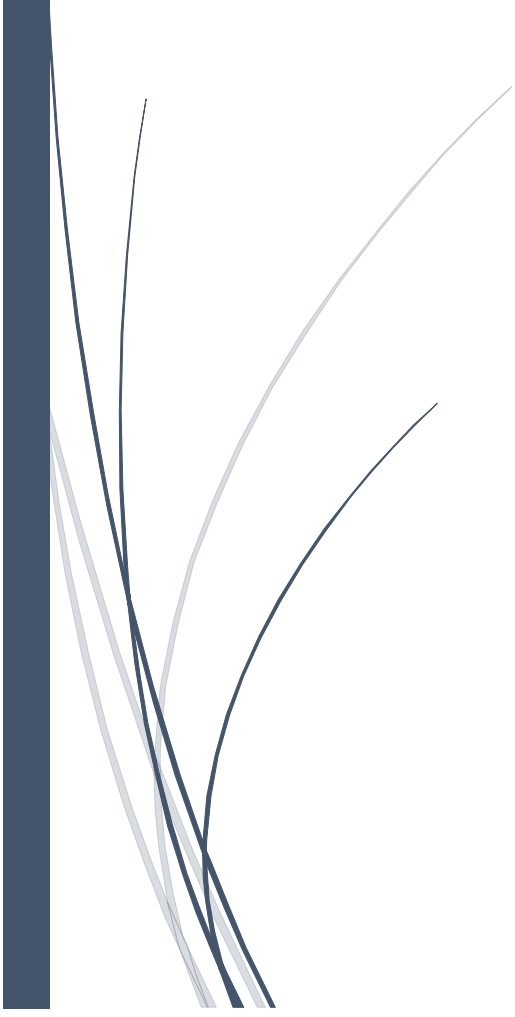
14/01/2024

Assignment RTOS



Vincent Soucourre

Table des matières

Abstract

This report presents the implementation of a Real-Time Operating System (RTOS) using FreeRTOS for an embedded system. The objective is to develop a reliable task scheduling mechanism with periodic functionalities. Key tasks include temperature conversion, large number multiplication, and inter-task communication via a queue. The analysis involves determining Worst Case Execution Times (WCET), selecting appropriate task periods and priorities, and ensuring the schedulability of the system.

Introduction

The adoption of Real-Time Operating Systems (RTOS) is crucial for ensuring timely execution of tasks. This report delves into the design and analysis of an RTOS using FreeRTOS, focusing on effective task scheduling for tasks with periodic requirements. Task scheduling is a critical aspect of embedded systems, especially those with strict real-time constraints.

The analysis aims to determine the Worst-Case Execution Times (WCET) for each task, ensuring that their execution fits within predefined time frames. Additionally, careful consideration is given to task periods and priorities, aligning them with the real-time requirements of the system. The schedulability of the entire task set is evaluated using the Rate Monotonic Scheduling (RMS) algorithm, a common approach in real-time systems design.

Method

Task Execution Time Analysis

For each task we will determine the WCET. For that we will compute the execution time of each task and collect the maximum execution time. The highest values will be the Worst-case execution time.

```
Temperature time of execution: 0.193937 seconds, WCET: 0.150000 seconds: WCET ex
ceeded.
```

In this case we choose a WCET of 0.15 seconds. If the execution time is higher than the WCET, the message WCET exceed will be print. Else, the message “task successfully completed” will be print.

```
Temperature time of execution: 0.188646 seconds, WCET: 0.250000 seconds: Task su
ccessfully completed.
```

In case of the task cannot be executed because the WCET is overpassed, the algorithm stopping completely.

Task Periods and Priorities

To schedule our system, we have determined different frequency for each task. The task with the highest frequency will be the first to be execute.

```
/* Priorities at which the tasks are created. */
#define mainQUEUE_RECEIVE_TASK_PRIORITY    ( tskIDLE_PRIORITY + 2 )
#define mainQUEUE_SEND_TASK_PRIORITY       ( tskIDLE_PRIORITY + 1 )
#define mainTEMPERATURE_CONVERSION_TASK_PRIORITY  ( tskIDLE_PRIORITY + 3 )
#define mainMULTIPLICATION_TASK_PRIORITY   ( tskIDLE_PRIORITY + 4 )
#define mainBINARY_SEARCH_TASK_PRIORITY      ( tskIDLE_PRIORITY + 5 )

/* The rate at which data is sent to the queue.  The times are converted from
 * milliseconds to ticks using the pdMS_TO_TICKS() macro. */
#define mainTASK_SEND_FREQUENCY_MS         pdMS_TO_TICKS(200UL )
#define mainTIMER_SEND_FREQUENCY_MS        pdMS_TO_TICKS(400UL )
#define mainTEMPERATURE_CONVERSION_FREQUENCY_MS   pdMS_TO_TICKS(800UL )
#define mainMULTIPLICATION_TASK_PERIOD_MS   pdMS_TO_TICKS(1600UL)
#define mainBINARY_SEARCH_TASK_PERIOD_MS      pdMS_TO_TICKS(3200UL)
/* The number of items the queue can hold at once. */
```

TASK	WCET	Priority
Receive		1
Temperature conversion	0.25 seconds	2
Multiplication	0.40 seconds	3
Binary search	0.45 seconds	4

Task description

Our tasks are defined in three phases: first, we define the various parameters required for the task, such as temperature, numbers to multiply, etc.

```
/* Fixed Fahrenheit temperature value to be converted. */
#define mainFIXED_FAHRENHEIT_TEMPERATURE   104.6

/* Two long integers to be multiplied */
#define mainBIG_NUMBER_1   5678
#define mainBIG_NUMBER_2   3456

/* For task 4 */
static const char* binaryWord = "tasktoexecute";
static const char searchElement = 'x';

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

/* The tasks as described in the comments at the top of this file. */
static void prvQueueReceiveTask(void *pvParameters);
static void prvQueueSendTask(void *pvParameters);
static void prvTemperatureConversionTask(void *pvParameters);
static void prvMultiplicationTask(void *pvParameters);
static void prvBinarySearchTask(void *pvParameters);
```

Then, for each task, we incorporate them into the `main_blinky` function to enqueue them

```
xTaskCreate(prvQueueReceiveTask, "Rx", configMINIMAL_STACK_SIZE, NULL, mainQUEUE_RECEIVE_TASK_PRIORITY, NULL);

xTaskCreate(prvQueueSendTask, "TX", configMINIMAL_STACK_SIZE, NULL, mainQUEUE_SEND_TASK_PRIORITY, NULL);

xTaskCreate(prvTemperatureConversionTask,"TempConv",configMINIMAL_STACK_SIZE,NULL,mainTEMPERATURE_CONVERSION_TASK_PRIORITY,NULL);

xTaskCreate(prvMultiplicationTask,"Multiplication",configMINIMAL_STACK_SIZE,NULL,mainMULTIPLICATION_TASK_PRIORITY,NULL);

xTaskCreate(prvBinarySearchTask, "BinarySearch", configMINIMAL_STACK_SIZE, NULL, mainBINARY_SEARCH_TASK_PRIORITY, NULL);
```

Finally, we implement the functions as required.

```
static void prvTemperatureConversionTask(void *pvParameters)
{
    const TickType_t xTaskFrequency = mainTEMPERATURE_CONVERSION_FREQUENCY_MS;
    TickType_t xNextWakeTime = xTaskGetTickCount();

    (void)pvParameters;

    for (;;)
    {
        clock_t start_time = clock();
        vTaskDelayUntil(&xNextWakeTime, xTaskFrequency);


        double celsius = (mainFIXED_FAHRENHEIT_TEMPERATURE - 32) * 5 / 9;
        console_print("Temperature: %.2f°F = %.2f°C\n", mainFIXED_FAHRENHEIT_TEMPERATURE, celsius);
        clock_t end_time = clock();
        double execution_time_temp = ((double)(end_time - start_time)) / CLOCKS_PER_SEC;
        double WCET_temp=0.25;
```

Results

Finally, our final RTOS look like that:

```
Binary search: Element 'x' found at position 7 in the word 'tasktoexecute'
Binary search time of execution: 0.000002 seconds, WCET: 0.450000 seconds: Task
successfully completed.
Bignumber time of execution: 0.387088 seconds, WCET: 0.400000 seconds: Task succ
essfully completed.
Temperature: 104.60°F = 40.33°C
Temperature time of execution: 0.187119 seconds, WCET: 0.250000 seconds: Task su
ccessfully completed.
Good Working
Working
Working
Good Working
Working
Working
```

We can see that different information are printed, like the WCET for each task etc..