第 22 章 12 位高速逐次逼近寄存器 （SAR） 模数转换器 （ADC）

本章包括下列主题：

22.1 简介 22-2
22.2 控制寄存器 22-6
22.3 ADC 工作原理 22-61
22.4 ADC 模块配置 22-65
22.5 其他 ADC 功能 22-86
22.6 中断 22-108
22.7 节能模式期间的操作 22-114
22.8 复位的影响 22-116
22.9 传递函数 22-116
22.10 ADC 采样要求 22-117
22.11 连接注意事项 22-117
22.12 相关应用笔记 22-118
22.13 版本历史 22-119

> **注：** 本系列参考手册章节旨在用作对器件数据手册的补充。根据不同的器件，本手册章节可能并不适用于所有 PIC32 器件。
>
> 请参见当前器件数据手册中 **“ADC”** 章节开头部分的注，以检查本文档是否支持您所使用的器件。
>
> 器件数据手册和系列参考手册章节可从Microchip网站下载：http://www.microchip.com。

22.1 简介

PIC32 12 位高速逐次逼近寄存器（Analog-to-Digital Converter，SAR）模数转换器（Successive Approximation Register， ADC）模块包括以下特性：

- 12 位分辨率
- 最多 8 个带有专用采样 / 保持 （S&H）电路的 ADC 模块
- 两个专用 ADC 模块可以在 Turbo 模式下组合使用，从而提供双倍转换速率
- 单端和 / 或差分输入
- 能在休眠模式下工作
- 支持触摸传感应用
- 最多 6 个数字比较器
- 最多 6 个数字滤波器，支持两种模式：
  - 过采样模式
  - 平均模式
- 用于专用 ADC 模块的 FIFO 和 DMA 引擎 （见注）
- 提前中断产生，可更快速地处理转换数据
- 设计用于电机控制、电源转换和通用应用

> **注：** 该特性并非在所有器件上都可用。要确定可用性，请参见具体器件数据手册的 **“ADC”** 章节。

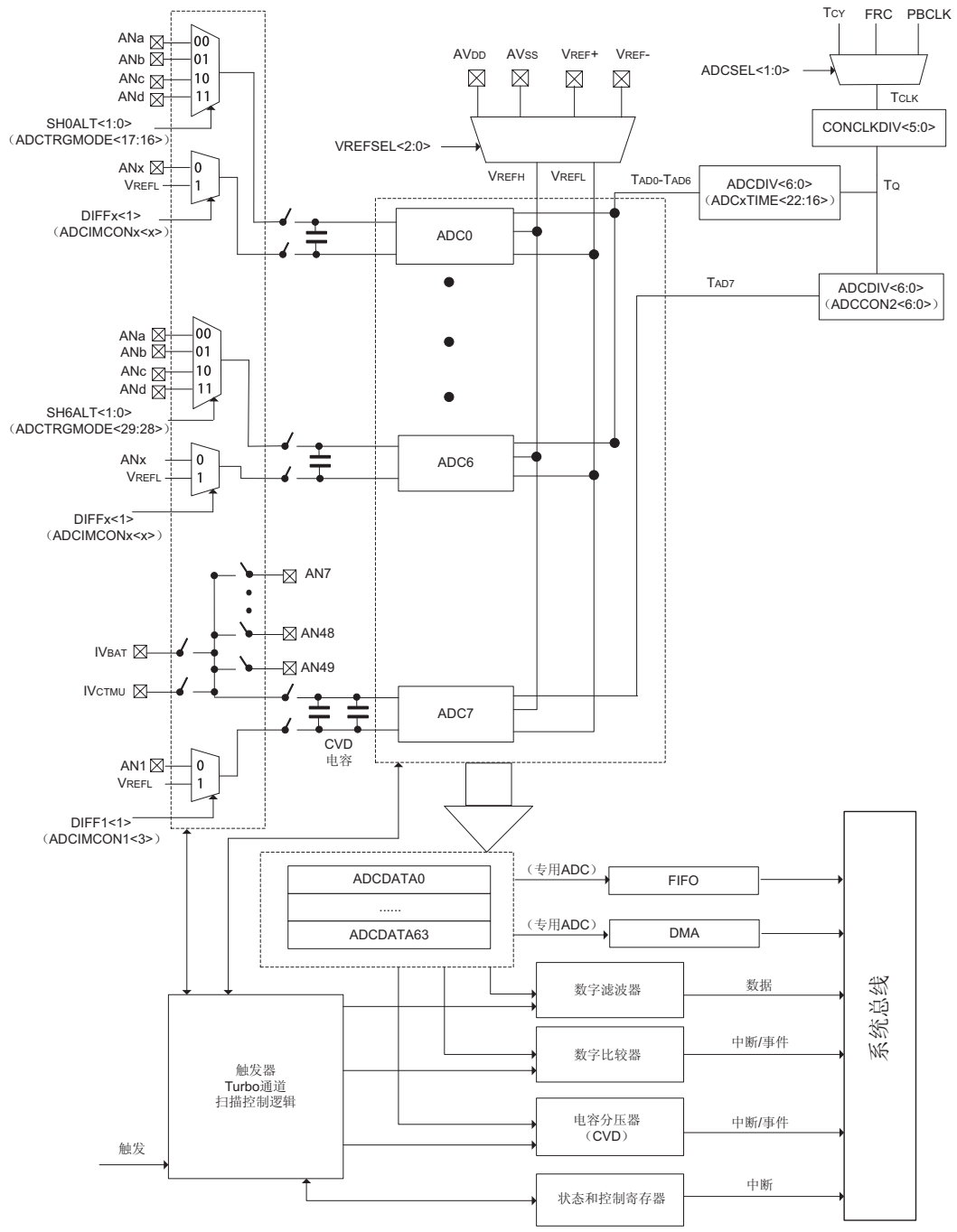
图 22-1 给出了 ADC 模块的简化框图。

根据不同的器件，ADC 最多具有 7 个专用 ADC 模块和 1 个共用 ADC 模块。专用 ADC 使用单个输入 （或其备用输入），它用于对时间敏感或瞬态输入进行高速的精确采样，而共用 ADC 模块在输入上具有一个多路开关，便于连接较大的一组输入 （采样速率较低），并通过输入扫描逻辑提供灵活的自动扫描选项。

对于每个 ADC 模块，模拟输入会被连接到 S&H 电容。每个 ADC 模块的时钟、采样时间和输出数据分辨率可以独立设置。ADC 模块会基于在寄存器中设置的配置对输入模拟信号执行转换。转换完成时，最终结果会被存储到特定模拟输入的结果缓冲区中，并且如果数字滤波器和数字比较器配置为使用来自该特定采样的数据，则会被传送到数字滤波器和数字比较器。

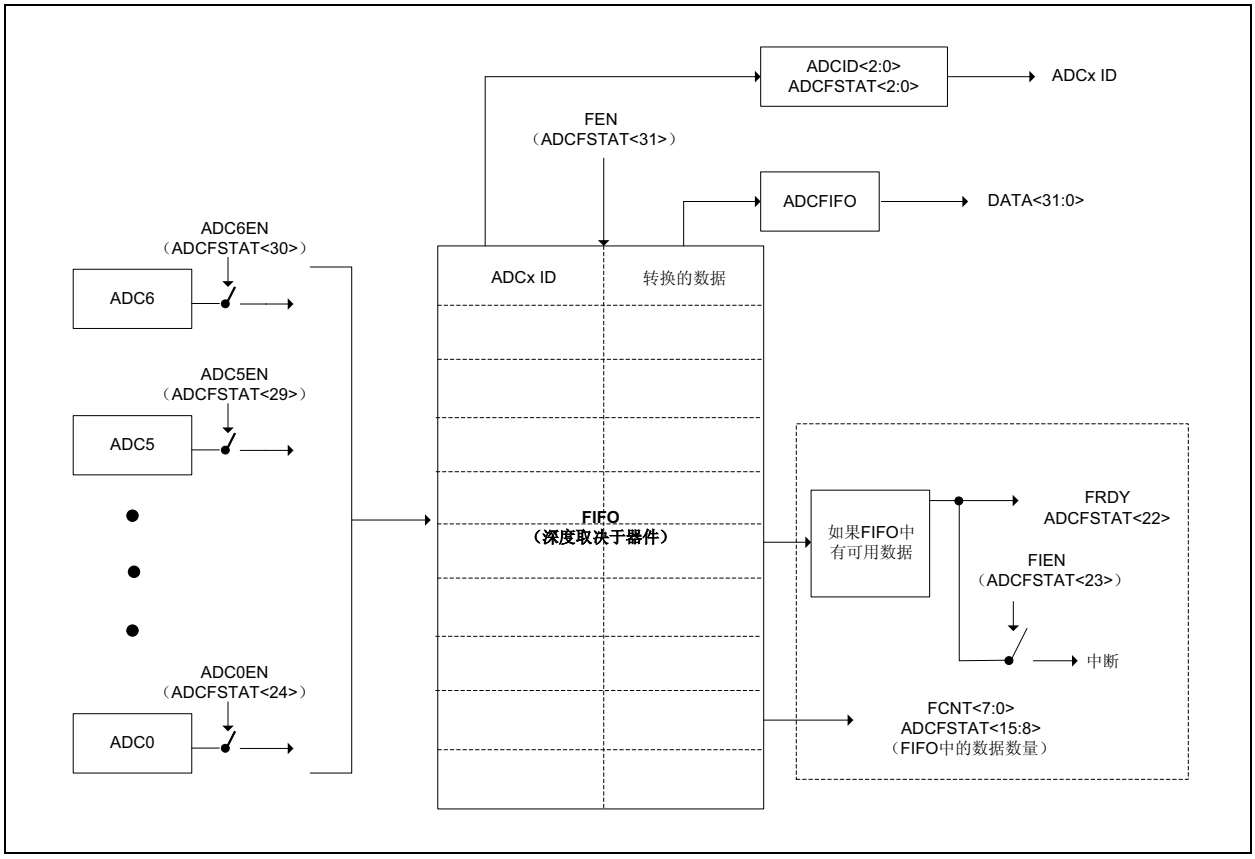
 © 2015 Microchip Technology Inc.

图 22-1： ADC 框图



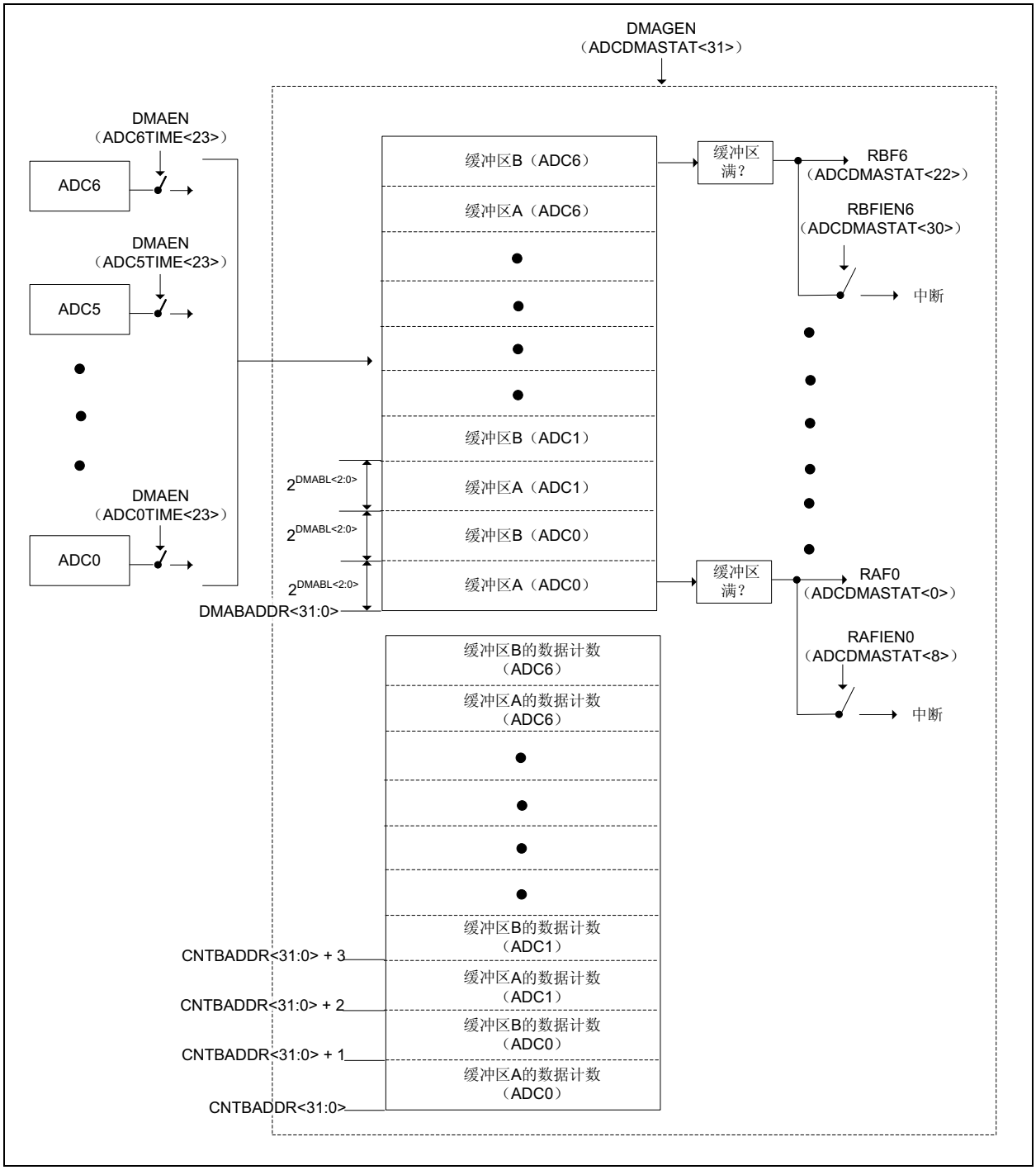
注： 模拟输入 ANa、ANb、ANc 和 ANd 以及 FIFO 和 DMA 功能作为示例给出。关于实际的 ANx 选择以及 FIFO 和 DMA 功能的可用性，请参见具体器件数据手册。

图 22-2： FIFO 框图



 © 2015 Microchip Technology Inc.

图 22-3： DMA 框图


DMAGEN
（ADCDMASTAT<31>）
DMAEN
（ADC6TIME<23>）
ADC6
DMAEN
（ADC5TIME<23>）
ADC5
DMAEN
（ADC0TIME<23>）
ADC0
缓冲区B（ADC6）
缓冲区A（ADC6）
缓冲区B（ADC1）
缓冲区A（ADC1）
缓冲区B（ADC0）
缓冲区A（ADC0）
2DMABL<2:0>
2DMABL<2:0>
2DMABL<2:0>
DMABADDR<31:0>
缓冲区满？
缓冲区满？
RBF6
（ADCDMASTAT<22>）
RBFIEN6
（ADCDMASTAT<30>）
中断
RAF0
（ADCDMASTAT<0>）
RAFIEN0
（ADCDMASTAT<8>）
中断
缓冲区B的数据计数
（ADC6）
缓冲区A的数据计数
（ADC6）
缓冲区B的数据计数
（ADC1）
缓冲区A的数据计数
（ADC1）
缓冲区B的数据计数
（ADC0）
缓冲区A的数据计数
（ADC0）
CNTBADDR<31:0> + 3
CNTBADDR<31:0> + 2
CNTBADDR<31:0> + 1
CNTBADDR<31:0>

22.2 控制寄存器

PIC32 12 位高速 SAR ADC 模块具有以下特殊功能寄存器 （SFR）：

- **ADCCON1：ADC 控制寄存器 1**
  该寄存器控制所有 ADC 模块的基本操作，包括休眠和空闲模式下的行为以及数据格式。该寄存器还用于指定中断控制器的向量移位量。其他 ADCCON1 功能包括控制 ADC 的 Turbo 功能、 DMA 模式下的 RAM 缓冲区长度，以及电容分压 （CVD）。
- **ADCCON2：ADC 控制寄存器 2**
  该寄存器控制所有 ADC 模块的参考电压选择、共用 ADC 模块 （ADC7）的采样时间、参考电压中断允许、提前中断选择和共用 ADC （ADC7）的时钟分频选择。
- **ADCCON3：ADC 控制寄存器 3**
  该寄存器用于使能 ADC 时钟选择、使能 / 禁止专用和共用 ADC 模块的数字功能，以及控制手动 （软件）采样和转换。
- **ADCTRGMODE：专用 ADC 的 ADC 触发模式寄存器**
  该寄存器用于选择备用模拟输入，并包含专用 ADC 模块 （ADC0-ADC6）的触发设置。
- **ADCIMCON1：ADC 输入模式控制寄存器 1** 至
  ADCIMCON4：ADC 输入模式控制寄存器 4
  通过这些寄存器，用户可以在单端和差分操作之间进行选择，以及在有符号和无符号数据格式之间进行选择。
- **ADCGIRQEN1：ADC 全局中断允许寄存器 1** 和
  ADCGIRQEN2：ADC 全局中断允许寄存器 2
  这些寄存器用于指定哪些独立输入转换中断可以产生全局 ADC 中断。
- **ADCCSS1：ADC 公共扫描选择寄存器 1** 和
  ADCCSS2：ADC 公共扫描选择寄存器 2
  这些寄存器用于指定由公共扫描触发扫描的模拟输入。
- **ADCDSTAT1：ADC 数据就绪状态寄存器 1** 和
  ADCDSTAT2：ADC 数据就绪状态寄存器 2
  这些寄存器包含各个模拟输入转换的中断状态。每个位代表其关联的转换结果的数据就绪状态。
- **ADCCMPENx：ADC 数字比较器 x 使能寄存器 （x = 1 至 6）**
  这些寄存器用于选择哪些模拟输入转换结果将由数字比较器处理。
- **ADCCMPx：ADC 数字比较器 x 限值寄存器 （x = 1 至 6）**
  这些寄存器包含供数字比较器使用的上限和下限数字比较值。
- **ADCFLTRx：ADC 数字滤波器 x 寄存器 （x = 1 至 6）**
  这些寄存器提供过采样滤波器累加器的控制和状态位，并且还包含 16 位滤波器输出数据。
- **ADCTRG1：ADC 触发源 1 寄存器**
  该寄存器控制 AN0 至 AN3 模拟输入的触发源选择。
- **ADCTRG2：ADC 触发源 2 寄存器**
  该寄存器控制 AN4 至 AN7 模拟输入的触发源选择。
- **ADCTRG3：ADC 触发源 3 寄存器**
  该寄存器控制 AN8 至 AN11 模拟输入的触发源选择。
- **ADCTRG4：ADC 触发源 4 寄存器**
  该寄存器控制 AN12 至 AN15 模拟输入的触发源选择。
- **ADCTRG5：ADC 触发源 5 寄存器**
  该寄存器控制 AN16 至 AN19 模拟输入的触发源选择。

 © 2015 Microchip Technology Inc.

- **ADCTRG6：ADC 触发源 6 寄存器**
  该寄存器控制 AN20 至 AN23 模拟输入的触发源选择。
- **ADCTRG7：ADC 触发源 7 寄存器**
  该寄存器控制 AN24 至 AN27 模拟输入的触发源选择。
- **ADCTRG8：ADC 触发源 8 寄存器**
  该寄存器控制 AN28 至 AN31 模拟输入的触发源选择。
- **ADCCMPCON1：ADC 数字比较器 1 控制寄存器**
  该寄存器控制数字比较器 1 的操作，包括中断产生、要使用的比较条件，以及提供发生比较器事件时的状态。此外，该寄存器还提供 CVD 的输出数据。
- **ADCCMPCONx：ADC 数字比较器 x 控制寄存器 （x = 2 至 6）**
  这些寄存器控制数字比较器 2 至 6 的操作，包括中断产生和要使用的比较条件。该寄存器还可提供发生比较器事件时的状态。
- **ADCFSTAT：ADC FIFO 状态寄存器**
  该寄存器用于指定专用 ADC 模块 FIFO 的状态。
- **ADCFIFO：ADC FIFO 数据寄存器**
  该寄存器用于指定专用 ADC 模块 FIFO 的输出值。
- **ADCBASE：ADC 基址寄存器**
  这些寄存器用于指定用户 ADC 中断服务程序 （ISR）跳转表的基址。
- **ADCDMASTAT：ADC DMA 状态寄存器**
  该寄存器包含 DMA 状态位。
- **ADCCNTB：ADC 采样计数基址寄存器**
  该寄存器包含 RAM 中的采样计数的基址。除了在 RAM 中存储每个专用 ADC 模块的转换数据之外， DMA 还会存储转换采样计数。
- **ADCDMAB：ADC DMA 基址寄存器**
  该寄存器包含 DMA 引擎的 RAM 基址。
- **ADCTRGSNS：ADC 触发电平 / 边沿敏感寄存器**
  该寄存器包含每个 ADC 模拟输入的触发电平设置。
- **ADCxTIME：专用 ADCx 时序寄存器 x （x = 0 至 6）**
  这些寄存器包含专用模拟输入的时间和时钟设置。
- **ADCEIEN1：ADC 提前中断允许寄存器 1** 和
  ADCEIEN2：ADC 提前中断允许寄存器 2
  这些寄存器包含用于允许或禁止各个模拟输入的提前中断的位。
- **ADCEISTAT1：ADC 提前中断状态寄存器 1** 和
  ADCEISTAT2：ADC 提前中断状态寄存器 2
  这些寄存器包含各个模拟输入的提前中断的状态位。
- **ADCANCON：ADC 模拟预热控制寄存器**
  该寄存器包含 ADC 模块的模拟和偏置电路的预热控制设置。
- **ADCDATAx：ADC 输出数据寄存器 （x = 0 至 63）**
  这些寄存器是模数转换输出数据寄存器。 ADCDATAx 寄存器与每个模拟输入 0-63 关联。
- **ADCxCFG：ADCx 配置寄存器 x （x = 0 至 7）**
  这些寄存器用于指定 ADC 模块配置数据。
- **ADCSYSCFG0：ADC 系统配置寄存器 0** 和
  ADCSYSCFG1：ADC 系统配置寄存器 1
  这些寄存器包含对应于模拟输入的只读位。

表 22-1 汇总了所有 ADC 特殊功能寄存器 （SFR）。该汇总表之后列出了相应的寄存器，其中包含了每个位的详细说明。

表 22-1： ADC SFR 汇总

寄存器名称	位范围	Bit 31/15	Bit 30/14	Bit 29/13	Bit 28/12	Bit 27/11	Bit 26/10	Bit 25/9	Bit 24/8	Bit 23/7	Bit 22/6	Bit 21/5	Bit 20/4	Bit 19/3	Bit 18/2	Bit 17/1	Bit 16/0
ADCCON1	31:16	TRBEN	TRBERR	TRBMST<2:0>			TRBSLV<2:0>			FRACT	SELRES<1:0>		STRGSRC<4:0>				
	15:0	ON	—	SIDL	AICPMPEN	CVDEN	FSSCLKEN	FSPBCLKEN	—	—	IRQVS<2:0>			STRGLVL	DMABL<2:0>		
ADCCON2	31:16	BGVRRDY	REFFLT	EOSRDY	CVDCPL<2:0>		SAMC<9:0>										
	15:0	BGVRIEN	REFFLTIEN	EOSIEN	ADCEIOVR	ECRIEN	ADCEIS<2:0>			—	ADCDIV<6:0>						
ADCCON3	31:16	ADCSEL<1:0>		CONCLKDIV<5:0>						DIGEN7	DIGEN6	DIGEN5	DIGEN4	DIGEN3	DIGEN2	DIGEN1	DIGEN0
	15:0	VREFSEL<2:0>			TRGSUSP	UPDIEN	UPDRDY	SAMP	RQCNVRT	GLSWTRG	GSWTRG	ADINSEL<5:0>					
ADCTRGMODE	31:16	—	—	SH1ALT<1:0>		SH5ALT<1:0>		SH4ALT<1:0>		SH3ALT<1:0>		SH2ALT<1:0>		SH1ALT<1:0>		SH0ALT<1:0>	
	15:0	—	STRGEN6	STRGEN5	STRGEN4	STRGEN3	STRGEN2	STRGEN1	STRGEN0	—	SSAMPEN6	SSAMPEN5	SSAMPEN4	SSAMPEN3	SSAMPEN2	SSAMPEN1	SSAMPEN0
ADCIMCON1	31:16	DIFF15	SIGN15	DIFF14	SIGN14	DIFF13	SIGN13	DIFF12	SIGN12	DIFF11	SIGN11	DIFF10	SIGN10	DIFF9	SIGN9	DIFF8	SIGN8
	15:0	DIFF7	SIGN7	DIFF6	SIGN6	DIFF5	SIGN5	DIFF4	SIGN4	DIFF3	SIGN3	DIFF2	SIGN2	DIFF1	SIGN1	DIFF0	SIGN0
ADCIMCON2	31:16	DIFF31	SIGN31	DIFF30	SIGN30	DIFF29	SIGN29	DIFF28	SIGN28	DIFF27	SIGN27	DIFF26	SIGN26	DIFF25	SIGN25	DIFF24	SIGN24
	15:0	DIFF23	SIGN23	DIFF22	SIGN22	DIFF21	SIGN21	DIFF20	SIGN20	DIFF19	SIGN19	DIFF18	SIGN18	DIFF17	SIGN17	DIFF16	SIGN16
ADCIMCON3	31:16	DIFF47	SIGN47	DIFF46	SIGN46	DIFF45	SIGN45	DIFF44	SIGN44	DIFF43	SIGN43	DIFF42	SIGN42	DIFF41	SIGN41	DIFF40	SIGN40
	15:0	DIFF39	SIGN39	DIFF38	SIGN38	DIFF37	SIGN37	DIFF36	SIGN36	DIFF35	SIGN35	DIFF34	SIGN34	DIFF33	SIGN33	DIFF32	SIGN32
ADCIMCON4	31:16	DIFF63	SIGN63	DIFF62	SIGN62	DIFF61	SIGN61	DIFF60	SIGN60	DIFF59	SIGN59	DIFF58	SIGN58	DIFF57	SIGN57	DIFF56	SIGN56
	15:0	DIFF55	SIGN55	DIFF54	SIGN54	DIFF53	SIGN53	DIFF52	SIGN52	DIFF51	SIGN51	DIFF50	SIGN50	DIFF49	SIGN49	DIFF48	SIGN48
ADCGIRQEN1	31:16	AGIEN31	AGIEN30	AGIEN29	AGIEN28	AGIEN27	AGIEN26	AGIEN25	AGIEN24	AGIEN23	AGIEN22	AGIEN21	AGIEN20	AGIEN19	AGIEN18	AGIEN17	AGIEN16
	15:0	AGIEN15	AGIEN14	AGIEN13	AGIEN12	AGIEN11	AGIEN10	AGIEN9	AGIEN8	AGIEN7	AGIEN6	AGIEN5	AGIEN4	AGIEN3	AGIEN2	AGIEN1	AGIEN0
ADCGIRQEN2	31:16	AGIEN63	AGIEN62	AGIEN61	AGIEN60	AGIEN59	AGIEN58	AGIEN57	AGIEN56	AGIEN55	AGIEN54	AGIEN53	AGIEN52	AGIEN51	AGIEN50	AGIEN49	AGIEN48
	15:0	AGIEN47	AGIEN46	AGIEN45	AGIEN44	AGIEN43	AGIEN42	AGIEN41	AGIEN40	AGIEN39	AGIEN38	AGIEN37	AGIEN36	AGIEN35	AGIEN34	AGIEN33	AGIEN32
ADCCSS1	31:16	CSS31	CSS30	CSS29	CSS28	CSS27	CSS26	CSS25	CSS24	CSS23	CSS22	CSS21	CSS20	CSS19	CSS18	CSS17	CSS16
	15:0	CSS15	CSS14	CSS13	CSS12	CSS11	CSS10	CSS9	CSS8	CSS7	CSS6	CSS5	CSS4	CSS3	CSS2	CSS1	CSS0
ADCCSS2	31:16	CSS63	CSS62	CSS61	CSS60	CSS59	CSS58	CSS57	CSS56	CSS55	CSS54	CSS53	CSS52	CSS51	CSS50	CSS49	CSS48
	15:0	CSS47	CSS46	CSS45	CSS44	CSS43	CSS42	CSS41	CSS40	CSS39	CSS38	CSS37	CSS36	CSS35	CSS34	CSS33	CSS32
ADCDSTAT1	31:16	ARDY31	ARDY30	ARDY29	ARDY28	ARDY27	ARDY26	ARDY25	ARDY24	ARDY23	ARDY22	ARDY21	ARDY20	ARDY19	ARDY18	ARDY17	ARDY16
	15:0	ARDY15	ARDY14	ARDY13	ARDY12	ARDY11	ARDY10	ARDY9	ARDY8	ARDY7	ARDY6	ARDY5	ARDY4	ARDY3	ARDY2	ARDY1	ARDY0
ADCDSTAT2	31:16	ARDY63	ARDY62	ARDY61	ARDY60	ARDY59	ARDY58	ARDY57	ARDY56	ARDY55	ARDY54	ARDY53	ARDY52	ARDY51	ARDY50	ARDY49	ARDY48
	15:0	ARDY47	ARDY46	ARDY45	ARDY44	ARDY43	ARDY42	ARDY41	ARDY40	ARDY39	ARDY38	ARDY37	ARDY36	ARDY35	ARDY34	ARDY33	ARDY32
ADCCMPENx x = 1-6	31:16	CMPE31	CMPE30	CMPE29	CMPE28	CMPE27	CMPE26	CMPE25	CMPE24	CMPE23	CMPE22	CMPE21	CMPE20	CMPE19	CMPE18	CMPE17	CMPE16
	15:0	CMPE15	CMPE14	CMPE13	CMPE12	CMPE11	CMPE10	CMPE9	CMPE8	CMPE7	CMPE6	CMPE5	CMPE4	CMPE3	CMPE2	CMPE1	CMPE0
ADCCMPx x = 1-6	31:16	DCMPHI<15:0>															
	15:0	DCMPLO<15:0>															
ADCFLTRx x = 1-6	31:16	AFEN	DATA16EN	DFMODE	OVRSAM<2:0>			AFGIEN	AFRDY	—	—	—	CHNLID<4:0>				
	15:0	FLTRDATA<15:0>															
ADCTRG1	31:16	—	—	—	TRGSRC3<4:0>					—	—	—	TRGSRC2<4:0>				
	15:0	—	—	—	TRGSRC1<4:0>					—	—	—	TRGSRC0<4:0>				
ADCTRG2	31:16	—	—	—	TRGSRC7<4:0>					—	—	—	TRGSRC6<4:0>				
	15:0	—	—	—	TRGSRC5<4:0>					—	—	—	TRGSRC4<4:0>				
ADCTRG3	31:16	—	—	—	TRGSRC11<4:0>					—	—	—	TRGSRC10<4:0>				
	15:0	—	—	—	TRGSRC9<4:0>					—	—	—	TRGSRC8<4:0>				
ADCTRG4	31:16	—	—	—	TRGSRC15<4:0>					—	—	—	TRGSRC14<4:0>				
	15:0	—	—	—	TRGSRC13<4:0>					—	—	—	TRGSRC12<4:0>				
ADCTRG5	31:16	—	—	—	TRGSRC19<4:0>					—	—	—	TRGSRC18<4:0>				
	15:0	—	—	—	TRGSRC17<4:0>					—	—	—	TRGSRC16<4:0>				
ADCTRG6	31:16	—	—	—	TRGSRC23<4:0>					—	—	—	TRGSRC22<4:0>				
	15:0	—	—	—	TRGSRC21<4:0>					—	—	—	TRGSRC20<4:0>				
ADCTRG7	31:16	—	—	—	TRGSRC27<4:0>					—	—	—	TRGSRC26<4:0>				
	15:0	—	—	—	TRGSRC25<4:0>					—	—	—	TRGSRC24<4:0>				

初稿

© 2015 Microchip Technology Inc.

表 22-1： ADC SFR 汇总 （续）

寄存器名称	位范围	Bit 31/15	Bit 30/14	Bit 29/13	Bit 28/12	Bit 27/11	Bit 26/10	Bit 25/9	Bit 24/8	Bit 23/7	Bit 22/6	Bit 21/5	Bit 20/4	Bit 19/3	Bit 18/2	Bit 17/1	Bit 16/0
ADCTRG8	31:16	—	—	—	TRGSRC31<4:0>					—	—	—	TRGSRC30<4:0>				
	15:0	—	—	—	TRGSRC29<4:0>					—	—	—	TRGSRC28<4:0>				
ADCCMPCON1	31:16	CVDDATA<15:0>															
	15:0	—	—	AINID<5:0>						ENDCMP	DCMPGIEN	DCMPED	IEBTWN	IEHIHI	IEHILO	IELOHI	IELOLO
ADCCMPCONx x = 2-6	31:16	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
	15:0	—	—	—	AINID<4:0>					ENDCMP	DCMPGIEN	DCMPED	IEBTWN	IEHIHI	IEHILO	IELOHI	IELOLO
ADCFSTAT	31:16	FEN	ADC6EN	ADC5EN	ADC4EN	ADC3EN	ADC2EN	ADC1EN	ADC0EN	FIEN	FRDY	FWROVERR	—	—	—	—	—
	15:0	FCNT<7:0>								FSIGN	—	—	—	—	ADCID<2:0>		
ADCFIFO	31:16	DATA<31:16>															
	15:0	DATA<15:0>															
ADCBASE	31:16	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
	15:0	ADCBASE<15:0>															
ADCDMASTAT	31:16	DMAGEN	RBFIEN6	RBFIEN5	RBFIEN4	RBFIEN3	RBFIEN2	RBFIEN1	RBFIEN0	DMAWROVERR	RBF6	RBF5	RBF4	RBF3	RBF2	RBF1	RBF0
	15:0	DMACNTEN	RAFIEN6	RAFIEN5	RAFIEN4	RAFIEN3	RAFIEN2	RAFIEN1	RAFIEN0	—	RAF6	RAF5	RAF4	RAF3	RAF2	RAF1	RAF0
ADCCNTB	31:16	CNTBADDR<31:16>															
	15:0	CNTBADDR<15:0>															
ADCDMAB	31:16	DMABADDR<31:16>															
	15:0	DMABADDR<15:0>															
ADCTRGSNS	31:16	LVL31	LVL30	LVL29	LVL28	LVL27	LVL26	LVL25	LVL24	LVL23	LVL22	LVL21	LVL20	LVL19	LVL18	LVL17	LVL16
	15:0	LVL15	LVL14	LVL13	LVL12	LVL11	LVL10	LVL9	LVL8	LVL7	LVL6	LVL5	LVL4	LVL3	LVL2	LVL1	LVL0
ADCxTIME x = 0-6	31:16	—	—	—	ADCEIS<2:0>			SELRES<1:0>		DMAEN	ADCDIV<6:0>						
	15:0	—	—	—	—	—	—	SAMC<9:0>									
ADCEIEN1	31:16	EIEN31	EIEN30	EIEN29	EIEN28	EIEN27	EIEN26	EIEN25	EIEN24	EIEN23	EIEN22	EIEN21	EIEN20	EIEN19	EIEN18	EIEN17	EIEN16
	15:0	EIEN15	EIEN14	EIEN13	EIEN12	EIEN11	EIEN10	EIEN9	EIEN8	EIEN7	EIEN6	EIEN5	EIEN4	EIEN3	EIEN2	EIEN1	EIEN0
ADCEIEN2	31:16	EIEN63	EIEN62	EIEN61	EIEN60	EIEN59	EIEN58	EIEN57	EIEN56	EIEN55	EIEN54	EIEN53	EIEN52	EIEN51	EIEN50	EIEN49	EIEN48
	15:0	EIEN47	EIEN46	EIEN45	EIEN44	EIEN43	EIEN42	EIEN41	EIEN40	EIEN39	EIEN38	EIEN37	EIEN36	EIEN35	EIEN34	EIEN33	EIEN32
ADCEISTAT1	31:16	EIRDY31	EIRDY30	EIRDY29	EIRDY28	EIRDY27	EIRDY26	EIRDY25	EIRDY24	EIRDY23	EIRDY22	EIRDY21	EIRDY20	EIRDY19	EIRDY18	EIRDY17	EIRDY16
	15:0	EIRDY15	EIRDY14	EIRDY13	EIRDY12	EIRDY11	EIRDY10	EIRDY9	EIRDY8	EIRDY7	EIRDY6	EIRDY5	EIRDY4	EIRDY3	EIRDY2	EIRDY1	EIRDY0
ADCEISTAT2	31:16	EIRDY63	EIRDY62	EIRDY61	EIRDY60	EIRDY59	EIRDY58	EIRDY57	EIRDY56	EIRDY55	EIRDY54	EIRDY53	EIRDY52	EIRDY51	EIRDY50	EIRDY49	EIRDY48
	15:0	EIRDY47	EIRDY46	EIRDY45	EIRDY44	EIRDY43	EIRDY42	EIRDY41	EIRDY40	EIRDY39	EIRDY38	EIRDY37	EIRDY36	EIRDY35	EIRDY34	EIRDY33	EIRDY32
ADCANCON	31:16	—	—	—	—	WKUPCLKCNT<3:0>				WKIEN7	WKIEN6	WKIEN5	WKIEN4	WKIEN3	WKIEN2	WKIEN1	WKIEN0
	15:0	WKRDY7	WKRDY6	WKRDY5	WKRDY4	WKRDY3	WKRDY2	WKRDY1	WKRDY0	ANEN7	ANEN6	ANEN5	ANEN4	ANEN3	ANEN2	ANEN1	ANEN0
ADCDATAx （x = 0-63）	31:16	DATA<31:16>															
	15:0	DATA<15:0>															
ADCxCFG x = 0-7	31:16	ADCCFG<31:16>															
	15:0	ADCCFG<15:0>															
ADCSYSCFG0	31:16	AN<31:16>															
	15:0	AN<15:0>															
ADCSYSCFG1	31:16	AN<63:48>															
	15:0	AN<47:32>															

© 2015 Microchip Technology Inc.

初稿

寄存器 22-1： ADCCON1：ADC 控制寄存器 1

位范围	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	R/W-0	R-0, HS, HC	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	TRBEN	TRBERR	TRBMST<2:0>			TRBSLV<2:0>		
23:16	R/W-0	R/W-1	R/W-1	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	FRACT	SELRES<1:0>		STRGSRC<4:0>				
15:8	R/W-0	U-0	R/W-0	R/W-1	R/W-0	R/W-0	R/W-0	U-0
	ON	—	SIDL	AICPMPEN	CVDEN	FSSCLKEN	FSPBCLKEN	—
7:0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	—	IRQVS<2:0>			STRGLVL	DMABL<2:0>		

图注：	HC = 硬件置 1 位	HS = 硬件清零位	
R = 可读位	W = 可写位	U = 未实现位，读为 0	
-n = POR 值	1 = 置 1	0 = 清零	x = 未知

bit 31 **TRBEN**：Turbo 通道使能位
1 = 使能 Turbo 通道
0 = 禁止 Turbo 通道

bit 30 **TRBERR**：Turbo 通道错误状态位
1 = 设置 Turbo 通道时发生错误，无论 TRBEN 位是否设置为 1， Turbo 通道功能都会被禁止。
0 = 未发生 Turbo 通道错误

注： 该位的状态仅在 TRBEN 位置 1 之后有效。

bit 29-27 **TRBMST<2:0>**：Turbo 主 ADCx 位
111 = 保留
110 = 选择 ADC6 作为 Turbo 主模块
•
•
•
000 = 选择 ADC0 作为 Turbo 主模块

bit 26-24 **TRBSLV<2:0>**：Turbo 从 ADCx 位
111 = 保留
110 = 选择 ADC6 作为 Turbo 从模块
•
•
•
000 = 选择 ADC0 作为 Turbo 从模块

bit 23 **FRACT**：小数数据输出格式位
1 = 小数
0 = 整数

bit 22-21 **SELRES<1:0>**：共用 ADC （ADC7）分辨率位
11 = 12 位 （默认）
10 = 10 位
01 = 8 位
00 = 6 位

bit 20-16 **STRGSRC<4:0>**：扫描触发源选择位
11111 - 00100 = 关于触发源选择，请参见具体器件数据手册中的 **“ADC”**章节
00011 = 保留
00010 = 全局电平软件触发 （GLSWTRG）不会自动清零
00001 = 全局软件触发 （GSWTRG）在下一个时钟周期自动清零
00000 = 无触发

bit 15 **ON**：ADC 模块使能位
1 = 使能 ADC 模块
0 = 禁止 ADC 模块

注： ON 位仅应在配置 ADC 模块之后置 1。

bit 14 未实现：读为 0

 © 2015 Microchip Technology Inc.

寄存器 22-1： ADCCON1：ADC 控制寄存器 1 （续）

bit 13 **SIDL：**空闲模式停止位

1 = 当器件进入空闲模式时，模块停止工作
0 = 在空闲模式下模块继续工作

bit 12 **AICPMPEN：**模拟输入电荷泵使能位

1 = 使能模拟输入电荷泵 （默认）
0 = 禁止模拟输入电荷泵

bit 11 **CVDEN：**电容分压使能位

1 = 使能 CVD 操作
0 = 禁止 CVD 操作

bit 10 **FSSCLKEN：**快速同步系统时钟与 ADC 控制时钟位

1 = 使能快速同步系统时钟与 ADC 控制时钟
0 = 禁止快速同步系统时钟与 ADC 控制时钟

bit 9 **FSPBCLKEN：**快速同步外设时钟与 ADC 控制时钟位

1 = 使能快速同步外设时钟与 ADC 控制时钟
0 = 禁止快速同步外设时钟与 ADC 控制时钟

bit 8-7 **未实现：**读为 0

bit 6-4 **IRQVS<2:0>：**中断向量移位位

为了确定中断向量地址，这些位指定在与 ADCBASE 寄存器值相加之前，对 ADCDSTAT1 和 ADCDSTAT2 寄存器中的 ARDYx 状态位进行左移的移位量（更多信息，请参见**第 22.6.2 节 “ADC 基址寄存器（ADCBASE）用法”**）。

中断向量地址 = ADCBASE 的读取值 = 写入 ADCBASE 的值 + x << IRQVS<2:0> ；其中，“x”是来自 ADCDSTAT1 或 ADCDSTAT2 寄存器的最小有效输入 ID （其优先级最高）。

111 = 将 x 左移 7 位
110 = 将 x 左移 6 位
101 = 将 x 左移 5 位
100 = 将 x 左移 4 位
011 = 将 x 左移 3 位
010 = 将 x 左移 2 位
001 = 将 x 左移 1 位
000 = 将 x 左移 0 位

bit 3 **STRGLVL：**扫描触发高电平 / 正边沿敏感位

1 = 扫描触发为高电平敏感。选择 STRIG 模式 （ADCTRGx 寄存器中的 TRGSRCx<4:0>）之后，会一直对所有选定的模拟输入进行扫描触发，直到 STRIG 选项被取消为止。
0 = 扫描触发为正边沿敏感。选择 STRIG 模式 （ADCTRGx 寄存器中的 TRGSRCx<4:0>）之后，仅产生单次扫描触发，它会完成所有选定模拟输入的扫描。

bit 2-0 **DMABL<2:0>：**DMA 缓冲区长度大小位

111 = 在 RAM 中为每个模拟输入分配 128 个单元
110 = 在 RAM 中为每个模拟输入分配 64 个单元
101 = 在 RAM 中为每个模拟输入分配 32 个单元
100 = 在 RAM 中为每个模拟输入分配 16 个单元
011 = 在 RAM 中为每个模拟输入分配 8 个单元
010 = 在 RAM 中为每个模拟输入分配 4 个单元
001 = 在 RAM 中为每个模拟输入分配 2 个单元
000 = 在 RAM 中为每个模拟输入分配 1 个单元

注： 由于每个输出数据为 16 位宽，所以 1 个单元包含 2 个字节。

寄存器 22-2： ADCCON2：ADC 控制寄存器 2

位范围	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	R-0, HS, HC	R-0, HS, HC	R-0, HS, HC	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	BGVRRDY	REFFLT	EOSRDY	CVDCPL<2:0>			SAMC<9:9>	
23:16	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	SAMC<7:0>							
15:8	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	BGVRIEN	REFFLTIEN	EOSIEN	ADCEIOVR	ECRIEN	ADCEIS<2:0>		
7:0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	—	ADCDIV<6:0>						

图注：	HC = 硬件置 1 位	HS = 硬件清零位	r = 保留
R = 可读位	W = 可写位	U = 未实现位，读为 0	
-n = POR 值	1 = 置 1	0 = 清零	x = 未知

bit 31 **BGVRRDY：**带隙电压 /ADC 参考电压状态位

1 = 带隙电压和 ADC 参考电压 （VREF）均已就绪
0 = 带隙电压和 / 或 ADC 参考电压 （VREF）未就绪

只有 BGVRRDY 位由硬件置 1 之后，数据处理才是有效的，所以应用程序代码必须检查 BGVRRDY 位是否置 1，以确保数据的有效性。当 ON 位 （ADCCON1<15>） = 0 时，该位会被设置为 0。

bit 30 **REFFLT：**带隙 /VREF/AVDD BOR 故障状态位

1 = 带隙或 VREF 电压在 ON 位 （ADCCON1<15>）置 1 时发生故障。导致带隙或 VREF 故障最有可能的原因是模拟 VDD 电源发生 BOR。
0 = 带隙和 VREF 电压正常工作

当 ON 位 （ADCCON1<15>） = 0 且 BGVRRDY 位 = 1 时，该位会被清零。

bit 29 **EOSRDY：**扫描结束中断状态位

1 = 需通过扫描触发扫描的所有模拟输入 （ADCCSS1 和 ADCCSS2 寄存器中指定的所有模拟输入）都已完成扫描
0 = 扫描尚未完成

用软件读取 ADCCON2<31:24> 时，该位会被清零。

bit 28-26 **CVDCPL<2:0>：**电容分压器 （CVD）设置位

111 = 7 * 2.5 pF = 17.5 pF
110 = 6 * 2.5 pF = 15 pF
101 = 5 * 2.5 pF = 12.5 pF
100 = 4 * 2.5 pF = 10 pF
011 = 3 * 2.5 pF = 7.5 pF
010 = 2 * 2.5 pF = 5 pF
001 = 1 * 2.5 pF = 2.5 pF
000 = 0 * 2.5 pF = 0 pF

bit 25-16 **SAMC<9:0>：**共用 ADC （ADC7）的采样时间位

1111111111 = 1025 TAD7
.
.
.
0000000001 = 3 TAD7
0000000000 = 2 TAD7

其中， TAD7 = 共用 ADC （ADC7）由 ADCDIV<6:0> 位控制的 ADC 转换时钟周期。

bit 15 **BGVRIEN：**带隙 /VREF 电压就绪中断允许位

1 = 在 BGVRRDY 位置 1 时产生中断
0 = 在 BGVRRDY 位置 1 时不产生中断

 © 2015 Microchip Technology Inc.

寄存器 22-2： ADCCON2：ADC 控制寄存器 2 （续）

bit 14 **REFFLTIEN**：带隙 /VREF 电压故障中断允许位
1 = 在 REFFLT 位置 1 时产生中断
0 = 在 REFFLT 位置 1 时不产生中断

bit 13 **EOSIEN**：扫描结束中断允许位
1 = 在 EOSRDY 位置 1 时产生中断
0 = 在 EOSRDY 位置 1 时不产生中断

bit 12 **ADCEIOVR**：提前中断请求改写位
1 = 改写提前中断产生，中断产生由 ADCGIRQEN1 和 ADCGIRQEN2 寄存器控制
0 = 不改写提前中断产生，中断产生由 ADCEIEN1 和 ADCEIEN2 寄存器控制

bit 11 **ECRIEN**：外部转换请求接口使能位
1 = 使能从外部模块 （如 PTG）启动 ADC 转换
0 = 外部模块无法启动 ADC 转换

bit 10-8 **ADCEIS<2:0>**：共用 ADC （ADC7）提前中断选择位
这些位用于选择在有效数据到达之前的多少个时钟 （TAD7）数处产生关联的中断。
111 = 在转换结束之前的 8 个 ADC 时钟处产生数据就绪中断
110 = 在转换结束之前的 7 个 ADC 时钟处产生数据就绪中断
•
•
•
001 = 在转换结束之前的 2 个 ADC 模块时钟处产生数据就绪中断
000 = 在转换结束之前的 1 个 ADC 模块时钟处产生数据就绪中断

注： 在选定分辨率 （由 SELRES<1:0> 位 （ADCCON1<22:21>）设置）为 12 位或 10 位时，所有选项都可用。选定分辨率为 8 位时，从 000 到 101 的选项是有效的。选定分辨率为 6 位时，从 000 到 011 的选项是有效的。

bit 7 **未实现**：读为 0

bit 6-0 **ADCDIV<6:0>**：共用 ADC （ADC7）分频比位
1111111 = 254 * TQ = TAD7
•
•
•
0000011 = 6 * TQ = TAD7
0000010 = 4 * TQ = TAD7
0000001 = 2 * TQ = TAD7
0000000 = 保留

ADCDIV<6:0> 位用于对 ADC 控制时钟 （TQ）进行分频，产生用于共用 ADC （ADC7）的时钟 （TAD7）。

寄存器 22-3： ADCCON3：ADC 控制寄存器 3

位范围	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	ADCSEL<1:0>		CONCLKDIV<5:0>					
23:16	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	DIGEN7	DIGEN6	DIGEN5	DIGEN4	DIGEN3	DIGEN2	DIGEN1	DIGEN0
15:8	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R-0, HS, HC	R/W-0	R-0, HS, HC
	VREFSEL<2:0>			TRGSUSP	UPDIEN	UPDRDY	SAMP(1,2,3,4)	RQCNVRT
7:0	R/W-0	R-0, HS, HC	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	GLSWTRG	GSWTRG	ADINSEL<5:0>					

图注：	HC = 硬件置 1 位	HS = 硬件清零位	
R = 可读位	W = 可写位	U = 未实现位，读为 0	
-n = POR 值	1 = 置 1	0 = 清零	x = 未知

bit 31-30 **ADCSEL<1:0>**：模数转换时钟源 （TCLK）位
11 = 系统时钟 （TCY）
10 = REFCLK3
01 = FRC 振荡器输出
00 = 外设总线时钟 （PBCLK）

bit 29-24 **CONCLKDIV<5:0>**：模数转换控制时钟 （TQ）分频比位
111111 = 126 * TCLK = TQ
•
•
•
000011 = 6 * TCLK = TQ
000010 = 4 * TCLK = TQ
000001 = 2 * TCLK = TQ
000000 = TCLK = TQ

bit 23 **DIGEN7**：共用 ADC （ADC7）数字使能位
1 = 数字使能 ADC7
0 = 数字禁止 ADC7

bit 22 **DIGEN6**：ADC6 数字使能位
1 = 数字使能 ADC6
0 = 数字禁止 ADC6

bit 21 **DIGEN5**：ADC5 数字使能位
1 = 数字使能 ADC5
0 = 数字禁止 ADC5

bit 20 **DIGEN4**：ADC4 数字使能位
1 = 数字使能 ADC4
0 = 数字禁止 ADC4

注 1： SAMP 位具有最高优先级，将该位置 1 会使 S&H 电路保持采样模式，直到该位清零为止。此外，使用 SAMP 位会导致 SAMC<9:0> 位 （ADCCON2<25:16>）的设置被忽略。

2： SAMP 位只会将 2 类和 3 类模拟输入与共用 ADC（ADC7）连接。所有 1 类模拟输入不会受 SAMP 位影响。

3： SAMP 位不是自清零位，应用程序软件需要负责先清零该位，然后才能通过将 RQCNVRT 位置 1 来启动模数转换。

4： 通常情况下，如果软件程序使用SAMP和RQCNVRT位，则应将所有TRGSRCx<4:0>位和STRGSRC<4:0>位设置为 00000，以禁止所有外部硬件触发，防止它们干扰软件控制的采样命令信号 SAMP 和软件控制的触发信号 RQCNVRT。

 © 2015 Microchip Technology Inc.

寄存器 22-3： ADCCON3：ADC 控制寄存器 3 （续）

bit 19 **DIGEN3**：ADC3 数字使能位
1 = 数字使能 ADC3
0 = 数字禁止 ADC3

bit 18 **DIGEN2**：ADC2 数字使能位
1 = 数字使能 ADC2
0 = 数字禁止 ADC2

bit 17 **DIGEN1**：ADC1 数字使能位
1 = 数字使能 ADC1
0 = 数字禁止 ADC1

bit 16 **DIGEN0**：ADC0 数字使能位
1 = 数字使能 ADC0
0 = 数字禁止 ADC0

bit 15-13 **VREFSEL<2:0>**：参考电压 （VREF）输入选择位

VREFSEL<2:0>	ADREF+	ADREF-
111	AVDD	内部 VREFL
110	内部 VREFH	AVSS
101	内部 VREFH	外部 VREFL
100	内部 VREFH	内部 VREFL
011	外部 VREFH	外部 VREFL
010	AVDD	外部 VREFL
001	外部 VREFH	AVSS
000	AVDD	AVSS

bit 12 **TRGSUSP**：触发暂停位
1 = 阻止触发信号启动新的模数转换，但 ADC 模块并未被禁止
0 = 不阻止触发信号

bit 11 **UPDIEN**：更新就绪中断允许位
1 = 在 UPDRDY 位由硬件置 1 时产生中断
0 = 不产生中断

bit 10 **UPDRDY**：ADC 更新就绪状态位
1 = 可以更新 ADC SFR
0 = 不能更新 ADC SFR

注： 只有 TRGSUSP 位置 1，并且所有 ADC 模块都没有任何正在运行的转换时，该位才有效。

bit 9 **SAMP**：2 类和 3 类模拟输入采样使能位 **(1,2,3,4)**
1 = ADC S&H 放大器处于采样模式
0 = ADC S&H 放大器处于保持模式

bit 8 **RQCNVRT**：单独 ADC 输入转换请求位
使用该位及其关联的 ADINSEL<5:0> 位，用户可以通过软件单独请求对某个模拟输入进行模数转换。
1 = 触发由 ADINSEL<5:0> 位指定的选定 ADC 输入的转换
0 = 不触发转换

注： 该位在下一个 ADC 时钟周期自动清零。

注 1： SAMP 位具有最高优先级，将该位置 1 会使 S&H 电路保持采样模式，直到该位清零为止。此外，使用 SAMP 位会导致 SAMC<9:0> 位 （ADCCON2<25:16>）的设置被忽略。

2： SAMP 位只会将 2 类和 3 类模拟输入与共用 ADC（ADC7）连接。所有 1 类模拟输入不会受 SAMP 位影响。

3： SAMP 位不是自清零位，应用程序软件需要负责先清零该位，然后才能通过将 RQCNVRT 位置 1 来启动模数转换。

4： 通常情况下，如果软件程序使用SAMP和RQCNVRT位，则应将所有TRGSRCx<4:0>位和STRGSRC<4:0>位设置为 00000，以禁止所有外部硬件触发，防止它们干扰软件控制的采样命令信号 SAMP 和软件控制的触发信号 RQCNVRT。

寄存器 22-3： ADCCON3：ADC 控制寄存器 3 （续）

bit 7 **GLSWTRG**：全局电平软件触发位

1 = 为已选择 GLSWTRG 位作为触发信号（通过 ADCTRGx 寄存器中的相关 TRGSRC<4:0> 位或 ADCCON1 寄存器中的 STRGSRC<4:0> 位）的 ADC 输入触发模数转换

0 = 不触发模数转换

bit 6 **GSWTRG**：全局软件触发位

1 = 为已选择 GSWTRG 位作为触发信号（通过 ADCTRGx 寄存器中的相关 TRGSRC<4:0> 位或 ADCCON1 寄存器中的 STRGSRC<4:0> 位）的 ADC 输入触发模数转换

0 = 不触发模数转换

注： 该位在下一个 ADC 时钟周期自动清零。

bit 5-0 **ADINSEL<5:0>**：模拟输入选择位

这些位用于选择要在 RQCNVRT 位置 1 时转换的模拟输入；其中， MAX_AN_INPUT 是器件上可用的最大模拟输入编号。关于可用的选择，请参见具体器件数据手册中的 **“ADC”** 章节。

MAX_AN_INPUT + 2 = IV_{TEMP}
MAX_AN_INPUT + 1 = IV_{REF}
MAX_AN_INPUT = AN[MAX_AN_INPUT]
•
•
•
000001 = AN1
000000 = AN0

注 **1**： SAMP 位具有最高优先级，将该位置 1 会使 S&H 电路保持采样模式，直到该位清零为止。此外，使用 SAMP 位会导致 SAMC<9:0> 位 （ADCCON2<25:16>）的设置被忽略。

2： SAMP 位只会将 2 类和 3 类模拟输入与共用 ADC（ADC7）连接。所有 1 类模拟输入不会受 SAMP 位影响。

3： SAMP 位不是自清零位，应用程序软件需要负责先清零该位，然后才能通过将 RQCNVRT 位置 1 来启动模数转换。

4： 通常情况下，如果软件程序使用SAMP和RQCNVRT位，则应将所有TRGSRCx<4:0>位和STRGSRC<4:0>位设置为 00000，以禁止所有外部硬件触发，防止它们干扰软件控制的采样命令信号 SAMP 和软件控制的触发信号 RQCNVRT。

 © 2015 Microchip Technology Inc.

寄存器 22-4： ADCTRGMODE：专用 ADC 的 ADC 触发模式寄存器

位范围	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	—	—	SH6ALT<1:0>		SH5ALT<1:0>		SH4ALT<1:0>	
23:16	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	SH3ALT<1:0>		SH2ALT<1:0>		SH1ALT<1:0>		SH0ALT<1:0>	
15:8	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	—	STRGEN6	STRGEN5	STRGEN4	STRGEN3	STRGEN2	STRGEN1	STRGEN0
7:0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	—	SSAMPEN6	SSAMPEN5	SSAMPEN4	SSAMPEN3	SSAMPEN2	SSAMPEN1	SSAMPEN0

图注：

R = 可读位	W = 可写位	U = 未实现位，读为 0	
-n = POR 值	1 = 置 1	0 = 清零	x = 未知

bit 31-30 未实现：读为 0

bit 29-28 **SH6ALT<1:0>**：ADC6 模拟输入选择位
11 - 01= 关于可用的选择，请参见具体器件数据手册中的 “ADC”章节。
00 = AN6

bit 27-26 **SH5ALT<1:0>**：ADC5 模拟输入选择位
11 - 01= 关于可用的选择，请参见具体器件数据手册中的 “ADC”章节。
00 = AN5

bit 25-24 **SH4ALT<1:0>**：ADC4 模拟输入选择位
11 - 01= 关于可用的选择，请参见具体器件数据手册中的 “ADC”章节。
00 = AN4

bit 23-22 **SH3ALT<1:0>**：ADC3 模拟输入选择位
11 - 01= 关于可用的选择，请参见具体器件数据手册中的 “ADC”章节。
00 = AN3

bit 21-20 **SH2ALT<1:0>**：ADC2 模拟输入选择位
11 - 01= 关于可用的选择，请参见具体器件数据手册中的 “ADC”章节。
00 = AN2

bit 19-18 **SH1ALT<1:0>**：ADC1 模拟输入选择位
11 - 01= 关于可用的选择，请参见具体器件数据手册中的 “ADC”章节。
00 = AN1

bit 17-16 **SH0ALT<1:0>**：ADC0 模拟输入选择位
11 - 01= 关于可用的选择，请参见具体器件数据手册中的 “ADC”章节。
00 = AN0

bit 15 未实现：读为 0

bit 14 **STRGEN6**：ADC6 预同步触发位
1 = ADC6 使用预同步触发
0 = ADC6 不使用预同步触发

bit 13 **STRGEN5**：ADC5 预同步触发位
1 = ADC5 使用预同步触发
0 = ADC5 不使用预同步触发

bit 12 **STRGEN4**：ADC4 预同步触发位
1 = ADC4 使用预同步触发
0 = ADC4 不使用预同步触发

bit 11 **STRGEN3**：ADC3 预同步触发位
1 = ADC3 使用预同步触发
0 = ADC3 不使用预同步触发

寄存器 22-4： ADCTRGMODE：专用 ADC 的 ADC 触发模式寄存器 （续）

bit 10 **STRGEN2：**ADC2 预同步触发位
1 = ADC2 使用预同步触发
0 = ADC2 不使用预同步触发

bit 9 **STRGEN1：**ADC1 预同步触发位
1 = ADC1 使用预同步触发
0 = ADC1 不使用预同步触发

bit 8 **STRGEN0：**ADC0 预同步触发位
1 = ADC0 使用预同步触发
0 = ADC0 不使用预同步触发

bit 7 未实现：读为 0

bit 6 **SSAMPEN6：**ADC6 同步采样位
1 = 对于空闲或禁止之后的第一个采样， ADC6 使用同步采样
0 = ADC6 不使用同步采样

bit 5 **SSAMPEN5：**ADC5 同步采样位
1 = 对于空闲或禁止之后的第一个采样， ADC5 使用同步采样
0 = ADC5 不使用同步采样

bit 4 **SSAMPEN4：**ADC4 同步采样位
1 = 对于空闲或禁止之后的第一个采样， ADC4 使用同步采样
0 = ADC4 不使用同步采样

bit 3 **SSAMPEN3：**ADC3 同步采样位
1 = 对于空闲或禁止之后的第一个采样， ADC3 使用同步采样
0 = ADC3 不使用同步采样

bit 2 **SSAMPEN2：**ADC2 同步采样位
1 = 对于空闲或禁止之后的第一个采样， ADC2 使用同步采样
0 = ADC2 不使用同步采样

bit 1 **SSAMPEN1：**ADC1 同步采样位
1 = 对于空闲或禁止之后的第一个采样， ADC1 使用同步采样
0 = ADC1 不使用同步采样

bit 0 **SSAMPEN0：**ADC0 同步采样位
1 = 对于空闲或禁止之后的第一个采样， ADC0 使用同步采样
0 = ADC0 不使用同步采样

 © 2015 Microchip Technology Inc.

寄存器 22-5： ADCIMCON1：ADC 输入模式控制寄存器 1

位范围	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	DIFF15	SIGN15	DIFF14	SIGN14	DIFF13	SIGN13	DIFF12	SIGN12
23:16	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	DIFF11	SIGN11	DIFF10	SIGN10	DIFF9	SIGN9	DIFF8	SIGN8
15:8	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	DIFF7	SIGN7	DIFF6	SIGN6	DIFF5	SIGN5	DIFF4	SIGN4
7:0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	DIFF3	SIGN3	DIFF2	SIGN2	DIFF1	SIGN1	DIFF0	SIGN0

图注：

R = 可读位	W = 可写位	U = 未实现位，读为 0	
-n = POR 值	1 = 置 1	0 = 清零	x = 未知

bit 31 **DIFF15**：AN15 模式位
1 = AN15 使用差分模式
0 = AN15 使用单端模式

bit 30 **SIGN15**：AN15 有符号数据模式位
1 = AN15 使用有符号数据模式
0 = AN15 使用无符号数据模式

bit 29 **DIFF14**：AN14 模式位
1 = AN14 使用差分模式
0 = AN14 使用单端模式

bit 28 **SIGN14**：AN14 有符号数据模式位
1 = AN14 使用有符号数据模式
0 = AN14 使用无符号数据模式

bit 27 **DIFF13**：AN13 模式位
1 = AN13 使用差分模式
0 = AN13 使用单端模式

bit 26 **SIGN13**：AN13 有符号数据模式位
1 = AN13 使用有符号数据模式
0 = AN13 使用无符号数据模式

bit 25 **DIFF12**：AN12 模式位
1 = AN12 使用差分模式
0 = AN12 使用单端模式

bit 24 **SIGN12**：AN12 有符号数据模式位
1 = AN12 使用有符号数据模式
0 = AN12 使用无符号数据模式

bit 23 **DIFF11**：AN11 模式位
1 = AN11 使用差分模式
0 = AN11 使用单端模式

bit 22 **SIGN11**：AN11 有符号数据模式位
1 = AN11 使用有符号数据模式
0 = AN11 使用无符号数据模式

bit 21 **DIFF10**：AN10 模式位
1 = AN10 使用差分模式
0 = AN10 使用单端模式

寄存器 22-5： ADCIMCON1：ADC 输入模式控制寄存器 1 （续）

bit 20 **SIGN10**：AN10 有符号数据模式位
1 = AN10 使用有符号数据模式
0 = AN10 使用无符号数据模式

bit 19 **DIFF9**：AN9 模式位
1 = AN9 使用差分模式
0 = AN9 使用单端模式

bit 18 **SIGN9**：AN9 有符号数据模式位
1 = AN9 使用有符号数据模式
0 = AN9 使用无符号数据模式

bit 17 **DIFF8**：AN8 模式位
1 = AN8 使用差分模式
0 = AN8 使用单端模式

bit 16 **SIGN8**：AN8 有符号数据模式位
1 = AN8 使用有符号数据模式
0 = AN8 使用无符号数据模式

bit 15 **DIFF7**：AN7 模式位
1 = AN7 使用差分模式
0 = AN7 使用单端模式

bit 14 **SIGN7**：AN7 有符号数据模式位
1 = AN7 使用有符号数据模式
0 = AN7 使用无符号数据模式

bit 13 **DIFF6**：AN6 模式位
1 = AN6 使用差分模式
0 = AN6 使用单端模式

bit 12 **SIGN6**：AN6 有符号数据模式位
1 = AN6 使用有符号数据模式
0 = AN6 使用无符号数据模式

bit 11 **DIFF5**：AN5 模式位
1 = AN5 使用差分模式
0 = AN5 使用单端模式

bit 10 **SIGN5**：AN5 有符号数据模式位
1 = AN5 使用有符号数据模式
0 = AN5 使用无符号数据模式

bit 9 **DIFF4**：AN4 模式位
1 = AN4 使用差分模式
0 = AN4 使用单端模式

bit 8 **SIGN4**：AN4 有符号数据模式位
1 = AN4 使用有符号数据模式
0 = AN4 使用无符号数据模式

bit 7 **DIFF3**：AN3 模式位
1 = AN3 使用差分模式
0 = AN3 使用单端模式

bit 6 **SIGN3**：AN3 有符号数据模式位
1 = AN3 使用有符号数据模式
0 = AN3 使用无符号数据模式

bit 5 **DIFF2**：AN2 模式位
1 = AN2 使用差分模式
0 = AN2 使用单端模式

 © 2015 Microchip Technology Inc.

寄存器 22-5： ADCIMCON1：ADC 输入模式控制寄存器 1 （续）

bit 4 **SIGN2**：AN2 有符号数据模式位
1 = AN2 使用有符号数据模式
0 = AN2 使用无符号数据模式

bit 3 **DIFF1**：AN1 模式位
1 = AN1 使用差分模式
0 = AN1 使用单端模式

bit 2 **SIGN1**：AN1 有符号数据模式位
1 = AN1 使用有符号数据模式
0 = AN1 使用无符号数据模式

bit 1 **DIFF0**：AN0 模式位
1 = AN0 使用差分模式
0 = AN0 使用单端模式

bit 0 **SIGN0**：AN0 有符号数据模式位
1 = AN0 使用有符号数据模式
0 = AN0 使用无符号数据模式

寄存器 22-6： ADCIMCON2：ADC 输入模式控制寄存器 2

位范围	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	DIFF31	SIGN31	DIFF30	SIGN30	DIFF29	SIGN29	DIFF28	SIGN28
23:16	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	DIFF27	SIGN27	DIFF26	SIGN26	DIFF25	SIGN25	DIFF24	SIGN24
15:8	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	DIFF23	SIGN23	DIFF22	SIGN22	DIFF21	SIGN21	DIFF20	SIGN20
7:0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	DIFF19	SIGN19	DIFF18	SIGN18	DIFF17	SIGN17	DIFF16	SIGN16

图注：

R = 可读位	W = 可写位	U = 未实现位，读为 0	
-n = POR 值	1 = 置 1	0 = 清零	x = 未知

bit 31 **DIFF31**：AN31 模式位
1 = AN31 使用差分模式
0 = AN31 使用单端模式

bit 30 **SIGN31**：AN31 有符号数据模式位
1 = AN31 使用有符号数据模式
0 = AN31 使用无符号数据模式

bit 29 **DIFF30**：AN30 模式位
1 = AN30 使用差分模式
0 = AN30 使用单端模式

bit 28 **SIGN30**：AN30 有符号数据模式位
1 = AN30 使用有符号数据模式
0 = AN30 使用无符号数据模式

bit 27 **DIFF29**：AN29 模式位
1 = AN29 使用差分模式
0 = AN29 使用单端模式

bit 26 **SIGN29**：AN29 有符号数据模式位
1 = AN29 使用有符号数据模式
0 = AN29 使用无符号数据模式

bit 25 **DIFF28**：AN28 模式位
1 = AN28 使用差分模式
0 = AN28 使用单端模式

bit 24 **SIGN28**：AN28 有符号数据模式位
1 = AN28 使用有符号数据模式
0 = AN28 使用无符号数据模式

bit 23 **DIFF27**：AN27 模式位
1 = AN27 使用差分模式
0 = AN27 使用单端模式

bit 22 **SIGN27**：AN27 有符号数据模式位
1 = AN27 使用有符号数据模式
0 = AN27 使用无符号数据模式

bit 21 **DIFF26**：AN26 模式位
1 = AN26 使用差分模式
0 = AN26 使用单端模式

 © 2015 Microchip Technology Inc.

寄存器 22-6： ADCIMCON2：ADC 输入模式控制寄存器 2 （续）

bit 20 **SIGN26**：AN26 有符号数据模式位
1 = AN26 使用有符号数据模式
0 = AN26 使用无符号数据模式

bit 19 **DIFF25**：AN25 模式位
1 = AN25 使用差分模式
0 = AN25 使用单端模式

bit 18 **SIGN25**：AN25 有符号数据模式位
1 = AN25 使用有符号数据模式
0 = AN25 使用无符号数据模式

bit 17 **DIFF24**：AN24 模式位
1 = AN24 使用差分模式
0 = AN24 使用单端模式

bit 16 **SIGN24**：AN24 有符号数据模式位
1 = AN24 使用有符号数据模式
0 = AN24 使用无符号数据模式

bit 15 **DIFF23**：AN23 模式位
1 = AN23 使用差分模式
0 = AN23 使用单端模式

bit 14 **SIGN23**：AN23 有符号数据模式位
1 = AN23 使用有符号数据模式
0 = AN23 使用无符号数据模式

bit 13 **DIFF22**：AN22 模式位
1 = AN22 使用差分模式
0 = AN22 使用单端模式

bit 12 **SIGN22**：AN22 有符号数据模式位
1 = AN22 使用有符号数据模式
0 = AN22 使用无符号数据模式

bit 11 **DIFF21**：AN21 模式位
1 = AN21 使用差分模式
0 = AN21 使用单端模式

bit 10 **SIGN21**：AN21 有符号数据模式位
1 = AN21 使用有符号数据模式
0 = AN21 使用无符号数据模式

bit 9 **DIFF20**：AN20 模式位
1 = AN20 使用差分模式
0 = AN20 使用单端模式

bit 8 **SIGN20**：AN20 有符号数据模式位
1 = AN20 使用有符号数据模式
0 = AN20 使用无符号数据模式

bit 7 **DIFF19**：AN19 模式位
1 = AN19 使用差分模式
0 = AN19 使用单端模式

bit 6 **SIGN19**：AN19 有符号数据模式位
1 = AN19 使用有符号数据模式
0 = AN19 使用无符号数据模式

bit 5 **DIFF18**：AN18 模式位
1 = AN18 使用差分模式
0 = AN18 使用单端模式

寄存器 22-6： ADCIMCON2：ADC 输入模式控制寄存器 2 （续）

bit 4 **SIGN18**：AN18 有符号数据模式位
1 = AN18 使用有符号数据模式
0 = AN18 使用无符号数据模式

bit 3 **DIFF17**：AN17 模式位
1 = AN17 使用差分模式
0 = AN17 使用单端模式

bit 2 **SIGN17**：AN17 有符号数据模式位
1 = AN17 使用有符号数据模式
0 = AN17 使用无符号数据模式

bit 1 **DIFF16**：AN16 模式位
1 = AN16 使用差分模式
0 = AN16 使用单端模式

bit 0 **SIGN16**：AN16 有符号数据模式位
1 = AN16 使用有符号数据模式
0 = AN16 使用无符号数据模式

 © 2015 Microchip Technology Inc.

寄存器 22-7： ADCIMCON3：ADC 输入模式控制寄存器 3

位范围	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	DIFF47	SIGN47	DIFF46	SIGN46	DIFF45	SIGN45	DIFF44	SIGN44
23:16	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	DIFF43	SIGN43	DIFF42	SIGN42	DIFF41	SIGN41	DIFF40	SIGN40
15:8	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	DIFF39	SIGN39	DIFF38	SIGN38	DIFF37	SIGN37	DIFF36	SIGN36
7:0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	DIFF35	SIGN35	DIFF34	SIGN34	DIFF33	SIGN33	DIFF32	SIGN32

图注：

R = 可读位	W = 可写位	U = 未实现位，读为 0	
-n = POR 值	1 = 置 1	0 = 清零	x = 未知

bit 31 **DIFF47**：AN47 模式位
1 = AN47 使用差分模式
0 = AN47 使用单端模式

bit 30 **SIGN47**：AN47 有符号数据模式位
1 = AN47 使用有符号数据模式
0 = AN47 使用无符号数据模式

bit 29 **DIFF46**：AN46 模式位
1 = AN46 使用差分模式
0 = AN46 使用单端模式

bit 28 **SIGN46**：AN46 有符号数据模式位
1 = AN46 使用有符号数据模式
0 = AN46 使用无符号数据模式

bit 27 **DIFF45**：AN45 模式位
1 = AN45 使用差分模式
0 = AN45 使用单端模式

bit 26 **SIGN45**：AN45 有符号数据模式位
1 = AN45 使用有符号数据模式
0 = AN45 使用无符号数据模式

bit 25 **DIFF44**：AN44 模式位
1 = AN44 使用差分模式
0 = AN44 使用单端模式

bit 24 **SIGN44**：AN44 有符号数据模式位
1 = AN44 使用有符号数据模式
0 = AN44 使用无符号数据模式

bit 23 **DIFF43**：AN43 模式位
1 = AN43 使用差分模式
0 = AN43 使用单端模式

bit 22 **SIGN43**：AN43 有符号数据模式位
1 = AN43 使用有符号数据模式
0 = AN43 使用无符号数据模式

bit 21 **DIFF42**：AN42 模式位
1 = AN42 使用差分模式
0 = AN42 使用单端模式

寄存器 22-7： ADCIMCON3：ADC 输入模式控制寄存器 3 （续）

bit 20 **SIGN42：**AN42 有符号数据模式位
1 = AN42 使用有符号数据模式
0 = AN42 使用无符号数据模式

bit 19 **DIFF41：**AN41 模式位
1 = AN41 使用差分模式
0 = AN41 使用单端模式

bit 18 **SIGN41：**AN41 有符号数据模式位
1 = AN41 使用有符号数据模式
0 = AN41 使用无符号数据模式

bit 17 **DIFF40：**AN40 模式位
1 = AN40 使用差分模式
0 = AN40 使用单端模式

bit 16 **SIGN40：**AN40 有符号数据模式位
1 = AN40 使用有符号数据模式
0 = AN40 使用无符号数据模式

bit 15 **DIFF39：**AN39 模式位
1 = AN39 使用差分模式
0 = AN39 使用单端模式

bit 14 **SIGN39：**AN39 有符号数据模式位
1 = AN39 使用有符号数据模式
0 = AN39 使用无符号数据模式

bit 13 **DIFF38：**AN38 模式位
1 = AN38 使用差分模式
0 = AN38 使用单端模式

bit 12 **SIGN38：**AN38 有符号数据模式位
1 = AN38 使用有符号数据模式
0 = AN38 使用无符号数据模式

bit 11 **DIFF37：**AN37 模式位
1 = AN37 使用差分模式
0 = AN37 使用单端模式

bit 10 **SIGN37：**AN37 有符号数据模式位
1 = AN37 使用有符号数据模式
0 = AN37 使用无符号数据模式

bit 9 **DIFF36：**AN36 模式位
1 = AN36 使用差分模式
0 = AN36 使用单端模式

bit 8 **SIGN36：**AN36 有符号数据模式位
1 = AN36 使用有符号数据模式
0 = AN36 使用无符号数据模式

bit 7 **DIFF35：**AN35 模式位
1 = AN35 使用差分模式
0 = AN35 使用单端模式

bit 6 **SIGN35：**AN35 有符号数据模式位
1 = AN35 使用有符号数据模式
0 = AN35 使用无符号数据模式

bit 5 **DIFF34：**AN34 模式位
1 = AN34 使用差分模式
0 = AN34 使用单端模式

 © 2015 Microchip Technology Inc.

寄存器 22-7： ADCIMCON3：ADC 输入模式控制寄存器 3 （续）

bit 4 **SIGN34：**AN34 有符号数据模式位
1 = AN34 使用有符号数据模式
0 = AN34 使用无符号数据模式

bit 3 **DIFF33：**AN33 模式位
1 = AN33 使用差分模式
0 = AN33 使用单端模式

bit 2 **SIGN33：**AN33 有符号数据模式位
1 = AN33 使用有符号数据模式
0 = AN33 使用无符号数据模式

bit 1 **DIFF32：**AN32 模式位
1 = AN32 使用差分模式
0 = AN32 使用单端模式

bit 0 **SIGN32：**AN32 有符号数据模式位
1 = AN32 使用有符号数据模式
0 = AN32 使用无符号数据模式

寄存器 22-8： ADCIMCON4：ADC 输入模式控制寄存器 4

位范围	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	DIFF63	SIGN63	DIFF62	SIGN62	DIFF61	SIGN61	DIFF60	SIGN60
23:16	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	DIFF59	SIGN59	DIFF58	SIGN58	DIFF57	SIGN57	DIFF56	SIGN56
15:8	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	DIFF55	SIGN55	DIFF54	SIGN54	DIFF53	SIGN53	DIFF52	SIGN52
7:0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	DIFF51	SIGN51	DIFF50	SIGN50	DIFF49	SIGN49	DIFF48	SIGN48

图注：

R = 可读位	W = 可写位	U = 未实现位，读为 0	
-n = POR 值	1 = 置 1	0 = 清零	x = 未知

bit 31 **DIFF63**：AN63 模式位
1 = AN63 使用差分模式
0 = AN63 使用单端模式

bit 30 **SIGN63**：AN63 有符号数据模式位
1 = AN63 使用有符号数据模式
0 = AN63 使用无符号数据模式

bit 29 **DIFF62**：AN62 模式位
1 = AN62 使用差分模式
0 = AN62 使用单端模式

bit 28 **SIGN62**：AN62 有符号数据模式位
1 = AN62 使用有符号数据模式
0 = AN62 使用无符号数据模式

bit 27 **DIFF61**：AN61 模式位
1 = AN61 使用差分模式
0 = AN61 使用单端模式

bit 26 **SIGN61**：AN61 有符号数据模式位
1 = AN61 使用有符号数据模式
0 = AN61 使用无符号数据模式

bit 25 **DIFF60**：AN60 模式位
1 = AN60 使用差分模式
0 = AN60 使用单端模式

bit 24 **SIGN60**：AN60 有符号数据模式位
1 = AN60 使用有符号数据模式
0 = AN60 使用无符号数据模式

bit 23 **DIFF59**：AN59 模式位
1 = AN59 使用差分模式
0 = AN59 使用单端模式

bit 22 **SIGN59**：AN59 有符号数据模式位
1 = AN59 使用有符号数据模式
0 = AN59 使用无符号数据模式

bit 21 **DIFF58**：AN58 模式位
1 = AN58 使用差分模式
0 = AN58 使用单端模式

 © 2015 Microchip Technology Inc.

寄存器 22-8： ADCIMCON4：ADC 输入模式控制寄存器 4 （续）

bit 20 **SIGN58**：AN58 有符号数据模式位
1 = AN58 使用有符号数据模式
0 = AN58 使用无符号数据模式

bit 19 **DIFF57**：AN57 模式位
1 = AN57 使用差分模式
0 = AN57 使用单端模式

bit 18 **SIGN57**：AN57 有符号数据模式位
1 = AN57 使用有符号数据模式
0 = AN57 使用无符号数据模式

bit 17 **DIFF56**：AN56 模式位
1 = AN56 使用差分模式
0 = AN56 使用单端模式

bit 16 **SIGN56**：AN56 有符号数据模式位
1 = AN56 使用有符号数据模式
0 = AN56 使用无符号数据模式

bit 15 **DIFF55**：AN55 模式位
1 = AN55 使用差分模式
0 = AN55 使用单端模式

bit 14 **SIGN55**：AN55 有符号数据模式位
1 = AN55 使用有符号数据模式
0 = AN55 使用无符号数据模式

bit 13 **DIFF54**：AN54 模式位
1 = AN54 使用差分模式
0 = AN54 使用单端模式

bit 12 **SIGN54**：AN54 有符号数据模式位
1 = AN54 使用有符号数据模式
0 = AN54 使用无符号数据模式

bit 11 **DIFF53**：AN53 模式位
1 = AN53 使用差分模式
0 = AN53 使用单端模式

bit 10 **SIGN53**：AN53 有符号数据模式位
1 = AN53 使用有符号数据模式
0 = AN53 使用无符号数据模式

bit 9 **DIFF52**：AN52 模式位
1 = AN52 使用差分模式
0 = AN52 使用单端模式

bit 8 **SIGN52**：AN52 有符号数据模式位
1 = AN52 使用有符号数据模式
0 = AN52 使用无符号数据模式

bit 7 **DIFF51**：AN51 模式位
1 = AN51 使用差分模式
0 = AN51 使用单端模式

bit 6 **SIGN51**：AN51 有符号数据模式位
1 = AN51 使用有符号数据模式
0 = AN51 使用无符号数据模式

bit 5 **DIFF50**：AN50 模式位
1 = AN50 使用差分模式
0 = AN50 使用单端模式

寄存器 22-8： ADCIMCON4：ADC 输入模式控制寄存器 4 （续）

bit 4 **SIGN50**：AN50 有符号数据模式位
1 = AN50 使用有符号数据模式
0 = AN50 使用无符号数据模式

bit 3 **DIFF49**：AN49 模式位
1 = AN49 使用差分模式
0 = AN49 使用单端模式

bit 2 **SIGN49**：AN49 有符号数据模式位
1 = AN49 使用有符号数据模式
0 = AN49 使用无符号数据模式

bit 1 **DIFF48**：AN48 模式位
1 = AN48 使用差分模式
0 = AN48 使用单端模式

bit 0 **SIGN48**：AN48 有符号数据模式位
1 = AN48 使用有符号数据模式
0 = AN48 使用无符号数据模式

 © 2015 Microchip Technology Inc.

寄存器 22-9： **ADCGIRQEN1：ADC 全局中断允许寄存器 1**

位范围	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	AGIEN31	AGIEN30	AGIEN29	AGIEN28	AGIEN27	AGIEN26	AGIEN25	AGIEN24
23:16	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	AGIEN23	AGIEN22	AGIEN21	AGIEN20	AGIEN19	AGIEN18	AGIEN17	AGIEN16
15:8	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	AGIEN15	AGIEN14	AGIEN13	AGIEN12	AGIEN11	AGIEN10	AGIEN9	AGIEN8
7:0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	AGIEN7	AGIEN6	AGIEN5	AGIEN4	AGIEN3	AGIEN2	AGIEN1	AGIEN0

图注：

R = 可读位	W = 可写位	U = 未实现位，读为 0	
-n = POR 值	1 = 置 1	0 = 清零	x = 未知

bit 31-0 **AGIEN31:AGIEN0**：ADC 中断允许位

1 = 允许选定模拟输入的中断。在转换数据就绪 （通过 ADCDSTAT1 寄存器的 ARDYx 位 （x = 31-0）指示）之后产生中断

0 = 禁止中断

寄存器 22-10： **ADCGIRQEN2：ADC 全局中断允许寄存器 2**

位范围	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	AGIEN63	AGIEN62	AGIEN61	AGIEN60	AGIEN59	AGIEN58	AGIEN57	AGIEN56
23:16	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	AGIEN55	AGIEN54	AGIEN53	AGIEN52	AGIEN51	AGIEN50	AGIEN49	AGIEN48
15:8	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	AGIEN47	AGIEN46	AGIEN45	AGIEN44	AGIEN43	AGIEN42	AGIEN41	AGIEN40
7:0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	AGIEN39	AGIEN38	AGIEN37	AGIEN36	AGIEN35	AGIEN34	AGIEN33	AGIEN32

图注：

R = 可读位	W = 可写位	U = 未实现位，读为 0	
-n = POR 值	1 = 置 1	0 = 清零	x = 未知

bit 31-0 **AGIEN63:AGIEN32**：ADC 中断允许位

1 = 允许选定模拟输入的中断。在转换数据就绪 （通过 ADCDSTAT2 寄存器的 ARDYx 位 （x = 63-32）指示）之后产生中断

0 = 禁止中断

寄存器 22-11： ADCCSS1：ADC 公共扫描选择寄存器 1

位范围	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	CSS31	CSS30	CSS29	CSS28	CSS27	CSS26	CSS25	CSS24
23:16	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	CSS23	CSS22	CSS21	CSS20	CSS19	CSS18	CSS17	CSS16
15:8	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	CSS15	CSS14	CSS13	CSS12	CSS11	CSS10	CSS9	CSS8
7:0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	CSS7	CSS6	CSS5	CSS4	CSS3	CSS2	CSS1	CSS0

图注：

R = 可读位	W = 可写位	U = 未实现位，读为 0	
-n = POR 值	1 = 置 1	0 = 清零	x = 未知

bit 31-0 **CSS31:CSS0**：模拟公共扫描选择位
1 = 对于输入扫描选择 ANx
0 = 对于输入扫描跳过 ANx

注 1： 如果要通过 CSSx 位扫描 1 类和 2 类模拟输入，除了将该寄存器中的相应位置 1 之外， 1 类和 2 类模拟输入还必须选择 STRIG 输入作为触发源。请参见 ADCTRGx 寄存器 （寄存器 22-18）中的位说明来选择 STRIG 选项。

2： 如果通过将 CSSx 位设置为 1 和将 TRGSRCx<4:0> 位设置为 STRIG 模式 （'0b11）而在扫描中包含某个 1 类或 2 类输入，则用户应用程序必须确保不使用 ADCCON3 寄存器中的 RQCNVRT 位或者硬件输入或任何数字滤波器为该输入产生其他触发信号。否则，扫描行为是不可预测的。

寄存器 22-12： ADCCSS2：ADC 公共扫描选择寄存器 2

位范围	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	CSS63	CSS62	CSS61	CSS60	CSS59	CSS58	CSS57	CSS56
23:16	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	CSS55	CSS54	CSS53	CSS52	CSS51	CSS50	CSS49	CSS48
15:8	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	CSS47	CSS46	CSS45	CSS44	CSS43	CSS42	CSS41	CSS40
7:0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	CSS39	CSS38	CSS37	CSS36	CSS35	CSS34	CSS33	CSS32

图注：

R = 可读位	W = 可写位	U = 未实现位，读为 0	
-n = POR 值	1 = 置 1	0 = 清零	x = 未知

bit 31-0 **CSS63:CSS32**：模拟公共扫描选择位
1 = 对于输入扫描选择 ANx
0 = 对于输入扫描跳过 ANx

注： 模拟输入 63 至 32 总是为 3 类输入，因为只有 32 个触发源可用。

 © 2015 Microchip Technology Inc.

寄存器 22-13： ADCDSTAT1：ADC 数据就绪状态寄存器 1

位范围	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	R-0, HS, HC	R-0, HS, HC	R-0, HS, HC	R-0, HS, HC	R-0, HS, HC	R-0, HS, HC	R-0, HS, HC	R-0, HS, HC
	ARDY31	ARDY30	ARDY29	ARDY28	ARDY27	ARDY26	ARDY25	ARDY24
23:16	R-0, HS, HC	R-0, HS, HC	R-0, HS, HC	R-0, HS, HC	R-0, HS, HC	R-0, HS, HC	R-0, HS, HC	R-0, HS, HC
	ARDY23	ARDY22	ARDY21	ARDY20	ARDY19	ARDY18	ARDY17	ARDY16
15:8	R-0, HS, HC	R-0, HS, HC	R-0, HS, HC	R-0, HS, HC	R-0, HS, HC	R-0, HS, HC	R-0, HS, HC	R-0, HS, HC
	ARDY15	ARDY14	ARDY13	ARDY12	ARDY11	ARDY10	ARDY9	ARDY8
7:0	R-0, HS, HC	R-0, HS, HC	R-0, HS, HC	R-0, HS, HC	R-0, HS, HC	R-0, HS, HC	R-0, HS, HC	R-0, HS, HC
	ARDY7	ARDY6	ARDY5	ARDY4	ARDY3	ARDY2	ARDY1	ARDY0

图注：	HS = 硬件置 1 位	HC = 硬件清零位	
R = 可读位	W = 可写位	U = 未实现位，读为 0	
-n = POR 值	1 = 置 1	0 = 清零	x = 未知

bit 31-0 **ARDY31:ARDY0：**相应模拟输入的转换数据就绪位
1 = 转换数据在数据寄存器中就绪时，该位置 1
0 = 读取相关的数据寄存器时，该位清零

寄存器 22-14： ADCDSTAT2：ADC 数据就绪状态寄存器 2

位范围	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	R-0, HS, HC	R-0, HS, HC	R-0, HS, HC	R-0, HS, HC	R-0, HS, HC	R-0, HS, HC	R-0, HS, HC	R-0, HS, HC
	ARDY63	ARDY62	ARDY61	ARDY60	ARDY59	ARDY58	ARDY57	ARDY56
23:16	R-0, HS, HC	R-0, HS, HC	R-0, HS, HC	R-0, HS, HC	R-0, HS, HC	R-0, HS, HC	R-0, HS, HC	R-0, HS, HC
	ARDY55	ARDY54	ARDY53	ARDY52	ARDY51	ARDY50	ARDY49	ARDY48
15:8	R-0, HS, HC	R-0, HS, HC	R-0, HS, HC	R-0, HS, HC	R-0, HS, HC	R-0, HS, HC	R-0, HS, HC	R-0, HS, HC
	ARDY47	ARDY46	ARDY45	ARDY44	ARDY43	ARDY42	ARDY41	ARDY40
7:0	R-0, HS, HC	R-0, HS, HC	R-0, HS, HC	R-0, HS, HC	R-0, HS, HC	R-0, HS, HC	R-0, HS, HC	R-0, HS, HC
	ARDY39	ARDY38	ARDY37	ARDY36	ARDY35	ARDY34	ARDY33	ARDY32

图注：	HS = 硬件置 1 位	HC = 硬件清零位	
R = 可读位	W = 可写位	U = 未实现位，读为 0	
-n = POR 值	1 = 置 1	0 = 清零	x = 未知

bit 31-0 **ARDY63:ARDY32：**相应模拟输入的转换数据就绪位
1 = 转换数据在数据寄存器中就绪时，该位置 1
0 = 读取相关的数据寄存器时，该位清零

寄存器 22-15： ADCCMPENx：ADC 数字比较器 x 使能寄存器 （x = 1 至 6）

位范围	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	CMPE31	CMPE30	CMPE29	CMPE28	CMPE27	CMPE26	CMPE25	CMPE24
23:16	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	CMPE23	CMPE22	CMPE21	CMPE20	CMPE19	CMPE18	CMPE17	CMPE16
15:8	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	CMPE15	CMPE14	CMPE13	CMPE12	CMPE11	CMPE10	CMPE9	CMPE8
7:0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	CMPE7	CMPE6	CMPE5	CMPE4	CMPE3	CMPE2	CMPE1	CMPE0

图注：

R = 可读位	W = 可写位	U = 未实现位，读为 0	
-n = POR 值	1 = 置 1	0 = 清零	x = 未知

bit 31-0 **CMPE31:CMPE0**：ADC 数字比较器 x 使能位

这些位使对应模拟输入的转换结果由数字比较器处理。CMPE0 用于使能 AN0， CMPE1 用于使能 AN1，如此类推。

注 **1**： CMPEx = ANx，其中的 x = 0-31 （数字比较器输入限于 AN0 至 AN31）。

2： 使能数字比较器 （ENDCMP = 1）时更改这些位会导致不可预测的行为。

寄存器 22-16： ADCCMPx：ADC 数字比较器 x 限值寄存器 （x = 1 至 6）

位范围	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	DCMPHI<15:8>(1,2,3)							
23:16	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	DCMPHI<7:0>(1,2,3)							
15:8	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	DCMPLO<15:8>(1,2,3)							
7:0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	DCMPLO<7:0>(1,2,3)							

图注：

R = 可读位	W = 可写位	U = 未实现位，读为 0	
-n = POR 值	1 = 置 1	0 = 清零	x = 未知

bit 31-16 **DCMPHI<15:0>**：数字比较器 x 上限值位 (1,2,3)

这些位存储上限值，数字比较器使用该值来与 ADC 转换数据进行比较。

bit 15-0 **DCMPLO<15:0>**：数字比较器 x 下限值位 (1,2,3)

这些位存储下限值，数字比较器使用该值来与 ADC 转换数据进行比较。

注 **1**： 使能数字比较器 （ENDCMP = 1）时更改这些位会导致不可预测的行为。

2： 限值的格式应与 ADC 转换值的格式在符号和小数设置方面匹配。

3： 对于在 CVD 模式下使用的数字比较器 1， DCMPHI<15:0> 和 DCMPLO<15:0> 位必须总是指定为有符号格式，因为 CVD 输出数据是差分的，并且总是有符号的。

 © 2015 Microchip Technology Inc.

寄存器 22-17： ADCFLTR*x*：ADC 数字滤波器 x 寄存器 （x = 1 至 6）

位范围	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R-0, HS, HC
	AFEN	DATA16EN	DFMODE	OVRSAM<2:0>			AFGIEN	AFRDY
23:16	U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	—	—	—	CHNLID<4:0>				
15:8	R-0, HS, HC	R-0, HS, HC	R-0, HS, HC	R-0, HS, HC	R-0, HS, HC	R-0, HS, HC	R-0, HS, HC	R-0, HS, HC
	FLTRDATA<15:8>							
7:0	R-0, HS, HC	R-0, HS, HC	R-0, HS, HC	R-0, HS, HC	R-0, HS, HC	R-0, HS, HC	R-0, HS, HC	R-0, HS, HC
	FLTRDATA<7:0>							

图注：	HS = 硬件置 1 位	HC = 硬件清零位	
R = 可读位	W = 可写位	U = 未实现位，读为 0	
-n = POR 值	1 = 置 1	0 = 清零	x = 未知

bit 31 **AFEN**：数字滤波器 x 使能位
1 = 使能数字滤波器
0 = 禁止数字滤波器，并清零 AFRDY 状态位

bit 30 **DATA16EN**：滤波器有效数据长度位
1 = 滤波器输出数据的全部 16 位都是有效的
0 = 只有前 12 位是有效的，后面跟随 4 个 0

注： 只有在 DFMODE = 1 （平均模式）且 FRACT （ADCCON1<23>） = 1 （小数输出模式）时，该位才有效。

bit 29 **DFMODE**：ADC 滤波器模式位
1 = 滤波器 x 以平均模式工作
0 = 滤波器 x 以过采样模式工作 （默认）

bit 28-26 **OVRSAM<2:0>**：过采样滤波率位
如果 DFMODE 为 0：
111 = 128 个采样 （将总和向右移 3 位，输出数据采用 15.1 格式）
110 = 32 个采样 （将总和向右移 2 位，输出数据采用 14.1 格式）
101 = 8 个采样 （将总和向右移 1 位，输出数据采用 13.1 格式）
100 = 2 个采样 （将总和向右移 0 位，输出数据采用 12.1 格式）
011 = 256 个采样 （将总和向右移 4 位，输出数据为 16 位）
010 = 64 个采样 （将总和向右移 3 位，输出数据为 15 位）
001 = 16 个采样 （将总和向右移 2 位，输出数据为 14 位）
000 = 4 个采样 （将总和向右移 1 位，输出数据为 13 位）

如果 DFMODE 为 1：
111 = 256 个采样 （256 个要进行平均的采样）
110 = 128 个采样 （128 个要进行平均的采样）
101 = 64 个采样 （64 个要进行平均的采样）
100 = 32 个采样 （32 个要进行平均的采样）
011 = 16 个采样 （16 个要进行平均的采样）
010 = 8 个采样 （8 个要进行平均的采样）
001 = 4 个采样 （4 个要进行平均的采样）
000 = 2 个采样 （2 个要进行平均的采样）

bit 25 **AFGIEN**：数字滤波器 x 中断允许位
1 = 允许数字滤波器中断，并根据 AFRDY 状态位产生
0 = 禁止数字滤波器

寄存器 22-17： ADCFLTR*x*：ADC 数字滤波器 x 寄存器 （x = 1 至 6）（续）

bit 24 **AFRDY**：数字滤波器 x 数据就绪状态位
1 = 数据在 FLTRDATA<15:0> 位中就绪
0 = 数据未就绪

注： 该位通过读取 FLTRDATA<15:0> 位或通过禁止数字滤波器模块（通过将 AFEN 设置为 0）来清零。

bit 23-21 未实现：读为 0

bit 20-16 **CHNLID<4:0>**：数字滤波器模拟输入选择位
这些位指定要用作过采样滤波器数据源的模拟输入。
11111 = AN31
•
•
•
00010 = AN2
00001 = AN1
00000 = AN0

注： 只有前 32 个模拟输入 （1 类和 2 类）可以使用数字滤波器。

bit 15-0 **FLTRDATA<15:0>**：数字滤波器 x 数据输出值位
滤波器输出数据采用 FRACT 位（ADCCON1<23>）中设置的小数格式。在使能滤波器时，不应更改 FRACT 位。如果在滤波器的操作结束后更改 FRACT 位的状态，不会更新 FLTRDATA<15:0> 位的值来反映新的格式。

 © 2015 Microchip Technology Inc.

寄存器 22-18： ADCTRG1：ADC 触发源 1 寄存器

位范围	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	—	—	—	TRGSRC3<4:0>				
23:16	U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	—	—	—	TRGSRC2<4:0>				
15:8	U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	—	—	—	TRGSRC1<4:0>				
7:0	U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	—	—	—	TRGSRC0<4:0>				

图注：

R = 可读位	W = 可写位	U = 未实现位，读为 0	
-n = POR 值	1 = 置 1	0 = 清零	x = 未知

bit 31-29 未实现：读为 0

bit 28-24 **TRGSRC3<4:0>**：模拟输入 AN3 转换触发源选择位
11111 - 00100 = 关于触发源选择，请参见具体器件数据手册中的 **“ADC”** 章节。
00011 = 扫描触发 （STRIG）
00010 = 选择全局电平软件触发 （GLSWTRG）
00001 = 全局软件边沿触发 （GSWTRG）
00000 = 无触发

对于 STRIG，除了设置触发之外，还需通过设定 STRGSRC<4:0> 位 （ADCCON1<20:16>）来选择触发源，并要求 ADCCSSx 寄存器中相应的 CSSx 位置 1。

bit 23-21 未实现：读为 0

bit 20-16 **TRGSRC2<4:0>**：模拟输入 AN2 转换触发源选择位
关于位值定义，请参见 bit 28-24。

bit 15-13 未实现：读为 0

bit 12-8 **TRGSRC1<4:0>**：模拟输入 AN1 转换触发源选择位
关于位值定义，请参见 bit 28-24。

bit 7-5 未实现：读为 0

bit 4-0 **TRGSRC0<4:0>**：模拟输入 AN0 转换触发源选择位
关于位值定义，请参见 bit 28-24。

寄存器 22-19： ADCTRG2：ADC 触发源 2 寄存器

位范围	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	—	—	—	TRGSRC7<4:0>				
23:16	U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	—	—	—	TRGSRC6<4:0>				
15:8	U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	—	—	—	TRGSRC5<4:0>				
7:0	U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	—	—	—	TRGSRC4<4:0>				

图注：			
R = 可读位	W = 可写位	U = 未实现位，读为 0	
-n = POR 值	1 = 置 1	0 = 清零	x = 未知

bit 31-29 **未实现：**读为 0

bit 28-24 **TRGSRC7<4:0>：**模拟输入 AN7 转换触发源选择位
11111 - 00100 = 关于触发源选择，请参见具体器件数据手册中的 **“ADC”**章节。
00011 = 扫描触发 （STRIG）
00010 = 选择全局电平软件触发 （GLSWTRG）
00001 = 全局软件边沿触发 （GSWTRG）
00000 = 无触发

对于 STRIG，除了设置触发之外，还需通过设定 STRGSRC<4:0> 位 （ADCCON1<20:16>）来选择触发源，并要求 ADCCSSx 寄存器中相应的 CSSx 位置 1。

bit 23-21 **未实现：**读为 0

bit 20-16 **TRGSRC6<4:0>：**模拟输入 AN6 转换触发源选择位
关于位值定义，请参见 bit 28-24。

bit 15-13 **未实现：**读为 0

bit 12-8 **TRGSRC5<4:0>：**模拟输入 AN5 转换触发源选择位
关于位值定义，请参见 bit 28-24。

bit 7-5 **未实现：**读为 0

bit 4-0 **TRGSRC4<4:0>：**模拟输入 AN4 转换触发源选择位
关于位值定义，请参见 bit 28-24。

 © 2015 Microchip Technology Inc.

寄存器 22-20： ADCTRG3：ADC 触发源 3 寄存器

位范围	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	—	—	—	TRGSRC11<4:0>				
23:16	U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	—	—	—	TRGSRC10<4:0>				
15:8	U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	—	—	—	TRGSRC9<4:0>				
7:0	U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	—	—	—	TRGSRC8<4:0>				

图注：			
R = 可读位	W = 可写位	U = 未实现位，读为 0	
-n = POR 值	1 = 置 1	0 = 清零	x = 未知

bit 31-29 未实现：读为 0

bit 28-24 **TRGSRC11<4:0>**：模拟输入 AN11 转换触发源选择位
11111 - 00100 = 关于触发源选择，请参见具体器件数据手册中的 **“ADC”** 章节。
00011 = 扫描触发 （STRIG）
00010 = 选择全局电平软件触发 （GLSWTRG）
00001 = 全局软件边沿触发 （GSWTRG）
00000 = 无触发

对于 STRIG，除了设置触发之外，还需通过设定 STRGSRC<4:0> 位 （ADCCON1<20:16>）来选择触发源，并要求 ADCCSSx 寄存器中相应的 CSSx 位置 1。

bit 23-21 未实现：读为 0

bit 20-16 **TRGSRC10<4:0>**：模拟输入 AN10 转换触发源选择位
关于位值定义，请参见 bit 28-24。

bit 15-13 未实现：读为 0

bit 12-8 **TRGSRC9<4:0>**：模拟输入 AN9 转换触发源选择位
关于位值定义，请参见 bit 28-24。

bit 7-5 未实现：读为 0

bit 4-0 **TRGSRC8<4:0>**：模拟输入 AN8 转换触发源选择位
关于位值定义，请参见 bit 28-24。

寄存器 22-21： ADCTRG4：ADC 触发源 4 寄存器

位范围	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	—	—	—	TRGSRC15<4:0>				
23:16	U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	—	—	—	TRGSRC14<4:0>				
15:8	U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	—	—	—	TRGSRC13<4:0>				
7:0	U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	—	—	—	TRGSRC12<4:0>				

图注：			
R = 可读位	W = 可写位	U = 未实现位，读为 0	
-n = POR 值	1 = 置 1	0 = 清零	x = 未知

bit 31-29 未实现：读为 0

bit 28-24 **TRGSRC15<4:0>**：模拟输入 AN15 转换触发源选择位
11111 - 00100 = 关于触发源选择，请参见具体器件数据手册中的 **“ADC”** 章节。
00011 = 扫描触发 （STRIG）
00010 = 选择全局电平软件触发 （GLSWTRG）
00001 = 全局软件边沿触发 （GSWTRG）
00000 = 无触发

对于 STRIG，除了设置触发之外，还需通过设定 STRGSRC<4:0> 位 （ADCCON1<20:16>）来选择触发源，并要求 ADCCSSx 寄存器中相应的 CSSx 位置 1。

bit 23-21 未实现：读为 0

bit 20-16 **TRGSRC14<4:0>**：模拟输入 AN14 转换触发源选择位
关于位值定义，请参见 bit 28-24。

bit 15-13 未实现：读为 0

bit 12-8 **TRGSRC13<4:0>**：模拟输入 AN13 转换触发源选择位
关于位值定义，请参见 bit 28-24。

bit 7-5 未实现：读为 0

bit 4-0 **TRGSRC12<4:0>**：模拟输入 AN12 转换触发源选择位
关于位值定义，请参见 bit 28-24。

 © 2015 Microchip Technology Inc.

寄存器 22-22： ADCTRG5：ADC 触发源 5 寄存器

位范围	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	—	—	—	TRGSRC19<4:0>				
23:16	U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	—	—	—	TRGSRC18<4:0>				
15:8	U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	—	—	—	TRGSRC17<4:0>				
7:0	U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	—	—	—	TRGSRC16<4:0>				

图注：			
R = 可读位	W = 可写位	U = 未实现位，读为 0	
-n = POR 值	1 = 置 1	0 = 清零	x = 未知

bit 31-29 未实现：读为 0

bit 28-24 **TRGSRC19<4:0>**：模拟输入 AN19 转换触发源选择位
11111 - 00100 = 关于触发源选择，请参见具体器件数据手册中的 **“ADC”** 章节。
00011 = 扫描触发 （STRIG）
00010 = 选择全局电平软件触发 （GLSWTRG）
00001 = 全局软件边沿触发 （GSWTRG）
00000 = 无触发

对于 STRIG，除了设置触发之外，还需通过设定 STRGSRC<4:0> 位 （ADCCON1<20:16>）来选择触发源，并要求 ADCCSSx 寄存器中相应的 CSSx 位置 1。

bit 23-21 未实现：读为 0

bit 20-16 **TRGSRC18<4:0>**：模拟输入 AN18 转换触发源选择位
关于位值定义，请参见 bit 28-24。

bit 15-13 未实现：读为 0

bit 12-8 **TRGSRC17<4:0>**：模拟输入 AN17 转换触发源选择位
关于位值定义，请参见 bit 28-24。

bit 7-5 未实现：读为 0

bit 4-0 **TRGSRC16<4:0>**：模拟输入 AN16 转换触发源选择位
关于位值定义，请参见 bit 28-24。

寄存器 22-23： ADCTRG6：ADC 触发源 6 寄存器

位范围	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	—	—	—	TRGSRC23<4:0>				
23:16	U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	—	—	—	TRGSRC22<4:0>				
15:8	U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	—	—	—	TRGSRC21<4:0>				
7:0	U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	—	—	—	TRGSRC20<4:0>				

图注：			
R = 可读位	W = 可写位	U = 未实现位，读为 0	
-n = POR 值	1 = 置 1	0 = 清零	x = 未知

bit 31-29 未实现：读为 0

bit 28-24 **TRGSRC23<4:0>**：模拟输入 AN23 转换触发源选择位
11111-00100 = 更多信息，请参见具体器件数据手册中的 **“ADC”** 章节。
00011 = 扫描触发 （STRIG）
00010 = 选择全局电平软件触发 （GLSWTRG）
00001 = 全局软件边沿触发 （GSWTRG）
00000 = 无触发

对于 STRIG，除了设置触发之外，还需通过设定 STRGSRC<4:0> 位 （ADCCON1<20:16>）来选择触发源，并要求 ADCCSSx 寄存器中相应的 CSSx 位置 1。

bit 23-21 未实现：读为 0

bit 20-16 **TRGSRC22<4:0>**：模拟输入 AN22 转换触发源选择位
关于位值定义，请参见 bit 28-24。

bit 15-13 未实现：读为 0

bit 12-8 **TRGSRC21<4:0>**：模拟输入 AN21 转换触发源选择位
关于位值定义，请参见 bit 28-24。

bit 7-5 未实现：读为 0

bit 4-0 **TRGSRC20<4:0>**：模拟输入 AN20 转换触发源选择位
关于位值定义，请参见 bit 28-24。

 © 2015 Microchip Technology Inc.

寄存器 22-24： ADCTRG7：ADC 触发源 7 寄存器

位范围	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	—	—	—	TRGSRC27<4:0>				
23:16	U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	—	—	—	TRGSRC26<4:0>				
15:8	U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	—	—	—	TRGSRC25<4:0>				
7:0	U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	—	—	—	TRGSRC24<4:0>				

图注：			
R = 可读位	W = 可写位	U = 未实现位，读为 0	
-n = POR 值	1 = 置 1	0 = 清零	x = 未知

bit 31-29 未实现：读为 0

bit 28-24 **TRGSRC27<4:0>**：模拟输入 AN27 转换触发源选择位
11111-00100 = 更多信息，请参见具体器件数据手册中的 **“ADC”** 章节。
00011 = 扫描触发 （STRIG）
00010 = 选择全局电平软件触发 （GLSWTRG）
00001 = 全局软件边沿触发 （GSWTRG）
00000 = 无触发

对于 STRIG，除了设置触发之外，还需通过设定 STRGSRC<4:0> 位 （ADCCON1<20:16>）来选择触发源，并要求 ADCCSSx 寄存器中相应的 CSSx 位置 1。

bit 23-21 未实现：读为 0

bit 20-16 **TRGSRC26<4:0>**：模拟输入 AN26 转换触发源选择位
关于位值定义，请参见 bit 28-24。

bit 15-13 未实现：读为 0

bit 12-8 **TRGSRC25<4:0>**：模拟输入 AN25 转换触发源选择位
关于位值定义，请参见 bit 28-24。

bit 7-5 未实现：读为 0

bit 4-0 **TRGSRC24<4:0>**：模拟输入 AN24 转换触发源选择位
关于位值定义，请参见 bit 28-24。

寄存器 22-25： ADCTRG8：ADC 触发源 8 寄存器

位范围	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	—	—	—	TRGSRC31<4:0>				
23:16	U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	—	—	—	TRGSRC30<4:0>				
15:8	U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	—	—	—	TRGSRC29<4:0>				
7:0	U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	—	—	—	TRGSRC28<4:0>				

图注：			
R = 可读位	W = 可写位	U = 未实现位，读为 0	
-n = POR 值	1 = 置 1	0 = 清零	x = 未知

bit 31-29 未实现：读为 0

bit 28-24 **TRGSRC31<4:0>**：模拟输入 AN31 转换触发源选择位
11111-00100 = 更多信息，请参见具体器件数据手册中的 **“ADC”** 章节
00011 = 扫描触发 （STRIG）
00010 = 选择全局电平软件触发 （GLSWTRG）
00001 = 全局软件边沿触发 （GSWTRG）
00000 = 无触发

对于 STRIG，除了设置触发之外，还需通过设定 STRGSRC<4:0> 位 （ADCCON1<20:16>）来选择触发源，并要求 ADCCSSx 寄存器中相应的 CSSx 位置 1。

bit 23-21 未实现：读为 0

bit 20-16 **TRGSRC30<4:0>**：模拟输入 AN30 转换触发源选择位
关于位值定义，请参见 bit 28-24。

bit 15-13 未实现：读为 0

bit 12-8 **TRGSRC29<4:0>**：模拟输入 AN29 转换触发源选择位
关于位值定义，请参见 bit 28-24。

bit 7-5 未实现：读为 0

bit 4-0 **TRGSRC28<4:0>**：模拟输入 AN28 转换触发源选择位
关于位值定义，请参见 bit 28-24。

 © 2015 Microchip Technology Inc.

寄存器 22-26： ADCCMPCON1：ADC 数字比较器 1 控制寄存器

位范围	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	R-0, HS, HC	R-0, HS, HC	R-0, HS, HC	R-0, HS, HC	R-0, HS, HC	R-0, HS, HC	R-0, HS, HC	R-0, HS, HC
	CVDDATA<15:8>							
23:16	R-0, HS, HC	R-0, HS, HC	R-0, HS, HC	R-0, HS, HC	R-0, HS, HC	R-0, HS, HC	R-0, HS, HC	R-0, HS, HC
	CVDDATA<7:0>							
15:8	U-0	U-0	R-0, HS, HC	R-0, HS, HC	R-0, HS, HC	R-0, HS, HC	R-0, HS, HC	R-0, HS, HC
	—	—	AINID<5:0>					
7:0	R/W-0	R/W-0	R-0, HS, HC	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	ENDCMP	DCMPGIEN	DCMPED	IEBTWN	IEHIHI	IEHILO	IELOHI	IELOLO

图注：	HS = 硬件置 1 位	HC = 硬件清零位	
R = 可读位	W = 可写位	U = 未实现位，读为 0	
-n = POR 值	1 = 置 1	0 = 清零	x = 未知

bit 31-16 **CVDDATA<15:0>**：CVD 数据状态位
在 CVD 模式下，每当发生数字比较器事件时，这些位会获取 CVD 差分输出数据 （CVD 正测量值减去负测量值）。这些位中的值遵从 FRACT 位 （ADCCON1<23>），并且总是有符号的。

bit 15-14 未实现：读为 0

bit 13-8 **AINID<5:0>**：数字比较器 1 模拟输入标识 （ID）位
发生数字比较器事件时 （DCMPED = 1），这些位标识数字比较器 1 正在监视的模拟输入。

注： 在正常 ADC 模式下，只有模拟输入 <31:0> 可以由数字比较器 1 处理。数字比较器 1 还支持 CVD 模式，在该模式下可以将 2 类和 3 类模拟输入存储在 AINID<5:0> 位中。

111111 = 正在监视 AN63
111110 = 正在监视 AN62
.
.
.
000001 = 正在监视 AN1
000000 = 正在监视 AN0

bit 7 **ENDCMP**：数字比较器 1 使能位
1 = 使能数字比较器 1
0 = 未使能数字比较器 1，且 DCMPED 状态位 （ADCCMPCON1<5>）清零

bit 6 **DCMPGIEN**：数字比较器 1 中断允许位
1 = 在 DCMPED 状态位 （ADCCMPCON1<5>）置 1 时产生数字比较器 1 中断
0 = 禁止数字比较器 1 中断

bit 5 **DCMPED**：数字比较器 1 “输出真”事件状态位
数字比较器获得 “真”的逻辑条件由 IEBTWN、 IEHIHI、 IEHILO、 IELOHI 和 IELOLO 位定义。

注： 该位通过读取 AINID<5:0> 位或通过禁止数字比较器模块（通过将 ENDCMP 设置为 0）来清零。

1 = 发生了数字比较器 1 输出真事件 （比较器的输出为 1）
0 = 数字比较器 1 输出为假 （比较器的输出为 0）

bit 4 **IEBTWN**：介于下限 / 上限间的数字比较器 1 事件位
1 = DCMPLO<15:0> ≤ DATA<31:0> < DCMPHI<15:0> 时发生数字比较器事件
0 = 不发生数字比较器事件

bit 3 **IEHIHI**：高于上限的数字比较器 1 事件位
1 = DCMPHI<15:0> ≤ DATA<31:0> 时发生数字比较器 1 事件
0 = 不发生事件

bit 2 **IEHILO**：低于上限的数字比较器 1 事件位
1 = DATA<31:0> < DCMPHI<15:0> 时发生数字比较器 1 事件
0 = 不发生事件

寄存器 22-26： ADCCMPCON1：ADC 数字比较器 1 控制寄存器

bit 1 **IELOHI**：高于下限的数字比较器 1 事件位
1 = DCMPLO<15:0> ≤ DATA<31:0> 时发生数字比较器 1 事件
0 = 不发生事件

bit 0 **IELOLO**：低于下限的数字比较器 1 事件位
1 = DATA<31:0> < DCMPLO<15:0> 时发生数字比较器 1 事件
0 = 不发生事件

 © 2015 Microchip Technology Inc.

寄存器 22-27： ADCCMPCONx：ADC 数字比较器 x 控制寄存器 （x = 2 至 6）

位范围	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	—	—	—	—	—	—	—	—
23:16	U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	—	—	—	—	—	—	—	—
15:8	U-0	U-0	U-0	R-0, HS, HC	R-0, HS, HC	R-0, HS, HC	R-0, HS, HC	R-0, HS, HC
	—	—	—	AINID<4:0>				
7:0	R/W-0	R/W-0	R-0, HS, HC	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	ENDCMP	DCMPGIEN	DCMPED	IEBTWN	IEHIHI	IEHILO	IELOHI	IELOLO

图注：	HS = 硬件置 1 位	HC = 硬件清零位	
R = 可读位	W = 可写位	U = 未实现位，读为 0	
-n = POR 值	1 = 置 1	0 = 清零	x = 未知

bit 31-13 **未实现：**读为 0

bit 12-8 **AINID<4:0>：**数字比较器 x 模拟输入标识 （ID）位
发生数字比较器事件时 （DCMPED = 1），这些位标识数字比较器正在监视的模拟输入。

注： 只有模拟输入 <31:0> 可以由数字比较器模块 x （x = 2-6）处理。

11111 = 正在监视 AN31
11110 = 正在监视 AN30
•
•
•
00001 = 正在监视 AN1
00000 = 正在监视 AN0

bit 7 **ENDCMP：**数字比较器 x 使能位
1 = 使能数字比较器 x
0 = 未使能数字比较器 x，且 DCMPED 状态位 （ADCCMPCONx<5>）清零

bit 6 **DCMPGIEN：**数字比较器 x 中断允许位
1 = DCMPED 状态位 （ADCCMPCONx<5>）置 1 时产生数字比较器 x 中断
0 = 禁止数字比较器 x 中断

bit 5 **DCMPED：**数字比较器 x “输出真”事件状态位
数字比较器获得 “真”的逻辑条件由 IEBTWN、 IEHIHI、 IEHILO、 IELOHI 和 IELOLO 位定义。

注： 该位通过读取 AINID<5:0> 位 （ADCCMPCON1<13:8>）或通过禁止数字比较器模块 （通过将 ENDCMP 设置为 0）来清零。

1 = 发生了数字比较器 x 输出真事件 （比较器的输出为 1）
0 = 数字比较器 *x* 输出为假 （比较器的输出为 0）

bit 4 **IEBTWN：**介于下限 / 上限间的数字比较器 x 事件位
1 = DCMPLO<15:0> 位 ≤ DATA<31:0> 位 < DCMPHI<15:0> 位时发生数字比较器事件
0 = 不发生数字比较器事件

bit 3 **IEHIHI：**高于上限的数字比较器 x 事件位
1 = DCMPHI<15:0> 位 ≤ DATA<31:0> 位时发生数字比较器 x 事件
0 = 不发生事件

bit 2 **IEHILO：**低于上限的数字比较器 x 事件位
1 = DATA<31:0> 位 < DCMPHI<15:0> 位时发生数字比较器 x 事件
0 = 不发生事件

寄存器 22-27： ADCCMPCONx：ADC 数字比较器 x 控制寄存器 （x = 2 至 6）（续）

bit 1 **IELOHI**：高于下限的数字比较器 x 事件位

1 = DCMPLO<15:0> 位 ≤ DATA<31:0> 位时发生数字比较器 x 事件
0 = 不发生事件

bit 0 **IELOLO**：低于下限的数字比较器 x 事件位

1 = DATA<31:0> 位 < DCMPLO<15:0> 位时发生数字比较器 x 事件
0 = 不发生事件

 © 2015 Microchip Technology Inc.

寄存器 22-28： ADCFSTAT：ADC FIFO 状态寄存器

位范围	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	FEN	ADC6EN	ADC5EN	ADC4EN	ADC3EN	ADC2EN	ADC1EN	ADC0EN
23:16	R/W-0	R-0, HS, HC	R-0, HS, HC	U-0	U-0	U-0	U-0	U-0
	FIEN	FRDY	FWROVERR	—	—	—	—	—
15:8	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
	FCNT<7:0>							
7:0	R-0	U-0	U-0	U-0	U-0	R-0	R-0	R-0
	FSIGN	—	—	—	—	ADCID<2:0>		

图注：			
R = 可读位	W = 可写位	U = 未实现位，读为 0	
-n = POR 值	1 = 置 1	0 = 清零	x = 未知

bit 31 **FEN**：FIFO 使能位
1 = 使能 FIFO
0 = 禁止 FIFO ；没有数据被保存到 FIFO 中

bit 30-24 **ADC6EN:ADC0EN**：ADCx 使能位 （x = 0 至 6）
1 = 将 ADCx 的转换输出数据存储到 FIFO 中
0 = 不将 ADCx 的转换输出数据存储到 FIFO 中

注： 使用 FIFO 时，输出数据另外还会存储到相应的输出数据寄存器 （ADCDATAx）中。

bit 23 **FIEN**：FIFO 中断允许位
1 = 允许 FIFO 中断；在 FRDY 位置 1 时产生中断
0 = 禁止 FIFO 中断

bit 22 **FRDY**：FIFO 数据就绪中断状态位
1 = FIFO 具有待读取的数据
0 = FIFO 中没有可用数据

注： 读取 ADCFIFO 中的 FIFO 输出数据，并且 FIFO 中没有其他就绪数据 （即， FIFO 为空）时，该位清零。

bit 21 **FWROVERR**：FIFO 写溢出错误状态位
1 = FIFO 中发生写溢出错误 （循环 FIFO）
0 = FIFO 中未发生写溢出错误

注： 用软件读取 ADCFSTAT<23:16> 后，该位清零。

bit 15-8 **FCNT<7:0>**：FIFO 数据条目计数状态位
这些位中的值指示 FIFO 中的数据条目数。

bit 7 **FSIGN**：FIFO 符号设置位
该位反映在 ADCFIFO 寄存器中存储的数据的符号。

bit 6-3 未实现：读为 0

bit 2-0 **ADCID<2:0>**：ADCx 标识符位 （x = 0 至 6）
这些位指定数据存储在 FIFO 中的 ADC 模块。
111 = 保留
110 = ADC6 的转换数据存储在 FIFO 中
•
•
•
000 = ADC0 的转换数据存储在 FIFO 中

寄存器 22-29： ADCFIFO：ADC FIFO 数据寄存器

位范围	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
	DATA<31:24>							
23:16	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
	DATA<23:16>							
15:8	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
	DATA<15:8>							
7:0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
	DATA<7:0>							

图注：			
R = 可读位	W = 可写位	U = 未实现位，读为 0	
-n = POR 值	1 = 置 1	0 = 清零	x = 未知

bit 31-0 **DATA<31:0>**：FIFO 数据输出值位

> **注：** 在使用备用输入作为专用 ADC 模块的输入源时，数据输出仍然从主输入数据输出寄存器中读取。

寄存器 22-30： ADCBASE：ADC 基址寄存器

位范围	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	—	—	—	—	—	—	—	—
23:16	U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	—	—	—	—	—	—	—	—
15:8	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	ADCBASE<15:8>							
7:0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	ADCBASE<7:0>							

图注：			
R = 可读位	W = 可写位	U = 未实现位，读为 0	
-n = POR 值	1 = 置 1	0 = 清零	x = 未知

bit 31-16 **未实现**：读为 0

bit 15-0 **ADCBASE<15:0>**：ADC ISR 基址位

读取该寄存器时，该寄存器包含用户的 ADC ISR 跳转表的基址。中断向量地址由 ADCCON1 寄存器的 IRQVS<2:0> 位决定，这些位指定在与 ADCBASE 寄存器相加之前，对 ADCDSTAT1 和 ADCDSTAT2 寄存器中的 ARDYx 状态位进行左移的移位量。

中断向量地址 = ADCBASE 的读取值 = 写入 ADCBASE 的值 + x << IRQVS<2:0>；其中，“x”是来自 ADCDSTAT1 或 ADCDSTAT2 寄存器的最小有效模拟输入 ID （其优先级最高）。

 © 2015 Microchip Technology Inc.

寄存器 22-31： ADCDATAx：ADC 输出数据寄存器 （x = 0 至 63）

位范围	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
	DATA<31:24>							
23:16	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
	DATA<23:16>							
15:8	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
	DATA<15:8>							
7:0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
	DATA<7:0>							

图注：			
R = 可读位	W = 可写位	U = 未实现位，读为 0	
-n = POR 值	1 = 置 1	0 = 清零	x = 未知

bit 31-0 **DATA<31:0>**：ADC 转换数据输出位

注	
1：	在使用备用输入作为专用 ADC 模块的输入源时，数据输出仍然从主输入数据输出寄存器中读取。
2：	在更改 FRACT 位后读取 ADCDATAx 寄存器值会将数据转换为 FRACT 位所指定的格式。

寄存器 22-32： ADCDMASTAT：ADC DMA 状态寄存器

位范围	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	DMAGEN	RBFIEN6	RBFIEN5	RBFIEN4	RBFIEN3	RBFIEN2	RBFIEN1	RBFIEN0
23:16	R-0, HS, HC	R-0, HS, HC	R-0, HS, HC	R-0, HS, HC	R-0, HS, HC	R-0, HS, HC	R-0, HS, HC	R-0, HS, HC
	DMAWROVERR	RBF6	RBF5	RBF4	RBF3	RBF2	RBF1	RBF0
15:8	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	DMACNTEN	RAFIEN6	RAFIEN5	RAFIEN4	RAFIEN3	RAFIEN2	RAFIEN1	RAFIEN0
7:0	U-0	R-0, HS, HC	R-0, HS, HC	R-0, HS, HC	R-0, HS, HC	R-0, HS, HC	R-0, HS, HC	R-0, HS, HC
	—	RAF6	RAF5	RAF4	RAF3	RAF2	RAF1	RAF0

图注：			
R = 可读位	W = 可写位	U = 未实现位，读为 0	
-n = POR 值	1 = 置 1	0 = 清零	x = 未知

bit 31 **DMAGEN**：DMA 全局使能位

1 = 对所有 ADC 模块全局使能 DMA 引擎。要使用 DMA，各个专用 ADC 模块应使能其 DMAEN 位（ADCxTIME<23>）

0 = 对所有 ADC 模块全局禁止 DMA 引擎

bit 30-24 **RBFIEN6:RBFIEN0**：RAM 缓冲区 B 满 ADC6-ADC0 中断允许位

1 = 允许中断，并在 RBFx 状态位置 1 时产生中断

0 = 禁止中断

bit 23 **DMAWROVERR**：DMA 写溢出错误位

1 = 发生了 DMA 写溢出错误 （循环缓冲区）

0 = 未发生 DMA 写溢出错误

注： 在软件读取 ADCDMASTAT 寄存器后，该位由硬件清零。

bit 22-16 **RBF6:RBF0**：RAM 缓冲区 B 满 （ADC6-ADC0）状态位

1 = RAM 缓冲区 B 已满

0 = RAM 缓冲区 B 未满

注： 用软件读取这些位时，这些位会自清零。

bit 15 **DMACNTEN**：DMA 缓冲区采样计数使能位

DMA 引擎会在每个采样写入 SRAM 中的相应缓冲区后，将每个缓冲区的当前采样计数保存在从 ADCCNTB 地址开始的表中。

bit 14-8 **RAFIEN6:RAFIEN0**：RAM 缓冲区 A 满 （ADC6-ADC0）中断允许位

1 = 允许中断，并在 RAFx 状态位置 1 时产生中断

0 = 禁止中断

bit 7 未实现：读为 0

bit 6-0 **RAF6:RAF0**：RAM 缓冲区 A 满 （ADC6-ADC0）状态位

1 = RAM 缓冲区 A 已满

0 = RAM 缓冲区 A 未满

注： 用软件读取这些位时，这些位会自清零。

 © 2015 Microchip Technology Inc.

寄存器 22-33： **ADCCNTB：ADC 采样计数基址寄存器**

位范围	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	CNTBADDR<31:24>							
23:16	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	CNTBADDR<23:16>							
15:8	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	CNTBADDR<15:8>							
7:0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	CNTBADDR<7:0>							

图注：			
R = 可读位	W = 可写位	U = 未实现位，读为 0	
-n = POR 值	1 = 置 1	0 = 清零	x = 未知

bit 31-0 **CNTBADDR<31:0>**：模数转换计数基址位

ADCCNTB 寄存器包含用户定义的 RAM 地址， DMA 引擎会从该地址开始保存输出采样 （输出采样已经写入每个 ADC 模块在系统 RAM 中的每个缓冲区）的当前计数（如果 DMACNTEN 位（ADCDMASTAT<15>）置 1）。ADCx 模块会将其缓冲区 A 当前采样计数保存在地址 ((ADCCNTB) + 2 * x) 处，将其缓冲区 B 当前采样计数保存在地址 ((ADCCNTB) + (2 * x + 1)) 处。其中，“x”为专用 ADC 模块 ID （0-6）。

寄存器 22-34： **ADCDMAB：ADC DMA 基址寄存器**

位范围	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	DMABADDR<31:24>							
23:16	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	DMABADDR<23:16>							
15:8	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	DMABADDR<15:8>							
7:0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	DMABADDR<7:0>							

图注：			
R = 可读位	W = 可写位	U = 未实现位，读为 0	
-n = POR 值	1 = 置 1	0 = 清零	x = 未知

bit 31-0 **DMABADDR<31:0>**：DMA 基址位

ADCDMAB 寄存器包含用户指定的 RAM 地址， DMA 引擎会从该地址开始保存转换数据。保存每个数据的地址按以下关系指定：

缓冲区 A 起始地址位于：ADCDMAB + (2 * x) * $2^{(ADCON1bits.DMABL + 1)}$

缓冲区 B 起始地址位于：ADCDMAB + (2 * (x + 1)) * $2^{(ADCON1bits.DMABL + 1)}$

其中，“x”为专用 ADC 模块 ID （0-6）。

寄存器 22-35： ADCTRGSNS：ADC 触发电平 / 边沿敏感寄存器

位范围	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	LVL31	LVL30	LVL29	LVL28	LVL27	LVL26	LVL25	LVL24
23:16	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	LVL23	LVL22	LVL21	LVL20	LVL19	LVL18	LVL17	LVL16
15:8	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	LVL15	LVL14	LVL13	LVL12	LVL11	LVL10	LVL9	LVL8
7:0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	LVL7	LVL6	LVL5	LVL4	LVL3	LVL2	LVL1	LVL0

图注：

R = 可读位	W = 可写位	U = 未实现位，读为 0	
-n = POR 值	1 = 置 1	0 = 清零	x = 未知

bit 31-0 **LVL31:LVL0**：触发电平和边沿敏感位

1 = 模拟输入对其触发信号的高电平敏感 （电平敏感性意味着只要触发信号保持高电平就会重新触发）
0 = 模拟输入对于其触发信号的正边沿敏感 （这是复位后的值）

注 **1**： 该寄存器指定模拟输入 0 至 31 的触发电平。
2： 编号更高的模拟输入 ID 属于 3 类输入，因此仅通过扫描触发。所有 3 类模拟输入都使用扫描触发，其电平 / 边沿由 STRGLVL 位 （ADCCON1<3>）定义。

 © 2015 Microchip Technology Inc.

寄存器 22-36： ADCxTIME：专用 ADCx 时序寄存器 x （x = 0 至 6）

位范围	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-1	R/W-1
	—	—	—	ADCEIS<2:0>			SELRES<1:0>	
23:16	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	DMAEN	ADCDIV<6:0>						
15:8	U-0	U-0	U-0	U-0	U-0	U-0	R/W-0	R/W-0
	—	—	—	—	—	—	SAMC<9:8>	
7:0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	SAMC<7:0>							

图注：			
R = 可读位	W = 可写位	U = 未实现位，读为 0	
-n = POR 值	1 = 置 1	0 = 清零	x = 未知

bit 31-29 **未实现：**读为 0

bit 28-26 **ADCEIS<2:0>：**ADCx 提前中断选择位

111 = 在转换结束之前的 8 个 ADC 时钟处产生数据就绪中断
110 = 在转换结束之前的 7 个 ADC 时钟处产生数据就绪中断
•
•
•
001 = 在转换结束之前的 2 个 ADC 时钟处产生数据就绪中断
000 = 在转换结束之前的 1 个 ADC 时钟处产生数据就绪中断

注： 在选定分辨率 （由 SELRES<1:0> 位 （ADCxTIME<25:24>）指定）为 12 位或 10 位时，所有选项都可用。选定分辨率为 8 位时，从 000 到 101 的选项是有效的。选定分辨率为 6 位时，从 000 到 011 的选项是有效的。

bit 25-24 **SELRES<1:0>：**ADCx 分辨率选择位

11 = 12 位
10 = 10 位
01 = 8 位
00 = 6 位

bit 23 **DMAEN：**DMA 使能位

1 = 对 ADC 模块使能 DMA 引擎。 ADC 可以将数据保存在系统 RAM 中。
0 = 对 ADC 模块禁止 DMA 引擎。转换数据只能从相应的数据寄存器中获取。

bit 22-16 **ADCDIV<6:0>：**ADCx 时钟分频比位

这些位用于对周期为 TQ 的 ADC 控制时钟进行分频，产生用于 ADCx 的时钟 （TADx）。

1111111 = 254 * TQ = TADx
•
•
•
0000011 = 6 * TQ = TADx
0000010 = 4 * TQ = TADx
0000001 = 2 * TQ = TADx
0000000 = 保留

bit 15-10 **未实现：**读为 0

bit 9-0 **SAMC<9:0>：**ADCx 采样时间位

其中， TADx = 专用 ADC 的 ADC 转换时钟周期，由 ADCDIV<6:0> 位控制。

1111111111 = 1025 TADx
•
•
•
0000000001 = 3 TADx
0000000000 = 2 TADx

寄存器 22-37： ADCEIEN1：ADC 提前中断允许寄存器 1

位范围	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	EIEN31	EIEN30	EIEN29	EIEN28	EIEN27	EIEN26	EIEN25	EIEN24
23:16	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	EIEN23	EIEN22	EIEN21	EIEN20	EIEN19	EIEN18	EIEN17	EIEN16
15:8	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	EIEN15	EIEN14	EIEN13	EIEN12	EIEN11	EIEN10	EIEN9	EIEN8
7:0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	EIEN7	EIEN6	EIEN5	EIEN4	EIEN3	EIEN2	EIEN1	EIEN0

图注：	HS = 硬件置 1 位	C = 可清零位	
R = 可读位	W = 可写位	U = 未实现位，读为 0	
-n = POR 值	1 = 置 1	0 = 清零	x = 未知

bit 31-0 **EIEN31:EIEN0**：模拟输入提前中断允许位

1 = 允许选定模拟输入的提前中断。在发生提前中断事件 （通过 ADCEISTAT1 寄存器的 EIRDYx 位（x = 31-0）指示）后产生中断

0 = 禁止中断

寄存器 22-38： ADCEIEN2：ADC 提前中断允许寄存器 2

位范围	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	EIEN63	EIEN62	EIEN61	EIEN60	EIEN59	EIEN58	EIEN57	EIEN56
23:16	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	EIEN55	EIEN54	EIEN53	EIEN52	EIEN51	EIEN50	EIEN49	EIEN48
15:8	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	EIEN47	EIEN46	EIEN45	EIEN44	EIEN43	EIEN42	EIEN41	EIEN40
7:0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	EIEN39	EIEN38	EIEN37	EIEN36	EIEN35	EIEN34	EIEN33	EIEN32

图注：	HS = 硬件置 1 位	C = 可清零位	
R = 可读位	W = 可写位	U = 未实现位，读为 0	
-n = POR 值	1 = 置 1	0 = 清零	x = 未知

bit 31-0 **EIEN63:EIEN32**：模拟输入提前中断允许位

1 = 允许选定模拟输入的提前中断。在发生提前中断事件 （通过 ADCEISTAT2 寄存器的 EIRDYx 位（x = 63-32）指示）后产生中断

0 = 禁止中断

 © 2015 Microchip Technology Inc.

寄存器 22-39： ADCEISTAT1：ADC 提前中断状态寄存器 1

位范围	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	R-0, HS, HC	R-0, HS, HC	R-0, HS, HC	R-0, HS, HC	R-0, HS, HC	R-0, HS, HC	R-0, HS, HC	R-0, HS, HC
	EIRDY31	EIRDY30	EIRDY29	EIRDY28	EIRDY27	EIRDY26	EIRDY25	EIRDY24
23:16	R-0, HS, HC	R-0, HS, HC	R-0, HS, HC	R-0, HS, HC	R-0, HS, HC	R-0, HS, HC	R-0, HS, HC	R-0, HS, HC
	EIRDY23	EIRDY22	EIRDY21	EIRDY20	EIRDY19	EIRDY18	EIRDY17	EIRDY16
15:8	R-0, HS, HC	R-0, HS, HC	R-0, HS, HC	R-0, HS, HC	R-0, HS, HC	R-0, HS, HC	R-0, HS, HC	R-0, HS, HC
	EIRDY15	EIRDY14	EIRDY13	EIRDY12	EIRDY11	EIRDY10	EIRDY9	EIRDY8
7:0	R-0, HS, HC	R-0, HS, HC	R-0, HS, HC	R-0, HS, HC	R-0, HS, HC	R-0, HS, HC	R-0, HS, HC	R-0, HS, HC
	EIRDY7	EIRDY6	EIRDY5	EIRDY4	EIRDY3	EIRDY2	EIRDY1	EIRDY0

图注：	HS = 硬件置 1 位	HC = 硬件清零位	
R = 可读位	W = 可写位	U = 未实现位，读为 0	
-n = POR 值	1 = 置 1	0 = 清零	x = 未知

bit 31-0 **EIRDY31:EIRDY0**：相应模拟输入就绪提前中断位

1 = 发生指定模拟输入的提前中断事件时，该位会置 1。如果在 ADCEIEN1 寄存器中允许提前中断，则会产生中断。对于 1 类模拟输入，该位将根据 ADCxTIME 寄存器中 ADCEIS<2:0> 位的配置置 1。对于共用 ADC 模块，该位将根据 ADCCON2 寄存器中 ADCEIS<2:0> 位的配置置 1。

0 = 禁止中断

寄存器 22-40： ADCEISTAT2：ADC 提前中断状态寄存器 2

位范围	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	R-0, HS, HC	R-0, HS, HC	R-0, HS, HC	R-0, HS, HC	R-0, HS, HC	R-0, HS, HC	R-0, HS, HC	R-0, HS, HC
	EIRDY63	EIRDY62	EIRDY61	EIRDY60	EIRDY59	EIRDY58	EIRDY57	EIRDY56
23:16	R-0, HS, HC	R-0, HS, HC	R-0, HS, HC	R-0, HS, HC	R-0, HS, HC	R-0, HS, HC	R-0, HS, HC	R-0, HS, HC
	EIRDY55	EIRDY54	EIRDY53	EIRDY52	EIRDY51	EIRDY50	EIRDY49	EIRDY48
15:8	R-0, HS, HC	R-0, HS, HC	R-0, HS, HC	R-0, HS, HC	R-0, HS, HC	R-0, HS, HC	R-0, HS, HC	R-0, HS, HC
	EIRDY47	EIRDY46	EIRDY45	EIRDY44	EIRDY43	EIRDY42	EIRDY41	EIRDY40
7:0	R-0, HS, HC	R-0, HS, HC	R-0, HS, HC	R-0, HS, HC	R-0, HS, HC	R-0, HS, HC	R-0, HS, HC	R-0, HS, HC
	EIRDY39	EIRDY38	EIRDY37	EIRDY36	EIRDY35	EIRDY34	EIRDY33	EIRDY32

图注：	HS = 硬件置 1 位	HC = 硬件清零位	
R = 可读位	W = 可写位	U = 未实现位，读为 0	
-n = POR 值	1 = 置 1	0 = 清零	x = 未知

bit 31-0 **EIRDY63:EIRDY32**：相应模拟输入就绪提前中断位

1 = 发生指定模拟输入的提前中断事件时，该位会置 1。如果在 ADCEIEN2 寄存器中允许提前中断，则会产生中断。对于 1 类模拟输入，该位将根据 ADCxTIME 寄存器中 ADCEIS<2:0> 位的配置置 1。对于共用 ADC 模块，该位将根据 ADCCON2 寄存器中 ADCEIS<2:0> 位的配置置 1。

0 = 禁止中断

寄存器 22-41： ADCANCON：ADC 模拟预热控制寄存器

位范围	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	U-0	U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0
	—	—	—	—	WKUPCLKCNT<3:0>			
23:16	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	WKIEN7	WKIEN6	WKIEN5	WKIEN4	WKIEN3	WKIEN2	WKIEN1	WKIEN0
15:8	R-0, HS, HC	R-0, HS, HC	R-0, HS, HC	R-0, HS, HC	R-0, HS, HC	R-0, HS, HC	R-0, HS, HC	R-0, HS, HC
	WKRDY7	WKRDY6	WKRDY5	WKRDY4	WKRDY3	WKRDY2	WKRDY1	WKRDY0
7:0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	ANEN7	ANEN6	ANEN5	ANEN4	ANEN3	ANEN2	ANEN1	ANEN0

图注：	HS = 硬件置 1 位	HC = 软件清零位	
R = 可读位	W = 可写位	U = 未实现位，读为 0	
-n = POR 值	1 = 置 1	0 = 清零	x = 未知

bit 31-28 **未实现：**读为 0

bit 27-24 **WKUPCLKCNT<3:0>：**唤醒时钟计数位

这些位代表在 ADC 模块可以执行转换之前进行预热所需的 ADC 时钟数。虽然每个 ADC 的时钟是特定的，但 WKUPCLKCNT 位是所有 ADC 模块共用的。

1111 = 2^{15} = 32,768 个时钟
•
•
•
0110 = 2^6 = 64 个时钟
0101 = 2^5 = 32 个时钟
0100 = 2^4 = 16 个时钟
0011 = 2^4 = 16 个时钟
0010 = 2^4 = 16 个时钟
0001 = 2^4 = 16 个时钟
0000 = 2^4 = 16 个时钟

bit 23 **WKIEN7：**共用 ADC （ADC7）唤醒中断允许位

1 = 允许中断，并在 WKRDY7 状态位置 1 时产生中断
0 = 禁止中断

bit 22-16 **WKIEN6:WKIEN0：**ADC6-ADC0 唤醒中断允许位

1 = 允许中断，并在 WKRDYx 状态位置 1 时产生中断
0 = 禁止中断

bit 15 **WKRDY7：**共用 ADC （ADC7）唤醒状态位

1 = 在 ANEN7 设置为 1 之后的唤醒计数 （$2^{WKUPEXP}$）个时钟后， ADC7 模拟和偏置电路准备就绪
0 = ADC7 模拟和偏置电路未就绪

注： 当 ANEN7 位清零时，该位由硬件清零

bit 14-8 **WKRDY6:WKRDY0：**ADC6-ADC0 唤醒状态位

1 = 在 ANEN*x* 设置为 1 之后的唤醒计数 （$2^{WKUPEXP}$）个时钟后， ADCx 模拟和偏置电路准备就绪
0 = ADCx 模拟和偏置电路未就绪

注： 当 ANEN*x* 位清零时，这些位由硬件清零

bit 7 **ANEN7：**共用 ADC （ADC7）模拟和偏置电路使能位

1 = 使能模拟和偏置电路。使能模拟和偏置电路之后， ADC 模块需要一段预热时间，该时间由 WKUPCLKCNT<3:0> 位定义。
0 = 禁止模拟和偏置电路

bit 6-0 **ANEN6:ANEN0：**ADC6-ADC0 模拟和偏置电路使能位

1 = 使能模拟和偏置电路。使能模拟和偏置电路之后， ADC 模块需要一段预热时间，该时间由 WKUPCLKCNT<3:0> 位定义。
0 = 禁止模拟和偏置电路

 © 2015 Microchip Technology Inc.

寄存器 22-42： ADCxCFG：ADCx 配置寄存器 x （x = 0 至 7）

位范围	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	ADCCFG<31:24>							
23:16	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	ADCCFG<23:16>							
15:8	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	ADCCFG<15:8>							
7:0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	ADCCFG<7:0>							

图注：			
R = 可读位	W = 可写位	U = 未实现位，读为 0	
-n = POR 值	1 = 置 1	0 = 清零	x = 未知

bit 31-0 **ADCCFG<31:0>**：ADC 模块配置数据位

注： 只有 ADCANCON 寄存器中适用的 ANENx 位清零时，这些位才能发生更改。

寄存器 22-43： ADCSYSCFG0：ADC 系统配置寄存器 0

位范围	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	HC, HS, R-0	HC, HS, R-0	HC, HS, R-0	HC, HS, R-0	HC, HS, R-0	HC, HS, R-0	HC, HS, R-0	HC, HS, R-0
	AN<31:23>							
23:16	HC, HS, R-0	HC, HS, R-0	HC, HS, R-0	HC, HS, R-0	HC, HS, R-0	HC, HS, R-0	HC, HS, R-0	HC, HS, R-0
	AN<23:16>							
15:8	HC, HS, R-0	HC, HS, R-0	HC, HS, R-0	HC, HS, R-0	HC, HS, R-0	HC, HS, R-0	HC, HS, R-0	HC, HS, R-0
	AN<15:8>							
7:0	HC, HS, R-0	HC, HS, R-0	HC, HS, R-0	HC, HS, R-0	HC, HS, R-0	HC, HS, R-0	HC, HS, R-0	HC, HS, R-0
	AN<7:0>							

图注：	HS = 硬件置 1 位	HC = 软件清零位	
R = 可读位	W = 可写位	U = 未实现位，读为 0	
-n = POR 值	1 = 置 1	0 = 清零	x = 未知

bit 31-0 **AN<31:0>**：ADC 模拟输入位

这些位反映系统配置，它们在启动期间发生更新。通过读取这些只读位，用户应用程序可以确定器件中的某个模拟输入是否可用。

AN<31:0>：反映是否存在相应的模拟输入 （AN31-AN0）。

寄存器 22-44： ADCSYSCFG1：ADC 系统配置寄存器 1

位范围	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	HC, HS, R-0	HC, HS, R-0	HC, HS, R-0	HC, HS, R-0	HC, HS, R-0	HC, HS, R-0	HC, HS, R-0	HC, HS, R-0
	AN<63:56>							
23:16	HC, HS, R-0	HC, HS, R-0	HC, HS, R-0	HC, HS, R-0	HC, HS, R-0	HC, HS, R-0	HC, HS, R-0	HC, HS, R-0
	AN<55:48>							
15:8	HC, HS, R-0	HC, HS, R-0	HC, HS, R-0	HC, HS, R-0	HC, HS, R-0	HC, HS, R-0	HC, HS, R-0	HC, HS, R-0
	AN<47:40>							
7:0	HC, HS, R-0	HC, HS, R-0	HC, HS, R-0	HC, HS, R-0	HC, HS, R-0	HC, HS, R-0	HC, HS, R-0	HC, HS, R-0
	AN<39:32>							

图注：	HS = 硬件置 1 位	HC = 软件清零位	
R = 可读位	W = 可写位	U = 未实现位，读为 0	
-n = POR 值	1 = 置 1	0 = 清零	x = 未知

bit 31-0 **AN<63:32>**：ADC 模拟输入位

这些位反映系统配置，它们在启动期间发生更新。通过读取这些只读位，用户应用程序可以确定器件中的某个模拟输入是否可用。

AN<63:32>：反映是否存在相应的模拟输入 （AN63-AN32）。

 © 2015 Microchip Technology Inc.

22.3 ADC 工作原理

高速逐次逼近寄存器（SAR）ADC 用于支持电源转换和电机控制应用，最多包含 8 个独立 ADC 模块。ADC 模块 ADC0 至 ADC6 可将单个模拟输入（进行备用输入选择之后）与其 S&H 电路连接。由于这些 ADC 模块用于对专用模拟输入进行采样，所以将它们称为“专用”ADC 模块。专用 ADC 模块用于测量 / 捕捉时间敏感或瞬变模拟信号。ADC7 模块可通过多路开关将多个模拟输入与其 S&H 电路连接。由于多个模拟输入共用该 ADC，所以将它称为“共用” ADC 模块。共用ADC模块用于测量频率较低的模拟信号，以及具有静态性质（即，不随时间显著变化）的信号。

与专用 ADC 模块（ADC0-ADC6）连接的模拟输入是 1 类输入。与共用 ADC 模块（ADC7）连接的模拟输入是 2 类和 3 类输入。为每个“类”指定的输入的数量取决于具体的器件。例如，对于具有 8 个 ADC 模块和 54 个模拟输入的器件，其安排如下：

- 1 类 = AN0 至 AN6
- 2 类 = AN7 至 AN31
- 3 类 = AN32 至 AN53

关于每个类的模拟输入的实际数量，请参见具体器件数据手册的 **“ADC”** 章节。

表 22-2 说明了每个模拟输入类的属性：

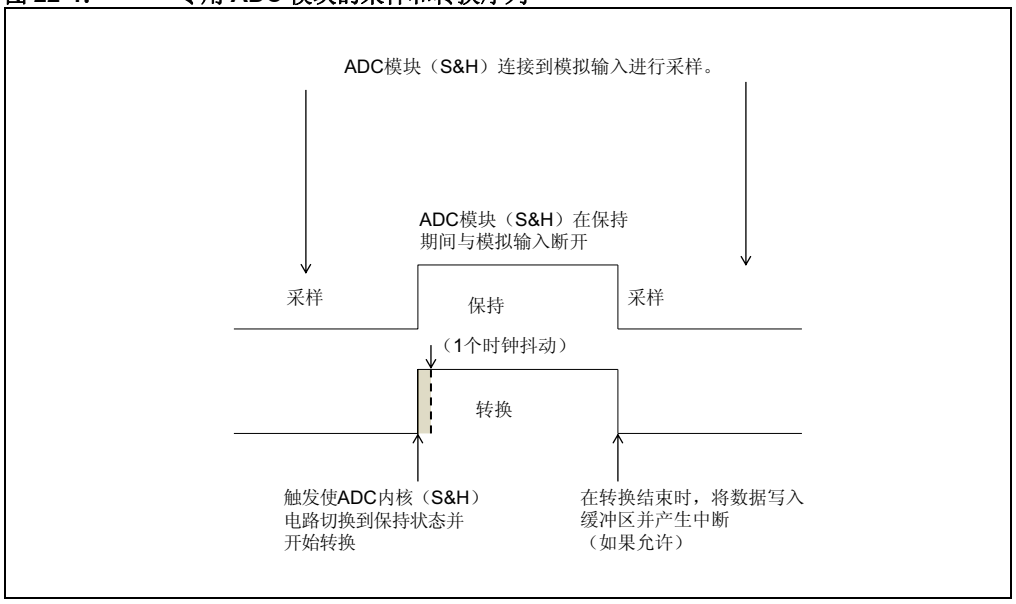
表 22-2：　模拟输入类

ADC 模块	模拟输入类	触发信号	触发操作
ADC0 至 ADC6（专用 ADC 模块）	1 类	独立触发源或扫描触发	结束采样并启动转换
ADC7（共用 ADC 模块）	2 类	独立触发源或扫描触发	启动采样序列或开始扫描序列
ADC7（共用 ADC 模块与输入扫描）	3 类	扫描触发	启动扫描序列

1 类模拟输入属性：

- 1 类输入与专用 ADC 模块相关联。在某一给定时间，每个专用 ADC 只有单个 1 类输入与它相关联。（备用）输入通过 ADCTRGMODE 寄存器中的 SHxALT<1:0> 位来进行选择。无论备用输入选择如何，触发源和结果寄存器都保持不变。
- 每个 1 类输入具有一个独有触发源（通过 ADCTRGx 寄存器进行选择），它们在触发信号到达时结束采样并启动转换。在转换完成时， ADC 模块会恢复为采样模式。如果使能某个 1 类输入，但不对它进行转换，则将总是对它进行采样。
- 所有 1 类输入的优先级相同，它们独立工作，并且可以在同一时间在所有 1 类输入上启动转换
- 1 类输入可以作为扫描列表的一部分，通过公共扫描触发源触发。

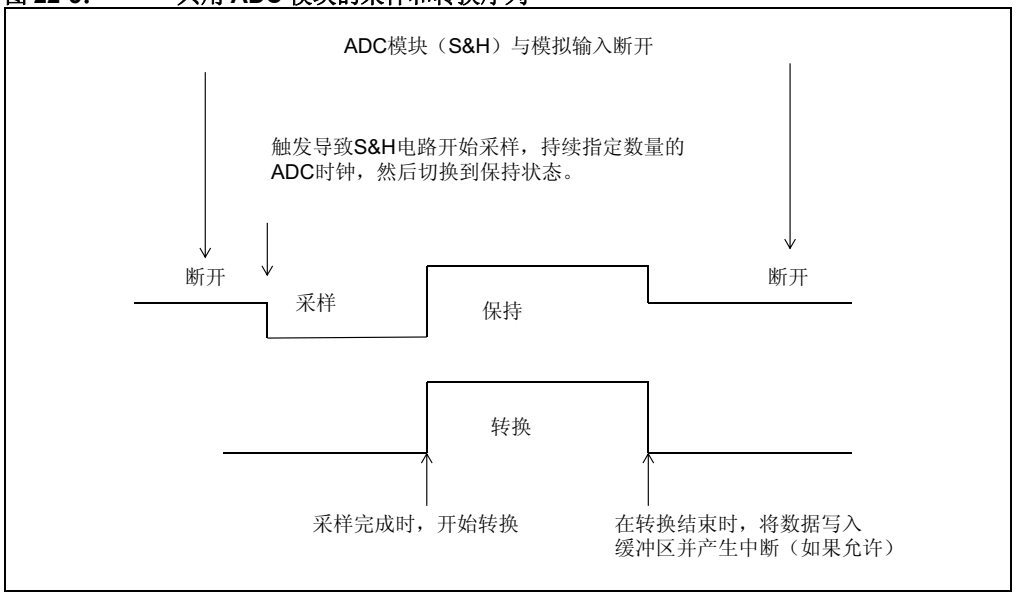
图 22-4： 专用 ADC 模块的采样和转换序列



2 类和 3 类模拟输入属性：

- 2 类输入在共用 ADC 模块上使用，单独触发或作为扫描列表中的一部分。单独使用时，它们通过由 ADCTRGx 寄存器选择的独有触发源触发。
- 共用 ADC 上的模拟输入采用自然优先级顺序 （AN7 的优先级最高， AN63 的优先级最低）
- 3 类输入专门用于扫描，共用一个公共触发源 （扫描触发）
- 由于 3 类模拟输入同时共用 ADC 模块和触发源，所以对它们进行转换的唯一方法是对于每个传入的扫描触发事件按顺序对它们进行扫描 （扫描按自然优先级顺序进行）
- 不同于 1 类模拟输入，共用 ADC 模块中的触发信号到达时只会启动采样。当触发信号到达时， ADC 模块会进入采样模式，其采样时间由 SAMC<9:0> 位 （ADCCON2<25:16>）决定。在采样结束时， ADC 会启动转换。在转换完成时，将按照自然优先级顺序，使用 ADC 模块来转换下一个排队的 2 类或 3 类输入。当共用模拟输入 （2 类或 3 类）完成所有转换，并且没有待处理的触发信号时， ADC 模块会与所有模拟输入断开。

图 22-5： 共用 ADC 模块的采样和转换序列

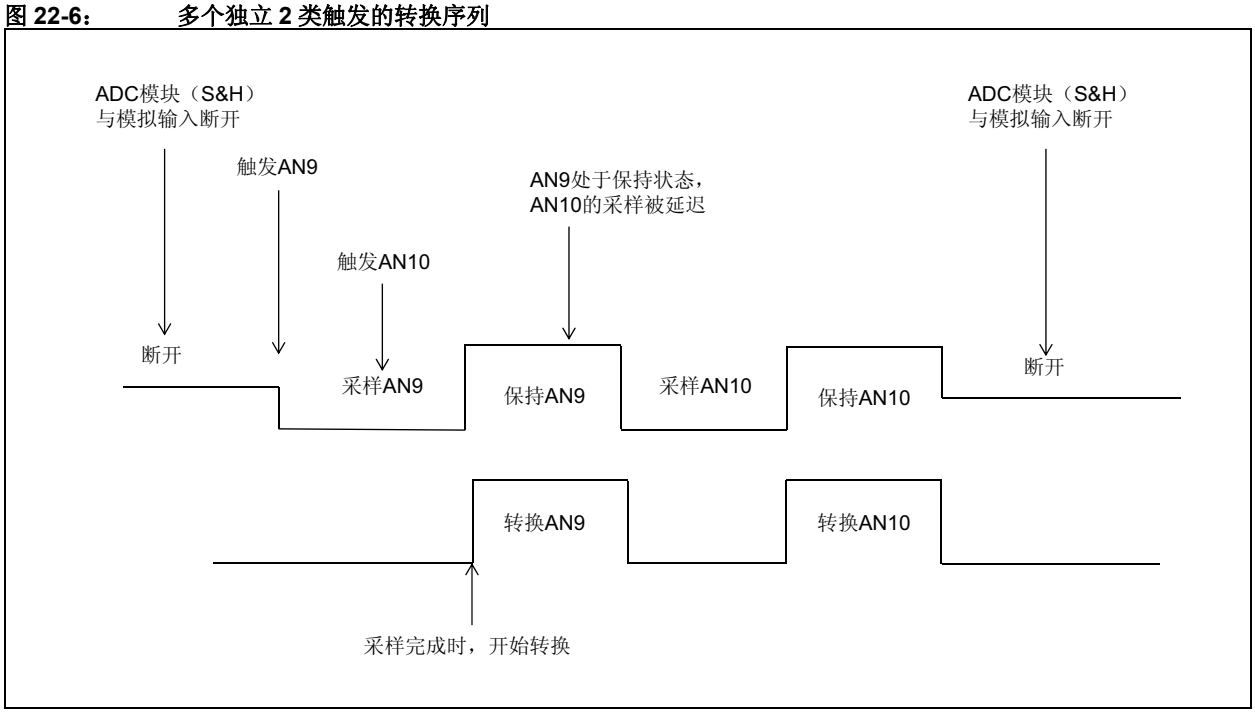


 © 2015 Microchip Technology Inc.

22.3.1 2 类触发

在触发单个 2 类输入时，会按照图 22-5 所示的序列通过共用 S&H 对它进行采样和转换。在触发多个 2 类输入时，了解触发时序所产生的结果是非常重要的。如果正在进行转换，并且又出现另一个 2 类触发信号，则对应于新触发信号的采样 - 保持 - 转换会被延后，直到正在进行的采样 - 保持周期完成为止，如图 22-6 所示。

图 22-6： 多个独立 2 类触发的转换序列

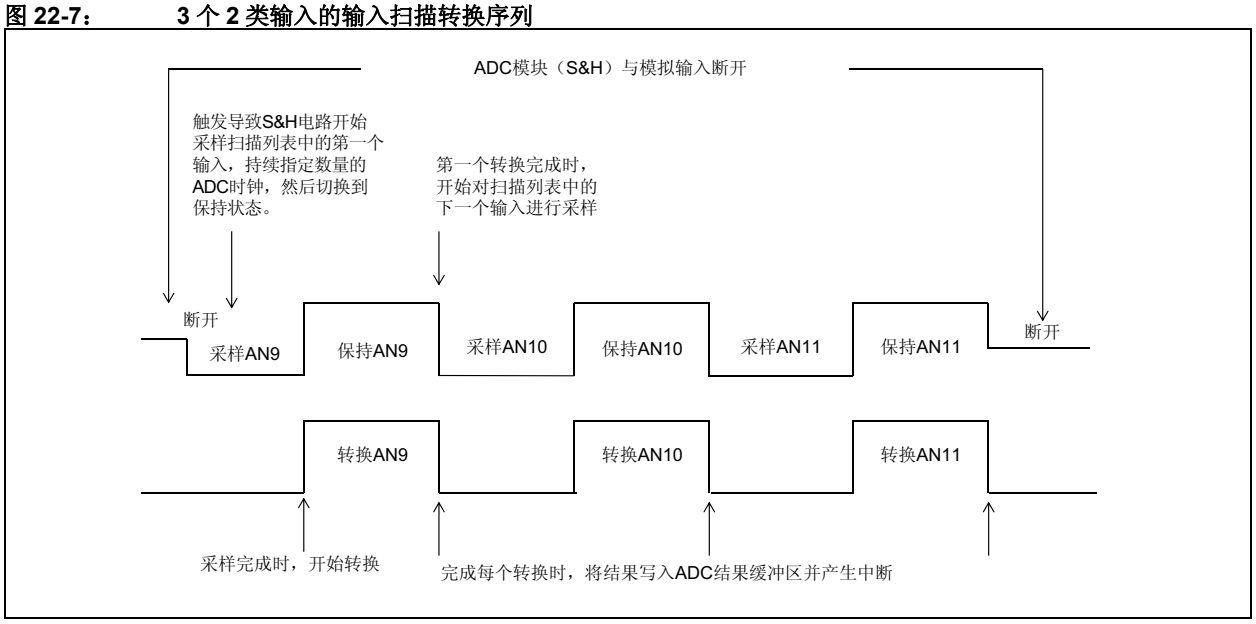


当共用 S&H 的多个输入同时被触发时，处理顺序由它们的自然优先级决定 （编号最低的输入具有最高优先级）。举例来说，如果 AN9、AN10 和 AN11 同时被触发，将先对 AN9 进行采样和转换，然后是 AN10，最后是 AN11。在共用 S&H 上使用独立 2 类触发时， SAMC<9:0> 位（ADCCON2<25:16>）决定所有输入的采样时间，而 ADCTRGx 寄存器中相应的 TRGSRC<4:0> 位 （见寄存器 22-18）决定每个输入的触发源。

22.3.2 输入扫描

输入扫描是一种用于对多个 1 类、 2 类或 3 类输入执行自动扫描序列的功能。所有 2 类和 3 类输入使用单个共用 S&H 进行扫描。1 类输入使用其专用 ADC 模块上的专用 S&H 进行扫描。用于扫描的模拟输入通过 ADCCSS1 和 ADCCSS2 寄存器的 CSSx 位进行选择。 1 类和 2 类输入通过在 ADCTRGx 寄存器中选择 STRIG 来进行触发， 3 类输入使用 ADCCON1<20:16> 寄存器的 STRGSRC<4:0> 进行触发。发生触发时，会同时捕捉所有 1 类输入并同时启动转换。对于 2 类或 3 类输入，采样和转换按自然输入顺序进行；编号较低的输入在编号较高的输入之前进行采样。

图 22-7： 3 个 2 类输入的输入扫描转换序列

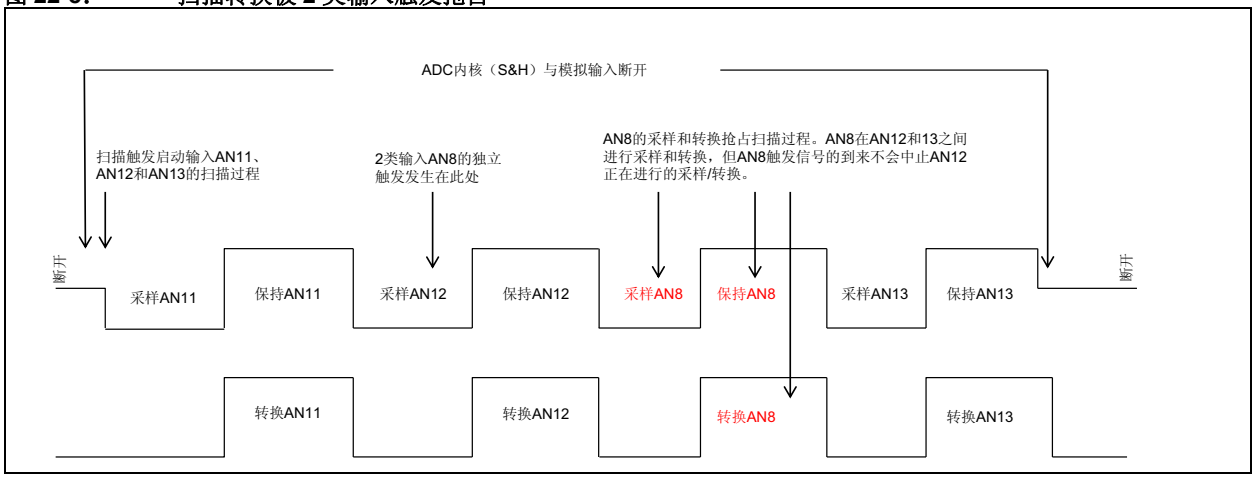


在扫描模式下使用共用模拟输入时，ADC 控制寄存器 2（ADCCON2<25:16>）中的 SAMC<9:0> 位决定所有输入的采样时间，而 ADC 控制寄存器 1 （ADCCON1<20:16>）中的扫描触发源选择位 （STRGSRC<4:0>）决定触发源。

为了确保可预测的结果，应在所有输入都已完成采样之后再重新触发扫描。在系统设计中应注意防止在正在进行扫描时重新触发扫描。

在扫描期间发生独立 2 类触发时，如果它们的优先级高于当前正在处理的采样，则它们会抢先扫描序列。在图 22-8 中，在 2 类输入 AN8 发生独立触发时，AN11、AN12 和 AN13 的扫描正在进行。扫描会被中断，以便对 AN8 进行采样和转换。

图 22-8： 扫描转换被 2 类输入触发抢占



 © 2015 Microchip Technology Inc.

22.4 ADC 模块配置

ADC 模块的操作通过特定寄存器中的位设置来指定。以下说明总结了操作和设置。每个配置步骤的选项和详细信息将在后面的章节中提供。

要配置 ADC 模块，请执行以下步骤：

1. 配置模拟端口引脚，如**第 22.4.1 节 “配置模拟端口引脚”**所述。
2. 选择 ADC 多路开关的模拟输入，如**第 22.4.2 节 “选择 ADC 多路开关模拟输入”**所述。
3. 选择 ADC 结果的格式，如**第 22.4.3 节 “选择 ADC 结果的格式”**所述。
4. 选择转换触发源，如**第 22.4.4 节 “选择转换触发源”**所述。
5. 选择参考电压源，如**第 22.4.5 节 “选择参考电压源”**所述。
6. 选择扫描的输入，如**第 22.4.6 节 “选择扫描的输入”**所述。
7. 选择模数转换时钟源和预分频比，如**第 22.4.7 节“选择模数转换时钟源和预分频比”**所述。
8. 指定任何额外的采集时间 （如需要），如**第 22.10 节 “ADC 采样要求”**所述。
9. 开启 ADC 模块，如**第 22.4.9 节 “开启 ADC”**所述。
10. 通过查询（或等待中断）确定参考电压是否就绪，如**第 22.4.5 节“选择参考电压源”**所述。
11. 使能所需 ADC 模块的模拟和偏置电路，并在 ADC 模块唤醒后使能数字电路，如**第 22.7.3 节 “ADC 低功耗模式”**所述
12. 配置 ADC 中断 （如需要），如**第 22.6 节 “中断”**所述。

22.4.1 配置模拟端口引脚

与模拟输入关联的 I/O 端口的 ANSELx 寄存器用于将相应引脚配置为模拟或数字引脚。当相应的 ANSELx 位 = 1 时，引脚配置为模拟输入。当 ANSELx 位 = 0 时，引脚设置为数字控制。ANSELx 寄存器在器件退出复位时设置，默认情况下会导致 ADC 输入引脚被配置为模拟输入。

TRISx 寄存器控制端口引脚的数字功能。如果要将端口引脚用作模拟输入，则必须在特定 TRISx 寄存器将其相应的位置 1，从而将引脚配置为输入。如果通过清零 TRISx 位将与 ADC 输入关联的 I/O 引脚配置为输出，则会对端口的数字输出电平 （V_{OH} 或 V_{OL}）进行转换。在器件复位后，所有 TRISx 位都会置 1。关于端口引脚配置的详细信息，请参见具体器件数据手册中的 **“I/O 端口”**章节。

注：	当读取与 ADC 共用引脚的 PORT 寄存器时，任何配置为模拟输入的引脚在读取 PORT 锁存器时都读为 0。对于任何定义为数字输入但未配置为模拟输入的引脚，加在引脚上的模拟电压可能导致输入缓冲器消耗的电流超出器件规范。

例 22-1: 初始化并使用 **ADC 1** 类输入

```
int main(int argc, char** argv) {
int result[3];

    /* Configure ADCCON1 */
    ADCCON1 = 0;   // No ADCCON1 features are enabled including: Stop-in-Idle, turbo,
                   // CVD mode, Fractional mode and scan trigger source.

    /* Configure ADCCON2 */
    ADCCON2 = 0;   // Since, we are using only the Class 1 inputs, no setting is
                   // required for ADCDIV

    /* Initialize warm up time register */
    ADCANCON = 0;
    ADCANCONbits.WKUPCLKCNT = 5; // Wakeup exponent = 32 * TADx

    /* Clock setting */
    ADCCON3 = 0;
    ADCCON3bits.ADCSEL = 0;       // Select input clock source
    ADCCON3bits.CONCLKDIV = 1;    // Control clock frequency is half of input clock
    ADCCON3bits.VREFSEL = 0;      // Select AVdd and AVss as reference source

    /* Select ADC sample time and conversion clock */
    ADC0TIMEbits.ADCDIV = 1;      // ADC0 clock frequency is half of control clock = TAD0
    ADC0TIMEbits.SAMC = 5;        // ADC0 sampling time = 5 * TAD0
    ADC0TIMEbits.SELRES = 3;      // ADC0 resolution is 12 bits
    ADC1TIMEbits.ADCDIV = 1;      // ADC1 clock frequency is half of control clock = TAD1
    ADC1TIMEbits.SAMC = 5;        // ADC1 sampling time = 5 * TAD1
    ADC1TIMEbits.SELRES = 3;      // ADC1 resolution is 12 bits
    ADC2TIMEbits.ADCDIV = 1;      // ADC2 clock frequency is half of control clock = TAD2
    ADC2TIMEbits.SAMC = 5;        // ADC2 sampling time = 5 * TAD2
    ADC2TIMEbits.SELRES = 3;      // ADC2 resolution is 12 bits

    /* Select analog input for ADC modules, no presync trigger, not sync sampling */
    ADCTRGMODEbits.SH0ALT = 0;    // ADC0 = AN0
    ADCTRGMODEbits.SH1ALT = 0;    // ADC1 = AN1
    ADCTRGMODEbits.SH2ALT = 0;    // ADC2 = AN2

    /* Select ADC input mode */
    ADCIMCON1bits.SIGN0 = 0;      // unsigned data format
    ADCIMCON1bits.DIFF0 = 0;      // Single ended mode
    ADCIMCON1bits.SIGN1 = 0;      // unsigned data format
    ADCIMCON1bits.DIFF1 = 0;      // Single ended mode
    ADCIMCON1bits.SIGN2 = 0;      // unsigned data format
    ADCIMCON1bits.DIFF2 = 0;      // Single ended mode

    /* Configure ADCGIRQENx */
    ADCGIRQEN1 = 0;               // No interrupts are used
    ADCGIRQEN2 = 0;

    /* Configure ADCCSSx */
    ADCCSS1 = 0;                  // No scanning is used
    ADCCSS2 = 0;

    /* Configure ADCCMPCONx */
    ADCCMPCON1 = 0;               // No digital comparators are used. Setting the ADCCMPCONx
    ADCCMPCON2 = 0;               // register to '0' ensures that the comparator is disabled.
    ADCCMPCON3 = 0;               // Other registers are “don't care” .
    ADCCMPCON4 = 0;
    ADCCMPCON5 = 0;
    ADCCMPCON6 = 0;

    /* Configure ADCFLTRx */
    ADCFLTR1 = 0;                 // No oversampling filters are used.
    ADCFLTR2 = 0;
    ADCFLTR3 = 0;
    ADCFLTR4 = 0;
    ADCFLTR5 = 0;
    ADCFLTR6 = 0;
```

 © 2015 Microchip Technology Inc.

例 22-1：　初始化并使用 ADC 1 类输入 （续）

```
    /* Set up the trigger sources */
    ADCTRGSNSbits.LVL0 = 0;       // Edge trigger
    ADCTRGSNSbits.LVL1 = 0;       // Edge trigger
    ADCTRGSNSbits.LVL2 = 0;       // Edge trigger
    ADCTRG1bits.TRGSRC0 = 1;      // Set AN0 to trigger from software.
    ADCTRG1bits.TRGSRC1 = 1;      // Set AN1 to trigger from software.
    ADCTRG1bits.TRGSRC2 = 1;      // Set AN2 to trigger from software.

    /* Early interrupt */
    ADCEIEN1 = 0;                 // No early interrupt
    ADCEIEN2 = 0;

    /* Turn the ADC on */
    ADCCON1bits.ON = 1;

    /* Wait for voltage reference to be stable */
    while(!ADCCON2bits.BGVRRDY);  // Wait until the reference voltage is ready
    while(ADCCON2bits.REFFLT);    // Wait if there is a fault with the reference voltage

    /* Enable clock to analog circuit */
    ADCANCONbits.ANEN0 = 1;       // Enable the clock to analog bias
    ADCANCONbits.ANEN1 = 1;       // Enable the clock to analog bias
    ADCANCONbits.ANEN2 = 1;       // Enable the clock to analog bias

    /* Wait for ADC to be ready */
    while(!ADCANCONbits.WKRDY0);  // Wait until ADC0 is ready
    while(!ADCANCONbits.WKRDY1);  // Wait until ADC1 is ready
    while(!ADCANCONbits.WKRDY2);  // Wait until ADC2 is ready

    /* Enable the ADC module */
    ADCCON3bits.DIGEN0 = 1;       // Enable ADC0
    ADCCON3bits.DIGEN1 = 1;       // Enable ADC1
    ADCCON3bits.DIGEN2 = 1;       // Enable ADC2

    while (1) {
        /* Trigger a conversion */
        ADCCON3bits.GSWTRG = 1;

        /* Wait the conversions to complete */
        while (ADCDSTAT1bits.ARDY0 == 0);
        /* fetch the result */
        result[0] = ADCDATA0;

        while (ADCDSTAT1bits.ARDY1 == 0);
        /* fetch the result */
        result[1] = ADCDATA1;

        while (ADCDSTAT1bits.ARDY2 == 0);
        /* fetch the result */
        result[2] = ADCDATA2;

        /*
        * Process results here
        *
        * Note: Loop time determines the sampling time since all inputs are Class 1.
        * If the loop time is small and the next trigger happens before the completion
        * of set sample time, the conversion will happen only after the sample time
        * has elapsed.
        *
        */
    }
return (1);
}
```

22.4.2 选择 ADC 多路开关模拟输入

每个 ADC 模块都具有两个输入，称为正输入和负输入。专用和共用 ADC 模块的输入选择选项有所不同，以下几节进行了介绍。

22.4.2.1 正输入选择

对于专用 ADC 模块（ADC0-ADC6），为每个正输入提供了 4 个备用选择。该备用输入可以使用 ADC 触发模式寄存器 （ADCTRGMODE）中的 SHxALT<1:0> 位进行选择，如下所示：

- SH0ALT<1:0> （ADCTRGMODE<17:16>）
- SH1ALT<1:0> （ADCTRGMODE<19:18>）
- SH2ALT<1:0> （ADCTRGMODE<21:20>）
- SH3ALT<1:0> （ADCTRGMODE<23:22>）
- SH4ALT<1:0> （ADCTRGMODE<25:24>）
- SH5ALT<1:0> （ADCTRGMODE<27:26>）
- SH6ALT<1:0> （ADCTRGMODE<29:28>）

关于 ADC 输入可能的备用选择，请参见器件数据手册。

对于共用 ADC 模块（ADC7），正输入在所有 2 类和 3 类输入之间共用。对于 2 类输入触发或者在 2 类或类 3 输入扫描期间，模拟输入 ANx 与 ADC7 的输入连接是自动完成的。**第 22.4.6 节 “选择扫描的输入”**单独介绍了扫描输入的选择。

22.4.2.2 负输入选择

负输入选择由 ADCIMCONx 寄存器的 DIFFx 位的设置决定。DIFFx 位允许输入为轨到轨输入，并且可以为单端或差分输入。ADCIMCONx 寄存器中的 SIGNx 和 DIFFx 位用于调节内部 ADC 模拟输入和参考电压，以及将数字结果配置与预期的满量程输出范围一致。

对于共用 ADC 模块（ADC7），各个模拟输入可单独设置 DIFFx 位。因此，用户可以选择某些输入作为单端输入，其他一些作为差分输入，并同时将它们与同一共用 ADC 模块连接。在采样时，信号会根据其相应的 DIFFx 位设置，动态地变为单端或差分信号。

表 22-3： 负输入选择

ADCIMCON1		输入配置	输入电压			输出
DIFFx	SIGNx					
1	1	差分补码	最小输入		VINP - VINN = -VREF	-2048
			最大输入		VINP - VINN = VREF	+2047
1	0	差分单极	最小输入		VINP - VINN = -VREF	0
			最大输入		VINP - VINN = VREF	+4095
0	1	单端补码	最小输入		VINP = VREF	-2048
			最大输入		VINP - VINN = VREF	+2047
0	0	单端单极	最小输入		VINP = VREF	0
			最大输入		VINP - VINN = VREF	+4095

图注： VINP = S&H 正输入； VINN = S&H 负输入； VREF = VREFH - VREFL

> **注：** 为了正确工作和防止器件损坏，输入电压不应超出具体器件数据手册的 **“电气特性”** 章节中列出的限制。

 © 2015 Microchip Technology Inc.

22.4.3 选择 ADC 结果的格式

ADC 结果寄存器中的数据可以使用 4 种受支持数据格式中的任意一种读取。用户可以从无符号整数、有符号整数、无符号小数或有符号小数之间进行选择。整数数据进行右对齐，小数数据进行左对齐。

- 对于所有模拟输入，整数或小数数据格式使用小数数据输出格式位 FRACT （ADCCON1<23>）来全局指定
- 使用 ADCIMCONx 寄存器中的 SIGNx 位，可以为每个独立模拟输入指定有符号或无符号数据格式。

表 22-4 说明了结果是如何格式化的。

表 22-4： ADC 结果格式结果

FRACT	SIGNx	说明	32 位输出数据格式			
0	0	无符号整数	0000 0000	0000 dddd	0000 dddd	0000 dddd
0	1	有符号整数	ssss ssss	ssss sddd	ssss dddd	ssss dddd
1	0	小数	dddd 0000	dddd 0000	dddd 0000	0000 0000
1	1	有符号小数	sddd 0000	dddd 0000	dddd 0000	dddd 0000

例 22-2: ADC 2 类配置和小数格式

```
int main(int argc, char** argv) {
    int result[3];

    /* Configure ADCCON1 */
    ADCCON1bits.FRACT = 1;     // use Fractional output format
    ADCCON1bits.SELRES = 3;    // ADC7 resolution is 12 bits
    ADCCON1bits.STRGSRC = 0;   // No scan trigger.

    /* Configure ADCCON2 */
    ADCCON2bits.SAMC = 5;          // ADC7 sampling time = 5 * TAD7
    ADCCON2bits.ADCDIV = 1;        // ADC7 clock freq is half of control clock = TAD7

    /* Initialize warm up time register */
    ADCANCON = 0;
    ADCANCONbits.WKUPCLKCNT = 5;  // Wakeup exponent = 32 * TADx

    /* Clock setting */
    ADCCON3 = 0;
    ADCCON3bits.ADCSEL = 0;        // Select input clock source
    ADCCON3bits.CONCLKDIV = 1;     // Control clock frequency is half of input clock
    ADCCON3bits.VREFSEL = 0;       // Select AVdd and AVss as reference source

    /* No selection for dedicated ADC modules, no pre sync trigger, not sync sampling */
    ADCTRGMODEbits = 0;

    /* Select ADC input mode */
    ADCIMCON1bits.SIGN7 = 0;  // unsigned data format
    ADCIMCON1bits.DIFF7 = 0;  // Single ended mode
    ADCIMCON1bits.SIGN8 = 0;  // unsigned data format
    ADCIMCON1bits.DIFF8 = 0;  // Single ended mode
    ADCIMCON1bits.SIGN9 = 0;  // unsigned data format
    ADCIMCON1bits.DIFF9 = 0;  // Single ended mode

    /* Configure ADCGIRQENx */
    ADCGIRQEN1 = 0;            // No interrupts are used
    ADCGIRQEN2 = 0;

    /* Configure ADCCSSx */
    ADCCSS1 = 0;               // No scanning is used
    ADCCSS2 = 0;

    /* Configure ADCCMPCONx */
    ADCCMPCON1 = 0;            // No digital comparators are used. Setting the ADCCMPCONx
    ADCCMPCON2 = 0;            // register to '0' ensures that the comparator is disabled.
    ADCCMPCON3 = 0;            // Other registers are “on't care”
    ADCCMPCON4 = 0;
    ADCCMPCON5 = 0;
    ADCCMPCON6 = 0;

    /* Configure ADCFLTRx */
    ADCFLTR1 = 0;              // No oversampling filters are used.
    ADCFLTR2 = 0;
    ADCFLTR3 = 0;
    ADCFLTR4 = 0;
    ADCFLTR5 = 0;
    ADCFLTR6 = 0;
    /* Set up the trigger sources */
    ADCTRGSNSbits.LVL7 = 0;    // Edge trigger
    ADCTRGSNSbits.LVL8 = 0;    // Edge trigger
    ADCTRGSNSbits.LVL9 = 0;    // Edge trigger
    ADC1TRG2bits.TRGSRC7 = 1;  // Set AN7 to trigger from software
    ADC2TRG3bits.TRGSRC8 = 1;  // Set AN8 to trigger from software
    ADC2TRG3bits.TRGSRC9 = 1;  // Set AN9 to trigger from software
```

 © 2015 Microchip Technology Inc.

例 22-2： ADC 2 类配置和小数格式 （续）

```
    /* Early interrupt */
    ADCEIEN1 = 0;                 // No early interrupt
    ADCEIEN2 = 0;

    /* Turn the ADC on */
    ADCCON1bits.ON = 1;

    /* Wait for voltage reference to be stable */
    while(!ADCCON2bits.BGVRRDY);  // Wait until the reference voltage is ready
    while(ADCCON2bits.REFFLT);    // Wait if there is a fault with the reference voltage

    /* Enable clock to analog circuit */
    ADCANCONbits.ANEN7 = 1;       // Enable the clock to analog bias

    /* Wait for ADC to be ready */
    while(!ADCANCONbits.WKRDY7);  // Wait until ADC7 is ready

    /* Enable the ADC module */
    ADCCON3bits.DIGEN7 = 1;       // Enable ADC7

    while (1) {
        /* Trigger a conversion */
        ADCCON3bits.GSWTRG = 1;

        /* Wait the conversions to complete */
        while (ADCDSTAT1bits.ARDY7 == 0);
        /* fetch the result */
        result[0] = ADCDATA7;

        while (ADCDSTAT1bits.ARDY8 == 0);
        /* fetch the result */
        result[1] = ADCDATA8;

        while (ADCDSTAT1bits.ARDY9 == 0);
        /* fetch the result */
        result[2] = ADCDATA9;

        /*
        * Process results here
        *
        * Note 1: Loop time determines the sampling time since all inputs are Class 2.
        * If the loop time happens is small and the next trigger happens before the
        * completion of set sample time, the conversion will happen only after the
        * sample time has elapsed.
        *
        * Note 2: Results are in fractional format
        *
        */
    }
    return (1);
}
```

22.4.4 选择转换触发源

ADC 模块的 1 类和 2 类输入可以单独触发转换，也可以作为扫描序列的一部分而触发转换。3 类输入只能作为扫描序列的一部分触发。单独触发或扫描触发的信号源可以为：板载定时器或输出比较外设事件、与 INT0 连接的外部数字电路、与模拟比较器连接的外部模拟电路，或者用软件将 SFR 中的触发位置 1。

> **注：** 当多个 2 类模拟输入同时发生转换触发时，将基于所使用的模拟输入按照自然顺序优先级方案来为它们分配优先级。AN7 具有最高优先级，AN8 具有下一最高优先级，如此类推。

22.4.4.1 1 类和 2 类输入的触发选择

对于每个 1 类和 2 类输入，用户应用程序可以独立指定转换触发源。模拟输入 x 的独立触发源通过位于寄存器 ADCTRG1 至 ADCTRG8 中的 TRGSRC<4:0> 位指定。关于可用转换触发选项的更多信息，请参见具体器件数据手册中的 **“ADC”** 章节。例如，这些触发源可能包括：

- 通用（GP）定时器：当 32 位定时器 Timer3/2 或 Timer5/4，或者 16 位定时器 Timer1、Timer3 或 Timer5 发生周期匹配时，定时器会产生特殊的 ADC 触发事件信号。其他定时器不存在该功能。更多信息，请参见**第 14 章 “定时器”**（DS60001105）和具体器件数据手册的 **“定时器”** 章节。
- 输出比较：输出比较外设 OC1、 OC3 和 OC5 可以用于产生 ADC 触发信号，然后输出从低电平转换为高电平状态。更多信息，请参见**第 16 章 “输出比较”**（DS60001111）和具体器件数据手册中的 **“输出比较”** 章节。
- 比较器：模拟比较器可以用于在输出从低电平状态转换为高电平状态时产生 ADC 触发信号。更多信息，请参见**第19章“比较器”**（DS60001110）和具体器件数据手册的**“比较器”**章节。
- 外部 INT0 引脚触发：在该模式下，ADC 模块在 INT0 引脚上发生有效电平变换时启动转换。INT0 引脚可以设定为使用上升沿输入或下降沿输入来触发转换过程。
- 全局软件触发：ADC 模块可以配置为对所有选择了该触发选项的输入手动触发转换。用户可以通过将全局软件触发位 GSWTRG （ADCCON3<6>）置 1 来手动触发转换。

22.4.4.2 1 类模拟输入的预同步触发和异步采样

ADCTRGx 寄存器用于 ADC 触发。

- 转换数据可以存储在专用输出寄存器 （ADCDATAx）中
- 转换数据可以存储在 FIFO 中
- 转换数据可以使用 DMA 引擎存储在系统 RAM 中

ADCTRGMODE 寄存器包含 ADC 模块的触发模式，触发模式由 STRGEN 位（预同步触发使能）和 SSAMPEN 位 （还对于空闲或禁止之后的第一个采样使用同步采样）决定。

> **注 1：** 触发和同步基于“控制时钟（TQ）”工作。但采样时间（t_{SAMCX}）基于 T_{ADX} 时钟（它基于 TQ 信号通过分频器之后得到）。
>
> **2：** 当使用同步采样（SSAMPEN = 1）时，预同步触发值会被忽略（STRGEN =“无关”）。

 © 2015 Microchip Technology Inc.

22.4.4.2.1 同步采样 （SSAMPEN = 1） →

在发生触发事件之后，将正好在预同步触发之后的 (2 * TQ + *t*SAMCX) 时间处或在前一次转换之后的 *t*SAMC 时间处，结束采样并开始转换；其中， *t*SAMCX 是由 SAMC<9:0> （ADCxTIME<9:0>）位设置的采样时间。

22.4.4.2.2 异步采样 （SSAMPEN = 0） →

在发生触发事件后，如果达到并已完成（已经过）预同步触发之后的 (2 * TQ + *t*SAMCX) 时间或前一次转换之后的 *t*SAMCX 时间，则立即结束采样并开始转换 （不论是否进行预同步）。如果未达到该时间，采样将持续 *t*SAMCX 的时间，之后才会开始转换。实际上，这会变为同步采样，但只有在发生上一次异步触发之前未达到 *t*SAMCX 时间要求的情况下才会如此。如果在发生上一次异步触发之前达到了 *t*SAMCX 时间要求，则实际的上一次触发会发出异步采样结束。只有转换开始是始终同步的。在异步采样结束和接下来的按时钟正边沿对齐的转换开始之间将存在可变的延时，该延时通过采样电容来补偿，即采样电容会使模拟电压值一直保持不变，直到可以通过同步 ADC 对它进行转换为止。

表 22-5： 1 类 ADC 触发

STRGEN	SSAMPEN	说明
0	0	无预同步触发和异步采样。
1	0	预同步触发和异步采样。
x	1	同步采样。

图 22-9 以图形方式说明了对应于 STRGEN 和 SSAMPEN 位的各种情形。所有触发本质上都是异步；类似地， ADC 模块的触发信号也是异步的，可以在相对于单个 TQ 时钟的任意时刻到达。这称为 “（1 周期抖动）”。在接收到触发信号时，会在实际触发之后的 1 个 TQ 时钟处发生 “预同步触发” 。请注意，预同步触发是一个内部信号。

假设这样一种情形：在 STRGENx 和 SSAMPENx 均为 0 （无预同步触发和异步采样）时，异步触发导致 ADC 从采样模式切换为转换模式。也即，在这种情况下，预同步触发会被忽略 （即，不使用）。只有在达到采样时间 （*t*SAMCX）要求时，才会出现这种情形。

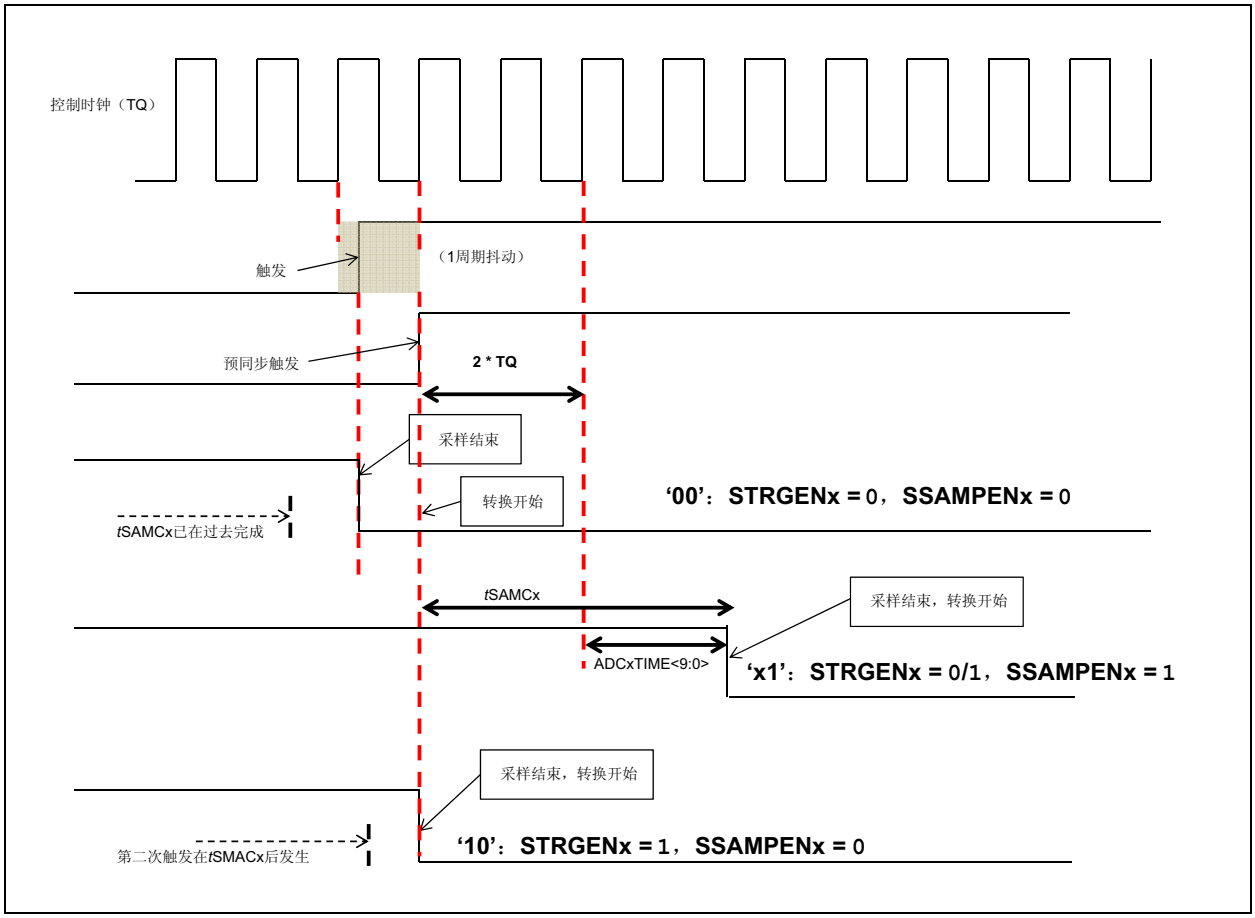
假设下一种情形：在 STRGENx = 1 且 SSAMPENx = 0 （预同步触发和异步采样）时，预同步触发是导致 ADC 从采样模式切换为转换模式的信号。只有在达到采样时间 （*t*SAMCX）要求时，才会出现这种情形。

对于前面提到的两种情形（STRGENx 和 SSAMPENx = 00 或 STRGENx 和 SSAMPENx = 10），如果达到采样时间 （*t*SAMCX）要求，则会发生所说明的行为。对于发生重复触发且达到采样时间（*t*SAMCX）要求的情形，图 22-10 以图形方式显示了所说明的行为。该图中显示了在经过 *t*SAMCX 时间后发生第二次触发。因此，第二次触发会导致 ADC 从采样模式切换为转换模式。请注意，触发信号（第一次或第二次触发）为异步触发或预同步触发，具体取决于 STRGENx 和 SSAMPENx 设置为 00 还是 10。

在未达到 *t*SAMCX 时间要求的情况下，触发信号 （无论是异步触发还是预同步触发）不会导致从采样模式切换为转换模式。 ADC 会等待达到 *t*SAMCX 时间，之后再从采样模式切换为转换模式。图 22-11 对此进行了说明。该图中，在经过 *t*SAMCX 时间之前发生第二次触发。因此，这次触发不会导致开始转换。只有在完成 *t*SAMCX 时间之后，才会从采样模式切换为转换模式。换句话说，即使在将 ADC 设置为异步采样模式（SSAMPEN = 0）时，如果未达到每个触发信号之间的采样时间 （*t*SAMCX）要求， ADC 的行为也会是同步的。

回到图 22-9，假设存在 STRGENx = x 且 SSAMPENx = 1 （同步采样）的情形，则 ADC 会在预同步触发之后的 (2 * TQ + *t*SAMCX) 时间处从采样模式切换为转换模式。

图 22-9： 预同步触发 （STRGEN 位）和同步采样 （SSAMPEN 位）



 © 2015 Microchip Technology Inc.

图 22-10： 异步采样（SSAMPEN = 0）（在触发速率可使采样时间达到 t_{SAMCx} 要求时， ADC 允许异步触发信号启动转换）

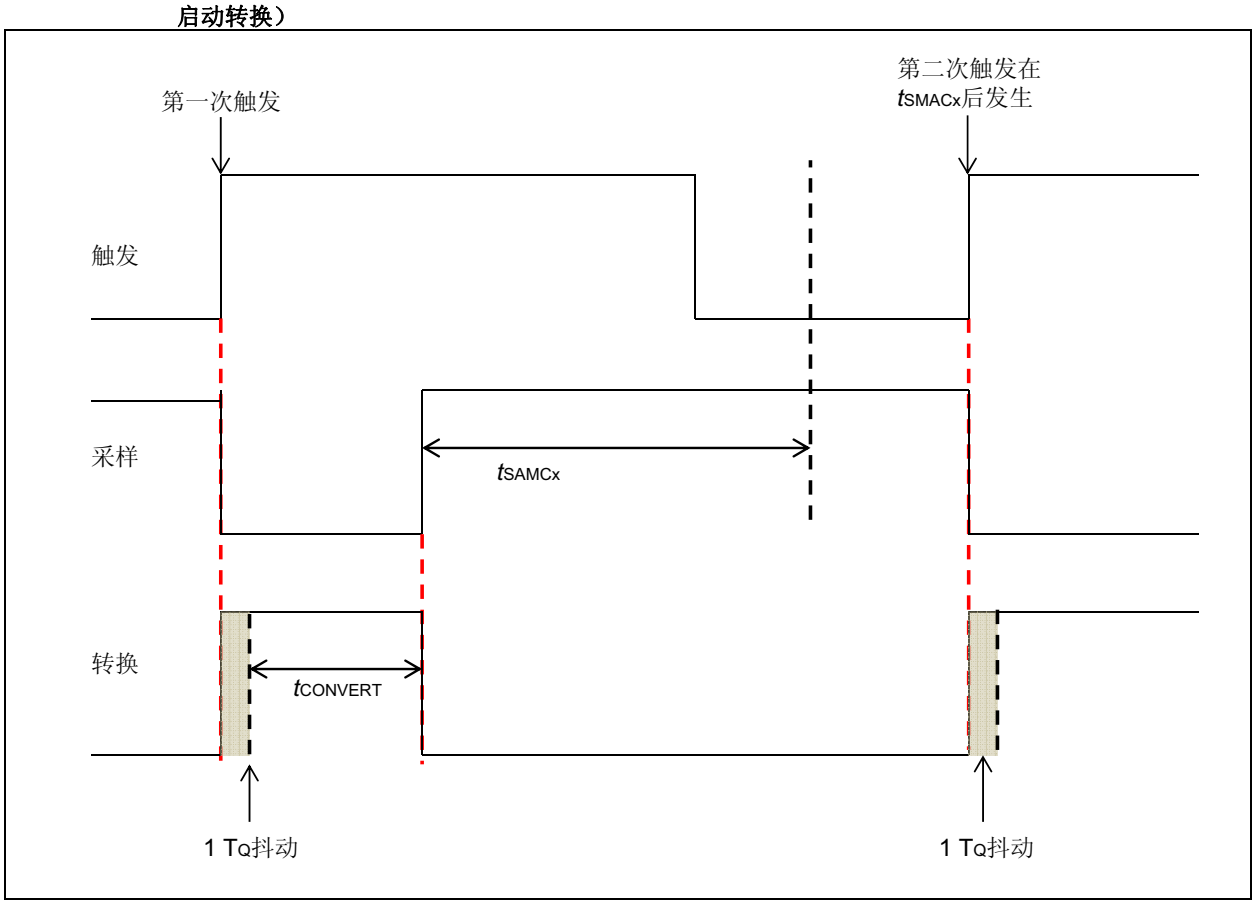
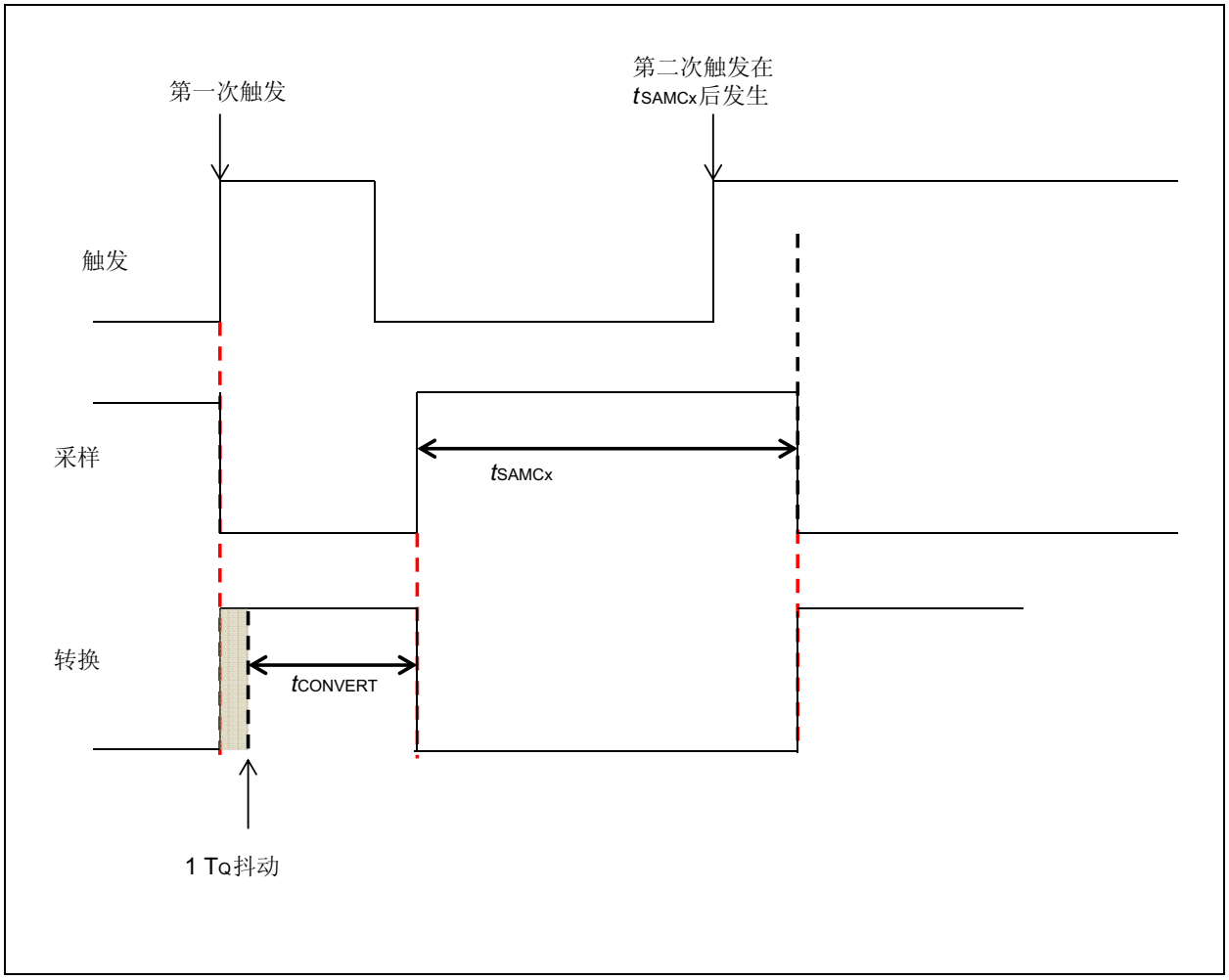


图 22-11： 异步采样（SSAMPEN = 0）（触发速率太快（在 *t*SAMCx 完成之前），ADC 会强制确保达到最小采样时间（*t*SAMCx），并默认为同步采样）

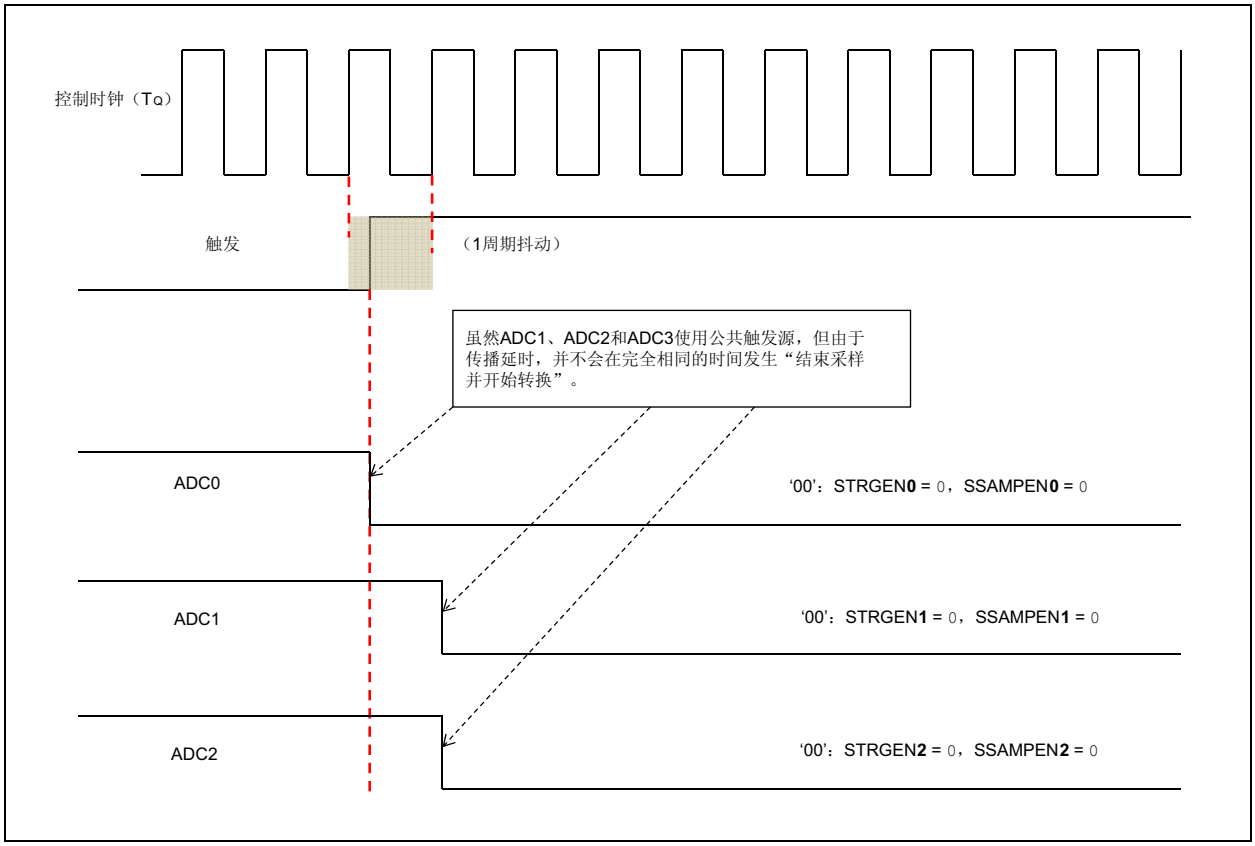


1 类输入通常用于转换具有快速和瞬变特性的模拟信号。此外，需要使用多个 1 类输入来转换相互之间存在相位关系的不同模拟信号。此类信号的一个示例就是电机的三相电流信号。

假设这样一种情形：使用 ADC0、 ADC1 和 ADC2 来对三相电流进行采样，所有 ADC 模块都使用 STRGENx 和 SSAMPENx = 00 （无预同步触发和异步采样）的设置，每个 ADC 的触发时间略有不同 （由于触发信号的传播延时）。图 22-12 对此进行了说明。这种触发时间差会导致采样的模拟信号中存在相位误差。为了避免此类误差，应使用预同步触发和异步采样（即，STRGENx 和 SSAMPENx = 10）。使用预同步触发可以确保所有 ADC 模块可以在完全相同的时间接收到触发信号。

 © 2015 Microchip Technology Inc.

图 22-12： 预同步触发波形 （使用多个 1 类输入来转换三相电机的相电流时）



由于触发和同步使用控制时钟（TQ）工作，而 ADC 模块使用它们自己的时钟（TADX）工作，所以应在两个时钟域之间维持适当的同步。为了维持适当的同步，提供了两个位 FSSCLKEN（ADCCON1<10>）和 FSPBCLKEN （ADCCON1<9>）。这些位的使用方式如下：

- 当以下条件之一为真时，将 FSSCLKEN 位置 1：
  - ADCSEL<1:0> = 11 （SYSCLK）
  - ADCSEL<1:0> = 10 （REFCLK3），并且 REFCLK3 还是基于系统时钟 （SYSCLK）产生的分频时钟
  - ADCSEL<1:0> = 01 （FRC），并且系统时钟还设置为由 FRC 驱动
  - ADCSEL<1:0> = 00 （PBCLK），并且 PBCLK 是基于系统时钟产生的分频时钟
- 当以下条件之一为真时，将 FSPBCLKEN 位置 1：
  - ADCSEL<1:0> = 11 （SYSCLK），并且外设时钟是基于系统时钟产生的分频时钟，而 ADC 的速率不高于外设时钟。这意味着外设时钟和 ADC 时钟都是基于相同系统时钟进行分频得到的，但 ADC 时钟的分频比高于或等于外设时钟的分频比，这会使外设时钟与 ADC 时钟同步，但速率不低于 ADC 时钟。
  - ADCSEL<1:0> = 10 （REFCLK3），并且 REFCLK3 与外设时钟同步，但速率低于外设时钟 （如果外设时钟也是基于 REFCLK3 产生的，但速率不低于 ADC 时钟，则此条件为真）
  - ADCSEL<1:0> = 01 （FRC），并且外设时钟也是基于 FRC 产生的，其速率不低于 ADC 时钟
  - ADCSEL<1:0> = 00 （外设时钟）

22.4.4.3 转换触发源和控制

以下是每个触发信号的可能来源。

- 通过 ADCTRGx 寄存器中的 TRGSRCx<4:0> 位选择的外部触发源。只有 1 类和 2 类模拟输入支持该功能。通常情况下，用户通过指定特定触发源来启动特定输入的转换。

  如果需要，所有模拟输入可以选择同一触发源。在这种情况下，产生的结果类似于 “扫描转换”，其完成顺序将遵从与同一触发源关联的输入的优先级。第一个触发选择为 00000（无触发），这相当于暂时禁止该特定触发源，因而暂时禁止对该模拟输入进行转换。接下来的两个触发源选择 （GSWTRG 和 GLSWTRG）是软件产生的触发源。第二个软件产生的触发选择是全局软件触发 （GSWTRG）。该触发源与 ADCCON3 寄存器中的 GSWTRG 位相链接，用户应用程序可以通过该位来启动单次转换。由于 GSWTRG 是自清零位，所以在被用户应用程序置 1 之后，它会在下一个 ADC 时钟周期清零自身。第三个软件产生的触发选择是全局电平软件触发 （GLSWTRG），它与 ADCCON3 寄存器中的 GLSWTRG 位相链接。用户应用程序可以使用该触发选择来启动一连串的连续采样，因为 GLSWTRG 位不是自清零位。第四个触发选择是一个特殊选择，即扫描触发选择，它允许包含 1 类和 2 类模拟输入作为所有输入的全局扫描的成员。其余的触发选择 5 至 31 取决于器件。更多信息，请参见具体器件的数据手册。
- 通过 ADCCON1 寄存器中的 STRGSRC<4:0> 位和 ADCCSSx 寄存器中的选定位选择的扫描触发源。该模式通常用于启动一组模拟输入的转换。该功能适用于 1 类、 2 类和 3 类模拟输入，但通常用于 3 类输入，因为它们没有各自的关联的 TRGSRC 位。触发选择之一是 ADCCON3 寄存器中的 GSWTRG 位，用户软件可以使用该位来启动转换。
- 用户通过 ADCCON3 寄存器中的 ADINSEL<5:0> 位和 RQCNVRT 位启动的触发。用户应用程序可以通过该模式为指定模拟输入产生单独的转换触发请求。通过使用该模式，用户应用程序可以触发输入的转换，而无需更改 ADC 的触发源配置。这在一些错误处理情况下会很有用：另一个软件模块希望获取 ADC 信息，而又不破坏 ADC 的正常操作。这也是产生首次触发来启动数字滤波器序列的首选方法。
- 用户通过 ADCCON3 寄存器中的 ADINSEL<5:0> 位和 SAMP 位控制的 2 类和 3 类输入采样。将 SAMP 位置 1 会使 2 类和 3 类输入处于采样模式，忽略 SAMC<9:0> 位的选择。该模式对于软件 ADC 转换也很有用，采样时间可由软件选择。
- 外部模块 （如 PTG）可以通过将 ADCCON2 寄存器中的 ECRIEN 位来指定转换的模拟输入。这种方法独立于通常的 TRGSRC 和 STRGSRC 方法工作。外部模块仍然可以使用独立触发信号，并通过通常的 TRGSRC 和 STRGSRC 方法启动转换。

 © 2015 Microchip Technology Inc.

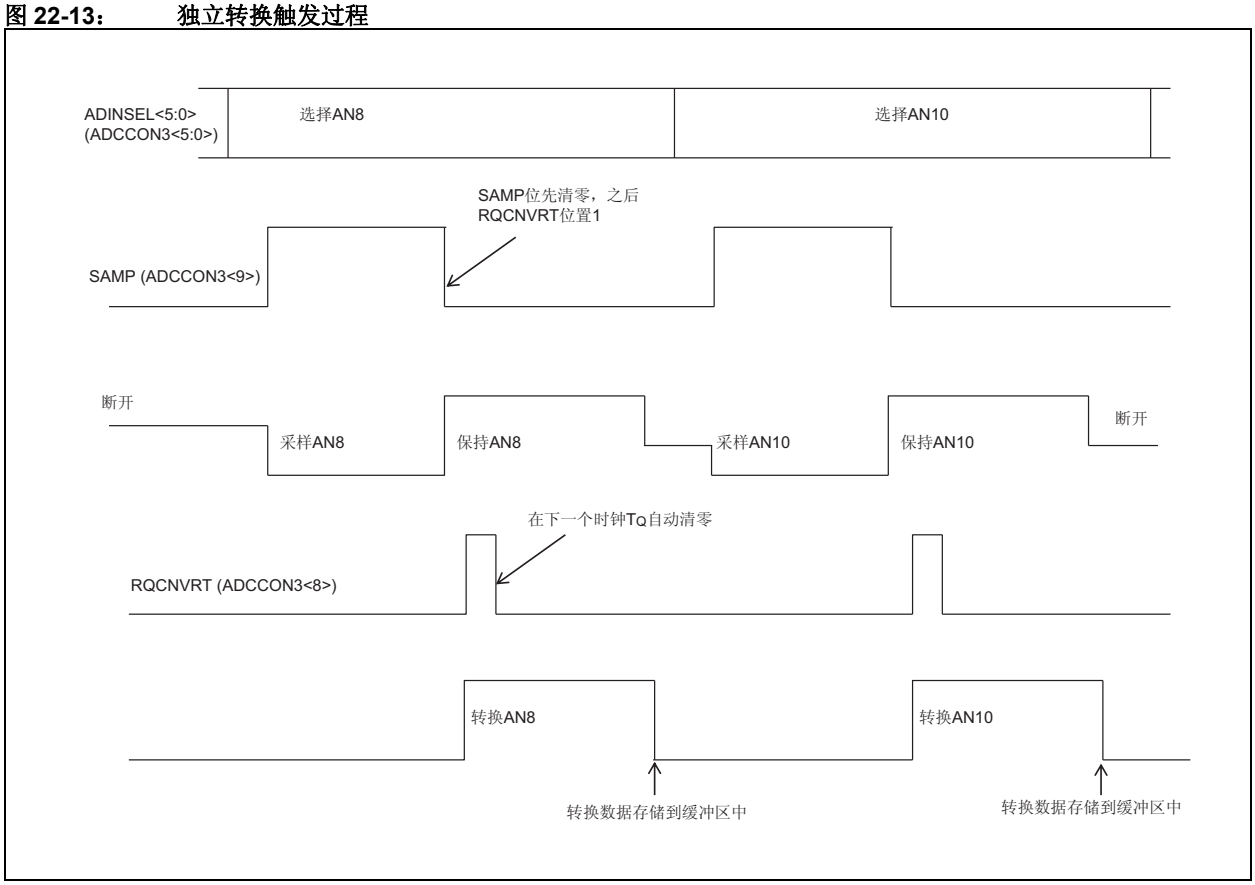
22.4.4.4 用户请求的独立转换触发 （软件 ADC 转换）（仅对于 2 类和 3 类输入）

在程序执行期间，用户可以在任意时刻 （用软件）显式请求对任何选定模拟输入进行单次转换，而无需更改 ADC 的触发源配置。进行转换需要执行的步骤如下：

- 通过 ADC 输入选择位 ADINSEL<5:0> （ADCCON3<5:0>）指定要转换的模拟输入 ID。
- 通过将 SAMP 位 （ADCCON3<9>）置 1 来启动模拟输入的采样
- 经过必要的采样时间 （延时）之后， SAMP 位会被清零。
- 通过将 RQCNVRT 位 （ADCCON3<8>）置 1 来启动采样信号的转换
- 在转换完成时， ADCDSTATx 寄存器的 ARDYx 位会置 1。此时可以从 ADCDATAx 寄存器中读取数据。

图 22-13 以图形形式说明了转换过程：

图 22-13： 独立转换触发过程



22.4.5 选择参考电压源

用户应用程序可以选择 ADC 模块的参考电压，它可以是内部或外部电压。参考电压输入选择位 VREFSEL<2:0> （ADCCON3<15:13>）用于选择模数转换的参考电压。参考电压高电压（VREFH）和参考电压低电压（VREFL）可以用作内部 AVDD 和 AVSS 电压轨或带隙参考电压发生器，或者作为外部 VREF+ 和 VREF- 输入引脚。当参考电压和带隙参考电压就绪时， BGVRRDY（ADCCON2<31>）位会置 1。如果参考电压发生故障（如欠压），REFFLT 位（ADCCON2<30>）会置 1。如果 BGVRIEN 位 （ADCCON2<15>）和 REFFLTIEN 位 （ADCCON2<14>）置 1，BGVRRDY 和 REFFLT 位还可以分别产生中断。

加到外部参考电压引脚上的电压必须符合特定的规范。关于电气规范，请参见具体器件数据手册中的 **“电气特性”** 章节。

当选定参考电压之间的差（VREFH - VREFL）小于 0.65 * (AVDD - AVSS) 时，应将模拟输入电荷泵使能位 AICPMPEN （ADCCON1<12>）置 1。将该位置 1 并不会增大参考电压；但是，将该位置 1 会减小到采样电容的串联源电阻。这可以最大限度地提高使用较小参考电压轨进行模数转换时的 SNR。

> 注 1： 外部 VREF+ 和 VREF- 引脚可以与其他模拟外设共用。更多信息，请参见具体器件数据手册中的 **“引脚表”** 章节。此外， VREF+ 和 VREF- 引脚的 ANSELx 位必须设置为模拟模式。
>
> 2： 带隙参考电压源并非在所有器件上都可用。要确定该特性的可用性，请参见具体器件的数据手册。

22.4.6 选择扫描的输入

所有可用模拟输入都可以配置为用于扫描。1 类输入使用其专用 ADC 模块进行采样。2 类和 3 类输入使用共用 ADC 模块进行采样。对于选择用于扫描的所有输入，使用 STRGSRC<4:0> 位（ADCCON1<20:16>）来选择单个转换触发源。在每次发生转换触发时， ADC 模块会按自然优先级来转换用户指定的扫描列表（ADCCSS1 或 ADCCSS2）的中指定的所有输入。对于 1 类输入，采样在触发时结束，然后它们全部并行开始转换。对于 2 类和 3 类输入，触发信号会按自然优先级顺序来启动顺序采样 / 转换过程。

如果模拟输入满足以下条件，则它属于扫描的一部分：

- 3 类输入。对于 3 类输入，扫描是进行转换的唯一机制。
- 1 类或 2 类输入，并且它们通过位于 ADCTRG1 至 ADCTRG8 寄存器中的 TRGSRCx<4:0> 位选择了 STRIG 选项，从而选择扫描触发作为其触发源。

可用于扫描的触发选项与可用于独立触发 1 类和 2 类输入的触发选项相同。对于属于扫描一部分的所有 1 类或 2 类输入，必须在 TRGSRCx<4:0> 位中选择 STRIG 选项作为其触发源。

> 注 1： 由于每个专用ADC模块（ADCxTIME寄存器中的SELRES、ADCDIV和SAMC<9:0>位）和共用 ADC 模块（ADCCON1 寄存器中的 SELRES，ADCCON2 寄存器中的 ADCDIV 和 SAMC<9:0> 位）的时钟分频比、分辨率和采样时间都通过单独的寄存器进行处理，所以如果要设置的扫描序列同时涉及专用和共用输入，则必须单独初始化这些寄存器中的每个寄存器，即使是在扫描模式下。
>
> 2： 只有所有 1 类输入都已完成扫描，并且最后一个共用输入转换已完成，才会产生扫描结束 （EOS）事件。扫描序列会一直保持有效，直到这两个条件都满足为止。因此， EOS 中断可以用于任何扫描序列，用于输入类型的任意组合。

 © 2015 Microchip Technology Inc.

例 22-3: ADC 扫描多个输入

```
int main(int argc, char** argv) {
    int result[3];

    /* Configure ADCCON1 */
    ADCCON1 = 0;              // No ADCCON1 features are enabled including: Stop-in-Idle, turbo,
                              // CVD mode, Fractional mode and scan trigger source.
    ADCCON1bits.SELRES = 3;   // ADC7 resolution is 12 bits
    ADCCON1bits.STRGSRC = 1;  // Select scan trigger.

    /* Configure ADCCON2 */
    ADCCON2bits.SAMC = 5;     // ADC7 sampling time = 5 * TAD7
    ADCCON2bits.ADCDIV = 1;   // ADC7 clock freq is half of control clock = TAD7

    /* Initialize warm up time register */
    ADCANCON = 0;
    ADCANCONbits.WKUPCLKCNT = 5; // Wakeup exponent = 32 * TADx

    /* Clock setting */
    ADCCON3bits.ADCSEL = 0;       // Select input clock source
    ADCCON3bits.CONCLKDIV = 1;    // Control clock frequency is half of input clock
    ADCCON3bits.VREFSEL = 0;      // Select AVdd and AVss as reference source

    ADC0TIMEbits.ADCDIV = 1;      // ADC0 clock frequency is half of control clock = TAD0
    ADC0TIMEbits.SAMC = 5;        // ADC0 sampling time = 5 * TAD0
    ADC0TIMEbits.SELRES = 3;      // ADC0 resolution is 12 bits

    /* Select analog input for ADC modules, no presync trigger, not sync sampling */
    ADCTRGMODEbits.SH0ALT = 0;    // ADC0 = AN0

    /* Select ADC input mode */
    ADCIMCON1bits.SIGN0 = 0;      // unsigned data format
    ADCIMCON1bits.DIFF0 = 0;      // Single ended mode
    ADCIMCON1bits.SIGN8 = 0;      // unsigned data format
    ADCIMCON1bits.DIFF8 = 0;      // Single ended mode
    ADCIMCON1bits.SIGN40 = 0;     // unsigned data format
    ADCIMCON1bits.DIFF40 = 0;     // Single ended mode

    /* Configure ADCGIRQENx */
    ADCGIRQEN1 = 0;               // No interrupts are used.
    ADCGIRQEN2 = 0;

    /* Configure ADCCSSx */
    ADCCSS1 = 0;                  // Clear all bits
    ADCCSS2 = 0;
    ADCCSS1bits.CSS0 = 1;         // AN0 (Class 1) set for scan
    ADCCSS1bits.CSS8 = 1;         // AN8 (Class 2) set for scan
    ADCCSS2bits.CSS40 = 1;        // AN40 (Class 3) set for scan

    /* Configure ADCCMPCONx */
    ADCCMPCON1 = 0;               // No digital comparators are used. Setting the ADCCMPCONx
    ADCCMPCON2 = 0;               // register to '0' ensures that the comparator is disabled.
    ADCCMPCON3 = 0;               // Other registers are 該on't care’.
    ADCCMPCON4 = 0;
    ADCCMPCON5 = 0;
    ADCCMPCON6 = 0;

    /* Configure ADCFLTRx */
    ADCFLTR1 = 0;                 // No oversampling filters are used.
    ADCFLTR2 = 0;
    ADCFLTR3 = 0;
    ADCFLTR4 = 0;
    ADCFLTR5 = 0;
    ADCFLTR6 = 0;
```

例 22-3： ADC 扫描多个输入 （续）

```
    /* Set up the trigger sources */
    ADCTRG1bits.TRGSRC0 = 3;      // Set AN0 (Class 1) to trigger from scan source
    ADCTRG3bits.TRGSRC8 = 3;      // Set AN8 (Class 2) to trigger from scan source
                                  // AN40 (Class 3) always uses scan trigger source

    /* Early interrupt */
    ADCEIEN1 = 0;                 // No early interrupt
    ADCEIEN2 = 0;

    /* Turn the ADC on */
    ADCCON1bits.ON = 1;

    /* Wait for voltage reference to be stable */
    while(!ADCCON2bits.BGVRRDY);  // Wait until the reference voltage is ready
    while(ADCCON2bits.REFFLT);    // Wait if there is a fault with the reference voltage

    /* Enable clock to analog circuit */
    ADCANCONbits.ANEN0 = 1;       // Enable the clock to analog bias ADC0
    ADCANCONbits.ANEN7 = 1;       // Enable, ADC7

    /* Wait for ADC to be ready */
    while(!ADCANCONbits.WKRDY0);      // Wait until ADC0 is ready
    while(!ADCANCONbits.WKRDY7);      // Wait until ADC7 is ready

    /* Enable the ADC module */
    ADCCON3bits.DIGEN0 = 1;           // Enable ADC0
    ADCCON3bits.DIGEN7 = 1;           // Enable ADC7

    while (1) {
        /* Trigger a conversion */
        ADCCON3bits.GSWTRG = 1;

        /* Wait the conversions to complete */
        while (ADCDSTAT1bits.ARDY0 == 0);
        /* fetch the result */
        result[0] = ADCDATA0;

        while (ADCDSTAT1bits.ARDY8 == 0);
        /* fetch the result */
        result[1] = ADCDATA8;

        while (ADCDSTAT2bits.ARDY40 == 0);
        /* fetch the result */
        result[2] = ADCDATA40;

        /*
        * Process results here
        *
        *
        */
    }
    return (1);
}
```

 © 2015 Microchip Technology Inc.

22.4.7 选择模数转换时钟源和预分频比

ADC 模块可以使用内部快速 RC （FRC）振荡器输出、系统时钟 （SYSCLK） 、参考时钟（REFCLK3）或外设总线时钟 （PBCLK）作为转换时钟源 （TQ）。要确定哪些参考时钟输出源可用，请参见具体器件数据手册中的 **“ADC”** 章节。

当 ADCSEL<1:0> 位 （ADCCON2<31:30>） = 01 时，将使用内部 FRC 振荡器作为 ADC 时钟源。当使用内部 FRC 振荡器时， ADC 模块可以在休眠和空闲模式下继续工作。

> **注：** 对于需要对 ADC 采集进行精确计时的应用程序，建议使用 SYSCLK 作为 ADC 的时钟源。

要进行正确的模数转换，一定不能超出转换时钟限制。 ADC 模块支持从 1 MHz 至 28 MHz 的时钟频率。

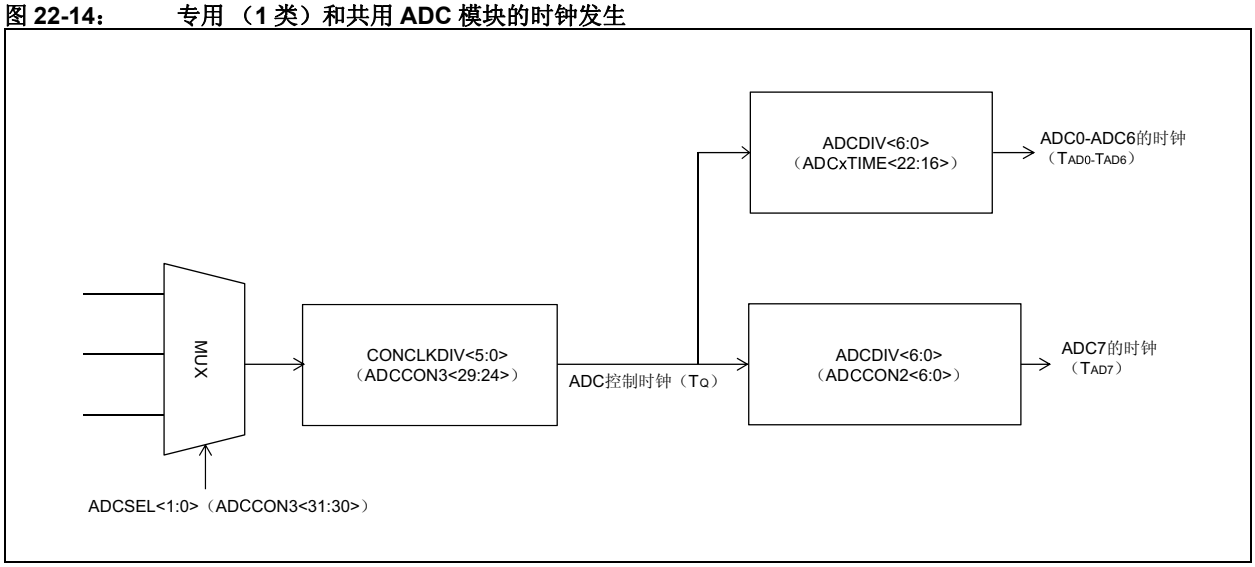
可以通过 ADC 模块完成的模数转换的最大速率 （有效转换吞吐率）为 2 Msps。但是，可以转换单个输入的最大速率取决于采样时间要求。此外，采样时间取决于模拟信号源的输出阻抗。**第 22.10 节 “ADC 采样要求”** 章节提供了关于采样时间的更多信息。

ADC 时钟结构设计为向专用 ADC 模块和共用 ADC 模块提供单独的独立时钟。ADC 的输入时钟源使用 ADCSEL<1:0> 位 （ADCCON3<31:30>）进行选择。输入时钟通过控制时钟分频比 CONCLKDIV<5:0> 位 （ADCCON3<29:24>）进一步进行分频。输出时钟称为 “ADC 控制时钟”，其时间周期为 TQ。

ADC 控制时钟按 ADCDIV<6:0> 位 （ADCxTIME<22:16>）进行分频。它用作各个专用 ADC 模块 （ADC0 至 ADC6）的时钟源，其时间周期为 TADx。

ADC 控制时钟在用于共用 ADC （ADC7）之前通过 ADCDIV<6:0> 位 （ADCCON2<6:0>）进行分频。该时钟的时间周期称为 TAD7。

图 22-14： 专用 （1 类）和共用 ADC 模块的时钟发生



22.4.8 采样和转换时间

公式 22-1： ADC0 至 ADC6 的采样时间

$$t_{SAMCx} = (ADCxTIME<9:0>) \cdot T_{ADx}$$

$$t_{conversion} = (2 + ADCxTIME<25:24>) \cdot T_{ADx}$$

公式 22-2： ADC7 的采样时间

$$t_{SAMC7} = (ADCCON2<25:16>) \cdot T_{AD7}$$

$$t_{conversion} = (2 + ADCCON1<22:21>) \cdot T_{AD7}$$

22.4.9 开启 ADC

开启 ADC 模块涉及到以下步骤：

当 ADC 模块使能位 ON （ADCCON1<15>）设置为 1 时，模块处于工作模式，处于完全供电状态，可使用所有的功能。当 ON 位为 0 时， ADC 模块会被禁止。在被禁止时， ADC 的数字和模拟部分会被关闭，以最大限度地节省电流消耗。除了将 ON 位置 1 之外，还应开启 ADC 的模拟和数字电路。更多详细信息，请参见**第 22.7.3 节 “ADC 低功耗模式”**。

注：	在 ADC 模块使能时，建议不要写入控制 ADC 时钟、输入分配、扫描、参考电压选择、 S&H 电路工作模式和中断配置的 ADC 控制位。

22.4.10 ADC 状态位

ADC 模块在 ADCANCON 寄存器中包含了 WKRDYx/WKRDY7 状态位，它指示 ADC 模拟和偏置电路的当前状态。在该位置 1 之前，用户应用程序不应执行任何 ADC 操作。

22.4.11 动态更新 SFR

一些应用程序需要在不禁止 ADC 的情况下更新专用和共用 ADC 模块的 SFR。虽然这对于 ADC 可以实现的，但用户代码必须遵循一些特定的准则，以防止数据破坏。如果 SFR 发生更改的实际时刻接近于 ADC 模块采样时间结束时，对 SFR 进行更新有可能会完全破坏输出数据。

任何 ADC 模块都可能处于 3 种状态：

- **空闲**：在空闲状态期间， ADC 模块会等待下一个触发事件。这不是更改 SFR 设置的安全时间，因为用户代码无法预测何时会发生下一次异步触发 （它指示采样结束）。如果在修改 SFR 时发生采样结束事件，输出数据会被破坏。
- **采样**：在 ADC 模块由于明显的原因而在进行采样时，无论如何都不应更改任何 SFR
- **转换**：这是更新 SFR 最安全的时间段，因为转换时间是固定的，处于 SAR 自身的控制下

要在 ADC 在进行转换时更新 SFR，可以使用提前中断允许寄存器和提前中断状态寄存器 （见寄存器 22-37 至寄存器 22-40）。

例如，如果 ADCEIS<2:0> 位（ADCxTIME<28:26>）设置为 `0b111`，则当 EIRDYx 变为高电平时， ADC 模块 x 的逐次逼近将正在执行转换的步骤 6 （= 14 - 8）。如果所有 SFR 写操作都可以在 ADC 模块 x 的 SAR 的步骤 13 之前就绪，则用户应用程序可以在此时修改 ADC 模块 x 的 SFR （使用 ADCSEL<1:0>、CONCLKDIV<5:0> 和 ADCDIV<6:0> 位更改时钟设置除外）。SFR 更新写操作的这种安全时间取决于 ADCDIV<6:0> 位的设置。这是因为，软件对 SFR

 © 2015 Microchip Technology Inc.

的写操作将以系统时钟频率进行。当 ADCDIV<6:0> 的值较高时，也即 ADC 速度较低时，可以安全地进行对 SFR 的写操作；与之相对，当 ADCDIV<6:0> 的值较低时，ADC 速度也会较高。ADC 速度较高时，用户应用程序可以查询 EIRDYx 位，而不是使用提前中断。这是因为处理提前 ISR 的中断延时需要一些周期，在查询模式下可以避免这些周期。

以下是可以在安全期中更新的 SFR 位的列表：

- STRGSRC<4:0> （ADCCON1<20:16>） —— 请参见寄存器 22-1
- SAMC<7:0> （ADCCON2<23:16>） —— 请参见寄存器 22-2
- STRGENx （ADCTRGMODE<14:8>） —— 请参见寄存器 22-4
- SSAMPENx （ADCTRGMODE<6:0>） —— 请参见寄存器 22-4
- SHxALT<1:0> （ADCTRGMODE<29:8>） —— 请参见寄存器 22-4
- DIFFx （ADCIMCONx） —— 请参见寄存器 22-5 至寄存器 22-8
- SIGNx （ADCIMCONx） —— 请参见寄存器 22-5 至寄存器 22-8
- CSS63:CSS0 （ADCCSS1<31:0>， ADCCSS2<63:32>） —— 请参见寄存器 22-11 和寄存器 22-12
- TRGSRCx<4:0> （ADCTRGx） —— 请参见寄存器 22-18
- LVL31:LVL0 （ADCTRGSNS<31:0>） —— 请参见寄存器 22-35
- SAMC<7:0> （ADCxTIME<7:0>） —— 请参见寄存器 22-36

除了使用提前中断之外，还可以使用触发暂停位 TRGSUSP （ADCCON3<12>）来向 SFR 写入新值。使用 TRGSUSP 位的步骤如下：

1. 将 TRGSUSP 位 （ADCCON3<12>）置 1。
2. 查询UPDRDY位（ADCCON3<10>）是否置1。可以通过使能UPDIEN位（ADCCON3<11>）来产生中断（可选）。当 UPDRDY 位置 1 时，它指示所有触发都被暂停，所有 ADC 模块都处于其空闲状态。
3. 将新值写入 SFR（使用 ADCSEL<1:0>、CONCLKDIV<5:0> 和 ADCDIV<6:0> 位更改时钟设置除外）。
4. 清零 TRGSUSP 位，使 ADC 可以继续工作。

如果要使用 ADCSEL<1:0>、CONCLKDIV<5:0> 和 ADCDIV<6:0> 位更新与 ADC 时钟设置相关的 SFR，只能通过禁止 ADC （ON 位 （ADCCON1<15>） = 0）来执行该操作。

22.5 其他 ADC 功能

本节介绍 ADC 模块的一些其他功能，它们包括：

- 数字比较器
- 过采样滤波器
- Turbo 模式
- CVD 模式

22.5.1 数字比较器

ADC 模块具有一些数字比较器，可用于监视选定模拟输入的转换结果，并在转换结果处于用户指定的范围内时产生中断。仍然需要使用转换触发信号来启动转换。比较会在转换完成时自动进行。该功能通过将数字比较器模块使能位 ENDCMP （ADCCMPCONx<7>）置 1 来使能。

在模数转换结果高于或低于在 ADCCMPx 寄存器中指定的上限值和下限值时可以产生中断，用户应用程序可以利用这些中断。上限值和下限值在 DCMPHI<15:0> 位 （ADCCMPx<31:16>）和 DCMPLO<15:0> 位 （ADCCMPx<15:0>）中指定。

ADCCMPENx 寄存器中的 CMPEx 位（x = 0 至 31）用于指定通过数字比较器监视哪些模拟输入（用于前 32 个模拟输入 ANx，其中的 x = 0 至 31）。 ADCCMPCONx 寄存器用于指定产生中断的比较条件，如下：

- 当 IEBTWN = 1 时，在 DCMPLO ≤ ADCDATA < DCMPHI 时产生中断
- 当 IEHIHI = 1 时，在 DCMPHI ≤ ADCDATA 时产生中断
- 当 IEHILO = 1 时，在 ADCDATA < DCMPHI 时产生中断
- 当 IELOHI = 1 时，在 DCMPLO ≤ ADCDATA 时产生中断
- 当 IELOLO = 1 时，在 ADCDATA < DCMPLO 时产生中断

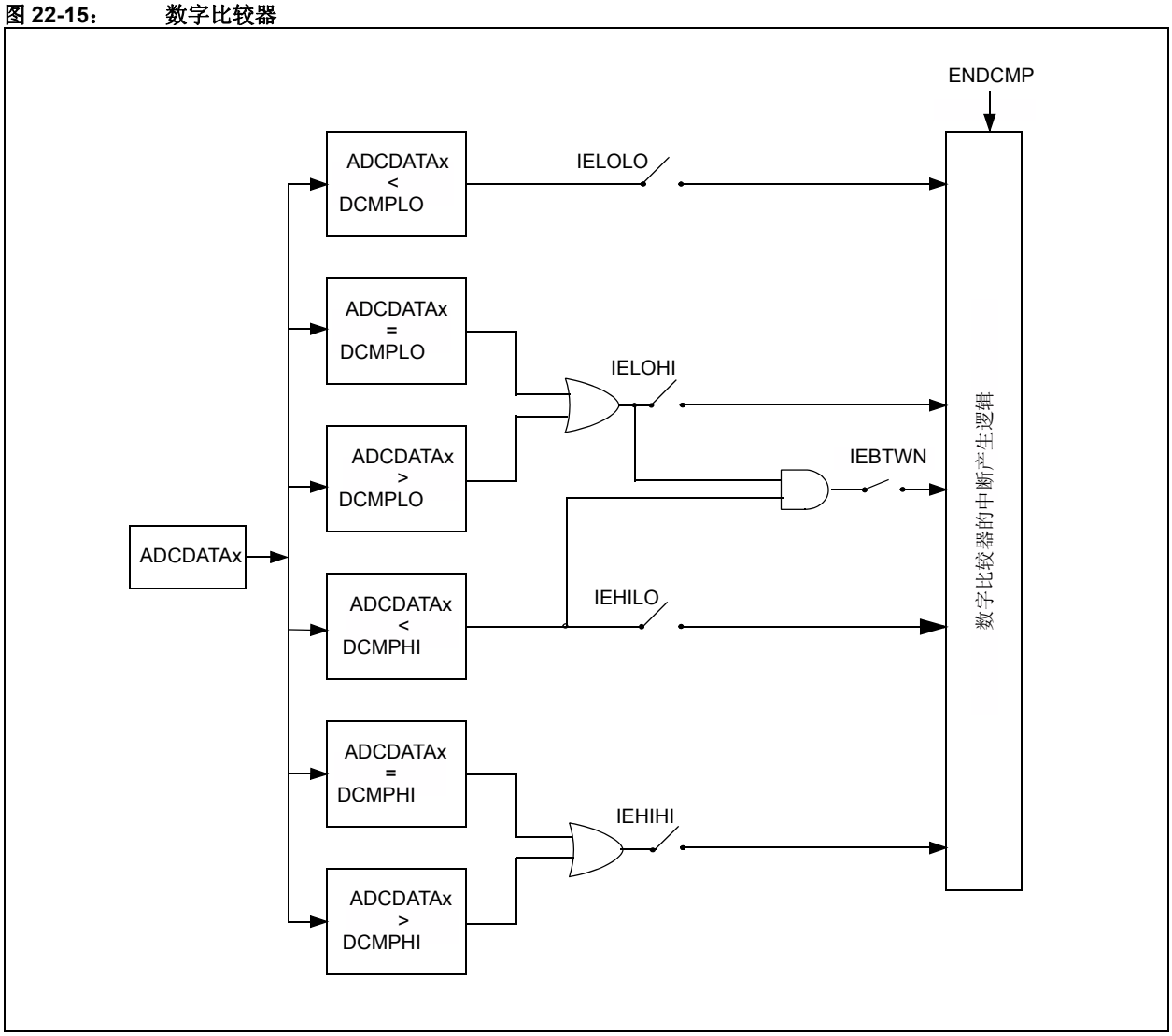
图 22-15 给出了比较器事件发生的图示。在 ADC 模块生成转换结果时，会将转换结果提供给比较器。比较器使用 ADCIMCONx 寄存器的 DIFFx 和 SIGNx 位（取决于使用的模拟输入）来确定所使用的数据格式，并相应地选择比较是有符号还是无符号。此外，全局 ADC 设置（通过 FRACT 位 （ADCCON1<23>）指定）也用于设置小数或整数格式。数字比较器会将 ADC 结果与 ADCCMPx 寄存器中的上限值和下限值 （取决于所选的比较条件）进行比较。

根据比较器结果，可能会产生数字比较器中断事件。如果发生比较器事件，检测到数字比较器中断事件状态位 DCMPED （ADCCMPCONx<5>）会置 1，模拟输入标识 （ID）位 AINID<4:0>（ADCCMPCONx<12:8>）会自动发生更新，使用户应用程序可以知道是哪个模拟输入产生了中断事件。

注 1：	数字比较器模块仅支持前 32 个模拟输入 （AN0 至 AN31）。
2：	用户软件必须设置包含在 ADCCMPx 寄存器中的值的格式，使之与转换数据格式匹配，即设置有符号或无符号，以及小数或整数。

 © 2015 Microchip Technology Inc.

图 22-15： 数字比较器


ENDCMP
ADCDATAx
<
DCMPLO
IELOLO
ADCDATAx
=
DCMPLO
IELOHI
ADCDATAx
>
DCMPLO
IEBTWN
ADCDATAx
ADCDATAx
<
DCMPHI
IEHILO
ADCDATAx
=
DCMPHI
IEHIHI
ADCDATAx
>
DCMPHI
数字比较器的中断产生逻辑

例 22-4： ADC 数字比较器

```
int main(int argc, char** argv) {
    int result = 0, eventFlag = 0;

    /* Configure ADCCON1 */
    ADCCON1 = 0;                  // No ADCCON1 features are enabled including: Stop-in-Idle,
                                  // turbo, CVD mode, Fractional mode and scan trigger source.
    ADCCON1bits.SELRES = 3;       // ADC7 resolution is 12 bits
    ADCCON1bits.STRGSRC = 0;      // No scan trigger.

    /* Configure ADCCON2 */
    ADCCON2bits.SAMC = 5;         // ADC7 sampling time = 5 * TAD7
    ADCCON2bits.ADCDIV = 1;       // ADC7 clock freq = TAD7

    /* Initialize warm up time register */
    ADCANCON = 0;
    ADCANCONbits.WKUPCLKCNT = 5;  // Wakeup exponent = 32 * TADx

    /* Clock setting */
    ADCCON3 = 0;
    ADCCON3bits.ADCSEL = 0;       // Select input clock source
    ADCCON3bits.CONCLKDIV = 1;    // Control clock frequency is half of input clock
    ADCCON3bits.VREFSEL = 0;      // Select AVdd and AVss as reference source

    /* No selection for dedicated ADC modules, no pre sync trigger, not sync sampling */
    ADCTRGMODEbits = 0;

    /* Select ADC input mode */
    ADCIMCON1bits.SIGN8 = 0;      // unsigned data format
    ADCIMCON1bits.DIFF8 = 0;      // Single ended mode

    /* Configure ADCGIRQENx */
    ADCGIRQEN1 = 0;               // No interrupts are used
    ADCGIRQEN2 = 0;

    /* Configure ADCCSSx */
    ADCCSS1 = 0;                  // No scanning is used
    ADCCSS2 = 0;

    /* Configure ADCCMPCONx */
    ADCCMP1 = 0;                  // Clear the register
    ADCCMP1bits.DCMPHI = 0xC00;   // High limit is a 3072 result.
    ADCCMP1bits.DCMPLO = 0x500;   // Low limit is a 1280 result.
    ADCCMPCON1bits.IEBTWN = 1;    // Create an event when the measured result is
                                  // >= low limits and < high limit.
    ADCCMPEN1 = 0;                // Clear all enable bits
    ADCCMPEN1bits.CMPE8 = 1;      // set the bit corresponding to AN8
    ADCCMPCON1bits.ENDCMP = 1;    // enable comparator
    ADCCMPCON2 = 0;
    ADCCMPCON3 = 0;
    ADCCMPCON4 = 0;
    ADCCMPCON5 = 0;
    ADCCMPCON6 = 0;

    /* Configure ADCFLTRx */
    ADCFLTR1 = 0;                 // No oversampling filters are used.
    ADCFLTR2 = 0;
    ADCFLTR3 = 0;
    ADCFLTR4 = 0;
    ADCFLTR5 = 0;
    ADCFLTR6 = 0;
```

 © 2015 Microchip Technology Inc.

例 22-4： ADC 数字比较器 （续）

```
    /* Set up the trigger sources */
    ADCTRG3bits.TRGSRC8 = 3;          // Set AN8 (Class 2) to trigger from scan source

    /* Early interrupt */
    ADCEIEN1 = 0;                 // No early interrupt
    ADCEIEN2 = 0;

    /* Turn the ADC on */
    ADCCON1bits.ON = 1;

    /* Wait for voltage reference to be stable */
    while(!ADCCON2bits.BGVRRDY);      // Wait until the reference voltage is ready
    while(ADCCON2bits.REFFLT);        // Wait if there is a fault with the reference voltage

    /* Enable clock to analog circuit */
    ADCANCONbits.ANEN7 = 1;           // Enable the clock to analog bias

    /* Wait for ADC to be ready */
    while(!ADCANCONbits.WKRDY7);      // Wait until ADC7 is ready

    /* Enable the ADC module */
    ADCCON3bits.DIGEN7 = 1;           // Enable ADC7

    while(1) {
        /* Trigger a conversion */
        ADCCON3bits.GSWTRG = 1;

        while (ADCDSTAT1bits.ARDY8 == 0);
        /* fetch the result */
        result = ADCDATA8;

        /* Note: It is not necessary to fetch the result for the digital
        * comparator to work. In this example we are triggering from
        * software so we are using the ARDY8 to gate our loop. Reading the
        * data clears the ARDY bit.
        */
        /* See if we have a comparator event*/
        if (ADCCMPCON1bits.DCMPED == 1) {
            eventFlag = 1;
            /*
            * Process results here
            */
        }
    }
    return (1);
}
```

22.5.2 过采样数字滤波器

ADC 模块最多支持 6 个过采样数字滤波器。过采样数字滤波器包含一个累加器和一个抽取器（下采样器），它们一起作为一个低通滤波器工作。通过以高于所需速率的采样率对模拟输入进行采样，然后通过过采样数字滤波器处理数据，可以提高 ADC 模块的有效分辨率，但代价是转换吞吐率会降低。

要获得 x 位的额外分辨率，所需的采样数 （高于奈奎斯特速率） = $(2^x)^2$：

- 4 倍过采样可以获得 1 位的额外分辨率 （共 13 位分辨率）
- 16 倍过采样可以获得 2 位的额外分辨率 （共 14 位分辨率）
- 64 倍过采样可以提供 3 位的额外分辨率 （共 15 位分辨率）
- 256 倍过采样可以提供 4 位的额外分辨率 （共 16 位分辨率）

此外，数字滤波器还具有平均模式，在该模式下它会对采样进行累加，然后除以采样数。

> **注 1：** 只有 1 类和 2 类模拟输入可以接入数字滤波器。因此， CHNLID<4:0> 位为 5 位宽（0 至 31）。
>
> **2：** 在使用共用 ADC 模块对 2 类输入进行滤波的情况下，在连串转换处理（重复触发，直到获得过采样所需的所有数据）期间，优先级较高的 ADC 输入仍然可以处理转换；优先级较低的 ADC 转换请求会被保持为等待状态，直到滤波器连串序列完成为止。
>
> **3：** 如果在数字滤波器序列期间出现优先级较高的请求，它们会延迟滤波处理的完成时间。该延时可能会影响结果的精度，因为多个采样将不是连续的。用户应对过采样滤波器的启动触发作出安排，使得在发生触发的时段中预期不会产生来自较高优先级 ADC 转换请求的中断。

用户应用程序应通过配置以下位来执行过采样转换：

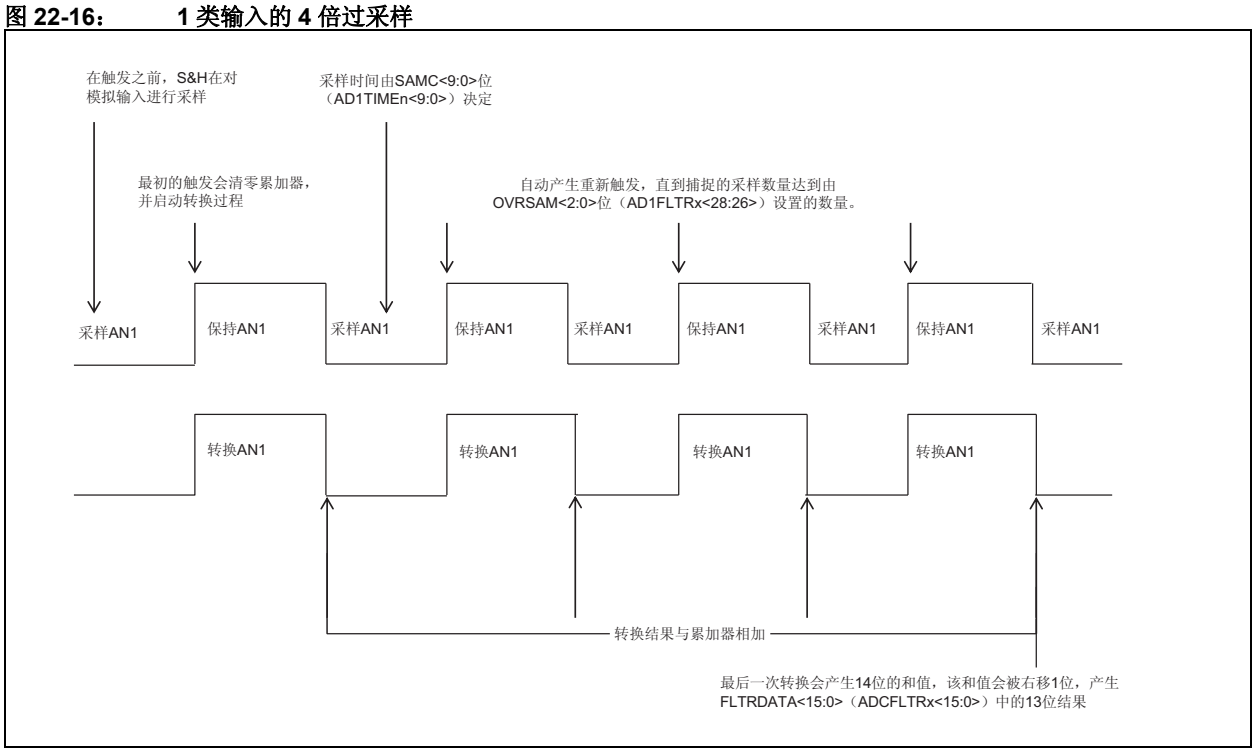
- 通过 ADC 滤波器寄存器 （ADCFLTRx<28:26>）中的过采样滤波器过采样率（OVRSAM<2:0>）位选择过采样量
- 使用 DFMODE 位 （ADCFLTRx<29>）将滤波器模式设置为过采样模式或平均模式
- 如果滤波器设置为平均模式，数据格式设置为小数 （FRACT 位），则通过置 1 或清零 DATA16EN 位 （ADCFLTRx<30>）来设置输出分辨率
- 设置后续采样的采样时间：
  - 如果使用 1 类输入，则使用 SAMC<9:0> 位 （ADCxTIME<9:0>）选择重复性转换的采样时间
  - 如果使用 2 类输入，则使用 SAMC<9:0> 位 （ADCCON2<25:16>）选择采样时间
- 通过配置模拟输入 ID 选择位 CHNLID<4:0> （ADCFLTRx<20:16>）选择要进行过采样的特定模拟输入
- 如果需要，通过将累加器滤波器全局中断允许位 AFGIEN （ADCFLTRx<25>）置 1，在全局 ADC 中断中包含过采样滤波器中断事件
- 通过将过采样滤波器累加器使能位 AFEN （ADCFLTRx<31>）置 1 来使能过采样滤波器

在配置数字滤波器模块之后，滤波器的控制逻辑会等待外部触发信号来启动处理。要进行过采样的模拟输入的触发信号会使累加器清零并启动第一个转换。触发信号会强制将触发敏感性设为电平模式，并且只要滤波器还需要采集用户通过 OVRSAM<2:0> 位（ADCFLTRx<28:26>）指定的采样数，就会一直将触发信号本身强制设为 1。每个采集的采样之间的延时由所设置的采样时间（对于 1 类输入为 ADCxTIME 寄存器中的 SAMC<9:0> 位，对于 2 类输入为 ADCCON2 寄存器中的 SAMC<9:0> 位）和转换时间决定。接收并处理由 OVRSAM<2:0> 设置的所需数量之后，数据会被存储在 FLTRDATA<15:0> 位 （ADCFLTRx<15:0>）中， AFRDY 位 （ADCFLTRx<24>）会置 1，并产生中断 （如果允许）。

 © 2015 Microchip Technology Inc.

图 22-16 给出了对 1 类输入进行 4 倍过采样的图示。在触发之前，1 类输入在对输入信号进行采样。触发信号会启动第一个转换。当转换完成时，ADC 会进入采样模式。当采样时间结束时，会发生新的转换序列。在转换每个采样之后，会将它与累加器相加。该序列一直重复，直到累加的采样数达到由 OVRSAM<2:0> 位指定的采样数。当转换最后一个采样后，会将其值与累加器相加。结果进行右移，然后储存在 FLTRDATA<15:0> 位中。

图 22-16： 1 类输入的 4 倍过采样

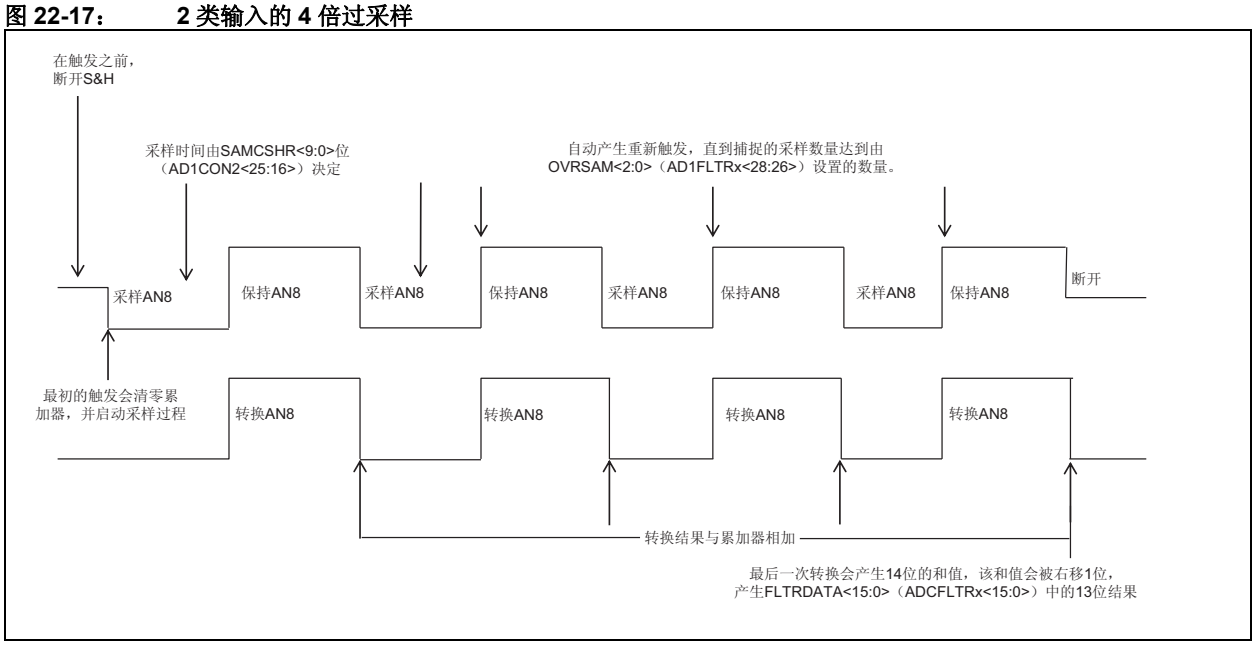


如果有多个 1 类输入使用滤波，它们将全部相互独立地并行工作。

图 22-17 给出了使用 2 类输入进行 4 倍过采样的图示。触发 2 类输入会启动采样，采样将持续由 SAMC<9:0> 位定义的时间长度。过采样逻辑产生的重新触发使用 SAMC<9:0> 位来设置采样时间。

由于 2 类输入使用共用 S&H，所以过采样会阻塞优先级较低的 2 类和 3 类触发。优先级较高的 2 类触发会完全破坏过采样处理，因此应完全避免。相同的优先级规则适用于使用两个数字滤波器的两个 2 类输入。在这种情况下，优先级较高的输入也会在连串模式下使用共用 ADC 模块，并阻止优先级较低的输入使用共用 ADC。只有在优先级较高的输入获取所需的所有采样之后，优先级较低的输入才能使用共用 ADC 来采集采样，进行它自己的数字滤波。

图 22-17： 2 类输入的 4 倍过采样



例 22-5： ADC 数字过采样滤波器

```
int main(int argc, char** argv) {
    int result;

    /* Configure ADCCON1 */
    ADCCON1 = 0;           // No ADCCON1 features are enabled including: Stop-in-Idle, turbo,
                           // CVD mode, Fractional mode and scan trigger source.

    /* Configure ADCCON2 */
    ADCCON2 = 0;           // Since, we are using only the Class 1 inputs, no setting is
                           // required for ADCDIV

    /* Initialize warm up time register */
    ADCANCON = 0;
    ADCANCONbits.WKUPCLKCNT = 5;  // Wake-up exponent = 32 * TADx

    /* Clock setting */
    ADCCON3 = 0;
    ADCCON3bits.ADCSEL = 0;       // Select input clock source
    ADCCON3bits.CONCLKDIV = 1;    // Control clock frequency is half of input clock
    ADCCON3bits.VREFSEL = 0;      // Select AVdd and AVss as reference source

    ADC0TIMEbits.ADCDIV = 1;      // ADC0 clock frequency is half of control clock = TAD0
    ADC0TIMEbits.SAMC = 5;        // ADC0 sampling time = 5 * TAD0
    ADC0TIMEbits.SELRES = 3;      // ADC0 resolution is 12 bits

    /* Select analog input for ADC modules, no pre sync trigger, not sync sampling */
    ADCTRGMODEbits.SH0ALT = 0;    // ADC0 = AN0

    /* Select ADC input mode */
    ADCIMCON1bits.SIGN0 = 0;      // unsigned data format
    ADCIMCON1bits.DIFF0 = 0;      // Single ended mode

    /* Configure ADCGIRQENx */
    ADCGIRQEN1 = 0;               // No interrupts are used
    ADCGIRQEN2 = 0;

    /* Configure ADCCSSx */
    ADCCSS1 = 0;                  // No scanning is used
    ADCCSS2 = 0;
```

 © 2015 Microchip Technology Inc.

例 22-5： ADC 数字过采样滤波器 （续）

```
    /* Configure ADCCMPCONx */
    ADCCMPCON1 = 0;                // No digital comparators are used. Setting the ADCCMPCONx
    ADCCMPCON2 = 0;                // register to '0' ensures that the comparator is disabled.
    ADCCMPCON3 = 0;                // Other registers are 該on't care’ .
    ADCCMPCON4 = 0;
    ADCCMPCON5 = 0;
    ADCCMPCON6 = 0;

    /* Configure ADCFLTRx */
    ADCFLTR1 = 0;                  // Clear all bits
    ADCFLTR1bits.CHNLID = 0;       // Use AN0 as the source
    ADCFLTR1bits.OVRSAM = 3;       // 16x oversampling
    ADCFLTR1bits.DFMODE = 0;       // Oversampling mode
    ADCFLTR1bits.AFEN = 1;         // Enable filter 1
    ADCFLTR2 = 0;                  // Clear all bits
    ADCFLTR3 = 0;
    ADCFLTR4 = 0;
    ADCFLTR5 = 0;
    ADCFLTR6 = 0;

    /* Set up the trigger sources */
    ADCTGSNSbits.LVL0 = 0;         // Edge trigger
    ADCTRG1bits.TRGSRC0 = 1;       // Set AN0 to trigger from software.

    /* Turn the ADC on */
    ADCCON1bits.ON = 1;

    /* Wait for voltage reference to be stable */
    while(!ADCCON2bits.BGVRRDY);  // Wait until the reference voltage is ready
    while(ADCCON2bits.REFFLT);    // Wait if there is a fault with the reference voltage

    /* Enable clock to analog circuit */
    ADCANCONbits.ANEN0 = 1;        // Enable the clock to analog bias and digital control

    /* Wait for ADC to be ready */
    while(!ADCANCONbits.WKRDY0);  // Wait until ADC0 is ready

    /* Enable the ADC module */
    ADCCON3bits.DIGEN0 = 1;        // Enable ADC0

    while(1) {
        /* Trigger a conversion */
        ADCCON3bits.GSWTRG = 1;

        /* Wait for the oversampling process to complete */
        while (ADCFLTR1bits.AFRDY == 0);
        /* fetch the result */
        result = ADCFLTR1bits.FLTRDATA;

        /*
        * Process result Here
        *
        * Note 1: Loop time determines the sampling time for the first sample.
        * remaining samples sample time is determined by set sampling + conversion time.
        *
        * Note 2: The first 5 samples may have reduced accuracy.
        *
        */
    }
    return (1);
}
```

22.5.3 FIFO 数据输出

专用 ADC 模块（ADC0 至 ADC6）可将输出数据存储到 FIFO 中。在使用 ADC 以极高吞吐率转换信号时，这可能会很有用。ADCFIFO 寄存器是 FIFO 的读取端口。FIFO 为 128 级深，本身是循环的。

需要使用 FIFO 的 ADC 模块通过 ADCxEN 位（ADCFSTAT<30:24>）进行选择。如果有多个 ADC 模块需要使用 FIFO，可以在 ADCxEN 中将所有这些相应的位置 1。从 FIFO 中读取数据时，应先读取 ADCID<2:0> 位（ADCFSTAT<2:0>），然后再从 ADCFIFO 寄存器中读取数据。通过这种方式，用户应用程序可以知道FIFO中的数据来源于哪个ADC模块。FEN位（ADCFSTAT31）置 1 可以使能 FIFO 操作。当 FIFO 中有数据可用时，FRDY 位（ADCFSTAT<22>）会置 1 并保持置 1，直到 FIFO 被完全读取并且没有任何数据为止。如果需要，可以通过将 FIEN 位（ADCFSTAT23）置 1 来产生中断。

> **注 1：** FIFO 深度取决于器件。请参见具体器件的数据手册。
>
> **2：** 从FIFO中获取数据时，用户代码应在每个连续读操作过程中同时读取ADCID<2:0>位和 ADCFIFO 寄存器。

例 22-6： 对 1 类输入使用 ADC FIFO

```
int main(int argc, char** argv) {
    int result[128], index = 0;

    /* Configure ADCCON1 */
    ADCCON1 = 0;           // No ADCCON1 features are enabled including: Stop-in-Idle, turbo,
                           // CVD mode, Fractional mode and scan trigger source.

    /* Configure ADCCON2 */
    ADCCON2 = 0;           // Since, we are using only the Class 1 inputs, no setting is
                           // required for ADCDIV

    /* Initialize warm up time register */
    ADCANCON = 0;
    ADCANCONbits.WKUPCLKCNT = 5;  // Wakeup exponent = 32 * TADx

    /* Clock setting */
    ADCCON3 = 0;
    ADCCON3bits.ADCSEL = 0;       // Select input clock source
    ADCCON3bits.CONCLKDIV = 1;    // Control clock frequency is half of input clock
    ADCCON3bits.VREFSEL = 0;      // Select AVdd and AVss as reference source

    ADC0TIMEbits.ADCDIV = 1;      // ADC0 clock frequency is half of control clock = TAD0
    ADC0TIMEbits.SAMC = 5;        // ADC0 sampling time = 5 * TAD0
    ADC0TIMEbits.SELRES = 3;      // ADC0 resolution is 12 bits

    /* Select analog input for ADC modules, no presync trigger, not sync sampling */
    ADCTRGMODEbits.SH0ALT = 0;    // ADC0 = AN0

    /* Select ADC input mode */
    ADCIMCON1bits.SIGN0 = 0;      // unsigned data format
    ADCIMCON1bits.DIFF0 = 0;      // Single ended mode

    /* Configure ADCGIRQENx */
    ADCGIRQEN1 = 0;               // No interrupts are used
    ADCGIRQEN2 = 0;

    /* Configure ADCCSSx */
    ADCCSS1 = 0;                  // No scanning is used
    ADCCSS2 = 0;

    /* Configure ADCCMPCONx */
    ADCCMPCON1 = 0;               // No digital comparators are used. Setting the ADCCMPCONx
    ADCCMPCON2 = 0;               // register to '0' ensures that the comparator is disabled.
    ADCCMPCON3 = 0;               // Other registers are ‘don't care’ .
    ADCCMPCON4 = 0;
    ADCCMPCON5 = 0;
```

 © 2015 Microchip Technology Inc.

例 22-6： 对 1 类输入使用 ADC FIFO （续）

```
    /* Configure ADCFLTRx */
    ADCFLTR1 = 0;                 // Clear all bits
    ADCFLTR2 = 0;
    ADCFLTR3 = 0;
    ADCFLTR4 = 0;
    ADCFLTR5 = 0;
    ADCFLTR6 = 0;

    /* Set up the trigger sources */
    ADCTGSNSbits.LVL0 = 0;        // Edge trigger
    ADCTRG0bits.TRGSRC0 = 1;      // Set AN0 to trigger from software.

    /* Early interrupt */
    ADCEIEN1 = 0;                 // No early interrupt
    ADCEIEN2 = 0;

    /* Set FIFO */
    ADCFSTAT = 0;                 // Clear all bits
    ADCFSATbits.ADC0EN = 1;       // Select ADC0
    ADCFSATbits.FEN = 1;          // Enable FIFO

    /* Turn the ADC on */
    ADCCON1bits.ON = 1;

    /* Wait for voltage reference to be stable */
    while(!ADCCON2bits.BGVRRDY);  // Wait until the reference voltage is ready
    while(ADCCON2bits.REFFLT);    // Wait if there is a fault with the reference voltage

    /* Enable clock to analog circuit */
    ADCANCONbits.ANEN0 = 1;       // Enable the clock to analog bias and digital control

    /* Wait for ADC to be ready */
    while(!ADCANCONbits.WKRDY0);  // Wait until ADC0 is ready

    /* Enable the ADC module */
    ADCCON3bits.DIGEN0 = 1;       // Enable ADC0

    while(1) {
        /* Trigger a conversion */
        ADCCON3bits.GSWTRG = 1;

        /* Wait the conversions to complete and data available in FIFO */
        while (ADCFSTATbits.FRDY == 0);

        /* Once FIFO bit is set, read all possible data, until bit is clear */
        while(ADCFSTATbits.FRDY)
        {
            /* If data overflow occurred in FIFO, break read process */
            if(ADCFSTATbits.FWROVERR)
            {
                break;
            }
            /* if ADC ID is really '0', read data.
            This is more appropriate when multiple ADC modules are
            using the FIFO, therefore, reading the ADC ID will
            identify the ADC and data relation */
            if(ADCFSTATbits.ADCID == 0)
            {
                /* Read data from FIFO */
                result[index] = ADCFIFO;
                /* For each FIFO read, ADCFSTATbits.FCNT will keep reducing */
                index++;
                if(index >= 128)
                index = 0;
            }
        }
        /*
        * Process results here
        *
        */
    }
    return (1);
}
```

22.5.4 基于专用 DMA 的数据输出

根据不同的器件，每个专用 ADC 模块具有一个 DMA 引擎，可用于将转换数据直接存储到系统存储器（RAM）中。要确定该特性对于您所用器件是否可用，请参见具体器件数据手册中的**“ADC”**章节。

如果 ADCDMASTAT 寄存器中的 DMACNTEN 位置 1，则 DMA 引擎还会将 ADC 模块当前采样计数写入系统 RAM 中，从 ADCCNTB 寄存器中指定的地址处开始。

用于存储的缓冲区的大小由 DMABL<2:0> 位 （ADCCON1<2:0>）指定。此外，存储输出数据的 RAM 地址由 ADCDMAB 寄存器中的 DMABADDR<31:0> 位指定。每个专用 ADC 模块具有两个缓冲区：缓冲区 A 和缓冲区 B。

由于使用 DMABL<2:0> 来对数据数量进行计数，而每个数据为 16 位，所以每个 ADC 模块的缓冲区的实际大小 （以字节为单位）可根据以下公式得到：

$$= 2^{(ADCCON1bits.DMABL + 1)}$$

通道存储缓冲区空间在系统 RAM 中是连续的，从基址（由 ADCDMAB 寄存器给定）开始是 ADC0 缓冲区 A，紧接着是 ADC0 缓冲区 B，紧接着是 ADC1 缓冲区 A，紧接着是 ADC1 缓冲区 B，按通道编号分配的自然顺序如此类推。

表 22-6： 由多个 **ADC** 模块使用时的 **DMA** 缓冲区格式

ADC 模块 ID	缓冲区	RAM 地址空间
0	A	(DMABADDR<31:0>) + 0 至 (DMABADDR<31:0>) + 255
0	B	(DMABADDR<31:0>) + 256 至 (DMABADDR<31:0>) + 511
1	A	(DMABADDR<31:0>) + 512 至 (DMABADDR<31:0>) + 767
1	B	(DMABADDR<31:0>) + 768 至 (DMABADDR<31:0>) + 1023
2	A	(DMABADDR<31:0>) + 1024 至 (DMABADDR<31:0>) + 1279
2	B	(DMABADDR<31:0>) + 1280 至 (DMABADDR<31:0>) + 1535
⋮	⋮	⋮
6	A	(DMABADDR<31:0>) + 3072 至 (DMABADDR<31:0>) + 3327
6	B	(DMABADDR<31:0>) + 3328 至 (DMABADDR<31:0>) + 3583

如表 22-6 所示，对于每个 ADC 模块，在由于新数据覆盖旧数据而发生数据丢失之前，用户最多可以获取 128 个采样 （每个采样两个字节）。对于每个通道， DMA 引擎会在为自己分配的 RAM 空间内进行地址回绕。

ADCx （x = 0 至 6）缓冲区 A 的起始地址为：

$$= ADCDMAB + (2 * x) * 2^{(ADCCON1bits.DMABL + 1)}$$

ADCx （x = 0 至 6）缓冲区 B 的起始地址为：

$$= ADCDMAB + (2 * (x + 1)) * 2^{(ADCCON1bits.DMABL + 1)}$$

ADCCNTB 寄存器包含用户给定的地址，DMA 引擎会从该地址开始保存输出采样（输出采样已经写入每个ADC模块在系统RAM中的每个缓冲区）的当前计数（如果DMACNTEN位（ADCDMASTAT<15>）置 1）。 ADCx 会将其缓冲区 A 当前采样计数保存在地址 ((ADCCNTB) + 2 * x) 处，将其缓冲区 B 当前采样计数保存在地址 ((ADCCNTB) + (2 * x + 1)) 处。每个 ADC 缓冲区计数只需要 1 个字节的 RAM 存储空间，因为其最大值为 128。用户可以在任意时刻，通过读取由 ADCCNTB 寄存器指定的存储单元，确定每个ADC（0至6）和每个缓冲区（A和B）的可用数据数量；即使是在RAFx或RBFx位置1之前。

 © 2015 Microchip Technology Inc.

表 22-7： DMA 缓冲区计数表

ADC 模块 ID	缓冲区	采样计数
0	A	通过读取存储器（由 ADCCNTB 寄存器指定）来确定存储的采样数。
0	B	通过读取存储器 （由 ADCCNTB 寄存器指定） + 1 来确定存储的采样数。
1	A	通过读取存储器 （由 ADCCNTB 寄存器指定） + 2 来确定存储的采样数。
1	B	通过读取存储器 （由 ADCCNTB 寄存器指定） + 3 来确定存储的采样数。
2	A	通过读取存储器 （由 ADCCNTB 寄存器指定） + 4 来确定存储的采样数。
2	B	通过读取存储器 （由 ADCCNTB 寄存器指定） + 5 来确定存储的采样数。
⋮	⋮	⋮
6	A	通过读取存储器 （由 ADCCNTB 寄存器指定） + 12 来确定存储的采样数。
6	B	通过读取存储器 （由 ADCCNTB 寄存器指定） + 13 来确定存储的采样数。

在用户通过将 DIGEN*x* 位（ADCCON3 寄存器）置 1 而使能 ADC 模块（0 至 6）之后，DMA 引擎会立即将 ((ADCCNTB) + 2 * x) 地址处的 ADCx 缓冲区 A 采样计数和 ((ADCCNTB)+(2 * x + 1)) 地址处的缓冲区 B 采样计数复位为零。

22.5.4.1 DMA 乒乓模式

以下步骤介绍了用于单个 ADC 模块时的 DMA 引擎乒乓工作模式：

1. DMA 引脚将转换数据保存到缓冲区 A 在系统 RAM 中的缓冲区中
2. 当缓冲区 A 满时，所用 ADC 的 RAFx 位（ADCDMASTAT<6:0>）会置 1。同时，DMA 引擎会开始使用并将数据保存到缓冲区 B。
3. 如果选定 ADC 的 RAFIENx 位 （ADCDMASTAT<14:8>）置 1，则会产生中断。
4. 在 ISR 内，用户应用程序必须通过读取 RAFx 或 RBFx 位确定缓冲区 ID （A 或 B）。
5. 无论是哪个位置 1，用户应用程序都必须从缓冲区 ID 读取转换数据，并对转换数据执行适当的处理。
6. 当用户应用程序在处理转换数据时， DMA 引擎会将数据保存到缓冲区 B 中。
7. 当用户应用程序完成对缓冲区 A 数据的处理时，它会等待缓冲区 B 完成。

如果应用程序需要在缓冲区满之前使用存储在缓冲区中的数据，则需要执行以下步骤：

1. 为了定位保存在缓冲区中的数据，用户代码应使能 DMACNTEN 位（ADCDMASTAT<15> 寄存器）。
2. 可以从 (ADCCNTB + (2 * x + 1)) 处获取当前采样计数（假设使用的是缓冲区 B），然后再读取在 ADCDMAB 寄存器中指定的适用的存储器偏移量。
3. 当缓冲区 B 满时，所用通道的 RBFx 位会置 1。同时，DMA 引擎会开始将数据保存到缓冲区 A。

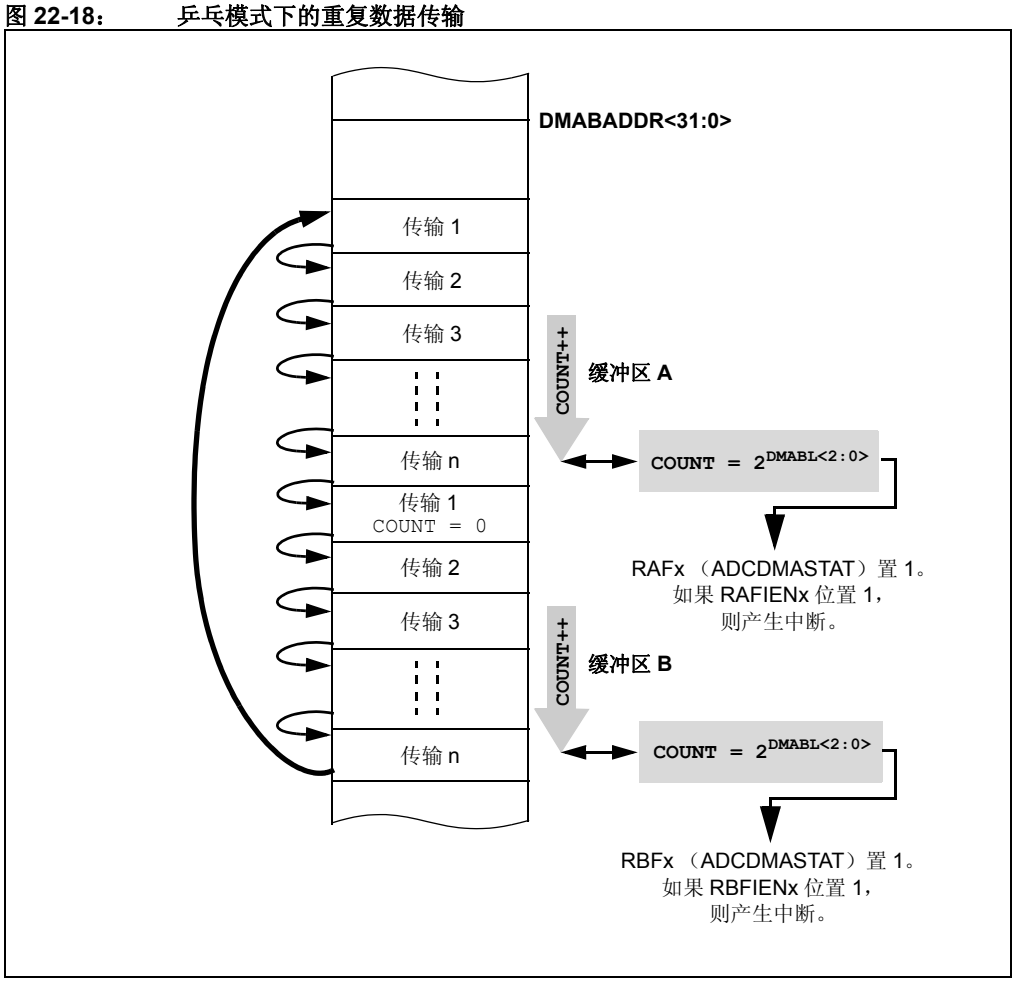
注 1： 作为一项完全留给软件的任务，需要在使能 DMA 引擎之前将缓冲区采样计数表初始化为全零。为了避免在每个缓冲区中保存第一个采样之前，缓冲区采样计数表中存在垃圾数据，应当执行该操作。

2： 读取 ADCDMASTAT 寄存器会清零该寄存器中的所有状态位。由于专用 ADC 模块完全相互独立工作，所以多个状态位可以同时置 1。用户必须读取 ADCDMASTAT 寄存器，并将其内容保存到一个独立变量中来进行逻辑位电平操作，以确定哪些专用 ADC 模块缓冲区在该时刻已满。

图 22-18 给出了乒乓模式下连续操作的图示。

 © 2015 Microchip Technology Inc.

图 22-18： 乒乓模式下的重复数据传输



22.5.5 电容分压器 （CVD）模式

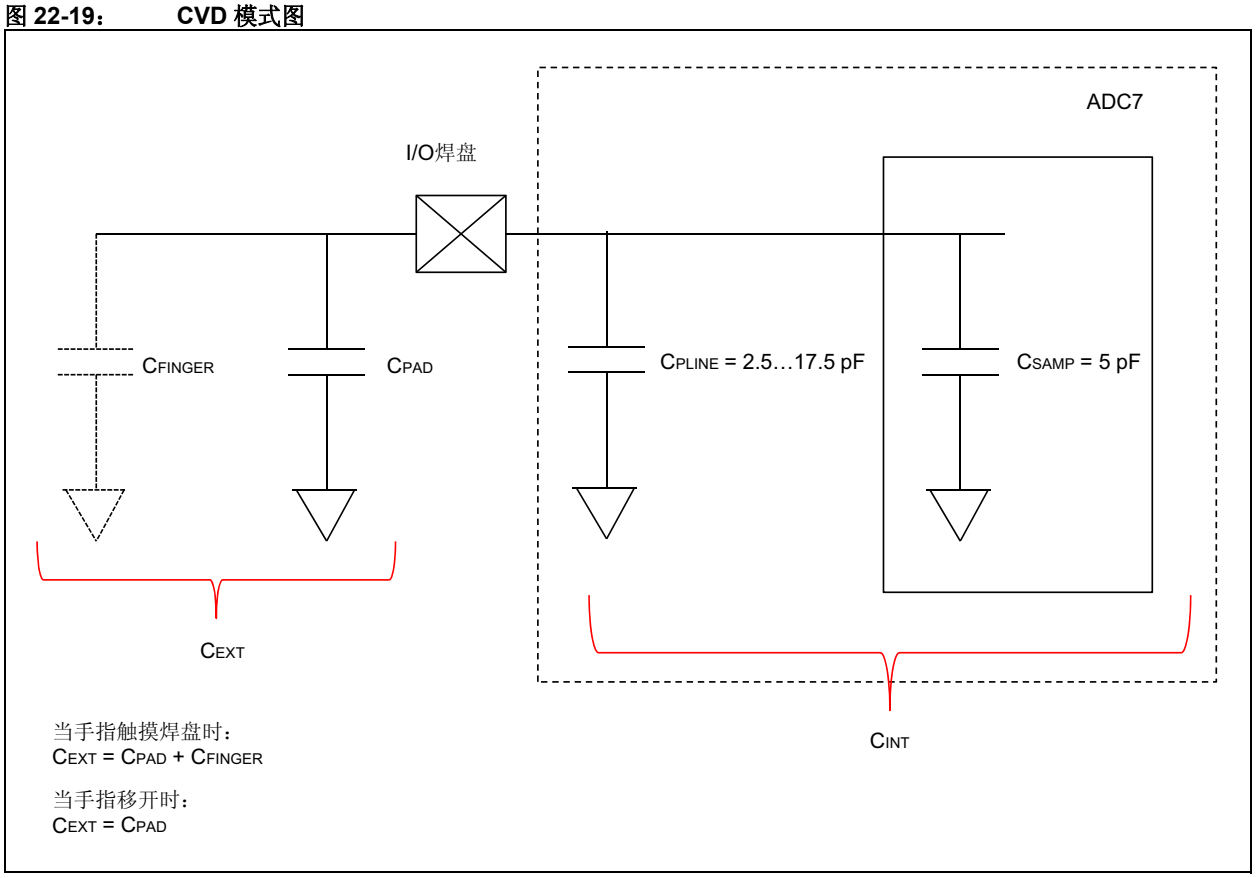
ADC 具有 CVD 模式，该模式可以用于在触摸传感器应用中检测触摸事件。CVD 测量的原理是，先在两个电容之间实现电荷平衡，然后测量电压。

在触摸传感应用中，焊盘附近出现手指会改变焊盘的电容。CVD 模块可以感应到这种电容变化，从而检测手指的出现 （以及随后的消失）。

注：	只有共用模拟输入 （2 类和 3 类）可以用于 CVD 测量。

- 当手指触摸传感器焊盘时，外部电容 = CEXT = CPAD + CFINGER
- 焊盘的外部电容 （手指没有触摸时） = CEXT = CPAD
- 内部电容 = CINT = CPLINE + CSAMP

图 22-19： CVD 模式图

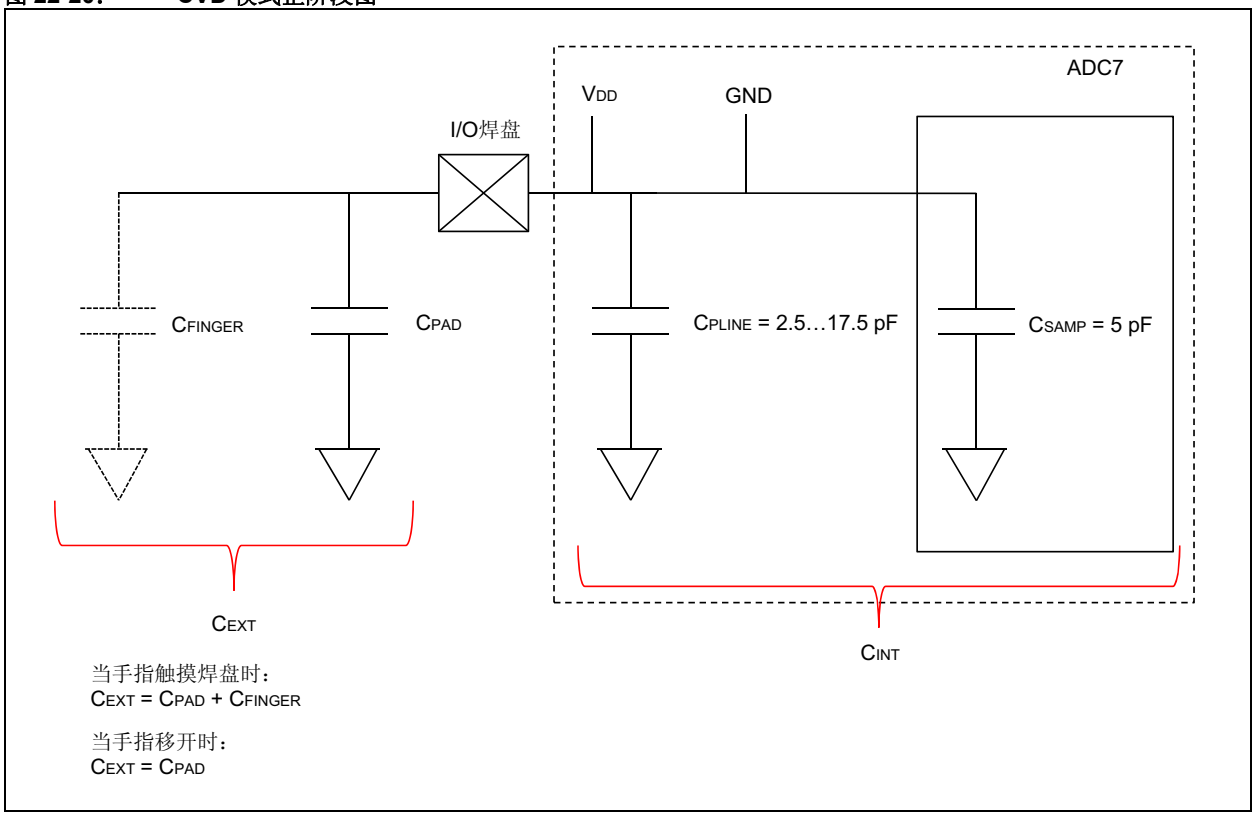


 © 2015 Microchip Technology Inc.

CVD 测量包括两个阶段：正阶段和负阶段。

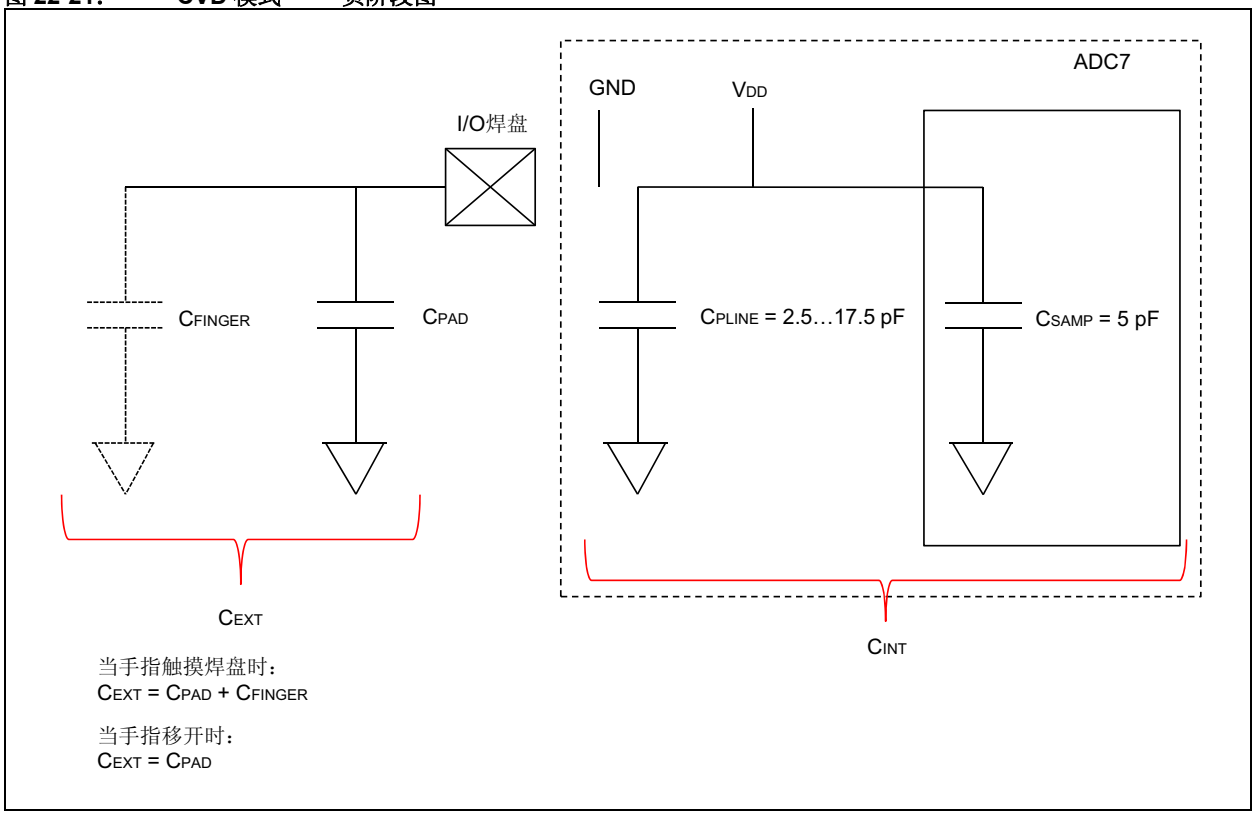
正阶段：CEXT 与 VDD 连接， CINT 接地 （GND）（如图 22-20 所示）。然后，将两个电容并联，持续时间为采样时间 （通过 SAMC<9:0> 位 （ADCCON2<25:16>）设置）的一半。在一半采样时间完成后，通过 ADC 模块对电压进行转换，并命名为 VINP。

图 22-20： CVD 模式正阶段图



负阶段：CEXT 接地， **CINT** 与 **VDD** 连接，如图 22-21 所示。类似于正阶段，测量负阶段的电压，并命名为 **VINN**。

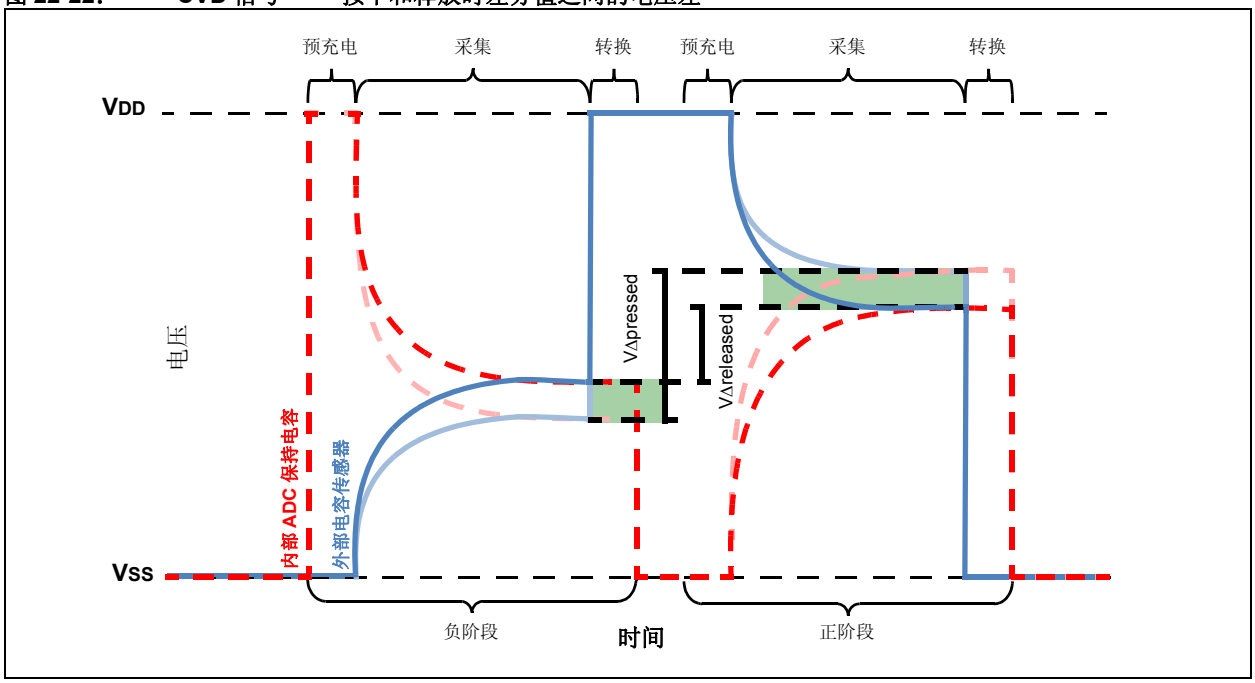
图 22-21： CVD 模式 —— 负阶段图



初稿
© 2015 Microchip Technology Inc.

CVD 模块会在内部计算 VINP 和 VINN 之间的差值，并将数据以有符号格式存储在 CVDDATA<15:0> 位 （ADCCMPCON1<31:16>）中。图 22-22 中显示了在触摸事件期间，外部电容升高是如何导致 (VINP - VINN) 高于无触摸条件的 （外部电容降低）。

图 22-22： CVD 信号 —— 按下和释放时差分值之间的电压差



公式 22-3 显示了关系式 (VINP - VINN)。

公式 22-3： VINP 和 VINN 的关系

$$(V_{IN}P - V_{IN}N) = V_{DD} \cdot \left(\frac{(C_{EXT} - C_{INT})}{(C_{EXT} + C_{INT})}\right)$$

在触摸条件下， CEXT 会升高， (VINP - VINN) 也会升高。

在无触摸条件下， CEXT 会降低， (VINP - VINN) 会减小。

数字比较器 1 与 CVD 相联结，通过将 IEHIHI 位（ADCCMPCON13）置 1，它可以用于在检测到触摸时产生比较器事件和中断。数字比较器可以在测量的 (VINP - VINN) 高于在 DCMPHI<15:0> 位（ADCCMP1<31:16>）中设置的值时产生事件。当通过读取 DCMPED 位（ADCCMPCON1<5>）检测到比较器事件时，或者在 ISR 内，可以从 AINID<5:0> 位 （ADCCMPCON1<13:8>）读取导致触摸事件的模拟输入。

为了确保最高的灵敏度， CINT 的值应与 CPAD 的值相近，这可以通过使用 CVDCPL<2:0> 位（ADCCON2<28:26>）适当地选择 CPLINE 电容的值来实现。

> **注：** 在通过将 CVDEN 位置 1 来使能 CVD 之前，应通过清零 TRGSRC<4:0> 和 STRGSRC<4:0> 位禁止所有 2 类和 3 类输入的外部触发。

例 22-7: ADC CVD 模式

```
unsigned char touchDetected = 0;

void __ISR(_ADC_DC1_VECTOR, IPL3AUTO) _IntHandlerDrvAdc(void)
{
    IFS1bits.ADCDC1IF = 0;
    /* Check and clear event flag for comparator-1 */
    if(ADCCMPCON1bits.DCMPED == 1)
    {
        /* Verify if comparator event really due to AN21 */
        if(ADCCMPCON1bits.AINID == 21)
        {
            touchDetected = 1;
        }
    }
}

int main (void)
{

    /* Configure ADCCON1 */
    ADCCON1bits.SELRES = 3; // ADC resolution is 12 bits
    ADCCON1bits.STRGSRC = 0; // No scan trigger.
    ADCCON1bits.AICPMPEN = 0;
    ADCCON1bits.FRACT = 0;// Integer format

    /* Configure ADCCON2 */
    ADCCON2bits.SAMC = 32; // ADC7 sampling time
    ADCCON2bits.ADCDIV = 4; // ADC7 clock freq is half of control clock = TAD7
    ADCCON2bits.CVDCPL = 2;// 5pF is the CVD capacitor value selected

    /* Initialize warm up time register */
    ADCANCON = 0;
    ADCANCONbits.WKUPCLKCNT = 5; // Wakeup exponent = 32 * TADx

    /* Clock setting */
    ADCCON3 = 0;
    ADCCON3bits.ADCSEL = 0; // Select input clock source
    ADCCON3bits.CONCLKDIV = 1; // Control clock frequency is half of input clock
    ADCCON3bits.VREFSEL = 0; // Select AVdd and AVss as reference source

    /* No selection for dedicated ADC modules, no pre sync trigger, not sync sampling */
    ADCTRGMODE = 0;
    /* Select ADC input mode */
    ADCIMCON2bits.SIGN21 = 1; // signed data format
    ADCIMCON2bits.DIFF21 = 0; // Single ended mode

    /* Configure ADCGIRQENx */
    ADCGIRQEN1 = 0; // No interrupts are used.
    ADCGIRQEN2 = 0;

    /* Configure ADCCSSx */
    ADCCSS1 = 0; // No scanning is used
    ADCCSS2 = 0;
    ADCCSS1bits.CSS21 = 1; // AN21 (Class 3) set for CVD

    /* Configure ADCCMPCONx */
    ADCCMP1 = 0; // Clear the register
    ADCCMP1bits.DCMPHI = 1600;// High limit, depends on touch pad layout and selected cap,
sample time etc.
    ADCCMP1bits.DCMPLO = 0; // Low limit, not very important for sensing touch event
    ADCCMPCON1bits.IEHIHI = 1; // When touched, the CVDDATA > HI_LIMIT(DCMPHI)
```

 © 2015 Microchip Technology Inc.

例 22-7： ADC CVD 模式 （续）

```
    ADCCMPCON1bits.IEBTWN = 0;
    ADCCMPCON1bits.IEHILO = 0;
    ADCCMPCON1bits.IELOHI = 0;
    ADCCMPCON1bits.IELOLO = 0;
    ADCCMPCON1bits.DCMPGIEN = 1;// enable interrupt
    ADCCMPCON1bits.ENDCMP = 1; // enable comparator
    ADCCMPEN1 = 0; // Clear all enable bits, as it is irrelevant in CVD mode
    ADCCMPCON2 = 0;
    ADCCMPCON3 = 0;
    ADCCMPCON4 = 0;
    ADCCMPCON5 = 0;
    ADCCMPCON6 = 0;

    /* Configure ADCFLTRx */
    ADCFLTR1 = 0; // No oversampling filters are used.
    ADCFLTR2 = 0;
    ADCFLTR3 = 0;
    ADCFLTR4 = 0;
    ADCFLTR5 = 0;
    ADCFLTR6 = 0;

    /* Set up the trigger sources */
    ADCTRGSNS = 0;
    ADCTRG1 = 0;
    ADCTRG2 = 0;
    ADCTRG3 = 0;

    /* Early interrupt */
    ADCEIEN1 = 0; // No early interrupt
    ADCEIEN2 = 0;

    /* Turn the ADC on */
    ADCCON1bits.ON = 1;
    /* Wait for voltage reference to be stable */
    while(!ADCCON2bits.BGVRRDY); // Wait until the reference voltage is ready
    while(ADCCON2bits.REFFLT); // Wait if there is a fault with the reference voltage
    /* Enable clock to analog circuit */
    ADCANCONbits.ANEN7 = 1; // Enable the clock to analog bias
    /* Wait for ADC to be ready */
    while(!ADCANCONbits.WKRDY7); // Wait until the ADC7 is ready

    /* Enable interrupt setting for digital comparator-1 */
    IFS1bits.ADCDC1IF = 0;
    IEC1bits.ADCDC1IE = 1;
    IPC11bits.ADCDC1IP = 3;
    IPC11bits.ADCDC1IS = 3;

    INTCONbits.MVEC = 1;

    __builtin_enable_interrupts();

    /* Turn the CVD mode on */
    ADCCON1bits.CVDEN = 1;

    /* Enable the ADC module */
    ADCCON3bits.DIGEN7 = 1; // Enable ADC7

    /* Trigger a level sensitive conversion */
    ADCCON3bits.GLSWTRG = 1;
    while(1);

    return (1);
}
```

22.5.6 双倍速快速 Turbo 通道

在某些应用中，用户可能需要通过 ADC 实现更高的数据吞吐率，高于可通过单个专用 ADC 模块实现的数据吞吐率。在这种情况下，可以使用 Turbo 模式，该模式会交替使用两个专用 ADC 模块。通过交替使用两个专用模块，可以让一个 ADC 模块在进行采样，而另一个 ADC 模块在进行转换。因此，吞吐率可以提高到双倍速率。

22.5.6.1 配置 TURBO 通道

Turbo 模式包含两个专用 ADC 模块，称为主模块和从模块，分别通过 TRBMST<2:0> 位（ADCCON1<29:27>）和 TRBSLV<2:0> 位 （ADCCON1<26:24>）进行选择。当主 ADC 模块被通过 TRGSRCx<4:0> 位（ADCTRGx）选择的触发源触发时，主 ADC 会完成采样时间并进行转换。同时，主 ADC 会在 （t_{SAMC} + t_{CONV}）的中点触发从模块，从模块会开始采样。以下的分离步骤对 Turbo 模式操作进行了说明：

1. 主 ADC 接收到来自外部源 （通过 TRGSRCx<4:0> 选择）的触发信号。
2. 主 ADC 进入采样模式。
3. 在采样完成后，开始转换。此外，主 ADC 会在 (t_{SAMC} + t_{CONV}) 中点触发从 ADC。因此，从 ADC 会进入采样模式。
4. 当主 ADC 在转换数据时，从 ADC 会完成采样并开始转换。
5. 主 ADC 结束转换，从 ADC 也在一定时间后结束转换。

总之，与单个通道的数据速率相比，在转换结束时，数据速率会高得多。

在转换结束时，输出数据会进入与每个专用 ADC 模块的模拟输入关联的两个独立的数据缓冲区（ADCDATAx）中，或者进入 FIFO 或系统 RAM。

> **注：** 虽然两个专用 ADC 模块被配置为主模块和从模块，但它们仍然维持其独立的模拟输入引脚，所以用户需要负责在电路板电平短接这两个输入 （如果处于差分模式，则两两短接），以便将同一输入信号同时传送给主模块和从模块来进行采样和转换。

22.5.6.2 TURBO 通道错误

只有满足几个配置条件时， Turbo 通道才会正常工作。如果 Turbo 通道未正确配置， Turbo 通道错误位 TRBERR （ADCCON1<30>）会由硬件置 1。当 TRBERR 位 （ADCCON1<30>）置 1 时，硬件会禁止 Turbo 通道的功能，即使用户代码已将 TRBEN 位 （ADCCON1<31>）置 1。Turbo 通道错误可以确保用户为 Turbo 通道选择正确的设置。只有满足以下条件时， Turbo 通道才会正确工作；否则， TRBERR 位会由硬件置 1：

- ADC 模块时钟分频比：主 ADC 模块和从 ADC 模块的 ADCDIV<6:0> 位（ADCxTIME<22:16>）应相同
- ADC 分辨率：主 ADC 模块和从 ADC 模块的 SELRES<1:0> 位 （ADCxTIME<25:24>）应相同。此外，在 Turbo 模式下不支持 6 位分辨率 （SELRES<1:0> = 00）。因此，SELRES<1:0> 位的值应不等于 00。
- 采样时间：主 ADC 模块和从 ADC 模块的 SAMC<9:0> 位 （ADCxTIME<9:0>）应相同
- 主 ADC 模块和从 ADC 模块的 STRGEN 位 （ADCTRGMODE）都应置 1
- 主 ADC 模块和从 ADC 模块的 SSAMPEN 位 （ADCTRGMODE）都应置 1

 © 2015 Microchip Technology Inc.

22.5.6.3 TURBO 通道的有效性

只有采样时间 （*t*SAMC）小于转换时间 （*t*CONV）时， Turbo 通道才会有效。这意味着一个 ADC 模块将在另一个 ADC 模块完成转换之前完成采样。否则，两个 ADC 模块最终会同时在进行采样，这违背了 Turbo 通道的目的。对于单个 SAR ADC，ADC 转换时间是不产生任何输出数据的时间。使用 Turbo 通道时，转换时间会与其他通道的采样时间交错。

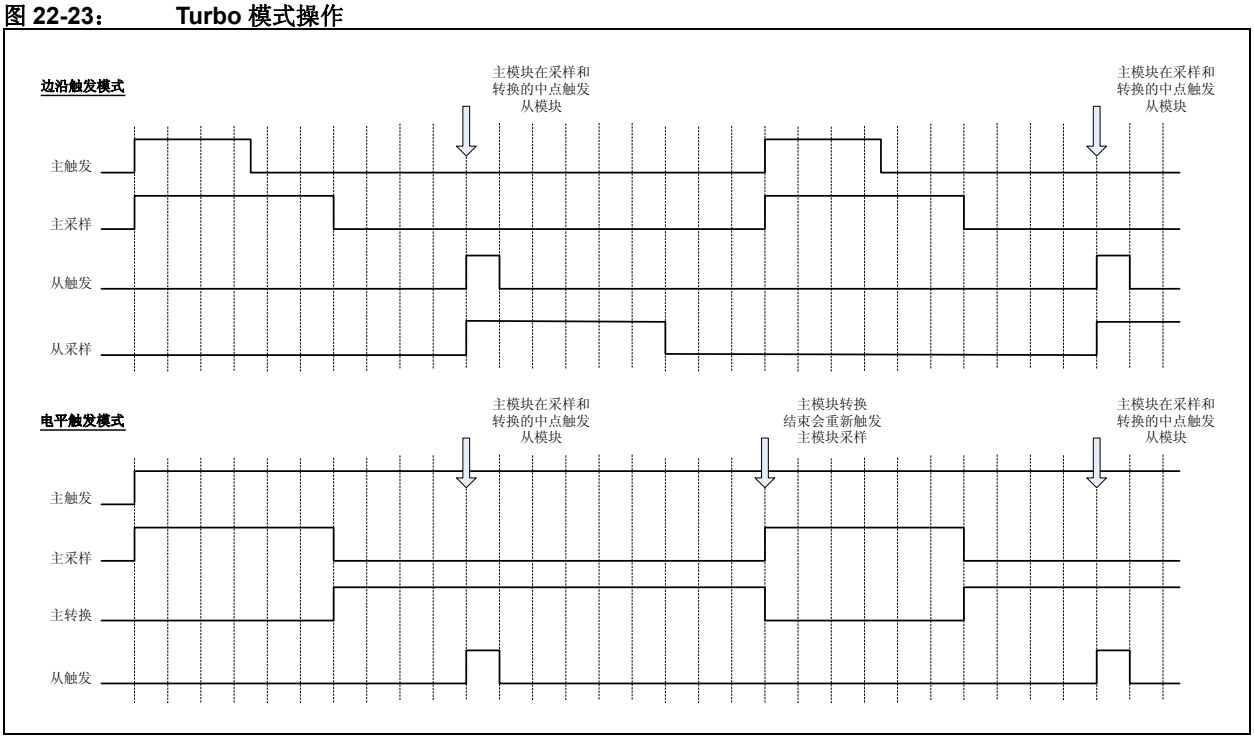
ADC 的采样时间取决于模拟源的输入电阻。为了减少采样时间，应减小模拟源的输入电阻。

如果为用户提供了 （*t*SAMC）大于转换时间 （*t*CONV）的应用 （由于设计约束或产品化的硬件等原因，无法进一步减小源阻抗），可以通过增大 ADCDIV<6:0> 位（ADCxTIME<22:16>）使转换时间 （计数 * TAD）等于采样时间 （不是计数）来降低时钟频率。请注意，由于时钟频率下降，应将采样时间 SAMC<9:0> 位 （ADCxTIME<9:0>）的计数设置得较低。

如果采样时间相对于所支持的最快转换时间 （14* TAD）来说非常高，则 Turbo 通道将无法在转换带宽方面实现显著提升。发生这种情况时，更好的选择是在单通道模式下以硬件支持的最快转换速率运行。

图 22-23 说明了在主模块触发信号处于边沿和电平敏感模式下时， Turbo 通道模式的操作。

图 22-23： Turbo 模式操作



处于中断模式时，用户应用程序应使能与主 ADC 模块和从 ADC 模块关联的模拟输入的 AGIENx 位。

22.6 中断

每个 ADC 模块都支持通过一系列中断源触发的中断，这些中断可以单独处理，也可以全局处理。此外还提供了提前中断功能，用于补偿中断响应延时。

产生允许的中断之后， CPU 将跳转到为该中断分配的向量处。然后， CPU 将在向量地址处开始执行代码。位于该向量地址处的用户软件应执行所需的操作，例如处理数据结果、清除中断标志，然后退出。关于中断和向量地址表详细信息的更多信息，请参见 《PIC32 系列参考手册》的**第 8 章 “中断”**（DS60001108），以及具体器件数据手册中的 **“中断控制器”**章节。

22.6.1 中断源

每个 ADC 都能够通过表 22-8 所列的事件产生中断。

表 22-8： ADC 中断源

中断事件	说明	中断允许位	中断状态位
ANx 数据就绪事件	在模拟输入源 （ANx）的转换完成时产生中断。在每个 ARDYx 位置 1 时，它们能够使用 ADCBASE 寄存器产生唯一中断。	ADCGIRQEN1 或 ADCGIRQEN2 寄存器的 AGIENx	ADCDSTATx 寄存器的 ARDYx
数字比较器事件	当某个已配置并使能的数字比较器满足转换的比较条件时。每个数字比较器都可以在其 DCMPED 位置 1 时产生唯一中断	ADCCMPCONx 寄存器的 DCMPGIEN	ADCCMPCONx 寄存器的 DCMPED
过采样滤波器数据就绪事件	当过采样滤波器完成累加 / 抽取处理并已存储结果时。	ADCFLTRx 寄存器的 AFGIEN	ADCFLTRx 寄存器的 AFRDY
带隙电压和 ADC 参考电压均就绪事件	在带隙电压和 ADC 参考电压均就绪时产生中断。	ADCCON2 寄存器的 BGVRIEN	ADCCON2 寄存器的 BGVRRDY
带隙故障 / 参考电压故障 /AVDD 欠压故障事件	在发生带隙故障 / 参考电压故障 /AVDD 欠压时产生中断。	ADCCON2 寄存器的 REFFLTIEN	ADCCON2 寄存器的 REFFLT
扫描结束事件	在所有选定输入完成扫描时产生中断	ADCCON2 寄存器的 EOSIEN	ADCCON2 寄存器的 EOSRDY
FIFO 数据就绪事件	在 FIFO 中数据就绪可供读取时产生中断。	ADCFSTAT 寄存器的 FIEN	ADCFSTAT 寄存器的 FRDY
DMA 缓冲区 B 满事件	在 DMA 缓冲区 B 满时产生中断。	ADCDMASTAT 寄存器的 RBFIEN6:RBFIEN0	ADCDMASTAT 寄存器的 RBF6:RBF0
DMA 缓冲区 A 满事件	在 DMA 缓冲区 A 满时产生中断。	ADCDMASTAT 寄存器的 RAFIEN6:RAFIEN0	ADCDMASTAT 寄存器的 RAF6:RAF0
ADC 模块唤醒事件	在 ADC 使能之后唤醒时产生中断。	ADCANCON 寄存器的 WKIEN6:WKIEN0 和 WKIEN7	ADCANCON 寄存器的 WKRDY6:WKRDY0 和 WKRDY7
更新就绪事件	在 ADC SFR 准备好 （并且可以安全地）使用新值更新时产生中断。	ADCCON3 寄存器的 UPDIEN	ADCCON3 寄存器的 UPDRDY
提前中断事件	（在转换结束之前）提前一定的 ADC 时钟量产生中断，提前时钟量由 ADCEIS<2:0> 位（ADCxTIME<28:26>）和 ADCEIS<2:0> 位（ADCCON2<10:8>）设置。	ADCEIENx 寄存器的 EIEN6:EIEN0	ADCEISTATx 寄存器的 EIRDY6:EIRDY0

 © 2015 Microchip Technology Inc.

22.6.2 ADC 基址寄存器 （ADCBASE）用法

在 ADC 的转换完成之后，如果中断通过向量跳转到一个所有模拟输入共用的函数，则需要通过求取 ADCDSTATx 中 ARDYx 位的值来确定 ADC 输入，这会花费一段较长的时间。为了避免花费该时间，提供了 ADCBASE 寄存器，其中将包含用户的 ADC ISR 跳转表的基址。读 ADCBASE 寄存器时，它会提供 ADCBASE 寄存器内容加上 ADCDSTATx 寄存器中设置的 ARDYx 位的编码之和。 ADCBASE 寄存器的这种用法支持生成中断向量地址，可以用来改善 ISR 的性能。

ARDYx 位采用二进制优先级编码，ARDY0 的优先级最高，而 ARDY63 的优先级最低。然后，对编码的优先级结果进行左移 （移位量等于在 ADCCON1 寄存器的 IRQVS<2:0> 位中指定的位数），之后与 ADCBASE 寄存器的内容相加。如果没有任何 ARDYx 位置 1，则 ADCBASE 寄存器的读取值将等于写入 ADCBASE 寄存器的值。

ADCBASE 寄存器中通常会被装入跳转表的基址，跳转表中则包含相应 ISR 的地址。第 k 个中断请求通过 ADCGIRQENx SFR （x = 1 或 2）之一中的 AGIENx 位 （0 ≤ x ≤ 63）使能。

例 22-8 说明了 ADCBASE 寄存器的用法。此外，例 22-9 中的代码说明了如何使用 ADCBASE 寄存器来获取不同 ADC 输入的中断向量。

例 22-8: ADCBASE 寄存器用法

情形 1：

```
ADCBASE = 0x1234;          // Set the address
ADCCON1bits.IRQVS = 2;     // left shift by 2
ADCGIRQEN1bits.AGIEN0 = 1; // enable interrupt when AN0 completion is done.
```

在 AN0 的 ADC 转换完成时， ADCDSTAT1 的 bit 0 = ARDY0 会置 1。

ADCBASE 的读取值 = 0x1234 + (0 << 2) = 0x1234。

因此，对于 AN0，应将 ISR 放置在地址 0x1234 处。

情形 2：

```
ADCBASE = 0x1234;          // Set the address
ADCCON1bits.IRQVS = 2;     // left shift by 2
ADCGIRQEN1bits.AGIEN0 = 2; // enable interrupt when AN2 completion is done.
```

在 AN2 的 ADC 转换完成时， ADCDSTAT1 的 bit 2 = ARDY2 会置 1。

ADCBASE 的读取值 = 0x1234 + (2 << 2) = 0x123C。

因此，对于 AN2，应将 ISR 放置在地址 0x123C 处。

注： ADCBASE 寄存器的内容不会改变。在读取 ADCBASE 寄存器时会执行求和，求和结果就是从 ADCBASE SFR 返回的读取值。

例 22-9: 获取不同 ADC 输入的中断向量

```
/* Number of ADC modules doing conversion */
#defineADC_MODULES3

/* Declare functions for each ADC handler */
void ADC0Handler(void);
void ADC1Handler(void);
void ADC2Handler(void);

void(*jumpTable[ADC_MODULES * 2])(void);

int ADC0Result;
int ADC1Result;
int ADC2Result;

int main(int argc, char** argv) {

    jumpTable[0] = &ADC0Handler;  // Set up jump table
    jumpTable[2] = &ADC1Handler;
    jumpTable[4] = &ADC2Handler;

    /* Configure ADCCON1 */
    ADCCON1 = 0;           // No ADCCON1 features are enabled including: Stop-in-Idle, turbo,
                           // CVD mode, Fractional mode and scan trigger source.

    /* Configure ADCCON2 */
    ADCCON2 = 0;           // Since, we are using only the Class 1 inputs, no setting is
                           // required for ADCDIV

    /* Initialize warm up time register */
    ADCANCON = 0;
    ADCANCONbits.WKUPCLKCNT = 5;  // Wakeup exponent = 32 * TADx

    /* Clock setting */
    ADCCON3 = 0;
    ADCCON3bits.ADCSEL = 0;       // Select input clock source
    ADCCON3bits.CONCLKDIV = 1;    // Control clock frequency is half of input clock
    ADCCON3bits.VREFSEL = 0;      // Select AVdd and AVss as reference source

    /* Select ADC sample time and conversion clock */
    ADC0TIMEbits.ADCDIV = 1;      // ADC0 clock frequency is half of control clock = TAD0
    ADC0TIMEbits.SAMC = 5;        // ADC0 sampling time = 5 * TAD0
    ADC0TIMEbits.SELRES = 3;      // ADC0 resolution is 12 bits
    ADC1TIMEbits.ADCDIV = 1;      // ADC1 clock frequency is half of control clock = TAD1
    ADC1TIMEbits.SAMC = 5;        // ADC1 sampling time = 5 * TAD1
    ADC1TIMEbits.SELRES = 3;      // ADC1 resolution is 12 bits
    ADC2TIMEbits.ADCDIV = 1;      // ADC2 clock frequency is half of control clock = TAD2
    ADC2TIMEbits.SAMC = 5;        // ADC2 sampling time = 5 * TAD2
    ADC2TIMEbits.SELRES = 3;      // ADC2 resolution is 12 bits

    /* Select analog input for ADC modules, no presync trigger, not sync sampling */
    ADCTRGMODEbits.SH0ALT = 0;    // ADC0 = AN0
    ADCTRGMODEbits.SH1ALT = 0;    // ADC1 = AN1
    ADCTRGMODEbits.SH2ALT = 0;    // ADC2 = AN2

    /* Select ADC input mode */
    ADCIMCON1bits.SIGN0 = 0;      // unsigned data format
    ADCIMCON1bits.DIFF0 = 0;      // Single ended mode
    ADCIMCON1bits.SIGN1 = 0;      // unsigned data format
    ADCIMCON1bits.DIFF1 = 0;      // Single ended mode
    ADCIMCON1bits.SIGN2 = 0;      // unsigned data format
    ADCIMCON1bits.DIFF2 = 0;      // Single ended mode

    /* Configure ADCGIRQENx */
    ADCGIRQEN1 = 0;
    ADCGIRQEN2 = 0;
    ADCGIRQEN1bits.AGIEN0 = 1;    // Enable data ready interrupt for AN0
    ADCGIRQEN1bits.AGIEN1 = 1;    // Enable data ready interrupt for AN1
    ADCGIRQEN1bits.AGIEN2 = 1;    // Enable data ready interrupt for AN2

    /* Configure ADBASE */
    ADCBASE = (int)(&jumpTable[0]);   // Initialize ADCBASE with starting address of jump table
    ADCCON1bits.IRQVS = 0;            // No left shift of address

    /* Configure ADCCSSx */
    ADCCSS1 = 0;                  // No scanning is used
    ADCCSS2 = 0;
```

 © 2015 Microchip Technology Inc.

例 22-9： 获取不同 ADC 输入的中断向量 （续）

```
    /* Configure ADCCMPCONx */
    ADCCMPCON1 = 0;           // No digital comparators are used. Setting the ADCCMPCONx
    ADCCMPCON2 = 0;           // register to '0' ensures that the comparator is disabled.
    ADCCMPCON3 = 0;           // Other registers are “n't care”
    ADCCMPCON4 = 0;
    ADCCMPCON5 = 0;
    ADCCMPCON6 = 0;

    /* Configure ADCFLTRx */
    ADCFLTR1 = 0;             // No oversampling filters are used.
    ADCFLTR2 = 0;
    ADCFLTR3 = 0;
    ADCFLTR4 = 0;
    ADCFLTR5 = 0;
    ADCFLTR6 = 0;

    /* Set up the trigger sources */
    ADCTRGSNSbits.LVL0 = 0;   // Edge trigger
    ADCTRGSNSbits.LVL1 = 0;   // Edge trigger
    ADCTRGSNSbits.LVL2 = 0;   // Edge trigger
    ADCTRG1bits.TRGSRC0 = 1;  // Set AN0 to trigger from software.
    ADCTRG1bits.TRGSRC1 = 1;  // Set AN1 to trigger from software.
    ADCTRG1bits.TRGSRC2 = 1;  // Set AN2 to trigger from software.

    /* Early interrupt */
    ADCEIEN1 = 0;             // No early interrupt
    ADCEIEN2 = 0;
    ADCCON2bits.ADCEIOVR = 1; // Override early interrupt

    /* Turn the ADC on */
    ADCCON1bits.ON = 1;

    /* Wait for voltage reference to be stable */
    while(!ADCCON2bits.BGVRRDY);  // Wait until the reference voltage is ready
    while(ADCCON2bits.REFFLT);    // Wait if there is a fault with the reference voltage

    /* Enable clock to analog circuit */
    ADCANCONbits.ANEN0 = 1;       // Enable the clock to analog bias and digital control
    ADCANCONbits.ANEN1 = 1;       // Enable the clock to analog bias and digital control
    ADCANCONbits.ANEN2 = 1;       // Enable the clock to analog bias and digital control

    /* Wait for ADC to be ready */
    while(!ADCANCONbits.WKRDY0);  // Wait until ADC0 is ready
    while(!ADCANCONbits.WKRDY1);  // Wait until ADC1 is ready
    while(!ADCANCONbits.WKRDY2);  // Wait until ADC2 is ready

    /* Enable the ADC module */
    ADCCON3bits.DIGEN0 = 1;       // Enable ADC0
    ADCCON3bits.DIGEN1 = 1;       // Enable ADC1
    ADCCON3bits.DIGEN2 = 1;       // Enable ADC2

    /* Trigger a conversion */
    ADCCON3bits.GSWTRG = 1;

    while (1);

    return (1);
}

/* Handler for the ADC interrupt */
void __ISR(_ADC_VECTOR, ipl3) ADCHandler1(void)
{
    /* call the corresponding ADC module handler */
    ((void(*)())*((int *)ADCBASE))();
}

void ADC0Handler(void)
{
    /* Verify if data for AN0 is ready. This bit is self cleared upon data read */
    if(ADCDSTAT1bits.ARDY0)
    {
        ADC0Result = ADCDATA0;
    }
}
```

例 22-9：　获取不同 ADC 输入的中断向量（续）

```
void ADC1Handler(void)
{
    /* Verify if data for AN1 is ready. This bit is self cleared upon data read */
    if(ADCDSTAT1bits.ARDY1)
    {
        ADC1Result = ADCDATA1;
    }
}

void ADC2Handler(void)
{
    /* Verify if data for AN2 is ready. This bit is self cleared upon data read */
    if(ADCDSTAT2bits.ARDY2)
    {
        ADC2Result = ADCDATA2;
    }
}
```

22.6.3　中断允许、优先级和向量获取

前面提到每个 ADC 事件都会在其关联的中断允许位 IE 置 1 时产生中断。此外，每个事件还具有关联的中断标志位 IF、优先级位 IP<2:0> 和子优先级位 IS<1:0>。关于如何允许中断和设置中断优先级的更多信息，请参见**第 8 章 “中断”**（DS60001108）。前面列出的每个 ADC 事件也具有关联的中断向量。关于与每个独立中断关联的向量位置和控制 / 状态位的更多信息，请参见具体器件数据手册中的 **“中断控制器”** 章节。

22.6.4　独立和全局中断

使用前面列出的独立中断可以使每个 ISR 专注于高效地处理特定事件，从而显著优化多个 ADC 事件的处理。此外，还可以根据所执行的任务简便地隔离不同 ISR，从而使用户软件可以更容易实现和维护。在一些情况下，可能会希望让单个 ISR 处理多个中断事件。为了方便这一点，可以对每个 ADC 事件进行逻辑 “或” 运算，产生单一的全局 ADC 中断。当对于某个 ADC 事件允许全局中断时，它会向量跳转到一个单一中断程序。这个单一的全局 ISR 需要负责通过查询来确定中断源并进行相应的处理。

使用全局中断需要配置它自己的唯一 IE、IF、IP 和 IS 位，以及配置其中断向量，如**第 22.6.3 节 “中断允许、优先级和向量获取”** 所述。

ADC 的中断可以配置为独立或全局，或同时采用两种形式，即一些中断单独处理，而其他一些则在全局 ISR 中处理。

© 2015 Microchip Technology Inc.

22.6.5 提前中断

提前中断功能使 ADC 模块可以在转换完成之前产生中断。即使输入仍然处于转换过程中，处理器应用程序软件也可以通过 “抢先起步”来开始执行进入 ISR。提前中断可以使 ADC 转换完成过程与和中断相关联的处理器开销交叠，从而提高系统的吞吐率。ADCEIS<2:0> 位（ADCxTIME<28:26>）和 ADCEIS<2:0> 位 （ADCCON2<10:8>）用于设置中断应提前的 ADC 时钟数 （分别用于专用和共用输入）。适当地配置这些位可以减小从对模拟信号进行采样的时刻到用户应用软件可以使用数据的时间点之间的延时。

在转换结束之前（即数据实际在 ADCDATAx 寄存器中可用之前）达到所设置的 ADC 时钟数时，ADCEISTAT1 或 ADCEISTAT2 寄存器中相应的提前中断就绪位 EIRDYx 会置 1。如果 ADCEIEN1 或 ADCEIEN2 寄存器中相应的中断允许位 EIENx 置 1，则会产生中断。

通过将提前中断改写位 ADCEIOVR （ADCCON2<12>）置 1，可以改写提前中断功能。当该位置 1 时，将 ADCEIEN1 或 ADCEIEN2 寄存器中的位置 1 对中断产生没有任何影响。之后，中断产生由 ADCGIRQEN1 和 ADCGIRQEN2 寄存器的设置控制。

注：	ADCEIS<2:0> 位设置不适用于过采样滤波器数据就绪信号 AFRDY。

22.7 节能模式期间的操作

节能模式 （休眠和空闲）可以最大程度减少 CPU、总线和其他外设的数字活动，可用于降低转换噪声。

22.7.1 休眠模式

当器件进入休眠模式时，系统时钟 （SYCCLK）会被暂停。如果 ADC 模块选择 SYSCLK 作为其时钟源，或选择REFCLK3作为其时钟源（REFCLK3基于SYSCLK产生），ADC会进入休眠模式。

当 SYSCLK 为时钟源 （直接或间接），并且在转换期间出现休眠模式时，转换会被中止。在从休眠模式退出时，转换器不会继续进行部分完成的转换。器件进入或退出休眠模式不会影响 ADC 寄存器的内容。如果 ADC 时钟源是基于 SYSCLK 之外的其他会在休眠模式期间工作的时钟源产生的，则 ADC 模块可以在休眠模式下工作。 FRC 时钟源是在休眠期间正常工作的合理选择；但是，如果REFCLK3时钟源具有在休眠模式期间正常工作的输入时钟，则也可以使用REFCLK3时钟源。

休眠模式期间的ADC操作可以减小来自转换的数字开关噪声。当完成转换时，模拟输入的ARDYx状态位会置 1，结果会被装入相应的 ADC 结果寄存器 （ADCDATAx）。

如果允许任何 ADC 中断，则器件会在发生 ADC 中断时从休眠模式唤醒。如果 ADC 中断的优先级大于当前 CPU 的优先级，程序将在 ADC ISR 处恢复执行。否则，程序将从使器件进入休眠模式的 `WAIT` 指令后的指令继续执行。

为了将数字噪声对 ADC 模块操作的影响降至最低，用户必须选择可确保模数转换可在休眠模式下进行的转换触发源。例如，在器件处于休眠模式时，可以使用外部中断引脚（INT0）转换触发选项 （TRGSRC<4:0> = `00100`）来执行采样和转换。

> **注：** 要使ADC模块在休眠模式下工作，ADC时钟源必须设置为内部FRC（ADCSEL<1:0>位 （ADCCON2<31:30>） = `01`）。或者，也可以使用 REFCLK3 时钟源；但是，用于 REFCLK3 的时钟源必须在休眠期间工作。对 ADC 时钟配置的任何更改都要求禁止 ADC。

22.7.2 空闲模式期间的 ADC 操作

对于 ADC，空闲模式停止位 SIDL （ADCCON1<13>）用于指定 ADC 模块是在空闲时停止还是在空闲时继续工作。如果 SIDL = `0`，当器件进入空闲模式时， ADC 模块将继续正常工作。如果允许任何 ADC 中断，则器件会在发生 ADC 中断时从空闲模式唤醒。如果 ADC 中断的优先级大于当前 CPU 的优先级，程序将在 ADC ISR 处恢复执行。否则，程序将从使器件进入空闲模式的 `WAIT` 指令后的指令继续执行。

如果 SIDL = `1`，则在空闲模式下 ADC 模块将停止。如果器件在转换期间进入空闲模式，则转换会被中止。在从空闲模式退出时，转换器不会继续进行部分完成的转换。

 © 2015 Microchip Technology Inc.

22.7.3 ADC 低功耗模式

通过禁止不运行的各个 ADC 模块的数字电路，可以将 ADC 模块置为低功耗状态。这可以通过将 ADCCON3 寄存器中的 DIGENx 位和 DIGEN7 位 （见寄存器 22-3）清零来实现。

通过禁止不运行的各个 ADC 模块的模拟和偏置电路，可以实现功耗更低的状态。这可以通过将 ADCANCON 寄存器中的 ANENx 位和 ANEN7 位 （见寄存器 22-41）清零来实现。与禁止并重新使能 ADC 模块的模拟和偏置电路相比，通过禁止数字电路来实现低功耗模式时，模块重新启动的速度明显更快。这是因为对于 ADC 模块，使用 ANENx 位和 ANEN7 位禁止并重新使能模拟和偏置电路需要唤醒时间 （典型的最小唤醒时间为 20 μs），之后它才可以使用。关于稳定时间的更多信息，请参见具体器件数据手册中的 **“电气特性”** 章节。

使能 ADC 模块的模拟和偏置电路之后，应使用唤醒就绪位 WKRDY6:WKRDY0 和 WKRDY7（它们应等于 1）来查询是否已唤醒 （或产生中断）。

22.8 复位的影响

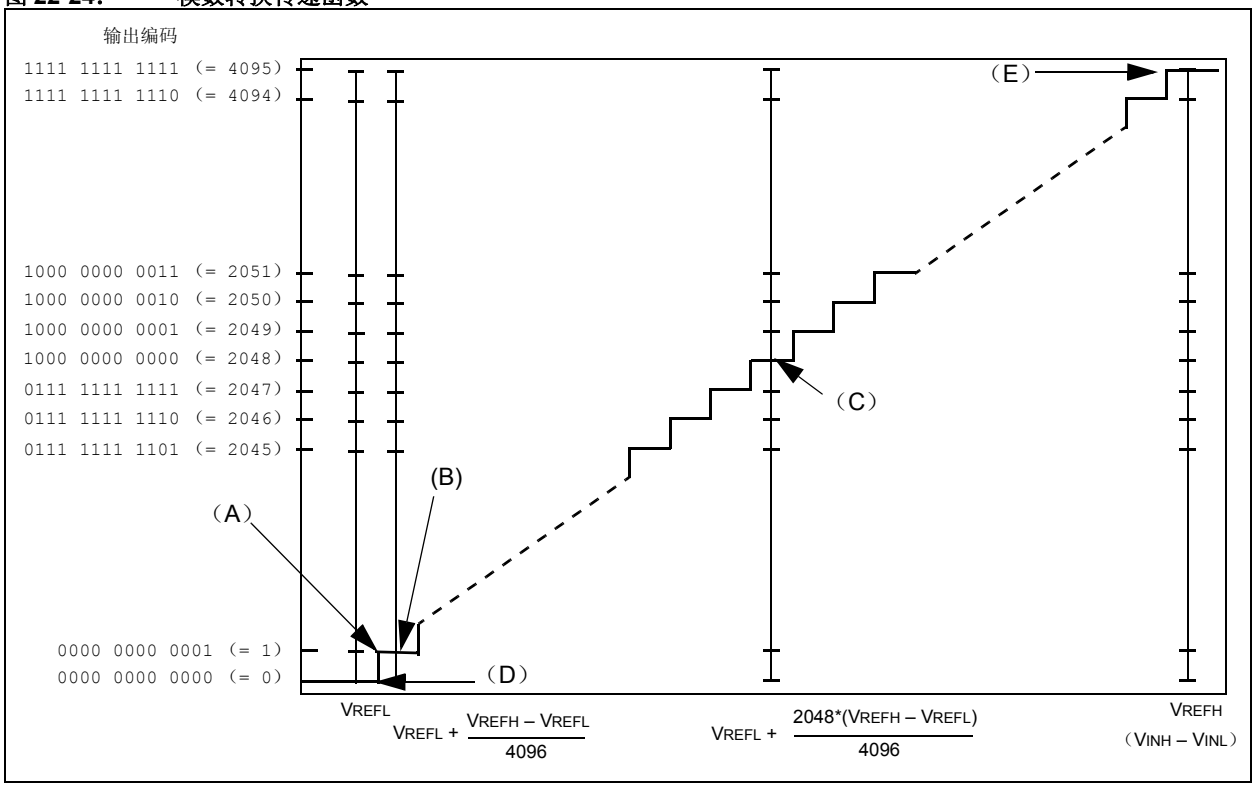
在发生任何复位事件之后，所有 ADC 控制和状态寄存器都会被复位为它们的默认值，控制位处于无效状态。这会禁止 ADC 模块，并将模拟输入引脚设置为模拟输入模式。任何正在进行的转换都会终止，结果不会被写入结果缓冲区。在器件复位期间，ADCDATAx 寄存器中的值会被初始化为 0x00000000。偏置电路也会被关闭，所以 ADC 需要通过查询 BGVRRDY 位 （ADCCON2 寄存器）来等待偏置电路达到稳定 （或请求通过该位产生中断），然后再恢复工作。

22.9 传递函数

图 22-24 给出了 12 位 ADC 的典型传递函数。输入电压差值 （VINH - VINL）与参考电压差值 （VREFH - VREFL）进行比较。

- 当输入电压为 （VREFH - VREFL/8192）或 0.5 LSb 时，发生第一个编码跳变 （A）
- 00 0000 0001 编码中点位于 （VREFH - VREFL/4096）或 1.0 LSb （B）
- 10 0000 0000 编码中点位于 （2048 * (VREFH - VREFL)/4096）（C）
- 小于 （1 * (VREFH - VREFL)/8192）的输入电压会被转换为 00 0000 0000 （D）
- 大于 （8192 * (VREFH - VREFL)/8192）的输入电压会被转换为 11 1111 1111 （E）

图 22-24： 模数转换传递函数



 © 2015 Microchip Technology Inc.

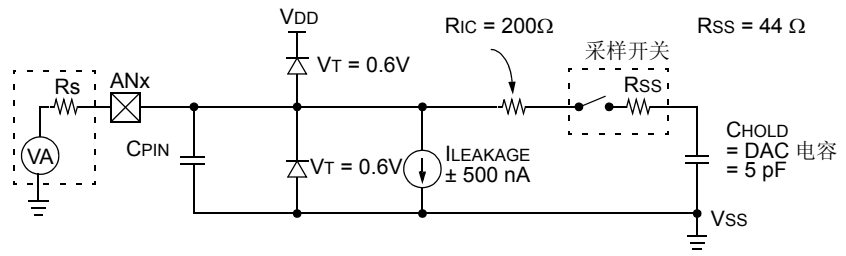
22.10 ADC 采样要求

图 22-25 给出了 12 位 ADC 的模拟输入模型。模数转换的总采集时间是内部电路稳定时间和保持电容充电时间的函数。

为了使 ADC 模块达到规定的精度，必须让充电保持电容 （C_{HOLD}）充分充电至模拟输入引脚上的电压。模拟输出信号源阻抗 （RS）、片内走线等效电阻 （RIC）和内部采样开关阻抗 （RSS）共同直接影响着电容 C_{HOLD} 充电所需的时间。因此，模拟源的合并阻抗必须足够小，以便在所选择的采样时间内对保持电容充分充电（达到所需电压的四分之一 LSb 范围内）。在每次采样操作前，内部保持电容将处于放电状态。

两次转换之间应留出至少 1 个 TAD 时间段作为采集时间。更多信息，请参见具体器件数据手册中的 **“电气特性”** 章节。

图 22-25： 12 位 ADC 模拟输入模型



注： CPIN 值取决于器件封装，未经测试。如果 RS ≤ 5 kΩ，CPIN 的影响可忽略。

图注：

CPIN = 输入电容

RSS = 采样开关电阻

RS = 源电阻

ILEAKAGE = 由各连接点在引脚上产生的泄漏电流

VT = 阈值电压

RIC = 片内走线等效电阻

CHOLD = 采样 / 保持电容

22.11 连接注意事项

由于模拟输入采用了静电放电（ESD）保护措施，它们具有连接到 VDD 和 VSS 的二极管；因此，模拟输入必须介于 VDD 和 VSS 之间。由于存在二极管，模拟引脚无法承受 5V 电压。如果输入电压超出此范围 0.3V 以上 （任一方向上），就会有一个二极管正向偏置，如果超出输入电流规范可能会损坏器件。

有时可以通过外接一个 RC 滤波器来对输入信号进行抗混叠滤波。选择的 R （电阻）元件应确保可以满足采集时间要求。任何通过高阻抗连接到模拟输入引脚上的外部元件（如电容和齐纳二极管等）在引脚上的泄漏电流都应极小。

22.12 相关应用笔记

本节列出了与手册本章内容相关的应用笔记。这些应用笔记可能并不是专为 PIC32 器件系列而编写的，但其概念是相近的，通过适当修改并受到一定限制即可使用。当前与 12 位高速逐次逼近寄存器 （SAR）模数转换器 （ADC）模块相关的应用笔记有：

标题	应用笔记编号
目前没有相关的应用笔记。	N/A

注： 如需获取更多 PIC32 系列器件的应用笔记和代码示例，请访问 Microchip 网站（www.microchip.com）。

 © 2015 Microchip Technology Inc.

22.13 版本历史

版本 A （2015 年 6 月）

这是本文档的初始版本。

注：

 © 2015 Microchip Technology Inc.

请注意以下有关 Microchip 器件代码保护功能的要点：

- Microchip 的产品均达到 Microchip 数据手册中所述的技术指标。
- Microchip 确信：在正常使用的情况下， Microchip 系列产品是当今市场上同类产品中最安全的产品之一。
- 目前，仍存在着恶意、甚至是非法破坏代码保护功能的行为。就我们所知，所有这些行为都不是以 Microchip 数据手册中规定的操作规范来使用 Microchip 产品的。这样做的人极可能侵犯了知识产权。
- Microchip 愿与那些注重代码完整性的客户合作。
- Microchip 或任何其他半导体厂商均无法保证其代码的安全性。代码保护并不意味着我们保证产品是 “牢不可破”的。

代码保护功能处于持续发展中。Microchip 承诺将不断改进产品的代码保护功能。任何试图破坏 Microchip 代码保护功能的行为均可视为违反了《数字器件千年版权法案 （Digital Millennium Copyright Act）》。如果这种行为导致他人在未经授权的情况下，能访问您的软件或其他受版权保护的成果，您有权依据该法案提起诉讼，从而制止这种行为。

提供本文档的中文版本仅为了便于理解。请勿忽视文档中包含的英文部分，因为其中提供了有关 Microchip 产品性能和使用情况的有用信息。Microchip Technology Inc. 及其分公司和相关公司、各级主管与员工及事务代理机构对译文中可能存在的任何差错不承担任何责任。建议参考 Microchip Technology Inc. 的英文原版文档。

本出版物中所述的器件应用信息及其他类似内容仅为您提供便利，它们可能由更新之信息所替代。确保应用符合技术规范，是您自身应负的责任。**Microchip 对这些信息不作任何明示或暗示、书面或口头、法定或其他形式的声明或担保，包括但不限于针对其使用情况、质量、性能、适销性或特定用途的适用性的声明或担保。** Microchip 对因这些信息及使用这些信息而引起的后果不承担任何责任。如果将 Microchip 器件用于生命维持和 / 或生命安全应用，一切风险由买方自负。买方同意在由此引发任何一切伤害、索赔、诉讼或费用时，会维护和保障 Microchip 免于承担法律责任，并加以赔偿。除非另外声明，在 Microchip 知识产权保护下，不得暗中或以其他方式转让任何许可证。

商标

Microchip 的名称和徽标组合、 Microchip 徽标、 dsPIC、 FlashFlex、 flexPWR、 JukeBlox、 KEELOQ、 KEELOQ 徽标、 Kleer、 LANCheck、 MediaLB、 MOST、 MOST 徽标、 MPLAB、 OptoLyzer、 PIC、 PICSTART、 PIC32 徽标、 RightTouch、 SpyNIC、 SST、 SST 徽标、 SuperFlash 及 UNI/O 均为 Microchip Technology Inc. 在美国和其他国家或地区的注册商标。

The Embedded Control Solutions Company 和 mTouch 为 Microchip Technology Inc. 在美国的注册商标。

Analog-for-the-Digital Age、 BodyCom、 chipKIT、 chipKIT 徽标、 CodeGuard、 dsPICDEM、 dsPICDEM.net、 ECAN、 In-Circuit Serial Programming、 ICSP、 Inter-Chip Connectivity、 KleerNet、 KleerNet 徽标、 MiWi、 motorBench、 MPASM、 MPF、 MPLAB Certified 徽标、 MPLIB、 MPLINK、 MultiTRAK、 NetDetach、 Omniscient Code Generation、PICDEM、PICDEM.net、PICkit、PICtail、 RightTouch 徽标、 REAL ICE、 SQI、 Serial Quad I/O、 Total Endurance、 TSHARC、 USBCheck、 VariSense、 ViewSpan、 WiperLock、 Wireless DNA 和 ZENA 均为 Microchip Technology Inc. 在美国和其他国家或地区的商标。

SQTP 为 Microchip Technology Inc. 在美国的服务标记。

Silicon Storage Technology 为 Microchip Technology Inc. 在除美国外的国家或地区的注册商标。

GestIC 为 Microchip Technology Inc. 的子公司 Microchip Technology Germany II GmbH & Co. & KG 在除美国外的国家或地区的注册商标。

在此提及的所有其他商标均为各持有公司所有。

© 2015, Microchip Technology Inc. 版权所有。

ISBN：978-1-63277-889-5

QUALITY MANAGEMENT SYSTEM CERTIFIED BY DNV

═ ISO/TS 16949 ═

Microchip 位于美国亚利桑那州 Chandler 和 Tempe 与位于俄勒冈州 Gresham 的全球总部、设计和晶圆生产厂及位于美国加利福尼亚州和印度的设计中心均通过了 ISO/TS-16949:2009 认证。 Microchip 的 PIC® MCU 与 dsPIC® DSC、KEELOQ® 跳码器件、串行 EEPROM、单片机外设、非易失性存储器和模拟产品严格遵守公司的质量体系流程。此外， Microchip 在开发系统的设计和生产方面的质量体系也已通过了 ISO 9001:2000 认证。

全球销售及服务网点

美洲

公司总部 Corporate Office
2355 West Chandler Blvd.
Chandler, AZ 85224-6199
Tel: 1-480-792-7200
Fax: 1-480-792-7277
技术支持：
http://www.microchip.com/support
网址：www.microchip.com

亚特兰大 Atlanta
Duluth, GA
Tel: 1-678-957-9614
Fax: 1-678-957-1455

奥斯汀 Austin, TX
Tel: 1-512-257-3370

波士顿 Boston
Westborough, MA
Tel: 1-774-760-0087
Fax: 1-774-760-0088

芝加哥 Chicago
Itasca, IL
Tel: 1-630-285-0071
Fax: 1-630-285-0075

克里夫兰 Cleveland
Independence, OH
Tel: 1-216-447-0464
Fax: 1-216-447-0643

达拉斯 Dallas
Addison, TX
Tel: 1-972-818-7423
Fax: 1-972-818-2924

底特律 Detroit
Novi, MI
Tel: 1-248-848-4000

休斯敦 Houston, TX
Tel: 1-281-894-5983

印第安纳波利斯 Indianapolis
Noblesville, IN
Tel: 1-317-773-8323
Fax: 1-317-773-5453

洛杉矶 Los Angeles
Mission Viejo, CA
Tel: 1-949-462-9523
Fax: 1-949-462-9608

纽约 New York, NY
Tel: 1-631-435-6000

圣何塞 San Jose, CA
Tel: 1-408-735-9110

加拿大多伦多 Toronto
Tel: 1-905-673-0699
Fax: 1-905-673-6509

亚太地区

亚太总部 Asia Pacific Office
Suites 3707-14, 37th Floor
Tower 6, The Gateway
Harbour City, Kowloon
Hong Kong
Tel: 852-2943-5100
Fax: 852-2401-3431

中国 - 北京
Tel: 86-10-8569-7000
Fax: 86-10-8528-2104

中国 - 成都
Tel: 86-28-8665-5511
Fax: 86-28-8665-7889

中国 - 重庆
Tel: 86-23-8980-9588
Fax: 86-23-8980-9500

中国 - 东莞
Tel: 86-769-8702-9880

中国 - 杭州
Tel: 86-571-8792-8115
Fax: 86-571-8792-8116

中国 - 香港特别行政区
Tel: 852-2943-5100
Fax: 852-2401-3431

中国 - 南京
Tel: 86-25-8473-2460
Fax: 86-25-8473-2470

中国 - 青岛
Tel: 86-532-8502-7355
Fax: 86-532-8502-7205

中国 - 上海
Tel: 86-21-5407-5533
Fax: 86-21-5407-5066

中国 - 沈阳
Tel: 86-24-2334-2829
Fax: 86-24-2334-2393

中国 - 深圳
Tel: 86-755-8864-2200
Fax: 86-755-8203-1760

中国 - 武汉
Tel: 86-27-5980-5300
Fax: 86-27-5980-5118

中国 - 西安
Tel: 86-29-8833-7252
Fax: 86-29-8833-7256

中国 - 厦门
Tel: 86-592-238-8138
Fax: 86-592-238-8130

中国 - 珠海
Tel: 86-756-321-0040
Fax: 86-756-321-0049

亚太地区

台湾地区 - 高雄
Tel: 886-7-213-7828

台湾地区 - 台北
Tel: 886-2-2508-8600
Fax: 886-2-2508-0102

台湾地区 - 新竹
Tel: 886-3-5778-366
Fax: 886-3-5770-955

澳大利亚 Australia - Sydney
Tel: 61-2-9868-6733
Fax: 61-2-9868-6755

印度 India - Bangalore
Tel: 91-80-3090-4444
Fax: 91-80-3090-4123

印度 India - New Delhi
Tel: 91-11-4160-8631
Fax: 91-11-4160-8632

印度 India - Pune
Tel: 91-20-3019-1500

日本 Japan - Osaka
Tel: 81-6-6152-7160
Fax: 81-6-6152-9310

日本 Japan - Tokyo
Tel: 81-3-6880-3770
Fax: 81-3-6880-3771

韩国 Korea - Daegu
Tel: 82-53-744-4301
Fax: 82-53-744-4302

韩国 Korea - Seoul
Tel: 82-2-554-7200
Fax: 82-2-558-5932 或
82-2-558-5934

马来西亚 Malaysia - Kuala Lumpur
Tel: 60-3-6201-9857
Fax: 60-3-6201-9859

马来西亚 Malaysia - Penang
Tel: 60-4-227-8870
Fax: 60-4-227-4068

菲律宾 Philippines - Manila
Tel: 63-2-634-9065
Fax: 63-2-634-9069

新加坡 Singapore
Tel: 65-6334-8870
Fax: 65-6334-8850

泰国 Thailand - Bangkok
Tel: 66-2-694-1351
Fax: 66-2-694-1350

欧洲

奥地利 Austria - Wels
Tel: 43-7242-2244-39
Fax: 43-7242-2244-393

丹麦 Denmark-Copenhagen
Tel: 45-4450-2828
Fax: 45-4485-2829

法国 France - Paris
Tel: 33-1-69-53-63-20
Fax: 33-1-69-30-90-79

德国 Germany - Dusseldorf
Tel: 49-2129-3766400

德国 Germany - Karlsruhe
Tel: 49-721-625370

德国 Germany - Munich
Tel: 49-89-627-144-0
Fax: 49-89-627-144-44

意大利 Italy - Milan
Tel: 39-0331-742611
Fax: 39-0331-466781

意大利 Italy - Venice
Tel: 39-049-7625286

荷兰 Netherlands - Drunen
Tel: 31-416-690399
Fax: 31-416-690340

波兰 Poland - Warsaw
Tel: 48-22-3325737

西班牙 Spain - Madrid
Tel: 34-91-708-08-90
Fax: 34-91-708-08-91

瑞典 Sweden - Stockholm
Tel: 46-8-5090-4654

英国 UK - Wokingham
Tel: 44-118-921-5800
Fax: 44-118-921-5820

07/14/15

© 2015 Microchip Technology Inc.