Application Report
SPRA675 - June 2000

Upgrading Applications to DSP/BIOS II

Stephen Lau *Digital Signal Processing Solutions*

ABSTRACT

DSP/BIOS II adds numerous functions to the DSP/BIOS kernel, while maintaining 100% Application Programming Interface (API) compatibility with DSP/BIOS I. In addition, new functionality has been added to Code Composer Studio™. This application note describes the transition from DSP/BIOS I to DSP/BIOS II and focuses on the enhancements and changes to existing DSP/BIOS I functions.

Contents

List of Figures

List of Tables

1 Code Composer Studio and DSP/BIOS Versions

The version of Code Composer Studio can be found by going to the “Help” menu and selecting “About Code Composer Studio”.

This application note focuses on the migration from BIOS I and BIOS II as implemented in the products listed in Table 1. Migration from Alpha and Beta products is not covered in this document.

Table 1. DSP/BIOS Versions

Code Composer Studio Version	DSP/BIOS Version
Code Composer Studio C6000 v1.00	BIOS I
Code Composer Studio C6000 v1.05 (6211 DSK)	BIOS I
Code Composer Studio C5000 v1.10	BIOS I
Code Composer Studio C5000 v1.11 (5402 DSK)	BIOS I
Code Composer Studio C6000 v1.20	BIOS II
Code Composer Studio C5000 v1.20	BIOS II

When installing Code Composer Studio C6000™ or C5000™ v1.20 on a computer where a previous version exists, it is recommended that the previous version be uninstalled first. If you wish to have Code Composer Studio C6000 and C5000 v1.2, it is highly recommended that the C6000 version be installed before the C5000 version.

If your code build environment utilizes the UNIX code generation tools, you will need to transfer the new DSP/BIOS II libraries to your UNIX environment. Refer to SPRA660, *Building DSP/BIOS Programs in UNIX* for additional details.

C6000 and C5000 are trademarks of Texas Instruments.

2 DSP/BIOS I and DSP/BIOS II

DSP/BIOS II introduces several new modules and significantly increases the functionality of several existing modules. DSP/BIOS services can be divided into three main categories: Memory Management, Event Handling, and Stream I/O.

Refer to SPRA646, *DSP/BIOS II Technical Overview* for additional details. The functionality of Code Composer Studio has also been extended through the addition of a new plug-in. This application note will focus on the changes to DSP/BIOS I modules.

2.1 New Modules

The following modules were added to DSP/BIOS II:

- ATM: Atomic functions
- C54 or C62: Target-specific functions
- SYS: System services manager
- QUE: Queue manager
- LCK: Resource lock manager
- SEM: Semaphore manager
- MBX: Mailbox manager
- DEV and Dxx: Device driver interface
- SIO: Streaming I/O manager
- TSK: Task manager

The new ATM module adds atomic functions for performing some basic operations on variables without allowing interrupts during execution of the function. The new C54 or C62 module (depending on your DSP family) provides functions for making dynamic changes to the hardware interrupts. The new QUE module manipulates a queue data structure. The new SYS module adds functions for handling formatted data output and program termination conditions.

DSP/BIOS's scheduling capabilities have been increased through the addition of task threads. Tasks have priority levels lower than those of software interrupts (SWIs) and higher than the idle thread. In contrast to SWI's, tasks are capable of blocking and feature independent stacks. Tasks can be suspended and it will wait until some other thread readies the task. You can use the LCK, MBX, and SEM modules to create objects used to manage task synchronization and data sharing. The MBX manager utilizes queues and semaphores and independent from the mailbox values used by SWIs.

DSP/BIOS now includes a streaming I/O model. This model provides a more structured approach to device-driver creation. It provides blocking functions to access the stream and a set of built-in device drivers. Refer to the *DSP/BIOS User's Guide* for a comparison of the pipes and stream I/O models.

2.2 Increased Functionality in Existing Modules

The MEM and SWI modules provide several new functions. The new MEM functions allow dynamic allocation of memory into distinct memory segments. New SWI functions allow dynamic creation, deletion, and modification of software interrupt objects. Many of the new modules in DSP/BIOS allow for dynamic creation, deletion, and modification of objects during run-time.

2.3 New Kernel/Object View Plug-in

A new plug-in, called the “Kernel/Object View” has been added to Code Composer Studio. This plug-in has six tabbed pages that allow you to view system-wide information (KNL), tasks (TSK), mailbox objects (MBX), semaphore objects (SEM), memory segments (MEM), and software interrupts (SWI).

2.4 New Examples and Documentation

Several new examples have been added to the “c:\ti\cXXXX\examples\bios” folder. Many of these examples are designed to utilize the new DSP/BIOS modules. The “DSP/BIOS User’s Guide” and the DSP/BIOS online help have been updated to include documentation for the new modules and other new features.

3 Upgrading from DSP/BIOS I

The new DSP/BIOS II is 100% compatible with the existing DSP/BIOS I Application Programming Interface (henceforth API). Existing code that uses the DSP/BIOS I API is 100% compatible with the new DSP/BIOS II API. The Configuration Tool will attempt to convert your configuration file and will add support for the new modules.

NOTE: This is a one-way conversion and care should be taken to backup the existing application project before the conversion. During the update process, the Configuration Tool will automatically make a backup and append a unique character string to the existing file name.

3.1 Converting Configuration Files

Follow these steps to convert your configuration file. The conversion adds new modules and objects to the configuration.

Step 1: Open your Configuration Database File (CDB) by going to the menu and choosing “File−>Open” or by double clicking on the CDB file it in the project view.

Step 2: Code Composer Studio will give you a message that the configuration file is out of date as seen in Figure 1.

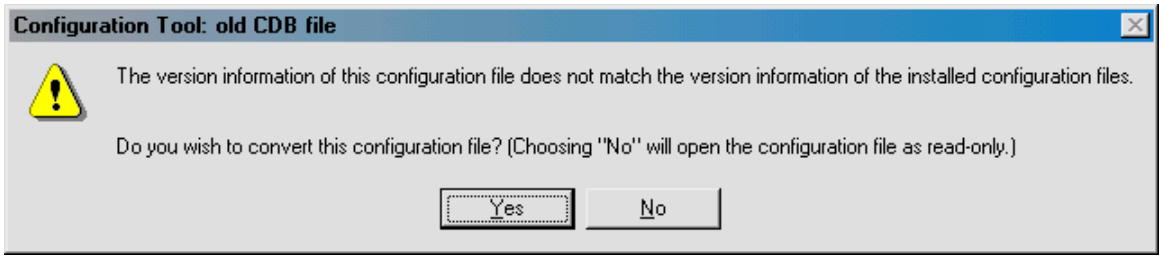

Figure 1. Configuration Tool Conversion Message

Step 3: Click Yes to convert the configuration file. Code Composer will give a message indicating that it has created a backup of the existing CDB file.

Step 4: Save the configuration file so that the header and linker command files are re-created.

NOTE: Because DSP/BIOS II more than doubles the amount of functionality available in DSP/BIOS I the conversion process may not be perfect. If possible, it is suggested that the most reliable method is to start a new DSP/BIOS II configuration file and manually configuring with the existing parameters.

3.2 Reviewing Existing Configuration Files

If you wish to review your existing CDB file as it was with DSP/BIOS I, you can choose to open it up in “Read Only Mode”.

Step 1: Open the DSP/BIOS I CDB file.

Step 2: Code Composer will give you a message that the configuration file is out of date, as seen in Figure 1.

Step 3: Choose “No” to open the CDB file as “read-only.”

Step 4: Code Composer Studio will inform you that the file is out of date as can be seen in Figure 2.

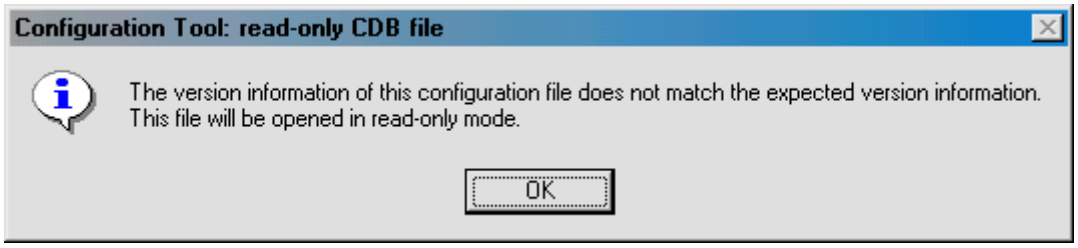

Figure 2. Configuration Tool Message

Step 5: The Configuration Tool will now show your CDB file as it existed under DSP/BIOS I.

3.3 Rebuilding Programs

After the conversion process, rebuild the program using “Rebuild All” rather than “Incremental Build”. This will ensure that the new changes have been compiled.

NOTE: Numerous linking errors will occur if the Configuration Database File (CDB) is not upgraded since it will contain references to DSP/BIOS constructs that may have changed. It is also necessary to ensure that the Configuration Tool generates new versions of the associated header and assembly files.

4 Changes to Existing Modules

4.1 Global Settings (GBL Manager)

During a typical boot procedure the C environment is initialized, followed by a routine that configures DSP/BIOS and initializes the DSP/BIOS objects. Afterwards, “main()” executes and exits to the DSP/BIOS scheduler. DSP/BIOS II allows the user to specify a initialization function, as can be seen in Figure 3, which will run before the “main()” routine and at the beginning of the DSP/BIOS module initialization phase. This function can perform special hardware setup or any other function. Since this function executes before DSP/BIOS initializes, the code in this function should not use any DSP/BIOS API calls.

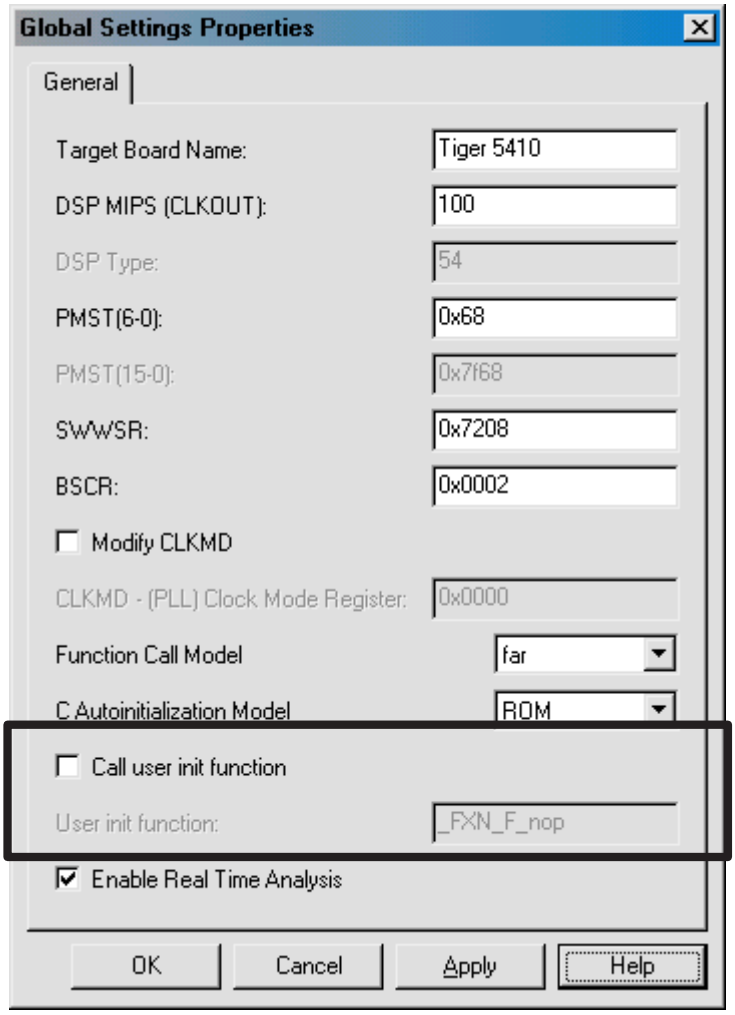

Figure 3. Global (GBL) Settings

4.2 Idle Function Manager (IDL Manager)

One idle (henceforth IDL) function has been added, “RTDX_DataPump.” This function calls “RTDX_Poll” which performs the DSP side polling of the RTDX channel and is only required on the 54x. If Real Time Analysis is disabled, then this IDL function is not automatically removed from the IDL loop, but IDL objects “LNK_dataPump”, “IDL_cpuLoad”, and “RTA_dispatcher” will be removed. This allows RTDX to be used without the DSP/BIOS real time analysis features. If desired, “RTDX_DataPump” can be removed afterwards by changing the Host (HST) Link Type to “None” and disabling RTDX in the RTDX module.

If “Auto calculate idle loop instruction count” is selected then DSP/BIOS runs one pass through the IDL functions at system startup to get an approximate value for the idle loop instruction count.

NOTE: If “Auto calculate idle loop instruction count” is checked, it is important to make sure that the IDL functions do not block on this first pass, otherwise the program will never get to main. User defined IDL functions can be excluded from the IDL loop count by un-checking the “Include in CPU Load calibration” property as seen in Figure 4.

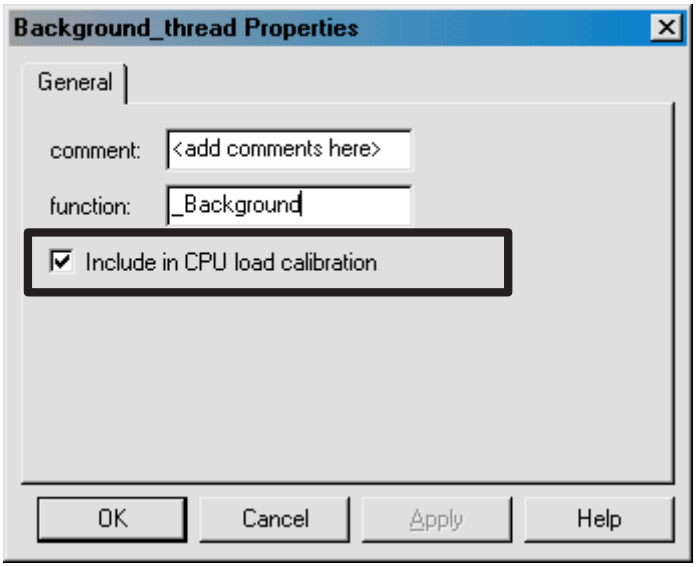

Figure 4. Idle (IDL) Object Properties

4.3 Memory Manager (MEM Manager)

The MEM module has been significantly enhanced. Fine grained control over memory sections have been added to the Configuration tool. On the DSP, new MEM API calls allow for DSP specific dynamic allocation.

NOTE: Note for TMS320C54x™ users: DSP/BIOS for the ’C54x was originally developed for the 16-bit addressing model of the early ’C54x devices. Program accesses in this ‘near’ model were limited to a single 64K word program page. Newer ’C54x devices incorporate ‘far’ extended addressing modes, and DSP/BIOS and DSP/BIOS II have been modified to work properly in this environment. Refer to SPRA599, *DSP/BIOS and TMS320C54x Extended Addressing* which describes how to configure DSP/BIOS for use in extended addressing applications. DSP/BIOS II extends the functionality in DSP/BIOS I and the procedures described in SPRA599 must be utilized, especially for DSP/BIOS I modules. The new DSP/BIOS II modules are not subject to same limitations.

4.3.1 Configuration Tool

4.3.1.1 Added Features

The Configuration Tool for DSP/BIOS II allows for much more control over the placement of defined sections. In addition, the Configuration tool can be programmed to only define sections for DSP/BIOS, simplifying integration with existing linker command files.

TMS320C54x is a trademark of Texas Instruments.

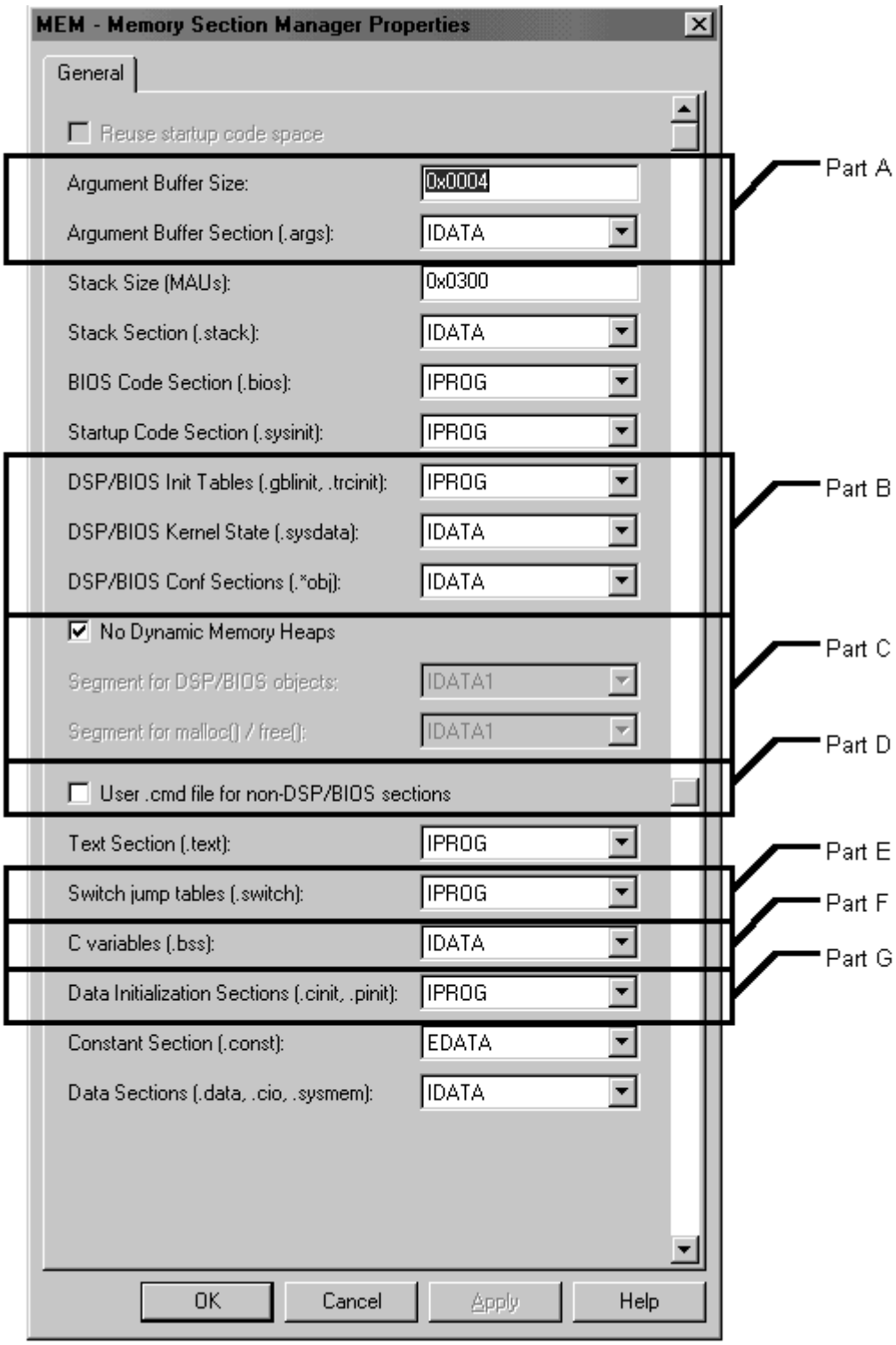

Figure 5. Memory Manager Properties

NOTE: "Reuse startup space" is only selectable if the ".sysinit" section is not a code section and has a heap defined.

Argument Buffer Size

The size of the ".args" section. The ".args" section contains the "argc", "argv", and "envp" arguments to the program's main function. Code Composer loads arguments for the main function into the ".args" section. The ".args" section is parsed by the boot file. Refer to Figure 5, Part A.

Argument Buffer Section (.args)

The memory section containing the ".args" section. Refer to Figure 5, Part A.

DSP/BIOS Init Tables (.gblinit, .trcinit)

The memory section containing the DSP/BIOS global and instrumentation initialization tables. Refer to Figure 5, Part B.

DSP/BIOS Kernel State (.sysdata)

The memory section containing system data about the DSP/BIOS kernel state. Refer to Figure 5, Part B.

DSP/BIOS Conf Sections (.obj)

The memory section containing configuration properties that can be read by the target program. Refer to Figure 5, Part B.

No Dynamic Memory Heaps

Put a checkmark in this box to completely disable the ability to dynamically allocate memory and the ability to dynamically create and delete objects. If this box is checked, your program may not call the MEM_alloc, MEM_valloc, and MEM_calloc functions or the XXX_create function for any DSP/BIOS module. If this box is checked and the program calls one of these functions, an error occurs when the program is linked. Refer to Figure 5, Part C.

Segment for DSP/BIOS objects

The default memory section that will contain objects created at run-time with an XXX_create function. Refer to Figure 5, Part C.

Segment for malloc / free

The memory section from which space is allocated when a program calls malloc and from which space is freed when a program calls free. Refer to Figure 5, Part C.

User .cmd file for non-DSP/BIOS sections

Put a checkmark in this box (Figure 5, Part D) if you want to have full control over the memory used for the sections that follow. You will need to create your own linker command file that begins by including the linker command file created by the Configuration Tool. Your linker command file should then assign memory for the items normally handled by the following properties. See the "Optimizing C Compiler User's Guide" for more details.

For example, if the name of my Configuration Database File (CDB) was "myprog.cdb", then the Configuration Tool would generate three files, "myprogcfg.cmd", "myprogcfg.sxx", and "myprogcfg.hxx". If "User .cmd file for non-DSP/BIOS sections" was checked (selected), then "myprogcfg.cmd" will not include any definitions for ".text", ".switch", ".bss", ".cinit", ".pinit", ".const", ".data", ".cio", and ".sysmem". These sections, any any other user defined ones would have to be specified in the user's linker command file. As seen in Figure 6, the user's linker command file would include a link to the Configuration Tool's linker command file, "myprogcfg.cmd" so that the DSP/BIOS objects can be properly resolved.

```
-l myprogcfg.cmd
SECTIONS
{
            my_code         :>      IPRAM
            .text           :>      IPRAM
            .bss            :>      IDRAM
}
```

Figure 6. Example User Defined Linker CommandFfile

Switch jump tables (.switch)

The memory section containing the switch statement tables. This section may be located in ROM or RAM. Refer to Figure 5, Part E.

4.3.1.2 Changed Features

C variables (.bss)

The memory section containing global and static C variables. At boot or load time, the data in the ".cinit" section is copied to this section. This section should be located in RAM. Refer to Figure 5, Part F. The Configuration Tool in DSP/BIOS II only defines the ".far" section when the configuring DSP/BIOS II for the c6000.

Data Initialization Section (.cinit, .pinit).

The memory section containing tables for explicitly initialized global and static variables and constants. This section may be located in ROM or RAM. Refer to Figure 5, Part G. The ".pinit" section is no longer being defined by the Configuration Tool in DSP/BIOS II.

Memory Section Properties

Each memory section can now host a heap from which the MEM functions utilize. Refer to Figure 7.

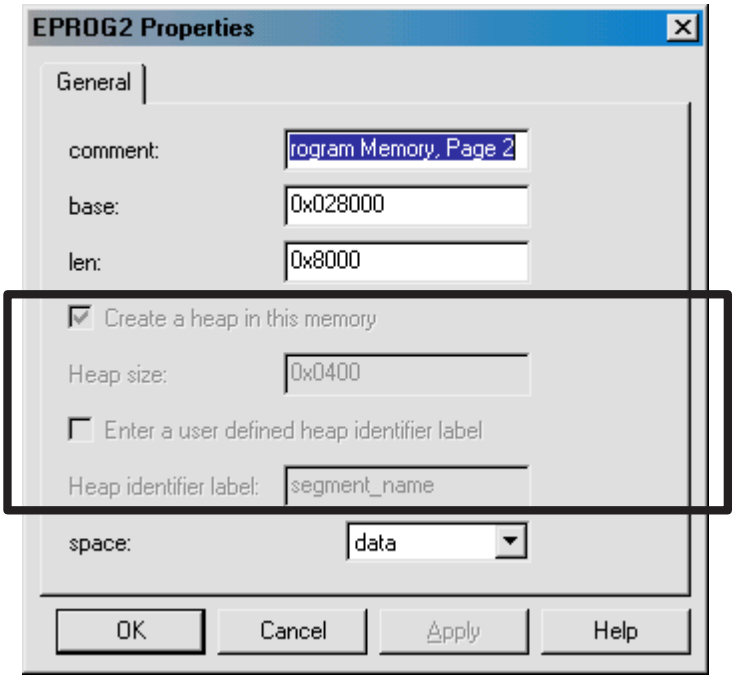

Figure 7. Memory Properties

NOTE: Migration of user defined sections that are originally assigned as “Code Section” may be incorrectly converted to “Data Section” and vice versa. Make sure you manually check that each memory section is defined in the correct space.

Create a heap in this memory

If checked, a heap is created in this memory section. If dynamic memory allocation is disabled in the Memory Manager, then this option is not available.

NOTE: The Configuration Tool may automatically enable the ”Dynamic Heaps” in pre-existing and newly created memory sections. To ensure that they are disabled, uncheck “No Dynamic Memory Heaps” as seen in Figure 5, Part C, and uncheck “Create a heap in this memory” as seen in Figure 7. Then, if no heaps are desired, check “No Dynamic Memory Heaps” as seen in Figure 5, Part C.

Heap size

The size of the heap to be created in this memory section.

User defined name

A user defined identifier label can be specified to make the use of the heap easier. For example, this label would typically be passed to DSP/BIOS as the segId during a call to MEM_alloc.

4.3.2 MEM API

The MEM module API provides functions that can allocate from one or more sections of memory as defined in the Configuration Tool.

New MEM API functions:

- MEM_alloc: Allocate from a memory section
- MEM_define: Define a new memory section
- MEM_calloc: Allocate and initialize to 0
- MEM_free: Free a block of memory
- MEM_redefine: Redefine an existing memory section
- MEM_stat: Return the status of a memory section
- MEM_valloc: Allocate and initialize to a value

MEM always allocates an even number of words and always aligns buffers on an even boundary. Code size can be reduced by disabling dynamic memory allocation and by statically defining the system. Disabling dynamic memory is accomplished by selecting “No Dynamic Memory Heaps” box in the Properties dialog for the MEM manager as shown in Figure 5, Part C. The standard C “malloc,” “free,” and “calloc” functions actually use the MEM functions to allocate and free memory. Refer to the *DSP/BIOS User’s Guide* or online help for additional information.

4.4 Pipe Module (PIP Module)

No changes were made to the Configuration Tool that configures pipes. Two new PIP API calls were added:

- PIP_peek. Get the pipe frame size and address without actually claiming the pipe frame.
- PIP_reset: Reset all fields of a pipe object to their original values.

4.5 RTDX Manager

One additional field has been added to the RTDX Manager properties, the “RTDX Interrupt Mask” as seen in Figure 8. This mask identifies RTDX clients and protects critical RTDX sections. The mask specifies the interrupts to be temporarily disabled inside RTDX critical sections. This also temporarily disables other RTDX clients and prevents another RTDX function call. Refer to the “RTDX online help” and the *DSP/BIOS User’s Guide* for more details.

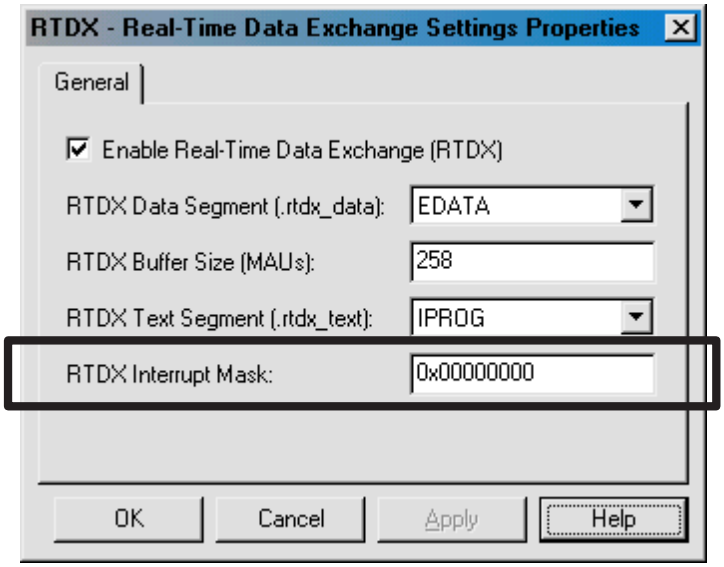

Figure 8. RTDX Manager Properties

4.6 Software Interrupts Manager (SWI Manager)

DSP/BIOS II extends the scheduling model of DSP/BIOS I by adding tasks. All tasks in DSP/BIOS II are scheduled out of one SWI which has a priority level of zero. Thus in DSP/BIOS II, there are only fourteen SWI priority levels if tasks are enabled. If tasks are disabled, fifteen priority levels will be available. If tasks capability is required, it is necessary to clear up SWI priority level zero by promoting the priority on all SWI's to a higher priority level.

The SWI API has also been extended to allow for dynamic SWI creation, deletion and modification. Refer to *DSP/BIOS User's Guide* or online help for additional information. New SWI API functions:

- SWI_create: Create a software interrupt
- SWI_delete: Delete a software interrupt
- SWI_getattrs: Get attributes of a software interrupt
- SWI_setattrs: Set attributes of a software interrupt

5 New Modules

5.1 System Module (SYS Manager)

The SYS module is a set of functions that provide basic system services, such as halting program execution and printing formatted text. Each SYS function is similar to functions normally found in the standard C library.

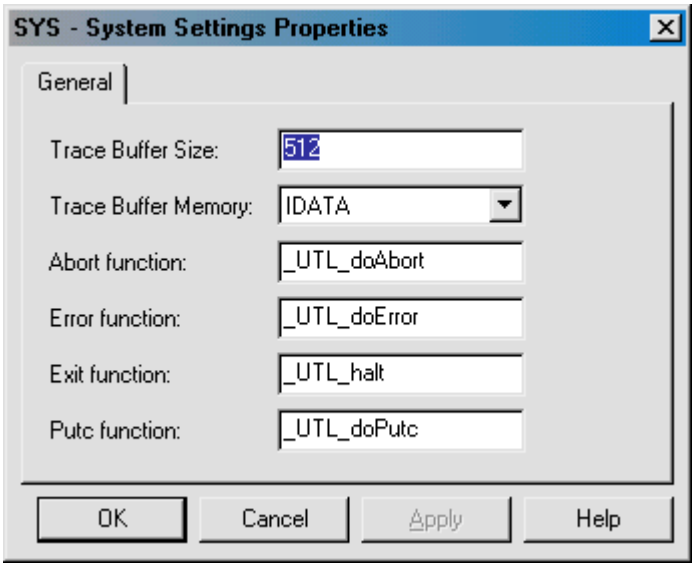

Figure 9. Figure 1 System Settings (SYS Manager)

As seen in Figure 9:

- Abort function: The function to run if the application aborts by calling “SYS_abort.” The default function is “UTL_doAbort,” which logs an error message and calls _halt.
- Error function: The function to run if an error flagged by “SYS_error” occurs. The default function is “UTL_doError,” which logs an error message.
- Exit function: The function to run when the application exits by calling SYS_exit. The default function is “UTL_halt,” which loops forever with interrupts disabled, preventing other processing.
- Putc function: The function to run if the application calls “SYS_putchar,” “SYS_printf,” or “SYS_vprintf.” The default function is “UTL_doPutc,” which writes a character to the trace buffer.

NOTE: During the conversion from DSP/BIOS I to DSP/BIOS II, the Configuration Tool may convert some System function default values to “FXN_F_nop.” Though harmless, “FXN_F_nop” may not yield the intended result.

5.2 Kernel Object View Plug-In

The Kernel/Object View plug-in, as seen in Figure 10 allows a view into the current configuration, state, and status of the DSP/BIOS objects currently running on the target. This tool shows both the dynamic and statically configured objects that exist on the target. To display the Kernel/Object View, choose the “Tools−>DSP/BIOS Kernel/Object View” menu item in Code Composer Studio.

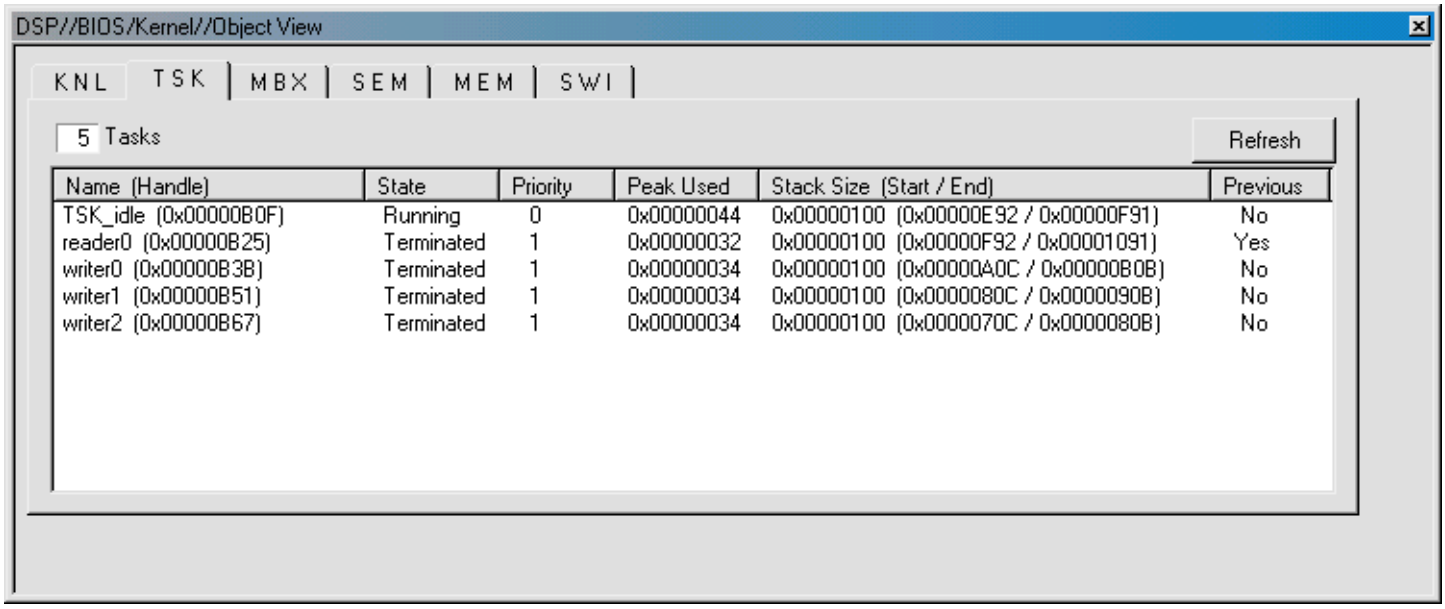

Figure 10. Kernel Object View Plug-In

This plug-in has following selection tabs:

- KNL: System-wide information.
- TSK: Task threads and state.
- MBX: Mailbox objects.
- SEM: Semaphore objects.
- MEM: Memory segments.
- SWI: Software interrupt threads.

NOTE: The Kernel Object view differs from the execution graph because it can show dynamically created objects. The Kernel/Object View may display names that are labels for other items on the target because some labels share memory locations. In this case you may see a name that does not match the configuration. If a label is not available for a memory location, a name is automatically generated and is indicated with angle brackets (e.g., <task1>).

6 Minimizing Code Size

If you do not plan to use task threads or dynamic memory allocation, you can disable support for these features in the TSK and MEM manager property windows. This helps to minimize the size of your program since DSP/BIOS II does not need to include the task module or the memory module. Functions that dynamically create (XXX_create) objects affect code size only if a program calls these functions. For example, if you are concerned about code size, create your SWI or TSK objects using the Configuration Tool, rather than calling SWI_create or TSK_create.

To disable the task manager, right-click on the TSK − Task Manager and choose Properties from the pop-up menu. In the Properties window, remove the checkmark from the “Enable TSK Manager” box as seen in Figure 11.

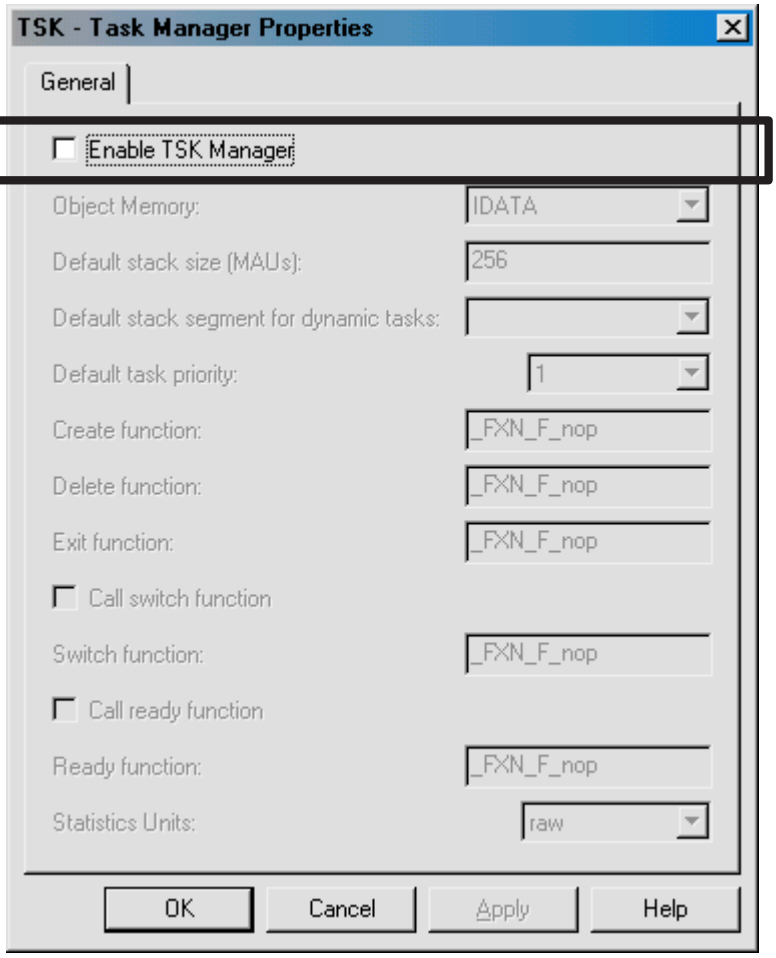

Figure 11. Task Manager Properties

To disable dynamic memory allocation, right-click on the MEM − Memory Section Manager and choose Properties from the pop-up menu. In the Properties window, put a checkmark in the No Dynamic Memory Heaps box and click OK. This is shown in , Part C.

For additional information, refer to the application note, *Guidelines for Optimizing DSP/BIOS Applications in Target Memory* and *DSP/BIOS II Sizing Guidelines for the TMS320C54x/62x*.

7 Acknowledgements

Special thanks to Yvonne DeGraw, Jack Greenbaum, Liz Keate, Robert Tivy, and Karl Wechsler.

8 References

1. *TMS320C5400 DSP/BIOS User's Guide*, SPRU326
2. *TMS320C6000 DSP/BIOS User's Guide*, SPRU303
3. DSP/BIOS online help
4. *Building DSP/BIOS Programs in UNIX*, SPRA660
5. *DSP/BIOS and TMS320C54x Extended Addressing*, SPRA599
6. Guidelines for OptimizingDSP/BIOS Applications in Target Memory
7. *DSP/BIOS II Sizing Guidelines for the TMS320C62x*, SPRA667
8. *DSP/BIOS II Sizing Guidelines for the TMS320C54x*, SPRA663

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.

Following are URLs where you can obtain information on other Texas Instruments products and application solutions:

Products		**Applications**	
Amplifiers	amplifier.ti.com	Audio	www.ti.com/audio
Data Converters	dataconverter.ti.com	Automotive	www.ti.com/automotive
DSP	dsp.ti.com	Broadband	www.ti.com/broadband
Interface	interface.ti.com	Digital Control	www.ti.com/digitalcontrol
Logic	logic.ti.com	Military	www.ti.com/military
Power Mgmt	power.ti.com	Optical Networking	www.ti.com/opticalnetwork
Microcontrollers	microcontroller.ti.com	Security	www.ti.com/security
		Telephony	www.ti.com/telephony
		Video & Imaging	www.ti.com/video
		Wireless	www.ti.com/wireless

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