DSP/BIOS™ LINK

DRX45X

LNK 138 USR

1.65.00.03

JUL 12, 2010

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 1.65.00.03 dated JUL 12, 2010.

2 Text Conventions

☞	This bullet indicates important information. Please read such text carefully.
❑	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.	PrOS earlier release.	REL_DRX45X_03.00.00.03.01.zip.

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_sla_1.65.00.03.tar.gz.

This document assumes the install path to be ***L:\dsplink*** *while 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™ 5.33.05 and CGTools

For compilation of DSP-side sources and applications, the CGTools version 6.0.14 can be used. This release has been validated with DSP/BIOS™ version 5.33.05.

This release will not work with BIOS version earlier than 5.33.05.

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:

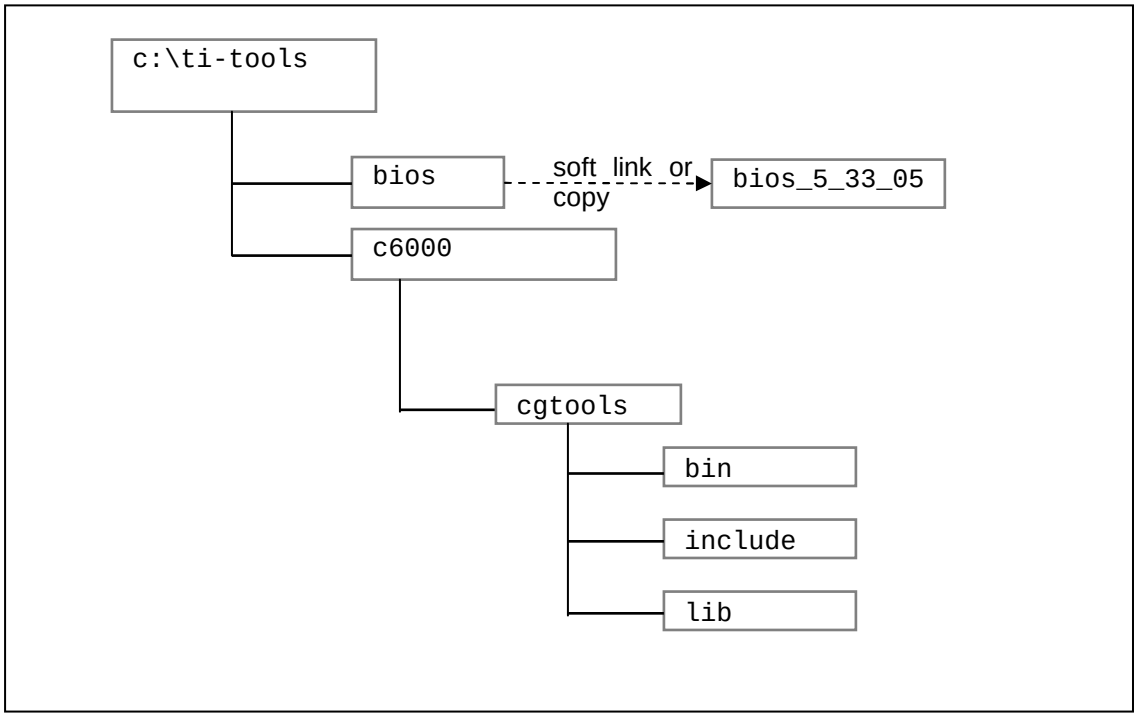

Figure 1. Expected directory structure on development host

- *For 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 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.*

6.1 Tools Directory Structure Layout (User-Defined)

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 can define the following environment variables on your desktop (My Computer -> Right Click -> Properties -> Advanced -> Environment Variables -> System Variables):

ENV variable name	Purpose	Example	Default
TI_TOOLS_BASE_DIR	Defines the base directory for the TI Tools. This should be defined only when your TI Tools are not under C:\ti_tools as the scripts and makefile expect. Refer to Figure 1 for more information.	c:\my_tools\ti_tools	C:\ti-tools
BIOS_INSTALL_DIR	Defines the base directory where the bios is located. This should be defined only when bios is not located under C:\ti-tools\bios or $(TI_TOOLS_BASE_DIR)\bios	c:\my_tools\bios_5_32_04	C:\ti-tools\bios
XDCROOT	Defines the base directory of XDC tools. This should be defined only when the XDC tools are not located in xdctools directory under your bios installation (i.e. $(BIOS_INSTALL_DIR)\xdctools).	C:\xdctools	C:\titools\bios\xdctools
CGTOOLS_INSTALL_DIR	Defines the base directory (not the bin the directory) for your CGTools installation. This should be defined only when the CGTools are not located under C:\titools\c6000\cgtools or $(TI_TOOLS_BASE_DIR)\C6000\cgtools	C:\C6000_v3_10_05	C:\titools\C6000\cgtools
BASE_PERL	Defines the base directory (not the bin directory) for your Perl installation	C:\my_tools\Perl	C:\Perl

Alternatively, you can also 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.

7 Configuring CCS

CCS can be configured to use the XDS510 emulator.

C. WORKING ON TARGET PLATFORM

7.1 PrOS

This release is for DRX45X using PrOS PSP. There is no support for Linux board support package for DRX45X. Hence the latest board specific code updates which maybe OS specific will be present in the PrOS drop.

We have to use CCS for loading and running DSP/BIOS Link through the sample project released in the DRX45X PSP package. Follow the steps given in the PSP release notes on loading and running the sample application.

7.2 Running the sample applications

Sample applications are provided with DSPLINK for the DRX45X 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 PrOS running on the GPP.

7.3 PrOS

For executing the applications on PrOS, the ARM-side executable must be built to include the sample library. The *DSPLINK* library is included into the sample library. In addition, the sample application must be invoked from within the sample_prkv4.c file in the PrOS psp installation. The sample_prkv4.c file must be modified to calls the DSP/BIOS Link sample applications.

7.3.1 Integrating the applications

Based on the DSP executable loader to be used, the procedure for usage of the target GPP file system differs.

The COFF loader is used in the default configuration provided with the release. This release has been verified with the pseudofs utility.

This release has not been verified with the binary or the static loader. The binary and static loader have been retained for reference.

When the static loader is used, no target file system is required for running the message sample when the DSP static loader is used. In this configuration, the DSP executable gets loaded automatically when the ARM executable is loaded. However, there is a constraint of including only a single DSP executable in the build of the GPP executable.

When the binary loader is used, the DSP binary executable 'file' is required while loading it on the target.

When the COFF or binary loader is used, the DSP file needs to be placed into the target file system. To provide for easy application integration and to satisfy the need of a file system, the release package contains the `pseudofs` utility.

The `pseudofs` utility, as the name suggests, creates a pseudo `filesystem` by reading from a file and creating a corresponding C array. This utility is present under directory `dsplink\etc\host\tools\generic\pseudofs`. It expects a text file as its input and the text file is expected to contain:

1. The list of files to be included in the `filesystem`.

2. The name of the file generated as the pseudo file-system.

The generated pseudo file-system file can be built with the required files. A sample input file `pseudofs.cfg` containing the list of files to be added in the file-system is included with the release package under directory `dsplink\etc\host\tools\generic\pseudofs\PrOS\DRX45X`. This file includes all the DSP executables, data files and scripts for running the samples or test-suite. It can be modified to comment out the files that are not required to be included into the file system. It can also be modified to include any additional files, or to modify the input paths for the existing files.

After the list of files has been modified, the file system can be built using the following command:

```
L:\> cd dsplink\etc\host\tools\generic\pseudofs
L:\> gmake –s filesys [debug | release]
```

This command generates the file-system and compiles it. The `FILESYS.LIB` file generated must be included within the build of the PrOS executable.

7.3.2 Building the PrOS executable

The procedure to build the final PrOS executable to be loaded onto the DRA44x EVM is:

1. Ensure that you have the installations for
   DRX45x :
   - TMS470 C/C++ CODE GENERATION TOOLS Release 4.1.4.
   - PrOS earlier release version.
   - CGTOOLS v6.0.14 with CCS3.3pre-drx40x-psp-rel0.1.0: TI PSP release.
   - BIOS 5.33.05
2. Configure the DSP/BIOS LINK for DRX45X
   The build configuration command must be executed to configure DSPLink for the various parameters such as platform, GPP OS, build configuration etc.

   - DRX45X for Seudofs

   perl dsplinkcfg.pl --platform=DRX45X --nodsp=1 --dspcfg_0=DRX45XGEMSHMEM --dspos_0=DSPBIOS5XX --gppos=PROS --comps=ponslrm --fs=PSEUDOFS
3. Follow the steps given in TI PSP release user guide to create a CCS project that consumes the PSP sample project. We will be using this Sample Project as a base to run Link samples. DSPLink uses the UART driver for printf implementation and MMC/SD driver for PrFile support.
   Link specific project settings while creating the CCS project are as follows:

   - Configurations about TMS320DRx45x for DRX45X

4. In the lnk.cmd
   ($HOME\REL_DRX45X_03.00.00.03.01\pspdrivers\system\drx45x\pros\evmdrx45x\build) check the address and replace the SDRAM with DDR. These address

should belongs to DDR.

```
 EXCPT_VECT      :   org =   0x80000004    len =   0x0000003C
INT_VECT        :   org =   0x80000040    len =   0x00000104
DDR_INIT_SECT   :   org =   0x80100000    len =   0x04000000
DDR_UNINIT_SECT :   org =   0x86100000    len =   0x04000000
DDR_NON_CACHE   :   org =   0x8A100000    len =   0x02000000
DDR_NON_CACHE_APPS  :   org =   0x8CB00000    len =  0x00400000
SECTIONS
{


   .cdesc      :   load        =   DDR_INIT_SECT /* MMU PT */
   .vects      :   load        =   EXCPT_VECT    /* Exception
vectors to be located in internal RAM starting at 0x4 */
   .intvects   :   load        =   INT_VECT      /* Interrupt
vector numbers to be located in internal RAM starting at 0x40 */
   .start      :   load        =   DDR_INIT_SECT /* Start of PSP
image (kstart.asm in PrKernel) */
   .text       :   load        =   DDR_INIT_SECT
   .const      :   load        =   DDR_INIT_SECT
   .cinit      :   load        =   DDR_INIT_SECT
   .bss        :   load        =   DDR_UNINIT_SECT
   .data       :   load        =   DDR_UNINIT_SECT
   .sysmem     :   load        =   DDR_UNINIT_SECT
   .nocache    :   load        =   DDR_NON_CACHE
   .stack      :   load        =   DDR_UNINIT_SECT
   .text:pagetable : load      =   DDR_INIT_SECT
}
```

6. Modify the psp_pros_main_sample.c file within the psp package to call MESSAGE_Main () with the appropriate arguments. The code to call the different samples with different sets of parameters is also shown below. The addresses to be passed for the readwrite example are platform dependant.

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

7.
```
MESSAGE_Main (“/opt/message.out”, “10000”, “0”) ;

RING_IO_Main ("/opt/ringio.out", "1024", "10240", “0”) ;

MP_LIST_Main ("/opt/mplist.out", "100", "10", “0”) ;

MPCSXFER_Main ("/opt/mpcsxfer.out", "128", "1000", “0”) ;

RDWR_Main ("/opt/readwrite.out", "2281308160", 2281308160, "1024",
1024, "1000", 1000,0) ;

MPCSXFER_Main ("/opt/mpcsxfer.out", "128", "1000", “0”) ;

MESSAGEMULTI_Main (“/opt/messagemulti.out”, “10000”, “1”, “0”) ;

MESSAGEMULTI_Main (“/opt/messagemulti.out”, “10000”, “2”, “0”) ;
```

```
MESSAGEMULTI_Main ("/opt/messagemulti.out", "10000", "3", "0") ;
MESSAGEMULTI_Main ("/opt/messagemulti.out", "10000", "4", "0") ;
MESSAGEMULTI_Main ("/opt/messagemulti.out", "10000", "5", "0") ;
MESSAGEMULTI_Main ("/opt/messagemulti.out", "10000", "6", "0") ;
MESSAGEMULTI_Main ("/opt/messagemulti.out", "10000", "7", "0") ;
MESSAGEMULTI_Main ("/opt/messagemulti.out", "10000", "8", "0") ;
MESSAGEMULTI_Main ("/opt/messagemulti.out", "10000", "9", "0") ;
MESSAGEMULTI_Main ("/opt/messagemulti.out", "10000", "10", "0") ;
MESSAGEMULTI_Main ("/opt/messagemulti.out", "10000", "11", "0") ;
MESSAGEMULTI_Main ("/opt/messagemulti.out", "10000", "12", "0") ;
MESSAGEMULTI_Main ("/opt/messagemulti.out", "10000", "13", "0") ;
MESSAGEMULTI_Main ("/opt/messagemulti.out", "10000", "14", "0") ;
MESSAGEMULTI_Main ("/opt/messagemulti.out", "10000", "15", "0") ;
MESSAGEMULTI_Main ("/opt/messagemulti.out", "10000", "16", "0") ;
```

8. Modify the DRX45x PSP project ($HOME/ REL_DRX45X_03.00.00.03.01\pspdrivers\system\drx45x\pros\evmdrx45x\build\evm_drx45x_psp_pros_sample.pjt) file available with the psp package to link the dsplink libraries. Right click on the project in the Build tab. Select Add Files to project and add the dsplink libraries i.e. DSPLINK.LIB, MESSAGEGPP.LIB etc to the project.

   Debug libraries can be added to the Debug Build whereas Release libraries can be added to the Release build.

   The libraries need to add to the project.

9. Follow steps given in PSP user guide to build the sample project file, register the resident unit, setup the target board and load and run the samples.

10. The output from the samples can be seen on the UART terminal.
    Connect Target COM (UART) to your Local Machine through RS232 Cable Configure the Hyper-Terminal or TeraTerm (or any Local Windows UART Application) at –
    - 115200 BAUD Rate,
    - 8 bit Data, No Parity, No Flow Control & Stop Bit equal to 1

❑ *The above instructions assume that the psp package for DRX45x is used for building and running the applications on PrOS. For a different system, please modify these instructions accordingly.*

❑ *DSPLINK uses DRX45X PSP's printf implementation. DSPLINK libraries call printf only when PSP "pspUartConsoleOut" function is not defined. So system integrators need to update the PRINT_Printf () API in gpp\src\osal\PrOS\print.c, if they don't want to use DRx40x PSP's prnitf.*

7.3.3 Launching the application using CCS

1) Launch CCS.

2) Connect to, and open the ARM-side CCS.

3) The PrKernel needs to be compiled in debug mode

4) After the loading the object file, press F10.

5) If you see Disconnect message thrown by CCS, Disconnect the CCS and reconnect again.

6) Press CTRL+SHIFT+L to reload the program binary.

7) Repeat step 4 and Press F5.

- ❑ *(Follow the steps given in the PSP release notes to run the PSP sample project.)*

8) Verify the correct execution of the samples through prints observed on the CCS output window.

D. ADDITIONAL INFORMATION

8 Tools and utilities

8.1 binloader script

When PrOS is used as the ARM-side Operating System, *DSPLINK* supports different types of DSP executable loaders for loading the DSP executable onto the target.

The `binloader` script is provided to generate the source and header files with information about the DSP executable to be loaded by the binary loader. It also converts the DSP executable from COFF into binary format, which can be loaded by the binary loader.

The files generated are:

1. 'C' file containing an instance of the `BinLoaderImageInfo` structure required by the binary loader. This generated file can be built with the ARM-side example to generate information used by the binary loader.
2. 'H' file containing declaration of the instance of the `BinLoaderImageInfo` structure in the 'C' file. This file can be included by the ARM-side application using the binary loader.
3. 'BIN' file generated through COFF to binary conversion of the DSP executable.
4. 'DAT' file, which is an intermediate file generated by the script and used for conversion of the DSP executable from COFF to binary format. This file is available to the user for reference about the memory map used for generation of the binary file. to load and unload the DSP/BIOS™ LINK device driver on the target platform.

This script is present at the following location within the *DSPLINK* installation:

```
L:\dsplink\etc\host\scripts\msdos\loaderutils\binary
```

Additional information about the script and its usage is provided within the `README.TXT` file present at the same location.

8.2 staticloader script

The DRX45x platform has a unified GPP-DSP memory map. When PrOS is used as the ARM-side Operating System, the DSP executable can be built into the ARM-side executable. This optimizes the loading time of the DSP executable on the target, since the DSP executable gets automatically loaded when the ARM executable is loaded onto the target.

The `staticloader` script is provided to generate source and command files corresponding to the DSP executable image specified. These generated files can be statically linked into the ARM-side executable to get a pre-loaded DSP image.

The files generated are:

1. 'C' file containing arrays of DSP sections. This file is generated in the asp-psp source directory for DRA45x, and included in the build.
2. 'H' file containing declarations of the DSP section arrays. This file is generated in `$(DSPLINK)\gpp\inc\PrOS\Jacinto` directory.

3. 'CMD' file containing linker placement of the DSP section arrays. This file is generated in the asp-psp build directory for DRA45x and included in the linker command file for the GPP executable.

This script is present at the following location within the *DSPLINK* installation:

```
L:\dsplink\etc\host\scripts\msdos\loaderutils\static
```

Additional information about the script and its usage is provided within the `README.TXT` file present at the same location.