DSP/BIOS Real-Time Analysis SDK for OSK5912

MontaVista Linux Professional Edition 3.1

Literature Number: SPRU819
August 2004

TEXAS INSTRUMENTS

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

Preface

Read This First

About This Manual

The DSP/BIOS Real-time Analysis (RTA) Software Development Kit (SDK) allows software developers who are using the OSK5912 Reference Platform to analyze and monitor, at run-time, DSP-side DSP/BIOS applications from the GPP ARM running Linux. This manual describes this SDK.

Notational Conventions

This document uses the following conventions:

❏ In file paths, <RTASDK_dir> is the directory where you installed the DSP/BIOS RTA SDK.

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

Related Documentation From Texas Instruments

The following books describe TMS320 devices and related support tools. You can find these books on the Texas Instruments web site at www.ti.com. Search for the literature number to find the book you want.

TMS320 DSP/BIOS User's Guide (literature number SPRU423) provides an overview and description of the DSP/BIOS real-time operating system.

TMS320C5000 DSP/BIOS Application Programming Interface (API) Reference Guide (literature number SPRU404) describes DSP/BIOS API functions, which are alphabetized by name. The API Reference Guide is the companion to this user's guide.

Trademarks

Contents

Chapter 1

About the DSP/BIOS Real-Time Analysis SDK

This chapter provides an overview of the DSP/BIOS RTA SDK.

Topic	Page

1.1 Purpose

The DSP/BIOS Real-time Analysis (RTA) Software Development Kit (SDK) allows software developers who are using the OSK5912 Reference Platform to analyze and monitor, at run-time, DSP-side DSP/BIOS applications from the GPP ARM running Linux. The SDK provides support for the following actions:

❏ Obtaining LOG and STS data, including CPU busy information.

❏ Transferring RTA data via the API from the DSP to the GPP.

❏ Using a fully-functional sample application to obtain standard RTA data without having to write any code.

In addition to retrieving LOG/STS data at run-time, the DSP/BIOS RTA SDK supports postmortem LOG/STS analysis.

The design goals of the SDK were to make it easy to determine whether the system is operating within its design constraints, to help resolve problems, and to find out if the system is meeting its performance requirements.

1.2 Software Architecture

The software architecture of DSP/BIOS RTA SDK is shown in Figure 1-1:

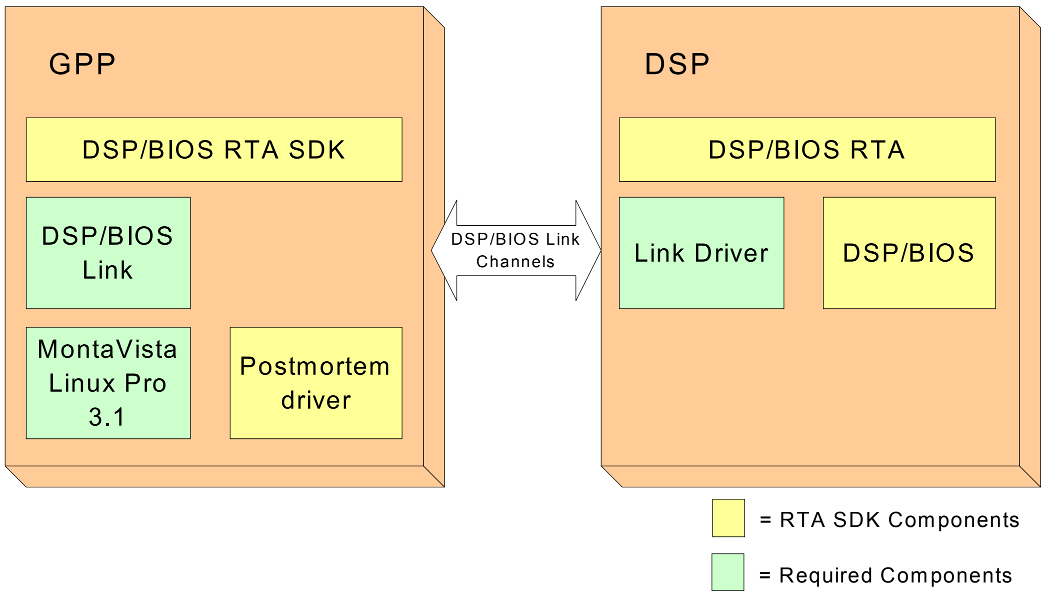

Figure 1-1. Software Architecture of DSP/BIOS RTA SDK

The DSP/BIOS RTA SDK allows a GPP-side application to retrieve DSP/BIOS LOG and STS information from the DSP-side application at run-time. The software architecture for accomplishing this has components running on both the GPP and the DSP.

1.2.1 Channels and Data Exchange

The RTA SDK uses DSP/BIOS Link to establish communication between GPP and DSP. It requires two dedicated Link channels: one for each communication direction (GPP-to-DSP and DSP-to-GPP).

The DSP/BIOS RTA SDK manages transferring DSP/BIOS LOG and STS data from the DSP to the GPP at run-time. To reduce the overhead and intrusiveness of the data transfer, all work is done while the system is running in the DSP/BIOS idle loop. A typical data exchange transaction can be illustrated conceptually as follows:

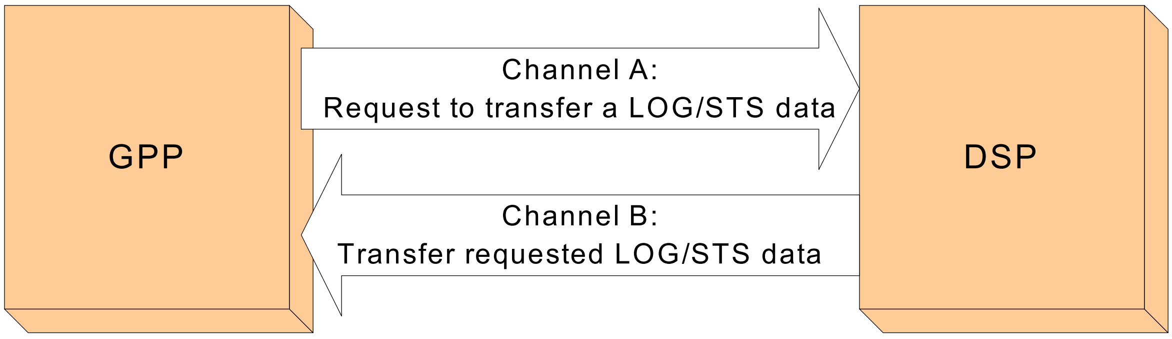
GPP
Channel A:
Request to transfer a LOG/STS data
DSP
Channel B:
Transfer requested LOG/STS data

Figure 1-2. Typical RTA SDK Data Exchange Transaction

1.2.2 DSP/BIOS Link Configuration

The bioslink_common.tci file in the <RTASDK_dir>/ti/bios/rta/examples/dsp/common directory shows how to configure DSP/BIOS for the RTA SDK. The following statements from this file are used to set the properties of various configuration modules.

```
/* BIOSLINK related settings */

bios.GIO.ENABLEGIO = true;              /* enable GIO */

bios.HST.CHNLDRVNAME = "default";       /* host channel driver name */
bios.HST.HOSTLINKTYPE = "BIOSLINK";     /* HST link type for BIOSLINK */
bios.HST.BIOSLINKDEVICE = "dsplink";    /* set BIOSLINK device */

bios.RTA_toHost.biosLinkChnlNbr = 14;   /* set DSP/BIOS Link channel # */
bios.RTA_fromHost.biosLinkChnlNbr = 15; /* set DSP/BIOS Link channel # */
```

- On DSP side, DSP/BIOS Link has an IOM driver implementation that is delivered through the Link installation. The RTA SDK uses the GIO class driver APIs to access that device driver. Hence, GIO support is enabled.
- The HST configuration allows you to select the communication link type and some of the global properties. In RTA SDK case:
  - CHNLDRVNAME = The name of the embedded host-side dynamically loadable channel driver. If the value is "default", the RTA SDK's GPP-side component uses the "libcd_bl.so" driver. If you want to define a driver explicitly, enter the file name without the file extension. For example, “libcd_bl”.
  - HOSTLINKTYPE = The option "BIOSLINK" is used for the RTA SDK.
  - BIOSLINKDEVICE = The name of the created DSP/BIOS Link device.
- The DSP/BIOS RTA SDK requires two dedicated Link channels. Hence, the selected channel numbers must not be in conflict with channel numbers used by other applications. For instance; if you are working with RF6, the Link channel numbers "0" and "1" are used by RF6.
  - RTA_toHost.biosLinkChnlNbr = The DSP/BIOS Link channel number reserved for communication from the DSP to the GPP.
  - RTA_fromHost.biosLinkChnlNbr = The DSP/BIOS Link channel number reserved for communication from the GPP to the DSP.

1.3 Compatibility

The DSP/BIOS RTA SDK is compatible with the following products:

- C55x Codegen v2.56
- DSP/BIOS v5.00
- DSP/BIOS Link v1.10.01
- MontaVista Linux 3.1

1.4 References

- *DSP/BIOS 5.00 Getting Started Guide*
- *DSP BIOS RTA SDK Release Notes*
- *DSP/BIOS Link OSK5912 Starter Kit*, MontaVista Linux Professional Edition 3.1 User's Guide Version 1.10.01 (LNK 058 USR)
- *DSP/BIOS Driver Developer's Guide*, dated November 2002 (SPRU616)
- *DSP/BIOS Application Programming Interface (API) Reference Guide*, dated April 2004 (SPRU404)
- *DSP/BIOS Real Time Analysis Application Programming Interface (API) Reference Guide*

1.5 Terms and Definitions

Table 1-1. Terms and Definitions

Term	Definition
API	Application Programming Interface
DSP	Digital Signal Processor
DSP/BIOS	Texas Instruments' Real-Time Operating System
DSP/BIOS Link	DSP/BIOS Link is runtime software that simplifies communication between embedded host and DSP.
Embedded Host	Embedded Host Platform Running on GPP.
Embedded Platform	Embedded development platform which is designed by using various CPU configurations such as one DSP, one GPP-DSP, or multiple of them.
GIO	General Input-Output module
GPP	General Purpose Processor (ARM)
HST	Host Channel Interface module
IOM	I/O Mini-Driver module
LOG	DSP/BIOS logging support module
MVL PRO31	MontaVista Linux Professional Edition 3.1
OMAP	Open Multimedia Architecture Platform
OSK	OSK5912 Starter Kit
Postmortem	Ability to obtain final RTA data after DSP has crashed
RF6	Texas Instruments' Reference Framework 6
RTA SDK	DSP/BIOS Real-Time Analysis Tools Software Development Kit
RTDX	Real-Time Data Exchange
STS	DSP/BIOS statistical data support module

Chapter 2

Installation and Setup

This chapter describes how to install, build, and configure applications associated with the DSP/BIOS RTA SDK.

2.1 Installing the SDK

The following steps explain how to install the product on a Linux machine:

1) Copy or download the distribution file specific to your system to a temporary location.
2) Create an installation directory on your system outside any CCStudio installations. The directory must not have any spaces in the path. For example, use "/usr/RTASDK-Rel". Do not install this SDK in the same directory as the DSP/BIOS 5.00 installation. The remainder of this document uses <RTASDK_dir> to refer to the directory you select here. Make sure that you have enough disk space allocated (~60 MB), or the installation will fail with a Java error.
3) Run the dsp_bios_rta_setuplinux_5.00.00.00.bin self-extracting installation file. Accept the End User License Agreement, and set the destination directory to the one created in the previous step. Use the "-console" flag if you are installing from the command line. For example:

   ```
   ./dsp_bios_rta_setuplinux_5.00.00.00.bin -console
   ```

2.2 Build Settings and Actions

This section describes the steps you must perform in order to be ready to rebuild the GPP-side and DSP-side sample applications.

2.2.1 Assumptions

The DSP/BIOS RTA SDK depends upon the following products. It assumes that they have all been installed, set up, and built successfully:

- ❏ C55x Codegen v2.56
- ❏ DSP/BIOS v5.00
- ❏ DSP/BIOS Link v1.10.01

2.2.2 DSP/BIOS Link Configuration

DSP/BIOS Link supports various configurations (for example, PROC and PROC+CHNL). Because the RTA SDK uses Link channels to establish communication between the GPP and DSP, the minimum required DSP/BIOS Link configuration is "PROC+CHNL". Please see the “Build Configuration” chapter in the *DSP/BIOS Link User's Guide* to configure DSP/BIOS Link.

After you configure and build DSP/BIOS Link successfully, the DSP/BIOS RTA SDK channel driver and the "loadandrun" example need to be rebuilt as well.

2.2.3 GPP-Side Settings Before Building

The following settings need to be made to build the sample examples on the GPP side:

1) Edit <RTASDK_dir>/ti/bios/rta/examples/gpp/common/config.mak to make the file to point to the DSP/BIOS Link and MontaVista Linux toolchain directories.

```
# Path to DSP/BIOS Link
DSPLINK_DIR    := <DSP/BIOS Link install dir>
# Path to MontaVista Linux toolchain
MVL_TOOLCHAIN   := <mv linux toolchain root dir>
```

2) Edit <RTASDK_dir>/ti/bios/rta/examples/gpp/rtatrace/makefile to make it point to the RTA SDK installation directory.

```
# Path to RTA SDK
RTA_SDK_DIR     := <RTA SDK install dir>
```

3) Edit the <RTASDK_dir>/ti/bios/rta/sdk/pmrdrv/makefile file to make it point to the MontaVista Linux Kernel include directory.

```
# Linux Kernel Include directory
KERNEL_INC_DIR  := <Linux Kernel Include dir>
```

4) Edit the <RTASDK_dir>/ti/bios/rta/sdk/chnldrv/makefile file to make it point to the RTA SDK installation directory.

```
# Path to RTA SDK
RTA_SDK_DIR     := <RTA SDK install dir>
```

2.2.4 DSP-Side Settings Before Building

If you want to rebuild the "hello" example or want to create your DSP project under the <RTASDK_dir>/ti/bios/rta/examples/dsp/basic directory, you need to point the following definitions in the <RTASDK_dir>/ti/bios/rta/examples/dsp/common/config.mak file to the correct locations:

❏ RTASDK_INSTALL_DIR = The root directory of the DSP/BIOS RTA SDK installation.

❏ SABIOS_INSTALL_DIR = The root directory of the DSP/BIOS installation.

❏ COMPONENT_ROOT = The root directory of the RTDX, CSL, and PSL components.

❏ C55X_CODEGEN_ROOT = The root directory of the C55x Codegen V2.56 installation.

❏ BIOSLINK_DIR = The directory containing **dsplink.l55l** in the DSP/BIOS Link installation.

If you want to use a project contained in the DSP/BIOS 5.00 installation, you must also add the following definitions to the <SABIOS_dir>/ti/bios/examples/common/config.mak file:

❏ LNKBIOSLINKLIB_DIR = <RTASDK_dir>/ti/bios/rta/rtalnk/lib

❏ BIOSLINK_DIR = <directory that contains **dsplink.l55l**>

2.2.5 Building the Examples

The examples in the DSP/BIOS RTA SDK installation are as follows:

❑ **rtatrace.** The DSP/BIOS RTA SDK installation contains a pre-built "rtatrace" application. If you make any changes at the source code level, you need to rebuild "rtatrace" as follows. After the build has completed successfully, the "rtatrace" application is located in the same directory.

```
cd <RTASDK_dir>/ti/bios/rta/examples/gpp/rtatrace
gmake clean
gmake all
```

❑ **loadandrun.** The DSP/BIOS RTA SDK installation has a pre-built "loadandrun" application. It is built with the PROC+CHNL+MSGQ DSP/BIOS Link configuration. If you rebuild DSP/BIOS Link, you need to rebuild "loadandrun" as well. After the build has completed successfully, the "loadandrun" application is located in the same directory.

```
cd <RTASDK_dir>/ti/bios/rta/examples/gpp/loadandrun
gmake clean
gmake all
```

❑ **chnldrv.** The DSP/BIOS RTA SDK installation has a pre-built "libcd_bl.so" application. It is built with the PROC+CHNL+MSGQ DSP/BIOS Link configuration. If you rebuild DSP/BIOS Link, you need to rebuild "libcd_bl.so" as well. After the build has completed successfully, the "libcd_bl.so" application is located in the same directory.

```
cd <RTASDK_dir>/ti/bios/rta/sdk/chnldrv
gmake clean
gmake all
```

❑ **pmrdrv.** The postmortem driver is a loadable kernel module, hence; if you rebuild the Linux Kernel, you need to rebuild the driver to eliminate any kernel version mismatch warnings from Linux. After the build has completed successfully, the "pmr_drv.o" file is located in the "release" and "debug" directories.

```
cd <RTASDK_dir>/ti/bios/rta/sdk/pmrdrv
gmake clean
gmake all
```

❑ **hello.** The DSP/BIOS RTA SDK installation has a pre-built "hello" application. After the build has completed successfully, the "hello.out" file is located in the same directory.

```
cd <RTASDK_dir>/ti/bios/rta/examples/dsp/basic/hello/osk5912
gmake clean
gmake
```

2.3 Preparing to Run the Example Applications

You must have root privileges in order to load the drivers and DSP programs and to run the rtaTrace application. Then, copy the following files to a file system that the GPP has access to, that is, an NFS-mounted drive on another LINUX machine.

❑ Copy the channel driver (libcd_bl.so) from <RTASDK_dir>/ti/bios/rta/sdk/lib to /usr/lib. Or this library can be located anywhere and LD_LIBRARY_PATH can be set to point to it. For example:

```
export LD_LIBRARY_PATH=/home/sdk
```

❑ Copy the postmortem driver (pmr_drv.o) from the <RTASDK_dir>/ti/bios/rta/sdk/lib directory.

  If you have the MontaVista Linux PreView Kit 3.1, you need to load the postmortem driver by executing the steps as explained in Section 5.1. If you have MontaVista Linux Pro 3.1, the "pmrdrvload.bash" script allows you to load the postmortem driver. The script and driver must be in same directory.

❑ Copy the trace application (rtaTrace) from the <RTASDK_dir>/ti/bios/rta/examples/gpp/rtatrace directory.

❑ Copy the load-and-run application (loadandrun) from the <RTASDK_dir>/ti/bios/rta/examples/gpp/loadandrun directory.

❑ Copy the DSP application.

2.4 Configuring DSP/BIOS Examples for RTA SDK

The examples installed as part of DSP/BIOS are configured to use the JTAG/RTDX infrastructure to transfer RTA data. To reconfigure any of the examples to use RTA SDK capabilities, perform the following steps. In these steps, the "latency" application is used as an example.

1) Create a new directory called “osk5912” under the <SABIOS_dir>/ti/bios/examples/advanced/latency directory.

2) Copy the following files from the omap1510 platform latency example to the new osk5912 directory you created.

   - latency.tcf
   - latency_omap1510_custom.tci
   - makefile

3) Change the name of the latency_omap1510_custom.tci file to latency_osk5912_custom.tci.

4) Copy the following files from the <RTASDK_dir>/src/ti/bios/rta/examples/dsp/common directory to the <SABIOS_dir>/src/ti/bios/examples/common directory.

   - osk5912_common.tci
   - dsplink-omap-base.tci
   - bioslink_common.tci

5) Open the <SABIOS_dir>/ti/bios/examples/common/config.mak file, and add the following definitions:

   - LNKBIOSLINKLIB_DIR=<RTASDK_dir>/ti/bios/rta/rtalnk/lib
   - BIOSLINK_DIR=<the directory that dsplink.l55l resides>

6) Open the makefile file that you copied to the <SABIOS_dir>/ti/bios/examples/advanced/latency/osk5912 directory, and make the following modifications:

   - Add the "-ml" flag to CC55FLAGS and AS55FLAGS
   - Add the "-l$(BIOSLINK_DIR)/dsplink.l55l" flag to LD55FLAGS
   - Add the following line to the file:

     ```
     LD55OPTS += -i$(LNKBIOSLINKLIB_DIR)
     ```

7) Open the latency.tcf file that you copied to the <SABIOS_dir>/ti/bios/examples/advanced/latency/osk5912 directory, and make the following modifications:

   - Change "omap1510_common.tci" to "osk5912_common.tci".
   - Change "latency_omap1510_custom.tci" to "latency_osk5912_custom.tci".
   - Add the following lines after the line that imports the latency.tci file:

     ```
     utils.importFile("dsplink-omap-base.tci");
     utils.importFile("bioslink_common.tci");
     ```

Chapter 3

Sample Applications

This chapter describes the sample applications provided with the DSP/BIOS RTA SDK.

Topic **Page**

3.1 RTA Trace Application

The rtaTrace sample application runs from the Linux/ARM command line. It obtains RTA data from a DSP/BIOS program running on the DSP by using DSP/BIOS Link as a transport. It can obtain LOG and STS data and either print it to standard output or send it to a log file. This application can also be used to obtain the final LOG and STS records from a DSP program crash, that is, to perform a postmortem analysis.

The source code and makefiles are provided so that you can modify it to fit your needs.

3.1.1 Command Line

Syntax: rtaTrace [options] executable

Options:

-a<msec>	set CPU load averaging period in msec, default is 1000
-c	print postmortem data after crash and then exit
-d	dump log names and ids, then exit
-h	print this list of options
-l<mask>	log id mask (hex), default is all but system log
-n<prior>	nice priority level
-o	print to standard out
-p<msec>	polling rate in msecs (rounded to secs), default 1000
-r	reset statistics
-s	print statistics
-u	print CPU busy utilization
-x	do not print to syslog

3.1.2 Description

If you want to send LOG and STS data to the syslog, you must configure syslog. Edit the "/etc/syslog/conf" file to include an entry for "local0" and then restart the syslog. See the syslog man page for details.

If you have the MontaVista Linux Preview Kit 3.1, all data is sent to the /var/log/messages file, which is mapped to the local file system on the local board memory. Note that this /tmp file system can run out of space quickly.

The typical sequence for running DSP programs and obtaining RTA data is as follows:

1) Load the DSP/BIOS Link driver. You must do this before RTA data is gathered, but you only need to do it once after the system has been rebooted. If you have MontaVista Linux Pro 3.1, you can use the dsplink.bash script. If you are using the MontaVista Linux PreView Kit 3.1, you need to run the following shell commands manually:

```
[root >] insmod -f dsplinkk.o
[root >] mknod /dev/dsplink c 230 0
```

2) Load the RTA SDK postmortem driver. You must do this before RTA data is gathered, but you only need to do it once after rebooting the system. If you have MontaVista Linux Pro 3.1, you can use pmrdrvload.bash script. If you are using MontaVista Linux PreView Kit 3.1, you need to run the following shell commands manually:

```
[root >] insmod -f pmr_drv.o
 /* The postmortem driver major number is assigned
    dynamically. Check "bios_pmr" in the
    following list */
[root >] cat /proc/devices
[root >] mknod /dev/bios_pmr c MAJOR 0
```

3) Start the DSP (and any GPP) programs. RF6 requires additional setup steps, so see the RF6 documentation if you plan to use it. For example, you might use a command like one of the following:

```
loadandrun hello.out &
rf6_gpp app.out
```

4) Run the rtaTrace application with the desired options. By default, log data is sent to syslog. For example:

```
rtaTrace hello.out
```

You can run rtaTrace at any time after starting the DSP program. You can stop rtaTrace with a kill -2 and restart it any number of times. If you stop rtaTrace any other way (for example, with CTRL+C), it cannot be restarted unless the DSP program is reloaded.

If the DSP program crashes and rtaTrace does not detected it, then a kill -3 "<rtaTrace process ID>" causes rtaTrace to perform a postmortem and stop. If rtaTrace was not running when the DSP program crashed, rtaTrace it can be started in postmortem mode with the -c option.

The rtaTrace program can be restarted and directed to obtain different logs. For example, with “rtaTrace -l0x1 hello.out" to obtain the system log.

If you do not start rtaTrace immediately after running the DSP program, then you should use the -r option to clear any overrun statistics if you are collecting STS data.

3.2 Load and Run Application

The RTA SDK also provides the loadandrun utility. This utility loads a DSP program using DSP/BIOS Link, and runs it with an optional set of parameters that can be sent to the DSP application.

3.2.1 Command Line

Syntax: loadandrun <coffFile> [<dsp args...>]

3.2.2 Description

Loadandrun does not exit after starting the DSP program; instead it waits for a signal to cleanup and shutdown gracefully.

To stop the DSP program, send a CTRL+C (or kill -2) to loadandrun. This will also shut down DSP/BIOS Link cleanly so that another DSP program can be loaded. Otherwise the OSK5912 might have to be rebooted.

In addition to retrieving LOG/STS data at run-time, the DSP/BIOS RTA SDK supports postmortem LOG/STS analysis. To use this feature, DSP/BIOS Link must continue running on the GPP side. Hence, if you decide to switch to postmortem analysis, do not terminate the "loadandrun" application. The postmortem analysis feature is only supported if the LOG/STS objects are located in the DSP's internal memory (DARAM/SARAM).

3.3 Questions and Answers

Q) How big is the rtaTrace application?

A) The rtaTrace application normally takes about 1.5 M bytes of virtual memory depending up the size and number of LOG and STS objects.

Q) How much CPU time does the rtaTrace application use?

A) Running the rtaTrace application normally takes about 4% of the CPU on the ARM. This can vary greatly depending upon the polling rate and the amount of LOG and STS data that is being obtained.

Q) Why does the STS data not look correct when I start up the rtaTrace application?

A) Data accumulated on the DSP for STS objects can overflow if not uploaded on a periodic basis. If the rtaTrace application has not been started immediately after the DSP program was loaded then start it with the "-r" flag which will reset the accumulators on the GPP side.

Q) Why does RTA data stop coming out after I run the DSP program for a while?

A) The DSP program might have crashed. You should perform
"kill -3 <rtaTrace process id>" to obtain any postmortem data.

Q) Why do I hear extra noise from the speakers while running rf6 every time the rtaTrace program gets some data?

A) The rtaTrace needs to be run at a lower priority; use the -n option.

Q) Why does the postmortem analysis not display any data?

A) The postmortem analysis is only supported for the DSP/BIOS LOG buffers that are located on DSP's internal memory.

Q) How does postmortem analysis work if the DSP is in sleep mode?

A) If the DSP is in sleep mode, the read attempt by GPP to DSP's internal memory stalls the GPP.

Q) Why I am getting an "RTA_readProgramError=0xa0000034" while starting "rtaTrace" program?

A) This error is caused because DSP/BIOS Link is no longer up and running, i.e. "loadandrun" application has been terminated.

Index