Application Report
ZHCA601 – March 2014

数字电源 UCD3138 程序中的快中断的屏蔽方法

Neil Li, Sundy Xu　　China Telecom Application Team

摘要

数字电源控制器 UCD3138 设计有多个中断向量并可以被映射为标准中断或快中断。UCD3138 的程序在执行一些关键语句前，需要暂时屏蔽快中断以防止该关键语句的执行被打断。可以通过 write_reqmask ( ) 和 disable_fast_interrupt ( ) 两种方法来屏蔽快中断，实际应用中，需要注意这两个方法适用在不同的程序位置中，包括背景环程序和标准中断子程序。本文详细分析了这两种方法适用在不同位置的原因，并通过演示实验说明，暂时屏蔽快中断不会造成快中断请求的丢失。

目录

图

1 引言

1.1 UCD3138 的软件架构

UCD3138 的软件架构如图 1 所示。在外设配置与初始化完成后， 软件运行背景环程序，并接受标准中断或快中断的中断进入。以下是背景环程序、标准中断和快中断的说明：

1）背景环程序：由 for 函数构成的无条件无限循环。内部主要包括 PMBus 通信程序的管理和操作以及时间响应慢的操作等。

2）标准中断：可以设置为每 200us 触发进入一次，子程序中主要包括状态机的管理和操作，故障的响应和处理等，是 UCD3138 软件的核心部分。

3）快中断：可以中断背景环和标准中断的执行，子程序中实现诸如 OVP（过压保护）等故障的快速响应和处理。

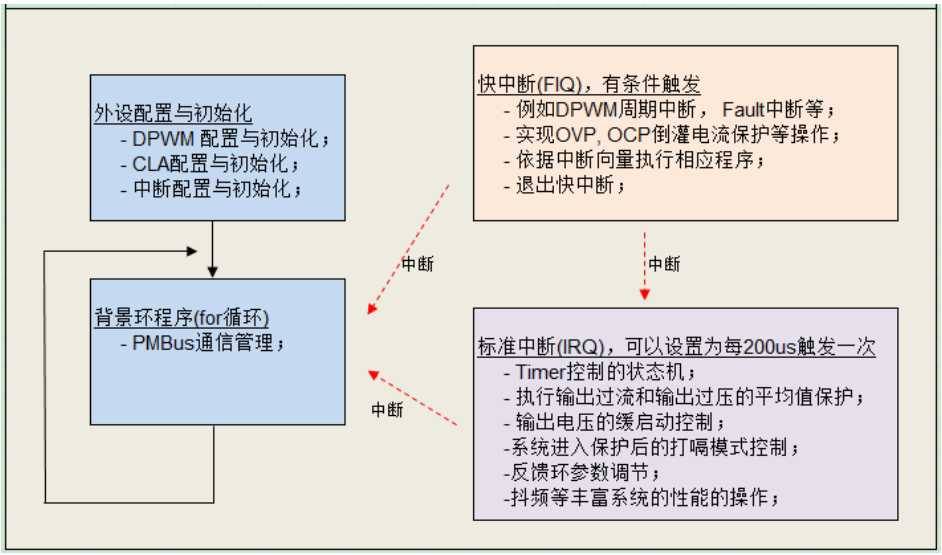

图 1：UCD3138 软件架构

1.2 中断的初始化与使能

在中断初始化函数中包括了下面的语句，用以完成标准中断和快中断的配置。

```
disable_interrupt();          //make sure interrupt is disabled
disable_fast_interrupt();     //make sure fast interrupt is disabled
write_reqmask(0x20000000 | 0x00008000 ); //enable pwm3cmp and DPWM1 interrupt
write_firqpr (0x20000000); // DPWM1 interrupt(End of 16th period)is mapped to FIQ
enable_fast_interrupt();      //enable interrupt
enable_interrupt();           //enable fast interrupt
```

如上代码所示，在配置前先禁止对标准中断请求和快中断请求的响应。然后，通过 write_reqmask( )语句和 write_firqpr( )语句来完成 32 个中断向量的配置（配置为标准中断请求或快中断请求）。上面代码设置 pwm3cmp 比较中断为标准中断请求，设置 DPWM1 周期中断为快中断请求。最后两行代码是使能对标准中断请求和快中断请求的响应。

需要注意的是，在配置为快中断请求前，必须先将其配置为标准中断请求。

1.3 快中断在程序中的屏蔽

程序运行过程中（已经完成了快中断的配置和使能），在执行某些关键语句前需要暂时屏蔽快中断以防止被打断，该语句执行完毕后即可取消对快中断的屏蔽。

对快中断的屏蔽一般是在标准中断中，也可以在背景环程序中。可以通过调用 disable_interrupt ( ) 完成，也可以通过调用 write_reqmask ( ) 来完成，但二者的原理并不相同，因此适用在不同的程序位置中（标准中断或背景环程序）。

下文详细介绍在标准中断和背景环程序中如何正确的屏蔽快中断。

2 快中断的屏蔽方法及详细分析

2.1 标准中断内部屏蔽快中断

1. 使用 write_reqmask ( )在标准中断屏蔽快中断

如下代码，在标准中断中（regulation 状态机）尝试使用 write_reqmask ( )屏蔽 DPWM1 周期快中断。其中，DPWM3A 配置为 GPIO，用做指示信息。在快中断程序中，已经配置为 GPIO 模式的 DPWM2A 先被赋值为 1，随后赋值为 0。

```
inline void handle_regulated_state(void)
{

  if (Interrupt_Flag==2)
  {
    Dpwm3Regs.DPWMCTRL1.bit.GPIO_A_VAL=1;
    //-----------------------------------
    write_reqmask(0x00008000);
    //-----------------------------------
    Interrupt_Flag=3;
  }

  else if (Interrupt_Flag==3)
  {
    Dpwm3Regs.DPWMCTRL1.bit.GPIO_A_VAL= 0;
    //-----------------------------------
    write_reqmask(0x00008000|0x20000000);
    //-----------------------------------
    Interrupt_Flag=2;
  }

}
```

实测发现，write_reqmask ( )可以很好的屏蔽快中断。

如图 2，当 DPWM3A 为低电平期间，DPWM2A 有高低电平的变化，表明标准中断程序被不停的打断并进入 DPWM1 周期快中断程序。当 DPWM3A（CH2）为高电平期间，DPWM2A（CH1）保持为低， 这表明快中断已被屏蔽；

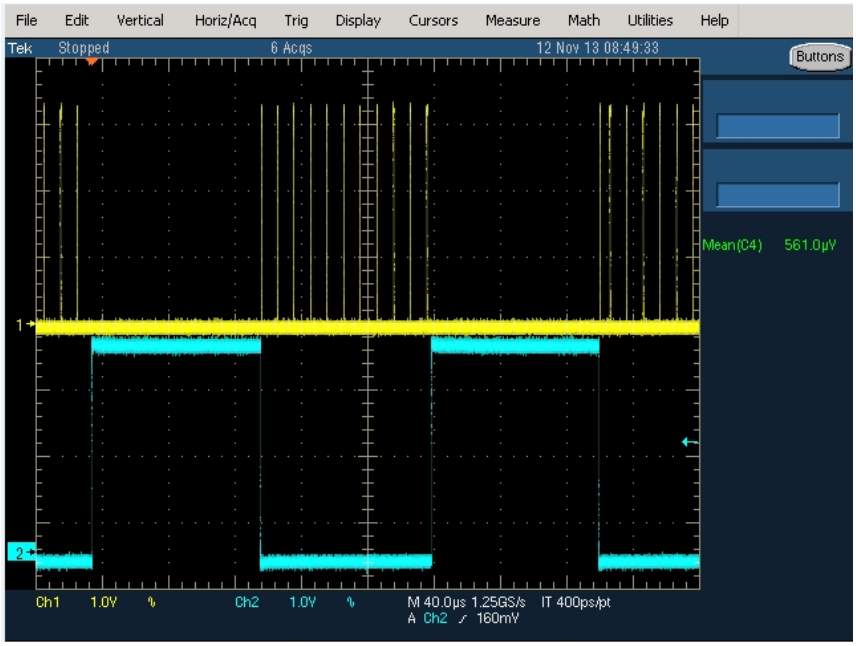

图 2：快中断屏蔽效果展示 1

2. 使用 disable_interrupt ( )在标准中断里屏蔽快中断

如下代码，在标准中断中（regulation 状态机）尝试使用 disable_interrupt ( )屏蔽 DPWM1 对应的标准中断请求；如果成功，DPWM1 快中断亦被屏蔽（*不然，仅仅屏蔽 DPWM1 的中断请求，其对应的标准中断请求还会被执行*）。代码中，DPWM3A 配置为 GPIO，用做指示信息。在 DPWM1 周期快中断程序中，已经配置为 GPIO 模式 DPWM2A 先被赋值为 1，随后赋值为 0。

```
inline void handle_regulated_state(void)
{

  if (Interrupt_Flag==2)
  {
    Dpwm3Regs.DPWMCTRL1.bit.GPIO_A_VAL=1;
    //---------------------------------
    disable_fast_interrupt();
    //---------------------------------
    Interrupt_Flag=3;
  }

  else if (Interrupt_Flag==3)
  {
    Dpwm3Regs.DPWMCTRL1.bit.GPIO_A_VAL= 0;
    //---------------------------------
    enable_fast_interrupt();
    //---------------------------------
    Interrupt_Flag=2;
  }

}
```

实测发现，disable_interrupt ( )在标准中断中不能屏蔽 DPWM1 标准中断，亦无法屏蔽该快中断。如图 3，无论 DPWM3A（CH2）为高电平期间还是低电平期间，DPWM2A（CH1）一直有高低电平的变化，这表明快中断没有能够被屏蔽。

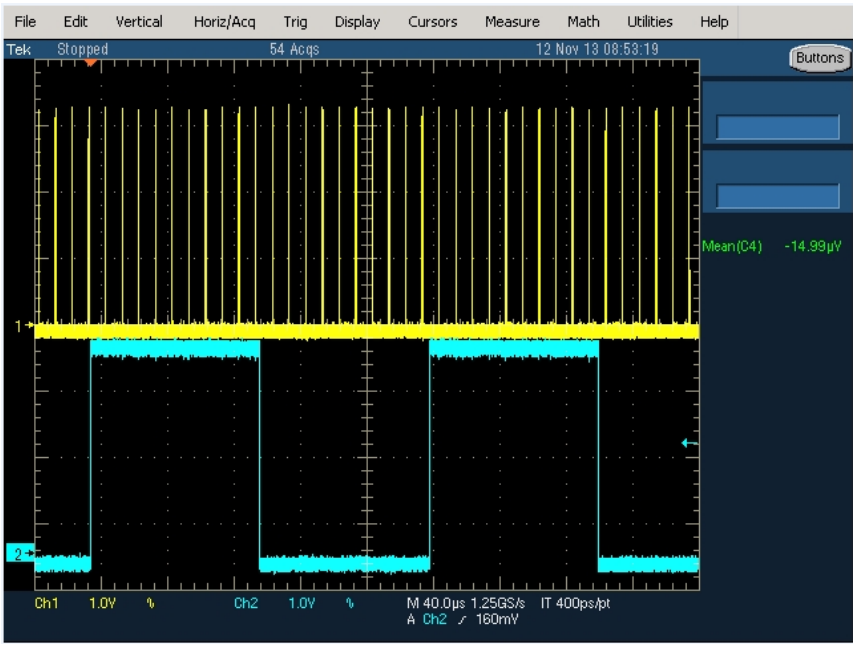

图 3：快中断屏蔽效果展示 2

2.2 背景环中屏蔽快中断

1. 使用 write_reqmask ( )在背景环中屏蔽快中断

如下代码，在背景环程序中，即 for 循环函数中尝试使用 write_reqmask ( )屏蔽 DPWM1 周期快中断。其中，DPWM3A 配置为 GPIO，用做指示信息。在快中断程序中，已经配置为 GPIO 模式的 DPWM2A 先被赋值为 1，随后赋值为 0。

程序运行后，DPWM1 周期快中断已经使能并不断触发。需要屏蔽快中断时，通过 PMBus 总线下发命令将变量 Interrupt_Flag 赋值为 3。

```
    for(;;)
    {
        if (erase_segment_counter > 0)
        {
          erase_task();
        }
        pmbus_handler();

       if (Interrupt_Flag==3)
       {
        Dpwm3Regs.DPWMCTRL1.bit.GPIO_A_VAL=1;
        write_reqmask(0x00008000);
       }
    }
}
```

实测发现，write_reqmask ( )在标准中断中可以很好的屏蔽快中断。

如图 4，当 DPWM3A（CH2）由低变为高后，DPWM2A（CH1）由原来的高低跳变状态变为低电平状态，这表明快中断已被屏蔽。

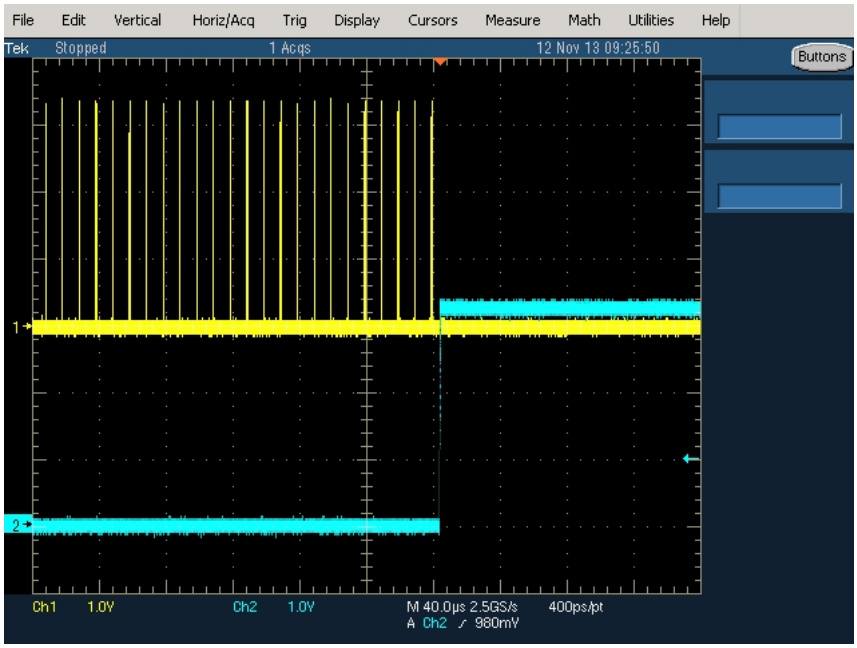

图 4：快中断屏蔽效果展示 3

2. 使用 disable_fast_interrupt ( )在背景环中屏蔽快中断

如下代码，在背景环程序中，即 for 循环函数中尝试使用 disable_fast_interrupt ( )屏蔽 DPWM1 周期快中断。其中，DPWM3A 配置为 GPIO，用做指示信息。在快中断程序中，已经配置为 GPIO 模式的 DPWM2A 先被赋值为 1，随后赋值为 0。

程序运行后，DPWM1 周期中断已经使能为快中断并不断触发。需要屏蔽快中断时，通过 PMBus 总线下发命令将变量 Interrupt_Flag 赋值为 3。

```
    for(;;)
    {
        if (erase_segment_counter > 0)
        {
          erase_task();
        }
        pmbus_handler();

       if (Interrupt_Flag==3)
       {
        Dpwm3Regs.DPWMCTRL1.bit.GPIO_A_VAL=1;
        disable_fast_interrupt();
       }
    }
}
```

实测发现，disable_fast_interrupt ( )在背景环程序中可以很好的屏蔽快中断。如图 5，当 DPWM3A（CH2）由低变为高后，DPWM2A（CH1）由原来的高低跳变状态变为低电平状态，这表明快中断已被屏蔽。

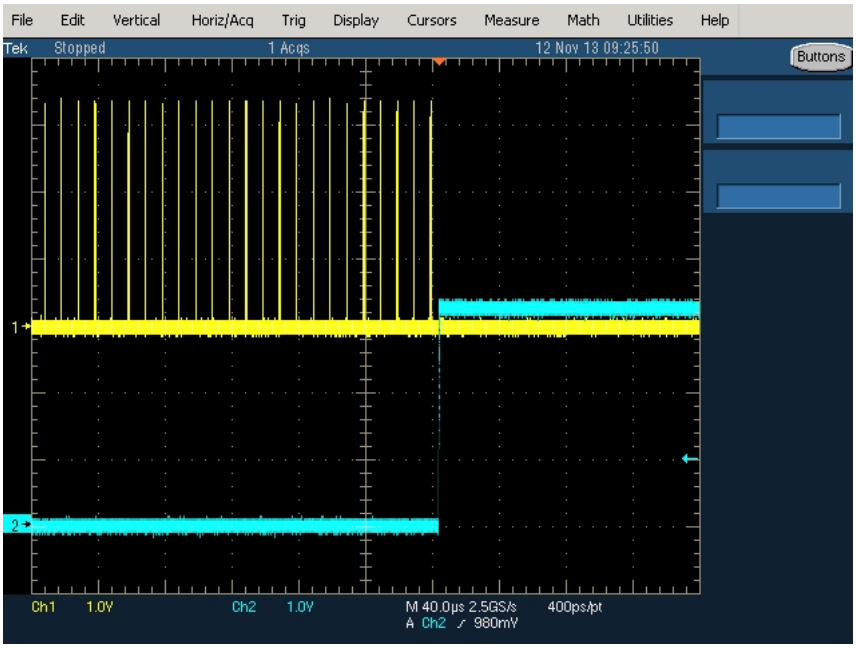

图 5：快中断屏蔽效果展示 4

2.3 两种屏蔽方式的分析

1. 为何在标准中断无法使用 disable_interrupt ( )

UCD3138 使用是的 ARM7TDMI 的内核。内核中包含 1 个程序状态寄存器 CPSR（Current Program Status Register）和 5 个供异常处理程序使用的程序状态保存寄存器 SPSR（Saved Program Status Register）。

- ARM7TDMI 内核所有处理器状态都保存在 CPSR 中。CPSR 包含了 2 个中断禁止位 I 和 F。当 I 置位，IRQ 中断被禁止；当 F 置位，FIQ 中断被禁止。
- 每个异常模式（快中断模式，中断模式，管理模式，中止模式，未定义模式）还带有一个程序状态保存寄存器 SPSR，用于保存任务在异常发生之前的 CPSR。

事实上，disable_interrupt ( )是一个软中断，优先级高于标准中断，中断程序内部的代码如下。**代码是将 SPSR 中 I 位置 1，而在退出软中断时，SPSR 会传递给 CPSR，最终即禁止了 ARM 核对标准中断请求的响应。**

```
asm(" MRS     r0, spsr "); //get saved psr
asm(" BIC     r0, r0, #0x80 "); // clear irq disable
asm(" MSR     spsr, r0"); //restore saved psr
```

在标准中断调用 disable_interrupt ( )后，会将 CPSR 的值改变。但在退出标准中断时，标准中断自身的 SPSR 又会赋值给 CPSR。因此，在退出标准中断后并没有将快中断有效禁止。

2. 为何在标准中断可以使用 write_reqmask ( )

write_reqmask ( ) 本身也是一个软中断。中断程序内部的代码是将 write_reqmask ( )的参数赋值给寄存器 REQMASK。寄存器 REQMASK 的 32 个位置 1 或置 0 则表示是否使能相应的中断请求通道。

在标准中断调用 write_reqmask ( )后，直接修改了 REQMASK 的值，因此中断被屏蔽。CPSR 的值并不需要改变。

3. 为何在背景环程序中可以使用 disable_fast_interrupt ( )

在背景环程序中所调用的 disable_fast_interrupt ( )同样是一个软中断，其代码如下。**代码中是将 SPSR 的 F 位置 1。当退出软中断时，SPSR 会传递给 CPSR，最终即禁止了 ARM 核对快中断请求的响应。**

```
asm(" MRS    r0, spsr "); //get saved psr
asm(" ORR    r0, r0, #0x40 "); // set fiq disable
asm(" MSR    spsr, r0"); //restore saved psr
```

4. 为何在背景环程序中可以使用 write_reqmask ( )

UCD3138 使用的 ARM7TDMI 的内核支持 7 种处理器模式：用户模式，快中断模式，中断模式，管理模式，中止模式，未定义模式和系统模式。除用户模式外，其它均为特权模式。

背景环程序的运行是处于正常工作模式，即“用户模式”。而用户模式为非特权模式，不可以直接修改 REQMASK 寄存器的值。因此，需要借助软中断进入到特权模式后去修改。调用 write_reqmask ( ) 可以进入软中断，并在中断内部修改 REQMASK 寄存器，快中断即被禁止。

综上所述，在标准中断和背景环中可以使用的屏蔽快中断的方法如下表。

屏蔽快中断的方法	应用位置	
	标准中断函数中	背景环函数中
write_reqmask ( )	可以使用	可以使用
disable_interrupt ( )	不能使用	/[1]
disable_fast_interrupt ( )	/[2]	可以使用

- *注 1：背景环函数中，可以使用 disable_fast_interrupt ( ) 直接屏蔽快中断。*
- *注 2：标准中断函数中，如果仅仅屏蔽快中断请求，其对应的标准中断请求还有可能继续执行（如果其优先级比较高）。*

3 快中断在标准中断中的屏蔽演示

3.1 演示代码设计

配置并使能 Fault1 快中断。用 Fault1 引脚电平的上升沿作为中断源来触发快中断。在中断初始化函数中的主要代码如下：

```
FaultMuxRegs.EXTFAULTCTRL.bit.FAULT1_POL = 1;    //rising edge
FaultMuxRegs.EXTFAULTCTRL.bit.FAULT1_INT_EN = 1; //enable fault detection pin interrupt
FaultMuxRegs.EXTFAULTCTRL.bit.FAULT1_DET_EN = 1; //enable fault detection
MiscAnalogRegs.IOMUX.bit.JTAG_DATA_MUX_SEL = 3;  //TDO/TDI Pin Mux Select as: Fault0 & Fault1

disable_interrupt();
disable_fast_interrupt();  //make sure fast interrupt is disabled
write_reqmask(CIMINT_ALL_FAULT_PIN | CIMINT_ALL_PWM2_COMP | CIMINT_ALL_FAULT_MUX);
write_firqpr (CIMINT_ALL_FAULT_PIN | CIMINT_ALL_FAULT_MUX);
enable_fast_interrupt();
enable_interrupt();
```

硬件电路上，将 DPWM3A 引脚连接到 Fault1 引脚。

在标准中断中，首先屏蔽快中断（禁止 ARM 核响应快中断请求），然后将 DPWM3A 赋值为高。DPWM3A 的上升沿会传递到了 Fault1 引脚。虽然此时已有中断请求，但由于快中断被屏蔽，无法进入快中断程序。随后的程序语句取消了对快中断的屏蔽。实现代码如下：

```
int32 temp;
CimRegs.REQMASK.all =  CIMINT_ALL_PWM2_COMP;

for(temp = 0; temp < 100; temp ++);
Dpwm3Regs.DPWMCTRL1.bit.GPIO_A_VAL=1;
for(temp = 0; temp < 100; temp ++);

CimRegs.REQMASK.all = CIMINT_ALL_FAULT_PIN | CIMINT_ALL_PWM2_COMP | CIMINT_ALL_FAULT_MUX;
```

上面代码是通过直接修改寄存器 REQMASK 去屏蔽或使能相关中断，因为标准中断是特权模式，可以直接修改寄存器 REQMASK。当然，在标准中断中调用 write_reqmask ( )去修改 REQMASK 的值也是可以的，上文也已经论证过。

快中断子程序代码中，将 DPWM2A 置高并清空寄存器 FAULTMUXINTSTAT。随后将 DPWM2A 和 DPWM3A 置低。

```
#pragma INTERRUPT(fast_interrupt,FIQ)
void fast_interrupt(void)
{
   register Uint32 temp;

   Dpwm2Regs.DPWMCTRL1.bit.GPIO_A_VAL= 1;
   temp = FaultMuxRegs.FAULTMUXINTSTAT.all;
   Dpwm2Regs.DPWMCTRL1.bit.GPIO_A_VAL= 0;

   Dpwm3Regs.DPWMCTRL1.bit.GPIO_A_VAL=0;

}
```

3.2 实测结果

如图 6，为上面演示实验的实测波形。当 DPWM3A（CH2）引脚（即 Fault1 引脚）出现上升沿时，因快中断被屏蔽，有中断请求却也无法进入快中断程序。随后，屏蔽被取消，快中断子程序得以进入并将 DPWM2A（CH1）置高。退出快中断前 DPWM2A 和 DPWM3A 都被置低。

该实验说明快中断可以在标准中断中被有效的屏蔽，且不会造成中断请求的丢失。

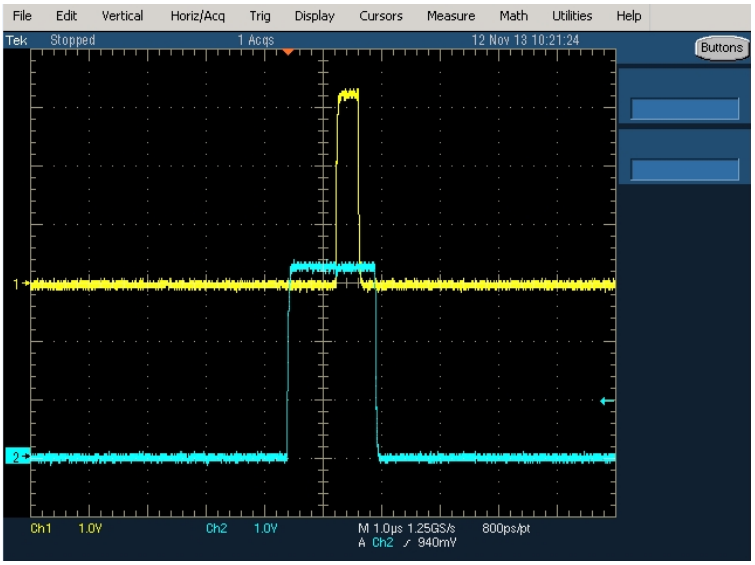

图 6：快中断屏蔽效果展示 5

4 结论

在 UCD3138 的软件中可以通过 write_reqmask ( )和 disable_fast_interrupt ( )来屏蔽快中断，其原理分别是禁止 ARM 核响应中断请求和是否将某个中断向量声明为快中断请求。因此，在实际应用中这两个方法适用在不同的程序位置中，包括背景环程序和标准中断程序。

另外，本文还通过演示实验说明，暂时屏蔽快中断不会造成快中断请求的丢失。

5 参考文献

1. UCD3138 datasheet, Texas Instruments Inc., 2012
2. UCD3138 Digital Power Peripherals Programmer’s Manual, Texas Instruments Inc., 2012
3. UCD3138 ARM and Digital System Programmer’s Manual, Texas Instruments Inc., 2012
4. UCD3138 Monitoring and Communications Programmer’s Manual, Texas Instruments Inc., 2012
5. 双快中断触发源在数字电源 UCD3028 软件中的应用说明，Application Report, 2013

重要声明

德州仪器(TI) 及其下属子公司有权根据 JESD46 最新标准, 对所提供的产品和服务进行更正、修改、增强、改进或其它更改， 并有权根据 JESD48 最新标准中止提供任何产品和服务。客户在下订单前应获取最新的相关信息, 并验证这些信息是否完整且是最新的。所有产品的销售都遵循在订单确认时所提供的TI 销售条款与条件。

TI 保证其所销售的组件的性能符合产品销售时 TI 半导体产品销售条件与条款的适用规范。仅在 TI 保证的范围内，且 TI 认为 有必要时才会使用测试或其它质量控制技术。除非适用法律做出了硬性规定，否则没有必要对每种组件的所有参数进行测试。

TI 对应用帮助或客户产品设计不承担任何义务。客户应对其使用 TI 组件的产品和应用自行负责。为尽量减小与客户产品和应 用相关的风险，客户应提供充分的设计与操作安全措施。

TI 不对任何 TI 专利权、版权、屏蔽作品权或其它与使用了 TI 组件或服务的组合设备、机器或流程相关的 TI 知识产权中授予 的直接或隐含权限作出任何保证或解释。TI 所发布的与第三方产品或服务有关的信息，不能构成从 TI 获得使用这些产品或服 务的许可、授权、或认可。使用此类信息可能需要获得第三方的专利权或其它知识产权方面的许可，或是 TI 的专利权或其它 知识产权方面的许可。

对于 TI 的产品手册或数据表中 TI 信息的重要部分，仅在没有对内容进行任何篡改且带有相关授权、条件、限制和声明的情况 下才允许进行复制。TI 对此类篡改过的文件不承担任何责任或义务。复制第三方的信息可能需要服从额外的限制条件。

在转售 TI 组件或服务时，如果对该组件或服务参数的陈述与 TI 标明的参数相比存在差异或虚假成分，则会失去相关 TI 组件 或服务的所有明示或暗示授权，且这是不正当的、欺诈性商业行为。TI 对任何此类虚假陈述均不承担任何责任或义务。

客户认可并同意，尽管任何应用相关信息或支持仍可能由 TI 提供，但他们将独力负责满足与其产品及在其应用中使用 TI 产品 相关的所有法律、法规和安全相关要求。客户声明并同意，他们具备制定与实施安全措施所需的全部专业技术和知识，可预见 故障的危险后果、监测故障及其后果、降低有可能造成人身伤害的故障的发生机率并采取适当的补救措施。客户将全额赔偿因 在此类安全关键应用中使用任何 TI 组件而对 TI 及其代理造成的任何损失。

在某些场合中，为了推进安全相关应用有可能对 TI 组件进行特别的促销。TI 的目标是利用此类组件帮助客户设计和创立其特 有的可满足适用的功能安全性标准和要求的终端产品解决方案。尽管如此，此类组件仍然服从这些条款。

TI 组件未获得用于 FDA Class III（或类似的生命攸关医疗设备）的授权许可，除非各方授权官员已经达成了专门管控此类使 用的特别协议。

只有那些 TI 特别注明属于军用等级或“增强型塑料”的 TI 组件才是设计或专门用于军事/航空应用或环境的。购买者认可并同 意，对并非指定面向军事或航空航天用途的 TI 组件进行军事或航空航天方面的应用，其风险由客户单独承担，并且由客户独 力负责满足与此类使用相关的所有法律和法规要求。

TI 已明确指定符合 ISO/TS16949 要求的产品，这些产品主要用于汽车。在任何情况下，因使用非指定产品而无法达到 ISO/TS16949 要求，TI不承担任何责任。

	产品		应用
数字音频	www.ti.com.cn/audio	通信与电信	www.ti.com.cn/telecom
放大器和线性器件	www.ti.com.cn/amplifiers	计算机及周边	www.ti.com.cn/computer
数据转换器	www.ti.com.cn/dataconverters	消费电子	www.ti.com/consumer-apps
DLP® 产品	www.dlp.com	能源	www.ti.com/energy
DSP - 数字信号处理器	www.ti.com.cn/dsp	工业应用	www.ti.com.cn/industrial
时钟和计时器	www.ti.com.cn/clockandtimers	医疗电子	www.ti.com.cn/medical
接口	www.ti.com.cn/interface	安防应用	www.ti.com.cn/security
逻辑	www.ti.com.cn/logic	汽车电子	www.ti.com.cn/automotive
电源管理	www.ti.com.cn/power	视频和影像	www.ti.com.cn/video
微控制器 (MCU)	www.ti.com.cn/microcontrollers		
RFID 系统	www.ti.com.cn/rfidsys		
OMAP应用处理器	www.ti.com/omap		
无线连通性	www.ti.com.cn/wirelessconnectivity	德州仪器在线技术支持社区	www.deyisupport.com

邮寄地址： 上海市浦东新区世纪大道1568 号，中建大厦32 楼邮政编码： 200122