Lab Session - 7

OS Clock Tick : Resolution & Accuracy

Name : Durvesh Naresh Patil
PRN : 2019BTEEN00035
Batch : EN-1 Sub. : RTOS

6.3 Purpose of the experiment.

1) Illustrate the purpose of the experiment.

⇒ (i) Difference betn theoretical & actual delay is ~~uder~~ understood through this experiment.

(ii) How tasks are affected when the cpu is more engaged.

2) If we do not understand this experiment what are the problems that will occur while handling RTOS based project in industry?

⇒ (i) Difference betn desired & actual delay

(ii) How efficiency of cpu degrades when it is too busy with many tasks.

(iii) For accuracy, we need to study this experiment

6.4 Theory

1) What is time tick? What is its normal value?

⇒ It is fixed interval periodic interrupt. Resolution of time is dependent on it. Simply, time unit on which delays are based on.

Normal value is betn 10 to 100 ms

2) How to change value of time tick ? why to change it ?

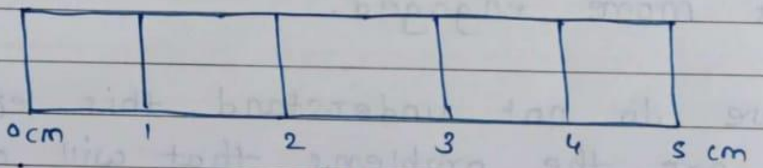
⇒ (i) To change value of time tick
config.h → include.h → os-cfg.h → OS_TICKS_PER_SEC

(ii) We change the value of time tick to get desired delay value

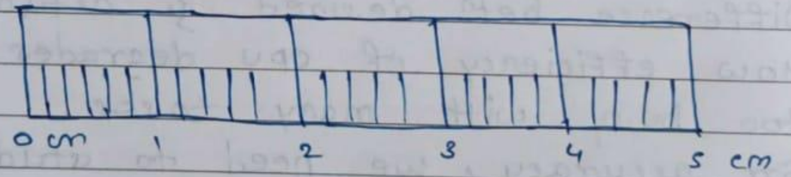
3) What is the resolution & what is accuracy ?

⇒ • <u>Resolution</u> : smallest unit we can measure

• <u>Accuracy</u> : closeness to correct / actual value



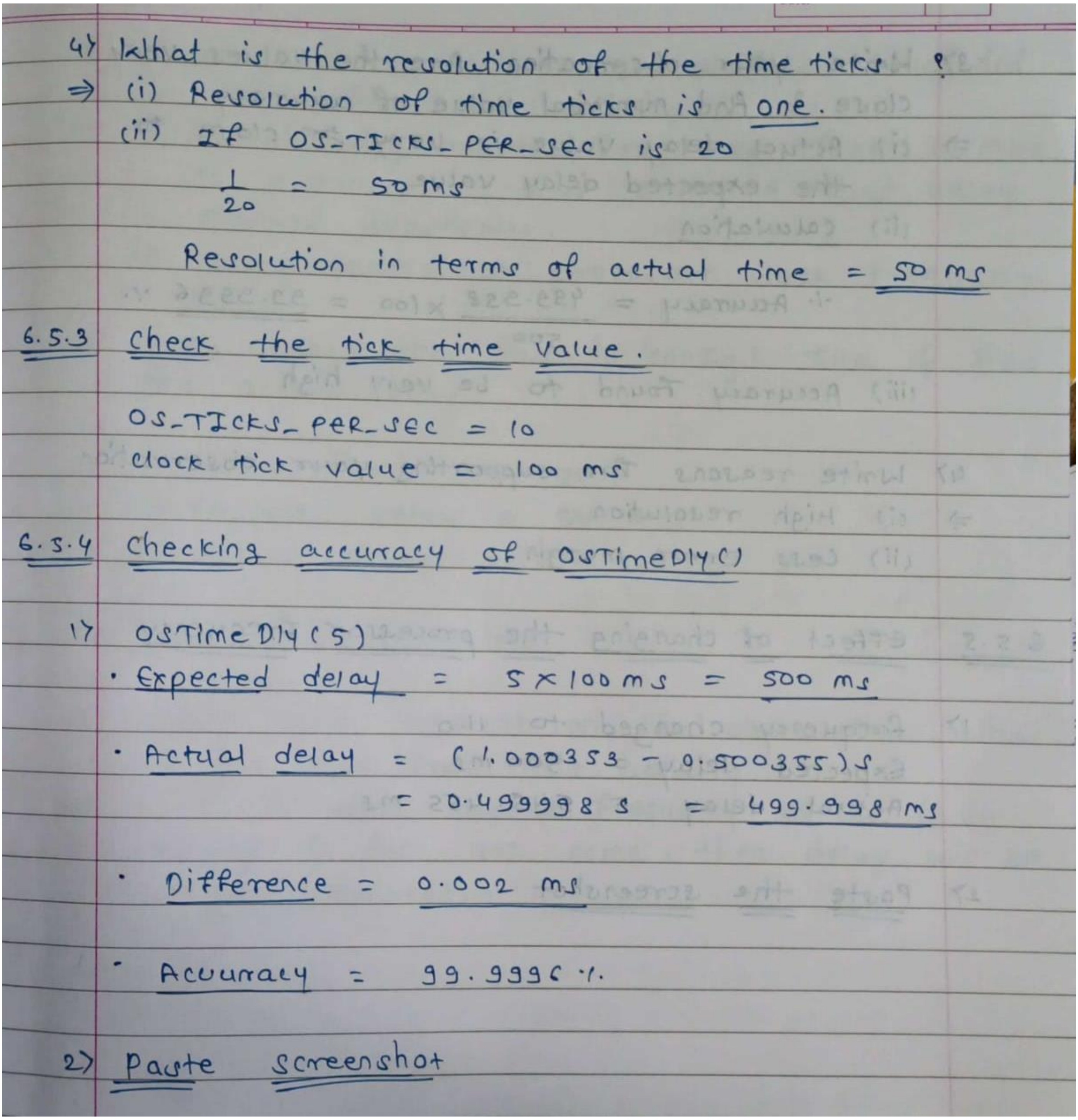
(A)



(B)

In scale A we measure distance upto cm only but in scale B we can measure mm accurately.

Scale B has more resolution than scale A



4) What is the resolution of the time ticks ?

⇒ (i) Resolution of time ticks is <u>one</u>.

(ii) If OS_TICKS_PER_SEC is 20

$$\frac{1}{20} = 50 \text{ ms}$$

Resolution in terms of actual time = 50 ms

6.5.3 Check the tick time value.

OS_TICKS_PER_SEC = 10

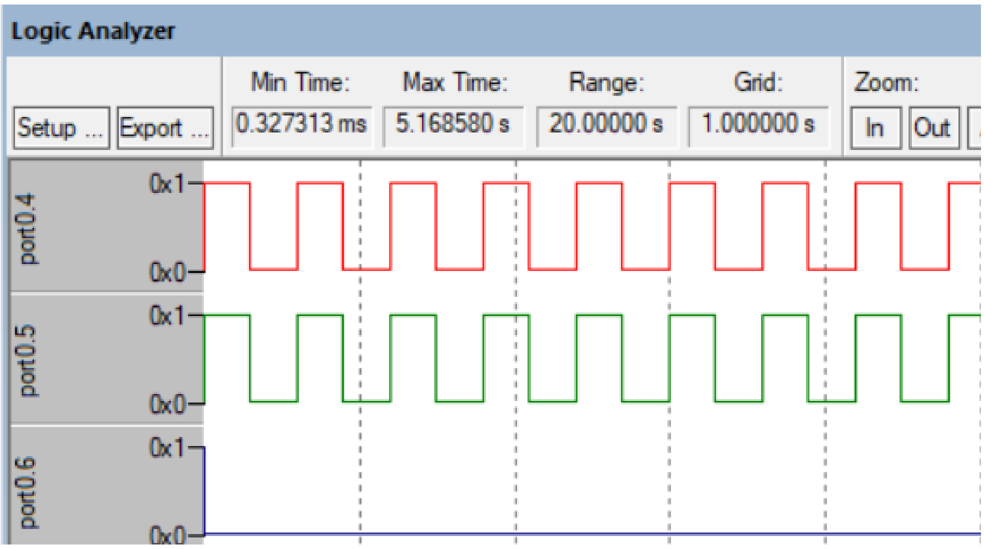
clock tick value = 100 ms

6.5.4 Checking accuracy of OSTimeDly()

1) OSTimeDly(5)

- Expected delay = 5 × 100 ms = 500 ms
- Actual delay = (1.000353 − 0.500355) s
  = 0.499998 s = 499.998 ms
- Difference = 0.002 ms
- Accuracy = 99.9996 %.

2) Paste screenshot



Logic Analyzer

Min Time: 0.327313 ms

Max Time: 5.168580 s

Range: 20.00000 s

Grid: 1.000000 s

3) Write your observation. Are the values very close & find numerical value of accuracy.

⇒ (i) Actual delay value is very ~~to~~ close to the expected delay value.

(ii) Calculation

$$\% \text{ Accuracy} = \frac{499.998}{500} \times 100 = \underline{99.9996} \%$$

(iii) Accuracy found to be very high

4) Write reasons for supporting your observation

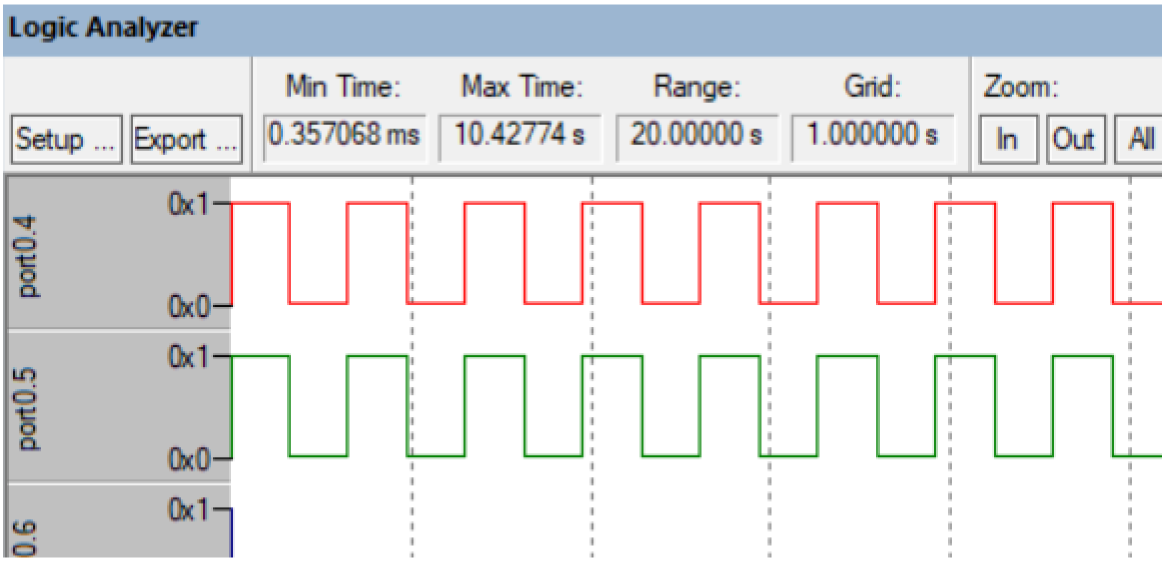
⇒ (i) High resolution

(ii) Less error margin

6.5.5 Effect of changing the processor frequency

1) frequency changed to 11.0
Expected delay = 500 ms
Actual delay = 545.455 ms

2) Paste the screenshot



Logic Analyzer

Min Time:	Max Time:	Range:	Grid:
0.357068 ms	10.42774 s	20.00000 s	1.000000 s

4> Write your comment on observations. What did you learn from this activity.

⇒ i> By changing crystal frequency from 12 MHz to 11 MHz, it was observed that delay changed drastically.

ii> Delay depend on processor clock frequency

5> Now, change the fosc in config.h file & find the actual delay.

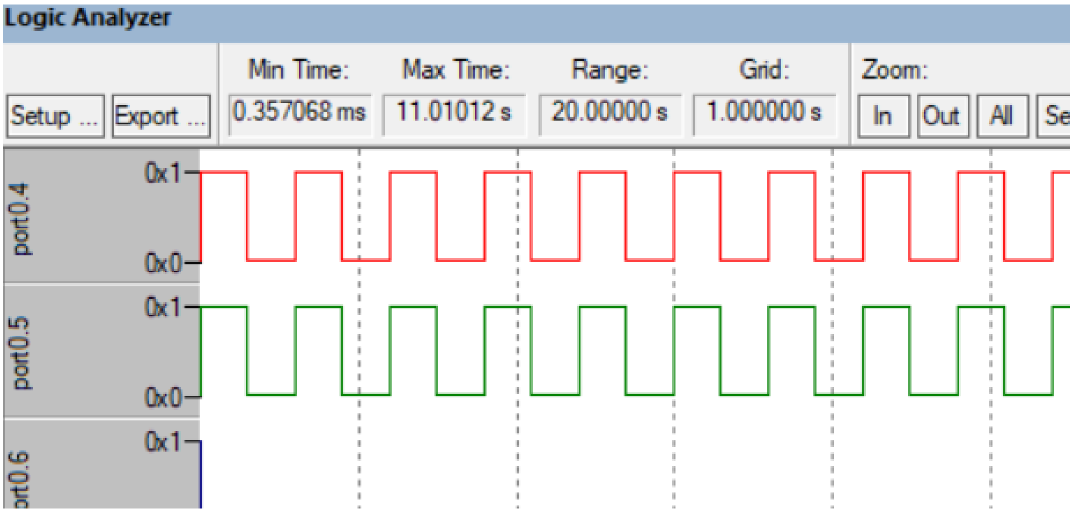
⇒

define fosc = 11000000

i> Expected delay = 500 ms

ii> Actual delay = 500.004 ms

6> Paste the screenshot.



Logic Analyzer

Min Time: 0.357068 ms

Max Time: 11.01012 s

Range: 20.00000 s

Grid: 1.000000 s

7> Write your comment on the observations. What did you learn from this activity?

⇒ When crystal frequency of μc in μvision & fosc are same, then delay will be as per expectation

6.5.6 Effect of changing ticks per sec setting

[Xtal (MHz) is 12.0

fosc = 12000000

OS_TICKS_PER_SEC 100]

OSTimeDly (5);

2> Expected delay = 50ms
Actual delay = 50.001 ms

3> Paste the screenshot

4> Write your opinion about the resolution of clock tick & resolution of clock period.

⇒ (i) Resolution of clock tick is one meaning it is integer value & we can't use fraction.

(ii) Lesser the resolution, greater is the accuracy

6.5.7 5> Effect of changing processor utilization on delay

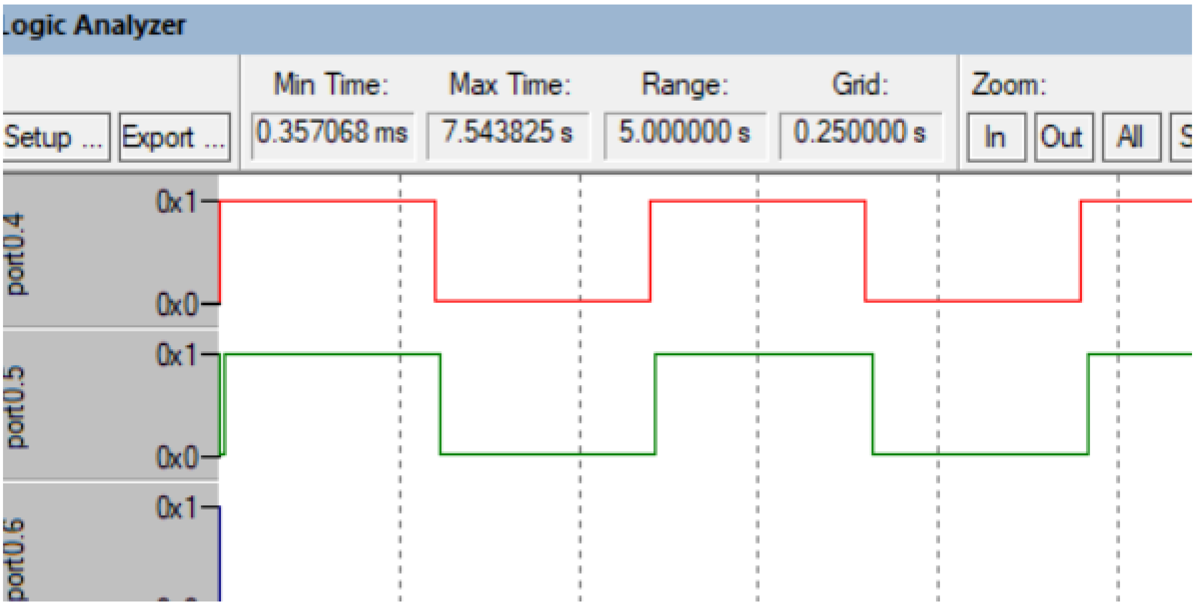
1. ① For Highest priority task
Expected delay = 40 ms
Actual delay = 40 ms

② For lower priority task
Expected delay = 40 ms
Actual delay = 100 ms

2. Paste the screenshots.

3. Comments :

Low priority task delay is extended i.e. it is not same as expected



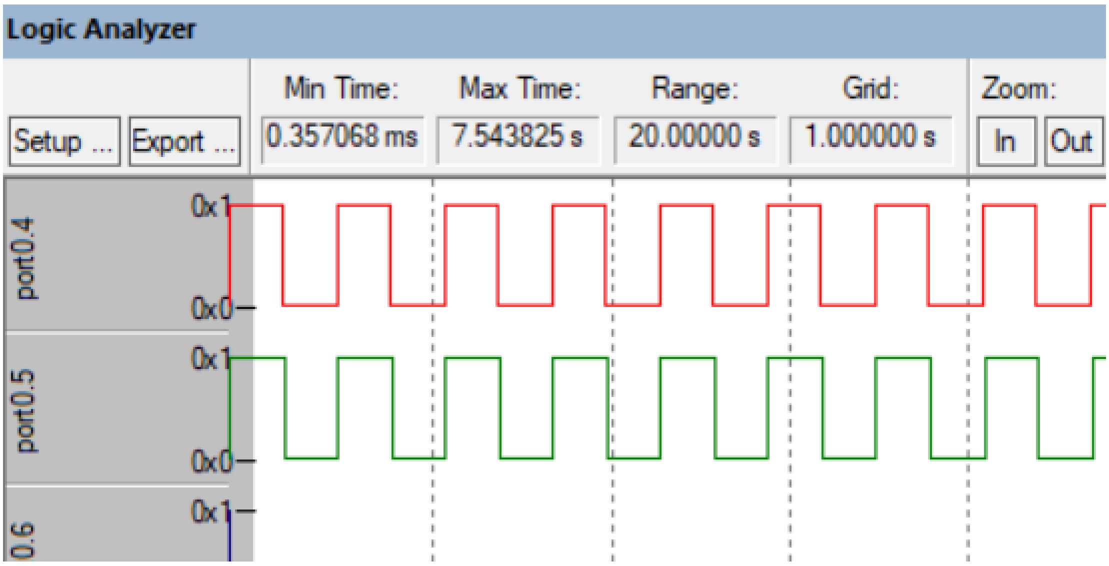
Logic Analyzer

Min Time: 0.357068 ms

Max Time: 7.543825 s

Range: 5.000000 s

Grid: 0.250000 s



Logic Analyzer

Min Time: 0.357068 ms

Max Time: 7.543825 s

Range: 20.00000 s

Grid: 1.000000 s

6.5.8 OSTimeDly() Function logic

1> OSTimeDly() takes number of ticks as an argument.

2> A task calls this function to suspend execution until some time expires. The calling Function will not execute until specified time expires

3> 3 modes are allowed by this fn
   (i) Relative
   (ii) Periodic
   (iii) Absolute

4> Pseudo code
   i> if ticks > 0, enters critical section
   ii> Delay current task by specific time ticks
   iii> Exit critical section

5> Max. limit for no. of ticks is 65535 because argument is of 16 bit.

6> If we increase to 32 bit or more limit thus increase

6.5.9 OSTimeDlyHMSM()

```
void TaskO (void *pdata)
{
  pdata = pdata ;
  while (1)
  {
     LED-ON(0) ;
     OSTime Dly HMSM (0,0,0,500);
     LED-off (0);
     OSTime Dly HMSM (0,0,0,500);
  }
}
```

Observations : syntax is as follows

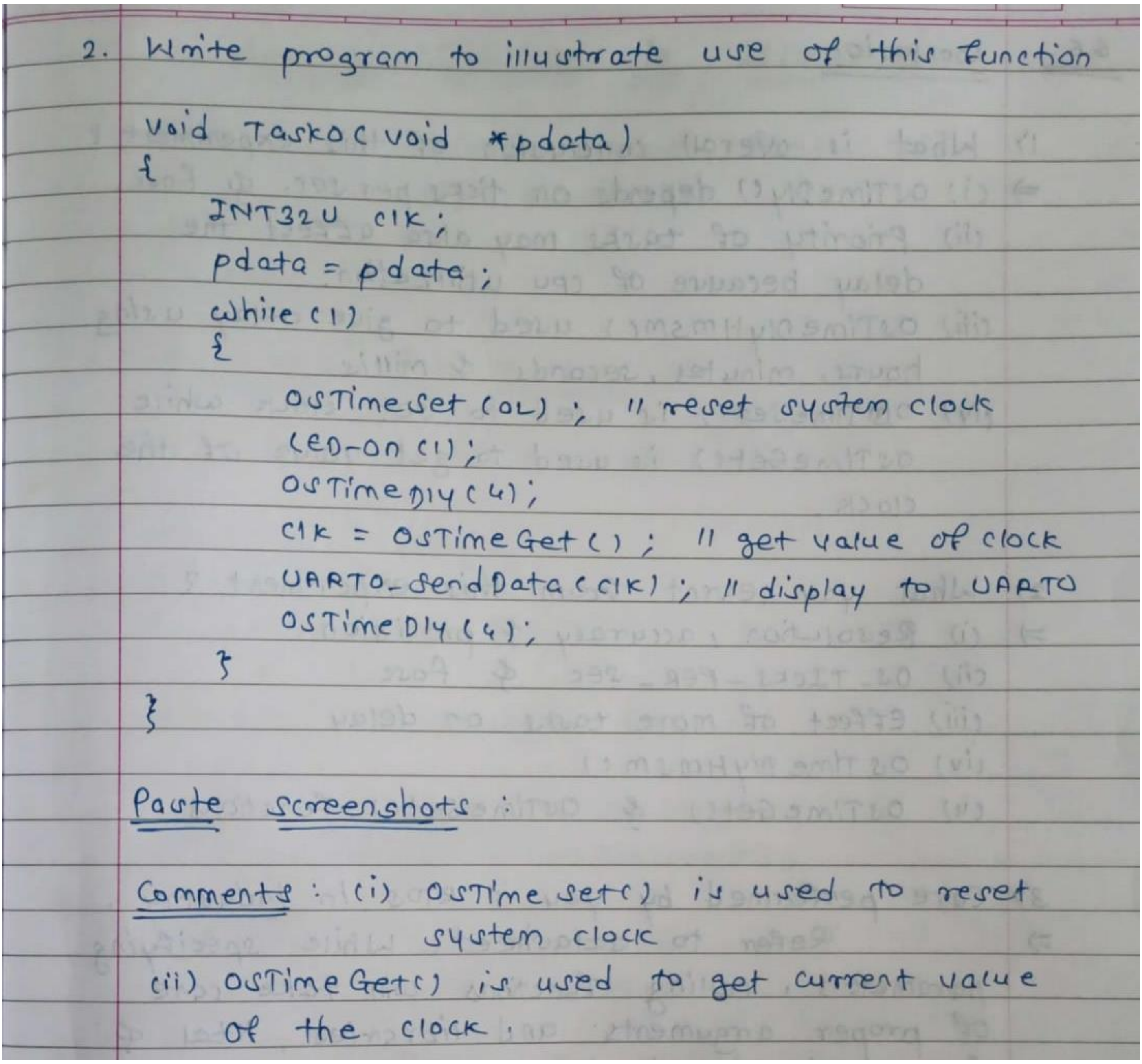
Void OSTimeDly(INT8 hours, minutes, seconds, milli) ;
Allows a task to delay itself for user specified amount of time in hours, minutes, seconds, milli.
~~At least~~

Comments : We provided 500 ms of delay to task0 via OSTimeDlyHMSM.
Through waveforms we can check the delay

6.3.10 OSTimeGet() & OSTimeSet()

1. (i) OSTime Get()
- Allows task to obtain current value of system clock.
- System clock is 32 bit counter that counts no. of clocks ticks as system clock last set
- Returns current system clock value.

(ii) OsTimeset()
- Allows task to set system clock
- Argument passed with desired value sets system clock in ticks



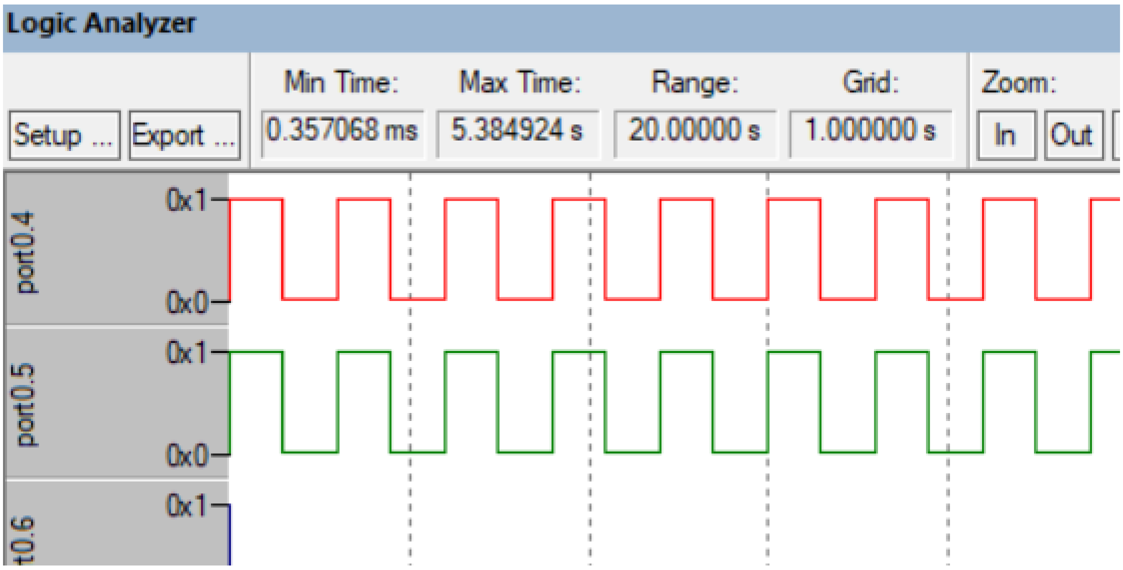
2. Wmite program to illustrate use of this function

```
void Task0( void *pdata)
{
    INT32U clk;
    pdata = pdata;
    while (1)
    {
        OSTimeSet (0L) ; // reset system clock
        LED-on (1);
        OSTimeDly (4);
        clk = OSTimeGet ( ) ; // get value of clock
        UART0-sendData (clk) ; // display to UART0
        OSTimeDly (4);
    }
}
```

Paste screenshots :

Comments : (i) OSTime set() is used to reset system clock

(ii) OSTimeGet() is used to get current value of the clock.



Logic Analyzer

Min Time: 0.357068 ms

Max Time: 5.384924 s

Range: 20.00000 s

Grid: 1.000000 s

6.6 Conclusion

1) What is overall conclusion of this experiment ?

⇒ (i) OSTimeDly() depends on ticks per sec. & fosc.

(ii) Priority of tasks may also affect the delay because of cpu utilization.

(iii) OSTimeDlyHmsm() used to give delay using hours, minutes, seconds & millis.

(iv) OSTimeSet() is used to set clock while OSTimeGet() is used to get value of the clock.

2) What you learnt from this experiment ?

⇒ (i) Resolution, accuracy & precision.

(ii) OS_TICKS_PER_sec & fosc

(iii) Effect of more tasks on delay

(iv) OS Time DlyHmsm()

(v) OSTimeGet() & OSTimeset() Functions

3) Care performed by you in RTOS in industry.

⇒ Refer to datasheet. While specifying parameters, calling functions will take care of proper arguments and libraries. Xtal & fosc both should be equal

6.7 Activity

⇒ (i) We can use Hard RTOS which is time crucial & bounds to the time.

(ii) Tasks are strictly delievered (executed within the given time..

(iii) Applications : missiles, aircraft, etc.