DSP/BIOS™ LINK

DM6446/DM6467/DM6467T Media Processor

LNK 110 USR

1.65

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 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 Processor e.g. ARM
DSP	Digital Signal Processor e.g. TMS320C5510
CGTools	Code Gen Tools, e.g. Compiler, Linker, Archiver

4 References

1.	Montavista Linux Professional Edition 5.0 - Installation Guide	InstallGuide-Pro50_highres.pdf
2.	REL_LSP_02_00_00_140.tgz	Release notes and User Guide.

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_1.65.tar.gz.

O 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.

O 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.

O 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

This release will not work with BIOS version earlier than 5.32.01.

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:

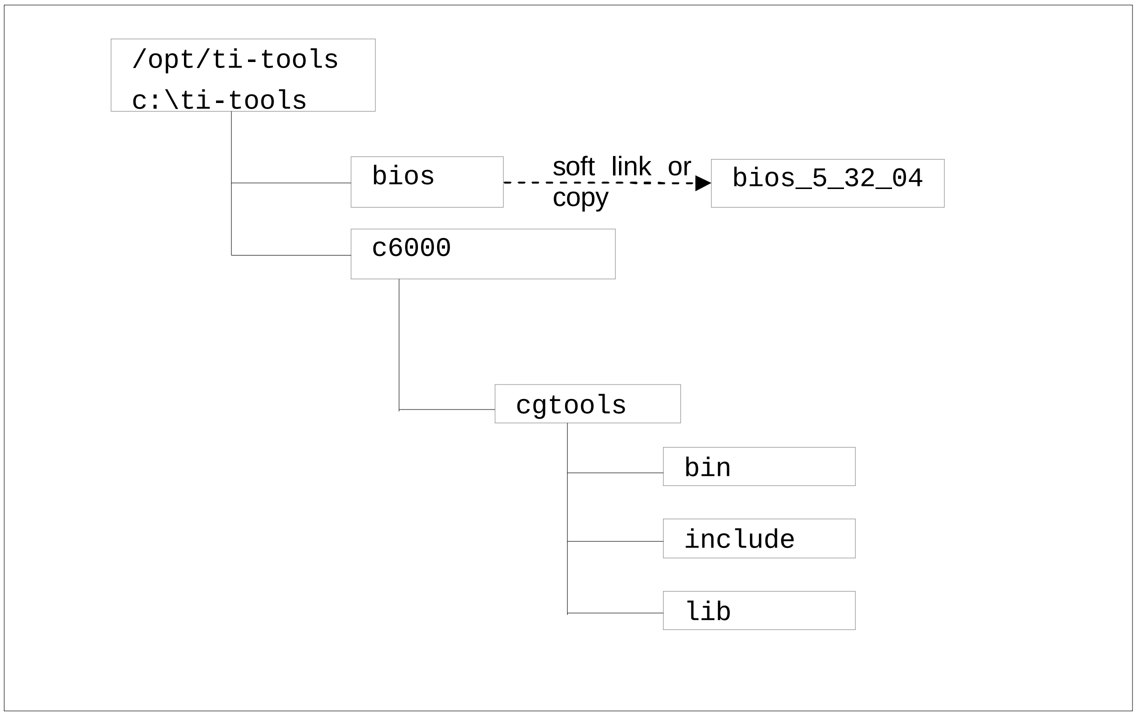

Figure 1. Expected directory structure on development host

- 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

7.1 Setting up workstation

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

1. Host PC with RedHat Enterprise Linux version 9.0 to install the MVL uclibc package.
2. Services telnetd, nfsd, ftpd are configured on this workstation.
3. The workstation is assigned a fixed IP address.
4. 3GB Space is required for the installation of tool chain for a single LSP of a single architecture type.

O 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.

O 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:

§ <user> represents the actual user name

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

7.1.1 Building the kernel

7.1.2 Building the kernel for DM6446 for Glibc 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
```

2. This step is for DM6446 part. Un tar the kernel that comes with the DM6467 LSP GIT 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.

```
$ make ARCH=arm CROSS_COMPILE=arm-none-linux-gnueabi- distclean
$ make ARCH=arm CROSS_COMPILE=arm-none-linux-gnueabi-
davinci_dm644x_defconfig
$ make ARCH=arm CROSS_COMPILE=arm-none-linux-gnueabi- uImage
```

7.1.3 Enable TFTP for downloading the kernel image to target

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

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.

7.1.4 Create target file system

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.

7.1.5 Enable TFTP for downloading the kernel image to target

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

c. TFTP

d. 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.

7.1.6 Create target file system

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.

7.2 Setting up Linux for DM6467 and DM6467T parts

7.2.1 Building the kernel for DM6467 and DM6467T for Glibc 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
   ```

2. This step is for DM6467 part. Un tar the kernel that comes with the DM6467 LSP GIT 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.

   ```
   $ export CROSS_COMPILE=arm-none-linux-gnueabi-
   #Note that this will configure the build for DM6467 594MHz part
   #Also note that the kernel built with this config can be used on
   #all supported DaVinci SoCs
   $ make davinci_dm646x_defconfig
   ```

2. This step is for DM6467T part. Un tar the kernel that comes with the DM6467T LSP GIT 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.

   ```
   $ export CROSS_COMPILE=arm-none-linux-gnueabi-
   #Note that this will configure the build for DM6467T (1GHz) part
   #Kernel built with this configuration should be used only on
   #DM6467T EVM
   $ make davinci_dm646x_1ghz_defconfig

   $ make uImage
   ```

8 Configuring CCS

8.1 DM6446 EVM and DM6467 EVM

To use CCS for debugging the DSP side application, you will need to configure CCS to use both ARM and DSP with the DM6446 EVM. The EVM must be configured to use the DSP self-boot mode.

- 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 DM6446 EVM and DM6467 EVM

9.1 Setting up Linux environment

Refer to the PSP Linux release/install guide document for instructions on setting up the Linux execution environment for the DM6446 EVM.

9.2 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.

9.3 Configure the DSP/BIOS LINK for DM6446 or DM6467

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

DM6446 (Pro 5 uclibc)

```
perl dsplinkcfg.pl --platform=DAVINCI --nodsp=1 --dspcfg_0=DM6446GEMSHMEM --
dspos_0=DSPBIOS5XX --gppos=DM6446LSPuc --comps=ponslrmc
```

DM6446 (Pro 5 glibc)

```
perl dsplinkcfg.pl --platform=DAVINCI --nodsp=1 --dspcfg_0=DM6446GEMSHMEM --
dspos_0=DSPBIOS5XX --gppos=DM6446LSP --comps=ponslrmc
```

DM6446 (Pro 5 glibc, Bios 6.xx)

```
perl dsplinkcfg.pl --platform=DAVINCI --nodsp=1 --dspcfg_0=DM6446GEMSHMEM --
dspos_0=DSPBIOS6XX --gppos=DM6446LSP --comps=ponslrmc
```

DM6446 (Pro 5 uclibc, Bios 6.xx)

```
perl dsplinkcfg.pl --platform=DAVINCI --nodsp=1 --dspcfg_0=DM6446GEMSHMEM --
dspos_0=DSPBIOS6XX --gppos= DM6446LSPuc --comps=ponslrmc"
```

DM6467 (GIT LSP with glibc)

```
perl dsplinkcfg.pl  --platform=DAVINCIHD --nodsp=1 --
dspcfg_0=DM6467GEMSHMEM --dspos_0=DSPBIOS5XX --gppos=DM6467LSP --
comps=ponslrmc
```

DM6467 (GIT LSP with uclibc)

```
perl dsplinkcfg.pl  --platform=DAVINCIHD --nodsp=1 --
dspcfg_0=DM6467GEMSHMEM --dspos_0=DSPBIOS5XX --gppos=DM6467LSPuc --
comps=ponslrmc
```

```
DM6467T (GIT LSP with glibc)
perl dsplinkcfg.pl  --platform=DAVINCIHD --nodsp=1 --
dspcfg_0=DM6467GEMSHMEM --dspos_0=DSPBIOS5XX --gppos=DM6467LSP --
comps=ponslrmc

DM6467T (GIT LSP with uclibc)
perl dsplinkcfg.pl  --platform=DAVINCIHD --nodsp=1 --
dspcfg_0=DM6467GEMSHMEM --dspos_0=DSPBIOS5XX --gppos=DM6467LSPuc --
comps=ponslrmc
```

- For details please refer user guide.
- In Case of BIOS6 for DM6446 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"

9.4 Building DSPLink

Please follow generic instructions in the User Guide to configure and build DSPLink.

10 Running the sample applications

Seven sample applications are provided with DSPLINK for the DM6446 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.

The steps for execution of the samples are given below with DM6446 variant of the DM6446 platform. Similar instructions can be used for the DM6467 variant.

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/DAVINCI/DEBUG/messagegpp
~/montavista/target/opt/dsplink/samples/message
$ cp gpp/export/BIN/Linux/DAVINCI/DEBUG/dsplinkk.ko
~/montavista/target/opt/dsplink/
```

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

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

O Enter the commands shown above in single line.

O 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

`~/montavista/target/opt/dsplink/samples/message`:

`dsplink/dsp/export/BIN/DspBios/DAVINCI/DM6446GEM_x/DEBUG/message.out`

For executing the RELEASE build:

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

   `dsplink/dsp/export/BIN/DspBios/Davinci/DM6446GEM_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
```

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

10.3 Invoking the application

10.3.1 Loop sample

To invoke the application enter the following commands:

```
$ cd /opt/dsplink/samples/loop
$ ./loopgpp loop.out <buffersize> <iterations> < DSP Processor Id >
```

- The sample can be executed for infinite iterations by specifying the number of iterations as 0.
- The DSP Processor Id is optional. By default it is zero.

e.g.

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

10.3.2 Message sample

```
$ cd /opt/dsplink/samples/message
$ ./messagegpp message.out <number of iterations> < DSP Processor Id >
```

- The sample can be executed for infinite iterations by specifying the number of iterations as 0.
- The DSP Processor Id is optional. By default it is zero.

e.g.

```
$ ./messagegpp message.out 10000
```

10.3.3 Scale sample

```
$ cd /opt/dsplink/samples/scale
$ ./scalegpp scale.out <buffer size> <iterations> < DSP Processor Id >
```

- The sample can be executed for infinite iterations by specifying the number of iterations as 0.
- The DSP Processor Id is optional. By default it is zero.

e.g.

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

10.3.4 Ring_IO sample

```
$ cd /opt/dsplink/samples/ring_io
$ ./ringiogpp ringio.out <RingIO data buffer size in bytes> < number of
Bytes to transfer> < DSP Processor Id >
```

- The sample can be executed for infinite iterations by specifying the number of Bytes to transfer as 0.
- The DSP Processor Id is optional. By default it is zero.

e.g.

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

10.3.5 Readwrite sample

```
$ cd /opt/dsplink/samples/readwrite
$ ./readwritegpp readwrite.out <DSP address> <buffer size> <iterations>
< DSP Processor Id >
```

- The sample can be executed for infinite iterations by specifying the number of iterations as 0.
- The DSP Processor Id is optional. By default it is zero.

e.g.

```
For DM6446
$ ./readwritegpp readwrite.out 2414804992 1024 10000
$ ./readwritegpp readwrite.out 293601280 1024 10000
$ ./readwritegpp readwrite.out 300957696 1024 10000
For DM6467
$ ./readwritegpp readwrite.out 2414804992 1024 10000
$ ./readwritegpp readwrite.out 293699584 1024 10000
```

10.3.6 MPCSXFER sample

```
$ cd /opt/dsplink/samples/mpcsxfer
$ ./mpcsxfergpp mpcsxfer.out <buffer size> <iterations> < DSP Processor
Id >
```

- The sample can be executed for infinite iterations by specifying the number of iterations as 0
- The DSP Processor Id is optional. By default it is zero.

e.g.

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

10.3.7 MP_LIST sample

```
$ cd /opt/dsplink/samples/mp_list
$ ./mplistgpp mplist.out <iterations> <number of elements> < DSP
Processor Id >
```

- The DSP Processor Id is optional. By default it is zero.

e.g.

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

10.3.8 MESSAGE_MULTI sample

```
$ cd /opt/dsplink/samples/message_multi
$ ./messagemultigpp messagemulti.out <number of transfers> <Application
instance number 1 -> MAX_APPS> < DSP Processor Id >
```

- The DSP Processor Id is optional. By default it is 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
```

10.4 Unloading the kernel module: dsplinkk.ko

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

```
$ cd /opt/dsplink
$ rmmod dsplinkk
```

D. ADDITIONAL INFORMATION

None.