TMS320 DSP
DESIGNER'S NOTEBOOK

Linking C Data Objects Separate From the .bss Section

APPLICATION BRIEF: SPRA258

Leor Brenman
Digital Signal Processing Products
Semiconductor Group

Texas Instruments
June 1995

TEXAS INSTRUMENTS

IMPORTANT NOTICE

Texas Instruments (TI) reserves the right to make changes to its products or to discontinue any semiconductor product or service without notice, and advises its customers to obtain the latest version of relevant information to verify, before placing orders, that the information being relied on is current.

TI warrants performance of its semiconductor products and related software to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are utilized to the extent TI deems necessary to support this warranty. Specific testing of all parameters of each device is not necessarily performed, except those mandated by government requirements.

Certain application using semiconductor products may involve potential risks of death, personal injury, or severe property or environmental damage ("Critical Applications").

TI SEMICONDUCTOR PRODUCTS ARE NOT DESIGNED, INTENDED, AUTHORIZED, OR WARRANTED TO BE SUITABLE FOR USE IN LIFE-SUPPORT APPLICATIONS, DEVICES OR SYSTEMS OR OTHER CRITICAL APPLICATIONS.

Inclusion of TI products in such applications is understood to be fully at the risk of the customer. Use of TI products in such applications requires the written approval of an appropriate TI officer. Questions concerning potential risk applications should be directed to TI through a local SC sales office.

In order to minimize risks associated with the customer's applications, adequate design and operating safeguards should be provided by the customer to minimize inherent or procedural hazards.

TI assumes no liability for applications assistance, customer product design, software performance, or infringement of patents or services described herein. Nor does TI warrant or represent that any license, either express or implied, is granted under any patent right, copyright, mask work right, or other intellectual property right of TI covering or relating to any combination, machine, or process in which such semiconductor products or services might be or are used.

TRADEMARKS

TI is a trademark of Texas Instruments Incorporated.

Other brands and names are the property of their respective owners.

CONTACT INFORMATION

US TMS320 HOTLINE	(281) 274-2320
US TMS320 FAX	(281) 274-2324
US TMS320 BBS	(281) 274-2323
US TMS320 email	dsph@ti.com

Contents

Linking C Data Objects Separately From the .bss Section

Abstract

The TMS320 DSP C compilers produce several relocatable blocks of code and data when C code is compiled. These blocks are called sections and can be allocated into memory in a variety of ways to conform to a variety of system configurations. The `.bss` section is used by the compiler for global and static variables. It is one of the default COFF sections that is used to reserve a specified amount of space in the memory map that can later be used for storing data. It is normally uninitialized. All global and static variables in a C program are placed in the `.bss` section. On the 'C80, PP static and global variables are placed in the `.pbss` section that is assumed to be on chip. The `far` keyword can be used to force static or global variables to reside in the `.bss` section, which is assumed to be off chip. However, often it is desirable to place some of your variables separate from the `.bss` or `.pbss` section.

This document discusses how to implement this.

Design Problem

How do I link a C data object, such as an array, separately from the `.bss` section?

Solution

The TMS320 DSP C compilers produce several relocatable blocks of code and data when C code is compiled. These blocks are called sections and can be allocated into memory in a variety of ways to conform to a variety of system configurations. The `.bss` section is used by the compiler for global and static variables. It is one of the default COFF sections that is used to reserve a specified amount of space in the memory map that can later be used for storing data. It is normally uninitialized. All global and static variables in a C program are placed in the `.bss` section. On the 'C80, PP static and global variables are placed in the `.pbss` section that is assumed to be on chip. The `far` keyword can be used to force static or global variables to reside in the `.bss` section, which is assumed to be off chip. However, often it is desirable to place some of your variables separate from the `.bss` or `.pbss` section.

For example, on the floating-point DSPs you might want to link all of your variables into off-chip memory but place a frequently used array in on-chip RAM Block 0. On the fixed-point DSPs, for single-cycle data moves (DMOV) to take place, as required for FIR filtering, the data must reside in on-chip DARAM. However, not all of the variables need to reside in on-chip DARAM. On the 'C80, most of the data would be processed in PP on-chip data RAM blocks 1–3, but tables for packet transfers might be created in on-chip parameter RAM.

Method I

One method to accomplish this task is to declare the variable that is to be separate from the .bss or .pbss section in a separate file. This method works for the fixed-point, floating-point, and 'C80 DSP C compilers. For example, declare a 32-word array, `tapDelay[ ]`, in a file called `array.c` as follows:

```
/* File: ARRAY.C */
int tapDelay[32];
/* End of file */
```

All files that reference the variable must declare it as `extern`. Consider the following file, `test.c`, that makes a reference to the array declared in file `array.c` as follows:

```
/* File: TEST.C */
.
extern int tapDelay[ ];
```

```
.
void main(void)
{
   int i;
   .
   tapDelay[i] = 0;
   .
}
/* End of file */
```

In the linker command file, link this variable separate from the `.bss` section in the SECTIONS section. The following linker command file segment illustrates how to link the array tapDelay[ ] onto the TMS320C50's DARAM B2 on-chip dual-access data RAM while linking the rest of the global and static variables into part of on-chip SARAM:

```
/* File: TEST.CMD */
.
test.obj
array.obj
.
MEMORY
{
   .
   PAGE 1: DARAMB2: origin = 0x0060, length 0x0020
   PAGE 1: INTDATA: origin = 0x0c000, length 0x1800
   .
}

SECTIONS
{
   .
   .bss : {} >INTDATA PAGE 1
   .
   tapdelayline : {array.obj(.bss)} >DARAMB2 PAGE 1
}
/* End of file */
```

Method II

Another method that is available in the floating-point DSP C compiler version 4.60 and the 'C80 C compiler version 1.10 is to use the pragma DATA_SECTION. This is described in the TMS320 Floating-Point DSP Code Generation Tools Release 4.60 Getting Started document and the 'C80 Code Generation Tools User's Guide. Consider the example described in Method 1. The following code segment uses the DATA_SECTION pragma to declare a 32-word array, tapDelay[ ], that will be placed separate from the other global and static variables:

```
/* File: TEST.C */
#pragma DATA_SECTION (tapDelay, ".tapdelayline")
int tapDelay[32];
```

```
.
.
void main(void)
{
   int i;
   .
   tapDelay[i] = 0;
   .
}
/* End of file */
```

In the linker command file, use the section name .tapdelayline to place the array tapDelay[ ] in RAM block 0 separate from the other global and static variables that are in the .bss section as follows:

```
/* File: TEST.CMD */
.
test.obj
.
MEMORY
{
   .
   EXT0: org = 0x100 len = 0x3f00
   RAM0: org = 0x809800 len = 0x400
   .
}
SECTIONS
{
   .
   .bss : {} EXT0
   .
   .tapdelayline : {} RAM0

}
/* End of file */
```