These three definitions are inserted before main{}

// These are defined by the linker (see F28335.cmd)

**extern** Uint16 RamfuncsLoadStart;

**extern** Uint16 RamfuncsLoadEnd;

**extern** Uint16 RamfuncsRunStart;

//extern Uint16 RamfuncsLoadSize;

These code are inserted after initPieVecTable();

// The RamfuncsLoadStart, RamfuncsLoadEnd, and RamfuncsRunStart

// symbols are created by the linker. Refer to the F28335.cmd file.

// memcpy(&RamfuncsRunStart, &RamfuncsLoadStart, (Uint32)&RamfuncsLoadSize);

memcpy(&RamfuncsRunStart, &RamfuncsLoadStart, (Uint32)&RamfuncsLoadEnd);

// Call Flash Initialization to setup flash waitstates

// This function must reside in RAM

**InitFlash**();

The following the error messages:

"../F28335.cmd", line 78: error #10263: RAML0 memory range has already been specified

"../F28335.cmd", line 78: error #10264: RAML0 memory range overlaps existing memory range RAML0

"../F28335.cmd", line 79: error #10264: RAML1 memory range overlaps existing memory range RAML1L2

"../F28335.cmd", line 80: error #10264: RAML2 memory range overlaps existing memory range RAML1L2

"../F28335.cmd", line 81: error #10263: RAML3 memory range has already been specified

"../F28335.cmd", line 81: error #10264: RAML3 memory range overlaps existing memory range RAML3

"../F28335.cmd", line 83: error #10264: ZONE7 memory range overlaps existing memory range ZONE7A

"../F28335.cmd", line 91: error #10263: CSM\_RSVD memory range has already been specified

"../F28335.cmd", line 91: error #10264: CSM\_RSVD memory range overlaps existing memory range CSM\_RSVD

"../F28335.cmd", line 92: error #10263: BEGIN memory range has already been specified

"../F28335.cmd", line 93: error #10263: CSM\_PWL memory range has already been specified

"../F28335.cmd", line 93: error #10264: CSM\_PWL memory range overlaps existing memory range CSM\_PWL

"../F28335.cmd", line 95: error #10263: ADC\_CAL memory range has already been specified

"../F28335.cmd", line 95: error #10264: ADC\_CAL memory range overlaps existing memory range ADC\_CAL

"../F28335.cmd", line 97: error #10263: IQTABLES memory range has already been specified

"../F28335.cmd", line 97: error #10264: IQTABLES memory range overlaps existing memory range IQTABLES

"../F28335.cmd", line 98: error #10263: IQTABLES2 memory range has already been specified

"../F28335.cmd", line 98: error #10264: IQTABLES2 memory range overlaps existing memory range IQTABLES2

"../F28335.cmd", line 99: error #10263: FPUTABLES memory range has already been specified

"../F28335.cmd", line 99: error #10264: FPUTABLES memory range overlaps existing memory range FPUTABLES

"../F28335.cmd", line 100: error #10264: ROM memory range overlaps existing memory range BOOTROM

"../F28335.cmd", line 101: error #10263: RESET memory range has already been specified

"../F28335.cmd", line 101: error #10264: RESET memory range overlaps existing memory range RESET

"../F28335.cmd", line 110: error #10263: RAMM1 memory range has already been specified

"../F28335.cmd", line 110: error #10264: RAMM1 memory range overlaps existing memory range RAMM1

"../F28335.cmd", line 111: error #10263: RAML4 memory range has already been specified

"../F28335.cmd", line 111: error #10264: RAML4 memory range overlaps existing memory range RAML4

"../F28335.cmd", line 112: error #10263: RAML5 memory range has already been specified

"../F28335.cmd", line 112: error #10264: RAML5 memory range overlaps existing memory range RAML5

"../F28335.cmd", line 113: error #10263: RAML6 memory range has already been specified

"../F28335.cmd", line 113: error #10264: RAML6 memory range overlaps existing memory range RAML6

"../F28335.cmd", line 114: error #10263: RAML7 memory range has already been specified

"../F28335.cmd", line 114: error #10264: RAML7 memory range overlaps existing memory range RAML7

warning #10247-D: creating output section "DLOG" without a SECTIONS specification

error #10010: errors encountered during linking; "FlashRun\_pmsm28335.out" not built

>> Compilation failure

The following is the F28335.cmd file.

/\*

// TI File $Revision: /main/9 $

// Checkin $Date: August 28, 2007 11:23:38 $

//###########################################################################

//

// FILE: F28335.cmd

//

// TITLE: Linker Command File For F28335 Device

//

//###########################################################################

// $TI Release: DSP2833x Header Files V1.01 $

// $Release Date: September 26, 2007 $

//###########################################################################

\*/

/\* ======================================================

// For Code Composer Studio V2.2 and later

// ---------------------------------------

// In addition to this memory linker command file,

// add the header linker command file directly to the project.

// The header linker command file is required to link the

// peripheral structures to the proper locations within

// the memory map.

//

// The header linker files are found in <base>\DSP2833x\_Headers\cmd

//

// For BIOS applications add: DSP2833x\_Headers\_BIOS.cmd

// For nonBIOS applications add: DSP2833x\_Headers\_nonBIOS.cmd

========================================================= \*/

/\* ======================================================

// For Code Composer Studio prior to V2.2

// --------------------------------------

// 1) Use one of the following -l statements to include the

// header linker command file in the project. The header linker

// file is required to link the peripheral structures to the proper

// locations within the memory map \*/

/\* Uncomment this line to include file only for non-BIOS applications \*/

/\* -l DSP2833x\_Headers\_nonBIOS.cmd \*/

/\* Uncomment this line to include file only for BIOS applications \*/

/\* -l DSP2833x\_Headers\_BIOS.cmd \*/

/\* 2) In your project add the path to <base>\DSP2833x\_headers\cmd to the

library search path under project->build options, linker tab,

library search path (-i).

/\*========================================================= \*/

/\* Define the memory block start/length for the F28335

PAGE 0 will be used to organize program sections

PAGE 1 will be used to organize data sections

Notes:

Memory blocks on F28335 are uniform (ie same

physical memory) in both PAGE 0 and PAGE 1.

That is the same memory region should not be

defined for both PAGE 0 and PAGE 1.

Doing so will result in corruption of program

and/or data.

L0/L1/L2 and L3 memory blocks are mirrored - that is

they can be accessed in high memory or low memory.

For simplicity only one instance is used in this

linker file.

Contiguous SARAM memory blocks can be combined

if required to create a larger memory block.

\*/

**MEMORY**

{

**PAGE** **0:** /\* Program Memory \*/

/\* Memory (RAM/FLASH/OTP) blocks can be moved to PAGE1 for data allocation \*/

ZONE0 : origin = 0x004000, length = 0x001000 /\* XINTF zone 0 \*/

RAML0 : origin = 0x008000, length = 0x001000 /\* on-chip RAM block L0 \*/

RAML1 : origin = 0x009000, length = 0x001000 /\* on-chip RAM block L1 \*/

RAML2 : origin = 0x00A000, length = 0x001000 /\* on-chip RAM block L2 \*/

RAML3 : origin = 0x00B000, length = 0x001000 /\* on-chip RAM block L3 \*/

ZONE6A : origin = 0x100000, length = 0x00FC00 /\* XINTF zone 6 - program space\*/

ZONE7 : origin = 0x200000, length = 0x100000 /\* XINTF zone 7 \*/

FLASHH : origin = 0x300000, length = 0x008000 /\* on-chip FLASH \*/

FLASHG : origin = 0x308000, length = 0x008000 /\* on-chip FLASH \*/

FLASHF : origin = 0x310000, length = 0x008000 /\* on-chip FLASH \*/

FLASHE : origin = 0x318000, length = 0x008000 /\* on-chip FLASH \*/

FLASHD : origin = 0x320000, length = 0x008000 /\* on-chip FLASH \*/

FLASHC : origin = 0x328000, length = 0x008000 /\* on-chip FLASH \*/

FLASHA : origin = 0x338000, length = 0x007F80 /\* on-chip FLASH \*/

CSM\_RSVD : origin = 0x33FF80, length = 0x000076 /\* Part of FLASHA. Program with all 0x0000 when CSM is in use. \*/

BEGIN : origin = 0x33FFF6, length = 0x000002 /\* Part of FLASHA. Used for "boot to Flash" bootloader mode. \*/

CSM\_PWL : origin = 0x33FFF8, length = 0x000008 /\* Part of FLASHA. CSM password locations in FLASHA \*/

OTP : origin = 0x380400, length = 0x000400 /\* on-chip OTP \*/

ADC\_CAL : origin = 0x380080, length = 0x000009 /\* ADC\_cal function in Reserved memory \*/

IQTABLES : origin = 0x3FE000, length = 0x000b50 /\* IQ Math Tables in Boot ROM \*/

IQTABLES2 : origin = 0x3FEB50, length = 0x00008c /\* IQ Math Tables in Boot ROM \*/

FPUTABLES : origin = 0x3FEBDC, length = 0x0006A0 /\* FPU Tables in Boot ROM \*/

ROM : origin = 0x3FF27C, length = 0x000D44 /\* Boot ROM \*/

RESET : origin = 0x3FFFC0, length = 0x000002 /\* part of boot ROM \*/

VECTORS : origin = 0x3FFFC2, length = 0x00003E /\* part of boot ROM \*/

**PAGE** 1 : /\* Data Memory \*/

/\* Memory (RAM/FLASH/OTP) blocks can be moved to PAGE0 for program allocation \*/

/\* Registers remain on PAGE1 \*/

BOOT\_RSVD : origin = 0x000000, length = 0x000050 /\* Part of M0, BOOT rom will use this for stack \*/

RAMM0 : origin = 0x000050, length = 0x0003B0 /\* on-chip RAM block M0 \*/

RAMM1 : origin = 0x000400, length = 0x000400 /\* on-chip RAM block M1 \*/

RAML4 : origin = 0x00C000, length = 0x001000 /\* on-chip RAM block L1 \*/

RAML5 : origin = 0x00D000, length = 0x001000 /\* on-chip RAM block L1 \*/

RAML6 : origin = 0x00E000, length = 0x001000 /\* on-chip RAM block L1 \*/

RAML7 : origin = 0x00F000, length = 0x001000 /\* on-chip RAM block L1 \*/

ZONE6B : origin = 0x10FC00, length = 0x000400 /\* XINTF zone 6 - data space \*/

FLASHB : origin = 0x330000, length = 0x008000 /\* on-chip FLASH \*/

}

/\* Allocate sections to memory blocks.

Note:

codestart user defined section in DSP28\_CodeStartBranch.asm used to redirect code

execution when booting to flash

ramfuncs user defined section to store functions that will be copied from Flash into RAM

\*/

**SECTIONS**

{

/\* Allocate program areas: \*/

**.cinit** : > FLASHA **PAGE** = 0

**.pinit** : > FLASHA, **PAGE** = 0

**.text** : > FLASHA **PAGE** = 0

codestart : > BEGIN **PAGE** = 0

ramfuncs : **LOAD** = FLASHD,

**RUN** = RAML0,

**LOAD\_START**(\_RamfuncsLoadStart),

**LOAD\_END**(\_RamfuncsLoadEnd),

**RUN\_START**(\_RamfuncsRunStart),

**PAGE** = 0

csmpasswds : > CSM\_PWL **PAGE** = 0

csm\_rsvd : > CSM\_RSVD **PAGE** = 0

/\* Allocate uninitalized data sections: \*/

**.stack** : > RAMM1 **PAGE** = 1

**.ebss** : > RAML4 **PAGE** = 1

.esysmem : > RAMM1 **PAGE** = 1

/\* Initalized sections go in Flash \*/

/\* For SDFlash to program these, they must be allocated to page 0 \*/

.econst : > FLASHA **PAGE** = 0

.switch : > FLASHA **PAGE** = 0

/\* Allocate IQ math areas: \*/

IQmath : > FLASHC **PAGE** = 0 /\* Math Code \*/

IQmathTables : > IQTABLES, **PAGE** = 0, **TYPE** = **NOLOAD**

IQmathTables2 : > IQTABLES2, **PAGE** = 0, **TYPE** = **NOLOAD**

FPUmathTables : > FPUTABLES, **PAGE** = 0, **TYPE** = **NOLOAD**

/\* Allocate DMA-accessible RAM sections: \*/

DMARAML4 : > RAML4, **PAGE** = 1

DMARAML5 : > RAML5, **PAGE** = 1

DMARAML6 : > RAML6, **PAGE** = 1

DMARAML7 : > RAML7, **PAGE** = 1

/\* Allocate 0x400 of XINTF Zone 6 to storing data \*/

ZONE6DATA : > ZONE6B, **PAGE** = 1

/\* .reset is a standard section used by the compiler. It contains the \*/

/\* the address of the start of \_c\_int00 for C Code. /\*

/\* When using the boot ROM this section and the CPU vector \*/

/\* table is not needed. Thus the default type is set here to \*/

/\* DSECT \*/

**.reset** : > RESET, **PAGE** = 0, **TYPE** = **DSECT**

vectors : > VECTORS **PAGE** = 0, **TYPE** = **DSECT**

/\* Allocate ADC\_cal function (pre-programmed by factory into TI reserved memory) \*/

.adc\_cal : load = ADC\_CAL, **PAGE** = 0, **TYPE** = **NOLOAD**

}

/\*

//===========================================================================

// End of file.

//===========================================================================

\*/

// TI File $Revision: /main/1 $

// Checkin $Date: August 18, 2006 13:46:33 $

//###########################################################################

//

// FILE: DSP2833x\_MemCopy.c

//

// TITLE: Memory Copy Utility

//

// ASSUMPTIONS:

//

//

//

// DESCRIPTION:

//

// This function will copy the specified memory contents from

// one location to another.

//

// Uint16 \*SourceAddr Pointer to the first word to be moved

// SourceAddr < SourceEndAddr

// Uint16\* SourceEndAddr Pointer to the last word to be moved

// Uint16\* DestAddr Pointer to the first destination word

//

// No checks are made for invalid memory locations or that the

// end address is > then the first start address.

//

//

//###########################################################################

// $TI Release: DSP2833x Header Files V1.01 $

// $Release Date: September 26, 2007 $

//###########################################################################

#include "DSP2833x\_Device.h"

void MemCopy(Uint16 \*SourceAddr, Uint16\* SourceEndAddr, Uint16\* DestAddr)

{

while(SourceAddr < SourceEndAddr)

{

\*DestAddr++ = \*SourceAddr++;

}

return;

}