

1 Introduction

In this lab you will create a simple stand alone fixed-point calculator. All arguments will be fixed numbers with format (-)x.x. The results will be decimal numbers with the format (-)xx.x without leading zeros. Operations include addition, subtraction, multiplication, and division. Data entry will be in reverse polish form with a maximum of two arguments.

2 Requirements

1. User input follows prompts for each argument and the operation. The results are then shown as follows: (in this case the user wants to add 1.0 to 2.0)

```
a1> 1.0↵  
a2> 2.0↵  
op> +↵  
r> 3.0↵
```

2. The operations are addition (+), subtraction (-), multiplication (*), and division (/). For division the numerator is the first argument and the denominator is the second argument.
3. Input format is fixed with one decimal point – x.x. If the user enters an incorrect entry, the program responds with the appropriate error message and prompts for the same argument. For example:

```
a1> 1.0↵  
a2> 20.0↵  
Too Large  
a2>
```

4. The results will be fixed also with one decimal point – xx.x. Leading zeros should be ignored. A result of zero must be displayed as 0.0. Results must be rounded to the nearest 0.x.
5. Errors detected include format errors, numbers too large, non-decimal digit, incorrect operation, and divide by zero.
6. After the result is displayed the program goes back and prompts for another first argument.
7. The program must be a standalone program stored in flash. Interrupts vectors are not required.
8. After MCU initialization but before starting the calculator, the program must calculate and display the checksum of the Flash block, \$C000-\$FFFF.

3 Program Design

1. It appears that you would need mixed numbers for this lab but that is not the case. The data entered has a single decimal place (not binary point) so, the easiest way to handle everything is to scale every number by 10. For example, if the number entered is 1.2, you would scale it to 12. When performing the division, it is easiest to scale the numerator by 100 before dividing. So, if 1.2 is the entered numerator, scale it to 120. This will result in the correct number of decimal places in the denominator.
2. You will have to modify the BasicIO subroutine, OUTDECW in order to display a decimal place and correctly display a result of zero. To do this, copy the routine from the Y: drive into your code, and then modify it. Note you will also have to change any BasicIO subroutine calls to upper case and include and equ for that routine.

4 Deliverables

1. Source programs must be emailed before midnight on the lab due date.
2. The write-up is due before 5:00pm on the Write-up due date. The write-up includes:

Introduction

Program Description. Including flow diagrams.

Listing. The assembled program listing.

Comments/Conclusions

Due Dates: Program Source – Friday, June 6, 2008. Write-up – Friday, June 6, 2008