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.

AN205336

F²MC-8FX 家族 MB95200H/210H 系列监视定时器

本应用笔记介绍了如何使用监视定时器，并描述了监视定时器的功能并举例说明。

目录

1 概要 1
2 监视定时器 1
2.1 主要特征 1
2.2 框图 2
2.3 寄存器 3
2.4 NVR（非易失性寄存器）功能 4
3 间隔时间 5
4 使用方法和示例 6
4.1 监视定时器的功能和操作 6
4.2 软件监视定时器 7
4.3 硬件监视定时器 8
5 监视定时器的使用注意事项 9
6 更多信息 9
6.1 样本代码 10
文档修改记录 14

1 概要

本应用笔记介绍了如何使用监视定时器。

本应用笔记描述了监视定时器的功能并举例说明。

2 监视定时器

本章介绍了监视定时器的基本功能。

2.1 主要特征

监视定时器用作计数器，可防止程序失控。监视定时器一旦启动，其计数器需以指定的间隔时间定期清零。如果定时器在第 3 章所述的事先设定间隔时间内未清零，则发生监视复位。监视定时器未清零可能是程序原因引发的，例如程序进入无限循环，堆栈超限或其他原因致使 CPU 挂机。

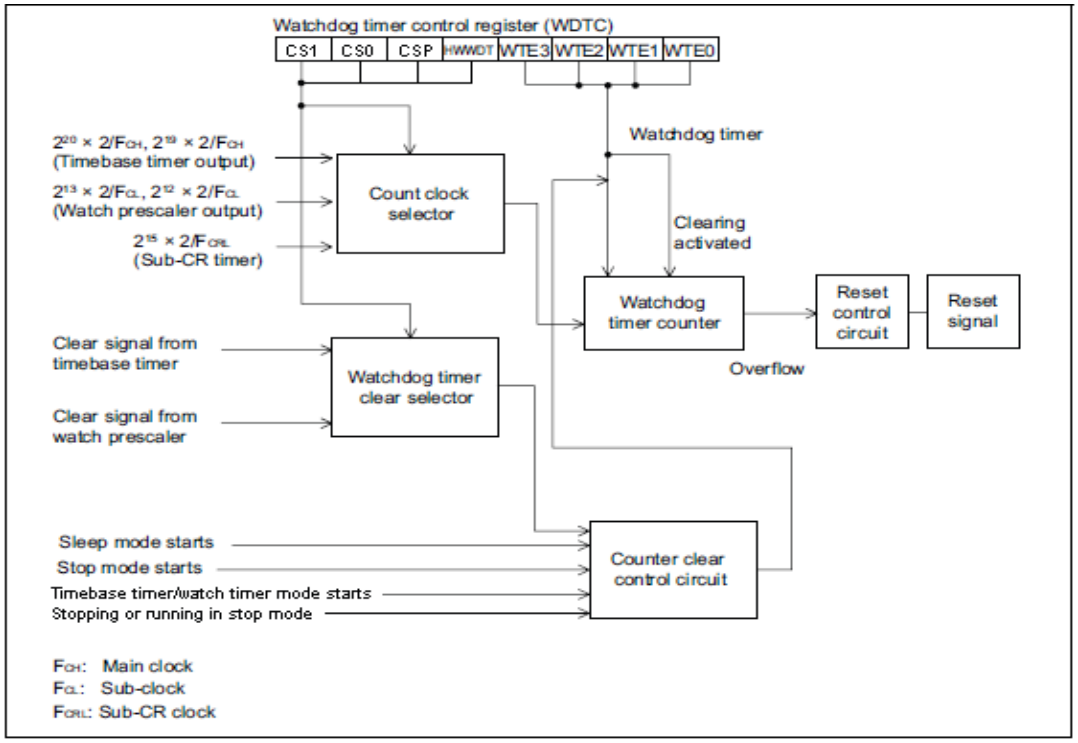
监视定时器具有以下特征：

- 计数时钟选择器
- 监视定时器的计数器
- WDT 计数器溢出时，复位控制电路用于生成复位信号
- 监视定时器清零选择器用于选择监视定时器清零信号
- 计数器定时器控制电路
- 硬件和软件监视定时器

2.2 框图

图 1 是监视定时器的内部框图。

图 1. 监视定时器的框图



2.3 寄存器

关于寄存器的详细设定，参照 MB95200H/210H 系列硬件手册的第 11 章和 22 章。

2.3.1 监视定时器控制寄存器(WDTC)

该寄存器用于启动或清零监视定时器。

图 2. WDTC

Address	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Initial value
$000C_H$	CS1	CS0	CSP	HWWDT	WTE3	WTE2	WTE1	WTE0	
software	R/W	R/W	R/W	R0,WX	R0/W	R0/W	R0/W	R0/W	00000000_B
hardware	R0/WX	R0/WX	R1/WX	R1,WX	R0/W	R0/W	R0/W	R0/W	00110000_B

2.3.2 监视定时器选择 ID 寄存器(WDTH、WDTL)

这两个寄存器用于选择硬件和软件监视定时器。

图 3. 监视定时器选择 ID 寄存器(WDTH、WDTL)

Address	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Initial value
$0FEB_H$ WDTH	WDTH7	WDTH6	WDTH5	WDTH4	WDTH 3	WDTH 2	WDTH 1	WDTH 0	xxxxxxxx
$0FEC_H$ WDTL	WDTL7	WDTL6	WDTL5	WDTL4	WDTL 3	WDTL 2	WDTL 1	WDTL 0	xxxxxxxx
	R/WX	R/WX	R/WX	R/WX	R/WX	R/WX	R/WX	R/WX	

R/W ：可读/可写（读值与写值相同）

R/WX ：只读（可读，写值无效）

R0/WX ：未定义位（读值为“0”，写值无效）

R1/WX ：未定义位（读值为“1”，写值无效）

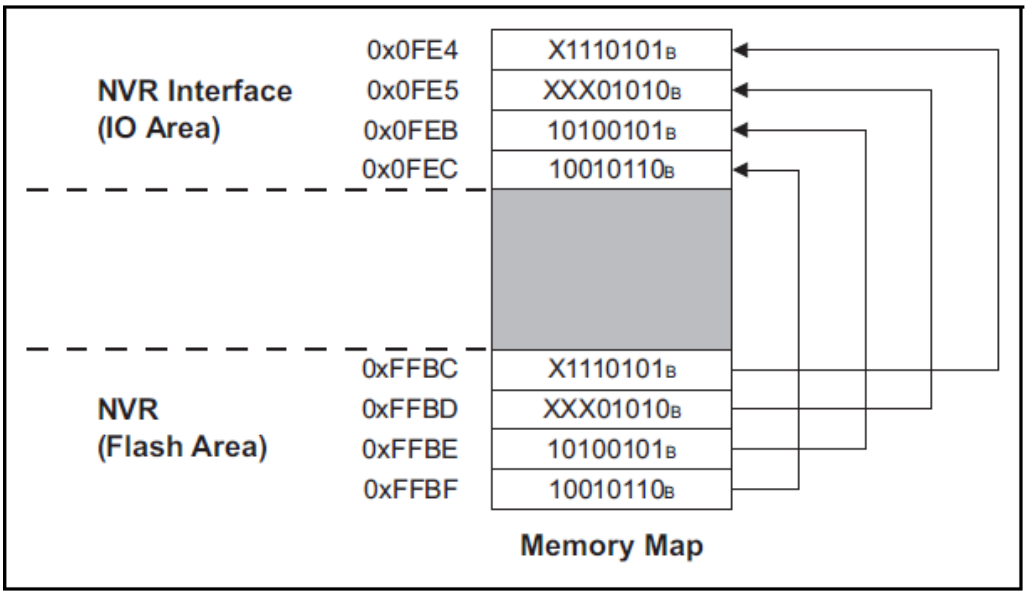
R0/W ：只写（可写，读“0”）

2.4 NVR（非易失性寄存器）功能

NVR 接口可使用户通过修改 16 位监视定时器选择 ID，来选择硬件或软件监视定时器。需要注意的是 CPU 工作期间，不可修改监视定时器选择 ID。因此，请先修改闪存区的 NVR，然后 MCU 在复位后自动将修改值复制到 NVR 接口（IO 区）。

图 4 显示的是闪存产品的串行编程连接的基本构成。

图 4. 复位时的 NVR 检索



需注意的是 SWWDT 和 HWWDT 在同一时间内只能有一个在工作，NVR 用于决定哪一个工作。

复位后 WDTH 和 WDTL 的 16 位从闪存地址 $FFBE_H$、$FFBF_H$ 下载。初始值由 NVR 闪存区预先下载的值决定。

对地址 $FFBE_H$、$FFBF_H$ 写值选择监视定时器的模式。

表 1 是关于监视定时器选择 ID 的功能。

表 1. 监视定时器选择 ID

WDTH[7:0]、WDTL[7:0]	功能
$A596_H$	禁止 HWWDT 并使能 SWWDT。
$A597_H$	选择 HWWDT 并禁止 SWWDT。[可在以下待机模式中停止（停止/休眠/时基定时器/计时模式）。]
其他值	选择 WHWDT 并禁止 SWWDT。[在以下待机模式中继续工作（停止/休眠/时基定时器/计时模式）。]

3 间隔时间

本节介绍了监视定时器的间隔时间。

监视定时器的间隔时间如表 2 所示。若未清零监视定时器的计数器，在最短和最长时间内发生监视器复位。

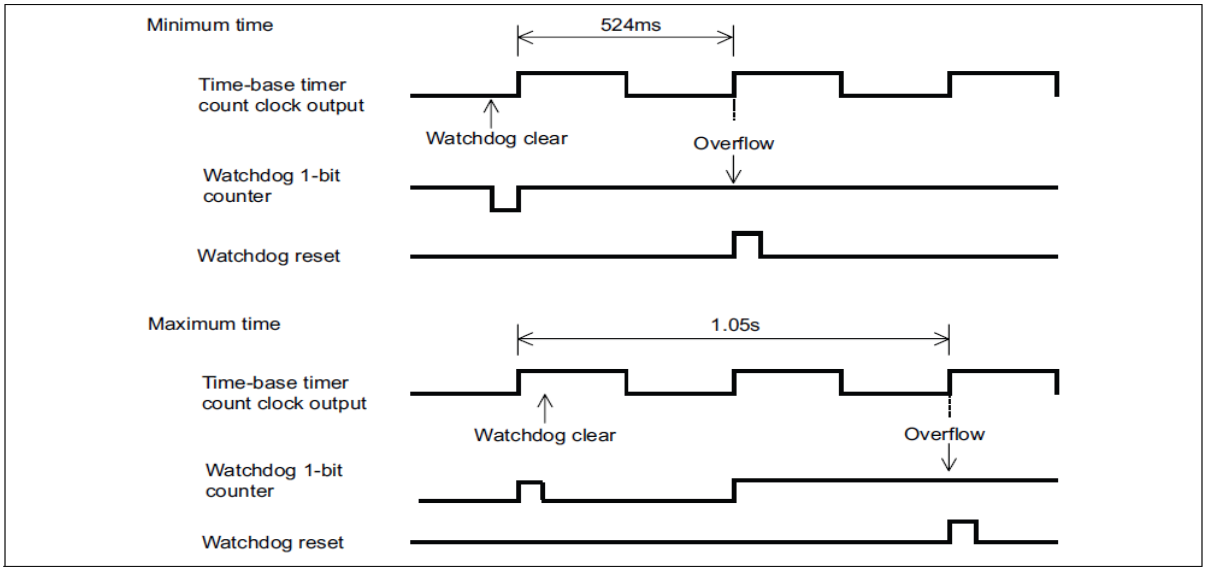
表 2. 监视定时器的间隔时间

计数时钟类型	计数时钟切换位 CS[1:0]、CSP	间隔时间		
		最短		最长
时基定时器输出（主时钟 = 4 MHz）	000_B(SWWDT)	$2^{21}/F_{CH}$	524 ms	1.05 s
	010_B(SWWDT)	$2^{20}/F_{CH}$	262 ms	524 ms
计时预分频器输出（副时钟 = 32.768 KHz）	100_B(SWWDT)	$2^{14}/F_{CL}$	500 ms	1.00 s
	110_B(SWWDT)	$2^{13}/F_{CL}$	250 ms	500 ms
副 CR 时钟（副 CR 时钟 = 50-200 KHz）	$XX1_B$(SWWDT) or HWWDT*1	$2^{16}/F_{CRL}$	328 ms	2.62 s

*1: CS [1:0] = 00_B、CSP = 1_B （只读）

间隔时间取决于清零监视定时器的时序。图 5 显示的是选择时基定时器输出 $2^{21}/F_{CH}$ （F_{CH}:主时钟）作为计数时钟（主时钟= 4 MHz）时，监视定时器的清零时序和间隔时间的关联性。

图 5. 监视定时器的清零时序和间隔时间



注：程序必须在最短时间内清零监视定时器的计数器。

4 使用方法和示例

本章介绍了监视定时器的功能并举例说明。

本节介绍了监视定时器的使用方法并举例说明。

4.1 监视定时器的功能和操作

对 NVR 闪存区地址 $FFBE_H$、$FFBF_H$ 写值以选择监视定时器的工作模式。关于模式选定设置，详情参照表 1。

监视定时器有两个工作模式。

- 软件监视

1. 对闪存地址 $FFBE_H$、$FFBF_H$ 写“$A596_H$”（使能软件监视定时器），复位后该值复制到监视定时器选择 ID 寄存器 WDTH/WDTL ($0FEB_H$/$0FEC_H$)。
2. 复位后第 1 次对监视定时器控制寄存器的监视控制位(WDTC: WTE3 ～ WTE0)写“0101B”时，监视定时器启动。监视定时器控制寄存器的计数时钟切换位(WDTC:CS1,CS0,CSP)也必须同时设置。
3. 一旦监视定时器启动，只有复位可停止其运行。监视计数器根据 WDTC:CS1,CS0,CSP 的设置以指定间隔定期清零。关于间隔时间，详情参照表 3-1。

- 硬件监视

1. 在闪存地址 $FFBE_H$、$FFBF_H$ 上写“$A597_H$”（待机模式下停止 WDT）或其他值（在全模式下使能 WDT）。勿写“$A596_H$”或“$A597_H$”，复位后该值将被复制到监视定时器选择 ID 寄存器 WDTH/WDTL ($0FEB_H$ /$0FEC_H$)。
2. 硬件监视定时器在复位后自动开始运行，并且不能停止。请以指定间隔定期清零计数器以防止监视定时器复位的发生。
3. 复位清零该定时器并在复位后恢复运行。

注：清零用作计数时钟（时基定时器或计时预分频器）的定时器时，监视定时器也清零。因此，如果设置软件，使所选定时器在监视定时器的间隔时间内重复清零，监视定时器的功能失效。

4.2 软件监视定时器

设置步骤示例

按照以下步骤设置软件监视定时器:

1. 确认地址 $FFBE_H$、$FFBF_H$ 的值是“$A596_H$”。（NVR 闪存区）
2. 选择计数时钟。(WDTC:CS1,CS0,CSP)
3. 启动监视定时器。(WDTC:WTE3 ～ WTE0 = 0101_B)
4. 在最短间隔时间内清零监视定时器。(WDTC:WTE3 ～ WTE0 = 0101_B)

参见表 2。

下例显示如何设置监视定时器来使用软件模式操作。

```
/* initial watchdog timer */
void InitWDT (void)
{
   WDTC = 0x05;      // set count clock is 2^21/F_CH
                     // start WDT counter
}


/* main routine */
void main (void)
{
       InitWDT();
       ...

   WDTC |= 0x05;     // clear WDT counter
}
```

main.c

```
#define HWD_DISABLE
;-------------------------------------------------------------------------
; Hard Watchdog
;-------------------------------------------------------------------------
#ifdef        HWD_DISABLE
              .SECTION  WDT, CONST, LOCATE=H'FFBE
              .DATA.W   0xA596
#endif
```

startup.asm

参考 project “SWWDT”的附录样本代码。

需注意的是一旦启动监视定时器，只能在复位发生后才可停止其运行。

4.3 硬件监视定时器

通过以下步骤设置硬件监视定时器：

1. 通过在闪存地址 $FFBE_H$、$FFBF_H$ 上写“$A597_H$”（待机模式下停止 WDT）或其他值（在全模式下使能 WDT）启动监视定时器。勿写“$A596_H$”或“$A597_H$”，该值将被复制到监视定时器选择 ID 寄存器 WDTH/WDTL ($0FEB_H$ /$0FEC_H$)。
2. 复位后 HWWDT 自动开始运行。
3. 在最短间隔时间内清零监视定时器。(WDTC:WTE3 ～ WTE0 = 0101_B)

使用 HWWDT 时，计数时钟切换位 CS1、CS0、CSP 固定在“001”，间隔时间也固定为 $2^{16}/F_{CRL}$。

下例显示如何设置监视定时器来使用硬件模式操作。

```
/* Hardware watchdog timer starts automatically after a reset and */
/* cannot be stopped. The internal time is fixed to 2^16/FCRL       */

/* main routine */
void main (void)
{
   ...

   WDTC |= 0x05;     // clear WDT counter

...
}
```

```
;------------------------------------------------------------------------
; Hard Watchdog
;------------------------------------------------------------------------
; #define HWD_DISABLE
#ifdef        HWD_DISABLE
              .SECTION  WDT, CONST, LOCATE=H'FFBE
                     .DATA.W   0xA596
#endif
```

main.c

startup.asm

参考 project “HWWDT”的附录样本代码。

需注意的是复位后硬件监视定时器自动开始运行且不能停止。

5 监视定时器的使用注意事项

使用监视定时器时，需注意以下各项。

- 停止监视定时器

  一旦启动，只有复位才可停止监视定时器。

  如选择 HWWDT，复位后恢复运行。

- 选择计数时钟

  软件监视定时器

  监视定时器启动后，将监视控制位 (WDTC:WTE3～WTE0) 设置为"0101B"，才可重写计数时钟切换位 (WDTC:CS1,CS0,CSP)。位操作指令不可对计数时钟切换位写入值。此外，定时器启动后不应改变位设置。

  副时钟模式下，时基定时器因主时钟停止振荡而停止工作。

  为了在副时钟模式下运行监视定时器，必须事先选择计时预分频器作为计数时钟并将"WDTC: CS1、CS0、CSP"设置为"100_B"、"110_B"或"$XX1_B$"。

  硬件监视定时器

  计数时钟固定在 $2^{16}/F_{CRL}$。

- 清零监视定时器

  清零用作监视定时器（时基定时器、计时预分频器或副 CR 定时器）计数时钟的计数器，也就清零了监视定时器的计数器。

  监视定时器进入休眠模式、停止模式或计时模式后，监视定时器的计数器清零，除非选择了硬件启动且待机模式下硬件监视定时器运行。

- 编程注意事项

  创建在主循环中重复清零监视定时器的程序时，含中断处理时间在内的主循环处理时间应该设定为最短监视定时器间隔时间或更短。

- 硬件监视（待机模式下定时器运行）

  监视定时器在停止模式、休眠模式、时基定时器模式或计时模式下不停止运行。因此，即使内部时钟停止，CPU 也不清零监视定时器。（在停止模式、休眠模式、时基定时器模式或计时模式下）。

  要定期让器件退出待机模式并清零监视定时器。但是在副时钟模式或副 CR 模式下，因振荡稳定等待时间设定寄存器的设置原因，CPU 退出停止模式后，有可能发生监视器复位。

  选择副时钟时，需特别注意副时钟稳定等待时间的设定。

6 更多信息

如欲了解有关 MB95200 产品的更多信息，敬请访问以下网址：

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

6.1 样本代码

6.1.1 工程名: SWWDT

软件监视定时器

main.c

```
#include "mb95200.h"
/*-----------------------------------------------------------------------
  name: Delay();
  function: delay function
-----------------------------------------------------------------------*/
void Delay (unsigned int i)
{
        while(i--)
        {
                asm("\tNOP");
        }
}
/*-----------------------------------------------------------------------
  name: InitWDT();
  function: initial watchdog timer
-----------------------------------------------------------------------*/
void InitCompTimer (void)
{
WDTC = 0x05;            // set count clock is 2^21/FCH
                                // start WDT counter
}
/*-----------------------------------------------------------------------
  name: main();
  function: main loop
-----------------------------------------------------------------------*/
void main(void)
{
        PDR0_P05 = 0;           // initial value
        DDR0_P05 = 1;           // set P05 as output

InitWDT ();

while(1)
```

```
{
        PDR0_P05 = ~PDR0_P05;          // show program is normal run
        Delay (500);

WDTC |= 0x05;    // clear WDT timer within a certain amount of time

}
}
```

startup.asm

```
;---------------------------------------------------------------------------
; variable define declaration
;---------------------------------------------------------------------------
#define HWD_DISABLE                    ; if define this, Hard Watchdog will disable.
;---------------------------------------------------------------------------
; Hard Watchdog
;---------------------------------------------------------------------------
#ifdef     HWD_DISABLE
                .SECTION  WDT, CONST, LOCATE=H'FFBE
                .DATA.W   0xA596
#endif
```

6.1.2 工程名: HWWDT

硬件监视

main.c

```
#include "mb95200.h"
/* Hardware watchdog timer starts automatically after a reset and cannot be stopped */
/* The internal time is fixed to 2^16/F_CRL                                          */
/*---------------------------------------------------------------------
  name: Delay();
  function: delay function
---------------------------------------------------------------------*/
void Delay (unsigned int i)
{
        while(i--)
        {
                asm("\tNOP");
        }
}
/*---------------------------------------------------------------------
  name: main();
  function: main loop
---------------------------------------------------------------------*/
void main(void)
{
        PDR0_P05 = 0;           // initial value
        DDR0_P05 = 1;           // set P05 as output

while(1)
{
        PDR0_P05 = ~PDR0_P05;           // show program is normal run
        Delay (500);

WDTC |= 0x05;    // clear WDT timer within a certain amount of time
}
}
```

startup.asm

```
;-------------------------------------------------------------------------
```

```
; variable define declaration
;-------------------------------------------------------------------------
;#define HWD_DISABLE                    ; if define this, Hard Watchdog will disable.
;-------------------------------------------------------------------------
; Hard Watchdog
;-------------------------------------------------------------------------
#ifdef     HWD_DISABLE
                    .SECTION  WDT, CONST, LOCATE=H'FFBE
                    .DATA.W   0xA596
#endif
```

文档修改记录

文档标题：AN205336 - F²MC-8FX 家族 MB95200H/210H 系列监视定时器

文档编号：002-05751

修订版	ECN	变更者	提交日期	变更说明
**	—	HUAL	03/20/2008	初稿
			07/18/2008	修改图号和样本编码
*A	5352935	HUAL	07/15/2016	已将 Spansion 应用手册《MCU-AN-500013-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。