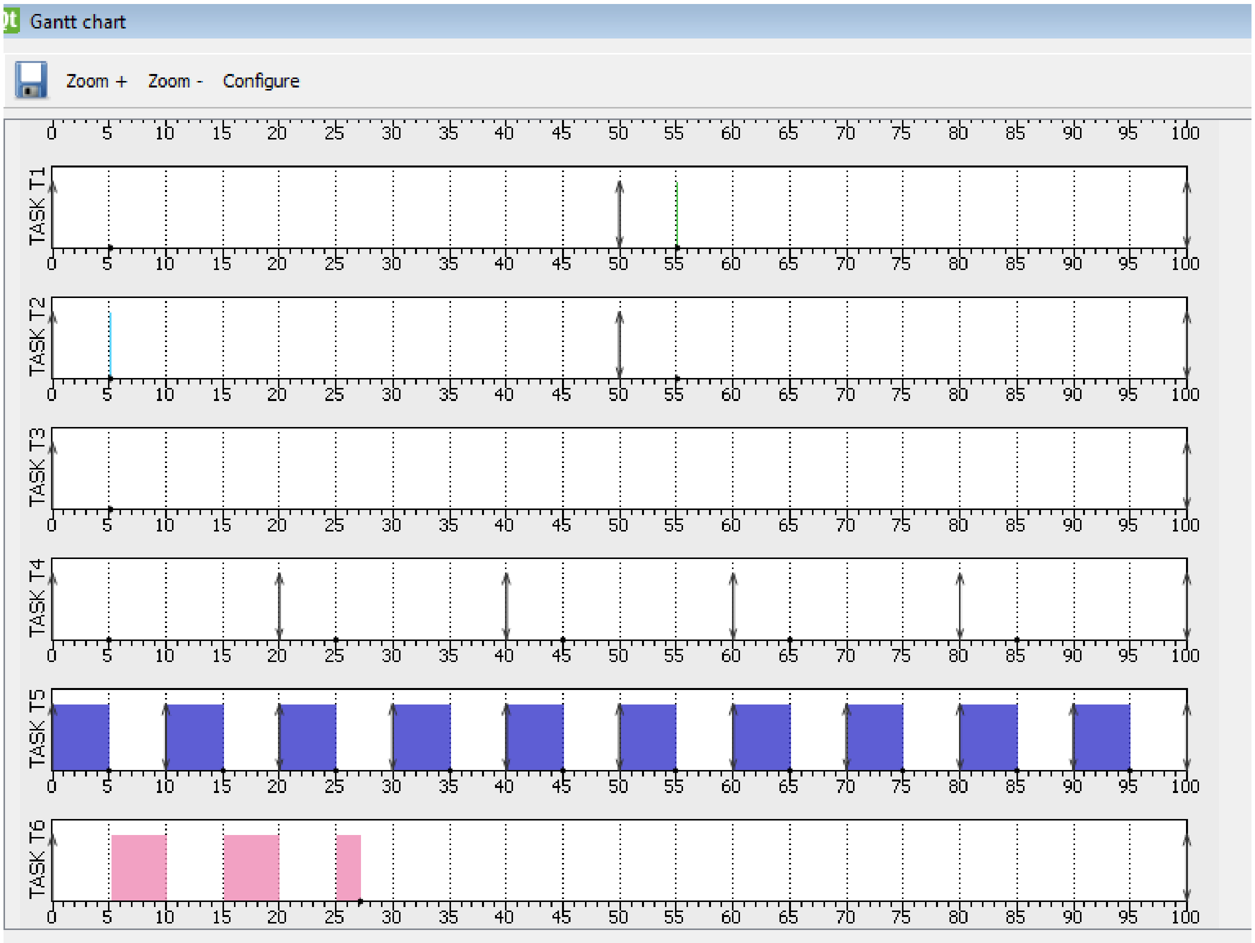
# Implementing EDF scheduler

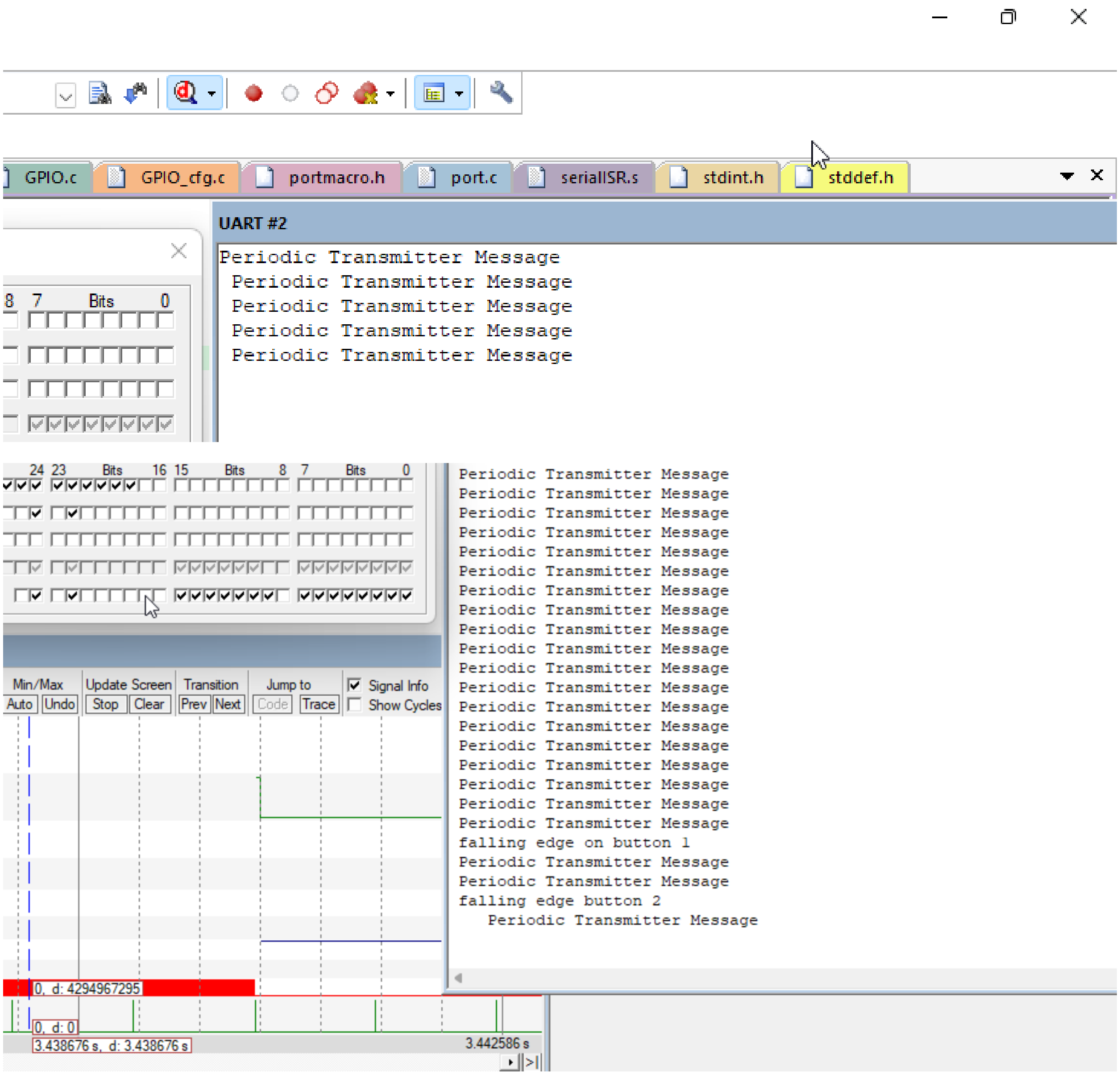
After Implementing all changes in the thesis, and implementing the modifications that are not in the thesis

I followed steps to verify that the algorithm is working correctly.
1- I applied the tick hook function and the idle hook function and assigned GPIO pins 9 and 8 respectively.
2- I implemented all tasks and verified they are working and no tasks are missing their deadlines.

I applied the same set of tasks on SIMSO



Verifying that UART is working correctly:

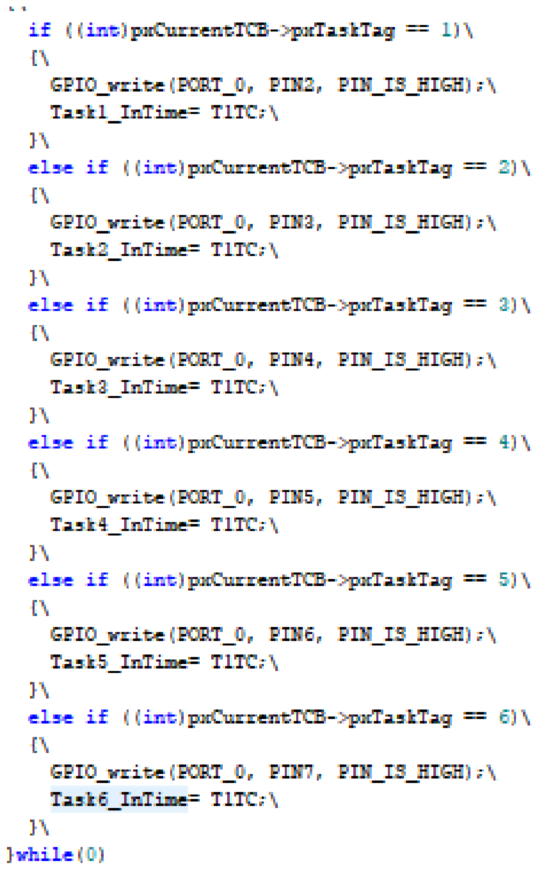


UART #2

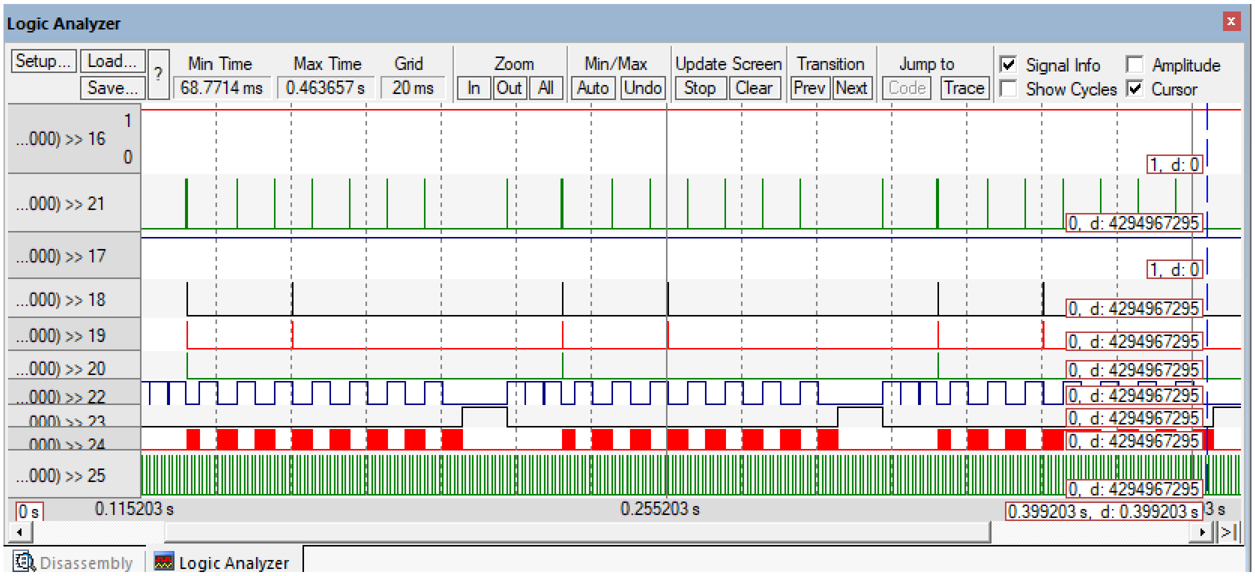
```
Periodic Transmitter Message
 Periodic Transmitter Message
 Periodic Transmitter Message
 Periodic Transmitter Message
 Periodic Transmitter Message
```

```
Periodic Transmitter Message
Periodic Transmitter Message
Periodic Transmitter Message
Periodic Transmitter Message
Periodic Transmitter Message
Periodic Transmitter Message
Periodic Transmitter Message
Periodic Transmitter Message
Periodic Transmitter Message
Periodic Transmitter Message
Periodic Transmitter Message
Periodic Transmitter Message
Periodic Transmitter Message
Periodic Transmitter Message
Periodic Transmitter Message
Periodic Transmitter Message
Periodic Transmitter Message
Periodic Transmitter Message
Periodic Transmitter Message
falling edge on button 1
Periodic Transmitter Message
Periodic Transmitter Message
falling edge button 2
   Periodic Transmitter Message
```

3- I applied trace macros and assigned GPIO pins to all tasks.



```
  if ((int)pxCurrentTCB->pxTaskTag == 1)\
  {\
    GPIO_write(PORT_0, PIN2, PIN_IS_HIGH);\
    Task1_InTime= T1TC;\
  }\
  else if ((int)pxCurrentTCB->pxTaskTag == 2)\
  {\
    GPIO_write(PORT_0, PIN3, PIN_IS_HIGH);\
    Task2_InTime= T1TC;\
  }\
  else if ((int)pxCurrentTCB->pxTaskTag == 3)\
  {\
    GPIO_write(PORT_0, PIN4, PIN_IS_HIGH);\
    Task3_InTime= T1TC;\
  }\
  else if ((int)pxCurrentTCB->pxTaskTag == 4)\
  {\
    GPIO_write(PORT_0, PIN5, PIN_IS_HIGH);\
    Task4_InTime= T1TC;\
  }\
  else if ((int)pxCurrentTCB->pxTaskTag == 5)\
  {\
    GPIO_write(PORT_0, PIN6, PIN_IS_HIGH);\
    Task5_InTime= T1TC;\
  }\
  else if ((int)pxCurrentTCB->pxTaskTag == 6)\
  {\
    GPIO_write(PORT_0, PIN7, PIN_IS_HIGH);\
    Task6_InTime= T1TC;\
  }\
}while(0)
```



Logic Analyzer

Min Time 68.7714 ms | Max Time 0.463657 s | Grid 20 ms

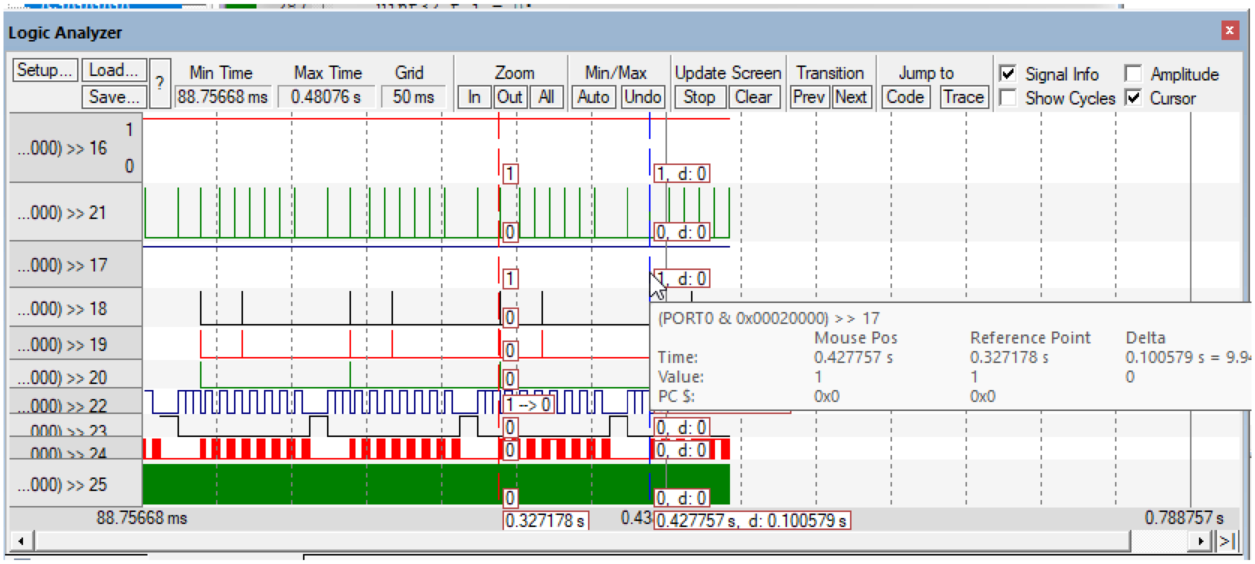
☑ Signal Info □ Amplitude □ Show Cycles ☑ Cursor

Now it’s time to verify the system analytically.

Calculating the hyperperiod:

The hyperperiod is detected when a pattern starts to happen and all tasks start together.

So I provided screenshots from Keil simulation and SIMSO to make sure I got the hyperperiod right



Logic Analyzer

Min Time: 88.75668 ms

Max Time: 0.48076 s

Grid: 50 ms

☑ Signal Info

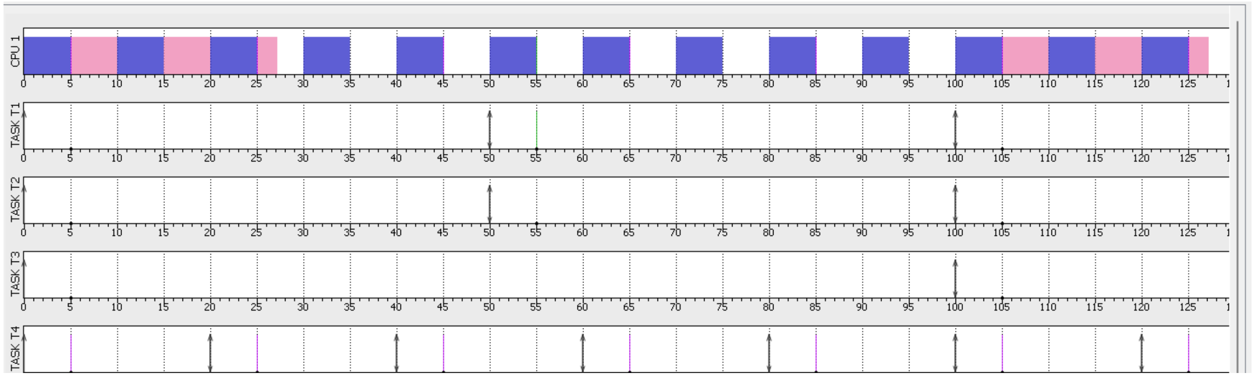
□ Show Cycles

□ Amplitude

☑ Cursor

(PORT0 & 0x00020000) >> 17

|  | Mouse Pos | Reference Point | Delta |
|---|---|---|---|
| Time: | 0.427757 s | 0.327178 s | 0.100579 s = 9.9 |
| Value: | 1 | 1 | 0 |
| PC $: | 0x0 | 0x0 |  |

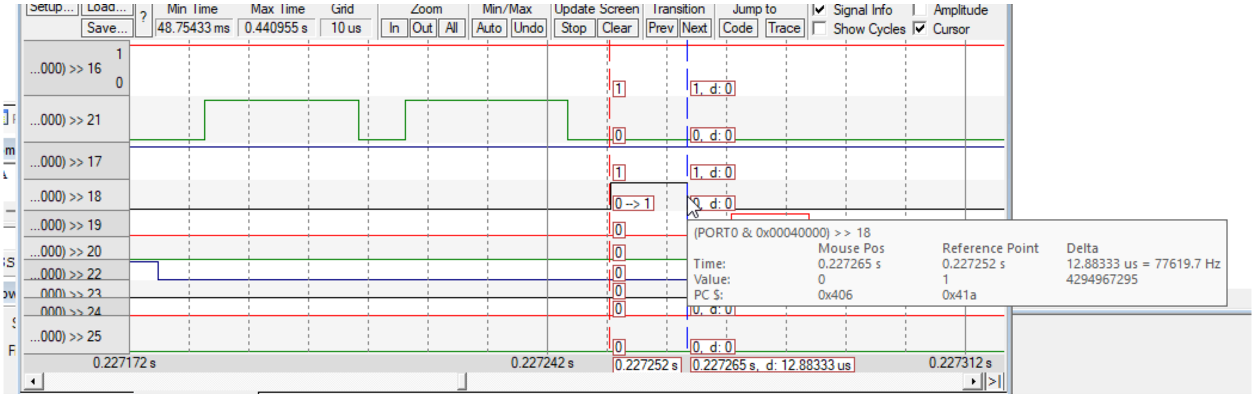


The hyperperiod is detected at 100.

Calculating the CPU load:

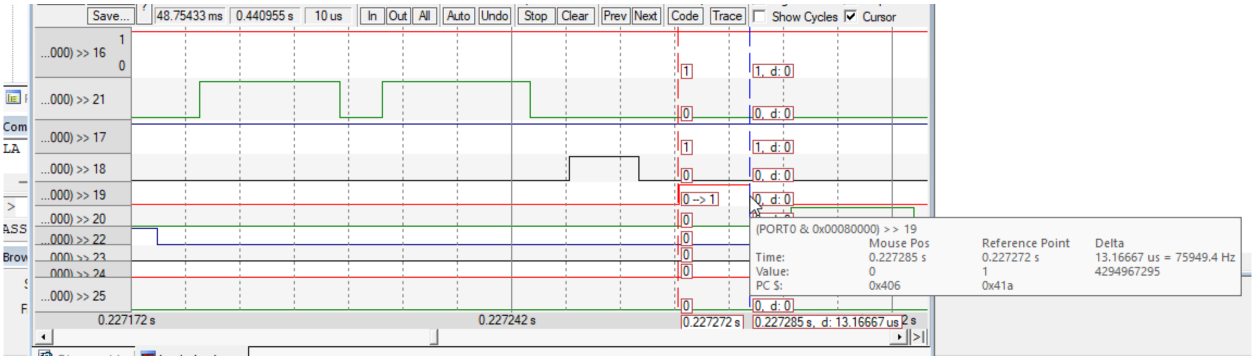
I first need to know all tasks' execution time, so using GPIOs and logic analyzer I succeeded in doing so.

Task 1:



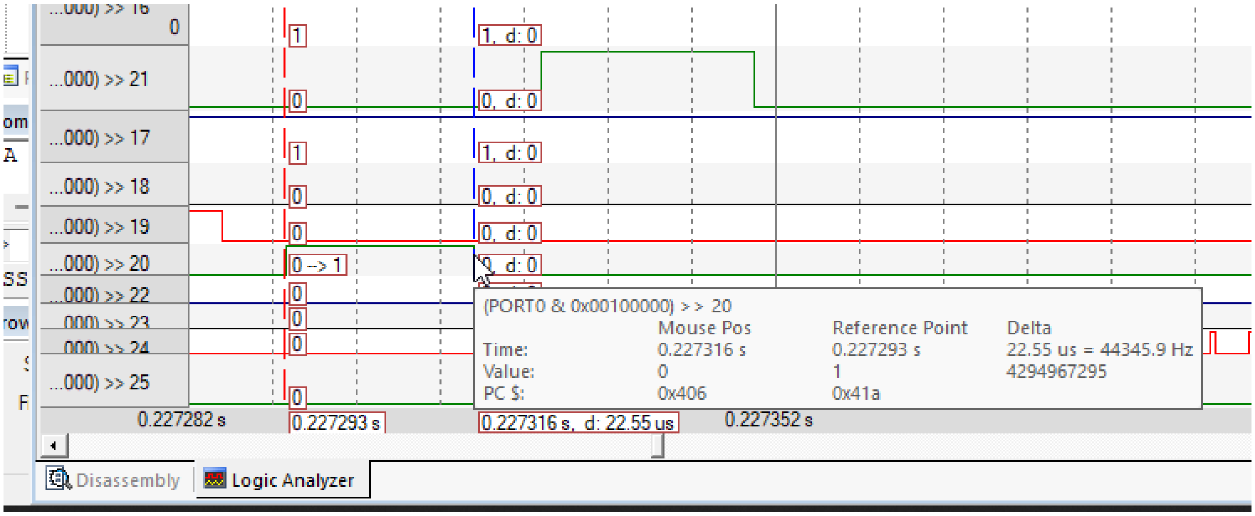
```
(PORT0 & 0x00040000) >> 18
           Mouse Pos     Reference Point    Delta
Time:      0.227265 s    0.227252 s         12.88333 us = 77619.7 Hz
Value:     0             1                  4294967295
PC $:      0x406         0x41a
```

Task 2:



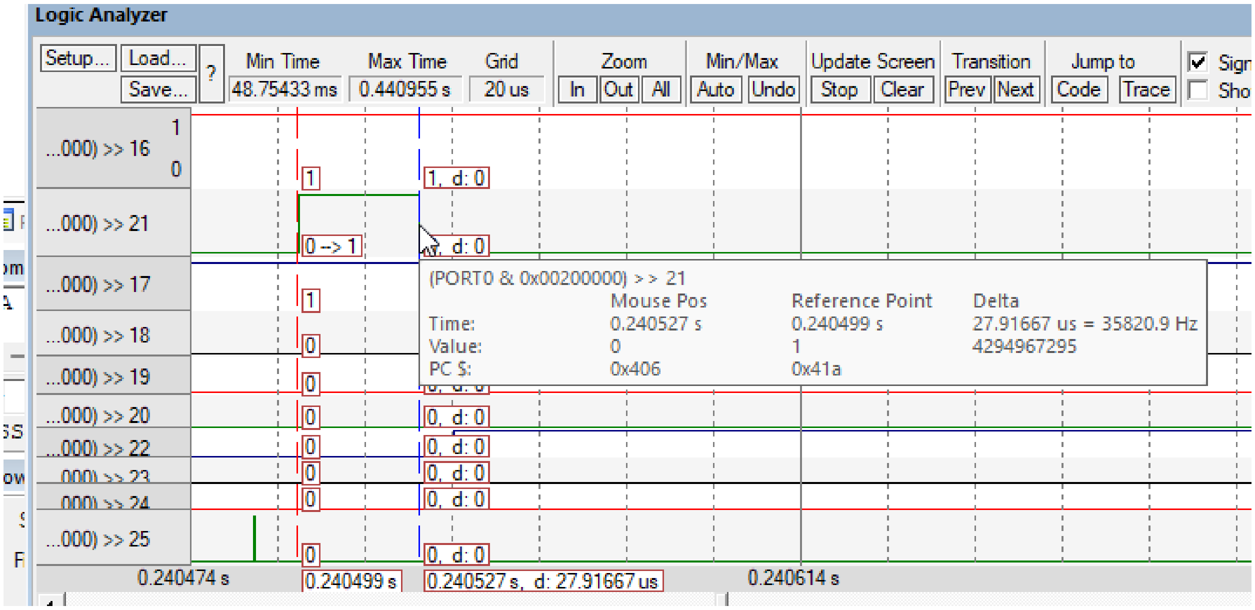
```
(PORT0 & 0x00080000) >> 19
           Mouse Pos     Reference Point    Delta
Time:      0.227285 s    0.227272 s         13.16667 us = 75949.4 Hz
Value:     0             1                  4294967295
PC $:      0x406         0x41a
```

Task 3:



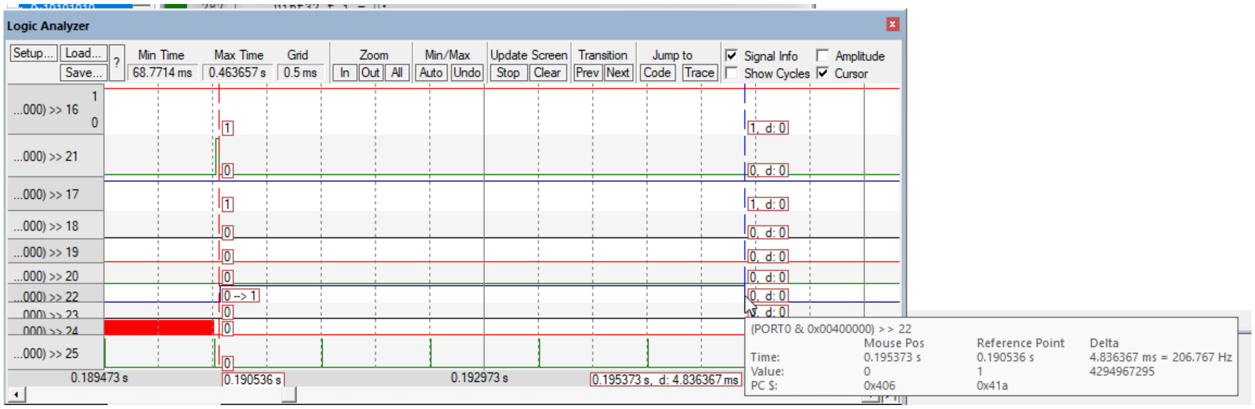
```
(PORT0 & 0x00100000) >> 20
           Mouse Pos     Reference Point    Delta
Time:      0.227316 s    0.227293 s         22.55 us = 44345.9 Hz
Value:     0             1                  4294967295
PC $:      0x406         0x41a
```

## Task 4:



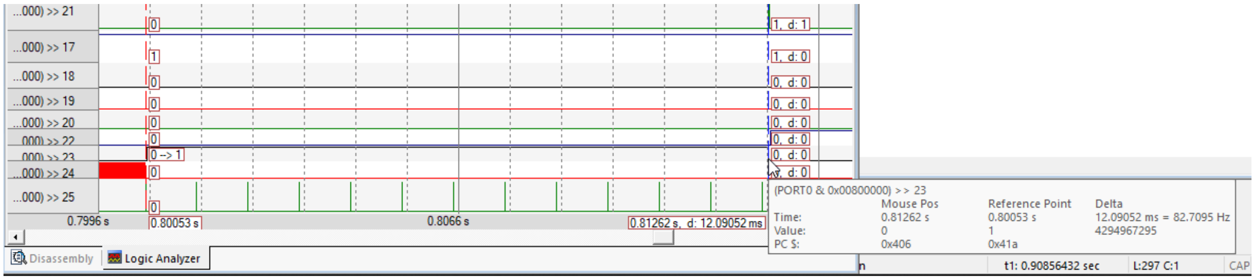
```
(PORT0 & 0x00200000) > > 21
          Mouse Pos      Reference Point     Delta
Time:     0.240527 s     0.240499 s          27.91667 us = 35820.9 Hz
Value:    0              1                   4294967295
PC $:     0x406          0x41a
```

## Task 5:



```
(PORT0 & 0x00400000) >> 22
          Mouse Pos      Reference Point     Delta
Time:     0.195373 s     0.190536 s          4.836367 ms = 206.767 Hz
Value:    0              1                   4294967295
PC $:     0x406          0x41a
```

## Task 6:

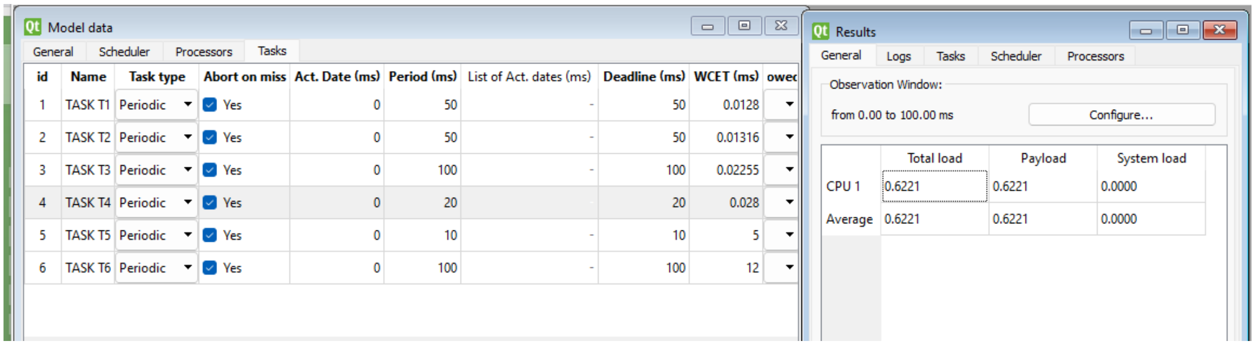


```
(PORT0 & 0x00800000) >> 23
          Mouse Pos      Reference Point     Delta
Time:     0.81262 s      0.80053 s           12.09052 ms = 82.7095 Hz
Value:    0              1                   4294967295
PC $:     0x406          0x41a
```

CPU load = task execution time / periodicity

= 12.8us/50ms + 13us/50ms + 22.5us/100ms + 28us/20ms + 5ms/10ms + 12ms/100ms = 62.21447% or 0.6221447

CPU load on SIMSO:



| id | Name | Task type | Abort on miss | Act. Date (ms) | Period (ms) | List of Act. dates (ms) | Deadline (ms) | WCET (ms) | owec |
|---|---|---|---|---|---|---|---|---|---|
| 1 | TASK T1 | Periodic | ☑ Yes | 0 | 50 | - | 50 | 0.0128 | |
| 2 | TASK T2 | Periodic | ☑ Yes | 0 | 50 | - | 50 | 0.01316 | |
| 3 | TASK T3 | Periodic | ☑ Yes | 0 | 100 | - | 100 | 0.02255 | |
| 4 | TASK T4 | Periodic | ☑ Yes | 0 | 20 | - | 20 | 0.028 | |
| 5 | TASK T5 | Periodic | ☑ Yes | 0 | 10 | - | 10 | 5 | |
| 6 | TASK T6 | Periodic | ☑ Yes | 0 | 100 | - | 100 | 12 | |

Observation Window:

from 0.00 to 100.00 ms

| | Total load | Payload | System load |
|---|---|---|---|
| CPU 1 | 0.6221 | 0.6221 | 0.0000 |
| Average | 0.6221 | 0.6221 | 0.0000 |

So the two values are so close.

System schedulability using URM:

U (CPU load) should be less than or equal to n*($2^{1/n}$ – 1) where n is the number of tasks.

n = 6

6*($2^{1/6}$ – 1) = 0.73477

U is calculated before and it equals 0.6221447

So U is less than n*($2^{1/n}$ – 1), so the system is schedulable.

System schedulability using time demand analysis:

Periodicity against priority:

| | |
|---|---|
| Task 1: (P:50, E: 0.0128) | 3 |
| Task 2: (P:50, E: 0.0131) | 4 |
| Task 3: (P:100, E: 0.02255) | 2 |
| Task 4: (P: 20, E: 0.028) | 5 |
| Task 5: (P:10, E:5) | 6 |
| Task 6: (P:100, E:12) | 1 |

Testing task 5:
$w(10) = 5 < 10$ so task 5 is schedulable

Testing task 4:
$w(20) = 0.028 + (20/10)*5 = 10.028 < 20$

so task 4 is schedulable

Testing task 2:
$w(50) = 0.0131 + (50/20)*0.028 + (50/10)*5 = 25.0831 < 50$

so task 2 is schedulable

Testing task 1:
$w(50) = 0.0128 + (50/50)*0.0131 + (50/20)*0.028 + (50/10)*5 = 25.0959 < 50$

so task 1 is schedulable

Testing task 3:

$w(100) = 0.02255 + (100/50)*0.0128 + (100/50)*0.0131 + (100/20)*0.028 + (100/10)*5 = 50.21435 < 100$

so task 3 is schedulable

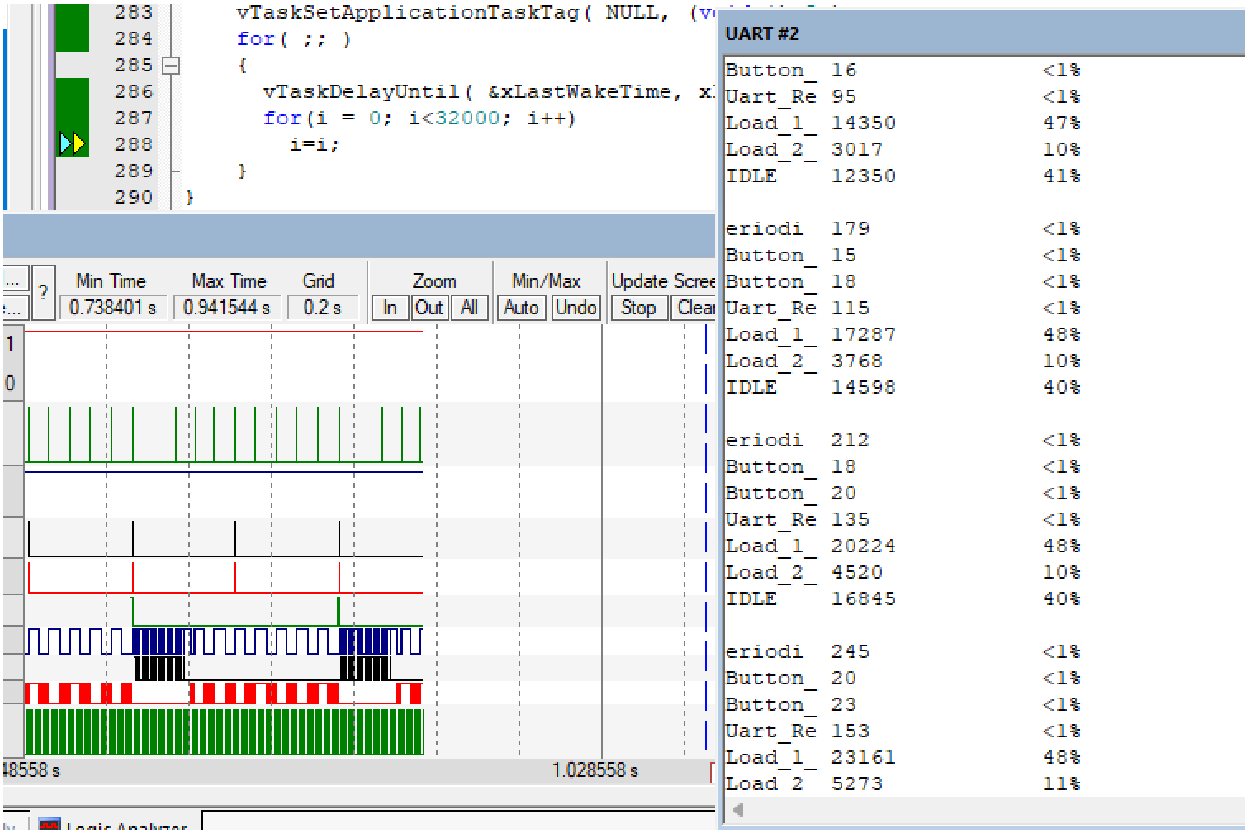
Testing task 6:

$w(100) = 12 + (100/100)*0.02255 + (100/50)*0.0128 + (100/50)*0.0131 + (100/20)*0.028 + (100/10)*5 = 62.21435 < 100$

so task 6 is schedulable

since all tasks are schedulable so the system is totally schedulable.

Calculating CPU load on run-time:



```
    vTaskSetApplicationTaskTag( NULL, (v
    for( ;; )
    {
      vTaskDelayUntil( &xLastWakeTime, x
      for(i = 0; i<32000; i++)
        i=i;
    }
}
```

Min Time 0.738401 s   Max Time 0.941544 s   Grid 0.2 s

UART #2

```
Button_  16           <1%
Uart_Re  95           <1%
Load_1_  14350        47%
Load_2_  3017         10%
IDLE     12350        41%

eriodi   179          <1%
Button_  15           <1%
Button_  18           <1%
Uart_Re  115          <1%
Load_1_  17287        48%
Load_2_  3768         10%
IDLE     14598        40%

eriodi   212          <1%
Button_  18           <1%
Button_  20           <1%
Uart_Re  135          <1%
Load_1_  20224        48%
Load_2_  4520         10%
IDLE     16845        40%

eriodi   245          <1%
Button_  20           <1%
Button_  23           <1%
Uart_Re  153          <1%
Load_1_  23161        48%
Load 2   5273         11%
```