Freescale Semiconductor
应用笔记

Document Number: AN4590
Rev 0, 9/2012

在 Kinetis K 系列上使用 DMA 模拟 ADC 灵活扫描模式

作者: Lukas Vaculik
Rožnov pod Radhoštem
Czech Republic

内容

1 简介 1
2 实施细节 1
2.1 DMA 传输术语 1
2.2 灵活扫描模式流程 2
2.3 DMA 特殊设置 2
3 示例 3
3.1 示例流程 3
3.2 主文件中的函数 5
3.3 DMA 通道初始化 5

1 简介

Kinetis K 系列微控制器所提供的模数转换器（ADC）支持针对两条以上输入通道的嵌入式扫描模式。通道数由 ADCx_SCA 和 ADCx_SCB 寄存器定义，且转换后，转换结果位于 ADCx_RA 和 ADCx_RB 寄存器中。Kinetis K 系列微控制器还可提供强大、复杂的 DMA 外设（具有多达 16 条通道），该外设与 ADC 组合后可扫描两条以上通道。

本应用笔记介绍如何将 ADC 和 DMA 整合成为强大、灵活的外设，以便将任意 ADC 输入的结果数据流传输至 SRAM。

2 实施细节

2.1 DMA 传输术语

在本应用笔记的 DMA 传输讨论中使用了以下术语：

- 在次循环中，每一个请求启动一个传输组。次循环可以基本单位的形式传输 1-4 GB。基本传输单位为 8 位。因此，对于 16 位数据，1 个次循环支持 2 次传输。
- 主循环可将多个次循环链接在一起。主循环可以修改源地址和目标地址；主循环完成之后，还支持循环缓冲区模式。中断请求可用来异步地结束主循环传

输与半传输事件。半周期传输和全周期传输完成一起可使用双缓冲原则实施。

- 链接是一种可以链接一个以上 DMA 通道的特殊 eDMA 功能。链接通道之后，您可以通过定义要转换的通道顺序使用单个请求启动一个以上的传输。该请求将启动 1 条 DMA 通道上的传输，当该通道上的传输完成后，启动下 1 条通道上的传输。可以为主循环和次循环完成单独定义通道链接。

2.2 灵活扫描模式流程

ADC 灵活扫描模式要求 1 个 ADC 转换器应具有 2 条 DMA 通道。优先级较高的 DMA 通道 1 将 ADCx_RA 寄存器中的 ADC 结果数据传输至 SRAM 中的存储缓冲区。优先级较低的 DMA 通道 0 将常量缓冲器中下一个 ADC 通道设置（输入多路复用器通道和单端/差分模式）存储在 SRAM 或 flash 存储器中。在灵活扫描模式下，将会执行以下步骤：

1. 转换完成标志 ADCxSC1A.COCO 请求通道 1 进行 DMA 传输。
2. 通道 1 传输完成后，将结果值传输至 SRAM 缓冲器。
3. 由于通道 1 和通道 0 相连，因此，通道 1 完成将请求在通道 0 上启动传输。

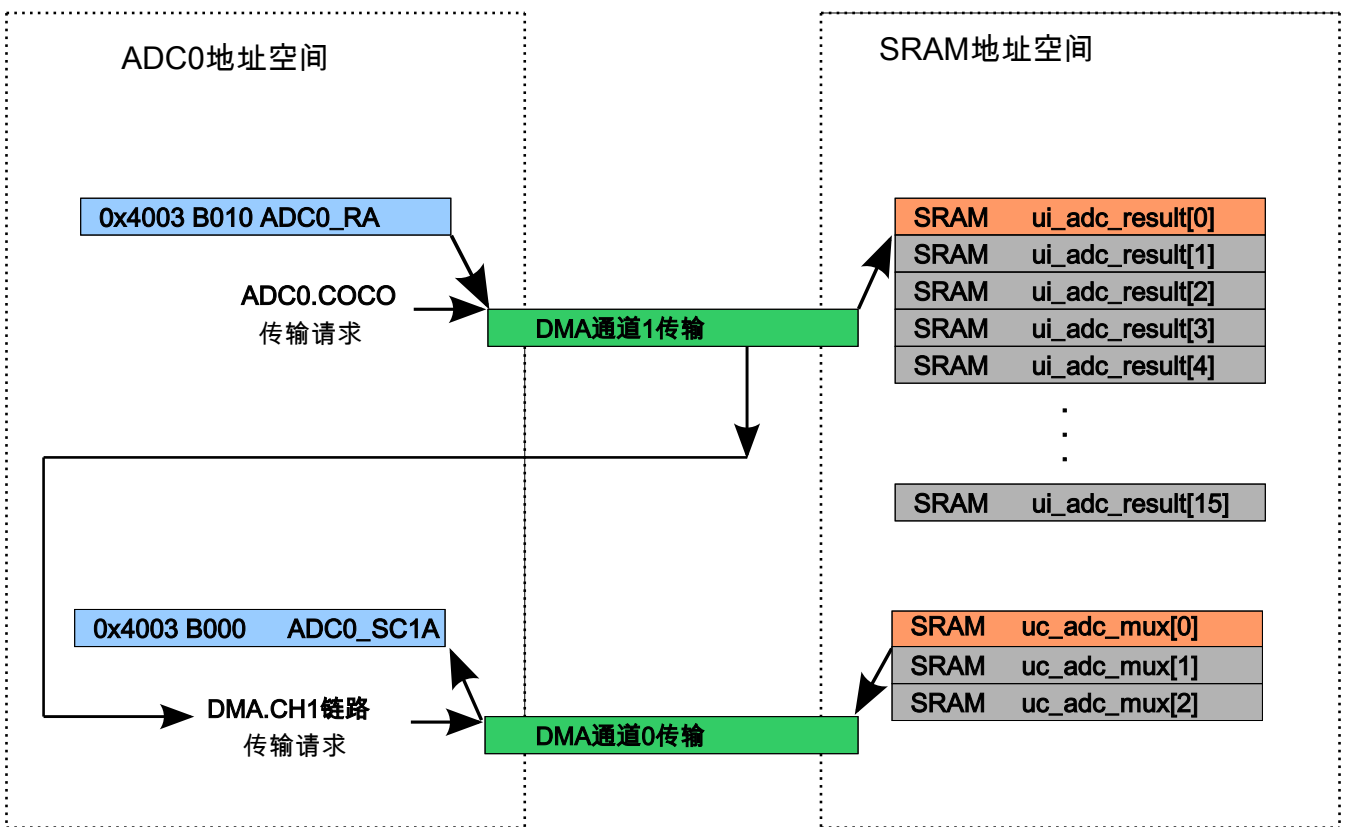

图 1. 外设和 SRAM 之间的数据流

2.3 DMA 特殊设置

2.3.1 通道优先级

DMA 通道优先级设置非常重要，因为在 ADC 软件触发模式下，写入 ADCn_SC1A 即可启动转换；因此，必须先读取结果数据，然后再写入下一个通道设置，因为写入 ADC 即会启动下一个转换。DMA 通道 1 用于将 ADC 结果数据传输至 SRAM 缓冲器。DMA 通道 0 用于更改 ADC 输入多路复用。

2.3.2 主循环和次循环链接

单次扫描模式（某些 ADC 通道仅需进行一次测量）应只使用次循环链接。如 DMA_TCDx_BITER_ELINKNO 和 DMA_TCDx_CITER_ELINKNO 寄存器中的定义。

持续扫描模式（某些 ADC 通道需要进行循环测量）需要使用主循环和次循环链接。如 DMA_TCDx_BITER_ELINKNO、DMA_TCDx_CITER_ELINKNO 和 DMA_TCD1_CSR 寄存器中的定义。持续扫描模式必须使用主循环和次循环链接，因为主循环完成后不会生成次循环完成请求。

3 示例

本示例使用 Freescale TRW-K60N512 开发板和 IAR Embedded Workbench®版本 6.30.1.3142 作为其测试环境。

3.1 示例流程

本应用笔记中提供的示例代码演示了三条 ADC 通道的持续扫描转换。

- 每条通道将被测量四次，因此，SRAM 结果缓冲器大小为 3×4=12（实际缓冲器大小为 16，以便演示仅写入 12 个数据字段）。
- ADC 在硬件触发器模式下工作，且 PDB 定时器用作触发源。
- 在持续模式下执行扫描；因此，主循环完成之后，将会重新加载结果缓冲器指针 address_0，并且从缓冲器起始地址再次启动转换。

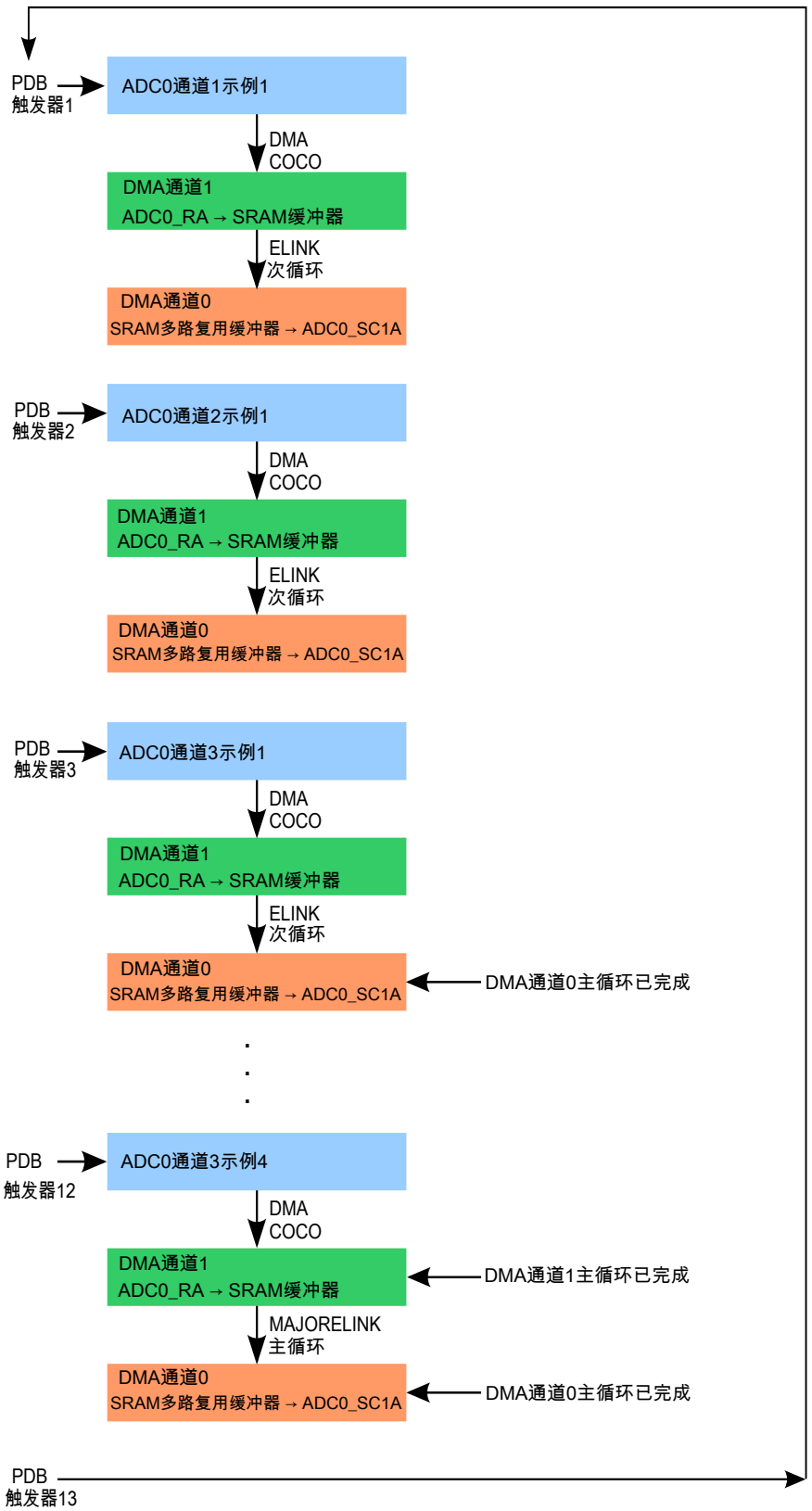

图 2. 示例流程

3.2 主文件中的函数

SIM_Init	初始化时钟和振荡器
FLL_Init	FLL 外设初始化
VREF_Init	电压基准初始化
PDBCH0TRG0_Init	PDB 通道 0 初始化，用于硬件触发器 ADC0
PDB_Init	通用 PDB 初始化
ADC_ExecCalib	ADC0 内部校准流程
ADC_Init	ADC0 初始化
DMACH0_CH1_init	DMA 通道 0 和通道 1 初始化

3.3 DMA 通道初始化

```
Channel 1 transfers ADC0 result data from ADC0_RA to SRAM buffer.

//*************************************************************
//**** DMA transfer request source - ADC0 conversion complete
//*************************************************************
DMAMUX_CHCFG1           = DMAMUX _CHCFG_ENBL|DMAMUX_CHCFG_SOURCE(28);
//*************************************************************
//**** Source address, ADC0_RA
//*************************************************************
DMA_TCD1_SADDR          = (uint32) &ADC0_RA;
//*************************************************************
//**** Source address increment; data is still read for the same address, no increment needed
//*************************************************************
DMA_TDC1_SOFF           = 0x00;

//*************************************************************
//**** Source address reload after major loop finishes, no reload needed
//*************************************************************
DMA_TDC1_SLAST          = 0x00;
//*************************************************************
//**** Destination address, SRAM buffer [0]
//*************************************************************
DMA_TDC1_DADDR          = (uint32)&ui_adc_result[0];
//*************************************************************
//**** Destination address increment in bytes, increment for next buffer address
//**** 16 bit => 2 bytes
//*************************************************************
DMA_TDC1_DOFF           = 0x02;
//*************************************************************
//**** Destination address reload after major loop finishes,
//**** must be subtracted from last pointer value, sample number is 12 each and 2 bytes long,
//**** 2 x 12 = 24 and must be subtract -24
//*************************************************************
DMA_TDC1_DLASTSGA       = -24;
//*************************************************************
//**** Number of bytes for minor loop (one data transfer), ADC0 result is 16 bits long, so
//**** 2-byte transfer
//*************************************************************
DMA_TDC1_NBYTES_MLO     = 0x02;
//*************************************************************
//**** Channel linking and major loop setting, linking after minor loop is enabled to
//**** channel 0 (0x0000), major loop transfers number 12 (0x0C)
```

示例

```
//*************************************************************
DMA_TDC1_BITER_ELONKNO  =  (DMA_BITER_ELINKNO_ELINK_MASK|0x0000|0x0C);

//*************************************************************
//**** Channel linking and major loop setting reload value after major loop finishes,
//**** linking after minor loop is enabled, major loop transfers number 12 (0x0C).
//*************************************************************
DMA_TDC1_CITER_ELONKNO  =  (DMA_CITER_ELINKNO_ELINK_MASK|0x0C);
//*************************************************************
//**** Source and destination data width specification, both source and destination is 16-bit
//*************************************************************
DMA_TDC1_ATTR           =  DMA_ATTR_SSIZE(1)| DMA_ATTR_SDIZE(1);
//*************************************************************
//**** Common channel setting, linking after major loop enable to channel 0,
//**** IRQ request is generated after major loop complete
//*************************************************************
DMA_TDC1_CSR            = (DMA_CSR_MAJORLINKCH(0)|DMA_CSR_MAJORLINCH_MASK|
                           DMA_CSR_INTMAJOR_MASK);


Channel 0 transfers next ADC0 input setting from constant buffer to ADC0_SC1A.

//*************************************************************
//**** DMA transfer request source - always requestor
//*************************************************************
DMAMUX_CHCFG0           = DMAMUX _CHCFG_ENBL|DMAMUX_CHCFG_SOURCE(36) ;
//*************************************************************
//**** Source address, constant buffer in SRAM
//*************************************************************
DMA_TCD1_SADDR          = (uint32) &uc_adc_mux[0];
//*************************************************************
//**** Source address increment, data is 8-bit, 1 byte
//*************************************************************
DMA_TDC1_SOFF           = 0x01;
//*************************************************************
//**** Source address reload after major loop finish, must be subtracted from last
//**** pointer value, sampling channel number is 3 each and 1 byte long, 1 × 3 = 3
//**** and must be subtract -3
//*************************************************************
DMA_TDC1_SLAST          = -3;
//*************************************************************
//**** Destination address, ADC0 control register
//*************************************************************
DMA_TDC1_DADDR          = (uint32)&ADC0_SC1A;
//*************************************************************
//**** Destination address increment in bytes, no increment needed
//*************************************************************
DMA_TDC1_DOFF           = 0x00;

//*************************************************************
//**** Destination address reload after major loop finish, no address reload needed
//*************************************************************
DMA_TDC1_DLASTSGA       = 0x00;

//*************************************************************
//**** Number of bytes for minor loop (one data transfer), ADC0 input setting value is
//**** 8 bits long, so 1-byte transfer
//*************************************************************
DMA_TDC1_NBYTES_MLO     = 0x01;

//*************************************************************
//**** Channel linking and major loop setting, no linking after minor loop,
//**** major loop transfers number 0x03
//*************************************************************
DMA_TDC1_BITER_ELONKNO  = (DMA_BITER_ELINKNO_ELINK_MASK|0x0000|0x0C);

//*************************************************************
//**** Channel linking and major loop setting reload value after major loop finish,
//**** no linking after minor loop, major loop transfers number 0x03
//*************************************************************
```

```
DMA_TDC1_CITER_ELONKNO  = (DMA_CITER_ELINKNO_ELINK_MASK|0x0C);

//*************************************************************
//**** Source and destination data width specification, both source and destination are 8-bit
//*************************************************************
DMA_TDC1_ATTR           = DMA_ATTR_SSIZE(1)| DMA_ATTR_SDIZE(1);

//*************************************************************
//**** Common channel setting, no linking after major loop, no IRQ request enable
//*************************************************************
DMA_TDC1_CSR            = 0x00;
```

How to Reach Us:

Home Page:
freescale.com

Web Support:
freescale.com/support

Document Number AN4590
Revision 0, 9/2012