Student: Ryan Enslow

Course: ECE 5210

Subject: Masters Lab, Plotting on the STM32F769I-DISCO

Date: April 23, 2024

---

1 Introduction

The STM32F769I-DISCO board, an Arm® Cortex®-M7 core-based STM32F769NI microcontroller with 2 Mbytes of flash memory and 532 Kbytes of RAM, is a versatile platform that can be utilized for various applications, including signal processing and visualization. It also includes a 4-inch 800 x 472-pixel capacitive touch TFT color LCD with serial interface, which could potentially be used for plotting signals. However, implementing such a feature on the STM32F769I-DISCO board presents several challenges.

First, plotting signals on the board can be resource-intensive as it requires one of the system clocks and vital ADC resources if any touch capabilities are utilized. This could potentially limit the board's ability to perform other tasks simultaneously. Secondly, the board has limited RAM and flash memory, and large or complex plots could quickly consume these resources, potentially causing performance issues or crashes. Thirdly, the board's display is relatively small and has a limited resolution, which could limit the detail and complexity of the plots that can be displayed. Lastly, while there are examples of graphical frameworks optimized for STM32 microcontrollers, implementing a full-featured plotting library could be a complex task requiring significant development effort.

Despite these challenges, the potential benefits of being able to plot signals directly on the STM32F769I-DISCO board are significant. It could provide a convenient way to visualize and analyze signals in real-time, potentially reducing the need for expensive external equipment like oscilloscopes. However, careful consideration would need to be given to the trade-offs involved, particularly in terms of resource usage and complexity.

2 Theory

The first challenge to overcome is the creation of a graphing interface for the STM32. Fortunately, there is an application that can significantly simplify this task. TouchGFX, a program developed by STM, is a powerful tool for designing graphical interfaces, including various styles of graphs. This tool was utilized to develop the graph used in this project, effectively addressing a lot of the backend complexities.

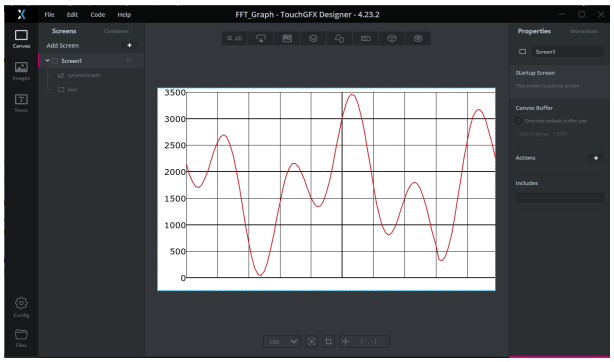

Figure 1: TouchGFX interface for creating graphs

The graph parameters in TouchGFX can be updated and configured to cater to a wide range of needs, from signal plotting to frequency analysis, and beyond. In one particular iteration of implementation, a touch button was added to the screen interface. This button, when activated, would take a sample and plot a Fast Fourier Transform (FFT), providing a frequency analysis of the signal. This feature not only enhances the functionality of the system but also enriches the user interaction with the device. TouchGFX serves as an excellent starting point for this project, paving the way for further enhancements and customization.

Next the Project was brought into STM32CubeIDE for further developement. In the IDE the input and output configurations need to be made. In previous labs an audio pass through was used that utilized the line in and line out ports on the board at pins CN6 and CN7 respectively. In order to do this some configurations to the header wm8994.h and the subsequent .c file need to be made as well as the main function. I did this by following the video here.

An alternative approach involves utilizing the pins on the bottom of the STM32F769I-DISCO board to read signals from an external source. Specifically, this method employs the CN11 pins, also commonly referred to as the Arduino pins. This option provides a flexible way to interface with a variety of signal sources. A video covering this implementation on a different board is covered here. Be aware that this implementation is for the STM32H7B3LIH so it's important to study up on the data sheet for the specific board to find the correct ADC implementation. Provided is a link to the data sheet here.

This table is very helpful for this implementation:

6.2 ARDUINO® Uno V3 connectors (CN11, CN14, CN13, and CN9)

Table 6. ARDUINO® connectors (CN11, CN14, CN13, and CN9)

Left connectors					-	Right connectors				
CN No.	Pin No.	Pin name	STM32 pin	Function	-	Function	STM32 pin	Pin name	Pin No.	CN No.
-						I2C1_SCL	PB8	D15	10	CN9 digital
						I2C1_SDA	PB9	D14	9	
						AVDD	-	AREF	8	
						Ground	-	GND	7	
CN11 power	1	NC	-	-		SPI2_SCK	PA12	D13	6	
	2	IOREF	-	3.3 V Ref		SPI2_MISO	PB14	D12	5	
	3	RESET	NRST	RESET		TIM12_CH2, SPI2_MOSI	PB15	D11	4	
	4	+3V3	-	3.3 V input/output		TIM1_CH4, SPI2_NSS	PA11	D10	3	
	5	+5 V	-	5 V output		TIM12_CH1	PH6	D9	2	
	6	GND	-	Ground		-	PJ4	D8	1	
	7	GND	-	Ground		-				
	8	VIN	-	Power input	-	-	PJ3	D7	8	CN13 digital
-						TIM11_CH1	PF7	D6	7	
CN14 analog	1	A0	PA6	ADC1_IN6		TIM3_CH3	PC8	D5	6	
	2	A1	PA4	ADC1_IN4		-	PJ0	D4	5	
	3	A2	PC2	ADC1_IN12		TIM10_CH1	PF6	D3	4	
	4	A3	PF10	ADC3_IN8		-	PJ1	D2	3	
	5	A4	PF8 or PB8(1)	ADC3_IN6 (PF8) or I2C1_SDA (PB9)		USART6_TX	PC6	D1	2	
	6	A5	PF9 or PB8(1)	ADC3_IN7 (PF9) or I2C1_SCL (PB8)		USART6_RX	PC7	D0	1	

1. For details refer to *Table 14: 32F769IDISCOVERY I/O assignment*.

Figure 2: Arduino pinout for ADC on the STM32F769I-Disco

Once the necessary implementations are in place to acquire the signal onto the STM32F769I-DISCO board, a variety of signal processing techniques can be applied, similar to those used in any typical laboratory setting. For instance, if the audio pass-through is being utilized, the signal can be processed and then displayed on the board in either an fft or the sampled signal.

The processed data is then transferred to a C++ file, specifically Screen1View.cpp. This file is responsible for managing the graphical interface of the application. The data points are plotted on the graph using the addDataPoint function of the graph1 object.

`graph1.addDataPoint(x, y)`

Where x represents the independent variable (typically time or frequency in signal processing applications), and y represents the dependent variable (the amplitude of the signal at a given time or frequency). This function call adds a new data point to the graph, allowing the signal to be visualized. This visualization capability can significantly aid in the analysis and interpretation of the signal, potentially leading to more effective and efficient signal processing implementations.

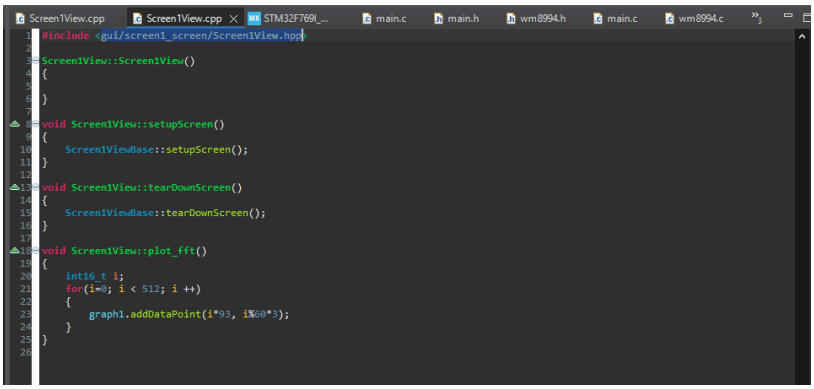

Figure 3: A function that plots a random set of numbers whenever the button is pressed.(to be replaced with collected fft data when implemented)

3 Results

Figure 4: A lot of Compile errors for line in line out use

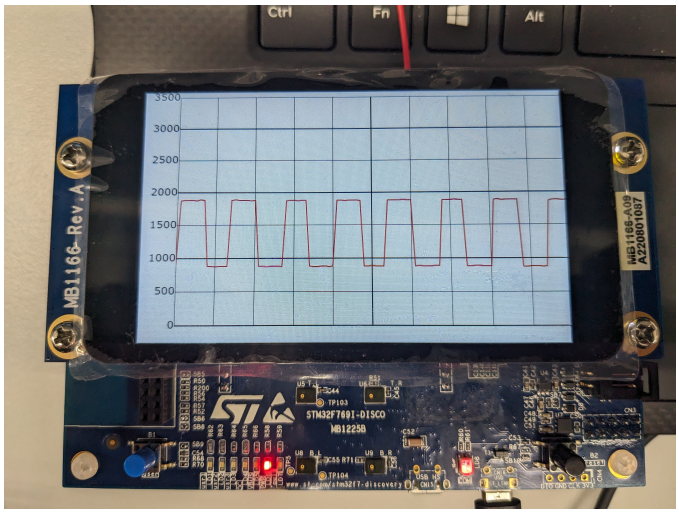

Figure 5: Square wave graphed on the board from Arduino Pin A0

4 Discussion and Conclusions

In the initial stages of the project, I was successful in getting the graph to display and populate with random data without any issues. Additionally, I was able to develop various signal processing algorithms, including a Fast Fourier Transform (FFT), leveraging the capabilities of the CMSIS libraries. These achievements marked significant progress in the project.

However, when I attempted to integrate these two components - the graphical display and the signal processing algorithms - I encountered considerable difficulties. The primary issue stemmed from the Digital-to-Analog Converters (DACs) utilized on the STM32F769I-DISCO board.

When I tried to use the line in and out, I discovered that the same DAC is employed for the touch capabilities of the LCD. This overlap caused a conflict as both functionalities attempted to use the same pins, resulting in a build failure. To resolve this, I attempted to reroute the signal through a different DAC. Unfortunately, this approach was unsuccessful as both signals followed the same routing path.

I successfully managed to configure the Arduino ports on the STM32F769I-DISCO board to read data from the Analog-to-Digital Converter (ADC). This data was then plotted directly on the screen, providing a real-time visualization of the signal. This achievement marked a significant milestone in the project, demonstrating the feasibility of using the board for signal analysis. It's important to note that the A0 pin only supports up to 12bit accuracy so the ADC value needs to be normalized to a max of 12 bits.

However, this approach does have some limitations. The sampling rate of the ADC, for instance, is a critical factor in determining the maximum frequency that can be accurately represented. In my tests, I found that the system could reliably capture signals up to about 2Hz. Beyond this frequency, the resolution of the plotted signal began to degrade. The sampling frequency of the ADC is set to be once every 16ms or 62.5Hz. Meaning that in order to not violate Nyquist the signal to be sampled needs to be about 31Hz or less. The board only being able to handle 2Hz is likely due to the time it takes the ADC to convert the value.

Another limitation I encountered pertains to the voltage range of the signal. The graph was able to plot signals with a peak-to-peak voltage of up to 2.7V. However, any signal with a voltage exceeding this threshold resulted in a flattened waveform at the peaks. This is due to the voltage limits on the Arduino pin out which are only rated for a max 3.3 volts. Also if the signal fluctuated rapidly or non-uniformly it

caused significant lag in the plotting.

This experience underscores the complexities involved in hardware-software integration, particularly when dealing with resource constraints and shared functionalities. It also highlights the importance of thorough planning and system design in embedded systems development to avoid such conflicts. Despite these challenges, the learnings from this process provide valuable insights that can guide future development efforts.

With more time I think I could make the on board microphones work or even create some better resolution in the Arduino ports. Unfortunately I just don't have the time at the end of the semester to fully flush this out and interpret the data sheet fully. The Potential is there but with current implementation of signal processing in labs isn't possible to merge the two from my understanding.