GEBZE TECHNICAL UNIVERSITY

ENGINEERING FACULTY

ELECTRONICS ENGINEERING

ELEC 335

MICROPROCESSORS

LAB 02

Name - Surname	Yağmur DERYA
Student ID	171024011

1. Introduction

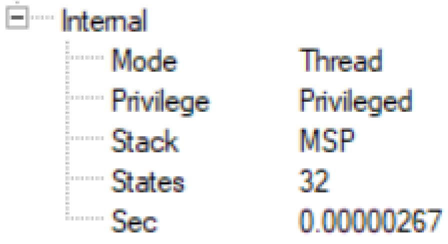
In this lab, it is aimed to write assembly code and practice connecting basic components to the board and see how it works.

2. Problems

2.1. Problem 1 – Toggle

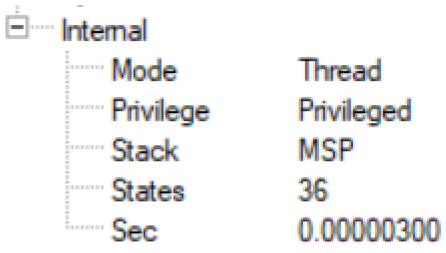
In this problem, the LED was connected to PA8. First, the code was written on STM32CubeIDE and then debugged. After being sure the code was working, “resume” button was clicked to see if the LED toggles. To see how it works, you could check the “Links” tittle.

In Lab01, approximate loop number was calculated in Problem 4 by examining how many seconds one delay loop takes.



```
Internal
  Mode        Thread
  Privilege   Privileged
  Stack       MSP
  States      32
  Sec         0.00000267
```

Figure 1. Before delay.



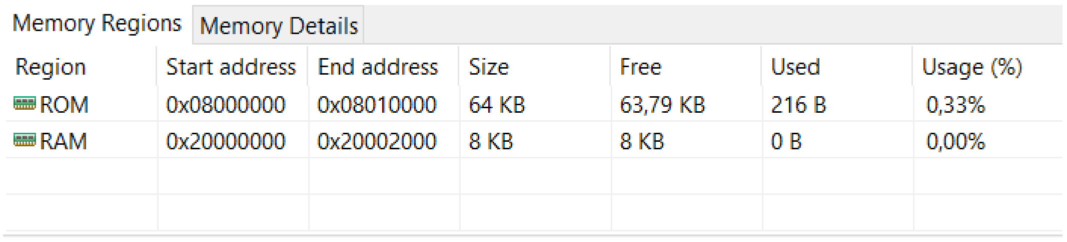
```
Internal
  Mode        Thread
  Privilege   Privileged
  Stack       MSP
  States      36
  Sec         0.00000300
```

Figure 2. After delay.

The loop takes $(300 - 267)x10^{-8} = 33x10^{-8}$ seconds. To make delay 1 second we need $\frac{1}{33x10^{-8}} = 3030303$ loops.

```
/*
 * ELEC335 Lab02
 * problem1.s
 * Yagmur Derya 171024011
 *
*/

.syntax unified
.cpu cortex-m0
.fpu softvfp
.thumb


/* make linker see this */
.global Reset_Handler

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


/* define peripheral addresses from RM0444 page 57, Tables 3-4 */
.equ RCC_BASE,         (0x40021000)          // RCC base address
.equ RCC_IOPENR,       (RCC_BASE   + (0x34)) // RCC IOPENR offset

.equ GPIOA_BASE,       (0x50000000)          // GPIOA base address
.equ GPIOA_MODER,      (GPIOA_BASE + (0x00)) // GPIOA MODER offset
```

```
.equ GPIOA_ODR,        (GPIOA_BASE + (0x14)) // GPIOA ODR offset
.equ GPIOA_IDR,        (GPIOA_BASE + (0x10)) // GPIOA IDR offset


/* vector table, +1 thumb mode *f/
.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

       /* initialize data and bss
        * not necessary for rom only code
        * */
       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
main:
        /* enable GPIOA clock, bit1 on IOPENR */
        ldr r6, =RCC_IOPENR
        ldr r5, [r6]
        /* movs expects imm8, so this should be fine */
        movs r4, 0x1
        orrs r5, r5, r4
        str r5, [r6]

        /* setup PA8 for led 01 for bits 17-16 in MODER */
        ldr r6, =GPIOA_MODER
        ldr r5, [r6]
        /* cannot do with movs, so use pc relative */
        ldr r4, =0x30000
        mvns r4, r4
        ands r5, r5, r4
        ldr r4, =0x10000
        orrs r5, r5, r4
        str r5, [r6]

toggle:
        /* turn on led connected to A8 in ODR */
        ldr r6, =GPIOA_ODR
        ldr r5, [r6]
        ldr r4, =0x100
        orrs r5, r5, r4
        str r5, [r6]

        ldr r0, =#3030303 /* counter for delay */
        bl delay

        /* turn off led connected to A8 in ODR */
        ldr r6, = GPIOA_ODR
        ldr r5, [r6]
        movs r4, #0x0
        ands r5, r5, r4
        str r5, [r6]


        ldr r0, =#3030303 /* counter for delay */
        bl delay

        b toggle

delay:
        subs r0, r0, #1
        bne delay
        bx lr

        /* for(;;); */
        b .

        /* this should never get executed */
        nop
```



Region	Start address	End address	Size	Free	Used	Usage (%)
ROM	0x08000000	0x08010000	64 KB	63,79 KB	216 B	0,33%
RAM	0x20000000	0x20002000	8 KB	8 KB	0 B	0,00%

Figure 3. Memory usage.

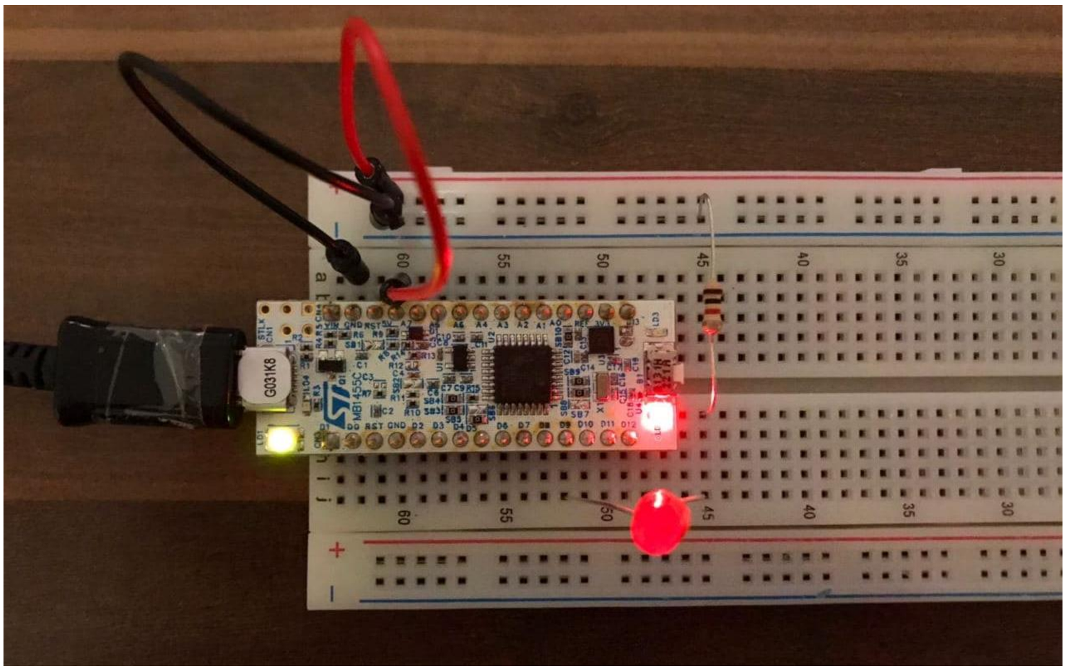


Figure 4. Problem 1 on board circuit.

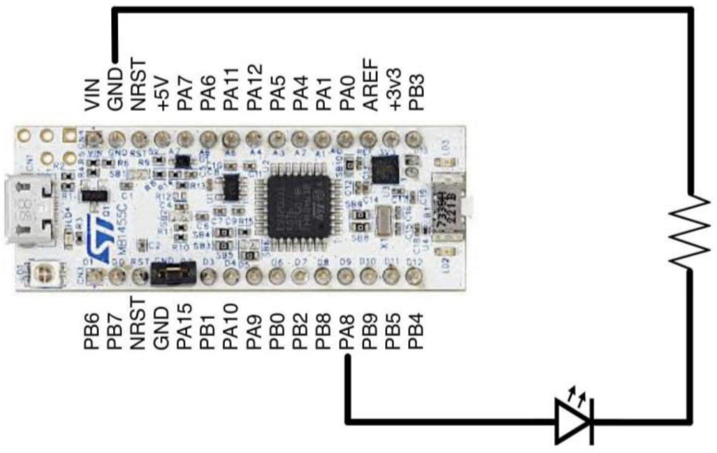


Figure 5. Schematic diagram.

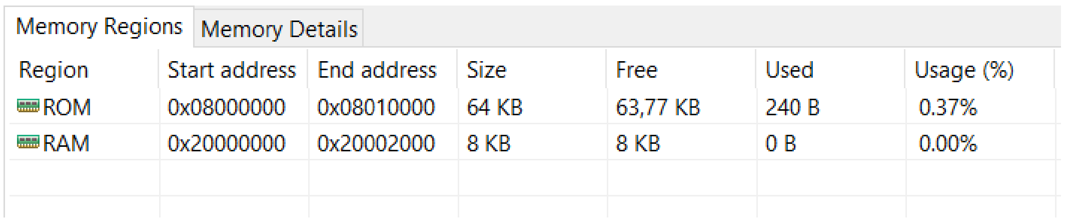
2.2. Problem 2 – Button

In this problem, a push button was added to the board. One leg of the button was connected to 5V when other leg was connected to series pin PA11 and resistor to ground.

When the button was pushed, because of the 5V input data register would be 1 too. When the button was not pushed, IDR would be 0. By checking this register, the led could be set or reset.

```
/*
 * ELEC335 Lab02
 * problem2.s
 * Yagmur Derya 171024011
 *
*/

.syntax unified
.cpu cortex-m0
.fpu softvfp
.thumb


/* make linker see this */
.global Reset_Handler

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


/* define peripheral addresses from RM0444 page 57, Tables 3-4 */
.equ RCC_BASE,          (0x40021000)          // RCC base address
.equ RCC_IOPENR,        (RCC_BASE   + (0x34)) // RCC IOPENR offset

.equ GPIOA_BASE,        (0x50000000)          // GPIOA base address
.equ GPIOA_MODER,       (GPIOA_BASE + (0x00)) // GPIOA MODER offset
.equ GPIOA_ODR,         (GPIOA_BASE + (0x14)) // GPIOA ODR offset
.equ GPIOA_IDR,         (GPIOA_BASE + (0x10)) // GPIOA IDR offset


/* vector table, +1 thumb mode *f/
.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

       /* initialize data and bss
        * not necessary for rom only code
        * */
       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
main:
        /* enable GPIOA clock, bit1 on IOPENR */
        ldr r6, =RCC_IOPENR
        ldr r5, [r6]
        /* movs expects imm8, so this should be fine */
        movs r4, 0x1
        orrs r5, r5, r4
        str r5, [r6]

        /* setup PA8 for led 01 for bits 17-16 in MODER */
        ldr r6, =GPIOA_MODER
        ldr r5, [r6]
        /* cannot do with movs, so use pc relative */
        ldr r4, =0x30000
        mvns r4, r4
        ands r5, r5, r4
        ldr r4, =0x10000
        orrs r5, r5, r4
        str r5, [r6]

        /* setup PA11 for push button for bits 23-22 in MODER*/
        ldr r6, =GPIOA_MODER
        ldr r5, [r6]
        ldr r4, =0xC00000
        mvns r4, r4
        ands r5, r5, r4
        ldr r4, =0x400000
        orrs r5, r5, r4
        str r5, [r6]

button:
        ldr r6, =GPIOA_IDR
        ldr r5, [r6]
        lsrs r5, r5, #11
        movs r4, #0x1
        ands r5, r5, r4

        cmp r5, #0x1
        beq led_on
        bne led_off

led_on:
        /* turn on led connected to A8 in ODR */
        ldr r6, =GPIOA_ODR
        ldr r5, [r6]
        ldr r4, =0x100
        orrs r5, r5, r4
        str r5, [r6]
        b button

led_off:
        /* turn off led connected to A8 in ODR */
        ldr r6, = GPIOA_ODR
        ldr r5, [r6]
        movs r4, #0x0
        ands r5, r5, r4
        str r5, [r6]
        b button

        /* for(;;); */
        b .

        /* this should never get executed */
        nop
```



Region	Start address	End address	Size	Free	Used	Usage (%)
ROM	0x08000000	0x08010000	64 KB	63,77 KB	240 B	0.37%
RAM	0x20000000	0x20002000	8 KB	8 KB	0 B	0.00%

Figure 6. Memory usage.

Although there isn't big difference, the memory usage in this problem is more than problem 1. The reason of the difference is the instruction number. To check button input and its value, the circuit has more instructions.

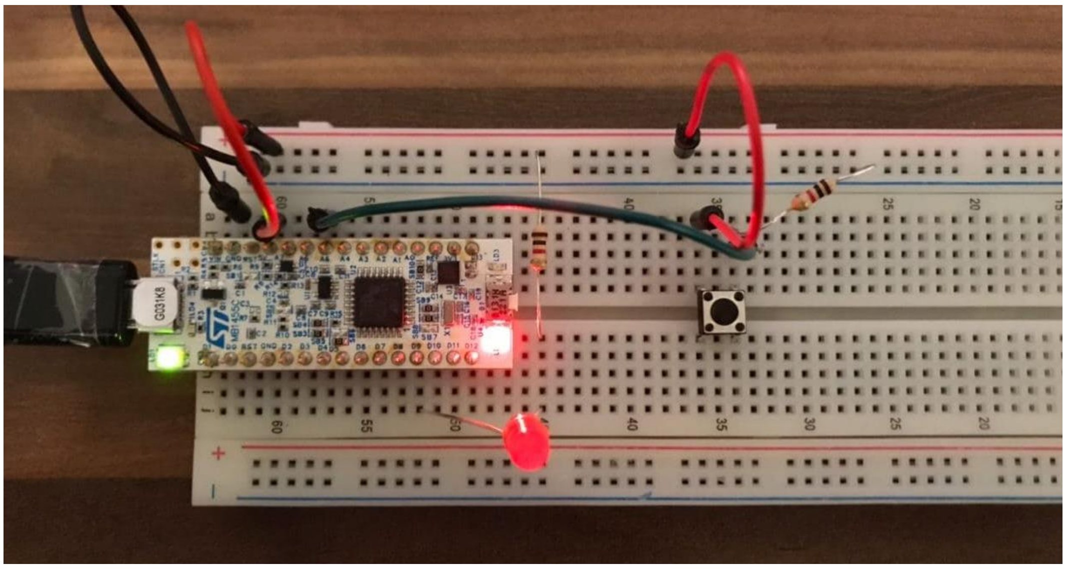


Figure 7. Problem 2 circuit on board, button = 0.

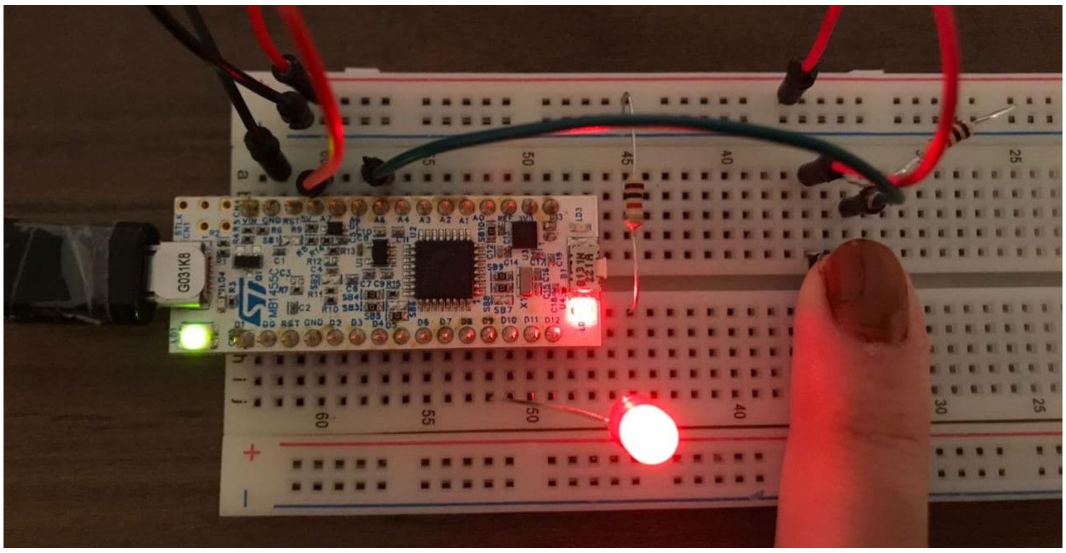


Figure 8. Problem 2 circuit on board, button = 1.

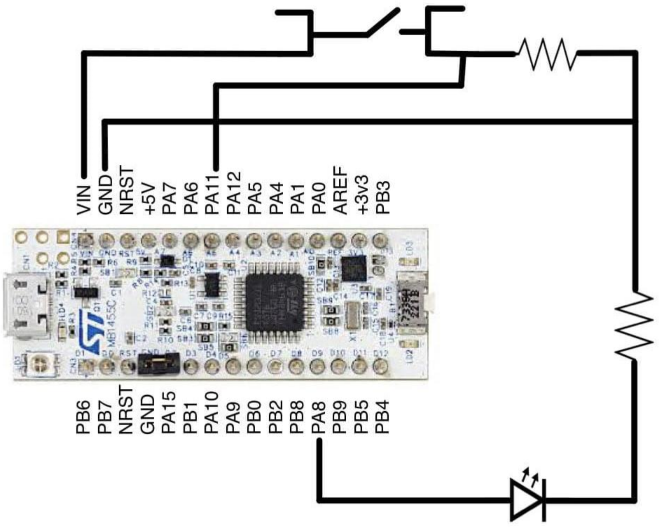


Figure 9. Schematic diagram.

2.3. Problem 3 – 7-LED Toggle

Actually, in this problem, it was asked to toggle 8 LEDs at the same time but because I have only 7, 7 LEDs were toggled.

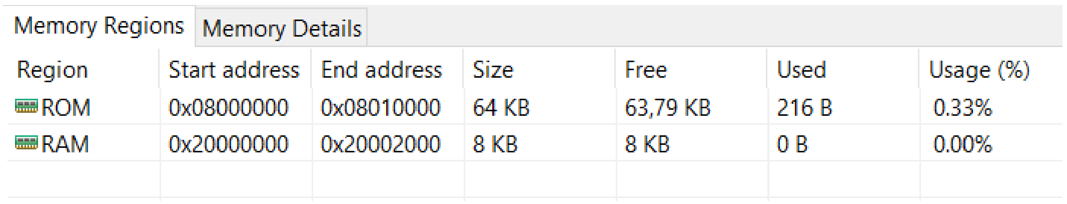
Since they toggle at the same time, it wasn't necessary to set them one by one. Because of that, the LEDs mode register (MODER) and output data register (ODR) assigned at once.

The LED which was connected to PA10 has low light but it toggles. To prove it, the LED was connected and disconnected in the video.

```
/*
 * ELEC335 Lab02
 * problem3.s
 * Yagmur Derya 171024011
 *
*/

.syntax unified
.cpu cortex-m0
.fpu softvfp
.thumb


/* make linker see this */
.global Reset_Handler

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


/* define peripheral addresses from RM0444 page 57, Tables 3-4 */
.equ RCC_BASE,         (0x40021000)          // RCC base address
.equ RCC_IOPENR,       (RCC_BASE   + (0x34)) // RCC IOPENR offset

.equ GPIOA_BASE,       (0x50000000)          // GPIOA base address
.equ GPIOA_MODER,      (GPIOA_BASE + (0x00)) // GPIOA MODER offset
.equ GPIOA_ODR,        (GPIOA_BASE + (0x14)) // GPIOA ODR offset
.equ GPIOA_IDR,        (GPIOA_BASE + (0x10)) // GPIOA IDR offset


/* vector table, +1 thumb mode *f/
.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

      /* initialize data and bss
       * not necessary for rom only code
       * */
      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
main:
       /* enable GPIOA clock, bit1 on IOPENR */
       ldr r6, =RCC_IOPENR
       ldr r5, [r6]
       /* movs expects imm8, so this should be fine */
       movs r4, 0x1
       orrs r5, r5, r4
       str r5, [r6]

       /* setup PA15-11-10-8-7-6-1 */
       ldr r6, =GPIOA_MODER
       ldr r5, [r6]
       /* cannot do with movs, so use pc relative */
       ldr r4, =0xC0F3F00C /* 1100_0000_1111_0011_1111_0000_0000_1100
*/
       mvns r4, r4
       ands r5, r5, r4
       ldr r4, =0x40515004 /* xx00_0000_xxxx_00xx_xxxx_0000_0000_xx00
*/
       orrs r5, r5, r4
       str r5, [r6]

toggle:
       /* turn on leds connected to A15-11-10-8-7-6-1 in ODR */
       ldr r6, =GPIOA_ODR
       ldr r5, [r6]
       ldr r4, =0x8DC2 /* x000_xx0x_xx00_00x0 */
       orrs r5, r5, r4
       str r5, [r6]

       ldr r0, =#3030303 /* counter for delay */
       bl delay

       /* turn off leds connected to A15-11-10-8-7-6-1 in ODR */
       ldr r6, = GPIOA_ODR
       ldr r5, [r6]
       movs r4, #0x0
       ands r5, r5, r4
       str r5, [r6]


       ldr r0, =#3030303 /* counter for delay */
       bl delay

       b toggle

delay:
       subs r0, r0, #1
       bne delay
       bx lr

       /* for(;;); */
       b .

       /* this should never get executed */
       nop
```



Region	Start address	End address	Size	Free	Used	Usage (%)
ROM	0x08000000	0x08010000	64 KB	63,79 KB	216 B	0.33%
RAM	0x20000000	0x20002000	8 KB	8 KB	0 B	0.00%

Figure 10. Memory usage.

Memory usage is equals problem 1s and it was expected because the instruction numbers are equal. Between these two problems, only R4 values are different.

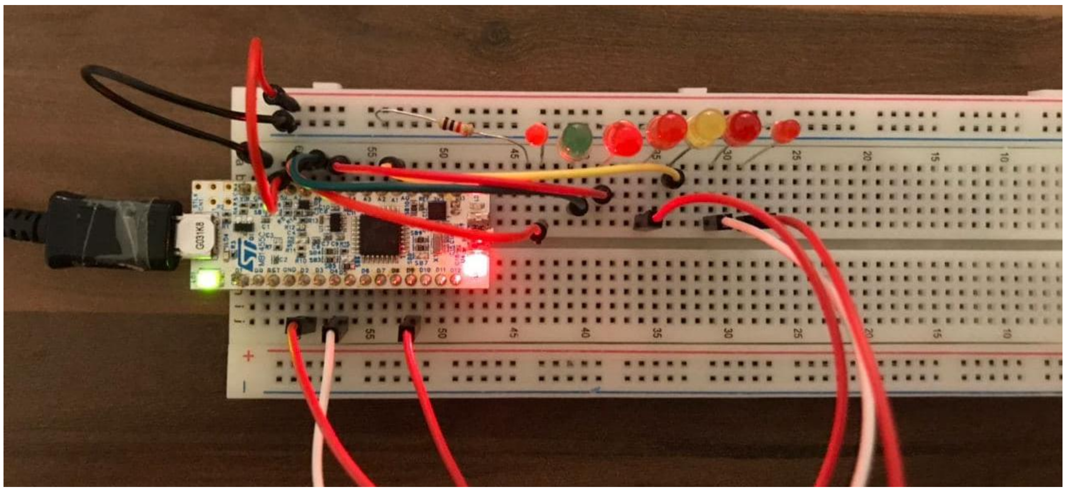


Figure 11. Problem 3 on board circuit.

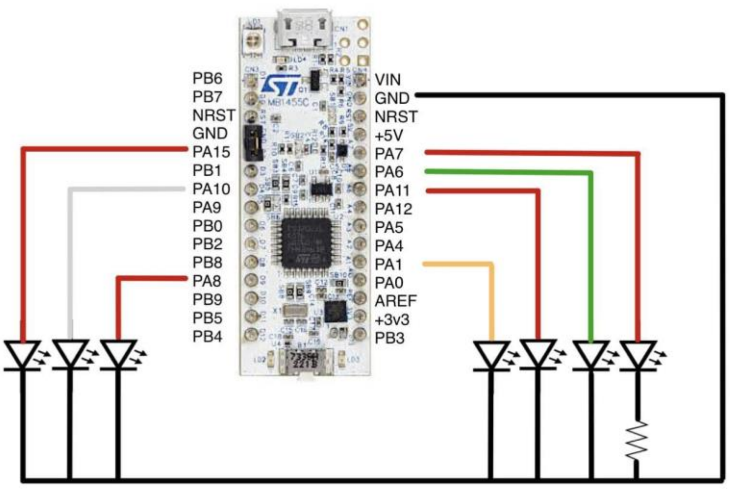


Figure 12. Schematic diagram.

2.4. Problem 4 – Shift Pattern

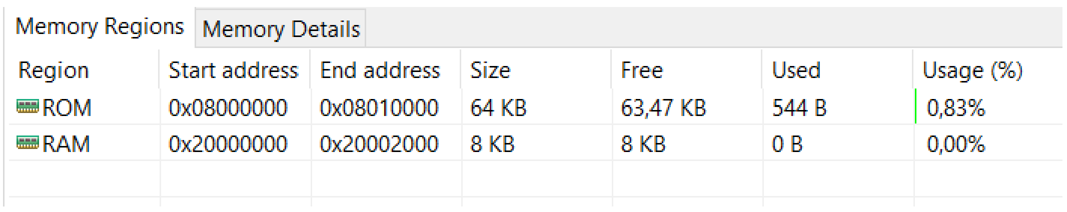
The loop takes $(300-267)x10^{-8} = 33x10^{-8}$ seconds. To make delay 100 ms (=10^{-1}s) we need $\frac{10^{-1}}{33x10^{-8}}$ ~ 303030 loops.

```
/*
 * ELEC335 Lab02
 * problem4.s
 * Yagmur Derya 171024011
 *
*/

.syntax unified
.cpu cortex-m0
.fpu softvfp
.thumb


/* make linker see this */
.global Reset_Handler

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


/* define peripheral addresses from RM0444 page 57, Tables 3-4 */
.equ RCC_BASE,         (0x40021000)          // RCC base address
.equ RCC_IOPENR,       (RCC_BASE   + (0x34)) // RCC IOPENR offset

.equ GPIOA_BASE,       (0x50000000)          // GPIOA base address
.equ GPIOA_MODER,      (GPIOA_BASE + (0x00)) // GPIOA MODER offset
.equ GPIOA_ODR,        (GPIOA_BASE + (0x14)) // GPIOA ODR offset
.equ GPIOA_IDR,        (GPIOA_BASE + (0x10)) // GPIOA IDR offset


/* vector table, +1 thumb mode *f/
.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

      /* initialize data and bss
       * not necessary for rom only code
```

```
        * */
       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
main:
       /* enable GPIOA clock, bit1 on IOPENR */
       ldr r6, =RCC_IOPENR
       ldr r5, [r6]
       /* movs expects imm8, so this should be fine */
```

```
        movs r4, 0x1
        orrs r5, r5, r4
        str r5, [r6]

        /* setup PA15-11-10-8-7-6-1 */
        ldr r6, =GPIOA_MODER
        ldr r5, [r6]
        /* cannot do with movs, so use pc relative */
        ldr r4, =0xC0F3F00C /* 1100_0000_1111_0011_1111_0000_0000_1100
*/
        mvns r4, r4
        ands r5, r5, r4
        ldr r4, =0x40515004 /* xx00_0000_xxxx_00xx_xxxx_0000_0000_xx00
*/
        orrs r5, r5, r4
        str r5, [r6]
        /* setup PA4 for push button for bits 9-8 in MODER*/
        ldr r6, =GPIOA_MODER
        ldr r5, [r6]
        ldr r4, =0x300
        mvns r4, r4
        ands r5, r5, r4
        ldr r4, =0x100
        orrs r5, r5, r4
        str r5, [r6]

button:
        ldr r6, =GPIOA_IDR
        ldr r5, [r6]
        lsrs r5, r5, #4
        movs r4, #0x1
        ands r5, r5, r4

        cmp r5, #0x1
        beq toggleR
        bne toggleL

toggleL:
        /* turn on leds connected to A11-7-6 in ODR */
        ldr r6, =GPIOA_ODR
        ldr r5, [r6]
        ldr r4, =0x8C0 /* x000_xx00_0000 */
        orrs r5, r5, r4
        str r5, [r6]

        ldr r0, =#303030 /* counter for delay */
        bl delay

        bl off

        /* turn on leds connected to A15-11-6 in ODR */
        ldr r6, =GPIOA_ODR
        ldr r5, [r6]
        ldr r4, =0x8840 /* x000_x000_0x00_0000 */
        orrs r5, r5, r4
        str r5, [r6]

        ldr r0, =#303030 /* counter for delay */
        bl delay
        bl off

        /* turn on leds connected to A15-11-1 in ODR */
        ldr r6, =GPIOA_ODR
        ldr r5, [r6]
        ldr r4, =0x8802 /* x000_x000_0000_00x0 */
        orrs r5, r5, r4
        str r5, [r6]

        ldr r0, =#303030 /* counter for delay */
        bl delay

        bl off

        /* turn on leds connected to A15-10-1 in ODR */
        ldr r6, =GPIOA_ODR
        ldr r5, [r6]
        ldr r4, =0x8402 /* x000_0x00_0000_00x0 */
        orrs r5, r5, r4
        str r5, [r6]

        ldr r0, =#303030 /* counter for delay */
        bl delay

        bl off

        /* turn on leds connected to A10-8-1 in ODR */
        ldr r6, =GPIOA_ODR
        ldr r5, [r6]
        ldr r4, =0x502 /* 0x0x_0000_00x0 */
        orrs r5, r5, r4
        str r5, [r6]

        ldr r0, =#303030 /* counter for delay */
        bl delay

        bl off

        /* turn on leds connected to A10-8-7 in ODR */
        ldr r6, =GPIOA_ODR
        ldr r5, [r6]
        ldr r4, =0x580 /* 0x0x_x000_0000 */
        orrs r5, r5, r4
        str r5, [r6]

        ldr r0, =#303030 /* counter for delay */
        bl delay

        bl off

        /* turn on leds connected to A8-7-6- in ODR */
        ldr r6, =GPIOA_ODR
        ldr r5, [r6]
        ldr r4, =0x1C0 /* 000x_xx00_0000 */
        orrs r5, r5, r4
        str r5, [r6]

        ldr r0, =#303030 /* counter for delay */
        bl delay

        bl off

        b button


toggleR:
        /* turn on leds connected to A8-7-6- in ODR */
        ldr r6, =GPIOA_ODR
        ldr r5, [r6]
        ldr r4, =0x1C0 /* 000x_xx00_0000 */
        orrs r5, r5, r4
        str r5, [r6]

        ldr r0, =#303030 /* counter for delay */
        bl delay

        bl off

        /* turn on leds connected to A10-8-7 in ODR */
        ldr r6, =GPIOA_ODR
        ldr r5, [r6]
        ldr r4, =0x580 /* 0x0x_x000_0000 */
        orrs r5, r5, r4
        str r5, [r6]

        ldr r0, =#303030 /* counter for delay */
        bl delay

        bl off

        /* turn on leds connected to A10-8-1 in ODR */
        ldr r6, =GPIOA_ODR
        ldr r5, [r6]
        ldr r4, =0x502 /* 0x0x_0000_00x0 */
        orrs r5, r5, r4
        str r5, [r6]

        ldr r0, =#303030 /* counter for delay */
        bl delay

        bl off

        /* turn on leds connected to A15-10-1 in ODR */
        ldr r6, =GPIOA_ODR
        ldr r5, [r6]
        ldr r4, =0x8402 /* x000_0x00_0000_00x0 */
        orrs r5, r5, r4
        str r5, [r6]

        ldr r0, =#303030 /* counter for delay */
        bl delay

        bl off

        /* turn on leds connected to A15-11-1 in ODR */
        ldr r6, =GPIOA_ODR
        ldr r5, [r6]
```

```
	ldr r4, =0x8802 /* x000_x000_0000_00x0 */
	orrs r5, r5, r4
	str r5, [r6]

	ldr r0, =#303030 /* counter for delay */
	bl delay

	bl off

	/* turn on leds connected to A15-11-6 in ODR */
	ldr r6, =GPIOA_ODR
	ldr r5, [r6]
	ldr r4, =0x8840 /* x000_x000_0x00_0000 */
	orrs r5, r5, r4
	str r5, [r6]

	ldr r0, =#303030 /* counter for delay */
	bl delay

	bl off

	/* turn on leds connected to A11-7-6 in ODR */
	ldr r6, =GPIOA_ODR
	ldr r5, [r6]
	ldr r4, =0x8C0 /* x000_xx00_0000 */
	orrs r5, r5, r4
	str r5, [r6]

	ldr r0, =#303030 /* counter for delay */
	bl delay

	bl off
	b button

off:
	/* turn off leds connected to A15-11-10-8-7-6-1 in ODR */
	ldr r6, = GPIOA_ODR
	ldr r5, [r6]
	movs r4, #0x0
	ands r5, r5, r4
	str r5, [r6]
	bx lr

delay:
	subs r0, r0, #1
	bne delay
	bx lr

	/* for(;;); */
	b .

	/* this should never get executed */
	nop
```


Memory Regions | Memory Details

Region	Start address	End address	Size	Free	Used	Usage (%)
ROM	0x08000000	0x08010000	64 KB	63,47 KB	544 B	0,83%
RAM	0x20000000	0x20002000	8 KB	8 KB	0 B	0,00%

Figure 13. Memory usage.

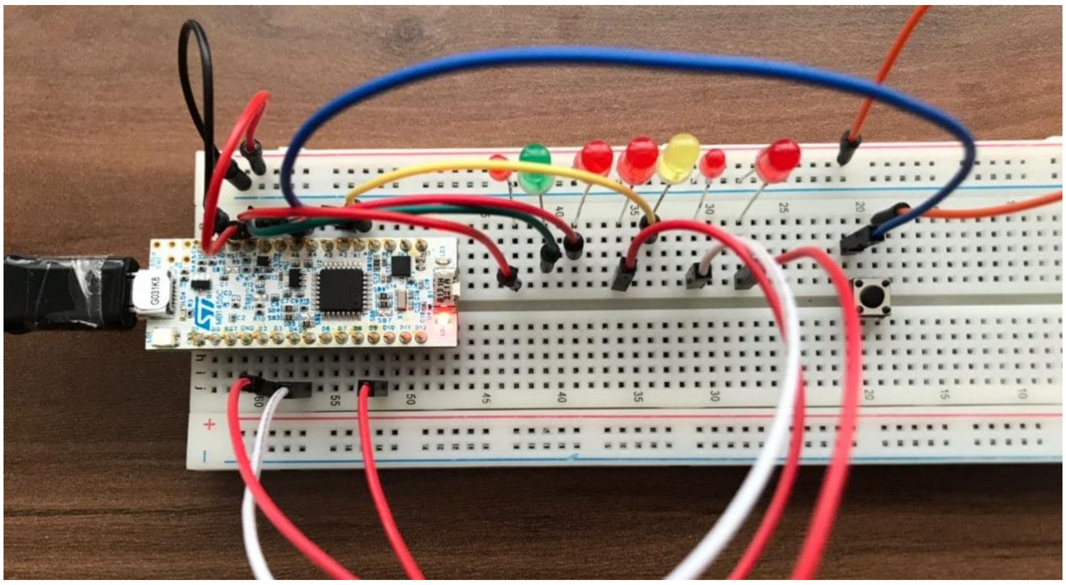


Figure 14. On board circuit.

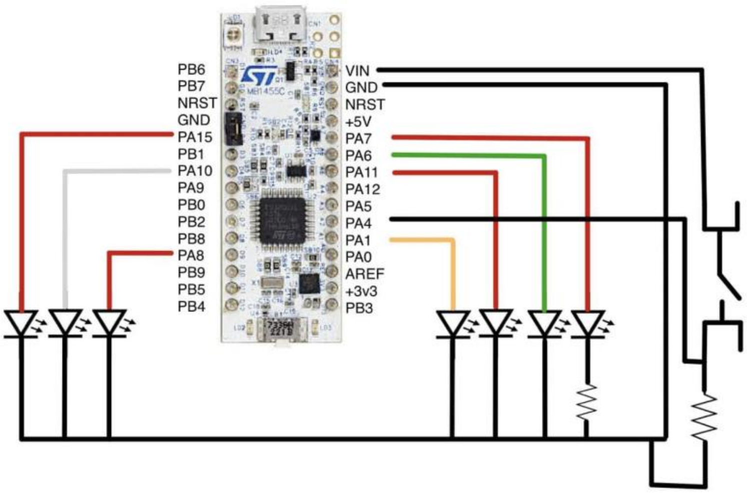


Figure 15. Schematic diagram.

3. Links

Problem 1: https://youtu.be/WXCk7BstPHs
Problem 2: https://youtu.be/kFloIBLH_HE
Problem 3: https://youtu.be/unpDpJO2XA8
Problem 4: https://youtu.be/ONP7LzG89N4

4. References

[1] RM0444 Reference manuel
[2] https://github.com/fcayci/stm32g0