连接 ADS8371 与 TMS3206713 DSP

Lijoy Philipose
数据采集 - 数字/模拟转换器

摘要

此应用报告提供了连接 ADS8371 - 16 位、750KSPS 并行接口转换器 - 与 TMS320™ DSP 系列的 TMS320C6713 数字信号处理器的解决方案。 此硬件解决方案由现有的硬件，特别是 ADS8371EVM、C6713 DSK 和 5-6K 接口板组成。 此软件演示了如何使用 EDMA 交替缓冲存储器和 Timer1 外设在 740kHz 收集数据。 同时还讨论了在编写此软件以及为您的应用修改此软件时要记住的某些要点。 已开发的软件可用于下载，可供用户讨论并在应用开发中作为采样代码使用。

内容

附图目录

附表目录

1 介绍

ADS8371 转换器为 16 位、750kHz 的并行接口模数转换器。 此应用报告提供了使用此转换器及将其与 TMS320C6713 DSP 连接的硬件和软件解决方案。 已开发的软件使用了 EDMA，并结合 Timer1 可收集 4096 个采样。 还讨论了连接转换器与主处理器时应记住的一些要点。

1.1 硬件

硬件解决方案包括 TMS320C6713 DSK (C6713 DSK)、5-6K 接口板和 ADS8371EVM。 此报告中使用的硬件可向德州仪器 (TI) 订购。

1.2 TMS320C6713 DSK

TMS320C6713 DSP 入门套件 (DSK) 不仅介绍了 C6000™ DSP 平台技术，其足够强大的功能还可用于网络、通讯、成像以及数据采集等其它应用领域的快速开发。 有关详情，请在 http://www.ti.com 上搜索器件型号 TMDSDSK6713。

1.3 ADS8371 EVM

ADS8371 评估板 (EVM) 是测试此 16 位模数转换器的功能和动态性能的一种简单方法。 此评估板中包含的电路对于显示转换器的性能以及将其连接至并行总线至关重要。 这些电路包含模拟输入、参考、电源、数字缓冲存储器电路以及解码逻辑。 数字输入和输出将被缓冲，以将转换器与大多数共享总线类型系统的常见数字噪声相隔离。 可以通过标准的 0.1 英寸 IDC 排针/排母 (P1) 或 SMA 连接器 (J1) 应用此模拟信号。 通过 0.1 英寸的 IDC 排针/排母连接器 (P3) 获取缓冲数据总线。 也可以通过标准的 0.1 英寸 IDC 排针/排母 (J2) 获取 ADS8371 控制输入。 用于生成转换器启动、读取和重置信号的解码逻辑将通过连接器 P2 进行控制。 通过这些标准连接器，可以将 EVM 插入大多数原型板中，以进行快速评估。 有关本产品的信息，请在 http://www.ti.com 中搜索 ADS8371EVM。 表 1 列出了此应用报告中使用的跳线设置。 此解码逻辑将查找三个输入 A0、A1 和 A2。 对于此应用，DSP 的地址行 A14、A15 和 A16 将分别映射至 A0、A1 和 A2。

表 1. EVM 的跳线设置

编号	说明	跳线	
		引脚 1-2	引脚 2-3
W1	应用 +5VD 至 +BVDD。	已安装	
	通过连接器连接 +3.3VD 和 +BVDD		未安装
W2	设置 A[2..0]= 0x1 以生成 RD 脉冲	已安装	
	设置 A[2..0]= 0x2 以生成 RD 脉冲		未安装
W3	设置 A[2..0]= 0x3 以生成转换脉冲	已安装	
	设置 A[2..0]= 0x4 以生成转换脉冲		未安装
W4	对 P2 的引脚 19 (INTC) 应用反向 BUSY 信号。	未安装	
	将 BUSY 脉冲设置至 INTc		已安装

1.4 5-6K 接口评估模块

德州仪器 (TI) 提供的许多数据采集评估板 (EVM) 都设计为具有一套通用的连接器以及连接器中使用的信号。 5-6K 接口板使设计者可以轻松地将这些 EVM 连接至 C5000™ DSP 和 C6000™ DSP 入门套件。

5-6K 接口板包含两个串行连接器、两个信号调节区和一个并行接口。 EVM 将插入连接器的 J10（模拟）、J17（数据总线）、J18（控制总线）和 JP5（电源）。 查看 TI 文献编号 SLAU104 以了解有关 5-6K 接口板的详情，或搜索关键字 *5-6K 接口*。

表 2 和 3 列出了 5-6K 接口板的跳线设置。 确保在 5-6K 接口板上将连接器 J13 的引脚 7 和 8 短路。 跳线经过 ADS8371EVM 的 INTc 连接至 5-6K 板上的 INTd，最后到达 DSP 上的外部中断编号 7 (INT7) 引脚。

表 2. 5-6K 接口板 A 版

参考编号	跳线设置		备注
	1 月 2 日	2 月 3 日	
W1	断开	无	
W2		短路	
W3		短路	
W4		短路	
W5		短路	
W6		短路	
J14	引脚 3 和 4 间短路		*OUTb* 旁的引脚
J13	引脚 7 和 8 间短路		DC_INTd

表 3. 5-6K 接口板 B 版

参考编号	跳线设置		备注
	1 月 2 日	2 月 3 日	
W1	断开	无	
W2		短路	

表 3. 5-6K 接口板 B 版(接上表)

参考编号	跳线设置		备注
	1 月 2 日	2 月 3 日	
W3		短路	
W4		短路	
W5		短路	
W6	短路		
W7	短路		
W8	短路		
W9	短路		
W10	短路		
W11	短路		
W12	短路		
W13	短路		
J14	引脚 3 和 4 间短路		*OUTb* 旁的引脚
J13	引脚 7 和 8 间短路		DC_INTd

1.5 硬件连接

将一个 ADS8371 映射至 C6713 DSP 的存储器空间 CE2。 使用 EVM 上的一个 3 至 8 解码器生成读取 - 转换启动信号。 使用地址行 A16、A15 和 A14 生成发送至模数转换器的读取 - 转换启动信号。 C6713 具有一个 32 位数据总线；因此，通过将 BYTE 线连接 LOW 即可启用 16 位总线操作。 将 BUSY 信号应用至 DSP 的外部中断引脚 7。 片选 ($\overline{CS}$) 信号接地。 如果在您的应用中尚未考虑功率消耗，则 $\overline{CS}$ 可以永久连接低电平。 用户仅需要提供读取 ($\overline{RD}$) - 转换启动 ($\overline{CONVST}$) 信号即可。

最后，通过数据总线连接 LSB 和 LSB。 硬件连接如图 1 所示。

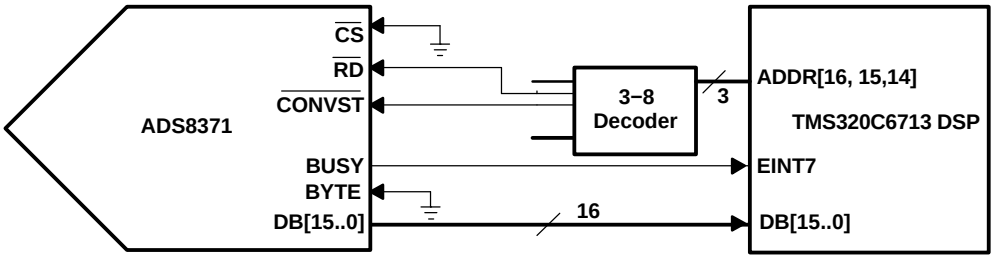

图 1. 硬件连接

2 软件接口

其目的是尽快收集大量采样，同时释放处理器以执行其它任务。 只有在准备好处理采样时，才能向处理器发出预警。 执行此任务的最高效的方法是使用几个 EDMA 通道、一个定时器以及中断。 此讨论假设读者已熟悉 DSP 和其外设。 如果情况不是如此，则读者需要研究此文档结尾参考部分中列出的应用报告和数据表。

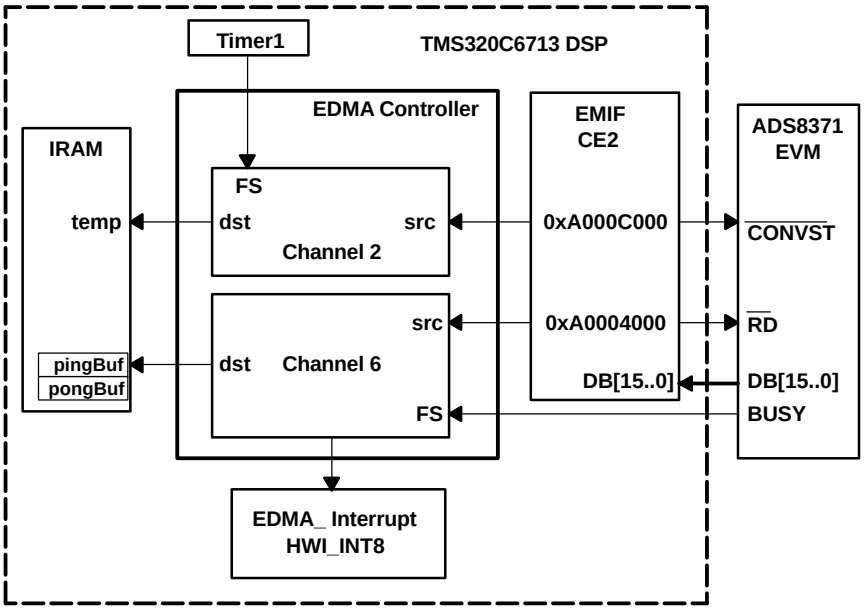

图 2. 使用 EDMA 的数据传输结构图

DSP 将在固定时间内完成此任务。 已启用的定时器将继续触发新的转换，而 EDMA 则继续启动新的转换并从该转换器中读取数据。 采样频率较高时，在禁用定时器和 EDMA 通道之前，CPU 需要运行几个转换周期。 EDMA 将从此转换器读取数据，并将数据存储在目标地址中。 出于此种考虑，建议在 EDMA 完成收集所需采样数量后，连接其通道以将数据传送至其它位置。 要查看其运行原理，请打开 Code Composer Studio™ 中的文件 Config.cdb，然后如 Figure 3 中所示展开。 EDMA 配置 7 (edmaCfgChan7) 将指示 EDMA 向 *pingBuf*[ ] 写入 2048 个字。 EDMA 配置 7A (edmaCfgChan7a) 将强制 EDMA 向 *pongBuf*[ ] 写入 2048 个字。 EDMA 配置 0 (edmaCfg0) 强制 EDMA 把转换器的数据写入变量 *temp*。 EDMA 继续将采样写入 *temp*，直至 CPU 禁用 Timer1 及相应的 EDMA 通道。

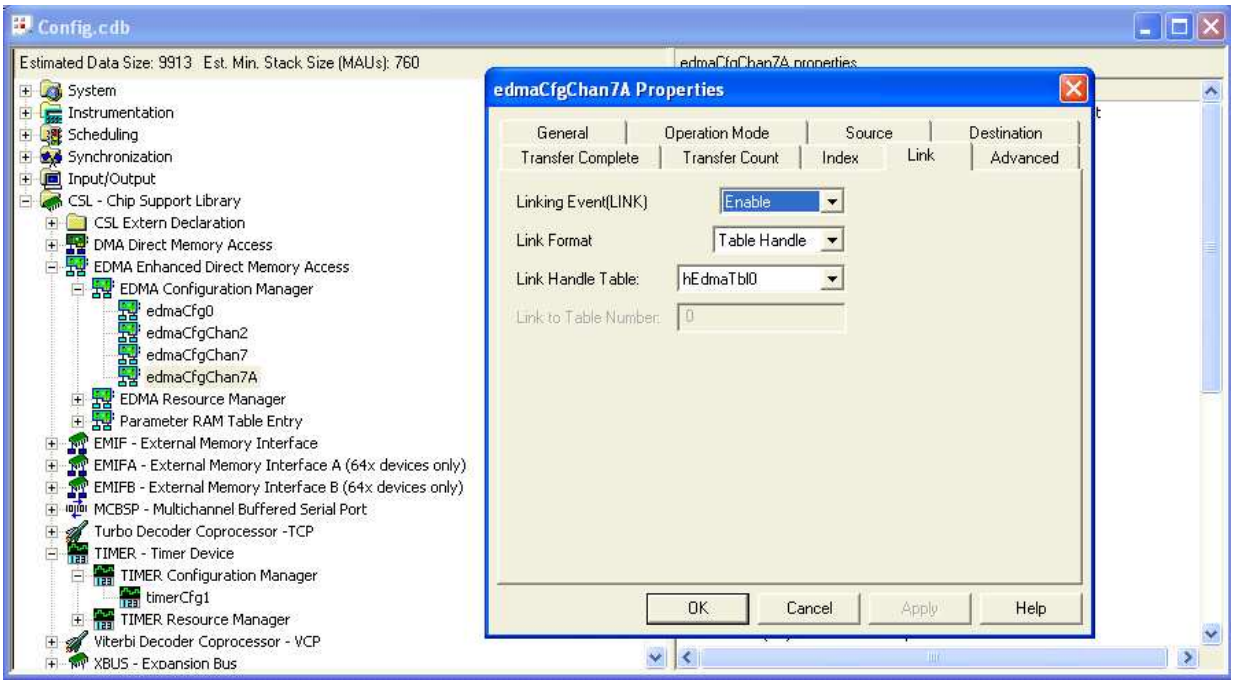

图 3. DSP/BIOS 配置文件的抓图

2.1 ADS8371

ADS8371 是一个逐次逼近寄存器 (SAR) 类型的转换器。 SAR 类型转换器可在采样与保持两种模式下工作。 在采样模式下，转换器的采样和保持电容器将连接至连接器的输入引脚。 在此模式下，模拟输入缓冲存储器必须将信号设置为半个 LSB。 假设参考电压为 4.096V，单端输入范围为 4.096V，则 LSB 的一半是 62.5ÂµV 在保持模式下，采样和保持电容器将与输入引脚断开。 转换器进入保持模式时，可在内部电容器阵列上捕获 +IN 和 – IN 输入间的电压差。

– IN 输入上的电压被限制在 – 0.2V 和 0.2V 之间，从而使此输入可以抑制 +IN 和 – IN 输入通用的小信号。 +IN 输入的范围介于 – 0.2V 和参考电压 + 0.2V 之间。 输入范围 (+IN – (– IN)) 被限制在 0V 到参考电压之间。

ADS8371 可以使用 2.5V 到 4.2V 之间的外部参考电压 (Vref) 源供电。 转换器的输入引脚 1 上的参考电压 (REFIN) 将从内部进行缓冲。 需要在此引脚上应用一个清洁、低噪声、良好去耦参考电压，以确保转换器的良好性能。

加电后前三个转换用于为特定器件加载出厂时存储的微调数据，以确保达到指定的精度。 前三个转换的结果无效，应该将其丢弃。

在 BUSY 信号处于高电平时，可以通过组合 $\overline{CS}$ 和 $\overline{CONVST}$ 重置器件。 首先，当 $\overline{CS}$ 处于低电平，并且内部 CONVERT 状态处于高电平时，就会发出一个 $\overline{CONVST}$。 $\overline{CONVST}$ 的下降沿将触发重置。 其次，当内部 CONVERT 状态处于高电平时，就会发出一个 $\overline{CS}$。 $\overline{CS}$ 的下降沿将触发重置。 上述两种情况中的任一种都会为转换器触发一次内部自清除重置。 一旦重置器件，所有输出闭锁将被清除（数据总线置零），并将 BUSY 信号设置为低电平。 内部重置后，立即在 BUSY 信号的下降沿启动一个新的采样期间。

采取某些预防措施是必要的，可以确保数字输入端上的转换不会影响转换器的性能。 转换器将在 $\overline{CONVST}$ 输入引脚的下降沿从采样模式转换为保持模式。 一个清洁且低抖动的下降沿对于实现转换器的最佳性能是至关重要的。 $\overline{CONVST}$ 引脚上的急速下降转换会影响采样 - 保持电容器上存储的电压。 要求下降转换的时间范围要介于 10ns 到 30ns 之间，以达到转换器的额定性能。 此报告中所显示的屏幕快照中的所有信号都是通过 ADS8371EVM 的连接器 J2 上的示波器探针获取的。 转换器引脚上的 $\overline{RD}$ 和 $\overline{CONVST}$ 的实际转换时间与屏幕快照中所显示的 BUSY 信号的转换时间很相似。

此外，为确保最佳性能，建议输入引脚上的所有活动都发生在转换期间的前 400ns 以及 $\overline{CONVST}$ 的下降沿前 125ns 这两个范围内。

2.2 C6713 DSP

需要设置 TMS320C6713 DSP 及其各自的外设（Timer1、EMIF 和 EDMA），以与转换器共同工作。 此处假设读者对主处理器的工作原理有一定的了解，因此本报告中将不会详细描述 DSP 设置。 查看第 4 部分中所列出的可下载代码和参考以了解更多信息。

可以在如图 3 所示的 config.cdb 文件中找到各种外设的设置，并可以对这些外设进行修改。 DSP/BIOS 配置文件的种子文件是 dskC6713.cdb 文件。

以下列出的代码显示的是 EDMA 通道 2 的寄存器设置，此通道可用于触发转换周期。 该源是通过 EVM 上的解码器生成转换启动脉冲的地址。 数据传输的目的地是一个称为 *temp* 的变量 —，该变量用于存储可以丢弃的数据。 EDMA 应该传输 NUMSAMPLES 或 2048 个采样数据点。 NUMSAMPLES 定义在 *dc_conf.h* 文件中。 每一个传输框架都与一个 Timer1 事件同步。 因此，每个传输的频率或转换启动脉冲就是 Timer1 的频率。

```
EDMA_Config edmaCfgChan2 = {
    0x28020003,        /*  Option  */
    0xA000C000,        /*  Source Address - Numeric    */
    0x00000000,        /*  Transfer Counter - Numeric  */
    (Uint32) &temp     /*  Destination Address - Extern Decl. Obj  */
    0x00000000,        /*  Index register - Numeric  */
    0x00010000         /*  Element Count Reload and Link Address  */
};
```

以下列出的代码显示的是 EDMA 通道 7 的寄存器设置。 正如上文提到的，此通道是用于从转换器读取数据的。 该源是使用 EVM 上的解码器生成读取脉冲的地址。 一个名为 *pingBuf[ ]* 的阵列是第一块 2048 个数据点的目的地。 第二块 2048 个数据的目的地是 *pongBuf[ ]*。 两个阵列都是 BLOCK_SZ (2048) 个字长。 EDMA 通道 7 将传输 NUMSAMPLES(2048)，然后加载第二个配置，从而将目标地址设置为 *pongBuf[ ]* 阵列。 从 A/D 转换器传输了 4096 个字后，它将加载第三个配置，从而将目标地址设置为 *temp* 的地址。 每个 EDMA 转换都与外部中断 7 同步。 在这种情况下，它是来自 ADS8371EVM 的 BUSY 信号。 EDMA 传输与帧同步信号的上升沿同步。 帧同步信号是 BUSY 信号的上升沿。 BUSY 的上升沿指示转换启动，而不是结束；因此应该丢弃 *pingBuf[ ]* 阵列中的第一个点。

```
EDMA_Config edmaCfgChan7 = {
    0x28360003,        /*  Option  */
    0xA0004000,        /*  Source Address - Numeric    */
    0x00000000,        /*  Transfer Counter - Numeric  */
    (Uint32) pingBuf,  /*  Destination Address - Extern Decl. Obj  */
    0x00010001,        /*  Index register - Numeric  */
    0x00010000         /*  Element Count Reload and Link Address  */
};
```

```
EDMA_Config edmaCfgChan7A = {
    0x28360003,        /*  Option  */
    0xA0004000         /*  Source Address - Numeric    */
    0x00000000,        /*  Transfer Counter - Numeric  */
    (Uint32) pongBuf,  /*  Destination Address - Extern Decl. Obj  */
    0x00010001,        /*  Index register - Numeric  */
    0x00010000         /*  Element Count Reload and Link Address   */
};
```

采样频率较高时，DSP 无法在收集所有所需采样后立即禁用定时器和 EDMA 通道。 只要一启用定时器和 EDMA 通道，它们就会继续触发并传输数据。 数据将被存储在 EDMA 通道 7 目标寄存器的指定地址中。 因此这就存在一种可能，那就是在 EDMA 通道在硬件中断服务例程中被禁用之前，数据阵列中的最后一个点已被进行多次改写。 为避免损坏采样数据，需要将 EDMA 通道连接至配置 edmaCfg0。 在 *读取* EDMA 通道捕获 4096 个采样后，EDMA 开始将采样数据存储到变量 *temp* 中。 通过 EDMA 的连接特性，可以轻松地实施交替缓冲存储器，并找到以 740KSPS 的转换率捕获采样的方法。

```
EDMA_Config edmaCfg0 = {
    0x48160003,        /*  Option  */
    0xA0004000,        /*  Source Address - Numeric    */
    0x00000000,        /*  Transfer Counter - Numeric   */
    (Uint32) &temp,    /*  Destination Address - Extern Decl. Obj  */
    0x00010001,        /*  Index register - Numeric  */
    0x00010000         /*  Element Count Reload and Link Address  */
};
```

设置采样率的 Timer1 的寄存器设置如下所示。 Timer1 的输入时钟源为 CPU CLOCK 除以 4。

Timer1 输入时钟 = 225E6/4 = 56.25MHz

设置采样频率 (F_s) 的公式是

F_s = 56.25e6 /（2*期间值） 或

期间值 = 56.25e6/(2*F_s)

```
TIMER_Config timerCfg1 = {
    0x000003D1,        /*  Control Register (CTL)   */
    0x00000026,        /*  Period Register (PRD)    */
    0x00000000         /*  Counter Register (CNT)   */
};
```

为时钟模式设置 Timer1 输出，并将期间值设置为 38d。

在转换开始后的前 400ns 内，此特定转换器不会受到接地参考细微变化的影响。 作为转换中的常规要求，IC 中应该不能流过开关电流，该电流可能会损坏转换器的接地参考平面。

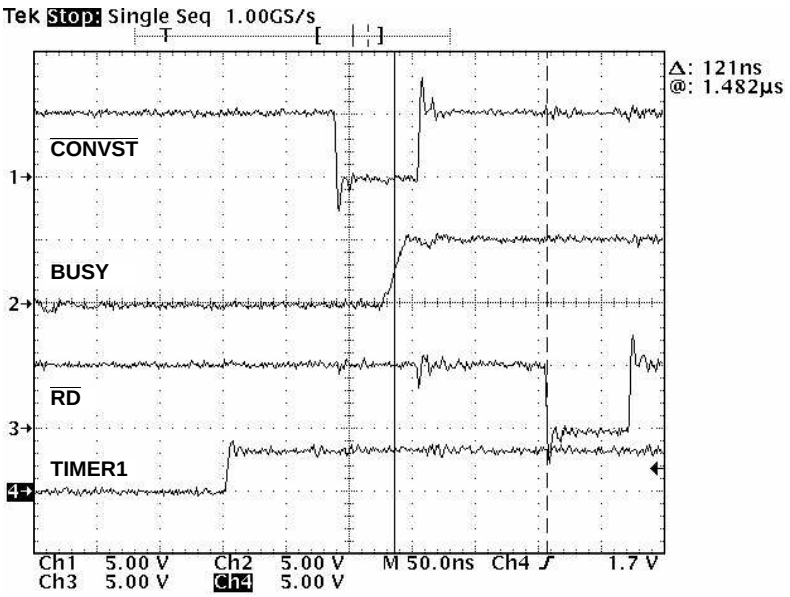

图 4. 读取脉冲延迟

ADS8371 要求在 CONVST 开始下降沿之前有一个 125ns 的静止区。 同时要求可以在转换开始后的 400ns 中进行所有 I/O 操作。 一旦受到触发，EDMA 需要花费 120ns 至 132ns 的时间生成脉冲（请参阅图 4）。 通过在 BUSY 的上升沿触发 EDMA 读取转换器通道，可以确保符合这些定时限制。 不幸的是，此种在转换周期中读取转换器的方案将导致第一个数据点无效。

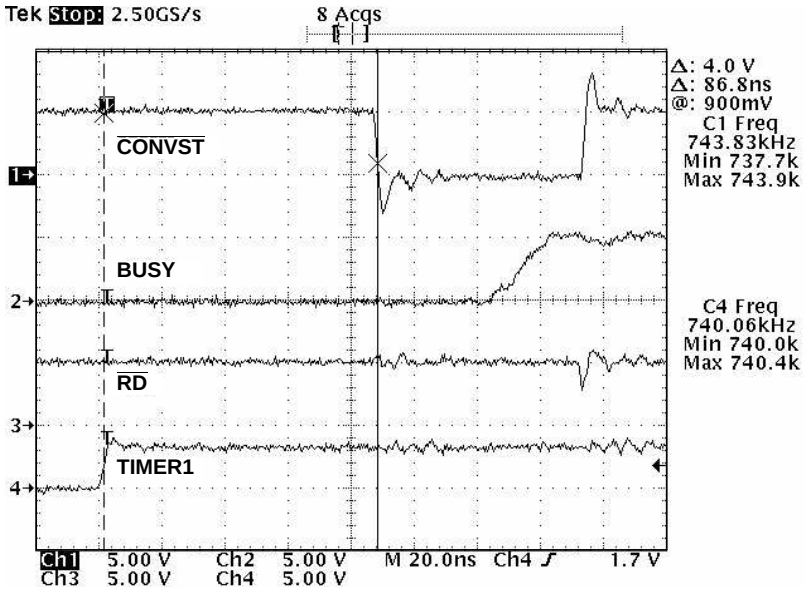

图 5. 转换启动抖动

本接口方案的固有缺陷是转换启动脉冲上具有 10ns 到 12ns 的抖动，而这是由 EDMA 产生的。 图 5 显示了 Timer1 的频率为 740kHz，但是转换启动脉冲频率呈周期性变化，始终介于 738kHz 至 744kHz 之间。 在不接收此抖动的应用中，用户可以选择将定时器输出端直接短接至转换器的 CONVST 引脚。 为反向脉冲模式（两个时钟宽）对 Timer1 进行重新编程。

要更改定时器频率，请打开 Code Composer Studio 中的文件 *Config.cdb*，并将其展开，如图 3 所示。 右击 timerCfg1，选择“属性”并选择“计数器控制”选项卡。

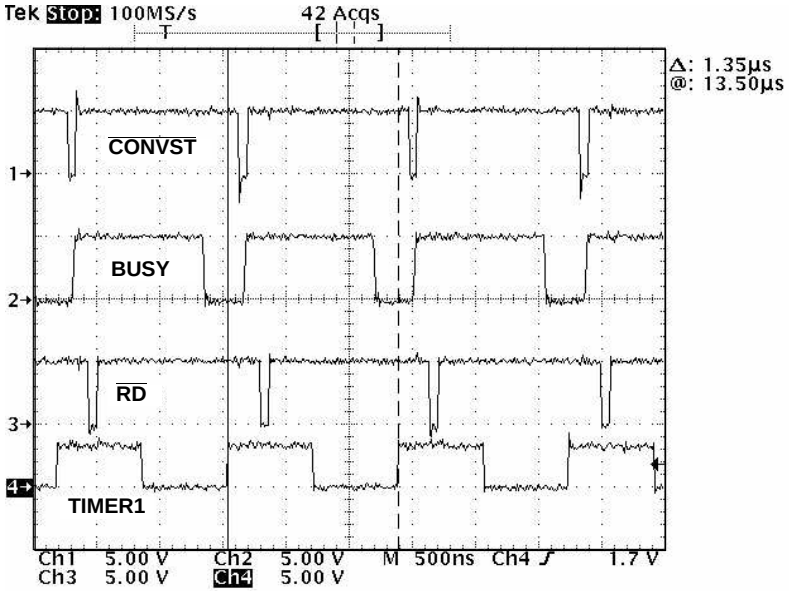

图 6. 完整周期定时

3 总结

此应用报告提供了连接 ADS8371 转换器与 C6713 DSP 的解决方案。 ADS8371EVM 插入 5-6K 接口板，接口板插入 C6713 DSK。 此报告中使用的所有硬件可向德州仪器 (TI) 订购。 软件解决方案将使用 DSP/BIOS 和配置工具（config.cdb 文件）以直观地设置 EDMA、定时器和中断。 EDMA 和 Timer1 用于在 740kHz 时触发转换周期。 使用 EDMA 的连接特性实施交替缓冲方案。 在此特定应用中，EDMA 将在收集 4096 个采样后暂停。 用户可以通过更改程序实现两个缓冲间的持续交替，方法为将 edmaCfgChan7A 中的连接句柄更改为 hEdmaTbl7，而不是 hEdmaTbl0。

4 参考

1. *TMS320C621x/TMS320C671x EDMA 结构* (SPRA996)
2. *TMS320C6000 增强型 DMA: 示例应用* (SPRA636)
3. *TMS320C6000 DSP 增强型直接存储器存取 (EDMA) 控制器参考指南* (SPRU234)
4. *ADS8371 - 带并行接口的 16 位、单极输入、微功率采样模数转换器数据表* (SLAS390)
5. *ADS8411 - 带并行接口和参考的 16 位、2MSPS、单极输入、微功率采样模数转换器数据表* (SLAS369)
6. *TMS320C6713 - 浮点数字处理器、数据表* (SPRS186)
7. *ADS8371/EVM 用户指南* (SLAU137)
8. *5-6 K 接口板 EVM 用户指南* (SLAU104)

附录 A — MAIN.C

```
/*********************************************************************/
/* File: main.c                                                      */
/* Description: Program for interfacing ADS8371                      */
/* to a 'C6713 DSK.  Program uses the Timer1 to trigger EDMA a       */
/* transfer which causes a convst# pulse, at about 740 kHz.          */
/* The BUSY signal is used to trigger EDMA channel 7 to              */
/* read from the A/D and store data into pingBuf[] and pongBuf[] data*/
/* arrays.                                                           */
/* Once 2048 of data is captured, the EDMA will interrupt CPU.       */
/* CPU will then halt EDMA channels and timer1 after the second time.*/
/* The program collects 4096 samples. EDMA triggers of the rising    */
/* edge of BUSY.                                                     */
/* AD converter address: CE2 memory space                            */
/*                       0xA0004000 (RD#)                            */
/*                       0xA000C000 (CONVST#)                        */
/* Hardware Connections:                                             */
/*   CS# => Shorted to Ground                                        */
/*   RD# => Generated from 3-8 decoder mapped to 0xA0004000          */
/*   CONVST# => Generated from 3-8  decoder mapped at 0xA000C000     */
/*   BUSY => EXTERNAL INT7                                           */
/* AUTHOR  : DAP Application Group, L. Philipose, Dallas             */
/*    CREATED 2004(C) BY TEXAS INSTRUMENTS INCORPORATED.             */
/* VERSION: 1.0                                                      */
/*********************************************************************/

/* Include Header File  */
#include "Configcfg.h"
#include "dc_conf.h"

/* function prototypes */
void init_dsk(void);
void init_adc();

/* Create the buffers. We want to align the buffers to be cache */
/* friendly by aligning them on an L2 cache line boundary.      */
#pragma DATA_ALIGN(pingBuf,BLOCK_SZ);
unsigned short pingBuf[BLOCK_SZ];  /*ping buffer*/
#pragma DATA_ALIGN(pongBuf,BLOCK_SZ);
unsigned short pongBuf[BLOCK_SZ];  /*pong buffer*/
unsigned short temp, n=0;

void main(void)
{
     int i;

    /* initialize the EMIF and A/D*/
    init_dsk();
    init_adc();
    /*Initialize data buffers     */
    for (i=0; i<=BLOCK_SZ; i++) {
        pingBuf[i]=0x0000;
        pongBuf[i]=0x0000;
    }

/* Enable the EDMA controller interrupt */
    IRQ_reset(IRQ_EVT_EDMAINT);             /*Reset EDMA interrupt       */
    IRQ_enable(IRQ_EVT_EDMAINT);
    EDMA_intDisable(TCCINTNUM6);            /*Disable EDMA interrupt     */
    EDMA_intClear(TCCINTNUM6);              /*Clear EDMA interrupt       */
    EDMA_intEnable(TCCINTNUM6);             /*Enable EDMA interrupt      */


/*Configure EDMA Channels, clear and enable them */
    EDMA_config(hEdmaCha7,&edmaCfgChan7);
    EDMA_config(hEdmaCha2,&edmaCfgChan2);
    EDMA_clearChannel(hEdmaCha2);
    EDMA_clearChannel(hEdmaCha7);
```

```
    EDMA_enableChannel(hEdmaCha2);        /*Enable EMDA channel 2 -CONVST#*/
    EDMA_enableChannel(hEdmaCha7);        /*Enable EDMA channel 7 -RD#    */

/*Timer1 was initialized by DSP/BIOS before entering main.c */
/*Only need to enable it here. */
    TIMER_start(hTimer1);                 /*Start A/D CONVST# trigger     */
/*Go sample at 740kSPS*/
}
/*****************************************************************/
/* end main.c                                                    */
/*****************************************************************/
```

附录 B Functions.C

```
/*****************************************************************/
/* File: functions.c                                             */
/*    Description: Functions for interfacing ADS8371 to C6713    */
/*  init_dsk(), init_adc(), hwiDMA_isr(), swiEnablePrephFunc()   */
/* AD converter address: CE2 memory space                        */
/*                       0xA0004000 (RD#)                        */
/*                       0xA000C000 (CONVST#)                    */
/* Hardware Connections:                                         */
/*    CS# => Grounded                                            */
/* RD# => Generated from 3-8 decoder mapped to 0xA0004000        */
/* CONVST# => Generated from 3-8  decoder mapped at 0xA000C000   */
/* BUSY => EXTERNAL INT7                                         */
/* AUTHOR : DAP Application Group, L. Philipose, Dallas          */
/*    CREATED 2004(C) BY TEXAS INSTRUMENTS INCORPORATED.         */
/* VERSION: 1.0                                                  */
/*****************************************************************/

/*  Include Header File  */
#include "configcfg.h"
#include <csl_legacy.h>
#include "dc_conf.h"

/*Function Prototypes*/
void swiEnablePrephFunc();

extern unsigned short  temp;  /* Raw buffer for AD data          */
int pingpong=0; /*=1 pingbuffers full                           */

/*****************************************************************/
/* init_dsk()                                                    */
/* This initializes the EMIF                                     */
/*****************************************************************/
void init_dsk(void)
{

UINT32 gblctl,ce0ctl,ce1ctl,ce2ctl,ce3ctl,sdctl,sdtim,sdext;

/* intialization of the EMIF */

/* RBTR8,SSCRT,CLK2EN,CLK1EN,SSCEN,SDCEN,NOHOLD       */
    gblctl = EMIF_MK_GBLCTL( 0, 0, 1, 0, 0, 0, 0);

/* RDHLD,MTYPE,RDSTRB,TA,RDSETUP,WRHLD,WRSTRB,WRSETUP */
    ce0ctl = EMIF_MK_CECTL( 0, 3, 0, 0, 0, 0, 0, 0);

/* RDHLD,MTYPE,RDSTRB,TA,RDSETUP,WRHLD,WRSTRB,WRSETUP */
    ce1ctl = EMIF_MK_CECTL( 0, 2, 0, 0, 0, 0, 0, 0);

/* RDHLD,MTYPE,RDSTRB,TA,RDSETUP,WRHLD,WRSTRB,WRSETUP */
    ce3ctl = EMIF_MK_CECTL( 0, 2, 0, 0, 0, 0, 0, 0);

/* TRC,TRP,TRCD,INIT,RFEN,SDWID,SDCSZ,SDRSZ,SDBSZ     */
    sdctl = EMIF_MK_SDCTL( 7, 1, 1, 1, 1, 0, 1, 0, 0);

/* PERIOD,XRFR */
    sdtim = EMIF_MK_SDTIM(   1562,   0);

sdext = EMIF_SDEXT_NA;

/* make CE2 control register value                    */
    /* This is the CE space used by the ADS8371.      */
    /* Use the timing values from dc_conf.h:              */

   ce2ctl = EMIF_MK_CECTL(
        EMIF_CECTL_RDHLD_OF   (RDHLD),     /* read  hold   */
        EMIF_CECTL_MTYPE_ASYNC32,
        EMIF_CECTL_RDSTRB_OF  (RDSTRB),    /* read  strobe */
        EMIF_CECTL_TA_NA,
        EMIF_CECTL_RDSETUP_OF (RDSETUP),   /* read  setup  */
        EMIF_CECTL_WRHLD_OF   (WRHLD),     /* write hold   */
        EMIF_CECTL_WRSTRB_OF  (WRSTRB),    /* write strobe */
        EMIF_CECTL_WRSETUP_OF (WRSETUP)    /* write setup  */
```

```
    );
/* configure the EMIF */
    EMIF_ConfigB(gblctl,ce0ctl,ce1ctl,ce2ctl,
        ce3ctl,sdctl,sdtim,sdext);
    return;
} /* end init_dsk() */
/*************************************************************/
/* init_adc()                                                */
/* This initializes the ADC                                  */
/*************************************************************/
void init_adc()
{ int i;
  /*Initialize A/D */
      temp = *ADS8371_CONVSTZ;       /*Discard the first three conversions*/
      for (i=0; i<=22; i++){};    /*after power up.*/
      temp = *ADS8371_RD;
      for (i=0; i<=1; i++){};
      temp = *ADS8371_CONVSTZ;
      for (i=0; i<=21; i++){};
      temp = *ADS8371_RD;
      for (i=0; i<=1; i++){};
      temp = *ADS8371_CONVSTZ;
      for (i=0; i<=21; i++){};
      temp = *ADS8371_RD;
}
/*************************************************************/
/*hwiDMA_isr():                                              */
/* Hardware Interrupt Function disables EDMA channels       */
/* and Timer1, Then post software interrupt.                 */
/*************************************************************/
void hwiDMA_isr()
{
   if (pingpong==1){
      TIMER_pause(hTimer1);
      EDMA_disableChannel(hEdmaCha2);   /*Disable EMDA channel 2-CONVST#*/
      EDMA_disableChannel(hEdmaCha7);   /*Disable EMDA channel 7 -RD#*/
//Uncomment next line to go again
//    swiEnablePrephFunc();
    }
    pingpong=!pingpong;
    IRQ_reset(IRQ_EVT_EDMAINT);        /*Reset EDMA interrupt    */
    IRQ_enable(IRQ_EVT_EDMAINT);
    EDMA_intDisable(TCCINTNUM6);       /*Disable EDMA interrupt  */
    EDMA_intClear(TCCINTNUM6);         /*Clear EDMA interrupt    */
    EDMA_intEnable(TCCINTNUM6);        /*Enable EDMA interrupt   */
}
/*************************************************************/
/*swiEnablePrephFunc:                                        */
/*     Software Interrupt Function configures EDMAC2 and    */
/*     EDMAC7, enables respective EDMA channels and Timer1  */
/*************************************************************/
void swiEnablePrephFunc()
{
  IRQ_enable(IRQ_EVT_EDMAINT);
  EDMA_intDisable(TCCINTNUM6);         /*Disable EDMA interrupt   */
  EDMA_intClear(TCCINTNUM6);           /*Clear EDMA interrupt     */
  EDMA_intEnable(TCCINTNUM6);          /*Enable EDMA interrupt    */
  EDMA_config(hEdmaCha7,&edmaCfgChan7);
  EDMA_config(hEdmaCha2,&edmaCfgChan2);
  EDMA_enableChannel(hEdmaCha2);       /*Enable EMDA channel 2 -CONVST# */
  EDMA_enableChannel(hEdmaCha7);       /*Enable EDMA channel 7 -RD# */
  EDMA_clearChannel(hEdmaCha7);
  EDMA_clearChannel(hEdmaCha2);
```

```
  TIMER_start(hTimer1);               /*Start A/D CONVST# trigger  */
}
/*****************************************************************/
/* End Functions.c                                               */
/*****************************************************************/
```

重要声明

德州仪器 (TI) 及其下属子公司有权在不事先通知的情况下，随时对所提供的产品和服务进行更正、修改、增强、改进或其它更改，并有权随时中止提供任何产品和服务。 客户在下订单前应获取最新的相关信息，并验证这些信息是否完整且是最新的。 所有产品的销售都遵循在订单确认时所提供的 TI 销售条款与条件。

TI 保证其所销售的硬件产品的性能符合 TI 标准保修的适用规范。 仅在 TI 保修的范围内，且 TI 认为有必要时才会使用测试或其它质量控制技术。 除非政府做出了硬性规定，否则没有必要对每种产品的所有参数进行测试。

TI 对应用帮助或客户产品设计不承担任何义务。 客户应对其使用 TI 组件的产品和应用自行负责。 为尽量减小与客户产品和应用相关的风险，客户应提供充分的设计与操作安全措施。

TI 不对任何 TI 专利权、版权、屏蔽作品权或其它与使用了 TI 产品或服务的组合设备、机器、流程相关的 TI 知识产权中授予的直接或隐含权限作出任何保证或解释。 TI 所发布的与第三方产品或服务有关的信息，不能构成从 TI 获得使用这些产品或服务的许可、授权、或认可。 使用此类信息可能需要获得第三方的专利权或其它知识产权方面的许可，或是 TI 的专利权或其它知识产权方面的许可。

对于 TI 的数据手册或数据表，仅在没有对内容进行任何篡改且带有相关授权、条件、限制和声明的情况下才允许进行复制。 在复制信息的过程中对内容的篡改属于非法的、欺诈性商业行为。 TI 对此类篡改过的文件不承担任何责任。

在转售 TI 产品或服务时，如果存在对产品或服务参数的虚假陈述，则会失去相关 TI 产品或服务的明示或暗示授权，且这是非法的、欺诈性商业行为。 TI 对此类虚假陈述不承担任何责任。

可访问以下 URL 地址以获取有关其它 TI 产品和应用解决方案的信息：

产品	
放大器	http://www.ti.com.cn/amplifiers
数据转换器	http://www.ti.com.cn/dataconverters
DSP	http://www.ti.com.cn/dsp
接口	http://www.ti.com.cn/interface
逻辑	http://www.ti.com.cn/logic
电源管理	http://www.ti.com.cn/power
微控制器	http://www.ti.com.cn/microcontrollers
应用	
音频	http://www.ti.com.cn/audio
汽车	http://www.ti.com.cn/automotive
宽带	http://www.ti.com.cn/broadband
数字控制	http://www.ti.com.cn/control
光纤网络	http://www.ti.com.cn/opticalnetwork
安全	http://www.ti.com.cn/security
电话	http://www.ti.com.cn/telecom
视频与成像	http://www.ti.com.cn/video
无线	http://www.ti.com.cn/wireless

邮寄地址：Texas Instruments, Post Office Box 655303, Dallas, Texas 75265