

AT32微控制器上移植ThreadX操作系统

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:

17557

AT32上移植ThreadX系统的方法和要点

目录

1	Thre	adX 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1	3
		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1	5



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	



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: <u>https://github.com/azure-rtos</u> 硬汉嵌入式论坛获取地址: <u>http://www.armbbs.cn/forum.php?mod=viewthread&tid=99514</u>
- 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

examples	
templates	

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

名称	修改日期	类型	大小
📙 cmake	2021/4/2 18:03	文件夹	
common	2021/4/2 18:03	文件夹	
common_modules	2021/4/2 18:03	文件夹	
common_smp	2021/4/2 18:03	文件夹	
docs	2021/4/2 18:03	文件夹	
ports	2021/4/2 18:03	文件夹	
ports_module	2021/4/2 18:03	文件夹	
ports_smp	2021/4/2 18:03	文件夹	
samples	2021/4/2 18:03	文件夹	
utility	2021/4/2 18:03	文件夹	
CMakeLists.txt	2021/4/2 18:03	文本文档	2 KB
CONTRIBUTING.md	2021/4/2 18:03	MD 文件	1 KB
LICENSE.txt	2021/4/2 18:03	文本文档	13 KB
LICENSED-HARDWARE.txt	2021/4/2 18:03	文本文档	2 KB
README.md	2021/4/2 18:03	MD 文件	5 KB
SECURITY.md	2021/4/2 18:03	MD 文件	3 KB

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

ArteryTel	< -AT32F40	3AVGT7	No. (111)	20	Code (ARM (Generation Compiler:	V6.14		•
Operatin	g system:	None							
System \	/iewer File:	,			V:	se MicroLl	B 🗆 E	lig Endian	
AT32F4	03Axx_v2.	svd			Floatin	ng Point H	ardware: Sing	le Precision	-
🗌 Use	Custom File	e							
Read/	Only Memo	ny Areas —			-Read/	Write Men	nory Areas		
default	off-chip	Start	Size	Startup	default	off-chip	Start	Size	Nolnit
	ROM1:			0		RAM1:			
	ROM2:			0		RAM2:			
	ROM3:					RAM3:			
	on-chip	, 				on-chip		,	
V	IROM1:	0x8000000	0x100000	ſ		IRAM1:	0x20000000	0x38000	
						IRAM2-			

STEP 4 ThreadX C 文件导入



Project Targets: 🖄 🗙 🛧 🖌	🖡 Groups: 👘 🗙 🛧 🗲	Files: 🗙 🗲 🗲
Template	USER BSP CMSIS AT32Periph Threax/common Thread/port README	tx_block_allocate.c tx_block_pool_cleanup.c tx_block_pool_create.c tx_block_pool_delete.c tx_block_pool_info_get.c tx_block_pool_performance_info_c tx_block_pool_performance_syster tx_block_pool_performance_syster tx_block_release.c tx_byte_pool_cleanup.c tx_byte_pool_cleanup.c tx_byte_pool_delete.c tx_byte_pool_delete.c tx_byte_pool_delete.c tx_byte_pool_of_ent.c tx_byte_pool_performance_info_get.c tx_byte_pool_performance_syster
Set as Current Target		Add Files

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



Manage Project Items	· · · · ·		×
Project Items Folders/Extensions	Books Project Info/Layer Groups: X + + USER BSP CMSIS AT32Periph Threax/common Thread/port README	Files: ★ ★ ↓ tx_thread_context_restore.S tx_thread_context_save.S tx_thread_interrupt_control.S tx_thread_schedule.S tx_thread_schedule.S tx_thread_system_retum.S tx_thread_system_retum.S tx_timer_interrupt.S tx_initialize_low_level.S	
Set as Current Target	OK Cancel	Add Files Kelp	

图 5. MDK5 导入 ThreadX S 文件

图 0. MDN3 您加 IIIfeaux 天义作龄	图	6.	MDK5	添加	ThreadX	头文件路征
----------------------------	---	----	------	----	---------	-------

	•••• ••• • •• ••••		1	 1		1
Prepro Fo	older Setup			?	×	
Def Se	tup Compiler Include Paths:			 <u>۲</u>)	
Under	\Templates			 		
	\\IreadX\ports\cortex_m4\a \\\TreadX\common\inc	c6\inc				
	\\Libraries\drivers\inc					
Exe	\\\	e_support support				
Optimiz	\\at32f403a_407_board					-
🗆 Link						r
🗆 Spli						
One One						
Inclu						i
M						_
Contr						
Comp						^
cont		ОК	Cancel			
su				 		~

STEP 6 头文件地址添加



STEP 7 设置 ASM 汇编选项

	get Output Listing User C/C++ (AC6) Asm Linker Debug Vtilities	
- Conditiona	Assembly Control Symbols	_
D C		
Define:		
Undefine		
	/ Code Generation	_
Assembler	Option: amclang (Auto Select)	
Read-	Only Position Independent Split Load and Store Multiple	
Read-	Write Position Independent	
Thumb) Mode	
🔲 No Wa	amings 📃 No Auto Includes	
Include Paths	\\\TreadX\ports\cortex_m4\keil\inc	
Misc		
Controls		
Assembler	-target=am-am-none-eabi -mcpu=cortex-m4 -mfpu=fpv4-sp-d16 -mfloat-abi=hard -masm=auto -c	
string	-gowair-3 -vva,amasm,-po,MICROLIB SETAT -1//./Tread/ports/cortex_m4/keii/inc	

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_in	itialize_low_level.S		
22 @	1		
23 @			
24	.global	_tx_thread_system_stack_ptr	
25	.global	tx_initialize_unused_memory	
26	.global	_tx_timer_interrupt	
27	.global	main	
28	.global	tx_SVCallHandler	
29	.global	tx_PendSVHandler	
30	.global	tx_NMIHandler	@ NMI
31	.global	tx_BadHandler	@ HardFault
32	.global	tx_SVCallHandler	@ SVCall
33	.global	tx_DBGHandler	@ Monitor
34	.global	tx_PendSVHandler	@ PendSV
35	.global	tx_SysTickHandler	@ SysTick
36	.global	tx_IntHandler	@ Int O
37	.global	_initial_sp	
38	.global	Vectors	
39 (
40 @			
41 S	YSTEM_CLOCK	= 24000000	
42 S	YSTICK_CYCLES	<pre>= ((SYSTEM_CLOCK / 1000) -1)</pre>	
43			
44	.text 32		
45	.align 4		
46	.syntax uni	ified	

图 9. 修改__initial_sp/ Vectors 语句

	1120_10W_10V01.5		
94 @	/* Disab	le interrupts during ThreadX initi	alization. */
95 @			
96	CPSID i		
97 @			
98 0	/* Set b	ase of available memory to end of	non-initialised RAM area. */
990			
100	LDR r	0, = tx_initialize_unused_memory	@ Build address of unused memory pointer
101	LDR r.	1, =initial_sp	0 Build first free address
102	ADD r.	1, T1, #4	
103	STR r.	1, [r0]	@ Setup first unused memory pointer
104 0	(* C-+	Washing Washing Office Designed #/	
105 0	/* Setup	vector lable Offset Register. */	
108 6	MOM	0 #0×E000E000	A Build address of NVIC registers
108	LDR r	1 = Vectors	@ Pickup address of vector table
109	STR r	1, [r0, #0xD081	@ Set vector table address
110 @		-, [, "]	
111 @	/* Set s	vstem stack pointer from vector va	lue. */
112 @			
113	LDR r	0, = tx thread system stack ptr	@ Build address of system stack pointer
114	LDR r	1, = Vectors	@ Pickup address of vector table
115	LDR r	1, [r1]	<pre>@ Pickup reset stack pointer</pre>
116	STR r	1, [r0]	@ Save system stack pointer
117 @			
118 @	/* Enabl	e the cycle count register. */	
119 @			
120	LDR r	0, =0xE0001000	@ Build address of DWT register
121	LDR r	1, [r0]	@ Pickup the current value
122	ORR r	1, r1, #1	@ Set the CYCCNTENA bit
123	STR r	1, [r0]	@ Enable the cycle count register
124 @			
125 @	/* Confi	gure SysTick for 100Hz clock, or 1	.6384 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
        * @retval none
 111
 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)
ł
   /* 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. */
```

475<u>-</u>751;

```
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! */
Т
     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

	_	_		^
nit ok ureadX rtos on AT-SART-Board	\sim	串口选择		
readx 2 application running readx 1 application running		COM20:USB	-SERIAL	~
readx 2 application running readx 2 application running readx 1 application running		波特率	115200	~
readx 2 application running		停止位	1	~
		数据位	8	~
		奇偶校验	无	~
		串口操作	·沃 💮	那串口
		保存窗口	清除	接收
			記示□白	底黑字
		RTS	DT:	R
	5	🗌 时间戳((以换行回:	车断帧



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。

© 2021 雅特力科技 保留所有权利