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 家族 MB95200H/210H 系列 A/D 转换器

本应用笔记介绍各个模式下如何使用 8/10 位 A/D 转换器以及模拟/数字转换器 (ADC) 的功能并举例说明。

目录

1 概要 1
2 功能 1
2.1 主要功能 1
2.2 框图 2
2.3 寄存器 3
2.4 8/10 位 A/D 转换器的中断 5
3 模拟输入和相关外部电路 6
3.1 电气规范 6
3.2 模拟输入的外部电路 7
4 采样时间的注意事项 8
5 ADC 示例 8
5.1 A/D 转换器设定步骤 8
5.2 A/D 转换器的启动方法 9
5.3 A/D 转换状态的判定方法 10
6 使用 8/10 位 A/D 转换器时的注意事项 12
6.1 编程设置的注意事项 12
6.2 中断请求的注意事项 12
6.3 误差 12
6.4 8/10 位 A/D 转换器模拟输入时序 12
7 更多信息 13
7.1 示例代码 13
文档修改记录 19

1 概要

本应用笔记介绍各个模式下如何使用 8/10 位 A/D 转换器。

以下应用笔记将介绍模拟/数字转换器 (ADC) 的功能并举例说明。

2 功能

本章介绍了 8/10 位 A/D 转换器的基本功能。

2.1 主要功能

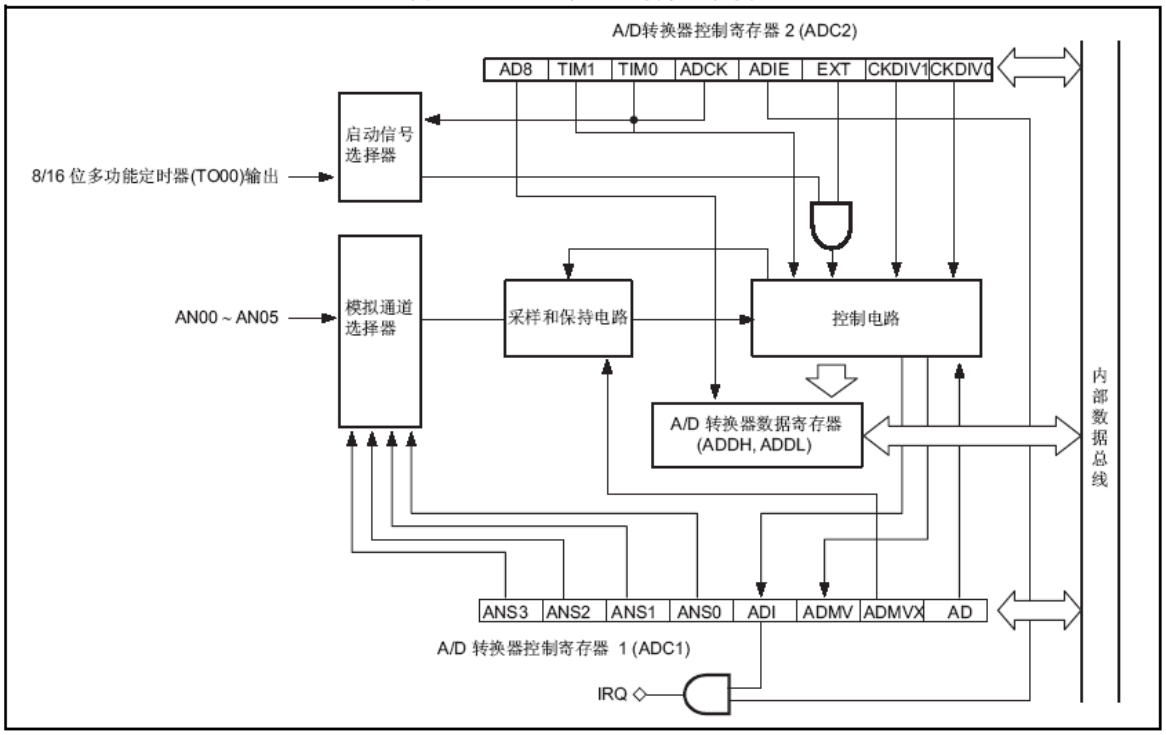
A/D 转换器将模拟输入引脚的模拟电压(输入电压)转换为 10 位数字值。

- 可从多种模拟输入引脚中选择一种。
- 可设定转换速度(根据工作电压和频率进行选择)。
- A/D 转换完成时，发生中断。通过 ADC1 寄存器的 ADI 位检查转换状态。
- 以下两种方法可激活 A/D 转换功能。
  a. 使用 ADC1 寄存器的 A/D 位激活
  b. 使用 8/16 位多功能定时器输出 TO00 连续激活

2.2 框图

图 1 是 ADC 的内部框图。

图 1. ADC 的内部框图



通过模拟通道选择器将模拟输入信号(AN00 ~ AN05)传送到采样保持电路并最后传输到控制电路。模拟→数字的转换结果保存到 A/D 转换器数据寄存器(ADDH、ADDL)。一旦转换开始或触发，则控制电路将先采样所选输入通道并保持采样保持电路中的电压电平稳定。

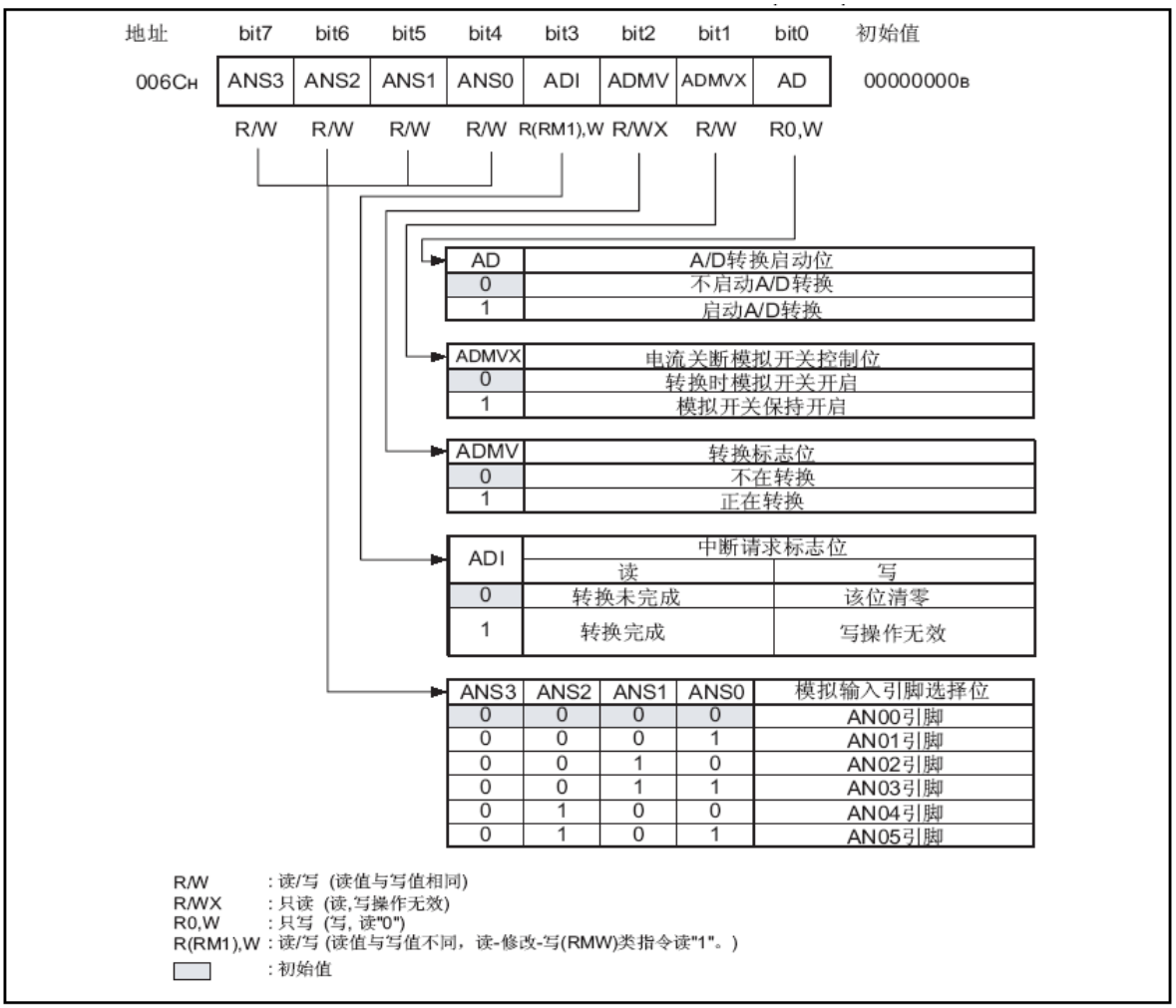
2.3 寄存器

关于寄存器的详细信息，参考 MB95200H/210H 系列硬件手册的“第 17 章”。

2.3.1 8/10 位 A/D 转换器控制寄存器 1 (ADC1)

该寄存器用于启用/禁止 8/10 位 A/D 转换器功能，选择模拟输入引脚，检查 A/D 转换状态。

图 2. 8/10 位 A/D 转换器控制寄存器 1 (ADC1)

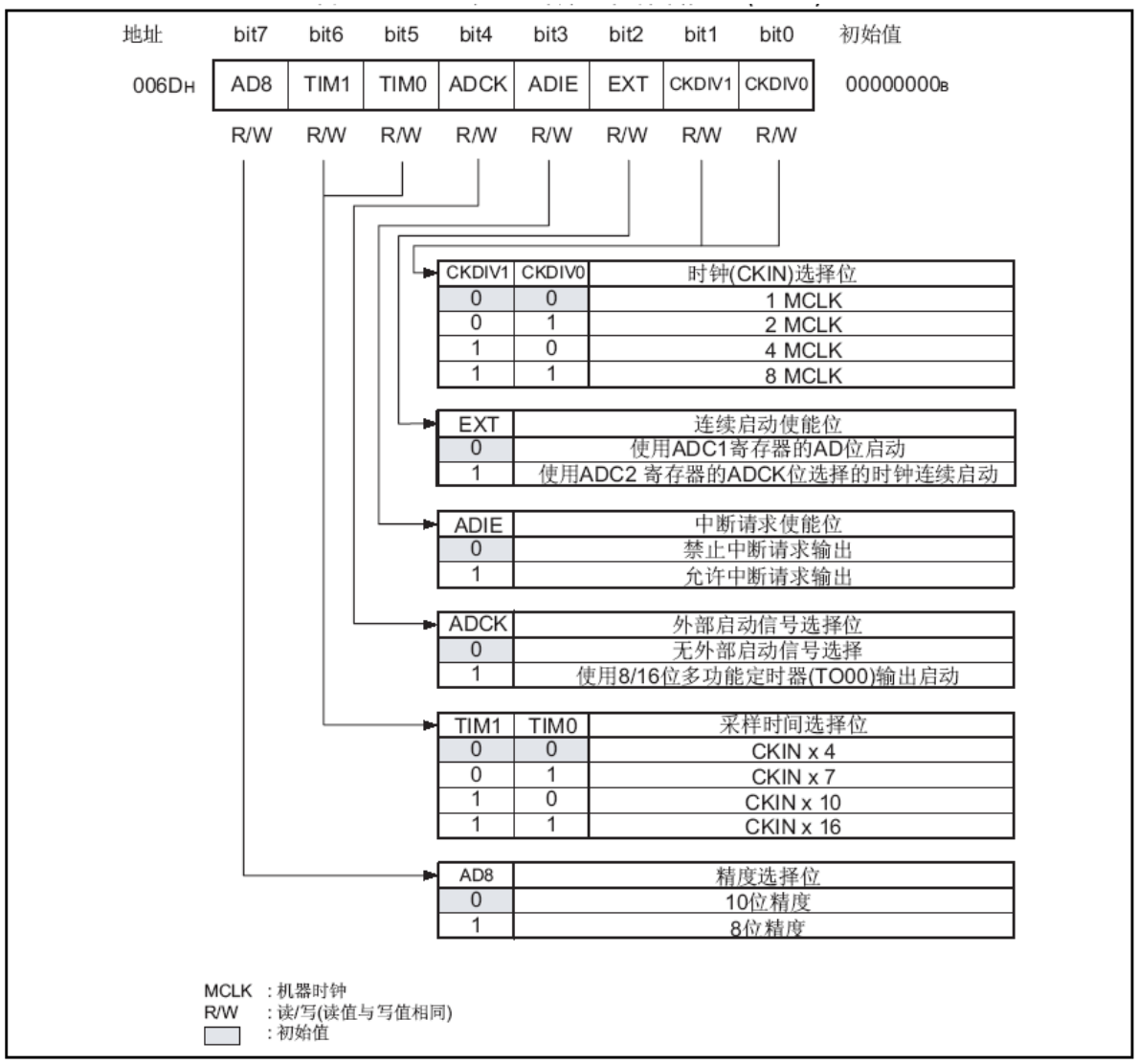


注意：模拟输入引脚号因产品系列而异。MB95200H 系列有模拟输入引脚(AN00~ AN05)的 6 路通道。但 MB95210H 系列仅有模拟输入引脚(AN05~ AN04)的 2 路通道。模拟输入引脚也可用作通用 I/O 端口。

2.3.2 8/10 位 A/D 转换器控制寄存器 2 (ADC2)

该寄存器用于选择 8/10 位 A/D 转换器功能，选择输入时钟，处理中断，检查 A/D 转换状态。

图 3. 8/10 位 A/D 转换器控制寄存器 2 (ADC2)



2.3.3 8/10 位 A/D 转换器数据寄存器高位/低位(ADDH、ADDL)

8/10 位 A/D 转换器数据寄存器高位/低位(ADDH、ADDL)指示 10 位 A/D 转换结果。10 位数据的高 2 位指示 ADDH 寄存器; 低 8 位指示 ADDL 寄存器。设定 ADC2:AD8=1 时，可从 ADDL 寄存器中读取数据的高 8 位，设定 ADC2:AD8=0 时, 该位选择 10 位精度。

图 4. 8/10 位 A/D 转换器数据寄存器高位/低位(ADDH、ADDL)

ADDH	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	初始值
地址	-	-	-	-	-	-	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	初始值
地址	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 ：只读 (可读，写操作无效)
R0/WX ：未定义位 (读值为 "0"，写操作无效)

2.4 8/10 位 A/D 转换器的中断

2.4.1 8/10 位 A/D 转换器的中断操作

A/D 转换完成后，中断请求标志位(ADC1:ADI)置“1”。若启用中断请求使能位(ADC2:ADIE=1)，则中断请求生成并传送到中断控制器。通过中断服务程序清零 ADI 位以清除中断请求。无论 ADIE 位的值如何，当 A/D 转换完成时，ADI 位置位。启用中断请求(ADC2:ADIE=)状态下，若中断请求标志位(ADC1:ADI)置“1”，则 CPU 无法从中断处理中返回。务必在中断服务程序内清零 ADI 位。

2.4.2 8/10 位 A/D 转换器中断的关联寄存器和向量表

表 1. 8/10 位 A/D 转换器中断的关联寄存器和向量表

中断源	中断请求号	中断级设置寄存器		向量表地址	
		寄存器	设置位	高位	低位
8/10 位 A/D	IRQ18	ILR4	L18	$FFD6_H$	$FFD7_H$

关于所有外设资源的中断请求号和向量表，详见 MB95200H/210H 系列硬件手册“附录 B 中断源一览表”。

3 模拟输入和相关外部电路

本章介绍了电气规范和基本外部电路。

3.1 电气规范

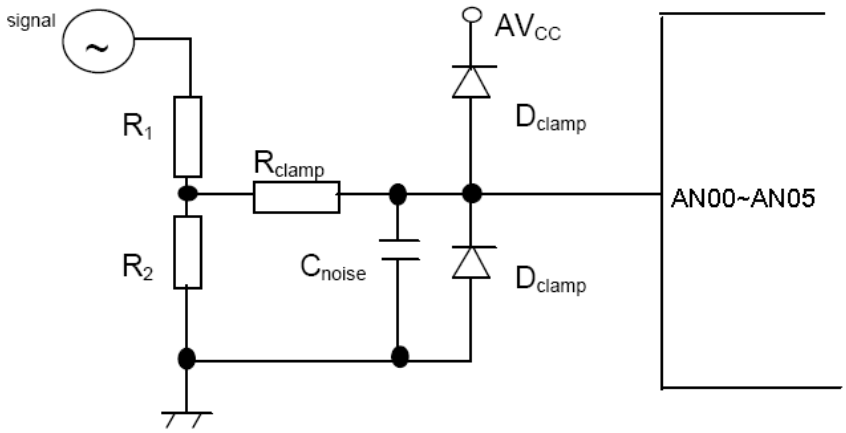
下表是 8FX 家族的主要电气规范。

表 2. A/D 转换器相关的电气规范

参数	符号	值			单位	备注
		最小	典型	最大		
分辨率	—	—	—	10	bit	
总误差		− 3	—	+ 3	LSB	
线性误差		− 2.5	—	+ 2.5	LSB	
微分线性误差		− 1.9	—	+ 1.9	LSB	
零转换电压	V_{OT}	V_{SS} − 1.5 LSB	V_{SS} + 0.5 LSB	V_{SS} + 2.5 LSB	V	
全面转换电压	V_{FST}	V_{CC} − 4.5 LSB	V_{CC} − 2 LSB	V_{CC} + 0.5 LSB	V	
比较时间	—	0.9	—	16500	µs	4.5 V ≤ V_{CC} ≤ 5.5 V
		1.8	—	16500	µs	4.0 V ≤ V_{CC} < 4.5 V
采样时间	—	0.6	—	∞	µs	4.5 V ≤ Vcc ≤ 5.5 V, 外部阻抗 < 5.4 kΩ
		1.2	—	∞	µs	4.0 V ≤ Vcc ≤ 4.5 V, 外部阻抗 < 2.4 kΩ
模拟输入电流	I_{AIN}	− 0.3	—	+ 0.3	µA	
模拟输入电压	V_{AIN}	V_{SS}	—	V_{CC}	V	

3.2 模拟输入的外部电路

图 5. A/D 转换的推荐连接



为保护过电压的模拟引脚，通常向输入引脚引接一个“钳位电阻器”。电阻器的最小值可通过以下算出：

$R_{clamp} = U_{overvoltage} / I_{clamp}$

以上公式中，Iclamp 指“数据手册”中规定的最大钳位电流。

很多应用中不使用较大的钳位电阻器。所以，在输入引脚和 VCC 引脚间，连接带低漏电流功能的外部钳位二极管。

某些情况下，由于输入电压高于微控制器规定的最大额定电压，因此传感器可能产生偏差。例如，汽车电子应用中，车载传感器直接使用车用电池（电压是 12V/24V）时可能产生偏差。电阻分压器（包含 R1/R2）通常用于跟踪传感器引脚的电压信号（保证引脚值等于或小于 VCC）（详见图 3-1）。R1 和 R2 的比率应满足以下公式：

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

影响 R1、R2 和 R_{clamp} 值的其他因素有相应的功耗预算和输入信号噪声控制，后者将会详细讨论。传感器的信号也可能成为噪声，该噪声（有一个小于采样时间 Tsampling 的时间常量）通过 ADC 传送到 MCU，以致引起错误输出。这种情况下，添加专用旁通电容器或钳位电阻器或电阻分压器，使其用作低通滤波器。大的电容器将降低 AC 阻抗并更有效的阻断噪声信号。通常而言，低通滤波器的时间常量(Rclamp + R1 || R2) x Cnoise 应大于采样时间（大于经验法则的 5 ~ 10 倍）。

然而，根据应用程序不同，时间常量应小于传感器信号中一个。这种情况下，模拟引脚应跟踪 ADC 的动态变化。选择电容器时，应考虑 R1/R2 或 Rclamp 的尺寸以避免滤掉重要的高频信号。

4 采样时间的注意事项

本章介绍了变量间的关系。

本章介绍了 MCU（MB95200H/210H 系列）中的采样时间。首先，我们介绍变量间的关系。

1 MCLK= 1/ PLL clock

A/D 转换速度受时钟模式、主时钟振荡频率和主时钟速度切换（换档功能）的影响。

例：采样时间=CKIN × （ADC2:TIM1/TIM0 设置）

比较时间=CKIN × 10 （固定值） + MCLK

转换时间=A/D 启动处理时间 + 采样时间 + 比较时间

- 根据 A/D 转换器的启动时序，可能发生最大 1CKIN-1MCLK 的误差。
- A/D 转换器中进行软件编程时，需满足“采样时间”和“比较时间”。详见《F²MC-8FX 家族 MB95200H/210H 系列数据手册》的“电气规范”部分。

5 ADC 示例

本章举例说明了 8/10 位 A/D 转换器。

通常而言，有两种方法可用来启动 8/10 位 A/D 转换。一是通过 8/16 位多功能定时器(TO00) 输出启动，另一种是设定 A/D 转换启动位(ADC1:AD=1)。另外，也有两种方法用来判定 A/D 转换是否完成。

5.1 A/D 转换器设定步骤

以下步骤用于设定 8/10 位 A/D 转换器。

5.1.1 初始化

- 设定输入端口 (DDR1)
- 设定中断级 (ILR4)
- 启用 A/D 输入 (ADC1:ANS0 ~ ANS3)
- 设定采样时间 (ADC2:TIM1, TIM0)
- 选择时钟 (ADC2:CKDIV1, CKDIV0)
- 设定 A/D 转换精度 (ADC2:AD8)
- 选择操作模式 (ADC2:EXT)
- 选择启动触发 (ADC2:ADCK)
- 启用中断 (ADC2:ADIE=1)
- 激活 A/D 转换器 (ADC1:AD=1)

5.1.2 中断处理

- 清零中断请求标志 (ADC1:ADI=0)
- 读取转换值 (ADDH, ADDL)
- 激活 A/D 转换器 (ADC1:AD=1)

5.2 A/D 转换器的启动方法

有两种方法用来启动 A/D 转换器。一是设定启动位以产生软件触发。另一种是通过 8/16 位多功能定时器(TO00)输出启动 A/D 转换器。以下是上述方法的示例。

5.2.1 通过设定启动位启动 A/D 转换器

使用 A/D 转换启动位(ADC1:AD=1)产生软件触发。

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

关于示例代码 project 1“AD_BASIC”，详见附录。

5.2.2 通过 8/16 位多功能定时器(TO00)输出

这种情况下，使用定时器 0。关于定时器 0 操作的详细信息，详见 F²MC-8FX 家族 MB95200H/210H 系列硬件手册的“第 14 章”。

以下是 8/16 位多功能定时器 0 的设定示例。

```
/*******************************************************
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
}
```

以下是 A/D 转换示例。

```
*************************************************
ame:           MCU_Initialization
unction:       Initialization MCU
*************************************************/
oid 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
                         //.....
```

关于示例代码 project 2 “AD-TIMER”，详见附录。

5.3 A/D 转换状态的判定方法

5.3.1 通过转换标志位判定

这种情况下，应使用转换标志位(ADC1:ADMV)。若读值是“1”，则 A/D 转换正在进行。

```
/*************************************************
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();
    }
}
```

关于示例代码 project 1 “AD_BASIC”，详见附录。

5.3.2 通过中断请求标志位判定

这种情况下，应使用中断请求标志位判定。若中断请求标志(ADC1:ADI=1)置位，则中断请求产生的状态下，A/D 转换完成。

```
/*************************************************************************
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
}
```

注意在标准临时工程的 Vectors.c 模块中，定义相应的中断向量和中断级。

```
oid InitIrqLevels(void)

   ILR4 = 0xDF;                    // IRQ16: 16-bit reload timer ch1 | I²C ch0
                                   // IRQ17: 16-bit PPG ch1
                                       // IRQ18: 10-bit AD-converter
                                       // IRQ19: Timebase timer


.............
_interrupt void ISR_ADC (void);
.............
pragma intvect ISR_ADC    18    // IRQ18: 10-bit AD-converter
```

关于示例代码 project 3“AD_INT”，详见附录。

6 使用 8/10 位 A/D 转换器时的注意事项

本章介绍了使用 8/10 位 A/D 转换器时的注意事项。

6.1 编程设置的注意事项

- 使用 A/D 转换功能时，A/D 转换完成后，保持 ADDH 和 ADDL 寄存器的内容。A/D 转换期间，保持上次转换的值。
- A/D 转换功能运行时，特别是连续激活时，切勿重选模拟输入引脚(ADC1:ANS3 ~ ANS0)。重选模拟输入通道前，禁止连续激活(ADC2:EXT=0)。
- 启动复位模式、停止模式或计时模式以停止 8/10 位 A/D 转换器并初始化各个寄存器。
- 启用中断请求(ADC2:ADIE=1)后，若中断请求标志位(ADC1:ADI)置“1”，则 CPU 无法从中断处理程序中返回。务必在中断服务程序中清零 ADI 位。

6.2 中断请求的注意事项

若 A/D 转换恢复运行(ADC1:AD=1)且同时终止，则中断请求标志位(ADC1:ADI)置位。

6.3 误差

|Vcc – Vss|变小时，误差相对增大。

6.4 8/10 位 A/D 转换器模拟输入时序

数字电源(VCC)打开后或打开的同时，打开模拟输入(AN00 ~ AN05)。另外，关闭模拟输入(AN00 ~ AN05)的同时或关闭后，关闭数字电源(VCC)。

打开/关闭 8/10 位 A/D 转换器时，注意不要让模拟输入电压超过数字电源电压。

7 更多信息

如欲了解有关 MB95200 产品的更多详情，敬请访问以下网址：

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

7.1 示例代码

7.1.1 工程 1

工程名称：AD_BASIC

功能：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 工程 2

工程名称： AD-TIMER

功能：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: 初始化 MCU
*****************************************************************************/
void MCU_initialization()
{
__DI();
SYCC=0x00;                  //MCLK = source clock = 4Mhz (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 工程 3

工程名称： AD_INT

功能： 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;              //  IRQ16: 16-bit reload timer ch1 | I2C ch0
                             //  IRQ17: 16-bit PPG ch1
                             //  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
}
```

文档修改记录

文档标题：AN205277 - F²MC-8FX 家族 MB95200H/210H 系列 A/D 转换器

文档编号：002-05740

修订版	ECN	变更者	提交日期	变更说明
**	—	HUAL	03/20/2008 07/15/2008	初稿
*A	5342871	HUAL	07/12/2016	已将 Spansion 应用手册《MCU-AN-500005-Z-11》转换成 Cypress 格式。

全球销售和设计支持

赛普拉斯公司拥有一个由办事处、解决方案中心、厂商代表和经销商组成的全球性网络。如果想要查找离您最近的办事处，请访问赛普拉斯所在地。

产品

ARM® Cortex® 微控制器	cypress.com/arm
汽车级	cypress.com/automotive
时钟与缓冲器	cypress.com/clocks
接口	cypress.com/interface
照明和电源控制	cypress.com/powerpsoc
存储器	cypress.com/memory
PSoC	cypress.com/psoc
触摸感应	cypress.com/touch
USB 控制器	cypress.com/usb
无线/射频	cypress.com/wireless

PSoC® 解决方案

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

赛普拉斯开发者社区

论坛 | 项目 | 视频 | 博客 | 培训 | 组件

技术支持

cypress.com/support

PSoC 是赛普拉斯半导体公司的注册商标。PSoC Creator 是赛普拉斯半导体公司的商标。 此处引用的所有其他商标或注册商标都归其各自所有者所有。

赛普拉斯半导体
198 Champion Court
San Jose, CA 95134-1709

电话 : 408-943-2600
传真 : 408-943-4730
网站地址 : www.cypress.com

©赛普拉斯半导体公司，2008-2016 年。本文件是赛普拉斯半导体公司及其子公司，包括 Spansion LLC（“赛普拉斯”）的财产。本文件，包括其包含或引用的任何软件或固件（“软件”），根据全球范围内的知识产权法律以及美国与其他国家签署条约由赛普拉斯所有。除非在本款中另有明确规定，赛普拉斯保留在该等法律和条约下的所有权利，且未就其专利、版权、商标或其他知识产权授予任何许可。如果软件并不附随有一份许可协议且贵方未以其他方式与赛普拉斯签署关于使用软件的书面协议，赛普拉斯特此授予贵方属个人性质的、非独家且不可转让的如下许可（无再许可权）（1）在赛普拉斯特软件著作权项下的下列许可权（一）对以源代码形式提供的软件，仅出于在赛普拉斯硬件产品上使用之目的且仅在贵方集团内部修改和复制软件，和（二）仅限于在有关赛普拉斯硬件产品上使用之目的将软件以二进制代码的形式向外部最终用户提供（无论直接提供或通过经销商和分销商间接提供），和（2）在被软件（由赛普拉斯公司提供，且未经修改）侵犯的赛普拉斯专利的权利主张项下，仅出于在赛普拉斯硬件产品上使用之目的制造、使用、提供和进口软件的许可。禁止对软件的任何其他使用、复制、修改、翻译或汇编。
在适用法律允许的限度内，赛普拉斯未对本文件或任何软件作出任何明示或暗示的担保，包括但不限于关于适销性和特定用途的默示保证。在适用法律允许的限度内，赛普拉斯保留更改本文件的权利，届时将不另行通知。赛普拉斯不对因应用或使用本文件所述任何产品或电路引起的任何后果负责。本文件，包括任何样本设计信息或程序代码信息，仅为参考之目的提供。文件使用人应负责正确设计、计划和测试信息应用和由此生产的任何产品的功能和安全性。赛普拉斯产品不应被设计为、设定为或授权用作武器操作、武器系统、核设施、生命支持设备或系统、其他医疗设备或系统（包括急救设备和手术植入物）、污染控制或有害物质管理系统中的关键部件，或产品植入之设备或系统故障可能导致人身伤害、死亡或财产损失的其他用途（“非预期用途”）。关键部件指，若该部件发生故障，经合理预期会导致设备或系统故障或会影响设备或系统安全性和有效性的部件。针对由赛普拉斯产品非预期用途产生或相关的任何索赔、费用、损失和其他责任，赛普拉斯不承担全部或部分责任且贵方不应追究赛普拉斯之责任。贵方应赔偿赛普拉斯因赛普拉斯产品任何非预期用途产生或相关的所有索赔、费用、损失和其他责任，包括因人身伤害或死亡引起的索赔，并使之免受损失。
赛普拉斯、赛普拉斯徽标、Spansion、Spansion 徽标，及上述项目的组合，及 PSoC、CapSense、EZ-USB、F-RAM 和 Traveo 应视为赛普拉斯在美国和其他国家的商标或注册商标。敬请访问 cypress.com 获取赛普拉斯商标的完整列表。其他名称和品牌可能由其各自所有者主张为该方财产。