Example 1:

SD card
把　　SysCtlClockSet(SYSCTL_SYSDIV_1 | SYSCTL_USE_OSC |
　　SYSCTL_XTAL_8MHZ | SYSCTL_OSC_MAIN);
改成　SysCtlClockSet(SYSCTL_SYSDIV_4 | SYSCTL_USE_PLL |
　　SYSCTL_XTAL_6MHZ | SYSCTL_OSC_MAIN);

上面主要是 RCC 寄存器 SYSCTL_SYSDIV_4 表示 PLL 频率调到 50MHz ，SYSCTL_XTAL_6MHZ 表示外部是采用 6MHz 的晶振

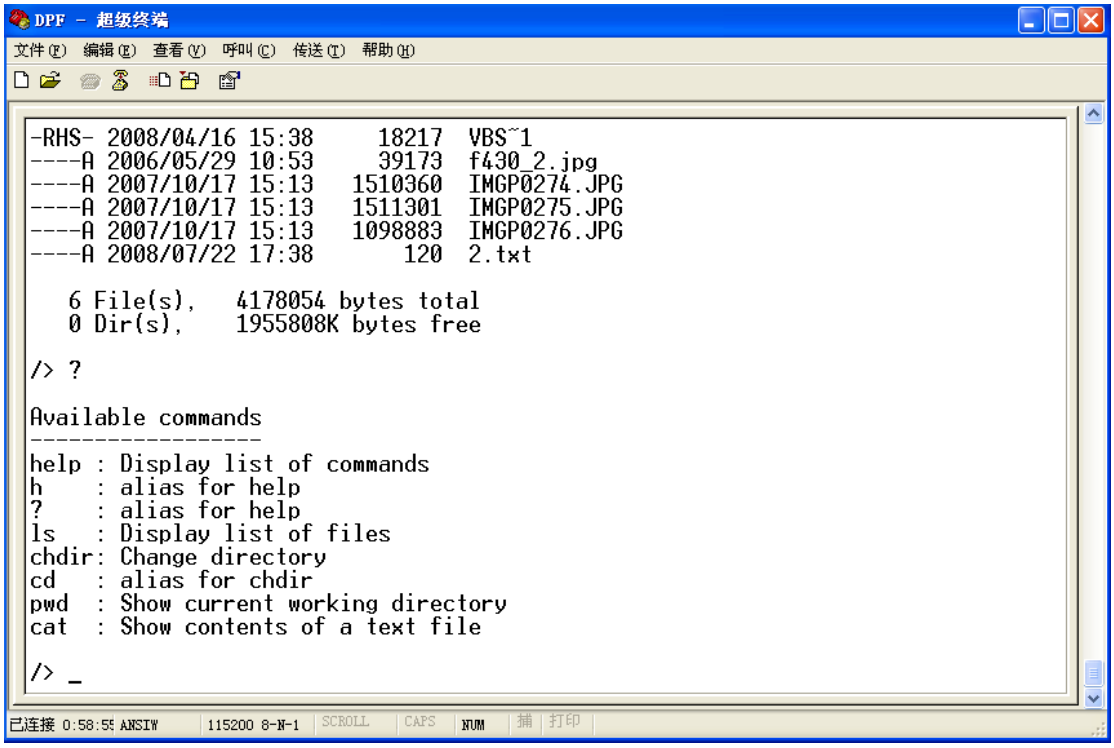
Example 2:

本例子来说明如何调整 SSI 的时钟频率，在你设置 SSI 任何寄存器之前先要关闭 SSI 。
设置完寄存器后使能 SSI 。
SSI 时钟频率的公式如下 ：
```
FSSIClk = FSysClk / (CPSDVSR * (1 + SCR))
```

主要是设置 CPSDVSR 和 SCR

```
HWREG(SSI0_BASE + SSI_O_CR1) &= 0x0B;//disable SSI
HWREG(SSI0_BASE + SSI_O_CR0) &= 0x00ff;//设置 SCR
HWREG(SSI0_BASE + SSI_O_CPSR) = 2;//设置 CPSDVSR
HWREG(SSI0_BASE + SSI_O_CR1) |= 0x040;//enable SSI
```

要使用上面这四句语句需要增加.h 头文件定义 。
#include "../../../hw_ssi.h"
按照公式算出来，SSI 的频率就是 25MHz



所有的数据通过 UART 传送，命令通过 UART 发送到开发板。开发板完成命令后把数据发送到 UART 。 主要可以列卡里面的目录文件，进下级目录，退到上级目录，把文件内容读出来传送到 UART ， 显示当前的工作目录等。

Example 3

关于使用 Luminary Micro PB7/TRST 脚用作 GPIO 的说明
当这个脚用作调试用接往调试口 JTAG 的时候，一般需要外部上拉，上拉电阻 4.7K，上拉电压 3.3V
但这个口也同样可以用作 GPIO 用，不需要接往 JTAG ，不需要上拉。
程序需要做如下修改 ：

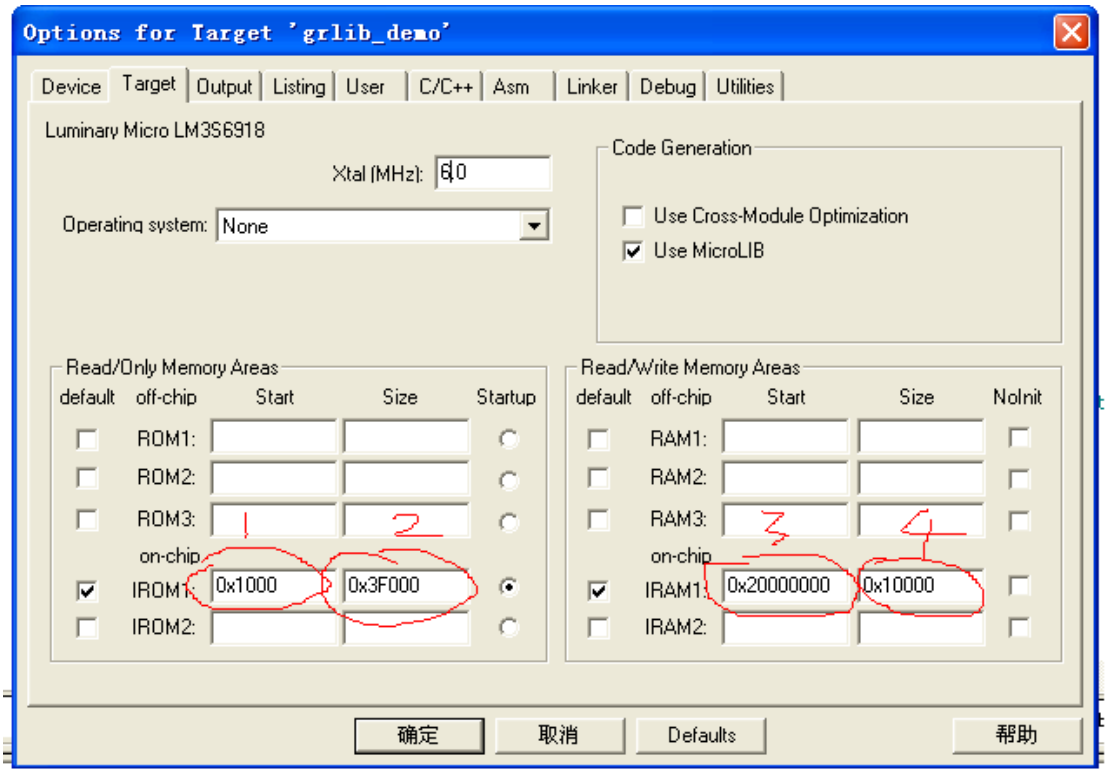
```
HWREG(GPIO_PORTB_BASE + GPIO_O_LOCK) = GPIO_LOCK_KEY;
HWREG(GPIO_PORTB_BASE + GPIO_O_CR) = 0x80;
HWREG(GPIO_PORTB_BASE + GPIO_O_AFSEL) &= 0x7f;

GPIOPinWrite(GPIO_PORTB_BASE, GPIO_PIN_7,GPIO_PIN_7);
```

在初始化 GPIO 的时候需要补上上面的几条语句 。其中最后一条是输出 PB7 高电平，你也可以改成输出低电平，就是调用 **GPIOPinWrite(GPIO_PORTB_BASE, GPIO_PIN_7,0);**
即使你有把 PB7 连到 JTAG ，也一样可以这样使用，不管你有没有接上拉。
但是测试中发现，如果不连 PB7 到 JTAG ，ULINK+KEIL 就无法下载和调试 ；但用 EKK+IAR 倒是可以 。不知道还有没有人遇到过这种情况。
这种情况终于得到解答了，原来在 Fury 的芯片内部 JTAG 的 IO 口内部是没有上拉电阻的。需要外面给上拉电阻。这点很重要，如果外面不接上拉，可能你往 PB7/TRST 送数据就得

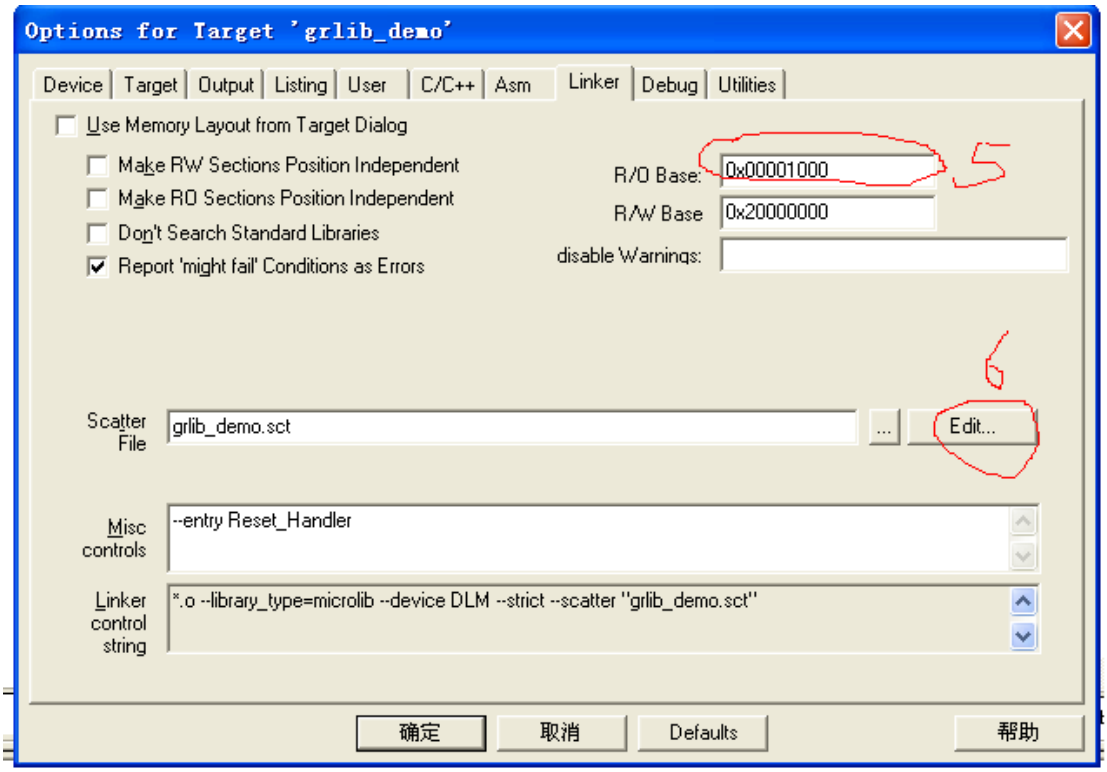
不到你想要的值 。例如你希望输出低，送的是 0 ，可能输出的是 1.312V 左右。
但是并不一定要把 PB7 连到 JTAG 调试口与 debugger 连接才可以调试。你不连也一样可以调试的 。这个情况在 errata 文件中有描述。

Example 4

Keil MDK 环境下面如何设置有 boot loader 的情况下用户程序的地址呢？



上图的 1 就是设置用户程序的开始地址。2 就是用户程序空间大小。3 就是用户 RAM 的开始地址，一般都不会变，统一为 0x20000000 开始。4 就是 RAM 的大小。
所以你的 boot loader 程序是放在 0~0x1000 地址空间的 。如果你不需要用到 boot loader ，那么就把 1 的值改为 0x00，2 的值改为 0x40000。



看上面这张页面，5 的值也就是刚刚 1 的值。指明用户程序从那个地址开始存放。如果你不需要 boot loader ，把 5 的值改为 0x00000000 就可以了。
紧接着需要点击 6，打开 sct 文件进行编辑。

```
LR_IROM 0x00001000 0x0003f000
{
    ;
    ; Specify the Execution Address of the code and the size.
    ;
    ER_IROM 0x00001000 0x0003f000
    {
        *.o (RESET, +First)
        * (InRoot$$Sections, +RO)
    }

    ;
    ; Specify the Execution Address of the data area.
    ;
    RW_IRAM 0x20000000 0x00010000
    {
        ;
        ; Uncomment the following line in order to use IntRegister().
        ;
        ;* (vtable, +First)
        * (+RW, +ZI)
    }
}
```

上面这个图就是 sct 文件的内容。说明了程序空间开始地址，空间大小。RAM 空间开始地

址,RAM 的大小。如果你不需要 boot loader 的话，把对应的值改成下面的值就可以了。
在这里我们使用的 FLASH 是 256K 的，RAM 是 64K 的，这个需要按照你的实际使用的芯片来改。我们知道 IDM 和 BLDC 使用的都是 256K 的芯片。

```
LR_IROM 0x00000000 0x00040000
{
    ;
    ; Specify the Execution Address of the code and the size.
    ;
    ER_IROM 0x00000000 0x00040000
    {
        *.o (RESET, +First)
        * (InRoot$$Sections, +RO)
    }

    ;
    ; Specify the Execution Address of the data area.
    ;
    RW_IRAM 0x20000000 0x00010000
    {
        ;
        ; Uncomment the following line in order to use IntRegister().
        ;
        ;* (vtable, +First)
        * (+RW, +ZI)
    }
}
```