Freescale Semiconductor
Application Note

Document Number:AN4587
Rev 0, 08/2013

ADC FIFO Application on the S08P Family

by: Wang Peng

Contents

1 Introduction

This application note serves as a guide for design engineer on how to use FIFO of ADC module on S08P family with demo code provided. There are many important features in S08P family, one of which supports ADC FIFO. This means that an analog input channel conversion result can be saved in the FIFO, and when all channel conversions in the input FIFO are done, it sets flag or generates interrupt. This is very useful for the applications which use many ADC channels and frequently process ADC conversion. The ADC module contains two FIFOs, analog input channel FIFO and conversion result FIFO:

- Analog input channel FIFO is used to buffer analog input channels. The analog input channel FIFO is accessed by ADCH bits in ADC_SC1, when FIFO function is enabled. The analog channel must be written to this FIFO in order.
- Conversion result FIFO is used to store analog results. The result FIFO is accessed by ADC_RH:ADC_RL registers, when FIFO function is enabled. The result must be read via these two registers in the same order of analog input channel FIFO to get the proper results.

The following figure describes the structure of FIFO:

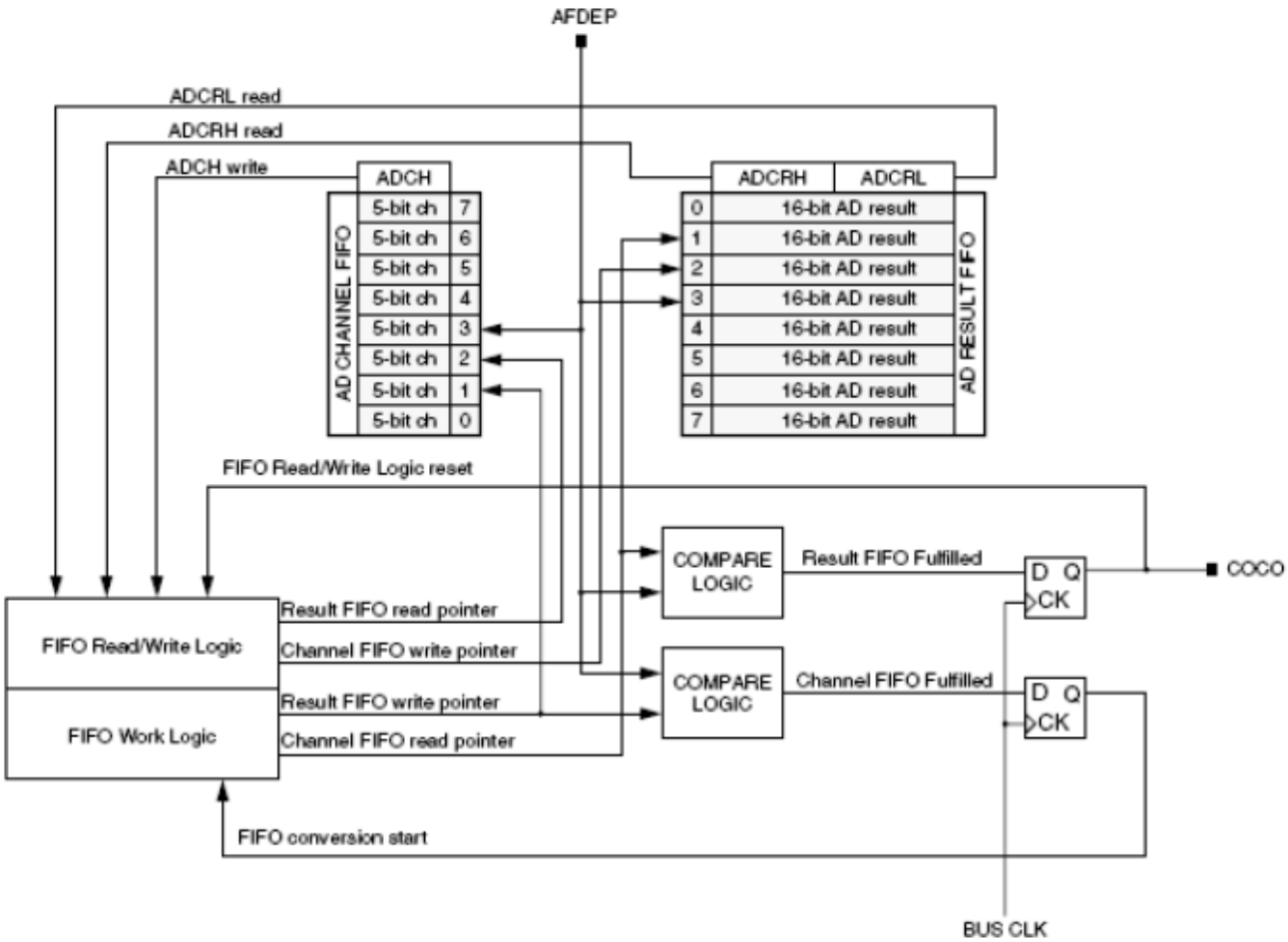

Figure 1. ADC FIFO structure

2 Application

This module supports up to 8-deep result FIFO and channel FIFO with selectable FIFO depth. Through register AFDEP, user can flexibly set FIFO depth depending on different requirements, and also select software or hardware trigger to start conversion. In some cases, if required to determine if the input analog channel is greater or less than the value specified. To do so, enable compare function and FIFO to select ‘AND’ or ‘OR’ to check the analog input state specified by the channel FIFO. When condition is true, it will set COCO flag, and if interrupt is enabled, it will generate interrupt.

2.1 Initialization

An initialization procedure must be performed before the ADC module can be used to start conversions with FIFO. A typical sequence of initialization procedure is as follows:

1. Update the configuration register (ADC_SC3) to select the input clock source and divide ratio used to generate the internal clock, ADCK. This register is also used for selecting sample time and low-power configuration.
2. Update the configuration register (ADC_SC4) to select the FIFO scan mode, FIFO compare function selection (OR or AND function), and FIFO depth.
3. Update status and control register 2 (ADC_SC2) to select the conversion trigger (hardware or software), compare function options if enabled.

4. Update status and control register 1 (ADC_SC1) to select whether conversions will be continuous or completed only once, and to enable or disable conversion complete interrupts. The input channel on which conversion will be performed is also selected here. The initial value of the input channel field ADCH can be 0b11111 which disables ADC module.

Example 1, ADC FIFO initialization is as follows:

```
Void ADC_Init( void )
{
/* The following code segment demostrates how to initialize ADC by long
sample time, bus frequency, and 12 bit mode,software triggerd from AD0, AD1, AD2, and AD3
external pins with 4-level FIFO enabled,and enable interrupt */
ADC_APCTL1 = 0x0f;
ADC_APCTL2 = 0x00;
ADC_SC3_ADIV = ADC_ADIV_DIVIDE_4;
ADC_SC3_MODE = ADC_MODE_12BIT;
ADC_SC3_ADLSMP = 1; // enable long sample
ADC_SC3_ADICLK = CLOCK_SOURCE_BUS_CLOCK;
ADC_SC4_AFDEP = 0x03; // FIFO level 4
ADC_SC1_AIEN = 1; // enable interrupt
}
```

2.2 Software trigger

When ADC module is enabled as software trigger, write to ADCH will fill one ADC channel to channel FIFO. Once fulfilled the channel FIFO on the level specified by the AFDEP, ADC conversion will be started immediately. Conversion result will be stored in the result FIFO, it is accessed by reading ADC_RH:ADC_RL registers, after conversion is complete, it will set COCO flag, if enable interrupt, and also generate an interrupt request. The following flow chart explains the software trigger process:

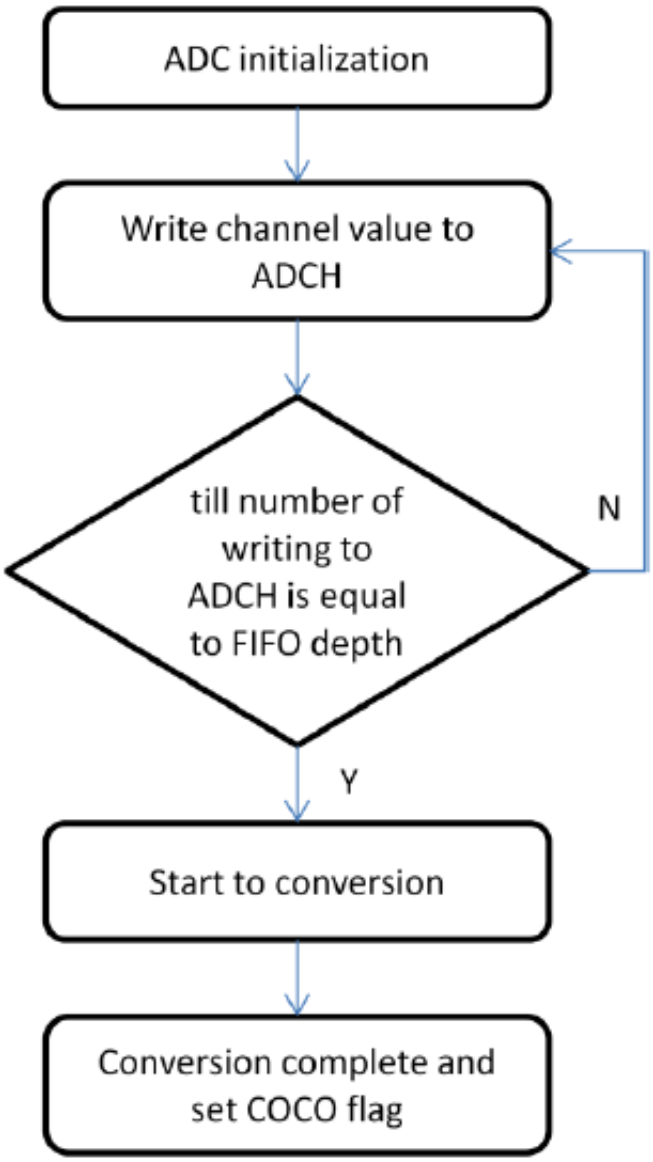

Figure 2. Software trigger process

If the number of writing to register ADCH is less than depth specified by register AEFDEP, it does not start conversion until the channel FIFO is full. When writing channel value to ADCH is continued after ADC starts conversion and before COCO flag is set, it will abort the current conversion. Selecting the continues mode after conversion completes will initialize a new conversion. The following figure describes the difference between single and continuous conversion:

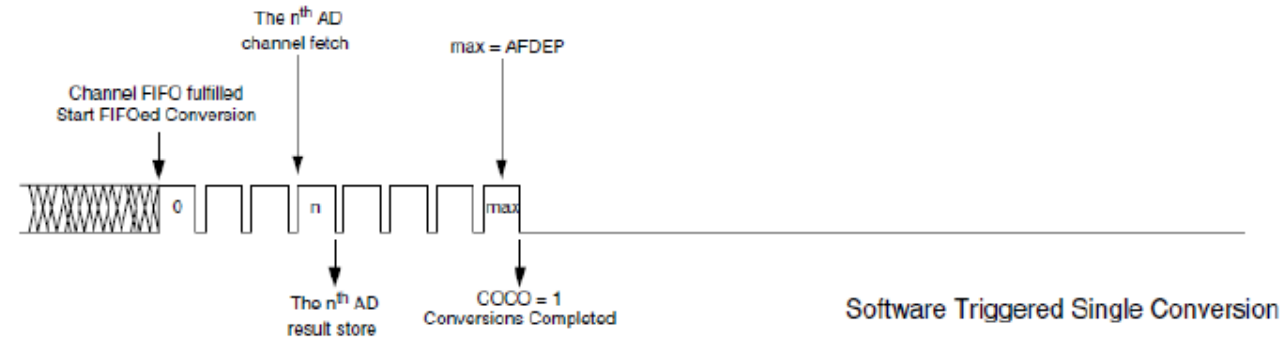

Figure 3. Software triggered single conversion

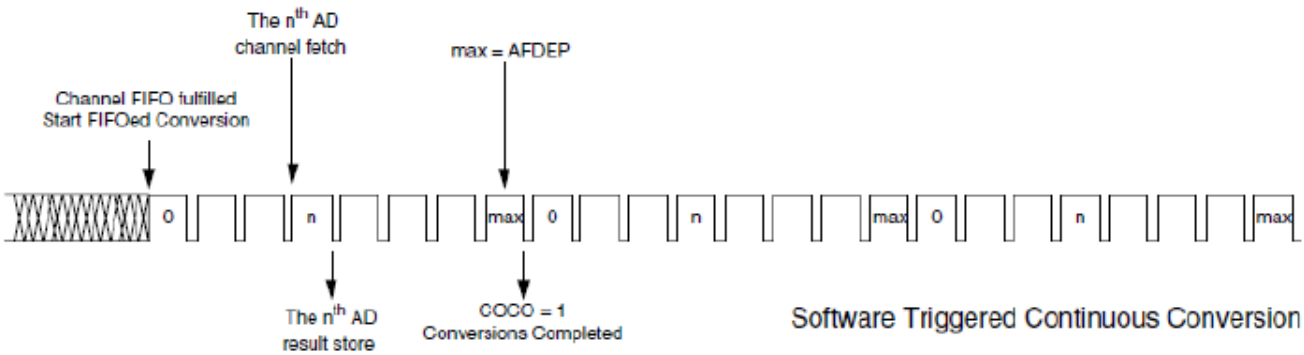

Figure 4. Software triggered continuous conversion

The sample code to initialize the ADC module with FIFO is as follows:

```
void ADC_ConversionWithFifo( void )
{ /* initialize the ADC module to enable FIFO , and software trigger mode */
ADC_APCTL1 = 0x0f;
ADC_APCTL2 = 0x00;
ADC_SC3_ADIV = ADC_ADIV_DIVIDE_4;
ADC_SC3_MODE = ADC_MODE_12BIT;
ADC_SC3_ADLSMP = 1;
ADC_SC3_ADICLK = CLOCK_SOURCE_ADACK; //select bus clock as clock source
ADC_SC4_AFDEP = 0x06; // FIFO level 7
ADC_SC4_ASCANE = 0; // disable scan mode
/* write channel FIFO till fulfilled*/
ADC_SC1_ADCH = 0x1d; // write channel fifo 1
ADC_SC1_ADCH = 0x02; // write channel fifo 2
ADC_SC1_ADCH = 0x03; // write channel fifo 3
ADC_SC1_ADCH = 0x1d; // write channel fifo 4
ADC_SC1_ADCH = 0x16; // write channel fifo 5
ADC_SC1_ADCH = 0x1d; // write channel fifo 6
ADC_SC1_ADCH = 0x1e; // write channel fifo 7
/* channel FIFO is ffulfilled, start to trigger conversion */
While( !ADC_SC1_COCO);
// COCO flag is set, read result
while( !ADC_SC2_FEMPTY )
{ // read data from ADC result register
uiADC_Buff[ucADC_Count++] = ADC_R;
}
}
```

2.3 Hardware trigger

When a hardware trigger occurs, it only fetches one analog channel in order from input analog channel FIFO and then saves conversion value to result FIFO. Whereas in software trigger, after completing the conversion, it fetches from the next input channel in order till all of channels in channel FIFO are completed. Once the conversion completes, an automatic fetch for next analog value from channel FIFO will be done.

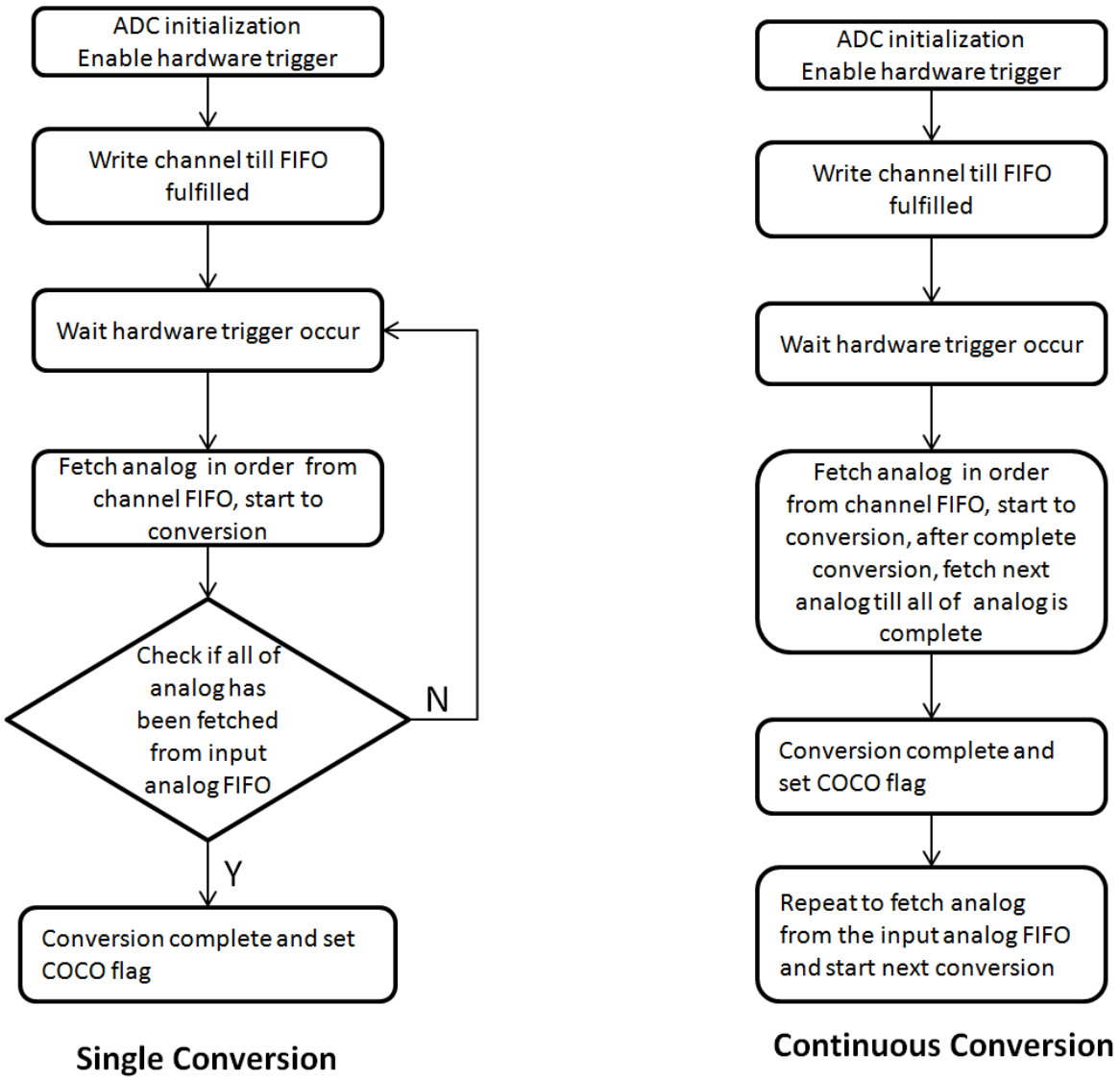

Figure 5. Hardware trigger flowchart

The preceding flowchart describes the difference between single conversion and continuous conversion with hardware trigger.

The following figure describes the progress of ADC conversion with hardware trigger.

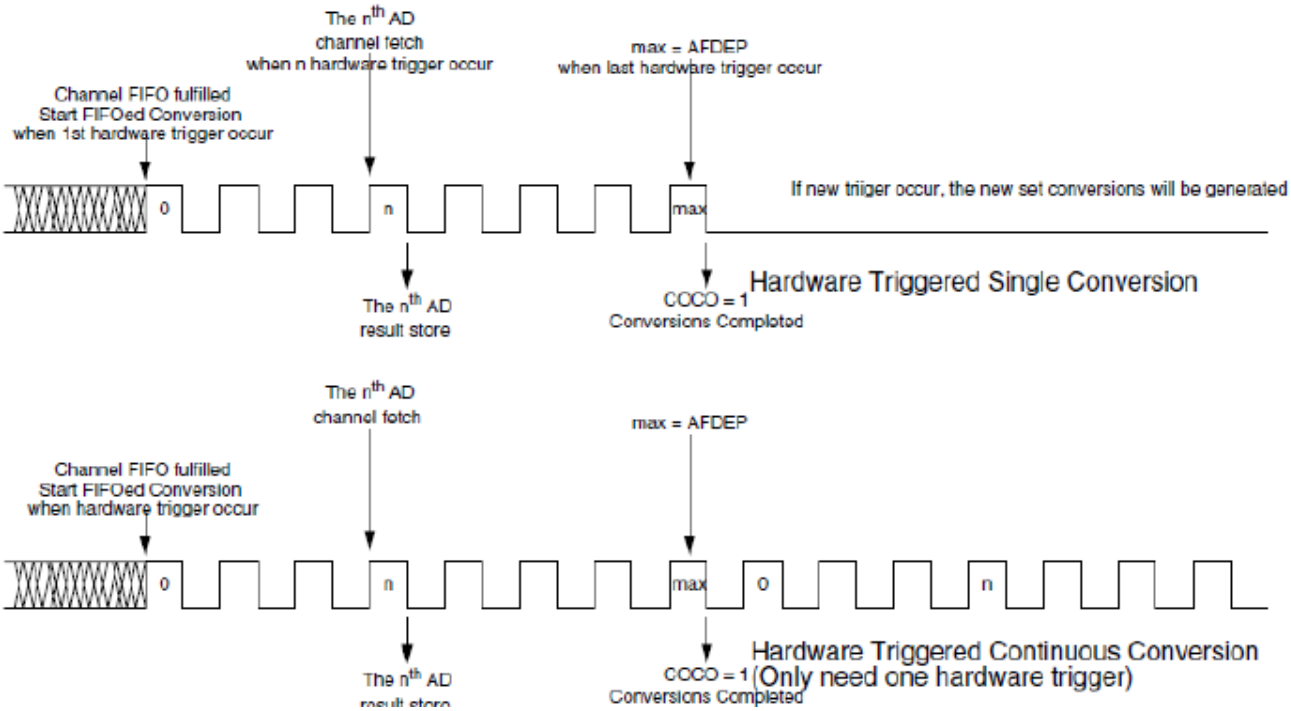

Figure 6. ADC FIFO conversion sequence

2.4 Scan mode

The FIFO also provides scan mode to simplify the dummy work of input channel FIFO. When the ASCANE bit in ADC_SC4 is set in FIFO mode, the FIFO will always use the first dummied channel in spite of the value in the input channel FIFO. The ADC conversion starts to work in FIFO mode as soon as the first channel is dummied. The following write operation to the input channel FIFO will cover the first channel element in this FIFO. In scan FIFO mode, the COCO bit is set when the result FIFO is fulfilled according to the depth indicated by the AFDEP bits in ADC_SC4 register.

The sample code of scan mode is as follows:

```
void ADC_ScanModeWithFIFO( void )
{
uint8_t i;
uint16_t uiADC_Result[5];
uint8_t ucCount;
ADC_APCTL1 = 0x0f; // enable channel 0 - 3
ADC_APCTL2 = 0x00;
ADC_SC3_ADIV = ADC_ADIV_DIVIDE_4;
ADC_SC3_MODE = ADC_MODE_12BIT;
ADC_SC3_ADLSMP = 0;
ADC_SC3_ADICLK = CLOCK_SOURCE_ADACK;
ADC_SC4_AFDEP = 0x03; // FIFO level 4
ADC_SC4_ASCANE = 1; // enable FIFO scan mode
//start to conversion
ADC_SC1 = 0x01; //write channel value
//wait conversion complete
ADC FIFO application on PT60
10 Freescale Semiconductor
while( !ADC_SC1_COCO );
ucCount = 0;
while( !ADC_SC2_FEMPTY )
{
uiADC_Result[ucCount] = ADC_R;
printf("0x%x,",uiADC_Result[ucCount]);
```

```
ucCount ++;
}
}
```

2.5 Compare function

COCO will not set if the following conditions are true:

- If none of the compare condition is true when ACFSEL is low.
- If not all the compare conditions are true when ACFSEL is high.

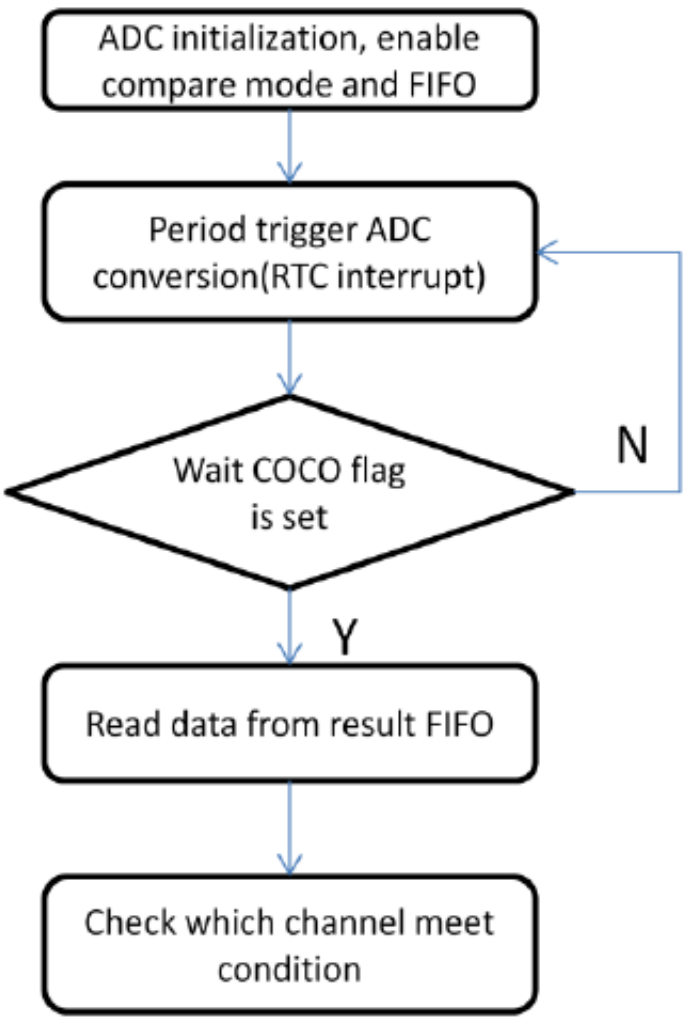

Figure 7. FIFO comparison flow chart

It supports automatic comparison of the ADC conversion result with value specified by ADC_CV, and sets only COCO flag when condition is true.

3 Conclusion

The ADC module supports FIFO operation to minimize the interrupts of CPU and improve the efficiency to access ADC module. It provides the software and hardware trigger as well as configurable FIFO depth, which makes it suitable for different applications. In addition, the compare function can work in low-power mode, and when condition is true, it can automatically wakes the MCU from low-power mode. This is very useful for low-power applications.

4 References

Following documents are available on freescale.com :

- MC9S08PT60RM: MC9S08PT60 Reference Manual
- AN1259: System Design and Layout Techniques for Noise Reduction in MCU-based Systems

5 Glossary

BDM	Background Debug Mode
CRC	Cyclic Redundancy Check
ECC	Error Correction Codes
RTC	Real Time Counter
TPM	Timer/PWM Module
FTM	Flextimer Module
MTIM	Modulo Timer
WDOG	Watchdog
TSI	Touch Sense Input

6 Revision history

Revision number	Date	Substantial changes
0	08/2013	Initial release

How to Reach Us:

Home Page:
freescale.com

Web Support:
freescale.com/support

Document Number AN4587
Revision 0, 08/2013