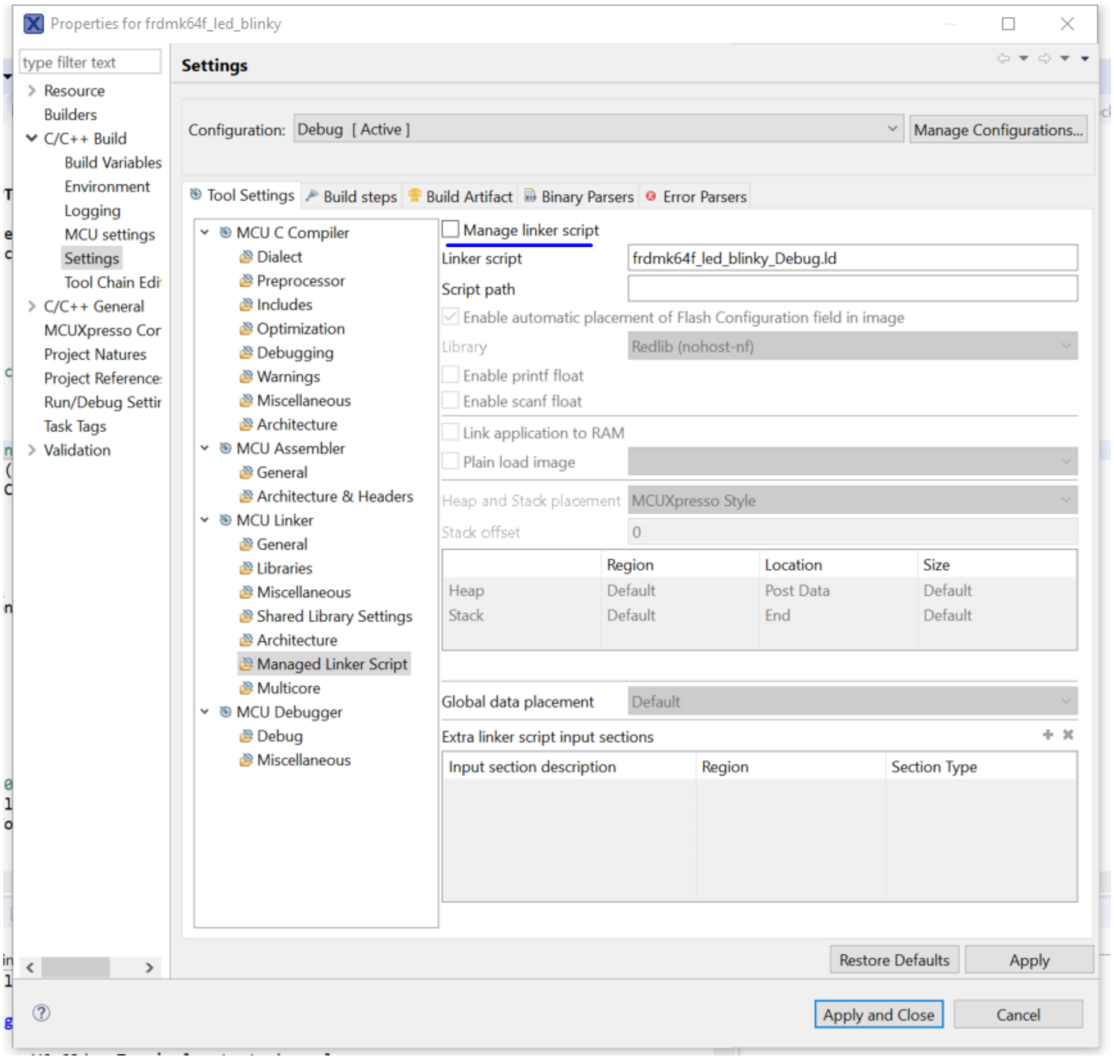
I use frdm-k64 and led demo to show you how to allocate the data at the specified the address.

Open the project and build it(You must build it). Then open the project properties. Cancel the click. Apply and close.



Properties for frdmk64f_led_blinky

Settings

Configuration: Debug [Active]

☐ Manage linker script

Linker script: frdmk64f_led_blinky_Debug.ld

Script path:

☑ Enable automatic placement of Flash Configuration field in image

Library: Redlib (nohost-nf)

☐ Enable printf float

☐ Enable scanf float

☐ Link application to RAM

☐ Plain load image

Heap and Stack placement: MCUXpresso Style

Stack offset: 0

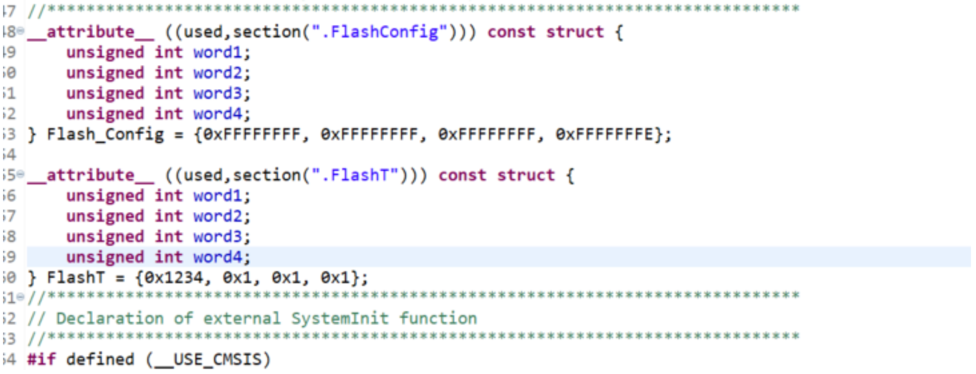
	Region	Location	Size
Heap	Default	Post Data	Default
Stack	Default	End	Default

Global data placement: Default

Extra linker script input sections

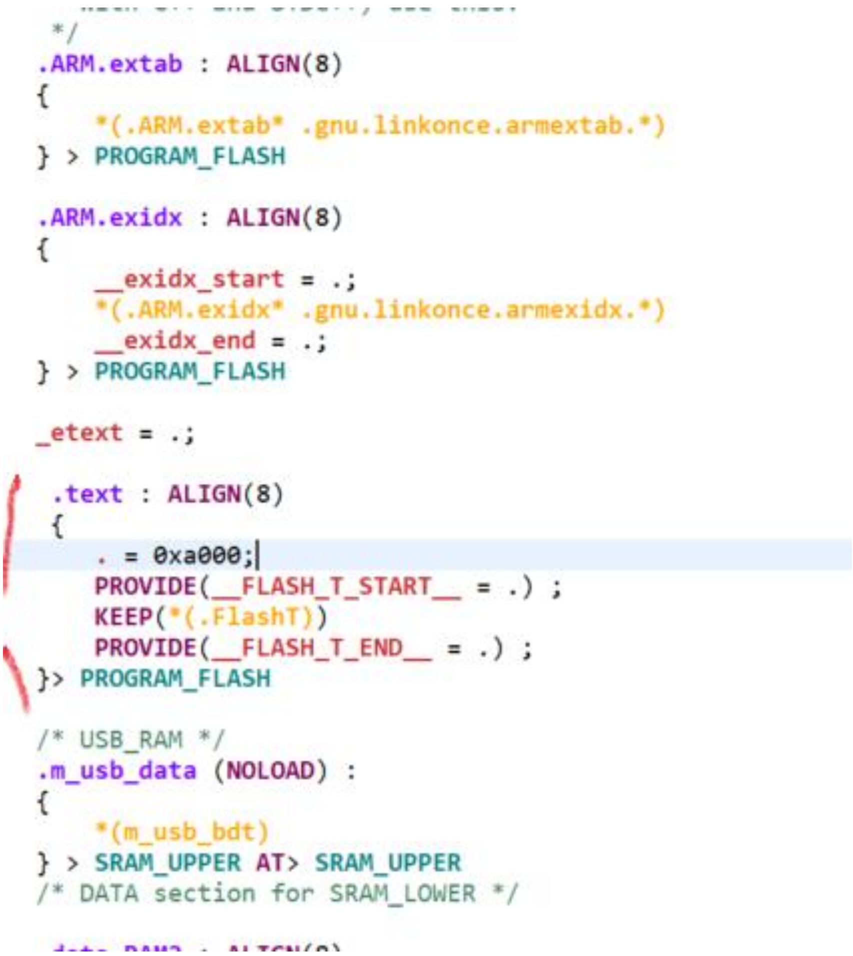
Input section description	Region	Section Type

Open startup_mk64f12.c. Create a structure—FlashT. And place it at section ".FlashT"



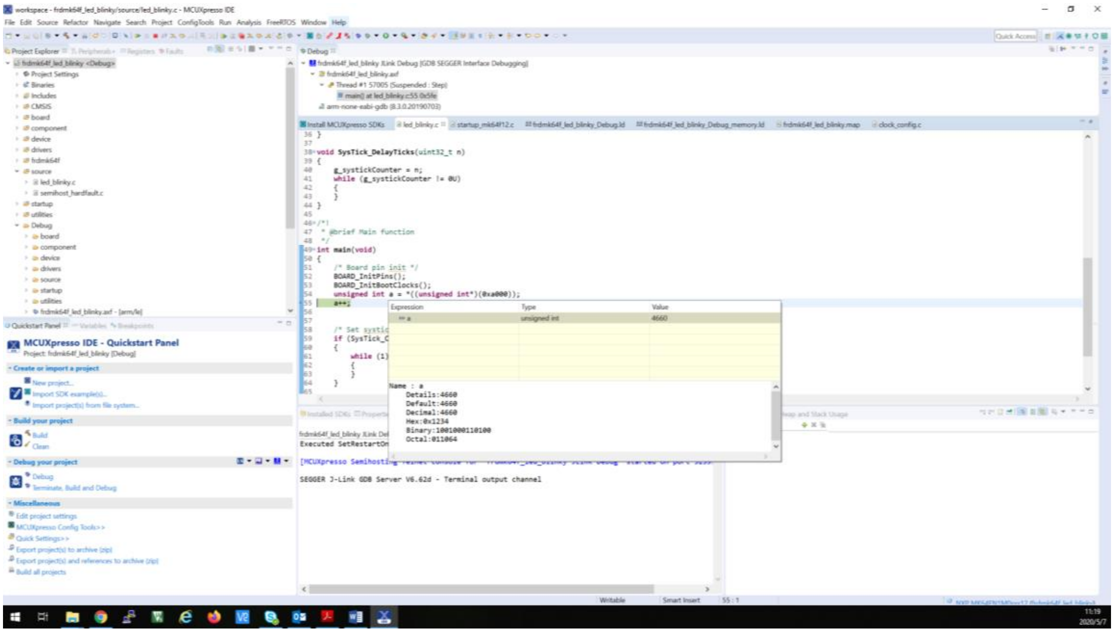
```
//*****************************************************************************
__attribute__ ((used,section(".FlashConfig"))) const struct {
    unsigned int word1;
    unsigned int word2;
    unsigned int word3;
    unsigned int word4;
} Flash_Config = {0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFE};

__attribute__ ((used,section(".FlashT"))) const struct {
    unsigned int word1;
    unsigned int word2;
    unsigned int word3;
    unsigned int word4;
} FlashT = {0x1234, 0x1, 0x1, 0x1};
//*****************************************************************************
// Declaration of external SystemInit function
//*****************************************************************************
#if defined (__USE_CMSIS)
```

Open frdmk64f_led_blinky_Debug.ld. Insert the code



```
  */
.ARM.extab : ALIGN(8)
{
    *(.ARM.extab* .gnu.linkonce.armextab.*)
} > PROGRAM_FLASH

.ARM.exidx : ALIGN(8)
{
    __exidx_start = .;
    *(.ARM.exidx* .gnu.linkonce.armexidx.*)
    __exidx_end = .;
} > PROGRAM_FLASH

_etext = .;

 .text : ALIGN(8)
 {
    . = 0xa000;
    PROVIDE(__FLASH_T_START__ = .) ;
    KEEP(*(.FlashT))
    PROVIDE(__FLASH_T_END__ = .) ;
}> PROGRAM_FLASH

/* USB_RAM */
.m_usb_data (NOLOAD) :
{
    *(m_usb_bdt)
} > SRAM_UPPER AT> SRAM_UPPER
/* DATA section for SRAM_LOWER */

 .data_RAM2 : ALIGN(8)
```

Then my structure will locate at 0xa000. You can see the value 0x1234 at address 0xa000.



workspace - frdmk64f_led_blinky/source/led_blinky.c - MCUXpresso IDE
File Edit Source Refactor Navigate Search Project ConfigTools Run Analysis FreeRTOS Window Help
Project Explorer
frdmk64f_led_blinky <Debug>
Debug
frdmk64f_led_blinky JLink Debug [GDB SEGGER Interface Debugging]
frdmk64f_led_blinky.axf
Thread #1 57005 (Suspended : Step)
main() at led_blinky.c:55 0x5fe
arm-none-eabi-gdb (8.3.0.20190703)
Install MCUXpresso SDKs
led_blinky.c
startup_mk64f12.c
frdmk64f_led_blinky_Debug.ld
frdmk64f_led_blinky_Debug_memory.ld
frdmk64f_led_blinky.map
clock_config.c
MCUXpresso IDE - Quickstart Panel
Writable
Smart Insert
55 : 1

```
}

void SysTick_DelayTicks(uint32_t n)
{
    g_systickCounter = n;
    while (g_systickCounter != 0U)
    {
    }
}

/*!
 * @brief Main function
 */
int main(void)
{
    /* Board pin init */
    BOARD_InitPins();
    BOARD_InitBootClocks();
    unsigned int a = *((unsigned int*)(0xa000));
    a++;
```

Expression	Type	Value
a	unsigned int	4660

```
Name : a
    Details:4660
    Default:4660
    Decimal:4660
    Hex:0x1234
    Binary:1001000110100
    Octal:011064
```

```
Executed SetRestartOn

SEGGER J-Link GDB Server V6.62d - Terminal output channel
```