TMS320 DSP
DESIGNER'S NOTEBOOK
Number 61
TEXAS INSTRUMENTS

Multipass Linking

Contributed by Tom Horner

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 executeable, 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 relocateable file to the system integrator. The system integrator would then link all the subsystem files together to create the final executeable. The advantage of this technique is that only the relocateable object file needs to be supplied to the system integrator, not all the original source files. This can make version control of the final executeable simpler because only the relocateable 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 relocateable output file. You use the -r switch to create a relocateable 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 relocateable 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 relocateable output files are now allocated into the defined memory space to create the final executeable 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),` relocateable 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 relocateable module (file1 and file3).

```
———————————————— MAKE.BAT ————————————————
rem STEP 1: Create relocateable output files
rem ———

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

dspa -v50 file2.asm
dsplnk file2.cmd

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

rem STEP 2: Create final executeable
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
```

Figure 1. file1.asm

```
.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
```

Figure 1. file1.asm *(continued)*

```
———————————————————————— 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
```

Figure 2. file2.asm

```
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
```

Figure 2. file2.asm *(continued)*