GEBZE TEKNİK ÜNİVERSİTESİ MÜHENDİSLİK FAKÜLTESİ ELEKTRONİK MÜHENDİSLİĞİ BÖLÜMÜ

ELEC 334

MICROPROCESSORS

Project 2 – Scientific Calculator

Date: 23.12.2020

Name – Surname	Mahmut Safa BULAT
Number	141024051

Objective

Main objective of this project/midterm is to create a fully operational scientific calculator in C. This calculator will have a keypad connected to enter the numbers and execute basic scientific and trigonometric functions. A 4-digit seven segment display (SSD) should be used to display these numbers.

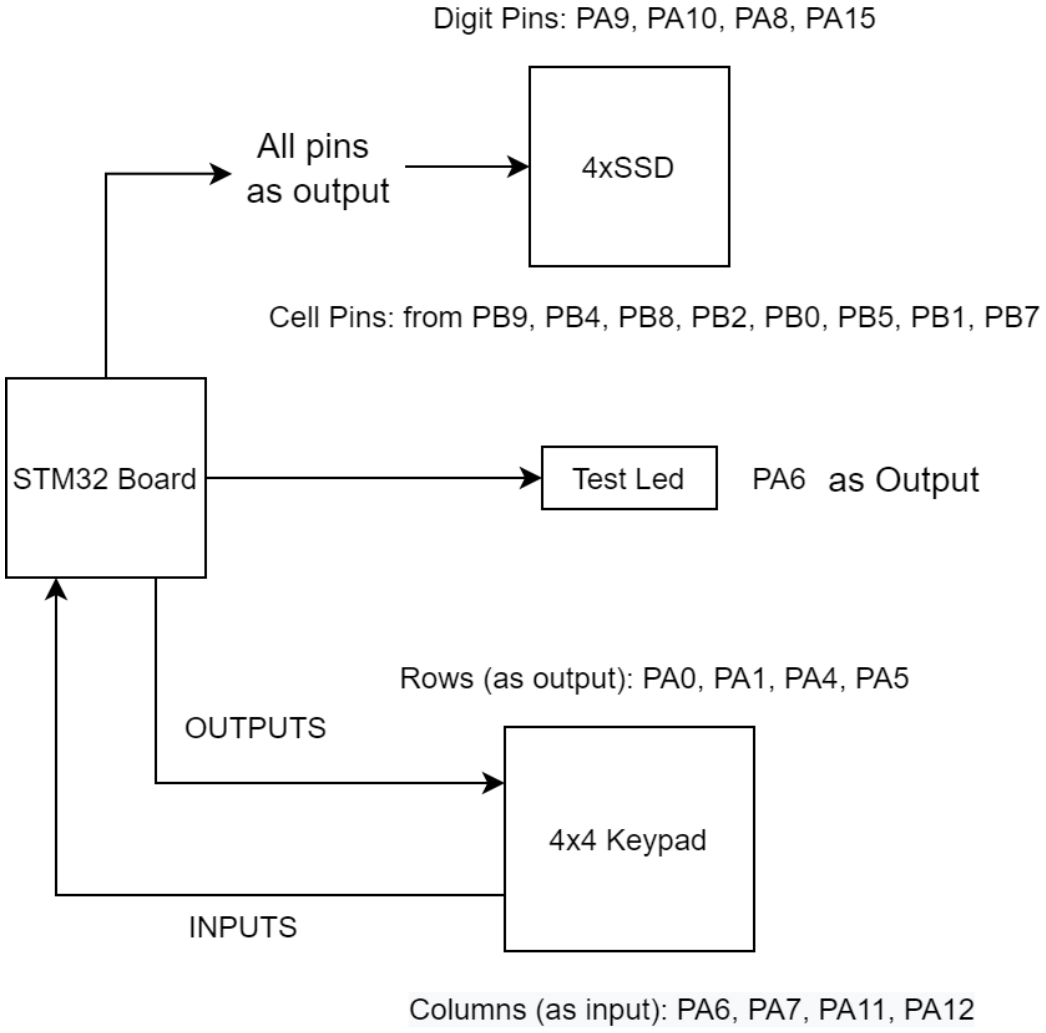
Assembling the Circuit

Before we move on to the actual project and coding, we must collect our (desired) modules and make a **part list**, then modelling our circuit with **block diagram** then finally build the circuit itself.

1. **Part List:**
   - STM32G031K8 Nucleo Board 145,44 TL
   - 4x4 Push Button Keypad 9,09 TL
   - 4xSSD 6,82 TL
   - Wiring cables 5,18 TL
   - Resistors 0,2 TL
   - Test LEDs (recommended but not necessary) 0,5 TL

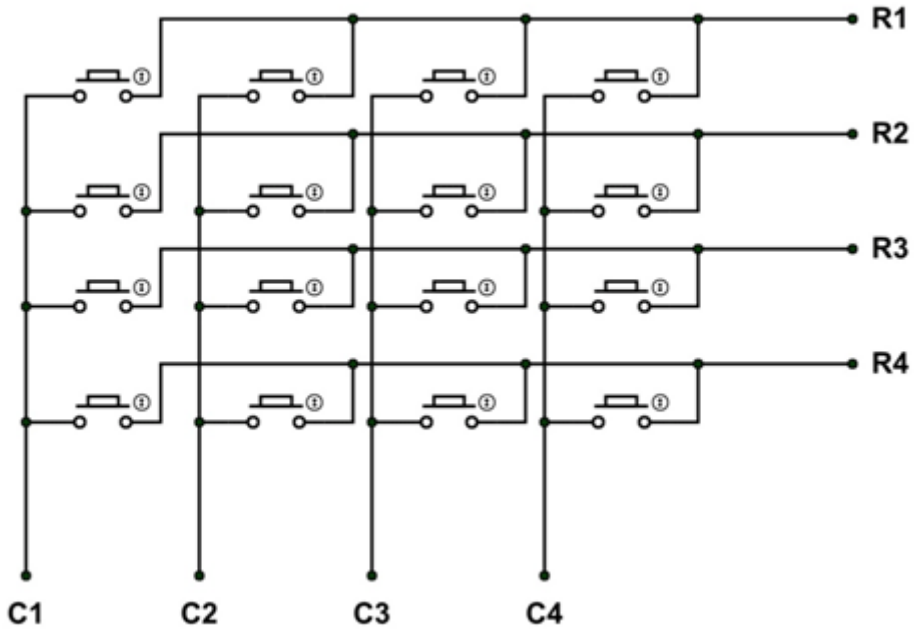
After listing and collecting the parts, we could model our circuitry.

2. **Block Diagram**



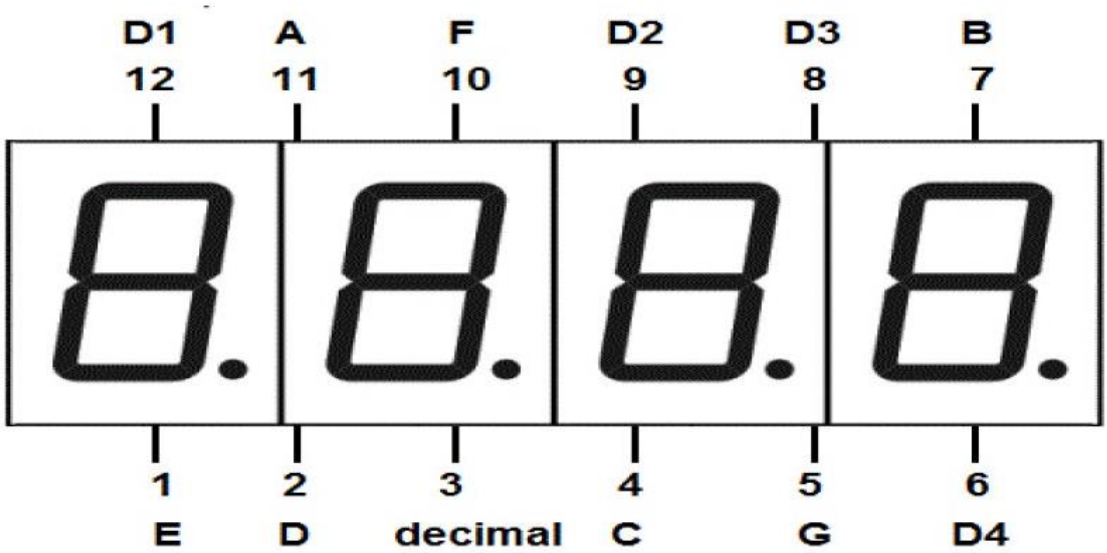
After modelling our circuit with block diagram, we need connection (wiring) diagram as well;

Keypad:



R1	R2	R3	R4	C1	C2	C3	C4
PA5	PA4	PA1	PA0	PA7	PA6	PA11	PA12

4xSSD:



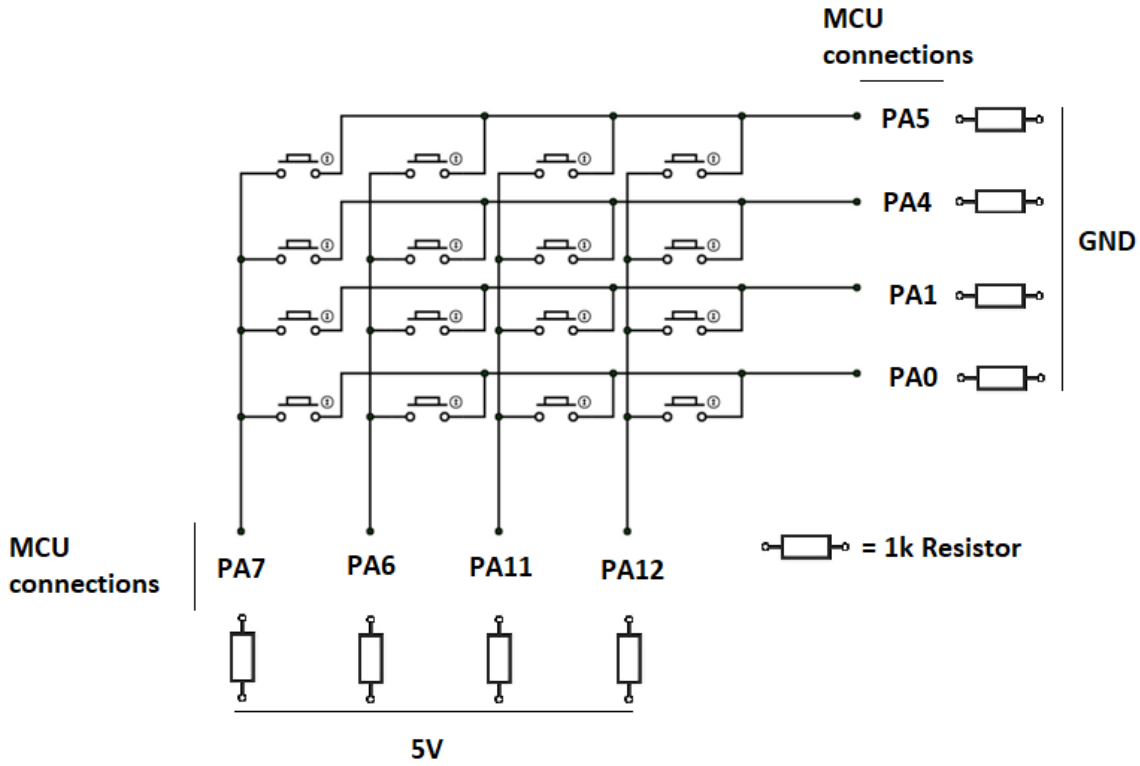
D1	D2	D3	D4	A	B	C	D	E	F	G	DOT
PA15	PA8	PA10	PA9	PB9	PB4	PB8	PB2	PB0	PB5	PB1	PB7

Not: Instead of ‘decimal’, I use ‘dot’ for the dot cell of SSD.

Test LED: Green Test LED is connected to PB6.

Before we move on to the actual circuit, we must examine our modules especially the keypad; because we don’t want to deal with bounce issues, and we must know that it is working with whether GND’ or Vcc’ configuration.

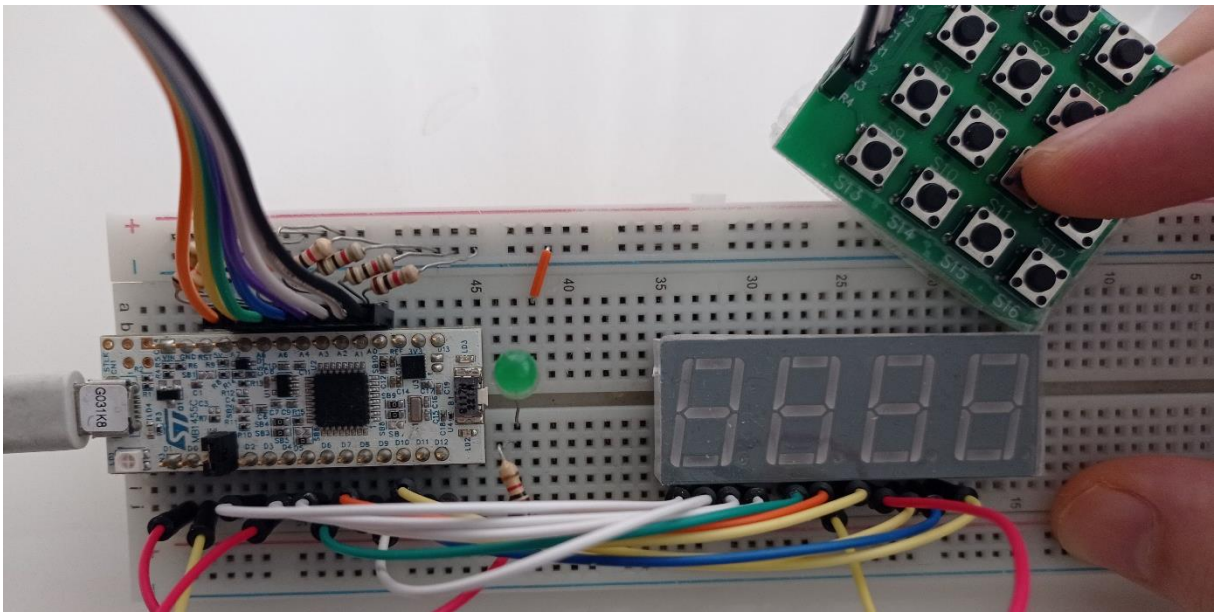
After I examine my module’s datasheets, and do some tests with logic analyzer and multimeter it is matrix structure (see figure at wiring diagram) , we could use 4xSSD as it is but we have to add some resistors prevent bounces when we using keypad. And since keypad works with ‘LOW Active’ configuration; My de-bounce circuit is;

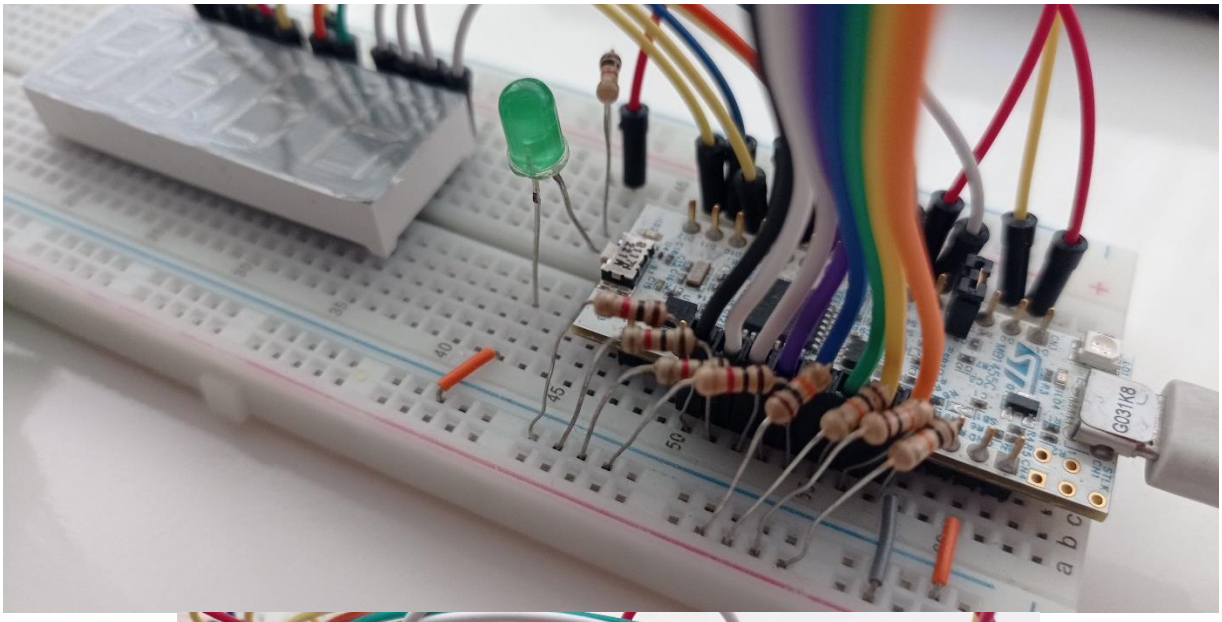


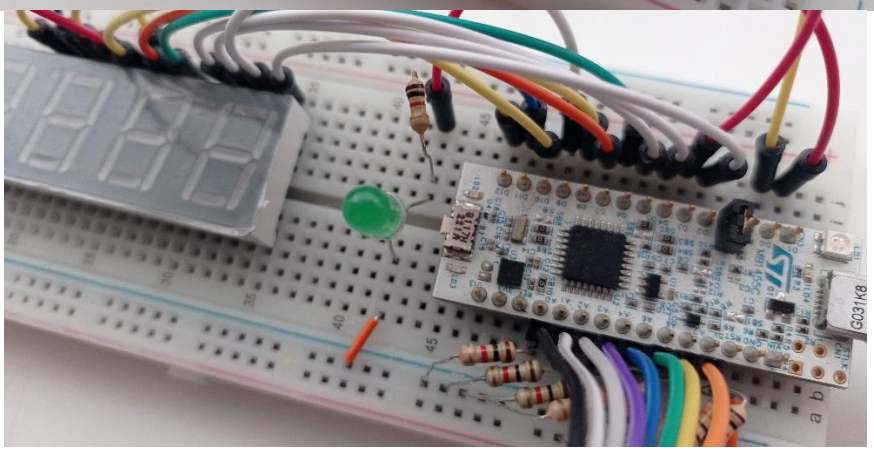
As ‘LOW Active’ circuit, when I pressed the button it sends GND to the MCU, except that they are connected the 5V to prevent bounces.

3. The Circuit

Now we can build our circuit as;



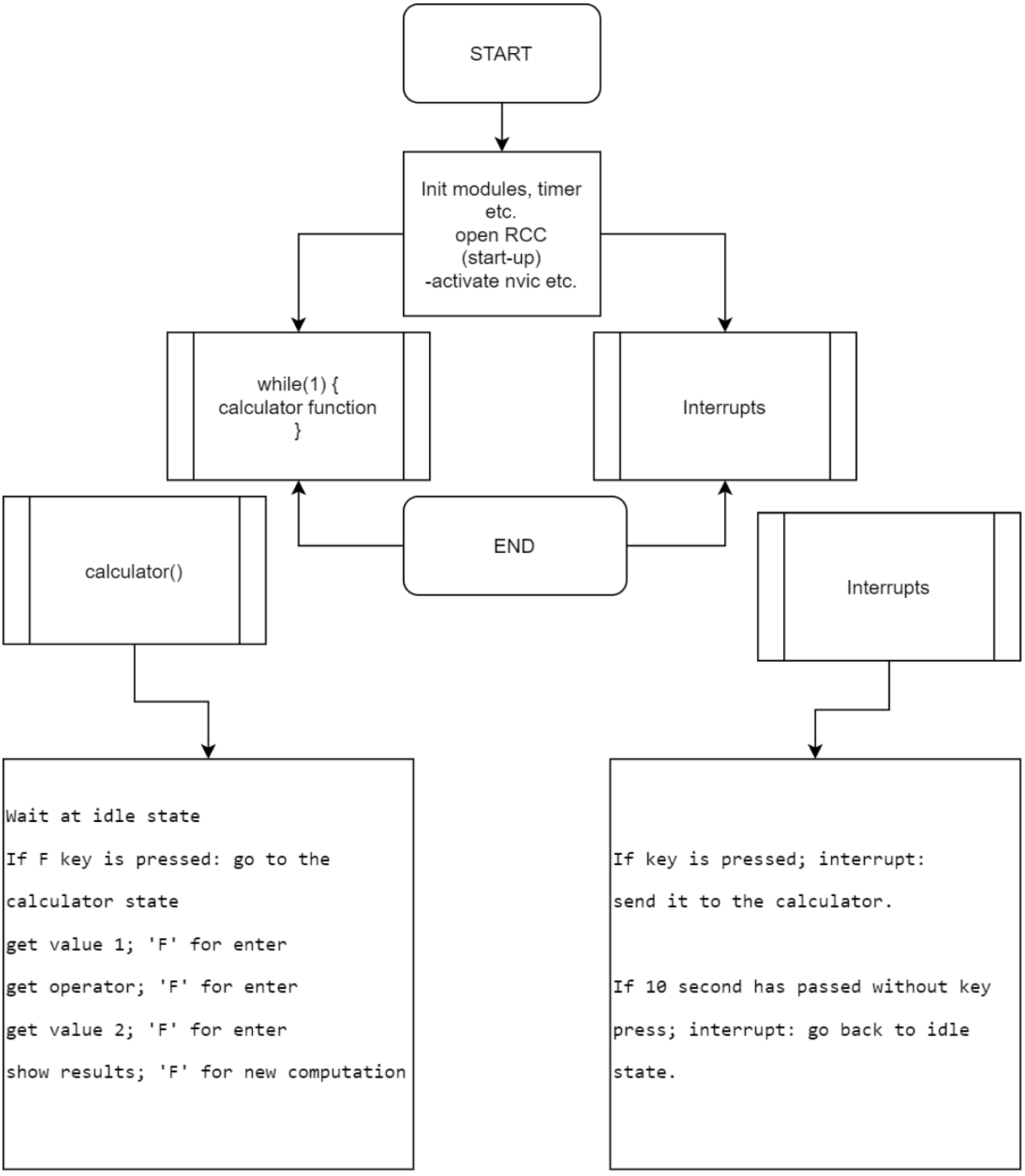




Coding

After our circuit is ready we can start coding in C. But before that we need to plan our methodology (when we coding) with **flowchart**.

- **Flowchart**



Our code draft is ready. We can move on the coding stage where the actual magic happens.

- **Code – Functions**

Note: Code will be attached in code folder.

```
/*
 * name: main.c
 *
 * author: Mahmut Safa BULAT
 *
 * description: This is a project2 main file for ELEC334 project2;
 * "A Fully Operational Scientific Calculator"
 * Every section is properly explained with comments.
 * You can find pin connections at the end of the code (down below the main)
 */

//**********************************Includes*****************************//
#include "stm32g0xx.h"
#include "math.h"
//**************************Includes * END*****************************//
//*********************************Definitions*****************************//
#define clear   (0x3B7) //Clear the SSD

#define seg1    (1<<9)  //Segment 1 definition
#define seg2    (1<<10) //Segment 2 definition
#define seg3    (1<<8)  //Segment 3 definition
#define seg4    (1<<15) //Segment 4 definition
//*********************************Definitions * END*****************************//

//*********************************Global Variables*****************************//
uint32_t digits[4]  ={0x0,0x0,0x0,0x0};
uint32_t digits2[4] ={0x0,0x0,0x0,0x0};
```

```
//*********************************Functions*****************************//
void init_timer1(){                 //initialization of the TIM1 ···
void delay(volatile uint32_t s) {   //delay function ···
void init_keypad(){                 //initialization of the Keypad ···
void init_4SSD(){                   //initialization of the 4xSSD ···
uint8_t keypad(){                   //keypad button configurations ···
void switcher(int entersth){        //switcher; char(int+48) to HEX ···
void capture_key(int keycapture){   //capture (switched) key ···
void segment1(uint32_t aa){         //drive : segment 1 ···
void segment2(uint32_t bb){         //drive : segment 2 ···
void segment3(uint32_t cc){         //drive : segment 3 ···
void segment4(uint32_t dd){         //drive : segment 4 ···
void printer(uint32_t segm1, uint32_t segm2, uint32_t segm3, uint32_t segm4){   //4xSSD driver function ···
void idle_state(){                  //idle state for calculator (stand by) ···
_Bool isInteger(double val){        //Little operation for detecting float numbers. ···
void calculator_functions(){        //(some) calculator functions ···
void calculator(){                  //the Calculator ···
//*********************************Functions * END*****************************//

//*************************Interrupts and main*************************//
void TIM1_BRK_UP_TRG_COM_IRQHandler(void){  //timer handler ···
int main(void) {                            //main ···
//*************************Interrupts and main * END*************************//
```

Task List, Completition status

Keypad Works properly?	✔
SSD Works properly?	✔
No bouncing on the buttons?	✔
No considerable delay with button presses?	✔
No flickering on the displays?	✔
Is code properly commented?	✔
On power-up idle state?	✔
If no button is pressed for 10 seconds go back to the idle state?	✔
Shift numbers to the left?	✔
If digits are full new number ignored (changed on purpose) ;; **If digits are full, new number resets** (like TV remote system) see **note[1]**	✔
Basic mode calculations? (+, -, *, /)	✔
Scientific mode calculations? (log, ln, sqrt, pow) (x^2 is changed on purpose) ;; **You can compute x^y with power instead of just x^2**	✔
Trigonometric mode?	✔
Replace with pi?	✔
F key is for enter/equal?	✔
Floating number display?	✔
Overflow check?	✔
Invalid check?	✔
Negative number display?	✔
Use previous result? (changed on purpose) ;; **If you press 'A' instead entering number, it replaces previous result with that number.** (like scientific calculators) see **note[2]**	✔

Note[1]: I changed this requirementation because; If I type wrong number or change my mind, I would be able to change the number right away. For example; I write 1234 instead 6453, now screen full with number 1234, if I press 6 it will reset the number and change like; _ _ _ 6 , and I can enter the rest like 6453. If I can do that, of course I can do 'if full, ignore next' …

Note[2]: Again I changed this requirementation because; In scientific calculators you can use previous result (with ANS key) everywhere. No let's say previous value is 3, I can press A (and enter) for value 1 and for example C (and enter) for multiplication and press A again (and enter) for replace previous answer with value 2. The result of this will be; A x A ; 3 x 3 = 9. I think this method is more useful than Invoking the last if nothing is entered.. Again, I can do that, of course I can do 'if no val1, replace with previous ans' …

Numerical Methodology

In this section we will see how to deal with numerical issues and parsing problems (from my perspective).

For most of the computation I used “math.” Library. And for others I just use the operators like = , + , - etc. But the tricky part is creating number from some hex codes (because I send digits to SSD with HEX) and parsing that number into hex files again. It needs some demonstration;
For example let’s examine some number like 1234, 247, -586 …

1234:

For 1234 I respectively send 0x110,0x217,0x316,0x132 to the SSD and hold them in some array like a=[0x110,0x217,0x316,0x132] via some functions. But I need actual number (values) for to compute or operate some operation. Then I switch back hex values to digits and hold them in an another array like b=[1,2,3,4] via some another function.

Now I need to represent these numbers as 4 digit (complete) number. I achieve such operation as following;
Number= (digit4 * 1000) + (digit3 * 100) + (digit2 * 10) + (digit1 * 1);
So digits become; 1*1000+2*100+3*10*4 = **1234** just like I wanted.

234: a=[0,0x217,0x132,0x310] → b=[0,2,4,7] number= (0 * 1000) + (2 * 100) + (4 * 10) + (7 * 1);

-568: a=[0x2,0x326,0x337,0x327] → b=[-,5,8,6] number= -((5 * 100) + (8 * 10) + (6 * 1));

And couple more operations similar that I described in video (you can find the link in video folder).

References

- “math.h” library (mat library)
- https://stackoverflow.com/ (for getting some help during coding)
- www.datasheetarchive.com (for datasheets)

Some Challenges that I faced..

Hardest part of this project was showing the floating numbers for me. Since I am holding values in some array digit to digit, it was hard to detect where is the decimal pointer (dot) or is number float or integer? How can I detect them and resolve that problem once and for all?

My solution was comparing number and numbers integer part with bool function;

```
_Bool isInteger(double val){        //Little operation for detecting float num
bers.
    int truncated = (int)val;
    return (val == truncated);
}
```

If the return is 0, number is float. And then I parse the number to its integer part like if the number is like 1,234 or like 12,34 … And that solved all of my problems.

Conclusion

In this project I've learned how I can use keypad, interrupt and timer applications configured with NVIC. It was educational and fun project to make. I think using 4xSSD for calculator is not a best option but it worked well. I would use 16*2 LCD display for that project; because I would see both numbers, operator and result in the same frame. And my number range would be wider…

But it was a good project I believe that I've achieved all the task and more… My coding could be more efficient maybe but it works flawless.