AN0079
应用笔记

AT32微控制器上移植ThreadX操作系统

前言

本应用笔记主要演示ThreadX操作系统移植到AT32 MCU的过程和方法。

注：本应用笔记对应的代码是基于雅特力提供的V2.x.x 板级支持包（BSP）而开发，对于其他版本BSP，需要注意使用上的区别。

支持型号列表：

支持型号	AT32F4 系列

目录

1 ThreadX 在 MDK 移植 5

1.1 软件资源准备 5

1.2 MDK 源码工程配置 5

1.3 源码修改 9

1.4 编写应用代码 11

2 示例快速使用 13

2.1.1 硬件资源 13

2.1.2 软件资源 13

2.1.3 demo 使用 13

3 文档版本历史 15

表目录

表 1. 文档版本历史 15

图目录

图 1.at32 bsp templates 文件夹 5
图 2. ThreadX common 和 ports 文件夹 6
图 3. MDK5 上配置 AC6 和 FPU 6
图 4. MDK5 导入 ThreadX C 文件 7
图 5. MDK5 导入 ThreadX S 文件 8
图 6. MDK5 添加 ThreadX 头文件路径 8
图 7. MDK5 汇编选项配置 9
图 8. 添加语句和修改主频 10
图 9. 修改__initial_sp/ Vectors 语句 10
图 10. 屏蔽 PendSV 和 Systick 中断服务函数 11
图 11 串口助手打印信息 14

1 ThreadX 在 MDK 移植

1.1 软件资源准备

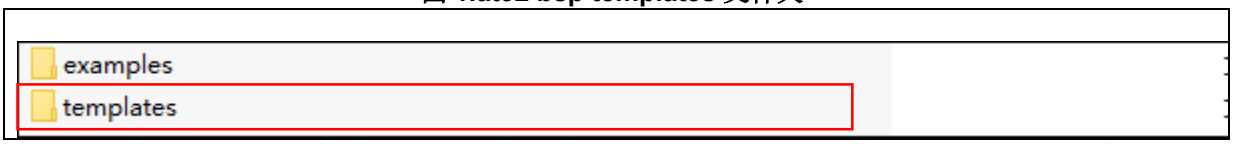
移植前需要提前准备好的软件资源有：

- AT32 BSP：雅特力论坛下载
- ThreadX 源码
  Github 获取地址：https://github.com/azure-rtos
  硬汉嵌入式论坛获取地址：http://www.armbbs.cn/forum.php?mod=viewthread&tid=99514
- MDK5.30 及以上版本：Keil 官方下载

1.2 MDK 源码工程配置

STEP 1 准备一个可以实现 printf 串口打印的裸机工程.
对于 AT32 MCU，可直接使用 bsp 中 Templates 工程。

图 1.at32 bsp templates 文件夹

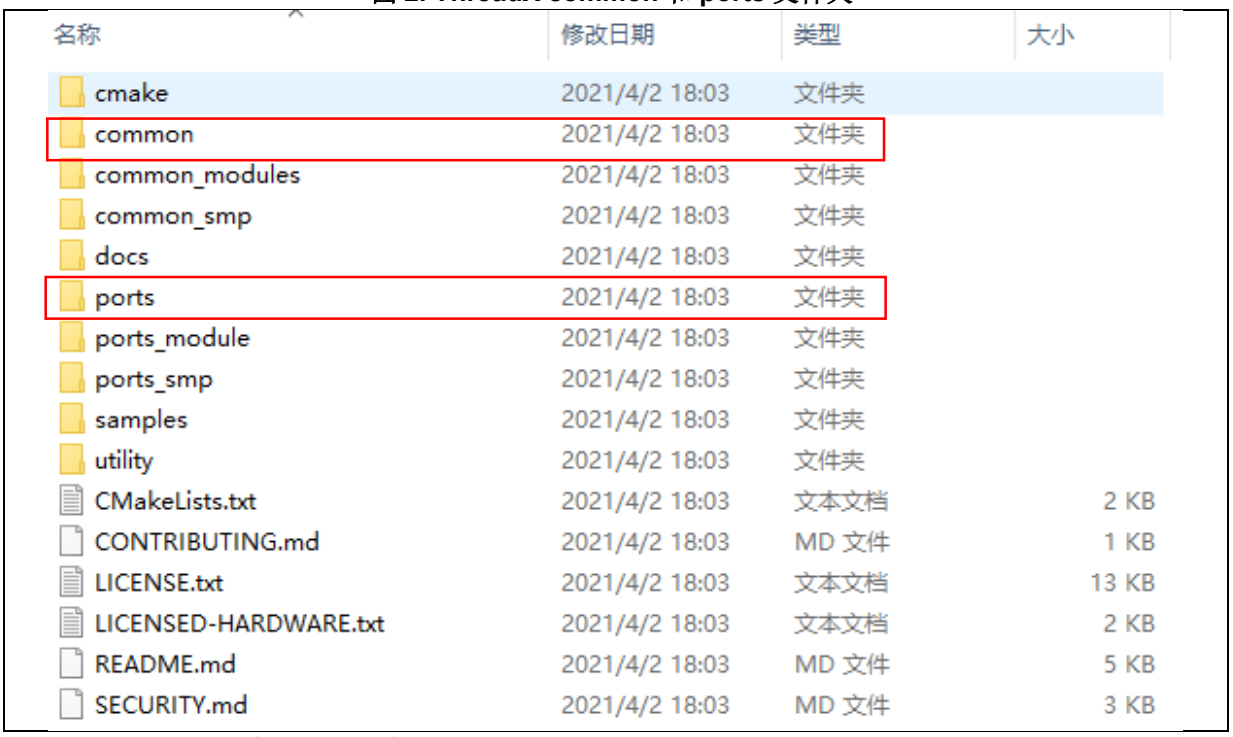


由于 AC6 编译器对 printf 初始化与 AC5 有点差异，__use_no_semihosting 部分需按如下代码处理方式，否则将无法正常 printf 打印信息。

```
/* printf function enable in arm compiler 6 */
__asm (".global __use_no_semihosting\n\t");
void _sys_exit(int x)
{
   x = x;
}

void _ttywrch(int ch)
{
      ch = ch;
}

FILE __stdout;

#if defined ( __GNUC__ ) && !defined (__clang__)
   #define PUTCHAR_PROTOTYPE int __io_putchar(int ch)
#else
   #define PUTCHAR_PROTOTYPE int fputc(int ch, FILE *f)
#endif /* __GNUC__ */
```

STEP 2 ThreadX 源码拷贝

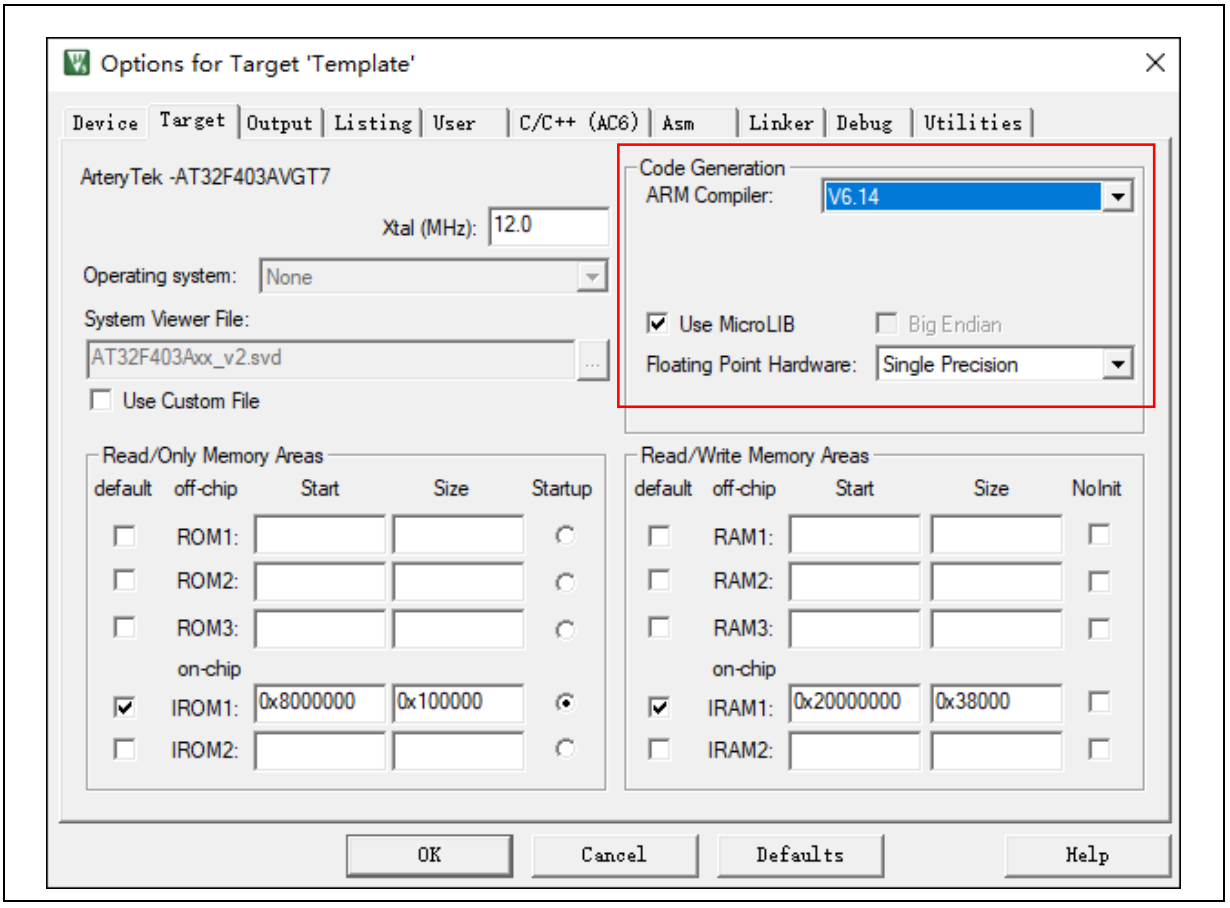
图 2. ThreadX common 和 ports 文件夹



STEP 3 配置 AC6 编译器并开启 FPU

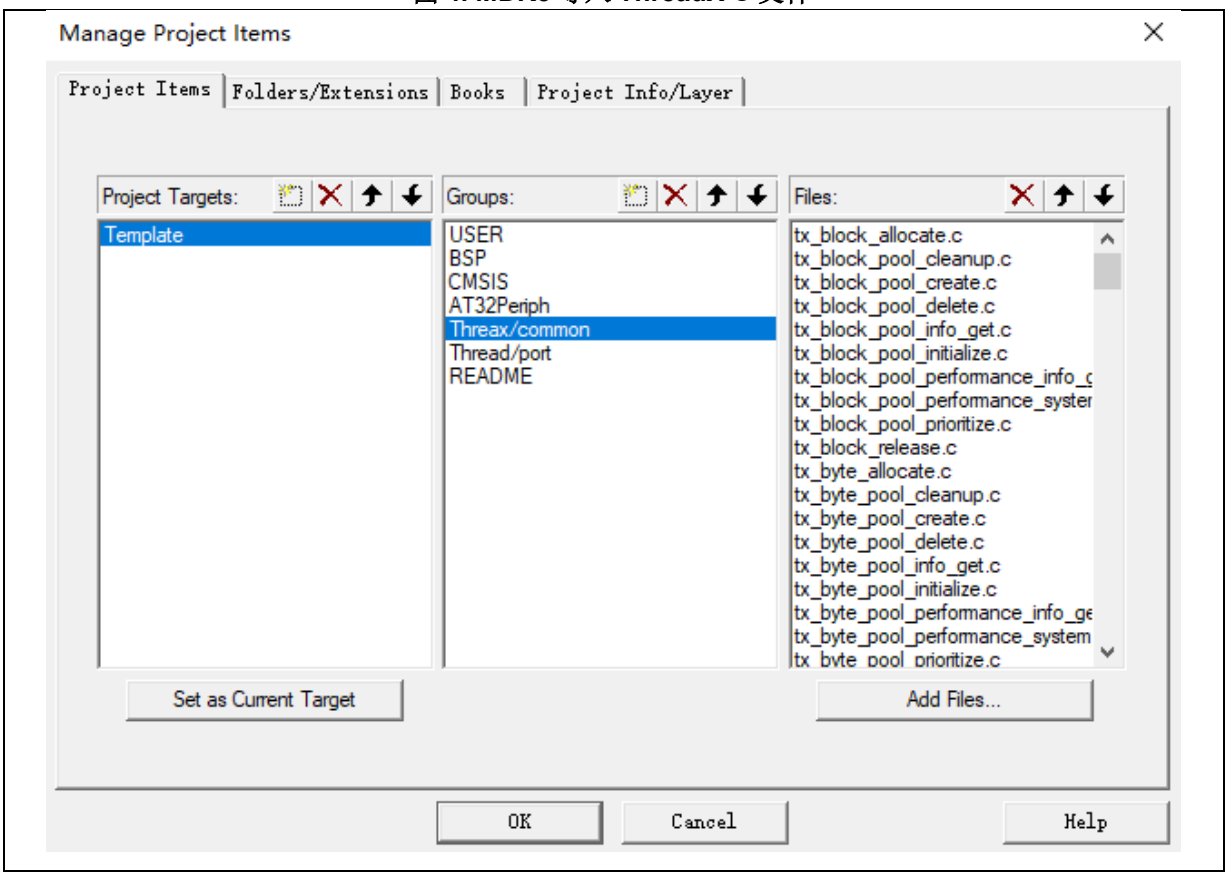
优化登记按 AC6 默认优化登记，切勿修改，即维持-Oz image size

图 3. MDK5 上配置 AC6 和 FPU



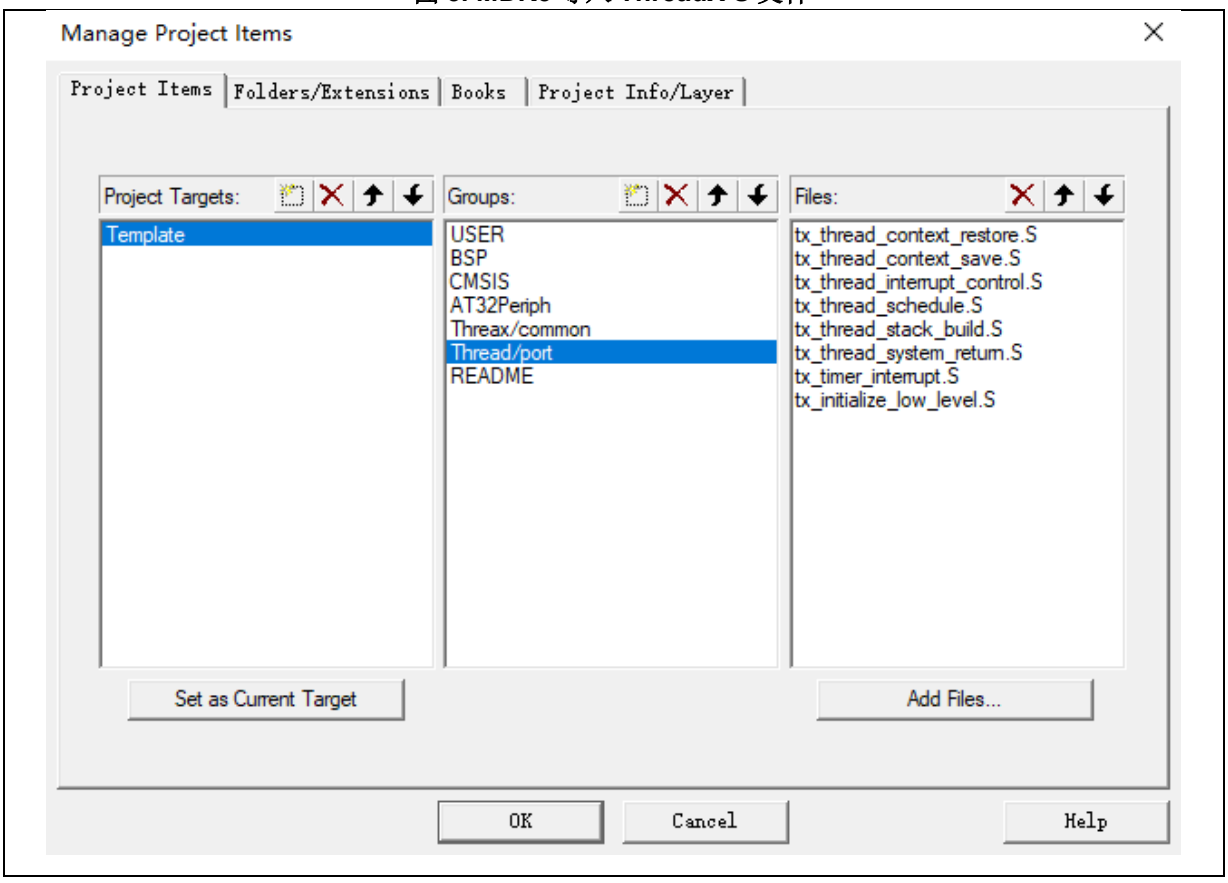
STEP 4 ThreadX C 文件导入

图 4. MDK5 导入 ThreadX C 文件



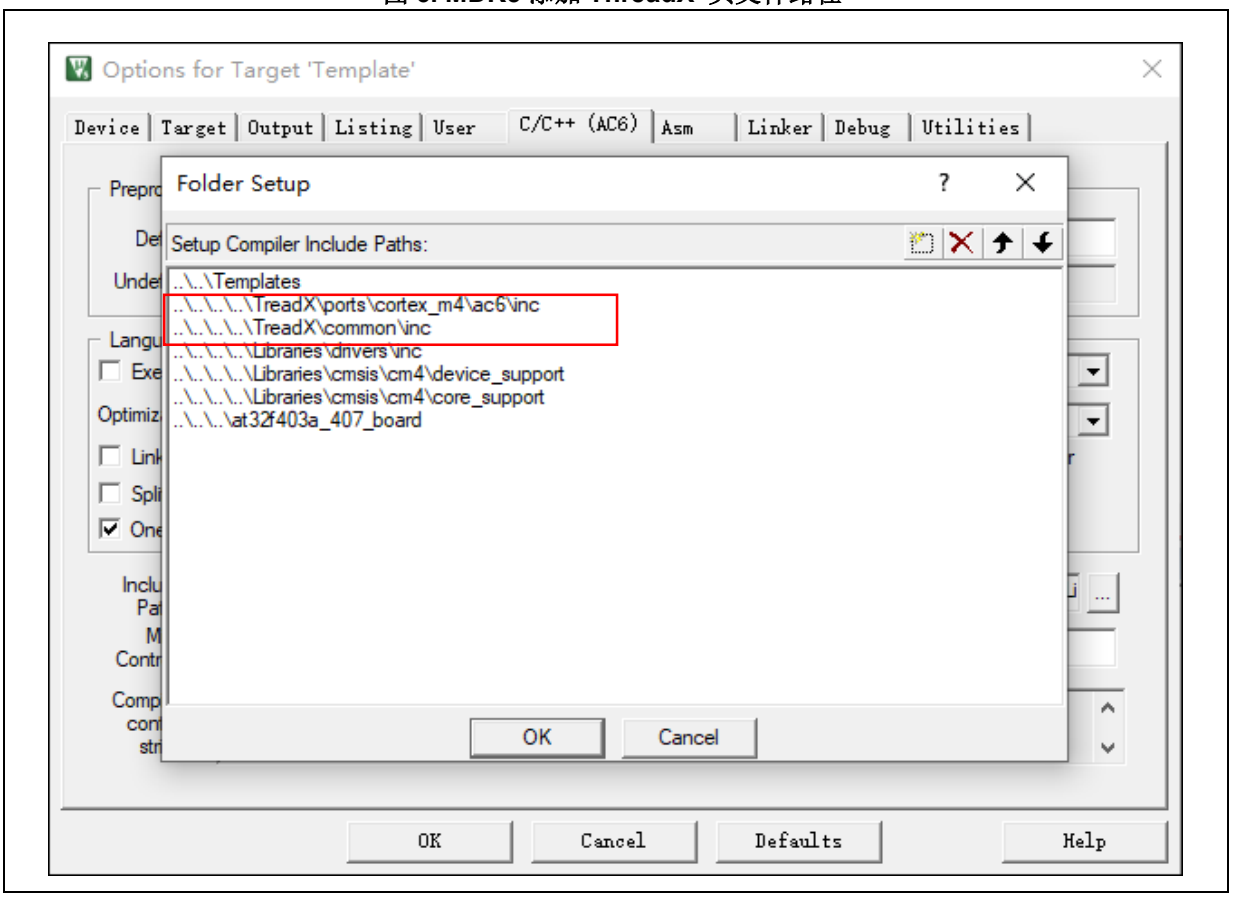
STEP 5 ThreadX S 文件导入
选择 ac6 路径：threadx-6.1.6_rel\ports\cortex_m4\ac6
tx_initialize_low_level.S 文件在 sample_threadx 文件夹中
TreadX\ports\cortex_m4\ac6\example_build\sample_threadx

图 5. MDK5 导入 ThreadX S 文件



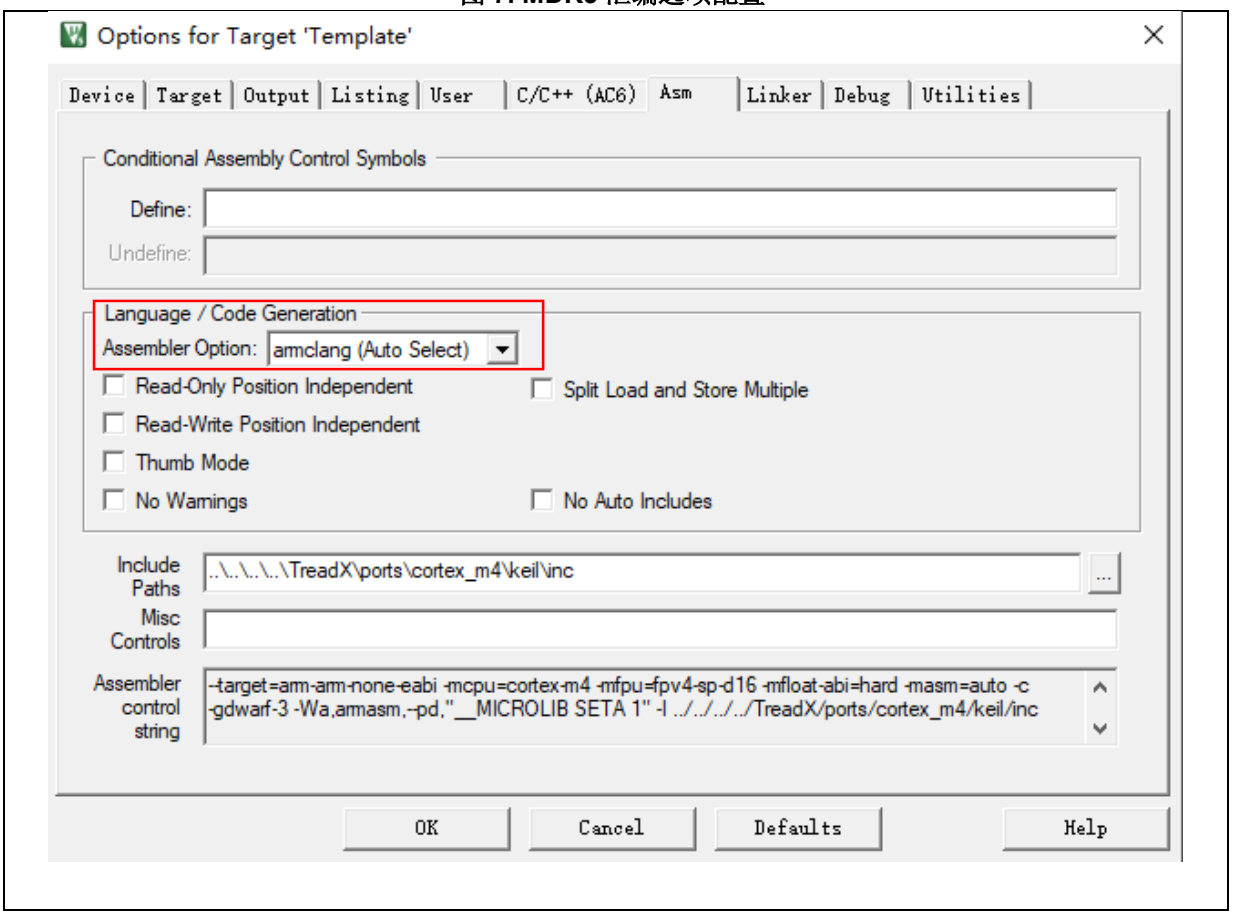
STEP 6 头文件地址添加

图 6. MDK5 添加 ThreadX 头文件路径



STEP 7 设置 ASM 汇编选项

图 7. MDK5 汇编选项配置



1.3 源码修改

STEP1 修改 tx_initialize_low_level.S 文件

- 增加如下语句，如图 8

```
.global     __initial_sp
.global     __Vectors
```

- 修改系统主频 SYSTEM_CLOCK，如图 8
- 修改对应__initial_sp/ Vectors 语句，如图 9

图 8. 添加语句和修改主频

tx_initialize_low_level.S

```
@
@
    .global     _tx_thread_system_stack_ptr
    .global     _tx_initialize_unused_memory
    .global     _tx_timer_interrupt
    .global     __main
    .global     __tx_SVCallHandler
    .global     __tx_PendSVHandler
    .global     __tx_NMIHandler                 @ NMI
    .global     __tx_BadHandler                 @ HardFault
    .global     __tx_SVCallHandler              @ SVCall
    .global     __tx_DBGHandler                 @ Monitor
    .global     __tx_PendSVHandler              @ PendSV
    .global     __tx_SysTickHandler             @ SysTick
    .global     __tx_IntHandler                 @ Int 0
  .global     __initial_sp
  .global     __Vectors
@
@
SYSTEM_CLOCK      =    240000000
SYSTICK_CYCLES    =    ((SYSTEM_CLOCK / 1000) -1)

    .text 32
    .align 4
    .syntax unified
```

图 9. 修改__initial_sp/ Vectors 语句

tx_initialize_low_level.S

```
    /* 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
@
    /* 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
@
    /* 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
@
    /* Configure SysTick for 100Hz clock, or 16384 cycles if no reference.  */
```

STEP2　屏蔽原 at32f403a_407_int.c 中的 PendSV 和 Systick 中断服务函数：

图 10. 屏蔽 PendSV 和 Systick 中断服务函数

at32f403a_407_int.c

```
110    * @param  none
111    * @retval none
112    */
113  void DebugMon_Handler(void)
114  {
115  }
116
117  /**
118    * @brief  this function handles pendsv_handler exception.
119    * @param  none
120    * @retval none
121    */
122  //void PendSV_Handler(void)
123  //{
124  //}
125
126  ///**
127  //  * @brief  this function handles systick handler.
128  //  * @param  none
129  //  * @retval none
130  //  */
131  //void SysTick_Handler(void)
132  //{
133  //}
```

移植完成，编译会提示有一个错误：
这个函数是留给用户自己定义的，接下来会创建。

1.4 编写应用代码

STEP1 包含进来 threadx 的头文件：

```
/* Private includes ----------------------------------------------------------*/
/* USER CODE BEGIN Includes */
#include <stdio.h>
#include "tx_api.h"
/* USER CODE END Includes */
```

STEP2 定义两个线程：

```
/* Private user code ---------------------------------------------------------*/
/* USER CODE BEGIN 0 */
TX_THREAD my_thread1;

TX_THREAD my_thread2;

void my_thread1_entry(ULONG thread_input)
{
   /* Enter into a forever loop. */
  while(1)
  {
     printf("threadx 1 application running...\r\n");
     /* Sleep for 1 tick. */
     tx_thread_sleep(1000);
  }
}
void my_thread2_entry(ULONG thread_input)
{
  /* Enter into a forever loop. */
```

```
      while(1)
      {
         printf("threadx 2 application running...\r\n");
         /* Sleep for 1 tick. */
         tx_thread_sleep(1000);
       }
   }

   void tx_application_define(void *first_unused_memory)
   {
      /* Create my_thread! */
T     x_thread_create(&my_thread1, "My Thread 1",
      My_thread1_entry, 0x1234, first_unused_memory, 1024, 3, 3, TX_NO_TIME_SLICE,
      TX_AUTO_START);

      tx_thread_create(&my_thread2, "My Thread 2",
      my_thread2_entry, 0x1234, first_unused_memory+1024, 1024, 1, 1, TX_NO_TIME_SLICE,
      TX_AUTO_START);
   }
```

STEP3　在 main 函数初始化完成之后启动内核：

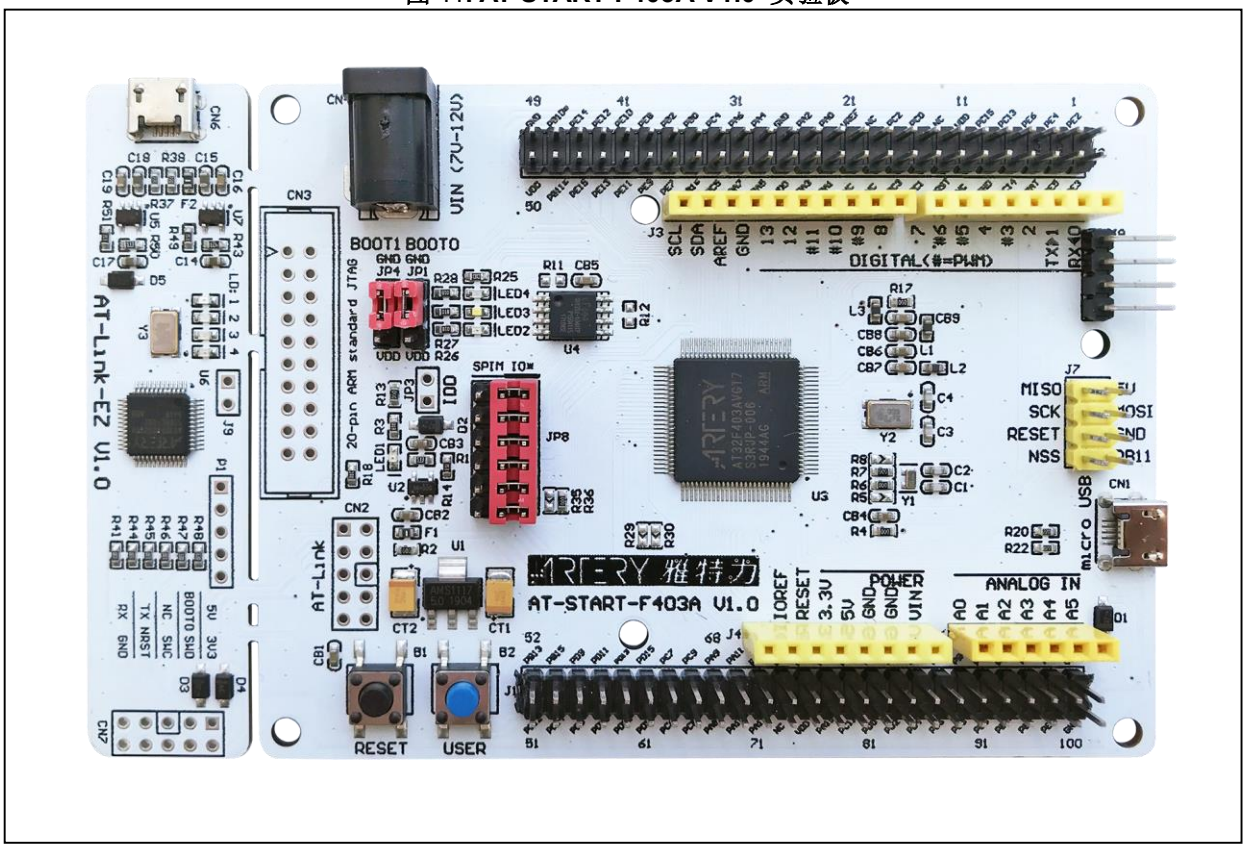
```
   /* USER CODE BEGIN 2 */

   printf("threadX RTOS on BearPi IoT Board\r\n");

   /* Enter the ThreadX kernel. */
   tx_kernel_enter( );

   /* USER CODE END 2 */
```

2 示例快速使用

2.1.1 硬件资源

1) 指示灯LED2/LED3
2) USART1(PA9/PA10)
3) AT-START-F403A V1.0 实验板

图 11. **AT-START-F403A V1.0 实验板**



注：文档中是基于AT32F403A的硬件条件为例，demo源代码还包括AT32其他型号，请编译烧录在相应AT-START开发板运行即可。

2.1.2 软件资源

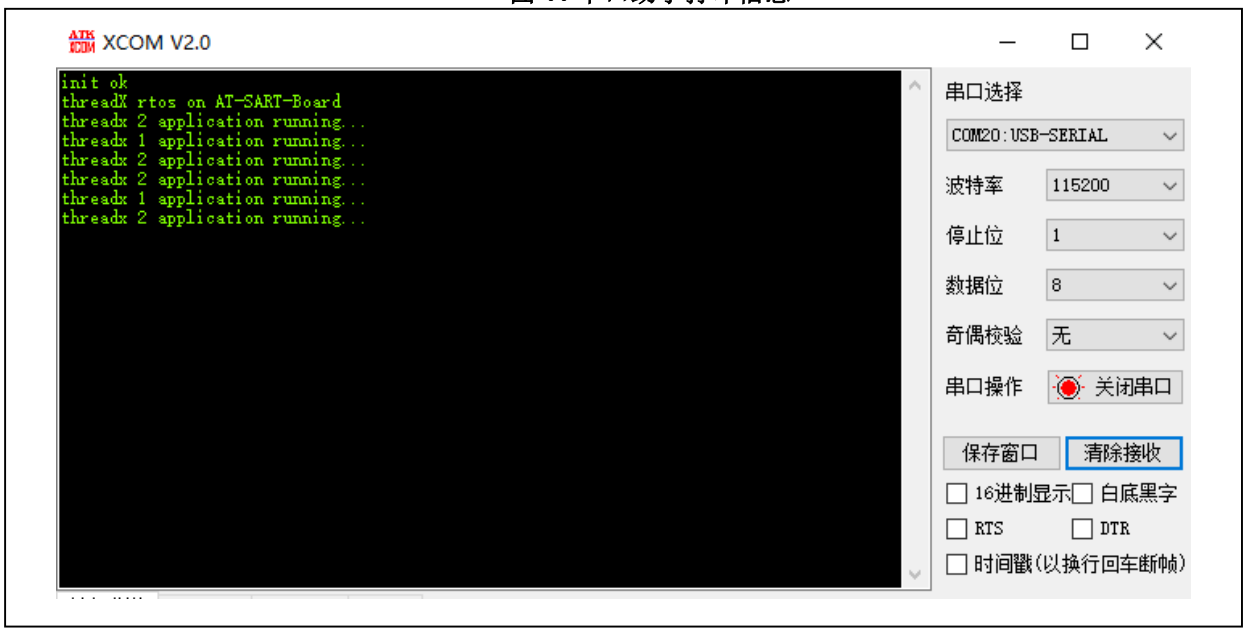
1) SourceCode
   - AT32F4xx_ThreadX_V2.x.x.zip
2) Doc
   - AN0079_AT32_MCU_On_ThreadX_OS_ZH_V2.x.x

注：所有project都是基于keil 5而建立，若用户需要在其他编译环境上使用，请参考AT32xxx_Firmware_Library_V2.x.x\project\at_start_xxx\templates中各种编译环境（例如IAR6/7,keil 4/5）进行简单修改即可。

2.1.3 demo 使用

1) 通过USB下载线连接AT-START，并在串口助手找到AT Link虚拟串口
2) 打开AT32F4xx_ThreadX工程源程序，编译后下载到实验板
3) 观察LED2/3闪烁状态和串口助手打印信息

图 11 串口助手打印信息



3 文档版本历史

表 1. 文档版本历史

日期	版本	变更
2021.12.03	2.0.0	最初版本

重要通知 - 请仔细阅读

买方自行负责对本文所述雅特力产品和服务的选择和使用，雅特力概不承担与选择或使用本文所述雅特力产品和服务相关的任何责任。

无论之前是否有过任何形式的表示，本文档不以任何方式对任何知识产权进行任何明示或默示的授权或许可。如果本文档任何部分涉及任何第三方产品或服务，不应被视为雅特力授权使用此类第三方产品或服务，或许可其中的任何知识产权，或者被视为涉及以任何方式使用任何此类第三方产品或服务或其中任何知识产权的保证。

除非在雅特力的销售条款中另有说明，否则，雅特力对雅特力产品的使用和/或销售不做任何明示或默示的保证，包括但不限于有关适销性、适合特定用途（及其依据任何司法管辖区的法律的对应情况），或侵犯任何专利、版权或其他知识产权的默示保证。

雅特力产品并非设计或专门用于下列用途的产品：（A）对安全性有特别要求的应用，例如：生命支持、主动植入设备或对产品功能安全有要求的系统；（B）航空应用；（C）航天应用或航天环境；（D）武器，且/或（E）其他可能导致人身伤害、死亡及财产损害的应用。如果采购商擅自将其用于前述应用，即使采购商向雅特力发出了书面通知，风险及法律责任仍将由采购商单独承担，且采购商应独力负责在前述应用中满足所有法律和法规要求。

经销的雅特力产品如有不同于本文档中提出的声明和/或技术特点的规定，将立即导致雅特力针对本文所述雅特力产品或服务授予的任何保证失效，并且不应以任何形式造成或扩大雅特力的任何责任。