Lab 2 - Pre-Lab & Pseudo Code

Lucy Utke and Noah Terry

Introduction

Describe the purpose of the lab in the context of the overall project in 1-2 paragraphs. What functionalities will be added, what will they enable?

The purpose of this lab is to interface with the ATmega32U4's Timer 0, an 8-bit timer. This timer can be set up such that it creates an interrupt within our programs every millisecond, allowing for reliable timing information. This timing information will be critical towards sampling sensor input, as well as interpreting encoder ticks. It will also add additional functionality to the MessageHandling section built within Lab 1, allowing for events to have associated execution times. In future this will allow for sensor input to be received within Lab 3, given that we will need to understand the sampling frequency of our received data which will require a timer within our code.

Lab Specific Additional Features

Define the major task functions you will need to add to Lab#_Tasks.h/c and what their purpose is. Describe how they will work together to accomplish the purpose of the lab.

Each proposed function should have all the inputs outputs and scope defined in a similar manner to the comments provided with the rest of the lab. For each function provide pseudo-code (steps to perform the task) as outlined for the scope. Make sure the pseudo-code is not language specific and is human-readable--it need (should) not compile and a middle-schooler should understand what is intended to happen.

We have added the pseudocode and its descriptions to our appendix at the bottom of this document. Within this lab we will be primarily working within a set of functions within Timing .h/.c. These functions will interface with the Time_t struct, which will use Timer 0 to give time feedback with multiple data types. The Timing .h/.c functions will handle initialization of Time_t and timer 0, as well as returning different interpretations and sections of Time_t. Additional functionality will also be added to Lab 2's MessageHandeling .h/.c, which will determine if actions related to a sent message should be done based on the message's timing information.

Planned Schedule and Task Assignment

Provide a specific plan for accomplishing the lab. Be specific, identify responsibilities for each party and when additional features need to be completed. Make sure to schedule in time for debugging!

This prelab and its pseudocode will be complete and submitted by Friday Feb 10th. Coding will begin the following Monday, Feb 12th. We will work digitally and as a pair, with Noah being the primary coder and Lucy giving input, getting function information and locating pins. The following Wednesday and Friday will be available to code further and debug, with opportunities for professor input after class on Tuesday and Thursday. Ideally the code will be finished Wednesday and the report finished by Friday Feb 17th. Saturday and Sunday will be left alone in case additional time is needed.

Pseudocode Appendix

Timing.h/c:

```
FUNCTION SetupTimer0();
    DESCRIPTION Timing.h/c Function SetupTimer0 initializes Timer0 to have a prescalar
    of XX and initializes the compare feature for use in an ISR. It also enables ISR's.
    INPUTS NONE
    RETURNS NONE

    // Timer register is TCCR0 and timer counter register is TCNT0
    INITIALIZE timer0 (TCCR0) with a prescaler
    INITIALIZE counter (TCNT0)

    // Output compare register A is OCR0A
    // Interrupt mask register is TIMSK0 set OCIE0A to 1 for output compare match
    ENABLE a compare interrupt
    ENABLE global interrupts

FUNCTION GetTime();
    DESCRIPTION This function gets the current time and returns it in a Time_t structure.
    INPUTS NONE
    RETURNS Current time as a Time_t structure

    GET the current value of the counter stored in TCNT0
    CONVERT the value into milliseconds
    CONVERT the value into microseconds

    INITIALIZE a Time_t variable
    SET the millisecond parameter in the Time_t variable to the tick value in milliseconds
    SET the microsecond parameter in the Time_t variable to the tick value in microseconds

    RETURN the Time_t variable
```

```
FUNCTION GetSTimeSec();
    DESCRIPTION This function gets the current time and returns it as a float.
    INPUTS NONE
    RETURNS Current Time

    CALL GetTime() to get the current time
    CONVERT the current time from GetTime() to seconds

    RETURN the current time in seconds


FUNCTION GetMilli();
    DESCRIPTION This function returns the millisecond parameter in the Time_t structure
    INPUTS NONE
    RETURNS current time in milliseconds

    CALL GetTime() to get the current time as a Time_t structure

    RETURN the current time in milliseconds from the Time_t structure


FUNCTION uint16_t GetMicro();
    DESCRIPTION This function returns the microsecond parameter in the Time_t structure
    INPUTS NONE
    RETURNS current time in microseconds

    CALL GetTime() to get the current time as a Time_t structure

    RETURN the current time in microseconds from the Time_t structure


FUNCTION Time_t SecondsSince( time_start_p );
    DESCRIPTION This function takes a start time and calculates the time since that time, it
    returns it in the Time_t structure.
    INPUTS start time as a pointer to a Time_t structure
    RETURNS seconds since input start time, stored as a Time_t struct

    INITIALIZE microsec_since TO
        (microsec OF timer MINUS microsec POINTED TO BY time_start_p)
    INITIALIZE uint32_t millisec_since TO
        (millisec OF timer MINUS millisec POINTED TO BY time_start_p)
    INITIALIZE Time_t time_since WITH millisec_since, microsec_since
    RETURN time_since
```

MEGN540_MessageHandeling.h/c:

```
FUNCTION MSG_FLAG_Execute(p_flag)
    DESCRIPTION Function MSG_FLAG_Execute indicates if the action associated with the
    message flag should be executed in the main loop both because its active and because
    its time.
    INPUTS pointer to a message flag
    RETURNS a boolean corresponding to if the action should be executed in the main loop

    IF the action is active AND it is time for the action to run
        RETURN true
    END IF

    RETURN false
```