

# PCap04

## 电容-数字转换器

### 概述

PCap04是一款集成在数字信号处理器（DSP）中的电容-数字转换器（CDC），用于片上数据后处理。其前端基于ams的PICOCAP原理。

该转换原理在功耗、分辨率和速度方面具有极高的灵活性。此外，PCap04的输入电容范围广泛，从几个飞法到数百纳法不等。配置PCap04以适应不同的电容测量任务非常简便，包括单端和差分传感器，连接方式可为接地或浮地。片上DSP支持实现线性化、温度补偿等传感器算法，数据输出方式包括数字（SPI或IIC）和模拟（PDM/PWM）。

订购信息和内容指南见数据手册末页。

### 主要优势与特性

PCap04电容数字转换器的优势与特性如下：

图1：  
使用PCap04的附加价值

优势	特性
•高灵活性：通过配置轻松适应各种应用——超低功耗、高分辨率或高速	•最多接地6个电容，浮空3个电容——电容范围1pF至100nF——内部参考1pF至31pF ——集成保护驱动器
•高分辨率或高速	•在2.5 Hz和10pF基础电容下最高8aF ——采样率最高50kHz——最高20位分辨率
•片上DSP用于传感器算法、信号后处理，包括线性和温度补偿	•32位DSP •3k ROM代码，1k NVRAM- 96 x 32位RAM-SPI / IIC接口- PDM / PWM输出，GPIO
•片上和外部温度测量能力	•供电电压2.1/3.0V至3.6V-工作电流降至3μA - PCap04-Bxxx -40°C至85°C - PCap04-Axxx -40°C至125°C - QFN24或芯片(1.588mm x 1.46mm)

## MEMS传感器应用

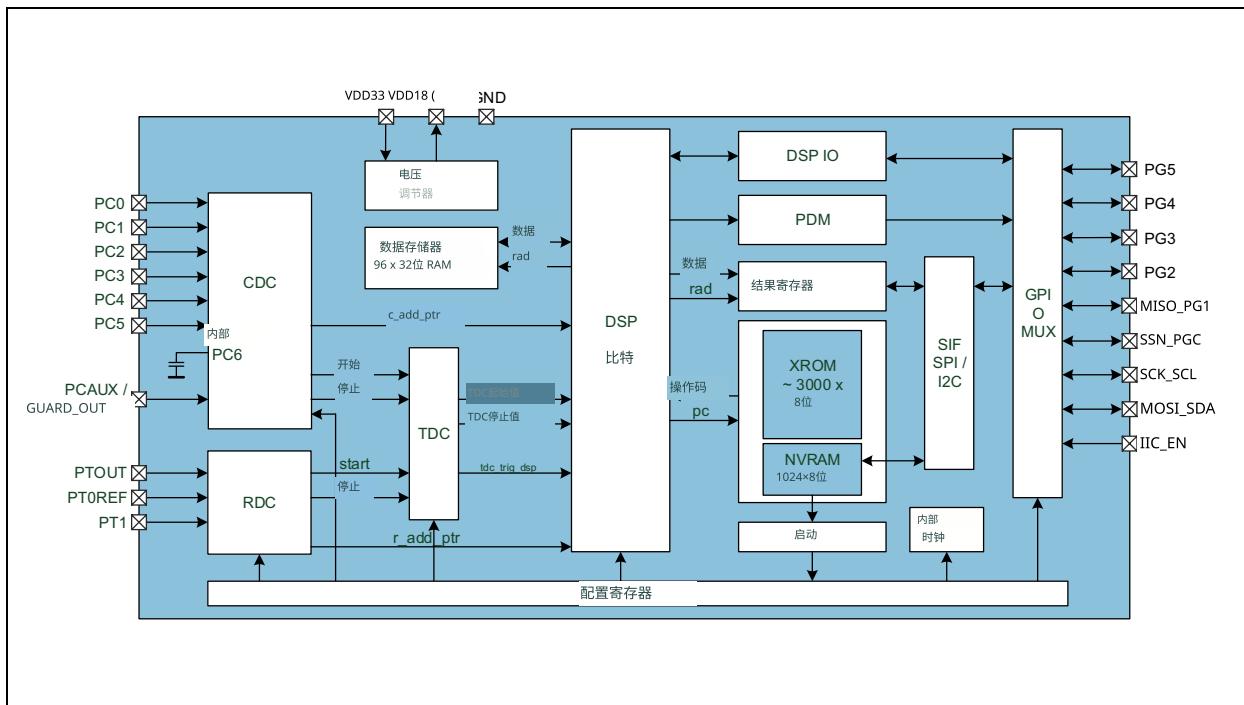
PCap04的应用包括：

- 位置传感器
- 压力传感器
- 力传感器
- 接近传感器
- 加速度传感器
- 倾角传感器
- 湿度传感器
- 露点传感器
- 倾斜传感器
- 角度传感器
- 无线应用
- 液位传感器

## 框图

本设备的功能模块如下所示：

图2：  
PCap04的功能块

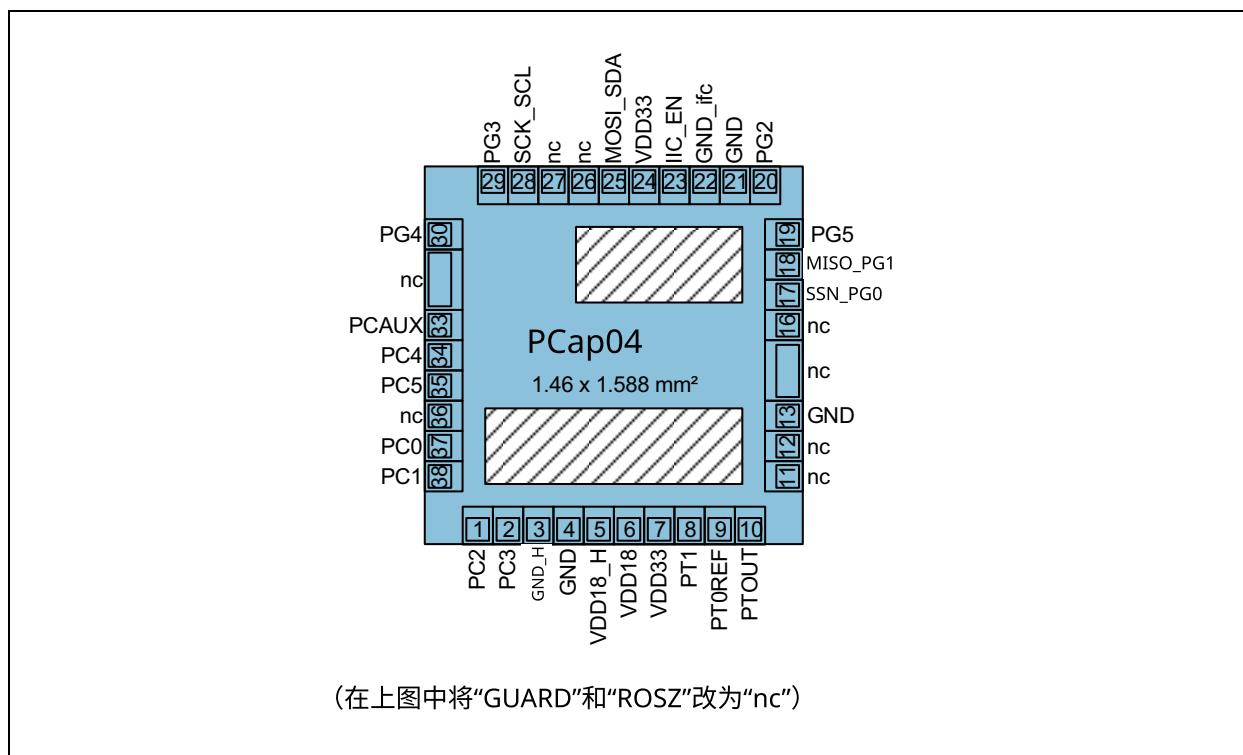


## 引脚分配

PCap04 will be available in QFN24 package or as pure die.

图3:

PCap04芯片引脚图

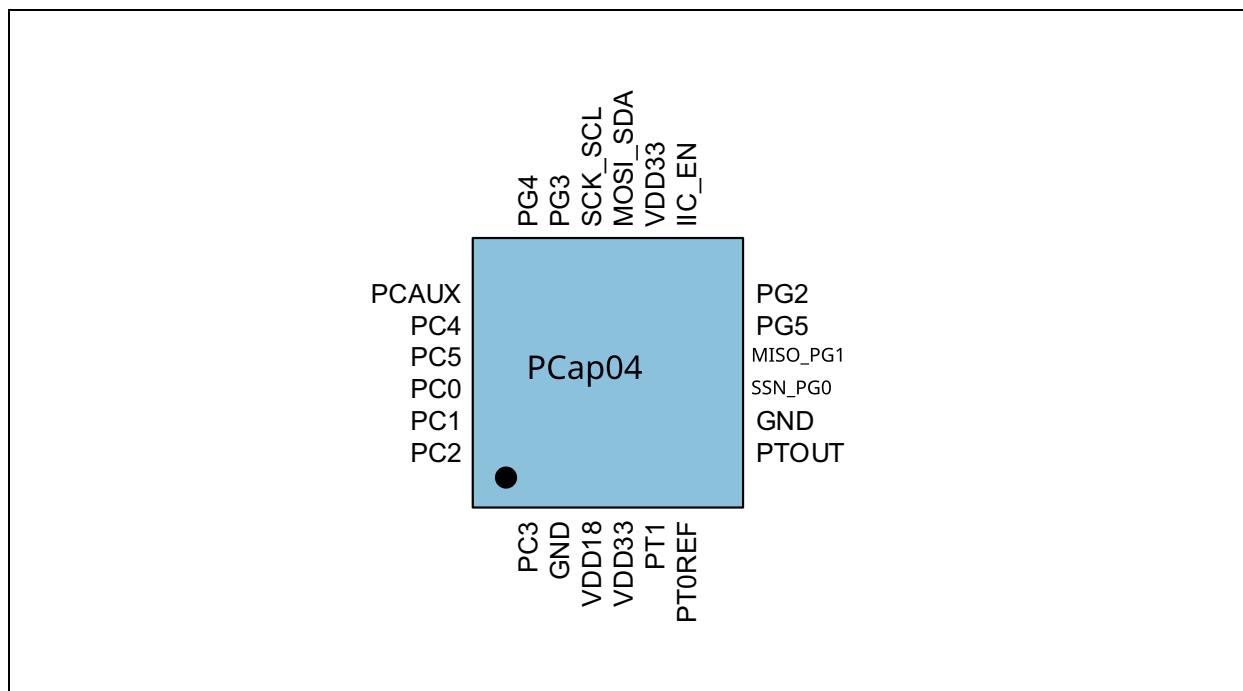


### 备注:

- 尺寸: 1.588毫米×1.46毫米 (带密封), 焊盘间距120μm, 焊盘开口为 85μm × 85μm。厚度为 290μm。

图4:

PCap04 QFN24引脚图



## amm引脚描述

图5：  
PCap04的引脚描述

引脚编号		引脚名称	描述	未使用
模具	24引脚QFN			
1	24	PC2	CDC端口	打开
2	1	PC3	CDC端口	打开
3		接地_H		GND
4	2	GND		GND
5		VDD18_H		VDD18
6	3	VDD18 (1)	核心电源电压	VDD18
7	4	VDD33 (2)	I/O电源电压	VDD33
8	5	PT1	RDC端口 (温度传感器)	打开
9	6	PTOREF	RDC端口 (温度传感器或外部参考)	打开
10	7	PTOUT (3)	RDC放电电容	打开
11		n.c.	无垫	
12		n.c.	无垫	
13	8	GND		
14/15		n.c.	无垫	
16		n.c.	始终开启	
17	9	SSN_PG0	串行选择线, 否则为通用I/O端口	打开
18	10	MISO_PG1	使用SPI时为主输出从输入, 否则为通用I/O端口	打开
19	11	PG5	通用I/O端口	打开
20	12	PG2	通用输入输出端口	开启
21		GND		GND
22		GND_ifc		GND
23	13	IIC_EN	串行接口选择, 0 = SPI使能1 = IIC使能	
24	14	VDD33		VDD33

引脚编号		引脚名称	描述	未使用
芯片	24引脚QFN			
25	15	MOSI_SDA	使用SPI时为主输出/从输入，否则为IIC的串行数据	
26		n.c.	始终开启	
27		n.c.	始终开启	
28	16	SCK_SCL	SPI/IIC的串行时钟	
29	17	PG3	通用输入输出端口	开启
30	17	PG4	通用输入输出端口	开启
31/32		n.c.	无垫片	
33	19	PCaux	辅助端口 •外部补偿电容或- 外部放电电阻-屏蔽输出	开路
34	20	PC4	CDC端口	开路
35	21	PC5	CDC端口	开路
36		n.c.	始终开启	开启
37	22	PC0	CDC端口	开启
38	23	PC1	CDC端口	开启

## 注意事项：

1. 连接缓冲电容  $\geq 4.7\mu\text{F}$
2. 连接缓冲电容  $\geq 10\mu\text{F}$
3. 连接10nF COG电容
4. 中心焊盘内部连接到地。禁止在其下方使用除地线以外的导线。
5. 建议不要使用中心焊盘，过多的焊膏可能影响焊接质量。
6. 适用插座：例如Plastronics 32QN50S15050D。

## amm Pad 坐标

图6：  
PCap04的焊盘坐标

焊盘编号	描述	X-POS (μm)	Y-POS (μm)
1	PC2	260.0	59.5
2	PC3	380.0	59.5
3	接地_H	488.0	59.5
4	GND	608.0	59.5
5	电源18V_H	728.0	59.5
6	VDD18	848.0	59.5
7	VDD33	968.0	59.5
8	PT1	1088.0	59.5
9	PTOREF	1208.0	59.5
10	PTOUT	1328.0	59.5
11	n.c.	无焊盘	无焊盘
12	n.c.	无垫	无垫
13	GND	1528.5	501.0
14/15	n.c.	无垫	无垫
16	n.c.	无垫	无垫
17	SSN_PG0	1528.5	965.0
18	MICO_PG1	1528.5	1085.0
19	PG5	1528.5	1205.0
20	PG2	1333.0	1400.5
21	GND	1213.0	1400.5
22	GND接口	1093.0	1400.5
23	IIC_EN	973.0	1400.5
24	VDD33	853.0	1400.5
25	MOSI_SDA	733.0	1400.5
26	n.c.	无焊盘	无填充
27	n.c.	无填充	无填充
28	SCK_SCL	374.7	1400.5

垫片编号	描述	X-位置 (μm)	Y-位置 (μm)
29	PG3	254.7	1400.5
30	PG4	59.5	1205.0
31/32	n.c.	无垫片	无垫片
33	PC AUX	59.5	859.9
34	PC4	59.5	739.8
35	PC5	59.5	619.8
36	n.c.	无垫	无垫
37	PC0	59.5	379.8
38	PC1	59.5	259.8

备注：

1. 垫片坐标为中心/中心 (x/y方向)，相对于芯片原点，芯片尺寸为1.588mm x 1.46mm (带封装，边长15μm)，垫片间距为120μm，垫片开口为 85μm × 85μm

## 绝对最大额定值

超出绝对最大额定值的应力可能会对器件造成永久性损坏。这些仅为应力等级。器件在这些或任何超出电气特性所示条件下的功能操作不作保证。长时间暴露于绝对最大额定值条件可能影响器件的可靠性。

图7：  
绝对最大额定值

符号	参数	最小值	马克斯	单位	评论
电气参数					
$V_{DD}/V_{GND}$	供电电压对地	-0.3	4.0	V	
$V_{IN}$	输入引脚对地电压	-0.3	4.0	V	
$I_{scr}$	输入电流 (抗锁存) @125°C	$\pm 100$		mA	JEDEC JESD78D 2011年11月
连续功耗 ( $T_A = 70^\circ\text{C}$ )					
$P_T$	连续功耗		1.44	mW	
静电放电					
$ESD_{HBM}$	静电放电HBM	$\pm 1000$		V	JS-001-2014
温度范围与存储条件					
$T_A$	工作环境温度  PCap04-Bxxx PCap04-Axxx	-40 - 40	85 125	°C °C	
$R_{THJA}$	环境热阻连接		28	°C/W	
$T_j$	工作结点温度PCap04- Bxxx  PCap04-Axxx		85 125	°C °C	
TSTRG	存储温度范围	-55	150	°C	

符号	参数	最小值	最大值	单位	备注
TBODY	回流过程中最大封装体温		260	°C	IPC/JEDEC J-STD-020 根据 IPC/JEDEC J-STD-020“非密封固态表面贴装器件的温度/回流敏感性分类”规定了回流峰值焊接温度（体温）。无铅引线封装的引线镀层为“哑光锡”（100%锡）。
RHNC	相对湿度（非冷凝）	5	85	%	
MSL	湿度敏感等级	3			最大存放时间为168小时
TSTRG-DOF	在箔上的DOF/芯片或晶圆的存储时间		3	月	指包装日期
TSTRG-DOF	用于DOF/芯片或箔片的存储温度	17	28	°C	
RHOPEN-DOF	开封包装中DOF/芯片或箔片的相对湿度		15	%	已开封包装
RH <sub>UNOPEN-DOF</sub>	封装中DOF/芯片或晶圆的相对湿度	40	60	%	密封袋
TSTRG-WP	晶圆或芯片在华夫包装中的存储时间		6	个月	17-28°C 40-60%的相对湿度，存放在原始Ultrapack盒中
tstrg-WP	晶圆或芯片在华夫包装中的存储时间		2	年	19-25°C 相对湿度低于15%，在密闭柜中用干燥空气存放
tstrg-WP	晶圆或芯片在华夫包装中的存储时间		5	年	19-25°C 相对湿度低于5%，在密闭柜中用干燥空气存放
TSTRG____WP	晶圆或芯片在华夫包装中的存储时间		10	年	19-25°C 相对湿度低于5%，存放于密闭柜和封闭的Ultrapak盒中，环境为保护性氮气气氛

## 电气特性

特性指设备保证正常工作的条件。有关测试条件的详细信息，请参见表格底部的注释。

图8：  
PCap04的电气特性

符号	参数	条件	最小	类型	最大	单位
$V_{DD}$	供电电压 (4)	相对于接地 断电: VDD在上电前必须低于0.05V	2.1		3.6	V
$V_{DD}$	电源电压	NVRAM 复位 -Bx: -40°C 至 85°C - Ax: -40°C 至 125°C	2.5		3.6	V
$V_{DD}$	电源电压	非易失存储器存储 范围: -25°C至60°C	3.0		3.6	V
VIO_DIGITAL	数字端口输入电压 (4)	相对于地	-0.6	3.3		V
VIO_DIGITAL	数字端口输入切换级别 (2)	高到低低到高	$0.7 * V_{DD}$		$0.3 * V_{DD}$	V
$V_{OH}$	数字端口 输出电压 (2)	HIGH	$V_{DD} - 0.4$		$V_{DD} + 0.1$	V
VOL	数字端口 输出电压 (2)	LOW	-0.1		0.4	V
泄漏	数字端口泄漏 (2)	IIC_EN=VDDHI GH	-0.1		1.0	μA
IleakPU	数字端口泄漏 (2)	内部上拉	-2.8		-5.2	μA
IleakL	数字端口漏电 (2)	IIC_EN=VDDLO W	-1.0		0.1	μA
ddq	静态电源 电流 (2)	-40°C 35°C 85°C 125°C		1.28 1.6 4.26 22.5	-5 18 50	μA
$R_P$	预充电 电阻 (2)	R_C_PCHGOR_C _PCHG1	7126	1018 0	1323 4	k欧姆
$R_D$	放电 电阻 (2)	R_DCHGO R_DCHG1R DCHG2 R_DCHG3	7 2163 126	10 3090 180	13 3911 7 234	k欧姆

符号	参数	条件	最小值	类型	最大值	单元
$C_{ref0}$ $C_{ref1}$ $C_{ref2}$ $C_{ref3}$ $C_{ref4}$	内部参考 (5)			0.41. 43.47 .415. 4		pF
$C_R$	内部RDC 排放容量 (2)	未连接	65	94	122	pF
IC_G7 IC_GLow	驱动器7 驱动低 (3)	$PC0 = V_{DD}$ $PCO = V_{DD} * 0.05$ $PCAUX = V_{DD}$		-3.5 -9		mA
VgardoVG UARD1 VGUARD2 Vguard3	增益保护运算放大器 (3)	$PC0 = 1.0V$		1.001 .01 1.02 1.03		V
R_DIS_SEN	RDC温度 Sensor (2) (6)	-40 °C 35°C 85°C 125 °C	- 1120 1200 1360	1047 1330 1572 1688	- 1680 1800 2040	欧姆
$R_{R\_DIS\_REF}$	RDC参考 (2)		1100	1476	1690	欧姆
$C_R$	RDC内部电容 (2)		65	94	122	pF
fOLF0 fOLF1 fOLF2 foLF3	OLF频率 (2)	高温 最小值, 低 最高温度 数值	3.6 25 47.6 92.8	10 60 100 200	16.5 96.4 171.4 305	kHz
$f_{OHF}$	OHF频率 (2)		1.36	2	2.6	兆赫
	NVRAM数据保持 (4) (7) PCap04-Bxxx PCap04-Axxx	3.0V至3.6V 85°C 125°C	20 20			年
	NVRAM耐久性(4)(7) PCap04-Bxxx PCap04-Axxx	25°C 85°C 25°C 125°C	$10^4$ $10^3$ $10^5$ $10^4$			循环次数

**注意事项：**

1.100%生产测试

2.在晶圆分选阶段进行100%生产测试，并在指定温度下通过设计和特性验证保证。

仅测试样品

参数由设计和特性测试保证

参数仅为典型值

100% 生产测试于 25°C，由设计和特性保证适用于工业温度范围

**重要提示：**我们仅在客户不更改寄存器62和63以及NVRAM地址654至959（唯一ID）的前提下，保证数据保持和耐久性。此外，必须严格按照第NVRAM和ROM部分描述的程序进行ERASENVRAM操作，否则我们将不再保证数据保持时间和耐久周期。

图9：

触发模式下，总电流I [ $\mu$ A] 随转换速率 (CONV\_TIME) 和分辨率 (C\_AVRG) 的变化

LP 振荡器频率 [kHz]	CONV TIME	测量速率 [Hz]	电流 I [ $\mu$ A]					
			C_AVRG (均方根分辨率 [Bits])					
			1	4	16	64	256	1024
			(13.6)	(14.6)	(15.6)	(16.6)	(17.8)	(18.6)
50	10000	2.5	2.3	2.5	2.7	4.6	11.8	44.0
50	2500	10	2.8	3.3	4.6	12.2	43.8	
50	1250	20	3.3	4.2	7.9	23.9		
50	625	40	4.6	6.0	14.0			
50	250	100	8.3	12.0	32.2			
50	125	200	14.3	22.3				
50	50	500	32.6	53.8				
50	25	1000	63.9					
50	12	2080	90					
200	24	4160	156					
200	12	9320	305					

**备注：**

- 除电容测量外，温度测量的附加值大约在  $2\mu$ A 和  $10\mu$ A 之间，具体取决于速度。以下总功耗值  $30\mu$ A 仅在以节能模式驱动片上1.8伏核心电源发生器时获得；最终的微安级节省还需要减慢DSP速度。典型数据。

**CDC特性**

图10：  
电气特性

频率 [Hz]	浮动 完全补偿			接地 内部补偿		
	均方 根噪 声[aF]	有效分辨率 [比特]		均方 根噪 声[aF]	有效分辨率 [比特]	
		10pF 基准	1pF 范围		10pF 基准	1pF 范围
2.5	8	20.2	16.9	9	20.1	16.8
5	12	19.7	16.4	13	19.6	16.3
10	19	19.0	15.7	19	19.0	15.7
25	28	18.4	15.1	26	18.5	15.2
100	56	17.4	14.1	52	17.5	14.2
250	91	16.7	13.4	78	17.0	13.7
1000	156	16.0	12.7	148	16.0	12.7
2000	218	15.5	12.2	192	15.6	12.3
4000	328	14.9	11.6	272	15.1	11.8
8000				385	14.6	11.3

## 注意事项：

1. 典型电容噪声与分辨率对输出数据速率的关系，10pF基准+1pF跨度，快速稳定，MR1，V=3.0V。跨度指传感器在应用中的最大电容变化。表格显示在3.0V电源电压下，使用最大采样容量进行芯片内平均时，噪声的均方根(RMS)以fF为单位，随输出数据速率(Hz)变化。比特值为噪声与跨度的二进制对数(BITs =  $\ln(\text{span/noise})/\ln(2)$ )。测量采用PCap04评估板，配置为最大分辨率，使用固定COG陶瓷电容器。

传感器和参考都可以连接为“浮空”或“接地”，如图所示。浮空模式下，激活内部和外部杂散电容的补偿机制；接地模式下，仅激活内部补偿。

图11：  
电压依赖偏移误差 (PSRR) AMS数据手册

基准/增益	模式	2.4V	2.7V	3.0V	3.3V	3.6V
10 pF	单端， 内部补偿	< 1fF	< 1fF	关闭	< 1fF	< 1fF
150 pF		< 1fF	< 1fF	关闭	< 1fF	< 1fF
10 pF	浮动， 全补偿	< 1fF	< 1fF	关闭	< 1fF	< 1fF
150 pF		< 1fF	< 1fF	Off	< 1fF	< 1fF

图12：  
电压依赖增益误差 (PSRR)

基准/增益	模式	2.4V	2.7V	3.0V	3.3V	3.6V
10pF / 4.7pF	单端, 内部补偿	1.9fF	0.9fF	关闭	- 0.7fF	- 1.2fF
150pF / 47pF		- 22fF	- 10fF	关闭	0.7fF	20fF
10pF / 4.7pF	悬浮, 全补偿	0.6fF	6fF	关闭	- 0.7fF	- 1.5fF
150pF / 47pF		- 11fF	- 11fF	Off	8fF	19fF

图13：  
温度相关的偏移和增益误差

误差类型	电容 [pF]	模式	温度范围	典型漂移	
偏移漂移	10	单端, 内部补偿	-10°C 至 85°C	2.5fF	
	150			9fF	
	10	浮动, 全补偿		1.5fF	
	150			12fF	
增益漂移	10 + 4.7	单端, 内部补偿		42fF = 94ppm/K	
	150 + 47			69fF = 15ppm/K	
	10 + 4.7	浮动, 全补偿		8fF = 18ppm/K	
	150 + 47			96fF = 22ppm/K	

## RDC特性

图14：  
分辨率RDC单元

测量条件	R2/Rref 类型	RMS 噪声 R2/Rref	典型RMS噪声温度 (1)
无平均, 2次虚假测量	0.899	31.7ppm	12.2mK
16倍平均, 8次虚假测 量	0.897	20.2ppm	7.8mK
测量条件	10nF @ PTOUT, 25°C		

注意事项：  
1. 后处理软件中的线性化处理第  
14页

假设温度与电阻率呈线性关系，内部AI温度计线性化及转换为温度后的典型线性误差：

- $20^{\circ}\text{C} < \text{温度} < 0^{\circ}\text{C} \rightarrow 290\text{mK}$
- $0^{\circ}\text{C} < \text{温度} < 80^{\circ}\text{C} \rightarrow 110\text{mK}$

## 时序特性

图15：  
PCap04 时序特性

符号	参数	条件	最小值	类型	最大值	单位
起始时间	启动时间		3.9		4.0	ms
$t_C$	CDC放电时间	测量范围1	0		20	us
$t_R$	RDC放电时间	测量范围1	0		20	us
$f_{SPI}$	SPI总线频率	时钟频率	0		20	MHz
$f_{I^2C}$	I <sup>2</sup> C总线频率	数据传输速率	0	100		kHz

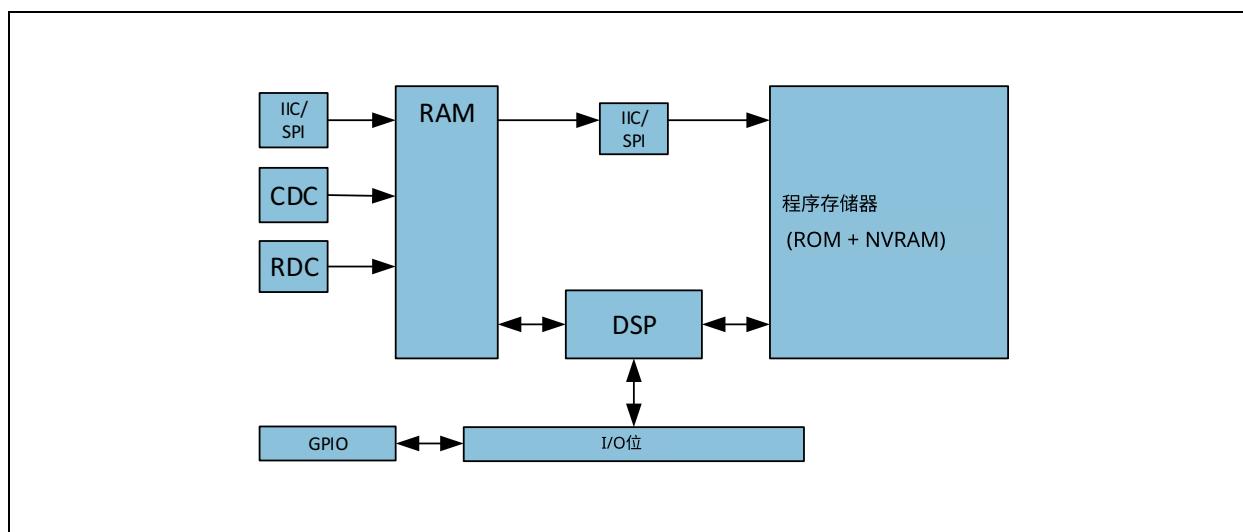
## 详细描述

PCap04是一款集成的电容式和电阻式传感器数字化解决方案，包括用于线性和温度补偿的数据处理DSP。6通道CDI允许在单端和差分模式下处理接地和浮地传感器。适用的电容范围从几皮法到数百纳法。RDC单元主要用于测量温度，通过内部传感器和参考，或使用外部电阻如PT1000。

哈佛架构的32位数字信号处理器（DSP）集成在PCap04中，负责接收CDC和RDC测量单元的数据，进行处理并提供给用户界面。CDC/RDC的原始数据以及DSP处理后的数据都存储在RAM中。DSP程序存放在NVRAM中。DSP可以从一组64I/O位中收集各种状态信息，并写回其中的16位。这样，DSP可以响应并控制PCap04的GPIO引脚。DSP的内部时钟频率约为60MHz，通过固件命令可以停止内部时钟以节省电能。DSP在接收到GPIO信号或“测量结束”条件时重新启动。

在最简单的形式中，DSP将纯时间测量信息从CDC/RDC传输到读寄存器，无需进一步处理。下一步是计算电容比，包括补偿测量中的信息，如ams标准固件版本PCap04\_standard\_v01.hex所提供的。最后，ams提供了现成的线性化固件，通过三次多项式进行线性化，并通过二次多项式进行温度补偿。许多线性化固件的功能块以ROM代码实现。这样，主固件可以非常紧凑，能够容纳在1k NVRAM中。

图16：  
CDC、RDC与DSP嵌入



读取寄存器的内容将始终取决于所使用的固件。使用标准固件时，它将是纯电容和电阻比值。使用线性化固件时，可能是线性化和校准后的结果，例如以帕斯卡为单位的压力或以百分比表示的湿度。

DSP是ams专有的，用于处理低功耗任务以及极高速数据传输。它采用汇编语言编程。配备图形界面的用户友好汇编软件、帮助提示弹出以及示例代码，支持编程工作。

## 寄存器描述

### 配置寄存器

PCap04提供48个寄存器用于配置硬件（CDC、RDC、时钟、PDM/PWM、DSP）。所有这些寄存器均为一字节大小。额外的四个寄存器用作特殊功能寄存器。48<sup>th</sup> 寄存器仅包含一个位，即RUNBIT，用于启用或禁用前端和DSP。

所有配置同时写入寄存器和NVRAM的RAM部分，并可读取回。

### 寄存器概述

图17：  
寄存器概述

地址	名称	<D7>	<D6>	<D5>	<D4>	<D3>	<D2>	<D1>	<D0>							
配置寄存器																
0	CFG0	12C_A		OLF_FTUNE				OLF_CTUNE								
1	CFG1	OX_DIS		OX_DIV4	OX_ 自动停 止距离	OX停止	OX运行									
2	CFG2	RDCHG内部选择1		RDCHG内部选择0		RDCHG 中断使能		RDCHG 外部使能								
3	CFG3		AUXPD 禁用	AUX _CINT	RDCHG_OPEN		RDCHG _PERM _EN	RDCHG_ EXT _PERM	RCHG _SEL							
4	CFG4	C_REF_I NT		C_ 扩展组 件	C_ 内部组 件			C_ 微分	C FLOATIN G							
5	CFG5	CY_PRE _MR1_S HORT		C端口模 式		CY_ HFCLK选 择	CY DIV4禁用	CY_PRE_ LONG	C_DC_ BALANCE							
6	CFG6			C_PORT_EN												
7	CFG7	C_AVRG<7:0>														
8	CFG8			C_AVRG<12:8>												
9	CFG9	转换时间<7:0>														
10	CFG10	转换时间<15:8>														
11	CFG11		转换时间<22:16>													
12	CFG12	放电时间<7:0>														

地址	名称	<D7>	<D6>	<D5>	<D4>	<D3>	<D2>	<D1>	<D0>			
13	CFG13	C_STARTONPIN			C_TRIG_SEL			放电时间 <9:8>				
14	CFG14	预充时间<7:0>										
15	CFG15			C_FAKE				预充时间 <9:8>				
16	CFG16	满充时间 <7:0>										
17	CFG17			C_REF_SEL				放电时间 <9:8>				
18	CFG18	C_G_OP_RUN	C_G_OP_EXT	C_G_EN								
19	CFG19	C_G_OP_VU		C_G_OP_ATTN		C_G_TIME						
20	CFG20	循环					C_G_OP_TR					
21	CFG21	R_TRIG_PREDIV<7:0>										
22	CFG22		R_TRIG_SEL			R_AVRG		R_TRIG_PREDIV <9:8>				
23	CFG23	R_PORT_EN		端口启用 IMES	端口启用 REF		伪造	启动引脚				
24	CFG24			TDC通道启用		TDC 超高开	噪声抑 制	运动速度				
25	CFG25	运动编号										
26	CFG26	QHA选择						噪声 循环抑 制				
27	CFG27	多功能启用				DSP速度		PG1x PG3	PG0x PG2			
28	CFG28	WD距离										
29	CFG29	DSP启动引脚				DSP前向输入						
30	CFG30	PG5_INTN_EN	PG4_INTN_EN			DSP_START_EN						
31	CFG31	PI1_切换英语	PIO切换 EN	PIO_RES		PIO PDM_SEL	PIO_CLK_SEL					
32	CFG32			PI1_RES		PI1_PDM_SEL	PI1_CLK_SEL					
33	CFG33	PG_DIR_IN				PG_PU						
34	CFG34	INT触发背景	DSP触发背景	BG PERM	AUTO START							

地址	名称	<D7>	<D6>	<D5>	<D4>	<D3>	<D2>	<D1>	<D0>
35	CFG35				CDC_GAIN_CORR<7:0>				
36	CFG36				-				
37	CFG37				-				
38	CFG38				背景时间				
39	CFG39			脉冲选择1			脉冲选择0		
40	CFG40				感知选择				
41	CFG41				反应选择				
42	CFG42		报警1选择		报警选择	EN ASYNC RD	HS MODE SEL	R MEDIAN EN	C MEDIAN EN
...									
47	CFG47								RUNBIT
48	CFG48						存储锁定		
49	CFG49				序列号<7:0>				
50	CFG50				序列号<15:8>				
...									
54	CFG54				存储控制				
...									
62	CFG62				充电泵<7:0>				
63	CFG63				充电泵<15:8>				

地址	名称	<D7>	<D6>	<D5>	<D4>	<D3>	<D2>	<D1>	<D0>
读取寄存器									
0	RES0	7							0
1		15							8
2		23							16
3		31							24
4	RES1	7							0
5		15							8
6		23							16
7		31							24
8	RES2	7							0
9		15							8
10		23							16
11		31							24
12	RES3	7							0
13		15							8
14		23							16
15		31							24
16	RES4	7							0
17		15							8
18		23							16
19		31							24
20	RES5	7							0
21		15							8
22		23							16
23		31							24
24	RES6	7							0
25		15							8
26		23							16
27		31							24

地址	名称	<D7>	<D6>	<D5>	<D4>	<D3>	<D2>	<D1>	<D0>
28	RES7	7							0
29		15							8
30		23							16
31		31							24
32	状态_0	PORFL LAG WDG	PORFL AGCO NFIG	IR FLAG COLL	AUTOBO OT		RDCR EADY	CDC ACTIVE	RUNBIT
33	状态_1					RDC_ERR	MUP ERR	ERR OVFL	COMB ERR
34	状态_2		C 端口错误 _内部	C PORTERR 5	C PORTERR 4	C PORTERR 3	C PORTERR 2	C PORTERR 1	C PORTERR 0

注意事项：

1. 寄存器35到42取决于固件。提供的数值适用于标准固件。
2. 结果寄存器的内容取决于固件。

## 详细配置寄存器描述

### 配置寄存器0 (地址0x0)

图18：  
寄存器0

位	位名称	设置	位描述
7:6	12C_A	0 to 3	对 I <sup>2</sup> C -地址的补充
5:2	OLF_FTUNE	0 : 最小值 7 : 典型值, 建议值15 : 最大值	微调低频时钟
1:0	OLF_CTUNE	0 : 10kHz 1 : 50kHz 2 : 100kHz 3 : 200kHz	粗调低频时钟

### 配置寄存器1 (地址0x1)

图19：  
寄存器1

位	位名称	设置	比特描述
7	OX_DIS	默认: 0	禁用OX时钟
5	OX_DIV4	0: 不分频; f_ox = 2MHz 1: 分频4; f_ox = 0.5MHz	OX时钟频率: 原始频率/4
4	OX_AUTOSTOP_DIS	默认值: 0	ams内部位
3	OX_STOP	默认值: 0	ams内部位
2:0	OX运行	0 : 发生器关闭 6 : OX延迟 = 1 / fOLF3 : OX 延迟 = 2 / fOLF2 : OX延迟 = 31 / fOLF1 : OX持续运行	控制OX发生器的持续性或延迟。延迟指测量开始前振荡器的稳定时间

### 配置寄存器2 (地址0x2)

图20：  
寄存器2

位	位名称	设置	位描述
7:6	RDCHG_INT_SEL1	0: 180kΩ 1: 90kΩ 2: 30kΩ (默认) 3: 10kΩ	相同, 但用于端口PC4 - PC5
5:4	RDCHG_INT_SELO		为CDC端口PC0 - PC3及内部端口PC6选择一个片上放电电阻
3	RDCHG_INT_EN	0: 关闭 1: 开启 (默认)	启用内部放电电阻
1	RDCHG_EXT_EN	0: 关闭 (默认) 1: 外部开启	在放电阶段启用PCAUX的外部放电电阻切换 (在预充和满充阶段为高阻抗) 注意: AUX_PD_DIS (PCAUX下拉禁用) : 1

### 配置寄存器3 (地址0x3)

图21：  
寄存器3

位	位名称	设置	位描述
6	AUX_PD_DIS	0: 启用下拉 1: 禁用下拉	禁用PCAUX的下拉
5	AUX_CINT	0: 正常 (默认) 1: 在内部c参考转换期间辅助激活	仅在内部c参考转换期间激活辅助端口PCAUX
4:3	RDCHG_OPEN	2: 推荐	ams内部位
2	RDCHG_PERM_EN	0: 关闭 (默认) 1: 开启	将芯片内部放电电阻永久连接。
1	RDCHG_EXT_PERM	0: 关闭 (默认) 1: 开启	永久激活辅助端口PCAUX, 用于a)永久放电或b)为每个充放电周期添加偏置电容。 注意: AUX_PD_DIS (PCAUX下拉禁用): 1
0	RCHG_SEL	0: 180kΩ 1: 10kΩ (默认)	在两个芯片内充电电阻中选择一个, 用于CDC, 限制充电电流, 避免瞬态

## 配置寄存器4 (地址0x4)

图22：  
寄存器4

位	位名称	设置	比特描述
7	C_REF_INT	0: 外部参考 (在PC0/GND或PC0/PC1) 1: 内部参考	在CDC特殊端口PC6使用片上参考电容
5	C_COMP_EXT	0: 空闲 1: 激活; 当C_FLOATING==0时必须避免	激活补偿机制以减小芯片外寄生电容
4	C_COMP_INT	0: 空闲 1: 激活	激活芯片寄生电容的补偿机制及增益补偿
1	C_DIFFERENTIAL	0: 普通1: 差分	选择单端或差分传感器
0	C_FLOATING	0: 接地1: 浮空	选择接地传感器或浮动传感器

## 配置寄存器5 (地址0x5)

图23：  
寄存器5

位	位名称	设置	位描述
7	CY_PRE_MR1_SHORT	0: 正常 (推荐) 1: 减小	减少内部时钟路径之间的延迟
5	C_PORT_PAT	0: 正常 1: 端口交替顺序	每个序列后测量端口的顺序将被反转。如果激活C_PORT_PAT，则C_AVRG + C_FAKE应为偶数
3	CY_HFCLK_SEL	0: OLF1 1: OHF	CDC的时钟源
2	CY_DIV4_DIS	0: 关闭 1: 开启	四倍时钟周期 (仅在CY_HFCLK_SEL == 1时生效)
1	CY_PRE_LONG	0: 关闭, 推荐1: 开启	在内部时钟路径之间增加安全延迟
0	C_DC_BALANCE	0: 关闭 (单一高阻) 1: 直流无 (双高阻)	仅适用于差分浮空模式 (其他模式为直流无), 更改端口控制以消除容量感应的直流

注意事项：

1. CY\_HFCLK\_SEL 和 CY\_DIV4\_DIS 在评估软件中合并为“周期时钟选择”

## 配置寄存器6 (地址0x6)

图24:  
寄存器6

位	位名称	设置	位描述
5:0	C_PORT_EN	0x00: 全部关闭, CDC 将无法工作; 0x01: 仅 端口PC0激活等; 0x3F: 所有端口激活	按位启用CDC端口, 从PC0到PC5, 位0 对应端口PC0, 位1对应PC1等

## 配置寄存器7:8 (地址0x7, 0x8)

图25:  
寄存器7:8

位	位名称	设置	位描述
12:0	C_AVRG	0,1 : 样本量 = 12 : 样本量 = 2 3 : 样本 量 = 3 ...8191 : 最 大样本量	用于平均 (计算均值) CDC测 量的样本量

## 配置寄存器 11:9 (地址 0x9;0xA;0xB)

图26:  
寄存器 11:9

位	位名称	设置	位描述
22:0	CONV_TIME	关于CDC, 触发测量 的特定时间段 $T_{conv./seq} = 2 * CONV$ TIME[.] / fOLF	转换触发周期或: 序列周期 (拉伸模式下)

## 配置寄存器12 (地址0xC)

图27:  
寄存器12

位	位名称	设置	位描述
7:0 & 寄存器 13:1:0	放电时间	OLF: $T_{\text{discharge}} = (\text{DISCHARGE\_TIME} + 1) * T_{\text{cycleclock}}$ OHF: $T_{\text{discharge}} = (\text{DISCHARGE\_TIME} + 0) * T_{\text{cycleclock}}$ 1023:关闭	设置CDC放电时间。此充电时间间隔用于放电时间测量的预留空间。

## 配置寄存器13 (地址0xD)

图28:  
寄存器13

位	位名称	设置	位描述
7:6	C_STARTONPIN	0: PG0, 1: PG1, 2: PG2, 3: PG3	选择允许触发CDC启动的GPIO端口
4:2	C_TRIG_SEL	0: 连续, 1: 读取触发, 2: 定时器触发, 3: 扩展定时器触发, 4: 无定义, 5: 引脚触发, 6: 操作码触发 (7: 连续扩展, 不推荐)	CDC触发模式
1:0 & Reg12:7: 0	放电时间	参见寄存器12	参见寄存器12

## 配置寄存器14 (地址0xE)

图29:  
注册 14

位	位名称	设置	位描述
7:0 & 寄存器 15:1:0	预充时间	OLF: $T_{\text{precharge}} = (\text{PRECHARGE\_TIME} + 1) * T_{\text{cycleclock}}$ OHF & FULLCHARGE_TIME = 1023: $T_{\text{precharge}} = (\text{PRECHARGE\_TIME} + 2) * T_{\text{cycleclock}}$ OLF & FULLCHARGE_TIME!= 0x3FF: $T_{\text{precharge}} = (\text{PRECHARGE\_TIME} + 1) * T_{\text{cycleclock}}$ 1023:关闭	设置CDC放电时间Tprecharge-时间间隔, 用于放电时间测量的保留区。

### 配置寄存器15 (地址0xF)

图30:  
寄存器15

位	位名称	设置	位描述
5:2	伪造	0: 无 1 假的... 15: 15 假货	"假"或"热身"次数 在"真实"测量之前进行的CDC测量； "假"值不计入
1:0 & Reg14 :7:0	预充电时间	参见寄存器14	参见寄存器14

### 配置寄存器16 (地址0x10)

图31:  
寄存器16

位	位名称	设置	位描述
7:0 & 寄存器 17:1:0	完全充电时间	OLF: $T_{fullcharge} = (FULLCHARGE\_TIME * t_{fullcharge}) / T_{cycleclock}$ OHF: $T_{fullcharge} = (FULLCHARGE\_TIME * 2) * T_{cycleclock}$	设置CDC放电时间 保留用于放电时间测量的时间间

### 配置寄存器17 (地址0x11)

图32:  
寄存器17

比特	比特名称	设置	比特描述
1:0 & 寄存器 16:7:0	完全充电时间	参见寄存器16	参见寄存器16
6:2	C_REF_SEL	0: 最小值1: 近似1 pF... 31: 最大值 (近似 31 pF)	设置片上 CDC的参考电容器注意：步宽从 0.3 pF 变化到 1.5 pF

## 配置寄存器18 (地址0x12)

图33：  
寄存器18

位	位名称	设置	位描述
7	C_G_OP_RUN	保护：操作模式	0：永久 1：脉冲（在转换之间将操作设置为睡眠模式）
6	C_G_OP_EXT	守卫：激活外部操作	0：内部操作 1：外部操作， PG3 作为 C_G_MUX_SEL
5:0	C_G_EN	守卫使能，每个端口	b'xxxxx1 : 激活端口 PC0b'xxxx1x : 激活端口 PC1b'xxx1xx : 激活端口 PC2b'xx1xxx : 激活端口 PC3b'x1xxxx : 激活端口 PC4b'1xxxxx : 激活端口PC5

### 配置寄存器19 (地址0x13)

图34：  
寄存器19

位	位名称	设置	比特描述
7:6	C_G_OP_VU	守卫：操作增益（从SensePort到 守卫）	0 : x 1.001 : x 1.012 : x 1.023 : x 1.03
5:4	C_G_OP_ATTN	守卫：操作衰减	0 : 0.5aF1 : 1.0aF2 : 1.5aF3 : 2.0aF
3:0	C_G_TIME	保护：时间段 预充电：将保护端口从“直接连 接”切换到OP	t : C_G_TIME * 500ns

### 配置寄存器20 (地址0x14)

图35：  
寄存器20

位	位名称	设置	位描述		
7	R_CY	根据OLF频率的RDC预充/充电/ 放电周期时间	OLF频率	R_CY=0	R_CY=1
			10kHz	100微秒	200微秒
			50千赫兹	20微秒	40微秒
			100千赫兹	10微秒	20微秒
			200千赫兹	20微秒	40微秒
2:0	C_G_OP_TR	守卫操作当前修正	0 ... 7 : 推荐		

## 配置寄存器21 (地址0x15)

图36：  
寄存器21

位	位名称	设置	位描述
7:0 & Reg22 :1:0	R_TRIG_PREDIV	0, 1 : 每个信号触发 2: 每2个 <sup>nd</sup> 信号触发 3: 每 3 <sup>rd</sup> 信号触发.....1023: 最大系数	预分频器，允许进行比电容测量更低温度的测量。这是CDC与RDC测量速率之间的系数。若使用OLF作为触发源，也用作OLF时钟分频器。

## 配置寄存器22 (地址0x16)

图37：  
寄存器22

位	位名称	设置	位描述
6:4	R_TRIG_SEL	0: 关闭 1: 定时器触发3: 引脚触发5: CDC异步 (推荐) 6: CDC同步	RDC5和6的触发源选择：由CDC转换结束触发
3:2	R_AVRG	0: 不平均1: 4次 平均2: 8次平均 3: 16次平均	RDC部分平均值计算的样本大小
1:0 & Reg21 :7:0	R_TRIG_PREDIV	0,1: 每个信号触发2: 每两个信号触发3: 每个 <sup>nd</sup> 信号触发...1023: 最大系数	预分频器，用于实现比电容测量更低的温度测量。这是CDC与RDC测量速率之间的系数，也作为OLF时钟分频器，如果OLF用作触发源。

### 配置寄存器23 (地址0x17)

图38：  
寄存器23

位	位名称	设置	位描述
7:6	R_PORT_EN	'bx0 : 禁用 'bx1 : 激活端口 PT0REF 'b0x : 禁用 'b1x : 激活端口 PT1	RDC部分的端口激活
5	R_PORT_EN_IMES	0: 禁用1: 启用	内部铝温度传感器端口激活
4	R_PORT_EN_IREF	0: 禁用1: 启用	内部参考电阻端口激活
3	-	-	-
2	R_FAKE	平均值的伪周期数：0.2；伪周期数：8	在实际测量之前进行的“伪”或“预热”测量次数； “伪”值不计入正式数据
1:0	R_STARTONPIN	0: PG01: PG1,2: PG23: PG3	选择允许触发RDC启动的GPIO端口

### 配置寄存器24 (地址0x18)

图39：  
寄存器24

位	位名称	设置	位描述
7:6	-	-	-
5:4	TDC通道使能	必填：3	ams内部位
3	TDC_ALUPERMOPEN	必填：0	ams内部位
2	TDC噪声抑制	必填：0	ams内部位
1:0	TDC_MUPU速度	必填：3	ams内部位

**配置寄存器25 (地址0x19)**

图40：  
寄存器25

位	位名称	设置	位描述
7:2	TDC_MUPU_NO	必填：1	ams内部位
1:0	-	-	-

**配置寄存器26 (地址0x1A)**

图41：  
注册26

位	位名称	设置	位描述
7:2	TDC_QHA_SEL	必填：20	ams内部位
1	TDC_NOISE_CY_DIS	必填：1	ams内部位
0	-	-	-

**配置寄存器27 (地址0x1B)**

图42：  
寄存器27

位	位名称	设置	位描述
7:6	DSP_MOFLO_EN	0: 关闭 3: 开启	在GPIO脉冲线上启用单稳态触发器（防抖滤波器）。
5:4	-	-	-
3:2	DSP_SPEED	0: 最快1: 快 2: 慢, 推荐3: 最慢	DSP速度
1	PG1xPG3	0: 在PG3输出脉冲; 1: 在PG1输出脉冲	切换PG1/PG3的接线到/从DSP
0	PG0xPG2	0: 在PG2输出脉冲; 1: 在PG0输出脉冲	切换PG0/PG2与DSP的接线

### 配置寄存器28 (地址0x1C)

图43：  
寄存器28

位	位名称	设置	位描述
7:0	WD_DIS	0x5A : 关闭看门狗 (关闭) 0x00 (推荐) /否则: 开启看门狗	禁用看门狗, 要禁用看门狗, 必须将0x5A写入此寄存器。看门狗周期在9秒到15秒之间

### 配置寄存器29 (地址0x1D)

图44：  
寄存器29

位	位名称	设置	位描述
7:4	DSP_STARTONPIN	按位设置PG0到PG3	启动DSP的引脚掩码——允许将一个或多个GPIO引脚分配给启动DSP
3:0	DSP_FF_IN	按位设置DSP_IN_0到DSP_IN_3	触发器激活的引脚掩码

### 配置寄存器30 (地址0x1E)

图45：  
寄存器30

位	位名称	设置	位描述
7	PG5_INTN_EN	0: PG5正常工作 1: PG5 <= INTN	将INTN信号路由到PG5
6	PG4_INTN_EN	0: PG4正常工作 1: PG4 <= INTN	将INTN信号路由到PG4
5:3	-	-	-
2:0	DSP启动使能	'bxxx1 : 由CDC结束触发'bxx1x : 由RDC结束触发 (推荐) 'bx1xx : 由定时器触发	DSP触发使能

## 配置寄存器31 (地址0x1F)

图46：  
寄存器X

位	位名称	设置	位描述
7	PI1_TOGGLE_EN	0: 正常操作1: 切换触发器激活	在脉冲接口1输出激活切换触发器，特别用于PDM以实现1:1占空比
6	PIO_TOGGLE_EN	0 : 正常操作1 : 切换触发器激活	在PulseInterface 0输出激活切换触发器，特别用于PDM以实现1:1占空比
5:4	PIO_RES	0 : 10位1 : 12位2 : 14位3 : 16位	脉冲编码接口的分辨率
3	PIO_PDM_SEL	0 : PWM1 : PDM	脉冲接口0 PWM / PDM切换
2:0	PIO_CLK_SEL	0 : 关闭 1 : OLF / 12 : OLF / 23 : OLF / 44 : OX / 15 : OX / 26 : OX / 47 : 未定义	脉冲接口0时钟选择

## 配置寄存器32 (地址0x20)

图47：  
寄存器32

位	位名称	设置	位描述
7:6	-	-	-
5:4	PI1_RES	0: 关闭 1: OLF / 12: OLF / 23: OLF / 44: OX / 15: OX / 26: OX / 47: 未定义	脉冲接口1时钟选择
3	PI1_PDM_SEL	0 : PWM1 : PDM	脉冲编码接口的分辨率
2:0	PI1_CLK_SEL	0 : 10位 : 12位2 : 14位3 : 16位	脉冲接口1PWM / PDM切换

## 配置寄存器33 (地址0x21)

图48：  
寄存器33

位	位名称	设置	位描述
7:4	PG_DIR_IN	0 : 输出1 : 输入	切换通用端口方向为输入或输出#4: PG0#5: PG1#6: PG2#7: PG3
3:0	PG_PU	0 : 禁用上拉1 : 启用上 拉	在通用端口激活保护性上拉电阻#0: PG0#1: PG1#2: PG2#3: PG3

## 配置寄存器34 (地址0x22)

图49：  
寄存器34

位	位名称	设置	位描述
7	INT_TRIG_BG	0: 禁用 1: 启用	读取触发带隙结束
6	DSP_TRIG_BG	0: 禁用 1: 启用	带隙刷新由DSPbit设置触发
5	BG_PERM	1: 永久启用带隙 0: 脉冲式带隙	永久激活带隙。设置BG_PERM = 1时，电流消耗大约增加 20μA
4	AUTOSTART	0: 禁用 1: 电源开启后触发CDC	单机操作时，在电源开启后触发CDC
3:0	-	强制: 7	ams内部位

## 配置寄存器35 (地址0x23)

图50：  
注册35

位	位名称	设置	位描述
7:0	CDC_GAIN_CORR[7:0]	推荐 1.25 ==> 0x40	增益校正系数的固件定义配置。 8fpp0.0n的第0到7位：1 + n/256

## 配置寄存器36 (地址0x24)

图51：  
寄存器36

位	位名称	设置	位描述
7:0	-		未使用

### 配置寄存器37 (地址0x25)

图52：  
寄存器37

位	位名称	设置	位描述
7:0	-		未使用

### 配置寄存器38 (地址0x26)

图53：  
寄存器38

位	位名称	设置	位描述
7:0	BG_TIME	0: 推荐	固件定义

### 配置寄存器39 (地址0x27)

图54：  
寄存器39

位	位名称	设置	位描述
7:4	PULSE_SEL1	0到5: Res0到Res5 (C0..5/Cref @PCap04_standard) 6 : Res6 (PT1/Ref @PCap04_standard) 7 : Res7 (Alu/Ref @PCap04_standard)	固件定义，选择PulseIF 1的信号源
3:0	PULSE_SELO	0到5: Res0到Res5 (C0..5/Cref @PCap04_standard) 6 : Res6 (PT1/Ref @PCap04_standard) 7 : Res7 (Alu/Ref @PCap04_standard)	固件定义，选择PulseIF 0的源

### 配置寄存器40 (地址0x28)

图55：  
寄存器40

位	位名称	设置	位描述
7:0	C_SENSE_SEL	PCap04_linearize 仅固件，选择 C 线性化比例0..5: C0到5 / Cref	由固件定义

## 配置寄存器41 (地址0x29)

图56：  
寄存器41

位	位名称	设置	比特描述
7:0	R_SENSE_SEL	仅限PCap04线性化固件， 选择 $R$ 比率用于温度测定	由固件定义

## 配置寄存器42 (地址0x30)

图57：  
寄存器42

位	位名称	设置	位描述
7	-	PCap04_linearize固件 专用极性选择 0: Z; 1: Theta	固件定义。选择PG0和PG1的报警 信号源，主动高电平
6	ALARM1_SELECT		
5	-		
4	ALARMO_SELECT		
3	EN_ASYNC_READ	1：激活	结果寄存器Res0至Res7中的值仅 在前一个值已读取后才会更新
2	HS_MODE_SEL	0：强制	ams内部位
1	R_MEDIAN_EN	PCap04_linearize固件 专用	在线性化固件中启用ci/ri的中值滤 波器
0	C_MEDIAN_EN		

## 配置寄存器47 (地址0x2F)

图58：  
寄存器47

位	位名称	设置	位描述
7:1	未使用		
0	RUNBIT	0：关闭=芯片系统空闲且受保 护1：开启=解除保护，系统可 以运行	前端和DSP的开关：在编程和修 改寄存器时应保持“关闭”，以保 护芯片免受不良或未定义状态 的影响

### 配置寄存器48 (地址0x30)

图59：  
寄存器48

位	位名称	设置	位描述
7:4	未使用		
3:0	MEM LOCK	'bxxx1 : NVRAM范围 `d0 到 703'bxx1x : NVRAM范围 `d704 到 831'bx1xx : NVRAM范围 `d832 到 959'b1xxx : NVRAM范围 `d960 到 1007 和NVRAM 范围 `d1022 到 1023	数据安全功能，用于通过SIF保护 NVRAM的安全部分，防止读取和写入

### 配置寄存器49 (地址0x31)

图60：  
注册49

位	位名称	设置	位描述
7:0	SERIAL_NUMBER[7:0]	客户免费处理。只能在字节为零时写入。之后可以通过完全擦除清除	低字节保留用于序列号

### 配置寄存器50 (地址0x32)

图61：  
寄存器50

位	位名称	设置	位描述
7:0	序列号[15:8]	客户免费处理。仅在字节为零时可写入。之后可通过完全擦除清除	高字节用于序列号保留

配置寄存器51到53：ams内部寄存器，  
0x00为必需  
配置寄存器54（地址0x36）

图62：

位	位名称	设置	位描述
7:0	MEM_CTRL	0x2d: NVRAM存储使能 0x59: NVRAM调用使能 0xb8: NVRAM擦除寄存器 在后续SIF操作后自动复位	存储控制

寄存器54配置寄存器55至61：ams内部  
寄存器，0x00为强制值  
配置寄存器62（地址0x3e）

图63：  
寄存器62

位	位名称	设置	位描述
7:0	充电泵[7:0]	单独的、设备特定设 置。不允许更改	NVRAM充电泵微调的低字节

配置寄存器63（地址0x3f）

图64：  
寄存器63

位	位名称	设置	位描述
7:0	充电泵[15:8]	个体设备特定设置。不 允许更改	NVRAM充电泵微调高字节

重要提示：我们仅在客户不更改寄存器62和  
63的前提下，保证数据的保持和耐久性。此  
外，必须严格按照第NVRAM和ROM部分中描  
述的擦除NVRAM的流程操作，否则我们将不  
再保证数据保持时间和耐久周期。

### 读取寄存器

PCap04具有35字节的RAM用于读取，按4字节的四元组组合。

图65：  
读取寄存器

地址	名称	<D7>	<D6>	<D5>	<D4>	<D3>	<D2>	<D1>	<D0>
0	RES0	7							0
1		15							8
2		23							16
3		31							24
4	RES1	7							0
5		15							8
6		23							16
7		31							24
8	RES2	7							0
9		15							8
10		23							16
11		31							24
12	RES3	7							0
13		15							8
14		23							16
15		31							24
16	RES4	7							0
17		15							8
18		23							16
19		31							24
20	RES5	7							0
21		15							8
22		23							16
23		31							24

地址	名称	<D7>	<D6>	<D5>	<D4>	<D3>	<D2>	<D1>	<D0>
24	RES6	7							0
25		15							8
26		23							16
27		31							24
28	RES7	7							0
29		15							8
30		23							16
31		31							24
32	状态_0	PORF LAG WDG	PORFL AGCO NFIG	IR FLAG COLL	AUTOBO OT		RDCR EADY	CDC_ ACTIVE	RUNBIT
33	状态_1					RDC 错误	MUP 错误	溢出 错误	组合错 误
34	状态_2		C PORTE RR_INT	C PORTE RR5	C_ PORTER R4	C PORTE RR3	C PORTE RR2	C PORTE RR1	C PORTE RR0

读取寄存器由7个结果寄存器组成。地址32到34包含状态寄存器。

### 结果寄存器

结果寄存器的内容取决于固件。以下描述了标准固件中使用的结果寄存器。

图66：  
结果寄存器

名称	长度	格式	位描述	
			接地C	差分C
Res0	32位	例如 C0 = 无符号 定点数：5 位整数27 位小数最小值 = 0x0 = = 0.0000000 最大值 = 0xFFFFFFFF = = 31.9999995	比率 C0 / Cref	比率 C1 / C0
Res1			比率 C1 / Cref	C3/C2
Res2			C2 / Cref	C5/C4
Res3			C3 / Cref	
Res4			C4 / Cref	零
Res5			C5 / Cref	零
Res6			PT1/PTref	PT1/PTref
Res7			PTinternal/PTref	PTinternal/PTref

用户可以在其固件中自由将任何数据分配给结果寄存器。

### 状态寄存器

图67：  
状态 -0 (地址32) amm

位	位名称	位描述
7	POR_Flag_Wdog	检测到看门狗溢出，已引发上电复位。可能固件卡在了不必要的无限循环中，或更可能是CDC/RDC触发信号丢失。
6	POR_Flag_Config	一个或多个配置位被干扰触发，导致上电复位。
5	POR_CDC_DSP_COLL	在DSP仍然激活时触发CDC序列会引发初始复位。
4	AutoBoot繁忙	
3		
2	RDC就绪	
1	CDC 活动	警告：接口上的流量可能增加测量中的噪声
0	RUNBIT	写寄存器47的RUNBIT在此处镜像

图68：  
状态1 (地址33)

位	位名称	位描述
7:4	n.c.	测试位
3	RDC_Err	RDC单元繁忙时发生某种错误
2	Mup_Err	CDC单元繁忙时发生特定类型的TDC错误
1	Err_Ovfl	CDC单元繁忙时发生溢出错误
0	Comb_Err	所有错误位，从此处起以位或方式组合

图69：  
STATUS\_2 (地址34)

位	位名称	端口	位描述
6	C_PortError 内部引用	PC内部引用	在CDC单元中，一个或多个端口受到类似短路接地的错误影响。也可能是充放电电阻过大、电容过大，或预充/满充/放电时间定义不明确。
5	C_PortError5	PC5	
4	C_PortError4	PC4	
3	C_PortError3	PC3	
2	C_PortError2	PC2	
1	C_PortError1	PC1	
0	C端错误0	PC0	

## 工作原理

### 转换器前端

该设备采用“放电时间测量”作为测量电容（CDC单元）或电阻（RDC单元）的原理。它以时间多路复用的方式访问所有端口（PC...，PT...），通过高分辨率的TDC（时间到数字转换器）进行时间测量。

### 电容到数字转换器（CDC）

#### 测量原理

在PCap04中，电容测量通过测量RC网络的放电时间实现。测量采用比值法，即将电容与固定参考或在差分传感器中与方向相反的电容进行比较。由于短时间间隔和特殊补偿方法，放电时间比值与电容比值成正比。放电时间由电容和所选放电电阻决定。

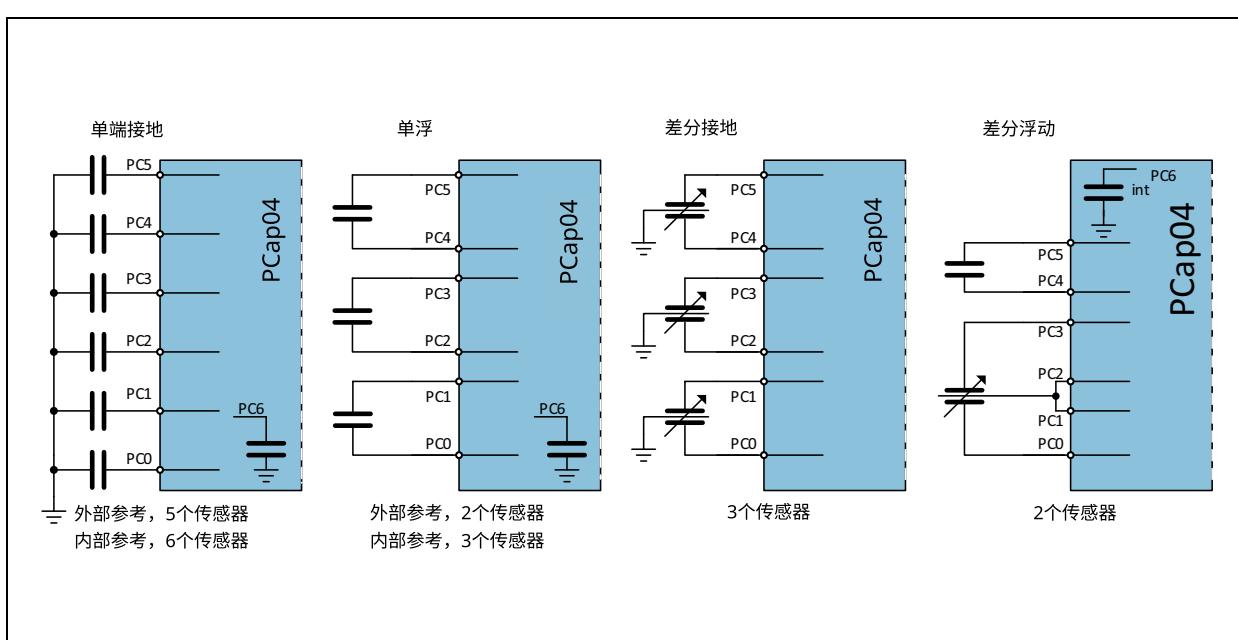
$$(EQ1) \frac{\tau_N}{\tau_{ref}} = \frac{C_N}{C_{ref}} \quad \tau = k \times R \times C$$

### 连接传感器

PCap04可连接接地或浮空的单端和差分传感器。

PCap04内置参考电容，范围可调，步长为1pF，从1pF到31pF。

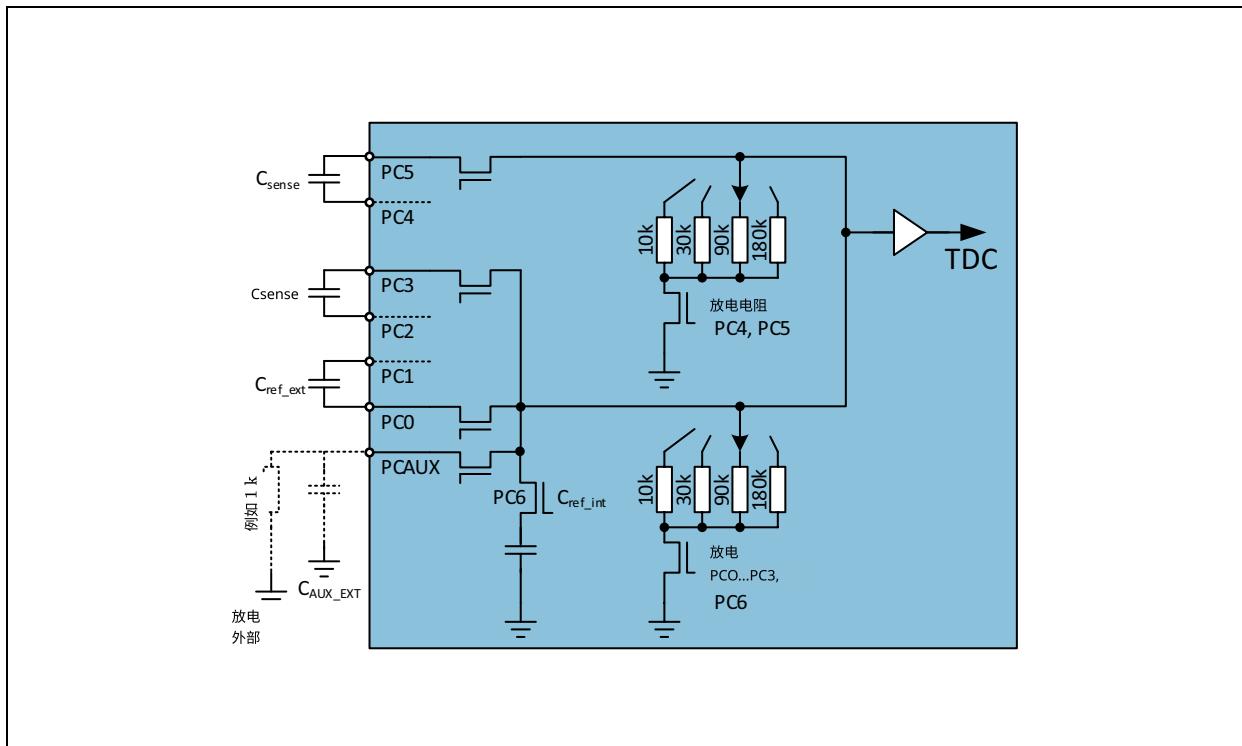
图70：  
连接传感器



## 放电电阻器

PCap04已集成两组放电电阻器。一组电阻用于测量端口PC0至PC3和内部参考端口PC6。另一组电阻用于端口PC4和PC5。这样可以用同一芯片测量具有较大偏差的不同传感器，如压力和湿度。参数RCHG\_xxx选择电阻。循环

图71：  
集成放电电阻器



用户可以定义用于内部补偿测量的电阻器。通过  
DSP\_RDCHG\_COMP\_INT\_SEL选择 (0 = RCHG0 (默认), 1 = RCHG1)。  
可以使用外部放电电阻器来处理大电容。

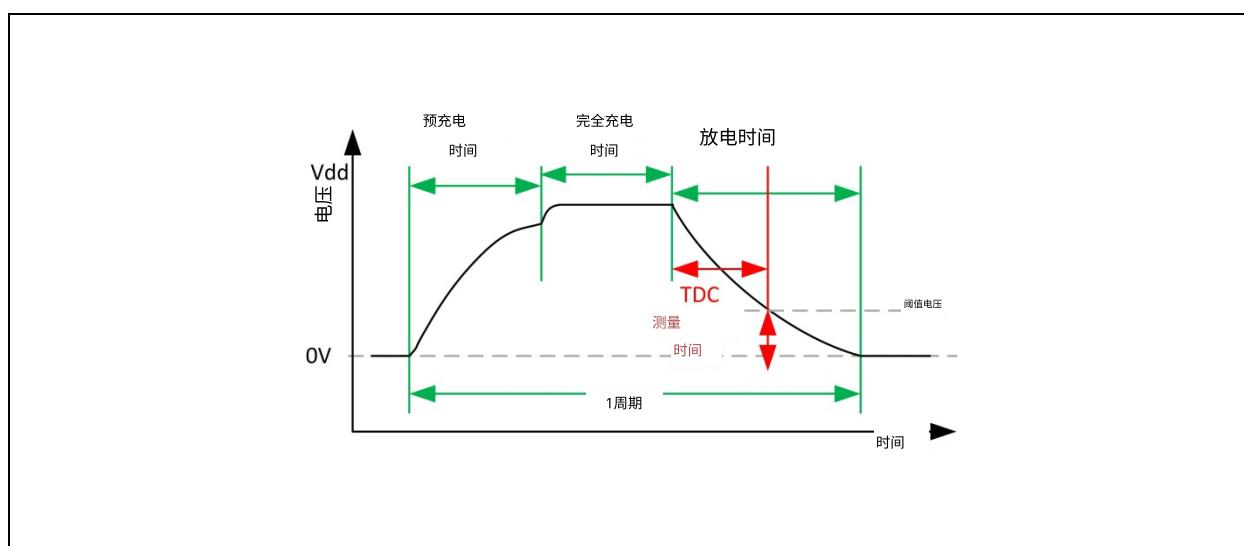
在PCap04中，测量原理基于三步循环。

- 在预充电阶段，电容通过串联电阻充至接近Vdd的电平。电阻减小充电电流，降低传感电容的机械应力。在某些MEMS应用中这是必要的。此外，这也是检测短路和限制电流的措施，即使在出现此类故障时也能起作用。

- 在完全充电阶段，电容最终在没有串联电阻的情况下充至Vdd。

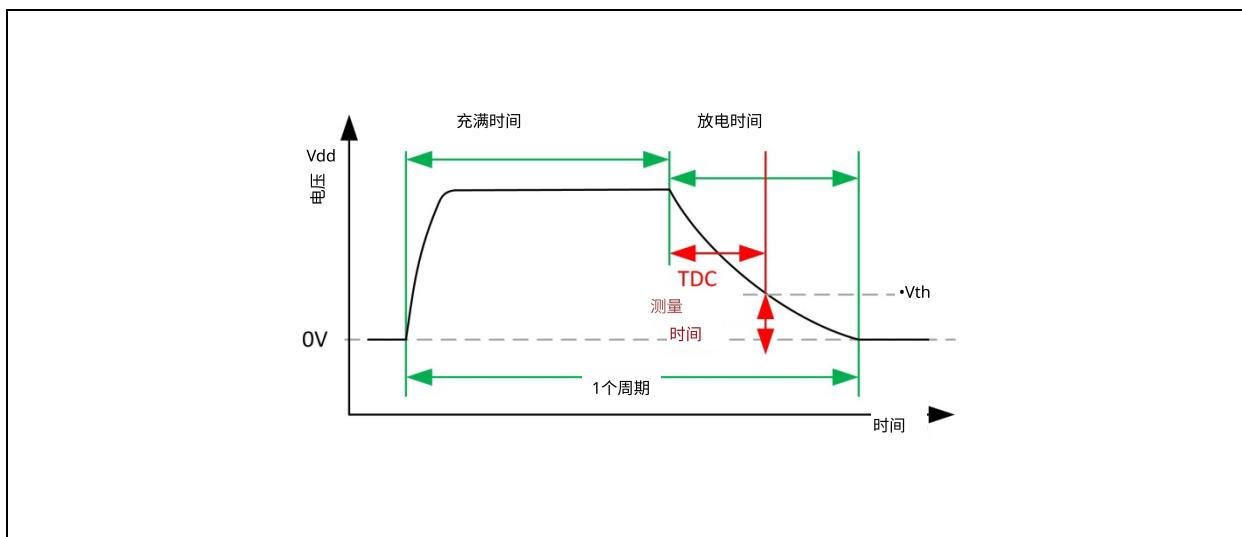
- 然后，在第三步中，电容通过放电电阻放电至0V。CDC测量直到触发电平被达到的时间间隔。所有这些过程统称为一个“周期”。

图72：  
单周期时序



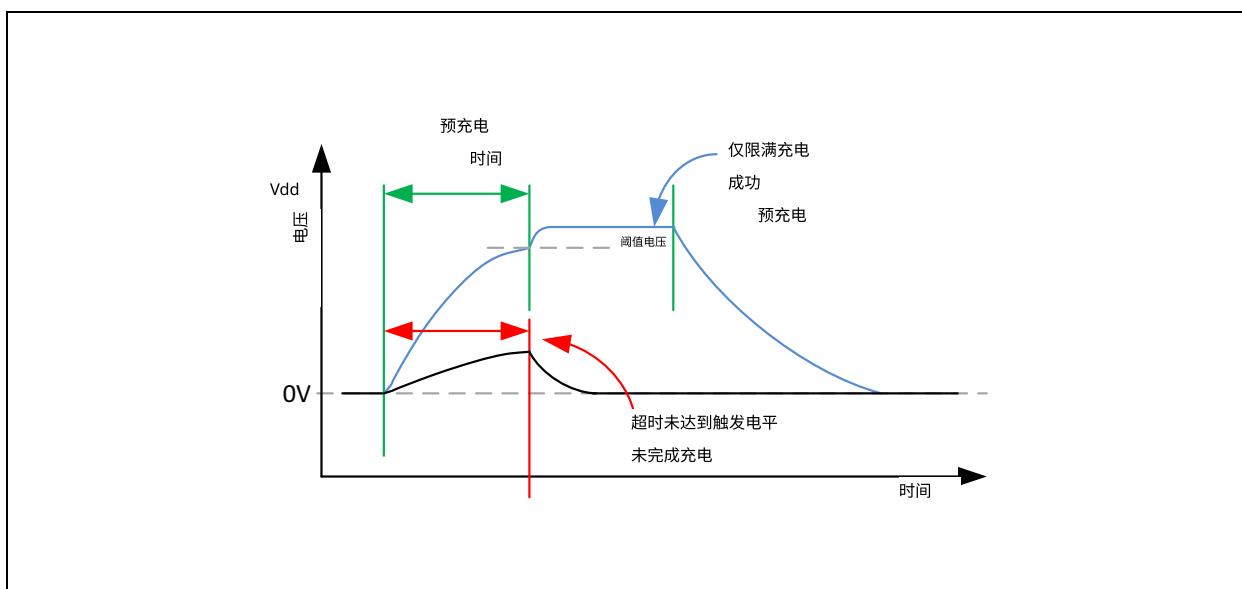
在不需要缓慢充电而追求高转换效率的应用中，可以禁用预充电选项，直接开始充电，无需串联电阻。

图73：  
单次循环，快速充电



两种情况下电容器都在放电时间内完全放电，然后连接到地。  
在短路情况下，电压从未达到比较器的触发电平。  
在这种情况下，测量周期停止，不会发生低阻抗的满充电。这样可以限制短路时的电流。

图74：  
短路检测



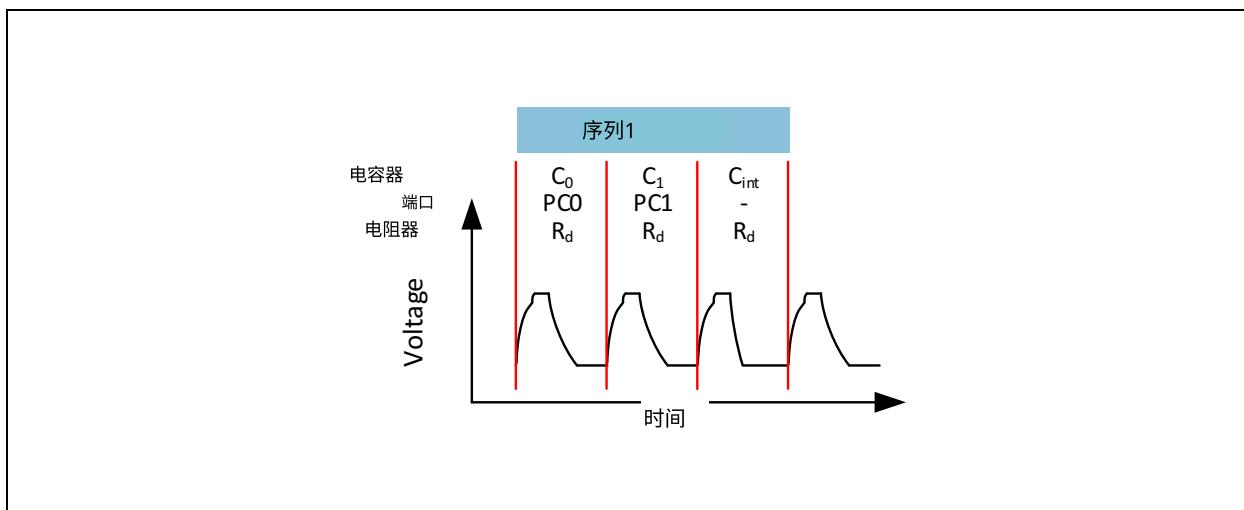
## 序列

“序列”由一组循环组成，包括各个有源端口的循环以及由补偿测量所示的它们的组合。单个循环的数量和类型取决于传感器的连接方式、电容器的数量以及所选的补偿选项。

对于接地传感器，序列总是以PC0（参考）开始，然后是一个或多个其他5个端口。通常，内部补偿被激活。因此，序列以测量内部寄生电容/延迟的Cint结束。为了补偿内部寄生电容和比较器延迟，CDC在所有端口关闭（Cint）时测量放电时间。

下图显示了带有内部补偿的接地传感器的序列。

图75：  
接地序列

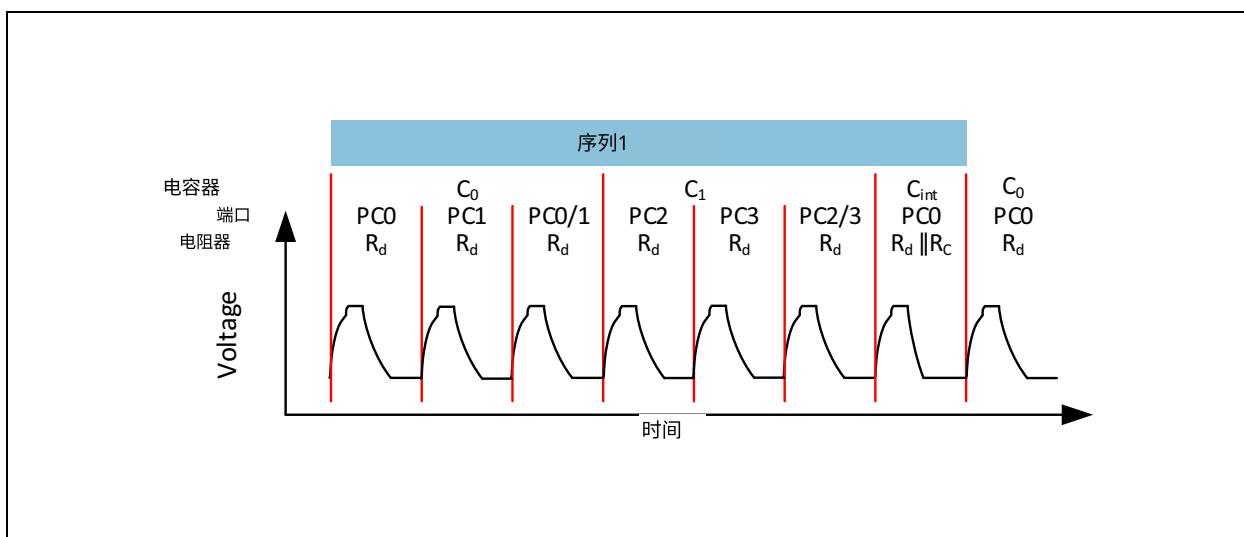


接地连接中1个参考和1个传感器的序列，补偿内部电容

对于浮动传感器，序列始终以PC0/PC1（参考）开始，随后是一个到三个端口对用于传感器。通常，内部和外部补偿都被激活。

为了补偿外部寄生电容，CDC对每个电容进行测量，两个端口都被打开。因此，每个电容进行3次测量，例如PC0, PC1 和 PC0 + PC1。序列以内部补偿测量Cint结束。下图显示了具有全补偿的单个浮动传感器的测序过程。

图76：  
序列浮动

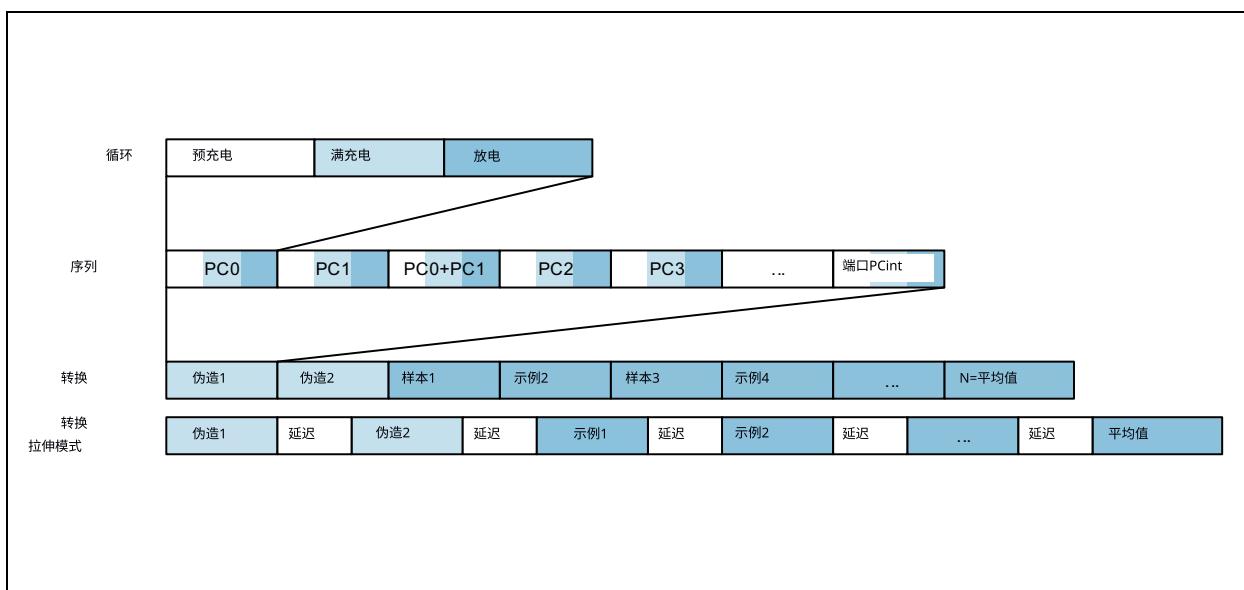


浮接方式下1个参考和1个传感器的序列，完全补偿寄生电容

### 转换

最后，各种序列及其间的延时组合定义了一个“转换”。一旦触发，转换自动完成，包括所有虚假测量和按采样量进行平均的真实测量。转换结束后，测量结果可用于后续处理和读取。转换结束由标志通知DSP和RDC单元。

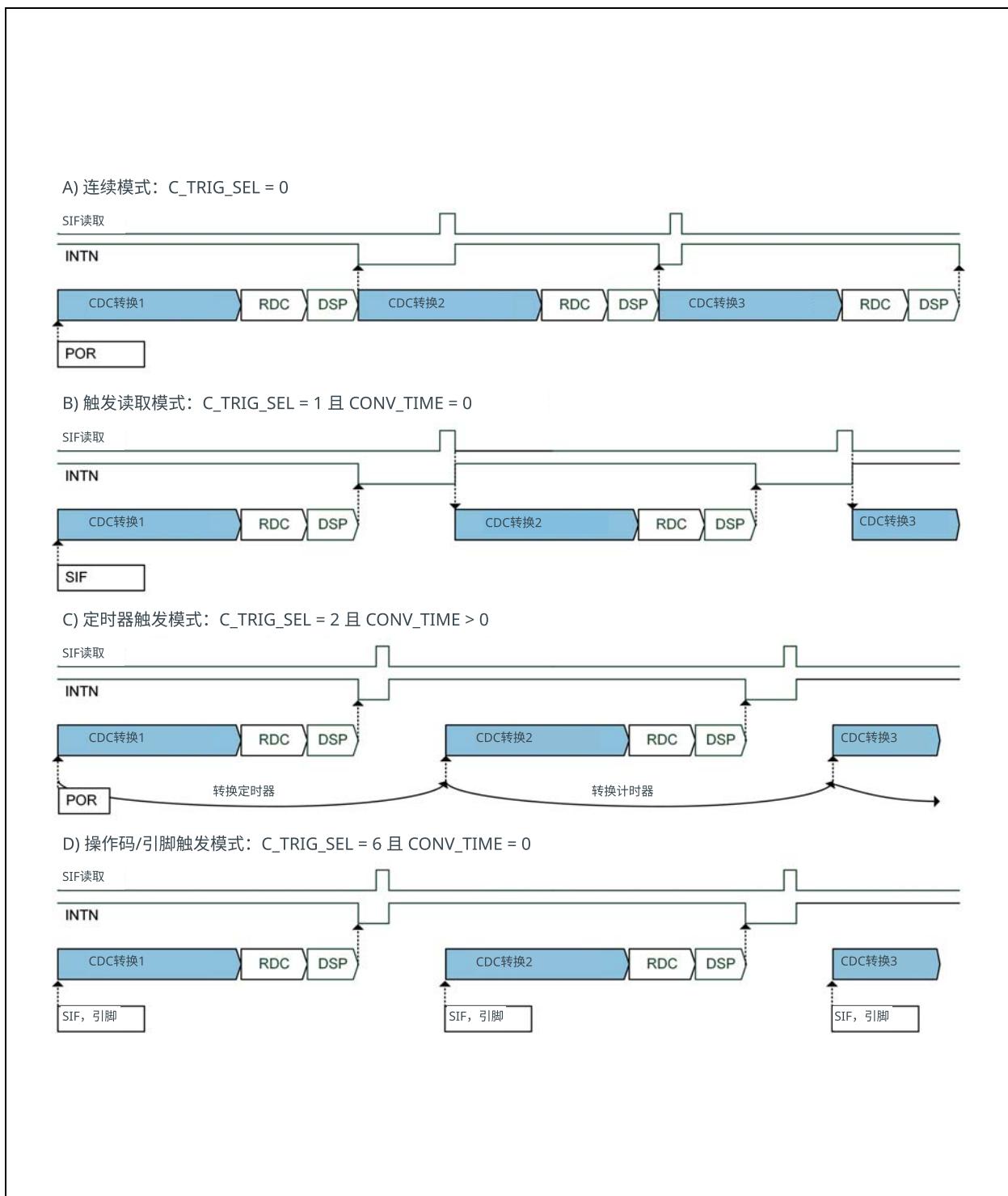
图77：  
周期-序列-转换



可以触发CDC的多种来源可用。

- 连续模式：**在此模式下，CDC在上电复位后自动启动。DSP处理结束后会直接触发下一次测量。此模式允许最高采样速率，但存在通信在启动下一次测量时持续进行的风险，噪声会增加。
- 读取触发模式：**在此模式下，首次测量通过SIF触发。测量完成后中断信号变为低电平。外部微控制器可以对此做出反应并读取数据。通过SIF完成读取后会触发下一次测量。此模式实现最快的测量速度，且不会被接口干扰测量过程。
- 定时器触发模式：**在此模式下，PCAP04定时器触发测量。适用于低采样速率的应用场景。
- 操作码或引脚触发模式：**在此模式下，外部微控制器完全控制何时触发测量。当测量需要与其他任务同步时，此方式尤为有用。

图78：  
转换模式



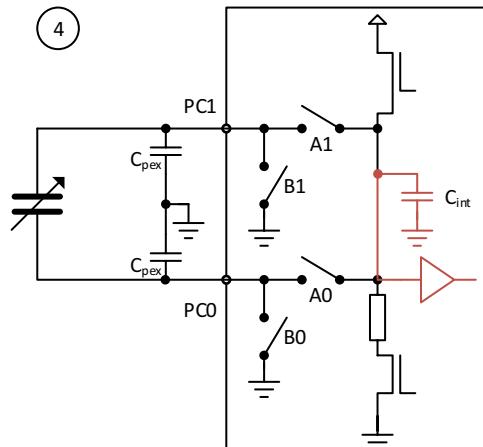
### CDC补偿选项

#### 内部补偿

对于内部补偿测量，A1和A0开关均断开。仅测量内部寄生电容和比较器的传播延迟。

建议在任何应用中启用内部补偿。

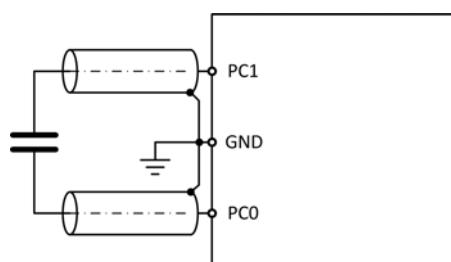
图79：  
内部补偿测量



#### 外部补偿

使用浮动电容器，我们可以额外补偿接地的外部寄生电容。在PCB上，导线电容通常指向地。对于较长的导线，建议使用屏蔽层，并在PCB端接地。

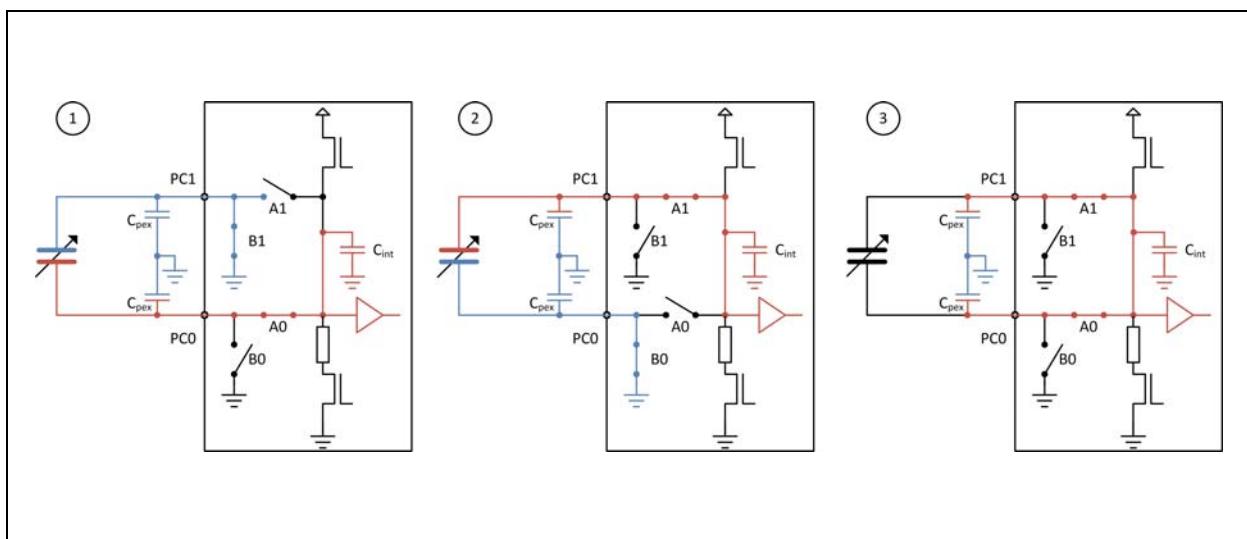
图80：  
屏蔽电缆 amm



如何连接屏蔽电缆以补偿外部寄生电容。

对于浮动传感器的每个电容器，需进行三次测量。见图81。首先，将PC0上的电极加载到VDD18。放电时间由传感器电容、连接的寄生电容（包括芯片焊盘）以及内部电容决定。第二次，对PC1上的电极进行相同测量。第三次，将两个电极都设置为VDD18，因此传感器两端的场强为零，不产生影响。放电时间仅包括连接电容、焊盘电容和内部电容。现在可以通过数学方法校正寄生电容，此校正由ams固件实现。

图81：  
外部补偿



浮动电容器、外部补偿测量，为每个浮动电容器进行的三次测量。

#### 直流平衡

在驱动浮动传感器时，传感器的供电通常是直流无干扰的。

使用差分浮动传感器会破坏对称性。因此，PCap04具有通过端口控制使传感器以直流无干扰方式工作（由C\_DC\_BALANCE设置）的可能性。在接地传感器的应用中，传感器原则上无法实现直流无干扰。

### 增益校正

与经典的模数转换器类似，PCap04再次显示出误差。但在PCap04中，增益误差主要由内部寄生电容和内部比较器的传播延迟引起。启用内部补偿后，该延迟会从原始测量中减去。温漂可以近似为线性，并通过一个增益因子进行数学校正。在标准固件中，增益校正因子可以在寄存器35中通过fpp 8设置。校正因子取决于放电时间及其RC组合。每个应用都需单独评估。例如，使用 22pF 和 30kΩ时，校正因子为1.25 (0x40)。CDC\_GAIN\_CORR = (gain\_corr-1)\*256。找到合适增益校正因子的方法：

用与参考电容相同尺寸（陶瓷COG）的温度稳定电容器替换传感器。（因此：商 = 1，增益 = 0）。将增益校正系数设为1.0。将系统（PCB上的PCap04）放入温度箱中，测量温度变化引起的偏移漂移。添加额外的温度稳定电容以模拟你的增益。测量增益漂移。增加增益校正系数，再次测量增益漂移。增益校正系数大于1.0时，漂移会减小。如果校正系数设置过大，会出现负的增益漂移，原因是过度补偿。找到合适的增益校正系数后，漂移会减小到你在初始偏移漂移测量中得到的水平。将新的CDC\_GAIN\_CORR值写入寄存器35。

### CDC 重要参数

#### 周期时钟

定义周期时间的基本周期  $t_{cycle}$  可以通过低频振荡器或高频振荡器导出。参数CY\_HFCLK\_SEL在两者之间选择，参数CY\_DIV4\_DIS在原始 2MHz 或由分频器4倍生成的 0.5MHz 之间选择。

图82：  
周期时钟配置

CY_HFCLK_SEL	CY DIV4 DIS	周期时间基准
0	0	$t_{cycle} = t_{OLF}$ ; $t_{OLF}$ = 周期低频振荡器
1	0	$t_{cycle} = 4*t_{OHF}$ ; $t_{OHF}$ = 周期高频振荡器。
1	1	$t_{cycle} = t_{OHF}$ ; $t_{OHF}$ = 周期高频振荡器。

## 周期时间

单个周期的预充电、满充电和放电时间以  $t_{cycle}$  的倍数定义。其选择方式为：

图83：  
周期时间配置

调节	配置参数	描述
14,15	预充时间	通过电阻进行电流限制的充电时间。取决于周期时钟，以及在OHF模式下，取决于FULLCHARGE_TIME。1023 = 无预充阶段OLF: 0到1022: $T_{precharge} = (\text{PRECHARGE\_TIME} + 1) * t_{cycle}$ OHF & FULLCHARGE_TIME = 1023: 0到1022: $T_{precharge} = (\text{PRECHARGE\_TIME} + 2) * t_{cycle}$ OHF & FULLCHARGE_TIME != 1023: 0到1022: $T_{precharge} = (\text{PRECHARGE\_TIME} + 1) * t_{cycle}$
16,17	完全充电时间	无电流限制的最终充电时间。取决于循环时钟。1023 = 无完全充电阶段OLF: 0 到 1022: $T_{fullcharge} = (\text{FULLCHARGE\_TIME} + 1) * t_{cycle}$ OHF: 0 到 1022: $T_{fullcharge} = (\text{FULLCHARGE\_TIME} + 2) * t_{cycle}$
12,13	放电时间	放电电容器的时间。取决于循环时钟。OLF: 0 到 1023: $T_{discharge} = (\text{DISCHARGE\_TIME} + 1) * t_{cycle}$ OHF: 0 到 1023: $T_{discharge} = (\text{DISCHARGE\_TIME} + 0) * t_{cycle}$

### 序列

序列的长度取决于传感器的类型和数量、所选补偿方法以及平均样本大小。以下参数影响序列：

图84：  
序列配置

注册	配置参数	描述
6	C_PORT_EN	电容端口PC0到PC50的按位使能：端口禁用1：端口激活
4	C_REF_INT	在外部和内部参考电容器之间切换。不能与差分传感器同时使用。0：外部，PC0或PC0 & PC11：内部，PC6
4	C_DIFFERENTIAL	在单端和差分传感器之间切换0：单端1：差分
4	C_FLOATING	在接地和浮地传感器之间切换0：接地1：浮地
4	C_COMP_INT	开启内部电容/延迟补偿0：关闭1：开启，推荐
4	C_COMP_EXT	开启外部寄生电容补偿，仅适用于浮动传感器。0：关闭1：开启，推荐
5	直流平衡	改变端口控制以实现差分浮动模式0：关闭（单高阻）1：直流无偏（双高阻）

## 转换

完整转换的持续时间由假测量次数、平均值和序列间延迟共同决定，具有下限：

图85：  
转换配置

寄存器	配置参数	描述
15	伪造	伪测量次数（忽略结果的周期）0：无伪周期1：1个伪周期...15：15个伪周期
7, 8	平均值	单次转换的采样数量。0 := 无平均...8191：最大采样数

下一次转换的开始取决于测量触发的选择：

图86：  
转换配置

寄存器	配置参数	描述
13	C_TRIG_SEL	CDC触发器0的首次触发选择： 连续1：读取触发2：定时器触发 3：定时器触发（拉伸）4：无数据5：引脚触发6：操作码触发 (7：连续_exp，不推荐)
13	C_STARTONPIN	选择触发CDC测量的GPIO
9, 10, 11	CONV_TIME	设置转换时间，单位为低频时钟周期的两倍

### 屏蔽

根据传感器拓扑结构，可能需要添加主动屏蔽以抑制干扰电容。屏蔽电极是位于感应电极后方的额外金属区域，与感应电极保持相同电位。因此，感应电极与屏蔽电极之间的所有材料都是无电位的，因此没有电容效应。它消除了传感器PCB上材料引起的温度漂移。此外，屏蔽后面的区域通过无电位区与传感器隔离开来。

在PCap04中，屏蔽驱动器集成在芯片内。该放大器需要具有低电容输入，以不干扰测量路径。增益理想值为1。可以通过C\_G\_OP\_VU实现 $> 1$ （过补偿），以便结合外部电压分压器匹配端口和线阻抗。屏蔽连接到引脚PCAUX（该引脚的其他功能将不可用）。非激活端口也会切换到屏蔽状态，以避免屏蔽与非激活端口之间出现额外电容。

图87：  
浮动电容屏蔽

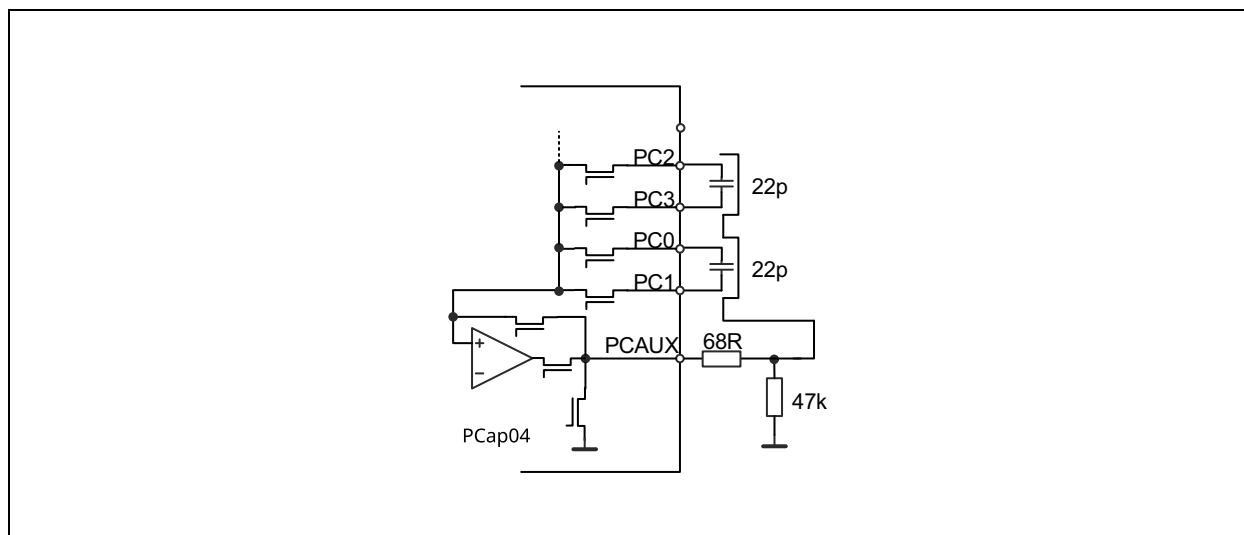


图88：  
接地电容保护

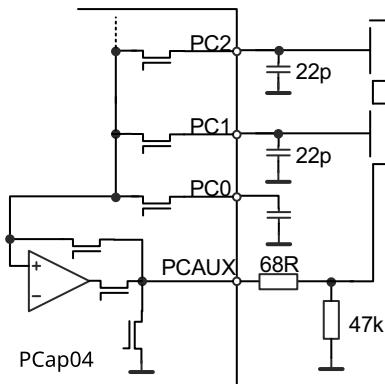
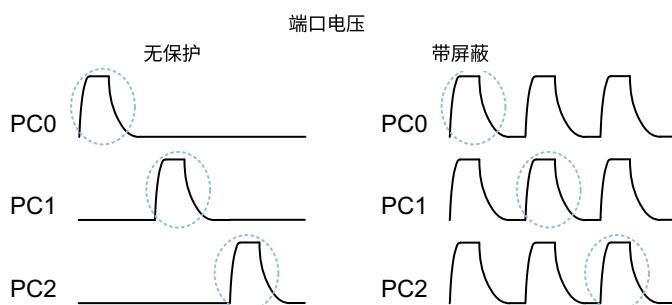


图89：  
带保护的端口电压



带屏蔽时，所有端口均设置为VDD18，包括未测量的端口。这确保这些端口与激活端口之间的电场为零。内部连接方面，这些端口不连接到时间测量路径，而是连接到屏蔽驱动器。如果屏蔽电极过大，无法由内部放大器驱动，可以添加外部放大器，还需要一个外部模拟多路复用器。外部放大器连接到PCAUX，多路复用器的SEL端口连接到PG3。

图90：  
外放大器保护

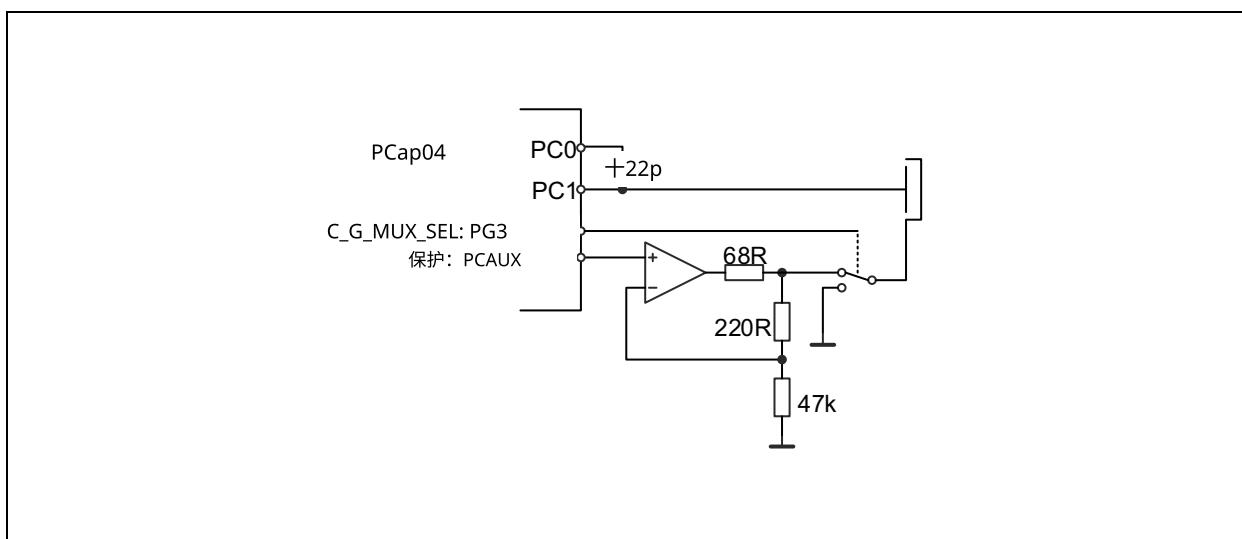


图91：  
寄存器18中设置的重要参数：

寄存器	配置参数	描述
18	C_G_EN[5:0]	为每个端口PC0至PC5启用单独的保护
18	C_G_OP_RUN	0：永久：保护操作持续激活（额外功耗）1：脉冲：保护操作在CDC转换间进入休眠模式
19	C_G_OP_ATTN	保护操作的电容衰减：0：0.5pF 1：1.0pF 2：1.5pF 3：2.0pF
19	C_G_OP_VU	OP增益（从感应端口到保护端口）0：1.001：1.012：1.023：1.03
20	C_G_OP_TR	调节保护OP的功耗和驱动强度0：最小...7：最大
18	C_G_OP_EXT	在内部保护OP和可选外部OP之间切换0：内部OP1：外部OP， PG3作为C_G_MUX_SEL
19	C_G_TIME	$t_{pp1} = t_{OHF} \cdot C\_G\_TIME$

- C\_G\_EN[5:0] 保护功能对每个端口PC0至PC5单独激活。可以启用一个或多个端口。保护输出（PCAUX）提供电压，启用的端口被激活。

- C\_G\_TIME 控制预充电阶段。由于内部电路，预充电阶段分为两个阶段：

- 预充电阶段1：PCAUX直接连接到激活的PC。

- 预充电阶段2：PCAUX由运算放大器驱动，  
 $V_{PCAUX} = gain\_guardOP - V_{PCactive}$ 。PCAUX由运算放大器驱动，直到完成当前端口周期。

注意：进行保护时，内部OX/OHF是强制要求：

- OX\_RUN > 0, OX\_DIS = 0; OX\_STOP = 0; OX\_DIV4 = 0;

- CY\_HFCLK\_SEL = 1;

### RDC 电阻-数字转换器

#### 测量原理

在PCap04中，电阻测量也通过测量放电时间进行。测量是比率式的。这意味着，温度敏感电阻与固定参考值进行比较。放电时间的比率与电阻的比值成正比。放电时间由电阻和负载电容共同决定。

$$(EQ2) \quad \frac{\tau_N}{\tau_{ref}} = \frac{R_\theta}{R_{ref}} \quad \tau = k \times R \times C$$

#### 连接传感器

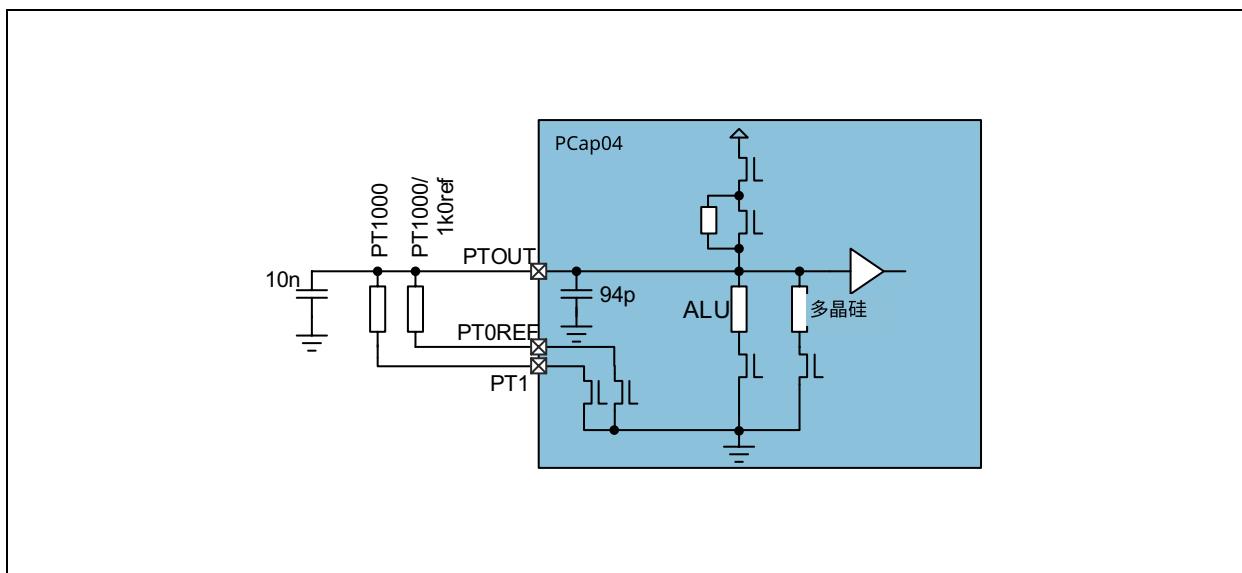
芯片器件具有两个片上电阻元件用于温度测量，一个铝条作为传感器，带有  $TK \approx 2800 \text{ ppm/K}$ ，另一个是多晶硅电阻，参考值接近“零”的TK。在范围  $0^\circ\text{C}$  到  $100^\circ\text{C}$  之间，铝传感器可以被良好地近似为温度的线性函数。

另一种方式是连接最多两个外部传感器。其中一个可以轮流用作外部参考。外部和内部温度计/参考可以混用，例如，外部PT1000可以与内部多晶硅电阻进行比较。

芯片内部具有大约  $94 \text{ pF}$  的电容。结合内部电阻，放电时间约为  $500 \text{ ns}$ ，电阻比的典型分辨率优于13位。为实现高精度测量，建议连接大约  $10 \text{ nF}$  的外部电容。使用  $10 \text{ nF}$  时，放电时间在  $20 \mu\text{s}$  左右，分辨率在15位左右。

放电时间不得超过  $20 \mu\text{s}$ 。对于电容器，陶瓷材料的性能最佳，而X7R材料的表现尚可。

图92：  
连接温度传感器



注意事项：

1. RDC测量基于交流原理。长电缆及其寄生电容和电阻会产生干扰，且建议使用短电缆 ( $\leq 0.5\text{ m}$ )，最好是扭绞屏蔽的。

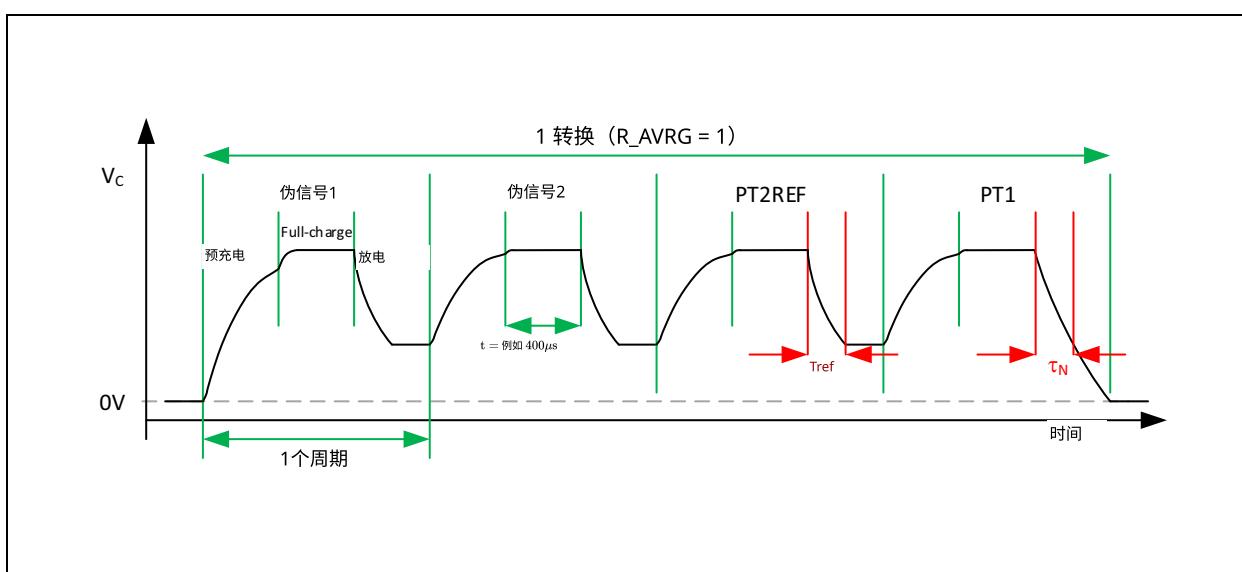
#### 循环与转换

在PCap04中，电阻测量以三阶段进行，类似于电容

#### 测量：

预充电 - 满充电 - 放电。时间由内部低频振荡器 (OLF) 控制。满充和放电阶段的持续时间可以是该参考的1或2个周期。转换开始时会进行2或8次虚假测量，以提高数据的稳定性。每次单次转换的平均值可以选择样本数1、4、8或16。

图93：  
RDC转换



RDC转换： $R_{AVRG} = 1$ ，参考和传感器，两个虚假测量

### 触发器

多种来源可以触发RDC。触发频率可以通过参数  $R_{TRIG\_PREDIV}$  (1到1023) 设置为CDC触发频率的分频值。  
参数  $R_{TRIG\_SEL}$  定义了触发阻抗测量的各种可能性：

- 串行接口命令、PIN或DSP (图64)
- 基于OLF的定时器触发 (图64)
- CDC转换结束
  - 异步：DSP由RDC转换结束触发。如果RDC速率低于CDC速率，DSP会直接由CDC触发以处理非活动的RDC转换 (图95)。
  - 同步：DSP由RDC转换结束触发。假设RDC速率低于CDC速率，非活动的RDC转换将被延迟取代 (图96)。

图94：  
RDC时序，由定时器、SIF或引脚触发

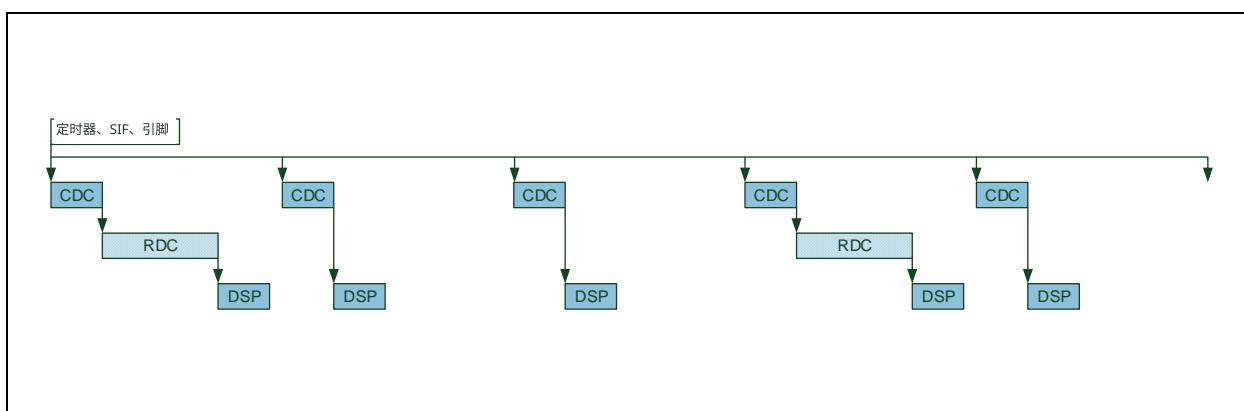
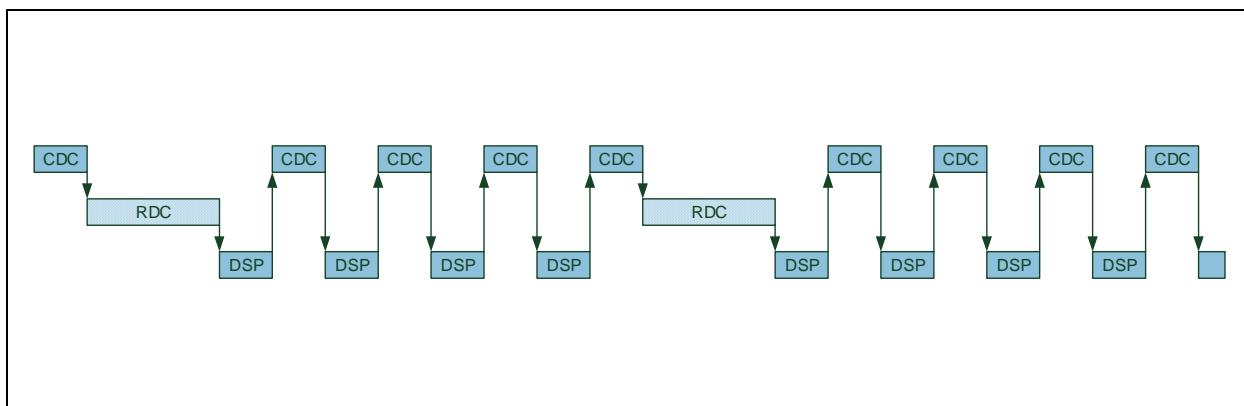
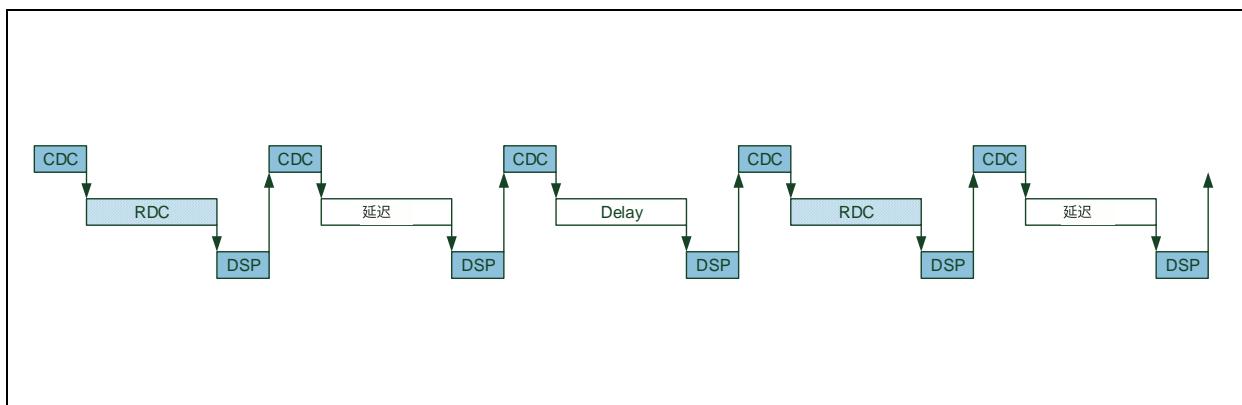


图95：  
异步模式下由DSP触发的RDC时序



$R_{TRIG\_PREDIV} = 3$ ,  $R_{TRIG\_SEL} = 3'b101$ ,  $DSP\_START\_EN$ :  $CDC\_TRIG\_EN = 0$ ,  $RDC\_TRIG\_EN = 1$

图96：  
同步模式下由DSP触发的RDC时序



`R_TRIGGER_PREDIV = 3, R_TRIGGER_SEL = 3'b110, DSP_START_EN: CDC_TRIGGER_EN = 1, RDC_TRIGGER_EN = 0`

### RDC关键参数

#### 周期时钟

温度测量的基准频率为低频振荡器 (OLF)。另一个位 R\_CY 指示是用1个还是2个周期定义三个阶段的长度。

图97：  
周期时钟配置

OLF频率	$t_{\text{precharge}} = t_{\text{fullcharge}} = t_{\text{discharge}}$	
	R_CY = 0	R_CY = 1
10 kHz	100 $\mu$ s	200 $\mu$ s
50 kHz	20 $\mu$ s	40微秒
100千赫	10微秒	20微秒
200千赫	20微秒	40微秒

## 序列

序列的主要设置包括端口数量、伪装、参考和平均。

图98：  
序列配置

调节	配置参数	描述
23	端口使能	启用端口PTOREF、PT1
23	内部参考电阻使能	启用内部参考电阻
23	内部参考电阻使能	启用内部温度传感器
22	R_AVRG	设置T测量的平均次数
23	R_FAKE	设置伪测量次数

## 转换

图99：  
RDC触发配置

寄存器	配置参数	描述
22	R_TRIG_SEL	RDC单元触发源选择
21, 22	R_TRIG_PREDIV	预分频器，用于将RDC速率设为CDC速率的分数，同时在选择OLF_CLK作为RDC触发源时也作用于OLF_CLK 0 = 1 : 每次CDC转换进行一次RDC转换2：每第二次CDC转换进行一次RDC转换...1023：最大设置
23	R_STARTONPIN	选择引脚进行引脚触发

## RDC结果，比例

PCap04\_standard 和 PCap04\_linearize 固件用于确定感测端口与参考端口之间的比例关系。

当内部参考端口被激活 (R\_PROT\_EN\_IREF: 1) 时，所有其他端口（内部感测、PT0 和 PT1）会自动选择内部参考作为比例依据。

如果 R\_PORT\_EN\_IREF: 0，则外部端口 PT0/Ref 被选为所有其他端口（内部感测、PT1）的参考值。

## 接口（串行 & PDM/PWM）

### 串行接口（SIF）

用于与微控制器通信和设备编程的串行接口有两种类型：SPI和IIC。一次只支持一种接口，通过引脚IIC\_EN选择。在两种接口上，PCap04只能作为从机运行。

图100：  
串行接口选择

引脚	描述
IIC_EN = 接地	4线SPI接口 通用I/O引脚PG0和PG1不可用
IIC_EN = VDD	2线 I <sup>2</sup> C 接口 所有通用I/O引脚均可用

IIC\_EN可能不能悬空。如无控制器接口需求，请将IIC\_EN连接至VDD。

串行接口允许对读寄存器（结果和状态）进行读取访问，对配置寄存器进行读写访问，以及对NVRAM（明确指的是NVRAM的SRAM部分）进行读写访问。

对存储器或配置/读寄存器的读写命令可以使用显式地址或地址自动递增方式。

amm 操作码 图  
101：操作码

描述	字节 2						字节 1					字节 0										
wr_mem (NVRAM)	1	0	1	0	0	0	add<9...0>(2)										data<7...0>					
rd_mem (NVRAM)	0	0	1	0	0	0	add<9...0>(2)										数据<7...0>					
写入 配置	1	0	1	0	0	0	1	1	1	1	添加<5...0>(2)			数据<7...0>								
读取 配置	0	0	1	0	0	0	1	1	1	1	添加<5...0>(2)			数据<7...0>								
读取结果 (RAM)	0	1	添加<5...0>(2) (3)						数据<7...0>													
上电复位	1	0	0	0	1	0	0	0														
初始化	1	0	0	0	1	0	1	0														
CDC 开始 转换	1	0	0	0	1	1	0	0														
RDC 开始 转换	1	0	0	0	1	1	1	0														
dsp触发	1	0	0	0	1	1	0	1														
nv存储(1)	1	0	0	1	0	1	1	0														
nv_recall(1)	1	0	0	1	1	0	0	1														
nv_erase(1)	1	0	0	1	1	1	0	0														
测试读取	0	1	1	1	1	1	1	0	0	0	1	0	0	0	1							

注意事项：

1. 在使用nv\_store、nv\_recall或nv\_erase前设置MEM\_CTRL
  2. 自动递增读写支持SPI和
- 读取结果的地址范围为0到 24 (8 × 32bit result registers and 3 × 8bit status registers)

串行接口最简单的测试方法是执行一次测试读取：

写入操作码0x7e到SIF并读取1字节。将此字节与以下模式进行比较：

- 0x11：预期值，读取周期正确执行
- 0x88：失败：存在大端/小端字节序交换
- 0xEE：失败：读取周期中所有位反转
- 0x77：失败：位反转且存在字节序交换

#### 同步读取

为了获得最佳效果，建议从由INTN信号同步的结果寄存器中读取所有值。可以通过寄存器30中的PG4\_INTN\_EN和PG5\_INTN\_EN将INTN信号路由到PG4或PG5。INTN信号为低有效，即INTN的负边沿表示Res0到Res7中的新值已就绪。INTN通过SSN（SPI）上的正边沿或停止条件被拉回高电平。

#### 异步读取

如果由于任何原因无法按上述方式同步读取，则必须在寄存器42中启用异步读取（EN\_ASYNC\_RD）。在此模式下，结果寄存器Res0到Res7中的值仅在前一次读取完成后更新（INTN通过SSN的正边沿或停止条件被复位为高电平）。

#### I<sup>2</sup>C兼容接口

本段概述了PCap04设备特定的I<sup>2</sup>C接口的使用。外部I<sup>2</sup>C主设备通过在SDA线出现下降沿时，SCL为高电平，发出起始条件，开始通信。通信通过在SDA线出现上升沿时，SCK为高电平，发出停止条件而结束。数据位在SCK的上升沿传输。

在I<sup>2</sup>C总线上，每个从设备拥有一个7位的设备地址，其中5位固定，2位可配置。该地址必须在起始条件后作为第一个字节发送，第八位指示后续数据传输的方向（R=读取=1，W=写入=0）。

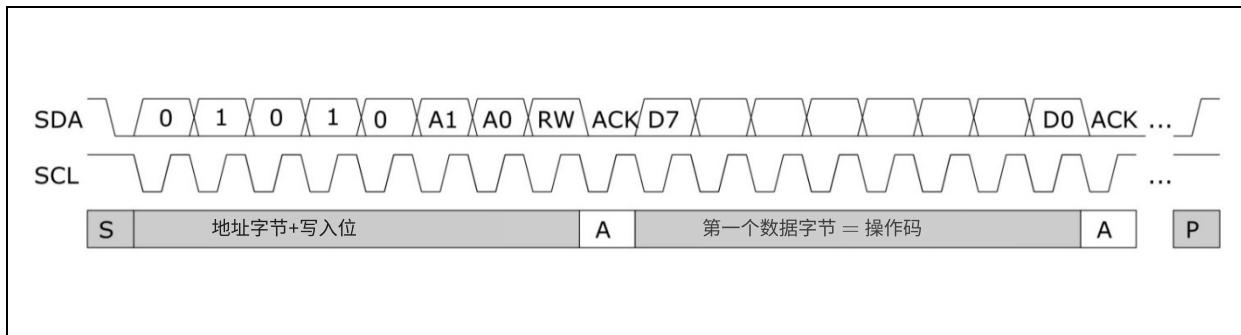
图102：  
地址字节

MSB							LSB
0	1	0	1	0	A1	A0	R/W
固定					可变		关键

$I^2C$  时间

地址字节后跟操作码，最终是有效载荷。每个字节后面跟一个确认位（= 0，当从设备确认时）。

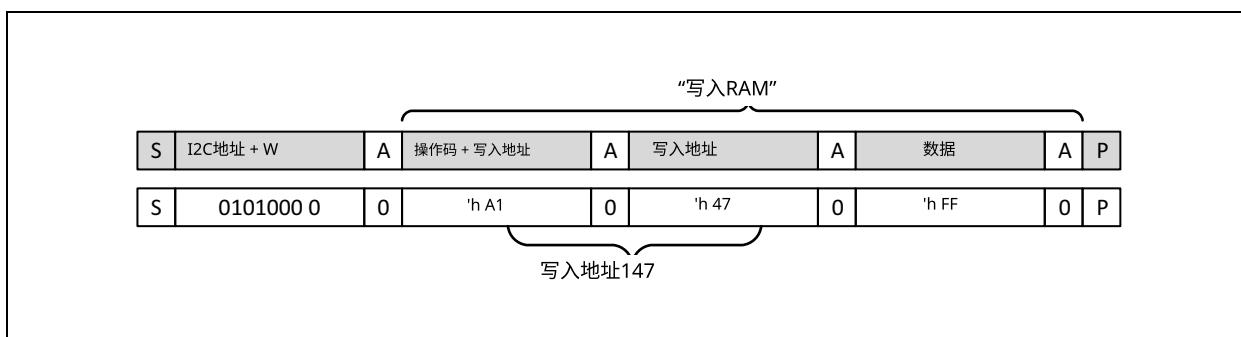
图103：  
I<sup>2</sup>C典型序列

I<sup>2</sup>C 写入

在写操作中，主设备单独发送数据，目标从设备仅返回确认位。主设备首先发送从设备地址和写入位，然后发送包括寄存器地址在内的PCap04特定操作码，最后发送有效载荷（“数据”）。

支持递增写入，即只需发送起始地址，连续的数据可以在一行中发送多个。

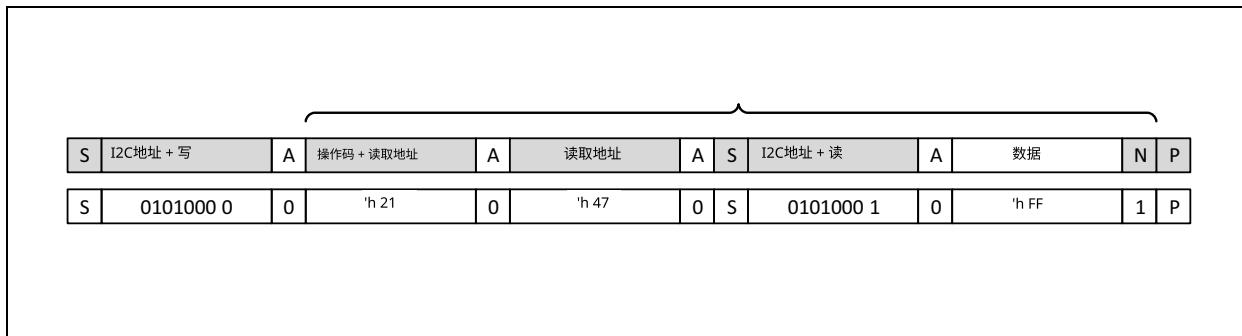
图104：  
I<sup>2</sup>C 写入流程  $I^2C$  读取



示例：将0xFF作为数据写入到地址0x147的SRAM中。

在读取事务中，通信方向必须切换。因此，主设备再次创建起始条件（或重启：中间不停止的起始条件）并发送从设备地址加读位以切换到读取模式。图105展示了一个“从SRAM读取操作码的示例。

图105：  
I<sup>2</sup>C 读取流程



示例：从SRAM地址0x147读取，发现之前已写入0xFF。

在接收到第一个（或任何）数据字节后，主机可以选择

- 非应答 = N = 1 表示“读取结束”，停止向从机发送，或
- 应答 = A = 0 表示“继续自动地址递增模式”，从而连续接收多个字节。如所见，自动地址递增在 I<sup>2</sup>C 接口中尤为实用高效。

#### SPI接口

时钟极性、时钟相位和位顺序：以下选项是确保成功操作所必需的。

图106：  
SPI设置

SPI - 参数	描述	设置
CPOL	时钟极性	0
CPHA	时钟相位	1
模式	SPI模式	1
DORD	位序	0, MSB优先

## SPI时序

图107:  
SPI写操作

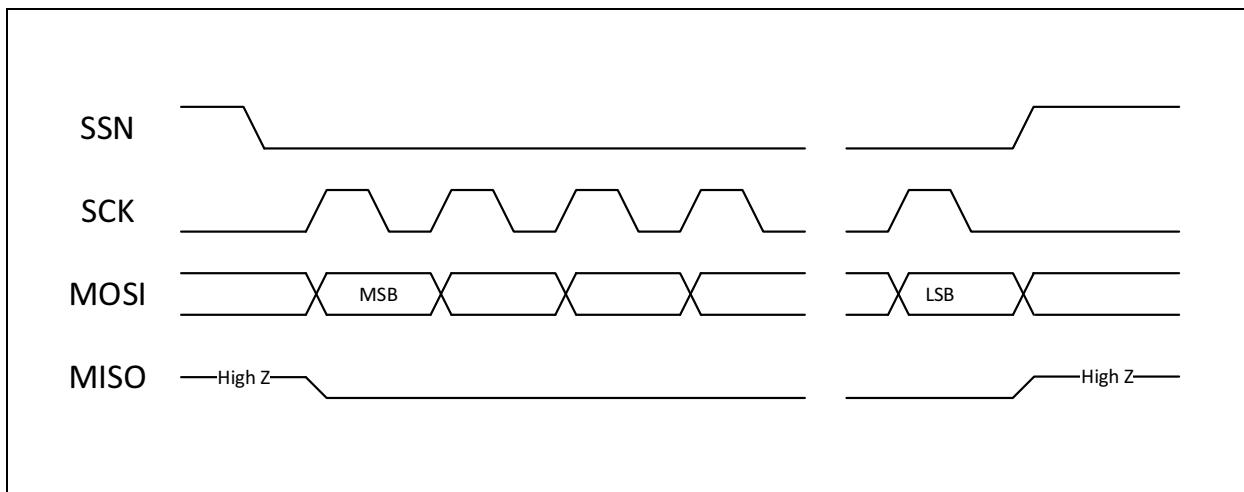


图108: SPI读  
操作

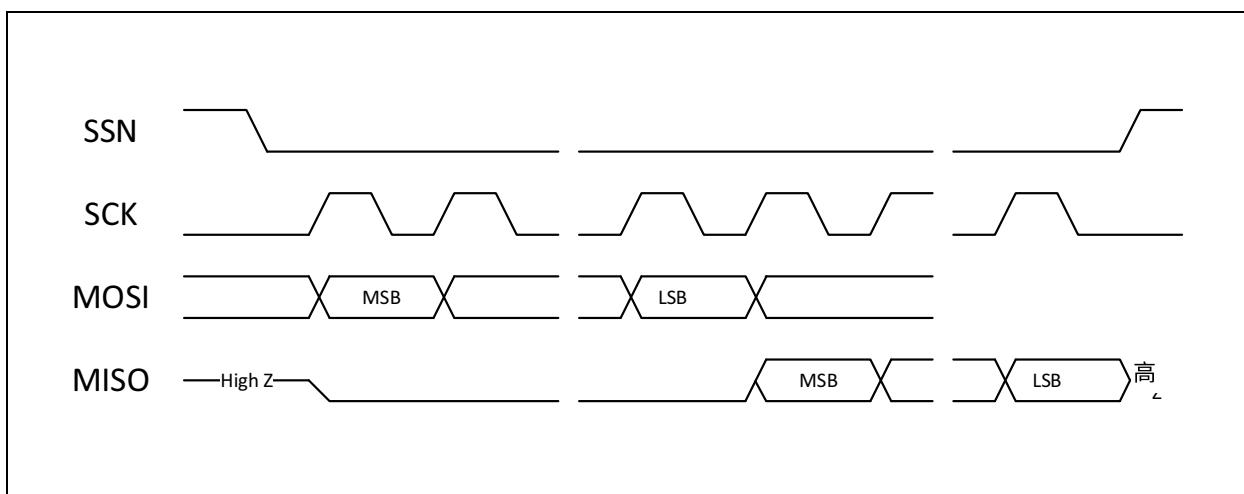


图109：  
SPI时序

名称	符号	$VDD = 2.2V$	$VDD=3.0V$	$VDD=3.6V$	单位
串行时钟频率	fSPI-bus	10	17	20	兆赫
串行时钟脉冲宽度高电平	$t_{pwh}$	50	30	25	ns
串行时钟脉冲宽度低电平	$t_{pwL}$	50	30	25	ns
SSN使能到有效锁存	$t_{sussn}$	10	8	7	ns
写入周期间的SSN脉冲宽度	$t_{pwssn}$	50	30	25	ns
时钟沿前的数据建立时间	$t_{sud}$	7	6	5	ns
时钟沿后的数据保持时间	$t_{hd}$	5	4	3	ns
时钟沿后的数据有效时间	$t_{vd}$	40	26	16	ns

### GPIO与PDM/PWM

本节介绍通用端口及其在脉冲密度调制/脉冲宽度调制输出（PDM/PWM）中的应用。PCap04在将各种GPIO引脚分配给DSP输入/输出方面具有极高的灵活性。下表显示了6个通用端口及其可能的分配方式。

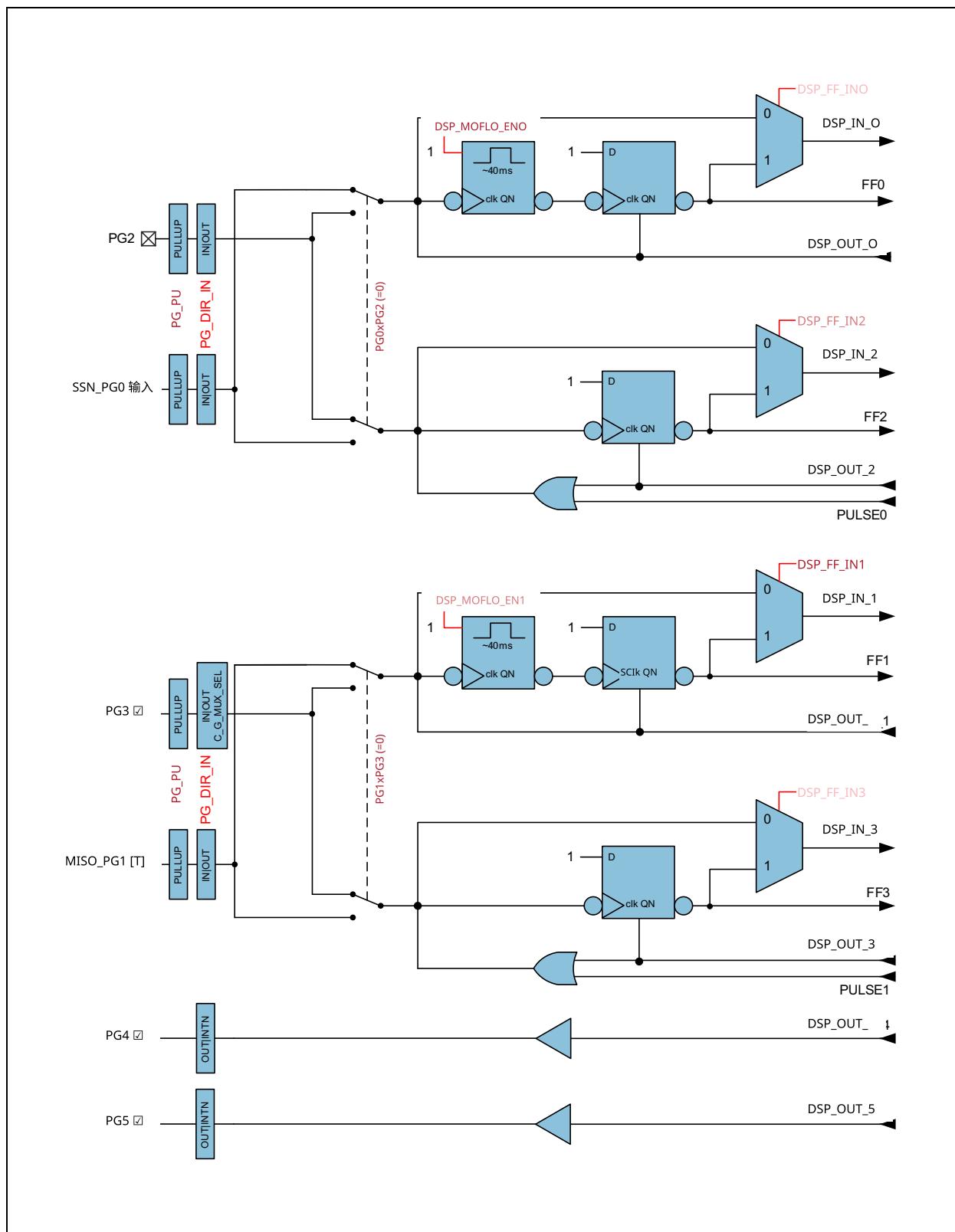
图110：  
通用端口分配

外部端口名称	描述	输入或输出方向
PG0	SSN (SPI模式下), 串行选择	In
	DSP0或DSP2, DSP的输入输出	输入 (1) / 输出
	FF0或FF2, 带触发器的DSP输入输出	In <sup>(1)</sup>
	脉冲0, PDM或PWM输出	输出
PG1	MISO (在SPI模式下)	输出
	DSP1或DSP3, DSP的输入输出	输入 (1) / 输出
	FF1或FF3, 带触发器的DSP输入输出	In (1)
	脉冲1, PDM或PWM输出	输出
PG2	DSP0或DSP2, DSP的输入/输出	输入 <sup>(1)</sup> / 输出
	FF0或FF2, 带触发器的DSP输入/输出	In (1)
	脉冲0, PDM或PWM输出	输出
PG3	DSP1或DSP3, DSP的输入输出	输出
	FF1或FF3, 带触发器的DSP输入输出	In (1)
	脉冲1, PDM或PWM输出	输出
	C_G_MUX_SEL 输出	输出
PG4	DSP4 (仅输出)	输出
	INTN	输出
PG5	DSP5 (仅输出)	输出
	INTN	输出

备注：

- 这些端口提供可选的去抖动滤波器和上拉电阻。

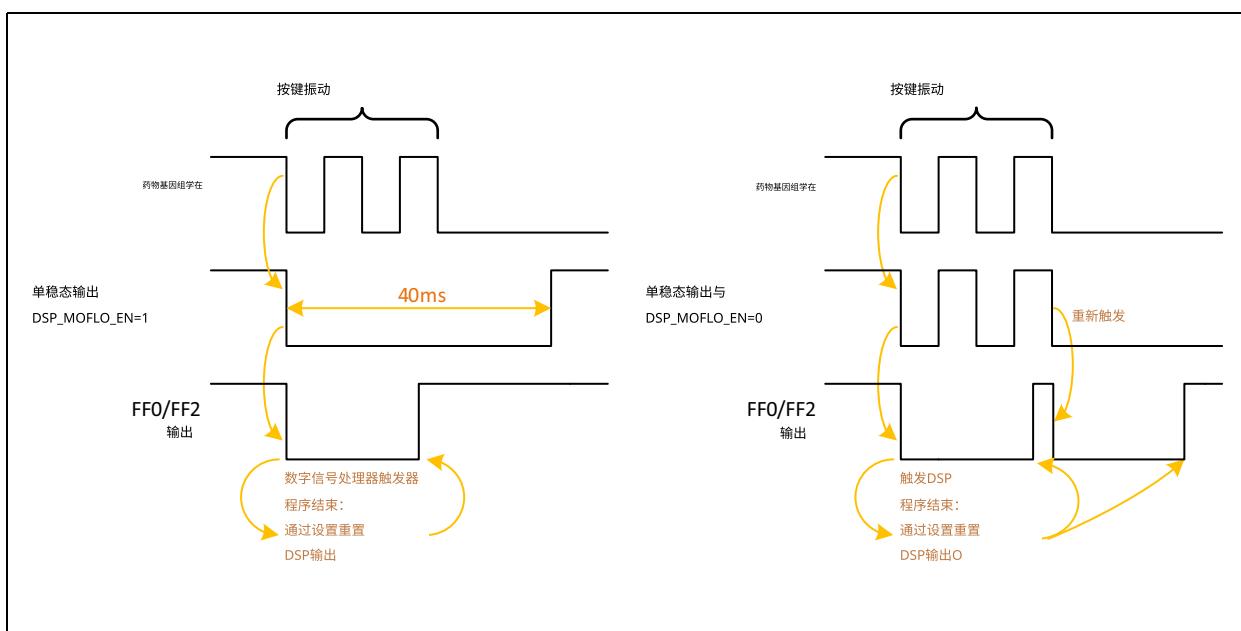
图111:  
GPIO 分配



### 去抖滤波器

在端口作为按钮输入时，有可能启用一个 40 ms 去抖滤波器（单稳态触发器）。这在 DSP 由引脚启动（信号 FF0、FF2）时尤其有用。图112显示了单稳态触发器的效果。

图112：  
端口触发时序



### PDM与PWM

存在生成两个脉冲密度调制输出信号的可能性。一般而言，优先采用PDM，因为其噪声性能更佳。输出基于 RAM 寄存器 PIO\_REF、PI1\_REF 的内容（DSP 写入地址为 109 和 110，宽度均为 16 位）。这些 RAM 单元的内容取决于固件。本数据手册中的描述基于标准固件，该固件将电容比写入 PIO\_REF，将电阻比写入 PI1\_REF。

脉冲接口可以单独开启。分辨率可编程为 10 到 16 位。可以选择多种时钟信号作为脉冲接口的基准，源自低频振荡器或内部振荡器。输出引脚可以是 PG0 或 PG2，也可以是 PG1 或 PG3。

PDM 信号可以通过简单的 RC 滤波器转换为模拟电压。由 100nF 的一级滤波器即可。用户可以通过选择电阻和电容值，优化反应时间与纹波之间的平衡。

#### 滤波器配置说明：

电阻应为  $\geq 50\text{k}\Omega$ 。输出缓冲器的内部直流电阻通常为  $100\Omega$ 。

调节时间（适用于PDM和PWM）如果输出值发生变化，达到90%的调节时间为 $2.3 \times \text{Tau}$ ,  $\text{Tau} = R \times C$ 示例:  
 $200\text{k}\Omega \times 100\text{nF} \times 2.3 = 50 \text{ ms}$

Tau越小，调节越快，但纹波越大  
 °

## 2. 电压纹波

计算方法：

$$V_{DD} \cdot \left(1 - e^{-\frac{1}{f_0 \cdot R \cdot C}}\right) \text{ with } \left(f_0 \ll \frac{1}{R \cdot C}\right)$$

$$v_{pp} = \frac{V_{DD}}{f_0 \cdot R \cdot C}$$

$v_{pp}$  = 纹波电压（峰值到峰值）

$$f_0 = \text{for PWM : } \frac{1}{\text{period}} = f_{clk}/2^{\text{PWMresolution|bit}}$$

$$\text{for PDM : } \frac{1}{t_{pulsewidth}}$$

在标准固件中，电容或温度的测量结果为一个32位值。DSP根据脉冲接口的分辨率设置对该32位结果进行线性化，参数 $\text{pi}_{<n>}_{\_result0}$ 、 $\text{pi}_{<n>}_{\_result1}$ 、 $\text{pi}_{<n>}_{\_pulse0}$ 和 $\text{pi}_{<n>}_{\_pulse1}$ 的线性函数在NVRAM的校准空间800至822中可配置。这些参数描述了简单比例（一级线性化）的边界，其中 $\text{pi}_{<n>}_{\_pulse0}$ 为最小裁剪值， $\text{pi}_{<n>}_{\_pulse1}$ 为最大裁剪值，用于脉冲输出。 $\text{pi}_{<n>}_{\_pulse0}$ 必须始终小于 $\text{pi}_{<n>}_{\_pulse1}$ 。对于负斜率，只有 $\text{pi}_{<n>}_{\_result0}$ 需要大于 $\text{pi}_{<n>}_{\_result1}$ 。12位分辨率限制结果值在0到4096之间。对于较低位分辨率，范围相应缩减。

脉冲输出的确定方式如下：

$$\text{pi}_{n\_out} = \frac{\text{pi}_{n\_pulse1} - \text{pi}_{n\_pulse0}}{\text{pi}_{n\_result1} - \text{pi}_{n\_result0}} \cdot (\text{result} - \text{pi}_{n\_result0}) + \text{pi}_{n\_pulse0}$$

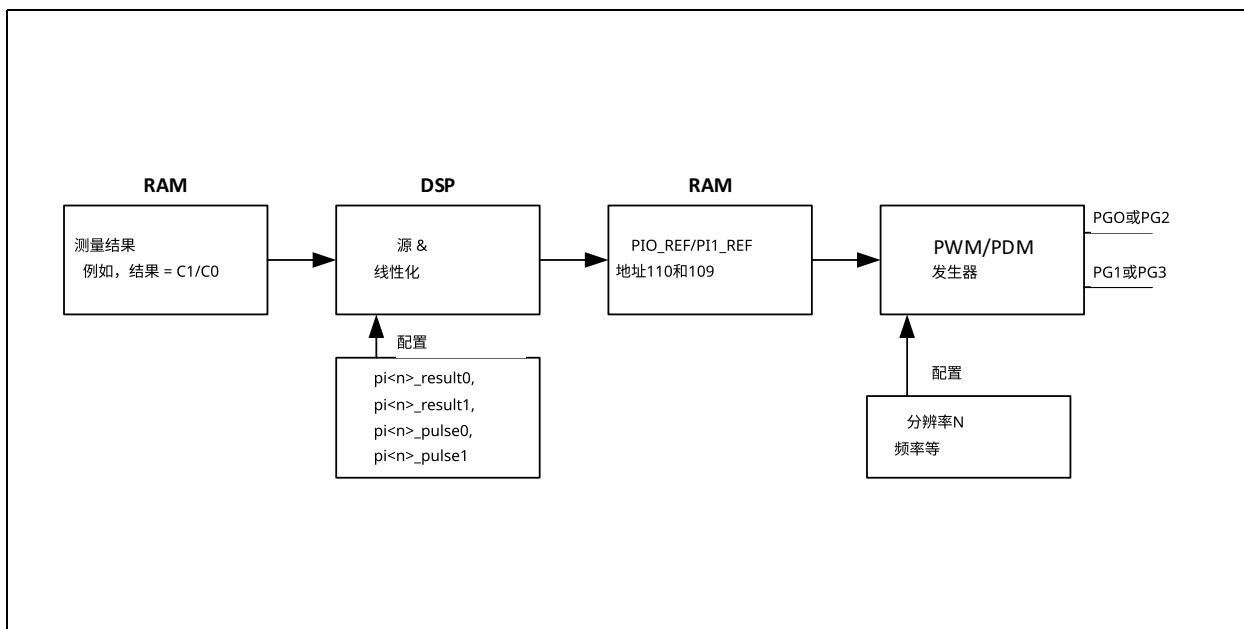
$$\text{pi}_{n\_result1} > \text{pi}_{n\_result0}$$

$$0 \leq \text{pi}_{n\_pulse1} \leq 2^{\text{pulse resolution}}$$

$$0 \leq \text{pi}_{n\_pulse0} \leq 2^{\text{pulse resolution}}$$

下图展示了如何处理结果以生成脉冲输出。

图113：  
PDM与PWM脉冲生成

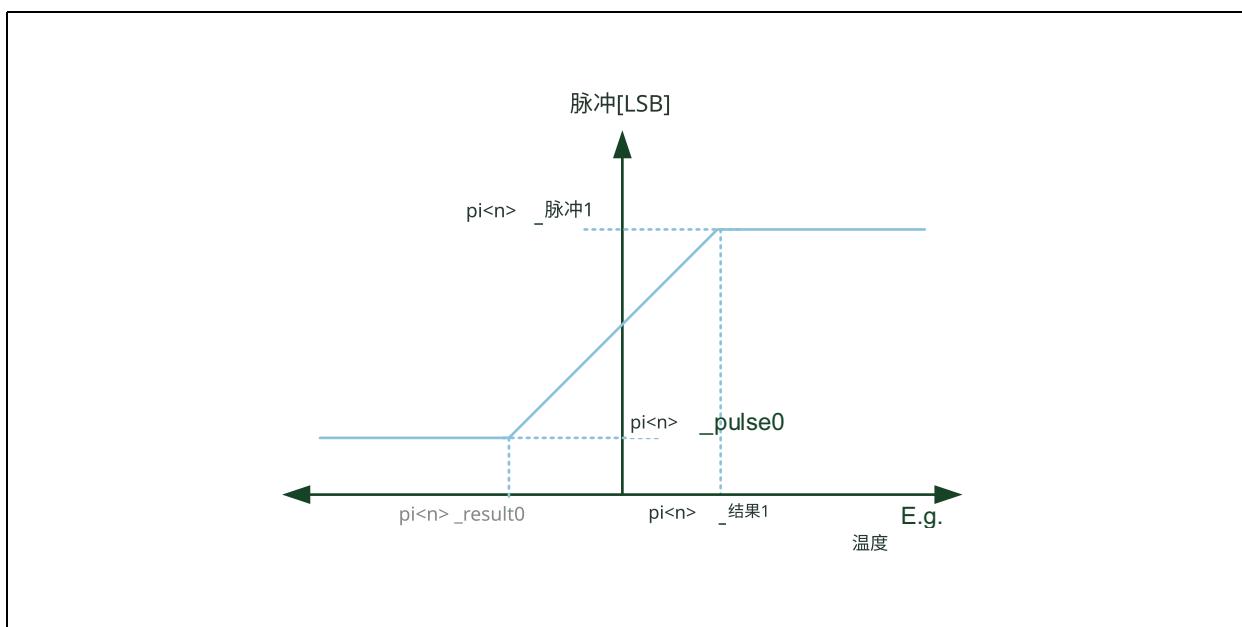


下图显示了一个线性函数及其参数的示意图。在该图中，结果C1/C0被放在x轴上，假设该结果用于脉冲调制。配置了12位分辨率。

PDM和PWM接口的设置在配置寄存器27及29到33中完成。

有效范围的下限 (pi<n>\_pulse0) 对应0%调制（所有位为0），上限 (pi<n>\_pulse1) 对应100%调制（所有位为1），即输出的最大值。12位分辨率意味着最大值为4095。较低分辨率时，该最大值相应降低。以电压为例，这两个极限对应0V和VDD。

图114:  
PDM与PWM线性化



#### 应用：

- 典型应用为通过PG0输出电容值，通过PG1输出温度值。计算和传输到输出寄存器由固件完成。
- 主要应用是最终客户需要模拟接口。
- 由于速度限制或其他原因，无法使用串行接口的应用场景。
- 最后，温度编码脉冲流可以经过低通滤波后，直接用于温度控制。

请注意，本文所述的线性化任务全部由固件完成，特别是PCap04\_standard和PCap04\_linearization固件。

**振荡器**

PCap04提供低频振荡器（OLF\_CLK）和集成的高频振荡器（OHF\_CLK）。OLF\_CLK始终运行，无法关闭。

**OLF\_CLK的用途：**

- CDC周期时间
- RDC周期时间
- PDM/PWM时间基准
- 独立应用的看门狗

**OHF\_CLK也可用于**

- CDC周期时间
- PDM/PWM时间基准

**OLF\_CLK可调节以适应多种典型频率：**

图115：  
OLF修剪

OLF CTUNE	OLF FTUNE	OLF频率
3 :(10kHz)	1	5kHz
3 :(10kHz)	7	10千赫
2 :(50千赫)	0	28千赫
2 :(50千赫)	3	48千赫
1 :(100千赫)	4	100kHz
0 :(200kHz)	5	200kHz

注意事项：内部振荡器的精度和稳定性有限。频率会因芯片、温度和电压而变化。

- 批次间变化 ±20%
- 温度变化 ±5%，  
电压变化。VDD ±2%

**OHF\_CLK可以关闭，也可以在后续测量等任务之前延迟开启，或持续开启：**

**OX\_RUN[2:0]0：发生器关闭**

6 : OX延迟 =  $1/f_{OLF}$

3 : OX延迟 =  $2/f_{OLF}$

2 : OX延迟 =  $31/f_{OLF}$

1 : OX持续运行

通过OX\_DIV4可以被4整除以产生500 kHz，

## DSP与存储器

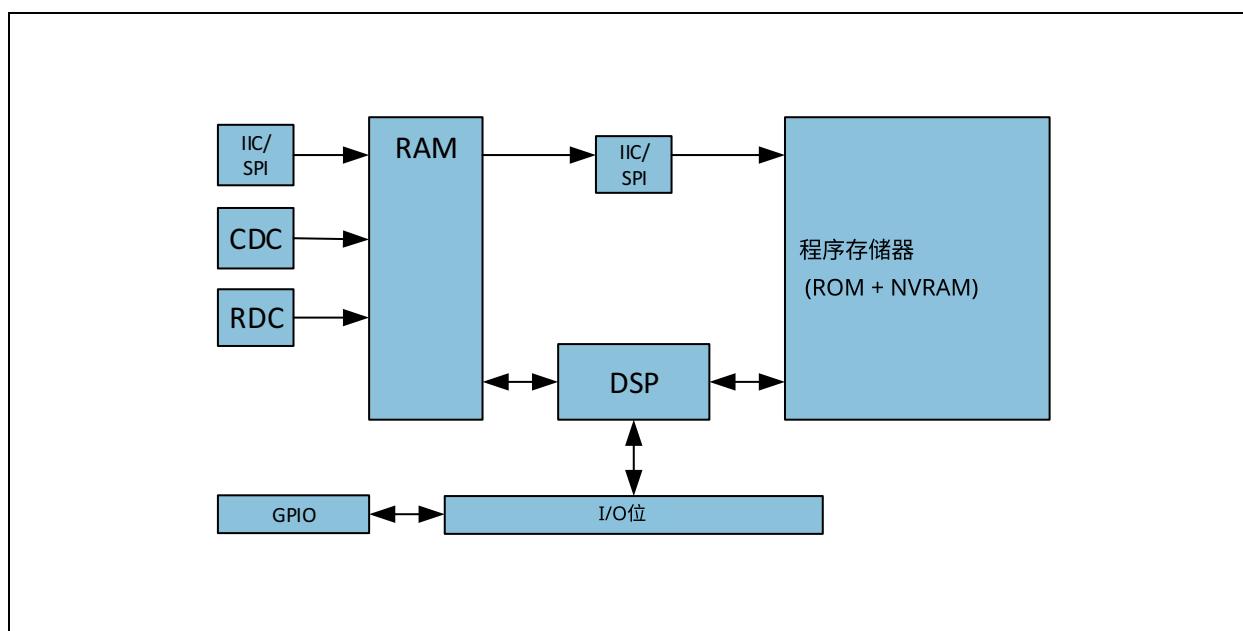
本节描述PCap04的32位数字信号处理器（DSP）。

在Harvard架构下集成的32位数字信号处理器

（DSP）负责接收CDC和RDC测量单元的数据，进行处理，并将结果提供给用户界面。CDC/RDC的原始数据以及DSP处理后的数据都存储在RAM中。DSP程序存放于非易失性存储器（NVRAM）中。DSP可以从一组 64I/O 位中采集各种状态信息，并写回其中的16位。这样，DSP可以对PCap04的GPIO引脚做出反应并进行控制。DSP的内部时钟频率约为 60MHz。通过固件命令可以停止内部时钟以节省电源。DSP在接收到GPIO信号或“测量结束”条件时重新启动。

在最简单的形式中，DSP将纯粹的时间测量信息从CDC/RDC传输到读寄存器，无需进一步处理。下一步是计算电容比，包括补偿测量中的信息，如ams标准固件版本PCap04\_standard\_v01.hex所提供的。最后，ams提供了现成的线性化固件，通过三次多项式进行线性化，并通过二次多项式进行温度补偿。许多线性化固件的功能块以ROM代码实现。这样，主固件可以非常紧凑，能够容纳在 1 k 非易失性存储器中。

图116：  
DSP嵌入



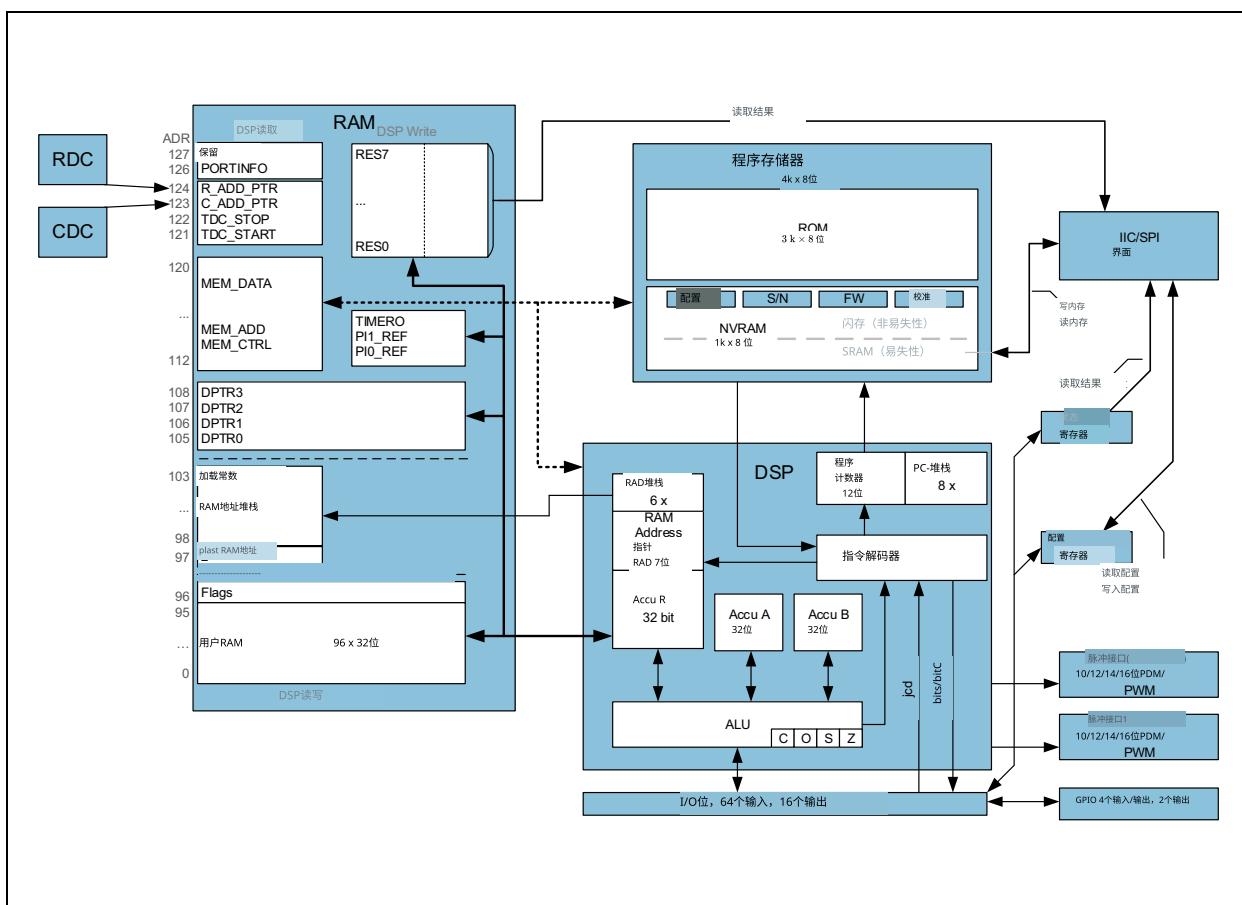
读取寄存器的内容始终取决于所使用的固件。使用标准固件时，内容为纯电容和电阻比值。使用线性化固件时，可能为线性化和校准后的结果，例如以帕斯卡为单位的压力或以百分比表示的湿度。

DSP是ams专有技术，旨在处理低功耗任务以及高速数据传输。它采用汇编语言编程。配备图形界面的用户友好型汇编软件，带有帮助提示弹窗和示例代码，支持编程工作。

### DSP与环境

DSP从RAM中读取RDC和CDC的原始数据，进行处理后将结果写回RAM。程序存储在NVRAM中，可能调用ROM中的子程序。DSP根据标志和控制标志作出反应，控制GPIO接口以及PDM/PWM接口。

图117：  
DSP环境



该DSP采用哈佛架构，支持32位宽的并行数据处理。它连接到一个  $128 \times 32$  位的RAM，容量为96x32位，均可自由访问。在读取时，DSP可以从地址空间112到120获取MEM\_DATA，从地址空间121到124获取CDC和RDC前端控制。

通过写访问，DSP将输出数据提供给PDM/PWM接口（地址109、110）。

关于RAM的详细描述见第2.1节。DSP使用两个累加器A和B，并且可以直接访问RAM，RAM可以看作第三个累加器。RAM地址指针为7位，并有一个6层堆栈用于存储RAM地址。

程序计数器为12位，并有一个8层堆栈用于存放程序计数器值。

最后，DSP可以从64个I/O位中获取大量信息。读取信息包括ALU状态、触发信息、部分配置位以及GPIO状态信息。其中16位可用作输出，用于设置GPIO和一些内部信息。DSP可以通过jcd（条件跳转）指令读取这些位，并通过bitS/bitC（置位/清零）指令设置这些位。

ALU标志位中的溢出、进位、相等/不相等以及正/负，直接用作jcd指令的条件，并在I/O位中反映出来。

#### RAM结构

RAM起着关键作用。它由128个字组成，最大字长为32位。DSP可以对寄存器地址0到96的所有32位宽的字进行自由读写。地址97到104、109到111以及115及以上的RAM空间在读写操作中具有不同的含义。

读取区的主要数据是来自CDC和RDC的原始数据。此外，参数存储在RAM中，作为配置寄存器的一部分，通过串行接口设置或从NVRAM复制。

DSP读取原始数据，进行数据处理，并将结果写回RAM的写入区。用户可以通过串行接口读取最终结果。

部分RAM单元专用于特殊功能，后续将详细介绍。

图118：  
RAM结构详解

RAM: DSP读取			RAM: DSP写入		
地址	描述	位	地址	描述	位
127	保留				-
126	PORTINFO	24			
124	R_ADD_PTR	2			
123	添加指针	4			
122	停止TDC	27			
121	启动TDC	27			
120	存储数据_u08b	32			
119	存储数据_u16b	32			
118	存储数据_u24b	32			
117	MEM_DATA_s08b	32			
116	MEM_DATA_s16b	32			
115	MEM_DATA_s24b	32			
114	MEM_DATA (写入&4字节读取)				32
113	MEM_ADD				10
112	MEM_CTRL				16
			111	TIMERO	16
			110	PI1_REF	16
			109	PIO_REF	16
108	DPTR3				7
107	DPTR2				7
106	DPTR1				7
105	DPTRO				7
			104	RES7	32
103	辐射堆叠_6b	6	103	RES6	32
102	辐射堆叠_12b	12	102	RES5	32
101	辐射堆叠_18b	18	101	RES4	32
100	辐射堆叠_24b	24	100	RES3	32

RAM: DSP读取			RAM: DSP写入			
地址	描述	位	地址	描述	位	
99	rad_stack_30b	30	99 98 97	RES2	32	
98	rad_stack_32b	32		RES1	32	
97	最后两个RAM地址	32		RES0	32	
96	标志与扩展GPIO				32	
95	(空闲) 用户内存				32	
...	...				32	
0	(空闲) 用户内存				32	

寄存器0到95， 用户内存

这是普通内存空间，没有特殊功能。可以通过

指令rad进行读写。

示例：

将内存地址12和13的内容相加，并将结果写入  
地址13 rad 12 move a, r rad 13 add r, a

寄存器96，标志位与内部控制信号

图119：标  
志

位	标志名	默认值 (复位后)	描述
0	FIRSTART_N	0	表示第一个DSP触发，直到固件设置为1
1..2	可自由使用	0	
3	RDCHG_COM_INT_SEL	0	0: 使用 RDCHG_IN_SEL; 1: 使用 RDCHG_IN_SEL1; 用于内部补偿
4	免费使用	0	
	FLAG_CDC_INV	0	临时参数，用于ROM例程 _ROM_cdc____0 : 确定反比参考/感应1 : 确定比率感应/参 考
5	SIGNED_VALUE_NV	0	临时参数，用于ROM例程 _ROM_NVblock_copy_32b_ROM_N Vblock_copy_24b_ROM_NVblock_c opy_16b_ROM_NVblock_copy_08b _0 : 将NVRAM中的数据视为无符号1 : 将NVRAM中的数据视为有符号
6..7	保留	0	临时在ROM例程中使用的标志
8	RST_RDC	脉冲	温度复位。每次RDC测量后必须将此标志设置为1，否则无法进行新的RDC测量。该标志会自动恢复为0
9..15	保留		
16..31	可自由使用	未知	

DSP读取寄存器97

该寄存器用于获取2的第N次方。指数 N 需要写入RAD堆栈。结果可以从寄存器81读取。在汇编中，所需的三个

指令合并为一条：

load2exp a, 10; a =  $2^{10} = 1024$

设置累加器 = 1 的非常简单且高效的方法是

load2exp b,

$0 ; b = 2^0 = 1$  DSP寄存器读取98到103

这些寄存器包含RAM地址堆栈的内容。32位数据由最后6个6位RAM地址组成。该地址可用于将程序存储器中的32位常数加载到数据空间。必要的指令由汇编器合并为一条指令。(提示：汇编器接受负值、十进制和十六进制数。根据要加载的常数，汇编器会将此指令转换为3到8个操作)

load a, 1715956 ; a = 1715956

与

rad 0x06	; $0x06 * 2^{18}$
rad 0x22	; $+0x22 * 2^{12}$
弧度 0x3b	; $+0x3b * 2^6$
弧度 0x34	; $+ 0x34 = 1715956$
弧度 100	; rad_stack_24b
移动 a, r	

DSP 读写寄存器105到108，数据指针  
这些寄存器可用于间接寻址。它们宽7位。

将寄存器加载为你要操作的地址：

加载 a, <myaddress>
辐射DPTRO
移动 r, a

加载 a, <myaddress>
辐射105
移动 r ,a

用DPTRO的内容加载到RAM地址指针：

rad\_at\_DPTRO；现在RAM地址指针已设置为DPTRO的内容

提示：在<pcap\_standard.h>中，“\_at\_DPTRO”到“\_at\_DPTR3”被设置为284到287的值。这些不是有效的RAM地址，而是指示汇编器生成相应操作码的标记。

示例直接内存地址：将一块内存从一个地址复制到另一个地址：

____sub_dma_:		
not b	; 初始化循环计数器	; DPTR1 : 源地址; DPTRO : 目标地址; b: DMA长度
英寸	; 与 - <length>	
_sub_dma_loop_:	;	
rad_at_DPTR1	; 复制 a : @DPTR1	
move a, r	;	
rad_at_DPTRO	; 复制 @DPTRO : a	
移动 r, a		
rad DPTRO	; 增加目标地址	
自增	; 地址	
rad DPTR1	; 增加源地址	
incr	;	
inc b	; 循环递增	
jNE _sub_dma_loop_jrt	; 计数器	
	; 循环体	

#### DSP 读取寄存器126, PORTINFO

(PORTERR<7...0>, PORTMASK<7...0>) 低8位反映端口使能设置，定义在寄存器12的配置参数C\_PORT\_EN中。第8到17位为电容端口的错误标志，包括内部参考端口。DSP 写入寄存器97到104, RES00...RES07, 这些是DSP必须写入输出数据的结果寄存器，用户可以通过SPI/IIC接口读取，作为Res 0到Res 7。

所有地址均为32位宽。

！！！注意：这些寄存器仅支持写入！DSP无法读取这些寄存器！！！

#### DSP写入寄存器109、110、PIO\_REF...PI1\_REF

这些寄存器包含用于生成PWM/PDM输出信号的数据。DSP在计算并缩放输出数据后，将其写入这两个寄存器。数据宽度为16位。

#### DSP写入寄存器111, TIMERO

DSP具有基于OLF时钟的16位定时器。该定时器可用于在DSP暂停时产生长延迟。必须设置DSP\_START\_EN中的第3位（定时器）！

通过向寄存器111写入值，定时器从0开始每个OLF时钟周期递增，直到达到写入的值，然后生成

#### DSP\_START\_TRIG。

如果DSP未暂停，仍然可以测试TIMERO\_IRQ\_N标志位。

**示例1 (不暂停DSP)：**

```
CONST wait_time_1ms50 ;50*20μs (@50kHz)
```

...

```
load a, wait_time_1ms
rad TIMERO
move r, a
```

timer\_wait\_loop:

jcd TIMERO\_IRQ\_N, 定时器等待循环

**示例2 (带暂停DSP, DSP在内部振荡器上运行)：**

```
CONST wait_time_1ms50; 50*20μs (@50kHz)
```

...

ORG 0

jcd TIMERO\_IRQ\_N, 跳过\_Timer0\_处理

jsb 由\_Timer0\_触发

跳过\_Timer0\_处理:

加载 a, 等待时间

辐射定时器

移动 r, a

停止

由定时器0触发:

DSP 读写寄存器112到120, 存储控制, 存储地址, 存储数据

这些寄存器由只读存储器例程用于在NVRAM和RAM之间传输数据。可以在NVRAM与累加器a和b之间传输数据, 传输长度为1到4字节, 支持有符号和无符号数据。

- MEM\_CTRL: 定义操作方式。四个选项为
  - MEM\_STORE: 写入NVRAM
  - MEM\_RECALL: 从NVRAM读取
  - MEM\_WE: 启用写入
  - MEM\_WR\_PROTECT: 防止任意写入
- MEM\_ADD: 定义NVRAM中的目标地址
- MEM\_DATA\_xxx: 用于写入或读取数据的寄存器。需要写入到NVRAM的数据存放在RAM地址114。已从NVRAM读取的数据存放在地址114到120之间, 具体取决于格式。

这些调用用于将常数和校准数据从NVRAM复制到RAM，例如。

示例：将NV\_C\_sens\_sel寄存器复制到RAM

```
load a, NV_C_sens_sel ; 内存地址: NV_C_sens_sel  
rad mem_add
```

```
move r, a
```

```
jsb _ROM_memory_rd_a_u08b_ ; A包含
```

NV\_C\_sens\_sel的内容

```
rad RAM_C_sens_sel
```

移动 ring

这些寄存器可由DSP动态调整配置。重要提示：写入NVRAM后，必须执行初始化重置。因此，DSP需要按照以下顺序设置标志DSP\_6和DSP\_7。

; 初始化重置

```
bitC 7
```

```
bitC 6
```

```
bits 6
```

```
bitC 6
```

```
bitS 6
```

DSP 读寄存器 121 至 124, TDC\_START,

TDC\_STOP, C\_ADD\_PTR, R\_ADD\_PTR

ams内部数据，由ROM例程\_\_\_\_\_tdc\_dispatch\_\_\_\_\_使用

### NVRAM与ROM

总程序存储器由 1 k NVRAM和 3 k ROM组成。NVRAM存储配置信息、960字节的用户代码及一些特殊寄存器。ROM包含实用的数学运算例程，提升编程效率。

### NVRAM结构

用户空间分为三个部分。原因是NVRAM可以按段进行读写保护。一个704字节的大段用于程序代码，两个128字节的小段可用于校准数据或额外固件。

图120：  
NVRAM组织结构

地址		非易失性存储器 (1k x 8位)		存储锁定
十进制	十六进制	内容	长度 [字节]	设置
1023 to 1022	3FF 到 3FE	充电泵	2	存储锁<3>
1021 to 1011	3FD 到 3F3	保留	11	
1010 to 1009	3F2到3F1	S/N客户	2	
1008	3F0	MEM_LOCK	1	
1007 to 960	3EF到3C0	配置注册表	48	MEM_LOCK<3>
959 to 832	3BF到340	用户空间 (固件/校准1)	128	MEM_LOCK<2>
831 to 704	33F到2C0	用户空间 (固件/校准0)	128	MEM_LOCK<1>
703 to 0	2BF到0	用户空间 (固件)	704	MEM_LOCK<0>

非易失性存储器 (NVRAM) 由两部分组成：易失性SRAM和非易失性存储器 (FLASH)。存在一种存储/回忆方法，可以将完整的SRAM内容存储到FLASH中，或从FLASH中回忆到SRAM中。

不同的操作方法适用：

•**独立模式：**

配置数据、固件和校准值存储在非易失性存储器中一次性写入，并选择自动启动。通电后，设备立即开始测量。

•**预配置模式：**

配置数据、固件和校准值存储在非易失性存储器中一次性写入，禁用RUNBIT和自动启动。通电后，设备已被编程和配置，但处于空闲状态，等待指令。

•**纯从属模式：**

配置数据、固件和校准值在每次通电后由外部  $\mu$ C 写入到SRAM (易失性存储器)。

### NVRAM访问

有三条命令用于操作NVRAM：存储、读取和擦除，每个命令在串行通信中都受到保护，以防止意外触发。必须先发送激活码以注册54（MEM\_CTRL），然后再发送相应的操作码（||=以结束SIF，例如设置SSN为高电平：

将SRAM内容存储到NVRAM：激活码在MEM\_CTRL中：  
0x2D

存储NVRAM操作码：0x96

（通过SIF发送：0xA3F62D ||  
0x96，等待至少12毫秒）

从NVRAM恢复到SRAM：在MEM\_CTRL中的激活：0x59从  
NVRAM获取指令码：0x99（通过SIF发送：0xA3F659 ||  
0x99）

擦除NVRAM：                   读取校准比特地址1022&1023  
                                    和唯一ID从地址954到959在  
                                    MEM\_CTRL中的激活：0xB8擦  
                                    除NVRAM指令码：0x9C（通  
                                    过SIF发送：0xA3F6B8 ||  
                                    0x9C，等待至少12毫秒）将校  
                                    准比特地址1022&1023和唯一  
                                    ID写回到地址954到959在  
                                    MEM\_CTRL中的激活码：0x2D  
                                    存储NVRAM指令码：0x96

**重要提示：**我们保证数据的保持和耐久性仅在客户不更改寄存器62和63以及NVRAM地址954到959（唯一ID）的前提下有效。此外，必须严格按照第NVRAM和ROM章节中描述的程序执行ERASE NVRAM操作。否则，我们将不再保证数据保持时间和耐久周期。

### ROM结构

NVRAM的容量限制通过在3k ROM中集成的多功能硬连线实现补偿。ROM例程涵盖从简单的移位功能到滤波器，再到多项式线性化4<sup>th</sup>的各种复杂操作。这使得用户代码非常紧凑。

### 汇编器配备有头文件

PCap04\_ROM\_addresses\_standard.h，列出了各个ROM例程的跳转地址。详情请参见“示例代码/库”部分。

### DSP输入与输出

DSP可以访问关于ALU状态、启动触发、配置、输入/输出引脚的64位信息。

此信息可以通过指令jcd或条件跳转进行解释。

条件跳转指令如下：

jcd  $p1, p2$ ：如果  $p1 == 1$ ，则跳转到  $p2$ ,  $p1$  = 标志编号  
DSP可以设置其中的16位，例如用于设置GPIO或在RDC  
与CDC数据之间切换。这些位由指令bitS / bitC（置位/清  
零）控制。

图121：  
DSP输入/输出

位名称	描述	类型	读位#	写位#
DSP_OUT<7...0>	8个通用DSP输出的状态反馈（写入位0到7）。	IN	56 to63	
SIF_TRIGGERED_N (1)	标志 = 低电平表示引脚出现下降沿或SPI/IIC指令已启动DSP。该标志由固件中的STOP指令在结束时复位。	启动触发	55	
PIN_TRIGGERED_N (1)	标志 = LOW 表示GPIO已启动DSP		54	
TDC_TRIGGERED_R_N (1)	标志 = LOW 表示电阻（温度）测量的单次时间值已可用，需由ROM例程 _ROM_tdc_dispatch_ 处理			
TDC_TRIGGERED_C_N (1)	标志 = LOW 表示电容测量的时间值已可用，需由ROM例程 _ROM_tdc_dispatch_ 处理	开始触发	52	
INTN_TRIGGERED_N (1)	标志 = 低表示DSP由INTN信号的上升沿启动	启动触发	51	
TIMER0_IRQ_N (1)	标志 = LOW 表示DSP由内部定时器启动	启动触发	50	
RDC_TRIGGERED_N (1)	标志 = LOW 表示RDC测量已启动DSP，因此必须设置DSPSTARTONTEMP（配置寄存器8）。该标志在固件结束时由STOP指令复位。	启动触发	49	
CDC_TRIGGERED_N (1)	表示DSP由电容转换结束触发启动	启动触发	48	

位名称	描述	类型	读位 编号	写位编 号
ALU_OFL_N	ALU溢出、进位、相等和符号标志。ALU标志由汇编器的跳转指令使用	状态	47	
ALU_OFL		状态	46	
ALU_CAR_N		状态	45	
算术逻辑单元_车辆		状态	44	
算术逻辑单元等于 / 零		状态	43	
算术逻辑单元不等 / 非零		状态	42	
算术逻辑单元正		状态	41	
算术逻辑单元负		状态	40	
标志寄存器_N[7:0]	低8位，来自FLAGREG（寄存器96）的反转标志	标志	32..39	
唤醒触发_N	将RUNBIT设置为1后，DSP立即被触发。此标志显示触发源，用于在首次测量前初始化原始结果寄存器（由ROM例程_ROM_tdc_dispatch_使用）	开始 触发	31	
TDC就绪	标志 = 低表示TDC-Ring振荡器正在运行	状态	28	
POR_CDC_DSP_COLL	标志 = 低表示由CDC / DSP碰撞引发的复位	状态	27	
LAST_CYCLE_ACTIVE_N	标志 = 低 表示这是当前序列中的最后一次CDC测量（用于ROM例程_ROM_tdc_dispatch_）	状态	26	
CYC_ACTIVE	状态寄存器的标志 = 位23。表示CDC前端处于激活状态（非取反）	状态	25	
POR_FLAG_WD	标志 = 低表示由看门狗超时强制重置	状态	24	
POR_FLAG_PARITY	标志 = 低表示由干扰引起的一个或多个配置位切换强制重置	状态	23	
连续_N	低：连续模式已激活	配置寄 存器	22	
自动启动_N	来自配置寄存器的位	配置寄 存器	21	
内部参考电压	配置寄存器的位	配置寄 存器	20	

位名称	描述	类型	读位编号	写位编号
TIMER_TRIG_DSP			19	
(TRUE)	常数1，可用于“跳转” jcd TRUE, <jump_address>		18	
INT_TRIG_BG_N	配置寄存器中的位	Config Reg	17	
CDC_TRIG_BG_N	配置寄存器中的位	配置寄存器	16	
C_COMP_EXT_N	来自配置寄存器的位	配置寄存器	15	
C_COMP_IN_N	来自配置寄存器的位	配置寄存器	14	
C_SINGLE / C_DIFFERENTIAL_N	配置寄存器中的位	配置寄存器	13	
C接地 /C浮空_N	配置寄存器中的位	Config Reg	12	
ERR_OVFLN	状态寄存器的 = 位16。表示TDC中的溢出或其他错误。	状态	11	
COMB_ERRN	状态寄存器的标志位 = 第16位。这是所有已知错误条件的组合状态。	状态	10	
CYC_ACTIVE_N	状态寄存器的标志位 = 第23位。表示CDC前端处于激活状态。(取反)	状态	9	
SIF_RES_RD_BSY		状态	8	
内存繁忙	表示NVRAM正忙	状态	7	
中断输入	端口INTN将在SSN (SPI) 上的正边沿或停止条件 ( $I^2C$ ) 时被复位，通过此标志可以检测INTN的当前状态	状态	6	
TEMPERR_N	标志 位0 状态寄存器1的第3位。指示温度测量过程中是否发生错误。0：错误，1：无错误	状态	5	
RDC_BUSY	标志 位0 状态寄存器的第2位。指示RDC单元是否繁忙。0：测量完成，1：测量进行中。	状态	4	
TRIG_BG	该参数启动带隙（与测量同步）（脉冲，自动设为0）	输出		15
(MEM_PUSH)	保留，仅ROM程序使用	退出		14

位名称	描述	类型	读位编号	写位编号
RST_CDC	CDC重置。此标志在每次CDC测量后必须设置为1，否则无法进行新的CDC测量。该标志会自动恢复为0	输出		13
(存储器读取)	保留，仅供ROM例程使用	输出		12
中断输出	设置中断（引脚PG4或PG5，见寄存器30）（脉冲，自动清零）	输出		11
(PAGE)	保留，禁止使用	输出		10
TRIG_RDC	此位启动新的RDC测量。（脉冲，自动清零）	输出		9
TRIG_CDC	此位启动新的CDC测量（脉冲式，自动设为0）	输出		8
DSP_7	这两个输出由DSP用于- 复位看门狗- DSP初始化复位（两个输出的模式组合用于防止意外触发这些操作）； 初始化复位	退出		7
DSP_6	bitC 7bitC 6bits 6bitC 6bitS 6	退出		6
DSP_5	设置通用引脚PG5的输出	输出		5
DSP_4	设置通用输出引脚PG4	输出		4
DSP_3	当Pulse1关闭时，此位可用于设置和清除通用输出引脚PG3。当Pulse1开启时，此位必须清除，以使Pulse1输出显示在PG3上。	输入/输出	3	3
DSP_2	当Pulse0关闭时，此位可用于设置和清除通用输出引脚PG2。当Pulse0开启时，必须清除此位，以使Pulse0输出显示在PG2上	输入/输出	2	2
DSP_1	设置或读取通用引脚 1/Osat 和 PG0 & PG1。引脚分配可编程，详细信息如下所示。	输入/输出	1	1
DSP_0		输入/输出	0	0

**注意事项：**

1. 这些输入的负边沿触发启动DSP。启动触发的状态会被存储，直到下一次复位或停止DSP。启动触发信息可通过jcd从输入48到55读取。

## 算术逻辑单元标志

每个ALU操作都会设置标志。ALU具有四个标志：溢出、进位、相等和符号。下表提供了概览：

图122：ALU  
标志

标志	描述	格式	修改说明：	解释说明：	范围
ON	无溢出	有符号	加、减、乘、除	jOvIC, jOvIS	$\geq -2^{31}$ 和 $\leq 2^{31} - 1$
O	溢出				$< -2^{31}$ 和 $> 2^{31} - 1$
CN	无进位 (1)	无符号	加法、减法、乘法、除法	jCarC, jCarS	$< 2^{32}$
C	进位 (1)				$\geq 2$
Z	等于/零	有符号/ 无符号	加法、减法、乘法、除法、移动、左移、右移	jEQ, jNE	$= 0$
ZN	不等于 / 不零				$\neq 0$
S	正数	带符号	加法、减法、乘法、除法、移动、左移、右移	jPos, jNeg	$\geq 0$
SN	负				$< 0$

## 注意事项：

- 在相加过程中，当最高有效位发生进位时，进位 C 被置位，否则保持在0。  
在相减过程中，默认进位 C 为1。只有当被减数小于减数时，进位 C 才会清零。

例如，对于  $A - B$ ：如果  $A \geq B \rightarrow C = 1$ ；如果  $A < B \rightarrow C = 0$ 。

换句话说，进位 C 实际上是加法运算  $A + 2^{\text{位数}} - B$  中的进位状态。

## DSPOUT – GPIO分配

PCap04在GPIO引脚分配到DSP输入/输出方面非常灵活。下表显示了可能的组合。

图123：  
引脚分配

外部端口	描述	输入/输出
PG0	SSN (SPI模式下)	In
	DSP_X_0或DSP_X_2	In <sup>(1)</sup> / 输出
	FF0或FF2	In <sup>(1)</sup>
	脉冲0	输出
PG1	MISO (SPI模式)	输出
	DSP_X_1或DSP_X_3	输入/输出
	FF1或FF3	In <sup>(1)</sup>
	脉冲1	输出
PG2	DSP_X_0或DSP_X_2	输出
	FF0或FF2	In <sup>(1)</sup>
	脉冲0	输出
PG3	DSP_X_1 或 DSP_X_3	In <sup>(1)</sup> / 输出
	FF1 或 FF3	In(1)
	脉冲1	输出
	C_G_MUX_SEL	输出
PG4	DSP_OUT_4 (仅输出)	输出
	INTN	输出
PG5	DSP_OUT_5 (仅输出)	输出
	INTN	输出

## 注意事项：

- 这些端口提供可选的去抖动滤波器和上拉电阻。

图124:  
GPIO分配

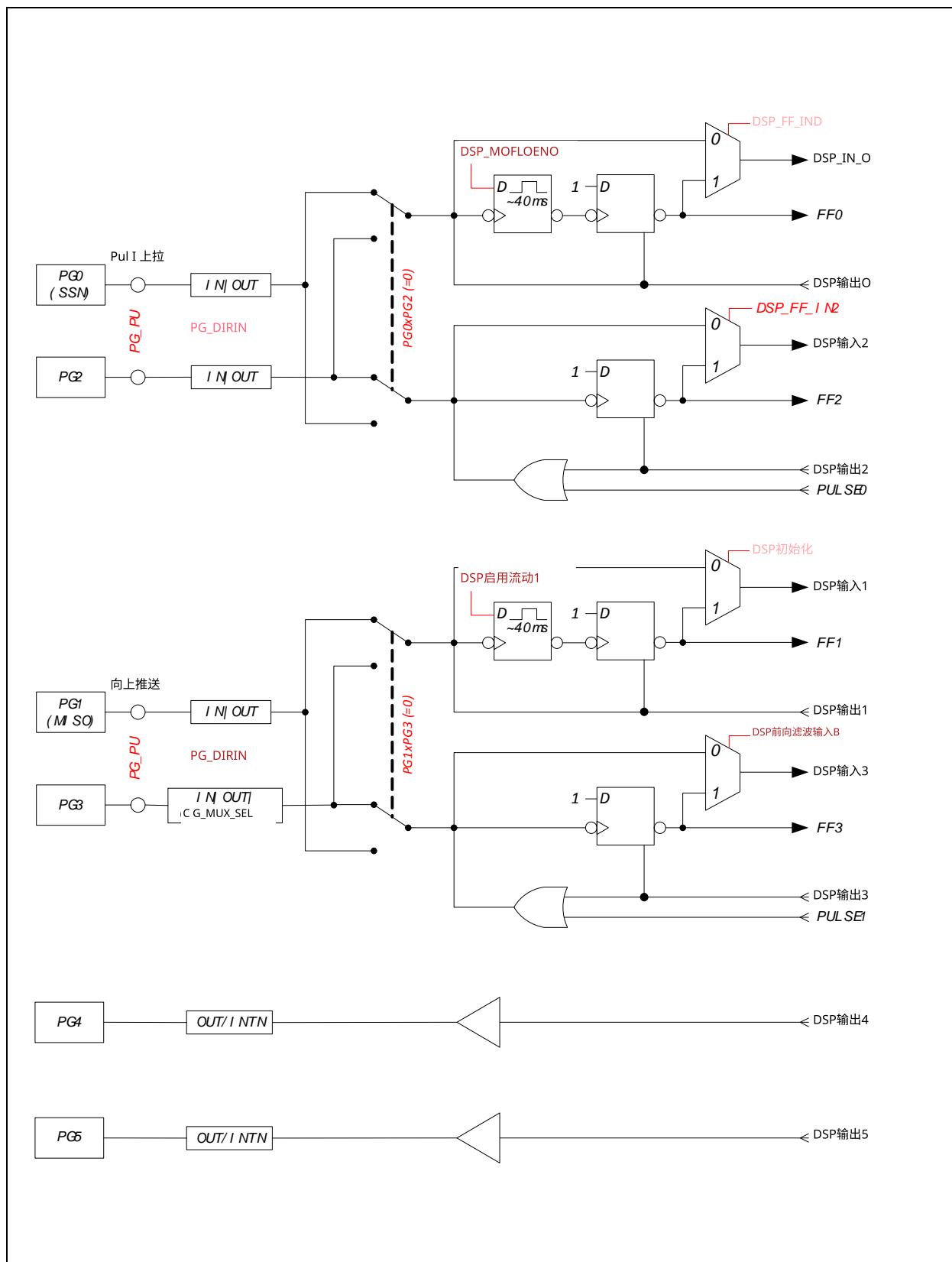
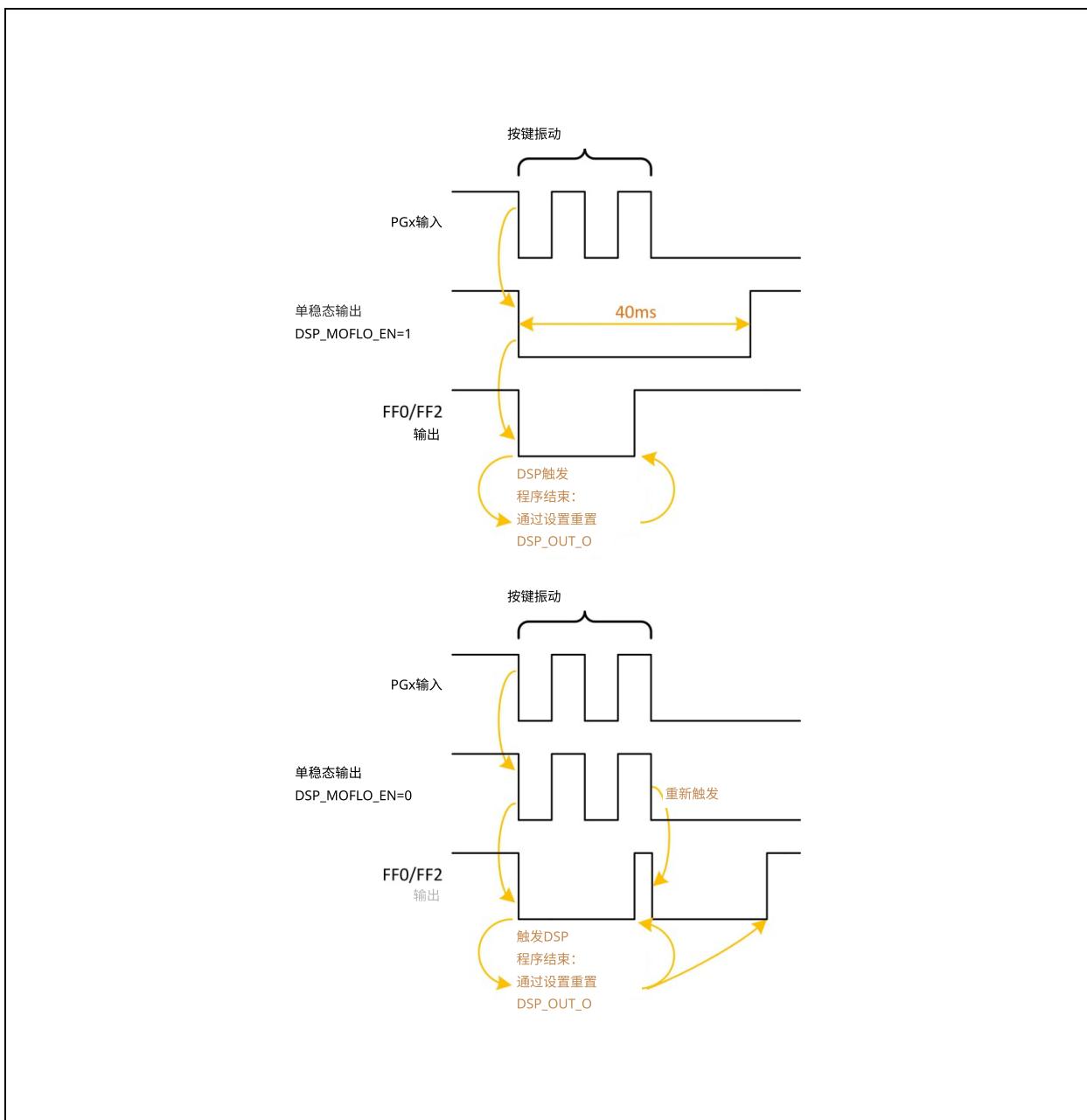


图125：  
端口触发时序



当端口用作输入时，有可能激活一个 40 ms 去抖滤波器（“单稳态”）。这在DSP由引脚（信号FF0、FF2）启动时尤为有用。图125显示了单稳态滤波器的效果。

此设置在以下配置寄存器中完成：

图126：  
脉冲输出端口

寄存器	参数	设置	描述
27	DSP_MOFLO_EN	第6位用于PG, 第7位用于PG1	在PG0和PG1线上激活防抖滤波器
27	PG0xPG2	0: PG01; 1: PG2	脉冲码可以在端口PG0 & PG1或PG2 & PG3输出。在 I <sup>2</sup> C 模式下，它们也可以选择在PG2和PG3上输出，而不是PG0和PG1。
27	PG1xPG3	0 : PG1 1 : PG3	
29	DSP_FF_IN	位 0 : PG0 位 1 : PG1 位 2 : PG2 位 3 : PG3	锁存触发器激活的引脚掩码
30	PG4_INTN_EN	第6位	在端口PG4激活INTN
30	PG5_INTN_EN	第7位	在端口PG5激活INTN
33	PG_DIR_IN	0 : 输出 1 : 输入	切换输出到输入 (PG3/bit7到PG0/bit4)。
33	PG_PU	位0: PG0 位1: PG1 位2: PG2 位3: PG3	在PG0至PG3线上激活上拉电阻；适用于机械开关。

**DSP配置**

配置寄存器8定义了DSP的操作。相关位为：

DSP\_SRAM\_SEL, DSP\_START,  
 DSP\_STARTONOVL, DSP\_STARTONTEMP,  
 DSP\_STARTPIN,  
 DSP\_WATCHDOG\_LENGTH, DSP\_SPEED

图127：  
DSP配置

调节	参数	设置	描述
27	DSP_SPEED	0: 最快 1: 快 2 : 推荐3 : 低电流 (慢)	设置DSP速度
29	DSP_STARTONPIN	0 : FF01 : FF12 : FF23 : FF3	DSP触发引脚掩码
30	DSP_START_EN<2..0>		DSP触发使能 'bxxx1 : 通过CDC结束触发'bxx1x : 通过RDC结束触发 (推荐) 'bx1xx : 通过定时器触发'b1xxx : 已废弃
34	DSP_TRIG_BG	0 : 禁用1 : 启用	带隙电压基准的刷新由DSP判断的开始触发

### DSP启动

触发DSP的方式多种多样。

在从机模式下：

- 通过外部控制器触发。方法是发送指令“CDC开始转换”或“DSP\_TRIG”。

在独立模式下：

- 通过引脚触发。触发引脚在配置参数DSP\_STARTPIN和PG0\_X\_PG2/PG1\_X\_PG3中选择，范围为PG0至PG3。信号FFx触发DSP。FFx必须在固件中通过设置DSP\_x（如BitS\_DSP\_2）进行复位。

BitC DSP\_2

- 触发时间至
  - CDC
  - RDC
  - 计时器

或者通过中断。该选项由配置参数DSP\_START\_EN选择。

(提示：DSP 也由

- 将RUNBIT从0切换到1。由标志“AWAKE\_TRIGGERED\_N”指示。
- 每次CDC或RDC循环后。由FlagsTDC\_C\_TRIGGERED\_N和TDC\_R\_TRIGGERED\_N指示

### 看门狗

看门狗基于OLF时钟，无论DSP是否停止都在计数。如果DSP在9秒到15秒内未重置看门狗，将触发上电复位，自动启动。状态标志POR\_FLAG\_Wdog被设置。

看门狗用于处理没有运行CDC或RDC的情况。

在作为从机的应用中，必须禁用看门狗。可以通过写入0x5A到WD\_DIS实现。如果使用看门狗，应在任何SIF通信之前禁用它。

### 系统复位

如果PCap04作为从机操作，而不是自启动模式，则在通电后需要执行以下操作：

1. 通过串行接口发送操作码“上电复位”，操作码为0x88。
2. 通过“写入到SRAM”操作码将固件写入SRAM。
3. 通过“写入配置”操作码写入配置寄存器，寄存器47（带有RUNBIT）必须是最后一个写入的寄存器。
4. 发送启动命令，操作码0x8C

### 指令集

PCap04的完整指令集由29条核心指令组成，具有由CPU解码的唯一操作码。此外，ams 提供一套库，包括常用常数定义和数学运算该库系列旨在不断扩展，在软件开发过程中提供极大帮助。

图128：

简单算术	杂项	内存访问	位操作
加法	初始化	径向	非
符号	无操作	清除	和
子	rst	加载	or
增量	停止	加载到指数	异或
	宽动态范围	数据传输与堆栈操作	

复数运算	移位与旋转	无条件跳转	按位
除法	左移	跳转到 jsb	位C
乘法	右移	jrt	位

条件跳转		
jcd	jEQ	jOfIC
jCarC	jNE	jOfIS
jCarS	jNeg	jPOS

## 说明

和	按位与
语法:	and p1,p2
参数:	p1 = ACCU [a,b,r]p2 = ACCU [a,b,r]p1 != p2
微积分:	p1 : p1 与 p2
影响的标志:	COSZ
字节:	1
描述:	按位与（与操作）
类别:	按位运算

加	加法
语法:	添加 p1,p2
参数:	p1 = ACCU [a,b,r] p2 = ACCU [a,b,r]
计算:	p1 : p1 + p2
影响的标志:	COSZ
字节数:	1
描述:	两个寄存器的相加
类别:	简单算术

bitC	清除单个位
语法:	bitC p1
参数:	p1 = 0到15的数字
计算:	设置DSP输出位的第p1位
影响的标志:	-
字节:	1
描述:	清除DSP输出位中的单个位
类别:	位运算

位	设置单个位
语法:	bitS p1
参数:	p1 = 数字0到15
微积分:	将DSP输出位的第p1位设置为1
影响的标志:	-
字节:	1
描述:	在DSP输出位中设置单个位
类别:	按位

清除	清除寄存器
语法:	清除 p1
参数:	p1 = ACCU [a,b,r]
计算:	p1 : 0
影响的标志:	S Z
字节:	2
描述:	寄存器清零
类别:	内存访问

div	无符号除法
语法:	div
参数:	-
计算:	单个div代码: $b : (a/r)$ , $a$ :余数 * $2^N$ div代码: $b : (a/r)*2^{(N-1)}$ , $a$ :余数 * $(2^N)$
影响的标志:	S Z
字节:	1
描述:	两个32位寄存器的无符号除法。当使用一次div指令时，结果商赋值给寄存器'b'。余数可以从'a'中计算。当连续使用N个div指令时，b中的结果为: $(a/r)*2^{(N-1)}$ 。另见ROM例程div_xx。在执行第一次除法步骤前，必须满足以下条件: 'b' = 0，且 $0 < a < 2^r$ 。如果不满足此条件，可以通过移位'a'直到满足。移位后，如果 $a \rightarrow a * (2 \uparrow ea)$ 和 $r \rightarrow r * (2 \uparrow er)$ ，则在 N 除法步骤中，结果商为 $b: (a/r) * 2^{(1+ea-er-N)}$ ，a为余数乘以( $2^N$ )。
类别:	复杂算术

inc	寄存器递增
语法:	inc p1
参数:	p1 = ACCU [a,b,r]
计算:	p1 : p1 + 1
影响的标志:	COSZ
字节:	1
描述:	增量寄存器
类别:	简单算术

初始化	初始化重置
语法:	初始化
参数:	-
微积分:	-
受影响的标志:	COSZ
字节:	1
描述:	初始化与重置。 重置CDC、RDC和CPU。将配置从NVRAM复制到配置寄存器中。
类别:	杂项

jCarC	跳转携带清除
语法:	jCarC p1
参数:	p1 = 跳转标签
微积分:	如果 (carry == 0) 则 PC : p1
影响的标志:	-
字节:	2
描述:	跳转时清除进位标志。若进位清零，程序计数器将设置为目标地址。目标地址由跳转标签指定。条件跳转不涉及堆栈，因此不能通过jrt返回。如果目标地址超出当前地址 (PC) 范围的-128/+127字节，则汇编软件会用以下优化替代此指令: jCarS new_label jsb p1 jrt new_label: 在这种情况下，堆栈将加载p1，因此堆栈容量将减少一个。
类别:	条件跳转

jCarS	跳转到携带集
语法:	jCarS p1
参数:	p1 = 跳转标签
微积分:	如果 (进位 == 1) PC : p1
受影响的标志:	-
字节:	2
描述:	在设置进位时跳转。如果设置了进位，程序计数器将跳转到目标地址。目标地址由跳转标签指定。条件跳转不影响堆栈，因此不能通过jrt返回。如果目标地址超出当前地址 (PC) 范围的-128/+127字节，汇编软件会用以下优化替代此指令：jCarC new_label jsb p1 jrt new_label：在这种情况下，堆栈将加载p1，因此堆栈容量将减少一个。
类别:	条件跳转

jcd	条件跳转
语法:	jcd p1, p2
参数:	p1 = 标志或输入端口位[63...0]。详见第2.3节DSP输入。p2 = 跳转标签
计算:	如果 (p1 == 1) 则PC: p2
影响的标志:	-
字节:	2
描述:	如果p1指示的位为1，则程序计数器设置为目标地址。目标地址通过跳转标签指定。条件跳转不操作堆栈，因此无法通过jrt返回。如果目标地址超出当前地址 (PC) 范围的-128/+127字节，则汇编软件会用以下优化替代此指令：jcd p1, new_label1 jsb new_label2 jrt new_label1：jsb p2 jrt new_label2：i...
类别:	条件跳转

jEQ	跳转到相等
语法:	jEQ p1
参数:	p1 = 跳转标签
微积分:	if (Z == 0) PC : p1
影响的标志:	-
字节:	2
描述:	在相等或零时跳转。如果前述结果为零，程序计数器将设置为目标地址。目标地址由跳转标签指定。条件跳转不影响堆栈，因此不能通过jrt返回。如果目标地址超出当前地址（PC）范围的-128/+127字节，汇编软件会用以下优化替代此操作码：jNE new_labeljsb p1jrtnew_label：在这种情况下，堆栈将加载p1，因此堆栈容量将减少一个。
类别:	条件跳转

jNE	不等跳转
语法:	jNE p1
参数:	p1 = 跳转标签
微积分:	如果 (Z == 1) 则PC: p1
影响的标志:	-
字节:	2
描述:	跳转条件为不等或非零。如果上述条件成立，程序计数器将跳转到目标地址。目标地址由跳转标签指定。条件跳转不影响堆栈，因此不能通过jrt返回。如果目标地址超出当前地址（PC）范围的±128字节，汇编软件会用以下优化替代此指令：jEQ new_labeljsb p1jrt new_label：在这种情况下，堆栈将加载p1，因此堆栈容量将减少一个。
类别:	条件跳转

jNeg	跳转到负面
语法:	jNeg p1
参数:	p1 = 跳转标签
微积分:	if (S == 1) PC : p1
受影响的标志:	-
字节:	2
描述:	在负数时跳转。如果前述结果为负（位 $31 == 1$ ），程序计数器将设置为目标地址。目标地址由跳转标签指定。如果目标地址超出当前地址（PC）范围的-128/+127字节，则汇编软件会用以下优化替代此指令：jPos new_labeljsb p1jrtnew_label：在这种情况下，堆栈将加载p1，因此堆栈容量将减少一个。
类别:	条件跳转

jOvIC	溢出清除跳转
语法:	jOvIC p1
参数:	p1 = 跳转标签
微积分:	如果 ( $O == 0$ ) PC : p1
影响的标志:	-
字节:	2
描述:	溢出清零时跳转。如果前一操作的溢出标志清除，程序计数器将跳转到目标地址。目标地址由跳转标签指定。条件跳转不影响堆栈，因此不能通过jrt返回。如果目标地址超出当前地址（PC）范围的-128/+127字节，汇编软件会用以下优化替代此指令：jOfIS new_labeljsb p1jrtnew_label：在这种情况下，堆栈将加载p1，因此堆栈容量将减少一个。
类别:	条件跳转

jOvIS	溢出设置跳转
语法:	jOvIS p1
参数:	p1 = 跳转标签
微积分:	if (O == 1) PC : p1
影响的标志:	-
字节:	2
描述:	设置溢出跳转。若前一操作的溢出标志被置位，程序计数器将跳转到目标地址。目标地址由跳转标签指定。条件跳转不影响堆栈，因此不能通过jrt返回。如果目标地址超出当前地址（PC）范围的-128/+127字节，汇编软件会用以下优化替代此指令：jOfIC new_labeljsb p1jrtnew_label：在这种情况下，堆栈将加载p1，因此堆栈容量将减少一个。
类别:	条件跳转

jPos	正向跳转
语法:	jPos p1
参数:	p1 = 跳转标签
微积分:	如果 (S == 0) 则PC: p1
影响的标志:	-
字节:	2
描述:	跳转到正数。若前述结果为正，程序计数器将设置为目标地址 (Bit 31 == 0)。目标地址由跳转标签指定。条件跳转不影响堆栈，因此不能通过jrt返回。如果目标地址超出当前地址（PC）范围的-128/+127字节，汇编软件将用以下优化替代此操作码：jNeg new_labeljsb p1jrtnew_label：在这种情况下，堆栈将加载p1，因此堆栈容量将减少一。
类别:	条件跳转

jrt	从子程序返回
语法:	jrt
参数:	-
计算:	PC : 来自jsub-call的PC
受影响的标志:	-
字节数:	1
描述:	从子程序返回。子程序可以通过'jsb'调用，使用jrt退出。程序将在jsb调用后的下一条指令继续执行。必须用jrt关闭子程序，否则不会返回。堆栈减1。
类别:	无条件跳转

跳转	无条件相对跳转
语法:	goto p1
参数:	p1 = jumplabel
微积分:	PC : p1
影响的标志:	-
字节:	2
描述:	跳转到jumplabel。程序计数器将被设置为目标地址。目标地址通过使用jumplabel指定。goto命令不维护堆栈，因此无法通过jrt返回。如果目标地址超出当前地址（PC）范围的±128字节，则汇编软件会用以下优化替代此指令：goto new_label1:jsb new_label2:jrtnew_label1:jsb p2:jrtnew_label2:i...
类别:	无条件跳转

jsb	无条件跳转
语法:	jsb p1
参数:	p1 = 跳转标签
微积分:	PC : 来自jsub-call的程序计数器
受影响的标志:	-
字节:	2
描述:	无条件跳转到子程序。程序计数器由跳转标签提供的地址加载。子程序执行直到出现关键词“jrt”。然后执行跳转回，并继续执行jsub-call之后的下一条指令。此操作码暂时占用程序计数器堆栈中的位置（见下文说明）。堆栈增加1。
类别:	无条件跳转

加载	加载累加器
语法:	load p1,p2
参数:	p1 = ACCU [a,b] p2 = 6..32位整数（正负，十进制或十六进制）
计算:	p1 : p2
受影响的标志:	S Z
字节:	3..8 (取决于p2)
描述:	将常数移动到p1 (p1=ACCU, p2=NUMBER) 不允许执行以下指令: load r, NUMBER  此指令是一个宏，替换为以下操作码: rad NUMBER[23:18]rad NUMBER[17:12]rad NUMBER[11:6]rad NUMBER[5:0]rad rad_stack_24bmove [a, b], r这里24位数字被分成四部分, [xx:yy]表示每一部分的比特范围。在操作过程中, RAM地址指针会发生变化, 请在编码时注意。
类别:	RAM访问

load2exp	将累加器加载为2的指数
语法:	load2exp p1,p2
参数:	p1 = 累加器 [a,b]p2 = 6位数字
微积分:	p1 : 2^p2
受影响的标志:	S Z
字节:	2
描述:	将 $2^{(p2)}$ 移动到p1 (p1=ACCU, p2=NUMBER) 不允许执行以下指令: load r, NUMBER 此指令是宏指令, 会被以下操作码替换: rad NUMBER[5:0] rad load2expmove [a, b], r
类别:	RAM访问

mov	移动
语法:	mov p1,p2
参数:	p1 = ACCU [a,b,r] p2 = ACCU [a,b,r]
计算:	p1 : p2
影响的标志:	S Z
字节:	1
描述:	将p2的内容移动到p1 汇编器也能识别旧的指令move
类别:	内存访问

乘法	乘
语法:	mult
参数:	-
计算:	$ab : (b * r)$
影响的标志:	S Z
字节数:	1
描述:	<p>ab和 r 寄存器内容的无符号乘法。</p> <p>ab 是寄存器 a 和 b 的组合, 形成一个 64 位长的寄存器, 其中 'a' 取最高有效位, 寄存器 'b' 取最低有效位。结果存储在组合的寄存器 a 和 b 中。寄存器 'a' 必须事先清零。此指令仅执行一次乘法步骤, 要完成完整的 32 位乘法, 需要执行 32 次此指令。这虽然编码繁琐, 但优点是只执行所需的算术操作, 如果不需要完整的 32 位乘法而只需要 N , 其中 <math>N &lt; 32</math> , 则只需执行 N 次乘法步骤即可完成对应的 N 位乘法。一次乘法步骤后, 寄存器 'a' 包含 <math>((a+(b[0]*r))&gt;&gt;1)</math>, 寄存器 'b' 包含 {a[0], b[3 : 1]}。例如: 设寄存器 'a' 的各个位为 a[31] , a[30], a[29], ..., a[2], a[1], a[0], 并用 a[3:0] 表示 'a' 的一段位范围, 代表寄存器 'a' 的最低 4 位。然后, 一次乘法步骤后, <math>a[30:0] = (a[31:0] + r[31:0] * b[0]) &gt;&gt; 1</math>, 其中 <math>&gt;&gt; 1</math> 表示右移一位; a[31] 的值为零, <math>b[31] = (a[0] + r[0] * b[0])</math>, 并且 <math>b[30 : 0] = b[31:1]</math>。寄存器 r 保持不变。</p>
类别:	复杂运算

nop	无操作
语法:	-
参数:	-
微积分:	-
影响的标志:	-
字节数:	1
描述:	占位符代码或时间调整 (无功能)
类别:	杂项

非	按位非
语法:	not p1
参数:	p1 = 累加器 [a,b,r]
微积分:	p1 : ~p1
受影响的标志:	COSZ
字节:	1
描述:	取反寄存器 (否定)
类别:	位运算

or	按位或
语法:	or p1,p2
参数:	p1 = ACCU [a,b,r] p2 = ACCU [a,b,r] p1 != p2
微积分:	p1 : p1   p2
影响的标志:	COSZ
字节:	1
描述:	按位或 (析取)
类别:	位运算

弹出	移除地址
语法:	弹出
参数:	-
微积分:	-
影响的标志:	-
字节:	1
描述:	回滚内存地址堆栈
类别:	内存访问

<b>rst</b>	上电复位
<b>语法:</b>	rst
<b>参数:</b>	-
<b>微积分:</b>	-
<b>受影响的标志:</b>	S Z
<b>字节数:</b>	5
<b>描述:</b>	这是一个符号操作码，等同于以下指令序列：bitC 54bitC 55bitS 55bitS 54bitC 55，汇编器也支持旧的PowerOnReset指令
<b>类别:</b>	杂项

<b>压入</b>	将数据放入堆栈内存
<b>语法:</b>	push p1
<b>参数:</b>	p1 = 数字 [6位]
<b>微积分:</b>	-
<b>影响的标志:</b>	-
<b>字节:</b>	1
<b>描述:</b>	将p1写入RAM地址堆栈（范围：0到63）。将常数值提交到ROM例程。push 22 ; CDC比率的小数位数push 4 ; 参考值的端口号（PC4）jsb _ROM_CDC_Note：仅供高级用户使用。建议使用rad
<b>类别:</b>	内存访问

rad	设置RAM地址指针
语法:	rad p1
参数:	p1 = 数字 [7位]
微积分:	-
受影响的标志:	-
字节:	2
描述:	将指针设置到RAM地址堆中的RAM地址（范围：0到127）。注意：内部RAM由2个页面组成，每页64个字。汇编器将位S/C和推送指令的组合转换为rad指令。rad 15move r, b 将移动 b 的内容到地址15
类别:	RAM访问

wdr	清除看门狗定时器
语法:	wdr
参数:	-
微积分:	-
影响的标志:	-
字节:	5
描述:	清除看门狗定时器。 这是一个符号操作码，等同于以下指令序列：bitC 54bitC 55bitS 54bitS 55bitC 54，汇编器也识别旧的resetWDG操作 码
类别:	其他

shiftL	左移
语法:	shiftL p1
参数:	p1 = 累加器 [a, b]
微积分:	p1 : p1 << 1
影响的标志:	S Z
字节数:	1
描述:	将p1左移 -> 将p1寄存器左移, 最低位用0填充, 最高位存入进位寄存器
类别:	移位与旋转

shiftR	向右移位
语法:	shiftR p1
参数:	p1 = ACCU [a, b]
微积分:	p1: p1>> 1
影响的标志:	S Z
字节数:	1
描述:	p1的带符号右移 -> 将p1向右移位, 最高位根据数字正负进行复制
类别:	平移与旋转

符号	符号
语法:	sign p1
参数:	p1 = ACCU [a,b]
微积分:	如果 SF = 0 => p1 :  p1 , SF : S(p1) 如果 SF = 1 => p1 : - p1 , SF : S(p1)
受影响的标志:	SZSF
字节:	1
描述:	此操作码的目的是在乘法或除法前取一个参数的绝对值，并在操作后恢复符号。假设符号标志为零，则取累加器的绝对值，并将其符号存入SF。第二次使用此操作码时，将从SF中恢复符号top1。假设零为正。
类别:	简单算术

停止	停止
语法:	停止
参数:	-
微积分:	-
受影响的标志:	-
字节数:	1
描述:	PCAP控制器停止。时钟发生器停止，PCAP控制器进入待机状态。可以通过外部事件如“看门狗定时器”、“外部开关”或“新的电容测量结果”实现重启。通常此指令是汇编列表中的最后一条命令。
类别:	其他

减	减法
语法:	sub p1,p2
参数:	p1 = ACCU [a,b,r] p2 = ACCU [a,b,r] p1 != p2
微积分:	p1: p1 - p2
影响的标志:	COSZ
字节数:	1
描述:	两个寄存器的减法。 以下指令不允许使用: sub a,a。 sub b,b。 sub r,r
类别:	简单算术

异或	按位异或
语法:	异或 p1,p2
参数:	p1 = ACCU [a,b,r] p2 = ACCU [a,b,r] p1 != p2
计算:	p1 : p1 ^ p2
影响的标志:	COSZ
字节数:	1
描述:	按位异或 (反值)
类别:	按位运算

## 指令详情

### 指针

将Cratio结果复制到RAM中的持久存储区

加载b, 6

加载a, \_\_\_\_sub\_cdc\_C0\_Ratio\_temp

rad DPTR1

移动器, a

加载a, C0\_Ratio\_RAM

辐射 DPTRO

移动器, a

jsb \_ ROM\_dma\_

rad\_at\_DPTR0到rad\_at\_DPTR3是间接寻址的特殊指令。\_at\_DPTR0到\_at\_DPTR3是固件中定义的特殊RAM地址284和287。地址105到108用作数据指针，命名为DPTRO到DPTPR3。

通过

CROLD

移动 r, a

将地址加载到 DPTRO。用

rad \_\_\_\_在\_DPTRO中

将DPTRO中的地址加载到。

示例1：将RAM内容按顺序从一个地址空间复制到另一个

加载 a, C0\_ratio

径向 DPTR1

移动 r, a

加载 a, RESO

径向 DPTRO

移动 r, a

加载b, 8

jsb \_\_\_\_ROM\_dma\_\_\_\_;调用ROM例程

示例2：将Rratio结果复制到持久存储区的RAM中

rad 4

rad rad\_stack\_6b

移动b, r

加载a, \_\_\_\_sub\_rdc\_R0\_Ratio\_temp; 复制源

辐射DPTR1

搬运器, a

辐射R0\_Ratio\_RAM; 复制目标

辐射rad\_stack\_6b

移动a, r

辐射DPTRO

移动器, a

jsb \_ ROM\_dma\_

## 带有压入和弹出操作的传递常数

子程序调用：

```
push FPP_CRATIO ; 栈-1 ---- 结果中的fpp数量  
push C_REF_PORT_NUMBER; 栈-0 ----> 参考端口编号  
jsb _ROM_cdc_ ; 调用ROM例程进行比例计算
```

```
_ROM_cdc_:  
bitS PAGESEL_OUT  
rad rad_stack_6b; (堆栈-0) 包含参考端口编号  
move b, r  
pop  
rad ____sub_cdc_RefPort  
move r, b ; 临时将参考端口编号存入RAM  
弹出  
弹出  
rad rad_stack_6b ; (栈 - 1) 包含结果中的小数位数  
结果 (cdc_fpp)  
移动 b, r ; B = 小数位数, Result_fpp
```

## 乘

指令“mult”仅为一次乘法步骤。要完成完整的32位乘法，该指令必须执行32次。乘数存放在累加器b和r中。每一步取b的最低位。如果为1，则将r加到累加器a，否则不加任何值。之后a和b向右移位。a的最低位变为b的最高位。在乘法的第一步之前，必须清零a。最终结果分布在累加器a和b中。通过使用ROM例程mult\_01到mult\_32，可以简化mult的使用。

在许多情况下，不需要执行全部32次乘法步骤，而只需执行更少的步骤。所需的步骤数由b的有效位数以及结果的有效位数决定。

但如果乘法步骤少于32次，结果可能分布在累加器a和b之间。通过对r中的乘数进行适当的右移，以及执行相应数量的乘法步骤，可以确保结果完全在a或b中。

通过推入和弹出传递常数

一种通过push和pop指令传递值在0到63之间常数的简便方法。以下是调用子程序的示例

子程序调用：

```
push FPP_CRATIO; 栈-1 -> 结果中的FPP数量
push C_REF_PORT_NUMBER; 栈-0 -> 参考端口编号
jsb _ROM_cdc_; 调用ROM例程进行比例计算
```

\_ROM\_cdc\_:

辐射 rad\_stack\_6b; (堆栈-0) 包含参考端口编号

移动

弹出

\_\_\_\_\_sub\_cdc\_RefPort

移动 r, b; 临时将参考端口号存入RAM

弹出

弹出

弧度 rad\_stack\_6b; (堆栈-1) 存放结果中的小数位数 (cdc\_fpp)

移动 b,r; B = 小数位数, Result\_fpp

### div

指令“div”与乘法一样，只是完整除法的单个步骤。完成除法所需的步骤数取决于结果的精度。被除数存放在累加器a中，除数在累加器 r 中。每个除法步骤包括以下操作：

- 左移b
- 比较a和 r。如果a大于或等于 r，则从a中减去 r，并将1加到b
- 左移a

起始条件：  $0 < a < 2 * r$ ,  $b = 0$

此外，为方便客户使用，ROM库中实现多步除法，调用div\_01至div\_32。例如，调用该库中的div\_24函数将执行24个除法步骤。结果存放在b中，余数在a中。

经过 N 除法步骤后，结果在 b :  $(a/r) + 2^{(N-1)}$ ,  
中：余数\*2的N次方。

示例1： $a = 2, r = 6$ ，整数除法

图129：

示例：除法 2/6

步骤	$a = 2$	<b>b</b>	$r = 6$	
	000000..000010	0.00000	0.0110	$a < r$ , 左移b, a
1	000000..000100	0.00000	0.0110	$a < r$ , 左移b, a
2	000000..001000	0.00000	0.0110	左移b, $a \geq r$ : $a -= r$ , $b += 1$ , 左移a
3	000000..000100	0.00001	0.0110	$a < r$ , 左移b, a
4	000000..000100	0.00010	0.0110	左移 b, $a \geq r$ : $a -= r$ , $b += 1$ , 左移 a
5	000000..000100	0.00101	0.0110	

$$\text{商} = b * 2^{(1-\text{steps})} = 0.3125, \text{余数}$$

$$= a * 2^{(-\text{steps})} = 4 * 2^{-5} = 0.125$$

以下两个更复杂的例子展示了除法相较于乘法的明显优势：在位级别上的分辨率由除法步骤的次数直接决定。有了这个知识，汇编程序可以非常高效地编写。只需使用必要的除法步骤数即可。

例子2： $A = 8.75, R = 7.1875$ ，带有4位小数的分数除法，A与R。

$$8.75/7.1875 = a^{*}2^{\exp A}/r^{*}2^{\exp R} = a^{*}2^{-4}/r^{*}2^{-4}$$

图130：

示例：除法 8.75/7.1875

步骤	$a = 140$	<b>b</b>	$r = 115$	
	1000 1100	0000 0000	01110011	左移 b, $a \geq r$ : $a -= r$ , $b += 1$ , 左移 a
1	0011 0010	0000 0001	0111 0011	$a < r$ , 左移 b, a
2	0110 0100	0000 0010	0111 0011	$a < r$ , 左移 b, a
3	1100 1000	0000 0100	0111 0011	左移b, $a \geq r$ : $a -= r$ , $b += 1$ , 左移a
4	1010 1010	0000 1001	0111 0011	左移b, $a \geq r$ : $a -= r$ , $b += 1$ , 左移a
5	0110 1110	0001 0011	0111 0011	$a < r$ , 左移b, a
6	1101 1100	0010 0110	0111 0011	左移b, $a \geq r$ : $a -= r$ , $b += 1$ , 左移a
7	11010010	0100 1101	0111 0011	左移b, $a \geq r$ : $a -= r$ , $b += 1$ , 左移a
8	1011 1110	1001 1011	01110011	

$$\text{商} = b * 2^{(1+\exp A - \exp R - \text{steps})} = 155 * 2^{(1-4+4-8)} = 1.2109$$

$$\text{余数} = a * 2^{(-\text{steps}-\exp R)} = 190 * 2^{-12} = 0.0463$$

示例3:  $A = 20, R = 1.2$ , 分数除法,  $R < A$ 。

$A$ 和 $R$ 向左移动以显示 $R$ 的小数位, 此外,  $R$ 必须继续向左移动, 直到大于 $A/2$ 。

$$20/1.2 = a * 2^{\exp A} / r * 2^{\exp R} = a * 2^{-4} / r * 2^{-8}$$

图131:

示例3: 除法 20/1.2

步骤	a = 320	b	r = 307	
	0001 0100 0000	0000 0000 0000	0001 0011 0011	左移b, $a \geq r$ 时: $a -= r$ , $b += 1$ , 左移a
1	0000 0001 1010	0000 0000 0001	0001 0011 0011	$a < r$ , 左移b, a
2	0000 0011 0100	0000 0000 0010	0001 0011 0011	$a < r$ , leftshift b, a
3	0000 0110 1000	0000 0000 0100	0001 0011 0011	$a < r$ , leftshift b, a
4	0000 1101 0000	0000 0000 1000	0001 0011 0011	$a < r$ , 左移b, a
5	0001 1010 0000	0000 0001 0000	0001 0011 0011	左移b, $a \geq r$ : $a -= r$ , $b += 1$ , 左移a
6	0000 1101 1010	0000 0010 0001	0001 0011 0011	$a < r$ , 左移b, a
7	0001 1011 0100	0000 0100 0010	0001 0011 0011	左移b, 当 $a \geq r$ 时: $a -= r$ , $b += 1$ , 左移a
8	0001 0000 0010	0000 1000 0101	0001 0011 0011	$a < r$ , 左移b, a
9	0010 0000 0100	0001 0000 1010	0001 0011 0011	左移b, $a \geq r$ : $a -= r$ , $b += 1$ , 左移a
10	0001 1010 0010	0010 0001 0101	0001 0011 0011	左移位b, $a \geq r$ : $a -= r$ , $b += 1$ , 左移位a
11	0000 1101 1110	0100 0010 1011	0001 0011 0011	$a < r$ , leftshift b, a
12	0001 1011 1100	1000 0101 0110	0001 0011 0011	

$$\begin{aligned}
 & \text{商} \\
 & = b * 2^{(1+\exp A-\exp R-\text{steps})} = 2134 * 2^{(1-4+8-12)} = 16.6719 \\
 & \text{余数总是小于除数的一半, 例如在本例中, 余数} < 1.2/2^{12} \\
 & = 0,0003
 \end{aligned}$$

$$\text{步骤} = 1 + \exp A - \exp B - \exp \text{Res}$$

### 只读存储器例程

以下例程以只读存储器代码实现:

图132:  
只读存储器例程

只读存储器例程	地址	只读存储器例程	地址
_ROM_Version_____	1024	中位数	1804
TDC调度	1029	RDC	1936
CDC周期	1047	RDC逆变	1962
RDC周期	1068	_ROM_cdc_	2021
_ROM_cdc_initialize_	1086		
_ROM_rdc_initialize_	1111		
将shiftR_B_32移至_00	1140		
将shiftL_B_32移至_00	1173		
将shiftR_A_32移至_00	1206	_ROM存储器读_a_32b_到_ROM 存储器写_08b_	2470
shiftL_A_32到_00	1239	_ROM存储器存储_	2650
mult_32到_00	1272	_ROM存储器回忆_	2673
div_33到_00	1305	_ROM_二维校准	2696
_ROM_除变量_	1339	_ROM_三次多项式	2825
_ROM_多变量_	1396	_子多项式加载系数	2874
_ROM_偏移a变量_	1452	多项式_ROM_四阶	2923
偏移_ROM_b变量_	1478	脉冲_ROM_	2969
DMA_ROM_	1504	加载的校准值_ROM_脉冲	2990
输入_ROM_	1520	_ROM_NVblock_copy_32b_ 到 _ROM_NVblock_copy_08b_	3138
_ROM_log10_____	1547	_ROM电容多项式	3236
_ROM_Id_____	1574	_ROM电容多项式4维	3437
_ROM有符号24到有符号32	1648		

## ROM例程的切换参数和RAM地址

图133：  
关键ROM例程的参数和RAM

ROM例程	参数	RAM地址
_ROM_cdc_	_sub_cdc_C0_Ratio_temp 到 _sub_cdc_C5_Ratio_temp	58 ..63
ROM_rdc_____ _ROM_rdc_inverse_	_sub_rdc_R0_Ratio_temp 到 _sub_rdc_R3_Ratio_temp	64. 67
_ROM_mult_variable_	_sub_standard_multiplier_	82
_ROM_div_variable_	___sub_standard_divisor_	82

## 以下对ROM

例程进行详细描述。

提示：建议始终包含以下文件：

```
#device PCap04v1
#include <pcap_standard.h>
#include <PCap04_ROM_addresses_standard.h>
```

声明地址到ROM例程。

\_ROM\_Version\_

功能：	此例程返回芯片中ROM的8位版本号
输入参数：	-
输出/返回值：	A-ACCU ROM版本
前提条件	无
依赖其他头文件	无
函数调用	jsb _ROM_version_; 地址 0x400
临时内存使用情况	-
示例：	jsb _ROM_Version_ rad RES5 移动 r, a

\_ROM\_tdc\_dispatch\_

功能：	TDC库，在最开始调用。根据触发器调用子程序 _ROM_cdc_cycle_or_ROM_rdc_cycle_ 和 _tdc_awake_
输入参数：	-
输出/返回值：	-
前提条件	无
依赖其他头文件	无
函数调用	jsb _ROM_tdc_dispatch_
临时内存使用	-
示例：	org 0 ____SOP__: ; 程序开始 jsb _ROM_tdc_dispatch_

### \_ROM\_cdc\_cycle\_

功能：	1. 计算电容测量端口的TDC值的起止差，并将其累加到测量值RAM寄存器中（根据C_ADD_PTR），对应被测端口。2. 若计算的放电时间过大，则用0xFFFFFFFF替代，表示溢出。3. 如果还有未完成的测量，DSP继续执行；最后一次测量后，DSP返回调用ROM例程的地方，继续执行后续处理固件（如有）。
输入参数：	-
输出/返回值：	对应RAM (88-95) 中可获得CDC放电时间测量值
前提条件	此函数必须在每次电容端口的测量周期后调用。
依赖的其他头文件	无
函数调用	push <MSB(RAM_startaddress)>push <LSB(RAM_startaddress)>jsb _ROM_cdc_cycle_
临时内存使用	此例程中会覆盖DPTR0。
示例：	push 1 ; 压入起始地址的第6位push M0 ; 压入起始地址的第5到0位jsb _ROM_cdc_cycle (不推荐使用，建议使用 _ROM_tdc_dispatch_)

\_ROM\_Rdc\_cycle\_

功能：	1. 计算温度测量端口的TDC值的起止差，并将其累积到测量值RAM寄存器中（根据R_ADD_PTR），对应被测端口。2. 如果计算出的放电时间过大，则用0xFFFFFFFF替代，表示溢出。3. 如果RDC测量结束触发了DSP，则此ROM例程调用将在完成上述任务后停止DSP，否则，DSP将返回到调用ROM例程的位置并继续执行固件。
输入参数：	-
输出/返回值：	对应RAM (84-87) 中可获得RDC放电时间测量值
前提条件	此函数必须在每个电阻端口的测量周期后调用。
依赖的其他头文件	无
函数调用	push <MSB(RAM_startaddress)>push <LSB(RAM_startaddress)>jsb _ROM_rdc_cycle_
临时内存使用	在此例程中覆盖了DPTRO。
示例：	push 1 ; 起始地址的第6位push TM0; 起始地址的第5..0位jsb _ROM_rdc_cycle (不推荐使用，建议改用 _ROM_tdc_dispatch_)

### \_ROM\_cdc\_initialize\_

功能：	此ROM例程将所有测量值RAM寄存器（地址88-95）清零。
输入参数：	待清零的RAM寄存器起始地址（88）在调用此子程序前应已压入RAM地址栈中。
输出/返回值：	-
前提条件	-
依赖其他头文件	无
函数调用	push <MSB(RAM_startaddress)>push <LSB(RAM_startaddress)>jsb _ROM_cdc_initialize_
临时内存使用	在此ROM例程中，地址DPTR0和_at_DPTR0被覆盖。
示例：	push 1 ; 起始地址的第6位push M0 ; 起始地址的5..0位jsb _ROM_cdc_initialize

### \_ROM\_rdc\_initialize\_

功能：	此ROM例程将所有测量值RAM寄存器（地址84-87）清零。
输入参数：	要清除的RAM寄存器（84）的起始地址，在调用此子程序前应将其压入RAM地址栈中。
输出/返回值：	-
前提条件	无
依赖的其他头文件	无
函数调用	push <MSB(RAM_startaddress)>push <LSB(RAM startaddress)>jsb _ROM_rdc_initialize_
临时内存使用	在此ROM例程中，地址DPTR0和_at_DPTR0会被覆盖。
示例：	推送1；起始地址的第6位推送M0；起始地址的5..0位跳转到_ROM_rdc_initialize

左移\_A\_xx; 右移\_A\_xx; 左移\_B\_xx; 右移\_B\_xx

功能：	调用A-Accu或B-Accu固定次数xx的左右移位的函数
输入参数：	A-ACCU或B-ACCU : 要移位的次数xx
输出/返回值：	shiftL_A_01 A: A * 2^(1) shiftL_A_02 A: A * 2^(2) shiftL_A_03 A: A * 2^(3) shiftL_A_31 A: A * 2^(31)
前提条件	无
依赖其他头文件函数调用	无
	jsb shiftL_A_01 jsb shiftR_A_01 ..... jsb shiftL_A_32 jsb shiftR_A_32  jsb 左移_B_01 jsb 右移_B_01 ..... jsb 左移_B_32 jsb 右移_B_32
临时内存使用	
示例：	加载b, dp_const_mjsb shiftL_B_08

mult\_xx

功能：	调用可变次数乘法步骤的函数
输入参数：	B-Accu : 乘数1 R-Accu : 乘数2
输出/返回值：	A-ACCU : 乘法结果
前提条件	A-Accu : 0B >= 0 R >= 0
依赖其他头文件	无
函数调用	jsb mult_01.jsb mult_32
临时内存使用	
示例：	符号 b；存储乘数 b 的符号；从 bclearrad_sub_standard_multiplier_jsbmult_15 取绝对值；a2 * theta 在 A-Akkusigna 中；将乘数的符号还原到结果中

## div\_xx

功能：	div_00 至 div_33：调用固定次数除法步骤的函数
输入参数：	A-Accu : 被除数 r : 除数
输出/返回值：	B-Accu : 被除数 / 除数
前提条件	B-Accu : 0 $0 < 'a' < 2 * 'r'$ 分别结果b需要是 $0 < b < 2$
依赖其他头文件	无
函数调用	jsb div_xx
临时内存使用	
示例：	<pre> rad M_internal_refm ove b, rshiftL b shiftL b ; b : b + 4, 确保被除数大于 2*arad 0move r, brad M0move a, r  ; a = M0 清除 brad 0 ; r = M_internal_ref jsb ; b = a/r = M0/M_internal_ref     </pre> <p style="text-align: center;"><small>&lt;div_29&gt;/&lt;/div_29&gt;</small></p>

\_ROM\_div\_variable\_

功能：	调用可变除法步骤的函数
输入参数：	B-Accu：除法步骤数A- Accu：被除数 参数 - 0：除数 (RAM地址82)
输出/返回值：	B-Accu : 被除数/除数
前提条件	$0 < 'a' < 2 * 'r'$ 分别结果b需要是 $0 < b < 2$
依赖其他头文件	无
函数调用	jsb _ROM_div_variable_
临时内存使用	
示例：	加载 a, 3 ; 除数 = 3 rad_arguments_ ; 通过 ramaddress 82 传递 hand over ; 将 r 移动到 aload a, 4 ; 被除数 = 4 加载 b, 23; 23 除法步骤 jsb _ROM_div_variable_ ; rad RESOOmover, b; B : 0x55555555 ; B : 4/3 = 1.333333 (fpp22)

### \_ROM\_mult\_variable\_

函数：	调用变量次数的乘法函数
输入参数：	B-Accu：乘数1A-ACCU ：乘法步骤数参数 - 0: 乘数2 (RAM 地址82)
输出/返回值：	-Accu : 乘积结果
前提条件	乘数1 > 0 乘数 2 > 0
依赖其他头文件	无
函数调用	jsb _ROM_mult_variable_
临时内存使用	
示例：	<pre>load b, 3 ; 乘数1 = 3 rad _arguments_ ; 通过ramaddress 82传递, bload b, 4 ; 乘数2 = 4 load a, 32; 32次乘法步骤 jsb _ROM_mult_variable_ ; rad RESOOmove r, b; B : 12____ ; B : 4*3 = 12 (fpp22)</pre>

\_ROM\_shift\_a\_variable\_

功能：	调用可变数量的A-Accu步数移位的函数
输入参数：	A-ACCU : 待移位的值 B-Accu 移位步数 B > 0 : 左移 B < 0 : 右移
输出/返回值：	A: $A * 2^B$
前提条件	无
依赖其他头文件	无
函数调用	jsb_ROM_shift_a_variable_
临时内存使用	
示例：	<pre> rad ; a : MyVar; MyVarmov e a, rload b, -4 ; 设置四倍右移 jsb_ROM_shift_a_variable_rad RESOOmove r, a </pre>

\_ROM\_shift\_b\_variable\_

功能：	调用可变次数移位B-Accu步骤的函数
输入参数：	B-Accu : 要移位的值 A-ACCU : 移位步数 A > 0 : 左移 A < 0 : 右移
输出/返回值：	$B : B * 2^A$
前提条件	无
依赖其他头文件	无
函数调用	jsb_ROM_shift_b_variable_
临时内存使用	
示例：	<pre> rad MyVar ; b : MyVar; move b, rload a, 13 ; 设置13倍左移 jsb_ROM_shift_b_variable_rad RESOOmove r, b </pre>

### \_ROM\_dma\_

功能：	“直接内存访问”——此例程将连续的RAM内容从一个地址空间复制到另一个地址空间。可以指定要复制的RAM值的数量。
输入参数：	B-Accu : 要复制的值的数量 DPTR1 源RAM块地址 DPTRO 目标RAM块地址
输出/返回值：	内容，即从源RAM块复制的指定数量的值到目标RAM块。
前提条件	-
函数调用	jsb _ROM_dma_
临时内存使用	-
RAM是否永久更改？	是的，目标RAM块
示例	<pre>; 将Cratio结果复制到持久存储区 ; 在RAM中 (cells_sub_cdc_C0_Ratio_temp to_sub_cdc_C0_Ratio_temp +5 转换为cells C0_Ratio_RAM 到 C0_Ratio_RAM + 5) 加载b, 6加载a, __sub_cdc_C0_Ratio_temprad DPTR1移动, 加载a, C0_Ratio_RAMrad DPTROmover, ajsb _ROM_dma_</pre>

\_ROM\_In\_, \_ROM\_log10\_, \_ROM\_Id\_

功能：	对数计算：log10（以10为底的对数）In（自然对数）Id（以2为底的对数）AccuA = Id(AccuA)  所有对数中，首先确定 $\ln(x)$ 。对于 log10 和 In，返回值随后除以 $\ln(10)$ 或 $\ln(e)$ ，对应 $\ln(x)$ 的公式参考 <a href="http://de.wikipedia.org/wiki/Logarithmus#Nat.C3.BCrlicher_Logarithmus">http://de.wikipedia.org/wiki/Logarithmus#Nat.C3.BCrlicher_Logarithmus</a>
输入参数：	A-Accu：包含以 10fpp 为底的对数双参数的参数
输出/返回值：	A-Accu：有符号16位值
前提条件	
函数调用	jsb _ROM_Id_jsb _ROM_In_jsb _ROM_log10_
临时内存使用	_____临时变量_____ - 3 到 _____临时变量_____
内存是否永久更改？	
示例	加载 a, 0xA01; A: 2.500977 (=2561*2^-10)jsb _ROM_Id_rad RESOOmove r, a; A: 1.322492 (0x1528E*2^-16)

\_ROM\_signed24\_to\_signed32\_

函数：	将24位有符号数转换为32位有符号数。此函数用于将24位有符号值e转换为32位有符号值。
输入参数：	Accu B：有符号24位值
输出/返回值：	Accu B：有符号32位值
前提条件	无
依赖其他头文件	无
函数调用	_ROM_signed24_to_signed32_
临时内存使用	1
示例：	jsb _ROM_signed24_to_signed32_____

### \_ROM\_median\_

功能:	中值滤波器：这是一种准中值滤波器。滤波深度由堆栈中的参数定义。每当有新值(X)时，将其与当前中值进行比较。如果新值小于或等于中值，则将列表中的最后一个值替换为X；否则，将列表中的第一个值替换为X。之后，整个列表会被排序。列表中间的值即为新的中值。
输入参数:	B-Accu : 滤波输入（同时也是输出） 堆栈-2 : 过滤器存储器的起始地址 堆栈-1 : 过滤器存储器的中间部分 堆栈-0 : 过滤器存储器的最后地址 滤波器阶数由最后地址减去起始地址定义。必须为其预留必要的内存区域，且不得用于其他用途
输出/返回值:	B-Accu : 滤波器输出（及输入）
前提条件	无
依赖其他头文件	无
函数调用	jsb_ROM_median_
临时内存使用	____temporary_variables_-4 到 ____temporary_variables_FILTER_START 到 FILTER_STOP
示例:	常数滤波器阶数 5常数滤 波器起始点 12 常数滤波器中间点 12 + (滤波器阶数/2)常数滤波 器终止点 12 + 滤波器阶数rad滤波器输入值移除 b, r推送滤波器起始点推送滤波器中间点推送滤 波器终止点jsb_ROM_median_

\_ROM\_cdc\_

功能：	此子程序用于确定电容比（或倒比）；根据测量方案和补偿模式而定
输入参数：	堆栈1 : CDC_FPP 结果中的小数位数。 堆栈0 : C_REF_PORT_NUMBER 参考端口编号（接地为0到6，浮空为0到2）。A-ACCU:sub_cdc_gain_corr_factor 用于增益校正，在Mi,8fpp 0 0上应用：增益校正=1，否则为1+ (A-Accu)FLAG_CDC_INV (位5) 在FLAGREG中：0表示反向电容比，1表示电容比
输出/返回值：	地址_sub_cdc_C0_Ratio_temp到_sub_cdc_C5_Ratio_temp将更新为相关的电容比（或其倒数）
前提条件	
函数调用	jsb _ROM_cdc_
临时内存使用	地址58到72
示例：	load2exp a, FLAG_CDC_INV ; 清除FLAGREG的FLAG_CDC_INV (第5位) 弧度 FLAGREG 和 r, a  载入 a, 0x40 ; 默认增益校正系数1.25  弧度 27 ; 堆栈 - 1 --> 结果弧度中的fpp数量；堆栈 - 0 -> 参考端口编号jsb _ROM_cdc_ ; 调用ROM例程进行比例计算rad _sub_cdc_C1_Ratio_tempmove a, r; 保存CDC的比例

\_ROM\_rdc\_; \_ROM\_rdc\_inverse\_

功能:	用于计算温度测量端口TM0 -> 内部参考电阻端口； TM1 -> 外部电阻端口， PT0TM2 -> 外部电阻端口， PT1TM3 -> 内部传感器电阻端口的电阻比或逆比的子程序
输入参数:	堆栈0 : 结果的定点位置FPP_RRATIO A-ACCU :REF_PORT_NUMBER 参考端口号， 0=内部 1=外部
输出/返回值:	_sub_rdc_R0_Ratio_temp TMO/Ref 或 Ref/TMO _sub_rdc_R1_Ratio_temp TM1/Ref 或 Ref/TM1 _sub_rdc_R2_Ratio_temp TM2/参考或参考/TM2 _sub_rdc_R3_Ratio_temp TM3/参考或参考/TM3
前提条件	无
依赖其他头文件	#include <memory.h>
函数调用	jsb_ROM_rdc_ jsb_ROM_rdc_inverse_
临时内存使用	_____临时变量_____ - 8 到 _____临时变量_____
示例:	加载 a, 0 ; 参考端口号 = 0rad25 ; 结果中的fpp数量； A-Akku：参考端口号（0 ... 3）； 堆栈 - 0：结果的fpp值 = _rdc_fpp_jsb_ROM_rdc_； 确定RDC比率 _ROM_rdc_inverse_； 用以确定逆RDC比率rad _sub_rdc_R1_Ratio_temp； 保存RDC比率move a, r

\_ROM\_memory\_ (读写易失性存储器)

功能:	用于存取非易失性存储器 (NVRAM) 的函数 读函数: * 有符号/无符号, 8/16/24/32位 RAM 读入累加器A或B 写函数: * 8/16/24/32位 (将数据写入RAM单元mem_data) RAM单元mem_add中的地址在读写操作中会自动递增。
输入参数:	-
输出/返回值:	Accu-A或Accu-B包含来自NVRAM的指定值
前提条件	无
依赖其他头文件	#include <memory.h>
函数调用	jsb_ROM_memory_rd_a_32b_ jsb_ROM_memory_rd_a_u24b_; jsb _ROM_memory_rd_a_s24b_jsb_ROM_memory_rd_a_u16b_; jsb _ROM_memory_rd_a_s16b_jsb_ROM_memory_rd_a_u08b_; jsb _ROM_memory_rd_a_s08b_jsb_ROM_memory_rd_b_32b_jsb _ROM_memory_rd_b_u24b_; ..._ROM_memory_wr_32b_;_ROM_memory_wr_24b_ROM_me mory_wr_16b_;_ROM_memory_wr_08b_
临时存储器使用	

示例：	<p>单次读取： 加载 a, &lt;read_mem_addr&gt; ; 内存地址 : &lt;read_mem_addr&gt;rad mem_addmove r, ajsb _ROM_memory_rd_a_u24b_Single 写入： 加载 a, MEM_WE ; 启用内存写入；（禁用写保护） rad mem_ctrlmove r, aload a, &lt;write_address&gt; ; 内存地址 : &lt;write_mem_addr&gt;rad mem_addmove r, aload a, &lt;存储数据&gt; ; 内存数据 : &lt;存储数据&gt;rad mem_datamove r, ajsb _ROM_memory_wr_32b_ ; 可选： 设置写保护加载 a, MEM_WR_PROTECTrad mem_ctrlmove r, a自动递增： 加载 a, 900 ; 地址（十进制） 900rad mem_addmove r, aload a, 0x012345; 数据 : 0x012345rad mem_datamove r, ajsb _ROM_memory_wr_24b_ ; 加 900: 0x45; 加 901: 0x23; 加 902: 0x01; 此操作后内存地址： 903</p>
-----	--

\_ROM\_MEMORY\_RECALL/STORE

功能：	非易失性存储器访问功能 所有非易失性存取均由RAM单元mem_ctrl保护。如需进行存储或回调操作，必须通过专用代码启用mem_ctrl以防止误操作。存储：mem_ctrl : MEM_STORE (0x2d00) 回调： mem_ctrl : MEM_RECALL (0x5900)
输入参数：	
输出/返回值：	
前提条件	无
依赖其他头文件	<memory.h>
函数调用	jsb _ROM_memory_store_jsb _ROM_memory_recall_
临时内存使用	
示例	存储： 加载 a, 存储到内存, 控制信号move r, a  跳转到 _ROM_memory_store_Recall : 加载 a, 存储到内存, 控制信 号move r, a, 跳转到 _ROM_memory_recall_

## \_ROM\_2点校准

功能:	该函数执行给定输入值集的两点校准。函数的数学公式如下（计算xi）：( xi_at_ccp1 - xi_at_ccp2 )xi = ( ci - ci_at_ccp1 ) + xi_at_ccp1( ci_at_ccp1 - ci_at_ccp2 )，输入值，即xi_at_ccp1、xi_at_ccp2、ci_at_ccp1、ci_at_ccp2，定义在校准数据中
输入参数:	<p>A-ACCU :  FPP_ci  -  FPP_ccp       FPP_ci 输入电容比的小数位数      FPP_ccp :电容比校准值的小数位数DPTRO)：包含常数的RAM起始地址，连续4个地址依次为：      xi_at_ccp1、xi_at_ccp2、ci_at_ccp1、ci_at_ccp2。堆栈0：输入电容比ci的RAM地址。注意：所有4个校准值的小数位数必须相同！！！</p>
输出/返回值:	A-Accu :带有FPP_ccp小数位的xi
前提条件	所有四个校准值必须具有相同的fpp。
函数调用	_ROM_2点校准_
临时内存使用	_____临时变量_-8 到 _____临时变量_load a, FPP差异； A 准确度 :( FPP_ci  -  FPP_ccp )弧度
示例:	<p>xi_at_ccp1；包含校准值列表的起始地址moveb, rradDPTRO；DPTRO 现在包含校准值的起始地址；r, brad C1_ratio；输入：；A 准确度 : (FPP_ci - FPP_ccp)；DPTRO : 校准值的起始地址</p> <p>move</p> <p>RAM中的值</p> <p>; 栈-0：输入电容比率ci的RAM地址；输出：；A 准确度：xi，带有FPP_ccp的小数位jsb _ROM_2PT_Calibrationad 2点结果；将返回值存储在RAM中move r, a</p>

\_ROM\_polynomial\_3rd\_degree

函数：	该函数计算给定输入值的三次多项式，系数已知。函数的数学表达式如下： $a_n = ((cc3n/xi + cc2n)/xi + cc1n)/xi + cc0n$ 其中cc3n、cc2n、cc1n和cc0n是三次多项式的系数。xi是给定的输入值，可以是电容或电阻比值。例如，可用于此形式的电容和电阻多项式。
输入参数：	DPTRO：包含以下内容的内存起始地址：cc3n：三次系数 cn_div3n：cc3n的除法步骤 cc2n：二次系数 cn_div2n：cc2n的除法步骤 cc1n：一次系数 cn_div1n：cc1n的除法步骤 ccOn：常数系数 A-Accu：输入电容或电阻比（或倒数比）
输出/返回值：	A-Accu：多项式的结果
前提条件	无
依赖其他头文件	无
函数调用	jsb _ROM_polynomial_3rd_degree
临时内存使用	_____temporary_variables_____
示例：	以下示例使用temperaturepolynomialrad Ratio_temp 计算温度；阻抗比；A-Akku包含R_ratioload b， cc3n_address；起始地址包含校准值列表；rad DPTRO；DPTRO现在包含系数和步长列表的起始地址； 输入：DPTRO：列表存储的起始地址；A-Akku：输入电 容或阻抗比（或逆比）；jsb _ ROM_polynomial_3rd_degree；A = theta rad theta move r, a

### \_ROM\_polynomial\_4th\_degree

函数：	该函数计算已知系数的三次多项式值。其数学表达式如下： $an = (((cc4n/xi + cc3n)/xi + cc2n)/xi + cc1n)/xi + cc0n$ 其中cc4n、cc3n、cc2n、cc1n和cc0n为三次多项式的系数。 xi为给定输入值，例如电容或电阻比值。此函数适用于此类电容和电阻的多项式计算。
输入参数：	DPTRO：包含以下内容的内存起始地址： cc4n：四阶系数 cn_div4n：cc4n的除法步骤 cc3n：三阶系数 cn_div3n：cc3n的除法步骤 cc2n：二阶系数 cn_div2n：cc2n的除法步骤 cc1n： 一次系数 cn_div1n：cc1n的除法步骤 cc0n：常数系数 A-Accu：输入电容或电阻比（或倒数比）
输出/返回值：	A-Accu：多项式的结果
前提条件	无
依赖其他头文件	无
函数调用	jsb _ROM_polynomial_4th_degree
临时内存使用	____temporary_variables_-3 到 ____temporary_variables_-1
示例：	以下示例使用temperaturepolynomialrad Ratio_temp计算温度；阻抗比；A-Akku包含 R_ratioload b, cc4n_address；校准值列表的起始地 址；rad DPTRO；DPTRO现在包含系数和步长列表的 起始地址；输入：DPTRO：列表存储器的起始地址； A-Akku：输入电容或阻抗比；（或逆比）b ROM_polynomial_4th_degree；A = θrad θ移动

\_ROM\_pulse\_

功能:	该函数用于确定脉冲输出，给定两个输入坐标 (result_1, pulse_out_1) 和 (result_2, pulse_out_2) 以及当前的result_n。输入坐标从NVRAM中复制，给定地址，必须满足前提条件。result_n、result_1和result_2可以是电容测量（如湿度、压力等）或温度测量（theta）的结果。将“result_n”转换为“Pulse_out”，然后可以赋值给PULSE0或PULSE1输出。该函数的数学公式如下：Pulse_out = (pulse_out_1 - pulse_out_2) * (result_n - result_1) + (pulse_out_1) * (result_1 - result_2) 此外，Pulse_out的范围被限制在pulse_out_min和pulse_out_max之间。常数pulse_out_1、pulse_out_2、result_1、result_2、pulse_out_min和pulse_out_max在校准存储器中定义，调用此函数前必须复制到RAM中。!!! 注意：result_n和result_1必须具有相同的格式。A-Accu: result_n的值B-Accu: 包含校准值列表的10位NVRAM起始地址
输入参数:	
输出/返回值:	A-Accu: 脉冲输出值（整数）
前提条件	result_1: 4字节（与result_2和result_n相同的浮点格式） result_2: 4字节（与result_1和result_n的fpp相同） 脉冲输出1: 2字节（整数）脉冲 输出2: 2字节（整数）脉冲输出 最大值: 2字节（整数）脉冲输出 最小值: 2字节（整数）
依赖其他头文件	-
函数调用	jsb _ROM_pulse_
临时内存使用	____临时变量_-8 到_临时变量_-
示例:	rad Z_result; 作为脉冲输出的值move a, rload b, result1_NVaddress; NVRAM中的起始地址；包含常数；输入：B-累加器：10位NVRAM地址；A-累加器：result_njsb _ ROM_pulse_的值；输出到A-累加器的是一个整数rad PULSE0；脉冲输出在PULSE0上move ring

### \_ROM\_pulse\_loaded\_cal\_vals

功能：	此ROM例程具有与_ROM_pulse_例程相同的功能。唯一的区别是，在调用此例程之前，固件必须将常数值从校准NVRAM内存复制到RAM中。
输入参数：	A-Akku: result_n的值 DPTR0: 包含常数值的RAM起始地址，连续6个地址，顺序为：result_1 (与result_2和result_n具有相同的fpp) result_2 (与result_1相同的 fpp) pulse_out_1 (整数) 脉冲输出2 (整数) 脉冲输出 最大值 (整数) 脉冲输出最 小值 (整数)
输出/返回值：	A-A电池：脉冲输出值为整数
前提条件	坐标值必须由固件从NVRAM复制到RAM空间。
依赖其他头文件	-
函数调用	jsb _ROM_pulse_loaded_cal_vals
临时内存使用	_____temporary_variables_- 8 到 _temporary_variables_-
示例：	<pre> d Z_result ; 作为脉冲输出的值move a, rload b, result1_RAMaddress ; RAM中的起始地址；包含 常数rad DPTR0mover, b; 输入：DPTR0: 起始 RAM地址；A-累加器：result_njsb _ROM_pulse_loaded_cal_vals的值；A中的输出为 整数_____move r, a </pre>

\_ROM\_NVblock\_copy\_

功能：	将一块数据从NVRAM复制到RAM
输入参数：	DPTRO: RAM的起始地址 B-Accu : NVRAM的起始地址 A-Accu: 要复制的值的数量 在FLAGREG中, SIGNED_VALUE_NV必须设置 (用于读取有符号值) 或清除 (用于读取无符号值) ——仅对24/16/08位值相关
输出/返回值：	RAM包含从NVRAM复制的指定数量的值
前提条件	无
依赖其他头文件	无
函数调用	jsb _ROM_NVblock_copy_32b_ ; jsb _ROM_NVblock_copy_24b_ ;jsb _ROM_NVblock_copy_16b_ ; jsb _ROM_NVblock_copy_08b_ ;
临时内存使用	<u>临时变量</u>
示例：	此示例将4个值 (每个32位) 从NVRAM复制到以xi_at_ccp1为起点的RAM地址：load a, RAM_address; DPTRO <- 起始RAM地址 rad DPTRO move ríngload2exp a, 2; 计数 =4 load b, NVRAM_address ; NVRAM起始地址；子程序将值从NVRAM复制到 RAM, ; 返回当前NVRAM地址 inB-Akku2 ROM_NVblock_copy_32b_

## ROM电容多项式

功能:	该函数根据电容比或其逆 (Cratio) 及温度 (theta) 计算电容多项式的值, $Z_{result} = (((a2 * theta) + a1) * theta) + a0$ 。a2、a1和a0是Cratio值的三次多项式， $= ((cc3n/xi + cc2n)/xi + cc1n)/xi + cc0n$ 其中cc3n .... cc0n是NVRAM 中三次多项式的系数
输入参数:	电容比或逆 (Cratio) DPTR1: 温度 (theta) 3的地址, NVRAM中系数值的起始地址Arg_6: Z_minArg_7: Z_maxFLAGREG, 位7 (LIN_3BYTE_COEFF): 1 → 3 每个系数中 的字节数 0 → 4 每个系数中的字节数
输出/返回值:	A-Accu: Z_result = 多项式的结果
前提条件	-
函数调用	_ROM_capacitance_polynomial_____
临时内存使用	_____临时变量_-8 到 _____临时变量_参数_0 到参数_7
示例:	<pre> ; 将Z_min和Z_max复制到参数_6和参数_7 加载a, 参数_6 ; DPTR0 &lt;- 起始参数 ; 内存地址 (RAM) rad DPTR0move ringload2exp a, 1  ; 计数=2 b, NV_Cal_vals; 校准起始地址; NVRAM中的值jsb _____ ROM_NVblock_copy_32b_ ; 复制Z_min和Z_max值的子程序, 从NVRAM到参数RAM ;-----   load a, C1_ratiорад  DPTR0; DPTR0: 电容比或逆比 (Cratio的逆) 地址      ; 比值或逆比 (Cratio的逆) move      r, a bad a a, theta; DPTR1: 温度电容比或逆比的地址, DPTR1移动器, 加载  b, NV_QUADRATIC_COEFFS; 系数地址 ;在NVRAM中的B- Accujsb_ROM_capacitance_polynomialradZ_resultmover, a </pre>

\_ROM\_capacitance\_polynomial\_4d

功能:	该函数根据电容比或其逆 (Cratio) 和温度 (theta) 计算电容多项式的值, Z_result = (((a2 * theta)+ a1) * theta )+a0。a2、a1 和 a0 是 Cratio 值的四次多项式, = (((cc4n/xi + cc3n)/xi + cc2n)/xi + cc1n)/xi + cc0n 其中 cc4n ... cc0n 是存储在 NVRAM 中的四次多项式系数。
输入参数:	电容比或其逆 (Cratio) 地址: DPTR1:1; 温度 (theta) 地址; NVRAM 中系数值的起始地址; Arg_6: Z_min; Arg_7: Z_max
输出/返回值:	A-Accu: Z_result = 多项式的结果
前提条件	-
函数调用	_ROM_capacitance_polynomial_____
临时内存使用	_____临时变量_-8 到 _____临时变量_参数_0 到参数_7
示例:	<pre> ; 将Z_min和Z_max复制到Arg_6和Arg_7 加载a a, Arg_6; DPTR0 &lt;- 起始参数; 内存地址 (RAM) rad DPTR0move r̄ingload2exp a, 1  ; 计数=2 b, NV_Cal_vals; 校准的起始地址; NVRAM中的值jsb ____ ROM_Nvblock_copy_32b_; 从NVRAM复制 Z_min和Z_max值到参数RAM  ;----- 加载a, C1_ratio  电容地址                                     比率或倒数 (Cratio) 移动      r, a 负载d a a, theta; DPTR1: 温度的地址  b, NV_QUADRATIC_COEFFS; 系数的地址; 在NVRAM中的B- Accujsb_ROM_capacitance_polynomial_4dradZ_resultmover, a </pre>

### 汇编程序

PCap04汇编器是一个多遍扫描的汇编器，将汇编语言文件转换为HEX文件，便于下载到设备中。为了方便，汇编器可以包含头文件。用户可以自行编写头文件，也可以集成ams提供的库文件。汇编程序由许多语句组成，包括指令和指令集。在前面的部分中，我们详细解释了指令。接下来的部分将介绍指令集和一些示例代码。

每行汇编程序只能包含一个指令或指令集语句。语句必须在一行内完成。

### 符号

符号是代表值的名称。符号由最多31个字符组成，字符集如下：

A – Z, a – z, 0 – 9, \_\_\_\_\_

符号不能以数字开头。汇编器区分大小写，因此需要注意这一点。

### 数字

数字可以用十六进制或十进制表示。十进制没有额外的说明符。十六进制以“Ox”开头表示。

### 表达式与运算符

表达式是符号、数字和运算符的组合。表达式在汇编时进行求值，用于计算原本难以确定的数值。

以下运算符具有相应的优先级：

级别	操作符	描述
1	0	括号，指定执行顺序
2	* /	乘法，除法
3	+ -	加法，减法

### 示例：

#### 常数值 1

等于  $((value + 3)/2)$

**指令**

汇编指令定义了汇编语言指令的处理方式。它们还提供了定义常量、预留内存空间和控制代码布局的可能性。指令本身不生成可执行代码。

下表概述了汇编指令。

图134：  
汇编指令概述

指令	描述	示例
CONST	常量定义, CONST [名称] [值], 值可以是数字、常数或两者之和	CONST Slope 42 CONST Slope 常数 + 1
LABEL:	跳转指令目标地址的标签。标签以冒号结尾。所有适用于符号名称的规则也适用于标签。	jsb LABEL1L ABEL1:...
;	注释, 可能用于解释代码的文本行。以分号字符开始。分号及其后所有字符将被汇编器忽略。注释可以单独成行, 也可以跟在指令后面。	; 这是一个注释
org	为后续语句设置程序存储器中的新起点。	org 0x23 equal 0x332211
equal	在程序存储器中插入三字节用户定义数据, 起始地址由org定义。	; 将0x11写入地址0x23, ; 将0x22写入地址0x24 ...
#device	指令定义, #device [dev_name] 可能是所用设备的名称。定义用于汇编器的库, 并定义子目录 \lib\ [dev_name], 用于包含的头文件或库文件。	#device PCap03-Z
#include	包含括号 <> 或引号 " " 中命名的头文件或库文件。代码将在包含命令所在行添加。括号中的名称指向带有定义子目录 \lib\ [dev_name] 的ams库。若不使用#device指令, 则指向固定子目录 \lib。引号中的内容可以是文件名 (如果在同一文件夹中), 也可以是完整路径。	#include <rdc.h>#include "rdc.h"

指令	描述	示例
#ifdef #elseif #endif	根据#define符号的值决定是否执行代码。例如，用于只在程序中包含一次头文件。	#ifdef __standard_h__#else#d efine __standard_h__...#en dif
#define	定义一个符号，在#ifndef指令分析时将被解释为真	

### 示例代码

以下展示了不同类型的编程循环的示例代码，  
用于加载指令和旋转指令的使用。

#### “for” 循环

汇编	C语言等价	注释
加载a, 最大值not ainc arad indexmove r, ado: ; { .. } rad indexincrjC arc do	for(index=-max; index < 0; index++)	max: 重复次数 最大补码 (~max+1) 存储 (-max) 到索引循环体循环递增重复while index < 0

#### “while” 循环

汇编器	C等价	注释
do: rad 表达式 move a, rjEQ 完成; {..}清除 ajEQ dodone;	while ( 表达式 ) { .. }	激活“表达式”的状态标志。若表达式为真，则跳转；无条件跳转到循环体，不写入程序计数器堆栈

#### do - while循环

汇编器	C等价	注释
do: .. 径向表达式 移动 a, r jNE do	do .. while (表达式)	循环体 激活状态标志 如果表达式为真则跳转 l = 0

## 带有两个指针的 do-while

汇编器	C 语言等价	注释
加载 a, MW7rad loopLimit  移动 r, a 加载 a, MW0 读取 DPTR0  移动 r, a 加载 a, RES0 辐射 DPTR1  移动 r, a do:  rad_at_DPTR0 移动 a 到 r rad_at_DPTR1 移动 r 到 a 循环限制 移动 a 到 r DPTR1 递增 径向DPTR0 递增寄存器a, r jCarS 执行	loopLimit = *MW7  ptrSource = *MW0;  ptrSink = *Res0;  执行 { *ptrSink++ = *ptrSource++ }  当 ( ptrSource <= MW7 ) 时	加载指针源的最大地址  用源地址加载 ptrSource  用接收器地址加载 ptrSink  循环体 从源加载值  写入值到接收器  将最大地址写入 a  递增接收地址  递增源 地址 limitLoop – ptrSource 如果 ptrSource <= 最大地址，则重复循环

### 将A寄存器右旋转到B寄存器

要将值从 Akku A 右旋转到 Akku B，Akku B 和 R 必须先设为零。之后，每次 mult 指令会执行一次“从 A 右旋转到 B””。此功能可用于将8位值移到寄存器的最高字节中，例如。

汇编器	C等价	注释
加载a, 0xa3清除 b移动r, b乘法； (8x) 乘 法-乘法	A = <U8bC>b = a << 40	

## 库

PICOCAP汇编器提供了实现库文件的可能性。通过这些库，固件可以以模块化方式编写。常用的库文件用于定义变量和常量名称。

当DSP需要由用户为特定应用编程，或需要修改固件时，这些库函数可以无需大量定制地集成到应用程序中。它们节省了已知、反复使用的重要功能的编程工作。有些库文件依赖于库中的其他文件。

在汇编软件中，库函数被称为头文件（扩展名为\*.hext），必须包含在主\*.asm程序中。库文件的路径应为\lib\[dev\_name]，位于汇编器所在的文件夹中。

以下是我们随汇编器一同提供的头文件，作为评估套件的一部分。

设备相关
pcap_standard.h
PCap04_ROM_addresses_standard.h
pcap_config.h

这些库函数的输入参数、输出参数、对RAM内容的影响等在下表中进行了说明。

注意事项：在标准固件和所有库文件中，符号“ufdN”用作注释，表示参数是有符号还是无符号，以及数字中的小数位数N。例如，ufd21表示该参数是具有21个小数位的无符号定点数。如果开头缺少u，则表示有符号数。

### pcap\_standard.h

功能：	这是PCap04固件项目的标准库。包含PCap04的主要地址映射和常量名称。注意：此文件应始终包含。它不包含命令，因此不会浪费程序空间。
定义（示例）：	<pre> ... ;- 临时变量和参数定义.CONST _arguments_82CONST _number_of_arguments_10 CONST ____临时变量____参数____1____参数数量 ____CONST Arg_0 ____参数____ - 8CONST Arg_1 __参数__ - 7...; - RAM地址 OUT (&amp;IN)CONST FLAGREG96CONST RESOO 97CONST RES01 98CONST RES02 99... </pre>

### PCap04\_ROM\_addresses\_standard.h

功能：	本文件声明了ROM例程的跳转标签
定义（示例）：	<pre> ... CONST_ROM_dma_0x5d1; 1489CONST_ROM_In_0x5e1 ; 1505CONST_ROM_log10_0x5fc; 1532..CONST_ROM_cdc_0x7d4; 2004  CONST_sub_cdc_C0_Ratio_temp0x3A; 58CONST_sub_cdc_C1_Ratio_temp0x3B; 59... </pre>

## pcap\_config.h

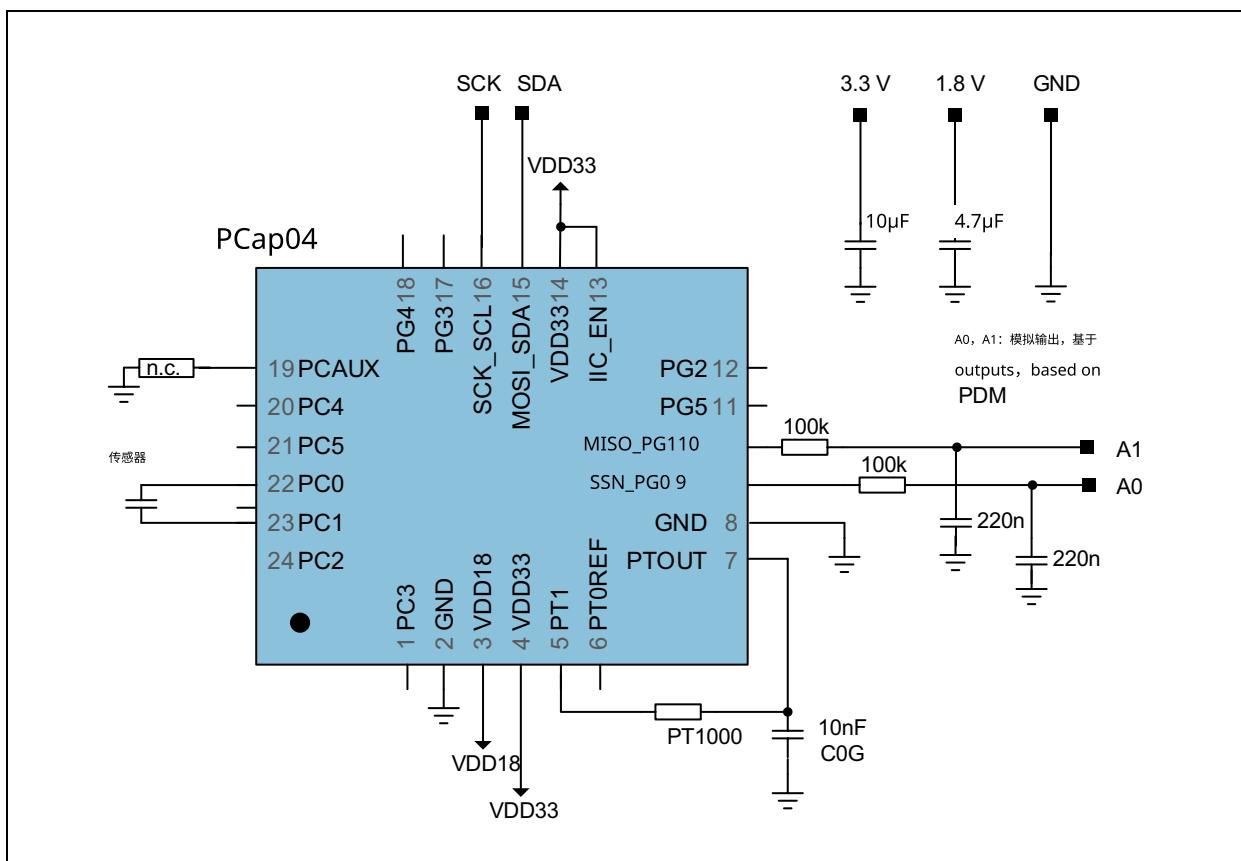
功能:	pcap04的配置寄存器地址和名称。为了方便，包含无指令，因此不会浪费程序空间。
定义 (示例):	CONST CFG_ADD_OFFSET 960 ; 配置在NVRAM中的起始地址; ----- 寄存器13CONST CFG_ADD_C_TRIG_SEL 13 + CFG_ADD_OFFSET...; ----- 寄存器42CONST CFG_ADD_EXTERNAL_FLAGS 42+ CFG_ADD_OFFSETCONST CFG_BM_C_MEDIAN_EN0x01 ; C_MEDIAN_EN的位掩码；启用CDC值的中值滤波器...

## 应用信息

### 原理图

PCap04 仅需少量外部元件即可工作。重要的是对电源电压进行充分缓冲。我们建议为VDD33使用  $10\mu F$ ，为VDD18使用  $4.7\mu F$ 。可以使用简单的RC网络将PDM输出整合以生成模拟输出信号。

图135：  
带有 I<sup>2</sup>C 接口和PDM模拟输出的典型原理图

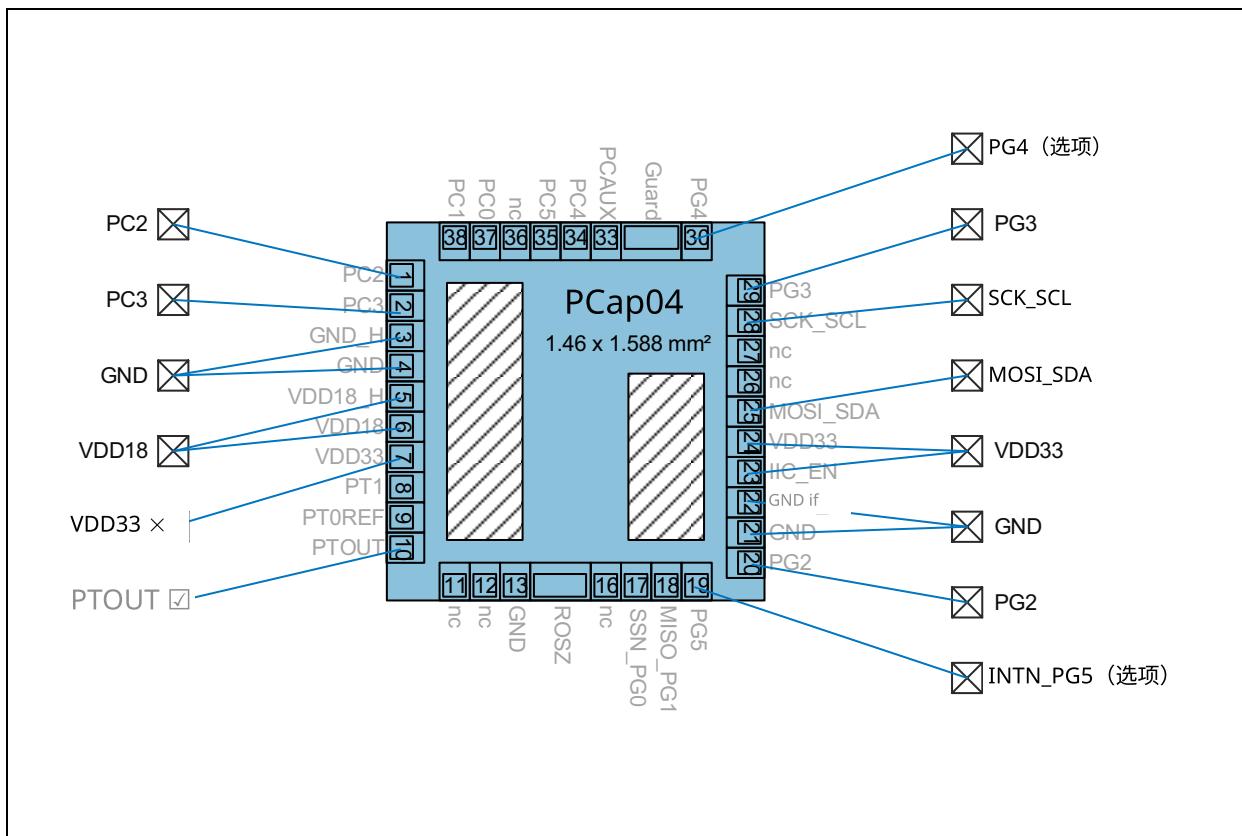


### 最小绑定

PCap04设计用于紧凑型单传感器应用，芯片可仅在两侧进行绑定。

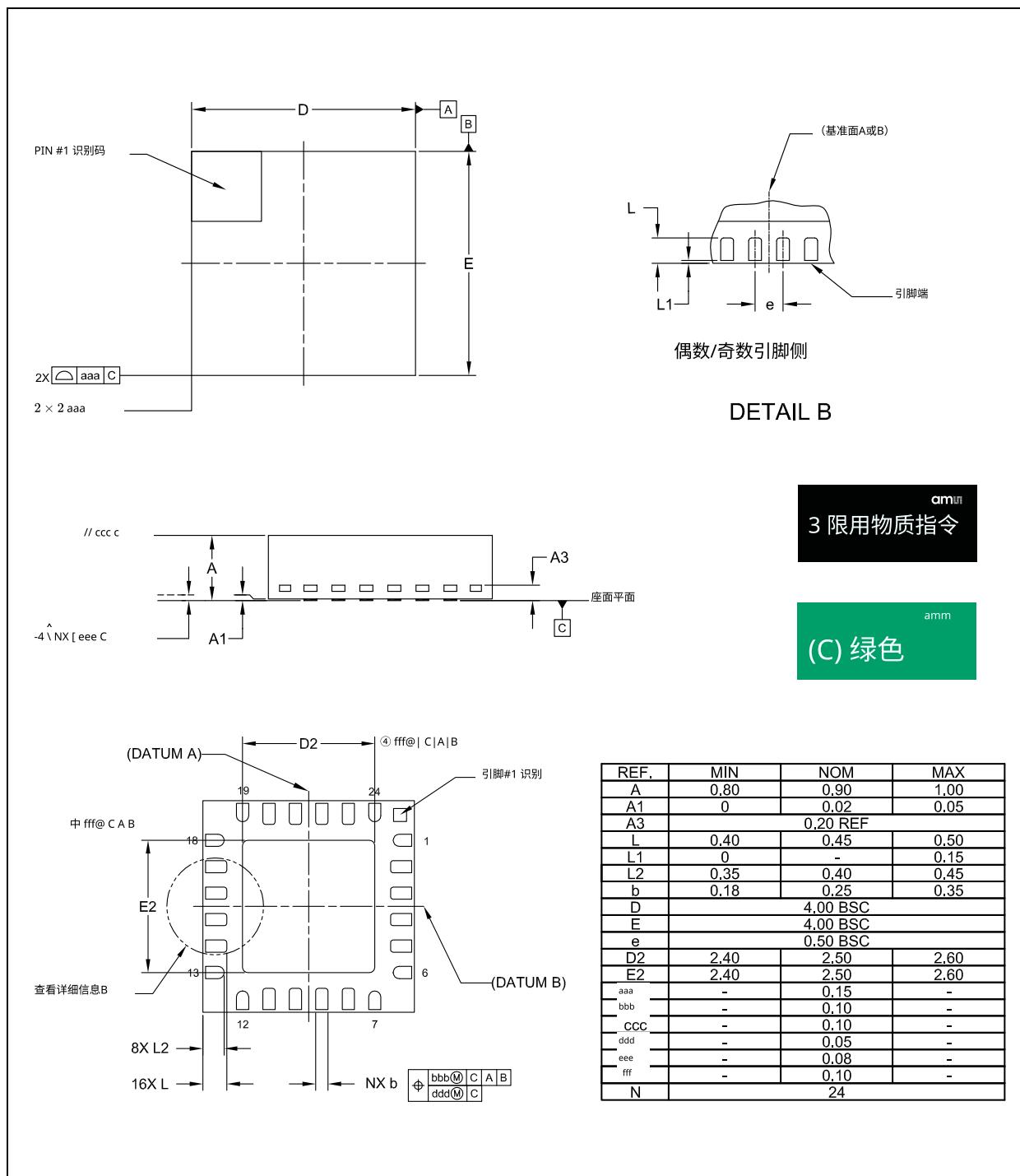
最小连接：GND: #3、4、21、22，VDD33: #7、24，VDD18: #5、6必须连接。

图136：  
双面绑定



## 封装图纸与标记

图137:  
封装图 (QFN24)



### 备注:

- 尺寸和公差符合ASME Y14.5M-1994标准
- 所有尺寸单位为毫米 (角度为度数)
- 端子金属化部分的尺寸b测量点在端子尖端的 0.25 mm 和 0.30 mm 之间。L1尺寸表示从封装边缘到端子完全后部的距离, 至 0.15 mm 止是允许的。
- 共面性适用于暴露的散热片和端子
- 端子上的半径为可选项
- N 为端子总数

图138：  
封装标记

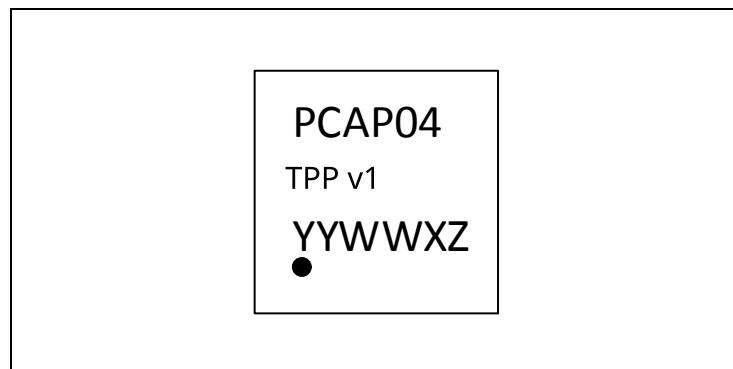


图139：  
封装代码

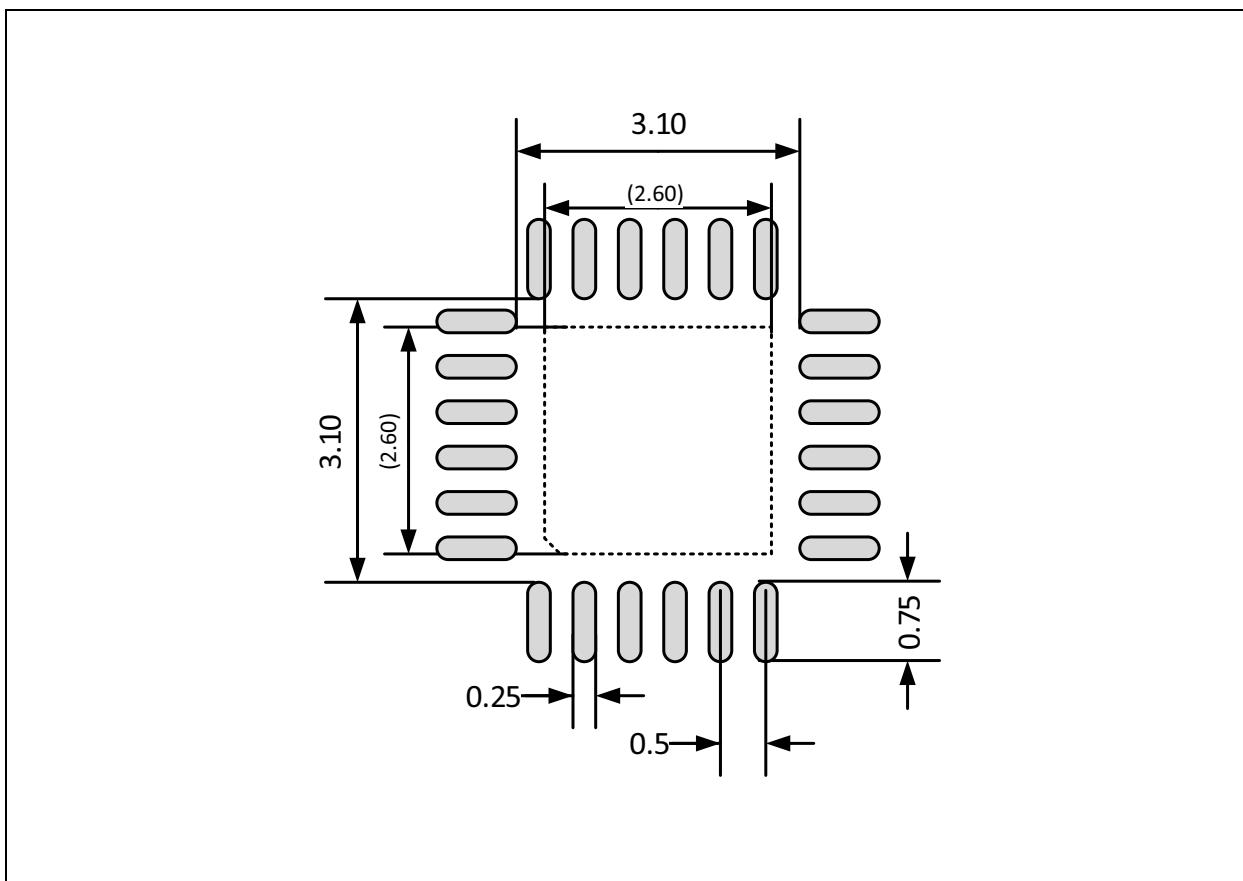
PCAP04	T	PP	v1	YY	WW	X	Z
部分	温度 A: -40°C至125°C B: -40°C至85°C	封装 QF: QFN24	硅片版 本	年份	周数	装配工厂 标识符	装配追溯 码

**PCB焊盘布局**

注意：中心焊盘内部连接到GND。除GND外，不允许在其下方有其他导线。  
建议不要焊接中心焊盘。过多的焊膏可能影响焊接质量。

适用插座：例如Plastronics 32QN50S15050D

图140：  
焊盘布局



注意事项：

1. 所有尺寸单位为 mm。

## 带卷信息

图141:  
封装概述（仅限QFN）

封装	器件	每卷设备数	
		7英寸卷盘	13英寸卷盘
QFN24	PCap04	1000	6000

带卷配置用于从制造商（ams AG）到客户的运输和存储，以及在客户制造工厂的使用。该配置旨在向自动贴片机供料，用于表面贴装在电路板组装中。完整配置包括带有连续单个腔体的载带，用于容纳单个元件，以及封盖带，用于封闭载带以保持元件在腔体中。单个卷盘以干包形式包装，并放入中间箱后发货。

图142:  
卷轴

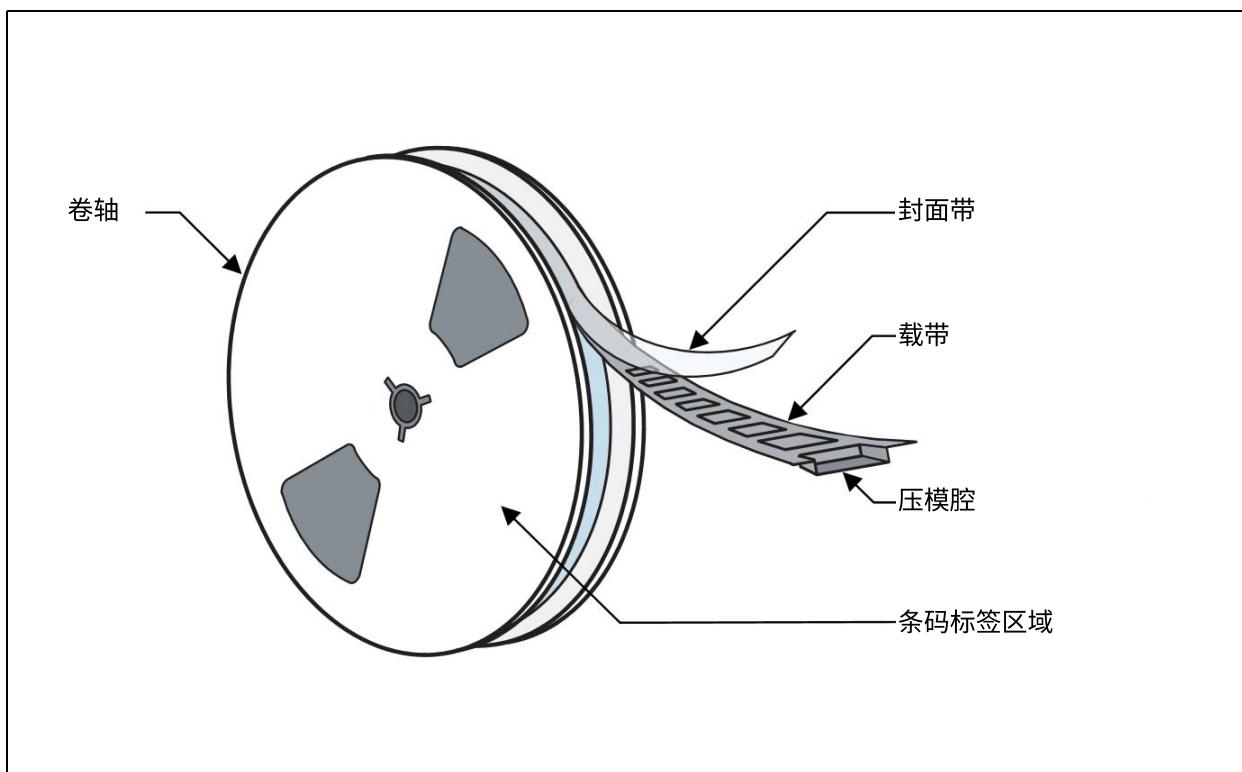
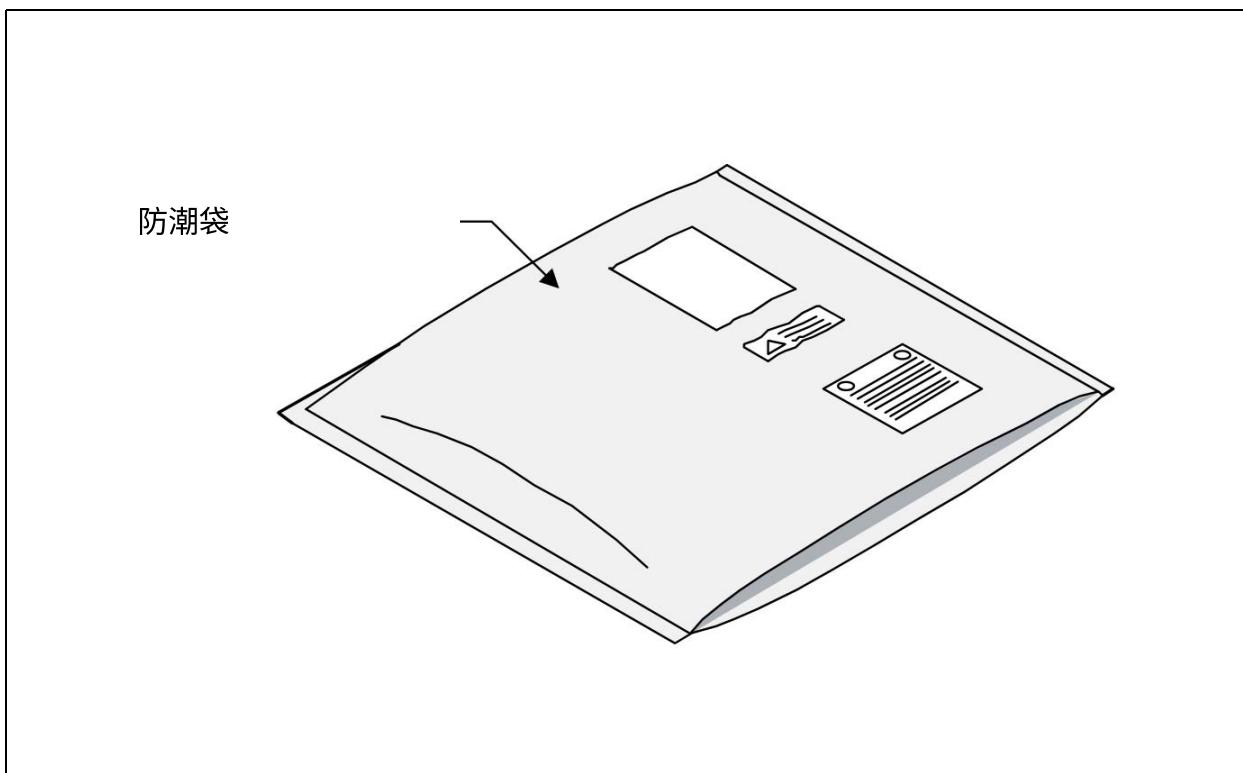
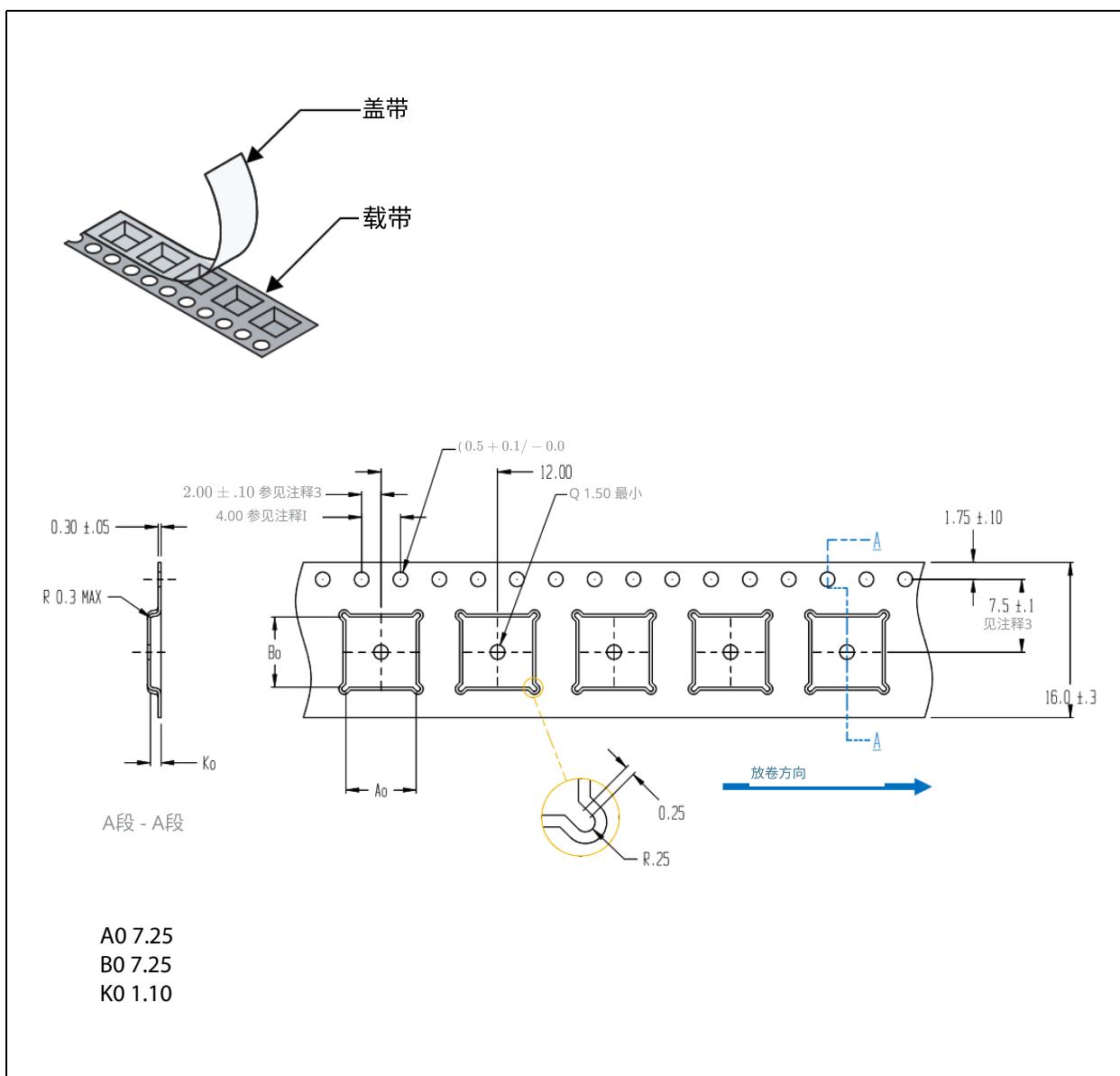


图143：  
干袋中的单卷带



载带广泛用于向点胶机和贴片机提供器件，以实现自动贴装到印刷电路板上。

图144:  
载带QFN24 (32)



注释:

1. 链轮孔距累计公差
2. 符合EIA 481的倾斜度
3. 孔位相对于链轮孔的测量为口袋的真实位置, 而非孔洞
4. 所有尺寸单位为毫米

焊接与存储  
信息

IPC/JEDEC J-STD-020

回流峰值焊接温度（本体温度）根据IPC/JEDEC J-STD-020“非密封固态表面贴装器件的湿气/回流敏感性分类”规定。无铅引线封装的引线镀层为“哑光锡”(100%锡)

## 订购与联系信息

图145：

订购代码	包装	标记	交付形式	交付数量
PCap04-AQFT-24	QFN24	PCAP04 AQF V1 YYWWXZZ	13英寸带子与卷 绕干包	6000个/卷
PCap04-AQFM-24			7英寸带子和干 包卷装	1000个/卷
PCap04-BQFT-24	QFN24	PCAP04 BQF V1 YYWWXZZ	13英寸带卷与卷 绕干包	6000个/卷
PCap04-BQFM-24			7英寸带卷与卷 绕干包	1000个/卷
PCap04-ASWB	n.a.	n.a.	盒中排序的晶圆	1片晶圆 = ~ 10000 芯片
PCap04-ASDF	n.a.	n.a.	蓝色箔上的排序晶圆	1片晶圆 ≈ 10000 个芯片
PCap04-BSWB	n.a.	n.a.	已排序的晶圆盒	1片晶圆 = ~ 10000 芯片

订购信息：在线购买我们的产品或获取免费样品：[www.ams.com/ICdirect](http://www.ams.com/ICdirect)技术支持：[www.ams.com/Technical-Support](http://www.ams.com/Technical-Support)对本文件的反馈：

[www.ams.com/Document-Feedback](http://www.ams.com/Document-Feedback)如需更多信息和请求，请发送电子邮件至：[ams\\_sales@ams.com](mailto:ams_sales@ams.com)销售办事处、经销商和代表请访问：[www.ams.com/contact](http://www.ams.com/contact)

## 总部

ams AG  
Tobelbader Strasse 30  
8141 Premstaetten

## 奥地利，欧洲

电话：+43 (0) 3136 500 0  
网站：[www.ams.com](http://www.ams.com)

## 符合RoHS标准 & ams绿色声明

RoHS：符合RoHS的产品意味着ams AG的产品完全符合当前的RoHS指令。我们的半导体产品不含所有六类化学物质，包括要求在均质材料中铅含量不超过0.1%的规定。对于设计用于高温焊接的产品，符合RoHS的产品适用于特定的无铅工艺。

ams绿色（符合RoHS且不含Sb/Br）：ams绿色定义除了符合RoHS外，我们的产品还不含溴（Br）和锑（Sb）基阻燃剂（在均质材料中Br或Sb不超过0.1%）。

重要信息：本声明中提供的信息代表ams AG截至提供之日的知识和信念。ams AG的知识和信念基于第三方提供的信息，并不对该等信息的准确性作出任何陈述或保证。正在努力更好地整合来自第三方的信息。ams AG已采取并持续采取合理措施以提供具有代表性和准确性的信息，但可能未对输入的材料和化学品进行破坏性测试或化学分析。ams AG及其供应商认为某些信息属于专有，因此CAS编号及其他有限信息可能无法公开披露。

## 版权与免责声明

版权归ams AG所有，地址：奥地利欧洲Premstaetten Tobelbader Strasse 30, 8141。商标已注册。保留所有权利。未经版权所有者事先书面同意，不得复制、改编、合并、翻译、存储或使用本文资料。

ams AG销售的设备受其通用贸易条款中的保修和专利赔偿条款保护。ams AG对本文所列信息不作任何明示、法定、暗示或描述性保证。ams AG保留随时更改规格和价格的权利，恕不另行通知。在将本产品集成到系统中之前，需与ams AG确认最新信息。本产品仅用于商业用途。对于需要扩展温度范围、特殊环境条件或高可靠性应用（如军事、医疗生命支持或生命维持设备），未经ams AG额外处理，不建议使用。本产品按“现状”提供，否认任何明示或暗示的保证，包括但不限于对适销性和特定用途适用性的暗示保证。

ams AG 对接收方或任何第三方因提供、执行或使用本技术数据而引起的任何损害，包括但不限于人身伤害、财产损失、利润损失、使用损失、业务中断或间接、特殊、偶发或后果性损害，概不负责。ams AG 提供技术或其他服务不构成对接收方或任何第三方的义务或责任。

## 文档状态

文档状态	产品状态	定义
产品预览	开发前	本数据表中的信息基于产品在开发规划阶段的构思。所有规格均为设计目标，未提供任何保证，可能会在不通知的情况下更改
初步数据表	预生产	本数据表中的信息基于处于设计、验证或资格认证阶段的产品。本文档中显示的性能和参数为初步，未提供任何保证，可能会在不通知的情况下更改
数据表	生产	本数据表中的信息基于逐步投产或已全面投产的产品，符合ams AG标准保修条款中的规格要求，详见通用贸易条款
数据表（已停产）	已停产	本数据表中的信息基于符合ams AG标准保修条款中规格要求的产品，但这些产品已被取代，不应在新设计中使用

## 修订信息

从1-02（2017年11月8日）到当前修订版1-03（2018年4月4日）的变更	页
更新图1	1
更新图8	10
图10下的更新说明	13
图77下的更新内容	53
更新的图78	54
图80下的更新内容	56
图89下的更新内容	62
触发器下的更新内容	66
更新的NVRAM访问内容	94

注意事项：

1. 上一版本的页码和图号可能与当前版本不同。
2. 未明确提及更正排版错误。

## 内容指南 1 一般描述

- 1 主要优点与特性
- 2 MEMS传感器应用
- 2 方框图

## 3 引脚分配

- 4脚描述
- 6焊盘坐标

## 8极限额定值

## 10电气特性

- 13CDC特性
- 14RDC特性

## 16 时序特性

## 17 详细描述

## 19 寄存器描述

## 19 配置寄存器

## 19 寄存器概述

## 4 详细配置寄存器描述

配置寄存器0 (地址0x0)

配置寄存器1 (地址0x1)

配置寄存器2 (地址0x2)

配置寄存器3 (地址0x3)

配置寄存器4 (地址0x4)

配置寄存器5 (地址0x5)

27 配置寄存器6 (地址0x6)

27 配置寄存器7:8 (地址0x7, 0x8)

27 配置寄存器11:9 (地址0x9;0xA;0xB)

28 配置寄存器12 (地址0xC)

28 配置寄存器13 (地址0xD)

28 配置寄存器14 (地址0xE)

29 配置寄存器 15 (地址 0xF)

29 配置寄存器 16 (地址 0x10)

29 配置寄存器 17 (地址 0x11)

30 配置寄存器 18 (地址 0x12)

31 配置寄存器 19 (地址 0x13)

31 配置寄存器 20 (地址 0x14)

32 配置寄存器 21 (地址 0x15)

32 配置寄存器 22 (地址 0x16)

33 配置寄存器 23 (地址 0x17)

33 配置寄存器 24 (地址 0x18)

34 配置寄存器 25 (地址 0x19)

34 配置寄存器 26 (地址 0x1A)

34 配置寄存器 27 (地址 0x1B)

35 配置寄存器 28 (地址 0x1C)

35 配置寄存器 29 (地址 0x1D)

35 配置寄存器 30 (地址 0x1E)

36 配置寄存器 31 (地址 0x1F)

37 配置寄存器 32 (地址 0x20)

37 配置寄存器33 (地址0x21)

38 配置寄存器34 (地址0x22)

38 配置寄存器 35 (地址 0x23)  
38 配置寄存器 36 (地址 0x24)  
39 配置寄存器 37 (地址 0x25)  
39 配置寄存器 38 (地址 0x26)  
39 配置寄存器 39 (地址 0x27)  
39 配置寄存器 40 (地址 0x28)  
40 配置寄存器 41 (地址 0x29)  
40 配置寄存器 42 (地址 0x30)  
40 配置寄存器 47 (地址 0x2F)  
41 配置寄存器 48 (地址 0x30)  
41 配置寄存器 49 (地址 0x31)  
41 配置寄存器 50 (地址 0x32)  
42 配置寄存器 51 至 53: ams 内部寄存器, 0x00 为强制  
42 配置寄存器 54 (地址 0x36)  
42 配置寄存器 55 至 61: ams 内部寄存器, 0x00 为强制  
42 配置寄存器 62 (地址 0x3e) 42 配置寄存器 63 (地址 0x3f) 43 读寄存器 45 结果寄存器 45 状态寄存器 47 工作原理 47 转换器前端

47 电容数字转换器 (CDC)  
47 测量原理  
47 连接传感器  
放电电阻器  
循环  
序列  
转换  
CDC 补偿选项  
内部补偿  
55 外部补偿  
56 DCBalance  
57 增益校正  
57 CDC 重要参数  
57 循环时钟  
58 循环时间  
59 序列  
60 转换  
61 保护  
64 RDC 数字转换器  
64 测量原理  
64 连接传感器  
65 循环与转换  
66 触发  
67 RDC 重要参数  
67 循环时钟  
68 序列  
68 转换  
68 RDC 结果, 比例  
69 接口 (串行和 PDM/PWM)

69 串行接口 (SIF)  
70 操作码  
71 I<sup>2</sup>C 兼容接口  
72 I<sup>2</sup>C 时序  
72 IPC 写入  
73 I<sup>2</sup>C 读取  
73 SPI接口  
74 SPI时序  
76 GPIO与PDM/PWM  
78 去抖滤波器  
78 PDM与PWM  
82 振荡器  
83 数字信号处理器与存储器  
84 数字信号处理器与环境  
85 RAM结构  
87 寄存器0至95, 用户RAM  
88 寄存器96, 标志位与内部控制信号  
88 数字信号处理器读取寄存器97  
89 DSP 读取寄存器 98 至 103  
89 DSP 读写寄存器 105 至 108, 数据指针  
90 DSP 读取寄存器 126, PORTINFO  
(PORTERR<7...0>, PORTMASK<7...0>)  
90 DSP 写入寄存器 97 至 104, RES00...RES07  
90 DSP 写入寄存器 109, 110, PIO\_REF...PI1\_REF  
90 DSP 写入寄存器 111, TIMERO  
91 DSP 读/写寄存器 112 至 120, MEM\_CTRL,  
MEM\_ADD, MEM\_DATA  
92 DSP 读寄存器 121 至 124, TDC\_START, TDC\_STOP,  
C\_ADD\_PTR, R\_ADD\_PTR  
93 NVRAM 和 ROM  
93 NVRAM 结构  
94 NVRAM 访问  
94 ROM 结构  
94 DSP 输入与输出  
99 ALU 标志  
100 DSPOUT – GPIO 分配  
104 DSP 配置  
105 DSP 启动  
105 看门狗  
106 指令集  
107 指令  
125 指令详情  
125 指针  
126 具有推送和弹出功能的传送常数  
126 多重  
127 常数的推入与弹出交接  
127 div  
130 ROM例程  
132 \_ROM\_Version\_  
132 \_ROM\_tdc\_dispatch\_  
133 \_ROM\_cdc\_cycle\_  
134 \_ROM\_Rdc\_cycle\_  
135 \_ROM\_cdc\_initialize\_  
135 \_ROM\_rdc\_initialize\_

- 136 shiftL\_A\_xx; shiftR\_A\_xx; shiftL\_B\_xx; shiftR\_B\_xx
- 136 mult\_xx
- 137 div\_xx
- 138 \_ROM\_div\_variable\_
- 139 \_ROM\_mult\_variable\_
- 140 \_ROM\_shift\_a\_variable\_
- 140 \_ROM\_shift\_b\_variable\_
- 141 \_ROM\_dma\_
- 142 \_ROM\_In\_, \_ROM\_log10\_, \_ROM\_Id\_
- 142 \_ROM\_signed24\_to\_signed32\_
- 143 \_ROM\_median\_
- 144 \_ROM\_cdc\_
- 145 \_ROM\_rdc\_:\_ROM\_rdc\_inverse\_
- 146 \_ROM\_存储器 (读写易失性存储器)
- 148 \_ROM\_存储器调用/存储
- 149 \_ROM\_二维校准
- 150 \_ROM\_三次多项式
- 151 \_ROM\_四次多项式
- 152 \_ROM\_pulse\_
- 153 \_ROM\_pulse\_loaded\_cal\_vals
- 154 \_ROM\_NVblock\_copy\_
- 155 \_ROM\_capacitance\_polynomial
- 156 \_ROM\_capacitance\_polynomial\_4d
- 157 汇编程序
- 158 指令**
- 160 示例代码**
- 160 “for” 循环
- 160 “while” 循环
- 160 “do - while” 循环
- 161 使用两个指针的“do - while”
- 161 右旋转A到B
- 162 库**
- 163 pcap\_standard.h
- 163 PCap04\_ROM\_addresses\_standard.h
- 164 pcap\_config.h
  
- 165 应用信息**
- 165 原理图**
- 166 最小化绑定**
  
- 167 封装图纸与标记**
- 169 PCB焊盘布局
- 170 带卷信息**
- 173 焊接与存储信息
- 174 订购与联系信息
- 175 符合RoHS及ams绿色声明
- 176 版权与免责声明
- 177 文档状态
- 178 版本信息