

ELM 334 - LAB #1

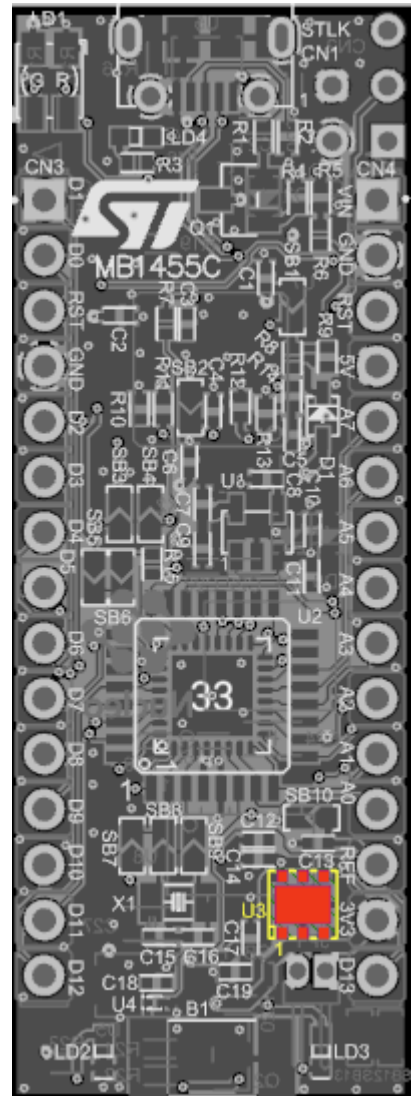
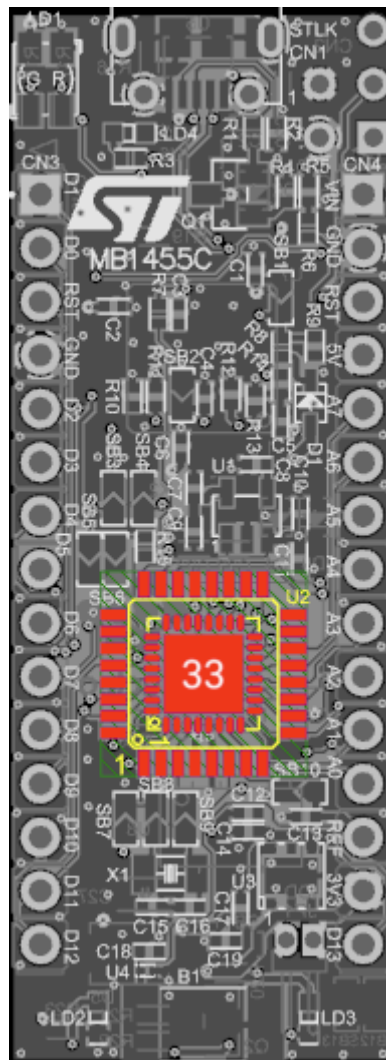
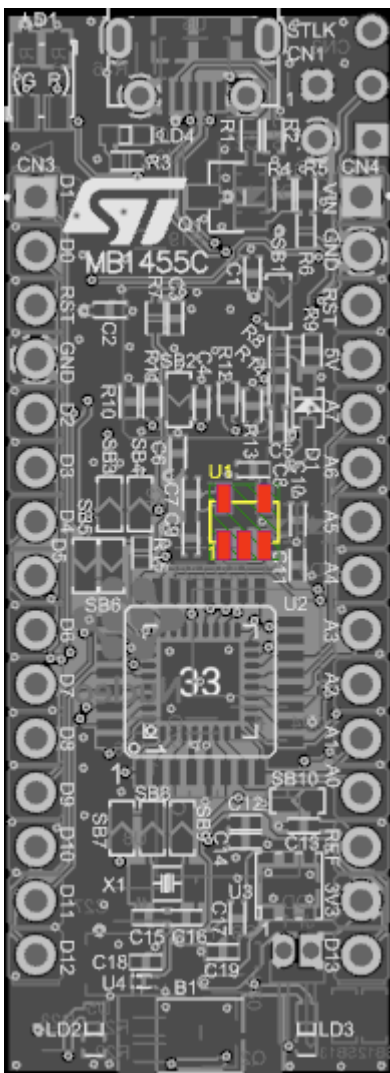
Ömer Emre Polat

1801022037



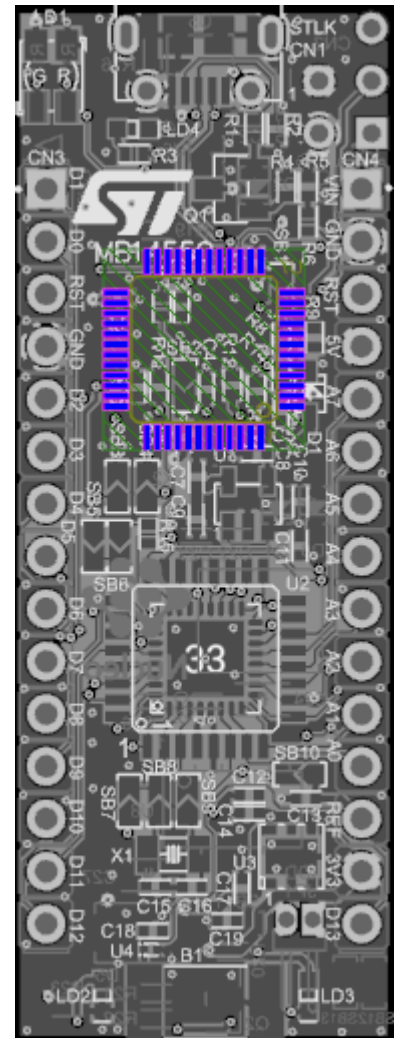
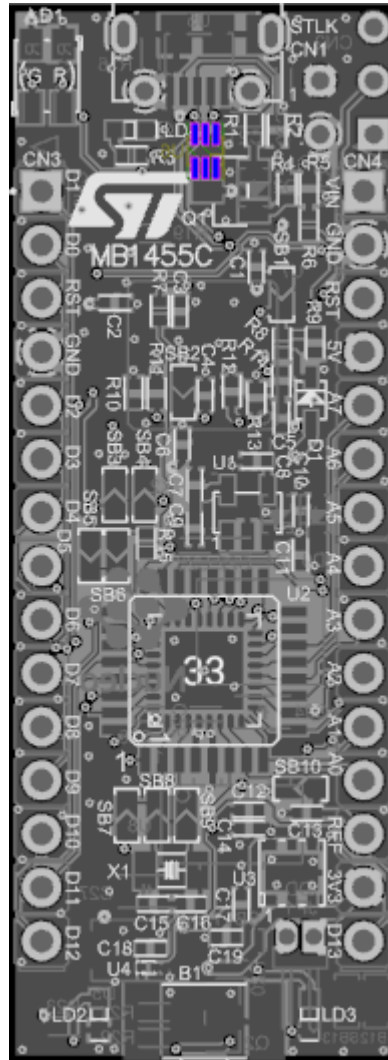
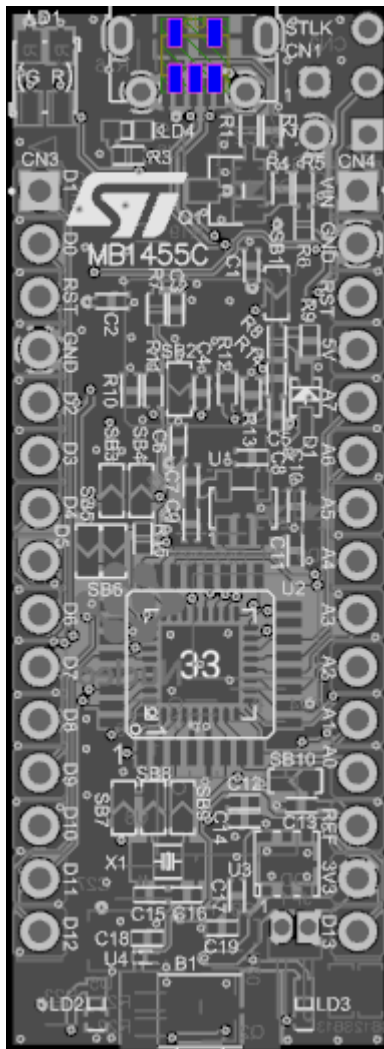
A. Problem 1

IC's are identified using an official circuit model of the STM32G031K8T6U board and a model preview software.



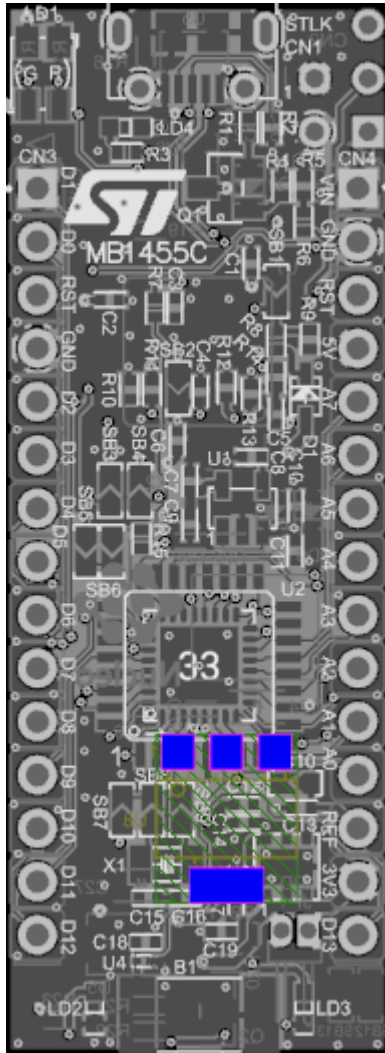
From left to right

- 1- LDK120M33R
200mA low quiescent current very low noise LDO
- 2- STM32G031K8T6U
Cortex M0+ microcontroller
- 3- LD39050PU33R
Low quiescent current and low noise voltage regulator



From left to right

- 1- STMPS2141STR
Enhanced single channel switch
- 2- USBLC6-2P6
USB OTG FS ESD protection
- 3- STM32F103CBT6
Arm based 32-bit MCU



From left to right

- 1- LD1117S50TR
5V REG LDO SOT223

B. Problem 2

For this problem we will have to look up for the value of GPIOA base address from the RM044.

The following table gives the boundary addresses of the peripherals.

Table 4. STM32G0x1 peripheral register boundary addresses

Bus	Boundary address	Size	Peripheral	Peripheral register map
-	0xE000 0000 - 0xE00F FFFF	1MB	Cortex [®] -M0+ internal peripherals	-
IOPORT	0x5000 1800 - 0x5FFF FFFF	~256 MB	Reserved	-
	0x5000 1400 - 0x5000 17FF	1 KB	GPIOF	Section 6.4.12 on page 211
	0x5000 1000 - 0x5000 13FF	1 KB	Reserved	-
	0x5000 0C00 - 0x5000 0FFF	1 KB	GPIOD	Section 6.4.12 on page 211
	0x5000 0800 - 0x5000 0BFF	1 KB	GPIOC	Section 6.4.12 on page 211
	0x5000 0400 - 0x5000 07FF	1 KB	GPIOB	Section 6.4.12 on page 211
	0x5000 0000 - 0x5000 03FF	1 KB	GPIOA	Section 6.4.12 on page 211

Now we need to find the ODR offset value from the RM044.

0x14	GPIOx_ODR (where x = A..D, F)	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	OD15	OD14	OD13	OD12	OD11	OD10	OD9	OD8	OD7	OD6	OD5	OD4	OD3	OD2	OD1	OD0
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

After finding the correct address of the value for peripherals we can write the code for the LED code.

```

Stack_Size      EQU      0x00000400

                AREA      STACK, NOINIT, READWRITE, ALIGN=3
Stack_Mem       SPACE    Stack_Size
__initial_sp


        THUMB


        AREA      RESET, DATA, READONLY
        EXPORT    __Vectors


__Vectors
        DCD      __initial_sp          ; Top of Stack
        DCD      Reset_Handler        ; Reset Handler
        DCD      NMI_Handler          ; NMI Handler
        DCD      HardFault_Handler    ; Hard Fault Handler


        AREA      |.text|, CODE, READONLY


; nmi handler
NMI_Handler     PROC
        EXPORT    NMI_Handler
        B        .
        ENDP

```

```

; hardfault handler
HardFault_Handler    PROC
    EXPORT  HardFault_Handler
    B .
    ENDP

; entry function
Reset_Handler        PROC
    EXPORT  Reset_Handler
    ; //////////////////////////////////

    LDR r0, =(0x50000000 + 0x14) ; GPIOA with ODR offset
    LDR r2, [r0] ; take the value from the r0 address
    LDR r1, =0x100 ; number with only the 8th bit as a 1
    ORRS r2, r2, r1 ; perform OR operation with the r1
    STR r2, [r0] ; write back the new value that has its 8th bit as 1

    ; //////////////////////////////////
    B .
    ENDP

END

```

Register	Value
Core	
R0	0x50000014
R1	0x00000100
R2	0x00000100
R3	0x00000000
R4	0x00000000
R5	0x00000000
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x20000400
R14 (LR)	0xFFFFFFFF
R15 (PC)	0x0800001E
xPSR	0x01000000
Banked	
System	
Internal	
Mode	Thread
Privilege	Privileged
Stack	MSP
States	11
Sec	0.00000092


```

47:      B .
->0x0800001E E7FE      B      0x0800001E
0x08000020 0014      DCW      0x0014
0x08000022 5000      DCW      0x5000
0x08000024 0100      DCW      0x0100
0x08000026 0000      DCW      0x0000
0x08000028 0000      DCW      0x0000

lab1_1.s
22 NMI_Handler    PROC
23     EXPORT NMI_Handler
24     B .
25     ENDP
26
27
28 ; hardfault handler
29 HardFault_Handler    PROC
30     EXPORT HardFault_Handler
31     B .
32     ENDP
33
34
35 ; entry function
36 Reset_Handler        PROC
37     EXPORT Reset_Handler
38     ; //////////////////////////////////
39
40     LDR r0, =(0x50000000 + 0x14) ; GPIOA with ODR offset
41     LDR r2, [r0] ; take the value from the r0 address
42     LDR r1, =0x100 ; number with only the 8th bit as a 1
43     ORRS r2, r2, r1 ; perform OR operation with the r1
44     STR r2, [r0] ; write back the new value that has its 8th bit as 1
45
46     ; //////////////////////////////////
47     B .
48     ENDP
49
50     END

```

As we can see, the register r2 contains the value of the GPIOA with ODR offset address. 0x0100 has its 8th bit as a 1.

C. Problem 3

We can use the same code above with a little modification to write addresses and ORR values to light up multiple bits.

```
Stack_Size      EQU      0x00000400

                AREA     STACK, NOINIT, READWRITE, ALIGN=3
Stack_Mem       SPACE    Stack_Size
__initial_sp


                THUMB


                AREA     RESET, DATA, READONLY
                EXPORT   __Vectors


__Vectors
    DCD      __initial_sp          ; Top of Stack
    DCD      Reset_Handler        ; Reset Handler
    DCD      NMI_Handler          ; NMI Handler
    DCD      HardFault_Handler    ; Hard Fault Handler


                AREA     |.text|, CODE, READONLY


; nmi handler
NMI_Handler     PROC
                EXPORT   NMI_Handler
                B .
                ENDP


; hardfault handler
HardFault_Handler  PROC
                EXPORT   HardFault_Handler
                B .
                ENDP


; entry function
Reset_Handler     PROC
                EXPORT   Reset_Handler
                ; ///////////

                LDR r0, =(0x50000000 + 0x14) ; GPIOA with ODR offset
                LDR r2, [r0] ; take the value from the r0 address
                LDR r1, =0x1800 ; number with only the 12th and 11th bit as a 1
                ORRS r2, r2, r1 ; perform OR operation with the r1
                STR r2, [r0] ; write back the new value that has its 12th and 11th bit as 1


                LDR r0, =(0x50000400 + 0x14) ; GPIOB with ODR offset
                LDR r2, [r0] ; take the value from the r0 address
                LDR r1, =0x30 ; number with only the 4th and 5th bit as a 1
```



```

ORRS r2, r2, r1 ; perform OR operation with the r1
STR r2, [r0] ; write back the new value that has its 4th and 5th bit as 1

; ////////////
B .
ENDP

END

```

As stated above we use two different GPIO addresses and ORR values to light up multiple bits at different address values. We can determine the ORR values by the high bit positions. For example in the GPIOA part we use a number that's 12th and 11th bits are 1 and all the other bits are zero. Performing ORR operation on the address value with this number ensures that the other on or off values wont be lost after rewriting the address to light up the led.

Register	Value
Core	
R0	0x50000414
R1	0x00000030
R2	0x00000030
R3	0x00000000
R4	0x00000000
R5	0x00000000
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x20000400
R14 (LR)	0xFFFFFFFF
R15 (PC)	0x08000028
xPSR	0x01000000
Banked	
System	
Internal	
Mode	Thread
Privilege	Privileged
Stack	MSP
States	18
Sec	0.00000150

Address	Label	Value	Comment
0x08000028	E7FE	B	0x08000028
0x0800002A	0000	DCW	0x0000
0x0800002C	0014	DCW	0x0014
0x0800002E	5000	DCW	0x5000
0x08000030	1800	DCW	0x1800


```

lab1_2.s
28 ; hardfault handler
29 HardFault_Handler PROC
30     EXPORT HardFault_Handler
31     B .
32     ENDP
33
34 ; entry function
35 Reset_Handler PROC
36     EXPORT Reset_Handler
37     ; ////////////
38
39
40     LDR r0, =(0x50000000 + 0x14) ; GPIOA with ODR offset
41     LDR r2, [r0] ; take the value from the r0 address
42     LDR r1, =0x1800 ; number with only the 12th and 11th bit as a 1
43     ORRS r2, r2, r1 ; perform OR operation with the r1
44     STR r2, [r0] ; write back the new value that has its 12th and 11th bit as 1
45
46     LDR r0, =(0x50000400 + 0x14) ; GPIOB with ODR offset
47     LDR r2, [r0] ; take the value from the r0 address
48     LDR r1, =0x30 ; number with only the 4th and 5th bit as a 1
49     ORRS r2, r2, r1 ; perform OR operation with the r1
50     STR r2, [r0] ; write back the new value that has its 4th and 5th bit as 1
51
52     ; ////////////
53     B .
54     ENDP
55
56     END

```

As expected after modifying the W/O addresses on the linker settings, we don't get any write or read violation errors and the program works as expected.

D. Problem 4

With this problem we have to implement a led delay loop that will turn the led on wait for around 1 seconds turn the led off wait for additional 1 seconds and jump back to the beginning.

```
Stack_Size      EQU      0x00000400

                AREA      STACK, NOINIT, READWRITE, ALIGN=3
Stack_Mem        SPACE    Stack_Size
__initial_sp


                THUMB


                AREA      RESET, DATA, READONLY
                EXPORT    __Vectors


__Vectors
    DCD      __initial_sp          ; Top of Stack
    DCD      Reset_Handler        ; Reset Handler
    DCD      NMI_Handler          ; NMI Handler
    DCD      HardFault_Handler    ; Hard Fault Handler


                AREA      |.text|, CODE, READONLY


; nmi handler
NMI_Handler      PROC
    EXPORT    NMI_Handler
    B .
    ENDP


; hardfault handler
HardFault_Handler  PROC
    EXPORT    HardFault_Handler
    B .
    ENDP


; entry function
Reset_Handler     PROC
    EXPORT    Reset_Handler
    ; ////////////

LED_ON ; branch to turn the led on

    LDR r3, =(0x50000000 + 0x14) ; GPIOA with ODR offset
    LDR r2, [r3]
    LDR r1, =0x100 ; number with only the 8th bit as 1
    ORRS r2, r2, r1 ; or operation to set the given bit high
    STR r2, [r3]

    LDR r4,=0x7A1200 ; cycles for 1 sec total delay
```


ON_DELAY ; on time delay

```
CMP r4, #0x0
BEQ LED_OFF
SUBS r4, r4, #1
B ON_DELAY
```

LED_OFF ; branch to turn off the led

```
LDR r3, =(0x50000000 + 0x14) ; GPIOA with ODR offset
LDR r2, [r3]
LDR r1, =0xFFFFFEFF ; number with only the 8th bit as 0
ANDS r2, r2, r1 ; and operation to set the given bit low
STR r2, [r3]
```

```
LDR r4,=0x249F60
```

OFF_DELAY ; off time delay

```
CMP r4, #0x0
BEQ LED_ON
SUBS r4, r4, #1
B OFF_DELAY
```

```
; ///////////
B .
ENDP
```

```
END
```

With the code written we can test it if it loops continuously.

Register	Value
Core	
R0	0x00000000
R1	0x00000100
R2	0x00000100
R3	0x50000014
R4	0x00000000
R5	0x00000000
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x20000400
R14 (LR)	0xFFFFFFFF
R15 (PC)	0x0800001E
xPSR	0x21000000
Banked	
System	
Internal	
Mode	Thread
Privilege	Privileged
Stack	MSP
States	104001025
Sec	8.66675208


```

62:      LDR r1, =0xFFFFFFFF ; number with only the 8th bit as 0
0x0800002C 4907      LDR      r1, [pc, #28] ; @0x0800004C
63:      ANDS r2, r2, r1 ; and operation to set the given bit low
0x0800002E 400A      ANDS      r2, r2, r1
64:      STR r2, [r3]
65:
< 0x08000030 6013      STR      r2, [r2, #0x001]

lab1_3.s
47
48      LDR r4, =0x7A1200 ; cycles for 1 sec total delay
49
50
51      ON_DELAY ; on time delay
52
53      CMP r4, #0x0
54      BEQ LED_OFF
55      SUBS r4, r4, #1
56      B ON_DELAY
57
58      LED_OFF ; branch to turn off the led
59
60      LDR r3, =(0x50000000 + 0x14) ; GPIOA with ODR offset
61      LDR r2, [r3]
62      LDR r1, =0xFFFFFFFF ; number with only the 8th bit as 0
63      ANDS r2, r2, r1 ; and operation to set the given bit low
64      STR r2, [r3]
65
66      LDR r4, =0x249F60
67
68      OFF_DELAY ; off time delay
69
70      CMP r4, #0x0
71      BEQ LED_ON
72      SUBS r4, r4, #1
73      B OFF_DELAY
74
75      ; ///////////
76      B .
77      ENDP
78
79      END

```

As we can see after doing multiple loops the code still works with the proper branch jumps.

References

[1]. RM0444 https://www.st.com/resource/en/reference_manual/dm00371828-stm32g0x1-advanced-armbased-32bit-mcus-stmicroelectronics.pdf