**Міністерство Освіти І НАУКИ України**

**Національний університет "Львівська політехніка"**

**ІКНІ**

Кафедра **ПЗ**

**Звіт**

до лабораторної роботи № 2

**на тему: «**Робота з LCD дисплеєм WH1602B**»**

**з дисципліни:** “Програмування мікроконтролерів”

**Лектор:**

Марусенкова Т.А.

**Виконала:**

ст. гр. ПЗ-41

Бреднєва Є.Є.

**Прийняв:**

К.т.н.

Чопей Р.С.

«\_\_\_\_\_»  \_\_\_\_\_\_\_\_\_\_\_\_  2020 р.

∑ = \_\_\_\_\_\_    \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Львів 2020

**Тема:** Робота з LCD дисплеєм WH1602B

**Мета:** Навчитися організовувати взаємодію між мікроконтролером і LCD дисплеєм через паралельний інтерфейс.

**Завдання**

1. Використовуючи наведений приклад коду додати функції для запису байту даних та команд.
2. Додати функцію, що забезпечує: ініціалізацію дисплея (процедура ініціалізація зображена на Рис.3 ), встановлення курсора за певними координатами, вивід символа, вивід стрічки, а також здійснює очистку дисплея.
3. Реалізувати індивідуальне завдання:

4. Вивід одного з 5 константних повідомлень при натисканні на відповідну користувацьку клавішу. (HAL)

**Хід роботи**

#include "main.h"

#include "lcd.h"

uint8\_t mode=0;

void SystemClock\_Config(void);

static void MX\_GPIO\_Init(void);

int main(void)

{

HAL\_Init();

SystemClock\_Config();

MX\_GPIO\_Init();

lcdInit();

while (1)

{

if(mode!=0){

if(mode==1){

clear();

LCD\_string("Hello");

mode=0;

}

if(mode==2){

clear();

LCD\_string("How");

mode=0;

}

if(mode==3){

clear();

LCD\_string("Do");

mode=0;

}

if(mode==4){

clear();

LCD\_string("You");

mode=0;

}

if(mode==5){

clear();

LCD\_string("Do");

mode=0;

}

}

}

}

void SystemClock\_Config(void)

{

RCC\_OscInitTypeDef RCC\_OscInitStruct = {0};

RCC\_ClkInitTypeDef RCC\_ClkInitStruct = {0};

\_\_HAL\_RCC\_PWR\_CLK\_ENABLE();

\_\_HAL\_PWR\_VOLTAGESCALING\_CONFIG(PWR\_REGULATOR\_VOLTAGE\_SCALE1);

/\*\* Initializes the CPU, AHB and APB busses clocks

\*/

RCC\_OscInitStruct.OscillatorType = RCC\_OSCILLATORTYPE\_HSE;

RCC\_OscInitStruct.HSEState = RCC\_HSE\_ON;

RCC\_OscInitStruct.PLL.PLLState = RCC\_PLL\_ON;

RCC\_OscInitStruct.PLL.PLLSource = RCC\_PLLSOURCE\_HSE;

RCC\_OscInitStruct.PLL.PLLM = 4;

RCC\_OscInitStruct.PLL.PLLN = 168;

RCC\_OscInitStruct.PLL.PLLP = RCC\_PLLP\_DIV2;

RCC\_OscInitStruct.PLL.PLLQ = 4;

if (HAL\_RCC\_OscConfig(&RCC\_OscInitStruct) != HAL\_OK)

{

Error\_Handler();

}

/\*\* Initializes the CPU, AHB and APB busses clocks

\*/

RCC\_ClkInitStruct.ClockType = RCC\_CLOCKTYPE\_HCLK|RCC\_CLOCKTYPE\_SYSCLK

|RCC\_CLOCKTYPE\_PCLK1|RCC\_CLOCKTYPE\_PCLK2;

RCC\_ClkInitStruct.SYSCLKSource = RCC\_SYSCLKSOURCE\_PLLCLK;

RCC\_ClkInitStruct.AHBCLKDivider = RCC\_SYSCLK\_DIV1;

RCC\_ClkInitStruct.APB1CLKDivider = RCC\_HCLK\_DIV4;

RCC\_ClkInitStruct.APB2CLKDivider = RCC\_HCLK\_DIV2;

if (HAL\_RCC\_ClockConfig(&RCC\_ClkInitStruct, FLASH\_LATENCY\_5) != HAL\_OK)

{

Error\_Handler();

}

}

static void MX\_GPIO\_Init(void)

{

GPIO\_InitTypeDef GPIO\_InitStruct = {0};

/\* GPIO Ports Clock Enable \*/

\_\_HAL\_RCC\_GPIOH\_CLK\_ENABLE();

\_\_HAL\_RCC\_GPIOE\_CLK\_ENABLE();

\_\_HAL\_RCC\_GPIOC\_CLK\_ENABLE();

\_\_HAL\_RCC\_GPIOA\_CLK\_ENABLE();

/\*Configure GPIO pin Output Level \*/

HAL\_GPIO\_WritePin(GPIOE, GPIO\_PIN\_7|GPIO\_PIN\_10|GPIO\_PIN\_11|GPIO\_PIN\_12

|GPIO\_PIN\_13|GPIO\_PIN\_14|GPIO\_PIN\_15, GPIO\_PIN\_RESET);

/\*Configure GPIO pins : PE7 PE10 PE11 PE12

PE13 PE14 PE15 \*/

GPIO\_InitStruct.Pin = GPIO\_PIN\_7|GPIO\_PIN\_10|GPIO\_PIN\_11|GPIO\_PIN\_12

|GPIO\_PIN\_13|GPIO\_PIN\_14|GPIO\_PIN\_15;

GPIO\_InitStruct.Mode = GPIO\_MODE\_OUTPUT\_PP;

GPIO\_InitStruct.Pull = GPIO\_NOPULL;

GPIO\_InitStruct.Speed = GPIO\_SPEED\_FREQ\_LOW;

HAL\_GPIO\_Init(GPIOE, &GPIO\_InitStruct);

/\*Configure GPIO pins : PC6 PC8 PC9 PC11 \*/

GPIO\_InitStruct.Pin = GPIO\_PIN\_6|GPIO\_PIN\_8|GPIO\_PIN\_9|GPIO\_PIN\_11;

GPIO\_InitStruct.Mode = GPIO\_MODE\_IT\_FALLING;

GPIO\_InitStruct.Pull = GPIO\_NOPULL;

HAL\_GPIO\_Init(GPIOC, &GPIO\_InitStruct);

/\*Configure GPIO pin : PA15 \*/

GPIO\_InitStruct.Pin = GPIO\_PIN\_15;

GPIO\_InitStruct.Mode = GPIO\_MODE\_IT\_FALLING;

GPIO\_InitStruct.Pull = GPIO\_NOPULL;

HAL\_GPIO\_Init(GPIOA, &GPIO\_InitStruct);

}

void HAL\_GPIO\_EXTI\_Callback(uint16\_t GPIO\_Pin)

{

if(GPIO\_Pin== GPIO\_PIN\_6) {

mode=1;

}

if(GPIO\_Pin== GPIO\_PIN\_8) {

mode=2;

}

if(GPIO\_Pin== GPIO\_PIN\_9) {

mode=3;

}

if(GPIO\_Pin== GPIO\_PIN\_11) {

mode=4;

}

if(GPIO\_Pin== GPIO\_PIN\_15) {

mode=5;

}

}

void Error\_Handler(void)

{

/\* USER CODE BEGIN Error\_Handler\_Debug \*/

/\* User can add his own implementation to report the HAL error return state \*/

/\* USER CODE END Error\_Handler\_Debug \*/

}

#ifdef USE\_FULL\_ASSERT

/\*\*

\* @brief Reports the name of the source file and the source line number

\* where the assert\_param error has occurred.

\* @param file: pointer to the source file name

\* @param line: assert\_param error line source number

\* @retval None

\*/

void assert\_failed(uint8\_t \*file, uint32\_t line)

{

/\* USER CODE BEGIN 6 \*/

/\* User can add his own implementation to report the file name and line number,

tex: printf("Wrong parameters value: file %s on line %d\r\n", file, line) \*/

/\* USER CODE END 6 \*/

}

#endif /\* USE\_FULL\_ASSERT \*/

**Висновок**

Я навчилася організовувати взаємодію між мікроконтролером і LCD дисплеем через паралельний інтерфейс та реалізувала індивідуальне завдання: вивід одного з 5 константних повідомлень при натисканні на відповідну користувацьку клавішу.