ENGR 101: Robotics
Lecture 2 – Text Programming

- Outline
  - Introduction to PBASIC
  - Variables, I/O, Arithmetic
  - Controlling the LEDs

- References
  - http://csserver.evansville.edu/~richardson/
    - PBASIC Programming Guide: Setting Up
    - PBASIC Programming Guide: Writing Programs
    - BASIC Stamp Syntax and Reference Manual

Lecture 2 – Text Programming
Introduction to PBASIC

- The Scribbler robot uses a Parallax BASIC Stamp 2 (BS2) microcontroller. The BS2 understands only binary instructions.
- We write programs for the BS2 in PBASIC (a human readable programming language). The BASIC Stamp Editor translates PBASIC to the binary form understood by the BS2 when we download the program to the Scribbler. (PBASIC allows for more precise control of the robot than the GUI language.)

Lecture 2 – Text Programming
The BASIC Stamp Editor

- Connect the Scribbler to the serial port of the PC and turn it on. Start the Basic Stamp Editor and click on the Identify icon to check the connection.

- Add an “END” statement to create an empty, do-nothing (but complete) program.
  ' {$STAMP BS2}
  ' {$PBASIC 2.5}
  END

- Download the program to the Scribbler by clicking on the Run icon.

Lecture 2 – Text Programming
Introduction to PBASIC

- Click on the BS2 icon and the PBASIC 2.5 icon to add the correct editor directives to your program. (You can type these in by hand if you want to.) This ensures that the correct binary instructions for the Scribbler are generated by the editor.

- Something a little more exciting ...
  ' {$STAMP BS2}
  ' {$PBASIC 2.5}
  DEBUG "Hello world!"
  END

- After downloading the program, press the RESET button on the Scribbler to run the program multiple times. What happens at the computer output terminal?
Lecture 2 – Text Programming
Introduction to PBASIC

- The DEBUG statement causes the Scribbler to send information back to the computer. We only see this information when the serial cable is connected.
- The computer terminal receiving the information interprets all data as ASCII code. (Refer to the partial ASCII code table on pg 36 of the Writing Programs document.)

What will this statement do?
```
DEBUG "Hello ", 119, 111, 2*57, 100+8, 200-100, 33, 13
```

What happens when you press RESET now? (13 is the non-printable ASCII code for a carriage-return, you can use the CR control code instead.)

What if we want to print a number? Use the DEC formatter:
```
This is a comment
DEBUG CLS    ' Clear screen
DEBUG "6*7 = ", DEC 6*7, CR
```

The DEC formatter converts the number 42 to the ASCII code for 4 followed by the ASCII code for 2. Without DEC the ASCII value 42 is sent to the terminal and we would see a ‘*’.

One last DEBUG example:
```
num1 VAR Word    ' define variables
num2 VAR Word
DO                   ' an infinite loop
  DEBUG CLS      ' clear screen
  DEBUG "Enter first number: "
  DEBUGIN DEC num1 ' DEBUGIN reads terminal
  DEBUG "Enter second number: "
  DEBUGIN DEC num2
  DEBUG "Product is ", DEC num1*num2, CR
  DEBUG "Press ENTER to repeat ..."
  DEBUGIN num1    ' Read a dummy value
LOOP
```

Lecture 2 – Text Programming
Controlling LEDs

- The BS2 has sixteen I/O pins (0 – 15). Pins 8, 9, and 10 are connected to the right, center, and left LEDs. Use the PBASIC HIGH and LOW commands to turn the LEDs on and off. Use PAUSE to pause for a desired number of milliseconds:
```
DO
  HIGH 9    ' Center LED ON
  PAUSE 500 ' 1/2 sec
  LOW 9    ' Center LED OFF
  PAUSE 500 ' 1/2 sec
LOOP
```

Write a program that turns on the LEDs one by one from right to left, 1/10 a second apart. Then, turn them off again one at a time, from right to left, again 1/10 a second apart. Make the pattern repeat in an infinite loop.

Turn in a print-out of your program and be prepared to demonstrate that your program works at the beginning of next class period.

Lecture 2 – Text Programming
Assignment 2 – Due Next Time!