DSP/BIOS 6.20 Getting Started Guide

April 30, 2009

Texas Instruments

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 TI 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. Information of third parties may be subject to additional restrictions.

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
DLP® Products	www.dlp.com	Broadband	www.ti.com/broadband
DSP	dsp.ti.com	Digital Control	www.ti.com/digitalcontrol
Clocks and Timers	www.ti.com/clocks	Military	www.ti.com/military
Interface	interface.ti.com	Optical Networking	www.ti.com/opticalnetwork
Logic	logic.ti.com	Security	www.ti.com/security
Power Mgmt	power.ti.com	Telephony	www.ti.com/telephony
Microcontrollers	microcontroller.ti.com	Video & Imaging	www.ti.com/video
RFID	www.ti-rfid.com	Wireless	www.ti.com/wireless
RF/IF and ZigBee® Solutions	www.ti.com/lprf		

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265

Preface

Read This First

About This Manual

This document describes the process of installing and using DSP/BIOS 6 for use with Code Composer Studio v4 (CCSv4).

How to Use This Manual

This document provides installation instructions in Chapter 1 and step-by-step instructions for starting to use DSP/BIOS in Chapter 2.

After you install DSP/BIOS, you might want to review the release notes in the installation before reading further.

After you have read this document, you should see the *DSP/BIOS 6 User's Guide* (SPRUEX3) and the online CDOC reference for more information.

Notational Conventions

This document uses the following conventions:

- Program listings, program examples, and interactive displays are shown in a special typeface. Examples use a bold version of the special typeface for emphasis.

  Here is a sample program listing:

```
#include <xdc/runtime/System.h>

int main(){
    System_printf("Hello World!\n");
    return (0);
}
```

- Square brackets ([and]) identify an optional parameter. If you use an optional parameter, you specify the information within the brackets. Unless the square brackets are in a bold typeface, do not enter the brackets themselves.

Related Documentation from Texas Instruments

DSP/BIOS 6 User's Guide (SPRUEX3)

RTSC-Pedia wiki: http://rtsc.eclipse.org/docs-tip

Code Composer Studio Mediawiki: http://tiexpressdsp.com/wiki/index.php?title=CCSv4

CDOC API Reference online help system

Trademarks

The Texas Instruments logo and Texas Instruments are registered trademarks of Texas Instruments. Trademarks of Texas Instruments include: TI, Code Composer, Code Composer Studio, DSP/BIOS, SPOX, TMS320, TMS320C54x, TMS320C55x, TMS320C62x, TMS320C64x, TMS320C67x, TMS320C28x, TMS320C5000, TMS320C6000 and TMS320C2000.

Windows is a registered trademark of Microsoft Corporation.

Linux is a registered trademark of Linus Torvalds.

All other brand or product names are trademarks or registered trademarks of their respective companies or organizations.

Contents

Chapter 1

Using DSP/BIOS with CCSv4

This chapter describes how to build DSP/BIOS examples and applications.

Topic **Page**

1.1 Installing DSP/BIOS as Part of CCSv4

If you are installing Code Composer Studio v4 (CCSv4), all the components you need to use DSP/BIOS 6.20 are installed as part of the CCSv4 installation. When you perform the CCSv4 installation, leave the checkboxes for installing XDCtools and DSP/BIOS 6 checked.

1.2 Creating Projects that Use DSP/BIOS 6.20

Follow these steps to use CCSv4 to create a CCS project that can use DSP/BIOS 6.x:

1) Open CCSv4 and choose **File > New > CCS Project** from the menu bar.

2) In the New CCS Project dialog, type a Project Name. For example, to begin creating a project using the "hello world" example code provided with DSP/BIOS, you can type "hellobios6". The default project location automatically reflects the project name. Then, click **Next**.

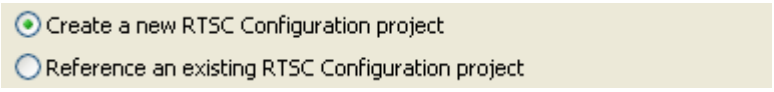
3) Select your platform type in the Project Type field. For example, you might select "C6000". Then, click **Next**.

4) Click **Next** in the "Additional Project Settings" page.

5) In the **Device Variant** row of the "CCS Project Settings" page, select a filter in the first field. This shortens the list of device variants in the second field. Then, select the actual device you are using. For example, you might select “TMS320C64X” in the filter field and "Generic C64x+ Device" in the second field. Depending on your device, you might also need to adjust the **Device Endianness** and **Runtime Support Library** settings.

6) In the Target content area, select the **Enable RTSC support** option. Then, click **Next**.

7) In the “Referenced RTSC Configuration” page, choose whether you want to create a new RTSC configuration or use an existing one. If you are getting started with DSP/BIOS 6, you will probably need to create a new configuration. Then, click **Next**.



CCSv4 creates a separate project to contain files related to RTSC configuration, which allows you to statically configure DSP/BIOS modules and objects. The RTSC configuration project is referenced by the application project, so that when you build the application project, the RTSC configuration is also rebuilt if it has been changed.

8) On the “New RTSC Configuration Project” page, accept the default name and location for the configuration project. Then, click **Next**.

9) On the “RTSC Configuration Settings” page, make sure the versions of XDCtools, DSP/BIOS, and other products that you want to use are selected. By default, the most recent versions are selected.

10) Also on the “RTSC Configuration Settings” page, click the drop-down arrow next to the **RTSC Platform** field. CCSv4 scans the available packages for available platforms. Click on the list and choose the platform you want to use.

11) You can choose which libraries you want the configuration build to link with in the **RTSC Build-Profile** field. The **RTSC Target** setting is based on device settings you made on earlier pages, and should not need to be changed. Then, click **Next**.

12) On the “RTSC Configuration Templates” page, check the **Create a project using one of the templates** box if you want to use a provided template as the basis for your configuration. For example, you can create a project using the "hello world" example by selecting the “Hello Example” under the “DSP/BIOS” category. See Section 1.3 for information about choosing a template. Then, click **Finish**.

This adds two projects to the C/C++ Projects list in CCS: the main project for your application and the RTSC configuration project. The RTSC configuration project contains a *.cfg file, that will contain the static configuration of XDCtools and DSP/BIOS modules and objects.

1.3 Configuration Templates

When you are creating a CCSv4 project that enables RTSC support, you can select a RTSC Configuration Template. The actual files for the DSP/BIOS templates are copied from BIOS_INSTALL_DIR\packages\ti\sysbios\genx\templates. When you highlight a template, a brief description is provided to the right.

The following DSP/BIOS templates are available when you choose **File > New > CCS Project** and get to the RTSC Configuration Templates page of the project creation wizard. These templates copy *both* a default .c file into the CCS Project and a default .cfg file into the RTSC Configuration Project.

- **Clock Example.** This template uses the clock.c and common.cfg files.
- **Error Example.** This template uses the error.c and error.cfg files.
- **Event Example.** This template uses the event.c and common.cfg files.
- **Hello Example.** This template uses the hello.c and common.cfg files.
- **Legacy Example.** This template uses the legacy.c and legacy.cfg files.
- **Log Example.** This template uses the log.c and log.cfg files.
- **Memory Example.** This template uses the memory.c and memory.cfg files.
- **Mutex Example.** This template uses the mutex.c and common.cfg files.
- **RTA Example.** This template uses the stairstep.c and stairstep.cfg files.
- **Static Example.** This template uses the static.c and static.cfg files.
- **Swi Example.** This template uses the swi.c and common.cfg files.

Additional templates are available for the Inter-Processor Communication (IPC) product, which works with DSP/BIOS in applications. See the *DSP/BIOS Inter-Processor Communication (IPC) User's Guide* (SPRUGO6) for details.

The following DSP/BIOS templates are available when you choose **File > New > RTSC Configuration Project** and get to the RTSC Configuration Templates page of the project creation wizard. These templates copy *only* a default .cfg file into the RTSC Configuration Project (but no .c file).

- **Full BIOS Configuration.** Enables the use of Task, Swi, Idle, and Clock scheduling. Also enables the Event, Mailbox, and Semaphore modules. It enables the Memory module and creates a default memory heap called "heap".
- **Full BIOS with RTA Configuration.** In addition to the configuration in the previous template, this configuration enables and configures logging-related modules. Configures a loggerBuf, diagnostics, the RTA agent, and the RTDX driver.

1.4 Additional Ways to Create Configurations

In addition to creating new projects as described in the previous sections, you can create a standalone RTSC configuration project that you will later reference from other CCS projects. To do so in CCSv4, choose **File > New > RTSC Configuration Project**. Use the wizard to define the configuration project. The wizard is similar to the one you use for a new CCS project, but omits pages and fields that are not required.

To add a RTSC configuration file to an existing project, choose **File > New > RTSC Configuration File**. Then click **Next**. Type a name for the file and click **Finish**.

1.5 Rules for Working with CCS Build Settings

After you have created a RTSC Configuration Project, you can change the properties of the project in CCSv4 by right-clicking the project name and choosing **Properties**.

However, you should be careful about changing the CCS Build Settings for a RTSC configuration project. The build settings for the configuration project much match or be compatible with those of all application projects that reference the RTSC configuration project. So, if you change the CCS Build Settings for a RTSC configuration project, you should also change the build settings for the application projects that use that configuration.

In the “CCS Build Settings” category of the Properties dialog, the **General** tab applies to compiler settings, the **RTSC Configuration** tab applies to the “configuro” utility used to process the .cfg file, and the **Link Order** tab applies to the linker settings.

Note that if there is any platform-specific configuration in your .cfg file, you much change those settings in addition to any changes you make to the CCS Build Settings.

1.6 Using XGCONF to Configure DSP/BIOS Modules and Objects

XGCONF is a tool that allows you to graphically create and view RTSC configuration scripts. (If you have used DSP/BIOS 5.x, it is somewhat similar to the DSP/BIOS Configuration Tool.)

To start XGCONF, right-click on a *.cfg file in the "C/C++ Projects" view and select **Open with > XGCONF**.

For information about using XGCONF, see the RTSC-Pedia page about XGCONF at http://rtsc.eclipse.org/docs-tip/XGCONF_User's_Guide.

1.7 XDCtools and RTSC Documentation

XDCtools is a set of tools that enable the creation and use of RTSC packages. RTSC (pronounced rit-see) stands for Real-Time Software Components. It allows products to be developed as reusable packages, and is described in the **RTSC-pedia** at http://rtsc.eclipse.org/docs-tip.

XDCtools and RTSC define the configuration language used by DSP/BIOS (and other products that make use of RTSC). For DSP/BIOS 6.x versions, the configuration scripts are stored in .cfg and other files.

For further documentation on RTSC and XDCtools, use the Windows Start menu to choose **Texas Instruments > XDCtools 3.xx > XDCtools Documentation.** Also see the XDCtools release notes.

Chapter 2

Using DSP/BIOS as a Standalone Product

This chapter describes how to use DSP/BIOS outside the CCSv4 environment, for example, from a command line.

Topic **Page**

2.1 Using DSP/BIOS as a Standalone Product

This chapter applies only if you are using DSP/BIOS outside the CCSv4 environment. The steps described here are not required if you are using CCSv4 projects.

2.2 Installing DSP/BIOS as a Standalone Product

If you are installing DSP/BIOS 6 as a standalone software product, use the steps for installing XDCtools and DSP/BIOS in the following subsections.

2.2.1 Installing XDCtools

You must install XDCtools in order to use DSP/BIOS 6. Please refer to the *XDCtools Getting Started Guide* for details. XDCtools provides tools to enable the creation and use of Real-Time Software Components (RTSC) packages.

2.2.2 Installing DSP/BIOS 6

To install DSP/BIOS as standalone software, follow these steps:

1) Place the distribution file into a temporary location.

2) Double-click on the distribution file to start the installation process. The installation directory, <*bios_install_dir*>, can be anywhere on your system, but make sure there are no spaces in the full path to <*bios_install_dir*>.

3) View documentation for the DSP/BIOS 6 packages at <*bios_install_dir*>/docs. See the *XDCtools Getting Started Guide* for information on using the xdc.tools.cdoc.sg tool to generate and view documentation for repositories.

2.3 Setting Up the XDCPATH Environment Variable

XDCPATH is a required environment variable. It is the path where XDCtools looks to locate all packages. All DSP/BIOS configuration scripts only mention packages by name, not by their location, and therefore the scripts are physically *portable*. If you copy a package to a different location or a different system, only the XDCPATH environment variable needs to be changed. XDCtools provides a tool, xdc.tools.path.sg, to view and manage XDCPATH. See *XDCtools Getting Started Guide* for details.

Using XDCPATH, we can select repositories (e.g. select among different DSP/BIOS installs) and select a config.bld script.

Add the DSP/BIOS repository <*bios_install_dir*>/packages to the XDCPATH environment variable. In the Windows System Properties in the Control Panel, select Environment Variables under the Advanced tab. Add a new system variable named XDCPATH, with the value <*bios_install_dir*>/packages.

Other operating systems require a slightly different command to set environment variables. See the documentation for your operating system for details.

2.4 Managing the config.bld File

The build configuration script, config.bld, is a master setup script that completely defines the RTSC targets and the environment in which individual RTSC packages will build. For practical purposes, however, this script is usually just a list of targets and paths to their Code Gen tools.

The intention behind the build configuration script is the following: it is a small, non-portable piece of the build flow, because it defines the location of the various Code Gen tools on your system. Another use for the build configuration script is to provide a common build setup for multiple packages.

DSP/BIOS 6 ships a default config.bld script at

```
<bios_install_dir>/etc/config.bld.default
```

1) View this script. This default script names all the targets and platforms that DSP/BIOS 6 supports. It also points to the Code Gen tools. Please refer to the release notes on supported versions of Code Gen tools.

2) Create a directory <localRepository> anywhere on your system. Copy config.bld.default into <localRepository>/config.bld. All targets are disabled by default. Please uncomment the target of interest in the Build.targets array in config.bld. You will also need to edit this script for the following reasons.

   - Location of your Code Gen tools is different.
   - You are interested in a particular platform.

3) Add <localRepository> to the XDCPATH environment variable.

2.5 Generating and Building DSP/BIOS Examples

DSP/BIOS does not ship any pre-built examples. Instead it ships a package ti.sysbios.genx which will generate the examples in a directory of your choice. ti.sysbios.genx allows the examples to be customized for your config.bld script. It also allows re-generation of the examples in case of corruption.

View options for ti.sysbios.genx by typing the following:

```
xs ti.sysbios.genx --help
```

Generate DSP/BIOS 6 examples as follows:

```
xs ti.sysbios.genx <myExamples>
```

Examples are generated in the <myExamples> folder for all targets and platforms listed in the config.bld referred to in your XDCPATH environment variable.

You can optionally specify a single target and a platform listed in the config.bld on the command line.

```
xs ti.sysbios.genx -t ti.targets.C64
  -p ti.platforms.dsk6416 <myExamples>
```

By default, the examples are generated without debug support. If you want to generate the examples for debugging, you can use the –r option to select a profile for building the examples. For example, the following command line generates the examples for debugging:

```
xs ti.sysbios.genx -t ti.targets.C64 -p ti.platforms.sim6xxx
-r debug <myExamples>
```

The –r options are: debug, whole_program, and whole_program_debug. The default is whole_program. The "debug" profile allows you to step through the sources if sources are available. The "whole_program" profile has negligible debug symbols. The "whole_program_debug" profile has more debug symbols and improves the debug experience when compared to "whole_program". It is recommended that production code be built with whole_program or whole_program_debug.

Many examples use the same DSP/BIOS configuration. Use incremental builds to build the examples if you want to prevent unnecessary builds of the configuration package.

To use makefiles to build the examples, use commands like the following:

- Use the top-level makefile to build all examples.

```
cd <myExamples>
gmake
```

- To build one example, build <myExamples>/common/<platform_dir> first. Many examples depend on <myExamples>/common/<platform_dir>.

```
cd <myExamples>\common\<platform_dir>
gmake
```

- Use gmake to build the individual examples:

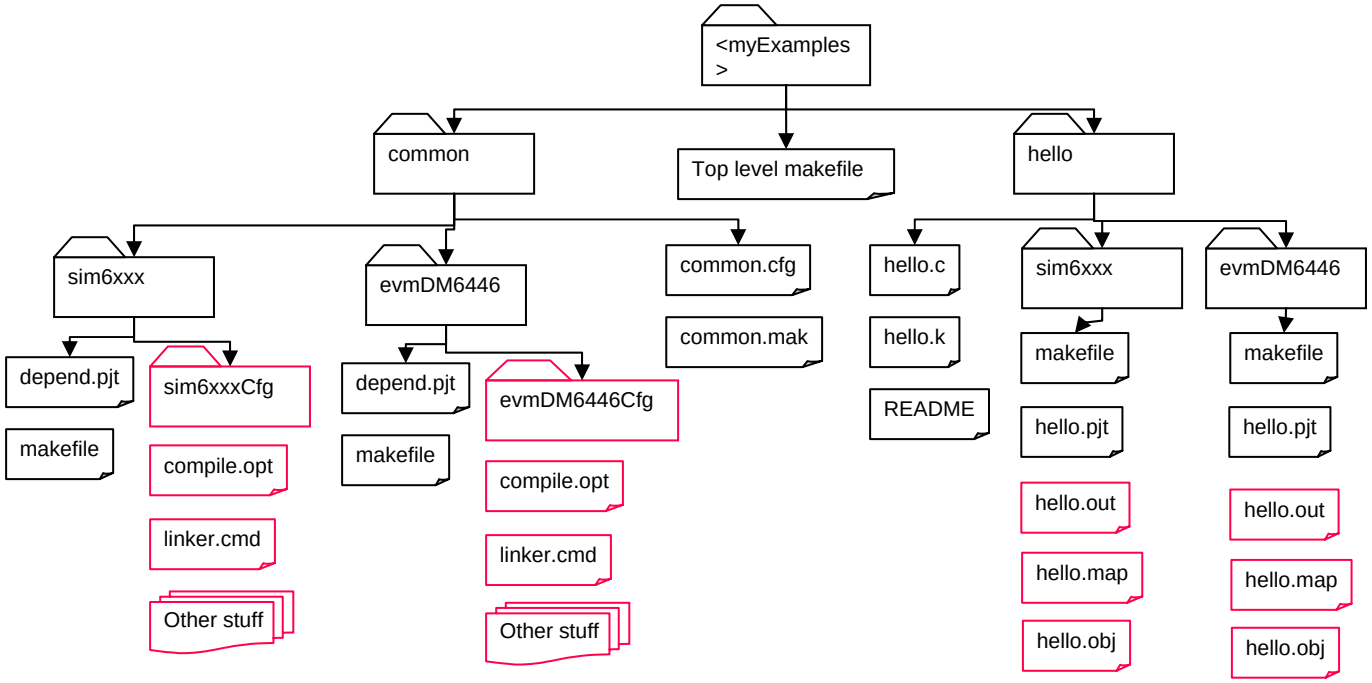
```
cd <myExamples>\hello\<platform_dir>
gmake
```

NOTE: During the installation, the XDCROOT environment variable is set by the installer. In addition, %XDCROOT% is added to the beginning of the system and user path(s). However, the <examples>/common/common.mak file references the specific version of XDCtools it was installed with. If you update your XDCtools installation and want to rebuild the examples with the new version, you need to update the XDCROOT definition in the <examples>/common/common.mak file.

See the README for individual examples for more details on how to build them.

You can use <myExamples>/common/common.mak to pass additional linker (e.g. --xml_link_info) and compiler options (e.g. –pdr) to the makefiles used by all examples.

<myExamples> has the following structure when built. The expected output is in testcase.k files.



2.6 Configuring DSP/BIOS

DSP/BIOS configuration allows clients to specify modules to be used in an application, create static objects, and modify their properties as part of the application build process. DSP/BIOS configuration also allows clients to modify program-level configuration parameters such as stack size and the section map.

2.6.1 DSP/BIOS Configuration Script

DSP/BIOS configuration is done through a script. The script is a file with ".cfg" extension that contains a RTSC script. Such scripts are an extension of JavaScript. Please see <xdc_install_dir>/packages/xdc/cdk/langref/xdcScript.pdf for details.

The DSP/BIOS configuration script by itself is not sufficient to create a DSP/BIOS configuration; you must also specify the target and platform that DSP/BIOS will be running on. The script is the client's portable input into a DSP/BIOS configuration package.

In the simple case, this script can list all the modules used. See <myExamples>/common/common.cfg. Creating static objects and manipulating them is an optimization and is the focus of the static example.

1) Create a <cfgScript>. This file can be located along with the application code (static example has its own static.cfg script) or maintained separately (hello

example uses common/common.cfg). For example client.cfg may be located at C:\MyProjects\client\client.cfg.

2) Edit this file using any text editor. Refer to the DSP/BIOS 6 documentation for details on configuration parameters.

2.6.2 Create and Build a DSP/BIOS Configuration Package

A DSP/BIOS configuration package is created using xdc.tools.configuro.

The configuration package is specific to a DSP/BIOS configuration script, platform/target pair and development environment.

View options supported by xdc.tools.configuro as follows:

```
xs xdc.tools.configuro --help
```

Use xdc.tools.configuro to create and build a DSP/BIOS configuration package. The configuration package <myCfgPkg> can be located anywhere on your system.

```
xs xdc.tools.configuro --cb -t<target> -p<platform>
  -o <myCfgPkg> -r whole_program  <cfgScript>
```

For example:

```
xs xdc.tools.configuro --cb –t ti.targets.C64
 -p ti.platforms.sim6xxx
 -o C:\MyProjects\client\configPkg
 -r whole_program C:\MyProjects\client\client.cfg
```

This step of running xdc.tools.configuro can be integrated into a makefile. See the *XDCtools Getting Started Guide* for details on running xdc.tools.configuro in a makefile. Also see the makefiles used by the examples.

The configuration package needs to be rebuilt for any change in the <cfgScript>.

2.7 Building an Executable

2.7.1 Executable Build Flow Changes

A compiler.opt file located in the DSP/BIOS configuration package needs to be added to the compiler options using -@ for ti targets.

The linker.cmd file located in the DSP/BIOS configuration package needs to be added to linker options. This file specifies the obj files and command file.

Add the RTS library to linker options from Code Gen tools specified in your config.bld. **NOTE:** The RTS library must come after the linker.cmd file in the link order. Ensure this is the case in your makefile.

2.7.2 C Source File Changes

- #include <xdc/std.h> to get RTSC types.
- #include <xdc/cfg/global.h> to get access to Program.globals generated by the DSP/BIOS configuration.
- Include module header files.

2.8 Microsoft Visual Studio C++ 2005 Installation Notes

Skip this section if you are not going to build your application on Windows using the windows emulation support in DSP/BIOS 6.

Building on Microsoft Windows requires Microsoft Visual Studio C++ 2005 Express Edition to be installed on your computer. Download and install the following components (available for free) from the Microsoft Web Site http://msdn.microsoft.com/vstudio/express/

- Microsoft Visual C++ 2005 Express Edition
- Microsoft Visual C++ 2005 Express Service Pack 1
- Microsoft Platform SDK for Visual C++ 2005 Express (Microsoft Windows Server 2003 R2 Platform SDK)

Make sure to set the environment variables required for Visual Studio C++. Run the following batch script at the DOS prompt before building the examples:

```
C:\Program Files\Microsoft Visual Studio 8\Common7\Tools\vsvars
32.bat
```

Update the Win32.rootDir in your config.bld script to point to your Visual C++ installation folder. The rootDir must not contain spaces or backslash characters. You must use the short pathname to the Visual C++ installation folder. Use the following command in a DOS Shell to display the short pathname of the Visual C++ installation folder. (Enter the command all on one line.) Then replace all backslash characters with forward slashes.

```
for /f "delims=" %i in ("C:\Program Files\Microsoft Visual
Studio 8") do @echo %~fsi
```

For example, here is how Win32.rootDir would be set on a typical system:

```
Win32.rootDir = "C:/PROGRA~1/MID05A~1";
```

Update the Win32.vcPath["VC8"].sdkPath in your config.bld script to point to your Platform SDK installation folder. As with the Win32.rootDir, the sdkPath must not contain spaces or backslash characters, and you must use the short pathname to the installation folder.

For example, here is how the sdkPath would be set on a typical system:

```
Win32.vcPath["VC8"].sdkPath = " C:/PROGRA~1/MI9547~1";
```