

ThreadX最小移植及tx_initialize_low_level.s的简要分析

 blog.csdn.net/mdzz6666/article/details/116752547

写在开头

本篇文章的知识大多源于安富莱_STM32-V6开发板ThreadX内核教程（Vo.4）.pdf，有兴趣的同学可以前往讨论学习。

<http://www.armbbs.cn/forum.php?mod=viewthread&tid=99514>

不同于硬汉哥的移植那样复杂，本篇文章仅进行了最简洁的ThreadX移植，类似于单片机最小系统，未作多余的开发。

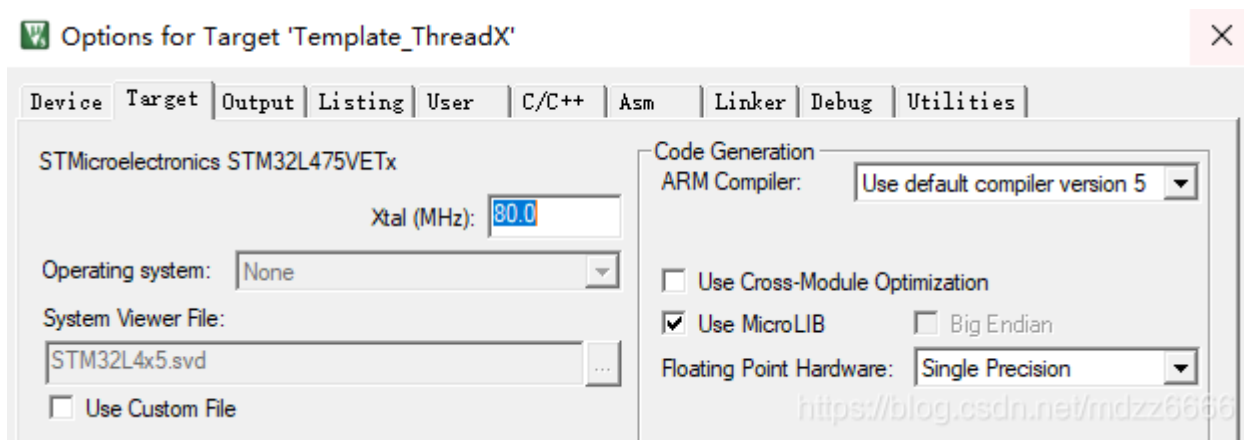
平台介绍

硬件平台：STM32L475VET6（M4内核的芯片应该都适用）；

ThreadX版本：6.1.3；

IDE：KEIL5 v5.31.0.0版本、STM32CubeMX；

ARM编译器：AC5



移植

准备一个简单的裸机程序

使用STM32CubeMX生成一个MDK-ARM平台的LED工程。

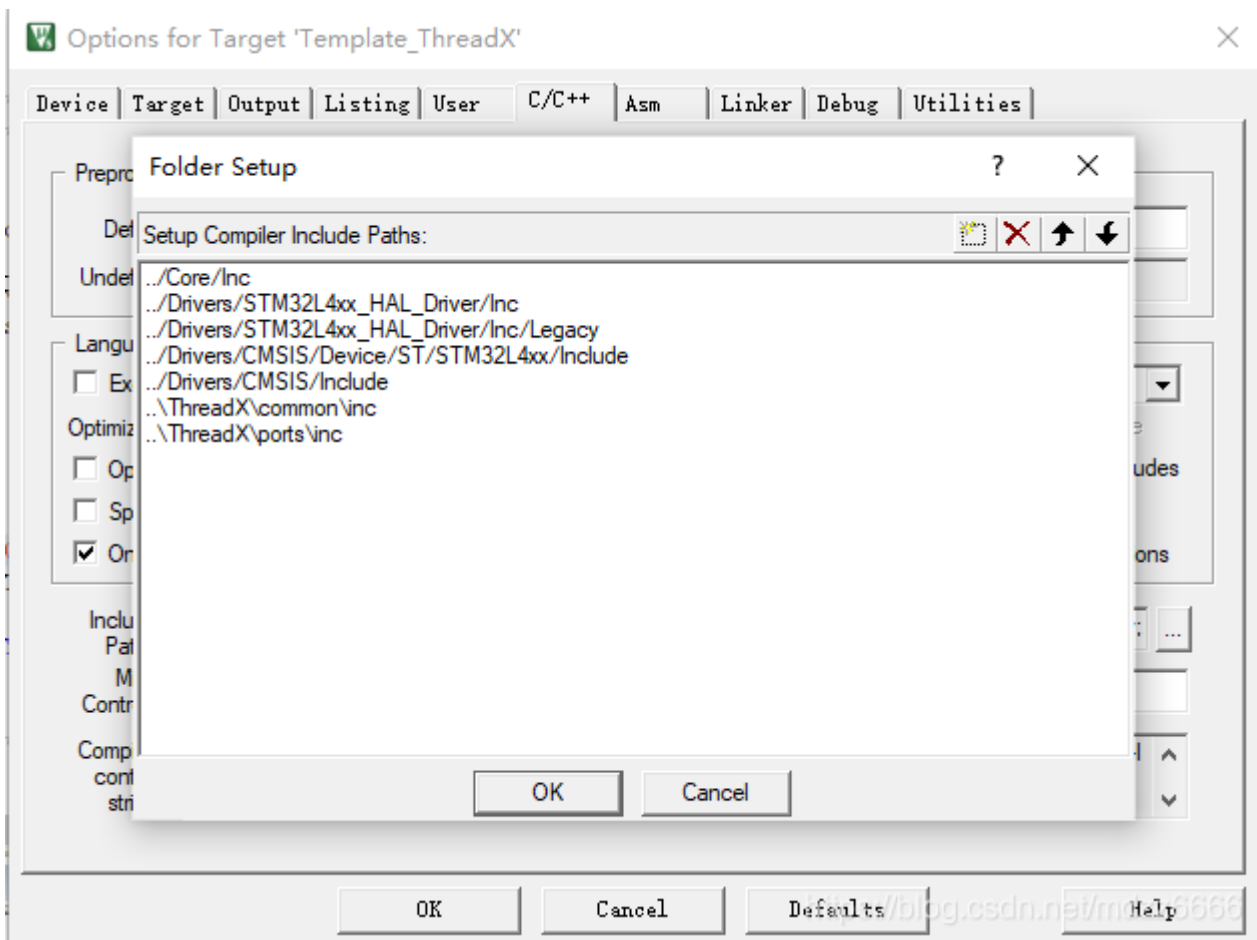
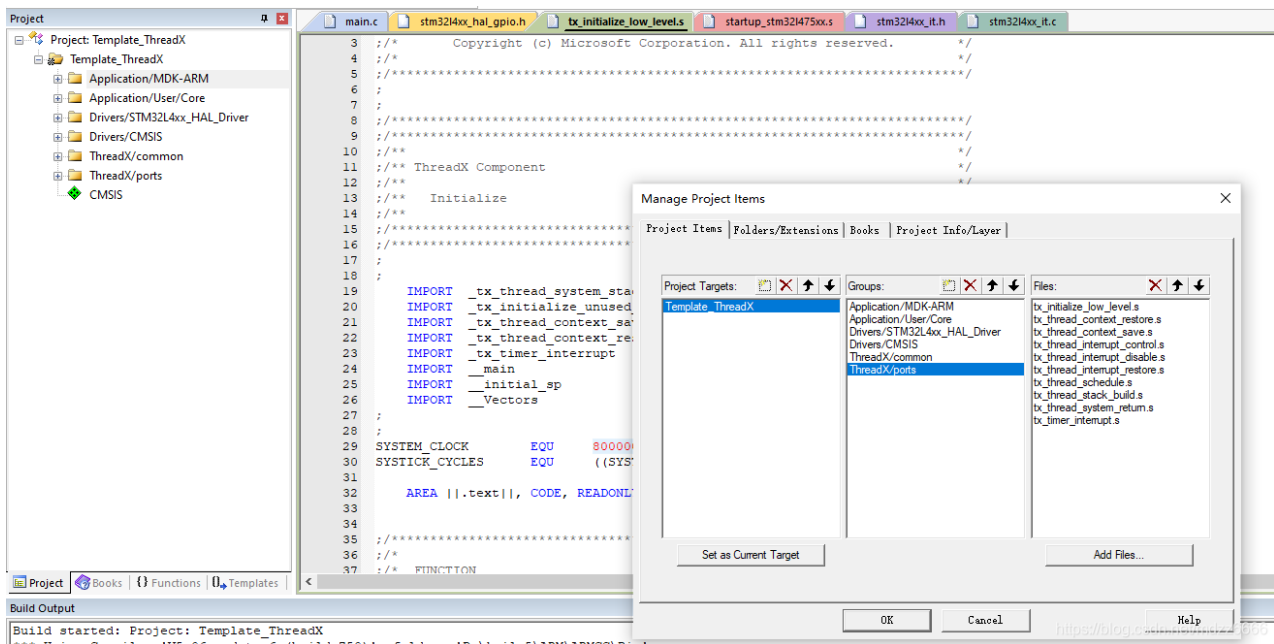
复制ThreadX源码到LED工程文件夹

将ThreadX源码下的common文件夹，ports>cortex_m4>ac5文件夹拷贝到LED工程文件夹下。

注：将ports>cortex_m4>ac5>example_build文件夹下的tx_initialize_low_level.s文件拷贝到ports>cortex_m4>ac5>src文件夹下，方便后期在工程中添加文件。详细步骤可参考安安富莱_STM32-V6开发板ThreadX内核教程（Vo.4）.pdf第4.4.1~4.4.3章节。

在LED工程中添加ThreadX文件

在LED工程中添加common和port文件、设置好头文件路径。

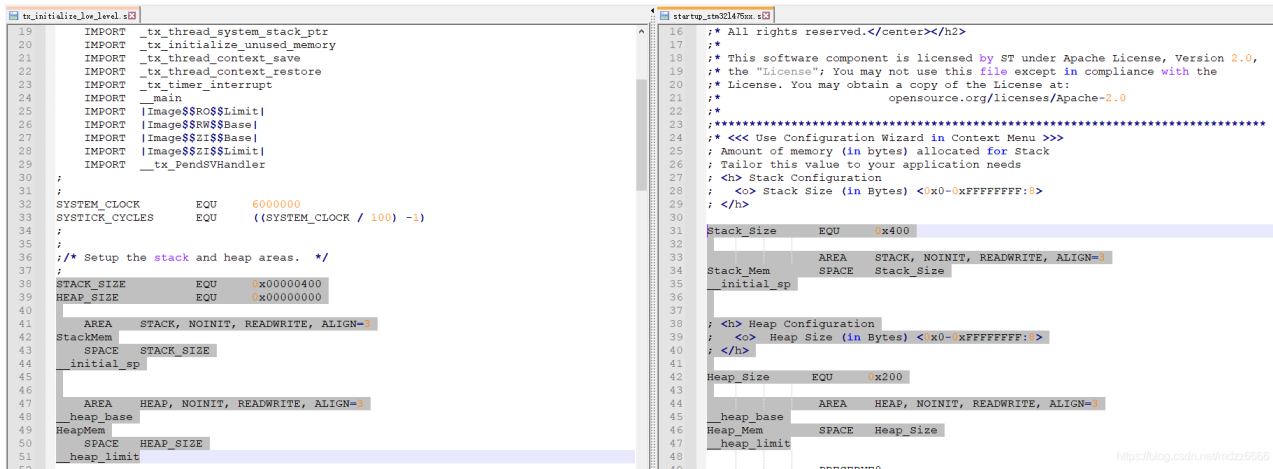


修改tx_initialize_low_level.s文件

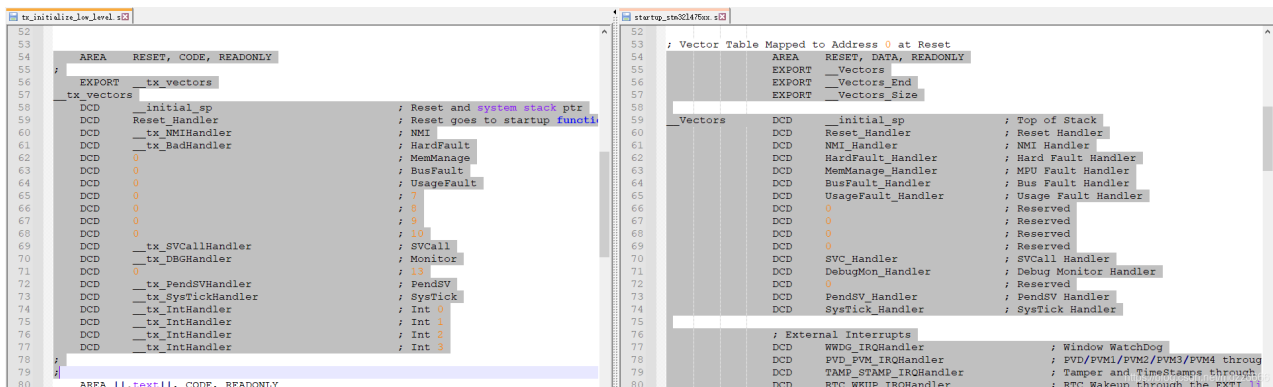
修改思路：tx_initialize_low_level.s是ThreadX提供启动文件，其满足ThreadX需求却不满足mcu需求。而LED工程本身自带启动文件startup_stm32l475xx.s又不满足ThreadX的需求。所以我们需要将这两个文件结合起来使用，针对两个文件交集的部分，我们以LED工程自带的启动文件为准。

tx_initialize_low_level.s与startup_stm32l475xx.s的简单分析

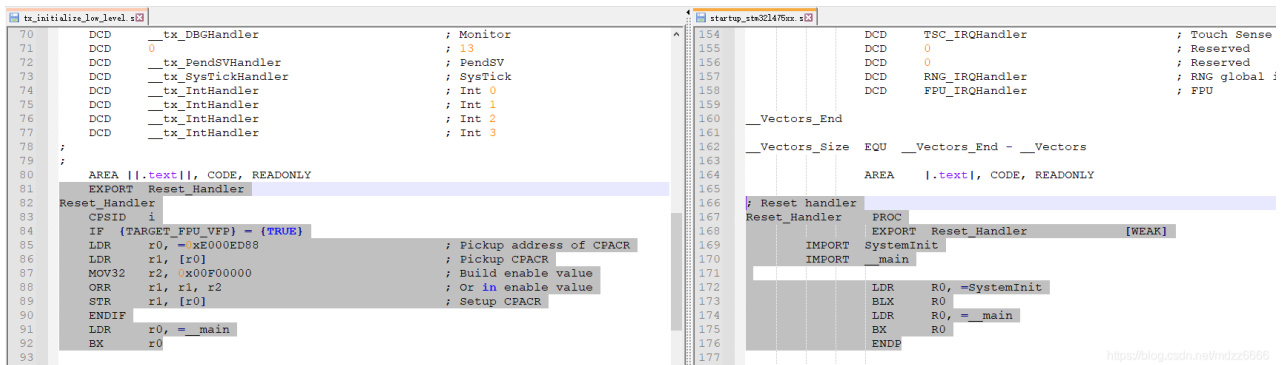
①两个启动文件都有对于堆栈指针的描述，我们删除左边，以右边为准



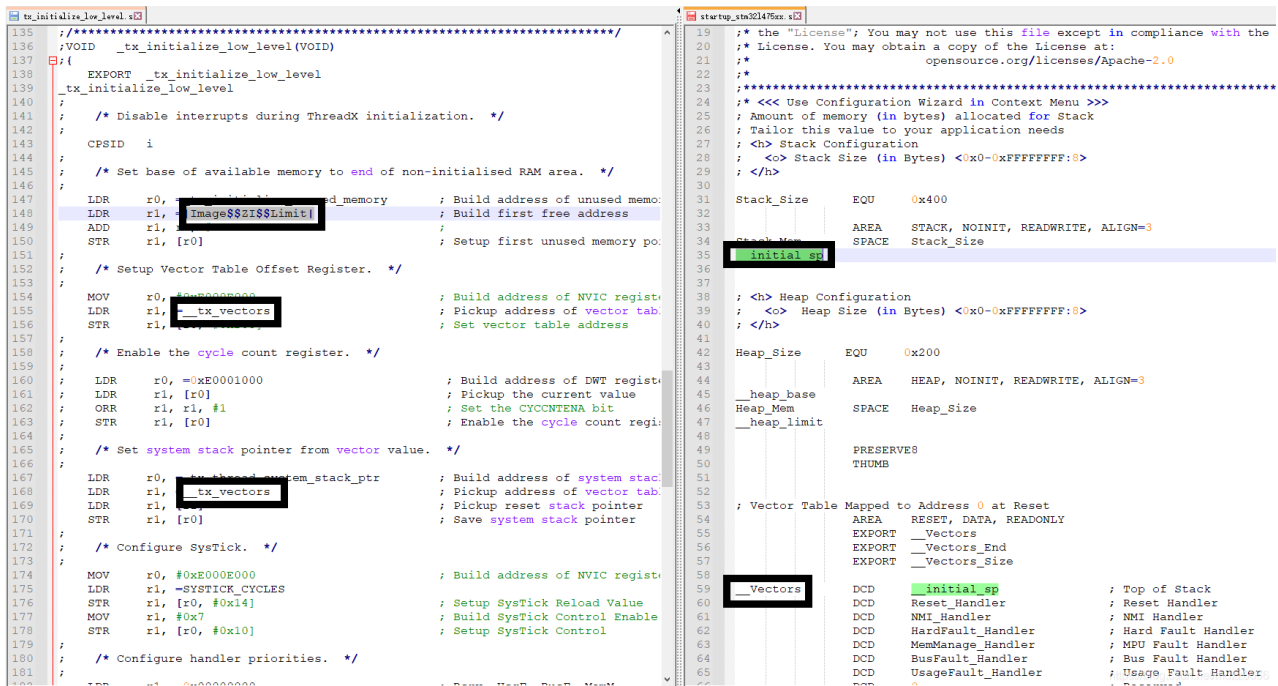
②两个启动文件都有中断向量表，但tx_initialize_low_level.s只描述了部分中断向量，不够完整，所以我们删除左边，以右边为准



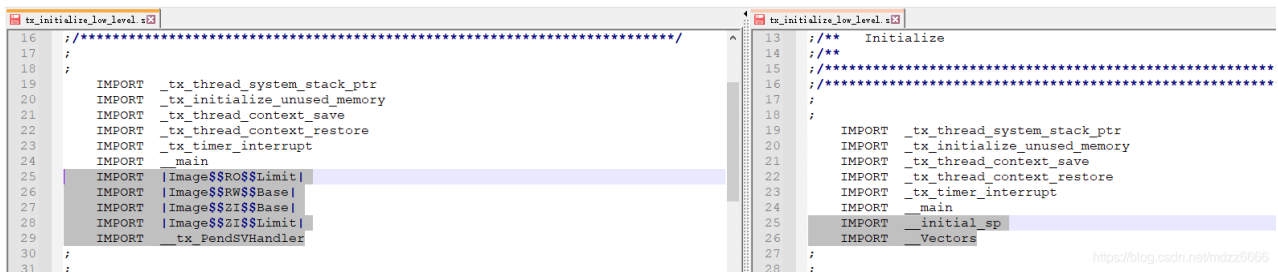
③两个文件都有中断处理函数，我们删除左边，以右边为准



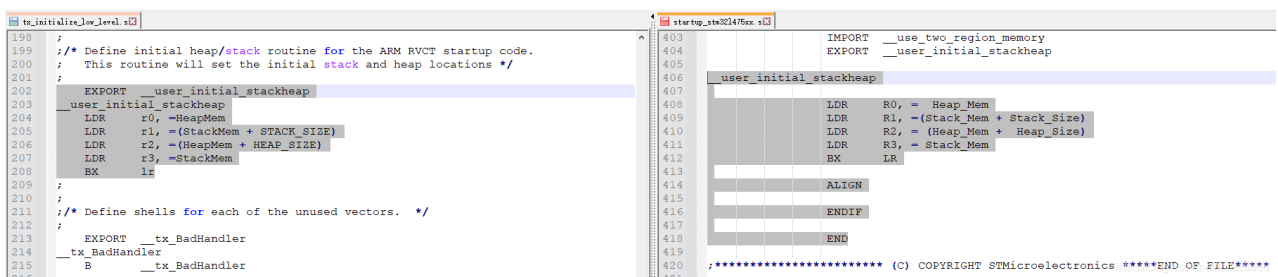
④修改tx_initialize_low_level.s文件中的VOID tx_initialize_low_level(VOID)代码段。将 |Image\$\$ZI\$\$Limit| 和 __tx_vectors 替换为startup_stm32l475xx.s文件中的 __initial_sp 和 __Vectors。



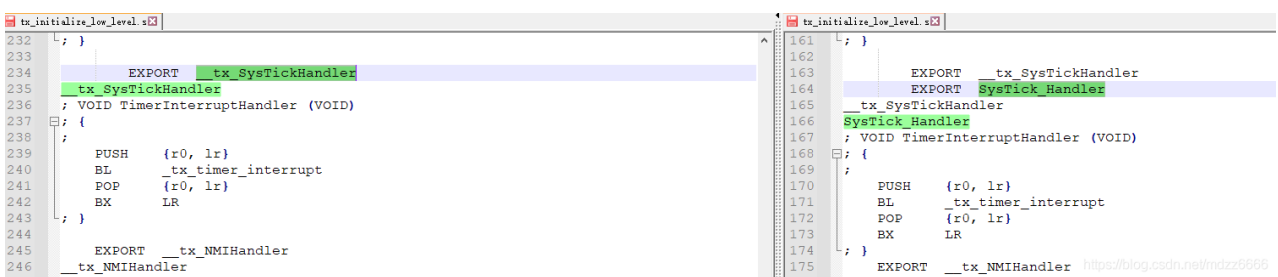
因为 `__initial_sp` 和 `__Vectors` 声明在 `startup_stm32l475xx.s` 文件，所以应该在 `tx_initialize_low_level.s` 文件中引入上述两个变量。 `IMPORT |Image$$RO$$Limit| IMPORT |Image$$RW$$Base| IMPORT |Image$$ZI$$Base| IMPORT |Image$$ZI$$Limit| IMPORT __tx_PendSVHandler` 未使用，可以删除，修改好的代码如下图所示



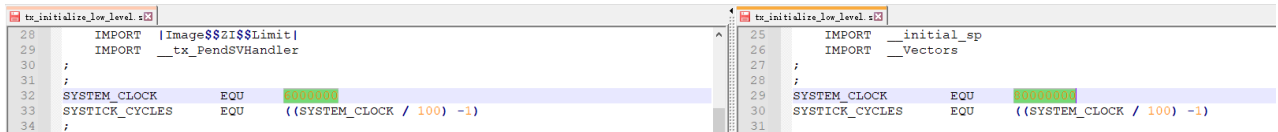
⑤两个文件都有初始化堆，我们删除左边，以右边为准



⑥在 `tx_initialize_low_level.s` 文件定义 `SysTick_Handler` 代码段，只需添加两行代码，修改完成后如右图所示



⑦修改tx_initialize_low_level.s文件中系统时钟，其与STM32CubeMX中设置的时钟一致，否则会出现奇怪的运行结果。这里我设置的是80M，如右图所示



```
tx_initialize_low_level.s
28      IMPORT |Image$$ZI$$Limit|
29      IMPORT __tx_PendSVHandler
30      ;
31      ;
32      SYSTEM_CLOCK      EQU      80000000
33      SYSTICK_CYCLES     EQU      ((SYSTEM_CLOCK / 100) - 1)
34      ;

tx_initialize_low_level.s
25      IMPORT __initial_sp
26      IMPORT __Vectors
27      ;
28      ;
29      SYSTEM_CLOCK      EQU      80000000
30      SYSTICK_CYCLES     EQU      ((SYSTEM_CLOCK / 100) - 1)
31      ;
```

修改完成后的tx_initialize_low_level.s文件如下所示：

```

;/**
;
;/**      Copyright (c) Microsoft Corporation. All rights reserved.
;/**
;/**
;/**
;
;
;
;/**
;/**
;/**
;/**      ThreadX Component
;/**
;/**
;/**      Initialize
;/**
;/**
;/**
;/**
;
;
;
IMPORT  _tx_thread_system_stack_ptr
IMPORT  _tx_initialize_unused_memory
IMPORT  _tx_thread_context_save
IMPORT  _tx_thread_context_restore
IMPORT  _tx_timer_interrupt
IMPORT  __main
IMPORT  __initial_sp
IMPORT  __Vectors
;
;
SYSTEM_CLOCK      EQU      80000000
SYSTICK_CYCLES    EQU      ((SYSTEM_CLOCK / 100) -1)

AREA ||.text||, CODE, READONLY

;/**
;/**
;/**      FUNCTION                      RELEASE
;/**
;/**      _tx_initialize_low_level      Cortex-M4/AC5
;/**      6.1
;/**      AUTHOR
;/**
;/**      William E. Lamie, Microsoft Corporation.
;/**
;/**      DESCRIPTION
;/**
;/**      This function is responsible for any low-level processor
;/**      initialization, including setting up interrupt vectors, setting
;/**      up a periodic timer interrupt source, saving the system stack
;/**      pointer for use in ISR processing later, and finding the first
;/**      available RAM memory address for tx_application_define.
;/**
;/**      INPUT
;/**
;/**      None
;/**
;/**      OUTPUT
;/**
;/**      None
;/**

```

```

; /* CALLS */
; /* */
; /* None */
; /* */
; /* CALLED BY */
; /* */
; /* _tx_initialize_kernel_enter ThreadX entry function */
; /* */
; /* RELEASE HISTORY */
; /* */
; /* DATE NAME DESCRIPTION */
; /* */
; /* 09-30-2020 William E. Lamie Initial Version 6.1 */
; /* */
; /****** */
; VOID _tx_initialize_low_level(VOID)
; {
;     EXPORT _tx_initialize_low_level
;     _tx_initialize_low_level
;
;     /* Disable interrupts during ThreadX initialization. */
;
;     CPSID i
;
;     /* Set base of available memory to end of non-initialised RAM area. */
;
;     LDR r0, =_tx_initialize_unused_memory ; Build address of unused memory
pointer
;     LDR r1, =__initial_sp ; Build first free address
;     ADD r1, r1, #4 ;
;     STR r1, [r0] ; Setup first unused memory pointer
;
;     /* Setup Vector Table Offset Register. */
;
;     MOV r0, #0xE000E000 ; Build address of NVIC registers
;     LDR r1, =__Vectors ; Pickup address of vector table
;     STR r1, [r0, #0xD08] ; Set vector table address
;
;     /* Enable the cycle count register. */
;
;     LDR r0, =0xE0001000 ; Build address of DWT register
;     LDR r1, [r0] ; Pickup the current value
;     ORR r1, r1, #1 ; Set the CYCCNTENA bit
;     STR r1, [r0] ; Enable the cycle count register
;
;     /* Set system stack pointer from vector value. */
;
;     LDR r0, =_tx_thread_system_stack_ptr ; Build address of system stack
pointer
;     LDR r1, =__Vectors ; Pickup address of vector table
;     LDR r1, [r1] ; Pickup reset stack pointer
;     STR r1, [r0] ; Save system stack pointer
;
;     /* Configure SysTick. */
;
;     MOV r0, #0xE000E000 ; Build address of NVIC registers
;     LDR r1, =SYSTICK_CYCLES
;     STR r1, [r0, #0x14] ; Setup SysTick Reload Value
;     MOV r1, #0x7 ; Build SysTick Control Enable Value
;     STR r1, [r0, #0x10] ; Setup SysTick Control

```

```

;
;   /* Configure handler priorities.  */
;
    LDR    r1, =0x00000000                ; Rsrv, UsqF, BusF, MemM
    STR    r1, [r0, #0xD18]              ; Setup System Handlers 4-7 Priority
Registers

    LDR    r1, =0xFF000000                ; SVC1, Rsrv, Rsrv, Rsrv
    STR    r1, [r0, #0xD1C]              ; Setup System Handlers 8-11 Priority
Registers

; Note: SVC must be lowest priority,
which is 0xFF

    LDR    r1, =0x40FF0000                ; SysT, PnSV, Rsrv, DbgM
    STR    r1, [r0, #0xD20]              ; Setup System Handlers 12-15 Priority
Registers

; Note: PnSV must be lowest priority,
which is 0xFF

;
;   /* Return to caller.  */
;
    BX     lr
; }
;
;

; /* Define shells for each of the unused vectors.  */
;
    EXPORT __tx_BadHandler
__tx_BadHandler
    B      __tx_BadHandler

    EXPORT __tx_SVCallHandler
__tx_SVCallHandler
    B      __tx_SVCallHandler

    EXPORT __tx_IntHandler
__tx_IntHandler
; VOID InterruptHandler (VOID)
; {
    PUSH    {r0, lr}

;   /* Do interrupt handler work here */
;   /* .... */

    POP     {r0, lr}
    BX     LR
; }

    EXPORT __tx_SysTickHandler
    EXPORT SysTick_Handler
__tx_SysTickHandler
SysTick_Handler
; VOID TimerInterruptHandler (VOID)
; {
;
    PUSH    {r0, lr}
    BL      __tx_timer_interrupt
    POP     {r0, lr}
    BX     LR

```



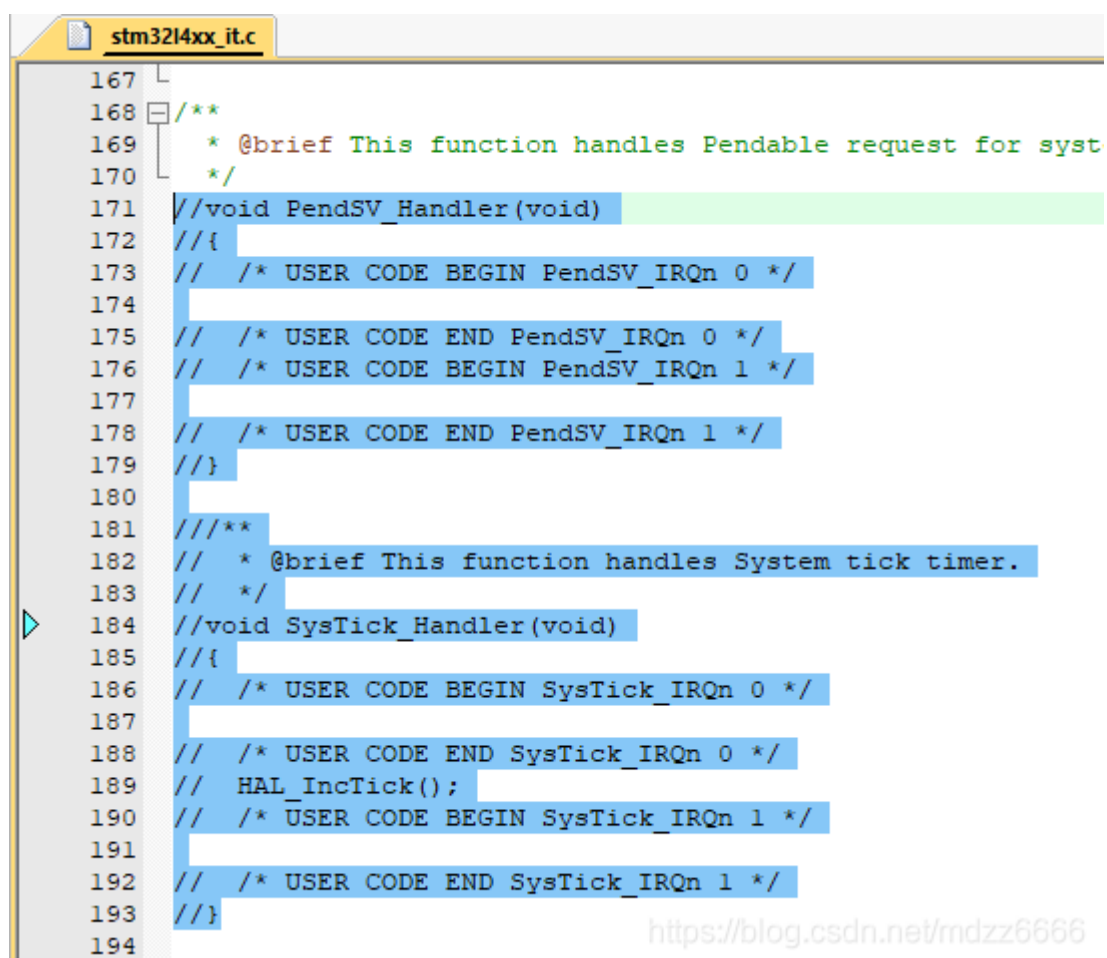
```
; }
```

```
EXPORT __tx_NMIHandler
__tx_NMIHandler
B      __tx_NMIHandler

EXPORT __tx_DBGHandler
__tx_DBGHandler
B      __tx_DBGHandler

ALIGN
LORG
END
```

注释LED工程原本的 `void PendSV_Handler(void);` 和 `void SysTick_Handler(void);`;



```
stm32l4xx_it.c
167
168 /**
169  * @brief This function handles Pendable request for syst
170  */
171 //void PendSV_Handler(void)
172 //{
173 //  /* USER CODE BEGIN PendSV_IRQn 0 */
174 //
175 //  /* USER CODE END PendSV_IRQn 0 */
176 //  /* USER CODE BEGIN PendSV_IRQn 1 */
177 //
178 //  /* USER CODE END PendSV_IRQn 1 */
179 //}
180
181 /**
182  * @brief This function handles System tick timer.
183  */
184 //void SysTick_Handler(void)
185 //{
186 //  /* USER CODE BEGIN SysTick_IRQn 0 */
187 //
188 //  /* USER CODE END SysTick_IRQn 0 */
189 //  HAL_IncTick();
190 //  /* USER CODE BEGIN SysTick_IRQn 1 */
191 //
192 //  /* USER CODE END SysTick_IRQn 1 */
193 //}
194
```

<https://blog.csdn.net/midzz6666>

```
37  /* USER CODE END ET */
38
39  /* Exported constants -----
40  /* USER CODE BEGIN EC */
41
42  /* USER CODE END EC */
43
44  /* Exported macro -----
45  /* USER CODE BEGIN EM */
46
47  /* USER CODE END EM */
48
49  /* Exported functions prototypes -----
50  void NMI_Handler(void);
51  void HardFault_Handler(void);
52  void MemManage_Handler(void);
53  void BusFault_Handler(void);
54  void UsageFault_Handler(void);
55  void SVC_Handler(void);
56  void DebugMon_Handler(void);
57  //void PendSV_Handler(void);
58  //void SysTick_Handler(void);
59  /* USER CODE BEGIN EFP */
60
61  /* USER CODE END EFP */
62
63  #ifdef __cplusplus
64  }
65  #endif
66
67  #endif /* __STM32L4xx_IT_H */
68
```

提供一个简单的测试任务

```
uint8_t my_buff[1024];
uint8_t you_buff[1024];

TX_THREAD my_thread,you_thread;
void my_thread_entry(ULONG thread_input)
{
    /* Enter into a forever loop. */
    while(1)
    {
        tx_thread_sleep(3);
    }
}

void you_thread_entry(ULONG thread_input)
{
    while(1)
    {
        tx_thread_sleep(3);
    }
}

void tx_application_define(void *first_unused_memory)
{
    /* Create my_thread! */
    tx_thread_create(&my_thread, "My Thread",
        my_thread_entry, 0x1234, my_buff, 1024,
        3, 3, TX_NO_TIME_SLICE, TX_AUTO_START);

    tx_thread_create(&you_thread, "You Thread",
        you_thread_entry, 0x1234, you_buff, 1024,
        3, 3, TX_NO_TIME_SLICE, TX_AUTO_START);
}
```

在main函数中添加头文件 `#include "tx_api.h"` 后，调用 `tx_kernel_enter();` 即可运行任务。

记录一下踩过的坑

- ①STM32CubeMX生成的工程默认不勾选Reset and Run，程序下载后不运行，建议勾选。
- ②tx_initialize_low_level.s文件中的系统时钟最好与STM32CubeMX中保持移植，否则可能出现未知的运行结果。
- ③开发时建议将KEIL的代码优化设置为o