Project #2
EGR 424 - The Design of Microcontroller Applications
Instructor: Dr. Chirag Parikh
Renzo Garza Motta
November 7th, 2022

Objectives

The objective of this project is to develop the following skills:

- Reading and interpreting technical documentation and sample code
- Writing well-structured, low-level device drivers
- Development and debugging of interrupt-driven code on an actual hardware system

Project Requirements

- Implement, debug, and document device drivers.
- The Application must use device drivers
- User interrupts to ensure that there are no busy-wait/polling loops
- All interrupt service routines must assume that there will be other interrupt service routines in the system and must therefore “play nice.”

Function

This project aims to showcase low-level driver development for hardware for the MSP432P401R. Specifically using the HM-10 bluetooth module and UART communication.

The way this project showcases the above applications is by implementing a Snake Game. This is implemented by having two MSP432s connected HM-10 such that there is a Master-Slave connection between two HM-10s. The master MSP432 is connected to its respective HM-10 (via UART) which connects to the slave HM-10 and displays the game on a 1.9” ST7735 TFT display. The Master MSP handles all the logic for the game, while the slave handles user input as a remote controller.

Additionally, the Slave MSP432 becomes available to the Master HM-10 and once connected all data transmitted to the HM-10 is send and received by the MSP via UART. The slave is also tasked with handling user input via interrupts to determine the direction of the snake displayed by the Master. Such directions can include U, D, L, and R for Up, Down, Left, and Right respectively.

To showcase two-way communication, as the slave sends the user input, when the game changes, such that points are gained or the game is over, the master then sends these changes to the slave which are showcased by the use of a green and red LED (Green for points and red for game over).

Additionally implementing PWM signals, and complimenting the LEDs by including different vibrations of the motor dependent on the game status.

A button was implemented to provide the user with a manual way to attempt a bluetooth connection between the master and slave in the case that the connection is not established upon bootup.
Lastly, the Watch-Dog Timer (WDT) was used to induce a reset of the system in the case of a game-over state. The master, before resetting commands the slave to reset itself to re-establish a connection for proper function during the game.

Summary

The master uses the snake.h library and the UART.h library as the drivers for this project. These libraries still require initialization of the watchdog timer, random number generator seed, initialization of the UART module, and the connection of the Bluetooth modules.

Once the infinite while-loop is entered, the gameLogic() function is called through each loop.

```
while(1){
        gameLogic();
        if(recon){  //Check if the reconnect button has been pressed
            connectModule();    //Attempt to reconnect Master and Slave
            recon = 0;  //Reset flag
        }
        wdt_reset();    //Reset watchdog timer to avoid accidental reset
    }
```

While the master MSP is constantly on the lookout for its RX buffer to be full, the logic consistently updates the location of the player on the board, stores previous locations, checks if apples were eaten and updates the score accordingly.

Once there is data on the buffer, which means that there was user input on the slave, the respective commands are interpreted and the game updated accordingly.

For the slave MSP, upon initialization of the four U, D, L, and R buttons, the infinite while-loop is entered waiting for interrupts to be flagged via the ISR, and handled.

Such handling results in flags respective to their intended action such as the snake movement in the game. These commands are send via UART to the master MSP and decoded accordingly. A format of “Both NL & CR” is followed as this is the format required by the HM-10.

Additionally, the slave is on the lookout for incoming data on it respective RX buffer. When the buffer is full, this is interpreted accordingly as to “celebrate” gaining points via motor vibration using PWM with the MSP’s Timer A capabilities and LED blinks.

```
while(1){
           if(sendR){
```

```
            UART1_strOut("R\r\n\0");
            sendR = 0;
        }else if(sendL){
            UART1_strOut("L\r\n\0");
         sendL = 0;
    }else if(sendU){
        UART1_strOut("U\r\n\0");
        sendU = 0;
    }else if(sendD){
        UART1_strOut("D\r\n\0");
        sendD = 0;
    }
       if((cmd == 'A') || (cmd == 'a')){
            //Controller Celebration Feedback
            vibrateMotor(3, GREEN_LED);
            cmd = 0;
       }
       if((cmd == 'G') || (cmd == 'g')){
            //Controller Game Over Feedback
         vibrateMotor(7, RED_LED);
         __delay_cycles(48000 * 7000);
         WDT_A->CTL = WDT_A_CTL_CNTCL;
         cmd = 0;
    }
       TIMER_A0->CCR[1] = 0; // reset register
       wdt_reset();
   }
```

For fully commented code, and library implementation, refer to the source code provided with this project.