UDACITY

Ministry of Communications and Information Technology

EDF Scheduler Implementation using FreeRTOS

Verification Report

Submitted by:

Ahmed Essam Salem

ahmed-medo1999@hotmail.com

Contents

Introduction

Embedded systems technological evolution has made the execution of increasingly complex applications possible. Due to this increasing complexity, the adoption of an operating system to manage the interaction between tasks and their scheduling is becoming preferable and even necessary, also for little embedded systems. This thesis examines FreeRTOS scheduler. FreeRTOS is a real time operating system specially developed for small embedded systems. After an in-depth description of the priority-based scheduler adopted by FreeRTOS, two new schedulers are proposed: the first one is based on the well-known Earliest Deadline First algorithm (EDF).

In this report, we will verify our system implementation with the EDF scheduler using the following methods:
"1. Using analytical methods to calculate the system hyperperiod, the CPU load for the given set of tasks. Also, check system schedulability using URM and time demand analysis techniques (Assuming the given set of tasks are scheduled using a fixed priority rate -monotonic scheduler)

For all the tasks we should calculate the execution time from the actual implemented tasks using GPIOs and the logic analyzer.

"2. Using Simso offline simulator, simulating the given set of tasks assuming fixed priority rate monotonic scheduler.

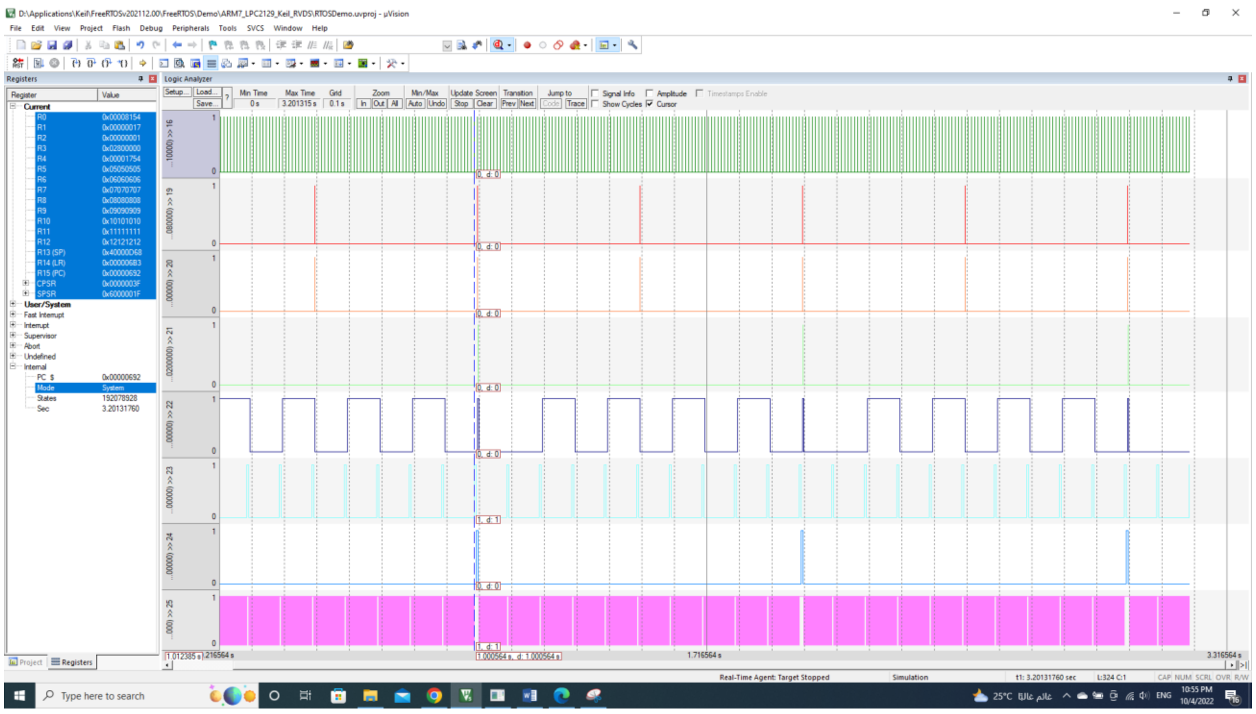
"3. Using Keil simulator in run-time and perform the following on given set of tasks:

- Calculate the CPU usage time using timer 1 and trace macros.
- Using trace macros and GPIOs, plot the execution of all tasks, tick, and the idle task on the logic analyzer.

Calculations using Analytical Methods

1. System Hyperperiod

The Hyperperiod time is the period that the scheduler pattern is repeated as well as it is equal to Least Common Multipler of all task periodicities (H = LCM(Pi)).


D:\Applications\Keil\FreeRTOSv202112.00\FreeRTOS\Demo\ARM7_LPC2129_Keil_RVDS\RTOSDemo.uvproj - µVision

Register	Value
Current	
R0	0x00008154
R1	0x00000017
R2	0x00000001
R3	0x02800000
R4	0x00001754
R5	0x05050505
R6	0x06060606
R7	0x07070707
R8	0x08080808
R9	0x09090909
R10	0x10101010
R11	0x11111111
R12	0x12121212
R13 (SP)	0x40000D68
R14 (LR)	0x000006B3
R15 (PC)	0x00000692
CPSR	0x0000003F
SPSR	0x6000001F
User/System	
Fast Interrupt	
Interrupt	
Supervisor	
Abort	
Undefined	
Internal	
PC $	0x00000692
Mode	System
States	192078928
Sec	3.20131760

Logic Analyzer: Min Time 0 s, Max Time 3.201315 s, Grid 0.1 s, Cursor ✓

Using GPIOs in actual implemented tasks and Logic Analyzer and LCM(10,20,50,50,100,100), we can deduce that the hyperperiod is 100ms.

2. CPU Load

CPU load is calculated using the following technique:

U = R/C

U = Utilization
R = Requirements which in simple terms is the BUSY TIME
C = Capacity which is simple terms is BUSY TIME + IDLE TIME

Task Name	Execution Time	Number of Occurences during Hyperperiod
Button 1 Monitor	10us	2
Button 1 Monitor	10us	2
Periodic Transmitter	9us	1
Uart Receiver	2ms	5
Load 1 Simulation	5ms	10
Load 2 Simulation	12ms	1

CPU Utilization = Total execution time of tasks during hyperperiod / Hyperperiod

= ((((10*2) + (10*2) + (9*1))/1000) + (2*5) + (5*10) + (12*1)) / 100

= 0.72 = 72%

3. System Schedulability

a) **Using Rate Monotonic Utilization Bound**

$$U = \sum_{i=1}^{n} \frac{C_i}{P_i} \leq n(2^{\frac{1}{n}} - 1)$$

U = Total Utilization
C = Execution time
P = Periodicity
N = Number of tasks

URM = 6 * [2 ^(1/6) – 1] = 0.73

U = 0.62

Therefore, U < URM

System **guarantees schedulability**.

b) Using Time-Demand Analysis

$$w_i(t) = e_i + \sum_{k=1}^{i-1} \left\lceil \frac{t}{p_k} \right\rceil e_k \quad for\ 0 < t \le p_i$$

W = Worst response time
E = Execution time
P = Periodicity
T = Time instance

Tasks list in critical instant from higher to lower priority (critical instant = 100ms):

Task	Time Provided (D) [ms]	Execution Time
Load 1 Simulation	10	5ms
Uart Receiver	20	2ms
Button 1 Monitor	50	10us
Button 2 Monitor	50	10us
Periodic Transmitter	100	9us
Load 2 Simulation	100	12ms

For Load 1 Simulation: {P: 10, E: 5, D: 10}

$W_5(10)$ = 5 + 0 = 5ms, 5 < 10

Load 1 Simulation task is schedulable.

For UART Receiver: {P: 20, E: 2, D: 20}

$W_4(20)$ = 2 + (20/10) * 5 = 5ms, 12 < 20

Uart Receiver task is schedulable.

For Button 1 Monitor: {P: 50, E: 0.010, D: 50}

$W_1(50)$ = 0.010 + (50/10) * 5 + (50/20) * 2 = 30.01ms, 30.01 < 50

Button 1 Monitor task is schedulable.

For Button 2 Monitor: {P: 50, E: 0.010, D: 50}

$W_2(50)$ = 0.010 + (50/10) * 5 + (50/20) * 2 + (50/50) * 0.01 = 30.02ms, 30.02 < 50

Button 2 Monitor task is schedulable.

For Periodic Transmitter: {P: 100, E: 0.009, D: 100}

$W_3(100)$ = 0.009 + (100/10) * 5 + (100/20) * 2 + (100/50) * 0.001 + (100/50) * 0.001 = 60.05, 60.05 < 100

Periodic Transmitter task is schedulable.

For Load 2 Simulation: {P: 100, E: 12, D: 100}

$W_6(100) = 12 + (100/10) * 5 + (100/20) * 2 + (100/50) * 0.001 + (100/50) * 0.001 + (100/100) * 0.009 = 72.05$, $72.05 < 100$

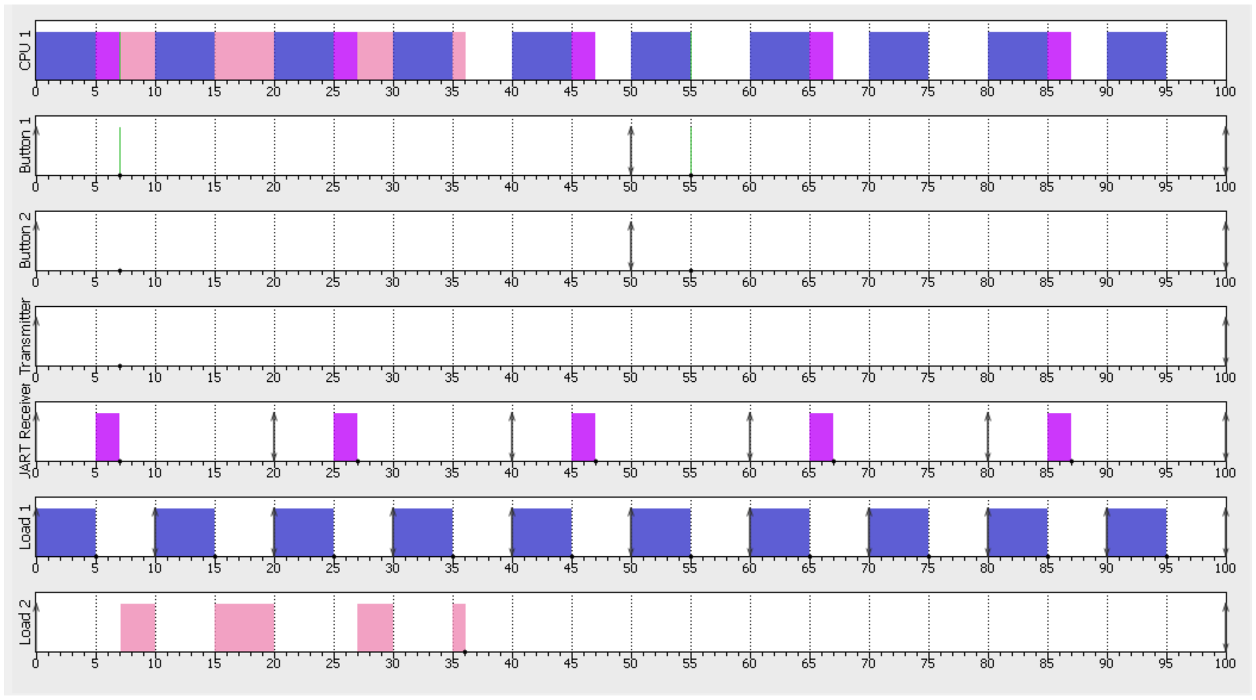
Load 2 Simulation task is schedulable.

Therefore, system **guarantees schedulability**.

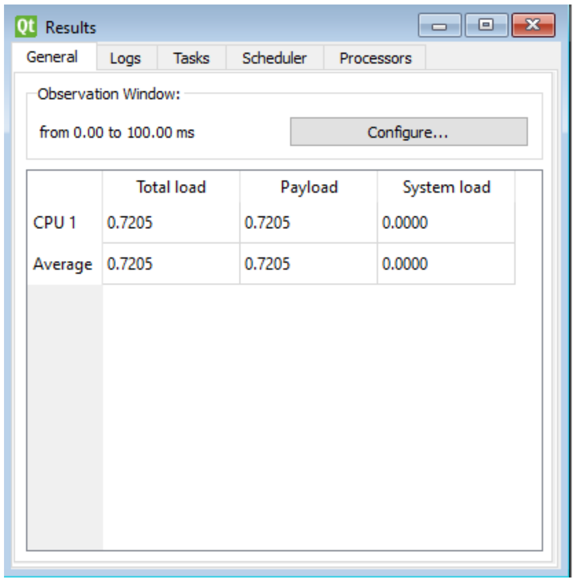
Simso Offline Simulator

- Using Fixed Priority Rate Monotonic Scheduler

Timeline:



Results:



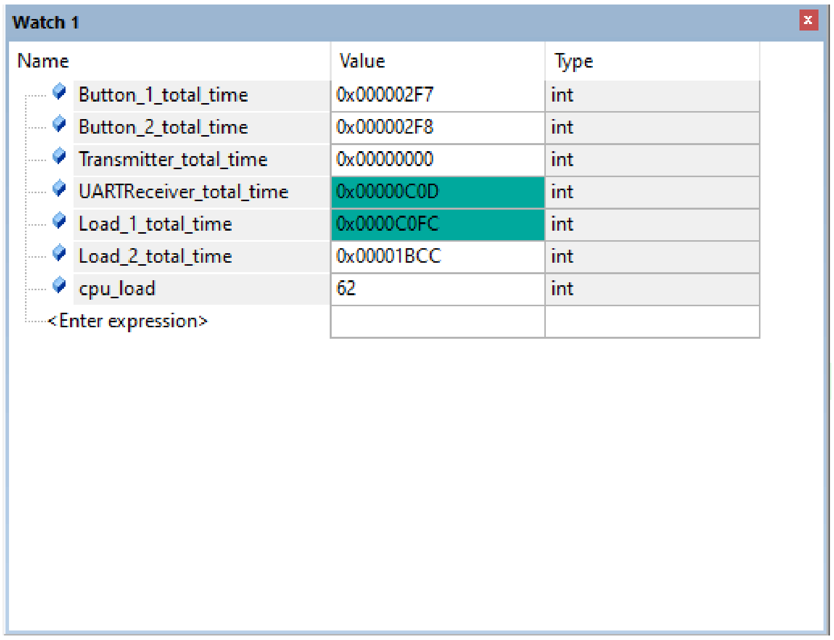
Observation Window:

from 0.00 to 100.00 ms

	Total load	Payload	System load
CPU 1	0.7205	0.7205	0.0000
Average	0.7205	0.7205	0.0000

Keil Simulator

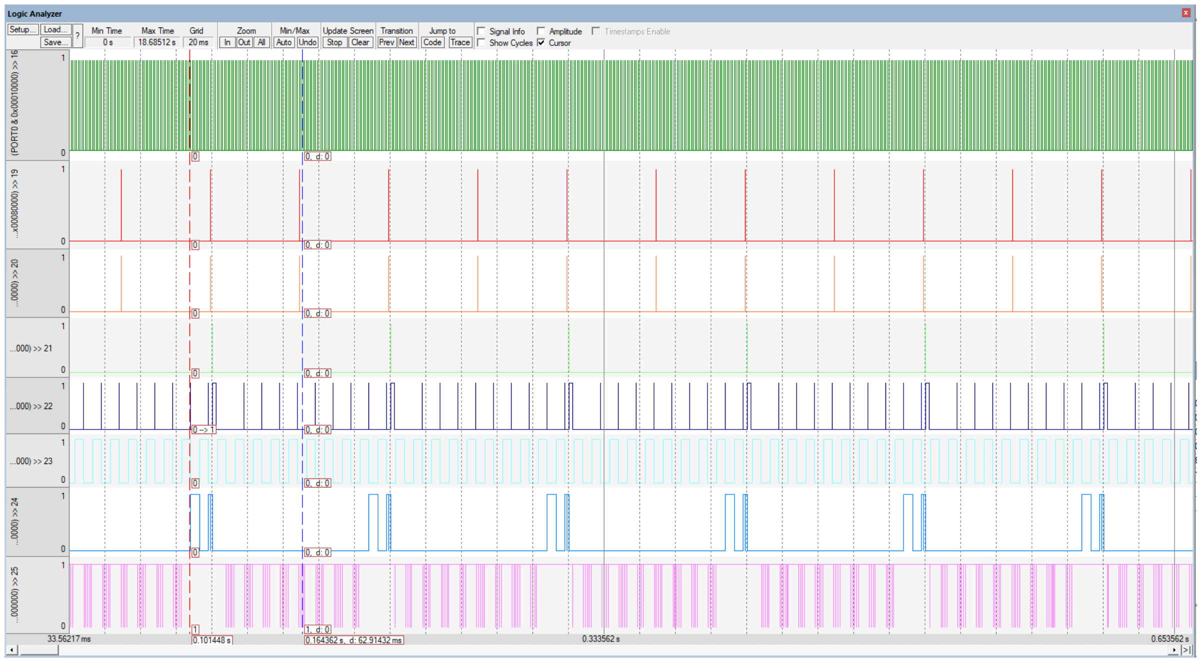
- CPU usage time using timer 1 and trace macros: 62%



Watch 1

Name	Value	Type
Button_1_total_time	0x000002F7	int
Button_2_total_time	0x000002F8	int
Transmitter_total_time	0x00000000	int
UARTReceiver_total_time	0x00000C0D	int
Load_1_total_time	0x0000C0FC	int
Load_2_total_time	0x00001BCC	int
cpu_load	62	int
<Enter expression>		

- Execution Plot of all tasks, tick, and the idle task on the logic analyzer using trace macros and GPIOs:



Comments

- **Are the results as expected ?**

  Yes, the results are as expected as CPU load is mostly the same using all analytical methods except for a 10% difference in simulation as well as system is guaranteed scheduable.

- **Does the results indicate a successful implementation ?**

  Yes, the results indicate success as all tasks don’t miss their deadline which is main objective of EDF scheduler.