DSP/BIOS™ LINK

OMAP3530 EVM

LNK 173 USRef

Version 1.65

This page has been intentionally left blank.

IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, modifications, enhancements, improvements, and other changes to its products and services at any time and to discontinue any product or service without notice. Customers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All products are sold subject to TI's terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its hardware products to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by government requirements, testing of all parameters of each product is not necessarily performed.

TI assumes no liability for applications assistance or customer product design. Customers are responsible for their products and applications using TI components. To minimize the risks associated with customer products and applications, customers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any TI patent right, copyright, mask work right, or other TI intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information published by TI regarding third–party products or services does not constitute a license from TI to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. Reproduction of this information with alteration is an unfair and deceptive business practice. TI is not responsible or liable for such altered documentation.

Resale of TI products or services with statements different from or beyond the parameters stated by TI for that product or service voids all express and any implied warranties for the associated TI product or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

Mailing Address:

Texas Instruments

Post Office Box 655303

Dallas, Texas 75265

This page has been intentionally left blank.

TABLE OF CONTENTS

TABLE OF FIGURES

A. INTRODUCTION

1 Purpose

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.

2 Text Conventions

O	This bullet indicates important information. Please read such text carefully.
q	This bullet indicates additional information.
[arg1 | arg2]	In context of the commands, contents enclosed in square brackets are the optional arguments to the command. Different values of these arguments are separated by "|".

3 Terms & Abbreviations

CCS	Code Composer Studio
IPC	Inter Processor Communication
GPP	General Purpose e.g. ARM
DSP	Digital Signal Processor e.g. TMS320C5510
CGTools	Code Gen Tools, e.g. Compiler, Linker, Archiver

4 References

1.	UserGuide_1_0_2.pdf	PSP Linux release 1.0.2 User Guide
2.	Various	Documentation included with the OMAP3530 hardware.

B. INSTALLATION

5 Basic Installation

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

1. Unzip and untar the file dsplink_<version>.tar.gz.

- 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.
- 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.
- 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.

5.1 Installing Standalone DSP/BIOS™ and CGTools

For compilation of DSP-side sources and applications for OMAP3530, the CGTools version 6.0.18 can be used. This release has been validated with DSP/BIOS™ version 5.32.04.

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.

https://www-a.ti.com/downloads/sds_support/targetcontent/bios/index.html

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.

5.2 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.

1. Copy and untar make-3.81.tar.gz to your home directory.
2. cd to make-3.81 directory

3. 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

4. Type `make' and press enter to compile the package.

5. Optionally, type `./make check' and press enter to run any self-tests that come with the package.

6. Type `make install' and press enter to install the programs and any data files and documentation.

7. For additional details refer to INSTALL file located under make-3.81 directory.

6 Creating development workspace

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

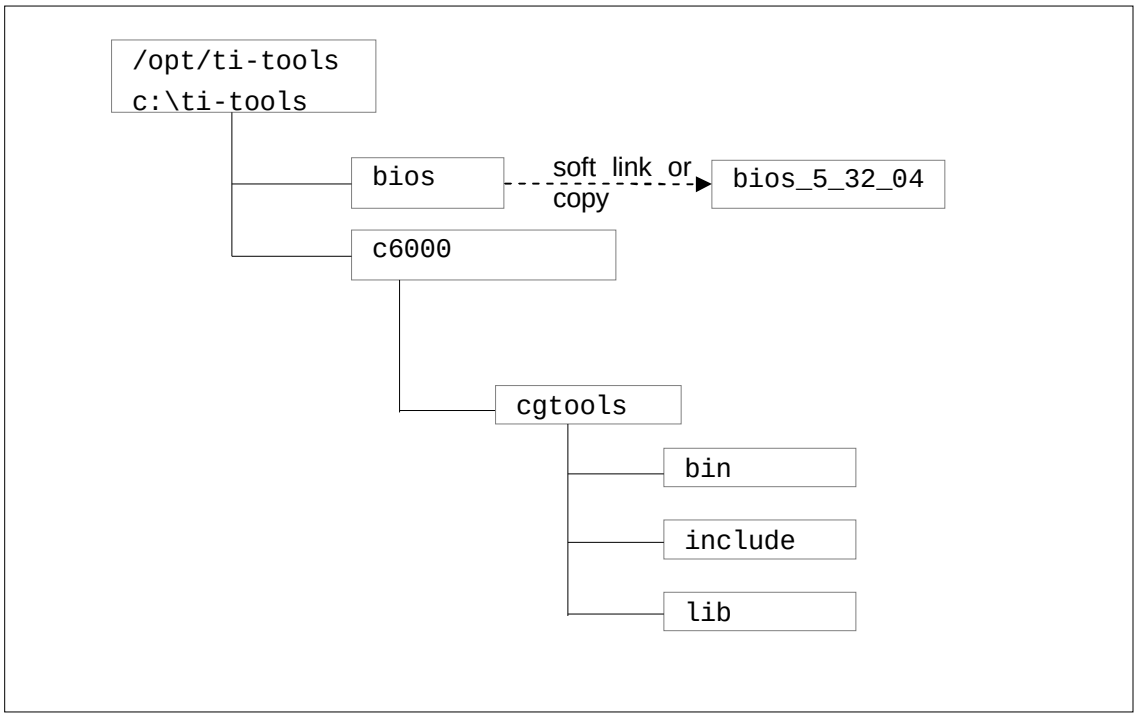

Figure 1. Expected directory structure on development host

- 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 is installed in the 'ti-tools/c6000' directory on the <ROOT-DRIVE>.

- 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 is installed in the 'ti-tools\c6000\' directory on the <ROOT-DRIVE>.

- 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.

7 Setting up Linux Workstation

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

1. The workstation is running on Redhat Linux version 9.0
2. Services telnetd, nfsd, ftpd are configured on this workstation.
3. The workstation is assigned a fixed IP address.

- The release package has been tested on RedHat Linux version 9.0. You may be able to build on a different version depending on the compatibility of the build tools in your version with version 9.0.
- A fixed IP address is preferred, as the IP address needs to be specified while booting the target board. This allows the boot loader configuration to be saved in flash.

This document and the scripts included in the release assume the following directory structure in your home directory on the Linux workstation:

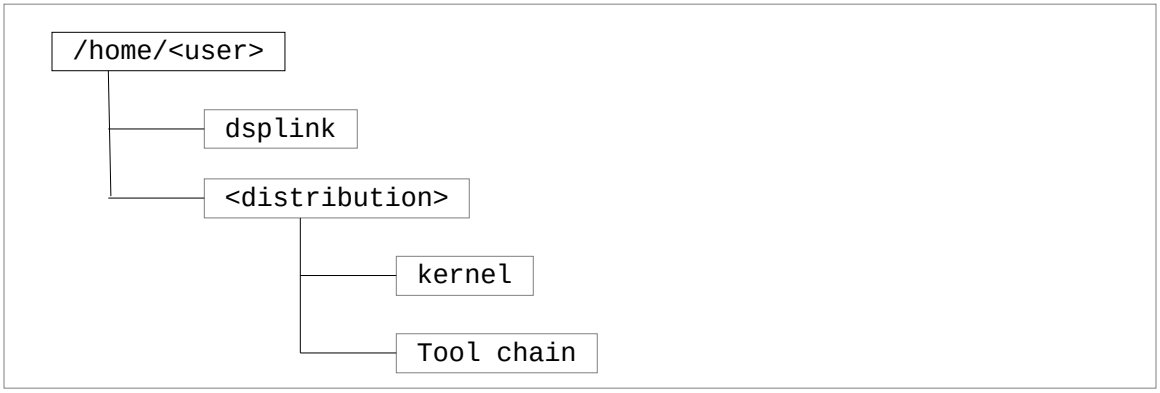

Figure 2. Expected directory structure on the Linux workstation

In this diagram:

§ <user> represents the actual user name

§ <distribution> represents the actual distribution of Linux being used.

This section assumes that REL_OMAP35x_02.01.00.02 release is installed on the development workstation at `${HOME}/omap3530`

7.1 Building the kernel

7.1.1 Tool chain

1. The tool chain arm-2009q1-203 - arm-2009q1-203-arm-none-linux-gnueabi.bin can be downloaded from :-

   http://www.codesourcery.com/sgpp/lite/arm/portal/subscription?@template=lite

   Install the tool chain by running and installing the binary executable.

```
$ ./arm-2009q1-203-arm-none-linux-gnueabi.bin
$ export PATH=$PATH:${HOME}/omap3530/arm-2009q1-203/bin
```

7.1.2 Kernel

1. Un tar the kernel that comes with the 03.00.00.05 release. Build the kernel according to the PSP Linux User Guide document. Refer to the PSP u-boot related documentation on how to build the mkimage utility which is needed for creation of uImage. Along with the tool chain, location of mkimage utility should also be there in the path.

```
$ tar –zxvf OMAP35x-PSP-SDK-03.00.00.05.tgz
```

7.2 U-Boot boot-loader

Please refer the user guide to load the u.boot and x-loader to the target.

7.3 LPM

1. Download the ti_bios_power,omap3530.tar local_power_manager_1_23_00_01.tar.gz present at http://www.sanb.design.ti.com/web/power/local_power_manager_1_23_00_01/

   Untar the LPM source code (ti_bios_power,omap3530.tar) and module (ti_bios_power,omap3530_mod.tar).

```
$ tar -xvf ti_bios_power,omap3530.tar
```

2. Copy the LPM module (lpm_omap3530.ko) from $HOME/LPM/omap3530/lpm to target NFS. If the insmod step is giving error while inserting the module then the existing binaries needs to be re-compile with proper OS and tool chain.

   To build the LPM module modify the Makefile present at $HOME/omap3530/lpm/Makefile

```
#Export the toolchain path
PATH=$PATH:${HOME}/arm-2009q1-203/bin

cd $HOME/LPM/omap3530/lpm
vi Makefile

# Set PROFILE to DEBUG or RELEASE
PROFILE = RELEASE
LINK_PATH = $(LINK_DIR)/gpp/export/BIN/Linux/OMAP3530/RELEASE

# Edit the LSP and toolchain path
# LINUXKERNEL_INSTALL_DIR = the Linux kernel source directory
# MVTOOL_PREFIX = the toolchain directory and decorated name
prefix ($(HOME)/arm-2009q1-203/bin/arm-none-linux-gnueabi-)
```

```
# DSPLINK_REPO = the repository which contains DSPLINK

make
```

3. Download the local_power_manager_1_23_00_01.tar from same location. Copy LPM On (lpmON.x470uC) and OFF (lpmOFF.x470uC) binaries in your target file system.

```
For DEBUG build:
local_power_manager_1_23_00_01/packages/ti/bios/power/test/bin/ti_platforms_evm3530/linux/debug
For RELEASE build:
local_power_manager_1_23_00_01/packages/ti/bios/power/test/bin/ti_platforms_evm3530/linux/release
```

7.4 Enable TFTP for downloading the kernel image to target

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

a. TFTP

b. Serial Port

This section configures the Linux development host as a TFTP server.

Modify the '`xinet.d/tftp`' file to enable TFTP:

1. Make the following changes:

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

2. Restart the network service

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

O 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.

7.5 Create target file system

The target device needs a file system to boot from. The file system can be exported to the target through NFS.

7.5.1 Exporting target file system through NFS

A directory on the development host can be setup and exported for this purpose.

1. $(HOME)/REL_OMAP35x_LINUX_PSP_1.0.2/PSP_1_0_2/bin contains the NFS file system nfs.tar.gz.

```
tar xzvf nfs.tar.gz
```

O 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 ~/omap3530/target can be a soft link to the actual location.

2. libpthread libraries required for DSP/BIOS™ LINK are not available by default within the target file system. Copy this from the tool-chain.

```
$ cp ${HOME}/omap3530/arm-2007q3/arm-none-linux-
gnueabi/libc/lib/libpthread* ~/omap3530/target/lib
```

- This step is not required if libpthread libraries are available by default in the target file system.

3. The directory ~/omap3530/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>/omap3530/target  *(rw,no_root_squash)
```

- Replace "*/home/<user>*" in the path above with the actual path of your home directory on the development workstation.

8 Configuring CCS

8.1 OMAP3530

To use CCS for debugging the DSP side application, you will need to configure CCS to use both ARM and DSP with the OMAP3530.

- CCS can attach to only ARM in the beginning. It can attach to the DSP only after the ARM-side application releases it from reset through a call to *PROC_Start ()*.

C. WORKING ON TARGET PLATFORM

9 OMAP3530

9.1 Configuring Kernel Parameters

DSP/BIOS™ LINK requires a few specific arguments to be passed to the Linux kernel during boot up. 2MB of memory is used by DSP/BIOS™ LINK for communication between GPP and DSP, and for DSP external memory. This must be reserved by specifying 2MB less as available for the Linux kernel for its usage.

9.2 Configure the DSP/BIOS LINK

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=OMAP3530 --nodsp=1 --
dspcfg_0=OMAP3530SHMEM --dspos_0=DSPBIOS5XX --gppos=OMAPLSP  --
comps=ponslrmc
```

9.3 Building DSP/BIOS LINK

Please refer the User Guide for generic instructions to configure and build DSPLink

9.4 Using LPM

When cache is enabled, a MMU fault is seen when one application is run followed by a run of another application. This issue is not seen when one application is run followed by a run of the same application. To overcome this issue the LPM module is used to power cycle the EVM between runs.

9.5 Build the DSP/BIOS LINK

Following are the steps to build the DSP/BIOS Link using kernel build system.

1. Run the configuration command.

   ```
   perl dsplinkcfg.pl  --platform=OMAP3530 --nodsp=1 --
   dspcfg_0=OMAP3530SHMEM --dspos_0=DSPBIOS5XX --gppos=OMAPLSP  --
   comps=ponslrmc
   ```

2. Modify the distribution file to update the BASE_BUILDOS (Base directory for the GPP OS) and BASE_TOOLCHAIN (Base for toolchain)

   ```
   vi $(DSPLINK)/make/Linux/omap3530_2.6.mk
   ```

3. Modify the Rules.mk file to update the KERNEL_DIR (Base directory for the GPP OS) and TOOL_PATH (Base for toolchain)

   ```
   vi $(DSPLINK)/gpp/src/Rules.mk
   KERNEL_DIR    :=
   $(HOME)/REL_OMAP35x_03.00.00.03/PSP_03.00.00.03/src/linux-
   03.00.00.03
   TOOL_PATH     := $(HOME)/arm-2009q1-203/bin
   ```

4. Build the GPP sources and samples

   ```
   cd $(DSPLINK)/gpp/src/api
   ```

```
make –s release
make –s debug
or
make -s
cd $(DSPLINK)/gpp/src
make –s release
make –s debug
cd $(DSPLINK)/gpp/src/samples
make –s release
make –s debug
or
make -s
```

5. Build the DSP sources and samples

```
cd $(DSPLINK)/dsp/src
make –s release
make –s debug
cd $(DSPLINK)/dsp/src/samples
make –s release
make –s debug
```

10 Running the sample applications

Seven sample applications are provided with DSP/BIOS™ LINK for the OMAP3530 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 message sample. However, similar instructions can be used for the other applications also.

The steps for execution of the samples are given below for execution with Linux running on the GPP.

10.1 Copying files to target file system

The generated binaries on the GPP side and DSP side and the data files must be copied to the target directory. The commands below demonstrate this for the 'message' sample application as reference. Appropriate sample directory name must be used for other sample applications.

GPP Side

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

```
$ cd ~/dsplink
$ cp gpp/export/BIN/Linux/OMAP/3530/DEBUG/messagegpp
../omap3530/target/opt/dsplink/samples/message
$ cp gpp/export/BIN/Linux/OMAP/3530/DEBUG/dsplinkk.ko
../omap3530/target/opt/dsplink/
```

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

```
$ cd ~/dsplink
$ cp gpp/export/BIN/Linux/OMAP/3530/RELEASE/messagegpp
../omap3530/target/opt/dsplink/samples/message
$ cp gpp/export/BIN/Linux/OMAP/3530/RELEASE/dsplinkk.ko
../omap3530/target/opt/dsplink/
```

- Enter the commands shown above in single line.
- 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.

DSP Side

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

After the binaries have been built, they must be copied into the target file system. If the binaries are generated on Windows PC, any FTP client can be used for transferring these to the target file system.

For executing the DEBUG build:

1. Copy the following file into the directory ~/omap3530/target/opt/dsplink/samples/message:

   ```
   dsplink/dsp/export/BIN/DspBios/OMAP3530/OMAP3530_x/DEBUG/message
   .out
   ```

For executing the RELEASE build:

1. Copy the following file into the directory ~/omap3530/target/opt/dsplink/samples/message:

```
dsplink/dsp/export/BIN/DspBios/OMAP3530/OMAP3530_x/RELEASE/message.out
```

10.2 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
$ insmod dsplinkk.ko
$ mknod /dev/dsplink c 230 0
```

10.3 Loading the LPM module

For executing the debug build:

1. Copy the LPM module (lpm_omap3530.ko) to your target file system.

```
cp $HOME/LPM/omap3530/lpm/lpm_omap3530.ko
~/omap3530/target/opt/dsplink/
```

2. Copy the LPM modules lpmON.x470uc and lpmOFF.x470uc) binaries to your target file system.

```
cp
$HOME/LPM/ti/bios/power/test/bin/ti_platforms_evm3530/linux/debug/lpmON.x470uc ~/omap3530/target/opt/dsplink/
cp
$HOME/LPM/ti/bios/power/test/bin/ti_platforms_evm3530/linux/debug/lpmOFF.x470uc ~/omap3530/target/opt/dsplink/
```

For executing the release build:

1. Copy the LPM module (lpm_omap3530.ko) to your target file system.

```
cp $HOME/LPM/omap3530/lpm/lpm_omap3530.ko
~/omap3530/target/opt/dsplink/
```

2. Copy the LPM modules lpmON.x470uc and lpmOFF.x470uc) binaries to your target file system.

```
cp
$HOME/LPM/ti/bios/power/test/bin/ti_platforms_evm3530/linux/release/lpmON.x470uc ~/omap3530/target/opt/dsplink/
cp
$HOME/LPM/ti/bios/power/test/bin/ti_platforms_evm3530/linux/release/lpmOFF.x470uc ~/omap3530/target/opt/dsplink/
```

Load the kernel module

```
# insert lpm
$ insmod lpm_omap3530.ko
$ rm -f /dev/lpm0
$ mknod /dev/lpm0 c `awk "/lpm/ {print \\$1}" /proc/devices` 0
```

10.4 Invoking the application

10.4.1 Loop sample

To invoke the application enter the following commands:

```
# run Loop sample
$ cd /opt/dsplink/samples/loop
$ ./loopgpp loop.out <buffersize> <iterations> <processor identifier>
```

- The sample can be executed for infinite iterations by specifying the number of iterations as 0.
- Argument processor identifier is optional, if it not provided assumed as default processor (zero).

e.g.

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

10.4.2 Message sample

```
# LPM off
# LPM on
# run message sample
$ cd /opt/dsplink/samples/message
$ ./messagegpp message.out <number of iterations> <processor
identifier>
```

- The sample can be executed for infinite iterations by specifying the number of iterations as 0.
- Argument processor identifier is optional, if it not provided assumed as default processor (zero).

e.g.

```
$ ./lpmOFF.x470uc
$ ./lpmON.x470uc
$ ./messagegpp message.out 10000
```

10.4.3 Scale sample

```
# LPM off
# LPM on
# run scale sample
$ cd /opt/dsplink/samples/scale
$ ./scalegpp scale.out <buffer size> <iterations> <processor
identifier>
```

- The sample can be executed for infinite iterations by specifying the number of iterations as 0.
- Argument processor identifier is optional, if it not provided assumed as default processor (zero).

e.g.

```
$ ./lpmOFF.x470uc
$ ./lpmON.x470uc
$ ./scalegpp scale.out 1024 10000
```

10.4.4 Ring_IO sample

```
# LPM off
# LPM on
# run ringio sample
$ cd /opt/dsplink/samples/ring_io
$ ./ringiogpp ringio.out <RingIO data buffer size> <Number of bytes to
transfer> <processor identifier>
```

- The sample creates two RingIOs with the data buffer size equal to RingIO Data Buffer Size specified through the command line arguments. The minimum value that can be specified is 1024 Bytes. The maximum value depends on the size of the memory configured for DSPLINK.
- The RingIO Data buffer size can be given between 1k bytes to 200k bytes with the Default memory configuration provided with the link.
- The sample can be executed infinitely by specifying Number of Bytes to transfer as zero.
- Argument processor identifier is optional, if it not provided assumed as default processor (zero).

e.g.

```
$ ./lpmOFF.x470uc
$ ./lpmON.x470uc
$ ./ringiogpp ringio.out 10240 10240
$ ./ringiogpp ringio.out 10240 10752
$ ./ringiogpp ringio.out 9472 10240
```

10.4.5 Readwrite sample

```
# LPM off
# LPM on
# run readwrite sample
$ cd /opt/dsplink/samples/readwrite
$ ./readwritegpp readwrite.out <DSP address> <buffer size> <iterations>
<processor identifier>
```

- The sample can be executed for infinite iterations by specifying the number of iterations as 0.
- Argument processor identifier is optional, if it not provided assumed as default processor (zero).

e.g.

```
$ ./lpmOFF.x470uc
$ ./lpmON.x470uc
```

```
$ ./readwritegpp readwrite.out 2280587264 1024 10000
$ ./readwritegpp readwrite.out 276791296 1024 10000
$ ./readwritegpp readwrite.out 284180480 1024 10000
```

10.4.6 MPCSXFER sample

```
# LPM off
# LPM on
# run mpcsxfer sample
$ cd /opt/dsplink/samples/mpcsxfer
$ ./mpcsxfergpp mpcsxfer.out <buffer size> <iterations> <processor
identifier>
```

- The sample can be executed for infinite iterations by specifying the number of iterations as 0
- Argument processor identifier is optional, if it not provided assumed as default processor (zero).

e.g.

```
$ ./lpmOFF.x470uc
$ ./lpmON.x470uc
$ ./mpcsxfergpp mpcsxfer.out 1024 10000
```

10.4.7 MP_LIST sample

```
# LPM off
# LPM on
# run mp_list sample
$ cd /opt/dsplink/samples/mp_list
$ ./mplistgpp mplist.out <iterations> <number of elements> <processor
identifier>
```

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

e.g.

```
$ ./lpmOFF.x470uc
$ ./lpmON.x470uc
$ ./mplistgpp mplist.out 1000 20
```

10.4.8 MESSAGE_MULTI sample

```
# LPM off
# LPM on
# run message_multi sample
$ cd /opt/dsplink/samples/message_multi
$ ./messagemultigpp messagemulti.out <number of transfers> <Application
instance number 1 -> MAX_APPS> <processor identifier>
```

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

e.g.

```
$ ./lpmOFF.x470uc
$ ./lpmON.x470uc
$ ./messagemultigpp messagemulti.out 10000 1
```

10.5 Unloading the kernel module: dsplinkk.ko

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

```
$ cd /opt/dsplink
$ rmmod lpm_omap3530.ko
$ rmmod dsplinkk
```

D. ADDITIONAL INFORMATION

None.