TEXAS INSTRUMENTS

DSP/BIOS™ Link

Installation Guide

1.65

IMPORTANT NOTICE

Following are URLs where you can obtain information on other Texas Instruments products and application solutions:

Products		**Applications**	
Amplifiers	amplifier.ti.com	Audio	www.ti.com/audio
Data Converters	dataconverter.ti.com	Automotive	www.ti.com/automotive
DSP	dsp.ti.com	Broadband	www.ti.com/broadband
Interface	interface.ti.com	Digital Control	www.ti.com/digitalcontrol
Logic	logic.ti.com	Military	www.ti.com/military
Power Mgmt	power.ti.com	Optical Networking	www.ti.com/opticalnetwork
Microcontrollers	microcontroller.ti.com	Security	www.ti.com/security
		Security	www.ti.com/security
		Telephony	www.ti.com/telephony
		Video & Imaging	www.ti.com/video
		Wireless	www.ti.com/wireless

Mailing Address:

Texas Instruments,
Post Office Box 655303,
Dallas, Texas 75265

Table of Contents

List of Figures

List of Tables

Read This First

About This Manual

This document describes how to install and run samples for DA830 i.e. DA8XX from the new DSP/BIOS™ Link.

How to Use This Manual

This document includes the following chapters:

- Chapter 1, *Install Guide for DA8XX* - describes the steps required to install and run samples for DA8xx.

Please go through the Release Notes document available in the release package before starting the installation.

Notation of information elements

The document may contain these additional elements:

Warning

This is an example of warning message. It usually indicates a non-recoverable change.

Caution

This is an example of caution message.

Important

This is an example of important message.

Note

This is an example of additional note. This usually indicates additional information in the current context.

Tip

This is an example of a useful tip.

If You Need Assistance

For any assistance, please send an mail to software support.

Trademarks

Install Guide for DA8XX

Abstract

This chapter describes how to install and run samples for DA8XX from the new DSP/BIOS™ Link.

Table of Contents

1.1. Introduction

1.1.1. Purpose and Scope

DSP/BIOS™ LINK is foundation software for the inter-processor communication across the GPP-DSP boundary. It provides a generic API that abstracts the characteristics of the physical link connecting GPP and DSP from the applications. It eliminates the need for customers to develop such link from scratch and allows them to focus more on application development.

This document provides the users necessary information to install DSP/BIOS™ LINK on the development host.

This document corresponds to the product release Version 1.65.

1.1.2. Terms and Abbreviations

CCS	Code Composer Studio
IPC	Inter Processor Communication
GPP	General Purpose e.g. ARM
DSP	Digital Signal Processor e.g. DM648
DSPLink	A generic term used for DSP/BIOS™ Link. It appears in italics in all usages
CGTools	Code Gen Tools, e.g. Compiler, Linker, Archiver

Table 1.1. Terms and Abbreviations

1.1.3. References

2	LSP_03.00.20.11 User Guide

Table 1.2. References

1.2. INSTALLATION

1.2.1. Basic Installation

The DSP/BIOS™ LINK is made available as a tar.gz file. To install the product follow the steps below:

- Unzip and untar the file dsplink_linux_#version#.tar.gz.

Note

This document assumes the install path to be in the user home directory if working on a Linux PC. This path will be used in remainder of this document

---

Note

This document assumes the install path to be L:\dsplink if working on a Windows PC. This path will be used in remainder of this document.

Note

If the installation was done at different location, make appropriate changes to the commands listed in the document.

It is advisable to archive the released sources in a configuration management system. This will help in merging:

- The updates delivered in the newer releases of DSP/BIOS™ LINK.
- The changes to the product, if any, done by the users.

1.2.1.1. Installing Standalone DSP/BIOS™, XDC and CGTools.

For compilation of DSP-side sources and applications, the CGTools version 6.1.5 can be used. This release has been validated with DSP/BIOS™ version 6.20.00.37, IPC 1.00.00.40 and XDC version xdctools_3_15_00_50.

The standalone DSP/BIOS™ and standalone CGTools are available for Linux platform as well. Refer to the URL mentioned below for getting the distribution of DSP/BIOS™ and the associated installation instructions
DSP/BIOS™ and XDC

The directory structure specified in Figure 1 is expected by the build system of DSP/BIOS™ LINK. If you install the tools to a different directory, you will also need to modify the make system and the scripts contained in the release package. You may need to copy the directories to create the structure expected for compiling sources. Refer to section on "Understanding The MAKE System" in the User Guide for details.

1.2.1.2. Installing GNU make 3.81

For compilation of DSPLINK sources the GNU make 3.81 can be used. Download the make 3.81 from the URL http://ftp.gnu.org/pub/gnu/make/make-3.81.tar.gz

The following are the installation steps required to install make on the development host machine.

- Cd to make-3.81 directory
- Copy and untar make-3.81.tar.gz to your home directory.
- Type './configure' and press enter to configure the package for your system. Running `configure' takes awhile. By default, make package's files will be installed in `/usr/local/bin', `/usr/local/man', etc. You can specify an installation prefix other than `/usr/local' by giving `configure' the option `-- prefix=PREFIX'. For example, To install make at /usr/local/bin run the configure command like below.

  ```
  . /configure --prefix=/usr/local.
  ```

  To install make at /usr/bin run the configure command like below.

  ```
  ./configure --prefix=/usr
  ```

- Type `make' and press enter to compile the package.
- Optionally, type `./make check' and press enter to run any self-tests that come with the package.
- Type `make install' and press enter to install the programs and any data files and documentation.
- For additional details refer to INSTALL file located under make-3.81 directory

1.2.2. Creating development workspace

This document and the scripts included in the release assume the following directory on your development host:

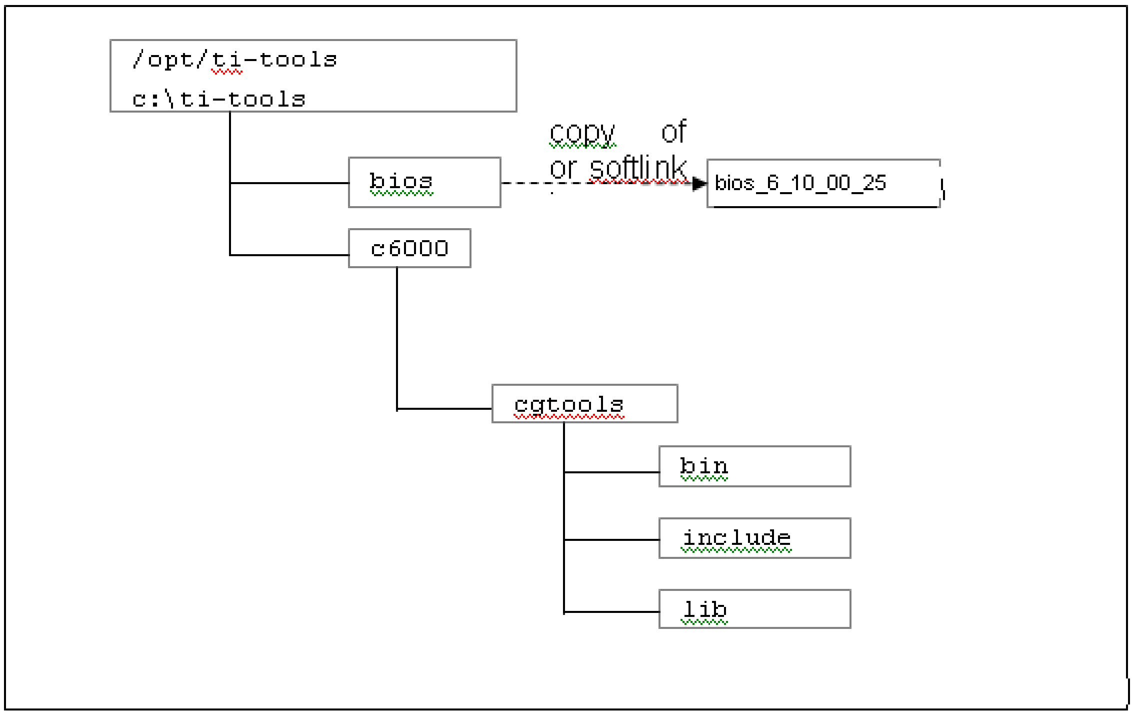


Figure 1.1. Development Workspace

Note

For Linux, the build system shipped with DSP/BIOS™ LINK assumes that the standalone DSP/BIOS™ is installed in the /opt/'ti-tools/' directory on the <ROOT-DRIVE> and CGTools and CSL are installed in the 'ti-tools/c6000' directory on the <ROOT-DRIVE>.

Note

For the Windows development host, the build system shipped with DSP/BIOS™ LINK assumes that the standalone DSP/BIOS™ is installed in the 'ti-tools\' directory on the ROOT-DRIVE and CGTools and CSL are installed in the 'ti-tools\c6000\' directory on the <ROOT-DRIVE>.

Note

To support multiple installations of DSP/BIOS with a single DSP/BIOS™ LINK DSP-side distribution file, a standard /opt/ti-tools/bios on Linux and c:\ti-tools\bios directory is used for the BIOS installation. This can be a soft link or copy to the actual DSP/BIOS installation directory

1.3. Setting up Linux Workstation

The description in this section is based on the following assumptions:

- The workstation is running on Red hat Fedora Core 7 (Kernel ver 2.6.21). Fedora 7
- Services telnetd, nfsd, ftpd are configured on this workstation.

Note

The release package has been tested on Red hat Fedora Core 7 (Kernel ver 2.6.21) You may be able to build on a higher version depending on the compatibility of the build tools in your version with the tested version.

Note

Ensure that kernel sources for Linux have also been installed on the workstation.

1.3.1. Creating target system

This release of DSP/BIOS™ LINK for DA8XX has validated on Linux kernel versions 2.6.32. It is recommended to use this version to use DSP/BIOS™ Link.

1.3.1. Setting up to build the kernel

This step is not required if a ARM tool chain already available on your setup. This tool-chain required for building the LSP and DSPLink.

- The tool chain arm-2009q1-203 can be downloaded from http://www.codesourcery.com/sgpp/lite/arm/portal/subscription3057 .Install the tool chain and add the tool chain directory in your path.
- `$ ./arm-2009q1-203-arm-none-linux-gnueabi.bin`
- `$ export PATH=$PATH:${HOME}/toolchains/2009q1-203/bin`

1.3.2. Setting up the Linux PSP for DA8XX

- Configure the kernel for the DA8XX platform. Please refer to the user guide on how to build the Linux kernel.

1.3.3. Enable TFTP for downloading the kernel image to target

U-boot can be configured to download the kernel onto the target by various mechanisms:

- TFTP

- Serial Port

This section configures the Linux development host as a TFTP server. Modify the xinet.d/ tftp file to enable TFTP:

- Make the following changes:

```
disable     = no
server_args = -s /tftpboot
```

- Restart the network service

```
$ /etc/init.d/xinetd restart
```

The above configuration assumes that a directory tftpboot has been created at the root directory. The files in this directory are exposed through the TFTP protocol.

1.3.4. Create target file system and export through NFS

The target device needs a file system to boot from. The file system can either be exported to the target through NFS, or a RAMDISK can be used. A directory on the development host can be setup and exported for this purpose.

- Copy the target file system to your workspace.

```
$ cp -a ${HOME}/DaVinci-PSP-SDK-03.20.00.11/images/fs/nfs ~/da8xx/target
```

Important

Enter the command shown above in a single line. You need to be root to successfully execute this command. The file system can be copied to a different location. In such a case ~/da8xx/target can be a soft link to the actual location.

- The directory ~/da8xx/target will be mounted as root directory on the target through NFS. To do so, add the following line to the file /etc/exports.

```
/home/user/da8xx/target  *(rw,no_root_squash)
```

Replace the directory in the path above with the actual path of your home directory on the development workstation.

1.4. Configuring CCS

Note

CCS 3.3 can be used for working with DA8XX.

1.5. WORKING ON TARGET PLATFORM

1.5.1. To get address information to run the samples and use for NOLOADER

The addresses to be passed as parameters for NOLOADER.

```
ofd6x.exe -v "dsp executable" | grep -rn A 2 DSPLINK_shmBaseAddress
```

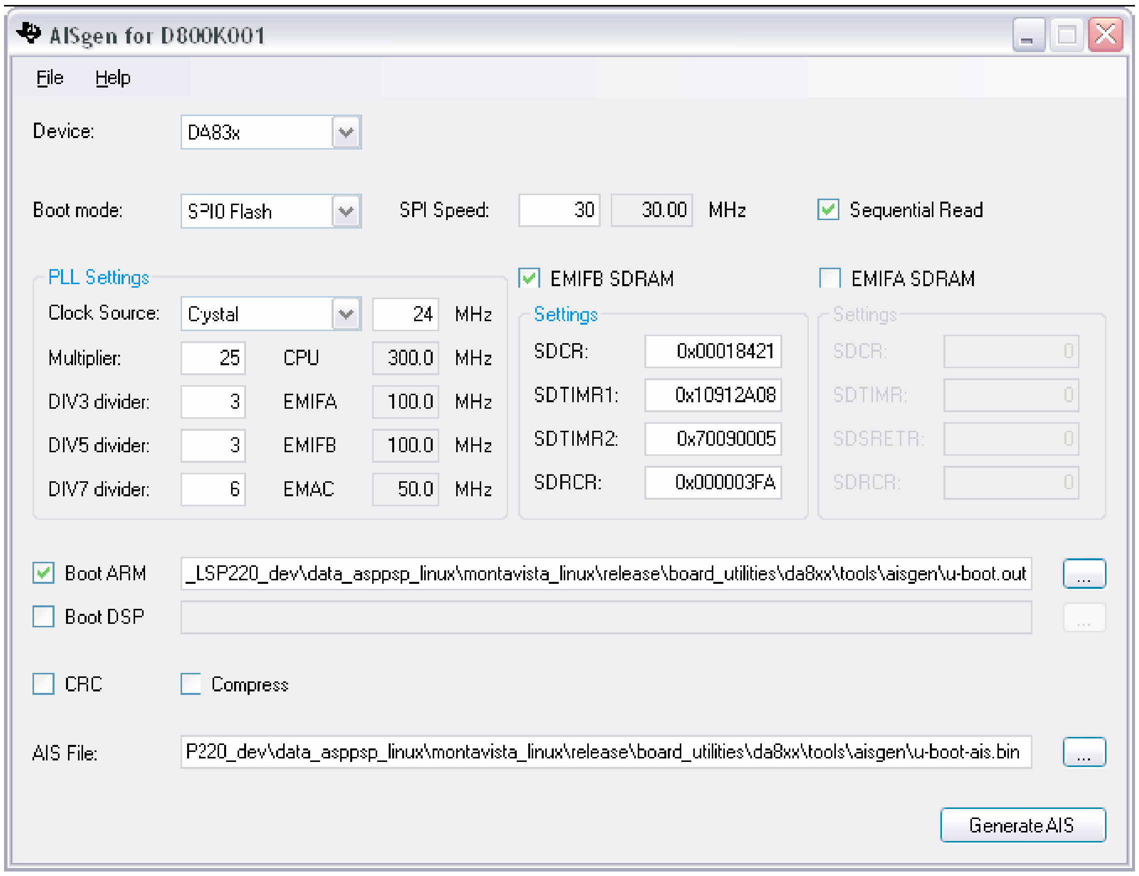
1.5.2. Steps required to run samples for DSP_BootMode_NoBoot mode and NOLOADER mode:

Important

USB JTAG is embedded into the board, so after every power cycle, please close and reopen the CCS.

1.5.2.1. Generating AIS binary file

The DSP boot ROM on DA8xx devices wakes up the ARM and boots the U-Boot. For the DSP boot ROM to boot-up the U-Boot image it needs to be in COFF file format and as part of AIS file format binary. The DSP boot ROM also boots the DSP with the DSP application.This DSP application needs to be as part of AIS file format binary. The DSP boot ROM uses this AIS file to boot ARM and DSP.



Device: DA83x

Boot mode: SPI0 Flash   SPI Speed: 30   30.00 MHz   ✓ Sequential Read

PLL Settings

Clock Source:	Cystal		24	MHz
Multiplier:	25	CPU	300.0	MHz
DIV3 divider:	3	EMIFA	100.0	MHz
DIV5 divider:	3	EMIFB	100.0	MHz
DIV7 divider:	6	EMAC	50.0	MHz

✓ EMIFB SDRAM — Settings

SDCR:	0x00018421
SDTIMR1:	0x10912A08
SDTIMR2:	0x70090005
SDRCR:	0x000003FA

☐ EMIFA SDRAM — Settings

SDCR:	0
SDTIMR:	0
SDSRETR:	0
SDRCR:	0

✓ Boot ARM   _LSP220_dev\data_asppsp_linux\montavista_linux\release\board_utilities\da8xx\tools\aisgen\u-boot.out

☐ Boot DSP

☐ CRC   ☐ Compress

AIS File: P220_dev\data_asppsp_linux\montavista_linux\release\board_utilities\da8xx\tools\aisgen\u-boot-ais.bin

Figure 1.2. AIS Gen Tool

Important

By default precompiled uboot binary supports booting from SPI Flash. It stores ENV varibles at an offset 128k in SPI FLASH. If total AIS binary size becomes more than 128k ,users need to rebuild the uboot by shifting the CFG_ENV_OFFSET such that CFG_ENV_OFFSET does not fall in the AIS binary region. This is required as AIS binary will be flashed at offset 0 in SPI Flash. If CFG_ENV_OFFSET falls within the AIS binary area in SPI Flash, AIS binary contents will be corrupted by uboot environmental varibales.

- Copy the rawtocoff.exe from REL_LSP_03_20_00_11/PSP_0#_##_##_##/board_utilitie s/da8xx/tools/aisgen On to a windows host machine.
- Copy the u-boot.bin file to a directory where rawtocoff.exe is present. From the command prompt window, run the following command to convert uboot.bin to coff format.

```
raw2coff.exe u-boot.out 0xC1080000 u-boot.bin
```

The u-boot.out file thus obtained is a COFF format file.

- The AIS file generator AISgenD800K001-0.3.0.0-setup.exe is included in the folder REL_LSP_02_20_##_##/PSP_02_20_##_##/board_utilitie s/da8xx/ tools/aisgen. Run the installer and follow the onscreen instructions to install the tool. This installation step is to be done for the first time. On subsequent runs, skip directly to the next step.

- Run the AIG generation tool and fill in the values as shown in figure below. In the "Boot ARM" text box, choose the u-boot.out file created in step above. In the "Boot DSP" text box, choose the dsp application that needs to be booted on DSP. Choose an AIS file name in the "AIS File:" text box.

- Click "Generrate AIS" button. The AIS file to be flashed is generated.

1.5.2.2. Flashing Generated AIS binary file on to SPI Flash

- Open CCS3.3 setup and ensure that the DSP and ARM GEL files are pointing to the GEL files are in place.gel files and CCS setup files cab be obtained from the REL_LSP_02_20_##_##/PSP_02_20_##_##/board_utilitie s/da8xx/ tools/gel folder. dskda830_arm.gel is the ARM side GEL file and dskda830_dsp.gel is the DSP side GEL file.

- Open CCS and connect to the ARM on DA8xx device. Since DSP boots first on DA8xx, the DSP needs to be connected to first. When connected, the DSP GEL file wakes up ARM. Once ARM is woken up, CCS can connect to ARM.

- Load the SPI flasher tool shipped in REL_LSP_02_20_##_##/ PSP_02_20_##_##/bin directory of the LSP release package.i.e.REL_LSP_02.20.00.01.tgz) on to the ARM.

- Run the SPI flasher program. On CCSv3.3 you will be prompted for the input AIS file name and offset (in decimal) within the serial flash where the image needs to be burnt. The offset should be given as 0 when burning AIS file.

- Once the SPI flash has been written with AIS file, switch off the board and set the SW2 switch on DSK board as follows Pin# 7 2 1 0 3 Position 0 1 0 1 x

- Power up the board again and you should boot from SPI flash.

- This release has not been tried with CCS 4

- Now run the sample on Linux console.

1.5.3. Running the sample applications

Eight sample applications are provided with DSPLINK for the DA8XX platform. All the sample applications are described in detail in the user guide. This section describes the way to execute the sample applications

The specific instructions shown below refer to the loop sample. However, similar instructions can be used for the other applications also.

1.5.3.1. Configure the DSP/BIOS LINK for DA8XX

The build configuration command must be executed to configure DSPLink for the various parameters such as platform, GPP OS, build configuration etc

```
perl dsplinkcfg.pl --platform=DA8XX --nodsp=1 --dspcfg_0=DA8XXGEMSHMEM --
dspos_0=DSPBIOS6XX --gppos=ARM --comps=ponslrmc
```

```
perl dsplinkcfg.pl --platform=DA8XX --nodsp=1 --dspcfg_0=DA8XXGEMSHMEM --
dspos_0=DSPBIOS6XX --gppos=DA8XXLSPuc  --comps=ponslrmc
```

Important

Enter the commands shown above in single line

Important

For details please refer user guide.

Important

To build the DSP side samples the XDCPATH needs to be export. export XDCPATH="/toolchains/bios6/bios_6_20_00_37/packages;/toolchains/bios6/ipc_1_00_00_40/packages"

Important

To build the GPP side update the Kernel (KERNEL_DIR) and toolchain (TOOL_PATH) path in dsplink/gpp/src/Rules.mk file.

1.5.3.2. Copying files to target file system

1.5.3.2.1. GPP Side

For executing the DEBUG build, follow the steps below to copy the relevant binaries:

```
$ cd ~/dsplink

$ cp gpp/export/BIN/Linux/DA8XX/DEBUG/loopgpp  /opt/dsplink/samples/loop

$ cp gpp/export/BIN/Linux/DA8XX/DEBUG/dsplinkk.* /opt/dsplink/
```

For executing the RELEASE build, follow the steps below to copy the relevant binaries:

```
$ cd ~/dsplink

$ cp gpp/export/BIN/Linux/DA8XX/RELEASE/loopgpp  /opt/dsplink/samples/loop

$ cp gpp/export/BIN/Linux/DA8XX/RELEASE/dsplinkk.* /opt/dsplink/
```

Important

Enter the commands shown above in single line

1.5.3.2.2. DSP Side

The DSP binaries can be built either on the Linux workstation or the Windows host.

After the binaries have been built, Need to generate AIS binary file to boot DSP with the specific DSP sample.

For executing the DEBUG build, follow the steps below to copy the relevant binaries:

```
$ cd ~/dsplink
```

```
$ cp dsp/export/BIN/DspBios/DA8XX/DA8XXGEM_x/DEBUG/loop.out  /opt/dsplink/
samples/loop
```

For executing the RELEASE build, follow the steps below to copy the relevant binaries:

```
$ cd ~/dsplink
```

```
$ cp dsp/export/BIN/DspBios/DA8XX/DA8XXGEM_x/RELEASE/loop.out  /opt/dsplink/
samples/loop
```

Important

Enter the commands shown above in single line

Important

To Build DSPLINK and samples for DSP_BootMode_NoBoot (where GPP does not boot load the DSP) ,

- Change the LOADERNAME to NOLOADER and DOPOWERCTRL to DSP_BootMode_NoBoot in the LINKCFG_dspObject defined in CFG_DA8XXGEM_SHMEM.c.Build the Gpp side DSPLINK
- Define the macro DSP_BOOTMODE_NOBOOT in the COMPONENT file of dsp side sample applications .
- Comment out the dsplink-dio.tci , dsplink-iom.tci file inclusions in the DSP side sample application's TCF/cfg file.
- Build DSP side sample applications.

1.5.3.3. Loading the kernel module: dsplinkk.ko

To load the device driver, login as 'root' and enter following commands on the command prompt.

```
$ cd /opt/dsplink
```

```
$ mknod /dev/dsplink c 230 0
```

```
$ insmod dsplinkk.o
```

This action generates a warning indicating that the kernel module does not contain the GPL license. This warning can be safely ignored.

1.5.3.4. Invoking the application

- In NoBootMode(DSP_BootMode_NoBoot), After flashing the DSP application as part of AIS binary, DSP boots up and the flashed DSP sample application runs on DSP side. From linux console corresponding GPP sample application needs to be run. for example if loop.out is a part of the flashed AIS binary , on linux console, run loop sample. To run the other sample, need to flash the AIS binary again with the corrsponding DSP application. .
- In BootMode(DSP_BootMode_BootPwr), DSP UBL boots ARM UBL and in turn ARM UBL brings U-boot up. Run the sample and the sample loads and boots the DSP with the provided dsp mage(corresponding dsp side sample application)

1.5.3.4.1. Loop sample

To invoke the application enter the following commands:

```
$ cd /opt/dsplink/samples/loop
```

```
$ ./loopgpp loop.out <buffersize> <iterations>   <processor identifier>
```

Note

Argument processor identifier is optional, if it not provided assumed as default processor (zero).

e.g.

```
$ ./loopgpp loop.out 1024 10000
```

Note

For NOLOADER and DSP_BootMode_NoBoot mode for DA8XX, they are constraints in using the args (arguments passed main) on DSP side. Currently DSP side sample application uses hard coded values instead of using the arguments passed to main function of DSP side sample. On DSP side, number of transfers is hard coded to 10000, buffer size is hard coded to 1024.

1.5.3.4.2. Message sample

```
$ cd /opt/dsplink/samples/message
```

```
$ ./messagegpp message.out <number of iterations>  <processor identifier>
```

Note

Argument processor identifier is optional, if it not provided assumed as default processor (zero).

e.g.

```
$ ./messagegpp message.out
```

Note

For NOLOADER and DSP_BootMode_NoBoot mode for DA8XX, they are constraints in using the args (arguments passed main) on DSP side. Currently DSP side sample application uses hard coded values instead of using the arguments passed to main function of DSP side sample. On DSP side, numTransfers is hard coded to 10000.

1.5.3.4.3. Scale sample

```
$ cd /opt/dsplink/samples/scale
```

```
$ ./scalegpp scale.out <buffer size> <iterations>   <processor identifier>
```

Note

Argument processor identifier is optional, if it not provided assumed as default processor (zero).

e.g.

```
$ ./scalegpp scale.out 1024 10000
```

Note

For NOLOADER and DSP_BootMode_NoBoot mode for DA8XX, they are constraints in using the args (arguments passed main) on DSP side. Currently DSP side sample application uses hard coded values instead of using the arguments passed to main function of DSP side sample. On DSP side, number of transfers is hard coded to 1000 and buffer size is hard coded to 1024.

1.5.3.4.4. Ring_IO sample

```
$ cd /opt/dsplink/samples/ring_io
```

```
$ ./ringiogpp ringio.out <RingIO data buffer size in bytes> <number of Bytes to
 transfer>    <processor identifier>
```

Note

Argument processor identifier is optional, if it not provided assumed as default processor (zero).

e.g.

```
$ ./ringiogpp ringio.out 1024 10240
```

Important

By default Ring_IO sample runs in multithread mode. To run the sample in multi process mode ,define RINGIO_MULTI_PROCESS flag in $DSPLINK\gpp\src\samples\ring_io \Linux\COMPONENT file and build the sample

Note

For NOLOADER and DSP_BootMode_NoBoot mode for DA8xx, they are constraints in using the args (arguments passed main) on DSP side. Currently DSP side sample application uses hard coded values instead of using the arguments passed to main function of DSP side sample. On DSP side, buffer size is hard coded to 10240.

1.5.3.4.5. Readwrite sample

```
$ cd /opt/dsplink/samples/readwrite

$ ./readwritegpp readwrite.out <DSP address> <buffer size> <iterations>
 <processor identifier>
```

Note

Argument processor identifier is optional, if it not provided assumed as default processor (zero).

e.g.

```
$ ./readwritegpp readwrite.out 0xE0510000 1024 1000
```

Note

For NOLOADER and DSP_BootMode_NoBoot mode for DA8XX, they are constraints in using the args (arguments passed main) on DSP side. Currently DSP side sample application uses hard coded values instead of using the arguments passed to main function of DSP side sample. On DSP side, number of transfers is hard coded to 1000.

1.5.3.4.6. MPCSXFER sample

```
$ cd /opt/dsplink/samples/mpcsxfer

$ ./mpcsxfergpp mpcsxfer.out <buffer size> <iterations>    <processor
 identifier>
```

Note

The sample can be executed for infinite iterations by specifying the number of iterations as 0.

Note

Argument processor identifier is optional, if it not provided assumed as default processor (zero).

e.g.

Note

For NOLOADER and DSP_BootMode_NoBoot mode for DA8XX, they are constraints in using the args (arguments passed main) on DSP side. Currently DSP side sample application uses hard coded values instead of using the arguments passed to main function of DSP side sample. On DSP side, number of transfers is hard coded to 1000 and buffer size is hard coded to 1024.

```
$ ./mpcsxfergpp mpcsxfer.out 1024 10000
```

1.5.3.4.7. MP_LIST sample

```
$ cd /opt/dsplink/samples/mp_list
```

```
$ ./mplistgpp mplist.out <iterations> <number of elements>    <processor
 identifier>
```

Note

The sample can be executed for infinite iterations by specifying the number of iterations as 0.

Note

Argument processor identifier is optional, if it not provided assumed as default processor (zero).

e.g.

```
$ ./mplistgpp mplist.out 1000 20
```

Note

For NOLOADER and DSP_BootMode_NoBoot mode for DA8XX, they are constraints in using the args (arguments passed main) on DSP side. Currently DSP side sample application uses hard coded values instead of using the arguments passed to main function of DSP side sample. On DSP side, number of iterations is hard coded to 1000 and number of elements is hard coded to 20.

1.5.3.4.8. MESSAGE_MULTI sample

```
$ cd /opt/dsplink/samples/message_multi
```

```
$ ./messagemultigpp messagemulti.out <number of transfers> <Application instance
 number 1 -> MAX_APPS>    <processor identifier>
```

Note

Argument processor identifier is optional, if it not provided assumed as default processor (zero).

e.g.

```
$ for i in 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16;
```

```
do ./messagemultigpp messagemulti.out 10000 $i    & done
```

Note

For NOLOADER and DSP_BootMode_NoBoot mode for DA8XX, they are constraints in using the args (arguments passed main) on DSP side. Currently DSP side sample application uses hard coded values instead of using the arguments passed to main function of DSP side sample. On DSP side, numTransfers is hard coded to 10000.

Note

In earlier versions of DSPLINK(1.61) ,Address of DSPLINK_shmbaseAddres needed to be passed from GPP sample applications. Now sample applications are updated and now GPP samples need not pass this Address.Linker command files of DSP sample applications have been updated updated to inform the DSP application about the location of the DSPLINK shared memory(i.e DSPLINK_shmBaseAddress)

1.5.3.5. Unloading the kernel module: dsplinkk.ko

To unload the device driver, enter following commands on the command prompt.

```
$ cd /opt/dsplink
```

```
$ rmmod dsplinkk
```

```
$ rm /dev/dsplink
```