```
;
;   EVM320C243 Test Code
;   Copyright (c) 1997.
;   Spectrum Digital, Inc.
;   ALL RIGHTS RESERVED
;


;
;   This is a modification of the boot routine V6.60 from the TI run time
;   support library. This was chosen as a starting point because TI
;   split the C5x and 2x/2xx combination.
;


;
;
;   This module contains the following definitions :
;
;         __stack    - Stack memory area
;         _c_int0    - Boot function
;         _var_init  - Function which processes initialization tables
;
;
        .global  _c_int0, cinit
        .global  _main,_abort
        .global .bss, end


WD_CNTR         .set    07023h          ;WD Counter reg
WD_KEY          .set    07025h          ;WD Key reg
WD_CNTL         .set    07029h          ;WD Control reg

                .include x24x_app.h


;-------------------------------------------------------------------------------
; Debug directives
;-------------------------------------------------------------------------------
;               .def    GPR0                    ;General purpose registers.
;               .def    GPR1
;               .def    GPR2
;               .def    GPR3


;-------------------------------------------------------------------------------
; Variable Declarations for on chip RAM Blocks
;-------------------------------------------------------------------------------
;               .bss    GPR0,1          ;General purpose registers.
;               .bss    GPR1,1
;               .bss    GPR2,1
;               .bss    GPR3,1
;               .bss    REG5,1
;               .bss    REGA,1

;-------------------------------------------------------------------------------
; M A C R O - Definitions
;-------------------------------------------------------------------------------
SBIT0           .macro  DMA, MASK       ;Clear bit Macro
                LACC    DMA
                AND     #(0FFFFh-MASK)
                SACL    DMA
                .endm
```



```
SBIT1           .macro  DMA, MASK       ;Set bit Macro
                LACC    DMA
                OR      #MASK
                SACL    DMA
                .endm

KICK_DOG        .macro                  ;Watchdog reset macro
                LDP     #00E0h
                SPLK    #05555h, WDKEY
                SPLK    #0AAAAh, WDKEY
                LDP     #0h
                .endm


GPR0    .word 0                 ; Gen purp reg


;
; CONST COPY OPTION
; If your system cannot support allocating an initialized section to data
; memory, and you want the boot routine to copy .const from program to
; data memory, then set this CONST_COPY variable to 1
;
; Note the code that does the copy depends on you having the following
; in your linker command file
;
;       MEMORY
;       {
;         PAGE 0 : PROG : ...   /* 'PROG' AND 'DATA' ARE EXAMPLE NAMES */
;         PAGE 1 : DATA : ...
;         ...
;       }
;
;       SECTIONS
;       {
;         ...
;         .const : load = PROG PAGE 0, run = DATA PAGE 1
;                  {
;                    __const_run = . ;
;                    *(.c_mark)
;                    *(.const)
;                    __const_length = . - __const_run;
;                  }
;         ...
;       }
;
CONST_COPY      .set    0

;
; For CONST COPY, Define the load address of the .const section
; depends on linker command file being written as above
;
        .if     CONST_COPY
        .sect   ".c_mark"
        .label  __const_load

        .global __const_run, __const_length

        .text
        .endif ; CONST_COPY

;
```

```
; Declare the stack.  Size is determined by the linker option -stack
;
__stack:        .usect  ".stack",0

;
; FUNCTION DEF : _c_int0
;
;   1) Set up stack
;   2) Set up proper status
;   3) If "cinit" is not -1, init global variables
;   4) call users' program
;
;
_c_int0:                            ; entry point from reset vector

                LDP             #0h
       SETC     INTM                ;Disable interrupts
       SETC     CNF
;
; Initialize status bit fields *NOT* initialized at reset
;
       CLRC     XF                  ; turn off xf bit
       CLRC     SXM                 ;Clear Sign Extension Mode
       CLRC     OVM                 ;Reset Overflow Mode
       SPLK #0000h,IMR  ; Mask all core interrupts
       LACC IFR                         ; Read Interrupt flags
       SACL IFR                         ; Clear all interrupt flags

       LDP     #0E0h
       SPLK             #0000h,SCSR1
       SPLK    #006Fh,WDCR
       KICK_DOG
       SPLK #0h,GPR0            ; Set wait state generator for
       OUT GPR0,WSGR            ; external address space

;      LDP     #00E0h
;      SPLK    #00CBh,PLL_CNTL2     ;CLKIN(XTAL)=10MHz,CPUCLK=20MHz
;      SPLK    #00C3h,PLL_CNTL1     ;CLKMD=PLL Enable,SYSCLK=CPUCLK/2,
;      SPLK    #40C0h,SYSCR         ;CLKOUT=CPUCLK, no reset,

;       LDP     #0000h
;       SPLK    #4h,GPR3
;       OUT     GPR3,WSGR            ;Set XMIF to run w/no(0) wait states
                                     ;1 wait state for offchip I/O reads


;
;  Set up initial stack and frame pointers
;
       LRLK    AR0,__stack             ; set up frame pointer
       LRLK    AR1,__stack             ; set up stack pointer

;
; Initialize status bit fields which are set to these same values by reset.
; If you run this routine from reset, you can comment out this code.
;
       SPM     0                       ; product shift count of zero
       MAR     *,AR0                   ; AR = 0, mls 10/07/96

       SSXM                            ; set SXM=1 for next instruction
;
; If cinit is not -1, process initialization tables
```



```
;
       LALK    cinit                   ; get pointer to init tables
       ADDK    1
       BZ      skip                    ; if (cinit == -1)

       CALL    _var_init,AR1           ; var_init()

;
;  Call the user's program
;
skip:
       .if     CONST_COPY
       CALL    const_copy
       .endif

                                ; start of hw stack init code
       LACK    #0014h     ; vectors for underflow
       SACL    *
       PSHD    *
       PSHD    *
       PSHD    *
       PSHD    *
       PSHD    *
       PSHD    *
       PSHD    *
       PSHD    *

       CALL    _main,AR1
       CALL    _abort,AR1              ; to never return...

       .page
;
;  FUNCTION DEF : _var_init
;
;  PROCESS INITIALIZATION TABLES.  TABLES ARE IN
;  PROGRAM MEMORY IN THE FOLLOWING FORMAT :
;
;       .word  <length of init data in words>
;       .word  <address of variable to initialize>
;       .word  <init data>
;       .word  ...
;
;  The init table is terminated with a zero length
;
;

_var_init:

;
; C2xx Version
;
       ADRK    2                       ; allocate two words of local memory
       LALK    cinit                   ; load accumulator with base of table
       LARP    AR0

;
;  Read init record header.
;  An init record with a zero length terminates the list.
;
loop:
       TBLR    *+                      ; read length
       ADDK    1
```

```
        TBLR    *                       ; read address

        LAR     AR2,*-                  ; load variable address into ar2
        LAR     AR3,*,AR3               ; load count into ar3
        BANZ    copy,*-,AR2             ; check for end of table

;
;  At end of list, return to caller
;
        LARP    AR1
        SBRK    2                       ; deallocate locals
        RET                             ; return to _c_int0

;
;  Perform the copy of data from program to data
;
copy:
        ADDK    1                       ; increment pointer to data
        TBLR    *+,AR3                  ; copy data from program to variable
        BANZ    copy,*-,AR2             ; until count is zero

        ADDK    1                       ; point to beginning of next record
        B       loop,AR0                ; go process next record


        .page

;
; CONST COPY Routine - copies the .const section from program to data memory
;
        .if CONST_COPY
const_copy:

;
; C2xx version - must use 'RPT *' because RPTK count isn't big enough
;
        LALK    #__const_length         ; load length of const section
        BZ      quit                    ; if 0, quit
        LRLK    AR2,#__const_run        ; AR2 = const address in data
        LALK    #__const_length-1       ; load length - 1
        SACL    *                       ; write to temp

        RPT     *,AR2                   ; repeat length times
        BLKP    #__const_load,*+        ; block copy from program

        LARP    AR1                     ; restore ARP to SP
quit:
        RET                             ; return

        .endif ; CONST_COPY

        .text
_abort          B _abort

        .end
```