ADuCM360/ADuCM361硬件用户指南
UG-367

One Technology Way • P.O. Box 9106 • Norwood, MA 02062-9106, U.S.A. • Tel: 781.329.4700 • Fax: 781.461.3113 • www.analog.com

使用集成双通道Σ-Δ型ADC和ARM Cortex-M3的低功耗精密模拟微控制器
ADuCM360/ADuCM361

范围

本用户指南详细说明ADuCM360/ADuCM361的功能和特性。

免责声明

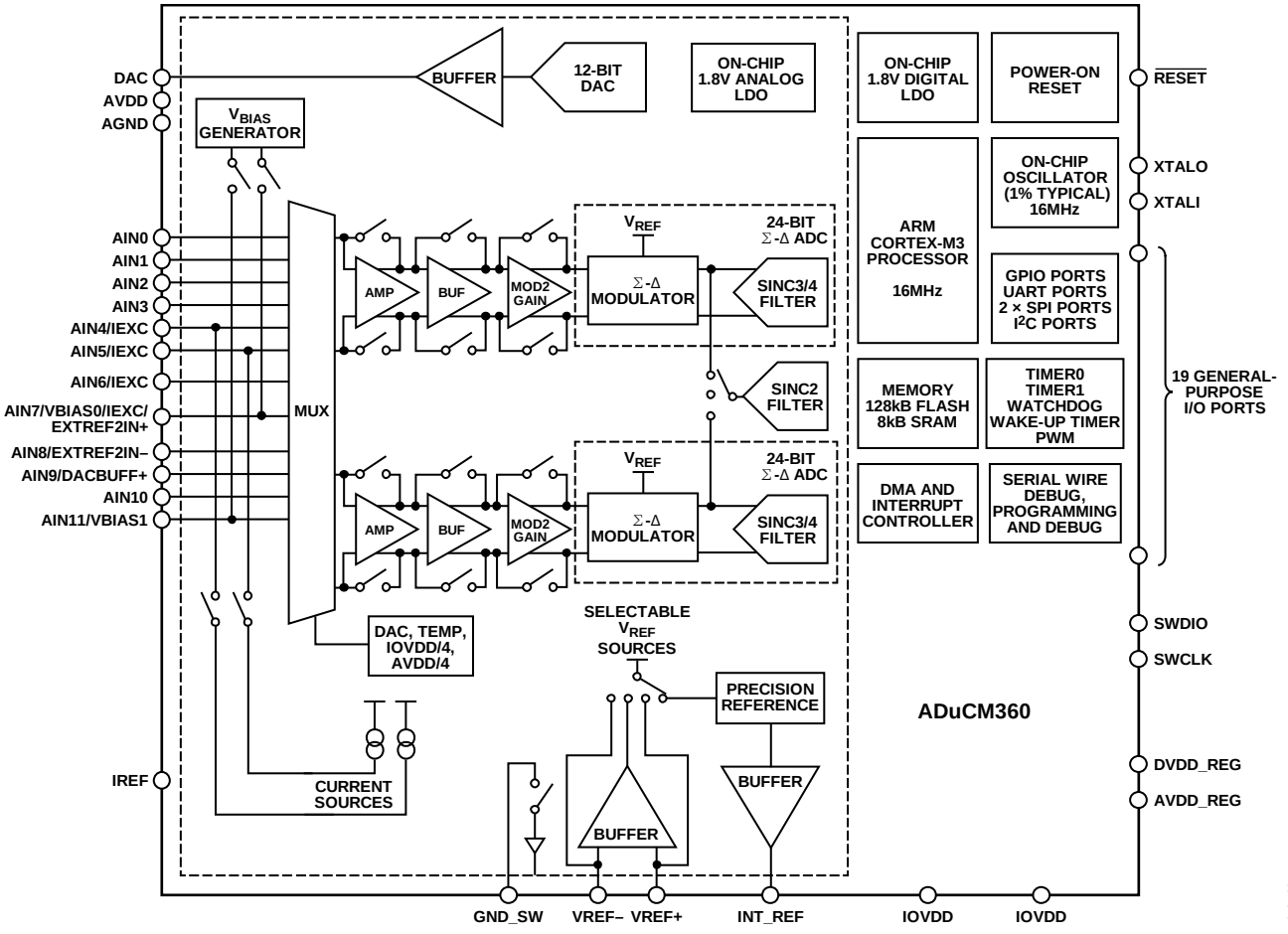


图1. ADuCM360框图

ADuCM360/ADuCM361硬件用户指南

目录

ADuCM360/ADuCM361硬件用户指南

修订历史

2014年9月—修订版C至修订版D

移动“存储器”部分 11
更改“ADC电路概述”部分 20
更改“数字滤波器选项”部分 28
更改表16 29
更改表30 41
更改表48 49
更改图18 52
更改表65 66
更改表66 68
更改表67 69
更改“闪存保护：用户读保护”部分 88
更改“I^2C特性”部分 107
更改表131 117
更改表140 120
更改表141 121
更改表208中引脚36和引脚36的描述 172

2014年5月 — 修订版B至修订版C

更改第61页 61

2013年7月—修订版A至修订版B

更改图1 1
移动“修订历史”部分 4
增加图2，重新排序；更改“单通道12位电压输出DAC”部分 7
更改“存储器”部分 8
更改图3 10
更改表4 11
更改“时钟架构控制寄存器0”部分和表5 12
更改“电源管理单元工作原理”部分 13
更改“ADC电路概述”部分和图4 20
更改“模拟输入通道配置”部分 21
删除“步进检测电路—工作模式1，DETCON[3] = 1”部分 24
“步进检测电路—工作模式0，DETCON[3] = 0”部分更改为“步进检测电路”部分；更改图7标题 26
更改“偏置电压(V_{BIAS})发生器电路”部分、“数字滤波器选项”部分和图10 28
更改表16和表17；增加“同步采样”部分 29
更改“ADC数据寄存器”部分和图11标题 34
更改表22的位2 37
更改“失调校准寄存器”部分和表25 39
更改表29的位15和位14 40
更改表42和表43 47
更改表48的位6 49
更改图17 52
更改“DAC NPN晶体管驱动器模式，DACCON[8] = 1”部分 53
更改表55 56
更改“闪存保护：用户读保护”部分 88
更改“闪存控制器系统IRQ中止使能寄存器”部分和表109 95
更改“闪存控制器系统IRQ中止使能寄存器”部分和表110 97
更改“闪存控制器系统IRQ中止使能寄存器”部分和表111 98
更改“I/O上拉使能”部分 101
更改“I^2C概述”部分 107
更改“DMA模式”部分 139
更改表169的位15 149
更改“中断/唤醒信号”部分 155
更改“唤醒定时器比较寄存器A”部分 159
更改“看门狗定时器清零中断寄存器”部分 162
更改PWM部分和表200 163
更改图33和表201 164
更改表202 165
更改“H桥模式”部分和表203 166
更改“PWM触发功能中断”部分和“PWM输出对中断”部分 167
更改表205 168
更改“低压差稳压器”部分 170
更改图38 174
增加“相关链接”部分 176

2012年11月—修订版0至修订版A

更改引脚35和引脚36的描述 169

2012年10月—修订版0：初始版

使用ADuCM360/ADuCM361硬件用户指南

数字记法

表1. 数字记法

记法	描述
位N	各位按从小到大顺序格式编号，即一个数的最低有效位表示为位0。
V[X:Y]	位域表示法，涵盖一个值或一个字段(V)的位X至位Y。
0xNN	十六进制数加前缀0x。
0bNN	二进制数加前缀0b'。
NN	十进制数无前缀和后缀。

寄存器访问约定

表2. 寄存器访问约定

访问类型	描述
RW	存储器位置可读可写。
R	存储器位置只能读取。 始终读出0，除非另有说明。
W	存储器位置只能写入。

首字母缩写词和缩略语

表3. 首字母缩写词和缩略语

首字母缩写词/缩略语	描述
Σ-Δ	Sigma delta
ADC	模数转换器
AF	平均系数
AFE	模拟前端
ARM	高级精简指令集机器
CD	时钟分频器
DMA	直接存储器访问
DSB	数据同步屏障
FSE	满量程误差：增益误差加失调误差
JTAG	联合测试行动小组
LSB	最低有效字节/位
MMR	存储器映射寄存器
MSB	最高有效字节/位
NMI	无法屏蔽的中断
NVIC	嵌套矢量中断控制器
PGA	可编程增益放大器
PMU	电源管理单元
POR	上电复位
PSM	电源监控器
PWM	脉冲宽度调制器
RMS	均方根
Rx	接收
SF	Sinc3/sinc4滤波器
SIL	安全完整性水平
SPI	串行外设接口
Tx	发送
UART	通用异步发送器

ADuCM360/ADuCM361硬件用户指南

ADuCM360/ADuCM361简介

ADuCM360是完全集成的4 kSPS、24位数据采集系统，在单芯片上集成双路高性能多通道Σ-Δ型模数转换器(ADC)、32位ARM Cortex-M3®处理器和Flash/EE存储器。该器件设计为与外部精密传感器直接连接，既适合有线应用，也适合电池供电应用。

除ADC0外，ADuCM361在功能上与ADuCM360完全相同。ADuCM361仅有一个ADC，即ADC1。

该器件自带一个片内32 kHz振荡器和一个内部16 MHz高频振荡器。该时钟信号通过一个可编程时钟分频器进行中继，在其中产生处理器时钟工作频率。最大时钟速度为16 MHz，且不受工作电压或温度的限制。

微控制器采用ARM的低功耗Cortex-M3处理器，它是32位RISC机器，提供最高达20 MIPS的峰值性能。 Cortex-M3处理器集成一个灵活的11通道DMA控制器，其支持如下特性：2个SPI、UART、ADC和DAC。片内还集成128 kB非易失性Flash/EE存储器和8 kB SRAM。

拟子系统由双通道ADC组成，每个ADC均连接到一个灵活的输入多路复用器。两个ADC都可在全差分和单端模式下工作。其他的片内ADC功能包括：双通道可编程激励电流源、诊断电流源和偏置电压产生器AVDD_REG/2(900 mV)，可设置输入通道的共模电压。低端内部接地开关可在两次转换之间关断桥电路。ADC内置两个并联滤波器：一个sinc3或sinc4与一个sinc2并联。sinc3或sinc4滤波器用于精密测量。sinc2滤波器用于快速测量和输入信号的步进变化检测。该器件还内置一个低噪声、低漂移带隙基准电压源，但在采用比例式测量配置时，可配置成接受最多两个外部基准电压源。片内还集成了可缓冲外部基准电压输入的选项。片内还有一个单通道缓冲电压输出DAC。ADuCM360/ADuCM361还集成了一系列片内外设，可以根据应用需要通过微控制器软件控制进行配置。这些外设包括：UART、I²C和双通道SPI串行I/O通信控制器；19引脚GPIO端口；两个通用定时器；唤醒定时器及系统看门狗定时器。同时提供一个带6个输出通道的16位PWM。

ADuCM360/ADuCM361专为要求低功耗工作的电池供电应用而设计。微控制器可配置为普通工作模式，功耗为290 µA/MHz(包括Flash/SRAM I_{DD})，所有外设都工作时的总系统功耗为1 mA。

通过直接编程控制，该器件亦可配置为多种低功耗工作模式，包括休眠模式(内部唤醒定时器有效)，此时的功耗仅为4 µA。在休眠模式下，诸如外部中断或内部唤醒定时器等外设可以唤醒该器件。这使得该器件可在功耗极低的工作模式下运行，同时仍然响应外部异步或周期事件。

片内出厂固件支持通过串行线接口(2引脚JTAG系统)和UART进行串行在线下载，还支持通过串行线接口进行非介入式仿真。这些特性都集成在一个支持此精密模拟微控制器系列的低成本QuickStart™开发系统中。

该器件采用外部1.8 V至3.6 V电源供电，额定温度范围为−40°C至+125°C工业温度范围。

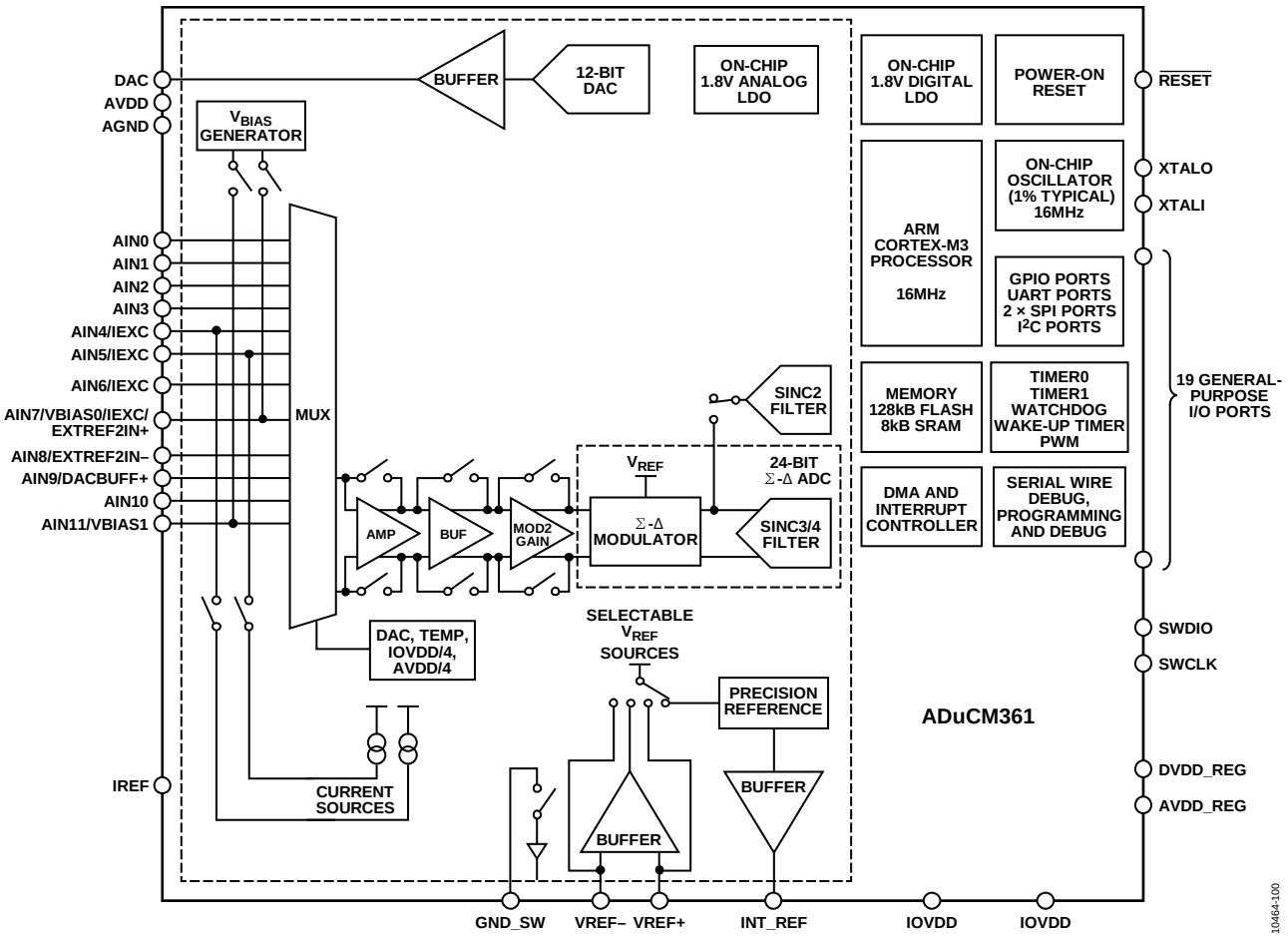


图2. ADuCM361框图

ADuCM360/ADuCM361主要特性

模拟I/O

- 两个超高精度、多通道、24位ADC
- 单端和全差分输入
- 独立可编程的ADC输出速率(3.5 Hz至3.906 kHz)
  - 50 Hz/60 Hz同时抑制：
  - 50 SPS连续转换模式
- 16.67 SPS单次转换模式
- 两个ADC均通过灵活的输入多路复用器选择输入通道
- ADC0和ADC1(24位)ADC通道
  - 6个差分或11个单端输入通道
  - 4个内部通道，用于监控DAC、温度传感器、IOVDD和AVDD(仅ADC1)
  - 可编程增益(1至128)
  - 可选输入范围(详情参见ADuCM360/ADuCM361数据手册)
  - RMS噪声(详情参见ADuCM360/ADuCM361数据手册)
- 可编程传感器激励电流源
  - 10 μA/50 μA/100 μA/150 μA/200 μA/250 μA/300 μA/400 μA/450 μA/500 μA/600 μA/750 μA/800 μA和1 mA电流源选项
- 片内精密基准电压源(详情参见ADuCM360/ADuCM361数据手册)

单通道12位电压输出DAC

- 电压输出DAC
- 也可配置为NPN晶体管驱动器模式和插值模式

微控制器

- ARM Cortex-M3 32位处理器
- 串行线下载和调试
- 用于唤醒定时器的内部时钟晶体
- 具有8路可编程分频器的16 MHz振荡器

电源

- 3.0 V电池直接供电
- 电源电压范围：1.8 V至3.6 V
- 针对低功耗应用，提供灵活的工作模式
- 功耗：
  - 处理器活动模式：内核功耗290 μA/MHz
  - 活动模式：1 mA(所有外设有效，处理器工作频率为500 kHz)
  - 关断模式：4 μA(唤醒定时器有效)

片内外设

- UART、I^2C和两个SPI串行I/O通信控制器
- 19引脚GPIO端口
- 2个通用定时器
- 唤醒定时器
- 看门狗定时器
- 16位PWM控制器
- 多通道DMA和中断控制器

封装和温度范围

- 48引脚LFCSP (7 mm × 7 mm)封装，−40°C至+125°C

工具

- 低成本QuickStart开发系统
- 支持第三方编译器和仿真器工具

应用

- 工业自动化和过程控制
- 智能精密检测系统
- 4 mA至20 mA环路供电智能传感器系统

存储器结构

本节说明ADuCM360/ADuCM361存储器结构。

用户可以访问三个不同的存储器模块：

- 8 kB SRAM，从0x20000000到0x20001FFF
- 128 kB片内Flash/EE存储器供用户使用，从0x00000到0x1FFFF
- 另有2 kB保留用于内核空间，从0x20000到0x207FF

这些模块根据Cortex-M3存储器映射进行映射，如图3所示。 所有片内外设通过位于位带区中的存储器映射寄存器访问。

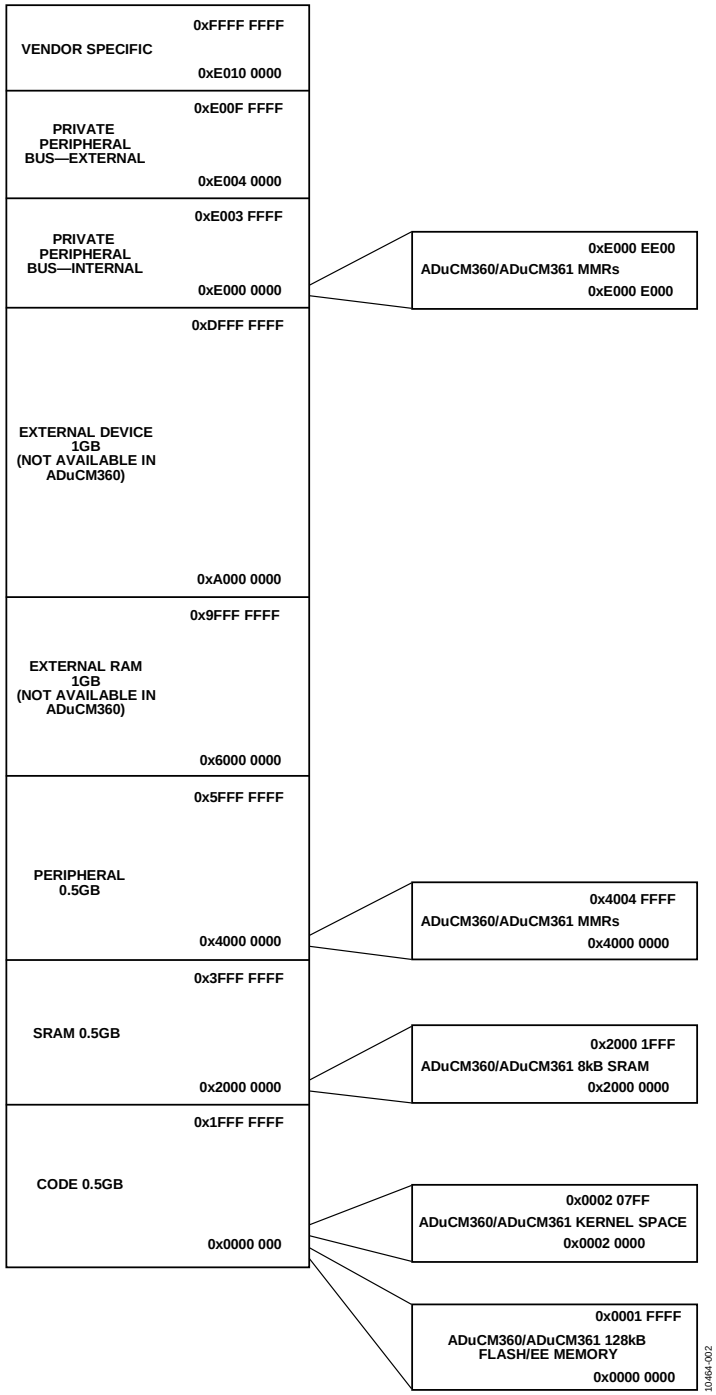


图3. ADuCM360/ADuCM361 Cortex-M3存储器映射图

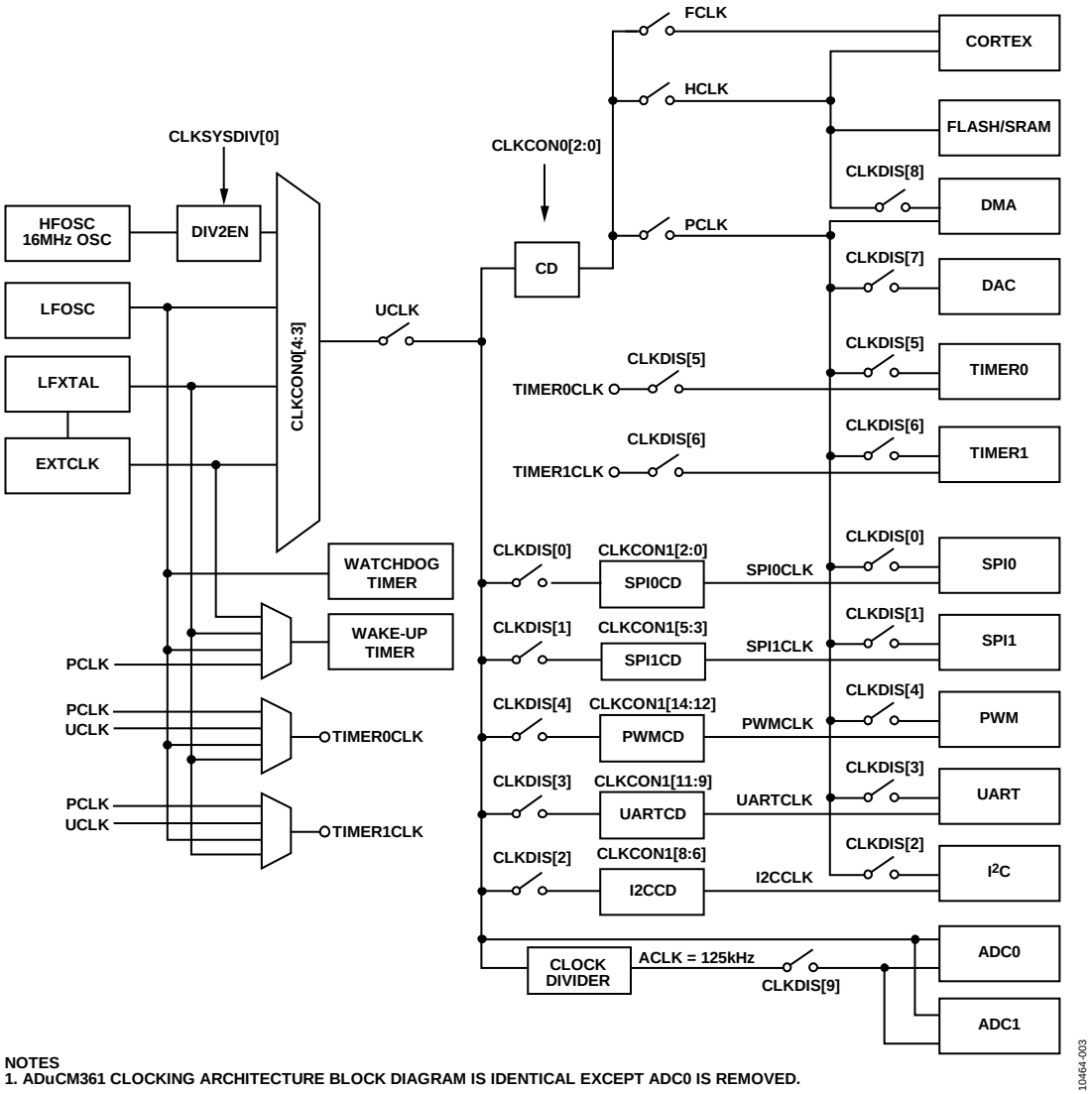
时钟架构

时钟架构特性

ADuCM360/ADuCM361集成两个片内振荡器和外部晶体电路：

- LFOSC为一个32 kHz低功耗内部振荡器，用于低功耗模式。
- HFOSC为一个16 MHz内部振荡器，用于活动模式。
- LFXTAL为32 kHz外部晶体。
- 省电时钟机制；每个外设都可以使能或禁用。

时钟架构功能框图



NOTES
1. ADuCM361 CLOCKING ARCHITECTURE BLOCK DIAGRAM IS IDENTICAL EXCEPT ADC0 IS REMOVED.

图4. ADuCM360时钟架构框图
注意：除了没有ADC0之外，ADuCM361时钟架构框图完全相同

时钟架构概述

三个时钟源—LFOSC、HFOSC和LFXTAL—可以用作系统时钟。P1.0上的外部时钟也可以用于测试目的。

在内部，系统时钟分频为以下五个时钟：

- FCLK，用于cortex处理器
- UCLK系统时钟
- HCLK，用于闪存、SRAM和DMA
- ACLK，用于模拟前端和ADC时钟
- PCLK，用于其它外设

图4显示了所有可用的时钟，包括用于电源管理的时钟门控。 有关时钟门控的更多信息，参见“电源管理单元”部分。

时钟架构工作原理

上电时，处理器从16 MHz内部振荡器分频获得时钟，以1 MHz速率运行。用户代码可选择系统时钟的时钟源，并以2^{CD}的系数将该时钟分频，CD为时钟分频器位(CLKCON0[2:0])。这样可以减慢代码执行速度，降低功耗。

注意，通过时钟控制寄存器切换时钟源之前，先必须将P1.0配置为时钟输入。

存储器

- 128 kB Flash/EE存储器，8 kB SRAM
- Flash/EE耐久性：10000周期
- Flash/EE保持时间：10年
- 通过串行线和UART在线下载

时钟架构存储器映射寄存器

表4. 时钟控制存储器映射寄存器地址表

地址	名称	描述	访问类型	默认值
0x40002000	CLKCON0	系统时钟控制寄存器0	RW	0x0004
0x40002004	CLKCON1	系统时钟控制寄存器1	RW	0x0000
0x4000202C	CLKDIS	外设系统时钟使能/禁用	RW	0x01FF
0x40002410	XOSCCON	晶振控制寄存器	RW	0x0000
0x40002444	CLKSYSDIV	系统时钟2分频控制寄存器	RW	0x0001

时钟架构控制寄存器0

地址：0x40002000；复位：0x0004；名称：CLKCON0

表5. CLKCON0寄存器位功能描述

位	名称	描述
15:8	保留	保留
7:5	CLKOUT	时钟输出复用器选择位(仅用于测试目的) 000: UCLK(默认) 001: UCLK 010: PCLK 011: 保留 100: 保留 101: HFOSC 110: LFOSC 111: LFXTAL
4:3	CLKMUX	时钟输入复用器选择位 00: HFOSC(默认) 01: LFXTAL 10: LFOSC 11: EXTCLK
2:0	CD	时钟分频位[1] 000: DIV1: UCLK/1 001: DIV2: UCLK/2 010: DIV4: UCLK/4 011: DIV8: UCLK/8 100: DIV16: UCLK/16(默认) 101: DIV32: UCLK/32 110: DIV64: UCLK/64 111: DIV128: UCLK/128

[1] 若CLKSYSDIV[0]设为1，则需要额外的2分频。

时钟架构控制寄存器1

地址：0x40002004；复位：0x0000；名称：CLKCON1[1]

表6. CLKCON1寄存器位功能描述

位	名称	描述
15	保留	
14:12	PWMCD	PWM系统时钟的时钟分频位[2] 000: UCLK/1 = 16 MHz 001: UCLK/2 = 8 MHz 010: UCLK/4 = 4 MHz 011: UCLK/8 = 2 MHz 100: UCLK/16 = 1 MHz 101: UCLK/32 = 500 kHz 110: UCLK/64 = 250 kHz 111: UCLK/128 = 125 kHz
11:9	UARTCD	UART系统时钟的时钟分频位[2] 000: UCLK/1 = 16 MHz 001: UCLK/2 = 8 MHz 010: UCLK/4 = 4 MHz 011: UCLK/8 = 2 MHz 100: UCLK/16 = 1 MHz 101: UCLK/32 = 500 kHz 110: UCLK/64 = 250 kHz 111: UCLK/128 = 125 kHz
8:6	I2CCD	I²C系统时钟的时钟分频位[2] 000: UCLK/1 = 16 MHz 001: UCLK/2 = 8 MHz(支持400 kHz I²C波特率的最小值) 010: UCLK/4 = 4 MHz 011: UCLK/8 = 2 MHz(支持100 kHz I²C波特率的最小值) 100: UCLK/16 = 1 MHz 101: UCLK/32 = 500 kHz 110: UCLK/64 = 250 kHz 111: UCLK/128 = 125 kHz
5:3	SPI1CD	SPI1系统时钟的时钟分频位[2] 000: UCLK/1 = 16 MHz 001: UCLK/2 = 8 MHz 010: UCLK/4 = 4 MHz 011: UCLK/8 = 2 MHz 100: UCLK/16 = 1 MHz 101: UCLK/32 = 500 kHz 110: UCLK/64 = 250 kHz 111: UCLK/128 = 125 kHz
2:0	SPI0CD	SPI0系统时钟的时钟分频位[2] 000: UCLK/1 = 16 MHz 001: UCLK/2 = 8 MHz 010: UCLK/4 = 4 MHz 011: UCLK/8 = 2 MHz 100: UCLK/16 = 1 MHz 101: UCLK/32 = 500 kHz 110: UCLK/64 = 250 kHz 111: UCLK/128 = 125 kHz

[1] 外设时钟必须大于或等于FCLK。 FCLK由CLKCON0[2:0]设置。

[2] 计算针对UCLK = 16 MHz且CLKSYSDIV[0] = 0。CLKSYSDIV[0]置1时，需要额外的二分频。

时钟架构禁用寄存器

地址：0x4000202C；复位：0x01FF；名称：CLKDIS

表7. CLKDIS寄存器位功能描述

位	名称	描述
15:10	保留	这些位保留，应写入0。
9	DISADCCLK	0: 使能ADC系统时钟(默认)。 1: 禁用ADC系统时钟。
8	DISDMACLK	0: 使能DMA系统时钟。 1: 禁用DMA系统时钟。
7	DISDACCLK	0: 使能DAC系统时钟。 1: 禁用DAC系统时钟。
6	DIST1CLK	0: 使能Timer1系统时钟。 1: 禁用Timer1系统时钟。
5	DIST0CLK	0: 使能Timer0系统时钟。 1: 禁用Timer0系统时钟。
4	DISPWMCLK	0: 使能PWM系统时钟。 1: 禁用PWM系统时钟。
3	DISUARTCLK	0: 使能UART系统时钟。 1: 禁用UART时钟。
2	DISI2CCLK	0: 使能I^2C系统时钟。 1: 禁用I^2C系统时钟。
1	DISSPI1CLK	0: 使能SPI1系统时钟。 1: 禁用SPI1系统时钟。
0	DISSPI0CLK	0: 使能SPI0系统时钟。 1: 禁用SPI0系统时钟。

外部晶振控制寄存器

地址：0x40002410；复位：0x0000；名称：XOSCCON

表8. XOSCCON寄存器位功能描述

位	名称	描述
7:3	保留	这些位保留，应写入0。
2	DIV2	2分频使能位。 0: 禁用时钟分频器。 1: 使能时钟分频器。
1	Reserved	此位保留，应写入0。
0	ENABLE	晶振电路使能。 0: 禁用振荡器电路。 1: 使能振荡器电路。

时钟架构分频寄存器

地址：0x40002444；复位：0x0001；名称：CLKSYSDIV

表9. CLKSYSDIV寄存器位功能描述

位	名称	描述
7:1	保留	这些位保留，应写入0。
0	DIV2EN	2分频使能位。此位默认值为1，意味着系统时钟为8 MHz。 低功耗系统应使能此位。 0: 禁用时钟分频器，系统时钟为16 MHz。 1: 使能时钟分频器，系统时钟为8 MHz(默认)。

电源管理单元

电源管理单元特性

电源管理单元(PMU)控制ADuCM360/ADuCM361的不同功耗模式。

有6种功耗模式：

- 活动
- MCUHALT
- PERHALT
- SYSHALT
- TOTALHALT
- 休眠

电源管理单元概述

PMU控制ADuCM360/ADuCM361的功耗模式。

Cortex-M3睡眠模式已链接到PMU模式，本部分将予以说明。PMU始终开启。每种模式都可以降低功耗，不过功能也会相应地减少。限制最多模式下的关断功耗约为4 μA，限制最少模式下的关断功耗取决于用户选择的CD时钟速率，比率大约为290 μA/MHz。

ADuCM360/ADuCM361数据手册提供了所有电流值。

表10. 功耗模式总结

模式	名称	描述	估计电流[1] CLKSYSDIV = 0	估计电流 CLKSYSDIV = 1	唤醒时间
0	活动	完全活动模式。	250 μA + 290 μA/MHz	145 μA + 145 μA/MHz	
1	MCUHALT	当Cortex-M3也处于睡眠模式时，关闭HCLK。	435 μA + 50 μA/MHz	355 μA + 50 μA/MHz	3 × FCLK至5 × FCLK
2	PERHALT	当Cortex-M3也处于睡眠模式时，关闭PCLK。	425 μA + 60 μA/MHz	345 μA + 60 μA/MHz	3 × FCLK至5 × FCLK
3	SYSHALT	当Cortex-M3处于睡眠模式时，关闭HCLK、ACLK和PCLK。	420 μA + 16 μA/MHz	345 μA + 16 μA/MHz	3 × FCLK至5 × FCLK
4	TOTALHALT	当Cortex-M3处于深度睡眠模式时，关闭FCLK。	~4 μA	~4 μA	30.8 μs
5	休眠	关闭所有数字模块，始终开启的模块除外。	~4 μA	~4 μA	30.8 μs

[1] 估计电流 = 静态电流 + 动态电流，其中动态电流与频率相关。给出的值为T_A = 25°C时的典型值。

电源管理单元工作原理

通过设置调试逻辑中的位，调试工具可以防止Cortex-M3完全进入省电模式。只有上电复位才能复位调试逻辑。因此，若应用代码包含WFI指令，则使用串行线调试之后，应当对器件进行周期供电。

功耗模式： 活动模式，模式0

系统全部功能都有效。存储器和所有用户使能的外设都有时钟，Cortex-M3处理器执行指令。注意，Cortex-M3处理器管理其内部时钟，可以处于部分时钟选通状态。此时钟门控仅影响内部Cortex-M3处理内核。所有模块都使用自动时钟门控，这对用户是透明的。用户代码可利用WFI命令将Cortex-M3处理器置于睡眠模式，它独立于PMU的功耗模式设置。

当ADuCM360/ADuCM361从任一低功耗模式唤醒时，器件返回到模式0。

功耗模式： MCUHALT模式，模式1

Cortex-M3处理器进入睡眠模式之后，系统很快就会关闭HCLK。HCLK是SRAM、闪存和DMA模块使用的时钟。关闭此时钟会停止所有这些外设。Cortex-M3处理器FCLK有效，器件利用NVIC唤醒。若在DMAENSET寄存器中使能ADC0、ADC1或sinc2 DMA通道，则DMA模块和SRAM的时钟仍然有效。

功耗模式： PERHALT模式，模式2

Cortex-M3处理器进入睡眠模式之后，系统很快就会关闭PCLK。PCLK是用户外设的时钟。关闭此时钟会停止这些外设的所有时钟。Cortex-M3处理器FCLK有效，器件利用NVIC唤醒。若在DMAENSET寄存器中使能ADC0、ADC1或sinc2 DMA通道，则DMA模块和SRAM的时钟仍然有效。

功耗模式： SYSHALT模式，模式3

Cortex-M3处理器进入睡眠模式之后，系统很快就会关闭HCLK和PCLK。Cortex-M3处理器FCLK有效，器件利用NVIC唤醒。

功耗模式： TOTALHALT模式，模式4

Cortex-M3处理器进入深度睡眠模式之后，系统很快就会关闭FCLK、HCLK和PCLK。FCLK是Cortex-M3处理器的自由振荡时钟。关闭FCLK、HCLK和PCLK时钟会停止某些NVIC功能。因此，Cortex-M3处理器FCLK停止，仅表55列出的外设可以唤醒Cortex-M3处理器。进入模式4时，低功耗LDO自动开启，高功耗LDO自动关闭。

功耗模式： 休眠模式，模式5

系统关闭数字内核的电源。关闭此电源期间会保留所有状态，对用户而言，在Cortex-M3处理器进入深度睡眠模式之后，系统很快就会关闭FCLK、HCLK和PCLK，但漏电流更低，响应时间不同(即器件需要较长时间才能完成上电)。因此，Cortex-M3处理器FCLK停止，仅表55列出的外设可以唤醒Cortex-M3处理器。

在功耗模式4和功耗模式5下使能的低功耗LDO仅能提供100 μA电流(典型值)。因此，用户需要注意，切勿在数字输入上提供高频时钟信号(即大于1 MHz的信号)，因为这种信号可能导致内部电路的功耗超过此100 μA限值。故而当P1.0上存在外部时钟时，不支持功耗模式4和功耗模式5。然而，若数字引脚用作数字输出并提供电流，则支持这些模式，因为此电流由IOVDD电源提供，而不是由低功耗LDO提供。

当处理器处在中断处理程序中时，若进入功耗模式1至功耗模式5，则只能通过复位或更高优先级中断源退出关断模式。

电源管理单元存储器映射寄存器

功耗模式由位于始终开启的部分中的单个寄存器控制，其地址为0x40002400。

表11. 系统时钟存储器映射寄存器地址(基地址：0x40002400)

偏移	名称	描述	访问类型	默认值
0x0000	PWRMOD	功耗模式寄存器	RW	0x00
0x0004	PWRKEY	PWRMOD的密钥保护	RW	不适用

功耗模式控制寄存器

地址：0x40002400；复位：0x0000；名称：PWRMOD

表12. PWRMOD寄存器位功能描述

位	名称	描述
7:3	保留	保留。 应由用户代码向这些位写入0。
2:0	MOD	功耗模式控制位。 选择要进入的功耗模式。读取时，这些位包含 通过用户代码进入的最后功耗模式值。 000: 完全活动。 001: MCUHALT。 010: PERHALT。 011: SYSHALT。 100: TOTALHALT。 101: 休眠。 其它: 保留。

当器件处于模式4或模式5时，要将cortex置于深度睡眠模式，必须将Cortex-M3处理器系统控制寄存器(地址0xE000ED10)的位2 (SCR[2])配置为1。

功耗模式密钥寄存器

地址：0x40002400；复位：不适用；名称：PWRKEY

表13. PWRKEY寄存器位功能描述

位	名称	描述
15:0	值	功耗控制密钥寄存器。PWRMOD寄存器受密钥保护。要更改PWRMOD寄存器的值，需要两次写入密钥：先写入0x4859，再写入0xF27B。然后应写入PWRMOD寄存器。若在写入PWRMOD之前写入其它寄存器，则保护将返回锁定状态。

进入省电模式的代码示例

```
SCB->SCR = 0x04;                    // sleepdeep mode
pADI_PWRCTL->PWRKEY = 0x4859;       // key1
pADI_PWRCTL->PWRKEY = 0xF27B;       // key2
pADI_PWRCTL->PWRMOD = 0x4;
asmwfi();
__asm void asmwfi()
{
                        nop
                        nop
                        nop
                        WFI
                        nop
                        nop
                        BX LR
}
```

CORTEX-M3处理器

CORTEX-M3处理器特性

高性能

- 1.25 DMIPS/MHz。
- 包括乘法在内的很多指令都是单周期指令。
- 数据和指令各有总线，因此数据访问和指令访问可同时进行。
- 针对单周期闪存使用进行了优化。

低功耗

- 低待机电流。
- 内核利用高级时钟门控实现，仅主动使用的逻辑产生动态功耗。
- 支持省电模式(睡眠和深度睡眠模式)。设计具有不同的时钟，允许停止不使用的处理器部分。

高级中断处理

- 嵌套矢量中断控制器(NVIC)最多支持240个中断。ADuCM360/ADuCM361支持其中的38个中断。矢量中断特性大大减少了中断延迟，因为无需通过软件来确定使用哪个中断处理程序。另外也无需通过软件实现嵌套中断支持。
- Cortex-M3处理器自动在中断入口将寄存器推到堆栈上，然后在中断出口弹回寄存器。这就减少了中断处理延迟，使得中断处理程序可以是普通C函数。
- 动态控制各中断的优先级。
- 利用迟到中断接受机制和尾链中断入口减少延迟。
- 对于安全关键型应用，立即执行无法屏蔽的中断请求。

系统特性

- 支持位带操作和不对齐的数据访问。
- 高级故障处理特性包括多种异常类型和故障状态寄存器。

调试支持

- 串行线调试接口(SW-DP)。
- 利用闪存补丁和断点(FPB)单元实现断点。以六个活动断点为限。
- 利用数据观察点和触发(DWT)单元实现观察点触发资源和系统分析。以两个活动观察点为限。

CORTEX-M3处理器概述

ADuCM360/ADuCM361内置一个嵌入式ARM Cortex-M3处理器(r2p0版本，AT420)。ARM Cortex-M3提供一种高性能、低成本平台，其满足存储器最小、引脚数量少、功耗低的系统要求，同时具有出色的计算性能和非凡的系统中断响应性能。

CORTEX-M3处理器工作原理

Cortex-M3的多个组件在实现上具有灵活性。本节详细说明ADuCM360/ADuCM361中这些组件的实际实现。

串行线调试(SW/JTAG-DP)

ADuCM360/ADuCM361仅通过SWCLK和SWDIO引脚支持串行线接口。它不支持5线JTAG接口。

ROM表

ADuCM360/ADuCM361实现了默认ROM表。

嵌套矢量中断控制器中断(NVIC)

Cortex-M3处理器内置一个嵌套矢量中断控制器(NVIC)，其具有如下几个特性：

- 支持嵌套中断
- 支持矢量中断
- 支持动态优先级变更
- 中断屏蔽

此外，NVIC有一个无法屏蔽中断(NMI)输入。

NVIC在ADuCM360/ADuCM361上实现，更多详情参见“系统异常和外设中断”部分。

唤醒中断控制器(WIC)

ADuCM360/ADuCM361有一个改良的WIC，其提供尽可能低的关断电流。此特性对用户是透明的，更多详情参见“电源管理单元”部分。

注意，如果器件在处理中断期间进入省电模式，则只有更高优先级的中断源才能将其唤醒。

µDMA

ADuCM360/ADuCM361实现了ARM µDMA。更多详情参见“DMA控制器”部分。

相关文件

- Cortex-M3 r2p0版技术参考手册(DDI0337)
- Cortex-M3勘误通知：Cortex-M3和带ETM的Cortex-M3 (AT420/AT425)
- ARMv7-M架构参考手册(DDI0403)
- ARMv7-M架构参考手册勘误标记
- ARM调试接口v5 (IHI0031)
- PrimeCell µDMA控制器(PL230)技术参考手册r0p0版(DDI0417)

ADC电路

ADC电路特性

- ADuCM360：两个Σ-Δ ADC (ADC0和ADC1)
- ADuCM361：一个Σ-Δ ADC (ADC1)
- 6路全差分输入
- 11路单端输入

ADC电路功能框图

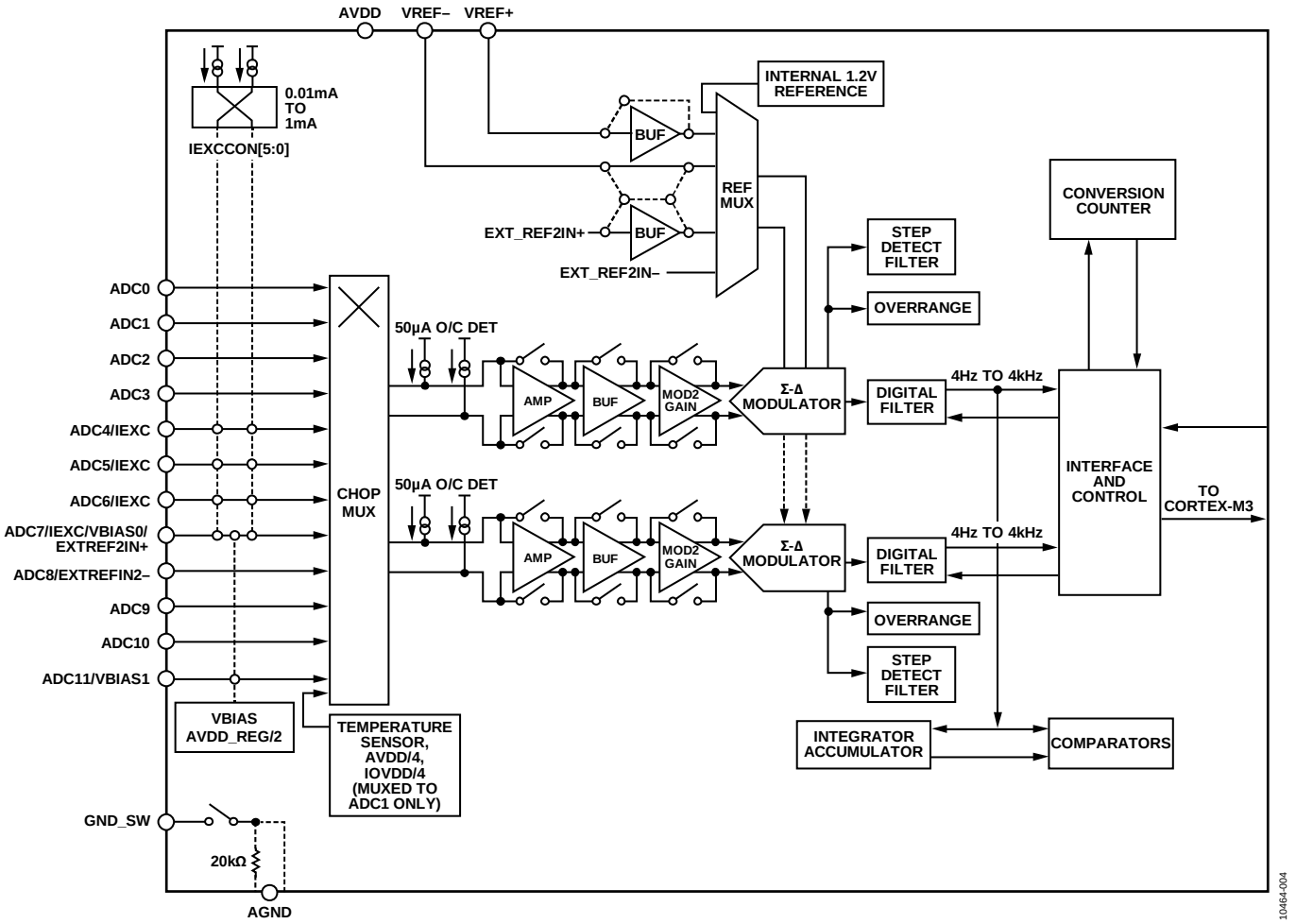


图5. ADC框图

ADC电路概述

ADuCM360集成两个独立的多通道Σ-Δ ADC (ADC0和ADC1)。

ADuCM361集成一个多通道Σ-Δ ADC (ADC1)。

两个ADC连接到同一输入多路复用器，可配置为6路全差分输入或11路单端输入。该器件还内置一个低噪声、低漂移带隙基准电压源。该器件可接受两个外部基准电压源，提供了缓冲这些输入的选项。其它的片内ADC特性包括：两个可编程激励电流源、诊断电流源和AVDD_REG/2的偏置电压产生器(用于设置输入通道的共模电压)。低端内部接地开关在ADC转换过程中关断外部传感器电源。

两个ADC共用一个输入多路复用器。每个ADC有单独的输入缓冲器和低噪声可编程增益放大器(PGA)，小幅度信号可直接连到各ADC的Σ-Δ调制器。模拟输入引脚可配置为全差分输入或单端输入。在单端模式下，ADCxCON寄存器中的ADCCN选择的通

道充当ADC的负输入通道基准。对于所有其它单端正输入，该通道可用作负输入。这些输入可以接地。ADC1还支持4个内部通道：AVDD/4、IOVDD/4、温度传感器和DAC。ADC彼此独立工作，各有自己的Cortex-M3处理器中断源和配置寄存器集。

为了快速处理ADC输入，ADC0和ADC1均可选择较快的sinc2滤波器与普通sinc3或sinc4滤波器并行工作。 此滤波器每2 ms输出一次数据，可连续读取。或者，sinc2滤波器也可用于快速检测ADC输入通道上的步进变化。此外，为了在高采样速率(488 Hz至4 kHz)时实现更高的精度，提供了一个sinc4滤波器。

若在ADC转换过程中或校准例程中出错(例如丢失基准电压)，则ADCxDAT寄存器会读出特定增益设置所对应的满量程，ADCxSTA[4]置1表示发生上溢错误。对于下溢错误，ADCxDAT报告负满量程，ADCxSTA[4]同样置1。

ADuCM361无ADC0。

ADuCM361的ADC1与ADuCM360的ADC1完全相同。

ADC电路工作原理

模拟输入通道配置

两个ADC连接到同一输入多路复用器。

ADC控制寄存器的ADCCP (ADCxCON[9:5])和ADCCN (ADCxCON[4:0])位用于配置各ADC的输入信号，如下所示：

- ADC0CON[9:5]：这些位选择ADC0的正输入通道。可选择下列任意输入通道作为来源： AIN0/AIN1/AIN2/AIN3/AIN4/AIN5/AIN6/AIN7/AIN8/AIN9/AIN10/AIN11或AGND。
- ADC0CON[4:0]：这些位选择ADC0的负输入通道。可选择下列任意输入通道作为来源： AIN0/AIN1/AIN2/AIN3/AIN4/AIN5/AIN6/AIN7/AIN8/AIN9/AIN10/AIN11或AGND。
- ADC1CON[9:5]：这些位选择ADC1的正输入通道。可选择下列任意输入通道作为来源： AIN0/AIN1/AIN2/AIN3/AIN4/AIN5/AIN6/AIN7/AIN8/AIN9/AIN10/AIN11或任意内部通道(即DAC输出、温度传感器通道、AVDD ÷ 4或IOVDD ÷ 4)。
- ADC1CON[4:0]：这些位选择ADC1的负输入通道。可选择下列任意输入通道作为来源： AIN0/AIN1/AIN2/AIN3/AIN4/AIN5/AIN6/AIN7/AIN8/AIN9/AIN10/AIN11或任意内部通道(即DAC输出、温度传感器通道、AVDD或IOVDD)。

输入通道可以设置为缓冲或无缓冲。在缓冲模式下，所选输入通道馈入放大器的高阻抗输入级。这使得ADC能够耐受较大的源阻抗，并且可以与阻性传感器直接相连，例如RTD和应变计等。在无缓冲模式下，此输入缓冲器被旁路，输入电压范围更宽。缓冲器由ADCxCON[17:14]控制。

详情参见ADuCM360/ADuCM361数据手册中的技术规格。

ADuCM360支持这样一种工作模式，即无论增益如何设置，每个PGA皆可直接驱动ADC调制器，因而可以旁路并关断输入缓冲器。因为每个正负输入缓冲器的典型功耗为35 µA，因此，当两个ADC均使能时，上述模式可节省140 µA的I_{DD}。ADuCM361也支持这种工作模式，不过仅在ADC1上支持。从完成切换输入通道到获得第一个有效(完全建立)结果的时间如表16中的$t_{SETTLING}$所示。

将同一输入通道连接到两个ADC的正输入(仅ADuCM360)

ADuCM360有两个独立的ADC和一个高度灵活的输入多路复用器。因此，两个ADC的正输入可连接到同一输入通道。

多数应用会将各ADC的输入通道分开。但在某些情况下，例如工业应用中，可能需要在两个ADC上测量同一输入以改善安全完整性水平(SIL)级别。

若PGA使能(增益 ≥ 2)，则两个ADC对同一输入进行采样时的ADC输出值与仅一个ADC对此输入进行采样时的ADC输出值不同。原因在于，模拟输入连接到两个ADC时的输入电流与仅连接到一个ADC时的输入电流不同。

用两个ADC测量同一输入通道时，建议各ADC使用各自的零电平和满量程校准值。这应通过ADC的系统校准模式产生。

PGA增益配置

两个ADC均集成一个片内可编程增益放大器(PGA)。PGA有8种不同的设置，范围为1到128。增益由ADC0MDE[7:4]和ADC1MDE[7:4]控制寄存器控制。PGA可以放大极小幅度的信号，同时仍能保持低噪声性能。在单极性模式(基于1.2 V ADC基准电压源)下，增益为1时，输入范围是0 V到1.2 V；增益为128时，输入范围是0 mV到7.8125 mV。在双极性模式下，增益为1时，输入范围是0 V到±1.2 V；增益为128时，输入范围是0 mV到±7.8125 mV。

选择增益为1时，PGA不使能。仅当增益为2、4、8、16、32、64和128时，才会使能PGA。当PGA使能时，PGA输出电压不得超过1 V。因此，当增益为4时，可施加的最大输入电压为±250 mV；当增益为8时，可施加的最大输入电压为±125 mV。关于所有ADC的有效输入电压范围的完整列表，请参见ADuCM360/ADuCM361数据手册。为了确定ADC输入是否会超过1 V的PGA输出阈值，可以使用ADC比较器。下面的代码示例说明了如何将比较器阈值设置为1 V。若超过阈值，则置位一个中断。此代码对从2到128的所有ADC0 PGA增益设置有效。

代码示例

```
void ADC0INIT(void)
{
        pADI_ADC0->MSKI = 0x5;                  // Enable ADC0 /rdy IRQ          + ADCxTHEX interrupt
          pADI_ADC0->PRO = 0x4;                 // Enable threshold detection
         pADI_ADC0->TH = 0x6AA2;                // Set threshold to 1 V max PGA output
                                                // Add rest of ADC init code after this
}
In the interrupt handler, add the following:
 uiADCSTA = pADI_ADC0->STA;                     // read ADC status register
   if ((uiADCSTA & 0x4) == 0x4)                 // Check for ADC0TH exceeded condition
  {
                ucADC0ERR = 4;                  // 1 V threshold exceeded
               pADI_ADC0->MSKI &= 0xFB;         // Disable threshold detection to clear this interrupt source
                pADI_ADC0->PRO = 0;             // Disable threshold detection to clear this interrupt source
 }

Optional—reenable in main loop:
                                                // Disable threshold detection to clear this interrupt source
                if (ucADC0ERR == 0x4)
                {
                        ucADC0ERR = 0;
                       pADI_ADC0->MSKI |= 0x4;
                       pADI_ADC0->PRO = 4;

                }
```

ADC调制器中可增加一个可选的额外增益2，也就是说，PGA输出可增加2倍的增益。1 V PGA输出限值仍然存在，但此附加特性可增加0.5 LSB rms位的分辨率。

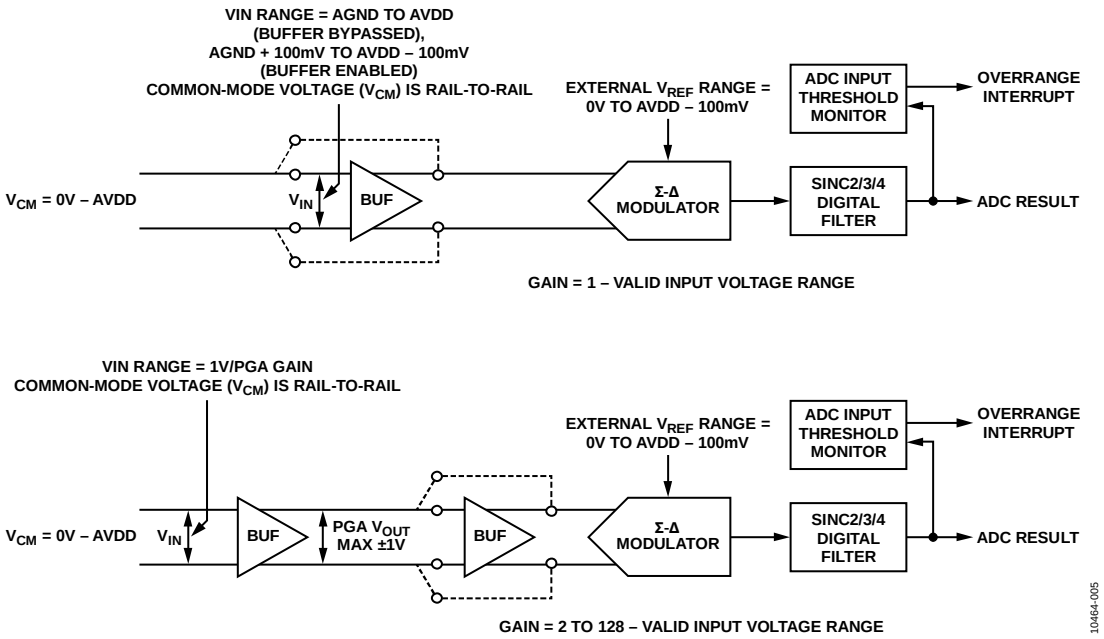


图6. ADC输入

表14. 所有增益对应的绝对输入电压和共模电压范围

增益	绝对输入电压范围[1] (缓冲器关闭)	绝对输入电压范围[1] (缓冲器开启)	V_{CM}范围 (输入缓冲器开启)	V_{CM}范围 (输入缓冲器关闭)
1	AGND至AVDD	AGND + 100 mV至AVDD – 100 mV	AGND + 100 mV至AVDD – 100 mV	AGND至AVDD[2]
2	0 V至500 mV	0 V至500 mV	AGND + 100 mV至AVDD – 100 mV	AGND至AVDD[2]
4	0 V至250 mV	0 V至250 mV	AGND + 100 mV至AVDD – 100 mV	AGND至AVDD[2]
8	0 V至125 mV	0 V至125 mV	AGND + 100 mV至AVDD – 100 mV	AGND至AVDD[2]
16	0 V至62.5 mV	0 V至62.5 mV	AGND + 100 mV至AVDD – 100 mV	AGND至AVDD[2]
16 + MOD2	0 V至62.5 mV	0 V至62.5 mV	AGND + 100 mV至AVDD – 100 mV	AGND至AVDD[2]
32	0 V至22.18 mV	22.18 mV	AGND + 100 mV至AVDD – 100 mV	AGND至AVDD[2]
32 + MOD2	0 V至15.625 mV	0 V至15.625 mV	AGND + 100 mV至AVDD – 100 mV	AGND至AVDD[2]
64	0 V至15.625 mV	0 V至10.3125 mV	AGND + 100 mV至AVDD – 100 mV	AGND至AVDD[2]
64 + MOD2	0 V至7.8125 mV	0 V至7.8125 mV	AGND + 100 mV至AVDD – 100 mV	AGND至AVDD[2]
128	0 V至7.8125 mV	0 V至3.98 mV	AGND + 100 mV至AVDD – 100 mV	AGND至AVDD[2]

[1] 绝对输入电压为差分电压范围AIN+至AIN–。

[2] 若旁路ADC输入缓冲器，共模电压(V_{CM})可配置为轨到轨。 但在供电轨的200 mV范围内，输入电流会提高。

双极性/单极性配置

ADuCM360/ADuCM361的模拟输入可以接受单极性或双极性输入电压范围。

双极性范围不代表器件可以处理相对于地的负电压。它只是意味着，只要不超过绝对输入电压范围，输入范围便可在共模电压上下以V_{REF}的幅值变化。

在单极性模式下，模拟输入电压范围降为共模电压与正输入电平之和，因此，正输入不得低于共模电压。

例如，若ADC0配置为差分模式，采用外部1.0 V基准电压源，且ADC负输入设置为1.5 V(共模电压 = 1.5 V)，则双极性模式下的有效输入电压范围为0.5 V至2.5 V，单极性模式下为1.5 V至2.5 V。此外，在单极性模式下，若输入电压为负值(AIN– > AIN+)，则ADCxDAT箝位在0x00000000，ADCxSTA[4](错误位)置1。

全差分和单端输入

两个ADC均支持两种电压输入类型。在全差分模式下，用户可以选择两个ADC的任意输入对。外部电路决定共模电压，因此，外部电路的配置必须确保差分信号不超过该输入的最小/最大绝对输入电压。或者，用户可将AIN7或AIN11配置为共模输出引脚(V_{BIAS}模式)，然后使用该引脚来偏置差分对。

在单端模式下，负输入由ADCxCON[4:0]控制位选择。若负输入引脚接地且PGA增益设置为1，则必须旁路输入缓冲器。否则，必须将负输入偏置到最小输入电压值以上。

无论增益设置为何，两个ADC的PGA均可直接驱动ADC调制器。因此，负输入可直接连到AGND。

ADC典型工作模式

ADC可配置为三种一般工作模式中的一种：转换模式、校准模式或关断模式。

ADC转换模式

ADC转换模式支持固定速率的连续转换或软件触发的单次转换。

- 单次转换：在软件中将ADC0MDE/ADC1MDE寄存器的位1置1 (ADCxMDE[2:0] = 010)，可启动ADC单次转换。单次转换完成后，ADC回到空闲模式。
- 连续转换：在软件中将ADC0MDE/ADC1MDE寄存器的位0置1 (ADCxMDE[2:0] = 001)，可启动连续转换。在连续转换模式下，ADCxDAT寄存器以ADCxFLT寄存器选择的采样速率更新。

在任一模式下，当转换完成时，ADCxRDY位(ADCxSTA[0])都会置位，表示ADC0DAT、ADC1DAT或STEPDAT寄存器中存在ADC结果，可供读取。ADCxRDY和sinc2滤波器的输出可将一个中断标志置位，以告知ARM Cortex-M3处理器。若任一ADC的输入电压欠范围或超范围，导致转换出错，则相应的ADCxSTA[4]寄存器错误位会置1。

ADC中断处理程序还应检查ADC比较器阈值位(ADCxSTA[2])，确保不超过1 V PGA输出限值(更多信息参见“PGA增益配置”部分)。

ADC校准模式

ADC校准模式用于消除ADC和可能的系统误差。

建议使用所需的系统PGA增益设置，执行系统零点和系统满量程校准。

进入校准模式之前，要校准的ADC必须处于空闲模式(ADCxMDE[2:0] = 011)。此外，仅当ADC处于空闲模式时，才能写入ADCxOF和ADCxGN寄存器。

- 内部零点校准。在此模式下，利用内部产生的0 V信号对任何使能的ADC进行失调校准。校准是在用户编程的ADC设置下进行的；因此，和正常单次ADC转换一样，需要两到三个ADC转换周期，校准结果才能完全建立。校准结果自动写入相应ADC的ADCxOF寄存器，ADCxSTA[5]置1表示校准命令已完成。器件复位后，ADCxOF寄存器重载工厂校准值。
- 内部满量程校准(仅当增益 = 1时可用)。在此模式下，参照内部满量程基准电压对所有使能的ADC进行增益校准。增益校准分为两个阶段，耗时是失调校准的两倍。校准结果自动写入相应ADC的ADC增益寄存器，ADCxSTA[5]置1表示校准命令已完成。内部满量程校准仅对增益 = 1有效。器件复位后，ADC增益寄存器重载工厂校准值。
- 系统零点校准。在该模式下，参照ADC输入引脚处驱动的外部零电平电压，对使能的ADC通道进行零电平校准。所选的通道通常在外部予以短路。校准结果自动写入相应ADC的ADCxOF MMR，ADCxSTA[5]置1表示校准命令已完成。器件复位后，ADCxOF寄存器重载工厂校准值。
- 系统满量程校准。在此模式下，参照ADC输入引脚处驱动的外部满量程电压，对使能的ADC通道进行满量程校准。完成满量程校准序列之后，ADC增益寄存器更新，ADCxSTA[5]置1表示校准命令已完成。器件复位后，ADC增益寄存器重载工厂校准值。

注意，仅当ADC处于空闲模式(ADCxMDE[2:0] = 011)时，用户才能更改ADC增益和ADCOF寄存器。当基准电压源为内部基准电压源、外部基准电压源或AVDD电源时，ADC分别使用如下ADC增益寄存器：ADCxINTGN、ADCxEXTGN和ADCxVDDGN。仅ADCxINTGN加载工厂校准值。完成复位序列后，用户必须更新ADCxEXTGN和ADCxVDDGN寄存器。

ADC关断模式

特别是在单次转换之间，ADC应关断以降低系统总功耗。

在完全关断模式下，ADC和输入放大器完全掉电，功耗最低。在空闲模式下，ADC完全上电，但保持复位状态。校准操作完成后或单次转换之间，器件进入该模式。这种模式下的功耗比完全活动模式要低。

其它ADC支持电路

基准电压源

ADuCM360/ADuCM361支持4个基准电压源：内部基准电压源、外部基准电压源1、外部基准电压源2和AVDD。

内部基准电压源

内部基准电压源为1.2 V精密、低噪声、低漂移带隙基准电压源。更多信息参见ADuCM360/ADuCM361数据手册。

外部基准电压源1

外部基准电压源1连接到VREF+和VREF−引脚。另外还提供了内部缓冲外部基准电压的选项。正负外部基准电压输入各有一个缓冲器。如果负输入为地，则应旁路负输入缓冲器。

外部基准电压源2

外部基准电压源2可连接到AIN7/VBIAS0/IEXC/EXTREF2IN+和AIN8/EXTREF2IN−引脚。外部基准电压源2提供了缓冲EXTREF2IN+正输入的选项(详情参见表29中的ADCxCFG[1:0]描述)。

AVDD

可选择器件的AVDD作为基准电压源。

基准电压检测电路

有一个内部电路可用来检测外部基准电压引脚上的错误。该检测电路检查VREF+和VREF−引脚有无开路。它还会检查VREF+与VREF−之间的差分电压是否大于400 mV。若检测到错误状况，DETSTA[3]寄存器中的错误标志就会置1。

选择外部基准电压源时，若在VREF+或VREF−引脚上检测到开路，或者VREF+/VREF−上的差分输入基准电压低于400 mV，片内基准电压检测电路就会将错误标志置1。正常工作时，采样速率通过ADCxFLT寄存器配置。

EXTREF2IN±引脚没有基准电压检测电路可用。

快速响应输出/输入步进检测电路

两个ADC的Σ-Δ调制器的输出均馈入sinc3或sinc4数字滤波器。然而，为了加快响应时间，可选择sinc2滤波器与两个ADC的sinc3或sinc4滤波器并行工作，但sinc3和sinc4滤波器不能同时工作。该sinc2滤波器的更新速率可配置，速度更快(2 ms)，有效分辨率为14位。sinc2滤波器的输出馈入单独的结果寄存器STEPDAT。

sinc2滤波器也可用在步进检测模式下，用于检测输入信号的步进变化。这种模式下，sinc2滤波器的输出与之前的两个输出值相比较。如果最新值与之前两个值的差大于预定步进阈值，DETSTA寄存器中的检测到步进标志就会置位，并触发一个中断。或者，将sinc2滤波器的输出与sinc3或sinc4滤波器的最后输出相比较，也可以触发步进。如果差值大于所配置的阈值，检测到步进标志就会置位。

sinc2滤波器提供了两个可能的中断源：

- 使用DMA控制器和相关的ADC DMA中断；当存储器已存储预配置数量的sinc2样本时，此中断标志置1。
- 针对步进检测滤波器提供了一个单独的中断标志(DETSTA[1])。

DETCON寄存器配置sinc2滤波器和步进检测电路。

通过轮询DETSTA[0]，可读取sinc2滤波器的每个输出。

步进检测电路

如图7所示，步进检测滤波器对每个样本周期使用256 × ADC时钟。各滤波器周期结束时的输出为256个样本的平均值。步进检测硬件将各输出与前三个输出相比较。如果当前输出与之前任一输出的差值大于所选阈值，则说明检测到步进。通过设置DETCON[1:0]，可将sinc2滤波器的过采样率配置为256/512/768/1024 × ADC时钟，对应的采样周期分别为2 ms/4 ms/6 ms/ 8 ms。

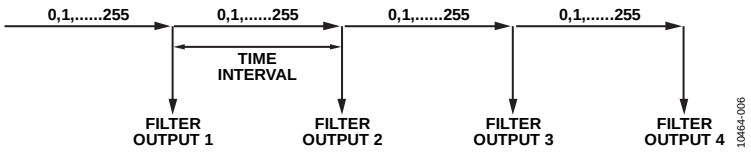


图7. ADuCM360/ADuCM361步进检测时序图

如图8所示，电平1和电平2的差值大于所选阈值：

- 若输出4 − 输出3 = 结果 < 阈值，则未检测到步进。
- 若输出4 − 输出2 = 结果 > 阈值，则检测到步进。
- 若输出4 − 输出1 = 结果 > 阈值，则检测到步进。

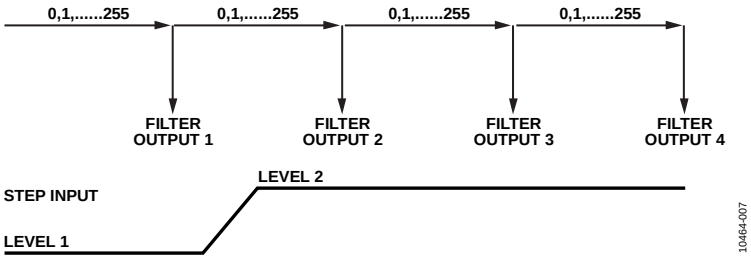


图8. 步进检测电路

步进检测电路有如下主要特性：

- 此步进检测滤波器默认以500 Hz的采样速率工作，因而输出速率为2 ms。利用滤波器控制寄存器，可将此值扩展为4 ms、6 ms或8 ms。
- 步进阈值可利用10位寄存器STEPTH编程。
- 最小阈值设置为满量程的0.05%。
- 无噪声输出分辨率为14位。
- 针对此滤波器输出提供了一个单独的步进检测中断和STEPDAT寄存器。
- 采样周期为2 ms时，步进状况与滤波器将步进标志置位之间的最小延迟时间为6 ms。此延迟时间按3倍的采样时间计算。
- 此滤波器可以处理的最慢压摆率为一个输出的采样周期。例如，若步进检测范围选择为0.25%或更大的步进，则当采样速率选择为500 Hz时，该步进的压摆率不得大于2 ms。

SINC2滤波器数据输出寄存器—STEPDAT

sinc2滤波器的输出通过STEPDAT寄存器提供。STEPDAT输出值会自动考虑增益。

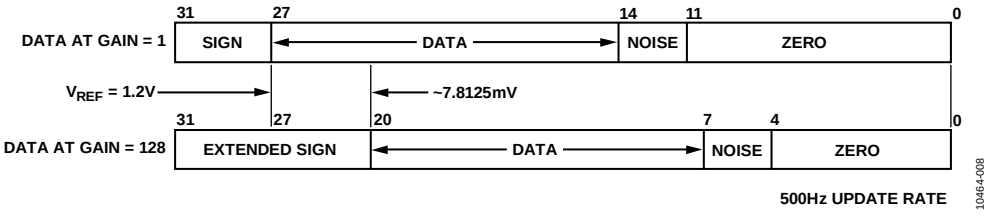


图9. ADC Sinc2数据寄存器500 Hz更新速率

ADC比较器和累加器

两个ADC都有数字比较器和累加器，用于对ADC结果进行后处理。

每个ADC转换结果都可以和ADCxPRO[3:2]中的预设阈值(ADCxTH)进行比较。如果ADC结果的绝对值(符号无关)大于预设比较器阈值，就会产生中断。作为比较器的扩展功能，用户代码可以配置一个阈值计数器(ADCxTHV)来监控转换结果高于或低于预设阈值的次数。同样，当阈值计数器达到预设值(ADCxRCR)时，就会产生ADC中断。

可配置32位累加器(ADCxACC)功能(ADCxPRO[5:4])，使ADC加上(或减去)多个采样结果。用户代码可直接读取累加值(ADCxACC)，不需要任何进一步软件处理。累加器也有一个比较器选项，如果累加器值超过用户定义的阈值(ADCxACC > ADCxATH)，可以产生一个独立中断。

诊断电流源

为了检测外部传感器的连接故障，ADuCM360/ADuCM361在两个ADC的选定模拟输入通道上整合了一个50 µA恒流(开路检测)源。通过ADCxCON[11:10]位可将其开启或关闭。 诊断电流源的精度为±10%。

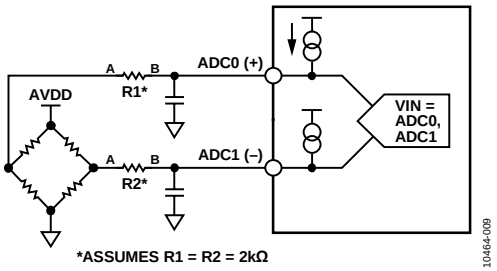


图10. 采用诊断电流源的示例电路

表15. 采用诊断电流源的示例场景

诊断测试		正常结果	故障结果	检测到的故障测量值
寄存器设置	描述			
ADCxCON[11:10] = 00	正常转换AINx/AINy，同时禁用诊断电流。			
ADCxCON[11:10] = 01	设置ADCxCON[11:10] = 01以在AINx上使能50 µA诊断电流源。转换AINx和AINy。	ADC变化幅度为ΔV = +50 µA × R1。例如，R1 = 2 kΩ时，变化幅度约为100 mV。	AINx和AINy之间短路。R1_a和R1_b之间短路。	无论PGA设置为何，ADC读数约为0 V。
	以单端模式转换AINx，同时使能诊断电流。	AINx上的期望电压。	AINx开路或R1开路。	ADC读数为正满量程，即使PGA设置值最低。
ADCxCON[11:10] = 11	设置ADCxCON[11:10] = 11以在AINx和AINy上使能50 µA诊断电流源。 转换AINx和AINy。	ADC读数变化幅度为ΔV = 50 µA × (R1 − R2)，即容差为10%时，变化幅度约为10 mV。	R1与R2不匹配。	ADC读数>期望值的10 mV。

偏置电压(V_{BIAS})发生器电路

ADuCM360/ADuCM361提供了一个用于产生偏置电压或AVDD_REG/2共模电压的电路。利用ADCxCFG[10:8]，可将AVDD_REG/2共模电压连接到两个模拟引脚AIN7和AIN11中的一个。因此，用户可以不将此电压内部连接到PGA输出，而是外部连接。

此功能对于热电偶设置特别有用。如果AIN7用作V_{BIAS}发生器，应将其连接到任何输入滤波的热电偶侧。

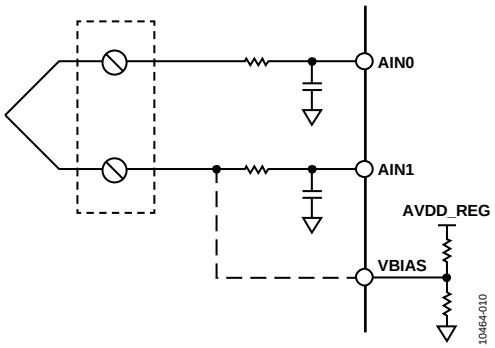


图11. 利用V_{BIAS}电压设置热电偶共模电压

注意，仅有一个模拟引脚可配置为V_{BIAS}输出。ADCxCFG[10:8]用于选择VBIAS输出引脚。

V_{BIAS}输出的上电时间取决于该引脚上放置的负载电容大小。为了缩短上电时间，可以利用V_{BIAS}升压模式让V_{BIAS}发生器电路向V_{BIAS}引脚输出更多电流，由此可以减少上电时间。ADCxCFG[13]将默认输出电流提高30倍。升压位使能时，V_{BIAS}发生器电路最多消耗150 µA的额外电流。当给外部10 nF负载电容充电时，升压模式禁用情况下的上电时间为30 ms，使能情况下为1 ms。

其它ADC细节

数字滤波器选项

ADuCM360/ADuCM361主ADC数字滤波器设置由ADC0FLT和ADC1FLT寄存器控制。sinc3或sinc4滤波器的抽取系数利用ADCxFLT[6:0]设置。有关抽取系数值的更多信息，请参见表16和表17。sinc3或sinc4滤波器(SF)字的工作范围取决于斩波功能是否使能。斩波禁用时，允许的最小SF字为1，最大SF字为127，导致ADC吞吐速率范围为3.5 Hz至3.906 kHz。斩波使能时，最高频率为1.262 kHz。有关如何根据ADCxFLT[6:0]值计算ADC采样频率的更多信息，请参阅“ADC电路存储器映射寄存器”部分中的ADCxFLT寄存器描述。

对于488 Hz或更大的采样频率，应设置ADCxFLT[12] = 1以使能sinc4滤波器选项。

对于488 Hz以下的采样频率，应设置ADCxFLT[12] = 0以使能sinc3滤波器选项。

ADCxFLT[7]提供了增加一个额外陷波频率的选项，其等于主陷波频率的1.2倍。此特性最常用于50 Hz/60 Hz同时抑制，其中50 Hz是主滤波器陷波频率。

还有一个sinc2滤波器选项，它可以与ADC0或ADC1上的sinc3或sinc4滤波器并行工作。sinc2滤波器可用作步进检测滤波器，或用于加快输出测量结果(但分辨率会降低)。

要将sinc2用作ADC0或ADC1的快速滤波器，请设置DETCON[7] = 1。

表16. ADC转换速率和建立时间

Sinc4使能 ADCxFLT[12]	斩波使能 ADCxFLT[15]	平均系数 (AF) ADCxFLT[11:8]	移动平均A DCxFLT[14]	f_{ADC}正常模式	$t_{SETTLING}$[1,2]
否(0)	否(0)	0	否(0)	$\frac{125{,}000}{[SF+1]\times 16}$	$\frac{3}{f_{ADC}}$
否(0)	否(0)	0	是(1)	$\frac{125{,}000}{[SF+1]\times 16}$	$\frac{4}{f_{ADC}}$
否(0)	否(0)	>0	否(0)	$\frac{125{,}000}{[SF+1]\times 16\times[3+AF]}$	$\frac{1}{f_{ADC}}$
否(0)	否(0)	>0	是(1)	$\frac{125{,}000}{[SF+1]\times 16\times[3+AF]}$	$\frac{2}{f_{ADC}}$
否(0)	是(1)	任意值	任意值	$\frac{125{,}000}{[SF+1]\times 16\times[3+AF]+3}$	$\frac{2}{f_{ADC}}$
是(1)	否(0)	0	否(0)	$\frac{125{,}000}{[SF+1]\times 16}$	$\frac{4}{f_{ADC}}$
是(1)	否(0)	0	是(1)	$\frac{125{,}000}{[SF+1]\times 16}$	$\frac{5}{f_{ADC}}$
是(1)	是(1)	0	任意值	$\frac{125{,}000}{([SF+1]\times 16\times 4)+3}$	$\frac{2}{f_{ADC}}$

[1] 在第一个ADC转换结果可用之前，每个ADC需要约256 µs的额外时间。

[2] 更改配置会重启ADC，并需要经过滤波器的建立时间后才能获得有效数据。

表17. SF和AF的允许组合

	AF					
SF	**0**	**1:2**	**3:7**	**8:9**	**10:14**	**15**
0:123	是	是	是	是	是	是
124	是	是	是	是	是	仅当陷波2开启时
125	是	是	是	仅当陷波2开启时	否	否
126	是	仅当陷波2开启时	否	否	否	否
127	是	否	否	否	否	否

ADC斩波

ADuCM360/ADuCM361的ADC实现了一种斩波方案，其中ADC会不断地反转其输出。因此，sinc3或sinc4滤波器的抽取数字输出值有一个与之相关的正负失调项。这导致ADC有一个最终求和级，它将滤波器的各输出值与滤波器前一输出值相加并求平均值，然后将此新值送至ADC数据MMR。这种斩波方案使得直流失调和失调漂移规格非常出色，对需要抑制漂移和噪声的应用极为有利。

噪声/分辨率表

两个ADC的噪声和分辨率数值参见ADuCM360/ADuCM361数据手册。

同步采样

ADuCM360有两个相同的ADC，它们可以配置为同步采样。ADC1的滤波器设置(位于ADC1FLT中)会自动应用于ADC0。

要使能同步采样，请执行如下配置步骤：

- 配置ADC0MDE和ADC1MDE。
- 配置ADC1FLT。
- 根据需要配置ADC0CON和ADC1CON，但二者的位19必须为0。
- 设置ADCCFG[15] = 1。

然后，两个ADC便会同时开始转换，并以相同速率输出数据。

内部通道

ADuCM360/ADuCM361有四个内部通道选项：

- 温度传感器
- DAC监控器
- AVDD ÷ 4
- IOVDD ÷ 4

温度传感器设置

ADuCM360/ADuCM361提供从片内带隙基准电压源输出并与ADC1的绝对温度成正比的电压。此电压输出也可以通过ADC1多路复用器的前端连接到ADC模拟输入通道，这样就可以很方便地形成一个内部温度传感器通道，用于测量芯片温度。内部温度传感器并非设计用作绝对环境温度计算器，而是用作ADuCM360/ADuCM361芯片温度的近似指示器。

ADC温度传感器转换与标准ADC电压不同。ADC性能规格并不适用于温度传感器。

温度传感器设置如下：

- 使能ADC1的温度传感器；设置ADC1CON[9:0] = 0xBC211。
- 使能PGA。 建议使用增益 = 2。
- ADCMDE = 0x11.

计算芯片温度的公式：

$$T - T_{REF} = (V_{ADC} - V_{TREF}) \times K$$

其中：

T为温度结果。

T_{REF}为25°C。

V_{ADC}为从两个连续转换结果得出的ADC转换结果平均值。

V_{TREF}为82.1 mV，对应于ADuCM360/ADuCM361数据手册所述的T_{REF} = 25°C时的值。

K为ADC在温度传感器模式下的增益，由特性数据确定，K = 4 °C/mV。

这对应于ADuCM360/ADuCM361数据手册所述的1/V TC。

使用ADuCM360/ADuCM361数据手册中未经任何校准的默认值，此公式变为：

$$T - 25°C = (V_{ADC} - 82.1) \times 4$$

其中：

V_{ADC}单位为毫伏。

关于最新数值，请检查ADuCM360/ADuCM361数据手册的最新版本。

要提高精度，可在受控温度值执行单点校准。对于未执行校准时的计算，(T_{REF}, V_{TREF}) = (25°C, 82.1 mV)。单点校准的思路是使用其它已知的(T_{REF}, V_{TREF})值来取代各器件的通用值(25°C, 82.1 mV)。有些用户可能无法取得此类数值对。

DAC监控器设置

DAC监控器设置如下：

- 增益 = 1
- 0x1 pADI_ADC1->MDE = 0x1
- pADI_ADC1-> ON = 0xBC18F, AIN+ = DAC, AIN− = AGND

AVDD/4设置

AVDD/4设置如下：

- 选择DAC作为AIN−，并将DAC输出设置为600 mV
- pADI_DAC->DACCON = 0x410
- pADI_ADC1->DAT = 0x8000000
- pADI_ADC1->CON = 0xBC1AC

- pADI_ADC1->MDE = 0x1
- AVDD电源电压 = (ADC电压 + DAC) × 4

注意，为使该通道正常工作，该输出必须相对于DAC输出测量，而不是相对于AGND测量。

IOVDD/4设置

IOVDD/4设置如下：

- 选择DAC作为AIN−，并将DAC输出设置为600 mV
- pADI_DAC->DACCON = 0x410
- pADI_ADC1->DAT = 0x8000000
- pADI_ADC1->CON = 0xBC1CC
- pADI_ADC1->MDE = 0x1
- AVDD电源电压 = (ADC电压 + DAC) × 4

注意，为使该通道正常工作，该输出必须相对于DAC输出测量，而不是相对于AGND测量。

ADC DMA细节

ADC0、ADC1和sinc2输出可使用直接存储器访问功能。此功能有助于降低应对多个ADC转换中断的处理开销。

ADC有两种不同的DMA功能：

- 从存储器到ADC控制寄存器的DMA写操作。注意，此功能仅适用于ADC0和ADC1，不适用于sinc2输出。
- 从ADC0DAT、ADC1DAT或STEPDAT寄存器到存储器的直接DMA写操作。

关于设置DMA控制器的步骤列表和示例代码，参见“示例代码：ADC0结果直接移动到存储器的DMA配置”部分和“示例代码：利用DMA设置ADC0控制寄存器的DMA配置”部分。

示例代码： ADC0结果直接移动到存储器的DMA配置

```
pADI_ADC0->MDE = 0x00000003;                                    // Allow writing to the calibration
registers by placing ADC in idle mode
pADI_ADC0->CON = 0x80001;                                       // Enable ADC0, AIN0/AIN1 input
channels, Internal reference
pADI_ADC0-> ADCCFG = 0x00;                                      // Enable input buffers, disable
ground switch
pADI_ADC0->FLT = 0x7d;                                          // 50Hz sampling rate, chop off
Dma_Init();                                                     // Init DMA structures
pADI_ADCDMA-> ADCDMACON = 0x0000003;                            // Enable ADC0 DMA channel for ADC
results to memory
ADC0DMAREAD(uxADC0Data, 32);                                    // Setup DMA control registers
NVIC_EnableIRQ(DMA_ADC0_IRQn);                                  // Enable DMA ADC0 Interrupt
pADI_DMA->DMAENSET = 0x200;                                     // Select ADC0 channel in DMA
controller
pADI_ADC0->MDE = 0x21;                                          // G = 4, continuous conversions


void ADC0DMAREAD(unsigned long *pucRX_DMA, unsigned int iNumRX)
{
   DmaDesc Desc;                                                // Common configuration of all the
descriptors used here
    Desc.ctrlCfg.bits.cycle_ctrl        = cyclectrl_basic;
    desc.ctrlcfg.bits.next_useburst     = 0x0;
    desc.ctrlcfg.bits.r_power           = 0;
    Desc.ctrlCfg.Bits.src_prot_ctrl     = 0x0;
```

```
    Desc.ctrlCfg.Bits.dst_prot_ctrl     = 0x0;
    Desc.ctrlCfg.Bits.src_size          = SRCSIZE_WORD;
    Desc.ctrlCfg.Bits.dst_size          = DSTSIZE_WORD;

                                                        // RX Primary Descriptor
     Desc.srcEndPtr                     = (unsigned int)&ADC0DAT;
    Desc.destEndPtr                     = (unsigned int)(pucRX_DMA + iNumRX - 0x1); //
    Desc.ctrlCfg.Bits.n_minus_1         = iNumRX - 0x1;
    Desc.ctrlCfg.Bits.src_inc           = INC_NO;
    Desc.ctrlCfg.Bits.dst_inc           = INC_WORD;
    *Dma_GetDescriptor(DMA_REQ_ADC0,FALSE) = Desc;

}
void Dma_Init(void)
{
                                                        // Clear all the DMA descriptors
(individual blocks will update their descriptors
   memset(dmaChanDesc,0x0,sizeof(dmaChanDesc));

                                                        // Set up the DMA base pointer.
pADI_DMA->DMAPDBPTR = (unsigned int) &dmaChanDesc;

                                                        // Enable the  µDMA (Write only reg)
pADI_DMA->DMACFG = 0x1;
}
DmaDesc * Dma_GetDescriptor(unsigned int iChan,boolean bAlternate)
{
    if (iChan < DMACHAN_NUM)
        {
        if (bAlternate)
            return &(dmaChanDesc[iChan + 12]); //DMA_ALT_OFFSET]);
        else
            return &(dmaChanDesc[iChan ]);
        }
    else
        return 0x0;
}
void DMA_ADC0_Int_Handler()
{
                      unsigned char n = 0;
                      NVIC_DisableIRQ(DMA_ADC0_IRQn);      // Clear Interupt source
   ucDMA_ADC0_COMPLETE = 1;
}
```

示例代码：利用DMA设置ADC0控制寄存器的DMA配置

```
unsigned long uxADC0SETUP[] = {0x1,                     // ADC0MSKI
0x80001,                                                // ADC0CON
0x0000,                                                 // ADC0OF
```

```
0x5555,                                                 // ADC0INTGN
0x5555,                                                 // ADC0EXTGN
0x5555,                                                 // ADC0VDDGN
0x00,                                                   // ADCxCFG
0x3,                                                    // ADC0FLT
0x113};                                                 // ADC0MDE    - G = 4 Not sure what
you are asking here (Just wanted to make sure that the placement was okay. Also can we add a
space before and after the equal sign?) OK


ucDMA_ADC0_COMPLETE = 0;                                // Flag to indicate DMA complete -
set in IRQ handler
pADI_ADC0->MDE;                                         // Allow writing to the calibration
registers by placing ADC in idle mode
Dma_Init();                                             // Init DMA structures
pADI_ADCDMA-> ADCDMACON = 0x0000002;                            // Enable ADC0 DMA channel
for moving array contents in memory into ADC0 control registers
ADC0DMAWRITE(uxADC0SETUP, 9);                                   // Setup DMA control
registers
NVIC_EnableIRQ(DMA_ADC0_IRQn);                                  // Enable DMA ADC0 Interrupt
pADI_DMA->DMAENSET = 0x200;                                     // Select ADC0 channel in DMA
controller
pADI_TM1->LD = 0x28;                                            // On pre-release silicon,
timer overflow required to trigger DMA write
pADI_TM1->CON = 0x18;


void ADC0DMAWRITE(unsigned long *pucTX_DMA, unsigned int iNumRX)
{
   DmaDesc Desc;
                                                                // Common configuration of
all the descriptors used here
    Desc.ctrlCfg.bits.cycle_ctrl        = cyclectrl_basic;
    desc.ctrlcfg.bits.next_useburst     = 0x0;
    desc.ctrlcfg.bits.r_power           = 0;
    Desc.ctrlCfg.Bits.src_prot_ctrl     = 0x0;
    Desc.ctrlCfg.Bits.dst_prot_ctrl     = 0x0;
    Desc.ctrlCfg.Bits.src_size          = SRCSIZE_WORD;
    Desc.ctrlCfg.Bits.dst_size          = DSTSIZE_WORD;
                                                                // TX Primary Descriptor
Desc.srcEndPtr                          = (unsigned long)(pucTX_DMA + iNumRX - 0x1);
Desc.destEndPtr                         =  (unsigned long)&ADC0MDE;
Desc.ctrlCfg.Bits.n_minus_1             = iNumRX - 0x1;
Desc.ctrlCfg.Bits.src_inc               = INC_WORD;
Desc.ctrlCfg.Bits.dst_inc               = INC_WORD;
*Dma_GetDescriptor(DMA_REQ_ADC0,FALSE) = Desc;
}
```

更多信息请参见本文档的“DMA控制器”部分。

ADC数据寄存器

ADCxDAT寄存器输出格式便于用户计算ADC测得的电压。

双极性模式下，计算公式为：

$(V_{REF})/2^{28}) \times ADCxDAT$ (有符号结果)

单极性模式下，计算公式为：

$(V_{REF})/2^{28}) \times ADCxDAT$ (无符号结果，仅正值)

增益值由ADC硬件自动调整。

每个ADC提供一个24位转换结果。根据ADCCODE位(ADCxCON[18])的设置，数据为二进制补码或单极性值。读取ADCxDAT MMR会清除任何有效的RDY标志。

虽然ADC标称值为24位，但在3.53 Hz更新速率和1倍增益时，会产生约1.05 μV 的均方根噪声。因此，有用的信号范围是±1.2 V至1.05 μV，相当于约2,285,000:1或21位。结果置于ADCxDAT的位[27:7]。

当增益为128且更新速率为3.53 Hz时，均方根噪声为52 nV。因此，有用的信号范围是约7.8125 mV至42 nV，相当于约300,000:1或18.5位。结果置于ADCxDAT的位[20:3]。

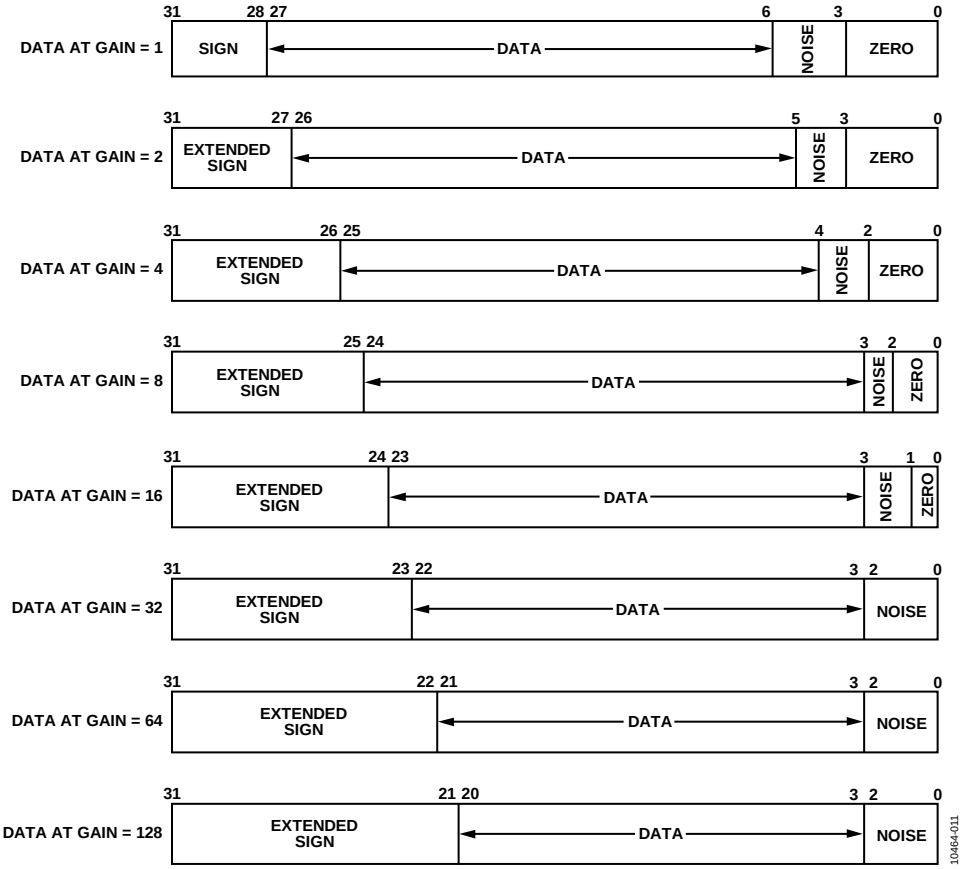


图12. ADC数据寄存器3.53 Hz更新速率

更新速率更快时，例如50 Hz，噪声系数更差，分辨率几乎要低2位(参见图13)。

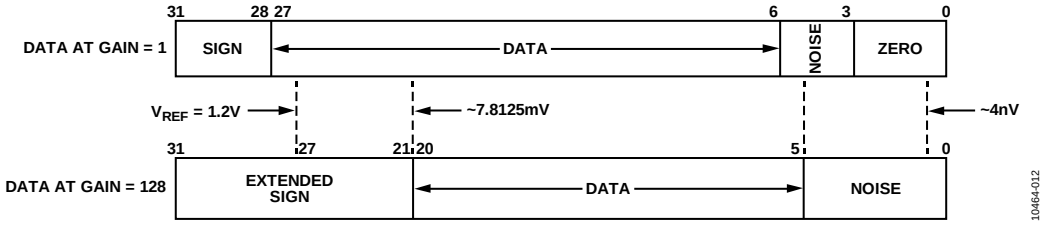


图13. ADC数据寄存器50 Hz更新速率

如果改善噪声性能或应用额外的滤波，底部仍有很大的空间。

如果相关RDY位置1，则ADC不会将新数据写入ADCxDAT MMR。然而，若ADuCM360/ADuCM361处于关断模式且DMA使能，情况就会不同，ADCxDAT总是包含最新的ADC数据。

应注意如下事项：

- 错误位ADCxSTA[4]可用于检测大致的(30%以上)ADC上溢或下溢状况。
- 当使能PGA时，关于1 V输出电压限值的详细信息，请参见“PGA增益配置”部分。
- 若RDY为低电平，无法保证读取ADCxDAT MMR时它们处于稳定状态。
- 如果ADCxRCR计数器有效，则数据寄存器要等到RDY置1后才更新(即当ADCxRCR计数器达到设定的限值时)。如果处理器关闭则不是如此，这种情况下，当处理器唤醒时，ADCxDAT MMR包含最新的ADC结果。

激励电流源

ADuCM360/ADuCM361内置两个匹配的软件可配置电流源。这些激励电流由AVDD提供，可单独配置，提供10 μA至1 mA电流输出。电流输出值通过IEXCDAT配置。这些电流源可以用来激励外部阻性电桥或RTD传感器。IEXCCON MMR控制激励电流源。IEXCCON[2:0]配置激励电流源0的输出引脚。类似地，IEXCCON[5:3]配置激励电流源1的输出引脚。

还可以配置这两个激励电流源向单个输出引脚输出电流，使输出电流加倍。这样，单个激励引脚可获得最多2 mA的输出电流。例如，设置IEXCCON = 0x64且IEXCDAT = 0x3E，便可在AIN4引脚上产生2 mA输出电流。

激励电流源设计默认使用内部基准电阻。使用内部基准电阻时，典型输出电流漂移性能为200 ppm/°C。使用150 kΩ低漂移外部基准电阻并设置IEXCCON[6] = 0，可降低输出电流值的漂移。

ADC电路存储器映射寄存器

表18. ADC0寄存器汇总(仅ADuCM360)

地址	名称	描述	访问类型	默认值
0x40030000	ADC0STA	ADC状态寄存器	R	0x00
0x40030004	ADC0MSKI	中断控制寄存器	RW	0x00
0x40030008	ADC0CON	主控制寄存器	RW	0x3038C
0x4003000C	ADC0OF	失调校准寄存器	RW	生产测试中设置
0x40030010	ADC0INTGN	使用内部基准电压源时的增益校准寄存器	RW	生产测试中设置
0x40030014	ADC0EXTGN	使用外部基准电压源时的增益校准寄存器	RW	生产测试中设置
0x40030018	ADC0VDDGN	使用AVDD作为ADC基准电压源时的增益校准寄存器	RW	0x5555
0x4003001C	ADC0CFG	V_{BIAS}电压发生器、接地开关和外部基准电压缓冲器的控制寄存器	RW	0x0000
0x40030020	ADC0FLT	滤波器配置寄存器	RW	0x007D
0x40030024	ADC0MDE	模式控制寄存器	RW	0x0003
0x40030028	ADC0RCR	产生ADC中断前的ADC0转换次数	RW	001
0x4003002C	ADC0RCV	当前ADC0转换结果数量	R	0x0000
0x40030030	ADC0TH	设置阈值	RW	0x0000
0x40030034	ADC0THC	必须发生的累加ADC0转换结果读数超过ADC0TH的次数	RW	0x01
0x40030038	ADC0THV	8位阈值超过计数器寄存器	R	0x00
0x4003003C	ADC0ACC	32位累加器寄存器	R	0x00000000
0x40030040	ADC0ATH	保存累加器比较器的阈值	RW	0x00000000
0x40030044	ADC0PRO	ADC0结果后处理的配置寄存器	RW	0x00
0x40030048	ADC0DAT	转换结果寄存器	R	0x00000000

表19. ADC1寄存器汇总(ADuCM360和ADuCM361)

地址	名称	描述	访问类型	默认值
0x40030080	ADC1STA	ADC状态寄存器	R	0x00
0x40030084	ADC1MSKI	中断控制寄存器	RW	0x00
0x40030088	ADC1CON	主控制寄存器	RW	0x303FF
0x4003008C	ADC1OF	失调校准寄存器	RW	生产测试中设置
0x40030090	ADC1INTGN	使用内部基准电压源时的增益校准寄存器	RW	生产测试中设置
0x40030094	ADC1EXTGN	使用外部基准电压源时的增益校准寄存器	RW	生产测试中设置
0x40030098	ADC1VDDGN	使用AVDD作为ADC基准电压源时的增益校准寄存器	RW	0x5555
0x4003009C	ADC1CFG	V_{BIAS}电压发生器、接地开关和外部基准电压缓冲器的控制寄存器	RW	0x0000
0x400300A0	ADC1FLT	滤波器配置寄存器	RW	0x007D
0x400300A4	ADC1MDE	模式控制寄存器	RW	0x0003
0x400300A8	ADC1RCR	产生ADC中断前的ADC1转换次数	RW	0x0001
0x400300AC	ADC1RCV	当前ADC1转换结果数量	R	0x0000
0x400300B0	ADC1TH	设置阈值	RW	0x0000
0x400300B4	ADC1THC	必须发生的累加ADC1转换结果读数超过ADC1TH的次数	RW	0x01
0x400300B8	ADC1THV	8位阈值超过计数器寄存器	R	0x00
0x400300BC	ADC1ACC	32位累加器寄存器	R	0x00000000
0x400300C0	ADC1ATH	保存累加器比较器的阈值	RW	0x00000000
0x400300C4	ADC1PRO	ADC1结果后处理的配置寄存器	RW	0x00
0x400300C8	ADC1DAT	转换结果寄存器	R	0x00000000

表20. ADCSTEP寄存器汇总

地址	名称	描述	访问类型	默认值
0x400300E0	DETCON	基准电压检测和步进检测滤波器的控制寄存器	RW	0x0000
0x400300E4	DETSTA	用于检测的状态寄存器	R	0x00
0x400300E8	STEPTH	步进检测滤波器的阈值	RW	0x0000
0x400300EC	STEPDAT	提供步进检测滤波器输出的粗略数据	R	0x00000000
0x400300F8	ADCDMACON	ADC DMA模式配置寄存器	RW	0x0000

表21. 其它ADC寄存器汇总

地址	名称	描述	访问类型	默认值
0x40008840	REFCTRL	内部基准电压控制寄存器	RW	0x0000
0x40008850	IEXCCON	控制片内激励电流源	RW	0xC0
0x40008854	IEXCDAT	设置两个激励电流源的输出电流	RW	0x06

ADC寄存器

ADC状态寄存器

地址：0x40030000；复位：0x00；名称：ADC0STA(仅ADuCM360)

地址：0x40030080；复位：0x00；名称：ADC1STA(ADuCM360和ADuCM361)

表22. ADCxSTA的位功能描述

位	名称	描述
7:6	保留	
5	ADCxCAL	ADC校准状态位。 ADC校准完成时置1。 读取ADCxSTA后清0。
4	ADCxERR	ADC转换错误状态位。 发生欠范围或超范围错误时置1。 读取ADCxSTA后清0。
3	ADCxATHEX	ADC累加器比较器阈值状态位。仅当通过ADCxPRO寄存器使能ADC累加器时有效。 当ADCxACC中的ADC累加器值超过ADC比较器阈值寄存器ADCxATH中的阈值时，该位置1。 当ADCxACC的值未超过ADCxATH中的值时，该位清0。
2	ADCxTHEX	ADC比较器阈值。仅当通过ADCxPRO寄存器使能ADC比较器时有效。 若ADC转换结果绝对值超过ADCxTH寄存器中写入的值时，该位置1。 若使用ADC阈值计数器(ADCxRCR)，该位仅在指定的ADC转换次数等于 ADCxTHV寄存器值时置1。 否则，该位清0。重新配置ADC时，硬件将该位清0。更多信息参见示例代码。
1	ADCxOVR	ADC超范围位。 通过ADCxPRO寄存器使能超范围检测功能时置1。若ADC输入大致超范围(约30%以上)， 硬件将此位置1。此位每500 µs更新一次。 当ADCxPRO[1]清0或ADCxMDE改变ADC增益时，该位由软件清0。
0	ADCxRDY	有效转换结果。 当ADC通道使能时，一旦将有效转换结果写入ADCDAT寄存器，硬件便会将该位置1。 若使用ADC计数器寄存器(ADCxRCR)，该位仅在指定的ADC转换次数等于ADCxRCR 寄存器值时置1。 读取ADCDAT后清0。

中断控制寄存器

地址：0x40030004；复位：0x00；名称：ADC0MSKI(仅ADuCM360)

地址：0x40030084；复位：0x00；名称：ADC1MSKI(ADuCM360和ADuCM361)

表23. ADCxMSKI的位功能描述

位	名称	描述
7:4	保留	
3	ATHEX	ADC累加器比较器阈值状态位屏蔽。 0: 禁用中断(默认)。 1: 当ADCxSTA寄存器中的ADCxATHEX位置1时，使能中断。
2	THEX	ADC比较器阈值屏蔽。 0: 禁用中断(默认)。 1: 当ADCxSTA寄存器中的ADCxTHEX位置1时，使能中断。
1	OVR	ADC超范围位屏蔽。 0: 禁用中断(默认)。 1: 当ADCxSTA寄存器中的ADCxOVR位置1时，使能中断。
0	RDY	有效转换结果屏蔽。 0: 禁用中断(默认)。 1: 当ADCxSTA寄存器中的ADCxRDY位置1时，使能中断。

控制寄存器

地址：0x40030008；复位：0x3038C；名称：ADC0CON(仅ADuCM360)

地址：0x40030088；复位：0x303FF；名称：ADC1CON(ADuCM360和ADuCM361)

表24. ADCxCON的位功能描述

位	名称	描述
31:20	保留	
19	ADCEN	使能位。 0: 关断ADC，并将ADCxRDY位清0。 1: 使能ADC。
18	ADCCODE	ADC输出编码。此位也会影响累加器(ADCxACC)和ADCxTH的编码。 0: 二进制补码(双极性)，注意ADCDAT >> 1。 1: 单极性。
17	BUFPOWN	负缓冲器电源。 0: 使能负缓冲器。 1: 关断负缓冲器。
16	BUFPOWP	正缓冲器电源。 0: 使能正缓冲器。 1: 关断正缓冲器。
15	BUFBYPP	正缓冲器旁路。 0: 不旁路正缓冲器。 1:旁路正缓冲器。
14	BUFBYPN	负缓冲器旁路。 0: 不旁路负缓冲器。 1: 旁路负缓冲器。
13:12	ADCREF	基准电压源选择。 00: INTREF-AGND。 01: EXTREF。外部缓冲器模式在ADCxCFG寄存器中设定。 10: EXTREF2IN(仅对ADC1有效)。 EXTREF2IN+缓冲器通过ADCxCFG控制。 11: AVDD-AGND。
11:10	ADCDIAG	诊断电流位。 00: 电流源关闭。 01: 在选定正输入端(例如AIN0)使能电流(50 µA)。 10: 在选定负输入端(例如AIN1)使能电流(50 µA)。 11: 在选定输入端(例如AIN0和AIN1)使能电流(50 µA)。
9:5	ADCCP	正输入通道选择。 00000: AIN0。 00001: AIN1。 ... 01011: AIN11。 01111: AGND。 11100: 所有通道关闭(默认)。 01100: DAC输出(仅对ADC1有效)。 01101: AVDD/4(仅对ADC1有效)。 01110: IOVDD/4(仅对ADC1有效)。 10000: 温度传感器(仅对ADC1有效)。 11111: 所有通道关闭(默认)(仅对ADC1有效)。
4:0	ADCCN	负输入通道选择。 00000: AIN0。 00001: AIN1。 ... ... 01011: AIN11。 01111: AGND。 01100: DAC输出。 11111: 保留。 10001: 温度传感器(仅对ADC1有效)。 对于所有其它内部通道测量，应选择AGND。

失调校准寄存器

失调寄存器是有符号二进制补码形式。代码0000表示减去0失调。如果ADCxCON中的ADCCODE位置1，此代码并不能将其变为单极性。

只有在ADC空闲时，用户才能写入失调寄存器。但在写入ADCxMDE和写入失调/增益寄存器之间，需要6 μs的软件延迟时间。

ADCxOF寄存器的有效范围是0xA000到0x5FFF。使用此范围之外的值会产生欠范围/超范围错误。根据增益设置，先补偿失调，再移出ADC结果。因此，ADCxOF寄存器的LSB权重与增益设置无关(LSB权重 = $2 \times V_{REF}/49150 = 48.8\ \mu V$)，更改增益设置时必须调整失调。

地址：0x4003000C；复位：工厂定义值；名称：ADC0OF(仅ADuCM360)

地址：0x4003008C；复位：工厂定义值；名称：ADC1OF(ADuCM360和ADuCM361)

表25. ADCxOF的位功能描述

位	名称	描述
15:0	VALUE	ADC失调校准系数。有效范围： 0xA000至0x5FFF。

增益校准寄存器

有三类增益寄存器：

- 使用内部基准电压源配置时的增益寄存器
- 使用外部基准电压源配置时的增益寄存器
- 使用AVDD基准电压源配置时的增益寄存器

根据基准电压源配置，使用相应的ADCGN。

增益寄存器为无符号数，代表一个比例因子。最大值为0xFFFF，标称值为0x5555。

上电或复位时，增益寄存器载入工厂校准值。若由用户通过ADCxMDE寄存器启动增益校准，则会覆写增益寄存器。

ADC内部满量程校准只能在增益为1时进行。仅当ADC处于空闲或关断模式，或ADCEN = 0 (ADCxCON[19])时，用户才能写入增益寄存器。但在写入ADCxMDE和写入失调/增益寄存器之间，需要一定的软件延迟时间。建议延迟6 μs。

工厂校准仅针对使用内部基准电压源的情况执行，存储在ADCxINTGN中。使用该器件之前，用户须填充ADCxEXTGN和ADCxVDDGN。

使用内部基准电压源时的增益校准寄存器

地址：0x40030010；默认值：工厂定义值；名称：ADC0INTGN(仅ADuCM360)

地址：0x40030090；默认值：工厂定义值；名称：ADC1INTGN(ADuCM360和ADuCM361)

表26. ADCxINTGN的位功能描述

位	名称	描述
15:0	VALUE	ADC 16位增益校准系数。选择内部基准电压源作为基准源时使用。

使用外部基准电压源时的增益校准寄存器

地址：0x40030014；默认值：工厂定义值；名称：ADC0EXTGN(仅ADuCM360)

地址：0x40030094；默认值：工厂定义值；名称：ADC1EXTGN(ADuCM360和ADuCM361)

表27. ADCxEXTGN的位功能描述

位	名称	描述
15:0	VALUE	ADC 16位增益校准系数。选择外部基准电压源作为基准源时使用。

使用AVDD作为ADC基准电压源时的增益校准寄存器

地址：0x40030018；复位：0x5555；名称：ADC0VDDGN(仅ADuCM360)

地址：0x40030098；复位：0x5555；名称：ADC1VDDGN(ADuCM360和ADuCM361)

表28. ADCxVDDGN的位功能描述

位	名称	描述
15:0	VALUE	ADC 16位增益校准系数。选择AVDD作为基准电压源时使用。

配置寄存器

地址： 0x4003001C；复位： 0x0000；名称： ADC0CFG(仅ADuCM360)

地址： 0x4003009C；复位： 0x0000；名称： ADC1CFG(ADuCM360和ADuCM361)

ADC0CFG和ADC1CFG是同一物理寄存器，由两个ADC共用，但对于ADC DMA写操作，该寄存器有两个地址。

表29. ADCxCFG的位功能描述

位	名称	描述
15	SIMU (仅对ADuCM360有效)。	0: 使能同步采样。ADC1FLT设置用于两个ADC(仅对ADuCM360有效)。 1: ADC独立采样。(仅对ADuCM360有效)。 ADuCM361保留。
14	保留	
13	BOOST30	将V_{BIAS}电流源能力提高30倍。注意，这样做会导致功耗增加。 0: 禁用电流升高。 1: 使能电流升高。
12:11	保留	
10:8	PINSEL	使能V_{BIAS}发生器，并将V_{BIAS}发送至选定的AIN引脚。 000: 禁用。 001: 保留。 010: 保留。 011: 保留。 100: 送至AIN7。 101: 保留。 110: 送至AIN11。 111: 禁用。
7	GNDSWON	GND_SW。 0: 模拟地开关与外部引脚断开。 1: 将外部GND_SW引脚连接到内部模拟地参考点。在程序控制下，该位可用于将外部电路和元件接地或断开，从而在不使用外部电路或元件时，将直流功耗降到最低。
6	GNDSWRESEN	20 kΩ电阻与GND_SW串联。 0: 禁用20 kΩ电阻与接地开关串联。 1: 使能20 kΩ电阻与接地开关串联。
5:2	保留	
1:0	EXTBUF	使能外部缓冲器。 00: 两个外部基准电压缓冲器均关断并旁路。 01: 使能外部缓冲器。选择VREF+和VREF−输入作为缓冲器输入。 10: 使能外部缓冲器。选择VREF+和EXTREF2IN+输入作为缓冲器输入。 11: 使能VREF+上的外部缓冲器。VREF−上的外部缓冲器关断并旁路。

滤波器配置寄存器

地址：0x40030020；复位：0x007D；名称：ADC0FLT(仅ADuCM360)

地址：0x400300A0；复位：0x007D；名称：ADC1FLT(ADuCM360和ADuCM361)

表30. ADCxFLT的位功能描述

位	名称	描述
15	CHOP	使能系统斩波。使能系统斩波可将失调误差和漂移降至非常低的水平。 0: 禁用系统斩波。 1: 使能系统斩波。
14	RAVG2	使能以2为基数的移动平均。使能以2为基数的移动平均功能可降低ADC噪声。 斩波使能时，此特性自动激活；斩波无效时，此特性是可选特性。在斩波 关闭的情况下，它不会降低输出速率，但建立时间会延长一个转换周期。 0: 禁用以2为基数的移动平均功能。 1: 使能以2为基数的移动平均功能。
13	保留	
12	SINC4EN	使能sinc4滤波器而禁用sinc3滤波器。注意，当输出速率小于488 Hz时， 此位不得置1。当输出速率大于或等于488 Hz时，此位置1会使能sinc4 滤波器。当输出速率大于488 Hz时，推荐使用sinc4滤波器；这种情况下， sinc3滤波器不足以滤除量化噪声。 0: 禁用sinc4滤波器而使能sinc3滤波器。 1: 使能sinc4滤波器而禁用sinc3滤波器。
11:8	AF	平均系数，#Avgs = AF + 1，其中AF实现了一个可编程一阶sinc后置滤波器。 此附加平均系数(AF)会进一步降低ADC输出速率。 SF和AF的最大允许值存在限制(参见表16和表17)。
7	NOTCH2	Sinc3/Sinc 4修正。 0: 禁用滤波器陷波。 1: 用户将该位置1，可修正标准Sinc3/sinc 4频率响应，以便将滤波器阻带抑制增加约5 dB。 通过插入第二个陷波(NOTCH2) $f_{NOTCH2} = 1.2 \times f_{NOTCH}$即可实现，其中$f_{NOTCH}$是响应中第一个陷波 所在位置。第二个陷波一般为主sinc陷波的(6/5)倍，以便产生50 Hz和60 Hz两个陷波。
6:0	SF	Sinc3/Sinc4滤波器抽取系数。SF控制sinc3/sinc4滤波器的过采样率/抽取系数。 这些位的默认值为0x7D。表16给出了sinc3/sinc4滤波器的转换速率，表17列出 了SF和AF的允许组合。

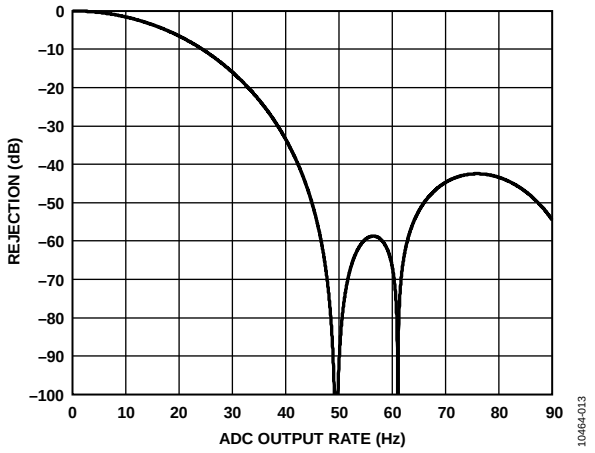


图14. 数字滤波器频率响应： 陷波2滤波器开启

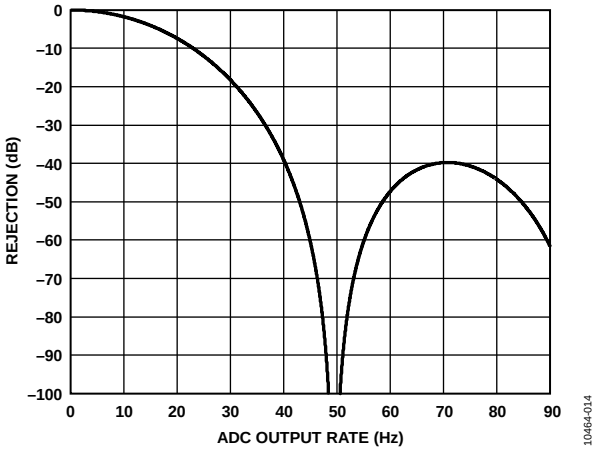


图15. 数字滤波器频率响应： 陷波2滤波器关闭

模式控制寄存器

写入ADCxMDE会复位对应的ADC，包括RDY位和其它ADCxSTA标志。取决于ADC所用的时钟，写入该寄存器到设置对ADC逻辑生效可能存在一个延迟。因此，复位RDY位的更可靠方法是读取ADCxDAT。

每次校准后，ADC回到空闲模式，ADCxSTA中的RDY和CAL位置1。

仅当ADC处于空闲或关断模式，或ADCEN = 0 (ADCxCON[19])时，用户才能写入失调和增益寄存器。但在写入ADCMDE和写入失调/增益寄存器之间，需要一定的软件延迟时间。

地址：0x40030024；复位：0x0003；名称：ADC0MDE(仅ADuCM360)

地址：0x400300A4；复位：0x0003；名称：ADC1MDE(ADuCM360和ADuCM361)

表31. ADCxMDE的位功能描述

位	名称	描述	
15:8	保留		
7:4	ADCxPGA	PGA增益选择位。 0000 PGA增益 = 1。 0001 PGA增益 = 2。 0010 PGA增益 = 4。 0011 PGA增益 = 8。 0100 PGA增益 = 16。 0101 PGA增益 = 32。 0110 PGA增益 = 64。 0111 PGA增益 = 128。 1000至1111 = 保留。	
3	ADCMOD2	调制器2倍增益。此位用于将PGA输出放大2倍以提高分辨率。 0: 禁用调制器2倍增益。 1: 使能调制器2倍增益。	
2:0	ADCMD	ADC模式位。	
		设置	**功能**
		000	ADC关断模式。ADC电路关断。这将关闭ADC和PGA。关断处理器之前，务必保留7 µs的软件延迟时间。
		001	连续转换模式。使能的ADC以F_{ADC}连续产生转换结果。RDY必须清0，新数据才能写入ADCxDAT。
		010	单次转换模式。在使能的ADC上执行单次转换。RDY置1后，ADC进入空闲模式。
		011	空闲模式。ADC上电但处于复位状态。校准后进入。
		100	内部零电平校准。利用内部产生的0 V进行。ADC输入短路连接到选定通道的负引脚。结果写入各使能ADC的ADCxOF寄存器。当斩波关闭以校准ADC失调时，可使用此校准。注意，器件必须处于空闲模式，然后才能进入任何校准模式。
		101	内部满量程校准。利用内部/外部V_{REF}进行。增益 = 1。详情参见"ADC校准模式"部分。注意，器件必须处于空闲模式，然后才能进入任何校准模式。
		110	系统零电平校准。在选定通道上进行；应在外部将通道短路。注意，器件必须处于空闲模式，然后才能进入任何校准模式。
		111	系统满量程校准。 用户将正确FS电压增加到输入引脚。每次校准后，ADC回到空闲模式，ADCxSTA中的RDY和CAL位置1。 注意，器件必须处于空闲模式，然后才能进入任何校准模式。

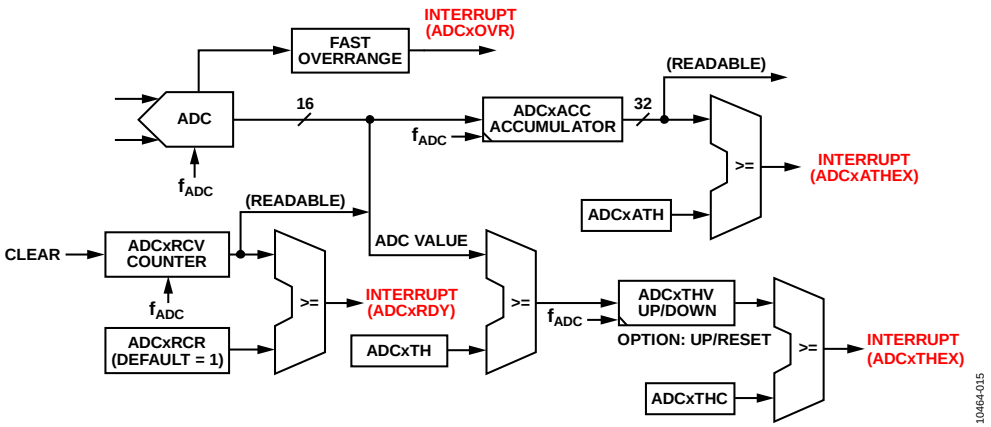


图16. ADC累加器/比较器/计数器功能框图

注意，结果计数器不会直接引起中断；相反，此计数器会屏蔽ADCxRDY中断。

计数器寄存器：ADCxRCR和ADCxRCV

地址：0x40030028；复位：0x0001；名称：ADC0RCR(仅ADuCM360)

地址：0x400300A8；复位：0x0001；名称：ADC1RCR(ADuCM360和ADuCM361)

ADCxRCR设置产生ADC中断前所必需的转换次数。此特性在ADCxPRO中使能。

表32. ADCxRCR的位功能描述

位	名称	描述
15:0	VALUE	ADC计数器寄存器。只有完成此寄存器指定的ADC样本数之后，ADCxRDY中断才会置位。 默认情况下，此寄存器的值为1，意味着每个样本都会产生中断。仅当ADCxPRO[0]置1时有效。

地址：0x4003002C；复位：0x0000；名称：ADC0RCV(仅ADuCM360)

地址：0x400300AC；复位：0x0000；名称：ADC1RCV(ADuCM360和ADuCM361)

ADCxRCV保存当前的ADC转换结果数量。计数器值用于屏蔽ADC中断，从而降低ADuCM360/ADuCM361的中断频率。当ADCxACC中必须累积某一数量的样本时，此值也很重要。必须从该MMR读取累积数量的样本以计算平均结果。

表33. ADCxRCV的位功能描述

位	名称	描述
15:0	VALUE	ADC计数器值寄存器。此只读寄存器指示自上次中断以来完成的ADC转换次数。

后处理寄存器：ADCxTH、ADCxTHC、ADCxTHV、ADCxACC、ADCxACC、ADCxATH

地址：0x40030030；复位：0x0000；名称：ADC0TH(仅ADuCM360)

地址：0x400300B0；复位：0x0000；名称：ADC1TH(ADuCM360和ADuCM361)

ADCxTH控制ADC输出上的比较器。该MMR保存的阈值与ADC转换结果的绝对值进行比较。比较以F_{ADC}的速率进行，使得ADuCM360/ADuCM361在高输入事件中可以中断，即便利用ADCxRC计数器屏蔽ADC RDY中断。

对于二进制补码编码，此寄存器的MSB会被忽略，因为进行的是绝对值比较(ADCxTH[14:0])。注意，0x8000映射到0x7FFF。

对于单极性编码则执行16位比较(ADCxTH[15:0])。

表34. ADCxTH的位功能描述

位	名称	描述
15:0	VALUE	ADC比较器阈值。 在双极性模式下，位[14:0]与ADCxDAT[27:13]进行比较。位15是符号位。 在单极性模式下，位[15:0]与ADCxDAT[27:12]进行比较。位15是符号位。

地址：0x40030034；复位：0x01；名称：ADC0THC(仅ADuCM360)

地址：0x400300B4；复位：0x01；名称：ADC1THC(ADuCM360和ADuCM361)

ADCxTHC决定必须有多少累积转换结果读数超过阈值(ADCxTH)，ADCxSTA中的ADC比较器阈值位才会置1，并且ADCxSTA[2]产生中断。低于阈值的值则使计数值递减或复位到0。

表35. ADC1THC的位功能描述

位	名称	描述
7:0	VALUE	ADC比较器计数设置，仅当ADCxPRO[3:2] = 10或11时有效。若有此寄存器选择数量的ADC样本超过比较器阈值，则ADCxTH中断置位。例如，若ADC0PRO[3:2] = 10且ADC0THC = 30，则要求有30个连续样本对应的ADC0DAT寄存器值高于ADC0TH中存储的阈值。

地址：0x40030038；复位：0x00；名称：ADC0THV(仅ADuCM360)

地址：0x400300B8；复位：0x00；名称：ADC1THV(ADuCM360和ADuCM361)

每当ADC转换结果的绝对值|Result| ≥ ADCxTH时，ADC0THV就会递增。每当ADC转换结果的绝对值|Result| < ADC0TH时，该寄存器就会递减或复位至0。通过设置ADCxPRO[3:2] MMR中的ADC比较器位，可对该功能进行配置。

表36. ADCxTHV的位功能描述

位	名称	描述
7:0	VALUE	ADC比较器计数设置，仅当ADCxPRO[3:2] = 10或11时有效。 此只读寄存器返回超过ADCxTH阈值的ADC0DAT值数量。 当ADCxTHC复位至0时，这些位复位至0。

地址： 0x4003003C；复位： 0x00000000；名称： ADC0ACC(仅ADuCM360)

地址： 0x400300BC；复位： 0x00000000；名称： ADC1ACC(ADuCM360和ADuCM361)

ADCxACC返回累加器状态。ADCxACC寄存器更新比ADCxDAT要早一两个ADC时钟。累加器上溢时，不会产生警告。65,535个样本(这是不会上溢的最大可能样本数)之后，可利用ADCxRCV复位ADCxACC寄存器。累加器是带符号二进制补码/单极性寄存器。有一种模式可将结果箝位至0，或者累加器可配置为增加负值。在ADCxPRO中禁用累加器或者重新配置ADC，可复位累加器。

注意，对于所有增益设置，ADCxDAT值都会右移12位。

表37. ADCxACC的位功能描述

位	名称	描述
31:0	VALUE	ADC累加器。ADCxPRO[5:4]使能累加器模式。

地址：0x40030040；复位：0x00000000；名称：ADC0ATH(仅ADuCM360)

地址：0x400300C0；复位：0x00000000；名称：ADC1ATH(ADuCM360和ADuCM361)

ADCxATH控制ADC输出上的累加器比较器。该MMR保存的阈值与累积ADC转换结果的绝对值进行比较。对于二进制补码编码，此寄存器的MSB会被忽略，因为进行的是绝对值比较(ADCxTH[14:0])。对于单极性编码则执行16位比较(ADCxTH[15:0])。

表38. ADCxATH的位功能描述

位	名称	描述
31:0	VALUE	ADC累加器比较器阈值。ADCxACC寄存器中的值与此寄存器中存储的值进行比较。 若ADCxACC > ADCxATH且ADCxPRO[5:4] = 11，则ADCxATH中断源置位。

后处理控制寄存器

地址：0x40030044；复位：0x00；名称：ADC0PRO(仅ADuCM360)

地址：0x400300C4；复位：0x00；名称：ADC1PRO(ADuCM360和ADuCM361)

写入ADCxPRO不会复位对应的ADC。

超范围或比较器中断置位时，清除中断屏蔽、禁用比较器(对于累加器则是清除ACC)或重新配置ADC会清除这些中断。

表39. ADCxPRO的位功能描述

位	名称	描述	
7:6	保留		
5:4	ADCxACCEN	ADC累加器使能。	
		设置	**功能**
		00	累加器禁用。
		01	累加器使能。负ADC结果会使累加器递减，下限为0。
		10	累加器使能。负ADC结果会使累加器递减，无下限。
		11	累加器和比较器使能。若ADCxACC中的值超过ADCxATH，则产生一个中断。要使能此中断，ADCxMSKI[3]必须置1。发生此中断时，ADCxSTA[3]置1，表示已发生ADC累加器中断。
3:2	ADCxCMPEN	ADC比较器使能。要使能ADC比较器中断，ADCxMSKI[2]必须置1。发生此中断时，ADCxSTA[2]置1，表示已发生ADC比较器中断。	
		设置	**功能**
		00	比较器禁用。
		01	比较器使能。若ADCxDAT > ADCxTH，则产生中断。
		10	带计数器的比较器使能。对于ADCxTHC中指定数量的样本，若ADCxDAT > ADCxTH，则产生中断。一个ADCxDAT < ADCxTH的样本结果就会将计数器复位至0。
		11	带计数器的比较器使能。对于ADCxTHC中指定数量的样本，若ADCxDAT > ADCxTH，则产生中断。一个ADCxDAT < ADCxTH的样本结果会使计数器递减，下限为0。
1	ADCxOREN	ADC超范围使能。	
0	ADCxRCEN	ADC结果计数器使能位。	

数据寄存器

地址：0x40030048；复位：0x00000000；名称：ADC0DAT(仅ADuCM360)

表40. ADC0DAT的位功能描述

位	名称	描述
31:0	VALUE	ADC0数据输出寄存器。更多信息参见“ADC数据寄存器”部分。

地址：0x400300C8；复位：0x00000000；名称：ADC1DAT(ADuCM360和ADuCM361)

表41. ADC1DAT的位功能描述

位	名称	描述
31:0	VALUE	ADC1数据输出寄存器。更多信息参见“ADC数据寄存器”部分。

步进检测MMR

步进检测控制寄存器

地址：0x400300E0；复位：0x0000；名称：DETCON

基准电压检测和步进检测滤波器的控制寄存器。

表42. DETCON的位功能描述

位	名称	描述
15:9	保留	保留。始终置为0。
8	REFDET	使能外部基准电压检测电路。 0: 禁用外部基准电压检测电路。 1: 使能外部基准电压检测电路。
7	SINC2	使能sinc2滤波器。 0: 禁用sinc2滤波器。 1: 使能sinc2滤波器。 最新sinc2数据与之前的3个输出相比较。若差值超过预定义阈值，则标志位(DETSTA[1])置1。
6:3	保留	保留。始终置为0。
2	ADCSEL	选择ADuCM360的ADC。此位允许用户选择步进检测滤波器是针对ADC0还是ADC1使能。 对于ADuCM361，此位保留，必须置1。 0: sinc2滤波器针对ADC0使能(仅对ADuCM360有效)。 1: sinc2滤波器针对ADC1使能(仅对ADuCM360有效)。
1:0	RATE	控制sinc2滤波器的内部间隔。 00: sinc2滤波器每2 ms输出一次。 01: sinc2滤波器每4 ms输出一次。 10: sinc2滤波器每6 ms输出一次。 11: sinc2滤波器每8 ms输出一次。

步进检测状态寄存器

地址：0x400300E4；复位：0x00；名称：DETSTA

表43. DETSTA的位功能描述

位	名称	描述
7:5	保留	
4	REFSTA	置1时，表示外部基准电压低于0.4 V或开路。 向DETSTA[3]写入1可将此位清0。 默认清0。
3	DATOF	数据上溢状态位。 置1时，表示在读取前一值之前，STEPDAT又被写入。 用户及时读取STEPDAT时清0。
2	STEPERR	置1时，表示由于超范围或欠范围等事件，sinc2滤波器转换已被箝位至+FS/−FS。 清0。
1	STEPFLAG	置1时，表示检测到大于阈值的步进变化。 读取STEPDAT后清0。
0	STEPDATRDY	置1时，表示检测到大于阈值的步进变化；随后提供STEPDAT中的粗略数据， 其为步进检测滤波器的输出。 STEPDATRDY表示STEPDAT可用。 每隔2 ms/4 ms/6 ms/8 ms提供一次STEPDAT，由DETCON[1:0]配置。

步进检测滤波器寄存器

地址：0x400300E8；复位：0x0000；名称：STEPTH

表44. STEPTH的位功能描述

位	名称	描述
15:10	保留	
9:0	VALUE	步进检测滤波器的10位阈值寄存器；*阈值 = (Value + 1)/2048 × V_{REF}* 0000000000 = (0 + 1)/2048 × V_{REF} ≈ 0.05% × V_{REF} 0000000001 = (1 + 1)/2048 × V_{REF} ≈ 0.1% × V_{REF} ... 0000000011 = (3 + 1)/2048 × V_{REF} ≈ 0.2% × V_{REF} ... 1111111111 = (1023 + 1)/2048 × V_{REF} ≈ 50% × V_{REF}

步进检测滤波器寄存器

地址：0x400300EC；复位：0x00000000；名称：STEPDAT

表45. STEPDAT的位功能描述

位	名称	描述
31:0	VALUE	ADC0或ADC1的16位转换结果，由DETCON[2]配置。

ADC DMA模式配置寄存器

地址：0x400300F8；复位：0x0000；名称：ADCDMACON

利用DMA控制器写入ADC MMR寄存器时，确保先完成对ADC0寄存器的写操作，再使能对ADC1寄存器的写操作。

支持同时使能ADC0和ADC1值的DMA读取。

表46. ADCDMACON的位功能描述

位	名称	描述
15:5	保留	
4	SINC2DMAEN	sinc2通道使能DMA。 0: 禁用sinc2 DMA访问。 1: 使能sinc2 DMA访问。
3	ADC1DMAEN	ADC1寄存器的DMA使能——ADC1CON、ADC1FLT、ADC1CP、ADC1CN、ADC1GN和ADC1OF。为使此位生效，ADC1DMAEN必须置1。 0: 利用DMA从用户存储器写入ADC1寄存器。 1: 利用DMA将ADC1DAT读出到用户存储器。
2	ADC1CTRL	ADC1通道使能DMA。 0: 禁用ADC1 DMA访问。 1: 使能ADC1 DMA访问。
1	ADC0DMAEN	ADC0寄存器的DMA使能——ADC0CON、ADC0FLT、ADC0CP、ADC0CN、ADC0GN和ADC0OF。为使此位生效，ADC0DMAEN必须置1。 0: 利用DMA从用户存储器写入ADC0寄存器(仅对ADuCM360有效)。 1: 利用DMA将ADC0DAT读出到用户存储器(仅对ADuCM360有效)。
0	ADC0CTRL	ADC0通道使能DMA。 0: 禁用ADC0 DMA访问(仅对ADuCM360有效)。 1: 使能ADC0 DMA访问(仅对ADuCM360有效)。

内部基准电压控制寄存器

地址：0x40008840；复位：0x0000；名称：REFCTRL

表47. REFCTRL的位功能描述

位	名称	描述
15:1	保留	
0	REFPD	关断基准电压源。要使ADC工作，此位必须清0，无论是否选择外部基准电压源。 0: 使能内部基准电压源模块。 1: 关断内部基准电压源模块。

激励电流源控制寄存器

地址：0x40008850；复位：0xC0；名称：IEXCCON

表48. IEXCCON的位功能描述

位	名称	描述
7	PD	IEXC关断。 0: 使能激励电流源模块。 1: 关断激励电流源模块。
6	REFSEL	IREF源。 0: 选择IREF引脚上的外部电流基准电阻源。 1: 选择内部电流基准电阻源。
5:3	IPSEL1	选择IEXC1输出引脚。IEXCCON[5]置1时，使能激励电流源1。 IEXCCON[5]清0时，禁用激励电流源1。 000: 保留。 … 011: 保留。 100: IEXC1在AIN4上输出。 101: IEXC1在AIN5上输出。 110: IEXC1在AIN6上输出。 111: IEXC1在AIN7上输出。
2:0	IPSEL0	选择IEXC0输出引脚。IEXCCON[2]置1时，使能激励电流源0。 IEXCCON[2]清0时，禁用激励电流源0。 000: 保留。 … 011: 保留。 100: IEXC0在AIN4上输出。 101: IEXC0在AIN5上输出。 110: IEXC0在AIN6上输出。 111: IEXC0在AIN7上输出。

激励电流数据寄存器

地址：0x40008854；复位：0x06；名称：IEXCDAT

表49. IEXCDAT的位功能描述

位	名称	描述
7:6	保留	
5:1	IDAT	输出电流。这5位设置各引脚的输出电流。详情参见表50。
0	IDAT0	10 μA使能。 0: 禁用10 μA输出电流。 1: 使能10 μA输出电流。

表50. 基于IEXCDAT[5:0]的输出电流值

IEXCDAT[5:3]	IEXCDAT[2:1] = 11	IEXCDAT[2:1] = 10	IEXCDAT[2:1] = 01	IEXCDAT[2:1] = 00	IEXCDAT[0] = 1
000	0 μA	0 μA	0 μA	0 μA	如果此位置1，则左侧的值增加10 μA。
001	200 μA	150 μA	100 μA	50 μA	
010	400 μA	300 μA	200 μA	100 μA	
011	600 μA	450 μA	300 μA	150 μA	
100	800 μA	600 μA	400 μA	200 μA	
101	1000 μA	750 μA	500 μA	250 μA	
110	1000 μA	750 μA	500 μA	250 μA	
111	1000 μA	750 μA	500 μA	250 μA	

DAC

DAC特性

- 12位电压输出DAC
- 两种可选范围：
  - 0 V至V_{REF}(内部1.2 V带隙基准电压)
  - 0 V至AVDD_REG (1.8 V)

DAC概述

ADuCM360/ADuCM361集成一个电压输出DAC。在正常模式下，DAC的分辨率为12位。在插值模式下，DAC的分辨率为16位(14个有效位)。DAC有一个轨到轨电压输出缓冲器，能够驱动5 kΩ||100 pF负载。DAC也可配置为NPN晶体管驱动器模式。在这种工作模式下，DAC输出缓冲器可配置为利用极少的外部元件控制一个双线4 mA至20 mA环路接口。当DAC处于NPN晶体管驱动器模式时，不支持DAC插值。

DAC有两个可选范围：

- 0 V至V_{REF} (内部1.2 V带隙基准电压)
- 0 V至AVDD_REG (1.8 V)

注意：要使能DAC模块的系统时钟，CLKDIS[7]必须清0。此位必须清0才能使能DAC。

DAC工作原理

DAC可通过一个控制寄存器和一个数据寄存器进行配置。片内DAC架构由一电阻串DAC和一个输出缓冲放大器构成。

用户可通过软件来选择DAC基准电压源：

- 0至V_{REF}模式：DAC输出传递函数范围为0 V至1.2 V内部基准电压V_{REF}。
- 0至AVDD_REG模式：DAC输出传递函数范围为0 V至1.8 V。

DAC可被配置为以下三种用户模式：正常模式、DAC插值模式和NPN晶体管驱动器模式。

DAC正常模式，DACCON[3] = 0

在该模式下，DAC配置为12位电压输出DAC。在默认的情况下，可以使用DAC缓冲器，但输出缓冲器可以被禁用。当DAC输出缓冲器禁用时，DAC仅可驱动10 pF的容性负载。通过设置DACCON[6] (DACBUFBYPASS)，可以禁用DAC缓冲器。

每个DAC输出缓冲放大器都有一个真轨到轨输出级。也就是说，当输出空载时，DAC输出摆幅通常能够达到AGND和AVDD_REG的5 mV范围以内。当驱动一个5 kΩ阻性负载到地时，除了代码0至100(在0至AVDD_REG模式中为代码3995至4095)外，整个传递函数都能保证符合DAC线性度规格要求。地和AVDD_REG附近的线性度下降是由输出放大器的饱和引起的，图17反映了这种效应的一般表现(失调和增益误差忽略不计)。

图17中的虚线为理想的传递函数，实线代表转换可能具有端点非线性(由输出放大器饱和引起)的传递函数。注意，图17仅代表输入范围为0至AVDD_REG模式时的传递函数。在0至V_{REF}模式下，下半部分的非线性度是相似的。然而，传递函数的上半部分一直到端点都表现为理想的线性，这说明DAC输出没有端点线性误差。

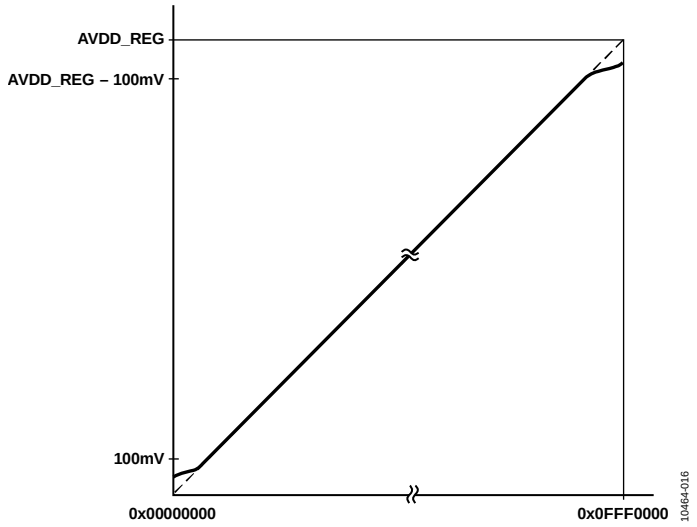


图17. 放大器饱和引起的DAC端点非线性

当有输出负载时，图17所示的端点非线性会变得更差。在正常模式下，ADuCM360/ADuCM361数据手册中大多数技术参数都是在DAC输出与一个接地的5 kΩ阻性负载相连的条件下得到的。由于DAC输出被强制提供更多的源电流或吸电流，相应的顶部或底部非线性区域将变得更大。而当需要更大电流时，这会明显地限制输出电压摆幅。

DAC插值模式，DACCON[3] = 1

与正常模式相比，插值模式下DAC的输出分辨率较高(16位，约14个有效位)且更新速率较慢。更新速率由插值时钟速率控制，且在DACCON[2]中设置。在该模式下，为形成稳定的电压，需要连接一个外部RC滤波器。由于外部滤波器和较慢的更新速率，这种模式下DAC带宽较低。

DAC NPN晶体管驱动器模式，DACCON[8] = 1

在DAC NPN晶体管驱动器模式或4 mA至20 mA驱动器模式下，DAC输出用于控制双线4 mA至20 mA环路网络中的NPN晶体管的基极。DAC输出缓冲器用作普通运算放大器，如图18所示。

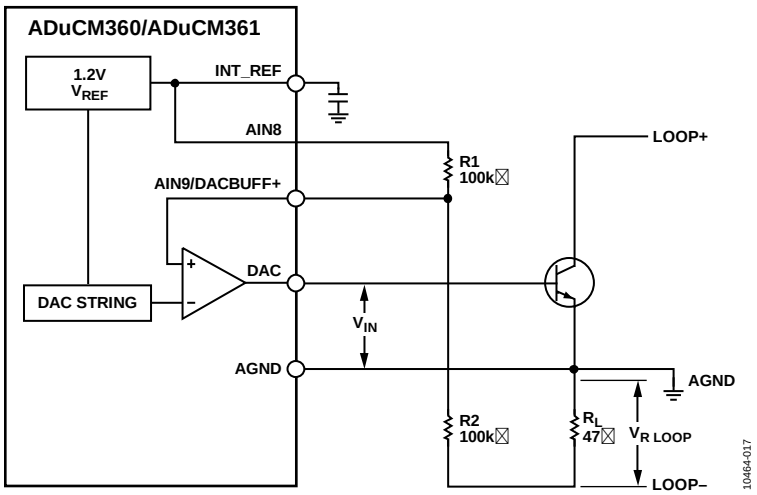


图18. DAC NPN模式： 典型操作

4 mA至20 mA反馈电路由ADuCM360/ADuCM361 12位DAC输出控制。DAC输出控制电路中标记为RLoop的47 Ω电阻上的电压。通过控制此电阻上的电压，可以精确控制反馈到4 mA至20 mA主机的电流。

请注意，RLoop的顶端连接ADuCM360/ADuCM361地。RLoop的底端连接环路地。由于这个原因，ADuCM360/ADuCM361的DAC输出缓冲器和稳压器电流，加上DAC输出设置的电流，一同流过RLoop。

当DACCON[8] = 1时，AIN8上存在V_{REF}。1.2 V片内基准电压源在外部连接到R1。R1和R2之间的电压(VR12)连接到DAC输出缓冲器的同相输入端。VR12设置运算放大器的输出电压范围。若R1 = R2，则此范围为0 mV至600 mV。

R1和R2的结点电压可表示为：

$$Vr12 = (VRLoop + Int\_Vref) \times R2/(R1 + R2) - VRLoop$$

环路建立后，Vin = Vr12。

由于R1 = R2，便得到

$$Vin = (VRLoop + Int\_Vref)/2 - VRLoop = Int\_Vref/2 - VRLoop/2$$

或

$$VRLoop = Int\_Vref - 2Vin$$

当Vin = 0 (DACDAT = 0)时流过满量程电流，此时VRLoop = Int_Vref/2。

因此，满量程电流为Int_Vref/RLoop，或者约24 mA。

当Vin = Int_Vref/2 (DAC = 600 mV)时(其中Int_Vref为1.2 V内部基准电压)，无电流流过RLoop。

DAC DMA操作

DMA通道8为DAC DMA通道。如果用户希望将DAC配置为波形发生器，但不想在每次更新DAC输出电压时中断处理器，这会是一个有用的特性。定时器1必须设置为触发对DACDAT寄存器的DMA写操作。

示例代码：配置DAC的DMA操作

```
void DACDMAINIT( void)
{
pADI_DAC->DACCON = 0x410;                    // Enable DAC, 0-INT_VREF range, timer 1
triggered
Dma_Init();                                  // Init DMA structures
DACDMAWRITE(uxDACDMA, 16);                   // Setup DMA control registers
NVIC_EnableIRQ(DMA_DAC_IRQn);                // Enable DMA DAC Interrupt
pADI_DMA->DMACFG = 0x1;                      // Enable DMA mode in DMA controller
pADI_DMA->DMAENSET = 0x100;                  // Select DAC channel in DMA controller
pADI_TM1->LD = 0x00000088;                   // the min clock number is 0x17;
pADI_TM1->CON = 0x00000018;
}

void DACDMAWRITE(unsigned long *pucTX_DMA, unsigned int iNumRX)
{
   DmaDesc Desc;

                                             // Common configuration of all the descriptors
used here
    Desc.ctrlCfg.Bits.cycle_ctrl       = cyclectrl_basic;
    desc.ctrlcfg.bits.next_useburst    = 0x0;
    desc.ctrlcfg.bits.r_power          = 0;
    Desc.ctrlCfg.Bits.src_prot_ctrl    = 0x0;
    Desc.ctrlCfg.Bits.dst_prot_ctrl    = 0x0;
    Desc.ctrlCfg.Bits.src_size         = SRCSIZE_WORD;
    Desc.ctrlCfg.Bits.dst_size         = DSTSIZE_WORD;

                                                    // TX Primary Descriptor
```

```
Desc.srcEndPtr                              = (unsigned long)(pucTX_DMA + iNumRX - 0x1);
Desc.destEndPtr                             = (unsigned long)&DACDAT;
Desc.ctrlCfg.Bits.n_minus_1                 = iNumRX - 0x1;
Desc.ctrlCfg.Bits.src_inc                   = INC_WORD;
Desc.ctrlCfg.Bits.dst_inc                   = INC_NO;
*Dma_GetDescriptor(DMA_DAC_REQ_OUT,FALSE)   = Desc;
}

void DMA_DAC_Out_Int_Handler()
{
NVIC_DisableIRQ(DMA_DAC_IRQn);                      // Clear Interrupt source
}
```

DAC存储器映射寄存器

表51. DAC存储器映射寄存器地址(基地址：0x40020000)

偏移	名称	描述	访问类型	默认值
0x0000	DACCON	控制寄存器	RW	0x0200
0x0004	DACDAT	数据寄存器	R	0x00000000

DAC控制寄存器

地址：0x40020000；复位：0x200；名称：DACCON

表52. DACCON寄存器位功能描述

位	名称	描述
15:11	保留	保留。
10	DMAEN	0: DAC正常模式。 1: DAC DMA模式使能。
9	PN	0: 使能DAC。 1: 关断DAC输出(DAC输出为三态)。(默认)
8	NPN	0: 在正常模式下使能DAC输出缓冲器。 1: 在NPN晶体管驱动器模式下使能DAC输出缓冲器。
7	保留	保留。
6	BUFBYP	0: 缓冲DAC输出。 1: 将输出缓冲器旁路，并直接向输出引脚发送DAC输出。
5	CLK	0: 在UCLK的负沿更新DAC。 1: 在定时器1的负沿更新DAC。对波形生成来说，这是一个非常理想的模式。 在这种模式下，波形的下一个值以固定的定时器1间隔写入DACDAT。
4	CLR	写入此位会立即影响DAC输出。 0: 清除DAC输出并将DACDAT设为0。 1: DAC正常工作。
3	MDE	0: DAC在12位正常模式下工作。 1: DAC在16位插值模式下工作。
2	RATE	此位在插值模式下使用。 0: 将插值时钟配置为UCLK/32。 1: 将插值时钟配置为UCLK/16。
1:0	RNG	DAC范围。 00: 0 V至V_{REF} (1.2 V)(内部基准电压源)。 01: 保留。 10: 保留。 11: 0 V至AVDD_REG (1.8 V)。

DAC数据寄存器

地址： 0x40020004；复位： 0x00000000；名称：DACDAT

表53. DACDAT寄存器位功能描述

位	名称	描述
31:28	保留	
27:16	DAT12	12位DAC数据。
15:12	DAT16	额外4位，用于插值模式。
11:0	保留	

系统异常和外设中断

CORTEX-M3和故障管理

ADuCM360/ADuCM361集成一个Cortex-M3处理器，其支持多种系统异常和外设产生的中断。

表54. 系统异常

编号	类型	优先级	描述
1	复位	−3(最高)	任何复位。
2	NMI	−2	连接至ADuCM360/ADuCM361电源监控器的不可屏蔽中断。
3	硬故障	−1	未使能相应故障处理程序时的所有故障条件。
4	存储器管理故障	可编程	存储器管理故障；访问非法位置。
5	总线故障	可编程	预取故障、存储器存取故障、数据中止和其他地址/存储器相关故障。
6	使用故障	可编程	同执行未定义指令或尝试非法转换状态。
7至10	保留	不适用	
11	SVCall	可编程	采用SVC指令的系统服务调用。用于系统功能调用。
12	调试监视	可编程	调试监视(断点、观察点或外部调试请求)。
13	保留	不适用	
14	PendSV	可编程	可挂起的系统服务请求。用于让系统调用排队，直至其它任务和中断得到处理为止。
15	SYSTICK	可编程	系统节拍定时器。

外设中断由NVIC控制，如表55所示。所有中断源都可以将器件从模式1、模式2或模式3唤醒。仅有限数量的中断可将处理器从低功耗模式(模式4和模式5)唤醒，如表55所示。当从模式4或模式5唤醒器件时，它回到模式0。如果处理器正在进行中断处理，同时处理器进入模式1到模式5中的任意功耗模式，则只有优先级高于当前中断的中断源才能唤醒器件。

配置中断通常需要两步：

- 配置外设产生NVIC中断请求。
- 针对该外设请求配置NVIC。

表55. 中断矢量表

位置编号	矢量	从模式1唤醒处理器	从模式2或模式3唤醒处理器	从模式4或模式5唤醒处理器
0	唤醒定时器	是	是	是
1	外部中断0	是	是	是
2	外部中断1	是	是	是
3	外部中断2	是	是	是
4	外部中断3	是	是	是
5	外部中断4	是	是	是
6	外部中断5	是	是	是
7	外部中断6	是	是	是
8	外部中断7	是	是	是
9	看门狗定时器	是	是	是
10	保留	不适用	不适用	不适用
11	定时器0	是	否	否
12	定时器1	是	否	否
13	ADC0	是	否	否
14	ADC1	是	否	否

位置编号	矢量	从模式1唤醒处理器	从模式2或模式3唤醒处理器	从模式4或模式5唤醒处理器
15	sinc2	是	否	否
16	闪存控制器	是	否	否
17	UART	是	否	否
18	SPI0	是	否	否
19	SPI1	是	否	否
20	I²C从机	是	否	否
21	I²C主机	是	否	否
22	DMA错误	是	否	否
23	DMA SPI_1 Tx IRQ	是	否	否
24	DMA SPI_1 Rx IRQ	是	否	否
25	DMA UART Tx IRQ	是	否	否
26	DMA UART Rx IRQ	是	否	否
27	DMA I²C从机Tx	是	否	否
28	DMA I²C从机Rx	是	否	否
29	DMA I²C主机Tx	是	否	否
30	DMA I²C主机Rx	是	否	否
31	DMA DAC输出	是	否	否
32	DMA ADC0	是	否	否
33	DMA ADC1	是	否	否
34	DMA sinc2输出/步进检测	是	否	否
35	PWM_TRIP	是	否	否
36	PWM0中断	是	否	否
37	PWM1中断	是	否	否
38	PWM2中断	是	否	否

在Cortex-M3处理器内部，最高用户可编程优先级(0)被当作第四优先级对待，位于复位、NMI和硬故障之后。注意，0是所有可编程优先级的默认优先级。如果将相同优先级赋予给两个或更多中断，其硬件优先级(位置编号越低则优先级越高)决定处理器激活中断的顺序。例如，若SPI0和SPI1均为优先级1，则SPI0具有更高优先级。

要使能表55中从0到31的任何外设的中断，请设置ISER0寄存器中的相应位。ISER0是一个32位寄存器，每一位与表55中的前32条对应。

例如，要使能NVIC中的ADC0中断源，请设置ISER0[13] = 1。同样，要禁用ADC0中断，请设置ICER0[13] = 1。

要使能表55中从32到38的任何外设的中断，请设置ISER1寄存器中的相应位。ISER1是一个32位寄存器，位0至位6与表55中的第32到38条对应。

例如，要使能NVIC中的PWM0中断源，请设置ISER1[4] = 1。同样，要禁用PWM0，请设置ICER1[4] = 1。

表56列出了使能和禁用相关中断以及设置优先级的寄存器。

表56. NVIC寄存器

地址	ADI公司头文件名称	描述	访问类型
0xE000E004	ICTR	显示NVIC支持的中断线数。	R
0xE000E010	STCSR	SYSTICK控制和状态寄存器。	RW
0xE000E014	STRVR	SYSTICK重载值寄存器。	RW
0xE000E018	STCVR	SYSTICK当前值寄存器。	RW
0xE000E01C	STCR	SYSTICK校准值寄存器。	R
0xE000E100	ISER0	设置IRQ0至IRQ31使能。每一位对应于表55中的中断0至中断31。	RW
0xE000E104	ISER1	设置IRQ32至IRQ38使能。每一位对应于表55中的中断32至中断38。	RW
0xE000E180	ICER0	通过设置相应的位清除IRQ0至IRQ31。每一位对应于表55中的中断0至中断31。	RW
0xE000E184	ICER1	通过设置相应的位清除IRQ32至IRQ38。每一位对应于表55中的中断32至中断38。	RW
0xE000E200	ISPR0	设置IRQ0至IRQ31挂起。每一位对应于表55中的中断32至中断38。	RW
0xE000E204	ISPR1	设置IRQ32至IRQ38挂起。每一位对应于表55中的中断32至中断38。	RW
0xE000E280	ICPR0	清除IRQ0至IRQ31挂起。每一位对应于表55中的中断32至中断38。	RW
0xE000E284	ICPR1	清除IRQ32至IRQ38挂起。每一位对应于表55中的中断32至中断38。	RW
0xE000E300	IABR0	IRQ0至IRQ31有效位。	RW
0xE000E304	IABR1	IRQ32至IRQ38有效位。	RW
0xE000E400	IPR0	IRQ0至IRQ3优先级。	RW
0xE000E404	IPR1	IRQ4至IRQ7优先级。	RW
0xE000E408	IPR2	IRQ8至IRQ11优先级。	RW
0xE000E40C	IPR3	IRQ12至IRQ15优先级。	RW
0xE000E410	IPR4	IRQ16至IRQ19优先级。	RW
0xE000E414	IPR5	IRQ20至IRQ23优先级。	RW
0xE000E418	IPR6	IRQ24至IRQ27优先级。	RW
0xE000E41C	IPR7	IRQ28至IRQ31优先级。	RW
0xE000E420	IPR8	IRQ32至IRQ35优先级。	RW
0xE000E424	IPR9	IRQ36至IRQ38优先级。	RW
0xE000ED00	CPUID	CPUID基址寄存器。	R
0xE000ED04	ICSR	中断控制和状态寄存器。	RW
0xE000ED08	VTOR	矢量表偏移寄存器。	RW
0xE000ED0C	AIRCR	应用程序中断/复位控制寄存器。	RW
0xE000ED10	SCR	系统控制寄存器。	RW
0xE000ED14	CCR	配置控制寄存器。	RW
0xE000ED18	SHPR1	系统处理程序寄存器1。	RW
0xE000ED1C	SHPR2	系统处理程序寄存器2。	RW
0xE000ED20	SHPR3	系统处理程序寄存器3。	RW
0xE000ED24	SHCRS	系统处理程序控制和状态。	RW
0xE000ED28	CFSR	可配置故障状态。	RW
0xE000ED2C	HFSR	硬故障状态。	RW
0xE000ED34	MMAR	MemManage故障地址寄存器。	RW
0xE000ED38	BFAR	总线故障地址。	RW
0xE000EF00	STIR	软件触发中断寄存器。	W

外部中断配置

实现了8个外部中断。这8个外部中断可以独立配置来检测以下类型事件的任意组合：

- 边沿：上升沿、下降沿或上升沿与下降沿同时检测。检测到低到高跃迁、高到低跃迁或低到高/高到低跃迁时，向NVIC发生一个中断信号(脉冲)。
- 电平： 高或低。在NVIC中产生一个中断信号并保持置位，直到产生中断解除条件。电平必须维持至少一个内核时钟周期才能被检测到。

外部中断检测单元模块位于始终开启部分，外部中断可以将器件从休眠模式唤醒。

中断存储器映射寄存器

中断检测单元的MMR包含在始终开启部分中，基地址为0x40002400。

表57. 中断检测单元存储器映射寄存器地址(基地址：0x40002400)

偏移	名称	描述	访问类型	默认值
0x0020	EI0CFG	外部中断配置寄存器0	RW	0x0000
0x0024	EI1CFG	外部中断配置寄存器1	RW	0x0000
0x0030	EICLR	外部中断清零寄存器	RW	0x0000
0x0034	NMICLR	无法屏蔽的中断清零	RW	0x0000

外部中断配置寄存器0

地址：0x40002420；复位：0x0000；名称：EI0CFG

表58. EI0CFG寄存器位功能描述

位	名称	描述
15	IRQ3EN	外部中断3使能位 0: 禁用外部中断3 1: 使能外部中断3
14:12	IRQ3MDE	外部中断3模式寄存器 000: 上升沿 001: 下降沿 010: 上升沿和下降沿 011: 高电平 100: 低电平 101: 下降沿(同001) 110: 上升沿和下降沿(同010) 111: 高电平(同011)
11	IRQ2EN	外部中断2使能位 0: 禁用外部中断2 1: 使能外部中断2
10:8	IRQ2MDE	外部中断2模式寄存器 000: 上升沿 001: 下降沿 010: 上升沿和下降沿 011: 高电平 100: 低电平 101: 下降沿(同001) 110: 上升沿和下降沿(同010) 111: 高电平(同011)
7	IRQ1EN	外部中断1使能位 0: 禁用外部中断1 1: 使能外部中断1
6:4	IRQ1MDE	外部中断1模式寄存器 000: 上升沿 001: 下降沿 010: 上升沿和下降沿 011: 高电平 100: 低电平 101: 下降沿(同001) 110: 上升沿和下降沿(同010) 111: 高电平(同011)
3	IRQ0EN	外部中断0使能位 0: 禁用外部中断0 1: 使能外部中断0
2:0	IRQ0MDE	外部中断0模式寄存器 000: 上升沿 001: 下降沿 010: 上升沿和下降沿 011: 高电平 100: 低电平 101: 下降沿(同001) 110: 上升沿和下降沿(同010) 111: 高电平(同011)

外部中断配置寄存器1

地址：0x40002424；复位：0x0000；名称：EI1CFG

表59. EI1CFG寄存器位功能描述

位	名称	描述
15	IRQ7EN	外部中断7使能位 0: 禁用外部中断7 1: 使能外部中断7
14:12	IRQ7MDE	外部中断7模式寄存器 000: 上升沿 001: 下降沿 010: 上升沿和下降沿 011: 高电平 100: 低电平 101: 下降沿(同001) 110: 上升沿和下降沿(同010) 111: 高电平(同011)
11	IRQ6EN	外部中断6使能位 0: 禁用外部中断6 1: 使能外部中断6
10:8	IRQ6MDE	外部中断6模式寄存器 000: 上升沿 001: 下降沿 010: 上升沿和下降沿 011: 高电平 100: 低电平 101: 下降沿(同001) 110: 上升沿和下降沿(同010) 111: 高电平(同011)
7	IRQ5EN	外部中断5使能位 0: 禁用外部中断5 1: 使能外部中断5
6:4	IRQ5MDE	外部中断5模式寄存器 000: 上升沿 001: 下降沿 010: 上升沿和下降沿 011: 高电平 100: 低电平 101: 下降沿(同001) 110: 上升沿和下降沿(同010) 111: 高电平(同011)
3	IRQ4EN	外部中断4使能位 0: 禁用外部中断4 1: 使能外部中断4
2:0	IRQ4MDE	外部中断4模式寄存器 000: 上升沿 001: 下降沿 010: 上升沿和下降沿 011: 高电平 100: 低电平 101: 下降沿(同001) 110: 上升沿和下降沿(同010) 111: 高电平(同011)

外部中断清零寄存器

地址：0x40002430；复位：0x0000；名称：EICLR

表60. EICLR寄存器位功能描述

位	名称	描述
15:8	保留	保留。
7	IRQ7	外部中断7清零位。 1: 清除内部中断标志。 硬件自动清0。
6	IRQ6	外部中断6清零位。 1: 清除内部中断标志。 硬件自动清0。
5	IRQ5	外部中断5清零位。 1: 清除内部中断标志。 硬件自动清0。
4	IRQ4	外部中断4清零位。 1: 清除内部中断标志。 硬件自动清0。
3	IRQ3	外部中断3清零位。 1: 清除内部中断标志。 硬件自动清0。
2	IRQ2	外部中断2清零位。 1: 清除内部中断标志。 硬件自动清0。
1	IRQ1	外部中断1清零位。 1: 清除内部中断标志。 硬件自动清0。
0	IRQ0	外部中断0清零位。 1: 清除内部中断标志。 硬件自动清0。

从中断处理程序回到正常工作之前，确保寄存器写操作已全部完成。使用DSB指令，例如：

```
void Ext_Int4_Handler ()
{
 EiClr(EXTINT4);
 __DSB();
}
```

这样可确保先完成待处理的存储器操作，再开始执行新指令。

无法屏蔽的中断清零寄存器

地址：0x40002434；复位：0x0000；名称：NMICLR

表61. NMICLR寄存器位功能描述

位	名称	描述
15:1	保留	保留。
0	CLEAR	NMI清零位。 由用户置1，清除NMI的内部标志。(内部标志对用户不可见。) IDU置位外部中断后，会产生一个中断脉冲发送至Cortex-M3。除非清除该外部中断，否则IDU会忽略同一端口上随后的中断，不会产生更多脉冲。 硬件自动清0。

从中断处理程序回到正常工作之前，确保寄存器写操作已全部完成。使用DSB指令，例如：

```
void Ext_Int4_Handler ()
{
 EiClr(EXTINT4);
 __DSB();
}
```

DMA控制器

DMA特性

- 12个专用DMA通道

DMA概述

直接存储器访问(DMA)用于在外设与存储器之间提供高速数据传输。DMA可以迅速移动数据而不需要任何处理器操作。这样可以腾出处理器资源用于执行其它操作。

DMA控制器总共有12个通道。这12个通道专门用来管理来自特定外设的DMA请求。通道分配如表62所示。

注意：要使能DMA模块的系统时钟，CLKDIS[8]必须清0，从而使能DMA操作。

表62. DMA通道分配

通道	外设
0	SPI1 Tx
1	SPI1 Rx
2	UART Tx
3	UART Rx
4	I^2C从机Tx
5	I^2C从机Rx
6	I^2C主机Tx
7	I^2C主机Rx
8	DAC DMA输出
9	ADC0
10	ADC1
11	SINC2输出/步进检测

这些通道连接到专用硬件DMA请求；各通道还支持软件触发。此配置由软件完成。

各DMA通道都有一个可编程优先级：默认或高。在一个优先级内，利用DMA通道号所确定的固定优先级进行仲裁。

DMA控制器支持多种DMA传输数据宽度：独立的来源和目标传输大小(字节、半字和字)。来源/目标地址的数据大小必须一致。

在DMA传输期间，当发生总线错误时，会产生DMA传输错误中断。

DMA控制器支持外设到存储器、存储器到外设和存储器到存储器传输，以及作为来源和目标访问闪存或SRAM。

DMA工作原理

DMA控制器与Cortex-M3处理器共享系统总线，从而执行直接存储器传输。当处理器和DMA指向相同的目标(存储器或外设)时，DMA请求可以让处理器暂停访问系统总线若干总线周期。

错误管理

读取或写入保留的地址空间，会产生DMA传输错误。在DMA读或写访问期间发生DMA传输错误时，会自动禁用故障通道。若NVIC中使能了该DMA错误中断，则还会产生一个中断。

中断

各DMA通道传输完成时，可以产生一个中断。针对各DMA通道，NVIC提供了独立的中断使能位。

DMA控制器取出位于SRAM存储器中的通道控制数据结构以执行数据传输。当允许使用DMA操作时，支持DMA的外设请求DMA控制器进行传输。在一个通道的设定DMA传输次数结束时，DMA控制器产生一个对应该通道的中断。此中断表示DMA传输已完成。

DMA优先级

一个通道的优先性由其编号和优先级决定。各通道可以有两个优先级：默认或高。具有高优先级的所有通道优先于具有默认优先级的所有通道。优先级相同时，编号较低的通道优先于编号较高的通道。通过写入DMAPRISET寄存器中的相应位，可以更改DMA通道优先级。

通道控制数据结构

各通道有两个与之相关的控制数据结构：主要数据结构和备选数据结构。对于简单传输模式，DMA控制器既可使用主要数据结构，也可使用备选数据结构。对于较复杂的数据传输模式，如乒乓式或分散/聚集式，DMA控制器同时使用主要数据结构和备选数据结构。各控制数据结构(主要或备选)在存储器中占用4个32位位置，如表63所示。整个通道控制数据结构如表64所示。

表63. 通道控制数据结构

偏移	名称	描述
0x00	SRC_END_PTR	来源端指针
0x04	DST_END_PTR	目标端指针
0x08	CHNL_CFG	控制数据配置
0x0C	保留	保留

在控制器能够执行DMA传输之前，必须在系统存储器SRAM的指定位置设置DMA通道相关的数据结构。

- 来源端指针存储器位置包含来源数据的端地址。
- 目标端指针存储器位置包含目标数据的端地址。
- 控制数据配置存储器位置包含通道配置控制数据。

编程确定来源和目标数据大小、传输次数和仲裁次数。

表64. 主要和备选DMA结构的存储器映射

通道	主要结构		备选结构	
通道11	保留，置0	0x0BC	保留，置0	0x1BC
	控制	0x0B8	控制	0x1B8
	目标端指针	0x0B4	目标端指针	0x1B4
	来源端指针	0x0B0	来源端指针	0x1B0
…	…	…	…	…
通道1	保留，置0	0x01C	保留，置0	0x11C
	控制	0x018	控制	0x118
	目标端指针	0x014	目标端指针	0x114
	来源端指针	0x010	来源端指针	0x110
通道0	保留，置0	0x00C	保留，置0	0x10C
	控制	0x008	控制	0x108
	目标端指针	0x004	目标端指针	0x104
	来源端指针	0x000	来源端指针	0x100

用户必须在源代码中定义DMA结构，如“示例代码：定义DMA结构”部分中的例子所示。定义结构之后，必须将其起始地址赋予给DMA基址指针寄存器DMAPDBPTR。

这样，各DMA通道的各寄存器便处于偏移地址(如表64之规定)加上DMAPDBPTR寄存器值。

示例代码： 定义DMA结构

```
memset(dmaChanDesc,0x0,sizeof(dmaChanDesc));
                                                // Setup the DMA base address pointer register.
pADI_DMA->DMAPDBPTR = (unsigned int) &dmaChanDesc;
```

控制数据配置

对于每次DMA传输，CHNL_CFG存储器位置向控制器提供DMA传输控制信息。

表65. 控制数据配置

位	名称	描述		
31:30	DST_INC	目标地址增量。地址增量取决于来源数据宽度，如下所示：		
		来源数据宽度	**DST_INC**	**目标地址增量**
		字节	00	字节。
			01	半字。
			10	字。
			11	不递增。地址仍为DST_END_PTR存储器位置包含的值。
		半字	00	保留。
			01	半字。
			10	字。
			11	不递增。地址仍为DST_END_PTR存储器位置包含的值。
		字	00	保留。
			01	保留。
			10	字。
			11	不递增。地址仍为DST_END_PTR存储器位置包含的值。
29:28	DST_SIZE	目标数据大小。必须始终与SRC_SIZE相同。 00: 字节。 01: 半字。 10: 字。 11: 保留。		
27:26	SRC_INC	来源地址增量。地址增量取决于来源数据宽度，如下所示：		
		来源数据宽度	**DST_INC**	**来源地址增量**
		字节	00	字节。
			01	半字。
			10	字。
			11	不递增。地址仍为SRC_END_PTR存储器位置包含的值。
		半字	00	保留。
			01	半字。
			10	字。
			11	不递增。地址仍为SRC_END_PTR存储器位置包含的值。
		字	00	保留。
			01	保留。
			10	字。
			11	不递增。地址仍为SRC_END_PTR存储器位置包含的值。
25:24	SRC_SIZE	来源数据大小。必须始终与DST_SIZE相同。 00: 字节。 01: 半字。 10: 字。 11: 保留。		
23:18	保留	未定义。写入0。		
17:14	R_POWER	设置这些位以控制在控制器重新仲裁之前有多少DMA传输可以发生。 对于所有涉及外设的DMA传输，必须设为0000。注意：对于涉及外设的DMA传输，若在此位置写入非0000的值，则DMA操作将是不确定的。		

位	名称	描述
13:4	N_MINUS_1	针对该通道配置的传输次数减1。10位值表示DMA传输次数(非总字节数)减1。 可能的值有： 0x000: 1次DMA传输。 0x001: 2次DMA传输。 0x002: 3次DMA传输。 … 0x3FF: 1024次DMA传输。
3	保留	未定义。写入0。
2:0	CYCLE_CTRL	DMA周期的传输类型。 000: 停止(无效)。 001: 基本。 010: 自动请求。 011: 乒乓。 100: 存储器分散/聚集式主要。 101: 存储器分散/聚集式备选。 110: 外设分散/聚集式主要。 111: 外设分散/聚集式备选。

在DMA传输过程中，但在仲裁之前，将CHNL_CFG写回系统存储器，并改变N_MINUS_1字段以反映尚未完成的传输次数。

最后，当DMA周期完成时，CYCLE_CTRL位变为无效，表示传输已完成。

DMA传输类型(CHNL_CFG[2:0])

DMA控制器支持5种类型的DMA传输。要选择不同的类型，须向控制数据结构的CHNL_CFG位置中的CYCLE_CTRL位(位[2:0])写入适当的值。

无效(CHNL_CFG[2:0] = 000)

这意味着该通道未使能任何DMA传输。控制器完成一个DMA周期之后，将周期类型设为无效，以防止重复相同的DMA周期。

基本(CHNL_ CFG[2:0] = 001)

在此模式下，控制器可配置为使用主要数据结构或备选数据结构。外设必须为每次数据传输提供一个请求。通道使能之后，当控制器收到请求时，它便执行如下操作：

1. 控制器执行一次传输。如果剩余传输次数为0，则转到第3步。
2. 控制器仲裁：
   a. 如果有更高优先级通道在请求服务，则控制器服务该通道。
   b. 如果外设或软件向控制器发出请求，则转到第1步。
3. 传输结束时，控制器在NVIC中产生对应的DMA通道中断。

自动请求(CHNL_CFG[2:0] = 010)

当控制器工作在此模式下时，控制器只需要收到一个请求以使能它便可完成整个DMA周期。这样可以进行大量数据传输而不会显著增加服务更高优先级请求的延迟时间，或者要求处理器或外设提供多个请求。这种模式对存储器到存储器复制应用非常有用。

自动请求不适合外设使用。

在此模式下，控制器可配置为使用主要数据结构或备选数据结构。通道使能之后，当控制器收到请求时，它便执行如下操作：

1. 控制器为该通道执行$\min(2^{R_POWER}, N)$次传输，其中R_POWER为控制数据配置寄存器的位[17:14]，N为传输次数。如果剩余传输次数为0，则转到第3步。
2. 自动产生该通道的请求。控制器仲裁。如果该通道具有最高优先级，则DMA周期转到第1步。
3. 传输结束时，控制器为对应的DMA通道产生中断。

乒乓式(CHNL_CFG[2:0] = 011)

在乒乓模式下，控制器利用其中一个数据结构执行一个DMA周期，然后利用另一个数据结构执行一个DMA周期。控制器不停地交替使用主要数据结构和备选数据结构，直至读取到无效数据结构或者主机处理器禁用该通道。

这种模式适合于利用存储器中的不同缓冲器来将数据从外设传输到存储器。在典型应用中，启动传输之前，主机必须同时配置主要数据结构和备选数据结构。传输进行时，主机随后可以在对应的传输结束时，在中断服务例程中配置主要或备选控制数据结构。

完成与各控制数据结构相关的传输之后，DMA控制器中断处理器。利用主要或备选控制数据结构的各次传输的工作方式与基本DMA传输完全相同。

存储器分散/聚集式(CHNL_CFG[2:0] = 100或101)

在存储器分散/聚集模式下，控制器必须配置为同时使用主要数据结构和备选数据结构。控制器利用主要数据结构设置备选数据结构的控制配置。备选数据结构用于实际数据传输，其与自动请求DMA传输类似。每次主要传输完成后，控制器执行仲裁。控制器只需一个请求就能完成全部传输。这种模式用于执行多个存储器到存储器复制任务的场合。处理器可以同时配置所有任务，无需在任务之间进行干预。当利用基本周期完成全部分散/聚集处理时，控制器在NVIC中产生对应的DMA通道中断。

这种模式下，控制器接收到初始请求后，利用主要数据结构执行4次DMA传输，以设置备选数据结构的控制结构。完成此传输之后，控制器利用备选数据结构启动DMA周期。完成该周期之后，控制器再次利用主要数据结构执行4个DMA周期。控制器不停地交替使用主要数据结构和备选数据结构，直至处理器将备选数据结构配置为基本周期或者DMA读取到无效数据结构。

表66列出了主要数据结构的CHNL_CFG存储器位置字段；对于存储器分散/聚集模式，这些字段必须写入定值。

表66. 存储器分散/聚集模式下主要数据结构的CHNL_CFG，CHNL_CFG[2:0] = 100

位	名称	描述
31:30	DST_INC	10: 配置控制器对地址使用字增量。
29:28	DST_SIZE	10: 配置控制器使用字转换。
27:26	SRC_INC	10: 配置控制器对地址使用字增量。
25:24	SRC_SIZE	10: 配置控制器使用字转换。
23:18	保留	未定义。写入0。
17:14	R_POWER	0010: 表示DMA控制器要执行4次传输。
13: 4	N_MINUS_1	配置控制器执行N次DMA传输，其中N为4的倍数。
3	保留	未定义。写入0。
2:0	CYCLE_CTRL	100: 配置控制器执行存储器分散/聚集DMA周期。

外设分散/聚集式(CHNL_CFG[2:0] = 110或111)

在外设分散/聚集模式下，控制器必须配置为同时使用主要数据结构和备选数据结构。控制器利用主要数据结构设置备选数据结构的控制结构。备选数据结构用于实际数据传输，每次传输采用备选数据结构和基本DMA传输进行。每次主要传输完成后，控制器不执行仲裁。这种模式用于要执行多个外设到存储器DMA任务的场合。Cortex-M3可以同时配置所有任务，无需在任务之间进行干预。除了仲裁和请求要求之外，这与存储器分散/聚集模式非常相似。当利用基本周期完成全部分散/聚集处理时，控制器在NVIC中产生对应的DMA通道中断。

在外设分散/聚集模式下，控制器从外设接收到初始请求后，利用主要数据结构执行4次DMA传输，以设置备选控制数据结构。然后，控制器利用备选数据结构立即启动DMA周期，而不重新仲裁。

完成此周期之后，控制器重新仲裁；如果其从具有最高优先级的外设收到请求，则利用主要数据结构再次执行4次DMA传输。然后，控制器利用备选数据结构立即启动DMA周期，而不重新仲裁。控制器不停地交替使用主要数据结构和备选数据结构，直至处理器将备选数据结构配置为基本周期或者DMA读取到无效数据结构。

表67列出了主要数据结构的CHNL_CFG存储器位置字段；对于外设分散/聚集模式，这些字段必须写入定值。

表67. 外设分散/聚集模式下主要数据结构的CHNL_CFG，CHNL_CFG[2:0] = 110

位	名称	描述
31:30	DST_INC	10: 配置控制器对地址使用字增量。
29:28	DST_SIZE	10: 配置控制器使用字转换。
27:26	SRC_INC	10: 配置控制器对地址使用字增量。
25:24	SRC_SIZE	10: 配置控制器使用字转换。
23:18	保留	未定义。写入0。
17:14	R_POWER	0010: 表示DMA控制器执行了4次传输而不重新仲裁。
13:4	N_MINUS_1	配置控制器执行N次DMA传输，其中N为4的倍数。
3	保留	未定义。写入0。
2:0	CYCLE_CTRL	110: 配置控制器执行存储器分散/聚集DMA周期。

地址计算

DMA控制器根据SRC_END_PTR的内容、CHNL_CFG中的来源地址增量设置和N_MINUS_1的当前值(CHNL_CFG[13:4])计算来源读取地址。

同样，目标写入地址根据DST_END_PTR的内容、CHNL_CFG中的目标地址增量设置和N_MINUS_1的当前值(CHNL_CFG[13:4])来计算。

SRC_INC = 0、1、2时：来源读取地址 = SRC_END_PTR − (N_MINUS_1 << (SRC_INC))

SRC_INC = 3时：来源读取地址 = SRC_END_PTR

DST_INC = 0、1、2时：目标写入地址 = DST_END_PTR − (N_MINUS_1 << (DST_INC))

DST_INC = 3时：目标写入地址 = DST_END_PTR

其中，N_MINUS_1为针对该通道配置的传输次数减1。

示例代码：配置DMA控制器

```
typedef struct
{
   unsigned int cycle_ctrl    :3;
   unsigned int next_useburst :1;
   unsigned int n_minus_1     :10;
   unsigned int r_power       :4;
   unsigned int src_prot_ctrl :3;
   unsigned int dst_prot_ctrl :3;
   unsigned int src_size      :2;
   unsigned int src_inc       :2;
   unsigned int dst_size      :2;
   unsigned int dst_inc       :2;
} CtrlCfgBits;

                                                    // Define the structure of a DMA
descriptor.
typedef struct dmaDesc
{
   unsigned int srcEndPtr;
   unsigned int destEndPtr;
   union
   {
      unsigned int   ctrlCfgVal;
      CtrlCfgBits    Bits;
   } ctrlCfg ;
   unsigned int reserved4Bytes;
} DmaDesc;

DmaDesc * Dma_GetDescriptor(unsigned int iChan,boolean bAlternate)
{
    if (iChan < DMACHAN_NUM)
        {
        if (bAlternate)
            return &(dmaChanDesc[iChan + 12]);       //DMA_ALT_OFFSET]);
        else
            return &(dmaChanDesc[iChan ]);
        }
    else
        return 0x0;
}
void Dma_Init(void)
{
                                                    // Clear all the DMA descriptors
(individual blocks will update their
                                                    // descriptors
   memset(dmaChanDesc,0x0,sizeof(dmaChanDesc));
```

```
                                                    // Setup the DMA base address pointer
register.
   pADI_DMA->DMAPDBPTR = (unsigned int) &dmaChanDesc;

                                                    // Enable the  uDMA (WO reg)
   pADI_DMA->DMACFG = 0x1;
}
void DACDMAINIT( void)
{
pADI_DAC->DACCON = 0x410;                           // Enable DAC, 0-INT_VREF range, timer 1
triggered
Dma_Init();                                         // Init DMA structures
DACDMAWRITE(uxDACDMA, 16)                           // Setup DMA control registers
NVIC_EnableIRQ(DMA_DAC_IRQn);                       // Enable DMA DAC Interrupt
pADI_DMA->DMACFG = 0x1;                             // Enable DMA mode in DMA controller
pADI_DMA->DMAENSET = 0x100;                         // Select DAC channel in DMA controller
pADI_TM1->LD = 0x00000088;                          // the min clock number are 0x17 – Timer 1
overflow to trigger DAC write
pADI_TM1->CON = 0x00000018;
}

void DACDMAWRITE(unsigned long *pucTX_DMA, unsigned int iNumRX)
{
   DmaDesc Desc;

                                                    // Common configuration of all the
descriptors used here
    desc.ctrlcfg.bits.cycle_ctrl       = cyclectrl_basic;
    desc.ctrlcfg.bits.next_useburst    = 0x0;
    desc.ctrlcfg.bits.r_power          = 0;
    Desc.ctrlCfg.Bits.src_prot_ctrl    = 0x0;
    Desc.ctrlCfg.Bits.dst_prot_ctrl    = 0x0;
    Desc.ctrlCfg.Bits.src_size         = SRCSIZE_WORD;
    Desc.ctrlCfg.Bits.dst_size         = DSTSIZE_WORD;

                                                    // TX Primary Descriptor
Desc.srcEndPtr                          = (unsigned long)(pucTX_DMA + iNumRX - 0x1);
    Desc.destEndPtr                     = (unsigned long)&DACDAT;
    Desc.ctrlCfg.Bits.n_minus_1         = iNumRX - 0x1;
    Desc.ctrlCfg.Bits.src_inc           = INC_WORD;
    Desc.ctrlCfg.Bits.dst_inc           = INC_NO;
    *Dma_GetDescriptor(DMA_DAC_REQ_OUT,FALSE) = Desc;
}
```

DMA存储器映射寄存器

表68. DMA控制器存储器映射寄存器地址表(基地址：0x40010000)

地址	名称	描述	访问类型	默认值
0x40010000	DMASTA	状态寄存器	R	0x000B0000
0x40010004	DMACFG	配置寄存器	RW	0x00000000
0x40010008	DMAPDBPTR	主要控制数据基地址指针寄存器	RW	0x00000000
0x4001000C	DMAADBPTR	备选控制数据基地址指针寄存器	R	0x00000100
0x40010014	DMASWREQ	软件请求寄存器	R	0x00000000
0x40010020	DMARMSKSET	请求屏蔽设置寄存器	RW	0x00000000
0x40010024	DMARMSKCLR	请求屏蔽清零寄存器	R	0x00000000
0x40010028	DMAENSET	使能设置寄存器	RW	0x00000000
0x4001002C	DMAENCLR	使能清零寄存器	W	0x00000000
0x40010030	DMAALTSET	主要-备选设置寄存器	RW	0x00000000
0x40010034	DMAALTCLR	主要-备选清除寄存器	W	0x00000000
0x40010038	DMAPRISET	优先级设置寄存器	RW	0x00000000
0x4001003C	DMAPRICLR	优先级清零寄存器	W	0x00000000
0x4001004C	DMAERRCLR	总线错误清零寄存器	RW	0x00000000
0x40010FD0	DMAPERID4	DMA外设ID 4寄存器	R	0x00000004
0x40010FE0	DMAPERID0	DMA外设ID 0寄存器	R	0x00000030
0x40010FE4	DMAPERID1	DMA外设ID 1寄存器	R	0x000000B2
0x40010FE8	DMAPERID2	DMA外设ID 2寄存器	R	0x0000000B
0x40010FEC	DMAPERID3	DMA外设ID 3寄存器	R	0x00000000
0x40010FF0	DMAPCELLID0	DMA PrimeCell ID 0寄存器	R	0x0000000D
0x40010FF4	DMAPCELLID1	DMA PrimeCell ID 1寄存器	R	0x000000F0
0x40010FF8	DMAPCELLID2	DMA PrimeCell ID 2寄存器	R	0x00000005
0x40010FFC	DMAPCELLID3	DMA PrimeCell ID 3寄存器	R	0x0000000B1

DMA状态寄存器

地址：0x40010000；复位：0x000B0000；名称：DMASTA

表69. DMASTA寄存器位功能描述

位	名称	描述
31:21	保留	保留。
20:16	CHNLSMINUS1	可用DMA通道数减1。例如，若有12个通道可用，寄存器的这些位将读出0xB。
15:8	保留	保留。未定义。
7:4	STATE	DMA控制器状态机的当前状态。读取此寄存器可了解DMA执行的操作。 0000: IDLE: 空闲。 0001: RDCHNLDATA: 读取通道控制器数据。 0010: RDSRCENDPTR: 读取来源数据端指针。 0011: RDDSTENDPTR: 读取目标端指针。 0100: RDSRCDATA: 读取来源数据。 0101: WRDSTDATA: 写入目标数据。 0110: WAITDMAREQCLR: 等待DMA请求清零。 0111: WRCHNLDATA: 写入通道控制器数据。 1000: STALLED: 停止。 1001: DONE: 完成。 1010: SCATRGATHR: 外设分散/聚集过渡。 1011到1111: 保留。
3:1	保留	保留。未定义。
0	ENABLE	控制器使能状态。 0: 控制器已禁用。 1: 控制器已使能。

DMA配置寄存器

地址：0x40010004；复位：0x00000000；名称：DMACFG

表70. DMACFG寄存器位功能描述

位	名称	描述
31:1	保留	保留。
0	ENABLE	控制器使能。 0: 控制器已禁用。 1: 控制器已使能。

DMA主要控制数据基地址指针寄存器

地址：0x40010008；复位：0x00000000；名称：DMAPDBPTR

表71. DMAPDBPTR寄存器位功能描述

位	名称	描述
31:0	CTRLBASEPTR	指向主要数据结构基址的指针。注意：5 + log(2) M LSB被保留，必须写入0；M为通道数。 DMAPDBPTR寄存器必须设置为指向系统存储器中的主要通道控制基地址指针。 必须赋予DMA控制器的系统存储器量取决于所用的DMA通道数以及是否使用备选通道 控制数据结构。当DMA控制器处于复位状态时，无法读取此寄存器。

DMA备选控制数据基地址指针寄存器

地址：0x4001000C；复位：0x00000100；名称：DMAADBPTR

表72. DMAADBPTR寄存器位功能描述

位	名称	描述
31:0	ALTCBPTR	备选数据结构的基址。DMAADBPTR只读寄存器返回备选通道控制数据结构的基址。此寄存器使得应用软件无需计算备选数据结构的基址。当DMA控制器处于复位状态时，无法读取此寄存器。

DMA软件请求寄存器

地址： 0x40010014；复位： 0x00000000；名称： DMASWREQ

当DMA通道未被外设使用时，DMA软件请求寄存器用于存储器到存储器DMA传输。它用于触发DMA传输。将适当的位置1即可在对应的DMA通道上产生软件DMA请求。完成对应的软件请求之后，这些位自动由硬件清零。

表73. DMASWREQ寄存器位功能描述

位	名称	描述
31:12	保留	保留。读回0。
11	SINC	0: 不产生sinc输出步进检测的DMA请求。 1: 产生sinc输出步进检测的DMA请求。
10	ADC1	0: 不产生ADC1的DMA请求。 1: 产生ADC1的DMA请求。
9	ADC0	0: 不产生ADC0的DMA请求(仅对ADuCM360有效)。 1: 产生ADC0的DMA请求(仅对ADuCM360有效)。
8	DAC	0: 不产生DAC DMA输出的DMA请求。 1: 产生DAC DMA输出的DMA请求。
7	I2CMRX	0: 不产生I2CMRX的DMA请求。 1: 产生I2CMRX的DMA请求。
6	I2CMTX	0: 不产生I2CMTX的DMA请求。 1: 产生I2CMTX的DMA请求。
5	I2CSRX	0: 不产生I2CSRX的DMA请求。 1: 产生I2CSRX的DMA请求。
4	I2CSTX	0: 不产生I2CSTX的DMA请求。 1: 产生I2CSTX的DMA请求。
3	UARTRX	0: 不产生UARTRX的DMA请求。 1: 产生UARTRX的DMA请求。
2	UARTTX	0: 不产生UARTTX的DMA请求。 1: 产生UARTTX的DMA请求。
1	SPI1RX	0: 不产生SPI1RX的DMA请求。 1: 产生SPI1RX的DMA请求。
0	SPI1TX	0: 不产生SPI1TX的DMA请求。 1: 产生SPI1TX的DMA请求。

DMA请求屏蔽设置寄存器

地址：0x40010020；复位：0x00000000；名称：DMARMSKSET

此寄存器禁用外设的DMA请求。该寄存器的每一位代表DMA控制器中的对应通道。将适当的位置1即可屏蔽对应DMA通道的请求。

表74. DMARMSKSET寄存器位功能描述

位	名称	描述
31:12	保留	保留。读回0。
11	SINC2	读取时： 0: 使能sinc2的请求。 1: 禁用sinc2的请求。 写入时： 0: 不起作用。使用DMARMSKCLR寄存器使能DMA请求。 1: 禁用与I2CMRX相关的外设产生DMA请求。
10	ADC1	读取时： 0: 使能ADC1的请求。 1: 禁用ADC1的请求。 写入时： 0: 不起作用。使用DMARMSKCLR寄存器使能DMA请求。 1: 禁用与ADC1相关的外设产生DMA请求。
9	ADC0	读取时： 0: 使能ADC0的请求(仅对ADuCM360有效)。 1: 禁用ADC0的请求(仅对ADuCM360有效)。 写入时： 0: 不起作用。使用DMARMSKCLR寄存器使能DMA请求(仅对ADuCM360有效)。 1: 禁用与ADC0相关的外设产生DMA请求(仅对ADuCM360有效)。
8	DAC	读取时： 0: 使能DAC的请求。 1: 禁用DAC的请求。 写入时： 0: 不起作用。使用DMARMSKCLR寄存器使能DMA请求。 1: 禁用与DAC相关的外设产生DMA请求。
7	I2CMRX	读取时： 0: 使能I2CMRX的请求。 1: 禁用I2CMRX的请求。 写入时： 0: 不起作用。使用DMARMSKCLR寄存器使能DMA请求。 1: 禁用与I2CMRX相关的外设产生DMA请求。
6	I2CMTX	读取时： 0: 使能I2CMTX的请求。 1: 禁用I2CMTX的请求。 写入时： 1: 禁用与I2CMTX相关的外设产生DMA请求。 0: 不起作用。使用DMARMSKCLR寄存器使能DMA请求。
5	I2CSRX	读取时： 0: 使能I2CSRX的请求。 1: 禁用I2CSRX的请求。 写入时： 0: 不起作用。使用DMARMSKCLR寄存器使能DMA请求。 1: 禁用与I2CSRX相关的外设产生DMA请求。

位	名称	描述
4	I2CSTX	读取时： 0: 使能I2CSTX的请求。 1: 禁用I2CSTX的请求。 写入时： 0: 不起作用。使用DMARMSKCLR寄存器使能DMA请求。 1: 禁用与I2CSTX相关的外设产生DMA请求。
3	UARTRX	读取时： 0: 使能UARTRX的请求。 1: 禁用UARTRX的请求。 写入时： 0: 不起作用。使用DMARMSKCLR寄存器使能DMA请求。 1: 禁用与UARTRX相关的外设产生DMA请求。
2	UARTTX	读取时： 0: 使能UARTTX的请求。 1: 禁用UARTTX的请求。 写入时： 0: 不起作用。使用DMARMSKCLR寄存器使能DMA请求。 1: 禁用与UARTTX相关的外设产生DMA请求。
1	SPI1RX	读取时： 0: 使能SPI1RX的请求。 1: 禁用SPI1RX的请求。 写入时： 0: 不起作用。使用DMARMSKCLR寄存器使能DMA请求。 1: 禁用与SPI1RX相关的外设产生DMA请求。
0	SPI1TX	读取时： 0: 使能SPI1TX的请求。 1: 禁用SPI1TX的请求。 写入时： 0: 不起作用。使用DMARMSKCLR寄存器使能DMA请求。 1: 禁用与SPI1TX相关的外设产生DMA请求。

DMA请求屏蔽清零寄存器

地址：0x40010024；复位：0x00000000；名称：DMARMSKCLR

此寄存器通过清除DMARMSKSET寄存器设置的屏蔽来使能外设的DMA请求。该寄存器的每一位代表DMA控制器中的对应通道。

将适当的位置1即可将DMARMSKSET寄存器中对应的位清0。

表75. DMARMSKCLR寄存器位功能描述

位	名称	描述
31:12	保留	保留。
11	SINC2	0: 不起作用。使用DMARMSKSET寄存器禁用DMA请求。 1: 使能与sinc2相关的外设产生DMA请求。
10	ADC1	0: 不起作用。使用DMARMSKSET寄存器禁用DMA请求。 1: 使能与ADC1相关的外设产生DMA请求。
9	ADC0	0: 不起作用。使用DMARMSKSET寄存器禁用DMA请求(仅对ADuCM360有效)。 1: 使能与ADC0相关的外设产生DMA请求(仅对ADuCM360有效)。
8	DAC	0: 不起作用。使用DMARMSKSET寄存器禁用DMA请求。 1: 使能与DAC相关的外设产生DMA请求。
7	I2CMRX	0: 不起作用。使用DMARMSKSET寄存器禁用DMA请求。 1: 使能与I2CMRX相关的外设产生DMA请求。
6	I2CMTX	0: 不起作用。使用DMARMSKSET寄存器禁用DMA请求。 1: 使能与I2CMTX相关的外设产生DMA请求。

位	名称	描述
5	I2CSRX	0: 不起作用。使用DMARMSKSET寄存器禁用DMA请求。 1: 使能与I2CSRX相关的外设产生DMA请求。
4	I2CSTX	0: 不起作用。使用DMARMSKSET寄存器禁用DMA请求。 1: 使能与I2CSTX相关的外设产生DMA请求。
3	UARTRX	0: 不起作用。使用DMARMSKSET寄存器禁用DMA请求。 1: 使能与UARTRX相关的外设产生DMA请求。
2	UARTTX	0: 不起作用。使用DMARMSKSET寄存器禁用DMA请求。 1: 使能与UARTTX相关的外设产生DMA请求。
1	SPI1RX	0: 不起作用。使用DMARMSKSET寄存器禁用DMA请求。 1: 使能与SPI1RX相关的外设产生DMA请求。
0	SPI1TX	0: 不起作用。使用DMARMSKSET寄存器禁用DMA请求。 1: 使能与SPI1TX相关的外设产生DMA请求。

DMA使能设置寄存器

地址：0x40010028；复位：0x00000000；名称：DMAENSET

使能DMA通道。

此寄存器用于使能DMA通道。读取该寄存器将返回通道的使能状态。该寄存器的每一位代表DMA控制器中的对应通道。将适当的位置1即可使能对应的通道。

表76. DMAENSET寄存器位功能描述

位	名称	描述
31:12	保留	保留
11	SINC2	0: 不起作用。使用DMAENCLR寄存器禁用该通道。 1: 使能sinc2 DMA通道。
10	ADC1	0: 不起作用。使用DMAENCLR寄存器禁用该通道。 1: 使能ADC1 DMA通道。
9	ADC0	0: 不起作用。使用DMAENCLR寄存器禁用该通道(仅对ADuCM360有效)。 1: 使能ADC0 DMA通道(仅对ADuCM360有效)。
8	DAC	0: 不起作用。使用DMAENCLR寄存器禁用该通道。 1: 使能DAC DMA通道。
7	I2CMRX	0: 不起作用。使用DMAENCLR寄存器禁用该通道。 1: 使能I2CMRX DMA通道。
6	I2CMTX	0: 不起作用。使用DMAENCLR寄存器禁用该通道。 1: 使能I2CMTX DMA通道。
5	I2CSRX	0: 不起作用。使用DMAENCLR寄存器禁用该通道。 1: 使能I2CSRX DMA通道。
4	I2CSTX	0: 不起作用。使用DMAENCLR寄存器禁用该通道。 1: 使能I2CSTX DMA通道。
3	UARTRX	0: 不起作用。使用DMAENCLR寄存器禁用该通道。 1: 使能UARTRX DMA通道。
2	UARTTX	0: 不起作用。使用DMAENCLR寄存器禁用该通道。 1: 使能UARTTX DMA通道。
1	SPI1RX	0: 不起作用。使用DMAENCLR寄存器禁用该通道。 1: 使能SPI1RX DMA通道。
0	SPI1TX	0: 不起作用。使用DMAENCLR寄存器禁用该通道。 1: 使能SPI1TX DMA通道。

DMA使能清零寄存器

地址：0x4001002C；复位：0x00000000；名称：DMAENCLR

禁用DMA通道。

此寄存器用于禁用DMA通道。读取该寄存器将返回通道的使能状态。该寄存器的每一位代表DMA控制器中的对应通道。

注意：当控制器完成DMA周期时，它会自动禁用通道，即把相应的位置1将适当的位置1即可禁用对应的通道。

表77. DMAENCLR寄存器位功能描述

位	名称	描述
31:12	保留	
11	SINC2	0: 不起作用。使用DMAENSET寄存器使能该通道。 1: 禁用sinc2 DMA通道。
10	ADC1	0: 不起作用。使用DMAENSET寄存器使能该通道。 1: 禁用ADC1 DMA通道。
9	ADC0	0: 不起作用。使用DMAENSET寄存器使能该通道(仅对ADuCM360有效)。 1: 禁用ADC0 DMA通道(仅对ADuCM360有效)。
8	DAC	0: 不起作用。使用DMAENSET寄存器使能该通道。 1: 禁用DAC DMA通道。
7	I2CMRX	0: 不起作用。使用DMAENSET寄存器使能该通道。 1: 禁用I2CMRX DMA通道。
6	I2CMTX	0: 不起作用。使用DMAENSET寄存器使能该通道。 1: 禁用I2CMTX DMA通道。
5	I2CSRX	0: 不起作用。使用DMAENSET寄存器使能该通道。 1: 禁用I2CSRX DMA通道。
4	I2CSTX	0: 不起作用。使用DMAENSET寄存器使能该通道。 1: 禁用I2CSTX DMA通道。
3	UARTRX	0: 不起作用。使用DMAENSET寄存器使能该通道。 1: 禁用UARTRX DMA通道。
2	UARTTX	0: 不起作用。使用DMAENSET寄存器使能该通道。 1: 禁用UARTTX DMA通道。
1	SPI1RX	0: 不起作用。使用DMAENSET寄存器使能该通道。 1: 禁用SPI1RX DMA通道。
0	SPI1TX	0: 不起作用。使用DMAENSET寄存器使能该通道。 1: 禁用SPI1TX DMA通道。

DMA主要/备选设置寄存器

地址：0x40010030；复位：0x00000000；名称：DMAALTSET

控制结构状态/选择备选结构。

返回通道控制数据结构状态，或者为对应DMA通道选择备选数据结构。

注意：对于乒乓式、存储器分散/聚集式和外设分散/聚集式传输，DMA控制器会视需要自动将这些位置1/清0。

表78. DMAALTSET寄存器位功能描述

位	名称	描述
31:12	保留	
11	SINC2	读取时： 0: DMA sinc2正在使用主要数据结构。 1: DMA sinc2正在使用备选数据结构。 写入时： 0: 不起作用。使用DMAALTCLR寄存器将sinc2设为0。 1: 为sinc2选择备选数据结构。
10	ADC1	读取时： 0: DMA ADC1正在使用主要数据结构。 1: DMA ADC1正在使用备选数据结构。 写入时： 0: 不起作用。使用DMAALTCLR寄存器将ADC1设为0。 1: 为ADC1选择备选数据结构。
9	ADC0	读取时： 0: DMA ADC0正在使用主要数据结构(仅对ADuCM360有效)。 1: DMA ADC0正在使用备选数据结构(仅对ADuCM360有效)。 写入时： 0: 不起作用。使用DMAALTCLR寄存器将ADC0设为0(仅对ADuCM360有效)。 1: 为ADC0选择备选数据结构(仅对ADuCM360有效)。
8	DAC	读取时： 0: DMA DAC正在使用主要数据结构。 1: DMA DAC正在使用备选数据结构。 写入时： 0: 不起作用。使用DMAALTCLR寄存器将DAC设为0。 1: 为DAC选择备选数据结构。
7	I2CMRX	读取时： 0: DMA I2CMRX正在使用主要数据结构。 1: DMA I2CMRX正在使用备选数据结构。 写入时： 0: 不起作用。使用DMAALTCLR寄存器将I2CMRX设为0。 1: 为I2CMRX选择备选数据结构。
6	I2CMTX	读取时： 0: DMA I2CMTX正在使用主要数据结构。 1: DMA I2CMTX正在使用备选数据结构。 写入时： 0: 不起作用。使用DMAALTCLR寄存器将I2CMTX设为0。 1: 为I2CMTX选择备选数据结构。
5	I2CSRX	读取时： 0: DMA I2CSRX正在使用主要数据结构。 1: DMA I2CSRX正在使用备选数据结构。 写入时： 0: 不起作用。使用DMAALTCLR寄存器将I2CSRX设为0。 1: 为I2CSRX选择备选数据结构。

位	名称	描述
4	I2CSTX	读取时： 0: DMA I2CSTX正在使用主要数据结构。 1: DMA I2CSTX正在使用备选数据结构。 写入时： 0: 不起作用。使用DMAALTCLR寄存器将I2CSTX设为0。 1: 为I2CSTX选择备选数据结构。
3	UARTRX	读取时： 0: DMA UARTRX正在使用主要数据结构。 1: DMA UARTRX正在使用备选数据结构。 写入时： 0: 不起作用。使用DMAALTCLR寄存器将UARTRX设为0。 1: 为UARTRX选择备选数据结构。
2	UARTTX	读取时： 0: DMA UARTTX正在使用主要数据结构。 1: DMA UARTTX正在使用备选数据结构。 写入时： 0: 不起作用。使用DMAALTCLR寄存器将UARTTX设为0。 1: 为UARTTX选择备选数据结构。
1	SPI1RX	读取时： 0: DMA SPI1RX正在使用主要数据结构。 1: DMA SPI1RX正在使用备选数据结构。 写入时： 0: 不起作用。使用DMAALTCLR寄存器将SPI1RX设为0。 1: 为SPI1RX选择备选数据结构。
0	SPI1TX	读取时： 0: DMA SPI1TX正在使用主要数据结构。 1: DMA SPI1TX正在使用备选数据结构。 写入时： 0: 不起作用。使用DMAALTCLR寄存器将SPI1TX设为0。 1: 为SPI1TX选择备选数据结构。

DMA主要/备选清零寄存器

地址：0x40010034；复位：0x00000000；名称：DMAALTCLR

选择主要数据结构。

将适当的位置1即可为对应DMA通道选择主要数据结构。

注意：对于乒乓式、存储器分散/聚集式和外设分散/聚集式传输，DMA控制器会视需要自动将这些位置1/清0。

表79. DMAALTCLR寄存器位功能描述

位	名称	描述
31:12	保留	
11	SINC2	0: 不起作用。使用DMAALTSET寄存器选择备选数据结构。 1: 为sinc2选择主要数据结构。
10	ADC1	0: 不起作用。使用DMAALTSET寄存器选择备选数据结构。 1: 为ADC1选择主要数据结构。
9	ADC0	0: 不起作用。使用DMAALTSET寄存器选择备选数据结构(仅对ADuCM360有效)。 1: 为ADC0选择主要数据结构(仅对ADuCM360有效)。
8	DAC	0: 不起作用。使用DMAALTSET寄存器选择备选数据结构。 1: 为DAC选择主要数据结构。
7	I2CMRX	0: 不起作用。使用DMAALTSET寄存器选择备选数据结构。 1: 为I2CMRX选择主要数据结构。

位	名称	描述
6	I2CMTX	0: 不起作用。使用DMAALTSET寄存器选择备选数据结构。 1: 为I2CMTX选择主要数据结构。
5	I2CSRX	0: 不起作用。使用DMAALTSET寄存器选择备选数据结构。 1: 为I2CSRX选择主要数据结构。
4	I2CSTX	0: 不起作用。使用DMAALTSET寄存器选择备选数据结构。 1: 为I2CSTX选择主要数据结构。
3	UARTRX	0: 不起作用。使用DMAALTSET寄存器选择备选数据结构。 1: 为UARTRX选择主要数据结构。
2	UARTTX	0: 不起作用。使用DMAALTSET寄存器选择备选数据结构。 1: 为UARTTX选择主要数据结构。
1	SPI1RX	0: 不起作用。使用DMAALTSET寄存器选择备选数据结构。 1: 为SPI1RX选择主要数据结构。
0	SPI1TX	0: 不起作用。使用DMAALTSET寄存器选择备选数据结构。 1: 为SPI1TX选择主要数据结构。

DMA优先级设置寄存器

地址：0x40010038；复位：0x00000000；名称：DMAPRISET

将通道配置为高优先级。

此寄存器用于将一个DMA通道配置为使用高优先级。

读取该寄存器将返回通道优先级屏蔽的状态。该寄存器的每一位代表DMA控制器中的对应通道。

返回通道优先级屏蔽状态，或者将通道优先级设置为高。

表80. DMAPRISET寄存器位功能描述

位	名称	描述
31:12	保留	
11	SINC2	读取时： 0: DMA sinc2正在使用默认优先级。 1: DMA sinc2正在使用高优先级。 写入时： 0: 不起作用。使用DMAPRICLR寄存器将sinc2设为默认优先级。 1: sinc2使用高优先级。
10	ADC1	读取时： 0: DMA ADC1正在使用默认优先级。 1: DMA ADC1正在使用高优先级。 写入时： 0: 不起作用。使用DMAPRICLR寄存器将ADC1设为默认优先级。 1: ADC1使用高优先级。
9	ADC0	读取时： 0: DMA ADC0正在使用默认优先级(仅对ADuCM360有效)。 1: DMA ADC0正在使用高优先级(仅对ADuCM360有效)。 写入时： 0: 不起作用。使用DMAPRICLR寄存器将ADC0设为默认优先级(仅对ADuCM360有效)。 1: ADC0使用高优先级(仅对ADuCM360有效)。
8	DAC	读取时： 0: DMA DAC正在使用默认优先级。 1: DMA DAC正在使用高优先级。 写入时： 0: 不起作用。使用DMAPRICLR寄存器将DAC设为默认优先级。 1: DAC使用高优先级。

位	名称	描述
7	I2CMRX	读取时： 0: DMA I2CMRX正在使用默认优先级。 1: DMA I2CMRX正在使用高优先级。 写入时： 0: 不起作用。使用DMAPRICLR寄存器将I2CMRX设为默认优先级。 1: I2CMRX使用高优先级。
6	I2CMTX	读取时： 0: DMA I2CMTX正在使用默认优先级。 1: DMA I2CMTX正在使用高优先级。 写入时： 0: 不起作用。使用DMAPRICLR寄存器将I2CMTX设为默认优先级。 1: I2CMTX使用高优先级。
5	I2CSRX	读取时： 0: DMA I2CSRX正在使用默认优先级。 1: DMA I2CSRX正在使用高优先级。 写入时： 0: 不起作用。使用DMAPRICLR寄存器将I2CSRX设为默认优先级。 1: I2CSRX使用高优先级。
4	I2CSTX	读取时： 0: DMA I2CSTX正在使用默认优先级。 1: DMA I2CSTX正在使用高优先级。 写入时： 0: 不起作用。使用DMAPRICLR寄存器将I2CSTX设为默认优先级。 1: I2CSTX使用高优先级。
3	UARTRX	读取时： 0: DMA UARTRX正在使用默认优先级。 1: DMA UARTRX正在使用高优先级。 写入时： 0: 不起作用。使用DMAPRICLR寄存器将UARTRX设为默认优先级。 1: UARTRX使用高优先级。
2	UARTTX	读取时： 0: DMA UARTTX正在使用默认优先级。 1: DMA UARTTX正在使用高优先级。 写入时： 0: 不起作用。使用DMAPRICLR寄存器将UARTTX设为默认优先级。 1: UARTTX使用高优先级。
1	SPI1RX	读取时： 0: DMA SPI1RX正在使用默认优先级。 1: DMA SPI1RX正在使用高优先级。 写入时： 0: 不起作用。使用DMAPRICLR寄存器将SPI1RX设为默认优先级。 1: SPI1RX使用高优先级。
0	SPI1TX	读取时： 0: DMA SPI1TX正在使用默认优先级。 1: DMA SPI1TX正在使用高优先级。 写入时： 0: 不起作用。使用DMAPRICLR寄存器将SPI1TX设为默认优先级。 1: SPI1TX使用高优先级。

DMA优先级清零寄存器

地址：0x4001003C；复位：0x00000000；名称：DMAPRICLR

将通道配置为默认优先级。DMAPRICLR只写寄存器用于将一个DMA通道配置为使用默认优先级。该寄存器的每一位代表DMA控制器中的对应通道。将适当的位置1即可为对应DMA通道选择默认优先级。

表81. DMAPRICLR寄存器位功能描述

位	名称	描述
31:12	保留	
11	SINC2	0: 不起作用。使用DMAPRISET寄存器将sinc2设为高优先级。 1: sinc2使用默认优先级。
10	ADC1	0: 不起作用。使用DMAPRISET寄存器将ADC1设为高优先级。 1: ADC1使用默认优先级。
9	ADC0	0: 不起作用。使用DMAPRISET寄存器将ADC0设为高优先级(仅对ADuCM360有效)。 1: ADC0使用默认优先级(仅对ADuCM360有效)。
8	DAC	0: 不起作用。使用DMAPRISET寄存器将DAC设为高优先级。 1: DAC使用默认优先级。
7	I2CMRX	0: 不起作用。使用DMAPRISET寄存器将I2CMRX设为高优先级。 1: I2CMRX使用默认优先级。
6	I2CMTX	0: 不起作用。使用DMAPRISET寄存器将I2CMTX设为高优先级。 1: I2CMTX使用默认优先级。
5	I2CSRX	0: 不起作用。使用DMAPRISET寄存器将I2CSRX设为高优先级。 1: I2CSRX使用默认优先级。
4	I2CSTX	0: 不起作用。使用DMAPRISET寄存器将I2CSTX设为高优先级。 1: I2CSTX使用默认优先级。
3	UARTRX	0: 不起作用。使用DMAPRISET寄存器将UARTRX设为高优先级。 1: UARTRX使用默认优先级。
2	UARTTX	0: 不起作用。使用DMAPRISET寄存器将UARTTX设为高优先级。 1: UARTTX使用默认优先级。
1	SPI1RX	0: 不起作用。使用DMAPRISET寄存器将SPI1RX设为高优先级。 1: SPI1RX使用默认优先级。
0	SPI1TX	0: 不起作用。使用DMAPRISET寄存器将SPI1TX设为高优先级。 1: SPI1TX使用默认优先级。

DMA总线错误清零寄存器

地址：0x4001004C；复位：0x00000000；名称：DMAERRCLR

表82. DMAERRCLR寄存器位功能描述

位	名称	描述
31:1	保留	保留。
0	ERROR	总线错误状态。此寄存器用于读取和清除DMA总线错误状态。 如果控制器在执行传输时遇到总线错误，就会设置错误状态。 如果一个通道发生总线错误，控制器会自动禁用该通道。 其它通道不受影响。写入1可将此位清0。 读取时： 0: 未发生总线错误。 1: 有一个总线错误待处理。 写入时： 0: 不起作用。 1: 位清0。

DMA外设ID 4寄存器

地址：0x40001FD0；复位：0x00000004；名称：DMAPERID4

表83. DMAPERID4寄存器位功能描述

位	名称	描述
31:14	保留	保留。
13:0	PID4	DMA外设ID 4。默认值为0x04。

DMA外设ID 0寄存器

地址：0x40001FE0；复位：0x00000030；名称：DMAPERID0

表84. DMAPERID0寄存器位功能描述

位	名称	描述
31:14	保留	保留。
13:0	PID0	DMA外设ID 0。默认值为0x30。

DMA外设ID 1寄存器

地址：0x40001FE4；复位：0x000000B2；名称：DMAPERID1

表85. DMAPERID1寄存器位功能描述

位	名称	描述
31:14	保留	保留。
13:0	PID1	DMA外设ID 1。默认值为0xB2。

DMA外设ID 2寄存器

地址： 0x40001FE8；复位： 0x0000000B；名称： DMAPERID2

表86. DMAPERID2寄存器位功能描述

位	名称	描述
31:14	保留	保留。
13:0	PID2	DMA外设ID 2。默认值为0x0B。

DMA外设ID 3寄存器

地址：0x40001FEC；复位：0x00000000；名称：DMAPERID3

表87. DMAPERID3寄存器位功能描述

位	名称	描述
31:14	保留	保留。
13:0	PID3	DMA外设ID 3。默认值为0x00。

DMA PrimeCell ID 0寄存器

地址：0x40001FF0；复位：0x0000000D；名称：DMAPCELLID0

表88. DMAPCELLID0寄存器位功能描述

位	名称	描述
31:14	保留	保留。
13:0	PCELLID0	DMA PrimeCell标识寄存器0。默认值为0x0D。

DMA PrimeCell ID 1寄存器

地址：0x40001FF4；复位：0x000000F0；名称：DMAPCELLID1

表89. DMAPCELLID1寄存器位功能描述

位	名称	描述
31:14	保留	保留。
13:0	PCELLID1	DMA PrimeCell标识寄存器1。默认值为0xF0。

DMA PrimeCell ID 2寄存器

地址：0x40001FF8；复位：0x00000005；名称：DMAPCELLID2

表90. DMAPCELLID2寄存器位功能描述

位	名称	描述
31:14	保留	保留。
13:0	PCELLID2	DMA PrimeCell标识寄存器2。默认值为0x05。

DMA PrimeCell ID 3寄存器

地址：0x40001FFC；复位：0x000000B1；名称：DMAPCELLID3

表91. DMAPCELLID3寄存器位功能描述

位	名称	描述
31:14	保留	保留。
13:0	PCELLID3	DMA PrimeCell标识寄存器3。默认值为0xB1。

闪存控制器

闪存控制器特性

- 128 kB Flash/EE
- 2 kB信息空间
- 闪存控制器

闪存控制器概述

ADuCM360/ADuCM361具有128 kB Flash/EE存储器、2 kB信息空间和一个闪存控制器。

读写闪存通过直接访问进行。

支持的命令

- 批量擦除和页面擦除。
- 产生单页或多页签名。
- 命令中止。

当一个命令正在执行中时，处理器对闪存的任何访问都会暂停，直至该命令完成。

闪存保护

- 用户空间的写保护。
- 能够锁定串行线接口以实现读保护。

闪存完整性

- 复位时自动检查内核空间的签名。
- 应用代码的用户签名。

闪存结构

ADuCM360/ADuCM361控制器支持128 kB用户闪存和2 kB信息空间。信息空间通过存储器映射在用户闪存空间上方，如图19所示。128 kB主空间分为32个闪存块。每个闪存块包含8个闪存页面，每个页面的大小为512字节。闪存在块段中受保护，但在页段中可进行擦除和写入操作。

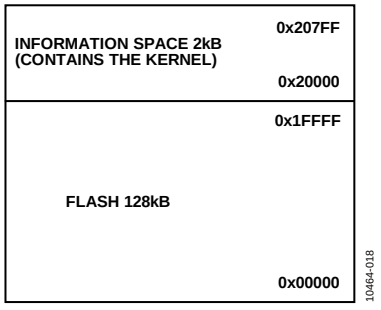


图19. ADuCM360/ADuCM361上的信息空间和用户空间存储器映射

用户空间

闪存最上面的20个字节如图20所示。 如果用户试图读取或写入不可用的存储器部分，就会返回总线错误。

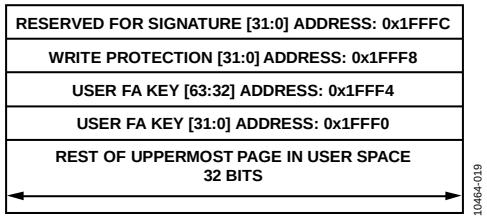


图20. 用户存储器最上面的页面

内核空间

内核空间保留用于测试数据，串行下载器受到读保护。

写入FLASH/EE存储器

写入闪存可直接进行，类似SRAM。要使能对闪存的写操作，用户代码必须先将FEECON0[2]置1。当FEESTA[3]指示闪存写序列已完成时，用户代码应将FEECON0[2]清0。配置这些位且禁用闪存保护之后，用户代码便可写入闪存。当闪存擦除或写操作正在进行时，Cortex-M3处理器与闪存断开连接，直至擦除/写操作完成为止。

注意：仅支持32位写操作，不支持16位和8位写操作。支持对闪存的突发写操作。因为只向闪存写入0，故而在必要时，可使用屏蔽来写入个别位或字节。在不擦除的情况下，一个位置只能写入两次。如果寻址的位置有写保护，则控制器返回硬故障系统异常错误。

如果一个写操作之后紧接着对闪存下一位置的写操作，而且其与上一写操作位于同一行，则闪存写入时间较快，因为控制器无需更换行地址。注意，只有存储多个汇编指令能够利用这一闪存特性。

对Flash/EE存储器执行写操作之前，应读取并检查FEESTA寄存器，确保没有其它命令或写操作正在执行。

示例代码：写入闪存

```
void WriteToFlash(unsigned long *pArray, unsigned long ulFlashPage, unsigned int uiSize)
{
 unsigned int uiPollFEESTA = 0;
 volatile unsigned long *ulStartAddress;
 unsigned int Size = 0;
 unsigned int n = 0;

 Size = uiSize;
 ulStartAddress = ( unsigned long      *)ulFlashPage;
pADI_FEE->FEECON0 |= 0x4;                                  // Enable a write to Flash memory
 uiPollFEESTA = pADI_FEE->FEESTA;                          // Read Status to ensure it is clear
 uiPollFEESTA = 0;
 for (n = 0; n < Size; n=n+4)
 {
  uiPollFEESTA = 0;
  *ulStartAddress  = *pArray;
  while ((uiPollFEESTA & 0x8) == 0x0)                      // Wait for Command to complete
  {
  uiPollFEESTA = pADI_FEE->FEESTA;
  }
  ulStartAddress = ulStartAddress++;
  pArray++;
 }
 pADI_FEE->FEECON0 &= 0xFB;                                // disable a write to Flash memory
}
```

擦除FLASH/EE存储器

用户代码可以调用2个闪存擦除命令：

- 批量擦除。 此命令擦除全部用户闪存。 将用户保护密钥输入FEEKEY寄存器之后，将批量擦除命令写入FEECMD。
- 页面擦除。此命令擦除FEEADR0L/FEEADR0H所选择的512字节闪存页面。将用户保护密钥输入FEEKEY寄存器之后，将要擦除的页面地址载入FEEADR0L/FEEADR0H寄存器。最后，将页面擦除命令写入FEECMD。CMDDONE (FEESTA[2])表示页面擦除命令已完成。

在页面或批量擦除操作序列期间，闪存控制器和闪存模块会消耗额外的电流。

示例代码：批量擦除闪存

```
pADI_FEE->FEEKEY = 0xF456;                                 // Enter Flash User Protection key

pADI_FEE->FEEKEY = 0xF123;
pADI_FEE->FEECMD = 0x3;                                    // Load Mass Erase value
```

闪存控制器性能和命令执行时间

闪存控制器性能和命令执行时间如下所示。这些闪存时间不依赖于时钟分频器。

- 直接单次写访问(32位位置)：46 µs
- 批量擦除：21 ms
- 页面擦除：21 ms
- 直接写入一页(128个32位位置)：3.04 ms
- 全部用户空间的批量验证：2.05 ms(128 kB和16 MHz HCLK)
- 全部用户空间的签名：2.05 ms(128 kB和16 MHz HCLK)

闪存保护

实现了三种保护类型：

- 密钥保护
- 读保护
- 写保护

闪存保护：密钥保护

某些闪存控制器寄存器受密钥保护，以免意外写入这些寄存器。

用户密钥为0xF123F456。它通过16位密钥寄存器输入，先输入0xF456，再输入0xF123。必须输入此密钥才能运行某些用户命令，写入闪存中的某些位置，或者使能对设置寄存器的写操作。输入后，密钥保持置位状态，除非向FEECMD寄存器写入一个命令。命令完成时，密钥自动清零。如果输入此密钥以使能对设置寄存器的写操作，或使能对闪存某些位置的写操作，则完成后需要由用户代码清零。要清除密钥，须向密钥寄存器写入任意16位值。

闪存保护： 用户读保护

用户空间读保护通过禁用串行线访问来实现，使得闪存内容无法啊通过调试或下载接口来读取。用户可以向DBG(FEECON1[0])写入0以禁用串行线访问。注意，写入FEECON1之前，必须加载FEEKEY寄存器。

内核正在运行时，串行线访问被禁用；否则，串行线访问可能会阻止内核运行完毕。内核完成运行后，会向闪存FEECON1寄存器的DBG写入1以使能串行线访问。

闪存保护：用户写保护

用户写保护旨在防止意外写入用户空间中的页面，以及在下载额外代码到闪存时保护用户代码块。如果检测到对受保护位置的写或擦除操作，ADuCM360/ADuCM361闪存控制器会产生一个异常。

写保护存储在用户空间的顶部。最高4字节保留用于签名，其后4字节用于写保护。复位后，写保护由闪存控制器上传到本地寄存器。上传后，32个写保护位可通过存储器映射寄存器FEEPRO[31:0]读取。要写入这些写保护位，用户必须向密钥寄存器先写入0xF456，再写入0xF123。写入写保护后，要重新写入必须完全擦除用户空间。批量擦除后，器件必须复位以上传的写保护位副本。

下面是写保护FEEPRO[31:0]位编程的操作序列：

1. 向密钥寄存器先写入0xF456，再写入0xF123。
2. 直接向闪存写入所需的写保护。写入0以使能保护。对于128 kB闪存，写保护地址为0x1FFF8。
3. 通过轮询状态寄存器FEESTA[3]，或者使能写完成中断，验证上述写操作已完成。
4. 复位器件，从内核空间上传写保护字段，并通过闪存控制器将其激活。

如果闪存中的写保护没有编程，即在上电时从闪存上传0xFFFFFFFF，则用户代码可直接写入FEEPRO寄存器。利用这个特性，用户可以先验证有无写保护，再将其运用于闪存。

如果闪存中的写保护已经编程，写保护的MSB必须为0，以防擦除写保护块。对于写保护，存储器分为32块，每块8页或4 kB(每页512字节)。若在事先没有设置FEEKEY的情况下尝试写入写保护字，则会产生总线错误(硬故障系统异常错误)。

闪存控制器故障分析密钥

尽管使能了读保护，但可能需要对用户退回的器件进行故障分析。有一种方法可通过用户故障分析密钥来对受保护存储器进行故障分析。

此密钥有64位，存储在闪存中用户空间的顶部，如图20所示。当串行线接口已被锁定时，它用于获取对用户代码的访问权。用户负责将此密钥设置为一个值。为使ADI公司能够访问用户代码，必须将该密钥提供给ADI公司。

闪存完整性签名特性

签名用于检查闪存的完整性。软件可以偶尔调用签名检测命令，或者每当要执行新的代码块时调用。签名是一种24位循环冗余检验(CRC)，采用多项式$x^{24} + x^{23} + x^6 + x^5 + x + 1$。

签名命令可用来产生一个签名，并检查一个代码块的签名，这里的块可以是单页或多页。使用一个24位LFSR来产生签名。硬件假定，一个模块的签名保存在其最高有效页的最高4字节中。因此，产生签名时不包括这4个字节。

使用以下步骤生成签名：

1. 把代码块的起始地址写入FEEADR0L/FEEADR0LH寄存器。
2. 把代码块的结束地址写入FEEADR1L/FEEADR1LH寄存器。
3. 把签名命令写入命令寄存器(FEECMD = 10)。

当命令执行完毕时，签名就在签名寄存器中。将该签名与保存在代码块最高有效页的上四个字节中的数据进行比较。如果数据与签名不匹配，则状态寄存器中返回故障状态(FEESTA[5:4] = 10)。

计算签名时，对闪存的所有其它访问都暂停。对于128 kB块，需要32k次读操作。

注意：FEEADR0L/FEEADR0H和FEEADR1L/FEEADR1H地址是字节地址，但仅需识别页面，因为硬件会忽略低9位。

注意，用户必须首次运行用户代码中的CRC多项式，以生成CRC值，然后把该值写入代码块最高页的最高四个字节中。该操作完成时，调用签名特性的任何操作都会把这个4字节的值与签名检测功能的结果进行比较。

内核完整性

硬件在复位后自动检测内核的完整性。出现故障时，FEESTA[6]置1，用户代码无法运行。只有在使能串行线接口时，该位才可通过串行线读操作读取。

中止使用中断

收到表55所列的中断时，可以中止命令(擦除、签名或批量验证)和写操作。向FEECMD寄存器写入中止命令也可以实现中止。然而，如果闪存正在编程并且控制编程的例程位于闪存中，则无法使用中止命令来中止周期，因为无法读取指令。故而提供了在任何系统中断置位时中止一个周期的功能。FEEAENx寄存器用于使能收到中断时中止。

通过系统中断中止一个命令或写操作时，FEESTA[5:4]会指示中止状态(FEESTA[5:4] = 11)。

要中止闪存写入或擦除周期，先必须等待闪存内核中的高压放电完毕。也就是说，写周期完成后必须等待6.6 µs，以便高压放电。擦除周期完成后必须等待6.6 µs。批量擦除周期完成后必须等待121 µs。

根据写周期在中止置位时所处的状态，写周期可能已完成或未完成。如果写入或擦除周期未成功完成，可在状态寄存器中读取到中止故障状态。

在擦除或编程周期中，若需立即响应中断，则必须将中断服务例程和中断矢量表移动到SRAM并保持该周期的持续时间。

如果设置DMA引擎将一个数据块写入闪存，则可以设置一个中断来中止当前写操作；不过，DMA引擎会立即开始下一个写操作。引起中止的中断保持置位状态，因而在这种情况下，在处理器能够访问闪存之前，会出现多个中止的写周期。

当中断触发中止时，所有命令会被不断中止，直至相应的FEEAENx位清零或中断源被清除。

闪存控制器存储器映射寄存器

表92. 闪存控制器存储器映射寄存器地址表(基地址：0x40002800)

偏移	名称	描述	访问类型	默认值
0x0000	FEESTA	状态寄存器。	R	0x0000
0x0004	FEECON0	命令控制寄存器。	RW	0x0000
0x0008	FEECMD	命令寄存器。	RW	0x0000
0x0010	FEEADR0L	低页地址寄存器。	RW	0x0000
0x0014	FEEADR0H	高页地址寄存器。	RW	0x0000
0x0018	FEEADR1L	低页地址寄存器。	RW	0x0000
0x001C	FEEADR1H	高页地址寄存器。	RW	0x0000
0x0020	FEEKEY	密钥寄存器。	W	0x0000
0x0028	FEEPROL	写保护寄存器的低16位。	RW	0xFFFF
0x002C	FEEPROH	写保护寄存器的高16位。	RW	0xFFFF
0x0030	FEESIGL	签名的低16位。	R	由内核更改
0x0034	FEESIGH	签名的高16位。	R	由内核更改
0x0038	FEECON1	用户设置寄存器。	RW	由内核更改
0x0048	FEEADRAL	写操作中止地址寄存器的低16位。	R	0x0800
0x004C	FEEADRAH	写操作中止地址寄存器的高16位。	R	0x0002
0x0078	FEEAEN0	中断中止使能寄存器。中断15至中断0。	RW	0x0000
0x007C	FEEAEN1	中断中止使能寄存器。中断31至中断16。	RW	0x0000
0x0080	FEEAEN2	中断中止使能寄存器。中断38至中断32。	RW	0x0000

闪存状态寄存器

地址：40002800；复位：0x0000；名称：FEESTA

表93. FEESTA寄存器位功能描述

位	名称	描述		
15:7	保留	读取时返回0。		
6	SIGNERR	复位时内核空间签名检查错误。 签名检查失败时置1。 复位后，闪存控制器自动检查信息空间签名。如果签名检查失败，此位置1。 用户只能通过串行线检查此位是否置1。如果此位置1，用户代码不会执行。		
5:4	CMDRES	这两位指示命令完成时的状态或写操作的状态。如果执行多个命令， 或者有多个写操作而没有读取状态寄存器，则存储第一个遇到的错误。 读取时，这些位清零为00。		
		设置	**状态**	**功能**
		00	SUCCESS	表示命令或写操作成功完成。
		01	PROTECTED	表示尝试擦除受保护位置。
		10	VERIFYERR	表示读取验证错误。擦除后，控制器读取对应的字以验证处理是否成功完成。如果读取的数据不是全F，则这就是所得状态。如果执行签名命令且所得签名与模块中高位页面的高位4字节数据不一致，则这就是所得状态。
		11	ABORT	表示命令或写操作因为中止命令而中止，或系统中断引起中止。
3	WRDONE	写操作完成。若有多个写操作(或突发写操作)，则该状态位在写入第一个长字后置位， 并且保持置位状态，直至被读取。若是对闪存执行突发写操作，则每写入一个长字， 此位都会置位(假定每写入一个长字后，便通过读取来将此位清零)。 写操作完成时置1。 读取时清0。		
2	CMDDONE	命令完成。若有多个命令，则该状态位在第一个命令完成之后置位， 并且保持置位状态，直至被读取。 命令完成时置1。 读取时清0。		
1	WRBUSY	写操作繁忙。 当闪存模块正在执行写操作时，该位置1。		
0	CMDBUSY	命令繁忙。 当闪存模块正在执行通过命令寄存器输入的命令时，该位置1。		

闪存控制寄存器

地址： 0x40002804；复位： 0x0000；名称： FEECON0

表94. FEECON0寄存器位功能描述

位	名称	描述
15:3	保留	读取时返回0。
2	WREN	写入使能。此位设为0时，闪存写操作会引起硬故障系统异常错误，写操作不会发生。 0: 禁用对闪存的写操作。 1: 使能对闪存的写操作。
1	IENERR	错误中断使能。当命令或闪存写操作出错时，产生一个中断。 0: 禁用。 1: 使能。
0	IENCMD	命令完成中断使能。则当命令或闪存写操作完成时，产生一个中断。 0: 禁用。 1: 使能。

闪存命令寄存器

地址：0x40002808；复位：0x0000；名称：FEECMD

表95. FEECMD寄存器位功能描述

位	名称	描述
15:4	保留	读取时返回0。
3:0	CMD	命令。

闪存模块支持下列命令。

对于重复的页擦除命令，执行每个命令之前都必须输入密钥。如果输入一个命令而事先没有输入密钥，则不会发生操作，CMDDONE不会置位。

命令完成之前，内核对闪存的任何访问都会暂停。

表96. 闪存控制器命令(FEECMD[3:0])

FEECMD[3:0]	名称	描述
0000	IDLE	不执行任何命令。
0001	ERASEPAGE	将要擦除的页面地址写入FEEADR0L/H，然后将此代码写入FEECMD寄存器， 闪存就会擦除该页。当擦除完毕时，闪存会读取该页中的每个位置， 验证该页中的所有字都已被擦除。如果发生读取验证错误， FEESTA会指示出来。 要擦除多页，请等待前一页已擦除完毕，检查状态，然后发出命令启动下一页擦除。 输入此命令之前，必须将密钥写入密钥寄存器，先写入0xF456，再写入0xF123。
0010	SIGN	使用此命令为一个数据块产生签名。签名逐页产生。要产生签名， 须将数据块第一页的地址输入FEEADR0L/FEEADR0H，将最后一页 地址输入FEEADR1L/FEEADR1H，然后将此代码写入FEECMD寄存器。 当该命令执行完毕时，签名便可在FEESIGL/FEESIGH中读取。 数据块中最后一页的最后4字节保留用于存储签名。 输入此命令之前，必须将密钥写入密钥寄存器，先写入0xF456，再写入0xF123。
0011	MASSERASE	擦除所有用户空间。要使能此操作，必须先将密钥写入FEEKEY寄存器，先写入0xF456， 再写入0xF123(这是为了防止意外擦除)。 批量擦除完毕时，控制器读取每个位置以验证所有位置都是0xFFFFFFFF。如果发生读取 验证错误，FEESTA会指示出来(FEESTA[5:4] = 0x2)。
0100	ABORT	如果发出此命令，当前正在执行的任何命令都会停止。状态指示出错的命令 (FEESTA[5:4] = 0x3)。注意：这是唯一一个可在其它命令正在执行时发出的命令。 此命令也可用来停止正在执行的写操作。 如果一个写操作被中止，正在写入的位置地址可通过FEEADRAL/FEEADRAH寄存器读取。 当闪存控制器正在写入一个长字时，流水线中可能有另一个来自Cortex-M3或DMA引擎 的写操作(取决于软件如何实现写操作)。因此，这两个写操作都可能需要中止。 如果中止写入或擦除操作，就会违反闪存时序，因而无法确定写入或擦除操作 是否成功完成。 要使能此操作，必须先将密钥写入FEEKEY寄存器，先写入0xF456，再写入0xF123 (这是为了防止意外中止)。

闪存控制器页面地址0寄存器

对于擦除命令，FEEADR0x指示要擦除的页面地址。

对于签名命令，FEEADR0x指示计算签名的起始页面。

低页地址寄存器

地址：0x40002810；复位：0x0000；名称：FEEADR0L

表97. FEEADR0L寄存器位功能描述

位	名称	描述
15:9	VALUE	闪存页面地址的位[15:9]。由擦除和签名命令使用。
8:0	Reserved	字节地址的9个保留位。字节地址的低9位会被忽略， 因为擦除和签名命令使用页面地址。读取时返回0x0。

高页地址寄存器

地址：0x40002814；复位：0x0000；名称：FEEADR0H

FEEADR0H为2位宽，用于访问ADuCM360/ADuCM361的信息空间。

表98. FEEADR0H寄存器位功能描述

位	名称	描述
15:1	保留	设为0。
0	VALUE	页面地址的位16。

闪存控制器页面地址1寄存器

对于签名命令，FEEADR1x指示计算签名的结束页面。

低页地址寄存器

地址： 0x40002818；复位： 0x0000；名称：FEEADR1L

表99. FEEADR1L寄存器位功能描述

位	名称	描述
15:9	VALUE	闪存页面地址的位[15:9]。用于识别签名命令使用的最后一页。
8:0	Reserved	字节地址的9个保留位。字节地址的低9位会被忽略，因为签名命令使用页面地址。 读取时返回0x0。

高页地址寄存器

地址：0x4000281C；复位：0x0000；名称：FEEADR1H

表100. FEEADR1H寄存器位功能描述

位	名称	描述
15:1	保留	设为0。
0	VALUE	页面地址的位16。

闪存控制器密钥寄存器

地址： 0x40002820；复位： 0x0000；名称： FEEKEY

表101. FEEKEY寄存器位功能描述

位	名称	描述
15:0	VALUE	先输入0xF456，再输入0xF123。读取时返回0x0。

闪存控制器写保护寄存器

低16位

地址：0x40002828；复位：0xFFFF；名称：FEEPROL

表102. FEEPROL寄存器位功能描述

位	名称	描述
15:0	VALUE	写保护的低16位。如果设置了闪存的写保护，则此寄存器为只读。 0: 保护部分闪存。 1: 闪存块不受保护。

高16位

地址：0x4000282C；复位：0xFFFF；名称：FEEPROH

表103. FEEPROH寄存器位功能描述

位	名称	描述
15:0	VALUE	写保护的高16位。如果设置了闪存的写保护，则此寄存器为只读。 0: 保护部分闪存。 1: 闪存块不受保护。

闪存控制器签名寄存器

低16位

地址：0x40002830；复位：由内核更改；名称：FEESIGL

表104. FEESIGL寄存器位功能描述

位	名称	描述
15:0	VALUE	签名的低16位。签名[15:0]。

高16位

地址：0x40002834；复位：由内核更改；名称：FEESIGH

表105. FEESIGH寄存器位功能描述

位	名称	描述
15:8	保留	保留。
7:0	VALUE	签名的高8位。签名[23:16]。

用户设置寄存器

地址：0x40002838；复位：由内核更改；名称：FEECON1

FEECON1寄存器受密钥保护。必须将密钥输入FEEKEY。写入FEECON1之后，必须再次向FEEKEY写入一个16位值以重新置位密钥保护。

表106. FEECON1寄存器位功能描述

位	名称	描述
15:1	保留	读取时返回0。
0	DBG	串行线调试使能。内核执行完毕时会将此位设为1，以使能用户进行调试访问。 0: 禁用通过串行线调试接口访问。 1: 使能通过串行线调试接口访问。

闪存控制器写入中止地址寄存器低16位

地址：0x40002848；复位：0x0800；名称：FEEADRAL

表107. FEEADRAL寄存器位功能描述

位	名称	描述
15:0	VALUE	FEEADRA寄存器的低16位。如果一个写操作被中止，这些位将包含写操作中止时写入的位置地址。

闪存控制器写入中止地址寄存器高16位

地址：0x4000284C；复位：0x0002；名称：FEEADRAH

表108. FEEADRAH寄存器位功能描述

位	名称	描述
15:0	VALUE	FEEADRA寄存器的高16位。

闪存控制器系统IRQ中止使能寄存器

地址：0x40002878；复位：0x0000；名称：FEEAEN0

要使一个系统中断能够中止一个写操作或命令(擦除、签名或批量验证)，应向此寄存器中的相应位写入1。例如，若需要外部IRQ2来中止一个闪存命令，应设置FEEAEN0 = 0x8。

FEEAEN0适用于表55中的0号至15号中断矢量。

表109. FEEAEN0寄存器位功能描述

位	位名称	描述	复位	访问类型
15	SINC2	SINC2中断中止使能位。 0: DIS。禁用SINC2中断中止。 1: EN。使能SINC2中断中止。	0x0	RW
14	ADC1	ADC1中断中止使能位。 0: DIS。禁用ADC1中断中止。 1: EN。使能ADC1中断中止。	0x0	RW
13	ADC0 (ADuCM360) 保留(ADuCM361)	ADC0中断中止使能位。 0: DIS。禁用ADC0中断中止。 1: EN。使能ADC0中断中止。	0x0	RW
12	T1	定时器1(通用定时器1)中断中止使能位。 0: DIS。禁用定时器1中断中止。 1: EN。使能定时器1中断中止。	0x0	RW
11	T0	定时器0(通用定时器0)中断中止使能位。 0: DIS。禁用定时器0中断中止。 1: EN。使能定时器0中断中止。	0x0	RW
10	保留			
9	T3	定时器3(看门狗定时器)中断中止使能位。 0: DIS。禁用定时器3中断中止。 1: EN。使能定时器3中断中止。	0x0	RW
8	EXTINT7	外部中断7中止使能位。 0: DIS。禁用外部中断7中止。 1: EN。使能外部中断7中止。	0x0	RW
7	EXTINT6	外部中断6中止使能位。 0: DIS。禁用外部中断6中止。 1: EN。使能外部中断6中止。	0x0	RW
6	EXTINT5	外部中断5中止使能位。 0: DIS。禁用外部中断5中止。 1: EN。使能外部中断5中止。	0x0	RW

位	位名称	描述	复位	访问类型
5	EXTINT4	外部中断4中止使能位。 0: DIS。禁用外部中断4中止。 1: EN。使能外部中断4中止。	0x0	RW
4	EXTINT3	外部中断3中止使能位。 0: DIS。禁用外部中断3中止。 1: EN。使能外部中断3中止。	0x0	RW
3	EXTINT2	外部中断2中止使能位。 0: DIS。禁用外部中断2中止。 1: EN。使能外部中断2中止。	0x0	RW
2	EXTINT1	外部中断1中止使能位。 0: DIS。禁用外部中断1中止。 1: EN。使能外部中断1中止。	0x0	RW
1	EXTINT0	外部中断0中止使能位。 0: DIS。禁用外部中断0中止。 1: EN。使能外部中断0中止。	0x0	RW
0	T2	定时器2(唤醒定时器)中断中止使能位。 0: DIS。禁用定时器2中断中止。 1: EN。使能定时器2中断中止。	0x0	RW

闪存控制器系统IRQ中止使能寄存器

地址：0x4000287C；复位：0x0000；名称：FEEAEN1

要使一个系统中断能够中止一个写操作或命令(擦除、签名或批量验证)，应向此寄存器中的相应位写入1。相应的位由中止闪存命令所需的中断决定。FEEAEN1适用于表55中的16号至31号中断矢量。

表110. FEEAEN1寄存器位功能描述

位	位名称	描述	复位	访问类型
15	DMADAC	DAC输出DMA中断中止使能位。 0: DIS。禁用DAC DMA中断中止。 1: EN。使能DAC DMA中断中止。	0x0	RW
14	DMAI2CMRX	I²C主机Rx DMA中断中止使能位。 0: DIS。禁用I²C主机Rx DMA中断中止。 1: EN。使能I²C主机Rx DMA中断中止。	0x0	RW
13	DMAI2CMTX	I²C主机Tx DMA中断中止使能位。 0: DIS。禁用I²C主机Tx DMA中断中止。 1: EN。使能I²C主机Tx DMA中断中止。	0x0	RW
12	DMAI2CSRX	I²C从机Rx DMA中断中止使能位。 0: DIS。禁用I²C从机Rx DMA中断中止。 1: EN。使能I²C从机Rx DMA中断中止。	0x0	RW
11	DMAI2CSTX	I²C从机Tx DMA中断中止使能位。 0: DIS。禁用I²C从机Tx DMA中断中止。 1: EN。使能I²C从机Tx DMA中断中止。	0x0	RW
10	DMAUARTRX	UARTRx DMA中断中止使能位。 0: DIS。禁用UARTRx DMA中断中止。 1: EN。使能UARTRx DMA中断中止。	0x0	RW
9	DMAUARTTX	UARTTx DMA中断中止使能位。 0: DIS。禁用UARTTx DMA中断中止。 1: EN。使能UARTTx DMA中断中止。	0x0	RW
8	DMASPI1RX	SPI1RXx DMA中断中止使能位。 0: DIS。禁用SPI1Rx DMA中断中止。 1: EN。使能SPI1Rx DMA中断中止。	0x0	RW
7	DMASPI1TX	SPI1TX DMA中断中止使能位。 0: DIS。禁用SPI1Tx DMA中断中止。 1: EN。使能SPI1Tx DMA中断中止。	0x0	RW
6	DMAERROR	DMA错误中断中止使能位。 0: DIS。禁用DMA错误中断中止。 1: EN。使能DMA错误中断中止。	0x0	RW
5	I2CM	I²C主机中断中止使能位。 0: DIS。禁用I²C主机中断中止。 1: EN。使能I²C主机中断中止。	0x0	RW
4	I2CS	I²C从机中断中止使能位。 0: DIS。禁用I²C从机中断中止。 1: EN。使能I²C从机中断中止。	0x0	RW
3	SPI1	SPI1中断中止使能位。 0: DIS。禁用SPI1中断中止。 1: EN。使能SPI1中断中止。	0x0	RW
2	SPI0	SPI0中断中止使能位。 0: DIS。禁用SPI0中断中止。 1: EN。使能SPI0中断中止。	0x0	RW
1	UART	UART中断中止使能位。 0: DIS。禁用UART中断中止。 1: EN。使能UART中断中止。	0x0	RW

位	位名称	描述	复位	访问类型
0	FEE	闪存控制器中断中止使能位。 0: DIS。禁用闪存控制器中断中止。 1: EN。使能闪存控制器中断中止。		

闪存控制器系统IRQ中止使能寄存器

地址：0x40002880；复位：0x0000；名称：FEEAEN2

要使一个系统中断能够中止一个写操作或命令(擦除、签名或批量验证)，应向此寄存器中的相应位写入1。相应的位由中止闪存命令所需的中断决定。FEEAEN2适用于表55中的32号至38号中断矢量。

表111. FEEAEN2寄存器位功能描述

位	位名称	描述	复位	访问类型
[15:7]	保留	保留。	0x00	R
6	PWM2	PWM2中断中止使能位。 0: DIS。禁用PWM2中断中止。 1: EN。使能PWM2中断中止。	0x0	RW
5	PWM1	PWM1中断中止使能位。 0: DIS。禁用PWM1中断中止。 1: EN。使能PWM1中断中止。	0x0	RW
4	PWM0	PWM0中断中止使能位。 0: DIS。禁用PWM0中断中止。 1: EN。使能PWM0中断中止。	0x0	RW
3	PWMTRIP	PWMTRIP中断中止使能位。 0: DIS。禁用PWMTRIP中断中止。 1: EN。使能PWMTRIP中断中止。	0x0	RW
2	DMASINC2	SINC2 DMA中断中止使能位。 0: DIS。禁用SINC2 DMA中断中止。 1: EN。使能SINC2 DMA中断中止。	0x0	RW
1	DMAADC1	ADC1 DMA中断中止使能位。 0: DIS。禁用ADC1 DMA中断中止。 1: EN。使能ADC1 DMA中断中止。	0x0	RW
0	DMAADC0(ADuCM360) 保留(ADuCM361)。	ADC0 DMA中断中止使能位。 0: DIS。禁用ADC0 DMA中断中止。 1: EN。使能ADC0 DMA中断中止。	0x0	RW

复位

复位特性

有四种复位形式：

- 外部复位
- 上电复位
- 看门狗超时
- 软件系统复位

复位工作原理

软件系统复位是Cortex-M3处理器的一部分。若要产生一次软件系统复位，应用中断/复位控制寄存器必须写入0x05FA0004。该寄存器是NVIC寄存器的一部分，位于地址0xE000ED0C。

RSTSTA寄存器存储复位原因，直至通过写入RSTCLR寄存器将其清零。在复位异常服务程序执行时，可以使用RSTSTA和RSTCLR来识别复位源。

注意，复位后看门狗定时器默认使能。默认超时周期约为32秒。

调试时或不需要看门狗定时器时，用户代码应在代码开始处禁用看门狗定时器。

```
pADI_WDT->T3CON = 0x00 ;                          // Disable watchdog timer
```

表112. 器件复位含义

	影响					
复位	将外部引脚复位到默认状态	执行内核	复位所有MMR (RSTSTA除外)	复位所有外设	有效SRAM	复位事件后的RSTSTA
SWRST	是[1]	是	是	是	是/否[2]	RSTSTA[3] = 1
WDRST	是[1]	是	是	是	是/否[2]	RSTSTA[2] = 1
外部复位引脚	是[1]	是	是	是	是/否[2]	RSTSTA[1] = 1
POR	是[1]	是	是	是	否	RSTSTA[0] = 1

[1] P0.7回到默认状态，即POR输出。它仅在发生POR事件时为低电平；在所有其它情况下，它保持高电平。

[2] 在UART下载后复位的情况下RAM无效。

复位存储器映射寄存器

表113. 复位存储器映射寄存器地址(基地址：0x40002400)

偏移	名称	描述	访问类型	默认值
0x0040	RSTSTA	状态寄存器(只读)。	R	0x01，因复位类型而异
0x0040	RSTCLR	状态清零寄存器(只写)。	W	不适用

复位状态/状态清零寄存器

地址：0x40002440；复位：0x01，因复位类型而异/不适用；名称：RSTSTA/RSTCLR

表114. RSTSTA/RSTCLR寄存器位功能描述

位	名称	描述
7:4	保留	保留。
3	SWRST	软件复位。 0: 通过设置对应RSTCLR位清0。 1: 产生Cortex-M3系统复位时自动置1。
2	WDRST	看门狗超时。 0: 通过设置对应RSTCLR位清0。 1: 发生看门狗复位时自动置1。
1	EXTRST	外部复位。 0: 通过设置对应RSTCLR位清0。 1: 发生外部复位时自动置1。
0	POR	上电复位。 0: 通过设置对应RSTCLR位清0。 1: 发生上电复位时自动设置。

数字I/O

数字I/O特性

ADuCM360/ADuCM361有多个双向通用输入/输出(GPIO)引脚。多数GPIO引脚具有多重功能，可通过用户代码进行配置。上电时，这些引脚中只有三个不是配置为GPIO，其中一个引脚反映POR状态，另外两个数字引脚配置为串行线接口。用户代码可以将这三个引脚配置为GPIO(模式01)。

数字I/O功能框图

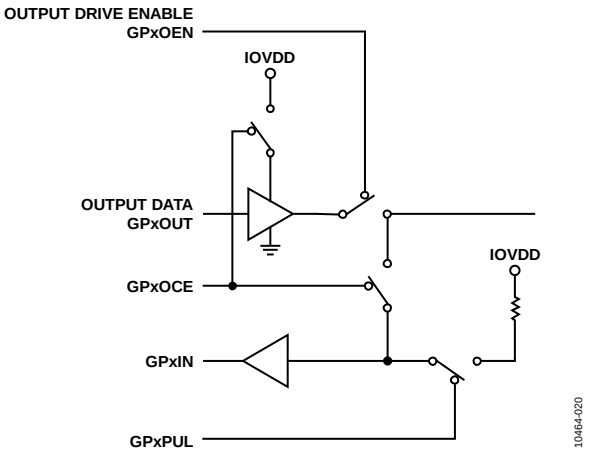


图21. GPIO结构

数字I/O概述

这些GPIO分组为3个端口：端口0包含8个GPIO，端口1包含8个GPIO，端口2包含3个GPIO。各GPIO可配置为输入、输出或完全开路，并内置可编程上拉电阻，驱动能力为1 mA。所有I/O引脚都可以在全部电源电压范围(IOVDD = 1.8 V至3.6 V最大值)内工作；逻辑输入电压指定为电源电压的百分比，如下所示：

V_{INL} = 0.2 × *IOVDD* 最大值

V_{INH} = 0.7 × *IOVDD* 最小值

ADuCM360/ADuCM361进入省电模式后，GPIO引脚保持原来的状态。绝对最大输入电压为IOVDD + 0.3 V，配置为输入或开路的GPIO典型漏电流为每个GPIO 50 nA。注意，驱动外设无法驱动引脚。也就是说，如果UART正在驱动一个引脚，则当进入深度睡眠模式时，它会与该引脚相隔离，电源会被切断。其状态和控制会在唤醒时恢复。

数字I/O工作原理

I/O上拉使能

所有GPIO引脚都带有一个内部上拉电阻，其驱动能力为1 mA。当引脚配置为输入时，利用GPxPUL寄存器可使能或禁用引脚的上拉电阻。当GPIO引脚设置为输出或使能开路时，上拉电阻自动禁用。

上拉采用MOS器件，因此，上拉电阻值随引脚上的电压而改变。

对于低功耗工作模式，当数字I/O配置为输入时，应确保其不浮空。务必使能上拉电阻以确保器件处于已定义的逻辑状态。

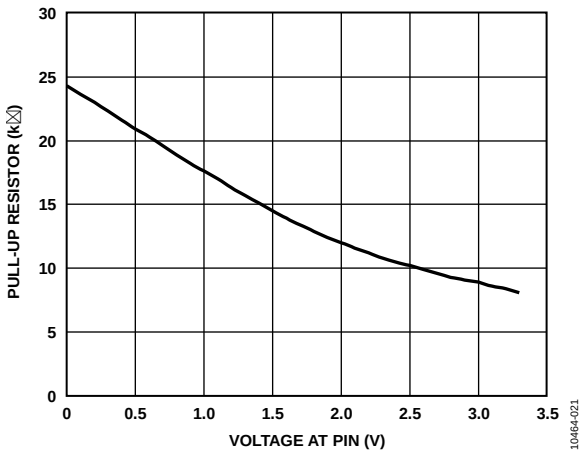


图22. GPIO上拉电阻值

I/O数据输入

配置为输入(默认)时，GPIO输入电平通过GPxIN提供。

开路使能

如果引脚设置为输出，使能开路将会禁用输入路径。要禁用输入且不驱动引脚，应设置开路并驱动逻辑1。开路使能时，外部中断不可用。

I/O数据输出

当GPIO配置为输出时，GPxOUT值会反映在GPIO上。

位设置

位设置模式用于设置一个端口中的一个或多个GPIO数据输出，而不影响其它输出。仅与等于1的写入数据位相对应的GPIO被设置，其余GPIO不受影响。

位清零

位清零模式用于清零一个端口中的一个或多个GPIO数据输出，而不影响其它输出。仅与等于1的写入数据位相对应的GPIO被清零，其余GPIO不受影响。

位反转

位反转模式用于反转一个端口中的一个或多个GPIO数据输出，而不影响其它输出。仅与等于1的写入数据位相对应的GPIO被反转，其余GPIO不受影响。

I/O数据输出使能

使能数据输出路径，GPxOUT值会反映在GPIO上。

表115. GPIO状态

GPxOCE	GPxOEN	GPxOUT[1]	GPIO输入状态/中断	GPIO输出状态
0	0	X	始终可用	不可用。配置为输入。
0	1	X	始终可用	输出逻辑取决于OUT(驱动逻辑0还是逻辑1)。
1	0	X	始终可用	不可用。配置为输入。
1	1	0	不可用	驱动0输出(开路)。
1	1	1	不可用	输出浮空(开漏)。

[1] X = 无关。

数字端口复用

此模块用于控制特定引脚的GPIO功能，因为某些引脚提供了功能选择，既可用作GPIO，也可起到其它作用。

表116. GPIO复用表

	配置模式			
GPIO	**00**	**01**	**10**	**11**
GP0—GP0CON控制如下位				
P0.0	GPIO	SPI1 MISO (GP0CON |= 0x1)		
P0.1	GPIO	SPI1 SCLK (GP0CON |= 0x4)	SCL (GP0CON |= 0x8)	UART RxD (GP0CON |= 0xC)
P0.2	GPIO	SPI1 MOSI (GP0CON |= 0x10)	SDA (GP0CON |= 0x20)	UART TxD (GP0CON |= 0x30)
P0.3	GPIO/IRQ0	SPI1 CS (GP0CON |= 0x40)		
P0.4	GPIO	UART RTS (GP0CON |= 0x100)	ECLK OUT (GP0CON |= 0x200)	
P0.5	GPIO/IRQ1	UART CTS (GP0CON |= 0x400)		
P0.6	GPIO/IRQ2	UART RXD (GP0CON |= 0x1000)		
P0.7	$\overline{\text{POR}}$	GPIO (GP0CON |= 0x4000)	UART TXD (GP0CON |= 0x8000)	
GP1—GP1CON控制如下位				
P1.0	GPIO/IRQ3	PWM SYNC (GP1CON |= 0x1)	EXT CLK IN (EXTCLK) (GP1CON |= 0x2)	
P1.1	GPIO/IRQ4	PWM TRIP (GP1CON |= 0x4)		UART DTR (GP1CON |= 0xC)
P1.2	GPIO	PWM0 (GP1CON |= 0x10)		UART RI (GP1CON |= 0x30)
P1.3	GPIO	PWM1 (GP1CON |= 0x40)		UART DSR (GP1CON |= 0xC0)
P1.4	GPIO	PWM2 (GP1CON |= 0x100)	SPI0 MISO (GP1CON |= 0x200)	
P1.5	GPIO/IRQ5	PWM3 (GP1CON |= 0x400)	SPI0 SCLK (GP1CON |= 0x800)	
P1.6	GPIO/IRQ6	PWM4 (GP1CON |= 0x1000)	SPI0 MOSI (GP1CON |= 0x2000)	
P1.7	GPIO/IRQ7	PWM5 (GP1CON |= 0x4000)	SPI0 CS (GP1CON |= 0x8000)	
GP2—GP2CON控制如下位				
P2.0	GPIO	SCL (GP2CON |= 0x1)		UART CLKIN (GP2CON |= 0x3)
P2.1	GPIO	SDA (GP2CON |= 0x4)		UART DCD (GP2CON |= 0xC)
P2.2	GPIO			
P2.3	SWCLK(默认)[1]			
P2.4	SWD(默认)[2]			
P2.5				
P2.6				
P2.7				

[1] P2.3默认为串行线接口引脚。

[2] P2.4默认为串行线接口引脚。

GPIO存储器映射寄存器

表117. GPIO接口存储器地址表(基地址：0x40006000)

偏移	名称	描述	访问类型	默认值
0x0000	GP0CON	GPIO端口0配置	RW	0x0000
0x0004	GP0OEN	GPIO端口0输出使能	RW	0x00
0x0008	GP0PUL	GPIO端口0输出上拉使能	RW	0xFF
0x000C	GP0OCE	GPIO端口0开路使能	RW	0x00
0x0014	GP0IN[1]	GPIO端口0数据输入	R	不适用
0x0018	GP0OUT	GPIO端口0数据输出	RW	0x00
0x001C	GP0SET	GPIO端口0数据输出设置	W	0x00
0x0020	GP0CLR	GPIO端口0数据输出清零	W	0x00
0x0024	GP0 TGL	GPIO端口0数据输出反转	W	0x00
0x0030	GP1CON	GPIO端口1配置	RW	0x0000
0x0034	GP1OEN	GPIO端口1输出使能	RW	0x00
0x0038	GP1PUL	GPIO端口1输出上拉使能	RW	0xFF
0x003C	GP1OCE	GPIO端口1开路使能	RW	0x00
0x0044	GP1IN[1]	GPIO端口1数据输入	R	不适用
0x0048	GP1OUT	GPIO端口1数据输出	RW	0x00
0x004C	GP1SET	GPIO端口1数据输出设置	W	0x00
0x0050	GP1CLR	GPIO端口1数据输出清零	W	0x00
0x0054	GP1TGL	GPIO端口1引脚反转	W	0x00
0x0060	GP2CON	GPIO端口2配置	RW	0x00
0x0064	GP2OEN	GPIO端口2输出使能	RW	0x00
0x0068	GP2PUL	GPIO端口2输出上拉使能	RW	0xFF
0x006C	GP2OCE	GPIO端口2开路使能	RW	0x00
0x0074	GP2IN[1]	GPIO端口2数据输入	R	不适用
0x0078	GP2OUT	GPIO端口2数据输出	RW	0x00
0x007C	GP2SET	GPIO端口2数据输出设置	W	0x00
0x0080	GP2CLR	GPIO端口2数据输出清零	W	0x00
0x0084	GP2TGL	GPIO端口2引脚反转	W	0x00

[1] GPxIN寄存器内容取决于对应引脚的逻辑电平。

GPIO配置寄存器

表118. GPxCON寄存器位功能描述(GP0CON地址：0x40006000；GP1CON地址：0x40006030；GP2CON地址：0x40006060)

位	名称	描述
15:14	CON7	Px.7的配置位，参见表116。
13:12	CON6	Px.6的配置位，参见表116。
11:10	CON5	Px.5的配置位，参见表116。
9:8	CON4	Px.4的配置位，参见表116。
7:6	CON3	Px.3的配置位，参见表116。
5:4	CON2	Px.2的配置位，参见表116。
3:2	CON1	Px.1的配置位，参见表116。
1:0	CON0	Px.0的配置位，参见表116。

GPIO输出使能寄存器

表119. GPxOEN寄存器位功能描述(GP0OEN地址：0x40006004；GP1OEN地址：0x40006034；GP2OEN地址：0x40006064)

位	名称	描述
7:0	OEN[7:0]	输出使能。 0: 禁用端口x上对应GPIO的输出。 1: 使能端口x上对应GPIO的输出。

GPIO上拉寄存器

表120. GPxPUL寄存器位功能描述(GP0PUL地址：0x40006008；GP1PUL地址：0x40006038；GP2PUL地址：0x40006068)

位	名称	描述
7:0	PUL[7:0]	输出使能。 0: 禁用端口x上对应GPIO的内部上拉电阻。 1: 使能端口x上对应GPIO的内部上拉电阻。

GPIO开路使能寄存器

表121. GPxOCE寄存器位功能描述(GP0OCE地址：0x4000600C；GP1OCE地址：0x4000603C；GP2OCE地址：0x4000606C)

位	名称	描述
7:0	OCE[7:0]	输出使能。设置端口x上的GPIO焊盘为开路模式。

GPIO数据输入寄存器

表122. GPxIN寄存器位功能描述(GP0IN地址：0x40006014；GP1IN地址：0x40006044；GP2IN地址：0x40006074)

位	名称	描述
7:0	IN[7:0]	反映非开路模式下GPIO引脚的电平。

GPIO数据输出寄存器

表123. GPxOUT寄存器位功能描述(GP0OUT地址：0x40006018；GP1OUT地址：0x40006048；GP2OUT地址：0x40006078)

位	名称	描述
7:0	OUT[7:0]	数据输出寄存器。回读GPIO输出值。例如：写入GP0OUT = 0x12将驱动P0.1和P0.4为1，其余GPIO为低电平(假设这些引脚配置为输出)。 0: 用户清0时，将对应的GPIO驱动到低电平。 1: 用户置1时，将对应的GPIO驱动到高电平。

GPIO位设置寄存器

表124. GPxSET寄存器位功能描述(GP0SET地址：0x4000601C；GP1SET地址：0x4000604C；GP2SET地址：0x4000607C)

位	名称	描述
7:0	SET[7:0]	数据输出寄存器。 0: 无操作。 1: 用户置1时，将对应的GPIO驱动到高电平。

GPIO位清零寄存器

表125. GPxCLR寄存器位功能描述(GP0CLR地址：0x40006020；GP1CLR地址：0x40006050；GP2CLR地址：0x40006080)

位	名称	描述
7:0	CLR[7:0]	数据输出寄存器。 0: 用户清0；不起作用。 1: 用户置1时，将对应的GPIO驱动到低电平。

GPIO反转引脚寄存器

表126. GPxTGL寄存器位功能描述(GP0TGL地址：0x40006024；GP1TGL地址：0x40006054；GP2TGL地址：0x40006084)

位	名称	描述
7:0	TGL[7:0]	Toggle pin。 0: 用户清0；不起作用。 1: 用户置1时，反转对应的GPIO。

I²C串行接口

I²C特性

- 主机或从机模式，具有2字节发送和接收FIFO
- 模式支持
  - 7位寻址模式和10位寻址模式
  - 四个7位器件地址或一个10位地址和两个7位地址(从机模式)
  - 主机模式和从机模式下重复起始
  - 总线上的其它器件可以使能时钟延展，这不会给ADuCM360/ADuCM361带来任何问题，不过ADuCM360/ADuCM361无法使能时钟延展
  - 主机仲裁
  - 主机连续读取模式或最多以512字节速率固定读取
  - 内部和外部回送
- 主机模式和从机模式均支持DMA
- 对从机NACK信号进行软件控制

I²C指最初由Philips Semiconductors(现为NXP Semiconductors)开发的一种通信协议。

I²C概述

I²C数据传输使用串行时钟引脚(SCL)和串行数据引脚(SDA)。这些引脚配置为“线与”格式，可以在多主机系统中进行仲裁。SCL和SDA均为双向，必须用一个上拉电阻器连接到正极电源。

I²C系统的传输过程为：当总线处于空闲状态时，主机通过产生起始条件来启动传输；在起始地址发送期间，主机发送从机的地址和数据发送器的方向。如果主机没有输掉仲裁且从机应答了初始地址传输，那么就会开始数据传输。传输会持续到主机发送一个停止条件为止，然后总线进入空闲状态。图23所示为典型I²C主机。

可配置主机产生串行时钟。频率由用户在串行时钟分频寄存器I2CDIV中设置。主机通道经过设置，可以在快速模式(400 kHz)或标准模式(100 kHz)下工作。

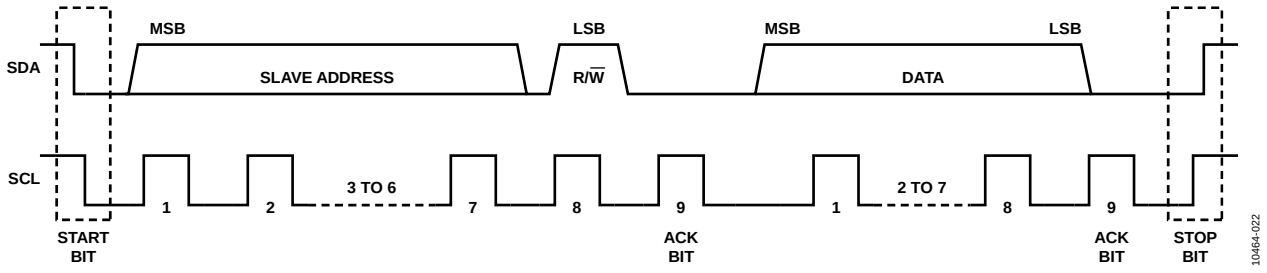


图23. 典型I²C传输序列

I²C总线系统的外设地址由用户编程设定。没有进行传输时，可随时修改这个ID。用户最多可以设置4个由外设识别的从机地址。外设的发送和接收移位寄存器各有2字节FIFO。控制寄存器中的IRQ和状态位用于告知处理器内核何时需要处理FIFO。

I²C工作原理

I²C启动

运行I²C外设需要下列步骤：

1. 在CLKDIS寄存器和CLKCON1寄存器中配置I²C时钟。
2. 配置数字引脚为I²C工作模式(P0.1/P0.2、P2.0/P2.1)。
3. 根据需要将I²C寄存器配置为从机或主机操作。
4. 根据需要使能I²C从机或主机中断源。

注意：使用I²C时，用户应通过GP0PUL寄存器禁用I²C引脚的内部上拉电阻。

表127. GPIO复用

GPIO	配置模式(10)
P0.1, P2.0	SCL
P0.2, P2.1	SDA

寻址模式

7位寻址

I2CID0寄存器、I2CID1寄存器、I2CID2寄存器和I2CID3寄存器包含从机器件ID。器件将四个I2CIDx(x=0,1,2,3)寄存器中的数据与地址字节做比较。为确保寻址准确，每一个ID寄存器的7个MSB必须与最先接收到的地址字节的7个MSB相同。在地址识别过程中，ID寄存器的LSB(传输方向位)被忽略。

主机利用I2CADR0寄存器寻址器件。

10位寻址

对于主机和从机模式，此特性通过设置I2CSCON[1]来使能。

从机的10位地址存储在I2CID0和I2CID1中，其中I2CID0包含地址的第一个字节，R/W位和高5位必须设置为11110，如图24所示。I2CID1包含10位地址的余下8位。I2CID2和I2CID3仍可设置为7位地址。

主机利用I2CADR0和I2CADR1寄存器与10位地址从机通信。格式如图24所示。

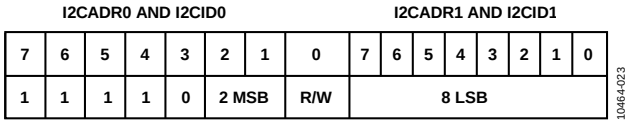


图24. 10位地址格式

重复起始条件是指向从机发送第二个起始条件，且第一个和第二个起始条件之间没有发送停止条件。它允许主机通过改变R/$\overline{W}$位逆转传输方向，但不必放弃对总线的控制权。

图25所示为一个传输序列示例。这通常在第一个数据发送至设置读取寄存器地址的元件时使用。

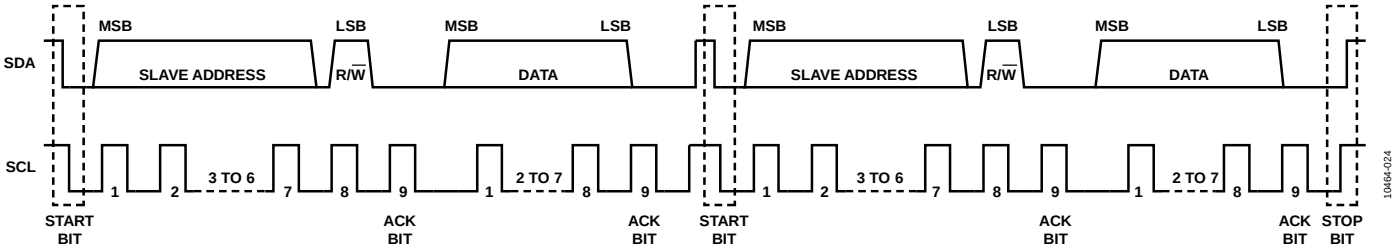


图25. I²C重复起始序列

在从机侧，当接收到重复起始条件和从机地址时，会产生一个中断(若I2CSCON寄存器使能了中断)。使用I2CSSTA MMR中的START和REPSTART状态位可以区分是重复起始条件加从机地址时还是起始条件加从机地址。

在主机侧，当主机仍忙于处理时，如果写入I2CADR0寄存器，主机会产生重复起始条件。状态机开始发送从机地址后，便可安全地写入I2CADR0寄存器。

例如，若需要一个涉及到写入、重复起始然后读/写的处理，则应在状态机开始发送器件地址之后，或者在收到第一个TXREQ中断之后，写入I2CADR0寄存器。当发送FIFO清空时，产生重复起始条件。

同样，若需要一个涉及到读取、重复起始然后读/写的处理，则应在状态机开始发送器件地址之后，或者在收到第一个RXREQ中断之后，写入主机第一地址字节寄存器I2CADR1。当达到请求的接收数量时，产生重复起始条件。

I²C时钟控制

I²C外设时钟由一个门控16 MHz系统时钟(UCLK)提供。CLKCON1[8:6]位用于让I²C模块以较慢的时钟速度工作，即对16 MHz时钟进行分频，这有助于降低功耗。

注意：对于100 kHz I²C主机时钟，所需的最小I²C时钟为2 MHz。

对于400 kHz I²C主机时钟，所需的最小I²C时钟为8 MHz。

系统中的I²C主机生成传输串行时钟。主机通道经过配置，可以在快速模式(400 kHz)或标准模式(100 kHz)下工作。

I2CDIV寄存器中的比特率定义如下：

$$f_{SCL} = fI_{2CCLK}/(LOW + HIGH + 3)$$

其中：

$f_{I2CCLK} = f_{UCLK}/(CLKSYSDIV \times I2CCD)$。
f_{UCLK}为系统时钟16 MHz。
*CLKSYSDIV*为1或2，取决于CLKSYSDIV[0]位的设置。
*I2CCD*是时钟分频值，由CLKCON1[8:6]设置。
*HIGH*为时钟的高电平周期，I2CDIV[15:8] = (REQD_HIGH_TIME/UCLK_PERIOD) − 2。
*LOW*为时钟的低电平周期，I2CDIV[7:0] = (REQD_LOW_TIME/UCLK_PERIOD) − 1。

对于100 kHz SCL操作，低电平时间为5000 ns，高电平时间5000 ns，UCLK频率为16 MHz，

$HIGH$ = (5000 ns/(1/16000000)) − 2 = 78 = 0x4E

LOW = (5000 ns/(1/16000000)) − 1 = 79 = 0x4F

f_{SCL} = 16000000/(79 + 78 + 3) = 100 kHz

对于400 kHz SCL操作，低电平时间为1250 ns，高电平时间1250 ns，UCLK频率为16 MHz，

$HIGH$ = (1250 ns/(1/16000000)) − 2 = 18 = 0x12

LOW = (1250 ns/(1/16000000)) − 1 = 19 = 0x13

f_{SCL} = 16000000/(18 + 19 + 3) = 400 kHz

I²C工作模式

主机传输启动

如果主机使能位(I2CMCON[0]，MASEN)置1，则向I2CADRx寄存器写入一个值便会启动主机传输序列。如果I2CMTX寄存器中的数据有效，它便是在写入序列期间地址字节后传输的第一个字节。

从机传输启动

如果从机使能位(I2CSCON[0]，SLVEN)置1，则会监控寄存器I2CID0、寄存器I2CID1、寄存器I2CID2或寄存器I2CID3中的器件地址的从机传输序列。如果识别了器件地址，器件就会加入从机传输序列中。

注意：从机操作始终是从置位三个中断源——读请求(RXREQ)、写请求(TXREQ)和广播(GCINT)中断——中的一个开始；软件务必寻找一个停止中断，确保处理正确完成，并解除置位停止中断状态位。

Rx/Tx数据FIFO

发送数据路径包括主机和从机Tx FIFO(I2CMTX和I2CSTX，均为2字节深)以及一个发送移位器。发送状态位I2CMSTA[1:0]和I2CSSTA[0]表示Tx FIFO中是否存在有效数据。当一个串行字节开始传输时，来自Tx FIFO的数据载入Tx移位器。在一个主动传输序列期间，如果Tx FIFO未满，I2CMSTA或I2CSSTA中的发送请求位(TXREQ)就会置位。

在从机中，当加载Tx移位器时，如果没有有效数据可供发送，发送下溢状态位(TXUR)就会置位(I2CMSTA[12]、ISCSSTA[1])。

如果发送FIFO中没有数据且主机正在写入数据，主机就会产生停止条件。

接收数据路径包括主机和从机Rx FIFO(I2CMRX和I2CSRX，均为2字节深)。I2CMSTA或I2CSSTA中的接收请求中断位(RXREQ)表示Rx FIFO中是否存在有效数据。每收到一个字节后，数据便载入Rx FIFO。如果Rx FIFO中的有效数据被Rx移位器覆写，接收上溢状态位(RXOF)就会置位(I2CMSTA[9]或I2CSSTA[4])。

主机NACK

接收数据时，如果主机FIFO已满且试图向FIFO再写入一个字节，主机就会用NACK回应。最后收到的字节不会写入FIFO，而是丢失。

从机无应答

如果从机不想应答一个读取访问，那么不将数据写入从机发送FIFO就会产生一个NACK。

如果从机不想应答主机写操作，应将从机控制寄存器中的NACK位I2CSCON[7]置位。

通常，从机会应答(ACK)写入接收FIFO中的所有字节。如果接收FIFO已满，从机将无法写入更多字节，因而不会应答未写入FIFO的字节。此时，主机应停止处理。

如果读/写位置1且发送FIFO为空，则从机不会应答一个匹配器件地址。因此，微控制器响应从机发送请求和置位ACK的时间非常短。由于这个原因，建议置位EARLYTXR (I2CSCON[5])。

广播

I^2C广播用于对I^2C总线的每个器件进行寻址。广播地址为0x00或0x01。第一个字节是地址字节，之后是命令字节。

如果地址字节为0x00，则第二字节(即命令字节)可以是如下值之一：

- 0x6：I^2C接口(主机和从机)复位。广播中断状态置位，广播ID位GCID (I2CSSTA[9:8])为0x1。用户代码应采取纠正措施来复位整个系统，或者只是重新使能I2C接口。
- 0x4：广播中断状态位置位，广播ID位(GCID)为0x2。

如果地址字节为0x01，则发出硬件广播。

- 这种情况下，第二字节为硬件主机地址。

对于任何广播，收到第二字节之后，广播中断状态位就会置1，用户代码应采取纠正措施重新设置器件地址。

如果GCEN置位，从机总是会应答广播的第一字节。如果第二字节是0x04或0x06，或者第二字节是硬件广播且HGCEN (I2CSCON[3])置位，则它也会应答广播的第二字节。

I2CALT寄存器包含硬件广播序列的备选器件ID。如果硬件广播使能位、HGCEN、GCEN和SLVEN全都置1，器件就会识别硬件广播。当发出一个广播序列且该序列的第二字节与ALT相同时，器件会辨识出硬件广播序列。

I^2C复位模式

向SLVEN写入0时，从机状态机复位。

向MASEN写入0时，主机状态机复位。

I^2C测试模式

将LOOPBACK位(I2CMCON[2])置1，可使器件处于内部回送模式。共有4个FIFO(主机Tx和Rx以及从机Tx和Rx)，因此，完全能够设置I^2C外设与自己交谈。如果设置主机对从机寻址，就可以实现外部回送。

I^2C低功耗模式

如果主机和从机均禁用(MASEN = SLVEN = 0)，I^2C部分便关闭。 要完全关断I^2C模块，应设置CLKCON1[8:6] = 111且CLKDIS[2] = 1以禁用芯片I^2C部分的时钟。

DMA请求

为I^2C主机和从机服务需要4个DMA通道。从机控制寄存器和主机控制寄存器均提供了DMA使能位。下面是显示如何配置DMA控制器的四种设置的示例代码

示例代码：I²C从机写入DMA周期

```
void I2CSLAVETXDMAINIT(void)
{
pADI_GP0->GPCON = 0x28;                                    // Configure P0.[1:2] as I2C pins
pADI_CLKCTL-> CLKCON1&= 0xFF0F;                            // Enable clock to I2C block
pADI_I2C->I2CSCON = I2CSCON_SLVEN                          // Enable Slave mode
+ I2CSCON_EARLYTXR                                         // Enable early Tx request interrupt
source
+ I2CSCON_IENSTOP    + I2CSCON_IENRX                       // Enable stop and Rx interrupts
+ I2CSCON_IENTX    + I2CSCON_IENREPST                      // Enable Tx and repeated Start
interrupts
+ 0x4000;                                                  // Enable Tx DMA
pADI_I2C->I2CID0 = 0xA2;                                   // Set Slave ID0 register
pADI_I2C->I2CID1 = 0xA4;                                   // Set Slave ID1 register
pADI_I2C->I2CID2 = 0xA6;                                   // Set Slave ID2 register
pADI_I2C->I2CID3 = 0xA0;                                   // Set Slave ID3 register
pADI_I2C->I2CFSTA = 0x100;                                 // Flush Slave Tx FIFO
pADI_I2C->I2CFSTA &= ~0x100;
                                                           // Enable I2C Slave,
Dma_Init();                                                // See ADC DMA example for this
function

I2CSLAVEDMAWRITE(uxI2CTXData, 16);
pADI_DMA->DMACFG = 0x1;                                    // Enable DMA mode in DMA controller
pADI_DMA->DMAENSET = 0x10;                                 // Enable I2C_TX_DMA Channel
NVIC_EnableIRQ(DMA_I2CS_TX_IRQn);                          // I2C Tx DMA interrupt sources  -
M360
}

void I2CSLAVEDMAWRITE(unsigned char *pucTX_DMA, unsigned int iNumRX)
{
   DmaDesc Desc;

                                                           // Common configuration of all the
descriptors used here
    Desc.ctrlcfg.bits.cycle_ctrl       = cyclectrl_basic;   //cyclectrl_basic;
    desc.ctrlcfg.bits.next_useburst    = 0x1;
    desc.ctrlcfg.bits.r_power          = 0;
    Desc.ctrlCfg.Bits.src_prot_ctrl    = 0x0;
    Desc.ctrlCfg.Bits.dst_prot_ctrl    = 0x0;
    Desc.ctrlCfg.Bits.src_size         = SRCSIZE_BYTE;
    Desc.ctrlCfg.Bits.dst_size         = DSTSIZE_BYTE;

                                                           //  TX Primary Descriptor
Desc.srcEndPtr                     = (unsigned int)(pucTX_DMA + iNumRX - 0x1);
    Desc.destEndPtr                    =  (unsigned int)&I2CSTX;
    Desc.ctrlCfg.Bits.n_minus_1        = iNumRX - 0x1;
    Desc.ctrlCfg.Bits.src_inc          = INC_BYTE;
```

```
    Desc.ctrlCfg.Bits.dst_inc          = INC_NO;
    *Dma_GetDescriptor(DMA_REQ_I2C0_SLV_TX,FALSE) = Desc;
}
                                                          // Slave DMA Tx IRQ handler
void DMA_I2C0_STX_Int_Handler()
{
NVIC_DisableIRQ(DMA_I2CS_TX_IRQn);                        // Clear Interupt source
}
```

示例代码：I²C从机读取DMA周期

```
void I2CSLAVERXDMAINIT(void)
{
pADI_GP0->GPCON = 0x28;                                   // Configure P0.[1:2] as I2C pins
pADI_CLKCTL-> CLKCON1&= 0xFF0F;                           // Enable clock to I2C block
pADI_I2C->I2CSCON = I2CSCON_SLV                           // Enable Slave mode
+ I2CSCON_EARLYTXR                                        // Enable early Tx request interrupt
source
+ I2CSCON_IENSTOP    + I2CSCON_IENRX                      // Enable stop and Rx interrupts
+ I2CSCON_IENTX    + I2CSCON_IENREPST                     // Enable Tx and repeated Start
interrupts
+ 0x2000;                                                 // Enable Rx DMA
pADI_I2C->I2CID0 = 0xA2;                                  // Set Slave ID0 register
pADI_I2C->I2CID1 = 0xA4;                                  // Set Slave ID1 register
pADI_I2C->I2CID2 = 0xA6;                                  // Set Slave ID2 register
pADI_I2C->I2CID3 = 0xA0;                                  // Set Slave ID3 register
                                                          // Enable I2C Slave,
Dma_Init();

I2CSLAVEDMAREAD(uxI2CRXData, 16);
pADI_DMA->DMACFG = 0x1;                                   // Enable DMA mode in DMA controller
pADI_DMA->DMAENSET = 0x20;                                // Enable I2C_RX_DMA Channel
NVIC_EnableIRQ(DMA_I2CS_RX_IRQn);                         // I2C Rx DMA interrupt sources  -
M360
}

void I2CSLAVEDMAREAD(unsigned char *pucRX_DMA, unsigned int iNumRX)
{
   DmaDesc Desc;

                                                          // Common configuration of all the
descriptors used here
    desc.ctrlcfg.bits.cycle_ctrl       = cyclectrl_basic;
    desc.ctrlcfg.bits.next_useburst    = 0x0;
    desc.ctrlcfg.bits.r_power          = 0;
    desc.ctrlcfg.bits.src_prot_ctrl    = 0x0;
    Desc.ctrlCfg.Bits.dst_prot_ctrl    = 0x0;
    Desc.ctrlCfg.Bits.src_size         = SRCSIZE_BYTE;
    Desc.ctrlCfg.Bits.dst_size         = DSTSIZE_BYTE;
```

```
                                                                    // RX Primary Descriptor
Desc.srcEndPtr                      = (unsigned int)&I2CSRX;
    Desc.destEndPtr                     = (unsigned int)(pucRX_DMA + iNumRX - 0x1); //
    Desc.ctrlCfg.Bits.n_minus_1         = iNumRX - 0x1;
    Desc.ctrlCfg.Bits.src_inc           = INC_NO;
    Desc.ctrlCfg.Bits.dst_inc           = INC_BYTE;
    *Dma_GetDescriptor(DMA_REQ_I2C0_SLV_RX,FALSE) = Desc;
}
                                                                    // I2C DMA Rx IRQ handler
void DMA_I2C0_SRX_Int_Handler()
{
NVIC_DisableIRQ(DMA_I2CS_RX_IRQn);                                  // Clear Interupt source
}
```

示例代码：I²C主机写入DMA周期

```
void I2CMASTERTXDMAINIT(void)
{
pADI_GP0->GPCON = 0x28;                                             // Configure P0.[1:2] as I2C
pins
pADI_CLKCTL-> CLKCON1&= 0xFF0F;                                     // Enable clock to I2C block
                                                                    // Enable I2C master, IRQ on
byte Received
                                                                    // Enable IRQ on Transmit
complete, Arbitration lost, NACK received, Complete transaction (Stop detected)
                                                                    // Decrement Tx FIFO status
on byte full transmitted.
pADI_I2C->I2CMCON = I2CMCON_MASEN + I2CMCON_IENRX
+ I2CMCON_IENTX        + I2CMCON_IENALOST   + I2CMCON_IENNACK
+ I2CMCON_IENCMP
+ 0x800;                                                           // Enable Rx DMA
pADI_I2C->I2CFSTA = 0x200;                                          // Flush Master Tx FIFO
pADI_I2C->I2CFSTA &= ~0x200;


                                                                    // Enable I2C Master,
Dma_Init();


I2CMASTERDMAWRITE(uxI2CTXData, 16);
pADI_DMA->DMACFG = 0x1;                                             // Enable DMA mode in DMA
controller
pADI_DMA->DMAENSET = 0x40;                                          // Enable I2C_TX_DMA Channel
NVIC_EnableIRQ(DMA_I2CM_TX_IRQn);                                   // I2C Tx DMA interrupt
sources- M360
pADI_I2C->I2CADR0 =    0xA0;
}


void I2CMASTERDMAWRITE(unsigned char *pucTX_DMA, unsigned int iNumRX)
{
   DmaDesc Desc;
                                                                    // Common configuration of
all the descriptors used here
```

```
    Desc.ctrlCfg.bits.cycle_ctrl       = cyclectrl_basic;   //cyclectrl_basic;
    desc.ctrlcfg.bits.next_useburst    = 0x1;
    desc.ctrlcfg.bits.r_power          = 0;
    Desc.ctrlCfg.Bits.src_prot_ctrl    = 0x0;
    Desc.ctrlCfg.Bits.dst_prot_ctrl    = 0x0;
    Desc.ctrlCfg.Bits.src_size         = SRCSIZE_BYTE;
    Desc.ctrlCfg.Bits.dst_size         = DSTSIZE_BYTE;

                                                          // TX Primary Descriptor
Desc.srcEndPtr                     = (unsigned int)(pucTX_DMA + iNumRX - 0x1);
    Desc.destEndPtr                    =  (unsigned int)&I2CMTX;
    Desc.ctrlCfg.Bits.n_minus_1        = iNumRX - 0x1;
    Desc.ctrlCfg.Bits.src_inc          = INC_BYTE;
    Desc.ctrlCfg.Bits.dst_inc          = INC_NO;
    *Dma_GetDescriptor(DMA_REQ_I2C0_MST_TX,FALSE) = Desc;
}
                                                          // Master DMA Tx IRQ handler
void DMA_I2C0_MTX_Int_Handler()
{
NVIC_DisableIRQ(DMA_I2CM_TX_IRQn);                        // Clear Interupt source
}
```

示例代码：I²C主机读取DMA周期

```
void I2CMASTERRXDMAINIT(unsigned char ucNumBytes)
{
pADI_GP0->GPCON = 0x28;                                   // Configure P0.[1:2] as I2C pins
pADI_CLKCTL-> CLKCON1&= 0xFF0F;                           // Enable clock to I2C block
                                                          // Enable I2C master, IRQ on byte
Received
                                                          // Enable IRQ on Transmit complete,
Arbitration lost, NACK received, Complete transaction (Stop detected)
                                                          // Decrement Tx FIFO status on byte
full transmitted.
pADI_I2C->I2CMCON = I2CMCON_MASEN + I2CMCON_IENRX
+ I2CMCON_IENTX        + I2CMCON_IENALOST   + I2CMCON_IENNACK
+ I2CMCON_IENCMP
+ 0x400;                                                  // Enable Rx DMA
pADI_I2C->I2CMRXCNT = ucNumBytes;

Dma_Init();

I2CMASTERDMAREAD(uxI2CRXData, ucNumBytes);
pADI_DMA->DMACFG = 0x1;                                   // Enable DMA mode in DMA controller
pADI_DMA->DMAENSET = 0x80;                                // Enable I2C_RX_DMA Channel
NVIC_EnableIRQ(DMA_I2CM_RX_IRQn);                         // I2C Rx DMA interrupt sources-M360
pADI_I2C->I2CADR0 =    0xA1;
}

void I2CMASTERDMAREAD(unsigned char *pucRX_DMA, unsigned int iNumRX)
```

```
{
DmaDesc Desc;

                                                                    // Common configuration of
all the descriptors used here
    desc.ctrlcfg.bits.cycle_ctrl       = cyclectrl_basic;
    desc.ctrlcfg.bits.next_useburst    = 0x0;
    desc.ctrlcfg.bits.r_power          = 0;
    Desc.ctrlCfg.Bits.src_prot_ctrl    = 0x0;
    Desc.ctrlCfg.Bits.dst_prot_ctrl    = 0x0;
    Desc.ctrlCfg.Bits.src_size         = SRCSIZE_BYTE;
    Desc.ctrlCfg.Bits.dst_size         = DSTSIZE_BYTE;

                                                                    //  RX Primary   Descriptor
Desc.srcEndPtr                     = (unsigned int)&I2CMRX;
    Desc.destEndPtr                    = (unsigned int)(pucRX_DMA + iNumRX - 0x1);
    Desc.ctrlCfg.Bits.n_minus_1        = iNumRX - 0x1;
    Desc.ctrlCfg.Bits.src_inc          = INC_NO;
    Desc.ctrlCfg.Bits.dst_inc          = INC_BYTE;
    *Dma_GetDescriptor(DMA_REQ_I2C0_MST_RX,FALSE) = Desc;
}
                                                                    // I2C DMA Rx IRQ handler
void DMA_I2C0_MRX_Int_Handler()
{
NVIC_DisableIRQ(DMA_I2CM_RX_IRQn);                                  // Clear Interupt source
}
```

I²C存储器映射寄存器

I²C接口MMR的基地址为0x40003000。

表128. I²C接口存储器地址表，主机寄存器(基地址：0x40003000)

偏移	名称	描述	访问类型	默认值
0x0000	I2CMCON	主机控制寄存器。读/写寄存器。	RW	0x0000
0x0004	I2CMSTA	主机状态、错误和IRQ寄存器。	R	0x0000
0x0008	I2CMRX	主机数据接收寄存器。	R	0x0000
0x000C	I2CMTX	主机数据发送寄存器。	W	0x0000
0x0010	I2CMRXCNT	主机数据接收计数寄存器。	RW	0x0000
0x0014	I2CMCRXCNT	主机当前数据接收计数寄存器。	R	0x0000
0x0018	I2CADR0	第一主机地址字节寄存器。	RW	0x0000
0x001C	I2CADR1	第二主机地址字节寄存器(仅10位地址)。	RW	0x0000
0x0024	I2CDIV	串行时钟分频寄存器。	RW	0x1F1F

I²C从机映射寄存器

表129. I²C接口存储器地址表，从机寄存器(基地址：0x40003000)

偏移	名称	描述	访问类型	默认值
0x0028	I2CSCON	从机控制寄存器。	RW	0x0000
0x002C	I2CSSTA	从机状态、错误和IRQ寄存器。	R	0x0001
0x0030	I2CSRX	从机数据接收寄存器。	R	0x0000
0x0034	I2CSTX	从机数据发送寄存器。	W	0x0000
0x0038	I2CALT	硬件广播ID寄存器。	RW	0x0000
0x003C	I2CID0	第一从机地址器件ID。	RW	0x0000
0x0040	I2CID1	第二从机地址器件ID。	RW	0x0000
0x0044	I2CID2	第三从机地址器件ID。	RW	0x0000
0x0048	I2CID3	第四从机地址器件ID。	RW	0x0000

I²C映射共享寄存器

表130. I²C接口存储器地址表，共享寄存器(基地址：0x40003000)

偏移	名称	描述	访问类型	默认值
0x004C	I2CFSTA	主机和从机Rx和Tx FIFO状态寄存器。	RW	0x0000

I²C主机控制寄存器

地址：0x40003000；复位：0x0000；名称：I2CMCON

表131. I2CMCON寄存器位功能描述

位	名称	描述
15:12	保留	保留位。
11	TXDMA	使能主机Tx DMA请求。 0: 禁用Tx DMA模式。 1: 使能I²C主机Tx DMA请求。
10	RXDMA	使能主机Rx DMA请求。 0: 禁用Rx DMA模式。 1: 使能I²C主机Rx DMA请求。
9	保留	始终清0。
8	IENCMP	处理完成(或检测到停止条件)中断使能。 0: 禁用。 1: 使能处理完成中断。若此位置位，则检测到停止条件时产生中断。
7	IENNACK	收到NACK中断使能。 0: 禁用。 1: 使能。
6	IENALOST	仲裁丢失中断使能。 0: 禁用。 1: 使能。
5	IENTX	发送请求中断使能。 0: 禁用。 1: 使能。
4	IENRX	接收请求中断使能。 0: 禁用。 1: 使能。
3	保留	保留。应向此位写入0值。
2	LOOPBACK	内部回送使能。离开器件的SCL和SDA复用到其对应的输入上。注意：只要器件地址一致，主机也可以回送传输到从机，即外部回送。 0: 禁用。 1: 使能。
1	COMPETE	启动延时禁用。 0: 禁用。 1: 允许器件争夺所有权，即使另一个器件正在产生一个起始条件。
0	MASEN	主机使能。主机不使用时应予以禁用，从而关闭主机时钟，节省功耗。在处理完成之前，不得将此位清0(参见主机状态寄存器(表132)中的TCOMP位)。注意，此位不会复位可写寄存器位。 0: 禁用主机，主机状态机处于复位状态。 1: 使能主机。

I²C主机状态寄存器

地址：0x40003004；复位：0x0000；名称：I2CMSTA

表132. I2CMSTA寄存器位功能描述

位	名称	描述
15:13	保留	保留位。读取时返回0。
12	TXUR	发送FIFO下溢。 当I²C主机因为Tx FIFO变空条件而结束处理时置1。只有IENTX (I2CMCON[10])置1，此位才会置1。 读取I2CMSTA寄存器时，此位清0。
11	MSTOP	I²C主机驱动的停止条件。 当I²C主机在I²C总线上产生一个停止条件时，此位置1，表示处理完成、Tx下溢、Rx上溢或从机NACK。它不同于TCOMP，因为当由于I²C总线上的任何其它主机而出现停止条件时，此位不会置位。此位不产生中断。关于停止条件相关的可用中断，参见TCOMP描述。 读取I2CMSTA寄存器时，此位清0。
10	LINEBUSY	线路繁忙。 当在I²C总线上检测到起始条件时置1。 当在I²C总线上检测到停止条件时清0。
9	RXOF	接收FIFO上溢。 接收FIFO已满后，又有一个字节写入FIFO时，此位置1。 读取I2CMSTA寄存器时，此位清0。
8	TCOMP	处理完成(或检测到停止条件)。(可产生中断。) 只有使能主机(MASEN = 1)，此位才会置位。应利用此位来确定何时可以安全地禁用主机。当此主机仲裁失效后，它也可以用来等待I²C总线上的另一个主机处理完成。 当在I²C总线上检测到停止条件时置1。如果IENCMP为1 (I2CMCON [8])，则当此位置位时会产生中断。 读取I2CMSTA寄存器时，此位清0。
7	NACKDATA	收到对数据写入的NACK响应。(可产生中断。) 当收到对数据写入传输的NACK响应时，此位置1。如果IENNACK为1 (I2CMCON [7])，则当此位置位时会产生中断。 读取I2CMSTA寄存器时，此位清0。
6	BUSY	主机繁忙。 当主机状态机正在处理事务时，该位置1。 当状态机空闲或者另一器件取得I²C总线控制权时，此位清0。
5	ALOST	仲裁失效。(可产生中断。) 当主机输掉仲裁时，此位置1。如果IENALOST为1 (I2CMCON [6])，则当此位置位时会产生中断。 读取I2CMSTA寄存器时，此位清0。
4	NACKADDR	收到对地址的NACK响应。(可产生中断。) 当收到对地址的NACK响应时，此位置1。如果IENNACK为1 (I2CMCON [7])，则当此位置位时会产生中断。 读取I2CMSTA寄存器时，此位清0。
3	RXREQ	接收请求。(可产生中断。) 当接收FIFO中存在数据时，此位置1。如果IENRX为1 (I2CMCON [4])，则当此位置位时会产生中断。 读取I2CMRX寄存器时，此位清0。
2	TXREQ	发送请求。(可产生中断。) 当方向位为0且发送FIFO为空或不满时，此位置1。如果IENTX为1 (I2CMCON [5])，则当此位置位时会产生中断。 写入I2CMTX寄存器时，此位清0。
1:0	TXFSTA	发送FIFO状态。 00: FIFO空。 01: 保留。 10: FIFO中有1个字节。 11: FIFO满。

I²C主机接收寄存器

地址： 0x40003008；复位： 0x0000；名称： I2CMRX

表133. I2CMRX寄存器位功能描述

位	名称	描述
15:8	保留	保留位。读取时返回0。
7:0	VALUE	接收寄存器。只读。默认置0。此寄存器允许访问接收数据FIFO。该FIFO可容纳2字节。

I²C主机发送寄存器

地址：0x4000300C；复位：0x0000；名称：I2CMTX

I2CMTX是一个只写寄存器。读取时，所得值不确定。

表134. I2CMTX寄存器位功能描述

位	名称	描述
15:8	保留	保留位。读取时返回0。
7:0	VALUE	发送寄存器。默认置0。写入此寄存器会向Tx FIFO加载一个字节。该FIFO可容纳2字节。

I²C主机接收数据计数寄存器

地址：0x400030010；复位：0x0000；名称：I2CMRXCNT

表135. I2CMRXCNT寄存器位功能描述

位	名称	描述
15:9	保留	保留位。读取时返回0。
8	EXTEND	扩展读操作。当一个读操作需要读取的字节数多于256时，使用此位。 0: 禁用扩展读操作。 1: 使能扩展读操作。例如，要接收412字节，应向此寄存器(I2CMRXCNT)写入0x100 (EXTEND = 1)。等待接收第一个字节，然后每收到一个字节便检查I2CMCRXCNT寄存器。当I2CMCRXCNT返回0时，便已收到256字节。 然后向此寄存器(I2CMRXCNT)写入0x09C(412 – 256 = 156(十进制，相当于0x9C)，EXTEND位设为0)。
7:0	COUNT	接收计数。将需要的字节数减1值写入此寄存器。 若只需1个字节，则将0写入此寄存器。 若所需字节数超过256，请使用EXTEND。

I²C主机当前接收计数寄存器

地址：0x40003014；复位：0x0000；名称：I2CMCRXCNT

表136. I2CMCRXCNT寄存器位功能描述

位	名称	描述
15:8	保留	保留位。读取时返回0。
7:0	VALUE	当前接收计数。此寄存器指示目前接收到的总字节数。如果请求256字节，当处理完毕时，此寄存器读出0。

I²C主机第一地址字节寄存器

地址：0x40003018；复位：0x0000；名称：I2CADR0

表137. I2CADR0寄存器位功能描述

位	名称	描述
15:8	保留	保留位。读取时返回0。
7:0	VALUE	地址字节。 如果需要7位地址，则I2CADR0[7:1]设置地址，I2CADR0[0]设置方向(读或写)。 如果需要10位地址，则I2CADR0[7:3]设为11110，I2CADR0[2:1]设置地址的两个MSB，I2CADR0[0]设置方向(读或写)。

I²C主机第二地址字节寄存器

地址：0x4000301C；复位：0x0000；名称：I2CADR1

表138. I2CADR1寄存器位功能描述

位	名称	描述
15:8	保留	保留位。读取时返回0。
7:0	VALUE	地址字节。仅当用10位寻址从机时才需要此寄存器。I2CADR1[7:0]设置地址的低8位。

I²C串行时钟周期分频寄存器

地址：0x40003024；复位：0x1F1F；名称：I2CDIV

表139. I2CDIV寄存器位功能描述

位	名称	描述
15:8	HIGH	串行时钟高电平时间。此寄存器控制时钟高电平时间。 详情见I²C时钟控制部分。
7:0	LOW	串行时钟低电平时间。此寄存器控制时钟低电平时间。 详情见I²C时钟控制部分。

I²C从机控制寄存器

地址：0x40003028；复位：0x0000；名称：I2CSCON

表140. I2CSCON寄存器位功能描述

位	名称	描述
15	保留	保留位。
14	TXDMA	使能从机Tx DMA请求。 0: 禁用DMA模式。 1: 使能I²C从机DMA请求。
13	RXDMA	使能从机Rx DMA请求。 0: 禁用DMA模式。 1: 使能I²C从机DMA请求。
12	IENREPST	重复起始中断使能。 0: 当REPSTART状态位置位时禁用中断。 1: 当REPSTART状态位置位时产生中断。
11	保留	保留。
10	IENTX	发送请求中断使能。 0: 禁用发送请求中断。 1: 使能发送请求中断。
9	IENRX	接收请求中断使能。 0: 禁用接收请求中断。 1: 使能接收请求中断。

位	名称	描述
8	IENSTOP	检测到停止条件中断使能。 0: 禁用停止条件检测中断。 1: 使能停止条件检测中断。
7	NACK	不应答(NACK)下一通信。 0: 默认。 1: 允许不应答(NACK)下一通信。这可以用于如下之类的情况：在24xx I²C串行EEPROM式访问期间，尝试写入系统存储器中的只读位置或不存在的位置。在24xx I²C串行EEPROM式写操作中，它是间接地址，指向不可写的存储位置。
6	保留	保留。应向此位写入0值。
5	EARLYTXR	提早发送请求模式。 0: 允许在方向位(R/W) SCL时钟脉冲的负沿之后立刻产生发送请求。 1: 允许在方向位(R/W) SCL时钟脉冲的正沿之后立刻产生发送请求。
4	GCSBCLR	广播状态位清0。只有写入此位或完全复位，才能复位广播状态和广播ID位。 广播ID = I2CSSTA[9:8]。 0: 禁用广播状态清零。 1: 广播状态和广播ID位清零。
3	HGCEN	硬件广播使能。 0: 禁用硬件广播。 1: 当此位和广播使能位置1时，如果收到一个广播信号、地址0x00和一个数据字节，器件将对I2CALT的内容和接收移位寄存器的内容进行比较。如果数据匹配，表明器件接收到一个硬件广播。当器件需要紧急呼叫一个主机而又不知道呼叫哪一个时，可使用该功能。该广播消息会发送到总线上的所有主机。要求主机注意的器件会将自己的地址嵌入到消息中。I2CALT寄存器的LSB应该始终写入1。
2	GCEN	广播使能。 0: 禁用I²C从机应答(ACK) I²C广播，地址0x00(写操作)。 1: 使能I²C从机应答(ACK) I²C广播，地址0x00(写操作)。
1	ADR10EN	使能10位寻址。从机支持一个10位地址，其存储在I2CID0和I2CID1中，I2C1ID0包含该地址的第一个字节，高5位必须为11110。I2CID2和I2CID3均可写入7位地址。 0: 如果此位清0，从机可以支持4个从机地址，分别在寄存器I2CID0至寄存器I2CID3中设置。 1: 使能10位寻址。
0	SLVEN	从机使能。注意，可写寄存器位不会复位。 0: 禁用从机，所有从机状态机触发器都处于复位状态。 1: 使能从机。

I²C从机状态寄存器

地址：0x4000302C；复位：0x0001；名称：I2CSSTA

表141. I2CSSTA寄存器位功能描述

位	名称	描述
15	保留	保留位。读取时返回0。
14	START	起始和匹配地址。 如果SCL/SDA上检测到起始条件，且下列条件之一为真，则此位置1：器件地址匹配；收到广播(GC = 0000_0000)代码且GC已使能；收到高速(HS = 0000_1XXX)代码。 收到停止或起始条件时，此位清0。
13	REPSTART	重复起始和匹配地址。(可产生中断。) 如果START (I2CSSTA[14])已经置位，然后检测到重复起始条件，则此位置1。 读取时或收到停止条件时，此位清0。读取I2CxSSTA寄存器时，该位也会清0。

位	名称	描述
12:11	IDMAT	器件ID匹配。 接收到的地址匹配ID寄存器0时，设为00。 接收到的地址匹配ID寄存器1时，设为01。 接收到的地址匹配ID寄存器2时，设为10。 接收到的地址匹配ID寄存器3时，设为11。
10	STOP	起始和匹配地址后停止。(可产生中断。) 在上一个起始条件和匹配地址后，如果从机接收到一个停止条件，则此位置1。 如果IENSTOP (I2CSCON[8])置1，当此位置1时，从机中断请求置位。 读取状态寄存器后该位清0。
9:8	GCID	广播ID。广播复位不会清除这些状态位。向GCSBCLR (I2CSCON[4])写入1可将这些位清0。 00: 无广播。 01: 广播复位和程序地址。 10: 广播程序地址。 11: 广播匹配可供选择的ID。
7	GCINT	广播中断。(始终产生中断。) 如果中断是广播复位，所有寄存器恢复为默认值。 如果中断是硬件广播，Rx FIFO将保存广播的第2个字节，它可以与I2CALT寄存器内容进行比较。 从机接收到任何形式的广播后置1。 向GCSBCLR (I2CSCON[4])写入1可将此位清0。
6	BUSY	从机繁忙。 如果从机收到I²C起始条件，则此位置1。 在下列任意条件下，该位被硬件清0：地址与ID寄存器内容不匹配，从机收到I²C停止条件，或重复起始地址不匹配。
5	NACK	从机产生的NACK。用于表示从机用NACK响应其器件地址。 在下列任意条件下，该位置1：如果没有数据要发送且序列是一个从机读取序列，则不应答(NACK)器件地址；从机控制寄存器中的NACK位置1且寻址该器件。 读取I2CSSTA寄存器时，此位清0。
4	RXOF	接收FIFO上溢。 接收FIFO已满后，又有一个字节写入FIFO时，此位置1。 读取I2CSSTA寄存器时，此位清0。
3	RXREQ	接收请求。(可产生中断。) 在读入字节最后一个数据位的SCL时钟脉冲下降沿置1。 接收FIFO不空时置1。 读取或清空接收FIFO时清0。
2	TXREQ	发送请求。(可产生中断。) 如果EARLYTXR = 0，则当收到的传输方向位为高电平时，此位置1。然后，只要发送FIFO未满，此位便保持置位状态。最初，此位在读入方向位的SCL脉冲负沿置位(如果器件地址也匹配)。 如果EARLYTXR = 1，则当收到的传输方向位为高电平时，此位置1。然后，只要发送FIFO未满，此位便保持置位状态。最初，此位在读入方向位的SCL脉冲正沿后置位(如果器件地址也匹配)。 读取I2CSSTA寄存器时，此位清0。
1	TXUR	发送FIFO下溢。 如果主机要求从机发送数据，且在SCL上升沿Tx FIFO为空，则此位置1。 写入I2CSTX寄存器时，此位清0。
0	TXFSEREQ	Tx FIFO状态。EARLYTXR = 1时无效。 只要从机Tx FIFO为空，该位就会置1(默认)。 从机Tx FIFO不为空时，该位清0。

I²C从机接收数据寄存器

地址：0x40003030；复位：0x0000；名称：I2CSRX

表142. I2CSRX寄存器位功能描述

位	名称	描述
15:8	保留	保留位。读取时返回0。
7:0	VALUE	接收寄存器。

I²C从机数据发送寄存器

地址：0x40003034；复位：0x0000；名称：I2CSTX

I2CSTX是一个只写寄存器。 读取此寄存器时，所得值不确定。

表143. I2CSTX寄存器位功能描述

位	名称	描述
15:8	保留	保留位。读取时返回0。
7:0	VALUE	发送寄存器。

I²C从机Alt寄存器

地址：0x40003038；复位：0x0000；名称：I2CALT

表144. I2CALT寄存器位功能描述

位	名称	描述
15:8	保留	保留位。读取时返回0。
7:0	VALUE	此寄存器与HGCEN (I2CSCON[3])一起使用来匹配一个产生硬件广播的主机。 其用在如下场合：主机无法设置从机地址，相反要由从机辨识主机地址。

I²C从机ID寄存器

表145. I2CIDx寄存器位功能描述(I2CID0地址：0x4000303C；I2CID1地址：0x40003040)
(I2CID2地址：0x40003044； I2CID3地址：0x40003048)

位	名称	描述
15:8	保留	保留位。读取时返回0。
7:0	VALUE	此类寄存器共有四个：I2CID0至I2CID3。I2CID[7:1]写入器件ID。I2CID[0]为无关位。 关于如何将10位地址写入这些寄存器的说明，参考表140中的I2CSCON[1]位。

I²C状态寄存器

地址：0x4000304C；复位：0x0000；名称：I2CFSTA

表146. I2CFSTA寄存器位功能描述

位	名称	描述
15:10	保留	保留位。读取时返回0。
9	MFLUSH	0: 正常工作。 1: 清空主机发送FIFO。如果仲裁失效或从机用NACK响应，主机发送FIFO必须清空。
8	SFLUSH	0: 正常工作 1: 清空从机发送FIFO。
7:6	MRXFSTA	主机接收FIFO状态。该状态表示FIFO中的字节数。 00: FIFO空。 01: FIFO中有1个字节。 10: FIFO中有2个字节。 11: 保留。

位	名称	描述
5:4	MTXFSTA	主机发送FIFO状态。该状态表示FIFO中的字节数。 00: FIFO空。 01: FIFO中有1个字节。 10: FIFO中有2个字节。 11: 保留。
3:2	SRXFSTA	从机接收FIFO状态。该状态表示FIFO中的字节数。 00: FIFO空。 01: FIFO中有1个字节。 10: FIFO中有2个字节。 11: 保留。
1:0	STXFSTA	从机发送FIFO状态。该状态表示FIFO中的字节数。 00: FIFO空。 01: FIFO中有1个字节。 10: FIFO中有2个字节。 11: 保留。

串行外设接口

SPI特性

提供了两个完整的具有如下标准的SPI特性硬件串行外设接口：

- 串行时钟相位模式和串行时钟极性模式
- LSB优先传输选项
- 回送模式
- 主机或从机模式
- 传输和中断模式
- 连续传输模式
- Tx/Rx FIFO
- 中断模式，一个、两个、三个或四个字节后中断
- Rx上溢模式和Tx下溢模式
- 开路数据输出模式
- 支持全双工通信(同时发送/接收)

SPI概述

ADuCM360/ADuCM361集成两个完整的硬件串行外设接口(SPI)。PI是一个工业标准同步串行接口，允许8位数据的同步发送和同时接收，即全双工通信。ADuCM360/ADuCM361上实现的两个SPI能以主机和从机模式工作，最大比特率达8 Mbps。

SPI1还具有DMA特性。它有两个DMA通道，其与ARM Cortex-M3通道的μDMA控制器接口。一个DMA通道用于发送，另一个用于接收。

注意，SPI0不支持DMA。

SPI工作模式配置

SPI端口可配置为主机或从机工作模式，由4个引脚组成：MISO、MOSI、SCLK和$\overline{CS}$。

注意：用于SPI通信的GPIO必须配置为SPI模式，然后使能SPI外设；当使用SPI时，应通过GPxPUL寄存器禁用SPI引脚内置的上拉电阻。

MISO(主机输入、从机输出)引脚

在主机模式下，MISO引脚被配置为输入线路；在从机模式下，配置为输出线路。主机上的MISO线路(数据输入)应与从机内的MISO线路(数据输出)相连。传送的数据是以字节(8位)为单位的串行数据，MSB优先。

MOSI(主机输出、从机输入)引脚

在主机模式下，MOSI引脚被配置为输出线路；在从机模式下，配置为输入线路。主机上的MOSI线路(数据输出)应与从机内的MOSI线路(数据输入)相连。传送的数据是以字节(8位)为单位的串行数据，MSB优先。

SCLK(串行时钟输入/输出)引脚

主机串行时钟(SCLK)用于同步通过MOSI SCLK周期发送和接收的数据。所以，发送/接收一个字节需要8个SCLK周期。在主机模式下，SCLK引脚配置成输出端，而在从机模式下，配置成输入端。

在主机模式下，时钟的极性和相位由SPIxCON寄存器控制，SPIxDIV寄存器的值决定了比特率。比特率的计算公式如下：

$$f_{SERIALCLOCK} = \frac{UCLK / DIV}{2 \times (1 + SPIxDIV)}$$

其中，*UCLK/DIV*为16 MHz系统时钟由CLKCON1和CLKSYSDIV寄存器中设置的系数分频后的值。

可以对SPI0和SPI1的时钟分别分频：

- CLKCON1[2:0]和CLKDIS[0]控制SPI0的UCLK/DIV。
- CLKCON1[5:3]和CLKDIS[1]控制SPI1的UCLK/DIV。

通过降低SPI模块的时钟速率，可以降低SPI模块的功耗。

最大数据传输速率是8 Mbps。

在从机模式下，必须用预期输入时钟的相位和极性对SPIxCON寄存器进行配置。从机从外部主机接收数据，速率可达8 Mbps。

在主机模式和从机模式下，数据都在SCLK信号的一个沿发送，并在另一个沿采样。因此，从机时钟的极性和相位必须与主机的配置一致。

片选($\overline{CS}$输入)引脚

在SPI从机模式时，置位$\overline{CS}$引脚将启动数据传输，该引脚是一个低电平有效输入信号。然后，SPI端口开始发送和接收8位数据，直到传输结束为止，此时$\overline{CS}$无效。在从机模式下，$\overline{CS}$总是输入。

在SPI主机模式下，$\overline{CS}$是低电平有效输出信号。传输开始后，它自动置位；传输完成后，它自动解除置位。

SPI传输启动

在主机模式下，传输和中断模式位SPIxCON[6]决定SPI串行传输的启动方式。如果模式位置1，则在写入Tx FIFO后启动串行传输。如果模式位清0，则在读取Rx FIFO后启动串行传输；读操作必须在SPI接口空闲时进行。在活动传输期间执行的读操作不会启动另一次传输。

对于SPIxCON[1]和SPIxCON[6]的任何设置，SPI都会同时接收和发送数据。因此，在数据传输期间，SPI也会接收数据并填充Rx FIFO。如果不从Rx FIFO读取数据，当FIFO开始上溢时，会发生上溢中断。如果用户不想读取Rx数据或接收上溢中断，可将SPIxCON[12]置1，这样接收数据就不会保存到Rx FIFO。

同样，如果用户只想接收数据而不想将数据写入Tx FIFO，可将SPIxCON[13]置1，从而避免接收来自Tx FIFO的下溢中断。

Tx启动的传输

对于由写入Tx FIFO启动的传输，一旦将第一个字节写入FIFO，SPI就会启动传输，而不考虑SPIxCON[15:14]的配置。立即从FIFO读取第一个字节，写入Tx移位寄存器，传输开始。

如果连续传输使能位SPIxCON[11]置1，传输将连续进行，直到Tx FIFO内无有效数据为止。传输之间不存在$\overline{CS}$解除置位的停转周期；在传输期间内，$\overline{CS}$保持置位，直到Tx FIFO变空。确定传输何时停止不取决于SPIxCON[15:14]；当FIFO中没有有效数据时，传输停止。只要FIFO中仍存在有效数据，传输就会继续。

如果连续传输使能位SPIxCON[11]清0，则每次传输仅包括一个8位串行传输。如果Tx FIFO中存在有效数据，那么在一个停转周期($\overline{CS}$解除置位)后启动新传输。

Rx启动的传输

由读取Rx FIFO启动的传输取决于FIFO中收到的字节数。如果SPIxCON[15:14] = 11，读取Rx FIFO时，SPI将启动一个4字节传输。如果设置了连续模式，4个字节将连续传输，字节之间$\overline{CS}$不会解除置位。如果未设置连续模式，各字节传输之间将存在$\overline{CS}$解除置位的停转周期。当SPI正在接收数据时，读取Rx FIFO不会在当前传输完毕后启动另一次传输。

在从机模式下，传输由置位$\overline{CS}$启动。

注意：在从机模式下，仅$\overline{CS}$有效。

器件作为从机发送和接收8位数据，直到传输结束，此时$\overline{CS}$解除置位。

SPI传输协议图(参见图26和图27)显示了SPI的数据传输协议，以及控制寄存器中的CPHA和CPOL位对该协议的影响。

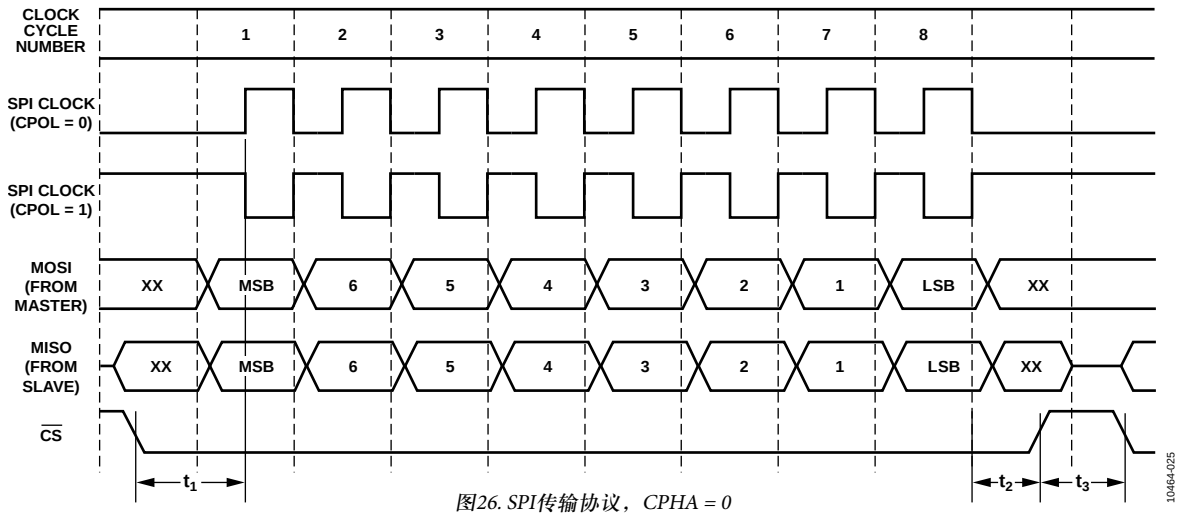


图26. SPI传输协议，CPHA = 0

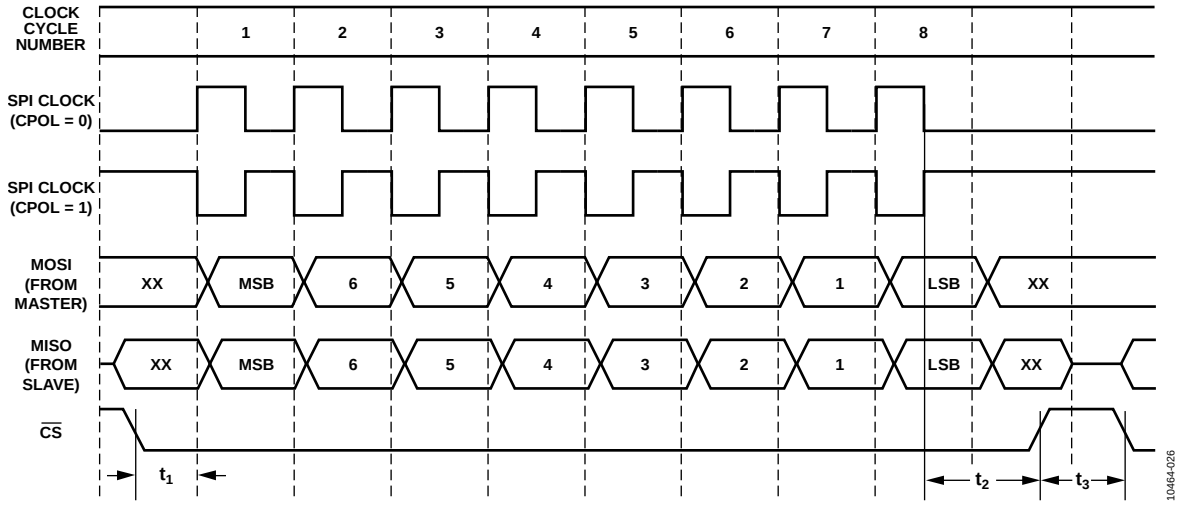


图27. SPI传输协议，CPHA = 1

SPI数据下溢和上溢

如果发送下溢模式位ZEN (SPIxCON[7])清0，则在启动传输后，若FIFO中无有效数据，将移出上次传输的最后字节。如果ZEN置1，则在启动传输后，若FIFO中无有效数据，将传输0。

如果Rx上溢覆盖使能位RXOF (SPIxCON[8])置1，当Rx FIFO中没有空间时，收到的新串行字节将覆盖FIFO中的有效数据。如果RXOF清0，当FIFO中没有空间时，收到的新串行字节将被丢弃。

当Rx FIFO中的有效数据被覆写时，先覆写时间最久的字节，再覆写次久的字节，以此类推。

全双工操作

SPI支持读/写同时进行。

在主机模式下实施全双工传输时，请使用如下程序：

1. 通过一个发送操作在MOSI引脚上启动一个传输序列。设置SPIxCON[6] = 1。如果使能了SPI中断，发生发送中断时将触发中断，但接收字节时不会触发中断。
2. 如果使用中断，SPIxSTA[5]指示的SPI Tx中断或Tx FIFO下溢中断(SPIxSTA[4])将在第一字节传输的约3 × SPICLK到4 × SPICLK周期后置位。如有必要，写入SPIxTX以将一个字节重新载入Tx FIFO。
3. 通过MISO引脚收到第一字节不会更新Rx FIFO状态位(SPIxSTA[10:8])，直到$\overline{CS}$变为低电平的12 × SPICLK周期之后。因此，在能够处理第一接收字节之前，可能会出现两个发送中断。
4. 出现最后一个发送中断之后，可能需要再读取两个字节。处理完最后一个发送中断之后，建议在SPI中断处理程序外部轮询SPIxSTA[10:8]。

SPI中断

每个SPI有一条中断线，并且有四个中断源。SPIxSTA[0]反映中断线的状态，SPIxSTA[7:4]反映四个源的状态。

SPI产生TIRQ或RIRQ。二者不能同时使能。利用TIM位SPIxCON[6]使能适当的中断。TIM = 1时使能TIRQ，TIM = 0时使能RIRQ。

此外，注意SPI0和SPI1中断源必须在NVIC寄存器ISER0中使能(ISER0[18] = SPI0，ISER0[19] = SPI1)。

Tx中断

如果TIM置1，则Tx FIFO状态会引起中断。SPIxCON[15:14]控制中断何时发生，如表147所示。

表147. SPIxCON[15:14] IRQ模式位

SPIxCON[15:14]	中断条件
00	每发送一个字节后产生一个中断。从FIFO读取字节并将其写入移位寄存器时，中断发生。
01	每发送两个字节后产生一个中断。
10	每发送三个字节后产生一个中断。
11	每发送四个字节后产生一个中断。

中断产生取决于发送的字节数，而不是FIFO中的字节数。这与Rx中断不同，后者取决于Rx FIFO中的字节数，而不是接收的字节数。

读取状态寄存器会清除发送中断。此中断的状态可通过读取SPIxSTA[5]得知。若SPIxCON[13]保持高电平，则禁用中断。

写入控制寄存器SPIxCON会将发送字节计数器复位到0。例如，若SPIxCON[15:14]设置为11，且在发送3个字节后写入SPIxCON，则Tx中断要等到再发送4个字节时产生。

Rx中断

如果SPIxCON[6]清0，则Rx FIFO状态会引起中断。SPIxCON[15:14]控制中断何时发生。读取SPIxSTA会清除中断。此中断的状态可通过读取SPIxSTA[6]得知。

中断仅在写入数据到FIFO时产生。例如，若SPIxCON[15:14]设置为0x00，则收到第一字节后就会产生中断。读取状态寄存器时，中断变为无效。若未从FIFO读取该字节，不会重新产生中断。只有FIFO再收到一个字节，才会产生另一个中断。

中断取决于FIFO中的有效字节数，而不是接收的字节数。例如，当SPIxCON[15:14]设置为0x01时，若FIFO中有两个或更多字节，则收到一个字节后就会产生中断。不是每收到两个字节产生一个中断。

若SPIxCON[12]保持高电平，则禁用中断。

下溢/上溢中断

SPIxSTA[7]和SPIxSTA[4]产生SPI中断。

若在Tx FIFO中没有数据的情况下启动传输，SPIxSTA[4]会置1，指示下溢状况。这会引起中断。读取状态寄存器会清除中断(和状态位)。无论SPIxCON[15:14]如何设置，此中断都会发生。将SPIxCON[13]置1时，此中断禁用。

当收到数据且Rx FIFO已满时，SPIxSTA[7]会置1，指示上溢状况。这会引起中断。读取状态寄存器会清除中断(和状态位)。无论SPIxCON[15:14]如何设置，此中断都会发生。将SPIxCON[12]置1时，此中断禁用。

读取状态寄存器或SPIxCON[0]解除置位时，所有中断都会被清除。如果相关清空位置位，Rx和Tx中断也会被清除。否则，即使重新配置SPI，中断也会继续有效。

线或模式(WOM)

在多主机或多从机系统中使用SPI时，为防止竞争，数据输出引脚MOSI和MISO可配置为像开路驱动器那样工作。选择此特性时，需要一个外部上拉电阻。WOM位(SPIxCON[4])控制数据线的焊盘使能输出。

CSERR状况

CSERR位(SPIxSTA[12])指示所有8个SCLK周期完成之前，是否检测到$\overline{CS}$信号错误解除置位情况。此位会产生中断，并且在所有工作模式下均可用：从机、主机和DMA传输期间。

如果CSERR位(SPIxSTA[12])产生中断，应禁用SPI ENABLE位(SPIxCON[0])并重启，实现干净的恢复。这样可以确保随后的传输不出错。在从机模式和主机模式下，BCRST位(SPIxDIV[7])应始终置1，除非SPI通信需要中间字节停转。这种情况下，CSERR标志置1，但可以忽略。

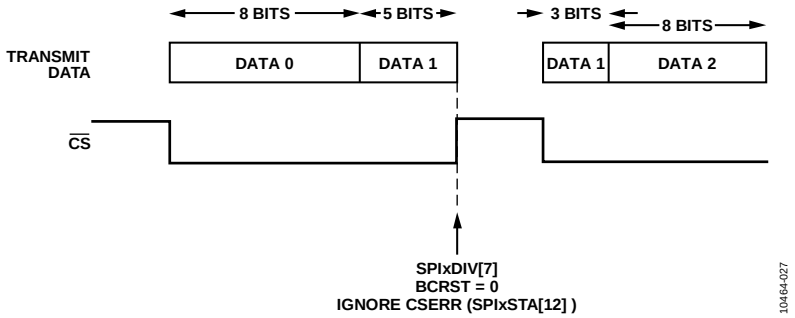


图28. SPI通信：中间字节停转

注意：SPI只能在$\overline{CS}$信号为高电平时重新使能。

SPI1 DMA

仅SPI1通道提供DMA操作。两个DMA通道专门用于发送和接收。SPI1 DMA通道应在ARM Cortex-M3通道的µDMA控制器中配置。

将SPI1DMA寄存器中的DMA请求位设置为接收或发送，可以使能一个通道或同时使能两个通道上的DMA请求。若仅使能DMA发送请求(SPI1DMA[1])，则Rx FIFO在SPI传输期间会上溢并产生上溢中断，除非用户代码读取收到的数据。为避免产生上溢中断，Rx FIFO清空位应置1，或者在NVIC中禁用SPI中断。若仅使能DMA接收请求(SPI1DMA[2])，Tx FIFO会下溢。为避免产生下溢中断，应禁用SPI中断。

使用DMA时不会产生SPI Tx (SPI1STA[5])和SPI Rx (SPI1STA[6])中断。使用DMA时会产生SPI TXUR (SPI1STA[4])和RXOF (SPI1STA[7])中断。SPI1CON[15:14]在发送模式下不使用，在接收模式下应设为0x00。

ENABLE位(SPI1DMA[0])控制DMA传输的开始。只有ENABLE = 1才会产生DMA请求。DMA传输结束时，也就是收到DMA SPI传输中断时，此位需清0，以防对µDMA控制器产生额外的DMA请求。若在Tx模式下，Tx FIFO中留有的数据会被发送。

DMA主机发送配置

应配置DMA SPI Tx通道。

要使能DMA Tx主机中断(ISER0[23])，应配置NVIC。

SPI模块应按如下方式进行配置：

```
pADI_SPI1->SPIDIV = SPI_serial_freq;                    //Configures serial clock frequency
where Fserial clock = fuclk/(2*(1+SPIDIV))
pADI_SPI1->SPIDCON = 0x1043;                            //Enable SPI in master mode and transmit
mode, Rx FIFO flush enabled.
pADI_SPI1->SPIDMA = 0x3;                                //Enable DMA mode, enable Tx DMA request
```

当DMA缓冲器中的所有数据都发送完毕时，DMA产生中断。用户代码应禁用DMA请求。数据仍在Tx FIFO中，因为只要Tx FIFO中有自由空间，就会产生DMA请求，使得FIFO总是保持填满状态。用户代码可在FIFO状态寄存器中检查FIFO中仍有多少字节。

示例代码：初始化SPI1 DMA发送序列

```
void SPIDMAINIT(unsigned char ucMasterHSlaveL, unsigned int uiBaudrate)
{
pADI_GP0->GPCON = 0x55;                         // Configure P0.[3:0] as SPI pins
                                                // Slave only implemented
if ( ucMasterHSlaveL == 0)
{
pADI_SPI1->SPICON = 0xB85;
}
else
pADI_SPI1->SPICON = 0xBC7;
pADI_SPI1->SPIDMA = 0x7;                        // Enable DMA + SPI Rx/Tx DMA
pADI_DMA->DMACFG = 0x1;                         // Enable DMA mode in DMA controller

Dma_Init();
pADI_SPI1->SPICON |=0x3000;                     // Flush Rx/Tx FIFOs
pADI_SPI1->SPICON &=0xCFFF;                     // Flush Rx/Tx FIFOs
SPIDMAWRITE(uxSPITXData, 16);
pADI_DMA->DMAENSET = 0x103;
NVIC_EnableIRQ(DMA_SPI1_RX_IRQn);
NVIC_EnableIRQ(DMA_SPI1_TX_IRQn);

                                                // Enable DMA SPI1_RX/SPI1Tx Interrupt
}
void SPIDMAWRITE(unsigned char *pucTX_DMA, unsigned int iNumRX)
{
   DmaDesc Desc;
                                                // Common configuration of all the descriptors
used here
    Desc.ctrlCfg.Bits.cycle_ctrl       = cyclectrl_basic;
    desc.ctrlcfg.bits.next_useburst    = 0x0;
    desc.ctrlcfg.bits.r_power          = 0;
    desc.ctrlcfg.bits.src_prot_ctrl    = 0x0;
    Desc.ctrlCfg.Bits.dst_prot_ctrl    = 0x0;
    Desc.ctrlCfg.Bits.src_size         = SRCSIZE_BYTE;
    Desc.ctrlCfg.Bits.dst_size         = DSTSIZE_BYTE;

                                                //  TX Primary   Descriptor
Desc.srcEndPtr                         = (unsigned int)(pucTX_DMA + iNumRX - 0x1);
    Desc.destEndPtr                    = (unsigned int)&SPI1TX;
    Desc.ctrlCfg.Bits.n_minus_1        = iNumRX - 0x1;
    Desc.ctrlCfg.Bits.src_inc          = INC_BYTE;
    Desc.ctrlCfg.Bits.dst_inc          = INC_NO;
    *Dma_GetDescriptor(DMA_REQ_SPI1_TX,FALSE) = Desc;
                       pADI_SPI1->SPITX = *pucTX_DMA;
}
```

DMA主机接收配置

SPI1CNT寄存器仅适用于DMA接收主机模式。它设置SPI主机需要接收的字节数，或主机需要产生的时钟数。当收到需要的字节数时，就不会启动更多传输。要启动DMA主机接收传输，用户代码应完成一次伪读取。应将此伪读取增加到SPI1CNT数目上。

当SPI1CON[0]禁用SPI时，或者用户代码更改SPI1CNT寄存器时，计数接收字节数的计数器复位。

执行SPI1 DMA主机接收

应配置DMA SPI Rx通道。

要使能DMA Rx主机中断(ISER0[24])，应配置NVIC。

SPI模块应按如下方式进行配置：

```
pADI_SPI1->SPIDIV = SPI_serial_freq;                    // Configures serial clock frequency where
```

$F_{serial\ clock} = f_{uclk} / (2*(1+SPIDIV))$

```
pADI_SPI1->SPICON = 0x2003;                             // Enable SPI in master mode and receive
mode, 1 byte transfer.
pADI_SPI1->SPIDMA = 0x5;                                // Enable DMA mode, enable Rx DMA request
pADI_SPI1->SPICNT = XXX;                                // Number of bytes to transferred + 1
A =;                                                    // Dummy read
```

传输完所需字节数时，DMA传输便停止。注意：为在传输完成时产生DMA中断，DMA缓冲器与SPI1CNT必须大小一致。

示例代码：初始化SPI1 DMA接收序列

```
void SPIDMAINIT(unsigned char ucMasterHSlaveL, unsigned int uiBaudrate)
{
pADI_GP0->GPCON = 0x55;                                 // Configure P0.[3:0] as SPI pins
                                                        // Slave only implemented
if ( ucMasterHSlaveL == 0)
{
= 0xB85;
}
else
pADI_SPI1->SPICON = 0xBC7;
= 0x7;                                                  // Enable DMA + SPI Rx/Tx DMA
pADI_DMA->DMACFG = 0x1;                                 // Enable DMA mode in DMA controller

Dma_Init();
pADI_SPI1->SPICON |=0x3000;                             // Flush Rx/Tx FIFOs
&=0xCFFF;                                               // Flush Rx/Tx FIFOs
SPIDMAREAD (uxSPIRXData, 16);
pADI_DMA->DMAENSET = 0x103;
                                                        // Enable DMA SPI1_RX/SPI1Tx Interrupt
NVIC_EnableIRQ(DMA_SPI1_RX_IRQn);
NVIC_EnableIRQ(DMA_SPI1_TX_IRQn);

}
void SPIDMAREAD(unsigned char *pucRX_DMA, unsigned int iNumRX)
{
   DmaDesc Desc;
                                                // Common configuration of all the descriptors
used here
    desc.ctrlcfg.bits.cycle_ctrl       = cyclectrl_basic;
    desc.ctrlcfg.bits.next_useburst    = 0x0;
    desc.ctrlcfg.bits.r_power          = 0;
    Desc.ctrlCfg.Bits.src_prot_ctrl    = 0x0;
    Desc.ctrlCfg.Bits.dst_prot_ctrl    = 0x0;
    Desc.ctrlCfg.Bits.src_size         = SRCSIZE_BYTE;
    Desc.ctrlCfg.Bits.dst_size         = DSTSIZE_BYTE;

                                               //  RX Primary   Descriptor
    Desc.srcEndPtr                     = (unsigned int)&SPI1RX;
    Desc.destEndPtr                    = (unsigned int)(pucRX_DMA + iNumRX - 0x1);
    Desc.ctrlCfg.Bits.n_minus_1        = iNumRX - 0x1;
    Desc.ctrlCfg.Bits.src_inc          = INC_NO;
    Desc.ctrlCfg.Bits.dst_inc          = INC_BYTE;
    *Dma_GetDescriptor(DMA_REQ_SPI1_RX,FALSE) = Desc;
}
```

SPI和关断模式

在主机模式下，进入关断模式之前，建议通过SPIxCON[0]禁用SPI模块。在从机模式下，无论是中断驱动还是DMA工作模式，都应通过GPIO寄存器检查$\overline{CS}$线电平，确保SPI不在进行通信，并且在$\overline{CS}$线为高电平时禁用SPI模块。上电时可以重新使能SPI模块。

SPI存储器映射寄存器

SPI0接口MMR的基地址为0x40004000。SPI1接口MMR的基地址为0x40004400。

表148. SPI外设存储器地址表(SPI0基地址：0x40004000；SPI1基地址：0x40004400)

偏移	名称[1]	描述	访问类型	默认值
0x0000	SPIxSTA	状态寄存器	R	0x0000
0x0004	SPIxRX	8位接收寄存器	R	0x0000
0x0008	SPIxTX	8位发送寄存器	W	0x0000
0x000C	SPIxDIV	16位比特率选择寄存器	RW	0x0000
0x0010	SPIxCON	16位配置寄存器	RW	0x0000
0x0014	SPI1DMA	DMA使能寄存器(仅通道SPI1提供DMA操作)	RW	0x0000
0x0018	SPIxCNT	8位接收字节计数寄存器	R	0x0000

[1] x为0或1，对应于SPI0或SPI1。

SPI状态寄存器

地址：0x40004000；复位：0x0000；名称：SPI0STA

地址：0x40004400；复位：0x0000；名称：SPI1STA

表149. SPIxSTA寄存器位功能描述

位	名称	描述	
15:13	保留	保留位。	
12	CSERR	检测到$\overline{CS}$突然解除置位。 当$\overline{CS}$线突然解除置位时(甚至完整数据字节尚未传输完毕)，此位置1。此位会引起中断。如果CSERR位置1，建议将SPIxCON寄存器中的ENABLE位清0，以实现干净的恢复。 读取SPISTA寄存器时清0。	
11	RXS	SPI Rx FIFO存在过剩字节。表示Rx FIFO中的字节数多余Rx中断指示的字节数。它取决于SPIxCON[15:14]。FIFO中的字节数不超过SPIxCON[15:14]中的字节数时，此位清0。读取SPISTA寄存器时，该位不清0。此位不依赖于SPIxCON[6]，且不会引起中断。	
		SPIxCON[15:14]	RXS
		00	如果Rx FIFO中有两个或更多字节，则RXS置1。
		01	如果Rx FIFO中有三个或更多字节，则RXS置1。
		10	如果Rx FIFO中有四个或更多字节，则RXS置1。
		11	RXS不置1。
10:8	RXFSTA	SPI Rx FIFO状态位。 000: Rx FIFO为空时设为000。 001: FIFO中有一个有效字节时设为001。 010: FIFO中有两个有效字节时设为010。 011: FIFO中有三个有效字节时设为011。 100: FIFO中有四个有效字节时设为100。	
7	RXOF	SPI Rx FIFO上溢状态位(中断)。 Rx FIFO已满，再次向FIFO载入新数据时，此位置1。除非SPIxCON寄存器的RFLUSH位置1，否则该位会产生中断。 读取SPIxSTA寄存器时清0。	
6	RX	SPI Rx IRQ状态位。DMA模式下不可用。 产生接收中断时置1。SPIxCON的TIM清0时，在接收到所需字节数后，此位置1。 读取SPISTA寄存器时清0。	

位	名称	描述
5	TX	SPI Tx IRQ状态位。DMA模式下不可用。 产生发送中断时置1。SPIxCON的TIM置1时，在发送所需字节数后，此位置1。 读取SPIxSTA寄存器时清0。
4	TXUR	SPI Tx FIFO下溢(中断)。 当启动发送但Tx FIFO内没有有效数据时，此位置1。 除非SPIxCON寄存器的TFLUSH位置1，否则该位会产生中断。 读取SPIxSTA寄存器时清0。
3:1	TXFSTA	表示SPI Tx FIFO中有多少有效字节。 000: Tx FIFO为空时设为000。 001: FIFO中有一个有效字节时设为001。 010: FIFO中有两个有效字节时设为010。 011: FIFO中有三个有效字节时设为011。 100: FIFO中有四个有效字节时设为100。
0	IRQ	SPI中断状态位。 SPI中断发生时，该位被置1。 读取SPIxSTA后清0。

SPI接收寄存器

地址：0x40004004；复位：0x0000；名称：SPI0RX

地址：0x40004404；复位：0x0000；名称：SPI1RX

表150. SPIxRX寄存器位功能描述

位	名称	描述
15:8	保留	这些位保留，应由用户代码写入0。
7:0	VALUE	8位接收寄存器。读取Rx FIFO返回要从FIFO读取的下一个字节。读取空FIFO时返回0。

SPI发送寄存器

地址：0x40004008；复位：0x0000；名称：SPI0TX

地址：0x40004408；复位：0x0000；名称：SPI1TX

表151. SPIxTX寄存器位功能描述

位	名称	描述
15:8	保留	这些位保留，应由用户代码写入0。
7:0	VALUE	8位发送寄存器。写入Tx FIFO地址空间会将数据写入Tx FIFO中下一可用位置。 如果FIFO已满，FIFO中时间最久的数据字节会被覆写。读取此地址位置返回0。

SPI时钟分频寄存器

地址：0x4000400C；复位：0x0000；名称：SPI0DIV

地址：0x4000440C；复位：0x0000；名称：SPI1DIV

表152. SPIxDIV寄存器位功能描述

位	名称	描述
15:7	保留	这些位保留，应由用户代码写入0。
7	BCRST	CSERR的复位模式。此位用于配置$\overline{CS}$突然解除置位后SPI接口逻辑的预期行为。 0: SPI接口逻辑从停止处继续。当$\overline{CS}$置位时，SPI可接收剩余的位，用户代码必须忽略CSERR中断。 1: 发生CSERR状态后，SPI接口逻辑复位，用户代码应将SPIxCON的SPI ENABLE位清0。
6	保留	保留。

位	名称	描述
5:0	DIV	用于对UCLK分频以产生串行时钟的系数：$f_{SERIAL\ CLOCK} = f_{UCLK/DIV}/(2\times(1 + SPIDIV))$。 串行时钟最大频率为½ UCLK/DIV。这些位仅用于主机模式。在从机模式下， 无需设置串行时钟频率。SPI从主机接收时钟。 对于SPI0，UCLK/DIV由CLKCON1[2:0]和CLKDIS[0]设置。 对于SPI1，UCLK/DIV由CLKCON1[5:3]和CLKDIS[1]设置。

注意：设置SPI串行时钟时必须考虑FCLK频率。FCLK不得小于SPI串行时钟频率的一半。

例如，若SPI时钟分频寄存器设为0x0000(SCLK频率 = ½ UCLK/DIV频率)，则CLKCON0中的CD位可以设置的最大值为2(FCLK频率 = ¼ UCLK/DIV频率)。

SPI控制寄存器

地址：0x40004010；复位：0x0000；名称：SPI0CON

地址：0x40004410；复位：0x0000；名称： SPI1CON

表153. SPIxCON寄存器位功能描述

位	名称	描述	
15:14	MOD	SPI IRQ模式位。如果TIM置1，当传输过程中发生Tx/Rx中断时配置这些位。 如果使用SPI1 DMA模式，这些位必须为00。	
		MOD	**功能**
		00	TX1RX1：传输完一个字节时，产生Tx中断；FIFO接收到一个或以上字节时，产生Rx中断。
		01	TX2RX2：传输完两个字节时，产生Tx中断；FIFO接收到两个或以上字节时，产生Rx中断。
		10	TX3RX3：传输完三个字节时，产生Tx中断；FIFO接收到三个或以上字节时，产生Rx中断。
		11	TX4RX4：传输完四个字节时，产生Tx中断；FIFO已满或有四个字节时，产生Rx中断。
13	TFLUSH	SPI Tx FIFO清空使能位。 0: 禁用Tx FIFO清空。 1: 清空Tx FIFO。该位无法自清0；需要单次清空操作时，应反转该位。如果该位保持为1，则将发送最后被发送的值或0x00，具体取决于ZEN位(SPIxCON[7])的值。该位为1时，无法对Tx FIFO进行写操作。	
12	RFLUSH	SPI Rx FIFO清空使能位。该位置1后，所有传入的数据都将被忽略，且不产生中断。 如果此位清0且TIM = 0，则读取Rx FIFO将启动一次传输。 0: 禁用Rx FIFO清空。 1: 清空Rx FIFO。该位无法自清0；需要单次清空操作时，应反转该位。	
11	CON	连续传输使能。 0: 禁用连续传输。每一次传输都是单独的8位串行传输。如果SPITX寄存器中存在有效数据，那么在一个串行时钟停转周期后会重新开始发送数据。在这一个串行时钟周期内，$\overline{CS}$线无效。 1: 使能连续传输。在主机模式下，数据传输连续进行，直到发送寄存器内无有效数据为止。$\overline{CS}$置位，并在每一次8位串行传输期间保持置位，直到发送寄存器为空。	
10	LOOPBACK	回送使能位。 0: 正常模式。 1: MISO连接到MOSI，因此，Tx寄存器发送的数据回送到Rx寄存器。 为使回送模式正常工作，MASEN位(SPIxCON [1])必须置1。	
9	OEN	从机MISO输出使能位。 0: 禁用MISO引脚上的输出驱动器。此位清0后，MISO引脚变为开路。 1: MISO正常工作。	
8	RXOF	SPI Rx上溢覆盖使能。 0: 新接收到的串行数据会被丢弃。 1: 新接收到的串行字节覆盖Rx寄存器中的有效数据。	

位	名称	描述
7	ZEN	Tx FIFO为空时，SPI发送0。 0: 当Tx FIFO中无有效数据时，发送上次发送的值。 1: 当Tx FIFO中无有效数据时，发送0x00。
6	TIM	SPI传输和中断模式。 0: RXRD: 通过读取SPI Rx寄存器启动传输。当TIM = 0时，要启动一次传输， 读操作必须在SPI接口处于空闲模式时完成。只有当Rx已满时，才产生中断。 1: TXWR: 通过写入SPI Tx寄存器启动传输。只有当Tx为空时，才产生中断。
5	LSB	LSB优先传输使能位。 0: MSB优先传输。 1: LSB优先传输。
4	WOM	SPI线或模式使能位。 0: 正常输出电平。 1: 开路数据输出使能。 在主机模式下： 当MOSI引脚传输0时，使能输出驱动器。 当MOSI引脚传输1时，禁用输出驱动器，并且需要外部上拉电阻来将该引脚拉至高电平。 典型电阻值为1 kΩ。 在从机模式下： 当MISO引脚传输0时，使能输出驱动器。 当MISO引脚传输1时，禁用输出驱动器，并且需要外部上拉电阻来将该引脚拉至高电平。 典型电阻值为1 kΩ。
3	CPOL	串行时钟极性模式位。 0: 串行时钟空闲低电平。 1: 串行时钟空闲高电平。
2	CPHA	串行时钟相位模式位。 0: 串行时钟脉冲出现在第一个数据位传输的中间。 1: 串行时钟脉冲出现在第一个数据位的开始。
1	MASEN	主机模式使能位 0: 使能从机模式。 1: 使能主机模式。
0	ENABLE	SPI使能位。 0: 禁用SPI。此位清0也会复位所有FIFO相关逻辑和输出位移位计数器，以实现干净的启动。 1: 使能SPI。

请注意：

- 更改配置时，务必不要在数据传输期间进行，以免破坏数据。建议在该模块禁用时(禁用SPI，ENABLE = 0)更改配置，然后重新配置和使能SPI (ENABLE = 1)。
- 从从机模式更改为主机模式或相反时，两个FIFO均须为空。
- 使用DMA时，MOD (SPIxCON[15:14])设置不相关。
- 虽然中断产生逻辑与MOD (SPIxCON[15:14])无关，但主机传输启动逻辑仍要依赖MOD。然而，DMA服务仅逐字节发生，而不是突发。因此，对于DMA，只能使用MOD = 0的值。

SPI1 DMA使能寄存器(仅SPI1)

地址：0x40004414；复位：0x0000；名称：SPI1DMA

表154. SPI1DMA寄存器位功能描述

位	名称	描述
15:3	保留	这些位保留，应由用户代码写入0。
2	IENRXDMA	0: 默认清0。 1: 使能接收DMA请求。
1	IENTXDMA	0: 默认清0。 1: 使能发送DMA请求。
0	ENABLE	0: DMA传输结束时，由用户代码清0。 1: 启动DMA传输。需要将此位清0，防止对μDMA控制器产生额外的DMA请求。

请注意：

- 当DMA ENABLE = 0 (SPI1DMA[0])时，不会产生DMA请求。
- DMA传输结束时，必须将此位复位，防止对μDMA控制器产生额外的DMA请求。
- 当ENABLE (SPI1DMA[0]) = 1时，Tx (SPI1STA[5])和Rx (SPI1STA[6])中断自动禁用。然而，TXUR (SPI1STA[4])和RXOF (SPI1STA[7])中断仍然可用，分别指示Tx FIFO下溢和Rx FIFO上溢。
- 使用SPI1 DMA模式时，SPI1CON[15:14]必须清零为0x00。

SPI接收字节计数寄存器

地址： 0x40004018；复位： 0x0000；名称： SPI0CNT

地址： 0x40004418；复位： 0x0000；名称： SPI1CNT

表155. SPIxCNT寄存器位功能描述

位	名称	描述
15:8	保留	这些位保留，应由用户代码写入0。
7:0	VALUE	主机模式需要的Rx字节数。Tx模式下禁用。当收到需要的字节数时， 就不会启动更多传输。如果SPI禁用，或者SPIxCON[0] / SPI1CNT更新， 计数接收字节数的计数器会复位。

UART串行接口

UART特性

- 工业标准16450 UART外设
- 支持DMA

UART概述

UART外设是全双工通用异步接收器/发射器，兼容工业标准16450。UART负责数据的串行格式和并行格式转换。串行通信遵循异步协议，支持各种字长、停止位以及奇偶校验生成等选项。

该UART还包含调制解调器控制和中断处理硬件。该UART有一个小数分频器，有利于高精度波特率的生成。

可从多种独特事件产生中断，如数据缓冲器满/空、传输错误检测和断开检测等。

UART工作原理

串行通信

其遵循异步串行通信协议，具有如下选项：

- 5到8个数据位
- 1、2或1½个停止位
- 无、偶数或奇数奇偶校验
- 波特率 = UCLK/DIV ÷ (2 × 16 × COMDIV) ÷ (M + N ÷ 2048)，其中COMDIV = 1至65536，M = 1至3，N = 0至2047
- UCLK/DIV为分频16 MHz时钟，通过CLKCON1和CLSYSDIV配置

所有数据字都需要一个起始位和至少一个停止位。这样，每个字具有7到12位。写入发送保持寄存器(COMTX)，即可启动发送操作。经过一个同步延迟后，数据移入内部发送移位寄存器(TSR)，并在该寄存器中以UCLK ÷ (2 × 16 × COMDIV) ÷ (M + N ÷ 2048)的波特率(比特率)移出，同时按要求添加起始位、停止位和奇偶校验位。所有数据字都从一个趋低起始位开始。COMTX到TSR的传输引起发送寄存器变空状态标志置1。

除停止位数始终为1外，接收操作使用的数据格式与发送配置相同。检测到起始位后，接收字移位至内部接收移位寄存器(RSR)。收到正确的位数(包括停止位)后，数据和相关状态会更新，RSR传输到接收缓冲寄存器(COMRX)。接收到的字传输至接收缓冲器并经过适当的同步延迟后，接收缓冲寄存器已满状态标志更新。

一个速率等于波特率16倍的采样时钟用来对数据进行采样，采样时刻尽可能接近位的中点。还有一个接收滤波器，用来移除小于采样时钟周期两倍的杂散脉冲。

注意，数据以LSB优先方式进行发送和接收。这与用户认为的传输方式可能不一致。然而，这是协议规定的标准方式。

为了省电，可以通过CLKCON1寄存器(CLKCON1[11:9]和CLKDIS[3])降低UART模块的时钟。UART时钟默认禁用(CLKDIS[3] = 1)。

编程I/O模式

这种模式下，软件负责将数据移入移出UART。这通常是通过中断服务例程实现的，其响应发送和接收中断，读取或写入适当的数据。这种模式对软件有一些限制，软件必须在一定时间内响应，以防止接收通道发生上溢错误。

处理器需要频繁地轮询状态标志，因此通常不使用这种模式，除非系统能够忍受相关开销。中断可利用COMIEN寄存器禁用。

在COMTX非空时写入COMTX，或在COMRX未满时读取COMRX，会产生错误结果，应避免这样做。对于前者，COMTX会被新字覆盖，原字得不到发送。对于后者，先前收到的字会再次被读取。必须在软件中通过正确使用中断或状态寄存器轮询，避免这两种错误。 硬件不会检测这些错误。

使能/禁用位

在ADuCM360/ADuCM361进入关断模式之前，建议禁用串行接口。UART控制寄存器中提供了一位来禁用UART串行外设。此位可禁用该外设的时钟。此位置1时，必须在软件中采取措施，确保不发送或接收任何数据。如果在通信过程中将此位置1，数据传输将不完整，接收或发送寄存器仅包含一部分数据。还建议设置CLKCON1[11:9] = 111，以使关断模式下的功耗最低。

中断

UART外设的Rx和Tx中断均通过一个中断输出连接中断控制器。要确定中断原因，必须由软件读取COMIIR寄存器。注意在DMA模式下，断开和调制解调器状态中断不可用。

在I/O模式下，当接收时，下列情况会引起中断：

- COMRX已满
- 接收上溢错误
- 接收奇偶校验错误
- 接收帧错误
- 断开中断(UART RxD保持低电平)
- 调制解调器状态中断(变为CTS)
- COMTX变空

缓冲器要求

此UART采用双缓冲(保持寄存器和移位寄存器)。

DMA模式

这种模式下，不是由用户代码将数据移入移出UART。进入外部DMA模块的DMA请求信号指示UART已准备好发送或接收数据。这些DMA请求信号可利用COMIEN寄存器禁用。

DMA模式不使用调制解调器功能。

设置UART接收DMA通道的示例代码

```
void UARTRXDMAINIT(void)
{
pADI_UART->COMCON = 0;
pADI_UART->COMIEN = 0x20;
                                                    // Enable DMA Tx transfers
pADI_UART->COMLCR = COMLCR_WLS_8BITS + COMLCR_STOP;  // 8-data bits + 1 Stop bit.
pADI_UART->COMDIV = 0x11;
                                                    // Set UART Baud rate
pADI_UART->COMDIV = 0x11;
                                                    // COMDIV = 17
pADI_UART->COMFBR = COMFBR_ENABLE_EN + 0x1883;      // DIVM = 3, DIVN = 131.
                      |= 0x9000;
                                                    // Configure P0.7/P0.6 for UART
                     Dma_Init();

                     UARTDMAREAD(uxUARTRXData, 4);
                     pADI_DMA->DMACFG = 0x1;
                                                    // Enable DMA mode in DMA controller
                     pADI_DMA->DMAENSET = 0x8;
                                                    // Enable UART_RX_DMA Channel
                                                    // UART Rx DMA interrupt sources -
M360
}
void UARTDMAREAD(unsigned char *pucRX_DMA, unsigned int iNumRX)
```

```
{
   DmaDesc Desc;
                                                        // Common configuration of all the
descriptors used here
    Desc.ctrlCfg.bits.cycle_ctrl       = cyclectrl_basic;
    desc.ctrlcfg.bits.next_useburst    = 0x0;
    desc.ctrlcfg.bits.r_power          = 0;
    Desc.ctrlCfg.Bits.src_prot_ctrl    = 0x0;
    Desc.ctrlCfg.Bits.dst_prot_ctrl    = 0x0;
    Desc.ctrlCfg.Bits.src_size         = SRCSIZE_BYTE;
    Desc.ctrlCfg.Bits.dst_size         = DSTSIZE_BYTE;
                                                        //  RX Primary   Descriptor
    Desc.srcEndPtr                     = (unsigned int)&COMRX;
    Desc.destEndPtr                    = (unsigned int)(pucRX_DMA + iNumRX - 0x1);
    Desc.ctrlCfg.Bits.n_minus_1        = iNumRX - 0x1;
    Desc.ctrlCfg.Bits.src_inc          = INC_NO;
    Desc.ctrlCfg.Bits.dst_inc          = INC_BYTE;
    *Dma_GetDescriptor(DMA_REQ_UART_RX,FALSE) = Desc;
}
                                                        // UART DMA Rx IRQ handler
void DMA_UART_RX_Int_Handler()
{
NVIC_DisableIRQ(DMA_UART_RX_IRQn);                      // Clear Interrupt source
}
```

配置UART发送DMA通道的示例代码

```
void UARTTXDMAINIT(void)
{
  pADI_UART->COMCON = 0;
  pADI_UART->COMIEN = 0x10;                             // Enable DMA Tx transfers
  pADI_UART->COMLCR = COMLCR_WLS_8BITS + COMLCR_STOP;   // 8-data bits + 1 Stop bit.
  pADI_UART->COMDIV = 0x11;
  // Set UART Baud rate
  pADI_UART->COMFBR = COMFBR_ENABLE_EN + 0x1883;        // DIVM = 3, DIVN = 131.
  pADI_GP0->GPCON |= 0x3C;                              // Configure P0.1/P0.2 for UART
Dma_Init();
UARTDMAWRITE(uxUARTTXData, 16);
pADI_DMA->DMACFG = 0x1;                                 // Enable DMA mode in DMA controller
pADI_DMA->DMAENSET = 0x4;                               // Enable UART_TX_DMA Channel
NVIC_EnableIRQ(DMA_UART_TX_IRQn);                       // UART Tx DMA interrupt sources -
M360
}



void UARTDMAWRITE(unsigned char *pucTX_DMA, unsigned int iNumRX)
{
   DmaDesc Desc;
```

```
    // Common configuration of all the descriptors used here
    Desc.ctrlCfg.Bits.cycle_ctrl       = CYCLECTRL_BASIC;   //CYCLECTRL_BASIC;
    Desc.ctrlCfg.Bits.next_useburst    = 0x1;
    Desc.ctrlCfg.Bits.r_power          = 0;
    Desc.ctrlCfg.Bits.src_prot_ctrl    = 0x0;
    Desc.ctrlCfg.Bits.dst_prot_ctrl    = 0x0;
    Desc.ctrlCfg.Bits.src_size         = SRCSIZE_BYTE;
    Desc.ctrlCfg.Bits.dst_size         = DSTSIZE_BYTE;

                                                        //  TX Primary   Descriptor
                        Desc.srcEndPtr                      = (unsigned int)(pucTX_DMA + iNumRX -
0x1);
    Desc.destEndPtr                    =  (unsigned int)&COMTX;
    Desc.ctrlCfg.Bits.n_minus_1        = iNumRX - 0x1;
    Desc.ctrlCfg.Bits.src_inc          = INC_BYTE;
    Desc.ctrlCfg.Bits.dst_inc          = INC_NO;
    *Dma_GetDescriptor(DMA_REQ_UART_TX,FALSE) = Desc;

}

                                                        // UART DMA Tx IRQ handler
void DMA_UART_TX_Int_Handler()
{
                        NVIC_DisableIRQ(DMA_UART_TX_IRQn);
                                                        // Clear Interrupt source
}
```

UART存储器映射寄存器

UART接口MMR的基地址为0x40005000。

表156. UART接口存储器地址表(基地址：0x40005000)

偏移值	名称	描述	访问类型	默认值
0x0000	COMTX	发送保持寄存器	W	0x0000
0x0000	COMRX	接收缓冲寄存器	R	0x0000
0x0004	COMIEN	中断使能寄存器	RW	0x0000
0x0008	COMIIR	中断识别寄存器	R	0x0000
0x000C	COMLCR	线路控制寄存器	RW	0x0000
0x0010	COMMCR	调制解调器控制寄存器	RW	0x0000
0x0014	COMLSR	线路状态寄存器	R	0x0000
0x0018	COMMSR	调制解调器状态寄存器	R	0x0000
0x0024	COMFBR	小数波特率寄存器	RW	0x0000
0x0028	COMDIV	波特率分频器寄存器	RW	0x0000
0x0030	COMCON	UART控制寄存器	RW	0x0000

UART发送和接收寄存器

地址：0x40005000；复位：0x0000；名称：COMRX/COMTX

COMRX和COMTX共用地址，但实现为不同的寄存器。如果写入这些寄存器，则用户访问发送保持寄存器(COMTX)。如果读取这些寄存器，则用户访问接收缓冲寄存器(COMRX)。

COMRX

这是一个8位寄存器，用户可从中读取收到的数据。如果COMIEN寄存器中的ERBFI位置1，当此寄存器通过串行输入端口满载接收数据时，会产生中断。

注意：如果用户代码在COMRX已满时将ERBFI置1，将立即产生中断。

COMTX

这是一个8位寄存器，用户可向其中写入要发送的数据。如果COMIEN寄存器中的ETBEI位置1，当COMTX为空时，会产生中断。

注意：如果用户代码在COMTX已然变空时将ETBEI置1，将立即产生中断。

表157. COMRX/COMTX寄存器位功能描述

位	名称	描述
7:0	VALUE	接收缓冲寄存器/发送保持寄存器。

UART中断使能寄存器

地址：0x40005004；复位：0x0000；名称：COMIEN

COMIEN是中断使能寄存器，用于配置哪个中断源产生中断。在此寄存器中，仅低4位用于使能中断。位4和位5用于使能UART DMA信号。 UART DMA通道和中断必须在DMA模块中配置。

表158. COMIEN寄存器位功能描述

位	名称	描述
7:6	保留	保留。
5	EDMAR	发送模式下的DMA请求。 0: 禁用。 1: 使能。
4	EDMAT	接收模式下的DMA请求。 0: 禁用。 1: 使能。
3	EDSSI	调制解调器状态中断。(当COMMSR[3:0]中的任意位置1时，产生此中断。) 0: 禁用。 1: 使能。
2	ELSI	Rx状态中断。(当COMLSR[4:1]中的任意位置1时，产生此中断。) 0: 禁用。 1: 使能。
1	ETBEI	发送缓冲器变空中断。 0: 禁用。 1: 使能。
0	ERBFI	接收缓冲器变满中断。 0: 禁用。 1: 使能。

UART中断识别寄存器

地址：0x40005008；复位：0x0000；名称：COMIIR

表159. COMIIR寄存器位功能描述

位	名称	描述
7:3	保留	保留。
2:1	STA	状态位。当NIRQ为低电平时，状态位用于给中断状态编码。详情参见表160。
0	NINT	中断标志。 0: 表示下列任意状况：发生接收缓冲器已满中断、发送缓冲器变空中断、线路状态或调制解调器状态中断。 1: 无中断(默认)。

表160. 中断识别表

位[2:1]，STA	位0，NIRQ	优先级	定义	清除操作
00	1	不适用	无中断	不适用
11	0	1	接收线路状态中断	读取COMLSR寄存器
10	0	2	接收缓冲器已满中断	读取COMRX寄存器
01	0	3	发送缓冲器变空中断	将数据写入COMTX或读取COMIIR
00	0	4	调制解调器状态中断	读取COMMSR寄存器

UART线路控制寄存器

地址：0x4000500C；复位：0x0000；名称：COMLCR

表161. COMLCR寄存器位功能描述

位	名称	描述
7	保留	保留。
6	BRK	设置断开。 1: 强制TxD为0。 0: 以正常模式工作。
5	SP	强制奇偶校验。 0: 奇偶校验不是基于EPS和PEN值的强制值。 1: 强制奇偶校验为如下基于EPS和PEN值的确定值：
4	EPS	偶校验选择位。 0: 奇校验。 1: 偶校验。
3	PEN	奇偶校验使能位。 0: 不发送或检查奇偶校验。 1: 发送并检查奇偶校验位。
2	STOP	停止位。 0: 在发送的数据中产生一个停止位。 1: 字长为5位时发送1.5个停止位；字长为6、7或8位时发送2个停止位。 不论所选停止位的个数是多少，接收器只检查第一个停止位。
1:0	WLS	字长选择位。 00: 5位。 01: 6位。 10: 7位。 11: 8位。

EPS	PEN	SP = 1时的功能
1	1	奇偶校验强制为1。
0	1	奇偶校验强制为0。
X	0	不发送奇偶校验。

UART调制解调器控制寄存器

地址：0x40005010；复位：0x0000；名称：COMMCR

表162. COMMCR寄存器位功能描述

位	名称	描述
7:5	保留	保留。
4	LOOPBACK	回送。 0: 正常模式。 1: 使能回送模式。 在回送模式下，强制UART TXD为高电平。调制解调器信号直接连接到状态输入(即RTS连接到CTS)。
3	OUT1	保留。
2	OUT2	保留。
1	RTS	请求发送。 0: 强制RTS输出为1。 1: 强制RTS输出为0。
0	DTR	数据终端就绪位。 0: 强制DTR输出为1。 1: 强制DTR输出为0。

UART线路状态寄存器

地址：0x40005014；复位：0x0000；名称：COMLSR

表163. COMLSR寄存器位功能描述

位	名称	描述
7	保留	保留。
6	TEMT	COMTX和移位寄存器变空状态位。 1: 若COMTX和移位寄存器变空，此位指示数据已被发送，即移位寄存器不再有数据(默认)。 写入COMTX时清0。
5	THRE	COMTX变空状态位。 1: 若COMTX变空，只要此位置1，便可向COMTX写入数据；但前面数据可能还未发送，且可能仍保存在移位寄存器内(默认)。 写入COMTX时清0。
4	BI	断开指示符。 1: 当UART RXD保持低电平超过最大字长时。 自动清0。
3	FE	帧错误。 1: 当停止位无效时。 自动清0。
2	PE	奇偶校验错误。 1: 当发生奇偶校验错误时。 自动清0。
1	OE	上溢错误。 1: 若当前数据在读取前被覆盖，该位自动置1。 自动清0。
0	DR	数据就绪。 1: 当COMRX已满时，该位自动置1。 读取COMRX后清0。

UART调制解调器状态寄存器

地址：0x40005018；复位：0x0000；名称：COMMSR

表164. COMMSR寄存器位功能描述

位	名称	描述
7	DCD	数据载波检测。 若DCD当前为逻辑低电平则置1。 若DCD当前为逻辑高电平则清0。
6	RI	响铃指示。 若RI当前为逻辑低电平则置1。 若RI当前为逻辑高电平则清0。
5	DSR	数据准备就绪。 若DSR当前为逻辑低电平则置1。 若DSR当前为逻辑高电平则清0。
4	CTS	清除发送。 若CTS当前为逻辑低电平则置1。 若CTS当前为逻辑高电平则清0。
3	DDCD	ΔDCD。 上一次读取COMMSR后，如果DCD状态改变，则此位置1。 上一次读取COMMSR后，如果DCD未改变状态，则此位清0。读取COMMSR后清0。
2	TERI	下降沿RI。 上一次读COMMSR后，如果NRI由0变为1，则此位置1。 读取COMMSR后清0。
1	DDSR	ΔDSR。 上一次读取COMMSR后，如果DSR状态改变，则此位置1。 读取COMMSR后清0。
0	DCTS	ΔCTS。 上一次读取COMMSR后，如果CTS状态改变，则此位置1。 读取COMMSR后清0。

UART小数波特率分频器寄存器

地址： 0x40005024；复位： 0x0000；名称： COMFBR

表165. COMFBR寄存器位功能描述

位	名称	描述
15	ENABLE	小数波特率发生器使能位，可产生更准确的波特率。小数波特率的产生可通过如下公式来描述，其中*UCLK/DIV*为16 MHz分频时钟，分频配置由CLKCON1和CLKSYSDIV决定。CLKDIS[3]应置0以使能该时钟。UART操作的最终波特率如图29所示。 $$Baud\ Rate = \frac{UCLK / DIV}{2 \times (M + N / 2048)}{16 \times COMDIV}$$ 1: 使能。 0: 禁用。
14:13	保留	保留。
12:11	DIVM	小数波特率M分频位(1至3)。此位不应为0。
10:0	DIVN	小数波特率N分频位(0至2047)。

表166. 波特率示例—所有例子的UCLK/DIV = 16 MHz

波特率	COMDIV	DIVM	DIVN	实际	百分比误差
9600	17	3	131	9599.25	−0.0078%
19,200	8	3	523	19,199.04	−0.0050%
38,400	4	3	523	38,398.08	−0.0050%
57,600	8	1	174	57,605.76	+0.0100%
115,200	2	2	348	115,211.5	+0.0100%
230,400	2	1	174	230,423	+0.0100%
460,800	1	1	174	460,846.1	+0.0100%

UART分频器寄存器

地址：0x40005028；复位：0x0000；名称：COMDIV

波特率分频器寄存器是16位寄存器，用于产生UART数据传输的波特率。无小数分频的波特率是主时钟的分频版本，如图29所示。

$$波特率 = UCLK/DIV \div (2 \times 16 \times COMDIV) \div (M + N \div 2048)$$

其中，*UCLK/DIV*为分频16 MHz时钟，通过CLKCON1[11:9]和CLKDIS[3]配置。

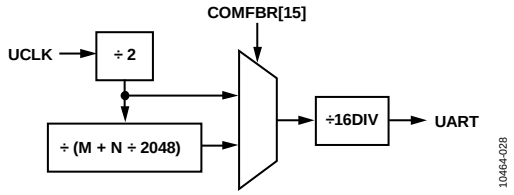


图29. 波特率产生

表167. COMDIV寄存器位功能描述

位	名称	描述
15:0	VALUE	设置波特率。COMDIV寄存器不应为0。

UART控制寄存器

地址：0x40005030；复位：0x0000；名称：COMCON

表168. COMCON寄存器位功能描述

位	名称	描述
8:1	保留	保留
0	DISABLE	UART禁用位。建议在进入关断模式之前禁用UART。 0: 使能UART。 1: 禁用UART。这会复位状态机和内部计数器，但MMR内容保持不变。

通用定时器

通用定时器特性

- 两个相同的16位递增/递减通用定时器
  - 定时器0和定时器1
- 由4个不同时钟源提供时钟(时钟源可通过1、16、256或32,768倍预分频器进行向下分频)。
  - 16 MHz系统时钟或8 MHz系统时钟(条件是CLKSYSDIV = 1 (UCLK))
  - 外设时钟(PCLK)
  - 32 kHz内部振荡器(LFOSC)
  - 32 kHz外部晶振(LFXTAL)
- 两种模式
  - 自由振荡
  - 周期
- 捕捉事件特性
  - 每个定时器能够捕捉15个不同的事件

通用定时器功能框图

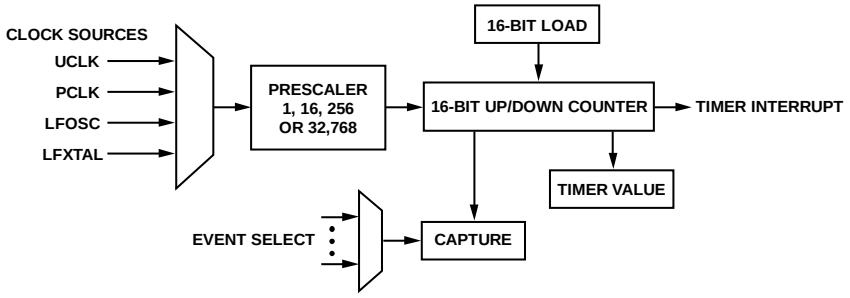


图30. 通用定时器功能框图

通用定时器概述

定时器0和定时器1是两个相同的16位递增/递减通用定时器。其时钟可以来自4个不同的时钟源：UCLK、PCLK、32 kHz内部振荡器(LFOSC)和32 kHz外部晶振(LFXTAL)。这些时钟源可通过1、16、256或32,768倍预分频器进行向下分频。

定时器可以是自由振荡或周期式。

- 在自由振荡模式下，计数器从最大值递减到0或从0递增到最大值，然后重新开始。
- 在周期模式下，计数器以加载寄存器(TxLD MMR)中的值为起始值，递减/递增计数至0或最大值，然后再以加载寄存器中的值为起始值，重新开始计数。

通过访问计数器的值寄存器(TxVAL)，可以随时读出计数器的值。

此寄存器与UCLK同步。因此，当一个定时器的时钟源不是UCLK时，TxVAL反映最新定时器值。

TxCON寄存器选择定时器模式，配置时钟源，选择递增/递减，启动计数器，并控制事件捕捉功能。

每当计数器的值达到0(递减计数)或最大值(递增计数)时，都会产生一个中断信号。向某一定时器(TxCLRI)的时间清除寄存器写入1，可以清除IRQ。

此外，定时器0和定时器1均有捕捉寄存器(TxCAP)，其可以被选定的IRQ中断源初始置位触发。中断触发时，定时器的当前值被复制到TxCAP内，定时器继续运行。此功能可用来更精确地判断事件置位。每个定时器可捕捉16个不同事件，如表169所示。

通用定时器工作原理

自由振荡模式

在自由振荡模式下，将使能位(TxCON[4])置1且将MOD位(TxCON[3])置0可启动定时器。若是递增/递减计数，定时器将从0/最大值递增/递减至最大值/0。最大值为$2^{16} - 1$或0xFFFF(二进制格式)。达到最大值(或0)时会发生超时中断，TxSTA[0]置1。要清除中断，用户代码必须向TxCLRI[0]写入1。如果TxCON[7]置1，定时器会继续计数，并在写入TxCLRI寄存器时重载。

周期模式

在周期模式下，首先应载入初始TxLD值，然后将使能位(TxCON[4])置1以启动定时器。定时器从TxLD中的值递增至最大值或递减至0，取决于TxCON[2]设置(递增/递减)。达到最大值或0时，定时器产生中断。TxLD重新载入TxVAL，定时器继续递增或递减。更改TxCON或TxLD寄存器之前，应当禁用定时器。如果在加载定时器时更改TxLD寄存器，可能会产生不明结果。默认情况下，计数器会在产生中断信号时自动重载。如果TxCON[7]置1，定时器也会在用户代码写入TxCLRI时重载。这样，用户对TxLD的更改可以立即生效，而无需等到下一次超时。

定时器间隔可通过下式计算：

如果将定时器设置为递减计数，那么

间隔 = (TxLD × 预分频器)/源时钟

例如，若TxLD = 0x100、预分频器 = 4且时钟源 = UCLK，则间隔为64 μs(其中UCLK = 16 MHz)。

如果将定时器设置为递增计数，那么

间隔 = ((最大值 − TxLD) × 预分频器)/源时钟

异步时钟源

将某一定时器的控制寄存器的使能位(TxCON[4])置1，可以启动相应的定时器。

然而，如果定时器时钟源是LFXTAL或LFOSC，必须注意以下几点：

- 如果TxSTA[6]为1，则不应写入控制寄存器(TxCON)。因此，配置控制寄存器(TxCON)之前应读取TxSTA。当TxSTA[6]为0时，便可更改控制寄存器。这样做的目的是确保处理器与定时器时钟域之间的定时器控制同步完毕。
- 清除TxCLRI中的中断后，必须确保寄存器写操作已全部完成，然后才能从中断处理程序返回。需要时可使用数据同步屏障(DSB)指令，并确认TxSTA[7] = 0。

```
__asm void asmDSB()
{
nop
DSB
BX LR
}
```

- 通过访问计数器的值寄存器(TxVAL)，可以随时读出计数器的值。在异步配置中，TxVAL始终应该读两次。如果两次读到的结果不同，应该再读一次以得到正确的值。

在任何时候，TxVAL都包含一个待读取的有效值，且与PCLK同步。

使能位置1或清0后，应先读取TxSTA，再写入任意定时器寄存器。当TxSTA[7]为0时，便可更改这些寄存器。这样做的目的是确保定时器已完成处理器与定时器时钟域之间的同步。典型同步时间为2个定时器时钟周期。

在任何时候，TxVAL都包含一个待读取的有效值，且与PCLK时钟同步。TxCON寄存器用于使能定时器，选择模式，选择预分频器值，以及控制事件捕捉功能。

注意：要使能定时器1的系统时钟，CLKDIS[6]必须清0。此位必须清0才能使能定时器1。同样，要使能定时器0的系统时钟，CLKDIS[5]必须清0。此位必须清0才能使能定时器0。

捕捉事件功能

通用定时器可以捕捉30个中断事件。这些事件按定时器分为两组，每组16个输入，如表169所示。与通用定时器相关的任何事件都能导致16位TxVAL寄存器的内容被捕捉到16位TxCAP寄存器中。TxCON有一个4位字段用于选择16个事件中要捕捉的事件。

发生选定的中断事件时，TxVAL寄存器的内容便会被复制到TxCAP寄存器中。TxSTA[1]置1表示有捕捉事件待处理。向TxCLRI[1]写入1可将此位清0。TxCAP寄存器也会保持其值，直到向TxCLRI[1]写入1才能覆盖该值。

表169. 定时器捕捉事件

事件选择位(TxCON[11:8])	定时器0捕捉源	定时器1捕捉源
0	唤醒定时器	UART
1	外部中断0	定时器0
2	外部中断1	SPI0
3	外部中断2	SPI1
4	外部中断3	I²C从机
5	外部中断4	I²C主机
6	外部中断5	DMA错误
7	外部中断6	DMA完成，任意DMA通道
8	外部中断7	外部中断1
9	看门狗定时器	外部中断2
10	定时器1	外部中断3
11	ADC0 (ADuCM360)/保留(ADuCM361)	PWMTRIP
12	ADC1	PWM0
13	步进检测	PWM1
14	DMA完成，任意DMA通道	PWM2
15	闪存控制器	保留。

通用定时器存储器映射寄存器

通用定时器0寄存器映射

表170. 通用定时器0存储器映射寄存器地址表(定时器0基地址：0x40000000)

偏移	名称	描述	访问类型	默认值
0x0000	T0LD	16位加载值	RW	0x0000
0x0004	T0VAL	16位定时器值，只读	R	0x0000
0x0008	T0CON	控制寄存器	RW	0x000A
0x000C	T0CLRI	清除中断寄存器	RW	0x0000
0x0010	T0CAP	捕捉寄存器	R	0x0000
0x001C	T0STA	状态寄存器	R	0x0000

通用定时器1寄存器映射

表171. 通用定时器1存储器映射寄存器地址表(定时器1基地址：0x40000400)

偏移	名称	描述	访问类型	默认值
0x0000	T1LD	16位加载值	RW	0x0000
0x0004	T1VAL	16位定时器值，只读	R	0x0000
0x0008	T1CON	控制寄存器	RW	0x000A
0x000C	T1CLRI	清除中断寄存器	RW	0x0000
0x0010	T1CAP	捕捉寄存器	R	0x0000
0x001C	T1STA	状态寄存器	R	0x0000

通用定时器加载寄存器

地址：0x40000000；复位：0x0000；名称：T0LD

地址：0x40000400；复位：0x0000；名称：T1LD

表172. T0LD和T1LD寄存器的位功能描述

位	名称	描述
15:0	VALUE	加载值，默认为0。

通用定时器值寄存器

地址：0x40000004；复位：0x0000；名称：T0VAL

地址：0x40000404；复位：0x0000；名称：T1VAL

表173. T0VAL和T1VAL寄存器的位功能描述

位	名称	描述
15:0	VALUE	当前计数器值，只读。

通用定时器控制寄存器

地址：0x40000008；复位：0x000A；名称：T0CON

地址：0x40000408；复位：0x000A；名称：T1CON

如果TxSTA[6]或TxSTA[7]为1，则不应写入对应的TxCON寄存器。注意：定时器时钟必须比FCLK(系统时钟)慢4倍，因此，对于不同的CD值，选择预分频器时应小心。

表174. T0CON和T1CON寄存器的位功能描述

位	名称	描述
15:13	保留	保留。这些位应该写入0。
12	EVENTEN	事件选择位。 0: 禁用事件的时间捕捉。 1: 使能事件的时间捕捉。
11:8	EVENT	事件选择范围(0至15)。事件见表169。
7	RLD	周期模式的重载控制位。 0: 仅在超时时重载。 1: 写入TxCLRI时更改加载值。
6:5	CLK	时钟选择。 00: UCLK。 01: PCLK。 10: LFOSC(32 kHz内部振荡器)。 11: LFXTAL(32 kHz外部晶振)。
4	ENABLE	定时器使能位。定时器从初始值(递增模式为0，递减模式为0xFFFF)开始计数。 此位清0将复位定时器，包括TxVAL寄存器。 0: 禁用定时器。 1: 使能定时器。
3	MOD	定时器模式。 0: 工作在自由振荡模式。 1: 工作在周期模式。
2	UP	递增计数。 0: 定时器递减计数。 1: 定时器递增计数。
1:0	PRE	预分频器。若所选时钟源为UCLK，则选定的预分频器还会进行4分频。 00: DIV1: 源时钟/1。 01: DIV16: 源时钟/16。 10: DIV256: 源时钟/256。 11: DIV32768: 源时钟/32,768。

通用定时器控制寄存器

地址：0x4000000C；复位：0x0000；名称：T0CLRI

地址：0x4000040C；复位：0x0000；名称：T1CLRI

表175. T0CLRI和T1CLRI寄存器的位功能描述

位	名称	描述
15:2	保留	保留。这些位应该写入0。
1	CAP	清除已捕捉事件中断。 1: 清除已捕捉事件中断。 此位读出值始终为0。
0	TMOUT	清除超时中断。 1: 清除超时中断。 此位读出值始终为0。

从中断处理程序回到正常工作之前，确保寄存器写操作已全部完成。需要时可使用数据同步屏障(DSB)指令。下面的代码示例说明了如何在一个C程序中实现DSB ARM Cortex-M3指令。

```
__asm void asmDSB()
{
       nop
       DSB
BX LR
}
void GP_Tmr0_Int_Handler()
{
  T0CLRI = 0x1;
  asmDSB();
}
```

通用定时器捕捉寄存器

地址：0x40000010；复位：0x0000；名称：T0CAP

地址：0x40000410；复位：0x0000；名称：T1CAP

表176. T0CAP和T1CAP寄存器的位功能描述

位	名称	描述
15:0	VALUE	16位捕捉值。只读。TxCAP会保持其值到用户代码将TxCLRI[1]置1为止。 如果相同事件再次发生，TxCAP不会被覆盖。

通用定时器状态寄存器

地址：0x4000001C；复位：0x0000；名称：T0STA

地址：0x4000041C；复位：0x0000；名称：T1STA

表177. T0STA和T1STA寄存器的位功能描述

位	名称	描述
15:8	保留	保留。这些位应该写入0。
7	PDOK	T0CLRI/T1CLRI同步。 当T0CLRI/T1CLRI值正在定时器时钟域中进行更新时，此位自动置1，表示定时器配置尚未生效。 在定时器时钟域中清除中断时清0。
6	BUSY	定时器繁忙。 置1表示定时器尚未准备好接收写入TxCON的命令。 TxCON值的上次改变尚未在定时器时钟域中同步。 清0表示定时器已准备好接收写入TxCON的命令。 TxCON值的上次改变已在定时器时钟域中同步。
5:2	保留	保留。这些位应该写入0。
1	CAP	捕捉事件待处理。 置1表示有捕捉事件待处理。 向TxCLRI[1]写入1可将此位清0。
0	TMOUT	发生超时事件。 置1表示发生了超时事件。对于递增计数模式，这说明计数器已达到最大值。 对于递减计数模式，这说明计数器已达到0。 向TxCLRI[0]写入1可将此位清0。

唤醒定时器

唤醒定时器特性

- 32位计数器(向下计数模式或向上计数模式)
- 四个可编程预分频器(1、16、256或32,768)的时钟源
  - 外设时钟(PCLK)
  - 32 kHz外部晶振(LFXTAL)
  - 32 kHz内部振荡器(LFOSC)
  - 施加于P1.0的外部时钟(EXTCLK)
- 四个比较点，一个原子式递增

通用定时器功能框图

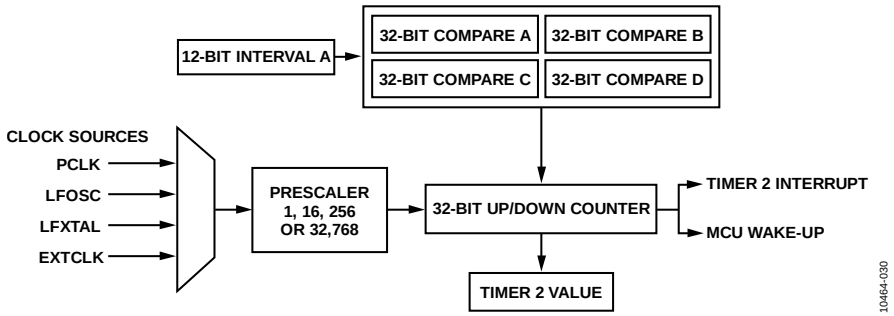


图31. 唤醒定时器功能框图

唤醒定时器概述

唤醒定时器模块(定时器2)由一个32位计数器组成，其时钟来源有4个：系统时钟(PCLK)、外部晶振(LFXTAL)、内部振荡器(LFOSC)或施加于P1.0的外部时钟(EXTCLK)。所选的时钟源可通过预分频器进行1、16、256或32,768分频。当PCLK时钟被禁用时，唤醒定时器会继续工作，独立于所用的时钟源。

该定时器可以在自由振荡或周期模式下工作。在自由振荡模式下，定时器从0x00000000计数到0xFFFFFFFF，然后再次从0x00000000开始。在周期模式下，定时器从0x00000000计数到T2WUFD(T2WUFD0和T2WUFD1)。

此外，唤醒定时器有4个特定时间字段用来与唤醒计数器相比较：T2WUFA、T2WUFB、T2WUFC和T2WUFD。所有四个唤醒比较点都能产生中断或唤醒信号。在自由振荡模式时，必须在软件中重新配置T2WUFA、T2WUFB、T2WUFC和T2WUFD以产生周期中断。

唤醒定时器工作原理

设置该定时器之前，必须配置唤醒定时器比较器寄存器。写入控制使能位(T2CON[7])可启动定时器。定时器递增，直至其值达到最大值(自由振荡模式下)或T2WUFD与唤醒值T2VAL匹配。

唤醒定时器是一个32位定时器。其当前值存储在两个16位寄存器中：T2VAL1存储高16位，T2VAL0存储低16位。

读取T2VAL0时，T2VAL1被冻结在当前值，直至随后被读取。控制位FREEZE (T2CON[3])必须置1，以便在低位读取和高位读取之间冻结T2VAL寄存器。

时钟选择

时钟通过设置T2CON[10:9]来选择。

如果选择LFXTAL (T2CON[10:9] = 01)，必须确保该时钟电路开启(XOSCCON[0] = 1)。

如果选择PCLK (T2CON[10:9] = 00)，配置T2CON[1:0] = 00将导致预分频器系数为4。

与LFOSC和LFXTAL时钟域的同步由硬件自动完成，定时器0、定时器1和定时器3中所述的关于异步时钟的注意事项不适用于唤醒定时器。

比较字段寄存器

硬件更新字段

T2INC是一个12位间隔寄存器，用于通过硬件更新T2WUFAx中的比较值。将新值写入T2INC时，内部32位比较寄存器(T2WUFAx)的位[16:5]便会载入新的T2INC值。如果新比较值小于T2WUFD值(周期模式)或0xFFFFFFFF(自由振荡模式)，则每次唤醒计数器达到此比较寄存器中的值时，此32位比较寄存器就会自动用T2INC中的内容递增(移动5位)。如果新比较值大于上述限值，它就会按如下公式重新计算：

在自由振荡模式下，新T2WUFA = 原T2WUFA + (32 × T2INC) − 0xFFFFFFFF。

在周期模式下，新T2WUFA = 原T2WUFA + (32 × T2INC) − T2WUFD。

最大可编程间隔为4秒多一点点。

T2INC与定时器值的位[16:5]进行比较。由于它左移5位，因此其值必须乘以32才能获得比较值。

对于默认值0xC8(为了计算目的，0xC8等于十进制200)，预分频系数 = 1，且选择32 kHz时钟，

间隔 = ((200 × 32) + 1) × 1/32,768 = 195.3155 ms

要更改间隔值，必须停止定时器，以便间隔寄存器能够加载比较寄存器(若T2CON[11] = 0)。

要更改间隔值，请在定时器运行时设置STOPINC (T2CON[11] = 1)。

下一个唤醒字段A中断发生之后，新的T2INC值生效。如果用户在定时器使能的情况下写入此寄存器，应先将STOPINC位置1，再写入此寄存器，在更新之后将STOPINC清0。

软件更新字段

T2WUFB、T2WUFC和T2WUFD是32位值，由用户通过T2WUFx0和T2WUFx1寄存器编程(x = B、C或D)。当唤醒定时器配置为周期模式时，T2WUFD包含加载值。

T2WUFBx和T2WUFCx寄存器可以随时写入，但对应的中断使能位(T2IEN[1]或T2IEN[2])必须禁用。更新寄存器之后，可以重新使能中断。

在周期模式下，只有禁用定时器时才能写入T2WUFDx寄存器。在自由振荡模式下，可以在定时器运行时写入T2WUFDx寄存器。这样做之前，必须禁用对应的中断使能位(T2IEN[3])。更新寄存器之后，可以重新使能中断。

在自由振荡模式下，T2WUFB、T2WUFC和T2WUFD可以随时写入，但T2IEN寄存器中的对应中断使能位必须禁用。更新寄存器之后，可以重新使能中断。在周期模式下，这仅适用于T2WUFB和T2WUFC。

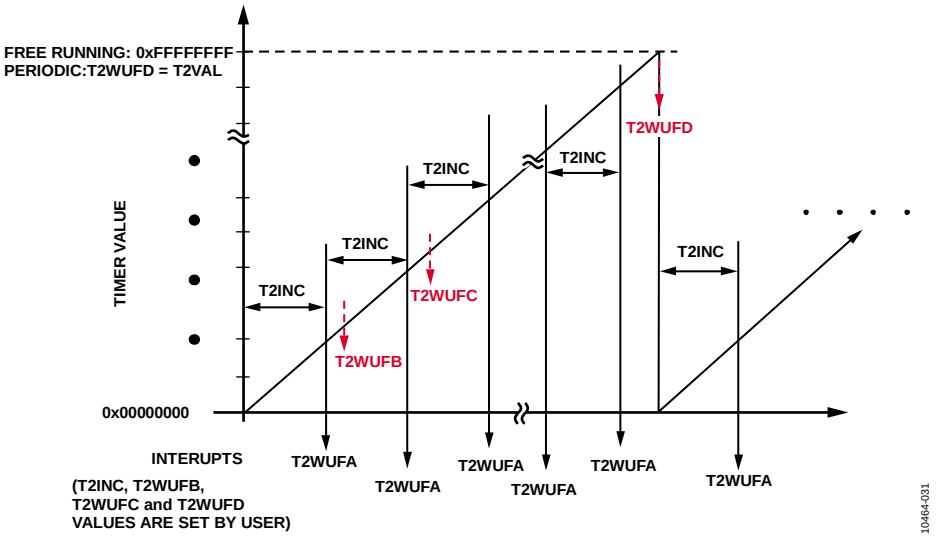


图32. 唤醒定时器字段操作

中断/唤醒信号

当计数器值与任一比较点或自由振荡模式下的最大值一致时，就会产生中断。定时器继续计数或复位至0。

唤醒定时器可产生5个可屏蔽的中断。这些中断通过T2IEN寄存器使能。将T2CLRI寄存器中的特定位置1，可以清除相应的中断。

注意：当使用32 kHz时钟时，为使清除中断生效，需要2个32 kHz时钟周期。

从中断处理程序回到正常工作之前，确保寄存器写操作已全部完成。需要时可使用数据同步屏障(DSB)指令。

下面的代码示例说明了如何在一个C程序中实现DSB ARM Cortex-M3指令。

```
void WakeUp_Int_Handler()
{
WutClrInt(iSource);
__DSB();
}
```

在此期间，不应将器件置于任何省电模式。IRQCRY (T2STA[6])指示是否能将器件置于省电模式。

将T2CON寄存器中的定时器使能位(T2CON[7])清0，定时器就会停止并复位。

唤醒定时器存储器映射寄存器

表178. 唤醒定时器存储器映射寄存器地址表(基地址：0x40002500)

偏移	名称	描述	访问类型	默认值
0x0000	T2VAL0	当前计数值LSB	R	0x0000
0x0004	T2VAL1	当前计数值MSB	R	0x0000
0x0008	T2CON	控制寄存器	RW	0x0040
0x000C	T2INC	唤醒字段A的12位间隔寄存器	RW	0x00C8
0x0010	T2WUFB0	唤醒字段B LSB	RW	0x1FFF
0x0014	T2WUFB1	唤醒字段B MSB	RW	0x0000
0x0018	T2WUFC0	唤醒字段C LSB	RW	0x2FFF
0x001C	T2WUFC1	唤醒字段C MSB	RW	0x0000
0x0020	T2WUFD0	唤醒字段D LSB	RW	0x3FFF
0x0024	T2WUFD1	唤醒字段D MSB	RW	0x0000
0x0028	T2IEN	中断使能	RW	0x0000
0x002C	T2STA	状态	R	0x0000
0x0030	T2CLRI	清除中断	W	不适用
0x003C	T2WUFA0	唤醒字段A LSB	RW	0x1900
0x0040	T2WUFA1	唤醒字段A MSB	RW	0x0000

唤醒定时器计数值寄存器

地址：0x40002500；复位：0x0000；名称：T2VAL0

地址：0x40002504；复位：0x0000；名称：T2VAL1

表179. T2VAL0寄存器位功能描述(T2VAL0地址：0x40002500)

位	名称	描述
15:0	VALUE	唤醒定时器当前值的低16位

表180. T2VAL1寄存器位功能描述(T2VAL1地址：0x40002504)

位	名称	描述
15:0	VALUE	唤醒定时器当前值的高16位

唤醒定时器控制寄存器

地址：0x40002508；复位：0x0040；名称：T2CON

表181. T2CON寄存器位功能描述

位	名称	描述
15:12	保留	未用位的位置。
11	STOPINC	用户利用此位可安全地更新间隔寄存器。 0: 允许更新唤醒字段A。 1: 禁止唤醒字段A用间隔寄存器值更新。
10:9	CLK	时钟选择：用于选择定时器的时钟源。 00: PCLK(默认)。 01: LFXTAL: 32 kHz外部晶振。 10: LFOSC: 32 kHz内部振荡器。 11: EXTCLK: 来自P1.0的外部时钟。
8	WUEN	时间字段值的唤醒使能位。该位必须置1。 0: 禁用异步唤醒定时器。中断条件不会将器件从睡眠模式唤醒。 1: 使能异步唤醒定时器，即使处理器时钟已关闭。当时间值之一等于T2WUFA时， 产生唤醒输出信号。
7	ENABLE	定时器使能位。更改任何控制信息或定时器字段值时，此位必须为0(低电平)。 0: 禁用定时器(默认)。 1: 使能定时器。
6	MOD	定时器自由振荡使能。 0: PERIODIC: 以周期模式工作，即递增计数至T2WUFD中的值。 1: FREERUN: 以自由振荡模式工作(默认)，即从0计数到FFFF FFFF，然后再次从0开始。
5:4	保留	保留。
3	FREEZE	冻结使能位。 0: 禁用冻结使能位(默认)。 1: 从T2VAL0读取低16位后，冻结高16位。确保软件读取定时器的原子式快照。 读取高位(T2VAL1)后，整个T2VAL寄存器解冻
2	保留	保留。
1:0	PRE	时钟预分频器选择。如果时钟预分频器选择 = 00 (T2CON[1:0] = 00)且所选时钟源为 PCLK (T2CON[10:9] = 00)，则预分频器系数为4。 00: DIV1: 源时钟/1(默认)。 01: DIV16: 源时钟/16。 10: DIV256: 源时钟/256。 11: DIV32768: 源时钟/32,768。

注意：定时器时钟必须比PCLK(系统时钟)慢4倍，因此，对于不同的CD值，选择预分频器时应小心。

唤醒定时器间隔寄存器：间隔寄存器

地址：0x4000250C；复位：0x00C8；名称：T2INC

表182. T2INC寄存器位功能描述

位	名称	描述
15:12	保留	保留
11:0	VALUE	唤醒间隔

唤醒定时器唤醒字段寄存器B

地址： 0x40002510；复位： 0x1FFF；名称： T2WUFB0

地址： 0x40002514；复位： 0x0000；名称： T2WUFB1

T2WUFBx寄存器可以随时写入，但对应的中断使能位(T2IEN[1])必须禁用。 更新寄存器之后，可以重新使能中断。

表183. T2WUFB0寄存器位功能描述

位	名称	描述
15:0	VALUE	唤醒字段B的低16位

表184. T2WUFB1寄存器位功能描述

位	名称	描述
15:0	VALUE	唤醒字段B的高16位

唤醒定时器唤醒字段寄存器C

地址：0x40002518；复位：0x2FFF；名称：T2WUFC0

地址：0x4000251C；复位：0x0000；名称：T2WUFC1

T2WUFCx寄存器可以随时写入，但对应的中断使能位(T2IEN[2])必须禁用。更新寄存器之后，可以重新使能中断。

表185. T2WUFC0寄存器位功能描述

位	名称	描述
15:0	VALUE	唤醒字段C的低16位

Table 186. T2WUFC1 Register Bit Descriptions

位	名称	描述
15:0	VALUE	唤醒字段C的高16位

唤醒定时器唤醒字段寄存器D

地址：0x40002520；复位：0x3FFF；名称：T2WUFD0

地址：0x40002524；复位：0x0000；名称：T2WUFD1

在周期模式下，只有禁用定时器时才能写入T2WUFDx寄存器。在自由振荡模式下，可以在定时器运行时写入T2WUFDx寄存器。这样做之前，必须禁用对应的中断使能位(T2IEN[3])。更新寄存器之后，可以重新使能中断。

表187. T2WUFD0寄存器位功能描述

位	名称	描述
15:0	VALUE	唤醒字段D的低16位

表188. T2WUFD1寄存器位功能描述

位	名称	描述
15:0	VALUE	唤醒字段D的高16位

唤醒定时器中断使能寄存器

地址：0x40002528；复位：0x0000；名称：T2IEN

表189. T2IEN寄存器位功能描述

位	名称	描述
15:5	保留	未用位的位置。
4	ROLL	计数器翻转时的中断使能位。仅发生在自由振荡模式下。 0: 禁用翻转中断(默认)。 1: 定时器2翻转时产生中断。
3	WUFD	T2WUFD中断使能。 0: 禁用T2WUFD中断(默认)。 1: T2VAL达到T2WUFD时产生中断。
2	WUFC	T2WUFC中断使能。 0: 禁用T2WUFC中断(默认)。 1: T2VAL达到T2WUFC时产生中断。
1	WUFB	T2WUFB中断使能。 0: 禁用T2WUFB中断(默认)。 1: T2VAL达到T2WUFB时产生中断。
0	WUFA	T2WUFA中断使能。 0: 禁用T2WUFA中断(默认)。 1: T2VAL达到T2WUFA时产生中断。

唤醒定时器状态寄存器

地址：0x4000252C；复位：0x0000；名称：T2STA

表190. T2STA寄存器位功能描述

位	名称	描述
15:9	保留	未用位的位置。
8	PDOK	指示使能位变更是否已同步到32 kHz时钟域。 当控制寄存器中的使能位(T2CON[7])置1或清0时，此位置1。 当使能位变更已同步到32 kHz时钟域时，此位清0。
7	FREEZE	T2VAL冻结状态。 读取T2VAL0时置1，表示T2VAL已被冻结。 读取T2VAL1时清0，表示T2VAL已解冻。
6	IRQCRY	关断控制的唤醒信号状态。 当外部晶振时钟域中的任何中断仍然有效时，此位置1。 当中断被清除时，此位自动清0。
5	保留	未用位的位置。
4	ROLL	计数器翻转时的中断使能位。仅发生在自由振荡模式下。 当中断使能寄存器已使能且T2VALS计数器寄存器等于全1时，此位置1。 清0。
3	WUFD	T2WUFD中断标志。 此位置1表示发生了比较器中断(若中断使能寄存器中的对应位已使能)。 写入T2CLRI中对应的位后清0。
2	WUFC	T2WUFC中断标志。 此位置1表示发生了比较器中断(若中断使能寄存器中的对应位已使能)。 写入T2CLRI中对应的位后清0。
1	WUFB	T2WUFB中断标志。 此位置1表示发生了比较器中断(若中断使能寄存器中的对应位已使能)。 写入T2CLRI中对应的位后清0。
0	WUFA	T2WUFA中断标志。 此位置1表示发生了比较器中断(若中断使能寄存器中的对应位已使能)。 写入T2CLRI中对应的位后清0。

唤醒定时器清除中断寄存器

地址：0x40002530；复位：不适用；名称：T2CLRI

表191. T2CLRI寄存器位功能描述

位	名称	描述
15:5	保留	未用位的位置。
4	ROLL	计数器翻转时的中断清除位。仅发生在自由振荡模式下。 0: 同步后自动清0。 1: 清除ROLL中断标志。
3	WUFD	T2WUFD中断标志。 0: 同步后自动清0。 1: 清除T2WUFD中断标志。
2	WUFC	T2WUFC中断标志。 0: 同步后自动清0。 1: 清除T2WUFC中断标志。
1	WUFB	T2WUFB中断标志。 0: 同步后自动清0。 1: 清除T2WUFB中断标志。
0	WUFA	T2WUFA中断标志。 0: 同步后自动清0。 1: 清除T2WUFA中断标志。

从中断处理程序回到正常工作之前，确保寄存器写操作已全部完成。需要时可使用数据同步屏障(DSB)指令。DSB示例代码参见“中断/唤醒信号”部分。

唤醒定时器比较寄存器A

地址：0x4000253C；复位：0x1900；名称：T2WUFA0

地址：0x40002540；复位：0x0000；名称：T2WUFA1

当该定时器使能时，可写入T2WUFAx寄存器。先必须写入T2WUFA0；写入T2WUFA1后，两个寄存器值均改变。如果所需的值很小，那么将0x0000写入T2WUFA1即足以让T2WUFA0更新。无论STOPINC (T2CON[11])位置1与否，都可以写入T2WUFAx。若置1，则STOPINC (T2CON[11])位禁止T2WUFAx由硬件递增。

T2WUFAx是32 kHz时钟域中的一个异步寄存器。因此，应两次读取它以确认读取的值正确。

表192. T2WUFA0寄存器位功能描述

位	名称	描述
15:0	VALUE	比较寄存器A的低16位

表193. T2WUFA1寄存器位功能描述

位	名称	描述
15:0	VALUE	比较寄存器A的高16位

看门狗定时器

看门狗定时器特性

- 16位递减计数定时器，可用于从无效软件状态恢复到正常状态。
- 由32 kHz内部振荡器(LFOSC)通过一个可编程预分频器(1、16、256或4096分频)提供时钟。

看门狗定时器功能框图

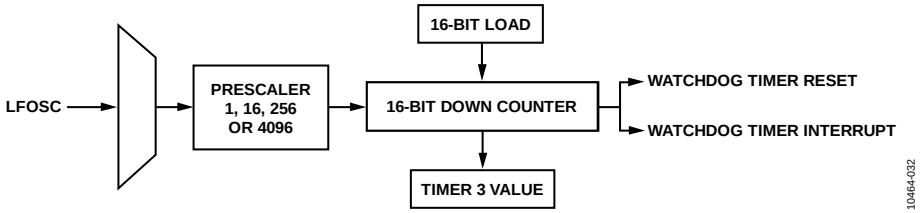


图33. 看门狗定时器功能框图

看门狗定时器概述

看门狗定时器(定时器3)用于从无效软件状态恢复到正常状态。一旦使能，它便需要周期服务来阻止其强制器件复位。调试时，可配置此定时器产生中断而不是复位。

看门狗定时器由内部32.768 kHz振荡器LFOSC提供时钟。除复位期间外，它会一直提供时钟。

看门狗定时器是16位递减定时器，带可编程预分频器。预分频器系数可选，可将LFOSC以1、16、256或4096的系数分频。

看门狗定时器工作原理

使能定时器

复位后，看门狗定时器默认使能。

调试时或不需要看门狗定时器时，用户代码应在代码开始处禁用看门狗定时器。

```
T3CON = 0x00;                                   // Disable watchdog timer
```

使能看门狗定时器(设置T3CON[5] = 1)也会为T3CON和T3LD提供写保护。这意味着在内核执行完成之后，用户代码可以禁用定时器，然后在T3CON[5] = 1下对其重新配置(仅一次)。这样，T3CON和T3LD就会受到写保护。T3STA[5]指示定时器配置是否已锁定。只有复位才能将T3CON[5]清0，解锁T3CON和T3LD，并允许重新配置定时器。

如果未更改T3CON，用户代码可随时更改T3LD。如果T3CON[5]清0，则禁用定时器。设置可以更改，定时器可以重新使能。

可编程超时

如果看门狗定时器设置为自由振荡(T3CON[6] = 0)，则看门狗定时器值从0x1000递减到0，再绕回到0x1000，然后继续递减。要实现大于或小于0x1000(使用默认预分频T3CON[3:2] = 10、预分频系数 = 256时约为32秒)的超时值，应使用周期模式(T3CON[6] = 1)，并且T3LD和T3CON[3:2](预分频)应写入与所需超时周期对应的值。最大超时约为8192秒(T3LD = 0xFFFF，T3CON[3:2] = 11，预分频系数 = 4096)。

复位/中断或刷新定时器

默认超时周期约为32秒。当看门狗定时器递减到0时，看门狗定时器超时可以产生复位或中断。T3CON[1]位用于选择中断还是复位。这可用于调试。向只写寄存器T3CLRI写入0xCCCC，可以清除中断。

到期前向T3CLRI写入0xCCCC，可以防止产生中断或复位。写入T3CLRI会引起看门狗定时器立即重载T3LD(自由振荡模式下为0x1000)，以开始新的超时周期并重新计数。如果写入非0xCCCC值，则会产生中断或复位(由T3CON[1]选择)。

异步时钟源

看门狗定时器由内部32.768 kHz振荡器LFOSC提供时钟。T3STA寄存器中提供了3位用于确保定时器时钟和处理器时钟域正确同步。写入T3CON、T3LD或T3CLRI后，用户代码应等到对应状态位清0为止。

T3VAL寄存器始终包含同步到内核时钟域的定时器值。

看门狗定时器存储器映射寄存器

表194. 看门狗定时器存储器映射寄存器地址表(基地址：0x40002580)

偏移	名称	描述	访问类型	默认值
0x0000	T3LD	加载值。	RW	0x1000
0x0004	T3VAL	当前计数器值，只读。	R	0x1000
0x0008	T3CON	控制寄存器。	RW	0x00E9
0x000C	T3CLRI	清除中断。	W	不适用
0x0018	T3STA	状态寄存器。	R	0x0000

看门狗定时器加载寄存器

地址：0x40002580；复位：0x1000；名称：T3LD

表195. T3LD寄存器位功能描述

位	名称	描述
15:0	VALUE	加载值。

看门狗定时器当前值寄存器

地址：0x40002584；复位：0x1000；名称：T3VAL

表196. T3VAL寄存器位功能描述

位	名称	描述
15:0	VALUE	当前计数值。只读寄存器。

看门狗定时器控制寄存器

地址：0x40002588；复位：0x00E9；名称：T3CON

表197. T3CON寄存器位功能描述

位	名称	描述
15:7	保留	保留。这些位应该写入0。
6	MOD	定时器模式。 0: FREERUN: 工作在自由振荡模式。注意在自由振荡模式下，它会在0x1000时绕回。 1: PERIODIC: 工作在周期模式(默认)。
5	ENABLE	定时器使能。 0: 禁用定时器。只能进行一次。 1: 使能定时器(默认)。
4	保留	保留
3:2	PRE	预分频器。 00: DIV1: 源时钟/1。 01: DIV16: 源时钟/16。 10: DIV256: 源时钟/256(默认)。 11: DIV4096: 源时钟/4096。
1	IRQ	定时器中断。 0: 超时时产生复位(默认)。 1: 定时器超时时产生中断。此特性用于调试目的，仅在活动模式下可用。
0	PD	关断清零。 0: 退出关断模式时继续计数。 1: 退出关断模式时复位、重载并重启定时器计数(默认)。

看门狗定时器清除中断寄存器

地址：0x4000258C；复位：不适用；名称：T3CLRI

表198. T3CLRI寄存器位功能描述

位	名称	描述
15:0	VALUE	看门狗清零。 用户写入0xCCCC以复位/重载/重启T3或清除中断。 写入任何其它值都会引起看门狗复位/中断。只写，读出0。

从中断处理程序回到正常工作之前，确保寄存器写操作已全部完成。需要时可使用数据同步屏障(DSB)指令。下面的代码示例说明了如何在一个C程序中实现DSB ARM Cortex-M3指令。

```
void WDog_Tmr_Int_Handler()
{
 WdtClrInt();        // clear watchdog timer interrupt
__DSB();
}
```

看门狗定时器状态寄存器

地址：0x40002598；复位：0x0000；名称：T3STA

表199. T3STA寄存器位功能描述

位	名称	描述
15:5	保留	保留。
4	LOCK	锁定状态位。 当用户代码将T3CON[5]置1时，此位由硬件自动置1。 复位后清0，直到用户代码将T3CON[5]置1。
3	CON	T3CON写同步进行中。 置1表示定时器尚未准备好接收写入T3CON的命令。T3CON值的上次改变尚未在定时器时钟域中同步。 清0表示定时器已准备好接收写入T3CON的命令。T3CON值的上次改变已在定时器时钟域中同步。
2	LD	T3LD写同步进行中。 置1表示T3LD值的上次改变尚未在定时器时钟域中同步。 清0表示T3LD值的上次改变已在定时器时钟域中同步。
1	CLRI	T3CLRI写同步进行中。 当T3CLRI值正在定时器时钟域中进行更新时，此位自动置1，表示定时器配置尚未生效。 在定时器时钟域中清除中断时清0。
0	IRQ	WD_IRQ. T3中断待处理时置1。 无T3中断待处理时清0。

PWM

PWM特性

- 3对独立控制的PWM输出
- 2对支持H桥模式

PWM概述

ADuCM360/ADuCM361集成了一个6通道PWM接口。这6个输出分为三对(0至2)。0对和1对可配置为标准模式或驱动H桥。2对和3对只能配置为标准模式。表200列出了所有PWM对和模式。

表200. PWM通道分组

对	端口名称	描述	可用PWM模式
0	PWM0	高端PWM输出	H桥和标准
	PWM1	低端PWM输出	H桥和标准
1	PWM2	高端PWM输出	H桥和标准
	PWM3	低端PWM输出	H桥和标准
2	PWM4	高端PWM输出	标准
	PWM5	低端PWM输出	标准

上电后，0对和1对的PWM输出默认为H桥模式。所有模式下，用户既可以控制每一对输出引脚的工作周期，又可以单独控制每一个输出端的占空比。

当出现外部故障时，PWM_{TRIP}引脚上的下降沿可以迅速关闭PWM控制器。所有PWM输出都被置于关断状态，低端处于低电平状态，高端处于高电平状态，并且可以产生PWM_{TRIP}中断。

PWM工作原理

通过PWMCON0，可将PWM时钟频率设定为以下值之一：UCLK除以2、4、8、16、32、64、128或256。PWMxCOMx MMR用于控制所有模式下改变PWM输出状态的时间点。 实例如图34所示。

每一对都有一个相关的计数器。PWM周期长度由PWMxLEN定义。

PWM波形由16位定时器的计数值和比较寄存器的内容来决定。

当定时器计数值达到PWM0LEN时，低端波形(PWM1)变为高电平；当定时器计数值达到PWM0COM2内所保存的数值或者当高端波形(PWM0)变为低电平时，PWM1变为低电平。

当定时器计数值达到PWM0COM0内所保存的数值时，高端波形(PWM0)变为高电平；当定时器计数值达到PWM0COM1内所保存的数值时，PWM0变为低电平。

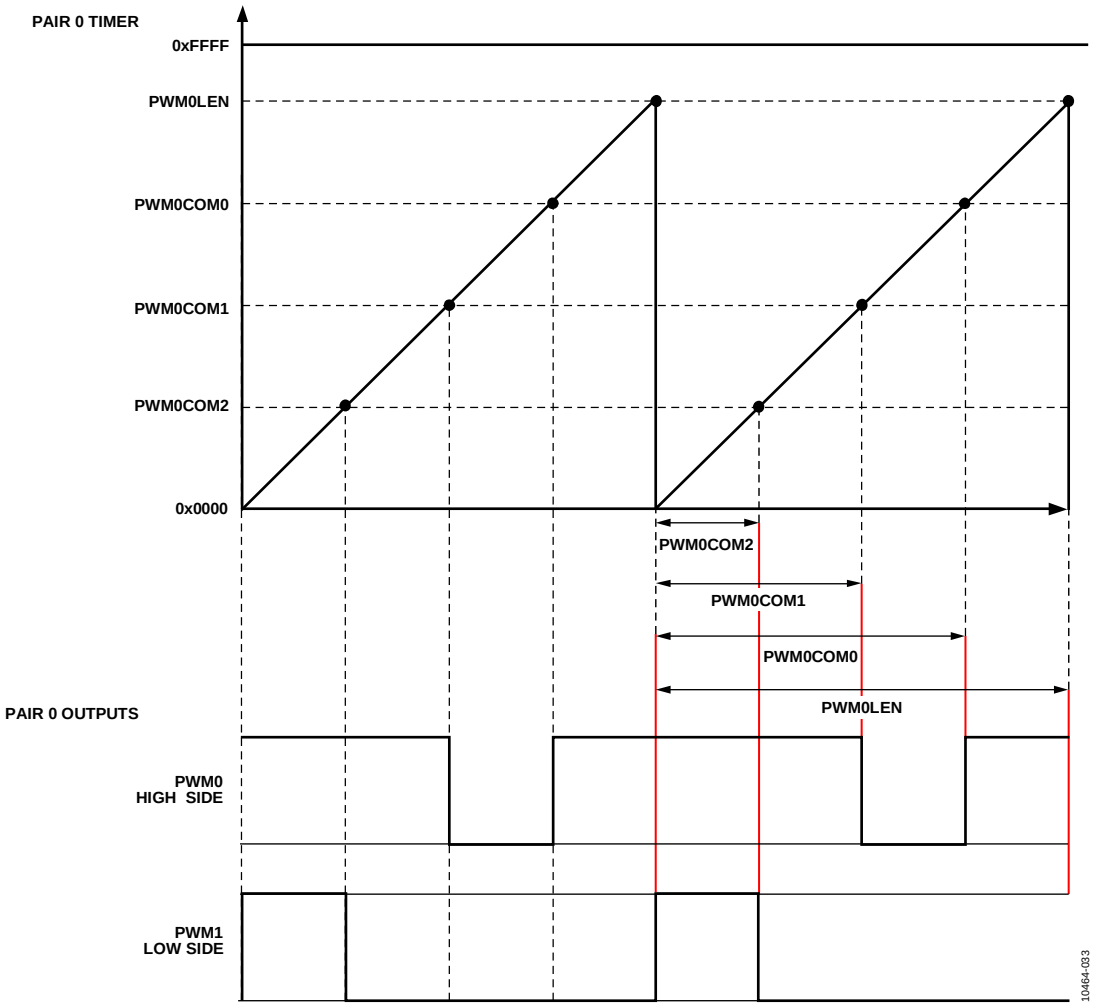


图34. 标准模式下PWM通道对波形

注意：每个通道的高端PWM输出的高电平持续时间必须大于或等于低端输出的高电平持续时间。例如，PWM0的高电平周期必须等于或大于PWM1的高电平周期。

表201列出了一个PWM通道的两个输出的周期和时长计算公式。

表201. PWM计算公式

PWM	周期	时长
低端(PWM1)	$t_{UCLK/DIV} \times (PWM0LEN + 1) \times N_{PRESCALE}$	高电平时长 如果(PWM0COM2 < PWM0COM1)，则 $t_{UCLK/DIV} \times (PWM0LEN - PWM0COM2) \times N_{PRESCALE}$ 其它 $t_{UCLK/DIV} \times (PWM0LEN - PWM0COM1) \times N_{PRESCALE}$
高端(PWM0)	$t_{UCLK/DIV} \times (PWM0LEN + 1) \times N_{PRESCALE}$	低电平时长 $t_{UCLK/DIV} \times (PWM0COM0 - PWM0COM1) \times N_{PRESCALE}$

请注意：

- $t_{UCLK/DIV}$为通过CLKCON1[14:12]和CLKSYSDIV[0]选择的PWM时钟频率。
- $N_{PRESCALE}$为PWMCON0[8:6]所决定的预分频器值。

标准模式

在标准模式下，各对可由所选寄存器单独控制，如表202所示。

表202. 标准模式下的比较寄存器描述(基址：0x40001000)

对	名称	描述
0	PWM0COM0	当PWM定时器计数达到该寄存器所保存的计数值时，PWM0输出变为高电平。
	PWM0COM1	当PWM定时器计数达到该寄存器所保存的计数值时，PWM0输出变为低电平。
	PWM0COM2	当PWM定时器计数达到该寄存器所保存的计数值时，PWM1输出变为低电平。
	PWM0LEN	当PWM定时器计数达到该寄存器所保存的计数值时，PWM1输出变为高电平。
1	PWM1COM0	当PWM定时器计数达到该寄存器所保存的计数值时，PWM2输出变为高电平。
	PWM1COM1	当PWM定时器计数达到该寄存器所保存的计数值时，PWM2输出变为低电平。
	PWM1COM2	当PWM定时器计数达到该寄存器所保存的计数值时，PWM3输出变为低电平。
	PWM1LEN	当PWM定时器计数达到该寄存器所保存的计数值时，PWM3输出变为高电平。
2	PWM2COM0	当PWM定时器计数达到该寄存器所保存的计数值时，PWM4输出变为高电平。
	PWM2COM1	当PWM定时器计数达到该寄存器所保存的计数值时，PWM4输出变为低电平。
	PWM2COM2	当PWM定时器计数达到该寄存器所保存的计数值时，PWM5输出变为低电平。
	PWM2LEN	当PWM定时器计数达到该寄存器所保存的计数值时，PWM5输出变为高电平。

以下代码将PWM0/PWM1输出的占空比配置为如下：PWM0高电平占整个周期的30%，PWM1高电平占整个周期的20%。

```
pADI_CLKCTL-> CLKCON0 = 0;
pADI_CLKCTL-> CLKCON1= 0;
pADI_PWM->PWM0LEN = 0x50;                       // Set PWM period to 21.9uS (456 kHz)
ucValid = PWM0_1DutyCycle(30,20);
                                                        // Pass values between 0-99 for duty cycle
of PWM0/1    - PWM 1 must have a duty cycle <=PWM0
unsigned char  PWM0_1DutyCycle( int iPWM0High, int iPWM1High)
{
   if ((iPWM1High >=100) ||(iPWM0High >=100) )
         return 1;                              // Error
   if  (iPWM0High >= iPWM1High)
   {
          pADI_PWM->PWM0COM0      =  PWM0LEN;
          pADI_PWM->PWM0COM1      =  ( int)((PWM0LEN * (iPWM0High)/100));
       pADI_PWM->PWM0COM2 = ( int)((PWM0LEN * iPWM1High)/100);
             return 0;
   }
   else
   {
         return 1;                              // PWM0 High period must be > PWM1 High period
   }
}
```

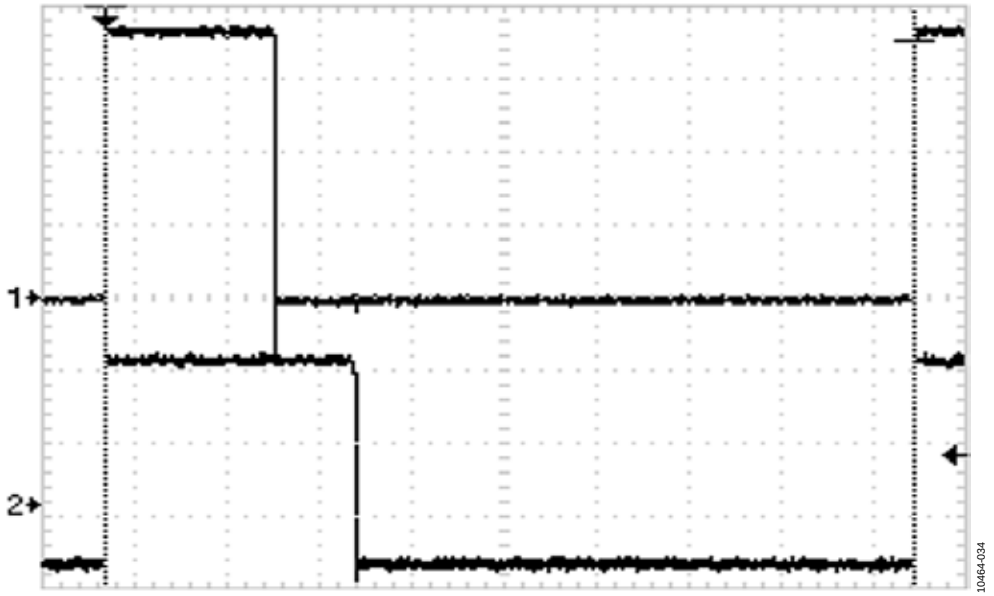


图35. 以上代码产生的PWM0和PWM1引脚上的PWM输出(PWM0为通道2)

H桥模式

在H桥模式下，4个输出的周期和占空比通过0对寄存器进行控制：PWM0COM0、PWM0COM1、PWM0COM2和PWM0LEN。另外，输出状态通过PWMCON0的位9、位5、位4和位2进行控制，如表203所示。

H桥配置的实例如图36所示。注意：仅PWM0至PWM3参与H桥模式；其它输出(PWM4和PWM5)不参与，继续产生标准模式输出。

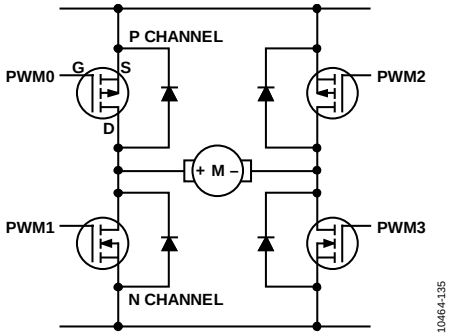


图36. H桥配置实例

表203. H桥模式下的PWM输出

PWM控制位				PWM输出[1]				
ENA PWMCON0[9]	POINV PWMCON0[5]	HOFF PWMCON0[4]	DIR PWMCON0[2]	PWM0	PWM1	PWM2	PWM3	电机状态
0	X	0	X	1 (禁用)	1 (使能)	1(禁用)	1(使能)	制动
X	X	1	X	1 (禁用)	0 (禁用)	1 (禁用)	0(禁用)	自由运行
1	0	0	0	0 (使能)	0 (禁用)	HS	LS	运动受PWM3上的LS控制
1	0	0	1	HS	LS	0 (使能)	0 (禁用)	运动受PWM1上的LS控制
1	1	0	0	$\overline{LS}$	$\overline{HS}$	1 (禁用)	1 (使能)	运动受PWM0上的$\overline{LS}$控制
1	1	0	1	1 (禁用)	1 (使能)	$\overline{LS}$	$\overline{HS}$	运动受PWM2上的$\overline{LS}$控制

[1] HS = 高端，LS = 低端，$\overline{HS}$ = 高端反向，$\overline{LS}$ = 低端反向，按照PWM0寄存器的设置。

PWM中断产生

PWM触发功能中断

当PWM触发功能使能(TRIPEN，PWMCON1[6])且PWM触发输入信号变为低电平(下降沿)时，PWM外设禁用其自身(PWMCON0[0] = 0)。它还会产生PWM触发中断。将PWMCLRI[4]置1可清除该中断。

使用PWM触发中断时，在退出中断服务程序(ISR)之前，应清除PWM中断。这样可以防止产生多个中断。

PWM输出对中断

在标准模式下，各PWM对都有一个专用中断：IRQPWM0、IRQPWM1、IRQPWM2。

当使能中断产生(PWMCON0[10])且0对的计数值从PWM0LEN变为0时，它也会产生IRQPWM0中断。

将PWMCLRI[0]置1可清除该中断。

当使能中断产生(PWMCON0[10])且1对的计数值从PWM1LEN变为0时，它也会产生IRQPWM1中断。

将PWMCLRI[1]置1可清除该中断。

当使能中断产生(PWMCON0[10])且2对的计数值从PWM2LEN变为0时，它也会产生IRQPWM2中断。

将PWMCLRI[2]置1可清除该中断。

在H桥模式下，0对和1对用于电桥配置，仅产生一个中断IRQPWM0。在0对和1对用于H桥模式的同时，2对可用于标准模式，并且可产生IRQPWM2中断。

PWM存储器映射寄存器

表204. PWM存储器映射寄存器地址表(基地址：0x40001000)

偏移	名称	描述	访问类型	默认值
0x000	PWMCON0	PWM控制寄存器0	RW	0x0012
0x004	PWMCON1	触发控制寄存器	RW8[1]	0x0000
0x008	PWMCLRI	PWM中断清除寄存器	W	0x0000
0x010	PWM0COM0	PWM0和PWM1的比较寄存器0	RW	0x0000
0x014	PWM0COM1	PWM0和PWM1的比较寄存器1	RW	0x0000
0x018	PWM0COM2	PWM0和PWM1的比较寄存器2	RW	0x0000
0x01C	PWM0LEN	PWM0和PWM1的周期值寄存器	RW	0x0000
0x020	PWM1COM0	PWM2和PWM3的比较寄存器0	RW	0x0000
0x024	PWM1COM1	PWM2和PWM3的比较寄存器1	RW	0x0000
0x028	PWM1COM2	PWM2和PWM3的比较寄存器2	RW	0x0000
0x02C	PWM1LEN	PWM2和PWM3的周期值寄存器	RW	0x0000
0x030	PWM2COM0	PWM4和PWM5的比较寄存器0	RW	0x0000
0x034	PWM2COM1	PWM4和PWM5的比较寄存器1	RW	0x0000
0x038	PWM2COM2	PWM4和PWM5的比较寄存器2	RW	0x0000
0x03C	PWM2LEN	PWM4和PWM5的周期值寄存器	RW	0x0000

[1] RW8为8位，读或写。

PWM控制寄存器0

地址：0x40001000；复位：0x0020；名称：PWMCON0

表205. PWMCON0寄存器位功能描述

对	名称	描述
15	SYNC	PWM同步。 0: 忽略PWMSYNC引脚的跃迁。 1: 检测到PWNSYNC引脚的下降沿之后，所有PWM计数器都在下一个时钟沿复位。
14	保留	
13	PWM5INV	2对低端极性(PWM5)。仅在标准模式下可用。 0: PWM5正常。 1: PWM5反转。
12	PWM3INV	1对低端极性(PWM3)。仅在标准模式下可用。 0: PWM3正常。 1: PWM3反转。
11	PWM1INV	0对低端极性(PWM1)。仅在标准模式下可用。 0: PWM1正常。 1: PWM1反转。
10	PWMIEN	0: 禁用PWM中断。 1: 使能PWM中断。
9	ENA	如果HOFF = 0且HMODE = 1。仅在H桥模式下可用。 0: 禁用PWM输出。 1: 使能PWM输出。
8:6	PRE	PWM时钟预分频器。设置UCLK分频系数。 000: UCLK/2。 001: UCLK/4。 010: UCLK/8。 011: UCLK/16。 100: UCLK/32。 101: UCLK/64。 110: UCLK/128。 111: UCLK/256。
5	POINV	PWM极性。仅在H桥模式下可用。 0: PWM输出正常。 1: 所有PWM输出都反转。
4	HOFF	屏蔽高端。仅在H桥模式下可用。 0: PWM输出正常。 1: 强制PWM0和PWM2输出高电平，PWM1和PWM3输出低电平(默认)。
3	LCOMP	载入比较寄存器。在H桥和标准模式下均可用。 0: 使用内部比较寄存器中先前存储的值。 1: 在PWM定时器下一次从0x00变为0x01时(PWM定时器上溢)，将PWMxCOMx的值载入内部比较寄存器。
2	DIR	方向控制。仅在H桥模式下可用。 0: 使能PWM2和PWM3用作输出信号，PWM0和PWM1保持低电平。 1: 使能PWM0和PWM1用作输出信号，PWM2和PWM3保持低电平。
1	MOD	PWM工作模式。 0: PWM处于标准模式。 1: 使能H桥模式和PWMCON0的位[5:2](默认)。
0	ENABLE	PWM输出使能。 0: 禁用所有PWM输出。 1: 使能所有PWM输出。

请注意：

- 除LCOMP外，PWMCON0寄存器的所有其它位只能在ENABLE为低电平时更改。
- 向LCOMP写入值1时，它将保持该值，直到新值载入所有通道的比较寄存器。

PWM1控制寄存器

地址：0x40001004；复位：0x0020；名称：PWMCON1

表206. PWMCON1寄存器位功能描述

位	名称	描述
15:7	保留	保留。
6	TRIPEN	0: 禁用PWM触发功能。 1: 使能PWM触发功能。
5:0	保留	保留。

PWMCLRI寄存器

地址：0x40001008；复位：0x0020；名称：PWMCLRI

表207. PWMCLRI寄存器位功能描述

位	名称	描述
15:5	保留	保留。
4	TRIP	1: 清除锁存的TRIPIRQPWM中断。 此位读出值始终为0。
3	保留	此位读出值始终为0。
2	PWM2	1: 清除锁存的IRQPWM2中断。 此位读出值始终为0。
1	PWM1	1: 清除锁存的IRQPWM1中断。 此位读出值始终为0。
0	PWM0	1: 清除锁存的IRQPWM0中断。 此位读出值始终为0。

电源支持电路

电源支持电路特性

低压差稳压器

ADuCM360/ADuCM361集成一个片内稳压器(基于AVDD_REG的LDO)，其通过AVDD电源直接驱动来产生一个1.8 V内部电源。

然后，该稳压电源用作ARM Cortex-M3和外设的电源，包括那些片内精密模拟电路。LDO有自己的输出引脚AVDD_REG，它需要一个0.47 µF外部电容以保持稳定。选择器件工作模式时，LDO自动开启和关闭。不能过驱LDO。

引脚7 (DVDD_REG)必须通过一个470 nF电容连接至DGND。引脚18 (AVDD_REG)必须通过一个470 nF电容连接至AGND。引脚7 (DVDD_REG)和引脚18 (AVDD_REG)还必须互连。参见图39。

上电复位

同时集成了上电复位(POR)功能，用以确保处理器安全工作。POR电路旨在保证基于Flash/EE存储器的处理器在上电和关断周期中完全正常工作。

如图37所示，当AVDD上的电源电压达到最小工作电压1.6 V且LDO输出为1.65 V时，POR信号将处理器保持在复位状态下50 ms。POR输出通过P0.6提供。

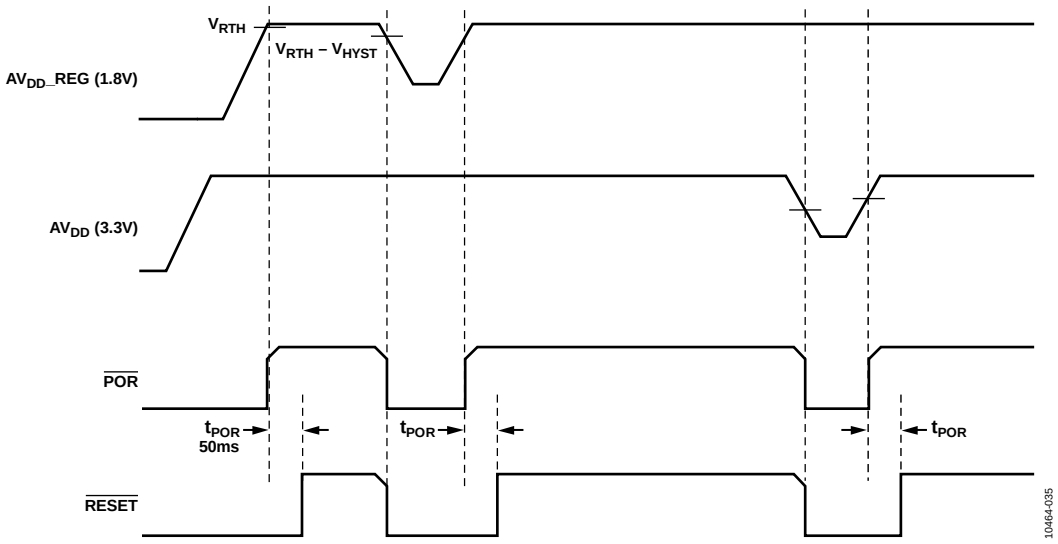


图37. 典型上电周期

硬件设计考虑

引脚配置和功能描述

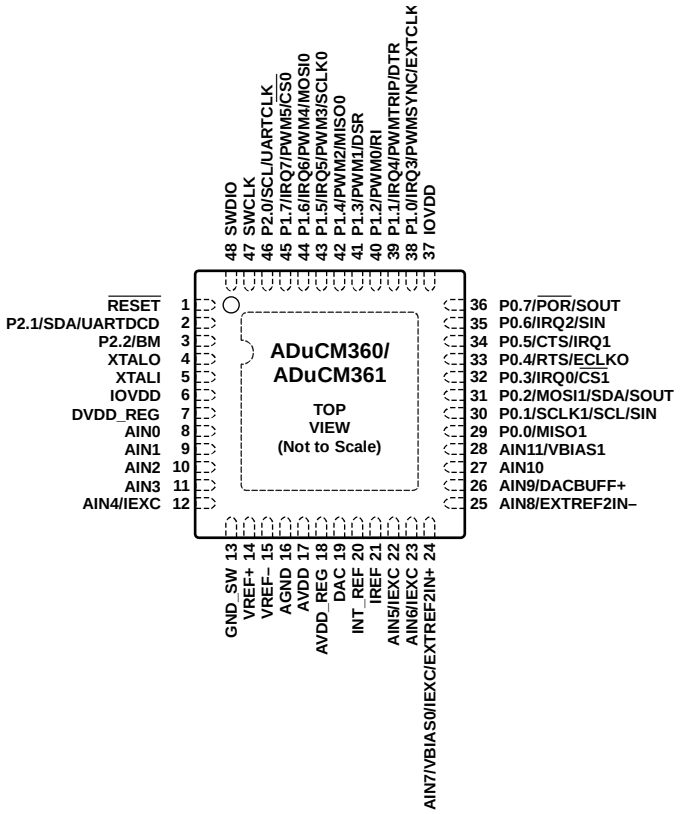


图38. 引脚配置

表208. 引脚功能描述

引脚编号	引脚名称	描述
1	RESET	复位引脚，低电平输入有效。提供一个内部上拉电阻。
2	P2.1/SDA/UARTDCD	通用输入/输出P2.1/I²C串行数据引脚/UART数据载波检测引脚。
3	P2.2/BM	通用输入/输出P2.2/引导模式输入检测引脚。 当该引脚在任意复位序列中及其后的较短时间内保持低电平时，该器件进入UART下载模式。
4	XTALO	外部晶体振荡器输出引脚。针对实时时钟的可选32.768 kHz源。
5	XTALI	外部晶体振荡器输入引脚。针对实时时钟的可选32.768 kHz源。
6	IOVDD	数字系统电源引脚。此引脚必须通过一个0.1 µF电容连接至DGND。
7	DVDD_REG	此引脚必须通过一个470 nF电容连接至DGND，并且连接到引脚18 (AVDD_REG)，如图39所示。
8	AIN0	ADC模拟输入0。该引脚能够以差分或单端模式配置为任意ADC的正或负输入。
9	AIN1	ADC模拟输入1。该引脚能够以差分或单端模式配置为任意ADC的正或负输入。
10	AIN2	ADC模拟输入2。该引脚能够以差分或单端模式配置为任意ADC的正或负输入。
11	AIN3	ADC模拟输入3。该引脚能够以差分或单端模式配置为任意ADC的正或负输入。
12	AIN4/IEXC	ADC模拟输入4/激励电流源。 该引脚能够以差分或单端模式配置为任意ADC的正或负输入(AIN4)。 该引脚还可配置为激励电流源0或激励电流源1(IEXC)的输出引脚。
13	GND_SW	传感器电源切换至模拟地基准电压。

引脚编号	引脚名称	描述
14	VREF+	外部基准电压正输入。外部基准电压可施加在VREF+和VREF−引脚之间。
15	VREF−	外部基准电压负输入。外部基准电压可施加在VREF+和VREF−引脚之间。
16	AGND	模拟系统地基准引脚。
17	AVDD	模拟系统电源引脚。此引脚必须通过一个0.1 µF电容连接至AGND。
18	AVDD_REG	内部模拟稳压器电源输出。此引脚必须通过一个470 nF电容连接至AGND，并且连接到引脚7 (DVDD_REG)，如图39所示。
19	DAC	DAC电压输出。
20	INT_REF	内部基准电压源。此引脚必须通过一个470 nF去耦电容连接至地。
21	IREF	针对激励电流源的可选基准电流电阻连接。用于激励电流源的基准电流通过一个连接至该引脚的低漂移(5 ppm/°C)外部电阻设置。
22	AIN5/IEXC	ADC模拟输入5/激励电流源。该引脚能够以差分或单端模式配置为任意ADC的正或负输入(AIN5)。该引脚还可配置为激励电流源0或激励电流源1(IEXC)的输出引脚。
23	AIN6/IEXC	ADC模拟输入6/激励电流源。该引脚能够以差分或单端模式配置为任意ADC的正或负输入(AIN6)。该引脚还可配置为激励电流源0或激励电流源1(IEXC)的输出引脚。
24	AIN7/VBIAS0/IEXC/EXTREF2IN+	ADC模拟输入7/偏置电压输出/激励电流源/外部基准电压2正输入。该引脚能够以差分或单端模式配置为任意ADC的正或负输入(AIN7)。该引脚还可配置为：模拟输出引脚，以产生偏置电压(AVDD_REG/2的VBIAS0，VBIAS0)；激励电流源0或激励电流源1(IEXC)的输出引脚；或外部基准电压2的正输入(EXTREF2IN+)。
25	AIN8/EXTREF2IN−	ADC模拟输入8/外部基准电压2负输入。该引脚能够以差分或单端模式配置为任意ADC的正或负输入(AIN8)。该引脚还可配置为外部基准电压2的负输入(EXTREF2IN−)。
26	AIN9/DACBUFF+	ADC模拟输入9/DAC输出缓冲器的同相输入。该引脚能够以差分或单端模式配置为任意ADC的正或负输入(AIN9)。当DAC配置为NPN模式时，该引脚还可配置为DAC输出缓冲器的同相输入(DACBUFF+)。
27	AIN10	ADC模拟输入10。该引脚能够以差分或单端模式配置为任意ADC的正或负输入。
28	AIN11/VBIAS1	ADC模拟输入11/偏置电压输出。该引脚能够以差分或单端模式配置为任意ADC的正或负输入(AIN11)。该引脚还可配置为模拟输出引脚，以生成偏置电压(AVDD_REG/2的VBIAS1，VBIAS1)。
29	P0.0/MISO1	通用输入/输出P0.0/SPI1主机输入、从机输出引脚。
30	P0.1/SCLK1/SCL/SIN	通用输入/输出P0.1/SPI1串行时钟引脚/I²C串行时钟引脚/UART串行输入(UART下载器的数据输入)。
31	P0.2/MOSI1/SDA/SOUT	通用输入/输出P0.2/SPI1主机输出、从机输入引脚/I²C串行数据引脚/UART串行输出(UART下载器的数据输出)。
32	P0.3/IRQ0/$\overline{CS1}$	通用输入/输出P0.3/外部中断请求0/SPI1芯片选择引脚(低电平有效)。
33	P0.4/RTS/ECLKO	通用输入/输出P0.4/UART请求发送信号/用于测试的外部时钟输出引脚。
34	P0.5/CTS/IRQ1	通用输入/输出P0.5/UART清零发送信号/外部中断请求1。
35	P0.6/IRQ2/SIN	通用输入/输出P0.6/外部中断请求2/UART串行输入。UART下载器不使用。
36	P0.7/$\overline{POR}$/SOUT	通用输入/输出P0.7/上电复位引脚(低电平有效)/UART串行输出。UART下载器不使用。
37	IOVDD	数字系统电源引脚。此引脚必须通过一个0.1 µF电容连接至DGND。
38	P1.0/IRQ3/PWMSYNC/EXTCLK	通用输入/输出P1.0/外部中断请求3/PWM外部同步输入/外部时钟输入引脚。
39	P1.1/IRQ4/PWMTRIP/DTR	通用输入/输出P1.1/外部中断请求4/PWM外部触发输入/UART数据终端就绪引脚。
40	P1.2/PWM0/RI	通用输入/输出P1.2/PWM0输出/UART响铃指示引脚。
41	P1.3/PWM1/DSR	通用输入/输出P1.3/PWM1输出/UART数据设置就绪引脚。
42	P1.4/PWM2/MISO0	通用输入/输出P1.4/PWM2输出/SPI0主机输入、从机输出引脚。

引脚编号	引脚名称	描述
43	P1.5/IRQ5/PWM3/SCLK0	通用输入/输出P1.5/外部中断请求5/PWM3输出/SPI0串行时钟引脚。
44	P1.6/IRQ6/PWM4/MOSI0	通用输入/输出P1.6/外部中断请求6/PWM4输出/SPI0主机输出、从机输入引脚。
45	P1.7/IRQ7/PWM5/$\overline{CS0}$	通用输入/输出P1.7/外部中断请求7/PWM5输出/SPI0芯片选择引脚(低电平有效)。
46	P2.0/SCL/UARTCLK	通用输入/输出P2.0/I²C串行时钟引脚/仅用于UART模块的输入时钟引脚。
47	SWCLK	串行线路调试时钟输入引脚。
48	SWDIO	串行线路调试数据输入/输出引脚。
	EP	裸露焊盘。LFCSP具有exposed pad，出于机械方面的考虑，必须将其焊接在PCB的金属片及DGND上。

典型系统配置

图39显示ADuCM360/ADuCM361的典型配置。该图展示了某些硬件考虑因素。LFCSP封装的底部具有exposed pad，出于机械方面的考虑，必须将其焊接在PCB的金属片及DGND上。PCB上的金属片可以连接到地。AVDD_REG和DVDD_REG引脚上的0.47 µF电容应尽可能靠近引脚放置。在高噪声环境下，可添加一个额外的1 nF电容至IOVDD和AVDD。

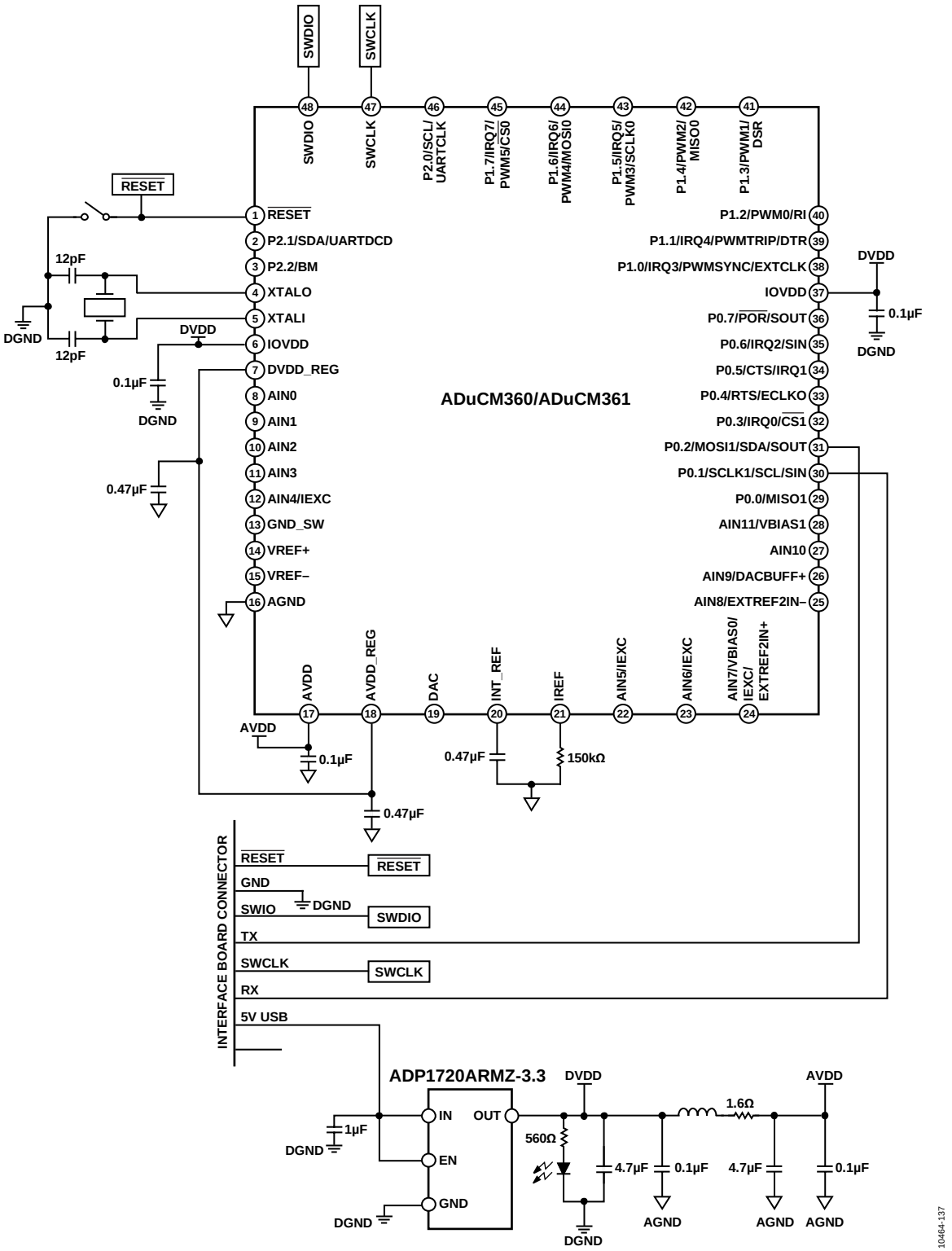


图39. 典型系统配置

串行线调试接口

串行线调试(SWD)为引脚有限的封装提供了一个调试端口。SWD用一个时钟引脚(SWDCLK)和一个双向数据引脚(SWDIO)取代了5引脚JTAG端口，提供所有一般的JTAG调试和测试功能。对于ARM 20引脚JTAG接口，SWDIO和SWCLK叠加在TMS和TCK引脚上。

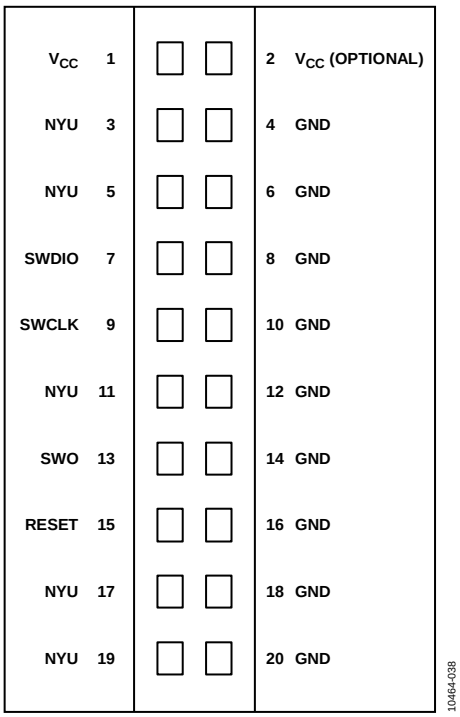


图40. SWD 20引脚连接器布局

表209. SWD连接

信号	连接到
SWDIO	数据I/O引脚。使用一个100 kΩ上拉电阻接VCC。
SWO	不连接。
SWCLK	时钟引脚。使用一个100 kΩ上拉电阻接VCC。
VCC	正电源电压—JTAG接口驱动器的电源。
GND	数字地。
RESET	不连接。

相关链接

资源	描述
ADuCM360	ADuCM360产品页面
ADuCM361	ADuCM361产品页面
ADuCM360/ADuCM361 Data sheet	ADuCM360/ADuCM361数据手册

I²C指最初由Philips Semiconductors(现为NXP Semiconductors)开发的一种通信协议。

ESD警告

ESD(静电放电)敏感器件。带电器件和电路板可能会在没有察觉的情况下放电。尽管本产品具有专利或专有保护电路，但在遇到高能量ESD时，器件可能会损坏。因此，应当采取适当的ESD防范措施，以避免器件性能下降或功能丧失。

UG10464sc-0-9/14(D)

www.analog.com