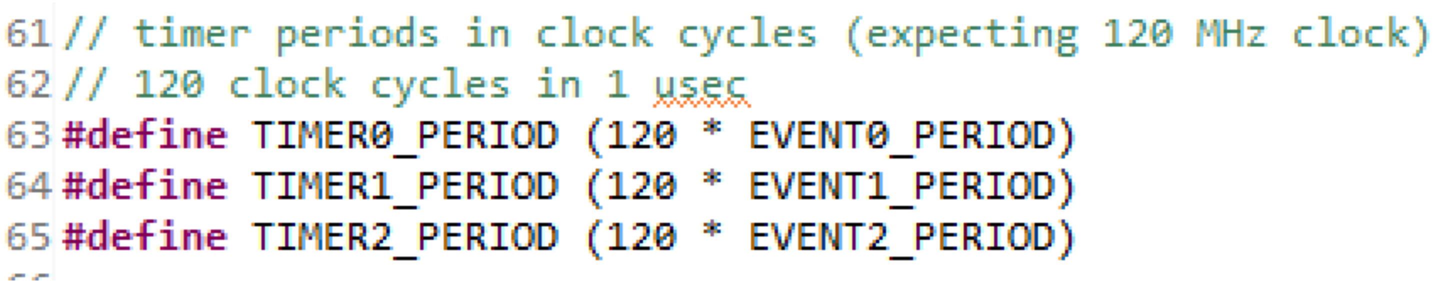
ece3849_int_latency
TM4C1294NCPDT Timer Summery

- CORRECTION !!!!!!!
- There are two General Purpose Timers (GPTM Modules)
  - There are 8 GPTM timer modules.
  - Each GPTM timer has two sub timers - TimerA and B can be used individually in 16-bit mode or together for 32-bit counts.
  - They can count up or down.
  - They can be a one shot timer or a periodic timer that reloads itself when the time is reached.
  - See datasheet for more details.
- This example uses 3 GPTM timers using submodule TimerA.
  - Each event uses an interval timer with an interrupt output to trigger the event handler task.
- The the timer units are in number of CPU clocks.
  - The CPU is running at 120 MHz.
  - There are a 120 clocks in 1 usec.

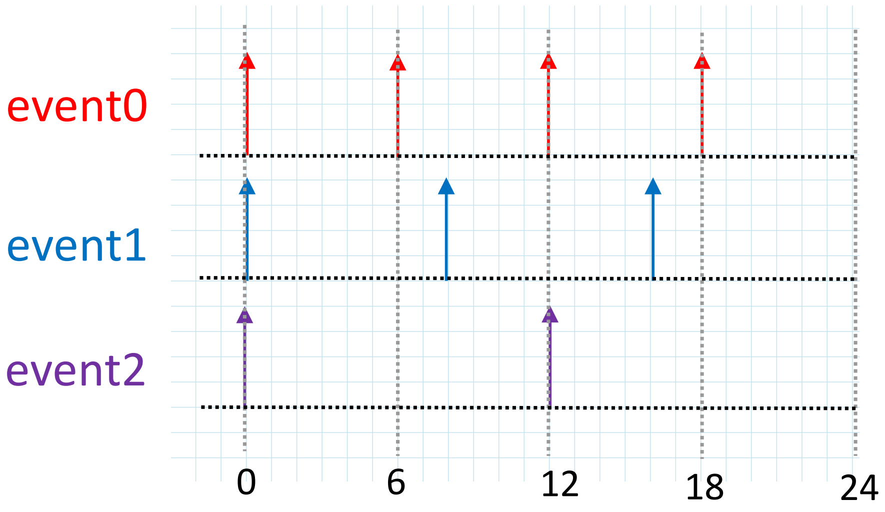


```
// timer periods in clock cycles (expecting 120 MHz clock)
// 120 clock cycles in 1 usec
#define TIMER0_PERIOD (120 * EVENT0_PERIOD)
#define TIMER1_PERIOD (120 * EVENT1_PERIOD)
#define TIMER2_PERIOD (120 * EVENT2_PERIOD)
```

Canonical Real-Time Systems

- System Assumptions / Rules
  1. All Events are periodic.
  2. The relative deadline = period.
     - The event needs to finish before it can be called again.
  3. The events are not phase aligned and can happen at anytime relative to each other.
- Example: There are three events: event0, event1 and event2

Event	Period	Execution Time
event0	6 ms	2 ms
event1	8 ms	1 ms
event2	12 ms	2.5 ms

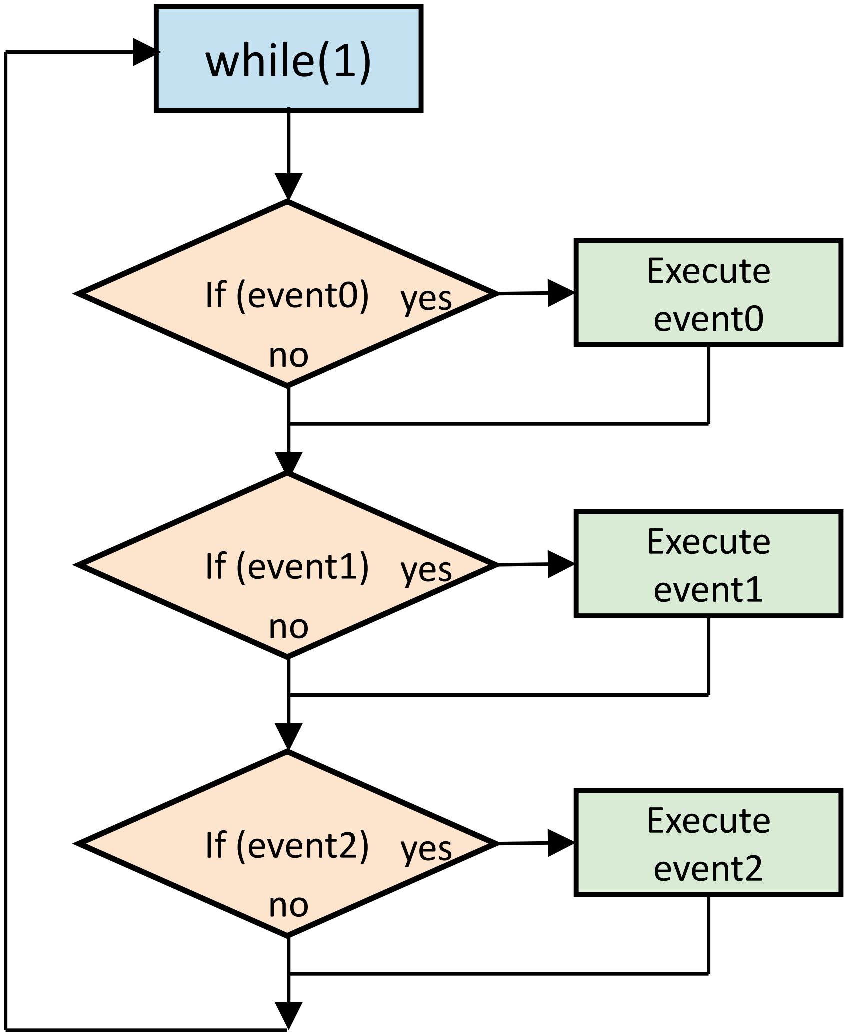


We have three events to service, any concerns?

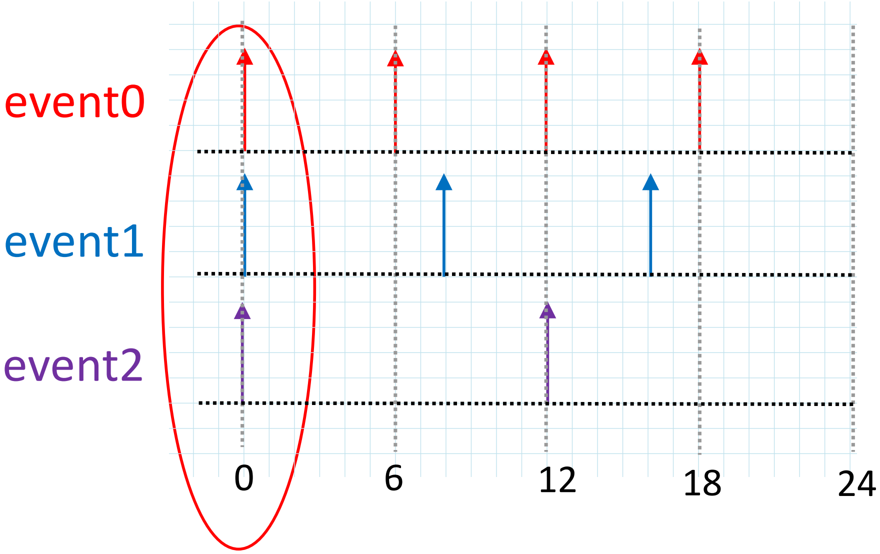
- How do we know if they make deadline?
- Are they preemption? Do we have it? We not have defined a scheduling algorithm.

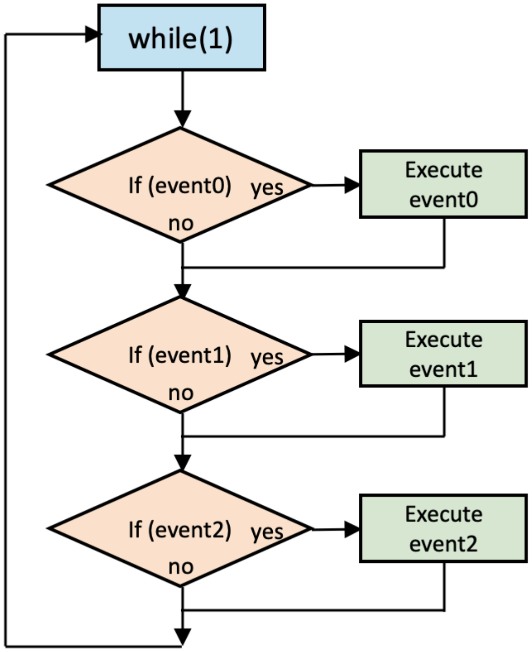
Polling Loops Without Preemption

- Simplest way to run is a polling loop.
  - No priority => no preemption.
  - Each event is checked in sequence.
    - Each event is guaranteed to run. (starvation-free)
- Round-Robin Scheduling
  - The status of each event is checked (polled).
  - If an event occurred, it executes to completion.
  - Then repeats process for the next event.
  - When all events have been checked returns to the beginning



Round Robin Polling

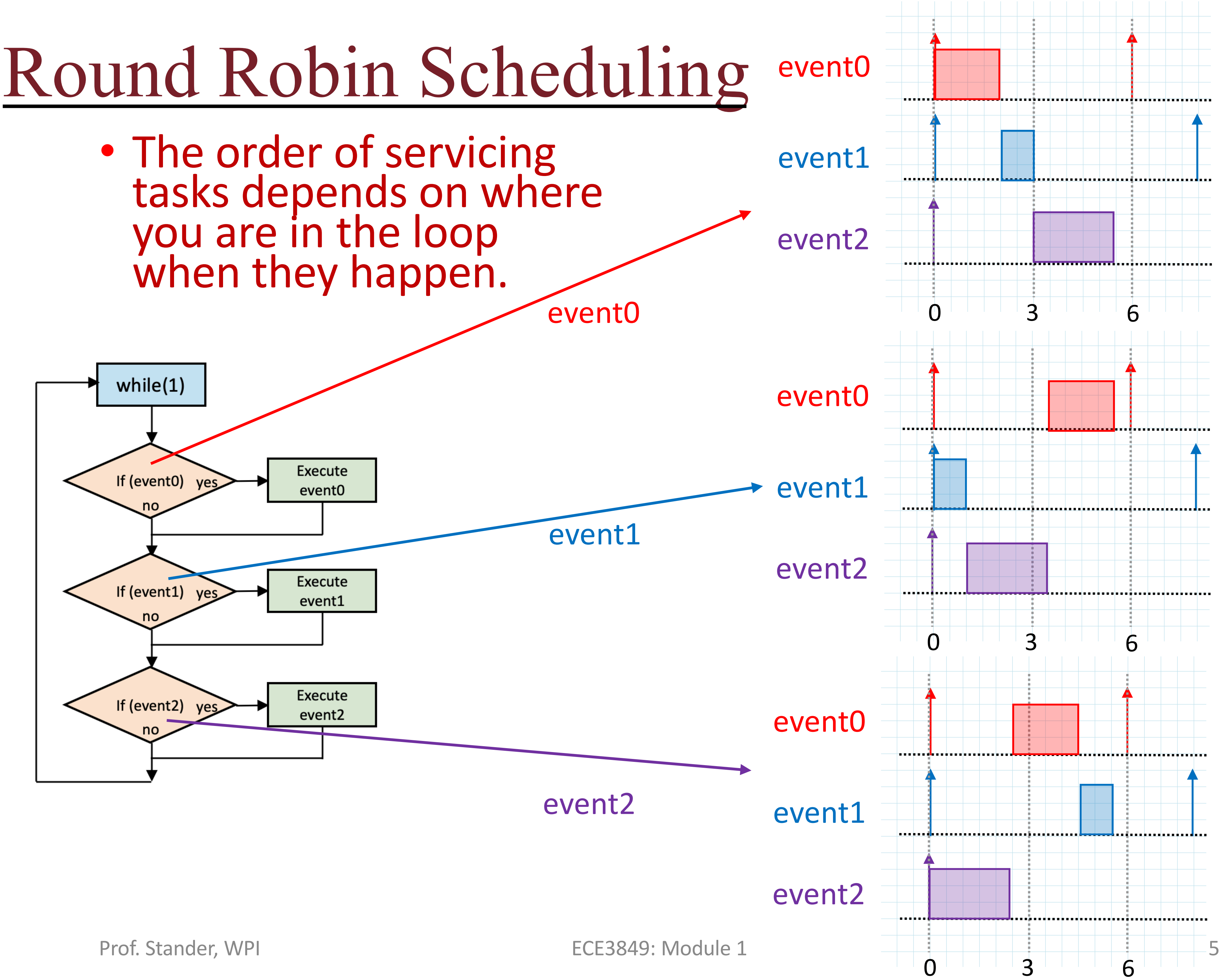




- For the example,
  - All three events occur at relative time 0.
  - Polling loop has been running for a while.
- Which task get serviced first?
  - All equality important no priority?
  - We naturally want to 0 cause it is first, but loop can be any where in its execution.
  - We don’t know.

Round Robin Scheduling

- The order of servicing tasks depends on where you are in the loop when they happen.



Are the tasks schedulable?

Event	Period	Execution Time
event0	6 ms	2 ms
event1	8 ms	1 ms
event2	12 ms	2.5 ms

Remember, we care only about maximum value for these calculations.

```
void main (void) {
  <init>;  // pseudo-code in <...>

  while (1) {
    if (<event0 occurred>) {
      <handle event0>;
    }
    if (<event1 occurred>) {
      <handle event1>;
    }
    if (<event2 occurred>) {
      <handle event2>;
    }
  }
}
```

t_{exec0} = 2 ms

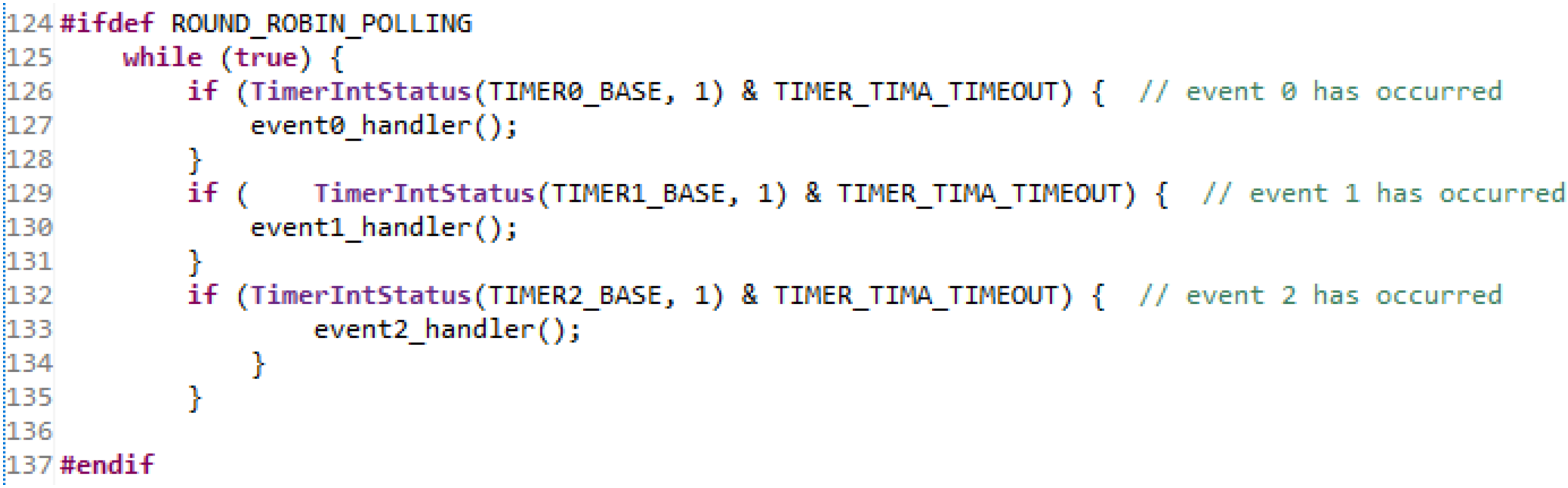
t_{exec1} = 1 ms

t_{exec2} = 2.5 ms

Event	Period	Execution Time	Latency	Response Time	Relative Deadline	Schedulable ?
event0	6 ms	2 ms	1 event 1 + event 2 = 1 + 2.5 =3.5	Latency+ execution max = 3.5 + 2m = 5.5m	5.5 m < 6m	YES
event1	8 ms	1 ms	2 + 2.5 = 4.5m	1 + 4.5 = 5.5	5.5 m < 8m	YES
event2	12 ms	2.5 ms	1 + 2 = 3	5.5	5.5 < 12 m	YES

ece3849_int_latency Round Robin Polling

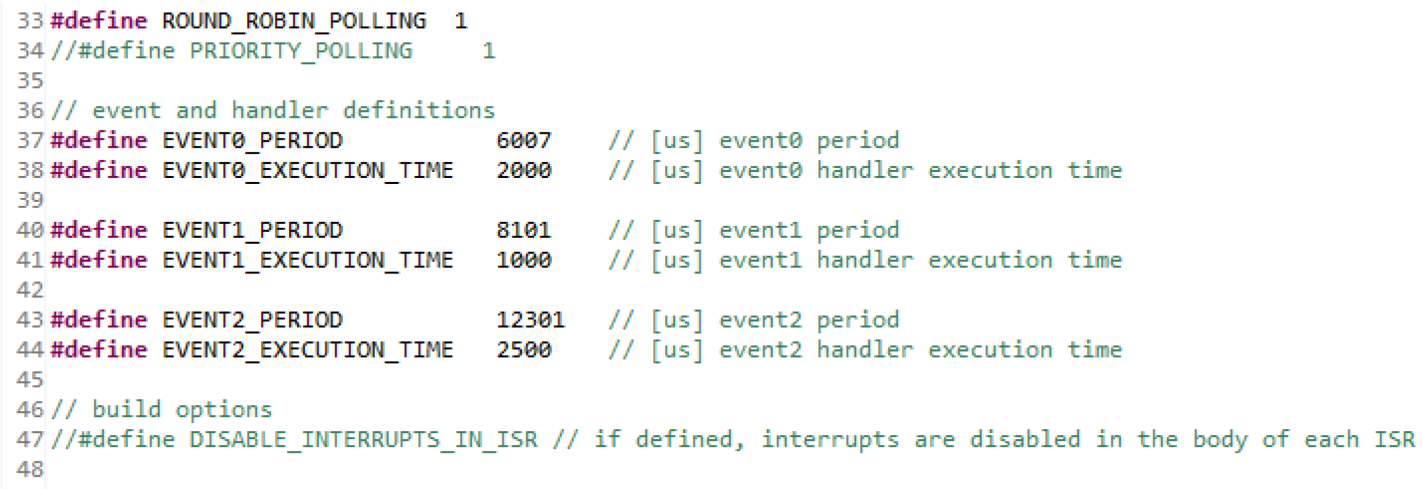
- In the Round Robin Polling option the infinite polling loop contains three if statements.
- The Interrupt status of each event is checked to see if an event occurred.
    - It is masked with the TimerA timeout interrupt bit, TIMER_TIMA_TIMEOUT.
        - This separates the event status from other unrelated interrupts.
    - If the Interrupt for that event occurred, the interrupt handler function is called.



```
#ifdef ROUND_ROBIN_POLLING
  while (true) {
    if (TimerIntStatus(TIMER0_BASE, 1) & TIMER_TIMA_TIMEOUT) {  // event 0 has occurred
      event0_handler();
    }
    if (  TimerIntStatus(TIMER1_BASE, 1) & TIMER_TIMA_TIMEOUT) {  // event 1 has occurred
      event1_handler();
    }
    if (TimerIntStatus(TIMER2_BASE, 1) & TIMER_TIMA_TIMEOUT) {  // event 2 has occurred
        event2_handler();
      }
    }

#endif
```

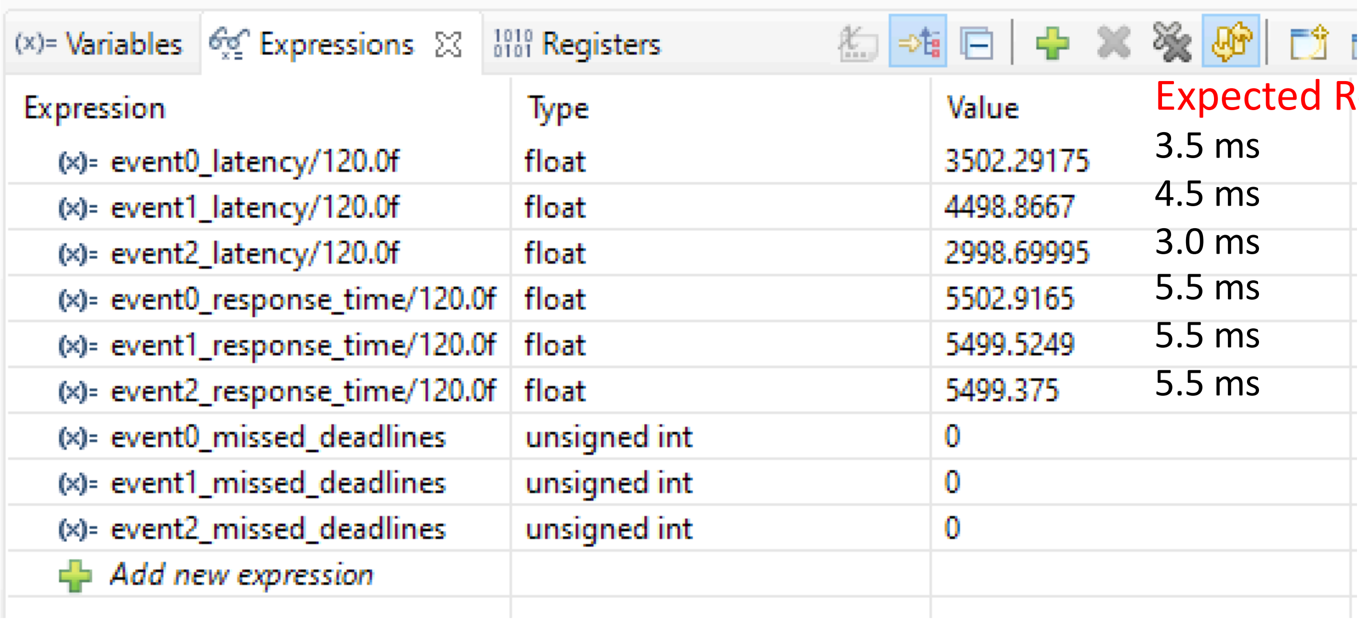
ece3849_int_latency
Example 1: event2 = 2.5

- Run settings



```
#define ROUND_ROBIN_POLLING  1
//#define PRIORITY_POLLING     1

// event and handler definitions
#define EVENT0_PERIOD          6007    // [us] event0 period
#define EVENT0_EXECUTION_TIME  2000    // [us] event0 handler execution time

#define EVENT1_PERIOD          8101    // [us] event1 period
#define EVENT1_EXECUTION_TIME  1000    // [us] event1 handler execution time

#define EVENT2_PERIOD          12301   // [us] event2 period
#define EVENT2_EXECUTION_TIME  2500    // [us] event2 handler execution time

// build options
//#define DISABLE_INTERRUPTS_IN_ISR // if defined, interrupts are disabled in the body of each ISR

```

- Run results



Expression	Type	Value	Expected Results
(x)= event0_latency/120.0f	float	3502.29175	3.5 ms
(x)= event1_latency/120.0f	float	4498.8667	4.5 ms
(x)= event2_latency/120.0f	float	2998.69995	3.0 ms
(x)= event0_response_time/120.0f	float	5502.9165	5.5 ms
(x)= event1_response_time/120.0f	float	5499.5249	5.5 ms
(x)= event2_response_time/120.0f	float	5499.375	5.5 ms
(x)= event0_missed_deadlines	unsigned int	0	
(x)= event1_missed_deadlines	unsigned int	0	
(x)= event2_missed_deadlines	unsigned int	0	
Add new expression			

Example 2: Event 2 = 3.5 ms

- Lets try Round Robin Again

t_{exec0} = 2 ms

t_{exec1} = 1 ms

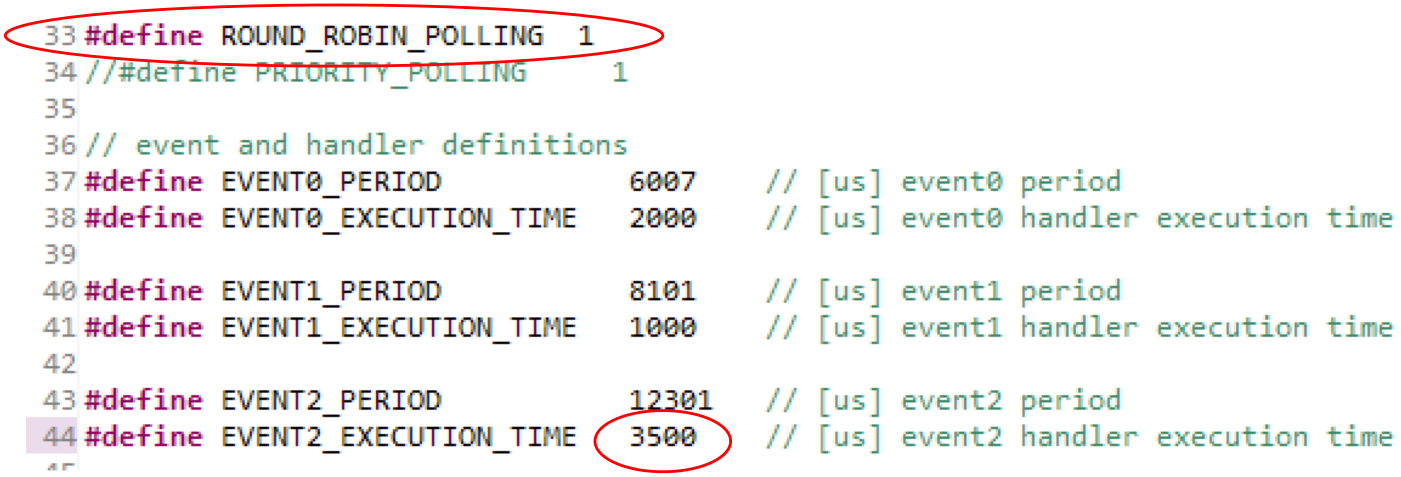
t_{exec2} = 3.5 ms

```
void main (void) {
  <init>;  // pseudo-code in <...>

  while (1) {
    if (<event0 occurred>) {
      <handle event0>;
    }
    if (<event1 occurred>) {
      <handle event1>;
    }
    if (<event2 occurred>) {
      <handle event2>;
    }
  }
}
```

Event	Period	Execution Time	Latency (sum of other events execution times)	Response Time (Latency + execution time)	Relative Deadline (Period)	Schedulable ? YES = response < deadline
event0	6 ms	2 ms	1 + 3.5 m = 4.5	4.5 + 2 = 6.5	6.5 > 6	NO
event1	8 ms	1 ms	2 + 3.5 = 5.5	1 + 5.5 = 6.5	6.5 < 8	YES
event2	12 ms	3.5 ms	2+1 = 3	3.5 + 3 = 6.5	6.5 < 12	YES

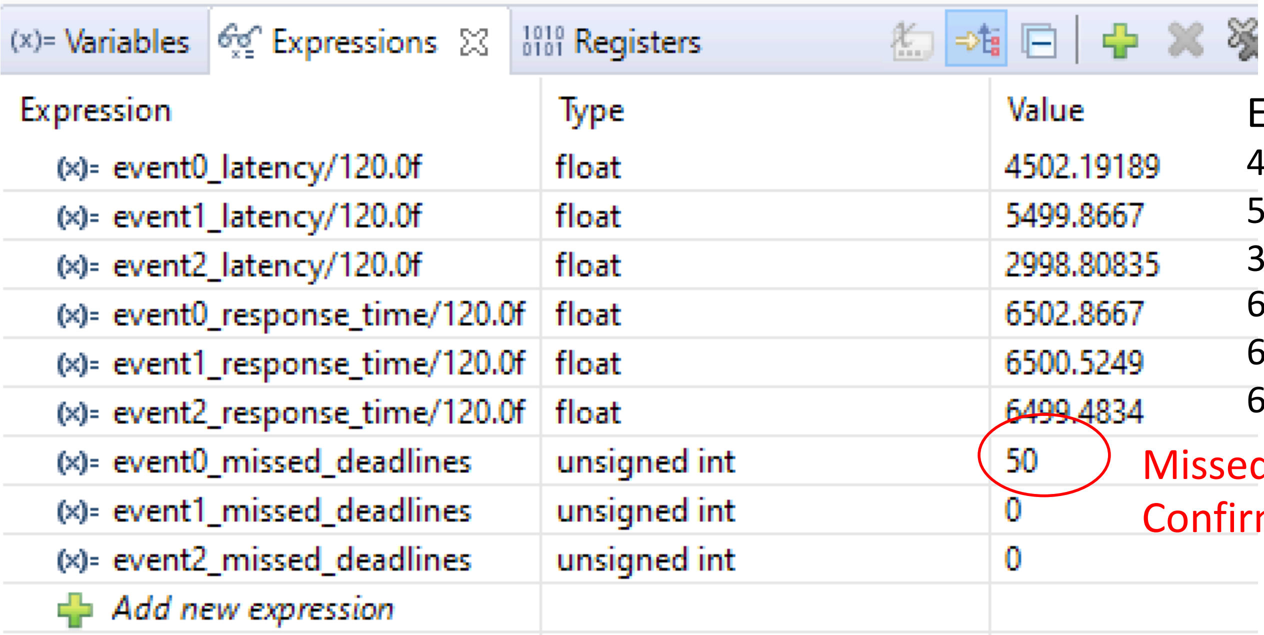
ece3849_int_latency Example 2: event2 = 3.5 ms

- Run settings



```
#define ROUND_ROBIN_POLLING  1
//#define PRIORITY_POLLING     1

// event and handler definitions
#define EVENT0_PERIOD           6007    // [us] event0 period
#define EVENT0_EXECUTION_TIME   2000    // [us] event0 handler execution time

#define EVENT1_PERIOD           8101    // [us] event1 period
#define EVENT1_EXECUTION_TIME   1000    // [us] event1 handler execution time

#define EVENT2_PERIOD           12301   // [us] event2 period
#define EVENT2_EXECUTION_TIME   3500    // [us] event2 handler execution time
```

- Run results



Expression	Type	Value	Expected Results
(x)= event0_latency/120.0f	float	4502.19189	4.5 ms
(x)= event1_latency/120.0f	float	5499.8667	5.5 ms
(x)= event2_latency/120.0f	float	2998.80835	3.0 ms
(x)= event0_response_time/120.0f	float	6502.8667	6.5 ms
(x)= event1_response_time/120.0f	float	6500.5249	6.5 ms
(x)= event2_response_time/120.0f	float	6499.4834	6.5 ms
(x)= event0_missed_deadlines	unsigned int	50	
(x)= event1_missed_deadlines	unsigned int	0	
(x)= event2_missed_deadlines	unsigned int	0	
Add new expression			

Missed Deadlines Confirmed!

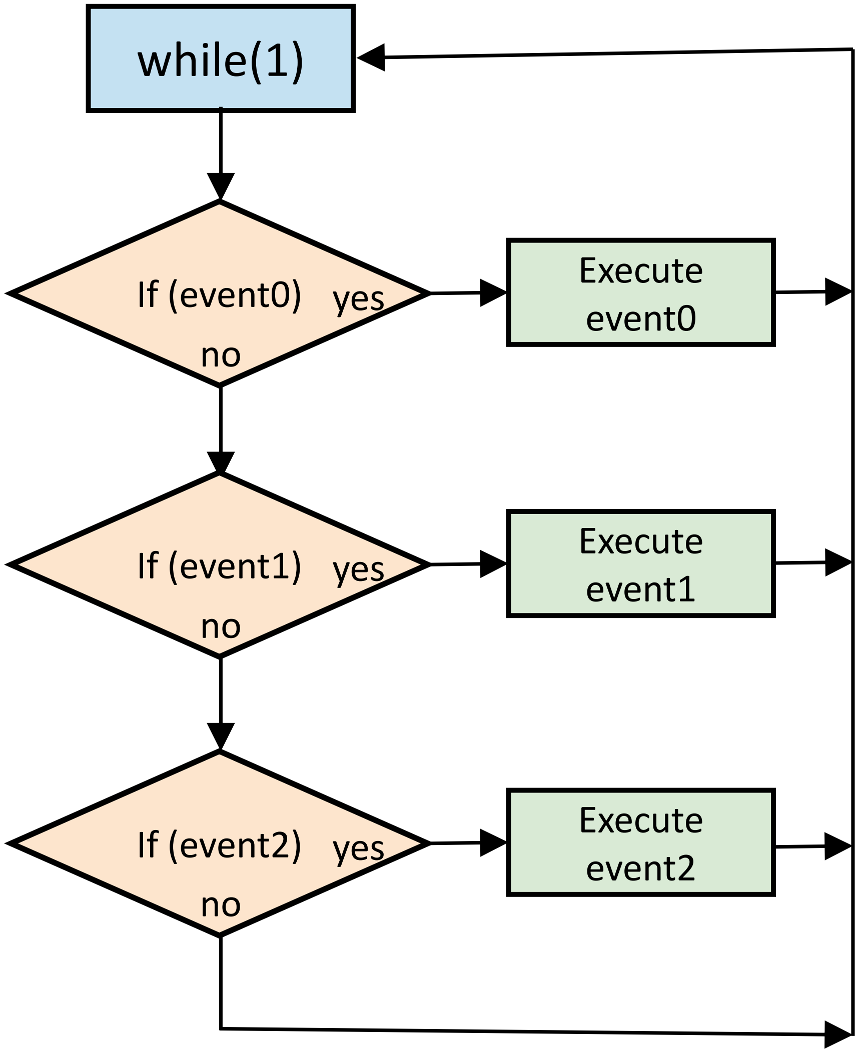
Example 2, event2 = 3.5 ms Round Robin is not schedulable

- The execution time of all the events factor into the response time of ALL the events.

- What is the bottleneck in this case?
  - Short event has the most problems?
- How might we schedule differently?
  - Recheck our requirements and see if we could the period long?
  - Different polling – prioritize shortest first.

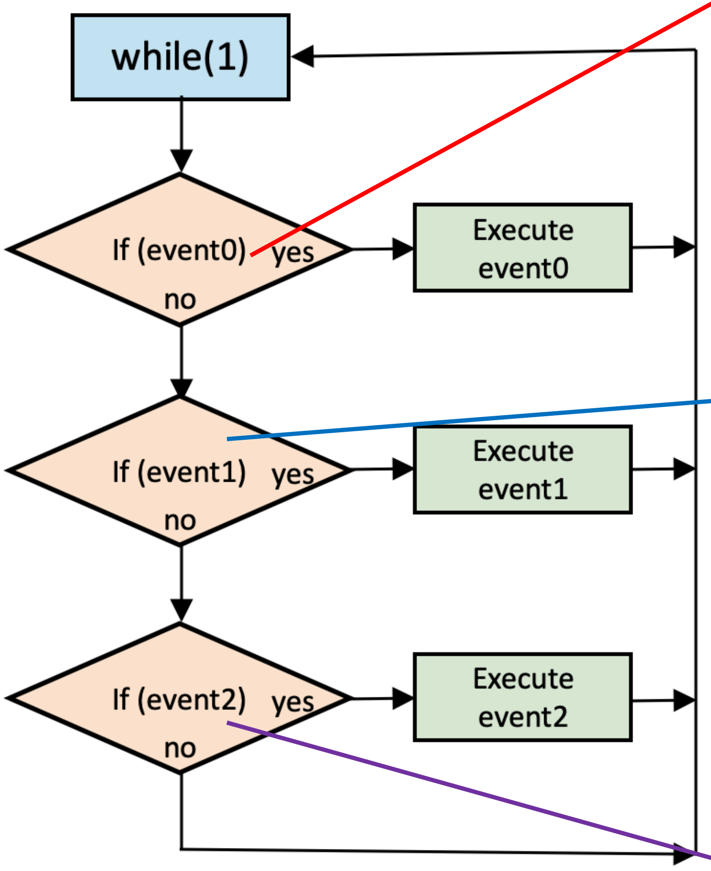
Priority Polling: Shortest deadline first

- event0 will always be serviced.
- Only if event0 is not waiting will event 1 be serviced.
- Only if event0 and event1 are not waiting will event 2 be serviced.



- What is likely to happen if event0 takes most of the CPU time?
    - Possibly 1 or 2 is starved?

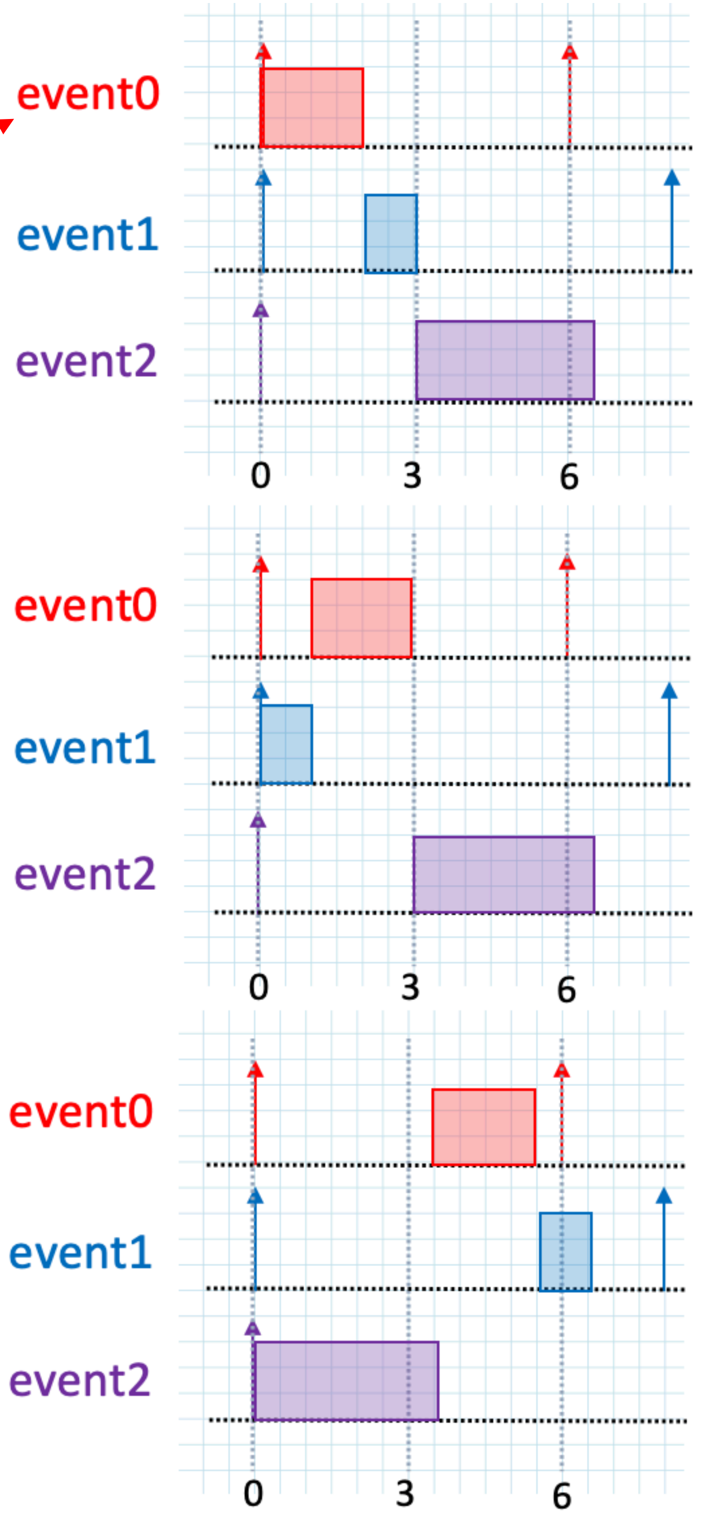
Priority Polling: event0 latency



event0 latency = 0

event0 Latency = event1 execution time

event0 latency = Event2 execution time



- Max event0 latency = max(event1 execution time , event2 execution time)

Priority Polling

```
void main (void) {
  <init>;

  while (1) {
    if (<event0 occurred>) {
      <handle event0>;
    }
    else if (<event1 occurred>) {
      <handle event1>;
    }
    else if (<event2 occurred>) {
      <handle event2>;
    }
  }
}
```

t_{exec0} = 2 ms

t_{exec1} = 1 ms

t_{exec2} = 3.5 ms

Event	Period	Execution Time	Latency	Response Time (Latency + execution time)	Relative Deadline (Period)	Schedulable ? YES = response < deadline
event0	6 ms	2 ms	3.5 m (max of 1 OR 3.5)	5.5	5.5 < 6	YES
event1	8 ms	1 ms				
event2	12 ms	3.5 ms				

- What will happen if event0 period gets shorter?
- ?