

请注意赛普拉斯已正式并入英飞凌科技公司。

此封面页之后的文件标注有“赛普拉斯”的文件即该产品为此公司最初开发的。请注意作为英飞凌产品组合的部分,英飞凌将继续为新的及现有客户提供该产品。

文件内容的连续性

事实是英飞凌提供如下产品作为英飞凌产品组合的部分不会带来对于此文件的任何变更。未来的变更将在恰当的时候发生,且任何变更将在历史页面记录。

订购零件编号的连续性

英飞凌继续支持现有零件编号的使用。下单时请继续使用数据表中的订购零件编号。

PSoC®3 和 PSoC 5LP — DMA 入门手册

作者: **Anu M D、Lakshmi Natarajan**

相关项目: 有

相关器件系列: 所有 **PSoC® 3** 和 **PSoC 5LP** 器件

软件版本: **PSoC® Creator™ 3.0 SP1** 以及更高版本

相关应用笔记: **AN61102、AN84810**

AN52705 提供了 PSoC® 3 和 PSoC 5LP 上的直接存储器访问 (DMA) 模块简介。PSoC DMA 可以在片上外设和存储器之间传输数据, 无需 CPU 的干预。该应用笔记说明了如何使用示例项目为简单数据传输配置 DMA, 包括外设到存储器、存储器到外设、各外设间以及各存储器间的数据传输。

目录

1 简介	1	10 示例 5: TD 链路	16
2 DMA 的基本概念	2	10.1 示例 5: DMA 配置	17
3 DMA 配置	3	10.2 示例 5: 项目文件	18
3.1 通道配置	3	11 总结	18
3.2 TD 配置	4	12 关于作者	18
4 DMA 组件概述	5	附录 A. DMA 配置步骤	19
4.1 DMA 组件的硬件连接	5	A.1 其他重要的 DMA API 函数	21
5 DMA 的固件配置	6	附录 B. DMA 向导配置	22
6 示例 1: 外设至外设的传输	7	附录 C. 设置 DMA 通道的优先级	25
6.1 示例 1: DMA 配置	8	附录 D. 示例项目 — 测试设置	26
6.2 示例 1: 项目文件	8	D.1 示例 1: 外设至外设的传输 —	
6.3 示例 1: DMA 配置代码	9	Eg1_ADC_DMA_DAC	26
7 示例 2: 外设至存储器的传输	10	D.2 示例 2: 外设至存储器的传输 —	
7.1 示例 2: DMA 配置	11	Eg2_ADC_DMA_Mem	26
7.2 示例 2: 项目文件	12	D.3 示例 3: 存储器至外设的传输 —	
8 示例 3: 存储器至外设的传输	12	Eg3_Mem_DMA_DAC	27
8.1 示例 3: DMA 配置	13	D.4 示例 4: 存储器至存储器的传输 —	
8.2 示例 3: 项目文件	14	Eg4_Mem_DMA_Mem	27
9 示例 4: 存储器至存储器的传输	14	D.5 示例 5: TD 链路 — Eg5_TD_Chaining	28
9.1 示例 4: DMA 配置	15	附录 E. 常见问题解答	30
9.2 示例 4: 项目文件	16	文档修订记录	31
		全球销售和设计支持	32

1 简介

PSoC 3 和 PSoC 5LP 的 DMA 控制器 (DMAC) 可以将数据从源地址传输到目标地址, 并且无需 CPU 的干预。因此在 DMA 传输数据时, CPU 仍能够处理其他任务, 从而获得“多处理”环境。

PSoC DMA 控制器 (DMAC) 非常灵活 — 它可以在存储器和片上外设 (包括 ADC、DAC、滤波器、USB、UART 和 SPI) 间无缝传输数据。拥有 24 个独立的 DMA 通道。

该应用笔记描述了如何为简单数据传输配置 DMA。它包括显示下面各种不同类型 DMA 传输的项目:

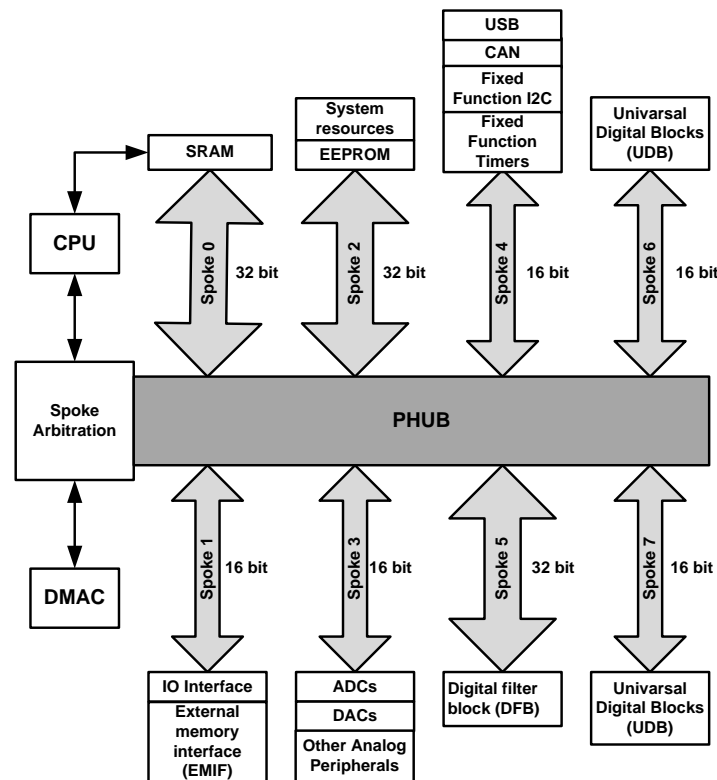
- 外设到存储器
- 存储器到外设
- 各外设之间
- 各存储器之间

本应用笔记假定您已经熟悉使用 PSoC 3 或 PSoC 5LP 的 PSoC Creator 进行应用开发。如果您刚刚接触 PSoC 3 或 PSoC 5LP，则可以通过 [AN54181 — PSoC 3 入门手册](#) 和 [AN77759 — PSoC 5 入门手册](#) 加深对该产品的了解。有关高级 DMA 主题，请参考 [AN84810 应用笔记](#)。如果您尚未了解 PSoC Creator，请参考 [PSoC Creator 主页](#)。

2 DMA 的基本概念

PSoC 3 和 PSoC 5LP 上的 DMAC 是连接片上外设的中央集线器（被称为外设集线器（PHUB））的一部分，如图 1 所示。DMAC 是 PHUB 的两个总线主器件中的一个。

图 1. 外设集线器



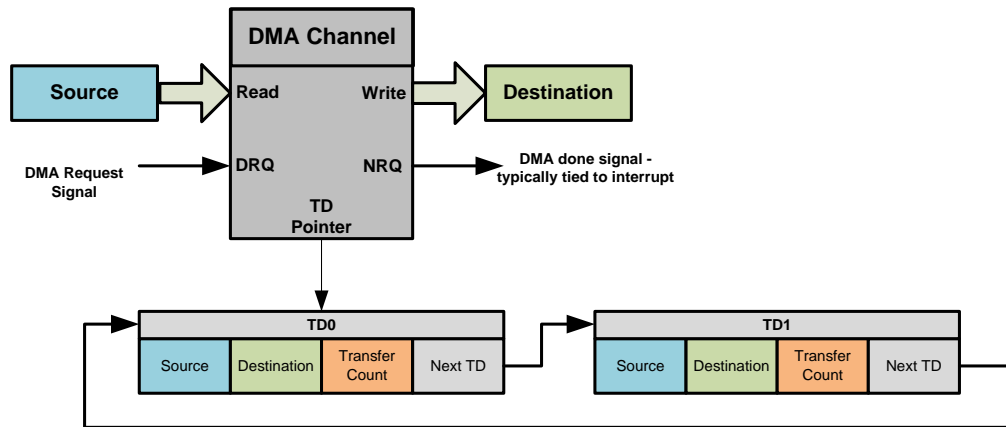
PHUB 具有八个被称为轮辐的数据总线。每个轮辐将 CPU 和 DMAC 连接到一个或多个外设。轮辐的数据宽度是 16 位或 32 位。连接到轮辐的外设有 8 位、16 位或 32 位的数据宽度。

外设的数据宽度通常不大于外设连接到的轮辐的数据宽度。如果外设的数据宽度大于所连接的轮辐的数据宽度，则 PHUB 会以轮辐的宽度与外设进行数据传输。

PHUB 有两个总线主器件，即 CPU 和 DMAC。CPU 和 DMAC 可以同时访问不同的 PHUB 轮辐。如果 CPU 和 DMAC 同时尝试对同一个轮辐进行访问，将发生总线仲裁。有关详细信息，请参见 [PSoC 3 和 PSoC 5LP 技术参考手册](#)。

24 个 DMA 通道中的每一个都可以独立进行数据传输。每个通道都具有一个数据操作描述符（TD）链，如图 2 所示。TD 包括以下信息：源地址、目标地址、传输次数以及链中的下一个 TD。最多有 128 个 TD。通道和 TD 的组合描述了完整的 DMA 传输。

图 2. DMA 通道

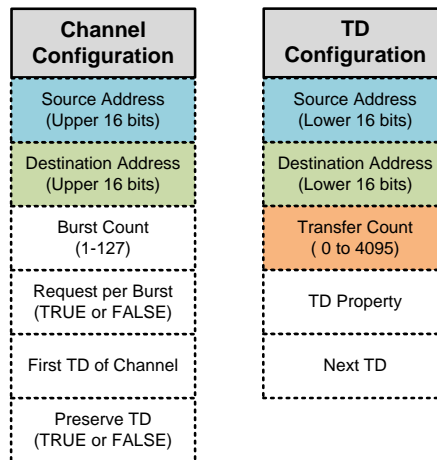


每个 DMA 通道都有一个单独的 DMA 请求输入，用于启动事务。一个 DMA 请求可以由 CPU 或外设启动。当收到 DMA 请求时，DMAC 将访问连接到源和目标地址的轮辐，并传输数据（如通道和有关 TD 的配置）。

3 DMA 配置

通过使用通道和 TD 配置寄存器对 DMA 传输进行配置。图 3 显示了通道和 TD 的配置参数。

图 3. DMA 配置



3.1 通道配置

下述内容解释了 DMA 通道配置参数：

- **源地址的高 16 位**
在通道配置寄存器中对 32 位源地址的高 16 位进行配置。
- **目标地址的高 16 位**
在该通道配置寄存器中对 32 位目标地址的高 16 位进行配置。
- **突发传输计数（1 至 127）**
定义 DMA 通道在释放轮辐之前必须从源地址传输到目标地址的字节数。DMAC 获取轮辐，并将指定的字节数从源地址传输到目标地址，然后释放轮辐。在下一个突发传输中，它会再次获取轮辐。
内部轮辐 DMA 传输的突发计数要小于或等于 16。

■ 每次突发请求（0 或 1）

完成 DMA 数据传输需要进行多次突发传输时，该位用于指定突发的性质。

0: 表示第一次突发后的所有后续突发可自动完成，不需要单独请求 DMA。只要求第一次突发传输具有 DMA 请求。

1: 每次突发传输都需要一个单独的请求。

■ 初始 TD

定义与通道相关联的第一个 TD。第一个 TD 的指针被存储在通道配置存储器中。后续 TD 指针被存储在 TD 配置存储器中，如链表所示。

■ 保留 TD（0 或 1）

指定是否保存原始的 TD 配置，以供后续 DMA 传输重新使用。保留典型的 TD 配置。

0: 不保留 TD 配置。

1: 保留 TD 配置。

如果 TD 被保留，则通道使用一个单独 TD 存储器（与通道编号相应）以跟踪正在运行的数据操作；否则将原始 TD 配置寄存器作为工作寄存器使用，以跟踪正在运行的数据操作。

3.2 TD 配置

以下内容显示的是 TD 配置参数：

■ 源地址的低 16 位

在 TD 配置寄存器上对源地址的低 16 位进行配置

■ 目标地址的低 16 位：

32 位目标地址的低 16 位

■ 传输计数（0 至 4095 个字节）

从源地址到目标地址传输的字节总数量。

传输计数与突发计数参数一起使用。比如，如果您要将 50 个大小为 2 字节的数据字从 16 位外设传输到存储器缓冲区，突发计数和传输计数分别被设置为 2 和 100。

■ 下一个 TD

下一个 TD，如链表所示。

■ TD 属性

表 1 显示的是由 TD 属性配置寄存器中的位字段定义的 TD 属性。

表 1. TD 的属性

属性	描述
递增源地址	如果设置该位，进行 DMA 数据操作时，将递增源地址。
递增目标地址	如果设置该位，进行 DMA 数据操作时，将递增目标地址。
交换使能	如果设置该位，DMA 将数据从源地址传输到目标地址时，它会交换数据字节。
交换大小	如果交换使能位被设置，该位用于指定交换操作的大小。 0: 在 DMA 传输过程中，对每两个字节执行字节序交换。 1: 在 DMA 传输过程中，对每四个字节执行字节序交换。
自动执行下一个 TD	0: 仅在执行下一个 DMA 请求后，才会执行链中的下一个 TD。 1: 当前一个 TD 传输完成后，会自动执行链中的下一个 TD。
DMA 完成事件	如果设置该位，则数据传输完成后，将生成一个 DMA “完成信号”。数据传输完成后，通常通过该属性创建一个中断。
使能 TD 终止	如果设置，通过使用硬件信号可以终止正在运行的数据操作。

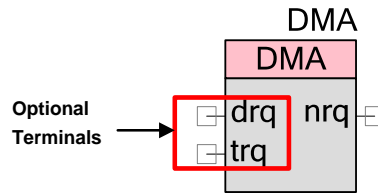
PSoC 3 Keil 编译器使用高位优先格式存储 16 位和 32 位变量。但 PSoC 3 外设寄存器和存储器使用低位优先格式。因此，当 DMA 将多字节数据在 PSoC 3 的外设寄存器和存储器之间进行传输时，必须配置 DMA 以执行字节交换。对于 PSoC 5LP，不用这样做，因为外设和存储器使用同一个字节序。

现在让我们了解如何使用 PSoC Creator 配置 DMA。

4 DMA 组件概述

图 4 显示了 PSoC Creator 的 DMA 通道组件。可以在 Component（组件）目录中的 Systems（系统）选项卡中找到该组件。

图 4. DMA 通道组件

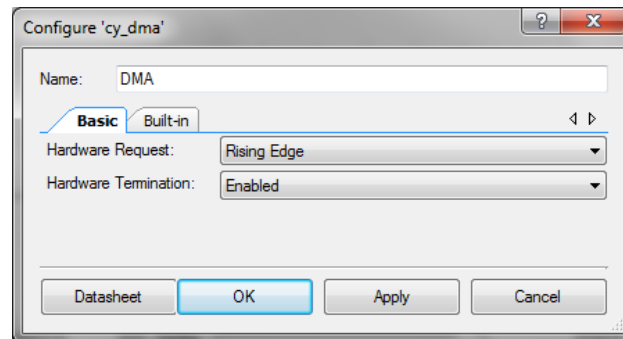


使用 DMA 通道组件和一个相关的 API 进行配置 DMA 以传输数据。

4.1 DMA 组件的硬件连接

您可以在组件配置窗口上设置以下参数，如图 5 显示。

图 5. DMA 组件配置



硬件请求 (drq)：该设置可定义用于触发 DMA 通道的信号类型（上升沿/电平）。除“禁用”外，该参数的其他选项会将 drq 终端添加到组件内。可以将 drq 连接至任何硬件信号，以触发 DMA 通道。

如果不存在 drq 终端，那么只有 CPU 会触发 DMA 数据操作。

该参数被设置为“派生 (derived)”时，DMA 触发类型（边沿/电平）由 DMA 触发源决定。更多信息，请参考 [DMA 组件数据手册](#)。

硬件终端 (trq)：当该选项被设置为真 (true) 时，其他输入终端 (trq) 将显示在组件中。如果使能了 TD 终止，则该终端上的上升沿将停止正在运行的 DMA 数据操作。请注意：仅当一个 DMA 突发数据操作正在运行时，trq 才会终止 TD 链。有关详细信息，请参考“组件数据手册”中的内容。

传输完成 (nrq)：为了指出 DMA 传输已经完成，当完成传输时，可以配置 TD 以便在 DMA 通道的 NRQ 终端上创建一个宽度为 2 个总线时钟的脉冲。可以将 nrq 终端连接至一个中断或者其他组件以便执行其他操作。

设置 TD 属性以指定是否在 nrq 终端上生成一个信号，以及是否用 trq 来允许 TD 终止。

5 DMA 的固件配置

在编译项目过程中，DMA 组件为每个 DMA 实例生成一个源文件和相应的头文件。例如，如果在您的设计中 DMA 组件实例的名称为“DMA_1”，则在编译过程中，将创建 *DMA_1_dma.c* 和 *DMA_1_dma.h* 文件。这些文件包含了用于初始化 DMA 通道的“DmaInitialize” API。Generated Source 文件夹中的 *CyDmac.c* 和 *CyDmac.h* 文件包含了其他通道和 TD 配置函数。

按照下述步骤进行配置 DMA 的固件：

1. 启动 DMA 通道

```
Channel_Handle = DMA_DmaInitialize(DMA_BYTES_PER_BURST, DMA_REQUEST_PER_BURST,  
                                   HI16(Source Address), HI16(Destination  
                                   Address))
```

2. 创建 TD 的一个实例

```
TD_Handle = CyDmaTdAllocate();
```

3. 设置 TD 配置

```
CyDmaTdSetConfiguration(TD_Handle, Transfer_Count, Next_TD, TD_Property);
```

4. 设置 TD 地址

```
CyDmaTdSetAddress(TD_Handle, LO16(Source Address), LO16(Destination Address))
```

5. 设置通道的初始 TD。

```
CyDmaChSetInitialTd(Channel_Handle, TD_Handle)
```

6. 启用 DMA 通道

```
CyDmaChEnable(Channel_Handle, preserve_TD)
```

有关固件配置步骤的详细信息，请参考第 19 页上的 [DMA 配置步骤](#)。可以使用 DMA 向导自动生成代码，以配置 DMA 通道；有关详细信息，请参考第 22 页上的 [DMA 向导配置](#)。

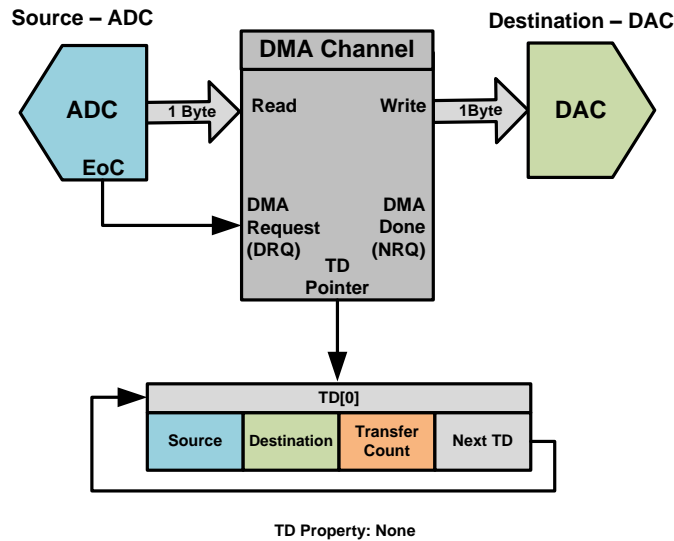
请注意：DMA 向导仅在有限的 PSoC 外设集中支持 DMA 数据操作。如果 DMA 向导不支持外设，您需要使用附录 A 所详细描述的函数来手动配置 DMA。

下面是一组（四个）示例，通过它们详细描述了如何在存储器和外设之间进行 DMA 传输。第五个示例介绍的是如何构建多 TD 链。

6 示例 1：外设至外设的传输

该示例介绍了如何使用 DMA 在各外设之间进行简单传输，具体内容是从 ADC 数据输出寄存器传输到 DAC 数据输入寄存器，如图 6 显示。

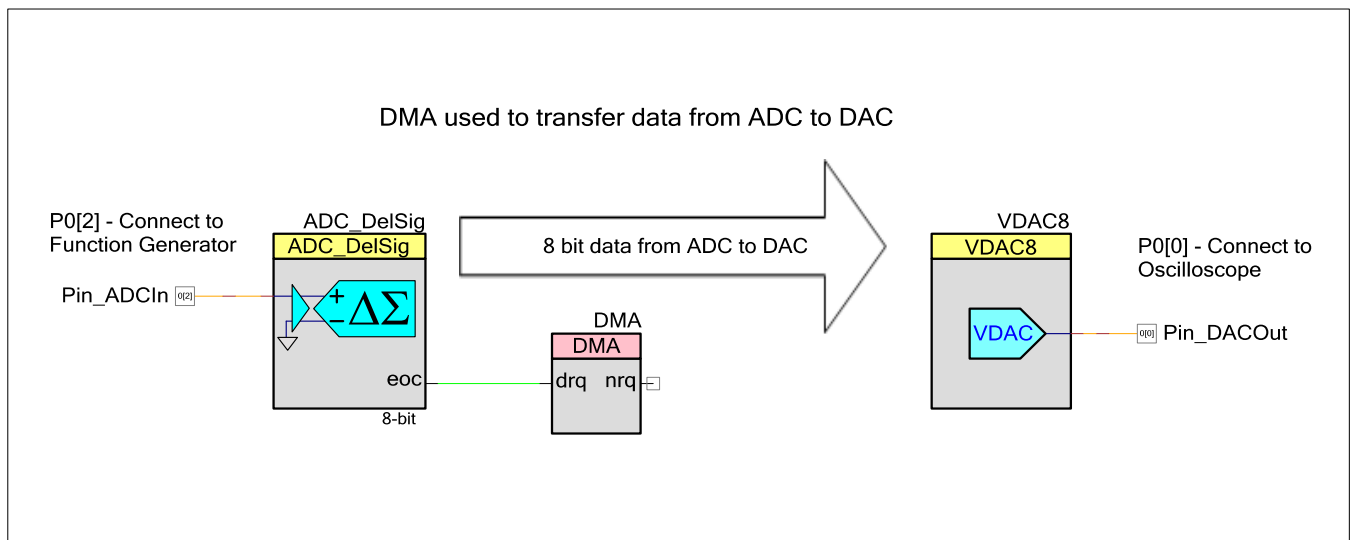
图 6. 外设至外设传输的框图



如图 7 显示，配置 ADC 使其在 8 位单端模式下运行，以符合 VDAC 的数据格式（其格式为单端 8 位电压 DAC）的要求。DMA 通道的硬件请求（DRQ）被使能，并连接至 ADC EoC 信号，以便在 ADC 结果可用时，ADC 可以发出数据传输的请求。

收到请求后，DMA 通道从 ADC 输出寄存器读取一个数据字节，并将其写入到 DAC 数据寄存器内。

图 7. 外设至外设传输的顶层设计



6.1 示例 1: DMA 配置

表 2 和表 4 分别显示的是关于该项目的 DMA 通道和 TD 配置。

表 2. 通道配置设置

参数	项目设置
高位源地址	HI16 (CYDEV_PERIPH_BASE)
高位目标地址	HI16 (CYDEV_PERIPH_BASE)
突发计数	1 (一个字节)
每次突发请求	1 (True)
初始 TD	TD[0]
保留 TD	1 (True)

对于源地址和目标地址，通道配置都具有 32 位地址中的高 16 位。

CYDEV_PERIPH_BASE (属于 PSoC Creator 自动生成的 *cydevice.h* 文件) 确定了所有 PSoC 外设的基本地址，包括 ADC 和 DAC。

HI16 是一个 PSoC Creator 宏，它会返回 32 位数值的高 16 位。使用该宏可获取源地址和目标地址的高 16 位数值。

作为一种替代方法，您可以分配有关组件寄存器的高位源地址和高位目标地址，如表 3 所示。可以在 ADC_DelSig.h 和 VDAC8.h 组件文件中分别找到地址定义。

表 3. 代替高地址

参数	项目设置
高位源地址	HI16 (ADC_Delsig_DEC_OUTSAMP_PTR)
高位目标地址	HI16 (VDAC8_DATA_PTR)

可以将 TD 视为链接 TD 的数组；这样我们仅需要一个元素数组 TD[0]。

表 4. TD[0]配置设置

参数	项目设置
低位源地址	LO16 (ADC_Delsig_DEC_OUTSAMP_PTR)
低位目标地址	LO16 (VDAC8_DATA_PTR)
传输计数	1 (一个字节)
TD 的属性	无 (0)
下一个 TD	TD[0] (回送相同的 TD)

LO16 宏返回 32 位数值的低 16 位。

对于每个 DMA 请求，DMA 通道都要传输一个字节，因此突发计数被设置为 1 字节，并将每次突发的请求设置为 True。

所执行的下一个 TD 被设置为相同的 TD (回送)，因此在每 DMA 请求中，重复同样的数据操作。保留 TD 的参数被设置为 True。

6.2 示例 1: 项目文件

AN52705.zip 文件 (本应用笔记随附的) 中的 Eg1_ADC_DMA_DAC 项目演示了该示例。请参考[示例项目 — 测试设置](#)，了解如何测试该项目。

下述内容显示的是该示例的 DMA 配置代码请参考[DMA 向导配置](#)，了解如何通过使用 DMA 向导生成该配置代码。

6.3 示例 1: DMA 配置代码

```

/* Define for DMA Configuration */
#define DMA_BYTES_PER_BURST 1
#define DMA_REQUEST_PER_BURST 1
#define DMA_SRC_BASE (CYDEV_PERIPH_BASE)
#define DMA_DST_BASE (CYDEV_PERIPH_BASE)

/* Variable declarations for the DMA channel.
 * DMA_Chan is used to store the DMA channel */
uint8 DMA_Chan;
/* DMA_TD array is used to store all of the TDs associated with the channel
 * Since there is only one TD in this example, DMA_TD array contains only one element
 */
uint8 DMA_TD[1];

/* DMA Configuration steps */

/* Step 1 */
/* DMA Initializations done for both the DMA Channels
 * Burst count = 1, (8 bit data transferred to VDAC one at a time)
 * Request per burst = 1 (transfer burst only on new request)
 * High byte of source address = Upper 16 bits of ADC data register
 * High byte of destination address = Upper bytes of the VDAC8 data register
 * DMA_Chan holds the channel handle returned by the 'DmaInitialize' function. This is
 * used for all further references of the channel */
DMA_Chan = DMA_DmaInitialize(DMA_BYTES_PER_BURST, DMA_REQUEST_PER_BURST,
                             HI16(DMA_SRC_BASE), HI16(DMA_DST_BASE));

/* Step 2 */
/* Allocate TD for DMA Channel
 * DMA_TD[0] is a variable that holds the TD handle returned by the TD allocate
 function.
 * This is used for all further references of the TD */
DMA_TD[0] = CyDmaTdAllocate();

/* Step 3 */
/* Configure TD[0]
 * Transfer count = 1 (total number of bytes to transfer from the ADC to DAC)
 * Next Td = DMA_TD[0]. The same td has to repeat itself for every ADC EoC.
 * Configuration = No special TD configurations required */
CyDmaTdSetConfiguration(DMA_TD[0], 1, DMA_TD[0], 0);

/* Step 4 */
/* Configure the td address
 * Source address = Lower 16 bits of ADC data register
 * Destination address = Lower 16 bits of VDAC8 data register */
CyDmaTdSetAddress(DMA_TD[0], LO16((uint32)ADC_Delsig_DEC_SAMP_PTR),
                  LO16((uint32)VDAC8_Data_PTR));

/* Step 5 */
/* Map the TD to the DMA Channel */
CyDmaChSetInitialTd(DMA_Chan, DMA_TD[0]);

/* Step 6 */
/* Enable the channel
 * The Channel is enabled with Preserve TD parameter set to 1. This preserves the
 * original TD configuration and reload it after the transfer is complete so that the
 TD
 * can be repeated */
CyDmaChEnable(DMA_Chan, 1);

```

7 示例 2：外设至存储器的传输

该示例显示的是如何执行从 ADC 数据输出寄存器到 16 位存储器阵列的外设至存储器的传输，如图 8 所示。

图 8. 外设至存储器传输的框图

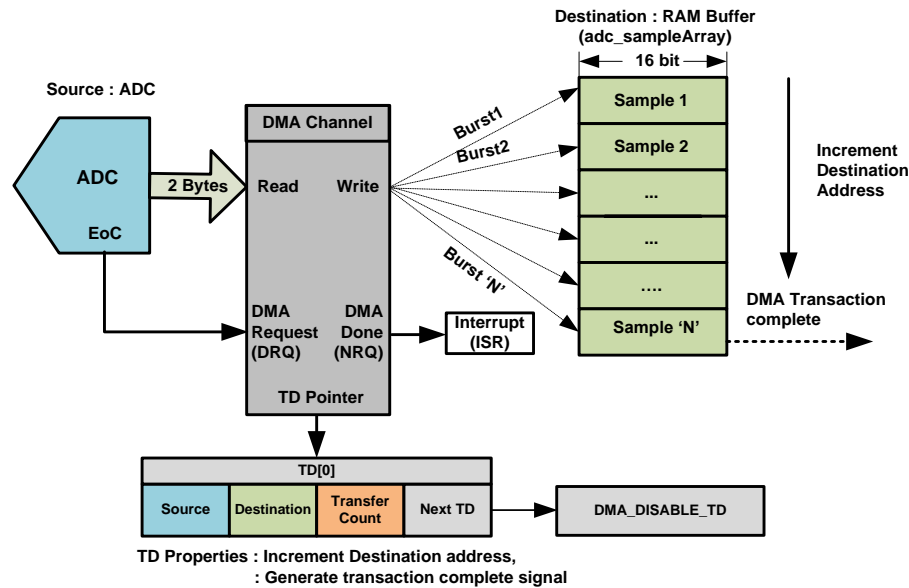
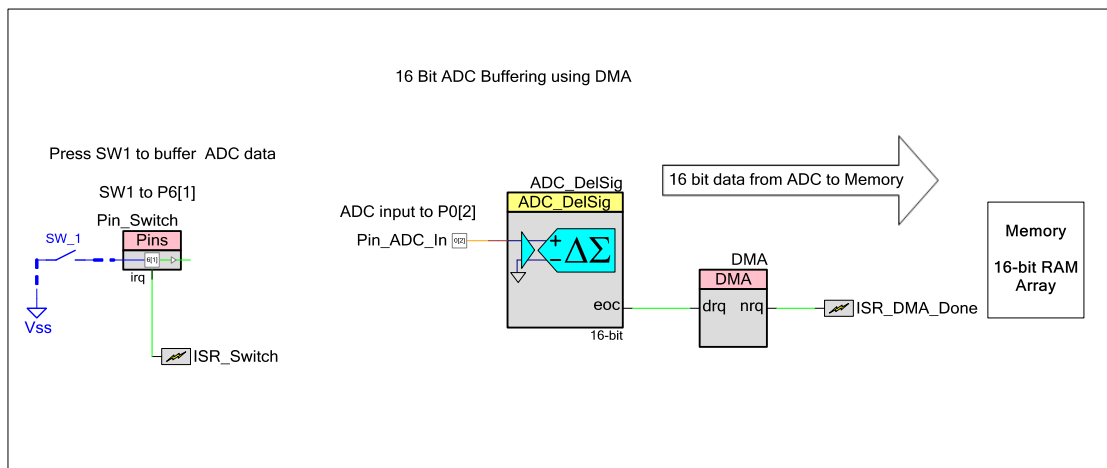


图 9 显示的是项目的顶层设计。每次按下 Pin_Switch，ISR_Switch 便被触发，并且在 isr 上设置一个标志以便使能 DMA 通道。一旦 DMA 通道被使能，ADC 的 EoC 信号将激活 DMA 通道请求。

每个 DMA 请求中，DMA 从源地址（ADC 输出寄存器）提取 2 个字节，并将其写入目标 RAM 缓冲器，并且目标地址的值增加 2。每次突发传输后，传输计数将减去 2。重复进行该操作，直到传输计数为 0，这时会在 DMA 组件的 NRQ 终端上生成“传输结束”信号（其激活 ISR_DMA_Done 中断）。

在中断服务程序中，设置一个标志以指示事务已完成。事务完成时禁用 DMA 通道，再次按下开关时会重新启用 DMA 通道。

图 9. 外设至存储器传输的顶层设计



7.1 示例 2: DMA 配置

表 5 和表 6 分别显示的是关于该项目的 DMA 通道和 TD 配置。

表 5. 通道配置设置

参数	项目设置
高位源地址	HI16 (CYDEV_PERIPH_BASE)
高位目标地址	HI16 (CYDEV_SRAM_BASE)
突发计数	2 (两个字节)
每次突发请求	1 (True)
初始 TD	TD[0]
保留 TD	1 (True)

对于源地址和目标地址，通道配置都具有 32 位地址中的高 16 位。源地址与示例 1 中所示的相同。

CYDEV_SRAM_BASE (属于 PSoC Creator 自动生成的 *cydevice.h* 文件) 指定了 SRAM 基本地址。与 HI16 宏一起使用，以指出目标地址的高 16 位数值。

作为替换方案，RAM 阵列指针可与 HI16 宏一起使用，以指定 PSoC 5LP 的源地址的高 16 位，但不用于 PSoC 3。这是因为在 PSoC 3 中 RAM 变量地址的高 16 位为零，但 Keil 编译器会将 Keil 指定信息存储在变量地址的高 16 位中。因此，HI16 (&adc_sampleArray) 与 PSoC 3 — Keil 编译器一起使用时，将返回错误地址。

该示例中，每个 DMA 请求都要将 2 字节 ADC 结果从 ADC 传输到 RAM 阵列，并将突发计数设置为 2，将每次突发请求设置为 True。

保留 TD 被设置为 1 (TRUE)，使原始的 TD 设置 (即源地址、目标地址和传输计数) 得以保留，并且能够重复数据操作。

源地址和目标地址的低 16 位数据由数据操作描述符 (TD[0]) 配置指定，如表 6 所示。

表 6. TD[0]配置设置

参数	项目设置
低位源地址	LO16 (ADC_Delsig_DEC_OUTSAMP_PTR)
低位目标地址	LO16 (&adc_sampleArray)
传输计数	200 x 2 (样本数量 x 每个样本的字节数量)
TD 的属性	递增目标地址， 生成 DMA 完成事件， 只有 PSoC 3 需要交换使能。 (TD_INC_DST_PTR DMA__TD_TERMOUT_EN TD_SWAP_EN)
下一个 TD	DMA_DISABLE_TD

传输计数被设置为 ‘400’，它是被缓冲的样本数量和每个样本字节数的乘积。

设置 TD 属性，以便在每次突发传输后递增目标地址，并且指定的样本数量被缓冲时，将生成“传输结束”信号。在 PSoC 3 项目中，也会配置 TD，以便在数据从 ADC 传输到存储器时可以交换字节 (如“TD 属性”一节所示)。PSoC Creator 自动生成的 CyDmac.h 文件定义了与每个 TD 属性相应的位。将所需属性的位字段进行“或”运算以设置 TD 属性的值。

为了在缓冲样本的指定数量后停止 DMA 传输，需要将 TD[0]连接至用于禁用 DMA 通道的 ‘DMA_DISABLE_TD’。

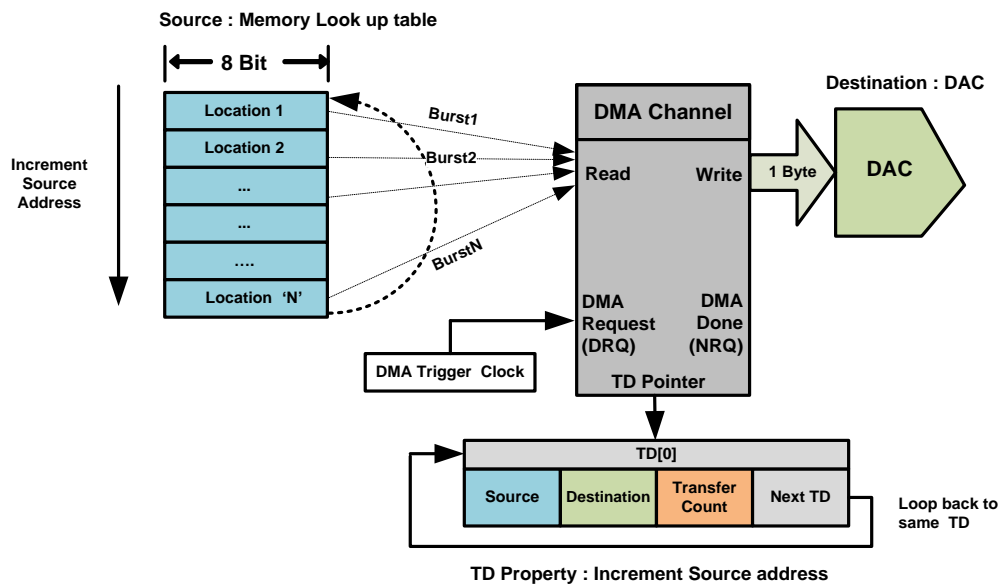
7.2 示例 2：项目文件

AN52705.zip 文件（本应用笔记随附的）中的 Eg2_ADC_DMA_Mem 项目演示了该示例。该示例的 DMA 配置代码类似于示例 1 所展示的。通道和上面的 TD 配置表已经显示了传递到函数的各个参数。请参考[示例项目 — 测试设置](#)，了解如何测试该项目。

8 示例 3：存储器至外设的传输

该示例介绍了存储器至外设的传输如何使用 DMA。该示例演示了如何通过使用 DAC 生成方波，如[图 10](#) 所示。

图 10. 存储器至外设传输的框图

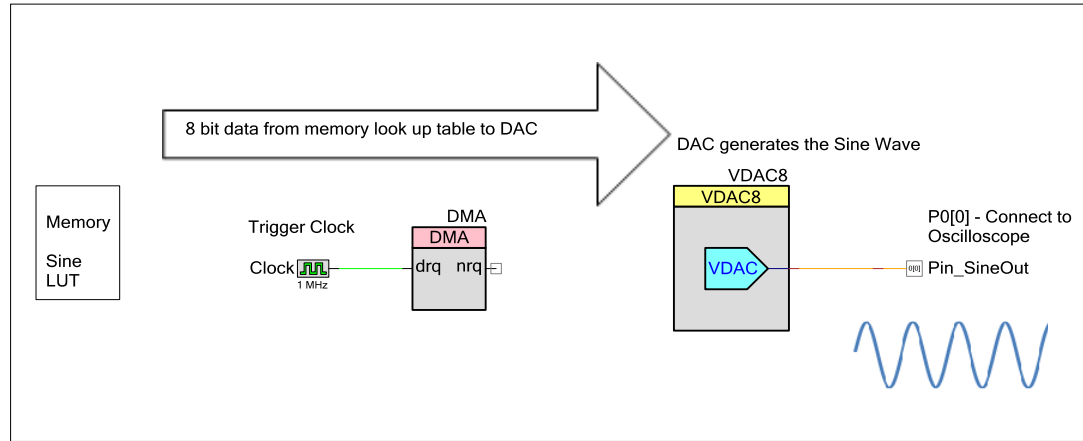


具有 128 点的正弦查询表被存储在闪存存储器中。通过使用 DMA 将这些值依次发送到 DAC 来创建正弦波。[图 11](#) 显示的是项目的顶层设计。

使用时钟组件来定期生成 DMA 请求（drq）。收到请求时，DMA 通道会从查询表读取一个数据字节，并将它写入 DAC 数据寄存器。每次突发传输后，源地址的值增加 1，传输计数的值减 1。持续该情况，直到表中所有值被发送到 DAC。

传输结束时，保留和重新加载 TD 配置，以便生成一个连续正弦波。正弦波的频率等于 DMA 触发时钟频率除以查询表中的点数所得到的结果。

图 11. 存储器至外设传输的顶层设计



8.1 示例 3: DMA 配置

表 7 和表 8 分别显示的是关于该项目的 DMA 通道和 TD 配置。

表 7. 通道配置

参数	项目设置
高位源地址	PSoC 3 的 HI16 (CYDEV_FLS_BASE) PSoC 5LP 的 HI16 (&sineTable)
高位目标地址	HI16 (CYDEV_PERIPH_BASE)
突发计数	1 (一个字节)
每次突发请求	1 (True)
初始 TD	TD[0]
保留 TD	1 (True)

DMA 传输的源是保持在闪存存储器中的 sineTable 阵列。HI16(&sineTable)设置了 PSoC 5LP 源地址的高 16 位，而 HI16(CYDEV_FLS_BASE)被用于识别 PSoC 3 源地址的高 16 位（前面的示例已提到原因）。

对于每一个 DMA 请求，DMA 通道需要将一个字节从查询表阵列传输到 DAC。因此，突发计数被设置为 1 字节，并且每次突发请求被设置为 True。

保留原始 TD 配置，以便可以重复使用。

使用表 8 所示的 LO16 宏来设置源地址和目标地址的低 16 位。

表 8. TD[0]配置

参数	项目设置
低位源地址	LO16 (&sineTable)
低位目标地址	LO16 (VDAC8_DATA_PTR)
传输计数	128 (在正弦波查询表中的字节数量)
TD 属性	递增源地址 (TD_INC_SRC_ADR)
下一个 TD	TD[0] — 重新回送同样的 TD

传输计数被设为正弦波查询表中的总字节数量。

每次突发传输后，都会对 TD 进行配置，以便递增源地址（即查询表指针）。

传输结束后，将在 DAC 输出上生成一个完整周期的正弦波形。TD 被保留并回环给自己，以便生成连续波。

8.2 示例 3：项目文件

AN52705.zip 文件（本应用笔记随附的）上的 Eg3_Mem_DMA_DAC 项目演示的是该示例。该示例的 DMA 配置代码类似于示例 1 所展示的。通道和上面的 TD 配置表已经显示了传递到函数的各个参数。请参考[示例项目 — 测试设置](#)，了解如何测试该项目。

9 示例 4：存储器至存储器的传输

该示例显示了如何使用 DMA 执行存储器至存储器的传输。它还演示了如何使用 CPU 来触发 DMA。在该示例中，CPU 请求时，8 字节闪存阵列被复制到 8 字节的 RAM 阵列中，如图 12 所示。

图 12. 存储器至存储器传输的框图

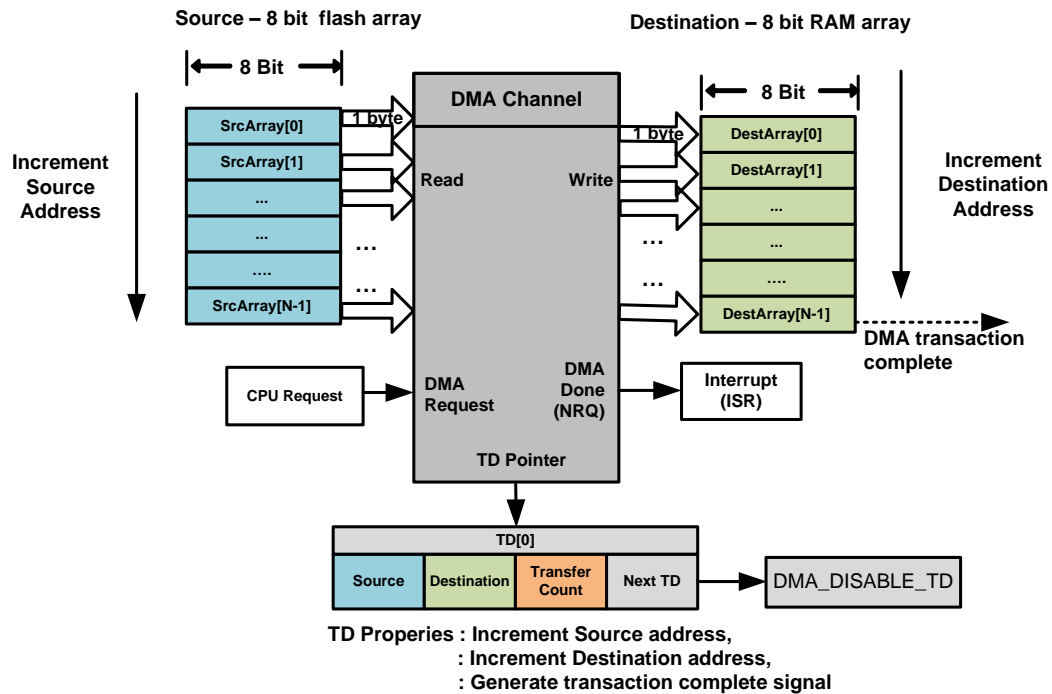
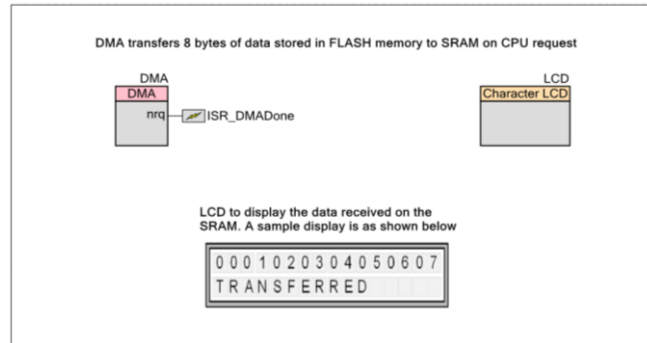


图 13 显示的是项目的顶层设计。使用 CyDmaChSetRequest 函数，以便在器件上电大约一秒后激活 DMA 传输。

当收到 CPU 的请求时，DMA 将 8 字节数据从闪存阵列传输到 RAM 阵列（如通道和 TD 配置寄存器所配置的）。执行传输时，TD 源地址和目标地址的值会递增。

传输完成时，将在 DMA 的 nrq 信号终端生成一个脉冲。这样会激活 ISR_DMADone 中断，从而设置表示传输已经完成的标志。然后，在 LCD 上会显示新的 RAM 内容。

图 13. 存储器至存储器传输的顶层设计



9.1 示例 4: DMA 配置

表 9 和表 10 分别显示的是关于该项目的 DMA 通道和 TD 配置。

表 9. 通道配置

参数	项目设置
高位源地址	PSoC 3 的 HI16 (CYDEV_FLS_BASE) PSoC 5LP 的 HI16 (&sourceArray)
高位目标地址	HI16 (CYDEV_SRAM_BASE)
突发计数	1 (一个字节)
每次突发请求	0 (False)
初始 TD	TD[0]
保留 TD	0 (False)

DMA 传输的源地址是闪存存储器所定义的 ‘sourceArray’。其目标地址是 RAM 的 ‘destinationArray’。在 PSoC 5LP 和 PSoC 3 上，闪存中源地址的高 16 位分别被设置为 HI16(&sourceArray)和 HI16(CYDEV_FLS_BASE)，如上述示例所示。同样，使用宏 HI16 (CYDEV_SRAM_BASE) 来设置 SRAM 中目标地址的高 16 位。

突发计数被设置为 1 字节，以便 DMA 从闪存上读取每个字节，并将它们写到 RAM 阵列内。您可以将突发计数设置为 8 个字节，以提高数据传输的速度。但是，您通常应该将突发计数设置为低数值，以便其他 DMA 通道共用轮辐。

每次突发请求的参数被设置为 “False”，以便使每一个突发传输不再需要单独的请求。

表 10. TD[0]配置

参数	项目设置
源地址	LO16 (&sourceArray)
目标地址	LO16 (&destinationArray)
传输计数	8 (字节)
TD 属性	递增源地址 递增目标地址 生成 “DMA 完成” 信号 (TD_INC_SRC_ADR TD_INC_DST_ADR DMA__TD_TERMOUT_EN)
下一个 TD	DMA_DISABLE_TD (0xFE)

对于 TD 配置，源地址和目标地址的低 16 位由 LO16 宏识别。

传输计数被设置为 8，以便将总共 8 个字节数据从源地址传输到目标地址。

每次突发传输后，都要配置 TD，以便递增源地址（即闪存阵列指针）和目标地址（即 RAM 阵列指针）。从闪存中将所有 8 个字节传输到 RAM 阵列后，将对 TD 进行配置，以在 nrq 线路上生成 termout 脉冲。该脉冲用于触发 ISR，以表示传输已经完成。传输完成后，下一个 TD 将被设置为 DMA_DISABLE_TD (0xFE) 以禁用 DMA 通道。

因为数据操作仅发生一次，所以无需保留 TD 配置。

9.2 示例 4：项目文件

AN52705.zip 文件（本应用笔记随附的）的 Eg4_Mem_DMA_Mem 项目演示了该示例。该示例的 DMA 配置代码类似于示例 1 所展示的。通道和上面的 TD 配置表已经显示了传递到函数的各个参数。请参考[示例项目 — 测试设置](#)，了解如何测试该项目。

10 示例 5：TD 链路

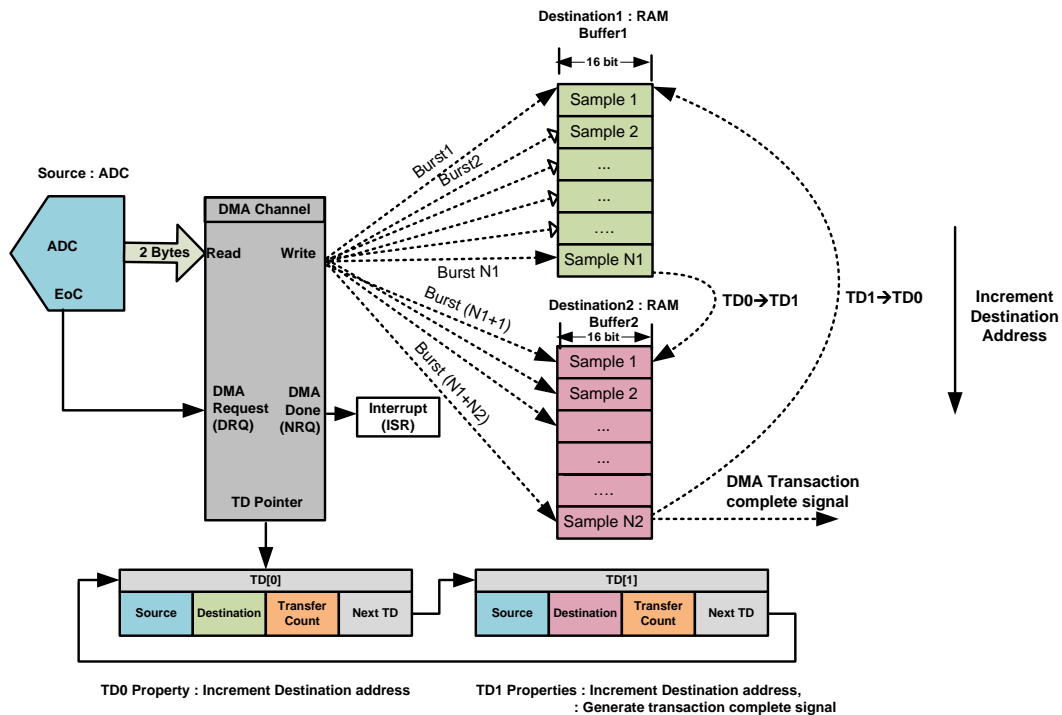
该示例显示了如何使用多 TD 和一个通道，并将各 TD 互相链接起来。在该示例中，通过使用单个 DMA 通道和两个 TD，ADC 数据依次被发送到两个单独的 RAM 缓冲器内。

配置 DMA 通道以执行下面两个数据操作：

- 第一个数据操作：ADC 至 RAM 缓冲器 1
- 第二个数据操作：ADC 至 RAM 缓冲器 2

通过使用两个单独的数据操作描述符（TD[0]和 TD[1]）配置这两个数据操作，并且使用 DMA 的 TD 链路特性将这两个数据操作互相链接起来，如[图 14](#) 显示。

图 14. TD 链路框图



可以使用这种 TD 配置形式来克服单一 TD 最大传输计数的限制（对于每一个 DMA 通道，它被限制为 4096 个字节）。注意：对于链路上的所有 TD，源地址和目标地址的高 16 位必须相同。

项目的顶层设计类似于示例 2 所示的内容。

10.1 示例 5: DMA 配置

表 11、表 12 和表 13

表 11. 通道配置

参数	设置
高位源地址	HI16 (CYDEV_PERIPH_BASE)
高位目标地址	HI16 (CYDEV_SRAM_BASE)
突发计数	2 (两个字节)
每次突发请求	1 (True)
初始 TD	TD[0]
保留 TD	1 (True)

通道和 TD 配置与示例 2 的相同。TD[0]的下一个 TD 参数被设置为 TD[1] (反之亦然) 以便链接各个数据操作。

表 12. TD[0]配置

参数	项目设置
低位源地址	LO16 (ADC_Delsig_DEC_OUTSAMP_PTR)
低位目标地址	LO16 (adc_sampleArray1)
传输计数	N1×2 (样本数量 × 每个样本的字节数量)
TD 的属性	递增目标地址: TD_INC_DST_ADR 生成“DMA 完成”事件: DMA__TD_TERMOUT_EN PSoC 3 所需的交换使能: TD_SWAP_EN
下一个 TD	TD[1]

表 13. TD[1]配置

参数	项目设置
低位源地址	LO16 (ADC_Delsig_DEC_OUTSAMP_PTR)
低位目标地址	LO16 (adc_sampleArray2)
传输计数	N2×2 (样本数量 × 每个样本的字节数)
TD 的属性	递增目标地址: TD_INC_DST_ADR 生成“DMA 完成”事件: DMA__TD_TERMOUT_EN PSoC 3 所需的交换使能: TD_SWAP_EN
下一个 TD	TD[0]

10.2 示例 5：项目文件

AN52705.zip 文件（本应用笔记随附的）的 **Eg5_TD_Chaining** 演示了该示例。该示例的 DMA 配置代码类似于示例 1 所展示的。通道和上面的 TD 配置表已经显示了传递到函数的各个参数。请参考第 26 页上的[示例项目 — 测试设置](#)，了解如何测试该项目。

11 总结

本应用笔记描述了 PSoC 3 和 PSoC 5LP 的 DMA 控制器。通过使用简单的 PSoC Creator 示例项目，应用笔记还显示了对不同数据传输类型如何进行 DMA 的配置。更多高级信息，请参考 [PSoC 3](#) 和 [PSoC 5LP 技术参考手册](#) 和 [PSoC Creator DMA 组件数据手册](#)。

12 关于作者

姓名： Anu M D

职称： 高级应用工程师

背景： Model Engineering College 电子通信学学士学位。

附录A. DMA 配置步骤

第一步：DMA 通道初始化

```
Channel_Handle = DMA_DmaInitialize(
    DMA_BYTES_PER_BURST,
    DMA_REQUEST_PER_BURST,
    HI16(Source Address),
    HI16(Destination Address))
```

API 函数 DmaInitialize() 对以下几个 DMA 通道参数进行配置：

- **DMA_BYTES_PER_BURST:** 由 DMA 通道在一个突发中所读取和写入的字节数量。
例如，如果您要定义 DMA 来收集 8 位 ADC 数据，将该参数设置为 1，因为 DMA 通道每次发送请求时都要将 1 字节从源地址传输到目标地址。或者，如果您要收集 16 位 ADC 数据，请将该参数设置为 2。
- **DMA_REQUEST_PER_BURST:** 每次突发是否必须单独请求。
如果将该位设置为 1，则必须单独请求每次突发传输。如果将其设置为 0，则第一个突发后自动执行所有后续突发，不需要单独请求。（只有第一个突发传输具有 DMA 请求。）
- **HI16 (源地址):** 源地址的高 16 位。HI16 是由 PSoC Creator 所创建的宏，以指定 32 位值或 32 位地址的高 16 位。
- **HI16 (目标地址):** 目标地址的高 16 位。在 PSoC 3 中，使用上表所提供的宏来识别源地址和目标地址的高 16 位。

PSoC 3 Keil 编译器将 Keil 指定信息存储于变量地址的高 16 位。因此，使用表 14 所显示的常量。在 *CyDevice.h* 文件中定义它们和 HI16 宏，以便配置 PSoC 3 的源地址或目标地址高 16 位（特别在 DMA 传输的源地址和目标地址是 RAM 或闪存存储器的情况）

表 14. 高 16 位地址宏

源	DMA_SRC_BASE
外设	CYDEV_PERIPH_BASE
RAM	CYDEV_SRAM_BASE
闪存	CYDEV_FLS_BASE

第二步：TD 分配

```
TD_Handle = CyDmaTdAllocate();
```

API 函数 `CyDmaTdAllocate()` 创建 TD 的实例，并将句柄返回到原来那个 TD。其他 API 使用 TD 句柄配置 TD。多次调用函数以创建多个 TD。

第三步：TD 配置

```
CyDmaTdSetConfiguration(TD_Handle,
                        Transfer_Count,
                        Next_TD,
                        TD_Property);
```

API 函数 `CyDmaTdSetConfiguration()` 通过使用以下参数配置 TD：

- **TD_Handle:** 由 `CyDmaTdAllocate()` 函数先前返回的句柄
- **Transfer_Count:** 从源地址传输到目标地址的字节总数。
- **Next_TD:** TD 链中下一个 TD 索引。如果它是链中最后一个 TD，则 TD 传输成功后，将使用 `DMA_DISABLE_TD` 宏（0xFE）来禁用 DMA 通道。
- **TD_Property:** 使用第 20 上的表 15 中显示的 TD 配置寄存器标志进行设置 DMA 数据操作的属性。对标志进行“或”运算，以便配置 TD 属性。比如，在数据传输过程中，为了配置 TD 以交换 4 个字节，请使用：

```
(TD_SWAP_EN | TD_SWAP_SIZE4)
```

表 15. TD 的属性

配置标志	函数
TD_SWAP_EN	执行字节序交换；当将数据从源地址传输到目标地址时，将交换字节。
TD_SWAP_SIZE4	将交换大小设置为 4 个字节。交换大小的默认值为 2 个字节。
TD_AUTO_EXEC_NEXT	完成当前 TD 后，链中的下一个 TD 会自动被激活。
TD_TERMIN_EN	如果 trq 输入线上产生一个上升沿，则结束该 TD。在一次突发过程中必须发生上升沿。只有这时 DMAC 才会对其进行侦听。
DMA__TD_TERMOUT_EN	如果使用该标志，TD 传输完成时，将在 nrq 线上生成一个脉冲。该标志是 DMA 组件实例的特例，并由组件实例头文件定义。比如，在顶层设计中，如果 DMA 组件实例名称是 DMA_1，则实例的 termout 宏是包含在 DMA_1_dma.h 内的 'DMA_1_TD_TERMOUT_EN'。
TD_INC_DST_ADR	根据突发中每个数据操作的大小递增目标地址。
TD_INC_SRC_ADR	根据突发中每个数据操作的大小递增源地址。

第四步：配置 TD 源地址和目标地址

```
CyDmaTdSetAddress(TD_Handle,
                  LO16(source),
                  LO16(destination))
```

API 函数 `CyDmaTdSetAddress()` 通过使用以下函数设置 TD 的源地址和目标地址：

- **TD_Handle:** 由 `CyDmaTdAllocate()` 函数先前返回的句柄
- **LO16(source):** 源地址的低 16 位
- **LO16(destination):** 目标地址的低 16 位

PSoC 具有高度可编程性，许多组件由可编程数字和模拟模块创建，并且外设的物理位置可根据设计制定。因此，传统的寄存器映射无法列出所有源地址和目标地址。

但在编译过程中，每组件的寄存器均由 PSoC Creator 所生成的组件 API 头文件定义。您应该查看这些头文件，以识别组件的寄存器地址。

第五步：将 TD 连接到通道

```
CyDmaChSetInitialTd(Channel_Handle,  
                    TD_Handle)
```

API 函数 CyDmaChSetInitialTD() 设置 DMA 通道的第一个 TD：

- **Channel_Handle:** DMA_DmaInitialize() 函数返回的 DMA 实例句柄
- **TD_Handle:** 由 CyDmaTdAllocate() 函数先前返回的句柄

第六步：使能 DMA 通道

```
CyDmaChEnable(Channel_Handle,  
              Preserve_TD)
```

API 函数 CyDmaChEnable() 用于使能 DMA 通道：

- **Channel_Handle:** DMA_DmaInitialize() 函数返回的 DMA 实例句柄
- **Preserve_TD:** 如果为 TRUE，则 DMA 通道将保留 TD 配置（源地址、目标地址以及传输计数），从而能够重复 TD

A.1 其他重要的 DMA API 函数

要想通过 CPU 请求激活 DMA 通道，需要使用以下函数：

```
CyDmaChSetRequest(Channel_Handle, CPU_REQ);
```

为了禁用 DMA 通道，需要使用以下函数：

```
CyDmaChDisable(Channel_Handle);
```

附录B. DMA 向导配置

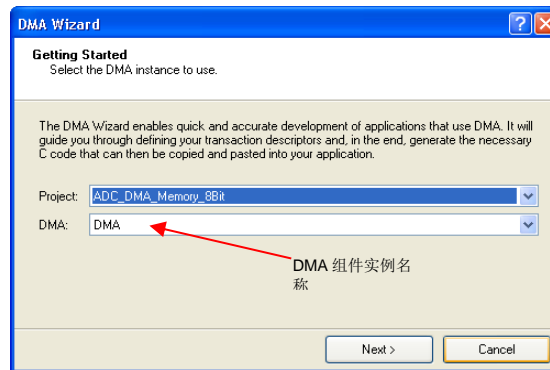
作为附录 A 所描述各步骤的代替方法，通过 DMA 向导，对 DMA 通道和 TD 的固件配置进行定义变得非常容易。但是，该向导仅支持某些外设（如 DMA 源地址或目标地址）。如果某个外设不受支持，请按照附录 A 中所描述的配置步骤执行。

为了启动 DMA 向导，跳转至 **PSoC Creator > Tools > DMA Wizard**。

第一步：选择 DMA 通道（DMA 组件实例）

选择需要配置的 DMA 通道，如图 15 显示：

图 15. 选择 DMA 通道



根据下面的内容选择对话框参数：

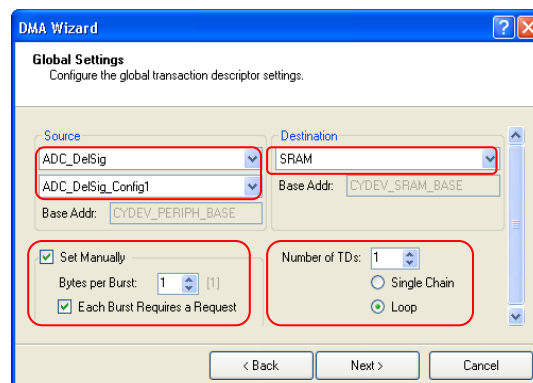
- **Project (项目)**：PSoC Creator 项目的名称
- **DMA**：项目的 DMA 组件实例名称

该操作完成后，请点击 **Next**。

第二步：选择全局设置

选择 DMA 传输全局设置，如图 16 显示：

图 16. 全局设置



在该对话框内选择 DMA 通道配置参数：

Source and Destination (源地址和目标地址)：源地址和目标地址的高 16 位

Bytes per Burst (每突发的字节)：在单一的突发中所传输的字节数量

Each Burst Requires a Request (每突发需要一个请求)：每个突发是否需要单独的请求

Number of TDs (TD 数量)：有关 DMA 通道的数据操作描述符数量（1 至 128）。

Single Chain or Loop (单链接或循环)：该函数为链中最后 TD 定义‘下一个 TD’。如果是单链接，则下一个 TD 为 DMA_DISABLE_TD (0xFE)。如果是循环，它是第一个 TD。

该操作完成后，请点击 **Next**。

第三步：定义通道的数据传输描述符。

选择 DMA 传输全局设置，如图 17 显示。表 16 描述每个 TD 配置参数。

图 17. 添加数据操作描述符

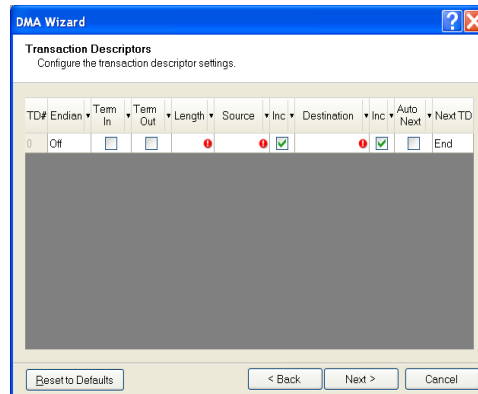


表 16. TD 配置详细信息

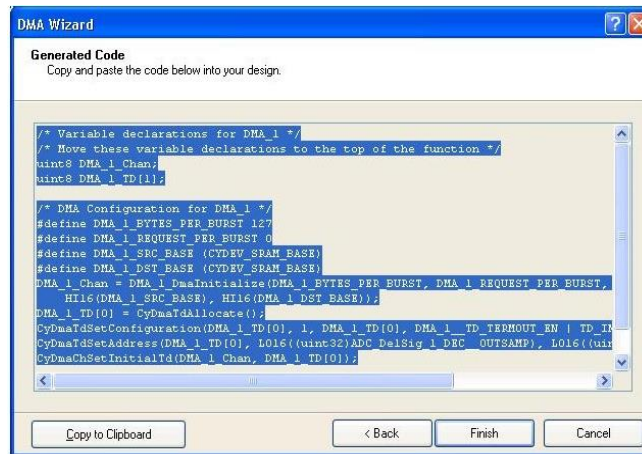
字段	描述
TD#	显示了数据操作描述符的逻辑编号。
字节序	允许按 2 或 4 字节的字节序进行交换。当数据从源地址传输到目标地址时，该字段会交换字节。必须将 每个突发字节 设置为 Endian 选择的整数倍。由于字节序存在差异，所以通常将该字段用在 PSoC 3 存储器与外设之间的 DMA 传输。
Term In	在 TERMIN (trq) 信号的上升沿上结束 TD 数据传输。
Term Out	当 TD 完成时，将会创建 TERMOUT (nrq) 信号。
长度	该字段指定了 TD 的传输计数（单位为字节，从 0 至 4095）。这是 DMA 需要传输的字节总数，以完成数据传输。
源地址	DMA 传输所需要的低 16 位源地址。如果选定的源地址是一个组件（而不是存储器），则 DMA 向导将提供源地址的下拉列表。您可以手动编辑或进入源地址。
Inc (Source)	当 DMA 执行数据传输时，会增加源地址。如果该字段被使能，每次 DMA 读取源中的数据，源地址会以 DMA 所读取的字节数量递增。DMA 增加源地址，直到整个数据操作（传输计数）完成为止。
目标地址	DMA 传输所需的低 16 位目标地址。如果选定的目标地址是一个组件（而不是存储器），则 DMA 向导会提供目标地址的下拉列表。您也可以手动编辑或进入目标地址。
Inc(Destination)	当 DMA 执行数据传输时，会增加目标地址。DMA 增加目标地址，直到整个数据操作（传输计数）完成为止。
Auto Next (自动执行下一个)	自动执行下一个 TD，无需其他 DMA 请求。
Next TD (下一个 TD)	在 TD 链中的下一个逻辑 TD。如果该 TD 链结束于这个 TD，请将该字段设置为 END。

该操作完成后，请点击 **Next**。然后会生成所需的代码。

第四步：复制 DMA 向导所创建的代码

完成 DMA 通道和 TD 的配置后，向导将为 DMA 通道创建代码。该代码包含 DMA 通道和 TD 的配置，在 DMA 向导对话框的窗口中生成代码，如图 18 显示。要使用代码，请选择窗口内的所有内容，然后复制它们，并将这些内容粘贴在 *main.c* 中。

图 18. 生成代码

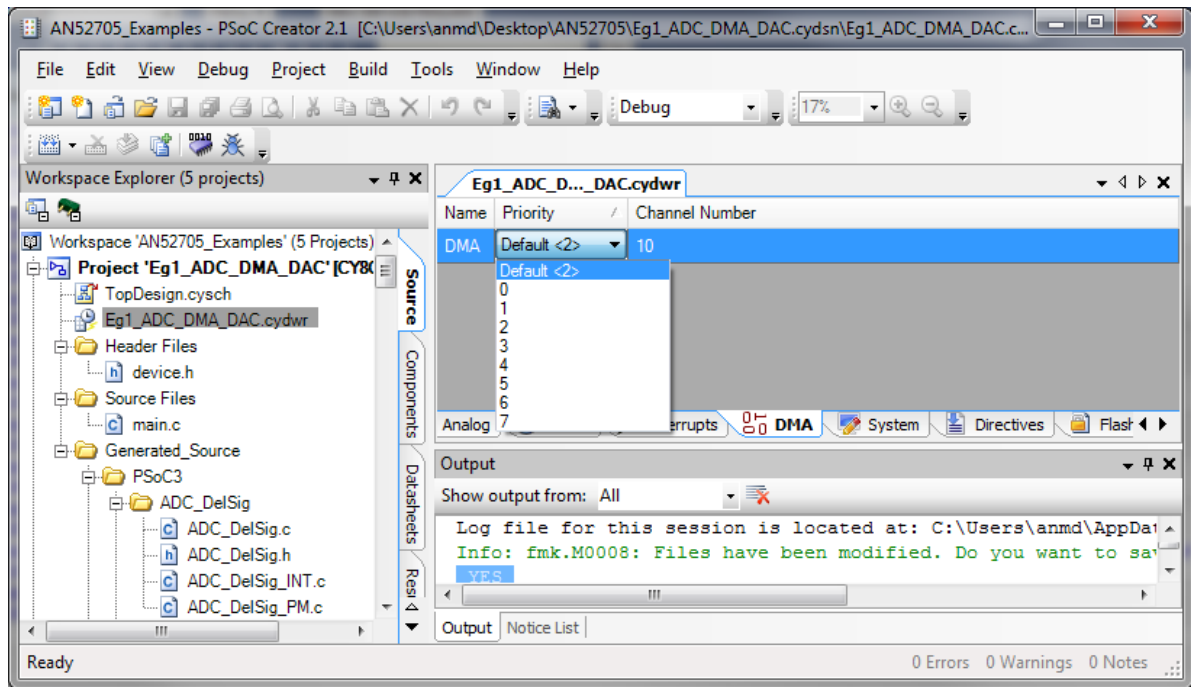


有关向导的详细信息，请参考 PSoC Creator 帮助文件。

附录C. 设置 DMA 通道的优先级

当激活多个 DMA 通道请求时，DMAC 会根据通道优先级设置处理 DMA 通道。可以将某个 DMA 通道设置为八个不同优先级中的某一级。在 PSoC Creator 窗口中 **Design Wide Resources (*.cydwr)** > **DMA** 选项卡下可以设置 DMA 通道的优先级，如图 19 显示。

图 19. 设置 DMA 通道的优先级



当 CPU 和 DMAC 同时请求访问 PHUB 上的同一个轮辐时，CPU 在默认情况下有优先权。PHUB 管理着 DMA 和 CPU 间、各个 DMA 通道间的仲裁。有关更多信息，请参考 [PSoC® 3](#)、[PSoC® 5LP 架构 TRM](#)。

附录D. 示例项目 — 测试设置

D.1 示例 1：外设至外设的传输 — Eg1_ADC_DMA_DAC

在该示例项目中，ADC 采样频率（fs）为 384 kHz。如果输入的频率小于或等于 84 kHz，则最好重建输出，因为 delta sigma ADC 有低通特性（频率为 0.22 fs 时信号会消减-3dB）。按照以下步骤可实现测试设置：

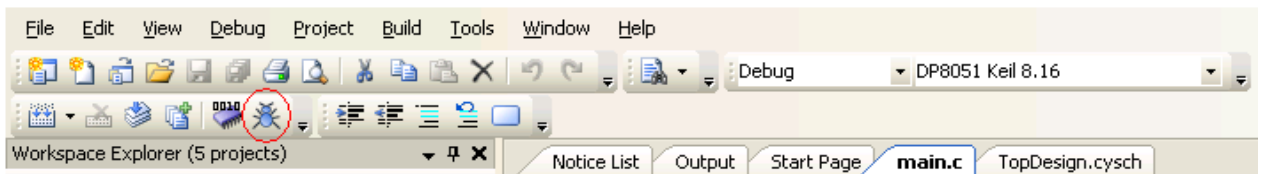
1. 将功能生成器和输入分别连接到 P0[2]引脚和 ADC。
2. 设置功能生成器以使正弦波的频率为 100 Hz。
3. 将示波器探头分别连接至 P0[0]引脚和 VDAC 输出。
4. 编译项目并编程器件。
5. 从示波器上的引脚 P0[0]查看输出。输出端是一个频率为 100 Hz 的正弦波，与输入端相同。

D.2 示例 2：外设至存储器的传输 — Eg2_ADC_DMA_Mem

按照以下步骤可实现测试设置：

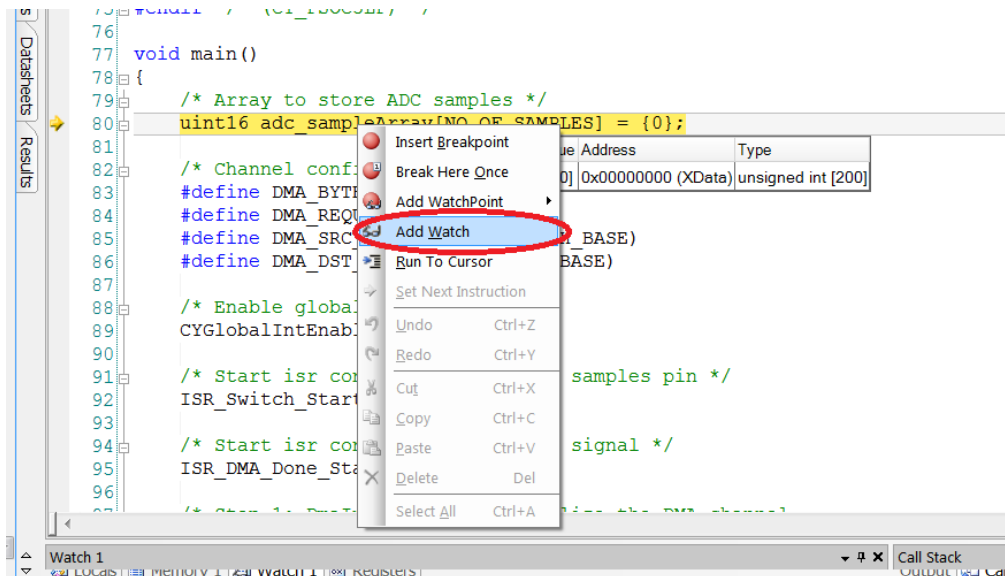
1. 将输入信号和输入分别连接到 P0[2]引脚和 ADC。确保输入电压位于 ADC 范围内（从 V_{SSA} 至 2.048 V）。
2. 将 P6[1]连接至 DVK 上的开关（SW1）。
3. 编译项目。
4. 按 F5 或点击调试图标，如图 20 显示，以下载该程序并开始调试。

图 20. 调试按键



5. 将 adc_sampleArray 作为查看变量，如图 21 显示：

图 21. 查看变量



- 将断点放在 if(DMADone_flag) 循环中，如图 22 所示。

图 22. 添加断点

```

154 |
155 |     /* If statement ends here */
156 |
157 |     /* DMADone_flag is set inside ISR_DMA_Done after the
158 |      * of ADC samples are buffered */
159 |     if(DMADone_flag)
160 |     {
161 |         /* Put a break point here to view the data in the
162 |          * DMADone_flag = 0;
163 |     }
164 |     /* If statement ends here */
165 |
166 | } /* for loop ends here */
167 |
168 | /* Place your application code here. */
169 | }
170 |

```

- 按下 F5，以运行程序。按住连接至 P6[1] 的开关（SW1），让 DMA 启动 ADC 采样缓冲。DMA 将指定的样本数量从 ADC 传输到存储器后，执行数据操作将停止在断点上。通过监视窗口上的 **adc_sampleArray** 可以验证结果，如图 23 显示：

图 23. 在监视窗口中进行 ADC 采样

Watch 1				
Name	Value	Address	Type	Radix
adc_sampleArray	[200]	0x00000000 (XData)	unsigned int [200]	Default
0	0x7FFE	0x00000000 (XData)	unsigned int	Default
1	0x7FFD	0x00000002 (XData)	unsigned int	Default
2	0x7FFE	0x00000004 (XData)	unsigned int	Default
3	0x7FFF	0x00000006 (XData)	unsigned int	Default
4	0x7FFC	0x00000008 (XData)	unsigned int	Default
5	0x7FFD	0x0000000A (XData)	unsigned int	Default
6	0x7FFE	0x0000000C (XData)	unsigned int	Default

D.3 示例 3：存储器至外设的传输 — Eg3_Mem_DMA_DAC

按照以下步骤可实现测试设置：

- 将示波器探头分别连接至 P0[0] 引脚和 VDAC 输出。
- 编译项目并编程器件。
- 在示波器上观察频率为 7.8 kHz 的正弦波。

D.4 示例 4：存储器至存储器的传输 — Eg4_Mem_DMA_Mem

按照以下步骤可实现测试设置：

- 将字符 LCD 模块连接至 **CY8CKIT-001 PSoC 开发套件** 的 P18 头（LCD 模块 — 端口 2）。
- 确保 J12 跳线处于 ON 位置，以为 LCD 供电。
- 编译项目并编程器件。
- 查看 LCD 显示屏。第一行显示的是目标阵列的内容。最初所有值为零。经过一秒后，第一行显示从 00 到 07 各数值，表示 DMA 已经成功将数据从闪存传输到 RAM。第二行显示的是‘已传输（TRANSFERRED）’信息。图 24 显示的是 LCD 显示的一个示例：

图 24. DMA 传输的 LCD 显示



D.5 示例 5: TD 链路 — Eg5_TD_Chaining

该示例的测试设置与示例 2 的相同。按照以下步骤可实现测试设置：

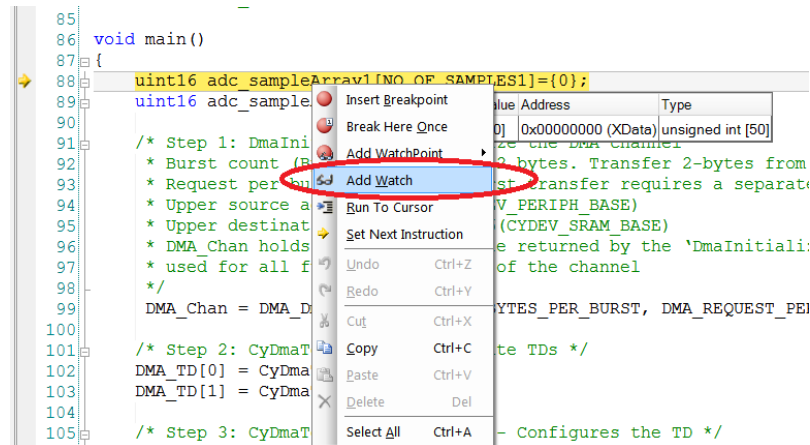
1. 将输入信号和输入分别连接到 P0[2]引脚和 ADC。确保输入电压位于 ADC 范围内（从 V_{SSA} 至 2.048 V）。
2. 将 P6[1]连接至 DVK 上的开关（SW1）。
3. 编译项目。
4. 按 F5 或点击调试图标，如图 25 显示，以下载该程序并开始调试。

图 25. 调试按键



5. 添加 adc_samplearray1 和 adc_samplearray2，作为查看变量使用，如图 26 所示。

图 26. 查看变量



6. 将断点放在 if (DMADone_flag) 循环中，如图 27 所示。

附录E. 常见问题解答

1. 如何使用 DMA 缓冲超过 4095 个字节？

TD 的最大传输计数被限制为 4095 个字节。如果您需要使用单个 DMA 通道传输多于 4095 个字节，请使用多个 TD 并互相链接它们，如示例 5 所示。

2. 在 DMA 数据传输中，如何找到外设的源地址和目标地址？

PSoC 具有高度可编程性，许多组件由可编程数字和模拟模块创建，并且外设的物理位置可根据设计制定。因此，传统的寄存器映射无法列出所有源地址和目标地址。

但在编译过程中，每组件的寄存器均由 PSoC Creator 所生成的组件 API 头文件定义。您应该查看这些头文件，以识别组件的寄存器地址。

3. 如何联合使用 DMA 和通信协议（UART、SPI 等）？

使用 UART 和 SPI 等通信协议时，请将缓冲区大小设置为 4 或更小，以便不会触发数据传输的内部中断。将硬件 FIFO 指针作为 DMA 的读写数据地址，并通过使用配置为中断的 FIFO 级别状态来触发 DMA。将 DMA 通道的硬件请求配置为电平触发，以便联合使用它和 FIFO 级别。

4. DMA 传输的时序详细信息？

可以在 [PSoC 3 和 PSoC 5LP 技术参考手册](#) 中找到有关 DMA 传输的时序详细信息。有关 DMA 时序的详细讨论超出了本应用笔记的范围。

文档修订记录

文档标题: AN52705 — PSoC® 3 和 PSoC 5LP — DMA 入门手册

文档编号: 001-93054

版本	ECN	变更者	提交日期	变更说明
**	4477365	MSON	08/18/2014	本文档版本号为 Rev**, 译自英文版 001-52705 Rev*H。
*A	4908666	XZNG	09/04/2015	本文档版本号为 Rev*A, 译自英文版 001-52705 Rev*I。
*B	6013587	XITO	01/04/2018	本文档版本号为 Rev*B, 译自英文版 001-52705 Rev*J。
*C	6651886	XZNG	08/19/2019	本文档版本号为 Rev*C, 译自英文版 001-52705 Rev*K。

全球销售和设计支持

赛普拉斯公司拥有一个由办事处、解决方案中心、厂商代表和经销商组成的全球性网络。要想查找离您最近的办事处，请访问[赛普拉斯所在地](#)。

产品

Arm® Cortex®微控制器	cypress.com/arm
汽车级产品	cypress.com/automotive
时钟与缓冲器	cypress.com/clocks
接口	cypress.com/interface
物联网	cypress.com/iot
存储器	cypress.com/memory
微控制器	cypress.com/mcu
PSoC	cypress.com/psoc
电源管理 IC	cypress.com/pmics
触摸感应	cypress.com/touch
USB 控制器	cypress.com/usb
无线连接	cypress.com/wireless

PSoC®解决方案

[PSoC 1](#) | [PSoC 3](#) | [PSoC 4](#) | [PSoC 5LP](#) | [PSoC 6 MCU](#)

赛普拉斯开发者社区

[社区](#) | [项目](#) | [视频](#) | [博客](#) | [培训](#) | [组件](#)

技术支持

cypress.com/support

此处引用的所有其它商标或注册商标都归其各自所有者所有。



赛普拉斯半导体
198 Champion Court San
Jose, CA 95134-1709

© 赛普拉斯半导体公司，2009-2019 年。本文件是赛普拉斯半导体公司及其子公司，包括 Spansion LLC（“赛普拉斯”）的财产。本文件，包括其包含或引用的任何软件或固件（“软件”），根据全球范围内的知识产权法律以及美国与其他国家签署条约归赛普拉斯所有。除非在本款中另有明确规定，赛普拉斯保留在该等法律和条约下的所有权利，且未就其专利、版权、商标或其他知识产权授予任何许可。如果软件没有附带许可协议且贵方未以其他方式与赛普拉斯签署关于使用软件的书面协议，赛普拉斯特此授予贵方适用于个人的、非独占性、不可转让的许可（无转授许可权）（1）在版权保护下的软件（a）以源代码形式提供的软件，只能是在组织内部为了使用赛普拉斯的硬件去修改和复制。（b）以二进制代码形式从外部发到终端用户（直接或间接通过经销商和分销商），仅用于赛普拉斯硬件产品单元。（2）在软件（由赛普拉斯公司提供，且未经修改）侵犯赛普拉斯专利的权利主张下，仅许可在赛普拉斯硬件产品上制造、使用、提供和导入软件。禁止对软件的任何其他使用、复制、修改、翻译或编译。

赛普拉斯不在此材料提供任何类型的明示或暗示保证，包括但不限于针对特定用途的适销性和适用性的暗示保证。没有任何电子设备是绝对安全的。因此，尽管赛普拉斯在其硬件和软件产品中采取了必要的安全措施，但是赛普拉斯并不承担任何由于使用赛普拉斯产品而引起的安全问题及安全漏洞的责任，例如未经授权访问或使用赛普拉斯产品。此外，本材料中所介绍的赛普拉斯产品有可能存在设计缺陷或设计错误，从而导致产品的性能与公布的规格不一致。（如果发现此类问题，赛普拉斯会提供勘误表）赛普拉斯保留更改本文件的权利，届时将不另行通知。在适用法律允许的范围内，赛普拉斯不对因应用或使用本文件所述任何产品或电路引起的任何后果负责。本文件，包括任何样本设计信息或程序代码信息，仅为供参考之目的提供。文件使用人应负责正确设计、计划和测试信息应用和由此生产的任何产品的功能和安全性。赛普拉斯产品不应被设计为、设定为或授权用作武器操作、武器系统、核设施、生命支持设备或系统、其他医疗设备或系统（包括急救设备和手术植入物）、污染控制或有害物质管理系统中的关键部件，或产品植入之设备或系统故障可能导致人身伤害、死亡或财产损失其他用途（“非预期用途”）。关键部件指，若该部件发生故障，经合理预期会导致设备或系统故障或会影响设备或系统安全性和有效性的部件。针对由赛普拉斯产品非预期用途产生或相关的任何主张、费用、损失和其他责任，赛普拉斯不承担全部或部分责任且贵方不应追究赛普拉斯之责任。贵方应赔偿并保护赛普拉斯免受所有索赔的损害，包括因人身伤害或死亡引起的索赔、费用、损失和其它责任。

赛普拉斯、赛普拉斯徽标、Spansion、Spansion 徽标，及上述项目的组合，WICED，及 PSoC、CapSense、EZ-USB、F-RAM 和 Traveo 应视为赛普拉斯在美国和其他国家的商标或注册商标。请访问 cypress.com 获取赛普拉斯商标的完整列表。其他名称和品牌可能由其各自所有者主张为该方财产。