- / TM2PWM / PG2PWM	IO ST / CMOS / Analog	此引脚可用做： (1) 端口 A 位 3，并可编程设定为输入或输出，弱上拉电阻模式。 (2) ADC 模拟输入通道 8。 (3) 比较器的负输入源 0。 (4) Timer2 的 PWM 输出。 (5) 11 位计数器 PWMG2 的输出 。 当用做模拟输入功能时，为减少漏电流，请用 padier 寄存器位 3 关闭其数字输入功能。这个引脚可以设定在睡眠中唤醒系统的功能；当寄存器 *padier* 位 3 为”0”时，唤醒功能是被关闭的。

引脚名称	引脚&缓冲器类型	功能描述
PA0 / AD10 / CO / INT0 / PG0PWM	IO ST / CMOS / Analog	此引脚可用做： (1) 端口 A 位 0，并可编程设定为输入或输出，弱上拉电阻模式。 (2) ADC 模拟输入通道 10。 (3) 比较器的输出。 (4) 外部中断源 0，上升沿和下降沿都可触发中断。 (5) 11 位计数器 PWMG0 的输出。 当用做模拟输入功能时，为减少漏电流，请用 padier 寄存器位 0 关闭其数字输入功能。这个引脚可以设定在睡眠中唤醒系统的功能；当寄存器 padier 位 0 为”0”时，唤醒功能是被关闭的。
PB7 / AD7 / CIN5- / TM3PWM / PG1PWM	IO ST / CMOS / Analog	此引脚可用做： (1) 端口 B 位 7，并可编程设定为输入或输出，弱上拉电阻模式。 (2) ADC 模拟输入通道 7。 (3) 比较器的负输入源 5。 (4) Timer3 的 PWM 输出。 (5) 11 位计数器 PWMG1 的输出。 当用做模拟输入功能时，为减少漏电流，请用 *pbdier* 寄存器位 7 关闭其数字输入功能。这个引脚可以设定在睡眠中唤醒系统的功能；当寄存器 *pbdier* 位 7 为”0”时，唤醒功能是被关闭的。
PB6 / AD6 / COM4 / CIN4- / TM3PWM / PG1PWM / PG0PWM	IO ST / CMOS / Analog	此引脚可用做： (1) 端口 B 位 6，并可编程设定为输入或输出，弱上拉电阻模式。 (2) ADC 模拟输入通道 6。 (3) COM4 口，提供 1/2 V_{DD} 驱动 LCD 驱动显示。 (4) 比较器的负输入源 4。 (5) Timer3 的 PWM 输出。 (6) 11 位计数器 PWMG1 的输出。 (7) 11 位计数器 PWMG0 的输出。 当用做模拟输入功能时，为减少漏电流，请用 *pbdier* 寄存器位 6 关闭其数字输入功能。这个引脚可以设定在睡眠中唤醒系统的功能；当寄存器 *pbdier* 位 6 为”0”时，唤醒功能是被关闭的。
PB5 / AD5 / COM3 / INT0 / TM3PWM / PG0PWM / PG2PWM	IO ST / CMOS / Analog	此引脚可用做： (1) 端口 B 位 5，并可编程设定为输入或输出，弱上拉电阻模式。 (2) ADC 模拟输入通道 5。 (3) COM3 口，提供 1/2 VDD 驱动 LCD 驱动显示。 (4) 外部中断源 0，上升沿和下降沿都可触发中断。 (5) Timer3 的 PWM 输出。 (6) 11 位计数器 PWMG0 的输出。 (7) 11 位计数器 PWMG2 的输出。 当用做模拟输入功能时，为减少漏电流，请用 pbdier 寄存器位 5 关闭其数字输入功能。这个引脚可以设定在睡眠中唤醒系统的功能；当寄存器 pbdier 位 5 为”0”时，唤醒功能是被关闭的。

引脚名称	引脚&缓冲器类型	功能描述
PB4 / AD4 / TM2PWM / PG0PWM	IO ST / CMOS / Analog	此引脚可用做： (1) 端口 B 位 4，并可编程设定为输入或输出，弱上拉电阻模式。 (2) ADC 模拟输入通道 4。 (3) Timer2 的 PWM 输出。 (4) 11 位计数器 PWMG0 的输出。 当用做模拟输入功能时，为减少漏电流，请用 pbdier 寄存器位 4 关闭其数字输入功能。 这个引脚可以设定在睡眠中唤醒系统的功能；当寄存器 pbdier 位 4 为”0”时，唤醒功能是被关闭的。
PB3 / AD3 / PG2PWM	IO ST / CMOS / Analog	此引脚可用做： (1) 端口 B 位 3.并可编程设定为输入或输出，弱上拉/弱下拉电阻模式。 (2) ADC 模拟输入通道 3。 (3) 11 位计数器 PWMG2 的输出。 当用做模拟输入功能时，为减少漏电流，请用 *pbdier* 寄存器位 3 关闭其数字输入功能。 这个引脚可以设定在睡眠中唤醒系统的功能；当寄存器 *pbdier* 位 3 为”0”时，唤醒功能是被关闭的。
PB2 / AD2 / COM2 / TM2PWM / PG2PWM	IO ST / CMOS / Analog	此引脚可用做： (1) 端口 B 位 2，并可编程设定为输入或输出，弱上/弱下拉拉电阻模式。 (2) ADC 模拟输入通道 2。 (3) COM2 口，提供 1/2 VDD 驱动 LCD 驱动显示。 (4) Timer2 的 PWM 输出。 (5) 11 位计数器 PWMG2 的输出。 当用做模拟输入功能时，为减少漏电流，请用 pbdier 寄存器位 2 关闭其数字输入功能。 这个引脚可以设定在睡眠中唤醒系统的功能；当寄存器 pbdier 位 2 为”0”时，唤醒功能是被关闭的。
PB1 / AD1 / COM1 / Vref	IO ST / CMOS / Analog	此引脚可用做： (1) 端口 B 位 1，并可编程设定为输入或输出，弱上拉电阻模式。 (2) ADC 模拟输入通道 1。 (3) COM1 口，提供 1/2 VDD 驱动 LCD 驱动显示。 (4) ADC 的外部参考高电压。 当用做模拟输入功能时，为减少漏电流，请用 pbdier 寄存器位 1 关闭其数字输入功能。 这个引脚可以设定在睡眠中唤醒系统的功能；当寄存器 pbdier 位 1 为”0”时，唤醒功能是被关闭的。
PB0 / AD0 / COM0 / INT1	IO ST / CMOS / Analog	此引脚可用做： (1) 端口 B 位 0，并可编程设定为输入或输出，弱上拉电阻模式。 (2) ADC 模拟输入通道 0。 (3) COM0 口，提供 1/2 VDD 驱动 LCD 驱动显示。 (4) 外部中断源 1，上升沿和下降沿都可触发中断。 当用做模拟输入功能时，为减少漏电流，请用 pbdier 寄存器位 0 关闭其数字输入功能。 这个引脚可以设定在睡眠中唤醒系统的功能；当寄存器 pbdier 位 0 为”0”时，唤醒功能是被关闭的。

引脚名称	引脚&缓冲器类型	功能描述
PC3 / PG1PWM	IO ST / CMOS	此引脚可用做： (1) 端口 C 位 3，并可编程设定为输入或输出，弱上拉电阻模式。 (2) 11 位计数器 PWMG1 的输出。 pcdier 寄存器位 3 设”0”关闭数字输入功能，同时唤醒功能是被关闭的。
PC2 / AD12 / PG0PWM	IO ST / CMOS / Analog	此引脚可用做： (1) 端口 C 位 2，并可编程设定为输入或输出，弱上拉电阻模式。 (2) ADC 模拟输入通道 12。 (3) 11 位计数器 PWMG0 的输出。 当用做模拟输入功能时，为减少漏电流，请用 pcdier 寄存器位 2 关闭其数字输入功能。这个引脚可以设定在睡眠中唤醒系统的功能；当寄存器 pcdier 位 2 为”0”时，唤醒功能是被关闭的。
PC1 / AD11	IO ST / CMOS / Analog	此引脚可用做： (1) 端口 C 位 1，并可编程设定为输入或输出，弱上拉/弱下拉电阻模式。 (2) ADC 模拟输入通道 11。 当用做模拟输入功能时，为减少漏电流，请用 *pcdier* 寄存器位 1 关闭其数字输入功能。这个引脚可以设定在睡眠中唤醒系统的功能；当寄存器 *pcdier* 位 1 为”0”时，唤醒功能是被关闭的。
PC0 / PG2PWM	IO ST / CMOS	此引脚可用做： (1) 端口 C 位 1，并可编程设定为输入或输出，弱上拉/弱下拉电阻模式。 (2) 11 位计数器 PWMG2 的输出。 *pcdier* 寄存器位 0 设”0”关闭数字输入功能，同时唤醒功能是被关闭的。
VDD / AVDD	VDD / AVDD	VDD：数字正电源 AVDD：模拟正电源 VDD 是 IC 电源，而 AVDD 是 ADC 专用电源。在 IC 内部，AVDD 与 VDD 连在一起(double bonding)，而外部为相同引脚。
GND / AGND	GND / AGND	GND：数字负电源 AGND：模拟负电源 GND 是 IC 接地引脚，而 AGND 是 ADC 接地引脚。在 IC 内部，AGND 与 GND 连在一起(double bonding)，而外部为相同引脚。
注意：IO：输入/输出；**ST**：施密特触发器输入；**OD**：开漏输出；**Analog**：模拟输入引脚； **CMOS**：CMOS 电压基准位		

4. 器件电气特性

4.1. 直流交流电气特性

下列所有数据除特别列明外，皆于 Ta = -20ºC ~ 85 ºC，V_{DD}=5.0V，f_{SYS}=2MHz 之条件下获得。

符号	特性	最小值	典型值	最大值	单位	条件(Ta=25ºC)
V_{DD}	工作电压	2.2#	5.0	5.5	V	# 受限于 LVR 公差
LVR%	低电压复位公差	-5		5	%	
f_{SYS}	系统时钟(CLK)* = IHRC/2 IHRC/4 IHRC/8 ILRC	 0 0 0	 93K	 8M 4M 2M	Hz	 V_{DD} ≧ 3.5V V_{DD} ≧ 2.5V V_{DD} ≧ 2.2V V_{DD} = 5.0V
P_{cycle}	烧录次数	1000			cycles	
V_{POR}	上电复位电压		2.0*			* 由 LVR 的精度而定
I_{OP}	工作电流		0.75 87		mA uA	f_{SYS}=IHRC/16=1MIPS@5.0V f_{SYS}=ILRC=96KHz@5.0V
I_{PD}	掉电模式消耗电流 （使用 ***stopsys*** 命令）		1 0.5		uA uA	f_{SYS}= 0Hz,V_{DD}=5.0V f_{SYS}= 0Hz,V_{DD}=3.3V
I_{PS}	省电模式消耗电流 （使用 ***stopexe*** 命令）		4		uA	V_{DD} =5.0V; f_{SYS}= ILRC 仅使用 ILRC 的模式下
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
I_{OL}	IO 灌电流					
	PB4, PB5（正常） PB4, PB5（强） PC0, PC2, PC3 其他 IO		10 40 30 10		mA	V_{DD}=5.0V, V_{OL}=0.5V
I_{OH}	IO 驱动电流					
	PB4, PB5（正常） PB4, PB5（强） PA5 其他 IO		5 20 0 5		mA	V_{DD}=5.0V, V_{OH}=4.5V
V_{IN}	输入电压	-0.3		V_{DD}+0.3	V	
$I_{INJ\ (PIN)}$	脚位的引入电流			1	mA	V_{DD} +0.3≧V_{IN}≧ -0.3
R_{PH}	上拉电阻		76 100 52 200 30 450		KΩ	V_{DD} =5.0V, PB2/PB3/PC0/PC1 V_{DD} =5.0V, 其他 IO V_{DD} =3.3V, PB2/PB3/PC1/PC1 V_{DD} =3.3V, 其他 IO V_{DD} =2.2V, PB2/PB3/PC1/PC1 V_{DD} =2.2V, 其他 IO
R_{PL}	下拉电阻		65 45 28		KΩ	V_{DD} =5.0V, PB2/PB3/PC0/PC1 V_{DD} =3.3V, PB2/PB3/PC0/PC1 V_{DD} =2.2V, PB2/PB3/PC0/PC1

©Copyright 2023, PADAUK Technology Co. Ltd

符号	特性	最小值	典型值	最大值	单位	条件(Ta=25°C)
V_{BG}	Bandgap 参考电压	1.145*	1.20*	1.255*	V	V_{DD} =2.2V ~ 5.5V -20°C <Ta<85°C*
f_{IHRC}	IHRC 输出频率（校准后）*	15.76*	16*	16.24*	MHz	25°C, V_{DD} =2.2V~5.5V
		15.20*	16*	16.80*		V_{DD} =2.2V~5.5V, 0°C <Ta<85°C*
t_{INT}	中断脉冲宽度	30			ns	V_{DD} = 5.0V
V_{ADC}	ADC 工作电压	2.2		V_{DD}	V	
V_{AD}	AD 输入电压	0		V_{DD}	V	
ADrs	ADC 分辨率		12 10		bit	0°C <Ta<50°C* -20°C <Ta<85°C*
ADcs	ADC 消耗电流		0.9 0.8		mA	@5V @3V
ADclk	ADC 时钟周期		2		us	2.2V ~ 5.5V
t_{ADCONV}	ADC 转换时间 （t_{ADCLK} 是 AD 转换时钟周期）		16		t_{ADCLK}	12 位分辨率
AD DNL	ADC 微分非线性		±4*		LSB	12 位分辨率 LSB
AD INL	ADC 积分非线性		±8*		LSB	12 位分辨率 LSB
ADos	ADC 失调电压		5*		mV	@ V_{DD} =3V
V_{DR}	RAM 数据保留电压*	1.5			V	在关机模式下
t_{WDT}	看门狗超时溢出时间		8k		T_{ILRC}	misc[1:0]=00（默认）
			16k			misc[1:0]=01
			64k			misc[1:0]=10
			256k			misc[1:0]=11
t_{WUP}	快速唤醒时间		45		T_{ILRC}	T_{ILRC} 是 ILRC 的时钟周期
	慢速唤醒时间		3000			
t_{SBP}	系统上电开机时间（慢开机）		50		ms	V_{DD} =5V
	系统上电开机时间（快开机）		750		us	V_{DD} =5V
t_{RST}	外部复位脉冲宽度	120			us	@ V_{DD} =5V
CPos	比较器偏压*	-	±10	±20	mV	
CPcm	比较器共模输入电压*	0		V_{DD} -1.5	V	
CPspt	比较器响应时间*		100	500	ns	上升沿和下降沿一样
CPmc	比较器模式改变稳定时间		2.5	7.5	us	
CPcs	比较器电流消耗		20		uA	V_{DD} = 3.3V

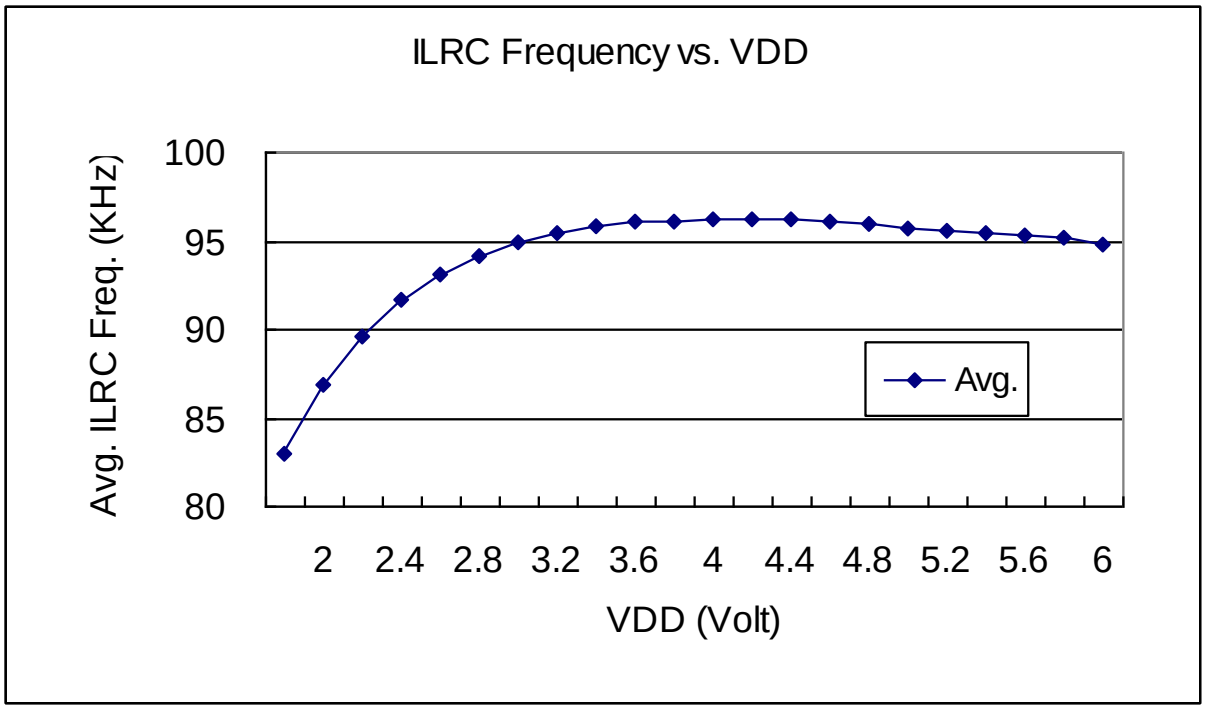
*这些参数是设计参考值，并不是每个芯片测试。

©Copyright 2023, PADAUK Technology Co. Ltd

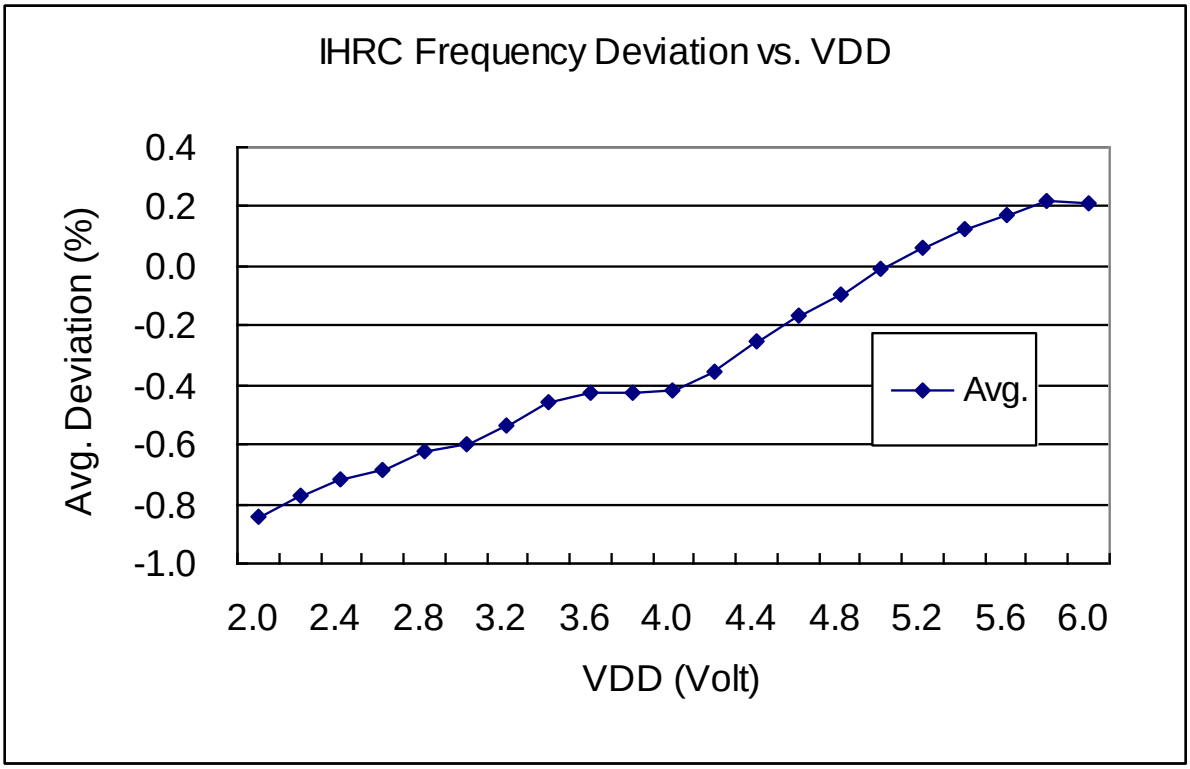
4.2. 绝对最大值

- 电源电压 ..... 2.2V ~ 5.5V
  *如果 V_{DD} 超过最大值，会损坏 IC。
- 输入电压 ..... -0.3V ~ V_{DD} + 0.3V
- 工作温度 ..... -20°C ~ 85°C
- 节点温度 ..... 150°C
- 存储温度 ..... -50°C ~ 125°C

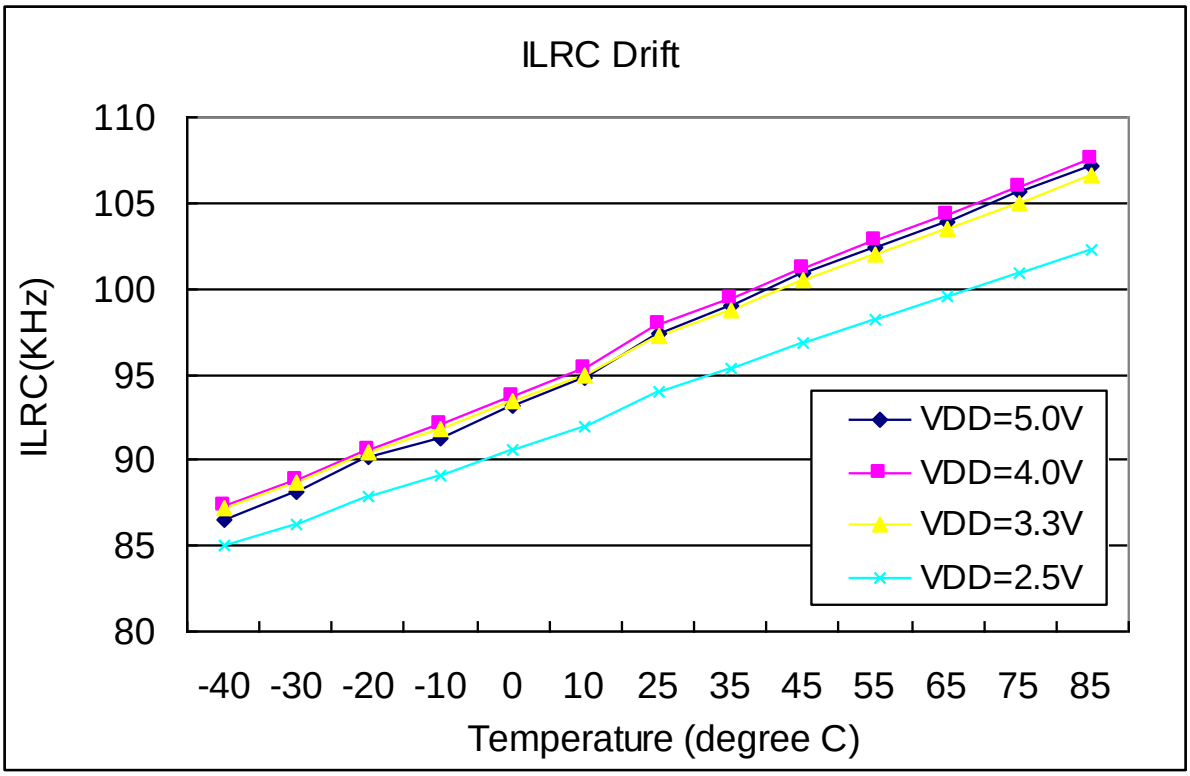
4.3. ILRC 频率与 VDD 的关系曲线图



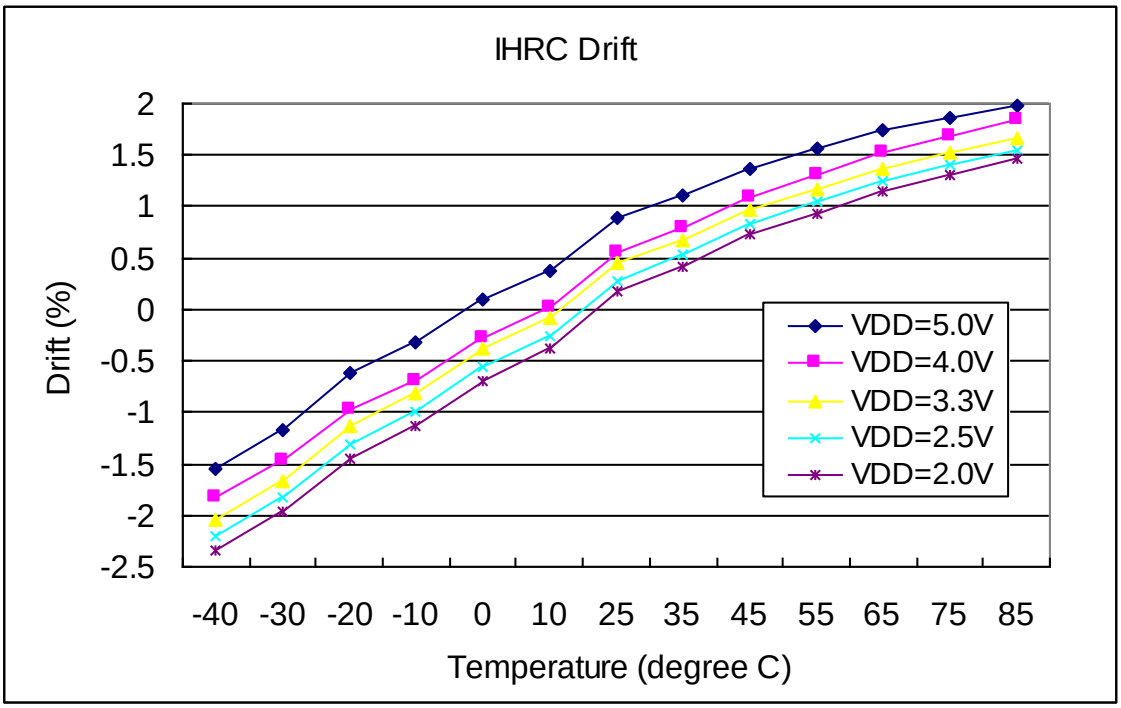
4.4. IHRC 频率与 VDD 的关系曲线图（校准到 16MHz）



4.5. ILRC 频率与温度的关系曲线图



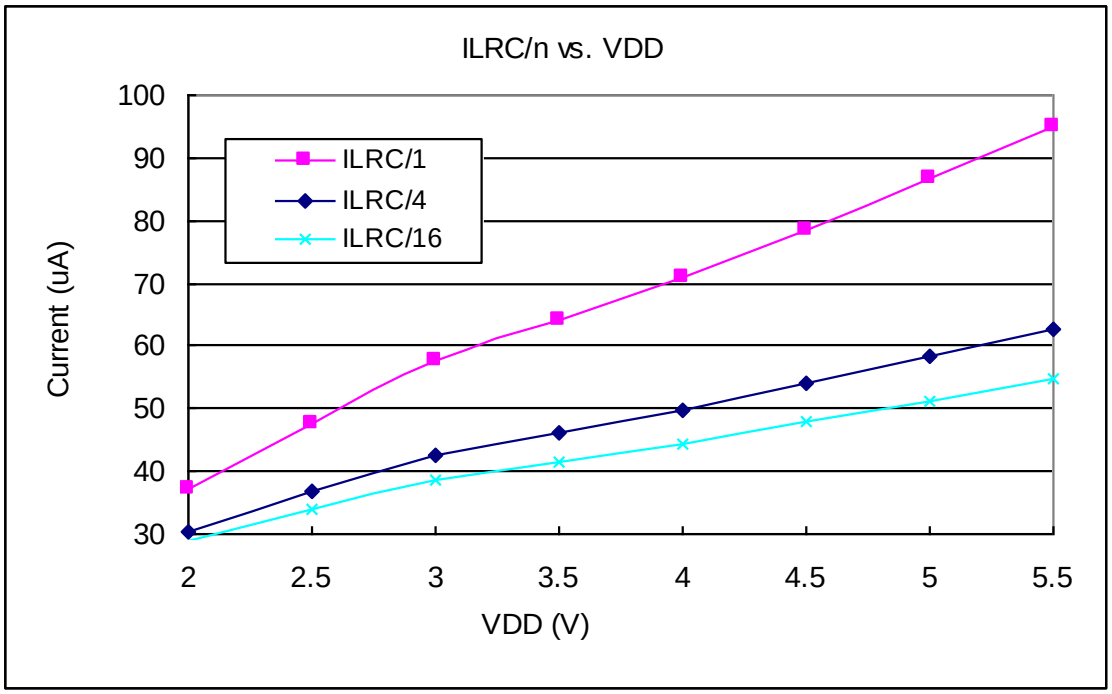
4.6. IHRC 频率与温度的关系曲线图（校准到 16MHz）



4.7. 工作电流与 VDD、系统时钟 CLK = ILRC/n 关系曲线图

条件：**开启**：ILRC，Bandgap，LVR；**关闭**：IHRC，EOSC，T16，TM2，TM3，ADC modules；

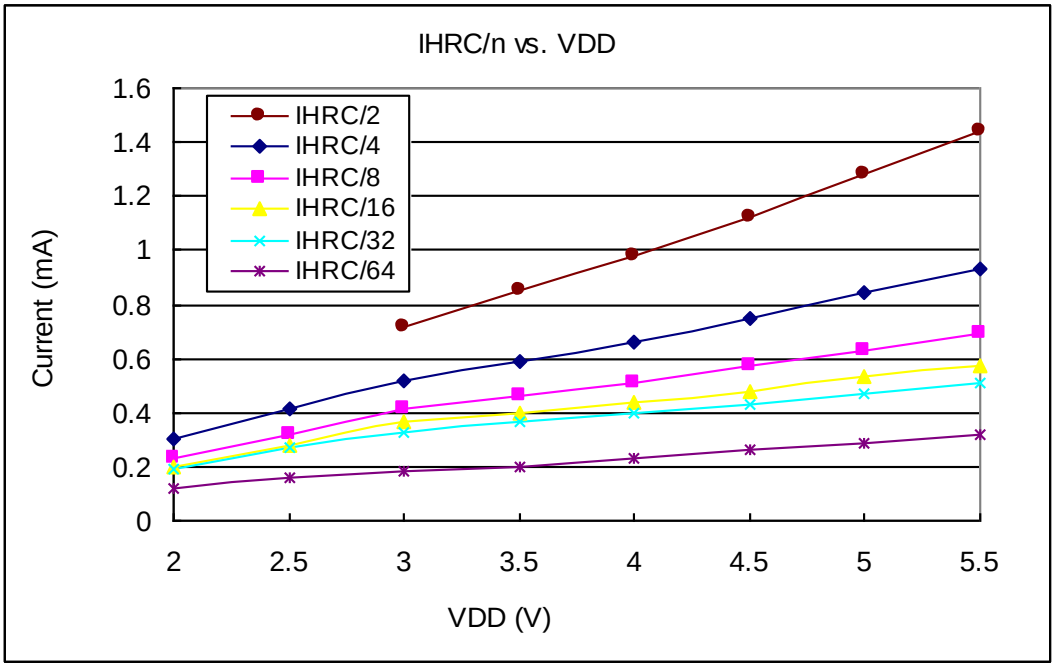
IO：PA0 以 0.5Hz 频率高低电压交换输出，无负载；**其他**：设为输入且不浮空



4.8. 工作电流与 VDD、系统时钟 CLK = IHRC/n 关系曲线图

条件：**开启**：IHRC，Bandgap，LVR；**关闭**：ILRC，EOSC，T16，TM2，TM3，ADC modules；

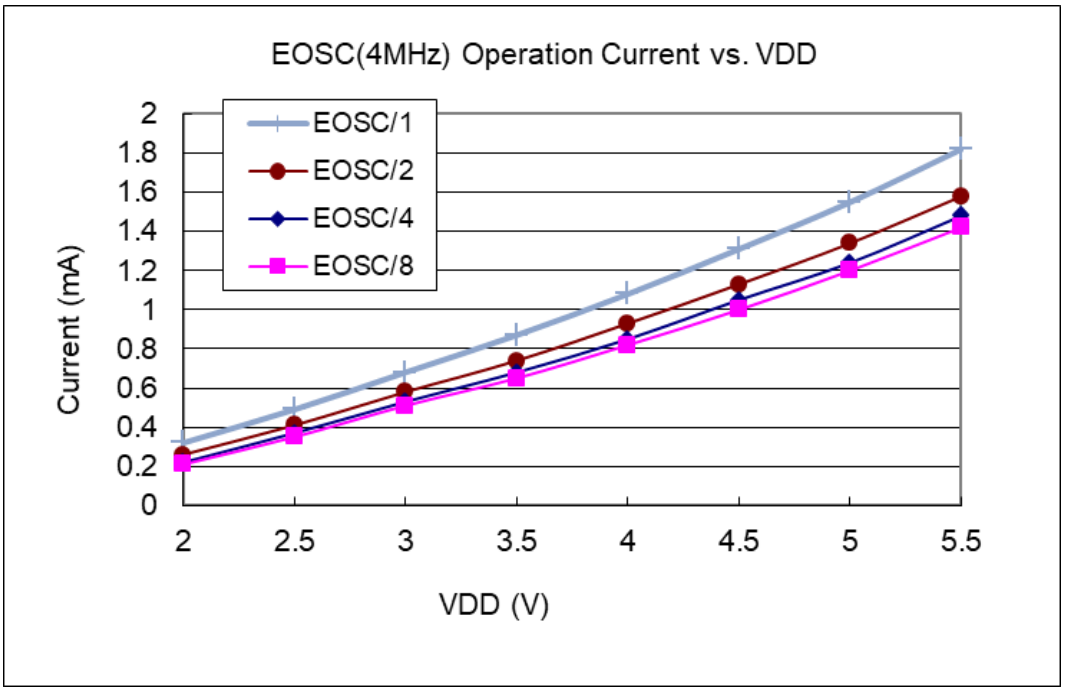
IO：PA0 以 0.5Hz 频率高低电压交换输出，无负载；**其他**：设为输入且不浮空



4.9. 工作电流与 VDD、系统时钟 CLK = 4MHz EOSC / n 关系曲线图

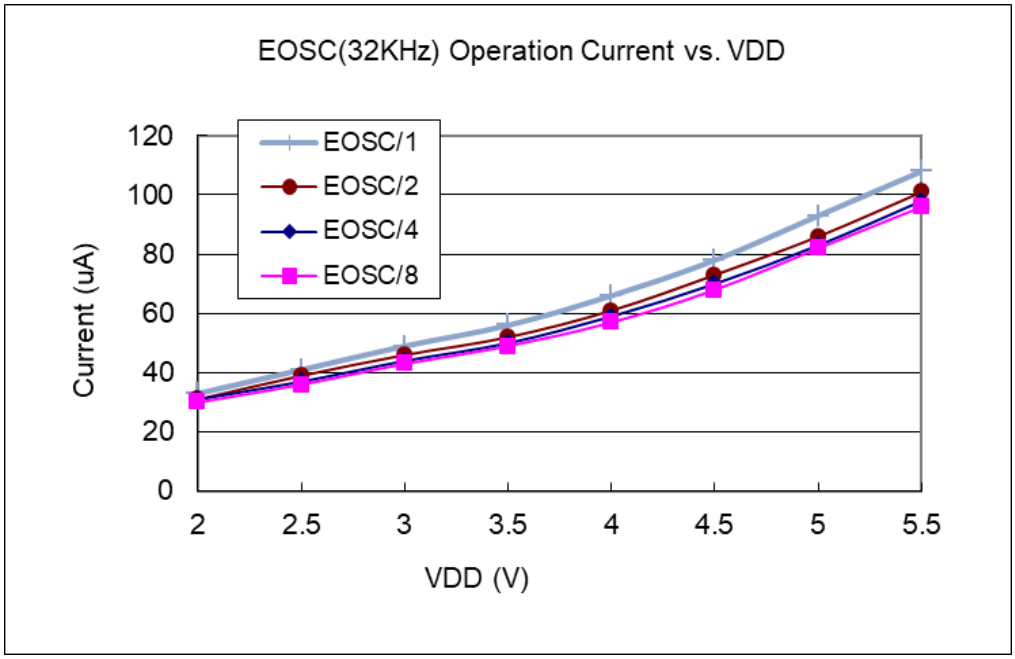
条件：**开启**：EOSC[6,5] = [1,1] ，Bandgap，LVR；**关闭**：IHRC，ILRC，T16，TM2，TM3，ADC modules；

IO：PA0 以 0.5Hz 频率高低电压交换输出，无负载；**其他**：设为输入且不浮空



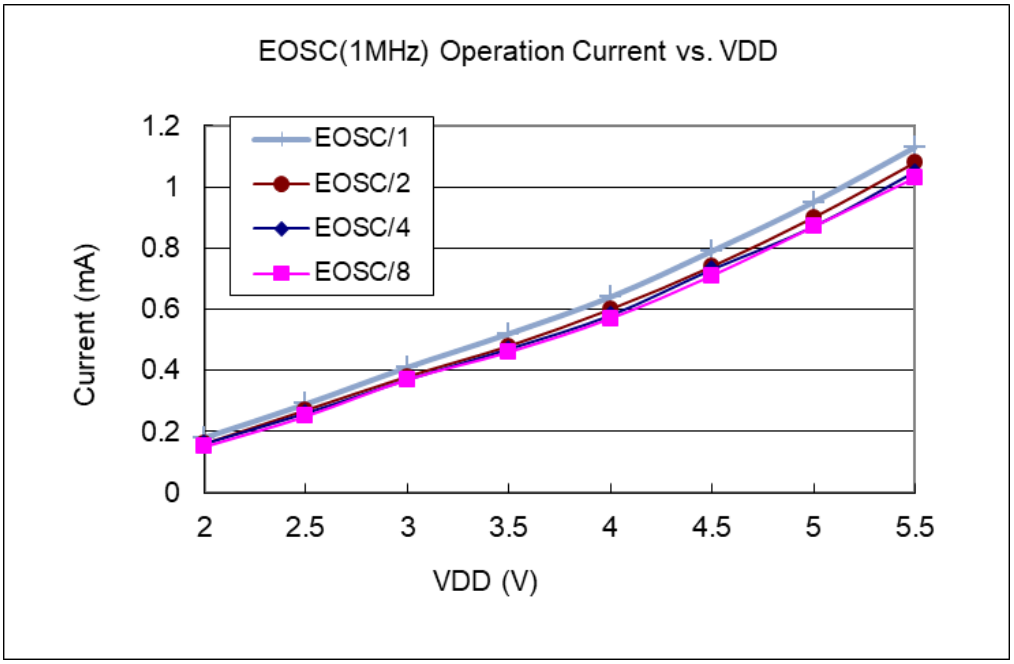
4.10. 工作电流与 VDD、系统时钟 CLK = 32KHz EOSC / n 关系曲线图

条件：**开启**：EOSC[6,5] = [0,1]，Bandgap，LVR；**关闭**：IHRC，ILRC，T16，TM2，TM3，ADC modules；
IO：PA0 以 0.5Hz 频率高低电压交换输出，无负载；**其他**：设为输入且不浮空



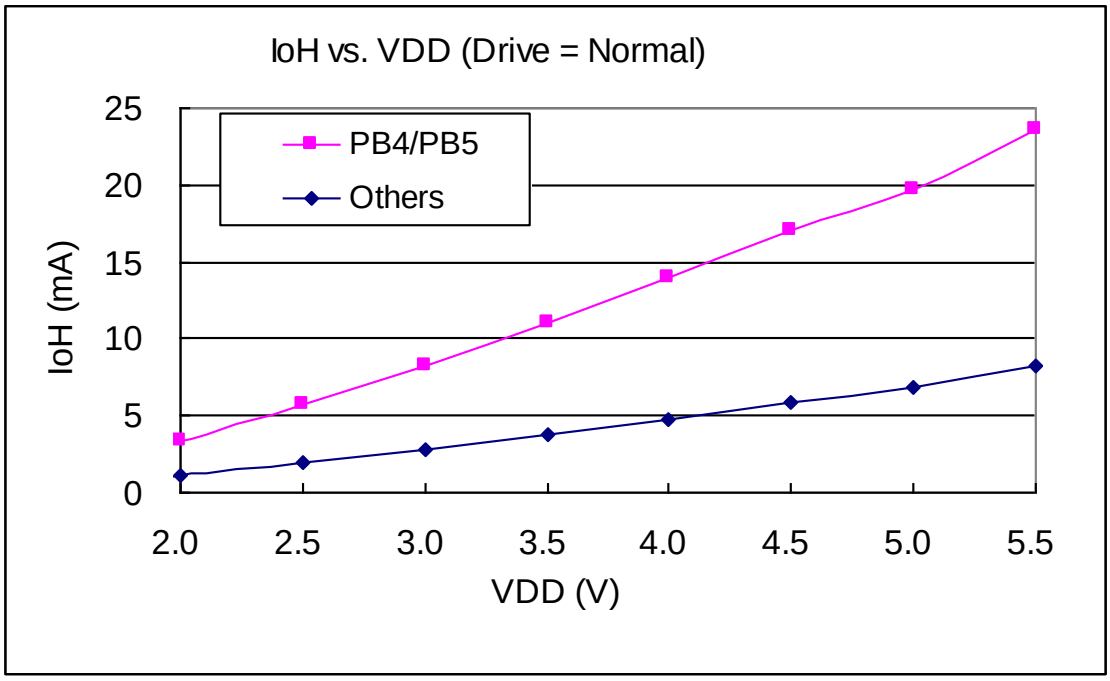
4.11.工作电流与 VDD、系统时钟 CLK = 1MHz EOSC / n 关系曲线图

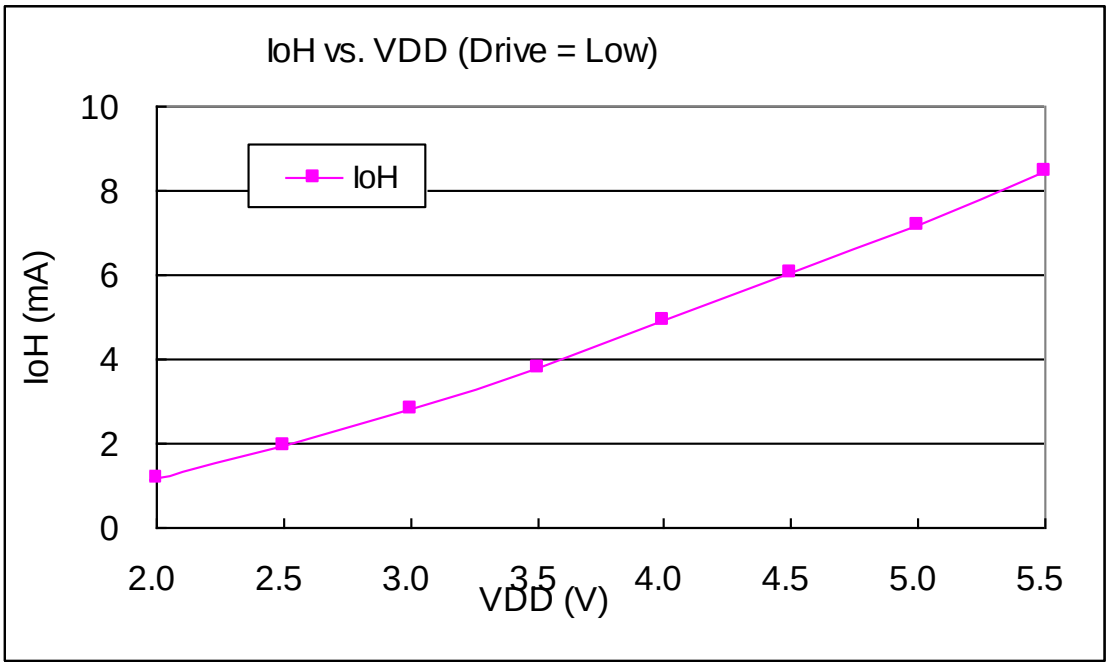
条件：**开启**：EOSC[6,5] = [1,0]，Bandgap，LVR；**关闭**：IHRC，ILRC，T16，TM2，TM3，ADC modules；
IO：PA0 以 0.5Hz 频率高低电压交换输出，无负载；**其他**：设为输入且不浮空



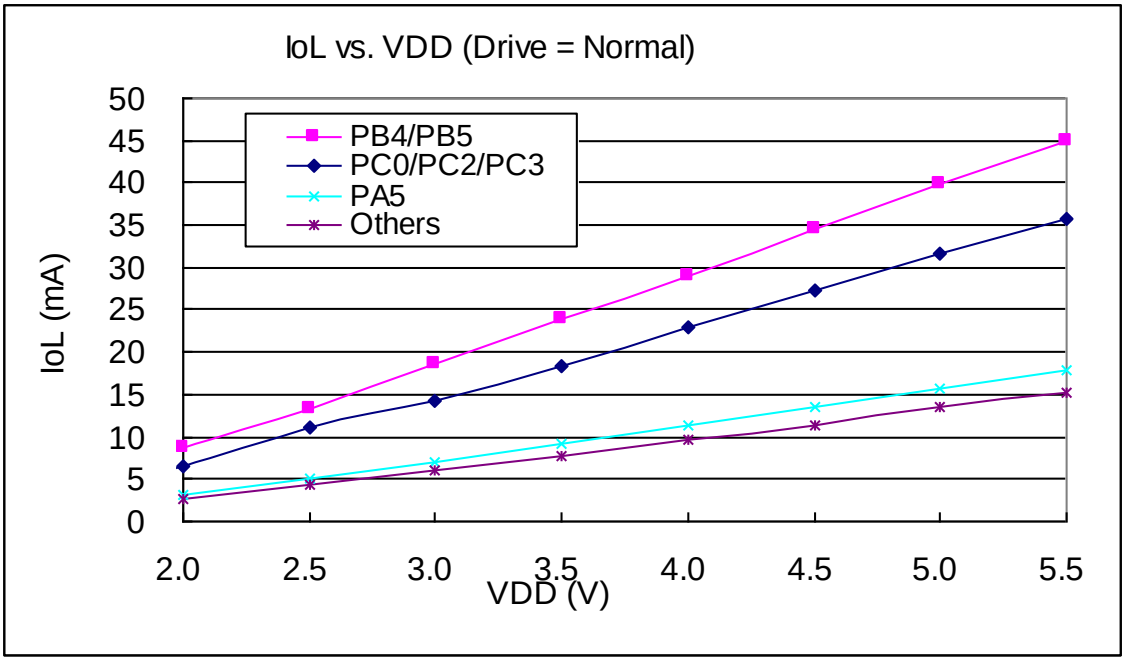
4.12. IO 引脚输出驱动电流(I_{OH}) 和灌电流 (I_{OL})曲线图

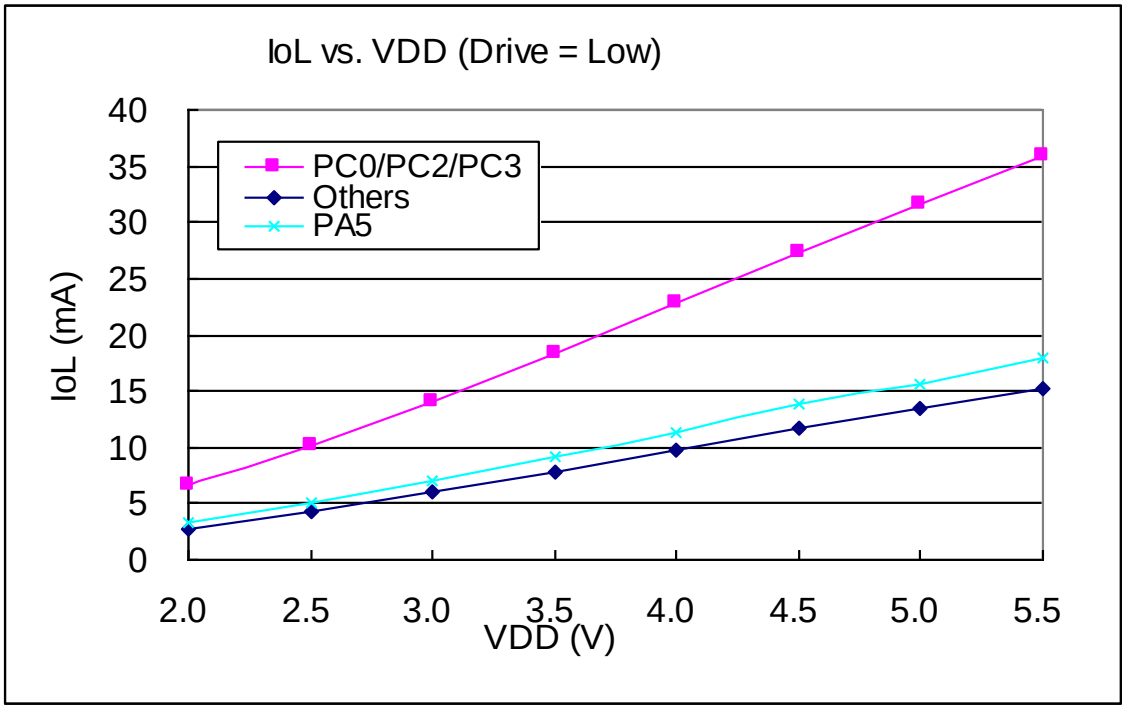
（VOH=0.9*VDD, VOL=0.1*VDD）



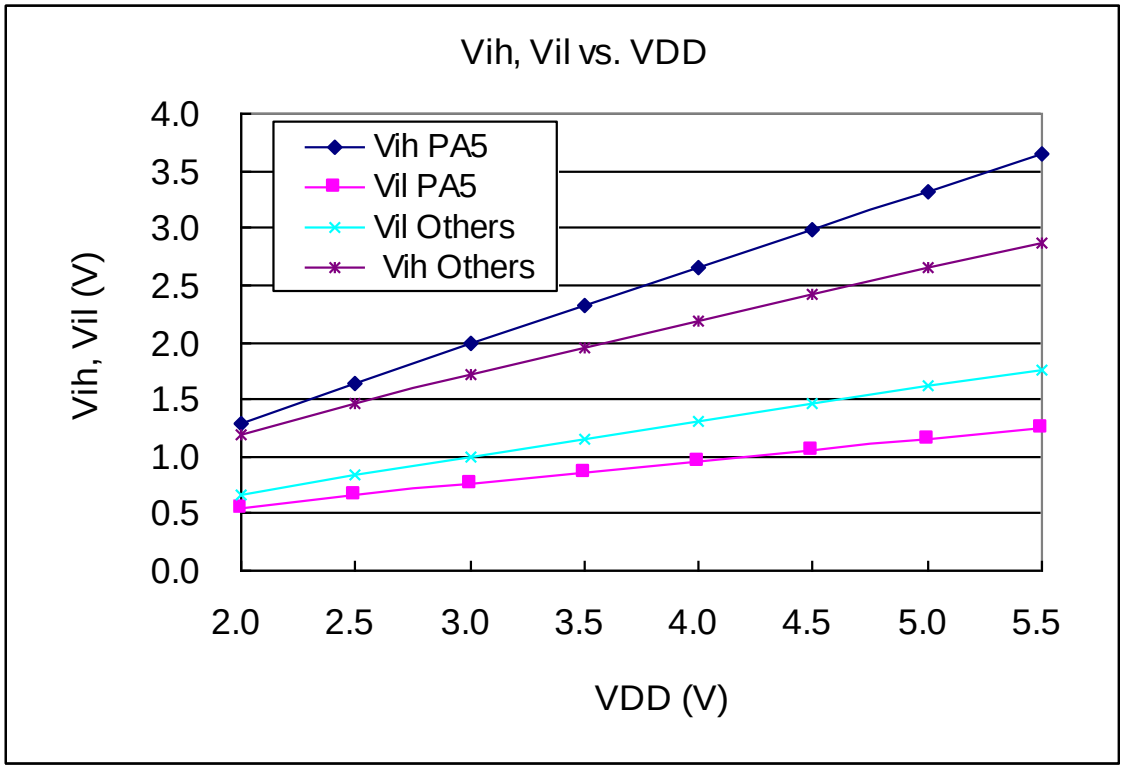


©Copyright 2023, PADAUK Technology Co. Ltd

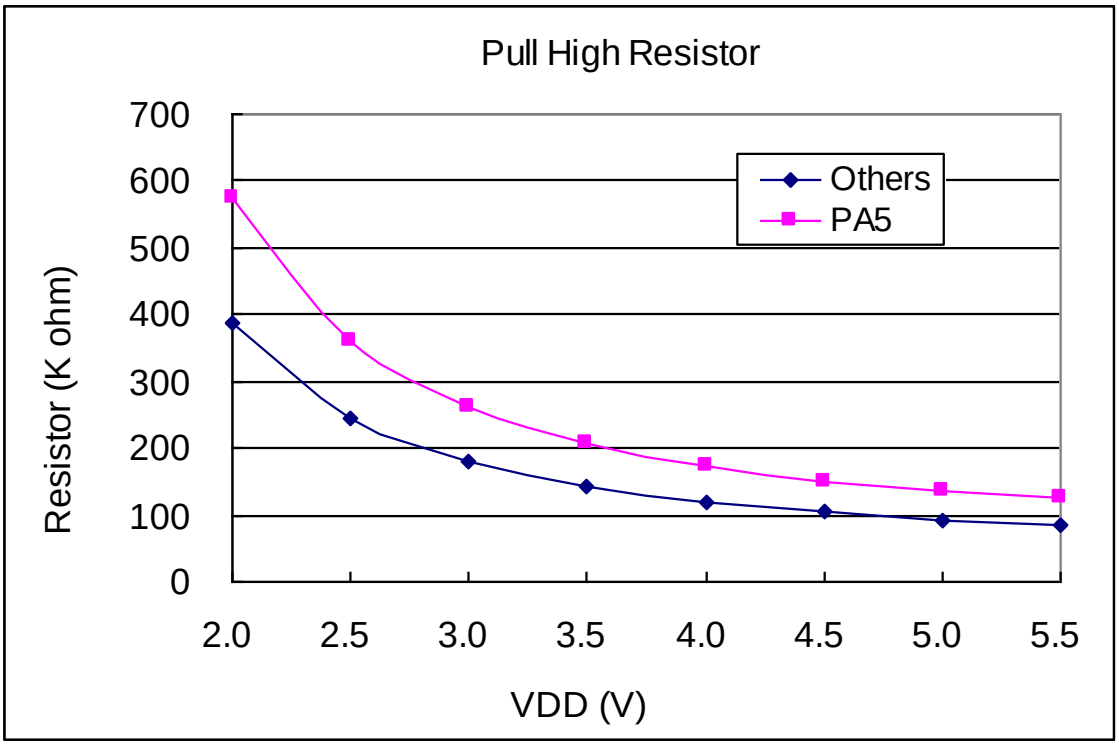

IoL vs. VDD (Drive = Normal)
IoL (mA)
VDD (V)
PB4/PB5
PC0/PC2/PC3
PA5
Others
0
5
10
15
20
25
30
35
40
45
50
2.0
2.5
3.0
3.5
4.0
4.5
5.0
5.5

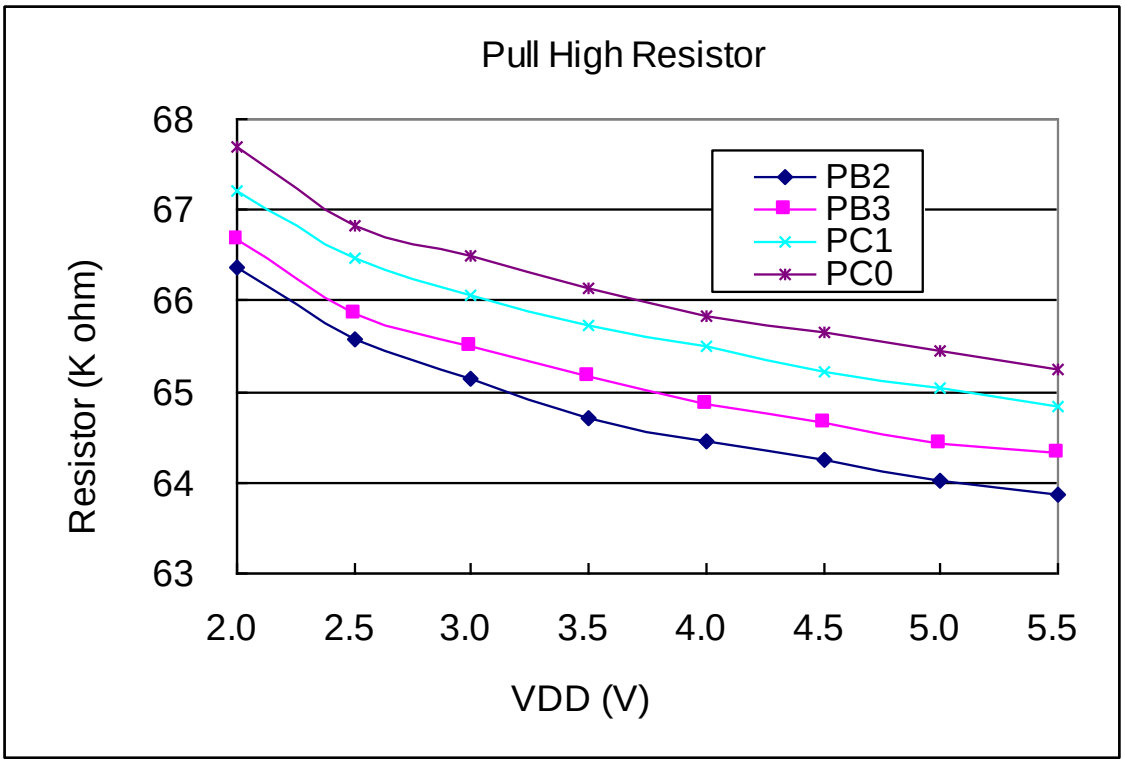

IoL vs. VDD (Drive = Low)
IoL (mA)
VDD (V)
PC0/PC2/PC3
Others
PA5
0
5
10
15
20
25
30
35
40
2.0
2.5
3.0
3.5
4.0
4.5
5.0
5.5

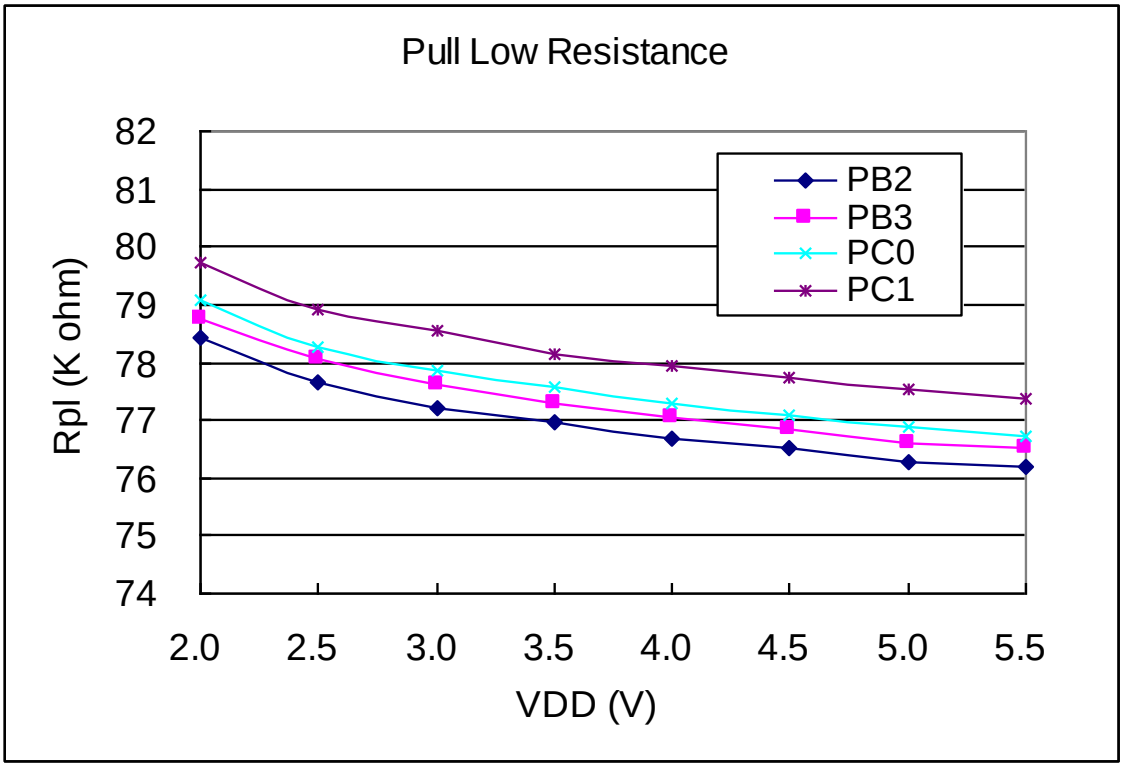
4.13. IO 引脚输入高/低阈值电压(V_{IH}/V_{IL})曲线图



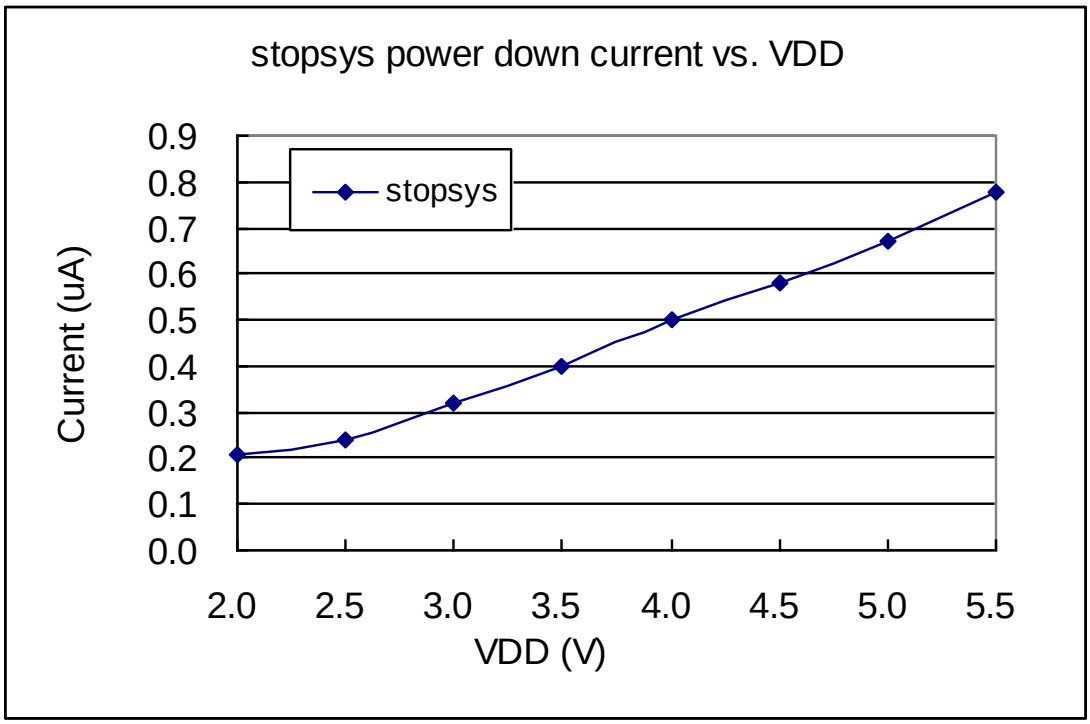
4.14. IO 引脚上拉/下拉阻抗曲线图

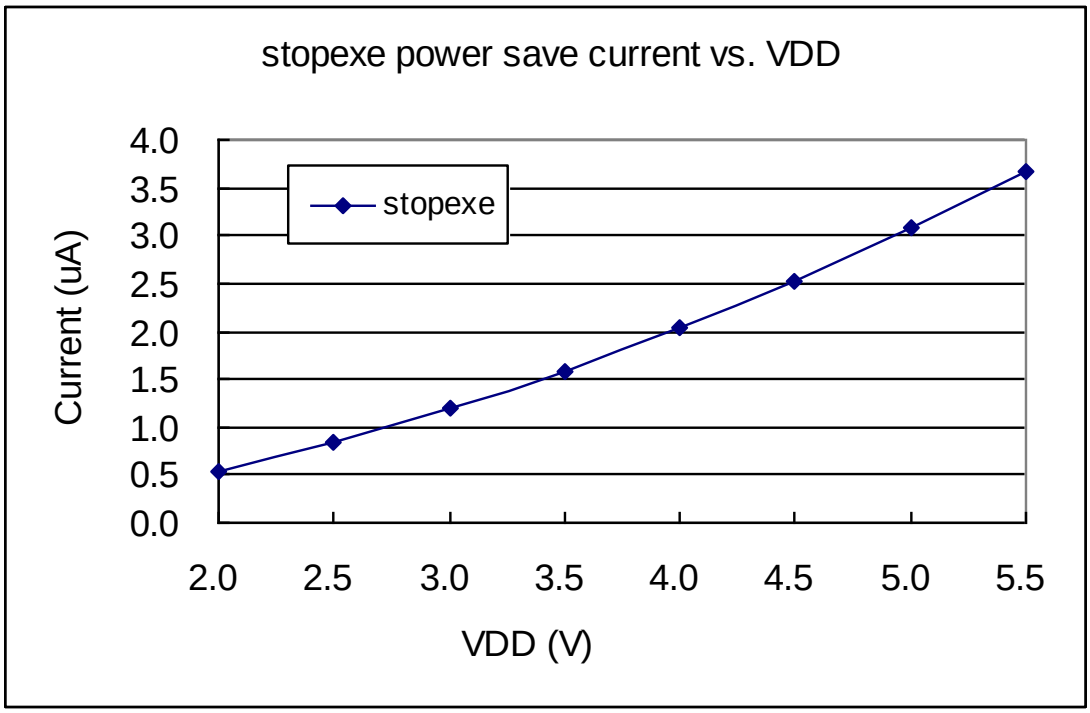



Pull High Resistor
Resistor (K ohm)
68
67
66
65
64
63
2.0
2.5
3.0
3.5
4.0
4.5
5.0
5.5
VDD (V)
PB2
PB3
PC1
PC0


Pull Low Resistance
Rpl (K ohm)
82
81
80
79
78
77
76
75
74
2.0
2.5
3.0
3.5
4.0
4.5
5.0
5.5
VDD (V)
PB2
PB3
PC0
PC1

4.15. 掉电模式消耗电流(I_{PD})与省电模式消耗电流(I_{PS})曲线图





5. 功能概述

5.1. 程序存储器 - MTP

MTP（多次可编程）程序存储器用来存放要执行的程序指令。MTP 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 的程序从初始地址 0x000 开始（通常中 GOTO FPPA0），中断入口是 0X010；MTP 程序存储器最后 32 个地址空间是被保留给系统使用，如：校验，序列号等。PFS123 的 MTP 程序存储器容量为 3KW，如表 1 所示。MTP 存储器从地址 0xBC0 到 0xBFF 供系统使用，地址从 0x001 到 0x00F 和从 0x011 到 0xBBF 是用户的程序空间。

地址	功能
0x000	GOTO 指令
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
•	•
0xBBF	用户程序区
0XBC0	系统使用
•	•
0xBFF	系统使用

表 1：程序存储器结构

5.2. 开机流程

开机时，POR（上电复位）是用于复位 PFS123。开机时间可选快开机或者普通模式。不管哪种开机模式，用户必须确保上电后电源电压稳定，开机时间 t_{SBP}，如图 1 所示。

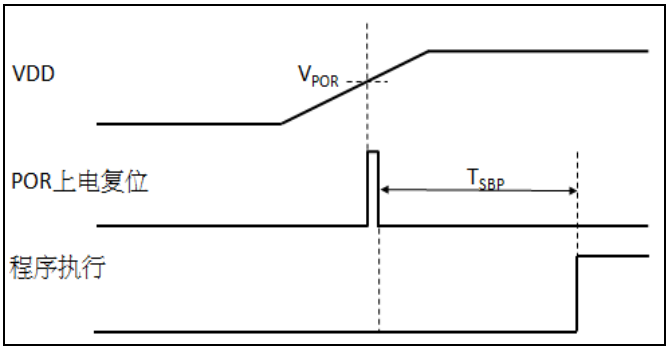
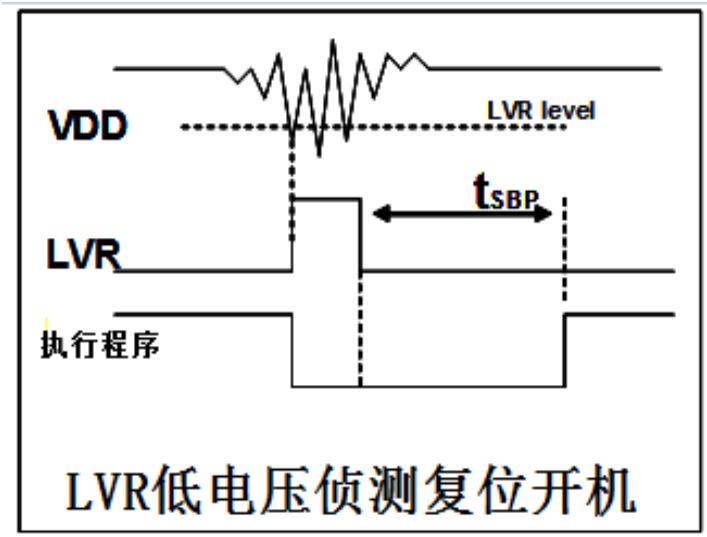
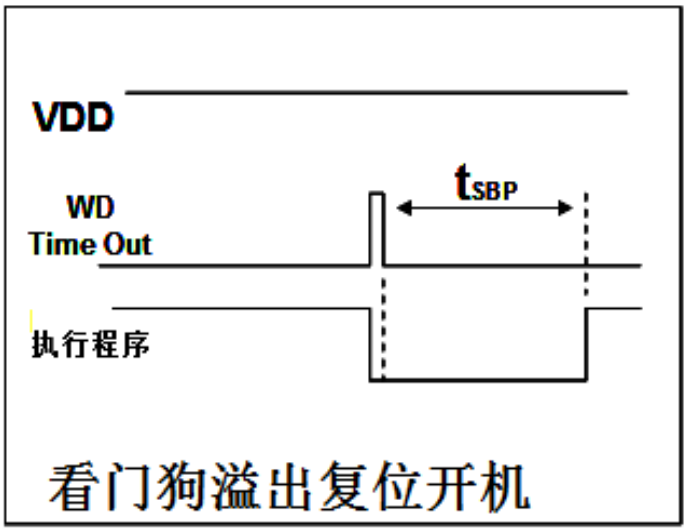
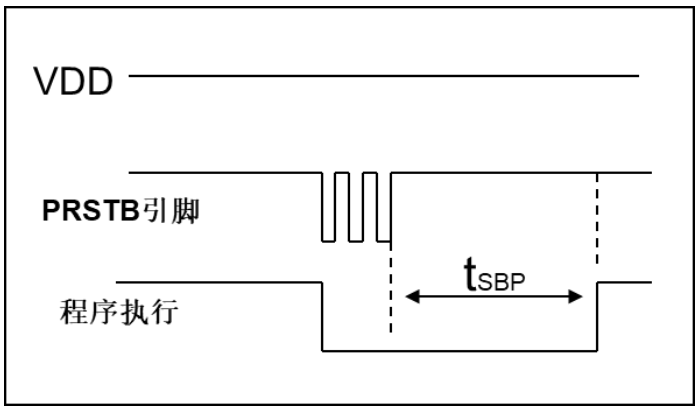


图 1：上电复位时序

5.2.1.复位时序图


VDD
LVR level
tSBP
LVR
执行程序
LVR低电压侦测复位开机


VDD
tSBP
WD
Time Out
执行程序
看门狗溢出复位开机


VDD
PRSTB引脚
tSBP
程序执行

5.3. 数据存储器 - SRAM

数据存取可以是字节或位的操作。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。

堆栈存储器是定义在数据储存器里。堆栈存储器的堆栈指针是定义在堆栈指针寄存器；堆栈存储器深度是由使用者定义的。用户可以依其程序需求来订定所需要堆栈存储器的大小，以保持最大的弹性。

数据存储器的间接存取方式，是以数据存储器当作数据指针来存取数据字节。所有的数据存储器，都可以拿来当作数据指针，这可以让单片机的资源利用率最大化。因为数据宽度是 8 位，PFS123 的数据存储器 256 字节全部都可以用间接方式来存取。

5.4. 振荡器和时钟

PFS123 提供 3 个振荡器电路：外部晶体振荡器 (EOSC)，内部高频振荡器 (IHRC)与内部低频振荡器 (ILRC) ，这 3 个振荡器可以分别用寄存器 eoscr.7，clkmd.4 与 clkmd.2 启用或停用。使用者可以选择这 3 个振荡器之一作为系统时钟源，并透过 *clkmd* 寄存器来改变系统时钟频率，以满足不同的系统应用。

振荡器硬件	启用或停用选择
EOSC	eoscr.7
IHRC	clkmd.4
ILRC	clkmd.2

表 2：3 个振荡器电路

5.4.1. 内部高频振荡器和低频振荡器

开机后，内部高频(IHRC)和低频(ILRC)振荡器都会开启。内部高频振荡器的频率(IHRC)透过 *ihrcr* 寄存器来消除工厂生产引起的频率漂移；IHRC 振荡器通常被校准到 16MHz，通常校准后的频率偏差都在 1%以内；且校准后 IHRC 的频率仍然会因电源电压和工作温度而略有漂移。请参阅 IHRC 频率和 V_{DD}、温度的测量图表。

ILRC 的频率会因工厂生产、电源电压和温度而变化，请参阅 DC 规格表。需要精确定时的应用时请不要使用 ILRC 的时钟当作参考时间。

5.4.2. 芯片校准

IHRC 的输出频率可能因工厂制造变化而有所差异，PFS123 提供 IHRC 输出频率校准，来消除工厂生产时引起的变化。这个功能是在编译用户的程序时序做选择，校准命令以及选项将自动插入到用户的程序，校准命令如下所示：

.ADJUST_IC SYSCLK=IHRC/(**p1**), IHRC=(**p2**)MHz, V_{DD}=(**p3**)V;

p1=2, 4, 8, 16, 32; 以提供不同的系统时钟。

p2=14 ~ 18; 校准芯片到不同的频率，通常选择 16MHz。

p3=2.5 ~ 5.5; 根据不同的电源电压校准芯片。

5.4.3. IHRC 频率校准与系统时钟

用户在程序编译期间，IHRC 频率校准以及系统时钟的选项，如表 3 所示：

SYSCLK	CLKMD	IHRCR	描述
○ Set IHRC / 2	= 34h (IHRC / 2)	有校准	IHRC 校准到 16MHz, CLK=8MHz (IHRC/2)
○ Set IHRC / 4	= 14h (IHRC / 4)	有校准	IHRC 校准到 16MHz, CLK=4MHz (IHRC/4)
○ Set IHRC / 8	= 3Ch (IHRC / 8)	有校准	IHRC 校准到 16MHz, CLK=2MHz (IHRC/8)
○ Set IHRC / 16	= 1Ch (IHRC / 16)	有校准	IHRC 校准到 16MHz, CLK=1MHz (IHRC/16)
○ Set IHRC / 32	= 7Ch (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	没改变	没改变	IHRC 不校准, CLK 不改变

表 3：IHRC 频率校准选项

通常情况下，ADJUST_IC 将是开机后的第一个命令，以设定系统的工作频率。程序代码在写入 MTP 的时候，IHRC 频率校准的程序会执行一次，以后，它就不会再被执行了。如果 IHRC 校准选择不同的选项，开机后的系统状态也是不同的。下面显示在不同的选项下，PFS123 不同的状态：

(1) .ADJUST_IC SYSCLK=IHRC/2, IHRC=16MHz, V_{DD}=5V

开机后，CLKMD = 0x34：

- IHRC 的校准频率为 16MHz@V_{DD}=5V，启用 IHRC 的硬件模块
- 系统时钟 = IHRC/2 = 8MHz
- 看门狗被禁止，启用 ILRC，PA5 是在输入模式

(2) .ADJUST_IC SYSCLK=IHRC/4, IHRC=16MHz, V_{DD}=3.3V

开机后，CLKMD = 0x14：

- IHRC 的校准频率为 16MHz@V_{DD}=3.3V，启用 IHRC 的硬件模块
- 系统时钟 = IHRC/4 = 4MHz
- 看门狗被禁止，启用 ILRC，PA5 是在输入模式

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

开机后，CLKMD = 0x3C：

- IHRC 的校准频率为 16MHz@V_{DD}=2.5V，启用 IHRC 的硬件模块
- 系统时钟 = IHRC/8 = 2MHz
- 看门狗被禁止，启用 ILRC，PA5 是在输入模式

(4) .ADJUST_IC SYSCLK=IHRC/16, IHRC=16MHz, V_{DD}=2.5V

开机后，CLKMD = 0x1C：

- IHRC 的校准频率为 16MHz@V_{DD}=2.5V，启用 IHRC 的硬件模块
- 系统时钟 = IHRC/16 = 1MHz
- 看门狗被禁止，启用 ILRC，PA5 是在输入模式

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

开机后，CLKMD = 0x7C：

- IHRC 的校准频率为 16MHz@V_{DD}=5V，启用 IHRC 的硬件模块
- 系统时钟 = IHRC/32 = 500KHz
- 看门狗被禁止，启用 ILRC，PA5 是在输入模式

©Copyright 2023, PADAUK Technology Co. Ltd

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

开机后，CLKMD = 0xE4：

- IHRC 的校准频率为 16MHz@V_{DD}=5V, 停用 IHRC 的硬件模块
- 系统时钟 = ILRC
- 看门狗被禁止，启用 ILRC，PA5 是在输入模式

(7) .ADJUST_IC　　DISABLE

开机后，CLKMD 寄存器没有改变（没任何动作）：

- IHRC 不校准并且 IHRC 模块停用
- 系统时钟 = ILRC or IHRC/64
- 看门狗被启用，启用 ILRC，PA5 是在输入模式

5.4.4. 外部晶体振荡器

如果要使用晶体振荡器，就需要再在 X1 和 X2 之间放置晶体或谐振器。图 2 显示了使用晶体振荡器的硬件连接；晶体振荡器的工作频率范围可以从 32KHz 至 4MHz，取决于放置的晶体，PFS123 不支持比 4MHz 更高的频率振荡器。

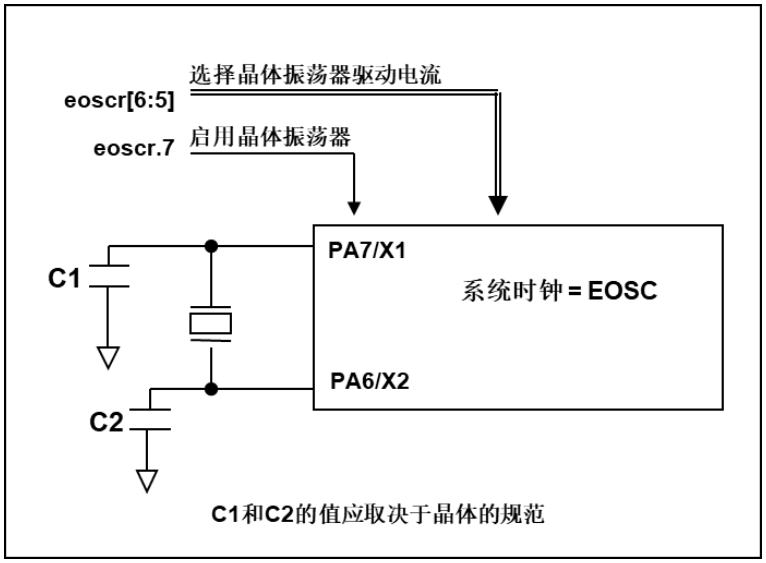


图 2：晶体振荡器的的硬件连接

除了晶振的选择外，外部电容器和 PFS123 寄存器 eoscr (0x0a)相关选项也应该适度调整以求得有良好的正弦波。*eoscr*.7 是用开启晶体振荡器硬件模块，*eoscr*.6 和 *eoscr*.5 用于设置振荡器不同的驱动电流，以满足晶体振荡器不同频率的的要求：

- *eoscr*.[6:5]=01：驱动电流低，适用于较低的频率，例如：32KHz 晶体振荡器
- *eoscr*.[6:5]=10：中度驱动电流，适用于中间的频率，例如：1MHz 的晶体振荡器
- *eoscr*.[6:5]=11：驱动电流高，适用于较高的频率，例如：4MHz 晶体振荡器

表 4 显示了不同的晶体振荡器 C1 和 C2 的推荐值，同时也显示其对应的条件下测量的起振时间。由于晶体或谐振器有其自身的特点，不同类型的晶体或谐振器的启动时间可能会略有不同，请参考其规格并选择恰当的 C1 和 C2 电容值。

©Copyright 2023, PADAUK Technology Co. Ltd

频率	C1	C2	起振时间	条件
4MHz	4.7pF	4.7pF	6ms	(*eoscr*[6:5]=11, *misc*.6=0)
1MHz	10pF	10pF	11ms	(*eoscr*[6:5]=10, *misc*.6=0)
32KHz	22pF	22pF	450ms	(*eoscr*[6:5]=01, *misc*.6=0)

表 4：晶体振荡器 C1 和 C2 推荐值

当使用晶体振荡器，使用者必须特别注意振荡器的稳定时间，稳定时间将取决于振荡器频率、晶型、外部电容和电源电压。在系统时钟切换到晶体振荡器之前，使用者必须确保晶体振荡器是稳定的，相关参考程序如下所示：

```
void    FPPA0 (void)
{
    . ADJUST_IC   SYSCLK=IHRC/16, IHRC=16MHz, VDD=5V
$   EOSCR     Enable, 4MHz;          // EOSCR = 0b110_00000;

$   T16M  EOSC, /1, BIT13;           // T16 receive 2^14=16384 个振荡时钟周期.
                                     // Intrq.T16 =>1，此時晶体振荡器已稳定
    WORD      count =     0;
    stt16  count;
    Intrq.T16   =     0;
    do
    {   nop; }while(!Intrq.T16);     // 计数从 0x0000 to 0x2000，然后 INTRQ.T16 触发
    clkmd =     0xB4;                // 切换系统时钟到 EOSC;

    Clkmd.4 = 0;                     // 关闭 IHRC
    ...
```

需要注意，在进入掉电模式前，为保证不会被误唤醒，要确保外部晶体振荡器已完全关闭。

5.4.5. 系统时钟和 LVR 基准位

系统时钟的时钟源有 EOSC，IHRC 和 ILRC，PFS123 的时钟系统的硬件框图如图 3 所示。

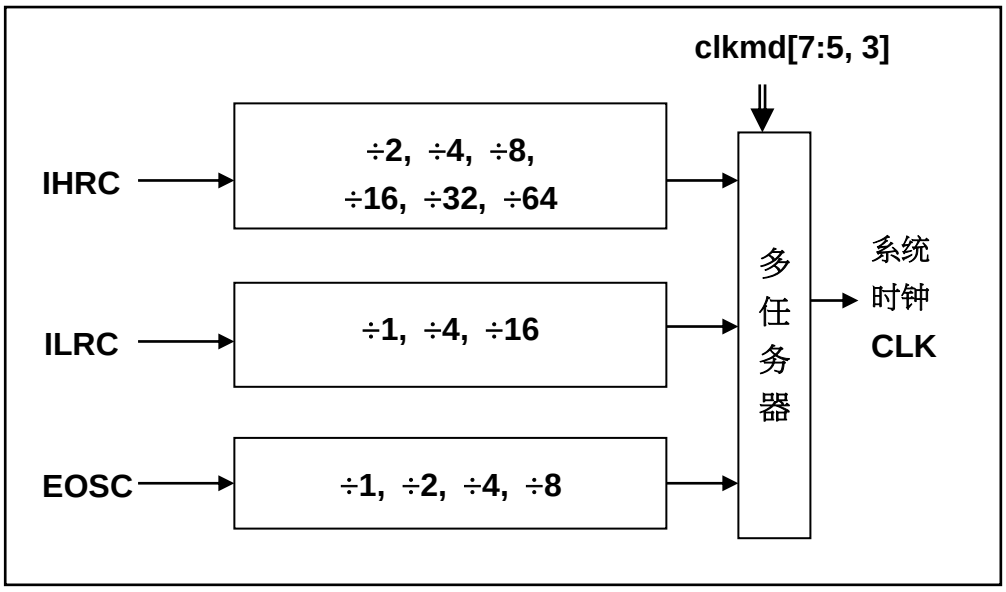


图 3：系统时钟源选择

使用者可以在不同的需求下选择不同的系统时钟，选定的系统时钟应与电源电压和 LVR 的水平结合，才能使系统稳定。LVR 的水平是在编译过程中选择，不同系统时钟对应的 LVR 设定，请参考章节 4.1 中系统时钟的最低工作电压。

©Copyright 2023, PADAUK Technology Co. Ltd

5.4.6. 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，PFS123 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/2

```
...                                 //  系统时钟是 ILRC
CLKMD.4         =   1;              //  先打开 IHRC，可以提高抗干扰能力
CLKMD           =   0x34;           //  切换到 IHRC/2，ILRC 不能在这里停用
//  CLKMD.2     =   0;              //  假如需要，ILRC 可以在这里停用
...
```

例 2: 系统时钟从 ILRC 切换到 EOSC

```
...                                 //  系统时钟是 ILRC
CLKMD           =   0xA6;           //  切换到 IHRC，ILRC 不能在这里停用
CLKMD.2         =   0;              //  ILRC 可以在这里停用
...
```

例 3: 系统时钟从 IHRC/2 切换到 ILRC

```
...                                 //  系统时钟是 IHRC/2
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 4: 系统时钟从 IHRC/2 切换到 EOSC

```
...                                 //  系统时钟是 IHRC/2
CLKMD           =   0XB0;           //  切换到 EOSC，IHRC 不能在这里停用
CLKMD.4         =   0;              //  IHRC 可以在这里停用
...
```

例 5: 系统时钟从 IHRC/2 切换到 IHRC/4

```
...                                 //  系统时钟是 IHRC/2，ILRC 在这里是启用的
CLKMD           =   0X14;           //  切换到 IHRC/4
...
```

例 6: 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 ILRC
CLKMD           =   0x30;           //  不能从 ILRC 切换到 IHRC/2 同时关闭 ILRC 振荡器
```

5.5. 比较器

PFS123 内置一个硬件比较器，图 4 所示比较器硬件原理框图，它可以比较两个引脚之间的信号或者与内部参考电压 $V_{internal\ R}$ 或者与内置 bandgap(1.2v)做比较。两个信号进行比较，一个是正输入，另一个是负输入。比较器的负输入可以是 PA3，PA4，内置 bandgap(1.2v)，PB6，PB7，或者内部参考电压 $V_{internal\ R}$，并由寄存器 gpcc 的[3:1]位来选择，比较器的正输入可以是 PA4 或者 $V_{internal\ R}$，并由 gpcc 寄存器的位 0 来选择。

比较器的输出结果可以选择直接输出到 PA0，或者通过 Timer2 计数器时钟模块(TM2_CLK)采样，另外，信号是否反极性也是可选的，比较器输出结果可以用产生中断信号或者通过 *gpcc* 寄存器的方式读取。

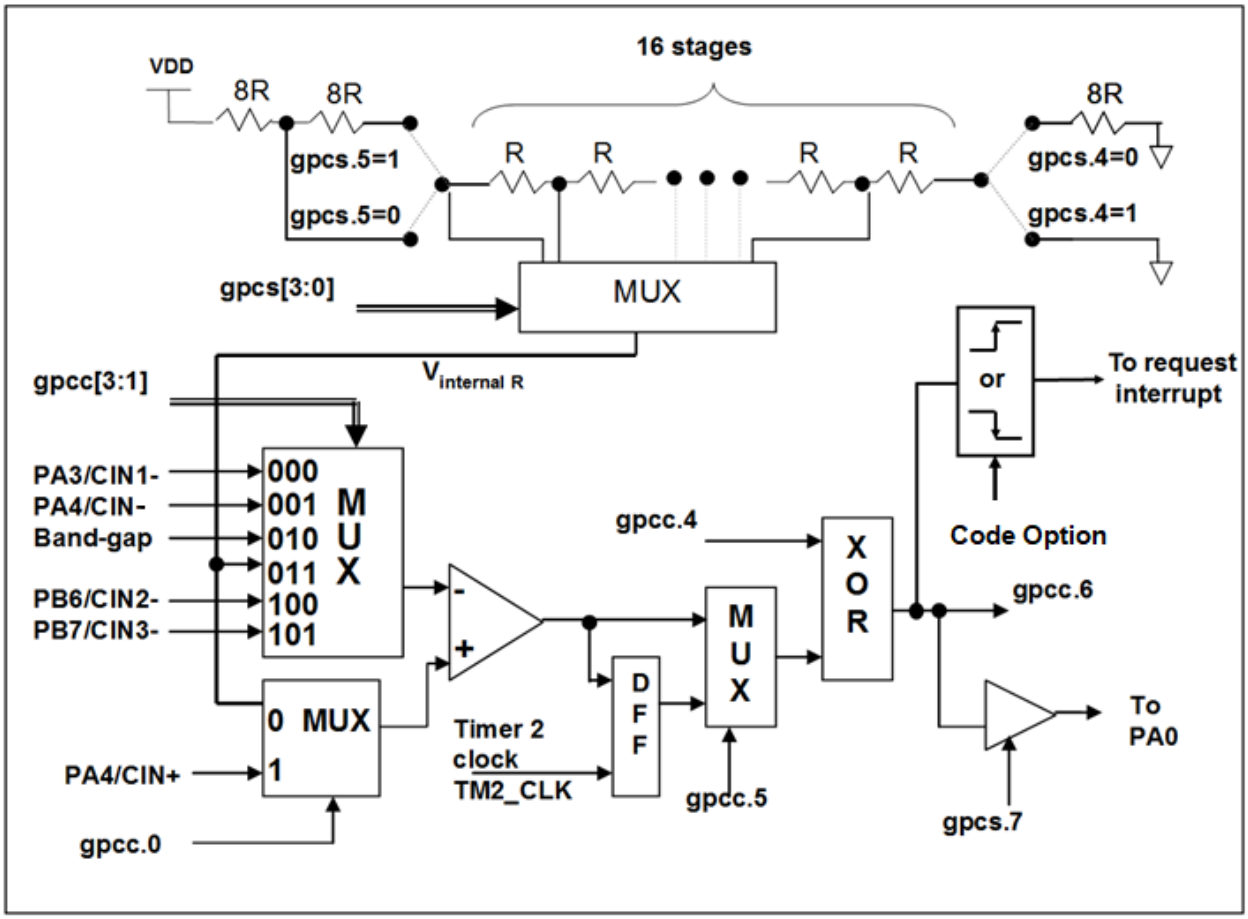


图 4：比较器硬件原理框图

5.5.1 内部参考电压 ($V_{internal\ R}$)

内部参考电压 $V_{internal\ R}$ 由一连串电阻所组成，可以产生不同层次的参考电压，*gpcs* 寄存器的位 4 和位 5 是用来选择 $V_{internal\ R}$ 的最高和最低值，位[3:0]用于选择所要的电压水平，这电压水平是由 $V_{internal\ R}$ 的最高和最低值均分 16 等份，由位[3:0]选择出来。图 5 ~ 图 8 显示四个条件下有不同的参考电压 $V_{internal\ R}$。内部参考电压 $V_{internal\ R}$ 可以通过 *gpcs* 寄存器来设置，范围从(1/32)*V_{DD} 到 (3/4)*V_{DD}。

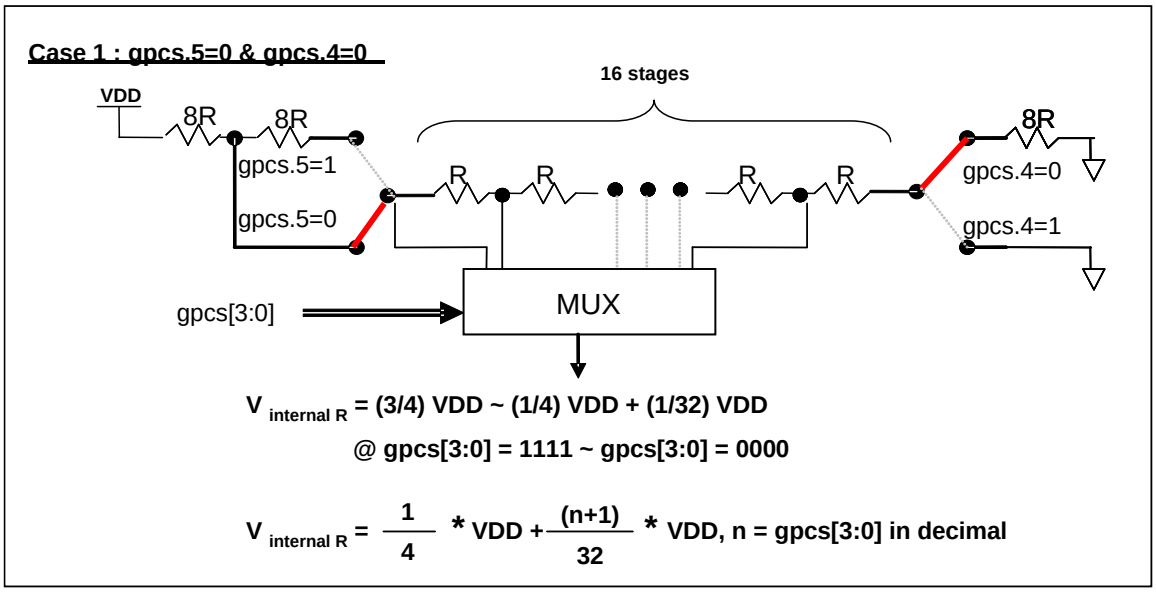


图 5：$V_{internal\ R}$ 硬件接法(gpcs.5=0 & gpcs.4=0)

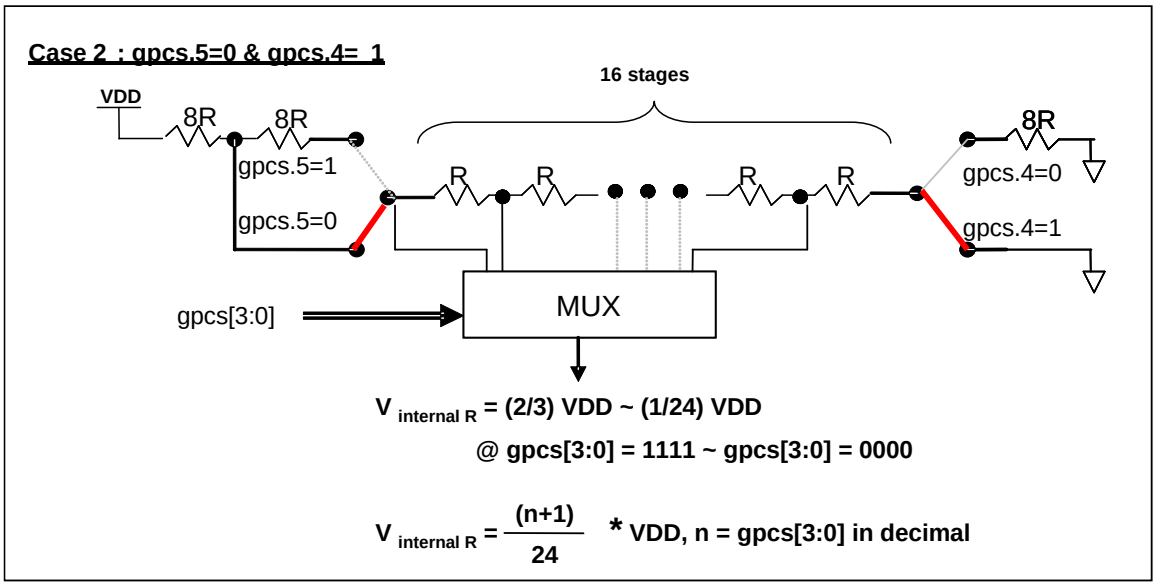


图 6：$V_{internal\ R}$ 硬件接法(gpcs.5=0 & gpcs.4=1)

©Copyright 2023, PADAUK Technology Co. Ltd

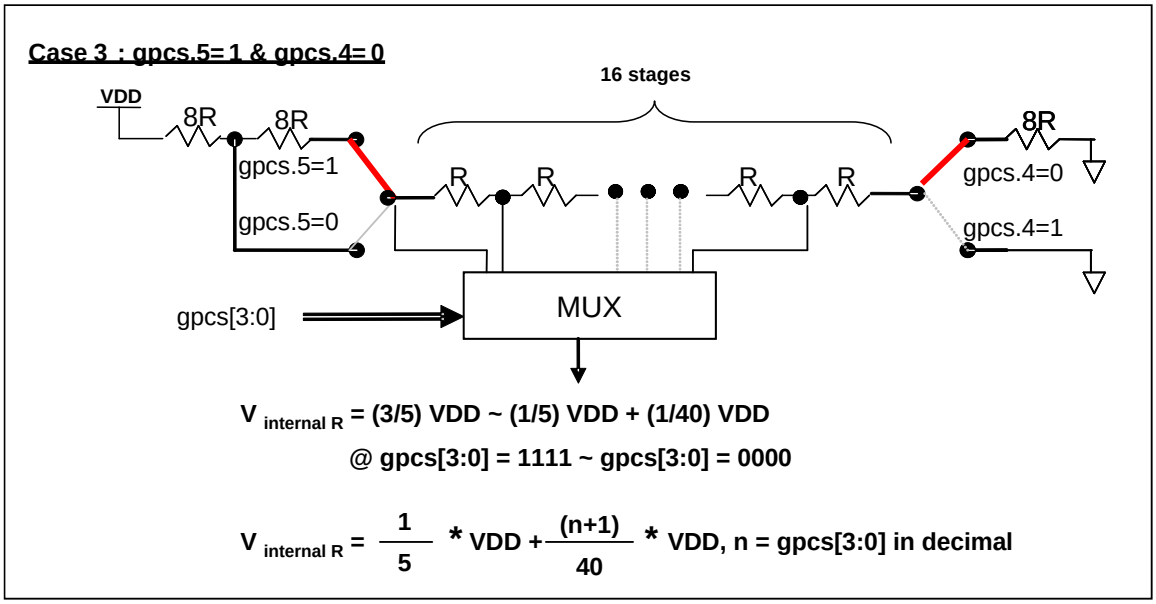


图 7：$V_{internal\ R}$ 硬件接法(gpcs.5=1 & gpcs.4=0)

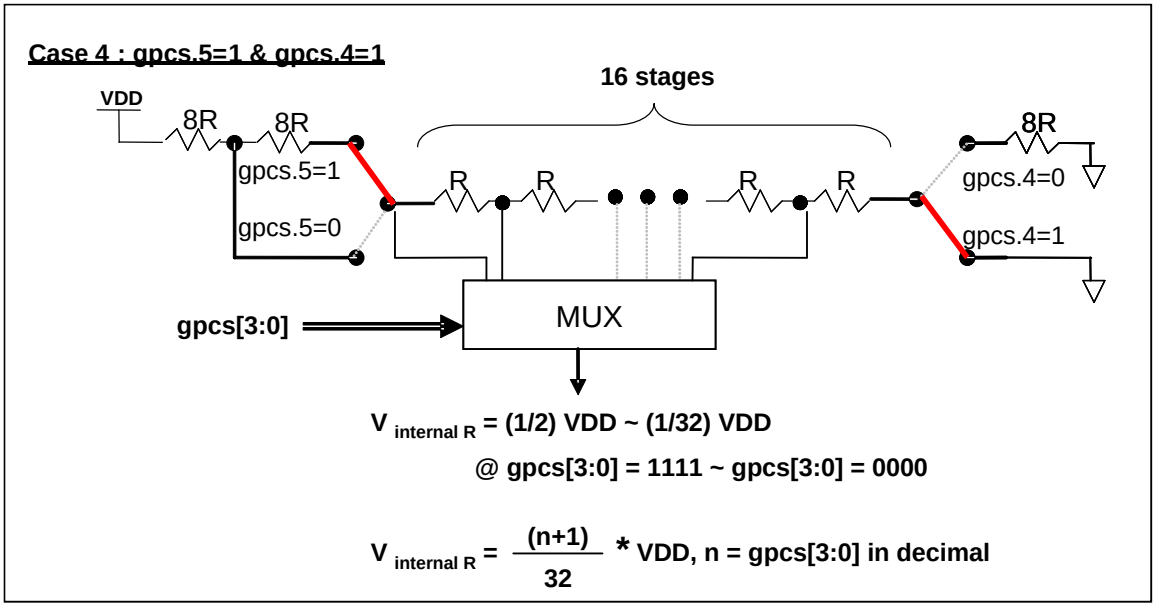


图 8：$V_{internal\ R}$ 硬件接法(gpcs.5=1 & gpcs.4=1)

5.5.2 使用比较器

例一：

选择 PA3 为负输入和 $V_{internal\ R}$ 的电压为(18/32)*V_{DD} 作为正输入。$V_{internal\ R}$ 选择上图 gpcs[5:4] = 2b'00 的配置方式，gpcs[3:0] = 4b'1001 (n=9) 以得到 $V_{internal\ R}$ = (1/4)*V_{DD} + [(9+1)/32]*V_{DD} = [(9+9)/32]*V_{DD} = (18/32)*V_{DD} 的参考电压。

```
gpcs    = 0b0_0_00_1001;          // Vinternal R = VDD*(18/32)
gpcc    = 0b1_0_0_0_000_0;        // 启用比较器，负输入：PA3，正输入：Vinternal R
padier  = 0bxxxx_0_xxx;           // 停用 PA3 数字输入防止漏电（x：由客户自定）
```

或者

```
$ GPCS      VDD*18/32;
$ GPCC      Enable, N_PA3, P_R;   //  N_xx 是负输入，P_R 代表正输入是内部参考电压
PADIER = 0bxxxx_0_xxx;
```

例二：

选择 $V_{internal\ R}$ 为负输入，$V_{internal\ R}$ 的电压为(22/40)*V_{DD}，选择 PA4 为正输入，比较器的结果将反极性并输出到 PA0。$V_{internal\ R}$ 选择上图的配置方式 "gpcs[5:4] = 2b'10" 和 gpcs[3:0] = 4b'1101 (n=13) 得到 $V_{internal\ R}$ = (1/5)*V_{DD} + [(13+1)/40]*V_{DD} = [(13+9)/40]*V_{DD} = (22/40)*V_{DD}。

```
gpcs    = 0b1_0_10_1101;          // 输出到 PA0，Vinternal R = VDD*(22/40)
gpcc    = 0b1_0_0_1_011_1;        // 反极性输出，负输入=Vinternal R，正输入=PA4
padier  = 0bxxx_0_xxxx;           // 停用 PA4 数字输入防止漏电（x：由客户自定）
```

或者

```
$ GPCS      Output, VDD*22/40;
$ GPCC      Enable, Inverse, N_R, P_PA4;   //  N_R 代表负输入是内部参考电压，P_xx 是正输入
PADIER=0bxxx_0_xxxx;
```

注意：当选择 PA0 做比较器结果输出时，GPCS 会影响 PA3 的仿真输出功能，但不影响实际 IC 的功能，请在仿真时需避开这个情况。

5.5.3 使用比较器和 bandgap 1.20V

内部 Bandgap 参考电压生成器可以提供 1.20V，它可以测量外部电源电压水平。该 Bandgap 参考电压可以选做负输入去和正输入 $V_{internal\ R}$ 比较。 $V_{internal\ R}$ 的电源是 V_{DD}，利用调整 $V_{internal\ R}$ 电压水平和 Bandgap 参考电压比较，就可以知道 V_{DD} 的电压。如果 N（gpcs[3:0]十进制）是让 $V_{internal\ R}$ 最接近 1.20V，那么 V_{DD} 的电压就可以透过下列公式计算：

对于 Case 1 而言：V_{DD} = [32 / (N+9)] * 1.20 volt ;

对于 Case 2 而言：V_{DD} = [24 / (N+1)] * 1.20 volt ;

对于 Case 3 而言：V_{DD} = [40 / (N+9)] * 1.20 volt ;

对于 Case 4 而言：V_{DD} = [32 / (N+1)] * 1.20 volt ;

例一：

```
$ GPCS VDD*12/40;          //  4.0V * 12/40 = 1.2V
$ GPCC Enable, BANDGAP, P_R;  //  BANDGAP 是负输入，P_R 代表正输入是内部参考电压
....
if  (GPC_Out)              //  或写成 GPCC.6
{                          //  当 VDD > 4V
}
else
{                          //  当 VDD < 4V
}
```

5.6 VDD/2 LCD Bias 电压生成器

PFS123 有 5 个引脚：PB0、PB1、PB2、PB5 和 PB6，可以作为 LCD 应用的 COM 端口。通过设定 misc.4=1 这五个 COM 端口能够输出高电位(V_{DD})、输入(VDD/2)、输出低电位(GND)三种电压。

COM 端口和正常的 IO 端口一样在输出模式 (***pbc.x***=1) 下通过选择 ***pb.x*** 的 1 或者 0 输出 V_{DD} 和 GND 电压。同样，COM 端口通过进入输入模式 (***pbc.x***=0)能输出 VDD/2 电压。然而，要注意关闭上拉电阻 ***pbph.x*** 和 ***pbdier.x*** 防止输出电压受到干扰。图 9 显示如何使用此功能。

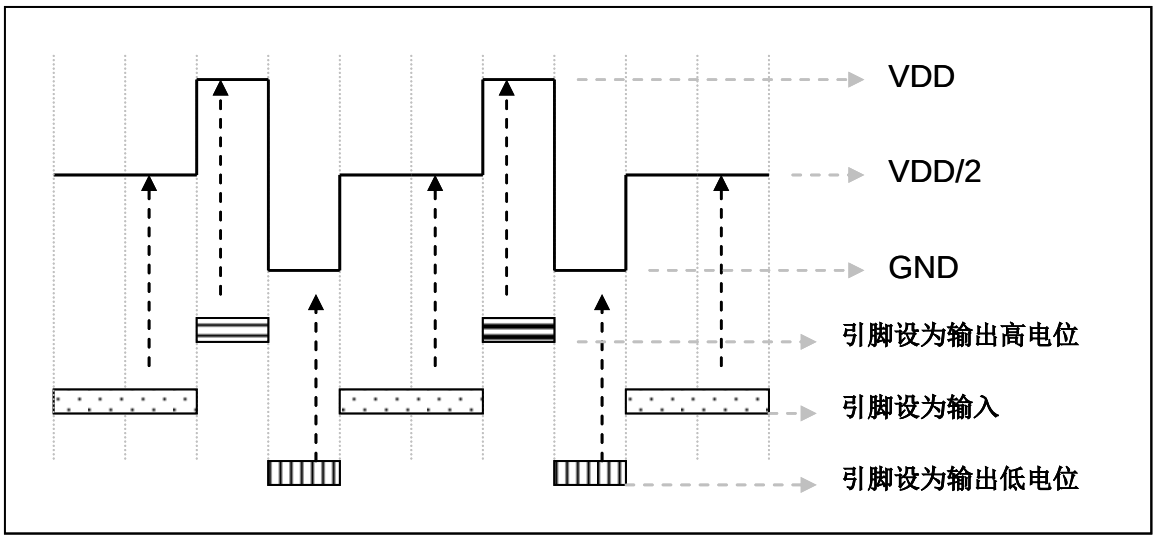


图 9：使用 VDD/2 LCD bias 电压生成器

注意：仿真器不支持 PB0 的 VDD/2 功能。

5.7 16 位计数器(Timer16)

PFS123 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)、外部晶体振荡器时钟(EOSC)、内部高频振荡时钟(IHRC)、内部低频振荡时钟(ILRC)、PA4 和 PA0，一个多任务器用来选择时钟输出的时钟来源。在送到 16 位计数器之前，1 个可软件编程的预分频器提供÷1、÷4、÷16、÷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。可软件编程的选择器用于选择 Timer16 的中断条件，当计数器溢出时，Timer16 可以触发中断。Timer16 模块框图如图 10 所示。中断源是来自 16 位计数器的位 8 到位 15，中断类型可以上升沿触发或下降沿触发，定义在寄存器 ***integs*.5**（IO 地址是 0x0C）。

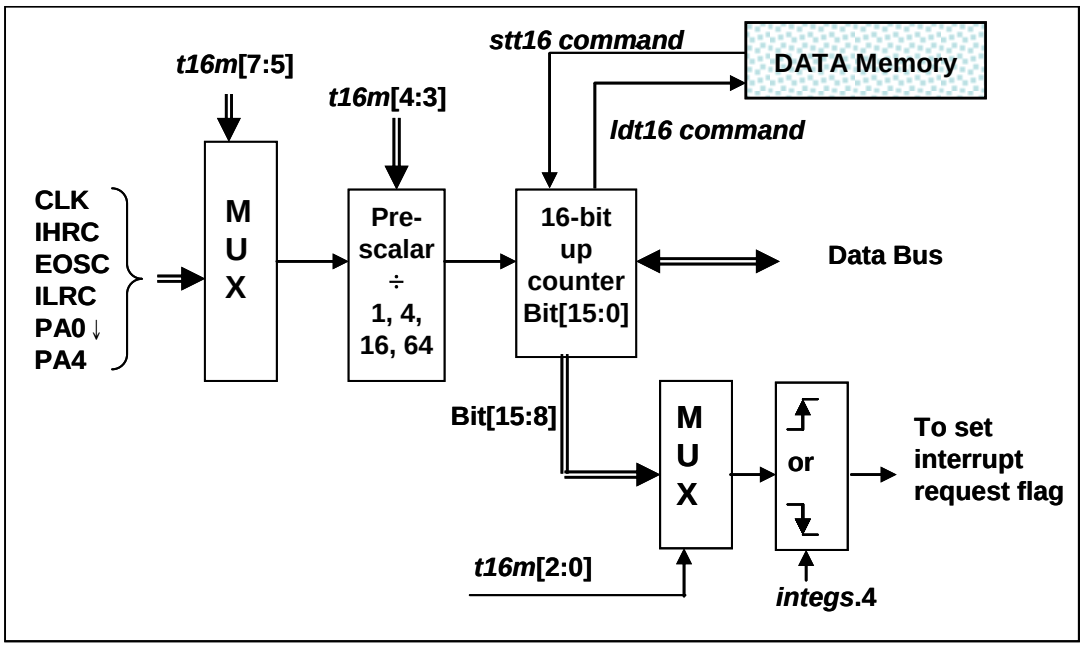


图 10：Timer16 模块框图

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, EOSC, 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 / 2 = 8 MHz
// SYSCLK/64 = 8 MHz/64 = 125kHz，约每 524 mS 产生一次 INTRQ.2=1

$ T16M EOSC, /1, BIT13;
// 选择(EOSC/1)当 Timer16 时钟源，每 2^14 个时钟周期产生一次 INTRQ.2=1
//  EOSC=32768 Hz, 32768 Hz/(2^14) = 2Hz，每 0.5S 产生一次 INTRQ.2=1

$ T16M PA0_F, /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
```

假如 Timer16 是不受干扰自由运行，中断发生的频率可以用下列式子描述：

$$F_{INTRQ_T16M} = F_{clock\ source} \div P \div 2^{n+1}$$

其中，F 是 Timer16 的时钟源频率；

P 是 t16m [4:3]的选项(比如 1, 4, 16, 64)；

N 是中断要求选择的位，例如：选择位 10，那么 n=10。

5.8 8 位 PWM 计数器 (Timer2, Timer3)

PFS123 内置 2 个 8 位硬件 PWM 计数器(Timer2/Timer3)。以下描述只以 Timer2 为例，因为 Timer3 和 Timer2 结构是一样的。图 11 为 Timer2 硬件框图，计数器的时钟源可以来自系统时钟(CLK)，内部高频 RC 振荡器时钟(IHRC)，内部低频 RC 振荡器时钟(ILRC)，外部晶体振荡器(EOSC)，PA0，PB0，PA4 和比较器。寄存器 tm2c 的位[7:4]用来选择 Timer2 的时钟。如果 IHRC 作为 Timer2 的时钟源，当仿真器停住时，IHRC 时钟仍然会送到 Timer2，所以 Timer2 仍然会计数。根据 tm2c 寄存器位[3:2]的设定，Timer2 的输出可以是 PB2，PA3 或 PB4 引脚。利用软件编程寄存器 ***tm2s*** 位[6:5]，时钟预分频模块提供÷1，÷4，÷16 和÷64 的选择，另外，利用软件编程寄存器 ***tm2s*** 位[4:0]，时钟分频器的模块提供了÷1~÷32 的功能。在结合预分频器以及分频器，Timer2 时钟(TM2_CLK)频率可以广泛和灵活，以提供不同产品应用。

8 位 PWM 定时器只能执行 8 位上升计数操作，经由寄存器 ***tm2ct***，定时器的值可以设置或读取。当 8 位定时器计数值达到上限寄存器设定的范围时，定时器将自动清除为零，上限寄存器用来定义定时器产生波形的周期或 PWM 占空比。8 位 PWM 定时器有两个工作模式：周期模式和 PWM 模式；周期模式用于输出固定周期波形或中断事件；PWM 模式是用来产生 PWM 输出波形，PWM 分辨率可以为 6 位到 8 位。图 11 显示出 Timer2 周期模式和 PWM 模式的时序图。

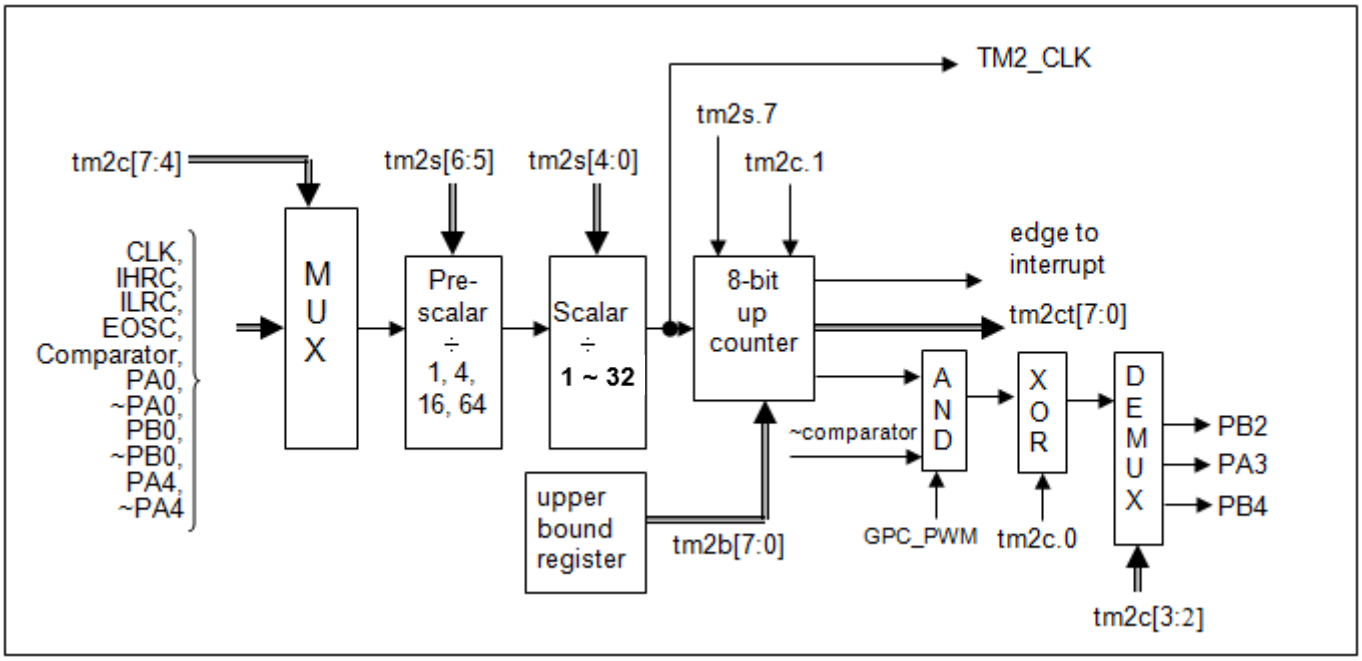


图 11：Timer2 硬件框图

Timer3 的输出可以是 PB5，PB6 或 PB7。

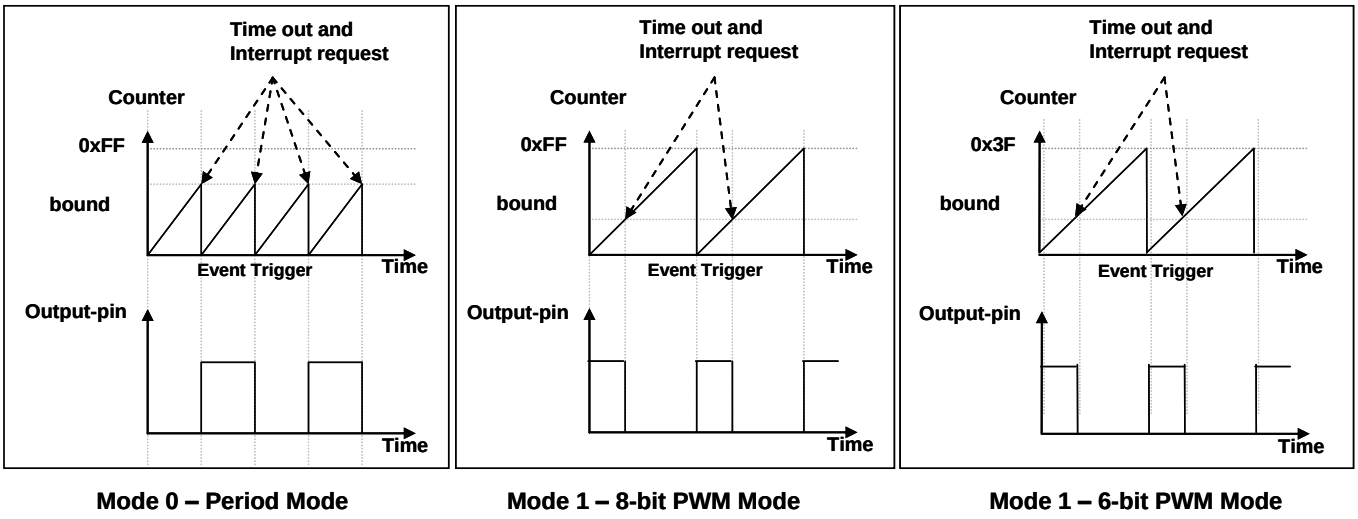


图 12：Timer2 周期模式和 PWM 模式的时序图(tm2c.1=1)

程序选项" GPC_PWM "是指根据需求由比较器结果控制生成 PWM 波形的功能。如果程序选项"GPC_PWM" 被选中后，此时当比较器输出是 1 时，PWM 停止输出；而比较器输出是 0 时，PWM 恢复输出，如图 13 所示。

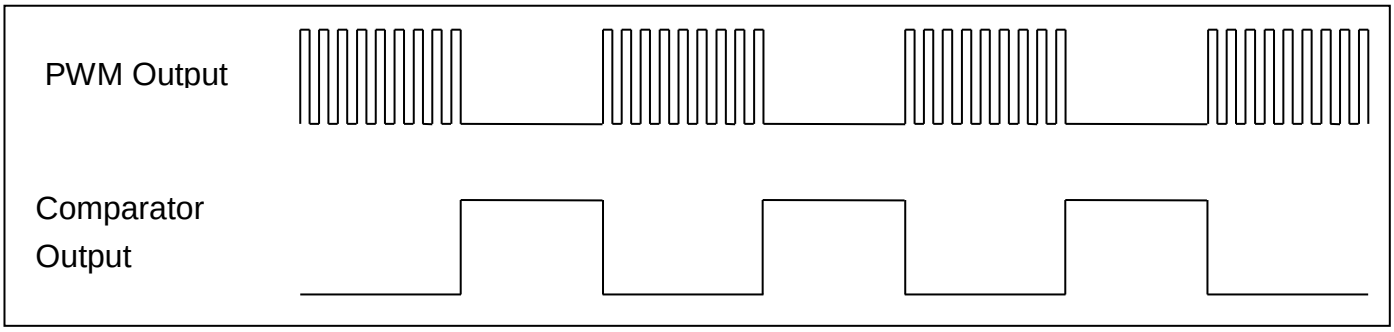


图 13：比较器控制 PWM 波形的输出

5.8.1 使用 Timer2 产生周期波形

如果选择周期模式的输出，输出波形的占空比总是 50%，其输出频率与寄存器设定，可以概括如下：

输出频率= Y ÷ [2 × (K+1) × S1 × (S2+1)]

Y = tm2c[7:4] : Timer2 所选择的时钟源频率

K = tm2b[7:0] : 上限寄存器设定的值（十进制）

S1 = tm2s[6:5] : 预分频器设定值 (S1= 1, 4, 16, 64)

S2 = tm2s[4:0] : 分频器值（十进制，S2= 0 ~ 31）

例 1:

tm2c = 0b0001_1000, Y=8MHz

tm2b = 0b0111_1111, K=127

tm2s = 0b0_00_00000, S1=1, S2=0

输出频率 = 8MHz ÷ [2 × (127+1) × 1 × (0+1)] = 31.25KHz

例 2:

tm2c = 0b0001_1000, Y=8MHz

tm2b = 0b0111_1111, K=127

tm2s[7:0] = 0b0_11_11111, S1=64 , S2 = 31

输出频率= 8MHz ÷ (2 × (127+1) × 64 × (31+1)) =15.25Hz

例 3:

tm2c = 0b0001_1000, Y=8MHz

tm2b = 0b0000_1111, K=15

tm2s = 0b0_00_00000, S1=1, S2=0

输出频率= 8MHz ÷ (2 × (15+1) × 1 × (0+1)) = 250KHz

例 4:

tm2c = 0b0001_1000, Y=8MHz

tm2b = 0b0000_0001, K=1

tm2s = 0b0_00_00000, S1=1, S2=0

输出频率= 8MHz ÷ (2 × (1+1) × 1 × (0+1)) =2MHz

使用 Timer2 定时器从 PA3 引脚产生周期波形的示例程序如下所示：

```
Void   FPPA0 (void)
{
    . ADJUST_IC    SYSCLK=IHRC/2, IHRC=16MHz, VDD=5V
    …
    tm2ct = 0x00;
    tm2b = 0x7f;
    tm2s = 0b0_00_00001;                  //    8-bit PWM，预分频 = 1，分频 = 2
    tm2c = 0b0001_10_0_0;                 //    系统时钟，输出=PA3，周期模式
    while(1)
    {
        nop;
    }
}
```

5.8.2 使用 Timer2 产生 8 位 PWM 波形

如果选择 8 位 PWM 的模式，应设立 tm2c [1] = 1，tm2s [7] = 0，输出波形的频率和占空比可以概括如下：

输出频率= Y ÷ [256 × S1 × (S2+1)]

输出占空比= [(K＋1) ÷ 256]×100%

Y = tm2c[7:4] : Timer2 所选择的时钟源频率

K = tm2b[7:0] : 上限寄存器设定的值（十进制）

S1= tm2s[6:5] : 预分频器设定值 (S1= 1, 4, 16, 64)

S2 = tm2s[4:0] : 分频器值（十进制，S2= 0 ~ 31）

例 1:

tm2c = 0b0001_1010, Y=8MHz
tm2b = 0b0111_1111, K=127
tm2s = 0b0_00_00000, S1=1, S2=0
输出频率 = 8MHz ÷ (256 × 1 × (0+1)) = 31.25KHz
输出占空比 = [(127+1) ÷ 256] × 100% = 50%

例 2:

tm2c = 0b0001_1010, Y=8MHz
tm2b = 0b0111_1111, K=127
tm2s = 0b0_11_11111, S1=64, S2=31
输出频率 = 8MHz ÷ (256 × 64 × (31+1)) = 15.25Hz
输出占空比 = [(127+1) ÷ 256] × 100% = 50%

例 3:

tm2c = 0b0001_1010, Y=8MHz
tm2b = 0b1111_1111, K=255
tm2s = 0b0_00_0000, S1=1, S2=0
PWM 输出是高电平
输出占空比 = [(255+1) ÷ 256] × 100% = 100%

例 4:

tm2c = 0b0001_1010, Y=8MHz
tm2b = 0b0000_1001, K = 9
tm2s = 0b0_00_0000, S1=1, S2=0
输出频率 = 8MHz ÷ (256 × 1 × (0+1)) = 31.25KHz
输出占空比 = [(9+1) ÷ 256] × 100% = 3.9%

©Copyright 2023, PADAUK Technology Co. Ltd

使用 Timer2 定时器从 PA3 产生 PWM 波形的示例程序如下所示：

```
void    FPPA0 (void)
{
    .ADJUST_IC      SYSCLK=IHRC/2, IHRC=16MHz, VDD=5V
    tm2ct = 0x00;
    tm2b = 0x7f;
    tm2s = 0b0_00_00001;                // 8-bit PWM，预分频 = 1，分频 = 2
    tm2c = 0b0001_10_1_0;               //系统时钟，输出=PA3，PWM 模式
    while(1)
    {
        nop;
    }
}
```

5.8.3 使用 Timer2 产生 6 位 PWM 波形

如果选择 6 位 PWM 的模式，应设立 tm2c [1] = 1，tm2s [7] = 1，输出波形的频率和占空比可以概括如下：

输出频率= Y ÷ [64 × S1 × (S2+1)]

输出占空比= [(K＋1) ÷ 64] × 100%

tm2c[7:4] = Y : Timer2 所选择的时钟源频率
tm2b[7:0] = K : 上限寄存器设定的值（十进制）
tm2s[6:5] = S1 : 预分频器设定值 (S1= 1, 4, 16, 64)
tm2s[4:0] = S2 : 分频器值（十进制，S2= 0 ~ 31）

用户可以通过设置程序选项中的 TMx_Bit 把 Timer2 由 6 位 PWM 模式改成 7 位 PWM 模式。此时，要把计算式中的 64 改为 128。

例 1:

tm2c = 0b0001_1010, Y=8MHz
tm2b = 0b0001_1111, K=31
tm2s = 0b1_00_00000, S1=1, S2=0
输出频率 = 8MHz ÷ (64 × 1 × (0+1)) = 125KHz
输出占空比 = [(31+1) ÷ 64] × 100% = 50%

例 2:

tm2c = 0b0001_1010, Y=8MHz
tm2b = 0b0001_1111, K=31
tm2s = 0b1_11_11111, S1=64, S2=31
输出频率 = 8MHz ÷ (64 × 64 × (31+1)) = 61.03 Hz
输出占空比 = [(31+1) ÷ 64] × 100% = 50%

©Copyright 2023, PADAUK Technology Co. Ltd

例 3:

tm2c = 0b0001_1010, Y=8MHz
tm2b = 0b0011_1111, K=63
tm2s = 0b1_00_00000, S1=1, S2=0
PWM 输出高电平
输出占空比 = [(63+1) ÷ 64] × 100% = 100%

例 4:

tm2c = 0b0001_1010, Y=8MHz
tm2b = 0b0000_0000, K=0
tm2s = 0b1_00_00000, S1=1, S2=0
输出频率 = 8MHz ÷ (64 × 1 × (0+1)) = 125KHz
输出占空比 = [(0+1) ÷ 64] × 100% =1.5%

5.9 11 位 PWM 计数器

PFS123 内置一组三路 11 位 SuLED (Super LED)硬件 PWM 生成器(PWMG0、PWMG1 和 PWMG2)。各路输出端口如下：

- PWMG0 – PA0, PB4, PB5, PC2, PB6（适用于 PWMG1 没有选择 PB6 的情况下）
- PWMG1 – PA4, PB6, PB7, PC3
- PWMG2 – PA3, PB2, PB3, PA5（只有开漏输出）, PC0, PB5（适用于 PWMG0 没有选择 PB5 的情况下）

请注意：5S-I-S01/2(B)不支持仿真整组 11 位的 SuLED PWM 生成器的功能。

5.9.1 PWM 波形

PWM 输出波形（图 14）有一个时基（T_{Period} = 时间周期）和一个周期里输出高电平的时间（占空比）。PWM 输出的频率取决于时基($f_{PWM} = 1/T_{Period}$)。

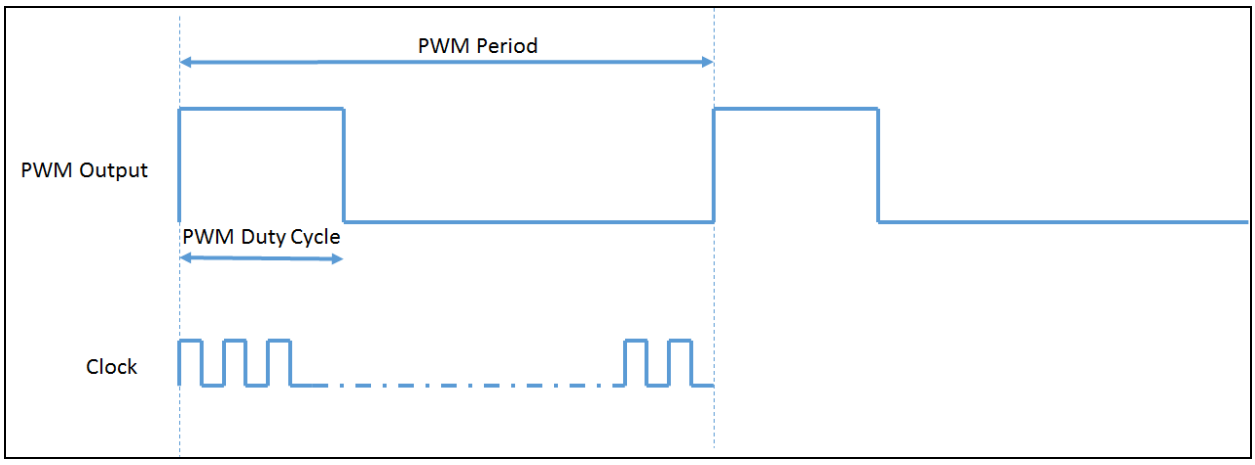


图 14：PWM 输出波形

5.9.2 硬件框图

图 15 所示是整组 SuLED 11 位 PWM 生成器的硬件方框图。这三组 PWM 生成器使用共同的 Up-Counter 和时钟源选择开关来产生时基，所以 PWM 周期的起点（上升沿）是同步的，时钟源可以是 IHRC 或者系统时钟。PWM 信号输出引脚通过 pwmgxc 寄存器来选择。PWM 波形的周期由 PWM 上限高和低寄存器决定，各路 PWM 波形的占空比由各路 PWM 占空比高和低寄存器决定。

在 PWMG0 通道的那两个附带的 OR 和 XOR 逻辑门是用于产生互补非重叠并有死区的开关控制波形的。用户也可以通过使用程序选项 GPC_PWM，使比较器结果也能控制 PWM 波形的输出。

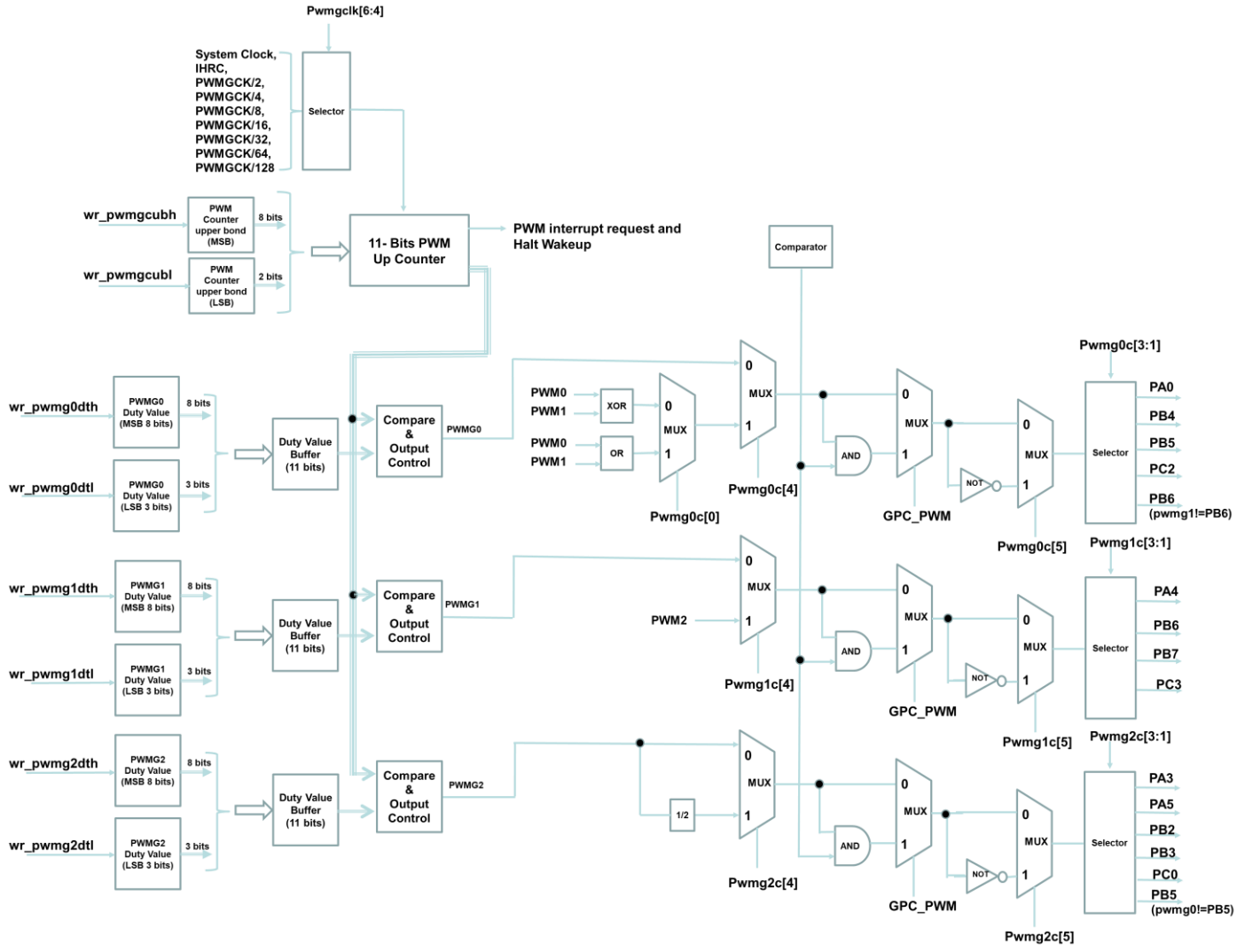


图 15：整组 SuLED 三路 11 位 PWM 生成器硬件框图

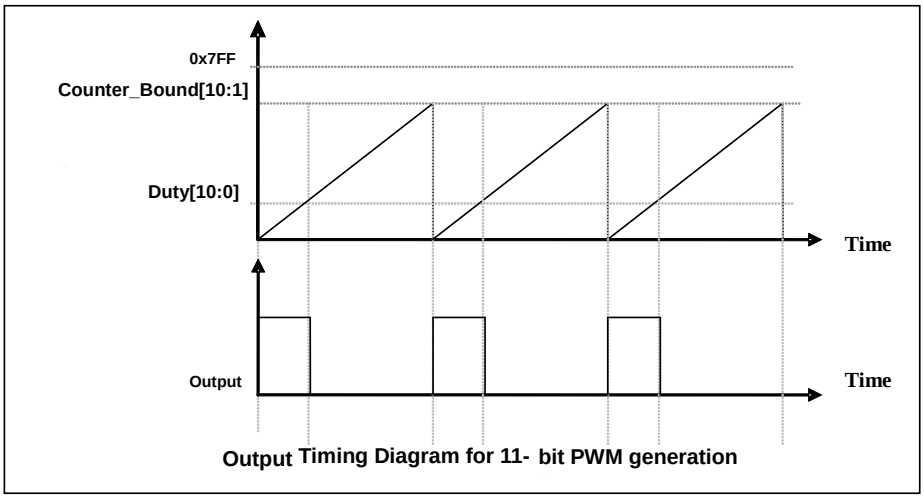


图 16：11 位 PWM 生成器输出时序图

5.9.3 11 位 PWM 生成器计算公式

PWM 输出频率 F_{PWM} = $F_{clock\ source}$ ÷ [P × (CB10_1 + 1)]

PWM 占空比（时间） = (1 / F_{PWM}) × (DB10_1 + DB0 × 0.5 + 0.5) ÷ (CB10_1 + 1)

PWM 占空比（百分比）= (DB10_1 + DB0 × 0.5 + 0.5) ÷ (CB10_1 + 1) × 100%

这里,

P=*PWMGCLK*[6:4]; 预分频 **P**=1,2,4,8,16,32,64,128

DB10_1 = Duty_Bound[10:1] = {*PWMGxDTH* [7:0],*PWMGxDTL*[7:6]}, 占空比

DB0 = Duty_Bound[0] = *PWMGxDTL*[5]

CB10_1 = Counter_Bound[10:1] = {*PWMGCUBH*[7:0],*PWMGCUBL*[7:6]}, 计数器

5.9.4 带互补死区的 PWM 波形范例

基于 PFS123 独特的 11 bit PWM 结构，在此采用 PWM2 输出、PWM0 与 PWM1 异或后通过 PWM0 反极性输出，来获得两路互补带死区 PWM 波形。示例如下：

```
#define   dead_zone       10      //   死区时间 = 10% * (1/PWM_Frequency) us
#define   PWM_Pulse       50      //   该互补死区 PWM 占空比为 50%

#define   PWM_Pulse_1     35      //   该互补死区 PWM 占空比为 35%
#define   PWM_Pulse_2     60      //   该互补死区 PWM 占空比为 60%
#define   switch_time     400*2   //   切换占空比时，用于调整切换时间
//Note：为防止杂波产生，switch_time 应为 PWM 周期的倍数。此例 PWM 周期：1/2.5KHz = 400 us，故切
//换时间为 400*2 us

void FPPA0 (void)
{
.ADJUST_IC      SYSCLK=IHRC/16, IHRC=16MHz, VDD=5V;
//******* 产生固定占空比 ************************
//------- 设置计数上限及占空比 --------------------
PWMG0DTL      =     0x00;
PWMG0DTH      =     PWM_Pulse + dead_zone;

PWMG1DTL      =     0x00;
PWMG1DTH      =     dead_zone;                  // PWMG0 与 PWMG1 异或后，PWM 占空比
                                                // 为 PWM_Pulse%
PWMG2DTL      =     0x00;
PWMG2DTH      =     PWM_Pulse + dead_zone*2;

PWMGCUBL      =     0x00;
PWMGCUBH      =     100;
//---- 统一配置 PWM 时钟及分频 ----------------
$ PWMGCLK     Enable, /1, sysclk;
//------- 输出控制 -----------------------------
$ PWMG0C  Enable,Inverse,PWM_Gen,PA0,gen_xor;   //   PWMG0 与 PWMG1 异或后，从
                                                //   PA0 脚反极性输出
$ PWMG1C  Enable, PWMG1,disable;                //   PWMG1 不输出
$ PWMG2C      Enable, PA3;                      //   PWMG2 PA3 输出

while(1)
{
    //******** 切换占空比 *********************************
    // 切换占空比时，为避免可能出现的瞬间死区消失，应遵循如下顺序。
    // 占空比由大变小：50%/60%  →  35%
    PWMG0DTL =      0x00;
    PWMG0DTH =      PWM_Pulse_1 + dead_zone;
```

```
    PWMG2DTL =      0x00;
    PWMG2DTH =      PWM_Pulse_1 + dead_zone*2;
    .delay      switch_time

    //占空比由小变大：35% → 60%
    PWMG2DTL =      0x00;
    PWMG2DTH =      PWM_Pulse_2 + dead_zone*2;
    PWMG0DTL =      0x00;
    PWMG0DTH =      PWM_Pulse_2 + dead_zone;
    .delay      switch_time
}
}
```

上述程序中，固定占空比时对应的 PWM0/PWM2 波形如图 17 所示。

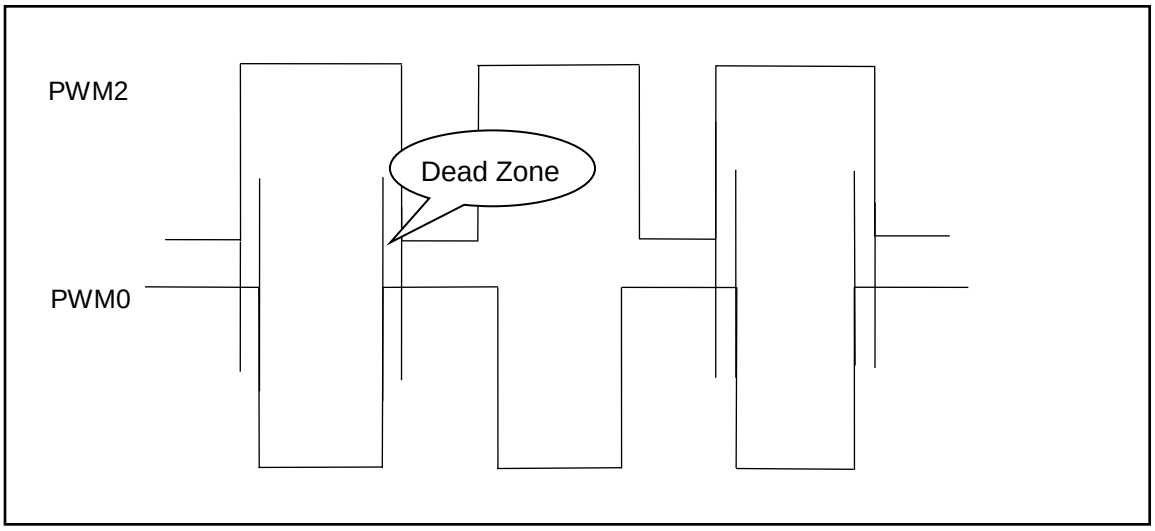


图 17：两路互补 PWM 波形

切换占空比时对应的 PWM0/PWM2 波形如图 18。

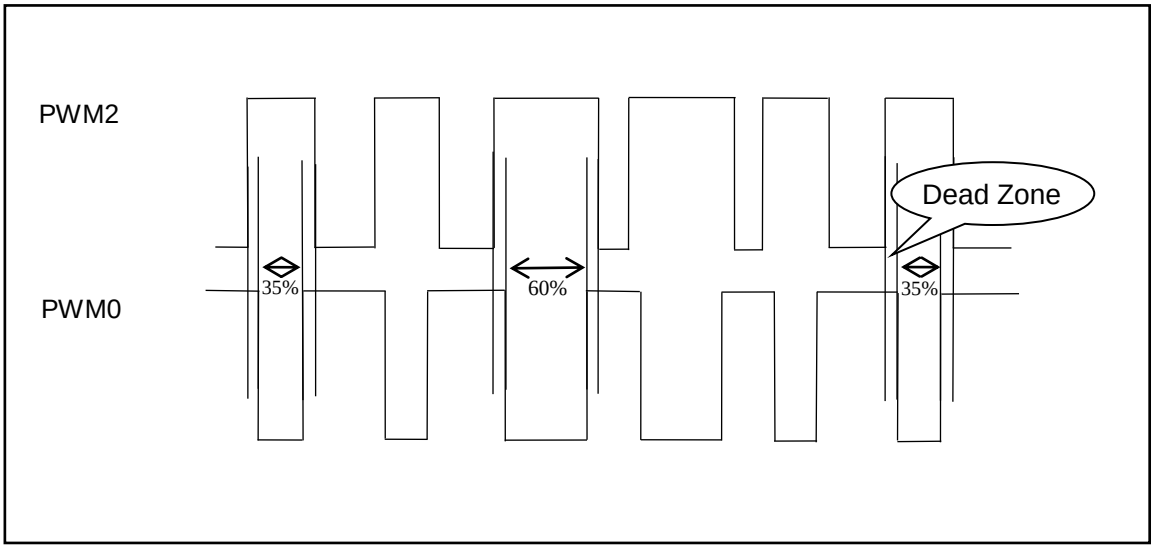


图 18：两路互补 PWM 波形

©Copyright 2023, PADAUK Technology Co. Ltd

可以发现，上述例程所得波形，其死区是两组 PWM 同时为高。若用户需要 PWM 同时为低的死区，仅需改变各自输出控制的 Inverse 即可。如：

```
$ PWMG0C    Enable,PWM_Gen,PA0,gen_xor;
$ PWMG2C    Enable, Inverse, PA3;
```

5.10 看门狗

看门狗是一个计数器，其时钟源来自内部低频振荡器(ILRC)，可以通过上电复位和 wdreset 指令随时清零看门狗计数，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，如下：

- 当 misc[1:0]=00（默认）时：8k ILRC 时钟周期
- 当 misc[1:0]=01 时：16k ILRC 时钟周期
- 当 misc[1:0]=10 时：64k ILRC 时钟周期
- 当 misc[1:0]=11 时：256k 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，PFS123 将复位并重新运行程序。看门狗时序图如图 19 所示。

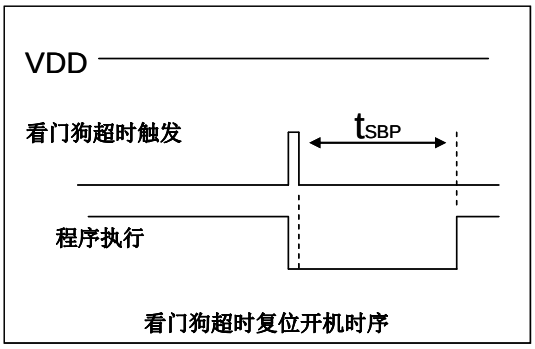


图 19：看门狗超时溢出时序图

5.11 中断

PFS123 有 8 个中断源：

- 外部中断源 PA0/PB5
- 外部中断源 PB0/PA4
- ADC 中断源
- Timer16 中断源
- GPC 中断源
- PWMG 中断源
- Timer2 中断源
- Timer3 中断源

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 20 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 ***integs*** 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。

注：可在 Code Option Interrupt Src0 或 Interrupt Src1 中切换外部中断源。

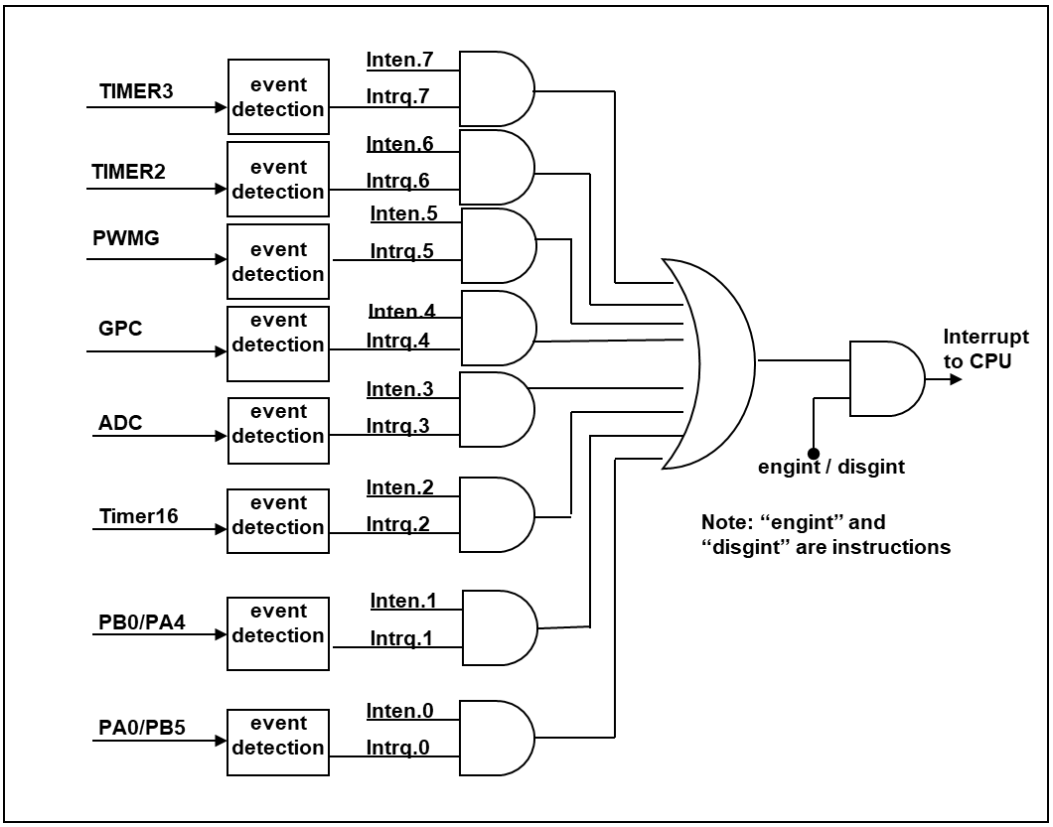


图 20：中断控制器硬件框图

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。

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 在主程序会动态开和关，则表达式中可以判断 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;  // 只须清除相对应的位 (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                  //回复 ALU 和 FLAG 寄存器
}
```

5.12 省电与掉电

PFS123 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。

5.12.1 省电模式(“*stopexe*”)

使用 ***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。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：

(1) IHRC 和 EOSC 振荡器模块：没改变，如果被启用，则仍然保持运行状态；
(2) ILRC 振荡器模块：必须保持启用，唤醒时需要靠 ILRC 启动。
(3) 系统时钟：停用，因此 CPU 停止运行；
(4) MTP 存储器关闭；
(5) Timer 计数器：若 Timer 计数器的时钟源是系统时钟或其相应的时钟振荡器模块被停用，则 Timer 停止计数；否则，仍然保持计数。（其中，Timer 包含 Timer16，TM2，TM3，PWMG0，PWMG1，PWMG2）；
(6) 唤醒源：
    a. IO Toggle 唤醒： IO 在数字输入模式下的电平变换（*PxC* 位是 0，*PxDIER* 位是 1）
    b. Timer 唤醒：如果计数器 (Timer)的时钟源不是系统时钟，则当计数到设定值时，系统会被唤醒。
    c. 比较器唤醒：使用比较器唤醒时，需同时设定 *GPCC*.7 为 1 与 *GPCS*.6 为 1 来启用比较器唤醒功能。但请注意：内部 1.20V Bandgap 参考电压不适用于比较器唤醒功能。

以下例子是利用 Timer16 来唤醒系统因 ***stopexe*** 的省电模式：

```
$ T16M      ILRC, /1, BIT8          // Timer16 设置
$ INTEGS    BIT_R, xxx;             // BITx 0 到 1 会触发（默认）

...
WORD        count =     0;
STT16       count;
stopexe;
...
```

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

5.12.2 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。在下达 *stopsys* 指令之前建议将 *GPCC*.7 设为 0 来关闭比较器。输入引脚的唤醒可视为程序正常运行，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。下面显示发出 stopsys 命令后，如下是 PFS123 内部详细的状态：

(1) 有的振荡器模块被关闭；
(2) MTP 存储器被关闭；
(3) SRAM 和寄存器内容保持不变；
(4) 唤醒源：IO 在数字输入模式下电平变换（PxDIER 位是 1）

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;          //    停用 IHRC
    …
    while (1)
{
              STOPSYS;          //    进入断电模式
              if  (…)  break;   //    假如发生唤醒而且检查 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/2
```

5.12.3 唤醒

进入掉电或省电模式后，PFS123 可以通过切换 IO 引脚恢复正常工作，而 Timer 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	计时器唤醒	比较器唤醒
STOPSYS	是	否	否
STOPEXE	是	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 PFS123， *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 3000 个 ILRC 时钟周期，另外，PFS123 提供快速唤醒功能，透过 *misc* 寄存器选择快速唤醒大约 45 个 ILRC 时钟周期。

模式	唤醒模式	切换 IO 引脚的唤醒时间(t_{WUP})
STOPEXE 省电模式 STOPSYS 掉电模式	快速唤醒	45 * T_{ILRC}, 这里的 T_{ILRC} 是指 ILRC 时钟周期
STOPEXE 省电模式 STOPSYS 掉电模式	正常唤醒	3000* T_{ILRC}, 这里的 T_{ILRC} 是指 ILRC 时钟周期

请注意：当使用快速开机模式时，不管寄存器 misc.5 是否选择了唤醒模式，都会强制使用快速唤醒模式。如果选择正常开机模式，即由寄存器 misc.5 来选择唤醒模式。

©Copyright 2023, PADAUK Technology Co. Ltd

5.13 IO 引脚

PFS123 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, pc***)，控制寄存器(***pac, pbc, pcc***)和上拉寄存器(***paph, pbph, pcph***) 。其中有四个引脚 PB2、PB3、PC0 和 PC1 增加下拉寄存器(***pbpl.2, pbpl.3, pcpl.0, pcpl.1***)。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/下拉电阻会自动关闭.如果用户要读取端口上的电位状态，一定要先设置成输入模式，因为在输出模式下，读取到的数据是数据寄存器的值，而不是 IO 端口的值。举例，表 6 为端口 PA0 位的设定配置表。 图 21 显示了 IO 缓冲区硬件图。

pa.0	*pac.0*	*paph.0*	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（弱上拉电阻自动关闭）
1	1	0	输出高电位，没有弱上拉电阻
1	1	1	输出高电位，有弱上拉电阻

表 6：PA0 设定配置表

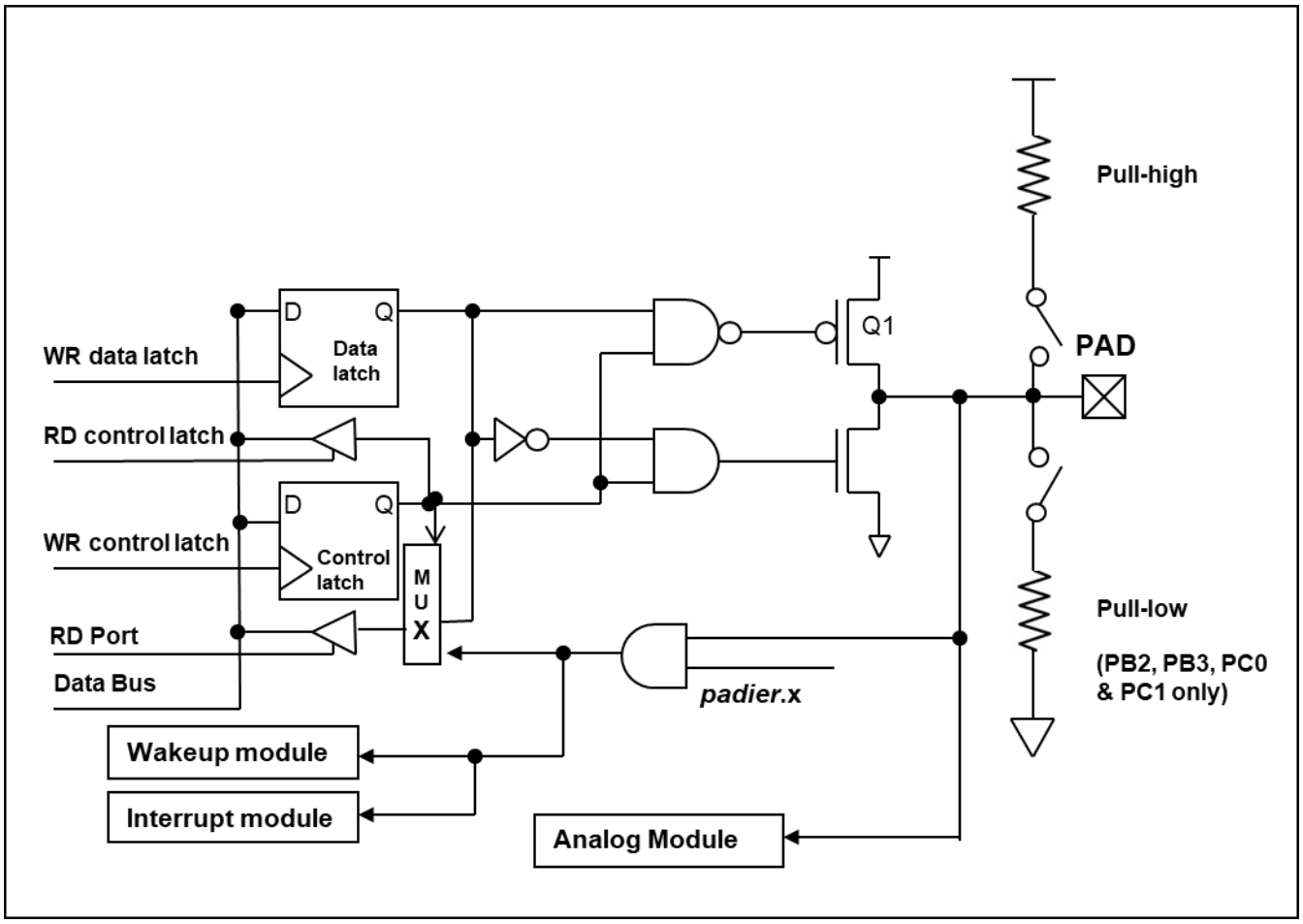


图 21：IO 引脚缓冲区硬件图

PB4 和 PB5 通过程序选项 ***PB4_PB5_Drive*** 来调整驱动电流和灌电流。

©Copyright 2023, PADAUK Technology Co. Ltd

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 / pcdier*** 相应位设置为低，以防止漏电流，当 PFS123 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。因此，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 ***pxdier*** 为高电平。同样的原因，当 PA0 用作外部中断引脚时，padier.0 应设置为高，诸如 pbdier.0 对于 PB0，padier.4 对于 PA4 和 pbdier.5 对于 PB5，都是同样的用法。

5.14 复位和 LVR

5.14.1 复位

引起 PFS123 复位的原因很多，一旦复位发生，PFS123 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 0x00。当发生上电复位或 LVR 复位，若 VDD 大于 V_{DR}（数据保存电压），数据存储器的值将会被保留，但若在重新上电后 SRAM 被清除，则数据无法保留；若 VDD 小于 V_{DR}，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。

5.14.2 LVR 复位

通过程序选项(code option)可以看到，有很多不同级别的 LVR 复位电压可供选择，通常情况下，使用者在选择 LVR 复位电压时，必须结合单片机工作频率和电源电压，以便让单片机稳定工作。

5.15 模拟-数字转换器(ADC) 模块

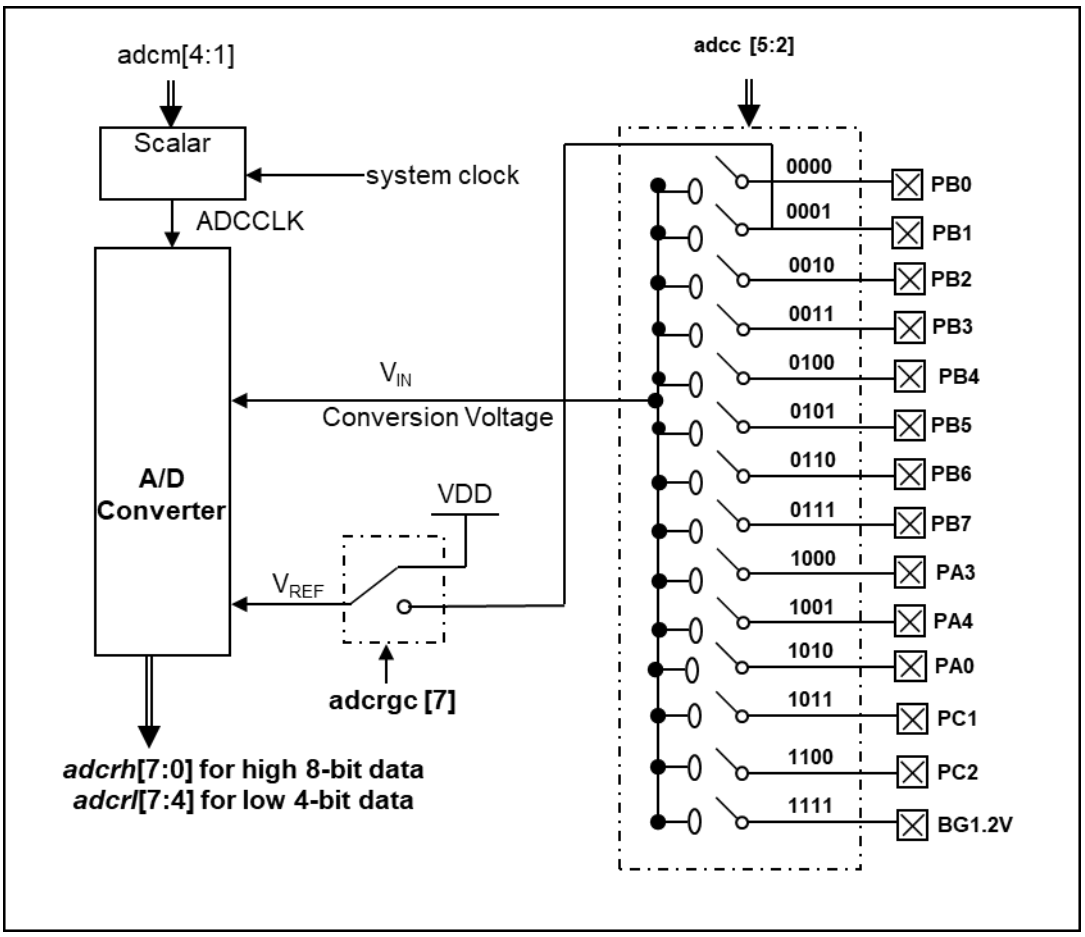


图 22：ADC 模块框图

当使用 ADC 模块时有 8 个寄存器需要配置，它们是：

- ADC 控制寄存器 (***adcc***)
- ADC 调节控制寄存器(***adcrgc***)
- ADC 模式寄存器 (***adcm***)
- ADC 结果寄存器(***adcrh, adcrl***)
- 端口 A/B/C 数字输入启用寄存器 (***padier***, ***pbdier, pcdier***)

如下是 ADC 转换进程的步骤：

(1) 通过寄存器 ***adcrgc*** 配置参考高电压

(2) 通过 ***adcm*** 寄存器配置 AD 转换时钟信号

(3) 通过 ***padier***，***pbdier, pcdier*** 寄存器配置模拟输入引脚

(4) 通过 ***adcc*** 寄存器选择 ADC 输入通道

(5) 通过 ***adcc*** 寄存器启用 ADC 模块

(6) 执行 AD 转换并检查 ADC 转换数据是否已经完成
addc.6 设置 1 开启 AD 转换并且检测 ***addc***.6 是否是‘1’

(7) 从 ADC 寄存器读取转换结果

5.15.1 AD 转换的输入要求

为了满足 AD 转换的精度要求，电容的保持电荷(C_{HOLD})必须完全充电到参考高电压的水平和放电到参考低电压的水平。模拟输入电路模型如图 23 所示，信号驱动源阻抗(Rs)和内部采样开关阻抗(Rss)会直接影响到电容 C_{HOLD} 充电所需求的时间。内部采样开关的阻抗可能会因 ADC 充电电压而产生变化；信号驱动源阻抗会影响模拟输入信号的精度。使用者必须确保在采样前，被测信号的稳定，因此，信号驱动源阻抗的最大值与被测信号的频率高度相关。建议，在输入频率为 500khz 下，模拟信号源的最大阻抗值不要超过 10KΩ。

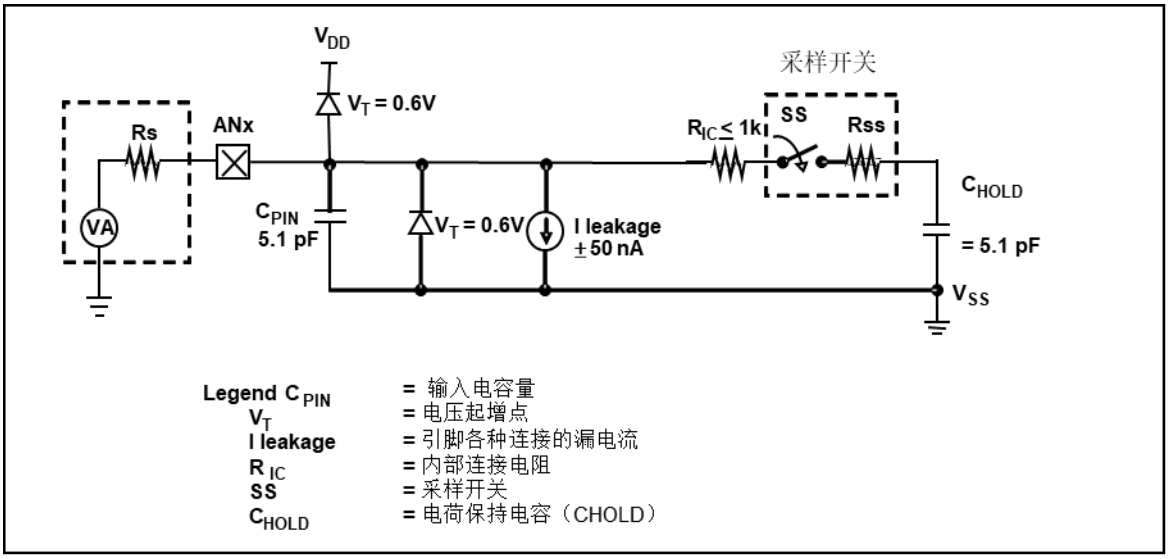


图 23：模拟输入模型

5.15.2 选择参考高电压

ADC 参考高电压能够通过寄存器 adcrgc 的位[7]来选择，并且它的选择有 V_{DD} 或者来自 PB1 外部引脚。

5.15.3 ADC 时钟选择

在使用 AD 转换之前，必须确认所选的模拟输入信号的采集时间应符合要求，ADCLK 的选择必须满足最短信号采集时间。

ADC 模块的时钟(ADCLK)能够通过 *adcm* 寄存器来选择，ADCLK 从 CLK÷1 到 CLK÷128 一共有 8 个选项可被选择（CLK 是系统时钟）。由于信号采集时间 TACQ 是 ADCLK 的一个时钟周期，所以 ADCLK 必须满足这一要求，建议 ADC 时钟周期是 2us。

5.15.4 配置模拟引脚

有 14 模拟信号可以被 AD 转换选择：13 来自外部引脚的模拟输入信号和一个 bandgap 参考电压 1.2V。这些外部引脚定义为模拟输入时，为避免从共享 IO 端口的数字电路产生的漏电，请务必记住要禁止数字输入的功能（设置寄存器 ***padier, pbdier or pcdier*** 对应位为 0）。

因为 ADC 的测量信号属于小信号，为避免测量信号在测量期间被干扰，被选定的引脚应：

(1) 设为输入模式，

(2) 关闭弱上拉电阻，

(3) 通过端口 A/B/C 寄存器 (***padier / pbdier / pcdier***).设置模拟输入并关闭数字输入。

5.15.5 使用 ADC

下面的示例演示使用 PB0～PB3 来当 ADC 输入引脚：

首先，定义所选择的引脚：

```
PBC     =   0B_XXXX_0000;     //   PB0 ~ PB3 作为输入
PBPH    =   0B_XXXX_0000;     //   PB0 ~ PB3 没有弱上拉电阻
PBPL    =   0B_XXXX_00XX;     //   PB2 ~ PB3 没有弱下拉电阻
PBDIER  =   0B_XXXX_0000;     //   PB0 ~ PB3 停用数字输入
```

下一步，设定 ADCC 寄存器，示例如下：

```
$   ADCC   Enable, PB3;       //   设置 PB3 作为 ADC 输入
$   ADCC   Enable, PB2;       //   设置 PB2 作为 ADC 输入
$   ADCC   Enable, PB0;       //   设置 PB0 作为 ADC 输入
```

下一步，设定 ADCM 和 ADCRGC 寄存器，示例如下：

```
$   ADCM   /16;               //   建议 /16 @系统时钟=8MHz
$   ADCM   /8;                //   建议 /8 @系统时钟=4MHz
$   ADCRGC VDD
```

©Copyright 2023, PADAUK Technology Co. Ltd

下一步，延迟 400us，示例如下：

```
.Delay 8*400;                    //    系统时钟=8MHz
.Delay 4*400;                    //    系统时钟=4MHz
```

接着，开始 ADC 转换：

```
AD_START =    1;                 //  开始 ADC 转换
while(!AD_DONE) NULL;            //  等待 ADC 转换结果
```

最后，当 AD_DONE 高电位时读取 ADC 结果：

```
WORD          Data;              //    两字节结果：放在 ADCRH 和 ADCRL
Data$1    =   ADCRH;
Data$0    =   ADCRL;
Data      =   Data >> 4;
```

ADC 也可以利用下面方法停用：

```
$   ADCC   Disable;
```

或

```
ADCC          =    0;
```

5.15.6 如何计算 ADC 输入电压 V_{IN}

由于 PFS122 的 ADC 不能使用 1.2V bandgap 电压，而只能使用 VDD，作为参考电压（V_{REF}）。在 IC 的 VDD 为非稳压的电压时，用户是需要利用 1.2V bandgap 电压的读数，以比值计算输入电压（V_{IN}）。原理如下：

$V_{BG} / V_{DD} = N_{BG} / 4096$ ....(1)

$V_{IN} / V_{DD} = N_{IN} / 4096$ ....(2)

此处：V_{IN} 为 ADC 模拟输入信号的电压

V_{BG} 为 1.2V bandgap 电压

N_{IN} 为 V_{IN} 的对应 ADC 读数

N_{BG} 为 V_{BG} 的对应 ADC 读数

V_{DD} 为测量瞬间的电源电压

4096 为 $V_{IN} = V_{DD}$ 时的满量程 ADC 读数（12bit：$2^{12} = 4096$）

(2)/(1) 可得：$V_{IN}/V_{BG} = N_{IN}/N_{BG}$

所以：$V_{IN} = N_{IN} / N_{BG} * V_{BG}$

也就是用户可在短时间内，以 VDD 不变为前提，先后读取 V_{IN} 和 V_{BG} 的读数，并以乘法和除法程序，或查表方式计算，便可得到 V_{IN} 的电压值。

若需要相关例程参考，请与 FAE 联络。

©Copyright 2023, PADAUK Technology Co. Ltd

6. IO 寄存器

6.1. ACC 状态标志寄存器(*flag*), IO 地址 = 0x00

位	初始值	读/写	描述
7 - 4	-	-	保留。
3	0	读/写	OV（溢出标志）。溢出时置 1。
2	0	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	0	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 shift 指令影响。
0	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	
			类型 0，clkmd[3]=0	类型 1，clkmd[3]=1
			000：IHRC÷4 001：IHRC÷2 010：保留 011：EOSC÷4 100：EOSC÷2 101：EOSC 110：ILRC÷4 111：ILRC（默认值）	000：IHRC÷16 001：IHRC÷8 010：ILRC÷16（仿真器不支持） 011：IHRC÷32 100：IHRC÷64 101：EOSC÷8 11x：保留
4	1	读/写	内部高频 RC 振荡器功能。 0/1：停用/启用	
3	0	读/写	时钟类型选择。这个位是用来选择位 7~位 5 的时钟类型。 0 / 1：类型 0 /类型 1	
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	读/写	启用从 Timer3 的溢出中断。0/1：停用/启用
6	0	读/写	启用从 Timer2 的溢出中断。0/1：停用/启用
5	0	读/写	启用从 PWMG 的溢出中断。0/1：停用/启用
4	0	读/写	启用从比较器的溢出中断。0/1：停用/启用
3	0	读/写	启用从 ADC 的溢出中断。0/1：停用/启用
2	0	读/写	启用从 Timer16 的溢出中断。0/1：停用/启用
1	0	读/写	启用从 PB0/PA4 的溢出中断。0/1：停用/启用
0	0	读/写	启用从 PA0/PB5 的溢出中断。0/1：停用/启用

6.5. 中断请求寄存器(*intrq*), IO 地址 = 0x05

位	初始值	读/写	描述
7	-	读/写	Timer3 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	-	读/写	Timer2 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	-	读/写	PWMG 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
4	-	读/写	比较器的中断请求，此位是由硬件置位并由软件清零。 0/1：不要求/请求
3	-	读/写	ADC 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
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	PB0/PA4 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	PA0/PB5 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

©Copyright 2023, PADAUK Technology Co. Ltd

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：停用 001：CLK（系统时钟） 010：保留 011：PA4 下降沿（从外部引脚） 100：IHRC 101：EOSC 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：Timer16 位 8 1：Timer16 位 9 2：Timer16 位 10 3：Timer16 位 11 4：Timer16 位 12 5：Timer16 位 13 6：Timer16 位 14 7：Timer16 位 15

6.7. 外部晶体振荡器控制寄存器(*eoscr*), IO 地址 = 0x0a

位	初始值	读/写	描述
7	0	只写	使能外部晶体振荡器。0 / 1：停用/使能
6 - 5	00	只写	晶体振荡器的选择。 00：保留 01：低驱动电流，适用于中等频率晶体，例如：32KHz 10：中驱动电流，适用于中等频率晶体，例如：1MHz 11：高驱动电流，适用于较高频率晶体，例如：4MHz
4 - 0	-	-	保留。请设为 0。

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

位	初始值	读/写	描述
7 - 5	-	-	保留。
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	PB0/PA4 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/PB5 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	1	只写	使能 PA7 数字输入和唤醒事件。1 / 0：启用/ 停用 当使用外部晶体振荡器的时候，该位设为 0 防止耗电。如果这个位设为 0，PA7 则不能用来唤醒系统。
6	1	只写	使能 PA6 数字输入和唤醒事件。1 / 0：启用/ 停用 当使用外部晶体振荡器的时候，该位设为 0 防止耗电。如果这个位设为 0，PA6 则不能用来唤醒系统。
5	1	只写	使能 PA5 数字输入和唤醒事件。1 / 0：启用/ 停用 该位设为 0，PA5 无法唤醒系统.
4	1	只写	使能 PA4 数字输入、唤醒事件和中断请求。1 / 0：启用/ 停用 当 PA4 作为 AD 输入时，该位设为 0 可以防止耗电。如果这个位设为 0，PA4 则不能用来唤醒系统，并且停用中断请求。
3	1	只写	使能 PA3 数字输入和唤醒事件。1 / 0：启用/ 停用 当 PA4 作为 AD 输入时，该位设为 0 可以防止耗电。如果这个位设为 0，PA3 则不能用来唤醒系统。
2 - 1	-	-	保留。（建议写 00）
0	1	只写	使能 PA0 数字输入、唤醒事件和中断请求。1 / 0：启用 / 停用 当 PA0 作为 AD 模拟输入时，该位设为 0 可以防止耗电。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。

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 - 6	11	只写	使能 PB7~PB6 数字输入和唤醒事件。0 / 1：停用 / 启用 当 PB7~PB6 作为 AD 输入时，这些位设 0 可以防止漏电。当选择停用时，这些引脚的唤醒功能也被停用。
5	1	只写	使能 PB5 数字输入、唤醒事件和中断请求。1 / 0：启用 / 停用 当 PB5 作为 AD 模拟输入时，该位设为 0 可以防止耗电。如果这个位设为 0，PB5 则不能用来唤醒系统，并且停用中断请求。
4 -1	1111	只写	使能 PB4~PB1 数字输入和唤醒事件。0 / 1：停用 / 启用 当 PB4~PB1 作为 AD 输入时，这些位设 0 可以防止漏电。当选择停用时，这些引脚的唤醒功能也被停用。
0	1	只写	使能 PB0 数字输入、唤醒事件和中断请求。1 / 0：启用 / 停用 当 PB0 作为 AD 模拟输入时，该位设为 0 可以防止耗电。如果这个位设为 0，PB0 则不能用来唤醒系统，并且停用中断请求。

6.11. 端口 C 数字输入使能寄存器(*pcdier*), IO 地址= 0x0f

位	初始值	读/写	描述
7 - 4	-	-	保留。
3 - 0	0xF	只写	使能 PC3~PC0 数字输入和唤醒事件。0 / 1：停用 / 启用 当 PC2~PC1 作为 AD 输入时，这些位设 0 可以防止漏电。当选择停用时，这些引脚的唤醒功能也被停用。

注意：详细设置请参考 9.2 部分。

6.12. 端口 A 数据寄存器(*pa*), IO 地址 = 0x10

位	初始值	读/写	描述
7 - 0	0x00	读/写	数据寄存器的端口 A。

6.13. 端口 A 控制寄存器(*pac*), IO 地址 = 0x11

位	初始值	读/写	描述
7 - 0	0x00	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 请注意：PA5 当输入或输出低，当 PA5 设为输出高时，为 OC/OD 输出。

6.14. 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	0x00	读/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

6.15. 端口 B 数据寄存器(*pb*), IO 地址 = 0x13

位	初始值	读/写	描述
7 - 0	0x00	读/写	数据寄存器的端口 B。

6.16. 端口 B 控制寄存器(*pbc*), IO 地址 = 0x14

位	初始值	读/写	描述
7 - 0	0x00	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

6.17. 端口 B 上拉控制寄存器(*pbph*), IO 地址 = 0x15

位	初始值	读/写	描述
7 - 0	0x00	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

6.18. 端口 C 数据寄存器((*pc*), IO 地址= 0x16

位	初始值	读/写	描述
7 - 4	-	-	保留。
3 - 0	0x00	读/写	数据寄存器的端口 C。

6.19. 端口 C 控制寄存器 (*pcc*), IO 地址= 0x17

位	初始值	读/写	描述
7 - 4	-	-	保留.
3 - 0	0x00	读/写	端口 C 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

6.20. 端口 C 上拉控制寄存器(*pcph*), IO 地址= 0x18

位	初始值	读/写	描述
7 - 4	-	-	保留。
3 - 0	0x00	读/写	端口 C 上拉控制寄存器。这些寄存器是用来控制上拉高端口 C 每个相应的引脚。 0/1：停用/启用

6.21. 端口 B 下拉控制寄存器 (*pbpl*), IO 地址= 0x19

位	初始值	读/写	描述
7 - 4	-	-	保留。
3	0	读/写	端口 PB3 下拉寄存器。0/1：停用/启用
2	0	读/写	端口 PB2 下拉寄存器。0/1：停用/启用
1 - 0	-	-	保留。

6.22. 端口 C 下拉控制寄存器(*pcpl*), IO 地址= 0x1a

位	初始值	读/写	描述
7 - 2	-	-	保留。
1	0	读/写	端口 PC1 下拉寄存器。0/1：停用/启用
0	0	读/写	端口 PC0 下拉寄存器。0/1：停用/启用

©Copyright 2023, PADAUK Technology Co. Ltd

6.23. ADC 控制寄存器(*adcc*), IO 地址 = 0x20

位	初始值	读/写	描述
7	0	读/写	启用 ADC 功能。0/1：停用/启用
6	0	读/写	ADC 转换进程控制位： 读到 “1” 表明 ADC 已经准备好，或已转换完成。
5 - 2	0000	读/写	通道选择。 以下 4 位用来选择 AD 转换的输入信号： 0000：PB0/AD0， 0001：PB1/AD1， 0010：PB2/AD2， 0011：PB3/AD3， 0100：PB4/AD4， 0101：PB5/AD5， 0110：PB6/AD6， 0111：PB7/AD7， 1000：PA3/AD8， 1001：PA4/AD9， 1010：PA0/AD10， 1011：PC1/AD11，（ICE 上是 PA1） 1100：PC2/AD12，（ICE 上是 PC1） 1111：（通道 F）Bandgap 参考电压 其他：保留
0 - 1	-	-	保留。（写 0）

6.24. ADC 模式寄存器(*adcm*), IO 地址 = 0x21

位	初始值	读/写	描述
7 - 4	-	-	保留（写 0）。
3 - 1	000	读/写	ADC 时钟源选择： 000：CLK（系统时钟）÷ 1， 001：CLK（系统时钟）÷ 2， 010：CLK（系统时钟）÷ 4， 011：CLK（系统时钟）÷ 8， 100：CLK（系统时钟）÷ 16， 101：CLK（系统时钟）÷ 32， 110：CLK（系统时钟）÷ 64， 111：CLK（系统时钟）÷ 128，
0	-	-	保留。

6.25. ADC 高位结果寄存器(*adcrh*), IO 地址= 0x22

位	初始值	读/写	描述
7 - 0	-	只读	这 8 个只读位是 ADC 转换结果的位[11:4]，寄存器的位 7 是 ADC 转换结果的最高位。

6.26. ADC 低位结果寄存器 (*adcrl*), IO 地址 = 0x23

位	初始值	读/写	描述
7 - 4	-	只读	这 4 个只读位是 ADC 转换结果的位[3:0]。
3 - 0	-	-	保留。

6.27. ADC 调节控制寄存器(*adcrgc*), IO 地址 = 0x24

位	初始值	读/写	描述
7	0	只写	ADC 参考高电压： 0：V_{DD} 1：外部引脚 (PB1)
6 - 0	-	-	保留。

6.28. 杂项寄存器(*misc*), IO 地址 = 0x26

位	初始值	读/写	描述
7 - 6	-	-	保留。（写 0）
5	0	只写	快唤醒功能。快速唤醒功能 EOSC 模式下不支持。 0：正常唤醒。唤醒时间是 3000 个 ILRC 时钟（不适用快速开机）。 1：快速唤醒。唤醒时间为 45 个 ILRC 时钟。
4	0	只写	使能 VDD/2 LCD bias 电压生成器。 0 / 1：停用/ 开启（ICE 不能实时转换） 如果 Code Option 有选择 LCD 输出，但 MISC.4 没有设为 1，则在 IC 上还是无法输出 VDD/2 bias，但仿真器则恒可以，此处两边现象不同。
3	-	-	保留。
2	0	只写	停用 LVR 功能。 0 / 1：开启/ 停用
1 - 0	00	只写	看门狗时钟超时时间设定： 00：8k ILRC 时钟周期 01：16k ILRC 时钟周期 10：64k ILRC 时钟周期 11：256k ILRC 时钟周期

6.29. 比较器控制寄存器(*gpcc*), IO 地址 = 0x2b

位	初始值	读/写	描述
7	0	读/写	启用比较器。0 / 1：停用/启用 当此位被设置为启用，请同时设置相应的模拟输入引脚是数字停用，以防止漏电。
6	-	只读	比较器结果。 0：正输入 < 负输入 1：正输入 > 负输入
5	0	读/写	选择比较器的结果是否由 TM2_CLK 采样输出。 0：比较器的结果没有 TM2_CLK 采样输出 1：比较器的结果是由 TM2_CLK 采样输出
4	0	读/写	选择比较器输出的结果是否反极性。 0：比较器输出的结果没有反极性 1：比较器输出的结果是反极性
3 - 1	000	读/写	选择比较器负输入的来源。 000：PA3 001：PA4 010：内部 1.20 V bandgap 参考电压（不适用于比较器唤醒功能） 011：$V_{internal\ R}$ 100：PB6 101：PB7 11X：保留
0	0	读/写	选择比较器正输入的来源。 0：$V_{internal\ R}$ 1：PA4

6.30. 比较器选择寄存器(*gpcs*), IO 地址 = 0x2c

位	初始值	读/写	描述
7	0	只写	比较器输出启用（到 PA0）。 0 / 1：停用/启用
6	0	只写	比较器唤醒启用。（gpcc.6 发生电平变化时才可唤醒） 0/1：停用/启用
5	0	只写	选择比较器参考电压 $V_{internal\ R}$ 最高的范围。
4	0	只写	选择比较器参考电压 $V_{internal\ R}$ 最低的范围。
3 - 0	0000	只写	选择比较器参考电压 $V_{internal\ R}$。 0000（最低） ~ 1111（最高）

©Copyright 2023, PADAUK Technology Co. Ltd

6.31. Timer2 控制寄存器(*tm2c*), IO 地址 = 0x30

位	初始值	读/写	描述
7 - 4	0000	读/写	Timer2 时钟源选择： 0000：停用 0001：CLK（系统时钟） 0010：IHRC or IHRC *2（由 code option TMx_ source 决定） 0011：EOSC 0100：ILRC 0101：比较器输出（仿真器不支持） 1000：PA0（上升沿） 1001：~PA0（下降沿） 1010：PB0（上升沿） 1011：~PB0（下降沿） 1100：PA4（上升沿） 1101：~PA4（下降沿） 其他：保留 **注意**：在 ICE 模式且 IHRC 被选为 Timer2 定时器时钟，当 ICE 停下时，发送到定时器的时钟是不停止，定时器仍然继续计数。
3 - 2	00	读/写	Timer2 输出选择： 00：停用 01：PB2 10：PA3 11：PB4
1	0	读/写	Timer2 模式选择： 0 / 1：定周期模式 / PWM 模式。
0	0	读/写	启用 Timer2 反极性输出： 0 / 1：停用/启用

6.32. Timer2 计数寄存器(*tm2ct*), IO 地址 = 0x31

位	初始值	读/写	描述
7 - 0	0x00	读/写	Timer2 定时器位[7:0]。

6.33. Timer2 分频寄存器(*tm2s*), IO 地址 = 0x32

位	初始值	读/写	描述
7	0	只写	PWM 分辨率选择。 0：8 位 1：6 位或者 7 位（由 code option TMx_Bit 决定）
6 - 5	00	只写	Timer2 时钟预分频器。 00：÷ 1 01：÷ 4 10：÷ 16 11：÷ 64
4 - 0	00000	只写	Timer2 时钟分频器。

©Copyright 2023, PADAUK Technology Co. Ltd

6.34. Timer2 上限寄存器(*tm2b*), IO 地址 = 0x33

位	初始值	读/写	描述
7 - 0	0x00	只写	Timer2 上限寄存器。

6.35. Timer3 控制寄存器(*tm3c*), IO 地址 = 0x34

位	初始值	读/写	描述
7 - 4	0000	读/写	Timer3 时钟选择。 0000：disable 0001：CLK（系统时钟） 0010：IHRC or IHRC *2（由 code option TMx_ source 决定） 0011：EOSC 0100：ILRC 0101：比较器输出 1000：PA0（上升沿） 1001：~PA0（下降沿） 1010：PB0（上升沿） 1011：~PB0（下降沿） 1100：PA4（上升沿） 1101：~PA4（下降沿） 其他：保留 **注意**：在 ICE 模式且 IHRC 被选为 Timer3 定时器时钟，当 ICE 停下时，发送到定时器的时钟是不停止，定时器仍然继续计数。
3 - 2	00	读/写	Timer3 输出选择。 00：停用 01：PB5 10：PB6 11：PB7
1	0	读/写	Timer3 模式选择。 0：周期模式 1：PWM 模式
0	0	读/写	启用 Timer3 反极性输出。 0/1：停用/启用

6.36. Timer3 计数寄存器(*tm3ct*), IO 地址 = 0x35

位	初始值	读/写	描述
7 - 0	0x00	读/写	Timer3 定时器位[7:0]。

©Copyright 2023, PADAUK Technology Co. Ltd

6.37. Timer3 分频寄存器 (*tm3s*), IO address = 0x36

位	初始值	读/写	描述
7	0	只写	PWM 分辨率选择。 0：8 位 1：6 位或者 7 位（由 code option TMx_Bit 决定）
6 - 5	00	只写	Timer3 时钟预分频器。 00：÷ 1 01：÷ 4 10：÷ 16 11：÷ 64
4 - 0	00000	只写	Timer3 时钟分频器。

6.38. Timer3 上限寄存器(*tm3b*), IO 地址 = 0x37

位	初始值	读/写	描述
7 - 0	0x00	只写	Timer3 上限寄存器。

6.39. PWMG0 控制寄存器(*pwmg0c*), IO 地址 = 0x40

位	初始值	读/写	描述
7	-	-	保留。
6	-	只读	PWMG0 生成器输出状态。
5	0	只写	选择 PWMG0 的输出的结果是否反极性： 0/1：停用/启用
4	0	只写	PWMG0 输出选择。 0：PWMG0 输出 1：PWMG0 XOR PWMG1 或者 PWMG0 OR PWMG1（通过 pwmg0c.0 位来选择）
3 - 1	000	读/写	PWMG0 输出端口选择。 000：输出停用 001：PB5 010：PC2 011：PA0 100：PB4 101：PB6（仅适用于 PB6 不配置成 PWMG1 的输出） 其他：保留
0	0	读/写	PWMG0 输出预选择。 0：PWMG0 XOR PWMG1 1：PWMG0 OR PWMG1

6.40. PWMG 时钟寄存器(*pwmgclk*), IO 地址 = 0x41

位	初始值	读/写	描述
7	0	只写	PWMG 停用/ 启用。 0：PWMG 停用 1：PWMG 启用
6 - 4	000	只写	PWMG 时钟预分频。 000：÷1 001：÷2 010：÷4 011：÷8 100：÷16 101：÷32 110：÷64 111：÷128
3 - 1	-	-	保留。
0	0	只写	PWMG 时钟源选择。 0：系统时钟 1：IHRC 或者 IHRC*2（由 code option PWM_Source 决定）

6.41. PWMG0 占空比高位寄存器(*pwmg0dth*), IO 地址 = 0x42

位	初始值	读/写	描述
7 - 0	-	只写	PWMG0 占空比值。位[10:3]。

6.42. PWMG0 占空比低位寄存器((*pwmg0dtl*), IO 地址 = 0x43

位	初始值	读/写	描述
7 - 5	-	只写	PWMG0 占空比值。位[2:0]。
4 - 0	-	-	保留。

注意: PWMG0 占空比低位寄存器的值必须写在 PWMG0 占空比高位寄存器之前。

6.43. PWMG 计数上限高位寄存器(*pwmgcubh*), IO 地址 = 0x44

位	初始值	读/写	描述
7 - 0	-	只写	PWMG 上限寄存器。位[10:3]。

6.44. PWMG 计数上限低位寄存器(*pwmgcubl*), IO 地址= 0x45

位	初始值	读/写	描述
7 - 6	-	只写	PWMG 上限寄存器。位[2:1]。
5 - 0	-	-	保留。

©Copyright 2023, PADAUK Technology Co. Ltd

6.45. PWMG1 控制寄存器 (*pwmg1c*), IO 地址= 0x46

位	初始值	读/写	描述
7	-	-	保留。
6	-	只读	PWMG1 生成器输出状态。
5	0	读/写	选择 PWMG1 的输出的结果是否反极性： 0/1：停用/启用。
4	0	读/写	PWMG1 输出选择： 0：PWMG1 1：PWMG2
3 - 1	000	读/写	PWMG1 输出端口选择： 000：输出停用 001：PB6 010：PC3 011：PA4 100：PB7 1xx：保留
0	-	读/写	保留。

6.46. PWMG1 占空比高位寄存器 (*pwmg1dth*), IO 地址= 0x48

位	初始值	读/写	描述
7 - 0	-	只写	PWMG1 占空比值。位[10:3]。

6.47. PWMG1 占空比低位寄存器(*pwmg1dtl*), IO 地址= 0x49

位	初始值	读/写	描述
7 - 5	-	只写	PWMG1 占空比值。位[2:0]。
4 - 0	-	-	保留。

注意：PWMG1 占空比低位寄存器的值必须写在 PWMG1 占空比高位寄存器之前。

6.48. PWMG2 控制寄存器(*pwmg2c*), IO 地址 = 0x4C

位	初始值	读/写	描述
7	-	-	保留。
6	-	只读	PWMG2 生成器输出状态。
5	0	读/写	选择 PWMG2 的输出的结果是否反极性： 0/1：停用/启用
4	0	读/写	PWMG2 输出选择: 0：PWMG2 1：PWMG2 ÷2
3 - 1	000	读/写	PWMG2 输出端口选择： 000：输出停用 001：PB3 010：PC0 011：PA3 100：PB2 101：PA5 110：PB5（仅适用于 PB5 不配置成 PWMG0 的输出） 111：保留
0	-	读/写	保留。

6.49. PWMG2 占空比高位寄存器(*pwmg2dth*), IO 地址= 0x4E

位	初始值	读/写	描述
7 - 0	-	只写	PWMG2 占空比值。位[10:3]。

6.50. PWMG2 占空比低位寄存器(*pwmg2dtl*), IO 地址= 0x4F

位	初始值	读/写	描述
7 - 5	-	只写	PWMG2 占空比值。位[2:0]。
4 - 0	-	-	保留。

注意：PWMG2 占空比低位寄存器的值必须写在 PWMG2 占空比高位寄存器之前。

©Copyright 2023, PADAUK Technology Co. Ltd

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
I	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)
M.n	只允许寻址在地址 0~0x7F (0~127) 的位置

©Copyright 2023, PADAUK Technology Co. Ltd

7.1. 数据传输类指令

mov a, I	移动即时数据到累加器。 例如： *mov* a, 0x0f; 结果： a ← 0fh; 受影响标志位： Z：『不变』， C：『不变』， AC：『不变』， OV：『不变』
mov M, a	移动数据由累加器到存储器。 例如：*mov* MEM, a; 结果：MEM ← a 受影响标志位： Z：『不变』， C：『不变』， AC：『不变』， OV：『不变』
mov a, M	移动数据由存储器到累加器。 例如：*mov* a, MEM ; 结果：a ← 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』
mov a, IO	移动数据由 IO 到累加器。 例如：*mov* a, pa ; 结果：a ← 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』
mov IO, a	移动数据由累加器到 IO。 例如：*mov* pb, a; 结果：pb ← a 受影响标志位：Z：『不变』， C：『不变』， AC：『不变』， OV：『不变』

ldtabh index	通过使用 index 作为 OTP 地址把数据的高位存进程序存储器给累加器。该操作需要 2T 指令周期。
	例如：*ldtabh index;*
	结果：a ← {bit 15~8 of OTP [index]};
	受影响标志位：『N』Z 『N』C 『N』AC 『N』OV
	应用范例：
	--- 	
	word ROMptr ; // 在 RAM 里面声明指针
	...
	mov a, la@TableA ; // 指针指向 ROM 里面的 TableA (LSB)
	mov lb@ROMptr, a ; // 保存指针给 RAM(LSB)
	mov a, ha@TableA ; // 指针指向 ROM 里面的 TableA (MSB)
	mov hb@ROMptr, a ; // 保存指针给 RAM(MSB)
	...
	ldtabh ROMptr ; // 存储 TableA MSB 给 ACC (ACC=0X02)
	...
	TableA : dc 0x0234, 0x0042, 0x0024, 0x0018 ;
	--- 	

ldtabl index

通过使用 index 作为 OTP 地址把数据的低位存进程序存储器给累加器。该操作需要 2T 指令周期。
例如： *ldtabl index;*
结果： a ← {bit7~0 of OTP [index]};
受影响标志位：『N』Z 『N』C 『N』AC 『N』OV
应用范例：

```
word    ROMptr ;          // 在 RAM 里面声明指针
...
mov     a, la@TableA ;     // 指针指向 ROM 里面的 TableA (LSB)
mov     lb@ROMptr, a ;     // 保存指针给 RAM (LSB)
mov     a, ha@TableA ;     // 指针指向 ROM 里面的 TableA (MSB)
mov     hb@ROMptr, a ;     // 保存指针给 RAM (MSB)
...
ldtabl  ROMptr ;           // 存储 TableA LSB 给 ACC (ACC=0x34)
...
TableA :  dc  0x0234, 0x0042, 0x0024, 0x0018 ;
```

ldt16 word

将 Timer16 的 16 位计算值复制到 RAM。
例如： *ldt16* word;
结果： word ← 16-bit timer
受影响标志位： Z：『不变』， C：『不变』， AC：『不变』， OV：『不变』
应用范例：

```
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
….
```

stt16 word

将放在 word 的 16 位 RAM 复制到 Timer16。
例如： *stt16* word;
结果： 16-bit timer ← word
受影响标志位： Z：『不变』， C：『不变』， AC：『不变』， OV：『不变』
应用范例：

```
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
…
```

idxm 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： *idxm* a, index;
	结果： a ← [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：
	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 PFS123 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

Idxm index, a	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： *idxm* a, index;
	结果： a ← [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：
	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)，必须为 0
	mov hb@RAMIndex, a ; // 将指针存到 RAM (MSB)
	…
	mov a, 0xA5 ;
	idxm RAMIndex, a ; // 0xA5 存入 RAM 地址为 0x5B

xch M	累加器与 RAM 之间交换数据。 例如： *xch* 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』
popaf	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：popaf; 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』

pushaf	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：*pushaf*;
	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
	pushaf ; // 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
	popaf ; // 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
	reti ;
	--- 	

nmov M, a	将累加器的负逻辑(2 补码)放入 RAM
	例如： mov MEM, a;
	结果： MEM ← ∓a
	受影响标志位：Z：『不变』， C：『不变』， AC：『不变』， OV：『不变』
	应用范例：
	--- 	
	mov a, 0xf5 ; // ACC is 0xf5
	nmov ram9, a; // ram9 is 0x0b, ACC is 0xf5
	--- 	

nmov a, M	将 RAM 的负逻辑(2 补码)放入累加器
	例如： mov a, MEM ;
	结果： a ← ∓MEM; Flag Z is set when ∓MEM is zero.
	受影响标志位：Z：『受影响』， C：『不变』， AC：『不变』， OV：『不变』
	应用范例：
	--- 	
	mov a, 0xf5 ;
	mov ram9, a ; // ram9 is 0xf5
	nmov a, ram9 ; // ram9 is 0xf5, ACC is 0x0b
	--- 	

7.2. 算术运算类指令

add 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： *add* a, 0x0f ; 结果： a ← 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』
add a, M	将 RAM 与累加器相加，然后把结果放入累加器。 例如： *add* a, MEM ; 结果： a ← 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』
add M, a	将 RAM 与累加器相加，然后把结果放入 RAM。 例如： *add* MEM, a; 结果： MEM ← 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』
addc 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： *addc* a, MEM ; 结果： a ← 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』
addc M, a	将 RAM、累加器以及进位相加，然后把结果放入 RAM。 例如： *addc* MEM, a ; 结果： MEM ← 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』
addc a	将累加器与进位相加，然后把结果放入累加器。 例如： *addc* a ; 结果： a ← 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』
addc M	将 RAM 与进位相加，然后把结果放入 RAM。 例如： *addc* MEM ; 结果： MEM ← 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』
nadd a, M	将累加器的负逻辑(2补码)与RAM相加，然后把结果放入累加器。 例如: *nadd* a, MEM ; 结果: a ← ∓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』
nadd M, a	将RAM的负逻辑(2补码)与累加器相加，然后把结果放入RAM。 例如: *nadd* MEM, a ; 结果: MEM ← ∓MEM + a 受影响的标志位: Z：『受影响』， C：『受影响』， AC：『受影响』， OV：『受影响』
sub a, I	累加器减立即数据，然后把结果放入累加器。 例如： *sub* a, 0x0f; 结果： a ← a - 0fh (a + [2's 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』
sub a, M	累加器减 RAM，然后把结果放入累加器。 例如： *sub* a, MEM ; 结果： a ← a - MEM (a + [2's 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』

sub M, a	RAM 减累加器，然后把结果放入 RAM。 例如： *sub* MEM, a; 结果： MEM ← MEM - a (MEM + [2's 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』
subc a, M	累加器减 RAM，再减进位，然后把结果放入累加器。 例如： *subc* a, MEM; 结果： a ← 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』
subc M, a	RAM 减累加器，再减进位，然后把结果放入 RAM。 例如： *subc* MEM, a ; 结果： MEM ← 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』
subc a	累加器减进位，然后把结果放入累加器。 例如： *subc* a; 结果： a ← 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』
subc M	RAM 减进位，然后把结果放入 RAM。 例如： *subc* MEM; 结果： MEM ← 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』
inc M	RAM 加 1。 例如： *inc* MEM ; 结果： MEM ← 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』
dec M	RAM 减 1。 例如： dec MEM; 结果： MEM ← 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』
clear M	清除 RAM 为 0。 例如： *clear* MEM ; 结果： MEM ← 0 受影响的标志位： Z：『不变』， C：『不变』， AC：『不变』， OV：『不变』

7.3. 移位运算类指令

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

7.4. 逻辑运算类指令

and a, I	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：*and* a, 0x0f ; 结果：a ← a & 0fh 受影响的标志位： Z：『受影响』， C：『不变』， AC：『不变』， OV：『不变』
and 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：*and* a, RAM10 ; 结果：a ← a & RAM10 受影响的标志位： Z：『受影响』， C：『不变』， AC：『不变』， OV：『不变』
and M, a	累加器和 RAM 执行逻辑 AND，然后把结果保存到 RAM。 例如： *and* MEM, a ; 结果： MEM ← a & MEM 受影响的标志位： Z：『受影响』， C：『不变』， AC：『不变』， OV：『不变』
or a, I	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： *or* a, 0x0f ; 结果： a ← a | 0fh 受影响的标志位： Z：『受影响』， C：『不变』， AC：『不变』， OV：『不变』
or 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：*or* a, MEM ; 结果：a ← a | MEM 受影响的标志位： Z：『受影响』， C：『不变』， AC：『不变』， OV：『不变』
or M, a	累加器和 RAM 执行逻辑 OR，然后把结果保存到 RAM。 例如： *or* MEM, a ; 结果： MEM ← a | MEM 受影响的标志位： Z：『受影响』， C：『不变』， AC：『不变』， OV：『不变』
xor a, I	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： *xor* a, 0x0f ; 结果： a ← a ^ 0fh 受影响的标志位： Z：『受影响』， C：『不变』， AC：『不变』， OV：『不变』

xor 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： *xor pa, a ;*
	结果： pa ← a ^ 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』

xor 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： *xor* a, MEM ; 结果： a ← a ^ RAM10 受影响的标志位： Z：『受影响』， C：『不变』， AC：『不变』， OV：『不变』
xor M, a	累加器和 RAM 执行逻辑 XOR，然后把结果保存到 RAM。 例如： *xor* MEM, a ; 结果： MEM ← a ^ MEM 受影响的标志位： Z：『受影响』， C：『不变』， AC：『不变』， OV：『不变』

©Copyright 2023, PADAUK Technology Co. Ltd

not a	累加器执行 1 补码运算，结果放在累加器。
	例如： *not* a ;
	结果： a ← ～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：
	--- 	
	mov a, 0x38 ; // ACC=0X38
	not a ; // ACC=0XC7
	--- 	

not M	RAM 执行 1 补码运算，结果放在 RAM。
	例如： *not* MEM ;
	结果： MEM ← ～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：
	--- 	
	mov a, 0x38 ;
	mov mem, a ; // mem = 0x38
	not mem ; // mem = 0xC7
	--- 	

neg a	累加器执行 2 补码运算，结果放在累加器。
	例如： *neg* a;
	结果： a ←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：
	--- 	
	mov a, 0x38 ; // ACC=0X38
	neg a ; // ACC=0XC8
	--- 	

neg *M*	RAM 执行 2 补码运算，结果放在 RAM。
	例如： neg MEM;
	结果： MEM ← 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：
	--- 	
	mov a, 0x38 ;
	mov mem, a ; // mem = 0x38
	not mem ; // mem = 0xC8
	--- 	

comp a, M	比较累加器和 RAM 的内容。
	例如： *comp a, MEM;*
	结果： 等效于(a - MEM)，并改变标志位 Flag。
	受影响的标志位： Z：『受影响』， C：『受影响』， AC：『受影响』， OV：『受影响』
	应用范例：
	--- 	
	mov a, 0x38 ;
	mov mem, a ;
	comp a, mem ; // Z 标志被设为 1
	mov a, 0x42 ;
	mov mem, a ;
	mov a, 0x38 ;
	comp a, mem ; // C 标志被设为 1
	--- 	

comp M, a	比较累加器和 RAM 的内容。 例如： *comp MEM, a;* 结果： 等效于(MEM - a)，并改变标志位 Flag。 受影响的标志位： Z：『受影响』， C：『受影响』， AC：『受影响』， OV：『受影响』

7.5. 位运算类指令

set0 IO.n	IO 口的位 N 拉低电位。 例如： *set0* 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』
set1 IO.n	IO 口的位 N 拉高电位。 例如：*set1* pb.5 ; 结果： PB5=1 受影响的标志位： Z：『不变』， C：『不变』， AC：『不变』， OV：『不变』

swapc IO.n	IO 口的位 N 与 C 位互换。
	例如： swapc IO.0;
	结果： C ← IO.0 , IO.0 ← C
	当 IO.0 是输出端口，进位 C 数值给 IO.0;
	当 IO.0 是输入端口，IO.0 数值给进位 C;
	受影响的标志位： 『不变』Z 『受影响』C 『不变』AC 『不变』OV
	应用范例 1（连续输出）：
	--- 	
	...
	set1 pac.0 ; // 设置 PA.0 作为输出
	...
	set0 flag.1 ; // C=0
	swapc pa.0 ; // 送 C 给 PA.0 （位操作），PA.0=0
	set1 flag.1 ; // C=1
	swapc pa.0 ; // 送 C 给 PA.0 （位操作），PA.0=1
	...
	--- 	
	应用范例 2 （连续输入） ：
	--- 	
	...
	set0 pac.0 ; // 设置 PA.0 作为输入
	...
	swapc pa.0 ; // 读 PA.0 的值给 C（位操作）
	src a ; // 把 C 移位给 ACC 的位 7
	swapc pa.0 ; // 读 PA.0 的值给 C（位操作）
	src a ; // 把新进 C 移位给 ACC 的位 7，上一个 PA.0 的值给 ACC 的位 6
	...
	--- 	

set0 M.n	RAM 的位 N 设为 0。 例如：*set0* 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』
set1 M.n	RAM 的位 N 设为 1。 例如： *set1* 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

ceqsn a, I	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 ← a - I)相同。 例如： *ceqsn* a, 0x55 ; *inc* MEM ; *goto* 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』
ceqsn 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 ← a - M)相同。 例如：*ceqsn* 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』
cneqsn a, M	比较累加器和 RAM 的值，如果不相等就跳到下一条指令. 标志改变与(a ← a - M)相同。 例如： *cneqsn* *a, MEM;* 结果： 如果 a≠MEM, 跳到下一条指令 受影响的标志位： Z：『受影响』， C：『受影响』， AC：『受影响』， OV：『受影响』
cneqsn a, I	比较累加器和立即数的值，如果不相等就跳到下一条指令.标志改变与(a ← a - I) 。 例如： *cneqsn* *a,0x55 ;* *inc* *MEM ;* *goto* *error ;* 结果：如果 a≠0x55, 然后 “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』
t0sn IO.n	如果 IO 的指定位是 0，跳过下一个指令。 例如：*t0sn* 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』
t1sn IO.n	如果 IO 的指定位是 1，跳过下一个指令。 例如：*t1sn* 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』
t0sn M.n	如果 RAM 的指定位是 0，跳过下一个指令。 例如： *t0sn* 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』
t1sn M.n	如果 RAM 的指定位是 1，跳过下一个指令。 例如： *t1sn* MEM.5 ; 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』
izsn a	累加器加 1，若累加器新值是 0，跳过下一个指令。 例如： *izsn* a; 结果： a ← 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』

©Copyright 2023, PADAUK Technology Co. Ltd

dzsn a	累加器减 1，若累加器新值是 0，跳过下一个指令。 例如： *dzsn* a; 结果： a ← 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』
izsn M	RAM 加 1，若 RAM 新值是 0，跳过下一个指令。 例如： *izsn* MEM; 结果： MEM ← 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』
dzsn M	RAM 减 1，若 RAM 新值是 0，跳过下一个指令。 例如： *dzsn* MEM; 结果： MEM ← 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

call label	函数调用，地址可以是全部空间的任一地址。 例如： *call* function1; 结果： [sp] ← pc + 1 pc ← function1 sp ← sp + 2 受影响的标志位： Z：『不变』， C：『不变』， AC：『不变』， OV：『不变』
goto label	转到指定的地址，地址可以是全部空间的任一地址。 例如： *goto* error; 结果： 跳到 error 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』
ret I	将立即数据复制到累加器，然后返回。 例如： *ret* 0x55; 结果： A ← 55h ret ; 受影响的标志位： Z：『不变』， C：『不变』， AC：『不变』， OV：『不变』
ret	从函数调用中返回原程序。 例如： *ret;* 结果： sp ← sp - 2 pc ← [sp] 受影响的标志位： Z：『不变』， C：『不变』， AC：『不变』， OV：『不变』
reti	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： *reti*; 受影响的标志位： Z：『不变』， C：『不变』， AC：『不变』， OV：『不变』
nop	没任何动作。 例如： *nop*; 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』

©Copyright 2023, PADAUK Technology Co. Ltd

engint	允许全部中断。 例如：*engint*; 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』
stopsys	系统停止。 例如：*stopsys*; 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』
stopexe	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： *stopexe*; 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』
reset	复位整个单片机，其运行将与硬件复位相同。 例如： *reset*; 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』
wdreset	复位看门狗。 例如： *wdreset* ; 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』

pcadd a	目前的程序计数器加累加器是下一个程序计数器。
	例如： *pcadd a*;
	结果： pc ← 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：
	--- 	
	…
	mov a, 0x02 ;
	pcadd a ; // PC <- PC+2
	goto err1 ;
	goto correct ; // 开始跳
	goto err2 ;
	goto err3 ;
	…
	correct: // 跳到这里
	…
	--- 	

disgint	禁止全部中断。 例如：*disgint* ; 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』

7.8. 指令执行周期综述

2 个周期		*goto, call, idxm, pcadd, ret, reti, ldtabl, ldtabh*
2 个周期	条件满足时	*ceqsn, cneqsn,t0sn, t1sn, dzsn, izsn*
1 个周期	条件不满足时	
1 个周期		其他

7.9. 指令影响标志综述

Instruction	Z	C	AC	OV	Instruction	Z	C	AC	OV	Instruction	Z	C	AC	OV
mov a, I	-	-	-	-	*mov* M, a	-	-	-	-	*mov* a, M	Y	-	-	-
mov a, IO	Y	-	-	-	*mov* IO, a	-	-	-	-	*ldt16* word	-	-	-	-
stt16 word	-	-	-	-	*idxm* a, index	-	-	-	-	*idxm* index, a	-	-	-	-
xch M	-	-	-	-	*pushaf*	-	-	-	-	*popaf*	Y	Y	Y	Y
add a, I	Y	Y	Y	Y	*add* a, M	Y	Y	Y	Y	*add* M, a	Y	Y	Y	Y
addc a, M	Y	Y	Y	Y	*addc* M, a	Y	Y	Y	Y	*addc* a	Y	Y	Y	Y
addc M	Y	Y	Y	Y	*nadd a, M*	Y	Y	Y	Y	*nadd M, a*	Y	Y	Y	Y
sub a, I	Y	Y	Y	Y	*sub* a, M	Y	Y	Y	Y	*sub* M, a	Y	Y	Y	Y
subc a, M	Y	Y	Y	Y	*subc* M, a	Y	Y	Y	Y	*subc* a	Y	Y	Y	Y
subc M	Y	Y	Y	Y	*inc* M	Y	Y	Y	Y	*dec* M	Y	Y	Y	Y
clear M	-	-	-	-	*sr* a	-	Y	-	-	*src* a	-	Y	-	-
sr M	-	Y	-		*src* M	-	Y	-	-	*sl* a	-	Y	-	-
slc a	-	Y	-	-	*sl* M	-	Y	-	-	*slc* M	-	Y	-	-
swap a	-	-	-	-	*and* a, I	Y	-	-	-	*and* a, M	Y	-	-	-
and M, a	Y	-		-	*or* a, I	Y	-	-	-	*or* a, M	Y	-	-	-
or M, a	Y	-	-	-	*xor* a, I	Y	-	-	-	*xor* IO, a	-	-	-	-
xor a, M	Y	-	-	-	*xor* M, a	Y	-	-	-	*not* a	Y	-	-	-
not M	Y	-	-	-	*neg* a	Y	-	-	-	*neg* M	Y	-	-	-
comp a, M	Y	Y	Y	Y	*comp M, a*	Y	Y	Y	Y	*set0* IO.n	-	-		
set1 IO.n	-	-	-	-	*set0* M.n	-	-	-	-	*set1* M.n	-	-	-	-
swapc IO.n	-	Y	-	-	*ceqsn* a, I	Y	Y	Y	Y	*ceqsn* a, M	Y	Y	Y	Y
cneqsn a,M	Y	Y	Y	Y	*cneqsn a, I*	Y	Y	Y	Y	*t0sn* IO.n	-	-	-	-
t1sn IO.n	-	-	-	-	*t0sn* M.n	-	-	-	-	*t1sn* M.n	-	-	-	-
izsn a	Y	Y	Y	Y	*dzsn* a	Y	Y	Y	Y	*izsn* M	Y	Y	Y	Y
dzsn M	Y	Y	Y	Y	*call* label	-	-	-	-	*goto* label	-	-	-	-
ret I	-	-	-	-	*ret*	-	-	-	-	*reti*	-	-	-	-
nop	-	-	-	-	*pcadd* a	-	-	-	-	*engint*	-	-	-	-
disgint	-	-	-	-	*stopsys*	-	-	-	-	*stopexe*	-	-	-	-
reset	-	-	-	-	*wdreset*	-	-	-	-	*ldtabl* index	-	-	-	-
ldtabh index	-	-	-	-	*nmov* M, a	-	-	-	-	*nmov* a, M	Y	-	-	-

©Copyright 2023, PADAUK Technology Co. Ltd

7.10. 位定义

RAM 的位定义仅适用于地址 0x00 到 0x7F。

8. 程序选项(Code Options)

配置	选项	描述
Security	Enable	MTP 内容加密，程序不允许被读取
	Disable	MTP 内容不加密，程序可以被读取
LVR	4.0V	选择 LVR = 4.0V
	3.5V	选择 LVR = 3.5V
	3.0V	选择 LVR = 3.0V
	2.7V	选择 LVR = 2.7V
	2.5V	选择 LVR = 2.5V
	2.2V	选择 LVR = 2.2V
	2.0V	选择 LVR = 2.0V
	1.8V	选择 LVR = 1.8V
Boot-up_Time	Slow	请参考在 4.1 部分的 t_{WUP} 和 t_{SBP}
	Fast	请参考在 4.1 部分的 t_{WUP} 和 t_{SBP}
Interrupt Src0	PA.0	INTEN/ INTRQ.Bit0 源自 PA.0 中断
	PB.5	INTEN/ INTRQ.Bit0 源自 PB.5 中断
Interrupt Src1	PB.0	INTEN/ INTRQ.Bit1 源自 PB.0 中断
	PA.4	INTEN/ INTRQ.Bit1 源自 PA.4 中断
PB4_PB5_Drive	Normal	PB4 & PB5 驱动/ 灌电流(正常)
	Strong	PB4 & PB5 驱动/ 灌电流（强）（仿真器不支持）
Comparator Edge	All_Edge	比较器在上升沿/下降沿都会触发中断
	Rising_Edge	比较器在上升沿会触发中断
	Falling_Edge	比较器在下降沿会触发中断
GPC_PWM	Disable	比较器不控制全部的 PWM 输出
	Enable	比较器会控制全部的 PWM 输出（仿真器不支持）
PWM_Source	16MHZ	当 Pwmgclk.0= 1, PWMG 时钟源 = IHRC = 16MHZ
	32MHZ	当 Pwmgclk.0= 1, PWMG 时钟源 = IHRC*2 = 32MHZ （ICE 不支持）
TMx_Source	16MHZ	当 tm2c[7:4]= 0010, TM2 时钟源 = IHRC = 16MHZ 当 tm3c[7:4]= 0010, TM3 时钟源 = IHRC = 16MHZ
	32MHZ	当 tm2c[7:4]= 0010, TM2 时钟源 = IHRC*2 = 32MHZ 当 tm3c[7:4]= 0010, TM3 时钟源 = IHRC*2 = 32MHZ （仿真器不支持）

©Copyright 2023, PADAUK Technology Co. Ltd

配置	选项	描述
TMx_Bit	6 Bit	当 tm2s.7=1, TM2 PWM 精度是 6 位 当 tm3s.7=1, TM3 PWM 精度是 6 位
	7 Bit	当 tm2s.7=1, TM2 PWM 精度是 7 位 当 tm3s.7=1, TM3 PWM 精度是 7 位 （仿真器不支持）

9. 特别注意事项

此章节提醒使用者在使用 PFS123 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

- 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。
- 内部上拉电阻值也将随着电压、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

- 将 IO 设为输入。
- 用 PXDIER 寄存器，将对应的位设为 1。
- 为了防止 PA 中那些没有用到的 IO 口漏电，PADIER[1:2]需要常设为 0。
- PFS123 的 PCDIER 寄存器功能在 ICE 上有所不同。详细请参考 9.2 部分。

(3) PA5 作为输出

- PA5 只能做 Open Drain 输出，输出高时需要外加上拉电阻。

(4) PA5 作为 PRSTB 输入

- 设定 PA5 为输入。
- 设定 CLKMD.0=1，使 PA5 为外部 PRSTB 输入脚位。

(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 欧电阻。
- 应尽量避免使用 PA5 作为输入。

(6) PA7 和 PA6 作为外部晶体振荡器。

- PA7 和 PA6 设定为输入。
- PA7 和 PA6 内部上拉电阻设为关闭。
- 用 PADIER 寄存器将 PA6 和 PA7 设为模拟输入。
- EOSCR 寄存器位[6:5]选择对应的晶体振荡器频率：
  - 01 ： 低频，例如：32KHz
  - 10 ： 中频，例如：455KHz、1MHz
  - 11 ： 高频，例如：4MHz
- 设置 EOSCR.7 =1 启用晶体振荡器。
- 从 IHRC 或 ILRC 切换到 EOSC，要先确认 EOSC 已经稳定振荡。

注意：请务必仔细阅读 PMC-APN013 之内容，并据此合理使用晶体振荡器。如因用户的晶体振荡器的质量不佳、使用条件不合理、PCB 清洁剂残留漏电、或是 PCB 板布局不合理等等用户原因，造成的慢起振或不起振情况，我司不对此负责。

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。一般步骤如下：

```
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。

(3) 有两组 IO 口外部中断源，每组由程序选项(code option)中的 Interrupt Src0 和 Interrupt Src1 决定对应的中断引脚。请根据寄存器 ***inten / intrq / integs*** 来选择 IO 引脚。

9.1.3. 系统时钟切换

利用 CLKMD 寄存器可切换系统时钟源。但必须注意，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，然后再透过 CLKMD 寄存器关闭 A 时钟源振荡器。

◆ 例：系统时钟从 ILRC 切换到 IHRC/2

```
CLKMD   =   0x36;          // 切到 IHRC，但 ILRC 不要 disable
CLKMD.2 =   0;             // 此时才可关闭 ILRC
```

◆ 错误的写法：ILRC 切换到 IHRC，同时关闭 ILRC

```
CLKMD   =   0x50;          // MCU 会死机
```

9.1.4. 看门狗

看门狗默认为开，但程序执行 ADJUST_IC 十，会将看门狗关闭，若要使用看门狗，需重新配置打开。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。

9.1.6. IHRC

(1) IHRC 的校正操作是于使用 writer 烧录时进行的。

(2) 因为 IC 的塑封材料（不论是封装用的或 COB 用的黑胶）的特性，是会对 IHRC 的频率有一定影响。所以如果用户是在 IC 盖上塑封材料前，就对 IC 进行烧录，及后再封上盖上塑封材料的，则可能造成 IHRC 的特性偏移超出规格的情况。正常情况是频率会变慢一些。

(3) 此种情况通常发生在用户是使用 COB 封装，或者是委托我司进行晶圆代烧 (QTP)时。此情况下我司将不对频率的超出规格的情况负责。

(4) 用户可按自身经验进行一些补偿性调整，例如把 IHRC 的目标频率调高 0.5%-1%左右，令封装后 IC 的 IHRC 频率更接近目标值。

9.1.7. LVR

LVR 水平的选择在程序编译时进行。使用者必须结合单片机工作频率和电源电压来选择 LVR，才能让单片机稳定工作。

下面是工作频率、电源电压和 LVR 水平设定的建议：

SYSCLK	VDD	LVR
2MHz	≧ 2.2V	≧ 2.2V
4MHz	≧ 2.5V	≧ 2.5V
8MHz	≧ 3.5V	≧ 3.5V

表 7：LVR 设置参考

(1) 只有当 IC 正常起动后，设定 LVR（1.8V ~ 4.0V）才会有效。

(2) 可以设定寄存器 MISC.2 为 1 将 LVR 关闭，但此时应确保 V_{DD} 在最低工作电压以上，否则 IC 可能工作不正常。

在省电模式 stopexe 和掉电模式 stopsys 下，LVR 功能无效。

9.1.8. 烧录方法

请使用 5S-P-003 进行烧录。3S-P-002 或之前的烧录器皆不支持烧录 PFS123。

Jumper 连接：可依照烧录器软件上的说明，连接 jumper 即可。

请用户依据实际情况选择以下两种烧录模式。

普通烧录模式

适用范围：

- 单独封装 IC，并在烧录器的 IC 插座或连接分选机烧录。
- 合封（MCP）IC，但与 PFS123 合封的 IC 及元件不会被以下电压破坏，也不会钳制以下电压的产生。

普通烧录模式电压条件：

(1) VDD 等于 7.5V，而最大供给电流最高可达约 20mA。

(2) PA5 等于 8.0V。

(3) 其他烧录引脚（GND 除外）等于 VDD。

重要提示：

- **如在 handler 上对 IC 进行烧录，请务必按照 APN004 及 APN011 的指示进行。**
- **为对抗烧录时的杂讯干扰，请于烧录时在分选机连接 IC 连接器一端的 VDD 和 GND 之间连接 0.01uF 电容。但切忌连接标值 0.01uF 以上的电容，以免影响普通烧录模式的运行。**

限压烧录模式

适用范围：

- 在板烧录（On-board Writing），但其周边电路及组件不会被以下电压破坏，也不会钳制以下电压的产生。请参考在板烧录章节的详细说明。
- 合封（MCP）IC，但与 PFS123 合封的 IC 及元件不会被以下电压破坏，也不会钳制以下电压的产生。

限压烧录模式电压条件：

(1) VDD 等于 5.0V，而最大供给电流最高可达约 20mA。

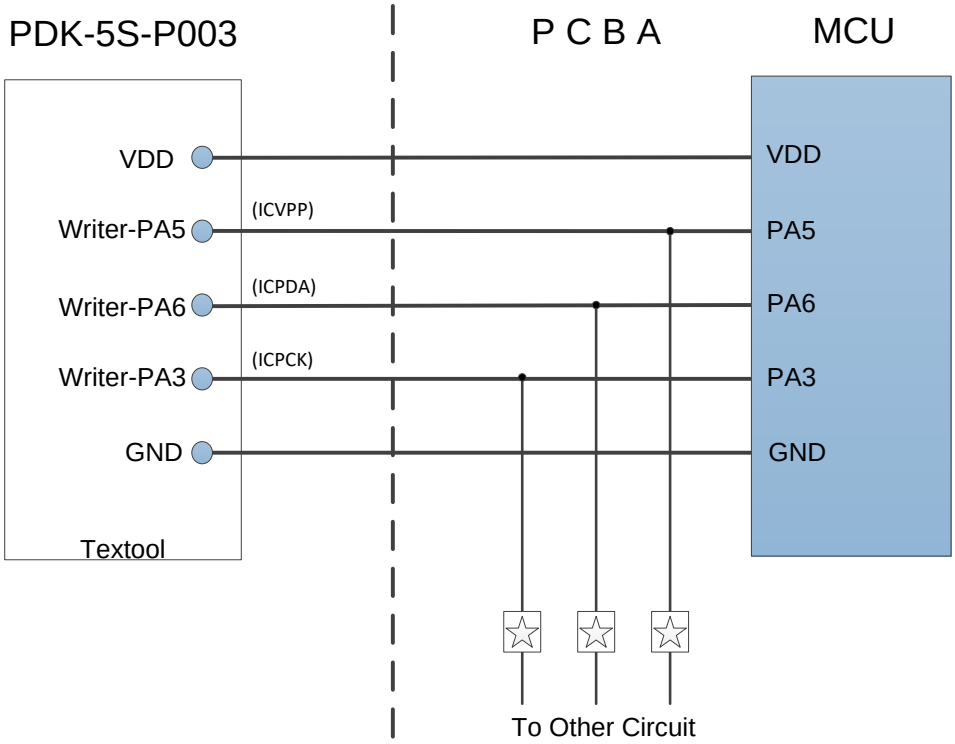
(2) PA5 等于 8.0V。

(3) 其他烧录引脚（GND 除外）等于 VDD。

若要启动限压烧录模式，请于烧录器界面上选择“MTP On-board VDD limitation” 或 “On-board Program”（请参考烧录器 5S-P-003 的用户手册）。

在板烧录（On-Board Writing）

PFS123 可以支持在板烧录。所谓在板烧录，是指 IC 及其他周边电路及组件，皆已经焊接到 PCB 上，并对 IC 进行烧录的情况。在板烧录需要使用 5S-P-003 上五根引线：ICPCK、ICPDA、VDD、GND 和 ICVPP，用于与 IC 上的 PA3、PA6、VDD、GND 和 PA5 对应相连。



上图为 PFS123 在板烧录时接线示意图。图中的 ☆ 为电阻或电容，用于隔离烧录引线和其他电路。电阻应≧10KΩ，电容应≦220pF。

注意：

- 一般来说，在板烧录应使用限压烧录模式。请参考在限压烧录模式的详细说明。
- PCB 上的 VDD 与 GND 之间**不可**连接有 5.0V 或以下的稳压二极管或其他钳制 5.0V 产生的电路或元件。
- PCB 上的 VDD 与 GND 之间**不可**连接有标值 500uF 或以上的电容器。
- 一般来说，用于烧录讯号的 PA3，PA5 及 PA6 引脚，**不能**作为强输出脚。

9.2. 使用 ICE

(1) 5S-I-S01/2(B) 支持 PFS123 MCU 仿真，如下是使用 5S-I-S01/2 仿真 PFS123 的注意事项：

- 5S-I-S01/2(B) 不支持指令 NMOV/SWAP/NADD/COMP。
- 5S-I-S01/2(B) 不支持 SYSCLK=ILRC/16。
- 5S-I-S01/2(B) 不支持动态设定 ***misc.4***（仅适用 0 或者 1）。
- 5S-I-S01/2(B) 不支持 ***Tm2.gpcrs***/***Tm3.gpcrs***。
- 5S-I-S01/2(B) 不支持 ADCRGC [3:2]的 ADC channel F 的 bandgap 参考电压。仅存在稳定的 1.2V。
- 5S-I-S01/2(B) 不支持程序选项：PB4_PB5_Drive, GPC_PWM, TMx_source, PWM_Source 和 TMx_bit。
- 5S-I-S01/2(B) 不支持 SuLED PWM 产生器及其他有关的寄存器。
- 5S-I-S01/2(B) 不支持 PB0 的 VDD/2 功能。
- 5S-I-S01/2(B) 在使用 adcc 时，PC2 和 PC1 有不同的设置。
- S-I-S01/2(B) 只有 240 bytes 数据存储器。
- 5S-I-S01/2(B) PCDIER 寄存器不同于实际 IC。5S-I-S01/2(B)的 PCDIER[0]用于设定 PC0~PC3 的数字输入，PCDIER[1]用于设定 PC4~ PC7 的数字输入。建议不用设置 PCDIER。
- 当 PB1 在 ADCRGC 里使用时，PA1 需悬空。
- 当 GPCC 输出时，PA3 会受到影响。
- 不支持下拉的仿真。
- 仿真 PWM 波形时，建议用户在程序运行期间查看波形，当仿真器暂停或单步运行时波形可能会与实际不符。
- 用 5S-I-S01/2(B)仿真时，在 Timer2/Timer3 定周期模式下，改变 tm2ct/tm3ct 的值会影响占空比，对于实际 IC 则不会。
- 用 5S-I-S01/2(B)仿真时，当快速唤醒被使能，看门狗溢出复位时间会变得很短。实际 IC 没有影响。
- 5S-I-S01/2(B)仿真器的 ILRC 频率与实际 IC 不同，且未经校准，其频率范围大约在 34K~38KHz。
- 快速唤醒时间和使用 5S-I-S01/2(B)仿真不同 (5S-I-S01/2(B): 128 SysClk，PFS123: 45 ILRC)。
- 看门狗溢出时间和使用 5S-I-S01/2(B)仿真不同，如下：

WDT 溢出时间	5S-I-S01/2(B)	PFS123
misc[1:0]=00	2048 * T_{ILRC}	8192 * T_{ILRC}
misc[1:0]=01	4096 * T_{ILRC}	16384 * T_{ILRC}
misc[1:0]=10	16384 * T_{ILRC}	65536 * T_{ILRC}
misc[1:0]=11	256 * T_{ILRC}	262144 * T_{ILRC}

©Copyright 2023, PADAUK Technology Co. Ltd