1 简介

本文是基于 TL-MCFW-RDK 的 SYSLINK 双核通信例程演示，请确认 Ubuntu 下安装了交叉编译工具链、TL-MCFW-RDK 开发包，按照《搭建 Linux 开发环境》搭建环境。

2 SYSLINK DSP 算法例程

2.1 a8_syslink_dsp_fft — 双核快速傅里叶变换

表 1

开发板	是否支持本例程
TL8148-EasyEVM	支持
TL8148-EVM	支持
TL8127-EasyEVM	支持
TL8127-EVM	支持

2.1.1 例程说明

程序演示了在 MCFW 框架的 SYSLINK 下，ARM 协调 DSP 进行快速傅里叶变换的功能。此例程是一个双核例程，除了生成运行于 ARM Linux 的可执行程序外，还要编译出新的 DSP 固件，用来加载运行。

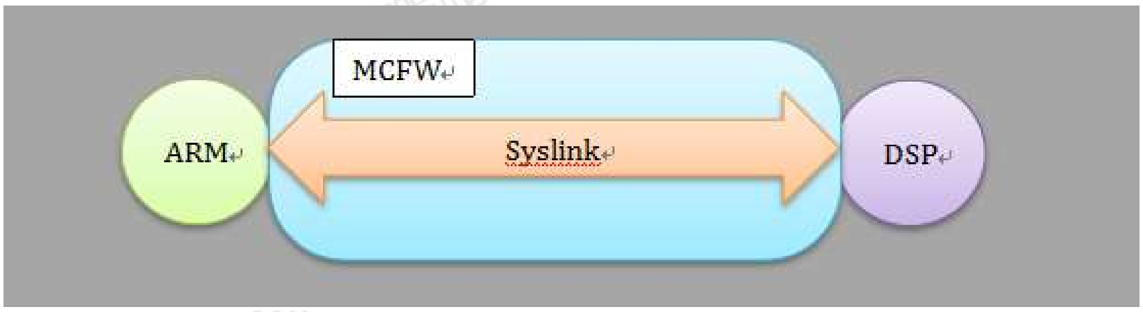


图 1

ARM（Cortex-A8）和 DSP 之间通过 MCFW 的 chain 来通信，数据在一个双向队列中往返 ARM（Cortex-A8）和 DSP，所以 ARM（Cortex-A8）能轻松获取 DSP 处理后的数据。

需要特别注意的是，本例程的 chain 中的 AppLink 是新增的自定义 Link，而非 MCFW

本来就包含。可以在此 Link 中很方便地加入数据处理代码。

文件结构如下：

```
├── dsp
|   ├── app_task.c
|   ├── DSPLib
|   |   ├── include
|   |   └── lib
|   ├── main_c6xdsp.c
|   ├── makefile
|   ├── makefile.mk
|   ├── MathLib
|   |   ├── include
|   |   └── lib
|   ├── resource_sync.c
|   └── resource_sync.h
├── host
|   ├── main.c
|   ├── makefile
|   ├── resource_sync.c
|   └── resource_sync.h
├── makefile
├── Rules.make
└── shared
    ├── protocol.h
    └── sys_config.h
```

主要分为 DSP 和 Host 两部分，分别是生成 DSP firmware 和 ARM binary。两个程序有各自的编译规则，由上层的 makefile 统一控制，分别编译。shared 目录下是两个程序都用到的共同定义。

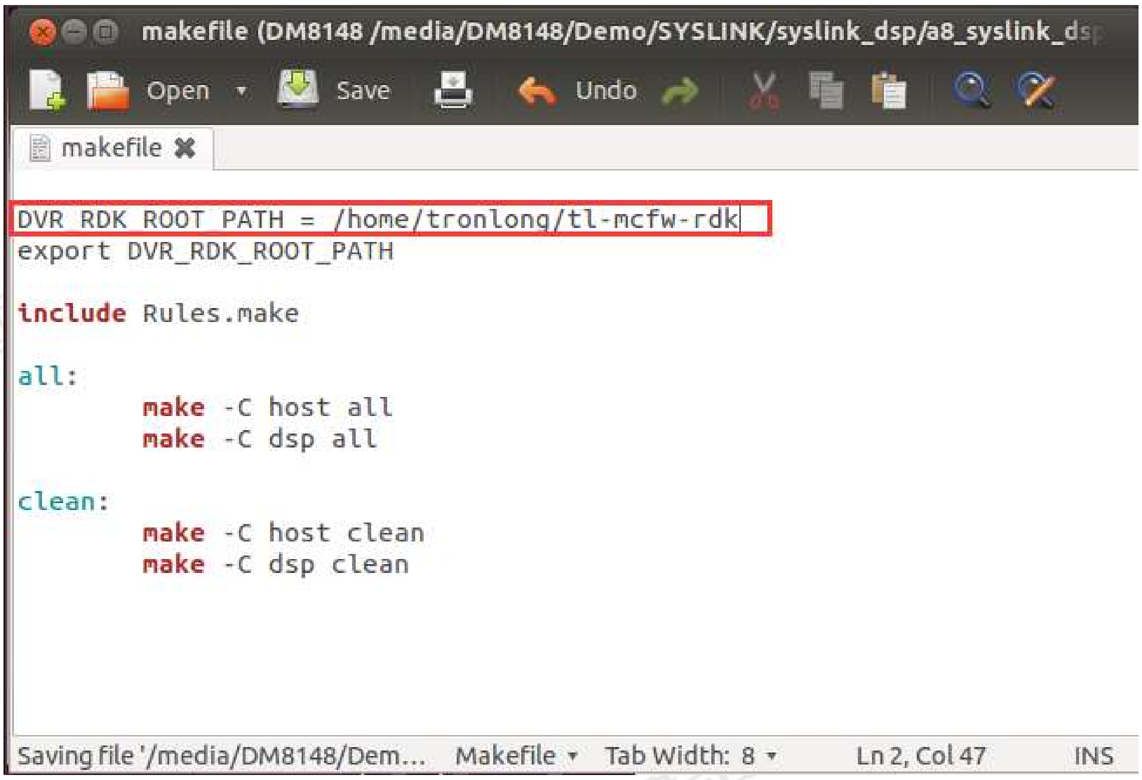
2.1.2 编译

例程代码路径：光盘资料\Demo\SYSLINK\syslink_dsp\a8_syslink_dsp_fft

将例程源码拷贝到 Ubuntu 下，进入 src 目录，修改 makefile：

DVR_RDK_ROOT_PATH：tl-mcfw-rdk 开发包安装路径；

Host# gedit makefile


makefile (DM8148 /media/DM8148/Demo/SYSLINK/syslink_dsp/a8_syslink_dsp
Open
Save
Undo
makefile

```
DVR_RDK_ROOT_PATH = /home/tronlong/tl-mcfw-rdk
export DVR_RDK_ROOT_PATH

include Rules.make

all:
	make -C host all
	make -C dsp all

clean:
	make -C host clean
	make -C dsp clean
```

Saving file '/media/DM8148/Dem...
Makefile
Tab Width: 8
Ln 2, Col 47
INS

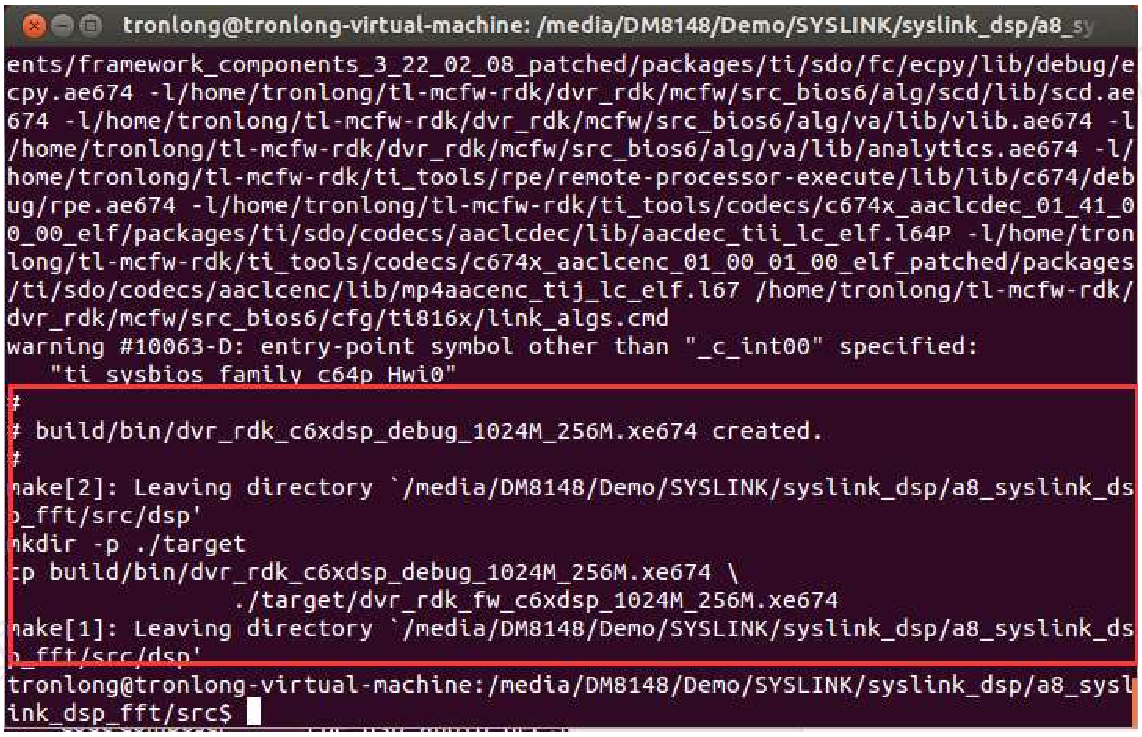
图 2

只需修改顶层 makefile，上图中红框中的路径指向开发环境，DSP 和 Host 目录下的 makefile 无需修改。编译完成后，在 DSP 和 Host 目录下会生成 target 目录，包含目标可执行文件或者固件模块。

进入例程代码 src 目录后，如下进行编译操作：

Host# make clean

Host# make



```
tronlong@tronlong-virtual-machine: /media/DM8148/Demo/SYSLINK/syslink_dsp/a8_sy
ents/framework_components_3_22_02_08_patched/packages/ti/sdo/fc/ecpy/lib/debug/e
cpy.ae674 -l/home/tronlong/tl-mcfw-rdk/dvr_rdk/mcfw/src_bios6/alg/scd/lib/scd.ae
674 -l/home/tronlong/tl-mcfw-rdk/dvr_rdk/mcfw/src_bios6/alg/va/lib/vlib.ae674 -l
/home/tronlong/tl-mcfw-rdk/dvr_rdk/mcfw/src_bios6/alg/va/lib/analytics.ae674 -l/
home/tronlong/tl-mcfw-rdk/ti_tools/rpe/remote-processor-execute/lib/lib/c674/deb
ug/rpe.ae674 -l/home/tronlong/tl-mcfw-rdk/ti_tools/codecs/c674x_aaclcdec_01_41_0
0_00_elf/packages/ti/sdo/codecs/aaclcdec/lib/aacdec_tii_lc_elf.l64P -l/home/tron
long/tl-mcfw-rdk/ti_tools/codecs/c674x_aaclcenc_01_00_01_00_elf_patched/packages
/ti/sdo/codecs/aaclcenc/lib/mp4aacenc_tij_lc_elf.l67 /home/tronlong/tl-mcfw-rdk/
dvr_rdk/mcfw/src_bios6/cfg/ti816x/link_algs.cmd
warning #10063-D: entry-point symbol other than "_c_int00" specified:
   "ti_sysbios_family_c64p_Hwi0"
#
# build/bin/dvr_rdk_c6xdsp_debug_1024M_256M.xe674 created.
#
make[2]: Leaving directory `/media/DM8148/Demo/SYSLINK/syslink_dsp/a8_syslink_ds
p_fft/src/dsp'
mkdir -p ./target
cp build/bin/dvr_rdk_c6xdsp_debug_1024M_256M.xe674 \
                ./target/dvr_rdk_fw_c6xdsp_1024M_256M.xe674
make[1]: Leaving directory `/media/DM8148/Demo/SYSLINK/syslink_dsp/a8_syslink_ds
p_fft/src/dsp'
tronlong@tronlong-virtual-machine:/media/DM8148/Demo/SYSLINK/syslink_dsp/a8_sysl
ink_dsp_fft/src$
```

图 3

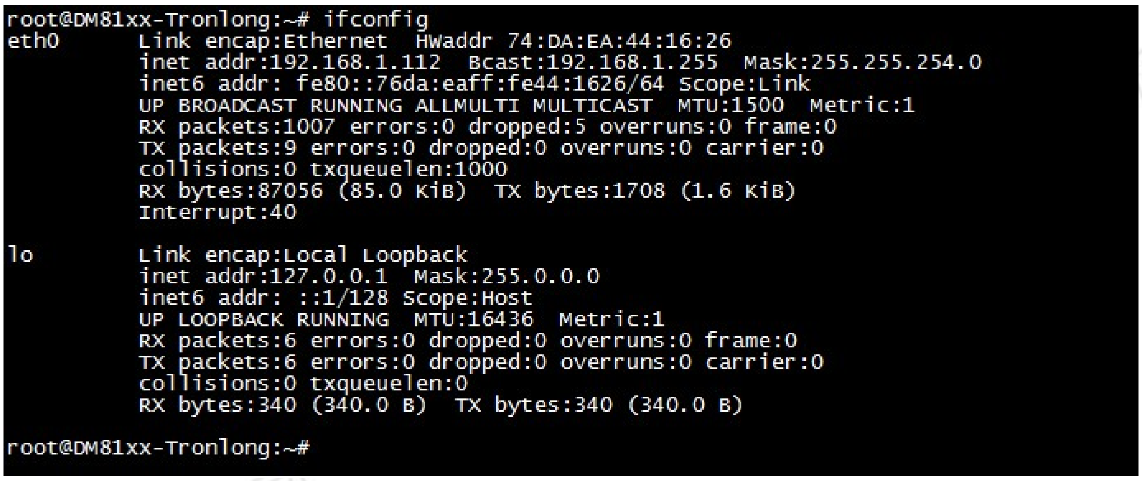
编译完成后，例程运行文件在"src/host/target/"目录中生成，DSP 固件文件在"src/dsp/target/"目录中生成，例程运行文件用户可以拷贝到开发板随机位置运行，DSP 固件必须拷贝到开发板系统"/opt/dvr_rdk/ti814x/firmware"目录。

例程编译耗时 1～2min。

2.1.3 运行

为了调试方便，可以在 Ubuntu 上使用 scp 命令拷贝例程文件到开发板系统，操作如下：

Target# ifconfig //启动开发板，查询开发板 ip

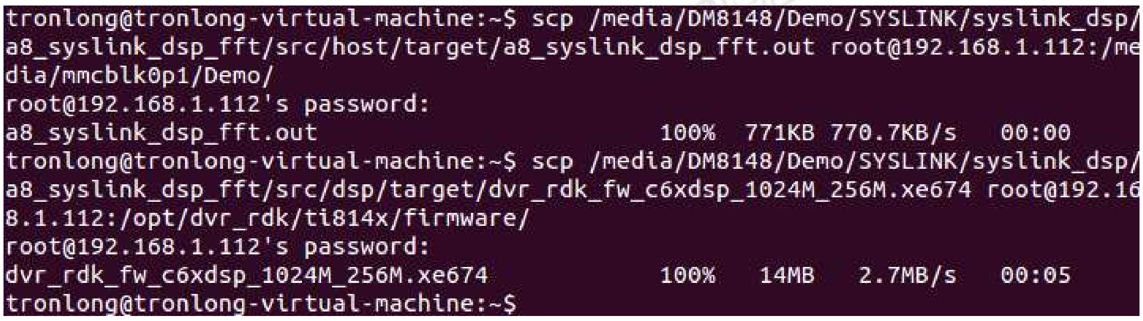


```
root@DM81xx-Tronlong:~# ifconfig
eth0      Link encap:Ethernet  HWaddr 74:DA:EA:44:16:26
          inet addr:192.168.1.112  Bcast:192.168.1.255  Mask:255.255.254.0
          inet6 addr: fe80::76da:eaff:fe44:1626/64 Scope:Link
          UP BROADCAST RUNNING ALLMULTI MULTICAST  MTU:1500  Metric:1
          RX packets:1007 errors:0 dropped:5 overruns:0 frame:0
          TX packets:9 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:87056 (85.0 KiB)  TX bytes:1708 (1.6 KiB)
          Interrupt:40

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:6 errors:0 dropped:0 overruns:0 frame:0
          TX packets:6 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:340 (340.0 B)  TX bytes:340 (340.0 B)

root@DM81xx-Tronlong:~#
```

图 4

Host# scp /media/DM8148/Demo/SYSLINK/syslink_dsp/a8_syslink_dsp_fft/src/host/target/a8_syslink_dsp_fft.out root@192.168.1.112:/media/mmcblk0p1/Demo/ //拷贝例程文件到开发板系统

Host# scp /media/DM8148/Demo/SYSLINK/syslink_dsp/a8_syslink_dsp_fft/src/dsp/target/dvr_rdk_fw_c6xdsp_1024M_256M.xe674 root@192.168.1.112:/opt/dvr_rdk/ti814x/firmware/ //拷贝 DSP 固件到开发板系统



```
tronlong@tronlong-virtual-machine:~$ scp /media/DM8148/Demo/SYSLINK/syslink_dsp/
a8_syslink_dsp_fft/src/host/target/a8_syslink_dsp_fft.out root@192.168.1.112:/me
dia/mmcblk0p1/Demo/
root@192.168.1.112's password:
a8_syslink_dsp_fft.out                         100%  771KB 770.7KB/s   00:00
tronlong@tronlong-virtual-machine:~$ scp /media/DM8148/Demo/SYSLINK/syslink_dsp/
a8_syslink_dsp_fft/src/dsp/target/dvr_rdk_fw_c6xdsp_1024M_256M.xe674 root@192.16
8.1.112:/opt/dvr_rdk/ti814x/firmware/
root@192.168.1.112's password:
dvr_rdk_fw_c6xdsp_1024M_256M.xe674             100%   14MB   2.7MB/s   00:05
tronlong@tronlong-virtual-machine:~$
```

图 5

在开发板系统下，先重启开发板系统，加载新 DSP 固件，再执行例程程序如下操作：

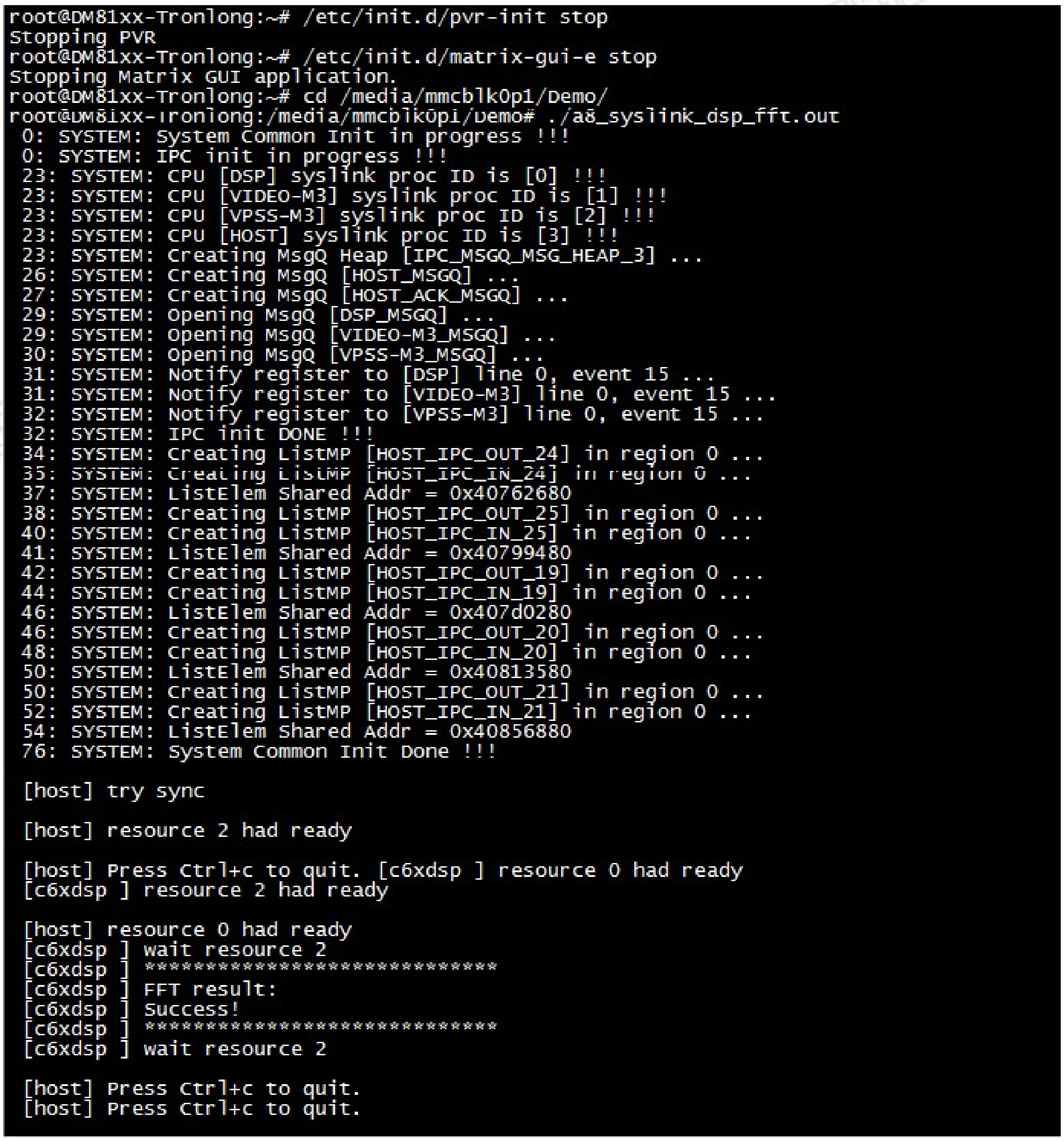
Target# reboot //重启系统，不要直接进行开发板断电重启或者用 reset 按键重启，否则可能会损坏开发板数据

Target# /etc/init.d/pvr-init stop //关闭 PVR 功能

Target# /etc/init.d/matrix-gui-e stop //关闭 MATRIX 功能

Target# cd /media/mmcblk0p1/Demo/ //进入例程文件所在目录，根据实际修改

Target# ./a8_syslink_dsp_fft.out



```
root@DM81xx-Tronlong:~# /etc/init.d/pvr-init stop
Stopping PVR
root@DM81xx-Tronlong:~# /etc/init.d/matrix-gui-e stop
Stopping Matrix GUI application.
root@DM81xx-Tronlong:~# cd /media/mmcblk0p1/Demo/
root@DM81xx-Tronlong:/media/mmcblk0p1/Demo# ./a8_syslink_dsp_fft.out
 0: SYSTEM: System Common Init in progress !!!
 0: SYSTEM: IPC init in progress !!!
 23: SYSTEM: CPU [DSP] syslink proc ID is [0] !!!
 23: SYSTEM: CPU [VIDEO-M3] syslink proc ID is [1] !!!
 23: SYSTEM: CPU [VPSS-M3] syslink proc ID is [2] !!!
 23: SYSTEM: CPU [HOST] syslink proc ID is [3] !!!
 23: SYSTEM: Creating MsgQ Heap [IPC_MSGQ_MSG_HEAP_3] ...
 26: SYSTEM: Creating MsgQ [HOST_MSGQ] ...
 27: SYSTEM: Creating MsgQ [HOST_ACK_MSGQ] ...
 29: SYSTEM: Opening MsgQ [DSP_MSGQ] ...
 29: SYSTEM: Opening MsgQ [VIDEO-M3_MSGQ] ...
 30: SYSTEM: Opening MsgQ [VPSS-M3_MSGQ] ...
 31: SYSTEM: Notify register to [DSP] line 0, event 15 ...
 31: SYSTEM: Notify register to [VIDEO-M3] line 0, event 15 ...
 32: SYSTEM: Notify register to [VPSS-M3] line 0, event 15 ...
 32: SYSTEM: IPC init DONE !!!
 34: SYSTEM: Creating ListMP [HOST_IPC_OUT_24] in region 0 ...
 35: SYSTEM: Creating ListMP [HOST_IPC_IN_24] in region 0 ...
 37: SYSTEM: ListElem Shared Addr = 0x40762680
 38: SYSTEM: Creating ListMP [HOST_IPC_OUT_25] in region 0 ...
 40: SYSTEM: Creating ListMP [HOST_IPC_IN_25] in region 0 ...
 41: SYSTEM: ListElem Shared Addr = 0x40799480
 42: SYSTEM: Creating ListMP [HOST_IPC_OUT_19] in region 0 ...
 44: SYSTEM: Creating ListMP [HOST_IPC_IN_19] in region 0 ...
 46: SYSTEM: ListElem Shared Addr = 0x407d0280
 46: SYSTEM: Creating ListMP [HOST_IPC_OUT_20] in region 0 ...
 48: SYSTEM: Creating ListMP [HOST_IPC_IN_20] in region 0 ...
 50: SYSTEM: ListElem Shared Addr = 0x40813580
 50: SYSTEM: Creating ListMP [HOST_IPC_OUT_21] in region 0 ...
 52: SYSTEM: Creating ListMP [HOST_IPC_IN_21] in region 0 ...
 54: SYSTEM: ListElem Shared Addr = 0x40856880
 76: SYSTEM: System Common Init Done !!!

 [host] try sync

 [host] resource 2 had ready

 [host] Press Ctrl+c to quit. [c6xdsp ] resource 0 had ready
 [c6xdsp ] resource 2 had ready

 [host] resource 0 had ready
 [c6xdsp ] wait resource 2
 [c6xdsp ] ******************************
 [c6xdsp ] FFT result:
 [c6xdsp ] Success!
 [c6xdsp ] ******************************
 [c6xdsp ] wait resource 2

 [host] Press Ctrl+c to quit.
 [host] Press Ctrl+c to quit.
```

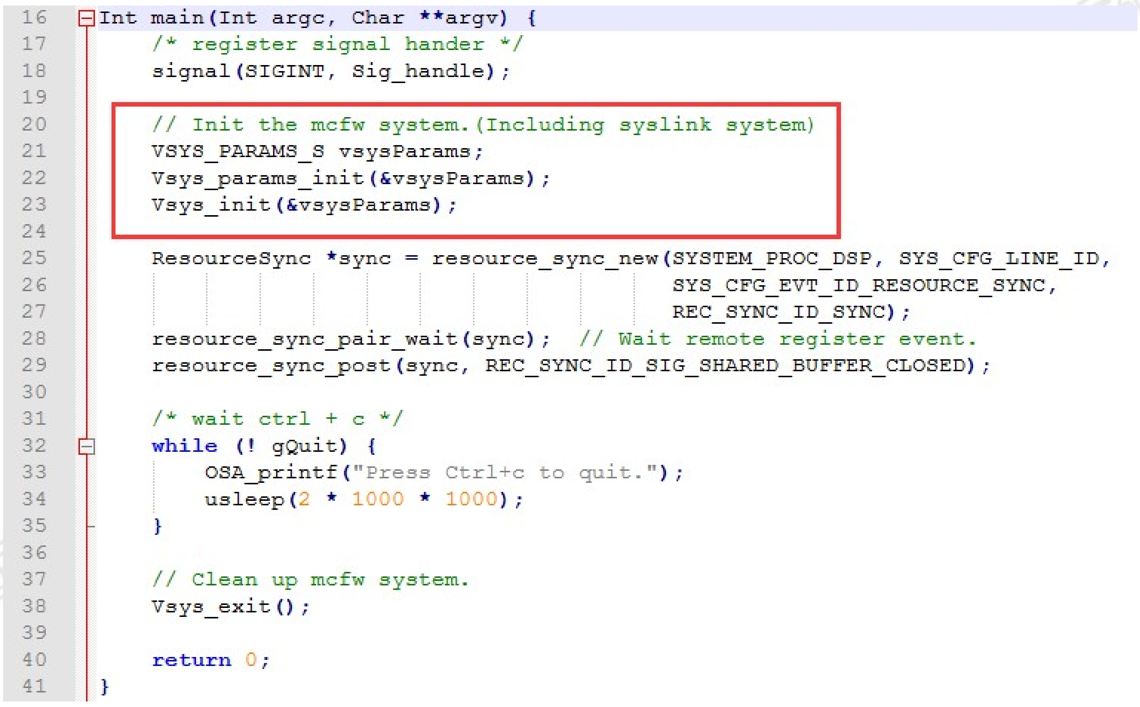
图 6

程序首先进行 SYSLINK 的同步，然后进行 FFT 运算，并对结果做 IFFT 运算，得出的结果与原始数据进行比较。

2.1.4 关键代码

（1）Host 端

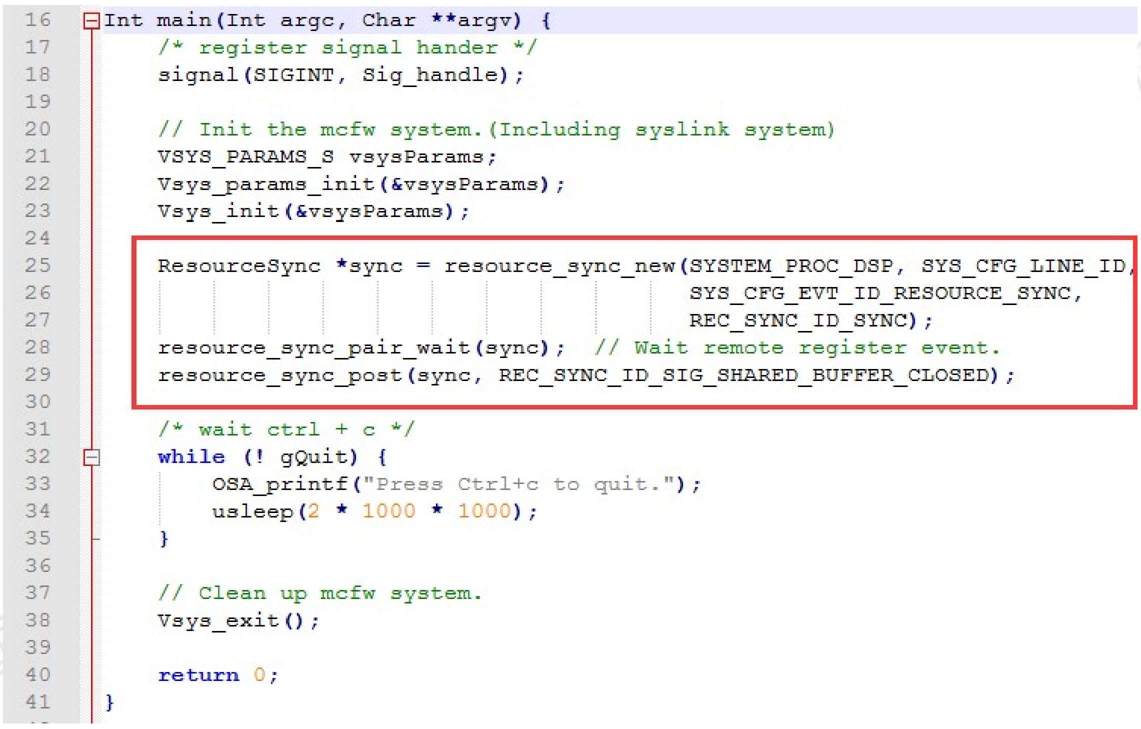
"src\host\main.c"文件中 main(...)函数实现 MCFW 初始化：



```
Int main(Int argc, Char **argv) {
    /* register signal hander */
    signal(SIGINT, Sig_handle);

    // Init the mcfw system.(Including syslink system)
    VSYS_PARAMS_S vsysParams;
    Vsys_params_init(&vsysParams);
    Vsys_init(&vsysParams);

    ResourceSync *sync = resource_sync_new(SYSTEM_PROC_DSP, SYS_CFG_LINE_ID,
                                           SYS_CFG_EVT_ID_RESOURCE_SYNC,
                                           REC_SYNC_ID_SYNC);
    resource_sync_pair_wait(sync);  // Wait remote register event.
    resource_sync_post(sync, REC_SYNC_ID_SIG_SHARED_BUFFER_CLOSED);

    /* wait ctrl + c */
    while (! gQuit) {
        OSA_printf("Press Ctrl+c to quit.");
        usleep(2 * 1000 * 1000);
    }

    // Clean up mcfw system.
    Vsys_exit();

    return 0;
}
```

图 7

"src\host\main.c"文件中 main(...)函数中 ResourceSync 是对 SysLink Notify 的封装，这里的目的是确保 DSP 端程序准备好进行 FFT 运算：

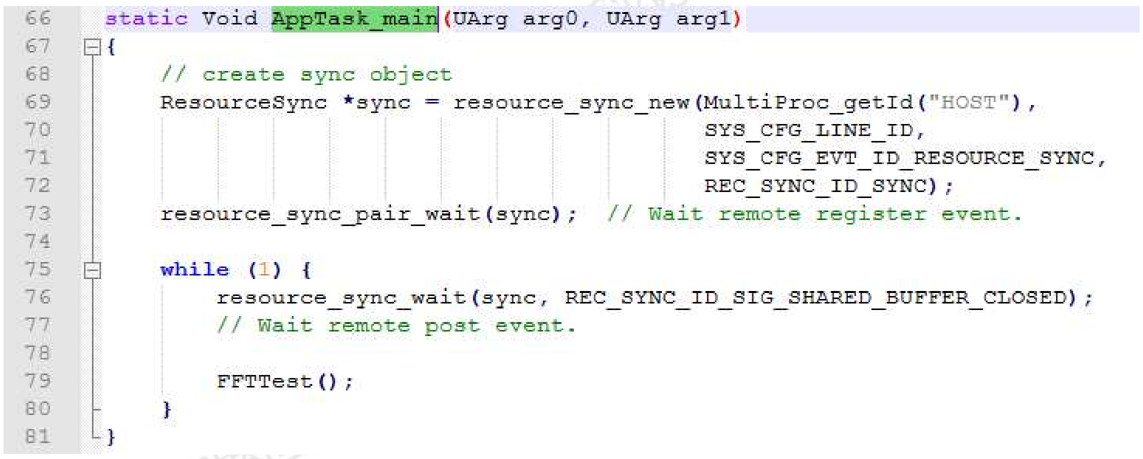


```
Int main(Int argc, Char **argv) {
    /* register signal hander */
    signal(SIGINT, Sig_handle);

    // Init the mcfw system.(Including syslink system)
    VSYS_PARAMS_S vsysParams;
    Vsys_params_init(&vsysParams);
    Vsys_init(&vsysParams);

    ResourceSync *sync = resource_sync_new(SYSTEM_PROC_DSP, SYS_CFG_LINE_ID,
                                           SYS_CFG_EVT_ID_RESOURCE_SYNC,
                                           REC_SYNC_ID_SYNC);
    resource_sync_pair_wait(sync);  // Wait remote register event.
    resource_sync_post(sync, REC_SYNC_ID_SIG_SHARED_BUFFER_CLOSED);

    /* wait ctrl + c */
    while (! gQuit) {
        OSA_printf("Press Ctrl+c to quit.");
        usleep(2 * 1000 * 1000);
    }

    // Clean up mcfw system.
    Vsys_exit();

    return 0;
}
```

图 8

（2）DSP 端

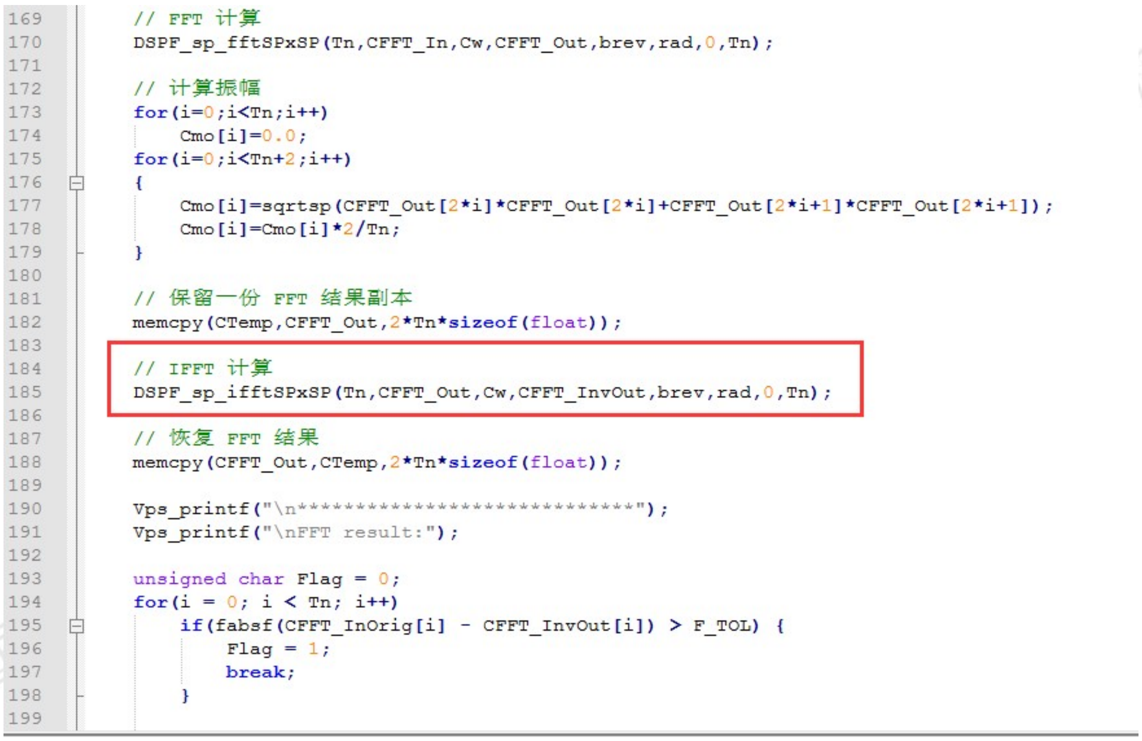
"\src\dsp\app_task.c"文件中 AppTask_main(...)函数与 HOST 同步后等待信号开始 FFT 运算：



```
static Void AppTask_main(UArg arg0, UArg arg1)
{
    // create sync object
    ResourceSync *sync = resource_sync_new(MultiProc_getId("HOST"),
                                           SYS_CFG_LINE_ID,
                                           SYS_CFG_EVT_ID_RESOURCE_SYNC,
                                           REC_SYNC_ID_SYNC);
    resource_sync_pair_wait(sync);  // Wait remote register event.

    while (1) {
        resource_sync_wait(sync, REC_SYNC_ID_SIG_SHARED_BUFFER_CLOSED);
        // Wait remote post event.

        FFTTest();
    }
}
```

图 9

"\src\dsp\app_task.c"文件中 FFTTest(...)函数实现 FFT 运算后再进行 IFFT 运算：



```
    // FFT 计算
    DSPF_sp_fftSPxSP(Tn,CFFT_In,Cw,CFFT_Out,brev,rad,0,Tn);

    // 计算振幅
    for(i=0;i<Tn;i++)
        Cmo[i]=0.0;
    for(i=0;i<Tn+2;i++)
    {
        Cmo[i]=sqrtsp(CFFT_Out[2*i]*CFFT_Out[2*i]+CFFT_Out[2*i+1]*CFFT_Out[2*i+1]);
        Cmo[i]=Cmo[i]*2/Tn;
    }

    // 保留一份 FFT 结果副本
    memcpy(CTemp,CFFT_Out,2*Tn*sizeof(float));

    // IFFT 计算
    DSPF_sp_ifftSPxSP(Tn,CFFT_Out,Cw,CFFT_InvOut,brev,rad,0,Tn);

    // 恢复 FFT 结果
    memcpy(CFFT_Out,CTemp,2*Tn*sizeof(float));

    Vps_printf("\n*******************************");
    Vps_printf("\nFFT result:");

    unsigned char Flag = 0;
    for(i = 0; i < Tn; i++)
        if(fabsf(CFFT_InOrig[i] - CFFT_InvOut[i]) > F_TOL) {
            Flag = 1;
            break;
        }
```

图 10

2.2 a8_syslink_dsp_fir — 有限长单位冲击响应滤波器测试

表 2

开发板	是否支持本例程
TL8148-EasyEVM	支持
TL8148-EVM	支持
TL8127-EasyEVM	支持
TL8127-EVM	支持

2.2.1 例程说明

程序演示了在 MCFW 框架的 SYSLINK 下，ARM 协调 DSP 进行有限长单位冲激响应滤波器测试的功能。

此例程是一个双核例程，除了生成运行于 ARM Linux 的可执行程序外，还要编译出新的 DSP 固件，用来加载运行。

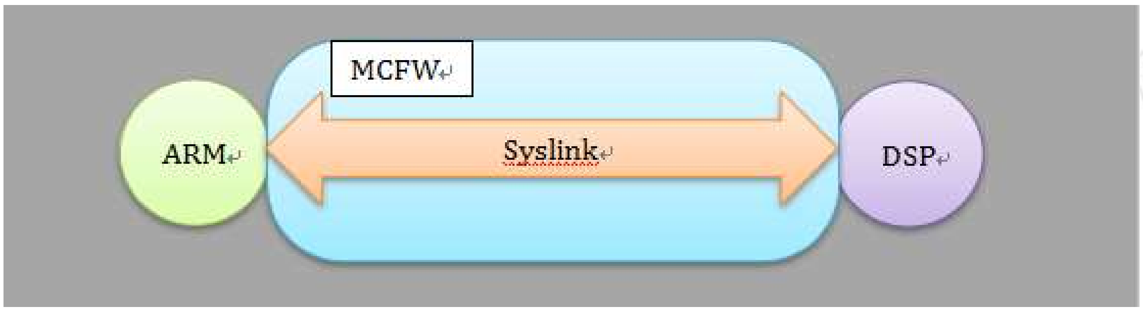


图 11

ARM（Cortex-A8）和 DSP 之间通过 MCFW 的 chain 来通信，数据在一个双向队列中往返 ARM（Cortex-A8）和 DSP，所以 ARM（Cortex-A8）能轻松获取 DSP 处理后的数据。

需要特别注意的是，本例程的 chain 中的 AppLink 是新增的自定义 Link，而非 MCFW 本来就包含。可以在此 Link 中很方便地加入数据处理代码。

文件结构如下：

```
├── dsp
|   ├── app_task.c
|   ├── DSPLib
|   |   ├── include
|   |   └── lib
|   ├── main_c6xdsp.c
|   ├── makefile
|   ├── makefile.mk
|   ├── MathLib
|   |   ├── include
|   |   └── lib
|   ├── resource_sync.c
|   └── resource_sync.h
├── host
|   ├── main.c
|   ├── makefile
|   ├── resource_sync.c
|     └── resource_sync.h
├── makefile
├── Rules.make
└── shared
    ├── protocol.h
    └── sys_config.h
```

主要分为 DSP 和 Host 两部分，分别是生成 DSP firmware 和 ARM binary。两个程序有各自的编译规则，由上层的 makefile 统一控制，分别编译。shared 目录下是两个程序都用到的共同定义。

本例程需要借用 CCS 查看输出波形，所以先按光盘资料软件安装中《Windows 版本 CCS5.5 安装》

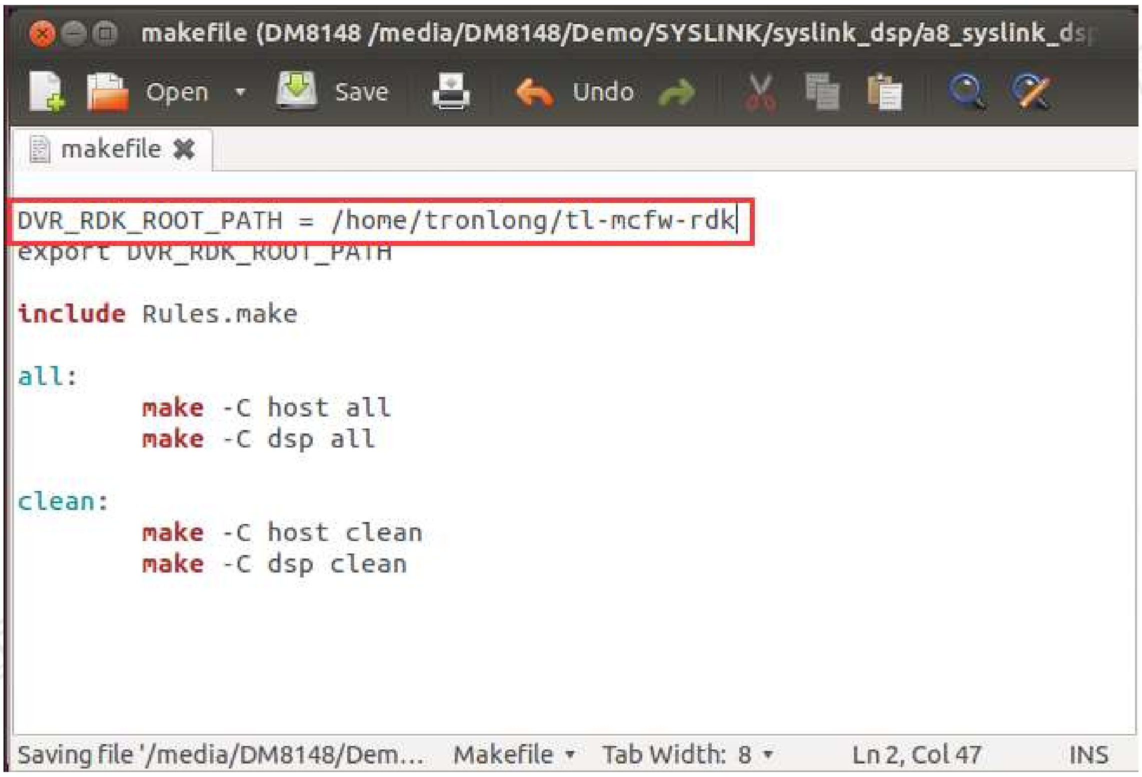
2.2.2 编译

例程代码路径：光盘资料\Demo\SYSLINK\syslink_dsp\a8_syslink_dsp_fir

将例程源码拷贝到 Ubuntu 下，进入 src 目录，修改 makefile：

DVR_RDK_ROOT_PATH：tl-mcfw-rdk 开发包安装路径；

Host# gedit makefile


makefile (DM8148 /media/DM8148/Demo/SYSLINK/syslink_dsp/a8_syslink_ds
Open
Save
Undo
makefile

```
DVR_RDK_ROOT_PATH = /home/tronlong/tl-mcfw-rdk
export DVR_RDK_ROOT_PATH

include Rules.make

all:
	make -C host all
	make -C dsp all

clean:
	make -C host clean
	make -C dsp clean
```

Saving file '/media/DM8148/Dem...
Makefile
Tab Width: 8
Ln 2, Col 47
INS

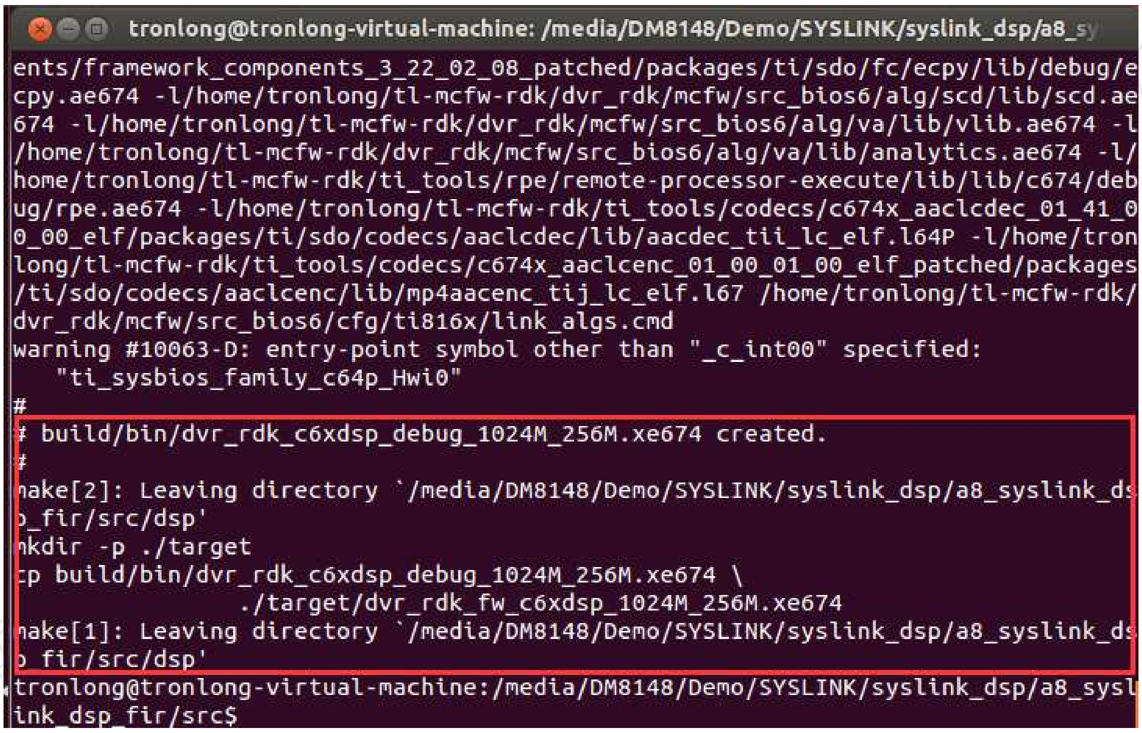
图 12

只需修改顶层 makefile，上图中红框中的路径指向开发环境，DSP 和 Host 目录下的 makefile 无需修改。编译完成后，在 DSP 和 Host 目录下会生成 target 目录，包含目标可执行文件或者固件模块。

进入例程代码 src 目录后，如下进行编译操作：

Host# make clean

Host# make



```
tronlong@tronlong-virtual-machine: /media/DM8148/Demo/SYSLINK/syslink_dsp/a8_sy
ents/framework_components_3_22_02_08_patched/packages/ti/sdo/fc/ecpy/lib/debug/e
cpy.ae674 -l/home/tronlong/tl-mcfw-rdk/dvr_rdk/mcfw/src_bios6/alg/scd/lib/scd.ae
674 -l/home/tronlong/tl-mcfw-rdk/dvr_rdk/mcfw/src_bios6/alg/va/lib/vlib.ae674 -l
/home/tronlong/tl-mcfw-rdk/dvr_rdk/mcfw/src_bios6/alg/va/lib/analytics.ae674 -l/
home/tronlong/tl-mcfw-rdk/ti_tools/rpe/remote-processor-execute/lib/lib/c674/deb
ug/rpe.ae674 -l/home/tronlong/tl-mcfw-rdk/ti_tools/codecs/c674x_aaclcdec_01_41_0
0_00_elf/packages/ti/sdo/codecs/aaclcdec/lib/aacdec_tii_lc_elf.l64P -l/home/tron
long/tl-mcfw-rdk/ti_tools/codecs/c674x_aaclcenc_01_00_01_00_elf_patched/packages
/ti/sdo/codecs/aaclcenc/lib/mp4aacenc_tij_lc_elf.l67 /home/tronlong/tl-mcfw-rdk/
dvr_rdk/mcfw/src_bios6/cfg/ti816x/link_algs.cmd
warning #10063-D: entry-point symbol other than "_c_int00" specified:
   "ti_sysbios_family_c64p_Hwi0"
#
# build/bin/dvr_rdk_c6xdsp_debug_1024M_256M.xe674 created.
#
make[2]: Leaving directory `/media/DM8148/Demo/SYSLINK/syslink_dsp/a8_syslink_ds
p_fir/src/dsp'
mkdir -p ./target
cp build/bin/dvr_rdk_c6xdsp_debug_1024M_256M.xe674 \
                ./target/dvr_rdk_fw_c6xdsp_1024M_256M.xe674
make[1]: Leaving directory `/media/DM8148/Demo/SYSLINK/syslink_dsp/a8_syslink_ds
p_fir/src/dsp'
tronlong@tronlong-virtual-machine:/media/DM8148/Demo/SYSLINK/syslink_dsp/a8_sysl
ink_dsp_fir/src$
```

图 13

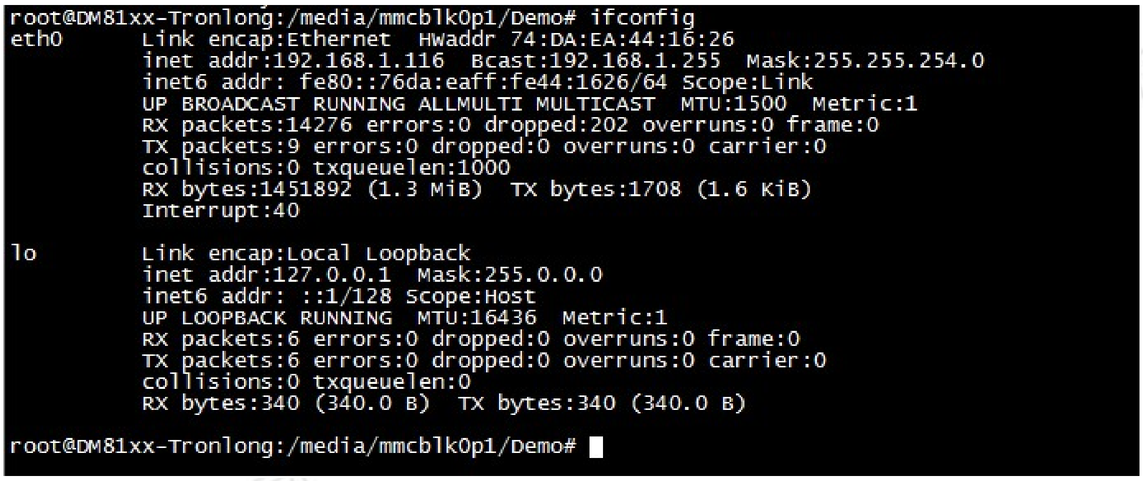
编译完成后，例程运行文件在"src/host/target/"目录中生成，DSP 固件文件在"src/dsp/target/"目录中生成，例程运行文件用户可以拷贝到开发板随机位置运行，DSP 固件必须拷贝到开发板系统"/opt/dvr_rdk/ti814x/firmware"目录。

例程编译耗时 1～2min。

2.2.3 运行

为了调试方便，可以在 Ubuntu 上使用 scp 命令拷贝例程文件到开发板系统，操作如下：

Target# ifconfig //启动开发板，查询开发板 ip

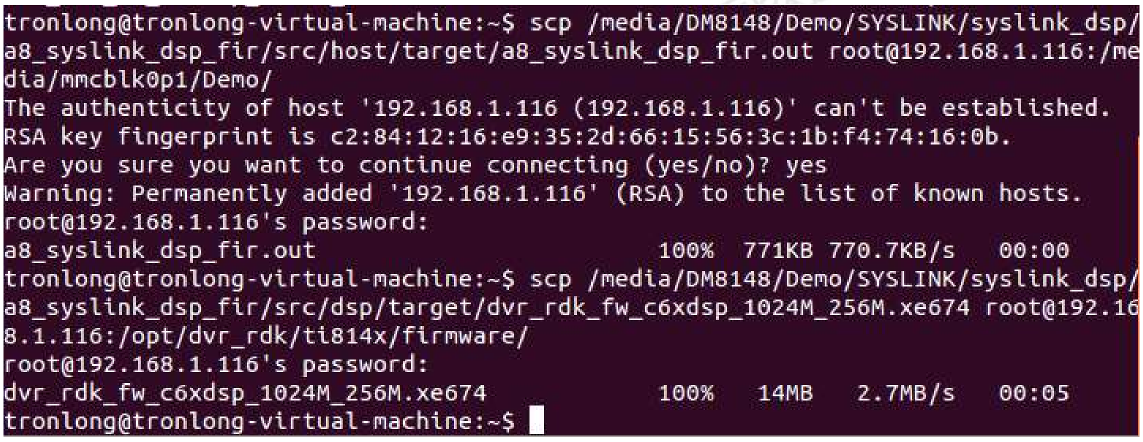


```
root@DM81xx-Tronlong:/media/mmcblk0p1/Demo# ifconfig
eth0      Link encap:Ethernet  HWaddr 74:DA:EA:44:16:26
          inet addr:192.168.1.116  Bcast:192.168.1.255  Mask:255.255.254.0
          inet6 addr: fe80::76da:eaff:fe44:1626/64 Scope:Link
          UP BROADCAST RUNNING ALLMULTI MULTICAST  MTU:1500  Metric:1
          RX packets:14276 errors:0 dropped:202 overruns:0 frame:0
          TX packets:9 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:1451892 (1.3 MiB)  TX bytes:1708 (1.6 KiB)
          Interrupt:40

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:6 errors:0 dropped:0 overruns:0 frame:0
          TX packets:6 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:340 (340.0 B)  TX bytes:340 (340.0 B)

root@DM81xx-Tronlong:/media/mmcblk0p1/Demo#
```

图 14

Host# scp /media/DM8148/Demo/SYSLINK/syslink_dsp/a8_syslink_dsp_fir/src/host/target/a8_syslink_dsp_fir.out root@192.168.1.116:/media/mmcblk0p1/Demo/ //拷贝例程文件到开发板系统

Host# scp /media/DM8148/Demo/SYSLINK/syslink_dsp/a8_syslink_dsp_fir/src/dsp/target/dvr_rdk_fw_c6xdsp_1024M_256M.xe674 root@192.168.1.116:/opt/dvr_rdk/ti814x/firmware/ //拷贝 DSP 固件到开发板系统



```
tronlong@tronlong-virtual-machine:~$ scp /media/DM8148/Demo/SYSLINK/syslink_dsp/
a8_syslink_dsp_fir/src/host/target/a8_syslink_dsp_fir.out root@192.168.1.116:/me
dia/mmcblk0p1/Demo/
The authenticity of host '192.168.1.116 (192.168.1.116)' can't be established.
RSA key fingerprint is c2:84:12:16:e9:35:2d:66:15:56:3c:1b:f4:74:16:0b.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.1.116' (RSA) to the list of known hosts.
root@192.168.1.116's password:
a8_syslink_dsp_fir.out                        100%  771KB 770.7KB/s   00:00
tronlong@tronlong-virtual-machine:~$ scp /media/DM8148/Demo/SYSLINK/syslink_dsp/
a8_syslink_dsp_fir/src/dsp/target/dvr_rdk_fw_c6xdsp_1024M_256M.xe674 root@192.16
8.1.116:/opt/dvr_rdk/ti814x/firmware/
root@192.168.1.116's password:
dvr_rdk_fw_c6xdsp_1024M_256M.xe674            100%   14MB   2.7MB/s   00:05
tronlong@tronlong-virtual-machine:~$
```

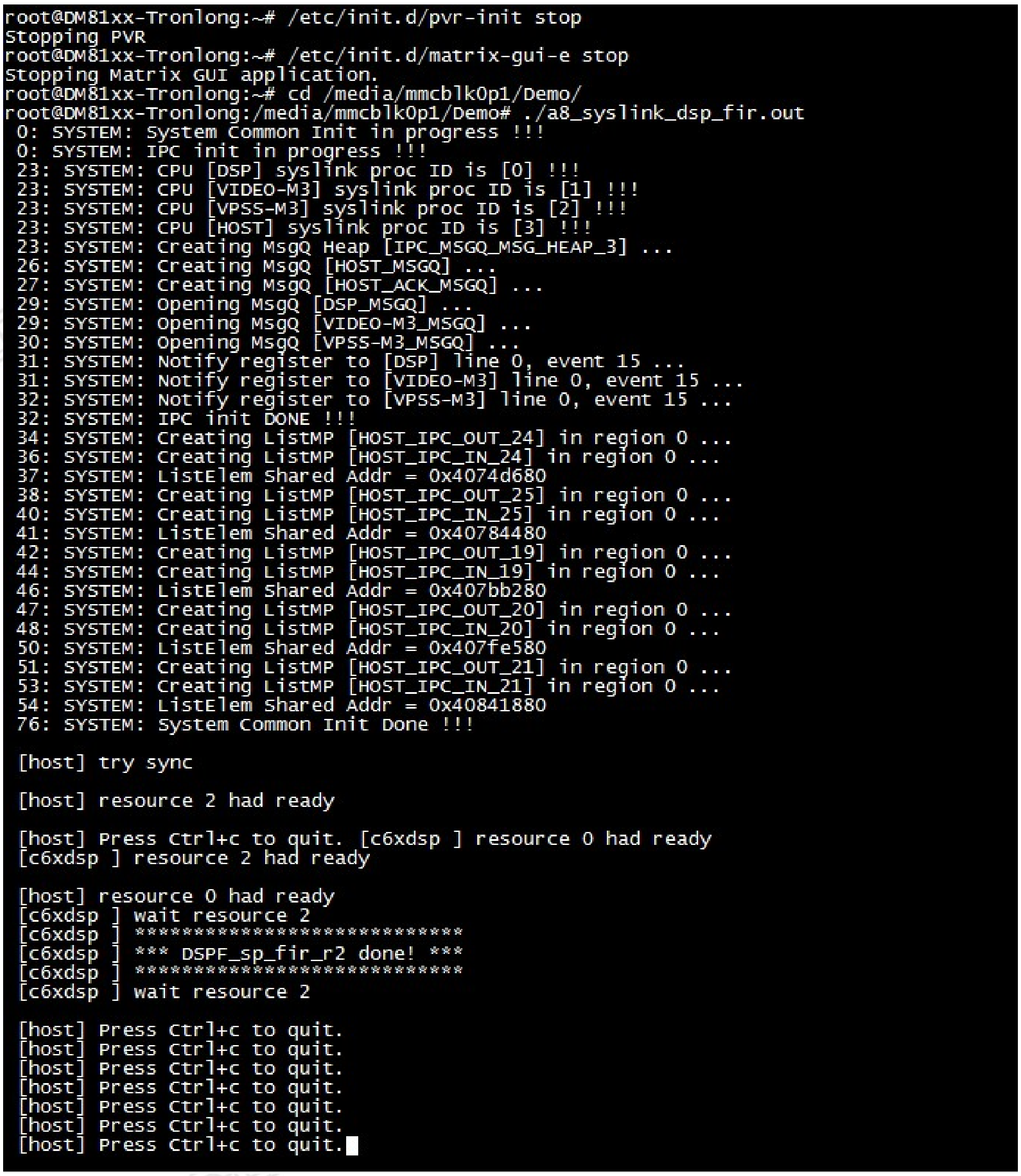
图 15

在开发板系统下，先重启开发板系统，加载新 DSP 固件，再执行例程程序如下操作：

Target# reboot //重启系统，不要直接进行开发板断电重启或者用 reset 按键重启，否则可能会损坏开发板数据

Target# /etc/init.d/pvr-init stop //关闭 PVR 功能

```
Target#     /etc/init.d/matrix-gui-e stop   //关闭 MATRIX 功能
Target#     cd /media/mmcblk0p1/Demo/      //进入例程文件所在目录，根据实际修改
Target#     ./a8_syslink_dsp_fir.out
```



```
root@DM81xx-Tronlong:~# /etc/init.d/pvr-init stop
Stopping PVR
root@DM81xx-Tronlong:~# /etc/init.d/matrix-gui-e stop
Stopping Matrix GUI application.
root@DM81xx-Tronlong:~# cd /media/mmcblk0p1/Demo/
root@DM81xx-Tronlong:/media/mmcblk0p1/Demo# ./a8_syslink_dsp_fir.out
 0: SYSTEM: System Common Init in progress !!!
 0: SYSTEM: IPC init in progress !!!
 23: SYSTEM: CPU [DSP] syslink proc ID is [0] !!!
 23: SYSTEM: CPU [VIDEO-M3] syslink proc ID is [1] !!!
 23: SYSTEM: CPU [VPSS-M3] syslink proc ID is [2] !!!
 23: SYSTEM: CPU [HOST] syslink proc ID is [3] !!!
 23: SYSTEM: Creating MsgQ Heap [IPC_MSGQ_MSG_HEAP_3] ...
 26: SYSTEM: Creating MsgQ [HOST_MSGQ] ...
 27: SYSTEM: Creating MsgQ [HOST_ACK_MSGQ] ...
 29: SYSTEM: Opening MsgQ [DSP_MSGQ] ...
 29: SYSTEM: Opening MsgQ [VIDEO-M3_MSGQ] ...
 30: SYSTEM: Opening MsgQ [VPSS-M3_MSGQ] ...
 31: SYSTEM: Notify register to [DSP] line 0, event 15 ...
 31: SYSTEM: Notify register to [VIDEO-M3] line 0, event 15 ...
 32: SYSTEM: Notify register to [VPSS-M3] line 0, event 15 ...
 32: SYSTEM: IPC init DONE !!!
 34: SYSTEM: Creating ListMP [HOST_IPC_OUT_24] in region 0 ...
 36: SYSTEM: Creating ListMP [HOST_IPC_IN_24] in region 0 ...
 37: SYSTEM: ListElem Shared Addr = 0x4074d680
 38: SYSTEM: Creating ListMP [HOST_IPC_OUT_25] in region 0 ...
 40: SYSTEM: Creating ListMP [HOST_IPC_IN_25] in region 0 ...
 41: SYSTEM: ListElem Shared Addr = 0x40784480
 42: SYSTEM: Creating ListMP [HOST_IPC_OUT_19] in region 0 ...
 44: SYSTEM: Creating ListMP [HOST_IPC_IN_19] in region 0 ...
 46: SYSTEM: ListElem Shared Addr = 0x407bb280
 47: SYSTEM: Creating ListMP [HOST_IPC_OUT_20] in region 0 ...
 48: SYSTEM: Creating ListMP [HOST_IPC_IN_20] in region 0 ...
 50: SYSTEM: ListElem Shared Addr = 0x407fe580
 51: SYSTEM: Creating ListMP [HOST_IPC_OUT_21] in region 0 ...
 53: SYSTEM: Creating ListMP [HOST_IPC_IN_21] in region 0 ...
 54: SYSTEM: ListElem Shared Addr = 0x40841880
 76: SYSTEM: System Common Init Done !!!

 [host] try sync

 [host] resource 2 had ready

 [host] Press Ctrl+c to quit. [c6xdsp ] resource 0 had ready
 [c6xdsp ] resource 2 had ready

 [host] resource 0 had ready
 [c6xdsp ] wait resource 2
 [c6xdsp ] ****************************
 [c6xdsp ] *** DSPF_sp_fir_r2 done! ***
 [c6xdsp ] ****************************
 [c6xdsp ] wait resource 2

 [host] Press Ctrl+c to quit.
 [host] Press Ctrl+c to quit.
 [host] Press Ctrl+c to quit.
 [host] Press Ctrl+c to quit.
 [host] Press Ctrl+c to quit.
 [host] Press Ctrl+c to quit.
 [host] Press Ctrl+c to quit.
```

图 16

运行例程后，接上仿真器，这里使用的是 XDS200，打开 CCS，选择 File -> New -> Target Configuration File，弹出如下界面，新建 DM8148 仿真模块：