创龙 PCIe 与 PC 通信开发

Revision History

Draft Date	Revision No.	Description
2016/01/18	V1.0	1.初始版本。

目　录

说明：本使用指南仅提供基于 TL665x-EasyEVM 开发板的 PCIe 与 PC 机通信开发方法，其中使用的开发工具仅供参考使用。本文测试中使用的 Linux 系统发行版本为 LinuxDeepin 2014.3（32 位）、使用的 Windows 系统版本为 Windows10 专业版（64 位），暂不确定其他操作系统下的测试效果。

在调试过程中还会用到 CCS 集成开发环境以及串口调试工具，本文默认已经将 DSP 开发的软、硬件相关环境配置正确。

1. 创龙 C66x DSP TMS320C665x Linux 系统下通过 PCIe 加载启动

操作中用到的工程源码在"Demo\PCIe Linux Loader"下。

物理机安装 Linux 系统

由于 PCIe 设备需要支持 VT-d 指令集的硬件及虚拟化软件，才能够映射到虚拟机中使用，所以这里以安装在物理机上的 Linux 系统完成测试。

可以通过第三方工具检查 CPU 是否支持 VT-d 指令集。

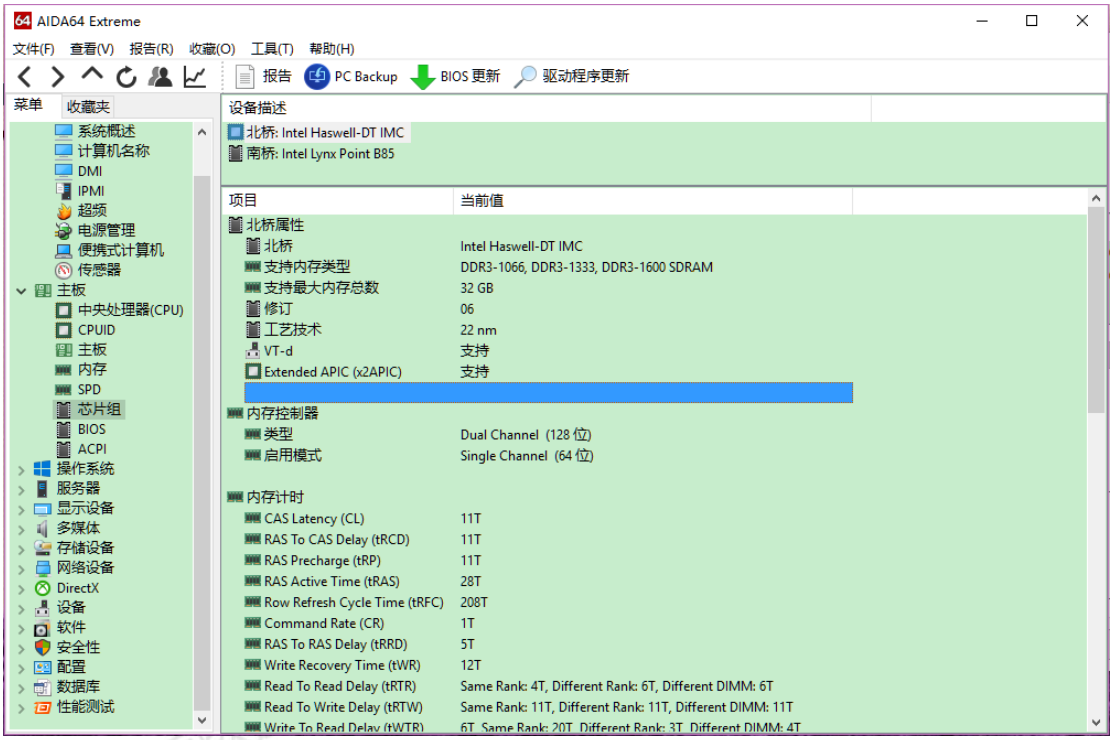

图 1

修改 IBL 启动模式为 PCIe

由于 KeyStone I（DSP C665x 及 DSP C667x）内置的 ROM BootLoader 在对 PCIe 启动

的配置中缺失或者某些配置项目或者配置值不合适，导致不能被 PC 正确识别（在 Windows 系统下会出现无法正确分配资源的错误导致驱动无法正确加载）。所以，需要依赖 IIC EEPROM 中的 IBL 辅助配置。

请参阅《TMS320C665x 开发例程使用手册》中仿真器加载程序章节内容，将 IBL 配置参数做出如下修改，启动的时候将拨码开关配置为 OFF OFF ON ON ON。

ibl.RomBoot.Enable = TRUE;

ibl.RomBoot.Mode = 0x1809;

关于 0x1809 的含义，请参阅参考文档《DSP C665x 启动模式汇总》。

1.1. 创龙 C66x DSP TMS320C665x 连接 DSP 板卡到 PCIe x4 或者 x16 插槽

KeyStone I DSP 支持单个 PCIe 端口（Port）以及两个 PCIe 通道（Lane），但是 PC 上一般只有 x1、x4 以及 x16 插槽。

板卡可以直接从 PCIe 供电，所以请勿额外连接电源适配器。

1.2. 创龙 C66x DSP TMS320C665x 启动 PC 到 Linux

可能需要在 Linux 加载内核前复位（Full Reset）板卡，以便可以被系统枚举到。

1.3. 创龙 C66x DSP TMS320C665x 查询设备

打开终端（Shell），执行 lspci 命令。可以找到 Multimedia controller: Texas Instruments Device b005（rev 01）设备。

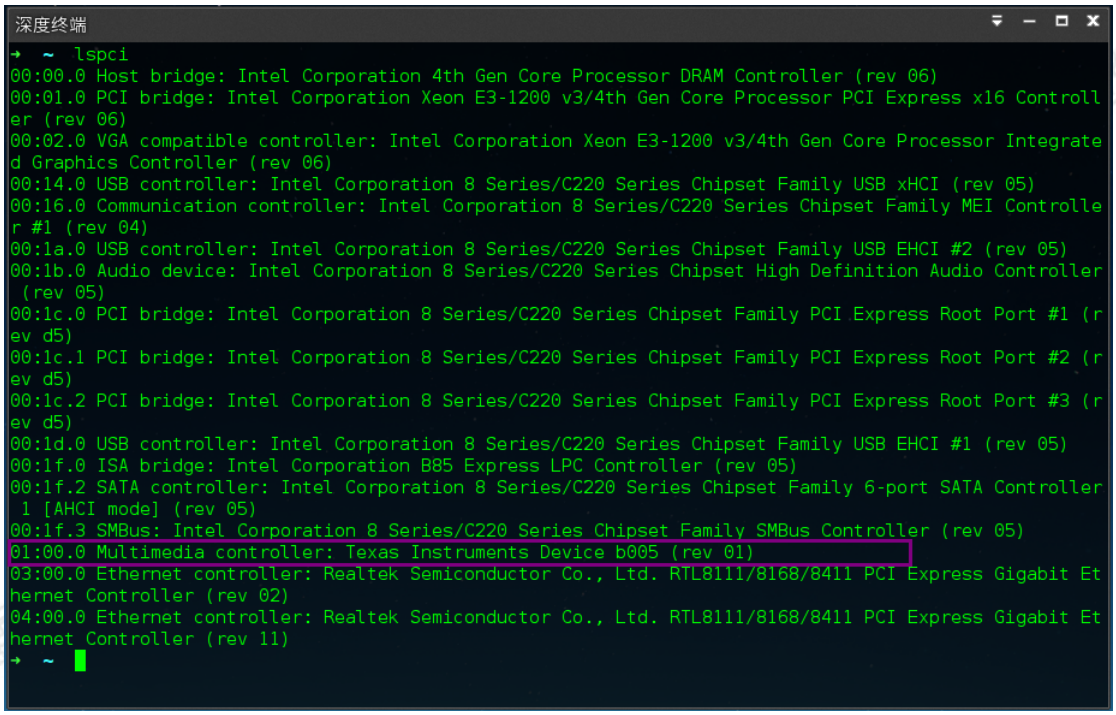

图 2

还可以通过 lspci -v 命令查看设备详细信息，例如内存映射。

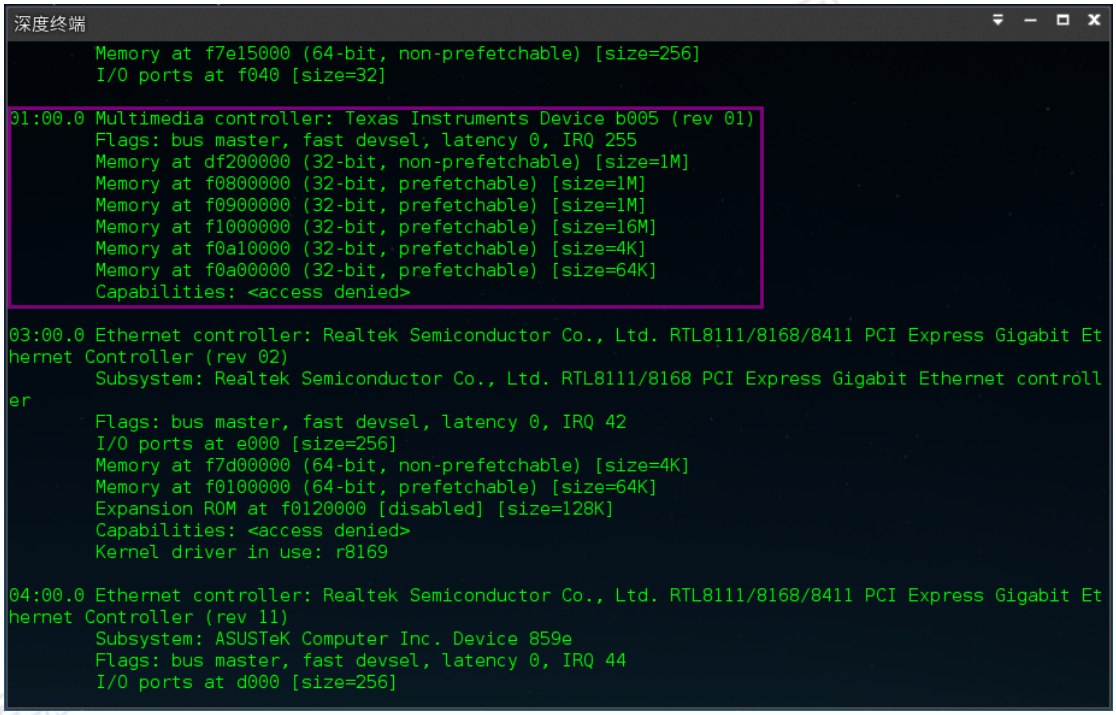

图 3

在 IBL 对 PCIe 启动配置代码中，配置了四组 BAR Mask 寄存器值

BAR0 Mask=4KB

BAR1 Mask=1MB

BAR2 Mask=1MB

BAR3 Mask=16MB

操作系统在枚举 PCIe 设备的时候会根据该值为 PCIe 设备分配内存空间。

IBL 相关配置代码如下：

```
void iblPCIeWorkaround()
{
    UINT32  v, flag_6678 = 0, flag_6670 = 0, flag_6657 = 0, MAGIC_ADDR;
    UINT32  i;

     /* Power up PCIe */
    devicePowerPeriph (TARGET_PWR_PCIE);
    for(i=0; i<1000; i++) asm (" NOP");

    DEVICE_REG32_W ((PCIE_BASE_ADDR + PCIE_APP_SERDES_CFG0), 0x00062320);  /* ss clock */
    DEVICE_REG32_W ((PCIE_BASE_ADDR + PCIE_APP_SERDES_CFG1), 0x00022320);  /* ss clock */

    /* Wait for PCIe PLL lock */
    while(!(DEVICE_REG32_R(PCIE_STS_REG) & 1));

    /* Determine 6670 or 6678 */
    v = *((Uint32 *)DEVICE_JTAG_ID_REG);
    v &= DEVICE_JTAG_ID_MASK;

    if (v == DEVICE_C6678_JTAG_ID_VAL) {
        MAGIC_ADDR = 0x87fffc;
        flag_6678 = 1;
```

```
    }
    if (v == DEVICE_C6670_JTAG_ID_VAL) {
        MAGIC_ADDR = 0x8ffffc;
        flag_6670 = 1;
    }
    if (v == DEVICE_C6657_JTAG_ID_VAL) {
                MAGIC_ADDR = 0x8ffffc;
                flag_6657 = 1;
        }

    DEVICE_REG32_W ((PCIE_BASE_ADDR + PCIE_CLASSCODE_REVID), 0x04800001);  /* class 0x04, sub-class 0x80, Prog I/F 0x00, Other multimedia device */
    DEVICE_REG32_W ((PCIE_BASE_ADDR + PCIE_LINK_STAT_CTRL), 0x10110080);  /* extended sync, slot_clk_cfg = 1 */

    DEVICE_REG32_W ((PCIE_BASE_ADDR + PCIE_VENDER_DEVICE_ID), 0xb005104c);  /* Vendor and Device ID */
    DEVICE_REG32_W ((PCIE_BASE_ADDR + PCIE_DEVICE_CAP), 0x288701); /* L0 = 4, L1 = 3 */

    DEVICE_REG32_W ((PCIE_BASE_ADDR + PCIE_APP_OB_SIZE), 0x00000003);       /* OB_SIZE = 8M */
    DEVICE_REG32_W ((PCIE_BASE_ADDR + PCIE_PL_GEN2), 0x0000000F);    /* num_fts = 0xF*/

    DEVICE_REG32_W ((PCIE_BASE_ADDR + PCIE_APP_CMD_STATUS), 0x0020); /* Set dbi_cs2 to allow access to the BAR registers */

    if (flag_6678)  {
        /* 6678 */
        DEVICE_REG32_W ((PCIE_BASE_ADDR + PCIE_BAR0), 0x00000FFF);    /* 4K */
```

```
        DEVICE_REG32_W ((PCIE_BASE_ADDR + PCIE_BAR1), 0x0007FFFF);    /* 512K */
        DEVICE_REG32_W ((PCIE_BASE_ADDR + PCIE_BAR2), 0x003FFFFF);    /* 4M */
        DEVICE_REG32_W ((PCIE_BASE_ADDR + PCIE_BAR3), 0x00FFFFFF);    /* 16M */
    }

    if (flag_6670)  {
        /* 6670 */
        DEVICE_REG32_W ((PCIE_BASE_ADDR + PCIE_BAR0), 0x00000FFF);    /* 4K */
        DEVICE_REG32_W ((PCIE_BASE_ADDR + PCIE_BAR1), 0x000FFFFF);    /* 1M */
        DEVICE_REG32_W ((PCIE_BASE_ADDR + PCIE_BAR2), 0x001FFFFF);    /* 2M */
        DEVICE_REG32_W ((PCIE_BASE_ADDR + PCIE_BAR3), 0x00FFFFFF);    /* 16M */
     }
    if (flag_6657)  {
                /* 6657 */
                DEVICE_REG32_W ((PCIE_BASE_ADDR + PCIE_BAR0), 0x00000FFF);    /* 4K */
                DEVICE_REG32_W ((PCIE_BASE_ADDR + PCIE_BAR1), 0x000FFFFF);    /* 1M */
                DEVICE_REG32_W ((PCIE_BASE_ADDR + PCIE_BAR2), 0x000FFFFF);    /* 1M */
                DEVICE_REG32_W ((PCIE_BASE_ADDR + PCIE_BAR3), 0x00FFFFFF);    /* 16M */
        }
    DEVICE_REG32_W ((PCIE_BASE_ADDR + PCIE_APP_CMD_STATUS), 0x0);      /* dbi_cs2=0 */

    DEVICE_REG32_W ((PCIE_BASE_ADDR + PCIE_STATUS_CMD), 0x00100146); /* ENABLE mem acc
ess */
    DEVICE_REG32_W ((PCIE_BASE_ADDR + PCIE_DEV_STAT_CTRL), 0x0000281F); /* Error control */
    DEVICE_REG32_W ((PCIE_BASE_ADDR + PCIE_ACCR), 0x000001E0); /* Error control */
    DEVICE_REG32_W ((PCIE_BASE_ADDR + PCIE_BAR0), 0); /* non-prefetch, 32-bit, mem bar */

    DEVICE_REG32_W ((PCIE_BASE_ADDR + PCIE_APP_CMD_STATUS), 0x0000007);      /* enable LTS
SM, IN, OB */
```

```
    while((DEVICE_REG32_R(PCIE_BASE_ADDR + PCIE_DEBUG0) & 0x11)!=0x11);      /* Wait for trai
ning to complete */


    /* Wait for the Boot from Host */
    DEVICE_REG32_W(MAGIC_ADDR, 0);
    waitForBoot(MAGIC_ADDR);
    /* Will never reach here */
    return;
}
```

1.4. 创龙 C66x DSP TMS320C665x 编译驱动

将 Linux_Host_Loader 目录从光盘中复制到 Linux 文件系统中的任意目录。路径中不要有空格及非 ASCII 字符（中文等）。

打开终端并进入该目录，执行 make clean 及 make 命令。成功编译后会在当前目录生成 PCIeLoader.ko（注意大小写）的驱动模块。请确保 x86 的 gcc 编译器安装及配置正确，通常情况下 Linux 发行版都内置该工具。

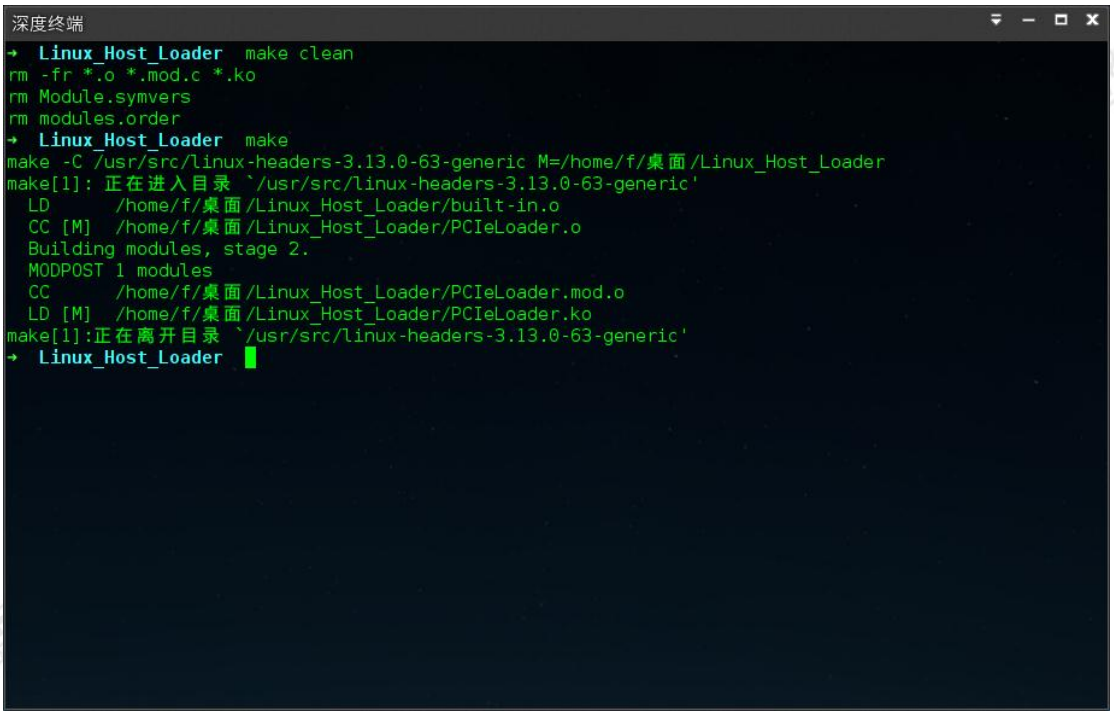

图 4

1.5. 创龙 C66x DSP TMS320C665x 加载驱动

在终端中执行 sudo insmod PCIeLoader.ko 命令，然后输入当前用户密码，即可成功加载该驱动。此时，可以看到底板 LED 循环点亮。

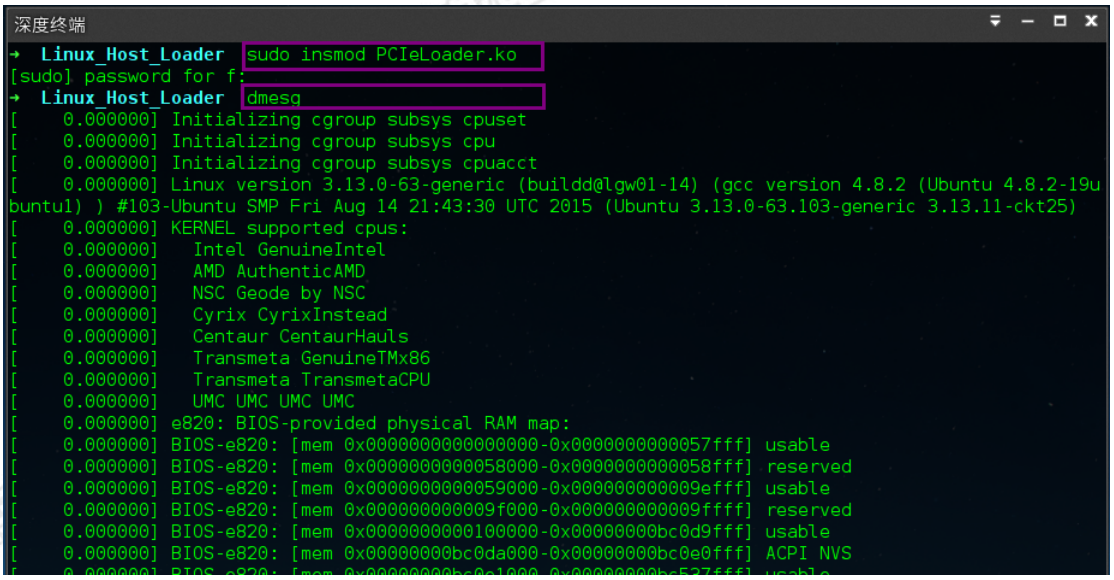

图 5

在终端中执行 dmesg 命令，可以看到驱动加载产生的内核日志。驱动的基本流程是先将程序按照段加载到相应的内存区域，然后写_C_int00 地址到 Boot Magic。由于 IBL 在一直查询 Boot Magic，所以当该地址值变为有效后跳转到该地址，从而完成启动操作。

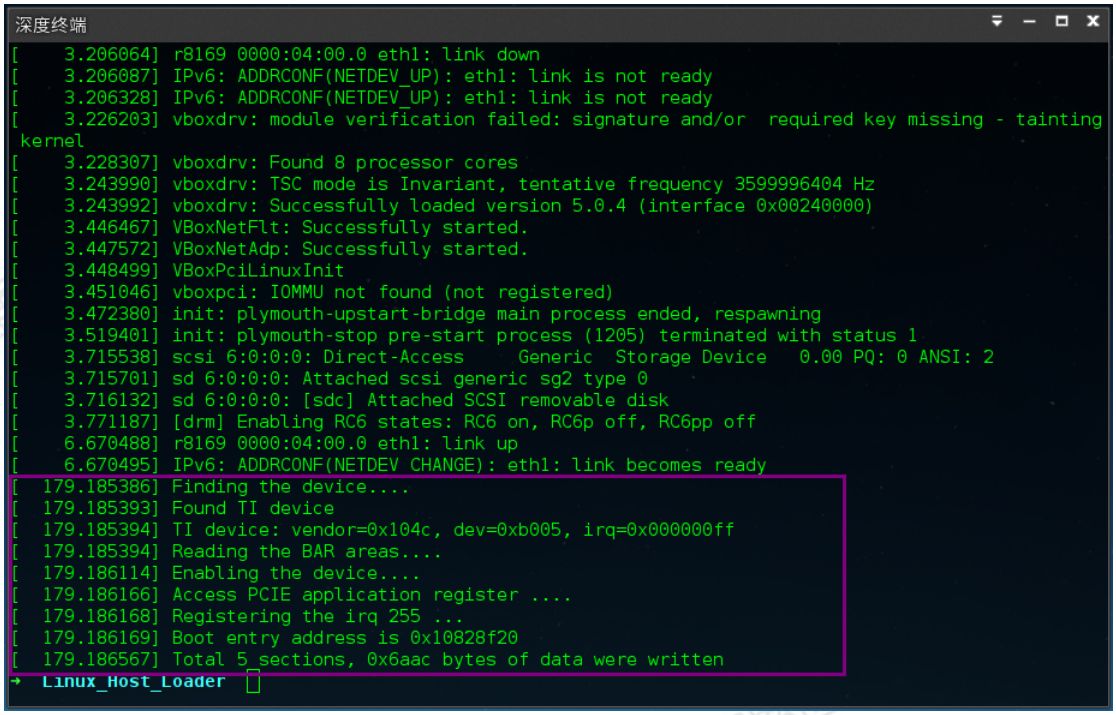

图 6

1.6. 创龙 C66x DSP TMS320C665x 通过 CCS 调试

请参阅相关文档安装 Linux 下的 CCS 集成开发环境并配置仿真器配置文件，这里以 XDS200 仿真器为例。

(1) 打开 CCS，默认情况下 Linux 版本 CCS 安装到/opt/ti 目录下，可执行主程序为/opt/ti/ccsv5/eclipse/ccstudio。不需要使用 root 权限运行。

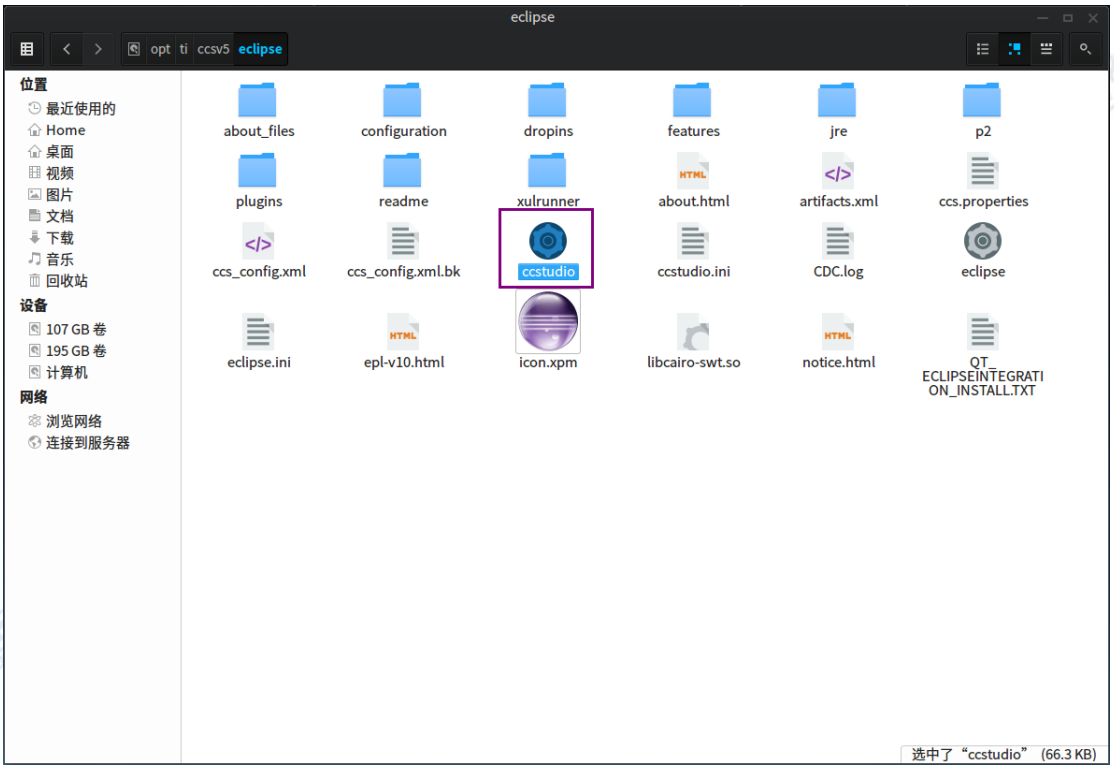

图 7

(2) 加载仿真器配置文件进入 Debug 模式。

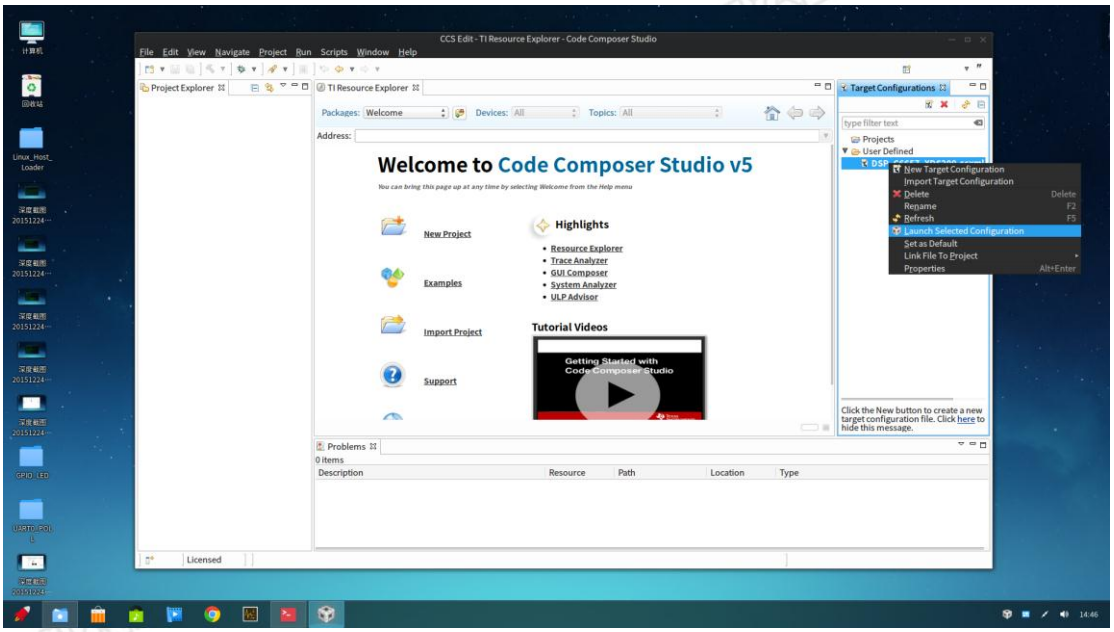

图 8

(3) 连接 DSP C6657 核心 0。

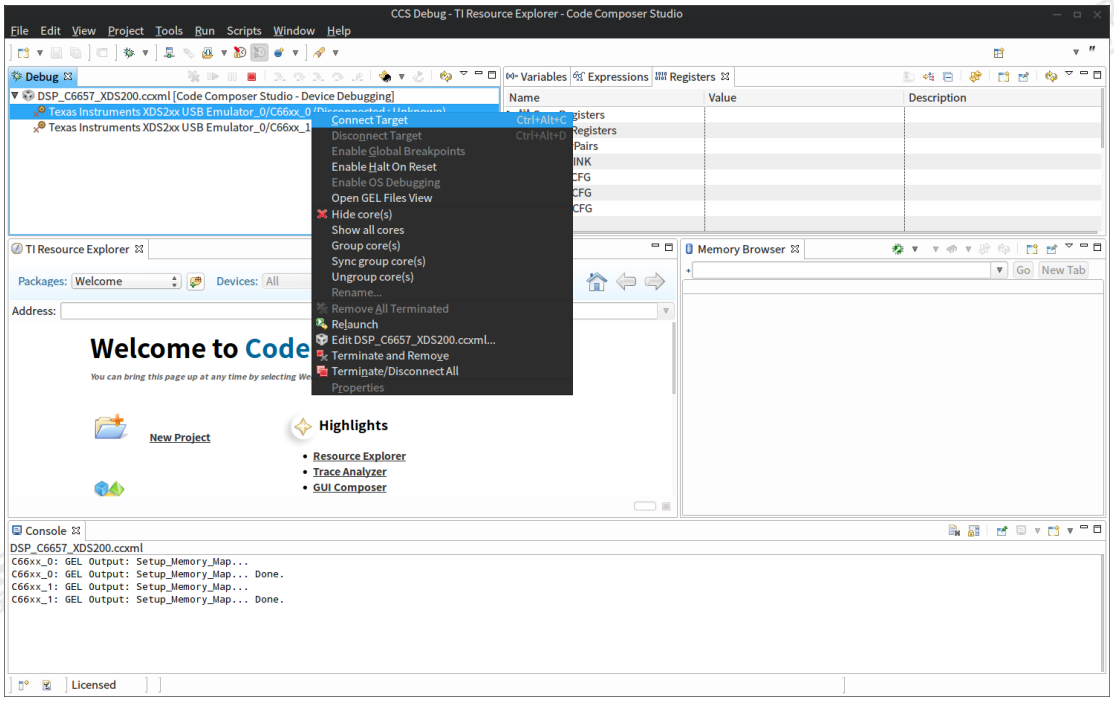

图 9

(4) 加载程序符号（GPIO_LED.out 文件）。

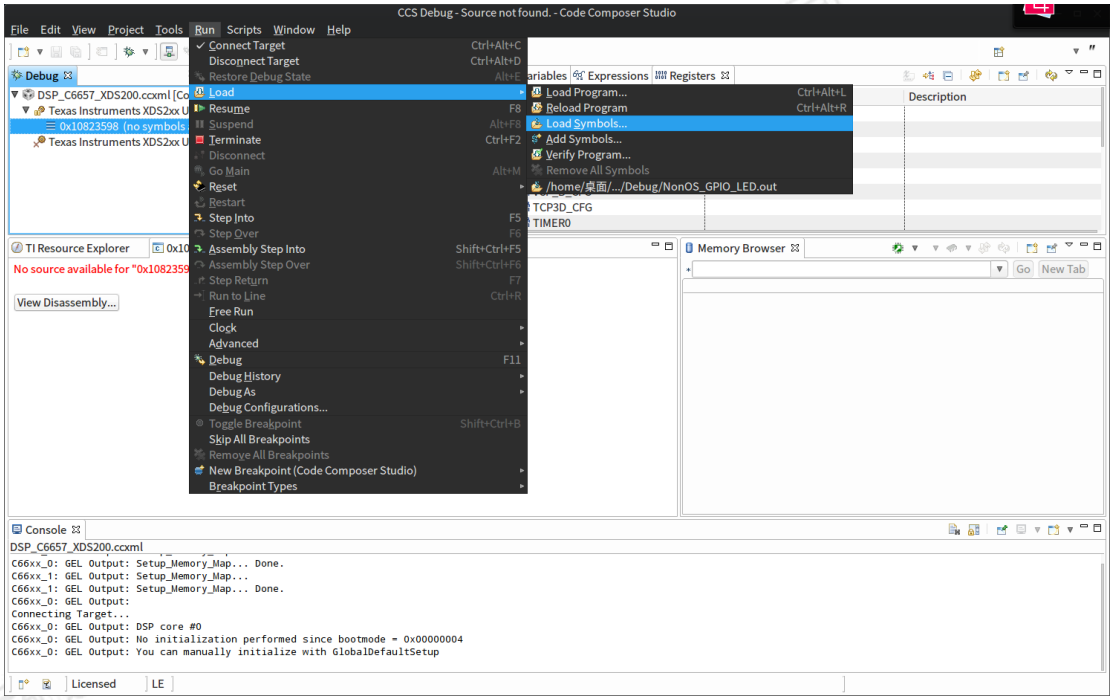

图 10

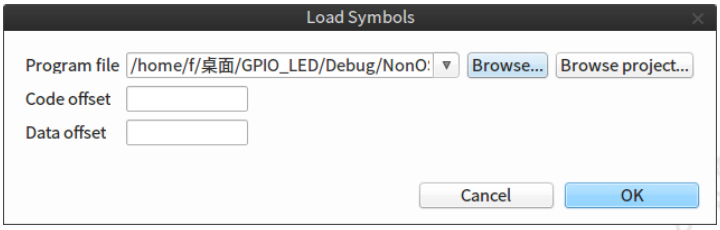

图 11

(5) 加载成功后就可以对该程序进行调试。

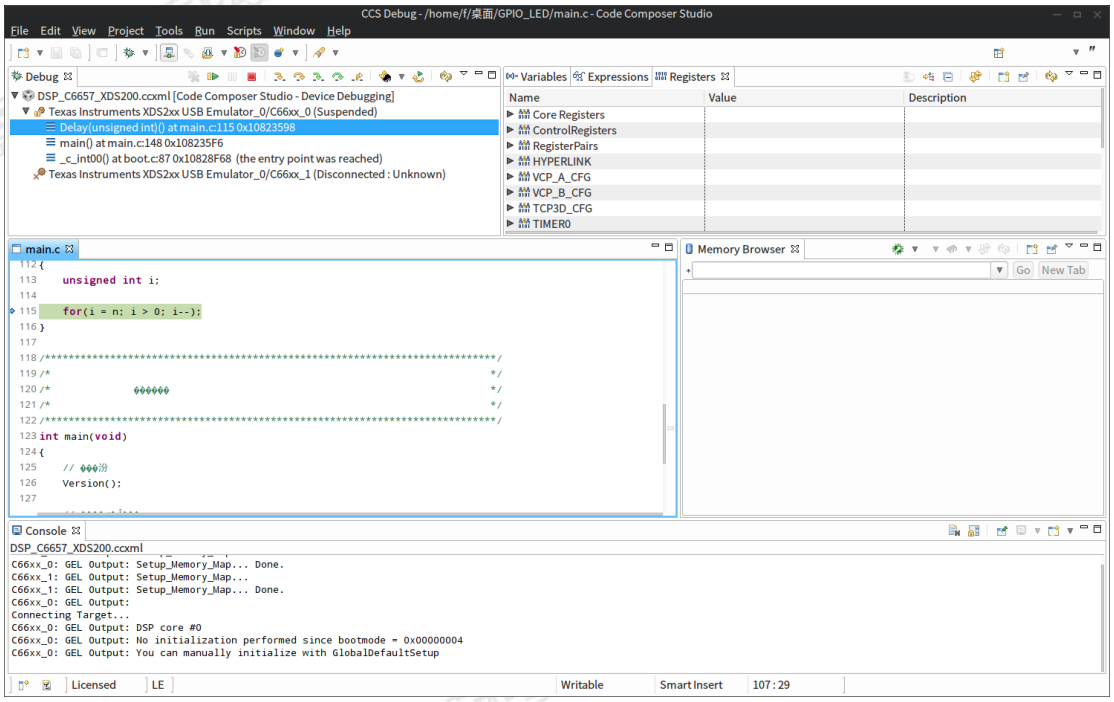

图 12

1.7. 创龙 C66x DSP TMS320C665x 测试串口例程

打开 PCIeLoader.c 源文件，修改宏 GPIO_LED 为 0，UART0_POLL 为 1。

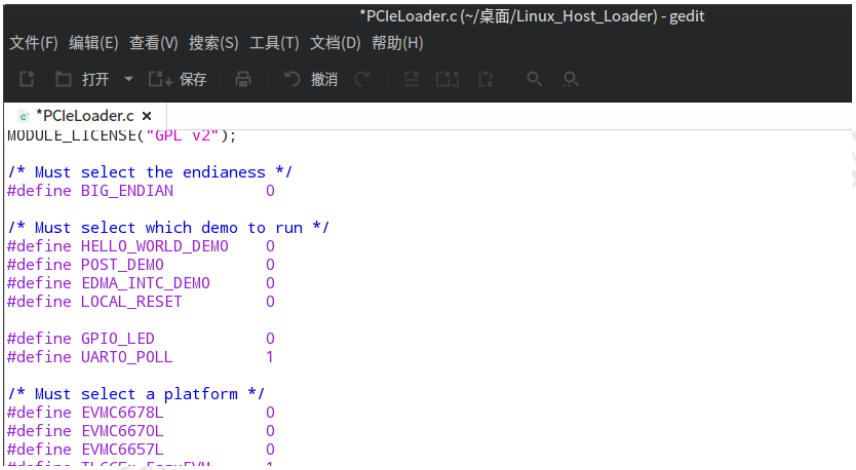

图 13

重新编译该驱动。然后，配置串口终端，这里以 putty 为例。

查看 USB 转串口设备。在终端中执行 ls /dev 命令其中 ttyUSB0 就是 USB 转串口设备，不同电脑可能有所不同。

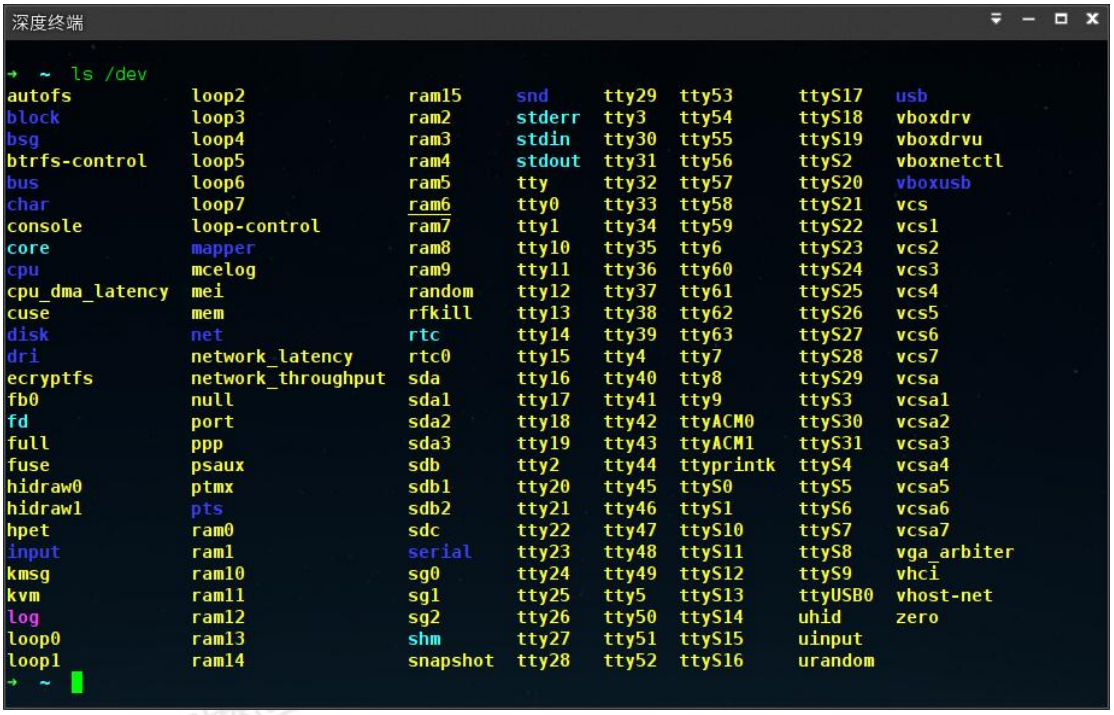

图 14

在终端中执行 sudo putty 命令，输入当前用户密码后，配置串口参数为波特率 115200，数据位 8，无检验位，停止位 1，关闭硬件流控。配置完成后点击 open 按钮。

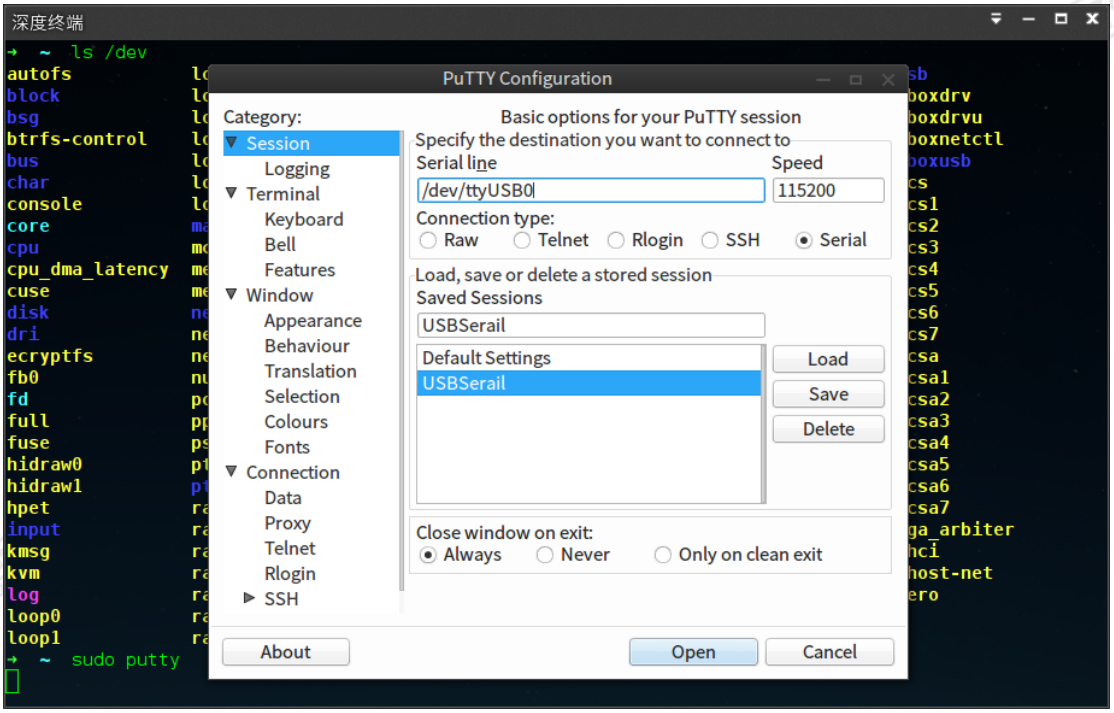

图 15

加载重新编译后的 PCIeLoader.ko 驱动。由于，之前已经完成启动操作，所以加载新的程序需要重新启动开发板及 PC。可以看到串口输入提示字符，还可以随意输入字符 DSP 会回显输入的字符。

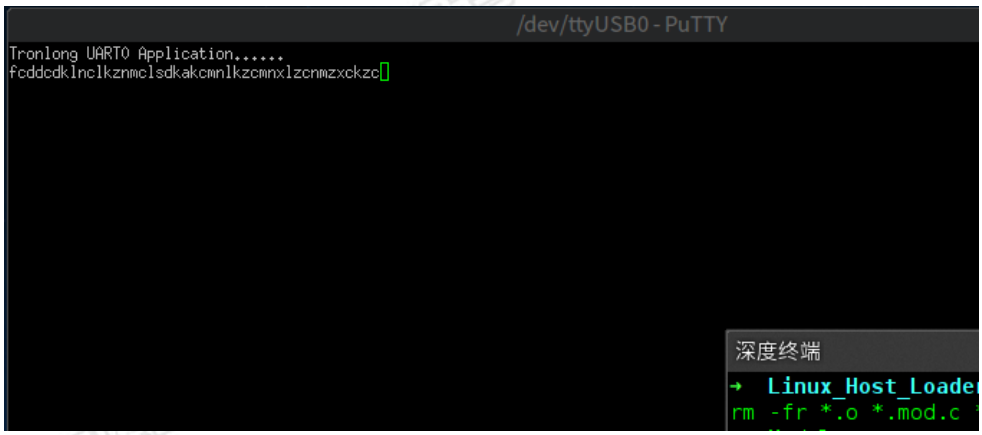

图 16

1.8. 创龙 C66x DSP TMS320C665x 生成启动文件

前述的测试都仅仅使用预生成的启动文件，可以根据需要将任意 DSP 程序转换成启动文件。

打开 Boot Image 目录下 Build.cmd 文件，修改编译工具链路径为实际安装路径。

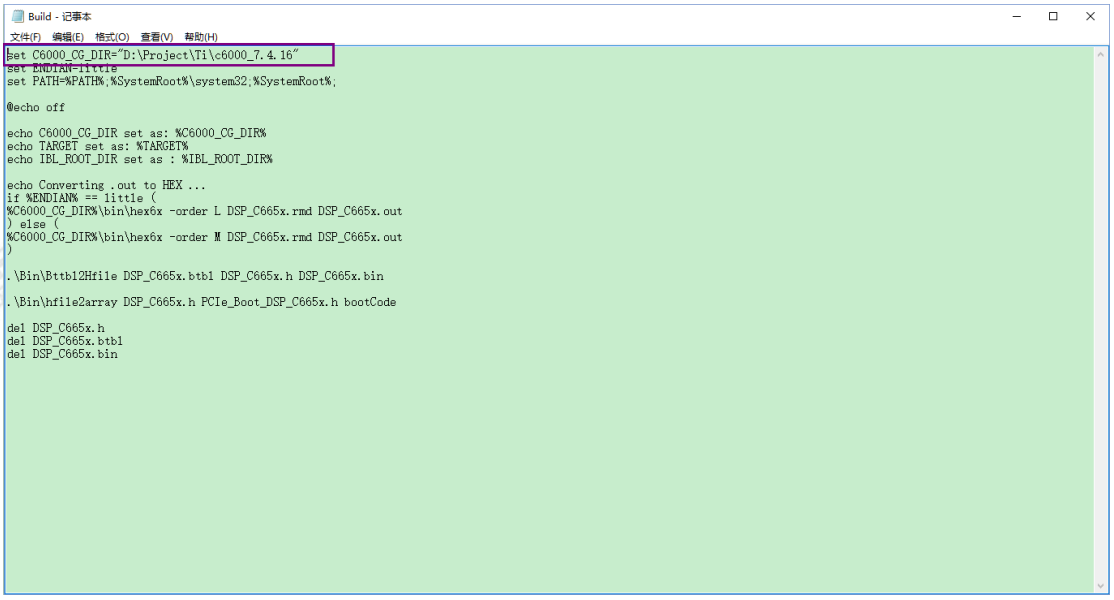

图 17

将需要转换的 DSP 程序更名为 DSP_C665x.out，然后双击 Build.cmd 文件完成转换，可以看到生成 PCIe_Boot_DSP_C665x.h 的文件。之后，用该文件替换 Linux_Host_Loader 目录下的 GPIO_LED.h 或者 UART0_POLL.h 文件（与 PCIeLoader.c 中的宏有关），重新编译该驱动即可。

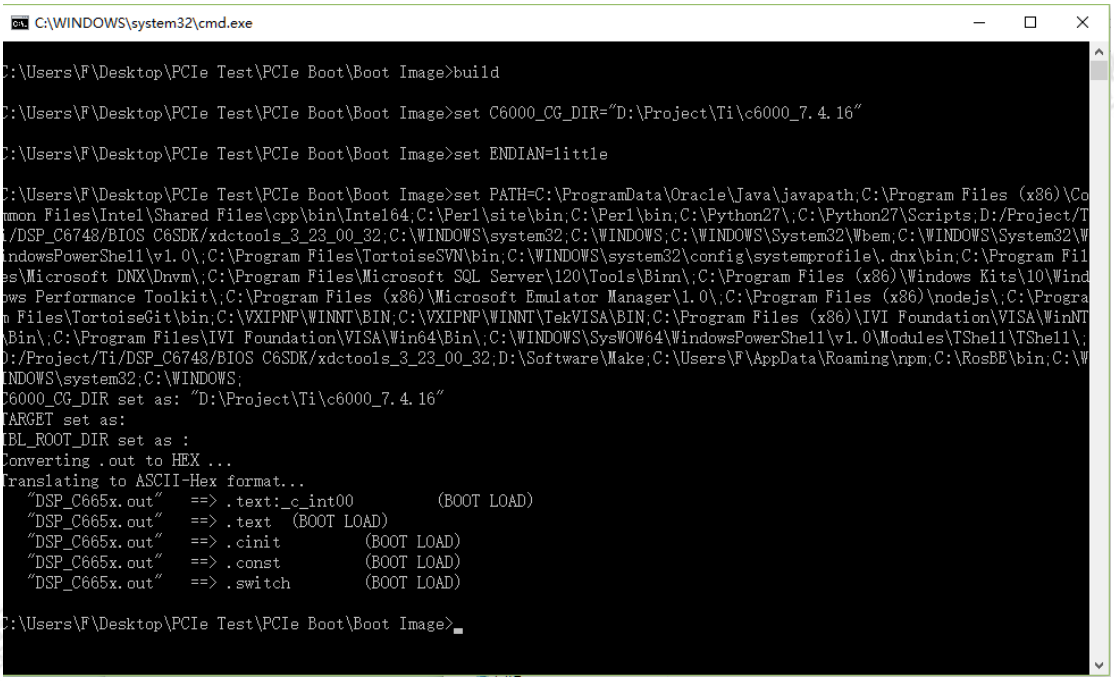

图 18

2. 创龙 C66x DSP TMS320C665x Windows 系统下通过 PCIe 通信

本文测试中使用的 Windows 系统版本为 Windows 10 专业版（64 位）。

安装 WinDriver 软件

WinDriver 是一款可以简化 Windows 下 USB 及 PCI/PCIe 驱动开发的软件，可以不理解 Windows 系统底层就可以开发出相应功能的驱动。该软件是共享软件而且授权费用不菲，同时处于性能上面的考虑，一般情况下只做测试使用或者要求不高时使用。可到 Jungo 官网下载：http://www.jungo.com/st/products/windriver/。

本文测试中使用的 WinDriver 软件版本为 11.9。

使用前需将启动模式配置为 IBL PCIe Boot 的方式，或者将拨码开关配置为 No Boot，通过 CCS 加载 PCIe_EndPoint_PC 程序。建议 DSP 端使用 PCIe_EndPoint_PC 这个程序，加载这个程序可以通过 GEL 文件初始化 DDR3 以便可以访问 DDR3，同时禁用了 L1 及 L2 缓存，否则还需要在 DSP 端维护缓存一致性来保证从 PC 端写入的数据与 DSP 端读取到的数据是一致的。

安装 Microsoft Visual Studio

使用 Visual Studio 才能将生成的库及驱动编译成.sys 及.dll 文件，同时也可以生成相应的测试程序文件。可以在微软官网下载到免费的 Express 版本。

本文测试中使用的 Visual Studio 软件版本为 2015。

2.1. 创龙 C66x DSP TMS320C665x 正确枚举设备

进入 Windows 系统后，复位板卡（Full Reset），打开设备管理器点击扫描检测硬件改动可以看到系统会检测出新的“多媒体控制器设备”。

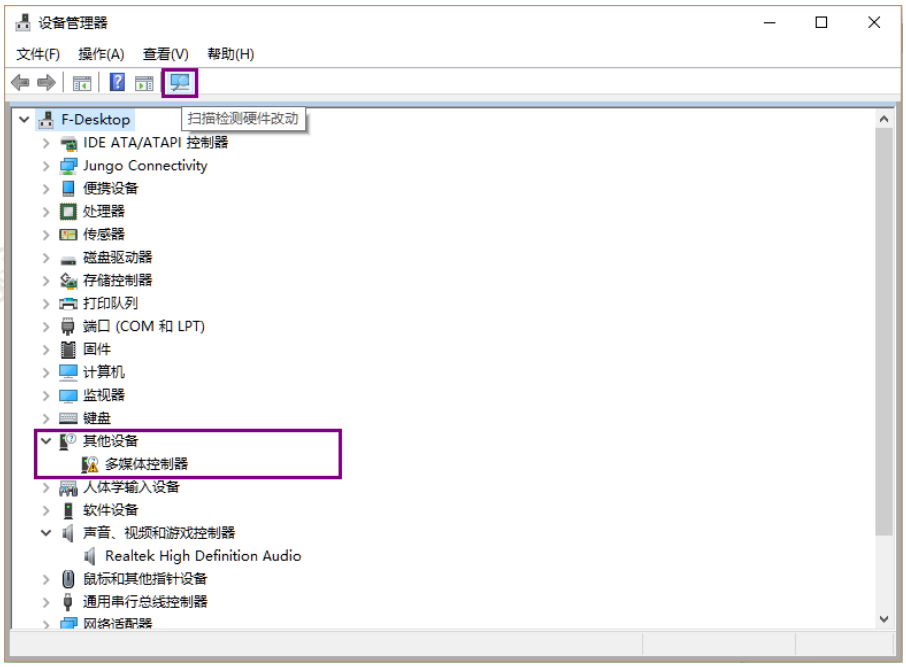

图 19

2.2. 创龙 C66x DSP TMS320C665x 安装 INF 文件

打开 WinDriver 软件，选择 New host driver project 功能。

图 20

选择 Generate .INF file。

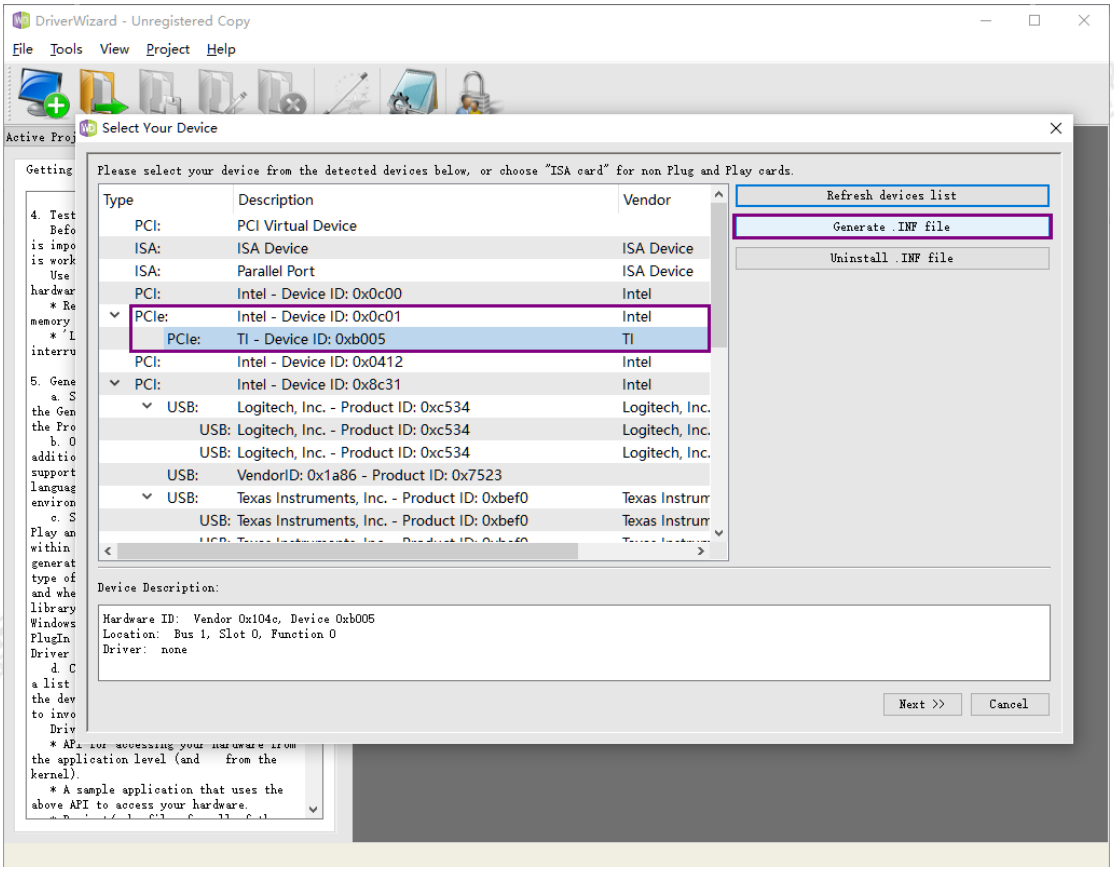

图 21

修改 INF 配置，注意不要勾选 Automatically install the INF file，点击 Next 保存文件。

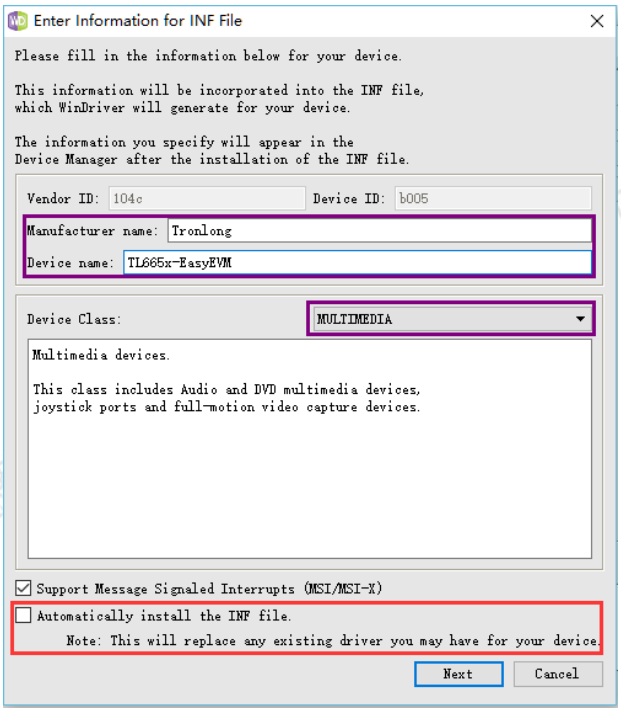

图 22

若需禁用 Windows 系统的驱动签名，不同版本的系统方法不同。在 Windows 8/8.1/10 下需要进入“高级启动配置”，再选择“疑难解答>启动设置>7 禁用驱动程序强制签名”然后重启 PC。

图 23

重启 PC 后，打开“设备管理器>其它设备>多媒体控制器>更新驱动程序软件(P)...>浏览计算机以查找驱动程序软件”。

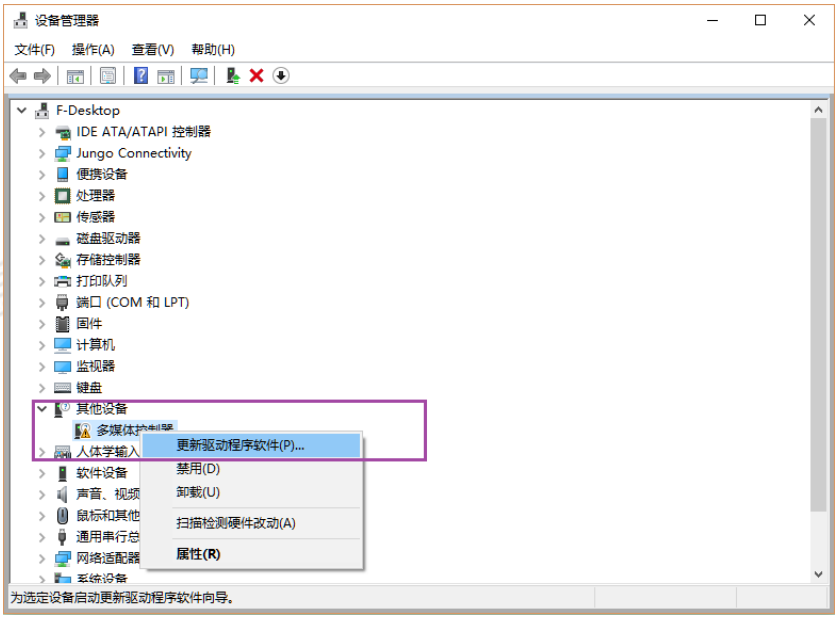

图 24

选择 INF 文件所在的目录，然后点击下一步开始安装。出现如下提示的时候，需要选择始终安装此驱动程序。

图 25

安装成功后就可以看到设备被正确识别。

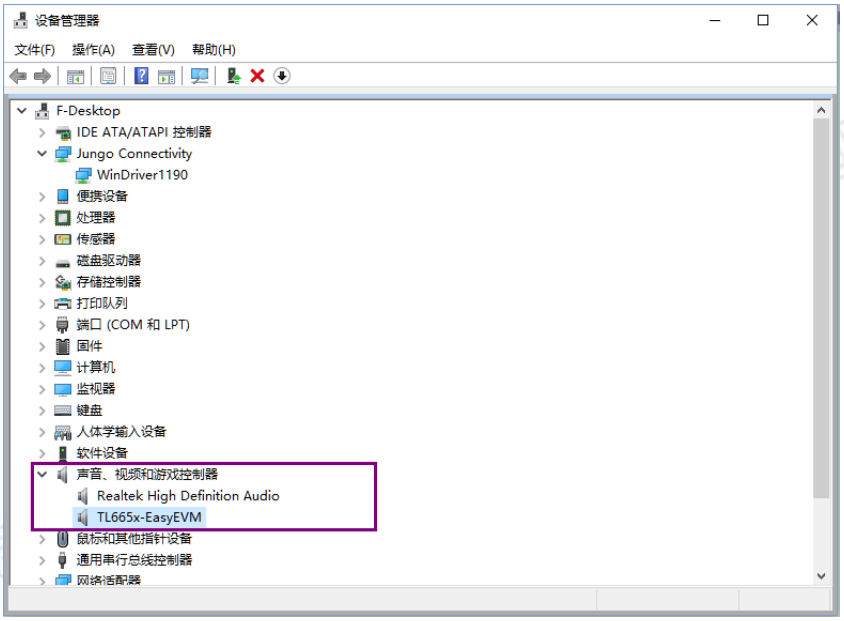

图 26

在设备属性中可用看到系统为该设备分配的资源。

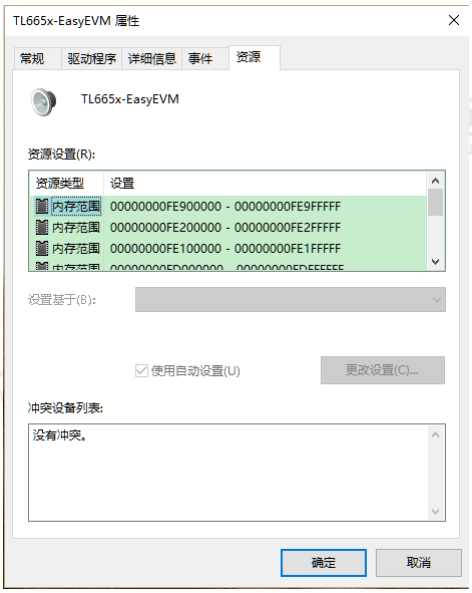

图 27

使用第三方软件还可以看到更详细的信息。总线类型 PCI Express 2.0 x2。可以看出板卡支持的 PCIe 版本为 2.0，使用两条数据通道（Lane）。

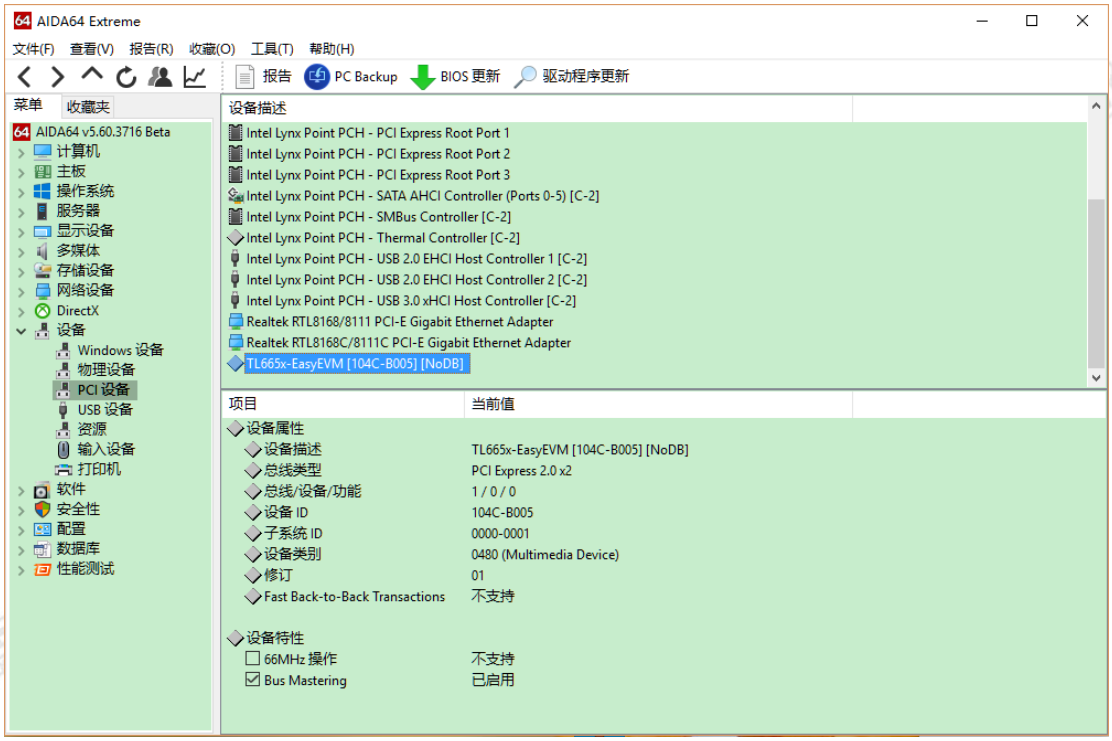

图 28

还可以看到资源分配情况。

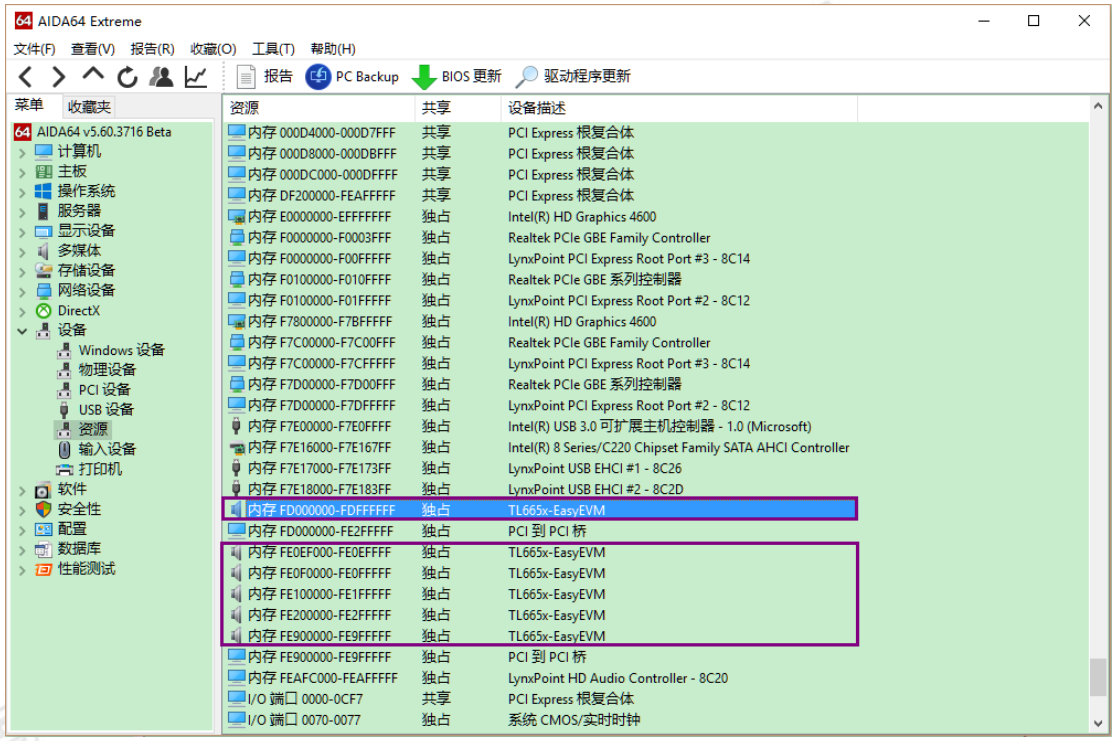

图 29

2.3. 创龙 C66x DSP TMS320C665x 寄存器配置

打开 WinDriver 软件，选择 New host driver project 功能。找到 DSP 板卡 TI-Device ID:0xb005 点击 Next 按钮。

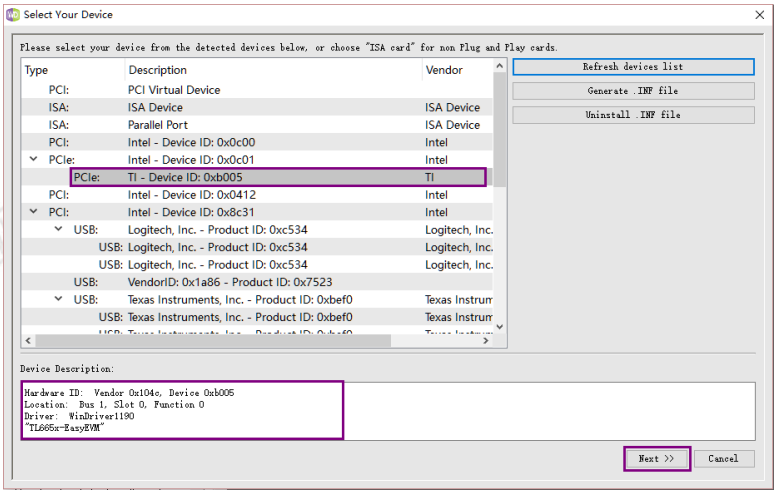

图 30

在不做任何配置的情况下，已经可以通过 PCIe 的配置空间读写 PCIe 寄存器了。

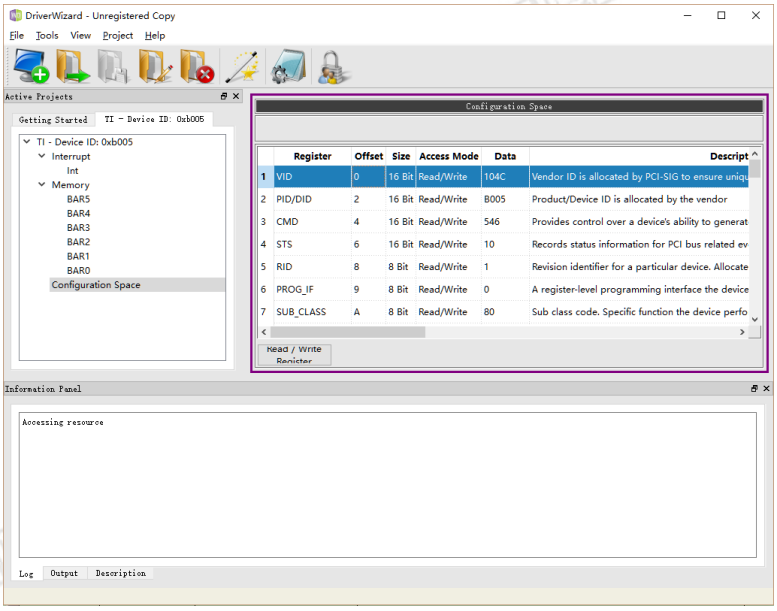

图 31

这里需要注意一下 BAR 寄存器值，之后会用到，不同 PC 这里的值不一定相同。BAR

寄存器值是 PC 在通过 PCIe 链路枚举板卡时写入的，这是 PC 为板卡在 PC 端分配的地址空间，它的大小与板卡的 BAR Mask 寄存器配置有关。 PCIe 设置为端点（End Point）模式时有六组 BAR 可用，不过在本文的测试之中，只用到四组，其中 BAR0 固定映射到 PCIe 应用寄存器。

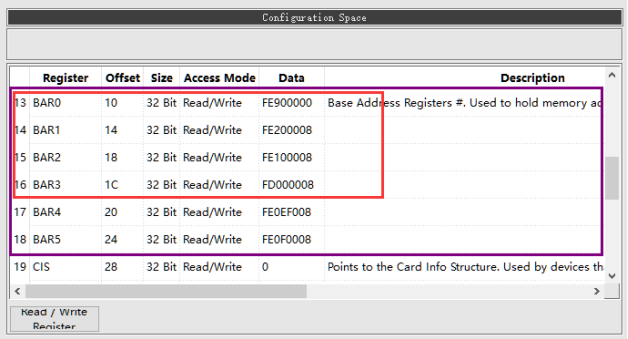

图 32

打开”Memory>BAR 0”添加如下寄存器配置，这些寄存器都是与入站（Inbound）地址翻译有关，只有地址翻译配置正确才能正常访问 DSP 内存。添加完成之后对寄存器值进行配置。

IB_BARn 值为对应的 BAR 空间[0-5]，IB_STARTn_LO 为 PC 为板卡分配的地址空间起始地址，要与 BARn 对应。IB_STARTn_HI 为 64 位地址模式时高 32 位地址，这里使用的是 32 位地址模式，所以该寄存器配置为 0 即可。IB_OFFSET0 为 DSP 内部地址空间起始地址。最后两个寄存器 SERDES_CFGn 是与 SerDes 配置有关的，这里只是为了验证通信是否正确，可以通过 CCS 查看这两个寄存器的值来验证。请参考下图进行配置。

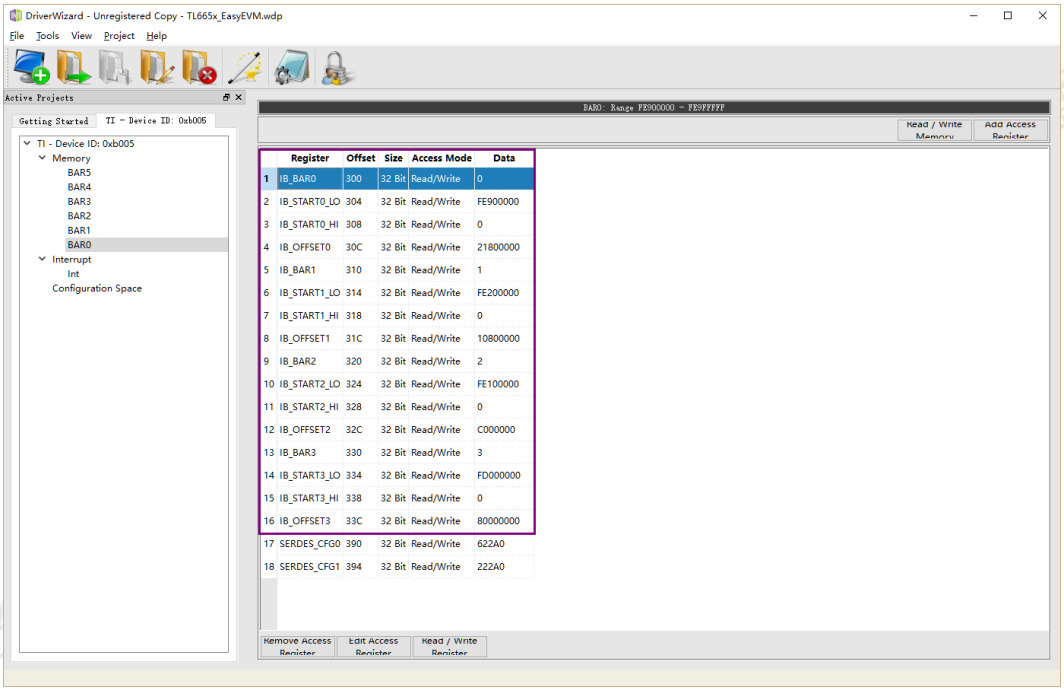

图 33

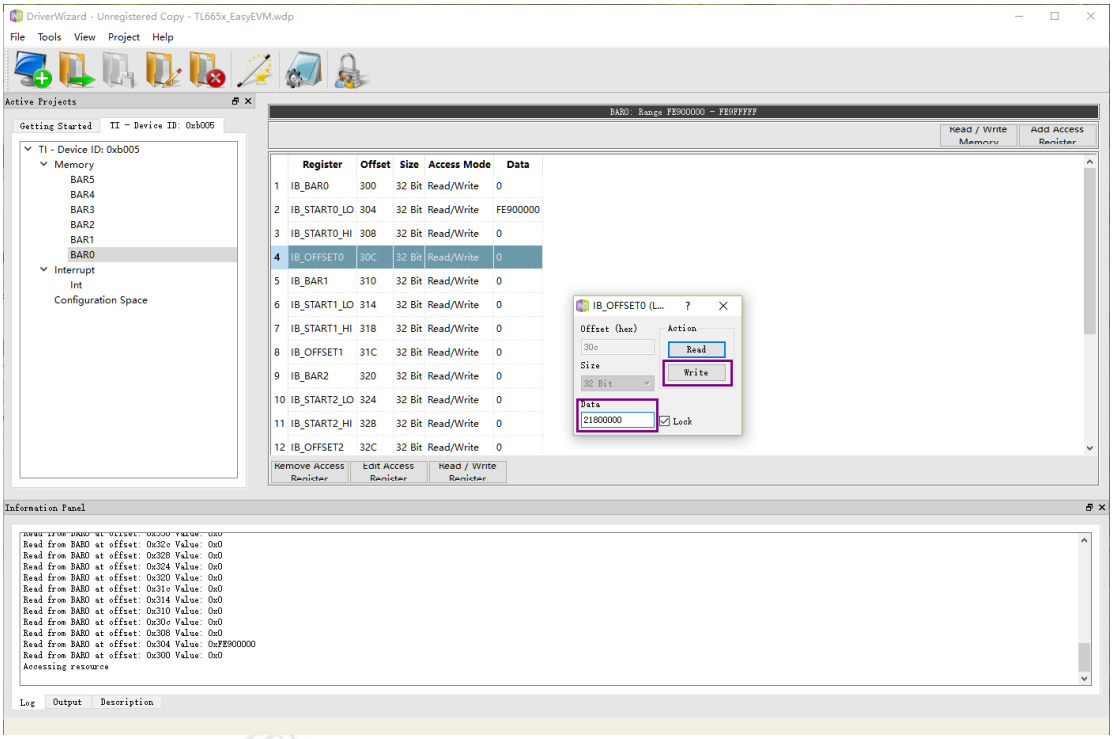

图 34

通过上述配置，完成了如下映射关系：

图 35

应用寄存器这里分配了 1MB 空间，但实际上只有 4KB 有效。L2SRAM 这里分配的是核心 0 的 L2 RAM，需要注意的是这里需要使用 0x10800000 这个全局地址，不能使用 0x00800000 本地地址，本地地址只能被 DSP 子系统内部访问，例如 CPU 核心及 IDMA。而这里 PCIe 在接收到数据后是使用 PCIe 主端口通过内部总线（TeraNet）写入数据，此过程无需 CPU 及 DMA 参与。

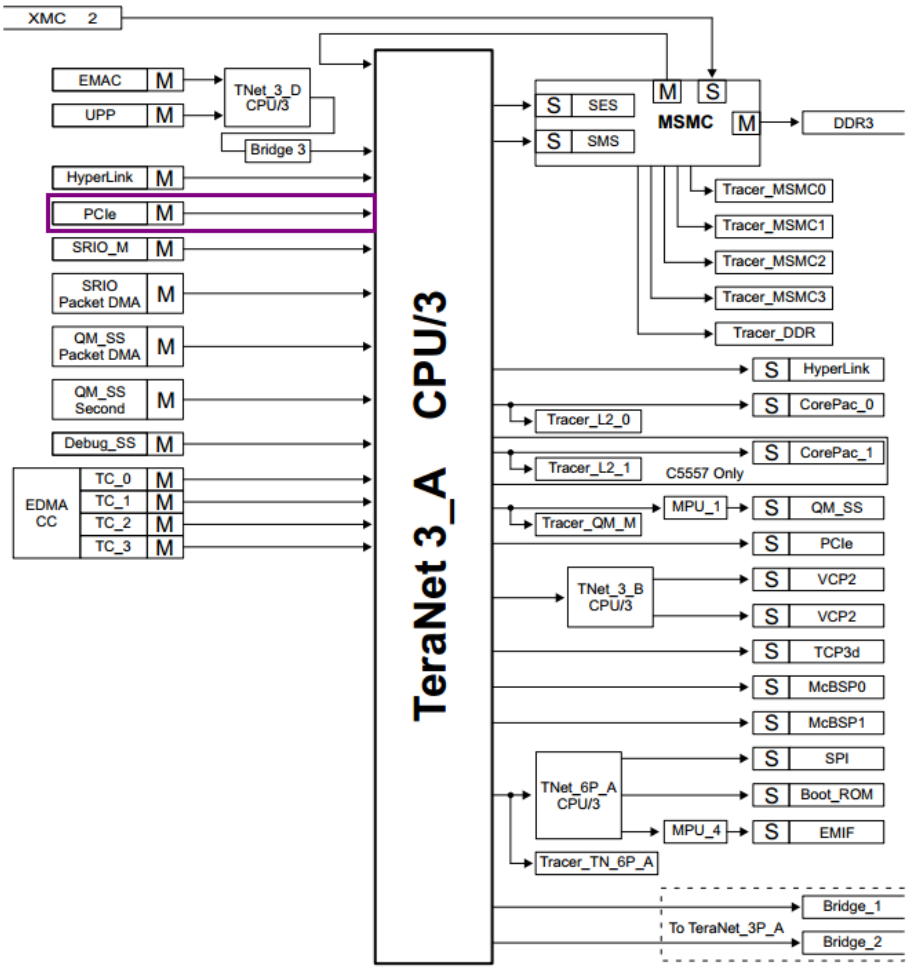

图 36

那么，为什么这些寄存器的配置要通过 RC 端来配置呢？PCIe 标准是这样规定的，当然实际开发过程中完全可以在 DSP 端进行配置。这样做的意义在于，RC 端可以动态调整寄存器的配置以便可以更加灵活的访问 EP 端内存空间。

KeyStone I 系列 DSP 内部 36 条地址线，理论可以访问 64GB 内存空间。但是，PCIe 专用保留内存空间只有 256MB。如果 RC 端要访问这么大的地址范围，一般情况下就得为 PCIe 设备保留这么大的地址空间。但现在可以通过动态调整映射寄存器来动态访问所需要的地址空间，更利于资源优化。

在配置完地址翻译之后，实际上已经可以通过 PC 访问 DSP 内存了，读写的时候是以字节为单位的。打开 CCS 集成开发环境并进入 Debug 模式，还需要打开 CCS Memory Browser 工具栏窗口，并开启实时刷新。具体步骤请参阅开发例程使用手册相关内容。

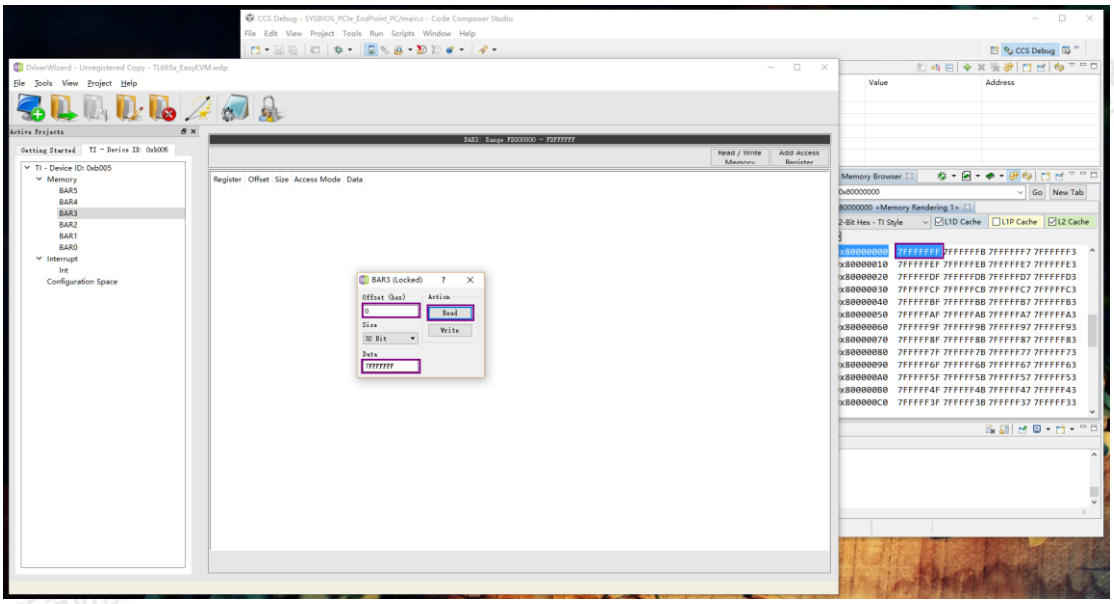

图 37 读 DDR3 内存

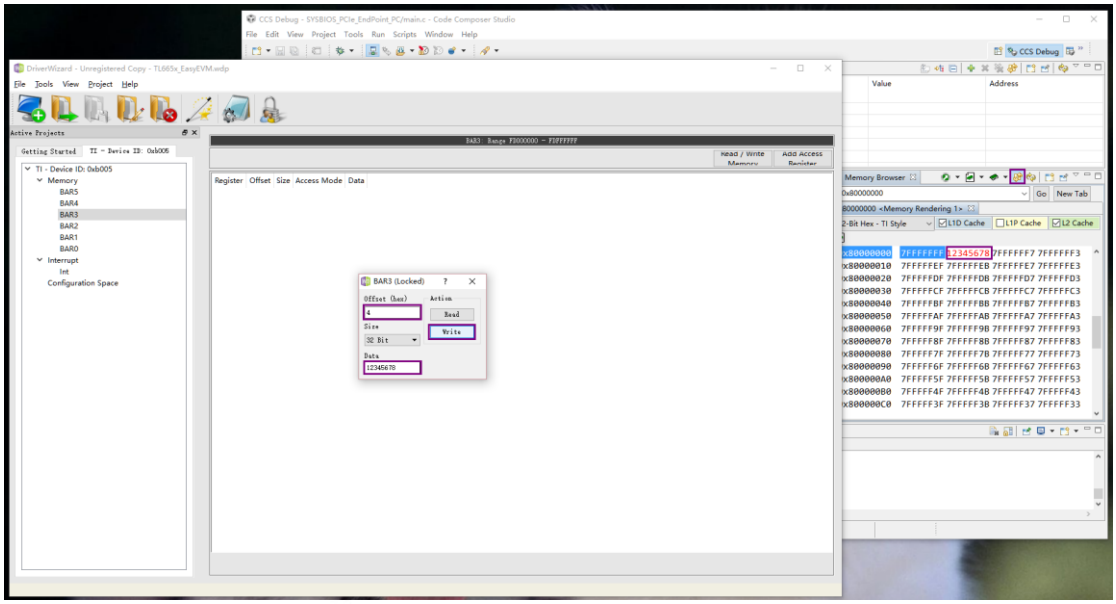

图 38 写 DDR3 内存

也可以尝试修改 IB_BAR 为其它值，以便可以访问到全部的 DSP 内存空间包括寄存器，需要注意的是寄存器对应的 BAR 需要禁用预取（Prefech）。

如果在查看内存的时候遇到数据不一致的情况（如图），说明当前 DSP 开启了缓存。不同颜色代表数据所位于的缓存类型。此时，实际写入的值可能会与 CCS 看到的数值不同，建议加载禁用缓存的程序进行测试。

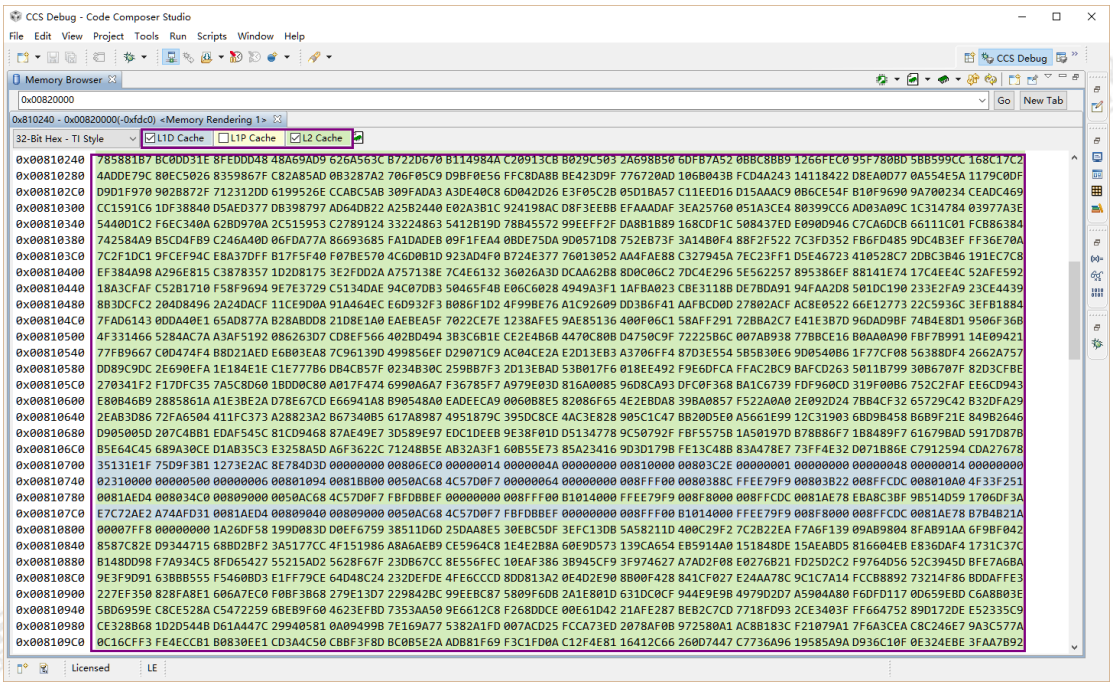

图 39

如果在查看 DDR3 内存的时候遇到数据不断变化的情况（如图），说明当前 DSP 未对 DDR3 进行正确初始化。此时，无法对 DDR3 进行读写操作。

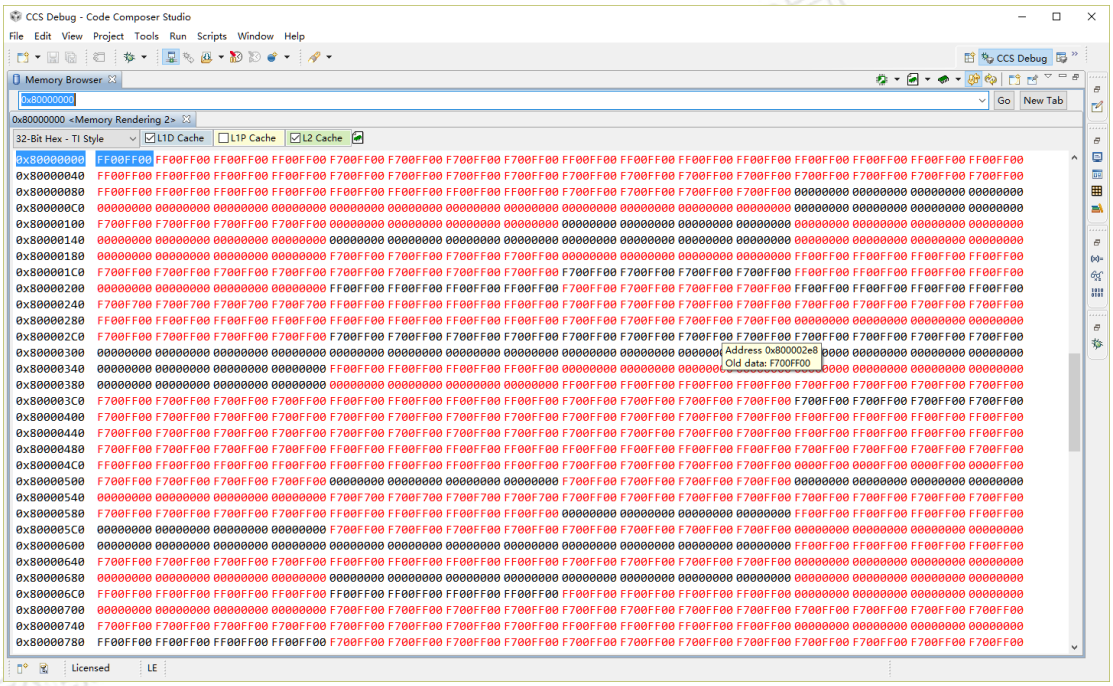

图 40

2.4. 创龙 C66x DSP TMS320C665x 生成驱动程序及应用程序

保存 WinDrive 工程，然后点击 Project 菜单项，选择 Generate Code。

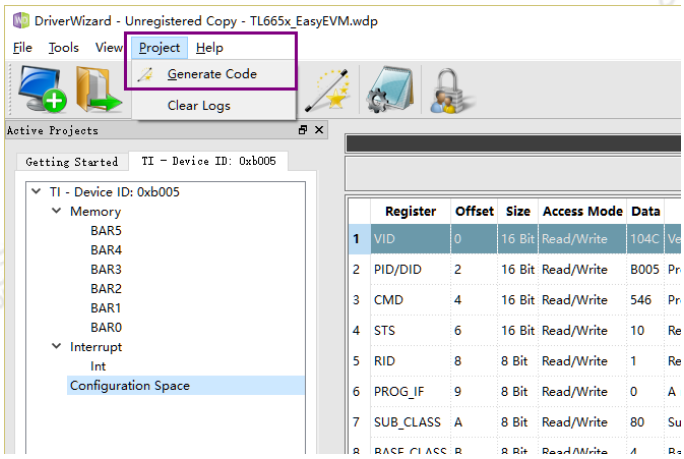

图 41

进行代码生成配置。代码语言有 C 语言或者 C#语言可选。还可以选择目标开发环境，通常情况下选择微软的 VS 集成开发环境。此外，如果选择 IDE to Invoke 代码生成成功后会自动打开相应的 IDE。

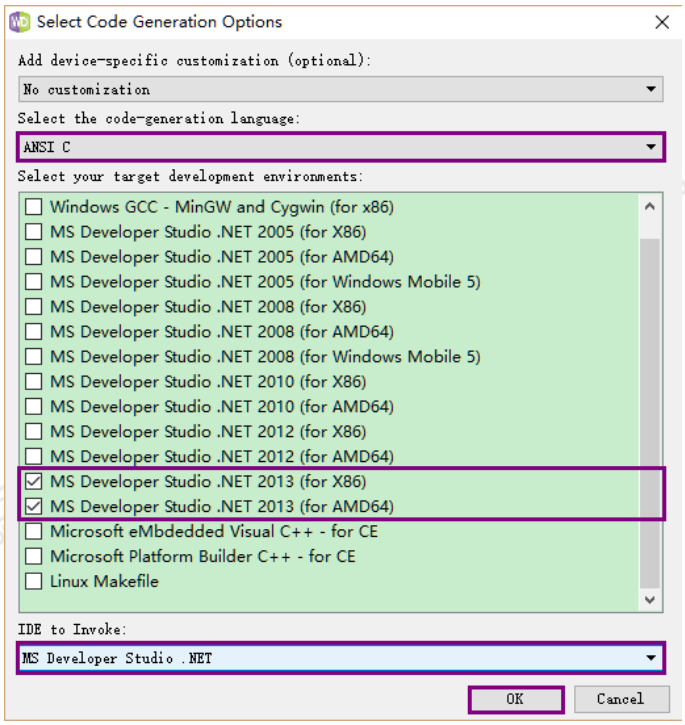

图 42

额外的代码生成配置。在这里可以根据需要勾选相应的选项。不过，如果勾选 Kernel PlugIn（for advanced users），需要在 Windows 下安装 WDK（Windows Driver Kit）组件，否则无法编译成功。

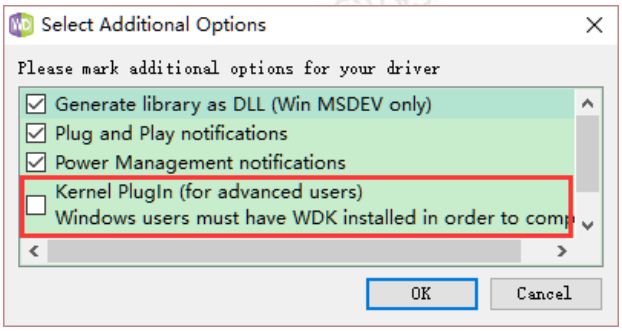

图 43

点击 OK 按钮后，可以看到生成成功的提示。

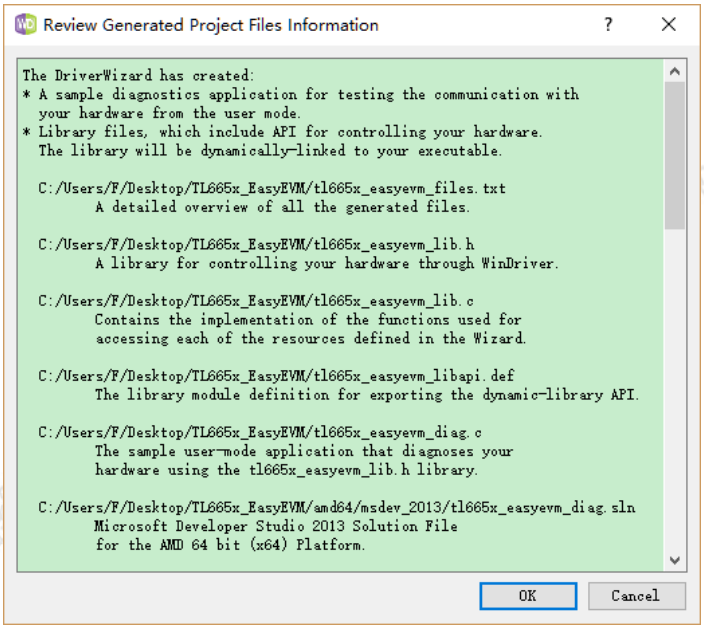

图 44

再次点击 OK 按钮，WinDriver 会自动打开 VS 集成开发环境（IDE to Invoke 选择有效），同时给出提示。提示的意思是需要把 WinDriver 工程关闭才能够在 VS 中正常调试。

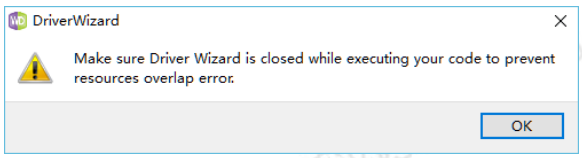

图 45

如果，使用的 VS 版本比 2013 新，比如这里使用的是 2015 就会出现需要升级编译器和库的提示，直接点击确认即可。

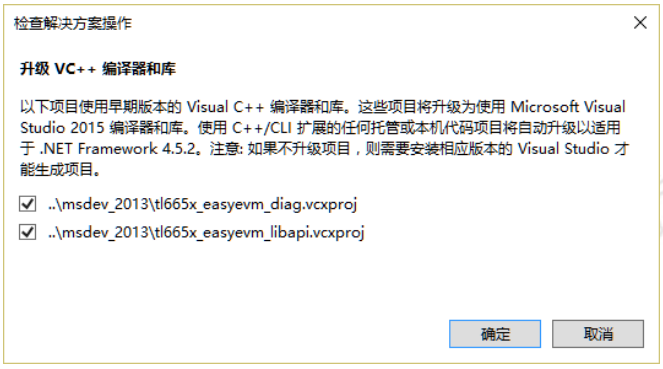

图 46

选择“生成>生成解决方案”菜单项就可以生成依赖的动态链接库（DLL）和测试应用程序。

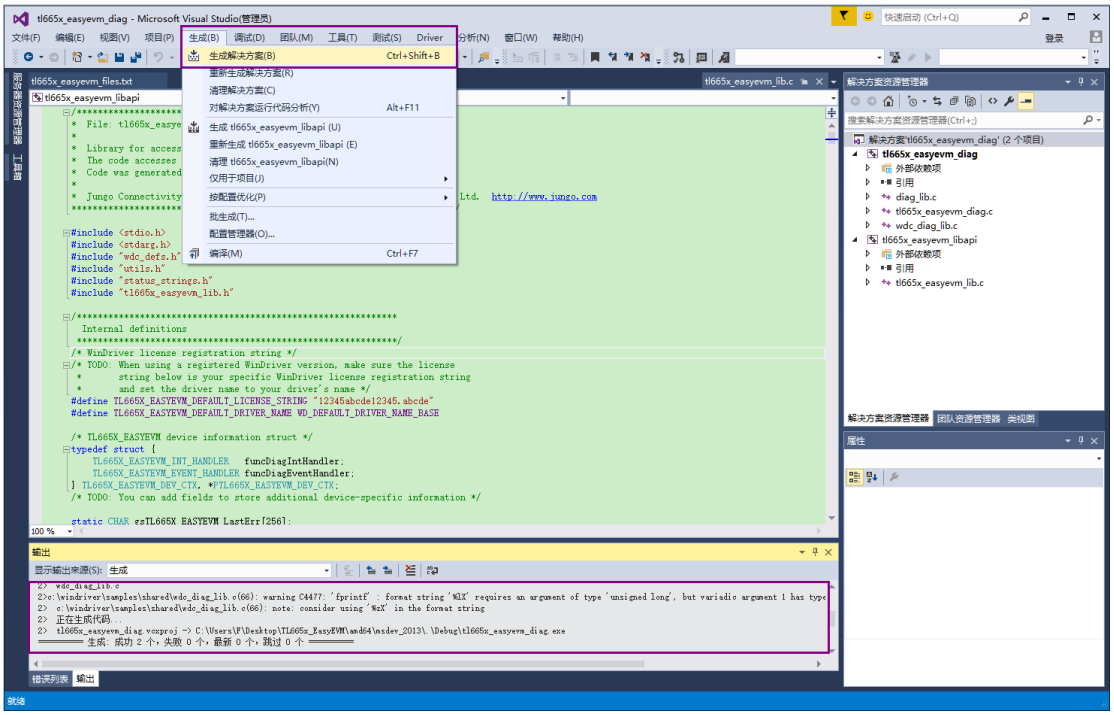

图 47

点击快捷工具栏上的“本地 Windows 调试器”按钮，进入调试模式。

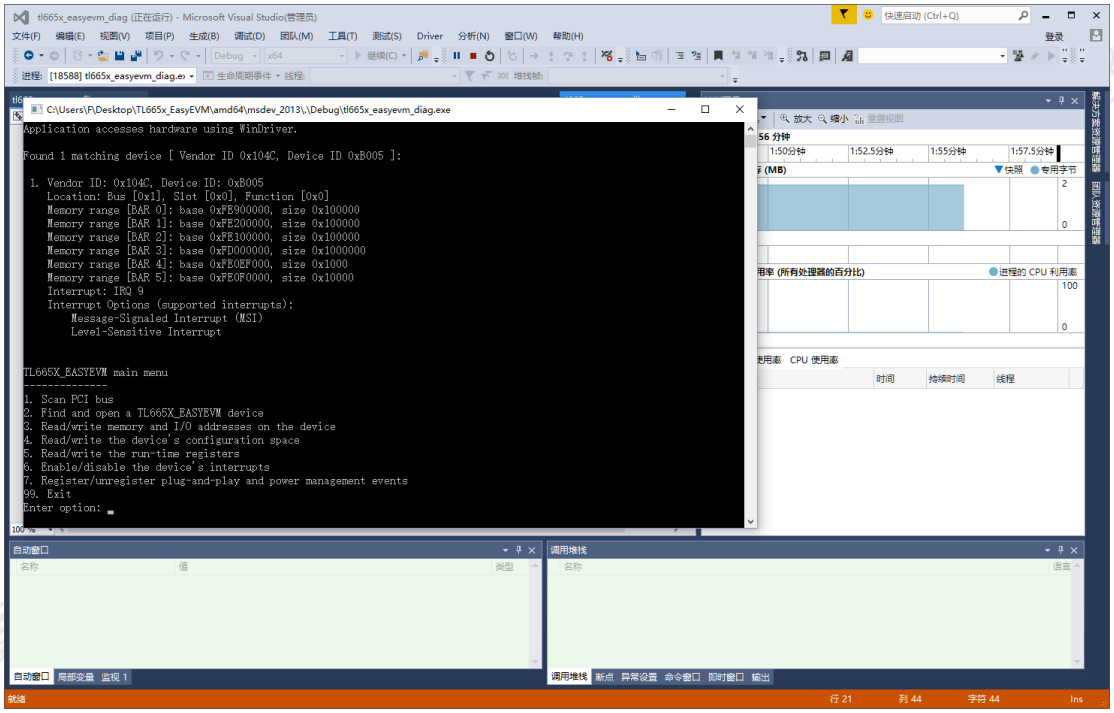

图 48

程序没有图形界面，会打开一个令行提示符窗口，可以根据提示进行操作。程序会自动检测并打开设备，前提是在 WinDriver 中关闭了相应工程，否则会报错。

例如，输入 5 然后按回车键。会打印出之前在 WinDriver 中添加的寄存器项目。

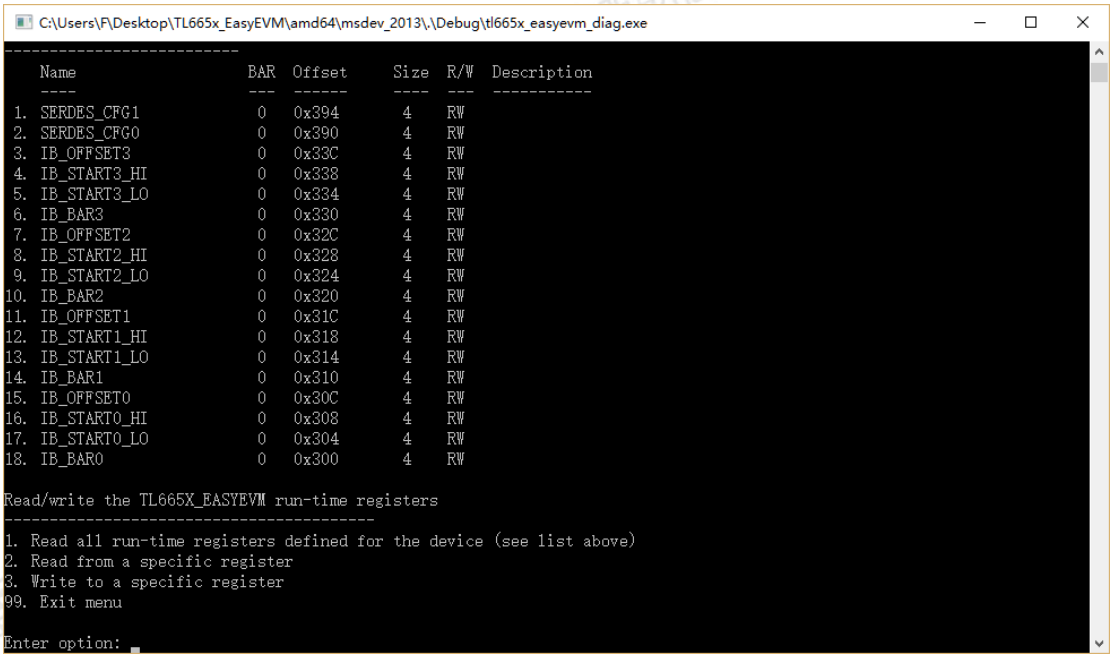

图 49

然后输入 1 并按回车键。程序会通过 PCIe 从 DSP 读取上图所列寄存器的值。

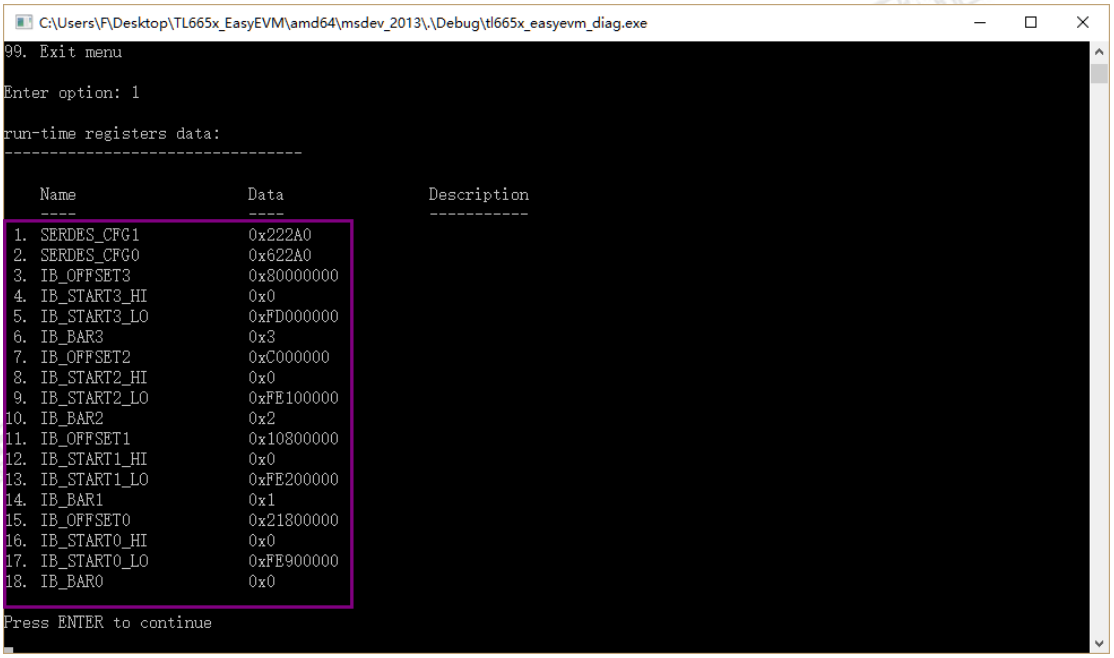

图 50

也可以通过"3. Read/write memory and I/O addresses on the device"菜单读写 DSP 寄存器或者内存。这里首先将当前地址空间选择为 BAR3，也就是之间通过 IB_BAR 寄存器配置的映射到 DSP DDR3 的 16MB（0x1000000）内存空间。

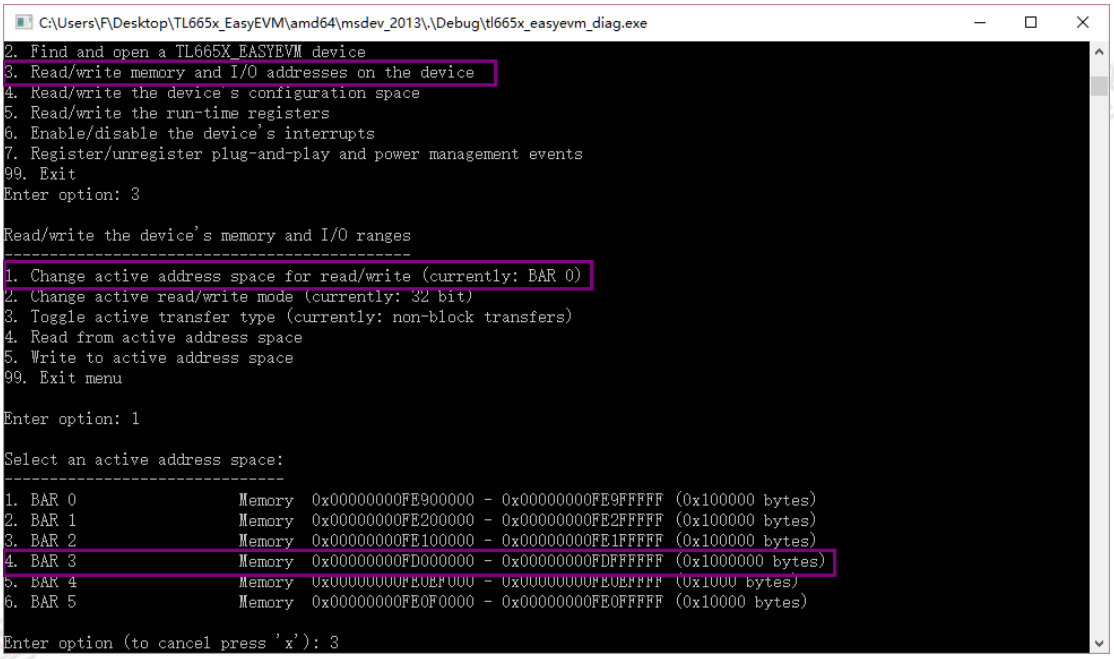

图 51

当然，也可以选择"3. Toggle active transfer type（currently: non-block transfers）"将读写方式由块（Block）方式和非块方式切换。最后，选择"4. Read from active address space"或者"5. Write to active address space"。

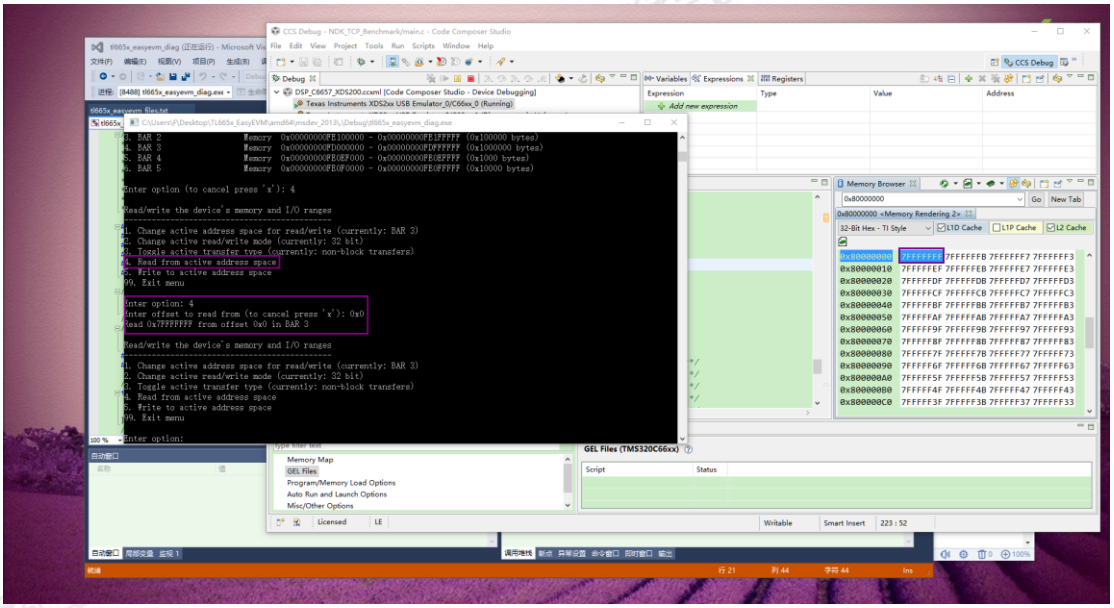

图 52 读 DDR3

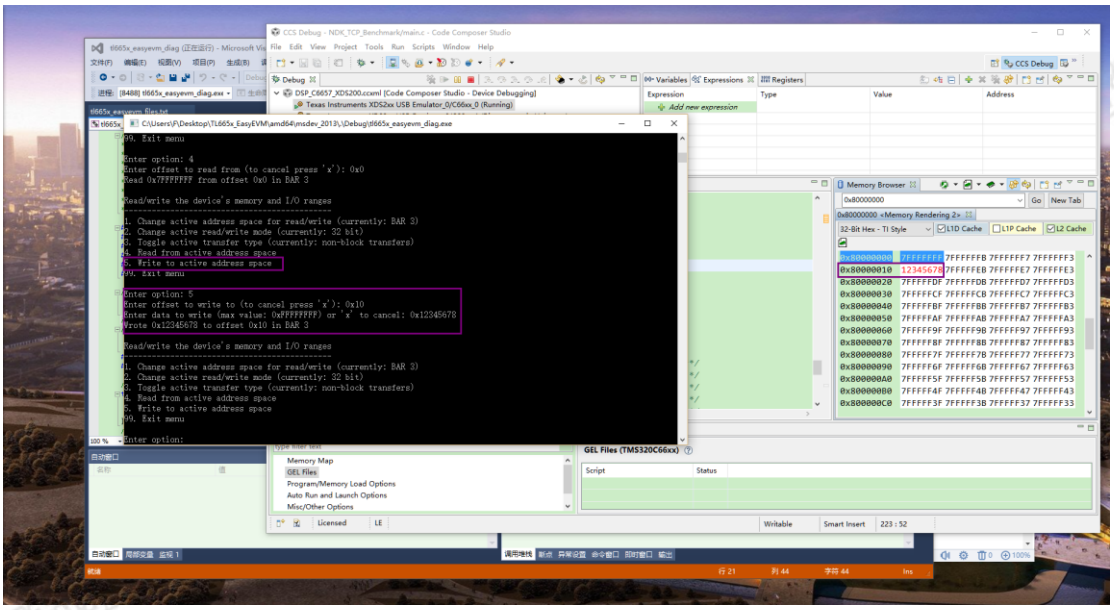

图 53 写 DDR3

WinDriver 提供的读写函数都比较简单易用，可以根据需要编写相应的应用程序。

按地址读写内存

```
void WDC_DIAG_ReadWriteAddr(WDC_DEVICE_HANDLE hDev, WDC_DIRECTION direction,
DWORD dwAddrSpace, WDC_ADDR_MODE mode)
```

按块读写内存

```
void WDC_DIAG_ReadWriteBlock(WDC_DEVICE_HANDLE hDev, WDC_DIRECTION direction,
DWORD dwAddrSpace)
```

而之前在 WinDriver 驱动向导中配置的寄存器，WinDriver 自动生成了相应的读写函数，位于 tl665x_easyevm_libapi 工程 tl665x_easyevm_lib.c 文件（不同的工程名文件名会不同）。可以直接在应用程序中调用。

```
// Function: TL665X_EASYEVM_ReadSERDES_CFG1()
//   Read from SERDES_CFG1 register.
// Parameters:
//   hDev [in] handle to the card as received from TL665X_EASYEVM_DeviceOpen().
// Return Value:
//   The value read from the register.
UINT32 TL665X_EASYEVM_ReadSERDES_CFG1 (WDC_DEVICE_HANDLE hDev)
```

```
{
    UINT32 data;

    WDC_ReadAddr32(hDev, TL665X_EASYEVM_SERDES_CFG1_SPACE, TL665X_EASYEVM_SERDES_CFG1
_OFFSET, &data);
    return data;
}

// Function: TL665X_EASYEVM_WriteSERDES_CFG1()
//   Write to SERDES_CFG1 register.
// Parameters:
//   hDev [in] handle to the card as received from TL665X_EASYEVM_DeviceOpen().
//   data [in] the data to write to the register.
// Return Value:
//   None.
void TL665X_EASYEVM_WriteSERDES_CFG1 (WDC_DEVICE_HANDLE hDev, UINT32 data)
{
    WDC_WriteAddr32(hDev, TL665X_EASYEVM_SERDES_CFG1_SPACE, TL665X_EASYEVM_SERDES_CFG
1_OFFSET, data);
}

// Function: TL665X_EASYEVM_ReadSERDES_CFG0()
//   Read from SERDES_CFG0 register.
// Parameters:
//   hDev [in] handle to the card as received from TL665X_EASYEVM_DeviceOpen().
// Return Value:
//   The value read from the register.
UINT32 TL665X_EASYEVM_ReadSERDES_CFG0 (WDC_DEVICE_HANDLE hDev)
{
```

```
    UINT32  data;

    WDC_ReadAddr32(hDev,  TL665X_EASYEVM_SERDES_CFG0_SPACE,  TL665X_EASYEVM_SERDES_CFG0
_OFFSET,  &data);
    return  data;
}
..... 略
```

2.5. 创龙 C66x DSP TMS320C665x 发布驱动程序及应用程序

■ 驱动程序位于

TL665x_EasyEVM\tl665x_easyevm_installation\redist

TL665x_EasyEVM_install.bat 为安装文件（不同的工程名文件名会不同）

■ 应用程序位于

TL665x_EasyEVM\amd64\msdev_2013\Debug（64 位）

TL665x_EasyEVM\x86\msdev_2013\Debug（32 位）

tl665x_easyevm_diag.exe 为应用程序

tl665x_easyevm_libapi.dll 为依赖的动态链接库

注意：只有购买正式授权版本的 WinDriver 生成的驱动才能够自由发布。试用版本的 WinDriver 在超过试用期后，驱动也会失效。所以，建议在使用 WinDriver 测试通过后，再基于 WDF（Windows Driver Foundation，Vista 及以后版本系统）开发驱动。微软提供的 Windows 驱动开发组件为免授权费用的。

创龙 C66x DSP TMS320C665x 更多帮助

销售邮箱：sales@tronlong.com

技术邮箱：support@tronlong.com

创龙总机：020-8998-6280

技术热线：020-3893-9734

创龙官网：www.tronlong.com

技术论坛：www.51ele.net

线上商城：https://tronlong.taobao.com