Università di Genova

Embedded Systems

On-going Assessment

Prof. Enrico Simetti

On-going Assessment

1/ Simulate an algorithm that needs 7 ms for its execution and needs to work at 100 Hz. This is to emulate a real-world scenario.

2/ Make LD2 blink at 1 Hz (500 on, 500 off).

3/ Read characters from UART1. If a string $RATE,xx* is received, the xx dictates the frequency of the magnetometer feedback on the UART. Valid data for xx are 0, 1, 2, 4, 5, 10 Hz. Invalid values (e.g., -1, 3.2, 3, 11) should be discarded and a message $ERR,1* should be sent on the UART. Initialize the system with a value of 5 Hz.

4/ Acquire the three magnetometer axes at 25 Hz (set the data rate to 0b110 in the 0x4C register).

5/ Do the average of the last 5 measurements.

6/ Send it to the UART at xx Hz using the protocol $MAG,x,y,z*, where x is the average value on the x-axis, y is the average value of the y-axis, and z is the average z-axis value.

7/ Compute the angle to the magnetic North using atan2(y_avg, x_avg)

8/ Send the computed angle at 5 Hz using the message $YAW,x*, where x is the angle in degrees.

Suggestions

- The availability of only 3 ms to perform operations suggests the use of interrupts to handle the UART, both in reception and transmission.
- The suggested way is to use separate circular buffers to store data coming and going to the UART.
- Given that the IMU allows an SPI clock up to 7.5 MHz, the SPI interactions can be consider negligible (two bytes require around 2 microseconds at 7.5 MHz).
- Make sure to handle properly shared data between interrupts and the main (i.e., disabling interrupts and creating a critical region in the main code).
- Do you have any warning when compiling? Solve them! Sometimes, a warning hides a nasty bug (e.g., truncation due to overflow.)

Evaluation Criteria

- Are the tasks executed at the correct frequency? Are there missed deadlines because of some erroneous handling of the peripherals? Are busy-waiting performed when allowed?
- Is the shared data handled correctly? Is there any possibility of wrong sequence of interrupts that might cause some erroneous behaviour?
- Are circular buffers sized correctly? i.e., have you actually thought on how fast data is produced/received? Memory is a scarce resource in embedded systems.
- Are the interrupts short in time? Are you performing unnecessary busy-waiting inside interrupts? Note: not all while loops are busy-waiting loops.
- Is the code sufficiently commented, well written, formatted? Maintainability and quality of the code is key in embedded applications.

Final Notes

- In case of doubt, ask! Interactions are not forbidded, actually they are recommended.
- Only a single group participant should submit the final project on Aulaweb. Please submit the whole project folder.
- Use your group ID when creating the project in MPLAB X.

Template Code

```
void algorithm() {
  tmr_wait_ms(TIMER2, 7);
}
int main() {
  tmr_setup_period(TIMER1, 10);
  while(1) {
    algorithm();
    // code to handle the assignment
    ret = tmr_wait_period(TIMER1);
  }
}
```

Università
di Genova