The following document contains information on Cypress products. The document has the series name, product name, and ordering part numbering with the prefix “MB”. However, Cypress will offer these products to new and existing customers with the series name, product name, and ordering part number with the prefix “CY”.

How to Check the Ordering Part Number

1. Go to www.cypress.com/pcn.
2. Enter the keyword (for example, ordering part number) in the **SEARCH PCNS** field and click **Apply**.
3. Click the corresponding title from the search results.
4. Download the Affected Parts List file, which has details of all changes

For More Information

Please contact your local sales office for additional information about Cypress products and solutions.

About Cypress

Cypress is the leader in advanced embedded system solutions for the world's most innovative automotive, industrial, smart home appliances, consumer electronics and medical products. Cypress' microcontrollers, analog ICs, wireless and USB-based connectivity solutions and reliable, high-performance memories help engineers design differentiated products and get them to market first. Cypress is committed to providing customers with the best support and development resources on the planet enabling them to disrupt markets by creating new product categories in record time. To learn more, go to www.cypress.com.

AN205277

F²MC-8FX Family, MB95200H/210H Series A/D Converter

This application note describes how to use 8/10-bit A/D converter in each mode and intended to describe the functions of the Analog/Digital Converter (ADC) and gives some examples.

Contents

1 Introduction....1
2 Feature....1
2.1 Key Features....1
2.2 Block Diagram....2
2.3 Registers....3
2.4 Interrupts of 8/10-bit A/D Converter....5
3 Analog Input and Related External Circuits....6
3.1 Electrical Characteristics....6
3.2 External Circuits for Analog Input....7
4 Consideration for Sampling Time....8
5 Examples for ADC....8
5.1 A/D General Procedure....8
5.2 How to Start A/D....9
5.3 How to Judge if A/D Converter has Finished....10
6 Notes on Using 8/10-bit A/D Converter....12
6.1 Notes on Programming Setup....12
6.2 Note on Interrupt Requests....12
6.3 Error....12
6.4 8/10-bit A/D Converter Analog Input Sequences....12
7 Additional Information....13
7.1 Sample Code....13
8 Document History....19

1 Introduction

This application note describes how to use 8/10-bit A/D converter in each mode.

The following application note is intended to describe the functions of the Analog/Digital Converter (ADC) and gives some examples.

2 Feature

This chapter introduces the basic functions of 8/10-bit A/D converter.

2.1 Key Features

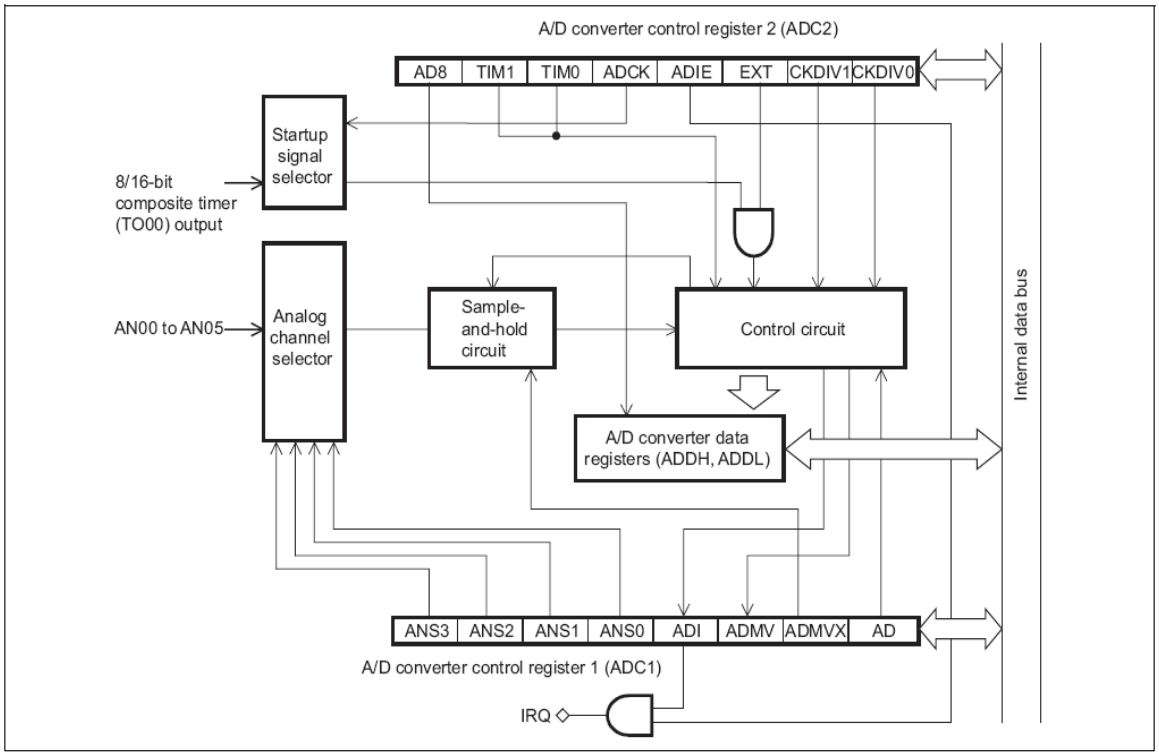
The A/D converter converts analog voltage (input voltage) input to an analog input pin to 10-bit digital value.

- One of multiple analog input pins can be selected.
- The conversion speed is programmable to be configured (selected according to operating voltage and frequency).
- An interrupt is generated when A/D conversion completes. The completion state of conversion can also be checked with ADI bit in the ADC1 register.
- To activate A/D conversion function, two methods there are given below.
  a. Activation using the A/D bit in the ADC1 register
  b. Continuous activation using the 8/16-bit composite timer output TO00

2.2 Block Diagram

Figure 1 shows the internal block diagram of ADC.

Figure 1. Block Diagram of ADC



The analog input signal (AN00 ~ AN05) is passed by an analog channel selector to a sample and hold stage and finally to a control circuit. The conversion results from analog to digital are conserved in A/D converter data register (ADDH, ADDL). Once a conversion is started or triggered, the control block will first sample the selected input channel and keep the voltage level stable in the sample and hold stage.

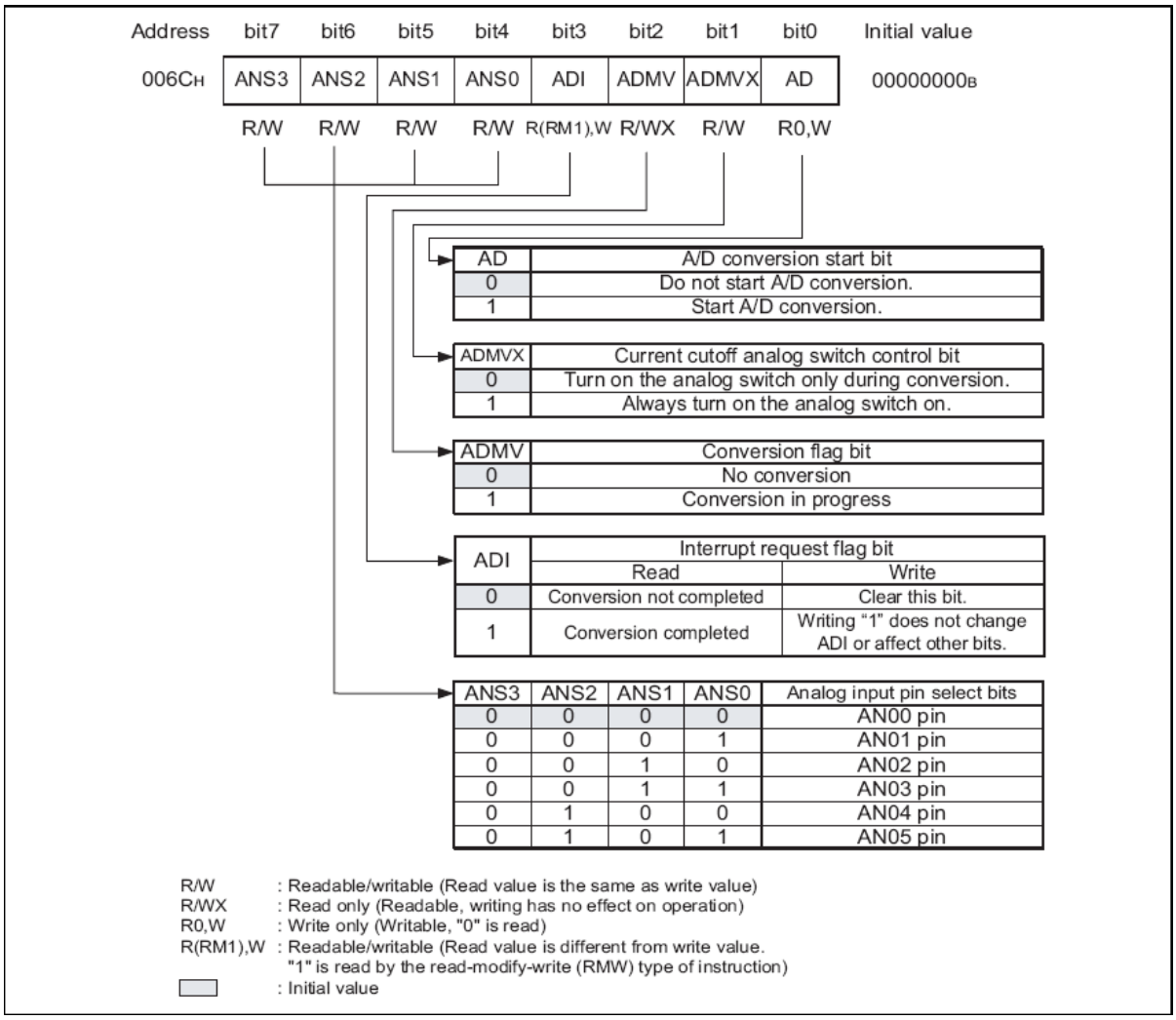
2.3 Registers

Please refer to Chapter 17 of the MB95200H/210H Series Hardware Manual for detailed register setting.

2.3.1 8/10-bit A/D Converter Control Register 1 (ADC1)

This register is used to enable and disable individual function of 8/10-bit A/D converter, select an analog input pin and check the status of A/D conversion.

Figure 2. 8/10-bit A/D Converter Control Register 1 (ADC1)

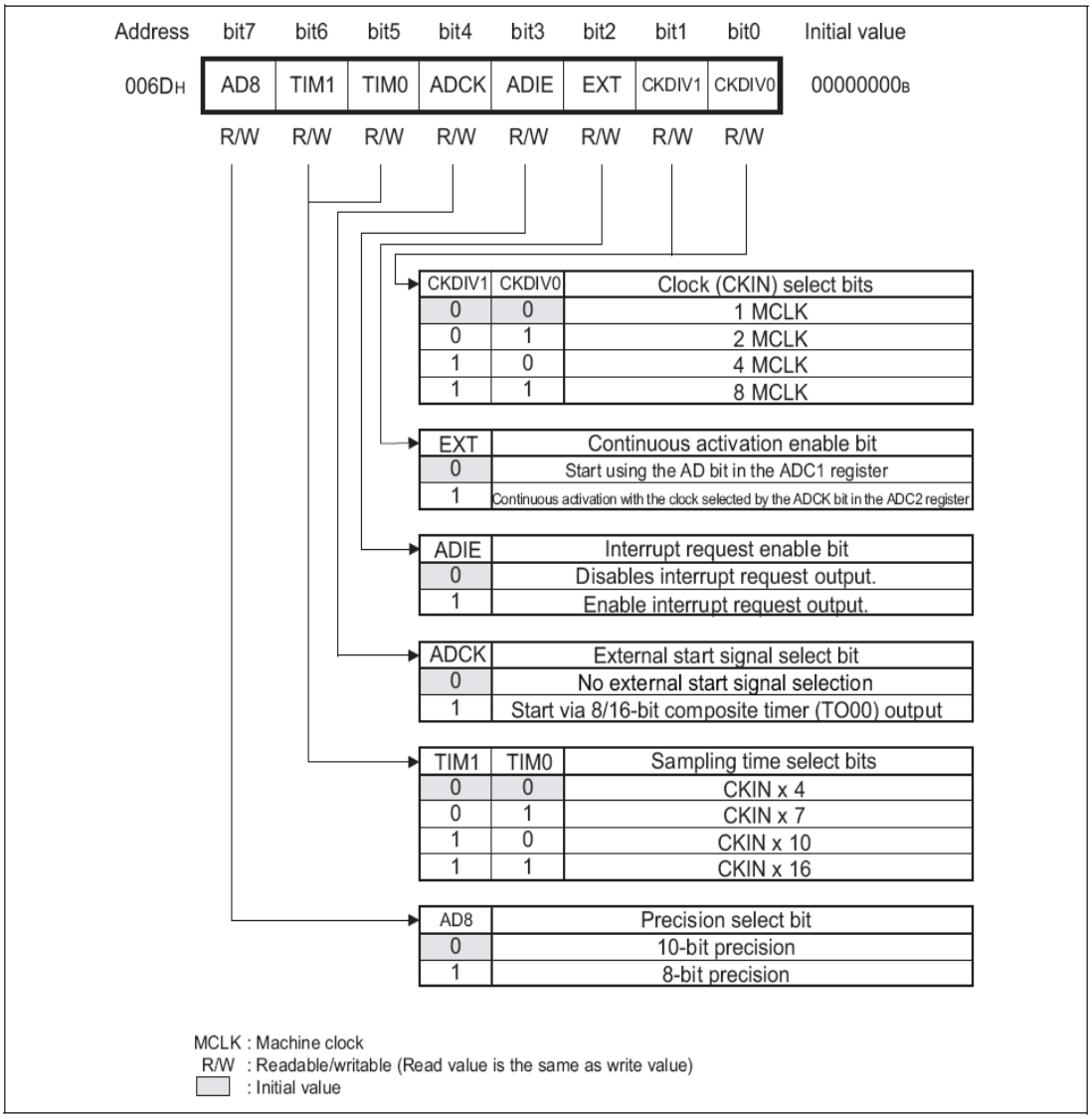


Note: The number of analog input pins varies depending on the series. MB95200H series has 6 channels of analog input pins (AN00~AN05). But MB95210H series has only 2 channels of analog input pins (AN05 ~ AN04). Analog input pins also serve as general-purpose I/O ports.

2.3.2 8/10-bit A/D Converter Control Register 2 (ADC2)

This register is used to select 8/10-bit A/D converter function, select input clock and perform an interrupt and check the status of A/D converter.

Figure 3. 8/10-bit A/D Converter Control Register 2 (ADC2)



2.3.3 8/10-bit A/D Converter Data Registers Upper/Lower (ADDH, ADDL)

The 8/16-bit A/D converter data registers upper/lower (ADDH, ADDL) contain the results of 10-bit A/D conversion. The upper two bits of 10-bit data correspond to ADDH register; the lower eight bits correspond to ADDL register. Set ADC2:AD8=1, in which case eight bits of data can be read from ADDL register, set ADC2:AD8=0, the bit selects 10-bit precision.

Figure 4. 8/10-bit A/D Converter Data Registers Upper/Lower (ADDH, ADDL)

ADDH	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Initial value
Address	-	-	-	-	-	-	SAR9	SAR8	00000000_B
$006E_H$	R0/WX	R0/WX	R0/WX	R0/WX	R0/WX	R0/WX	R/WX	R/WX	

ADDL	bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8	Initial value
Address	SAR7	SAR6	SAR5	SAR4	SAR3	SAR2	SAR1	SAR0	00000000_B
$006F_H$	R/WX	R/WX	R/WX	R/WX	R/WX	R/WX	R/WX	R/WX	

R/WX : Read only (Readable, writing has no effect on operation)
R0/WX : Undefined bit (Read value is "0", writing has no effect on operation)

2.4 Interrupts of 8/10-bit A/D Converter

2.4.1 Interrupts in 8/10-bit A/D Converter Operation

When A/D conversion is completed, the interrupt request flag bit (ADC1: ADI) is set to "1". If interrupt request enable bit is enabled (ADC2: ADIE = 1), an interrupt request is generated to the interrupt controller. Writing "0" to the ADI bit using the interrupt service routine to clear the interrupt request. The ADI bit is set when A/D conversion is completed, regardless of the value of ADIE bit. CPU cannot return from interrupt processing if interrupt request flag bit (ADC1: ADI) is set to “1” with interrupt requests enabled (ADC2: ADIE = 1). Be sure to clear ADI bit within the interrupt service routine.

2.4.2 Register and Vector Table Related to 8/10-bit A/D Converter Interrupts

Table 1. Register and Vector Table Related to 8/10-bit A/D Converter Interrupts

Interrupt Source	Interrupt Request No.	Interrupt Level Setting Register		Vector Table Address	
		Register	Setting bit	Upper	Lower
8/10-bit A/D	IRQ18	ILR4	L18	$FFD6_H$	$FFD7_H$

Please refer to APPENDIX B Table of Interrupt Sources of the MB95200H/210H Series Hardware Manual for interrupt request numbers and vector tables of all peripheral resources.

3 Analog Input and Related External Circuits

This chapter explains electrical characteristics and basic external circuits

3.1 Electrical Characteristics

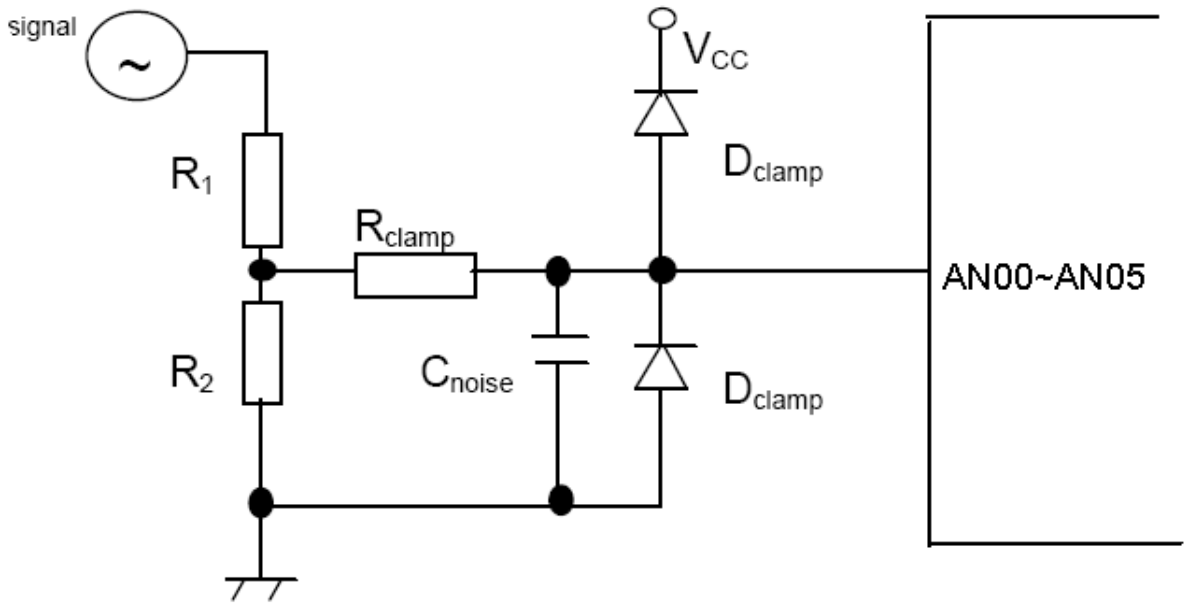
This table shows the main electrical characteristics of the 8FX Family.

Table 2. Electrical Characteristics Related to A/D converter

Parameter	Symbol	Value			Unit	Remarks
		Min	Typ	Max		
Resolution	—	—	—	10	bit	
Total error		– 3	—	+ 3	LSB	
Linearity error		– 2.5	—	+ 2.5	LSB	
Differential linear error		– 1.9	—	+ 1.9	LSB	
Zero transition voltage	V_{OT}	V_{SS} – 1.5 LSB	V_{SS} + 0.5 LSB	V_{SS} + 2.5 LSB	V	
Full-scale transition voltage	V_{FST}	V_{CC} – 4.5 LSB	V_{CC} – 2 LSB	V_{CC} + 0.5 LSB	V	
Compare time	—	0.9	—	16500	µs	4.5 V ≤ V_{CC} ≤ 5.5 V
		1.8	—	16500	µs	4.0 V ≤ V_{CC} < 4.5 V
Sampling time	—	0.6	—	∞	µs	4.5 V ≤ Vcc ≤ 5.5 V, with external impedance < 5.4 kΩ
		1.2	—	∞	µs	4.0 V ≤ Vcc ≤ 4.5 V, with external impedance < 2.4 kΩ
Analog input current	I_{AIN}	– 0.3	—	+ 0.3	µA	
Analog input voltage	V_{AIN}	V_{SS}	—	V_{CC}	V	

3.2 External Circuits for Analog Input

Figure 5. A Suggested Connection for A/D Conversion



To protect the analog pins from an over-voltage, a so-called “clamping resistor” is usually added to the input pins. The minimum value of the resistor can be chosen as

Rclamp = Uovervoltage / Iclamp

Where, Iclamp is the specified maximum clamping current in the Data Sheet.

For some applications, a large clamping resistor is sometimes unacceptable. As a compromise, an external clamping diode with low leakage current could be added between the input pin and VCC pin.

In some cases, the sensor has been biased with a voltage supply higher than the maximum guaranteed voltage for the microcontroller. For example, in the automotive applications, the sensors could be biased directly with the car battery, which exhibits a voltage of 12V/24V. A resistor divider consisting of R1/R2 is commonly used to tail the sensor voltage signal “seen” on the pin down to the value which is equal to or smaller than VCC (see Figure 3-1). The ratio between R1 and R2 should satisfy the following constraint:

$$\frac{R_1}{R_2} \geq \frac{U_{Signal}}{AV_{CC}} - 1$$

Other factor which affects the size dimension of R1, R2 and Rclamp, is related to current consumption budget and the input signal noise suppressing. The second factor will be discussed here with more details. The signal from the sensors could be also noisy. The noise, which has a time constant smaller than the sampling time Tsampling, is transparent to the ADC, resulting in distorted output. In this case, an additional dedicated bypass capacitor the clamping resistor or resistor divider, works as a low pass filter. A larger capacitor will lower the AC impedance and will be more effective at shunt away the noise signal. Generally, the time constant of this low pass filter (Rclamp + R1 || R2) x Cnoise should be chosen considerably larger than the sampling time (5 to 10 times larger with a rule of thumb).

However, at the same time this time constant should be also considerably smaller than the one of the sensor signal, depending on the applications. In this way, the analog pin is able to follow the dynamic changes, which the ADC is being used to track. These, along with the dimension of R1/R2 or Rclamp must be considered when choosing the capacitor dimension to avoid rolling off any high frequency signal components of interest.

4 Consideration for Sampling Time

This chapter introduces the relationship between variables.

This section introduces the sampling time in MCU (MB95200H/210H). First, we introduce the relationship between variables.

1 MCLK= 1/ PLL clock

The speed of A/D conversion is affected by clock mode, main clock oscillation frequency and main clock speed switching (gear function).

Example: Sampling time= CKIN× (ADC2: TIM1/TIM0 setting)

Compare time= CKIN×10 (fixed value) +MCLK

Conversion time= A/D start processing time + sampling time +compare time

- The error max.1CKIN-1MCLK may occur depending on the startup timing of AD converter.
- Program the software satisfied with "sampling time" and "compare time" in A/D Converter. Refer to the section" Electrical Characteristics" in Data Sheet of F2MC-8FX MB95200H/210H Series.

5 Examples for ADC

This chapter gives examples on 8/10-bit A/D converter

Generally, there are two methods to select 8/10-bit A/D conversion activation. One is to start by 8/16-bit composite timer (TO00) output, and the other is to set A/D conversion start bit (ADC1: AD=1). Also two methods there can be used to check whether the conversion has been completed or not.

5.1 A/D General Procedure

The following procedure is used to set up 8/10-bit A/D converter:

5.1.1 Initialization

- Set the port for input (DDR1).
- Set the interrupt level (ILR4).
- Enable A/D input (ADC1:ANS0 to ANS3).
- Set the sampling time (ADC2:TIM1, TIM0).
- Select the clock (ADC2:CKDIV1, CKDIV0).
- Set A/D conversion precision (ADC2:AD8).
- Select the operation mode (ADC2: EXT).
- Select the startup trigger (ADC2: ADCK).
- Enable interrupts (ADC2:ADIE=1).
- Activate A/D converter (ADC1:AD=1).

5.1.2 Interrupt Processing

- Clear the interrupt request flag (ADC1: ADI=0).
- Read convertsion values (ADDH, ADDL).
- Activate AD converter (ADC1:AD=1).

5.2 How to Start A/D

There are two methods to start A/D converter. One method is to set the start-bit to generate a software trigger. The other method is to start A/D converter by 8/10-bit composite timer (TO00) output. Examples for above are gave as follow.

5.2.1 By Setting Start-bit to Start A/D Converter

Use the A/D conversion start bit (ADC1: AD = 1) to generate a software trigger.

```
ADC2    = 0x81;         //8-bit resolution, 2×MCLK
ADC1    = 0x00;         //AN00 as input
ADC1_AD   = 1;          //start AD converter
```

Refer to appendix for sample code project1"AD_BASIC"

5.2.2 By 8/16-bit Composite Timer (TO00) Output

In this case, the timer0 has been used. For details on Timer0 operation, please refer to Chapter 14 in Hardware Manual of of MB95200H/210H Series.

The following example shows how to set up the 8/16-bit composite timer 0.

```
/*************************************************************
Name:        InitCompTimer
Function:    TO00 control AD converter
*************************************************************/
void InitCompTimer(void)
{
    T00DR = 0xE0;         // set count value (low 8 bit)
    TMCR0 = 0x40;         // 8-bit, no filtering
    T00CR0 = 0x01;        // interval timer with continuous mode
    T00CR1 = 0x81;       // enable output, start timer
}
```

Also the example for A/D conversion shows as follows

```
/*************************************************
Name:         MCU_Initialization
Function:     Initialization MCU
*************************************************/
void MCU_initialization()
{
                        //.....
    ADC1=0x00;          //AN00 as input
    ADC2=0x81;          //8-bit resolution, 2×MCLK
    ADC1_AD = 0;     //Don't to start AD conversion.
    ADC2_ADCK =1;    //Start via 8/16-bit composite timer (TO00) output
    ADC2_EXT =1;     //Continuous activation with the clock selected by
                        the ADCK bit in the ADC2 register
                        //.....
}
```

Refer to appendix for sample code project2 "AD-TIMER"

5.3 How to Judge if A/D Converter has Finished

5.3.1 Checking with the Conversion Flag Bit

In this case, we should pay attention to the conversion flag bit (ADC1: ADMV). If the read value is "1", the A/D conversion is in progress.

```
/***********************************************
Name:       AD_Sample
Function:   Use AN00 as the sample voltage channel
***********************************************/
void AD_Sample(void)
{
    Distem = ADDL/50;   //Read AD Sample Value
    ADC1_AD=1;          //Start AD again
}
/*********************************************************************
Name:       syinit
Function:   Initial AD
*********************************************************************/
void syinit(void)
{
    ADC1   = 0x01;        // start AD converter
    ADC2   = 0x82;        //8-bit precision, 4 MCLK
}
/*********************************************************************
Name:       main
Function:   Main Routine
*********************************************************************/
void main(void)
{
    syinit();
    while (1)
    {
      while(ADC1_ADMV);       //Check converter finish flag,
                              //ADC1_ADMV = 0 means AD Sample finished
      AD_Sample();
    }
}
```

Refer appendix sample code project1 “AD_BASIC”

5.3.2 Checking with Interrupt Request Flag Bit

In this case, we should pay attention to check with interrupt request flag bit, if the interrupt request flag (ADC1: ADI=1), A/D conversion are completed with interrupt request generated.

```
/*************************************************************************
Name:       InitADC
Function:   initialize AD
*************************************************************************/
void InitADC(void)
{
    ADC2 = 0x89;                  // 8-bit resolution, enable interrupts
    ADC1 = 0x01;                  // AN0 pin as input
}
/*************************************************************************
Name:       main
 Function:  Main Routine
*************************************************************************/
void main(void)
{
    while(1)                    // waiting for interrupt ( no operation )
    __asm("nop");
}
/*************************************************************************
AD Interrupt process
*************************************************************************/
__interrupt void ISR_ADC (void)
{
    Temp=ADDL;                          // read the AD sample value
    ADC1 = 0x00;                        // clear interrupt flag
}
```

Please note, that the corresponding interrupt vector and level shall be defined in the Vectors.c module of our standard template project.

```
void InitIrqLevels(void)
{
    ILR4 = 0xDF;          // IRQ18: 10-bit AD-converter
                           // IRQ19: Timebase timer
}
..............
__interrupt void ISR_ADC (void);
..............
#pragma intvect ISR_ADC    18  // IRQ18: 10-bit AD-converter
```

Refer to appendix for sample code project3 “AD_INT”

6 Notes on Using 8/10-bit A/D Converter

This section summarizes notes on using 8/10-bit A/D converter.

6.1 Notes on Programming Setup

- When A/D conversion function is used, the contents of ADDH and ADDL registers are retained upon the Completion of A/D conversion. During A/D conversion, the values resulting from last conversion are loaded.
- Do not re-select the analog input pin (ADC1: ANS3 to ANS0) while A/D conversion function is running, especially, during continuous activation. Disable the continuous activation (ADC2: EXT = 0) before re-selecting the analog input channel.
- Starting the reset, stop, or watch mode stops 8/10-bit A/D converter and initializes each register.
- CPU cannot return from interrupt processing routine if the interrupt request flag bit (ADC1: ADI) is set to”1” with interrupt requests enabled (ADC2: ADIE = 1). Be sure to clear the ADI bit within interrupt service routine.

6.2 Note on Interrupt Requests

If A/D conversion is reactivated (ADC1: AD = 1) and terminated simultaneously, the interrupt request flag bit (ADC1: ADI) is set.

6.3 Error

As | Vcc - Vss | decreases, an error increases relatively.

6.4 8/10-bit A/D Converter Analog Input Sequences

Turn on analog input (AN00 to AN05) at the same as or after turning on the digital power supply (VCC). In addition, turn off digital power supply (VCC) either at the same time as or after turning off analog input (AN00 to AN05).

Be careful not to let analog input exceed the voltage of digital power supply when turning 8/10-bit A/D converter on and off.

7 Additional Information

For more Information on MB95200 products, visit the following website:

http://www.cypress.com/8fx-mb95200

7.1 Sample Code

7.1.1 Project 1

Name: AD_BASIC

Function: By setting start-bit and Check with the conversion flag bit

```
/*****************************************************************************
Name:           MAIN.C
*****************************************************************************/
#include "mb95200.h"
unsigned char Distem;
/*****************************************************************************
Name:           AD_Sample
Function:       Use AN00 as the sample voltage channel
*****************************************************************************/
void AD_Sample(void)
{
   Distem = ADDL;                       // Read AD Sample value
   ADC1_AD=1;                   // Start AD again
}
/*****************************************************************************
Name:           syinit
Function:       Initial AD, IO-PORT
*****************************************************************************/
void syinit(void)
{
   ADC1=00;                     //AN00 as input
   ADC2 = 0x81;                 //8-bit resolution£¬2¡ÁMCLK
   ADC1_AD = 1;                         //start A/D converter
   DDR0 = 0x00;                 //use P00-P05 input
   PDR0 = 0x00;
}
```

```
/*****************************************************************************
 Name:              main
 Function:          Main Routine
*****************************************************************************/
void main(void)
{
   syinit();
   InitIrqLevels();                    // initialize Interrupt level register and IRQ vector table
   __EI();                                    // global interrupt enable
   __set_il(3);                        // set global interrupt mask to allow all IRQ levels
   while(1)
   {
         while(ADC1_ADMV);          //Check converter finish flag
AD_Sample();
   }
}
```

7.1.2 Project 2

Name: AD-TIMER

Function: Start A/D by 8/16 bit compound timer and checking with the conversion flag bit

```
/*****************************************************************************
Name:           MAIN.C
Function:       Start AD By 8/16-bit composite timer (TO00) output.
*****************************************************************************/
#include "mb95200.h"
unsigned char ad_data;
/*****************************************************************************
Name:           MCU_Initialization
Function:       Initialization MCU
*****************************************************************************/
void MCU_initialization()
{
  __DI();
  SYCC=0x00;            //MCLK = source clock = 8 MHz (Main CR)
  DDR0_P05 = 1;
  AIDRL=0xfc;           //IO port
  WDTC=0x35;                    //Clear watch dog timer
  ADC1=0x00 ;           //AN00 as input
  ADC2=0x81;            //8-bit precision,disable interrupt, 2¡ÁMCLK
  ADC1_AD = 0;                  //No start A/D conversion.
  ADC2_ADCK =1;                 //Start AD by 8/16-bit composite timer (TO00) output
  // Continuous activation with the clock selected by the ADCK bit in the ADC2 register
  ADC2_EXT   =1;
}
/*****************************************************************************
Name:           InitCompTimer
Function:       TO00 control AD converter
*****************************************************************************/
void InitCompTimer(void)
{
  T00DR = 0xE0;         // set count value (low 8 bit)
  TMCR0 = 0x40;         // 8-bit, no filtering
  T00CR0 = 0x01;        // interval timer with continuous mode
  T00CR1 = 0x81;        // enable output, start timer
```

```
}
/*****************************************************************************
Name:               ad_sample
Function:           AD sample code
*****************************************************************************/
void ad_sample()
{
   while (ADC1_ADMV);               // If AD Complete?
   ad_data=ADDL;                              //Read AD Sample value
}
/*****************************************************************************
Name:               main
Function:           Main Loop
*****************************************************************************/
void main()
{
   MCU_initialization();
   InitCompTimer();
   while(1)
   {
ad_sample();
WDTC=0x35;                  //Clear watch dog timer
   }
}
```

7.1.3 Project 3

Name: AD_INT

Function: Start A/D by setting start-bit and checking with interrupt request flag bit

```
/*****************************************************************************
Name:           MAIN.C
Function:       Checking with interrupt request flag bit
*****************************************************************************/
#include "mb95200.h"
/*****************************************************************************
Name:           InitADC
Function:       initialize AD
*****************************************************************************/
int  tmp;
void InitADC(void)
{
   ADC2 = 0x89;             // 8-bit resolution, enable interrupts
   ADC1 = 0x01;             // AN0 pin as input, start AD
}
/*****************************************************************************
 Name:           main
 Function:       Main Routine
*****************************************************************************/
void main(void)
{
   PDR0 = 0x00;                 // Port 0:
   AIDRL = 0xfe;                // used as I/O-port (no analog inputs), P00 as AN-input
   DDR0 = 0x00;                 // Set to input
   InitIrqLevels();             // initialise Interrupt level register and IRQ vector table
   __EI();                      // global interrupt enable
   __set_il(3);             // set global interrupt mask to allow all IRQ levels
   InitADC();               // init AD - converter
   while(1)                 // waiting for interrupt ( no operation )
   {
__asm("nop");
   }
}
```

```
/*****************************************************************************
Interrupt process
*****************************************************************************/
#include “mb95200.h”
__interrupt void ISR_ADC (void)
{
   tmp = ADDL;                    // voltage calculating
   ADC1 = 0x01;                    // clear interrupt flag, start A/D again
}
/*****************************************************************************
  VECTORS.C
*****************************************************************************/
void InitIrqLevels(void)
{
   ILR4 = 0xDF;              //  IRQ18: 10-bit AD-converter
                             //  IRQ19: Timebase timer
}
__interrupt void DefaultIRQHandler (void);
__interrupt void ISR_ADC (void);
#pragma intvect ISR_ADC  18          //  IRQ18: 10-bit AD-converter
__interrupt void DefaultIRQHandler (void)
{
   __DI();                           // disable interrupts
   while(1)
   __wait_nop();                     // halt system
}
```

8 Document History

Document Title: AN205277 - F²MC-8FX Family, MB95200H/210H Series A/D Converter

Document Number: 002-05277

Revision	ECN	Orig. of Change	Submission Date	Description of Change
**	-	HUAL	03/20/2008	Initial release
			07/15/2008	Modification
*A	5270106	HUAL	05/13/2016	Migrated Spansion Application Note MCU-AN- 500005-E-11 to Cypress format

Worldwide Sales and Design Support

Cypress maintains a worldwide network of offices, solution centers, manufacturer's representatives, and distributors. To find the office closest to you, visit us at Cypress Locations.

Products

ARM® Cortex® Microcontrollers	cypress.com/arm
Automotive	cypress.com/automotive
Clocks & Buffers	cypress.com/clocks
Interface	cypress.com/interface
Lighting & Power Control	cypress.com/powerpsoc
Memory	cypress.com/memory
PSoC	cypress.com/psoc
Touch Sensing	cypress.com/touch
USB Controllers	cypress.com/usb
Wireless/RF	cypress.com/wireless

PSoC® Solutions

cypress.com/psoc

PSoC 1 | PSoC 3 | PSoC 4 | PSoC 5LP

Cypress Developer Community

Community | Forums |Blogs | Video |Training

Technical Support

cypress.com/support

PSoC is a registered trademark and PSoC Creator is a trademark of Cypress Semiconductor Corporation. All other trademarks or registered trademarks referenced herein are the property of their respective owners.

Cypress Semiconductor
198 Champion Court
San Jose, CA 95134-1709

Phone : 408-943-2600
Fax : 408-943-4730
Website : www.cypress.com

© Cypress Semiconductor Corporation, 2008-2016. This document is the property of Cypress Semiconductor Corporation and its subsidiaries, including Spansion LLC ("Cypress"). This document, including any software or firmware included or referenced in this document ("Software"), is owned by Cypress under the intellectual property laws and treaties of the United States and other countries worldwide. Cypress reserves all rights under such laws and treaties and does not, except as specifically stated in this paragraph, grant any license under its patents, copyrights, trademarks, or other intellectual property rights. If the Software is not accompanied by a license agreement and you do not otherwise have a written agreement with Cypress governing the use of the Software, then Cypress hereby grants you a personal, non-exclusive, nontransferable license (without the right to sublicense) (1) under its copyright rights in the Software (a) for Software provided in source code form, to modify and reproduce the Software solely for use with Cypress hardware products, only internally within your organization, and (b) to distribute the Software in binary code form externally to end users (either directly or indirectly through resellers and distributors), solely for use on Cypress hardware product units, and (2) under those claims of Cypress's patents that are infringed by the Software (as provided by Cypress, unmodified) to make, use, distribute, and import the Software solely for use with Cypress hardware products. Any other use, reproduction, modification, translation, or compilation of the Software is prohibited.

TO THE EXTENT PERMITTED BY APPLICABLE LAW, CYPRESS MAKES NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS DOCUMENT OR ANY SOFTWARE OR ACCOMPANYING HARDWARE, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. To the extent permitted by applicable law, Cypress reserves the right to make changes to this document without further notice. Cypress does not assume any liability arising out of the application or use of any product or circuit described in this document. Any information provided in this document, including any sample design information or programming code, is provided only for reference purposes. It is the responsibility of the user of this document to properly design, program, and test the functionality and safety of any application made of this information and any resulting product. Cypress products are not designed, intended, or authorized for use as critical components in systems designed or intended for the operation of weapons, weapons systems, nuclear installations, life-support devices or systems, other medical devices or systems (including resuscitation equipment and surgical implants), pollution control or hazardous substances management, or other uses where the failure of the device or system could cause personal injury, death, or property damage ("Unintended Uses"). A critical component is any component of a device or system whose failure to perform can be reasonably expected to cause the failure of the device or system, or to affect its safety or effectiveness. Cypress is not liable, in whole or in part, and you shall and hereby do release Cypress from any claim, damage, or other liability arising from or related to all Unintended Uses of Cypress products. You shall indemnify and hold Cypress harmless from and against all claims, costs, damages, and other liabilities, including claims for personal injury or death, arising from or related to any Unintended Uses of Cypress products.

Cypress, the Cypress logo, Spansion, the Spansion logo, and combinations thereof, PSoC, CapSense, EZ-USB, F-RAM, and Traveo are trademarks or registered trademarks of Cypress in the United States and other countries. For a more complete list of Cypress trademarks, visit cypress.com. Other names and brands may be claimed as property of their respective owners.