AN1982

tinyAVR® 0 和 1 系列及 megaAVR® 0 系列中的中断系统

特性

- 中断控制器概述
- 中断优先级配置
- 向量表配置
- 代码示例

简介

单片机（MCU）包含各种硬件模块（外设），旨在用于执行通信、时序和波形生成等专门任务。通常，每个外设都有许多信号用于指示其状态：操作完成、数据可用性或外设准备好接收新命令。中央处理单元（CPU）负责检查每个模块的相关更新。之后，CPU 必须处理每个模块的请求。随着外设模块越来越多，CPU 需要花费越来越多的时间来检查和处理所有模块。此 CPU 负荷会导致更长的响应时间和更高的功耗。传统上，MCU 使用两种主要方法来监视和处理外设：轮询和中断。

轮询意味着手动读取和检查由受监视外设更新的状态位。这为设计人员提供了很高的自由度，他们可以自由决定检查不同外设的频率和顺序。在应用程序仅等待特定状态的简单用例中，可以进行非常紧密的循环；检查此状态位并在出现特定状态时立即对其进行处理。尽管使用轮询方法可以带来非常快的响应速度，但仍然存在几个主要缺点。首先，响应能力会随着要检查的状态位数量的增加而降低。其次，CPU 在执行每个状态位的代码测试时需要以工作模式运行，而这会增加功耗。

随着应用程序中涉及的外设和状态位数的增加，中断系统可提供更好的响应能力。中断系统采用了一种外设可以发送请求来中断 CPU 执行的方案。由于外设会在需要处理时提示 CPU，因此 CPU 无需主动轮询状态位。但是，当某个外设可以随时向 CPU 请求服务时，它不会考虑其他外设的需求，这会带来中断请求堆积的风险。因此，需要使用中断控制器来确定 CPU 处理待处理中断的顺序。传统上，tinyAVR®和 megaAVR®器件的中断处理使用基于中断向量地址的预定义优先级。XMEGA® MCU 系列可以使用可编程多级中断控制器（Programmable Multilevel Interrupt Controller，PMIC）定制优先级队列，允许用户为中断分配三个优先级。在 tinyAVR 0 和 1 系列以及 megaAVR 0 系列中，中断处理技术具有更高的可配置性。本应用笔记详细介绍了新中断控制器的功能，它将预定义优先级和多级控制器组合使用。

目录

1. 相关器件

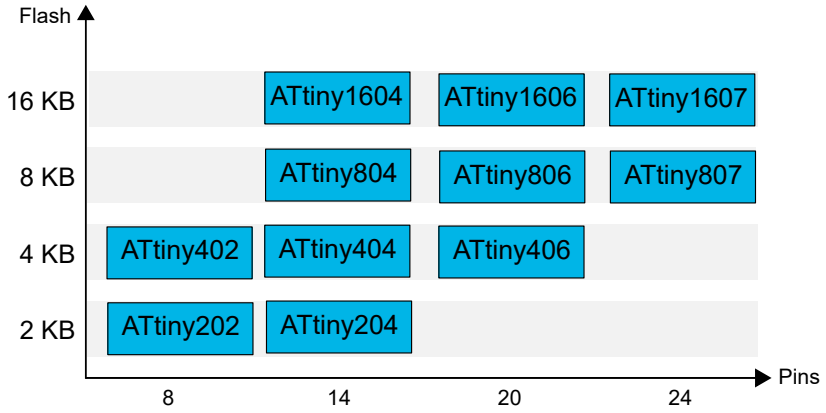
本章列出了文中涉及的相关器件。

1.1 tinyAVR® 0 系列

下图所示为 tinyAVR 0 系列器件，注明了不同的引脚数与存储器大小：

- 在垂直方向上，无需修改代码即可实现移植，因为这些器件的引脚和功能完全兼容。
- 水平向左移植会减少引脚数，进而减少可用的功能。

图 1-1. tinyAVR® 0 系列概览



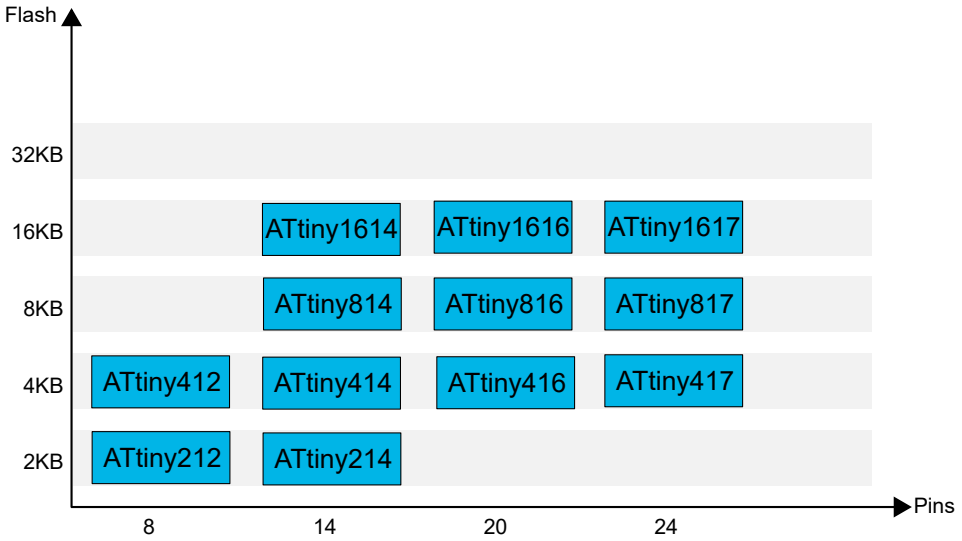
具有不同闪存大小的器件通常也具有不同的 SRAM 和 EEPROM。

1.2 tinyAVR® 1 系列

下图所示为 tinyAVR 1 系列器件，注明了不同的引脚数与存储器大小：

- 垂直向上移植无需修改代码，因为这些器件引脚兼容并可提供相同或更多的功能。而向下移植可能需要修改代码，因为某些外设的可用实例数减少。
- 水平向左移植会减少引脚数，进而减少可用的功能。

图 1-2. tinyAVR® 1 系列概览



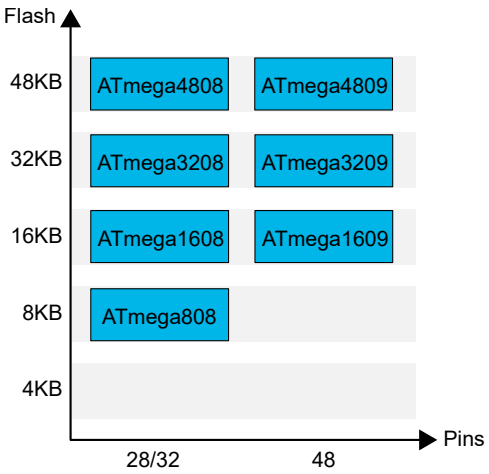
具有不同闪存大小的器件通常也具有不同的 SRAM 和 EEPROM。

1.3 megaAVR® 0 系列

下图所示为 megaAVR 0 系列器件，注明了不同的引脚数与存储器大小：

- 无需修改代码即可实现垂直移植，因为这些器件的引脚和功能完全兼容。
- 水平向左移植会减少引脚数，进而减少可用的功能。

图 1-3. megaAVR® 0 系列概览



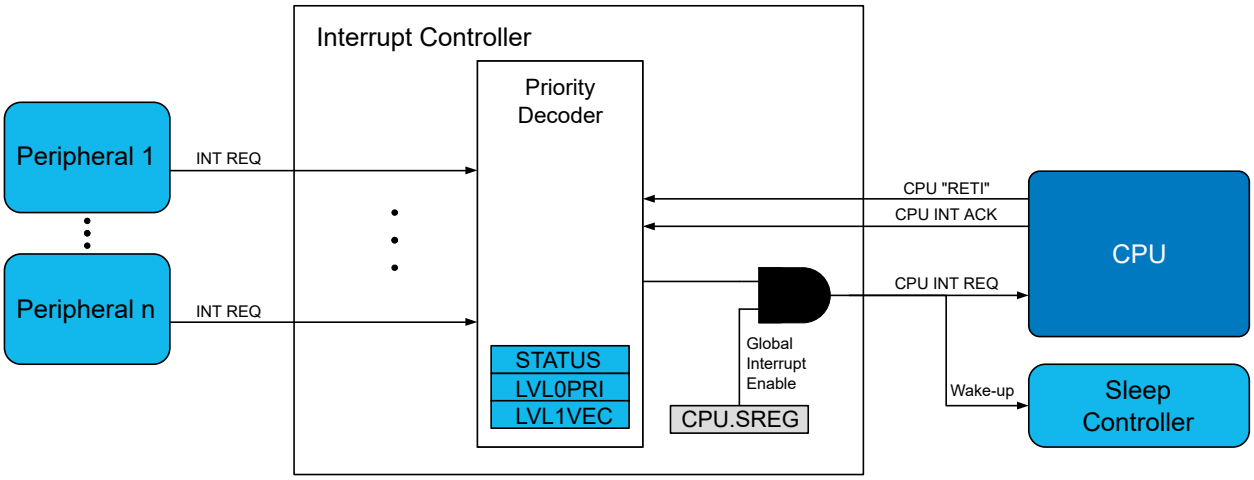
具有不同闪存大小的器件通常也具有不同的 SRAM 和 EEPROM。

2. 中断控制器概述

tinyAVR 0 和 1 系列以及 megaAVR 0 系列中断控制器支持多个独立的中断请求（Interrupt Request，IRQ），具有专用中断向量。这有助于缩短触发中断时的延时。

当允许中断且发生中断条件时，CPU 将从中断控制器接收 IRQ。根据中断的优先级和任何正在进行的中断服务程序（Interrupt Service Routine，ISR）的优先级，IRQ 可能得到响应（并且其 ISR 会执行），也可能保持待处理状态。有关中断控制器交互的图形表示，请参见图 2-1。

图 2-1. 中断控制器流程图



中断控制器通过中断优先级和向量地址的组合来决定待处理 ISR 的执行顺序。硬接线向量地址和不同优先级模式可通过 CPUINT 寄存器配置，具有以下优先级：

1. 不可屏蔽中断
2. 1 级优先级中断
3. 0 级优先级中断

有关中断向量映射，请参见器件数据手册。

中断标志

必须为要产生的 IRQ 设置全局中断允许标志（I 标志），不可屏蔽中断是惟一例外。相关信息，请参见 6. 不可屏蔽中断。

大多数中断标志都必须在相应的 ISR 中清零。这是因为某些中断源共用中断向量。进入 ISR 或读取数据寄存器时，其他标志由硬件清零。有关详细信息，请参见器件数据手册中每个外设的对应章节。

中断延时

CPU 的中断延时由以下操作所需的时钟周期数决定：

1. 完成正在进行的指令。
   - 1-3 个时钟周期，取决于指令时间。
2. 将程序计数器压入堆栈。
   - 2 个时钟周期。
3. 执行存储在中断向量表中的跳转指令。
   - 2-3 个时钟周期，取决于闪存大小。请参见表 2-1。

表 2-1. 中断跳转指令

闪存大小	跳转指令	跳转执行时间
≤ 8 kB	RJMP	2 个时钟周期
> 8 kB	JMP	3 个时钟周期

总中断延时将为 5 至 8 个时钟周期，具体取决于上面列出的参数。有关更多信息，请参见 AVR Instruction Set Manual。

如果器件处于休眠模式，除了所使用休眠模式下的 CPU 启动时间外，响应时间还会增加 5 个时钟周期。

3. 静态优先级中断

tinyAVR 0 和 1 系列以及 megaAVR 0 系列默认情况下使用具有静态优先级的中断向量表。这与其他经典的 tinyAVR®和 megaAVR® MCU 相同，对于大多数标准应用，这是首选的操作方法。

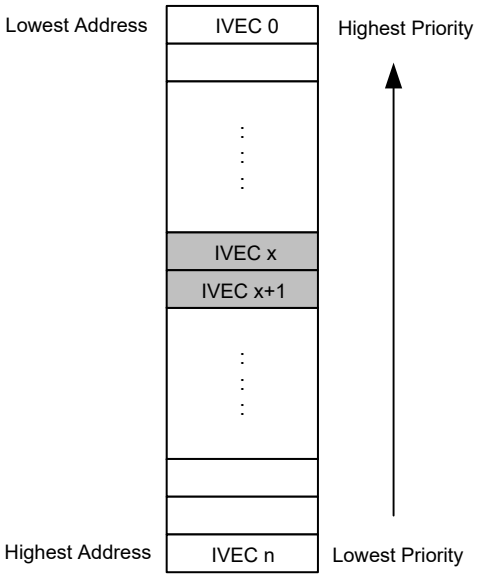
静态优先级中断方案仅适用于 0 级优先级中断。有关优先级中断的信息，请参见 5. 高优先级中断向量和 6. 不可屏蔽中断两章。

3.1 操作

使用静态优先级中断向量表时，最低中断向量地址具有最高优先级。这意味着，当两个或多个中断待处理时，执行顺序将由中断向量地址按升序设置。关于图形表示，请参见图 3-1。

当 CPU 正在执行 ISR 时，任何新的 IRQ 都将保持待处理状态，直到正在运行的中断处理程序完成，此时将首先执行具有最低向量地址（最高优先级）的中断。

图 3-1. 使用静态优先级方案的中断优先级



在静态优先级中断方案中，CPUINT.LVL0PRI 寄存器定义了最高优先级中断的起始点，因此可以通过将所需的中断向量地址写入 LVL0PRI 以静态方式改变向量表中的优先级。该寄存器的默认值为 0x00，因此复位后，最低地址中断向量的优先级最高。

3.2 示例

静态优先级中断方案是 tinyAVR 0 和 1 系列以及 megaAVR 0 系列器件中的中断控制器的默认设置，因此会在启动时为此配置 CPUINT 寄存器。以下代码段给出了有关如何配置静态优先级中断方案的示例。

静态优先级中断方案配置

```
//包含具有寄存器和向量定义的器件头文件 io.h
#include <avr/io.h>
//interrupt.h 包含与 ISR 相关的内容
#include <avr/interrupt.h>

void static_priority_interrupt_example(void) {
```

```
    //必要时，清零轮转调度使能位。
    CPUINT.CTRLA &= ~CPUINT_LVL0RR_bm;

    //有关如何改变中断优先级的示例。
    CPUINT.LVL0PRI = PORTA_PORT_vect_num;

    //允许全局中断
    sei();
}

ISR(PORTA_PORT_vect){
    //该中断将具有最高优先级。
}
```

4. 轮转优先级方案

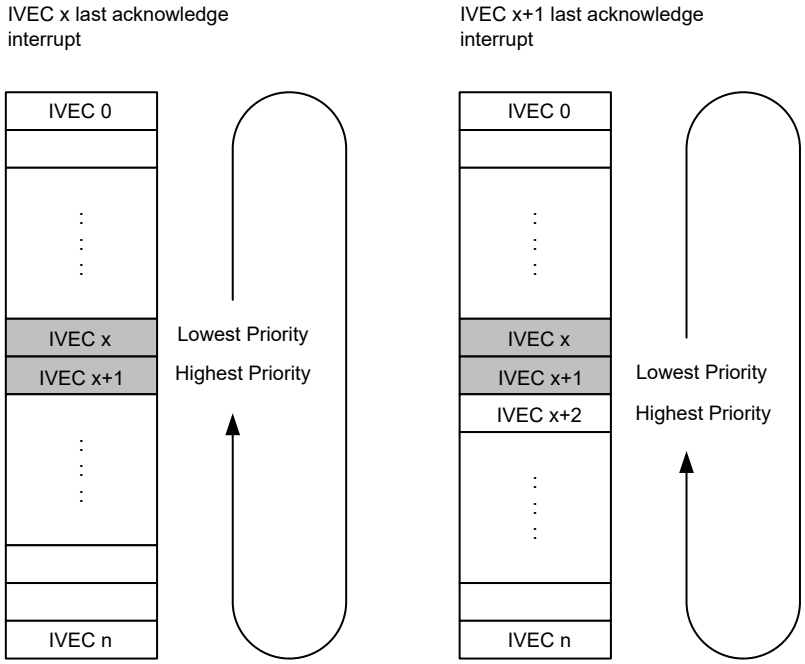
在使用静态优先级中断方案的某些情况下，可能会发生中断饥饿。当应用程序具有较高优先级（较低向量地址）的中断，而这些中断经常被触发时，会导致较低优先级中断得不到处理。轮转优先级方案旨在防止中断饥饿。

轮转优先级方案适用于优先级为 0 的中断。有关更高优先级中断的说明，请参见 5. 高优先级中断向量和 6. 不可屏蔽中断两章。

4.1 操作

使用轮转优先级方案时，中断控制器将存储最新响应中断的地址；如果有多个中断请求待处理，则在轮到处理下一个中断时，存储的地址将具有最低优先级。因此，将为下一个更高的中断向量地址分配最高优先级。有关方案的图形表示，请参见图 4-1。

图 4-1. 使用轮转中断方案时的中断优先级



通过将 1 写入 CPUINT.CTRLA 寄存器中的 LVL0RR 位来使能轮转优先级方案。CPUINT.LVL0PRI 寄存器将包含上个响应中断的地址。当中断控制器决定下一个要执行的中断时，上个响应中断将成为最低优先级的中断。

通过更改 CPUINT.LVL0PRI 中的值，CPU 可以更改下一个具有最低和最高优先级的中断。该寄存器的默认值为 0x00，因此复位后，最低地址中断向量的优先级最高。

4.2 示例

以下代码段给出了有关如何配置轮转调度方案的示例。

轮转中断调度配置

```
//包含具有寄存器和向量定义的器件头文件 io.h
#include <avr/io.h>
```

```
//interrupt.h 包含与 ISR 相关的内容
#include <avr/interrupt.h>

void round_robin_interrupt_scheduling_example(void) {
    //将轮转调度使能位置 1
    CPUINT.CTRLA |= CPUINT_LVL0RR_bm;

    //有关如何改变初始中断优先级的示例
    CPUINT.LVL0PRI = PORTA_PORT_vect_num;

    //允许全局中断
    sei();
}

ISR(PORTA_PORT_vect){
    //该中断最初具有最低优先级，但是
    //由于每次执行中断时 LVL0PRI 将由中断控制器更新，
    //因此将发生变化。
}
```

Atmel | START 示例

还提供使用 Atmel | START 的轮转中断调度的用例。有关更多信息，请参见 Get Source Code from Atmel | START 一章。

5. 高优先级中断向量

当需要比已定义中断优先级更高的中断优先级时，可以为一个中断向量分配优先级 1。该中断的优先级将高于所有 0 级优先级中断，并能够中断正在进行的 0 级优先级中断处理程序。

除了不可屏蔽中断（Non-Maskable Interrupt，NMI）外，默认情况下，将为所有中断源都分配 0 级优先级。不可屏蔽中断是优先级高于 1 级优先级中断的惟一中断。有关更多信息，请参见 6. 不可屏蔽中断一章。

5.1 操作

要使用高优先级向量功能，只需将所需的中断向量地址写入 CPUINT.LVL1VEC 寄存器，将为该中断分配 1 级优先级。当寄存器值为 0x00 时，将禁止该功能。有关中断向量映射的信息，请参见器件数据手册。

如果被 1 级中断中断，则 1 级中断处理程序完成后，0 级中断处理程序将继续执行。必须确保 1 级中断不会干扰任何时间关键型 0 级中断的执行。可通过读取 CPUINT.STATUS 寄存器中的 LVL0EX 位来检查 1 级中断是否已中断正在执行的 0 级 ISR。

5.2 示例

以下代码段给出了有关如何配置高优先级中断向量的示例。

高优先级中断配置

```
//包含具有寄存器和向量定义的器件头文件 io.h
#include <avr/io.h>
//interrupt.h 包含与 ISR 相关的内容
#include <avr/interrupt.h>

void high_priority_interrupt_example(void) {
    //设置 1 级中断向量
    CPUINT.LVL1VEC = PORTA_PORT_vect_num;

    //允许全局中断
    sei();
}

ISR(PORTA_PORT_vect){
    //在 CPUINT.LVL1VEC 中配置之后，
    //该中断将具有 1 级优先级。

    //检查 0 级 ISR 是否已被中断
    if (CPUINT.STATUS & CPUINT_LVL0EX_bm) {

    }
}
```

6. 不可屏蔽中断

不可屏蔽中断（NMI）通常是系统关键中断，并且始终优先于所有其他中断。此外，即使禁止全局中断，它们也会得到响应。

6.1 操作

存在具有硬接线 NMI 优先级的特定向量，其优先级无法更改。这些特定向量的使能方式与其他中断必须相同。有关可用的 NMI 源的信息，请参见器件数据手册的中断向量映射。

通过读取 CPUINT.STATUS 寄存器中的 LVL0EX 和 LVL1EX 位，应用程序可以检查 NMI 是否已中断任何 0 级或 1 级 ISR。

6.2 示例

以下代码段演示了如何实现不可屏蔽中断。

不可屏蔽中断配置，CRCSCAN

```
//包含具有寄存器和向量定义的器件头文件 io.h
#include <avr/io.h>
//interrupt.h 包含与 ISR 相关的内容
#include <avr/interrupt.h>

void nmi_example(void) {
    //将 CRCSCAN 外设配置为 NMI
    CRCSCAN.CTRLB = CRCSCAN_MODE_PRIORITY_gc | CRCSCAN_SRC_FLASH_gc;
    CRCSCAN.CTRLA = CRCSCAN_NMIEN_bm | CRCSCAN_ENABLE_bm;
}

ISR(CRCSCAN_NMI_vect) {
    //如果 CRC 扫描失败，将在此处理 NMI 中断
    //NMI 请求将保持有效状态，直到系统复位，并且无法禁止

    //检查 0 级或 1 级 ISR 是否已被中断
    if (CPUINT.STATUS & (CPUINT_LVL0EX_bm | CPUINT_LVL1EX_bm) {

    }
}
```

有关如何准备代码项目以便使用 CRCSCAN 外设的更多信息，请参见应用笔记 AN2521 - CRCSCAN on Devices in the tinyAVR® 1-Series。

7. 精简向量表

如果应用仅使用少量中断，则只需要这些中断对应的向量位置；因此，向量表的大部分空间都未使用。精简向量表（Compact Vector Table，CVT）是实现更高效中断管理方案的一种方法，因为该方法会将中断向量表截短为几个向量。

7.1 操作

tinyAVR 0 和 1 系列以及 megaAVR 0 系列 MCU 支持三向量 CVT 模式，其中中断向量表根据表 7-1 重新组织。每个向量将用于处理相应优先级的所有中断。

表 7-1. 精简向量映射

向量编号	外设源	定义
0	RESET	复位
1	NMI	不可屏蔽中断向量
2	LVL1	1 优先级中断向量
3	LVL0	0 优先级中断向量

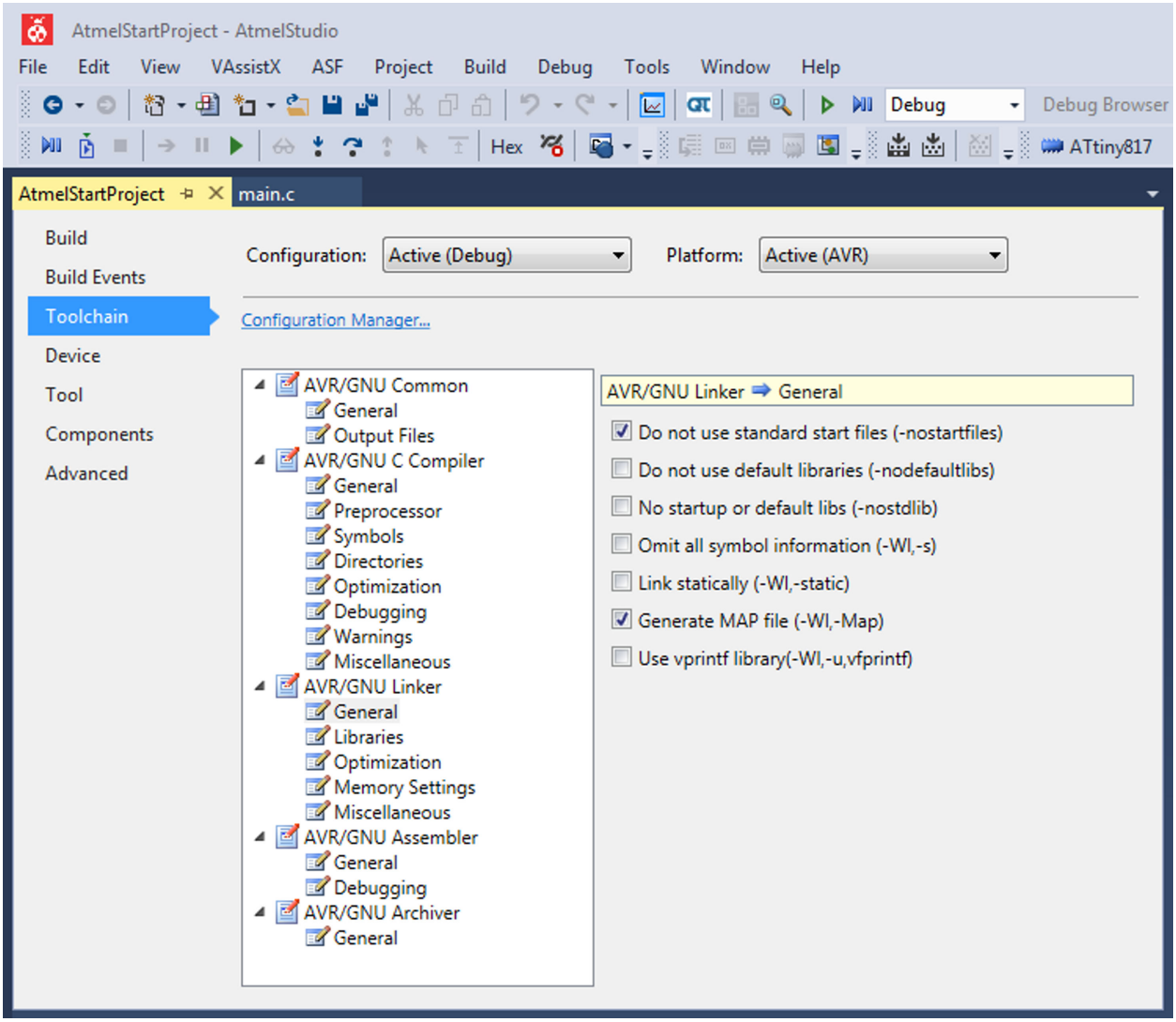
当应用只需要几个中断时，这非常有用。减小向量表大小意味着它将占用更少的闪存，从而释放空间以供应用程序代码使用。但是，请注意，在使用此方法时，可能需要在 ISR 中添加其他代码以找出实际请求处理的中断，这会延长执行时间。

通过将 1 写入 CPUINT.CTRLA 中的 CVT 位来使能精简向量表。该系统关键寄存器位具有可防止意外修改的配置更改保护（CCP）。有关 CCP 的详细信息，请参见相关器件数据手册中的 CPU 一章。

7.2 示例

要利用较小的向量表，需要替换 AVR-GCC 使用的标准启动文件。在编译项目时，通过设置链接器标志-*nostartfiles* 来禁止这些文件。在 Atmel Studio 7.0 中，可在 Project Properties（项目属性）（*Alt+F7*）- Toolchain（工具链）- AVR/GNU Linker（AVR/GNU 链接器）- General（通用）中，找到该选项，如图 7-1 所示。

图 7-1. 配置“Do not use standard start files”，Atmel Studio 7.0



以下代码段给出了替换标准启动文件所需的代码的示例。在将来的工具链版本中，配置精简向量表时不需要该步骤。

精简向量表的启动代码示例

```
//包含具有必要定义的器件头文件 io.h
#include <avr/io.h>

//声明 main 函数
extern int main(void);

//通过预先声明 ISR 以方便编译器处理
void __vector_cvt_nmi(void) __attribute__((weak, alias("dummy_handler")));
void __vector_cvt_lvl1(void) __attribute__((weak, alias("dummy_handler")));
void __vector_cvt_lvl0(void) __attribute__((weak, alias("dummy_handler")));

//设置向量段
//rjmp 指令可处理 8k 代码空间，因此用于
//具有 8k 闪存或更小闪存的器件上的向量表。其他器件
//使用 jmp 指令。
__attribute__((section(".vectors"), naked)) void vectors(void)
```

```
{
#if (PROGMEM_SIZE <= 0x2000)
    asm("rjmp __ctors_end");
    asm("rjmp __vector_cvt_nmi");
    asm("rjmp __vector_cvt_lvl1");
    asm("rjmp __vector_cvt_lvl0");
#else
    asm("jmp __ctors_end");
    asm("jmp __vector_cvt_nmi");
    asm("jmp __vector_cvt_lvl1");
    asm("jmp __vector_cvt_lvl0");
#endif
}

//初始化 AVR 内核
__attribute__((section(".init2"), naked)) void init2(void)
{
    //GCC 要求 r1 为 0
    asm("clr r1");
    //确保状态寄存器具有已知状态
    SREG = 0;
    //将堆栈指针指向堆栈末尾
    SPL  = INTERNAL_SRAM_END & 0xff; //（低字节）
    SPH  = (INTERNAL_SRAM_END >> 8); //（高字节）
}

//处理 main
__attribute__((section(".init9"), naked)) void init9(void)
{
    main();

    //跳转到 avr libc 定义的退出处理程序
    //（禁止中断并始终运行空循环）
    asm("jmp _exit");
}

//未使用 ISR 的空处理程序别名
void dummy_handler(void)
{
    while (1)
        ;
}
```

以下代码段给出了如何使用上述示例启动代码中定义的中断向量名称配置精简向量表的示例。

精简向量表配置代码

```
//包含具有寄存器和向量定义的器件头文件 io.h，
//还包含 xmega.h，内含 CCP macro _PROTECTED_WRITE()。
#include <avr/io.h>
//interrupt.h 包含与 ISR 相关的内容
#include <avr/interrupt.h>

void compact_vector_table_example(void) {
    //设置精简向量表
    //读-修改-写以保留其他配置位
    _PROTECTED_WRITE(CPUINT.CTRLA, (CPUINT.CTRLA | CPUINT_CVT_bm));

    //允许全局中断
    sei();
}

ISR(__vector_cvt_lvl0) {
    //将在此处理所有 0 级中断
    if (PORTA.INTFLAGS) {
        //有关如何在 CVT 模式下处理 PORTA 中断的示例
    }
}
```

```
ISR(__vector_cvt_lvl1) {
    //将在此处理 1 级优先级中断（如果允许该中断）
}

ISR(__vector_cvt_nmi) {
    //将在此处理 NMI 中断（如果允许该中断）
}
```

Atmel | START 示例

还提供使用 Atmel | START 的精简向量表的用例。有关更多信息，请参见 11. 从 Atmel | START 获取源代码章节。

8. 重定位向量表

默认情况下，向量表位于 tinyAVR 0 和 1 系列以及 megaAVR 0 系列中的应用程序段。但是，可以将向量表重定位到引导段的开头；自举程序可以在其代码中重定位向量表，然后在进入主应用程序之前将向量表更改到原来的位置。

也可以将向量表的重定位与精简向量表（CVT）结合使用。有关 CVT 的更多信息，请参见 7. 精简向量表 一章。

8.1 操作

通过将 1 写入 CPUINT.CTRLA 中的 IVSEL 位，可使能将中断向量表移至闪存中引导段的开头。该系统关键寄存器位具有可防止意外修改的配置更改保护（CCP）。有关 CCP 的详细信息，请参见相关器件数据手册中的 CPU 一章。

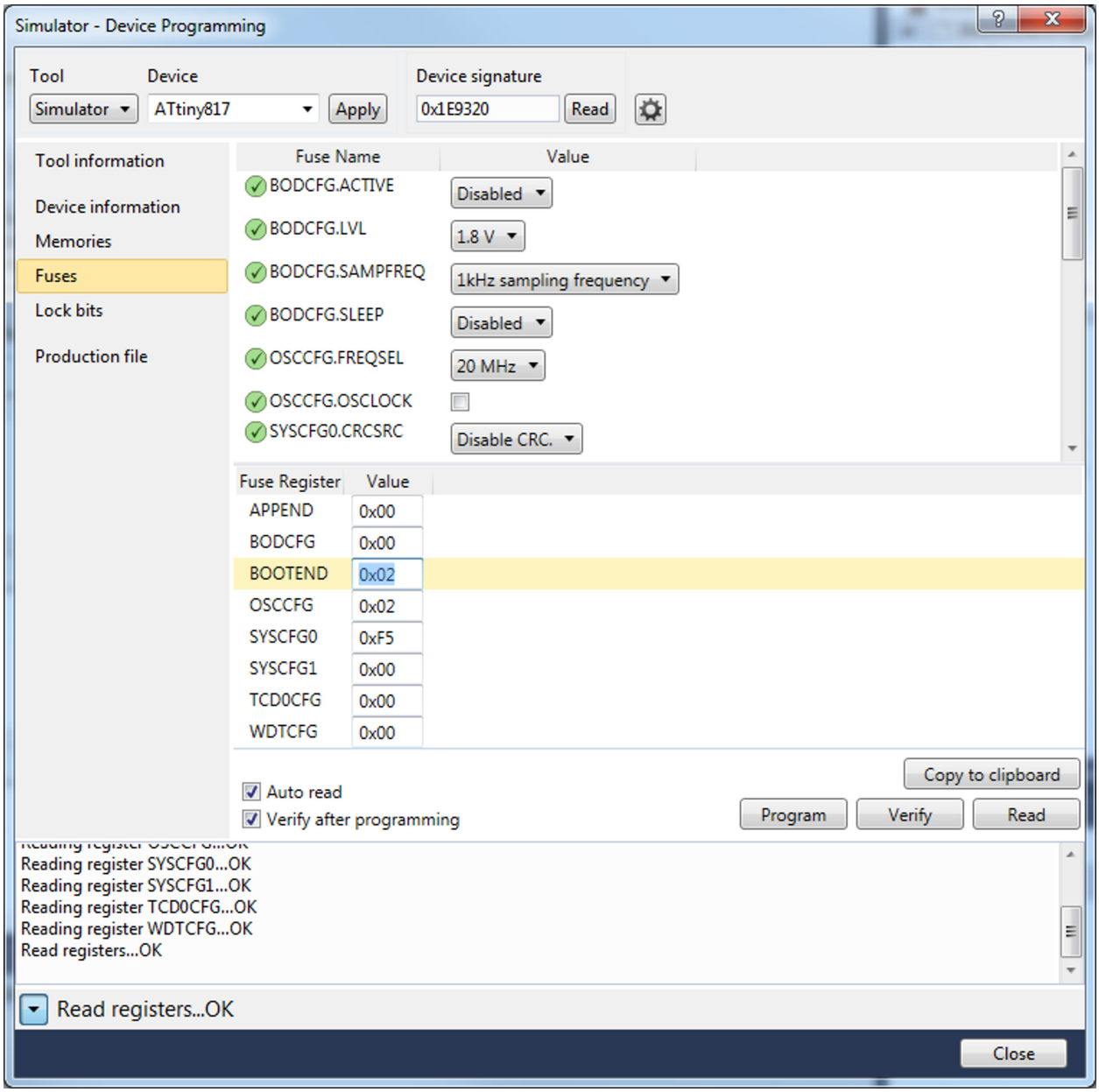
由于应用程序段位于引导段之后，因此中断向量表的实际地址由 BOOTEND 熔丝确定。该熔丝以 256 字节的块为单位设置引导段的大小。当 IVSEL 置 1 时，中断向量表将位于闪存的开头。请注意，如果 BOOTEND 熔丝和 APPEND 熔丝（应用程序段的大小）都设置为 0x00，则整个闪存配置为引导，中断向量表将位于闪存的开头。

两种 IVSEL 状态下的复位向量地址均为 0x0000（闪存开头）。这与引导中断向量表中的最低地址相对应，这意味着复位向量将是复位发生后执行的第一条指令。

8.2 示例

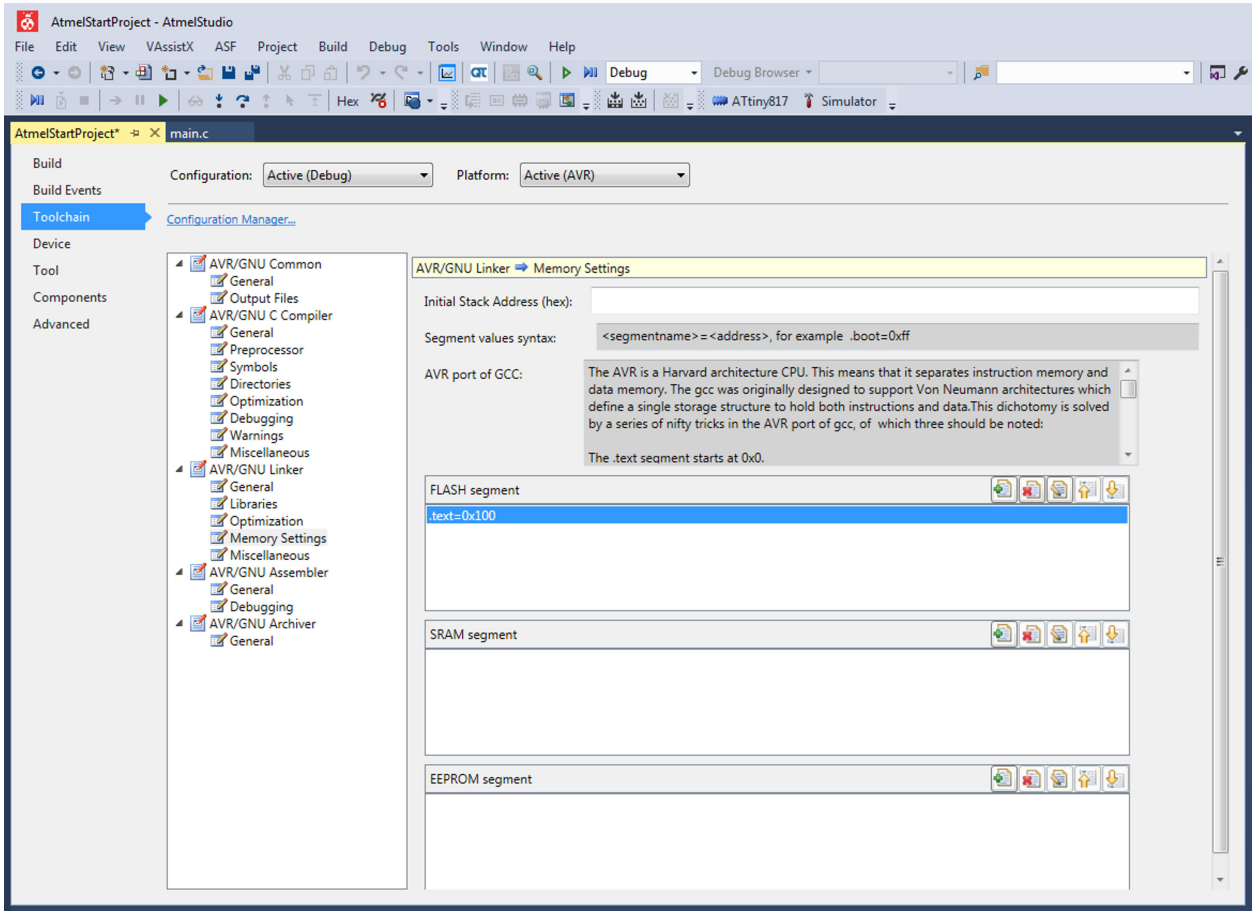
本示例将使用大小为 512 字节的自举程序段。这意味着 BOOTEND 熔丝需要配置为 512/256 = 0x02。在 Atmel Studio 7.0 中，可使用 Device Programming（器件编程）（*Ctrl+Shift+P*）- Fuses（熔丝）将该熔丝写入器件，如图 8-1 所示。

图 8-1. 配置 BOOTEND 熔丝，Atmel Studio 7.0



AVR® GCC 项目还必须配置为将应用程序段的开头移位 512 个字节。这是通过更改链接器中 *text* 段的开头来实现的，该值必须以十六进制字大小设置，例如：*-Wl,-section-start=bootloader=0x100*。在 Atmel Studio 7.0 中，可通过在 Project Properties（*Alt+F7*）- Toolchain - AVR/GNU Linker - Memory Settings（存储器设置）的 FLASH segment（闪存段）中添加.*text=0x100* 来找到相关内容，如图 8-2 所示。

图 8-2. 配置应用程序段的开头，AVR® GCC，Atmel Studio 7.0



以下代码示例给出了如何重定位向量表。

重定位向量表配置

```
//包含具有寄存器和向量定义的器件头文件 io.h，
//还包含 xmega.h，内含 CCP macro _PROTECTED_WRITE()。
#include <avr/io.h>
//interrupt.h 包含与 ISR 相关的内容
#include <avr/interrupt.h>

void relocating_vector_table_example(void) {
    //将中断向量选择位置 1
    //读-修改-写以保留其他配置位
    _PROTECTED_WRITE(CPUINT.CTRLA, (CPUINT.CTRLA | CPUINT_IVSEL_bm));

    //允许全局中断
    sei();
}

ISR(PORTA_PORT_vect){
    //将 IVSEL 位置 1 后，中断向量表
    //将从 BOOTEND*256 字节移位到闪存的开头（0x0000）。

    //在应用程序代码中使用中断时，IVSEL 必须清零，并且将
    //使用闪存中的.text 段编译的代码设置为 BOOTEND*128 字。
}
```

9. 总结

选择轮询还是中断在很大程度上取决于应用要求，但是，可以遵循一些准则。

首先，如果您的产品由电池供电或功耗受限，请避免轮询。轮询要求 CPU 保持工作模式，功耗可能比中断驱动系统高很多。但是，如果功耗增加不是问题，并且只需要检查一个或两个状态位，则很容易用软件实现轮询。

其次，以有效且可预测的方式管理和维护复杂的应用程序可能很困难。而使用提供的中断优先级方案就容易得多，此时可将一个中断提升到更高优先级。

第三，如果应用程序的代码大小受到限制，可以通过使用精简向量表来减小代码大小。在某些情况下，仅为自举程序使用精简向量表，然后切换回应用程序段中的全大小向量表可能较为有利。这可能会减少引导段大小，为应用程序段留出更多空间。

如本应用笔记所示，新的 tinyAVR 0 和 1 系列以及 megaAVR 0 系列 MCU 在中断处理方面提供功能强大的流程控制功能。这样一来，设计人员便能根据自己的设计要求定制功耗和响应能力。

10. 相关文档

以下文档与本应用笔记所涵盖的器件和主题相关。

- AVR Instruction Set Manual：
  - http://www.microchip.com/mymicrochip/filehandler.aspx?ddocname=en589391
- AN2521 - CRCSCAN on Devices in the tinyAVR® 1-Series：
  - http://www.microchip.com//wwwAppNotes/AppNotes.aspx?appnote=en599876

11. 从 Atmel | START 获取源代码

示例代码可通过 Atmel | START 获得，Atmel | START 是一种基于 Web 的工具，可通过图形用户界面（Graphical User Interface，GUI）配置应用程序代码。可以通过下面提供的直接示例代码链接或 Atmel | START 首页上的 *BROWSE EXAMPLES*（浏览示例）按钮，为 Atmel Studio 和 IAR Embedded Workbench®下载代码。

Atmel | START 网页： http://microchip.com/start

示例代码

- CPUINT 轮转中断方案：
  - http://start.atmel.com/#example/Atmel:cpuint_examples_tinyavr_megaavr_01:1.0.0::Application:CPUINT_Round-Robin_Scheme:
- CPUINT 精简向量表：
  - http://start.atmel.com/#example/Atmel:cpuint_examples_tinyavr_megaavr_01:1.0.0::Application:CPUINT_Compact_Vector_Table:

有关详细信息和示例项目的相关信息，请按 Atmel | START 中的 *User guide*（用户指南）按钮。*User guide* 按钮可以在该网页中找到，方法是在 Atmel | START 项目配置器中的仪表板视图中单击项目名称。

Atmel Studio

在 Atmel | START 的网页中单击 *DOWNLOAD SELECTED EXAMPLE*（下载所选示例），为 Atmel Studio 下载.atzip 文件形式的代码。要从 Atmel | START 下载文件，请单击 *EXPORT PROJECT*（导出项目），然后单击 *DOWNLOAD PACK*（下载包）。

双击下载的.atzip 文件，项目将导入到 Atmel Studio 7.0。

IAR Embedded Workbench

有关如何在 IAR Embedded Workbench 中导入项目的信息，请打开 Atmel | START 用户指南，选择 *Using Atmel Start Output in External Tools*（使用外部工具中的 Atmel Start 输出），然后选择 *IAR Embedded Workbench*。单击 Atmel | START 首页右上角的 *About*（关于）或项目配置器中右上角的 *Help And Support*（帮助和支持），均可找到 Atmel | START 用户指南的链接。

12. 版本历史

文档版本	日期	备注
B	2018 年 2 月	更新了中断向量选择（IVSEL）的说明
A	2017 年 12 月	文档初始版本

Microchip 网站

Microchip 网站 http://www.microchip.com/为客户提供在线支持。客户可通过该网站方便地获取文件和信息。只要使用常用的互联网浏览器即可访问，网站提供以下信息：

- **产品支持**——数据手册和勘误表、应用笔记和示例程序、设计资源、用户指南以及硬件支持文档、最新的软件版本以及归档软件
- **一般技术支持**——常见问题（FAQ）、技术支持请求、在线讨论组以及 Microchip 顾问计划成员名单
- **Microchip 业务**——产品选型和订购指南、最新 Microchip 新闻稿、研讨会和活动安排表、Microchip 销售办事处、代理商以及工厂代表列表

变更通知客户服务

Microchip 的变更通知客户服务有助于客户了解 Microchip 产品的最新信息。注册客户可在他们感兴趣的某个产品系列或开发工具发生变更、更新、发布新版本或勘误表时，收到电子邮件通知。

欲注册，请登录 Microchip 网站 http://www.microchip.com/。在“支持”（Support）下，点击“变更通知客户”（Customer Change Notification）服务后按照注册说明完成注册。

客户支持

Microchip 产品的用户可通过以下渠道获得帮助：

- 代理商或代表
- 当地销售办事处
- 应用工程师（FAE）
- 技术支持

客户应联系其代理商、代表或应用工程师（FAE）寻求支持。当地销售办事处也可为客户提供帮助。本文档后附有销售办事处的联系方式。

也可通过以下网站获得技术支持： http://www.microchip.com/support

Microchip 器件代码保护功能

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 愿意与关心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。

法律声明

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，否则在 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、AnyRate、AVR、AVR 徽标、AVR Freaks、BitCloud、chipKIT、chipKIT 徽标、CryptoMemory、CryptoRF、dsPIC、FlashFlex、flexPWR、Heldo、JukeBlox、KeeLoq、Kleer、LANCheck、LINK MD、maXStylus、maXTouch、MediaLB、megaAVR、MOST、MOST 徽标、MPLAB、OptoLyzer、PIC、picoPower、PICSTART、PIC32 徽标、Prochip Designer、QTouch、SAM-BA、SpyNIC、SST、SST 徽标、SuperFlash、tinyAVR、UNI/O 和 XMEGA 是 Microchip Technology Incorporated 在美国和其他国家或地区的注册商标。

ClockWorks、The Embedded Control Solutions Company、EtherSynch、Hyper Speed Control、HyperLight Load、IntelliMOS、mTouch、Precision Edge 和 Quiet-Wire 为 Microchip Technology Incorporated 在美国的注册商标。

Adjacent Key Suppression、AKS、Analog-for-the-Digital Age、Any Capacitor、AnyIn、AnyOut、BodyCom、CodeGuard、CryptoAuthentication、CryptoAutomotive、CryptoCompanion、CryptoController、dsPICDEM、dsPICDEM.net、Dynamic Average Matching、DAM、ECAN、EtherGREEN、In-Circuit Serial Programming、ICSP、INICnet、Inter-Chip Connectivity、JitterBlocker、KleerNet、KleerNet 徽标、memBrain、Mindi、MiWi、motorBench、MPASM、MPF、MPLAB Certified 徽标、MPLIB、MPLINK、MultiTRAK、NetDetach、Omniscient Code Generation、PICDEM、PICDEM.net、PICkit、PICtail、PowerSmart、PureSilicon、QMatrix、REAL ICE、Ripple Blocker、SAM-ICE、Serial Quad I/O、SMART-I.S.、SQI、SuperSwitcher、SuperSwitcher II、Total Endurance、TSHARC、USBCheck、VariSense、ViewSpan、WiperLock、Wireless DNA 和 ZENA 为 Microchip Technology Incorporated 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 是 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。

ISBN： 978-1-5224-4334-6

DNV 认证的质量管理体系

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 和 dsPIC® DSC、KEELOQ®跳码器件、串行 EEPROM、单片机外设、非易失性存储器及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	亚太地区	亚太地区	欧洲
公司总部 2355 West Chandler Blvd. 钱德勒，亚利桑那州 85224-6199 电话：480-792-7200 传真：480-792-7277 技术支持： http://www.microchip.com/support 网址： www.microchip.com **亚特兰大** 德卢斯，佐治亚州 电话：678-957-9614 传真：678-957-1455 **奥斯汀，德克萨斯州** 电话：512-257-3370 **波士顿** 韦斯特伯鲁，马萨诸塞州 电话：774-760-0087 传真：774-760-0088 **芝加哥** 艾塔斯卡，伊利诺伊州 电话：630-285-0071 传真：630-285-0075 **达拉斯** 阿迪森，德克萨斯州 电话：972-818-7423 传真：972-818-2924 **底特律** 诺维，密歇根州 电话：248-848-4000 **休斯顿，德克萨斯州** 电话：281-894-5983 **印第安纳波利斯** 诺布尔斯维尔，印第安纳州 电话：317-773-8323 传真：317-773-5453 电话：317-536-2380 **洛杉矶** 米慎维荷，加利福尼亚州 电话：949-462-9523 传真：949-462-9608 电话：951-273-7800 **罗利，北卡罗来纳州** 电话：919-844-7510 **纽约，纽约州** 电话：631-435-6000 **圣何塞，加利福尼亚州** 电话：408-735-9110 电话：408-436-4270 **加拿大 - 多伦多** 电话：905-695-1980 传真：905-695-2078	**澳大利亚 - 悉尼** 电话：61-2-9868-6733 **中国 - 北京** 电话：86-10-8569-7000 **中国 - 成都** 电话：86-28-8665-5511 **中国 - 重庆** 电话：86-23-8980-9588 **中国 - 东莞** 电话：86-769-8702-9880 **中国 - 广州** 电话：86-20-8755-8029 **中国 - 杭州** 电话：86-571-8792-8115 **中国 - 香港特别行政区** 电话：852-2943-5100 **中国 - 南京** 电话：86-25-8473-2460 **中国 - 青岛** 电话：86-532-8502-7355 **中国 - 上海** 电话：86-21-3326-8000 **中国 - 沈阳** 电话：86-24-2334-2829 **中国 - 深圳** 电话：86-755-8864-2200 **中国 - 苏州** 电话：86-186-6233-1526 **中国 - 武汉** 电话：86-27-5980-5300 **中国 - 西安** 电话：86-29-8833-7252 **中国 - 厦门** 电话：86-592-2388138 **中国 - 珠海** 电话：86-756-3210040	**印度 - 班加罗尔** 电话：91-80-3090-4444 **印度 - 新德里** 电话：91-11-4160-8631 **印度 - 浦那** 电话：91-20-4121-0141 **日本 - 大阪** 电话：81-6-6152-7160 **日本 - 东京** 电话：81-3-6880- 3770 **韩国 - 大邱** 电话：82-53-744-4301 **韩国 - 首尔** 电话：82-2-554-7200 **马来西亚 - 吉隆坡** 电话：60-3-7651-7906 **马来西亚 - 槟榔屿** 电话：60-4-227-8870 **菲律宾 - 马尼拉** 电话：63-2-634-9065 **新加坡** 电话：65-6334-8870 **台湾地区 - 新竹** 电话：886-3-577-8366 **台湾地区 - 高雄** 电话：886-7-213-7830 **台湾地区 - 台北** 电话：886-2-2508-8600 **泰国 - 曼谷** 电话：66-2-694-1351 **越南 - 胡志明市** 电话：84-28-5448-2100	**奥地利 - 韦尔斯** 电话：43-7242-2244-39 传真：43-7242-2244-393 **丹麦 - 哥本哈根** 电话：45-4450-2828 传真：45-4485-2829 **芬兰 - 埃斯波** 电话：358-9-4520-820 **法国 - 巴黎** 电话：33-1-69-53-63-20 传真：33-1-69-30-90-79 **德国 - 加兴** 电话：49-8931-9700 **德国 - 哈恩** 电话：49-2129-3766400 **德国 - 海尔布隆** 电话：49-7131-72400 **德国 - 卡尔斯鲁厄** 电话：49-721-625370 **德国 - 慕尼黑** 电话：49-89-627-144-0 传真：49-89-627-144-44 **德国 - 罗森海姆** 电话：49-8031-354-560 **以色列 - 若那那市** 电话：972-9-744-7705 **意大利 - 米兰** 电话：39-0331-742611 传真：39-0331-466781 **意大利 - 帕多瓦** 电话：39-049-7625286 **荷兰 - 德卢内市** 电话：31-416-690399 传真：31-416-690340 **挪威 - 特隆赫姆** 电话：47-72884388 **波兰 - 华沙** 电话：48-22-3325737 **罗马尼亚 - 布加勒斯特** 电话：40-21-407-87-50 **西班牙 - 马德里** 电话：34-91-708-08-90 传真：34-91-708-08-91 **瑞典 - 哥德堡** 电话：46-31-704-60-40 **瑞典 - 斯德哥尔摩** 电话：46-8-5090-4654 **英国 - 沃金厄姆** 电话：44-118-921-5800 传真：44-118-921-5820