Лабораторна робота №4. Передача даних інтерфейсом I2C

Дана робота виконується на основі мікроконтролера STM32F407VG, що входить до складу відлагоджувальної плати Discovery 1.

1. **Підготовка**

1. Для вибору даної плати після створення проекту у вікні «Target selection» відкрийте вкладку «Board selector».
2. У списку «MCU/MPU Series» оберіть опцію «STM32F4” (рис.1).
3. Зі списку плат, що відкриється в нижній частині вікна, оберіть плату «STM32F407VG-DISC1”
4. Після виконання наведених вище пунктів та визначення імені та попередніх налаштувань проекту, у спливаючому вікні обрати варіант «No”. (рис.2)

2. Ініціалізація периферії

1. Підключити інтерфейс I2C1. Параметри сигналу залишити за замовченням.
2. Підключити інтерфейс USART2. Параметри сигналу:

   Швидкість: 9600 бод

   Довжина повідомлення – 8 біт;

   Біт парності – відсутній;

   Стоп-біт – 1.

   Завдання:

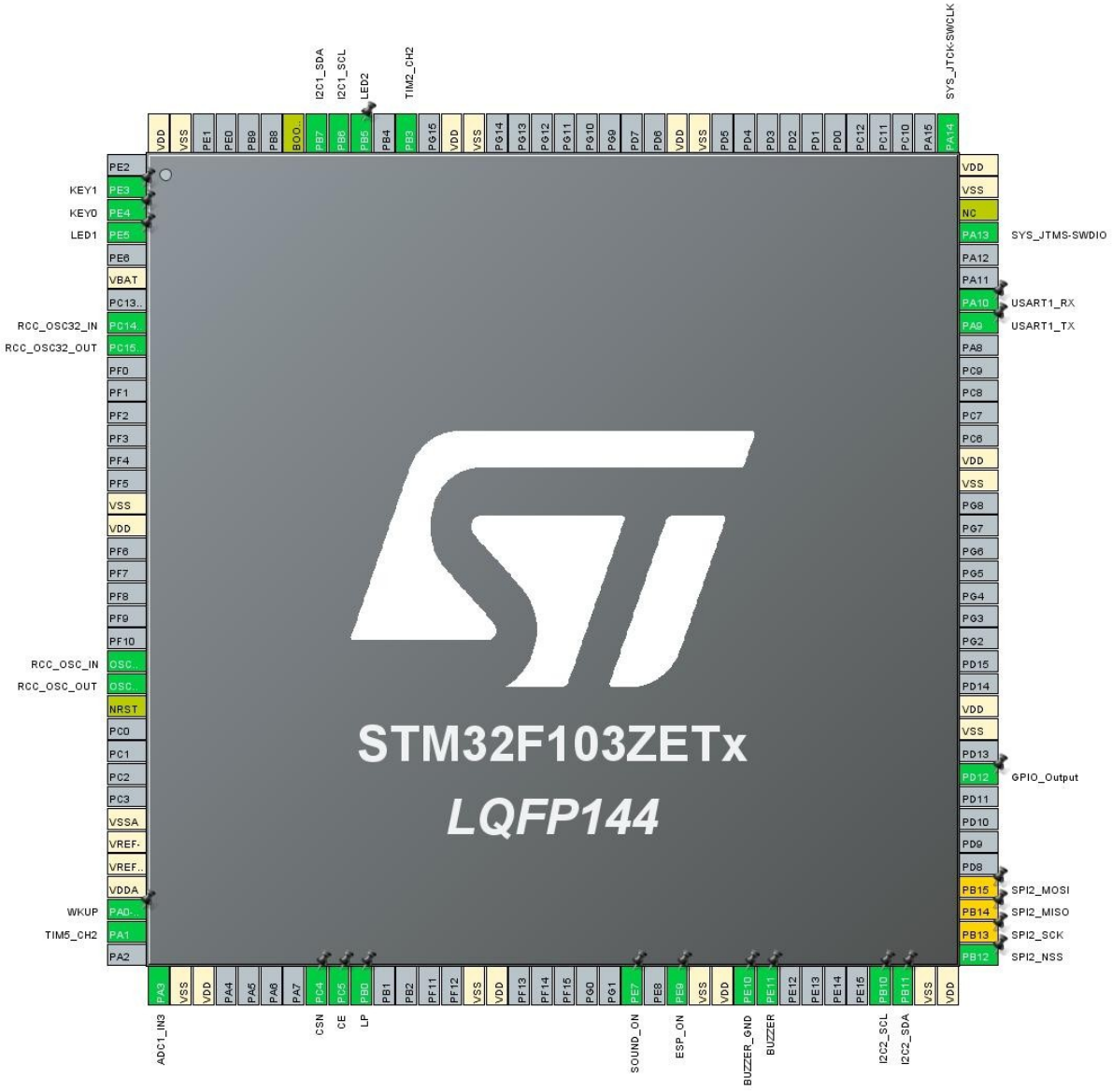
   Створити програмний продукт, що буде запускати зовнішній модуль годинника реального часу DS 1307 на відлік секунд від 0 до величини рівної номеру Вашого варіанту помноженого на 8. Після цього данні обнульовуються, та відлік починається спочатку з нульового значення. Процес відліку передається за допомогою інтерфейсу USART у термінал.

   Алгоритм роботи з зовнішнім модулем годинника реального часу DS 1307 наступний:

- Спочатку визначається адреса пристрою з яким буде встановлено зв’язок за допомогою інтерфейсу I2C. Доречно перевірити дану адресу вбудованою функцією бібліотеки HAL.
- Потім в одному повідомленні передається адреса реєстру в який буде відбуватися запис та повідомлення, що має бути туди записано. Необхідно переслати початкове значення часового інтервалу.
- Зчитувати стан секундного регістру та передавати його інтерфейсом USART. За досягнення верхньої межі розрахунку визначеної завданням обнулити значення секундного регістру модуля та перезапустити рахунок.

Інше завдання:

1. Створити проект у STM32CubeMX+STM32CubeIDE активувавши з периферії світлодіоди, I2C1 та I2C2. Зверніть увагу, щоб виводи обох I2C були такими ж, як і на скріншоті.



2. Використавши код, який дає викладач запустити LCD-дисплей та вивести на нього довільний рядок. Перевірити, вивантаживши на плату.

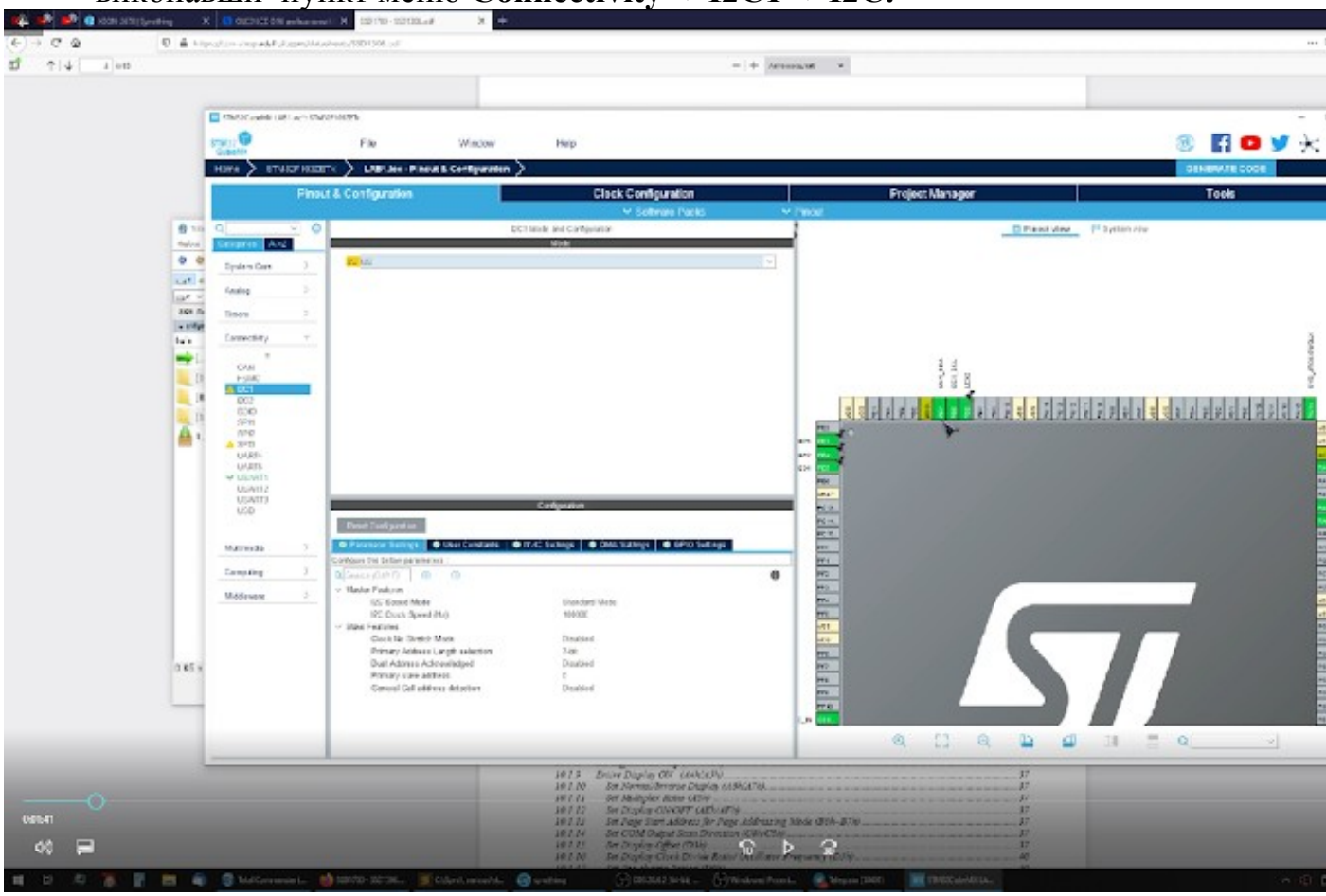
3. Запустити магнітометр та вивести його покази на дисплей. Перевірити, вивантаживши на плату. Рядок, на який виводити, має відповідати R=N %6+1, де N – номер варіанту.

4. Визначити мінімальну та максимальну частоти роботи I2C як для магнітометра, так і для дисплею.

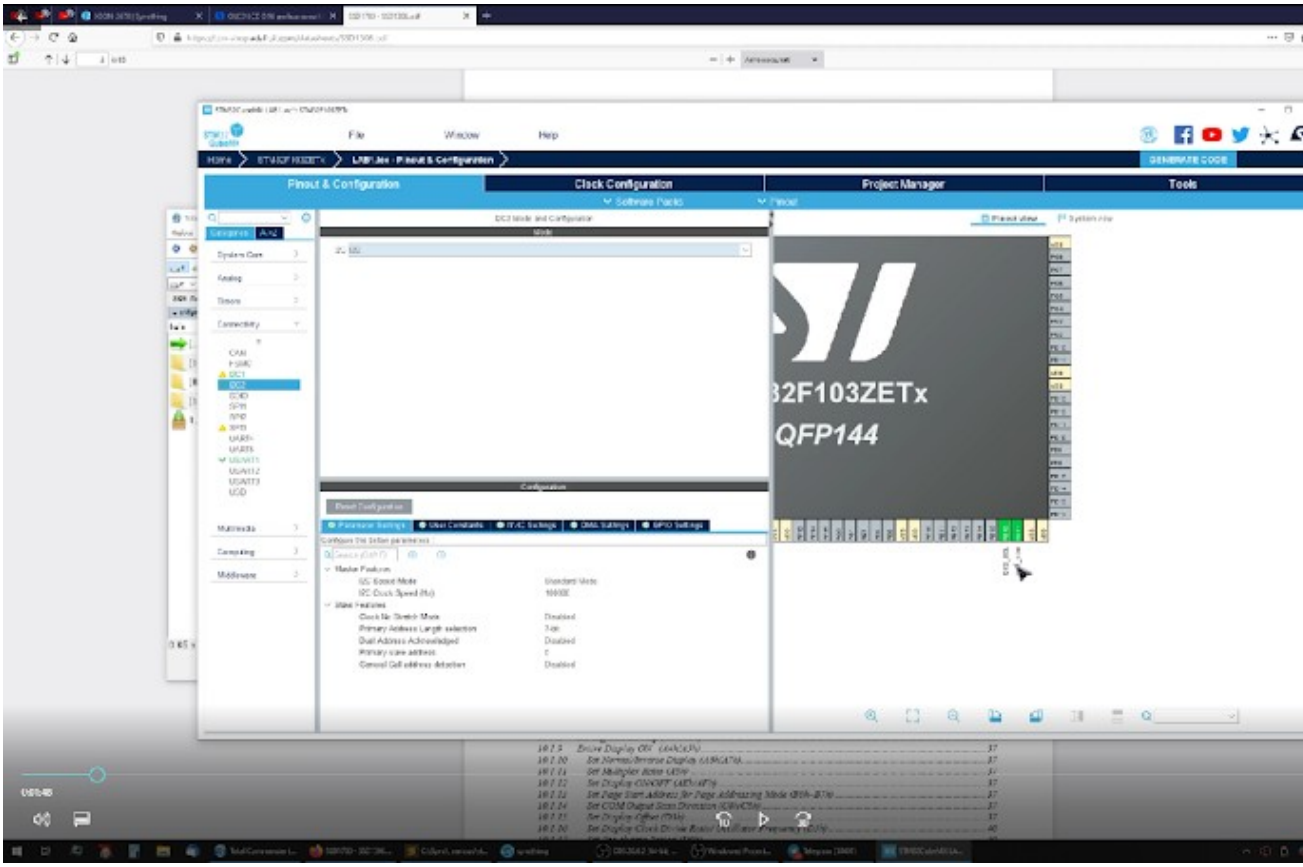
5. Змінити чутливість магнетометру відповідно до номеру бригади, вивести його покази на дисплей. Перерахувати покази, довести, що вони відповідають дійсності. Перевірити, вивантаживши на плату.

Порядок виконання роботи

1. Створити проект у STM32CubeMX+STM32CubeIDE активувавши з периферії шини I2C для екрану і магнітометра. На даній платформі існує дві таких шини. За основу можна використовувати старі проекти лабораторних робіт. Для цього необхідно дозволити його тактування виконавши пункт меню **Connectivity → I2C1 → I2C.**



Та вмикаємо другу шину у пункті меню **Connectivity → I2C2 → I2C.**



2. Підключаємо LCD-дисплей та вивести на нього довільний рядок. Для цього використовуємо код с файлу, що дає викладач(l4_Code.txt). Перевірити, вивантаживши на плату.
   1. Ініціалізація дисплея.

   Вказуємо адресу LCD – дисплею за допомогою директиви

```
#define OLED_ADDRESS 0x78
```

Об’являємо функцію WriteCmd():

```
void WriteCmd(unsigned char I2C_Command)
{
    I2C_WriteByte(0x00, I2C_Command);
}
```

WriteCmd(0x00) – це запис одного байту на шині I2C за внутрішньою адресою пристрою 0x00.

Опишемо функцію I2C_WriteByte:

```
void I2C_WriteByte(uint8_t addr,uint8_t data)
{

    HAL_I2C_Mem_Write(&hi2c1, OLED_ADDRESS, addr, 1, &data, 1, 1000);

}
```

Ця функція буде працювати через функцію HAL та I2C_Mem_Write. Тобто обмін даними з дисплеєм буде проходити так само, як обмін даними загалом з будь-якою пам’яттю I2C.

Функція HAL_I2C_Mem_Write приймає наступні аргументи:
&hi2c1 - посилання на шину I2C;
OLED_ADDRESS – адреса дисплея на шині I2C;
addr – внутрішня адреса
1 – те, що адреса займає 1 байт. Відповідно адреси будуть від 0 до 255;
&data – вказівник на данні;
1 – те, що в нас передається 1 байт даних;
1000 – тайм-аут виконання цієї задачі.

Увесь код для ініціалізації дисплея матиме наступний вигляд:

```
#define OLED_ADDRESS    0x78

void WriteCmd(unsigned char I2C_Command)//Ð´ÃüÁî
{
    I2C_WriteByte(0x00, I2C_Command);
}


void I2C_WriteByte(uint8_t addr,uint8_t data)
{

    HAL_I2C_Mem_Write(&hi2c1, OLED_ADDRESS, addr, 1, &data, 1, 1000);

}

void OLED_Init(void)
{
    HAL_Delay(100); //ÕâÀïµÄÑÓ³Ù²»¿É¼¤

    WriteCmd(0xAE); //display off
    WriteCmd(0x20); //Set Memory Addressing Mode
    WriteCmd(0x10); //00,Horizontal Addressing Mode;01,Vertical Addressing Mode;10,Page Addressing Mode (RESET);11,Invalid
    WriteCmd(0xb0);*//Set Page Start Address for Page Addressing Mode,0-7
    WriteCmd(0xc8); //Set COM Output Scan Direction
    WriteCmd(0x00); //---set low column address
    WriteCmd(0x10); //---set high column address
    WriteCmd(0x40); //--set start line address
    WriteCmd(0x81); //--set contrast control register
    WriteCmd(0xff); //¼¤¶¯µçÁ÷Öµ 0x00-0xff
    WriteCmd(0xa1); //--set segment re-map 0 to 127
    WriteCmd(0xa6); //--set normal display
    WriteCmd(0xa8); //--set multiplex ratio(1 to 64)
    WriteCmd(0x3F); //
    WriteCmd(0xa4); //0xa4,Output follows RAM content;0xa5,Output ignores RAM content
    WriteCmd(0xd3); //-set display offset
    WriteCmd(0x00); //-not offset
    WriteCmd(0xd5); //--set display clock divide ratio/oscillator frequency
    WriteCmd(0xf0); //--set divide ratio
    WriteCmd(0xd9); //--set pre-charge period
    WriteCmd(0x22); //
    WriteCmd(0xda); //--set com pins hardware configuration
    WriteCmd(0x12);
    WriteCmd(0xdb); //--set vcomh
    WriteCmd(0x20); //0x20,0.77xVcc
    WriteCmd(0x8d); //--set DC-DC enable
    WriteCmd(0x14); //
    WriteCmd(0xaf); //--turn on oled panel
}
/* USER CODE END 0 */
```

З детальним описом команд WriteCmd() можна ознайомитись в даташиті.

2. Вивід інформації на дисплей.

Для виводу інформації на екран використовується функція запису WriteDat().

```
void WriteDat(unsigned char I2C_Data)//Ð ÊÝ¼Ý
{
    I2C_WriteByte(0x40, I2C_Data);
}


void OLED_Fill(unsigned char fill_Data)//È«ÆÁÌî³ä
{
    unsigned char m,n;
    for(m=0;m<8;m++)
    {
        WriteCmd(0xb0+m);       //page0-page1
        WriteCmd(0x00);     //low column start address
        WriteCmd(0x10);     //high column start address
        for(n=0;n<128;n++)
            {
                WriteDat(fill_Data);
            }
    }
}
```

Вмикаємо екран и засвічуємо його повністю засвічуємо. Для цього ініціалізуємо екран за допомогою функції OLED_Init та вмикаємо всі пікселі за допомогою функції OLED_Fill(0xFF):

```
  MX_GPIO_Init();
  MX_TIM5_Init();
  MX_USART1_UART_Init();
  MX_I2C1_Init();
  MX_I2C2_Init();
  /* USER CODE BEGIN 2 */
  OLED_Init();
  OLED_Fill(0xFF);
  /* USER CODE END 2 */
```

Для того, щоб вивести текст на дисплей, потрібно створити масив шрифтів:

```
126 void OLED_SetPos(unsigned char x, unsigned char y) //встановлення позиції курсору на екрані
127 {
128     WriteCmd(0xb0+y);
129     WriteCmd(((x&0xf0)>>4)|0x10);
130     WriteCmd((x&0x0f)|0x01);
131 }
132
133 void OLED_CLS(void)//функція стирання
134 {
135     OLED_Fill(0x00);
136 }
170                 for(i=0;i<8;i++)
171                     WriteDat(F8X16[c*16+i]);
172                 OLED_SetPos(x,y+1);
173                 for(i=0;i<8;i++)
174                     WriteDat(F8X16[c*16+i+8]);
175                 x += 8;
176                 j++;
177             }
178         }break;
179     }
180 }
181
137 void OLED_ShowStr(unsigned char x, unsigned char y, unsigned char ch[], unsigned char TextSize)//функція виводу рядка
138 {
139     unsigned char c = 0,i = 0,j = 0;
140     switch(TextSize)
141     {
142         case 1:
143         {
144             while(ch[j] != '\0')
145             {
146                 c = ch[j] - 32;
147                 if(x > 126)
148                 {
149                     x = 0;
150                     y++;
151                 }
152                 OLED_SetPos(x,y);
153                 for(i=0;i<6;i++)
154                     WriteDat(F6x8[c][i]);
155                 x += 6;
156                 j++;
157             }
158         }break;
159         case 2:
160         {
161             while(ch[j] != '\0')
162             {
163                 c = ch[j] - 32;
164                 if(x > 120)
165                 {
166                     x = 0;
167                     y++;
168                 }
169                 OLED_SetPos(x,y);
```

Функція виводу рядка void OLED_ShowStr() буде приймати такі аргументи:
unsigned char x – встановлення позиції;
unsigned char y – встановлення позиції;
unsigned char ch[] – масив;
unsigned char TextSize – розмір тексту.
Далі з файлу, що дає l4_Code.txt вставляємо в код два масиви: F6x8[][6] та F8X16[].

Виводимо текст «Hello!» на екран, наприклад, на 4 рядок, з розміром тексту 2:

```
423   MX_GPIO_Init();
424   MX_TIM5_Init();
425   MX_USART1_UART_Init();
426   MX_I2C1_Init();
427   MX_I2C2_Init();
428   /* USER CODE BEGIN 2 */
429   OLED_Init();
430   OLED_Fill(0x0);
431   OLED_ShowStr(1, 4, "Hello!", 2);
432   /* USER CODE END 2 */
```

3. Запустити магнітометр та вивести його покази на дисплей. Перевірити, вивантаживши на плату. Рядок, на який виводити, має відповідати R=N %6+1, де N – номер варіанту.

```
#define QMC_5883_addr (0x1A)// адреса
  char H_init[]={0x01};// масив в який будемо записувати значення
  HAL_I2C_Mem_Write(&hi2c2,QMC_5883_addr,0x09,1,H_init,1,500);//команда ініціалізації
  H_init[0]=0x01;
  HAL_I2C_Mem_Write(&hi2c2,QMC_5883_addr,0x0B,1,H_init,1,500);//команда ініціалізації

int8_t M[6];// об'являємо массив для даних з датчика, розміром 6 байт
char c[50];//рядок в який ми будемо вписувати значення, перед тим як виводити на екран
int16_t MX, MY, MZ;
int counter=0;//змінна для рахування кількості циклів
  /* USER CODE END 2 */

  /* Infinite loop */
  /* USER CODE BEGIN WHILE */
  while (1)
  {
HAL_I2C_Mem_Read(&hi2c2,QMC_5883_addr,0,1,M,6,500);// змінні в які величини з магнітометру будуть записуватись в правильному порядку
MX=(M[1]<<8)|M[0];
MY=(M[3]<<8)|M[2];
MZ=(M[5]<<8)|M[4];
sprintf(c,"X: %d,Y: %d,Z:%d     ",MX,MY,MZ);//виводимо рядок
OLED_ShowStr(1, 4, c, 1);
sprintf(c,"cycles:%d     ",counter);//виводимо рахування циклу
OLED_ShowStr(1, 6, c, 1);
counter++;
HAL_Delay(500);
```

4. Визначити мінімальну та максимальну частоти роботи I2C як для магнітометра, так і для дисплею.

   Максимальну та мінімальну частоту роботи I2C визначається експериментально при зміні значення частоти шини I2C: **Connectivity → I2C1 → Master Features → I2C Clock Speed**. Рекомендується змінювати змінну шляхом множення\ділення на 2.

5. Змінити чутливість магнетометру відповідно до номеру варіанту, вивести його покази на дисплей. Перерахувати покази, довести, що вони відповідають дійсності. Перевірити, вивантаживши на плату.

   Чутливість змінюється через зміну регістра для магнітометру:

```
#define QMC_5883_addr (0x1A)// адреса
  char H_init[]={0x01};// масив в який будемо записувати значення
  HAL_I2C_Mem_Write(&hi2c2,QMC_5883_addr,0x09,1,H_init,1,500);//команда ініціалізації
  H_init[0]=0x01;
  HAL_I2C_Mem_Write(&hi2c2,QMC_5883_addr,0x0B,1,H_init,1,500);//команда ініціалізації
```

Необхідний регістр, можна взяти з даташиту до магнітометру, який використовується в даній лабораторній роботі:

Table 18. Control Register 1

Addr	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
09H	OSR[1:0]		RNG[1:0]		ODR[1:0]		MODE[1:0]	

Reg.	**Definition**	**00**	**01**	**10**	**11**
Mode	Mode Control	Standby	Continuous	Reserve	Reserve
ODR	Output Data Rate	10Hz	50Hz	100Hz	200Hz
RNG	Full Scale	2G	8G	Reserve	Reserve
OSR	Over Sample Ratio	512	256	128	64

Або інше завдання: Переслати код АЦП по інтерфейсу I2C на 16x2 LCD дисплей

Теоретичні відомості.

Порти що підтримують обробку даних за протоколом I2C наведено на рис.1

Рис. 1. Характеристики портів

Для створення проекту необхідно мати програми: STM32CubeMX, Keil µVision та Бібліотку HAL для роботи з дисплеєм.

Порядок виконання роботи:

В STM32CubeMX створюємо проект, рис.2.

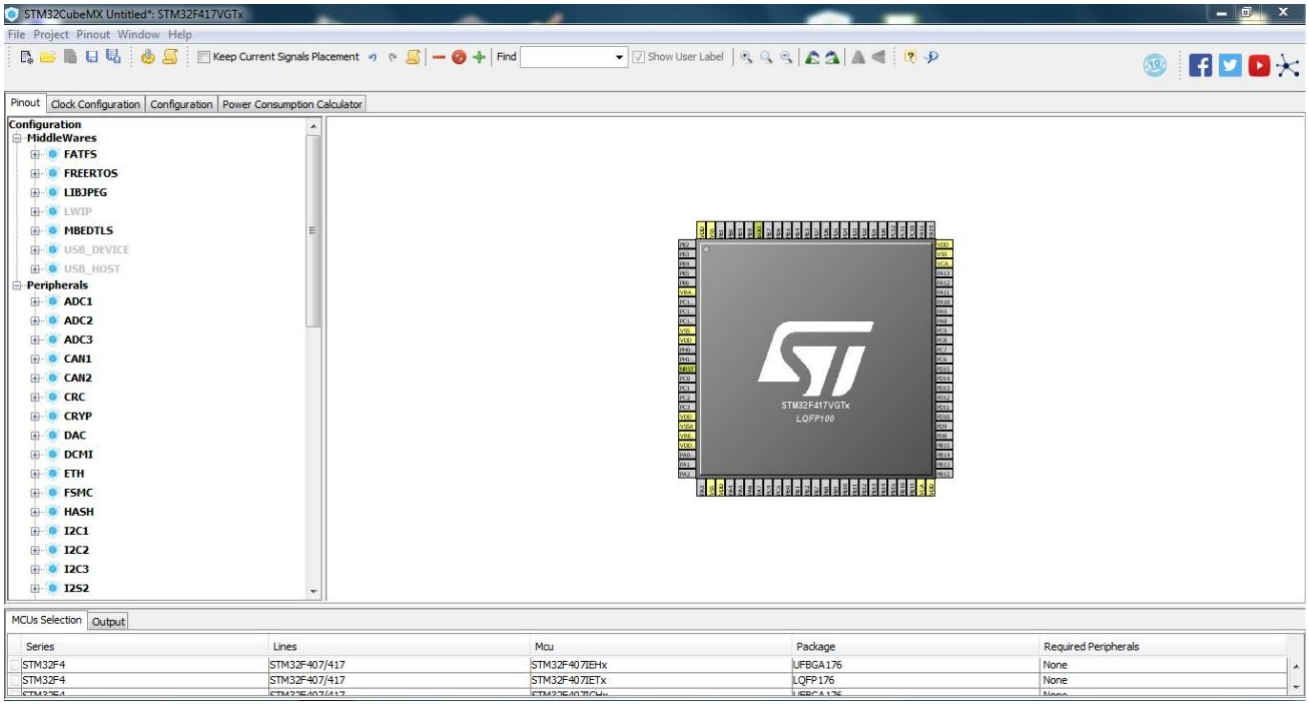


Рис 2 Вікно STM32CubeMX перед початком роботи

Обираємо перший АЦП та 5 вхід, рис.3

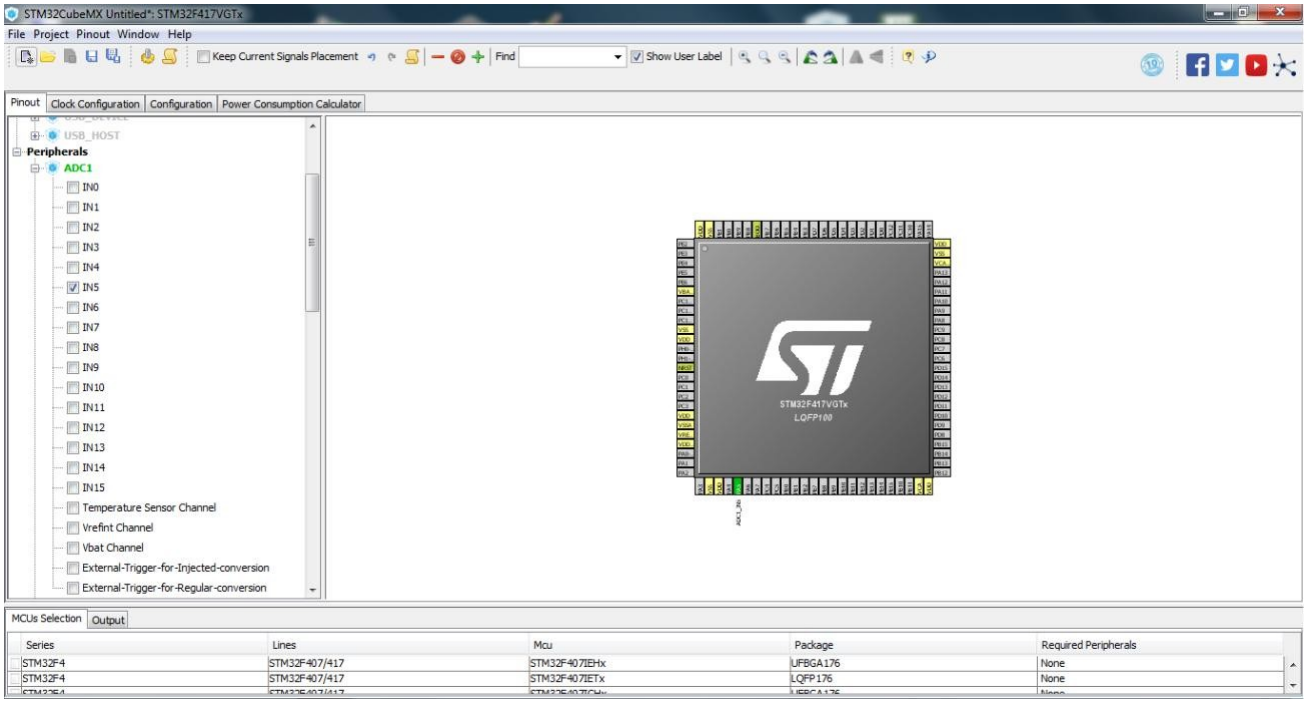


Рис 3 Вікно STM32CubeMX після вибору АЦП

Вигляд вікна програми після вибору входу АЦП наведено на рис.4

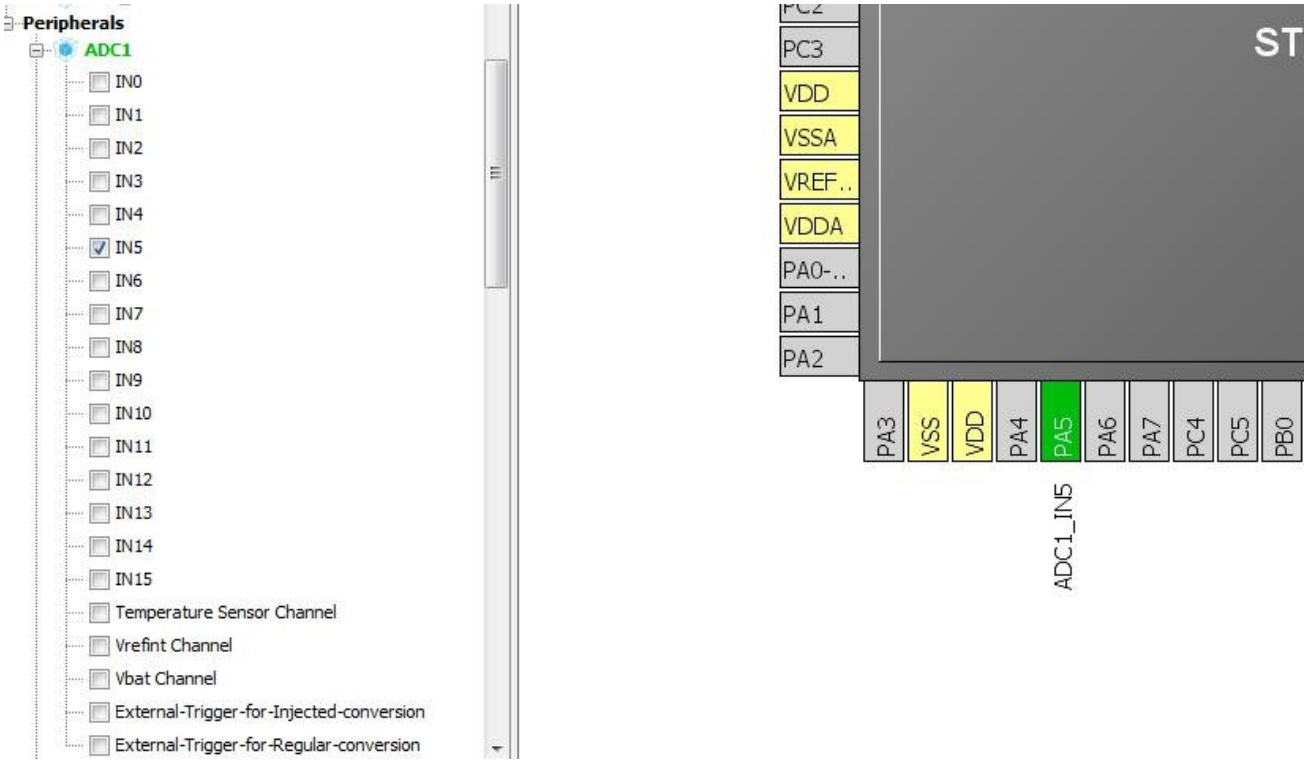


Рис.4

На цей вхід підключимо середній вивід потенцометра, інші 2 вивода - до +3V та GND. Меню конфігурації матиме вигляд, представлений на рис. 5 та рис.6.

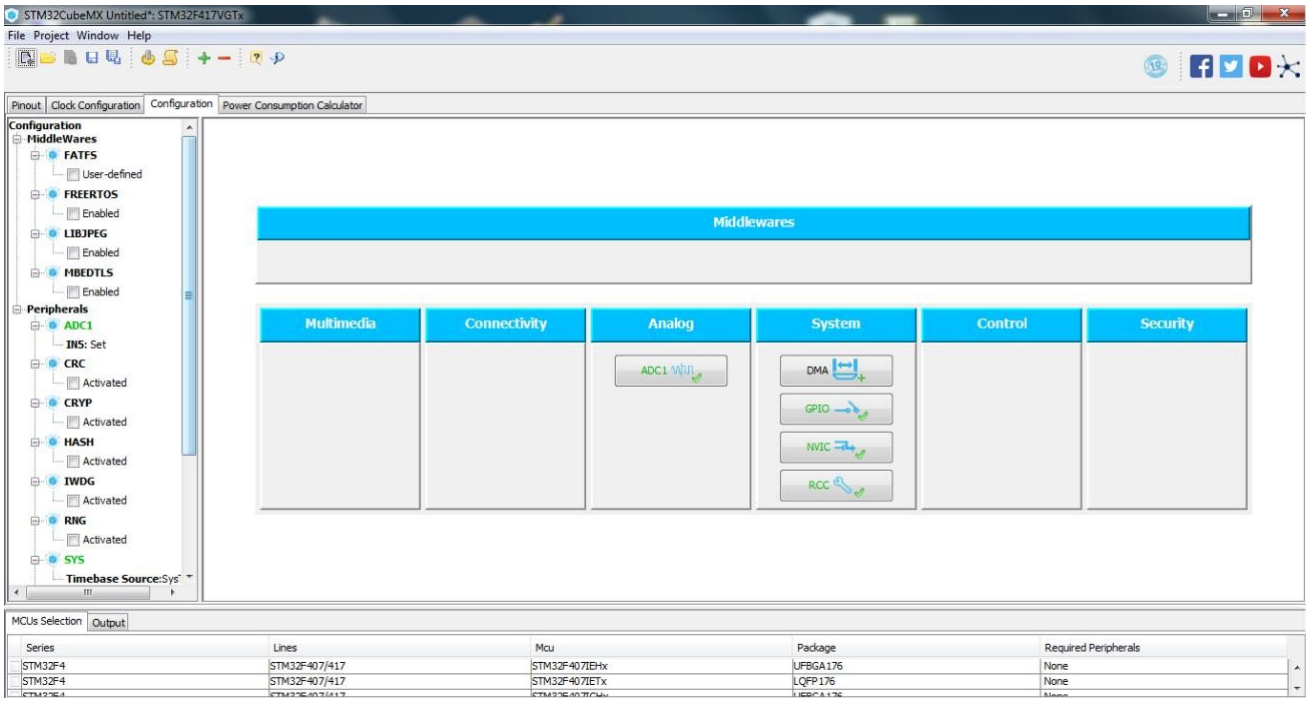


Рис.5

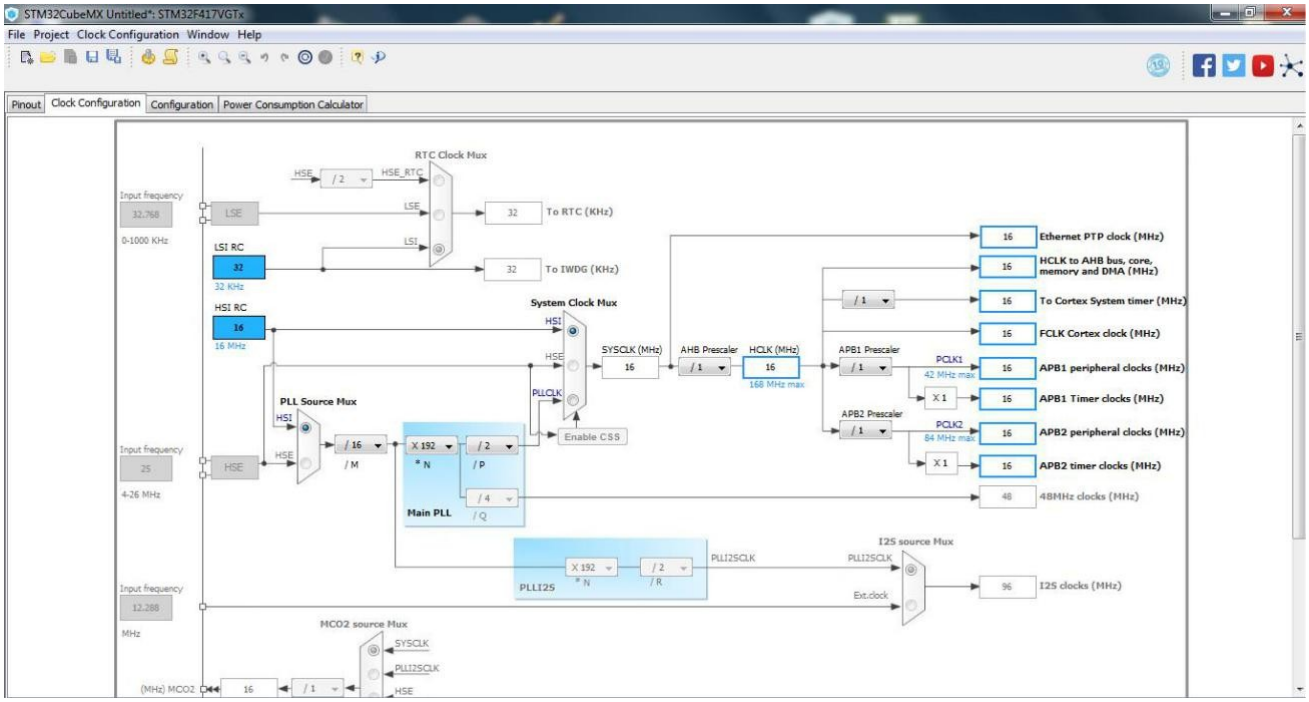


Рис.6

Налаштування АЦП здійснемо, клікнувши по ньому – відкриється меню підлаштувань (рис.7)

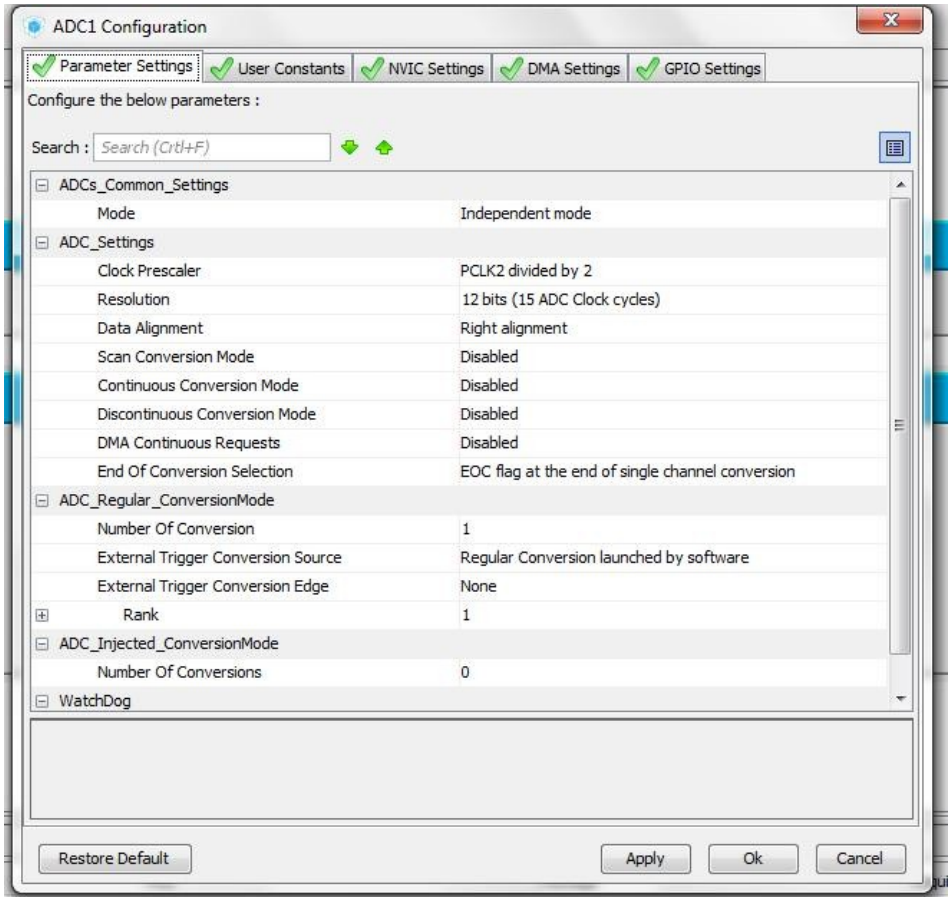


Рис. 7

Далі потрібно в Keil µVision створити необхідні змінні та згенерувати проект

```
34
35 int main(void)
36 {
37
38   /* USER CODE BEGIN 1 */
39   float u;
40   uint16_t i=0;
41   char str[9];
42   /* USER CODE END 1 */
43
44   /* MCU Configuration----------------
45
46   /* Reset of all peripherals, Initiali
47   HAL_Init();
48
```

float u;

uint16_t i=0;

char str[9];

Наприклад так оформити виведення тексту.

```
66    LCD_ini();
67    LCD_SetPos(5, 0);
68    sprintf(str,"Hello");
69    LCD_String(str);
70    HAL_Delay(500);
71    LCD_SetPos(1, 1);
72    sprintf(str,"Test 12bit ADC");
73    LCD_String(str);
74    HAL_Delay(3000);
75    LCD_Clear ();
76    LCD_SetPos(2, 0);
77    sprintf(str,"Stm32f407VG");
78    LCD_String(str);
79    HAL_Delay(2000);
80    LCD_Clear ();
81    LCD_SetPos(0, 1);
82    sprintf(str,"4095 -MAX Number");
83    LCD_String(str);
84    LCD_SetPos(0, 0);
85    sprintf(str,"0000 -Now Number");
86    LCD_String(str);
87    HAL_Delay(3000);
88    LCD_Clear ();
89    LCD_SetPos(5, 0);
90    sprintf(str,"-Now Number");
91    LCD_String(str);
92    LCD_SetPos(5, 1);
93    sprintf(str,"-Real Volt");
94    LCD_String(str);
```

```
LCD_ini();

LCD_SetPos(5, 0);

sprintf(str,"Hello");

LCD_String(str);

HAL_Delay(500);


LCD_SetPos(1, 1);

sprintf(str,"Test 12bit ADC");

LCD_String(str);

HAL_Delay(3000);

LCD_Clear ();

LCD_SetPos(2, 0);

sprintf(str,"Stm32f407VG");

LCD_String(str);

HAL_Delay(2000);

LCD_Clear ();
```

```
LCD_SetPos(0, 1);

sprintf(str,"4095 -MAX Number");

LCD_String(str);

LCD_SetPos(0, 0);

sprintf(str,"0000 -Now Number");

LCD_String(str);

HAL_Delay(3000);

LCD_Clear ();

LCD_SetPos(5, 0);

sprintf(str,"-Now Number");

LCD_String(str);

LCD_SetPos(5, 1);

sprintf(str,"-Real Volt");

LCD_String(str);
```

Також описуємо саму функцію АЦП та виводу зображення на дисплей.

```
/* USER CODE BEGIN WHILE */
while (1)
{
HAL_ADC_Start(&hadc1);
HAL_ADC_PollForConversion(&hadc1,100);
i=HAL_ADC_GetValue(&hadc1);
u=((float)HAL_ADC_GetValue(&hadc1)*3/4096);
HAL_ADC_Stop(&hadc1);
sprintf(str,"%04d",i);
LCD_SetPos (0,0);
LCD_String(str);
HAL_Delay(500);
sprintf(str,"%2.fv",u);
LCD_SetPos (0,1);
LCD_String(str);
}
/* USER CODE END 3 */
```

HAL_ADC_Start(&hadc1);

HAL_ADC_PollForConversion(&hadc1,100);

i=HAL_ADC_GetValue(&hadc1);

u=((float)HAL_ADC_GetValue(&hadc1)*3/4096);

HAL_ADC_Stop(&hadc1);

sprintf(str,"%04d",i);

LCD_SetPos (0,0);

LCD_String(str);

HAL_Delay(500);

sprintf(str,"%2.fv",u);

```
LCD_SetPos (0,1);

LCD_String(str);
```

Далі скомпілювавши проект та записав його в ядро, можна побачити, що при зміні опору потенціометра будуть змінюватися рівні напруги. Потенціометр варто взяти не менше 50К Ом.