基于 GUI 软件配置 UCD3138 数字电源 PMBUS 命令

Neil Li, Sundy Xu *China Telecom Application Team*

摘　要

可与数字电源 UCD3138 配套使用的 Fusion Digital Power Designer 软件拥有 Graphical User Interface (GUI) 界面，用户可在其上编辑数据并通过对应的 PMBUS 命令与 UCD3138 的软件交互。在 GUI 界面中，用户可以灵活的增加 GUI 软件支持的 PMBUS 命令，亦可以删除，因此大幅提高了 GUI 的灵活性。本文通过两个实例，详细分析了如何增加和删除 GUI 软件可以支持的 PMBUS 命令。

目　录

图

1 数字电源 GUI 软件及其配置功能

数字电源 GUI 软件运行于用户计算机，可以借助 PMBUS 总线与 UCD3138 数字电源通信，完成配置、设计和监控等功能。用户可以灵活的在界面中添加 PMBUS 命令支持的信息，亦可以在其中删除。

1.1 数字电源 GUI 软件

图 1 所示的是与 UCD3138 数字电源芯片配套使用，可用来对基于 UCD3138 数字电源进行配置，设计及监控的 GUI 软件：Fusion Digital Power。该软件安装并运行于用户的计算机上，通过 PMBUS 总线与 UCD3138 的软件进行交互。

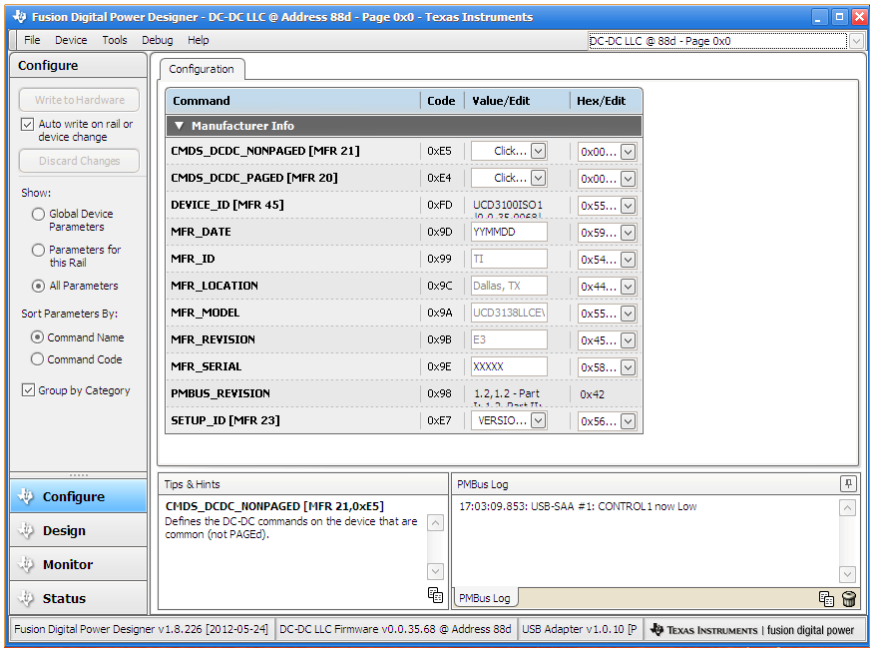

图 1：数字电源 GUI 软件

该软件主要包含以下功能：

1）配置：通过 PMBUS 命令对数字电源的参数进行配置，如输入电压的欠压保护点（对应的 PMBUS 命令为 VIN_ON 和 VIN_OFF），输出电流的过流保护（对应的 PMBUS 命令为 IOUT_OC_FAULT_LIMIT）。

2）设计：主要是对数字电源的环路参数进行设计和模拟。

3）监控：可以实时监控输入电压，输入电流和温度等诸多信息。

4）状态：可以显示电源板输入和输出等状态，告知用户当前是否存在故障。

1.2 GUI 软件的配置功能

图 1 所示的是 GUI 的配置界面，其显示的每一条信息都对应一条 PMBUS 命令，可以在用户计算机与 UCD3138 之间传递。例如制造商的位置信息（MFR_LOCATION），对应的便是 PMBUS 命令

MFR_LOCATION （0x9C），借助 PMBUS 总线 GUI 软件可以将 UCD3138 中存贮的信息读取并显示出来。用户也可以自行重新编辑该信息，新信息会传递到 UCD3138 芯片中并进行存储。

在实际应用中，不同的用户会关注不同的参数信息。为提高灵活性，GUI 软件支持用户添加和删除 PMBUS 命令支持的参数信息。下面将详细介绍如何在 GUI 中进行相关操作。

2 增加输出过流保护点信息

输出过流保护点对应的 PMBUS 命令为 IOUT_OC_FAULT_LIMIT，可以用来配置系统的输出过流保护点。本节详细介绍如何在 GUI 中添加信息栏来接收用户的输入，同时修改 UCD3138 的软件来对用户的输入信息进行处理并最终调整相应模拟比较器的阈值电压。

2.1 解除对 PMBUS 命令的屏蔽

在 UCD3138 软件中，定义了 CMD_DCDC_NONPAGED 变量，保存了每一个 PMBUS 命令的状态：用 0 和 1 表征屏蔽还是未屏蔽，如下代码所示。而“输出过流保护点”对应的 PMBUS 命令是 IOUT_OC_FAULT_LIMIT （0x46），需要首先在 UCD3138 的软件中去掉对该命令的屏蔽。

```
#define CMD_DCDC_NONPAGED  \
                    {0x00, 0x00, \
                     0x00, 0x00, \
                     0x00, 0x00, \
                     0x00, 0x00, \
                     0x02, 0x00, \
                     0x00, 0x00, \
                     0x00, 0x00, \
                     0x00, 0x00, \
                     0x00, 0x00, \
                     0x00, 0xFE, \
                     0x00, 0x00, \
                     0x00, 0x00, \
                     0x00, 0x00, \
                     0x00, 0x40, \
                     0x3D, 0x00, \
                     0x00, 0x14 \
                     }
```

Fusion Digital Designer 提供了 Bitmask tool，用来快速生成新的 CMD_DCDC_NONPAGED 变量。具体操作如下。

1） 如图 2，在菜单“Tools”中点击 Isolated GUI Bitmask Generator Tool；

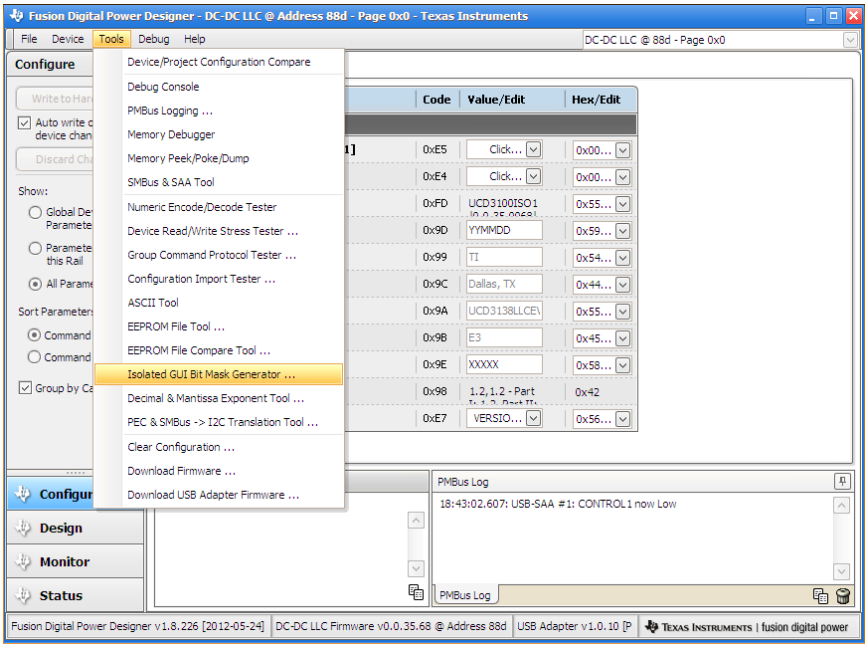

图 2：运行 Isolated GUI Bitmask Generator

2） 如图 3，在随后打开的界面中，勾选“PMBUS_CMD_IOUT_OC_FAULT_LIMIT”，即使能命令 IOUT_OC_FAULT_LIMIT。此时界面右侧的数据会有变化，该 PMBUS 对应的位由 0 变为了 1。

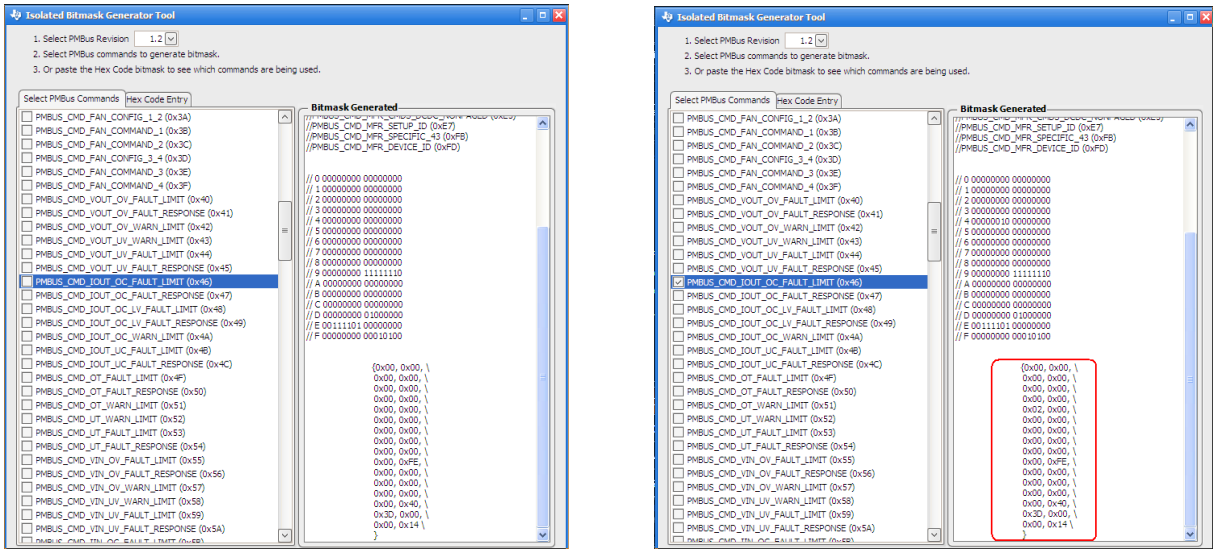

图 3：勾选相应 PMBUS 命令并复制输数据

3） 将图 3 右侧红色图内的数据复制到 UCD3138 的软件中，覆盖原来的 CMD_DCDC_NONPAGED 变量。

2.2 GUI 中增加新的信息栏

将上述新生成的软件编译后烧录到 UCD3138 中，此时通过 Fusion Digital Designer 与 UCD3138 建立连接后，会发现，GUI 中新增了一条信息输入栏，见图 4 中的红色框。

该信息栏的名称为“IOUT_OC_FAULT_LIMIT”，代码为 0x46，上述为固定信息，是由 GUI 软件自身设定，用户无法修改。后面的“Value/Edit”值则用来输入用户设定的过流保护点。

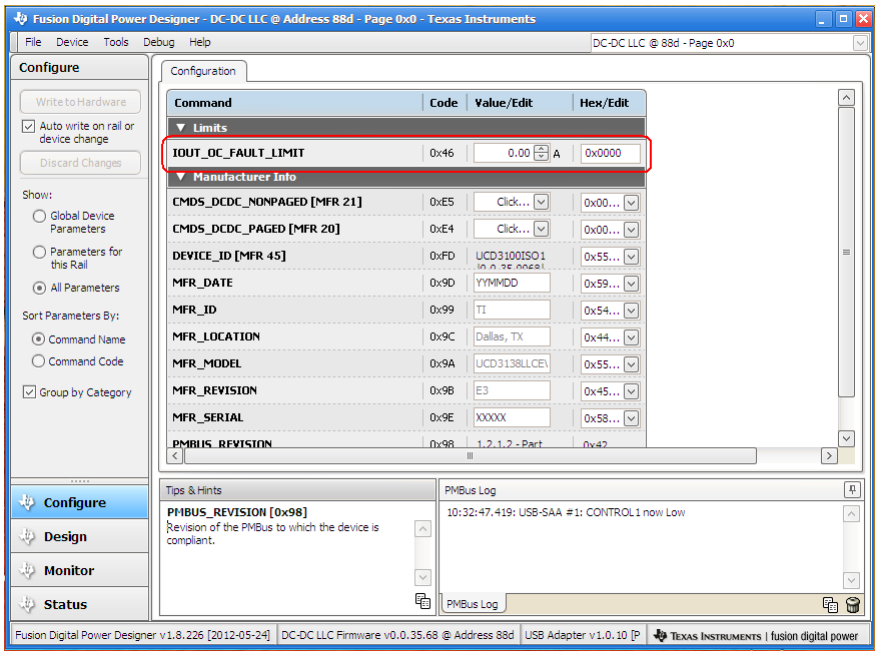

图 4：GUI 出现新增信息输入栏

2.3 UCD3138 软件中的数据处理

截止到上节，用户可以在 GUI 中新增一条信息栏并输入数据，然后通过对应的 PMBUS 命令，将该数据传输到 UCD3138 中。但为了使 UCD3138 接收该数据并调整相应的模拟比较器阈值，还需要修改 UCD3138 的软件。

1、定义变量和结构体

如下代码，定义了新的结构体变量 PMBUS_DCDC_CONFIG，包含成员 iout_oc_fault_limit 和 reserved。随后定义了该结构体类型的外部变量 pmbus_dcdc_config 和 pmbus_dcdc_config_translated，分别用于接收和保存 GUI 传输到 UCD3138 的数据和格式转换后的数据。

```
typedef struct
{
 Uint16 iout_oc_fault_limit;
 Uint16 reserved;
}PMBUS_DCDC_CONFIG; //must be even number of int16
```

```
EXTERN PMBUS_DCDC_CONFIG        pmbus_dcdc_config[1];
EXTERN PMBUS_DCDC_CONFIG        pmbus_dcdc_config_translated[1];
```

2、创建新的 PMBUS 读与写函数

由于从 GUI 传输过来的数据的首个字节是对应的 PMBUS 命令的代码，如 0x46，并会存放在 pmbus_buffer[0]中。因此，可以在函数 pmbus_write_message()中创建新的 case 语句，并返回一个新创建的函数 pmbus_write_iout_oc_fault_limit()，该新函数用来处理接收到的数据。这样就实现了一旦 UCD3138 接收到新的数据后，且该数据是用户重新编辑的“输出过流保护点”信息，则就会调用 pmbus_write_iout_oc_fault_limit()函数进行处理。

```
switch (pmbus_buffer[0])
  {
    case 0x46:
    return pmbus_write_iout_oc_fault_limit();
  }
```

同时，也需要在函数 pmbus_read_message()中创建新的 case 语句，并返回一个新创建的函数 pmbus_read_iout_oc_fault_limit()，用来返回接收到的信息到 GUI 中，以确保信息传输正确。这种读取后再返回验证的操作是 PMBUS 协议的规定。

```
switch (pmbus_buffer[0])
  {
    case 0x46:
    return pmbus_read_iout_oc_fault_limit();
  }
```

3、数据处理函数的设计

◎ 接收数据处理函数 pmbus_write_iout_oc_fault_limit()

该函数用来接收来自 GUI 的数据，并将该数据的格式由 Linear Data Format 转换为浮点型数据，最后强制转换为整数型赋给模拟比较器。关键代码分析如下：

上文提到，来自 GUI 的数据的首字节是对应的 PMBUS 命令代码。随后的两个字节便是 Linear Data 格式的数据。将该数据保存在 pmbus_dcdc_config 结构体的 iout_oc_fault_limit 成员中，如下代码所示。

```
Pmbus_dcdc_config[0].iout_oc_fault_limit = pmbus_buffer[1] + (pmbus_buffer[2] << 8);
```

下面代码是调用格式转换函数 linear11_to_float()，将上面接收到的数据转换为浮点型数据。

```
local_variable = linear11_to_float(pmbus_dcdc_config[0].iout_oc_fault_limit);
```

由于转换后的浮点型数据与最终需要赋给模拟比较器阈值的数据存在一定的比例，需要一个转换系数（scaler）。缩放后存放在 pmbus_dcdc_config_tanslated 结构体的 iout_oc_fault_limit 成员中。

```
pmbus_dcdc_config_translated[0].iout_oc_fault_limit = (int)(local_variable*2.54);
```

最终该值赋给模拟比较器的阈值，用来做快速保护。

FaultMuxRegs.ACOMPCTRL0.bit.ACOMP_B_THRESH = pmbus_dcdc_config_translated[0].iout_oc_fault_limit;

◎ 返回数据处理函数 pmbus_read_iout_oc_fault_limit

该函数用来返回 UCD3138 软件接收的数据到 GUI 中，以使 GUI 将写入和读取的数据做比较（比较判断等操作在 GUI 软件中完成），保证数据正确。该函数调用 pmbus_read_two_byte_handler()将保存在 Pmbus_dcdc_config[0].iout_oc_fault_limit 的数据返回到 GUI。关键代码如下：

pmbus_read_two_byte_handler(pmbus_dcdc_config[0].iout_oc_fault_limit);

而在 pmbus_read_two_byte_handler()函数中的关键代码为：

pmbus_buffer[1] = value >> 8;

pmbus_buffer[0] = value & 0xff;

pmbus_buffer 字节中的数据会最终上传到 GUI 软件中。

4、数据转换函数的设计

◎ Linear Data Format 数据格式

上文提到，来自 GUI 的数据遵循 PMBUS 协议，其格式为 Linear Data Format。如图 5，其低 11 位为“尾数”，以补码形式保存；高 5 位为“指数”，亦是以补码形式保存。该数据与实际数据的关系为：$X = Y \times 2^N$。

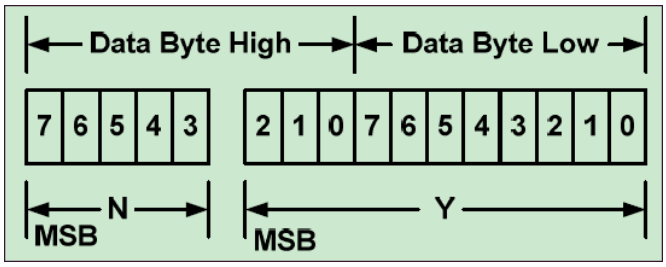

图 5：PMBUS 协议中的 Linear Data Format

◎ ARM 编译器中的浮点型数据

如图 6，为 ARM 编译器中对单精度浮点型数据的存储格式。其最高位为符号位，接下的 8 位为指数，后面的 23 位为尾数。在编译器中定义的浮点数据，将以该格式存储在硬件存储空间。

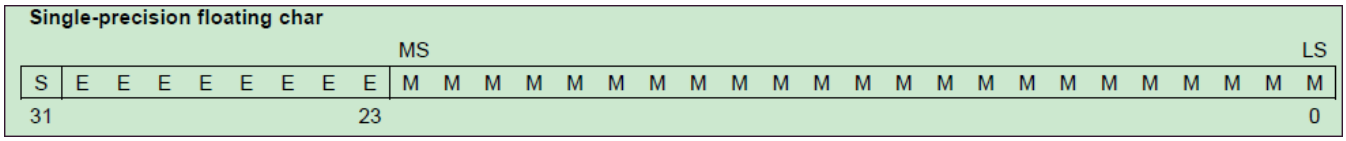

图 6：ARM 编译器中的浮点型数据格式

◎ 定义结构体和联合体

如下，定义了结构体 FLOAT_ELEMENTS，用以保存上文提到的浮点型数据。还定义了联合体 FLOAT_OPEN，成员包括浮点型数据“all”和结构体类型数据“bit”，用以保存转换完毕的浮点型数据。

```
struct FLOAT_ELEMENTS {
 Uint32 SIGN:1;
 Uint32 EXPONENT:8;
 Uint32 MANTISSA:23;
};

union FLOAT_OPEN {
 float                    all;
 struct FLOAT_ELEMENTS    bit;
 Uint32                   word;
};
```

◎ 转换函数 linear11_to_float()

该函数完成将来自 GUI 的 Linear Data Format 格式的数据转换为浮点型数据，并作为返回值返回。包含的关键代码如下。

定义变量，包括整型“mantissa”和“exponent”及结构体变量“final”。

```
int16 mantissa, exponent;
union FLOAT_OPEN final;
```

下面代码完成对输入参数的分析，并扩展到 16 位。如果输入参数的尾数是负值，前 5 位补 1；如果是正值，前 5 位补 0。

```
if(linear11 & 0x0400)// if mantissa is negative
{  mantissa = linear11 | 0xfc00; //put it in there at minimum mantissa  }
else
{  mantissa = linear11 & 0x07FF;  }
```

下面代码首先是将 mantissa 左移 16 位（16+16=32），以使其数据长度符合单精度浮点型数据的长度，然后 exponent 减去 16，保证了原始数据的大小没有变化。

```
final.all = ((int32)mantissa) << 16; //set it up for minimum exponent.
exponent = (linear11 >> 11) - 16; //get exponent to match shifted value
```

下面代码首先判断单精度浮点型的指数是否为负。如果是负，则改写其为零，原因是对应的“输出过流保护点”不会出现负值；如果是正，则将指数信息放置在单精度浮点型数据的指数位置。

```
if((final.bit.EXPONENT + exponent) < 0)//if it's so low it will wrap
{   final.bit.EXPONENT = 0;  }
```

```
else
{   final.bit.EXPONENT = final.bit.EXPONENT + exponent;  }
```

上述操作完毕后，final.all 中就保存了转换后的浮点型数据，因此可以作为返回值返回。

```
return final.all;
```

2.4 操作流程图

上述所有操作的流程图见图 7。

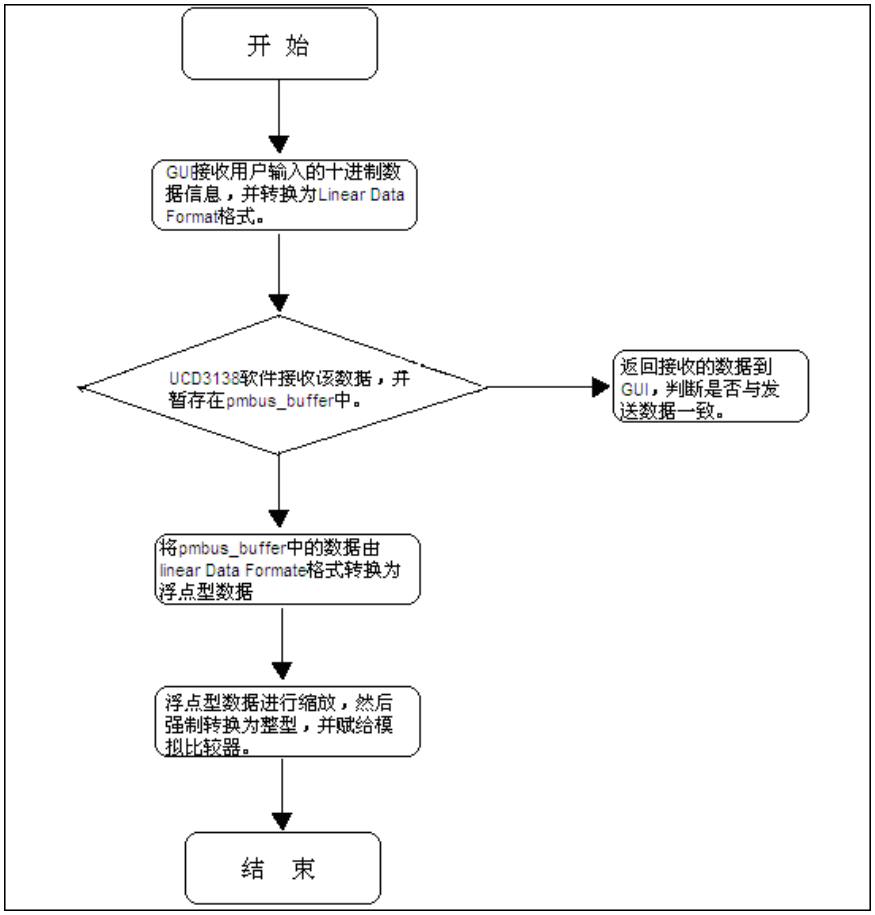

图 7：UCD3138 接收信息流程图

3 删除 GUI 信息栏

用户亦可以自行删除 GUI 中多余的信息栏，操作过程主要包括对应 PMBUS 命令的屏蔽，UCD3138 软件对应接收处理函数的删除等。如图 8，以删除红色框内的信息栏为例。查看其对应的 PMBUS 命令代码是 0xFA，故首先在 CMD_DCDC_NONPAGED 变量中屏蔽对应的位，可以使用 Bit Mask Tool。

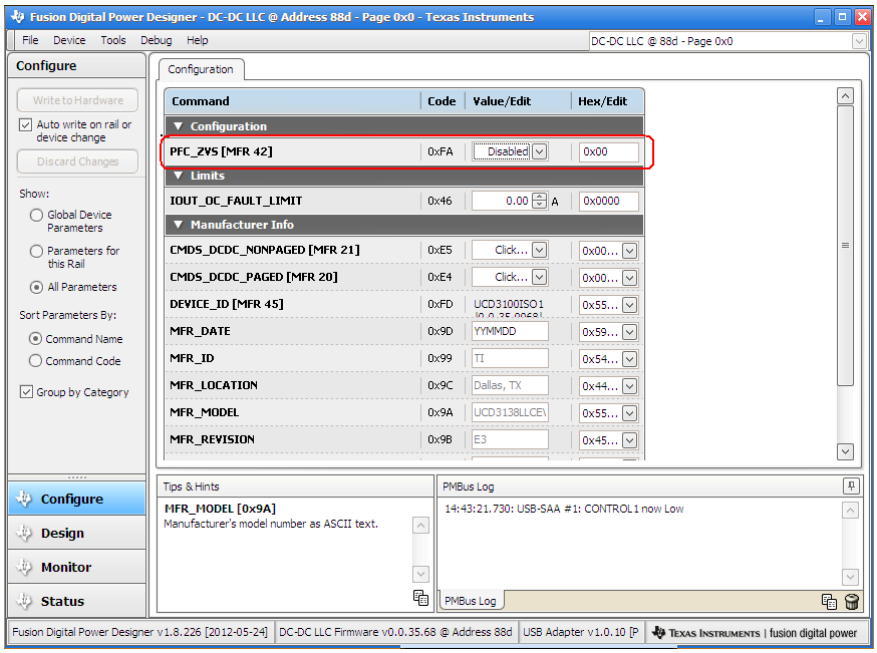

图 8：删除 GUI 中的信息栏

此时重新编译软件并下载到 UCD3138 后，GUI 中将不再会出现该信息栏，见图 4。但 UCD3138 软件中依然保留有对该 PMBUS 命令的接收、转换和处理等函数，亦需要删除，在此不再赘述。

4 小结

通过上文两个实例的分析可知，用户在 GUI 软件中可以灵活的添加或删除 PMBUS 命令对应的信息栏，提高了 GUI 的灵活性。该操作主要包括 PMBUS 命令的使能或屏蔽及 UCD3138 软件添加或删除相应处理函数等两大部分。

5 参考文献

1. UCD3138 datasheet, Texas Instruments Inc., 2011
2. PMBus_Specification_Part_I_Rev_1, 2007
3. PMBus_Specification_Part_II_Rev_1, 2010
4. ARM Optimizing C/C++ Compiler, v4.9, User's Guide, Texas Instruments, 2011

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
Copyright © 2013 德州仪器 半导体技术（上海）有限公司