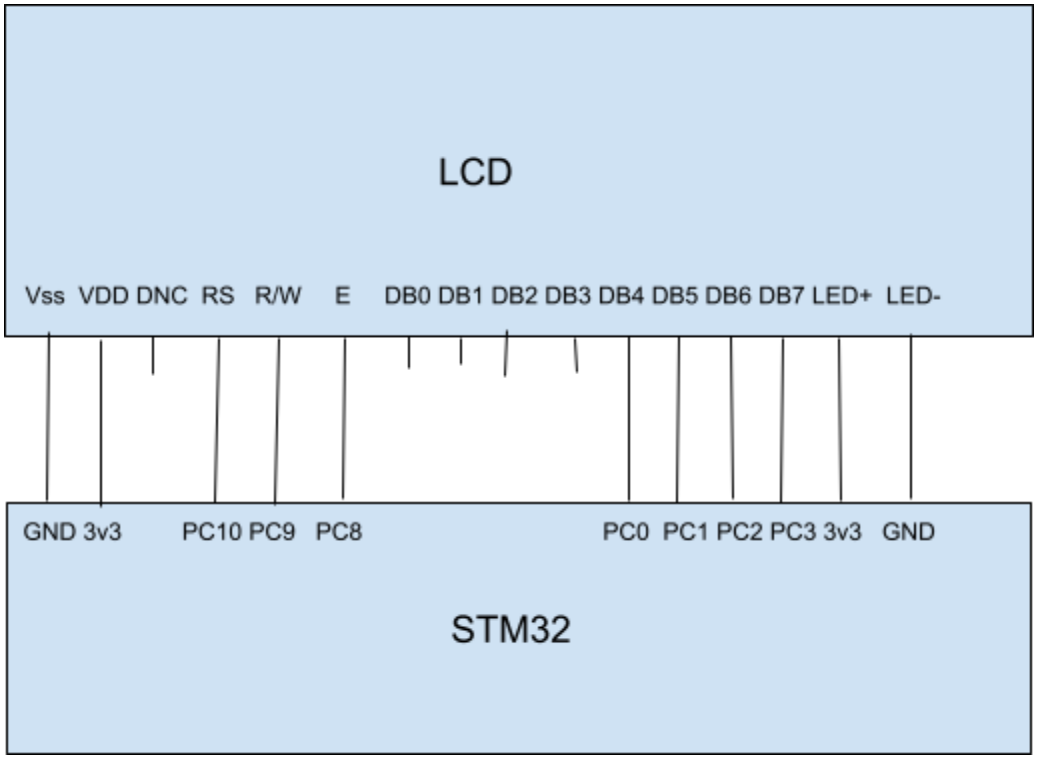
Benchohra, Sereen
Collison, Neil
CPE 329

A3 -LCD

Demo: In class to Tamara



---

main.c

```
#include "main.h"
#include <time.h>
#include "LCD.h"

void SystemClock_Config(void);
void SysTick_Init(void);

int main(void)
{

  HAL_Init();

  SystemClock_Config();

  SysTick_Init();

  LCD_init();
```

```
  LCD_write_string("Hello World");
  LCD_command(SECOND_LINE_HOME);
  LCD_write_string("Assignment 3");
  while (1);

}
```

```
void SysTick_Init(void)
{
    SysTick->CTRL |= (SysTick_CTRL_ENABLE_Msk |           // enable SysTick Timer
                      SysTick_CTRL_CLKSOURCE_Msk);     // select CPU clock
    SysTick->CTRL &= ~(SysTick_CTRL_TICKINT_Msk);      // disable interrupt,
                                                       // breaks HAL delay function
}
```

```
void SystemClock_Config(void)
{
  RCC_OscInitTypeDef RCC_OscInitStruct = {0};
  RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};

  /** Configure the main internal regulator output voltage
  */
  if (HAL_PWREx_ControlVoltageScaling(PWR_REGULATOR_VOLTAGE_SCALE1) != HAL_OK)
  {
    Error_Handler();
  }
  /** Initializes the RCC Oscillators according to the specified parameters
  * in the RCC_OscInitTypeDef structure.
  */
  RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_MSI;
  RCC_OscInitStruct.MSIState = RCC_MSI_ON;
  RCC_OscInitStruct.MSICalibrationValue = 0;
  RCC_OscInitStruct.MSIClockRange = RCC_MSIRANGE_6;
  RCC_OscInitStruct.PLL.PLLState = RCC_PLL_NONE;
  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_MSI;
  RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
  RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV1;
  RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV1;

  if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_0) != HAL_OK)
  {
    Error_Handler();
  }
}

/* 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 */

/************************ (C) COPYRIGHT STMicroelectronics *****END OF FILE****/
```

LCD.h

```
#ifndef SRC_LCD_H_
#define SRC_LCD_H_

#define RS_bit    (GPIO_ODR_OD10)
#define RW_bit    (GPIO_ODR_OD9)
#define E_bit     (GPIO_ODR_OD8)
#define data_bits (GPIO_ODR_OD0 | GPIO_ODR_OD1 | GPIO_ODR_OD2 | GPIO_ODR_OD3)
#define SECOND_LINE_HOME 0xC0
#define INCREMENT_CURSOR 0x06
#define CLEAR_SCREEN 0x01
#define CURSOR_BLINKER_ON 0x0F
#define FUNCTION_SET 0x28
#define FOUR_BIT_MODE 0x3


void LCD_init (void);
void LCD_command (uint8_t command_byte);
void LCD_write_nib (uint8_t data_nib);  // will only work in the command
subroutine?
void LCD_write_char_nib (uint8_t char_nib);
void LCD_write_char (uint8_t char_byte);

void delay_us(const uint16_t time_us);


#endif /* SRC_LCD_H_ */
```

LCD.c

```
#include "main.h"
#include "LCD.h"


void LCD_init (void)
{
```

```
       RCC->AHB2ENR    |=  (RCC_AHB2ENR_GPIOCEN); //enables bus C

       GPIOC->MODER    &= ~(GPIO_MODER_MODE0 | GPIO_MODER_MODE1 | GPIO_MODER_MODE2 |
GPIO_MODER_MODE3
                           | GPIO_MODER_MODE8 | GPIO_MODER_MODE9 |
GPIO_MODER_MODE10);

    GPIOC->MODER    |=  (GPIO_MODER_MODE0_0 | GPIO_MODER_MODE1_0 |
GPIO_MODER_MODE2_0 | GPIO_MODER_MODE3_0
                        | GPIO_MODER_MODE8_0 | GPIO_MODER_MODE9_0 |
GPIO_MODER_MODE10_0); //set c pins 0-3 and 8-10 to output (for LCD data then
control)

    GPIOC->OTYPER  &= ~(GPIO_OTYPER_OT0 | GPIO_OTYPER_OT1 | GPIO_OTYPER_OT2 |
GPIO_OTYPER_OT3
                        | GPIO_OTYPER_OT8 | GPIO_OTYPER_OT9 | GPIO_OTYPER_OT10);
//pins C0-3 and 8-10 to push pull

    GPIOC->OSPEEDR &= ~(GPIO_OSPEEDR_OSPEED0 | GPIO_OSPEEDR_OSPEED1 |
GPIO_OSPEEDR_OSPEED2 | GPIO_OSPEEDR_OSPEED3
                        | GPIO_OSPEEDR_OSPEED8 | GPIO_OSPEEDR_OSPEED9 |
GPIO_OSPEEDR_OSPEED10);  //slow speed on output pins c 0-3 and 8-10


       // C pins 0-3 will be for data (d4-7); C pin 10->RS; pin 9->RW pin 8->E

    for (int i=0; i< 400;i++)
    {
       delay_us(1000);
    }

       LCD_write_nib(FOUR_BIT_MODE); //4 bit mode
       delay_us(1000);
       LCD_command(FUNCTION_SET); //function set
       delay_us(1000);
       LCD_command( FUNCTION_SET); //again for some reason
       delay_us(1000);
       LCD_command (CURSOR_BLINKER_ON); //turns on cursor, blinker
       delay_us(1000);
       LCD_command(CLEAR_SCREEN); //clears screen
       delay_us(1000);
       // delay_us(520);
       delay_us(1000);

       LCD_command (INCREMENT_CURSOR);// increments cursor

}
```

```
void LCD_write_nib (uint8_t data_nib)
{

	GPIOC->ODR &=  ~(RW_bit | RS_bit); //sets pins connected to RS, RW, and data pins to zero

	GPIOC->ODR &=  ~(data_bits);

	GPIOC->ODR |=  (data_nib);

	GPIOC->ODR |=  (E_bit); //puts the ones corresponding to the data inputed in the ODR at bits 0-4

	delay_us(1000); //this one we need

	GPIOC->ODR &=  ~(E_bit); //sets Enable pin to zero

	return;
}


void LCD_command (uint8_t command_byte)
{
//	GPIOC->ODR &=  ~(RW_bit | RS_bit);  //maybe take out?
	uint8_t ls_nib = 0x0F & command_byte; // spilts command into upper and lower nibbles to send in 4 bits at a time

	uint8_t ms_nib = 0xF0 & command_byte;

	ms_nib= ms_nib>>4;

//	delay_us(1520);

	LCD_write_nib (ms_nib);

	delay_us(2000);   //this we need

	LCD_write_nib (ls_nib);

//	delay_us(1520);
}

void LCD_write_char_nib (uint8_t char_nib){

	GPIOC->ODR &=  ~(RW_bit); //sets pins connected to RS, and data pins to zero
```

```
	GPIOC->ODR &=  ~(E_bit);
	GPIOC->ODR |=   (RS_bit); // set RS bit for writing

	delay_us(1000);

	GPIOC->ODR &=  ~(char_nib); // clears data
	GPIOC->ODR |=  (char_nib);  // sets  4 bitcharacter data in ODR

	GPIOC->ODR |=  (E_bit); //puts the ones corresponding to the data inputed in
the ODR at bits 0-4

	delay_us(1000); //this one we need

	GPIOC->ODR &=  ~(E_bit); //sets Enable pin to zero

	delay_us(1000);

	return;
}

void LCD_write_char (uint8_t char_byte)
{
//	GPIOC->ODR &=  ~(RW_bit | RS_bit);  //maybe take out?
	uint8_t ls_nib= 0x0F & char_byte; // splits char data into two 4 bit data to
send data 4 bits at a time

	uint8_t ms_nib= 0xF0 & char_byte;

	ms_nib= ms_nib>>4;

//	delay_us(1520);

	LCD_write_char_nib (ms_nib);

	delay_us(2000);   //this we need

	LCD_write_char_nib (ls_nib);

   //delay_us(1520);
}

void LCD_write_string(char *string_in)
{
	for(int i = 0; string_in[i]!='\0'; i++)
		LCD_write_char(string_in[i]);
}
```

```
void  delay_us(const uint16_t time_us)
{
    // set the counts for the specified delay
    SysTick->LOAD = (uint32_t)((time_us * SystemCoreClock / 1000000) - 1);
    SysTick->VAL = 0;                                          // clear the timer count
    SysTick->CTRL &= ~(SysTick_CTRL_COUNTFLAG_Msk);            // clear the count flag
    while (!(SysTick->CTRL & SysTick_CTRL_COUNTFLAG_Msk)); // wait for the flag
}
```