DSP/BIOS™ Link

Installation Guide

1.64.00.03

Published on 20th OCT 2009

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.

TI products are not authorized for use in safety-critical applications (such as life support) where a failure of the TI product would reasonably be expected to cause severe personal injury or death, unless officers of the parties have executed an agreement specifically governing such use. Buyers represent that they have all necessary expertise in the safety and regulatory ramifications of their applications, and acknowledge and agree that they are solely responsible for all legal, regulatory and safety-related requirements concerning their products and any use of TI products in such safety-critical applications, notwithstanding any applications-related information or support that may be provided by TI. Further, Buyers must fully indemnify TI and its representatives against any damages arising out of the use of TI products in such safety-critical applications.

TI products are neither designed nor intended for use in military/aerospace applications or environments unless the TI products are specifically designated by TI as military-grade or "enhanced plastic." Only products designated by TI as military-grade meet military specifications. Buyers acknowledge and agree that any such use of TI products which TI has not designated as military-grade is solely at the Buyer's risk, and that they are solely responsible for compliance with all legal and regulatory requirements in connection with such use.

TI products are neither designed nor intended for use in automotive applications or environments unless the specific TI products are designated by TI as compliant with ISO/TS 16949 requirements. Buyers acknowledge and agree that, if they use any non-designated products in automotive applications, TI will not be responsible for any failure to meet such requirements.

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 PCI based DSP Device from the new DSP/BIOS™ Link.

How to Use This Manual

This document includes the following chapters:

- Chapter 1, *Install Guide for DM648* - describes the steps required to install and run samples for DM648.
- Chapter 2, *Install Guide for DM6437* - describes the steps required to install and run samples for DM6437.

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

DSP/BIOS™ is a trademark of Texas Instruments Incorporated.

All other trademarks are the property of the respective owner.

Install Guide for DM648

Abstract

This chapter describes how to install and run samples for DM648 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 1.64.00.03 dated OCT 20, 2009.

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

1	Evaluation Module (EVM) for the DM648 Quick Start Installation 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_1_64_00_03.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™ 5.32.04 and CGTools

For compilation of DSP-side sources and applications, 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
DSP/BIOS™ and CGTOOLS

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:

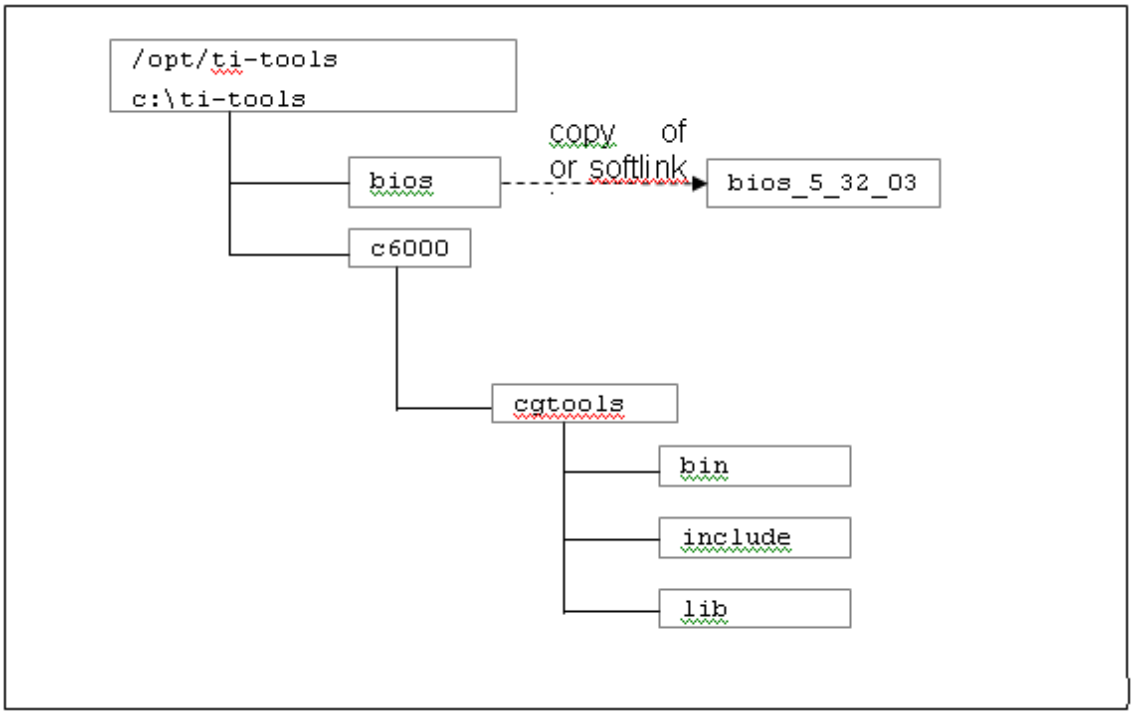

Figure 1.1. Development Workspace

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.
- The workstation must have at least 512 MB RAM, and 1 GHz CPU.

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 development workspace

This document and the scripts included in the release assume that DSP/BIOS™ LINK sources have been installed in /home/<user>/dsplink directory.

1.3.2. Creating target system

This release of DSP/BIOS™ LINK for DM648 has validated on Linux kernel versions 2.6.21. It is recommended to use this version to use DSP/BIOS™. This release of DSP/BIOS™ LINK for DM648 requires big physical area patch for both kernel version. The following steps can be performed to get the kernel patched and DSP/BIOS™ LINK running:

1.3.1. Linux 2.6.21

- Acquiring the Linux kernel sources:

  The kernel sources can be installed on the build machine through the kernel source package. This package can be obtained from GNU kernel repository site (www.kernel.org). The link given below can also be used to download the kernel sources for Linux:
  Kernel Sources tarball

- Transfer this package to the Linux build machine on to a work directory in home area. The tar utility must be used to explode (untar) this package

  ```
  $tar -xvjf linux-2.6.21.tar.bz2
  ```

- The patch for big physical area can downloaded from:

Big Physical Patch

```
$tar -xvjf linux-2.6.21.tar.bz2
```

Transfer the patch file kernel-2.6.21-bigphysarea.patch to the work directory. Change the current directory to the Linux source work directory

```
$cd /home/<user>/work/linux.2.6.21
```

Now patch the Linux source.

```
$patch -p1 < ../kernel-2.6.21-bigphysarea.patch
```

Important

Enter the commands shown above in single line

- The Linux sources must be configured to include the patch before build. The steps for configuration are:
  Make mrproper (This will erase any .config file) This cleans out the configuration files and any object files an older version might have.

```
$make mrproper
```

The next step is optional, depending on if you want to keep your old configuration or base your new kernel on your old configuration and add the new options found in the new kernel. If you want to base your new kernel on your old one, you can copy the config file from boot directory of the build machine to the Linux source root. $cp /boot/config-2.6.XX-X .config and run make command as follow: 1)

```
$make oldconfig
```

In a command line configuration system, the new option to select the big physical area patch is prompted. Press 'y' to continue. Or The other ways to complete configuration are 2)

```
$make config
```

Or 3)

```
$make menuconfig
```

Or 4)

```
$make xconfig
```

In a GUI configuration system, the option to select the big physical area patch is under processor feature and options menu. In a command line configuration system, the option to select the big physical area patch is prompted.

Important

The above steps will ensure that the patched kernel is configured.

Important

The uniprocessor kernel must be selected even if it is a multicore-CPU or multiprocessor machine.

- Make the dependencies. This ensures all the dependencies like include files are in place

```
$make dep
```

- Once configuration step is successfully completed, compile the Linux sources:

```
$make bzImage
```

This could take some time and should build without any error. Upon successful completion, Linux kernel image (bzImage) will be built under <linux-2.6.21>/arch/i386/boot directory. (Assuming you are working on Intel based PC).

- Build and install the modules required by this kernel image.

```
$make modules && make modules_install
```

This step will build the modules and install it in /lib/modules directory.

- Install the built kernel module.

```
$make install
```

- The grub configuration file will now be updated with the new kernel configuration.

The grub file will need to be updated to allocate 64 MB for big physical area memory allocations. You have to add a new line like below:

```
kernel /vmlinuz-2.6.21-bigphys ro root=/dev/VolGroup01/LogVol00 console=tty0
 console=ttyS0,115200n8 selinux=0 bigphysarea=16384
```

Edit the grub configuration file for updating the entry for the patched kernel.

```
$vi /etc/grub.conf

title fedora7 (2.6.21-bigphys)

root (hd0,4)

kernel /vmlinuz-2.6.21-bigphys ro root=/dev/VolGroup01/LogVol00 console=tty0
 console=ttyS0,115200n8 selinux=0 bigphysarea=16384

initrd /initrd-2.6.21-bigphys.img
```

Important

Enter the commands shown above in single line

Important

To increase the bigphys area change the bigphysarea value in the above command.e.g bigphysarea=65536

On rebooting the kernel, a new option to boot with the patch will be present on grub menu.

1.4. Configuring CCS

Note

CCS 3.3 can be used for working with DM648. It may need to be configured to use the XDS510 debugger with DM648

1.5. WORKING ON TARGET PLATFORM

1.5.1. Running the sample applications

Nine sample applications are provided with DSPLINK for the DM648 platform. All the sample applications are described in detail in the user guide. This section describes the way to execute the sample applications

Note

While building the kernel module (dsplinkk.ko), Build generates the following warnings due to issues in kernel header files

```
warning: pointer targets in passing argument 1 of native_cpuid differ in
 signedness
```

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

1.5.1.1. Configure the DSP/BIOS LINK for DM648

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=LINUXPC --nodsp=1 --dspcfg_0=DM648PCI --
dspos_0=DSPBIOS5XX --gppos=RHEL4 --comps=ponslrmc
```

Important

Enter the commands shown above in single line

Important

For details please refer user guide.

1.5.1.2. Copying files to target file system

1.5.1.2.1. GPP Side

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

```
$ cd ~/dsplink
```

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

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

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

Important

Enter the commands shown above in single line

Important

The generated kernel module has different names based on the kernel version being used. I.e. for 2.6.x series kernels the generated kernel module's extension is .ko and for 2.4.x series kernels the generated kernel module's extension is .o. The above commands copy the kernel module to the target filesystem so the appropriate command must be used if the 'wildcard' shown above is not used

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

1.5.1.2.2. DSP Side

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

After the binaries have been built, 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, follow the steps below to copy the relevant binaries:

```
$ cd ~/dsplink

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

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

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

Important

Enter the commands shown above in single line

Important

The generated kernel module has different names based on the kernel version being used. I.e. for 2.6.x series kernels the generated kernel module's extension is .ko and for 2.4.x series kernels the generated kernel module's extension is .o. The above commands copy the kernel module to the target filesystem so the appropriate command must be used if the 'wildcard' shown above is not used

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

1.5.1.3. 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.

1.5.1.4. Invoking the application

1.5.1.4.1. Loop sample

To invoke the application enter the following commands:

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

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

Note

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.

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

1.5.1.4.2. Message sample

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

```
$ ./messagegpp message.out <number of 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.

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

1.5.1.4.3. Scale sample

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

```
$ ./scalegpp scale.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.

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

1.5.1.4.4. 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

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.

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

1.5.1.4.5. Readwrite sample

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

```
$ ./readwritegpp readwrite.out <DSP address> <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.

```
$ ./readwritegpp readwrite.out 0xE0510000 1024 10000
```

1.5.1.4.6. Mapregion sample

```
$ cd /opt/dsplink/samples/mapregion
```

```
$ ./mapregiongpp readwrite.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.

```
$ ./mapregiongpp readwrite.out 1024 10000
```

1.5.1.4.7. 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.

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

1.5.1.4.8. 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
```

1.5.1.4.9. 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
```

1.5.1.5. 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
```

1.5.1.6. 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
```

1.6. ADDITIONAL INFORMATION

1.6.1. Tools and utilities

1.6.1.1. Big Physical Area patch instrumentation

The command $cat /proc/bigphysarea shows details regarding free bytes, allocate bytes, etc. This can be helpful when debugging memory situations. Big physical area patch allocates number of pages specified in the grub. conf at the boot time (pages are of 4kBytes size). So, it is advisable to specify sufficient amount of pages to be allocated.

The default configuration of DSPLink uses 4MB of physical buffer allocated through bigphysarea module. So the minimum page count in grub.conf must be 1024 (i.e. 1024 * 4 * 1024=4MB).

Install Guide for DM6437

Abstract

This chapter describes how to install and run samples for DM6437 from the new DSP/BIOS™ Link.

Table of Contents

2.1. Introduction

2.1.1. 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 1.64.00.03 dated OCT 20, 2009.

2.1.2. Terms and Abbreviations

CCS	Code Composer Studio
IPC	Inter Processor Communication
GPP	General Purpose e.g. ARM
DSP	Digital Signal Processor e.g. DM6437
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 2.1. Terms and Abbreviations

2.1.3. References

1	Evaluation Module (EVM) for the DM6437 Quick Start Installation Guide

Table 2.2. References

2.2. INSTALLATION

2.2.1. 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_1_64_00_03.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.

2.2.1.1. Installing Standalone DSP/BIOS™ 5.32.04 and CGTools

For compilation of DSP-side sources and applications, 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
DSP/BIOS™ and CGTOOLS

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.

2.2.1.2. 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

2.2.2. Creating development workspace

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

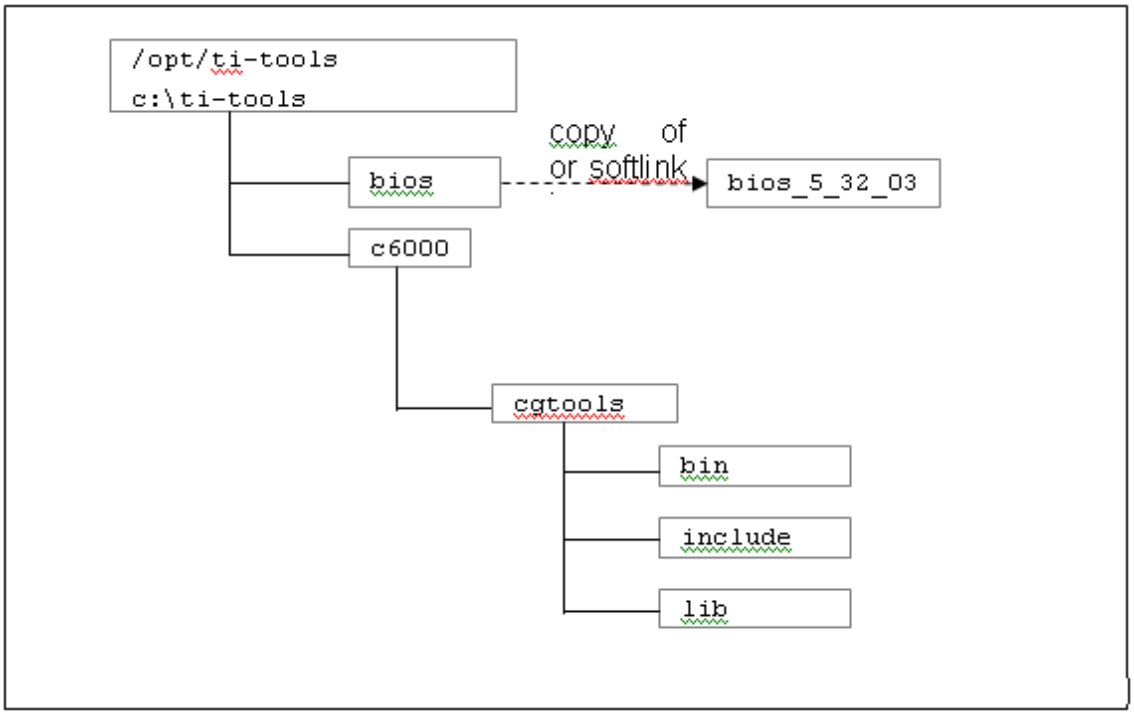

Figure 2.1. Development Workspace

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

2.3. 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.
- The workstation must have at least 512 MB RAM, and 1 GHz CPU.

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.

2.3.1. Creating development workspace

This document and the scripts included in the release assume that DSP/BIOS™ LINK sources have been installed in /home/<user>/dsplink directory.

2.3.2. Creating target system

This release of DSP/BIOS™ LINK for DM6437 has validated on Linux kernel versions 2.6.21. It is recommended to use this version to use DSP/BIOS™. This release of DSP/BIOS™ LINK for DM6437 requires big physical area patch for both kernel version. The following steps can be performed to get the kernel patched and DSP/BIOS™ LINK running:

2.3.1. Linux 2.6.21

- Acquiring the Linux kernel sources:

  The kernel sources can be installed on the build machine through the kernel source package. This package can be obtained from GNU kernel repository site (www.kernel.org). The link given below can also be used to download the kernel sources for Linux:
  Kernel Sources tarball

- Transfer this package to the Linux build machine on to a work directory in home area. The tar utility must be used to explode (untar) this package

  ```
  $tar -xvjf linux-2.6.21.tar.bz2
  ```

- The patch for big physical area can downloaded from:

Big Physical Patch

```
$tar -xvjf linux-2.6.21.tar.bz2
```

Transfer the patch file kernel-2.6.21-bigphysarea.patch to the work directory. Change the current directory to the Linux source work directory

```
$cd /home/<user>/work/linux.2.6.21
```

Now patch the Linux source.

```
$patch -p1 < ../kernel-2.6.21-bigphysarea.patch
```

Important

Enter the commands shown above in single line

- The Linux sources must be configured to include the patch before build. The steps for configuration are:
  Make mrproper (This will erase any .config file) This cleans out the configuration files and any object files an older version might have.

```
$make mrproper
```

The next step is optional, depending on if you want to keep your old configuration or base your new kernel on your old configuration and add the new options found in the new kernel. If you want to base your new kernel on your old one, you can copy the config file from boot directory of the build machine to the Linux source root. $cp /boot/config-2.6.XX-X .config and run make command as follow: 1)

```
$make oldconfig
```

In a command line configuration system, the new option to select the big physical area patch is prompted. Press 'y' to continue. Or The other ways to complete configuration are 2)

```
$make config
```

Or 3)

```
$make menuconfig
```

Or 4)

```
$make xconfig
```

In a GUI configuration system, the option to select the big physical area patch is under processor feature and options menu. In a command line configuration system, the option to select the big physical area patch is prompted.

Important

The above steps will ensure that the patched kernel is configured.

Important

The uniprocessor kernel must be selected even if it is a multicore-CPU or multiprocessor machine.

- Make the dependencies. This ensures all the dependencies like include files are in place

```
$make dep
```

- Once configuration step is successfully completed, compile the Linux sources:

```
$make bzImage
```

This could take some time and should build without any error. Upon successful completion, Linux kernel image (bzImage) will be built under <linux-2.6.21>/arch/i386/boot directory. (Assuming you are working on Intel based PC).

- Build and install the modules required by this kernel image.

```
$make modules && make modules_install
```

This step will build the modules and install it in /lib/modules directory.

- Install the built kernel module.

```
$make install
```

- The grub configuration file will now be updated with the new kernel configuration.

The grub file will need to be updated to allocate 64 MB for big physical area memory allocations. You have to add a new line like below:

```
kernel /vmlinuz-2.6.21-bigphys ro root=/dev/VolGroup01/LogVol00 console=tty0
 console=ttyS0,115200n8 selinux=0 bigphysarea=16384
```

Edit the grub configuration file for updating the entry for the patched kernel.

```
$vi /etc/grub.conf

title fedora7 (2.6.21-bigphys)

root (hd0,4)

kernel /vmlinuz-2.6.21-bigphys ro root=/dev/VolGroup01/LogVol00 console=tty0
 console=ttyS0,115200n8 selinux=0 bigphysarea=16384

initrd /initrd-2.6.21-bigphys.img
```

Important

Enter the commands shown above in single line

Important

To increase the bigphys area size change the value ofbigphysarea in the above command.e.g bigphysarea=65536

On rebooting the kernel, a new option to boot with the patch will be present on grub menu.

2.4. Configuring CCS

Note

CCS 3.3 can be used for working with DM6437. It may need to be configured to use the XDS510 debugger with DM6437

2.5. WORKING ON TARGET PLATFORM

2.5.1. Running the sample applications

Nine sample applications are provided with DSPLINK for the DM6437 platform. All the sample applications are described in detail in the user guide. This section describes the way to execute the sample applications

Note

While building the kernel module (dsplinkk.ko), Build generates the following warnings due to issues in kernel header files

```
warning: pointer targets in passing argument 1 of native_cpuid differ in
 signedness
```

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

2.5.1.1. Configure the DSP/BIOS LINK for DM6437

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=LINUXPC --nodsp=1 --dspcfg_0=DM6437PCI --
dspos_0=DSPBIOS5XX --gppos=RHEL4 --comps=ponslrmc
```

Important

Enter the commands shown above in single line

Important

For details please refer user guide.

2.5.1.2. Copying files to target file system

2.5.1.2.1. GPP Side

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

```
$ cd ~/dsplink
```

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

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

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

Important

Enter the commands shown above in single line

Important

The generated kernel module has different names based on the kernel version being used. I.e. for 2.6.x series kernels the generated kernel module's extension is .ko and for 2.4.x series kernels the generated kernel module's extension is .o. The above commands copy the kernel module to the target filesystem so the appropriate command must be used if the 'wildcard' shown above is not used

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

2.5.1.2.2. DSP Side

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

After the binaries have been built, 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, follow the steps below to copy the relevant binaries:

```
$ cd ~/dsplink

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

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

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

Important

Enter the commands shown above in single line

Important

The generated kernel module has different names based on the kernel version being used. I.e. for 2.6.x series kernels the generated kernel module's extension is .ko and for 2.4.x series kernels the generated kernel module's extension is .o. The above commands copy the kernel module to the target filesystem so the appropriate command must be used if the 'wildcard' shown above is not used

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

2.5.1.3. 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.

2.5.1.4. Invoking the application

2.5.1.4.1. Loop sample

To invoke the application enter the following commands:

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

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

Note

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.

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

2.5.1.4.2. Message sample

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

```
$ ./messagegpp message.out <number of 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.

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

2.5.1.4.3. Scale sample

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

```
$ ./scalegpp scale.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.

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

2.5.1.4.4. 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

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.

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

2.5.1.4.5. Readwrite sample

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

```
$ ./readwritegpp readwrite.out <DSP address> <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.

```
$ ./readwritegpp readwrite.out 0x80140000 1024 10000
```

2.5.1.4.6. Mapregion sample

```
$ cd /opt/dsplink/samples/mapregion
```

```
$ ./mapregiongpp readwrite.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.

```
$ ./mapregiongpp readwrite.out 1024 10000
```

2.5.1.4.7. 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.

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

2.5.1.4.8. MP_LIST sample

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

$ ./mplistgpp mplist.out <iterations> <number of elements> ^lt;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
```

2.5.1.4.9. 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
```

2.5.1.5. 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
```

2.6. ADDITIONAL INFORMATION

2.6.1. Tools and utilities

2.6.1.1. Big Physical Area patch instrumentation

The command $cat /proc/bigphysarea shows details regarding free bytes, allocate bytes, etc. This can be helpful when debugging memory situations. Big physical area patch allocates number of pages specified in the grub. conf at the boot time (pages are of 4kBytes size). So, it is advisable to specify sufficient amount of pages to be allocated.

The default configuration of DSPLink uses 4MB of physical buffer allocated through bigphysarea module. So the minimum page count in grub.conf must be 1024 (i.e. 1024 * 4 * 1024=4MB).