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.

AN205013

F²MC-8FX 家族 8 位微型控制器 MB95F430 系列 运算放大器

本文档介绍了如何在 MB95F430 系列上使用运算放大器功能。

目录

1 介绍 1
2 放大器概要.....2
2.1 运算放大器的结构图.....3
2.2 运算放大器的引脚 4
2.3 OPAMP 控制寄存器 5
3 运算放大器的操作.....7
4 放大器的设置程序.....8
5 放大器驱动器.....8
5.1 外围设备的使用.....8
5.2 驱动代码 8
6 典型应用.....10
6.1 硬件设计 10
6.2 范例代码 11
7 更多信息.....11
A 附录 12
A.1 范例代码 12
文档修改记录.....15

1 介绍

本文档介绍了如何在 MB95F430 系列上使用运算放大器功能。

第二章简要介绍了运算放大器。

第三章介绍了运算放大器的操作。

第四章介绍了运算放大器的设置程序。

第五章介绍了放大器驱动器。

第六章介绍了放大器的应用演示。

2 放大器概要

运算放大器用于感知地面电流，它支持 A/D 转换前的前端模拟信号调节，可在闭环模式或独立开环模式中运行。

- 闭环模式

运算放大器可配置为同相闭环运算放大器。

它有六个闭合回路增益选项（软件选择），可根据不同的电压值感知地面电流。

No.	Gain
1	10 V/V
2	20 V/V
3	30 V/V
4	40 V/V
5	50 V/V
6	60 V/V

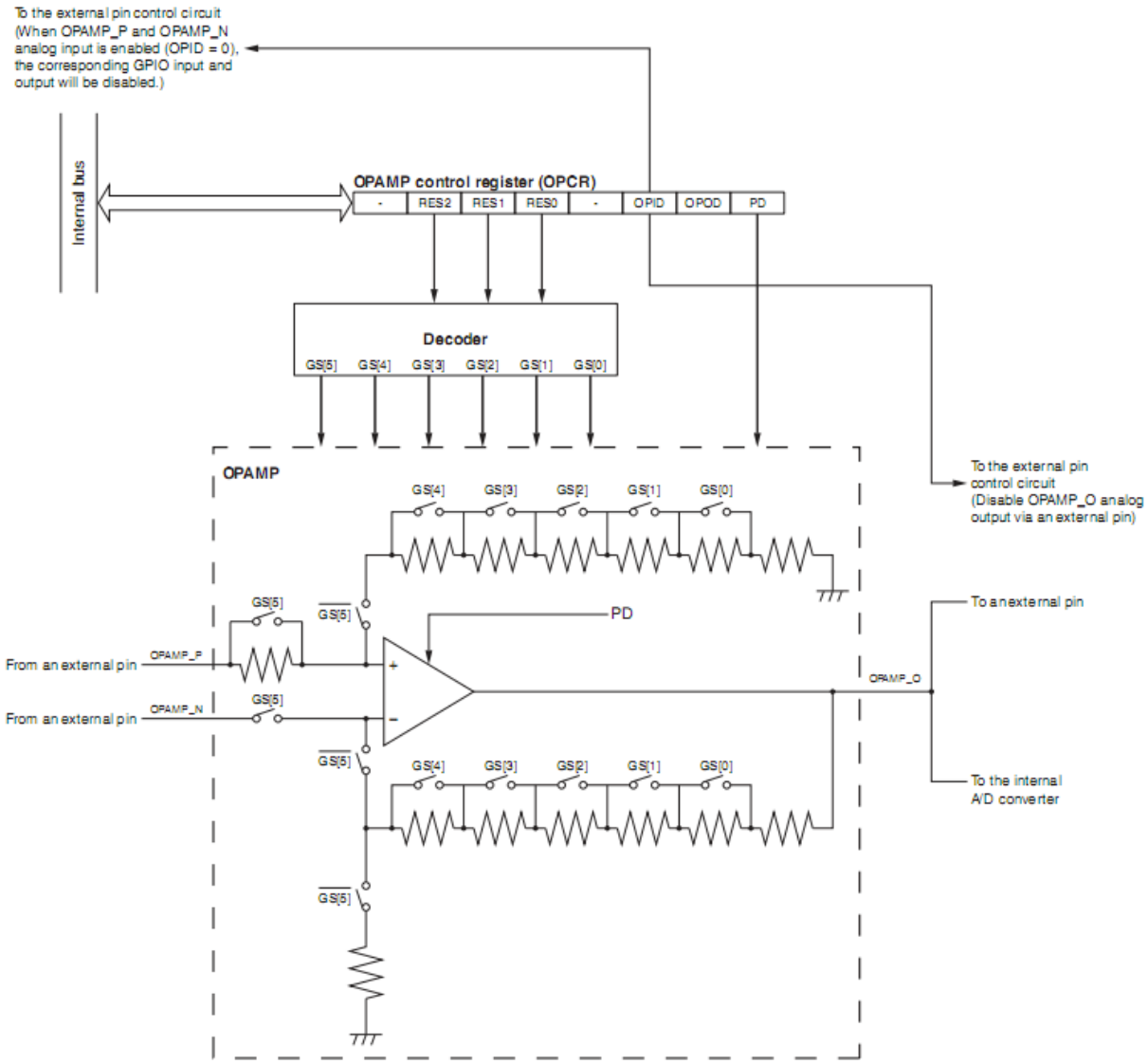
- 独立开环模式

在该模式下，运算放大器输入引脚被连接至外部信号，没有任何输出反馈。

该模式的外部电阻器让用户有更多的增益选择。

2.1 运算放大器的结构图

图 2-1. 运算放大器的结构图



2.2 运算放大器的引脚

OPAMP 把 OPAMP_P 引脚和 OPAMP_N 引脚用作运算放大器的模拟输入引脚；把 OPAMP_O 引脚用作运算放大器的模式输出引脚。

GS [5] 设置为“1B”且 GS [4:0] 设置为“00000B”时，OPAMP 用作一个独立开环运算放大器。

GS [5] 设置为“0B”时，OPAMP 用作一个同相闭合回路运算放大器，通过软件提供六种不同的闭环增益设置。

Pin Name	Pin Function	I/O Type	Pull-up Option	Standby Control	Settings Required for Using The Pin	Default Status
P60/OPAMP_P	GPIO/ OPAMP positive analog input	CMOS input/ CMOS output/ Analog input	Unavailable	Available	OPCR:OPID = 0 (Enables analog input)	GPIO input disabled; GPIO output disabled; analog input enabled
P61/OPAMP_N	GPIO/ OPAMP negative analog input	CMOS input/ CMOS output/ Analog input			OPCR:OPID = 0 (Enables analog input)	GPIO input disabled; GPIO output disabled; analog input enabled
P62/OPAMP_O	GPIO/ OPAMP analog output	CMOS input/ CMOS output/ Analog output			OPCR:OPOD = 0 (Enables analog output)	GPIO input enabled; GPIO output disabled; analog output disabled

2.3 OPAMP 控制寄存器

OPAMP 控制寄存器（OPCR）用于开启-关闭 OPAMP，启用-禁用 OPAMP 模拟输出，以及启用-禁用 OPAMP 模拟输入。

该寄存器还可以把 OPAMP 设置为一个独立开环运算放大器，或一个有六种不同闭环增益设置（可通过软件选择）的同相闭环运算放大器。

图 2-2. OPAMP 控制寄存器

Address	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Initial value
003EH	-	RES2	RES1	RES0	-	OPID	OPOD	PD	00000011B
	R0/WX	R/W	R/W	R/W	R0/WX	R/W	R/W	R/W	

PD	OPAMP power down control bit
0	Turns on the OPAMP.
1	Tunrs off the OPAMP.

OPOD	OPAMP analog output disable bit
0	Enables OPAMP analog output.
1	Disables OPAMP analog output.

OPID	OPAMP analog input disable bit
0	Enables OPAMP analog input.
1	Disables OPAMP analog input.

RES2	RES1	RES0	OPAMP loop gain setting bits
0	0	0	The OPAMP operates with a 60 V/V closed loop gain.
0	0	1	The OPAMP operates with a 50 V/V closed loop gain.
0	1	0	The OPAMP operates with a 40 V/V closed loop gain.
0	1	1	The OPAMP operates with a 30 V/V closed loop gain.
1	0	0	The OPAMP operates with a 20 V/V closed loop gain.
1	0	1	The OPAMP operates with a 10 V/V closed loop gain.
1	1	0	The OPAMP operates in open loop mode.
1	1	1	Setting prohibited

R/W : Readable/writable (The read value is the same as the write value.)
R0/WX : The read value is "0". Writing a value to it has no effect on operation.
- : Undefined bit
: Initial value

- OPAMP 控制寄存器（OPCR）位的功能

Bit name		Function
bit7	Undefined bit	The read value is always "0". Writing a value to it has no effect on operation.
bit6 to bit4	RES2, RES1, RES0: OPAMP loop gain setting bits	These bits select an OPAMP loop gain in closed loop mode from six options and can set the OPAMP to operate in open loop mode.
bit3	Undefined bit	The read value is always "0". Writing a value to it has no effect on operation.
bit2	OPID: OPAMP analog input disable bit	This bit enables and disables OPAMP analog input. **Writing "0":**enables OPAMP analog input. **Writing "1":**disables OPAMP analog input.
bit1	OPOD: OPAMP analog output disable bit	This bit enables and disables OPAMP analog output. **Writing "0":**enables OPAMP analog output. **Writing "1":**disables OPAMP analog output.
bit0	PD: OPAMP power down control bit	This bit turns on and off the OPAMP. **Writing "0":**turns on the OPAMP. **Writing "1":**turns off the OPAMP.

- OPAMP 运算模式设置

RES2	RES1	RES0	OPAMP Loop Gain Settings
0	0	0	The OPAMP operates with a 60 V/V closed loop gain.
0	0	1	The OPAMP operates with a 50 V/V closed loop gain.
0	1	0	The OPAMP operates with a 40 V/V closed loop gain.
0	1	1	The OPAMP operates with a 30 V/V closed loop gain.
1	0	0	The OPAMP operates with a 20 V/V closed loop gain.
1	0	1	The OPAMP operates with a 10 V/V closed loop gain.
1	1	0	The OPAMP operates in open loop mode.
1	1	1	Setting prohibited

注意：

- OPAMP 运行时，可以修改 RES2，RES1 和 RES0 的设置。但在 OPAMP 输出稳定前，不要使用 OPAMP 的输出信号，也不要执行 A/D 转换。
- 修改 RES2，RES1 和 RES0 设置前，建议关闭运算放大器。

3 运算放大器的操作

使用软件设置 OPCR 寄存器的 PD 位可激活运算放大器。根据 OPCR 寄存器中 RES2，RES1 和 RES0 位的设置，运算放大器可在闭环模式或开环模式下运行。

- 软件激活运算放大器

根据图 3-1 中的设置激活运算放大器。

图 3-1. 激活运算放大器的设置

	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
OPCR	-	RES2	RES1	RES0	-	OPID	OPOD	PD
	×	❍	❍	❍	×	0	0	0

❍ : Bit to be used
× : Unused bit
0 : Set to "0"

按照上图设置好 OPCR 寄存器中的所有位，运算放大器在稳定后，开始操作。

- 闭环模式的 OPAMP 操作

在激活前，事先设置 OPCR 寄存器的 RES [2:0] 为“000B”，“001B”，“010B”，“011B”，“100B ”或“101B”，可让运算放大器在闭环模式中运行。

闭环模式有六种不同的闭合回路增益。设置 OPCR 中的 RES[2:0] 至相应值，可选择期望的闭环回路增益。

注意：

- 建议在闭环模式下，接地 P61/OPAMP_N 引脚。
- OPAMP 运行时，可以修改 RES2，RES1 和 RES0 的设置。但在 OPAMP 输出稳定前，不要使用 OPAMP 的输出信号，也不要执行 A/D 转换。
- 修改 RES2，RES1 和 RES0 设置前，建议关闭运算放大器。

- 开环模式下的 OPAMP 操作

在激活前，事先设置 OPCR 寄存器的 RES [2:0] 为“110B”，可让运算放大器在开环模式中运行。

注意：

- OPAMP 运行时，允许开环模式和闭环模式之间的切换。但在 OPAMP 输出稳定前，不要使用 OPAMP 的输出信号，也不要执行 A/D 转换。

4 放大器的设置程序

本章举例说明了运算放大器的设置程序。

初始设置

1. 设置 OPCR：OPID 和 OPCR：OPOD 为“0”启用 OPAMP 模拟输入和 OPAMP 模拟输出。
2. 设置 OPCR 中的反馈电阻器和 RES [2:0]。
3. 设置 OPCR：PD 为“0”启动运算放大器。
4. 等待运算放大器稳定。
5. 如果需要，启动 A/D 转换。

5 放大器驱动器

本章描述了 OPAMP 驱动器。

5.1 外围设备的使用

MCU 引脚的使用如下：

OPAMP_N：用作放大器负输入

OPAMP_P：用作放大器正输入

OPAMP_O：用作放大器输出

5.2 驱动代码

5.2.1 一般定义

```
typedef unsigned char  BOOLEAN;
typedef unsigned char  INT8U;            /* Unsigned  8 bit quantity */
typedef signed   char  INT8S;            /* Signed    8 bit quantity */
typedef unsigned int    INT16U;           /* Unsigned 16 bit quantity */
typedef signed   int    INT16S;           /* Signed   16 bit quantity */
typedef unsigned long       INT32U;           /* Unsigned 32 bit quantity */
typedef signed   long  INT32S;           /* Signed   32 bit quantity */

#define BOOL               BOOLEAN
#define BYTE               INT8U
#define UBYTE              INT8U
#define WORD               INT16U
#define UWORD              INT16U
#define LONG               INT32S
#define ULONG       INT32U
#define UCHAR       INT8U
#define UINT                     INT16U
#define DWORD              INT32U


#define TRUE           1
#define FALSE          0

#define BYTE_LO(w)     ((UBYTE)(w))
#define BYTE_HI(w)     ((UBYTE)(((UWORD)(w)>>8)&0xFF))
```

5.2.2 放大器程序

void AmpOpenLoop()

返回 : 无

参数 : 无

描述 : 开环设置

示例 : AmpOpenLoop();

```
void AmpOpenLoop()
{
       DDR6_P60=0;
       DDR6_P61=0;
       DDR6_P62=1;
       OPCR=0x60;//Amplifier gain is R3/R1
}
```

void AmpCloseLoop()

返回 : 无

参数 : 无

描述 : 闭环设置

示例 : AmpCloseLoop();

```
void AmpCloseLoop()
{
       DDR6_P60=0;
       DDR6_P61=0;
       DDR6_P62=1;
       OPCR=0x40;//Amplifier gain is 20V/V
}
```

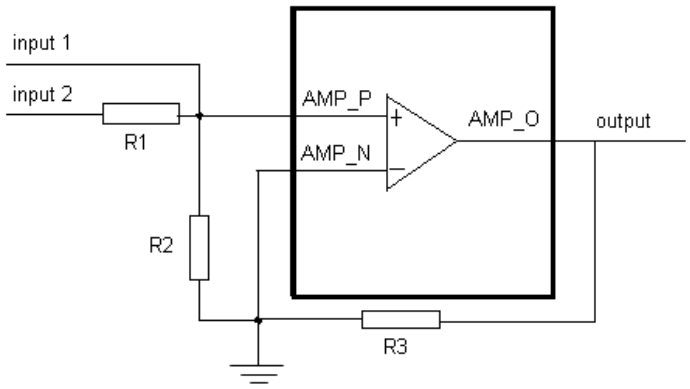
6 典型应用

本章介绍了运算放大器的典型应用。

6.1 硬件设计

该应用将测试 MB95F430K 的运算放大器。硬件设计如下图所示，R1，R2 和 R3 用于开环放大器。

图 6-1. 硬件设计



6.2 范例代码

void main(void)

返回 : 无

参数 : 无

描述 : 系统主程序

示例 : main();

```
void main(void)
{
       __DI();
       __set_il(3);
       InitIrqLevels();

       WDTH =0xA5;//Disable WTG
       WDTL =0x96;

       WATR =0xEE;
       SYCC =0xF0;//Main Clock
       SYCC2=0xF4;//Main Clock
       SYSC  =0xBC;//BUZZ(P01)
       SYSC2 =0x02;//PPG(P73),Disable I2C
       while(!STBC_MRDY);
       __EI();

       AmpOpenLoop();
       AmpCloseLoop();
}
```

7 更多信息

如欲获取有关任何技术问题的帮助，敬请联系您当地的支持团队。

A 附录

A.1 范例代码

main.c

```
#include "mb95430.h"
#include "TypeDef.h"

/*-----------------------------------------------------------------------------*/
/* Amplifier Setting
/*-----------------------------------------------------------------------------*/
void AmpOpenLoop()
{
      DDR6_P60=0;
      DDR6_P61=0;
      DDR6_P62=1;
      OPCR=0x60;//Amplifier gain is R3/R1
}

void AmpCloseLoop()
{
      DDR6_P60=0;
      DDR6_P61=0;
      DDR6_P62=1;
      OPCR=0x40;//Amplifier gain is 20V/V
}

void main(void)
{
      __DI();
      __set_il(3);
      InitIrqLevels();

      WDTH =0xA5;
      WDTL =0x96;

      WATR =0xEE;
      SYCC =0xF0;//Main Clock
      SYCC2=0xF4;//Main Clock
      SYSC  =0xBC;//BUZZ(P01)
      SYSC2 =0x02;//PPG(P73),Disable I2C
      while(!STBC_MRDY);

      __EI();

      AmpOpenLoop();
      AmpCloseLoop();
}
```

VECTORS.C

```
 #include "mb95430.h"

void InitIrqLevels(void)
{
/*  ILRx                IRQs defined by ILRx */

    ILR0 = 0xFF;      //  IRQ0:  external interrupt ch0 | ch4
                      //  IRQ1:  external interrupt ch1 | ch5
```

```
                    //  IRQ2:  external interrupt ch2 | ch6
                    //  IRQ3:  external interrupt ch3 | ch7

    ILR1 = 0xFF;    //  IRQ4:  UART/SIO ch0
                    //  IRQ5:  8/16-bit timer ch0 (lower)
                    //  IRQ6:  8/16-bit timer ch0 (upper)
                    //  IRQ7:  Output Compare ch0

    ILR2 = 0xFF;    //  IRQ8:  Output Compare ch1
                    //  IRQ9:  none
                    //  IRQ10: Voltage Compare ch0
                    //  IRQ11: Voltage Compare ch1

    ILR3 = 0xFF;    //  IRQ12: Voltage Compare ch2
                    //  IRQ13: Voltage Compare ch3
                    //  IRQ14: 16-bit free run timer
                    //  IRQ15: 16-bit PPG0

    ILR4 = 0xFF;    //  IRQ16: I2C ch0
                    //  IRQ17: none
                    //  IRQ18: 10-bit A/D-converter
                    //  IRQ19: Timebase timer

    ILR5 = 0xFF;    //  IRQ20: Watch timer
                    //  IRQ21: none
                    //  IRQ22: none
                    //  IRQ23: Flash Memory
}

/*-----------------------------------------------------------------------
  Prototypes

  Add your own prototypes here. Each vector definition needs is proto-
  type. Either do it here or include a header file containing them.
-----------------------------------------------------------------------*/
__interrupt void DefaultIRQHandler(void);

/*-----------------------------------------------------------------------
  Vector definiton

  Use following statements to define vectors.
  All resource related vectors are predefined.
  Remaining software interrupts can be added hereas well.
-----------------------------------------------------------------------*/
#pragma intvect DefaultIRQHandler 0   //  IRQ0:  external interrupt ch0 | ch4
#pragma intvect DefaultIRQHandler 1   //  IRQ1:  external interrupt ch1 | ch5
#pragma intvect DefaultIRQHandler 2   //  IRQ2:  external interrupt ch2 | ch6
#pragma intvect DefaultIRQHandler 3   //  IRQ3:  external interrupt ch3 | ch7

#pragma intvect DefaultIRQHandler 4   //  IRQ4:  UART/SIO ch0
#pragma intvect DefaultIRQHandler 5   //  IRQ5:  8/16-bit timer ch0 (lower)
#pragma intvect DefaultIRQHandler 6   //  IRQ6:  8/16-bit timer ch0 (upper)
#pragma intvect DefaultIRQHandler 7   //  IRQ7:  Output Compare ch0

#pragma intvect DefaultIRQHandler 8   //  IRQ8:  Output Compare ch1
#pragma intvect DefaultIRQHandler 9   //  IRQ9:  none
#pragma intvect DefaultIRQHandler 10  //  IRQ10: Voltage Compare ch0
#pragma intvect DefaultIRQHandler 11  //  IRQ11: Voltage Compare ch1
```

```
#pragma intvect DefaultIRQHandler 12  //  IRQ12: Voltage Compare ch2
#pragma intvect DefaultIRQHandler 13  //  IRQ13: Voltage Compare ch3
#pragma intvect DefaultIRQHandler 14  //  IRQ14: 16-bit free run timer
#pragma intvect DefaultIRQHandler 15  //  IRQ15: 16-bit PPG0

#pragma intvect DefaultIRQHandler 16  //  IRQ16: I2C ch0
#pragma intvect DefaultIRQHandler 17  //  IRQ17: none
#pragma intvect DefaultIRQHandler 18  //  IRQ18: 10-bit A/D-converter
#pragma intvect DefaultIRQHandler 19  //  IRQ19: Timebase timer

#pragma intvect DefaultIRQHandler 20  //  IRQ20: Watch timer
#pragma intvect DefaultIRQHandler 21  //  IRQ21: none
#pragma intvect DefaultIRQHandler 22  //  IRQ22: none
#pragma intvect DefaultIRQHandler 23  //  IRQ23: Flash Memory

/*------------------------------------------------------------------------
  DefaultIRQHandler()

  This function is a placeholder for all vector definitions.
  Either use your own placeholder or add necessary code here
  (the real used resource interrupt handlers should be defined in the main.c).
------------------------------------------------------------------------*/
__interrupt void DefaultIRQHandler(void)
{
    __DI();                                     // disable interrupts
    while(1)
        __wait_nop();                   // halt system
}
```

文档修改记录

文档标题：AN205013 - F²MC-8FX 家族 8 位微型控制器 MB95F430 系列 运算放大器

文档编号：002-05713

修订版	ECN	变更者	提交日期	变更说明
**	—	HUAL	03/22/2010	初稿
*A	5327392	HUAL	06/28/2016	已将 Spansion 应用手册《MCU-AN-500080-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 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，2010-2016 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。