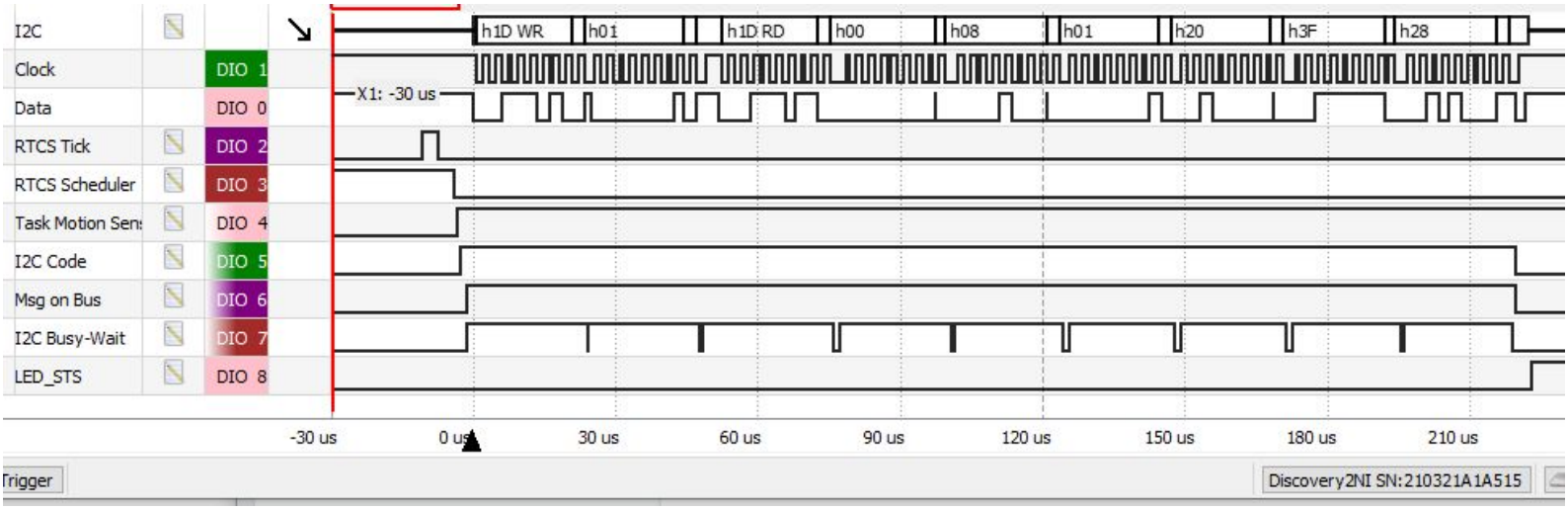
Bennett James
10/24/19
ECE560

1. Existing Read



Task Motion Sensor is active the whole time the code is running. Never goes low.

Task Motion Sensor runs every 20 ms as per the add task function that is rate and the Tick rate declared for it at 50Hz.

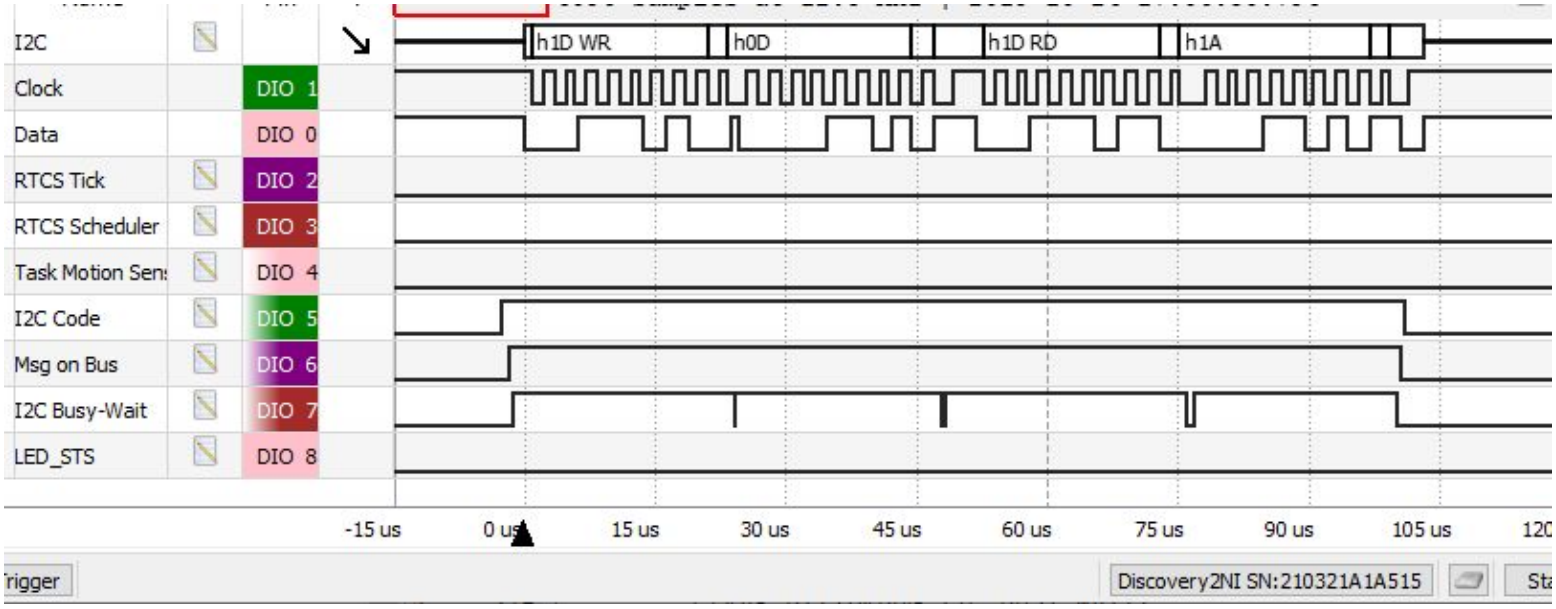
For one message the Message on bus signal is high for 221.4 microseconds.

For a single message transmission the I2C Busy wait signal is

25.35 + 23.14 + 27.69 + 23.53 + 22.49 + 22.49 + 22.62 + 22.49 + 22.62 = 212.42 microseconds

Bennett James
10/24/19
ECE560

2. Existing Write



The Msg on Bus signal is high the whole time write is called for which is about 102.6 microseconds.

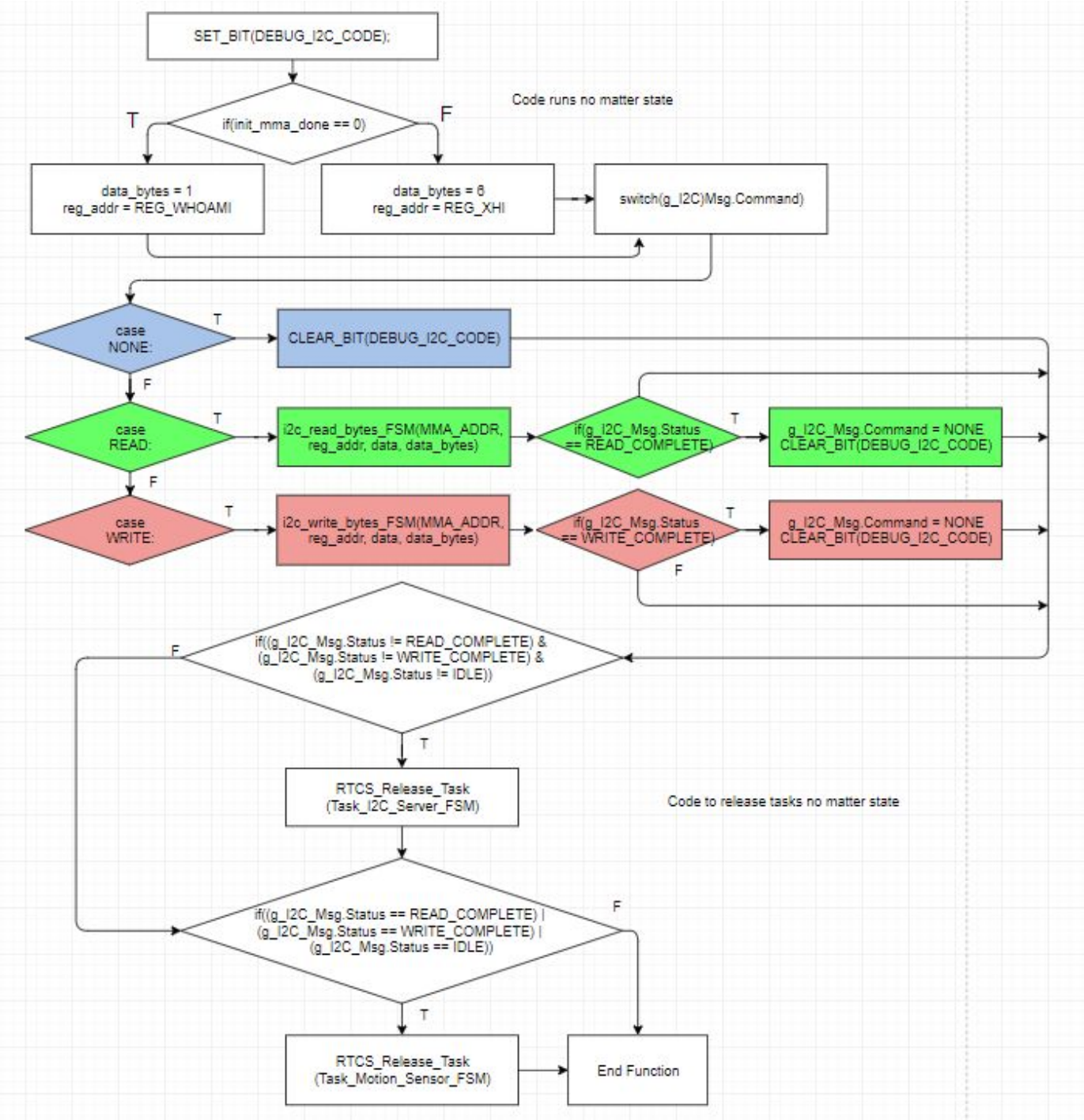
I2C Busy Wait is high for about 100.1 microseconds during write.

25.4 + 23.6 + 27.7 + 23.4 = 100.1 microseconds

Bennett James
10/24/19
ECE560

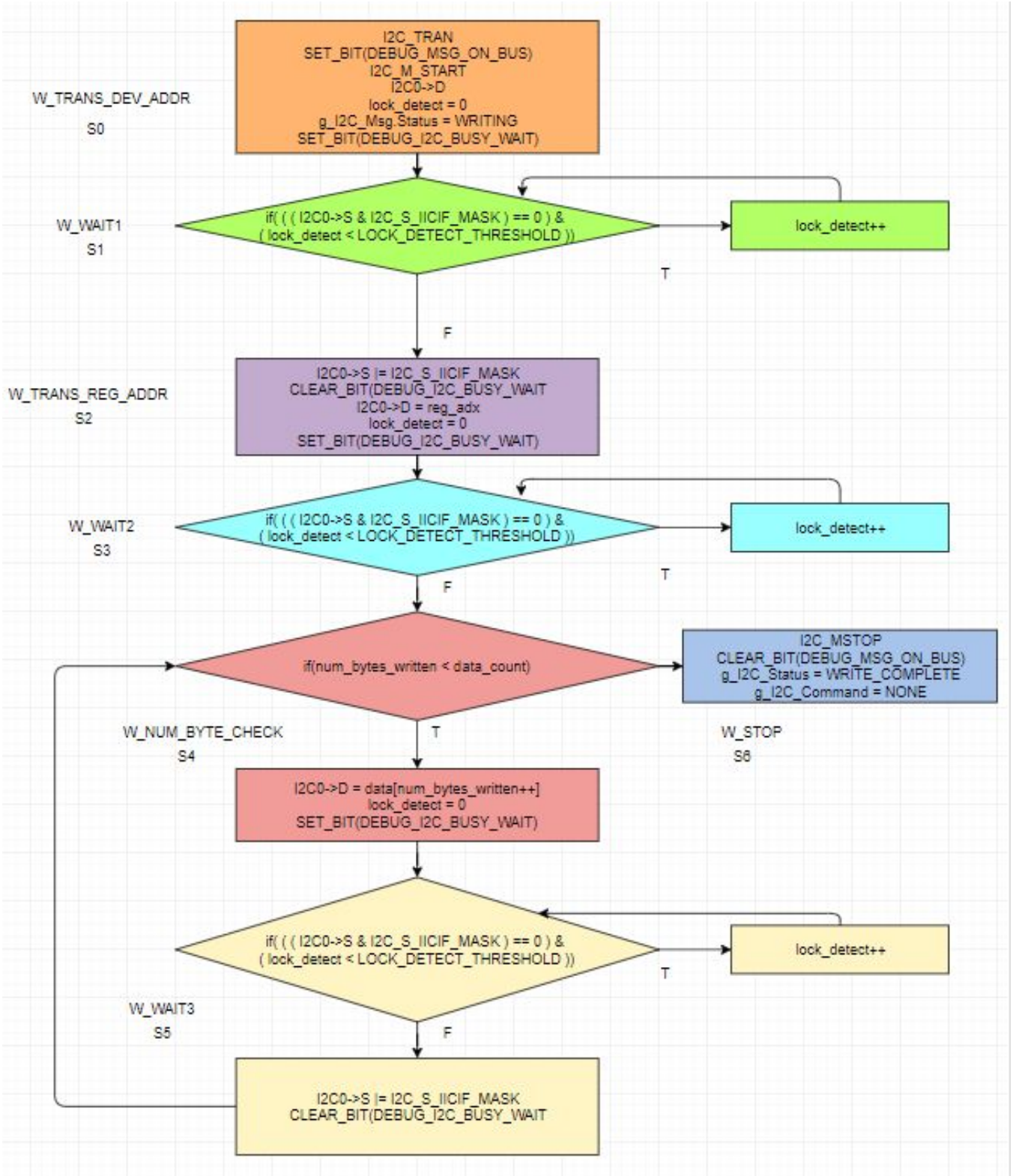
3. Task_I2C_Server_FSM function

For the Task_I2C_Server_FSM it is running in response to the Task_Motion_Sensor_FSM states. There are only 3 actual states that are described in the case statement which are the g_I2C_Msg.Command switch statement. The i2c_write_bytes_FSM and i2c_read_bytes_FSM have more FSM logic and are shown below as well.



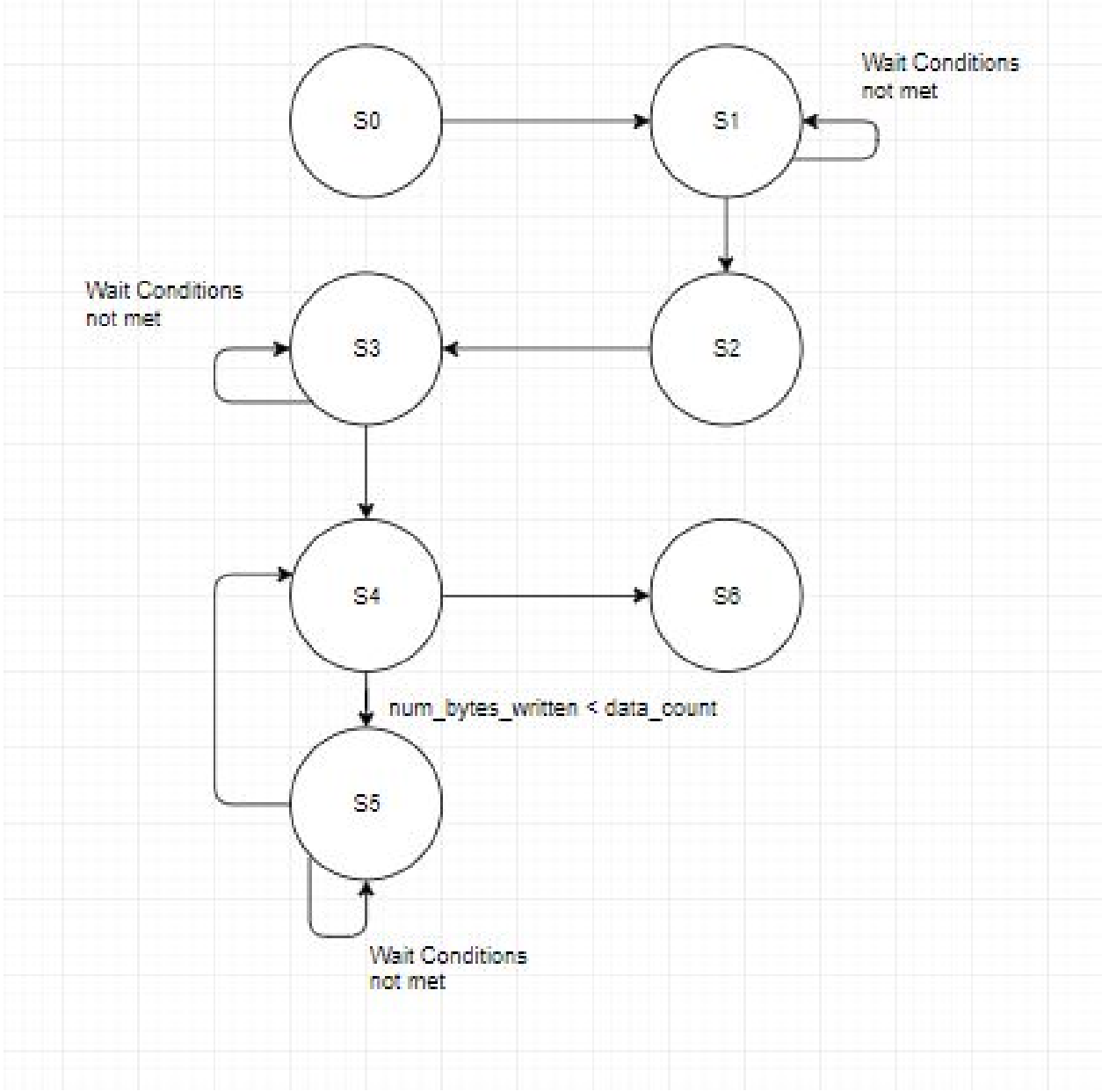
Bennett James
10/24/19
ECE560

Flowchart for i2c_write_bytes_FSM



Bennett James
10/24/19
ECE560

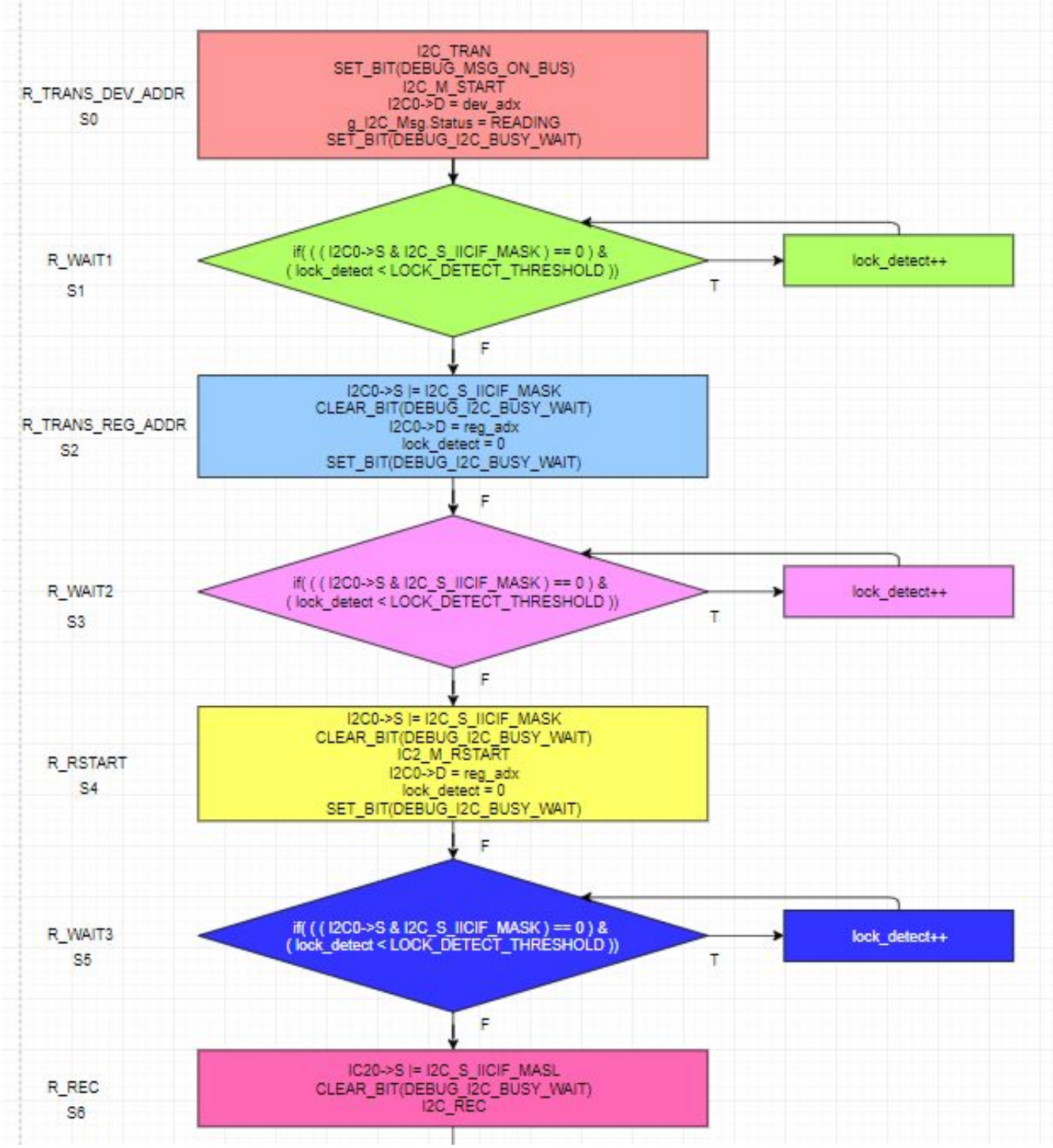
State Transition Diagram for Write FSM Operation



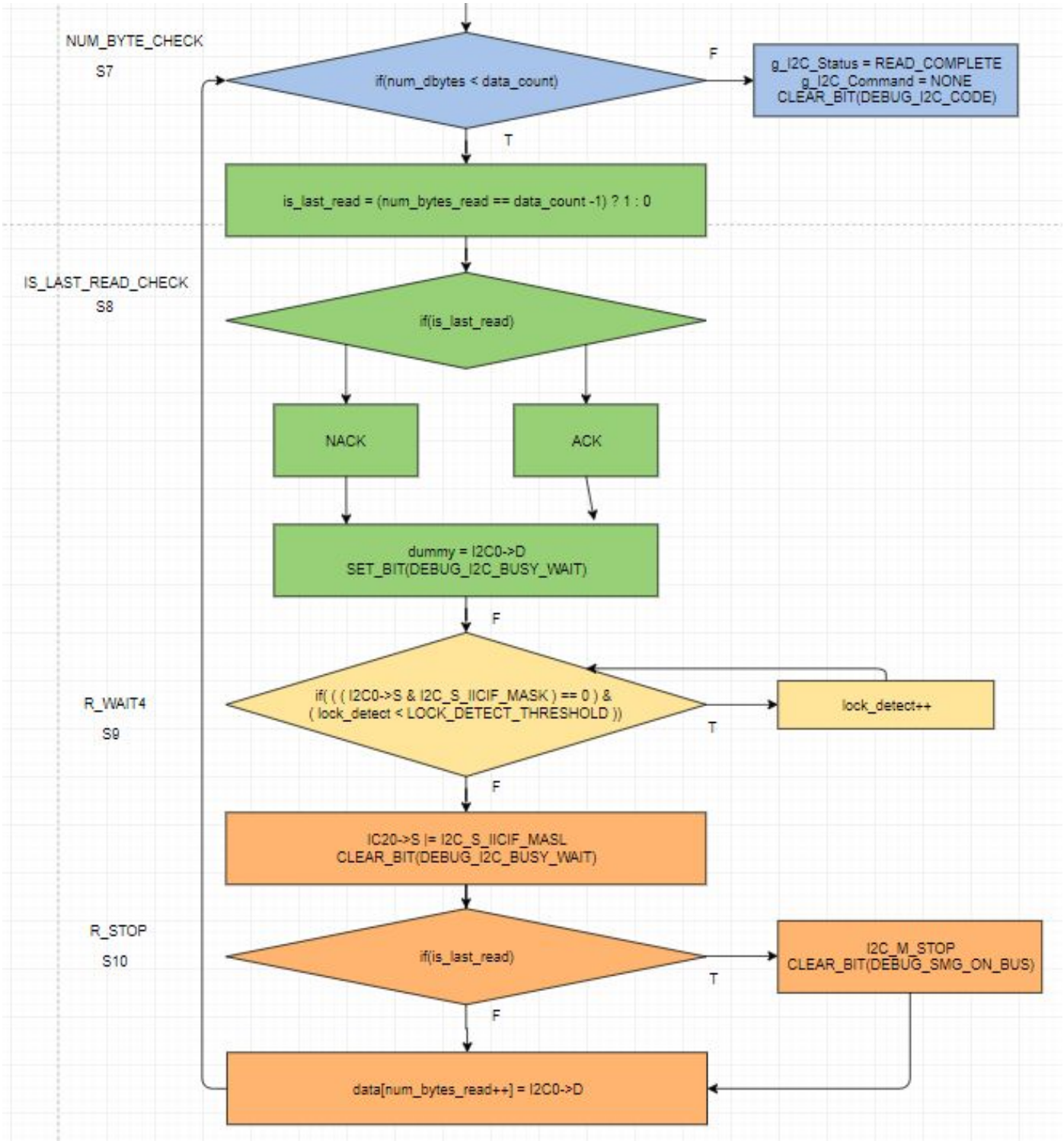
For ease of design the condition for “Wait Conditions” were not explicitly stated here but can easily be seen in each state in the corresponding states in the flowchart

Bennett James
10/24/19
ECE560

Flowchart for i2c_read_bytes_FSM

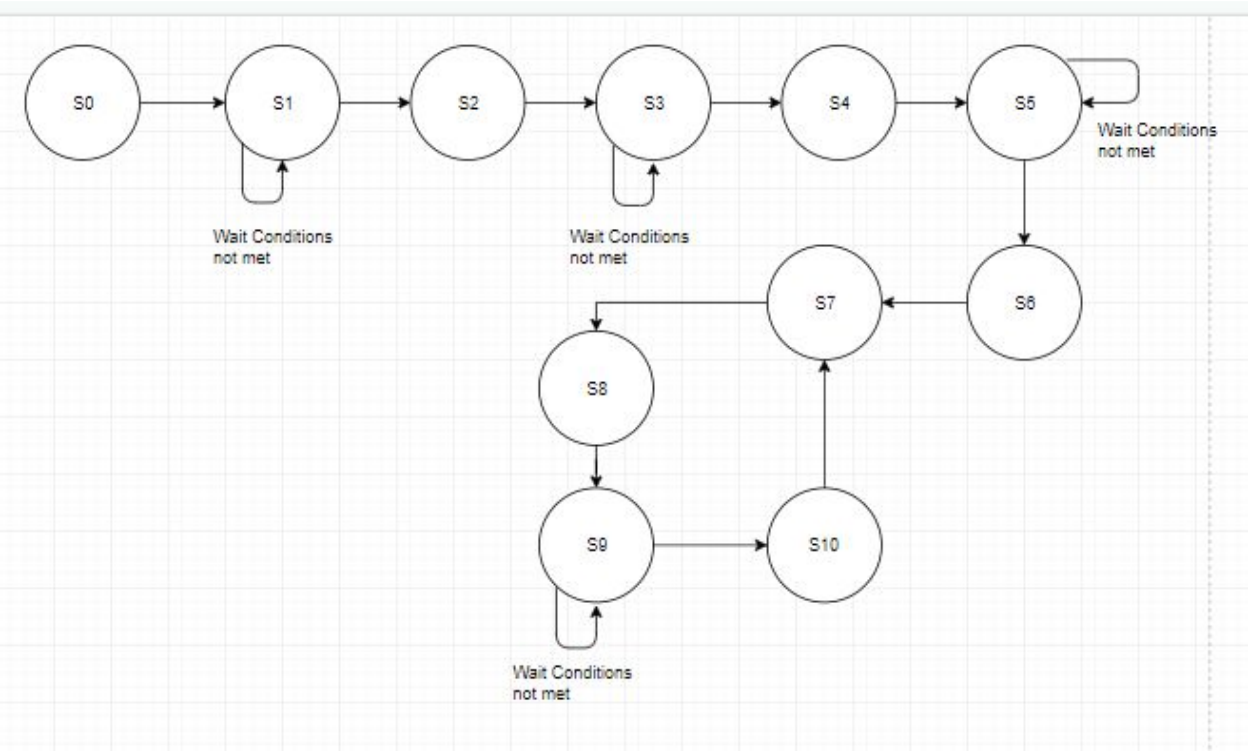


Bennett James
10/24/19
ECE560



Bennett James
10/24/19
ECE560

State Transition Diagram for Read FSM Operation

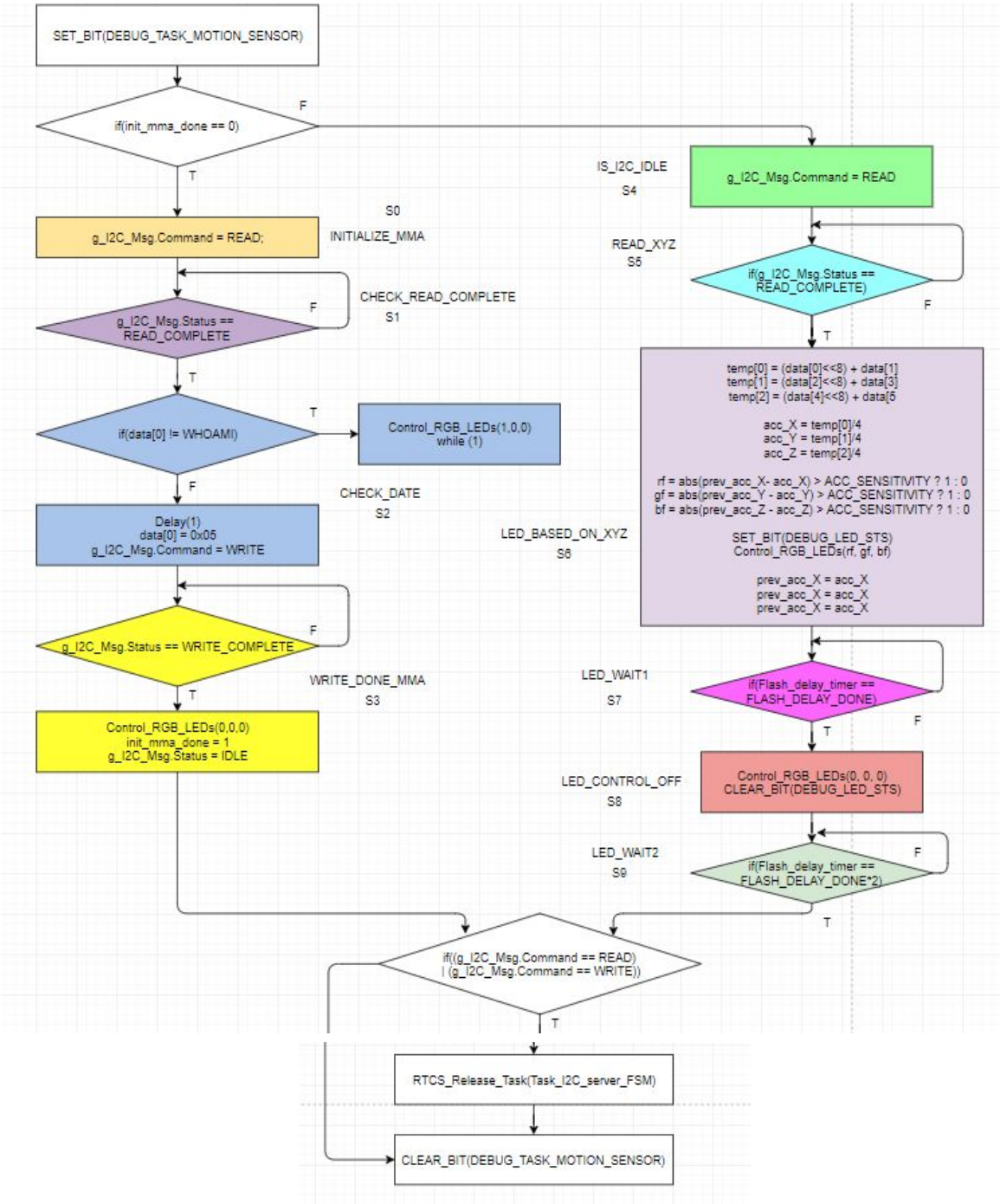


For ease of design the condition for “Wait Conditions” were not explicitly stated here but can easily be seen in each state in the corresponding states in the flowchart

Bennett James
10/24/19
ECE560

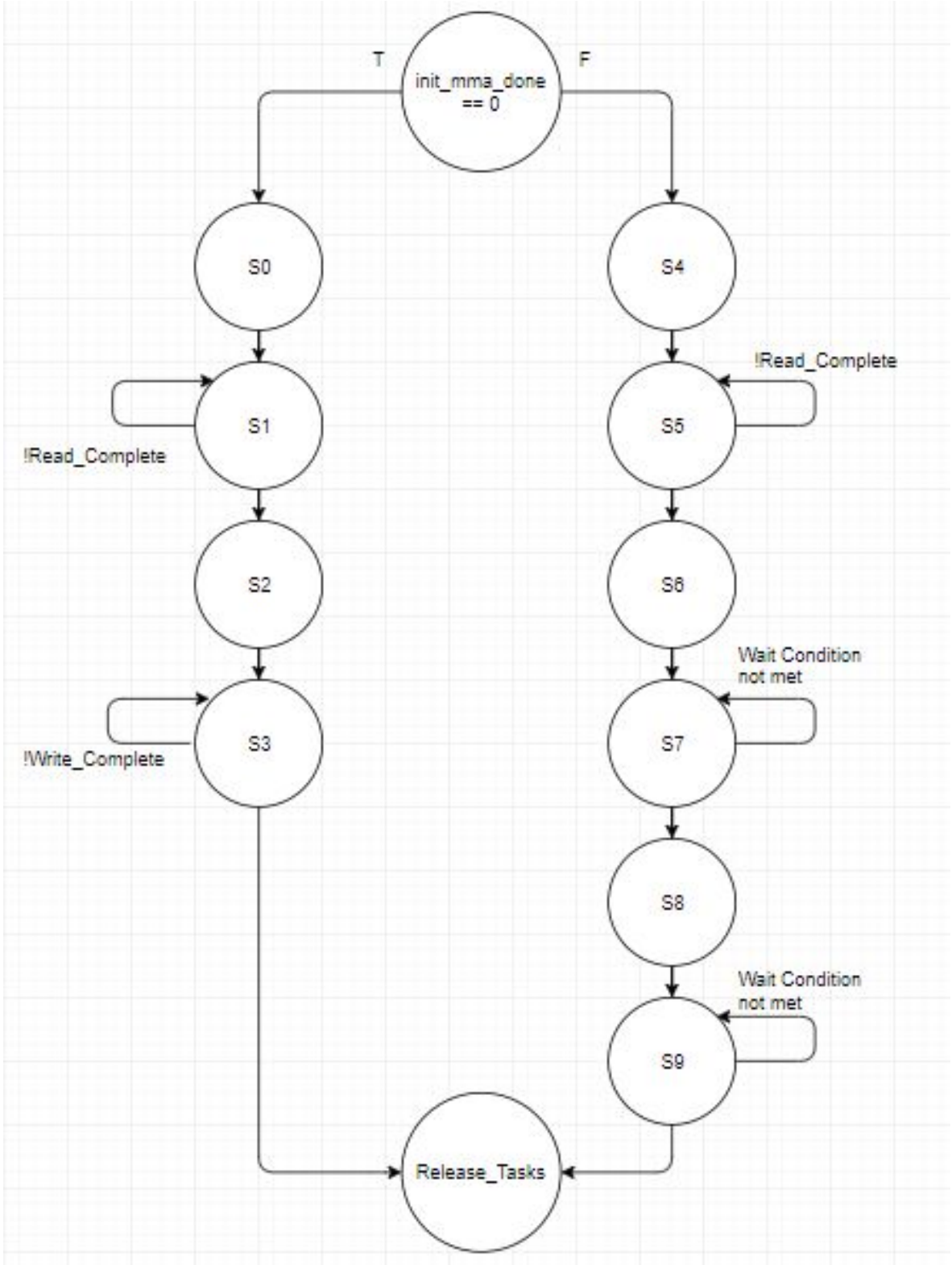
4. Task_Motion_Sensor_FSM function

This function must account for both the init_mma function and communicating with the Task_I2C_Server_FSM to read the accelerometer and once reading is complete to flash the LEDs accordingly.



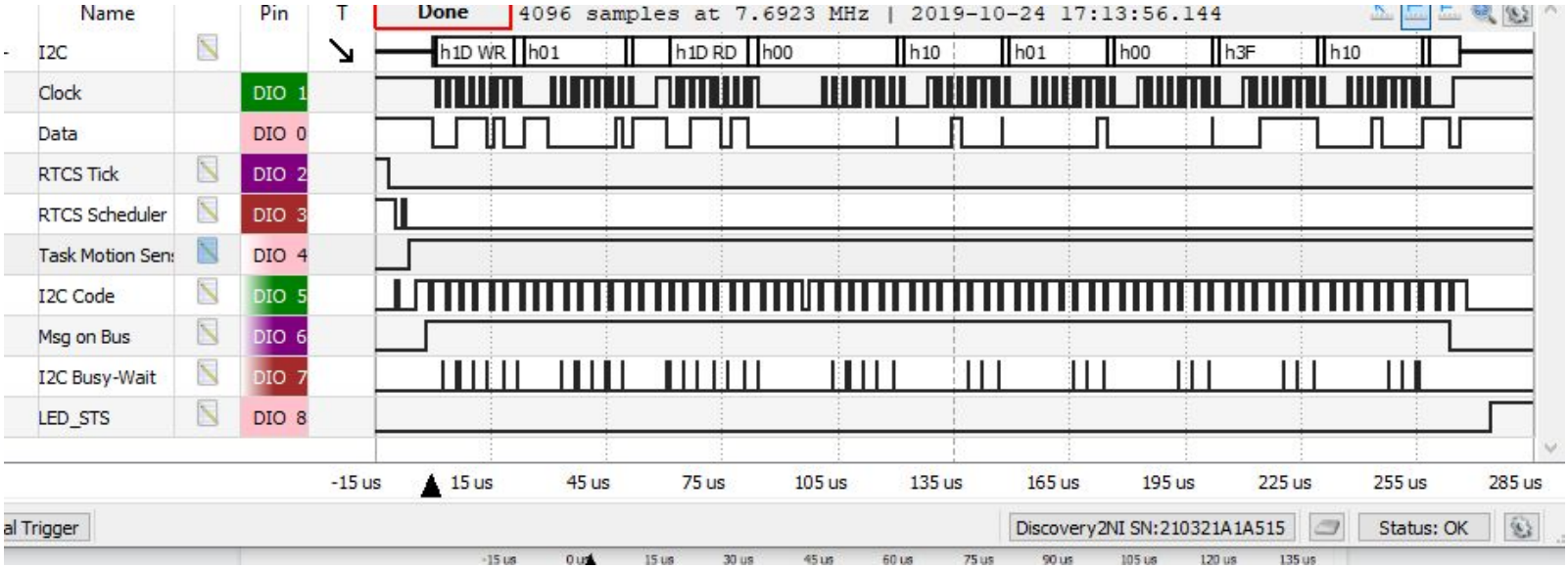
Bennett James
10/24/19
ECE560

State Transition Diagram for Read FSM Operation

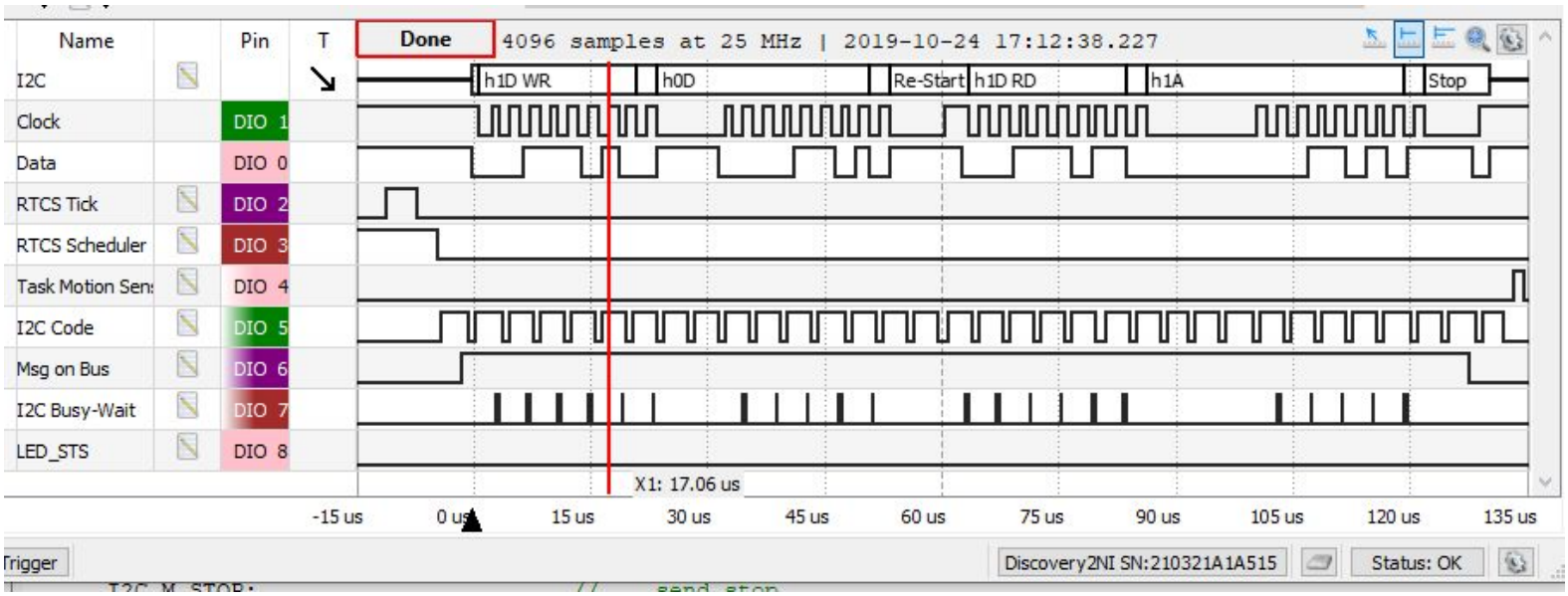


Bennett James
10/24/19
ECE560

5. I2C Read Operation Update



6. I2C Write Operation during accelerometer initialization

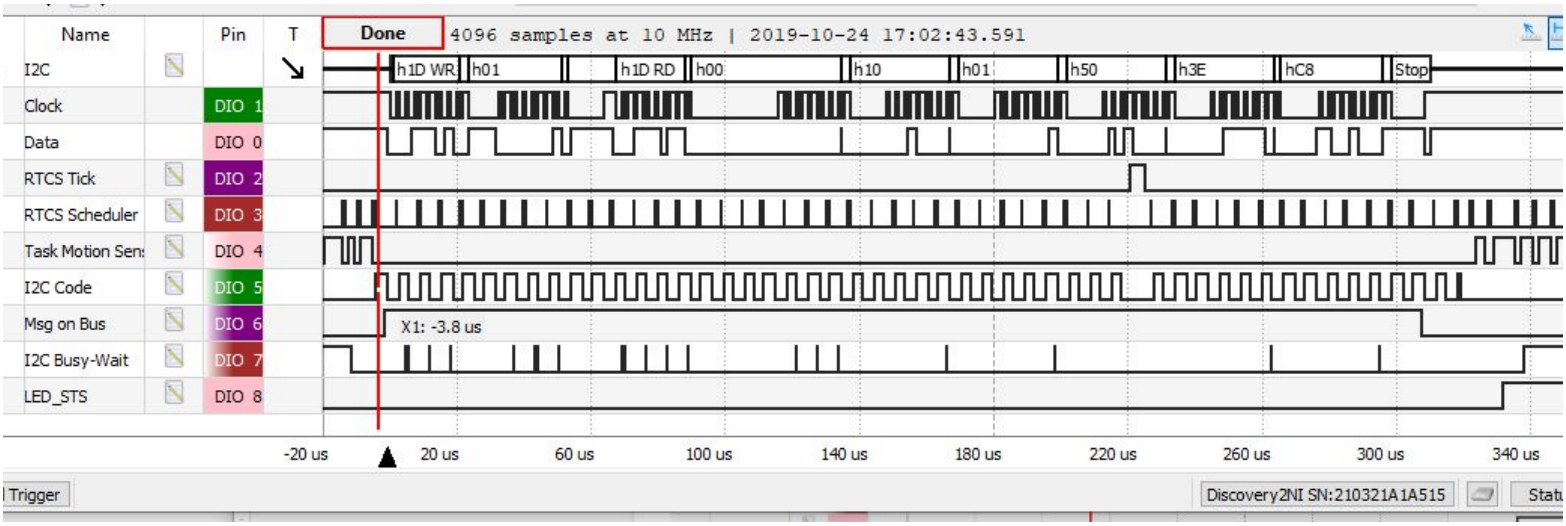


Bennett James
10/24/19
ECE560

7. Trigger Tasks Approach
    - When deciding how to trigger my tasks I used the trigger hints given and came up with for
        - Task_Motion_Sensor_FSM
            - This task would release Task_I2C_Server_FSM if g_I2C_Msg.Command was equal to read or write
            - It would also adjust the period for which Task_Motion_Sensor_FSM would occur to account for delay function2
        - Task_I2C_Server_FSM
            - Would trigger itself if status was equal to reading or writing
            - Would trigger Task_Motion_Sensor_FSM if status was equivalent to read_complete or write_complete.

Bennett James
10/24/19
ECE560

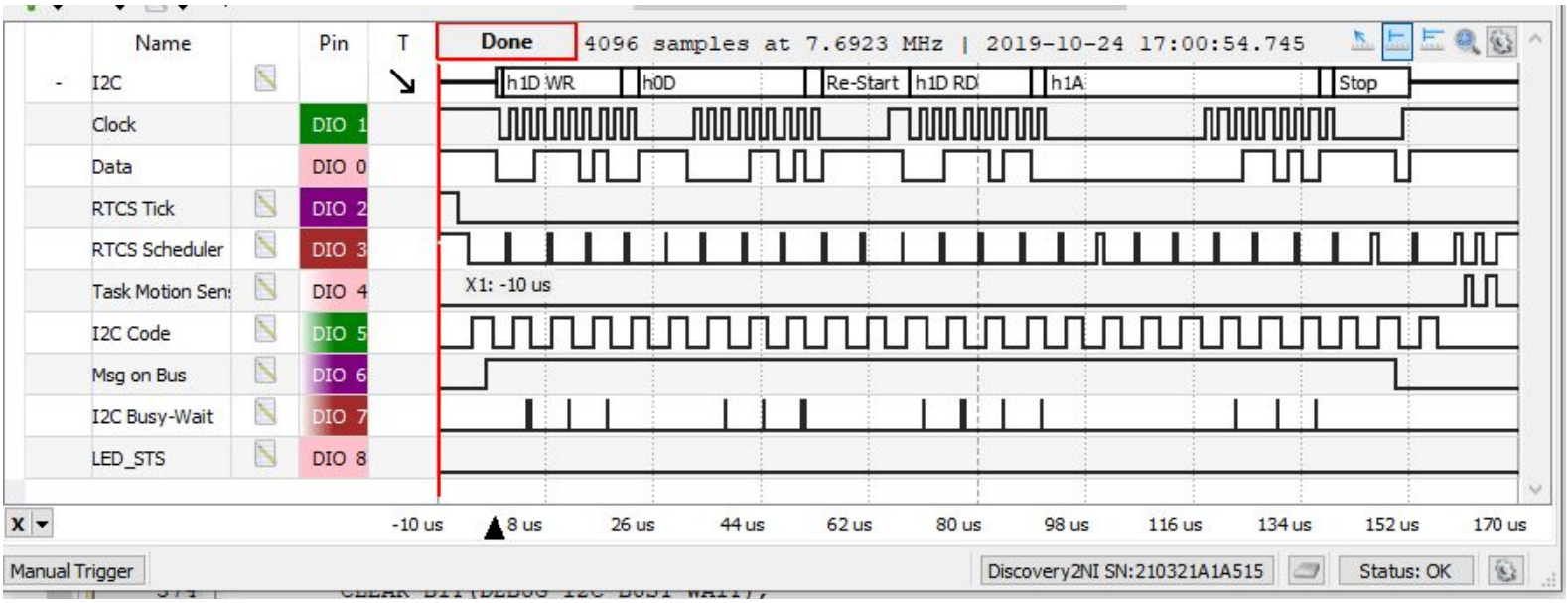
8. I2C Read Operation: Event Triggered



This screenshot shows the Task_Motion_Sensor_FSM goes high for a slight period of time at the beginning and calls Task_I2C_Server_FSM to run its FSM until the read is complete and once complete the Task_Motion_Sensor_FSM will process these values and light the LEDs accordingly. The other FSM is triggered through event triggering. At the end of the second picture you can also see the I2C_Busy_Wait signal go high signifying that the LED is in a delay and the LED is on.

Bennett James
10/24/19
ECE560

9. I2C Write Operation: Event Triggered



This shows the I2C communications that were in the init_mma function when in blocking but have been moved to an FSM.

10. Time Delay between sending a read request to the LED light turning on.
    a. Delay now in the optimized code with FSM it takes 281.5 microseconds from the time the read command is sent for the LED to turn on.
    b. Delay from initial code whenever just the FSM was used and before the Release_Task calls were used the delay was 278.4 microseconds.
    c. How does the delay depend on the period of each task?
        i. I based the delay that was used for the FLASH_DELAY based upon the PIT timer.. By adding a variable in the interrupt that incremented every time the interrupt was called for. I measured the timing off the Delays by running the delay function and determining how long many ticks of the PIT interrupt it went through.

If any of my diagrams are difficult to read they can also be seen at this link.
[Flowcharts Drawings, must open in draw.io to see all]