# 第 58 章 数据 EEPROM

## 特点

本章包括下列主题：


58.1 简介 ..... 58-2
58.2 控制寄存器 ..... 58-2
58.3 数据 EEPROM 操作 ..... 58-8
58.4 数据 EEPROM 中断 ..... 58-16
58.5 数据 EEPROM 解锁序列 ..... 58-17
58.6 数据 EEPROM 读访问控制 ..... 58-17
58.7 节能和调试模式下的操作 ..... 58-18
58.8 各种复位的影响 ..... 58-18
58.9 相关应用笔记 ..... 58-19
58.10 版本历史 ..... 58-20

**注：** 本系列参考手册章节旨在用作对器件数据手册的补充。根据不同的器件型号，本手册章节可能并不适用于所有 PIC32 器件。

请参见当前器件数据手册中 **“数据 EEPROM”** 章节开头部分的注，以检查本文档是否支持您所使用的器件。

器件数据手册和系列参考手册章节可从 Microchip 网站下载：http://www.microchip.com。

## 58.1 简介

本系列参考手册章节介绍 PIC32 系列器件中的数据 EEPROM 模块。

数据 EEPROM 存储器通过特殊功能寄存器（SFR）接口进行访问。32 位 EEPROM 存储器的大小可能因器件而异。

## 58.2 控制寄存器

数据 EEPROM 读写操作使用以下特殊功能寄存器 （SFR）进行控制：

- **EECON：EEPROM 控制寄存器**

  它是所有数据 EEPROM 访问的主寄存器。该寄存器同时提供命令和状态功能。

- **EEKEY：EEPROM 密钥寄存器**

  该模块会监视该寄存器的特定写入值。它不是一个物理寄存器，读取时将返回零。

- **EEADDR：EEPROM 地址寄存器**

  每个数据 EEPROM 命令的地址通过写入 EEADDR 寄存器来进行选择。

- **EEDATA：EEPROM 数据寄存器**

  在读周期之后，该寄存器将包含从数据 EEPROM 中读取的数据。在数据 EEPROM 写周期之前，需要先将数据写入 EEDATA 寄存器。


© 2015 Microchip Technology Inc.

表 58-1 简要汇总了与数据 EEPROM 模块相关的寄存器。该汇总表之后列出了相应的寄存器，并且每个位均附有详细的说明。

**表 58-1： 数据 EEPROM SFR 汇总**

| 名称 | | Bit 31/15 | Bit 30/14 | Bit 29/13 | Bit 28/12 | Bit 27/11 | Bit 26/10 | Bit 25/9 | Bit 24/8 | Bit 23/7 | Bit 22/6 | Bit 21/5 | Bit 20/4 | Bit 19/3 | Bit 18/2 | Bit 17/1 | Bit 16/0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| EECON(1) | 31:16 | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| | 15:0 | ON | RDY | SIDL | ABORT | — | — | — | — | RW | WREN | ERR<1:0> | | ILW | CMD<2:0> | | |
| EEKEY(2) | 31:16 | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| | 15:0 | EEKEY<15:0> | | | | | | | | | | | | | | | |
| EEADDR(1,3) | 31:16 | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| | 15:0 | — | — | — | — | EEADDR<11:0> | | | | | | | | | | | |
| EEDATA | 31:16 | EEDATA<31:16> | | | | | | | | | | | | | | | |
| | 15:0 | EEDATA<15:0> | | | | | | | | | | | | | | | |

**图注：** — = 未实现，读为 0。

**注 1：** 该寄存器具有关联的清零、置 1 和取反寄存器，分别位于 0x4、0x8 和 0xC 字节偏移处。这些寄存器的命名方式是在关联寄存器的名称末尾附加 CLR、SET 或 INV（如 EECONCLR）。向这些寄存器的任意位写入 1 时，会将关联寄存器中的有效位清零、置 1 或取反。对这些寄存器的读操作将被忽略。

**2：** 该寄存器是一个只写寄存器。读取结果总是为 0。

**3：** 由于 EEPROM 字大小为 32 位，所以对于读操作和写操作，最后两位（EEADDR<1:0>）必须总是为 0。

© 2015 Microchip Technology Inc.

**寄存器 58-1： EECON：EEPROM 控制寄存器**

| 位范围 | Bit 31/23/15/7 | Bit 30/22/14/6 | Bit 29/21/13/5 | Bit 28/20/12/4 | Bit 27/19/11/3 | Bit 26/18/10/2 | Bit 25/17/9/1 | Bit 24/16/8/0 |
|---|---|---|---|---|---|---|---|---|
| 31:24 | U-0 | U-0 | U-0 | U-0 | U-0 | U-0 | U-0 | U-0 |
| | — | — | — | — | — | — | — | — |
| 23:16 | U-0 | U-0 | U-0 | U-0 | U-0 | U-0 | U-0 | U-0 |
| | — | — | — | — | — | — | — | — |
| 15:8 | R/W-0, HC | R-0 | R/W-0 | R/W-0, HC | U-0 | U-0 | U-0 | U-0 |
| | ON | RDY | SIDL | ABORT | — | — | — | — |
| 7:0 | R/W-0, HC | R/W-0 | R/W-0, HS, HC | R/W-0, HS, HC | R/W-0, HS | R/W-0 | R/W-0 | R/W-0 |
| | RW | WREN(1) | ERR<1:0> | | ILW | CMD<2:0>(1) | | |

| 图注： | HS = 可由硬件置 1 的位 | HC = 可由硬件清零的位 |
|---|---|---|
| R = 可读位 | W = 可写位 | U = 未实现位，读为 0 |
| -n = POR 值 | 1 = 置 1 | 0 = 清零 |

bit 31-16 **未实现：**读为 0

bit 15 **ON：**数据 EEPROM 电源控制位

1 = 使能数据 EEPROM
0 = 禁止数据 EEPROM

如果 RW 位置 1，尝试清零该位不会产生任何作用。此外，如果 DEVCFG 寄存器中的 FSLEEP 位置 1，在休眠期间不会清零该位。

bit 14 **RDY：**数据 EEPROM 就绪位

1 = 数据 EEPROM 已准备好，可供访问
0 = 数据 EEPROM 未准备好，不可访问

RDY 由硬件在每次发生 POR 或 BOR 事件时清零。它由硬件在 ON 位 = 1 且上电延时定时器延时结束时置 1。

bit 13 **SIDL：**空闲模式停止位

1 = 当 CPU 进入空闲模式时，模块停止工作
0 = 在空闲模式下，模块继续工作

bit 12 **ABORT：**数据 EEPROM 中止操作控制位

1 = 由软件置 1，用以尽快中止正在进行的写命令
0 = 数据 EEPROM 面板已就绪 / 正常工作

bit 11-8 **未实现：**读为 0

bit 7 **RW：**启动命令执行控制位

当 WREN = 1 时：

1 = 启动存储器字编程或擦除命令
0 = 由硬件清零，用以指示编程或擦除操作已完成

当 WREN = 0 时：

1 = 启动存储器字读命令
0 = 由硬件清零，用以指示读操作已完成

如果 ON 位 = 0，或如果 ON 位 = 1 而上电延时定时器延时尚未结束，则不能将该位置 1。BOR 复位会通过强制终止任何正在执行的命令并接着清零 RW 来间接清零该位。

bit 6 **WREN：**数据 EEPROM 写使能控制位 (1)

1 = 使能编程或擦除操作
0 = 禁止编程或擦除存储单元，并使能读操作

**注 1：** RW 位 = 1 时，不能修改该位（或这些位）。


© 2015 Microchip Technology Inc.

**寄存器 58-1： EECON：EEPROM 控制寄存器 （续）**

bit 5-4 **ERR<1:0>**：数据 EEPROM 序列错误状态位

11 = 发生了 BOR 事件
10 = 尝试使用无效写命令或带有未对齐地址 （EEADDR<1:0> ≠ 00）的命令执行读或写操作
01 = 发生了批量或页擦除或字编程校验错误
00 = 无错误条件

这些位可以用软件清零，或由于成功执行下一个操作而清零，或在 ON 位 = 0 时清零。这些位也可以用软件置 1 （RW 位 = 0 时），这不会影响模块的操作。

bit 3 **ILW**：数据 EEPROM 即将长写状态位

1 = 对 EEPROM 地址 （存放在 EEADDR 寄存器中）的下一个写操作将需要比通常情况更多的时间 （约 20 ms）
0 = 对 EEPROM 地址 （存放在 EEADDR 寄存器中）的下一个写操作将为正常写周期

该位可以用软件清零，或由于写入 EEADDR 寄存器而清零。该位由硬件在写命令后置 1。

bit 2-0 **CMD<2:0>**：数据 EEPROM 命令选择位 **(1)**

111 = 保留
•
•
•
100 = 配置寄存器写命令 （WREN 位必须置 1）
011 = 数据 EEPROM 存储器批量擦除命令 （WREN 位必须置 1）
010 = 数据 EEPROM 存储器页擦除命令 （WREN 位必须置 1）
001 = 字写命令 （WREN 位必须置 1）
000 = 字读命令 （WREN 位必须清零）

**注 1：** RW 位 = 1 时，不能修改该位 （或这些位）。

**寄存器 58-2： EEKEY：EEPROM 密钥寄存器**

| 位范围 | Bit 31/23/15/7 | Bit 30/22/14/6 | Bit 29/21/13/5 | Bit 28/20/12/4 | Bit 27/19/11/3 | Bit 26/18/10/2 | Bit 25/17/9/1 | Bit 24/16/8/0 |
|---|---|---|---|---|---|---|---|---|
| 31:24 | U-0 | U-0 | U-0 | U-0 | U-0 | U-0 | U-0 | U-0 |
| | — | — | — | — | — | — | — | — |
| 23:16 | U-0 | U-0 | U-0 | U-0 | U-0 | U-0 | U-0 | U-0 |
| | — | — | — | — | — | — | — | — |
| 15:8 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 |
| | EEKEY<15:8> | | | | | | | |
| 7:0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 |
| | EEKEY<7:0> | | | | | | | |

| 图注： | | |
|---|---|---|
| R = 可读位 | W = 可写位 | U = 未实现位，读为 0 |
| -n = POR 值 | 1 = 置 1 | 0 = 清零 |

bit 31-16 **未实现：**读为 0

bit 15-0 **EEKEY<15:0>：**数据 EEPROM 密钥位

向该寄存器依次写入值 0xEDB7 和 0x1248 将解锁 EECON 寄存器，从而可以对它进行写操作或擦除操作。读取该寄存器没有任何作用，将会返回 0。

写入任何其他值会锁定 EECON 寄存器。

**寄存器 58-3： EEADDR：EEPROM 地址寄存器**

| 位范围 | Bit 31/23/15/7 | Bit 30/22/14/6 | Bit 29/21/13/5 | Bit 28/20/12/4 | Bit 27/19/11/3 | Bit 26/18/10/2 | Bit 25/17/9/1 | Bit 24/16/8/0 |
|---|---|---|---|---|---|---|---|---|
| 31:24 | U-0 | U-0 | U-0 | U-0 | U-0 | U-0 | U-0 | U-0 |
| | — | — | — | — | — | — | — | — |
| 23:16 | U-0 | U-0 | U-0 | U-0 | U-0 | U-0 | U-0 | U-0 |
| | — | — | — | — | — | — | — | — |
| 15:8 | U-0 | U-0 | U-0 | U-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
| | — | — | — | — | EEADDR<11:8>(1) | | | |
| 7:0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
| | EEADDR<7:0>(1) | | | | | | | |

| 图注： | | |
|---|---|---|
| R = 可读位 | W = 可写位 | U = 未实现位，读为 0 |
| -n = POR 值 | 1 = 置 1 | 0 = 清零 |

bit 31-12 **未实现：**读为 0

bit 11-0 **EEADR<11:0>：**数据 EEPROM 地址位 (1)

该寄存器存放 EEPROM 存储器中要执行操作的地址。 RW 位 （EECON<7>）置 1 时， EEADDR<1:0> 必须总是为 00，否则会发生错误。

**注 1：** RW 位 （EECON<7>） = 1 时，不能修改该寄存器中的位。


© 2015 Microchip Technology Inc.

**寄存器 58-4： EEDATA：EEPROM 数据寄存器**

| 位范围 | Bit 31/23/15/7 | Bit 30/22/14/6 | Bit 29/21/13/5 | Bit 28/20/12/4 | Bit 27/19/11/3 | Bit 26/18/10/2 | Bit 25/17/9/1 | Bit 24/16/8/0 |
|---|---|---|---|---|---|---|---|---|
| 31:24 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
| | EEDATA<31:24>(1) | | | | | | | |
| 23:16 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
| | EEDATA<23:16>(1) | | | | | | | |
| 15:8 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
| | EEDATA<15:8>(1) | | | | | | | |
| 7:0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
| | EEDATA<7:0>(1) | | | | | | | |

| 图注： | | |
|---|---|---|
| R = 可读位 | W = 可写位 | U = 未实现位，读为 0 |
| -n = POR 值 | 1 = 置 1 | 0 = 清零 |

bit 31-0 **EEDATA<31:0>**：数据 EEPROM 数据位 (1)

该寄存器存放要在写操作期间存储到 EEPROM 存储器中的数据，或在读操作之后来自存储器的数据。

**注 1**： RW 位 （EECON<7>） = 1 时，不能修改这些位。此外，在 RW 位 = 1 时读取该寄存器可能不会返回有效数据，因为读操作可能尚未完成。

## 58.3 数据 EEPROM 操作

在器件上电时，ON 位（EECON<15>）会清零。电源从数据 EEPROM 断开，数据 EEPROM 处于最低功耗模式。

当ON位置1时，会为数据EEPROM加电。在上电延时定时器的延时结束后，RDY位（EECON<14>）由硬件置 1，数据 EEPROM 已准备好，可供访问。

在就绪可供访问后 （RDY = 1），所有读写操作均使用数据 EEPROM 寄存器执行。在所有情况下，仅当 RDY = 1 且没有正在进行的命令（RW 位（EECON<7>）= 0）时，数据 EEPROM 才准备好供读取或写入数据。

通过相关的外设模块禁止（PMD）位或通过清零 ON 位，可以禁止对数据 EEPROM 的访问。要确定哪个 PMD 寄存器和位控制 EEPROM 的电源，请参见具体器件数据手册中的 **“节能特性”** 章节。

### 58.3.1 数据 EEPROM 初始化

#### 58.3.1.1 POR 事件后的数据 EEPROM 初始化

在发生 POR 事件之后，ON 位清零，数据 EEPROM 掉电。要使能数据 EEPROM，需要设置 ON 位 = 1，并等待 RDY 位由硬件置 1。

在使能数据 EEPROM 控制器之后，在全速访问 EEPROM 之前，需要先将配置值写入数据 EEPROM。这通过配置写命令 （CMD<2:0> 位 （EECON<2:0>） = 100）来完成。要写入数据 EEPROM 控制器的配置值存储在 DEVEE1 至 DEVEE8 寄存器中。

写入配置值时， WREN 位（EECON<6>）必须置 1。因此，必须使用解锁序列。例 58-1 给出了一个在对数据 EEPROM 上电之后解锁数据 EEPROM 并写入配置值的代码示例。

#### 58.3.1.2 BOR 事件后的数据 EEPROM 初始化

在发生 BOR 事件之后，如果数据 EEPROM 在发生 BOR 事件之前处于上电状态，则它会保持上电状态 （ON 位 = 1）。 ERR<1:0> 位 （EECON<5:4>）将包含 11，这指示发生了 BOR 事件。RDY 位会清零。在访问数据 EEPROM 之前，需要等待 RDY 位由硬件置 1。

#### 58.3.1.3 非 POR、非 BOR 事件后的数据 EEPROM 初始化

在发生非 POR 或非 BOR 事件之后，如果数据 EEPROM 在发生事件之前处于上电状态，则它会保持上电状态（ON 位 = 1）。如果 RDY 位先前已置 1，则它会保持置 1。对于非 POR 或非 BOR 事件，EEPROM 将会完成它的最后一个命令。用户软件只需要等待任何正在进行的命令完成，然后再继续。


© 2015 Microchip Technology Inc.

**例 58-1： 数据 EEPROM 初始化代码**

```
void EEWriteExecute(int waitForDone)
{
    /* Disable interrupts prior to unlock sequence */
    EEKEY = 0xEDB7;            // Unlock the EEPROM to enable writing
    EEKEY = 0x1248;

    EECONbits.RW = 1;           // Execute the write

    /* Re-enable interrupts */

    if (waitForDone)            // Wait for the write to finish if
        while (EECONbits.RW);   // desired
}

void EEInitialize(void)         // Basic EEPROM enable and initialization
{
    EECONbits.ON = 1;// Turn on the EEPROM

    while (EECONbits.RDY == 0);// Wait until EEPROM is ready (~125 us)

    EECONbits.WREN = 1;// Enable writing to the EEPROM

    EECONbits.CMD = 0b100;// Set the command to Configuration Write

    EEADDR = 0x00;// Addr 0x00 = DEVEE1;
    EEDATA = DEVEE1;
    EEWriteExecute(1);          // Execute write and wait for finish

    EEADDR = 0x04;// Addr 0x04 = DEVEE2;
    EEDATA = DEVEE2;
    EEWriteExecute(1);          // Execute write and wait for finish

    EEADDR = 0x08;// Addr 0x08 = DEVEE3;
    EEDATA = DEVEE3;
    EEWriteExecute(1);          // Execute write and wait for finish

    EEADDR = 0x0C;// Addr 0x0C = DEVEE4;
    EEDATA = DEVEE4;
    EEWriteExecute(1);          // Execute write and wait for finish

    EEADDR = 0x10;// Addr 0x10 = DEVEE5;
    EEDATA = DEVEE5;
    EEWriteExecute(1);          // Execute write and wait for finish

    EEADDR = 0x14;// Addr 0x14 = DEVEE6;
    EEDATA = DEVEE6;
    EEWriteExecute(1);          // Execute write and wait for finish

    EEADDR = 0x18;// Addr 0x18 = DEVEE7;
    EEDATA = DEVEE7;
    EEWriteExecute(1);          // Execute write and wait for finish

    EEADDR = 0x1C;// Addr 0x1C = DEVEE8;
    EEDATA = DEVEE8;
    EEWriteExecute(1);          // Execute write and wait for finish

    EECONbits.WREN = 0;         // Turn off writes.
}
```

### 58.3.2 数据 EEPROM 读命令

在读取数据 EEPROM 之前，必须先使能数据 EEPROM，并且 RDY 状态位 （EECON<14>）必须由硬件置1。对数据EEPROM发出新的命令时，不能有正在进行的命令（RW位（EECON<7>）= 0）。

通过执行以下操作来执行数据 EEPROM 读命令：

1. 装入要读取的数据 EEPROM 地址 （即， EEADDR = 0x100）。地址必须处于 32 位边界上，所以最后两位 （EEADDR<1:0>）必须为 00。
2. 将读命令装入 CMD<2:0> 位 （即， EECONbits.CMD = 0x0）。
3. 通过清零 WREN 位来使能读访问 （即， EECONbits.WREN = 0）。
4. 通过将 RW 位置 1 来开始读操作 （即， EECONbits.RW = 1）。
5. 等待读周期完成且硬件清零 RW 位 （即， while (EECONbits.RW = 1)）。
6. 从 EEDATA 寄存器中读取数据 （即，变量 = EEDATA）。
7. 如果发生错误， ERR<1:0> 将包含错误代码。

**例 58-2： 数据 EEPROM 读命令代码**

```
#define EEPROM_NOT_READY 8
/*
 * Function returns ERR<1:0> bits or EEPROM_NOT_READY.
 * Requested data is stored in ee_data
 * Returns zero = no error occurred during the read cycle
 * non-zero = error occurred during the read cycle
 * or Data EEPROM is not ready
 */
int data_EEPROM_read(unsigned int ee_addr, int *ee_data)
{
    if (EECONbits.RDY==1)                // If Data EEPROM to be ready
    {
        if (EECONbits.RW==0)             // If no operation underway
        {                                // Execute Read Command
            EEADDR = ee_addr & 0xFFC;    // Set address on 32-bit boundary
            EECONbits.CMD = 0;           // Load CMD<2:0> with
                                         // Data EEPROM read command
            EECONbits.WREN = 0;          // Access for read
            EECONbits.RW = 1;            // Start the operation

            while (EECONbits.RW==1);     // Wait until read is complete
            *ee_data = EEDATA;           // Read the data

            return EECONbits.ERR;
        }
    }
    return EEPROM_NOT_READY;
}
```

在通过将 RW 位置 1 来执行读命令时，软件可以清零 RW 位。这将中止读命令。

- 在中止读命令后， EEDATA 寄存器中的数据可能是无效的
- 中止读命令不会报告任何错误
- 中止读命令不会产生任何读操作完成中断

在 RW 位置 1 时，对 EECON （ABORT 除外）和 EEADDR 寄存器的写操作会被忽略。

读命令唯一可以返回的错误是“无效命令”（ERR<1:0> = 10）；如果 EEADDR 不是处于 32 位边界上，则会发生该错误。


© 2015 Microchip Technology Inc.

### 58.3.3 数据 EEPROM 写命令

在写入数据 EEPROM 之前，必须先使能数据 EEPROM，并且 RDY 状态位 （EECON<14>）必须由硬件置1。对数据EEPROM发出新的命令时，不能有正在进行的命令（RW位（EECON<7>）= 0）。

通过执行以下操作来执行数据 EEPROM 写命令：

1. 装入要写入的数据 EEPROM 地址 （即， EEADDR = 0x100）。
2. 通过将 WREN 位置 1 来使能写访问 （即， EECONbits.WREN = 1）。
3. 将编程命令装入 CMD<2:0> 位 （即， EECONbits.CMD = 0x01）。
4. 装入要写入 EEDATA 寄存器的数据 （即， EEDATA = 0x13572468）。
5. 使用 EEKEY 寄存器执行数据 EEPROM 解锁序列。
6. 通过将 RW 位置 1 来开始字写操作 （即， EECONbits.RW = 1）。
7. 等待读周期完成且硬件清零 RW （即， while (EECONbits.RW = 1)）。
8. 如果发生错误， ERR<1:0> 将包含错误代码。

**例 58-3： 数据 EEPROM 写操作代码**

```
#define EEPROM_NOT_READY 8
/*
 * Function returns ERR<1:0> bits
 * Returns zero = no error occurred during the write cycle
 * non-zero = error occurred during the write cycle
 * or Data EEPROM is not ready
 */
int data_EEPROM_write(unsigned int ee_addr, int ee_data)
{
    if (EECONbits.RDY==1)             // If Data EEPROM to be ready
    {
        if (EECONbits.RW==0)          // If no operation underway
        {                             // Execute Write Command
            EEADDR = ee_addr & 0xFFC; // Load address on a 32-bit boundary
            EECONbits.CMD = 1;        // Load CMD<2:0> with write command
            EECONbits.WREN = 1;       // Access for write
            EEDATA = ee_data;

            EEKEY = 0xED87;           // Write unlock sequence
            EEKEY = 0x1248;

            EECONbits.RW = 1;         // Start the write cycle

            while (EECONbits.RW==1);
            //Wait for write cycle to complete

            // Return ERR<1:0> bits. Zero if no error.
            return EECONbits.ERR;
        }
    }
    return EEPROM_NOT_READY;
}
```

不同于读命令，写命令不能用软件中止。在写周期期间，软件尝试写入 EECON 寄存器的任何操作都会被忽略。

**警告**

用户需要确保不要尝试同时写入 / 擦除用户程序闪存 （与 NVM 寄存器相关联）和数据 EEPROM。**未遵从这种限制可能导致其中一个目标的写操作失败，或者两个目标都失败。**

#### 58.3.3.1 字写周期时间差异

如果在执行时间关键型应用程序时向数据 EEPROM 中写入数据，用户应考虑数据 EEPROM 写命令时间的差异。

写入单个 EEPROM 地址时，前 16 个写操作各需要大约 20 μs。第 17 个写周期，以及之后每隔 16 个写周期，将需要较长的写周期，大约 20 ms。

完整的 20 μs 或 20 ms 周期都被视为是一个写周期。在该周期期间，与所有写周期一样，RW 位（EECON<7>）会保持高电平，不能尝试任何新的数据 EEPROM 命令，直到 RW 位由硬件清零为止。

#### 58.3.3.2 管理长写周期

在第 17 个写周期以及之后每隔 16 个写周期，在写入新的数据之前， EEPROM 逻辑需要一段较长的时间来擦除存储单元。该周期的时间显著长于绝大多数的写周期，称为 “长写周期”。

出现这种长写周期时，数据 EEPROM 逻辑会提供管理协助。如果某个写周期之后对同一 EEPROM 地址的下一个写操作需要长写周期，即将长写状态 （ILW）位 （EECON<3>）会在该写周期之后置 1。

用户可以选择忽略 ILW 位，对该特定 EEPROM 地址的下一个写周期将会出现这个 20 ms 的周期。

对于时间关键型应用程序，与在一个循环中等待写命令完成相比，使用数据 EEPROM 中断与相应的中断服务程序 （ISR）可能会有一定的优势。

或者，用户软件也可以在对数据 EEPROM 的每个字写操作后检查 ILW 位，并选择在对用户应用程序来说更为合宜的时间产生这个 20 ms 的周期。通过向同一地址重新写入相同数据 （或新数据），“长写周期”将在用户软件选择的时间发生，而不是在未来的某个随机时间点。

如例 58-4 所示，函数会向 EEPROM 写入数据。如果 ILW 位在第一个写操作之后置 1，则第二个写操作会启动长写周期。

**例 58-4： 加快长写周期的数据 EEPROM 建议过程**

```
/*
 * Function returns ERR bits
 * Called to write data.
 * The first write cycle is always a short write cycle.
 * If ILW is set after the first write,
 *    a second write is initiated to force the long write cycle.
 * Returns zero = no error occurred during the write cycle
 *        non-zero = error occurred during the write cycle
 *                   or Data EEPROM is not ready
 */
int data_EEPROM_write_managed_long_cycle(unsigned int ee_addr,
                                                    int ee_data)
{
    int ret_err;

    ret_err = data_EEPROM_write(ee_addr, ee_data); //short write cycle

    if ((EECONbits.ILW == 1) && (ret_err == 0))
    {
        // Rewrite the data to the same location if ILW==1
        // This forces the long write cycle.
        // It will take ~20 ms to complete
        ret_err = data_EEPROM_write(ee_addr, ee_data);
    }

    return ret_err;
}
```


© 2015 Microchip Technology Inc.

### 58.3.4 数据 EEPROM 强制字擦除

正常情况下，不需要尝试强制字擦除。数据 EEPROM 具有可自动管理所有读 / 擦除 / 写命令序列的内部逻辑。软件执行强制字擦除命令应用于响应写校验错误（写操作后 ERR<1:0> = 01）。

如果在写入数据 EEPROM 期间发生校验错误，用户可以尝试强制字擦除命令。

如例 58-5 所示，强制字擦除命令类似于写命令，只是 CMD<2:0> 位（EECON<2:0>）= 010，并且不需要向 EEDATA 写入任何数据。

**例 58-5： 数据 EEPROM 强制字擦除代码**

```
#define EEPROM_NOT_READY 8
/*
 * Function returns ERR<1:0> bits
 * Returns zero = no error occurred, word erased properly
 * non-zero = error occurred, word may have fixed bits
 * or Data EEPROM is not ready
 */
int data_EEPROM_forced_word_erase(unsigned int addr)
{
    if (EECONbits.RDY==1)             // If Data EEPROM to be ready
    {
        if (EECONbits.RW==0)          // If no operation underway
        {                             // Execute erase command
            EEADDR = ee_addr;
            EECONbits.CMD = 2;        // Load CMD<2:0> with erase command
            EECONbits.WREN = 1;       // Access for write or erase

            EEKEY = 0xED87;           // Write unlock sequence
            EEKEY = 0x1248;

            EECONbits.RW = 1;         // Start the erase cycle

            while (EECONbits.RW==1);
            //Wait for erase cycle to complete

            // Return ERR<1:0> bits. Zero if no error.
            return EECONbits.ERR;
        }
    }
    return EEPROM_NOT_READY;
}
```

不同于读命令，强制字擦除命令不能用软件中止。在擦除周期期间，软件尝试写入 EECON 寄存器的任何操作都会被忽略。

| 警告 |
| --- |
| 用户需要确保不要尝试同时写入 / 擦除用户程序闪存（与 NVM 寄存器相关联）和数据 EEPROM。**未遵从这种限制可能导致其中一个目标的写操作失败，或者两个目标都失败。** |

### 58.3.5 数据 EEPROM 批量擦除

如例 58-6 所示，批量擦除命令类似于写命令，只是 CMD<2:0> 位（EECON<2:0>）= 011，并且不需要向 EEDATA 写入任何数据。

**例 58-6： 数据 EEPROM 批量擦除代码**

```
#define EEPROM_NOT_READY 8
/*
 * Function returns ERR<1:0> bits
 * Returns zero = no error occurred, word erased properly
 * non-zero = error occurred, word may have fixed bits
 * or Data EEPROM is not ready
 */
int data_EEPROM_bulk_erase(void)
{
    if (EECONbits.RDY==1)              // If Data EEPROM to be ready
    {
        if (EECONbits.RW==0)           // If no operation underway
        {                              // Execute erase command
            EECONbits.CMD = 3;         // Load CMD<2:0> with erase command
            EECONbits.WREN = 1;        // Access for write or erase

            EEKEY = 0xED87;            // Write unlock sequence
            EEKEY = 0x1248;

            EECONbits.RW = 1;          // Start the erase cycle

            while (EECONbits.RW==1);
            //Wait for erase cycle to complete

            // Return ERR<1:0> bits. Zero if no error.
            return EECONbits.ERR;
        }
    }
    return EEPROM_NOT_READY;
}
```

不同于读命令，批量擦除命令不能用软件中止。在擦除周期期间，软件尝试写入 EECON 寄存器的任何操作都会被忽略。

**警告**

用户需要确保不要尝试同时写入 / 擦除用户程序闪存（与 NVM 寄存器相关联）和数据 EEPROM。**未遵从这种限制可能导致其中一个目标的写操作失败，或者两个目标都失败。**


© 2015 Microchip Technology Inc.

### 58.3.6 数据 EEPROM 错误处理

如果在数据 EEPROM 的操作过程中发生错误，可以通过清零 ERR<1:0> 位（EECON<5:4>）重新初始化数据 EEPROM 内部逻辑。

#### 58.3.6.1 BOR 错误 （ERR<1:0> = 11）

发生 BOR 事件时，会中止任何正在进行的命令。如果正在进行的是写命令或强制字擦除命令，则用户应校验最后一个写或擦除地址处的数据。

不会发出错误中断。 RDY 位 （EECON<14>）会被清零。

等待 RDY 位由硬件置 1，然后通过清零 ERR<1:0> 位清除错误。

更多信息，请参见**第 58.3.1.2 节 “BOR 事件后的数据 EEPROM 初始化”**。

#### 58.3.6.2 无效命令 （ERR<1:0> = 10）

如果用户尝试错误地启动命令或将 CMD<2:0> 位 （EECON<2:0>）设置为某个保留命令，数据 EEPROM 会返回 “无效命令”错误。

错误启动命令的示例包括：

- 在命令不是读命令 （CMD<2:0> ≠ 000）时启动数据 EEPROM 读命令 （WREN = 0， RW = 1）
- 在使能写操作 （WREN = 1）时启动数据 EEPROM 读命令 （CMD<2:0> = 000， RW = 1）
- 在地址未 32 位对齐 （EEADDR<1:0> ≠ 00）时启动数据 EEPROM 读或写命令
- 在命令不是写命令 （CMD<2:0> ≠ 001）时启动数据 EEPROM 写命令 （WREN = 1， RW = 1）
- 在禁止写操作 （WREN = 0）时启动数据 EEPROM 写命令 （CMD<2:0> = 001， RW = 1）

通过清零 ERR<1:0> 位可清除错误。

#### 58.3.6.3 校验错误 （ERR<1:0> = 01）

执行数据 EEPROM 写或擦除命令时，内部 EEPROM 逻辑会校验写入数据 EEPROM 的数据或验证单元是否已擦除。如果在写入或擦除数据时发生错误，会产生校验错误。

如果发生校验错误，软件可以尝试通过在发生错误的地址处执行强制字擦除命令 （CMD<2:0> = 010）来恢复数据 EEPROM 字存储单元。或者，软件也可以选择不同的数据 EEPROM 地址来存储数据。

## 58.4 数据 EEPROM 中断

要使能或禁止中断，以及设置所有中断的优先级，请参见《PIC32 系列参考手册》的**第 8 章 “中断”**（DS60001108）。

与数据 EEPROM 关联的中断有三个：

- 数据 EEPROM 错误中断 （EEERRIF）
- 数据 EEPROM 读操作完成中断 （EERDIF）
- 数据 EEPROM 写操作完成中断 （EEWRIF）

**表 58-2： EEPROM 中断条件和操作**

| 说明 | 条件 | EEPROM 中断 | 操作 |
|---|---|---|---|
| 读命令中止 | 在 WREN = 0，RW = 1 时，软件清零 RW。 | 无 | 下一个数据 EEPROM 命令。 |
| 读命令完成 | 软件启动读命令。在 RW 由硬件清零时，读周期完成。 | EERDIF | 从 EEDATA 中读取数据。 |
| 写命令完成 | 启动写命令。<br>在 RW 由硬件清零时，写周期完成。 | EEWRIF | 下一个数据 EEPROM 命令。 |
| 写命令发生错误 | ERR<1:0> = 01<br>软件启动写命令。<br>校验逻辑检测到写入问题。 | EEERRIF | 请参见**第 58.3.6.3 节 “校验错误（ERR<1:0> = 01）”**。 |
| 发生数据 EEPROM 命令错误 | ERR<1:0> = 10 | EEERRIF | 通过清零 ERR<1:0> （ERR<1:0> = 00）来复位数据 EEPROM 逻辑。如果错误是由写操作导致的，则校验最后写入的数据。 |
| 数据 EEPROM BOR 事件 | ERR<1:0> = 11 | 无 | 在数据 EEPROM 初始化时检查是否发生了 BOR 事件。如果最后一个命令是写命令，则校验最后写入的数据。 |
| POR 事件 | ON = 0 | 无 | 初始化数据 EEPROM。 |
| BOR 事件 | ON = 1， RDY = 0，<br>CMD<2:0> = 111 | 无 | 在访问数据EEPROM之前等待RDY = 1。 |

### 58.4.1 数据 EEPROM 错误中断 （ERRIF）

在执行读或写操作时，当操作完成时，如果 ERR<1:0> ≠ 00 指示发生了错误，则会触发数据 EEPROM 错误中断。

如果允许数据 EEPROM 错误中断，则其优先级应高于所有其他数据 EEPROM 中断。

### 58.4.2 数据 EEPROM 读操作完成 （EERDIF）和写操作完成 （EEWRIF）

当读操作完成且没有发生错误时 （ERR<1:0> == 00），会触发数据 EEPROM 读操作完成中断（EERDIF）。

当写操作完成且没有发生错误时 （ERR<1:0> == 00），会触发数据 EEPROM 写操作完成中断（EEWRIF）。


© 2015 Microchip Technology Inc.

## 58.5 数据 EEPROM 解锁序列

为了保护数据 EEPROM 中存储的数据，每次发出写命令、强制字擦除命令或批量擦除命令时，必须在将 RW 位 （EECON<7>）置 1 之前执行解锁序列 （见例 58-7）。

在 WREN 位 （EECON<6>）置 1 的情况下，只有在执行解锁序列之后才能将 RW 位置 1。

读命令不需要解锁序列。向 EEKEY 寄存器写入解锁序列时应禁止中断，以防止解锁序列被中断。

**例 58-7： 解锁数据 EEPROM 以进行写操作的代码**

```
// The Data EEPROM Unlock sequence must be executed prior to attempting a
// write or erase cycle

    EEKEY = 0xED87;         // Write unlock sequence
    EEKEY = 0x1248;
```

## 58.6 数据 EEPROM 读访问控制

在设置数据 EEPROM 时需要知道一些器件配置位。

### 58.6.1 CFGAPP3 配置寄存器设置 （EEWS<7:0>）

数据 EEPROM 读访问计数是在数据 EEPROM 存储器访问中使用的 SYSCLK 周期数。更多信息，请参见具体器件数据手册中的 **“数据 EEPROM”** 章节。

基于应用工作速度 （FSYS）来选择其值。对于 PIC32 器件系列，设置应为 1 至 20，具体取决于器件速度。有关基于系统时钟速度的建议值，请参见表 58-3。

**注：** 将该值设置为高于建议值会使器件读操作的速度下降。

**表 58-3： 数据 EEPROM 读访问配置 (3)**

| 最大系统时钟 （MHz） | EEWS<7:0>(1) 设置 | EEPROM 访问 （周期数 / 读操作） | 读访问 （周期数） (2) | 读访问 （ns） |
|---|---|---|---|---|
| < 200 | 00010011 | 20 | 109 | 545 |
| < 175 | 00010001 | 18 | 99 | 566 |
| < 150 | 00001110 | 15 | 84 | 560 |
| < 125 | 00001100 | 13 | 74 | 592 |
| < 100 | 00001001 | 10 | 59 | 590 |
| < 75 | 00000111 | 8 | 49 | 653 |
| < 50 | 00000100 | 5 | 34 | 680 |
| < 25 | 00000010 | 3 | 24 | 960 |

**注 1：** EEWS = EEPROM 访问时间 （ns） * 系统频率 （MHz）。

**2：** 由于有限状态机开销，读访问 （周期数） = [(EEWS + 2) * 5] + 4。

**3：** 该表作为示例提供，其 EEPROM 访问时间 = 100 ns。关于特定于器件的详细信息，请参见您所用器件的数据手册。

## 58.7 节能和调试模式下的操作

### 58.7.1 休眠模式下的操作

当器件进入休眠模式时， EEPROM 的状态取决于 DEVCFG 寄存器中的 FSLEEP 位的状态。如果 FSLEEP = 1，EEPROM 将在休眠模式下保持供电，EEPROM 操作可以在器件退出休眠模式时立即继续。

如果 FSLEEP = 0，则会断开 EEPROM 的电源， ON 位 （EECON<15>）会被清零。在继续 EEPROM 操作之前，必须先将 ON 位置 1，并且 RDY 位 （EECON<14>）必须由硬件置 1。

### 58.7.2 空闲模式下的操作

当操作正在进行时，空闲模式对 EEPROM 控制器没有任何作用。

### 58.7.3 调试模式下的操作

EEPROM 控制器未提供调试冻结功能，因此在编程操作正在进行时，它对于 EEPROM 控制器模块没有任何作用。CPU 会继续被占用，直到数据操作完成。该情形的唯一例外是 EEKEY 解锁序列，当处于调试模式时，它会被暂停，让用户可以单步执行解锁序列。

### 58.7.4 外设模块禁止 （PMD）

当与数据 EEPROM 相关联的外设模块禁止 （PMD）位置 1 时：

- 数据 EEPROM 会立即被禁止
- 任何正在进行的命令会立即被中止
- 模块时钟会被禁止；电源与模块断开

建议用户禁止数据 EEPROM 写中断 （如果已允许），并等待 RW 位 （EECON<7>）置 1，之后再将模块的 PMD 位置 1。这可以避免可能未完成的写周期。

**例 58-8： 数据 EEPROM 对应 PMD 位置 1 的建议过程**

```
void data_EEPROM_set_PMD( void )
{
     IEC5CLR = 1 << 26;       // Disable Data EEPROM write interrupt, if it
                              // was enabled
     while (EECONbits.RW);    // wait until last command is complete,
     PMD4bits.EEMD = 1;       // and then disable the module
}
```

## 58.8 各种复位的影响

### 58.8.1 器件复位

在器件复位时不会复位任何 EECON 位。只有 POR 复位才会复位所有其他 SFR 位；但是，器件复位会复位 EEKEY 的状态。

### 58.8.2 上电复位

在发生 POR 时，所有 EEPROM 控制器寄存器会被强制设为它们的复位状态。

### 58.8.3 看门狗和程序监控定时器复位

在发生看门狗或程序监控定时器复位时，所有 EEPROM 控制器寄存器不变。

### 58.8.4 58.8.4 出厂复位

为了确保数据 EEPROM 内容处于已知状态，应在第一次启动时 （即，在出厂设置和测试期间）对器件进行批量擦除。


© 2015 Microchip Technology Inc.

## 58.9 相关应用笔记

本节列出了与手册本章内容相关的应用笔记。这些应用笔记可能并不是专为 PIC32 器件系列而编写的，但其概念是相近的，通过适当修改并受到一定限制即可使用。当前与数据 EEPROM 模块相关的应用笔记有：

| 标题 | 应用笔记编号 |
|---|---|
| 目前没有相关的应用笔记。 | N/A |

**注：** 如需获取更多 PIC32 系列器件的应用笔记和代码示例，请访问 Microchip 网站（www.microchip.com）。

## 58.10 版本历史

### 版本 A（2015 年 4 月）

这是本文档的初始版本。

 © 2015 Microchip Technology Inc.

**请注意以下有关 Microchip 器件代码保护功能的要点：**

- Microchip 的产品均达到 Microchip 数据手册中所述的技术指标。
- Microchip 确信：在正常使用的情况下， Microchip 系列产品是当今市场上同类产品中最安全的产品之一。
- 目前，仍存在着恶意、甚至是非法破坏代码保护功能的行为。就我们所知，所有这些行为都不是以 Microchip 数据手册中规定的操作规范来使用 Microchip 产品的。这样做的人极可能侵犯了知识产权。
- Microchip 愿与那些注重代码完整性的客户合作。
- Microchip 或任何其他半导体厂商均无法保证其代码的安全性。代码保护并不意味着我们保证产品是 “牢不可破”的。

代码保护功能处于持续发展中。 Microchip 承诺将不断改进产品的代码保护功能。任何试图破坏 Microchip 代码保护功能的行为均可视为违反了《数字器件千年版权法案 （Digital Millennium Copyright Act）》。如果这种行为导致他人在未经授权的情况下，能访问您的软件或其他受版权保护的成果，您有权依据该法案提起诉讼，从而制止这种行为。

**提供本文档的中文版本仅为了便于理解。请勿忽视文档中包含的英文部分，因为其中提供了有关 Microchip 产品性能和使用情况的有用信息。Microchip Technology Inc. 及其分公司和相关公司、各级主管与员工及事务代理机构对译文中可能存在的任何差错不承担任何责任。建议参考 Microchip Technology Inc. 的英文原版文档。**

本出版物中所述的器件应用信息及其他类似内容仅为您提供便利，它们可能由更新之信息所替代。确保应用符合技术规范，是您自身应负的责任。**Microchip 对这些信息不作任何明示或暗示、书面或口头、法定或其他形式的声明或担保，包括但不限于针对其使用情况、质量、性能、适销性或特定用途的适用性的声明或担保。** Microchip 对因这些信息及使用这些信息而引起的后果不承担任何责任。如果将 Microchip 器件用于生命维持和 / 或生命安全应用，一切风险由买方自负。买方同意在由此引发任何一切伤害、索赔、诉讼或费用时，会维护和保障 Microchip 免于承担法律责任，并加以赔偿。除非另外声明，在 Microchip 知识产权保护下，不得暗中或以其他方式转让任何许可证。

**商标**

Microchip 的名称和徽标组合、 Microchip 徽标、 dsPIC、 FlashFlex、 flexPWR、 JukeBlox、 KEELOQ、 KEELOQ 徽标、 Kleer、 LANCheck、 MediaLB、 MOST、 MOST 徽标、 MPLAB、 OptoLyzer、 PIC、 PICSTART、 PIC32 徽标、 RightTouch、 SpyNIC、 SST、 SST 徽标、 SuperFlash 及 UNI/O 均为 Microchip Technology Inc. 在美国和其他国家或地区的注册商标。

The Embedded Control Solutions Company 和 mTouch 为 Microchip Technology Inc. 在美国的注册商标。

Analog-for-the-Digital Age、BodyCom、chipKIT、chipKIT 徽标、 CodeGuard、 dsPICDEM、 dsPICDEM.net、 ECAN、 In-Circuit Serial Programming、 ICSP、 Inter-Chip Connectivity、 KleerNet、 KleerNet 徽标、 MiWi、 MPASM、 MPF、 MPLAB Certified 徽标、 MPLIB、 MPLINK、 MultiTRAK、 NetDetach、 Omniscient Code Generation、 PICDEM、 PICDEM.net、 PICkit、 PICtail、 RightTouch 徽标、 REAL ICE、 SQI、 Serial Quad I/O、 Total Endurance、 TSHARC、 USBCheck、 VariSense、 ViewSpan、 WiperLock、 Wireless DNA 和 ZENA 均为 Microchip Technology Inc. 在美国和其他国家或地区的商标。

SQTP 为 Microchip Technology Inc. 在美国的服务标记。

Silicon Storage Technology 为 Microchip Technology Inc. 在除美国外的国家或地区的注册商标。

GestIC 为 Microchip Technology Inc. 的子公司 Microchip Technology Germany II GmbH & Co. & KG 在除美国外的国家或地区的注册商标。

在此提及的所有其他商标均为各持有公司所有。

© 2015, Microchip Technology Inc. 版权所有。

ISBN：978-1-63277-702-7

**QUALITY MANAGEMENT SYSTEM CERTIFIED BY DNV**

**═ ISO/TS 16949 ═**

*Microchip 位于美国亚利桑那州 Chandler 和 Tempe 与位于俄勒冈州 Gresham 的全球总部、设计和晶圆生产厂及位于美国加利福尼亚州和印度的设计中心均通过了 ISO/TS-16949:2009 认证。 Microchip 的 PIC® MCU 与 dsPIC® DSC、KEELOQ® 跳码器件、串行 EEPROM、单片机外设、非易失性存储器和模拟产品严格遵守公司的质量体系流程。此外， Microchip 在开发系统的设计和生产方面的质量体系也已通过了 ISO 9001:2000 认证。*

# 全球销售及服务网点

## 美洲

**公司总部 Corporate Office**
2355 West Chandler Blvd.
Chandler, AZ 85224-6199
Tel: 1-480-792-7200
Fax: 1-480-792-7277
技术支持：
http://www.microchip.com/support
网址：www.microchip.com

**亚特兰大 Atlanta**
Duluth, GA
Tel: 1-678-957-9614
Fax: 1-678-957-1455

**奥斯汀 Austin, TX**
Tel: 1-512-257-3370

**波士顿 Boston**
Westborough, MA
Tel: 1-774-760-0087
Fax: 1-774-760-0088

**芝加哥 Chicago**
Itasca, IL
Tel: 1-630-285-0071
Fax: 1-630-285-0075

**克里夫兰 Cleveland**
Independence, OH
Tel: 1-216-447-0464
Fax: 1-216-447-0643

**达拉斯 Dallas**
Addison, TX
Tel: 1-972-818-7423
Fax: 1-972-818-2924

**底特律 Detroit**
Novi, MI
Tel: 1-248-848-4000

**休斯敦 Houston, TX**
Tel: 1-281-894-5983

**印第安纳波利斯 Indianapolis**
Noblesville, IN
Tel: 1-317-773-8323
Fax: 1-317-773-5453

**洛杉矶 Los Angeles**
Mission Viejo, CA
Tel: 1-949-462-9523
Fax: 1-949-462-9608

**纽约 New York, NY**
Tel: 1-631-435-6000

**圣何塞 San Jose, CA**
Tel: 1-408-735-9110

**加拿大多伦多 Toronto**
Tel: 1-905-673-0699
Fax: 1-905-673-6509

## 亚太地区

**亚太总部 Asia Pacific Office**
Suites 3707-14, 37th Floor
Tower 6, The Gateway
Harbour City, Kowloon
Hong Kong
Tel: 852-2943-5100
Fax: 852-2401-3431

**中国 - 北京**
Tel: 86-10-8569-7000
Fax: 86-10-8528-2104

**中国 - 成都**
Tel: 86-28-8665-5511
Fax: 86-28-8665-7889

**中国 - 重庆**
Tel: 86-23-8980-9588
Fax: 86-23-8980-9500

**中国 - 东莞**
Tel: 86-769-8702-9880

**中国 - 杭州**
Tel: 86-571-8792-8115
Fax: 86-571-8792-8116

**中国 - 香港特别行政区**
Tel: 852-2943-5100
Fax: 852-2401-3431

**中国 - 南京**
Tel: 86-25-8473-2460
Fax: 86-25-8473-2470

**中国 - 青岛**
Tel: 86-532-8502-7355
Fax: 86-532-8502-7205

**中国 - 上海**
Tel: 86-21-5407-5533
Fax: 86-21-5407-5066

**中国 - 沈阳**
Tel: 86-24-2334-2829
Fax: 86-24-2334-2393

**中国 - 深圳**
Tel: 86-755-8864-2200
Fax: 86-755-8203-1760

**中国 - 武汉**
Tel: 86-27-5980-5300
Fax: 86-27-5980-5118

**中国 - 西安**
Tel: 86-29-8833-7252
Fax: 86-29-8833-7256

**中国 - 厦门**
Tel: 86-592-238-8138
Fax: 86-592-238-8130

**中国 - 珠海**
Tel: 86-756-321-0040
Fax: 86-756-321-0049

## 亚太地区

**台湾地区 - 高雄**
Tel: 886-7-213-7828

**台湾地区 - 台北**
Tel: 886-2-2508-8600
Fax: 886-2-2508-0102

**台湾地区 - 新竹**
Tel: 886-3-5778-366
Fax: 886-3-5770-955

**澳大利亚 Australia - Sydney**
Tel: 61-2-9868-6733
Fax: 61-2-9868-6755

**印度 India - Bangalore**
Tel: 91-80-3090-4444
Fax: 91-80-3090-4123

**印度 India - New Delhi**
Tel: 91-11-4160-8631
Fax: 91-11-4160-8632

**印度 India - Pune**
Tel: 91-20-3019-1500

**日本 Japan - Osaka**
Tel: 81-6-6152-7160
Fax: 81-6-6152-9310

**日本 Japan - Tokyo**
Tel: 81-3-6880-3770
Fax: 81-3-6880-3771

**韩国 Korea - Daegu**
Tel: 82-53-744-4301
Fax: 82-53-744-4302

**韩国 Korea - Seoul**
Tel: 82-2-554-7200
Fax: 82-2-558-5932 或
82-2-558-5934

**马来西亚 Malaysia - Kuala Lumpur**
Tel: 60-3-6201-9857
Fax: 60-3-6201-9859

**马来西亚 Malaysia - Penang**
Tel: 60-4-227-8870
Fax: 60-4-227-4068

**菲律宾 Philippines - Manila**
Tel: 63-2-634-9065
Fax: 63-2-634-9069

**新加坡 Singapore**
Tel: 65-6334-8870
Fax: 65-6334-8850

**泰国 Thailand - Bangkok**
Tel: 66-2-694-1351
Fax: 66-2-694-1350

## 欧洲

**奥地利 Austria - Wels**
Tel: 43-7242-2244-39
Fax: 43-7242-2244-393

**丹麦 Denmark-Copenhagen**
Tel: 45-4450-2828
Fax: 45-4485-2829

**法国 France - Paris**
Tel: 33-1-69-53-63-20
Fax: 33-1-69-30-90-79

**德国 Germany - Dusseldorf**
Tel: 49-2129-3766400

**德国 Germany - Karlsruhe**
Tel: 49-721-625370

**德国 Germany - Munich**
Tel: 49-89-627-144-0
Fax: 49-89-627-144-44

**意大利 Italy - Milan**
Tel: 39-0331-742611
Fax: 39-0331-466781

**意大利 Italy - Venice**
Tel: 39-049-7625286

**荷兰 Netherlands - Drunen**
Tel: 31-416-690399
Fax: 31-416-690340

**波兰 Poland - Warsaw**
Tel: 48-22-3325737

**西班牙 Spain - Madrid**
Tel: 34-91-708-08-90
Fax: 34-91-708-08-91

**瑞典 Sweden - Stockholm**
Tel: 46-8-5090-4654

**英国 UK - Wokingham**
Tel: 44-118-921-5800
Fax: 44-118-921-5820

07/14/15


初稿
© 2015 Microchip Technology Inc.