PUD310

8 位 OTP 型 PD 控制器

数据手册

第 0.04 版

2026 年 6 月 25 日

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。

目 录

修订历史

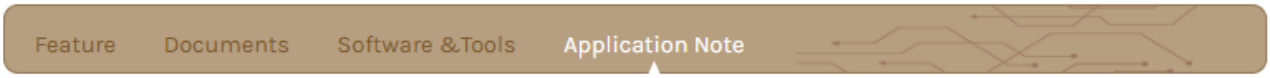
修订	日期	描述
0.02	2025/11/28	1. 第 1.2 章节: 特定的 USB CC1 和 CC2 引脚可支持 USB PD 规范应用，删除「可承受最高 24V 的高电压」的描述 2. 第 4.1 章节: 比较器偏置电压 删除「±20」数值 3. 第 4.15~4.18 章节: 删除测试相关的文字 4. 第 4.15 章节: 删除 V_{ODP5} Max 与 Min 的规格 5. 第 4.15 章节: 删除 V_{ODM5} Max 与 Min 的规格 6. 第 4.17 章节: 删除 V_{OLGATE} Max 与 Min 的规格 7. 第 4.17 章节: 删除 V_{OHGATE} Max 与 Min 的规格 8. 第 4.17 章节: 删除 I_{GATE} Max 的规格，增加 I_{GATE} Typ 的规格 9. 移除 5.12.3.9 内部与外部环回模式章节 10. 移除 5.12.4.5 传输 BIST 信号章节 11. 移除 5.12.4.6 内部和外部环回测试章节 12. 移除 5.13.2 方框图章节 13. 第 6.28.章节 PD PHY TXRX 寄存器(pd_txrx), IO 地址= 0x32: (1) 移除 Bit7 SEND_BIST definition 14. 第 6.29.章节 PD PHY 控制寄存器 (pd_ctrl), IO 地址= 0x33: (1) 移除 Bit7 RX_REQ definition (2) 移除 Bit6 RX_ACK definition (3) 移除 Bit5 LOOP_SEL definition (4) 移除 Bit4 TEST_SEL definition
0.03	2026/04/23	1. 移除 PA1 与 PA2 相关信息 2. 更新代码选项描述
0.04	2026/06/25	1. 第 6.4 章节: 删除「比率」及「源」

使用警告

在使用 IC 前，请务必认真阅读 PUD310 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：

https://www.padauk.com.tw/en/product/search_list.aspx?kw=PUD



Content	Description	Download (CN)	Download (EN)
APN002	Over voltage protection		
APN003	Over voltage protection		
APN004	Semi-Automatic writing handler		
APN007	Setting up LVR level		
APN011	Semi-Automatic writing Handler improve writing stability		
APN019	E-PAD PCB layout guideline		

1. 功能

1.1. 特性

- 通用系列
- USB 功率传输（PD）受电端应用
- USB 电池充电规范 1.2 版(BC1.2)检测
- 本产品可显示多种呼吸灯模式
- 工作温度范围：-40°C ~ 85°C

1.2. 快速充电规范应用功能

- 特定的 USB DP 和 DM 引脚可支持不同的充电规格应用：
  (1) 可选驱动电压电平：5.0V、3.3V、0.6V、0V
  (2) 可选电压比较器：3.0V、2.4V、1.2V、0.35V
  (3) 可承受最高 13V 的高电压
  (4) 测试模式下可配置为 OTP 下载电缆引脚
- 特定的 USB CC1 和 CC2 引脚可支持 USB PD 规范应用：
  (1) 一个 USB PD PHY 收发器
  (2) 一个用于 USB Type-C 5V/1.5A 检测的 0.66V 电压比较器
  (3) 一个用于 USB Type-C 5V/3A 检测的 1.23V 电压比较器
  (4) 精确的 1.125V CC1/CC2 驱动电压
- 一个数字 PD PHY 控制器可支持 USB PD 规范应用
- 一个特定的 GATE 引脚可连接到外部功率 MOSFET，以打开/关闭电源线：
  (1) 漏极开路 IO（V_{OL} = 0.5V 时的灌电流为 20mA）
  (2) 可承受最高 21V 的高电压
- 一个用于充电规格应用的 VBUS (VDD) 7V 电压比较器
- 工作电压范围：4V ~ 21V VBUS（VDD）

1.3. 系统特性

- 1.5KW OTP 程序存储器
- 96 Byte 数据存储器
- 一个硬件 16-位计数器
- 一个 8 位定时器（可作为 PWM 生成器，PWM 分辨率可以为 6 位、7 位或 8 位）
- 5 个 IO 引脚，带可选的上拉电阻和下拉电阻（PA0,PA3,PA5,PB0,PB1）
- 提供三组不同的 IO 驱动能力以满足不同的应用需求
  (1) PA4, PA6 驱动电流/灌电流= 0.33mA / 0.37mA
  (2) 其他 IO（除 PA5 外）驱动电流/灌电流= 16mA / 18mA
- 每个 IO 引脚（PA4 和 PA6 除外）均可配置为启用唤醒功能
- 时钟源：IHRC, ILRC & NILRC
- 对所有带有唤醒功能的 IO，都支持两种可选择的唤醒速度：正常唤醒和快速唤醒
- 16 段 LVR 复位电压设定：
  4.5V, 4.0V, 3.75V, 3.5V, 3.3V, 3.15V, 3.0V, 2.7V, 2.5V, 2.4V, 2.3V, 2.2V, 2.1V, 2.0V, 1.9V, 及 1.8V
- 一个外部中断引脚：PA0
- Bandgap 电路提供 1.2V 参考电压

1.4. CPU 特点

- 单一处理单元工作模式
- 提供 95 个有效指令
- 大部分都是 1T（单周期）指令
- 可程序设定的堆栈指针和堆栈深度
- 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

1.5. 订购/封装信息

- PUD310-U06A: SOT23-6 (60mil)
- PUD310-2N08A: DFN (2*2mm)
- PUD310-EY10A: ESSOP10 (150mil)
- PUD310-2N06A: DFN (2*2mm)
- PUD310-S08A: SOP8 (150mil)
- PUD310-4N12A: DFN (3*3mm)

- 有关封装尺寸的信息，请参阅官方网站文件：“封装信息”

2. 系统概述和方框图

PUD310 系列是一款 IO 类型，完全静态以 OTP 为程序基础的 CMOS 8-bit 微处理器。它运用 RISC 的架构并且所有的指令架构的执行周期都是一个指令周期，只有少部分指令需要两个指令周期。

PUD310 内置 1.5KW OTP 程序存储器以及 96 字节数据存储器，PUD310 还提供两个硬件定时器：一个 16 位计数器，一个 8 位 PWM 计数器（参见图 2.1）。

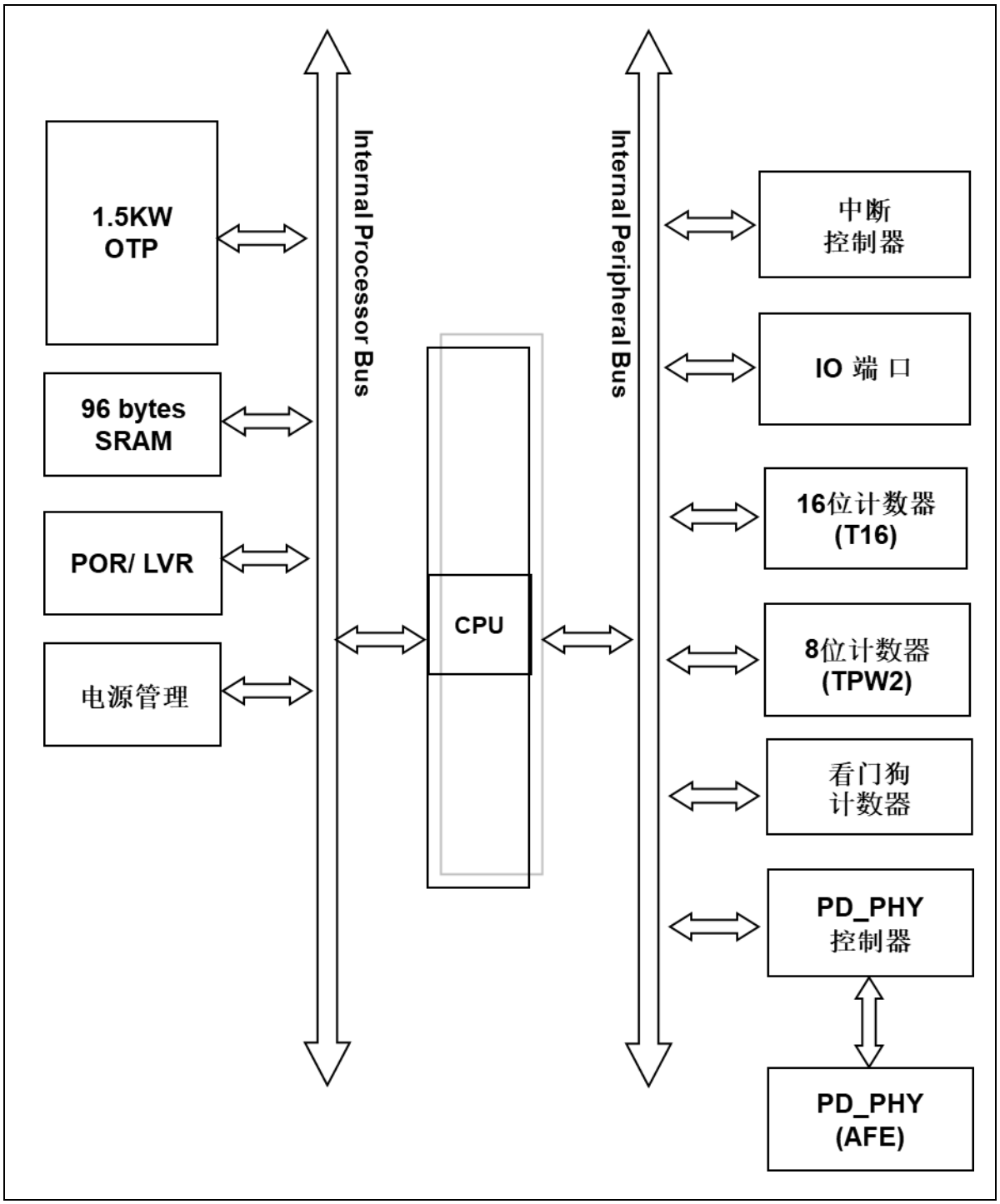
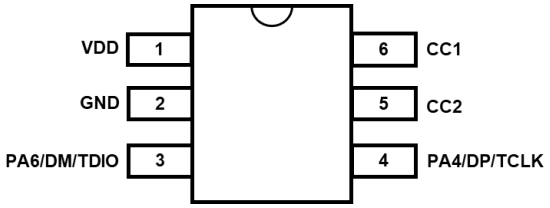
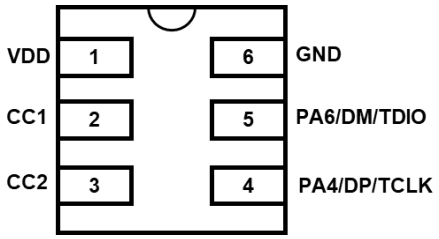


图. 2.1: PUD310 架构图

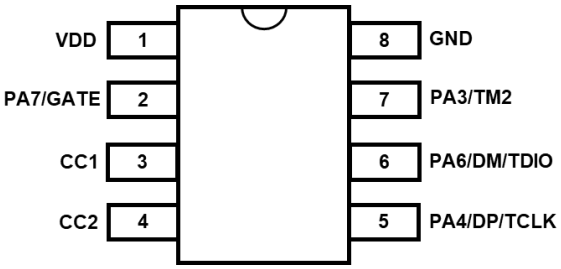
3. 引脚定义和功能描述



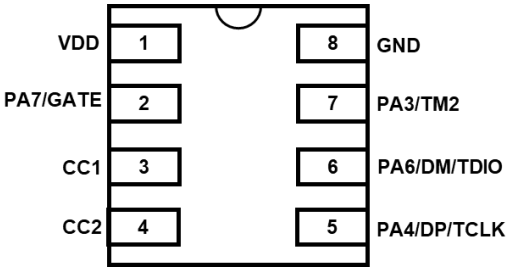
PUD310-U06A: SOT23-6 (60mil)



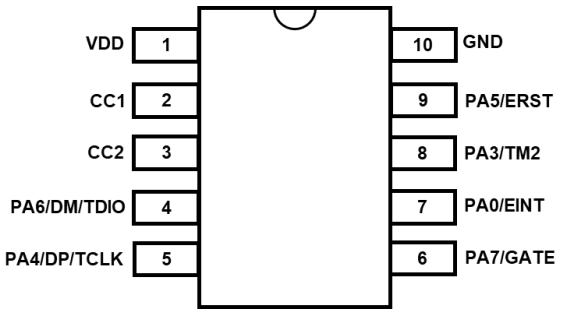
PUD310-2N06A: DFN (2*2mm)



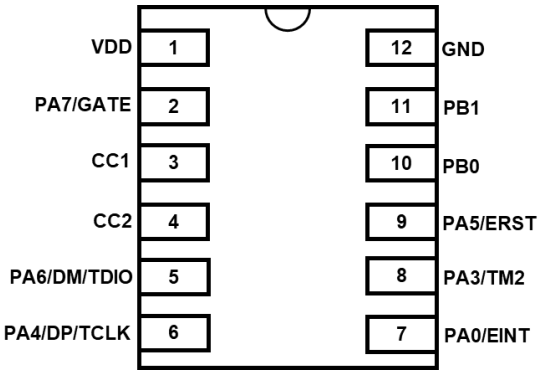
PUD310-S08A: SOP8 (150mil)



PUD310-2N08A: DFN (2*2mm)



PUD310-EY10A: ESSOP10 (150mil)



PUD310-4N12A: DFN (3*3mm)

引脚名称	引脚类型	描述
PA7 / GATE	OD HVT	该引脚的功能如下： (1) 用于连接外部功率 MOSFET，以开启/关闭电源线（PA7 不能用作通用输入输出（GPIO）引脚）。 可承受的最高电压为：21V。 当输出低电平（V_{OL}）为 0.5V 时，灌电流为 20mA。
PA6 / DM / TDIO	IO HVT	该引脚的功能如下： (1) USB DM 引脚（PA6 不能用作通用输入输出（GPIO）引脚）。 (2) 作为 OTP 编程的下载线 TDIO 引脚。 如果该引脚用作下载线引脚，则必须在测试模式下工作。 可承受的最高电压为：13V。
PA5 / PRSTB	IO (OD) ST / CMOS	此引脚可以用作： 端口 A 位 5，此引脚可以设定为输入或开漏输出(open drain)模式，弱上拉电阻模式。 硬件复位。 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。**另外，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 / DP / TCLK	IO HVT	该引脚的功能如下： (1) USB DP 引脚（PA4 不能用作通用输入输出（GPIO）引脚）。 (2) 作为 OTP 编程的下载线 TCLK 引脚。 如果该引脚用作下载线引脚，则必须在测试模式下工作。 可承受的最高电压为：13V。
PA3 / TM2PWM	IO ST / CMOS	此引脚可以用作： (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)Timer2 的 PWM 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；但是，当寄存器 ***padier*** 位 3 为”0”时，唤醒功能是被关闭的。
CC2	HVT	该引脚的功能如下： (1)USB 的 CC2 引脚。 可承受的最高电压为：24V。[注意 2]
CC1	HVT	该引脚的功能如下： (1)USB 的 CC1 引脚。 可承受的最高电压为：24V。[注意 2]

引脚名称	引脚类型	描述
PA0 / EINT	IO ST / CMOS	此引脚可以用作： (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) 外部中断源 0。它可用作外部中断源 0。上升沿和下降沿均会触发中断服务请求，并且可通过寄存器设置进行配置。 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 0 为"0"时，唤醒功能是被关闭的。
PB1	IO ST / CMOS	此引脚可以用作： 端口 B 位 1，并可编程设定为输入或输出，弱上拉电阻模式。 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 ***pbdier*** 位 1 为"0"时，唤醒功能是被关闭的。
PB0	IO ST / CMOS	此引脚可以用作： 端口 B 位 0，并可编程设定为输入或输出，弱上拉电阻模式。 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 ***pbdier*** 位 0 为"0"时，唤醒功能是被关闭的。
VDD	VDD	正电源。 [NOTE 2]
GND	GND	地

注意 1：**IO**：输入/输出；**ST**：施密特触发器输入；**OD**：开漏；**Analog**：模拟输入引脚；
CMOS：CMOS 电压基准位 **HVT**: 可承受的最高电压

注意 2：CC1 和 CC2 的容差按照下列波形进行测试（如图 3.1 所示）：

(1) 在 T1 时刻，提供 VDD = 24V 电源。
(2) 在 T1+30 毫秒时，对 CC1/CC2 提供 24V 电源，用于模拟 CC1/CC2 与 VDD（VBUS）短接的情况。
(3) 在 T1+60 毫秒时，关闭 VDD 和 CC1/CC2 的电源。
(4) 根据 USB PD 规范，当检测到 SINK 断开连接时，SOURCE 应在 20 毫秒内关闭 VBUS。
(5) 最大额定值 24V 是典型值，并且是在实验室室温条件下测试。

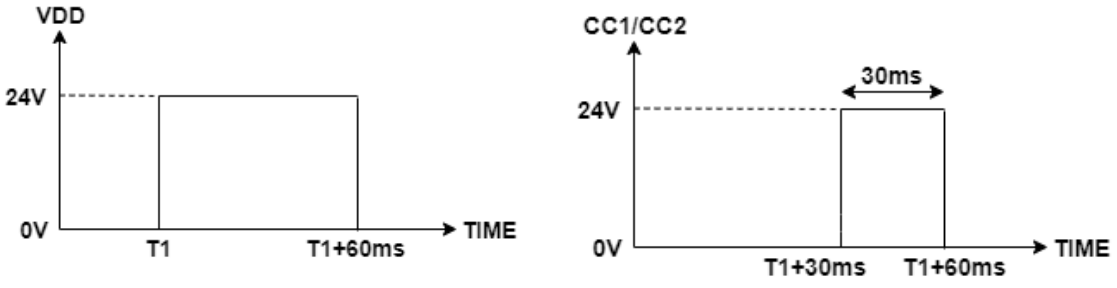


图 3.1：CC1/CC2 最大额定值测试方法

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 = -40°C ~ 85°C，V_{DD}=5.0V，f_{SYS}=3MHz 之条件下获得。

符号	描述	最小值	典型值	最大值	单位	条件(Ta=25°C)
V_{DD}	工作电压	4#		21	V	# 受限于 LVR 公差
LVR%	低电压复位公差	-7		7	%	
f_{SYS}	系统时钟(CLK)* = IHRC/1 IHRC/2 IHRC/4 IHRC/8 ILRC	 0 0 0 0	 41K	 12M 6M 3M 1.5M	Hz	 V_{DD} ≧ 3.0V V_{DD} ≧ 3.0V V_{DD} ≧ 3.0V V_{DD} ≧ 3.0V V_{DD} = 5.0V
V_{POR}	上电复位电压		1.8*			* 受限于 LVR 公差
I_{OP}	工作电流		1.7 30		mA A	f_{SYS} =IHRC/8 =1.5MIPS@5.0V f_{SYS} =ILRC =41KHz@5.0V
I_{PD}	掉电模式消耗电流 （使用 ***stopsys*** 命令）		0.24 0.15		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）		3.0		uA	V_{DD} =5.0V；f_{SYS} =ILRC 仅使用 ILRC 模式条件下
V_{IL}	输入低电压	0		0.2 V_{REG}	V	
V_{IH}	输入高电压	0.7 V_{REG}		V_{REG}	V	
I_{OL}	IO 输出灌电流					
	PA6, PA4 PA5, PA3, PA0, PB1, PB0		0.4 17		mA	V_{DD} =5.0V, V_{OL} =0.5V V_{DD} =5.0V, V_{OL} =0.5V
I_{OH}	IO 输出驱动电流					
	PA5, PA3, PA0, PB1, PB0 PA6, PA4		-13 -0.15		mA	V_{DD} =5.0V, V_{OH} =4.5V
V_{IN}	输入电压	-0.3		V_{DD} +0.3	V	
$I_{INJ\ (PIN)}$	引脚注入电流			1	mA	V_{DD} +0.3 ≧ V_{IN} ≧ -0.3
R_{PH}	上拉电阻		75		KΩ	V_{DD} =5.0V
R_{PL}	下拉电阻		75		KΩ	V_{DD} =5.0V
V_{BG}	Bandgap 参考电压	1.145*	1.20*	1.255*	V	V_{DD} =4.5V ~ 21V -40°C <Ta< 85°C*
f_{IHRC}	校准后 IHRC 频率 *	11.82*	12*	12.18*	MHz	25°C, V_{DD} =4.5V~21V
		11.40*	12*	12.60*		V_{DD} =4.5V~21V, -40°C <Ta< 85°C*
V_{REG}	稳压器输出电压		4.55			V_{DD} =5.0V
t_{INT}	中断脉冲宽度	30			ns	V_{DD} =5.0V

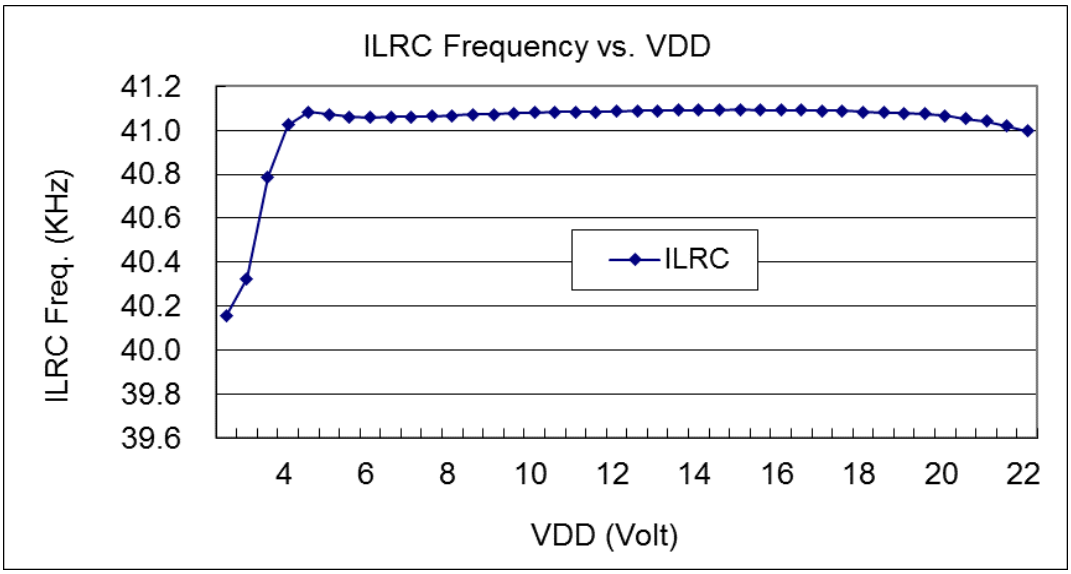
符号	描述	最小值	典型值	最大值	单位	条件(Ta=25°C)
V_{DR}	数据存储器数据保存电压*	1.5			V	待机模式下
t_{WDT}	看门狗超时溢出时间		8k		T_{ILRC}	misc[1:0]=00（默认）
			16k			misc[1:0]=01
			64k			misc[1:0]=10
			256k			misc[1:0]=11
t_{WUP}	快速唤醒时间		15		T_{ILRC}	T_{ILRC} 是 ILRC 时钟周期
	慢速唤醒时间		1024			
t_{SBP}	慢速启动时从开机开始的系统启动周期		25		ms	V_{DD} =5.0V
t_{RST}	外部复位脉冲宽度	120			us	@ V_{DD} =5.0V
CPos	比较器偏置电压*		±10		mV	
CPcm	比较器共模输入*	0		V_{DD} -1.5	V	
CPspt	比较器响应时间**		100	500	ns	上升沿和下降沿相同
CPmc	比较器模式改变所需的稳定时间		2.5	7.5	us	
CPcs	比较器电流消耗		20		uA	V_{DD} = 3.3V

* 这些参数并不是每个芯片测试。

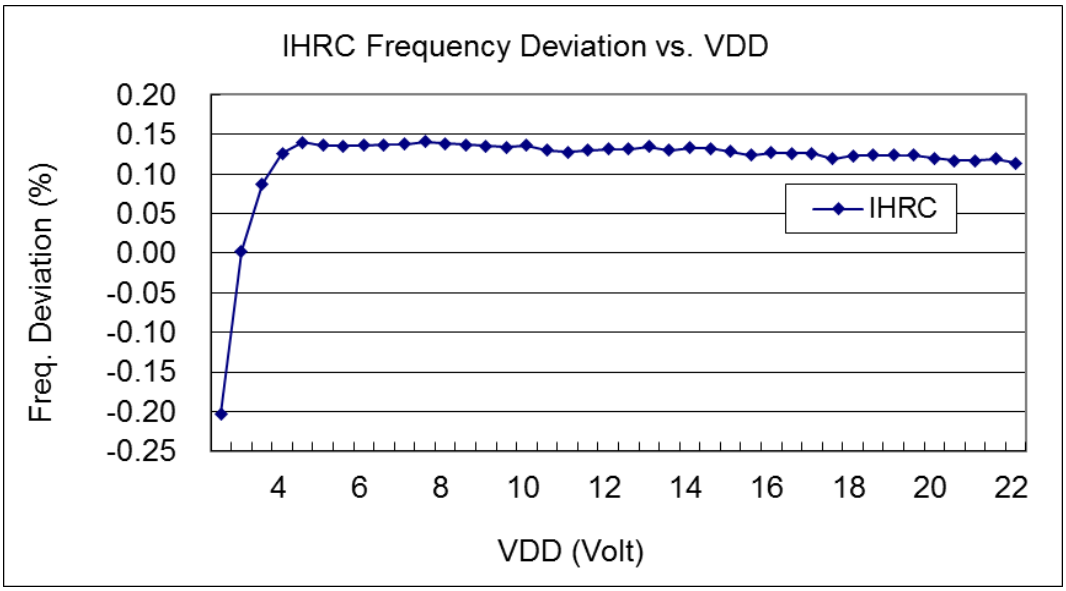
4.2. 绝对最大值范围

项目	最大额定值	备注
供应电压 (VDD)	24V	最大额定值按照图 3.1 进行测试。 电压（24V）或时间（30ms）过应力可能会导致永久性损坏。
GATE 电压	21V	若施加最大额定电压，可能导致 IC 永久损坏
CC1, CC2 电压	24V	按图 3.1 的波形测试；若时间（30ms）与电压（24V）超出范围，可能导致 IC 永久损坏
DP, DM 电压	13V	若施加最大额定电压，可能导致 IC 永久损坏
CC1/CC2 上的最大 C_{LOAD}	1.4nF	如果 CC1 或 CC2 上的总电容超过 1.4nF，可能会导致 USB PD 通信失败。
输入电压范围	-0.3V ~ V_{REG} + 0.3V	超出此范围可能损坏器件
操作温度范围	-40°C ~ 85°C	
储存温度范围	-50°C ~ 125°C	
接面温度	150°C	

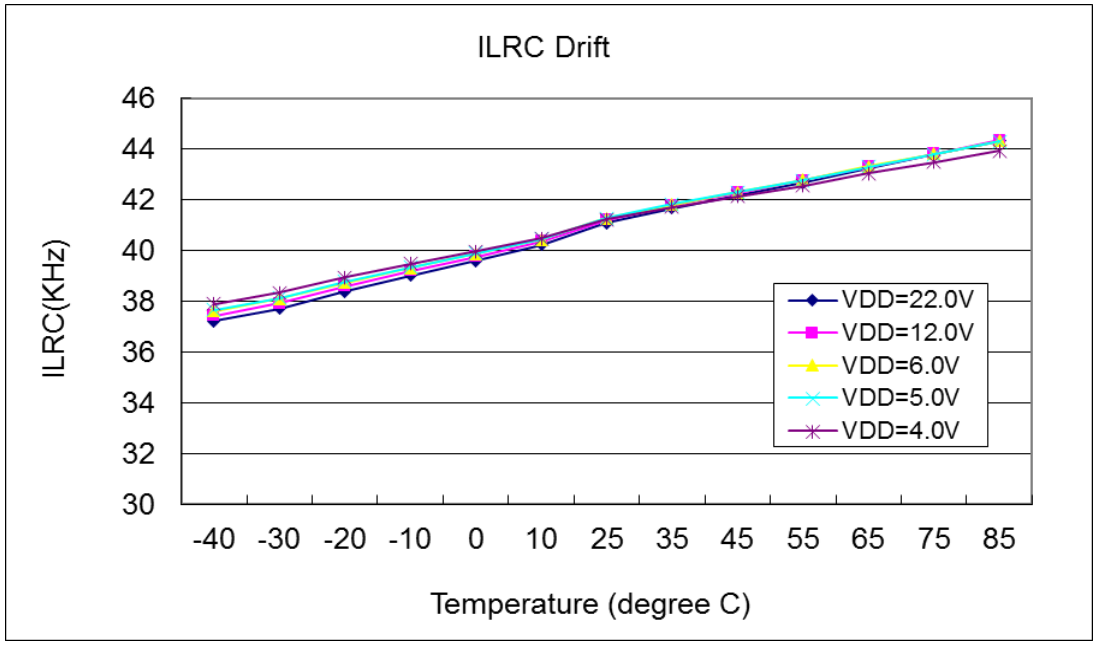
4.3. ILRC 频率与 VDD 关系曲线图



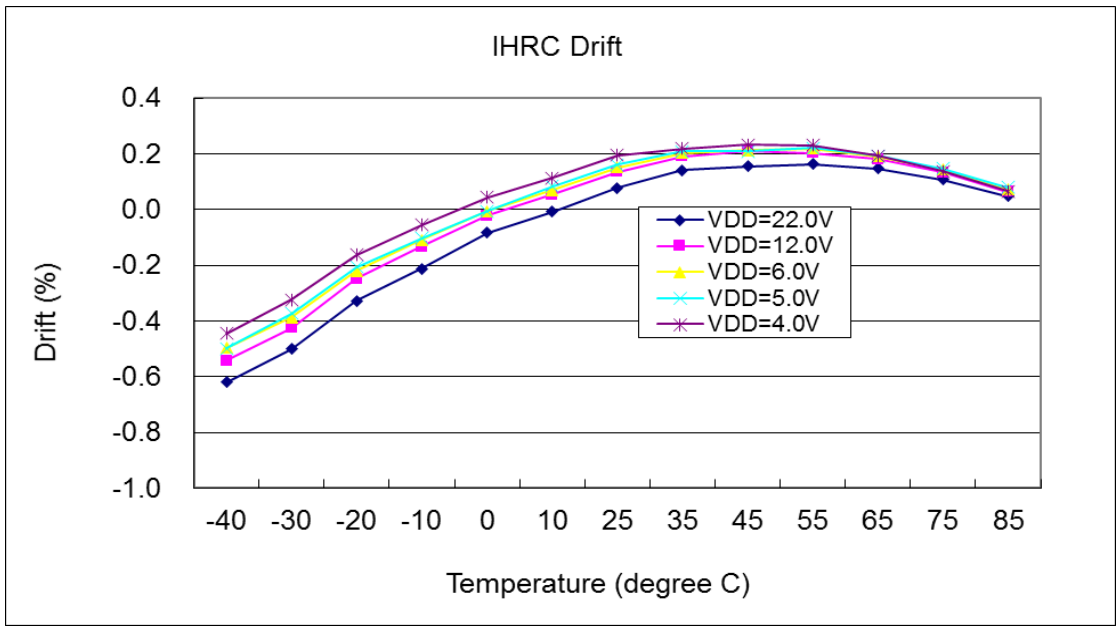
4.4. IHRC 频率与 VDD 关系曲线图（校准到 12MHz）



4.5. ILRC 频率与温度关系曲线图



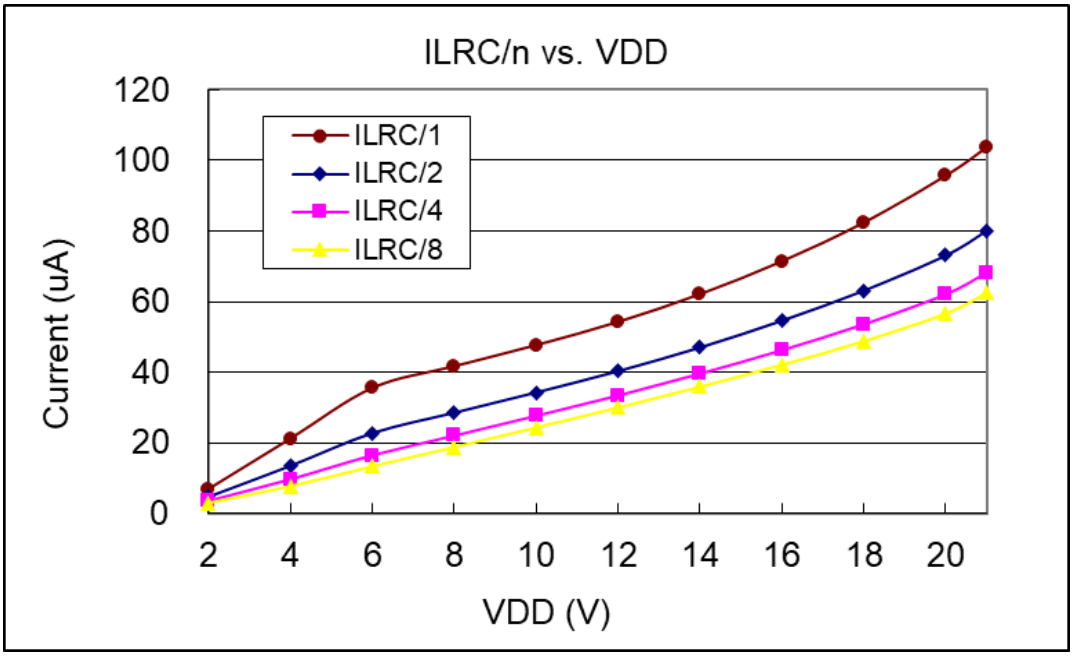
4.6. IHRC 频率与温度关系曲线图（校准到 12MHz）



4.7. 工作电流 vs.VDD @系统时钟= ILRC/n 关系曲线图

条件，**启用：** Bandgap，LVR，ILRC；**停用：** IHRC，T16，TM2；

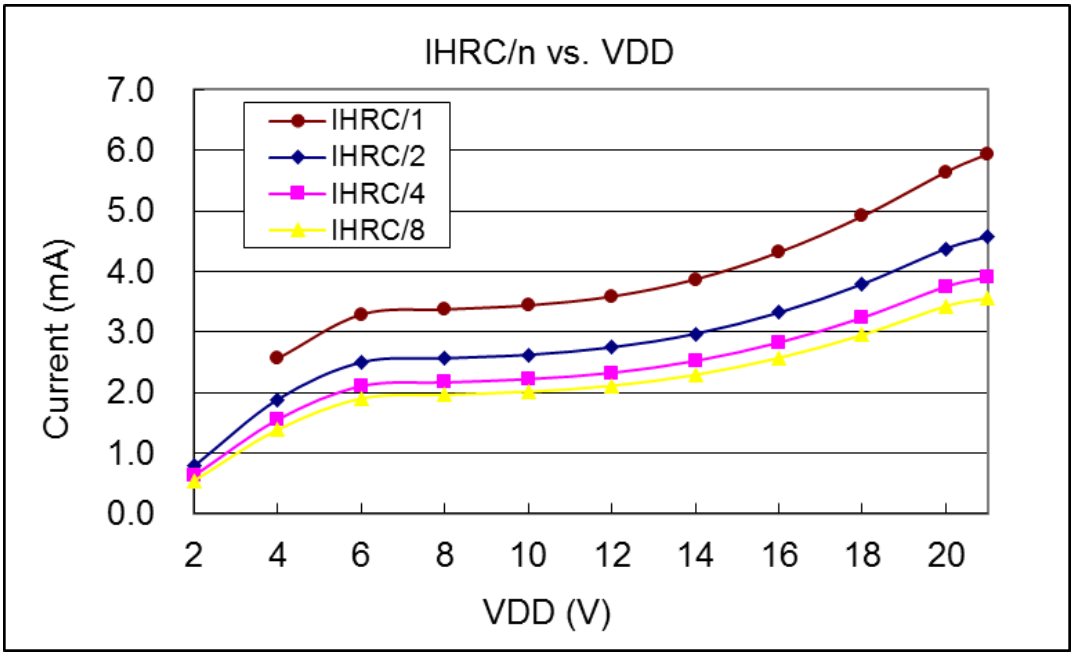
IO 引脚：PA0 以 0.5Hz 频率高低电压切换输出且无负载，**其他脚位**：设为输入且不浮空。



4.8. 工作电流 vs. VDD @系统时钟= IHRC/n 关系曲线图

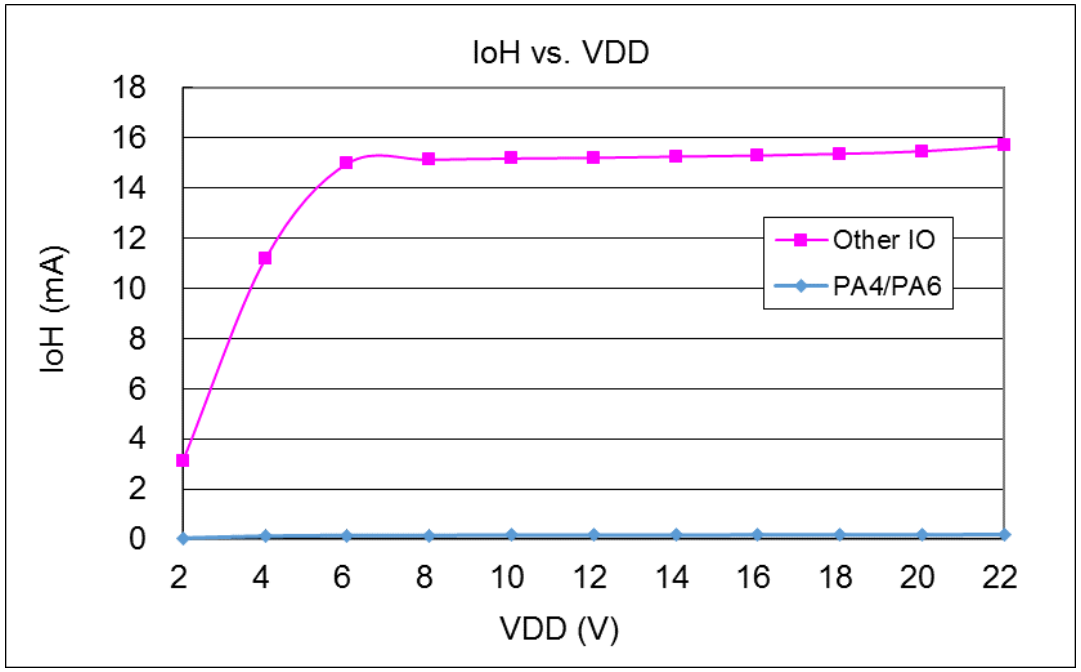
条件，**启用**：Bandgap，LVR，IHRC；**停用**：ILRC，T16，TM2；

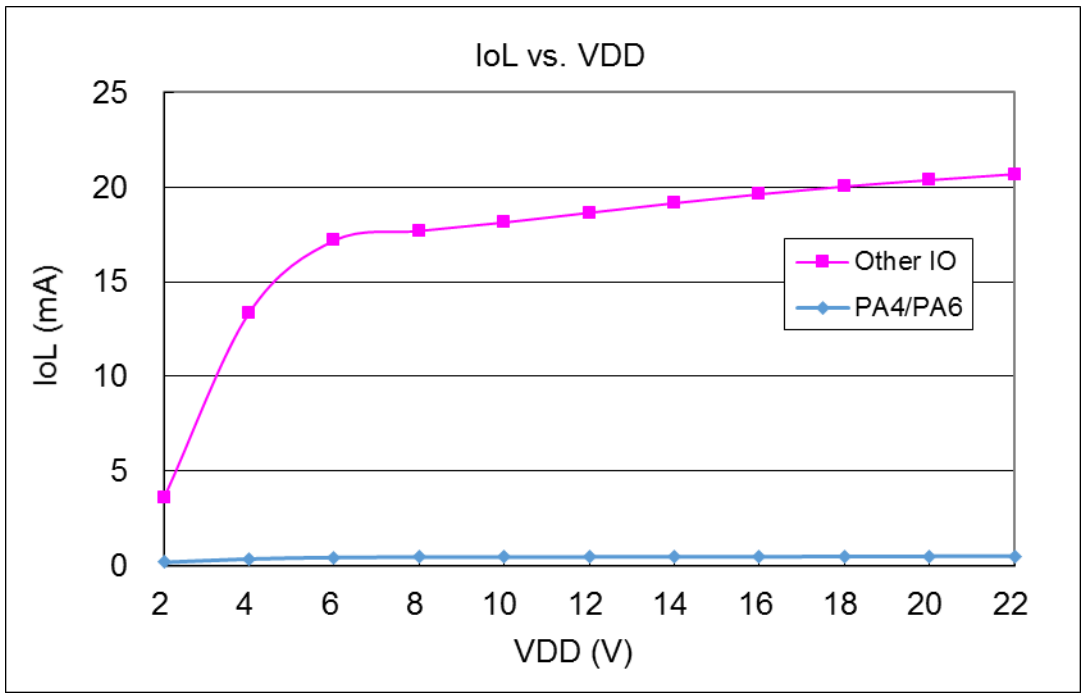
IO 引脚：PA0 以 0.5Hz 频率高低电压切换输出且无负载，**其他脚位**：设为输入且不浮空。



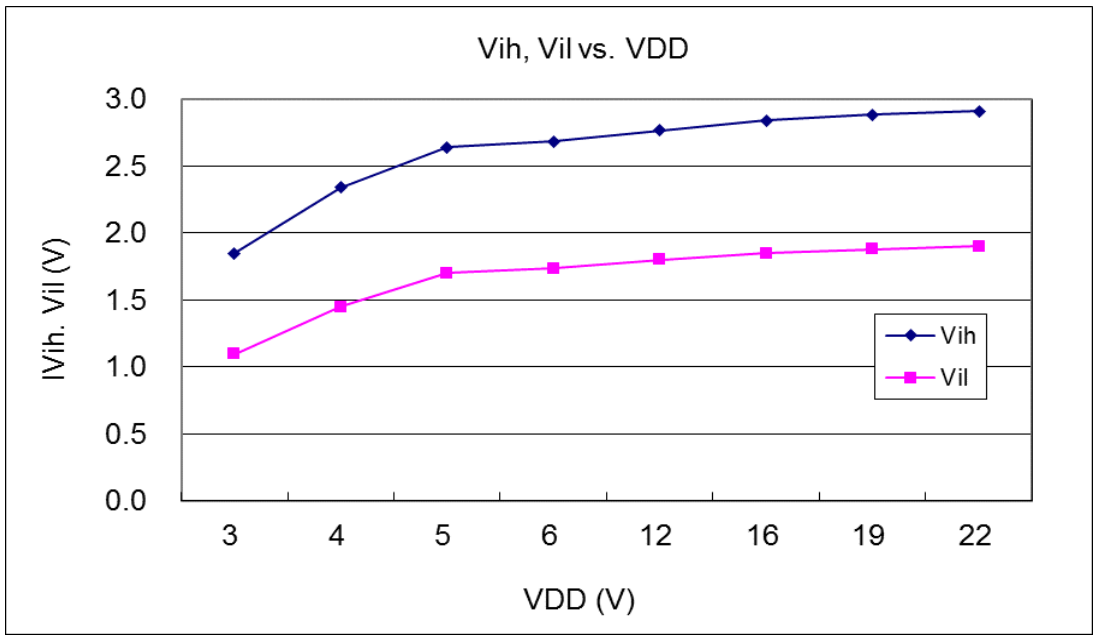
4.9. IO 驱动电流 (I_{OH}) 和灌电流 (I_{OL})关系曲线图

(VOH=0.9*V_{REG}, VOL=0.1*V_{REG})

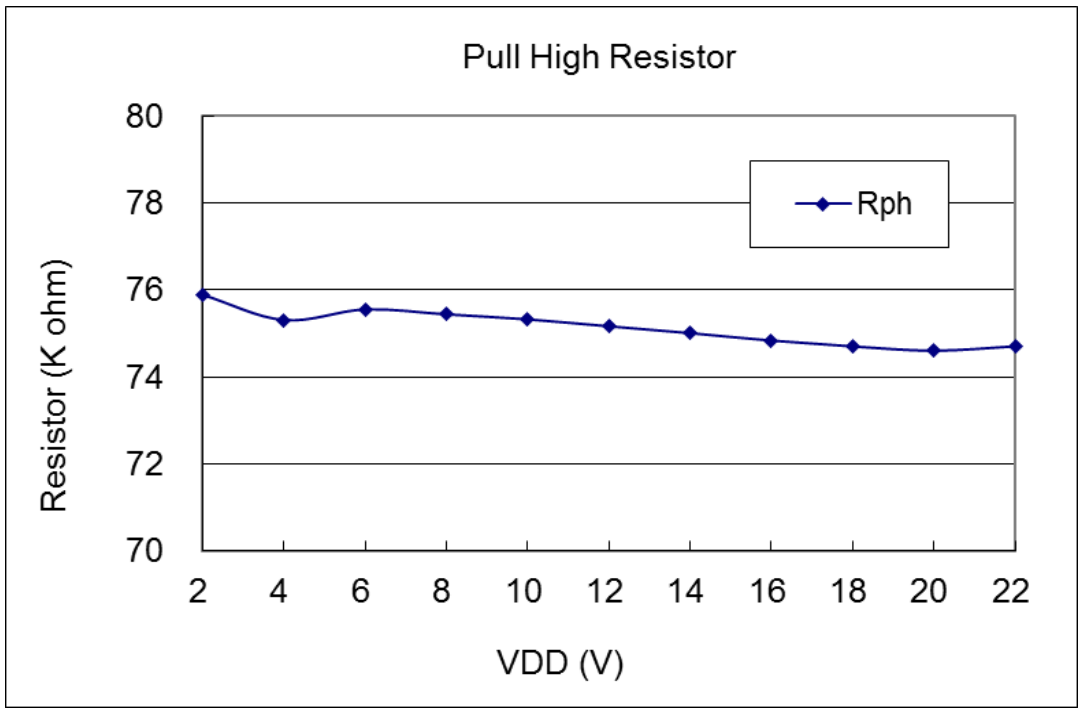




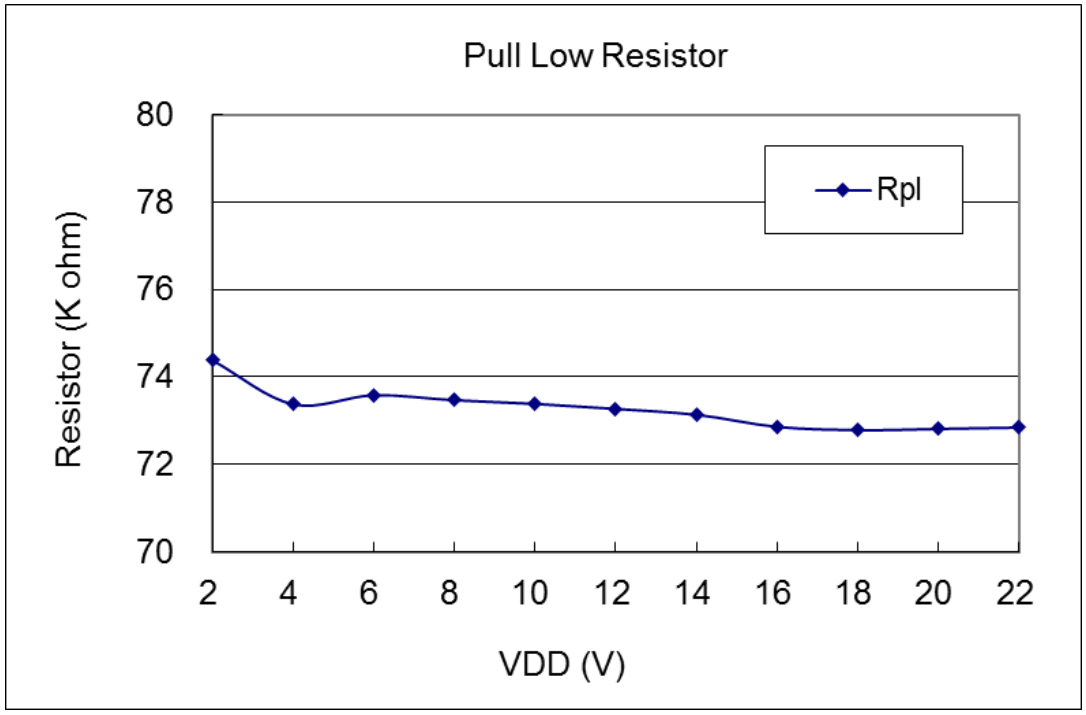
4.10. IO 输入高/低阈值电压 (V_{IH}/V_{IL})关系曲线图



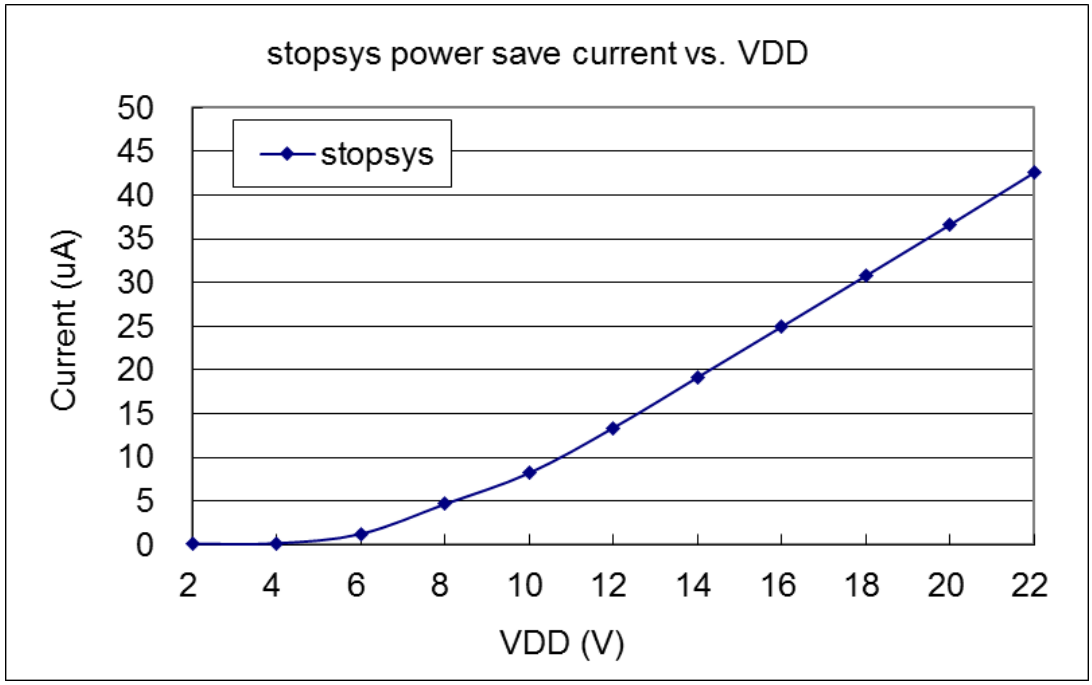
4.11. IO 引脚上拉阻抗曲线图

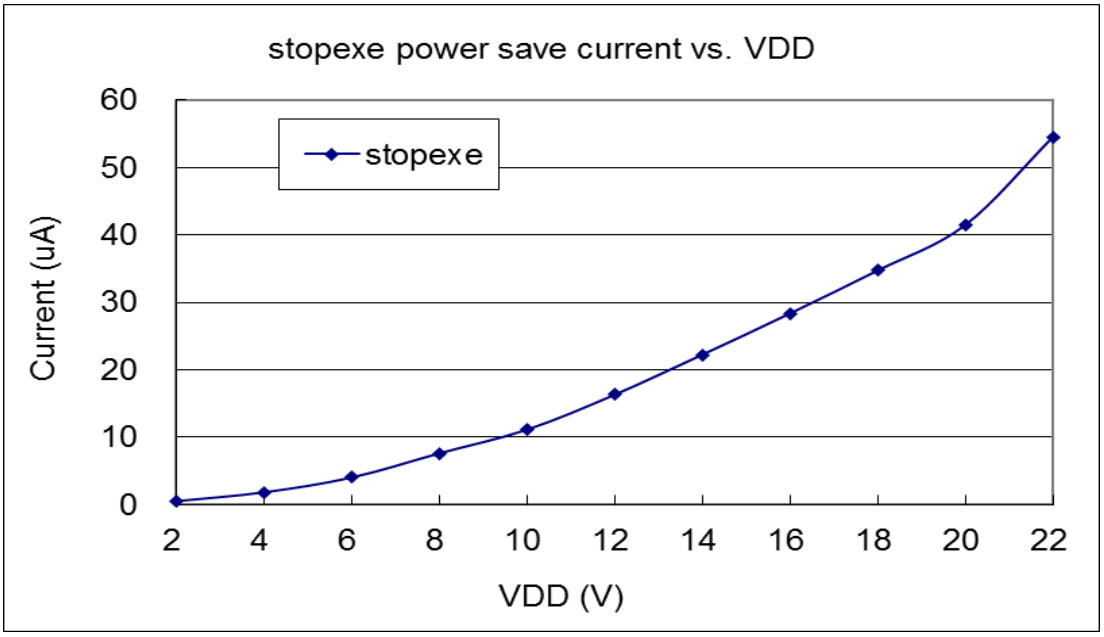


4.12. IO 引脚下拉阻抗曲线图

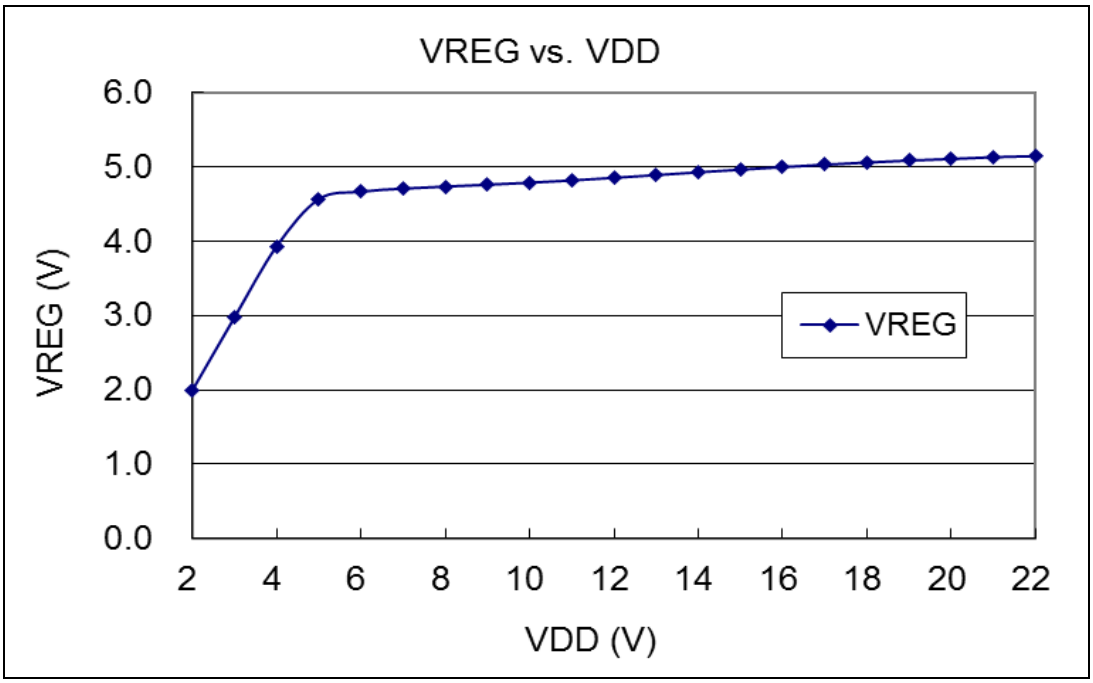


4.13. 掉电电流（I_{PD}）和省电电流（I_{PS}）关系曲线图





4.14. 不同 VDD 与 VREG 比较



4.15. DP/DM 引脚特性

条件：**开启**: Bandgap, LVR, IHRC; **关闭**: ILRC, T16

除非注明，否则所有数据均在 Ta= 25°C, V_{DD}=5.0V, f_{SYS} =12MHz 的条件下获得。

符号	描述	最小值	典型值	最大值	单位	条件(Ta=25°C)
V_{ODP5}	DP 5V 驱动电压		V_{REG}*		V	详见第 4.14 节不同 V_{DD} 与 V_{REG} 的关系
$V_{ODP3.3}$	DP 3.3V 驱动电压	3.0	3.3	3.6	V	
$V_{ODP0.6}$	DP 0.6V 驱动电压	0.5	0.6	0.7	V	
V_{ODP0}	DP 0V 驱动电压	-0.1	0.0	0.1	V	
V_{IDP3}	DP 比较器 3V 阈值	2.7	3.0	3.3	V	
$V_{IDP2.4}$	DP 比较器 2.4V 阈值	2.0	2.4	2.7	V	
$V_{IDP1.2}$	DP 比较器 1.2V 阈值	1.08	1.2	1.32	V	
$V_{IDP0.35}$	DP 比较器 0.35V 阈值	0.25	0.35	0.45	V	
V_{ODM5}	DM 5V 驱动电压		V_{REG}*		V	详见第 4.14 节不同 V_{DD} 与 V_{REG} 的关系
$V_{ODM3.3}$	DM 3.3V 驱动电压	3.0	3.3	3.6	V	
$V_{ODM0.6}$	DM 0.6V 驱动电压	0.5	0.6	0.7	V	
V_{ODM0}	DM 0V 驱动电压	-0.1	0.0	0.1	V	
V_{IDM3}	DM 比较器 3V 阈值	2.7	3.0	3.3	V	
$V_{IDM2.4}$	DM 比较器 2.4V 阈值	2.0	2.4	2.7	V	
$V_{IDM1.2}$	DM 比较器 1.2V 阈值	1.08	1.2	1.32	V	
$V_{IDM0.35}$	DM 比较器 0.35V 阈值	0.25	0.35	0.45	V	

*该参数取决于稳压器输出电压 (V_{REG})。

4.16. CC1/CC2 引脚特性

条件：**开启**: Bandgap, LVR, IHRC; **关闭**: ILRC, T16

除非注明，否则所有数据均在 Ta= 25°C, V_{DD}=5.0V, f_{SYS} =12MHz 的条件下获得。

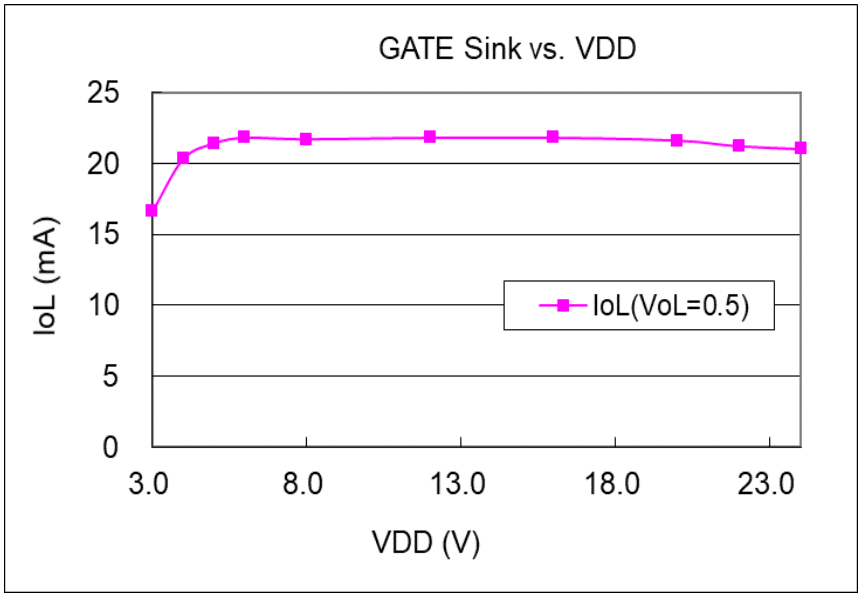
符号	描述	最小值	典型值	最大值	单位	条件(Ta=25°C)
V_{OHCC1}	CC1 1.125V 驱动电压	1.05	1.125	1.20	V	
V_{OLCC1}	CC1 0V 驱动电压	-0.1	0.0	0.1	V	
V_{ATTCC1}	CC1 比较器 0.2V 阈值	0.15	0.20	0.35	V	
V_{OHCC2}	CC2 1.125V 驱动电压	1.05	1.125	1.20	V	
V_{OLCC2}	CC2 0V 驱动电压	-0.1	0.0	0.1	V	
V_{ATTCC2}	CC2 比较器 0.2V 阈值	0.15	0.20	0.35	V	
$V_{1.5ACC}$	CC 比较器 0.66V 阈值	0.61	0.66	0.80	V	
V_{3ACC}	CC 比较器 1.23V 阈值	1.16	1.23	1.31	V	

4.17.GATE 引脚特性

条件：**开启**: Bandgap, LVR, IHRC; **关闭**: ILRC, T16

除非注明，否则所有数据均在 Ta= 25°C, V_{DD}=5.0V, f_{SYS} =12MHz 的条件下获得。

符号	描述	最小值	典型值	最大值	单位	条件(Ta=25°C)
V_{OLGATE}	栅极 0V 驱动电压		0.0		V	GATE 通过一个 31.5kΩ 电阻连接到 10V
V_{OHGATE}	栅极悬空输出测试		10.0		V	GATE 通过一个 31.5kΩ 电阻连接到 10V
V_{OHGATE}	栅极悬空输出测试		22		V	将栅极连接一个 55K 的电阻到 22V 的电源上
I_{GATE}	栅极灌电流		20		mA	在 V_{OL} = 0.5V 条件下



4.18. VBUS (VDD)比较器特性

条件: **开启**: Bandgap, LVR, IHRC; **关闭**: ILRC, T16

除非注明，否则所有数据均在 Ta= 25°C, V_{DD}=5.0V, f_{SYS} =12MHz 的条件下获得。

符号	描述	最小值	典型值	最大值	单位	条件(Ta=25°C)
V_{VBUS}	VBUS 比较器 7V 阈值	6.0	7.0	8.0	V	

5. 功能概述

5.1. OTP 程序存储器

OTP（一次性可编程）程序存储器用于存储要执行的程序指令。OTP 程序存储器可包含数据、表格和中断入口。复位后，初始地址 0x000 保留给系统使用，因此程序将从 0x001 开始，通常是 GOTO FPPA0 指令。如果使用，中断入口为 0x10，最后 16 个地址保留给系统使用，如校验和、序列号等。PUD310 的 OTP 程序存储器为 1.5KW，其分区如表 5.1 所示。地址 0x5F0 至 0x5FF 的 OTP 存储器供系统使用，地址空间 0x002 至 0x00F 和 0x011 至 0x5EF 为用户程序空间。

地址	功能
0x000	系统使用
0x001	GOTO FPPA0 指令
0x002	用户程序区
•	•
0x00F	用户程序区
0x010	中断入口地址
0x011	用户程序区
•	•
0x5EF	用户程序区
0x5F0	系统使用
•	•
0x5FF	系统使用

表 5.1： 程序存储器结构

5.2. 启动程序

开机时，POR（上电复位）是用于复位 PUD310。启动时间约为 1024 ILRC 时钟周期，用户在使用时，无论选择哪种开机方式，都必须确保上电后电源电压稳定，开机时序如图 5.1 所示，其中 t_{SBP} 是开机时间。

注意，上电复位(Power-On Reset)时，V_{DD} 必须先超过 V_{POR} 电压，MCU 才会进入开机状态。

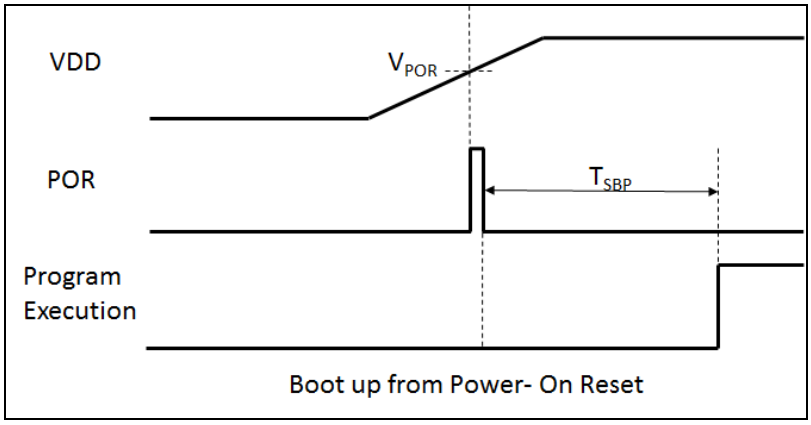
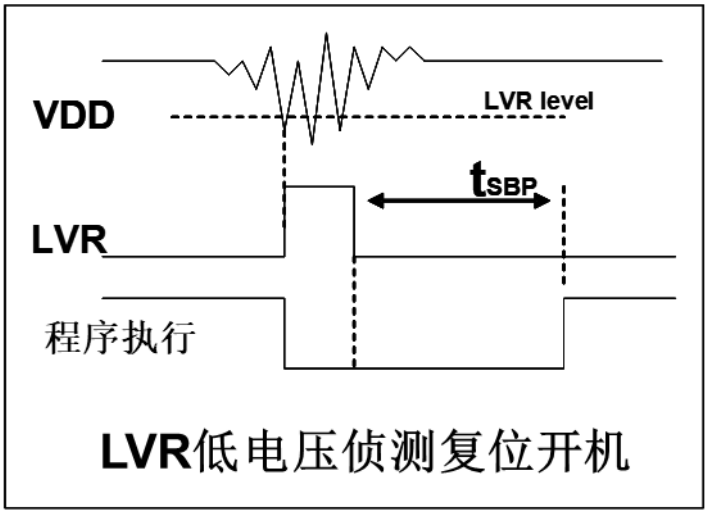
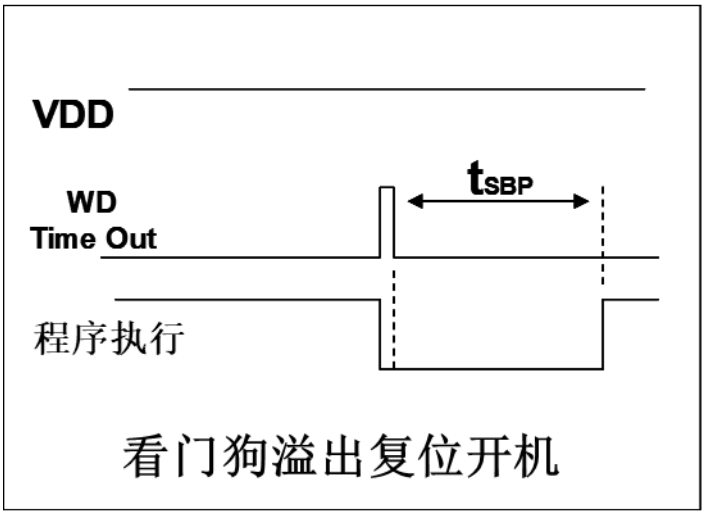
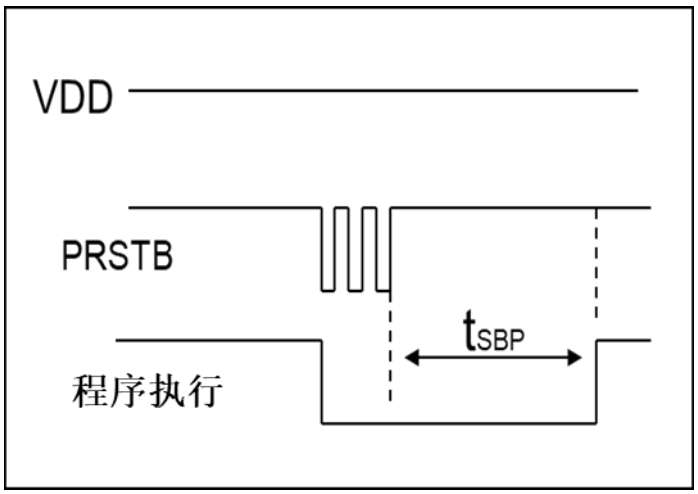


图 5.1：上电复位时序

5.2.1. 复位时序图


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


VDD
WD
Time Out
tSBP
程序执行
看门狗溢出复位开机


VDD
PRSTB
tSBP
程序执行

5.3. 数据存储器 - SRAM

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

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

对于间接存储指令而言，数据存储器可以用作数据指针来当作数据地址。所有的数据存储器都可以当作资料指针，这对于间接存储指令是相当灵活和有效的。由于数据宽度是 8 位，PUD310 的所有 96 字节的数据存储器都可以利用间接存取指令有存取。

5.4. 振荡器和时钟

PUD310 有 3 个振荡器电路：内部高速 RC 振荡器（IHRC）、内部低速 RC 振荡器（ILRC）以及内部超低速 RC 振荡器（NILRC）。IHRC 的启用或关闭由缓存器 clkmd.1 控制，clkmd.0 则用于在 ILRC 和 NILRC 之间切换。用户可以选择这三种振荡器中的任意一种作为系统时钟来源，并透过 clkmd 缓存器设定所需的系统时钟频率，以满足不同应用的需求，详见表 5.2。

振荡器模块	启用/停用
IHRC	**clkmd**.1
振荡器模块	**开关**
ILRC	**clkmd**.0 = 0
NILRC	**clkmd**.0 = 1

表 5.2：三个振荡器电路

5.4.1. 内部高频 RC 振荡器和内部低频 RC 振荡器

开机后，IHRC 和 ILRC 振荡器是被启用的。IHRC 频率能通过 ***ihrcr*** 寄存器校准，通常校准到 12MHz。详细请参考 IHRC 与 V_{DD} 及温度关系曲线图。

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

5.4.2. 芯片校准

在芯片生产制造时，IHRC 频率和 bandgap 参考电压都有可能稍微不同，PUD310 提供 IHRC 频率校准来消除这些差异，校准功能可以被用户的程序选择并编译，同时这个命令会自动嵌入用户的程序里面，校准命令如下所示：

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

其中，**p1**=2, 4, 8; 用以提供不同的系统时钟。

p2=1.5 ~ 12; 用以校准芯片到不同的频率，12MHz 是通用的选择。

p3=1.8 ~ 5.5; 用以在不同的工作电压下校准频率。

5.4.3. IHRC 频率校准和系统时钟

在用户编译程序时，IHRC 频率校准和系统时钟的选项如表 3 所示：

SYSCLK	CLKMD	IHRCR	描述
○ Set IHRC / 1	= 22h (IHRC / 1)	有校准	IHRC 校准到 12MHz, CLK=12MHz (IHRC/1)
○ Set IHRC / 2	= 62h (IHRC / 2)	有校准	IHRC 校准到 12MHz, CLK=6MHz (IHRC/2)
○ Set IHRC / 4	= A2h (IHRC / 4)	有校准	IHRC 校准到 12MHz, CLK=3MHz (IHRC/4)
○ Set IHRC / 8	= E2h (IHRC / 8)	有校准	IHRC 校准到 12MHz, CLK=1.5MHz (IHRC/8)
○ Set ILRC	= 02h (ILRC / 1)	有校准	IHRC 校准到 12MHz, CLK=ILRC
○ Disable	不改变	没改变	IHRC 不校准, CLK 不改变

表 5.3：IHRC 频率校准选项

通常，.ADJUST_IC 是开机后第一条指令，以设定系统的工作频率。IHRC 频率校准仅在烧录 OTP 程序代码的时候执行一次，之后不会重复执行了，如果用户选择了不同的频率校准选项，开机后的系统状态也会不同。以下所示为不同的选项开机后，PUD310 执行此命令后的状态：

(1) .ADJUST_IC SYSCLK=IHRC/1, IHRC=12MHz, V_{DD}=5V

开机后，RSTC = 0x04, CLKMD = 0x22：

- IHRC 的校准频率为 12MHz@V_{DD}=5V，启用 IHRC 模块。
- 系统时钟 = IHRC/1 = 12MHz
- 看门狗计数器停用，ILRC 启用，PA5 引脚是输入模式。

(2) .ADJUST_IC SYSCLK=IHRC/2, IHRC=12MHz, V_{DD}=3.3V

开机后，RSTC = 0x04, CLKMD = 0x62：

- IHRC 的校准频率为 12MHz@V_{DD}=3.3V，启用 IHRC 模块。
- 系统时钟 = IHRC/2 = 6MHz
- 看门狗计数器停用，ILRC 启用，PA5 引脚是输入模式。

(3) .ADJUST_IC SYSCLK=IHRC/4, IHRC=12MHz, V_{DD}=2.5V

开机后，RSTC = 0x04, CLKMD = 0xA2：

- IHRC 的校准频率为 12MHz@V_{DD}=2.5V，启用 IHRC 模块。
- 系统时钟 = IHRC/4 = 3MHz
- 看门狗计数器停用， ILRC 启用，PA5 引脚是输入模式。

(4) .ADJUST_IC　　SYSCLK=IHRC/8, IHRC=12MHz, V_{DD}=2.5V

开机后，RSTC = 0x04, CLKMD = 0xE2：

- IHRC 的校准频率为 12MHz@V_{DD}=2.5V，启用 IHRC 模块。
- 系统时钟 = IHRC/8 = 1.5MHz
- 看门狗计数器停用，ILRC 启用，PA5 引脚是输入模式。

(5) .ADJUST_IC　　SYSCLK=ILRC, IHRC=12MHz, V_{DD}=5V

开机后，RSTC = 0x04, CLKMD = 0x02：

- IHRC 的校准频率为 12MHz@V_{DD}=5V，启用 IHRC 模块。
- 系统时钟 = ILRC = 32KHz
- 看门狗计数器停用，ILRC 启用，PA5 引脚是输入模式。

(6) .ADJUST_IC　　DISABLE

开机后，CLKMD 寄存器没有改变（没任何动作）：

- IHRC 不校准
- 系统频率= ILRC
- 看门狗计数器启用，ILRC 启用，PA5 引脚是输入模式。

5.4.4. 系统时钟和 LVR 水平

系统时钟的时钟源来自 IHRC、ILRC 和 NILRC，PUD310 中系统时钟的硬件图如图 5.2 所示。

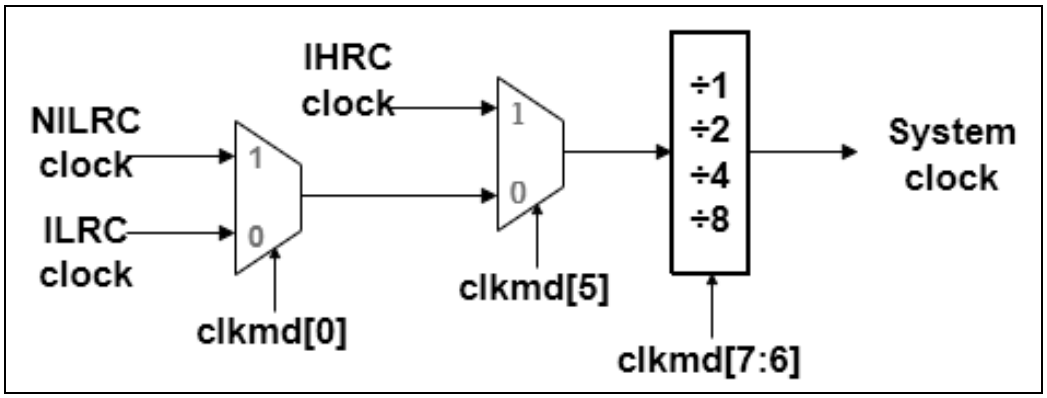


图 5.2：系统时钟选项

用户可根据自己的需求选择不同的操作系统时钟；所选的操作系统时钟应与电源电压和 LVR 水平相结合，以使系统稳定。LVR 水平将在编译时选择。请参阅第 4.1 节。

5.4.5. 系统时钟切换

IHRC 校准后，用户可能要求切换系统时钟到新的频率或者可能会随时切换系统时钟来优化系统性能及功耗。基本上，PUD310 的系统时钟能够随时通过设定寄存器 ***clkmd*** 在 IHRC，ILRC 和 EOSC 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”。

Case 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           =   0x62;           //  切换到 IHRC/2
...
```

Case 2： 系统时钟从 IHRC/2 切换到 ILRC

```
...                                 //  系统时钟是 IHRC/2
CLKMD           =   0x02;           //  切换到 ILRC，IHRC 不能在这里停用
CLKMD.2         =   0;              //  IHRC 可以在这里停用
...
```

Case 3： 系统时钟从 IHRC/2 切换到 IHRC/4

```
...                                 //  系统时钟是 IHRC/2，此处启用 ILRC
CLKMD           =   0XA2;           //  切换到 IHRC/4
...
```

Case 4： 如果同时切换系统时钟关闭原来的振荡器，**系统会宕机**

```
...                                 //  系统时钟是 IHRC/2
CLKMD           =   0x00;           //  不能在将时钟从 IHRC/2 切换到 ILRC 的同时
                                    //  同时关闭 ILRC 振荡器
```

5.5. 16 位计数器 (Timer16)

PUD310 内置一个 16 位硬件计数器 (Timer16)，计数器时钟可来自于系统时钟 (CLK)，内部高频振荡时钟 (IHRC)，内部低频振荡时钟 (ILRC)和 PA0。在送到 16 位计数器之前，1 个可软件程序设计的预分频器提供÷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。选择器用于选择 Timer16 的中断条件，无论何时发生溢出，都可以触发 Timer16 中断。计时器 16 的硬件框图如图 5.3 所示。16 位计数器的中断请求可以通过 16 位计数器的位[15:8]来选择，中断类型可以上升沿触发或下降沿触发，由 ***integ*** 寄存器（地址 0x09）的第 4 位指定。

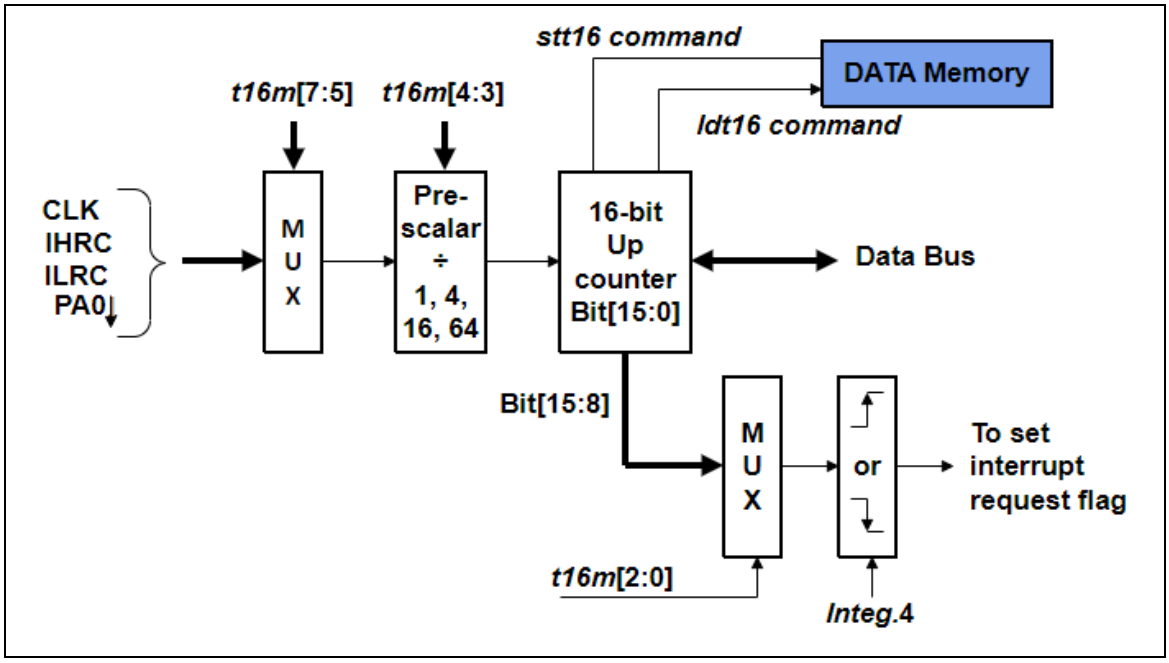


图 5.3：Timer16 硬件图

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        0x0B
$ 7~5: STOP, SYSCLK, X, X, IHRC, X, ILRC, PA0_F        //第一个参数
$ 4~3: /1, /4, /16, /64                                 //第二个参数
$ 2~0: BIT8, BIT9, BIT10, BIT11, BIT12, BIT13, BIT14, BIT15    //第三个参数
```

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

```
$ T16M SYSCLK, /64, BIT15;
    // 选择(SYSCLK/64) 当 Timer16 时钟源，每 2^16 个时钟周期产生一次 INTRQ.2=1
    // 系统时钟 = IHRC / 2 = 6MHz
    // SYSCLK/64 = 6 MHz/64 = 93.75KHz,约每 700 mS 产生一次 INTRQ.2=1

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

5.6. 8 位定时器(Timer2) / PWM 生成器

PUD310 内置 1 个 8 位硬件 PWM 计数器(Timer2)。图 5.4 为 Timer2 硬件框图，Timer2 的时钟源可以来自内部高 RC 振荡器 (IHRC)、2 倍内部高 RC 振荡器 (IHRC*2) 和内部低 RC 振荡器 (ILRC)。寄存器 tpw2c 的位[5：4]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。 Timer2 的输出可以是 PA3 引脚。利用软件编程寄存器 tpw2s 位[6：5]，时钟预分频模块提供÷1，÷4，÷16 和÷64 的选择，由 tpw2c 寄存器的[3:2]位控制；还提供一个分频范围为 1~32 的分频模块，由 tpw2s 寄存器的[4:0]位控制。在结合预分频器以及分频器，Timer2 PWM 时钟 (TPW2_CLK) 频率可以广泛和灵活，以提供不同产品应用。

Timer2 的 8 位计数器只能执行向上计数操作，在周期模式下，当 8 位计数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、7 位或 8 位。图 5.4 显示出 Timer2 周期模式和 PWM 模式的时序图。

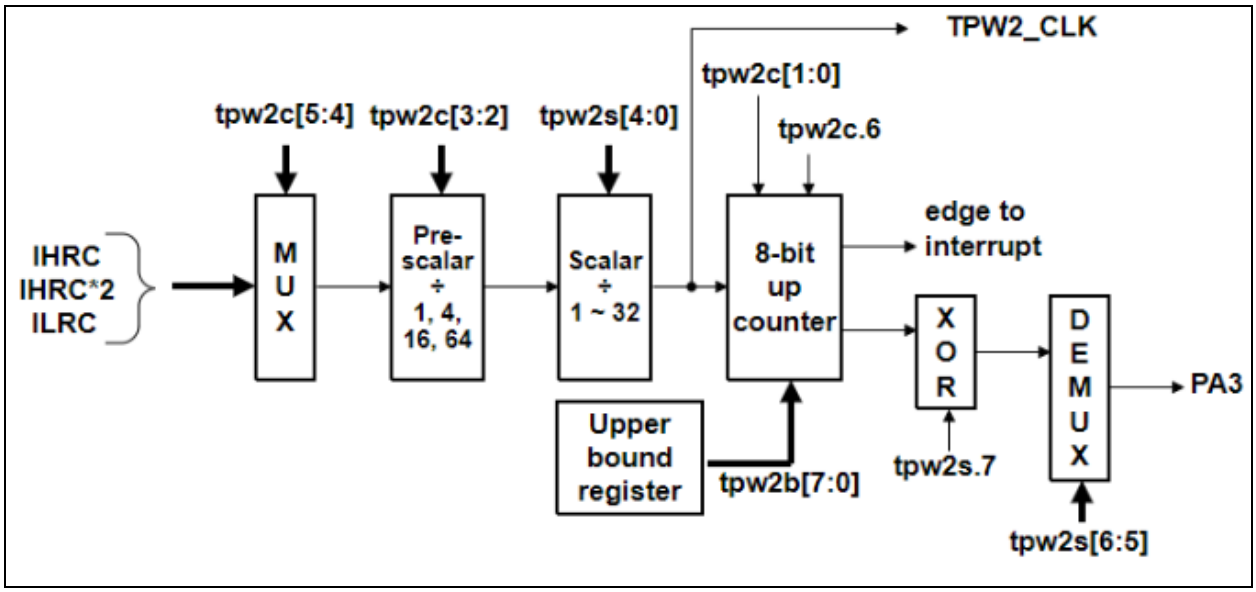


图 5.4：Timer2 硬件框图

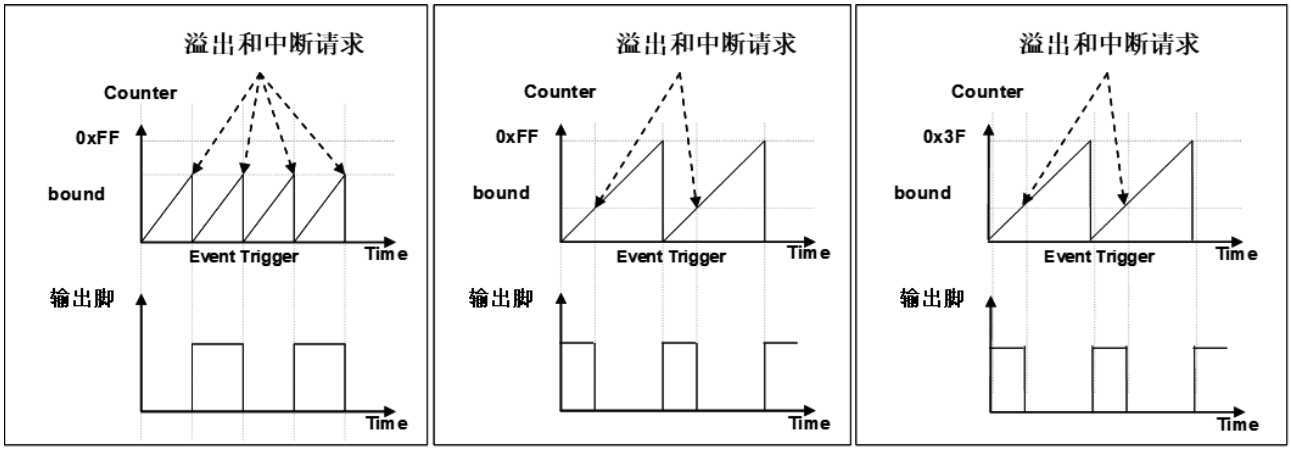


图 5.5：Timer2 周期模式和 PWM 模式的时序图（tpw2c.6=1）

5.6.1 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)]

其中, Y = tpw2c[5:4]：Timer2 所选择的时钟源频率
K = tpw2b[7:0]：上限寄存器设定的值（十进制）
S1 = tpw2c[3:2]：预分频器设定值 (S1= 1，4，16，64)
S2 = tpw2s[4:0]：分频器值（十进制，S2= 0 ~ 31）

例 1：

tpw2c = 0b0001_1000，Y=12MHz, S1=1
tpw2b = 0b0111_1111，K=127
tpw2s = 0b0000_00000，S2=0
➔ 输出频率 = 12MHz ÷ [2 × (127＋1) × 1 × (0＋1)] = 46.875KHz

例 2：

tpw2c = 0b0001_1000，Y=12MHz, S1=64
tpw2b = 0b0111_1111，K=127
tm2s[7:0] = 0b0111_11111，S2 = 31
➔ 输出频率 = 12MHz ÷ (2 × (127＋1) × 64 × (31＋1)) =22.89Hz

例 3：

tpw2c = 0b0001_1000, Y=12MHz, S1=1
tpw2b = 0b0000_1111，K=15
tpw2s = 0b0000_00000，S2=0
➔ 输出频率 = 12MHz ÷ (2 × (15＋1) × 1 × (0＋1)) = 375KHz

例 4：

tpw2c = 0b0001_1000，Y=12MHz, S1=1
tpw2b = 0b0000_0001，K=1
tm2s = 0b0000_00000，S2=0
➔ 输出频率 = 12MHz ÷ (2 × (1＋1) × 1 × (0＋1)) =3MHz

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=12MHz, VDD=5V
    …
    tpw2c = 0x0;
    tpw2b = 0x7f;
    tpw2s = 0b0_10_00001;          //    output=PA3,，分频 = 2
    tpw2c = 0b1000_00_00;          //    周期模式, IHRC，预分频 = 1
    while(1)
    {
        nop;
    }
}
```

5.6.2 使用 Timer2 产生 8 位 PWM 波形

如果选择 8 位 PWM 的模式，应设立 ***tpw2c***[6]=1　***tpw2c***[1:0]=2'b00,，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%

其中,　　Y = tpw2c[5:4]：所选择的时钟源频率

K = tpw2b[7:0]：上限寄存器设定的值（十进制）

S1= tpw2c[3:2]：预分频器设定值 (S1= 1，4，16，64)

S2 = tpw2s[4:0]：分频器值（十进制，S2= 0 ~ 31）

例 1：

tpw2c = 0b0100_0000, Y=12MHz, S1=1
tpw2b = 0b0111_1111, K=127
tpw2s = 0b010_00000, S2=0
➔ 输出频率 = 12MHz ÷ (256 × 1 × (0+1)) = 46.875KHz
➔ 输出占空比 = [(127+1) ÷ 256] × 100% = 50%

例 2：

tpw2c = 0b0100_1100, Y=12MHz, S1=64
tpw2b = 0b0111_1111, K=127
tpw2s = 0b010_11111, S2=31
➔ 输出频率 = 12MHz ÷ (256 × 64 × (31+1)) = 22.89Hz
➔ 输出占空比 = [(127+1) ÷ 256] × 100% = 50%

例 3：

tpw2c = 0b0100_0000, Y=12MHz, S1=1
tpw2b = 0b1111_1111, K=255
tpw2s = 0b010_00000, S2=0
➔ PWM 输出是高电平
➔ 输出占空比 = [(255+1) ÷ 256] × 100% = 100%

例 4：

tpw2c = 0b0100_0000, Y=12MHz, S1=1
tpw2b = 0b0000_1001, K = 9
tpw2s = 0b010_00000, S2=0
➔ 输出频率 = 12MHz ÷ (256 × 1 × (0+1)) = 46.875KHz
➔ 输出占空比 = [(9+1) ÷ 256] × 100% = 3.9%

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=12MHz, VDD=5V
    wdreset;
    tpw2c = 0x0;
    tpw2b = 0x7f;
    tpw2s = 0b0_10_00001;           //      output=PA3, 分频= 2
    tpw2c = 0b1100_00_00;           //      PWM 模式, IHRC, 预分频 = 1
                                    //      8 位 PWM
    while(1)
    {
        nop;
    }
}
```

5.6.3 使用 Timer2 产生 6 位 PWM 波形

如果选择 6 位 PWM 的模式，应设立 ***tpw2c***[6]=1 and ***tpw2c***[1:0]=2'b10,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%

其中, tpw2c[5:4] = Y：所选择的时钟源频率
tpw2b[7:0] = K：上限寄存器设定的值（十进制）
tpw2c[3:2] = S1：预分频器设定值 (S1= 1，4，16，64)
tpw2s[4:0] = S2：分频器值（十进制，S2= 0 ~ 31）

例 1：

tpw2c = 0b0001_1010，Y=12MHz, S1=1
tpw2b = 0b0001_1111，K=31
tpw2s = 0b1000_00000，S2=0
➔ 输出频率 = 12MHz ÷ (64 × 1 × (0+1)) = 187.5KHz
➔ 输出占空比 = [(31+1) ÷ 64] × 100% = 50%

例 2：

tpw2c = 0b0100_1110, Y=12MHz, S1=64
tpw2b = 0b0001_1111, K = 31
tpw2s = 0b010_11111, S2=31
➔ 输出频率= 12MHz ÷ (64 × 64 × (31+1)) = 91.55 Hz
➔ 输出占空比 = [(31+1) ÷ 64] × 100% = 50%

例 3：

tpw2c = 0b0100_0010, Y=12MHz, S1=1
tpw2b = 0b0011_1111, K=63
tpw2s = 0b010_00000, S2=0
➔ PWM 输出是高电平
➔ 输出占空比 = [(63+1) ÷ 64] × 100% = 100%

例 4：

tpw2c = 0b0100_0010, Y=12MHz, S1=1

tpw2b = 0b0000_0000，K=0

tpw2s = 0b010_00000, S2=0

➔ 输出频率= 12MHz ÷ (64 × 1 × (0+1)) = 187.5KHz

➔ 输出占空比 = [(0+1) ÷ 64] × 100% =1.5%

5.6.4 PWM 波形

PWM 波形（图 5.6）有一个时基（T_{Period} =时间周期）和一个周期里输出高的时间（占空比）。PWM 的频率取决于时基(f_{PWM} = 1/T_{Period})。

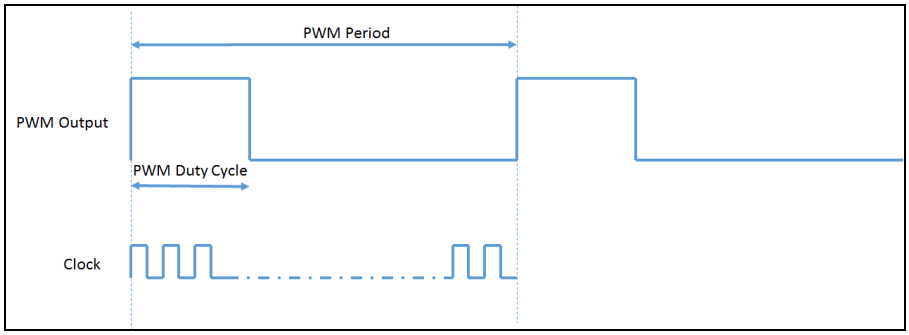


图 5.6：PWM 输出波形

5.7. 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 ***rstc*** 寄存器的选择，可以设定四种不同的看门狗超时时间，它是：

- 当 rstc[1:0]=00（默认）时：8k ILRC 时钟周期
- 当 rstc[1:0]=01 时：16k ILRC 时钟周期
- 当 rstc[1:0]=10 时：64k ILRC 时钟周期
- 当 rstc[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，PUD310 将复位并重新运行程序。看门狗时序图如图 5.7 所示。

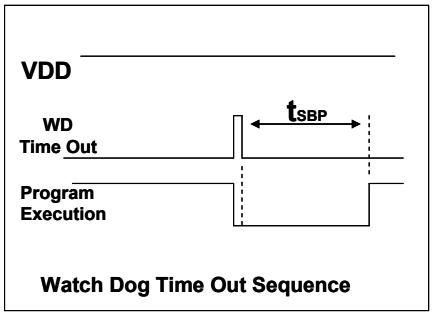


图 5.7：看门狗超时溢出时序图

5.8. 中断

PUD310 有 8 个中断源：

- 外部中断源 PA0
- Timer16 中断源
- PD_PHY 中断源
- TPW2 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 5.8 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。中断请求标志被设置时（仅外部 PA0 和 Timer 16 中断），根据寄存器 ***integs*** 的设置，可以是上升沿、下降沿或两者兼有。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。

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

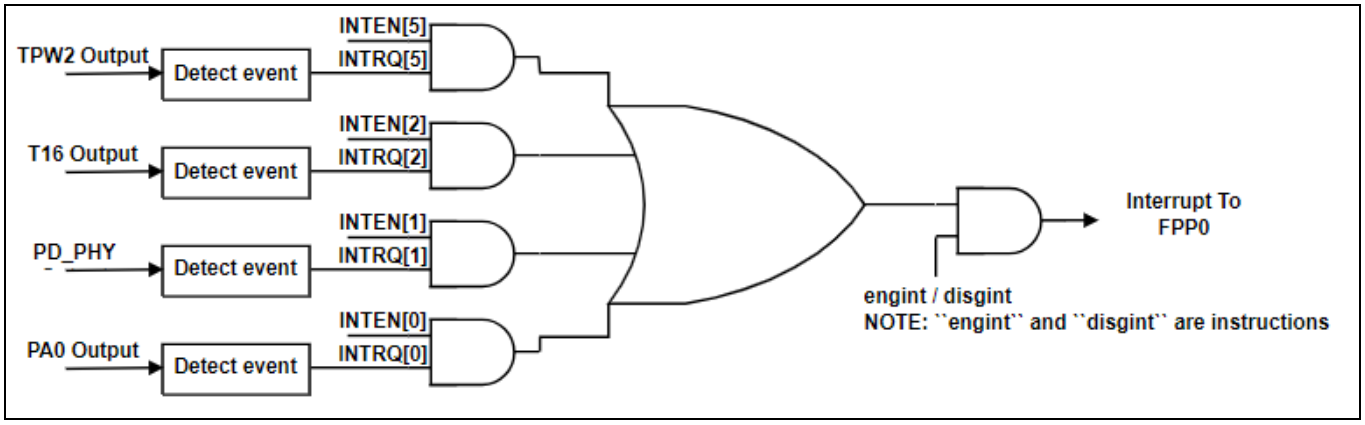


图 5.8：中断控制器硬件框图

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。依此类推，另外 ***pushaf*** 也需要两个字节。下面的示例程序演示了如何处理中断，请注意，处理一级中断和 ***pushaf*** 总共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
}
```

5.9. 省电与掉电

PUD310 有三个由硬件定义的操作模式，分别为：正常工作模式，电源省电模式和掉电模式。正常工作模式是所有功能都正常运行的状态，省电模式(***stopexe***)是在降低工作电流而且 CPU 保持在随时可以继续工作的状态，掉电模式(***stopsys***)是用来深度的节省电力。因此，省电模式适合在偶尔需要唤醒的系统工作，掉电模式是在非常低消耗功率且很少需要唤醒的系统中使用。表 5.4 显示省电模式(***stopexe***)和掉电模式(***stopsys***)之间在振荡器模块的差异，没改变就是维持原状态。

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

表 5.4：省电模式和掉电模式在振荡器模块的差异

5.9.1 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。***stopexe*** 的省电模式下，唤醒源可以是 IO 的切换，或者 Timer16 计数到设定值时（假如 Timer16 的时钟源是 IHRC 或者 ILRC）。系统唤醒后，单片机将继续正常的运行，省电模式的详细信息如下所示：

- IHRC 振荡器模块：没改变，如果被启用，则仍然保持运行状态。
- ILRC 振荡器模块：必须保持启用，唤醒时需要靠 ILRC 启动。
- 系统时钟：停用，因此 CPU 停止运行。
- OTP 存储器关闭。
- 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 TM16，TPW2）；
- 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，它在上升沿和下降沿都会被唤醒。

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

...
```

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

5.9.2 掉电模式 (“*stopsys*”)

掉电模式是深度省电的状态，所有的振荡器模块都会被关闭。通过使用“***stopsys***” 指令，芯片会直接进入掉电模式。下面显示发出 ***stopsys*** 命令后，PUD310 内部详细的状态：

- 所有的振荡器模块被关闭。
- OTP 存储器被关闭。
- SRAM 和寄存器内容保持不变。
- 唤醒源：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   =   0x22;       //  系统时钟从 IHRC 变为 ILRC，关闭看门狗时钟
CLKMD.1 =   0;          //  IHRC 停用
...
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   =   0x62;       //  系统时钟从 ILRC 变为 IHRC/2
```

5.9.3 唤醒

进入掉电或省电模式后，PUD310 可以通过切换 IO 引脚恢复正常工作；而 Timer16、Timer2 的唤醒只适用于省电模式。表 5.5 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	计时器中断
STOPSYS	是	否
STOPEXE	是	是

表 5.5：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 PUD310，IO 引脚必须大于一个操作指令，***pxdier*** 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 1024 个 ILRC 时钟周期，另外，PUD310 提供快速唤醒功能，通过 ***rstc*** 寄存器选择快速唤醒大约 16 个 ILRC 时钟周期。有关详细的时间参数，请参考表 5.6。

模式	唤醒模式	切换 IO 引脚的唤醒时间(t_{WUP})
STOPEXE 省电模式 或 STOPSYS 掉电模式	快速唤醒	16 * T_{ILRC}. 这里的 T_{ILRC} 是指 ILRC 时钟周期
STOPEXE 省电模式 或 STOPSYS 掉电模式	正常唤醒	1024* T_{ILRC}， 这里的 T_{ILRC} 是指 ILRC 时钟周期

表 5.6：掉电模式和省电模式在唤醒时间的差异

5.10. IO 引脚

PUD310 所有 IO 引脚都可以设定成输入或输出，通过数据寄存器 (***pa, pb***)，控制寄存器 (***pac, pbc***) 和弱上拉电阻 (***paph, pbph***) 设定，每一 IO 引脚都可以独立配置成不同的功能；所有这些引脚设置有施密特触发输入缓冲器和 CMOS 输出驱动电位水平。当这些引脚为输出低电位时，弱上拉电阻会自动关闭。如果要读取端口上的电位状态，一定要先设置成输入模式；在输出模式下，读取到的数据是数据寄存器的值。表 5.7 为端口 PA0 位的设定配置表。图 5.9 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 5.7： PA0 设定配置表

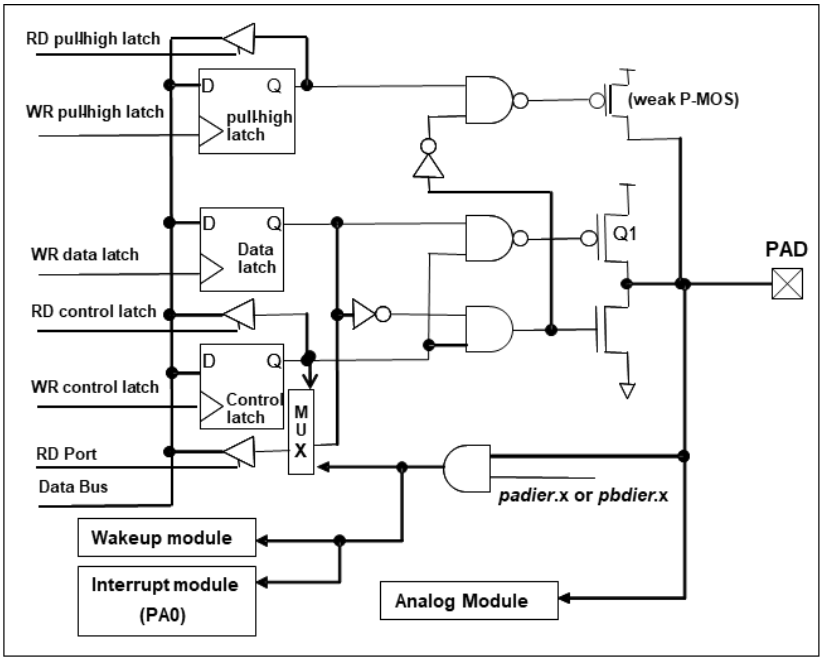


图 5.9：IO 引脚缓冲区硬件图

除了 PA7 之外，PA1、PA2、PA4、PA6 只有在设置为输出模式时才可以是开漏输出；PA1、PA2、PA4、PA6 是 PD 专用输入输出引脚。其他输入输出引脚具有相同的结构，对于那些被选择用作模拟功能的引脚，寄存器 ***padier*** / ***pbdier*** 相应位设置为低，以防止漏电流。当 PUD310 在掉电或省电模式，每一个引脚都可以切换其状态来唤醒系统。对于需用来唤醒系统的引脚，必须设置为输入模式以及寄存器 ***padier*** / ***pbdier*** 相应为高。同理，当 PA0 用作外部中断引脚时，***padier***.0 应设置为高电平，对于 PB0 则是 ***pbdier***.0，对于 PA3 则是 ***padier***.3。

5.11.复位和 LVR

5.11.1 复位

引起 PUD310 复位的原因很多，一旦复位发生，PUD310 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 V_{DD} 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 V_{DD} 小于 V_{DR}，数据存储器的值是在不确定的状态，然而，

发生上电复位或 LVR 复位后，若 V_{DD} 大于 V_{DR}（数据保存电压），数据存储器的值将会被保留；若 V_{DD} 小于 V_{DR}，数据存储器的值将转为未知的状态。发生复位，且程序中有加清除 SRAM 的指令或语法，则先前的数据将会在程序初始化中被清除，无法保留。

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。

5.11.2 LVR 复位

程序编译时，用户可以选择不同级别的 LVR。通常情况下，使用者在选择 LVR 复位水平时，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。

5.12. PD PHY 控制器

5.12.1 特性

PD PHY 控制器是具有 USB PD 功能的汇接端设备，负责在 CC 线上发送和接收信息。其主要功能是执行 PD 物理层信号通信，并为系统软件提供一个简单的访问接口，使用户可以对协议层进行最有效的管理。此外，该功能块还负责模拟前端（AFE）的所有信号控制，详见第 5.13 节。

- 硬件特性:
  - 用于寄存器访问的标准 MCU 系统总线接口。
  - 一个 PD PHY 系统中断，包括 TX 中断、RX 中断和 RX FIFO 半满中断。
  - 一个共享的 4 字节 TX/RX FIFO。
  - 通过 GPIO 支持内部信号监控功能，用于软件协议层开发。
- USB PD 物理层特性:
  - 只发送和接收两种 PD 信号：SOP 数据包和硬复位信令。控制器会自动省略 SOP'、SOP'' 和其他信令，从而减少物理层和协议层之间不必要的数据流。
  - 针对特定电缆和连接器应用调整接收数据能力。

5.12.2 方框图

下图 5.10 展示了 PD PHY 控制器和模拟前端 (AFE) 宏的框图。该控制器由发送和接收数据流控制器（TX/RX CTRL）、数据编码器和发送器（Transmitter）、数据解码器和接收器（Receiver）、共享 TX/RX CRC 生成器（CRC32）、共享 TX/RX 数据存储单元（TX/RX FIFO）和寄存器文件（Regfile）组成。发送数据路径（TX 数据路径）由发送器、CRC32 和 TX/RX FIFO 组成，而接收数据路径（RX 数据路径）由接收器、CRC32 和 TX/RX FIFO 组成。需要注意的是，CRC32 和 TX/RX FIFO 由 TX 和 RX 数据路径共享，因此数据传输和数据接收不能同时进行。

PD PHY 控制器和 AFE 宏的所有相关寄存器都分配在 Regfile 中。Regfile 提供一个标准 MCU 系统总线接口作为寄存器访问。Regfile 中的 AFE 寄存器用于管理 AFE 宏的所有操作。TX/RX CTRL 控制 TX/RX 数据流方向，处理所有协议错误，并与负责管理协议层所有事务的应用软件通信。TX/RX FIFO 在 TX 运行模式下保存要传输的数据，在 RX 运行模式下存储接收到的数据。TX/RX FIFO 是一个 4 字节的存储单元。在 RX 运行模式下，当 TX/RX FIFO 中存储了≧2 个字节时，将触发一个半满中断。

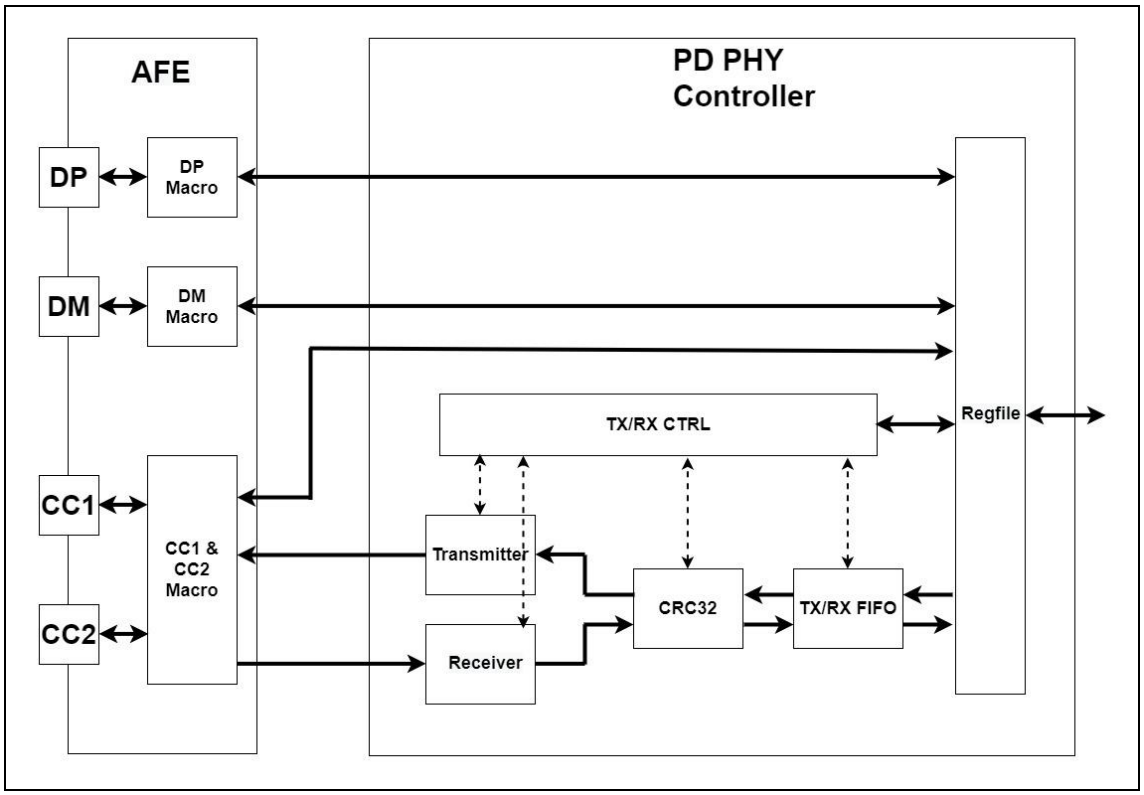


图 5.10：PD PHY 控制器和 AFE 框图

对于每个 SOP 数据包的传输，发射机首先会自动生成一个 64 位的前导码，然后从 TX/RX FIFO 和 CRC32 中读取数据并执行 4b5b 编码。发射机还会生成 SOP 和 EOP 符号，这些符号将被预先添加或附加到这些编码数据中。最后，在发送 SOP 数据包之前进行 BMC 编码。此外，发送器还可以生成 USB PD 规范中描述的硬复位信令。当成功执行一次 SOP 传输（包括硬复位信令）时，将触发一个 TX 中断。TX 中断必须由用户清除，详情请参阅第 5.12.3.4 节。

当 CC 线上的任何信号发生变化（从总线空闲变为总线活动）时，接收器将开始尝试进行 BMC 解码。用户也可以通过读取指定的寄存器来获取总线处于空闲或活动状态的信息。

如果检测到一个合法的 SOP 数据包，则在 4b5b 解码后，所有数据负载和 32 位 CRC 校验将被传送至 CRC32，并存储在 TX/RX FIFO 中。当 FIFO 中存储了两个字节的数据时，将触发第一次 RX 半满中断。每当发生 RX 半满中断时，用户必须尽快读取这些接收到的数据；否则，如果发生 FIFO 溢出，该正确的数据包将被视为一次失败的传输。

一旦首次 RX 半满中断被触发，当 CC 线重新回到总线空闲状态时，还会产生另一个 RX 中断。无论接收到的数据是否正确，用户都必须完成该数据包的处理流程。

此外，如果成功接收到一次硬复位（Hard Reset）信号，也会触发一次 RX 中断。

需要注意的是，接收器会自动忽略 SOP'、SOP" 数据包及其他信号（包括无意义信号或噪声），并不会向应用软件发送任何消息或中断。

5.12.3 功能描述

本节介绍所有 PD PHY 控制器功能和 TX/RX 操作步骤。还将提及一些有用信息。

5.12.3.1 TX/RX 硬件状态图

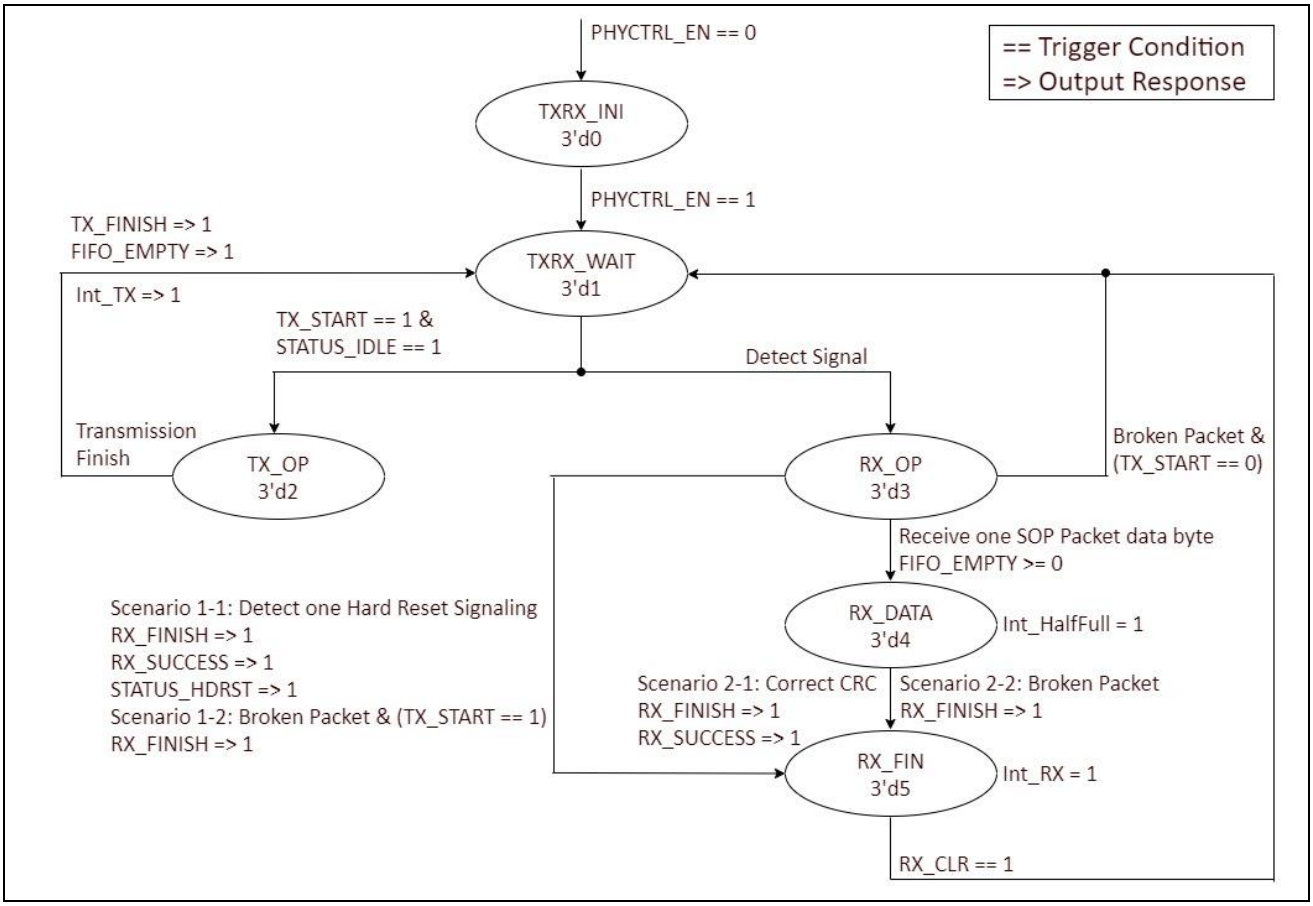


图 5.11：TX 和 RX 硬件状态图

运行模式	信号类型	注释
TX 运行模式	SOP 数据包	
TX 运行模式	Hard Reset singling	
RX 运行模式	SOP 数据包	
RX 运行模式	Hard Reset singling	

表 5.8：可接受的数据包和信令类型

图 5.11 展示了 PD PHY 控制器在发送或接收数据时的硬件行为。表 5.8 介绍了在控制器控制下可接受或操作的信号类型。

在开始任何操作之前，用户必须将 pd_txrx 寄存器的 PHYCTRL_EN 设置为 1 以启用控制器。将 PHYCTRL_EN 设置为 0 将强制控制器从任何其他状态（包括 PD PHY 中断和 pd_stat 寄存器中的所有状态标志）恢复到默认状态。它还可用作控制器的软复位。将 pd_txrx 寄存器的 TX_START 设置为 1，可启动一次新的 SOP 数据包传输，包括一次硬重置传输。传输完成后，将自动触发 Int_TX 中断。如果传输成功，pd_stat 的 TX_FINISH 和 FIFO_EMPTY 都将置 1。启用 TX_START 后，控制器极有可能同时检测 CC 线上的一个数据包。基本上，在硬件算法中，RX 操作的优先级高于 TX 操作，因此这次传输将被自动省略，TX_FINISH 也永远不会处于激活状态，也就是说，可能会发生一次意外的 TX/RX 碰撞。这种异常可能来自 PD 协议错误；用户必须提供适当的算法来解决这个问题（详情请参见第 5.12.3.7 节）。

当 CC 线上出现信号时，控制器会尝试检测并分析接收到的数据。PD PHY 控制器仅接受 SOP 数据包和 Hard Reset 信号，其他类型的信号将被视为无效信号并自动忽略。若成功检测到 Hard Reset 信号，将触发 Int_RX 中断，并将 pd_stat 寄存器中的 RX_FINISH、RX_SUCCESS 和 STATUS_HDRST 置为 1。若检测到 SOP 数据包，其数据载荷和 32 位 CRC 校验将被存储至 FIFO 中。当 FIFO 中接收到并存储了两个字节的数据后，将触发第一次 Int_HalfFull 中断。中断将持续直到通过读取 pd_fifo 寄存器使 FIFO 中少于两个字节为止。此后每次 FIFO 中存在两个字节（包括第 3 和第 4 字节）时，Int_HalfFull 中断会再次触发。当检测到数据包的 EOP 符号或无效数据时，接收过程将停止，同时触发 Int_RX 中断，并设置 RX_FINISH 位。RX_SUCCESS 位可用于判断数据包是否成功接收。接收完成后，用户必须将 pd_txrx 寄存器中的 RX_CLR 置位，使控制器恢复至初始状态，从而可以继续进行下一次 TX 或 RX 操作。

若用户希望监测图 5.11 中控制器的状态转换及内部信号，也可以通过 GPIO 引脚进行探测（详见第 5.12.3.9 节）。

5.12.3.2 TX 流量控制

图 5.12 描述了一个标准 SOP 数据包传输的基本流程图，图 5.13 描述了如何单独传输一个硬复位信号。(注：如果用户希望针对特定应用制定自己的策略，则不必遵循这些流程图）。

读取 pd_stat 寄存器的 STATUS_IDLE，以检查 CC 线是否处于总线空闲状态，从而避免同时发送和接收数据时发生碰撞。启用 pd_txrx 寄存器的 TX_START，开始一次新的传输，同时控制器也开始在 CC 线上自动生成一个 64 位的前导码。整个前导码传输时间足以为软件准备传输数据。将 SOP 数据包的有效载荷写入 pd_fifo 寄存器，并检查 pd_stat 寄存器的 FIFO_FULL，以决定下一次数据写入。在整个数据传输过程中，控制器始终通过 pd_stat 寄存器的 FIFO_EMPTY 监控 FIFO 状态。一旦发现 FIFO 为空，控制器就会认为没有数据要再次传输。传输完成后，将触发 Int_TX 中断。检查 pd_stat 寄存器中的 TX_FINISH，以确保数据传输已成功执行，且未出现异常。最后，设置 pd_txrx 寄存器的 TX_CLR，清除所有硬件状态和 PD PHY 中断。

硬复位信令传输的流程与 SOP 数据包完全相同，只是不需要传输数据有效载荷，而且必须先设置 pd_txrx 寄存器的 SEND_HDRST，然后再设置 TX_START。

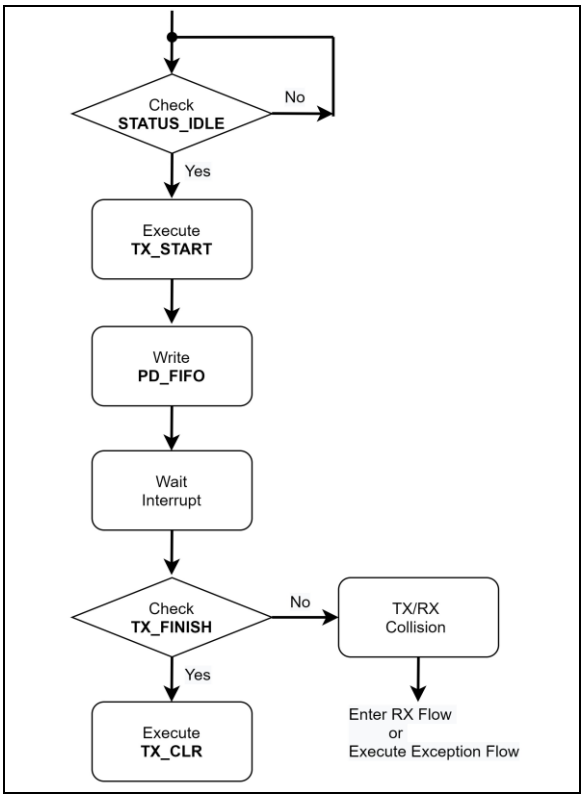


图 5.12: TX 流程图（SOP 数据包）

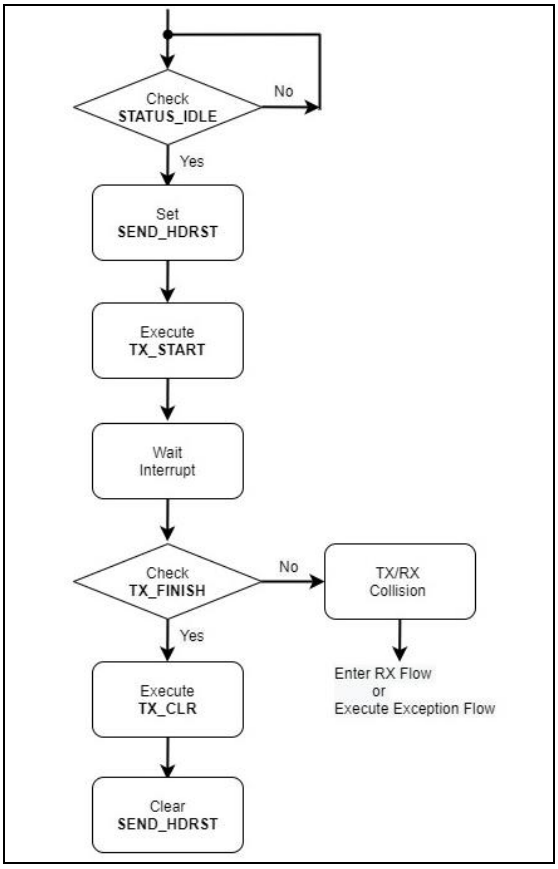


图 5.13: TX 流程图（硬复位信号）

5.12.3.3 RX 流量控制

图 5.14 如下所示，其展示了标准操作程序（SOP）数据包或硬复位信令接收的流程图。（注意：如果用户想要针对特定应用制定自己的策略，则无需遵循此流程图。）如果出现第一个 PD PHY 控制器中断，用户必须检查 pd_stat 寄存器 RX_FINISH 位，以确定它是 Int_RX 中断还是 Int_HalfFull 中断。如果 RX_FINISH 设置为 1，那么它必定是 Int_RX 中断，否则发生的是 Int_HalfFull 中断。如果这是 Int_RX 中断类型，那么读取 pd_stat 寄存器的 RX_SUCCESS 位，检查它是否设置为 1。如果 RX_SUCCESS 设置为 1，接收到的数据必定是一个硬复位信令，否则它被视为一个损坏的数据包或信令，且无需采取任何操作。此外，检查 pd_stat 寄存器的 STATUS_HDRST 位，以确认接收到的数据是硬复位信令。 如果第一个中断是 Int_HalfFull 中断，那么接收到的必定是一个 SOP 数据包。从 pd_fifo 寄存器读取接收到的数据，直到 pd_stat 寄存器的 FIFO_EMPTY 位恢复为默认值。等待下一个中断发生，检查 RX_FINISH 标志位，并再次从 FIFO 中读取接收到的数据。一旦控制器检测到最终的 EOP（end-of-packet，数据包结束）符号或损坏的数据，Int_RX 中断将被触发，且 RX_FINISH 被设置为 1。检查 RX_SUCCESS 位，以确定是否接收到了正确的数据包。最后，读取 FIFO 中的剩余数据，并将 pd_txrx 寄存器的 RX_CLR 位设置为 1，以清除 PD PHY 控制器中断和所有状态标志位。

表 5.9 描述了状态标志与接收信号类型之间的关系。根据该表的内容，可以很容易地判断控制器正在接收哪个数据包或信令。表 5.10 介绍了如何清除 PD PHY 中断源，表 5.11 显示了 pd_stat 寄存器中所有状态标志的清除方法。

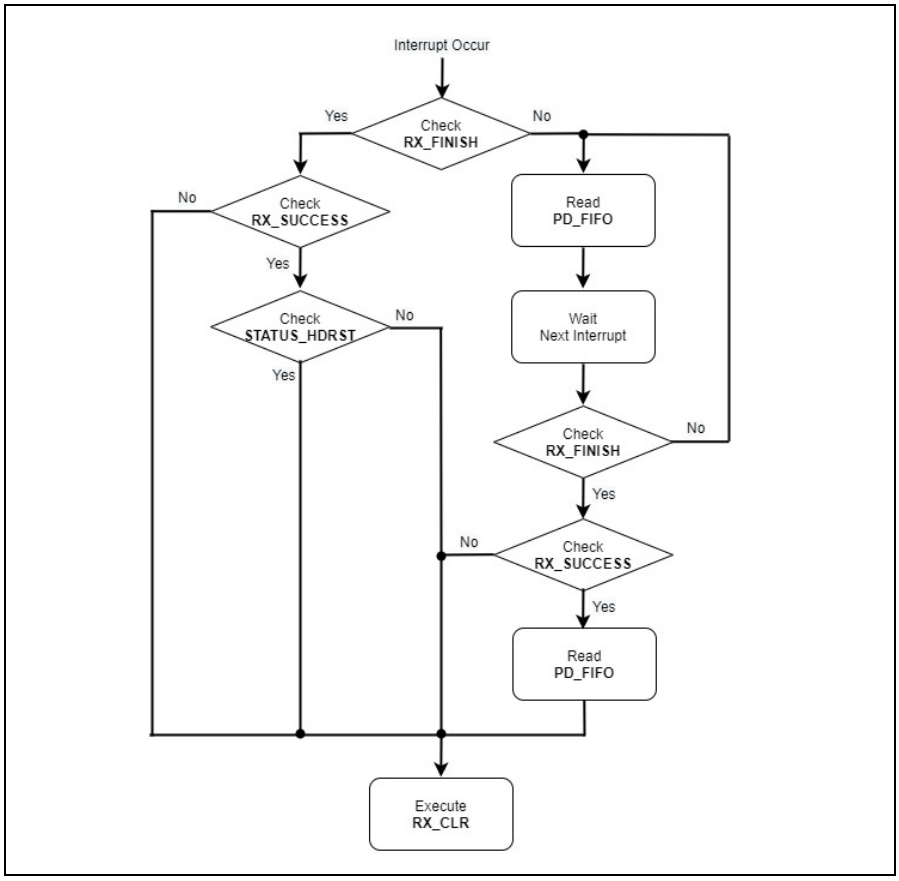


图. 5.14: RX 流程图

	RX_FINISH	RX_SUCCESS	STATUS_HDRST
SOP Packet	1	1	0
Hard Reset Singling	1	1	1
Broken Packet	1	0	0

表 5.9：状态标志与信号类型之间的关系

	PHYCTRL_EN = 0	TX_CLR = 1	RX_CLR = 1	Read FIFO
Int_TX	清除	清除	清除	N.A.
Int_RX	清除	清除	清除	N.A.
Int_HalfFull	清除	N.A.	清除	清除

表 5.10：PD PHY 中断清除方法

	PHYCTRL_EN = 0	TX_CLR = 1	RX_CLR = 1	Read FIFO
TX_FINISH	清除	清除	N.A.	N.A.
RX_FINISH	清除	N.A.	清除	N.A.
RX_SUCCESS	清除	N.A.	清除	N.A.
FIFO_EMPTY	清除	N.A.	清除	清除

	PHYCTRL_EN = 0	TX_CLR = 1	RX_CLR = 1	Read FIFO
FIFO_FULL	清除	N.A.	清除	清除
STATUS_HDRST	清除	N.A.	清除	N.A.
STATUS_IDLE	清除	N.A.	N.A.	N.A.

表 5.11：PD PHY 状态标记清除方法

5.12.3.4 PD PHY 中断类型

PD PHY 中断由三个独立的中断源组成，采用电平触发模式，如图 5.15 所示。系统中断使能由 inten 寄存器的 bit1 定义。系统中断状态在寄存器 intrq 的 bit1 中定义。中断重置方法如上表 5.10 所示。需要注意的是，为了减少软件操作的复杂性，Int_TX、Int_RX 和 Int_HalfFull 没有相应的标志。

PD PHY 中断 = Int_TX | Int_RX | Int_HalfFull：

- Int_TX：当 TX 发送了一个数据包或信令时将处于活动状态。
- Int_RX：当 RX 接收到一个合法数据包或信令，或数据包数据中断时将处于活动状态。
- Int_HalfFull：在 RX 运行模式下，当 FIFO 收到 2/3/4 字节时将激活。

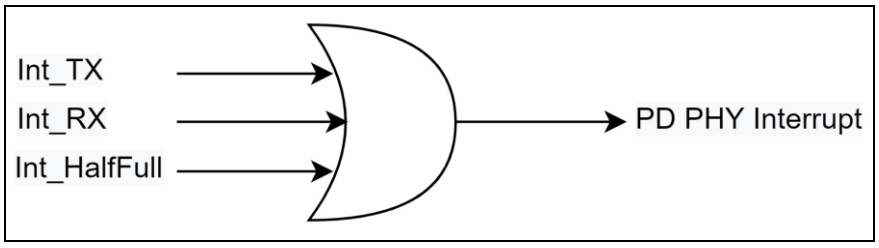


图 5.15：PD PHY 中断类型

5.12.3.5 TX/RX FIFO 存取

以下是 TX/RX FIFO 的一些特性：

1. 一个固定的 4 字节 FIFO。

2. FIFO 由 TX 和 RX 数据通道共享，因此不允许同时发送和接收数据。

3. 在 TX 运行模式下，FIFO 只允许用户写入；在 RX 运行模式下，FIFO 只允许用户读取。

4. 向 FIFO 写入一个字节后，必须先将 pd_txrx 寄存器的 FIFO_WR 设置为 1，然后再设置为 0（或者先设置为 0，然后再设置为 1，这是一种上升沿触发模式）。执行此操作可更新 FIFO 的写入索引。

5. 从 FIFO 读取一个字节后，必须先将 pd_txrx 寄存器的 FIFO_RD 设置为 1，然后再设置为 0（或者先设置 0，然后再设置 1。） 执行此操作可更新 FIFO 的读取索引。

6. Int_HalfFull 中断仅在 RX 运行模式下使用和触发。

7. 如果是正确的 SOP 数据包，接收数据的最后四个字节必须是 32 位 CRC 数据。

5.12.3.6 CRC32 特性

以下 CRC 信息来自 USB PD 规范。详情请参见相关文件。CRC-32 用于保护数据有效载荷的数据完整性。CRC-32 的定义如下：

- CRC-32 多项式应为 = 04C1 1DB7h
- CRC-32 初始值应为 = FFFF FFFFh
- CRC-32 应针对有效载荷的所有字节进行计算，不包括任何数据包成帧符号（即不包括 Preamble、SOP*、EOP）。
- CRC-32 计算应从字节 0 的第 0 位开始，一直计算到数据包每个字节的第 7 位。
- CRC-32 的剩余部分应进行补全。
- CRC-32 的余数应为 C704 DD7Bh

注：CRC-32 余数的这种反转增加了一个 FFFF FFFFh 的偏移量，将在接收端产生一个恒定的 CRC-32 余数 C704 DD7Bh。

5.12.3.7 TX 和 RX 碰撞过程

第 5.12.3.1 节提到 TX/RX 碰撞条件。在 TX 工作模式下，设置 TX_START 可开始一次新的传输；但控制器会省略这次传输，并强制将硬件状态切换到 RX 工作模式。出现这种情况时，主机（电源端）和设备（PD PHY 控制器）之间可能会失去通信同步，有时很难找到导致这种 PD 协议错误的根本原因。

USB PD 规范提出了许多解决方案来恢复电源端和汇流端之间的正常通信，但这需要协议层的辅助，即用户必须支持一种强大的软件算法来处理这种异常情况。事实上，如果发生意外错误，什么也不用做，只需等待主机发出硬复位信号。这可以使设备恢复到初始状态，并重新启动主机和设备之间的新通信。

5.12.3.8 接收信号质量调整

PD PHY 控制器的数据接收能力符合 USB PD 标准规范。如果用户希望针对特定应用调整接收数据的质量。控制器支持 pd_decoder 寄存器的 TH_DEB、TH_HIGH 和 TH_LOW，以满足这些要求。

可以使用 TH_DEB 将尖峰噪声等干扰消除。接收器使用的 BMC 解码原理是检测高/低电压的持续时间，以决定“长一/零”或“短一/零”。这种“长”或“短”的判断取决于 TH_HIGH 和 TH_LOW 的阈值。需要注意的是，这些值设置不当会导致 BMC 解码错误。TH_DEB、TH_HIGH 和 TH_LOW 的默认值通常已足够好，无需进一步调整。

5.12.3.9 内置硬件信号探测

第 5.12.3.1 节中的图 5.11 展示了控制器硬件行为的状态转换和相应标志的变化。这些信息有助于找到数据通信错误的根本原因，或用于开发协议层算法。这些实时信号可以通过 GPIO 进行探测。pd_ctrl 寄存器的 PROBE_SEL 和 opr1 寄存器的第 6:4 位均用于选择要监控的信号。表 5.12 显示了探测信号、寄存器设置值和相应输出 GPIO 引脚之间的关系。表 5.13 解释了这些探测信号的定义和功能。用大写字母拼写的信号代表状态寄存器名称，这些信号的说明在 pd_stat 和 cc_misc 寄存器中定义。st_phy[2:0]信号的状态名称和编号定义如下表 5.14 所示。

Bit 6:4 of opr1 = 001	PB0	PA5	PA3	PA0
PROBE_SEL = 00	int_pd_phy	st_phy[2]	st_phy[1]	st_phy[0]
PROBE_SEL = 01	fifo_ov	STATUS_HDRST	RX_SUCCESS	RX_FINISH
PROBE_SEL = 10	FIFO_FULL	FIFO_EMPTY	TX_FINISH	STATUS_IDLE
PROBE_SEL = 11	crc_cmp_out	sym_eop	pktdet_fail	BMC_BIT

表 5.12：PD PHY 控制器的探头信号

Probe Signal	描述
st_phy[2:0]	图 5.11：TX 和 RX 硬件状态图中的状态机名称
int_pd_phy	PD PHY 控制器中断
fifo_ov	FIFO 超限
crc_cmp_out	32 位 CRC 校验输出；1：接收数据无误，0：CRC 比较错误
sym_eop	查找 EOP 符号
pktdet_fail	接收到的数据无法正确解码。

表 5.13：探头信号定义

st_phy[2:0]	0	1	2	3	4	5
State Name	TXRX_INI	TXRX_WAIT	TX_OP	RX_OP	RX_DATA	RX_FIN

表 5.14：st_phy[2:0] 的状态名称和编号

5.12.4 编程序列

本节介绍如何执行正确的寄存器设置，以控制 PD PHY 控制器发送或接收数据。这些示例可用作基本编程顺序，但用户也可根据具体应用开发自己的程序代码，以更有效地操作控制器。

5.12.4.1 发送一个 SOP 数据包

下面的编程序列可传输一个带中断的 SOP 数据包：

步骤 1 设置 pd_txrx 寄存器的 PHYCTRL_EN，启用 PD PHY 控制器。

步骤 2 循环等待 CC 线空闲，直至 pd_stat 寄存器的 STATUS_IDLE 变为 1。

步骤 3 设置 pd_txrx 寄存器的 TX_START，开始新的传输。

步骤 4 循环等待 FIFO 未满，直到 pd_stat 寄存器的 FIFO_FULL 变成 0。

步骤 5 向 pd_fifo 寄存器写入一个字节的数据。

步骤 6 先向 pd_txrx 寄存器的 FIFO_WR 设置“1”，然后设置“0”，以更新 FIFO 的写入索引。

步骤 7 如果有更多数据要发送，则转到步骤 4。

步骤 8 等待 PD PHY 中断发生。

步骤 9 检查 pd_stat 寄存器的 TX_FINISH：

1. TX_FINISH = 1：传输成功。
2. TX_FINISH = 0：本次传输自动省略。

步骤 10 设置 pd_txrx 寄存器的 TX_CLR，清除 PD PHY 中断和所有硬件状态。

5.12.4.2 发送一次硬重置信号

下面的编程顺序将发送一个带中断的硬复位信号：

步骤 1 设置 pd_txrx 寄存器的 PHYCTRL_EN，启用 PD PHY 控制器。

步骤 2 循环等待 CC 线空闲，直至 pd_stat 寄存器的 STATUS_IDLE 变为 1。

步骤 3 设置 pd_txrx 寄存器的 SEND_HDRST，启用硬重置传输。

步骤 4 设置 pd_txrx 寄存器的 TX_START，开始新的传输。

步骤 5 等待 PD PHY 中断发生。

步骤 6 检查 pd_stat 寄存器的 TX_FINISH：

1. TX_FINISH = 1：传输成功。
2. TX_FINISH = 0：本次传输自动省略。

步骤 7 设置 pd_txrx 寄存器的 TX_CLR，清除 PD PHY 中断和所有硬件状态。

步骤 8 清除 pd_txrx 寄存器的 SEND_HDRST，禁用硬重置传输。

5.12.4.3 接收一个 SOP 数据包

以下编程顺序接收一个带中断的 SOP 数据包：
步骤 1 设置 pd_txrx 寄存器的 PHYCTRL_EN，启用 PD PHY 控制器。
步骤 2 等待 PD PHY 中断发生。
步骤 3 从 pd_fifo 寄存器中读取一个字节的数据。
步骤 4 先向 pd_txrx 寄存器的 FIFO_RD 设置“1”，然后设置“0”，以更新 FIFO 的读取索引。
步骤 5 如果 pd_stat 寄存器的 FIFO_EMPTY 不是“1”，则转到步骤 3 读取更多的接收数据。
(注：最后四个字节为 32 位 CRC 数据）
步骤 6 如果 pd_stat 寄存器的 RX_FINISH 不是 “1”，则转到步骤 2 等待下一次中断。
步骤 7 检查 pd_stat 寄存器的 RX_SUCCESS：
1. RX_SUCCESS = 1：接收到一个合法数据包。
2. RX_SUCCESS = 0：接收到一个损坏的数据包。
步骤 8 设置 pd_txrx 寄存器的 RX_CLR，清除 PD PHY 中断和所有硬件状态。

5.12.4.4 接收一次硬复位信令

以下编程顺序可接收一次带中断的硬复位：
步骤 1 设置 pd_txrx 寄存器的 PHYCTRL_EN，启用 PD PHY 控制器。
步骤 2 等待 PD PHY 中断发生。
步骤 3 检查 pd_stat 寄存器的 STATUS_HDRST：
1. STATUS_HDRST = 1：接收一次硬复位信令。
2. STATUS_HDRST = 0：接收其他信号。
步骤 4 设置 pd_txrx 寄存器的 RX_CLR，清除 PD PHY 中断和所有硬件状态。

5.12.4.5 CC 线手动控制

以下编程序列描述了如何直接控制 CC 线：

在 CC 线上写入逻辑 0 或 1：
步骤 1：设置 ***pd_ctrl*** 寄存器的 ***CC_SEL*** 位，以获取对 CC 线的控制权。
步骤 2：设置 ***cc_misc*** 寄存器的 ***BMCTX_EN*** 位，以开启发送（TX）功能。
步骤 3：在 ***cc_misc*** 寄存器的 ***BMCTX_DOUT*** 位中设置想要发送的数据。

在 CC 线上读取逻辑 0 或 1：
步骤 1：设置 ***pd_ctrl*** 寄存器的 ***CC_SEL*** 位，以获取对 CC 线的控制权。
步骤 2：设置 ***cc_misc*** 寄存器的 ***BMCRX_EN*** 位，以开启接收（RX）功能。
步骤 3：读取 ***cc_misc*** 寄存器的 ***BMC_BIT*** 位，以获取接收到的数据。

5.13. 模拟前端（DP/DM/CC1/CC2）

模拟前端（AFE）是适用于通用快速充电规范应用的模拟宏单元。模拟前端由一个 DP 宏单元、一个 DM 宏单元、一个 CC 宏单元、一个连接到 PD PHY 控制器的控制接口，以及一个通过 DP、DM、CC1 和 CC2 引脚实现的 USB 通信接口组成。DP 宏单元和 DM 宏单元是针对 DP 引脚和 DM 引脚的两个功能相同的独立电路，而 CC 宏单元是 CC1 引脚和 CC2 引脚共用的硬件部分。图 5.16 展示了模拟前端（AFE）宏单元的方框图。

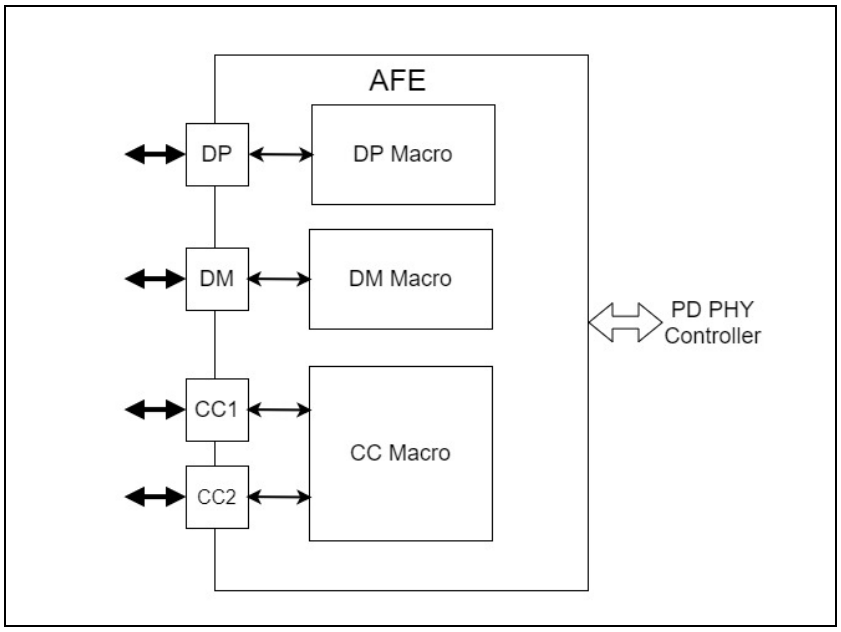


图 5.16: 模拟前端 (AFE) 框图

5.13.1 特性

- DP 和 DM 宏单元特性：
  - 支持四种不同的驱动电压电平：5V、3.3V、0.6V 和 0V
  - 支持一种可选的电压比较器：3V、2.4V、1.2V 和 0.35V
  - 在测试模式下，DP 和 DM 引脚可配置为一次性可编程（OTP）下载电缆引脚，其功能与 5V 通用输入输出（GPIO）引脚相同。
  - DP 和 DM 引脚也可通过通用输入输出（GPIO）控制模式进行操作。
- CC 宏单元特性：
  - 支持用作 USB 电力传输（PD）通信数据传输的发射器和接收器。发射器在 CC 线路上提供精确的 1.125V 驱动电压。
  - 支持用于 USB Type-C 5V/1.5A 检测的 0.66V 比较器。
  - 支持用于 USB Type-C 5V/3A 检测的 1.23V 比较器。
  - 支持两个独立的 0.2V 比较器，用作 CC1/CC2 连接检测。
  - 支持两个独立的 CC1/CC2 5.1k R_d 电阻。

5.13.2 功能描述

5.13.2.1 DP 和 DM 输出控制

根据各种充电协议，DP/DM 输出电压电平的要求会有所不同。模拟前端（AFE）的 DP/DM 宏单元提供可选择的驱动电压，以满足特定的应用需求。表 5.15 和表 5.16 描述了 dp_out 和 dm_out 寄存器的所有合法设置值以及相应的驱动电压。

注释 1：3.3V 和 0V 输出电压有两种设置配置，但用户可以选择其中任意一种作为设置值。

注释 2：dp_out 和 dm_out 寄存器的 DP_ISINK_EN 和 DM_ISINK_EN 用于控制一个下拉电阻，该下拉电阻在上电复位后提供一个初始值。在正常操作模式下，建议用户在系统启动完成后关闭此功能。

dp_dm_io_ctrl 的[1:0]位	00	00	00	00	00	00	00
DP_ISINK_EN	0	0	0	0	0	0	0
DP_BUFVREF_SEL	X	X	X	X	0	1	X
DP_BUF_EN	0	0	0	0	1	1	0
DP_TX_EN	0	0	1	1	0	0	0
DP_OUT_EN	1	1	0	0	0	0	0
DP_OUT	0	1	0	1	X	X	X
DP 驱动电压	0V	5V	0V	3.3V	0.6V	3.3V	Hi-Z

注释 1：在设置 dp_out 寄存器之前，必须先设置 dp_dm_io_ctrl 寄存器的[1:0]位。

注释 2：建议在 DP 正常运行时禁用 DP_ISINK_EN。

表 5.15：DP 合法设定值

dp_dm_io_ctrl 的[3:2]位	00	00	00	00	00	00	00
DM_ISINK_EN	0	0	0	0	0	0	0
DM_BUFVREF_SEL	X	X	X	X	0	1	X
DM_BUF_EN	0	0	0	0	1	1	0
DM_TX_EN	0	0	1	1	0	0	0
DM_OUT_EN	1	1	0	0	0	0	0
DM_OUT	0	1	0	1	X	X	X
DM 驱动电压	0V	5V	0V	3.3V	0.6V	3.3V	Hi-Z

注释 1：在设置 dm_out 寄存器之前，必须先设置 dp_dm_io_ctrl 寄存器的 [3:2] 位。

注释 2：建议在 DM 正常运行时禁用 DM_ISINK_EN。

表 5.16: DM 合法设定值

5.13.2.2 DP 和 DM 输入控制

DP 和 DM 宏单元支持一个可选择的比较器，该比较器提供四级参考电压。表 5.17 展示了 dp_in 和 dm_in 寄存器中的 DP_CMPVREF_SEL 和 DM_CMPVREF_SEL 的设置与比较器参考电压之间的关系。两个电压比较器的比较输出将分别在 dp_in 和 dm_in 寄存器的 DP_DIN 和 DM_DIN 中体现。dp_in 和 dm_in 寄存器中的 DP_CMP_EN 和 DM_CMP_EN 分别用于使能这两个比较器。

DP_CMPVREF_SEL DM_CMPVREF_SEL	00	01	10	11
比较器电压	0.35V	1.2V	2.4V	3V

表 5.17：可选参考电压比较器

5.13.2.3 DP 和 DM 通用输入输出控制模式

表 5.15 和表 5.16 中所示的用法可能并不适用于 DP 和 DM 引脚的操作。实际上，对于特定的充电协议而言，通常并非所有的驱动电压电平都是必需的。此外，控制这些相关寄存器的设置既不直观也不方便。AFE 宏单元还支持一种通用输入输出（GPIO）控制模式，以简化 DP 和 DM 的操作。DP 引脚将被切换到通用输入输出引脚 PA4，DM 引脚将被切换到通用输入输出引脚 PA6。dp_out、dp_in、dm_out 和 dm_in 寄存器的大多数字段可以重新映射到 pa、pac 和 padier 寄存器的相应字段。表 5.18 解释了 DP 和 PA4 寄存器字段之间的转换，表 5.19 解释了 DM 和 PA6 寄存器字段之间的转换。需要注意的是，在 dp_dm_io_ctrl 寄存器中定义了三种不同的转换类型。表 5.20 和表 5.21 描述了 DP/DM 状态与相应寄存器设置之间的关系。

Bit [1:0] of dp_dm_io_ctrl	DP	PA4
01	DP_ISINK_EN	N.A.
	DP_BUFVREF_SEL	pa[4]
	DP_BUF_EN	pac[4]
	DP_TX_EN	0
	DP_OUT_EN	0
	DP_OUT	0
	DP_CMPVREF_SEL	N.A.
	DP_CMP_EN	padier[4]
	DP_DIN	pa[4]

Bit [1:0] of dp_dm_io_ctrl	DP	PA4
10	DP_ISINK_EN	N.A.
	DP_BUFVREF_SEL	0
	DP_BUF_EN	0
	DP_TX_EN	pac[4]
	DP_OUT_EN	0
	DP_OUT	pa[4]
	DP_CMPVREF_SEL	N.A.
	DP_CMP_EN	padier[4]
	DP_DIN	pa[4]

Bit [1:0] of dp_dm_io_ctrl	DP	PA4
11	DP_ISINK_EN	N.A.
	DP_BUFVREF_SEL	0
	DP_BUF_EN	0
	DP_TX_EN	0
	DP_OUT_EN	pac[4]
	DP_OUT	pa[4]
	DP_CMPVREF_SEL	N.A.
	DP_CMP_EN	padier[4]
	DP_DIN	pa[4]

表 5.18：DP 与 PA4 之间的转换

Bit [3:2] of dp_dm_io_ctrl	DM	PA6
01	DM_ISINK_EN	N.A.
	DM_BUFVREF_SEL	pa[6]
	DM_BUF_EN	pac[6]
	DM_TX_EN	0
	DM_OUT_EN	0
	DM_OUT	0
	DM_CMPVREF_SEL	N.A.
	DM_CMP_EN	padier[6]
	DM_DIN	pa[6]

Bit [3:2] of dp_dm_io_ctrl	DM	PA6
10	DM_ISINK_EN	N.A.
	DM_BUFVREF_SEL	0
	DM_BUF_EN	0
	DM_TX_EN	pac[6]
	DM_OUT_EN	0
	DM_OUT	pa[6]
	DM_CMPVREF_SEL	N.A.
	DM_CMP_EN	padier[6]
	DM_DIN	pa[6]

Bit [3:2] of dp_dm_io_ctrl	DM	PA6
11	DM_ISINK_EN	N.A.
	DM_BUFVREF_SEL	0
	DM_BUF_EN	0
	DM_TX_EN	0
	DM_OUT_EN	pac[6]
	DM_OUT	pa[6]
	DM_CMPVREF_SEL	N.A.
	DM_CMP_EN	padier[6]
	DM_DIN	pa[6]

表 5.19: DM 和 PA6 之间的转换

DP Status	pa[4]	pac[4]	padier[4]
Bit [1:0] of dp_dm_io_ctrl = 01			
输出高 (3.3V)	1	1	X
输出低 (0.6V)	0	1	X
输入高 (> V_{th})	1	0	1
输入低 (< V_{th})	0	0	1
输出 Hi-Z	X	0	X
输入 Hi-Z	X	X	0
Bit [1:0] of dp_dm_io_ctrl = 10			
输出高 (3.3V)	1	1	X
输出低 (0V)	0	1	X
输入高 (> V_{th})	1	0	1
输入低 (< V_{th})	0	0	1
输出 Hi-Z	X	0	X
输入 Hi-Z	X	X	0
Bit [1:0] of dp_dm_io_ctrl = 11			
输出高 (5V)	1	1	X
输出低 (0V)	0	1	X
输入高 (> V_{th})	1	0	1
输入低 (< V_{th})	0	0	1
输出 Hi-Z	X	0	X
输入 Hi-Z	X	X	0

注：阈值电压（V_{th}）由 DP 比较器参考电压选择位（DP_CMPVREF_SEL）决定。

表 5.20：DP 状态与 PA4 寄存器设置之间的关系

DM Status	pa[6]	pac[6]	padier[6]
Bit [3:2] of dp_dm_io_ctrl = 01			
输出高 (3.3V)	1	1	X
输出低 (0.6V)	0	1	X
输入高 (> V_{th})	1	0	1
输入低 (< V_{th})	0	0	1
输出 Hi-Z	X	0	X
输入 Hi-Z	X	X	0
Bit [3:2] of dp_dm_io_ctrl = 10			
输出高 (3.3V)	1	1	X
输出低 (0V)	0	1	X
输入高 (> V_{th})	1	0	1
输入低 (< V_{th})	0	0	1
输出 Hi-Z	X	0	X
输入 Hi-Z	X	X	0
Bit [3:2] of dp_dm_io_ctrl = 11			
输出高 (5V)	1	1	X
输出低 (0V)	0	1	X
输入高 (> V_{th})	1	0	1
输入低 (< V_{th})	0	0	1
输出 Hi-Z	X	0	X
输入 Hi-Z	X	X	0

注：阈值电压（V_{th}）由 DM 比较器参考电压选择（DM_CMPVREF_SEL）决定

表 5.21：DM 状态与 PA6 寄存器设置之间的关系

5.13.2.4 CC1 和 CC2 控制

CC 宏为 USB PD 规范的应用提供了所有必要的功能。CC 宏包括一个 PD 数据流路径（BMC_TX 和 BMC_RX）、两个分别具有 0.66V 和 1.23V 参考电压的电压比较器（用于 USB Type - C 5V/1.5A 和 5V/3A 应用）、CC1/CC2 连接检测电路以及 Rd 下拉电阻。

5.13.2.4.1 USB PD 数据流路径

BMC_TX 和 BMC_RX 分别为发送器和接收器而设计，它们构成了 USB PD（通用串行总线功率传输）通信的数据流路径。BMC_TX 发送从 PD PHY 控制器发送的 BMC 编码比特流。在生产中，可以通过查找带隙基准电压的最佳微调值，来微调精确的 1.125V 输出电压。 BMC_RX 对 CC 线上的传入数据进行解码，并且这些解码后的数据将进一步传送到 PD PHY 控制器。BMCTX_EN 信号用于使能 BMC_TX 发送器，BMCTX_DOUT 信号用作发送的数据。BMCRX_EN 信号用于使能 BMC_RX 接收器，BMC_BIT 呈现 BMC_RX 的解码输出值。在正常操作模式下，上述所有信号都由 PD PHY 控制器进行控制。不过，这些信号也可以通过寄存器访问模式直接进行管理。cc_misc 寄存器中具有相同名称的所有字段都可用于访问这些信号。有关如何控制该寄存器的详细信息，请参阅 5.12.4.5 节中的相关编程序列。

5.13.2.4.2 USB Type - C 5V/1.5A 和 5V/3A 检测

在 CC 宏中有两个独立的电压比较器，用于 Type-C 5V/1.5A 和 5V/3A 的检测。一个参考电压为 0.66V 的比较器用于 5V/1.5A 的应用场景，另一个参考电压为 1.23V 的比较器用于 5V/3A 的应用场景。cc_stat 寄存器中的 CC_CMPSNKXA_EN 用于使能这两个比较器。cc_stat 寄存器中的 CC_SNK1P5A 和 CC_SNK3P0A 分别用于表示这两个比较器的输出状态。

5.13.2.4.3 CC1/CC2 连接检测

在 CC 宏中有两个独立的 CC 连接检测电路，分别对应 CC1 和 CC2 引脚。每个检测电路都由一个参考电压为 0.2V 的比较器和一个用于对电压比较器的输出值进行采样并存储的采样锁存器组成。cc_ctrl 寄存器中的 CC_CMPRA_EN 用于使能这两个比较器。并且，cc_ctrl 寄存器中的 CC1_ATT 和 CC2_ATT 用于表示这两个采样锁存器各自的输出状态。锁存器的采样操作由 cc_ctrl 寄存器中的 CC_CMPRA_LAT 定义，而锁存器的复位控制由 cc_ctrl 寄存器中的 CC_REGRA_RESB 定义。

下面的编程序列展示了如何检测 CC 线是否连接：

步骤 1：设置 ***cc_ctrl*** 寄存器中的 ***CC_CMPRA_EN*** 来使能两个电压比较器。

步骤 2：先设置为 1，然后将 ***cc_ctrl*** 寄存器中的 ***CC_CMPRA_LAT*** 设置为 0，以同时对这两个比较器的输出值进行采样并保持。

步骤 3：读取 ***cc_ctrl*** 寄存器中的 ***CC1_ATT*** 和 ***CC2_ATT***，以确定哪个激活引脚连接到了 USB 线缆上。

如下所示的表 5.22 通过读取 CC1_ATT 和 CC2_ATT2 的状态来解释连接检测结果。

CC1_ATT	CC2_ATT	描述
0	0	Reset value
1	0	CC1 attached
0	1	CC2 attached
1	1	N.A.

表 5.22：根据 CC1_ATT 和 CC2_ATT2 的状态得出的连接检测结果

5.14. GATE 引脚控制和连接

5.14.1 功能描述

特定 GATE 引脚专用于快充应用的电源管理，可通过外接功率 MOSFET 来通断电源线路。该引脚由 misc2 寄存器的 bit0 位控制，其输出特性详见表 5.23。此外，该 GATE 引脚具备以下特性：

(1) 开漏输出结构，在 V_{OL}=0.5V 时可提供 20mA 灌电流能力；

(2) 支持 21V 高电压容限。

misc2 寄存器的 Bit0	Gate 引脚
0	输出浮空（复位默认值）
1	0V 输出

表 5.23：GATE 引脚输出特性

5.14.2 GATE 引脚应用连接

图 5.17 展示了电源管理应用电路及外部功率 MOSFET 的连接方式。上电复位期间，GATE 引脚输出呈浮空状态，此时外部 MOSFET 关断，VBUS 至 VGATE 之间无电流通过。将 misc2 寄存器的 bit0 置 1，可强制 GATE 引脚输出零电平，从而导通 P 沟道 MOSFET，使充电电流从 VBUS 流向 VGATE。

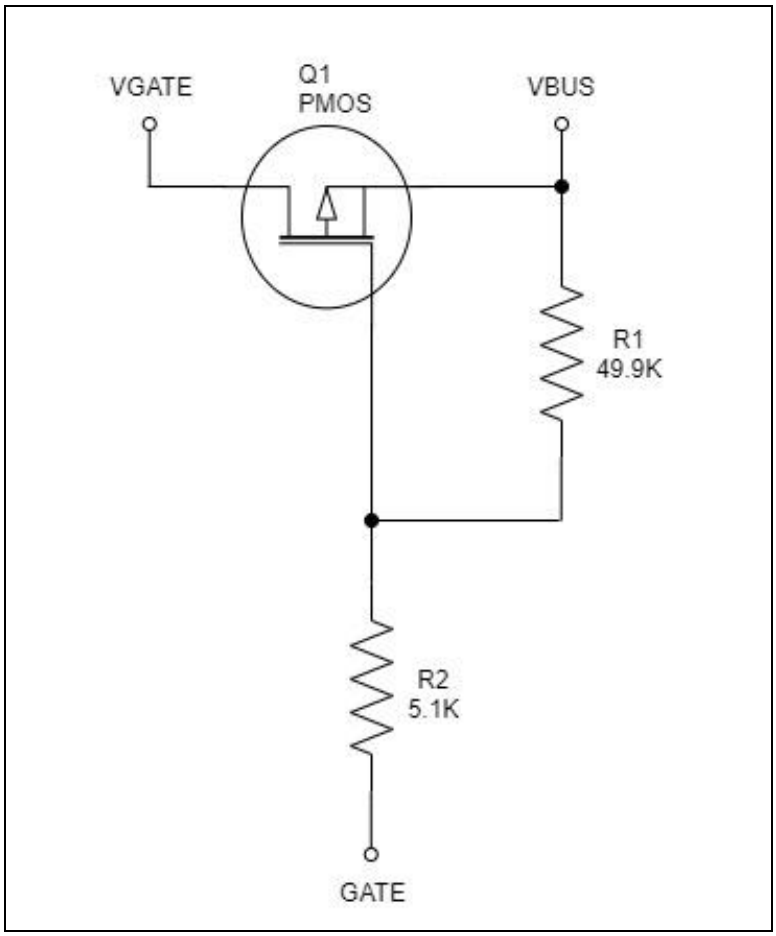


图 5.17：GATE 引脚应用连接示意图

5.15.VBUS (VDD) 电压比较器

5.15.1 功能概述

一个具有固定 7V 基准电压的 VBUS（VDD）比较器是针对特定的充电规范应用而设计的。misc2 寄存器的第 3 位用于启用电压比较器。比较结果可通过 misc2 寄存器的第 7 位读取。

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。(请不要使用)。
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. 复位控制寄存器 (rstc), IO 地址= 0x01

位	初始值	读/写	描述
7	0	读/写	外部复位 (PA5) 在测试模式下启用。 0 / 1：停用 / 启用
6	0	读/写	启用快速唤醒。 0: 普通唤醒。 唤醒时间为 1024 个 ILRC 时钟。 1: 快速唤醒。 唤醒时间为 16 个 ILRC 时钟。
5 - 4	00	读/写	看门狗超时周期 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
3	0	读/写	保留
2	1	读/写	LVR 复位启用。0 / 1：停用 / 启用
1	1	读/写	看门狗启用。 0 / 1：停用 / 启用
0	0	读/写	引脚 PA5/PRSTB 功能 。0 / 1: PA5 / PRSTB

6.3. 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。

6.4. 时钟模式寄存器(*clkmd*)，IO 地址 = 0x03

位	初始值	读/写	描述
7 - 6	00	读/写	系统时钟... 00：系统时钟 / 1 01：系统时钟 / 2 10：系统时钟 / 4 11：系统时钟 / 8
5	0	读/写	系统时钟选择。0 / 1：ILRC / IHRC
4 - 2	-	读/写	保留
1	1	读/写	内部高频 RC 使能。 0/1：停用/启用
0	0	读/写	内部低频 RC 模式选择。0 / 1: ILRC / NILRC

6.5. 中断允许寄存器(*inten*)，IO 地址 = 0x04

位	初始值	读/写	描述
7	0	读/写	保留
6	0	读/写	保留
5	0	读/写	启用从 Timer2 的溢出中断。0/1：停用/启用
4	0	读/写	保留
3	0	读/写	保留
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 PD_PHY 中断。0/1：停用/启用
0	0	读/写	启用 PA0 的中断。0/1：停用/启用

6.6. 中断请求寄存器(*intrq*)，IO 地址 = 0x05

位	初始值	读/写	描述
7	-	读/写	保留
6	-	读/写	保留
5	-	读/写	Timer2 的中断请求，此位是由硬件置位并由软件清零。 0/1：不请求/请求
4	-	读/写	保留
3	-	读/写	保留
2	-	读/写	Timer16 的中断请求，此位是由硬件置位并由软件清零。 0/1：不请求/请求
1	-	读/写	PD_PHY 的中断请求，此位是由硬件置位并由软件清零。 0/1：不请求/请求
0	-	读/写	PA0 中断请求，此位是由硬件置位并由软件清零。 0/1：不请求/请求

6.7. LVR 复位控制寄存器 (lvrc), IO 地址= 0x08

位	初始值	读/写	描述
7 – 4	0000	只写	LVR 复位电压选择 0000：1.8 V 0001：1.9 V 0010：2.0 V 0011：2.1 V 0100：2.2 V 0101：2.3 V 0110：2.4 V 0111：2.5 V 1000：2.7 V 1001：3.0 V 1010：3.15V 1011：3.3 V 1100：3.5 V 1101：3.75V 1110：4.0 V 1111：4.5 V
3 – 0	-	只写	保留

6.8. 中断边缘选择寄存器 (*integ*), IO 地址= 0x09

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	-	-	保留
1 - 0	00	只写	PA0 中断边缘选择： 00：上升缘和下降缘都请求中断 01：上升缘请求中断 10 / 11：下降缘请求中。

6.9. Timer16 控制寄存器(*t16m*)，IO 地址= 0x0B

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 100：IHRC 110：ILRC 111：PA0 下降沿（从外部引脚）
4 - 3	00	读/写	Timer16 时钟分频： 00：÷1

位	初始值	读/写	描述
			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 的第 8 位 1：Timer16 的第 9 位 2：Timer16 的第 10 位 3：Timer16 的第 11 位 4：Timer16 的第 12 位 5：Timer16 的第 13 位 6：Timer16 的第 14 位 7：Timer16 的第 15 位

6.10.DP/DM IO 控制寄存器 (*dp_dm_io_ctrl*), IO address = 0x0C

位	初始值	读/写	描述		
7 - 4	0000	读/写	保留		
3 - 2	00	读/写	DM IO 模式选择。		
			00	O_DM_BUF_EN O_DM_BUFVREF_SEL O_DM_TX_EN O_DM_DOUT_EN O_DM_DOUT O_DM_CMP_EN	= DM_BUF_EN = DM_BUFVREF_SEL = DM_TX_EN = DM_DOUT_EN = DM_DOUT = DM_CMP_EN
			01	O_DM_BUF_EN O_DM_BUFVREF_SEL O_DM_TX_EN O_DM_DOUT_EN O_DM_DOUT O_DM_CMP_EN	= pac[6] = pa[6] = 1'b0 = 1'b0 = 1'b0 = padier[6]
			10	O_DM_BUF_EN O_DM_BUFVREF_SEL O_DM_TX_EN O_DM_DOUT_EN O_DM_DOUT O_DM_CMP_EN	= 1'b0 = 1'b0 = pac[6] = 1'b0 = pa[6] = padier[6]
			11	O_DM_BUF_EN O_DM_BUFVREF_SEL O_DM_TX_EN O_DM_DOUT_EN O_DM_DOUT O_DM_CMP_EN	= 1'b0 = 1'b0 = 1'b0 = pac[6] = pa[6] = padier[6]

位	初始值	读/写	描述		
1 - 0	00	读/写	DP IO 模式选择。		
			00	O_DP_BUF_EN O_DP_BUFVREF_SEL O_DP_TX_EN O_DP_DOUT_EN O_DP_DOUT O_DP_CMP_EN	= DP_BUF_EN = DP_BUFVREF_SEL = DP_TX_EN = DP_DOUT_EN = DP_DOUT = DP_CMP_EN
			01	O_DP_BUF_EN O_DP_BUFVREF_SEL O_DP_TX_EN O_DP_DOUT_EN O_DP_DOUT O_DP_CMP_EN	= pac[4] = pa[4] = 1'b0 = 1'b0 = 1'b0 = padier[4]
			10	O_DP_BUF_EN O_DP_BUFVREF_SEL O_DP_TX_EN O_DP_DOUT_EN O_DP_DOUT O_DP_CMP_EN	= 1'b0 = 1'b0 = pac[4] = 1'b0 = pa[4] = padier[4]
			11	O_DP_BUF_EN O_DP_BUFVREF_SEL O_DP_TX_EN O_DP_DOUT_EN O_DP_DOUT O_DP_CMP_EN	= 1'b0 = 1'b0 = 1'b0 = pac[4] = pa[4] = padier[4]

6.11. 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
7-6	-	只写	保留。(请写 00 以备将来兼容)
5	0	只写	使能 PA5 数字输入和唤醒事件。1 / 0：启用/ 停用 可将该位设置为低电平，以禁用通过 PA5 引脚翻转来实现的唤醒功能。
4	-	只写	保留。
3	0	只写	使能 PA3 数字输入和唤醒事件。1 / 0：启用/ 停用 可将该位设置为低电平，以禁用通过 PA3 引脚翻转来实现的唤醒功能。
2 - 1	-	只写	保留。（请写 00 以备将来兼容）
0	0	只写	使能 PA0 数字输入、唤醒和中断请求。1 / 0：启用/ 停用。 可以将该位设置为低电平，以禁用通过 PA0 引脚翻转进行唤醒以及来自该引脚的中断请求。

6.12. 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 - 2	-	只写	保留。(请保留 00 以备将来兼容)
1	0	只写	使能 PB1 数字输入和唤醒功能。0 / 1：停用/ 启用 可将该位设置为低电平，以禁用通过 PB1 引脚翻转来实现的唤醒功能。
0	0	只写	使能 PB0 数字输入、唤醒和中断请求。1 / 0：启用/ 停用。 可将该位设置为低电平，以禁用通过 PB0 引脚翻转来实现的唤醒功能。

6.13. 端口 A 数据寄存器(*pa*)，IO 地址= 0x10

位	初始值	读/写	描述
7	-	读/写	保留
6 - 0	0x00	读/写	数据寄存器的端口 A。(PA7 引脚不可用)

6.14. 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 为 open drain 输出。PA1、PA2、PA4、PA6 和 PA7 不可用。

6.15. 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拉高端口 A 每个相应的引脚。 0/1：停用/启用。

6.16. 端口 A 下拉控制寄存器(*papl*)，IO 地址= 0x13

位	初始值	读/写	描述
7 - 0	0x00	读/写	端口 A 下拉控制寄存器。这些寄存器是用来控制拉低端口 A 每个相应的引脚。 0/1：停用/启用。 PA1、PA2、PA4、PA6 和 PA7 不可用。

6.17. 端口 B 数据寄存器(*pb*)，IO 地址= 0x14

位	初始值	读/写	描述
7 - 2	-	读/写	保留
1 - 0	00	读/写	数据寄存器的端口 B。(仅 PB0 和 PB1 可用)

6.18. 端口 B 控制寄存器(*pbc*)，IO 地址= 0x15

位	初始值	读/写	描述
7 - 2	-	读/写	保留
1 - 0	00	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。(仅 PB0 和 PB1 可用)

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

位	初始值	读/写	描述
7 – 2	-	读/写	保留
1- 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，该上拉功能仅在输入模式下有效。 0/1：停用 / 启用。

6.20. 端口 B 下拉控制寄存器(*pbpl*)，IO 地址= 0x17

位	初始值	读/写	描述
7 – 2	-	读/写	保留
1 – 0	00	读/写	端口 B 下拉控制寄存器。这些寄存器是用来控制拉低端口 B 每个相应的引脚，该下拉功能仅在输入模式下有效。 0/1：停用 / 启用。

6.21.Timer2 PWM 控制寄存器 (*tpw2c*), IO 地址= 0x1A

位	初始值	读/写	描述
7	0	读/写	Timer2 PWM 停用/启用。 0：TPW2 停用 1：TPW2 启用
6	0	读/写	Timer2 PWM 模式选择 0：周期模式 1：PWM 模式
5 - 4	00	读/写	Timer2 PWM 时钟源选择 00: IHRC 01: IHRC * 2 10: 保留 11: ILRC
3 - 2	00	读/写	Timer2 PWM 时钟预分频 00: ÷ 1 01: ÷ 4 10: ÷ 16 11: ÷ 64
1 - 0	00	读/写	Timer2 PWM 分辨率选择 00: 8-位 01: 7-位 10: 6-位 11: 保留

6.22.Timer2 PWM 分频寄存器 (*tpw2s*), IO 地址= 0x1B

位	初始值	读/写	描述
7	0	读/写	Timer2 PWM 输出结果的反极性。 0：极性不反转 1：极性反转
6 - 5	00	读/写	Timer2 PWM 输出端口选择。 00：关闭 TPW2 输出 01：保留 10：TPW2 输出到 PA3 11：保留
4 - 0	0x00	读/写	Timer2 PWM 时钟分频。

6.23. Timer2 PWM 上限寄存器 (*tpw2b*), IO 地址= 0x1C

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

6.24. 操作寄存器 (*opr1*), IO 地址= 0x1D

位	初始值	读/写	描述
6 - 4	000	读/写	输出端口信号。 000：关闭输出端口信号 001：输出 PD PHY 端口信号

6.25. 杂项寄存器 (*misc2*), IO 地址= 0x1F

位	初始值	读/写	描述
7	0	只写	VBUS (VDD) 电压比较器输出。 0: VBUS (VDD) < 7V 1: VBUS (VDD) > 7V
6	0	读/写	ILRC 在 StopSys 模式下的开关。 0：停用 1：启用 注意：请确保将其设置为**禁用**（Disable）
5	0	读/写	LVR 在 StopSys 模式下的开关。 0：停用 1：启用 注意：请确保将其设置为**禁用**（Disable）
4	1	读/写	LVR 电源开关。 0：关闭 1：开启（注意） 注意：请确保将其设置为**开启**（ON）。
3	1	读/写	VBUS (VDD) 电压比较器控制。 0：停用 1：启用
2	1	读/写	PD 控制器频率 6MHz 和 3MHz 切换。 0：停用 1：启用

位	初始值	读/写	描述
1	1	读/写	PD 控制器频率 12MHz 切换 0：停用 1：启用
0	0	读/写	GATE 驱动控制（GATE Drive Control） 0：停用(输出高阻，Output Floating) 1：启用(输出 0V，Output 0V)

6.26. PD PHY FIFO 寄存器(*pd_fifo*), IO 地址= 0x30

位	初始值	读/写	描述
7 - 0	-	读/写	先进先出存储器（FIFO）： 一个 4 字节的 FIFO 用于 PD PHY 的发送（TX）和接收（RX）数据存储。 发送模式（TX，写入，WO）：用于写入 PD SOP 数据包的数据载荷。 接收模式（RX，读取，RO）：用于读取接收到的数据包内容，包括 32 位 CRC 数据。

6.27. PD PHY 状态寄存器 (*pd_stat*), IO 地址= 0x31

位	初始值	读/写	描述
7	-	-	保留
6	1	只读	空闲状态（STATUS_IDLE）： 显示 USB（通用串行总线）CC（配置通道）线的状态。 它用于避免在 CC 线上发生信号冲突。 用户必须确保在开始传输数据之前总线处于空闲状态。 0：总线处于活动状态 1：总线处于空闲状态
5	0	只读	状态_硬复位（STATUS_HDRST）： PD PHY 控制器已成功接收一次硬复位信令。 该标志位仅在 RX_FINISH 和 RX_SUCCESS 同时有效时成立。 使能 RX_CLR 或禁用 pd_txrx 寄存器的 PHYCTRL_EN 时将自动清除。 0：接收到一个 PD 协议 SOP 包 1：接收到一次 PD 硬复位信令
4	0	只读	满显示位(FIFO_ FULL)： 显示 PD PHY 控制器先进先出存储器（pd_fifo 寄存器）的状态。 当启用 pd_txrx 寄存器的 RX_CLR 位或禁用 PHYCTRL_EN 位时，恢复为默认值。 0：未满 1：已满
3	1	只读	空显示位(FIFO_ EMPTY)： 显示 PD PHY 控制器先进先出存储器（pd_fifo 寄存器）的状态。 当启用 pd_txrx 寄存器的 RX_CLR 位或禁用 PHYCTRL_EN 位时，恢复为默认值。 0：非空 1：空
2	0	只读	接收成功显示(RX_ SUCCESS)： PD PHY 控制器已成功接收到一个数据包或信令。仅当 RX_FINISH 处于激活状态时，该位才有效。 当启用 pd_txrx 寄存器的 RX_CLR 位或禁用 PHYCTRL_EN 位时，该位会自动清零。

位	初始值	读/写	描述
			0：接收失败（接收到的数据可能是非法的或已损坏，应忽略该数据。） 1：接收成功
1	0	只读	接收完成显示(RX_FINISH)： PD PHY 控制器已完成一个数据包或信令的接收。 当启用 pd_txrx 寄存器的 RX_CLR 位或禁用 PHYCTRL_EN 位时，该位会自动清零。 0：如果 PD PHY 控制器开始接收一个数据包，则表示接收处于忙碌状态。 1：接收完成
0	0	只读	发送完成显示(TX_FINISH): 标志 PD PHY 控制器已完成一次数据包或信令的成功发送。 当启用 pd_txrx 寄存器的 TX_CLR 位或禁用 PHYCTRL_EN 位时，该位会自动清零。 注意：如果 PD PHY 控制器意外开始接收一个数据包，该位将不会处于激活状态，并且此次数据包传输将被取消。 0：如果 pd_txrx 寄存器的 TX_START 位已启用，则表示传输处于忙碌状态。 1：传输完成

6.28. PD PHY TXRX 寄存器(*pd_txrx*), IO 地址= 0x32

位	初始值	读/写	描述
7	-	-	保留
6	0	读/写	发送 Hard Reset 信号控制位(SEND_HDRST)： 使 PD PHY 控制器能够在 CC 线上发送一个硬复位信令。 0：停用 1：启用
5	0	读/写	PD PHY 控制器启用位(PHYCTRL_EN)： 启用 PD PHY 控制器。 注意：当用户禁用 PD PHY 控制器时，以下信号也将同时被清除： （1）pd_stat 寄存器的 TX_FINISH、RX_FINISH、RX_SUCCESS 和 STATUS_HDRST 位 （2）PD PHY 控制器中断信号 （3）pd_stat 寄存器的 FIFO_EMPTY 和 FIFO_FULL 位恢复为默认值 0：停用 1：启用
4	0	读/写	开始发送数据包或信号控制位(TX_START)： 使 PD PHY 控制器能够传输一个数据包或信令，包括内置自测试（BIST）信令。 0：无效果 1：启用（启用该位后自动清零）
3	0	读/写	接收完成后的状态清除控制位(RX_CLR)： 当 PD PHY 控制器完成一个数据包或信令的接收后，用户必须设置该位来清除所有接收状态和 PD PHY 中断。当启用该位时，以下信号也将同时被清除： （1）pd_stat 寄存器的 RX_FINISH、RX_SUCCESS 和 STATUS_HDRST 位。 （2）PD PHY 中断信号。 （3）pd_stat 寄存器的 FIFO_EMPTY 和 FIFO_FULL 位恢复为默认值。 0：无效果 1：启用（启用该位后自动清零）

位	初始值	读/写	描述
2	0	读/写	发送状态清除控制位(TX_CLR)： 当 PD PHY 控制器完成一个数据包或信令传输后，用户必须设置该位以清除所有发送状态和 PD PHY 控制器中断。当启用该位时，以下信号也将同时被清除： （1）pd_stat 寄存器的 TX_FINISH 位 （2）PD PHY 控制器中断信号 0：无效果 1：启用（启用该位后自动清零）
1	0	读/写	读指针更新控制位(FIFO_RO)： 用户从先进先出存储器（pd_fifo 寄存器）中读取一个字节的数据后，必须更新 PD PHY 控制器先进先出存储器的读索引指针。正确的操作是“先写入 1，然后写入 0” 或者“先写入 0，然后写入 1”（上升沿触发模式）。
0	0	读/写	写指针更新控制位(FIFO_WR)： 用户在向 PD PHY 控制器先进先出存储器（pd_fifo 寄存器）写入一个字节的数据后，必须更新该先进先出存储器的写索引指针。正确的操作是“先写入 1，然后写入 0”或“先写入 0，然后写入 1”（上升沿触发模式）。

6.29. PD PHY 控制寄存器 (*pd_ctrl*), IO 地址= 0x33

位	初始值	读/写	描述
7:4	-	-	保留
3:2	0	读/写	调试信号探测选择位(PROBE_SEL)： 当所有相关的通用输入输出（GPIO）引脚都切换到探测模式时，用于选择将电源传输 PD PHY 控制器的内部信号输出到指定的 GPIO 引脚上。这些信号在软件开发过程中对调试协议错误很有用。 注意：仅当 opr1 寄存器的[6:4]位为 001（输出 PD PHY 探测信号）时，这些位才有效。 00：选择 int_pd_phy, st_phy[2:0]信号。 01：选择 fifo_ov, STATUS_HDRST, RX_SUCCESS, RX_FINISH 信号。 10：选择 FIFO_FULL, FIFO_EMPTY, TX_FINISH, STATUS_IDLE 信号。 11：选择 crc_cmp_out, sym_eop, pktdet_fail, BMC_BIT 信号。
1	0	读/写	发送期间禁用接收路径控制位(DIS_RX): PD PHY 控制器在传输数据期间会自动关闭接收数据路径。 0: 停用 1: 启用
0	0	读/写	CC 数据路径控制权限选择(CC_SEL): 该寄存器用于切换 PD PHY 控制器数据通路的控制权限 0：停用（PD PHY 控制器自动管理所有的发送 / 接收数据路径。） 1：启用（用户可以通过设置 cc_misc 寄存器中的 BMCRX_EN、BMCTX_EN 和 BMCTX_DOUT 来控制整个数据通信。）

6.30. PD PHY 解码器寄存器(*pd_decoder*), IO 地址= 0x34

位	初始值	读/写	描述
7:5	4	读/写	TH_LOW： 用于调整 PD PHY 数据接收能力。 BMC 解码低压信号长度的阈值选择： 000：11T 时钟时间（166.66ns X 11） 001：12T 时钟时间（166.66ns X 12） 010：13T 时钟时间（166.66ns X 13） 011：14T 时钟时间（166.66ns X 14） 100：15T 时钟时间（166.66ns X 15） 101：16T 时钟时间（166.66ns X 16） 110：17T 时钟时间（166.66ns X 17） 111：18T 时钟时间（166.66ns X 18）
4:2	4	读/写	TH_HIGH： 用于调整 PD PHY 数据接收能力。 BMC 解码高压信号长度的阈值选择： 000：11T 时钟时间（166.66ns X 11） 001：12T 时钟时间（166.66ns X 12） 010：13T 时钟时间（166.66ns X 13） 011：14T 时钟时间（166.66ns X 14） 100：15T 时钟时间（166.66ns X 15） 101：16 T 时钟时间（166.66ns X 16） 110：17T 时钟时间（166.66ns X 17） 111：18T 时钟时间（166.66ns X 18）
1:0	1	读/写	TH_DEB: 用于调整 PD PHY 数据接收能力。 CC 线上接收数据的去抖时间选择： 00: 消抖 1T 时钟时间(166.66ns X 1) 01: 消抖 2T 时钟时间(166.66ns X 2) 10: 消抖 3T 时钟时间(166.66ns X 3) 11: 消抖 4T 时钟时间(166.66ns X 4)

6.31. DP 输出寄存器 (*dp_out*), IO 地址= 0x38

位	初始值	读/写	描述
7:6	-	-	保留
5	1	读/写	DP_ISINK_EN: 使能数据正极（DP）吸收电流路径（25 微安至 175 微安，弱下拉）。 注意：强烈建议在系统启动后将其关闭。其功能仅用于在 DP 引脚上提供一个初始值，以防止系统进入未知状态进行操作。 0：停用 1：启用
4	0	读/写	DP_BUFVREF_SEL: DP 驱动缓冲器的电压选择；该位仅在 DP_BUF_EN = 1 时有效。 0：0.6V

位	初始值	读/写	描述
			1：3.3V
3	0	读/写	DP_BUF_EN： 启用 DP 驱动缓冲。 0：停用 1：启用
2	0	读/写	DP_TX_EN： 启用 DP 驱动缓冲。 0：停用 1：启用
1	0	读/写	DP_OUT_EN： 启用 DP 数字输出。 0：停用 1：启用
0	0	读/写	DP_OUT： DP 驱动电压选择；该位仅在 DP_OUT_EN 或 DP_TX_EN = 1 时有效。 0：当 DP_OUT_EN 或 DP_TX_EN = 1 时为 0V 1：(1)当 DP_OUT_EN = 1 时为 5V（注） (2) DP_TX_EN = 1 时为 3.3V 注：实际输出电压由 5V 稳压器输出 (VREG) 决定。额外的压降可能导致 DP 驱动电压低于预期值（5V）。

6.32. DP 输入寄存器 (*dp_in*), IO 地址= 0x39

位	初始值	读/写	描述
7:4	-	-	保留
3:2	2	读/写	DP_CMPVREF_SEL： DP 电压比较器的参考电压选择。 00：0.35V 01：1.2V 10：2.4V 11：3.0V
1	1	读/写	DP_CMP_EN： 启用 DP 电压比较器。 0：停用 1：启用
0	0	只读	DP_DIN： DP 电压比较器的输出值。 0：逻辑 0 1：逻辑 1

6.33. DM 输出寄存器(*dm_out*), IO 地址= 0x3A

位	初始值	读/写	描述
7:6	-	-	保留
5	1	读/写	DM_ISINK_EN： 启用 DM 灌电流路径（25uA~175uA，弱下拉）。 注意：强烈建议在系统启动后将其关闭。其功能仅用于为 DM 引脚提供一个初始值，以防止未知状态进入系统运行。 0：停用 1：启用
4	0	读/写	DM_BUFVREF_SEL： DM 驱动缓冲器的电压选择；该位仅在 DM_BUF_EN = 1 时有效。 0：0.6V 1：3.3V
3	0	读/写	DM_BUF_EN： 启用 DM 驱动缓存。 0：停用 1：启用
2	0	读/写	DM_TX_EN： 启用 DM 驱动缓存。 0：停用 1：启用
1	0	读/写	DM_OUT_EN： 启用 DM 数字输出。 0：停用 1：启用
0	0	读/写	DM_OUT： DM 驱动电压选择；该位仅在 DM_OUT_EN 或 DM_TX_EN = 1 时有效。 0：当 DM_OUT_EN 或 DM_TX_EN = 1 时为 0V 1：(1) 当 DM_OUT_EN = 1 时为 5V（注 1） (2) 当 DM_TX_EN = 1 时为 3.3V 注 1：实际输出电压由 5V 稳压器输出 (VREG) 决定。额外的压降可能导致 DM 驱动电压低于预期值（5V）。

6.34. DM 输入寄存器(*dm_in*), IO 地址= 0x3B

位	初始值	读/写	描述
7:4	-	-	保留
3:2	2	读/写	DM_CMPVREF_SEL: DM 电压比较器的参考电压选择。 00: 0.35V 01: 1.2V 10: 2.4V 11: 3.0V
1	1	读/写	DM_CMP_EN: 启用 DM 电压比较器。 0: 停用 1: 启用
0	0	只读	DM_DIN: DM 电压比较器的输出值 0: 逻辑 0 1: 逻辑 1

6.35. CC 控制寄存器(cc_ctrl), IO 地址= 0x3D

位	初始值	读/写	描述
7:5	-	-	保留
4	0	读/写	CC_CMPRA_LAT： 对 CC1/CC2 连接检测比较器输出进行采样，并将比较值保存到 CC1/CC2 采样锁存器中。 注意：正确的操作是“先写 1，然后写 0”。 0：释放 1：锁存比较器输出
3	0	读/写	CC_REGRA_RESB： 复位 CC1/CC2 采样锁存器。 0：复位（低电平有效） 1：释放
2	1	读/写	CC_CMPRA_EN: 启用 CC1/CC2 连接检测比较器。 0：停用 1：启用
1	0	只读	CC2_ATT: CC2 采样寄存器输出 0：CC2 连接检测比较器输出＜0.2V 1：CC2 连接检测比较器输出≧0.2V
0	0	只读	CC1_ATT: CC1 采样寄存器输出 0：CC1 连接检测比较器输出 ＜0.2V 1：CC1 连接检测比较器输出 ≧0.2V

6.36. CC 状态寄存器 (cc_stat), IO 地址= 0x3E

位	初始值	读/写	描述
7:3	-	-	保留
2	0	读/写	CC_CMPSNKXA_EN: 启用两个独立的 CC1/CC2 电压比较器，以检测 USB Type-C 电流模式. 0：停用 1：启用
1	0	只读	CC_SNK3P0A： 用于 USB Type-C 5V/3A 检测的 CC1/CC2 电压比较器输出。 0：CC1/CC2 电压比较器输出 ＜1.23V 1： CC1/CC2 电压比较器输出 ≧1.23V
0	0	只读	CC_SNK1P5A： 用于 USB Type-C 5V/1.5A 检测的 CC1/CC2 电压比较器输出。 0：CC1/CC2 电压比较器输出 ＜0.66V 1：CC1/CC2 电压比较器输出 ≧0.66V

6.37. CC 杂项寄存器 (cc_misc), IO 地址= 0x3F

位	初始值	读/写	描述
7:4	-	-	保留
3	0	读/写	BMCTX_EN： 启用 CC1/CC2 数字输出。 仅当 pd_ctrl 寄存器的 CC_SEL = 1 时，该位才有效。 0：停用 1：启用
2	0	读/写	BMCTX_DOUT： CC1/CC2 数字输出数据。 仅当 pd_ctrl 寄存器的 CC_SEL = 1 时，该位才有效。 0：逻辑零 1：逻辑一
1	0	读/写	BMCRX_EN： 使能 CC1/CC2 接收器边沿检测器。 仅当 pd_ctrl 寄存器的 CC_SEL = 1 时，该位才有效。 0：停用 1：启用
0	0	只读	BMC_BIT： CC1/CC2 接收器边沿检测器输出。 0：逻辑 0 1：逻辑 1

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 (运算结果为无符号数制中的加法进位或减法借进位)
AC	辅助进位标志 (如果 ALU 运算结果后有低四位进位，则该位设置为 1)
M.n	只允许寻址在地址 0~0x3F (0~63) 的位置
IO.n	只允许寻址在地址 0~0x3F (0~63) 的位置

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』
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 PUD310 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 的地址并将累加器的数据读取并载入到 RAM。它需要 2T 时间执行这一指令。 例如： *idxm* index, a; 结果： [index] ← a; index 是以 word 定义。 受影响的标志位： Z：『不变』， C：『不变』， AC：『不变』， OV：『不变』 应用范例： `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 PMS154 要为 0` `mov hb@RAMIndex, a ; // 将指针存到 RAM (MSB)` `…` `mov a, 0Xa5 ;` `idxm RAMIndex, a ; // 将累加器数据读取并载入地址为 0x5B 的 RAM`

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』
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 ; ---
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』

7.2. 算数运算类指令

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』
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	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，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』
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』
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』
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 ; // 跳到这里 goto err2 ; goto err3 ; … correct: // 跳到这里 …
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』
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』
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』

7.8. 指令执行周期综述

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

7.9. 指令影响标志综述

指令	Z	C	AC	OV	指令	Z	C	AC	OV	指令	Z	C	AC	OV
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	*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	-	-	-	*set0* IO.n	-	-	-	-	*set1* IO.n	-	-	-	-
set0 M.n	-	-	-	-	*set1* M.n	-	-	-	-	*ceqsn* a, I	Y	Y	Y	Y
ceqsn a, M	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	-	-	-	-	*nadd M, a*	Y	Y	Y	Y	*cneqsn a, I*	Y	Y	Y	Y
cneqsn a, M	Y	Y	Y	Y	*comp a, M*	Y	Y	Y	Y	*nadd a, M*	Y	Y	Y	Y
comp M, a	Y	Y	Y	Y	*swapc IO.n*	-	Y	-	-					

7.10. BIT 定义

位寻址只能定义在 RAM 区地址的 0x00 到 0x3F。

8. 代码选项(Code Options)

配置	选项	描述
Security	Enable	OTP 内容加密，程序不允许被读取。
	Disable	OTP 内容不加密，程序可以被读取。
LVR	4.5V	选择 LVR = 4.5V
	4.0V	选择 LVR = 4.0V
	3.75V	选择 LVR = 3.75V
	3.5V	选择 LVR = 3.5V
	3.3V	选择 LVR = 3.3V
	3.15V	选择 LVR = 3.15V
	3.0V	选择 LVR = 3.0V
	2.7V	选择 LVR = 2.7V
	2.5V	选择 LVR = 2.5V
	2.4V	选择 LVR = 2.4V
	2.3V	选择 LVR = 2.3V
	2.2V	选择 LVR = 2.2V
	2.1V	选择 LVR = 2.1V
	2.0V	选择 LVR = 2.0V
	1.9V	选择 LVR = 1.9V
	1.8V	选择 LVR = 1.8V

9. 特别注意事项

此章节提醒使用者在使用 PUD310 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

- 设置 IO 为输入。
- 用 PADIER 和 PBDIER 寄存器，将对应的位设为 1。
- IO 翻转宽度必须大于一个系统时钟周期。

(3) PA7 设置为输出引脚。

- PA7 只能做 Open Drain 输出，输出高时需要外加上拉电阻。

(4) PA5 设置为 PRSTB 输入引脚。

- 设定 PA5 作输入。
- 设定 RSTC.0=1 来启用 PA5 作为 PRSTB 输入引脚。

注意：请务必仔细阅读 PMC-APN013 的内容，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，步骤如下：

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

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

9.1.3. 系统时钟选择

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

- 例子：系统时钟从 IHRC 切换到 ILRC

```
CLKMD   =   0x02;       // 切到 ILRC，此处不能禁用 IHRC
CLKMD.1 =   0;          // 此时才可关闭 IHRC
```

- **错误：** 将 IHRC 切换为 ILRC，同时关闭 IHRC

```
CLKMD   =   0x00;       // MCU 会死机
```

9.1.4. 看门狗

看门狗默认为开，但程序执行 ADJUST_IC 时，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：

系统时钟	VDD	LVR	温度
1.5MHz	≧ 1.8V	≧ 1.8V	25℃
3MHz	≧ 2.0V	≧ 2.0V	
6MHz	≧ 2.0V	≧ 2.0V	
12MHz	≧ 2.3V	≧ 2.3V	

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

(2) 用户可将 MISC2.4 设置为“0”，以禁用 LVR。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。

(3) 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

PUD310 的烧录脚为 DP(PA4), DM(PA6), VDD 和 GND. 这 4 只引脚。

请使用 5S-P-003x 烧录。3S-P-002 或旧版本不支持 PUD310 的烧录。

- 合封（MCP）或在板烧录（On-Board Writing）时的有关电压和电流的注意事项：
  (1) V_{BAT} 可能高于 9.5V，而最大供给电流最高可达约 20mA。
  (2) 其他烧录引脚（GND 除外）的电位与 V_{BAT} 相同。

请用户自行确认在使用本产品于合封或在板烧录时，周边元件及电路不会被上述电压破坏，也不会钳制上述电压。

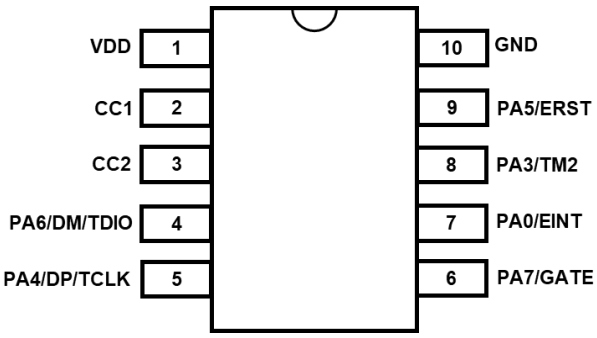
重要注意事项：

- **您必须按照 APN004 和 APN011 上的说明在处理器上编程 IC。**
- **在处理器端口的 V_{BAT} 和 GND 之间连接 0.01uF 电容器到 IC 有利于抑制干扰。但请不要连接＞0.01uF 的电容器，否则，编程可能会失败。**

使用 5S-P-003x 烧录 PUD310，使用 Jumper7 转接程序信号。信号的连接取决于 IC 封装。请参阅 Writer 用户手册的第 5 章，为目标 IC 封装制作 Jumper7 转接板。用户可以从以下网页链接获取用户手册。

http://www.padauk.com.tw/cn/technical/index.aspx?kind=27

以 ESSOP-10 为例，转接接线如下：



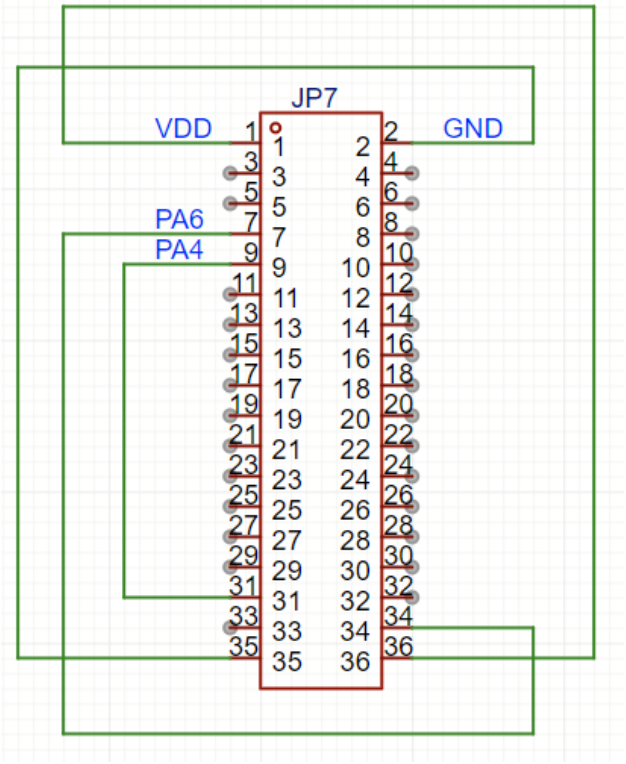
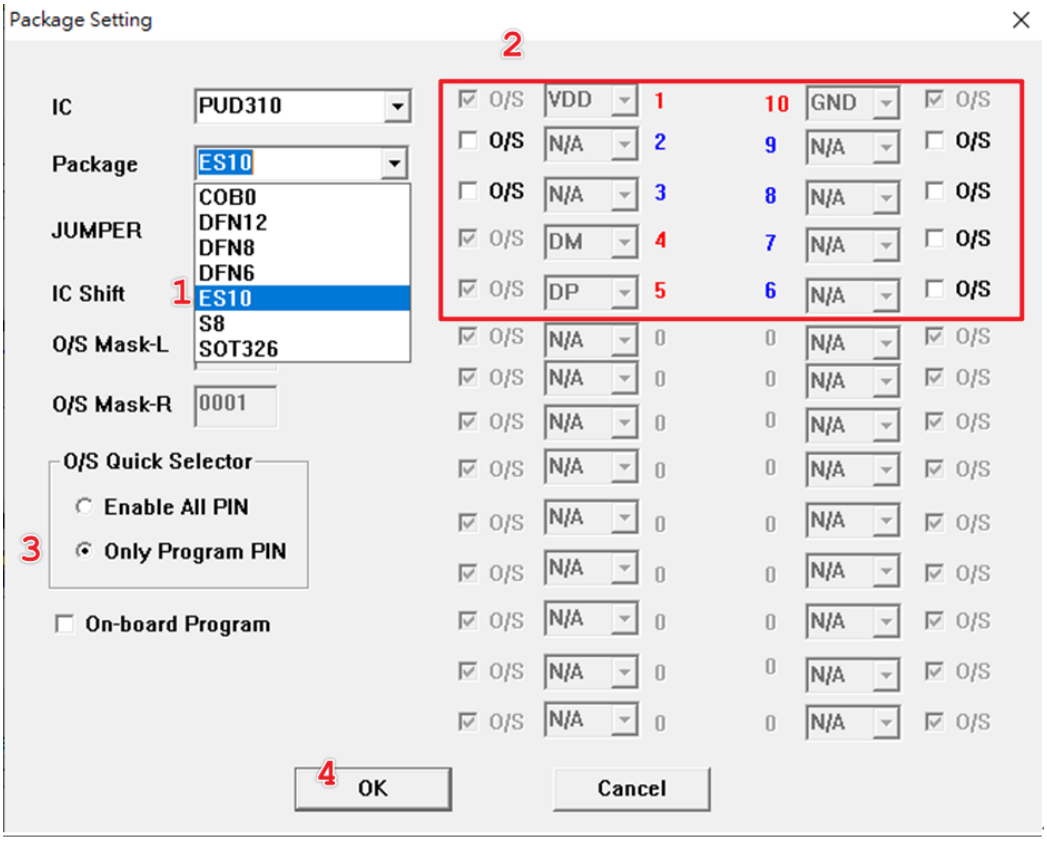


图. 9.1: P-003x 的 Jumper7 原理图

从图形用户界面加载 PDK，插入 JP7，然后在插座上输入 IC，无需移位。LCDM 显示 IC 就绪后，即可写入。


Package Setting
IC
PUD310
Package
ES10
COB0
DFN12
DFN8
DFN6
ES10
S8
SOT326
JUMPER
IC Shift
O/S Mask-L
O/S Mask-R
0001
O/S Quick Selector
Enable All PIN
Only Program PIN
On-board Program
1
2
3
4
OK
Cancel
O/S
VDD
DM
DP
GND
N/A