Name: Tashyab Raj

Roll: 2101214

Date: 11-03-2024

Midsemester Project Report

Aim: Design and implement a scientific calculator application utilizing the 8051 micro controller platform.

Pseudo-code:

Start

Define BAUDRATE as 4800

Define lcd\_data as P1

Define rs as P0^0

Define en as P0^2

Define i as integer

Function UART\_Init:

Set TMOD as 0x20

Set TH1 as 0xFA

Set TR1 as 1

Set SCON as 0x50

Function msdelay with parameter ms:

Loop ms times:

Set TH0 as 0xFC

Set TL0 as 0x66

Set TR0 as 1

Wait until TF0 is 0

Set TR0 as 0

Set TF0 as 0

Function LCD\_cmd with parameter a:

Set lcd\_data as a

Set rs as 0

Set en as 1

Call msdelay with parameter 1

Set en as 0

Call msdelay with parameter 5

Function LCD\_init:

Call msdelay with parameter 20

Call LCD\_cmd with parameter 0x38

Call LCD\_cmd with parameter 0x0C

Call LCD\_cmd with parameter 0x06

Call LCD\_cmd with parameter 0x01

Call LCD\_cmd with parameter 0x80

Function LCD\_char with parameter a:

Set lcd\_data as a

Set rs as 1

Set en as 1

Call msdelay with parameter 1

Set en as 0

Call msdelay with parameter 2

Function LCD\_string with parameter s:

Loop until end of string s:

Call LCD\_char with current character of s

Move to next character of s

Function UART\_SendChar with parameter c:

Set SBUF as c

Wait until TI is 1

Set TI as 0

Function UART\_ReceiveChar:

Wait until RI is 1

Set RI as 0

Return SBUF

Function displayString with parameter str:

Loop until end of string str:

Call UART\_SendChar with current character of str

Move to next character of str

Function displayMenu:

Define menu as string containing menu options

Call displayString with menu

Function getOperand:

Define buffer as character array of size 10

Define c as character

Define i as integer and set to 0

Loop until newline or end of buffer:

Read character c from UART

Add c to buffer at index i

Call UART\_SendChar with c

Increment i

Add null terminator to buffer

Convert buffer to float and return

Function performOperation with parameters choice, operand1, and operand2:

Define result as float

Define result\_str as character array of size 20

Switch on choice:

Case '1':

Set result as operand1 plus operand2

Case '2':

Set result as operand1 minus operand2

Case '3':

Set result as operand1 times operand2

Case '4':

If operand2 is not 0:

Set result as operand1 divided by operand2

Else:

Call displayString with error message

Return

Case '5':

Set result as operand1 raised to the power of operand2

Case '6':

If operand1 and operand2 are greater than 0:

Set result as log of operand1 divided by log of operand2

Else:

Call displayString with error message

Return

Case '7':

Convert operand1 from degrees to radians

Set result as sine of operand1

Case '8':

Convert operand1 from degrees to radians

Set result as cosine of operand1

Case '9':

Convert operand1 from degrees to radians

Set result as tangent of operand1

Default:

Call displayString with invalid choice message

Return

Convert result to string with two decimal places

Call displayString with result\_str

Call LCD\_string with result\_str

Function calculator:

Call UART\_Init

Loop indefinitely:

Define choice as character

Call displayMenu

Read choice from UART

Call LCD\_cmd with 0x01

If choice is one of '1', '2', '3', '4', '5', or '6':

Define operand1 and operand2 as floats

Call displayString with message to enter first number

Read operand1 from UART

Call displayString with message to enter second number

Read operand2 from UART

Call performOperation with choice, operand1, and operand2

Return

ElseIf choice is one of '7', '8', or '9':

Define operand1 as float

Call displayString with message to enter the number

Read operand1 from UART

Set operand2 as 0

Call performOperation with choice, operand1, and operand2

Return

Else:

Call displayString with invalid choice message

Return

Function main:

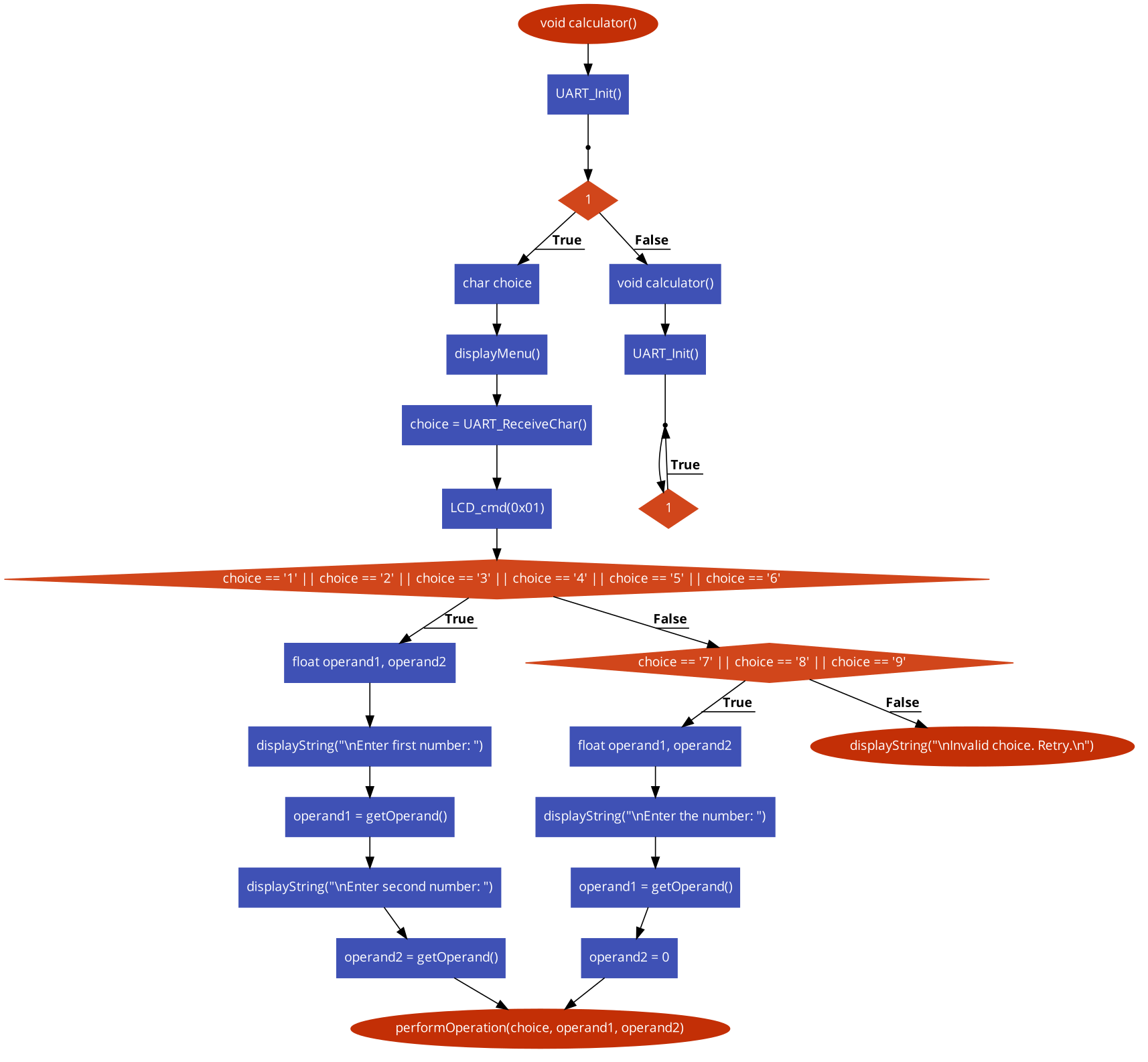
Call LCD\_init

Loop indefinitely:

Call calculator

End

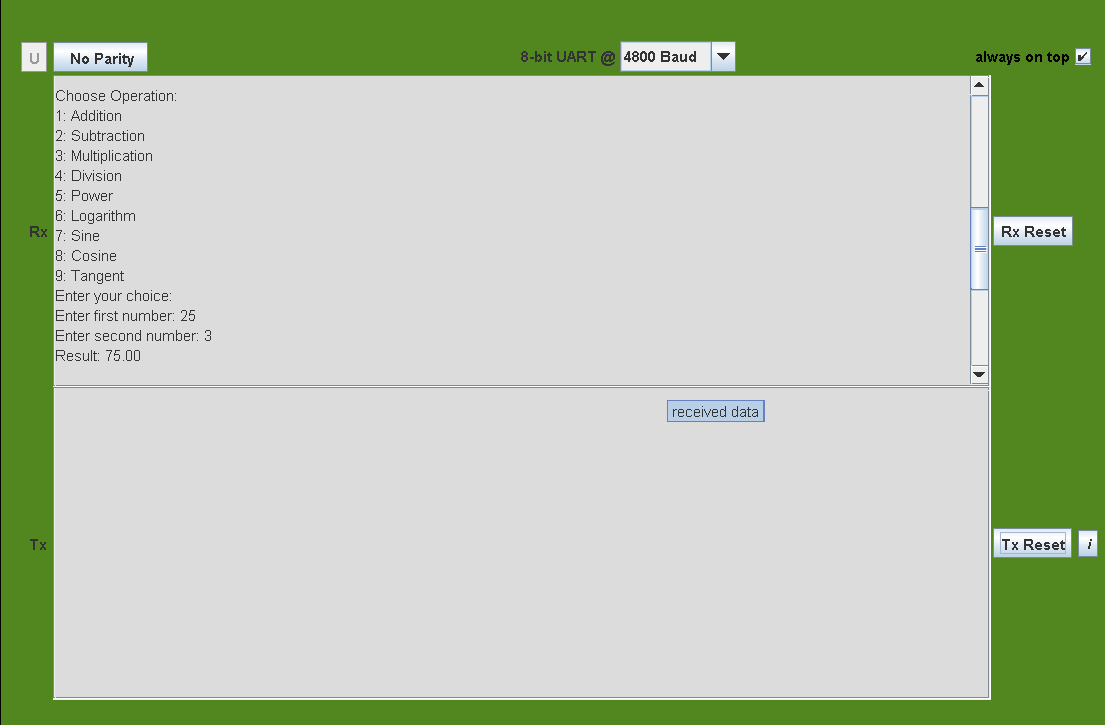
Flowchart:



Output (Examples):

1. Multiplication





1. Sine function





Remarks:

* The code demonstrates efficient interfacing with peripherals such as UART for communication and an LCD for display.
* It provides a comprehensive set of mathematical functions including addition, subtraction, multiplication, division, exponentiation, logarithm, trigonometric and other arithmetic operations.
* The program ensures robust error handling, including division by zero and invalid input.
* Proper initialization and configuration of hardware peripherals are evident, enhancing reliability and performance.
* The use of clear and concise functions aids in readability and maintainability of the code.