GEBZE TECHNICAL UNİVERSİTY

ELECTRONİC ENGINEERING

ELEC334

MICROPROCESSOR

PROJECT 01

PREPARED BY
ÖMER CEBECİ 171024007

1.INTRODUCTION:

In this Project, The program is written that when the users turn on board (STM32-g031k8) all digits zero along one second and then '4007'.When user push the button ,seven segment display count from random number to zero. If all digits are zero, external LED turns on. When users pushs button again at counting time,counting pause.If user push button again counting resume.

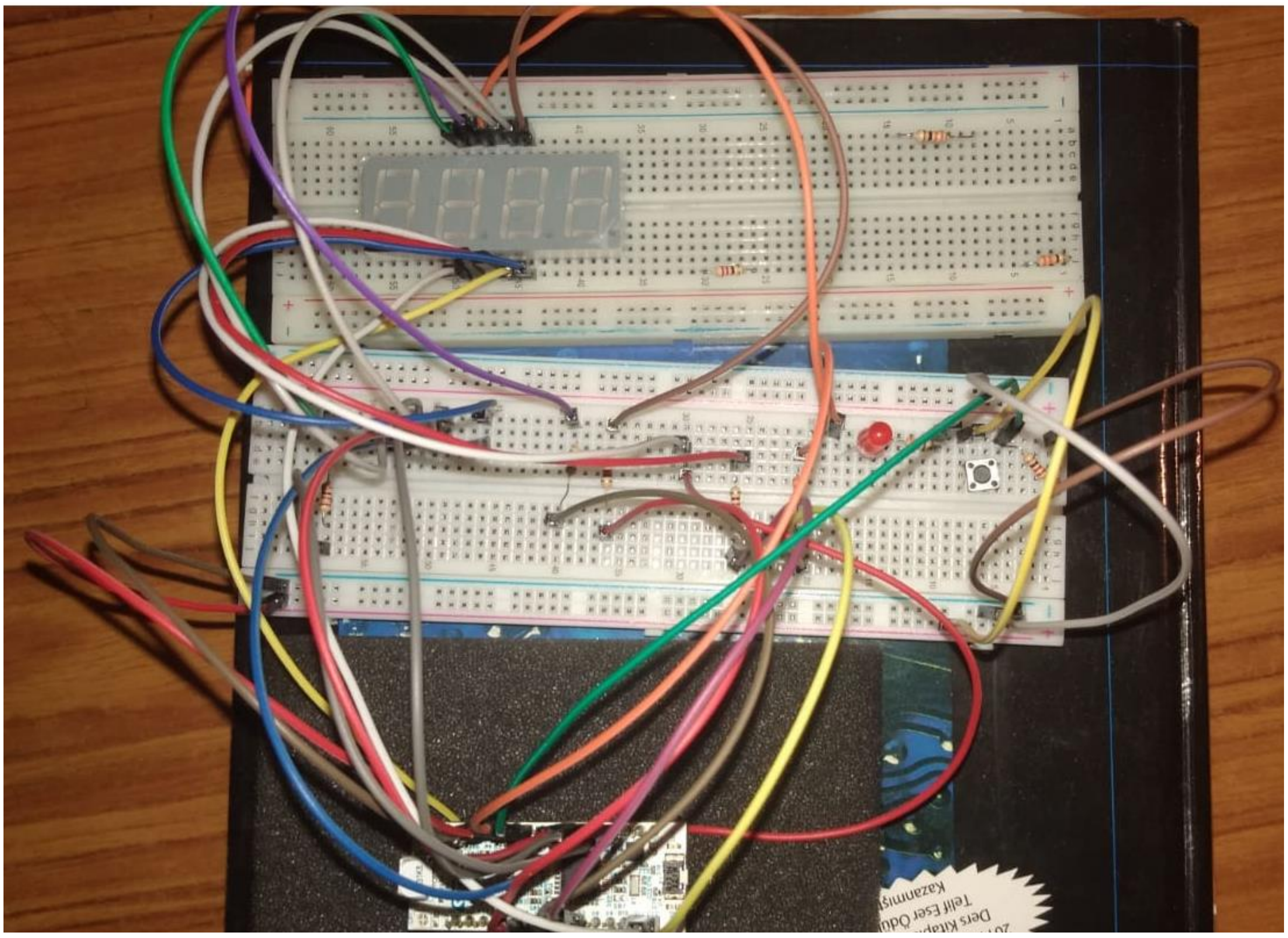
2.PROCESS OF CODE'S GENERATE:

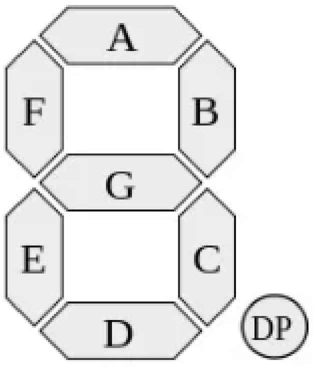
Firstly, how four digit seven segment Works was learned. And hardware was established.

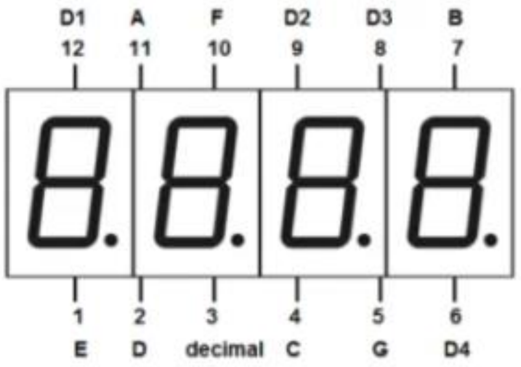
Secondly, four digits was turned on , and codes was written for numbers.There are special functions for every numbers.Then, only one digit was counted from nine to zero.

Thirdly, four digits were turned on as synchronous. Then countdown process was done.

Lastly, button and external LED were integrated to the code. There are pictures at below about this parts.


Picture of Hardware





Board and Seven segment's connections were done according to the these pictures

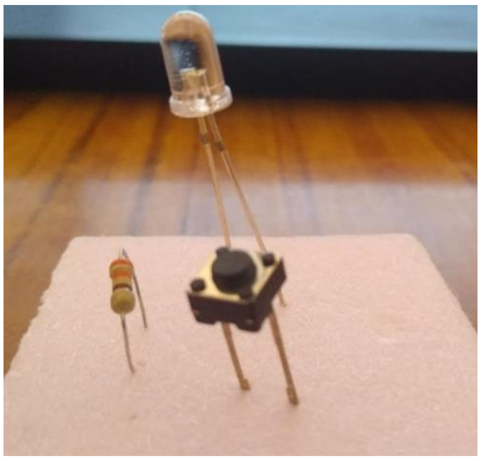
PA0->D1
PA1->D2
PA4->D3
PA5->D4
PA7->LED
PA6->BUTTON

PB0->A
PB1->B
PB2->C
PB3->D
PB4->E
PB5->F
PB6->G

Project materials are like that:

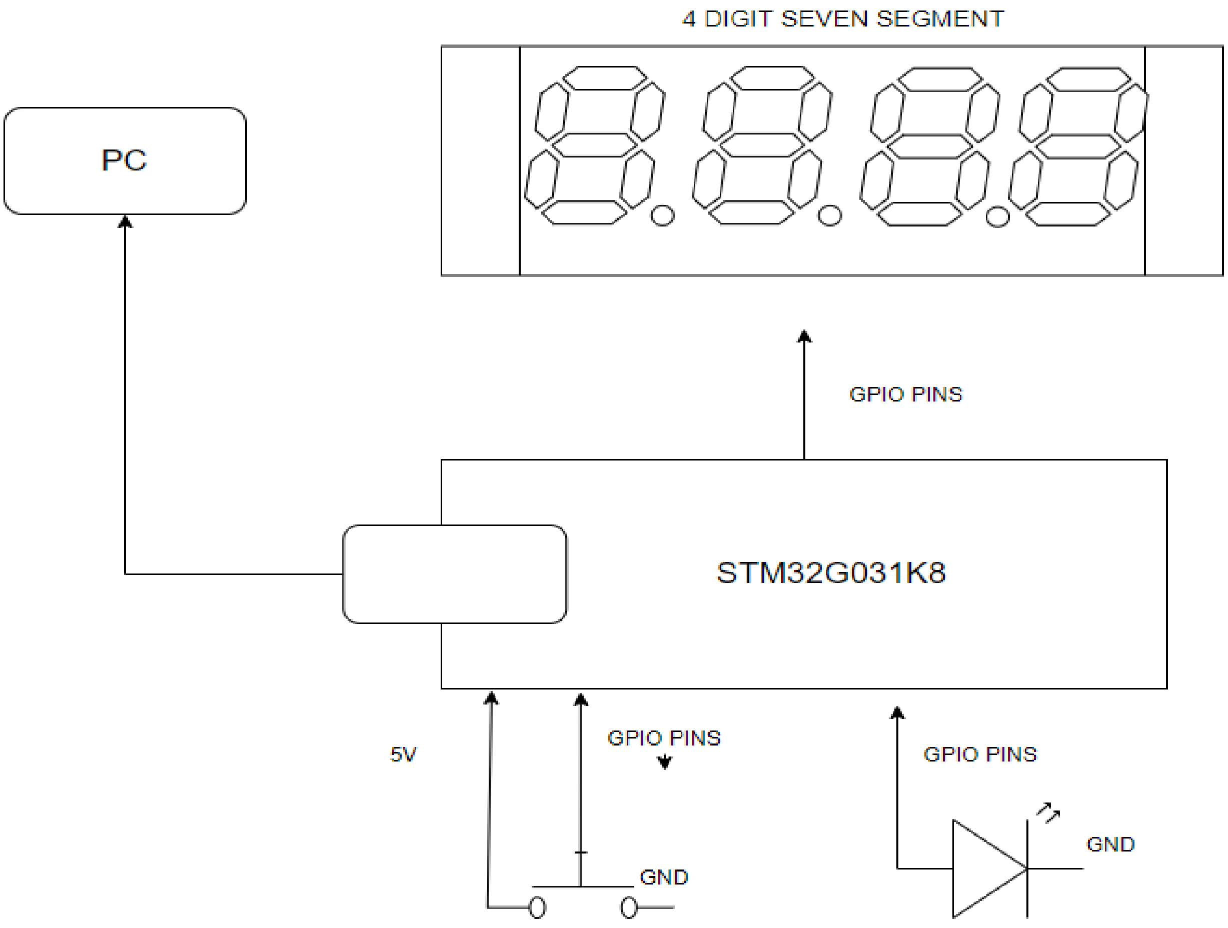
Man to Man Jumper/Man to Woman Jumper

4 Digit Seven Segment Display


LED,Resistors and Push-Button

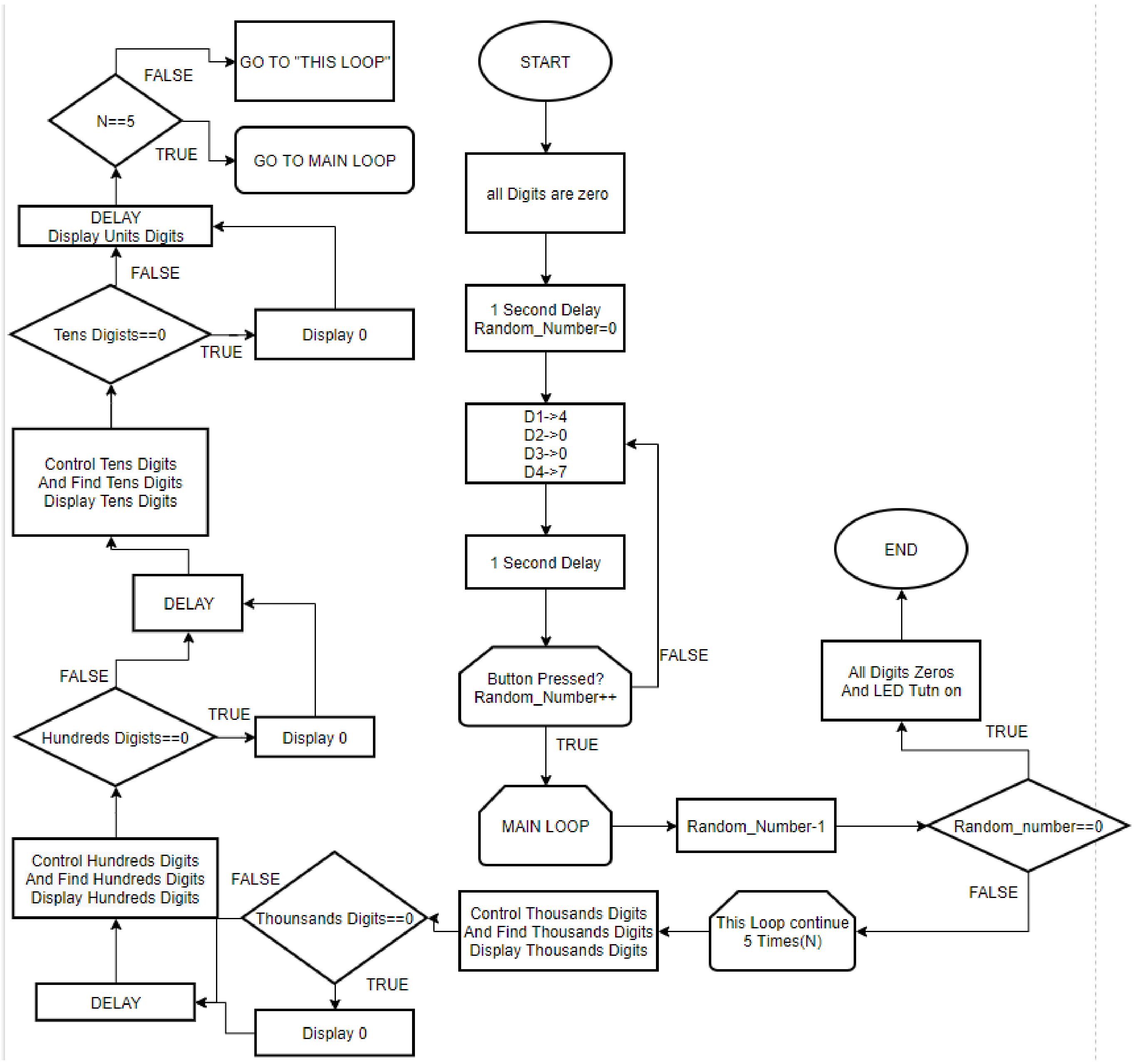
3.BLOCKDIAGRAM:

Blockdiagram is for this project as follows:



Blockdiagram

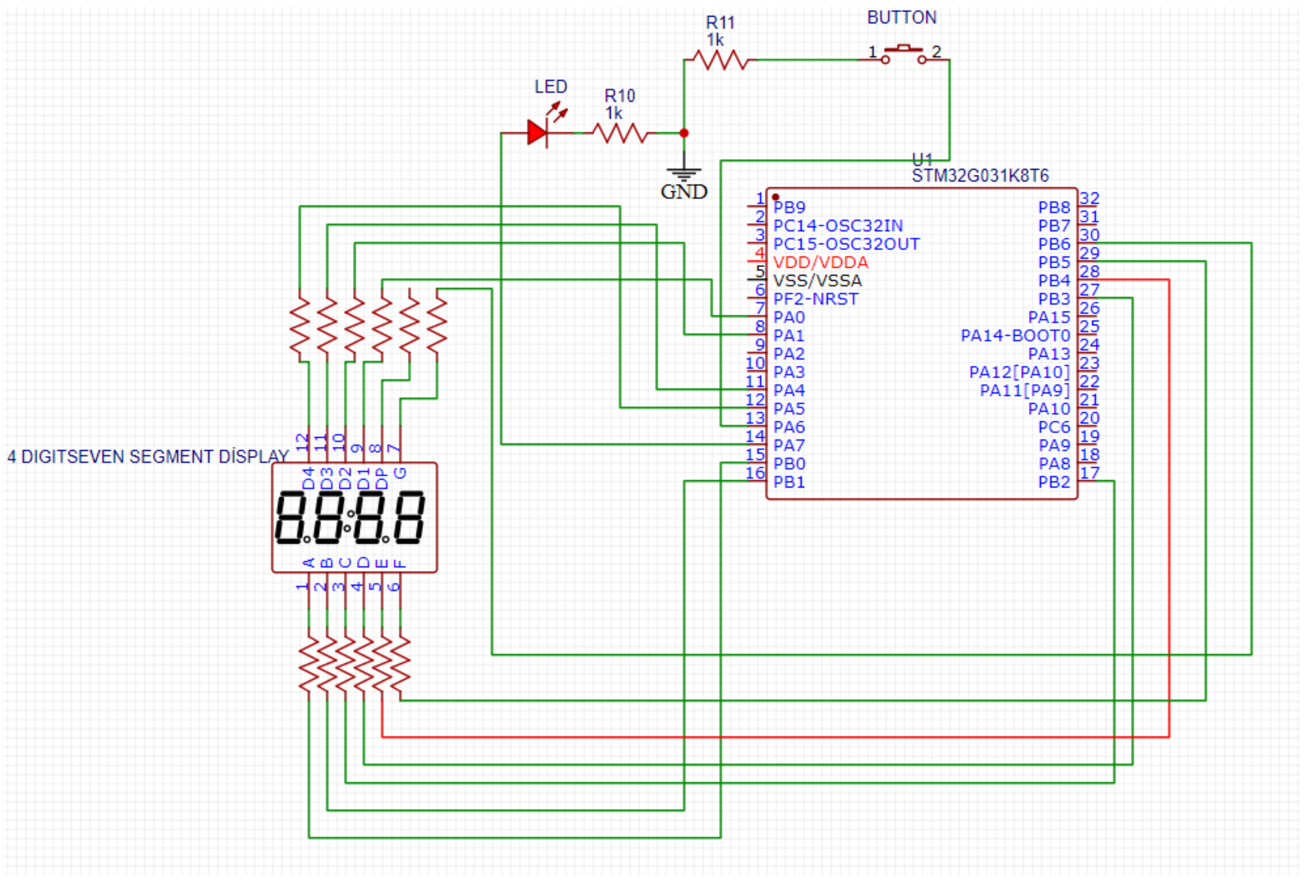
4.FLOWCHART:



Flowchart

5.SCHEMATIC:

Schematic is for this project as follows:



Schematic

6.CODE ALGORITHM:

First step in the code**,** when user power up the board, all digits have zero during one second. Then D1 have ‘4’ D2 have ‘0’ D3 have ‘0’ D4 have ‘7’ until user push the button. If user dont push the button ,this loop always continue and that time every loop ,code count one by one because of random number.This counting produce our random number.For example user push the button,when tenth loop,our random number is ten.

Second step in the code ,random number’s digits were found and these digits were displayed with very little delay so user saw all digits light synchronous. While the code finds the digits of the random number, it also checks if these numbers are zero.If these numbers are zero,pc go to the zero2 function to display zero.Code also check the button,When user push button again ,Counting stop and pc go to the nop function.User push button again counting resume.

Finally, when the all digits zero, counting finish and external led light up.

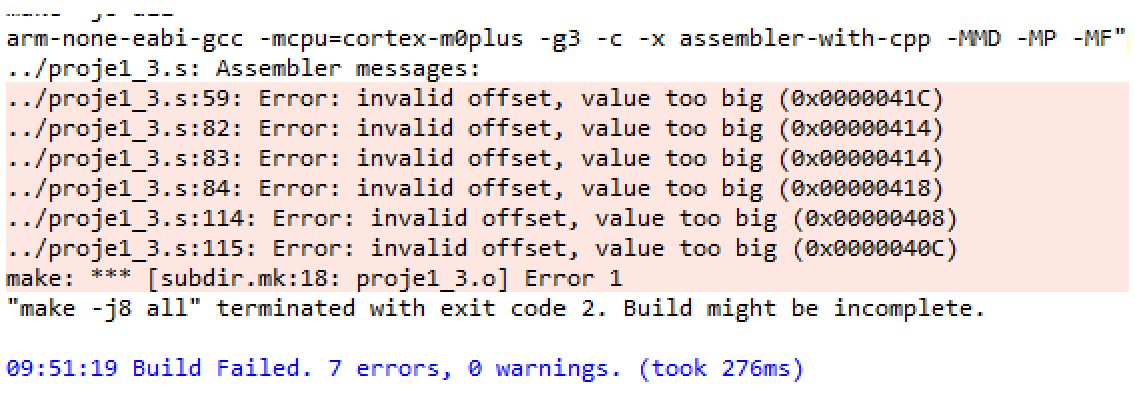
Here;There are some difficulties;

When this code was written ,understanding light synchronous was difficult.The other difficult thing was countdown.

When this code was written ,other homeworks and lab's informations were used.For example substract operation or produce random number.

Another problem in this Project, after 750 lines at the stm32CubeIde,compliler dont allow any new code. So LED cant light up because of this reason.

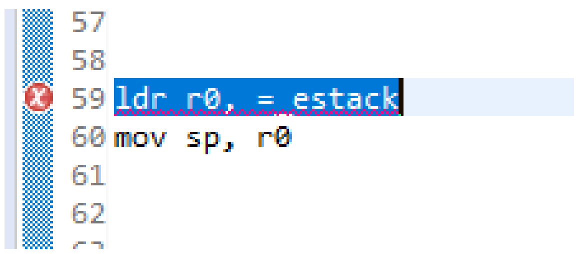
When user pushes the button while counting operation,code use the stack memory ,random number's digits at the stack memmory and in nop function code pop the these values and display at the seven segment but compiler dont allow for new codes.The error message like that:



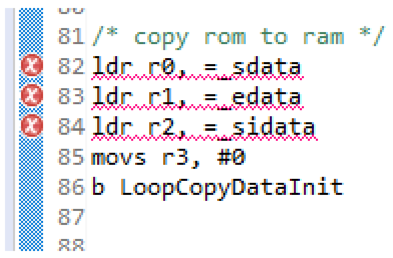
```
arm-none-eabi-gcc -mcpu=cortex-m0plus -g3 -c -x assembler-with-cpp -MMD -MP -MF"
../proje1_3.s: Assembler messages:
../proje1_3.s:59: Error: invalid offset, value too big (0x0000041C)
../proje1_3.s:82: Error: invalid offset, value too big (0x00000414)
../proje1_3.s:83: Error: invalid offset, value too big (0x00000414)
../proje1_3.s:84: Error: invalid offset, value too big (0x00000418)
../proje1_3.s:114: Error: invalid offset, value too big (0x00000408)
../proje1_3.s:115: Error: invalid offset, value too big (0x0000040C)
make: *** [subdir.mk:18: proje1_3.o] Error 1
"make -j8 all" terminated with exit code 2. Build might be incomplete.

09:51:19 Build Failed. 7 errors, 0 warnings. (took 276ms)
```

Error Message



```
57
58
59 ldr r0, = estack
60 mov sp, r0
61
62
```



```
81 /* copy rom to ram */
82 ldr r0, =_sdata
83 ldr r1, =_edata
84 ldr r2, =_sidata
85 movs r3, #0
86 b LoopCopyDataInit
87
```

Error is in these lines but this problem wasn't solve.

7.CODES:

The Project codes as follows

```
.syntax unified
.cpu cortex-m0plus
.fpu softvfp
.thumb



 /* make linker see this */

.global Reset_Handler

/* get these from linker script */
.word _sdata
.word _edata
.word _sbss
.word _ebss

.equ RCC_BASE, (0x40021000) // RCC base address
.equ RCC_IOPENR, (RCC_BASE + (0x34)) // RCC IOPENR address offset
.equ GPIOA_BASE, (0x50000000) // GPIOA base address
.equ GPIOA_MODER, (GPIOA_BASE + (0x00)) // GPIOA MODER address offset
.equ GPIOA_ODR, (GPIOA_BASE + (0x14)) // GPIOA ODR address offset
.equ GPIOA_IDR, (GPIOA_BASE+(0x10))
.equ GPIOB_BASE, (0x50000400)
.equ GPIOB_MODER, (GPIOB_BASE + (0x00))
.equ GPIOB_IDR, (GPIOB_BASE+(0x10))
.equ GPIOB_ODR, (GPIOB_BASE + (0x14))
 /* vector table, +1 thumb mode */

.section .vectors
vector_table:
.word _estack /* Stack pointer */
.word Reset_Handler +1 /* Reset handler */
.word Default_Handler +1 /* NMI handler */
.word Default_Handler +1 /* HardFault handler */
 /* add rest of them here if needed */


/* reset handler */
.section .text
Reset_Handler:
 /* set stack pointer */

ldr r0, =_estack
mov sp, r0

bl init_data

 /* call main */
bl main
 /* trap if returned */
b .

 /* initialize data and bss sections */
.section .text
init_data:

/* copy rom to ram */
ldr r0, =_sdata
ldr r1, =_edata
ldr r2, =_sidata
movs r3, #0
b LoopCopyDataInit
```

```
CopyDataInit:


 ldr r4, [r2, r3]
 str r4, [r0, r3]
 adds r3, r3, #4



LoopCopyDataInit:
 adds r4, r0, r3
 cmp r4, r1
 bcc CopyDataInit



/* zero bss */


 ldr r2, =_sbss
 ldr r4, =_ebss
 movs r3, #0
 b LoopFillZerobss


FillZerobss:


 str r3, [r2]
 adds r2, r2, #4



LoopFillZerobss:


 cmp r2, r4
 bcc FillZerobss



bx lr
/* default handler */


 .section .text
 Default_Handler:
 b Default_Handler


 /* main function */
.section .text
```

```
write_d1: //A1 HIGH gor writing first digit(D4)
        ldr r6, =GPIOA_ODR
        ldr r5, [r6]
        ldr r4, =0x7F
        mvns r4, r4
        ands r5, r5, r4
        ldr r4,=0x1
        orrs r5,r5,r4
        str r5 ,[r6]
        bx lr

write_d2:  // A2 HIGH for writing second digit (D2)
        ldr r6, =GPIOA_ODR
        ldr r5, [r6]
        ldr r4, =0x7F
        mvns r4, r4
        ands r5, r5, r4
        ldr r4,=0x2
        orrs r5,r5,r4
        str r5 ,[r6]
        bx lr
write_d3:  // A3 HIGH for writing third digit (D3)
        ldr r6, =GPIOA_ODR
        ldr r5, [r6]
        ldr r4, =0x7F
        mvns r4, r4
        ands r5, r5, r4
        ldr r4,=0x10
        orrs r5,r5,r4
        str r5 ,[r6]
        bx lr

write_d4:  // A4 HIGH for Writing fourth digit (D4)
        ldr r6, =GPIOA_ODR
        ldr r5, [r6]
        ldr r4, =0x7F
        mvns r4, r4
        ands r5, r5, r4
        ldr r4,=0x20
        orrs r5,r5,r4
        str r5 ,[r6]
        bx lr
number_1:
        ldr r6, =GPIOB_ODR      // arrange A B C D E F G
        ldr r5, [r6]       // for writing nummber 1
        ldr r4,=0x7F
        mvns r4,r4
        ands r5,r5,r4
        ldr r4,=0x79
        orrs r5,r5,r4
        str r5,[r6]
        pop {r6}
        bx r6
number_2:
        ldr r6, =GPIOB_ODR // arrange A B C D E F G
        ldr r5, [r6]       // for writing nummber 2
        ldr r4,=0xFF
        mvns r4,r4
        ands r5,r5,r4
        ldr r4,=0xA4
        orrs r5,r5,r4
        str r5,[r6]
        pop {r6}
        bx r6
```

```
number_3:
        ldr r6, =GPIOB_ODR  // arrange A B C D E F G
        ldr r5, [r6]             // for writing nummber 3
        ldr r4,=0xFF
        mvns r4,r4
        ands r5,r5,r4
        ldr r4,=0xB0
        orrs r5,r5,r4
        str r5,[r6]
        pop {r6}
        bx r6
number_4:
        ldr r6, =GPIOB_ODR // arrange A B C D E F G
        ldr r5, [r6]       // for writing nummber 4
        ldr r4,=0xFF
        mvns r4,r4
        ands r5,r5,r4
        ldr r4,=0x99
        orrs r5,r5,r4
        str r5,[r6]
        pop {r6}
        bx r6
number_5:
        ldr r6, =GPIOB_ODR // arrange A B C D E F G
        ldr r5, [r6]       // for writing nummber 5
        ldr r4,=0xFF
        mvns r4,r4
        ands r5,r5,r4
        ldr r4,=0x92
        orrs r5,r5,r4
        str r5,[r6]
        pop {r6}
        bx r6
number_6:
        ldr r6, =GPIOB_ODR // arrange A B C D E F G
        ldr r5, [r6]       // for writing nummber 6
        ldr r4,=0xFF
        mvns r4,r4
        ands r5,r5,r4
        ldr r4,=0x2
        orrs r5,r5,r4
        str r5,[r6]
        pop {r6}
        bx r6
number_7:
        ldr r6, =GPIOB_ODR // arrange A B C D E F G
        ldr r5, [r6]       // for writing nummber 7
        ldr r4,=0xFF
        mvns r4,r4
        ands r5,r5,r4
        ldr r4,=0xF8
        orrs r5,r5,r4
        str r5,[r6]
        pop {r6}
        bx r6
number_8:
        ldr r6, =GPIOB_ODR // arrange A B C D E F G
        ldr r5, [r6]      // for writing nummber 8
        ldr r4,=0xFF
        mvns r4,r4
        ands r5,r5,r4
        ldr r4,=0x0
        orrs r5,r5,r4
        str r5,[r6]
        pop {r6}
        bx r6
number_9:
        ldr r6, =GPIOB_ODR // arrange A B C D E F G
        ldr r5, [r6]       // for writing nummber 9
        ldr r4,=0xFF
        mvns r4,r4
        ands r5,r5,r4
        ldr r4,=0x90
        orrs r5,r5,r4
        str r5,[r6]
        pop {r6}
        bx r6
```

```
number_0:
        ldr r6, =GPIOB_ODR   // arrange A B C D E F G
        ldr r5, [r6]            // for writing nummber 0
        ldr r4,=0xFF
        mvns r4,r4
        ands r5,r5,r4
        ldr r4,=0x40
        orrs r5,r5,r4
        str r5,[r6]
        pop {r6}
        bx r6
```

```
control:
        push {lr}// we use stack memory because ,we branch control function from main but also  we branch
another function from control
                         // and we dont push lr right now,we will lose return address(main) and we cant come
back to the main
                         // we will pop lr after the display our number.

        ldr r4,=#0       // is number equal to 0?
        cmp r4,r6        // control the number and branch for display this number
        beq number_0

        ldr r4,=#1       // is number equal to 1?
        cmp r4,r6
        beq number_1     // control the number and branch for display this number

        ldr r4,=#9       // is number equal to 9?
        cmp r4,r6
        beq number_9

        ldr r4,=#8       // is number equal to 8?
        cmp r4,r6
        beq number_8     // control the number and branch for display this number

        ldr r4,=#7       // is number equal to 7?
        cmp r4,r6
        beq number_7     // control the number and branch for display this number

        ldr r4,=#6      // is number equal to 6?
        cmp r4,r6
        beq number_6    // control the number and branch for display this number

        ldr r4,=#5      // is number equal to 5?
        cmp r4,r6
        beq number_5     // control the number and branch for display this number


        ldr r4,=#4      // is number equal to 4?
        cmp r4,r6
        beq number_4    // control the number and branch for display this number

        ldr r4,=#3      // is number equal to 3?
        cmp r4,r6
        beq number_3     // control the number and branch for display this number

        ldr r4,=#2      // is number equal to 2?
        cmp r4,r6
        beq number_2     // control the number and branch for display this number
```

```
delay:

        adds r5,r5,#0x1  // codes delay function
        cmp r5,r7    // for every bl delay command pc register come here
        blt delay  //and every loop add r5 register and control with r7
        bx lr      // if r5 is equal r7 delay function finish
```

```
loop_button:
                bl write_d1 // activate first digit (thounsads digits)
                adds r1,r1,0x1
                ldr r6, =GPIOB_ODR
                ldr r5, [r6]
                ldr r4,=0xFF
                mvns r4,r4
                ands r5,r5,r4
                ldr r4,=0x99
                orrs r5,r5,r4
                str r5,[r6] //Write number 4 manually, Idont use number_4 function because pc cannot come back
//here I dont push anything

                ldr r7,=#3194  // we determine delay duration
                movs r5,#0x0   // delay function
                ands r5,r5
                bl delay

                bl write_d2 // activate second digit (hundreds digits)

                ldr r6, =GPIOB_ODR
                ldr r5, [r6]
                ldr r4,=0xFF
                mvns r4,r4
                ands r5,r5,r4
                ldr r4,=0x40
                orrs r5,r5,r4
                str r5,[r6]//Write number 0 manually, Idont use number_4 function because pc cannot come back
//here I dont push anything


                movs r5,#0x0   // delay function
                ands r5,r5
                bl delay

                bl write_d3 // activate third digit (tens digit)

                ldr r6, =GPIOB_ODR
                ldr r5, [r6]
                ldr r4,=0xFF
                mvns r4,r4
                ands r5,r5,r4
                ldr r4,=0x40
                orrs r5,r5,r4
                str r5,[r6]//Write number 0 manually, Idont use number_4 function because pc cannot come back
//here I dont push anything

                movs r5,#0x0   // delay function
                ands r5,r5
                bl delay

                bl write_d4  // activate fourth digit (units digit)

                ldr r6, =GPIOB_ODR
                ldr r5, [r6]
                ldr r4,=0xFF
                mvns r4,r4
                ands r5,r5,r4
                ldr r4,=0xF8
                orrs r5,r5,r4
                str r5,[r6]//Write number 7 manually, Idont use number_4 function because pc cannot come back
here I dont push anything
```

```
                movs r5,#0x0   // delay function
                ands r5,r5
                bl delay


                ldr r6,=GPIOA_IDR  // controlled the button
                ldr r5, [r6]
                ldr r4,=0x40
                ands r5,r5,r4
                ldr r4,=0x40
                cmp r5,r4
                bne loop_button // if user dont push button this loop continue
                b loop // if user push the button pc go to the 'loop' function
```

```
main:

                  //#3194

        ldr r6, =RCC_IOPENR
        ldr r5, [r6]
        /*GPIOA AND GPIOB CLOCK CONFIGURATON */
        movs r4, 0x3
        orrs r5, r5, r4
        str r5, [r6]

        ldr r6, =GPIOB_MODER // A B C D E F G configuration for numbers
        ldr r5, [r6]
        ldr r4, =0x3FFF
        mvns r4, r4
        ands r5, r5, r4
        //ldr r4, =0x1555
        ldr r4,=0x41555
        orrs r5, r5, r4
        str r5, [r6]

        ldr r6, =GPIOA_MODER // GPIOA configuration for control digits
        ldr r5, [r6]   // D1 PA0 D2 PA1 D3 PA4 D4 PA5 D4
        ldr r4,=0x3FFF
        mvns r4,r4
        ands r5,r5,r4
        ldr r4,=0x4505
        orrs r5,r5,r4
        str r5,[r6]




        loop_zero:
                ldr r7,=#3194 // determine the delay duration
                ldr r0,=#300  // determine the count_zero loop's numbers (300 times count_zero loop return)
                count_zero:
                        bl write_d1     // activate first digit (thounsads digits)
                        bl zero2       // zero2 function configure the GPIOB_ODR register for number zero
                        movs r5,#0
                        ands r5,r2     // delay function
                        bl delay

                        bl write_d2   // activate second digit (hundreds digits)
                        bl zero2     // zero2 function configure the GPIOB_ODR register for number zero
                        movs r5,#0
                        ands r5,r2
                        bl delay     // delay function

                        bl write_d3 // activate third digit (tens digit)
                        bl zero2   // zero2 function configure the GPIOB_ODR register for number zero
                        movs r5,#0
                        ands r5,r2
                        bl delay   // delay function

                        bl write_d4 // activate fourth digit (units digit)
                bl write_d4 // activate fourth digit (units digit)
                bl zero2    // zero2 function configure the GPIOB_ODR register for number zero
                movs r5,#0
                ands r5,r2
                bl delay    // delay function

                subs r0,r0,0x1
                cmp r0,0x0 // if r0 is equal to the zero ,pc go to the loop_button
                bne count_zero // control how many times count_zero loop return
                ldr  r3,=#3  // if r0 is equal to the zero ,we adjust how many times we will see same number
        ldr r1,=0x0 // our random number // r1 increase one by one until user push the button
        b loop_button
```

```
        loop:
                movs r0,0x0 // counter for loop_tekrar
                subs r1,r1,0x1 // every time we do this operation because of countdown



                loop_tekrar:
                        ldr r6,=GPIOA_IDR
                        ldr r5, [r6]
                        ldr r4,=0x40   // button control
                        ands r5,r5,r4
                        ldr r4,=0x40
                        cmp r5,r4
                        beq nop_loop// if user push the button pc go to the nop_loop

                        cmp r1,0x0  // we control the main number
                        beq result  // if main number is zero,countdown operation finish

                        ldr r7,=#350// detertmine the delay duration

                        movs r5,#0
                                                   // delay function
                        ands r5,r2
                        bl delay



                        bl write_d1
                        movs r2,0x0  // this number is used to find the digits
                        movs r6,0x0  // the number that display the seven segment
                        orrs r2,r2,r1

                        ldr r4 ,= #1000
                        cmp r2,r4
                        blt binleri_atlama

                        substract_binler: // every time we subs from 1000 to find tousands digits
                                adds r6,r6,0x1 // this register our thousands digits
                                subs r2,r2,r4
                                cmp r2,r4  // if r2 is smaller than 1000(r4) ,that means substract operation
//finish and we find thousands digits
                                bgt substract_binler

                        bl control  // r3 de birler bas birler bas yazılacak
                        delay_1:
                                movs r5,#0x0   // delay function
                                ands r5,r5
                                bl delay


                        bl write_d2

                        movs r6,0x0 //  the number that display the seven segment
                        movs r4 ,#100
                        cmp r2,r4 // we control the number after substract,because maybe we dont have
houndrands digits?
                        blt yuzlere_atlama// if houndrands digist is zero pc go to the this function to display
zero
                        substract_yuzler: // 100 a bölerek yüzler bas bulundu
                                adds r6,r6,0x1// this register our hundrands digits
                                subs r2,r2,r4
                                cmp r2,r4  // // if r2 is smaller than 100(r4) ,that means substract operation
finish and we find hundrands digits
                                bgt substract_yuzler

                        bl control  // this function is used to display random number's digits



                        delay_2:
                                movs r5,#0x0   // delay function
                                ands r5,r5
                                bl delay

                        bl write_d3


                        movs r6,0x0
                        movs r4 ,#10
                        cmp r2,r4 //// we control the number after substract,because maybe we dont have tens
//digits?
                        blt onlari_atlama// if tens digist is zero pc go to the this function to display zero
                        substract_onlar:
                                adds r6,r6,0x1// this register our hundrands digits
                                subs r2,r2,r4
                                cmp r2,r4
                                bgt substract_onlar // // if r2 is smaller than 100(r4) ,that means substract
//operation finish and we find hundrands digits

                        bl control  //

                        ldr r7,=#550
                        delay_3:
                                movs r5,#0x0   // delay function
                                ands r5,r5
                                bl delay


                        bl write_d4

                        movs r6,0x0  // //  the number that display the seven segment
                        orrs r6,r6,r2

                        bl control



                        ldr r7,=#550
                        movs r5,#0x0   // delay function
                        ands r5,r5
                        bl delay


                        adds r0,r0,0x1//loop_tekrar function continue 2 times for every main number
                        cmp r0,r3    // if loop_tekrar function finish,pc return the loop function ad then we
//find new number
                        beq loop     // loop_tekrar continue again for our new number

                        b loop_tekrar
```

```
yuzlere_atlama: // if our random number'S thounsands digits is zero,we use this function for display zero
        ldr r6, =GPIOB_ODR // A B C D E F G
        ldr r5, [r6]
        ldr r4,=0xFF
        mvns r4,r4
        ands r5,r5,r4
        ldr r4,=0x40
        orrs r5,r5,r4
        str r5,[r6]
        b delay_2
onlari_atlama:// if our random number'S hundrands digits is zero,we use this function for display zero
        ldr r6, =GPIOB_ODR // A B C D E F G
        ldr r5, [r6]
        ldr r4,=0xFF
        mvns r4,r4
        ands r5,r5,r4
        ldr r4,=0x40
        orrs r5,r5,r4
        str r5,[r6]
        b delay_3
binleri_atlama:// if our random number'S tens digits is zero,we use this function for display zero
        ldr r6, =GPIOB_ODR // A B C D E F G
        ldr r5, [r6]
        ldr r4,=0xFF
        mvns r4,r4
        ands r5,r5,r4
        ldr r4,=0x40
        orrs r5,r5,r4
        str r5,[r6]
        b delay_1

nop_loop:
                ////////////////////
                // if user push the button,countdown stop and noop_loop work until user push button again
                ldr r6,=GPIOA_IDR  // controlled the button
                ldr r5, [r6]
                ldr r4,=0x40
                ands r5,r5,r4
                ldr r4,=0x40
                cmp r5,r4
                beq loop_tekrar


                b nop_loop





result:// if count_down finish (that's means subs r1,0x1 -> r1=0),all digits have zero number
                ldr r7,=#0x3944
                movs r5,#0x0   // delay function
                ands r5,r5
                bl delay


                bl write_d1


                bl zero3

                movs r5,#0x0   // delay function
                ands r5,r5
                bl delay

                bl write_d2


                bl zero3

                movs r5,#0x0   // delay function
                ands r5,r5
                bl delay

                bl write_d3


                bl zero3
                movs r5,#0x0   // delay function
                ands r5,r5
                bl delay

                bl write_d4

                bl zero3


                movs r5,#0x0   // delay function
                ands r5,r5
                bl delay



                b result
zero2:
        ldr r6, =GPIOB_ODR //zero display
        ldr r5, [r6]   // we have extra function to display zero because,we can't use number_0 function(we dont
do push-pop or we dont use stck memmory)
        ldr r4,=0xFF   //
        mvns r4,r4
        ands r5,r5,r4
        ldr r4,=0x40
        orrs r5,r5,r4
        str r5,[r6]

        bx lr




b .



/* this should never get executed */



nop
```

8.CONCLUSION:

*In this Project, how to 4 digit seven segment display is controlled was learned,at that time how to use stack memory was learned.

*How to erase chip was learned.

*How to push button is used was learned.

*Personally result in this Project; design the hardware is good enough ,writing code is easy. Because for example wires placement is not good, you cant see the which cable go to which place on the seven segment or which cable is wrong.

*There are some mathematical operations for Project. D1 digit changing should be same with real second changing. delays within the loop adjusted considering these situation .One delay function lead 311 ns and in loop_tekrar function have five delays an every delay lead 3149 ns (5*3149) and every loop_tekrar function lead three times (5*3149)*3.After this calculation DA digit changing is same with real second changing.

*There is a another problem,if D1 digit changing is same with real second changing,user cant see the changing at the D4 digit.

9. REFERENCES:

* https://theokelo.co.ke/how-to-get-your-hs420361k-32-4-digit-7-segment-display-working-with-an-arduino/

* ARMv6-M Architecture Reference Manual

* http://www.tofla.iconbar.com/tofla/arm/arm02/index.htm

Video Link

https://drive.google.com/drive/folders/1vOwecJJsDa_lmSAxnhSQ4kXt1uFOhNSc?usp=sharig

*There are two videos in drive.First video is code explanation and second is debug.

*Copy url and paste the Google