UCD3xxx 数字电源控制器调试工具：内存调试器（Memory Debugger）的详细介绍

Sundy Xu, Neil Li *China Telecom Application Team*

摘要

边界扫描（JTAG）是一种常用的调试方法，大多数 MCU 都支持 JTAG 调试，比如 TI 数字信号处理器（DSP）、数字信号控制器（DSC）、以及基于 ARM 架构的处理器等。

JTAG 在电源调试应用中会有很大风险， 因为 JTAG 常常需要中止程序的正常运行去查看寄存器和变量。这里推荐的 UCD3xxx 主要调试方法是基于通用异步收发器（UART）和电源管理总线（PMBUS）。它们都支持在线调试功能。

UART 调试方法比较简单，本文不做论述，可以参考 UCD3xxx 示例代码。对于 PMBUS 调试方法，使用的主要工具就是内存调试器（Memory Debugger）。所以，本文主要详细介绍 Memory Debugger 的使用方法。

目录

图

1 引言

1.1 UCD3xxx 简介

UCD3xxx 系列芯片是 TI 公司开发的专门面向数字电源应用的控制器，主要应用在通讯电源、服务器电源、无线功放电源以及标准砖模块电源等领域。UCD30xx 系列产品主要是 UCD3028、UCD3020 以及 UCD3040。最新一代产品是 UCD31xx 系列，主要产品是 UCD3138、UCD3138064 以及 UCD3138128 等。

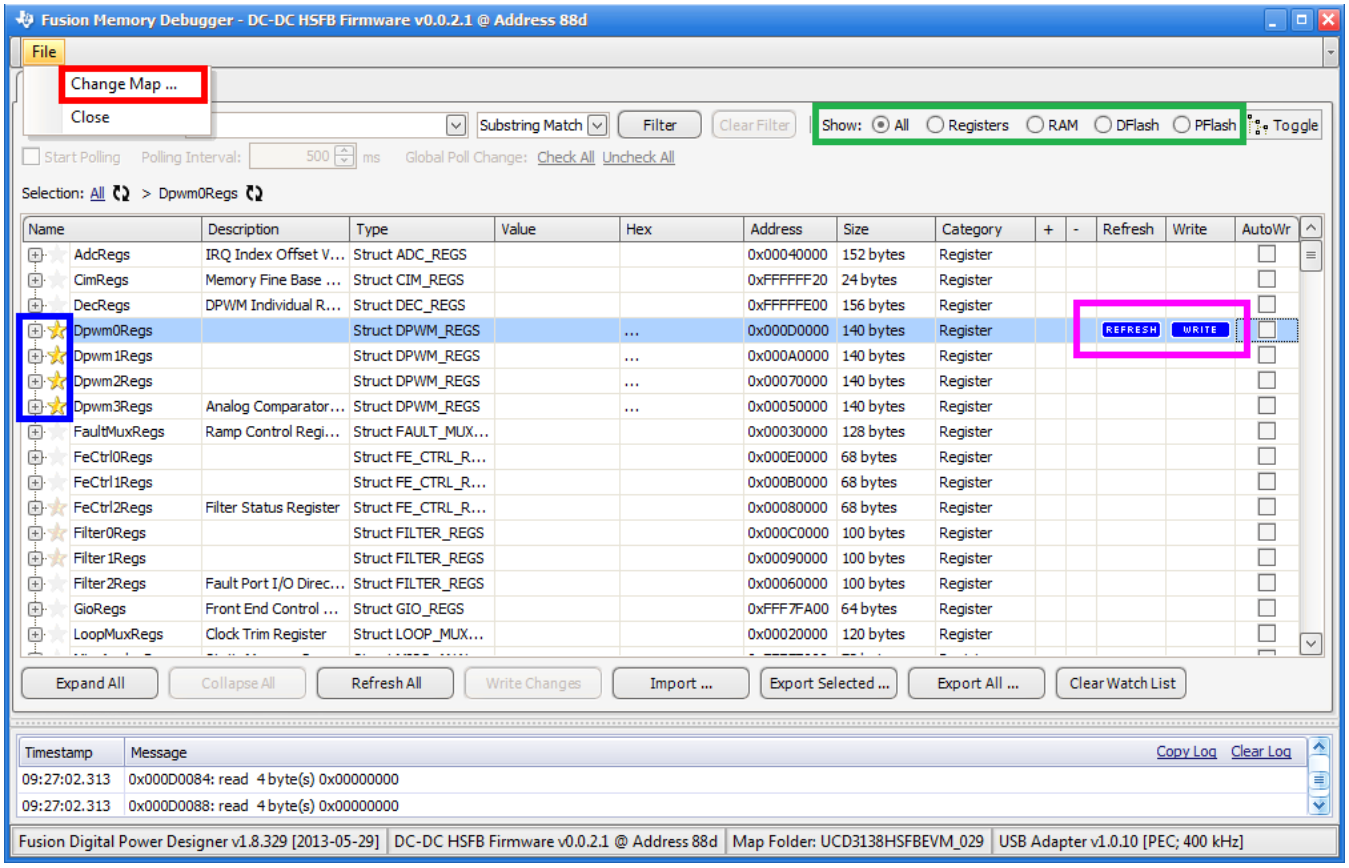

图1. Memory Debugger 界面

1.2 PMBUS 简介

PMBUS（电源管理总线）是一种开放标准的数字电源管理协议。可通过定义传输和物理接口以及命令语言来促进与电源转换器或其他设备的通信。该协议是由一群认为由于没有合适的标准而抑制了全数字电源管理解决方案发展的电源和半导体生产商共同建立的。目前使用标准是 PMBUS 1.2，PMBUS 1.3 正在制定中。

PMBUS 总共可以有 256 条命令，主要有几种：标准命令，是固定的不可改动的，如读取输出电压命令是 0x8B，那么 0x8B 就不能用做其它命令；保留命令，留作以后扩展使用；用户自定义命令，用户可以根据自己的应用来定义这些命令。具体命列表可以参考文献 1。

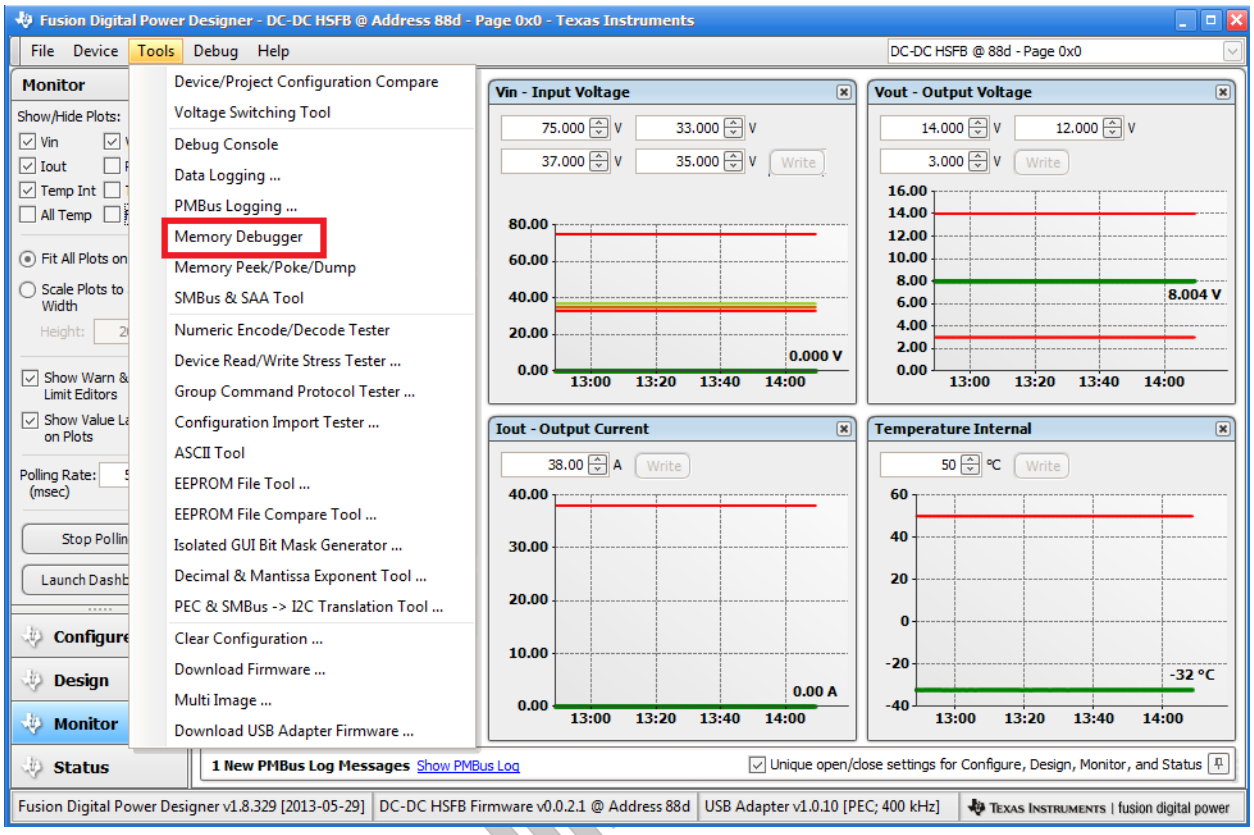

图2. Memory Debugger 在 Fusion Design Online 的位置

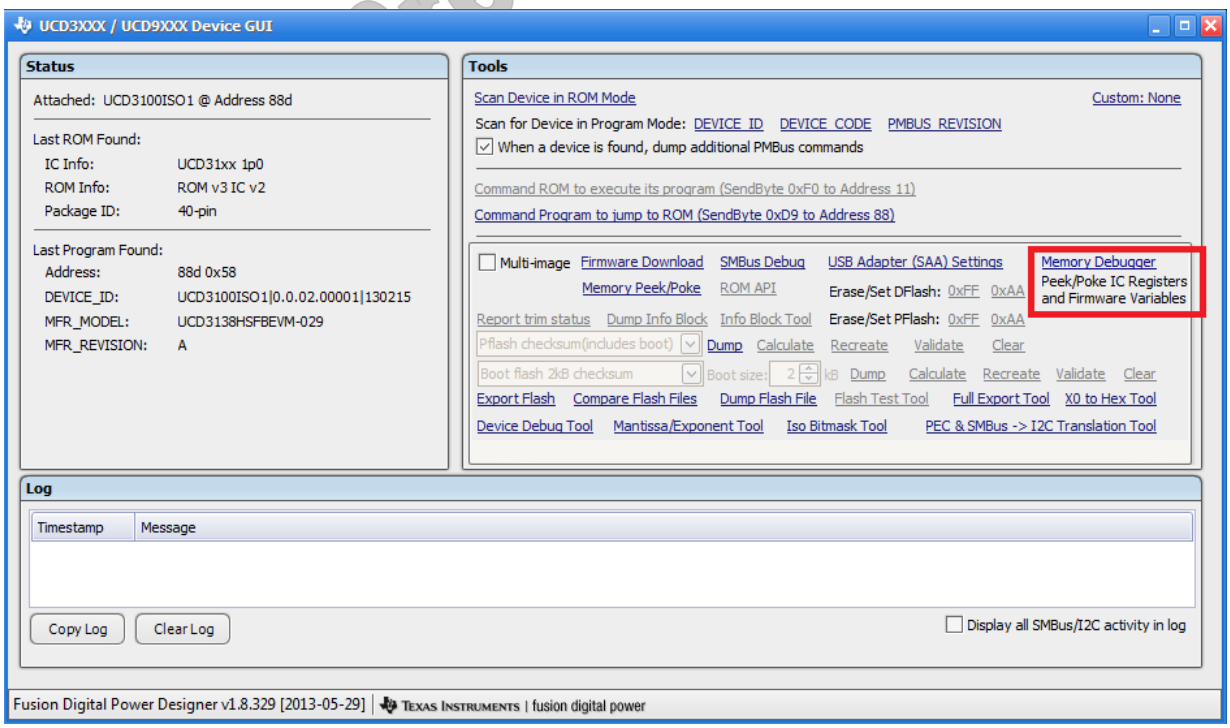

图3. Memory Debugger 在 UCD3xxx & UCD9xxx Device GUI 的位置

2 怎么找到 Memory Debugger

Memory Debugger 是德州仪器数字电源设计调试工具 Fusion Digital Power Designer（下载地址：http://www.ti.com/fusiongui，请下载最新版本，早期版本不支持 Memory Debugger 功能）的一部分，主要用来读写 UCD3xxx 内部全局变量和寄存器的值。Memory Debugger 界面如图 1 所示。如果是第一次调用 Memory Debugger，需要输入密码“forestln”。

那么如何调出 Memory Debugger 工具，有两条途径：一是执行 Fusion Design Online 可执行文件（桌面直接双击 Fusion Design Online 快捷方式，如果没有创建桌面快捷方式，在开始菜单中找到 Texas Instruments Fusion Digital Power Designer 目录下的 Fusion Digital Power Designer，点击即可），然后就可以在 Fusion Design Online 上的 Tools 菜单下面找到 Memory Debugger 项，如图 2 所示；二是执行 UCD3xxx & UCD9xxx Device GUI 可执行文件（桌面直接双击 UCD3xxx & UCD9xxx Device GUI 快捷方式，如果没有创建桌面快捷方式，在开始菜单中找到 Texas Instruments Fusion Digital Power Designer\Device GUIs 目录下的 UCD3xxx & UCD9xxx Device GUI，点击即可）,Memory Debugger 在图 3 红圈处。

3 如何产生 Memory Debugger 需要的.PP 和.MAP 文件

在详细介绍 Memory Debugger 之前，需要介绍 Memory Debugger 所需要的两种文件，预编译（.PP）文件和地址映像（.MAP）文件。在 CCS3.x 中，进入 Project\Build Options 选项，按图 4 红框中所示配置就可以生成所需要的文件。PP 文件主要保存预编译生成的文件，关于预编译的详细内容，可以参考文献 2。MAP 文件主要是把变量和它的地址都对应起来，如图 5 所示。MAP 文件的其他功能也可以参考文献 2。

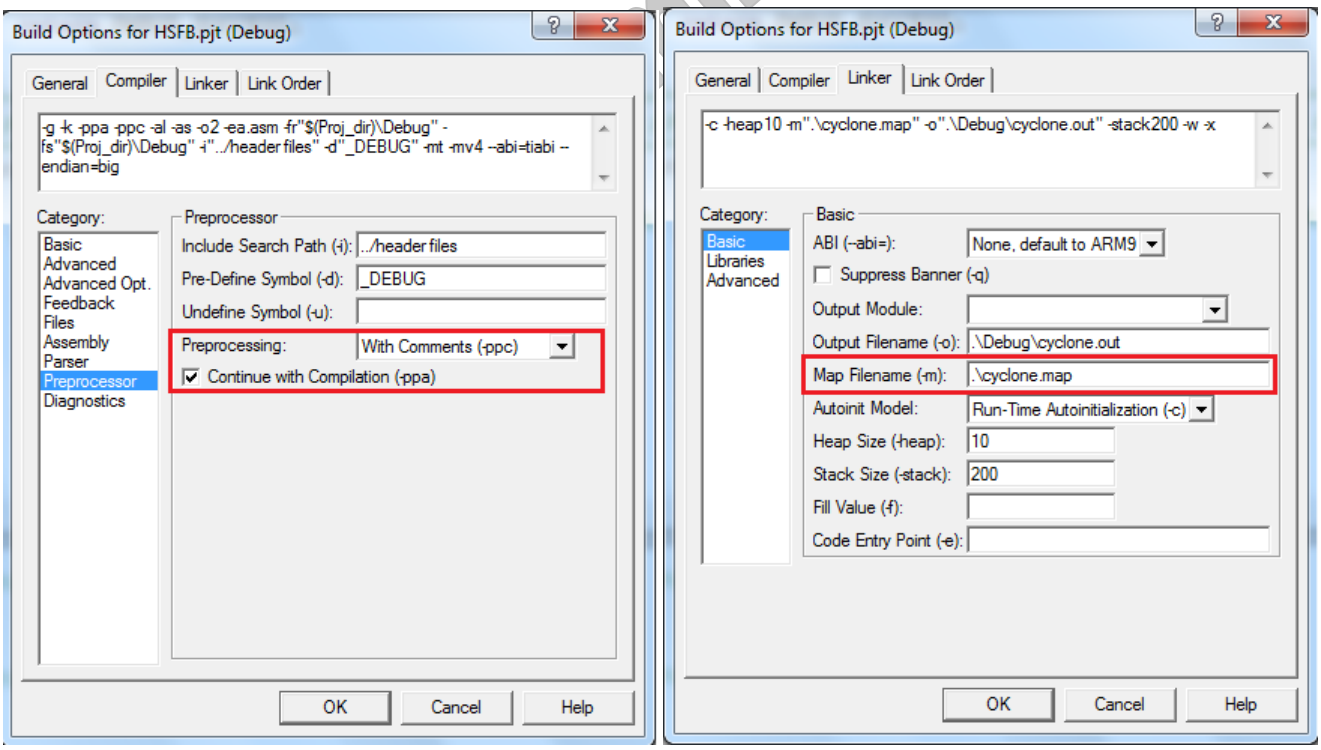

图4. 预编译（.PP）文件和地址映射（.MAP）文件的产生

```
MEMORY CONFIGURATION
                  name             origin    length      used     attr    fill
         ----------------------   --------  ----------  --------  ----  --------
         VECS                     00000000  00000020  00000020  RWIX
.....................................................................................
SECTION ALLOCATION MAP
 output                                  attributes/
section   page    origin      length       input sections
--------  ----  ----------  ----------   ----------------
.....................................................................................
.text       0    00000020    000057b4
.....................................................................................
GLOBAL SYMBOLS: SORTED ALPHABETICALLY BY Name
address     name
--------    ----
.....................................................................................
00019054    _debug_buffer
.....................................................................................
00019260    _supply_state
.....................................................................................
GLOBAL SYMBOLS: SORTED BY Symbol Address
address     name
--------    ----
.....................................................................................
00019054    _debug_buffer
.....................................................................................
00019260    _supply_state
.....................................................................................
```

图5. 地址映像（.MAP）文件

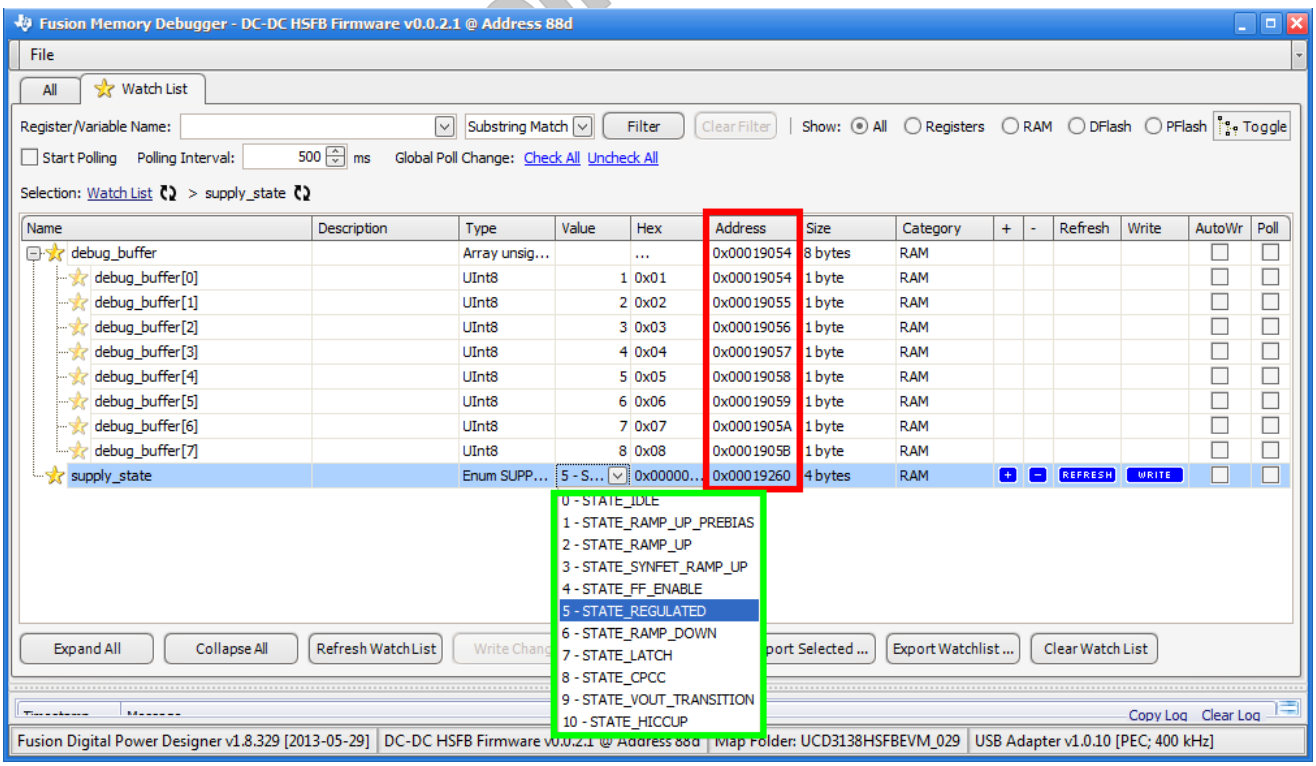

图6. Memory Debugger 变量显示

为什么 Memory Debugger 需要.PP 和.MAP 文件？Memory Debugger 通过读取.PP 文件可以知道变量是什么类型：结构体、数组、枚举等；通过.MAP 文件知道这些变量的地址。在图 5 中，可以看到变量 debug_buffer 和 supply_state 的首地址分别是 0x00019054 和 00019260。图 5 中红色虚线是指省略部分。在图 6 中，红框中是变量 debug_buffer 和 supply_state 的地址。debug_buffer 变量是 8 个字节的数组，如图 6 中红框所示，可以和图 5 地址对照一下，是一样的；supply_state 是枚举变量，如图 6 绿框所示，此时 supply_state 变量值是 5，处在 STATE_REGULATED 状态。

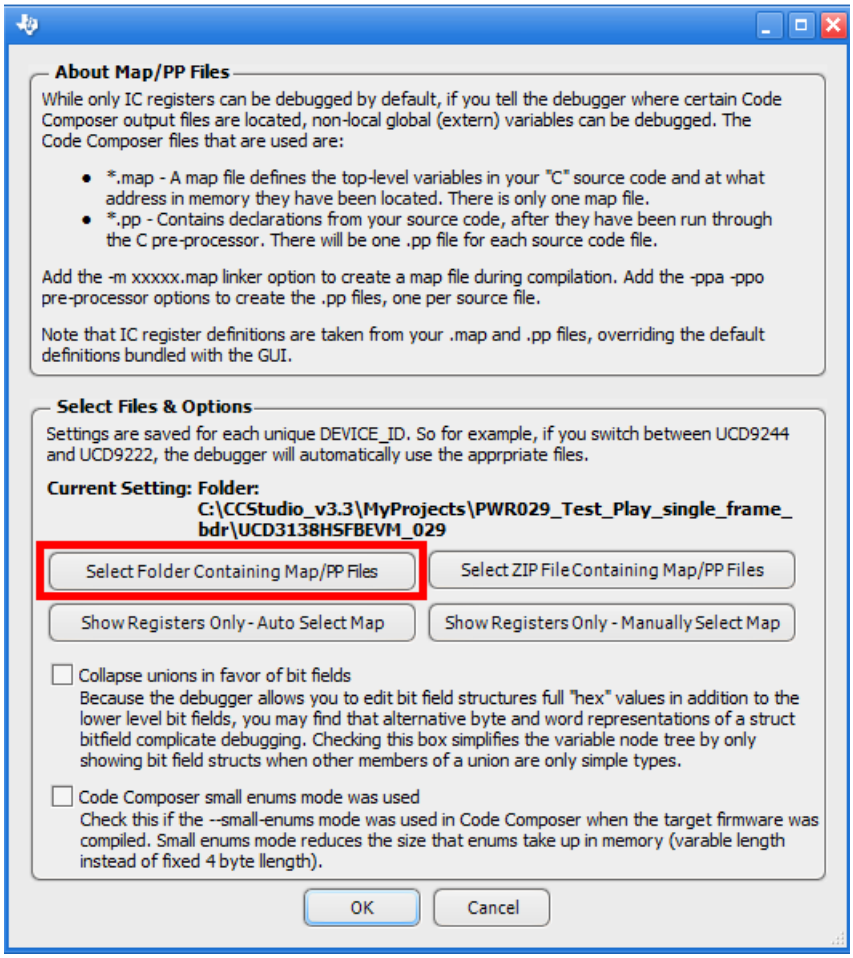

图7. 选择.PP 和.MAP 文件

4 Memory Debugger 的具体操作

Memory Debugger 主要使用命令 0xE2 和 0xE3 和 UCD3xxx 通讯，所以 UCD3xxx 代码需要支持这两个命令。这两个命令可以直接读取 UCD3xxx 内部程序 FLASH 和数据 FLASH 的值，可以读取或修改内部 RAM 的值。关于这两个命令的详细代码，请参考文献 3。

图 1 中绿框所示是切换 Memory Debugger 显示内容，包括显示所有（ALL）、寄存器（Register）、RAM、程序闪存（PFLASH）以及数据闪存（DFLASH）。如果想刷新某个变量，可以点击每个变量右边的刷新（REFRESH）按钮，如图 1 中粉红框所示。

4.1 选择.PP 和.MAP 文件

在 Memory Debugger 文件菜单下，选择 Change Map…（图 1 红框所示），会弹出图 7 所示界面。点击图 7 中红框所示，选择.PP 和.MAP 所在的目录（.PP 和.MAP 需要在同一个文件夹里面）。如果不需要看全局变量，只看寄存器的话，可以不需要执行这一步。因为对于寄存器变量，地址都固定的，Memory Debugger 可以识别这些寄存器。对于变量，每个项目都不一样，即使同一个变量，地址也可能不同，所以如果看全局变量，需要执行这一步。如果程序没有改动，执行一次就可以一直调试下去，Memory Debugger 会记住上一次的选择的.PP 和.MAP 文件。

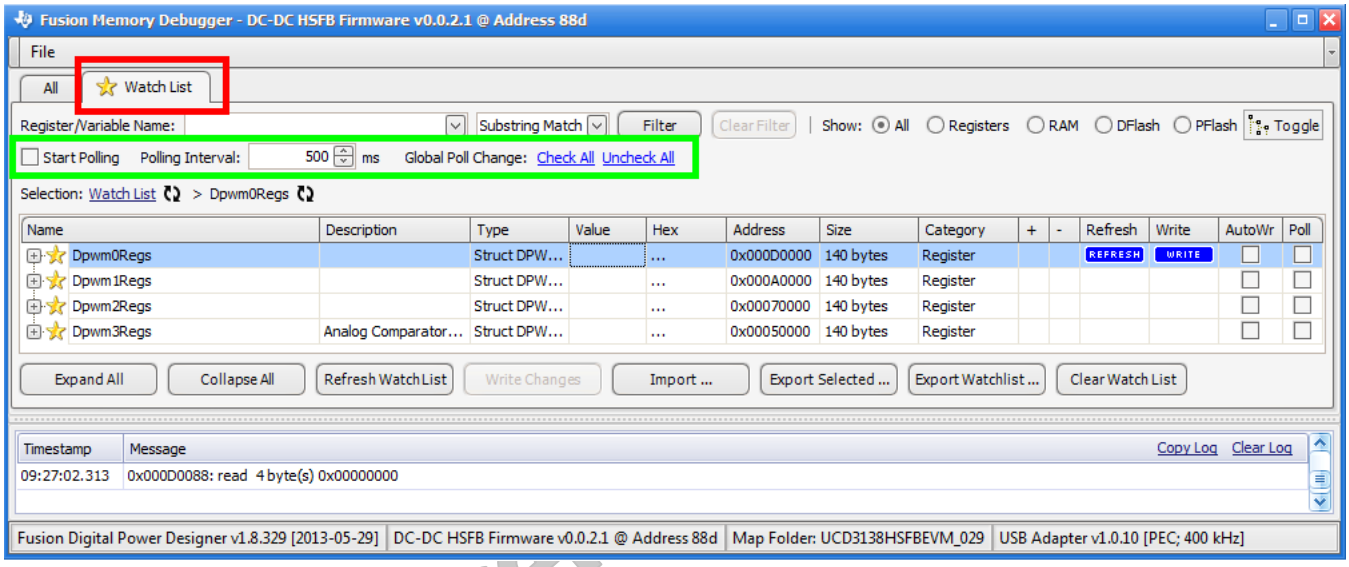

图8. 监控列表（Watch List）视图

4.2 放置变量到监控列表（Watch List）

进入 Memory Debugger 界面时，是显示所有的变量和寄存器，如图 1 所示。需要注意的是，UCD3xxx 有些寄存器位是读清除的，所以不能随便读取某个寄存器，需要结合编程手册来看这个寄存器是否可以读取。为了方便使用，可以把经常需要读取的寄存器和变量加入到监控列表（Watch List），如图 8 红框所示。在图 1 中，如果我们点击每个变量左边灰色的星号（蓝色框所示），那么就可以把想要监控的寄存器或变量加入到监控列表中，如图 8 所示。在监控列表中，可以设置自动刷新所监控的寄存器或变量，自动刷新频率也可以设置，如图 8 绿色方框所示。

4.3 实验演示

把寄存器 Fiter0Regs 放到监控列表，展开后找到 OUTPUT_CLAMP_HIGH，如图 9 所示。初始值是 0x2500。点击 0x2500，输入 0x1250，然后点击右边的 WRITE，写入 UCD3xxx，如图 10 所示。

图 11 是 DPWM0A/B（Ch1-DPWM0A, Ch2-DPWM0B）的波形，可以看出修改前后 DPWM 的最大占空比的变化。图 11 左边是最大占空比修改前，右边是最大占空比修改后。实验是在 TI 实验板（UCD3138OL40EVM-032，见文献 4）上验证的，把反馈短接到地，然后把示例代码各种保护屏蔽掉，那么程序就可以运转起来，环路会一直输出最大占空比。

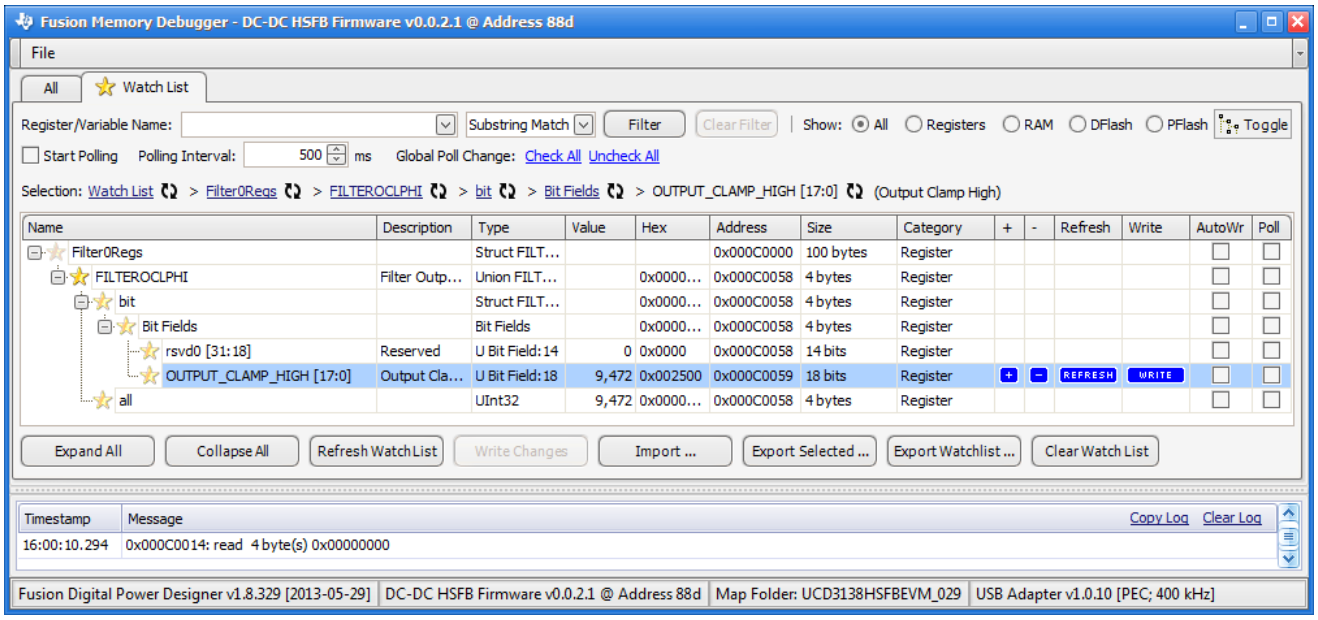

图9. DPWM 输出最大占空比修改前

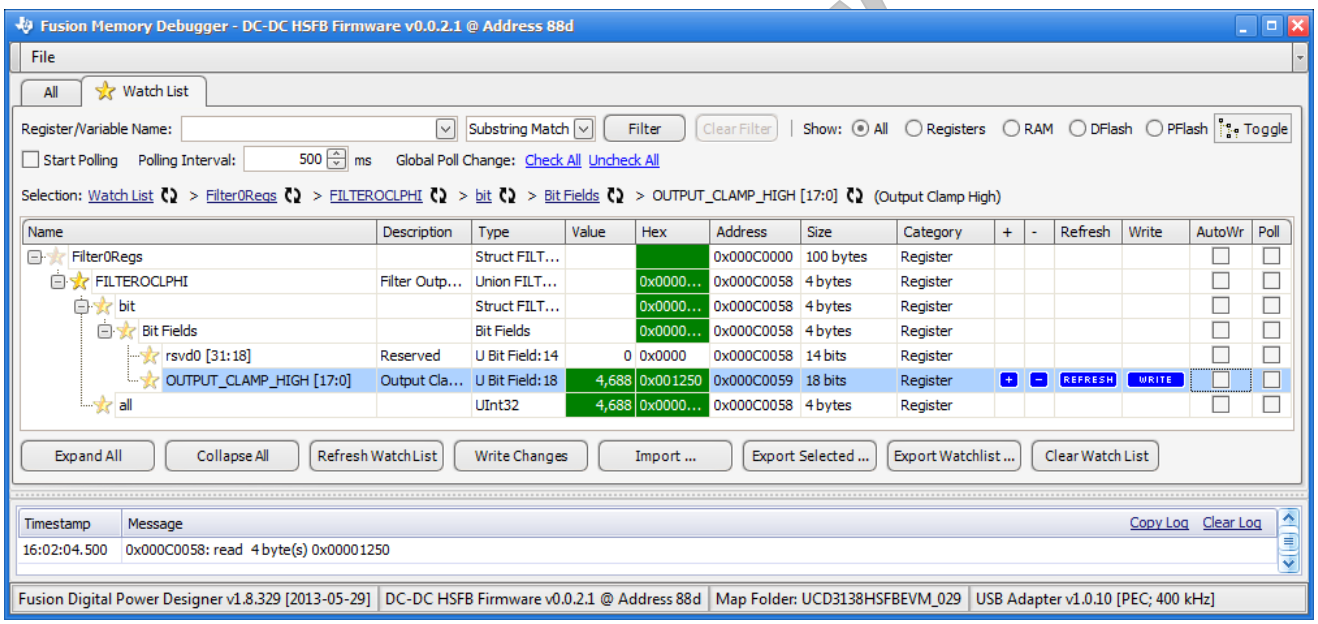

图10. DPWM 输出最大占空比修改后

5 总结

本文主要介绍使用 Memory Debugger 来调试 UCD3xxx。通过本文可以看出，利用 Memory Debugger，可以灵活地监控及修改变量，达到调试的目的。

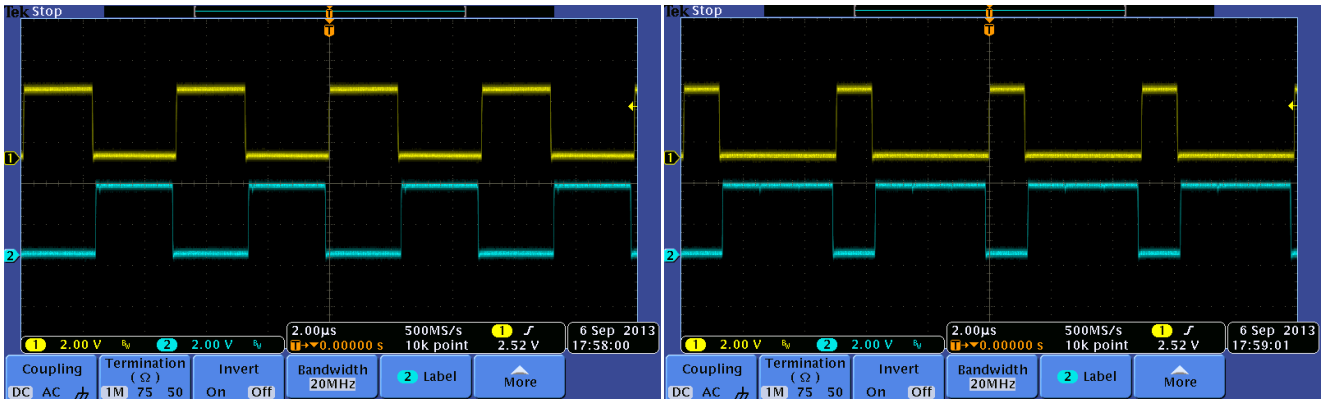

图11. DPWM 输出最大占空比修改前后的波形

6 参考文献

1. PMBUS Power System Management Protocol Specification Part II – Command Language, System Management Interface Forum (SMIF), Inc., 2007;

2. SPNU151H - ARM Optimizing C/C++ Compiler v5.0 User's Guide, Texas Instruments, 2012

3. UCD3138HSFBEVM-029 Firmware (PWR029_Test_Play_single_frame_bdr), Texas Instruments, 2013

4. SLUUA80 - Using the UCD3138OL40EVM-032 User's Guide, Texas Instruments, 2013

5. SLUSAP2C - UCD3138 Highly Integrated Digital Controller for Isolated Power, Texas Instruments, 2013