Lecture 04 Notes – More Synchronization

I. Overview

A. **Review:**

1. Looked at Sync and Do for triggering processing from events, time
2. Sync for triggering handler process = Detect event + (schedule handler process + dispatch handler process) + execute handler process
3. Basic Sync methods for software process
   a. Blocking loop, etc.

II. Sync (…) Do

A. **How to get information from ISR (producer) to main thread (consumer)?**

1. Why?
   a. Timing: doing all the work in the ISR delays other processing (lower-priority ISRs, all threads)
      i. WaveGen
      ii. Scope
   b. Other: software structure, etc.
2. Is one example of **inter-process synchronization.** What is **communication?**
   a. Sync: something happened
   b. Communication: data describing what happened, typically needs sync too
3. Starting Example: main loop + ISR
   a. Structure
      i. Shared Variable: Event Flag
         - 1 = it happened, 0 = nothing happened
      ii. Processes
         - ISR writes 1 to event flag
         - Main thread: While 1 loop
           - Tests each event flag
           - If flag is 1, clear it to 0 and do processing
      iii. SW scheduler uses SW and HW
         - Event Detection
         - Scheduling
         - Dispatch
         - Handler/Work/Compute

4. Consider behavior for abnormal cases
    a. OK for consumer to miss events (e.g. in event burst)?
        i. Yes: Count to 1, and no farther
        ii. No: Use integer variable to count number of pending events (happened but not processed)
            - Producer increments events_pending (ep)
            - Consumer decrements events_pending
    b. OK to produce events if consumer hasn't consumed enough?
        i. Buffer size limits, etc.
        ii. Producer needs to synchronize by checking events_pending before producing event
    c. Implementation
        i. Decide how system should behave, add to requirements
            - Hardware processes have behaviors defined for these cases
        ii. Implement the behavior
            - Configure hardware (if available)
            - Bare metal (no SW support): algorithms in your code
            - Support from OS/RTOS or programming language
5. Can also communicate data in shared variables
    a. Event Flag + Data Value = Synchronization + Communication
    b. Multiple pending events possible?
        i. Also need to save data for each event (queue, FIFO buffer)
        ii. How large to make buffer?
            - Depends on rates of data production and consumption, which depends on input events, time to execute processes, when/how many times processes get to execute
6. Deeper look at triggering sync behaviors possible.
    a. Can producer process generate another event if consumer process hasn't gotten it yet?
        i. No: Lock-step
        ii. Yes: How many events are possible
    b. Counting? Track number of pending, unserviced events,

III. Sync and Don't: Mutual Exclusion

A. **Motivating Example 1: Two processes updating shared variable**

B. **Motivating Example 2: Motor Position/Speed controller with Zero Limit Switch**

1. Processes
2. Failure Cases
    a. QD pulses while ZLS is closed – add test
    b. ZLS interrupt during QuadDec Inc/dec of position variable

C. **Support**

1. Hardware
2. Instructions & Algorithms
3. OS/RTOS, Programming Language?