Scientific Graphing Calculator Using Touch LCD

A MINI-PROJECT SUBMITTED IN PARTIAL FULFILMENT
FOR THE COURSE OF

E3-257 Embedded Systems Design

BY

UJJWAL CHAUDHARY, M. TECH. ESE 2023-25
ANANYA PAL, M. TECH. ESE 2023-25

SUBMITTED TO

PROF. HARESH DAGALE

DEPARTMENT OF ELECTRONIC SYSTEMS ENGINEERING
INDIAN INSTITUTE OF SCIENCE, BANGALORE

MAY 2024

Synopsis

The report presents the development of a comprehensive graphing calculator with advanced mathematical functions. The calculator is capable of tracing curves, computing derivatives and integrals, finding odd roots, calculating the area under curves, and performing zoom and shift operations.

All functions of the calculator are implemented in a header file named `evaluate.h`, which provides a convenient and easily accessible interface for users. This allows anyone to utilize the functionalities of the calculator by simply including the `evaluate.h` file in their code.

Moreover, the calculator is designed to be customizable, with certain macros that can be easily changed to suit specific requirements. This flexibility allows users to adapt the calculator to their needs and preferences, making it a versatile tool for mathematical analysis.

Acknowledgements

We would like to express our sincere gratitude to the Embedded Systems lab at the Department of Electronic Systems Engineering (DESE), Indian Institute of Science (IISc), Bangalore for providing us with the necessary components, including the Tiva board and Touch LCD Booster Pack, which were instrumental in the development of our graphing calculator.

We are also thankful to the Prof. Haresh Dagale and teaching assistants of the course E3-257 Embedded Systems Design for their lectures, assignments, and guidance, which helped us build a strong foundation of concepts that were crucial for the successful completion of our project.

Lastly, we would like to extend our gratitude to our friends and all those who supported and encouraged us throughout this project.

Contents

List of Figures

Chapter 1

Application and Usage

1.1 Introduction

In today's digital age, integrating technology into educational tools has become important, particularly in mathematics education. Graphing calculators have long been a staple in math classrooms, aiding students in visualizing complex functions and equations. However, with the advent of touchscreen technology, there lies an opportunity to revolutionize the traditional graphing calculator, making it more intuitive, interactive, and accessible.
The motivation behind building a graphing calculator on a touchscreen module as an embedded project stems from enhancing user experience, i.e., touchscreens offering a more intuitive interface than traditional button-based input systems. We have built a similar utility hardware tool like the Desmos graphing calculator website [1]. Moreover, integrating the graphing calculator into a touchscreen module makes the device more portable and versatile.

1.2 Features

There are several features available in the graphing calculator. We have implemented seven such features. The calculator's main panel, as shown in figure 1.1, enables the user to enter the mathematical expression he wants to visualize. For the sake of explanation of the calculator, we will take $f(x) = log(x) + sin(x) * cos(x) - 0.35$.

After writing down the expression, the user must tap on the "Plot" option. This will result in the screen shown in figure 1.2.

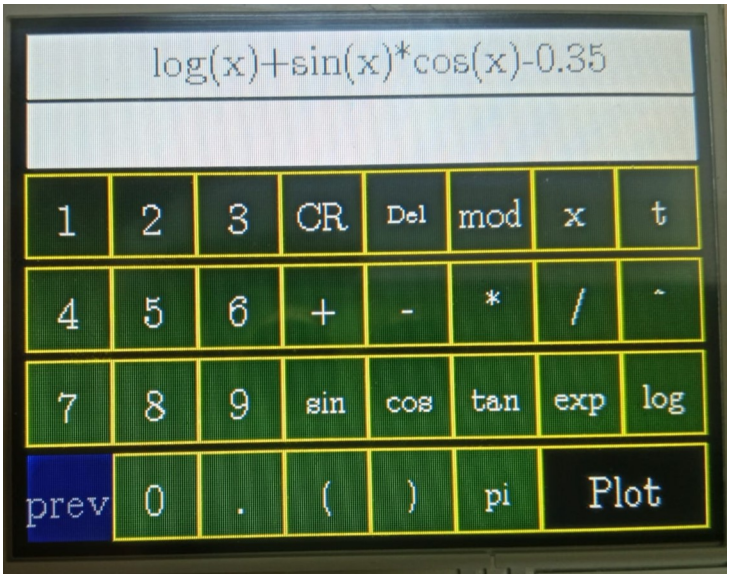

Figure 1.1: Graphing calculator main panel.

1.2.1 Plotting graph

Function $f(x)$ is plotted by the calculator as shown in figure 1.2. The value of $log(x)$ is not defined for negative values of x, thus, we see no plot for the same. By default, the function entered by the user is plotted within the range of -10 to 10. User can change this default range by changing the definition of gloabl variables `__X_MIN__` (starting value of x) and `__X_MAX__` (last value of x) in calculator.c file.

Figure 1.2: Graphing calculator graphing screen 1/2.

1.2.2 Derivative Calculator

To calculate the derivative curve of $f(x)$, the user needs to tap on the "f'(x)" option, which is given at the bottom of the screen as shown in figure 1.2. This will result in the plotting of the first derivative of $f(x)$, which can be seen in the figure 1.3.

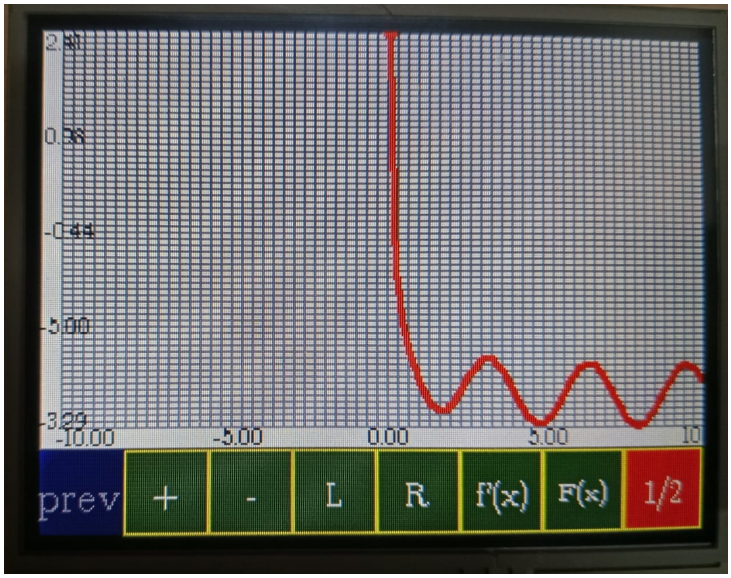

Figure 1.3: Derivative curve of $f(x)$ plotted by graphing calculator.

1.2.3 Integral Calculator

To calculate the integral curve of $f(x)$, the user needs to tap on the "F(x)" option, which is given at the bottom of the screen as shown in figure 1.2. This will result in the plotting of the curve defined by equation 1.1, which can be seen in the figure 1.4.

$$F(x) = \int_{\_\_X\_MIN\_\_}^{x} f(p)dp \tag{1.1}$$

1.2.4 Zoom in and out

To zoom into the curve, the user needs to tap on the "+" option, which is given at the bottom of the screen as shown in figure 1.2. This will result in decrement of both `__X_MIN__` and `__X_MAX__` by factor of `ZOOM_FACTOR`, thereafter, the graph is plotted as shown in figure 1.5(a).

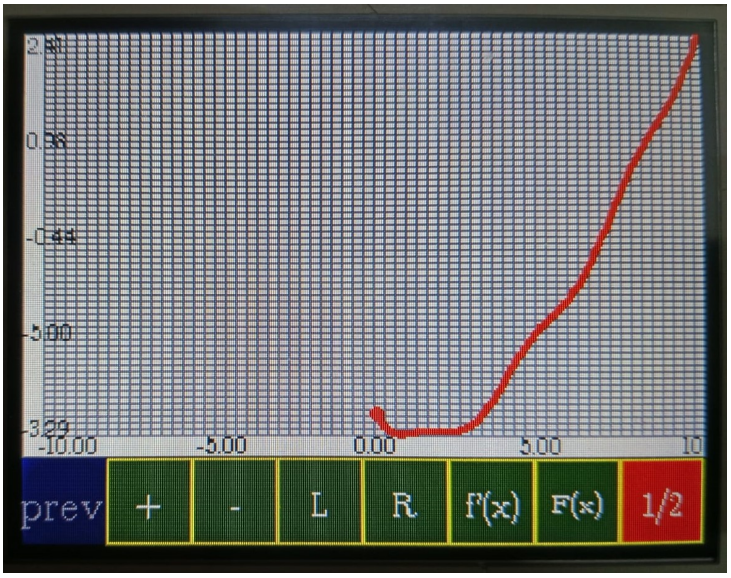

Figure 1.4: $F(x)$ curve plotted by graphing calculator.

To zoom out of the curve, the user needs to tap on the ”-” option, which is given at the bottom of the screen as shown in figure 1.2. This will result in increment of both `__X_MIN__` and `__X_MAX__` by factor of `ZOOM_FACTOR`, thereafter, the graph is plotted as shown in figure 1.5(b).

The default value of macro `ZOOM_FACTOR` is 2. This can be changed in `evaluate.h` header file.

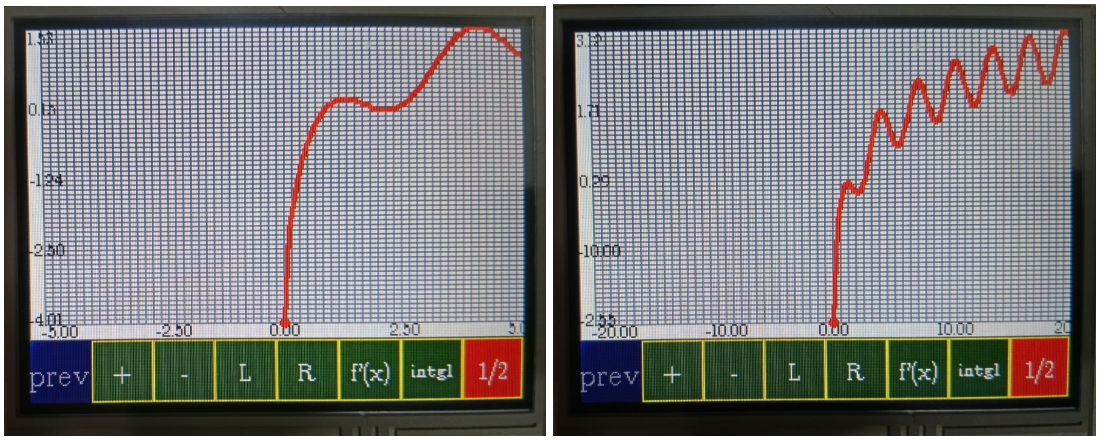

Figure 1.5: (a) Zoom-in plot of $f(x)$. (b) Zoom-out plot of $f(x)$.

1.2.5 Shift left and right

To right-shift the curve, the user needs to tap on the ”R” option, which is given at the bottom of the screen as shown in figure 1.2. This will result in decrement of both `__X_MIN__` and

__X_MAX__ by of (__X_MAX__ − __X_MIN__) /SHIFT_FACTOR, thereafter, the graph is plotted as shown in figure 1.6(a).

To zoom out of the curve, the user needs to tap on the "-" option, which is given at the bottom of the screen as shown in figure 1.2. This will result in increment of both __X_MIN__ and __X_MAX__ by of (__X_MAX__ − __X_MIN__) /SHIFT_FACTOR, thereafter, the graph is plotted as shown in figure 1.6(b).

The default value of macro SHIFT_FACTOR is 10. This can be changed in evaluate.h header file.

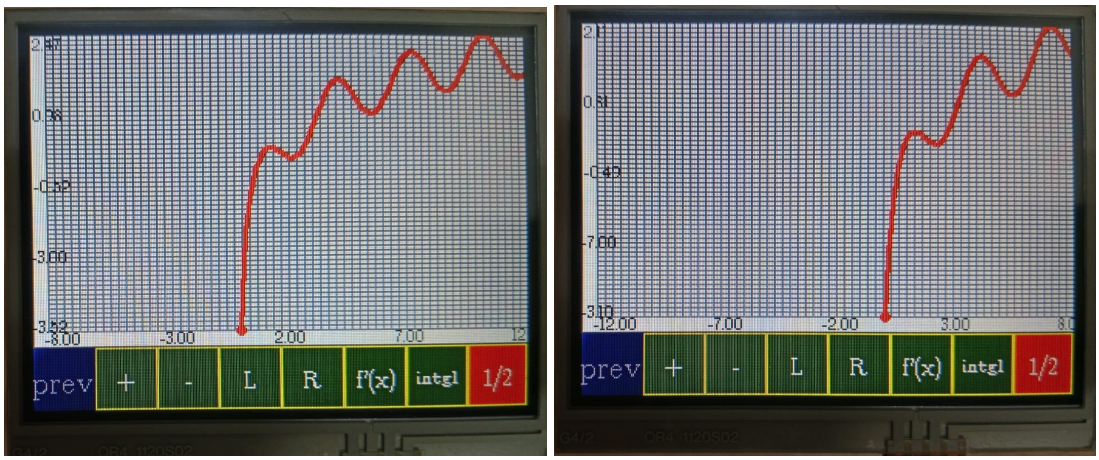

Figure 1.6: (a) Right shift plot of $f(x)$. (b) Left shift plot of $f(x)$.

1.2.6 Area under the curve

To calculate the area under the curve, which is defined by the equation 1.2. Users need to go to the second section of the panel by tapping on the "1/2" button of the screen as shown in figure 1.2. This will result in the screen shown in figure 1.7, thereafter, the user needs to tap on the "A" button which will display the area under the curve value as shown in figure 1.8

$$A = \int_{\_\_X\_MIN\_\_}^{\_\_X\_MAX\_\_} f(x)dx \tag{1.2}$$

1.2.7 Zeros of the function

The graphing calculator can calculate the odd roots of the curve within the given range of the curve. To calculate the roots of the curve. Users need to go to the second section of the panel by

Figure 1.7: Graphing calculator graphing screen 2/2.

Figure 1.8: Area under the curve calculated by graphing calculator.

tapping on the ”1/2” button of the screen as shown in figure 1.2. This will result in the screen shown in figure 1.7; thereafter, the user needs to tap on the ”Roots” button, which will highlight the roots of the curve with the values as shown in figure 1.9(a). To hide the values of the roots, the user can tap on the ”Hide” button, which will remove the values of the highlighted roots as shown in the figure 1.9(b).

The maximum number of roots that the graphing calculator is limited by macro MAX_ZEROS, whose default value is 10. When there are more than 10 roots, the calculator will only calculate

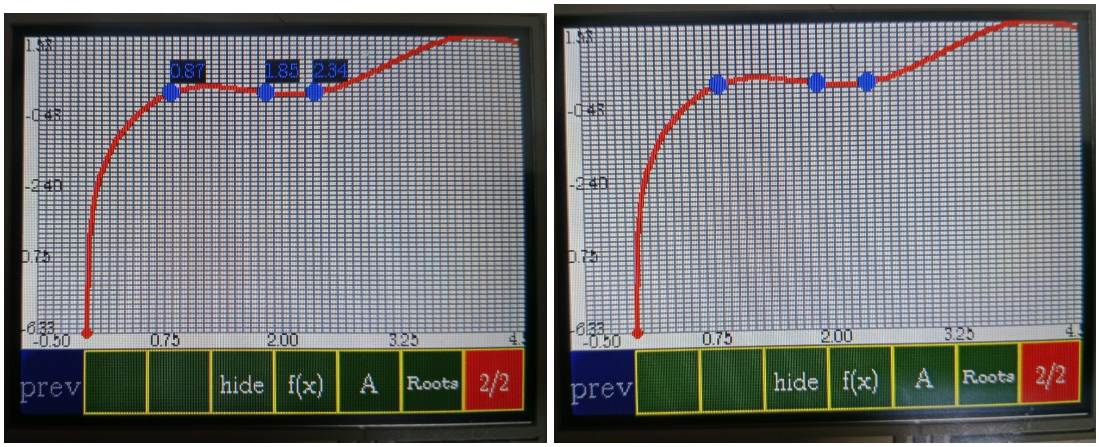

Figure 1.9: Odd roots of $f(x)$ (a) highlighted by graphing calculator with value and (b) without value.

the first 10 roots. The value of MAX_ZEROS cal is changed in `evaluate.h` header file.

Chapter 2

Working

2.1 Hardware

We are using here the BOOSTXL-K350QVG-S1 Kentec QVGA module [2] which is capable of driving a 320 x 240 pixel TFT LCD panel.

Integrating the BOOSTXL-K350QVG-S1 Kentec QVGA with TIVA microcontroller involves:

1. Configuring the interface (8-bit or 16-bit) depending on the host controller capabilities.
2. Initializing the SSD2119 controller to set up the screen resolution, color depth, and other features like scrolling and partial display modes.
3. Require to remove the resistor `R9` and `R10` on TIVA microcontroller and then connecting to through the pins in proper alignment pin configurations `<J2,J4>` and `<J1,J3>`.

The following are the resource requirements to build the system:

- BOOSTXL-K350QVG-S1 Kentec QVGA Touch Screen Graphics Display.
- TIVA C Series Microcontroller (TM4C123GH6PM)
- Code Composer Studio IDE.
- GCC

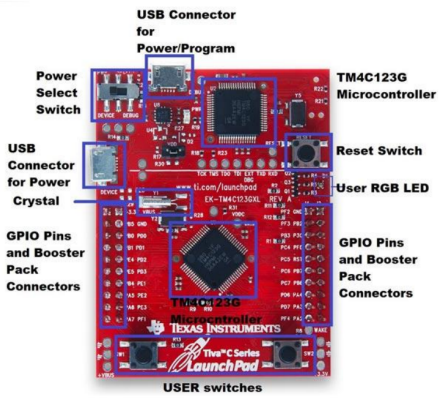

Figure 2.1: TIVA C series Launchpad Board

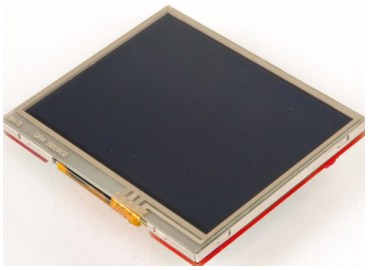

Figure 2.2: Kentec QVGA Touch Screen

2.2 Code Tree

The project is organized into directories and files with specific roles as outlined below:

- `Initials.h` - Contains initial setup functions for the calculator.
- `Panels.h` - Declarations related to different touchscreen panels.
- `Panel_6.h` - Interface functions for the mathematical expression input panel.
- `Panel_6_2.h` - Interface for the plotting panel.
- `Calculator.c` - The main entry point of the calculator program.
- `OnButtonPress_6.c` - Handles button presses in the expression input interface.
- `OnButtonPress_6_2.c` - Manages the plotting interface functionality.
- `ButtonPress_Plot.c` - Implements zooming, shifting, differentiation, integration, and other plotting features.

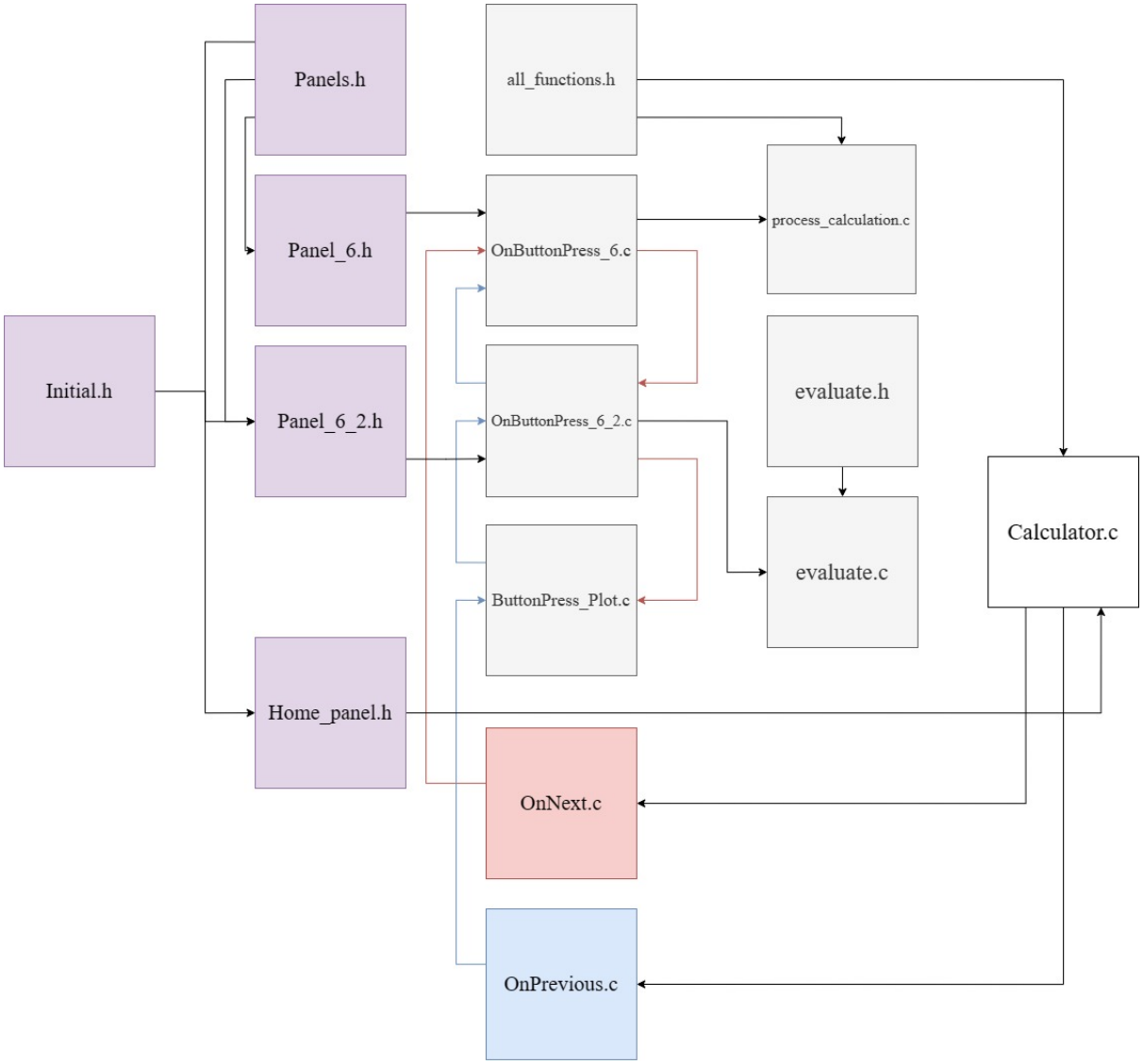

Figure 2.3: Code Hierarchy Tree

- `process_calculation.c` - Processes and evaluates mathematical expressions.
- `evaluate.c` - Converts expressions to pixel coordinate arrays for plotting.
- `startup_css` - Startup file.
- **drivers/** - `Kentec320x240x16_ssd2119_spi.c` and `touch.c` driver files to run touchscreen module specific libraries.

2.3 Evaluating the expression

The `eval()` function which is part of `evaluate.h` is used to evaluate a mathematical expression given as a string `expr`. It supports basic arithmetic operations ($+$, $-$, $*$, $/$), trigonometric functions (`sin`, `cos`, `tan`), logarithmic function (`log`), and modulus (`mod`) operation. The function evaluates the expression and returns the result as a float.

The function works on the principle of the shunting-yard algorithm, which is used to parse mathematical expressions specified in infix notation into postfix notation (also known as Reverse Polish Notation or RPN)[3]. Once the expression is in postfix notation, it is easier to evaluate using a stack.

The algorithm used in this function can be broken down into the following steps:

- Initialize two stacks, one for operands (`valStack`) and one for operators (`opStack`).
- Iterate through each character in the input expression.
- If the character is a space, skip it.
- If the character is a digit or a decimal point followed by a digit, extract the number and push it onto the operand stack (`valStack`).
- If the character is an alphabetic character (i.e., the start of a function name), check for supported functions (`sin`, `cos`, `tan`, `log`, `mod`) and push their corresponding tokens onto the operator stack (`opStack`).
- If the character is an opening parenthesis, push it onto the operator stack (`opStack`).
- If the character is a closing parenthesis, pop operators and operands from the stacks and perform the corresponding operation until an opening parenthesis is encountered. Push the result back onto the operand stack (`valStack`).
- If the character is an operator ($+$, $-$, $*$, $/$, etc.), pop operators from the stack as long as they have higher or equal precedence compared to the current operator, and then push the current operator onto the operator stack (`opStack`).
- After processing all characters, apply any remaining operators on the stacks to the operands to obtain the final result.

Overall, the function uses stacks to manage operators and operands, applying them in the correct order based on their precedence and associativity to evaluate the expression.

2.4 Plotting graph

Plotting graph on touch LCD involves two steps. First, the calculation of the points in the curve. Second, mapping them to the correct location on the LCD screen.

2.4.1 Calculation of points

The `__XY__` array is the global variable that will store the points of a curve. It is calculated in the following steps:

1. `x_vals()`: This function generates the x values for the expression. It calculates the step size between x values based on the minimum and maximum x values (`__X_MIN__` and `__X_MAX__`) and the total number of points (`N`). It then iterates over the `__XY__` array and fills in the x values.

2. `bracket_adder()`: This function adds brackets to the expression to ensure the correct order of operations. It handles cases where the expression contains trigonometric functions (`sin`, `cos`, `tan`), logarithmic function (`log`), and modulus (`mod`) operation. It also handles negative numbers in parentheses by replacing `(-` with `(0-`.

3. `val_replacer(float val)`: This function replaces the variable `x` in the expression with the given value `val`. It also replaces the constants `pi` and `e` with their respective values (3.14159265 and 2.71828182845).

4. `y_vals()`: This function calculates the y values corresponding to the x values. It first calls `bracket_adder()` to add brackets to the expression. Then, it iterates over the `__XY__` array, replacing the variable `x` with the current x value and evaluating the expression using the `eval()` function to calculate the corresponding y value.

5. `xy_vals()`: This function is a helper function that calls `x_vals()` and `y_vals()` to generate both x and y values for the expression.

Overall, the `__XY__` array is calculated by first generating the x values, then calculating the corresponding y values for each x value by evaluating the expression with the `x` variable replaced by the current x value.

The resolution of the curve depends on the value of the macro `N`. Default value of `N` is 800, which can be changed in `evaluate.h` header file.

2.4.2 Mapping to LCD screen

The function `map_xy()` is used to map a set of XY coordinates `__XY__` to a new set of coordinates `__MAPPED_XY__` based on certain scaling and offset values. The function operates as follows:

1. Calculate the range of X and Y values:
    - `x_range` is calculated as the difference between the maximum and minimum X values (`__X_MAX__` and `__X_MIN__`).
    - `y_range` is calculated as the difference between the maximum and minimum Y values (`__Y_MAX__` and `__Y_MIN__`).
2. Calculate the scaling factors:
    - `x_scale` is calculated as the difference between the maximum and minimum X values in the mapped coordinates (`__MP_X2__` and `__MP_X1__`), divided by the range of X values.
    - `y_scale` is calculated as the difference between the maximum and minimum Y values in the mapped coordinates (`__MP_Y2__` and `__MP_Y1__`), divided by the range of Y values.
3. Calculate the offset values:
    - `x_offset` is calculated as the difference between the minimum X value in the mapped coordinates (`__MP_X1__`) and the product of `x_scale` and the minimum X value from the original coordinates (`__X_MIN__`).
    - `y_offset` is calculated as the difference between the minimum Y value in the mapped coordinates (`__MP_Y1__`) and the product of `y_scale` and the minimum Y value from the original coordinates (`__Y_MIN__`).
4. Iterate over each pair of X and Y values in the input array `__XY__`:
    - Calculate the mapped X value by multiplying the X value by `x_scale` and adding `x_offset`.
    - Calculate the mapped Y value by multiplying the Y value by `y_scale` and adding `y_offset`.
    - Store the mapped X and Y values in the corresponding arrays in `__MAPPED_XY__`.

In summary, the `map_xy()` function performs a linear mapping of XY coordinates from one range to another, based on the specified mapping ranges and offsets.

2.5 Derivative Calculator

The `derivative()` function (which is part of `evaluate.h`) computes an approximation of the derivative of a set of XY coordinates stored in the array `__XY__`, and stores the results in the array `__DY_DX__`. Additionally, it calculates the minimum and maximum values of the derivative and stores them in `__Y_MIN_DY_DX__` and `__Y_MAX_DY_DX__` respectively.

The function operates as follows:

1. Iterate over XY coordinates: The function iterates over each pair of XY coordinates in the input array `__XY__`, except for the last pair ($N-1$), as the derivative calculation requires a subsequent point.
2. Calculate midpoint and derivative approximation: For each pair of XY coordinates, it calculates:
   - The midpoint X value: $(\texttt{__XY__[0][i + 1]} + \texttt{__XY__[0][i]})/2$
   - The derivative approximation: $(\texttt{__XY__[1][i + 1]} - \texttt{__XY__[1][i]})/(\texttt{__XY__[0][i + 1]} - \texttt{__XY__[0][i]})$
3. Calculate minimum and maximum derivative: After the loop, the function calculates the minimum and maximum values of the derivative in the `__DY_DX__[1]` array (which contains the derivative values) and stores them in `__Y_MIN_DY_DX__` and `__Y_MAX_DY_DX__` respectively, using the `min()` and `max()` functions.

In summary, the `derivative()` function computes an approximation of the derivative of a set of XY coordinates and calculates the minimum and maximum values of the derivative. Thereafter, `__DY_DX__` can be mapped to LCD screen using `map_dydx()` function.

2.6 Integral Calculator

The `integral()` (Which is part of `evaluate.h`) function computes an approximation of the integral of a set of XY coordinates stored in the array `__XY__`, and stores the results in the

array __INTEGRAL_XY__. Additionally, it calculates the minimum and maximum values of the integral and stores them in __Y_MIN_INTEGRAL__ and __Y_MAX_INTEGRAL__ respectively.

The function operates as follows:

1. Initialize last valid value: It initializes a variable `last_valid_value` to 0, which will be used to store the last valid integral value.

2. Iterate over XY coordinates: The function iterates over each pair of XY coordinates in the input array __XY__, except for the last pair ($N-1$), as the integral calculation requires a subsequent point.

3. Check for invalid values: For each pair of XY coordinates, it checks if either of the Y values is infinite or NaN (not a number). If so, it sets the integral value to NaN and continues to the next iteration.

4. Calculate midpoint and integral approximation: For each pair of XY coordinates with valid values, it calculates:

   - The midpoint X value: $(\texttt{\_\_XY\_\_[0][i + 1]} + \texttt{\_\_XY\_\_[0][i]})/2$
   - The integral approximation using the trapezoidal rule: $\texttt{last\_valid\_value} + (\texttt{\_\_XY\_\_[1][i]} + \texttt{\_\_XY\_\_[1][i + 1]}) \times (\texttt{\_\_XY\_\_[0][i + 1]} - \texttt{\_\_XY\_\_[0][i]})/2$

   The calculated integral value is stored in __INTEGRAL_XY__[1][i], and `last_valid_value` is updated to this value.

5. Calculate minimum and maximum integral: After the loop, the function calculates the minimum and maximum values of the integral in the __INTEGRAL_XY__[1] array (which contains the integral values) and stores them in __Y_MIN_INTEGRAL__ and __Y_MAX_INTEGRAL__ respectively, using the `min()` and `max()` functions.

Thus, the `integral()` function computes an approximation of the integral of a set of XY coordinates using the trapezoidal rule, handling invalid values, and calculates the minimum and maximum values of the integral. Thereafter, __INTEGRAL_XY__ can be mapped to LCD screen using `map_integral()` function.

2.7 Area Under the curve

The `area_under_curve()` (Which is part of `evaluate.h`) function computes the area under a curve defined by a set of XY coordinates stored in the array `__XY__`, and stores the result in the variable `__AREA__`.

The function operates as follows:

1. Initialize area and last valid value: It initializes the area `__AREA__` to 0 and a variable `last_valid_value` to 0, which will be used to store the last valid area value.

2. Iterate over XY coordinates: The function iterates over each pair of XY coordinates in the input array `__XY__`, except for the last pair ($N-1$), as the area calculation requires a subsequent point.

3. Check for invalid values: For each pair of XY coordinates, it checks if either of the Y values is infinite or NaN (not a number). If so, it skips the calculation for that pair and continues to the next iteration.

4. Calculate area approximation: For each pair of XY coordinates with valid values, it calculates the area under the curve using the trapezoidal rule:

   ```
   __AREA__ = last_valid_value + (__XY__[1][i] + __XY__[1][i + 1]) *
   (__XY__[0][i + 1] - __XY__[0][i]) / 2
   ```

   The calculated area value is stored in `__AREA__`, and `last_valid_value` is updated to this value.

In summary, the `area_under_curve()` function computes an approximation of the area under a curve defined by a set of XY coordinates using the trapezoidal rule, handling invalid values.

2.8 Zeros of the function

To calculate the odd roots of a curve within a given range, the following functions (Which are part of `evaluate.h`) are used:

1. `bisection_points()`: This function calculates the bisection points of a function represented by XY coordinates stored in the array `__XY__`. It populates the array `__BISECTION__`

with the bisection points, where each bisection point consists of an X and Y coordinate. Bisection points are the points where the Y values of two consecutive XY coordinates have opposite signs, indicating a change in sign of the function and potentially the presence of a root within that range.

2. `bisection_method(float point1[1][2], float point2[1][2])`: This function calculates the root of the function using the bisection method, given two bisection points. It first checks if the two points have opposite signs (indicating a potential root between them). If they do, it iteratively refines the root approximation using the bisection method until the desired accuracy is achieved or the maximum number of iterations is reached. It returns the calculated root or `NAN` if the root is not found.

3. `zeros_of_function()`: This function finds the zeros of the function within the given range. It first calls `bisection_points()` to find the bisection points. Then, it initializes two stacks to store the zeros (`__ZEROS__`) and the mapped zeros (`__MAPPED_ZEROS__`). It iterates over the bisection points, and for each pair of consecutive bisection points, it calls `bisection_method()` to calculate the root. If a root is found, it pushes it onto the `__ZEROS__` stack.

Overall, these functions work together to find the odd roots of a curve within a given range by first identifying the bisection points and then using the bisection method to calculate the roots.

2.9 Zoom in and out

Following functions (Which is part of `evaluate.h`) are used to implement zoom in and out functionality:-

`zoom_in()`: This function zooms in on the curve by reducing the range of X values displayed on the graph. It divides the current minimum and maximum X values (`__X_MIN__` and `__X_MAX__`) by the zoom factor (`ZOOM_FACTOR`). After adjusting the X range, it recalculates the XY values, maps the XY values to the display, computes the derivative, maps the derivative, calculates the integral, and finally maps the integral values.

`zoom_out()`: This function zooms out on the curve by increasing the range of X values displayed on the graph. It multiplies the current minimum and maximum X values (`__X_MIN__` and `__X_MAX__`) by the zoom factor (`ZOOM_FACTOR`). Similar to `zoom_in()`, it then recalculates the XY values, maps the XY values to the display, computes the derivative, maps the derivative, calculates the integral, and finally maps the integral values.

2.10 Left and right shift

Following functions (Which is part of `evaluate.h`) are used to implement shifting right and left functionality:-

`shift_right()`: This function shifts the curve to the right by increasing the range of X values displayed on the graph. It calculates the current range of X values (`__X_MAX__` - `__X_MIN__`) and then increases both the minimum and maximum X values by a fraction of the range (`x_range / SHIFT_FACTOR`). After adjusting the X range, it recalculates the XY values, maps the XY values to the display, computes the derivative, maps the derivative, calculates the integral, and finally maps the integral values.

`shift_left()`: This function shifts the curve to the left by decreasing the range of X values displayed on the graph. It calculates the current range of X values (`__X_MAX__` - `__X_MIN__`) and then decreases both the minimum and maximum X values by a fraction of the range (`x_range / SHIFT_FACTOR`). Similar to `shift_right()`, it then recalculates the XY values, maps the XY values to the display, computes the derivative, maps the derivative, calculates the integral, and finally maps the integral values.

Chapter 3

Concluding remarks

3.1 User instructions

The Scientific Graphing Calculator using Touch LCD is a revolutionary tool that allows users to visualize complex mathematical functions easily. It is designed to be intuitive, interactive, and portable, making it ideal for educational and professional use.

The calculator offers several features:

- Plotting Graphs: Users can enter mathematical expressions and plot graphs with a simple tap on the ”Plot” option.
- Derivative Calculator: Calculate the derivative curve of a function by tapping on the ”f’(x)” option.
- Integral Calculator: Calculate the integral curve of a function by tapping on the ”F(x)” option.
- Zoom In and Out: Zoom into or out of the graph by tapping on the ”+” or ”-” option, respectively.
- Shift Left and Right: Shift the graph left or right by tapping on the ”R” or ”L” option, respectively.
- Area Under the Curve: Calculate the area under the curve of a function by tapping on the ”A” option.

- Zeros of the Function: Calculate the odd roots of a function within a given range by tapping on the ”Roots” option.

To use the calculator, follow these steps:

1. Enter a mathematical expression in the main panel.
2. Tap on the ”Plot” option to plot the graph.
3. Explore different features like derivative, integral, zoom, shift, area, and zeros.

Users can customize the default range of the graph and other parameters by editing the macros in the header files.

3.2 Suggestions for next gen

The development of the calculator has significantly enhanced its functionality, providing users with powerful tools for mathematical analysis. However, to further improve its utility for future generations, several key enhancements can be implemented:

1. **Enable calculator to find even roots**: Implement algorithms and functions that allow the calculator to calculate even roots of functions accurately. This enhancement would expand the calculator's capability to handle a wider range of mathematical operations and make it more versatile.
2. **Enable calculator to handle vertical asymptotes**: Integrate features that enable the calculator to identify and handle vertical asymptotes in functions. This improvement would enhance the calculator's ability to analyze and graph functions with vertical asymptotes, providing more comprehensive results for users.

In conclusion, implementing these improvements would significantly enhance the calculator's functionality and make it a more powerful tool for mathematical analysis for future generations.

Appendix A

Data

Tables A.1, A.2, A.4, A.5, and A.6 gives the information of all the macros, global variables and functions in `evaluate.h` header file.

Table A.1: Macros in `evaluate.h` with Default Values

Macro	Usage	Read/Write	Default Value
MAX_STACK_SIZE	Maximum size of expression stack	Read/write	100
PI	Value of pi	Read	3.14159265
E	Value of e	Read	2.718281828459045
EPSILON	Value of epsilon - Bisection method	Read/write	0.000001
MAX_ITERATIONS	Maximum iterations for bisection method	Read/write	70
MAX_ZEROS	Maximum number of zeros	Read/write	10
N	Maximum number of points	Read/write	800
MIN_Y_SPACE	Minimum y space	Read/write	1
EXP_LEN	Maximum length of expression	Read/write	150
ZOOM_FACTOR	Zoom factor	Read/write	2
SHIFT_FACTOR	Shift factor	Read/write	10
mod(x)	Absolute value of x	Read	-

Table A.2: Global Variables in `evaluate.h`

Global Variable	Usage	Read/Write
`__EXPR__`	Expression to be evaluated f(x)	Write
`__EXPR_VAL__`	Expression to be evaluated f(number)	Read
`__BRACKET_FLAG__`	Flag to check if brackets are added	Read
`__X_MIN__`	Minimum x value	Write
`__X_MAX__`	Maximum x value	Write
`__Y_MIN__`	Minimum y value	Read
`__Y_MAX__`	Maximum y value	Read
`__Y_MIN_DY_DX__`	Minimum y value of dy/dx	Read
`__Y_MAX_DY_DX__`	Maximum y value of dy/dx	Read
`__Y_MIN_INTEGRAL__`	Minimum y value of integral of y	Read
`__Y_MAX_INTEGRAL__`	Maximum y value of integral of y	Read
`__MP_X1__`	x1 value for mapping	Write
`__MP_Y1__`	y1 value for mapping	Write
`__MP_X2__`	x2 value for mapping	Write
`__MP_Y2__`	y2 value for mapping	Write
`__XY__`	x values	Read
`__MAPPED_XY__`	x values mapped to screen coordinates	Read
`__DY_DX__`	derivative values	Read
`__MAPPED_DY_DX__`	mapped derivative values	Read
`__INTEGRAL_XY__`	integral values	Read
`__MAPPED_INTEGRAL_XY__`	mapped integral values	Read
`__AREA__`	area under the curve	Read
`__BISECTION__`	bisection points	Read
`__ZEROS__`	zeros of the function	Read
`__MAPPED_ZEROS__`	mapped zeros of the function	Read

Table A.3: Initialization Functions in `evaluate.h`

Function	Description
`initialize()`	Initialize global variables

Table A.4: Evaluation of Mathematical Expressions Functions in `evaluate.h`

Function	Description
`init(Stack* s)`	Initialize stack
`push(Stack* s, float val)`	Push item onto stack
`pop(Stack* s)`	Pop item from stack
`peek(Stack* s)`	Peek at the top item of the stack
`isempty(Stack* s)`	Check if stack is empty
`print_stack(Stack* s)`	Print stack
`precedence(char op)`	Operator precedence
`applyOp(float a, float b, char op)`	Apply operator to operands
`eval(const char* expr)`	Evaluate expression

Table A.5: Generation of X and Y Values Functions in `evaluate.h`

Function	Description
`bracket_adder()`	Envolopes sin, cos, tan, log, and mod in brackets
`x_vals()`	Generate x values
`val_replacer(float val)`	Replace variable with value
`y_vals()`	Generate y values corresponding to x values
`xy_vals()`	Generate x and y values

Table A.6: Mathematical Operations Functions in `evaluate.h`

Function	Description
`derivative()`	Calculate the derivative of a function
`integral()`	Calculate the integral of a function - Trapezoidal Rule - Integration from __X_MIN__ to x
`area_under_curve()`	Calculate the __AREA__ under the curve of a function
`bisection_points()`	Calculate the bisection points of a function
`bisection_method(float point1[1][2], float point2[1][2])`	Calculate the root using the bisection method with given points
`zeros_of_function()`	Calculate the zeros of a function
`map_xy()`	Map x and y values to screen coordinates
`map_dx_dy()`	Map dy/dx values to screen coordinates
`map_integral()`	Map integral y values to screen coordinates
`map_zeros()`	Map zeros of the function to screen coordinates
`zoom_in()`	Zoom in the graph
`zoom_out()`	Zoom out the graph
`shift_left()`	Shift the graph to the left
`shift_right()`	Shift the graph to the right

Appendix B

Program Listing

`evaluate.h` header file can be accessed from this GitHub link: https://bit.ly/evaluate-header

If one wishes to run the graphing calculator in their Tiva board directly, one can run the zip folder in Code Composition Studio. This zip folder is available at this link: https://bit.ly/graphing-calculator

Appendix C

Contributions

In the course of this project, a significant contribution was made through the development of the `evaluate.h` header file. This header file encapsulates a comprehensive set of functions for mathematical expression evaluation, graph plotting, and analysis. Some key contributions and novelties introduced in `evaluate.h` include:

- **Mathematical Expression Evaluation:** The `evaluate.h` header provides functions for evaluating mathematical expressions involving basic arithmetic operations, trigonometric functions, logarithmic functions, and more. It incorporates a stack-based approach for expression evaluation, ensuring efficiency and accuracy.
- **Graph Plotting and Analysis:** `evaluate.h` includes functions for generating x and y values, calculating derivatives, integrals, and areas under curves, and finding zeros of functions. These functionalities enable the user to analyze and visualize mathematical functions with ease.
- **Modularity and Reusability:** The design of `evaluate.h` emphasizes modularity and reusability. Each function is designed to perform a specific task, allowing users to easily integrate these functions into their projects for mathematical analysis and graph plotting.
- **Potential Intellectual Property:** The functionalities provided by `evaluate.h`, particularly the algorithms for expression evaluation and graph plotting, have the potential for intellectual property protection. The innovative approaches used in these algorithms may be considered novel and non-obvious.
- **Future Research and Publications:** The development of `evaluate.h` opens up avenues for future research and potential publications in conferences or journals. The algo-

rithms and methodologies used in `evaluate.h` can be further refined and extended to address more complex mathematical problems.

Overall, the development of `evaluate.h` represents a significant contribution to the project, providing a robust and versatile tool for mathematical expression evaluation and graph plotting. Its potential for intellectual property and future research highlights its importance in the project's success.

Bibliography

[1] Desmos studion PBC. Graphing calculator.

[2] Texas Instruments. Tiva c series tm4c129x microcontrollers silicon revisions 6 and 7 (spmu300e), 2022. Rev. E.

[3] GeeksforGeeks. Convert infix expression to postfix expression.