Lecture 6: Jitter

Jitter = time between actual arrival of a task period about to execute and its expected arrival; total jitter will be the quadrature sum of all independent causes of jitter on a task period's arrival.
Jitter tolerance = largest amount of jitter that can be handled by a task scheduler and still meet its deadlines; typically it is the largest time possible between a task period's actual completion and its deadline.

Sources of task timing jitter:

-poor scheduling
-poor priority handling
-bad algorithms in support packages (eg printf)
-spurious or noise interrupts
-bus contentions (I/O devices)
-non-fixed loop iteration count
-non-fixed number of recursive calls
-garbage collection
-conditional statement execution asymmetry (e.g. if A || B)
-hardware cache unpredictable/unrepeatable performance
-data dependent instruction execution time
-nested multiple priority interrupts
-clock vs delay granularity
-task dependencies (shared resource pre-emption, semaphore availability, synchronous communication)
-context switching of tasks
-hardware latency
-interrupt and priority handling
-scheduler bookkeeping
-clock latency
-spurious interrupting due to noise, power surges, electrical discharges
(watch out: spurious interrupts can cause death spiral of scheduler missed deadlines and run-time stack memory overflow)

Traditional way of measuring execution time:

```
Start_counter();
for(I=1; I<1000000; I++)Get_interrupt();
Stop_counter();
```

-naïve: function call overhead, context switch overhead, for loop time, etc.
-better to use logic analyzer or scope with a probed interrupt register flag
-easy to test and measure, hard to isolate
-interrupts are good for fast response time, but have high resource overhead
-polling is CPU intensive (slow), but cause low resource overhead
-polling jitter => polling period plus/minus epsilon
-interrupt jitter => higher priority interrupt preemption time

Real-time clock requirements and assumptions (clock overheads are clearly defined):
-correctness: (clock – real time) < epsilon always
-bounded drift: $dc/dt < p + 1$ (for crystals $p < 10^{-5}$ => 1 sec/day; for earth $p < 10^{-8}$ => 1 sec/year)
-monotonicity: $c_2 > c_1$ if $t_2 > t_1$ (eg no Y2K phenomena)
-chronoscopicity: $d^2c/dt^2 <$ gamma (i.e. if $t_2 - t_1 = t_4 - t_3$, then $c_2 - c_1 = c_4 - c_3$ +/- gamma)

Lecture 6: Jitter (cont.)

External clock issues:
-H/W register updating time and latency
-process synchronization and read computation time
-read process interrupt latency

GPS systems require above issues to be well managed and defined:
Basically how it works is that you solve 4 unknowns with 4 equations using at least 3 satellites:
t_i and r_i are the measured time and distance (respectively) between satellite i and the desired object at location (to, xo, yo, zo) emitting a signal that the satellite can detect that you are trying to locate.

$r_i = ct_i$, where c is the known speed of light (3×10^8 m/s).

$r_i = (|xo - x_i|^2 + |yo - y_i|^2 + |zo - z_i|^2)^{1/2}$ for satellites i=1 to 3.

With each of the r_i, we can now solve for the other 3 unknowns (xo,yo,zo) to find the desired object.