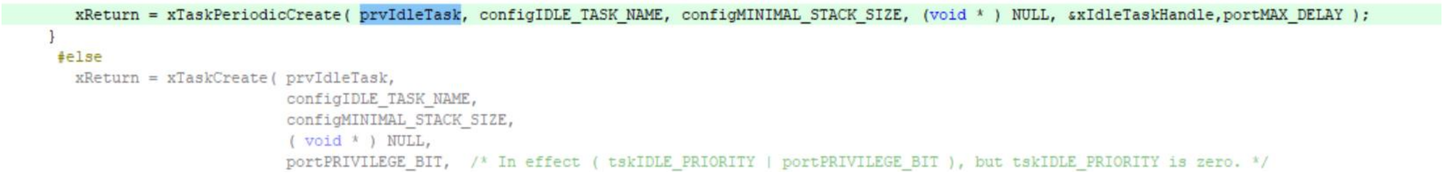
UDACITY

FREERTOS EDF Scheduler

Analytical Report

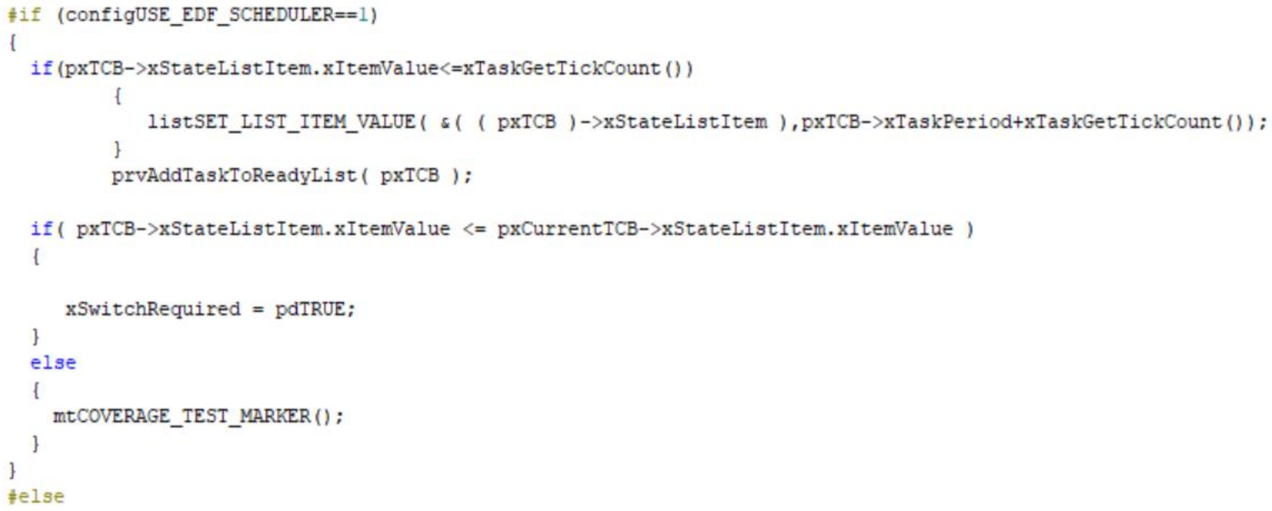
Ahmed Elebaby

Editing Part



```
    xReturn = xTaskPeriodicCreate( prvIdleTask, configIDLE_TASK_NAME, configMINIMAL_STACK_SIZE, (void * ) NULL, &xIdleTaskHandle,portMAX_DELAY );
  }
   #else
    xReturn = xTaskCreate( prvIdleTask,
                           configIDLE_TASK_NAME,
                           configMINIMAL_STACK_SIZE,
                           ( void * ) NULL,
                           portPRIVILEGE_BIT,  /* In effect ( tskIDLE_PRIORITY | portPRIVILEGE_BIT ), but tskIDLE_PRIORITY is zero. */
```

Creating the idle task and giving it the maximum periodicity to be always the last task in the EDF list.



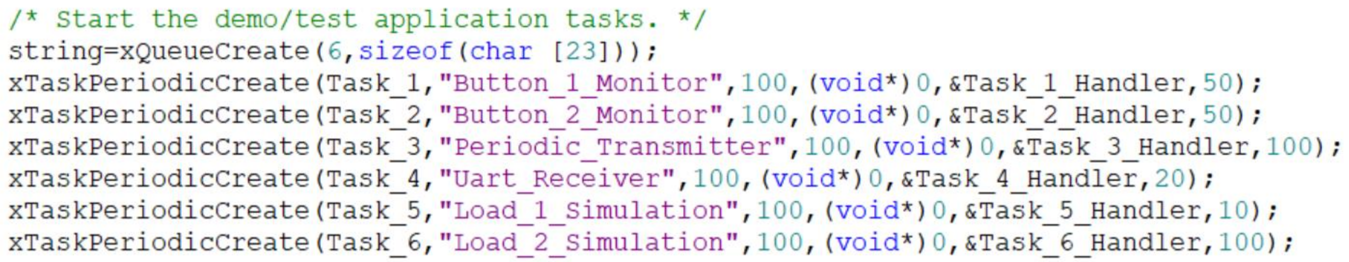
```
#if (configUSE_EDF_SCHEDULER==1)
{
  if(pxTCB->xStateListItem.xItemValue<=xTaskGetTickCount())
        {
          listSET_LIST_ITEM_VALUE( &( ( pxTCB )->xStateListItem ),pxTCB->xTaskPeriod+xTaskGetTickCount());
        }
        prvAddTaskToReadyList( pxTCB );

  if( pxTCB->xStateListItem.xItemValue <= pxCurrentTCB->xStateListItem.xItemValue )
  {

     xSwitchRequired = pdTRUE;
  }
  else
  {
    mtCOVERAGE_TEST_MARKER();
  }
}
#else
```

Adjust the list to fit the requirement and the algorism of EDF, so The task has the closest deadline, the task will be executed.

→Period time + current tick ←

The others change is respected to the thesis.

Tasks



```
/* Start the demo/test application tasks. */
string=xQueueCreate(6,sizeof(char [23]));
xTaskPeriodicCreate(Task_1,"Button_1_Monitor",100,(void*)0,&Task_1_Handler,50);
xTaskPeriodicCreate(Task_2,"Button_2_Monitor",100,(void*)0,&Task_2_Handler,50);
xTaskPeriodicCreate(Task_3,"Periodic_Transmitter",100,(void*)0,&Task_3_Handler,100);
xTaskPeriodicCreate(Task_4,"Uart_Receiver",100,(void*)0,&Task_4_Handler,20);
xTaskPeriodicCreate(Task_5,"Load_1_Simulation",100,(void*)0,&Task_5_Handler,10);
xTaskPeriodicCreate(Task_6,"Load_2_Simulation",100,(void*)0,&Task_6_Handler,100);
```

Task 1: Button_1_Monitor → {Periodicity: 50, Deadline: 50} monitor rising or falling edge

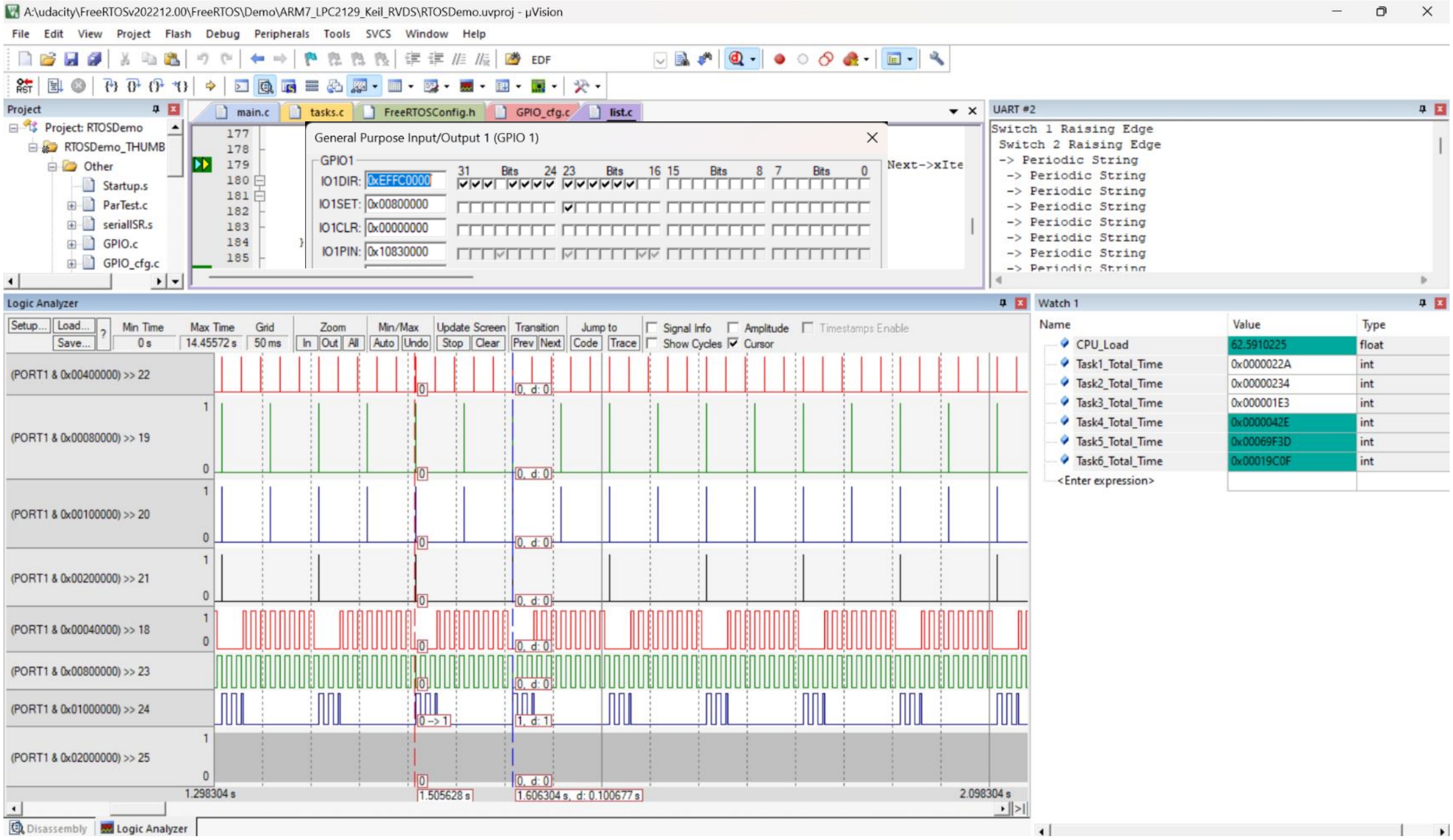
Task 2: Button_2_Monitor → {Periodicity: 50, Deadline: 50} monitor rising or falling edge

Task 3: Periodic_Transmitter → {Periodicity: 100, Deadline: 100} send preiodic string

Task 4: Uart_Receiver→ {Periodicity: 20, Deadline: 20} write on UART any received string from other tasks

Task 5: Load_1_Simulation→ {Periodicity: 10, Deadline: 10} Execution time: 5ms
Task 6: Load_2_Simulation→ {Periodicity: 100, Deadline: 100} Execution time: 12ms

Logic Analyzer


A:\udacity\FreeRTOSv202212.00\FreeRTOS\Demo\ARM7_LPC2129_Keil_RVDS\RTOSDemo.uvproj - µVision
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
EDF
Project
Project: RTOSDemo
RTOSDemo_THUMB
Other
Startup.s
ParTest.c
serialISR.s
GPIO.c
GPIO_cfg.c
main.c
tasks.c
FreeRTOSConfig.h
GPIO_cfg.c
list.c
Next->xIte
Logic Analyzer
Setup...
Load...
Save...
Zoom In Out All
Min/Max Auto Undo
Update Screen Stop Clear
Transition Prev Next
Jump to Code Trace
(PORT1 & 0x00400000) >> 22
(PORT1 & 0x00080000) >> 19
(PORT1 & 0x00100000) >> 20
(PORT1 & 0x00200000) >> 21
(PORT1 & 0x00040000) >> 18
(PORT1 & 0x00800000) >> 23
(PORT1 & 0x01000000) >> 24
(PORT1 & 0x02000000) >> 25
1.298304 s
1.505628 s
1.606304 s, d: 0.100677 s
2.098304 s
Disassembly
Logic Analyzer

General Purpose Input/Output 1 (GPIO 1)

GPIO1

IO1DIR:	0xEFFC0000
IO1SET:	0x00800000
IO1CLR:	0x00000000
IO1PIN:	0x10830000

Min Time: 0 s, Max Time: 14.45572 s, Grid: 50 ms

□ Signal Info □ Amplitude □ Timestamps Enable

□ Show Cycles ☑ Cursor

UART #2

```
Switch 1 Raising Edge
 Switch 2 Raising Edge
 -> Periodic String
  -> Periodic String
  -> Periodic String
  -> Periodic String
  -> Periodic String
  -> Periodic String
  -> Periodic String
  -> Periodic String
```

Watch 1

Name	Value	Type
CPU_Load	62.5910225	float
Task1_Total_Time	0x0000022A	int
Task2_Total_Time	0x00000234	int
Task3_Total_Time	0x000001E3	int
Task4_Total_Time	0x0000042E	int
Task5_Total_Time	0x00069F3D	int
Task6_Total_Time	0x00019C0F	int
<Enter expression>		

Keil Simulation 1

Note : PIN 22-> UART Receiver, PIN19-> Button 1,PIN20-> Button 2,PIN21-> Periodic Transmitter,PIN18->IDLE Task

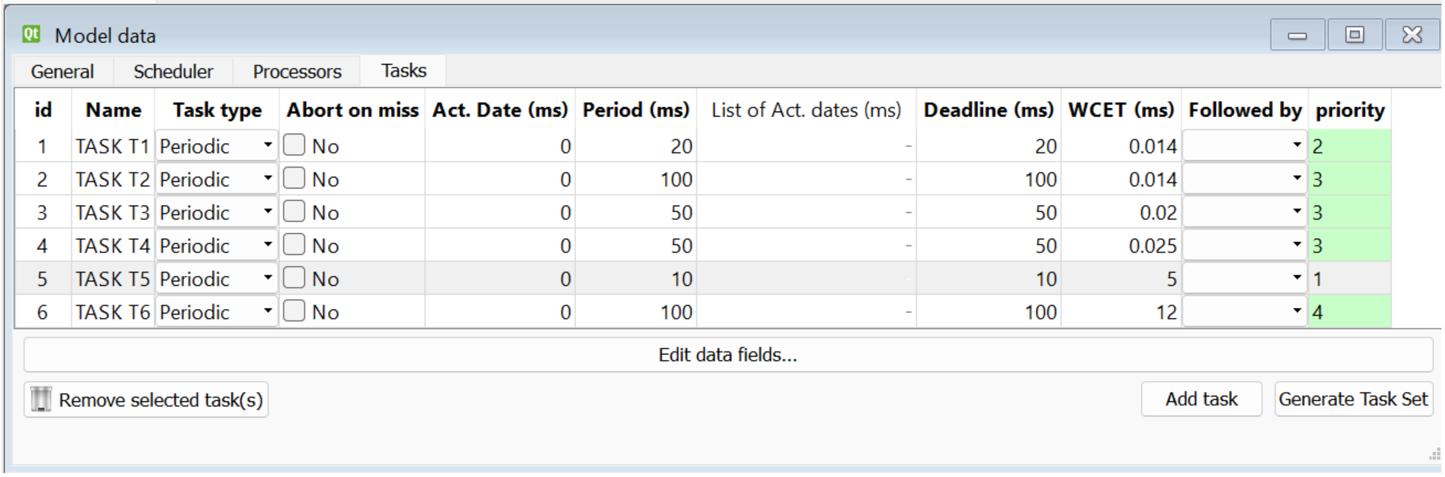
PIN23-> Load 1 Simulation,PIN24->Load 2 Simulation,PIN25->Tick Hook

Figured out from The Figure (Keil Simulation 1):

1. CPU Load =~ 62.5% and it will continue decreasing till 61% expected analytically.
2. Hyper period = 100 ms .
3. The behavior of the tasks according to EDF Scheduling.
4. UART receives a text periodically.

Simso simulator

1.



Model data

General | Scheduler | Processors | Tasks

id	Name	Task type	Abort on miss	Act. Date (ms)	Period (ms)	List of Act. dates (ms)	Deadline (ms)	WCET (ms)	Followed by	priority
1	TASK T1	Periodic	□ No	0	20	-	20	0.014		2
2	TASK T2	Periodic	□ No	0	100	-	100	0.014		3
3	TASK T3	Periodic	□ No	0	50	-	50	0.02		3
4	TASK T4	Periodic	□ No	0	50	-	50	0.025		3
5	TASK T5	Periodic	□ No	0	10		10	5		1
6	TASK T6	Periodic	□ No	0	100	-	100	12		4

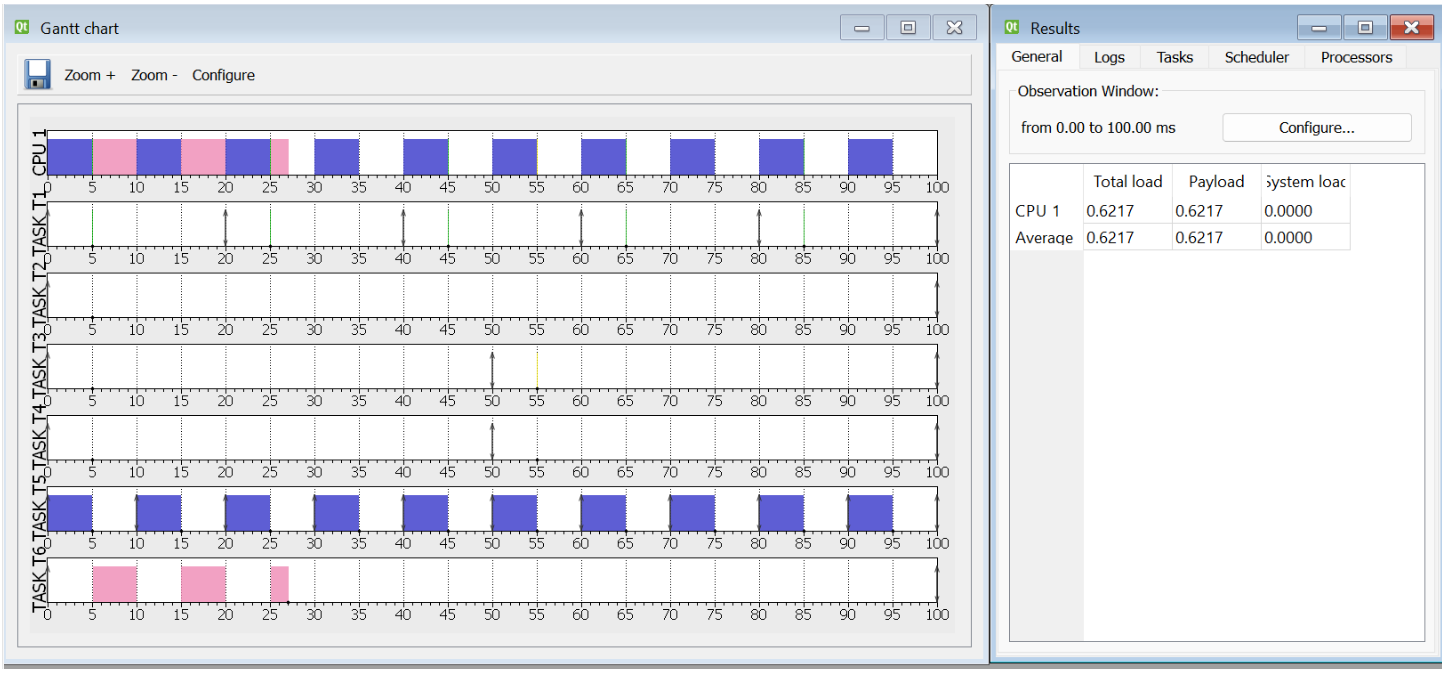
Edit data fields...

Remove selected task(s)

Add task

Generate Task Set

2. Offline Behavior :



Observation Window:

from 0.00 to 100.00 ms

	Total load	Payload	System loac
CPU 1	0.6217	0.6217	0.0000
Average	0.6217	0.6217	0.0000

Calculation

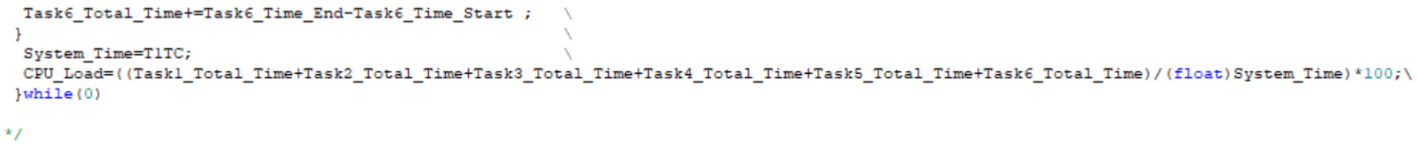
Hyper period :

The hyper period can be figured as the largest periodicity : in the project is 100 ms

Or can be figured from the logic analyzer as 100 ms

CPU Load:

It can be calculated from simso simulator as it shown it the figure or as shown in the project with the variable CPU_Load :



```
  Task6_Total_Time+=Task6_Time_End-Task6_Time_Start ;      \
 }                                                         \
  System_Time=T1TC;                                        \
  CPU_Load=((Task1_Total_Time+Task2_Total_Time+Task3_Total_Time+Task4_Total_Time+Task5_Total_Time+Task6_Total_Time)/(float)System_Time)*100;\
 }while(0)

*/
```

This variable is used to calculate the cpu load by taking the sum of the execution time of all tasks and dividing it by system time.

Or numerically as:

CPU Load% = (Total execution time of all running tasks / hyper period)*100 = 62,17%

Check system schedulability using URM:

URM = $\sum_{i=1}^{n} \frac{C_i}{P_i} = \frac{0.014}{20} + \frac{0.014}{100} + \frac{0.02}{50} + \frac{0.025}{50} + \frac{5}{10} + \frac{12}{100} = \ 0.62174$.

$n\left(2^{\frac{1}{n}} - 1\right) = 6\left(2^{\frac{1}{6}} - 1\right) = 0.735$.

Comment :

➔ **URM <** $n\left(2^{\frac{1}{n}} - 1\right)$: feasible in RM scheduler

---

Time demand analysis techniques :

- Load 1 Simulation Task (P:10,E:5,D:10)
- W(20)=5+0=5<D
- →Schedulable task.

- Uart Receiver Task (P:20,E:0.016,D:20)
- W(20)=0.016+(20/10)*5=10.016<D
  - Schedulable task.

- Button 1 Monitor Task (P:50,E:0.02,D:50)
- W(50)=0.02+(50/10)*5+(50/20)*0.016 =25.068<D

→Schedulable task.

- Button 2 Monitor Task (P:50,E:0.02,D:50)
- W(50)=0.02+(50/10)*5+(50/20)*0.016+(50/50)*0.02=25.07<D

  →Schedulable task.

- Periodic Transmitter Task (P:100,E:0.019,D:100)
  W(100) = 0.019+(100/10)*5+(100/20)*0.016+(100/50)*0.02+(100/50)*0.02=50.179<D
  →Schedulable task.

- Load 2 Simulation Task (P:100,E:12,D:100)
  W(100) = 12+(100/10)*5+(100/20)*0.016+(100/50)*0.02+(100/50)*0.02+(100/100)*0.019=62.179<D →Schedulable task .

Comment :

This application’s feasible in fixed priority schedulers.