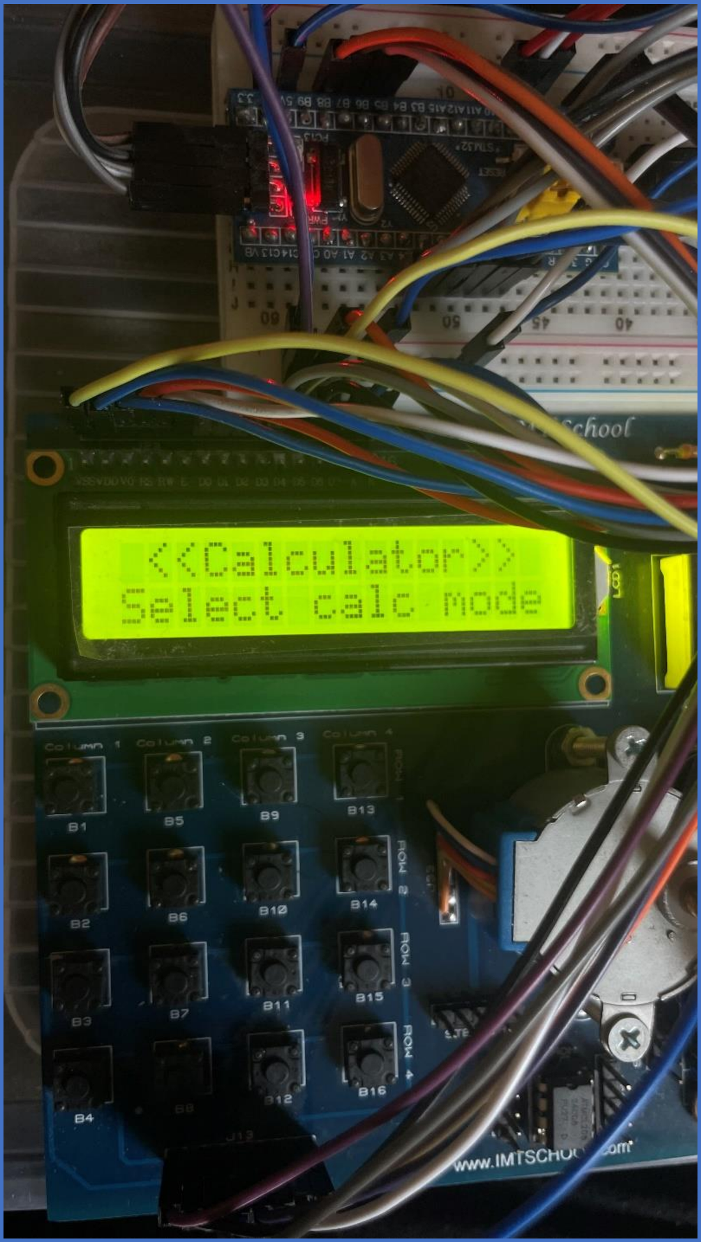
Calculator



Author: Eng./ Omar Mohamed Yamany

Contents

Case Study

A client expects software for a system with the following specifications:

- Software has a main menu that allows transition between system modes.
- Calculator mode that supports addition, subtraction, multiplication, and division operations on integer values.
- Division mode should protect the system from failing when dividing by zero.
- Numbering system mode that supports binary, decimal, octal, and hexadecimal numbering systems and allows conversion between each one and the other.

Assumptions:

- Calculator mode shall not support negative values, either as an input or as an output.
- Subtraction in calculator mode will always return the difference between the two numbers, or in other words, the absolute difference.
- Maximum number supported in Numbering system mode is 16-bit integer, in all numbering systems.
- User will not enter (A,B,C,D,E,F) in hexadecimal mode.

Methodology

Waterfall Method has been chosen for its simplicity.

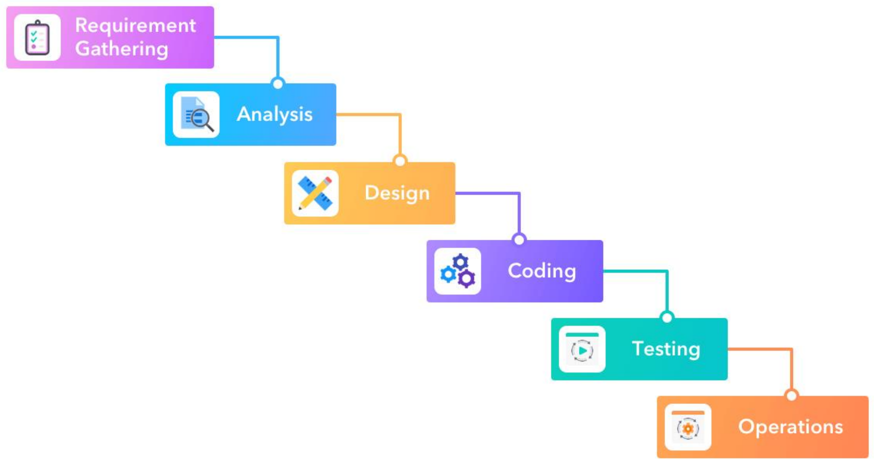


Fig (1) Waterfall Model

System Requirements

Requirement Diagram:

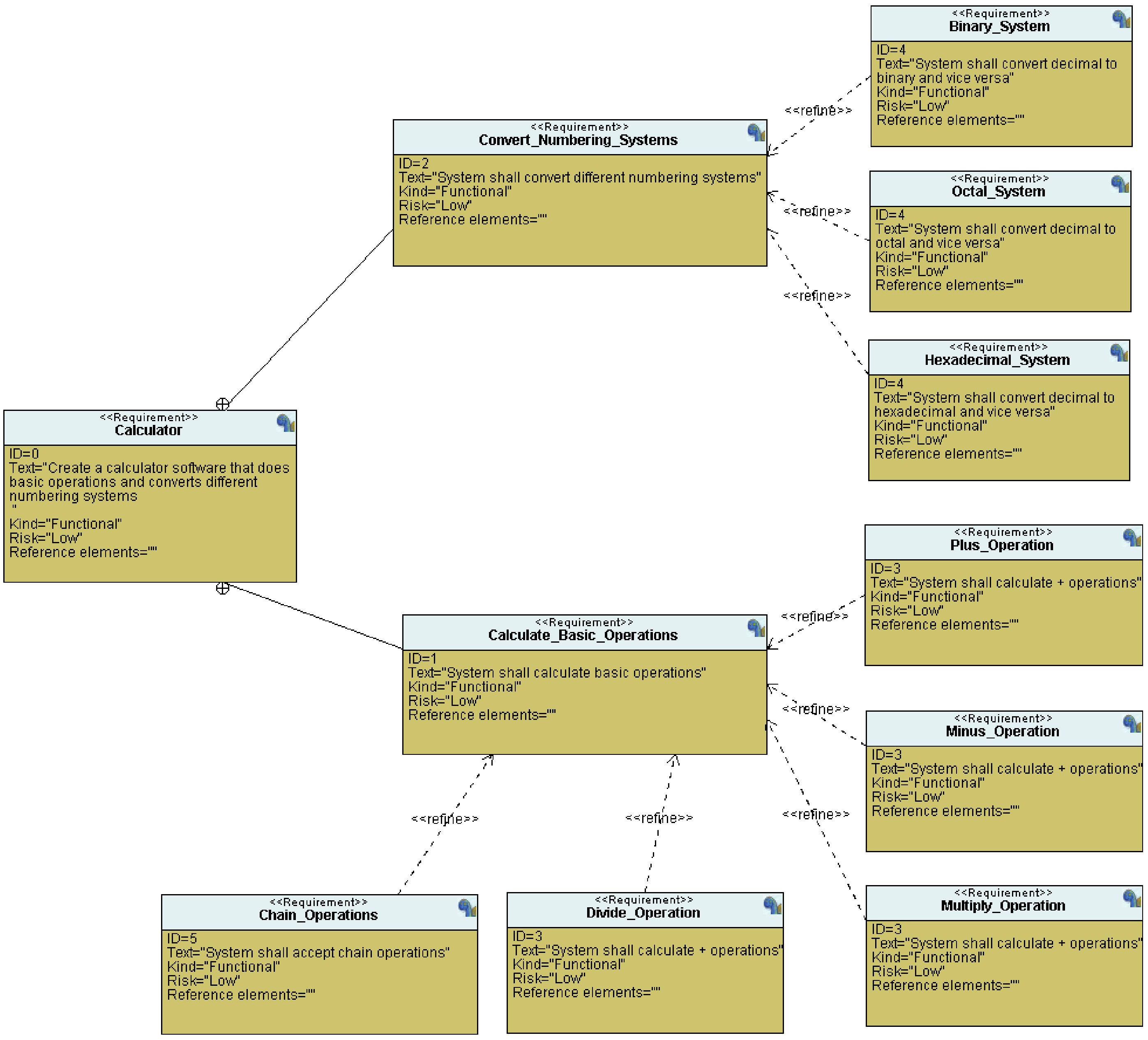


Fig (2) Requirement Diagram

Space Exploration

For the hardware, we have an STM32F106C8T6 microcontroller with a cortex-m3 processor that will be more than enough for this application. We will also use a 16x2 Graphical LCD and 4x4 Keypad to interface with the user.

System Analysis

Use Case

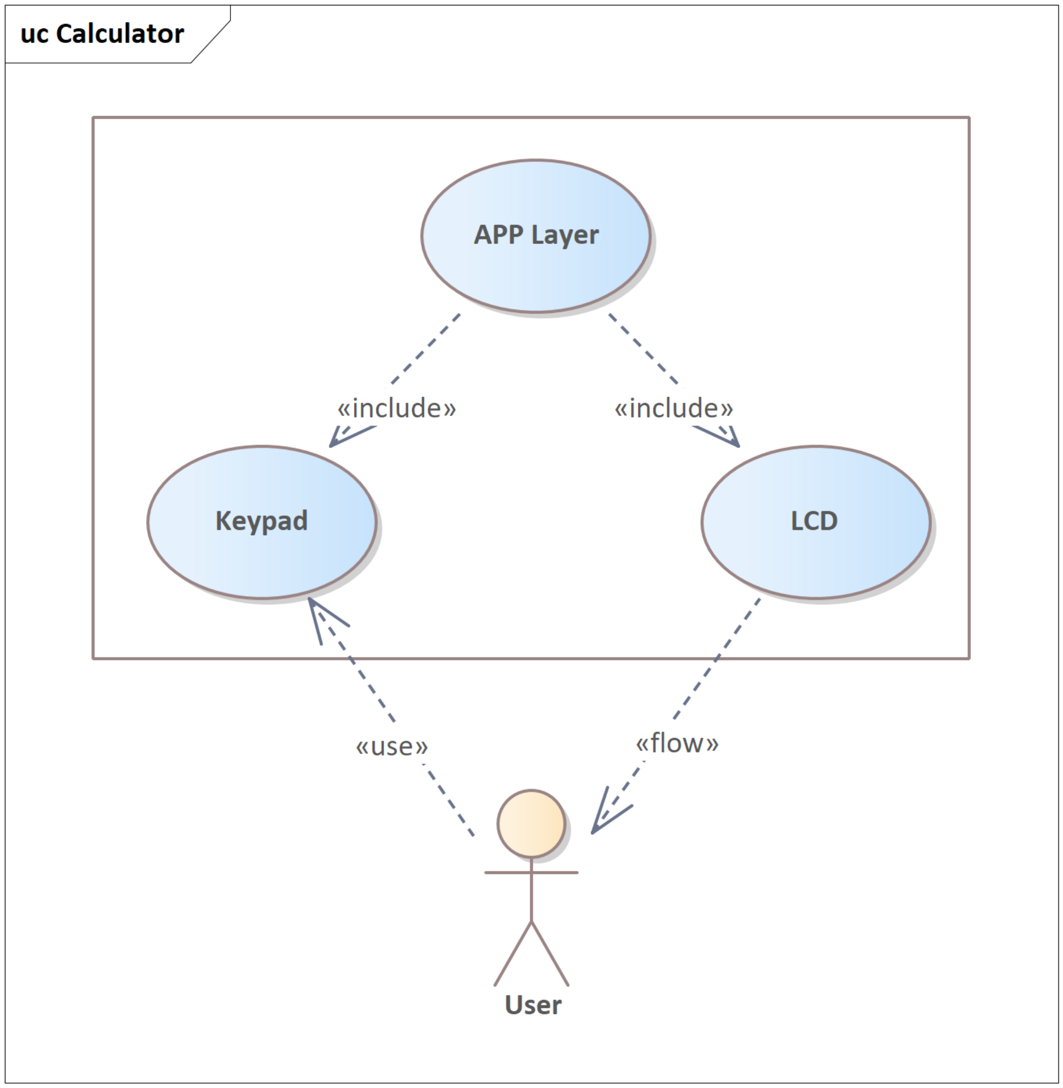


Fig (3) Use Case Diagram

Activity Diagram

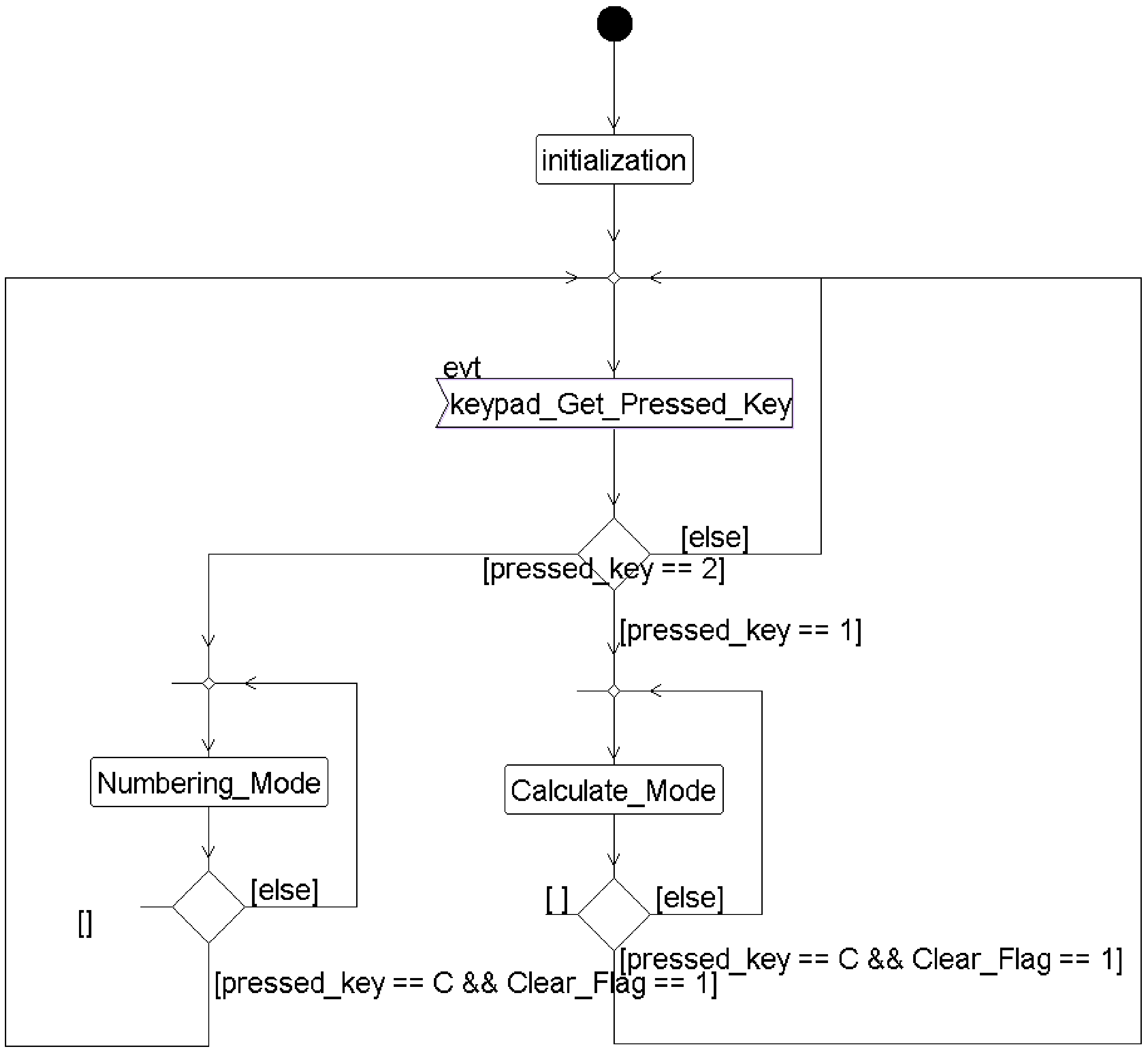


Fig (4) Activity Diagram

Sequence Diagram

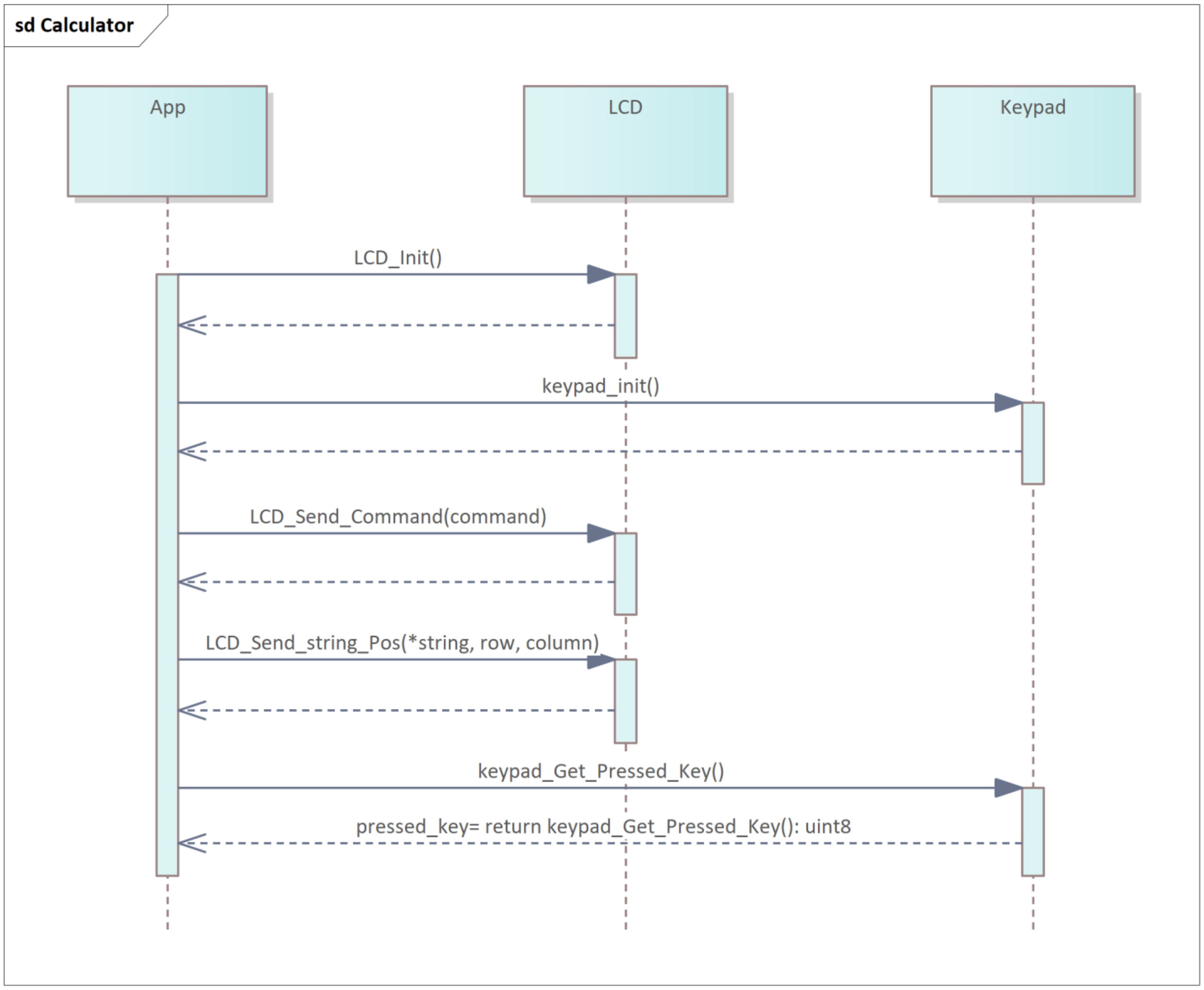

sd Calculator
App
LCD
Keypad
LCD_Init()
keypad_init()
LCD_Send_Command(command)
LCD_Send_string_Pos(*string, row, column)
keypad_Get_Pressed_Key()
pressed_key= return keypad_Get_Pressed_Key(): uint8

Fig (5) Sequence Diagram

System Design

Block Diagram

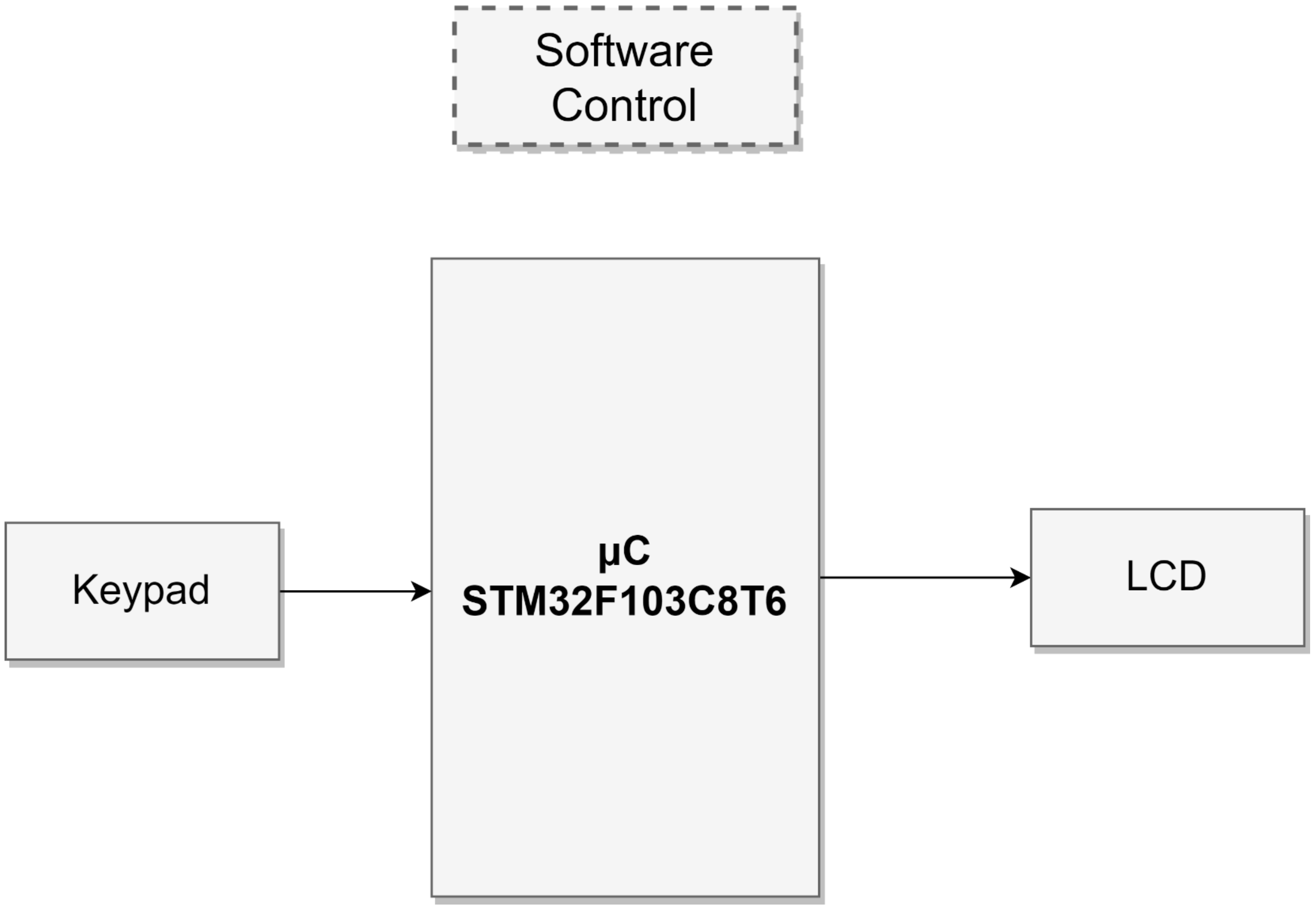


Fig (6) Block Diagram

Layered Architecture

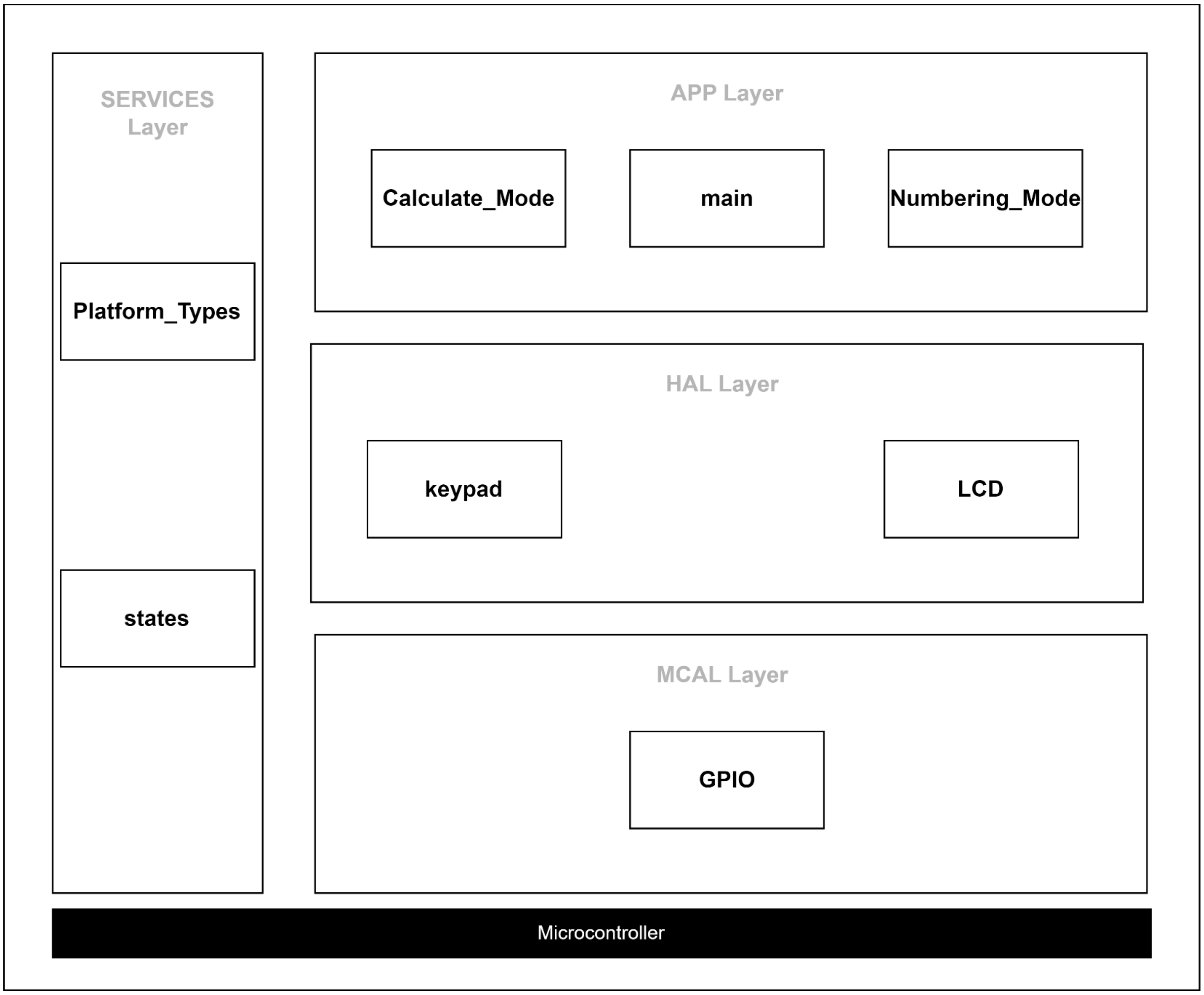


Fig (7) Layered Architecture

Class Diagram

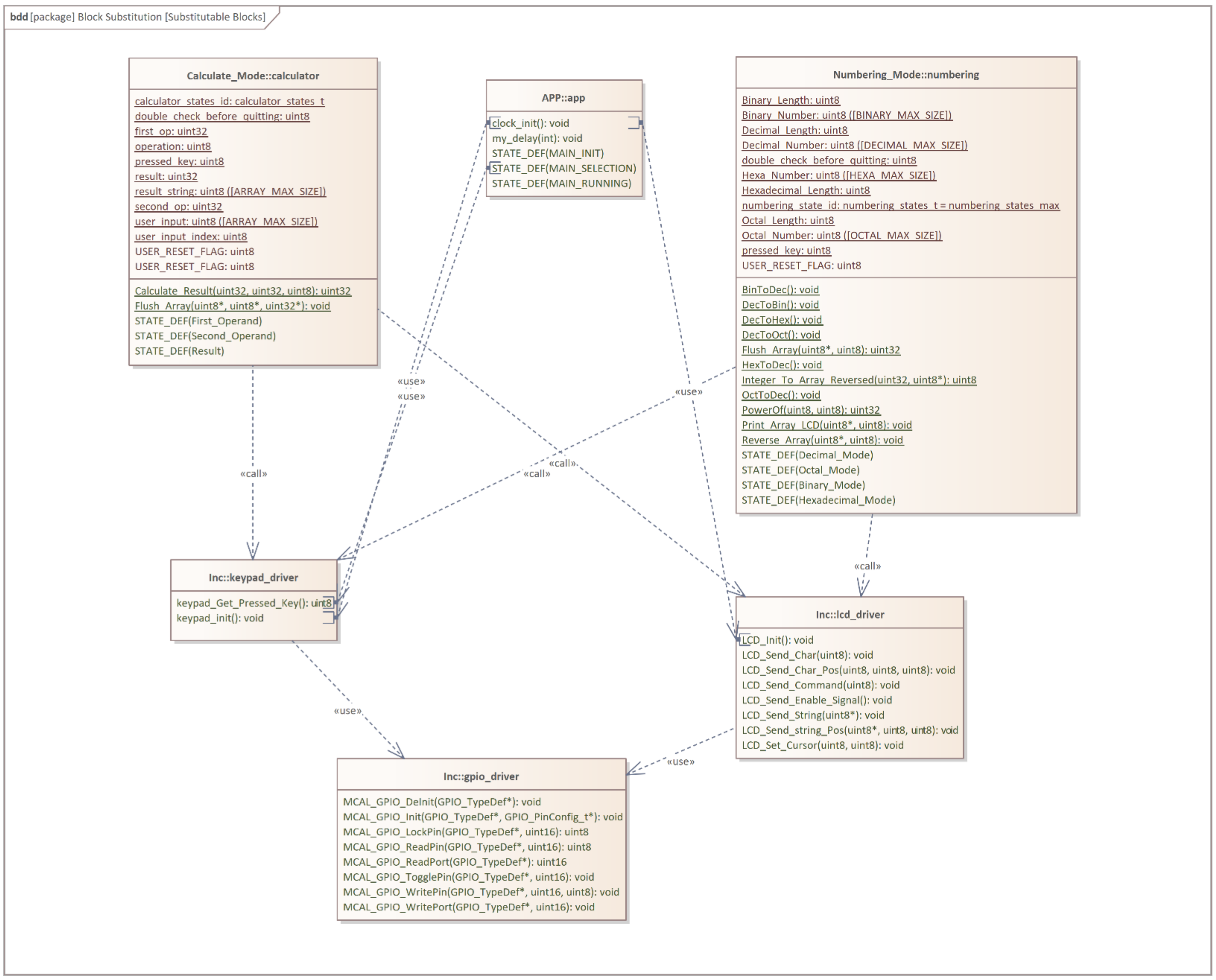

bdd [package] Block Substitution [Substitutable Blocks]
Calculate_Mode::calculator
calculator_states_id: calculator_states_t
double_check_before_quitting: uint8
first_op: uint32
operation: uint8
pressed_key: uint8
result: uint32
result_string: uint8 ([ARRAY_MAX_SIZE])
second_op: uint32
user_input: uint8 ([ARRAY_MAX_SIZE])
user_input_index: uint8
USER_RESET_FLAG: uint8
USER_RESET_FLAG: uint8
Calculate_Result(uint32, uint32, uint8): uint32
Flush_Array(uint8*, uint8*, uint32*): void
STATE_DEF(First_Operand)
STATE_DEF(Second_Operand)
STATE_DEF(Result)
APP::app
clock_init(): void
my_delay(int): void
STATE_DEF(MAIN_INIT)
STATE_DEF(MAIN_SELECTION)
STATE_DEF(MAIN_RUNNING)
Numbering_Mode::numbering
Binary_Length: uint8
Binary_Number: uint8 ([BINARY_MAX_SIZE])
Decimal_Length: uint8
Decimal_Number: uint8 ([DECIMAL_MAX_SIZE])
double_check_before_quitting: uint8
Hexa_Number: uint8 ([HEXA_MAX_SIZE])
Hexadecimal_Length: uint8
numbering_state_id: numbering_states_t = numbering_states_max
Octal_Length: uint8
Octal_Number: uint8 ([OCTAL_MAX_SIZE])
pressed_key: uint8
USER_RESET_FLAG: uint8
BinToDec(): void
DecToBin(): void
DecToHex(): void
DecToOct(): void
Flush_Array(uint8*, uint8): uint32
HexToDec(): void
Integer_To_Array_Reversed(uint32, uint8*): uint8
OctToDec(): void
PowerOf(uint8, uint8): uint32
Print_Array_LCD(uint8*, uint8): void
Reverse_Array(uint8*, uint8): void
STATE_DEF(Decimal_Mode)
STATE_DEF(Octal_Mode)
STATE_DEF(Binary_Mode)
STATE_DEF(Hexadecimal_Mode)
«use»
«use»
«use»
«call»
«call»
«call»
«call»
Inc::keypad_driver
keypad_Get_Pressed_Key(): uint8
keypad_init(): void
Inc::lcd_driver
LCD_Init(): void
LCD_Send_Char(uint8): void
LCD_Send_Char_Pos(uint8, uint8, uint8): void
LCD_Send_Command(uint8): void
LCD_Send_Enable_Signal(): void
LCD_Send_String(uint8*): void
LCD_Send_string_Pos(uint8*, uint8, uint8): void
LCD_Set_Cursor(uint8, uint8): void
«use»
«use»
Inc::gpio_driver
MCAL_GPIO_DeInit(GPIO_TypeDef*): void
MCAL_GPIO_Init(GPIO_TypeDef*, GPIO_PinConfig_t*): void
MCAL_GPIO_LockPin(GPIO_TypeDef*, uint16): uint8
MCAL_GPIO_ReadPin(GPIO_TypeDef*, uint16): uint8
MCAL_GPIO_ReadPort(GPIO_TypeDef*): uint16
MCAL_GPIO_TogglePin(GPIO_TypeDef*, uint16): void
MCAL_GPIO_WritePin(GPIO_TypeDef*, uint16, uint8): void
MCAL_GPIO_WritePort(GPIO_TypeDef*, uint16): void

Fig (8) Class Diagram

Main State Machine

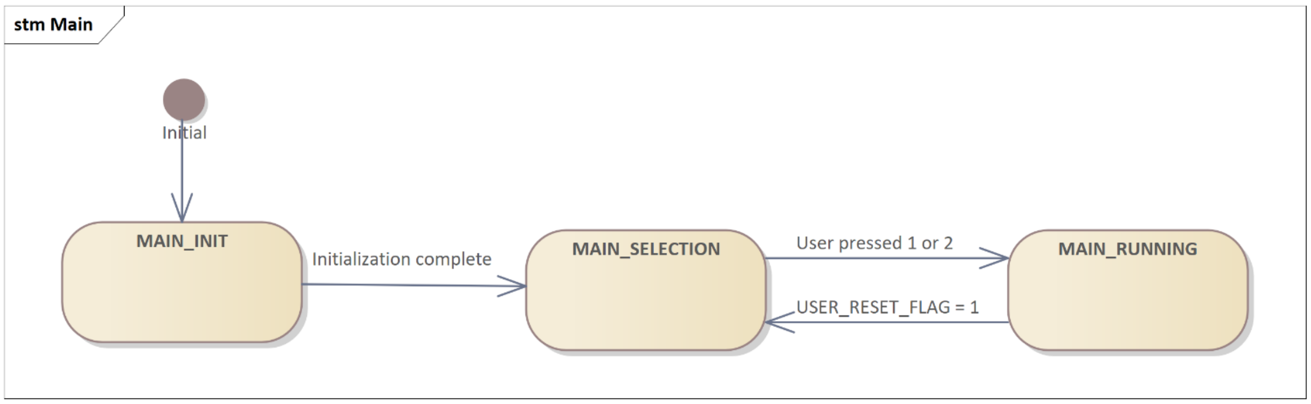


Fig (9) Main State Machine

Calculate_Mode State Machine

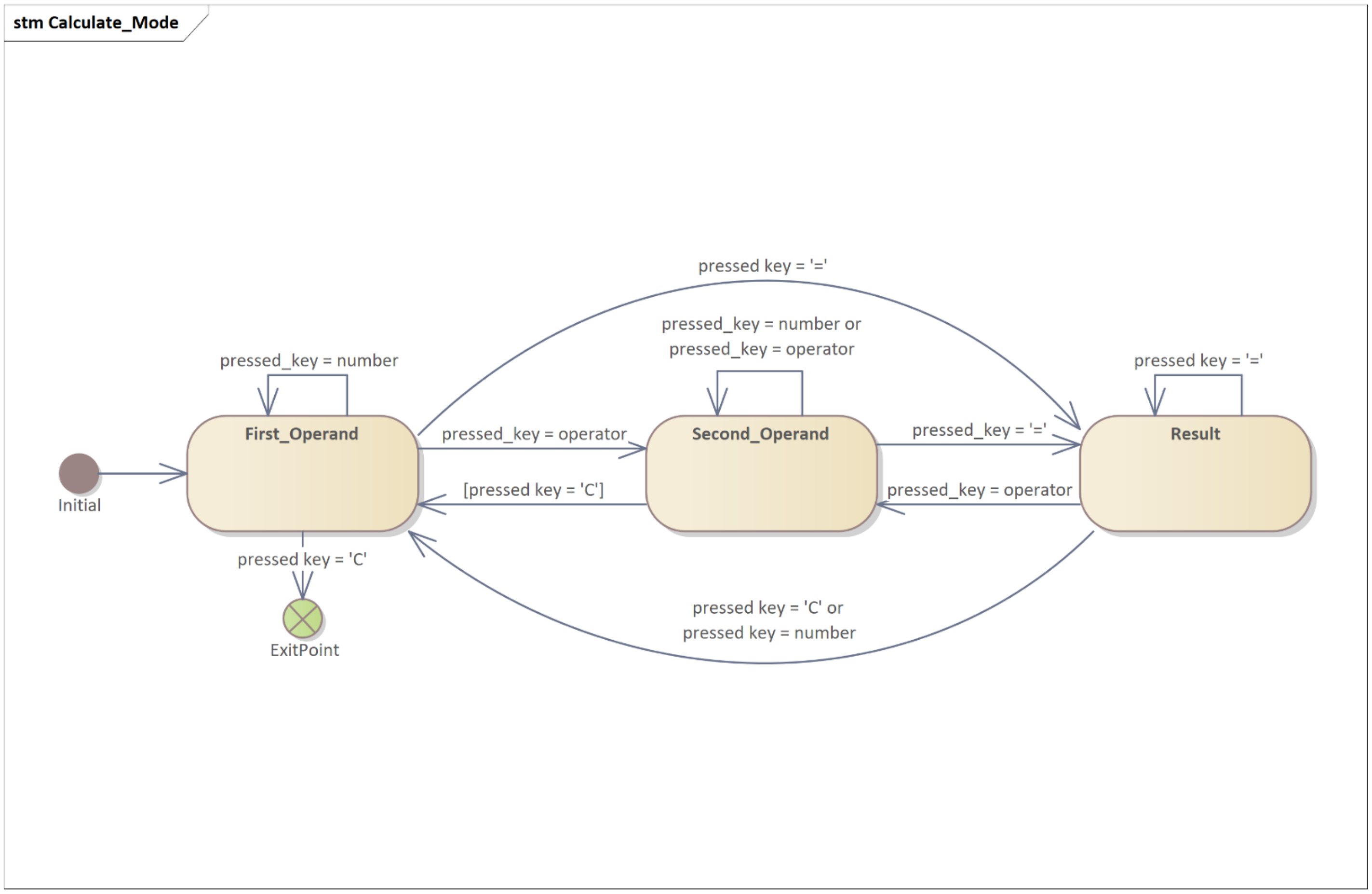


Fig (10) Calculate_Mode State Machine

Numbering_Mode State Machine

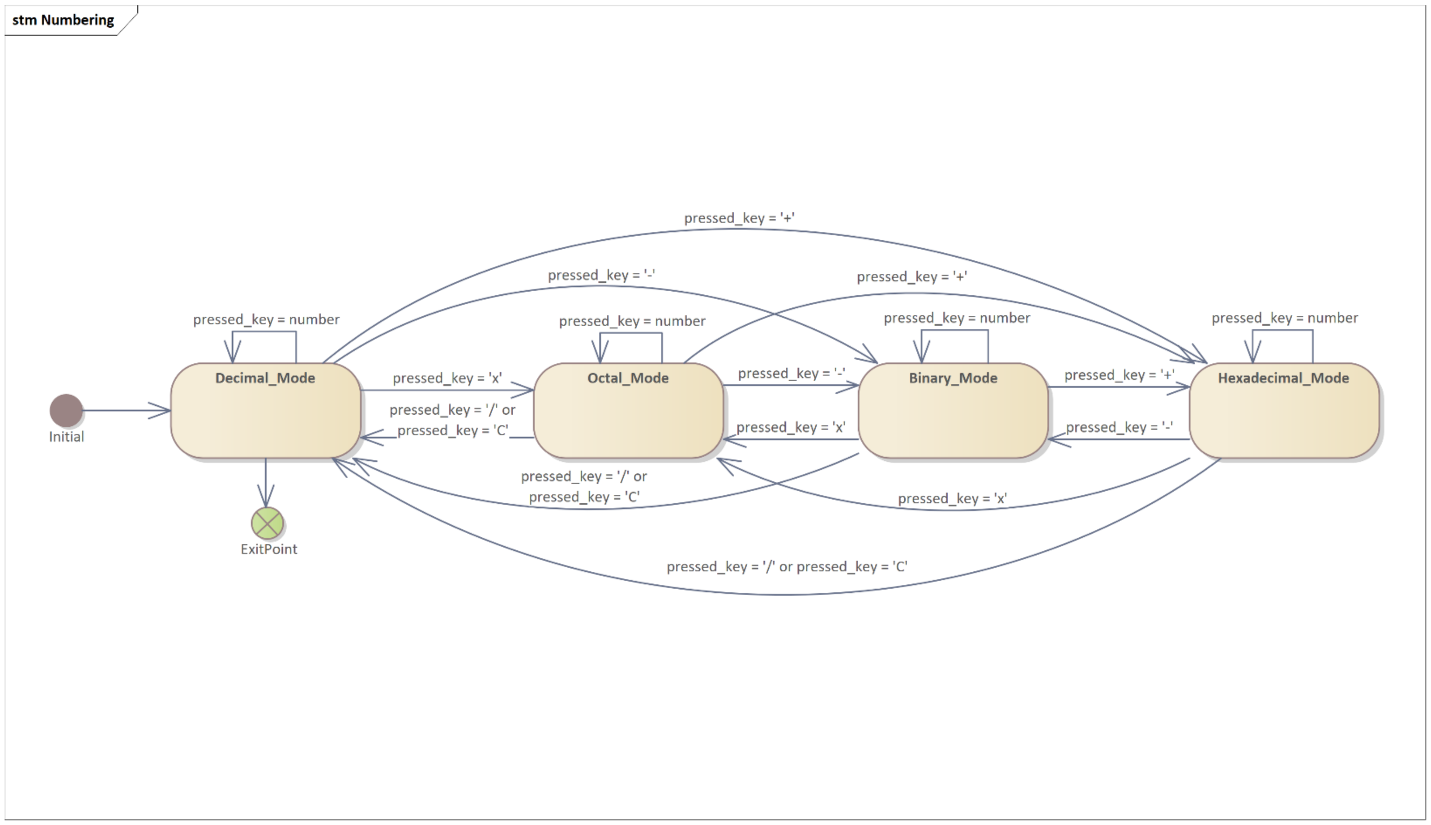


Fig (11) Numbering_Mode State Machine

System Coding

Application Layer

Main module

Flow Chart:

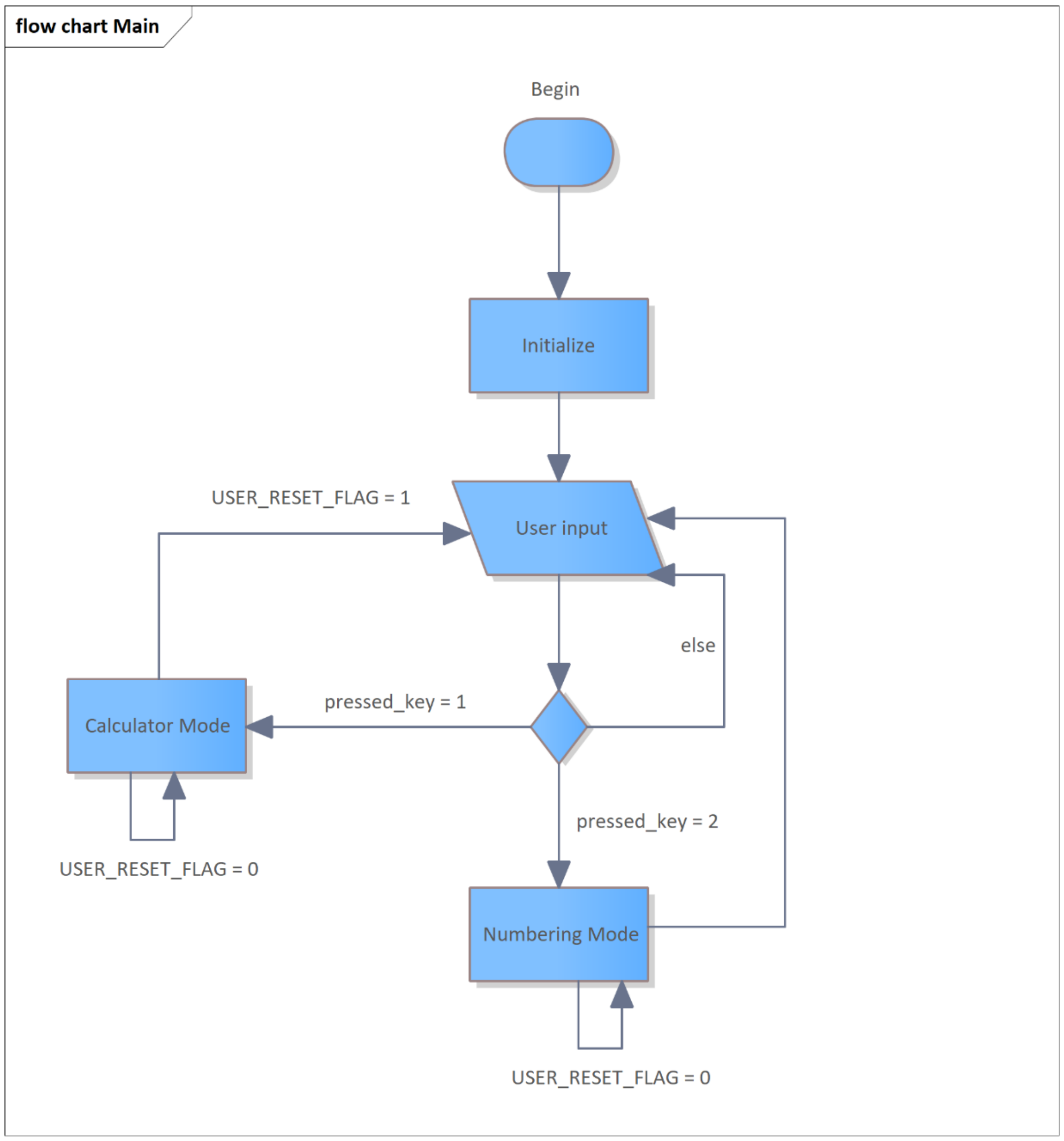


Fig (12) Main Module Flow Chart

APIs:

```
/**============================================
  * @Fn              - clock_init
  * @brief           - Initializes system clock
  * @param [in]      - None
  * @param [out]     - None
  * @retval          - None
  * Note             - Initializing GPIOA and GPIOB for LCD and Keypad
  */
void clock_init()
```

```
/**============================================
  * @Fn              - my_delay
  * @brief           - This function will make a delay without using a timer
  * @param [in]      - None
  * @param [out]     - None
  * @retval          - None
  * Note             - None
  */
void my_delay(int x)
```

```
/**============================================
  * @Fn              - MAIN_INIT
  * @brief           - This function initializes clock, peripherals, LCD, and keypad
  * @param [in]      - None
  * @param [out]     - None
  * @retval          - None
  * Note             - This function will be called in MAIN_INIT state
  */
STATE_DEF(MAIN_INIT)
```

```
/**============================================
  * @Fn              - MAIN_SELECTION
  * @brief           - This function asks the user to choose between calculator mode and
numbering systems mode
  * @param [in]      - None
  * @param [out]     - None
  * @retval          - None
  * Note             - This function will be called in MAIN_SELECTION state
  */
STATE_DEF(MAIN_SELECTION)
```

```
/**=============================================
  * @Fn              - MAIN_RUNNING
  * @brief           - This function will pass control to either calculator mode or numbering
system mode
  * @param [in]      - None
  * @param [out]     - None
  * @retval          - None
  * Note             - This function will be called in MAIN_RUNNING state
  */
STATE_DEF(MAIN_RUNNING)
```

Calculate_Mode Module

Flow Chart:

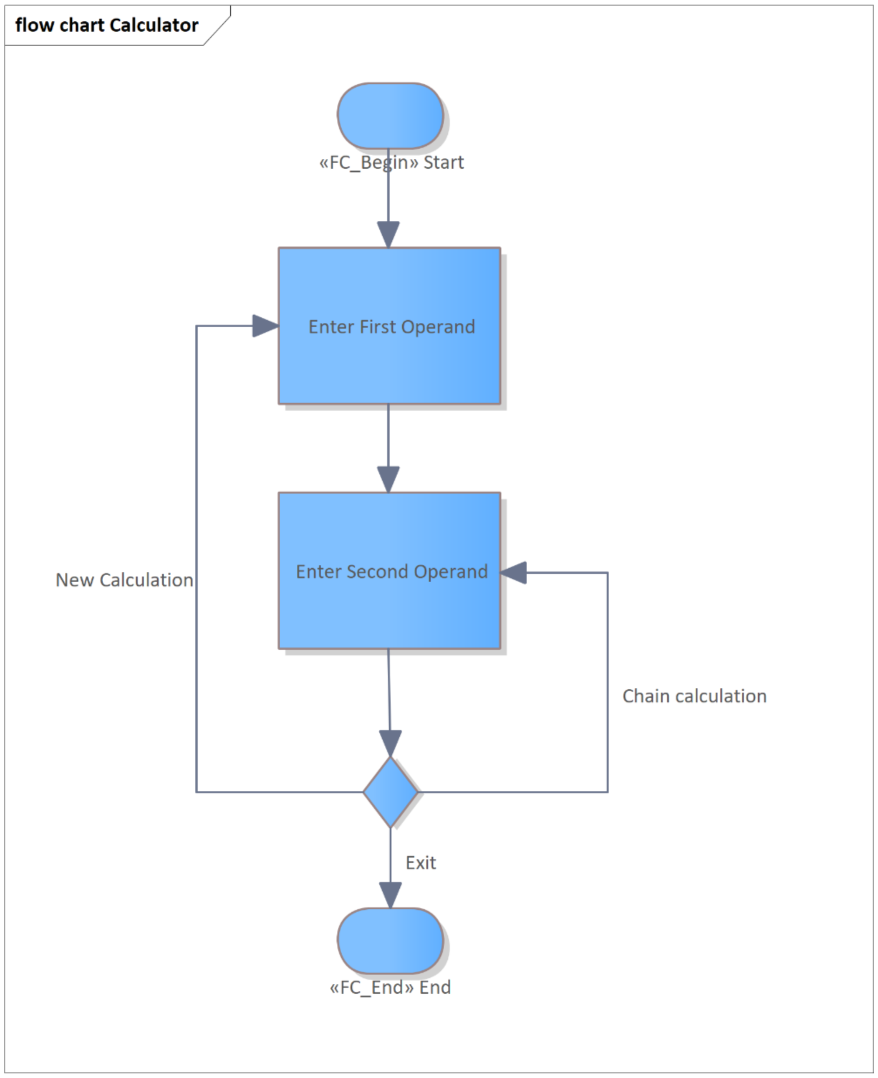


Fig (13) Calculate_Mode Flow Chart

APIs:

```
/**=============================================
  * @Fn              - Flush_Array
  * @brief           - This function copies elements of arr to destination
  * @param           - arr: Pointer to the source array
  * @param           - length: Length of data in the array
  * @param           - destination: Destination variable to save array elements into
  * @retval          - None
  * Note             - None
  */
static void Flush_Array(uint8 *arr, uint8 *length, uint32 *destination)
```

```
/**=============================================
  * @Fn              - Calculate_Result
  * @brief           - This function shall do the calculation and return the result
  * @param [in]      - op1: First operand
  * @param [in]      - op2: Second operand
  * @param [in]      - operator: Operation sign (+,-,x,/)
  * @param [out]     - None
  * @retval          - Result of the calculation
  * Note             - In minus operation, it will always return a positive integer which will
be the absolute difference
  *                  - If no operation is specified, it will return the first operand op1
  */
static uint32 Calculate_Result(uint32 op1, uint32 op2, uint8 operator)
```

```
/**=============================================
  * @Fn              - ST_First_Operand
  * @brief           - In this state, the system will store user entry in first operand array
  * @param [in]      - None
  * @param [out]     - None
  * @retval          - None
  * Note             - This function will be called in First_Operand state
  */
STATE_DEF(First_Operand)
```

```
/**============================================
  * @Fn               - ST_Second_Operand
  * @brief            - In this state, the system will store user entry in second operand array
  * @param [in]       - None
  * @param [out]      - None
  * @retval           - None
  * Note              - This function will be called in Second_Operand state
  */
STATE_DEF(Second_Operand)
```

```
/**============================================
  * @Fn               - ST_Result
  * @brief            - In this state, the system will calculate the result and show it on LCD
  * @param [in]       - None
  * @param [out]      - None
  * @retval           - None
  * Note              - This function will be called in Result state
  */
STATE_DEF(Result)
```

Numbering_Mode

Flow Chart:

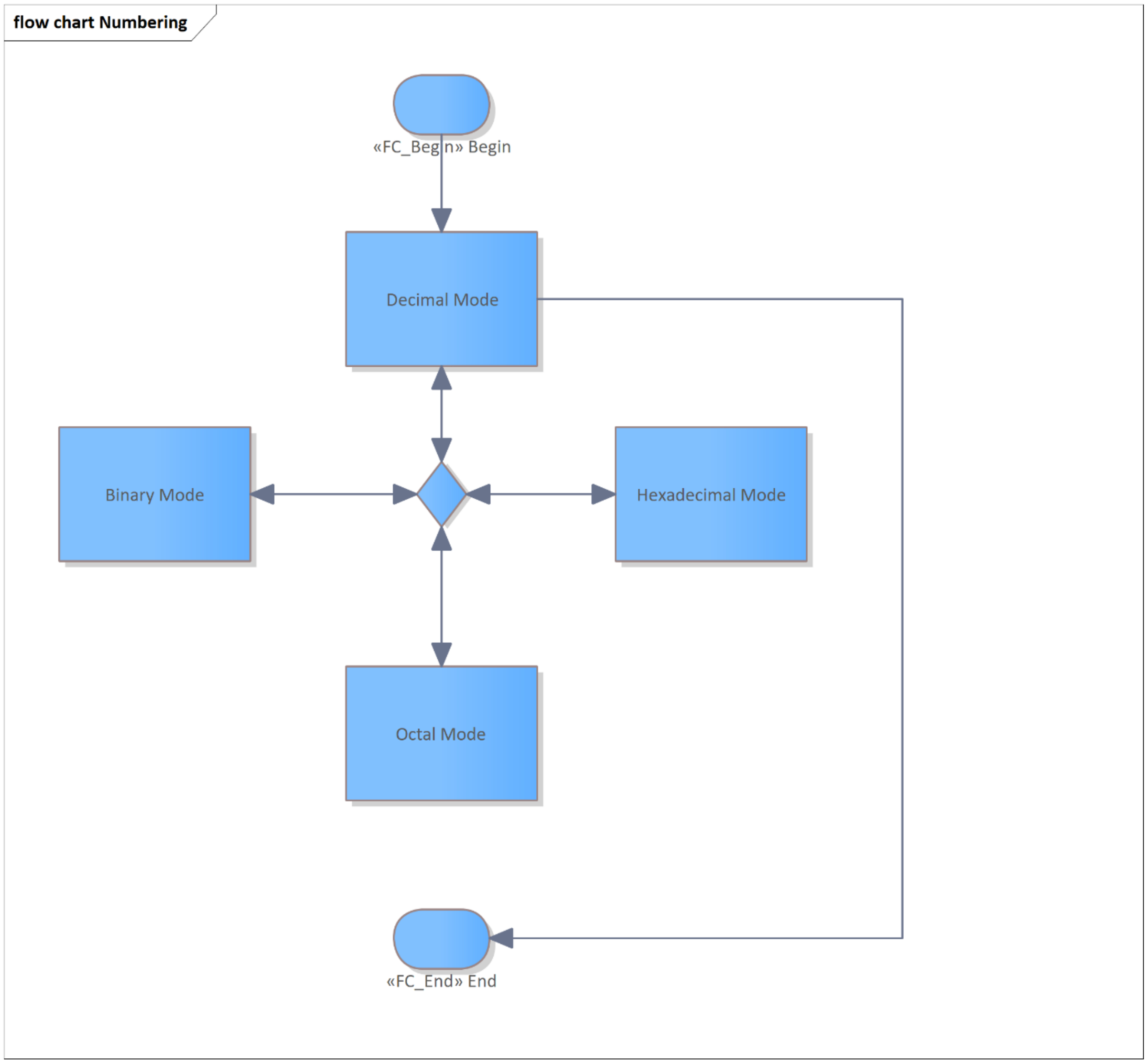


Fig (14) Numbering_Mode Flow Chart

APIs:

```
/**===========================================
  * @Fn                - PowerOf
  * @brief             - This function calculates the result of "base" power of "power"
  * @param [in]        - base: Base number
  * @param [in]        - power: Exponend number
  * @retval            - Result of "base" power of "power"
  * Note               - None
  */
static uint32 PowerOf(uint8 base, uint8 power)
```

```c
/**============================================
  * @Fn             - Print_Array_LCD
  * @brief          - This function will print an array on the LCD
  * @param [in]     - Array: Pointer to the array containing digits
  * @param [in]     - Length: Length of valid digits in the array
  * @retval         - None
  * Note            - Supports arrays containing hexadecimal digits (>=10)
  */
static void Print_Array_LCD(uint8 *Array, uint8 Length)
```

```c
/**============================================
  * @Fn             - Flush_Array
  * @brief          - This function will save array elements into a variable
  * @param [in]     - arr: Pointer to the array containing digits
  * @param [in]     - length: Length of valid digits in the array
  * @retval         - Extracted number out of the array
  * Note            - Can't be used if array elements can contain 2 digit number (>=10) EX:
Hexadecimal numbers
  */
static uint32 Flush_Array(uint8 *arr, uint8 length)
```

```c
/**============================================
  * @Fn             - Integer_To_Array_Reversed
  * @brief          - This function will save an array to an array in reversed order
  * @param [in]     - source_int: Integer variable that contains the decimal value
  * @param [out]    - Dest_Array: Pointer to the destination array to save digits into
  * @retval         - Length of valid elements in array
  * Note            - You must use function Reverse_Array after this if you want the values
to be normal not reversed
  */
static uint8 Integer_To_Array_Reversed(uint32 source_int, uint8 *Dest_Array)
```

```c
/**============================================
  * @Fn             - Reverse_Array
  * @brief          - This function will reverse elements in an array of given size
  * @param          - arr: Pointer to the array to be reversed
  * @param [in]     - length: Length of valid array elements
  * @param [out]    - None
  * @retval         - None
  * Note            - None
  */
static void Reverse_Array(uint8 *arr, uint8 length)
```

```
/**============================================
  * @Fn               - DecToBin
  * @brief            - This function shall convert decimal number from "Decimal_Number" array
to octal number and save it in "Octal_Number" array
  * @param [in]      - None
  * @param [out]     - None
  * @retval           - None
  * Note              - None
  */
static void DecToOct()
```

```
/**============================================
  * @Fn               - OctToDec
  * @brief            - This function shall convert octal number from "Octal_Number" array to
decimal number and save it in "Decimal_Number" array
  * @param [in]      - None
  * @param [out]     - None
  * @retval           - None
  * Note              - None
  */
static void OctToDec()
```

```
/**============================================
  * @Fn               - DecToBin
  * @brief            - This function shall convert decimal number from "Decimal_Number" array
to binary number and save it in "Binary_Number" array
  * @param [in]      - None
  * @param [out]     - None
  * @retval           - None
  * Note              - None
  */
static void DecToBin()
```

```
/**============================================
  * @Fn               - BinToDec
  * @brief            - This function shall convert binary number from "Binary_Number" array to
decimal number and save it in "Decimal_Number" array
  * @param [in]      - None
  * @param [out]     - None
  * @retval           - None
  * Note              - None
  */
static void BinToDec()
/**============================================
```

```c
  * @Fn              - DecToHex
  * @brief           - This function shall convert decimal number from "Decimal_Number" array
to hexadecimal number and save it in "Hexa_Number" array
  * @param [in]      - None
  * @param [out]     - None
  * @retval          - None
  * Note             - None
  */
static void DecToHex()
```

```c
/**=============================================
  * @Fn              - HexToDec
  * @brief           - This function shall convert hexadecimal number from "Hexa_Number" array
to decimal number and save it in "Decimal_Number" array
  * @param [in]      - None
  * @param [out]     - None
  * @retval          - None
  * Note             - None
  */
static void HexToDec()
```

```c
/**=============================================
  * @Fn              - ST_Decimal_Mode
  * @brief           - In this state, the number on the screen is displayed in decimal format
  * @param [in]      - None
  * @param [out]     - None
  * @retval          - None
  * Note             - Initial state
  */
STATE_DEF(Decimal_Mode)
```

```c
/**=============================================
  * @Fn              - ST_Octal_Mode
  * @brief           - In this state, the number on the screen is displayed in octal format
  * @param [in]      - None
  * @param [out]     - None
  * @retval          - None
  * Note             - None
  */
STATE_DEF(Octal_Mode)
```

```c
/**=============================================
  * @Fn               - ST_Binary_Mode
  * @brief            - In this state, the number on the screen is displayed in binary format
  * @param [in]       - None
  * @param [out]      - None
  * @retval           - None
  * Note              - None
  */
STATE_DEF(Binary_Mode)
```

```
/**=============================================
  * @Fn               - ST_Hexadecimal_Mode
  * @brief            - In this state, the number on the screen is displayed in hexadecimal
format
  * @param [in]       - None
  * @param [out]      - None
  * @retval           - None
  * Note              - None
  */
STATE_DEF(Hexadecimal_Mode)
```

HAL Layer

Keypad Module

APIs:

```
/**===========================================
  * @Fn              - keypad_init
  * @brief           - Initializes the keypad
  * @param [in]      - None
  * @param [out]     - None
  * @retval          - None
  * Note             - User must define GPIO pins for rows and columns in @ref
Keypad_PINS_define
  */
void keypad_init()
```

```
/**===========================================
  * @Fn              - keypad_Get_Pressed_Key
  * @brief           - Checks for any pressed key and returns the value of it
  * @param [in]      - None
  * @param [out]     - None
  * @retval          - Value of the pressed key, or F if no key is pressed
  * Note             - None
  */
uint8 keypad_Get_Pressed_Key()
```

LCD Module

APIs:

```
/**===========================================
  * @Fn              - LCD_Init
  * @brief           - Initialized LCD based on user defined configurations
  * @param [in]      - None
  * @param [out]     - None
  * @retval          - None
  * Note             - User must set configurations @ref LCD_CONFIG_define
  */
void LCD_Init()
```

```c
/**================================================
  * @Fn              - LCD_Send_Command
  * @brief           - Sends a command to the LCD to be executed
  * @param [in]      - command: command to be executed @ref LCD_COMMANDS_define
  * @param [out]     - None
  * @retval          - None
  * Note             - None
  */
void LCD_Send_Command(uint8 command)
```

```c
/**================================================
  * @Fn              - LCD_Send_Char
  * @brief           - Sends a char to the LCD to be displayed
  * @param [in]      - Char: ASCII character to be displayed on screen
  * @param [out]     - None
  * @retval          - None
  * Note             - None
  */
void LCD_Send_Char(uint8 Char)
```

```c
/**================================================
  * @Fn              - LCD_Send_Char_Pos
  * @brief           - Sends a char to the LCD to be displayed at a specific location
  * @param [in]      - Char: ASCII character to be displayed on screen
  * @param [in]      - row: Selects the row number of the displayed character @ref
LCD_ROWS_POS_define
  * @param [in]      - column: Selects the column number of the displayed character (1...16)
  * @param [out]     - None
  * @retval          - None
  * Note             - None
  */
void LCD_Send_Char_Pos(uint8 Char, uint8 row, uint8 column)
```

```c
/**================================================
  * @Fn              - LCD_Send_String
  * @brief           - Sends a string to the LCD to be displayed
  * @param [in]      - string: pointer to a string of characters to be displayed on LCD
  * @param [out]     - None
  * @retval          - None
  * Note             - None
  */
void LCD_Send_String(uint8 *string)
```

```c
/**===============================================
  * @Fn              - LCD_Send_string_Pos
  * @brief           - Sends a string to the LCD to be displayed at a specific location
  * @param [in]      - string: pointer to a string of characters to be displayed on LCD
  * @param [in]      - row: Selects the row number of the displayed character @ref
LCD_ROWS_POS_define
  * @param [in]      - column: Selects the column number of the displayed character (1...16)
  * @param [out]     - None
  * @retval          - None
  * Note             - None
  */
void LCD_Send_string_Pos(uint8 *string, uint8 row, uint8 column)
```

```c
/**===============================================
  * @Fn              - LCD_Send_Enable_Signal
  * @brief           - Sends enable signal to the LCD
  * @param [in]      - None
  * @param [out]     - None
  * @retval          - None
  * Note             - None
  */
void LCD_Send_Enable_Signal()
```

```c
/**===============================================
  * @Fn              - LCD_Set_Cursor
  * @brief           - Sets the location of the cursor
  * @param [in]      - row: Selects the row number of the displayed character @ref
LCD_ROWS_POS_define
  * @param [in]      - column: Selects the column number of the displayed character (1...16)
  * @param [out]     - None
  * @retval          - None
  * Note             - None
  */
void LCD_Set_Cursor(uint8 row, uint8 column)
```