Reading User Input

Last updated 6/21/21

Reading User Input

- Most embedded systems do not read in user input in the form of text or numbers
  - Digital signals are typically used for input
  - We will read user input to:
    - Aid in out programming practice
    - Debug our programs

- The input to a scanf() function is received from the standard input “stream” (stdin)
  - stdin in our case will be the Tera Term window

- The scanf() function halts our program until the required input is provided
  - Special care must be taken to prevent it from impacting the timing of our programs

Reading User Input

- User input must be stored in a variable

command: scanf()

argument: “%type”, &variable

This is the location where a variable of type will be read in - % is a special character to indicate the “type” follows next

type: i → int, f → float, c → char

This is the name of the variable where a value of type will be stored - & is a special character to indicate we are using a pointer (more later)

```
int ave;
scanf(“%i”, &ave);   //reads 1 int from the keyboard and stores the value in ave

float foo;
scanf(“%f”, &foo);   //reads 1 float from the keyboard and stores the value in foo

char initial;
scanf(“ %c”, &initial);   //reads 1 char from the keyboard and stores the value in initial
                          // note the space before %c – more later
```

Reading User Input

- Each variable in a single scanf statement needs its own format descriptor

Examples:

```
int count;

float ave;

char month;


printf("Enter an int for count, float for ave and character for month");

scanf("%i %f %c", &count, &ave, &month);
```

Reading User Input

- The scanf() function is very sensitive
  - Mismatch in type expected and type entered can lead to odd errors
- scanf uses pointers to access the storage variables
  - Don’t forget the &