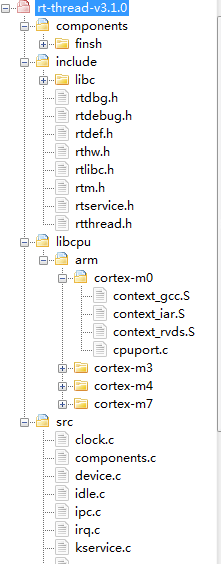
RT-Thread移植

一、移植内核文件

以rtt3.1.0为例，只需下面所示这些，组件暂时只有finsh。



二、RTT移植步骤

1、内核源码添加：将rt-thread-v3.1.0\src的源码添加到IAR/keil/gcc编译环境的工程中

2、**移植相关的cpu文件**：将rt-thread-v3.1.0\ libcpu下的各类型的芯片架构的context\_xxx.s和cpuport.c的源码添加到IAR/keil/gcc编译环境的工程中

3、系统时钟节拍接口提供：移植任何一款RTOS，其心跳得提供。

在core\_cm0.h、core\_cm3.h和core\_cm4.h内核头文件中可以查到一模一样的systick中断配置，如下函数：

\_\_STATIC\_INLINE uint32\_t SysTick\_Config(uint32\_t ticks)

{

if ((ticks - 1) > SysTick\_LOAD\_RELOAD\_Msk) return (1); /\* Reload value impossible \*/

SysTick->LOAD = ticks - 1; /\* set reload register \*/

NVIC\_SetPriority (SysTick\_IRQn, (1<<\_\_NVIC\_PRIO\_BITS) - 1); /\* set Priority for Systick Interrupt \*/

SysTick->VAL = 0; /\* Load the SysTick Counter Value \*/

SysTick->CTRL = SysTick\_CTRL\_CLKSOURCE\_Msk |

SysTick\_CTRL\_TICKINT\_Msk |

SysTick\_CTRL\_ENABLE\_Msk; /\* Enable SysTick IRQ and SysTick Timer \*/

return (0); /\* Function successful \*/

}

所以对m0/m3/m4，对**RTT的systick初始化**很便利：

**（1）**在RTT的rt\_hw\_board\_init函数中进行systick初始化，如下所示。

/\*其中CntValue 是当前m0/m3/m4的时钟频率\*/

/\*\*

\* @brief 启动系统滴答定时器 SysTick

\* @param 无

\* @retval 无

\*/

void SysTick\_Init(void)

{

/\* systick中断自适应

\*SystemFrequency / 1000 1ms中断1次

\* SystemFrequency / 100000 10us中断1次

\* SystemFrequency / 1000000 1us中断1次

\*/

RCC\_ClocksTypeDef RccClocks;

uint32\_t SystemClockFrequency = 0;

RCC\_GetClocksFreq(&RccClocks);

SystemClockFrequency = RccClocks.HCLK\_Frequency;

while (SysTick\_Config(SystemClockFrequency/RT\_TICK\_PER\_SECOND));

}

**（2）systick中断一般**放到board.c中，其他c文件也行。

void SysTick\_Handler(void)

{

/\* 进入中断 \*/

rt\_interrupt\_enter();

/\* 更新时基 \*/

rt\_tick\_increase();

/\* 离开中断 \*/

rt\_interrupt\_leave();

}

这就完成了systcik的配置

4.串口配置。

想用kprintf需要改写，这个还在探索。