|  |
| --- |
|  |
| **EE489 / EE589 Real-Time Embedded Systems Design** |
| Lab #7  *Dianzhi Yu* |
|  |
| *3/18/2018* |

|  |
| --- |
|  |

# Introduction

This lab project reviews how to create tasks and gets information of created tasks. Create two tasks, one is make LED on the board blink at a fixed rate; another task is to monitor the status of all running tasks. And modify the stack size and priority of the two created tasks and record the updated display information on the Tera term.

1. **A list of all FreeRTOS API functions being used**

#define TASK\_MONITOR\_STACK\_SIZE (2048/sizeof(portSTACK\_TYPE))

xStatus = xTaskCreate(task\_monitor, "Monitor", TASK\_MONITOR\_STACK\_SIZE, NULL, TASK\_MONITOR\_STACK\_PRIORITY + 3, NULL);

Create this new task to monitor the status of all running tasks with the priority at 3. And the stack size is 2048/ sizeof(portSTACK\_TYPE). The two parameters can be changed to see the differences on the display.

#define TASK\_LED\_STACK\_SIZE (1024/sizeof(portSTACK\_TYPE))

xStatus = xTaskCreate(task\_led, "Led", TASK\_LED\_STACK\_SIZE, NULL, TASK\_LED\_STACK\_PRIORITY + 3, NULL);

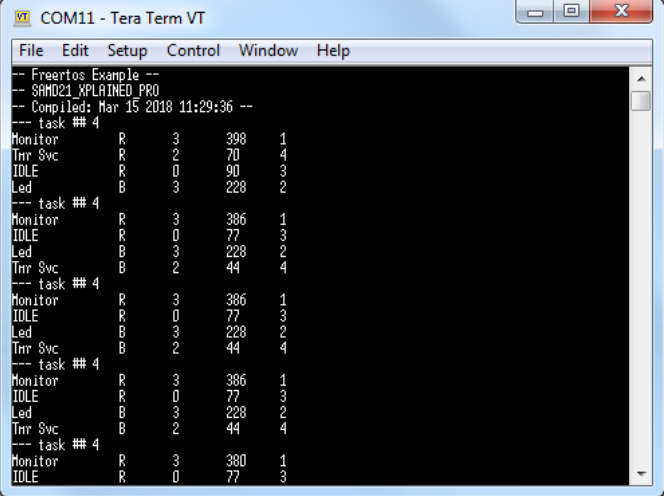
Create this task to make the led on the board blink at a fixed rate with the same priority as monitor. The stack size is 1024/sizeof(portSTACK\_TYPE).

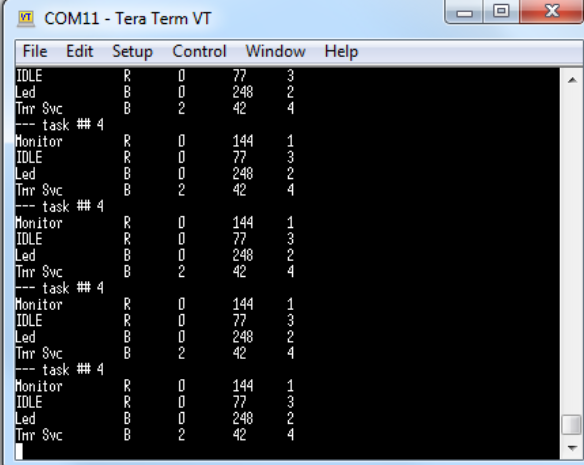
1. **Screen shots of the program execution results or debug windows of Keil µVision**

The first column is Task name; the second column is the state of each task; then, the third column is priority; the fourth column is stack size; the fifth column is thread ID number.

The first screen shot is shown that the stack size of monitor task is 2048/sizeof(portSTACK\_TYPE) with the highest priority and the stack size of led is 1024/sizeof(portSTACK\_TYPE). From the Tera Terminal, we can see the monitor is always running to monitor the led. The led will blink at a fixed rate so the led task will be blocked.

The second screen shot is the result of different stack sizes and priorities. Change the stack size of monitor task and led task.





1. **Conclusion**

It is important to set the marco definitions to configure the functions. The monitor task is on the ready state to observe other tasks. And according to the output display, we can calculate the stack size of monitor task and led task to make sure that the total stack size is enough.

1. **Appendix: The source code (main.c) with sufficient comments.**

/\*\*

\* \file

\*

\* \brief Empty user application template

\*

\*/

/\*\*

\* \mainpage User Application template doxygen documentation

\*

\* \par Empty user application template

\*

\* This is a bare minimum user application template.

\*

\* For documentation of the board, go \ref group\_common\_boards "here" for a link

\* to the board-specific documentation.

\*

\* \par Content

\*

\* -# Include the ASF header files (through asf.h)

\* -# Minimal main function that starts with a call to system\_init()

\* -# Basic usage of on-board LED and button

\* -# "Insert application code here" comment

\*

\*/

/\*

\* Include header files for all drivers that have been imported from

\* Atmel Software Framework (ASF).

\*/

/\*

\* Support and FAQ: visit <a href="http://www.atmel.com/design-support/">Atmel Support</a>

\*/

#include <asf.h>

#define TASK\_MONITOR\_STACK\_SIZE (2048/sizeof(portSTACK\_TYPE))

#define TASK\_MONITOR\_STACK\_PRIORITY (tskIDLE\_PRIORITY)

#define TASK\_LED\_STACK\_SIZE (1024/sizeof(portSTACK\_TYPE))

#define TASK\_LED\_STACK\_PRIORITY (tskIDLE\_PRIORITY)

extern void vApplicationIdleHook(void);

extern void vApplicationTickHook(void);

/\*\*

\* \brief This function is called by FreeRTOS idle task

\*/

extern void vApplicationIdleHook(void)

{

}

/\*\*

\* \brief This task, when activated, send every ten seconds on debug UART

\* the whole report of free heap and total tasks status

\*/

static void task\_monitor(void \*pvParameters)

{

static portCHAR szList[256];

UNUSED(pvParameters);

for (;;) {

printf("--- task ## %u\r\f", (unsigned int)uxTaskGetNumberOfTasks());

vTaskList((signed portCHAR \*)szList);

printf(szList);

vTaskDelay(1000);

}

}

/\*\*

\* \brief This task, when activated, make LED blink at a fixed rate

\*/

static void task\_led(void \*pvParameters)

{

UNUSED(pvParameters);

for (;;) {

port\_pin\_toggle\_output\_level(LED\_0\_PIN);

vTaskDelay(1000);

}

}

/\*\* UART module for debug. \*/

static struct usart\_module cdc\_uart\_module;

/\*\*

\* \brief Configure UART console.

\*/

static void configure\_console(void)

{

struct usart\_config usart\_conf;

usart\_get\_config\_defaults(&usart\_conf);

usart\_conf.mux\_setting = EDBG\_CDC\_SERCOM\_MUX\_SETTING;

usart\_conf.pinmux\_pad0 = EDBG\_CDC\_SERCOM\_PINMUX\_PAD0;

usart\_conf.pinmux\_pad1 = EDBG\_CDC\_SERCOM\_PINMUX\_PAD1;

usart\_conf.pinmux\_pad2 = EDBG\_CDC\_SERCOM\_PINMUX\_PAD2;

usart\_conf.pinmux\_pad3 = EDBG\_CDC\_SERCOM\_PINMUX\_PAD3;

usart\_conf.baudrate = 115200;

stdio\_serial\_init(&cdc\_uart\_module, EDBG\_CDC\_MODULE, &usart\_conf);

usart\_enable(&cdc\_uart\_module);

}

int main (void)

{

portBASE\_TYPE xStatus;

system\_init();

/\* Initialize the console uart \*/

configure\_console();

/\* Output demo infomation. \*/

printf("-- Freertos Example --\n\r");

printf("-- %s\n\r", BOARD\_NAME);

printf("-- Compiled: %s %s --\n\r", \_\_DATE\_\_, \_\_TIME\_\_);

/\* Create task to monitor processor activity \*/

xStatus = xTaskCreate(task\_monitor, "Monitor", TASK\_MONITOR\_STACK\_SIZE, NULL, TASK\_MONITOR\_STACK\_PRIORITY + 3, NULL);

if ( xStatus != pdPASS) {

printf("Failed to create Monitor task\r\n");

}

/\* Create task to make led blink \*/

xStatus = xTaskCreate(task\_led, "Led", TASK\_LED\_STACK\_SIZE, NULL, TASK\_LED\_STACK\_PRIORITY + 3, NULL); //

if ( xStatus != pdPASS) {

printf("Failed to create test led task\r\n");

}

/\* Start the scheduler. \*/

vTaskStartScheduler();

/\* Will only get here if there was insufficient memory to create the idle task. \*/

return 0;

}