EE 299 Lab 2
First Steps to Design

University of Washington - Department of Electrical Engineering

James K. Peckol

Introduction:

This lab has two main purposes. The first is to now utilize several of the peripheral drivers that we designed in the first lab in a more complex design and the second is to begin to learn the formal design process as we work through the development cycle.

This project is based upon one of the ideas submitted by a number of people at the start of the quarter. Our project will be to design and build a basic four function calculator.

Prerequisites:

Designed a number of the bits and pieces in the first lab that we'll now need and use. Have learned how to spell and pronounce C. Ready to develop and extend your design skills.

Interrequisites:

Start refining and extending your design, debug, and test skills by completing the development cycle, including the demo and report, for a basic four function calculator.

Postrequisites:

Only if you are twenty-one or more.

Background

In the first lab project, we observed that embedded systems development required several things. Let's now expand on that list. We need,

1. To recognize the problem we are trying to solve.
2. A good understanding of that problem and the requirements.
3. A specification of what and from which we are to design.
4. Some tools to work with.
5. A target platform on which to develop the application.
6. A mechanism for programming the target platform.

Such requirements really apply to the design of any kind of system.

We begin with a high level statement of the necessary requirements. What does our user want in the product?

Requirements

The project is to design and implement a basic four function calculator. The calculator must be able to perform addition, subtraction, multiplication, and division. The results of any calculations must be presented on a clear, easy to read display.

Inputs

Two numbers.

The operation to be performed on the two numbers.

Outputs

The result of performing the specified operation on the two numbers.

Functional Description

The user should be able to enter a number, the operation to be performed, a second number then be able to tell the calculator to perform the operation. The result should then be clearly displayed.

We now have a high-level description of what our user wants us to do. Unfortunately, we can't effectively design a system from such a view. We need specific details. We now put together a formal specification.

Design

The project is to design and implement a basic four-function calculator. The calculator must be able to perform addition, subtraction, multiplication, and division. The results of any calculations must be presented on a clear, easy to read display.

General Description

A basic four-function calculator is to be designed and implemented. The target platform will be the Arduino UNO and the system software will be implemented in the C language.

The calculator must support addition, subtraction, multiplication, and division. The numeric data and arithmetic operation entered by the user will be displayed on an LCD. After the arithmetic operation has been performed, the result will be displayed on an LCD.

The development of the system will be executed in three phases. The first two phases will deliver the basic functionality implemented on a single Arduino UNO. The third phase will decompose and migrate the system to two Arduino UNOs. Upon completion of the third phase, the final product will be delivered.

Inputs

Phase 1

Two numbers – the numbers shall be single digit integers in the range of 0..9.

The numbers will only be positive.

Phase 2

Two numbers – the numbers shall be double digit integers in the range of 00..99. The numbers will only be positive.

Phase 3

Two numbers – the numbers shall be double digit integers in the range of 00..99. The numbers may be positive or negative

All phases

The operation to be performed on the two numbers shall be addition, subtraction, multiplication, or division.

These operations shall be specified by the standard arithmetic operator symbols:

+	add
-	Subtract
*	Multiply
/	Divide

Outputs

As each number or operation is entered, it shall be displayed, in the sequence entered, on the first row of an LCD display.

The result of performing the specified operation on the two numbers shall be presented on the second row of an LCD display as an integer.

Functional Description

The system shall comprise three major functional blocks

- User Input
- Arithmetic Calculation
- Display user input and results of the calculation

User Input

The numbers and operation shall be entered via the Serial Monitor on the Arduino UNO in the following sequence

Phase 1

- Number 1
- Operation
- Number 2
- =

Phase2

- Number 1
  - MSD LSD – most significant digit least significant digit
- Operation
- Number 2
  - MSD LSD – most significant digit least significant digit
- =

Phase 3

- - dash or minus sign if negative
- Number 1
  - MSD LSD – most significant digit least significant digit
- Operation
- - dash or minus sign if negative
- Number 2
  - MSD LSD – most significant digit least significant digit
- =

After the = character is entered, the result should be displayed on the LCD.
All numeric entries should be integral values.

The result should be an integral value.

Arithmetic Calculation

Phases 1..3

The calculator shall support the four basic arithmetic functions:

Addition	+
Subtraction	-
Multiplication	*
Division	/

Each arithmetic function should only support operations between two user-entered numbers. Chained operations will not be supported.

Display

Phase1 and Phase 2

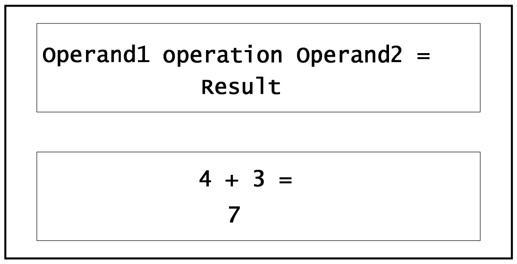
The output display must be a 2 row x 16 column LCD display.

Phase 3

The polarity of the operand shall be displayed with the operand

Phase 1.. 3

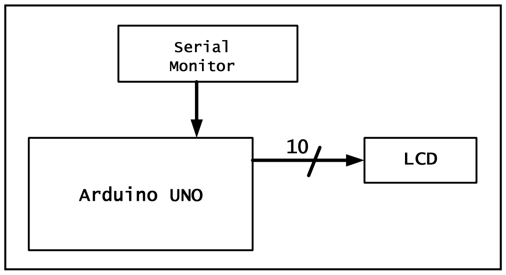
The layout and organization of the display is given in the following figure.



Architecture

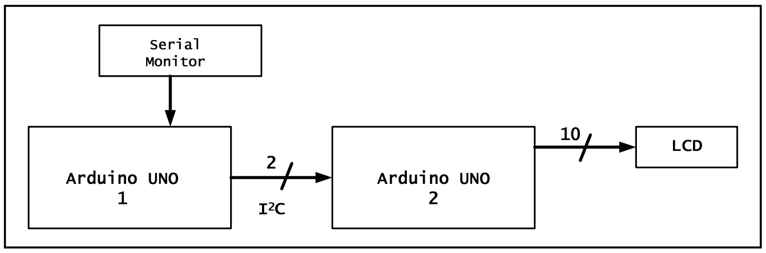
Phase 1 and Phase 2

The system shall be implemented utilizing a single Arduino UNO microprocessor as illustrated in the block diagram in the following figure,



Phase 3

The system shall be decomposed and implemented as a distributed system utilizing two Arduino UNO microprocessors as illustrated in the block diagram in the following figure,



In Phase 3, the system functions will be distributed as follows:

Arduino UNO 1 will manage

- ✓ User input
- ✓ Conversion from ASCII to integer
- ✓ All arithmetic calculations

Arduino UNO 2 will manage

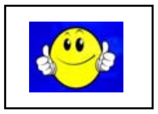
- ✓ All display functions

The user's inputs and the result of the calculation will be sent to UNO 2 for proper display on the LCD.

Now that we have a specification that we can design from…the next step is doing it.

Design – Implement – Debug

Your job !!!!!



Once the design is complete, we test it to make sure that it does what the specification requires. Sure hope that it does.

Test

Also your job !!!!!!!

Finally writing the final report

Report

Your job too !!!!!!!

Deliverables

A lab report containing

1. The annotated source code for all your *programs* / *sketches* for the UNOs.
2. Representative screen shots showing the results of executing the applications on the PC.
3. A short user’s manual for the calculator.

Appendix A

Converting ASCII Characters to Integers – Part 1

Data is entered into the system as ASCII characters. Before any arithmetic operations can be performed with the numbers, the ASCII characters representing those numbers must be converted into integers.

The ASCII encoding of the integer digits is given as follows,

Character	Binary	Hex
0	00110000	30
1	0011000	31
2	00110010	32
3	00110011	33
4	00110100	34
5	00110101	35
6	00110110	36
7	00110111	37
8	00111000	38
9	00111001	39

Looking at the Hex (hexadecimal or base 16) representation, we see that if we subtract the hex number 30 from the hex form of any of the ASCII characters, we will get the integer corresponding to that character. For example, consider the ASCII character ‘7’.

The character 7 is represented as the hex number 37, thus

37 – 30 = 7

While the above expression is correct, since the ASCII character ‘0’ is 30, it’s more clear to write

character – ‘0’ = 7

where character, in this case is the ASCII character ‘7’

Converting ASCII Characters to Integers – Part 2

Data is entered into the system as ASCII characters. Before any arithmetic operations can be performed with the numbers, the ASCII characters representing those numbers must be converted into integers.

For Phase 2 and Phase 3 of this lab we need to convert two digit ASCII characters into the corresponding integer number. This is not much more difficult that converting a single digit.

The two digit number, in ASCII format will comprise the corresponding two characters. Consider the integer number 89. This number is represented in ASCII by the two characters ‘8’ and ‘9’. Thus, ‘8’ ‘9’.

To convert to the corresponding two digit integer number, we can write

('8'– '0') ● 10 + ('9'– '0') → 89

Read from the Serial Port Monitor

A simple program, SerialIn0.ino, for reading user input is given on the class web page in the Lab 2 folder.

Input from the user is entered into a simple array. Thereafter, individual entries can be accessed as one would normally access the elements of an array.

It is important that the array is large enough to hold all of the data the user wishes to enter.

Example

```
char myData[10];                // an array to hold the user input up to 10 characters
```

Assume that the user entered the two digit number 19 followed by the enter key The array myData will now hold those characters. Now, let one and two be working char variables:

```
char one = 0;
char two = 0;
```

Next writing

```
one = myData[0];
two = myData[1];
```

will assign the ASCII character '1' to the variable one and the ASCII character '9' to the variable two.

We now must convert both character variables to integers as we did above to be able to use them in arithmetic operations.