# 基于SDK ADC-DMA多通道采样-LPC51U68

DAWEI

CAS

# 基于SDK ADC-DMA多通道采样-LPC51U68

LPC系统的ADC具有按顺序转换多通道ADC的功能，并且具有两个转换序列，SeqA&SeqB。对于需要多通道转换的场合比较适用。

当然也可以使用DMA传输多通道ADC转换值，使软件接口更简单，或者面向某些特殊的应用。

# 基于SDK ADC-DMA多通道采样-LPC51U68

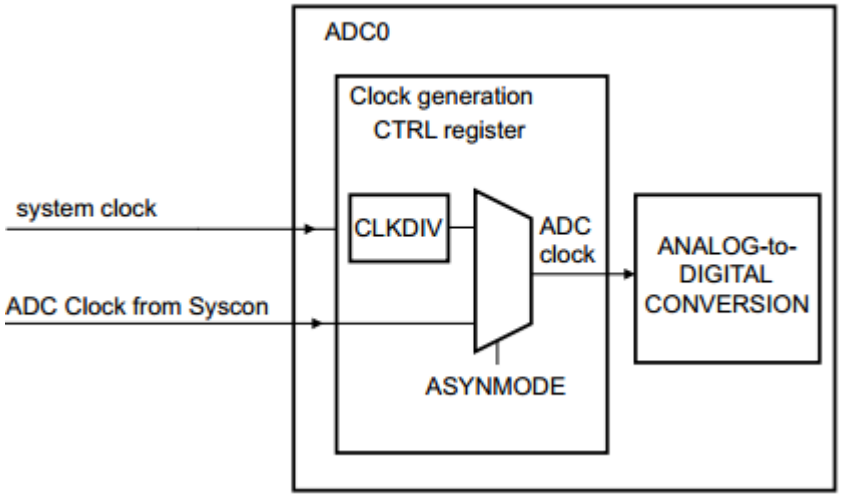
## 基本概念

# 基于SDK ADC-DMA多通道采样-LPC51U68 ADC

- LPC51U68有两个转换序列ADC0_SEQA & ADC0_SEQB
- ADC时钟可以选择系统时钟，也可以选择其他异步的可选时钟。最大时钟不超过80MHz.
- 带比较功能
- ADC转换可以软件触发，也可以硬件触发。硬件触发源有4个。定时采样选择SCT0_OUT7
- 采样时间：12-bit 精度下.

An ADC conversion requires 15 ADC clocks, plus any clocks specified by the TSAMP=0 field. 这样最高采样率为80/15=5.33Msps。但是实际上考虑到采保时间，应该设置的更低。

- BURST模式：可以使ADC持续进行多通道采样
- SINGLESTEP：选择一次触发只转换ADC通道列表中的一个通道，还是全部转换序列中的所有通道；
- MODE：转换完成中断是指单个通道完成还是整个序列完成
- 结果寄存器能够看到比较结果和OverRun的状态




## 28.5 General description

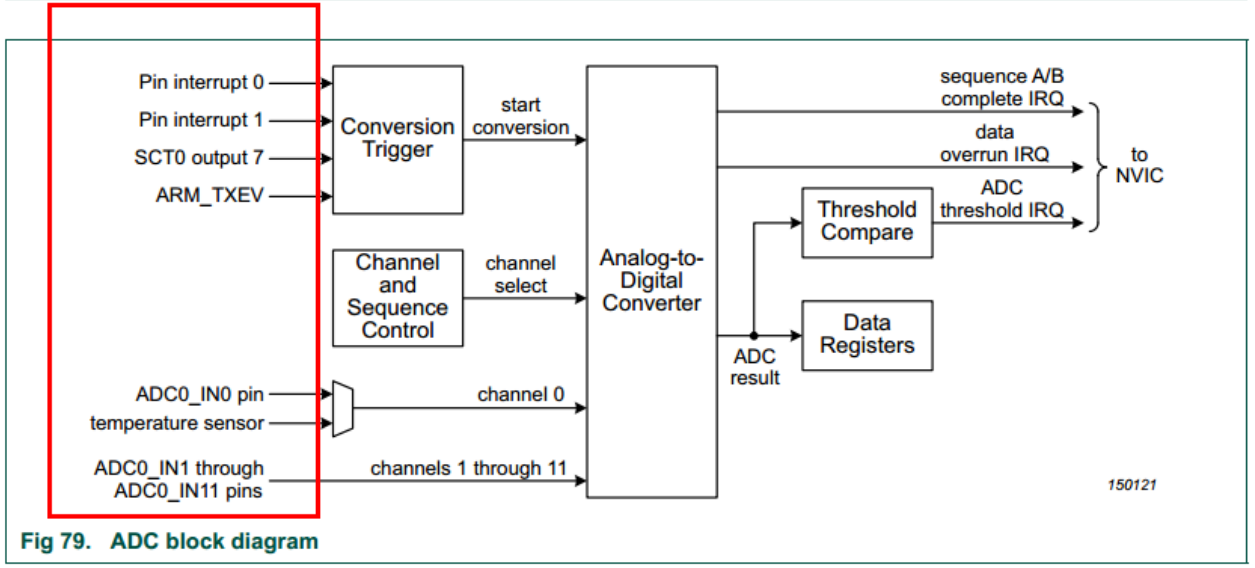



Fig 79. ADC block diagram

# 基于SDK ADC-DMA多通道采样-LPC51U68 INPUT MUX

- INPUT MUX能够给DMA产生Trigger
- Trigger的source可以给每个DMA Channel（共18个）单独配置。
- 比如ADC SeqA/B都可以产生DMA trigger
- 甚至从其他DMA的完成事件产生Trigger

### 10.5.2 DMA trigger input multiplexing

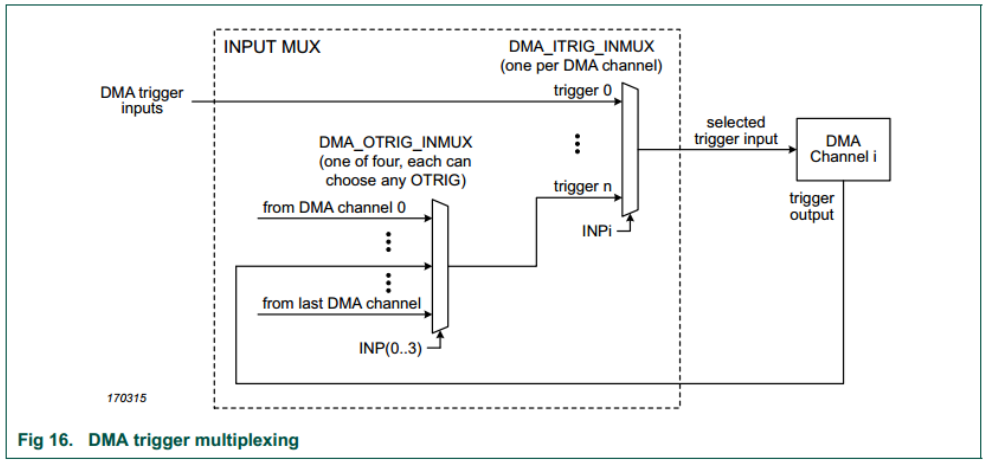



Fig 16. DMA trigger multiplexing

Table 198. DMA trigger Input mux registers (DMA_ITRIG_I

| Bit | Symbol | Description |
|---|---|---|
| 4:0 | INP | Trigger input number (decimal value) for DM<br>0 = ADC0 Sequence A interrupt<br>1 = ADC0 Sequence B interrupt<br>2 = SCT0 DMA request 0<br>3 = SCT0 DMA request 1<br>4 = Timer CTIMER0 Match 0<br>5 = Timer CTIMER0 Match 1<br>6 = Timer CTIMER1 Match 0<br>7 = RESERVED<br>8 = RESERVED<br>9 = Timer CTIMER3 Match 0<br>10 = RESERVED<br>11 = RESERVED<br>12 = Pin interrupt 0<br>13 = Pin interrupt 1<br>14 = Pin interrupt 2<br>15 = Pin interrupt 3<br>16 = DMA output trigger mux 0<br>17 = DMA output trigger mux 1<br>18 = DMA output trigger mux 2<br>19 = DMA output trigger mux 3 |
| 31:5 | - | Reserved. |

# 基于SDK ADC-DMA多通道采样-LPC51U68 DMA

- DMA的触发既可以通过Trigger也可以通过Request
- Request比如Flexcomm通信接口寄存器
- ADC Seq属于Trigger，因为当Seq产生的时候，具体读取哪个数据值需要自定义

- DMA的关键信息包括：

1.多个DMA通道的Channel配置字存放在内存中；通过SRAM base寄存器指定

2.配置字可以在自己工作完成后link到下一个配置字

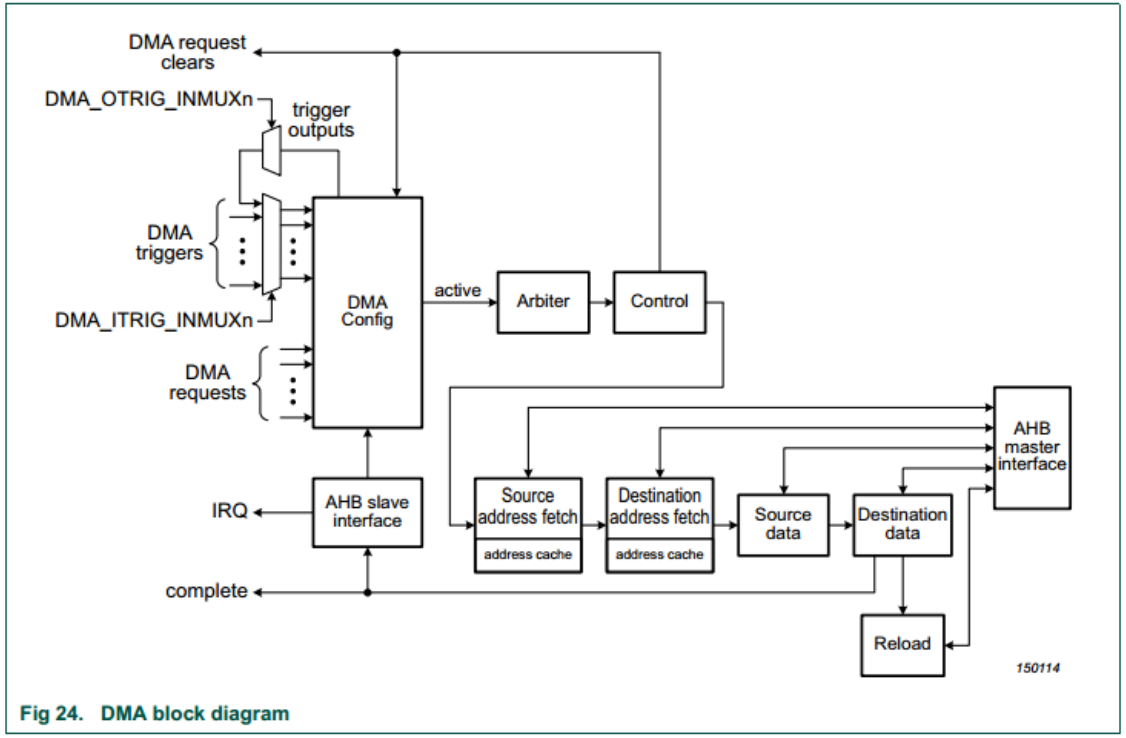



Fig 24. DMA block diagram

Table 245. Channel descriptor map

| Descriptor | Table offset |
|---|---|
| Channel descriptor for DMA channel 0 | 0x000 |
| Channel descriptor for DMA channel 1 | 0x010 |
| Channel descriptor for DMA channel 2 | 0x020 |
| Channel descriptor for DMA channel 3 | 0x030 |
| Channel descriptor for DMA channel 4 | 0x040 |
| Channel descriptor for DMA channel 5 | 0x050 |
| Channel descriptor for DMA channel 6 | 0x060 |
| Channel descriptor for DMA channel 7 | 0x070 |
| Channel descriptor for DMA channel 8 | 0x080 |
| Channel descriptor for DMA channel 9 | 0x090 |
| Channel descriptor for DMA channel 10 | 0x0A0 |
| Channel descriptor for DMA channel 11 | 0x0B0 |
| Channel descriptor for DMA channel 12 | 0x0C0 |
| Channel descriptor for DMA channel 13 | 0x0D0 |
| Channel descriptor for DMA channel 14 | 0x0E0 |

Table 238: Reload descriptors

| Offset | Description |
|---|---|
| + 0x0 | Transfer configuration. |
| + 0x4 | Source end address. This points to the address of the last entry of the source address range if the address is incremented. The address to be used in the transfer is calculated from the end address, data width, and transfer size. |
| + 0x8 | Destination end address. This points to the address of the last entry of the destination address range if the address is incremented. The address to be used in the transfer is calculated from the end address, data width, and transfer size. |
| + 0xC | Link to next descriptor. If used, this address must be aligned to a multiple of 16 bytes (i.e., the size of a descriptor). |

14.5.3 Single buffer

# 基于SDK ADC-DMA多通道采样-LPC51U68 DMA

- 乒乓模式：因为LPC的DMA并没有重载自身的功能，如果在一个配置字工作完成后，还想继续重新配置这个DMA通道，完成同样的功能。需要使用到乒乓模式。配置字A link到配置字B，先工作；传输完成后自动配置为配置字B，B link到配置字A，B的工作完成后又恢复到配置字A。这样就完成了重复加载的工作。无需软件干预。

### 14.5.4 Ping-Pong

乒乓模式，即可以将两个Channel描述符互相link起来，轮流完成，并在完成后可以产生中断

Ping-pong is a special case of a linked transfer. It is described separately because it is typically used more frequently than more complicated versions of linked transfers.

A ping-pong transfer uses two buffers alternately. At any one time, one buffer is being loaded or unloaded by DMA operations. The other buffer has the opposite operation being handled by software, readying the buffer for use when the buffer currently being used by the DMA controller is full or empty. Table 240 shows an example of descriptors for ping-pong from a peripheral to two buffers in memory.

Two reload descriptors are used in additional the Channel Descriptor due to the fact that the Channel Descriptor memory locations are actively used during DMA operation of the related DMA channel (as previously noted). In this example, Descriptor A is typically just a copy of the original Channel Descriptor.

**Table 240: Example descriptors for ping-pong operation: peripheral to buffer**

| Channel Descriptor | Descriptor B | Descriptor A |
|---|---|---|
| + 0x0 (not used) | + 0x0 Buffer B transfer configuration | + 0x0 Buffer A transfer configuration |
| + 0x4 Peripheral data end address | + 0x4 Peripheral data end address | + 0x4 Peripheral data end address |
| + 0x8 Buffer A memory end address | + 0x8 Buffer B memory end address | + 0x8 Buffer A memory end address |
| + 0xC Address of descriptor B | + 0xC Address of descriptor A | + 0xC Address of descriptor B |

# 基于SDK ADC-DMA多通道采样-LPC51U68 DMA

- **Interleaved transfers**

**实际上就是源地址和目的地址的累加，并不一定要按照每次传输的宽度累加，可以跳过一定的范围。**
类似源地址/目的地址增加长度，最大为Width的4倍。

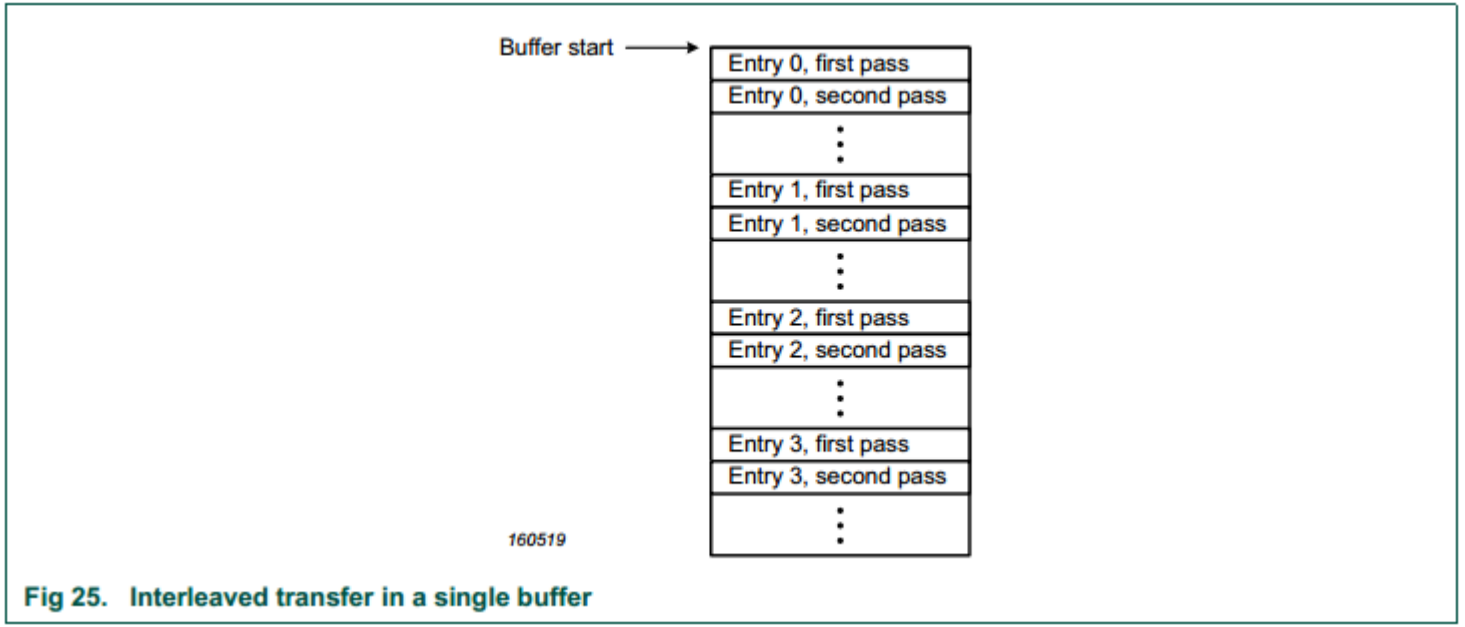



**Fig 25. Interleaved transfer in a single buffer**

# 基于SDK ADC-DMA多通道采样-LPC51U68 DMA

- **Link模式**

不止两个通道描述符，更多的描述符可以连接在一起

- **Channel chaining模式：将不同的通道chain在一起**

### 14.5.6 Linked transfers (linked list)

Link模式是乒乓模式的更多种链接

A linked transfer can use any number of descriptors to define a complicated transfer. This can be configured such that a single transfer, a portion of a transfer, one whole descriptor, or an entire structure of links can be initiated by a single DMA request or trigger.

An example of a linked transfer could start out like the example for a ping-pong transfer (Table 240). The difference would be that descriptor B would not link back to descriptor A, but would continue on to another different descriptor. This could continue as long as desired, and can be ended anywhere, or linked back to any point to repeat a sequence of descriptors. Of course, any descriptor not currently in use can be altered by software as well.

Ch X也有自动reload的功能，在前一个配置（完成）耗尽时，能够主动加载自身

To use channel chaining, first configure DMA channels x and y as if no channel chaining would be used. Then:

- For channel x:
  - If channel x is configured to auto reload the descriptor on exhausting of the descriptor (bit RELOAD in the transfer configuration of the descriptor is set), then enable 'clear trigger on descriptor exhausted' by setting bit CLRTRIG in the channel's transfer configuration in the descriptor.
- For channel y:
  - Configure the input trigger input mux register (DMA_ITRIG_INMUX[0:21]) for channel y to use any of the available DMA trigger muxes (DMA trigger mux 0/1).
  - Configure the chosen DMA trigger mux to select DMA channel x.
  - Enable hardware triggering by setting bit HWTRIGEN in the channel configuration register.
  - Set the trigger type to edge sensitive by clearing bit TRIGTYPE in the channel configuration register.
  - Configure the trigger edge to falling edge by clearing bit TRIGPOL in the channel configuration register.

Note that after completion of channel x the descriptor may be reloaded (if configured so), but remains un-triggered. To configure the chain to auto-trigger itself, setup channels x and y for channel chaining as described above. In addition to that:

Channel chain不同于link，他是不同channel间的链接，在一个通道工作完成后，触发下一个通道的工作

### Address alignment for data transfers

Transfers of 16 bit width require an address alignment to a multiple of 2 bytes. Transfers of 32 bit width require an address alignment to a multiple of 4 bytes. Transfers of 8 bit width can be at any address.

### 14.5.8 Channel chaining

Channel chaining is a feature which allows completion of a DMA transfer on channel x to trigger a DMA transfer on channel y. This feature can for example be used to have DMA channel x reading n bytes from UART to memory, and then have DMA channel y transferring the received bytes to the CRC engine, without any action required from the ARM core.

# LPC_MULTI-CHANNELS_ADC_DMA_SW_TRG

# 软件触发ADC-DMA多通道采样

Overview-lpc_multi-channels_adc_dma_sw_trg LPC的ADC要做多通道软件触发的转换，应该是比较容易的，其ADC本身就带有多通道转换功能。 这里的demo主要采用DMA的方式，完成6个ADC通道的转换，并把数据copy到RAM buffer中。使用中断也很容易实现，设置SeqA中的通道号，使能sequence转换，在sequence中断中读取对应的ADC结果寄存器；这边不单独列出。

1.采用DMA方式，如果设置sequence全部转换完成后触发DMA传输，ADC的结果寄存器并不一定连续，需要在硬件上将此6个ADC通道排成连续的。这边采用SeqA mode选用End of conversion产生DMA触发。

2.每次ADC有一个通道转换完成后，产生DMA触发，从 ADC Global Data register A取得最新数据。

3.DMA应该设置为TRIGBURST=1模式，BURSTPOWER设置为1次，

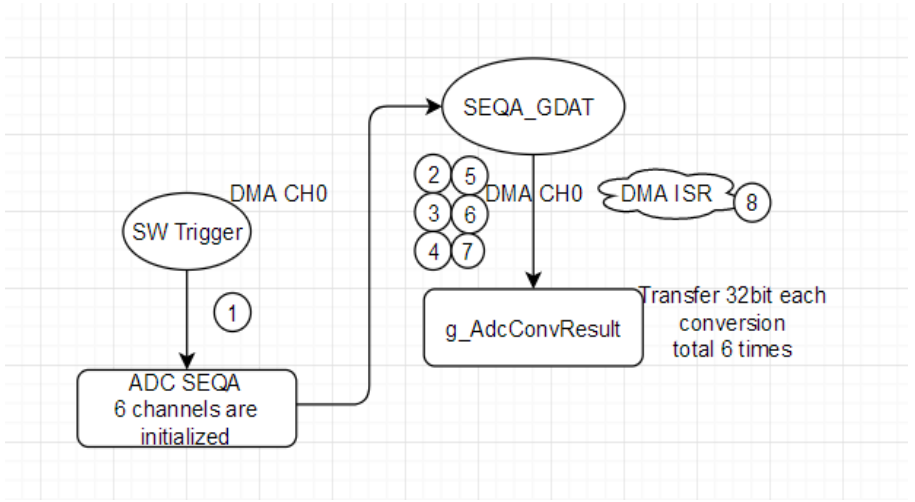
xfer cfg中的width设置为4字节，即每一次触发读取ADC Global Data register A到ram buffer即可。

源地址interleave不增加，目的地址interleave每次增加Width长度，一次排放转换结果。

另外传输总长度为6*4字节，按照手册，总长度必须为BURSTPOWER的偶数倍；

Source地址wrap，每次BURSTPOWER传输完回复到原位

Destination 地址不需要Wrap，依次累加。




程序执行顺序:

1.发起软件触发;

2,3,4,5,6,7,8都是每个ADC通道转换完成后触发conversion DMAtrigger，触发DMA CH0传输GDAT中最新的转换结果到RAM数组。 TRIGBURST=1, BURSTPOWER=1设置每次trigger只转换一次 single transfer, 总的转换长度由Xfer count确定,即6次之后产生中断。

# LPC_MULTI-CHANNELS_ADC_DMA_HW_TRG

# 硬件触发ADC-DMA多通道采样

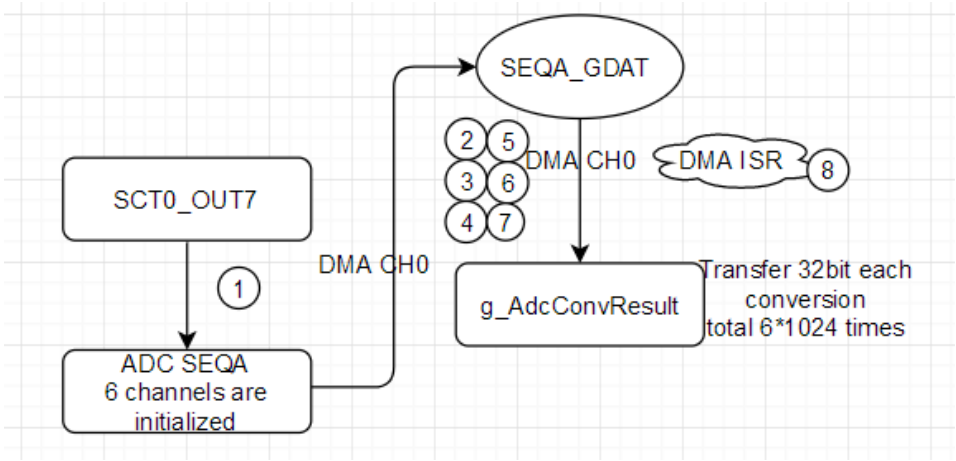
Overview-lpc_multi-channels_adc_dma_hw_trg

如果有例如以下需求：以1KHz的频率触发多个通道，比如6个，连续采样1024个采样点之后产生中断。

1.大部分设置如软件触发模式。首先应该使用SCT配置为1KHz模式，触发SeqA序列；

2.DMA设置应该设置传输总长度为6*1024*4字节，产生中断，并且源地址wrap（ADC Global Data register A 地址不增加），目的地址以Width长度累次增加。

3.如果ADC产生中断速度太快，DMA来不及读走，可能会产生overrun，需要在同步模式下适当降低ADC clock,确保DMA能够来得及搬走ADC Global Data register A中的数据。




```
* DMA descripter table used for ping-pong mode. */
MA_ALLOCATE_LINK_DESCRIPTORS(s_dma_table, DMA_DESCRIPTOR_NUM);
onst uint32_t g_XferConfig =
    DMA_CHANNEL_XFER(true,                          /* Reload link descriptor after current exhaust, */
                     false,                          /* 不主动clear trigger, 如果link到下一个descriptor的话，自动start */
                     true,                          /* Enable interruptA. */
                     false,                         /* Not enable interruptB. */
                     sizeof(uint32_t),              /* Dma transfer width.每次传输32bit */
                     kDMA_AddressInterleave0xWidth, /* Dma source address no interleave, 源地址不变*/
                     kDMA_AddressInterleave1xWidth, /* Dma destination address 每次都要增加*/
                     sizeof(uint32_t)*ADC_CH_NUM*SAMPLE_POINTS /* Dma transfer byte. 总长度是6个ADC通道*需要总计的SAMPLE_POINTS采样点数*/

                     );
```

在这个例程中，需要注意的几点设置，包括：

1.设置DEMO_ADC_CLOCK_DIVIDER 2，太高的ADC转换率会导致overrun

2.采用乒乓模式，设置两个传输描述符，在一个描述符耗尽的时候，自动link到下一个描述符，无缝切换。

3，描述符设置中，当描述符耗尽时，不主动clear trigger,这样在切换到下一描述符时，无需手动enable这个channel，保持这个channel为enable状态。

4.SCT_OUT7输出设置为Toggle模式，所以实际trigger频率为二分之一的SCT_OUT频率。因为只有上升沿或者下降沿能够触发。

SECURE CONNECTIONS
FOR A SMARTER WORLD