Sherman Lam
E155
October 28, 2014

Lab 6 Report: Wireless Calculator

1 Introduction

In this lab, I built a wireless calculator using a PIC32 microcontroller and a bluetooth module. The module used was the Sparkfun BlueSMiRF. The PIC interfaces with the BlueSMiRF through UART. On the other end of the bluetooth link is a generic USB bluetooth module connected to my laptop. I send mathematical expressions over the bluetooth link through a PuTTY terminal.

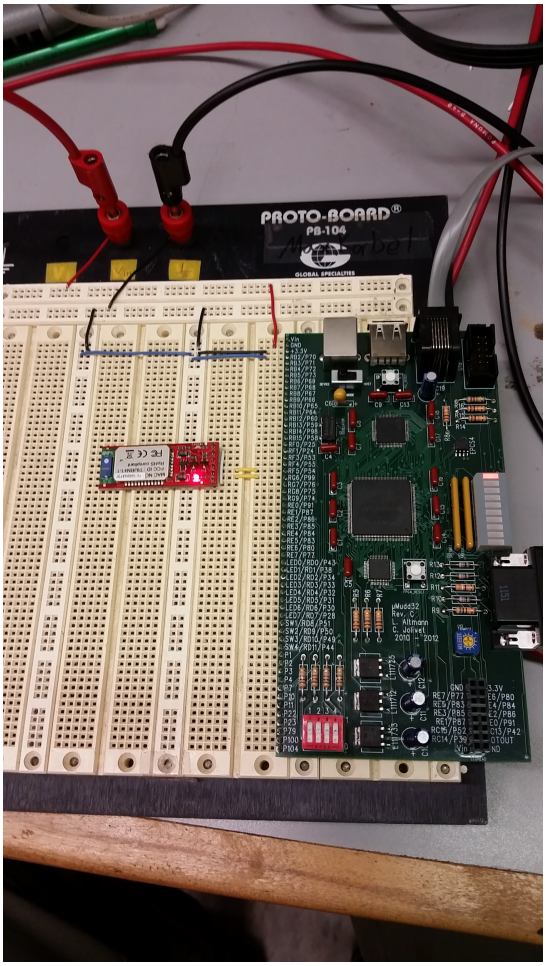

Figure 1: The latest number entered on the keypad is displayed on the bottom display. The second latest number is displayed on the top.

2 Design and Testing Methodology

2.1 Algorithm

When strings are received, they contain a wide variety of chars. This includes numbers, mathematical symbols, and string modification characters (e.g. delete, new line, and carriage return). To correctly interpret the input string, I first "clean" the string by removing all white spaces, new lines, and carriage returns. I then modify the string to account for backspaces. If the input was a valid mathematical expression, the final cleaned expression is of form A(B)C where (B) is an operator. The expression can then be parsed and evaluated in a consistent manner. The operations handled in this calculator are summation (+), subtraction (-), multiplication (*), and integer division (/).

2.2 Hardware

There were no special hardware connections needed. The BlueSMiRF was powered from 3.3V. The CTS and RTS pins on the BlueSMiRF were connected together as instructed by the lab. The TX pin on the BlueSMiRF was connected to the RX pin of the PIC's UART2 (RF4). The RX pin on the BlueSMiRF was connected to the TX pin of the PIC's UART22 (RF5).

2.3 UART

The PIC communicates with the BlueSMiRF through UART2. May computer communicates with the bluetooth dongle through a serial connection. The communication protocols used in each are defined as follows:

PIC-BlueSMiRF protocol:

Speed: 115.2k baud

Data bits: 8

Stop bits: 1

Parity: none

Flow control: none

Bluetooth Dongle - Computer protocol:

Speed: 9600 baud

Data bits: 8

Stop bits: 1

Parity: none

Flow control: none

When programming the PIC, one of the registers that must be set is the U2BRG. This register controls the baud rate of the UART connection. The BRG is calculated by the following:

$$f_{UART} = \frac{f_{peripheral-clock}}{16(BRG+1)}$$
$$BRG = \frac{f_{peripheral-clock}}{16 f_{UART}} - 1$$

So, for a baud rate (f_{UART}) of 115.2kHz and a peripheral clock of $20MHz$, the desired value of BRG is:

$$\begin{aligned} BRG &= \frac{f_{peripheral-clock}}{16 f_{UART}} - 1 \\ &= \frac{20MHz}{16 * 115.2kHz} - 1 \\ &\approx 10 \end{aligned}$$

2.4 Test Cases

Each of the supported operations where tested. Various incomplete expressions were also tested to check for error handling. The program executes the operations correctly and prints an error statement when asked to evaluate invalid or incomplete expressions. The following is a subset of the test cases used as well as their corresponding outputs.

You typed: 132+456
Answer: 588

You typed: 789-123
Answer: 666

You typed: 5-6
Answer: -1

You typed: 7*6
Answer: 42

You typed: 6/4
Answer: 1

You typed: 14/0
Cannot divide by 0

You typed: 0/456
Answer: 0

You typed: 1+
Incomplete equation

You typed: sdfg
Equation invalid

You typed: 123+abc
Equation invalid

3 Technical Documentation

This section provides the C-code used to program the PIC.

3.1 C code

```
1  #include <stdio.h>
2  #include <P32xxxx.h>
3
4  /* Functions prototypes */
5  void initMODE(void);
6  void initSTA(void);
7  void initBRG(void);
8  char getcharserial(void);
9  void getstrserial(char*);
10 void sendcharserial(char);
11 void sendstrserial(char*);
12 void clean(char*, char*);
13 void parse(char*);
14 int isNum(char);
15 int isOp(char);
16 void domath(int, int, char);
17
18
19 /*
20 This inits UART control register values.
21
22 Author: Sherman Lam
23 Email: slam@g.hmc.edu
24 Date: 10-16-14
25 */
26 void initUART(void){
27     // set I/O pins
28     TRISFbits.TRISF5 = 0;
29     TRISFbits.TRISF4 = 1;
30
31     //call all the individual init functions
32     initMODE();
33     initSTA();
34     initBRG();
35 }
36
37
38 /*
39 This inits the control register U2MODE. From lecture slides.
```

```
40
41  bit 31-16: unused
42  bit 15: ON = 1: enable UART
43  bit 14: FRZ = 0: don't care when CPU in normal state
44  bit 13: SIDL = 0: don't care when CPU in normal state
45  bit 12: IREN = 0: disable IrDA
46  bit 11: RTSMD = 0: don't care if not using flow control
47  bit 10: unused
48  bit 9-8: UEN = 00: enable U1TX and U1RX, disable U1CTSb and U1RTSb
49  bit 7: WAKE = 0: do not wake on UART if in sleep mode
50  bit 6: LPBACK = 0: disable loopback mode
51  bit 5: ABAUD = 0: don't auto detect baud rate
52  bit 4: RXINV = 0: U1RX idle state is high
53  bit 3: BRGH = 0: standard speed mode
54  bit 2-1: PDSEL = 00: 8-bit data, no parity
55  bit 0: STSEL = 0: 1 stop bit
56
57  Author: Sherman Lam
58  Email: slam@g.hmc.edu
59  Date: 10-16-14
60  */
61  void initMODE(void){
62      U2MODE = 0x8000;
63  }
64
65
66  /*
67  This inits the control register U2STA. From lecture slides.
68
69  bit 31-25: unused
70  bit 24-16: write 0 when not using auto address detect
71  bit 15-14: UTXISEL = 00: interrupt when TX buffer not full
72  bit 13: UTXINV = 0: U1TX idle state is high
73  bit 12: URXEN = 1: enable receiver
74  bit 11: UTXBRK = 0: disable break transmission
75  bit 10: UTXEN = 1: enable transmitter
76  bit 9: UTXBF: don't care (read-only)
77  bit 8: TRMT: don't care (read-only)
78  bit 7-6: URXISEL = 00: interrupt when receive buffer not empty
79  bit 5: ADDEN = 0: disable address detect
80  bit 4: RIDLE: don't care (read-only)
81  bit 3: PERR: don't care (read-only)
82  bit 2: FERR: don't care (read-only)
83  bit 1: OERR = 0: reset receive buffer overflow flag
84  bit 0: URXDA: don't care (read-only)
85
86  Author: Sherman Lam
87  Email: slam@g.hmc.edu
88  Date: 10-16-14
89  */
90  void initSTA(void){
91      U2STA = 0x1400;
92  }
93
94
95  /*
96  This inits the control register BRG. From lecture slides.
97
98  Want rate of 115.2 Kbaud
```

```
 99  Assuming PIC peripheral clock Fpb = Fosc / 2 = 20 MHz
100  based on default instructions in lab 1.
101  U3BRG = (Fpb / 4*baud rate) - 1
102  -> U3BRG = 10 (decimal)
103  Actual baud rate 113636.4 (-1.2% error)
104
105  Author: Sherman Lam
106  Email: slam@g.hmc.edu
107  Date: 10-16-14
108  */
109  void initBRG(void){
110      U2BRG = 10;
111  }
112
113
114  /*
115  This waits until there is a char available from the serial port and returns it
116
117  Author: Sherman Lam
118  Email: slam@g.hmc.edu
119  Date: 10-23-14
120  */
121  char getcharserial(void){
122      //wait for data to be available
123          //printf("Reading serial\n");
124      while(!(U2STA & 0x1)){}
125
126      //return char
127      return U2RXREG;
128  }
129
130
131  /* This reads a string from serial
132
133  Author: Sherman Lam
134  Email: slam@g.hmc.edu
135  Date: 10-23-14
136  */
137  void getstrserial(char* str){
138      int i = 0;
139      do {                                // read an entire string until detecting
140          str[i] = getcharserial();       // carriage return
141      } while (str[i++] != '\r');         // look for carraige return
142      str[i-1] = 0;                       // null-terminate the string
143  }
144
145
146  /* This writes a single char to the tx register
147
148  Author: Sherman Lam
149  Email: slam@g.hmc.edu
150  Date: 10-23-14
151  */
152  void sendcharserial(char data){
153      //wait until the transmit buffer has space
154      while(U2STA & 0x200){}              // if bit 9 = 1 -> buffer full
155
156      //write to buffer
157      U2TXREG = data;
```

```
158 }
159
160
161 /* This writes a string to the tx register
162
163 Author: Sherman Lam
164 Email: slam@g.hmc.edu
165 Date: 10-23-14
166 */
167 void sendstrserial(char* str){
168     //first send a newline and carriage return symbol to start
169     //at the beginning of a new line
170     sendcharserial('\n');
171     sendcharserial('\r');
172
173     //send the str
174     int i = 0;
175     while(str[i] != 0){
176         sendcharserial(str[i]);
177         i++;
178     }
179 }
180
181 /* This method checks is the char is number
182  *
183  * Author: Sherman Lam
184  * Email: slam@g.hmc.edu
185  * Date: 10-24-14
186  */
187
188
189 /* This method cleans the string of new line chars, spaces, etc
190  *
191  * Author: Sherman Lam
192  * Email: slam@g.hmc.edu
193  * Date: 10-24-14
194  */
195 void clean(char* input, char* output){
196     char c = '0';
197     int i = 0;       // index for iterating through input
198     int j = 0;       // index for iterating through output
199     do{
200         c = input[i];
201         if (c == ' '){}             //check if char is whitespace.
202         else if ((int)c == 127){    // check backspace
203             j--;                    // decrement index to overwrite last value
204             if (j<0){
205                 j=0;
206             }
207         }
208         else{                       // if good, write
209             output[j] = c;
210             j++;
211         }
212         i++;
213     } while (c != 0);               // break if we see null terminator
214
215     //write null terminator
216     output[j] = 0;
```

```
217
218 }
219
220
221 /* This method checks if the input char is a number
222  between 0 and 9
223
224  Author: Sherman Lam
225  Email: slam@g.hmc.edu
226  Date: 10-25-14
227  */
228 int isNum(char c){
229     return ((48<=c) && (c<=57));          // check ascii bounds
230 }
231
232
233 /* This method checks if the input char is a valid operator
234
235  Author: Sherman Lam
236  Email: slam@g.hmc.edu
237  Date: 10-25-14
238  */
239 int isOp(char c){
240     int state = 0;
241     //check all supported operators
242     state |= (c=='+');
243     state |= (c=='-');
244     state |= (c=='*');
245     state |= (c=='/');
246     return state;
247 }
248
249
250 /* This method does math given two integers and an operator
251
252  Name: Sherman Lam
253  Email: slam@g.hmc.edu
254  Date: 10-25-14
255  */
256 void domath(int a1, int a2, char op){
257     int answer = 0;
258     if (op == 43){            // +
259         answer = a1 + a2;
260     }
261     else if (op == 45){       // -
262         answer = a1 - a2;
263     }
264     else if (op == 42){       // *
265         answer = a1 * a2;
266     }
267     else if (op == 47){       // /
268         if (a2==0){
269             printf("Cannot divide by 0 \n");
270             return;
271         }
272         answer = a1 / a2;
273     }
274     else{
275         printf("Operation: %c is not supported.\n\r");
```

```
276             return;
277         }
278 
279         printf("Answer: %d \n",answer);
280     }
281 
282 
283 
284     /*
285      This method parses the string and chooses which operation to perform
286 
287      Author: Sherman Lam
288      Email: slam@g.hmc.edu
289      Date: 10-24-14
290      */
291     void parse(char* str){
292         int i = 0;          // for indexing through string
293         char c;             // a char
294         int a1 = 0;         // number argument 1
295         int a2 = 0;         // number argument 2
296         char op = 0;        // operator
297         int update1 = 0;    // whether a1 was set
298         int update2 = 0;    // whether a2 was set
299         //get the first number
300         do{
301             c = str[i];
302             if (isNum(c)){
303                 a1 += a1*10+(c-48);
304                 update1 = 1;
305             }
306             //throw an error if a number was not entered
307             else if (!isOp(c)){
308                 printf("Equation invalid \n");
309                 return;
310             }
311             i++;
312         } while(isNum(c));  // check for number
313 
314         //get the operator
315         op = c;
316 
317         //get the second number
318         do{
319             c = str[i];
320             if (isNum(c)){
321                 a2 += a2*10+(c-48);
322                 update2 = 1;
323             }
324             //throw an error if a number was not entered
325             else if (!(c==0)){
326                 printf("Equation invalid \n");
327                 return;
328             }
329             i++;
330         } while(c != 0);  // check for null terminator
331 
332         // if we have all the variables, do the math. Else, print error
333         if(update1 && update2){
334             domath(a1,a2,op);
```

```
335         }
336         else{
337             printf("Incomplete equation \n");
338         }
339     }
340
341
342     /*
343     This is the main loop for interfacing with the calculator
344
345     Author: Sherman Lam
346     Email: slam@g.hmc.edu
347     Date: 10-16-14
348     */
349     void main(void){
350         // initialize the UART communiations
351         initUART();
352
353         //loop
354         while(1){
355                 //prompt the user for an equation
356                 sendstrserial("Please enter an equation \n\r");
357
358                 //read the string
359                 char str[80];
360                 getstrserial(str);
361                 printf("\rYou typed: %s\n\r", str);
362
363                 //clean and parse the string
364                 char str1[80];
365                 clean(str,str1);
366                 parse(str1);
367             }
368     }
```

4 Results and Discussion

The calculator performs as expected. One of the problems I was unable to fix in time is associated with the printing of backspaces. If I enter in an expression that consists only of backspaces, the output line that states "You typed: ..." will be truncated to something along the lines of "You ty ". However, this does not affect the printing of the answer.

Also note that the backspace key does not correspond with the backspace ASCII code 8. Instead, it corresponds with the delete code 127.

5 Conclusion

5.1 Time Spent

Programming 6 hrs

Writing Report 2hrs

Total Time Spent 8hrs

5.2 Suggestions for lab

No suggestions for the lab.