DSP/BIOS™ LINK

LNK 111 REL

Version 1.65

OCT 22, 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

1 Introduction

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[1].

As the name suggests, DSP/BIOS operating system is expected to be running on the DSP with all of these platforms.

The platforms in this release are:

- OMAPL138
- DA850
- OMAP3530 EVM
- OMAP3730 EVM
- OMAP2530 EVM
- DM6446
- DM6467
- DM6467T
- DA8XX
- OMAPL1XX
- Linux PC + DM6437 over PCI
- Linux PC + DM648 over PCI
- DM357
- TNETV107X

Multi-DSP configurations:

- Linux PC + two DM6437 over PCI

Additionally, support for the following platforms is available in the release:

- DRA44x
- DRX416
- DRX45x
- Multi-DSP configuration: DRA44x + DM6437 over VLYNQ

1.1 Text Conventions

O	This bullet indicates important information. Please read such text carefully.

[1] Applications differentiate the products. The application developers would prefer to focus on the application rather than the IPC mechanism.

q	This bullet indicates additional information.

2 Out-of-Box Contents

This release of DSP/BIOS™ LINK contains the following:

1. Installer package for DSP/BIOS™ LINK containing:
   - TAR.GZ file containing the DSP/BIOS™ LINK installation package.
   - Scripts, tools etc., to use on the debug and development host machines.
2. Release Notes (this document) providing an overview of this release.
3. Installation Guide, Platform Guide, User Guide, Programmer's Guide and other technical documents.

   Complete list of the documents is available in the User Guide.

3 Minimum System Requirements

3.1 Hardware

Depending on the target platform configuration, DSP/BIOS™ Link sources can be compiled on a PC running either Microsoft Windows or Red Hat Linux.

3.1.1 Development / Debug Host Machine

Please refer to installation guide for the relevant platform.

3.2 Software

3.2.1 Generic Software Requirement

- PERL Installation
- TeraTerm (or any other terminal emulation program)
- One of GNU make version 3.81, 3.81beta1, 3.90 or 3.92.

O On hosts where the above mentioned make versions are not available, DSPLINK users can use gmake (from TI XDC tools) to build DSPLink.

O To build DSPLink using gmake (from TI XDC tools) run gmake as below:

```
<Absolute path of gmake from TI xdc tools>$ gmake –s [debug|release}.
```

For example,

1) `/opt/xdc/<xdctools install path>/gmake –s debug`

2) `/opt/ti-tools/<bios install path>/xdctools/gmake –s debug`

O GNU make 3.81 version can be downloaded and installed from the URL http://ftp.gnu.org/pub/gnu/make/make-3.81.tar.gz

O The software listed above is not part of the DSP/BIOS™ LINK release. For all the TI products, please contact your TI representative.

O ActivePerl - a distribution of PERL from Active State - can be obtained from the URL http://www.activestate.com/Products/ActivePerl/

3.2.2 Platform Specific Software Requirement

O For all platforms, this release will not work with versions of DSP/BIOS earlier than that specified in the dependencies.

3.2.2.1 For Working with DM6446

- DSP/BIOS™ version can be 5.32.04 or 6.20.00.37 (one of the below)
- For Standalone DSP/BIOS™ 5.32.04.
  - § DSP CG Tools version v6.0.18 or greater
  - § CCS 3.3 IDE for debugging purposes.
- For Standalone DSP/BIOS™ bios 6.20.00.37
  - § XDC release version xdctools 3.15.00.50

- § IPC 1.00.00.40
- § CCS 3.3.38.2 for debugging purposes.
- § DSP CG tools version v6.1.5.

- DM6446 PSP release 03.01 based on 2.6.32 rc2 git kernel version and tool chain version arm-2009q1-203

3.2.2.2 For Working with TNETV107X

- For Standalone DSP/BIOS™ 5.41.06.
- DSP CG Tools version v6.0.18 or greater
- XDC release version xdctools 3.15.04.70
- CCS 3.3 IDE for debugging purposes.
- Montavista Linux release 5.0 based on linux 2.6.18 kernel with glibc toolchain.

3.2.2.3 For Working with DM6467

- Standalone DSP/BIOS™ 5.32.04
- DSP CG Tools version v6.0.18 or greater
- CCS 3.3 IDE
- DM6446 PSP release 03.01 based on 2.6.32 rc2 Git kernel version and tool chain version arm-2009q1-203.

3.2.2.4 For Working with DM6467T

- Standalone DSP/BIOS™ 5.41.00.06
- XDC Tools 3.15.04
- DSP CG Tools version v6.0.18 or greater
- CCS 3.3 IDE
- DM6446 PSP release 03.01 based on 2.6.32 rc2 Git kernel version and tool chain version arm-2009q1-203.

3.2.2.5 For working with OMAP3530

- Standalone DSP/BIOS™ 5.32.04.
- DSP CG Tools version v6.0.18 or greater
- CCS 3.3 IDE
- PSP release PSP release AM35x-OMAP35x-PSP-SDK-03.00.00.05 with tool chain version arm-2009q1-203.

3.2.2.6 For working with OMAP2530

- Standalone DSP/BIOS™ 5.32.04
- DSP CG Tools version v6.0.18 or greater
- CCS 3.3 IDE
- Mistral release 1.1 and Mistral Additional Software Release 3.1 (OMAP2530EVM_ADDLSW_Rel_3_1.tar.bz2).

- Code sourcery tool-chain arm-2007q1

3.2.2.7 For Working with DM6437

- Standalone DSP/BIOS™ 5.32.04.
- DSP CG Tools version v6.0.18 or greater
- CCS 3.3 IDE
- Linux kernel version 2.6.21 with big physical area patch.

3.2.2.8 For Working with DM648

- Standalone DSP/BIOS™ 5.32.04.
- DSP CG Tools version v6.0.18 or greater
- CCS 3.3 IDE
- Linux kernel version 2.6.21 with big physical area patch.

O For additional details on installation and software/hardware requirements for a specific platform, please refer to the install guide for that platform provided with the DSPLink release.

3.2.2.9 For Working with DM357

- Standalone DSP/BIOS™ 5.33.03.
- DSP CG Tools version v6.1.2 or greater
- CCS 3.3 IDE
- U-boot release 1.2.0 showing date Dec 20 2007
- Montavista Linux release
    - 5.0 with DM357 EVM based on linux 2.6.18 kernel. Either glibc or uclibc toolchain can be installed.
        - With Davinci Engineering Release LSP-200 02.00.00.140

3.2.2.10 For Working with DA8XX

- DSP/BIOS™ version 6.20.00.37
- For Standalone DSP/BIOS™ bios 6.20.00.37.
    - XDC release version 3.15.00.50
    - IPC 1.00.00.40
    - CCS 3.3.38.2 for debugging purposes.
    - DSP CG tools version v6.1.5
- LSP kernel release DaVinci PSP SDK 03.20.00.11 based on 2.6.33 rc4 with GIT tool chain arm-2009q1-203.
- For DSPLINK_NOBOOT_MODE (where ARM does not load and start DSP)
    - This has not been validated with the latest LSP

For additional details on installation and software/hardware requirements for a specific platform, please refer to the install guide for that platform provided with the DSPLink release.

3.2.2.11 For Working with OMAPL1XX

- DSP/BIOS™ version 5.33.03
  - For Standalone DSP/BIOS™ 5.33.03.
    - § DSP CG Tools version v6.1.2 or greater
    - § CCS 3.3 IDE for debugging purposes.
- LSP kernel release DaVinci PSP SDK 03.20.00.11 based on 2.6.33 rc4 with GIT tool chain arm-2009q1-203.
- For DSPLINK_BOOT_MODE where ARM loads and start DSP
    - § This has not been validated with the latest LSP

For additional details on installation and software/hardware requirements for a specific platform, please refer to the install guide for that platform provided with the DSPLink release.

3.2.2.12 For Working with OMAPL138

- DSP/BIOS™ version 5.33.05
  - For Standalone DSP/BIOS™ 5.33.05.
    - § DSP CG Tools version v6.1.5 or greater
    - § CCS 3.3 IDE for debugging purposes.
- For DSP boot mode where ARM loads and start DSP
  - PSP Linux release package - LSP kernel release DaVinci PSP SDK SDK 03.20.00.11 based on 2.6.33 rc4 with GIT tool chain arm-2009q1-203.

3.2.2.13 For Working with DA850

- DSP/BIOS™ version can be 6.20.00.37
  - For Standalone DSP/BIOS™ bios 6.20.00.37
    - § XDC release version xdctools_3_15_00_50
    - § IPC 1.00.00.40
    - § CCS 3.3.38.2 for debugging purposes.
    - § DSP CG tools version v6.1.5.
- PSP Linux release package - LSP kernel release DaVinci PSP SDK SDK 03.20.00.11 based on 2.6.33 rc4 with GIT tool chain arm-2009q1-203.

For additional details on installation and software/hardware requirements for a specific platform, please refer to the install guide for that platform provided with the DSPLink release.

4 Functionality Supported

4.1 Features

4.1.1 Generic Features

4.1.1.1 Support for multiple platforms

DSPLink supports multiple platforms in specific configurations as mentioned in this document. The standard features of DSP/BIOS™ LINK are supported with this release. For additional information, please refer to the User Guide and Programmer's Guide documents.

4.1.1.2 Multi-process and multi-application support

Multiple applications or processes can directly call DSP/BIOS LINK PROC APIs to setup, control the DSP and destroy the DSPLink driver. They do not need to coordinate between themselves at run-time to access the DSPLink driver. The only thing they need to ensure is:

1. The same DSP executable is used by all applications simultaneously using DSPLink
2. The same dynamic configuration is used by all applications simultaneously using DSPLink

4.1.1.3 Support for multiple DSPs connected to GPP in star-topology

DSPLink supports multiple DSPs connected to the master GPP processor in a star-topology. With this feature, it is possible to connect multiple DSPs to the GPP in the system, over heterogeneous physical links. Example reference ports have also be included for multi-DSP configurations:

- Linux PC connected to two DM6437 devices over PCI
- DRA44x connected to external DM6437 over VLYNQ

The DSPLink static build configuration supports choosing the desired device and build configuration. The static build configuration supports command-line based usage to enable scripting.

4.1.2 PROC module

4.1.2.1 DSP boot-load

This release supports boot-loading the DSP from the ARM. This includes:

- Initializing the DSP & making it available for access from the GPP.

On OMAP platforms: Configuring the DSP MMU to allow straight-through access for specified memory regions.

- Loading code on the DSP. Multiple types of loaders can be used for this. DSPLink provides a few standard loaders as part of the release package. Additional loaders can be added and plugged into DSPLink.
- Starting execution from the run address specified in the executable.
- Reading from or writing to DSP memory.

- Stopping DSP execution.
- DSP/BIOS LINK for the OMAP platforms does not perform any power management of the DSP. This is expected to be done by u-boot or other power-management utilities.

If PROC-only scalability configuration build is used, DSPLink does not perform driver handshake and interrupts initialization. Due to this, DSPLink can be used in PROC-only configuration to load and start a non-DSPLink DSP executable, i.e. a DSP executable that has no DSPLink content or shared memory usage within it.

4.1.2.2 Support for multiple boot modes

DSPLink supports three different types of DSP boot modes:

- DSPLINK_BOOT_MODE: Default
  - GPP boots first
  - Uses DSPLink to load the DSP
  - Uses DSPLink to start the DSP running
- DSPLINK_NOLOAD_MODE: External DSP load
  - GPP boots first
  - Application/GPP boot-loader pre-loads the DSP
  - Uses DSPLink to optionally power up the DSP
  - Uses DSPLink to start the DSP running
- DSPLINK_NOBOOT_MODE: External DSP start. This supports two scenarios
  - GPP-based load
  - GPP boots first
  - Application/GPP boot-loader pre-loads the DSP
  - Application/GPP boot-loader starts the DSP running
  - Uses DSPLink only for IPC with the DSP

    OR
  - DSP-based load
  - DSP boots first, starts running an application
  - GPP comes up later and sets up DSPLink, which initializes shared memory
  - DSPLink is not used to load or start the DSP
  - Uses DSPLink only for IPC with the DSP

For complete details on the boot modes along with information on configuration and application usage, please refer to the Programmer's Guide document.

4.1.2.3 Dynamic configuration

DSP/BIOS™ LINK can be dynamically configured with parameters specific to the system configuration. This can be decided by the system integrator. Usage of a new dynamic configuration does not require re-build of DSPLink libraries.

4.1.3 Inter-processor communication modules

4.1.3.1 POOL

The POOL component supports configuring and using shared memory buffers across processors. This includes

- Configuring the shared memory region through open & close calls.
- Allocating and freeing buffers from the shared memory region.
- Translating address of a buffer allocated to different address spaces (e.g. GPP to DSP)
- Synchronizing contents of memory as seen by the different CPU cores (Not applicable for the OMAP3530 platform)

4.1.3.2 NOTIFY

The NOTIFY component allows applications to register for notification of events occurring on the remote processor and send event notification to the remote processor.

It uses the physical interrupt present between the processors and provides the facility of multiple prioritized events over the same physical interrupt.

4.1.3.3 MPCS

The MPCS component allows applications to achieve mutually exclusive access to shared data structures through a multi-processor critical section (MPCS) between GPP and DSP.

4.1.3.4 MPLIST

The MPLIST component provides a doubly-linked circular linked list based transport mechanism between GPP and DSP.

It provides APIs for pushing and popping buffers from the list, traversing the list, inserting and removing elements and other services. The list is shared between the two processors.

4.1.3.5 CHNL

The CHNL component provides issue-reclaim based data streaming. It is based on the SIO module in DSP/BIOS™.

4.1.3.6 MSGQ

The MSGQ component provides queue based messaging. It is an acronym for ‘message queue’.

This component is responsible for exchanging short messages of variable length between the GPP and DSP clients. It is based on the MSGQ module in DSP/BIOS™.

The messages are sent and received through message queues.

This component is based on the MSGQ module in DSP/BIOS™.

4.1.3.7 RING IO

The RingIO component provides circular ring buffer based data streaming. Each ring buffer can be operated on by a single reader and single writer.

4.2 Feature enhancements from previous release 1.64

This release is binary and API compatible to previous GA release 1.64

4.2.1 Addition of WinCE support on OMAPL1XX, DA8XX, OMAPL138, DA850

Support for WinCE has been added on OMAPL1XX, DA8XX, OMAPL138, DA850. This has been tested in the default boot mode only.

4.2.2 Move to Linux GIT kernel for DA8XX/OMAPL1XX

DSPLink has moved from Montavista Linux releases to GIT releases for Linux for DA8XX/OMAPL1xx. The tool chain used is arm 2009q1-203.

4.2.3 Move to Linux GIT kernel for DM6446

DSPLink has moved from Montavista Linux releases to GIT releases for Linux for DM6446. The tool chain used is arm 2009q1-203.

4.2.4 Added support for OMAP3730

This release adds support for OMAP3730 as same device as OMAP3530.

4.3 Generic Defect fixes from previous release 1.64

Identifier	Headline
SDOCM00064681	Runtime configuration of DSP clk is incorrect for the following platforms:DM6467, DM6467T and OMAP3530

Description :

DSPLINK should be passing clk_get_rate()/1000 up to the DSP for platforms:DM6467, DM6467T and OMAP3530.

Identifier	Headline
SDOCM00064915	Update DSPLink CFG files to set POOLSIZE to POOLMEMORYSIZE for all the devices

Description:

Update DSPLink CFG files to set POOLSIZE to POOLMEMORYSIZE for all the devices

Update DSPLink CFG files to set POOLSIZE to POOLMEMORYSIZE for all the devices as the value 0x70000 confuses the application writers

Expected Result:
DSPLink CFG files should use POOLSIZE as POOLMEMORYSIZE and not 0x70000 for all the devices
Release Note:
DSPLink CFG files have been updated to use POOLSIZE as POOLMEMORYSIZE and not 0x70000 for all the devices.

Identifier	Headline
SDOCM00066145	Macro Definition of OPT (Optional Parameter in DSPLink) conflicting with OPT filed of EDMA PaRAM defined in CSL

Description:

A macro named OPT is defined to nothing in dsplink.h. There is a field called OPT in the EDMA PARAM structure of CSL definition. Since the name of the macro and the field are same, there are compilation errors when any application which is using DSPLink and CSL is trying to populate the OPT field. Since this field is set to nothing by the pre-processor as per the definition in dsplink.h, and at the compilation stage it is actually "nothing" being assigned to a value. Thus the compilation error.
Expected Result:
System integrator should be able to build an application with both DSPLink and CSL headers in it.
Release Note:
Macro definition of OPT has been removed from DSPLink source code for both gpp and dsp sources.

Identifier	Headline

SDOCM00065499	In-place operations on the Reader RingIO buffer require cache invalidation before releasing the buffer(RingIO_release)

Description:

ARM is writer and DSP is reader, in DSP we are performing some operation on the RingIO Buffer i.e. DSP manipulates the data and writes to the same buffer(RingIO acquired). As soon as this buffer is released from DSP, ARM writes data to this buffer, this data is not reflecting in the DSP when it is acquired, it is partially modified. Looks some cache issue while releasing the buffer.
Cache write back in DSP is happening for part of the memory after GPP RingIO release that's why data gets corrupted.
BCACHE_inv() API is called before every RingIO release in DSP.
To use Reader RingIO buffer, is it required cache lines to write back before release

Expected Result:
RingIO should support in place buffer operations.
Release Note:
Cache invalidation has been added to the following RingIO API's to support in buffer operations

- RingIO_readerRelease
- RingIO_readerCancel
- RingIO_writerCancel

:Identifier	Headline
SDOCM00070247	DSPLink creates an un-named task for MSGQ

Description:

There is a request from one of our customers to add a "DSP/BIOS task name" for DSP Link.
Expected Result:
The customer should not see a high priority un-named task.
Release Note:
The task created for DSPLink in TSK mode has been updated with the name 'DSPLINK_ ZCPYMQT'

:

4.4 Platform-specific defect fixes

4.4.1 DM6446 specific

None.

4.4.2 DA8xx/OMAPL1XX specific

Identifier	Headline
SDOCM00072025	Default compilation flag for OMAPL138 DSP build does not support floating point

Description:

The build system of DSPLink for C674x based devices did not correctly support compilation for floating point operations.

Expected Result:
DSPLink should support compilation of code for floating point devices for DA8XX, DA850, OMAPL1XX and OMAPL138
Release Note:
The build system of DSPLink for C674x based devices did not correctly support compilation for floating point operations.

This has been corrected. In order to support this, the compile flag for C674x based devices is now updated to mv6740 from mv6400+.

The system integrator/application writer now has to edit below files to update the paths for BIOS, XDC, IPC and CG Tools installations in:

c674x_6.xx_linux.mk - To Build for BIOS 6 based devices (DA8XX, DA850) on a Linux host machine.

c674x_6.xx_windows.mk- To Build for BIOS 6 based devices (DA8XX, DA850) on a Windows host machine.

c674x_5.xx_linux.mk - To Build for BIOS 5 based devices (OMAPL1XX, OMAPL138) on a Linux host machine.

c674x_5.xx_windows.mk – To Build for BIOS 5 based devices (OMAPL1XX, OMAPL138) on a Windows host machine.

4.4.3 OMAP2530 specific

None.

4.4.4 OMAP3530 specific

Identifier	Headline
SDOCM00068950	InstallGuide_Linux_OMAP3530.pdf shows incorrect LPM usage sequence

Description:

In InstallGuide_Linux_OMAP3530.pdf section 10.4, the following is mentioned for running all sample applications:
LPM on

```
# LPM off
# run message sample

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

This is incorrect. It should be lpmOFF followed by lpmON.
Expected Result:
The document should have the correct lpm sequence.
Release Note:
The Installation Guide for WinCE and Linux for OMAP3530 has been updated with the correct lpm sequence.

4.4.5 DM6437 specific

None.

4.4.6 DM648 specific

None.

4.4.7 DM357

None.

4.4.8 DM6467T

None.

4.5 Operating System specific defect fixes

4.5.1 Linux-specific

Identifier	Headline
SDOCM00065115	Device driver file_operation structure does not have owner field set

Description :

I have stumbled across an easy bug on cmem and dsplink kernel modules. Neither of these modules does proper usage counting when a process opens their device file and this stems for the fact that their file_operations structure does not have the 'owner' field set to 'THIS_MODULE', simply adding ".owner = THIS_MODULE," in the code that initializes these structures fixes the problem. Not doing proper usage counting on the modules makes it possible to unload the modules even if they are in use, immediately leading to kernel crashes.

Identifier	Headline
SDOCM00070468	Gdb session hangs when trying to debug DSPLink based applications

Description:

Gdb session hangs when trying to debug DSPLink based applications on WindRiver Linux & tool chain in their custom setup.
Expected Result:
The protection should be complete and should not result into any data corruption.
Release Note:
The semaphore implementation in Linux OSAL was not handling signals correctly. This has been corrected.

Identifier	Headline
SDOCM00070927	MSGQ_get takes a long time ~1sec to complete

Description:

MSGQ_get takes a long time ~1sec to complete.

Expected Result:
MSGQ_get call should not take a long time when message from DSP is available.
Release Note:
A race condition between DPC_Schedule and DPC_CallBack caused the DPC to not get scheduled for 1 sec even though a message is available.

4.6 Upgrade and compatibility information

DSPLink 1.65 is API and binary compatible with DSPLink 1.64. Other changes that are relevant to applications are listed below.

4.6.1 Addition of GPP build specific defines

To match task mode and swi mode on DSP, corresponding build flags have been added to GPP specific defines as well.

- GPP_TSK_MODE – Task mode
- GPP_SWI_MODE – Swi mode

However these flags are not used in the code.

4.6.2 Move to Linux GIT kernel for DA8XX/OMAPL1XX

DSPLink has moved from Montavista Linux releases to GIT releases for Linux for DA8XX/OMAPL1xx. The tool chain used is arm 2009q1-203.

The distribution files have been renamed to be consistent with the naming convention used for working with GIT based Linux kernels. These files are present in $DSPLink/make folder.

These files are present in $DSPLink/make folder.

- omapl1xx_mvlpro5.0.mk renamed to -> omapl1xx_2.6.mk
- omapl1xx_uclibc5.0.mk renamed to -> omapl1xx_uclibc.mk
- da8xx_mvlpro5.0.mk renamed to -> da8xx_2.6.mk
- da8xx_uclibc5.0.mk renamed to -> da8xx_uclibc.mk

The options to configure DSPLink for DA8XX and OMAPL1xx using dsplinkcfg script have also changed.

For OMAPL1xx -

```
perl dsplinkcfg.pl --platform=OMAPL1XX --nodsp=1 --dspcfg_0=OMAPL1XXGEMSHMEM --
dspos_0=DSPBIOS5XX --gppos=ARM --comps=ponslrmc
or
perl dsplinkcfg.pl --platform=OMAPL1XX --nodsp=1 --dspcfg_0=OMAPL1XXGEMSHMEM --
dspos_0=DSPBIOS5XX --gppos=OMAPL1xxuc --comps=ponslrmc
```

For DA8xx -

```
perl dsplinkcfg.pl --platform=DA8XX --nodsp=1 --dspcfg_0=DA8XXGEMSHMEM --
dspos_0=DSPBIOS6XX --gppos=ARM --comps=ponslrmc
or
perl dsplinkcfg.pl --platform=DA8XX --nodsp=1 --dspcfg_0=DA8XXGEMSHMEM --
dspos_0=DSPBIOS6XX --gppos=DA8xxuc --comps=ponslrmc
```

4.6.3 Move to Linux GIT kernel for DM6446

DSPLink has moved from Montavista Linux releases to GIT releases for Linux for DM6446. The tool chain used is arm 2009q1-203.

The distribution files have been renamed to be consistent with the naming convention used for working with GIT based Linux kernels. These files are present in $DSPLink/make folder.

These files are present in $DSPLink/make folder.

- davinci_mvlpro5.0.mk renamed to -> davinci_2.6.mk
- davinci_uclibc5.0.mk renamed to -> davinci_uclibc.mk

The options to configure DSPLink for DM6446 using dsplinkcfg script have also changed.

For DM6446 -

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

or

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

4.6.4 Added support for OMAP3730

This release adds support for OMAP3730 as same device as OMAP3530. The same ARM/DSP binary executable will be run on both OMAP3730 and OMAP3530. Even though the memory map, clock rate may be different, no new platform configuration is exposed to the user. The user does not see a platform called OMAP3730 but updates the OMAP3530configuration for OMAP3730. A single binary is generated for both devices in the same device directory. User will not have side-by-side libraries/binaries generated for the two devices for both ARM & DSP. This implies that DSPLink will not be able to provide different default memory maps for the two devices.

4.6.5 LSP version upgrade for OMAP3530

The LSP version has been upgraded to PSP release PSP SDK AM35x-OMAP35x-PSP-SDK-03.00.00.05.

4.6.6 LSP version upgrade for DA850/OMAPL138

The LSP version has been upgraded to LSP kernel release DaVinci PSP SDK 03.20.00.11 with GIT tool chain arm-2009q1-203.

4.6.7 LSP version upgrade for DA8xx/OMAPL13xx

The LSP version has been upgraded to LSP kernel release DaVinci PSP SDK 03.20.00.11 with GIT tool chain arm-2009q1-203.

4.6.8 UClibc support for DA850 corrected

The dsplinkcfg script and distribution files for DA850 did not correctly support uclbc tool chain.

This has been corrected. The option for running dsplinkcfg script for DA850 for uclibc is

```
perl dsplinkcfg.pl --platform=DA850 --nodsp=1 --dspcfg_0=DA850GEMSHMEM --dspos_0=DSPBIOS6XX --gppos=DA850LSPuc --comps=ponslrmc
```

The distribution file has been renamed from

- da850_uclibc5.0.mk renamed to -> da850_uclibc.mk

4.6.9 Addition of WinCE support for OMAPL1XX, DA8XX, OMAPL138, DA850

Support for WinCE has been added on OMAPL1XX, DA8XX, OMAPL138, DA850.

4.6.9.1 To configure DSPLink for DA8XX

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

4.6.9.2 To configure DSPLink for DA850

perl dsplinkcfg.pl --platform=DA850 --nodsp=1 --dspcfg_0=DA850GEMSHMEM --dspos_0=DSPBIOS6XX --gppos=WINCE --comps=ponslrmc

4.6.9.3 To configure DSPLink for OMAPL1XX

perl dsplinkcfg.pl --platform=OMAPL1XX --nodsp=1 --dspcfg_0=OMAPL1XXGEMSHMEM --dspos_0=DSPBIOS5XX --gppos=WINCE --comps=ponslrmc

4.6.9.4 To configure DSPLink for OMAPL138

perl dsplinkcfg.pl --platform=OMAPL138 --nodsp=1 --dspcfg_0=OMAPL138GEMSHMEM --dspos_0=DSPBIOS5XX --gppos=WINCE --comps=ponslrmc

4.6.10 Correction of floating point support for OMAPL1XX, DA8XX, OMAPL138, DA850

The build system of DSPLink for C674x based devices did not correctly support compilation for floating point operations.

This has been corrected. In order to support this, the compile flag for C674x based devices is now updated to mv6740 from mv6400+.

The system integrator/application writer now has to edit below files to update the paths for BIOS, XDC, IPC and CG Tools installations in:

c674x_6.xx_linux.mk - To Build for BIOS 6 based devices (DA8XX, DA850) on a Linux host machine.

c674x_6.xx_windows.mk- To Build for BIOS 6 based devices (DA8XX, DA850) on a Windows host machine.

c674x_5.xx_linux.mk - To Build for BIOS 5 based devices (OMAPL1XX, OMAPL138) on a Linux host machine.

c674x_5.xx_windows.mk – To Build for BIOS 5 based devices (OMAPL1XX, OMAPL138) on a Windows host machine.

5 Validation Status

This product release has been validated for basic sanity using the following configurations:

Sr. No	Platform	GPP-side			DSP-side	
		OS distribution	Tool-chain	Base port	Tool-chain	DSP/ BIOS
1	DA850	PSP SDK 03.20.00.11	Glibc arm-2009q1-203	PSP SDK 03.20.00.11	6.1.5	BIOS - 6.20.00.37 XDC - 3.15.00.50 IPC 1.00.00.40
2	OMAPL138	PSP SDK 03.20.00.11	Glibc arm-2009q1-203	PSP SDK 03.20.00.11	6.1.5	BIOS - 5.33.05
3	DM6446	PSP SDK 03.01	Glibc arm-2009q1-203	PSP SDK 03.01	6.0.18	5.32.04

Distribution files for Uclibc tool-chains are provided for OMAP2530, OMAP3530, DM6446, DM6467, DM6467T, DA850, DA8xx, OMAPL1xx and OMAPL138.

6 Known Issues

6.1 Generic

Identifier	Headline
SDOCM00035970	DSPLINK does not build correctly in CYGWIN environment

Description :

DSPLink make system does not build DSP-side sample applications correctly in CYGWIN environment. However, DSPLink libraries on DSP-side and GPP-side build can be completed successfully in CYGWIN environment.

Workaround:

Remove CYGWIN from the path on MSDOS prompt and build DSP-side sample applications using gmake.

Identifier	Headline
SDOCM00056316	Intermittent crashes are seen when running DSPLink sample applications on Montavista Linux based devices

Description :

On devices running Montavista Linux on the ARM (DM6446, DM6467, DM357, DA8xx, OMAP-L1xx), intermittent crashes are seen when running the sample applications. Crash is not seen on devices where GIT kernel is used (OMAP3530, OMAP2530, DM6437, DM648).

Observation:

The crash appears to happen when interrupts are sent very frequently to the ARM from the DSP.

Workaround:

None.

6.2 Platform Specific Issues

6.2.1 DM6446

Identifier	Headline
SDOCM00056114	CE appln fails if user calls the sequence:1) eng = Engine_open("myEng",.);2) Engine_close(eng);3) eng = Engine_open("myEng

Description :

An application will fail if it tries to do the following:

```
eng = Engine_open("myEngine", ...);
Engine_close(eng);
eng = Engine_open("myEngine", ...);
```

As far as DSPLink is concerned, this boils down to:

```
    Engine_open:
        LAD_startupDsp() =======switch-to-LAD-process======>
PROC_setup(); PROC_attach(); etc.
        PROC_attach()
    Engine_close:
        LAD_releaseDsp() =======switch-to-LAD-process======>
PROC_destroy();
```

Workaround:

None.

6.2.2 DM6467

Identifier	Headline
DSPLN00000734	<PROC> DSP-side configuration mismatch/failure is thrown if a test is run infinite number of times

Description :

<PROC> DSP-side configuration mismatch/failure is thrown if a test is run infinite number of times

Workaround:

Power cycle the EVM and re-try.

6.2.3 DM648

None.

6.2.4 DM6437

Identifier	Headline
SDOCM00044596	<sample> scale sample hangs in SWI mode on both debug and release modes.

Description :

Observation :

scale sample hangs only in SWI mode on both debug and release modes.

Same sample works fine on TSK mode.

Workaround:

None.

6.2.5 OMAP2530

Identifier	Headline
SDOCM00035953	OMAP2530: First run of any application times out in PROC_start.

Description :

The first run of any DSPLink application times out in PROC_start. The subsequent application runs without re-booting the board in between work correctly.

Workaround:

The GPP sources have been updated to reset and release the DSP twice as a workaround to ensure that DSP starts correctly in the first run. This workaround is present in the DSPLink release.

6.2.6 DM357

None.

6.2.7 OMAPL1XX

6.2.7.1 OMAPL1XX

DIO and SIO create calls are not supported in DSPLink SWI Mode for DSP_BootMode_NoBoot boot mode because of a DSP/BIOS restriction in calling these API's before main. Sample applications i.e. loop, scale with SWI's will not work in DSP_BootMode_NoBoot mode. DSPLink supports SWI based sample applications in the other boot modes DSP_BootMode_Boot_NoPwr, DSP_BootMode_Boot_Pwr, DSP_BootMode_NoLoad_NoPwr, DSP_BootMode_NoLoad_Pwr

6.2.7.2 Defects

Identifier	Headline
DOCM00051547	<Samples> Loop sample stops execution in-between on swi release mode.

Description :

Observation :

Loop sample stops execution in-between on swi release mode on both DA8xx and OMAPL1xx platforms.

The sample works fine on swi debug mode.

The sample won't hang, after ctrl+c able to run other samples without rebooting EVM.

Workaround:

None.

6.2.8 DA8XX/DA850/OMAPL138

DIO and SIO create calls are not supported in DSPLink SWI Mode for DSP_BootMode_NoBoot boot mode because of a DSP/BIOS restriction in calling these API's before main. Sample applications i.e. loop, scale with SWI's will not work in DSP_BootMode_NoBoot mode. DSPLink supports SWI based sample applications in the other boot modes DSP_BootMode_Boot_NoPwr, DSP_BootMode_Boot_Pwr, DSP_BootMode_NoLoad_NoPwr, DSP_BootMode_NoLoad_Pwr

6.3 Operating System specific Issues

6.3.1 Linux

Identifier	Headline
DSPLN00000222	Warning message about struct_module is seen while inserting the DSPLINK kernel module.

Description :

Following warning message is seen while inserting the DSPLINK kernel module:

no version for "struct_module" found: kernel tainted.

Observation :

Workaround:

Add a -m option to the call to modpost in $DSPLINK/make/Linux/<distribution.mk >.

Rebuild the kernel module. The warning goes away.

Please note that this warning can be safely ignored as it has no impact on DSPLink run time.

Identifier	Headline
DSPLN00000851	Some cleanup issues on Ctrl C being used to kill the application.

Description :

For sample applications, some issues are seen rarely when Ctrl C is used to kill the application

Workaround:

Attach signal handler for Ctrl C in application, and disable signal handling in DSPLink. Refer to guidelines specified in the “Guidelines and Limitations” section in the Release Notes.

Identifier	Headline
DSPLN00000833	RingIO sample app causes segmentation fault or crash on Ctrl C.

Description :

Observation :

The following are the observations during executing the sample application if we apply ctrl C in between.

1) The ring_io sample application generates segmentation fault.

2) DSPLink driver crashes.

Some times the issue 1 or 2 is observed.

Expectation :

It should cleanup the process correctly.

We have seen same issue with OMAP 2530.

Workaround:

None.

6.3.2 DSP/BIOS

Identifier	Headline
CQ17490	BIOS/XDC conf tool breaks DSP build, if the DSPLink installation directory contains '.' Character.

Description :

DSP Side build system generates all files in the build folder instead of sample directories. This would cause error if the DSPLink installation directory has '.' Character. For example, if the installation path is

/home/user/dsplink.160/

then DSP Side build will break.

Workaround:

Install DSPLink without a '.' In the path.

7 Guidelines and Limitations

7.1 Usage with non POSIX compliant threading library not advised.

DSPLink expects a POSIX compliant threading library to enable enhanced multi-process and multi-application support and multi-process signal cleanup. These features work as expected when DSPLink is used with a POSIX compliant threading library but fail with a non POSIX compliant library.

DSPLink has been verified with The Native POSIX Thread Library (NPTL).

These features of DSPLink will not work correctly with a non POSIX compliant library for e.g. LinuxThreads. Normal execution of a multi threaded application will not be affected but following features will not work as expected:

- Multi-process cleanup support on Linux

  Point signal handling within LinuxThreads is conducted on a per-thread basis rather than on a per-process basis as each thread has a separate process ID. Since a signal is sent to a dedicated thread, signals are serialized -- that is, the signals are funneled through this thread to other threads. This is in contrast to the POSIX standard's requirements for parallel handling of signals. For example, under LinuxThreads, signals sent via kill() are delivered to individual threads rather than to the process as a whole. This means that if that thread is blocking the signal, then LinuxThreads will simply queue in that thread and execute the handler only when that thread unblocks the signal, instead of executing the handler immediately in the other thread that does not block the signal.

  getpid () call returns the same process ID for all threads in a process. However, for LinuxThreads, a different ID is returned for each process. Due to this, cleanup on a crash/Ctrl C does not work correctly in a multi-threaded application using DSPLink. This leaves the DSPLink kernel module in an unstable state and results in failure when the next application is invoked. A re-boot of the platform and reloading the kernel module is required to restore the correct state

- Any DSPLink finalization API‘s called from a thread different from the one in which the corresponding initialization API’s have been called will fail.

  When linked with a POSIX compliant library, within a process, one thread can call PROC_setup and another thread can call PROC_destroy and the PROC_destroy will pass. In LinuxThreads, different threads have different process ids. The following are pairs of calls, the former being initialization and the latter finalization:

  PROC_setup <-> PROC_destroy

  PROC_attach <-> PROC_detach

  PROC_start <-> PROC_stop

  POOL_open <-> POOL_close

  MSGQ_transportOpen <-> MSGQ_transportClose

  In the above pairs of calls, if any of the initialization and finalization pair is called from the same thread, it works correctly. If the finalization API in any pair is called from a different thread within a process, it will fail.

- POOL opened within a thread cannot be used in another thread of the same process.

7.2 DSPLink and signals

DSPLink supports clean up of the DSPLink driver when process termination signals are sent to any process/thread which is using the driver on the Linux operating system. By cleanup, the expectation is that the next run of the application without rebooting the EVM is possible.

Signal handling in the DSPLink driver is enabled by default. The signals handled by DSPLink are:

- SIGHUP
- SIGINT
- SIGILL
- SIGABRT
- SIGFPE
- SIGSEGV
- SIGALRM
- SIGTERM

The number of signals handled by DSPLink as well as whether the driver should handle signals or not can be set in the configuration OS specific file $DSPLINK/config/all/CFG_Linux.c This section details the behavior of the driver when signals are received at various stages:

- Certain debug assertions may be seen which can be ignored
- A crash dump may be seen. This crash dump occurs because the thread in user space is waiting on notification of events in the kernel driver. Since the kernel semaphore on which the thread is waiting is deleted, a crash is seen.

However, this crash can be safely ignored as clean up occurs. The crash looks like:

```
[<c022a3d8>] (__compat_down_interruptible+0x0/0x1e8) from [<c022a20c>]
(__compat_down_interruptible_failed+0xc/0x20)
                [<bf008c38>] (SYNC_WaitSEM+0x0/0x320 [dsplinkk]) from
[<bf007400>] (UEVENT_GetBuf+0xa8/0x190 [dsplinkk])
                [<bf007358>] (UEVENT_GetBuf+0x0/0x190 [dsplinkk]) from
[<c007f16c>] (vfs_read+0xc0/0xf8)
                [<c007f0ac>] (vfs_read+0x0/0xf8) from [<c007f3b0>]
(sys_read+0x44/0x70)
                  r8 = C002B154   r7 = 00000000   r6 = 00000000   r5 =
FFFFFFF7
```

- A zombie process may be seen.

  Signal handling from a library is insufficient to cleanup after the application. The application writer must disable default DSPLink signal handling and write an application level signal handler which cleans up all application resources. The application level signal handler in the parent process must clean up all DSPLink resources as well as wait for all child processes to terminate. If the application does not register a signal handler and instead cleanup happens via DSPLink default signal handler in a child process, a zombie process may be seen.

  The theory behind this is: When a child process terminates while its parent is calling a wait function, the child process vanishes and its termination status is passed to its parent via the wait call. If a child process terminates and the parent

is not calling wait, the child process turns into a zombie process. The information about its termination—such as whether it exited normally and, if so, what its exit status is—would be lost. If the parent process does not call wait. Instead, when a child process terminates, is becomes a zombie process.

A zombie process is a process that has terminated but has not been cleaned up yet. It is the responsibility of the parent process to clean up its zombie children. The wait functions do this, too, so it's not necessary to track whether your child process is still executing before waiting for it. Suppose, for instance, that a program forks a child process, performs some other computations, and then calls wait. If the child process has not terminated at that point, the parent process will block in the wait call until the child process finishes. If the child process finishes before the parent process calls wait, the child process becomes a zombie. When the parent process calls wait, the zombie child's termination status is extracted, the child process is deleted, and the wait call returns immediately.

- Clean up is not successful when signal is received in a process where PROC_attach has not yet been called. This is applicable only when the process which has done PROC_setup is different from the current one. This is because the signal handler is setup in PROC_attach and the resources reserved in PROC_setup will not get freed.

The recommendation from DSPLink is that application writers should write their own application specific cleanup functions for process termination signals. This is because DSPLink driver can clean up only its own resources whereas the application signal handler will clean up other application resources including those of the DSPLink driver.

8 Technical Support

8.1 Support for DSP/BIOS™ LINK

8.1.1 TI External wiki

A significant amount of information about DSPLink including FAQs and trouble-shooting guides are present and the TI external wiki. These can be used as a first-level source of information for DSPLink:

http://www.tiexpressdsp.com

8.1.2 Support for TI products

Procedure to report problems/issues for TI products is available at the TI support page:

External forums at: http://community.ti.com

Or: http://support.ti.com