**BỘ CÔNG THƯƠNG**

**TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP**

**TP.HCM KHOA CÔNG NGHỆ ĐIỆN TỬ**

BÁO CÁO

VI ĐIỀU KHIỂN VÀ ỨNG DỤNG

**BÀI 5:**

***BÀI THỰC HÀNH V***



#### Giảng viên hướng dẫn: Phạm Quang Trí

**NHÓM 3:**

**Sinh viên thực hiện:**

**Đoàn Mạnh Tú - 21007261**

**Hồ Bửu Sơn - 21119571**

***Thành phố Hồ Chí Minh, tháng 8 năm 2023***

**Bài tập mức độ 3:Viết chương trình cho STM32 thực hiện yêu cầu sau:**

**-Truyền một gói tin từ máy tính xuống STM32 có định dạng như sau:\*|X|DATA|# (bao gồm nhiều byte, trong đó “\*” là ký tự bắt đầu; “#” là ký tự kết thúc; “X”(X=1->16) là byte khai báo kích thước của Data; “Data” là các ký tự, có kích thước từ 1 đến 16 byte.Ví dụ:\*3ABC#.**

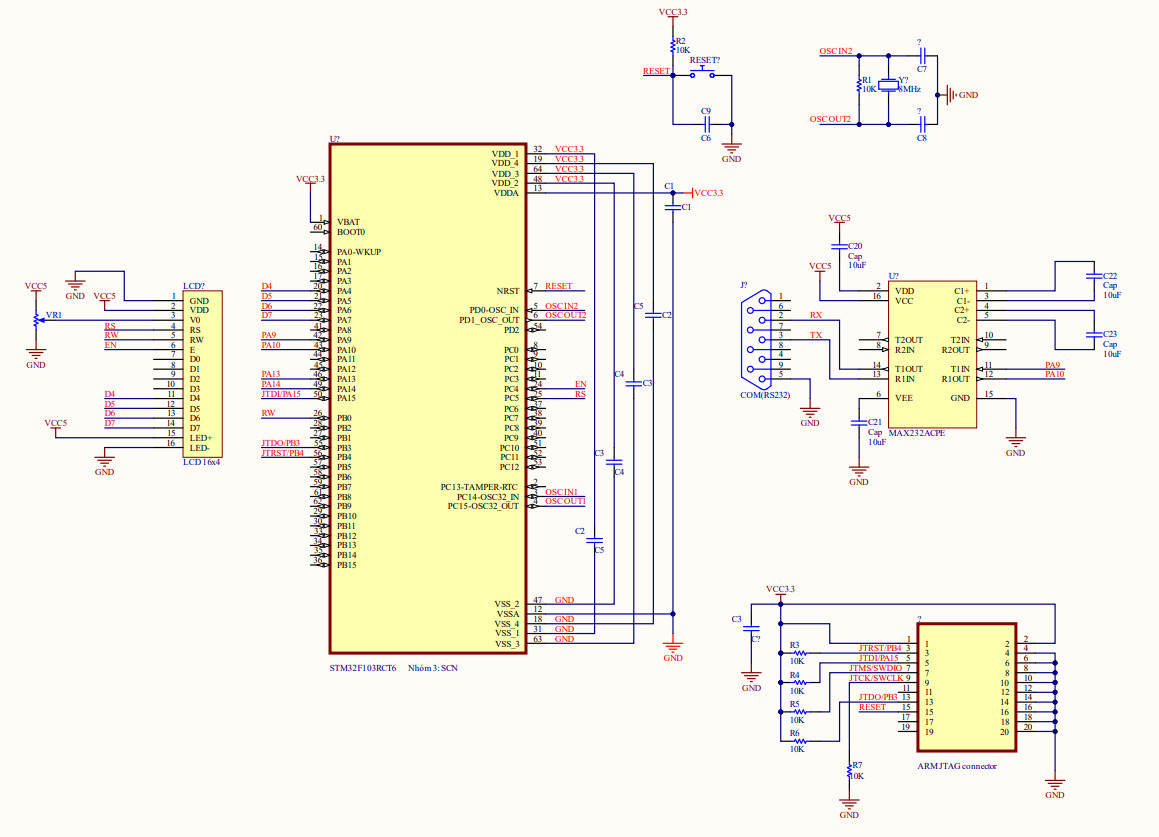
**-Khi nhận đúng ddingj dạng gói tin:hiển thị nội dung gói tin lên LCD 16x2, đồng thời truyền trở về máy tính chữ “OK” thông qua UART.**

**-Khi nhận sau định dạng gói tin:không thay đổi nội dung LCD,đồng thời truyền trở về máy tính chữ “ERROR” thông qua UART.**

**-Minh chứng phải thể hiện đầy đủ các trường hợp gửi đúng và gửi sai định dạng gói tin như sai ký tự bắt đầu, sai ký tự kết thúc,…**

**Báo cáo thực hành**

***Yêu cầu 1:***Vẽ sơ đồ nguyên lý kết nối phần cứng của toàn bộ hệ thống sử dụng vi điều khiển STM32.



***Yêu cầu 2:***Trình bày lưu đồ giải thuật và mã nguồn của chương trình.

**Mã nguồn:**

**Chương trình main**

/\* USER CODE BEGIN Header \*/

/\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* @file : main.c

\* @brief : Main program body

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* @attention

\*

\* Copyright (c) 2023 STMicroelectronics.

\* All rights reserved.

\*

\* This software is licensed under terms that can be found in the LICENSE file

\* in the root directory of this software component.

\* If no LICENSE file comes with this software, it is provided AS-IS.

\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*/

/\* USER CODE END Header \*/

/\* Includes ------------------------------------------------------------------\*/

**#include** "main.h"

/\* Private includes ----------------------------------------------------------\*/

/\* USER CODE BEGIN Includes \*/

/\* USER CODE END Includes \*/

/\* Private typedef -----------------------------------------------------------\*/

/\* USER CODE BEGIN PTD \*/

/\* USER CODE END PTD \*/

/\* Private define ------------------------------------------------------------\*/

/\* USER CODE BEGIN PD \*/

/\* USER CODE END PD \*/

/\* Private macro -------------------------------------------------------------\*/

/\* USER CODE BEGIN PM \*/

/\* USER CODE END PM \*/

/\* Private variables ---------------------------------------------------------\*/

UART\_HandleTypeDef huart1;

/\* USER CODE BEGIN PV \*/

uint8\_t OK[3] = {"OK\n"};

uint8\_t error[6] = {"ERROR\n"};

uint8\_t save[16];

uint8\_t data\_receive[16];

uint8\_t dem;

**int** t = 0;

/\* USER CODE END PV \*/

/\* Private function prototypes -----------------------------------------------\*/

**void** **SystemClock\_Config**(**void**);

**static** **void** **MX\_GPIO\_Init**(**void**);

**static** **void** **MX\_USART1\_UART\_Init**(**void**);

/\* USER CODE BEGIN PFP \*/

/\* USER CODE END PFP \*/

/\* Private user code ---------------------------------------------------------\*/

/\* USER CODE BEGIN 0 \*/

**void** **lcd\_write\_nibble**(uint8\_t rs, uint8\_t data){

HAL\_GPIO\_WritePin(RS\_GPIO\_Port, RS\_Pin,rs);

HAL\_GPIO\_WritePin(RW\_GPIO\_Port, RW\_Pin,0);

HAL\_GPIO\_WritePin(EN\_GPIO\_Port, EN\_Pin,1);

HAL\_GPIO\_WritePin(D4\_GPIO\_Port, D4\_Pin, (data>>0) & 0x01);

HAL\_GPIO\_WritePin(D5\_GPIO\_Port, D5\_Pin, (data>>1) & 0x01);

HAL\_GPIO\_WritePin(D6\_GPIO\_Port, D6\_Pin, (data>>2) & 0x01);

HAL\_GPIO\_WritePin(D7\_GPIO\_Port, D7\_Pin, (data>>3) & 0x01);

**for**(uint8\_t i=0; i<72; i++) **asm**("NOP");

HAL\_GPIO\_WritePin(EN\_GPIO\_Port, EN\_Pin,0);

**for**(uint8\_t i=0; i<72; i++) **asm**("NOP");

}

**void** **lcd\_send\_cmd**(uint8\_t cmd){

lcd\_write\_nibble(0, (cmd>>4)&0x0F);

lcd\_write\_nibble(0, cmd&0x0F);

HAL\_Delay(2);

}

**void** **lcd\_send\_data**(uint8\_t data){

lcd\_write\_nibble(1, (data>>4)&0x0F);

lcd\_write\_nibble(1, data&0x0F);

HAL\_Delay(2);

}

**void** **lcd\_init**(**void**){

HAL\_Delay(20);

lcd\_write\_nibble(0, 0x03); HAL\_Delay(5);

lcd\_write\_nibble(0, 0x03); HAL\_Delay(1);

lcd\_write\_nibble(0, 0x03); HAL\_Delay(1);

lcd\_write\_nibble(0, 0x02); HAL\_Delay(1);

lcd\_send\_cmd(0x28);

lcd\_send\_cmd(0x0C);

lcd\_send\_cmd(0x01);

lcd\_send\_cmd(0x06);

}

**void** **lcd\_gotoxy**(uint8\_t row, uint8\_t col){

uint8\_t coordinates = 0;

**switch**(row){

**case** 0:

coordinates = 0x80 | col;

**break**;

**case** 1:

coordinates = 0xC0 | col;

**break**;

}

}

**void** **lcd\_display**(**char** \*data){

**while**(\*data){

lcd\_send\_data(\*data++);

}

}

/\* USER CODE END 0 \*/

/\*\*

\* @brief The application entry point.

\* @retval int

\*/

**int** **main**(**void**)

{

/\* USER CODE BEGIN 1 \*/

/\* USER CODE END 1 \*/

/\* MCU Configuration--------------------------------------------------------\*/

/\* Reset of all peripherals, Initializes the Flash interface and the Systick. \*/

HAL\_Init();

/\* USER CODE BEGIN Init \*/

/\* USER CODE END Init \*/

/\* Configure the system clock \*/

SystemClock\_Config();

/\* USER CODE BEGIN SysInit \*/

/\* USER CODE END SysInit \*/

/\* Initialize all configured peripherals \*/

MX\_GPIO\_Init();

MX\_USART1\_UART\_Init();

/\* USER CODE BEGIN 2 \*/

lcd\_init();

lcd\_display("Nhom 3");

/\* USER CODE END 2 \*/

/\* Infinite loop \*/

/\* USER CODE BEGIN WHILE \*/

**while** (1)

{

/\* USER CODE END WHILE \*/

/\* USER CODE BEGIN 3 \*/

**if**(t==1){

**if**(save[(save[1] + 2)] == 0x23){

**if** (save[0] == 0x2a){

**if**(save[1] == dem - 3){

lcd\_send\_cmd(0x01);

HAL\_Delay(10);

lcd\_gotoxy(0,0);

**for** (**int** i = 0; i < save[1]; i++){

lcd\_send\_data(save[i + 2]);

}

t = 0;

dem = 0;

HAL\_UART\_Transmit(&huart1, &OK[0],3,10);

}

**else**{

t = 0;

dem = 0;

HAL\_UART\_Transmit(&huart1, &error[0],6,10);

}

}

**if**(save[0]!=0x2a){

t = 0;

dem = 0;

HAL\_UART\_Transmit(&huart1, &error[0],6,20);

}

}

**if**(save[(save[1] + 2)]!=0x23){

t = 0;

dem = 0;

HAL\_UART\_Transmit(&huart1, &error[0],6,20);

}

}

HAL\_Delay(500);

}

/\* USER CODE END 3 \*/

}

/\*\*

\* @brief System Clock Configuration

\* @retval None

\*/

**void** **SystemClock\_Config**(**void**)

{

RCC\_OscInitTypeDef RCC\_OscInitStruct = {0};

RCC\_ClkInitTypeDef RCC\_ClkInitStruct = {0};

/\*\* Initializes the RCC Oscillators according to the specified parameters

\* in the RCC\_OscInitTypeDef structure.

\*/

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

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

RCC\_OscInitStruct.HSEPredivValue = RCC\_HSE\_PREDIV\_DIV1;

RCC\_OscInitStruct.HSIState = RCC\_HSI\_ON;

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

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

RCC\_OscInitStruct.PLL.PLLMUL = RCC\_PLL\_MUL9;

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

{

Error\_Handler();

}

/\*\* Initializes the CPU, AHB and APB buses 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\_DIV2;

RCC\_ClkInitStruct.APB2CLKDivider = RCC\_HCLK\_DIV1;

**if** (HAL\_RCC\_ClockConfig(&RCC\_ClkInitStruct, FLASH\_LATENCY\_2) != *HAL\_OK*)

{

Error\_Handler();

}

}

/\*\*

\* @brief USART1 Initialization Function

\* @param None

\* @retval None

\*/

**static** **void** **MX\_USART1\_UART\_Init**(**void**)

{

/\* USER CODE BEGIN USART1\_Init 0 \*/

/\* USER CODE END USART1\_Init 0 \*/

/\* USER CODE BEGIN USART1\_Init 1 \*/

/\* USER CODE END USART1\_Init 1 \*/

huart1.Instance = USART1;

huart1.Init.BaudRate = 38400;

huart1.Init.WordLength = UART\_WORDLENGTH\_9B;

huart1.Init.StopBits = UART\_STOPBITS\_1;

huart1.Init.Parity = UART\_PARITY\_ODD;

huart1.Init.Mode = UART\_MODE\_TX\_RX;

huart1.Init.HwFlowCtl = UART\_HWCONTROL\_NONE;

huart1.Init.OverSampling = UART\_OVERSAMPLING\_16;

**if** (HAL\_UART\_Init(&huart1) != *HAL\_OK*)

{

Error\_Handler();

}

/\* USER CODE BEGIN USART1\_Init 2 \*/

/\* USER CODE END USART1\_Init 2 \*/

}

/\*\*

\* @brief GPIO Initialization Function

\* @param None

\* @retval None

\*/

**static** **void** **MX\_GPIO\_Init**(**void**)

{

GPIO\_InitTypeDef GPIO\_InitStruct = {0};

/\* USER CODE BEGIN MX\_GPIO\_Init\_1 \*/

/\* USER CODE END MX\_GPIO\_Init\_1 \*/

/\* GPIO Ports Clock Enable \*/

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

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

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

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

/\*Configure GPIO pin Output Level \*/

HAL\_GPIO\_WritePin(GPIOA, D4\_Pin|D5\_Pin|D6\_Pin|D7\_Pin, *GPIO\_PIN\_RESET*);

/\*Configure GPIO pin Output Level \*/

HAL\_GPIO\_WritePin(GPIOC, EN\_Pin|RS\_Pin, *GPIO\_PIN\_RESET*);

/\*Configure GPIO pin Output Level \*/

HAL\_GPIO\_WritePin(RW\_GPIO\_Port, RW\_Pin, *GPIO\_PIN\_RESET*);

/\*Configure GPIO pins : D4\_Pin D5\_Pin D6\_Pin D7\_Pin \*/

GPIO\_InitStruct.Pin = D4\_Pin|D5\_Pin|D6\_Pin|D7\_Pin;

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

GPIO\_InitStruct.Pull = GPIO\_NOPULL;

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

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

/\*Configure GPIO pins : EN\_Pin RS\_Pin \*/

GPIO\_InitStruct.Pin = EN\_Pin|RS\_Pin;

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

GPIO\_InitStruct.Pull = GPIO\_NOPULL;

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

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

/\*Configure GPIO pin : RW\_Pin \*/

GPIO\_InitStruct.Pin = RW\_Pin;

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

GPIO\_InitStruct.Pull = GPIO\_NOPULL;

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

HAL\_GPIO\_Init(RW\_GPIO\_Port, &GPIO\_InitStruct);

/\* USER CODE BEGIN MX\_GPIO\_Init\_2 \*/

/\* USER CODE END MX\_GPIO\_Init\_2 \*/

}

/\* USER CODE BEGIN 4 \*/

/\* USER CODE END 4 \*/

/\*\*

\* @brief This function is executed in case of error occurrence.

\* @retval None

\*/

**void** **Error\_Handler**(**void**)

{

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

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

\_\_disable\_irq();

**while** (1) {

}

/\* 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,

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

/\* USER CODE END 6 \*/

}

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

**Chương trình ngắt**

/\* USER CODE BEGIN Header \*/

/\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* @file stm32f1xx\_it.c

\* @brief Interrupt Service Routines.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* @attention

\*

\* Copyright (c) 2023 STMicroelectronics.

\* All rights reserved.

\*

\* This software is licensed under terms that can be found in the LICENSE file

\* in the root directory of this software component.

\* If no LICENSE file comes with this software, it is provided AS-IS.

\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*/

/\* USER CODE END Header \*/

/\* Includes ------------------------------------------------------------------\*/

**#include** "main.h"

**#include** "stm32f1xx\_it.h"

/\* Private includes ----------------------------------------------------------\*/

/\* USER CODE BEGIN Includes \*/

/\* USER CODE END Includes \*/

/\* Private typedef -----------------------------------------------------------\*/

/\* USER CODE BEGIN TD \*/

/\* USER CODE END TD \*/

/\* Private define ------------------------------------------------------------\*/

/\* USER CODE BEGIN PD \*/

/\* USER CODE END PD \*/

/\* Private macro -------------------------------------------------------------\*/

/\* USER CODE BEGIN PM \*/

/\* USER CODE END PM \*/

/\* Private variables ---------------------------------------------------------\*/

/\* USER CODE BEGIN PV \*/

/\* USER CODE END PV \*/

/\* Private function prototypes -----------------------------------------------\*/

/\* USER CODE BEGIN PFP \*/

/\* USER CODE END PFP \*/

/\* Private user code ---------------------------------------------------------\*/

/\* USER CODE BEGIN 0 \*/

/\* USER CODE END 0 \*/

/\* External variables --------------------------------------------------------\*/

**extern** UART\_HandleTypeDef huart1;

**extern** uint8\_t save[16];

**extern** uint8\_t data\_receive[16];

**extern** uint8\_t dem;

**extern** **int** t;

/\* USER CODE BEGIN EV \*/

/\* USER CODE END EV \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Cortex-M3 Processor Interruption and Exception Handlers \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*

\* @brief This function handles Non maskable interrupt.

\*/

**void** **NMI\_Handler**(**void**)

{

/\* USER CODE BEGIN NonMaskableInt\_IRQn 0 \*/

/\* USER CODE END NonMaskableInt\_IRQn 0 \*/

/\* USER CODE BEGIN NonMaskableInt\_IRQn 1 \*/

**while** (1)

{

}

/\* USER CODE END NonMaskableInt\_IRQn 1 \*/

}

/\*\*

\* @brief This function handles Hard fault interrupt.

\*/

**void** **HardFault\_Handler**(**void**)

{

/\* USER CODE BEGIN HardFault\_IRQn 0 \*/

/\* USER CODE END HardFault\_IRQn 0 \*/

**while** (1)

{

/\* USER CODE BEGIN W1\_HardFault\_IRQn 0 \*/

/\* USER CODE END W1\_HardFault\_IRQn 0 \*/

}

}

/\*\*

\* @brief This function handles Memory management fault.

\*/

**void** **MemManage\_Handler**(**void**)

{

/\* USER CODE BEGIN MemoryManagement\_IRQn 0 \*/

/\* USER CODE END MemoryManagement\_IRQn 0 \*/

**while** (1)

{

/\* USER CODE BEGIN W1\_MemoryManagement\_IRQn 0 \*/

/\* USER CODE END W1\_MemoryManagement\_IRQn 0 \*/

}

}

/\*\*

\* @brief This function handles Prefetch fault, memory access fault.

\*/

**void** **BusFault\_Handler**(**void**)

{

/\* USER CODE BEGIN BusFault\_IRQn 0 \*/

/\* USER CODE END BusFault\_IRQn 0 \*/

**while** (1)

{

/\* USER CODE BEGIN W1\_BusFault\_IRQn 0 \*/

/\* USER CODE END W1\_BusFault\_IRQn 0 \*/

}

}

/\*\*

\* @brief This function handles Undefined instruction or illegal state.

\*/

**void** **UsageFault\_Handler**(**void**)

{

/\* USER CODE BEGIN UsageFault\_IRQn 0 \*/

/\* USER CODE END UsageFault\_IRQn 0 \*/

**while** (1)

{

/\* USER CODE BEGIN W1\_UsageFault\_IRQn 0 \*/

/\* USER CODE END W1\_UsageFault\_IRQn 0 \*/

}

}

/\*\*

\* @brief This function handles System service call via SWI instruction.

\*/

**void** **SVC\_Handler**(**void**)

{

/\* USER CODE BEGIN SVCall\_IRQn 0 \*/

/\* USER CODE END SVCall\_IRQn 0 \*/

/\* USER CODE BEGIN SVCall\_IRQn 1 \*/

/\* USER CODE END SVCall\_IRQn 1 \*/

}

/\*\*

\* @brief This function handles Debug monitor.

\*/

**void** **DebugMon\_Handler**(**void**)

{

/\* USER CODE BEGIN DebugMonitor\_IRQn 0 \*/

/\* USER CODE END DebugMonitor\_IRQn 0 \*/

/\* USER CODE BEGIN DebugMonitor\_IRQn 1 \*/

/\* USER CODE END DebugMonitor\_IRQn 1 \*/

}

/\*\*

\* @brief This function handles Pendable request for system service.

\*/

**void** **PendSV\_Handler**(**void**)

{

/\* USER CODE BEGIN PendSV\_IRQn 0 \*/

/\* USER CODE END PendSV\_IRQn 0 \*/

/\* USER CODE BEGIN PendSV\_IRQn 1 \*/

/\* USER CODE END PendSV\_IRQn 1 \*/

}

/\*\*

\* @brief This function handles System tick timer.

\*/

**void** **SysTick\_Handler**(**void**)

{

/\* USER CODE BEGIN SysTick\_IRQn 0 \*/

/\* USER CODE END SysTick\_IRQn 0 \*/

HAL\_IncTick();

/\* USER CODE BEGIN SysTick\_IRQn 1 \*/

/\* USER CODE END SysTick\_IRQn 1 \*/

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* STM32F1xx Peripheral Interrupt Handlers \*/

/\* Add here the Interrupt Handlers for the used peripherals. \*/

/\* For the available peripheral interrupt handler names, \*/

/\* please refer to the startup file (startup\_stm32f1xx.s). \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*

\* @brief This function handles USART1 global interrupt.

\*/

**void** **USART1\_IRQHandler**(**void**)

{

/\* USER CODE BEGIN USART1\_IRQn 0 \*/

/\* USER CODE END USART1\_IRQn 0 \*/

HAL\_UART\_IRQHandler(&huart1);

/\* USER CODE BEGIN USART1\_IRQn 1 \*/

save[dem]=data\_receive[0];

HAL\_UART\_Receive\_IT(&huart1,&data\_receive[0],1);

dem++;

t = 1;

/\* USER CODE END USART1\_IRQn 1 \*/

}

/\* USER CODE BEGIN 1 \*/

/\* USER CODE END 1 \*/

**Lưu đồ:**

Chương trình con gửi 8 bit dữ liệu sang LCD chế độ 4 bit:

A diagram of a computer

Description automatically generated

Chương trình con gửi lệnh điều khiển sang LCD:

A diagram of a computer program

Description automatically generated

Chương trình con cho phép hiện thị 1 ký tự trên mô-đun LCD:

A diagram of a computer

Description automatically generated

Chương trình con khởi động LCD chế độ 4 bit:

A diagram of a program

Description automatically generated

Chương trình con cho phép hiển thị 1 chuỗi trên mô-đun LCD:

A diagram of a diagram

Description automatically generated

Chương trình con định vị con trỏ trên LCD:

A diagram of a graph

Description automatically generated

Chương trình main

A diagram of a flowchart

Description automatically generated

***Yêu cầu 3:***Video bài làm

https://youtu.be/eL3s2LXHyFY