ARC

FreeRTOS

支持	www.armrunc.com http://armrunc.taobao.com

Change History

版本号	更改描述	作者	修改时间
1.0.0	第一版	armrunc	2012-03-04

目录

第19章 FreeRTOS

19.1 FreeRTOS 简介

FreeRTOS 是一个可移植的、开源的小型实时内核，可以免费下载并且可以免费用于商业软件。FreeRTOS 支持多种处理器架构，包括 ARM7、ARM9、Cortex-M3、AVR、PIC 和 MSP430 等。FreeRTOS 系统的主要特点如下：

- 完全免费，使用没有限制，包括商业应用。
- 优先级调度，相同优先级任务可轮转调度，同时可设置为可波多内核或者不可剥夺内核；
- 任务可选择是否共享堆栈，并且没有任务数目限制；
- 消息队列，二值信号量，计数信号量，递归互斥体；
- 时间管理；
- 内存管理。

19.2 FreeRTOS 应用实例 ----- 创建任务

19.2.1 实例描述

本实例移植 FreeRTOS 到 ARC 开发板。为了是移植过程清晰，我们只创建了一个任务，这个任务只实现了最简单的跑马灯功能。

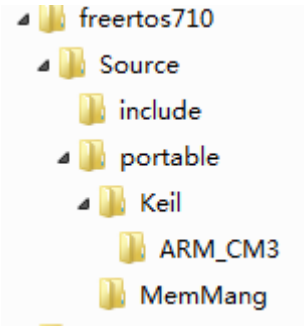
19.2.2 硬件设计

该实例用到了 ARC 平台上的两个 LED 灯，电路图参考 LED 那一章节。

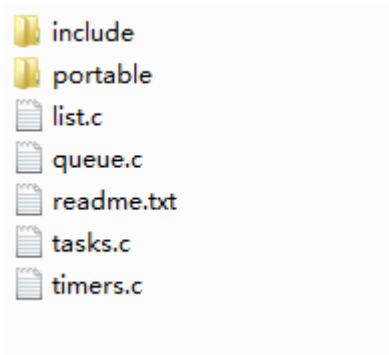
19.2.3 软件设计

首先到 FreeRTOS 官方网站 http://www.freertos.org/ 下载软件，该实例基于版本 7.1.0。

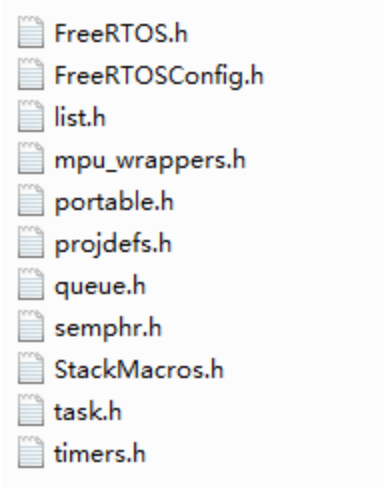
在本地创建一个 freertos710 目录，目录结构如下：



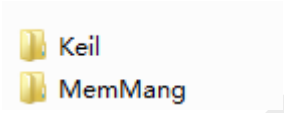
拷贝 FreeRTOS 以下文件到 Source 文件夹。



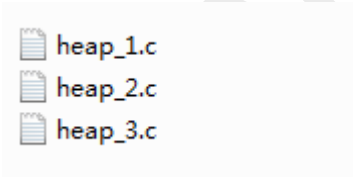
拷贝 FreeRTOS 以下文件到 freertos710\Source\include 文件夹。



在 freertos710\Source\portable\文件夹中创建以下文件夹。

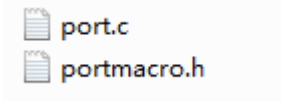


拷贝 FreeRTOS 以下文件到 freertos710\Source\portable\MemMang 文件夹。



在 freertos710\Source\portable\Keil 文件内创建 ARM_CM3 文件。

拷贝 FreeRTOS 以下文件到 freertos710\Source\portable\Keil\ARM_CM3 文件夹。



针对 STM32 CORTEX M3 内核，修改 port.c 文件，具体更改如下：

- 增加 void SVC_Handler(void) __attribute__ ((alias ("vPortSVCHandler"))); 使得 SVC_Handler 是 vPortSVCHandler 的别名，也就是他们两个实际是一个函数。

- 增加 void PendSV_Handler(void) __attribute__ ((alias ("xPortPendSVHandler")));
- 增加 void SysTick_Handler(void) __attribute__ ((alias ("xPortSysTickHandler")));

具体代码请参考文件 port.c。

在文件 freertos_main.c 内创建了任务，并开始任务调度，具体代码如下：

```
/**
  * @brief  Main program, freeRTOS immigration to ARC board example.
  * @param  None
  * @retval None
  */
int main(void)
{
    ARC_LED_Init();
    /* Create one task. */
    xTaskCreate(    vTask,      /* Pointer to the function that implements
the task. */
                    "Task 1",   /* Text name for the task.  This is to
facilitate debugging only. */
                    200,        /* Stack depth in words. */
                    NULL,       /* We are not using the task parameter. */
                    1,          /* This task will run at priority 1. */
                    NULL );     /* We are not using the task handle. */


    /* Start the scheduler so our tasks start executing. */
    vTaskStartScheduler();

    /* If all is well we will never reach here as the scheduler will now be
        running.  If we do reach here then it is likely that there was insufficient
        heap available for the idle task to be created. */
    while (1)
    {
    }
}
```

该任务的实现在文件 ARC_FreeRTOS.c：具体代码如下：

```
/**
  * @brief  Toggle LED0/LED1.
  * @param  *pvParameters: the task parameters.
  * @retval None
  */
void vTask( void *pvParameters )
```

```
{
    /* As per most tasks, this task is implemented in an infinite loop. */
    while(1)
    {
        ARC_LED_Toggle(0);
        ARC_LED_Toggle(1);
        vTaskDelay(1000 / portTICK_RATE_MS);
    }
}
```

文件 port.c

```
/*
    FreeRTOS V7.1.0 - Copyright (C) 2011 Real Time Engineers Ltd.


    ***************************************************************************
     *
*
     *    FreeRTOS tutorial books are available in pdf and paperback.
*
     *    Complete, revised, and edited pdf reference manuals are also
*
     *    available.
*
     *
*
     *    Purchasing FreeRTOS documentation will not only help you, by
*
     *    ensuring you get running as quickly as possible and with an
*
     *    in-depth knowledge of how to use FreeRTOS, it will also help
*
     *    the FreeRTOS project to continue with its mission of providing
*
     *    professional grade, cross platform, de facto standard solutions
*
     *    for microcontrollers - completely free of charge!
*
     *
*
     *    >>> See http://www.FreeRTOS.org/Documentation for details. <<<
*
     *
*
```

```
    *      Thank you for using FreeRTOS, and thank you for your support!
*
    *
*
    ***************************************************************************


    This file is part of the FreeRTOS distribution.

    FreeRTOS is free software; you can redistribute it and/or modify it under
    the terms of the GNU General Public License (version 2) as published by
the
    Free Software Foundation AND MODIFIED BY the FreeRTOS exception.
    >>>NOTE<<< The modification to the GPL is included to allow you to
    distribute a combined work that includes FreeRTOS without being obliged
to
    provide the source code for proprietary components outside of the
FreeRTOS
    kernel.  FreeRTOS is distributed in the hope that it will be useful, but
    WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY
    or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General
Public License for
    more details. You should have received a copy of the GNU General Public
    License and the FreeRTOS license exception along with FreeRTOS; if not
it
    can be viewed here: http://www.freertos.org/a00114.html and also
obtained
    by writing to Richard Barry, contact details for whom are available on the
    FreeRTOS WEB site.

    1 tab == 4 spaces!

    http://www.FreeRTOS.org - Documentation, latest information, license
and
    contact details.

    http://www.SafeRTOS.com - A version that is certified for use in safety
    critical systems.

    http://www.OpenRTOS.com - Commercial support, development, porting,
    licensing and training services.
*/
```

```
/*-----------------------------------------------------------
 * Implementation of functions defined in portable.h for the ARM CM3 port.
 *----------------------------------------------------------*/

/* Scheduler includes. */
#include "FreeRTOS.h"
#include "task.h"

#ifndef configKERNEL_INTERRUPT_PRIORITY
    #define configKERNEL_INTERRUPT_PRIORITY 255
#endif

/* Constants required to manipulate the NVIC. */
#define portNVIC_SYSTICK_CTRL       ( ( volatile unsigned long *) 0xe000e010 )
#define portNVIC_SYSTICK_LOAD       ( ( volatile unsigned long *) 0xe000e014 )
#define portNVIC_INT_CTRL           ( ( volatile unsigned long *) 0xe000ed04 )
#define portNVIC_SYSPRI2            ( ( volatile unsigned long *) 0xe000ed20 )
#define portNVIC_SYSTICK_CLK        0x00000004
#define portNVIC_SYSTICK_INT        0x00000002
#define portNVIC_SYSTICK_ENABLE     0x00000001
#define portNVIC_PENDSVSET          0x10000000
#define portNVIC_PENDSV_PRI         ( ( ( unsigned long ) configKERNEL_INTERRUPT_PRIORITY ) << 16 )
#define portNVIC_SYSTICK_PRI        ( ( ( unsigned long ) configKERNEL_INTERRUPT_PRIORITY ) << 24 )

/* Constants required to set up the initial stack. */
#define portINITIAL_XPSR            ( 0x01000000 )

/* Each task maintains its own interrupt status in the critical nesting variable. */
static unsigned portBASE_TYPE uxCriticalNesting = 0xaaaaaaaa;

/*
 * Setup the timer to generate the tick interrupts.
 */
static void prvSetupTimerInterrupt( void );

/*
 * Start first task is a separate function so it can be tested in isolation.
```

```
 */
void vPortStartFirstTask( void );

/*-----------------------------------------------------------*/

/*
 * See header file for description.
 */
portSTACK_TYPE *pxPortInitialiseStack( portSTACK_TYPE *pxTopOfStack,
pdTASK_CODE pxCode, void *pvParameters )
{
    /* Simulate the stack frame as it would be created by a context switch
    interrupt. */
    pxTopOfStack--; /* Offset added to account for the way the MCU uses the
stack on entry/exit of interrupts. */
    *pxTopOfStack = portINITIAL_XPSR;    /* xPSR */
    pxTopOfStack--;
    *pxTopOfStack = ( portSTACK_TYPE ) pxCode;     /* PC */
    pxTopOfStack--;
    *pxTopOfStack = 0;/* LR */
    pxTopOfStack -= 5;/* R12, R3, R2 and R1. */
    *pxTopOfStack = ( portSTACK_TYPE ) pvParameters;  /* R0 */
    pxTopOfStack -= 8;/* R11, R10, R9, R8, R7, R6, R5 and R4. */

    return pxTopOfStack;
}
/*-----------------------------------------------------------*/

__asm void vPortSVCHandler( void )
{
    PRESERVE8

    ldr  r3, =pxCurrentTCB       /* Restore the context. */
    ldr r1, [r3]                 /* Use pxCurrentTCBConst to get the
pxCurrentTCB address. */
    ldr r0, [r1]                 /* The first item in pxCurrentTCB is the task top of
stack. */
    ldmia r0!, {r4-r11}          /* Pop the registers that are not automatically
saved on exception entry and the critical nesting count. */
    msr psp, r0                      /* Restore the task stack pointer. */
    mov r0, #0
    msrbasepri, r0
    orr r14, #0xd
    bx r14
```

```
}
/*-----------------------------------------------------------*/

__asm void vPortStartFirstTask( void )
{
	PRESERVE8

	/* Use the NVIC offset register to locate the stack. */
	ldr r0, =0xE000ED08
	ldr r0, [r0]
	ldr r0, [r0]
	/* Set the msp back to the start of the stack. */
	msr msp, r0
	/* Globally enable interrupts. */
	cpsie i
	/* Call SVC to start the first task. */
	svc 0
	nop
}
/*-----------------------------------------------------------*/

/*
 * See header file for description.
 */
portBASE_TYPE xPortStartScheduler( void )
{
	/* Make PendSV, CallSV and SysTick the same priroity as the kernel. */
	*(portNVIC_SYSPRI2) |= portNVIC_PENDSV_PRI;
	*(portNVIC_SYSPRI2) |= portNVIC_SYSTICK_PRI;

	/* Start the timer that generates the tick ISR.  Interrupts are disabled
	here already. */
	prvSetupTimerInterrupt();

	/* Initialise the critical nesting count ready for the first task. */
	uxCriticalNesting = 0;

	/* Start the first task. */
	vPortStartFirstTask();

	/* Should not get here! */
	return 0;
}
/*-----------------------------------------------------------*/
```

```
void vPortEndScheduler( void )
{
    /* It is unlikely that the CM3 port will require this function as there
    is nothing to return to.  */
}
/*-----------------------------------------------------------*/

void vPortYieldFromISR( void )
{
    /* Set a PendSV to request a context switch. */
    *(portNVIC_INT_CTRL) = portNVIC_PENDSVSET;
}
/*-----------------------------------------------------------*/

void vPortEnterCritical( void )
{
    portDISABLE_INTERRUPTS();
    uxCriticalNesting++;
}
/*-----------------------------------------------------------*/

void vPortExitCritical( void )
{
    uxCriticalNesting--;
    if( uxCriticalNesting == 0 )
    {
        portENABLE_INTERRUPTS();
    }
}
/*-----------------------------------------------------------*/

__asm void xPortPendSVHandler( void )
{
    extern uxCriticalNesting;
    extern pxCurrentTCB;
    extern vTaskSwitchContext;

    PRESERVE8

    mrs r0, psp

    ldr  r3, =pxCurrentTCB          /* Get the location of the current TCB. */
    ldr  r2, [r3]
```

```
	stmdb r0!, {r4-r11}			/* Save the remaining registers. */
	str r0, [r2]				/* Save the new top of stack into the first
member of the TCB. */

	stmdb sp!, {r3, r14}
	mov r0, #configMAX_SYSCALL_INTERRUPT_PRIORITY
	msr basepri, r0
	bl vTaskSwitchContext
	mov r0, #0
	msr basepri, r0
	ldmia sp!, {r3, r14}

	ldr r1, [r3]
	ldr r0, [r1]				/* The first item in pxCurrentTCB is the task
top of stack. */
	ldmia r0!, {r4-r11}			/* Pop the registers and the critical nesting
count. */
	msr psp, r0
	bx r14
	nop
}
/*-----------------------------------------------------------*/

void xPortSysTickHandler( void )
{
unsigned long ulDummy;

	/* If using preemption, also force a context switch. */
	#if configUSE_PREEMPTION == 1
		*(portNVIC_INT_CTRL) = portNVIC_PENDSVSET;
	#endif

	ulDummy = portSET_INTERRUPT_MASK_FROM_ISR();
	{
		vTaskIncrementTick();
	}
	portCLEAR_INTERRUPT_MASK_FROM_ISR( ulDummy );
}
/*-----------------------------------------------------------*/

/*
 * Setup the systick timer to generate the tick interrupts at the required
 * frequency.
```

```
  */
void prvSetupTimerInterrupt( void )
{
	/* Configure SysTick to interrupt at the requested rate. */
	*(portNVIC_SYSTICK_LOAD) = ( configCPU_CLOCK_HZ /
configTICK_RATE_HZ ) - 1UL;
	*(portNVIC_SYSTICK_CTRL) = portNVIC_SYSTICK_CLK |
portNVIC_SYSTICK_INT | portNVIC_SYSTICK_ENABLE;
}
/*-----------------------------------------------------------*/

__asm void vPortSetInterruptMask( void )
{
	PRESERVE8

	push { r0 }
	mov r0, #configMAX_SYSCALL_INTERRUPT_PRIORITY
	msr basepri, r0
	pop { r0 }
	bx r14
}

/*-----------------------------------------------------------*/

__asm void vPortClearInterruptMask( void )
{
	PRESERVE8

	push { r0 }
	mov r0, #0
	msr basepri, r0
	pop { r0 }
	bx r14
}

void SVC_Handler( void ) __attribute__ (( alias ("vPortSVCHandler")));
void PendSV_Handler( void ) __attribute__ (( alias ("xPortPendSVHandler")));
void SysTick_Handler( void ) __attribute__ (( alias ("xPortSysTickHandler")));
```