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.

AN204728

F²MC-8FX 家族 MB95310/370 系列 8 位微型控制器 I²C 源代码 API

相关器件系列：MB95310/370 系列

本文档介绍了 I²C 代码的 API。

目录

1 概要 1
2 MB95F310 I²C 寄存器 1
2.1 IBCR00 和 IBCR10（I²C 总线控制寄存器） 1
2.2 IBSR0（I²C 总线状态寄存器） 3
2.3 IDDR0（I²C 数据寄存器） 4
2.4 IAAR0（I²C 地址寄存器） 4
2.5 ICCR0（I²C 时钟控制寄存器） 5
3 在软件中实现 I²C 的步骤 6
4 I²C 代码示例 7
5 用法演示 10
5.1 使用寄存器的注意事项 10
6 更多信息 10
文档修改记录 11

1 概要

本文档介绍了 I²C 代码的 API。

MB95F310 MCU 有一个 I²C 模块，可同步传输数据至其他 I²C 设备。以下章节将描述 MB95F310 寄存器，源代码及其编译。

2 MB95F310 I²C 寄存器

本章描述了 MB95F310 I²C 寄存器。

MB95F310 有六个寄存器：总线控制寄存器 IBCR00，总线控制寄存器 IBCR10，总线状态寄存器 IBSR0，I²C 数据寄存器 IDDR0，I²C 地址寄存器 IAAR0，和时钟控制寄存器 ICCR0。

以下章节将详细描述这些寄存器。

2.1 IBCR00 和 IBCR10（I²C 总线控制寄存器）

这两个寄存器主要用于选择操作模式，启用或禁用中断，响应，普通调用响应，和 MCU 待机唤醒功能。

图 1 描述了寄存器 IBCR00。

图 1. IBCR00 Register

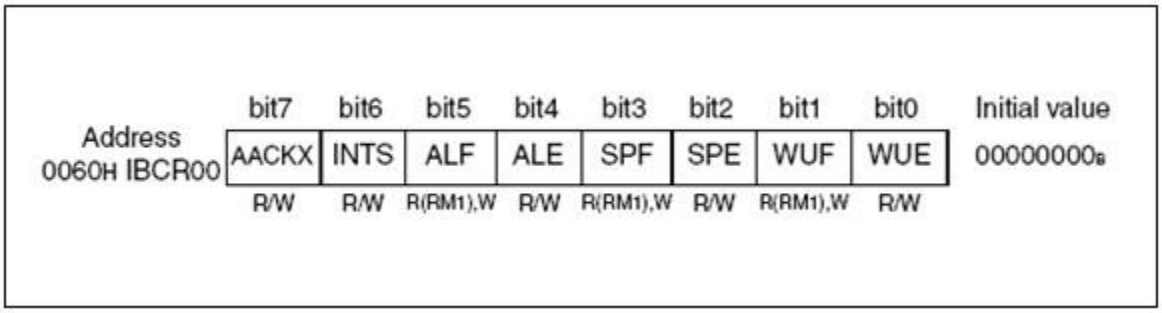


表 1 详细描述了 IBCR00 的位。

表 1. IBCR00 寄存器描述

寄存器	描述	
AACKX	在从设备模式下，该寄存器在接收第一个响应数据后，发送“1”。	
INTS	在第九个或第八个 SCL 周期中设置传送完成中断	0 ：第九个 SCL 周期的中断
		1 ：第八个 SCL 周期的中断
ALF	读取“1”表明检测到仲裁丢失；写“0”将清除丢失中断。	
ALE	写“1”将启用仲裁中断	
SPF	读取“1”表明检测到停止条件；写“0”将清除停止中断。	
SPE	写“1”启用停止中断。	
WUF	读取“1”表明检测到启动条件；写“0”将清除启动中断。	
WUE	写“1”将启用从待机模式中唤醒设备的功能；写“0”将禁用该功能。	

图 2 描述了寄存器 IBCR10

图 2. IBCR10 寄存器

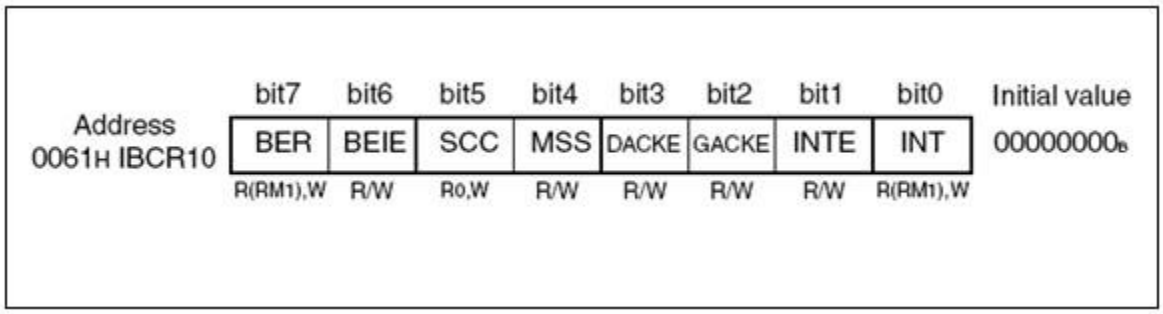


表 2 描述了寄存器 IBCR10

表 2. IBCR10 寄存器

寄存器	描述
BER	读取“1”表明检测到总线错误；烧写“0”将清除标志。
BEIE	写“1”将启用总线错误中断。
SCC	写“1”重新生成启动信号。
MSS	写“1”变成主设备模式，生成启动条件，开始地址数据传输。
	写“0”生成停止条件，并切换至从设备模式。
DACKE	写“1”启用数据响应。
GACKE	写“1”启用调用地址响应。
INTE	写“1”启用数据传输完成中断。
INT	读取“1”表明 1 字节数据传输完成。

2.2 IBSR0（I²C 总线状态寄存器）

这是一个只读寄存器，用于反馈 I²C 总线状态。

图 3 描述了寄存器 IBSR0。

图 3. IBSR0 寄存器

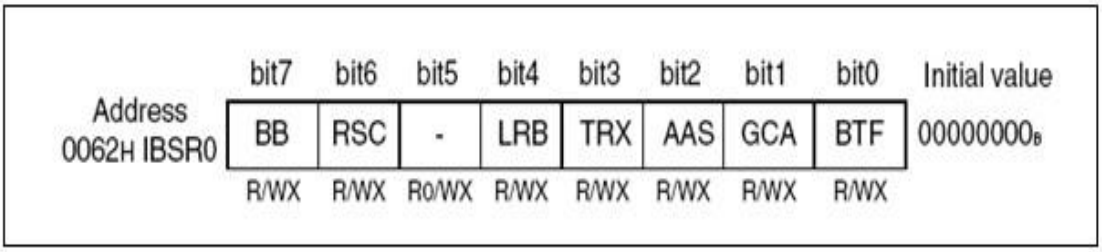


表 3 详细描述了 IBSR0 的位。

表 3. IBSR0 寄存器描述

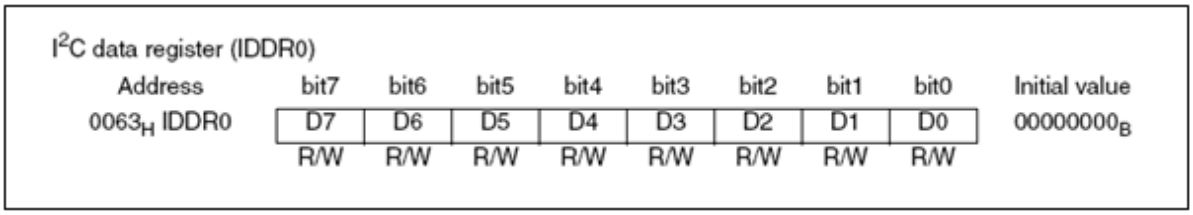
寄存器	描述	
BB	读取“1”表明总线繁忙，否则总线空闲。	
RSC	读取“1”表明检测到重复的启动条件。	
-	未使用	
LRB	读取“1”表明没有来自从设备芯片的响应。	
	读取“0”表明检测到响应，或检测到启动-停止条件。	
TRX	由写入地址的最后一位决定	读取“0”时为接收模式。
		读取“1”时为发送方式。
AAS	读取“1”表明 MCU 在从设备模式下寻址。	
	读取“0” 表明检测到启动或停止条件。	
GCA	读取“1”从设备模式接收到普通调用地址。	
FBT	读取“1”表明接收数据为第一地址数据。	

2.3 IDDR0（I²C 数据寄存器）

IDDR0 寄存器用于设置数据或地址，来发送和保持接收的数据或地址。

图 4 描述了寄存器 IDDR0。

图 4. IDDR0 寄存器

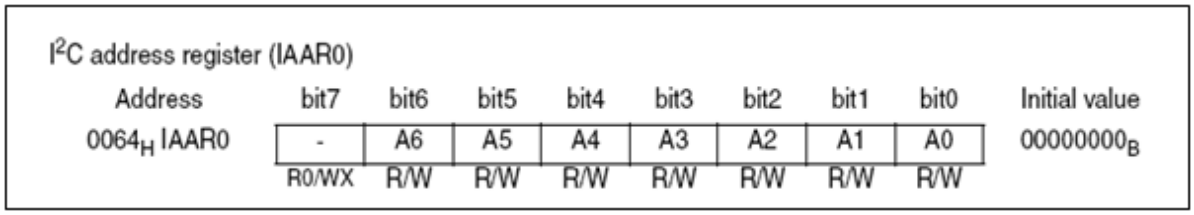


2.4 IAAR0（I²C 地址寄存器）

IAAR0 寄存器用于设置从设备地址，并在从设备模式中，把它与接收的地址作比较。

图 5 描述了寄存器 IAAR0。

图 5. IAAR0 寄存器



2.5 ICCR0（I²C 时钟控制寄存器）

ICCR0 寄存器用于启用 I²C 操作，并选择时钟频率。

图 6 描述了寄存器 ICCR0。

图 6. ICCR0 寄存器

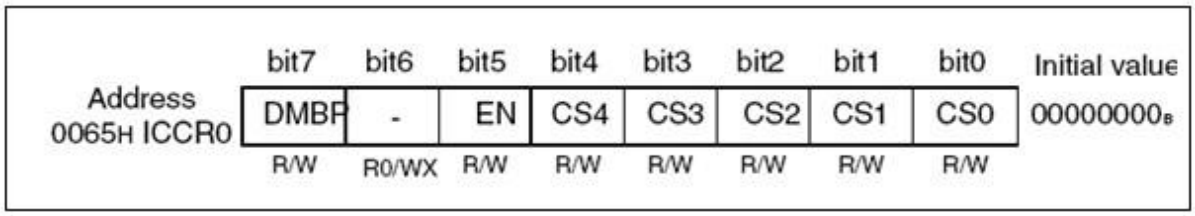


表 4 详细描述了 ICCR0 的位。

表 4. ICCR0 寄存器描述

寄存器	描述
DMBP	写“1”将旁路除数 m。
EN	写“1”将启用 I2C 操作。
CS4	除数 m，参见表 4-5
CS3	除数 m，参见表 4-5
CS2	除数 n，参见表 4-6
CS1	除数 n，参见表 4-6
CS0	除数 n，参见表 4-6

表 5 描述了除数 m。

表 5. 除数 m 描述

CS4	CS3	除数 m
0	0	5
0	1	6
1	0	7
1	1	8

表 6 描述了除数 n。

除数 n 描述

CS2	CS1	CS0	除数 n
0	0	0	4
0	0	1	8
0	1	0	22
0	1	1	38
1	0	0	98
1	0	1	128
1	1	0	256
1	1	1	512

3 在软件中实现 I²C 的步骤

本章描述了实现 I²C 的步骤。

I²C 寄存器分为初始化寄存器，状态寄存器，和数据寄存器。

以下部分将描述如何使用这些寄存器实现 I²C 功能。

- 第一步：初始化 I²C 。
  IBCR00 寄存器设置中断类型。
  IBCR10 寄存器设置 I²C 模式和响应生成条件。
  ICCR0 寄存器启用 I²C，并设置转换频率。

- 第二步：生成启动信号。
  IBSR0 寄存器反馈 I²C 状态。
  IDDR0 寄存器发送从设备地址。设置为主设备模式时，发送数据。

- 第三步：烧写数据至从设备。
  IBCR10 寄存器设置 I²C 可设置数据的条件。
  IDDR0 寄存器保存并发送数据。

- 第四步：设置读取以读取数据。
  IBCR10 寄存器反馈数据接收的状态。
  IDDR0 寄存器保存接收的数据。

- 第五步：设置条件检测响应。
  IBSR0 寄存器检测总线是否接收响应。

- 第六步：生成停止信号。
  IBCR10 寄存器设置停止模式。
  IBSR0 寄存器检测总线是否被释放。

4 I²C 代码示例

本章描述了 MB95F310 I²C 的 C 代码。

图 7 描述了初始函数。

图 7. 初始代码

```
void I2C_Init( void )
{
  ICCR0_EN  = 0;                  // clear I2C interface
  ICCR0_CS4 = 1;                  // set clock divider 'm' => 7
  ICCR0_CS3 = 0;
  ICCR0_CS2 = 1;                  // set clock divider 'n' => 98
  ICCR0_CS1 = 0;
  ICCR0_CS0 = 0;                  // Fsck = MCLK / (m * n +2) => 16MHz/(7*98 +2) = 16MHz/688 = 23kHz
  ICCR0_EN  = 1;                  // enable I2C interface
  IDDR0    = 000;          // clear data register
  IBCR00   = 0x04;         // enable address acknowledge bit,
  IBCR10   = 0x00;         // set to slave mode first, disable data acknowledge bit,
}
```

图 8 列出了启动代码。

图 8. 启动代码

```
void I2C_Start( unsigned char slave_address )
{
 do
 {
  IBCR10_BER = 0;             // clear bus error interrupt flag
  IDDR0     = slave_address;   // slave_address is sent out with start condition
  IBCR10_MSS = 1;             // set to master mode now and send start condition
  IBCR10_INT = 0;            // clear transfer end interrupt flag
  while(IBCR10_INT == 0);       // look if transfer is in process
 }
 while (IBCR10_BER == 1 | IBCR00_ALF == 1);       // retry if Bus-Error detected or arbitration lost
 while(IBSR0_LRB == 1)          // no acknowledge means device not ready
 {                        // maybe last write cycle not ended yet
  IBCR10_SCC = 1;             // try restart (= continue)
  while (IBCR10_INT == 0);      // wait that transfer is finished
 }
}
```

图 9 列出了烧写代码。

图 9. 烧写代码

```
void I2C_Write( unsigned char value )
{
 IDDR0  = value;              // load data or address in to register
 IBCR10_INT = 0;              // clear transfer end intrerupt flag
 while (IBCR10_INT == 0);     // look if transfer is in process
 I2C_Acknowlegde();           // wait for Acknowledge
}
```

图 10 列出了读取代码。

图 10. 读取代码

```
unsigned char I2C_Read( void )
{
 IBCR10_DACKE = 1;            // acknowledge has to be send
 IBCR10_INT = 0;              // clear transfer end interrupt flag
 while (IBCR10_INT == 0);     // wait that transfer is finished
 return(IDDR0);               // read received data out
}
```

图 11 列出了响应代码。

图 11. 响应代码

```
void I2C_Acknowlegde( void )
{
  while(IBSR0_LRB == 1);          // no anwser from slave, program stucks,
                    // here a timeout mechanism should be implemented
}
```

图 12 列出了停止代码。

图 12. 停止代码

```
void I2C_Stop( void )
{
 while (IBCR10_INT == 0);     // wait that transfer is finished
 IBCR10_MSS = 0;              // change to slave and release stop condition
 IBCR10_INT = 0;              // clear transfer end interrupt flag
 while (IBSR0_BB);            // wait till bus free
}
```

图 13 描述了烧写函数。

图 13. 烧写函数

```
Void Write_I2C_Proc(unsigned char MainAddr, unsigned char SubAddr, unsigned char I2Cdata)
{
        I2C_Init();
        I2C_Start(MainAddr);
        I2C_Acknowlegde();
        I2C_Write(SubAddr);
        I2C_Acknowlegde();
        I2C_Write(SubAddr);
        I2C_Acknowlegde();
        I2C_Stop();
}
```

图 14 描述了读取函数。

图 14. 读取函数

```
unsigned char RD_I2C( unsigned char Main_Addr, unsigned char Sub_Addr )
{
        unsigned char Temp;

        I2C_Start( (Main_Addr ) | I2C_WRITE );
        I2C_Write( Sub_Addr );
        I2C_Continue( (Main_Addr ) | I2C_READ );
       Temp = I2C_Read();
        I2C_Stop();
        return (Temp);
}
```

5 用法演示

本章描述了使用寄存器实现 I^2C 时的注意事项。

5.1 使用寄存器的注意事项

- 开始 I^2C 时，I^2C 总线必须为空闲。
- 如果主设备寄存器 IBSR0_LRB 为“1”，从设备不接收地址或数据。
- 读取地址比烧写地址大 1。

6 更多信息

如欲了解有关如何使用 MB95310 EV-board、BGM Adaptor 和 SOFTUNE 的更多详情，敬请参见 EV-Board MB2146-450-E 用户手册，或者访问以下网址：

http://www.cypress.com/documentation/application-notes/mb95310370-mb2146-450-e-lcd-evb-user-manual

文档修改记录

文档标题：AN204728 – F²MC-8FX 家族 MB95310/370 系列 8 位微型控制器 I^2C 源代码 API

文档编号：002-05685

修订版	ECN	变更者	提交日期	变更说明
**	-	HUAL	11/13/2009	初稿
*A	5320360	HUAL	06/23/2016	已将 Spansion 应用手册《MCU-AN-500051-Z-10》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 和 CapSense 是赛普拉斯半导体公司的注册商标。PSoC Designer 和 CapSense Express 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，2009-2016 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。