Application Report

SPRAA82−March 2005

OMAP591x: Tuning the System Memory Requirements of DSP/BIOS Link

Jon Hunter

Software Development System

ABSTRACT

The DSP/BIOS™ Link software has been designed to implement inter-processor communication between a host device and a DSP. For the case of OMAP591x devices, the default configuration of the DSP/BIOS Link software requires that 2MB of system memory is reserved for exclusive use by DSP/BIOS Link. The intention was to provide developers using DSP/BIOS Link plenty of memory to begin with and enable them to tune the system memory requirements per their specific needs. This application report discusses how to tune the system memory reserved for DSP/BIOS Link with regard to the needs of the application.

Contents

List of Figures

Trademarks

DSP/BIOS, OMAP, TMS320C55x, Code Compose Studio, C55x are trademarks of Texas Instruments.

1 Introduction

The DSP/BIOS Link software is designed for inter-processor communication between a host device and a DSP. This software provides three main services between the host and client device. These are:

- Basic processor control (attach, load, start, stop, etc)
- Data streaming
- Messaging

For the case of OMAP™ devices, by default the DSP/BIOS Link software requires 2MB of system memory is reserved for exclusive use by DSP/BIOS Link. This memory is used for two purposes, which are:

- Client program and data space (1MB)
- Shared memory between host and client for data streaming and messaging (1MB)

The intention was to provide developers using DSP/BIOS Link plenty of memory to begin with and enable them to tune the system memory requirements per their specific needs. This document describes how to modify the system memory requirements used by DSP/BIOS Link in an OMAP application.

2 Modifying the System Memory Requirements

DSP/BIOS Link is available for both the OMAP5910 and OMAP5912 devices. Both are dual-core devices that consist of an ARM9 processor, often referred to as the host or general purpose processor (GPP) and a TMS320C55x™ DSP. The DSP/BIOS Link software is essentially two separate drivers; one that exists on the GPP and the other that exists on the DSP. In order to modify the system memory requirements, it is necessary to modify both the driver on the GPP-side and the driver on the DSP-side. The following sections describe the changes that need to be made to both the GPP and DSP drivers.

2.1 Configuring the GPP-Side

On the GPP-side, DSP/BIOS Link is configured using a text file. For DSP/BIOS Link release 1.1x, this text file is called CFG_OMAP.txt. The CFG_OMAP.txt is located in the following directory under the DSP/BIOS Link root directory:

```
<dsplink root>\config\all\
```

A complete description of the parameters in the CFG_OMAP.txt file can be found in the DSP/BIOS Link User Guide. To change the system memory requirements, there are two section types in this configuration file that need to be modified. These are the Link Table(s) denoted as “[LINKTABLEn]” and the MMU Table(s) denoted as “[MMUTABLEn]”. In addition to modifying the Link Table(s) and MMU Table(s), it is also necessary to modify the file shm.c. This file is located in the following directory:

```
<dsplink root>\gpp\src\ldrv\
```

Steps 4-6 in Section 2.3 illustrate how the above items are modified when reconfiguring the system memory requirements.

2.1.1 Understanding the Link Tables

The Link Table(s) specify the attributes of the driver that is used to exchange data between a GPP and a DSP. There is one Link Table defined for each DSP in the system that will use DSP/BIOS Link to communicate with the GPP. For OMAP devices, there is only one Link Table defined. These devices only have a single DSP that will communicate with the GPP. Other systems may have more than one DSP and some more than one Link Table will be defined. The default Link Table in the CFG_OMAP.txt file is defined as follows:

```
[LINKTABLE0]
[0]
NAME                | S |   SHARED MEMORY DRIVER
ABBR                | S |   SHM
NUMCHANNELS         | N |   16
BASECHANNELID       | N |   0
MAXBUFSIZE          | N |   16384
INTERFACE           | A |   SHM_Interface
ARGUMENT1           | H |   0x11F00000
ARGUMENT2           | H |   0x0
[/0]
[/LINKTABLE0]
```

The parameters in the Link Table(s) are defined as follows:

- The NAME field simply defines the name of the link driver. For OMAP devices, data is exchanged between the GPP and DSP via a region of shared memory. Hence, in the Link Tables shown above the name of the driver is “SHARED MEMORY DRIVER”.
- The ABBR field is simply an abbreviation of the name in the Link Table. This is not currently used but will be used in a future release.
- The NUMCHANNELS field defines the total number of channels that the link driver can support for transferring data between the GPP and DSP.
- The BASECHANNELID field defines a base ID for the channels that are created between the GPP and DSP. In the above link tables, a BASECHANNELID of 0 is defined and channel IDs available for use by the application would be 0-15.
- The MAXBUFSIZE field defines the maximum size of the buffer that can be exchanged between the GPP and DSP. In the default link tables above, the maximum buffer size that can be transfer between GPP and DSP is 16KB.
- The INTERFACE field defines a pointer to a function table for link driver that is used.
- The ARGUMENT1 field defines a parameter for the link driver. In the case of the shared memory driver for OMAP, this argument defines the physical address of the shared system memory used to exchange data between the GPP and DSP.
- The ARGUMENT2 field defines a parameter for the link driver. This parameter is unused in the case of the shared memory driver for OMAP.

2.1.1.1 Understanding the MMU Tables

The TMS320C55x™ DSP only has a 24-bit (or 16MB) address space. Hence for an OMAP device, it is only possible for the C55x DSP to access portions of the 32-bit (or 4GB) address space available to the GPP. In order to allow the C55x DSP to access particular ranges within the 32-bit address space for the GPP, the OMAP devices contain a DSP MMU that maps portions of the 32-bit address space into the DSP’s 24-bit address space. The GPP is responsible for configuring the DSP’s MMU and controls the external memory that is accessible by the DSP. The MMU tables specify which external locations are accessible by the DSP and the GPP uses this information to configure the DSP’s MMU. The default MMU Tables defined by DSP/BIOS Link are as follows:

```
[MMUTABLE0]
[0]
ADDRVIRTUAL     | H |   0x00400000
ADDRPHYSICAL    | H |   0x11F00000
SIZE            | H |   0
ACCESS          | H |   0x3
PRESERVE        | H |   0x1
MAPINGPP        | B |   FALSE
[/0]
[1]
ADDRVIRTUAL     | H |   0x00600000
ADDRPHYSICAL    | H |   0x11E00000
SIZE            | H |   0
ACCESS          | H |   0x3
PRESERVE        | H |   0x1
MAPINGPP        | B |   TRUE
[/1]
[/MMUTABLE0]
```

The MMUTABLE0 identifier indicates that these are the MMU Table entries for DSP0. In the above example, we can see that there are two MMU Table entries for DSP0. In the case of OMAP, there is only a single DSP and there is only one MMUTABLE. The parameters in the MMU Table(s) are defined as follows:

- The ADDRVIRTUAL field defines the base address in the DSP memory space where the region of external memory will be mapped.
- The ADDRPHYSICAL field defines the physical base address in the GPP memory space that will be mapped into the DSP address space.
- The SIZE field defines the size of the region to be mapped. The DSP MMU supports 1MB, 64KB, 4KB and 1KB pages and the SIZE field indicates the page size that will be mapped. For instance, 0 defines a page size of 1MB, 1 defines a page size of 64KB, 2 defines a page size of 4KB, and 3 defines a page size of 1KB. Hence, the above MMU entries each map 1MB of external memory into the DSP's address space.
- The ACCESS field defines the access permission that the DSP has to this region. The DSP MMU supports three access permissions which are no-access (ACCESS = 0), read-only (ACCESS = 2) and read/write (ACCESS = 3). Hence, both of the above MMU entries allow the DSP to have read/write access to the mapped regions of external memory.
- The PRESERVE field defines whether the DSP MMU entry will be preserved in the MMU TLB (translation look-aside buffer). If the MMU entry is not preserved and the MMU table walking logic is enable then the MMU entry could be over-written. If PRESERVE=1 then the MMU entry will be preserved and if PERSERVE=0 then the entry will not be preserved.
- The MAPINGPP defines whether this region should also be mapped by the GPP's MMU as well.

Note:
The ADDRVIRTUAL and ADDRPHYSICAL addresses are defined in bytes.

2.2 Configuring the DSP-Side

On the DSP-side, when modifying the system memory requirements, there are two main areas that need to be re-configured and these are:

- The DSP memory map
- The shared memory driver that exists on the DSP-side

Steps 7 and 8 in the following section describes how to modify these two items.

2.3 Procedure For Configuring Memory Requirements of DSP/BIOS Link

This section will illustrate how the system memory reserved for DSP/BIOS Link could be reduced from 2MB to 72KB. It could be possible to reduce the system memory reserved to less than 72KB. The user application will determine how much memory needs to be reserved. The following steps describe the procedure for modifying the GPP-Side system memory configuration:

1. Calculate the total system memory to reserve for DSP/BIOS Link.
2. Reserve required system memory from the GPP OS.
3. Define the memory map for the system memory reserved for DSP/BIOS Link.
4. Modify the Link Table.
5. Modify the MMU Table.
6. Modify the GPP Source File shm.c.
7. Modify the DSP-side Memory Map.
8. Modify the DSP Source File shm.c.
9. Rebuild GPP and DSP DSP/BIOS Link Drivers.
10. Rebuild GPP and DSP Application.

The above steps will now be followed to reduce the system memory requirements from 2MB to 72KB.

Step 1 - Calculate the total system memory to reserve for DSP/BIOS Link

In the case of an OMAP application, to establish how much system memory you need to reserve for DSP/BIOS Link, you need to calculate the following:

- The size of the largest buffer that needs to be exchanged between the GPP and DSP
- The amount of external memory required for the DSP-side application

The DSP subsystem within an OMAP591x device includes 160KB for internal RAM. Therefore, external memory for a DSP application should only be used if the DSP application requires more than 160KB of RAM. For example purposes, let us suppose that our application has the following requirements:

- The size of the largest buffer that needs to be exchanged between the GPP and DSP is 3KB
- The maximum amount of external memory required for the DSP-side application is 64KB

The above information can now be used to calculate the total system memory that needs to be reserved for DSP/BIOS Link. However, before we can do this, it is necessary to understand how the shared memory driver is designed. The shared memory driver uses a region of system memory to exchange data between the GPP and DSP. This region is divided up into three segments, which are defined as:

- Control Flags (size of SHM_Control)
- DSP to GPP Data Buffer (size of MAXBUFSIZE)
- GPP to DSP Data Buffer (size of MAXBUFSIZE)

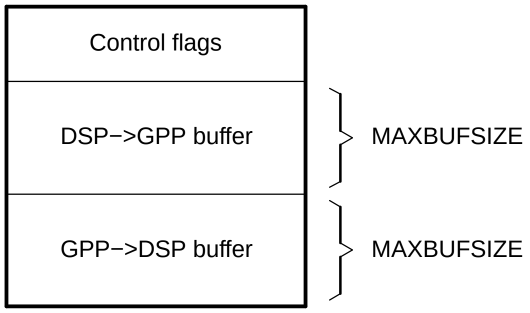


Figure 1. Shared Memory Driver

The size of the control flags region shown above is determined by the size of the structure, SHM_Control, defined by DSP/BIOS Link. The size of this structure may differ depending on DSP/BIOS Link configuration selected. To view the structure definition for SHM_Control, please refer to the shm.h located in the following directory:

```
<dsplink root>\gpp\src\ldrv\
```

For more details on the shared memory driver please refer to the document “DSP/BIOS Link Shared Memory IOM Driver for OMAP5910/5912."

Based on the above description, the total system memory used by the shared memory driver is ((2 * MAXBUFSIZE) + SHM_Control) bytes. Assuming that all DSP/BIOS Link modules (PROC, CHNL, and MSGQ) are being used the total size of the SHM_Control structure would be 28 bytes. By default, 1MB is reserved for the shared memory driver but in reality only 32796 bytes ((2 * 16384) + 28) is used. To calculate the total system memory requirements for a DSP/BIOS Link application, the user should use the following equation:
Total System Memory Required = ((2 * MAXBUFSIZE) + SHM_Control) + (External Memory for DSP Application)

Hence, for this example let us assume the total system memory required = ((2*3072) + 28) + (65536) = 71708 bytes.

Note:
When defining the total system memory requirement, you must also consider the MMU page size that is used by the GPP OS. For example, consider the case where the GPP OS uses 4KB pages. In the example above, the total system memory required is 71708 bytes and therefore, we should reserve 72KB (73728 bytes), which will be an even number of MMU pages. In other words, although the shared memory driver only requires 6172 ((2 * 3072) + 28) bytes of system memory, we will actually reserve 8KB of system memory for the driver.

Step 2 - Reserve required system memory from the GPP OS

To reserve a region of system memory, it may be necessary to refer to the configuration details of the embedded OS that is being used. For example, the Linux OS has a boot parameter called “mem” that indicates the total system memory available for the OS. Therefore when using Linux, to reserve system memory we simply redefine the boot parameter “mem” to be equal to the total system memory minus the total memory reserved for DSP/BIOS Link. The default configuration of DSP/BIOS Link requires 2MB of system memory and thus, for a system with 32MB of memory we would then define mem=30M. This would prevent the Linux OS from using the upper 2MB of the system memory. In the case of the above example, for a Linux based application, we would need to redefine mem=32696k. This would reserve the upper 72KB of system memory for DSP/BIOS Link.

Step 3 - Define the memory map for the system memory reserved for DSP/BIOS Link

The system memory reserved for DSP/BIOS Link is broken down into two regions: one region for use by the shared memory driver and the other region of external memory for the DSP application. In this example, 64KB is reserved for the DSP application and 8KB is reserved by the shared memory driver. When configuring the DSP MMU care must be taken when aligning these regions within the 72KB reserved. The DSP MMU for OMAP591x devices supports 1KB, 4KB, 64KB and 1MB pages. Hence, will we need to define 3 MMU entries when setting up the DSP MMU; two 4KB pages for the shared memory and one 64KB page for the DSP application. Regions should be defined such that they are aligned with regard to the page size being used. Consider that there is 32MB of system memory, in the case of the example, the regions should be aligned as follows:

- Region reserved for DSP application: 32704KB – 32768KB (64K boundary)
- Region reserved for Shared Memory Driver: 32696KB – 32704KB (8K boundary)

Step 4 - Modify the Link Table

The default Link Table is defined as follows:

```
[LINKTABLE0]
[0]
NAME             | S |   SHARED MEMORY DRIVER
ABBR             | S |   SHM
NUMCHANNELS      | N |   16
BASECHANNELID    | N |   0
MAXBUFSIZE       | N |   16384
INTERFACE        | A |   SHM_Interface
ARGUMENT1        | H |   0x11F00000
ARGUMENT2        | H |   0x0
[/0]
[/LINKTABLE0]
```

The parameters in the above Link Table that need to be modified are:

- MAXBUFSIZE
- ARGUMENT1

The MAXBUFSIZE field is the maximum size of the buffer to be exchanged between the GPP and DSP. Therefore, we simply redefined this field to be the size of the maximum buffer that will be exchanged. For this example, we would redefine this field as 3072 bytes.

The ARGUMENT1 field is the physical address where the shared memory is located. For an OMAP591x device, the physical base address for external SDRAM is 0x10000000. Hence, in the above LINKTABLE, the shared memory occupies the last MB in a 32MB SDRAM. In step 3, we decided that the shared memory would be located at an offset of 32696KB in a 32MB SDRAM. Therefore, for the example application ARGUMENT1 should be redefined as 0x11FEE000 (0x10000000 + 32969K) for a 32MB SDRAM.

Therefore, for this example the Link Table would be redefined as follows:

```
[LINKTABLE0]
[0]
NAME            | S |   SHARED MEMORY DRIVER
ABBR            | S |   SHM
NUMCHANNELS     | N |   16
BASECHANNELID   | N |   0
MAXBUFSIZE      | N |   3072
INTERFACE       | A |   SHM_Interface
ARGUMENT1       | H |   0x11FEE000
ARGUMENT2       | H |   0x0
[/0]
[/LINKTABLE0]
```

Step 5 - Modify the MMU Table

The default MMU Table is shown below:

```
[MMUTABLE0]
[0]
ADDRVIRTUAL     | H |   0x00400000
ADDRPHYSICAL    | H |   0x11F00000
SIZE            | H |   0
ACCESS          | H |   0x3
PRESERVE        | H |   0x1
MAPINGPP        | B |   FALSE
[/0]
[1]
ADDRVIRTUAL     | H |   0x00600000
ADDRPHYSICAL    | H |   0x11E00000
SIZE            | H |   0
ACCESS          | H |   0x3
PRESERVE        | H |   0x1
MAPINGPP        | B |   TRUE
[/1]
[/MMUTABLE0]
```

This MMU Table consists of two entries. The first MMU entry corresponds to the region of shared memory that is used to exchange data. The second entry corresponds to a region of system memory reserved for the DSP application. Each entry maps 1MB of system memory into the DSP's memory map. For this example, we need to define 3 MMU entries; two 4KB pages for the shared memory and one 64KB page for the DSP application. When defining these MMU entries, the parameters that need to be modified are:

- ADDRVIRTUAL
- ADDRPHYSICAL
- SIZE

The ADDRVIRTUAL is the DSP address that the location defined by ADDRPHYSICAL will be mapped to. When selecting the DSP address, ADDRVIRTUAL, you need to select a region in the DSP memory map that is not mapped to by any other internal or external memory. For OMAP591x devices, the range 0x28000-0xFF8000 (in bytes) is unused and can be mapped to an external location. For more details on the DSP Memory Map, refer to the *OMAP5912 Multimedia Processor DSP Subsystem Reference Guide* (SPRU750).

In step 3, we calculated that the base address for the shared memory region would be 0x11FEE000. For this example, the shared memory region spans 8KB, and so we define two 4KB MMU entries. The base address, ADDRPHYSICAL, of the first 4KB MMU entry would be 0x11FEE000 and the base address, ADDRPHYSICAL, of the second 4KB MMU entry would 0x11FEF000 (0x11FEE000 + 4K).

By default, the base address on the DSP (ADDRVIRTUAL) defined for the shared memory region is 0x400000. Since we are just changing the size of the shared memory region, we can use the same base address on the DSP for the location of this region. However, now that there are two 4KB MMU entries for the shared memory region we need to define the base address of each 4KB MMU entry on the DSP. On the DSP, the base address of the first 4KB MMU entry would be 0x400000 and the base address of the second 4KB MMU entry would be 0x401000 (0x400000 + 4K).

In step 2, we decided that the 64KB external memory region reserved for the DSP application would occupy the last 64KB our 32MB SDRAM. Given that the physical base address of the SDRAM is 0x10000000, the base address of the external region reserved for the DSP application would be 0x11FF0000 (0x10000000 + 32704K). Hence, when re-defining the ADDRPHYSICAL for the MMU entry that maps the external region for the DSP application this will be the address that we use. By default the base address on the DSP (ADDRVIRTUAL) defined for the external region of the DSP application is 0x600000. Since we are just changing the size of this we can use the same base address on the DSP. We could change the base address if we needed to, but for simplicity we shall leave this the same.

For this example, the MMU Table would be redefined as follows:

```
[MMUTABLE0]
[0]
ADDRVIRTUAL      | H |   0x00400000
ADDRPHYSICAL     | H |   0x11FEE000
SIZE             | H |   0x2
ACCESS           | H |   0x3
PRESERVE         | H |   0x1
MAPINGPP         | B |   FALSE
[/0]
[1]
ADDRVIRTUAL      | H |   0x00401000
ADDRPHYSICAL     | H |   0x11FEF000
SIZE             | H |   0x2
ACCESS           | H |   0x3
PRESERVE         | H |   0x1
MAPINGPP         | B |   FALSE
[/1]
[2]
ADDRVIRTUAL      | H |   0x00600000
ADDRPHYSICAL     | H |   0x11FF0000
SIZE             | H |   0x1
ACCESS           | H |   0x3
PRESERVE         | H |   0x1
MAPINGPP         | B |   TRUE
[/2]
[/MMUTABLE0]
```

An extra MMU entry has been added, so we need to also make one more modification to the configuration text file. If you view the configuration text you will notice that there is a section that lists parameters that are specific to the DSP(s) being used in the system. This section is denoted as [DSPn], where n is an index for the DSP in the system. In the case of OMAP, there is only one DSP and so there is only one DSP section. Within this section there is a parameter called “MMUENTRIES” that indicates the number of entries to be created in the DSP MMU. By default this value is two and because we have added another MMU entry this needs to be changed to three as shown below.

```
MMUENTRIES       | N |   3
```

Step 6 - Modify the GPP Source File shm.c

The two following changes need to be made to the function SHM_Initialize( ) in the GPP source file shm.c:

- By default, 1MB of memory is mapped for the shared memory driver. Since we have changed the amount of system memory reserved for the shared memory driver, we need to change the amount of memory that is mapped. For DSP/BIOS Link release v1.1x, the following line of the function SHM_Initialize( ) in shm.c needs to be modified (see below).

  ```
  mapInfo.size = 0x100000 ;
  ```

  This line defines the size of the memory region that will be mapped and should be changed accordingly. For our example, we would redefine this to be 8KB.

  ```
  mapInfo.size = 0x2000 ;
  ```

- Step 1 provided a description of how the shared memory region is architected. By default the pointer to the DSP to GPP buffer is fixed with an offset of ~0.5MB from the start of the shared memory region. Therefore, since we have changed the size of the shared memory region, we also need to modify this pointer. The default definition of this pointer is shown below (for DSP/BIOS Link v1.1x see function SHM_Initialize( ) in shm.c):

  ```
  shmInfo->ptrOutData = (Uint8 *)(shmInfo->ptrInpData
  + (0x100000-sizeof(SHM_Control))/2) ;
  ```

  This should be modified as follows:

  ```
  shmInfo->ptrOutData = (Uint8 *)(shmInfo->ptrInpData
  + shmInfo->maxBufSize) ;
  ```

Step 7 - Modify the DSP-side Memory Map

By default, the shared memory and external memory regions are defined as follows in the DSP memory map:

- Shared Memory (1MB): 0x200000-0x280000
- External Memory (1MB): 0x300000-0x380000

Note:
The DSP uses 16-bit word addressing whereas the GPP uses bytes addressing. Therefore, the address 0x200000 on the DSP-side actually corresponds to a byte address of 0x400000. Bearing this in mind, the addresses shown above should correspond to the ADDRVIRTUAL addresses defined in the Link Tables on the GPP-side.

The DSP memory map can be redefined either textually using the TCONF tool or graphically using the GCONF tool. Both tools are provided with TI’s Code Compose Studio™.

If you are using TCONF, then you can redefine the shared and external memory regions simply by editing the file dsplink-omap-base.tci (see <dsplink root>\dsp\src\tcf\). In the dsplink-omap-base.tci file, the shared and external memory regions are defined as follows:

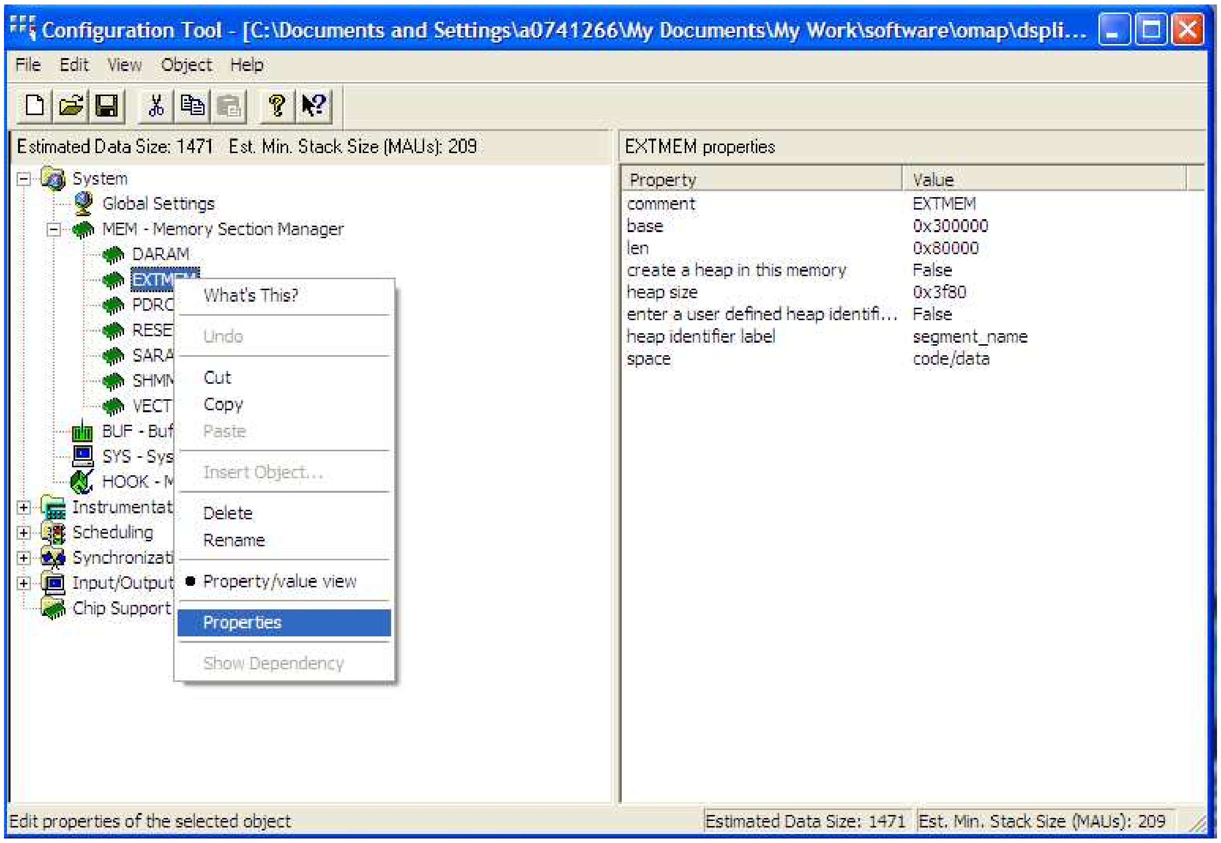
```
//=============================================================================
//  MEM : SHMMEM
//=============================================================================
SHMMEM = prog.module("MEM").create("SHMMEM");
prog.module("MEM").instance("SHMMEM").base = 2097152
prog.module("MEM").instance("SHMMEM").len = 524288
prog.module("MEM").instance("SHMMEM").createHeap = 0
prog.module("MEM").instance("SHMMEM").comment = "SHMMEM";
//=============================================================================
//  MEM : EXTMEM
//=============================================================================
EXTMEM = prog.module("MEM").create("EXTMEM");
prog.module("MEM").instance("EXTMEM").base = 3145728
prog.module("MEM").instance("EXTMEM").len = 524288
prog.module("MEM").instance("EXTMEM").createHeap = 0
prog.module("MEM").instance("EXTMEM").comment = "EXTMEM";
```

To redefine the shared and external memory regions such that they correspond to the GPP-side memory map defined in step 3, we would simply make the following changes. Remember that the length is defined in 16-bit words and not bytes. In the below changes, the base did not change. So long as the regions do not overlap, it is really up to the user on whether you wish to change the base because this is simply a virtual address. Once you have modified the dsplink-omap-base.tci, you can generate a new *.cdb file by running your tconf script.

```
//=============================================================================
//  MEM : SHMMEM
//=============================================================================
SHMMEM = prog.module("MEM").create("SHMMEM");
prog.module("MEM").instance("SHMMEM").base = 2097152
prog.module("MEM").instance("SHMMEM").len = 4096
prog.module("MEM").instance("SHMMEM").createHeap = 0
prog.module("MEM").instance("SHMMEM").comment = "SHMMEM";
//=============================================================================
//  MEM : EXTMEM
//=============================================================================
EXTMEM = prog.module("MEM").create("EXTMEM");
prog.module("MEM").instance("EXTMEM").base = 3145728
prog.module("MEM").instance("EXTMEM").len = 32768
prog.module("MEM").instance("EXTMEM").createHeap = 0
prog.module("MEM").instance("EXTMEM").comment = "EXTMEM";
```

If you are using GCONF, then you can redefine the shared and external memory regions simply by editing the *.cdb file directly. Do the following:

1. Open the *.cdb
2. Under the “System” folder, expand the “MEM – Memory Section Manager” icon.
3. Find the region that defines the external memory location (by default it is called “EXTMEM”). Right click on this item and select Properties.


Configuration Tool - [C:\Documents and Settings\a0741266\My Documents\My Work\software\omap\dspli...

Estimated Data Size: 1471 Est. Min. Stack Size (MAUs): 209

EXTMEM properties

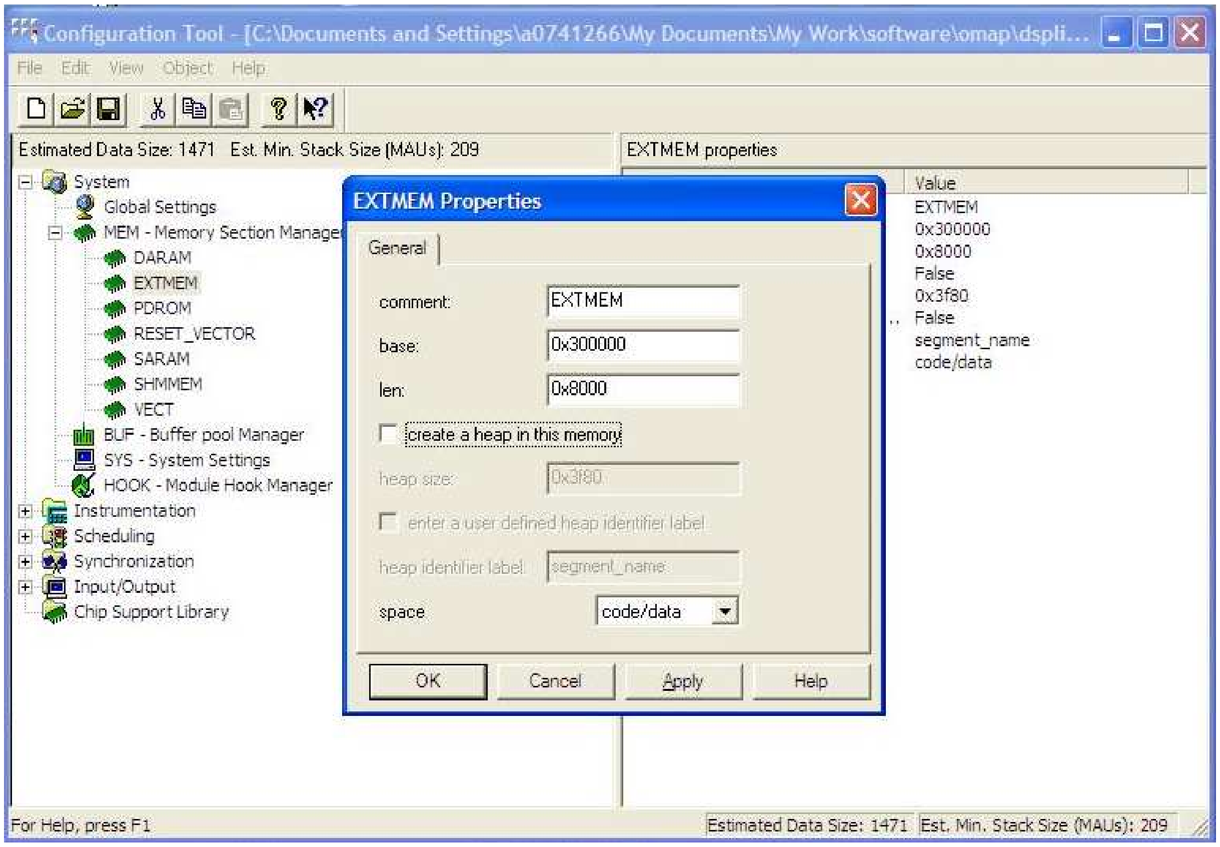
Property	Value
comment	EXTMEM
base	0x300000
len	0x80000
create a heap in this memory	False
heap size	0x3f80
enter a user defined heap identifi...	False
heap identifier label	segment_name
space	code/data

Edit properties of the selected object

Estimated Data Size: 1471 Est. Min. Stack Size (MAUs): 209

Figure 2. Memory Section Manager

4. Redefine the base and length of the external memory region appropriately. Click OK.


Configuration Tool - [C:\Documents and Settings\a0741266\My Documents\My Work\software\omap\dspli...

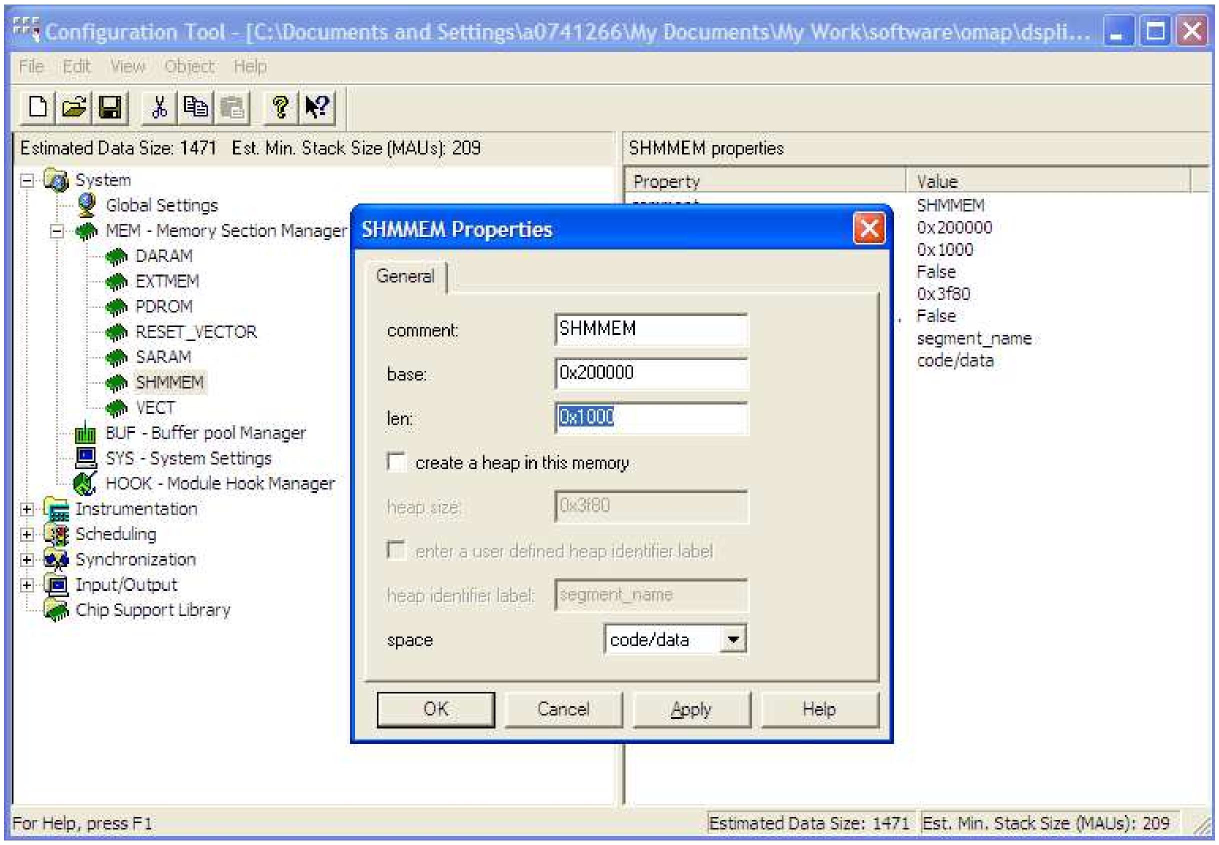
Estimated Data Size: 1471 Est. Min. Stack Size (MAUs): 209

EXTMEM Properties

comment: EXTMEM
base: 0x300000
len: 0x8000
create a heap in this memory
heap size: 0x3f80
enter a user defined heap identifier label
heap identifier label: segment_name
space: code/data

Figure 3. External Memory Region

5. Find the region that defines the shared memory location (by default it is called “SHMMEM”). Right click on this item and select Properties.
6. Redefine the base and length of the shared memory region appropriately. Click OK.


Configuration Tool - [C:\Documents and Settings\a0741266\My Documents\My Work\software\omap\dspli...
File Edit View Object Help
System
Global Settings
MEM - Memory Section Manager
DARAM
EXTMEM
PDROM
RESET_VECTOR
SARAM
SHMMEM
VECT
BUF - Buffer pool Manager
SYS - System Settings
HOOK - Module Hook Manager
Instrumentation
Scheduling
Synchronization
Input/Output
Chip Support Library
General
OK
Cancel
Apply
Help
For Help, press F1

Estimated Data Size: 1471 Est. Min. Stack Size (MAUs): 209

SHMMEM properties

Property	Value
	SHMMEM
	0x200000
	0x1000
	False
	0x3f80
	False
	segment_name
	code/data

SHMMEM Properties

comment: SHMMEM

base: 0x200000

len: 0x1000

[ ] create a heap in this memory

heap size: 0x3f80

[ ] enter a user defined heap identifier label

heap identifier label: segment_name

space: code/data

Figure 4. Shared Memory Region

7. Close and save the *.cdb file.

Note:
For debugging the DSP side of your DSP/BIOS Link application with Code Composer Studio (CCS), there is one more change that should be made. The CCS IDE prevents users from accessing non-valid memory regions by configuring the “Memory Map” of the target device within the IDE. The user can view the IDE’s memory map configuration for a target device by selecting the “Memory Map…” option under the “Tools” menu in CCS. Typically, the configuration of the IDE’s memory map is done via a startup GEL file that is executed when CCS is launched. Whatever startup GEL file you may be using, because we are changing the memory map of DSP we need to change the memory map configuration used to setup CCS.

DSP/BIOS Link provides GEL files that the user may use for configuring CCS when debugging a DSP/BIOS Link application. The GEL file used for the DSP is called "LINK_OMAP_BIOS.GEL" and can be found in the following directory:

```
<dsplink root>\etc\host\ccs\
```

If you view this file you will see that there is a function called InitMemoryMap( ) that configures the CCS memory mapping. In this function you will see that GEL functions called GEL_MapAdd( ) and GEL_MapAddStr( ) are used to configured the CCS memory mapping. For details on these functions please refer to CCS Help. These statements should be adjusted to reflect the changes that have been made. For example, by reconfiguring the system memory from 2MB to 72KB, we would need to make sure the following is defined:

```
/* Configure Program Space View - byte address! */
GEL_MapAdd (0x000100U, 0, 0x027F00U, 1, 1) ;    /* internal mem      */
GEL_MapAdd (0x600000U, 0, 0x010000U, 1, 1) ;    /* ext mem (64KB)    */
/* Configure Data Space View - word address! */
GEL_MapAdd (0x000080U, 1, 0x013F80U, 1, 1) ;    /* internal mem   */
GEL_MapAdd (0x200000U, 1, 0x001000U, 1, 1) ;    /* shm mem (4KW)  */
GEL_MapAdd (0x300000U, 1, 0x008000U, 1, 1) ;    /* ext mem (32KW) */
```

Step 8 - Modify the DSP Source File shm.c

In the DSP source file shm.c the following parameters need to be modified such that they are consistent with the GPP-side.

- SHM_BASE - base address (in 16-bit words) of shared memory on DSP-side
- SHM_LEN - length of shared memory region (in 16-bit words)
- SHM_BUFFER_LEN - length of GPP-to-DSP and DSP-to-GPP buffer (see step 1)

The SHM_BASE and SHM_LEN should be modified such that they match the base and length of the shared memory region defined in step 7.

```
const LgUns SHM_BASE = 0x200000 ;
const LgUns SHM_LEN  = 0x1000 ;
```

The SHM_BUFFER_LEN should be modified such that it matches the MAXBUFSIZE defined in step 4.

```
#define SHM_BUFFER_LEN    0x600
```

Note:
The SHM_BUFFER_LEN needs to be declared in words.

Step 9 - Rebuild GPP and DSP Link Drivers

Rebuild both the GPP-side and DSP-side DSP/BIOS Link drivers. For more details please refer to DSP/BIOS Link User Guide (see References).

Step 10 - Rebuild GPP and DSP Applications

Rebuild both the GPP-side and DSP-side application code using the new drivers.

Note:
When rebuilding the DSP-side application although it is possible to place sections in external memory (ie. in the EXTMEM section), for DSP/BIOS Link v1.1x it is not possible to place initialized data sections or certain uninitialized data sections in external memory. For example, placing the .cinit, .const, or .bss sections in external memory will cause the application to fail. This problem is caused due to the fact that the C55x™ DSP is a big endian device and the ARM9 is a little endian device. The OMAP591x devices provide endian conversion logic to help convert between little and big endian. However, DSP/BIOS Link v1.1x only supports the NOSWAP configuration of the endian conversion logic where endian conversion is not enabled. Therefore, when using DSP/BIOS Link v1.1x, it is recommended to keep data sections in internal memory and use external memory for non-runtime-critical code sections.

For more details on the OMAP591x endianism, please refer to *Using Endianess Conversion in the OMAP5910 Device* (SWPA027).

3 References

The following is a list of reference documentation that was used in conjunction with creating this application note.

1. *OMAP5910 Dual-Core Processor DSP Subsystems Reference Guide* (SPRU672)
2. *OMAP5910 Dual-Core Processor MPU Subsystems Reference Guide* (SPRU671)
3. *Programming the DSP MMU in the OMAP5910 Device* (SWPA038)
4. *OMAP5912 Multimedia Processor DSP Subsystem Reference Guide* (SPRU750)
5. *OMAP5912 Multimedia Processor OMAP3.2 Subsystem Reference Guide* (SPRU749)
6. *Using Endianess Conversion in the OMAP5910 Device* (SWPA027)

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