Freescale Semiconductor, Inc.
Application Note

Document Number: AN5222
Rev. 0, 12/2015

16-bit SAR ADC and DMA Operation Using MC56F84xxx

ADC interrupt and DMA transfer

By: Matus Plachy

1 Introduction

The DMA module enables transferring the result of the ADC conversion from the ADC result register to an application variable in the RAM without core intervention, saving some CPU machine cycles.
This application note describes the peripherals' limits and the correct approach for using DMA transfer when the Conversion Complete flag is set at the end of the AD conversion. It is expected that you have some experience with the DMA module, because not all details about the operation are described in this document. See Section 5, "References" for additional documentation.

Contents

freescale™

2 ADC interrupt and DMA transfer at the same time

There are several reasons to perform the DMA transfer and ADC interrupt at the same time at the end of the AD conversion; for example, to save some CPU machine cycles while moving the AD conversion result to the variable without the core intervention, or to perform the algorithm calculation in two separate interrupts in a single PWM period. The 16-bit SAR ADC has only one COCO (Conversion Complete) flag. The DMA controller has a higher priority on the Internal Peripheral Bus, which causes conflict resulting in erratic generation of the ADC interrupt. In a real-time control application, this represents a randomly occurring hazardous state which is difficult to debug.

CAUTION

Do not perform the ADC interrupt and the DMA transfer request on the same COCO flag. The following section describes the correct approach.

3 Software examples of real-time control application using DMA transfer of ADC result

This section describes two software examples of the DMA transfer use in a real-time control application:

- Performing two transfers of the ADC result and generating the interrupt request at the end of the second transfer, where the control algorithm is calculated. See Figure 2.
- Performing two transfers of the ADC result and generating the interrupt request after each transfer. The calculation of the control algorithm is split into two parts (interrupts). See Figure 3.

The software examples supporting this application note are developed in CodeWarrior 10.5.
The peripherals' configuration (as well as all peripherals' accesses in the code) is performed using Quick Start development tool. The TWR-56F8400 Tower System modular development platform is used as the target hardware.

3.1 Generation of ADC triggering signals

In both cases, the start of the conversion is triggered by the eFlexPWM module. Besides generating the PWM switching signals for up to four pairs of transistor switches, it generates multiple timing and triggering signals you can combine in the inter-peripheral crossbar switch XBAR B (with AND, OR, and INVERT functionality) then connect to the ADC via XBAR A, as shown in the following figure. The trigger signals are generated when a compare event for the VAL0 and VAL5 value registers of the eFlexPWM module occurs.

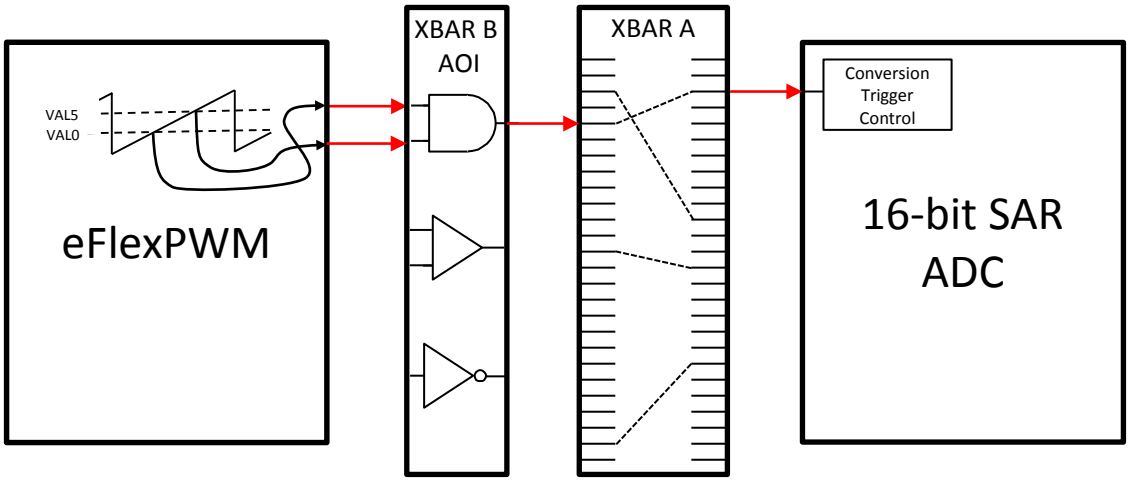

Figure 1. Connection of ADC triggering signals

3.2 Example 1—generating interrupt after sequence of samples

Figure 2 shows a case when the result of the AD conversion is transferred by the DMA0 channel from the ADC result register to a variable in RAM after the first triggering event. After the transfer, the DMA0 channel creates a DMA request for the DMA1 channel (linked operation). The DMA1 channel performs the transfer of a new configuration of the ADC status and control register from the variable stored in RAM into the ADC_SC1 register (updates the number of analog input channel). The DMA0 is configured to automatically increase the destination address after the transfer, so the result of the next AD conversion is placed into the consequent memory location in RAM. The total number of bytes to copy into the buffer in RAM (BCR bit filled in the DMA_DCR_BCR0 register) decreases.

The DMA controller then waits for the next COCO flag, which signals that a new sample is ready to be transferred. The procedure is the same as the one described above. Because the number of bytes left is decreased to zero, the DMA0 channel generates an interrupt request. The control algorithm is calculated in the interrupt service routine using the sampled analog values, and the DMA and ADC modules are reconfigured for the next sequence:

- The destination address is changed to the first member of the RAM buffer.
- The number of bytes that have to be transferred is updated; DMA0 is configured to four bytes (2 × transfer of the 16-bit ADCR register) and DMA1 is configured to eight bytes (2 × transfer of the 32-bit ADC_SC1 register).
- The analog channel input is updated.

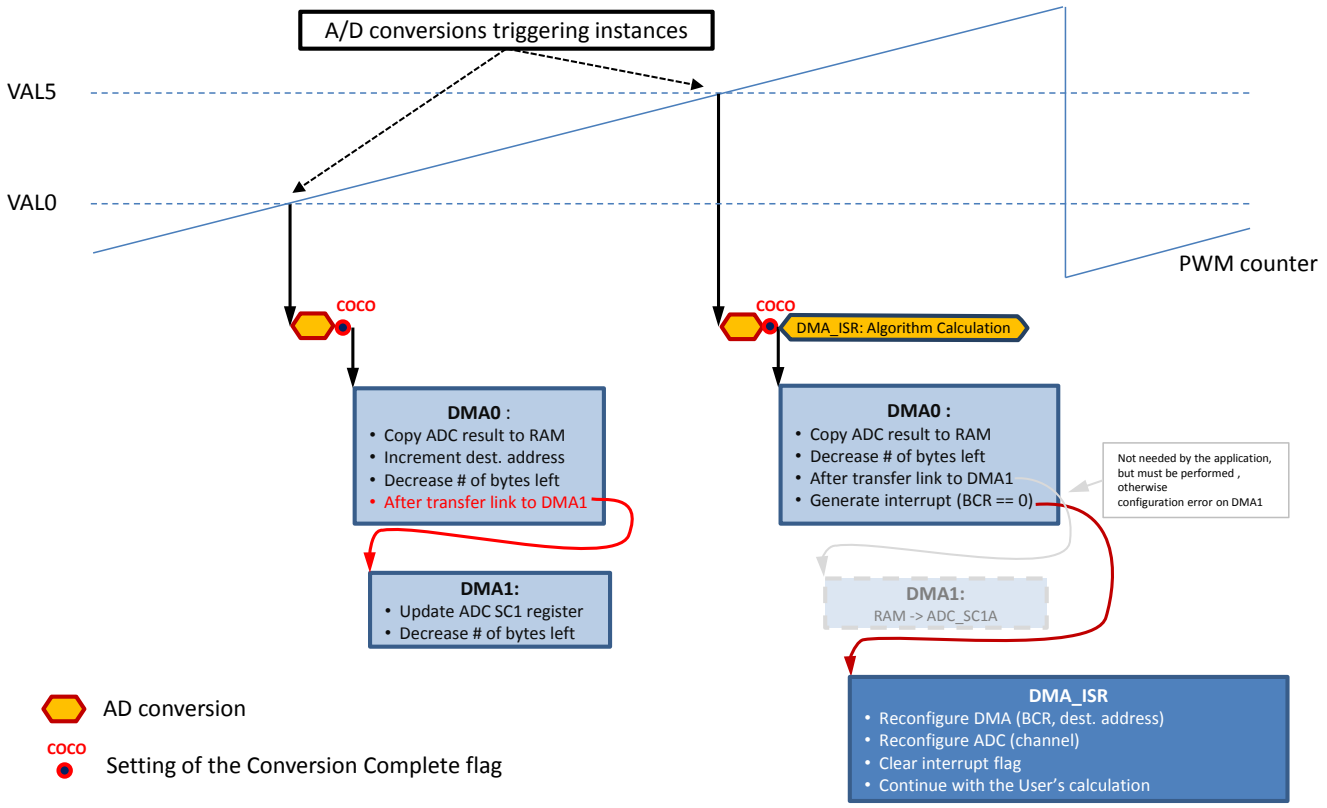

Figure 2. DMA transfer with interrupt at the end of the sequence of AD conversions

3.3 Example 2—generating interrupt after each sample

The second example demonstrates the situation where you must add one more interrupt service routine for any reason.

The first difference is the total number of bytes to be transferred. In this case, it is set to only two bytes, so the DMA0 performs only one 16-bit transfer of the AD conversion result. The linked-channels feature is used (as in the previous case), so that the DMA1 channel updates the ADC Status and Control registers with a new channel and enables the ADC interrupt request.

After the DMA1 channel transfers the new configuration to the ADC_SC1 register, it generates the interrupt request. The first part of the control algorithm is calculated in this interrupt.

The DMA0 channel is configured to automatically disable the peripheral request after the transfer, so it remains idle when the COCO flag is set again. When the next result of AD conversion is ready, the ADC interrupt request is generated. In the interrupt service routine, the result of the AD conversion is read and the next part of the control algorithm is executed. After that, the DMA and ADC modules are reconfigured:

- The total number of bytes to be transferred for DMA0 and DMA1 channels is set; DMA0 is set to two bytes (single transfer of a 16-bit ADCR register) and DMA1 is set to four bytes (single transfer of a 32-bit ADC_SC1 register).
- The peripheral request for DMA0 is enabled.
- The ADC input channel is updated and the ADC interrupt request is disabled.

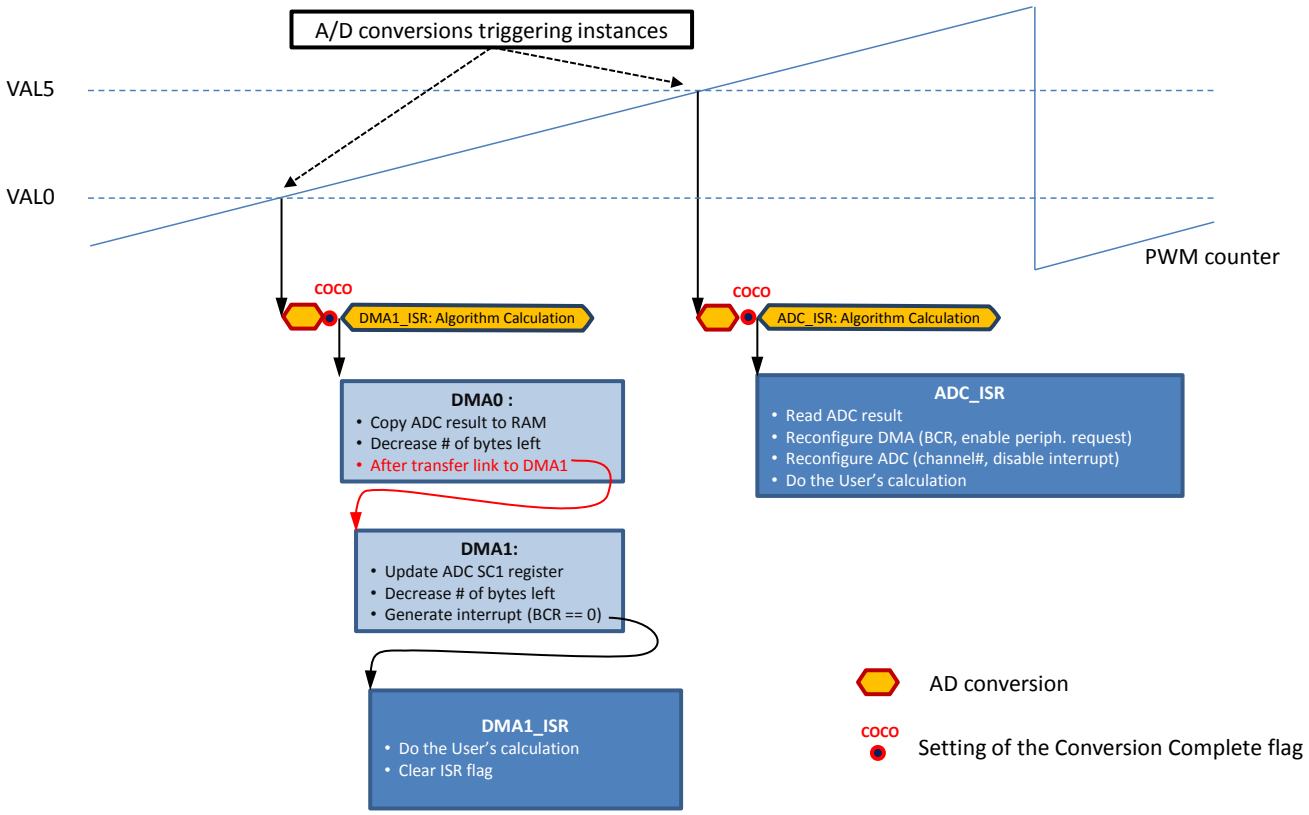

Figure 3. Interrupt generated after each transfer

3.4 Software description

Each example is supplied in a separate CodeWarrior project. The initial configuration of the core and all peripherals is done in the Graphical Configuration Tool, which is part of the DSC56800EX Quick Start development environment. Both complete example codes are placed in the *main.c* file of the particular project. This document does not provide a detailed description of the software, but the code is well-commented and the *ioctl* commands used for accessing the peripheral registers are more or less self-explanatory. Each software example contains FreeMASTER project *FmstrProject.pmp*, located in the *<Project_Directory>/FreeMASTER/* folder. FreeMASTER enables the visualization of sampled analog quantities.

4 Conclusion

This application note describes the hazardous state of an improper use of the ADC interrupt while a concurrent transfer of the AD conversion result is being performed by the DMA. It describes the proper approach using two example cases. The software examples are provided for quick evaluation and easy implementation into your project.

5 References

- *MC56F8458x Reference Manual* (document MC56F8458XRM)
- *Using DMA Transfers with Enhanced Flexible PWM on MC56F84xxx* (document AN4598)
- *DSC56800EX Quick Start User Guide* (document DSC56800EXQSUG)

6 Acronyms and abbreviations

This table lists the acronyms and abbreviated terms used in this application note:

Table 1. Acronyms and abbreviations

Acronym	Definition
AD	Analog-to-Digital
ADC	Analog-to-Digital Converter/Conversion
COCO	Conversion Complete flag
DMA	Direct Memory Access
ISR	Interrupt Service Routine
eFlex	enhanced Flexible
PWM	Pulse-Width Modulation
RAM	Random Access Memory
SAR	Successive Approximation Register—a type of the AD converter circuit
XBAR	Cross Bar Switch

7 Revision history

Table 2. Revision history

Revision number	Date	Substantive changes
0	12/2015	Initial release

How to Reach Us:

Home Page:
freescale.com

Web Support:
freescale.com/support

Document Number: AN5222
Rev. 0
12/2015