LECTURE 3

ENGR5: Intro to Engineering Practice

MEM Project

LEGO Robo-Soccer

Main Data Types

- int (16 bit integer number in IC)

  +/-32,767

- long (32 bit long integer number)

  +/-2,147,483,647

- float (32 bit floating point number)

  e.g. 3.1416

Integer Arithmetic

- + addition

  X + Y means X plus Y

- - subtraction

  X - Y means X minus Y

- * multiplication

  X * Y means X multiplied by Y

- / division

  X / Y means X divided by Y with the decimal truncated

- % modulus (remainder)

  X % Y means the remainder of X divided by Y

Floating Point Arithmetic

- +,-,* operate the same as with integers
- / yields a floating point number
- % is not defined with floating point numbers

NOTE: i=i+1; $\Leftrightarrow$ i++;

i=i-1; $\Leftrightarrow$ i--;

Variables Definition

- Retain data for later use
- Use in constructions such as arithmetic expressions
- Must be declared before use at start of function (local variables) or outside any function (global variables)

  Syntax: <data-type> <variable-name>

  - int i; [integer variable]
  - float x; [floating point variable]
- To put a value in a variable (Assignment):

  Syntax: <variable-name> = <expression>

Variables Definition

```
void main()
{
        int r,j;                        /* declare r and j */
        j = 3;                          /* assign a value to j */
        r = j*j + 1;                    /* calculate and assign */
        printf("result is %d\n",r);
}
```

Built-in Functions

LCD SCREEN: printf(format string, arg1, arg2)

Print formatted strings on the LCD.

– %d is used for an integer
– %f is used for a float
– \n starts next character back at upper left corner (or new line on terminal)
– commas separate all arguments after the quoted string

```
printf("Hello!");

int x;
x=1;
printf("Values is %d \n",x);

float y;
y=1.1;
printf("x=%d, y=%f \n",x,y);
```

Flow Control: LOOPS

- Loops are used when you want to do the same thing multiple times:

  - e.g., beep 20 times

- You could type beep(); 20 times, but there is a better way: a LOOP!

Flow Control: LOOPS - While

Syntax: while (<test>) {statements}

Example: Beep 20 times

```
void main()
{
        int num;                    /* declare counter */
        num = 1;                    /* initialize counter */
        while (num <= 20)           /* loop while num is <=20*/
        {
           beep();                  /* beep once */
           num = num + 1;           /* add one to the counter */
        }
}
```

Flow Control: LOOPS - For

Syntax: for (<expr-1>;<expr-2>;<expr-3>) {statements}

Example: Beep 20 times

```
void main()
{
        int num;                                /* declare counter */

        for (num=1;num<=20;num++)               /* execute 20 times*/
        {
           beep();                              /* beep once */
        }
}
```

Flow Control: While vs. For

for (<expr-1>;<expr-2>;<expr-3>)
{
    statements;
}
is equivalent to:
<expr-1>;
while(<expr-2>)
{
    statements;
    <expr-3>;
}

NOTE: break allows an early exit from a while or for loop!

Booleans Expressions

- Boolean expressions result in either 1 (true) or 0 (false)
- Boolean operators:

  == (two equals signs together, not one)

  <, <=, >, >=

  != (not equal)

  || (or), && (and)

  ! not

Making Decisions: If

Syntax: if (<test>) {statements}

- The statements are skipped if the test is false

```
void main()
{
        int j=-2;
        if (j < 0)                  /* skip if >= 0 */
        {
                j = -j;             /* change sign of x */
        }
        printf("magnitude is %d\n",j);
}
```

Making Decisions: If - Else

Syntax: if (<test>) {statements for true case}

else {statements for false case}

- If the test is true {statements for true case} are selected
- If the test is false {statements for false case} are selected

Making Decisions: If - Else

```
void main()
{
        int j=-2;
        if (j < 0)          /* select less than 0 case */
        {
                printf("%d is negative\n",j);
        }
        else                /* select greater than or equal to 0 case */
        {
                printf("%d is non-negative\n",j);
        }
}
```

Built-in Functions

MOTORS:

void motor(int m, int p)

Turns on motor m at power level p.
Power levels range from 100 for full on forward to –100 to full on backward.

void alloff()

Turns off all motors.

Built-in Functions

TIME: void sleep(float sec)

Waits for an amount of time equal to or slightly greater than sec seconds. sec is a floating point number!!!

```
/*Wait for 5 seconds*/
sleep(5.0)
```

SOUND: void beep()

Turns on motor m at power level p.
Power levels range from 100 for full on forward to –100 to full on backward.

Built-in Functions

SENSORS: int digital(int p)

Returns the value of the sensor in port p, as a true/false value (1 for true, 0 for false).

Built-in Functions

SENSORS: int light(int p)

Returns the value of the light sensor in port p.

int light_passive(int p)

Returns the value of the light sensor in port p when the LED emitter is turned off. NOT very useful in this case.

Built-in Functions

BUTTONS: int stop_button()

Returns the value of the stop/prgm button.
1 if pressed, 0 if released.

Assignment

- Build a program that prints the numbers 1 to 5 for one second each on the LCD. The program should repeat the printing sequence until the “STOP” button is pressed.

- Build a program that uses a touch sensor to invert the direction of the car. The number of inversions should be displayed on the LCD. The program must stop all the wheels and finish its execution after 10 inversions.