Creating Single Section for Boot Load When **.const** *Section is Defined*

APPLICATION REPORT: SPRA409

Jeff Axelrod

Digital Signal Processing Solutions
May 1998

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.

Copyright © 1998, Texas Instruments Incorporated

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

Figures

Tables

Creating Single Section for Boot Load When *.const* Section is Defined

Abstract

The generation of a `.const` section that must be included in code for on-chip boot load poses a special problem when linking. In order to include the `.const` section data it must be grouped with all other needed sections to create a single section to be loaded by the on-chip boot loader. After the code is loaded, the `.const` section data must then be copied to a run location in data memory before execution of compiled code.

Therefore the `.const` section must have both a load address, where it is placed in program memory as a result of the on-chip boot load, and a run address, where it must be copied before execution of the compiled code.

Unfortunately, in the linker there is no way to specify a different run address for an input section that is being included within the body of another output section.

This document discusses the solution to this issue. Included in this application report are the implementation instructions and code examples for resolving this.

Product Support

World Wide Web

Our World Wide Web site at www.ti.com contains the most up to date product information, revisions, and additions. Users registering with TI&ME can build custom information pages and receive new product updates automatically via email.

Introduction

Code for this example can be obtained via the Internet by downloading and executing the following self-extracting file:

ftp://ftp.ti.com/pub/tms320bbs/c2xxfiles/ROMCEXAM.EXE

The generation of a `.const` section that must be included in code for on-chip boot load poses a special problem when linking. In order to include the `.const` section data it must be grouped with all other needed sections to create a single section to be loaded by the on-chip boot loader. After the code is loaded, the `.const` section data must then be copied to a run location in data memory before execution of compiled code.

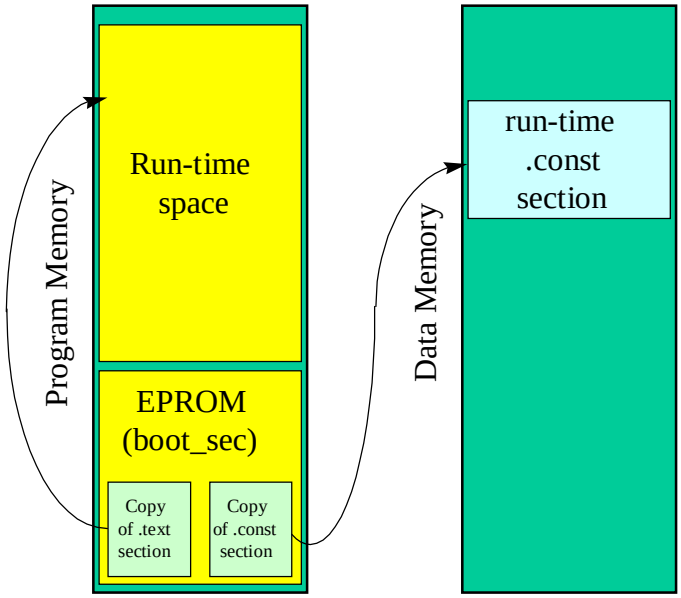
Therefore the `.const` section must have both a load address, where it is placed in program memory as a result of the on-chip boot load, and a run address, where it must be copied before execution of the compiled code.

Unfortunately, in the linker there is no way to specify a different run address for an input section that is being included within the body of another output section.

We'd like to be able to do the following, but this is not a valid linker command:

```
boot_sec : {   *(.text)
               *(.const): run = 0x800
```

Figure 1. Address Location Diagram



A different approach to setting both a load and run address for this section is therefore required.

This is accomplished by using partial linking. In a partial link all of the data for the `.const` sections are grouped together into a separate named section. This separate section contains the raw data and must be of the same size as the .const section. Two sections of the same size containing the same data are required to establish a run address and a load address for these data.

In the partial link all of the .const sections are gathered together into a single output section. The symbol names are hidden to prevent duplicate symbol definitions on the final link. The -g option reserves the necessary symbols for the load address and the length of the `.const` section that is to be copied.

Included Code and Files

This package includes a complete example C program and the necessary DOS batch files to build the object files and perform the two link stages. The dsphex utility is also called at the end of the batch file to generate an ASCII ROM image. Make sure you are using version 6.63 or greater of the dsphex utility. The previous versions did not generate boot ROMs correctly.

Table 1. Included files

File	Description
boot.asm	Boot loader extracted from rts.src and modified to copy the .const section from its load-time address to run-time address.
const.cmd	Linker command file for the first partial link stage.
Csample.bat	DOS batch file to build the complete application including dsphex ASCII ROM image.
Csample.c	Example C source that includes global initialized data in constant array x.
Csample.cmd	Linker command file for final link stage.
Cvectors.asm	Reset vector file which tells the device after boot loader completes to jump to _c_int0 (C initialization / boot loader)

To use the example, build the executable by typing csample.bat from a DOS command prompt, then load the debugger and type "go main2" at the prompt. In a UNIX environment examine the .bat file and execute the commands found inside.

The sections that follow show how to perform the required modifications to existing programs.

Partial Link Command File

Create a partial linker command file as shown:

```
-r              /* Perform partial link, retaining relocation
                information                                    */

-h              /* Hide symbol names to prevent duplicate
                definition errors on final link                */

boot.obj
csample.obj     /* List all .obj files to be included in the */
                /* Link. THESE MUST BE SPECIFIED IN SAME ORDER
                AS THEY WILL APPEAR IN THE FINAL LINK          */

-m part.map
-o const.out    /* Name output file to be included in final link */
-l rts2xx.lib
-g __const_load
-g __const_length

SECTIONS
{
  nconst: { __const_load = .;
            *(.const)
            __const_length = . - __const_load;
          }     /* COMBINE ALL .const SECTIONS INTO A NEWLY
                NAMED SECTION */
```

Final Link Command File

At the final link we are going to throw away all sections defined in the output file created by the partial link (const.out in the example above), except for the renamed section "nconst". This we will use to provide the data to be loaded as part of the on-chip boot load.

```
/* C203 Linker command file for generating single boot section */
/* combining .text and .cinit sections                         */
/*

NOTE:
Ignore the following linker errors:

>> warning: absolute symbol cinit being redefined
>> warning: output file has no .text section

*/

-c

-m csample.map
-o csample.out

csample.obj
boot.obj

-l rts2xx.lib

/* Hack to make it possible to go main by typing go main2 in
   the debugger */

_main2=_main;

MEMORY
{
PAGE 0:
        PROG: origin=0x0000 , LENGTH=0x8000
        EPROM: origin=0x8000, LENGTH=0x8000

PAGE 1:
DATA: origin=0x800 , LENGTH=0x4000
}

SECTIONS
{
```

```
/* This dummy section is to remove all sections from const.out
except for the .const section. This is necessary, since they will be
linked in at this stage, to prevent each of these sections from
being included twice. If any other sections are used that are linked
in at the partial link stage, they must be entered here also. */

dummy(DSECT) : {  const.out(.text)
const.out(.cinit)
const.out(.stack)
const.out(.data)
const.out(.sysmem)
const.out(.bss)
const.out(.switch)
}

.const(NOLOAD): {  __const_run = .;
*(.const)
         } load = DATA PAGE 1


  boot_sec: { /* vectors must be included first so they
                 fall into program memory location 0 */
              *(.vectors)

              boot.obj(.text)

              /* include .text section */
              *(.text)

              /* include .const section */
              *(nconst)  /* include raw data from partial link */

              /* Set start address for C init table */
              cinit = .;

              /* Include all cinit sections */
              *(.cinit)

/* Reserve a single space for the zero word to mark end of cinit */
              .+=1;

            } fill = 0x0000, /* Make sure fill value is 0 */
              load = PROG PAGE 0

  .data   : {} > DATA      PAGE 1
  .bss    : {} > DATA      PAGE 1
  .const  : {} > DATA      PAGE 1
  .sysmem : {} > DATA      PAGE 1
  .stack  : {} > DATA      PAGE 1

}
```

Modification of boot.asm File

The final section of the solution is to change the code in boot.asm to allow the copy of the .const section at run-time. This has already been performed and is included in this distribution. However, to do this yourself:

1) Extract the boot routine from rts.src file:

```
dspar x rts.src boot.asm
```

2) Edit boot.asm:

   change

```
CONST_COPY  .set  0
```

   to

```
CONST_COPY .set  1
```

3) Search for .if CONST_COPY and remove the lines:

```
.sect ".cmark"
.label __const_load
```

4) Re-assemble boot.asm:

```
dspa [options] boot.asm
```

This completes the solution.