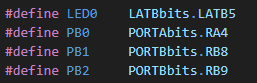
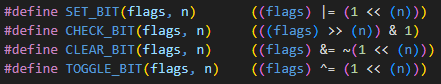
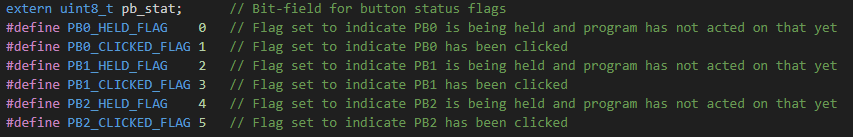
ENCM 511 Assignment 3

Preamble

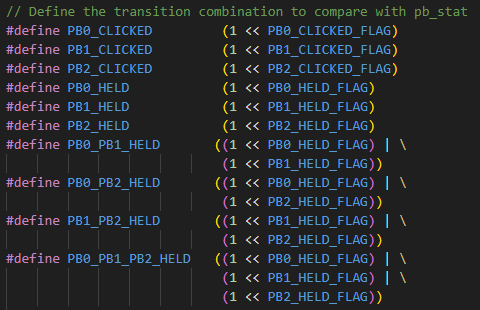
**common.h** contans the following which have been defined to be used throughout the program:

The LED and buttons have been abbreviated as shown.

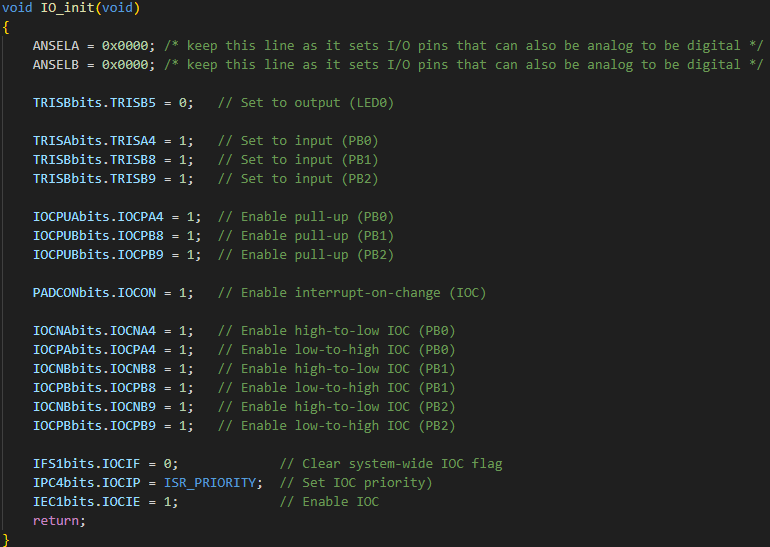
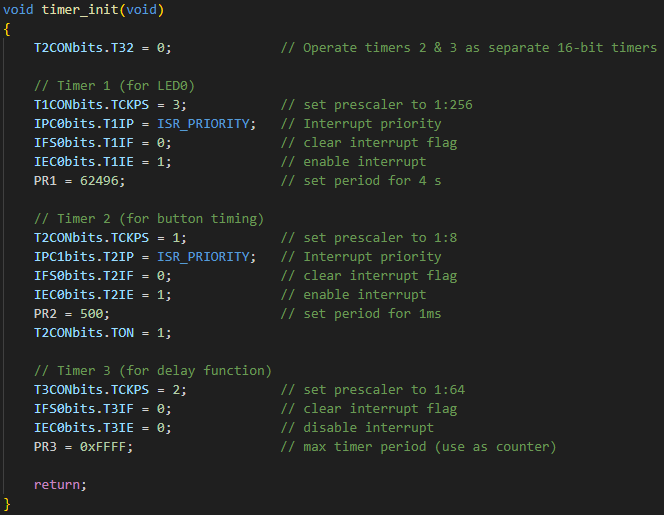
The following bit masks have been defined to allow for enhanced readablity. Several variables are used throughout the program to hold multiple flags (bit-fields). These defines allow those flgas to be accessed easily.

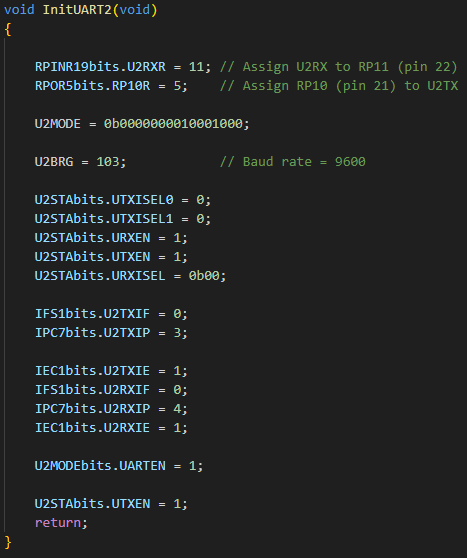
This is one such bit field. Purpose of each flag elucidated by its comment.

Here is an example of this bitfield being updated. In this case the PB0\_HELD\_FLAG of pb\_stat is set to 1.

The following have been created to simplify checking the flags in pb\_stat for commonly used combinations.

Peripherals

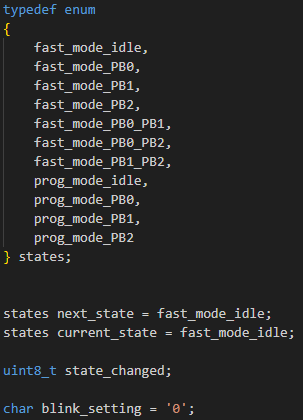
**init\_functions.c**  contains the initailization functions for the IO pins and the timers. Line functionality elucidated by comments. ISR\_PRIORITY is used to prevent ISRs from pre-empting each other.

**uart.c**  contains the initilization for UART2 which is taken from the provided lab files.

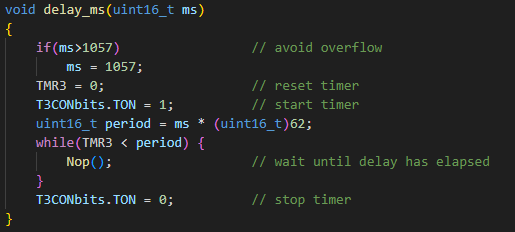
Datatypes and Initialization

The pb\_stat bit-field is initialized to 0.

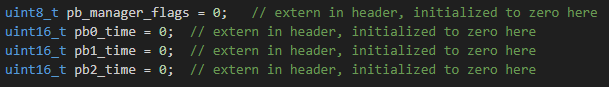
The states enum is used to control the state machine. Initial and next state are initialized to fast\_mode\_idel. The state\_changed variable is used to indicate when the machine has changed states. The blink setting controls the speed of LED blinking in certain states and is initialized to a char 0.



Key Functions

The delay\_ms function is held in the **delay.c** file and is used to for the debouncing of button inputs.

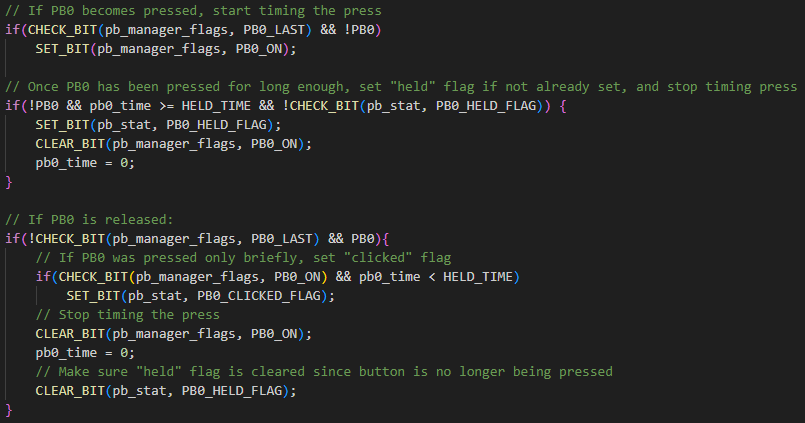
The **buttons.c** file contains the buttons\_update function. The function uses a bitfield pb\_manager\_flags as well as three unsigned 16 bit integers to record the number of milliseconds each button has been held for.

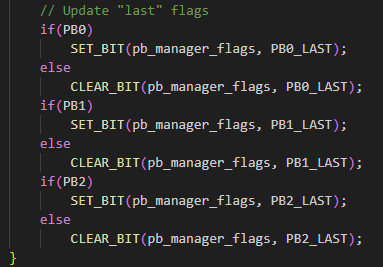


The buttons\_update function is called periodically (every 1ms) by main when the timer 2 ISR occurs or whenever the button ISR occurs.

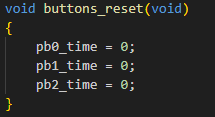
Initially it debounces the buttons.



Next the program evaluates the condition of PB0. This logic is then run for PB1 and PB2.

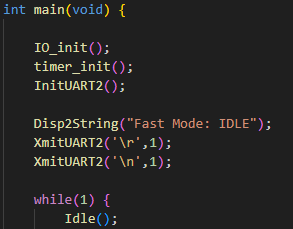
After all buttons have been checked their states are updated to be the new “last” flags.

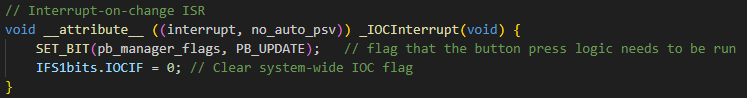
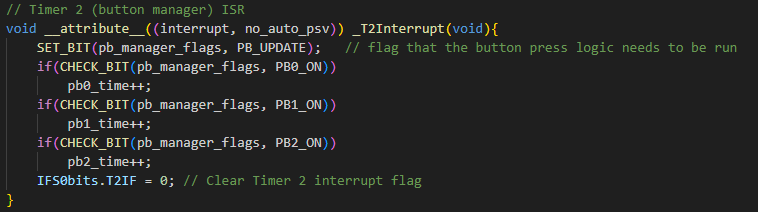
The buttons\_reset function is used to reset the millisecond counter of each button.

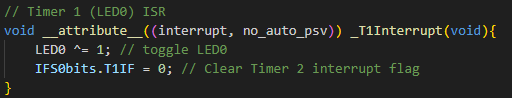


Operation

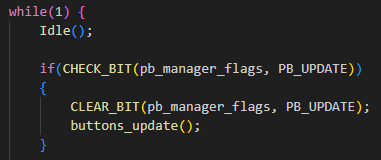
The code begins by calling all three initialization functions and printing the initial state to the terminal before entering its while loop and going into idle.



Assuming this is the first run of the program, i.e. the program is in a state where the LED blink timer is not running. The Idle will be interrupted by either a button press or the button manager ISR timer triggering.

If the program is in a state with the LED timer running there may also be an interrupt generated to toggle the LED.

If one of the ISRs set the PB\_UPDATE flag then main will call buttons\_update to check on the status of the buttons.



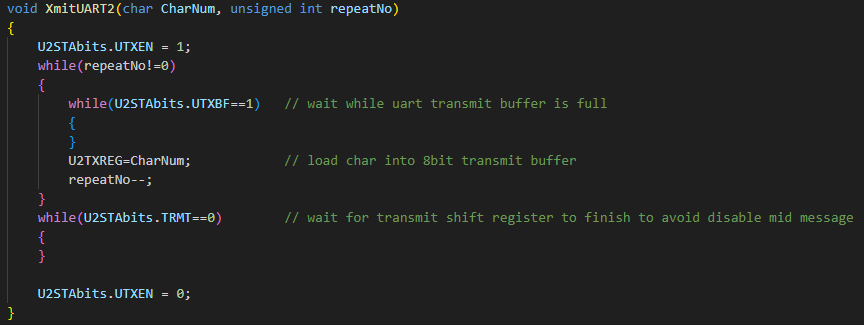
Answers to Lab Questions

*In part 1 (the lab portion), what was the target baud rates? What is the % error of the actual baud rates?*

The target baud rate was the UART standard 9600. The actual baud rate (with U2BRG = 103 and BRGH = 1) was:

The percent error is then:

*What are your comments/explanations for XmitUART2?*

**

*In your assignment part, how do you get the LED to blink while waiting for a character to be input? Is the character receive function a “busy wait,” i.e., does it stop/block the CPU from doing something else?*

The character receive function only attempts to read a character when the interrupt occurs that there is data in the Rx buffer, therefore it is not a busy wait. The toggle LED ISR can interrupt the while loop of waiting for a character input. The toggle ISR is fast enough and does not trigger often enough that it would cause the Rx buffer to overflow.