



ELEC 334 – MICROPROCESSORS

PROJECT 1 – RANDOMIZED COUNTER

AZİZ CAN AKKAYA 171024005

Contents

1.Introduction.....	2
1.1 Briefing.....	2
1.2 Objective.....	2
2.Technical Aspects.....	3
2.1 Materials.....	3
2.2 Method.....	3
2.1.1 Random Number Algorithm.....	4
2.3 Diagrams.....	5
2.3.1 Hardware Block Diagram.....	5
2.3.2 Software Flowchart.....	6
3.Output.....	7
4.Conclusion.....	7
5.Appendix.....	9
6.References.....	22

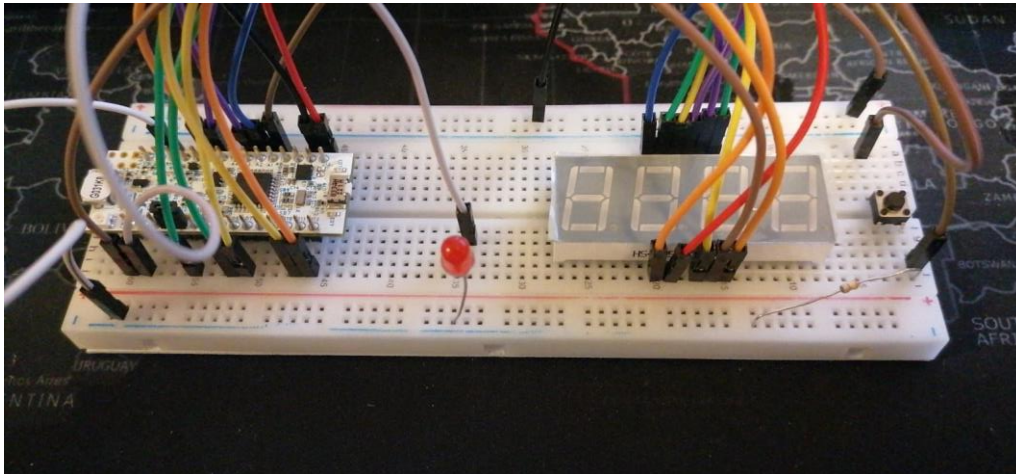
1 Introduction

1.1 Briefing

Document Version : 1.1

Document Published Date : 24/11/2020

Project Version : v4.7_D



1.2 Objective

This project is developed on a STM32G031K8T NUCLEO BOARD. The Arm Cortex-M0+ architecture gained functionality with “asm.s” file developed with assembly language. The aim of the Project is to understand and gain a brief knowledge about ARM Cortex-M0+, how the pins work and put a general knowledge how instructions work.

2 Technical Aspects

2.1 Materials

Material	Price
Breadboard x1	8,84 TL
Jumper x40	3.72 TL
390R Resistor x1	0,03 TL
4x Seven Segment Display x1	7,45 TL
STM32G031K8 x1	137,59 TL
5mm LED Red x1	0,19 TL
Tactile Switch x1	0,28 TL

2.2 Method

The system supports 3 different button press type, which we will cover further into the section. Program will display 0000 on the 4 digit 7 segment display for a second. After that program will be on a loop. In this loop board will display 4005 on 4 digit 7 segment display and check for button interrupt. If the button is pressed. It will start the countdown from 9000 for the first countdown. In every step of countdown the program will generate a random number on the r7 register. After the countdown the random number will be displayed and will be saved into the responsible registers. Lastly if the pressed again in the countdown the program will paused until a further interrupt.

- Display 0000 on the segment display for a second.
- Display 4005 on the segment display until the button pressed.
- If the button pressed start the countdown from 9000 and display every number on the segment display.
- If the button pressed stop the countdown and display the number that stopped. After that wait for an another interrupt fort o contiune.
- After the countdown display the random number for 1 second.
- Display 0000 for a second and wait for an interrupt again and this the countdown starts from the random number generated by the board.

2.2.1 Random Number Algorithm

We used the Pseudo Code Random Number Generator algorithm which depends on several elements. The formula we will use:

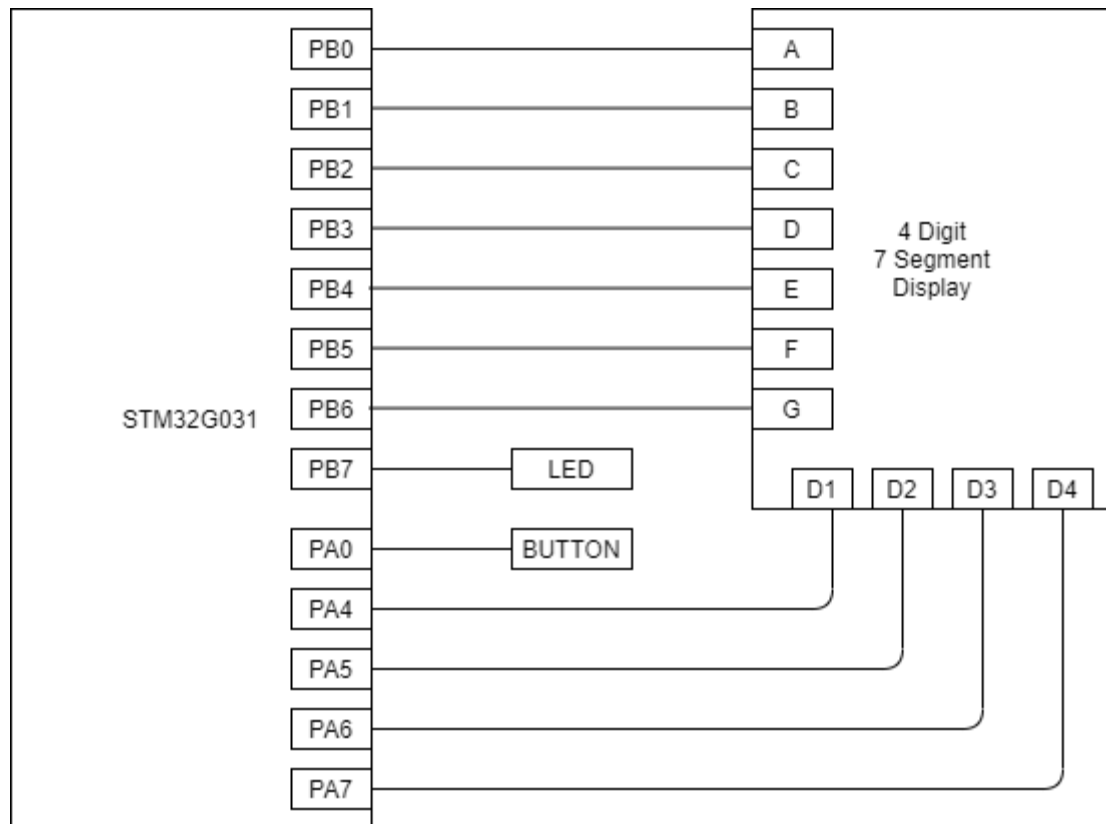
$$next = (next * a + c) \bmod m$$

a = 11035, c = 123456, m = 9000.

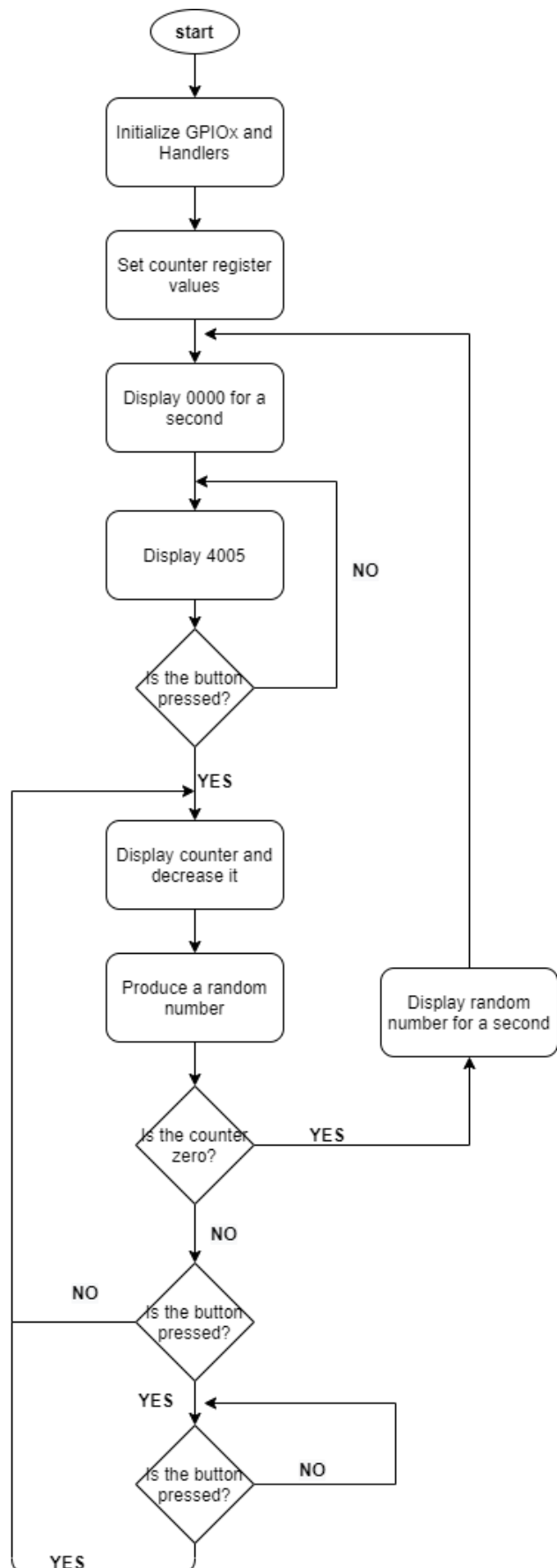
This formula will produce a number from 0 to 8999. After the countdown we add 1000 to number to maket this a number between 1000 and 9999.

2.3 Diagrams

2.3.1 Hardware Block Diagram



2.3.2 Software Block Diagram



3 Output

<https://youtu.be/DPQuOiU-mz8>

4 Conclusion

This project showed us a crucial points such as code length, data handling, register managment, bitwise module handling and problem solving.

Lets break it down and pieces shall we? First of all it was a great way to understand our C code that we programed on previous assignments. We had a chance to inspect how our code will work on machine wise and how it can be handled in different ways.

On the register handling part we learned how to preserve and manage our data for our usage. Since it is so easy to lose data that could be used in further instructions, it is important to store them in a good manner.

For code length we found clever methods fill the stack completely by shortening some parts but sacrificing readability. It can backfire for different readers but it efficient to shorten them.

Last setback that we faced is bitwise module activation. Putting right addresses and values into the registers played an important role on this process.

5 Appendix

```
6  /*
7   * project1.s
8   *
9   * author: Aziz Can Akkaya
10  * number: 171024005
11  */
12
13  .syntax unified
14  .cpu cortex-m0plus
15  .fpu softvfp
16  .thumb
17
18
19  /* make linker see this */
20  .global Reset_Handler
21
22  /* get these from linker script */
23  .word _sdata
24  .word _edata
25  .word _sbss
26  .word _ebss
27
28
29  /* define peripheral addresses from RM0444 page 57, Tables 3-4 */
30  .equ RCC_BASE,      (0x40021000)      // RCC base address
31  .equ RCC_IOPENR,    (RCC_BASE + (0x34)) // RCC IOPENR register offset
32
33  .equ GPIOC_BASE,    (0x50000800)      // GPIOC base address
34  .equ GPIOC_MODER,   (GPIOC_BASE + (0x00)) // GPIOC MODER register offset
35  .equ GPIOC_ODR,     (GPIOC_BASE + (0x14)) // GPIOC ODR register offset
36
37  .equ GPIOA_BASE,    (0x50000000)      //GPIOA base address
38  .equ GPIOA_MODER,   (GPIOA_BASE + (0x00)) //GPIOA MODER register offset
39  .equ GPIOA_ODR,     (GPIOA_BASE + (0x14)) //GPIOA ODR register
40  offset              (GPIOA_BASE + (0x10)) //GPIOA IDR register
41  offset
42
43  .equ GPIOB_BASE,    (0x50000400)      //GPIOB base address
44  .equ GPIOB_MODER,   (GPIOB_BASE + (0x00)) //GPIOB MODER register offset
45  .equ GPIOB_ODR,     (GPIOB_BASE + (0x14)) //GPIOB ODR register
46  offset
47
48  .equ leddelay,      (1000) //delay value
49
50  .equ a,              (11035)          //a value for random code generator
51  .equ c,              (12345)          //c value for random code generator
52  .equ m,              (9000)          //m value for random come generator
53
54  .equ ZERO,          (0x40)           //4-digit SSD pin set for GPIOA, nummber 0
55  .equ ONE,           (0x79)           //4-digit SSD pin set for GPIOA, nummber 1
56  .equ TWO,           (0x24)           //4-digit SSD pin set for GPIOA, nummber 2
57  .equ THREE,         (0x30)           //4-digit SSD pin set for GPIOA, nummber 3
58  .equ FOUR,          (0x19)           //4-digit SSD pin set for GPIOA, nummber 4
59  .equ FIVE,          (0x12)           //4-digit SSD pin set for GPIOA, nummber 5
```



```

58 .equ SIX,      (0x02)           //4-digit SSD pin set for GPIOA, nummber 6
59 .equ SEVEN,    (0x78)           //4-digit SSD pin set for GPIOA, nummber 7
60 .equ EIGHT,    (0x00)           //4-digit SSD pin set for GPIOA, nummber 8
61 .equ NINE,     (0x10)           //4-digit SSD pin set for GPIOA, nummber 9
62
63
64 /* vector table, +1 thumb mode */
65 .section .vectors
66 vector_table:
67     .word _estack                /*      Stack pointer */
68     .word Reset_Handler +1       /*      Reset handler */
69     .word Default_Handler +1     /*      NMI handler */
70     .word Default_Handler +1     /* HardFault handler */
71     /* add rest of them here if needed */
72
73
74 /* reset handler */
75 .section .text
76 Reset_Handler:
77     /* set stack pointer */
78     ldr r0, =_estack
79     mov sp, r0
80
81     /* initialize data and bss
82      * not necessary for rom only code
83      */
84     bl init_data
85     /* call main */
86     bl main
87     /* trap if returned */
88     b .
89
90
91 /* initialize data and bss sections */
92 .section .text
93 init_data:
94
95     /* copy rom to ram */
96     ldr r0, =_sdata
97     ldr r1, =_edata
98     ldr r2, =_sidata
99     movs r3, #0
100    b LoopCopyDataInit
101
102 CopyDataInit:
103     ldr r4, [r2, r3]
104     str r4, [r0, r3]
105     adds r3, r3, #4
106
107 LoopCopyDataInit:
108     adds r4, r0, r3
109     cmp r4, r1
110     bcc CopyDataInit
111
112 /* zero bss */
113 ldr r2, =_sbss
114 ldr r4, =_ebss
115 movs r3, #0
116 b LoopFillZerobss

```

```

117
118 FillZerobss:
119     str r3, [r2]
120     adds r2, r2, #4
121
122 LoopFillZerobss:
123     cmp r2, r4
124     bcc FillZerobss
125
126     bx lr
127
128
129 /* default handler */
130 .section .text
131 Default_Handler:
132     b Default_Handler
133
134
135 /* main function */
136 .section .text
137 main:
138     /* enable GPIOC clock, bit2 on IOPENR */
139     ldr r6, =RCC_IOPENR
140     ldr r5, [r6]
141     /* movs expects imm8, so this should be fine */
142     movs r4, 0x3
143     orrs r5, r5, r4
144     str r5, [r6]
145     //GPIOA PIN SET FOUR OUTPUT
146     ldr r6, =GPIOA_MODER
147     ldr r5, [r6]
148
149     movs r4, 0xFF
150     lsls r4, r4, #8
151     bics r5, r5, r4
152
153     movs r4, 0x55
154     lsls r4, r4, #8
155     orrs r5, r5, r4
156     ldr r4, =0xFFFFF0
157     ands r5, r5, r4
158     str r5, [r6]
159     //GPIOB PIN SET FOR OUTPUT
160     ldr r6, =GPIOB_MODER
161     ldr r5, [r6]
162     ldr r4, =0x5555
163     ands r5, r5, r4
164     str r5, [r6]
165     //GPIOA PIN SET FOR INPUT
166     ldr r6, =GPIOA_IDR
167     ldr r5, [r6]
168     ldr r4, =0x1
169     ands r5, r5, r4
170     str r5, [r6]
171     //COUNTER REGISTER SET
172     movs r0, #9
173     movs r1, #0
174     movs r2, #0
175     movs r3, #0

```

```

176
177 //Setting delay
178 set1:
179     ldr r7, =1000000
180 //diplay 0000 on the ssd for 1 second
181 start_loop:
182     ldr r6, = GPIOB_ODR
183     ldr r5, [r6]
184     movs r4, ZERO
185     orrs r5, r5, r4
186     str r5, [r6]
187
188     subs r7, r7, #1
189     ldr r6, = GPIOA_ODR
190     ldr r5, [r6]
191     movs r4, 0xF0
192     orrs r5, r5, r4
193     str r5, [r6]
194
195     ldr r4, =0
196     cmp r7, r4
197     bne start_loop
198 //clear output values
199 clear:
200     ldr r6, = GPIOA_ODR
201     ldr r5, [r6]
202     movs r4, 0x00
203     ands r5, r5, r4
204     str r5, [r6]
205
206     ldr r6, =GPIOB_ODR
207     ldr r5, [r6]
208     movs r4, 0x00
209     ands r5, r5, r4
210     str r5, [r6]
211 //b countdown
212 //button check
213 button:
214     bl id_mod
215     ldr r6, =GPIOA_IDR
216     ldr r5, [r6]
217     ldr r4, =0x1
218     ands r5, r5, r4
219     cmp r5, #1
220     beq countdown
221
222 //id display (4005) if the button is not pressed
223 id_mod:
224     push {lr}
225 //D1 OUTPUT
226     ldr r6, = GPIOB_ODR
227     ldr r5, [r6]
228     movs r4, FOUR
229     orrs r5, r5, r4
230     str r5, [r6]
231
232     ldr r6, = GPIOA_ODR
233     ldr r5, [r6]
234     movs r4, 0x10

```

```

235     orrs r5, r5, r4
236     str r5, [r6]
237
238     ldr r4, =leddelay
239
240     delay5:
241         subs r4, r4, #1
242         bne delay5
243         //CLEAR PINS
244         ldr r6, =GPIOA_ODR
245         ldr r5, [r6]
246         movs r4, 0x00
247         ands r5, r5, r4
248         str r5, [r6]
249
250         ldr r6, =GPIOB_ODR
251         ldr r5, [r6]
252         movs r4, 0x00
253         ands r5, r5, r4
254         str r5, [r6]
255         //D2 and D3 OUTPUT
256         ldr r6, =GPIOB_ODR
257         ldr r5, [r6]
258         movs r4, ZERO
259         orrs r5, r5, r4
260         str r5, [r6]
261
262         ldr r6, =GPIOA_ODR
263         ldr r5, [r6]
264         movs r4, 0x60
265         orrs r5, r5, r4
266         str r5, [r6]
267
268         ldr r4, =leddelay
269
270     delay7:
271         subs r4, r4, #1
272         bne delay7
273         //CLEAR PINS
274         ldr r6, =GPIOA_ODR
275         ldr r5, [r6]
276         movs r4, 0x00
277         ands r5, r5, r4
278         str r5, [r6]
279
280         ldr r6, =GPIOB_ODR
281         ldr r5, [r6]
282         movs r4, 0x00
283         ands r5, r5, r4
284         str r5, [r6]
285         //D4 OUTPUT
286         ldr r6, =GPIOB_ODR
287         ldr r5, [r6]
288         movs r4, FIVE
289         orrs r5, r5, r4
290         str r5, [r6]
291
292         ldr r6, =GPIOA_ODR
293         ldr r5, [r6]

```

```

294     movs r4, 0x80
295     orrs r5, r5, r4
296     str r5, [r6]
297
298     ldr r4, =leddelay
299
300     delay8:
301         subs r4, r4, #1
302         bne delay8
303
304     ldr r6, =GPIOA_ODR
305     ldr r5, [r6]
306     movs r4, 0x00
307     ands r5, r5, r4
308     str r5, [r6]
309
310     ldr r6, =GPIOB_ODR
311     ldr r5, [r6]
312     movs r4, 0x00
313     ands r5, r5, r4
314     str r5, [r6]
315     pop {pc}
316
317     //if the button pressed in start it will lead to this point
318     countdown:
319         bl button_check
320         bl check
321         bl led_on
322         bl display
323         bl random
324         //check and decrease control for units digit
325         cmp r3, #0
326         beq decrease_ten
327         subs r3, r3, #1
328         b countdown
329
330     decrease_ten:
331         //check and decrease control for tens digit
332         movs r3, #9
333         cmp r2, #0
334         beq decrease_hundred
335         subs r2, r2, #1
336         b countdown
337
338     decrease_hundred:
339         //check and decrease control for hundreds digit
340         movs r2, #9
341         cmp r1, #0
342         beq decrease_thousand
343         subs r1, r1, #1
344         b countdown
345
346     decrease_thousand:
347         //check and decrease control for thousands digit
348         movs r1, #9
349         cmp r0, #0
350         beq show1
351         subs r0, r0, #1
352         b countdown

```

```

353
354 //button check for pause
355 button_check:
356     push {lr}
357     ldr r6, =GPIOA_IDR
358     ldr r5, [r6]
359     ldr r4, =0x1
360     ands r5, r5, r4
361     cmp r5, #1
362     beq pause
363     pop {pc}
364
365 //pause loop, it will wait for an another press
366 pause:
367     bl display
368     bl led_on
369     ldr r6, =GPIOA_IDR
370     ldr r5, [r6]
371     ldr r4, =0x1
372     ands r5, r5, r4
373     cmp r5, #1
374     bne pause
375     pop {pc}
376
377 show1:
378     ldr r6, =0
379     ldr r4, =1000
380     adds r7, r7, r4
381     movs r5, r7
382     cmp r5, r4
383     bgt mod1
384
385 show2:
386     //setting up thousands value
387     movs r0, r6
388     ldr r6, =0
389     ldr r4, =100
390     cmp r5, r4
391     bgt mod2
392
393 show3:
394     //setting up hundreds value
395     movs r1, r6
396     ldr r6, =0
397     ldr r4, =10
398     cmp r5, r4
399     bgt mod3
400
401 show4:
402     //setting up tens value
403     movs r2, r6
404     ldr r6, =0
405     ldr r4, =1
406     cmp r5, r4
407     bgt mod4
408
409 show5:
410     //setting up units value
411     movs r3, r6

```

```

412     ldr r7, =1000
413     b display_result
414
415 display_result:
416     //displaying result for a second
417     bl led_off
418     bl display
419     subs r7, r7, #1
420     ldr r4, =0
421     cmp r7, r4
422     bne display_result
423     b set1
424
425     //general modding function for every digits
426 mod1:
427     subs r5, r5, r4
428     adds r6, r6, #1
429     cmp r5, r4
430     bge mod1
431     b show2
432
433 mod2:
434     subs r5, r5, r4
435     adds r6, r6, #1
436     cmp r5, r4
437     bge mod2
438     b show3
439
440
441 mod3:
442     subs r5, r5, r4
443     adds r6, r6, #1
444     cmp r5, r4
445     bge mod3
446     b show4
447
448 mod4:
449     subs r5, r5, r4
450     adds r6, r6, #1
451     cmp r5, r4
452     bge mod4
453     b show5
454
455     //random generator
456 random:
457     push {lr}
458     ldr r4, =a
459     //next*a
460     muls r7, r7, r4
461     ldr r4, =c
462     //next*a + c
463     adds r7, r7, r4
464     //mod m
465     ldr r4, =m
466     cmp r7, r4
467     bgt mod
468     pop {pc}
469
470 mod:

```

```

471     subs r7, r7, r4
472     cmp r7, r4
473     bge mod
474     pop {pc}
475
476     //checking if all the digits is 0 or not for the counter
477     check:
478         push {lr}
479         cmp r0, #0
480         beq check1
481         pop {pc}
482
483     check1:
484         cmp r1, #0
485         beq check2
486         pop {pc}
487
488     check2:
489         cmp r2, #0
490         beq check3
491         pop {pc}
492
493     check3:
494         cmp r3, #0
495         beq show1
496         pop {pc}
497
498     //led on function
499     led_on:
500         push {lr}
501         ldr r6, = GPIOB_ODR
502         ldr r5, [r6]
503         movs r4, 0x80
504         orrs r5, r5, r4
505         str r5, [r6]
506         pop {pc}
507
508     //led off function
509     led_off:
510         push {lr}
511         ldr r6, = GPIOB_ODR
512         ldr r5, [r6]
513         movs r4, 0x00
514         orrs r5, r5, r4
515         str r5, [r6]
516         pop {pc}
517
518     //switch case for to set r0 value
519     d1:
520         push {lr}
521         cmp r0, #0
522         beq zero
523
524         cmp r0, #1
525         beq one
526
527         cmp r0, #2
528         beq two
529

```



```

530    cmp r0, #3
531        beq three
532
533    cmp r0, #4
534        beq four
535
536    cmp r0, #5
537        beq five
538
539    cmp r0, #6
540        beq six
541
542    cmp r0, #7
543        beq seven
544
545        cmp r0, #8
546        beq eight
547
548        cmp r0, #9
549        beq nine
550
551    //switch case for to set r1 value
552    d2:
553        push {lr}
554        cmp r1, #0
555        beq zero
556
557        cmp r1, #1
558        beq one
559
560    cmp r1, #2
561        beq two
562
563    cmp r1, #3
564        beq three
565
566    cmp r1, #4
567        beq four
568
569    cmp r1, #5
570        beq five
571
572    cmp r1, #6
573        beq six
574
575    cmp r1, #7
576        beq seven
577
578        cmp r1, #8
579        beq eight
580
581        cmp r1, #9
582        beq nine
583
584    //switch case for to set r2 value
585    d3:
586        push {lr}
587        cmp r2, #0
588        beq zero

```

```

589
590     cmp r2, #1
591     beq one
592
593     cmp r2, #2
594     beq two
595
596     cmp r2, #3
597     beq three
598
599     cmp r2, #4
600     beq four
601
602     cmp r2, #5
603     beq five
604
605     cmp r2, #6
606     beq six
607
608     cmp r2, #7
609     beq seven
610
611     cmp r2, #8
612     beq eight
613
614     cmp r2, #9
615     beq nine
616
617 //switch case for to set r4 value
618 d4:
619     push {lr}
620     cmp r3, #0
621     beq zero
622
623     cmp r3, #1
624     beq one
625
626     cmp r3, #2
627     beq two
628
629     cmp r3, #3
630     beq three
631
632     cmp r3, #4
633     beq four
634
635     cmp r3, #5
636     beq five
637
638     cmp r3, #6
639     beq six
640
641     cmp r3, #7
642     beq seven
643
644     cmp r3, #8
645     beq eight
646
647     cmp r3, #9

```

```

648     beq nine
649
650 //setting values for GPIOA pin output
651 zero:
652     movs r4, ZERO
653     pop {pc}
654
655 one:
656     movs r4, ONE
657     pop {pc}
658
659 two:
660     movs r4, TWO
661     pop {pc}
662
663 three:
664     movs r4, THREE
665     pop {pc}
666
667 four:
668     movs r4, FOUR
669     pop {pc}
670
671 five:
672     movs r4, FIVE
673     pop {pc}
674
675 six:
676     movs r4, SIX
677     pop {pc}
678
679 seven:
680     movs r4, SEVEN
681     pop {pc}
682
683 eight:
684     movs r4, EIGHT
685     pop {pc}
686
687 nine:
688     movs r4, NINE
689     pop {pc}
690
691 //digit display function
692 display:
693     push {lr}
694     ldr r6, = GPIOA_ODR
695     ldr r5, [r6]
696     movs r4, 0x10
697     orrs r5, r5, r4
698     str r5, [r6]
699
700     ldr r6, = GPIOB_ODR
701     ldr r5, [r6]
702     bl d1
703     orrs r5, r5, r4
704     str r5, [r6]
705
706     ldr r4, =leddelay

```

```

707
708 delay1:
709     subs r4, r4, #1
710     bne delay1
711
712     ldr r6, = GPIOA_ODR
713     ldr r5, [r6]
714     movs r4, 0x00
715     ands r5, r5, r4
716     str r5, [r6]
717
718     ldr r6, = GPIOB_ODR
719     ldr r5, [r6]
720     movs r4, 0x00
721     ands r5, r5, r4
722     str r5, [r6]
723
724     ldr r6, = GPIOA_ODR
725     ldr r5, [r6]
726     movs r4, 0x20
727     orrs r5, r5, r4
728     str r5, [r6]
729
730     ldr r6, = GPIOB_ODR
731     ldr r5, [r6]
732     bl d2
733     orrs r5, r5, r4
734     str r5, [r6]
735
736     ldr r4, =leddelay
737
738 delay2:
739     subs r4, r4, #1
740     bne delay2
741
742     ldr r6, = GPIOA_ODR
743     ldr r5, [r6]
744     movs r4, 0x00
745     ands r5, r5, r4
746     str r5, [r6]
747
748     ldr r6, = GPIOB_ODR
749     ldr r5, [r6]
750     movs r4, 0x00
751     ands r5, r5, r4
752     str r5, [r6]
753
754     ldr r6, = GPIOA_ODR
755     ldr r5, [r6]
756     movs r4, 0x40
757     orrs r5, r5, r4
758     str r5, [r6]
759
760     ldr r6, = GPIOB_ODR
761     ldr r5, [r6]
762     bl d3 //movs r4, data
763     orrs r5, r5, r4
764     str r5, [r6]
765

```

```

766     ldr r4, =leddelay
767
768     delay3:
769     subs r4, r4, #1
770     bne delay3
771
772     ldr r6, = GPIOA_ODR
773     ldr r5, [r6]
774     movs r4, 0x00
775     ands r5, r5, r4
776     str r5, [r6]
777
778     ldr r6, = GPIOB_ODR
779     ldr r5, [r6]
780     movs r4, 0x00
781     ands r5, r5, r4
782     str r5, [r6]
783
784     ldr r6, = GPIOA_ODR
785     ldr r5, [r6]
786     movs r4, 0x80
787     orrs r5, r5, r4
788     str r5, [r6]
789
790     ldr r6, = GPIOB_ODR
791     ldr r5, [r6]
792     bl d4
793     orrs r5, r5, r4
794     str r5, [r6]
795
796     ldr r4, =leddelay
797
798     delay4:
799     subs r4, r4, #1
800     bne delay4
801
802     ldr r6, = GPIOA_ODR
803     ldr r5, [r6]
804     movs r4, 0x00
805     ands r5, r5, r4
806     str r5, [r6]
807
808     ldr r6, = GPIOB_ODR
809     ldr r5, [r6]
810     movs r4, 0x00
811     ands r5, r5, r4
812     str r5, [r6]
813     pop {pc}
814
815

```

6 References

- STM32G031K8 Data Sheet
- RM0444 Reference Manual for STM32G0x1 Devices
- ARMv6-M Architecture Reference Manual
- [https://www.geeksforgeeks.org/pseudo-random-number-generator-prng/#:~:text=Pseudo%20Random%20Number%20Generator\(PRNG\)%20refers%20to%20an%20algorithm%20that,state%20using%20a%20seed%20state.](https://www.geeksforgeeks.org/pseudo-random-number-generator-prng/#:~:text=Pseudo%20Random%20Number%20Generator(PRNG)%20refers%20to%20an%20algorithm%20that,state%20using%20a%20seed%20state.)
- <https://www.projehocam.com/arduino-ile-4-digit-7-segment-display-uygulamasi/>
- https://www.keil.com/support/man/docs/armasm/armasm_dom1361289850039.htm