

Xuecong Fan A01972388
Robert McKay A02080393
October 24, 2017

Lab 4

Introduction:

In this lab, we would learned how to implement an interrupt service routine(ISR) in C and we also need familiar PS/2 protocol for keyboard. Interrupts and configuration are the important part for this lab.

Accomplish:

In this lab our program accomplished the following requirements:

1. The ISR must be triggered for each clock tick.
2. The keystrokes must be stored in memory.
3. Our lab have a button that starts/stops the keylogger. Once the keylogger has been stopped, the captured keystrokes should be sent to the computer via UART0 and USB.
4. Our lab make use of interrupts to interface with the button. The button should have a higher priority than the clock interrupt.
5. The program only needs to be able to capture and display lowercase alphanumeric characters and space.

Hardware Design:

Cut the PS/2 extension cable in half (DO NOT CUT THE KEYBOARD CABLE!). Now peel back the outer insulator exposing the 5 wires on the inside on both sides of the extender. Strip back the insulators on the inside wires exposing the wire. Try not strip them longer then need to make proper connection in a bread board. Then we solder splice wire back together.

Use the logic analyzer and multimeter to identify the wires. First find out which wire is power and which is ground using the multimeter. Next connect the other wires to the logic analyzer. The remaining wires will be n/c, Clock, and Data. Then connect these to the board. Make sure Tiva C uses 3.3V logic.

Design Software:

Figure 1,2 are the code for this lab.

```

1  volatile int PA_DATA_R __attribute__((at(0x40004000))); //Initialize base address of port A
2  volatile int PE_DATA_R __attribute__((at(0x40024000))); //Initialize base address of port E
3  volatile int UART0_R __attribute__((at(0x4000C000))); //Make UART0 base volatile
4  volatile unsigned char ps2data[300];
5  volatile unsigned int ps2dataIndex = 0;
6  volatile unsigned char asciiData[300];
7  volatile unsigned int i = 0;
8  volatile unsigned int j = 0;
9  volatile unsigned int interruptCounter = 0;
10 volatile unsigned char bitholder[11];
11 unsigned char *systemClock = (unsigned char *) 0x400FE000;
12 unsigned char *portA = (unsigned char *) 0x40004000;
13 unsigned char *portE = (unsigned char *) 0x40024000;
14 unsigned int *uart0 = (unsigned int *) 0x4000C000;
15 unsigned int *corePeripheral = (unsigned int *) 0xE000E000;
16 // This is a look up table for basic key strokes.
17 // This translates from ps/2 to ascii.
18 // The table starts at 0x15, so you should shift
19 // the table by 0x15 when you reference it.
20 //
21 unsigned char ps2_to_ascii[] = {
22     'q','l',0x00,0x00,0x00,'z','s','a','w','2',0x00,
23     0x00,'c','x','d','e','4','3',0x00,0x00,' ','v','f','t','r','5',0x00,0x00,
24     'n','b','h','g','y','6',0x00,0x00,0x00,'m','j','u','7','8',0x00,
25     0x00,' ','k','i','o','0','9',0x00,0x00,'.',0x00,'1',0x00,'p'};
26
27 int main(void)
28 {
29     //INITIALIZATIONcaa
30     systemClock[0x618] = 0x1; //turn on UART0
31     systemClock[0x608] = 0x11; //turn on ports A and E
32     portE[0x420] = 0x0; //disable alternate functions for port E
33     portA[0x420] = 0x3; //enables alternate functions on PA0 and PA1, disables them for the ot
34     portA[0x400] = 0x10; //configures PA1, the transmit, to be output. Others are input.
35     portE[0x400] = 0x0; //configures port E to be input
36     portE[0x514] = 0x10; //sets pull down resistor for PE1, will be the dump button
37     portA[0x51C] |= 0xF; //digital enable for pins in A
38     portE[0x51C] |= 0x2; //digital enable for pins in E
39     portA[0x52C] |= 0x11; //configures UART0 alternate function on PA0 and PA1
40     uart0[0x30/4] = 0x0; //disables the UART before any of the control registers are reprogram
41     uart0[0x24/4] = 0x68; //Writes 104 to the integer part of UART0, for the baud rate divisor
42     uart0[0x28/4] = 0xB; //Writes 1/6 to the fractional baud rate divisor register of UART0
43     uart0[0x2C/4] = 0x70; //Tells the UARTLCRH (line control) we want 8 bits of data, and 1 st
44     uart0[0xFC8/4] = 0x0; //Tells the UART communication clock to use the system clock
45     uart0[0x30/4] = 0x101; //enable UART0 with Transmit
46     corePeripheral[0x100/4] = 0x11; //0x100 is the offset for the interrupt enable, bit 0 corr

```

Figure 1: code part 1

```

47   portA[0x404] = 0;
48   portA[0x408] = 0;
49   portA[0x40C] = 0;
50   portA[0x410] = 0x4; //Enables interrupt on PA2 for keyboard clock
51   portE[0x404] = 0;
52   portE[0x408] = 0;
53   portE[0x40C] = 0;
54   portE[0x410] = 0x2; //enables interrupt on PE1 for dump button
55   while(1) {
56   }
57 }
58
59 void GPIOE_Handler(void) {
60   portE[0x41C] = 0x2; //the interrupt has been serviced
61   for (i = 0; i < ps2dataIndex; i++) {
62     uart0[0x0] = asciiData[i];
63     for (j = 0; j < 5000; j++) {
64       //wait
65     }
66   }
67 }
68
69 void GPIOA_Handler(void) {
70   portA[0x41C] = 0x4; //the interrupt has been serviced
71   if (interruptCounter == 0) {
72     for(i = 0; i < 11; i++) {
73       bitholder[i] = 0;
74     }
75   }
76   bitholder[interruptCounter] = portA[0x20] >> 3; //PA3 corresponds to 00 10 00 (00) (last t
77   if (interruptCounter != 0 || interruptCounter < 9) { //We don't want to save it if we're g
78     ps2data[ps2dataIndex] |= bitholder[interruptCounter] << (interruptCounter - 1); //Left s
79   }
80   interruptCounter++;
81   if (interruptCounter == 33) { //increments the index after every key press and release
82     interruptCounter = 0;
83     asciiData[ps2dataIndex] = ps2_to_ascii[ps2data[ps2dataIndex]-0x15]; //Uses the conversion
84     ps2dataIndex++;
85   }
86 }
87

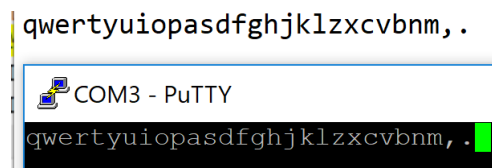
```

Figure 2: code part 2

Result:

Description: You can type any letters in keyboard on your screen then open Putty it will automatically type same letters which you typed in your screen.

Figure 3 shows that I typed "qwertyuiopasdfghjklzxcvbnm,." on keyboard.



qwertyuiopasdfghjklzxcvbnm,.

COM3 - PuTTY

qwertyuiopasdfghjklzxcvbnm,.

Figure 3: PUTTY

Detail for "a":

If we debug the code and pressed "a". Figure 4 shows the data when "a" pressed. Figure 5 shows the memory when "a" pressed

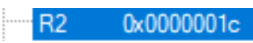


Figure 4: "a" pressed

Name	Value	Type
ps2data	0x20000064 ps2data[] ""	unsigned char[300]
[0]	0x1C	unsigned char

Figure 5: "a" memory

Figure 6 is the screen shot for we typed in letter.

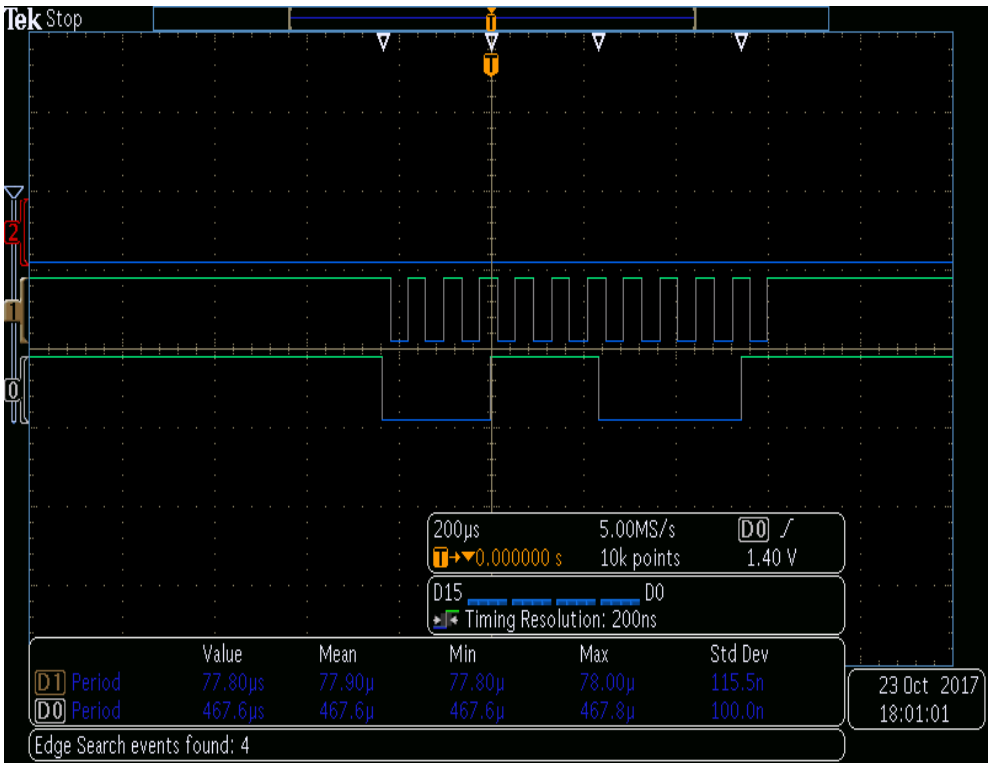


Figure 6: screen shot

Issue:

When you do this lab you have to be patient for connect wire. Be sure when you solder splice wire back together as hard as you can because it will infect your signal transmission and you won't get any feedback based on that.

Conclusions:

For this lab we learned how PS/2 protocol for keyboard works on the computer and how to use interrupt in our code and how to write it. More familiar how to find wires on device such as n/c, clock and data.