Blackfin®专用管脚复用插件

作者：Jagadeesh Rayala *Rev 1-May 15,2008*

简介

本文介绍了如何使用 VisualDSP++®开发工具(5.0 或更高版本的专用管脚复用插件来配置 ADSP-BF54x 和 ADSP-BF52x 的端口寄存器。使用管脚复用插件可以自动生成 C 或汇编代码来配置端口寄存器，这样可以大大降低工作量。

管脚复用

ADSP-BF52x 和 ADSP-BF54x 系列 Blackfin 处理器具有一系列丰富的外设，通过强大的管脚复用机制，为外部应用空间提供很高的灵活性。因为具有一系列丰富的外设端口，ADSP-BF52x 处理器将这些外设信号分成四个端口组，分别为 PORT F，PORT G，PORT H，PORT J。对于 ADSP-BF54x 来说，外设信号被分成了十个端口组，从 PORT A 到 PORT J。大多数相关管脚可以被多路信号来共享。通过多路复用器控制端口的功能。PORT A 到 PORT J 的每个管脚(ADSP-BF52x 处理器的 PORT F，PORT G，PORT H)还能作为通用的输入输出管脚。任何没有被外设功能占用的管脚都可以单独地被配置成 GPIO 模式。在默认的情况下，重启以后的所有管脚都是 GPIO 模式。但是 GPIO 的输入输出驱动在默认设置中都是未激活的。

每个端口都有一套自己的映射到内存的寄存器，用于控制端口复用和 GPIO 功能。使用外设功能需要明确的使能寄存器(PORTx_FER，对于 ADSP-BF52x 此处 x 表示 F，G，H)。端口里面外设间的复用由另一个多路控制寄存器来控制(PORTx_MUX)。如果清除 PORTx_FER 的相应位，任何管脚都可以被单独地设置成 GPIO 功能来取代外设功能。如果要把管脚设置为 GPIO 输出模式，必须设置好 PORTxIO_DIR 寄存器中相应的方向位(ADSP-BF54x 处理器中对应为 PORTx_DIR_SET)。为了使用管脚的数字输入功能，需要使能 PORTxIO_INEN 寄存器(ADSP-BF54x 对应为 PORTx_INEN)。默认情况下，所有的外设管脚在重启后都是被配置为输入模式。但是，为减小功耗或者减少无用管脚外部上拉电阻的数量，GPIO 输入驱动器也可以被关闭。关于管脚复用的更多信息，请参考处理器的*硬件参考*手册[1][2]。

对于外设和 GPIO 的配置，需要对下列信息有深入的了解，端口寄存器， 所有寄存器中不同的 bit 域对应不同的信号，所有寄存器各 bit 域中占用的 bit 数，以及所有寄存器中不同数值对应的不同信号。

专用的管脚复用插件为生成端口寄存器的配置代码提供了简便方法。这个专用的管脚复用工具使你能够无需考虑内部细节就能生成所需代码。

专用管脚复用插件工具的安装

在 VisualDSP++5.0 环境下安装专用管脚复用插件:

1. 从附带的.zip 文件(EE341v01.zip)中解压提取出 ExpertPinMux.dll 文件，放置到 VisualDSP++的 system 文件夹下。如果 VisualDSP 是安装在 C 盘(默认安装路径)，复制附带文件到如下文件夹: C:\Program Files\Analog Device\VisualDSP 5.0\System
2. 输入下列命令行注册 ExpertPinMux.dll
   C:\Windows\system32\regsvr32.exe ExpertPinMux.dll

   必须在<安装路径>\System 文件夹，而不是根目录下运行 regsvr32.exe。

现在，专用管脚复用工具能在 Preferences 对话框(settings ->Preferences)中的 Plugins 页下显示了。通过 Tools 目录便可以访问专用管脚复用实体。

此插件只能工作在VisualDSP++ 5.0或更高版本的ADSP-BF52x和ADSP-BF54x环境下

图 1 显示了默认状态下的专用管脚复用窗口。默认设置下，ADSP-BF522 处理器处于选中状态，所有列表框按照选中状态显示。

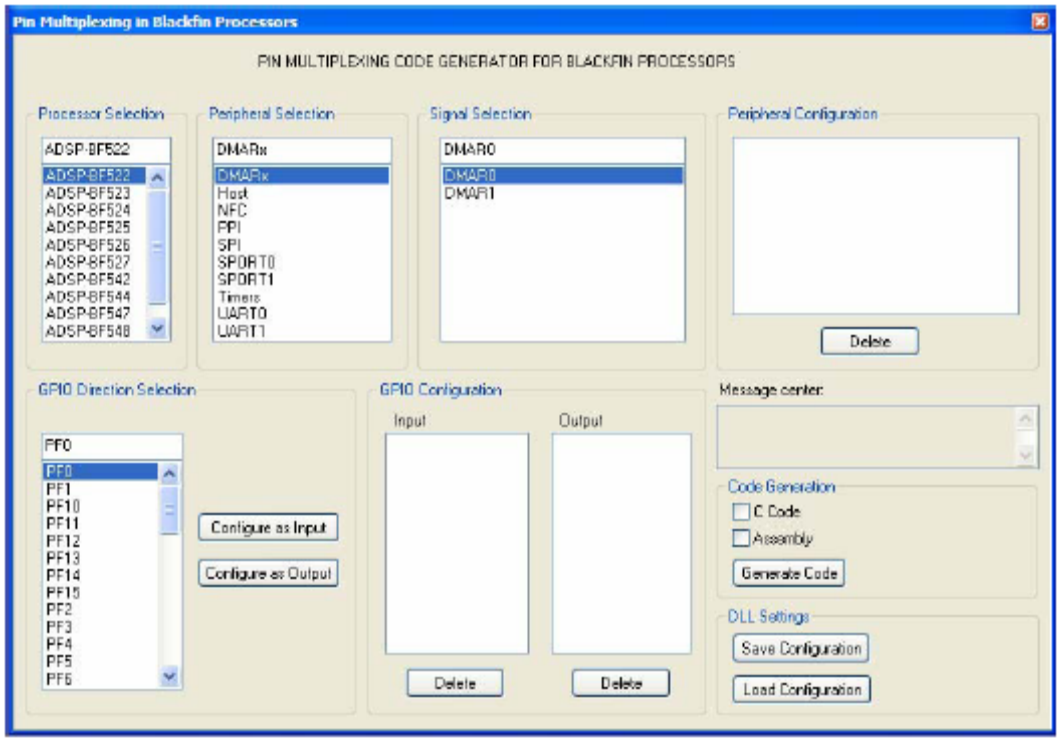

图1 专用管脚复用窗口

专用管脚复用插件的使用

生成代码：

1. 在 Processor Selection 中，选择需要生成代码的处理器型号。
2. 在 Peripheral Selection 下，选择所需的外设模块。

   当外设模块被选中，Signal Selection 列表框根据所选外设模块显示出所有相关的信号。
3. 添加一个外设信号，在 Signal Selection 列表框中选择相应信号并双击。

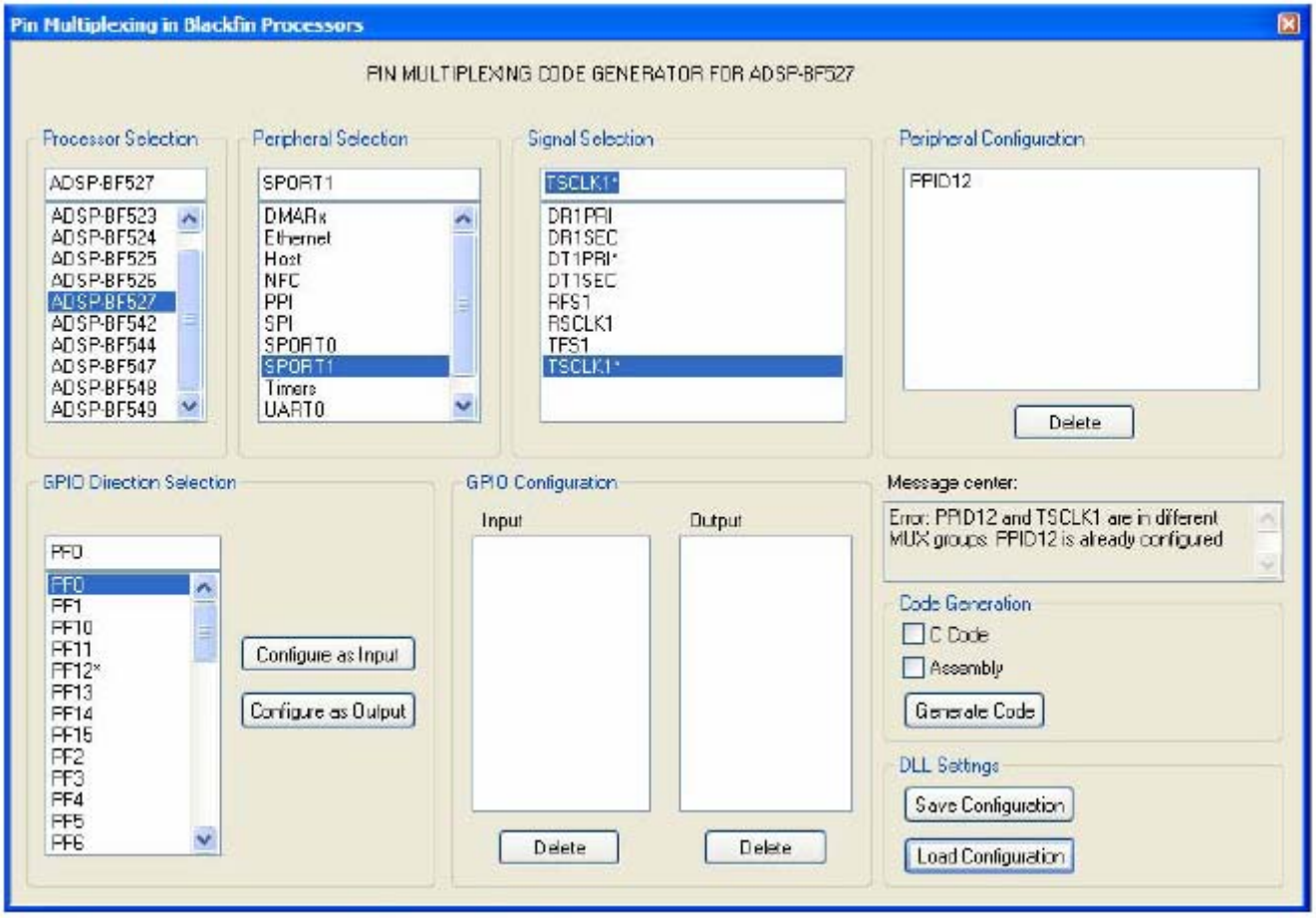

图2 配置受限的外设信号时，消息中心显示错误详情

当外设信号添加完成，Peripheral Configuration 列表框将进行相应更新。一旦外设信号添加完成，所有和该配置信号复用的信号(包括外设的和 GPIO)在此后的配置中将受到工具的限制。工具为这些受限信号提供可视的提示，在这些信号名称上附加“*”。例如，在 ADSP-BF527 上配置 PPID12。当 PPID12 与 DT1PRI，SPISEL2 以及 PF12 复用时，所有四个信号的名字上方都附加了一个“*”。注意，即使在 PPID13 没有配置的情况下，TSCLK1 和 SPISEL3 的信号名称上也会附加上 “*” 。

这是因为这两个信号(PPID12 和 PPID13)属于同一个复用组。如果一个受限信号被配置了，消息中心对话框将会显示错误消息。图 2 显示了在配置了 PPID12 的情况下试图配置 TSCLK1*时，消息中心对话框中显示的错误消息。

4. GPIO 管脚可以在 GPIO Direction Selection 列表中选择。点击 Configure as Input 或者 Configure as Output 按钮可以分别将 GPIO 管脚设置成输入或输出。

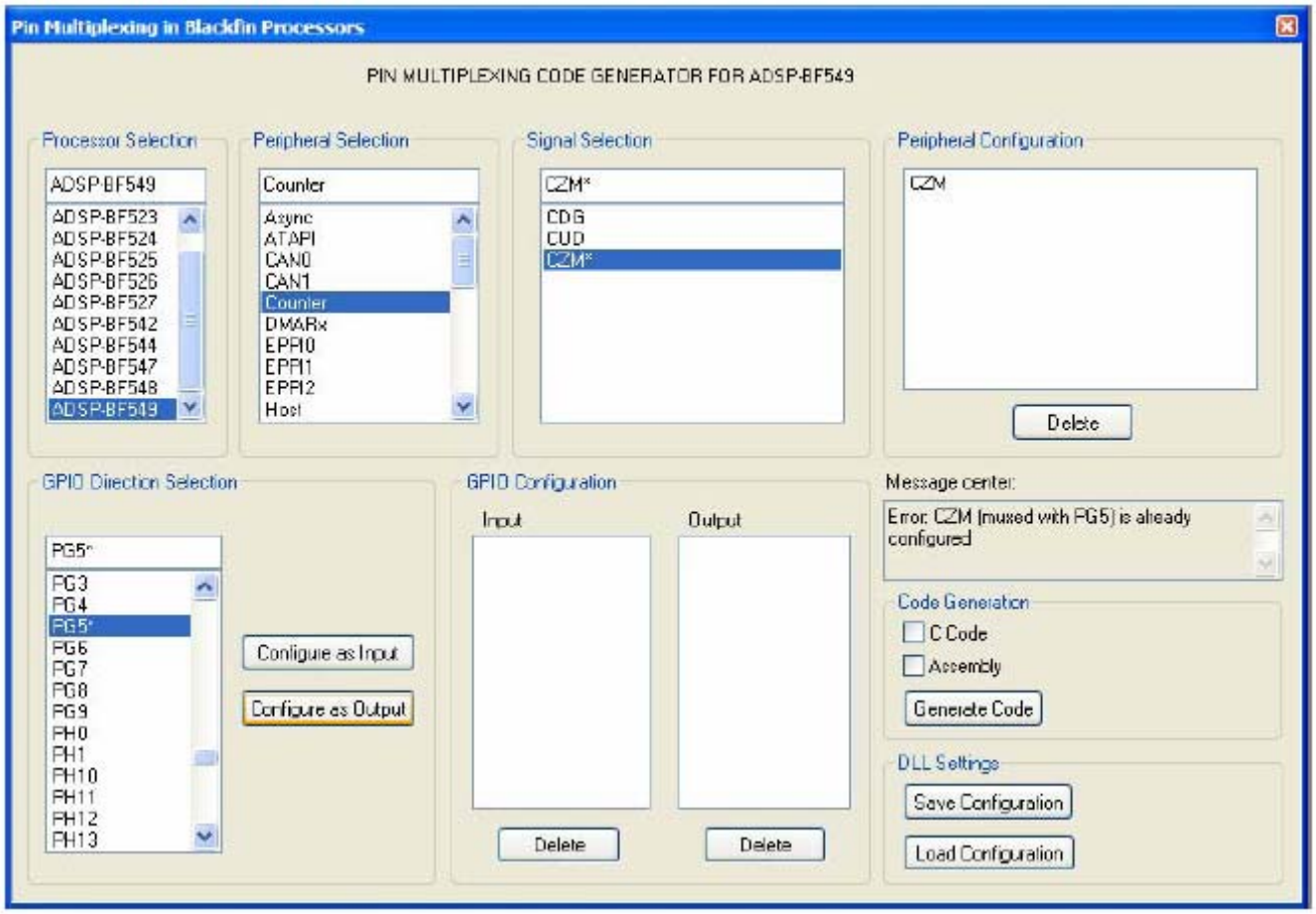

图3 消息中心显示关于配置受限的GPIO 信号的错误详情

添加了 GPIO 信号后，GPIO Configuration 列表将相应地进行更新。一旦 GPIO 信号被添加，工具就将限制所有与此 GPIO 管脚复用的其它信号的配置。如果 GPIO 管脚已经被配置为输出，工具还将限制把它配置成输出的操作，反之亦然。例如，在 ADSP-BF549 的 CZM 已经被配置的情况下，因为 CZM 与 SPI1SEL1，HOST_CE，PPI2_FS2 以及 PG5 复用，所有这五个信号名称上都将被附加“*”。如图 3，显示了当 CZM 配置完成后试图将 PG5*配置成输出时，消息中心对话框产生的错误消息。

5. 重复步骤 2 到 4，添加应用或系统设计中所有相关的外设信号或 GPIO。注意:对于外设和 GPIO 信号的配置是没有顺序限制的。

6. 选中恰当的选项，点击 Generate Code 按钮生成 C 和/或汇编代码。这个操作将打开一个对话框，要求你为生成的 C 或汇编代码选择存放路径以及文件名称。默认情况下，生成代码将以 pin.c 或 pin.asm 命名。

如果更换处理器型号，Peripheral Configuration 和 GPIO Configuration 中的数据将自动清除。同时，Peripheral Selection，Signal Selection 和 GPIO Direction Selection 框中的信号将根据所选处理器进行更新。

选中要删除的信号，点击 Delete 按钮来撤销对外设信号或者 GPIO 输入/输出管脚的配置。所有的列表对话框也将相应地更新。

专用管脚复用插件还提供以下功能:

- 保存配置。点击 Save Configuration 按钮保存以下所有信息:当前选定的处理器/外设/信号/GPIO，外设信号/GPIO 的配置，Message center 对话框内容以及每个列表和选择对话框中的当前状态。所有信息保存在扩展名为.cfg 的输出文件中(pin.cfg 是其默认名字)。
- 载入配置。点击 Load Configuration 按钮载入保存的配置(.cfg 文件)。按照提示选择一个.cfg 文件。选择好。cfg 文件后，Expert Pin Multiplexing 窗口将更新，显示.cfg 文件中的内容。此时可以为每个新设计添加或删除外设/GPIO 信号，重新生成代码。

代码生成

本节将通过一个例子来说明生成代码的过程。假设一个基于 ADSP-BF522 的应用，按照以下要求配置外设信号以及 GPIO：

- 外设
  - SPORT(DR1PRI，DT1PRT，RFS1，RSCLK1，TFS1 和 TSCLK1)
  - UART0(UART0RX 和 UART0TX)
  - HOST(HOST_ACK，HOST_ADDR，HOST_CE，HOST_RD，HOST_WR 以及 HOST_Dx:x=0 到 15)
  - 从模式下的 SPI(MISO，MOSI，SCK 和 SPISS)
- GPIO
  - 输入(PF3 和 PF5)
  - 输出(PF1 和 PG10)

图 4 示范了按照上述配置生成控制端口寄存器的 C 和汇编代码过程。生成的汇编和 C 代码分别在附录的列表 1 和列表 2 中。将生成的 C/汇编代码添加到 VisualDSP++工程中。C 语言工程，主函数需要调用 InitPorts()函数。汇编工程中，主程序调用_InitPorts 子程序。

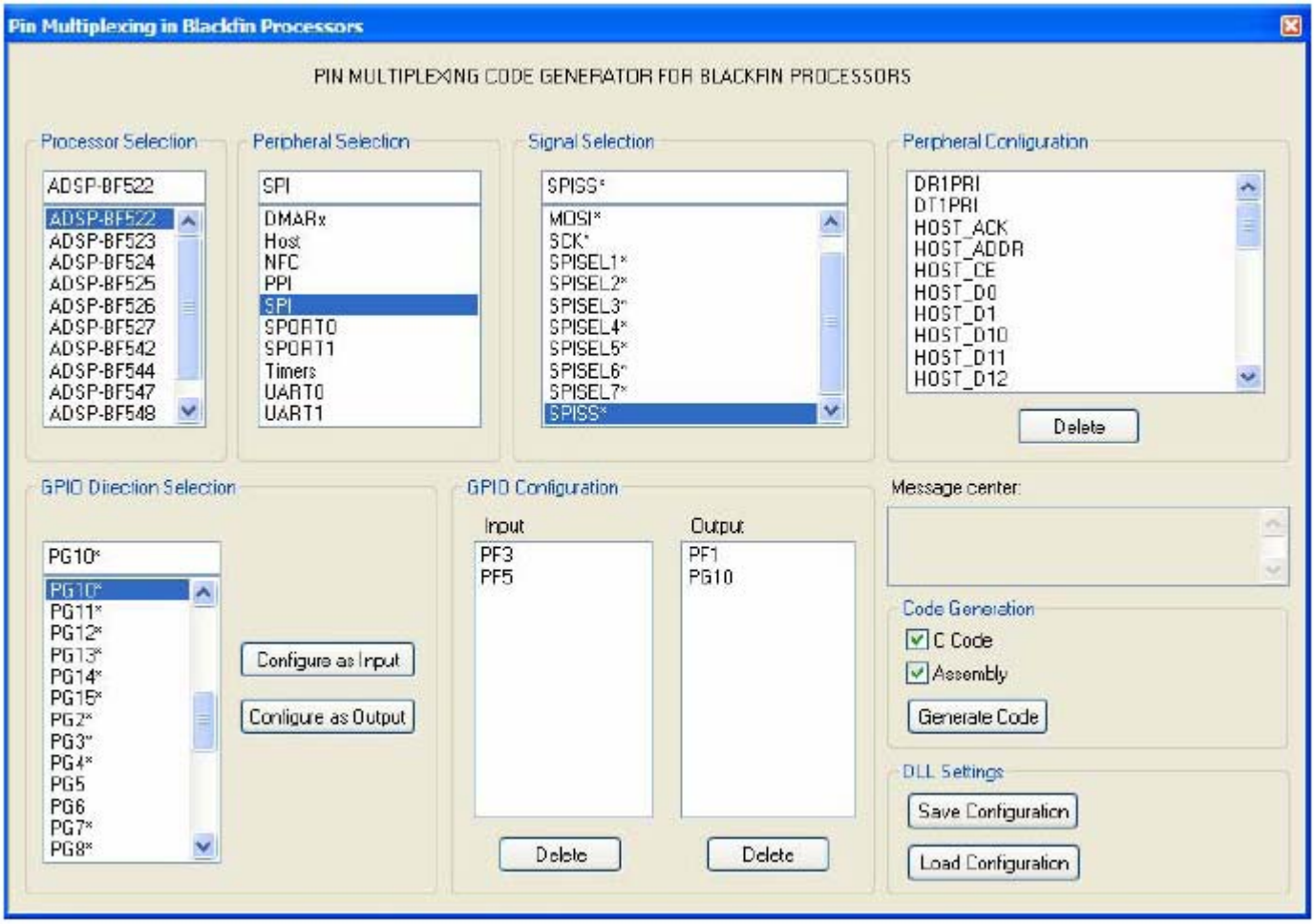

图4 生成配置ADSP-BF522处理器端口寄存器的C和汇编代码

附录

Pin.asm

```
/* Assembly code generated to configure PORTs and GPIOs.

Peripheral pins selected: DR1PRI, DT1PRI, HOST_ACK, HOST_ADDR, HOST_CE, HOST_D0, HOST_D1, HOST_D10,
HOST_D11, HOST_D12, HOST_D13, HOST_D14, HOST_D15, HOST_D2, HOST_D3, HOST_D4, HOST_D5, HOST_D6,
HOST_D7, HOST_D8, HOST_D9, HOST_RD, HOST_WR, MISO, MOSI, RFS1, RSCLK1, SCK, SPISS, TFS1, TSCLK1,
UART0RX, UART0TX

        GPIOs configured as Inputs: PF3, PF5

        GPIOs configured as Outputs: PF1, PG10
*/

#include <defBF52x_base.h>
// This function will setup the Port Control Registers
.section program ;
.global _InitPorts ;

_InitPorts :
        // First save registers
        [--sp] = rets;
        [--sp] = p0;
        [--sp] = r0;

        // PORTx_MUX registers
        p0.l = lo(PORTF_MUX);
        p0.h = hi(PORTF_MUX);
        r0.l = 0x154;
        w[p0] = r0;

        p0.l = lo(PORTG_MUX);
        p0.h = hi(PORTG_MUX);
        r0.l = 0x2820;
        w[p0] = r0;

        p0.l = lo(PORTH_MUX);
        p0.h = hi(PORTH_MUX);
        r0.l = 0x2a;
        w[p0] = r0;

        // PORTx_FER registers
        p0.l = lo(PORTF_FER);
        p0.h = hi(PORTF_FER);
        r0.l = 0x3f00;
        w[p0] = r0;

        p0.l = lo(PORTG_FER);
        p0.h = hi(PORTG_FER);
        r0.l = 0xf99e;
```

```
    w[p0] = r0;

    p0.l = lo(PORTH_FER);
    p0.h = hi(PORTH_FER);
    r0.l = 0xffff;
    w[p0] = r0;

    // PORTxIO_DIR registers
    p0.l = lo(PORTFIO_DIR);
    p0.h = hi(PORTFIO_DIR);
    r0.l = 0x2;
    w[p0] = r0;

    p0.l = lo(PORTGIO_DIR);
    p0.h = hi(PORTGIO_DIR);
    r0.l = 0x400;
    w[p0] = r0;

    p0.l = lo(PORTHIO_DIR);
    p0.h = hi(PORTHIO_DIR);
    r0.l = 0x0;
    w[p0] = r0;

    // PORTxIO_INEN registers
    p0.l = lo(PORTFIO_INEN);
    p0.h = hi(PORTFIO_INEN);
    r0.l = 0x28;
    w[p0] = r0;

    p0.l = lo(PORTGIO_INEN);
    p0.h = hi(PORTGIO_INEN);
    r0.l = 0x0;
    w[p0] = r0;

    p0.l = lo(PORTHIO_INEN);
    p0.h = hi(PORTHIO_INEN);
    r0.l = 0x0;
    w[p0] = r0;

    // Restore the registers
    r0 = [sp++];
    p0 = [sp++];
    rets = [sp++];

    // Return back from the subroutine
    rts;
```

列表 1 pin.asm

pin.c

```
/*
        C code generated to configure PORTs and GPIOs.

Peripheral pins selected: DR1PRI, DT1PRI, HOST_ACK, HOST_ADDR, HOST_CE, HOST_D0, HOST_D1, HOST_D10,
HOST_D11, HOST_D12, HOST_D13, HOST_D14, HOST_D15, HOST_D2, HOST_D3, HOST_D4, HOST_D5, HOST_D6,
HOST_D7, HOST_D8, HOST_D9, HOST_RD, HOST_WR, MISO, MOSI, RFS1, RSCLK1, SCK, SPISS, TFS1, TSCLK1,
UART0RX, UART0TX

        GPIOs configured as Inputs: PF3, PF5

        GPIOs configured as Outputs: PF1, PG10
*/

#include <cdefBF52x_base.h>

void InitPorts();

// This function will setup the Port Control Registers void InitPorts()
{
        // First Set PORTx_MUX registers
        *pPORTF_MUX = 0x154;
        *pPORTG_MUX = 0x2820;
        *pPORTH_MUX = 0x2a;

        // Set PORTx_FER registers
        *pPORTF_FER = 0x3f00;
        *pPORTG_FER = 0xf99e;
        *pPORTH_FER = 0xffff;

        // Set PORTxIO_DIR registers
        *pPORTFIO_DIR = 0x2;
        *pPORTGIO_DIR = 0x400;
        *pPORTHIO_DIR = 0x0;

        // Set PORTxIO_INEN registers
        *pPORTFIO_INEN = 0x28;
        *pPORTGIO_INEN = 0x0;
        *pPORTHIO_INEN = 0x0;
}
```

列表 2 pin.c

参考文献

[1] *ADSP-BF52x Blackfin Processor Hardware Reference (Volume 1 of 2).* Rev 0.3 (Preliminary), September 2007. Analog Devices, Inc.

[2] *ADSP-BF54x Blackfin® Processor Hardware Reference.* Rev 0.2, September 2007. Analog Devices, Inc.

[3] *ADSP-BF522/523/524/525/526/527 Blackfin® Embedded Processor Preliminary Data Sheet.* Rev PrD, December 2007. Analog Devices, Inc.

[4] *ADSP-BF542/BF544/BF547/BF548/BF549 Blackfin® Embedded Processor Preliminary Data Sheet.* Rev PrG, December 2007. Analog Devices, Inc.

文档记录

Revision	Description
Rev 1 – May 15, 2008 *R. Jagadeesh*	Initial release.