ENGIN 346: Microcontrollers

Engineering Department

University of Massachusetts, Boston

Semester: Fall 2020

Instructor: Dr. Honggang Zhang

Project Report for:

Project #5: ADC and DAC

By

Name: Tyler McKean

Student ID: 01098154

Date: Dec 8th, 2020

I pledge to uphold the governing principles of the Code of Student Conduct of the University of Massachusetts Boston. I will refrain from any form of academic dishonesty or deception, cheating, and plagiarism. I pledge that all the work submitted here is my own, and that I have clearly acknowledged and referenced other people's work. I am aware that it is my responsibility to turn in other students who have committed an act of academic dishonesty; and if I do not, then I am in violation of the Code. I will report to formal proceedings if summoned.

Signature: [illegible]

TABLE OF CONTENTS

NOTE: To automatically update ToC, right click above and select “Update Field”

Introduction

For Project 5, the main goal was:

1. Understanding the basic concepts of ADC and DAC.

The main objectives of the project are to use the onboard ADC to capture an analog voltage from an Infrared transmitter diode and Infrared phototransistor that varies the voltage when an obstacle moves closer to the sensors. The transmitter diode's infrared signals are captured more from the receiver when an obstacle is closer because more of the infrared signal bounces back towards the receiver. This causes the voltage to dip down by several millivolts, so by using the ADC we can capture a particular distance away from both the infrared diodes. After choosing a particular voltage, which is proportional to some distance, the Green LED of the Discovery Board should turn on, and trigger the DAC to omit a tone of 440Hz to a Piezoelectric buzzer. This can be achieved by counting the number of interrupts that occur in (1/440Hz) secs and programming the DAC to output a square wave with a 50% duty cycle and frequency of 440Hz, which is a musical A note. Both the ADC and DAC need to be initialized and triggered by the trigger output event (TRGO) of TIM4 as well.

Background and Design

The design of this project required the initialization of GPIO pins PA.1 for the ADC, pin PA.5 for the DAC, and TIM4 to be configured to trigger both the ADC and DAC process. The physical connections for the circuit can be seen in Figure 1 below.

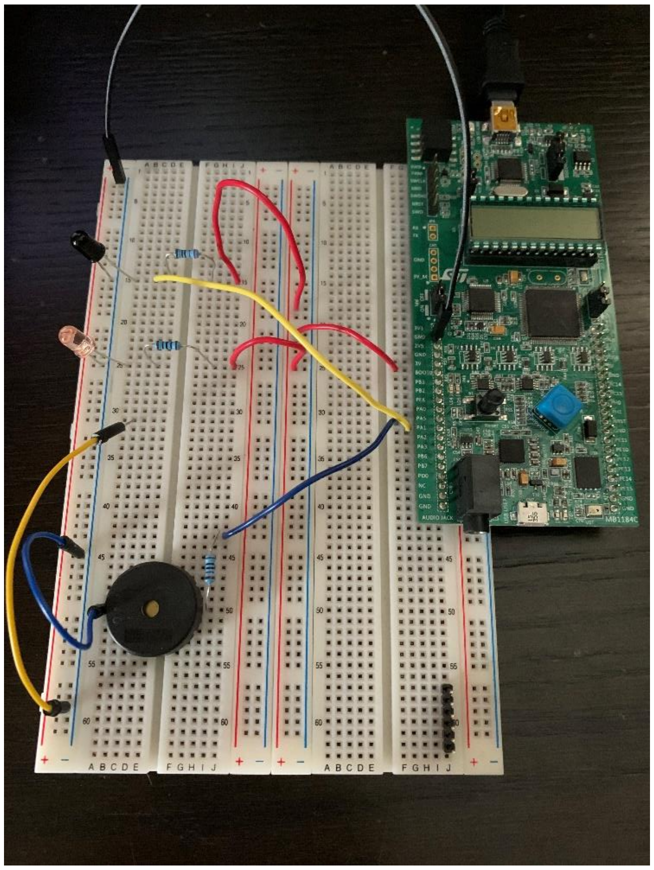

Figure 1 - Physical Connections for Project 5 to the Discovery Board

GPIO Pin PA.1 make use of ADC1 for the project and captures the voltage drop across the 5mm phototransistor. The phototransistor's anode was tied to a 2.2kΩ resistor which was tied to the 3V pin of the Discovery board, with the cathode plugged right into the GND rails of the breadboard. The Infrared LED had its cathode also plugged into the GND rails of the breadboard with its anode tied to a 100Ω resistor and the 3V pin of the Discovery board. The forward current of the Infrared LED needed to be close to 20mA in order to have a radiation intensity of about 20mW/sr to trigger the phototransistor. When enough of the infrared signal was captured by the phototransistor a base current in the transistor would turn it on and produce a low voltage level that PA.1 could capture. The voltage across the phototransistor drops more when an obstacle reflects more of the infrared signal from the infrared LED. The ADC output can be estimated by the formula:

$$\text{ADC Output} = \text{round}((2^N - 1) * V_{in}/V_{ref} \qquad \text{(Eq 1-1)}$$

where N is the bit resolution, V_{ref} is the reference voltage, and V_{in} is the voltage drop across the phototransistor. For this circuit, we are using the 3V pin on the Discovery board as our V_{ref} and the bit resolution was set as 12-bits in the ADC CFGR. So, when the ADC reads an analog voltage that's dropping across the phototransistor, that value will be converted into a digital value ranging from 0 to 4095 based on the bit resolution of the ADC. After the ADC converts the analog value into a digital value, the End of Conversion flag will be triggered, which will stop the main program and redirect its attention to the ADC1_2_IRQHandler. Within the handler, I was able to save the current value in the Data Register of the ADC to a variable called **result**. An if statement could then be written to check when that result variable was proportional to an analog voltage at some distance away from the phototransistor. If that statement was true, the handler would turn on the Green LED from the Green_LED_On() function from the included LED.c files added to project window of the program. If the statement was false, the Green LED would then be turned off. These statements made it possible to move an obstacle close to the IR sensor and turn the Green LED on and would turn it off when the obstacle backs away from the sensor.

Timer 4 was used as the output trigger event that would trigger both the ADC and DAC for the project. The timer was initialized using the TIM4_Init() in the main program which came from the included TIM.c file added to the project window of the project. The timer was initialized as PWM Mode 1 with a PSC value of 18, ARR value of 18, and CCR1 value of 9. These values scaled the system clock down from 80MHz to about 221kHz. The CCR1 value also made it so that the OC1REF had a duty cycle of about 50%. The trigger interrupt and update interrupt were also enabled in order to use the TIM4_IRQHandler. Afterwards, timer 4 was then enabled in the NVIC IRQ so that we could use the timer's handler to trigger the DAC output.

DAC channel 2 is tied to GPIO pin PA.5 on the Discovery board and was used to send a 440Hz square wave to the Piezoelectric buzzer when the ADC handler read a specific value in its DR. Pin Pa.5 was tied to a 1kΩ resistor that was tied to the buzzer that was then tied to the GND rails of breadboard. The initialization for DAC channel 2 was done in the DAC_Init() function provided from the DAC.c file added to the project window. TIM4_TRGO was selected as the DAC trigger for channel 2 in the DAC CR. I was able to output a 440Hz square wave pulse to the buzzer was utilizing the TIM4 handler and counting the number of interrupts that occurred within 1/440Hz seconds. Since I was using the 80MHz system clock, I was able to count about 503 times the interrupt occurred within that time frame. Since a square wave is essentially a high voltage and then low voltage for a 50% duty cycle. I wrote some if statements that included a temporary variable that was set to equal 1 whenever the ADC read a particular voltage with an obstacle at a close distance from the IR sensor. When this temp variable was 1, I set the DH12R2 output to be 4095, which is a digitized high voltage of 3V, for high the time frame of 1/440Hz and then set the DAC to output a zero voltage for the second half of that timeframe. This is how I was able to output a square wave at a frequency of 440Hz, which produced a musical A note from the buzzer.

Experiment and Results

This section of the report will discuss what pins and registers of the Discovery board needed to be initialized to achieve the project goals.

To start with the ADC, I included both an ADC.c and ADC.h files into my project, so I could simply call a function called ADC_Init() that would provide all the proper configurations in order to initialize the ADC for PA.1. The function starts by enabling the clock of the ADC in the AHB2ENR and then resetting and initializing the ADC interface in the AHB2RSTR. A function called ADC_Pin_Init() then enables the clock for GPIO Port A and configures pin PA.1 as Analog (11) in the MODER, sets the pin as no pull-up/pull-down in the PUPDR, and connects the analog switch to the ADC input by enabling bit GPIO_ASCR_EN_1 in the Analog Switch Control Register. The ADC_Init() function then executes and sets the ADC to have 12-bit resolution and the data alignment to be right aligned. In order for the ADC to be triggered by TIM4_TRGO event the ADC1 CFGR needs to set the EXTSEL bits to be 1100 and this is preciously what is done. The external trigger enable bits are then set to 01 for hardware trigger with detection on the rising edge and ADC1 is then set to single conversion in the ADC1 CFGR. Since we are only performing one channel conversion, the ADC regular sequency register is then programmed for only one conversion and specified to channel 6 by altering the bits in the SQR1 because PA.1 is assigned to ADC12_IN6. The channel is a single-ended input so the DIFSEL bit for channel 6 is cleared. The ADC sample time is then initialized for 24.5 ADC clock cycles at the system clock of 80MHz by clearing the SMPR1_SMP6 bits and setting them as 11. This would then mean that the conversion time for the ADC would be (24.5cycles/80MHz) = 0.3μsecs. The End of Regular Conversion interrupt is then set and enabled in the NVIC for the ADC1_2_IRQn. The handler will be programmed to check when the ADC reads a particular voltage level from the IR sensor to turn the Green LED on. The ADC_Init() function then finishes by enabling the ADC in the ADC control register and then checks if the ADC is ready in the ISR.

For the Timer, the clock was set first by enabling the APB1ENR1 from the RCC for Timer 4. The MMS bits in the TIM4 CR2 was then set to 100 for OC1REF as TRGO. The trigger interrupt and update interrupt bits were set in the DIER next. To initialize the timer as PWM mode 1, the OC1M bits were set as 0110 in the CCMR1. As mentioned before, the value of 18 was set for the PSC and ARR values, with the CCR1 set to 9 to achieve a 50% duty cycle. The timer was then enabled by the CEN bit in the CR1 and then shortly after the TIM4_IRQ was set in the NVIC.

For the DAC, GPIO pin PA.5 first needed to be initialized. The clock for GPIO Port A was enabled by the AHB2ENR from the RCC. To configure PA.5 as DAC1_OUT2 the pin needed to set as Analog (11) in the MODER and no pull-up/pull-down in the PUPDR, which was done by clearing bits 11 and 10. The DAC was then configured by enabling the DAC Clock by the APB1ENR1 to the DAC1EN bit in the RCC. The function DAC_Calibration_Channel() function then calibrated DAC Channel 2. The DAC mode control register was then set for DAC Channel 2 to be connected to an external pin with Buffer enabled by cleared the MODE2 bits in the MCR. Afterwards, the TEN2 bit was set in the DAC CR to enable the trigger. The TSEL2 bits were then configured as 101 for TIM4_TRGO as the trigger for Channel 2 of the DAC. Lastly, the EN2 bit in the DAC CR was set to enable DAC Channel 2.

In the ADC Handler, the interrupt request checks when the EOC flag was set which indicates when the ADC finishes converting an analog voltage to a digital value. The value in the ADC DR was then stored in the variable **result**, which clears the EOC flag. As mentioned before, two if statements checked when the result value was a particular digital value between 0 and 4095. I had the statement check when the analog output was less than or equal to 0xE74, which an analog value of about 2.71V across the phototransistor. This distance is about 2-3cms away from both the infrared LED and phototransistor. If this statement was true, I called the Green_LED_On() function and set the temp variable to 1, else the Green_LED_Off() function was called, and the temp variable was set to zero. The temp variable was then used a flag to tell the TIM4_Handler when to output a square wave to the buzzer.

In the TIM4_Handler, the SR checks whether the CC1IF is not equal to zero, indicating an interrupt request has occurred. The CC1IF is then reset back to zero. Another variable called **result_DAC** is then assigned as an integer i that continuously increments. A nested if statement checks if the variable **temp** is equal to 1 and then outputs the square wave if its true. The square wave is created by using the **result_DAC** with the modulus operation with the value of 503. This is because about 503 timer interrupts occur within 1/440Hz seconds, which is the period of a musical A note. If the result of this modular operation is less than or equal to the value 251, which is about half the period of the A note, the DHR12R2 of the DAC is set to a digital value of 4095, which is a voltage value of 3V, else the DH12R2 is equal to zero. This creates a pulse with a duty cycle of 50% and a period of 1/440Hz, which satisfies the requirements to synthesize a musical A note to the buzzer only when an obstacle is about 2-3cm away from the phototransistor. Lastly, the TIF and UIF flags are cleared to end the timer interrupt handler before continuing to resume the main program.

The last part of the main program consists of a while loop that uses the LCD.c functions to display the digitized output value from the ADC1_IN6. The loop then runs infinitely for as long as the Discovery board has power.

Observations and Conclusions

For voltage measurements and analysis of the Project 5 circuit, I used the oscilloscope of the Analog Discovery 2. I started my first observations by reading the voltage across the phototransistor when my hand was hovered over the IR from about 5cm away. Screenshots from the oscilloscope can be seen below.

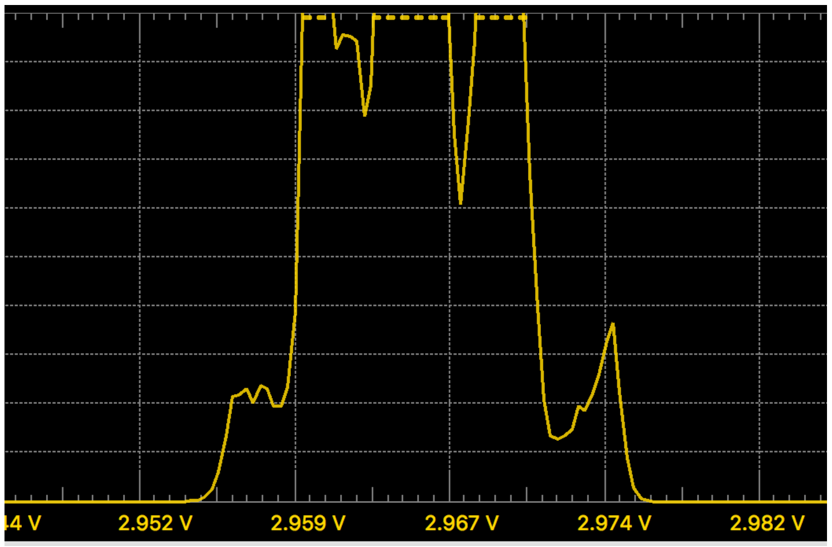

Figure 2 - Voltage drop across the Phototransistor at 1 cm away

I noticed the voltage reading was close to the V_{ref} value of 3V, so I scaled the voltage down to about 2.71V in the ADC Handler so that the obstacle had to be closer to turn on the LED and buzzer.

When hooking up the oscilloscope to the DAC output of PA.5, I observed the square wave shown in Figure 3 below.

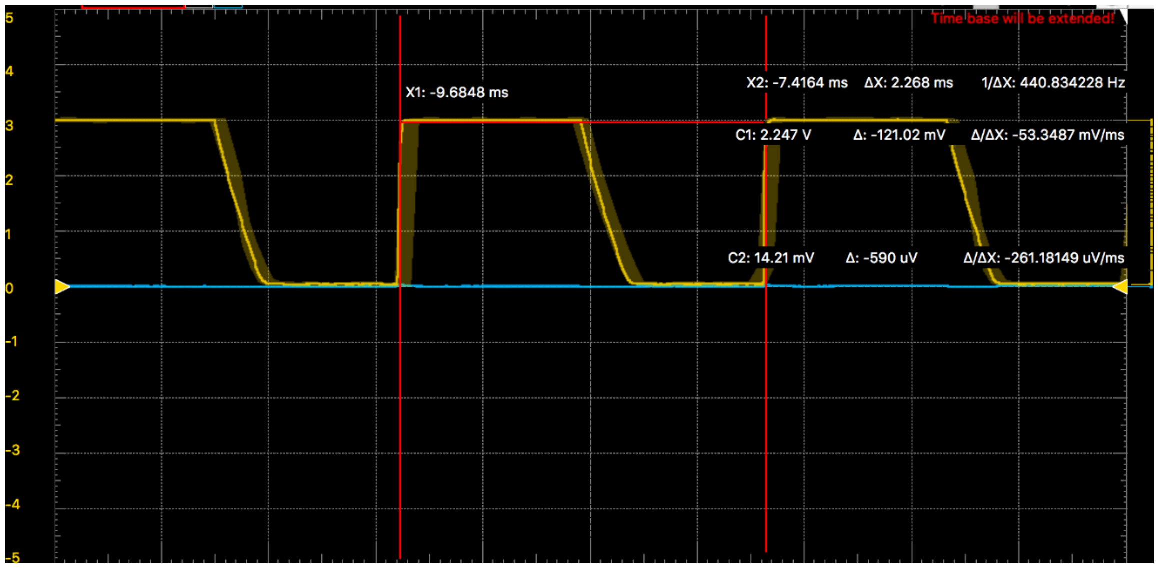

Figure 3 - Output Voltage from DAC of a square wave with frequency of 440.8Hz

The frequency of the square wave was about 440.8Hz which is about .2% off from the desired frequency.

A testing video for the IR sensor and buzzer is included in the zipped Project 5 folder

In order to test the circuit, power on the Discovery board by plugging the USB cable into the laptop and then open the Keil µVision 5 project file. Run the main.c code and then enter debug mode. Hitting the reset button will reset the Discovery board and immediately the LCD will display some values around 4050. Slowly approaching an obstacle towards to two diodes the LCD values will decrement until reaching 3700, which is about 2-3cm away, the green LED will turn on and the buzzer will emit the 440Hz musical A note. Backing the obstacle away from the sensor will turn the LED off and the buzzer will once again be silent. I happen to notice the test worked better with a flat obstacle like a folded piece of paper. This concludes the testing and observations of the report.

Bibliography

[1] Zhu, Y. (2018). *Embedded systems with ARM Cortex-M microcontrollers in assembly language and C* (pp. 384-412). United States of America: E-Man Press LLC.