Summer 2016

Lab Report 3

Xinxin Zhang, Jiaqi Zhang, Heng Zhou

Introduction

In this lab, we learned how to use 3 GPIO lines on Beaglebone Black to control the LCD display using a shift register. The shift register we used is 74HC595. We also wrote a kernel mode driver (LKM) which allows us to write software which mimics the driver.

Shift Register

We used a shift register to eliminate the use of GPIO pins. This way we are able to free more GPIO pins. The 74HC595 shift register has an 8 bit storage register and an 8 bit shift register. Data is written to the shift register serially, then latched onto the storage register. The storage register then controls 8 output lines. The figure below shows the 74HC595 pinout.

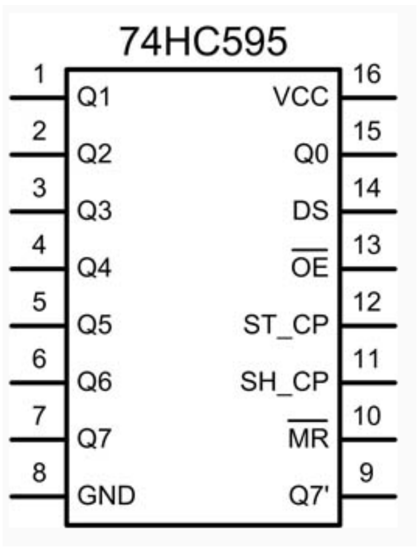
Pin 14 (SER) is the Data pin.

Pin 11 (SRCLK) is the serial in clock, when it goes from Low to High the value of DS is stored into the shift register and the existing values of the register are shifted to make room for the new bit.

Pin 12 (RCLK) is the serious out clock. It is held low whilst data is being written to the shift register. When it goes High the values of the shift register are latched to the storage register which are then outputted to pins Q0-Q7.

In our lab, we only used three GPIO outlines to control the shift register.

Vcc --- Sys 5V;
SER --- GPIO_67;
OE --- GND (Ground);
SRCLK --- GPIO_68;
RCLK --- GPIO_69;
MR --- 5V.
QH is left unused.



Kernel Mode

We wrote a kernel mode driver (LKM) which allows us to write software that uses the shift register from the kernel. This part is down by game_drive.c and game_driver.h. We are interfacing with an LCD screen using the Phytec board. We used the kernel to let the LCD board interacting with BBB's hardware.

Several libraries are included.
`<linux/gpio.h>` includes functions that we can use to directly call functionalities of gpios on the BBB board without using any dev files.

`<linux/delay.h>` includes functions that allow us to delay within the kernel modules.

By using `msleep();`, we are able to delay the procession time of the embedded system in milliseconds.

After compiling the game_drive.c file successfully, we generated the game_driver.ko file, which can be inserted directly to the kernel. Then we submit the file to BBB and install the module. After the installation, the cursor starts to blink on the left top corner of the LCD board.

The command line "`mknod /dev/lcd c <MAJORNUM> <MINORNUM>`" is used to enable us to interact with the module by writing characters to it. the Major number for our module is 239 and the minor number is 0.

We wrote several functions that enables us to write texts on the LCD board, and called the function in the 'write' funciton. By calling write(string) in the user space file (test_game.c) we were able to pass the string that we want to write to the board to the kernel file and then write it onto the board.

Game Description

To discover the basic functionality of the LCD board, we wrote a program that plays the Hangman game on the LCD board. All game descriptions are shown on LCD. (Descriptions are also shown on terminal for convenience.)

At the beginning, game asks user to choose a word. User should input a word with length less than 32 from the terminal, case does not matter. We used ‘`gets()`’ to read the line of word from the terminal and stored the word in a character array.

```
char letter = 0; // Store the word of choice.
printf(&quot;Please choose a word: \n&quot;);
char secretWord[50];
gets(secretWord);
```

The number of chances for guessing the word is set to 5 as default. Then user (possibly another player who does not know the word stored.) is asked to guess the word.

```
int totalCh = (unsigned)strlen(secretWord);
totallength = totalCh;
int letterFound[100] = {0}; // 0: Not guessed character. 1:Guessed
character.
int leftTimes = 5; // Chances left.
int m = 0;
printf("\nThe length of the word is%d", totallength);
unsigned char table10[100] = &quot;What is the word?&quot;;
gameInfo(table10);
printf("\nWhat is the word? \n");
```

We used “_” as place holders for the characters not guessed. For example, if the input word is “apple”, we would display “_ _ _ _ _” at the beginning of the game. Player guesses a single character each time. The input case of char dose not matter. If the char is in the word, the “_” would be replaced with the char, green light is turned on. If user inputs “p”, the LCD will display “_ PP_ _”. If the char is not in the word, display does not change, player loses one chance, red light turns on. We used an int array to keep track of the status of the word: 0 for not guessed, 1for guessed. The previous case would give this array: {0, 1, 1, 0, 0}. We used another char array for the display, “_” for 0, and the char for 1. Player is told the left number of chances every time he puts in a char.

```
    while(leftTimes > 0 && !win(letterFound)) {

            unsigned char table3[100] = "You have ";
            char strLeftTime[1];
            sprintf(strLeftTime, "%d", leftTimes);
            strcat(table3, strLeftTime);
            strcat(table3, " chances left.");
            strcpy(write_buf, table3);
            printf("\nYou have %d chances left. \n", leftTimes);
            write(fd, write_buf, sizeof(write_buf));

            unsigned char current[100] = "";

        for(m = 0; m < totallength; m++) {
            if (letterFound[m]) { // If the mth char is guessed.
                char secret[1];
                sprintf(secret, "%c", secretWord[m]);
                strcat(current, secret);
                printf("%c", secretWord[m]); // Show the char on screen
            } else {
                strcat(current, "_");
                printf("_"); // mth character not guessed:hold the place
with "_"._
            }
        }

            strcpy(write_buf, current);
            write(fd, write_buf, sizeof(write_buf)); // Shows the current
status of the game (e.g ___le <- apple)

            printf("\nPlease input a character: ");
            strcpy(write_buf, "Please input a character: ");
            write(fd, write_buf, sizeof(write_buf));

        letter = readCharacter();

        // Checks whether the gussing char is in the word or not.
        // Green light on for right, red light on for wrong.
        if (!researchLetter(letter, secretWord, letterFound)) {
                  leftTimes--; // Chances decrease by 1
                  strcpy(write_buf, "Try again!");
                  printf("Try again! \n"); // Guesses incorrectly
                  write(fd, write_buf, sizeof(write_buf));
                  green(0);
                  red(1);
        } else {
                  strcpy(write_buf, "Genius!");
                  printf("Genius! \n");   // Guesses correctly
                  write(fd, write_buf, sizeof(write_buf));
                  green(1);
                  red(0);
            }
      }
```

Determines if the user win or lose the game. Both green and red light on for winning and both lights are turned off for losing the game. Different music are played for winning and losing.

```
      if (win(letterFound)) {
            unsigned char table6[100] = "You win! The word is :";
            strcat(table6, secretWord);
            strcpy(write_buf, table6);
            write(fd, write_buf, sizeof(write_buf));
            green(1);
                  red(1);
          int win[] = {noteC5, noteD5, noteE5, noteF5};
          int winDuration[] = {2, 1, 1, 1};
          int i;
          while(1){
                for(i=0;i < 4; i++) {
                  int length = winDuration[i];
                  playNote(noteperiod, noteduty, win[i], win[i]/3);
                  usleep(250000*length);
                }
          }
      } else {
            unsigned char table7[100] = "You lost! The word is :";
            strcat(table7, secretWord);
            strcpy(write_buf, table7);
            write(fd, write_buf, sizeof(write_buf));
       }
          green(0);
          red(0);
          int lose[] = {noteF5, noteE5, noteD5, noteC5};
          int loseDuration[] = {2, 2, 2, 2};
          int i;
          while(1){
                for(i=0;i < 4; i++) {
                  int length = loseDuration[i];
                        playNote(noteperiod, noteduty, lose[i], lose[i]/3);
                        usleep(250000*length);
                }
         }
      }
    close(fd);
    return 0;
}
```