

ECE 355 Lab Section B01

Final Report

Due Date: November 29, 2021

Group Members: Joel Meuleman (V00919945)

Michael Pillon (V00973605)

Marking

Problem Description/Specs	/ 5
Design/Solution	/ 15
Testing/Results	/ 10
Discussion	/ 15
Code Design/Documentation	/ 15
Total	/ 60

Table of Contents

Project Specifications	3
Objective	3
Design Specifications	3
Design	6
Push Button / LEDs / Interrupts	6
Configuring the Push Button	6
Configuring the LEDs	6
Configuring the Interrupts	7
Interrupt Handling Routines	7
Frequency Measuring with TIM2	8
Configuring TIM2	8
Measuring Frequency	8
LCD	9
Configuring the STM32F0 for LCD Interfacing	9
Configuring the LCD	9
Writing Data to the LCD	11
Analog to Digital Converter	12
Configuring the ADC	12
Reading the ADC	14
Digital to Analog Converter	14
Configuring the DAC	14
Writing and Reading the DAC	15
Testing and Results	16
Push button, LEDs, and interrupts	16
Resistances, the ADC, and the DAC	16
Frequency Measurements	17
555 Timer vs Frequency Generator	18
Discussion	20
Design	20
Assumptions	20
Shortcomings	20
Lessons Learned	22
Appendix A - Main.c	23

Appendix B - adc.h and adc.c	26
Appendix C - dac.h and dac.c	27
Appendix D - timer.h and timer.c	28
Appendix E - lcd.h and lcd.c	29
Appendix F - Multivibrator Circuit Schematic [4]	31
References	32

Project Specifications

Objective

The objective of this project is to develop an embedded system based on the STM32F0 Discovery board which has the ability to measure the frequency of a PWM signal from either a function generator or a 555 timer. The purpose of this project is to gain an understanding for how to configure bare-metal microcontrollers and interface them with a variety of electronics.

Design Specifications

The system should be able to switch between measuring a PWM signal from a function generator and one generated by a 555 timer using the NE555 IC. A 4N35 optocoupler will be used to bias the 555 timer and will interface with the STM32F0 through a DAC output pin. The output from the STM320 into the optocoupler will be controlled by a potentiometer and an ADC pin on the STM320. Switching between the 2 signals will be done using a pushbutton and the interrupt functionality of the STM32F0. The currently active mode will be visible on the board based on whether the green or blue LED is on and the LEDs will swap when a pushbutton is pressed and the active mode is changed. The measured resistance value as well as the measured frequency of the active mode will be displayed on an LCD. A full functional block diagram of this system is shown in *Figure 1*.

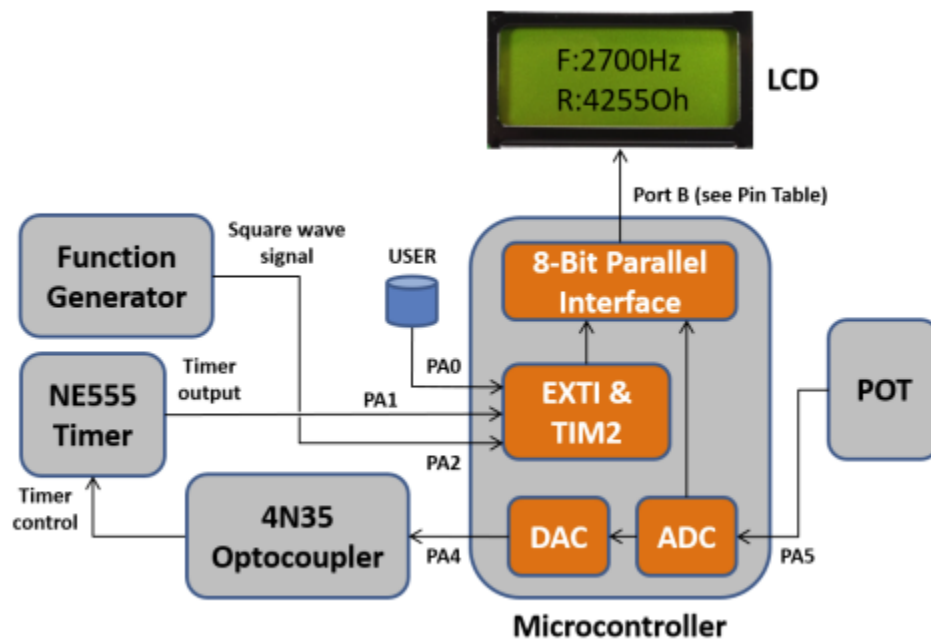


Figure 1: Overall System Diagram [1]

To simplify development and testing of this system, an independent program will be used to simulate and control the system in *Figure 1*. The application called “555 Timer Monitor” interfaces with the STM32F0 Discovery board and provides a virtual controller for the frequency generator, a monitor of the LCD pins, a simulation of the board, and a virtual LCD. The application is a standin for testing and if the discovery board is wired as per the table in *Figure 2*, a fully hardware based configuration should operate identically.

STM32F0	SIGNAL	DIRECTION
PA0	USER PUSH BUTTON	INPUT
PC8	BLUE LED	OUTPUT
PC9	GREEN LED	OUTPUT
PA1	555 TIMER	INPUT
PA2	FUNCTION GENERATOR	INPUT
PA4	DAC	OUTPUT (Analog)
PA5	ADC	INPUT (Analog)
PB4	ENB (LCD Handshaking: “Enable”)	OUTPUT
PB5	RS (0 = COMMAND, 1 = DATA)	OUTPUT
PB6	R/W (0 = WRITE, 1 = READ)	OUTPUT
PB7	DONE (LCD Handshaking: “Done”)	INPUT
PB8	D0	OUTPUT
PB9	D1	OUTPUT
PB10	D2	OUTPUT
PB11	D3	OUTPUT
PB12	D4	OUTPUT
PB13	D5	OUTPUT
PB14	D6	OUTPUT
PB15	D7	OUTPUT

Figure 2: Pin Assignments and functionality [1]

The software will be designed around the external interrupt functionality or EXTI of the STM32F0. After initializing the hardware, the main function of the program will consist of an empty infinite loop which exists to keep the program running while waiting for interrupt signals. 3 devices will be configured to create interrupts: a pushbutton, a 555 timer, and a function generator.

The pushbutton and 555 timer will be configured using the native *EXTI0_1_IRQHandler()* function which is a part of the Common Microcontroller Software

Interface Standard or CMSIS. When the STM32F0 detects an interrupt on EXTI0 or EXTI1, it enters the IRQ handler and the program checks which device triggered the interrupt. If the pushbutton is pressed, the system switches the active mode and resets necessary flags and registers. If the 555 timer triggers the interrupt, the system will call the *calculateFrequencyFromTIM2()* subroutine. Similarly, the function generator will be configured on EXTI2 and will use the *EXTI2_3_IRQHandler()* function to follow the same flow as the 555 timer.

A more in depth description of the software configuration and methods is given in the design section of this manual and a block diagram describing the overall program architecture is available in *Figure 3*.

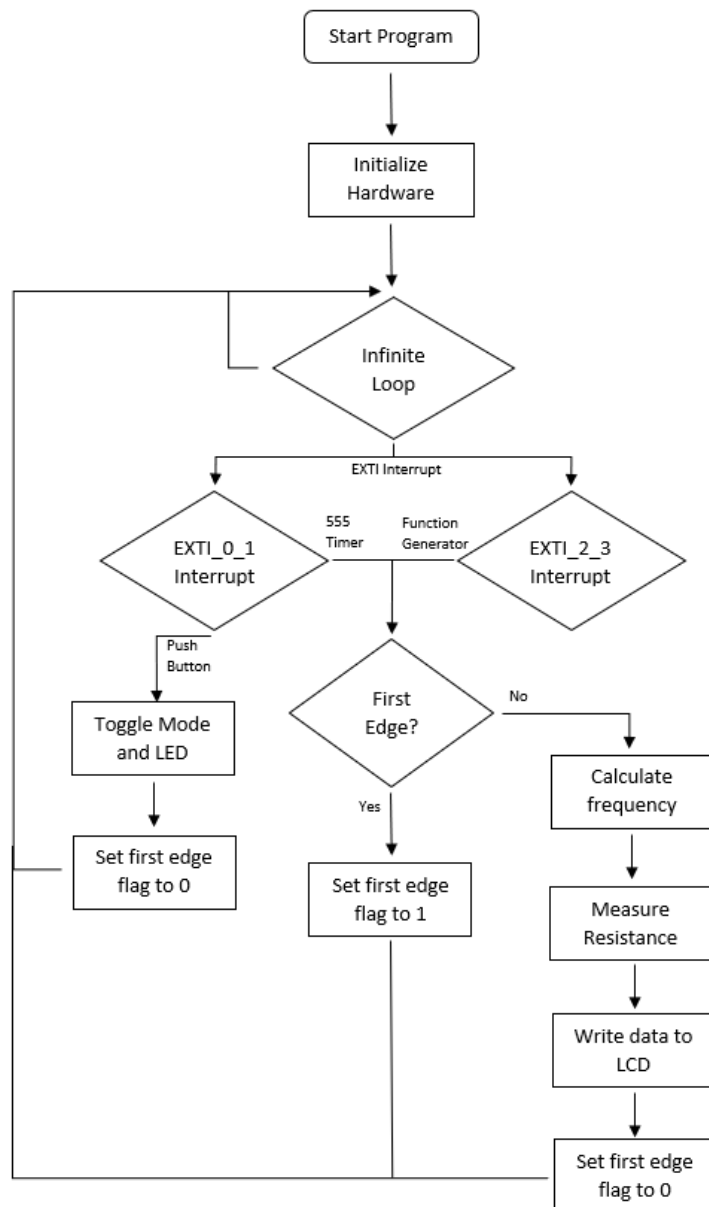


Figure 3 - Program flow block diagram describing the software of the system

Design

From the start, it was recognized that identifying and separating the project into its functional components would be the best way to proceed. By modularizing the project, code could be worked on independently before being integrated into the system and provides reusability for future projects. This modularization of the C code was done using header files. *Figure 1* shows the functional components of the project.

The design was divided into the following modules:

- The input push button that toggles the active external interrupt and the corresponding indicator LEDs
- The External Interrupt Controller and Timer to measure input frequency
- The parallel port I/O to write measured data to the LCD
- The Analog to Digital Converter to read the potentiometer resistance.
- The Digital to Analog Converter to drive the optocoupler to set the frequency of the 555 timer.

Push Button / LEDs / Interrupts

The core of this project is based around external interrupts by the push button, frequency generator, and 555 timer. The current mode of operation is stored in the global variable, *activeMode*, and uses the pin defines *MODE_FUNCTION_GENERATOR* and *MODE_555_TIMER* for its states and describes whether the PWM frequency will be calculated for either the function generator or 555 timer.

Configuring the Push Button

The push button was configured to be connected to pin PA2 and the configuration was handled by the *GPIOA_init()* function. First, the clock signal was enabled for port A by turning on the bit stored in *RCC_AHBENR_GPIOAEN* in the *RCC->AHBENR* register. Next, PA2 is configured as an input by ensuring the *GPIO_MODER_MODER2* bits of the *GPIOA->MODER* register are turned off. Lastly, the pushbutton is configured with internal no pull up or pull down resistors by turning off the *GPIO_PUPDR_PUPDR2* bits of the *GPIOA->PUPDR* register.

Configuring the LEDs

The LEDs were configured on pins PC8 and PC9 on the STM32F0 Discovery board and consist of a blue and a green surface mount resistor. Both LEDs are configured with physical resistors as active high, in other words the LEDs turn on when a 1 is written to the corresponding pin.

The first step in configuring the LEDs is to ensure the clock for port C is turned on by enabling the *RCC_AHBENR_GPIOCEN* bit of the *RCC->AHBENR* register. Next, the 2 pins are set as outputs by setting the corresponding *GPIO_MODER_MODERx_0* bit of the *GPIOC->MODER* register. The output type for both LEDs was set to push-pull through the *GPIOC->OTYPER* register, the pins were set to high-speed mode through the *GPIOC->OSPEEDR* register, and the pins were configured with no pull-up or pull-down resistors through the *GPIOC->PUPDR* register. Further details about the LED configuration can be seen in Appendix A or in the STM reference manual [2].

Configuring the Interrupts

To map a pin to an EXTI line, the appropriate bits of the *SYSCFG->EXTICR* register. The EXTI0 line was mapped to PA0 for the pushbutton, the EXTI1 line was mapped to PA1 for the 555 timer, and the EXTI2 line was mapped to PA2 for the function generator. All the interrupts were set to a rising-edge trigger by turning on the *EXTI_RTSTR_TRx* bit of the *EXTI->RTSR* register.

To initialize the program, the *EXTI->IMR* register is used to enable the push button interrupts and the function generator through the constants mapped to the *EXTI_IMR_MR0* and *EXTI_IMR_MR2* bits. *EXTI_CONTROL_BIT_x* constants are used to make the code more readable. The *NVIC_SetPriority()* function is called to set *EXTI0_1_IRQn* to a priority of 0, and *EXTI2_3_IRQn* to a priority of 1 so that the pushbutton will always have the highest priority regardless of which signal is being measured and triggering interrupts. The *EXTI0_1_IRQn* and *EXTI2_3_IRQn* lines are then enabled using the *NVIC_EnableIRQ()* function.

Interrupt Handling Routines

The *EXTI0_1_IRQHandler()* and *EXTI2_3_IRQHandler()* functions are called whenever an interrupt is called on either EXTI0/EXTI1 or EXTI2/EXTI3 respectively. Since EXTI0 and EXTI1 are both used in this system, the first step of the subroutine is to identify which line triggered the interrupt and this is done by checking which bits are set in the *EXTI->PR*. The *EXTI_PR_PRx* bits are stored in global constants as *PENDING_REGISTER_x* with the appropriate hardware name that triggered the interrupt. If a pending register bit is found to be set, after handling the interrupt the pending register is reset to 0 by writing a 1 to the *PENDING_REGISTER_x* bit to the *EXTI_PR_PRx* register.

The *PENDING_REGISTER_PUSH_BUTTON* bit is checked first to ensure that if the push button is pressed, it takes priority over the 555 timer in the *EXTI0_1_IRQHandler()* function. If the push button had triggered the interrupt, the program waits for the button to be released before moving on. The timer is disabled and the first edge flag is reset to 0 to ensure that after the mode of the program is changed, the first reading is accurate. The currently active LED is turned off using the *GPIOC->BRR* register, the *activeMode* register is switched to the new mode, and the appropriate LED for the new mode is turned on using the *GPIOC->BSRR* register.

The *EXTI->IMR* register is then used to enable interrupts on the pushbutton and the new mode while disabling interrupts for the previously active mode.

When an interrupt is detected in the *EXTI_PR_PRx* register for either the 555 timer or the function generator, the *calculateFrequencyFromTIM2()* function is called and handles the appropriate signal. This function uses the *activeMode* register to measure frequency on the appropriate pin.

Frequency Measuring with TIM2

To measure the frequency of input waveforms, general purpose timer TIM2 was used. By measuring the time between rising edges, frequency can be calculated. When a rising pulse edge was detected, TIM2 was started. When the next rising pulse edge was detected, the value TIM2 had counted to was recorded, and the frequency conversion made.

Configuring TIM2

The configuration of TIM2 involved many registers. First, the clock for TIM2 was enabled. Second, TIM2 was configured for count up mode by setting the appropriate *Center-Aligned Mode Selection* bits in the *TIM2 Control Register 1*. Third, the *Timer2 Event Generation Register* was configured to correctly reinitialize itself by setting the *Update Generation* bit. Fourth, the priority of the TIM2 interrupt was configured as highest priority in the Nested Vector Interrupt Controller (NVIC) by calling the corresponding function. Also in the NVIC, the TIM2 interrupt was enabled. Lastly, interrupts were enabled by setting the *Interrupt Enable* bit in the *TIM2 DIER* register.

Measuring Frequency

There are 2 states of the system when measuring frequency: the timer is not running and the timer is running. When the *calculateFrequencyFromTIM2()* function is called, the *firstEdgeFlag* is checked to determine which state the system is in. If the *firstEdgeFlag* is equal to 0, it is the first time the function is being called so the timer count is reset to 0, the *firstEdgeFlag* is set to 1, and the timer is enabled. When the system returns to the *calculateFrequencyFromTIM2()* method, the timer is immediately stopped to maximize accuracy and the *TIM2->CNT* register is read. This register is incremented once for every rising edge of the system clock which is available in the *SystemCoreClock* global variable. The count divided by the clock frequency gives the measured frequency of the signal. This value is then passed to the *writeToLCD()* function which measures the potentiometer and outputs both data points to the LCD as formatted strings. The *firstEdgeFlag* is then reset to 0.

LCD

Measured data was displayed using a simulated LCD modelled after the Hitachi HD44780. The simulated LCD was used to display measured frequency from a selectable source in addition to measured potentiometer resistance. The GPIOs of port B were used to interface the LCD to the microcontroller.

Configuring the STM32F0 for LCD Interfacing

Before the LCD can be directly configured, the interfacing GPIOs must be set up. Port B pins 4-6 and 8-15 were configured as output pins by setting the corresponding bits in the port B *Port Mode Register*. Pins 4-6 were designated as control pins, while Pins 8-15 were designated as data pins. These pin designations are seen in the LCD.h header file in Appendix E.

The output mode bits of the *Port Mode Register* were set for each pin using the *GPIO_MODER_MODERx_0* define as found in the STM32F0 header file. All port B I/O pins were configured for the highest speed by setting all bits in the *GPIO Port Output Speed Register*.

Configuring the LCD

To send commands and data to the LCD, a procedure is defined in the HD44780 datasheet and LCD interfacing slides provided in class[4][6]. This procedure is abstracted into a function `lcd_write(uint16_t data)` to make for simple writing of a data byte with relevant control bits to the LCD. To simplify the sending of commands, GPIO pins corresponding to the data and control bits of the LCD were defined as seen in Appendix E.

1. The LCD data bits DB0-DB7 are set with the data to be sent.
2. The LCD enable bit EN is set and the LCD reads the data lines.
3. The LCD busy/handshake bit HANDSHAKE is polled until it is asserted, meaning the LCD has begun instruction execution[6].
4. The LCD enable bit EN is deasserted.
5. The LCD busy/handshake bit HANDSHAKE is polled until it is deasserted, meaning the LCD has completed the last instruction[6].
6. The data bits are reset

```

void lcd_write(uint16_t data)
{
    GPIOB->BSRR |= data; /* Write data to pins */
    GPIOB->BSRR |= EN;
    while(!(GPIOB->IDR & HANDSHAKE));
    GPIOB->BSRR |= EN << 16;
    while((GPIOB->IDR & HANDSHAKE));
    GPIOB->BSRR |= data << 16;
}

```

Figure 4: LCD Write Function

Toggling of the command and data bits are done using the *GPIO Bit Set/Reset Register*. A write to bits in the range [15:0] sets the corresponding output pin high. A write to bits in the range [31:16] sets the corresponding bit in the range [15:0] low. Handshaking is performed by polling input pin 7 of port B to check if it is set (instruction execution in progress) or unset (no instruction in progress). With a function to transfer data to the LCD, commands can be given and configuration can proceed.

To display data correctly, several commands are given to the LCD. All commands are given as some combination of set data bits, as defined in the HT44780U datasheet. First, the LCD was configured for 8-bit mode (as it is interfaced by 8 GPIOs), both lines of the LCD were enabled, and a character size of 5x8 pixels was selected. The data bits corresponding to this command were DB3, DB4, and DB5. Next, the display was enabled, the cursor disabled, and blink-mode disabled. The data bits corresponding to this command were DB3 and DB2. Next, auto-increment was enabled, and the direction of the increment was set (this makes accessing DDRAM more convenient when attempting to display characters). The data bits corresponding to this command were DB1 and DB2. Finally, the display was cleared by setting DB0. Summarized, the display is configured for: 8-bit interfacing, two-line display, no cursor nor character blinking, and auto increment DDRAM address when writing successive characters.

Instruction	RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
Clear display	0	0	0	0	0	0	0	0	0	1
Return home	0	0	0	0	0	0	0	0	1	—
Entry mode set	0	0	0	0	0	0	0	1	I/D	S
Display on/off control	0	0	0	0	0	0	1	D	C	B
Cursor or display shift	0	0	0	0	0	1	S/C	R/L	—	—
Function set	0	0	0	0	1	DL	N	F	—	—
Set CGRAM address	0	0	0	1	ACG	ACG	ACG	ACG	ACG	ACG
Set DDRAM address	0	0	1	ADD	ADD	ADD	ADD	ADD	ADD	ADD
Read busy flag & address	0	1	BF	AC	AC	AC	AC	AC	AC	AC

Figure 5: Table of LCD Commands[6]

Writing Data to the LCD

Once the LCD display formatting was complete, data could be written to it. The LCD displays data by receiving a given address from its data bus, and displaying the character stored in internal DDRAM at that address. Conveniently, character addresses in DDRAM are the same as the ASCII codes of those same characters. This means that to display any ASCII character all that needs to be done is to set the corresponding data bus pins to the ASCII code of that character. As pins 8-15 (DB0-DB7) were designated as the data bus pins for interfacing the LCD, to put an ASCII character on the bus was a simple bit shift when sending the 16-bit data integer to `lcd_write(uint16_t data)`. In addition to the data bits, the data indicator bit (RS) must also be set to indicate a data write rather than a command. The line of the LCD to write also had to be selected by setting DB7 for line 1 and DB6 and DB7 for line 2.

Being able to efficiently write changing strings of data to the display was an important aspect of this project. To accomplish this, two functions were written, the one used depending on the line of the display to be written. These functions, `lcd_display_line1(char *data)` and `lcd_display_line2(char *data)` take a reference to a string, the data to be displayed, and iterate over the length of the string, calling `lcd_write(uint16_t data)` on each character iterated over. The remaining spaces not filled by characters from the input data are written with blank space. Filling the unused spaces with empty spaces proved to be a preferable solution to clearing the entire display upon one new reading.

```
void lcd_display_line1(char *data)
{
    /* Select line 1 */
    lcd_write(LINE_1);
    unsigned int i = 0;
    /* Write data */
    for(; i < strlen(data); i++){
        lcd_write(RS | data[i]<<8);
    }
    /* Ensure the rest of the line is empty */
    for(; i < LCD_LINE_LENGTH; i++){
        lcd_write(RS | ' ' << 8);
    }
}
```

Figure 6: `lcd_display_line1(char *data)`

Analog to Digital Converter

The analog to digital converter (ADC) of the STM32F0 was used to measure the resistance of an external user-set potentiometer. The voltage measured by the ADC was then mirrored by the digital to analog converter (DAC) which in turn drove the 555 timer.

Configuring the ADC

Configuration of pin A5 for analog to digital conversion is identified as two separate tasks: configuration of the I/O port as an analog input, as well as configuration of relevant ADC registers. Beginning with the I/O port configuration, the peripheral clock is enabled on Port A. Next, the Port A *Port Mode Register* register is configured. To use GPIO pin A5 as an analog input, the corresponding *Analog Mode* bits must be set in said register. For pin 5, these bits in the

Port Mode Register register are set by *GPIO_MODER_MODER5*, as defined in the STM32F0 header file.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MODER15[1:0]		MODER14[1:0]		MODER13[1:0]		MODER12[1:0]		MODER11[1:0]		MODER10[1:0]		MODER9[1:0]		MODER8[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MODER7[1:0]		MODER6[1:0]		MODER5[1:0]		MODER4[1:0]		MODER3[1:0]		MODER2[1:0]		MODER1[1:0]		MODER0[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 2y+1:2y **MODERy[1:0]**: Port x configuration bits (y = 0..15)

These bits are written by software to configure the I/O mode.

00: Input mode (reset state)

01: General purpose output mode

10: Alternate function mode

11: Analog mode

Figure 7: GPIO Mode Set Register

Next, configuration of the main ADC registers is considered. For this project, maximum accuracy in the reading of the potentiometer is desired. Accuracy in the ADC reading is directly proportional to the selected resolution, so the highest resolution of the STM32F0 was selected, 12 bits. 12 bits is the default resolution set in *ADC Configuration Register 1*. Additional bits set in the prior register are the *Overrun Management Mode* bit which allows old data to be overwritten if new data is available, and the *Continuous Conversion Mode* bit which forces ADC conversion to run continuously during program execution.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	AWDCH[4:0]					Res.	Res.	AWDEN	AWDSGL	Res.	Res.	Res.	Res.	Res.	DISCEN
	rw	rw	rw	rw	rw			rw	rw						rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AUTOFF	WAIT	CONT	OVRMOD	EXTEN[1:0]		Res.	EXTSEL[2:0]			ALIGN	RES[1:0]		SCAND IR	DMAC FG	DMAEN
rw	rw	rw	rw	rw			rw			rw	rw		rw	rw	rw

Figure 8: ADC Configuration Register

Relevant to the resolution of the ADC, the *Sample Rate* register is configured for maximum sampling time (239.5 clock cycles) [2]. A lengthy sample time is desirable for an accurate reading as it gives the internal capacitor time to charge to and hold the input voltage [3]. Then, the *Channel 5* bit (as pin A5 is being read) of the *ADC Channel Selection Register* is set, which configures the channel for conversion. Finally, after configuration of all other registers are complete, the *ADC Enable* bit is set in the *ADC Control Register*, as well as the *Analog to Digital Conversion Start* bit starting conversion. The *Analog to Digital Conversion Start* bit must be set only once configuration is complete as specified in the user manual.

Reading the ADC

Reading raw data from the ADC is done using a polling approach. To check for a new valid reading, the *End of Conversion Flag* of the *ADC Interrupt and Status Register* is polled. The *End of Conversion Flag* bit is set when a conversion is finished and new data is available in the *ADC Data Register*[2]. This flag is automatically reset upon reading the data register contents. The data register contents are converted to a voltage reading using *Equation 1* [2].

$$V_{\text{CHANNEL}x} = \frac{V_{\text{DDA}}}{\text{FULL_SCALE}} \times \text{ADC_DATA}_x$$

Equation 1: Channel Data to Channel Voltage

The reference voltage, V_{DDA} , of *Equation 1* is equal to ~3.3V as specified in the STM32F0 datasheet and user manual. The scale factor of *Equation 1* is equal to the configured resolution of the ADC, $2^{12} - 1$. To read the resistance set on the external potentiometer, a similar equation is used, where the resistance is calculated as the ratio of ADC input data to maximum resolution as seen in *Equation 2*.

$$\text{Potentiometer Value} = \frac{\text{Maximum Potentiometer Resistance}}{\text{Full Scale Resolution}} * \text{ADC Data}$$

Equation 2: Channel Data to Resistance Reading

The *Maximum Potentiometer Resistance* of *Equation 2* is equal to the maximum value of the potentiometer, 5kΩ.

Digital to Analog Converter

The Digital to Analog Converter (DAC) of the STM32F0 was used to control the frequency output of the 555 timer by providing a voltage varying from 0V to 3.3V as read from the ADC.

Configuring the DAC

Configuration of pin A4 for digital to analog conversion is identified as two separate tasks: configuration of the I/O port as an analog output, as well as configuration of relevant DAC registers. Beginning with I/O port configuration, the peripheral clock is enabled on port A; the DAC clock is also enabled. Pin A4 is set to *Analog Mode* in the *Port A Port Mode Register* register by setting the corresponding bits defined by *GPIO_MODER_MODER4* in the STM32F0 header file. Next, the DAC control registers are configured. The only configuration needed is to enable the DAC, by setting the *DAC Channel1 Enable* bit in the *DAC Control Register*.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	DMAU DRIE2	DMA EN2	MAMP2[3:0]				WAVE2[1:0]		TSEL2[2:0]			TEN2	BOFF2	EN2
		rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	DMAU DRIE1	DMA EN1	MAMP1[3:0]				WAVE1[1:0]		TSEL1[2:0]			TEN1	BOFF1	EN1
		rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW

Figure 9: DAC Control Register

Writing and Reading the DAC

The output voltage of the DAC is set by writing to an intermediate register, whose value is then shifted to the output register after a clock cycle. First, the *ADC Data Register* is read; this voltage reading is the voltage to be output from the DAC. Second, the *ADC Data Register* contents are written to the DAC intermediate data holding register. For simplicity, *DAC Channel 1 12-bit Right-Aligned Data Holding Register* is selected for writing to eliminate any bit shifting and scaling that would be needed if a 8 or 12 bit left-aligned register were selected. To check that the correct voltage is being output, the *DAC Channel 1 Data Output Register* is read and the data made readable by Equation 3. V_{DDA} is the same reference voltage as referenced in Equation 1; 4095 is the resolution of the DAC which is the same as the ADC.

$$\text{DACoutput} = V_{DDA} \times \frac{\text{DOR}}{4095}$$

Equation 3: Converting DAC Output Data to a Voltage Reading [2]

Testing and Results

Push button, LEDs, and interrupts

Testing for this part of the project was relatively straightforward as a button press should toggle the system between reading the function generator and reading the 555 timer while toggling the LED according to the specific mode of operation. The user button in the 555 timer application was pressed repeatedly and the status LED could be observed to be changing in both the application as well as on the physical board. When the user button was pressed it was also observed that the frequency measurements on the LCD would switch between the value of the function generator to the value of the 555 timer and vice versa. As a part of testing in the early stages of development, *trace_printf()* statements were used to indicate which mode was switched to and these print statements were left in the final version of the code.

Resistances, the ADC, and the DAC

The potentiometer readings were tested by adjusting the resistance in the 555 timer application from the minimum value of 0Ω to the maximum value of $5k\Omega$. These are the theoretical limits of the potentiometer however both are subject to practical limitations. The 0Ω minimum value is limited by any resistances in the terminals and traces however should likely be very close to 0Ω . The maximum reading of $5k\Omega$ also would be increased by terminal and trace resistances however is also subject to variability during the manufacturing process. There was no way to confirm this variability since the potentiometer was already integrated into the system.

When the ADC read values from the potentiometer the actual resistance limits were roughly 4935Ω and 68Ω as seen in *Figure 10* and *Figure 11*.

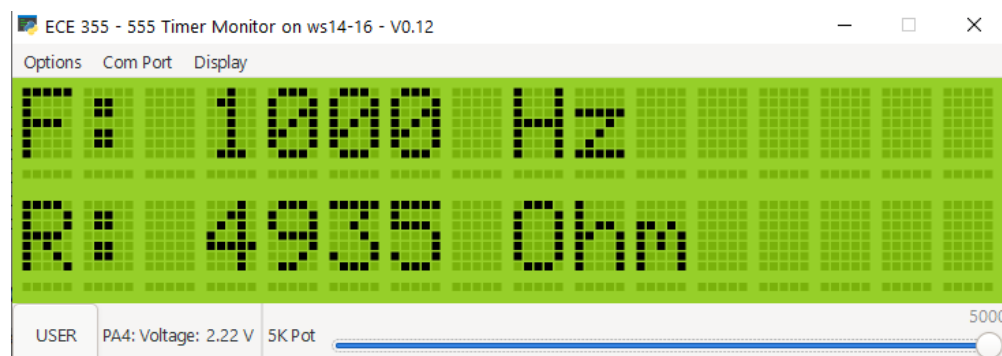


Figure 10 - Maximum measured resistance value

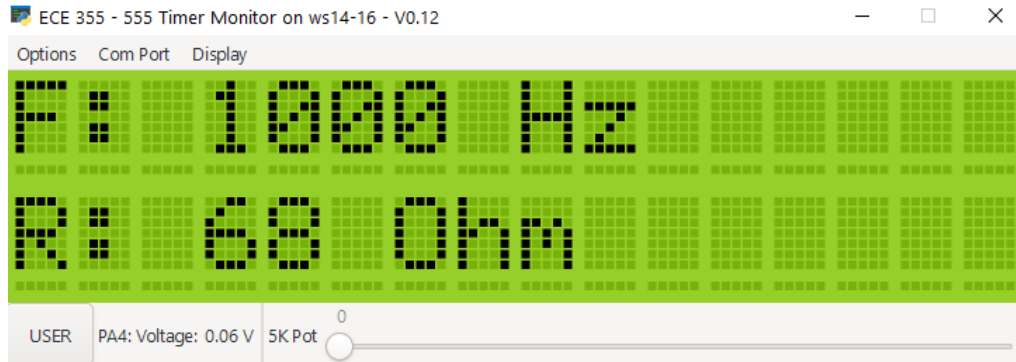


Figure 11 - Minimum measured resistance value

The minimum value resistance is attributed to additional resistances in the circuit as well as some practical limitations of the ADC. The resolution of resistance stems from the 12-bit ADC resolution and is equal to $\frac{5000\Omega}{2^{12}} = 1.22\Omega$ and is the minimum reading between 2 measurements. The accuracy of the system was a larger factor and the resistance measurements on the LCD could be seen varying by roughly $\pm 50\Omega$ which corresponds closely to the observed minimum and maximum readings.

During development, the `dac_read()` function was used to test the DAC and returns the value that is currently being output by the DAC. After the ADC value or any 12-bit value was written to the DAC, the read function was used to verify that the written value was being stored.

Frequency Measurements

Testing for this project consisted of measuring frequencies from 2 separate sources, a function generator and a 555 timer. These sources used separate input pins however both of them used TIM2 for calculation of frequency measurements. TIM2 was configured with no prescaler so the maximum theoretical frequency achievable was the period related to 1 clock pulse or 48MHz. Practically however the maximum readable frequency was much lower at just over 500KHz, see Figure 12.

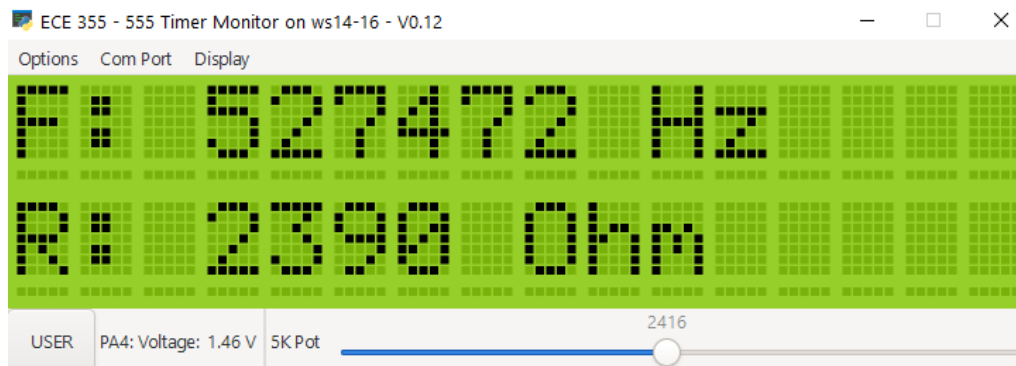


Figure 12 - Maximum practical frequency reading

This result was achieved by setting the frequency generator to 10MHz and observing the output on the LCD. Since the frequency was well above the practical limit of the system, the observed frequency was a result of the system responding as quickly as possible and was consistent as frequency was increased to any value above this practical limit.

This practical limit exists for a number of reasons but is mostly due to the overhead required to compute frequency. The interrupt routine as well as all the time required to process the code takes up time meaning that the interrupt routines, starting the timer, and stopping the timer are all accounted for when measuring frequency. Furthermore, all testing was done while using a debugger which has extra overhead compared to programming the STM Discovery board and having it run independently. These effects were noticeable in the practical limit of the frequency readings but also showed up for significantly lower frequency readings. *Figure 13* shows these effects as a 10kHz signal being generated by the function generator displays a value of 10012Hz and this discrepancy can be attributed to the same computation times which limit the maximum frequency readings. As frequencies are increased, these effects have a greater effect and at low frequencies these effects are negligible and don't show up for frequencies of a few kHz or lower.

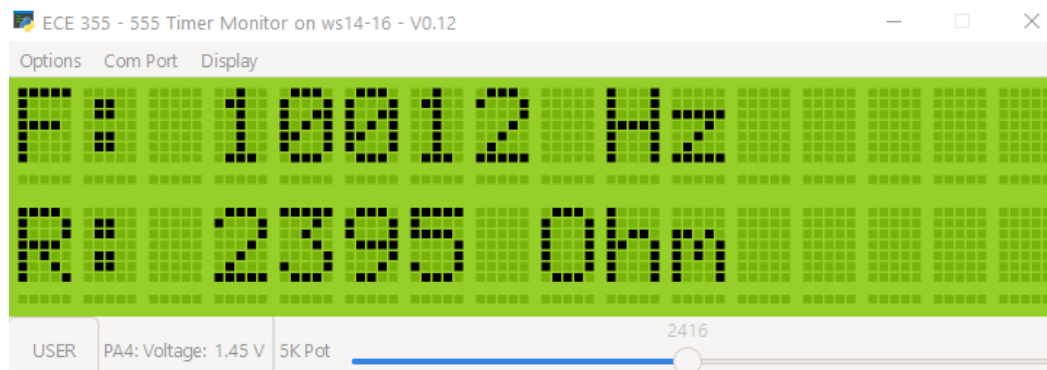


Figure 13 - A 10kHz frequency reading showing 12Hz of error due to computation time

The theoretical minimum for our frequency measurements was based on the resolution of TIM2. There was no prescaler value set and TIM32 is a 32-bit counter meaning the theoretical minimum frequency was $2^{32} = 233 \times 10^{-12} \text{ Hz}$ which gives a period of roughly 90 seconds.

555 Timer vs Frequency Generator

The theoretical and practical limitations directly applied to the frequency generator which has a much wider range of frequencies it could produce, however the 555 timer has a significantly smaller range of operation. The 555 timer works based on the configuration of external resistances and capacitances which give reference voltages at some of its pins. This project had limited flexibility in the range of these voltages since the DAC and optocoupler were limited to operating between 0V and 3.3V. This configuration gave a practical limit to the

frequency output of the 555 timer to between roughly 1kHz and 1.5kHz. There was also a limited effective range for the potentiometer as only about one third of the resistance range was observed to have any effect on the frequency produced by the 555 timer. These limits are available in *Figure 14* and *Figure 15*.

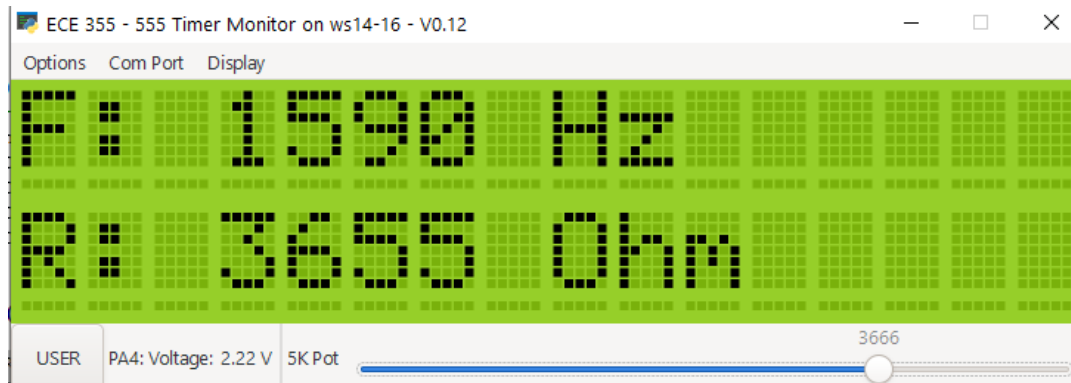


Figure 14 - Maximum frequency of the 555 timer achieved at the minimum resistance

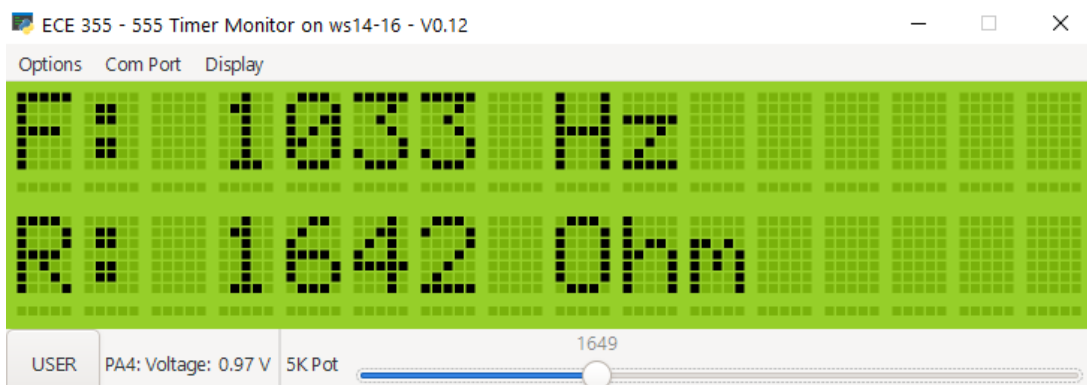


Figure 15 - Minimum frequency of the 555 timer achieved at the maximum resistance

When varying the potentiometer from 0 Ω to 5k Ω , the 555 timer only varied from roughly the 1.6k Ω to 3.6k Ω range but remained relatively constant when adjusting the potentiometer outside of this range. The constant variations in the ADC readings were also observed as similar variations in the 555 timer frequency of less than 5% of the measured frequency.

Discussion

Design

Our design was able to meet all specified project requirements: external frequencies were accurately measured and able to be toggled between, the potentiometer value was accurately read at all values (excluding 0Ω) with a relative error of less than 1%, modifying the DAC output voltage successfully varied the frequency output by the 555 timer, and the LCD correctly displayed the measured data with seamless transitions between values.

Assumptions

A first assumption is that the PWM input frequency falls within the accurately measurable frequency range of the microcontroller, which was determined to be 1Hz to ~500 kHz. A second assumption was that the internal reference voltage, $VDDA$, was 3.3V. This value should be very close to correct and is spec for an operating temperature of 30°C [2]. As this reference voltage is used in calculations of input voltage from the ADC and output voltage of the DAC, it is possible it introduced a small source of error in input and output voltage readings. The precise value of $VDDA$ can be calculated from the $VREFINT_CAL$ and $VREFINT_DATA$ registers but was deemed unnecessary. Per the datasheet, the precise value of $VDDA$ should only differ from 3.3V by a factor of millivolts[2].

Shortcomings

The most notable shortcoming was the small bandwidth of frequencies output from the 555 timer, 1kHz to 1.5kHz. The frequency was observed to change over a DAC output voltage of approximately 1V to 2.2V. Without knowing the exact circuit values used in the construction of the 555 multivibrator circuit, its performance is qualitatively analyzed. The frequency of the waveform output from the 555 is dependent on the rate of charging of the capacitor $C1$, seen in Appendix F. The current output from the optocoupler, the current that charges $C1$, varies with the LED current (forward current) of the optocoupler. The LED current is dependent on the supplied voltage from the DAC, and its magnitude in relation to its forward voltage drop. The collector-emitter current (output current) of the optocoupler's phototransistor is dependent on the forward current of the LED, which is approximately constant above a certain voltage as seen in *Figure 17*. This would give a maximum to the rate $C1$ is charged at, and subsequently a maximum to the frequency of the output waveform.

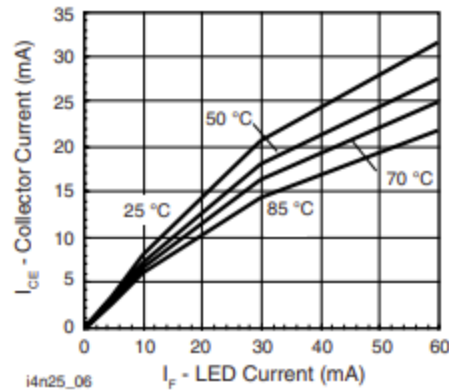


Fig. 6 - Collector Emitter Current vs. Temperature and LED Current

Figure 16: Optocoupler Output Current Dependence on LED Current[4]

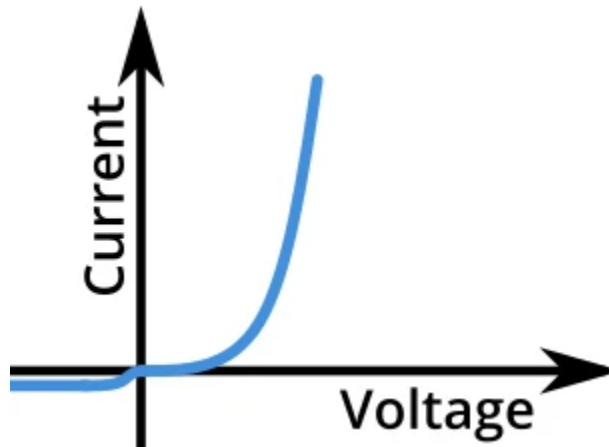


Figure 17: Diode IV Graph

Below 1V, no frequency change was observed. This was to be expected, as the forward voltage of the optocoupler LED is listed near this value, and no current will flow through the LED below this value[4].

The other shortcoming of the system was in its frequency measuring capability. The microcontroller was not able to measure frequencies over the entire calculated range. This is mainly due to time lost processing instructions as was discussed in **Testing: Frequency Measurements**.

Lessons Learned

The construction of this system was an awesome learning experience. Valuable experience was gained in embedded programming. Design of this system provided insight into the architecture of the STM32, an industry-standard microcontroller. This experience will surely prove useful in further embedded programming projects, whether that be on the STM32 or another microcontroller.

Appendix A - Main.c

```
1 //
2 // This file is part of the GNU ARM Eclipse distribution.
3 // Copyright (c) 2014 Liviu Ionescu.
4 //
5
6 /*
7  * School: University of Victoria, Canada.
8  * Course: ECE 355 "Microprocessor-Based Systems".
9  * ECE 355 lab project.
10  * By:
11  *   Joel Meuleman
12  *   Michael Pillon
13  *
14  * See "system/include/cmsis/stm32f0xx.h" for register/bit definitions.
15  * See "system/src/cmsis/vectors_stm32f0xx.c" for handler declarations.
16  */
17
18 #include <stdio.h>
19 #include <stdlib.h>
20 #include "diag/Trace.h"
21 #include "cmsis/cmsis_device.h"
22
23 #include "lcd.h"
24 #include "timer.h"
25 #include "adc.h"
26 #include "dac.h"
27
28 /* Sample pragmas to cope with warnings. Please note the related line at
29  * the end of this function, used to pop the compiler diagnostics status. */
30 #pragma GCC diagnostic push
31 #pragma GCC diagnostic ignored "-Wunused-parameter"
32 #pragma GCC diagnostic ignored "-Wmissing-declarations"
33 #pragma GCC diagnostic ignored "-Wreturn-type"
34
35 /* Interrupt mode constants */
36 #define EXTI_CONTROL_BIT_PUSH_BUTTON      EXTI_IMR_MR0
37 #define EXTI_CONTROL_BIT_555_TIMER        EXTI_IMR_MR1
38 #define EXTI_CONTROL_BIT_FUNCTION_GENERATOR EXTI_IMR_MR2
39 #define PENDING_REGISTER_PUSH_BUTTON      EXTI_PR_PR0
40 #define PENDING_REGISTER_555_TIMER        EXTI_PR_PR1
41 #define PENDING_REGISTER_FUNCTION_GENERATOR EXTI_PR_PR2
42
43 /* Hardware initialization */
44 void GPIOA_init(void);
45 void GPIOC_init(void);
46 void EXTI_init(void);
47
48 /* Function which formats data for use with lcd.h */
49 void writeToLCD(uint32_t frequency);
50 void calculateFrequencyFromTIM2(void);
51
52 /* Global variable indicating which LED is blinking */
53 #define MODE_555_TIMER (uint16_t)0x0100
54 #define MODE_FUNCTION_GENERATOR (uint16_t)0x0200
55 volatile uint16_t activeMode = MODE_FUNCTION_GENERATOR; /* Start program in mode defined here */
56 volatile uint8_t firstEdgeFlag = 0;
57
58 int main()
59 {
60     trace_printf("Initializing ECE355 project\n");
61     trace_printf("By: Joel Meuleman and Michael Pillon\n");
62     GPIOA_init(); /* Initialize push button on GPIOA */
63     TIM2_init(); /* Initialize timer TIM2 */
64     EXTI_init(); /* Initialize external interrupts */
65     GPIOC_init(); /* Initialize LEDs on GPIOC */
66     adc_init(); /* Initialize analog to digital converter */
67     dac_init(); /* Initialize digital to analog converter */
68     lcd_init(); /* Initialize LCD */
69     trace_printf("Successfully initialized project\n");
70
71     while (1); /* Keep program running with an infinite loop */
72     return EXIT_SUCCESS;
73 }
74
```



```

75  /* Initialize a pushbutton on GPIOA */
76  void GPIOA_init()
77  {
78      /* Enable clock for GPIOA peripheral */
79      RCC->AHBENR |= RCC_AHBENR_GPIOAEN;
80      /* Configure PA2 as input */
81      GPIOA->MODER &= ~(GPIO_MODER_MODER2);
82      /* Ensure no pull-up/pull-down for PA2 */
83      GPIOA->PUPDR &= ~(GPIO_PUPDR_PUPDR2);
84  }
85
86  /* Initialize 2 LEDs on GPIOC */
87  void GPIOC_init()
88  {
89      /* Enable clock for GPIOC peripheral */
90      RCC->AHBENR |= RCC_AHBENR_GPIOCEN;
91      /* Configure PC8 and PC9 as outputs for LEDs */
92      GPIOC->MODER |= (GPIO_MODER_MODER8_0 | GPIO_MODER_MODER9_0);
93      /* Ensure push-pull mode selected for PC8 and PC9 */
94      GPIOC->OTYPER &= ~(GPIO_OTYPER_OT_8 | GPIO_OTYPER_OT_9);
95      /* Ensure high-speed mode for PC8 and PC9 */
96      GPIOC->OSPEEDR |= (GPIO_OSPEEDER_OSPEEDR8 | GPIO_OSPEEDER_OSPEEDR9);
97      /* Ensure no pull-up/pull-down for PC8 and PC9 */
98      GPIOC->PUPDR &= ~(GPIO_PUPDR_PUPDR8 | GPIO_PUPDR_PUPDR9);
99      /* Initialize LED of current mode */
100     GPIOC->BSRR = activeMode;
101 }
102
103 /* Initialize external interrupts */
104 void EXTI_init()
105 {
106     /* Map EXTI0 line to PA0 for push button */
107     SYSCFG->EXTICR[0] |= SYSCFG_EXTICR1_EXTI0_PA;
108     /* Map EXTI1 line to PA1 for the 555 timer */
109     SYSCFG->EXTICR[0] |= SYSCFG_EXTICR1_EXTI1_PA;
110     /* Map EXTI2 line to PA2 for the function generator */
111     SYSCFG->EXTICR[0] |= SYSCFG_EXTICR1_EXTI2_PA;
112     /* EXTI line interrupts: set rising-edge trigger */
113     EXTI->RTSR |= EXTI_RTSR_TR0 | EXTI_RTSR_TR1 | EXTI_RTSR_TR2;
114     /* Unmask interrupts from EXTI line 0 and 2 to enable push button and function generator interrupts */
115     EXTI->IMR = EXTI_CONTROL_BIT_PUSH_BUTTON | EXTI_CONTROL_BIT_FUNCTION_GENERATOR;
116     /* Assign EXTI2 interrupt priority = 0 in NVIC */
117     NVIC_SetPriority(EXTI0_1_IRQn, 0);
118     NVIC_SetPriority(EXTI2_3_IRQn, 1);
119     /* Enable EXTI2 interrupts in NVIC */
120     NVIC_EnableIRQ(EXTI0_1_IRQn);
121     NVIC_EnableIRQ(EXTI2_3_IRQn);
122 }
123
124 /* Push button and 555 timer interrupt handler */
125 /* This handler is declared in system/src/cmsis/vectors_stm32f0xx.c */
126 void EXTI0_1_IRQHandler()
127 {
128     /* Check if PB on PA0 triggered interrupt */
129     if ((EXTI->PR & PENDING_REGISTER_PUSH_BUTTON) != 0){
130         /* Wait for button to be released */
131         while((GPIOA->IDR & GPIO_IDR_0) != 0);
132         /* Make sure timer is stopped and first interrupt with be the first edge */
133         TIM2->CR1 &= ~(TIM_CR1_CEN);
134         firstEdgeFlag = 0;
135         /* Turn off currently mode indicator LED */
136         GPIOC->BRR = activeMode;
137         /* Switch blinking LED */
138         activeMode ^= (MODE_FUNCTION_GENERATOR | MODE_555_TIMER);
139         /* Turn on switched LED */
140         GPIOC->BSRR = activeMode;
141
142         /* Enable interrupts for new mode */
143         if(activeMode == MODE_555_TIMER) {
144             trace_printf("Switched to 555 Timer\n");
145             EXTI->IMR = EXTI_CONTROL_BIT_PUSH_BUTTON | EXTI_CONTROL_BIT_555_TIMER;
146         } else {
147             trace_printf("Switched to Function Generator\n");
148             EXTI->IMR = EXTI_CONTROL_BIT_PUSH_BUTTON | EXTI_CONTROL_BIT_FUNCTION_GENERATOR;
149         }
150         /* Clear EXTI0 interrupt pending flag (EXTI->PR) */
151         EXTI->PR = PENDING_REGISTER_PUSH_BUTTON;
152     }

```

```

153     /* Check if 555 timer triggered interrupt */
154     if ((EXTI->PR & PENDING_REGISTER_555_TIMER) != 0){
155         /* Call timer handler to calculate frequency */
156         calculateFrequencyFromTIM2();
157         /* Clear EXTI1 interrupt pending flag (EXTI->PR) */
158         EXTI->PR = PENDING_REGISTER_555_TIMER;
159     }
160 }
161 }
162
163 /* Function generator interrupt handler */
164 /* This handler is declared in system/src/cmsis/vectors_stm32f0xx.c */
165 void EXTI2_3_IRQHandler()
166 {
167     /* Check if EXTI2 interrupt pending flag is indeed set */
168     if ((EXTI->PR & PENDING_REGISTER_FUNCTION_GENERATOR) != 0)
169     {
170         /* Call timer handler to calculate frequency */
171         calculateFrequencyFromTIM2();
172         /* Clear EXTI2 interrupt pending flag (EXTI->PR) */
173         EXTI->PR = PENDING_REGISTER_FUNCTION_GENERATOR;
174     }
175 }
176
177 void calculateFrequencyFromTIM2() {
178     if(firstEdgeFlag == 0){ /* If this is the first edge */
179         /* Clear count register */
180         TIM2->CNT = 0;
181         /* Set firstEdgeFlag */
182         firstEdgeFlag = 1;
183         /* Start timer */
184         TIM2->CR1 |= TIM_CR1_CEN;
185     } else { /* Else this is the second edge */
186         /* Stop timer */
187         TIM2->CR1 &= ~(TIM_CR1_CEN);
188         /* Read out count register */
189         uint32_t currentCount = TIM2->CNT;
190         /* Calculate signal frequency */
191         uint32_t signalFrequency = SystemCoreClock / currentCount;
192         writeToLCD(signalFrequency);
193         /* Reset firstEdgeFlag */
194         firstEdgeFlag = 0;
195     }
196 }
197
198 /* Write formatted output to LCD screen */
199 void writeToLCD(uint32_t frequency) {
200     /* Write Frequency to LCD */
201     char lcd_line1[LCD_LINE_LENGTH + 1];
202     sprintf(lcd_line1, "F: %li Hz", frequency);
203     lcd_display_line1(lcd_line1);
204
205     /* Get ADC value, convert to resistance and send the raw value to the DAC */
206     uint32_t rawAdcValue = adc_readRaw();
207     dac_write(rawAdcValue);
208
209     /* Write Resistance to LCD */
210     uint32_t adcResistance = (uint32_t)adc_convertRawToResistance(rawAdcValue);
211     char lcd_line2[LCD_LINE_LENGTH + 1];
212     sprintf(lcd_line2, "R: %li Ohm", adcResistance);
213     lcd_display_line2(lcd_line2);
214 }
215
216 #pragma GCC diagnostic pop
217

```

Appendix B - adc.h and adc.c

```
1  /*
2  * adc.h
3  *
4  */
5
6  #include "diag/Trace.h"
7  #include "cmsis/cmsis_device.h"
8  #include "dac.h"
9
10 #ifndef ADC_H_
11 #define ADC_H_
12
13 #define VDDA 3.3 /* Assume VDDA = 3.3V */
14 #define MAX_RESISTANCE_VALUE 5000 /* in Ohms */
15 #define ADC_RESOLUTION 4095 /* 2^12 - 1 */
16
17 /* Configure relevant ADC registers, start ADC */
18 void adc_init();
19
20 /* Reads the raw binary ADC value from 0 to ADC_RESOLUTION */
21 uint32_t adc_readRaw();
22 /* Helper to convert raw value to voltage */
23 double adc_convertRawToVoltage(uint32_t rawValue);
24 /* Reads the raw ADC value and converts it to a voltage */
25 double adc_readVoltage();
26 /* Helper to convert raw value to resistance */
27 double adc_convertRawToResistance(uint32_t rawValue);
28 /* Reads the raw ADC value and converts it to a resistance */
29 double adc_readResistance();
30
31 #endif /* ADC_H_ */
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2480
2481
2482
2483
2484
2485
2486
2487
2488
2489
2490
2491
2492
2493
2494
2495
2496
2497
2498
2499
2500
2501
2502
2503
2504
2505
2506
2507
2508
2509
2510
2511
2512
2513
2514
2515
2516
2517
2518
2519
2520
2521
2522
2523
2524
2525
2526
2527
2528
2529
2530
2531
2532
2533
2534
2535
2536
2537
2538
2539
2540
2541
2542
2543
2544
2545
2546
2547
2548
2549
2550
2551
2552
2553
2554
2555
2556
2557
2558
2559
2560
2561
2562
2563
2564
2565
2566
2567
2568
2569
2570
2571
2572
2573
2574
2575
2576
2577
2578
2579
2580
2581
2582
2583
2584
2585
2586
2587
2588
2589
2590
2591
2592
2593
2594
2595
2596
2597
2598
2599
2600
2601
2602
2603
2604
2605
2606
2607
2608
2609
2610
2611
2612
2613
2614
2615
2616
2617
2618
2619
2620
2621
2622
2623
2624
2625
2626
2627
2628
2629
2630
2631
2632
2633
2634
2635
2636
2637
2638
26
```

Appendix C - dac.h and dac.c

```
1  /*
2  * dac.h
3  *
4  */
5
6  #include "diag/Trace.h"
7  #include "cmsis/cmsis_device.h"
8
9  #ifndef DAC_H_
10 #define DAC_H_
11
12 /* Initialize DAC */
13 void dac_init();
14
15 /* Read DAC value */
16 uint32_t dac_read();
17 /* Write DAC register */
18 void dac_write(uint32_t value);
19
20 #endif /* DAC_H_ */
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2480
2481
2482
2483
2484
2485
2486
2487
2488
2489
2490
2491
2492
2493
2494
2495
2496
2497
2498
2499
2500
2501
2502
2503
2504
2505
2506
2507
2508
2509
2510
2511
2512
2513
2514
2515
2516
2517
2518
2519
2520
2521
2522
2523
2524
2525
2526
2527
2528
2529
2530
2531
2532
2533
2534
2535
2536
2537
2538
2539
2540
2541
2542
2543
2544
2545
2546
2547
2548
2549
2550
2551
2552
2553
2554
2555
2556
2557
2558
2559
2560
2561
2562
2563
2564
2565
2566
2567
2568
2569
2570
2571
2572
2573
2574
2575
2576
2577
2578
2579
2580
2581
2582
2583
2584
2585
2586
2587
2588
2589
2590
2591
2592
2593
2594
2595
2596
2597
2598
2599
2600
2601
2602
2603
2604
2605
2606
2607
2608
2609
2610
2611
2612
2613
2614
2615
2616
2617
2618
2619
2620
2621
2622
2623
2624
2625
2626
2627
2628
2629
2630
2631
2632
2633
2634
2635
2636
2637
2638
2639
2640
2641
2642
2643
2644
2645
2646
2647
2648
2649
2650
2651
2652
2653
2654
2655
2656
2657
2658
2659
2660
2661
2662
2663
2664
2665
2666
2667
2
```

Appendix D - timer.h and timer.c

```
1  /*
2  * timer.h
3  *
4  */
5
6  #include "diag/Trace.h"
7  #include "cmsis/cmsis_device.h"
8
9  #ifndef TIMER_H_
10 #define TIMER_H_
11
12 /* Clock prescaler for TIM2 timer: no prescaling */
13 #define TIM2_PRESCALER ((uint16_t)0x0000)
14 /* Default auto reload for TIM2 of maximum value */
15 #define TIM2_PERIOD ((uint32_t)0xFFFFFFFF)
16
17 /* Initialize TIM2 */
18 void TIM2_init();
19
20 #endif /* TIMER_H_ */
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2480
2481
2482
2483
2484
2485
2486
2487
2488
2489
2490
2491
2492
2493
2494
2495
2496
2497
2498
2499
2500
2501
2502
2503
2504
2505
2506
2507
2508
2509
2510
2511
2512
2513
2514
2515
2516
2517
2518
2519
2520
2521
2522
2523
2524
2525
2526
2527
2528
2529
2530
2531
2532
2533
2534
2535
2536
2537
2538
2539
2540
2541
2542
2543
2544
2545
2546
2547
2548
2549
2550
2551
2552
2553
2554
2555
2556
2557
2558
2559
2560
2561
2562
2563
2564
2565
2566
2567
2568
2569
2570
2571
2572
2573
2574
2575
2576
2577
2578
2579
2580
2581
2582
2583
2584
2585
2586
2587
2588
2589
2590
2591
2592
2593
2594
2595
2596
2597
2598
2599
2600
2601
2602
2603
2604
2605
2606
2607
2608
2609
2610
2611
2612
2613
2614
2615
2616
2617
2618
2619
2620
2621
2622
2623
2624
2625
2626
2627
2628
2629
2630
2631
2632
2633
2634
2635
2636
2637
2638
2639
2640
2641
2642
2643
2644
2645
2646
2647
2648
2649
2650
2651
2652
2653
2654
2655
2656
2657
2658
2659
2660
26
```

Appendix E - lcd.h and lcd.c

```
1  /*
2  * lcd.h
3  *
4  */
5  #include "stdint.h"
6  #include "string.h"
7
8  #include "diag/Trace.h"
9  #include "cmsis/cmsis_device.h"
10
11 #ifndef LCD_H_
12 #define LCD_H_
13 /* LCD Pin defines */
14 #define HANDSHAKE GPIO_IDR_7
15 #define DB0        0x0100
16 #define DB1        0x0200
17 #define DB2        0x0400
18 #define DB3        0x0800
19 #define DB4        0x1000
20 #define DB5        0x2000
21 #define DB6        0x4000
22 #define DB7        0x8000
23 #define RW         0x0040
24 #define RS         0x0020
25 #define EN         0x0010
26
27 /* LCD constants */
28 #define LINE_1     DB7
29 #define LINE_2     DB7 | DB6
30 #define LCD_LINE_LENGTH 16
31
32 void lcd_gpio_init();
33
34 /* Initialize LCD hardware */
35 void lcd_init();
36
37 void lcd_display_line1(char *data);
38 void lcd_display_line2(char *data);
39 void lcd_clearDisplay();
40
41 /*
42 * Write a single character or instruction
43 * Must be shifted to data pins ( << 8 )
44 * Must include bits for register select and read/write
45 */
46 void lcd_write(uint16_t data);
47
48 #endif /* LCD_H_ */
49
50 /*
51 * lcd.c
52 *
53 */
54
55 #include "lcd.h"
56
57 void lcd_gpio_init(){
58     /* Enable clock for GPIOB */
59     RCC->AHBENR |= RCC_AHBENR_GPIOBEN;
60     /* Set all needed pins as outputs */
61     GPIOB->MODER |= GPIO_MODER_MODER4_0
62         | GPIO_MODER_MODER5_0
63         | GPIO_MODER_MODER6_0
64         | GPIO_MODER_MODER8_0
65         | GPIO_MODER_MODER9_0
66         | GPIO_MODER_MODER10_0
67         | GPIO_MODER_MODER11_0
68         | GPIO_MODER_MODER12_0
69         | GPIO_MODER_MODER13_0
70         | GPIO_MODER_MODER14_0
71         | GPIO_MODER_MODER15_0;
72     /* Set high speed mode */
73     GPIOB->OSPEEDR |= 0xFFFFFFF;
74 }
75
76
```

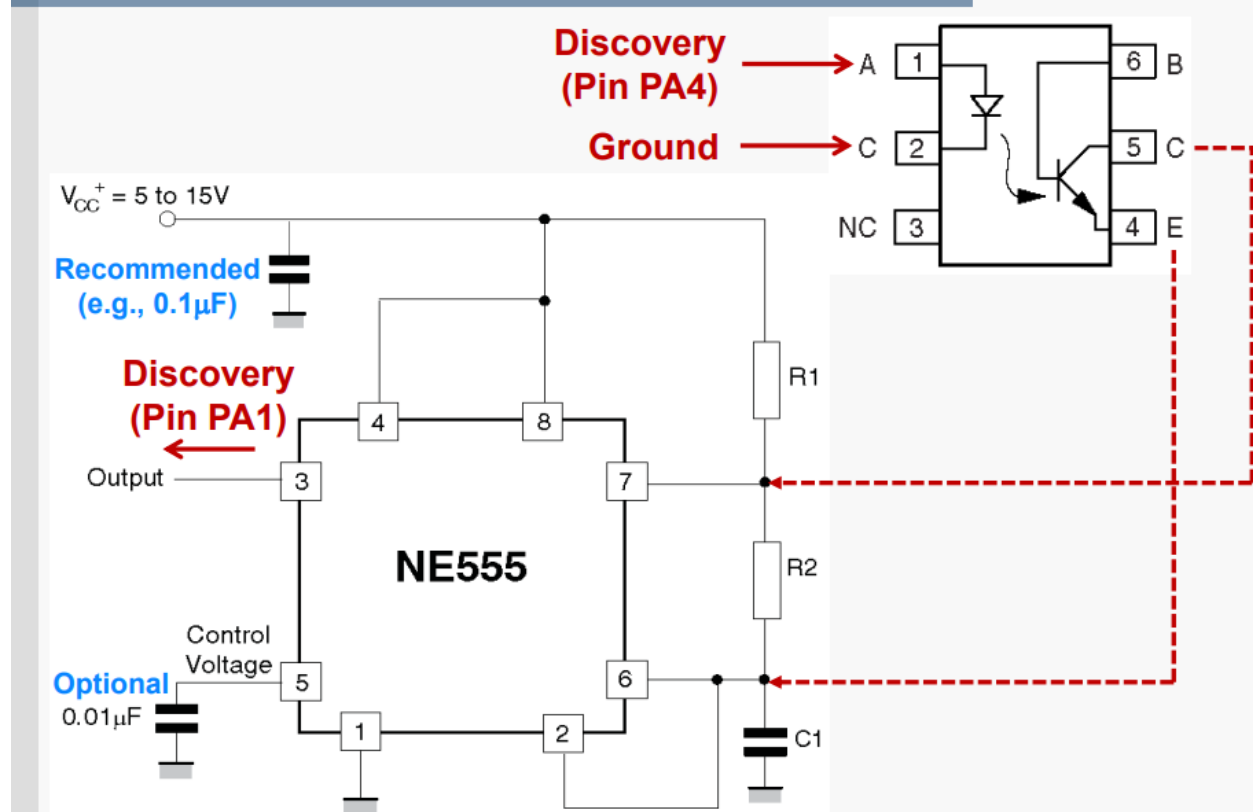
```

27 void lcd_init()
28 {
29     lcd_gpio_init(); /* Initialize pins for LCD */
30     lcd_write(DB5 | DB4 | DB3); /* 8-bit mode, 2 line, 5x7 characters */
31     lcd_write(DB3 | DB2); /* Display on, cursor off, blink off */
32     lcd_write(DB2 | DB1); /* Auto-increment enabled, increment to the right */
33     lcd_clearDisplay();
34 }
35
36 void lcd_write(uint16_t data)
37 {
38     GPIOB->BSRR |= data; /* Write data to pins */
39     GPIOB->BSRR |= EN;
40     while(!(GPIOB->IDR & HANDSHAKE));
41     GPIOB->BSRR |= EN << 16;
42     while((GPIOB->IDR & HANDSHAKE));
43     GPIOB->BSRR |= data << 16;
44 }
45
46 void lcd_clearDisplay()
47 {
48     lcd_write(DB0);
49 }
50
51 void lcd_display_line1(char *data)
52 {
53     /* Select line 1 */
54     lcd_write(LINE_1);
55     unsigned int i = 0;
56     /* Write data */
57     for(; i < strlen(data); i++){
58         lcd_write(RS | data[i]<<8);
59     }
60     /* Ensure the rest of the line is empty */
61     for(; i < LCD_LINE_LENGTH; i++){
62         lcd_write(RS | ' ' << 8);
63     }
64 }
65
66 void lcd_display_line2(char *data)
67 {
68     /* Select line 2 */
69     lcd_write(LINE_2);
70     unsigned int i = 0;
71     /* Write data */
72     for(; i < strlen(data); i++){
73         lcd_write(RS | data[i]<<8);
74     }
75     /* Ensure the rest of the line is empty */
76     for(; i < LCD_LINE_LENGTH; i++){
77         lcd_write(RS | ' ' << 8);
78     }
79 }

```

Appendix F - Multivibrator Circuit Schematic [4]

Example: 4N35 Connection



References

- [1] B. Sirna and D. Rakhamatov, “ECE 355: Microprocessor-Based Systems Laboratory Manual”, 2021, <https://ece.engr.uvic.ca/~ece355/lab/ECE355-LabManual-2021.pdf> (accessed Nov 29, 2021)

- [2] STMicroelectronics, “RM0091 Reference Manual”, 018940 Rev 5 datasheet, Jan. 2014, <https://www.ece.uvic.ca/~ece355/lab/supplement/stm32f0RefManual.pdf> (accessed Nov. 29, 21)

- [3] STMicroelectronics, “STM32F051xx”, 022265 Rev 4 datasheet, Jan. 2014, https://www.ece.uvic.ca/~ece355/lab/supplement/stm32f051r8t6_datatsheet_DM00039193.pdf (accessed Nov. 29, 21)

- [4] D.N. Rakhmatov. (2021). Interfacing [Online]. Available: <https://www.ece.uvic.ca/~daler/courses/ece355/interface.pdf>

- [5] Vishay Semiconductors, “Optocoupler, Phototransistor Output, with Base Connection”, 81181 Rev 1.2 datasheet, Jan. 2010, <https://www.ece.uvic.ca/~ece355/lab/supplement/4n35.pdf> (accessed Nov. 29, 21)

- [6] Hitachi, “HD44780U (LCD-II)”, ADE-207-272(Z) Rev 0 datasheet, 1998, <https://www.ece.uvic.ca/~ece355/lab/supplement/HD44780.pdf> (accessed Nov. 29, 21)