Forum: 32-bit PIC
Topic: AN1388 PIC32MM256GPM028 Bootloader jump application
Subject: AN1388 PIC32MM256GPM028 Bootloader jump application
作者: chawy_siao
2020年05月19日 09:04:43

大家好最近在做PIC32MM256GPM028 bootloader, 已經可以交握跟燒錄進去但在跳去應用程式時都會發生software breakpoint, 附檔是我bootloader程式、規畫的linker script 跟燒錄的hex檔案及map檔, 不知道小弟是哪裡沒有考量進去造成錯誤, 感激不盡~

Bootloader linker script

```
/* Default linker script, for normal executables */
OUTPUT_FORMAT("elf32-tradlittlemips")
OUTPUT_ARCH(pic32mx)
ENTRY(_reset)
/*
 * Provide for a minimum stack and heap size
 * - _min_stack_size - represents the minimum space that must be made
 *                                    available for the stack.    Can be overridden from
 *                                    the command line using the linker's --defsym option.
 * - _min_heap_size    - represents the minimum space that must be made
 *                                    available for the heap.    Can be overridden from
 *                                    the command line using the linker's --defsym option.
 */
EXTERN (_min_stack_size _min_heap_size)

/*****************************************************************************
 * Processor-specific object file.    Contains SFR definitions.
 *****************************************************************************/
INPUT("processor.o")

/*****************************************************************************
 * For interrupt vector handling
 *****************************************************************************/
PROVIDE(_vector_spacing = 0x00000001);
_ebase_address = 0x9D000000;

/*****************************************************************************
 * Memory Address Equates
 * _RESET_ADDR            -- Reset Vector
 * _BEV_EXCPT_ADDR    -- Boot exception Vector
 * _DBG_EXCPT_ADDR    -- In-circuit Debugging Exception Vector
 * _DBG_CODE_ADDR      -- In-circuit Debug Executive address
 * _DBG_CODE_SIZE      -- In-circuit Debug Executive size
 * _SIMPLE_TLB_REFILL_EXCPT_ADDR    -- Simple TLB-Refill Exception Vector
 * _GEN_EXCPT_ADDR    -- General Exception Vector
```

```
 ***************************************************************************/
_RESET_ADDR                  = 0xBFC00000;
_BEV_EXCPT_ADDR              = 0xBFC00380;
_DBG_EXCPT_ADDR              = 0xBFC00480;
_DBG_CODE_ADDR              = 0x9FC00490;
_DBG_CODE_SIZE               = 0x760;
_SIMPLE_TLB_REFILL_EXCPT_ADDR  = _ebase_address + 0;
_GEN_EXCPT_ADDR              = _ebase_address + 0x180;

/***************************************************************************
 * Memory Regions
 *
 * Memory regions without attributes cannot be used for orphaned sections.
 * Only sections specifically assigned to these regions can be allocated
 * into these regions.
 ***************************************************************************/
MEMORY
{
  kseg0_program_mem    (rx)    ORIGIN = 0x9D001000, LENGTH = 0x3000    /* All C Files
will be located here */
  kseg0_boot_mem              : ORIGIN = 0x9FC00490, LENGTH = 0x0      /* This memory
region is dummy */
  kseg1_boot_mem              : ORIGIN = 0xBFC00000, LENGTH = 0x490    /* C Startup code
*/
  config3                    : ORIGIN = 0xBFC00BF0, LENGTH = 0x4
    config2                  : ORIGIN = 0xBFC00BF4, LENGTH = 0x4
    config1                  : ORIGIN = 0xBFC00BF8, LENGTH = 0x4
    config0                  : ORIGIN = 0xBFC00BFC, LENGTH = 0x4
    kseg0_data_mem      (w!x)    ORIGIN = 0x80000000, LENGTH = 0x8000
    sfrs                       : ORIGIN = 0xBF800000, LENGTH = 0x100000
    debug_exec_mem             : ORIGIN = 0x9FC00490, LENGTH = 0x760
    configsfrs                 : ORIGIN = 0xBFC00BF0, LENGTH = 0x10
}

/***************************************************************************
 * Configuration-word sections
 ***************************************************************************/
SECTIONS
{
   config_BFC02FF0 : {
    KEEP(*(.config_BFC00BF0))
    } >config3
  .config_BFC02FF4 : {
    KEEP(*(.config_BFC00BF4))
    } >config2
  .config_BFC02FF8 : {
    KEEP(*(.config_BFC00BF8))
    } >config1
  .config_BFC02FFC : {
```

```
    KEEP(*(.config_BFC00BFC))
    } >config0
}
PROVIDE(_DBG_CODE_ADDR = 0x9FC00490) ;
PROVIDE(_DBG_CODE_SIZE = 0x760) ;
SECTIONS
{
  /* Boot Sections */
  .reset _RESET_ADDR :
    {
    KEEP(*(.reset))
    KEEP(*(.reset.startup))
    } >kseg1_boot_mem
  .bev_excpt _BEV_EXCPT_ADDR :
    {
    KEEP(*(.bev_handler))
    } >kseg1_boot_mem
  /* Debug exception vector */
  .dbg_excpt _DBG_EXCPT_ADDR (NOLOAD) :
    {
        . DEFINED (_DEBUGGER) ? 0x8 : 0x0);
    } >kseg1_boot_mem
  /* Space reserved for the debug executive */
  .dbg_code _DBG_CODE_ADDR (NOLOAD) :
    {
        . DEFINED (_DEBUGGER) ? _DBG_CODE_SIZE : 0x0);
    } >debug_exec_mem

  .app_excpt _GEN_EXCPT_ADDR :
    {
    KEEP(*(.gen_handler))
    } >kseg0_program_mem

  .vector_0 _ebase_address + 0x200 + ((_vector_spacing << 3) * 0) :
    {
      KEEP(*(.vector_0))
    } >kseg0_program_mem
    ASSERT(_vector_spacing == 0 || SIZEOF(.vector_0) <= (_vector_spacing << 3), "function
at exception vector 0 too large")
   vector_1 _ebase_address + 0x200 + ((_vector_spacing << 3) * 1) :
    {
      KEEP(*(.vector_1))
    } >kseg0_program_mem
    ASSERT(_vector_spacing == 0 || SIZEOF(.vector_1) <= (_vector_spacing << 3), "function
at exception vector 1 too large")
   vector_2 _ebase_address + 0x200 + ((_vector_spacing << 3) * 2) :
    {
      KEEP(*(.vector_2))
    } >kseg0_program_mem
```

```
    ASSERT(_vector_spacing == 0 || SIZEOF(.vector_2) <= (_vector_spacing << 3), "function
at exception vector 2 too large")
   vector_3 _ebase_address + 0x200 + ((_vector_spacing << 3) * 3) :
    {
      KEEP(*(.vector_3))
    } kseg0_program_mem
    ASSERT(_vector_spacing == 0 || SIZEOF(.vector_3) <= (_vector_spacing << 3), "function
at exception vector 3 too large")
   vector_4 _ebase_address + 0x200 + ((_vector_spacing << 3) * 4) :
    {
      KEEP(*(.vector_4))
    } kseg0_program_mem
    ASSERT(_vector_spacing == 0 || SIZEOF(.vector_4) <= (_vector_spacing << 3), "function
at exception vector 4 too large")
   vector_5 _ebase_address + 0x200 + ((_vector_spacing << 3) * 5) :
    {
      KEEP(*(.vector_5))
    } kseg0_program_mem
    ASSERT(_vector_spacing == 0 || SIZEOF(.vector_5) <= (_vector_spacing << 3), "function
at exception vector 5 too large")
   vector_6 _ebase_address + 0x200 + ((_vector_spacing << 3) * 6) :
    {
      KEEP(*(.vector_6))
    } kseg0_program_mem
    ASSERT(_vector_spacing == 0 || SIZEOF(.vector_6) <= (_vector_spacing << 3), "function
at exception vector 6 too large")
   vector_7 _ebase_address + 0x200 + ((_vector_spacing << 3) * 7) :
    {
      KEEP(*(.vector_7))
    } kseg0_program_mem
    ASSERT(_vector_spacing == 0 || SIZEOF(.vector_7) <= (_vector_spacing << 3), "function
at exception vector 7 too large")
   vector_8 _ebase_address + 0x200 + ((_vector_spacing << 3) * 8) :
    {
      KEEP(*(.vector_8))
    } kseg0_program_mem
    ASSERT(_vector_spacing == 0 || SIZEOF(.vector_8) <= (_vector_spacing << 3), "function
at exception vector 8 too large")
   vector_9 _ebase_address + 0x200 + ((_vector_spacing << 3) * 9) :
    {
      KEEP(*(.vector_9))
    } kseg0_program_mem
    ASSERT(_vector_spacing == 0 || SIZEOF(.vector_9) <= (_vector_spacing << 3), "function
at exception vector 9 too large")
   vector_10 _ebase_address + 0x200 + ((_vector_spacing << 3) * 10) :
    {
      KEEP(*(.vector_10))
    } kseg0_program_mem
    ASSERT(_vector_spacing == 0 || SIZEOF(.vector_10) <= (_vector_spacing << 3), "function
```

```
at exception vector 10 too large")
   vector_11 _ebase_address + 0x200 + ((_vector_spacing << 3) * 11) :
    {
      KEEP(*(.vector_11))
    } kseg0_program_mem
    ASSERT(_vector_spacing == 0 || SIZEOF(.vector_11) <= (_vector_spacing << 3), "function
at exception vector 11 too large")
   vector_12 _ebase_address + 0x200 + ((_vector_spacing << 3) * 12) :
    {
      KEEP(*(.vector_12))
    } kseg0_program_mem
    ASSERT(_vector_spacing == 0 || SIZEOF(.vector_12) <= (_vector_spacing << 3), "function
at exception vector 12 too large")
   vector_13 _ebase_address + 0x200 + ((_vector_spacing << 3) * 13) :
    {
      KEEP(*(.vector_13))
    } kseg0_program_mem
    ASSERT(_vector_spacing == 0 || SIZEOF(.vector_13) <= (_vector_spacing << 3), "function
at exception vector 13 too large")
   vector_14 _ebase_address + 0x200 + ((_vector_spacing << 3) * 14) :
    {
      KEEP(*(.vector_14))
    } kseg0_program_mem
    ASSERT(_vector_spacing == 0 || SIZEOF(.vector_14) <= (_vector_spacing << 3), "function
at exception vector 14 too large")
   vector_15 _ebase_address + 0x200 + ((_vector_spacing << 3) * 15) :
    {
      KEEP(*(.vector_15))
    } kseg0_program_mem
    ASSERT(_vector_spacing == 0 || SIZEOF(.vector_15) <= (_vector_spacing << 3), "function
at exception vector 15 too large")
   vector_16 _ebase_address + 0x200 + ((_vector_spacing << 3) * 16) :
    {
      KEEP(*(.vector_16))
    } kseg0_program_mem
    ASSERT(_vector_spacing == 0 || SIZEOF(.vector_16) <= (_vector_spacing << 3), "function
at exception vector 16 too large")
   vector_17 _ebase_address + 0x200 + ((_vector_spacing << 3) * 17) :
    {
      KEEP(*(.vector_17))
    } kseg0_program_mem
    ASSERT(_vector_spacing == 0 || SIZEOF(.vector_17) <= (_vector_spacing << 3), "function
at exception vector 17 too large")
   vector_18 _ebase_address + 0x200 + ((_vector_spacing << 3) * 18) :
    {
      KEEP(*(.vector_18))
    } kseg0_program_mem
    ASSERT(_vector_spacing == 0 || SIZEOF(.vector_18) <= (_vector_spacing << 3), "function
at exception vector 18 too large")
```

```
   vector_19 _ebase_address + 0x200 + ((_vector_spacing << 3) * 19) :
    {
      KEEP(*(.vector_19))
    } kseg0_program_mem
    ASSERT(_vector_spacing == 0 || SIZEOF(.vector_19) <= (_vector_spacing << 3), "function
at exception vector 19 too large")
   vector_20 _ebase_address + 0x200 + ((_vector_spacing << 3) * 20) :
    {
      KEEP(*(.vector_20))
    } kseg0_program_mem
    ASSERT(_vector_spacing == 0 || SIZEOF(.vector_20) <= (_vector_spacing << 3), "function
at exception vector 20 too large")
   vector_21 _ebase_address + 0x200 + ((_vector_spacing << 3) * 21) :
    {
      KEEP(*(.vector_21))
    } kseg0_program_mem
    ASSERT(_vector_spacing == 0 || SIZEOF(.vector_21) <= (_vector_spacing << 3), "function
at exception vector 21 too large")
   vector_22 _ebase_address + 0x200 + ((_vector_spacing << 3) * 22) :
    {
      KEEP(*(.vector_22))
    } kseg0_program_mem
    ASSERT(_vector_spacing == 0 || SIZEOF(.vector_22) <= (_vector_spacing << 3), "function
at exception vector 22 too large")
   vector_23 _ebase_address + 0x200 + ((_vector_spacing << 3) * 23) :
    {
      KEEP(*(.vector_23))
    } kseg0_program_mem
    ASSERT(_vector_spacing == 0 || SIZEOF(.vector_23) <= (_vector_spacing << 3), "function
at exception vector 23 too large")
   vector_24 _ebase_address + 0x200 + ((_vector_spacing << 3) * 24) :
    {
      KEEP(*(.vector_24))
    } kseg0_program_mem
    ASSERT(_vector_spacing == 0 || SIZEOF(.vector_24) <= (_vector_spacing << 3), "function
at exception vector 24 too large")
   vector_25 _ebase_address + 0x200 + ((_vector_spacing << 3) * 25) :
    {
      KEEP(*(.vector_25))
    } kseg0_program_mem
    ASSERT(_vector_spacing == 0 || SIZEOF(.vector_25) <= (_vector_spacing << 3), "function
at exception vector 25 too large")
   vector_26 _ebase_address + 0x200 + ((_vector_spacing << 3) * 26) :
    {
      KEEP(*(.vector_26))
    } kseg0_program_mem
    ASSERT(_vector_spacing == 0 || SIZEOF(.vector_26) <= (_vector_spacing << 3), "function
at exception vector 26 too large")
   vector_27 _ebase_address + 0x200 + ((_vector_spacing << 3) * 27) :
```

```
    {
      KEEP(*(.vector_27))
    } kseg0_program_mem
    ASSERT(_vector_spacing == 0 || SIZEOF(.vector_27) <= (_vector_spacing << 3), "function
at exception vector 27 too large")
   vector_28 _ebase_address + 0x200 + ((_vector_spacing << 3) * 28) :
    {
      KEEP(*(.vector_28))
    } kseg0_program_mem
    ASSERT(_vector_spacing == 0 || SIZEOF(.vector_28) <= (_vector_spacing << 3), "function
at exception vector 28 too large")
   vector_29 _ebase_address + 0x200 + ((_vector_spacing << 3) * 29) :
    {
      KEEP(*(.vector_29))
    } kseg0_program_mem
    ASSERT(_vector_spacing == 0 || SIZEOF(.vector_29) <= (_vector_spacing << 3), "function
at exception vector 29 too large")
   vector_30 _ebase_address + 0x200 + ((_vector_spacing << 3) * 30) :
    {
      KEEP(*(.vector_30))
    } kseg0_program_mem
    ASSERT(_vector_spacing == 0 || SIZEOF(.vector_30) <= (_vector_spacing << 3), "function
at exception vector 30 too large")
   vector_31 _ebase_address + 0x200 + ((_vector_spacing << 3) * 31) :
    {
      KEEP(*(.vector_31))
    } kseg0_program_mem
    ASSERT(_vector_spacing == 0 || SIZEOF(.vector_31) <= (_vector_spacing << 3), "function
at exception vector 31 too large")
   vector_32 _ebase_address + 0x200 + ((_vector_spacing << 3) * 32) :
    {
      KEEP(*(.vector_32))
    } kseg0_program_mem
    ASSERT(_vector_spacing == 0 || SIZEOF(.vector_32) <= (_vector_spacing << 3), "function
at exception vector 32 too large")
   vector_33 _ebase_address + 0x200 + ((_vector_spacing << 3) * 33) :
    {
      KEEP(*(.vector_33))
    } kseg0_program_mem
    ASSERT(_vector_spacing == 0 || SIZEOF(.vector_33) <= (_vector_spacing << 3), "function
at exception vector 33 too large")
   vector_34 _ebase_address + 0x200 + ((_vector_spacing << 3) * 34) :
    {
      KEEP(*(.vector_34))
    } kseg0_program_mem
    ASSERT(_vector_spacing == 0 || SIZEOF(.vector_34) <= (_vector_spacing << 3), "function
at exception vector 34 too large")
   vector_35 _ebase_address + 0x200 + ((_vector_spacing << 3) * 35) :
    {
```

```
      KEEP(*(.vector_35))
    } kseg0_program_mem
    ASSERT(_vector_spacing == 0 || SIZEOF(.vector_35) <= (_vector_spacing << 3), "function
at exception vector 35 too large")
   vector_36 _ebase_address + 0x200 + ((_vector_spacing << 3) * 36) :
    {
      KEEP(*(.vector_36))
    } kseg0_program_mem
    ASSERT(_vector_spacing == 0 || SIZEOF(.vector_36) <= (_vector_spacing << 3), "function
at exception vector 36 too large")
   vector_37 _ebase_address + 0x200 + ((_vector_spacing << 3) * 37) :
    {
      KEEP(*(.vector_37))
    } kseg0_program_mem
    ASSERT(_vector_spacing == 0 || SIZEOF(.vector_37) <= (_vector_spacing << 3), "function
at exception vector 37 too large")
   vector_38 _ebase_address + 0x200 + ((_vector_spacing << 3) * 38) :
    {
      KEEP(*(.vector_38))
    } kseg0_program_mem
    ASSERT(_vector_spacing == 0 || SIZEOF(.vector_38) <= (_vector_spacing << 3), "function
at exception vector 38 too large")
   vector_39 _ebase_address + 0x200 + ((_vector_spacing << 3) * 39) :
    {
      KEEP(*(.vector_39))
    } kseg0_program_mem
    ASSERT(_vector_spacing == 0 || SIZEOF(.vector_39) <= (_vector_spacing << 3), "function
at exception vector 39 too large")
   vector_40 _ebase_address + 0x200 + ((_vector_spacing << 3) * 40) :
    {
      KEEP(*(.vector_40))
    } kseg0_program_mem
    ASSERT(_vector_spacing == 0 || SIZEOF(.vector_40) <= (_vector_spacing << 3), "function
at exception vector 40 too large")
   vector_41 _ebase_address + 0x200 + ((_vector_spacing << 3) * 41) :
    {
      KEEP(*(.vector_41))
    } kseg0_program_mem
    ASSERT(_vector_spacing == 0 || SIZEOF(.vector_41) <= (_vector_spacing << 3), "function
at exception vector 41 too large")
   vector_42 _ebase_address + 0x200 + ((_vector_spacing << 3) * 42) :
    {
      KEEP(*(.vector_42))
    } kseg0_program_mem
    ASSERT(_vector_spacing == 0 || SIZEOF(.vector_42) <= (_vector_spacing << 3), "function
at exception vector 42 too large")
   vector_43 _ebase_address + 0x200 + ((_vector_spacing << 3) * 43) :
    {
      KEEP(*(.vector_43))
```

```
    } kseg0_program_mem
    ASSERT(_vector_spacing == 0 || SIZEOF(.vector_43) <= (_vector_spacing << 3), "function
at exception vector 43 too large")
   vector_44 _ebase_address + 0x200 + ((_vector_spacing << 3) * 44) :
    {
      KEEP(*(.vector_44))
    } kseg0_program_mem
    ASSERT(_vector_spacing == 0 || SIZEOF(.vector_44) <= (_vector_spacing << 3), "function
at exception vector 44 too large")
   vector_45 _ebase_address + 0x200 + ((_vector_spacing << 3) * 45) :
    {
      KEEP(*(.vector_45))
    } kseg0_program_mem
    ASSERT(_vector_spacing == 0 || SIZEOF(.vector_45) <= (_vector_spacing << 3), "function
at exception vector 45 too large")
   vector_46 _ebase_address + 0x200 + ((_vector_spacing << 3) * 46) :
    {
      KEEP(*(.vector_46))
    } kseg0_program_mem
    ASSERT(_vector_spacing == 0 || SIZEOF(.vector_46) <= (_vector_spacing << 3), "function
at exception vector 46 too large")
   vector_47 _ebase_address + 0x200 + ((_vector_spacing << 3) * 47) :
    {
      KEEP(*(.vector_47))
    } kseg0_program_mem
    ASSERT(_vector_spacing == 0 || SIZEOF(.vector_47) <= (_vector_spacing << 3), "function
at exception vector 47 too large")
   vector_48 _ebase_address + 0x200 + ((_vector_spacing << 3) * 48) :
    {
      KEEP(*(.vector_48))
    } kseg0_program_mem
    ASSERT(_vector_spacing == 0 || SIZEOF(.vector_48) <= (_vector_spacing << 3), "function
at exception vector 48 too large")
   vector_49 _ebase_address + 0x200 + ((_vector_spacing << 3) * 49) :
    {
      KEEP(*(.vector_49))
    } kseg0_program_mem
    ASSERT(_vector_spacing == 0 || SIZEOF(.vector_49) <= (_vector_spacing << 3), "function
at exception vector 49 too large")
   vector_50 _ebase_address + 0x200 + ((_vector_spacing << 3) * 50) :
    {
      KEEP(*(.vector_50))
    } kseg0_program_mem
    ASSERT(_vector_spacing == 0 || SIZEOF(.vector_50) <= (_vector_spacing << 3), "function
at exception vector 50 too large")
   vector_51 _ebase_address + 0x200 + ((_vector_spacing << 3) * 51) :
    {
      KEEP(*(.vector_51))
    } kseg0_program_mem
```

```
    ASSERT(_vector_spacing == 0 || SIZEOF(.vector_51) <= (_vector_spacing << 3), "function
at exception vector 51 too large")
   vector_52 _ebase_address + 0x200 + ((_vector_spacing << 3) * 52) :
    {
      KEEP(*(.vector_52))
    } kseg0_program_mem
    ASSERT(_vector_spacing == 0 || SIZEOF(.vector_52) <= (_vector_spacing << 3), "function
at exception vector 52 too large")
   vector_53 _ebase_address + 0x200 + ((_vector_spacing << 3) * 53) :
    {
      KEEP(*(.vector_53))
    } kseg0_program_mem
    ASSERT(_vector_spacing == 0 || SIZEOF(.vector_53) <= (_vector_spacing << 3), "function
at exception vector 53 too large")
   vector_54 _ebase_address + 0x200 + ((_vector_spacing << 3) * 54) :
    {
      KEEP(*(.vector_54))
    } kseg0_program_mem
    ASSERT(_vector_spacing == 0 || SIZEOF(.vector_54) <= (_vector_spacing << 3), "function
at exception vector 54 too large")
   vector_55 _ebase_address + 0x200 + ((_vector_spacing << 3) * 55) :
    {
      KEEP(*(.vector_55))
    } kseg0_program_mem
    ASSERT(_vector_spacing == 0 || SIZEOF(.vector_55) <= (_vector_spacing << 3), "function
at exception vector 55 too large")
   vector_56 _ebase_address + 0x200 + ((_vector_spacing << 3) * 56) :
    {
      KEEP(*(.vector_56))
    } kseg0_program_mem
    ASSERT(_vector_spacing == 0 || SIZEOF(.vector_56) <= (_vector_spacing << 3), "function
at exception vector 56 too large")
   vector_57 _ebase_address + 0x200 + ((_vector_spacing << 3) * 57) :
    {
      KEEP(*(.vector_57))
    } kseg0_program_mem
    ASSERT(_vector_spacing == 0 || SIZEOF(.vector_57) <= (_vector_spacing << 3), "function
at exception vector 57 too large")
   vector_58 _ebase_address + 0x200 + ((_vector_spacing << 3) * 58) :
    {
      KEEP(*(.vector_58))
    } kseg0_program_mem
    ASSERT(_vector_spacing == 0 || SIZEOF(.vector_58) <= (_vector_spacing << 3), "function
at exception vector 58 too large")
   vector_59 _ebase_address + 0x200 + ((_vector_spacing << 3) * 59) :
    {
      KEEP(*(.vector_59))
    } kseg0_program_mem
    ASSERT(_vector_spacing == 0 || SIZEOF(.vector_59) <= (_vector_spacing << 3), "function
```

```
at exception vector 59 too large")
   vector_60 _ebase_address + 0x200 + ((_vector_spacing << 3) * 60) :
    {
      KEEP(*(.vector_60))
    } kseg0_program_mem
    ASSERT(_vector_spacing == 0 || SIZEOF(.vector_60) <= (_vector_spacing << 3), "function
at exception vector 60 too large")
   vector_61 _ebase_address + 0x200 + ((_vector_spacing << 3) * 61) :
    {
      KEEP(*(.vector_61))
    } kseg0_program_mem
    ASSERT(_vector_spacing == 0 || SIZEOF(.vector_61) <= (_vector_spacing << 3), "function
at exception vector 61 too large")
   vector_62 _ebase_address + 0x200 + ((_vector_spacing << 3) * 62) :
    {
      KEEP(*(.vector_62))
    } kseg0_program_mem
    ASSERT(_vector_spacing == 0 || SIZEOF(.vector_62) <= (_vector_spacing << 3), "function
at exception vector 62 too large")
   vector_63 _ebase_address + 0x200 + ((_vector_spacing << 3) * 63) :
    {
      KEEP(*(.vector_63))
    } kseg0_program_mem
    ASSERT(_vector_spacing == 0 || SIZEOF(.vector_63) <= (_vector_spacing << 3), "function
at exception vector 63 too large")
   vector_64 _ebase_address + 0x200 + ((_vector_spacing << 3) * 64) :
    {
      KEEP(*(.vector_64))
    } kseg0_program_mem
    ASSERT(_vector_spacing == 0 || SIZEOF(.vector_64) <= (_vector_spacing << 3), "function
at exception vector 64 too large")
   vector_65 _ebase_address + 0x200 + ((_vector_spacing << 3) * 65) :
    {
      KEEP(*(.vector_65))
    } kseg0_program_mem
    ASSERT(_vector_spacing == 0 || SIZEOF(.vector_65) <= (_vector_spacing << 3), "function
at exception vector 65 too large")
   vector_66 _ebase_address + 0x200 + ((_vector_spacing << 3) * 66) :
    {
      KEEP(*(.vector_66))
    } kseg0_program_mem
    ASSERT(_vector_spacing == 0 || SIZEOF(.vector_66) <= (_vector_spacing << 3), "function
at exception vector 66 too large")
   vector_67 _ebase_address + 0x200 + ((_vector_spacing << 3) * 67) :
    {
      KEEP(*(.vector_67))
    } kseg0_program_mem
    ASSERT(_vector_spacing == 0 || SIZEOF(.vector_67) <= (_vector_spacing << 3), "function
at exception vector 67 too large")
```

```
   vector_68 _ebase_address + 0x200 + ((_vector_spacing << 3) * 68) :
    {
      KEEP(*(.vector_68))
    } kseg0_program_mem
    ASSERT(_vector_spacing == 0 || SIZEOF(.vector_68) <= (_vector_spacing << 3), "function
at exception vector 68 too large")
   vector_69 _ebase_address + 0x200 + ((_vector_spacing << 3) * 69) :
    {
      KEEP(*(.vector_69))
    } kseg0_program_mem
    ASSERT(_vector_spacing == 0 || SIZEOF(.vector_69) <= (_vector_spacing << 3), "function
at exception vector 69 too large")
   vector_70 _ebase_address + 0x200 + ((_vector_spacing << 3) * 70) :
    {
      KEEP(*(.vector_70))
    } kseg0_program_mem
    ASSERT(_vector_spacing == 0 || SIZEOF(.vector_70) <= (_vector_spacing << 3), "function
at exception vector 70 too large")
   vector_71 _ebase_address + 0x200 + ((_vector_spacing << 3) * 71) :
    {
      KEEP(*(.vector_71))
    } kseg0_program_mem
    ASSERT(_vector_spacing == 0 || SIZEOF(.vector_71) <= (_vector_spacing << 3), "function
at exception vector 71 too large")
   vector_72 _ebase_address + 0x200 + ((_vector_spacing << 3) * 72) :
    {
      KEEP(*(.vector_72))
    } kseg0_program_mem
    ASSERT(_vector_spacing == 0 || SIZEOF(.vector_72) <= (_vector_spacing << 3), "function
at exception vector 72 too large")
   vector_73 _ebase_address + 0x200 + ((_vector_spacing << 3) * 73) :
    {
      KEEP(*(.vector_73))
    } kseg0_program_mem
    ASSERT(_vector_spacing == 0 || SIZEOF(.vector_73) <= (_vector_spacing << 3), "function
at exception vector 73 too large")
   vector_74 _ebase_address + 0x200 + ((_vector_spacing << 3) * 74) :
    {
      KEEP(*(.vector_74))
    } kseg0_program_mem
    ASSERT(_vector_spacing == 0 || SIZEOF(.vector_74) <= (_vector_spacing << 3), "function
at exception vector 74 too large")
   vector_75 _ebase_address + 0x200 + ((_vector_spacing << 3) * 75) :
    {
      KEEP(*(.vector_75))
    } kseg0_program_mem
    ASSERT(_vector_spacing == 0 || SIZEOF(.vector_75) <= (_vector_spacing << 3), "function
at exception vector 75 too large")
   vector_76 _ebase_address + 0x200 + ((_vector_spacing << 3) * 76) :
```

```
    {
      KEEP(*(.vector_76))
    } kseg0_program_mem
    ASSERT(_vector_spacing == 0 || SIZEOF(.vector_76) <= (_vector_spacing << 3), "function
at exception vector 76 too large")
   vector_77 _ebase_address + 0x200 + ((_vector_spacing << 3) * 77) :
    {
      KEEP(*(.vector_77))
    } kseg0_program_mem
    ASSERT(_vector_spacing == 0 || SIZEOF(.vector_77) <= (_vector_spacing << 3), "function
at exception vector 77 too large")
   vector_78 _ebase_address + 0x200 + ((_vector_spacing << 3) * 78) :
    {
      KEEP(*(.vector_78))
    } kseg0_program_mem
    ASSERT(_vector_spacing == 0 || SIZEOF(.vector_78) <= (_vector_spacing << 3), "function
at exception vector 78 too large")
   vector_79 _ebase_address + 0x200 + ((_vector_spacing << 3) * 79) :
    {
      KEEP(*(.vector_79))
    } kseg0_program_mem
    ASSERT(_vector_spacing == 0 || SIZEOF(.vector_79) <= (_vector_spacing << 3), "function
at exception vector 79 too large")
   vector_80 _ebase_address + 0x200 + ((_vector_spacing << 3) * 80) :
    {
      KEEP(*(.vector_80))
    } kseg0_program_mem
    ASSERT(_vector_spacing == 0 || SIZEOF(.vector_80) <= (_vector_spacing << 3), "function
at exception vector 80 too large")
   vector_81 _ebase_address + 0x200 + ((_vector_spacing << 3) * 81) :
    {
      KEEP(*(.vector_81))
    } kseg0_program_mem
    ASSERT(_vector_spacing == 0 || SIZEOF(.vector_81) <= (_vector_spacing << 3), "function
at exception vector 81 too large")
   vector_82 _ebase_address + 0x200 + ((_vector_spacing << 3) * 82) :
    {
      KEEP(*(.vector_82))
    } kseg0_program_mem
    ASSERT(_vector_spacing == 0 || SIZEOF(.vector_82) <= (_vector_spacing << 3), "function
at exception vector 82 too large")
   vector_83 _ebase_address + 0x200 + ((_vector_spacing << 3) * 83) :
    {
      KEEP(*(.vector_83))
    } kseg0_program_mem
    ASSERT(_vector_spacing == 0 || SIZEOF(.vector_83) <= (_vector_spacing << 3), "function
at exception vector 83 too large")
   vector_84 _ebase_address + 0x200 + ((_vector_spacing << 3) * 84) :
    {
```

```
      KEEP(*(.vector_84))
    } kseg0_program_mem
    ASSERT(_vector_spacing == 0 || SIZEOF(.vector_84) <= (_vector_spacing << 3), "function
at exception vector 84 too large")
   vector_85 _ebase_address + 0x200 + ((_vector_spacing << 3) * 85) :
    {
      KEEP(*(.vector_85))
    } kseg0_program_mem
    ASSERT(_vector_spacing == 0 || SIZEOF(.vector_85) <= (_vector_spacing << 3), "function
at exception vector 85 too large")
   vector_86 _ebase_address + 0x200 + ((_vector_spacing << 3) * 86) :
    {
      KEEP(*(.vector_86))
    } kseg0_program_mem
    ASSERT(_vector_spacing == 0 || SIZEOF(.vector_86) <= (_vector_spacing << 3), "function
at exception vector 86 too large")
   vector_87 _ebase_address + 0x200 + ((_vector_spacing << 3) * 87) :
    {
      KEEP(*(.vector_87))
    } kseg0_program_mem
    ASSERT(_vector_spacing == 0 || SIZEOF(.vector_87) <= (_vector_spacing << 3), "function
at exception vector 87 too large")
   vector_88 _ebase_address + 0x200 + ((_vector_spacing << 3) * 88) :
    {
      KEEP(*(.vector_88))
    } kseg0_program_mem
    ASSERT(_vector_spacing == 0 || SIZEOF(.vector_88) <= (_vector_spacing << 3), "function
at exception vector 88 too large")
   vector_89 _ebase_address + 0x200 + ((_vector_spacing << 3) * 89) :
    {
      KEEP(*(.vector_89))
    } kseg0_program_mem
    ASSERT(_vector_spacing == 0 || SIZEOF(.vector_89) <= (_vector_spacing << 3), "function
at exception vector 89 too large")
   vector_90 _ebase_address + 0x200 + ((_vector_spacing << 3) * 90) :
    {
      KEEP(*(.vector_90))
    } kseg0_program_mem
    ASSERT(_vector_spacing == 0 || SIZEOF(.vector_90) <= (_vector_spacing << 3), "function
at exception vector 90 too large")
   vector_91 _ebase_address + 0x200 + ((_vector_spacing << 3) * 91) :
    {
      KEEP(*(.vector_91))
    } kseg0_program_mem
    ASSERT(_vector_spacing == 0 || SIZEOF(.vector_91) <= (_vector_spacing << 3), "function
at exception vector 91 too large")
   vector_92 _ebase_address + 0x200 + ((_vector_spacing << 3) * 92) :
    {
      KEEP(*(.vector_92))
```

```
    } kseg0_program_mem
    ASSERT(_vector_spacing == 0 || SIZEOF(.vector_92) <= (_vector_spacing << 3), "function
at exception vector 92 too large")
   vector_93 _ebase_address + 0x200 + ((_vector_spacing << 3) * 93) :
    {
      KEEP(*(.vector_93))
    } kseg0_program_mem
    ASSERT(_vector_spacing == 0 || SIZEOF(.vector_93) <= (_vector_spacing << 3), "function
at exception vector 93 too large")
   vector_94 _ebase_address + 0x200 + ((_vector_spacing << 3) * 94) :
    {
      KEEP(*(.vector_94))
    } kseg0_program_mem
    ASSERT(_vector_spacing == 0 || SIZEOF(.vector_94) <= (_vector_spacing << 3), "function
at exception vector 94 too large")
   vector_95 _ebase_address + 0x200 + ((_vector_spacing << 3) * 95) :
    {
      KEEP(*(.vector_95))
    } kseg0_program_mem
    ASSERT(_vector_spacing == 0 || SIZEOF(.vector_95) <= (_vector_spacing << 3), "function
at exception vector 95 too large")
   vector_96 _ebase_address + 0x200 + ((_vector_spacing << 3) * 96) :
    {
      KEEP(*(.vector_96))
    } kseg0_program_mem
    ASSERT(_vector_spacing == 0 || SIZEOF(.vector_96) <= (_vector_spacing << 3), "function
at exception vector 96 too large")
   vector_97 _ebase_address + 0x200 + ((_vector_spacing << 3) * 97) :
    {
      KEEP(*(.vector_97))
    } kseg0_program_mem
    ASSERT(_vector_spacing == 0 || SIZEOF(.vector_97) <= (_vector_spacing << 3), "function
at exception vector 97 too large")
   vector_98 _ebase_address + 0x200 + ((_vector_spacing << 3) * 98) :
    {
      KEEP(*(.vector_98))
    } kseg0_program_mem
    ASSERT(_vector_spacing == 0 || SIZEOF(.vector_98) <= (_vector_spacing << 3), "function
at exception vector 98 too large")
   vector_99 _ebase_address + 0x200 + ((_vector_spacing << 3) * 99) :
    {
      KEEP(*(.vector_99))
    } kseg0_program_mem
    ASSERT(_vector_spacing == 0 || SIZEOF(.vector_99) <= (_vector_spacing << 3), "function
at exception vector 99 too large")
   vector_100 _ebase_address + 0x200 + ((_vector_spacing << 3) * 100) :
    {
      KEEP(*(.vector_100))
    } kseg0_program_mem
```

```
    ASSERT(_vector_spacing == 0 || SIZEOF(.vector_100) <= (_vector_spacing << 3),
"function at exception vector 100 too large")
   vector_101 _ebase_address + 0x200 + ((_vector_spacing << 3) * 101) :
    {
      KEEP(*(.vector_101))
    } >kseg0_program_mem
    ASSERT(_vector_spacing == 0 || SIZEOF(.vector_101) <= (_vector_spacing << 3),
"function at exception vector 101 too large")

  /*    Starting with C32 v2.00, the startup code is in the .reset.startup section.
      *    Keep this here for backwards compatibility.
      */

  .startup ORIGIN(kseg0_boot_mem) :
    {
    KEEP(*(.startup))
    } >kseg0_boot_mem
  /* Code Sections - Note that input sections *(.text) and *(.text.*)
    ** are not mapped here. Starting in C32 v2.00, the best-fit allocator
    ** locates them, so that .text may flow around absolute sections
    ** as needed.
    */
  .text :
    {
       *(.stub .gnu.linkonce.t.*)
    KEEP (*(.text.*personality*))
       *(.mips16.fn.*)
       *(.mips16.call.*)
       *(.gnu.warning)
        ALIGN(4) ;
    }>kseg0_program_mem
  /* Global-namespace object initialization */
  .init    :
    {
    KEEP (*crti.o(.init))
    KEEP (*crtbegin.o(.init))
    KEEP (*(EXCLUDE_FILE (*crtend.o *crtend?.o *crtn.o ).init))
    KEEP (*crtend.o(.init))
    KEEP (*crtn.o(.init))
        ALIGN(4) ;
    }>kseg0_program_mem
  .fini    :
    {
    KEEP (*(.fini))
        ALIGN(4) ;
    }>kseg0_program_mem
  .preinit_array    :
    {
    PROVIDE_HIDDEN (__preinit_array_start = .);
```

```
  KEEP (*(.preinit_array))
  PROVIDE_HIDDEN (__preinit_array_end = .);
      ALIGN(4) ;
  }>kseg0_program_mem
.init_array    :
  {
  PROVIDE_HIDDEN (__init_array_start = .);
  KEEP (*(SORT(.init_array.*)))
  KEEP (*(.init_array))
  PROVIDE_HIDDEN (__init_array_end = .);
      ALIGN(4) ;
  }>kseg0_program_mem
.fini_array    :
  {
  PROVIDE_HIDDEN (__fini_array_start = .);
  KEEP (*(SORT(.fini_array.*)))
  KEEP (*(.fini_array))
  PROVIDE_HIDDEN (__fini_array_end = .);
      ALIGN(4) ;
  }>kseg0_program_mem
.ctors    :
  {
  /* XC32 uses crtbegin.o to find the start of
            the constructors, so we make sure it is
            first.    Because this is a wildcard, it
            doesn't matter if the user does not
            actually link against crtbegin.o; the
            linker won't look for a file to match a
            wildcard.    The wildcard also means that it
            doesn't matter which directory crtbegin.o
            is in.    */
  KEEP (*crtbegin.o(.ctors))
  KEEP (*crtbegin?.o(.ctors))
  /* We don't want to include the .ctor section from
            the crtend.o file until after the sorted ctors.
            The .ctor section from the crtend file contains the
            end of ctors marker and it must be last */
  KEEP (*(EXCLUDE_FILE (*crtend.o *crtend?.o ) .ctors))
  KEEP (*(SORT(.ctors.*)))
  KEEP (*(.ctors))
      ALIGN(4) ;
  }>kseg0_program_mem
.dtors    :
  {
  KEEP (*crtbegin.o(.dtors))
  KEEP (*crtbegin?.o(.dtors))
  KEEP (*(EXCLUDE_FILE (*crtend.o *crtend?.o ) .dtors))
  KEEP (*(SORT(.dtors.*)))
  KEEP (*(.dtors))
```

```
      ALIGN(4) ;
  }>kseg0_program_mem
/* Read-only sections */
.rodata    :
  {
      *(.gnu.linkonce.r.*)
      *(.rodata1)
      ALIGN(4) ;
  }>kseg0_program_mem
/*
    * Small initialized constant global and static data can be placed in the
    * .sdata2 section.    This is different from .sdata, which contains small
    * initialized non-constant global and static data.
    */
.sdata2 ALIGN(4) :
  {
     *(.sdata2 .sdata2.* .gnu.linkonce.s2.*)
      ALIGN(4) ;
  }>kseg0_program_mem
/*
    * Uninitialized constant global and static data (i.e., variables which will
    * always be zero).    Again, this is different from .sbss, which contains
    * small non-initialized, non-constant global and static data.
    */
.sbss2 ALIGN(4) :
  {
     *(.sbss2 .sbss2.* .gnu.linkonce.sb2.*)
      ALIGN(4) ;
  }>kseg0_program_mem
.eh_frame_hdr    :
  {
     *(.eh_frame_hdr)
  }>kseg0_program_mem
  . = ALIGN(4) ;
 eh_frame    : ONLY_IF_RO
{
  KEEP (*(.eh_frame))
  }>kseg0_program_mem
  . = ALIGN(4) ;
 gcc_except_table    : ONLY_IF_RO
{
     *(.gcc_except_table .gcc_except_table.*)
  }>kseg0_program_mem
  . = ALIGN(4) ;
 dbg_data (NOLOAD) :
  {
      . DEFINED (_DEBUGGER) ? 0x200 : 0x0);
  }>kseg0_data_mem
.jcr    :
```

```
  {
  KEEP (*(.jcr))
      ALIGN(4) ;
  }>kseg0_data_mem
.eh_frame    : ONLY_IF_RW
{
  KEEP (*(.eh_frame))
  }>kseg0_data_mem
  . = ALIGN(4) ;
 gcc_except_table    : ONLY_IF_RW
{
     *(.gcc_except_table .gcc_except_table.*)
  }>kseg0_data_mem
  . = ALIGN(4) ;
/* Persistent data - Use the new C 'persistent' attribute instead. */
.persist    :
  {
  _persist_begin = .;
     *(.persist .persist.*)
     *(.pbss .pbss.*)
      ALIGN(4) ;
  _persist_end = .;
  }>kseg0_data_mem
/*
    * Note that input sections named .data* are no longer mapped here.
    * Starting in C32 v2.00, the best-fit allocator locates them, so
    * that they may flow around absolute sections as needed.
    */
.data    :
  {
      *(.gnu.linkonce.d.*)
  SORT(CONSTRUCTORS)
     *(.data1)
      ALIGN(4) ;
  }>kseg0_data_mem
. = .;
_gp = ALIGN(16) + 0x7ff0;
 got ALIGN(4) :
  {
     *(.got.plt) *(.got)
      ALIGN(4) ;
  }>kseg0_data_mem /* AT>kseg0_program_mem */
  /*
    * Note that "small" data sections are still mapped in the linker
    * script. This ensures that they are grouped together for
    * gp-relative addressing. Absolute sections are allocated after
    * the "small" data sections so small data cannot flow around them.
    */
  /*
```

```
    * We want the small data sections together, so single-instruction offsets
    * can access them all, and initialized data all before uninitialized, so
    * we can shorten the on-disk segment size.
    */
  .sdata ALIGN(4) :
    {
    _sdata_begin = . ;
       *(.sdata .sdata.* .gnu.linkonce.s.*)
        ALIGN(4) ;
    _sdata_end = . ;
    } > kseg0_data_mem
  .lit8           :
    {
       *(.lit8)
    } > kseg0_data_mem
  .lit4           :
    {
       *(.lit4)
    } > kseg0_data_mem
  . = ALIGN (4) ;
  _data_end = . ;
  _bss_begin = . ;
   sbss ALIGN(4) :
    {
    _sbss_begin = . ;
       *(dynsbss)
       *(.sbss .sbss.* .gnu.linkonce.sb.*)
       *(scommon)
    _sbss_end = . ;
        ALIGN(4) ;
    } > kseg0_data_mem
  /*
    * Align here to ensure that the .bss section occupies space up to
    * _end.    Align after .bss to ensure correct alignment even if the
    * .bss section disappears because there are no input sections.
    *
    * Note that input sections named .bss* are no longer mapped here.
    * Starting in C32 v2.00, the best-fit allocator locates them, so
    * that they may flow around absolute sections as needed.
    *
    */
  .bss      :
    {
       *(dynbss)
      *(COMMON)
    /* Align here to ensure that the .bss section occupies space up to
          _end.    Align after .bss to ensure correct alignment even if the
          .bss section disappears because there are no input sections. */
    . = ALIGN(. != 0 ? 4 : 1);
```

```
  }>kseg0_data_mem
. = ALIGN(4) ;
_end = . ;
_bss_end = . ;
/* Starting with C32 v2.00, the heap and stack are dynamically
    * allocated by the linker.
    */
  /*
    * RAM functions go at the end of our stack and heap allocation.
    * Alignment of 2K required by the boundary register (BMXDKPBA).
    *
    * RAM functions are now allocated by the linker. The linker generates
    * _ramfunc_begin and _bmxdkpba_address symbols depending on the
    * location of RAM functions.
    */
_bmxdudba_address = LENGTH(kseg0_data_mem) ;
_bmxdupba_address = LENGTH(kseg0_data_mem) ;
  /* The .pdr section belongs in the absolute section */
  /DISCARD/ : { *(.pdr) }
 gptab.sdata : { *(.gptab.data) *(.gptab.sdata) }
 gptab.sbss : { *(.gptab.bss) *(.gptab.sbss) }
 mdebug.abi32 0 : { KEEP(*(.mdebug.abi32)) }
 mdebug.abiN32 0 : { KEEP(*(.mdebug.abiN32)) }
 mdebug.abi64 0 : { KEEP(*(.mdebug.abi64)) }
 mdebug.abi064 0 : { KEEP(*(.mdebug.abi064)) }
 mdebug.eabi32 0 : { KEEP(*(.mdebug.eabi32)) }
 mdebug.eabi64 0 : { KEEP(*(.mdebug.eabi64)) }
 gcc_compiled_long32 : { KEEP(*(.gcc_compiled_long32)) }
 gcc_compiled_long64 : { KEEP(*(.gcc_compiled_long64)) }
/* Stabs debugging sections.    */
.stab            : { *(.stab) }
 stabstr         : { *(.stabstr) }
 stab.excl       : {0*(.stab.excl) }
 stab.exclstr    0 { *(.stab.exclstr) }
 stab.index      : 0 *(.stab.index) }
 stab.indexstr 0 : { *(.stab.indexstr) }
 comment         : { *(.comment) }
/* DWARF debug sections.
        Symbols in the DWARF debugging sections are relative to the beginning
        of the section so we begin them at 0.    */
  /* DWARF 1 */
.debug           : { *.elf(.debug) *(.debug) }
 line            : { *.elf(.line) *(.line) }
/* GNU DWARF 1 extensions */
.debug_srcinfo    0 { *.elf(.debug_srcinfo) *(.debug_srcinfo) }
 debug_sfnames    0 { *.elf(.debug_sfnames) *(.debug_sfnames) }
/* DWARF 1.1 and DWARF 2 */
.debug_aranges    0 { *.elf(.debug_aranges) *(.debug_aranges) }
 debug_pubnames 0 : { *.elf(.debug_pubnames) *(.debug_pubnames) }
```

```
  /* DWARF 2 */
  .debug_info       : {0*.elf(.debug_info .gnu.linkonce.wi.*) *(.debug_info .gnu.linkonce
.wi.*) }
   debug_abbrev     :0{ *.elf(.debug_abbrev) *(.debug_abbrev) }
   debug_line       : {0*.elf(.debug_line) *(.debug_line) }
   debug_frame      : 0 *.elf(.debug_frame) *(.debug_frame) }
   debug_str        : { 0.elf(.debug_str) *(.debug_str) }
   debug_loc        : { 0.elf(.debug_loc) *(.debug_loc) }
   debug_macinfo    0 { *.elf(.debug_macinfo) *(.debug_macinfo) }
  /* SGI/MIPS DWARF 2 extensions */
  .debug_weaknames 0 : { *.elf(.debug_weaknames) *(.debug_weaknames) }
   debug_funcnames 0 : { *.elf(.debug_funcnames) *(.debug_funcnames) }
   debug_typenames 0 : { *.elf(.debug_typenames) *(.debug_typenames) }
   debug_varnames    0 { *.elf(.debug_varnames) *(.debug_varnames) }
   debug_pubtypes 0 : { *.elf(.debug_pubtypes) *(.debug_pubtypes) }
   debug_ranges     :0{ *.elf(.debug_ranges) *(.debug_ranges) }
   DISCARD/ : { *(.rel.dyn) }
   gnu.attributes 0 : { KEEP (*(.gnu.attributes)) }
   DISCARD/ : { *(.note.GNU-stack) }
   DISCARD/ : { *(.note.GNU-stack) *(.gnu_debuglink) *(.gnu.lto_*) *(.discard) }
}
```

Application linker script

```
[code]/* Default linker script, for normal executables */
OUTPUT_FORMAT("elf32-tradlittlemips")
OUTPUT_ARCH(pic32mx)
ENTRY(_reset)
/*
 * Provide for a minimum stack and heap size
 * - _min_stack_size - represents the minimum space that must be made
 *                     available for the stack.  Can be overridden from
 *                     the command line using the linker's --defsym option.
 * - _min_heap_size  - represents the minimum space that must be made
 *                     available for the heap.  Can be overridden from
 *                     the command line using the linker's --defsym option.
 */
EXTERN (_min_stack_size _min_heap_size)

/*****************************************************************************
 * Processor-specific object file.  Contains SFR definitions.
 *****************************************************************************/
INPUT("processor.o")

/*****************************************************************************
 * For interrupt vector handling
 *****************************************************************************/
PROVIDE(_vector_spacing = 0x00000001);
_ebase_address = 0x9D005000;
```

```
/*************************************************************************
 * Memory Address Equates
 * _RESET_ADDR      -- Reset Vector
 * _BEV_EXCPT_ADDR  -- Boot exception Vector
 * _DBG_EXCPT_ADDR  -- In-circuit Debugging Exception Vector
 * _DBG_CODE_ADDR   -- In-circuit Debug Executive address
 * _DBG_CODE_SIZE   -- In-circuit Debug Executive size
 * _SIMPLE_TLB_REFILL_EXCPT_ADDR  -- Simple TLB-Refill Exception Vector
 * _GEN_EXCPT_ADDR  -- General Exception Vector
 *************************************************************************/
_RESET_ADDR                  = 0x9D006000;
_BEV_EXCPT_ADDR              = 0x9D000380;
_DBG_EXCPT_ADDR              = 0x9D000480;
_DBG_CODE_ADDR              = 0x9D004000;
_DBG_CODE_SIZE               = 0x760;
_SIMPLE_TLB_REFILL_EXCPT_ADDR  = _ebase_address + 0;
_GEN_EXCPT_ADDR              = _ebase_address + 0x180;

/*************************************************************************
 * Memory Regions
 *
 * Memory regions without attributes cannot be used for orphaned sections.
 * Only sections specifically assigned to these regions can be allocated
 * into these regions.
 *************************************************************************/
MEMORY
{
  kseg0_program_mem    (rx)  : ORIGIN = 0x9D007000, LENGTH = 0x20000   /* All C Files
will be located here */
  kseg0_boot_mem             : ORIGIN = 0x9D027760, LENGTH = 0x0     /* This memory
region is dummy */
  kseg1_boot_mem             : ORIGIN = 0x9D027760, LENGTH = 0x490   /* C Startup code */
  kseg0_data_mem       (w!x)  : ORIGIN = 0x80000000, LENGTH = 0x8000
  sfrs                       : ORIGIN = 0xBF800000, LENGTH = 0x100000
  debug_exec_mem             : ORIGIN = 0x9D004000, LENGTH = 0x760
}

/*************************************************************************
 * Configuration-word sections
 *************************************************************************/
SECTIONS
{
  /* Boot Sections */
  .reset _RESET_ADDR :
  {
    KEEP(*(.reset))
    KEEP(*(.reset.startup))
  } > kseg1_boot_mem
  .bev_excpt _BEV_EXCPT_ADDR :
```

```
  {
    KEEP(*(.bev_handler))
  } > kseg1_boot_mem
  /* Debug exception vector */
  .dbg_excpt _DBG_EXCPT_ADDR (NOLOAD) :
  {
    . += (DEFINED (_DEBUGGER) ? 0x8 : 0x0);
  } > kseg1_boot_mem
  /* Space reserved for the debug executive */
  .dbg_code _DBG_CODE_ADDR (NOLOAD) :
  {
    . += (DEFINED (_DEBUGGER) ? _DBG_CODE_SIZE : 0x0);
  } > debug_exec_mem

  .app_excpt _GEN_EXCPT_ADDR :
  {
    KEEP(*(.gen_handler))
  } > kseg0_program_mem

  .vector_0 _ebase_address + 0x200 + ((_vector_spacing << 3) * 0) :
  {
     KEEP(*(.vector_0))
  } > kseg0_program_mem
  ASSERT (_vector_spacing == 0 || SIZEOF(.vector_0) <= (_vector_spacing << 3), "function
at exception vector 0 too large")
  .vector_1 _ebase_address + 0x200 + ((_vector_spacing << 3) * 1) :
  {
     KEEP(*(.vector_1))
  } > kseg0_program_mem
  ASSERT (_vector_spacing == 0 || SIZEOF(.vector_1) <= (_vector_spacing << 3), "function
at exception vector 1 too large")
  .vector_2 _ebase_address + 0x200 + ((_vector_spacing << 3) * 2) :
  {
     KEEP(*(.vector_2))
  } > kseg0_program_mem
  ASSERT (_vector_spacing == 0 || SIZEOF(.vector_2) <= (_vector_spacing << 3), "function
at exception vector 2 too large")
  .vector_3 _ebase_address + 0x200 + ((_vector_spacing << 3) * 3) :
  {
     KEEP(*(.vector_3))
  } > kseg0_program_mem
  ASSERT (_vector_spacing == 0 || SIZEOF(.vector_3) <= (_vector_spacing << 3), "function
at exception vector 3 too large")
  .vector_4 _ebase_address + 0x200 + ((_vector_spacing << 3) * 4) :
  {
     KEEP(*(.vector_4))
  } > kseg0_program_mem
  ASSERT (_vector_spacing == 0 || SIZEOF(.vector_4) <= (_vector_spacing << 3), "function
at exception vector 4 too large")
```

```
  .vector_5 _ebase_address + 0x200 + ((_vector_spacing << 3) * 5) :
  {
     KEEP(*(.vector_5))
  } > kseg0_program_mem
  ASSERT (_vector_spacing == 0 || SIZEOF(.vector_5) <= (_vector_spacing << 3), "function
at exception vector 5 too large")
  .vector_6 _ebase_address + 0x200 + ((_vector_spacing << 3) * 6) :
  {
     KEEP(*(.vector_6))
  } > kseg0_program_mem
  ASSERT (_vector_spacing == 0 || SIZEOF(.vector_6) <= (_vector_spacing << 3), "function
at exception vector 6 too large")
  .vector_7 _ebase_address + 0x200 + ((_vector_spacing << 3) * 7) :
  {
     KEEP(*(.vector_7))
  } > kseg0_program_mem
  ASSERT (_vector_spacing == 0 || SIZEOF(.vector_7) <= (_vector_spacing << 3), "function
at exception vector 7 too large")
  .vector_8 _ebase_address + 0x200 + ((_vector_spacing << 3) * 8) :
  {
     KEEP(*(.vector_8))
  } > kseg0_program_mem
  ASSERT (_vector_spacing == 0 || SIZEOF(.vector_8) <= (_vector_spacing << 3), "function
at exception vector 8 too large")
  .vector_9 _ebase_address + 0x200 + ((_vector_spacing << 3) * 9) :
  {
     KEEP(*(.vector_9))
  } > kseg0_program_mem
  ASSERT (_vector_spacing == 0 || SIZEOF(.vector_9) <= (_vector_spacing << 3), "function
at exception vector 9 too large")
  .vector_10 _ebase_address + 0x200 + ((_vector_spacing << 3) * 10) :
  {
     KEEP(*(.vector_10))
  } > kseg0_program_mem
  ASSERT (_vector_spacing == 0 || SIZEOF(.vector_10) <= (_vector_spacing << 3), "function
at exception vector 10 too large")
  .vector_11 _ebase_address + 0x200 + ((_vector_spacing << 3) * 11) :
  {
     KEEP(*(.vector_11))
  } > kseg0_program_mem
  ASSERT (_vector_spacing == 0 || SIZEOF(.vector_11) <= (_vector_spacing << 3), "function
at exception vector 11 too large")
  .vector_12 _ebase_address + 0x200 + ((_vector_spacing << 3) * 12) :
  {
     KEEP(*(.vector_12))
  } > kseg0_program_mem
  ASSERT (_vector_spacing == 0 || SIZEOF(.vector_12) <= (_vector_spacing << 3), "function
at exception vector 12 too large")
  .vector_13 _ebase_address + 0x200 + ((_vector_spacing << 3) * 13) :
```

```
  {
    KEEP(*(.vector_13))
  } > kseg0_program_mem
  ASSERT (_vector_spacing == 0 || SIZEOF(.vector_13) <= (_vector_spacing << 3), "function
at exception vector 13 too large")
  .vector_14 _ebase_address + 0x200 + ((_vector_spacing << 3) * 14) :
  {
    KEEP(*(.vector_14))
  } > kseg0_program_mem
  ASSERT (_vector_spacing == 0 || SIZEOF(.vector_14) <= (_vector_spacing << 3), "function
at exception vector 14 too large")
  .vector_15 _ebase_address + 0x200 + ((_vector_spacing << 3) * 15) :
  {
    KEEP(*(.vector_15))
  } > kseg0_program_mem
  ASSERT (_vector_spacing == 0 || SIZEOF(.vector_15) <= (_vector_spacing << 3), "function
at exception vector 15 too large")
  .vector_16 _ebase_address + 0x200 + ((_vector_spacing << 3) * 16) :
  {
    KEEP(*(.vector_16))
  } > kseg0_program_mem
  ASSERT (_vector_spacing == 0 || SIZEOF(.vector_16) <= (_vector_spacing << 3), "function
at exception vector 16 too large")
  .vector_17 _ebase_address + 0x200 + ((_vector_spacing << 3) * 17) :
  {
    KEEP(*(.vector_17))
  } > kseg0_program_mem
  ASSERT (_vector_spacing == 0 || SIZEOF(.vector_17) <= (_vector_spacing << 3), "function
at exception vector 17 too large")
  .vector_18 _ebase_address + 0x200 + ((_vector_spacing << 3) * 18) :
  {
    KEEP(*(.vector_18))
  } > kseg0_program_mem
  ASSERT (_vector_spacing == 0 || SIZEOF(.vector_18) <= (_vector_spacing << 3), "function
at exception vector 18 too large")
  .vector_19 _ebase_address + 0x200 + ((_vector_spacing << 3) * 19) :
  {
    KEEP(*(.vector_19))
  } > kseg0_program_mem
  ASSERT (_vector_spacing == 0 || SIZEOF(.vector_19) <= (_vector_spacing << 3), "function
at exception vector 19 too large")
  .vector_20 _ebase_address + 0x200 + ((_vector_spacing << 3) * 20) :
  {
    KEEP(*(.vector_20))
  } > kseg0_program_mem
  ASSERT (_vector_spacing == 0 || SIZEOF(.vector_20) <= (_vector_spacing << 3), "function
at exception vector 20 too large")
  .vector_21 _ebase_address + 0x200 + ((_vector_spacing << 3) * 21) :
  {
```

```
    KEEP(*(.vector_21))
  } > kseg0_program_mem
  ASSERT (_vector_spacing == 0 || SIZEOF(.vector_21) <= (_vector_spacing << 3), "function
at exception vector 21 too large")
  .vector_22 _ebase_address + 0x200 + ((_vector_spacing << 3) * 22) :
  {
    KEEP(*(.vector_22))
  } > kseg0_program_mem
  ASSERT (_vector_spacing == 0 || SIZEOF(.vector_22) <= (_vector_spacing << 3), "function
at exception vector 22 too large")
  .vector_23 _ebase_address + 0x200 + ((_vector_spacing << 3) * 23) :
  {
    KEEP(*(.vector_23))
  } > kseg0_program_mem
  ASSERT (_vector_spacing == 0 || SIZEOF(.vector_23) <= (_vector_spacing << 3), "function
at exception vector 23 too large")
  .vector_24 _ebase_address + 0x200 + ((_vector_spacing << 3) * 24) :
  {
    KEEP(*(.vector_24))
  } > kseg0_program_mem
  ASSERT (_vector_spacing == 0 || SIZEOF(.vector_24) <= (_vector_spacing << 3), "function
at exception vector 24 too large")
  .vector_25 _ebase_address + 0x200 + ((_vector_spacing << 3) * 25) :
  {
    KEEP(*(.vector_25))
  } > kseg0_program_mem
  ASSERT (_vector_spacing == 0 || SIZEOF(.vector_25) <= (_vector_spacing << 3), "function
at exception vector 25 too large")
  .vector_26 _ebase_address + 0x200 + ((_vector_spacing << 3) * 26) :
  {
    KEEP(*(.vector_26))
  } > kseg0_program_mem
  ASSERT (_vector_spacing == 0 || SIZEOF(.vector_26) <= (_vector_spacing << 3), "function
at exception vector 26 too large")
  .vector_27 _ebase_address + 0x200 + ((_vector_spacing << 3) * 27) :
  {
    KEEP(*(.vector_27))
  } > kseg0_program_mem
  ASSERT (_vector_spacing == 0 || SIZEOF(.vector_27) <= (_vector_spacing << 3), "function
at exception vector 27 too large")
  .vector_28 _ebase_address + 0x200 + ((_vector_spacing << 3) * 28) :
  {
    KEEP(*(.vector_28))
  } > kseg0_program_mem
  ASSERT (_vector_spacing == 0 || SIZEOF(.vector_28) <= (_vector_spacing << 3), "function
at exception vector 28 too large")
  .vector_29 _ebase_address + 0x200 + ((_vector_spacing << 3) * 29) :
  {
    KEEP(*(.vector_29))
```

```
  } > kseg0_program_mem
  ASSERT (_vector_spacing == 0 || SIZEOF(.vector_29) <= (_vector_spacing << 3), "function
at exception vector 29 too large")
  .vector_30 _ebase_address + 0x200 + ((_vector_spacing << 3) * 30) :
  {
     KEEP(*(.vector_30))
  } > kseg0_program_mem
  ASSERT (_vector_spacing == 0 || SIZEOF(.vector_30) <= (_vector_spacing << 3), "function
at exception vector 30 too large")
  .vector_31 _ebase_address + 0x200 + ((_vector_spacing << 3) * 31) :
  {
     KEEP(*(.vector_31))
  } > kseg0_program_mem
  ASSERT (_vector_spacing == 0 || SIZEOF(.vector_31) <= (_vector_spacing << 3), "function
at exception vector 31 too large")
  .vector_32 _ebase_address + 0x200 + ((_vector_spacing << 3) * 32) :
  {
     KEEP(*(.vector_32))
  } > kseg0_program_mem
  ASSERT (_vector_spacing == 0 || SIZEOF(.vector_32) <= (_vector_spacing << 3), "function
at exception vector 32 too large")
  .vector_33 _ebase_address + 0x200 + ((_vector_spacing << 3) * 33) :
  {
     KEEP(*(.vector_33))
  } > kseg0_program_mem
  ASSERT (_vector_spacing == 0 || SIZEOF(.vector_33) <= (_vector_spacing << 3), "function
at exception vector 33 too large")
  .vector_34 _ebase_address + 0x200 + ((_vector_spacing << 3) * 34) :
  {
     KEEP(*(.vector_34))
  } > kseg0_program_mem
  ASSERT (_vector_spacing == 0 || SIZEOF(.vector_34) <= (_vector_spacing << 3), "function
at exception vector 34 too large")
  .vector_35 _ebase_address + 0x200 + ((_vector_spacing << 3) * 35) :
  {
     KEEP(*(.vector_35))
  } > kseg0_program_mem
  ASSERT (_vector_spacing == 0 || SIZEOF(.vector_35) <= (_vector_spacing << 3), "function
at exception vector 35 too large")
  .vector_36 _ebase_address + 0x200 + ((_vector_spacing << 3) * 36) :
  {
     KEEP(*(.vector_36))
  } > kseg0_program_mem
  ASSERT (_vector_spacing == 0 || SIZEOF(.vector_36) <= (_vector_spacing << 3), "function
at exception vector 36 too large")
  .vector_37 _ebase_address + 0x200 + ((_vector_spacing << 3) * 37) :
  {
     KEEP(*(.vector_37))
  } > kseg0_program_mem
```

```
  ASSERT (_vector_spacing == 0 || SIZEOF(.vector_37) <= (_vector_spacing << 3), "function
at exception vector 37 too large")
  .vector_38 _ebase_address + 0x200 + ((_vector_spacing << 3) * 38) :
  {
     KEEP(*(.vector_38))
  } > kseg0_program_mem
  ASSERT (_vector_spacing == 0 || SIZEOF(.vector_38) <= (_vector_spacing << 3), "function
at exception vector 38 too large")
  .vector_39 _ebase_address + 0x200 + ((_vector_spacing << 3) * 39) :
  {
     KEEP(*(.vector_39))
  } > kseg0_program_mem
  ASSERT (_vector_spacing == 0 || SIZEOF(.vector_39) <= (_vector_spacing << 3), "function
at exception vector 39 too large")
  .vector_40 _ebase_address + 0x200 + ((_vector_spacing << 3) * 40) :
  {
     KEEP(*(.vector_40))
  } > kseg0_program_mem
  ASSERT (_vector_spacing == 0 || SIZEOF(.vector_40) <= (_vector_spacing << 3), "function
at exception vector 40 too large")
  .vector_41 _ebase_address + 0x200 + ((_vector_spacing << 3) * 41) :
  {
     KEEP(*(.vector_41))
  } > kseg0_program_mem
  ASSERT (_vector_spacing == 0 || SIZEOF(.vector_41) <= (_vector_spacing << 3), "function
at exception vector 41 too large")
  .vector_42 _ebase_address + 0x200 + ((_vector_spacing << 3) * 42) :
  {
     KEEP(*(.vector_42))
  } > kseg0_program_mem
  ASSERT (_vector_spacing == 0 || SIZEOF(.vector_42) <= (_vector_spacing << 3), "function
at exception vector 42 too large")
  .vector_43 _ebase_address + 0x200 + ((_vector_spacing << 3) * 43) :
  {
     KEEP(*(.vector_43))
  } > kseg0_program_mem
  ASSERT (_vector_spacing == 0 || SIZEOF(.vector_43) <= (_vector_spacing << 3), "function
at exception vector 43 too large")
  .vector_44 _ebase_address + 0x200 + ((_vector_spacing << 3) * 44) :
  {
     KEEP(*(.vector_44))
  } > kseg0_program_mem
  ASSERT (_vector_spacing == 0 || SIZEOF(.vector_44) <= (_vector_spacing << 3), "function
at exception vector 44 too large")
  .vector_45 _ebase_address + 0x200 + ((_vector_spacing << 3) * 45) :
  {
     KEEP(*(.vector_45))
  } > kseg0_program_mem
  ASSERT (_vector_spacing == 0 || SIZEOF(.vector_45) <= (_vector_spacing << 3), "function
```

```
at exception vector 45 too large")
  .vector_46 _ebase_address + 0x200 + ((_vector_spacing << 3) * 46) :
  {
     KEEP(*(.vector_46))
  } > kseg0_program_mem
  ASSERT (_vector_spacing == 0 || SIZEOF(.vector_46) <= (_vector_spacing << 3), "function
at exception vector 46 too large")
  .vector_47 _ebase_address + 0x200 + ((_vector_spacing << 3) * 47) :
  {
     KEEP(*(.vector_47))
  } > kseg0_program_mem
  ASSERT (_vector_spacing == 0 || SIZEOF(.vector_47) <= (_vector_spacing << 3), "function
at exception vector 47 too large")
  .vector_48 _ebase_address + 0x200 + ((_vector_spacing << 3) * 48) :
  {
     KEEP(*(.vector_48))
  } > kseg0_program_mem
  ASSERT (_vector_spacing == 0 || SIZEOF(.vector_48) <= (_vector_spacing << 3), "function
at exception vector 48 too large")
  .vector_49 _ebase_address + 0x200 + ((_vector_spacing << 3) * 49) :
  {
     KEEP(*(.vector_49))
  } > kseg0_program_mem
  ASSERT (_vector_spacing == 0 || SIZEOF(.vector_49) <= (_vector_spacing << 3), "function
at exception vector 49 too large")
  .vector_50 _ebase_address + 0x200 + ((_vector_spacing << 3) * 50) :
  {
     KEEP(*(.vector_50))
  } > kseg0_program_mem
  ASSERT (_vector_spacing == 0 || SIZEOF(.vector_50) <= (_vector_spacing << 3), "function
at exception vector 50 too large")
  .vector_51 _ebase_address + 0x200 + ((_vector_spacing << 3) * 51) :
  {
     KEEP(*(.vector_51))
  } > kseg0_program_mem
  ASSERT (_vector_spacing == 0 || SIZEOF(.vector_51) <= (_vector_spacing << 3), "function
at exception vector 51 too large")
  .vector_52 _ebase_address + 0x200 + ((_vector_spacing << 3) * 52) :
  {
     KEEP(*(.vector_52))
  } > kseg0_program_mem
  ASSERT (_vector_spacing == 0 || SIZEOF(.vector_52) <= (_vector_spacing << 3), "function
at exception vector 52 too large")
  .vector_53 _ebase_address + 0x200 + ((_vector_spacing << 3) * 53) :
  {
     KEEP(*(.vector_53))
  } > kseg0_program_mem
  ASSERT (_vector_spacing == 0 || SIZEOF(.vector_53) <= (_vector_spacing << 3), "function
at exception vector 53 too large")
```

```
  .vector_54 _ebase_address + 0x200 + ((_vector_spacing << 3) * 54) :
  {
     KEEP(*(.vector_54))
  } > kseg0_program_mem
  ASSERT (_vector_spacing == 0 || SIZEOF(.vector_54) <= (_vector_spacing << 3), "function
at exception vector 54 too large")
  .vector_55 _ebase_address + 0x200 + ((_vector_spacing << 3) * 55) :
  {
     KEEP(*(.vector_55))
  } > kseg0_program_mem
  ASSERT (_vector_spacing == 0 || SIZEOF(.vector_55) <= (_vector_spacing << 3), "function
at exception vector 55 too large")
  .vector_56 _ebase_address + 0x200 + ((_vector_spacing << 3) * 56) :
  {
     KEEP(*(.vector_56))
  } > kseg0_program_mem
  ASSERT (_vector_spacing == 0 || SIZEOF(.vector_56) <= (_vector_spacing << 3), "function
at exception vector 56 too large")
  .vector_57 _ebase_address + 0x200 + ((_vector_spacing << 3) * 57) :
  {
     KEEP(*(.vector_57))
  } > kseg0_program_mem
  ASSERT (_vector_spacing == 0 || SIZEOF(.vector_57) <= (_vector_spacing << 3), "function
at exception vector 57 too large")
  .vector_58 _ebase_address + 0x200 + ((_vector_spacing << 3) * 58) :
  {
     KEEP(*(.vector_58))
  } > kseg0_program_mem
  ASSERT (_vector_spacing == 0 || SIZEOF(.vector_58) <= (_vector_spacing << 3), "function
at exception vector 58 too large")
  .vector_59 _ebase_address + 0x200 + ((_vector_spacing << 3) * 59) :
  {
     KEEP(*(.vector_59))
  } > kseg0_program_mem
  ASSERT (_vector_spacing == 0 || SIZEOF(.vector_59) <= (_vector_spacing << 3), "function
at exception vector 59 too large")
  .vector_60 _ebase_address + 0x200 + ((_vector_spacing << 3) * 60) :
  {
     KEEP(*(.vector_60))
  } > kseg0_program_mem
  ASSERT (_vector_spacing == 0 || SIZEOF(.vector_60) <= (_vector_spacing << 3), "function
at exception vector 60 too large")
  .vector_61 _ebase_address + 0x200 + ((_vector_spacing << 3) * 61) :
  {
     KEEP(*(.vector_61))
  } > kseg0_program_mem
  ASSERT (_vector_spacing == 0 || SIZEOF(.vector_61) <= (_vector_spacing << 3), "function
at exception vector 61 too large")
  .vector_62 _ebase_address + 0x200 + ((_vector_spacing << 3) * 62) :
```

```
  {
    KEEP(*(.vector_62))
  } > kseg0_program_mem
  ASSERT (_vector_spacing == 0 || SIZEOF(.vector_62) <= (_vector_spacing << 3), "function
at exception vector 62 too large")
  .vector_63 _ebase_address + 0x200 + ((_vector_spacing << 3) * 63) :
  {
    KEEP(*(.vector_63))
  } > kseg0_program_mem
  ASSERT (_vector_spacing == 0 || SIZEOF(.vector_63) <= (_vector_spacing << 3), "function
at exception vector 63 too large")
  .vector_64 _ebase_address + 0x200 + ((_vector_spacing << 3) * 64) :
  {
    KEEP(*(.vector_64))
  } > kseg0_program_mem
  ASSERT (_vector_spacing == 0 || SIZEOF(.vector_64) <= (_vector_spacing << 3), "function
at exception vector 64 too large")
  .vector_65 _ebase_address + 0x200 + ((_vector_spacing << 3) * 65) :
  {
    KEEP(*(.vector_65))
  } > kseg0_program_mem
  ASSERT (_vector_spacing == 0 || SIZEOF(.vector_65) <= (_vector_spacing << 3), "function
at exception vector 65 too large")
  .vector_66 _ebase_address + 0x200 + ((_vector_spacing << 3) * 66) :
  {
    KEEP(*(.vector_66))
  } > kseg0_program_mem
  ASSERT (_vector_spacing == 0 || SIZEOF(.vector_66) <= (_vector_spacing << 3), "function
at exception vector 66 too large")
  .vector_67 _ebase_address + 0x200 + ((_vector_spacing << 3) * 67) :
  {
    KEEP(*(.vector_67))
  } > kseg0_program_mem
  ASSERT (_vector_spacing == 0 || SIZEOF(.vector_67) <= (_vector_spacing << 3), "function
at exception vector 67 too large")
  .vector_68 _ebase_address + 0x200 + ((_vector_spacing << 3) * 68) :
  {
    KEEP(*(.vector_68))
  } > kseg0_program_mem
  ASSERT (_vector_spacing == 0 || SIZEOF(.vector_68) <= (_vector_spacing << 3), "function
at exception vector 68 too large")
  .vector_69 _ebase_address + 0x200 + ((_vector_spacing << 3) * 69) :
  {
    KEEP(*(.vector_69))
  } > kseg0_program_mem
  ASSERT (_vector_spacing == 0 || SIZEOF(.vector_69) <= (_vector_spacing << 3), "function
at exception vector 69 too large")
  .vector_70 _ebase_address + 0x200 + ((_vector_spacing << 3) * 70) :
  {
```

```
    KEEP(*(.vector_70))
  } > kseg0_program_mem
  ASSERT (_vector_spacing == 0 || SIZEOF(.vector_70) <= (_vector_spacing << 3), "function
at exception vector 70 too large")
  .vector_71 _ebase_address + 0x200 + ((_vector_spacing << 3) * 71) :
  {
    KEEP(*(.vector_71))
  } > kseg0_program_mem
  ASSERT (_vector_spacing == 0 || SIZEOF(.vector_71) <= (_vector_spacing << 3), "function
at exception vector 71 too large")
  .vector_72 _ebase_address + 0x200 + ((_vector_spacing << 3) * 72) :
  {
    KEEP(*(.vector_72))
  } > kseg0_program_mem
  ASSERT (_vector_spacing == 0 || SIZEOF(.vector_72) <= (_vector_spacing << 3), "function
at exception vector 72 too large")
  .vector_73 _ebase_address + 0x200 + ((_vector_spacing << 3) * 73) :
  {
    KEEP(*(.vector_73))
  } > kseg0_program_mem
  ASSERT (_vector_spacing == 0 || SIZEOF(.vector_73) <= (_vector_spacing << 3), "function
at exception vector 73 too large")
  .vector_74 _ebase_address + 0x200 + ((_vector_spacing << 3) * 74) :
  {
    KEEP(*(.vector_74))
  } > kseg0_program_mem
  ASSERT (_vector_spacing == 0 || SIZEOF(.vector_74) <= (_vector_spacing << 3), "function
at exception vector 74 too large")
  .vector_75 _ebase_address + 0x200 + ((_vector_spacing << 3) * 75) :
  {
    KEEP(*(.vector_75))
  } > kseg0_program_mem
  ASSERT (_vector_spacing == 0 || SIZEOF(.vector_75) <= (_vector_spacing << 3), "function
at exception vector 75 too large")
  .vector_76 _ebase_address + 0x200 + ((_vector_spacing << 3) * 76) :
  {
    KEEP(*(.vector_76))
  } > kseg0_program_mem
  ASSERT (_vector_spacing == 0 || SIZEOF(.vector_76) <= (_vector_spacing << 3), "function
at exception vector 76 too large")
  .vector_77 _ebase_address + 0x200 + ((_vector_spacing << 3) * 77) :
  {
    KEEP(*(.vector_77))
  } > kseg0_program_mem
  ASSERT (_vector_spacing == 0 || SIZEOF(.vector_77) <= (_vector_spacing << 3), "function
at exception vector 77 too large")
  .vector_78 _ebase_address + 0x200 + ((_vector_spacing << 3) * 78) :
  {
    KEEP(*(.vector_78))
```

```
  } > kseg0_program_mem
  ASSERT (_vector_spacing == 0 || SIZEOF(.vector_78) <= (_vector_spacing << 3), "function
at exception vector 78 too large")
  .vector_79 _ebase_address + 0x200 + ((_vector_spacing << 3) * 79) :
  {
     KEEP(*(.vector_79))
  } > kseg0_program_mem
  ASSERT (_vector_spacing == 0 || SIZEOF(.vector_79) <= (_vector_spacing << 3), "function
at exception vector 79 too large")
  .vector_80 _ebase_address + 0x200 + ((_vector_spacing << 3) * 80) :
  {
     KEEP(*(.vector_80))
  } > kseg0_program_mem
  ASSERT (_vector_spacing == 0 || SIZEOF(.vector_80) <= (_vector_spacing << 3), "function
at exception vector 80 too large")
  .vector_81 _ebase_address + 0x200 + ((_vector_spacing << 3) * 81) :
  {
     KEEP(*(.vector_81))
  } > kseg0_program_mem
  ASSERT (_vector_spacing == 0 || SIZEOF(.vector_81) <= (_vector_spacing << 3), "function
at exception vector 81 too large")
  .vector_82 _ebase_address + 0x200 + ((_vector_spacing << 3) * 82) :
  {
     KEEP(*(.vector_82))
  } > kseg0_program_mem
  ASSERT (_vector_spacing == 0 || SIZEOF(.vector_82) <= (_vector_spacing << 3), "function
at exception vector 82 too large")
  .vector_83 _ebase_address + 0x200 + ((_vector_spacing << 3) * 83) :
  {
     KEEP(*(.vector_83))
  } > kseg0_program_mem
  ASSERT (_vector_spacing == 0 || SIZEOF(.vector_83) <= (_vector_spacing << 3), "function
at exception vector 83 too large")
  .vector_84 _ebase_addre
```

附加檔案:

PIC32MM_Bootloader.X.zip 大小: 336.87 KB; 下載次數: 13