DSP Bridge Install and Build Instructions

Authors
Omar Ramirez
Hari Kanigeri

Date
June 25th 2008

This document provides the details to build Bridge driver and its related components.
The following will be covered in this document

1. Setting up the Build environment
2. Build Bridge driver
3. Build Bridge API libraries
4. Build MPU samples
5. Build DSP samples
6. Configuring Target file system
7. Bridge driver installation
8. Examples

1. Setting up the Build environment

Place the MPU Bridge driver tar ball in a folder where you have permissions and unpack it, eg: install in directory named dspbridge.

cd /dspbridge

tar –xzf dspbridge_labrador_vx.y.tar.gz

On unpacking the main tarball, you will notice following 4 tarballs.

- dspbridge_mpu_driver.tar.gz (DSP Bridge Driver source files)
- dspbridge_mpu_api.tar.gz (Bridge API source files)
- dspbridge_samples.tar.gz (Bridge Samples)
- dspbridge_dsp.tar.gz (DSP libraries and sample binaries)

Unpack all the above 4 tarballs. This would result in the following directory structure from where you unpacked

```
/dspbridge
        /documents
        /dsp
        /mpu_api
        /mpu_driver
        /samples
```

Create the following folders from the top level where you installed Bridge driver (eg: dspbridge)

```
mkdir target
cd target
mkdir lib
```

After Installing the software and creating the new folders, your folder structure should look like:

```
/dspbridge
        /documents
        /dsp
        /mpu_api
        /mpu_driver
        /samples
        /target/lib
```

After creating the new folders, copy the pthread libraries to the "lib" folder.

```
cd /dspbridge/target/lib
cp -x /data/omapts/linux/arm-tc/cs-arm-2007q3/arm-none-linux-gnueabi/libc/lib/libpthread* .
ls -l
ln -s libpthread.so.0 libpthread.so
```

Export CROSS_COMPILE

```
export CROSS_COMPILE=arm-none-linux-gnueabi-
```

2. Building Bridge driver

The prerequisite for building Bridge driver is to install and compile the Linux kernel.

Bridge driver is built from the following location:

```
cd /dspbridge/mpu_driver/src
```

The following command builds the Bridge driver:

```
make PREFIX=<bridgedriver top level directory> PROJROOT=<mpu_driver directory>
KRNLSRC=<kernelsource> TGTROOT=<bridgedriver top level directory> BUILD=rel
CMDDEFS='GT_TRACE DEBUG'
```

Provide full path to KRNLSRC and TGTROOT. For example, if the kernel is in location "/linux/kernel_org_2.6_kernel", and the Bridge driver is installed in "HOME/dspbridge" folder, the command would look like:

```
make PREFIX=HOME/dspbridge PROJROOT=HOME/dspbridge/mpu_driver
KRNLSRC=/linux/kernel_org_2.6_kernel TGTROOT=HOME/dspbridge BUILD=rel
CMDDEFS='GT_TRACE DEBUG'
```

To clean the build, use the "clean" option to the build command. eg:

```
make PREFIX=HOME/dspbridge PROJROOT=HOME/dspbridge/mpu_driver
KRNLSRC=/linux/kernel_org_2.6_kernel TGTROOT=HOME/dspbridge BUILD=rel
CMDDEFS='GT_TRACE DEBUG'
```

At the end of successful build, the Bridge driver output (bridgedriver.ko) will be in the "/dspbridge/mpu_driver/src" folder.

3. Building Bridge API libraries

Bridge API libraries are built from the following location:

cd /dspbridge/mpu_api/src

The following command builds the Bridge API libraries:

make PREFIX=<bridgedriver top level directory> TGTROOT=<bridgedriver top level directory> BUILD=rel CMDDEFS='GT_TRACE DEBUG'

Provide full path to PREFIX and TGTROOT. For example, if the Bridge driver is installed in "HOME/dspbridge" folder, the command would look like:

make PREFIX=HOME/dspbridge TGTROOT=HOME/dspbridge BUILD=rel CMDDEFS='GT_TRACE DEBUG'

To clean the build, use the "clean" option to the build command. eg:

make PREFIX=HOME/dspbridge TGTROOT=HOME/dspbridge BUILD=rel CMDDEFS='GT_TRACE DEBUG'

At the end of successful build, two libraries are created. One is the Bridge API library (libbridge.so) that is created in "/dspbridge/mpu_api/src/bridge" folder, and the other is the QOS API library (libqos.a) that is created in "/dspbridge/mpu_api/src/qos" folder.

On executing the following command with install as an option, all the Bridge API folders will be copied to /dspbridge/target/lib folder
make PREFIX=HOME/dspbridge TGTROOT=HOME/dspbridge BUILD=rel CMDDEFS='GT_TRACE DEBUG' install

4. Building MPU Bridge samples

The prerequisite for building MPU side samples is to build the MPU Bridge API libraries

MPU Bridge samples are built from the following location:

cd /dspbridge/samples/mpu/src

Copy the Bridge API libraries (libbridge.so and libqos.a) that are generated from building Bridge API libraries to the MPU samples lib folder /dspbridge/samples/mpu/lib

The following command builds the Bridge Samples

make PREFIX= <Bridge samples top level folder> TGTROOT= <Bridge samples top level folder> BUILD=rel CMDDEFS='DEBUG'

Provide full path to PREFIX and TGTROOT. For example, if the Bridge driver is installed in "HOME/dspbridge" folder, the command would look like:

make PREFIX=HOME/dspbridge TGTROOT=HOME/dspbridge BUILD=rel CMDDEFS='DEBUG'

At the end of successful build, run the above command again with install option.
make PREFIX=HOME/dspbridge TGTROOT=HOME/dspbridge BUILD=rel CMDDEFS='DEBUG' install
This will copy all the samples from respective sample folders to following folder /dspbridge/target/dspbridge folder.

To clean the build, use the "clean" option to the build command. eg:

make PREFIX=HOME/dspbridge KRNLSRC=/linux/kernel_org/2.6_kernel TGTROOT=HOME/dspbridge BUILD=rel CMDDEFS='DEBUG' clean

5. Building DSP Bridge sample applications

DSP Bridge requires the presence of a depot folder. The purpose of this repository is to contain all the dependencies required to build DSP Bridge.

export DEPOT=/<BIOS_install_directory>

Build Procedure:
DSP samples are built from top level Bridge folder. Eg: dspbridge in this case.
cd /dspbridge
One way to verify that you are in the right folder is to check if the file "samplemakefile" is in that folder

Build the DSP samples using the following command
gmake -f samplemakefile .samples

Clean the DSP samples using the following command
gmake -f samplemakefile .clean
After successful build, the DSP-side sample binaries will be in the following folder /dspbridge/samples/dsp. These files have extensions *.dll64P and *.dof64P

6. Configuring Target file system

Once you have all the images built, you are now ready to test them on your target file system.
On target file system, create a folder named "dspbridge" to copy the files you built.

Copy the following images to dspbridge on your target file sytem

1. /dspbridge/mpu_driver/src/bridgedriver.ko (**Bridge driver**)
2. /dspbridge/target/dspbridge (**MPU samples**)
3. Copy the **DSP samples** that are in /dspbridge/samples/dsp
   samples/dsp/*.dll64P
   samples/dsp/*.dof64P

Copy the following images to lib folder on your target file sytem
4. /dspbridge/target/lib (**Bridge APIs and pthread libraries**)

Copy all the install bridge script files from /dspbridge/mpu_driver/utils folder
install_bridge install_bridge_128 uninstall_bridge

Refer to documents/Linux_Bridge_Integration_3430.pdf for more information on these scripts.

7. Bridge Driver Installation

Bridge is installed as a kernel module. Shared Memory size can be configured while installing the bridge. Actual values depend on OEM system configuration.

To install it use:

./install_bridge

or

insmod bridgedriver.ko <your_own_parameters>

To load DSP sample images, use
cexec.out.

8. Examples

1. Running the Ping sample

Execute the following commands to run the ping sample (example to illustrate the messaging between MPU and DSP)

./install_bridge

/cexec.out ddspbase_tiomap3430.dof64P

./ping.out

Refer to documents/ db_linux_pguide.pdf for more examples.