ECE 367 Experiment #10 Fall 2012 Due Week 13 (11/27/2013)

(Not so) Simple Three function Calculator

This experiment is worth double points: 50 points

Design and implement a system that will allow a user to do addition, subtraction, or multiplication.

You will need to use the entire matrix keypad. So, you will need to read all of the column outputs. Keys 0 to 9 will function as normal base 10 digits. You must assign (and implement) the function keys for addition (+), subtraction (-), multiplication (X), and equivalence (=).

Extra credit: You can earn 20% extra credit (10 points) on the experiment if you can implement BOTH clear (CL) and additive inverse (-) keys. Clear should clear the last digit entry with one push or clear out all digits and functions with two consecutive key presses (as is implemented in most standard calculators.) Additive inverse is the implementation of a sign key. i.e. keypad input of a negative number. The sign should be a toggle function -, +, -, +, etc..

You can earn 20% more extra credit (10 points) if you can implement division with two-decimal accuracy. Example 1/3 = 0.33

No partial extra credit. The extra credit functions must work completely

You are only required to implement 3 digit (or less) input. So, your calculator must be able to do the following type of calculations:

243 + 110

123 X 101

342 - 356

21 - 357

31 - 2 etc.

Display the complete solution including the sign if required.

Of course the display is the LCD!

You may set your own rules on how the user may use the calculator. i.e what causes an error, how to start a new calculation, if continuous calculation is allowed etc.

There will be a great deal of coding in this experiment. I strongly recommend that you break this up into many subroutines - Initialize, Getkey, Checkkey, Output, Update_digits, etc. Each subroutine has one job to do and

then returns.

15 points extra credit if you can do all four operations with any 4-digit numbers. Yep, 4-digit decimal accuracy for division.

No partial extra credit.

The report should follow the standard format.

Have fun with this experiment.

Last modified: Wed Nov 14 20:48:00 2012

ECE 367: Experiment 10

(Not So) Simple Four Function Calculator

Semester: Fall 2012

Name: Kai Zhao

Signature: _____

UIN: 670720413

Due Date: 2012 November 27

Lab Section: T11

TA: Chenjie Tang

User Manual:

This device is a not so simple four function calculator. The user should first supply power to the circuit and then press the reset button. The first operand is displayed on the top left, the second operand is displayed on the top right, and the solution is displayed on the bottom. On startup, the input is for the first operand, which is limited to 4 characters. After pressing any operation, then the input will be for the second operand, which is also limited to 4 characters. There is not an equal key because the calculator continuously equates the function.

'A' is to divide the first operand by the second operand

'B' is to multiply the two operands together

'C' is to subtract the second operand from the first operand

'D' is to add the two operands together

'E' is negate, which is a toggle and negates the sign of the current operand input

'F' is backspace/clear. Press 'F' once for backspace. Press 'F' twice to clear all inputs

Some example inputs and output:

$243 + 110 = 353$

$123 \times 101 = 12423$

$342 - 356 = -14$

$1 \div 3 = 0.3333$

$9999 + 9876 = 19875$

$9999 + (-9876) = 123$

$-9999 * 9998 = -99970002$

$9999 \div 11 = 909$

$157 \div 13 = 12.0769$

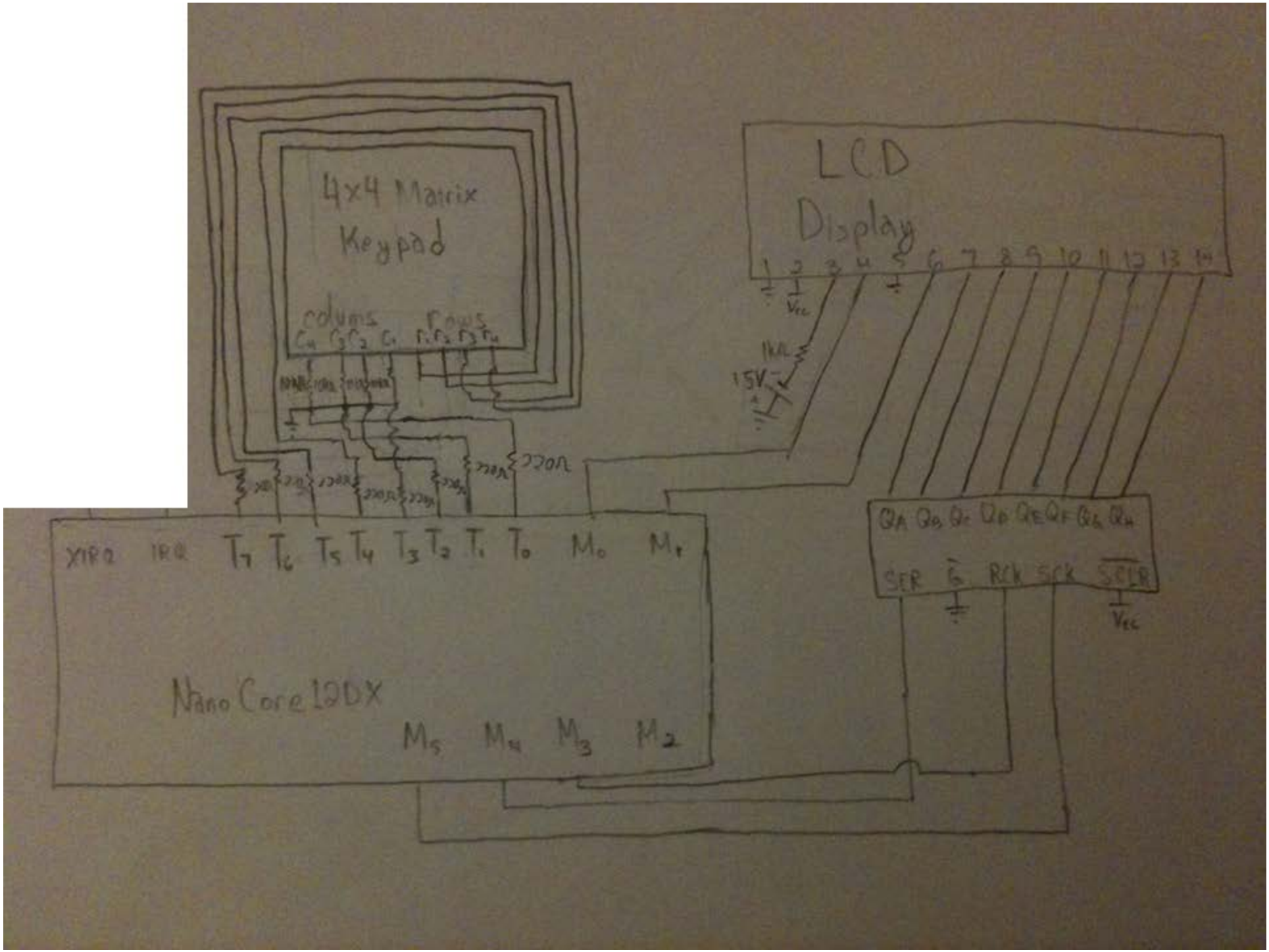
Conclusion:

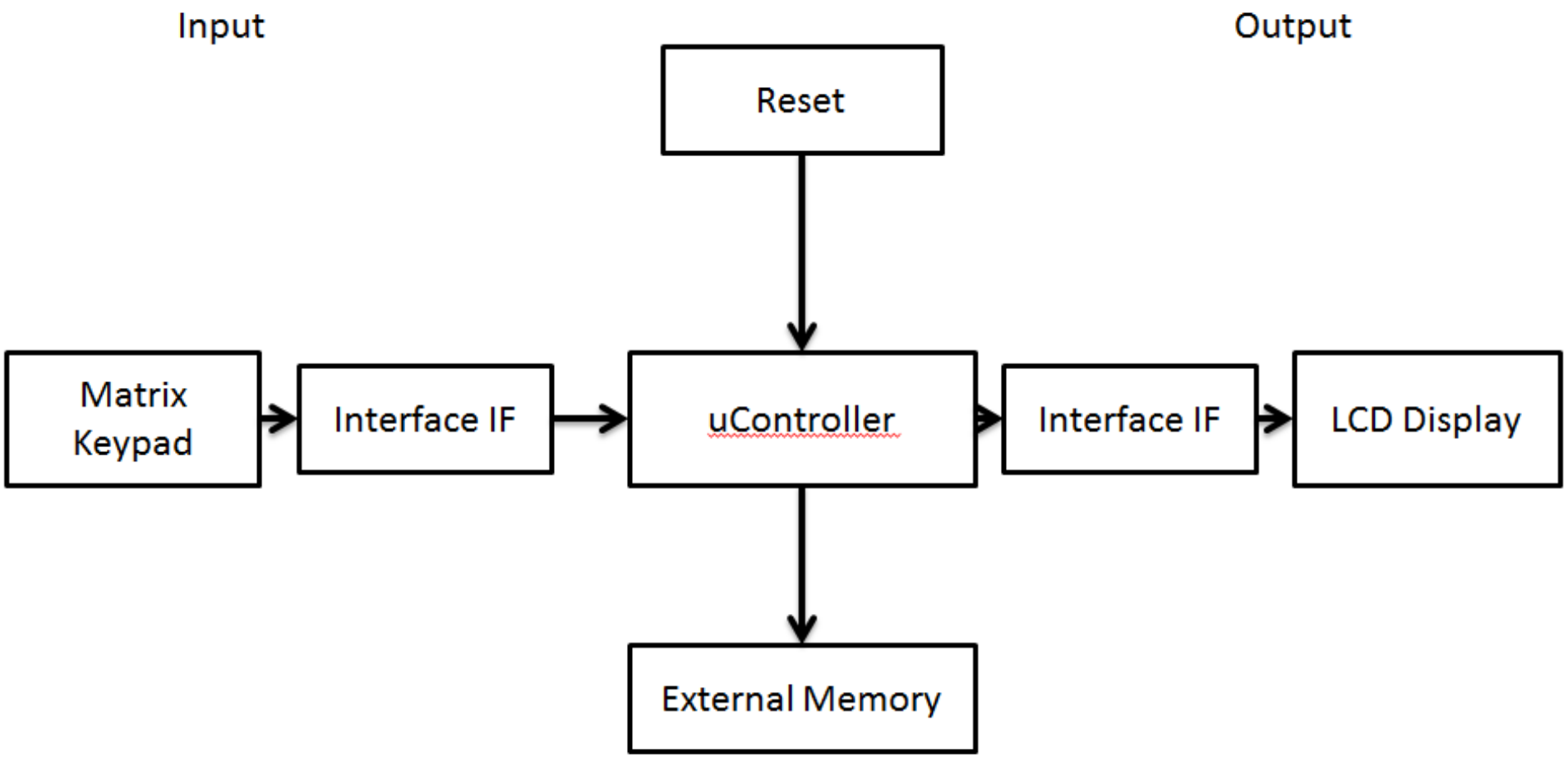
Yes, my circuit meets the specifications and function properly.

I had trouble converting sequential inputs into a whole number because of the way 16 bit numbers work and the way numbers are stored. In other words, when I used LDD, M is loaded onto A and M+1 is loaded onto B and there were no option for the reverse. I had trouble using the built in 16 and 32 bit multiple and divide functions because I had trouble converting each individual digit into 16 and 32 bits. Therefore, I had to compute each digit individually. I would do this project differently by combining the division for the quotient and the division to compute the decimal remainder.

I have implemented all the extra credit, which included the clear key, additive inverse, division, and implementation of 4 digit numbers. An extra feature I have implemented is that instead of having to press the equal key, the calculator automatically computes. This is a similar feature to Google's instant search where the results may already be shown before the full input, which increases the calculator's utility.

I learned about large number manipulation in assembly, which includes multiplication, division, and remainder decimal accuracy.


4x4 Matrix
Keypad
colums
rows
LCD
Display
1 2 3 4 5 6 7 8 9 10 11 12 13 14
Vcc
1kΩ
15V
220Ω
QA QB QC QD QE QF QG QH
SER G RCK SCK SCLR
Vcc
XIRQ IRQ T7 T6 T5 T4 T3 T2 T1 T0 M0 M1
NanoCore12DX
M5 M4 M3 M2


Input
Output
Reset
Matrix Keypad
Interface IF
uController
Interface IF
LCD Display
External Memory