Colorado School of Mines
Department of Mechanical Engineering
MEGN 540: Mechatronics

Laboratory 2: Timers and Timing

Deliverable: Report on time to complete a loop and other tasks.
Outcomes:

- Implement timer and prescalar to achieve desired resolution
- Initialize hardware timer and associated flags
- Use a interrupt service routine for handling comparison interrupts
- Understand time to send floats and run loops
- Implement state-machine based programming to control program flow.

Equipment:
· Lab Kit Mouse Keyboard Monitor Internet Connection

Background:
The atmega32u4 has for hardware based clocks built in. For this lab we will be focused on using Timer0. Timer 0 is an 8-bit timer with three interrupt options (overflow, compare A, and compare B). Your job will be to setup the Timer0 hardware to initiate an interrupt every millisecond for our timing needs. Once complete you'll be able to use this capability with the other required functions for providing timing information about the device. In addition to the below new functions, you'll also want to revisit some of the Message Handling functions from Lab1 to enable new functionalities.

Functions to add: (note the file can be found in MEGN540/c_lib):
void SetupTimer0( );
    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.
Time_t GetTime();
    Timing.h/c
    This function gets the current time and returns it in a Time_t structure.
float GetSTimeSec();
    Timing.h/c
    This function gets the current time and returns as a float.
uint16_t GetMilli();
    Timing.h/c
    These functions return the individual parts of the Time_t struct, useful if you only care about things on second or millisecond resolution.
uint16_t GetMicro();
    Timing.h/c

These functions return the individual parts of the Time_t struct, useful if you only care about things on second or millisecond resolution.

float SecondsSince(const Time_t* time_start_p);

Timing.h/c

This function takes a start time and calculates the time since that time, it returns it in the Time struct.

bool MSG_FLAG_Execute(MSG_FLAG_t* p_flag)

MEGN540_MessageHandeling.h/c

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.

Assignment:

You will enable timing and time-based interrupts for this assignment.

1. Review the provided functions, understand what they are doing, and how their called. Enable the functionality defined for the new functions.
2. Read the Timer0 documentation, look up with registers enable the timer, which control the action (internal or external), and how to setup the comparator. What prescalar will you need?
3. Enable the message handling for the new ‘t’ and ‘T’ messages to report on timing.
    a. How much time does one main-execution loop require?
4. It is important that you retain all of the Lab 1 functionality while adding the Lab 2 capabilities.
5. Implement a time-out flag on USB communications so that if the input USB buffer has an incomplete message (one that is not the appropriate size for parsing) for over 100ms, the input buffer gets flushed.

Deliverables:

1. Demo: Make a 1-2-Minute Video that talks through your code and how you completed the lab. Make sure to show me how you handled the functions added to Timing.h/c
2. Demo: Upload a video of you interacting with the car demoing each operation as well as error states (not sending a math operation).
    a. Make sure you show the car printing the current time once and every second.
    b. Make sure you show the car printing the time/loop once and every second.
    c. Make sure to demo at least one of the Lab 1 commands.
3. Lab Report: Prepare a summary of what you have completed for this assignment. Follow a typical lab report format. Make sure that you:
    a. Include the information that you found important and critical while completing this lab, and would be useful if you wanted to replicate it in the future.
    b. Indicate which pins on the electrical schematic were used in this project, and how you can tell which registers you needed to initialize and set. (Why Timer0 and not 1/2/3 for example?)
    c. Discuss how you might expand this capability later in the labs to help with future development.

Host -> Device Communication Messing Definition

	CMD char	CMD Hex	# Bytes	Format	Resulting Action
Lab 1	~	0x7e	1	c	Reset the program (return 0)
	*	0x2a	9	cff	Multiply the two floats and return the product.
	/	0x2f	9	cff	Divide the first float by the second and return the result.
	+	0x2b	9	cff	Sum the floats and return the result.
	-	0x2d	9	cff	Subtract the second from the first and return the result.
	UNDEF				If the command char is unrecognized, flush the input buffer, and respond with a '?' (0x3f) character
Lab 2	t	0x54	2	cc	Return the time it requested followed by the time to complete the action specified by the second input char. 0x00 -> Time Now 0x01 -> Time to complete a full loop iteration Others as you define
	T	0x74	6	ccf	Return the time it requested followed by the time to complete the action specified by the second input char and returns the time every X milliseconds. If the time is zero or negative it cancels the request.

Device -> Host Communication Messaging Definition
[MSG Length] [Format C-Str][Host Initiating CMD Char][DATA]
Time It Example (time it 0x01): [10] [ccf0x00] [T][0x01] [1.3] Multiply Example: [8] [cf0x00] [*] [-.239] Undefined Command Example (>) : [5] [cc0x00] [>][?] Acceleration Example: [18] [cfff0x00] [A][0.001 -0.09 9.89]