如何防止 UCD3138 无法再次烧录

Jack Tan, Frank Tang **High Voltage Power Solutions**

摘要

校验和是一种对空间与时间传送数据的完整性进行检查的一种简单的检验方法。但这种方式并不能检测数据的正确性。UCD3138 的出厂 Boot 也采用校验和的方式，在 Flash 程序执行这前，会对其完整性做一次检验。如果由于用户编写的程序本身有问题，则可能导致芯片无法再次烧录。本文主要介绍一种解决芯片无法再次烧录的方法，分析了芯片锁死的原因，并提供了相应的代码修改列表。

1 引言

在调试 UCD3138 的代码时，一般都会有用户会碰到该芯片无法再烧录。本文主要介绍如何在软件中设置来避免该问题。

2 芯片无法再次烧录的原因分析

如图 1 所示，为 UCD3138 上电后，Boot ROM 代码的流程。Boot ROM 的主要功能包含芯片的初始化，检验 Flash 的完整，以及 pmbus 的 boot loader。

Boot ROM 支持检验 Pflash 中的两个校验和：一个是 boot flash 中的校验和（0x7fc –0x7ff）；另一个是整个 flash 中的校验和（0x0000 – 0x7fff）。

如果任意一个检验和合法，那么 ROM 将把 flash 的地址在重映射至 0x0000 0000，同时跳转到 Flash 当中执行用户设定的代码。32KB 的代码检验时间大概需要 10ms 才能完成。检验和其实就是一个简单的加法运算，把存储器中所有的字节都按最一个无符号的 8 位数相加。在核对检验和之前，Boot ROM 首先会去验证在 Flash 的起始位置是否有 0xEA 代码，如果没有 0xEA，Boot ROM 不会执行核对校验和的程序，而是直接跳转到 ROM 中 PMBus 通信代码，这样可以有效的避免当存储器的值都为 0 时，出现芯片锁死的情况。

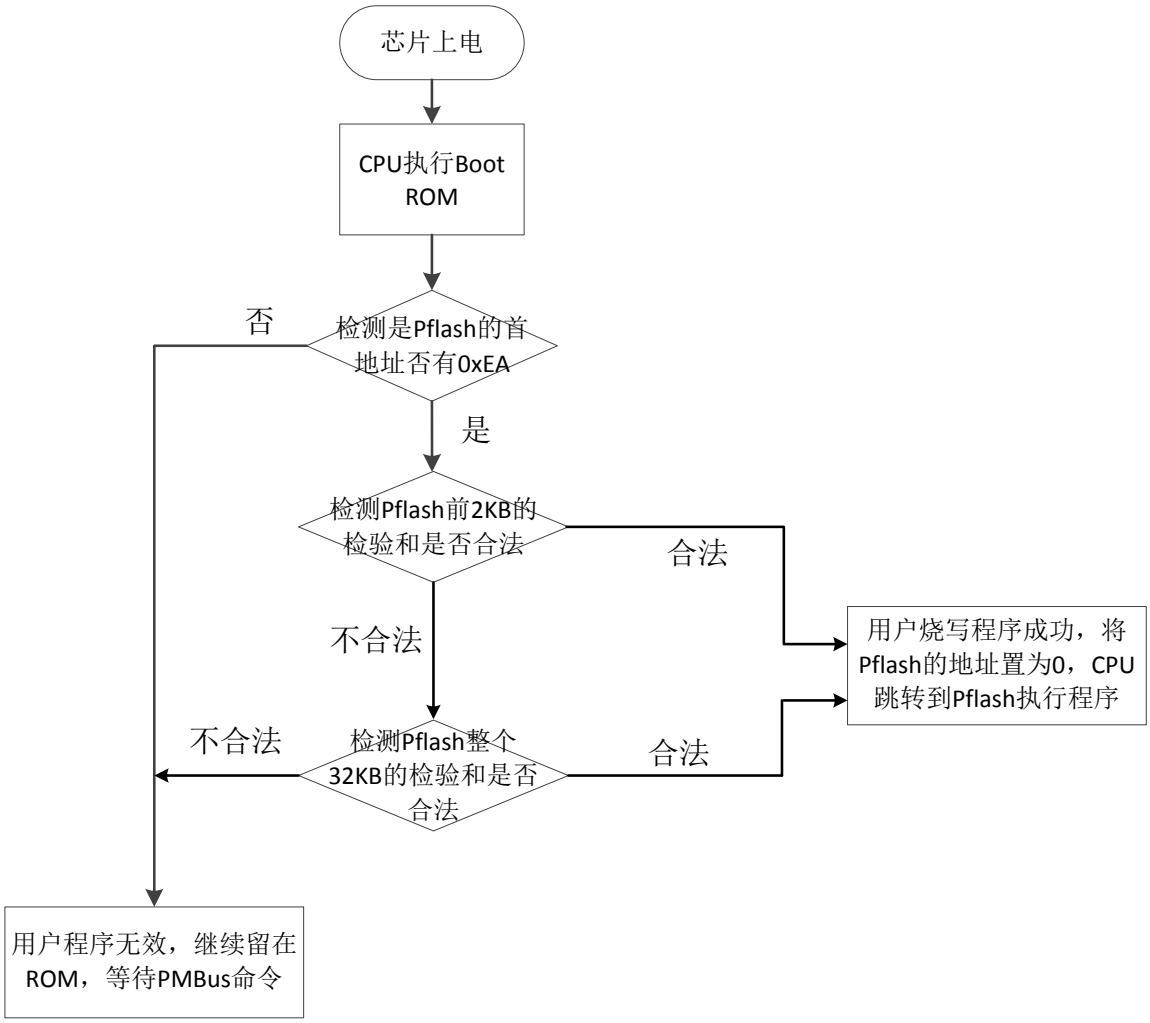

图 1. UCD3138 上电后执行程序的流程

从上面分析可以看到，如果 program flash 的 checksum 已被烧写（这里通常假设 checksum 是正确），那么上电后处理器都会去执行用户的代码。假设这时用户的代码有问题，导致 checksum 不能被擦除，这样就没有办法回到 ROM 模式，所以就会出现芯片无法再次烧录。造成 checksum 不能擦除的原因有多种可能，其中最常见的有：1. 中断没有清除标志，导致 Background loop 的代码没有机会执行，这种情况下，pmbus 是无法通信；2. Pmbus 可以正常通信，但是擦除 checksum 的代码有问题。

3 正确添加擦除 checksum 代码

擦除 checksum 其实就是将 Program flash 中原有的 checksum 用 0 来替代， 从而 program flash 的 checksum 是一个不正确的值。这样芯片复位后，就会回到 ROM 模式。

当 UCD3138 运行 program flash 的代码时，不能够再向 program flash 写数据。所以，擦除 checksum 的代码在执行之前应拷贝到 RAM 中，然后将 PC 指针跳转到 RAM，即将擦除 checksum 的代码在 RAM 中运行。下面提供一种正确擦除 checksum 的方法：

1. 打开 cyclone.cmd，在 Memory 中添加一个新的 RAM，这个区域用于执行擦除 checksum 的代码。如果是其它型号的 UCD 器件，可以查看相应的编程手册或数据手册获取相应的 RAM 地址与长度。

```
MEMORY
{
… …
/*-------------------------------------------------*/
/* RAM       4K    0x19000 - 0x19FFF               */
/*-------------------------------------------------*/
RAM_FOR_PROGRAM_AREA (RW) : org = 0x00019000, len = 0x00000080
RAM    (RW) : org = 0x00019080, len = 0x00000F7F
… …
}
```

2. 在 sections 的区域，添加一个 section，指定擦出 checksum 代码存储的位置与运行的位置。

```
SECTIONS
{
      … …

      UNION : run = RAM_FOR_PROGRAM_AREA
      {
      .ram_for_program_area
      .zero_out_integrity_word      :      load      =      PFLASH,
      start(_zero_out_integrity_word_start)
      {zero_out_integrity_word.obj }
      }

}
```

   - run = RAM_FOR_PROGRAM_AREA：该区域的代码的运行区域在 RAM_FOR_PROGRAM_AREA
   - .zero_out_integrity_word：是输出的 section
   - Load = PFLASH：指定代码的装载位置在 Program Flash
   - start(_zero_out_integrity_word_start)：装载代码的区域的启始地址(symbol)是_zero_out_integrity_word_out_start

- {zero_out_integrity_word.obj}: 装载在 Pflash 的目标文件

3. 新建一个 C 源文件，定义为 zero_out_integrity_word.c, 其具体的代码如下。(注意：c 源文件的名称应该与第二项的 obj 文件名称相同)。这部分代码的功能就是擦除 checksum。

```
#include "system_defines.h"
#include "cyclone_device.h"
#include "pmbus_commands.h"
#include "variables.h"
#include "software_interrupts.h"
#define program_flash_integrity_word (*((volatile unsigned long *)
0x7ffc))
//last word in flash, when executing from Flash.  used to store
integrity code

void zero_out_integrity_word(void)
{

    DecRegs.FLASHILOCK.all = 0x42DC157E;// Write key to Program
Flash Interlock Register
    DecRegs.MFBALR1.all = MFBALRX_BYTE0_BLOCK_SIZE_32K; //enable
program flash write
    program_flash_integrity_word = 0;
    DecRegs.MFBALR1.all = MFBALRX_BYTE0_BLOCK_SIZE_32K + //expand
program flash out to 4x real size
                          MFBALRX_BYTE0_RONLY;
    while(DecRegs.PFLASHCTRL.bit.BUSY != 0)
    {
            ; //do nothing while it programs
    }
   SysRegs.SYSECR.bit.RESET = 2;   //now reset processor.
    return;
}
```

4. 指定擦除 checksum 在软件中断的入口参数。这里使用 12。

```
void clear_integrity_word(Uint32 block)
{
      swi_single_entry(block,0,0,12);
}
```

5. 打开 software_interrupt.c（有些工程是 interrupts.c）,在 case 12 的添加如下代码，该部分代码主要是先将擦出 checksum 的代码从 program flash 拷贝到 ram，然后在 ram 中运行。

```
case 12: // clear integrity word.
{
```

```
    register Uint32 * program_index = (Uint32 *) program_area;
//store destination address for program
    register    Uint32    *    source_index    =    (Uint32    *)
zero_out_integrity_word_start;  //Used  for  source  address  of
PFLASH;
    register Uint32 counter;

    for(counter=0; counter < 32; counter++) //Copy program from
PFLASH to RAM
      {
            *(program_index++)=*(source_index++);
      }

      zero_out_integrity_word();
  }
```

6. 在软中断服务程序的所在源文件中的起始位置，申明如下变量与函数名称，并且定义一个存储在.ram_for_program_area 代码区的数组变量。

```
extern Uint32 zero_out_integrity_word_start[32];
extern void zero_out_integrity_word(void);

#pragma DATA_SECTION(program_area, ".ram_for_program_area")
Uint32 program_area[32]; // small RAM area for storing programs
which modify program flashes
```

7. 在代码调试阶段，为了避免芯片出现无法再次烧录，可以在 main 函数的起始位置让 clear_integrity_word 在某个条件下执行，比如，某个 IO 为高电平或低电平时才运行。如下是一个代码示例，当出现芯片被锁死时，可以将 FAULT3 引脚接地，然后，复位芯片或掉电重启即可清除 checksum 并回到 ROM 模式。

```
if(GioRegs.FAULTIN.bit.FLT3_IN == 0)// Re-Check pin assignment
{
      clear_integrity_word();
}
```

8. 另外，TI 的 demo 也支持 pmbus 命令可以让芯片回到 rom 模式，打开 pmbus_write_message()函数。

```
int32 pmbus_write_message(void)
{
      switch (pmbus_buffer[0])
      {
      … …
            case PMBUS_CMD_ROM_MODE:
```

```
                return pmbus_write_rom_mode();

        … …

        }

}

int pmbus_write_rom_mode(void)
{
        // Call a SWI to clear the integrity word.
        clear_integrity_word();
        return PMBUS_SUCCESS;                // Note: This line is never
reached.
}
```

9. 如果使用 CCS3.3，还得将 Linker order 中的 cmd 文件清除。右击 project -> Build options -> linker order

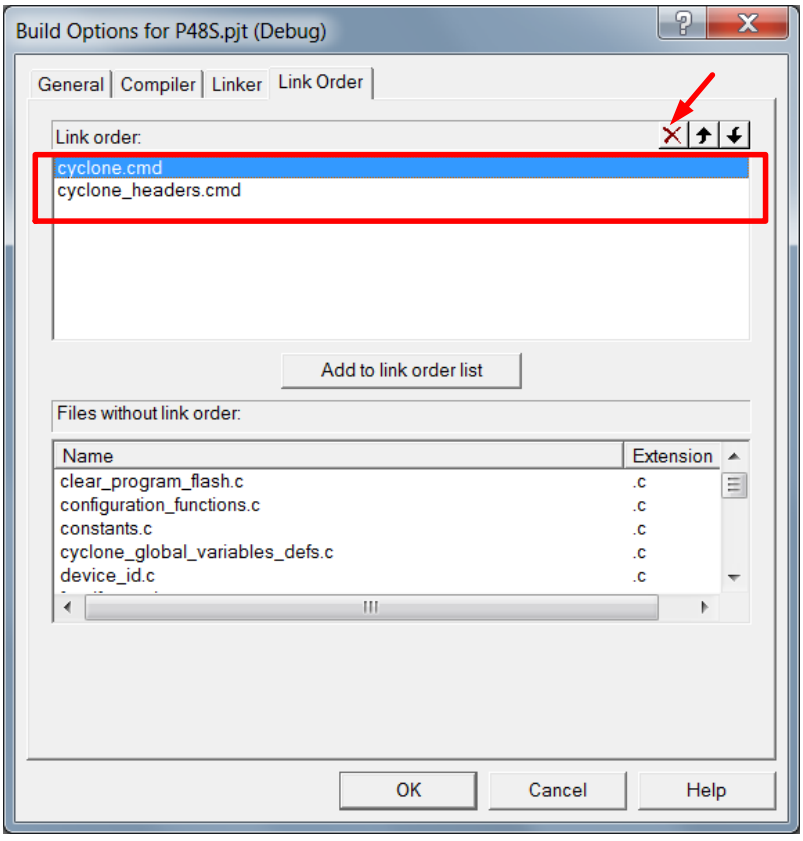

图 2 工程文件在 CCS3.3 中的配置

4 结论

按照第 3 部分修改代码后，将可执行文件以带 checksum 的形式烧录到芯片中，这种状况下芯片重新上电后，Program Flash 中的程序即可被执行。如果 PMBus 能正常通信，可以通过 GUI，发送 PMBus 命令：0Xd9，使得芯片回到 ROM 模式；如果 PMBus 不能正常通信，这时可以将 Fault 3 引脚接低电平，再复位芯片，这样也可以回到 ROM 模式。成功回到 ROM 模式后，你可以查看 Program Flash 的 checksum 值。UCD3138 的 ROM 模式下，Program Flash 的地址为：0x10000 – 0x17FFF。其它 UCD3138 的家族产品，Program Flash 在 ROM 模式下的地址会有所不一样，可以查看相应的 Program Manual 和 Datasheet 获取相应的地址信息。Checksum 都是存放在 Program Flash 的最后 4 个字节。

下图是芯片回到 ROM 模式后，使用 GUI 查看 Program Flash 的 checksum 的值。打开 Device GUI，在 Debug 的分组下，点击 Memory Peek/Poke，就会出现如图 3 所示的对话框。然后在相应的地址栏输入 0x17FFC，数据类型为 Uint32，点击 Read，就可以查看该地址的数据。如下图可以看出，从 Flash 模式回到 ROM 模式后，0x17FFC 地址对应的值为 0，即 Program Flash 的 checksum 值被覆盖。

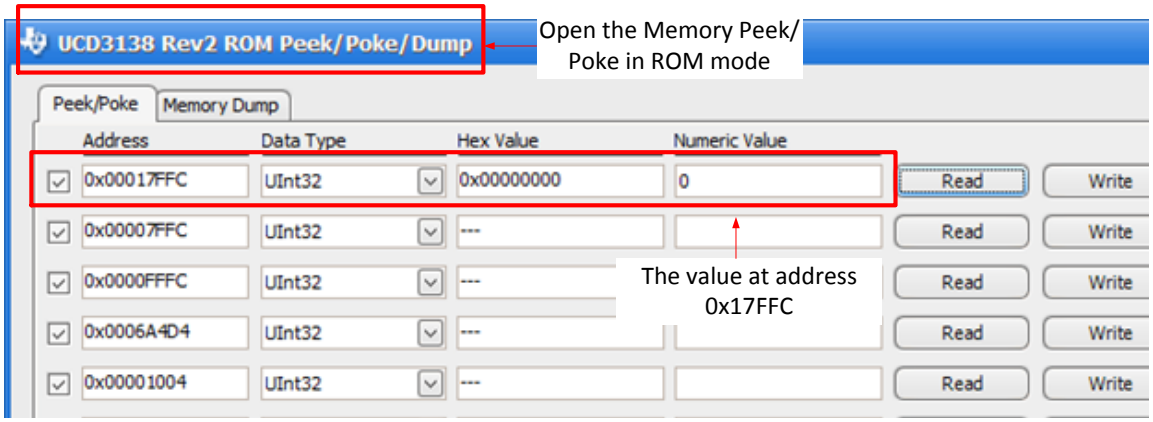

图 3 通过 GUI 查看 Program Flash 的 Checksum 的值

5 参考文献

1. UCD3138 ARM and System Programmer's Manual sluu994, Texas Instruments, 2013
2. UCD3138 Digital Power Peripherals Programmer's Manual SLUU995, Texas Instruments, 2013
3. Fusion Digital Power Designer GUI for Isolated Power Applications SLUA676, Texas Instruments, 2014

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

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265