

*TMS320 DSP
DESIGNER'S NOTEBOOK*

Multipass Linking

APPLICATION BRIEF: SPRA257

*Tom Horner
Digital Signal Processing Products
Semiconductor Group*

*Texas Instruments
February 1995*



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

Abstract.....	7
Design Problem	8
Solution	8

Examples

Example 1. Code Listing file1.asm	9
Example 2. Code Listing file2.asm	10

Multipass Linking



Abstract

Large projects are frequently broken down into subsystem teams, each of which develops their portion of the application independently of the others. In order to create a final executable, all the modules from all the subsystem teams need to be combined. One way to do this is to have each subsystem team supply a linked relocatable file to the system integrator. The system integrator would then link all the subsystem files together to create the final executable. The advantage of this technique is that only the relocatable object file needs to be supplied to the system integrator, not all the original source files. This can make version control of the final executable simpler because only the relocatable output files need to be tracked.

This document describes how the multipass link process is performed.



Design Problem

How do you perform multipass linking?

Large projects are frequently broken down into subsystem teams, which develop their portion of the application independently of the others. In order to create a final executable, all the modules from the subsystem teams need to be combined. One way to do this is to have each subsystem team supply a linked relocatable file to the system integrator. The system integrator would then link all the subsystem files together to create the final executable. The advantage of this technique is that only the relocatable object file needs to be supplied to the system integrator, not all the original source files. This can make version control of the final executable simpler because only the relocatable output files need to be tracked.

Solution

The multipass link process is performed as follows:

STEP 1:

Combine assembled source files with no allocation information into a relocatable output file. You use the `-r` switch to create a relocatable output file. Also, the output file must contain symbolic information. This is the default condition for the linker and can only be overridden by the `-s` switch. Therefore, for multipass linking does not use the `-s` switch when creating the output file. If a linker command file is used when creating a relocatable output file, it should only contain the input files, output file, and `-r` switch. Do not include the `MEMORY` or `SECTIONS` sections at this step.

STEP 2:

The sections defined in the relocatable output files are now allocated into the defined memory space to create the final executable file. It is at this step that all allocation, binding, and `MEMORY` directives are performed. A standard linker command file is used, with the exception that instead of object files (`.obj`), relocatable output files are used for the input.

An example set of files is shown in Figure 1 to illustrate the procedure for a 'C5x application. The `make.bat` file contains the commands to perform both link steps. These steps would probably all be performed independently of one another on a large project. Note that in STEP 1, it is not necessary to use a linker command file to create the relocatable module (`file1` and `file3`).



Example 1. Code Listing file1.asm

```
----- MAKE.BAT -----
rem STEP 1: Create relocatable output files
rem ---

dsps -v50 file1.asm
dsplnk -r -o file1.out file1.obj

dsps -v50 file2.asm
dsplnk file2.cmd

dsps -v50 file3.asm
dsplnk -r -o file3.out file3.obj

rem STEP 2: Create final executable
rem ---

dsplnk make.cmd
----- MAKE.CMD -----
file1.out
file2.out
file3.out
    •o final.out
    •m final.map

MEMORY
PAGE 0:          /***** PROGRAM SPACE *****/
    VECS:  org = 0x0    len = 0x30    /* INTERRUPT VECTORS */
    PROG:  org = 0x30    len = 0x7fd0 /* EXTERNAL PROGRAM MEMORY */

PAGE 1:          /***** DATA SPACE *****/
    MMR:    org = 0x0    len = 0x60    /* MEMORY MAP REGISTERS */
    BLKB2:  org = 0x60    len = 0x20    /* RAM BLOCK B2 */
    BLKB0:  org = 0x100   len = 0x200   /* RAM BLOCK B0 */
    BLKB1:  org = 0x300   len = 0x200   /* RAM BLOCK B1 */
    RAM:    org = 0x8000  len = 0x8000 /* EXTERNAL DATA MEMORY */

SECTIONS
    vectors      : { } > VECS PAGE 0    /* from file1 */
    .text        : { } > PROG PAGE 0    /* from file2 and file3 */
    .bss         : {file2.out(.bss)
                  file3.out(.bss)
                  file1.out(.bss)
                  } > BLKB2 PAGE 1      /* from all files */
----- FILE2.CMD -----
file2.obj
    •r
    •o file2.out
----- FILE1.ASM -----
    .mmregs      ;Enable memory map register labels
;GLOBAL VARIABLES
```




```
.global B0, B1, B2, SARAM
.global START, MAIN, INT0, INT1, INT2
.global TINT, RINT, XINT, TRINT, TXINT
.global INT3, TRAP, NMI
.global test1, test2, test3

;DATA MEMORY DEFINITION
.bss      test1, 1

;INTERRUPT VECTORS
.sect      "vectors"      ;Section for external interrupt vectors
B          START          ;Processor Reset
B          INT0            ;External Interrupt #0
B          INT1            ;External Interrupt #1
B          INT2            ;External Interrupt #2
B          TINT            ;Timer Interrupt
B          RINT            ;Serial Port Receive Interrupt
B          XINT            ;Serial Port Transmit Interrupt
B          TRINT           ;TDM Serial Port Receive Interrupt
B          TXINT           ;TDM Serial Port Transmit Interrupt
B          INT3            ;External Interrupt #3
.space     10*16          ;Reserved space - 10 words
b          TRAP            ;S/W Trap
b          NMI             ;Non-maskable external interrupt

.end
```

Example 2. Code Listing file2.asm

```
----- FILE2.ASM -----
.mmregs                      ;Enable memory map register labels
;GLOBAL VARIABLES

.global B0, B1, B2, SARAM
.global START, MAIN, INT0, INT1, INT2
.global TINT, RINT, XINT, TRINT, TXINT
.global INT3, TRAP, NMI
.global test1, test2, test3

;DEFINE CONSTANTS
B0          .set 0100h      ;Define constants for internal memory
B1          .set 0300h      ;RAM blocks start address
B2          .set 060h
SARAM       .set 0800h

;DATA MEMORY DEFINITION
.bss      test2, 1

.text                      ;Section for program code

;PROGRAM
START
    LDP          #0          ;Initialize data pointer
    setc         INTM        ;Global interrupt disable.
    SPLK         #0h,IMR     ;Clear interrupt mask register
```



```

:
:
:
MAIN
    idle
    b          MAIN
.end

----- FILE3.ASM -----
    .mmregs          ;Enable memory map register labels
;GLOBAL VARIABLES

.global B0, B1, B2, SARAM
    .global START, MAIN, INT0, INT1, INT2
    .global TINT, RINT, XINT, TRINT, TXINT
    .global INT3, TRAP, NMI
    .global test1, test2, test3
;DATA MEMORY DEFINITION
    .bss          test3,1

    .text

;INTERRUPT SERVICE ROUTINES
RINT          ;SERIAL PORT RECEIVE INTERRUPT
    ldp          #0
    lacc          DRR,4          ;Read latest AIC input w/ 16x gain
    samm          DXR          ;Echo to AIC output
    rete          ;Return to MAIN w/ interrupt enable
XINT          ;SERIAL PORT TRANSMIT INTERRUPT
    samm          DXR
    rete
;UNUSED INTERRUPT TRAPS
INT0  idle          ;External Interrupt #0
INT1  idle          ;External Interrupt #1
INT2  idle          ;External Interrupt #2
TINT  idle          ;Timer Interrupt
;RINT idle          ;Serial Port Receive Interrupt
;XINT idle          ;Serial Port Transmit Interrupt
TRINT idle          ;TDM Serial Port Receive Interrupt
TXINT idle          ;TDM Serial Port Transmit Interrupt
INT3  idle          ;External Interrupt #3
TRAP  idle          ;S/W Trap
NMI   idle          ;Non-maskable external interrupt

    .end
```