CMPE 121L: Microprocessor System Design Lab
Spring 2016
Lab Project: *Reversi* Game

In this project, you will use the PSoC-5 microcontroller, the 16x32 RGB LED display, a WiFi wireless module, and an SD card interface to design an Internet-connected two-player game called *Reversi* (also known as *Othello*). You will do the design independently, but will need to work with another student to test and demonstrate your design.

The Game

Reversi is a two-player board game played on an 8x8 board with discs that are black on one side and white on the other. Each of the players chooses a color and the player with the maximum number of discs of his/her color at the end of the game wins. In each move, the goal of the player is to turn as many discs of the opposite color to his/her color. A move consists of adding a single disc so that one or more of the opposing player's discs lie in a straight line (row, column or diagonal) between two of the discs of the player making the move. The opponent's disks that are "flanked" by the two discs can then be flipped to the color of the player making the move.

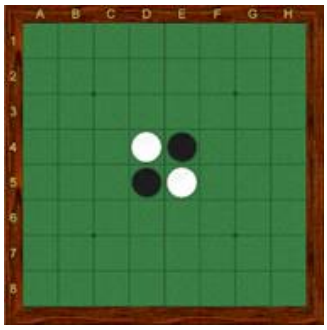
The rules of the game are explained here. The game ends when no empty square remains or when neither of the players is able to place a new disc to flank any of the opponent's discs.

Game Display

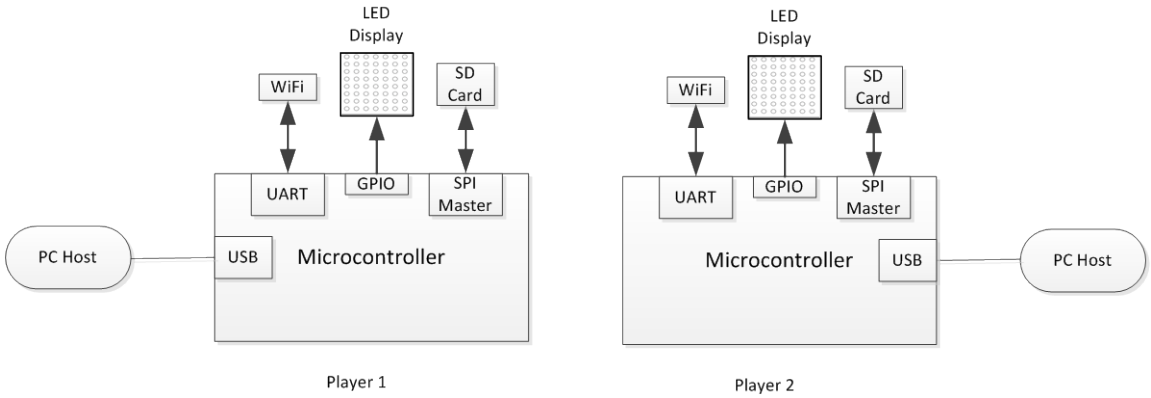
We will use two colors of LEDs, red to indicate the black side of the disc and blue to indicate the white side. Your implementation must support three board sizes: 4 x 4, 8 x 8, and 16 x 16. The board size must be a parameter in your code so that you can change the size by modifying only the parameter definition and recompiling the code. The 4 x 4 size is useful for testing the termination of the game, as it may take many moves to complete a game on the 8 x 8 board. The rows and columns will be numbered from 1 through 16.

The starting configuration of the game is shown below. The first move is made the player with the lower IP address (obtained from the WiFi card, explained in the document

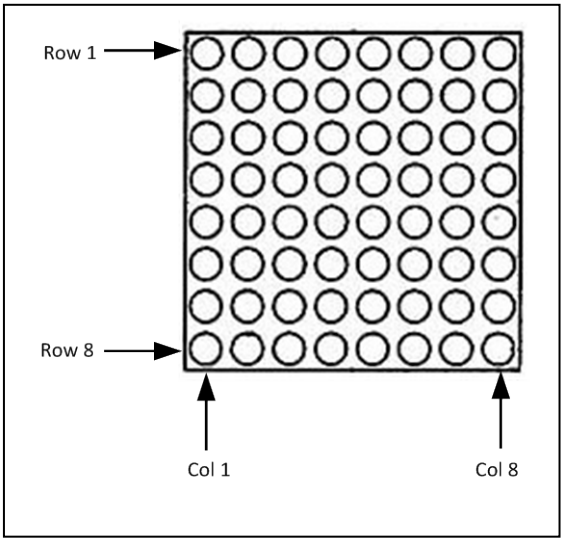
describing the wireless card). The player making the first move always selects black as his/her color, so the first move consists of adding a black disc to the board.



System Design



A block diagram of the system is shown above. The LED display is driven by the GPIO pins of the microcontroller and the WiFi module is connected to a UART in the microcontroller. The SD card is interfaced to the microcontroller through a SPI master block. User input is from the keyboard on the host PC.



The numbering convention for row and columns is shown above (Row 1 is the top row and column 1 is the leftmost column). You should use this numbering convention in all player-to-player communication. These numbers do not have to correspond to the row and column numbers in the RGB display (for example, the rows of the RGB display may form the columns of the Connect Four game if the display is mounted with its rows in a vertical orientation.).

When the game is being played, each player makes moves using the keys on his/her microcontroller board. At any time during the game, the displays of the two players should be synchronized to show the same state of the discs. This is accomplished by the microcontroller of each player communicating each of his/her moves to the other microcontroller. That is, when Player 1 makes a move, his/her local display is updated to show the newly added disc and any color changes to the existing discs as a result of the move. The move is also communicated to Player 2's system, which then updates its display to show the new state of the game.

The microcontroller must check the legality of each move, and should also check for terminating conditions.

Hardware Design

You should start with hardware schematics of the entire system. You may design this using Eagle. Hand-drawn schematics are acceptable, but must be detailed enough for someone else to wire up the system using these alone. Do not start wiring any components without hardwire schematics of the system. Designing the schematics requires a number of sub-tasks:

1. Determine a good physical placement of the components. Do not connect components using hanging wires.
2. Identify the GPIO pins to be used to connect to the various components.
3. Determine the power supply voltages of each component. The SD card has 3.3V power. If the PSoC-5 is configured for 5V I/O, voltage dividers must be used on the output pins of the PSoC driving SD card inputs, so that the voltage does not exceed 3.3 on the input pin when driven high. The LED display used 5V power and can tolerate 5V at its inputs. The WiFi card can be powered from either a 5V or 3.3V supply, but just like the SD card, its inputs can only tolerate a maximum of 3.3V.

Interfacing the LED Display

Refer to the Lab Exercise 5 handout for details of the display panel. For this project, you need to use all the 16 rows of the panel and 16 columns, but you only need to display two distinct colors. You may use any of the three methods to refresh the display: using interrupts, DMA, or a dedicated hardware controller. The only requirement is that the display should be refreshed fast enough to avoid flicker.

User Input

The user makes a move by selecting the corresponding row and column. The tentative position of the current move is indicated by a flashing LED, which acts as a cursor. The user specifies the moves by pressing one of the following keys on the host PC's keyboard:

- Left arrow: Move the cursor to the left by one column
- Right arrow: Move to the right by one column
- Up Arrow: Move up by one row (to a lower numbered row)
- Down arrow: Move down by one row (to a higher numbered row)
- Home: Move the disc to the top row and leftmost column.
- Return key: Execute the move and send it to the remote player.
- Typing the letter S, followed by Return, indicates that user wants to pass let the other player make the next move.

You need to use the USB-UART design from Lab 4 to receive keyboard data from the PC.

SD Card

The SD card is used to log the player moves, so that the game can be reconstructed later. The software should log the moves made by both players into a file in the order in which they were made. Each move is to be recorded in the following format:

Player -ID pass row-number column-number

The player ID is a character string that identifies the player who initiated the move. The row and column numbers correspond to the position of the newly added disc. The "pass" field is normally 0. A non-zero value indicates that the corresponding player skipped a move. The fields must be separated by spaces or tabs, and each record must end with a newline.

Documentation on the MicroSD breakout board can be found at:

https://www.sparkfun.com/products/544?

This page has links to a number of useful design resources, including the SD card specification. You need to use the emFile API functions to write to the card (Refer to the Form posting for instructions to use emFile).

Using the WiFi Modules

The WiFi module is the CC3200 Launchpad from Texas Instruments. This chip has its own ARM 32-bit microcontroller, but you are not required to design any software running on it. You will be provided the necessary software to transmit and receive data on the WiFi network. You will transmit and receive data to/from the CC3200 using a UART in the PSoC-5.

You need to connect only 4 of the pins of this board: Ground, 5V power, UART0_TX and UART0_RX.

	P1		P3	
Ref	Signal	Dev Pin#	Dev Pin#	Signal
1	3.3V			5V
2	ADC_CH1	58		GND
3	UART0_RX	4	57	ADC_CH0
4	UART0_TX	3	60	ADC_CH3
5	GPIO	61	58*	ADC_CH1
6	ADC_CH2	59	59*	ADC_CH2
7	SPI_CLK	5	63	AUD_SYNC
8	GPIO	62	53	AUD_CLK
9	I2C_SCL	1	64	AUD_DOUT
10	I2C_SDA	2	50	AUD_DIN

P4		P2		
Signal	Dev Pin#	Dev Pin#	Signal	Ref
PWM	2*		GND	1
PWM	1*	18	GPIO	2
PWM	17*	8	SPI_CS	3
PWM	64*	45	GPIO	4
CCAP/GPIO	21*		RESET_OUT	5
CCAP/GPIO	18*	7	SPI_DOUT	6
GPIO	62*	6	SPI_DIN	7
GPIO	60*	21	GPIO	8
GPIO	16	55	GPIO	9
GPIO	17	15	GPIO	10

Note: The locations of the UART RX and TX pins in the TI documentation were found to be incorrect. TX is on the pin marked P55 and RX is on P57.

Details of the API functions and protocols necessary to use the WiFi card will be provided in a separate document.

Debug Interface

You should use the USB-UART example of Lab Exercise 4 to provide a debug interface for your system. This will come in handy during software development, as well as for displaying various status messages during game play.

Assembly and Testing

The following sequence of steps is recommended for a smooth bring-up of the project.

1. Create detailed schematics of the design.
2. Determine a good placement for all the components on the PCB.
3. Solder components on the PCB whenever possible.

4. Do not start by populating all the components. Instead, start with the display first, and populate only the components needed to make the display functional. Write a test program to display various patterns and make sure that you are able to activate all the necessary LEDs in the matrix.
5. Test the operation of the USB-UART and make sure you can receive characters from the host PC keyboard.
6. Next, configure the UART in the PSoC-5 and test its operation. Connect your UART to someone else's and check if you can transmit and receive data.
7. Add the SD card and verify that you can write to it using emFile functions.
8. Once all the components have been tested, you can write the complete program and debug it. You should test the system by connecting the UARTs of two systems together by a pair of wires (don't forget the common ground), and debugging the game play before attempting to use the WiFi module.
9. Once the game has been fully debugged you can incorporate the WiFi module in your design by enhancing the functions used to transmit and receive data.

Testing with a UART-to-UART Link

Before connecting the WiFi modules, you may test the entire game play through a UART-to-UART serial link. The serial wires can then be replaced with the WiFi link later, with only minor changes to the communication protocols. To make the designs interoperable, you must follow common data formats and communication protocols across the serial link.

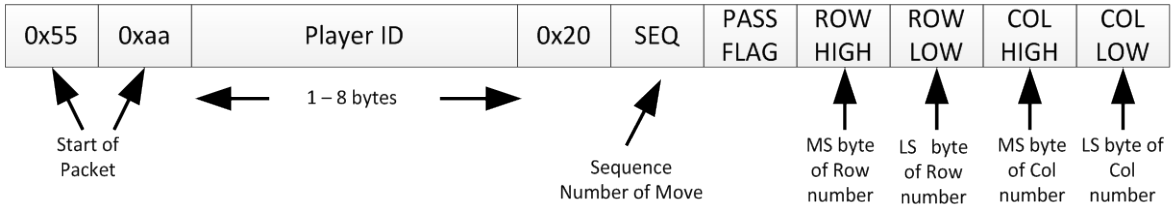
The UARTs must be configured with the following parameter settings:

- Full duplex, 8-bit data, no parity, 1 stop bit, no hardware flow control.
- Speed = 115200 baud

You may configure other parameters such as FIFO sizes and interrupts based on the requirements of your implementation.

Data Format

The information transmitted on the serial link must be formatted into packets. Each packet contains the information necessary to communicate a move in the game to the other player. The format of the packet is shown below:



The special pattern 0x55aa identifies the start of the packet. Player ID is an ASCII string of up to 8 characters identifying the player sending the move. The row and column numbers must be encoded in ASCII using two characters. For example, row number 1 is encoded as 0x30 0x31, and row number 15 is sent as 0x31, 0x35.

The sequence number field is an 8-bit number to identify the move, encoded in ASCII using three characters (for example, a sequence number value of 5 is transmitted as 0x30, 0x30, 0x35. A sequence number of 255 is transmitted as 0x32, 0x35, 0x35.). It is incremented by one (modulo 256) for every move initiated by the local player. This helps the remote player to distinguish new moves from duplicates. The pass flag is normally set to 0 ASCII (0x30 in hex). Any other value indicates that the player is unable to find a position that will allow him/her to flip the color of at least one disc, and wants to pass.

The sender must transmit a packet containing the most recent move by the local player approximately once every 500 milliseconds. The receiver must ignore any duplicate moves or packets with errors (bad player ID, illegal row/column numbers, etc.).

What to Submit?

- Schematics of your design.
- PSoC Creator files.
- Project report.

You should maintain detailed notes during the design and testing of your project to help you write the documentation.

We will discuss the organization of the report later in class.