Lab8
Digital system lab

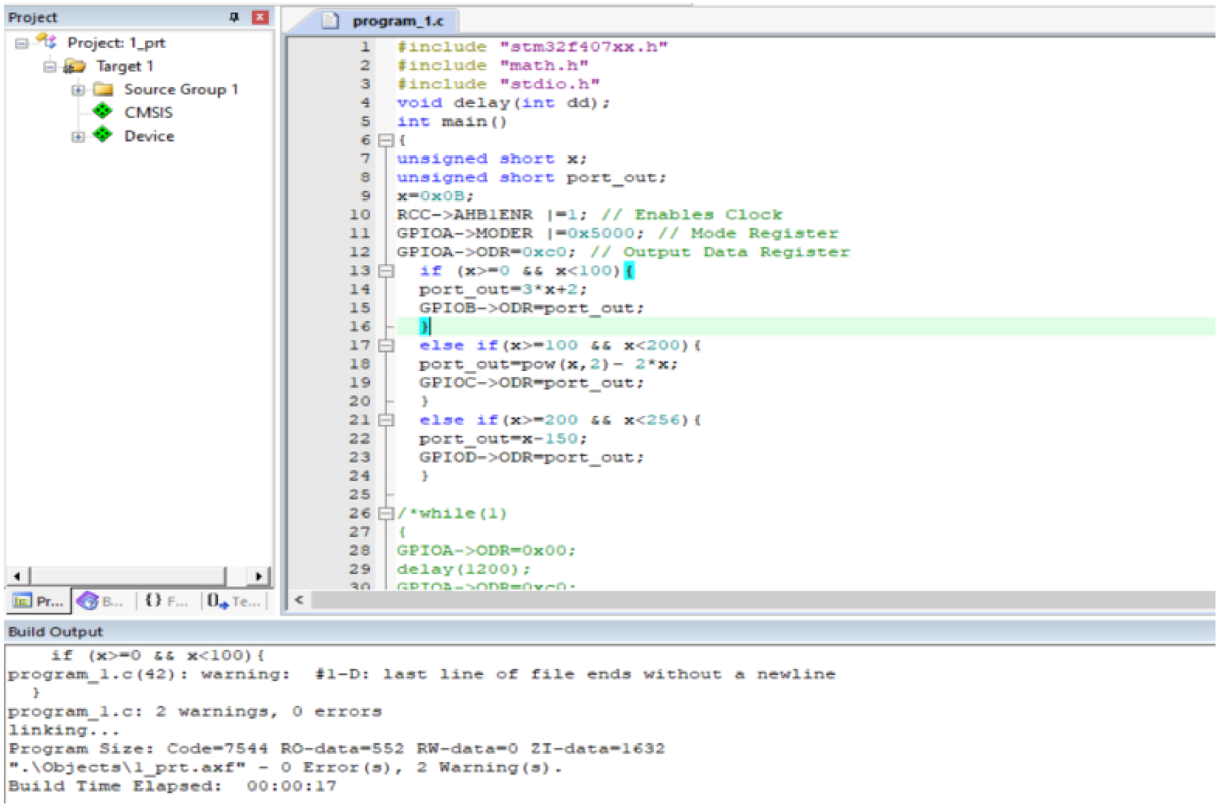
Ankit(B19CSE010)

Part1: Using GPIO

Task1: An unsigned 8-bit value is sensed/inputted at one of the MCU's ports and saved in a variable called x. Now, depending on the numerical value of x, it should be conditioned first before being transferred to ports as follows:

If $0 \le x < 100$, Send 3x + 2 to port-B.
If $100 \le x < 200$, Send x 2 − 2x to port-C.
If $200 \le x < 256$, Send x − 150 to port-D.

Solution:

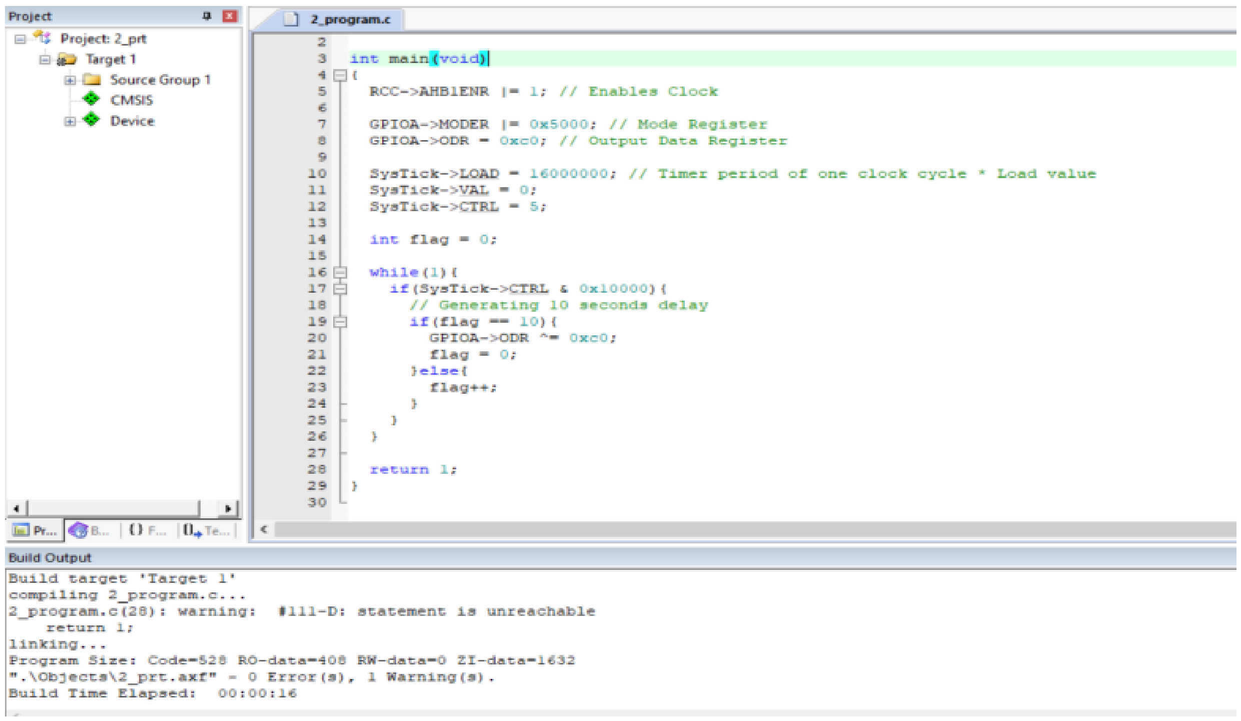


```c
#include "stm32f407xx.h"
#include "math.h"
#include "stdio.h"
void delay(int dd);
int main()
{
unsigned short x;
unsigned short port_out;
x=0x0B;
RCC->AHB1ENR |=1; // Enables Clock
GPIOA->MODER |=0x5000; // Mode Register
GPIOA->ODR=0xc0; // Output Data Register
  if (x>=0 && x<100){
  port_out=3*x+2;
  GPIOB->ODR=port_out;
  }
  else if(x>=100 && x<200){
  port_out=pow(x,2)- 2*x;
  GPIOC->ODR=port_out;
  }
  else if(x>=200 && x<256){
  port_out=x-150;
  GPIOD->ODR=port_out;
  }

/*while(1)
{
GPIOA->ODR=0x00;
delay(1200);
GPIOA->ODR=0xc0;
```

```
    if (x>=0 && x<100){
program_1.c(42): warning:  #1-D: last line of file ends without a newline
  }
program_1.c: 2 warnings, 0 errors
linking...
Program Size: Code=7544 RO-data=552 RW-data=0 ZI-data=1632
".\Objects\1_prt.axf" - 0 Error(s), 2 Warning(s).
Build Time Elapsed:  00:00:17
```

Explanation: As stated in the question, we declare x to be an unsigned integer, then use the if/else condition to apply certain conditions based on the value of x, and then send a specific output through each condition. This output is constructed by manipulating the value of x, so it should also be an unsigned integer. Then, for each condition, the appropriate output is transmitted to the port in accordance with the condition.

Part2: : Using SysTick Timer
Task2 : Write a program to provide a delay of 10 seconds using the system timer. Assume input clock frequency as 16MHz. (Hint: use a timer in a loop).

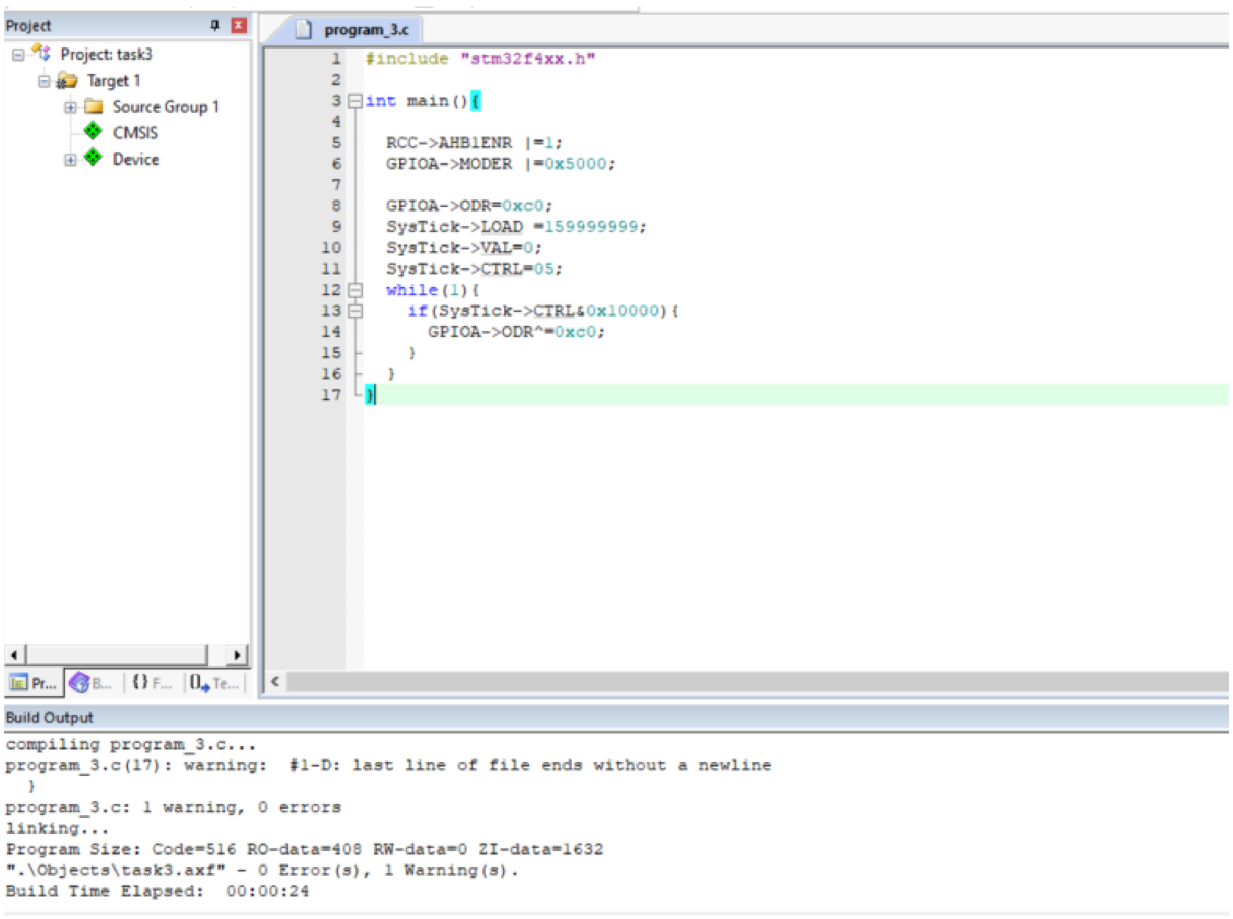


```
 2
 3  int main(void)
 4  {
 5    RCC->AHB1ENR |= 1; // Enables Clock
 6
 7    GPIOA->MODER |= 0x5000; // Mode Register
 8    GPIOA->ODR = 0xc0; // Output Data Register
 9
10    SysTick->LOAD = 16000000; // Timer period of one clock cycle * Load value
11    SysTick->VAL = 0;
12    SysTick->CTRL = 5;
13
14    int flag = 0;
15
16    while(1){
17      if(SysTick->CTRL & 0x10000){
18        // Generating 10 seconds delay
19        if(flag == 10){
20          GPIOA->ODR ^= 0xc0;
21          flag = 0;
22        }else{
23          flag++;
24        }
25      }
26    }
27
28    return 1;
29  }
30
```

```
Build target 'Target 1'
compiling 2_program.c...
2_program.c(28): warning:  #111-D: statement is unreachable
    return 1;
linking...
Program Size: Code=528 RO-data=408 RW-data=0 ZI-data=1632
".\Objects\2_prt.axf" - 0 Error(s), 1 Warning(s).
Build Time Elapsed:  00:00:16
```

Explanation: The frequency is set to 16 MHz, and we know that the SysTick Load register can only hold up to 24 bits, so we only save 16x10. Because load value 6 equates to a one-second delay, we loop roughly 10 times to achieve a ten-second delay.

Task 3: Write a program to toggle LED using System Timer (Sys_Tick). Assume that You need to toggle LEDs at PA6 and PA7.

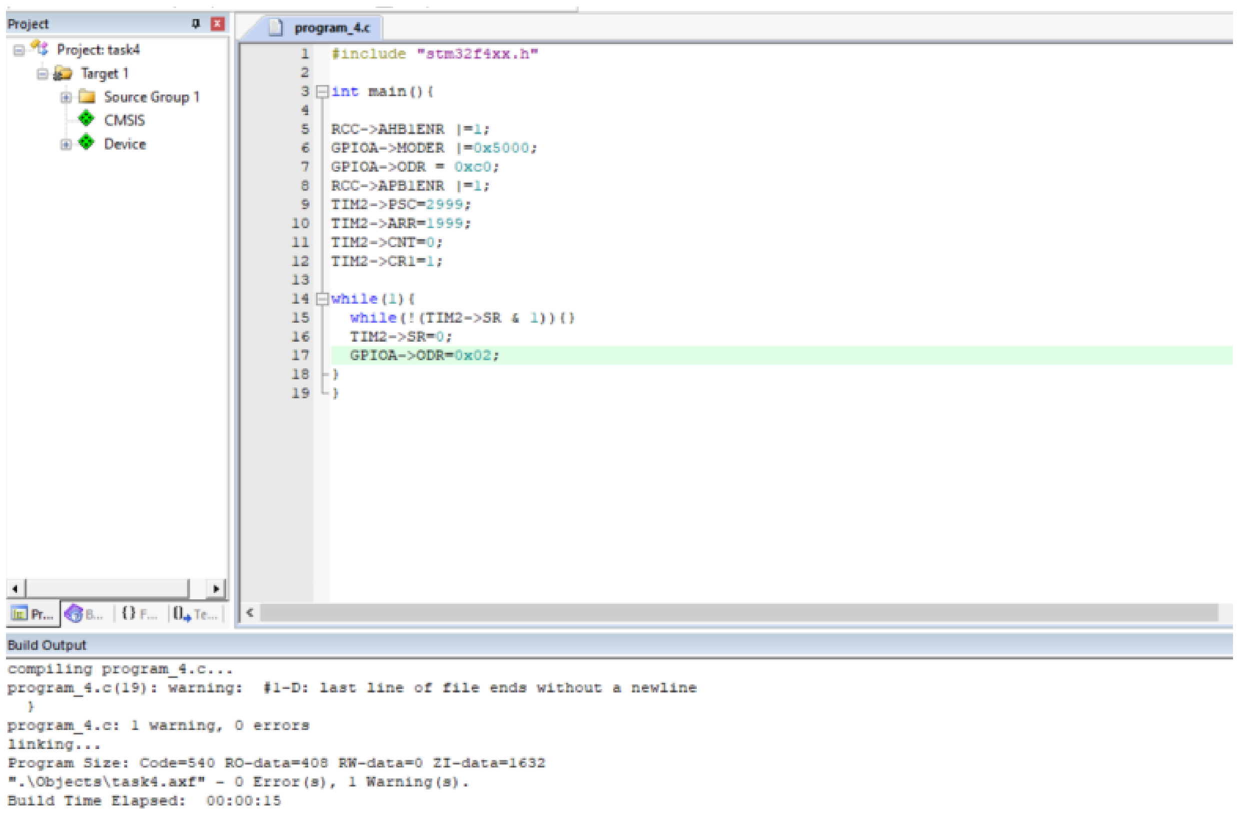


```c
#include "stm32f4xx.h"

int main(){

  RCC->AHB1ENR |=1;
  GPIOA->MODER |=0x5000;

  GPIOA->ODR=0xc0;
  SysTick->LOAD =159999999;
  SysTick->VAL=0;
  SysTick->CTRL=05;
  while(1){
    if(SysTick->CTRL&0x10000){
      GPIOA->ODR^=0xc0;
    }
  }
}
```

```
compiling program_3.c...
program_3.c(17): warning:  #1-D: last line of file ends without a newline
  }
program_3.c: 1 warning, 0 errors
linking...
Program Size: Code=516 RO-data=408 RW-data=0 ZI-data=1632
".\Objects\task3.axf" - 0 Error(s), 1 Warning(s).
Build Time Elapsed:  00:00:24
```

Explanation: Given that we must toggle the LED at PA6 and PA7, we must use port A and set the GPIO Moder and ODR values to set pin 6 and pin 7 to high.
To toggle the LED, we utilise Binary XOR, and the load value in SysTick is adjusted to match the delay we require. =delay*system clock is the SysTick Reload Value.

Part 3: Using General Timer:
Task 4: Generate a square wave of 05 KHz frequency using Timer 2. Obtain this square wave on PA1 (GPIO A, pin 1).



```c
#include "stm32f4xx.h"

int main(){

RCC->AHB1ENR |=1;
GPIOA->MODER |=0x5000;
GPIOA->ODR = 0xc0;
RCC->APB1ENR |=1;
TIM2->PSC=2999;
TIM2->ARR=1999;
TIM2->CNT=0;
TIM2->CR1=1;

while(1){
  while(!(TIM2->SR & 1)){}
  TIM2->SR=0;
  GPIOA->ODR=0x02;
}
}
```

```
compiling program_4.c...
program_4.c(19): warning:  #1-D: last line of file ends without a newline
  }
program_4.c: 1 warning, 0 errors
linking...
Program Size: Code=540 RO-data=408 RW-data=0 ZI-data=1632
".\Objects\task4.axf" - 0 Error(s), 1 Warning(s).
Build Time Elapsed:  00:00:15
```

Explanation:
To generate the square wave we need prescaler and Reload value, for this let us consider the clock frequency to be 30,000 MHz,
Now, timer frequency: Timer frequency=clock frequency/(prescaler+1)*(ARR+1)

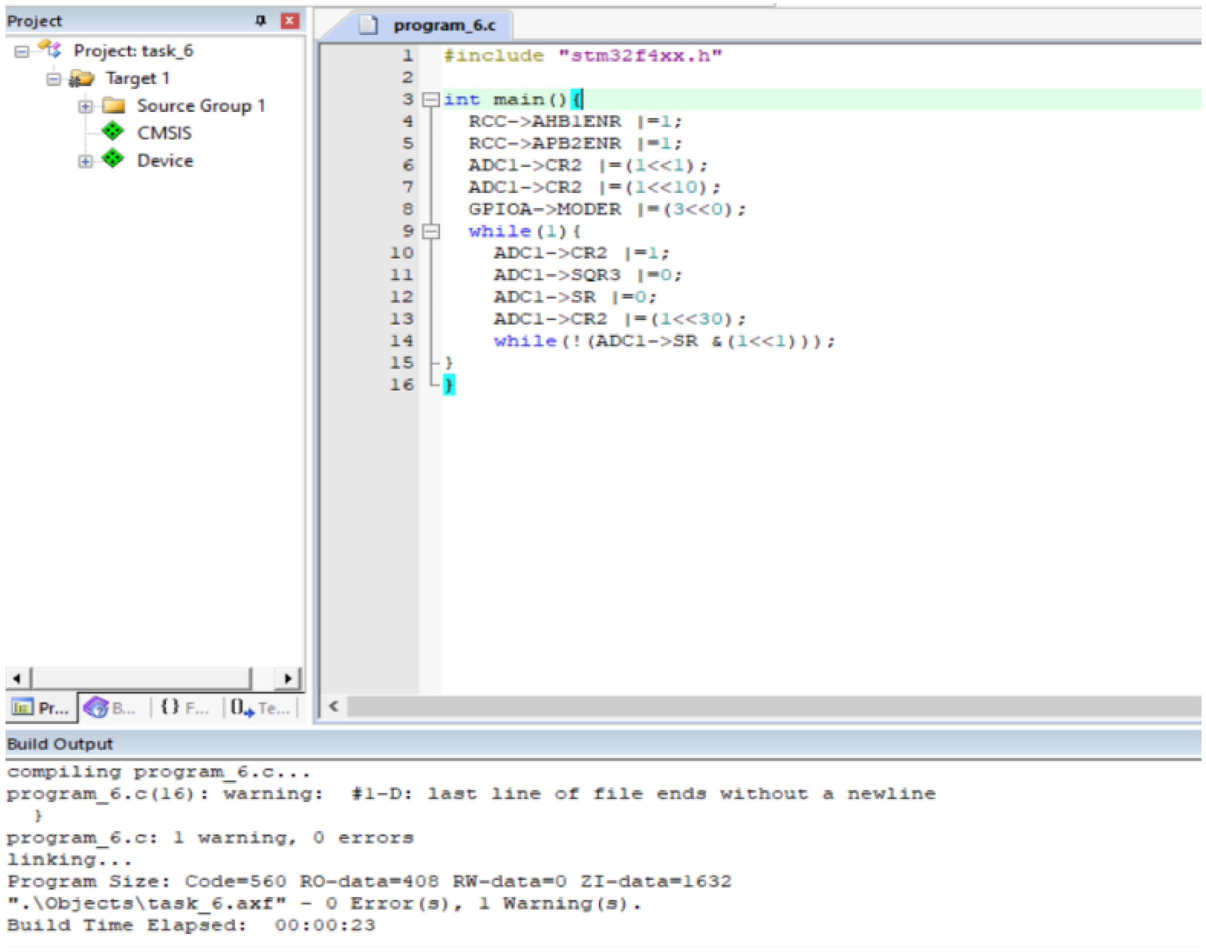
So, 5kHz=30,000 MHz/(prescaler+1)*(ARR+1)

Then, prescaler=2999, ARR=1999

Task 5: Generate a triangular wave of 50 KHz frequency using Timer 5. Obtain this triangular wave on PA7 (GPIO A, pin 7).

This triangle wave cannot be obtained with GPIO. We can, however, get square waves. This is because we need to alter the value from 0 to 1 for a triangular wave, but we can set the value to either 0 or 1 in the code.
To generate the triangle wave, we'll need different hardware. It could be possible to make it with an integrator machine that integrates the rectangular waveform.

Task 6: Configure ADC1 Channel IN_0 to read data at its respective GPIO pin



```
1  #include "stm32f4xx.h"
2
3  int main(){
4    RCC->AHB1ENR |=1;
5    RCC->APB2ENR |=1;
6    ADC1->CR2 |=(1<<1);
7    ADC1->CR2 |=(1<<10);
8    GPIOA->MODER |=(3<<0);
9    while(1){
10     ADC1->CR2 |=1;
11     ADC1->SQR3 |=0;
12     ADC1->SR |=0;
13     ADC1->CR2 |=(1<<30);
14     while(!(ADC1->SR &(1<<1)));
15   }
16 }
```

```
compiling program_6.c...
program_6.c(16): warning:  #1-D: last line of file ends without a newline
  }
program_6.c: 1 warning, 0 errors
linking...
Program Size: Code=560 RO-data=408 RW-data=0 ZI-data=1632
".\Objects\task_6.axf" - 0 Error(s), 1 Warning(s).
Build Time Elapsed:  00:00:23
```

Explanation:
At first, we enabled the clock for GPIO and the clock for ADC1 was enabled .

Then, using line 6 of the code, we activate the continuous conversation mode, in which the ADC analogue input is continuously written to the ADC result data register, and finally, using line 7 of the code, we exit the conversation mode.

Then, for GPIO port-A, the MODER value is configured to use the analogue mode for the PA-0 pin.
The ADC-1 module is now activated inside the endless loop, and the normal sequence register and the status register are both set to 0.
This conversion is restarted with the loop's start, and the condition inside the loop is checked once more until the conversion is complete.