Software / Hardware Manual for building Digital Interface of Vortex

Kyoungmo Koo and Mark Draelos

July 2025

1 Introduction

This manual is intended for users who want to build a digital interface compatible with the Optical Coherence Tomography (OCT) system, which includes the Saturn 1B Galvanometer (by ScannerMAX) controlled by the MachDSP Servo Driver (by ScannerMAX). Opensource software Vortex [1] will be used for scan pattern generation and imaging process.

2 Materials

- C232HM-DDHSL-0 : $38.88 Digikey
- Nucleo-64 L476RG : $14.85 Digikey
- Wire kit : $ 9.52 Digikey
- Jumper wires(Female to Female) : $4.40 Digikey
- Breadboard: $4.25 Digikey

3 Software Setup

After connecting the **C232HM-DDHSL-0** cable to a USB port on the desktop, install the **D2XX Driver** from FTDI (Download Link). Next, download the following folders and files from the GitHub Repository of our Digital Interface:

- `src/PC`
- `src/STM32`
- `demo/EnginePattern_raster_amp1_res512_hexadecimal_downsample.txt`
- `pattern_generation.py`

For detailed installation instructions, refer to the content of each zip file as described below.

- src/STM32: We recommend that the user use **Visual Studio** as the development environment. The folder contains `stm32.cpp`, which serves as the main file for establishing the **SPI (Serial Peripheral Interface)** link between the desktop and the STM32 board via the FTDI cable.

  The remaining files in the zip package include:

  - **Header Files**: `STM32.h`, `WinTypes.h`, `ftd2xx.h`, `libmpsse_spi.h`
  - **Library File**: `libmpsse.lib`

  To properly configure the project:

  1. Add `STM32.h`, `WinTypes.h`, `ftd2xx.h`, and `libmpsse_spi.h` to the **"Header Files"** section in the Visual Studio solution.

2. Add `libmpsse.lib` to the **"Resource Files"** section.

For a detailed explanation of the functions used in `stm32.cpp`, refer to the examples and source code. The **Visual Studio Solution Explorer** should appear as shown in Figure 1.

Before running the code, the user must customize the following lines:

1. **Specify the correct COM port**: Modify the following line in `stm32.cpp` to match the COM port assigned to the STM32 board in the **Device Manager**:

```
hComm = CreateFile(L"\\\\.\\COM6", GENERIC_READ | GENERIC_WRITE, 0, NULL,
                   OPEN_EXISTING, FILE_ATTRIBUTE_NORMAL, NULL);
```

2. **Set the input file for STM32**: Modify the file path in the following line to match the desired `.txt` file for transmission to the STM32 board:

```
fopen_s(&fr, "EnginePattern_raster_amp1_res512_hexadecimal_downsample.txt", "r");
```

3. **Adjust scanning parameters**:
   - Multiply the number of **B-Scans** by **32** and update the variable `COUNT`.
   - Customize the variables `CHUNK_NUM` and `NUM_OF_POSITIONS_PER_CHUNK`, ensuring they meet the following conditions:
     (a) `CHUNK_NUM` × `NUM_OF_POSITIONS_PER_CHUNK` = Total number of positions in the `.txt` file.
     (b) `NUM_OF_POSITIONS_PER_CHUNK` must be greater than **2000**.
     (c) `CHUNK_NUM` must be an **even number**.

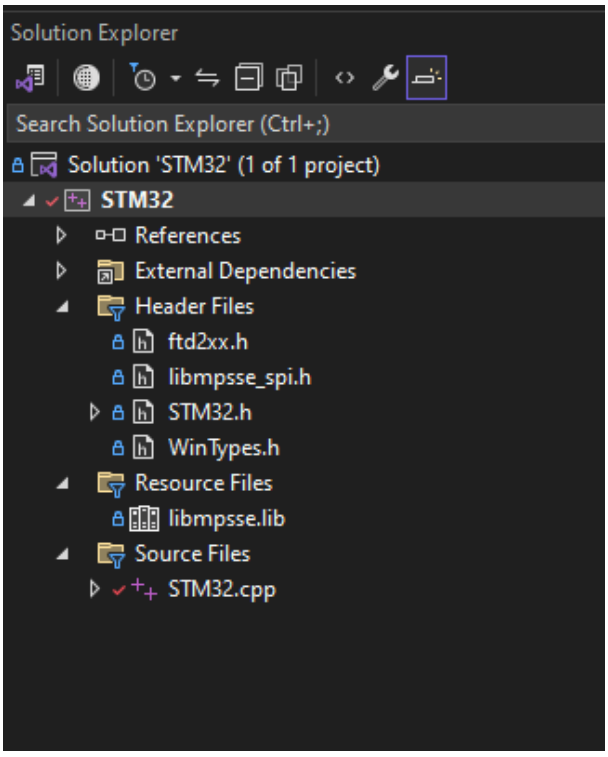

Figure 1: Visual Studio Solution Explorer after adding the required files.

- `src/STM32`: We recommend that the user use **STM32CubeIDE** as the development environment. Move the extracted `src/STM32` folder into the workspace directory of STM32CubeIDE, then open the project using the **"Import Projects from File System"** function. Once the project is opened, the **Solution Explorer** in STM32CubeIDE should appear as shown in Figure 2.

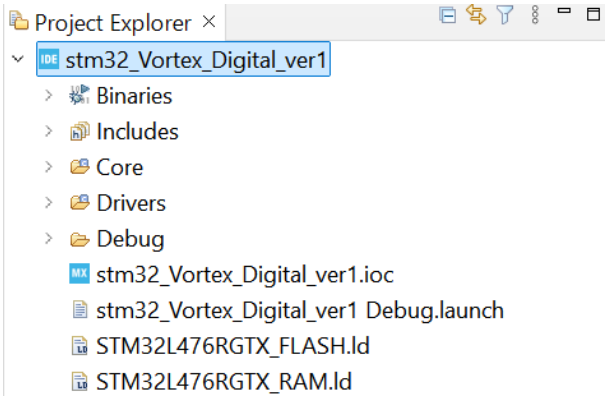

Figure 2: Screenshot of STM32CubeIDE after the user opens the project.

- `Files for imaging`: We recommend that the user use **Visual Studio Code** as the development environment for imaging. Our repository contains the following files:

  1. `pattern_generation.py`: The user can customize the variables `ascans_per_bscan` and `bscans_per_volume`, which determine the resolution of the B-Scan. Additionally, the variable `scan_dimension` can be modified to match the physical size of the image.
  2. `demo/EnginePattern_raster_amp1_res512_hexadecimal_downsample.txt`: This text file is used to run the demo of the digital interface. It contains a scan pattern for a raster scan with a 1°amplitude and a resolution of 512 × 512 pixels.

4 Hardware Setup

The user requires jumper wires to connect the three devices: the **Desktop**, **STM32 board**, and **Servo Driver**. In **STM32CubeIDE**, the user interface displays the `.ioc` file, which specifies the communication standard supported by each pin, as shown in Figure 3.

Figure 4 illustrates the pin configurations of the **FTDI SPI Cable** and the **digital port of the Servo Driver**. The image of the FTDI SPI cable was sourced from the *FTDI C232HM USB 2.0 Hi-Speed to MPSSE Cable Datasheet* [2]. The image of the Servo Driver was obtained from the *Mach-DSP Servo Driver and Associated Software User Manual* by ScannerMAX [3].

According to the manual, when the **Digital Scanner Interface (FB4 protocol)** is selected as the input source, each pin serves a specific function as follows.

- Pin 1: Ground
- Pin 2: Not used
- Pin 3: Position Feedback output for FB4
- Pin 4: XY-axis data input for FB4
- Pin 5: Serial Port Frame Sync Input
- Pin 6: Serial Port Clock input
- Pin 7: Not used
- Pin 8: Not used

Table 1 lists all the necessary connections required before operation. Since the STM32 board has both male and female ports, both **Male-to-Male** and **Female-to-Female** jumper wires are needed to complete all the connections.

The ports in the same row should be connected and must share the same voltage. If a port appears in multiple rows, such as `PB13`, it should be connected to all corresponding elements in both rows using a breadboard.

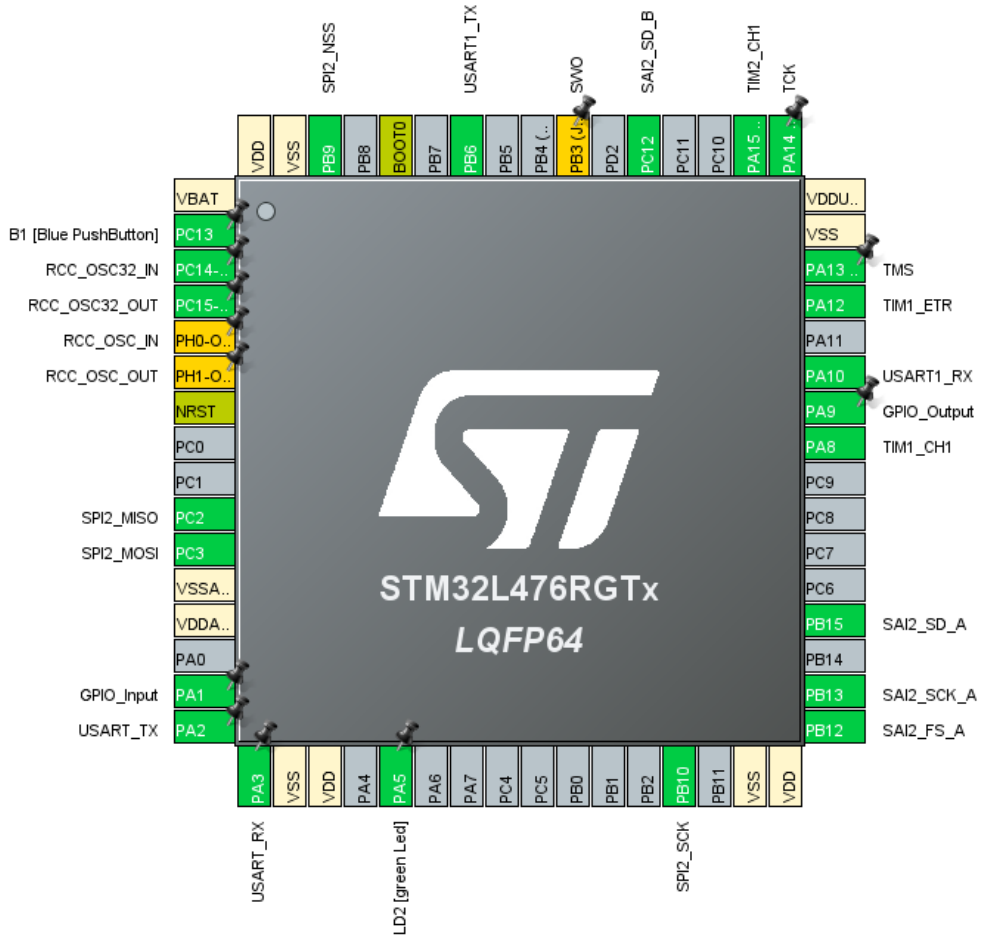

Figure 3: STM32 pinout configuration.

5 PCB Design

To streamline hardware setup and improve system robustness, a custom PCB has been developed for digital galvanometer control. This board can be ordered through standard PCB manufacturers such as JLCPCB by submitting the `Digital_Galvo_Gerber.zip` file available in our open-source repository. The PCB dimensions are specifically tailored to fit the Nucleo-64 L476RG development board, which follows the mechanical layout of its parent board, the MB1136 [4].

Figure 5a presents the full schematic of the control interface. Two female headers (P3 and P4; 2.54 mm pitch, 1×2 configuration) are designed to mechanically align with the male headers on the Nucleo board without carrying electrical signals. The primary digital control lines are interfaced through two 38-pin headers (U2 and U3, model: `PM254-2-19-Z-8.5`).

Signal output to the galvanometer servo driver is managed through connector J2, while J1 provides serial communication via an FTDI USB-to-SPI cable. A laser trigger can be interfaced through the SMA connector U1 (`HL-SMA-KWE175-02`). The pin mappings for J1 and J2 are summarized in Table 2.

All connectors—including headers, Micro-Fit, and the SMA trigger port—must be soldered to the PCB prior to use. Recommended component models for soldering are listed in Table 3. After assembly, individual signal pins can be easily connected to the main control board or external cables via those components.

Figure 6 presents the layout and design of the custom PCB developed for digital galvanometer control. Panel (a) shows the top copper layer with red fill, where traces from the main 38-pin headers (CN7 and

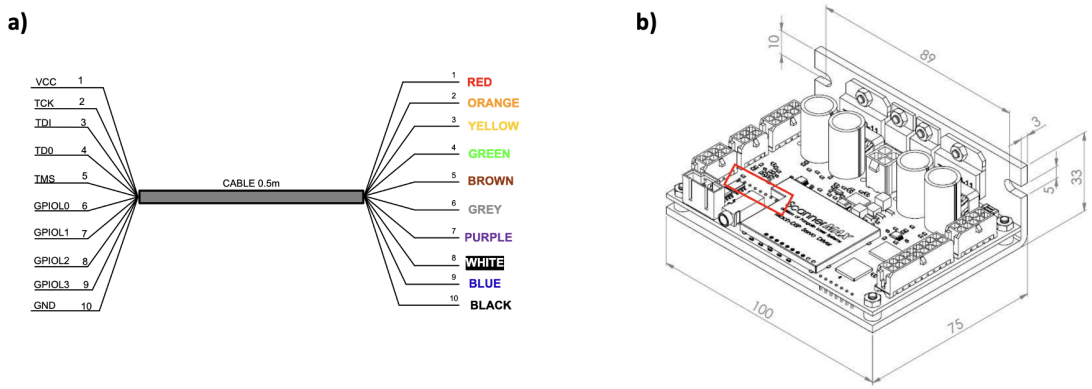

Figure 4: a) Configuration of FTDI SPI cable [2], b) MachDSP Servo Driver. The portion inside the red box is the port used for digital interface [3]

Function	FTDI Cable	STM32	Servo Driver
Ground(SAI Link)	-	GND (CN6)	Pin 1
Position Feedback output	-	PC12	Pin 3
XY-axis data input	-	PB15	Pin 4
Frame Sync	-	PB12	Pin 5
Clock(SAI Link)	-	PB13	Pin 6
Ground(SPI Link)	Black	GND (CN6)	-
VCC(SPI Link)	Red	5V (CN6)	-
Clock (SPI Link)	Orange	PB10	-
Data Output	Yellow	PC3	-
Data Input	Green	PC2	-
Chip Select	Brown	PB9	-
Laser Trigger	-	PA1, PA12	-
Timer Interface 1	-	PA8, PB12	-
Timer Interface 2	-	PA15, PB13	-

Table 1: Connection required between FTDI Cable, STM32, Servo Driver, and Laser Trigger

CN10) are routed toward peripheral connectors labeled SPI, LASER, and FB4. Panel (b) overlays the top and bottom copper layers, with red and blue regions indicating different planes. Panel (c) shows the silkscreen layer in blue, including text labels such as “CN7,” “SPI,” and “LASER,” which are outlined in white and printed on the board to aid assembly and orientation. Panel (d) displays the 3D rendering of the assembled PCB, including two stacked 38-pin headers, a 6-pin SPI header, and an SMA connector for the laser trigger. This hardware layout ensures compatibility with the Nucleo-64 L476RG board and supports robust interfacing with the galvanometer driver and external controllers.

6 Scan

Following is the detailed procedure for scan.

1. Turn on the Laser.
2. Turn on the **MachDSP Software** and set the **Input Source** to `Digital Serial Interface (FB4-Compatible)`. The user can verify whether the signal is being delivered correctly by using the **Scope** function and checking `XTP91 (Raw Command)` and `YTP91 (Raw Command)`.
3. Run `main.c` from the `stm32_Vortex_Digital_ver1` project in **STM32CubeIDE**.
4. In **Visual Studio**, run the main source file, `STM32.cpp`. If selecting ”Run” does not work, try **”Run Without Debugging”**.

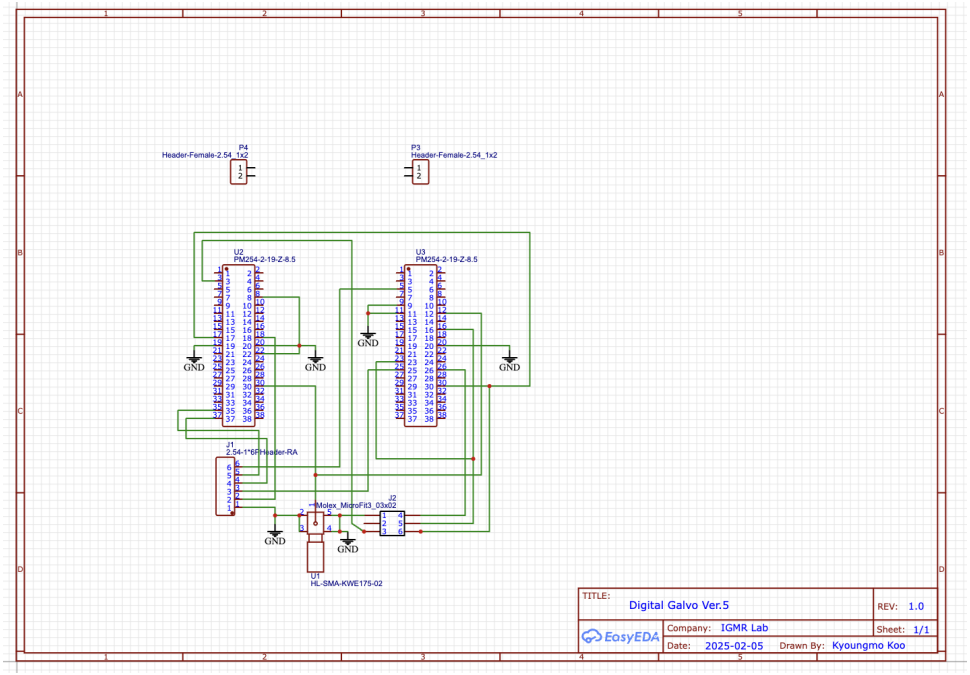

Figure 5: Schematic of PCB

Table 2: Pin assignments for J1 (Servo Driver Interface) and J2 (FTDI Cable Interface).

Pin	**J1 – FTDI Cable (SPI)**	**J2 – Servo Driver(FB4)**
1	Black (GND)	Servo Driver Pin 1
2	Red (VCC)	Not Connected
3	Orange (TXD)	Servo Driver Pin 3
4	Yellow (RXD)	Servo Driver Pin 4
5	Green (RTS)	Servo Driver Pin 5
6	Brown (CTS)	Servo Driver Pin 6

5. When the program starts, select the correct channel corresponding to the connected FTDI SPI Cable:
   - If no other FTDI cables are connected, press `"0"` and then **ENTER**.
6. Use the keyboard to control the galvanometer. A typical scan involves pressing `"1"` and `"2"`. The program will continue running until the count reaches the value of the `COUNT` variable specified by the user. Each keyboard input corresponds to each order as follows.
   - Button 1: Preload the buffer
   - Button 2: Stream
   - Button 3: Stop
   - Button 4: Ping
   - Button 7: Stream with Feedforward Control

     Feedforward Control: Feedforward Control is original algorithm of our interface to further minimize the galvanometer position error. Assuming the repetitive use of the same scan pattern, the pattern itself can be customized to account for errors induced by the galvanometer's response. Figure 7 illustrates the feedforward control algorithm implemented in our digital galvanometer system. Upon receiving the scan pattern from the desktop, the servo driver provides position feedback from the galvanometer. The error is then computed as the difference between the desired scan pattern and the actual position feedback. This error is subsequently added to the current scan pattern to generate an updated pattern, optimizing the system's performance.

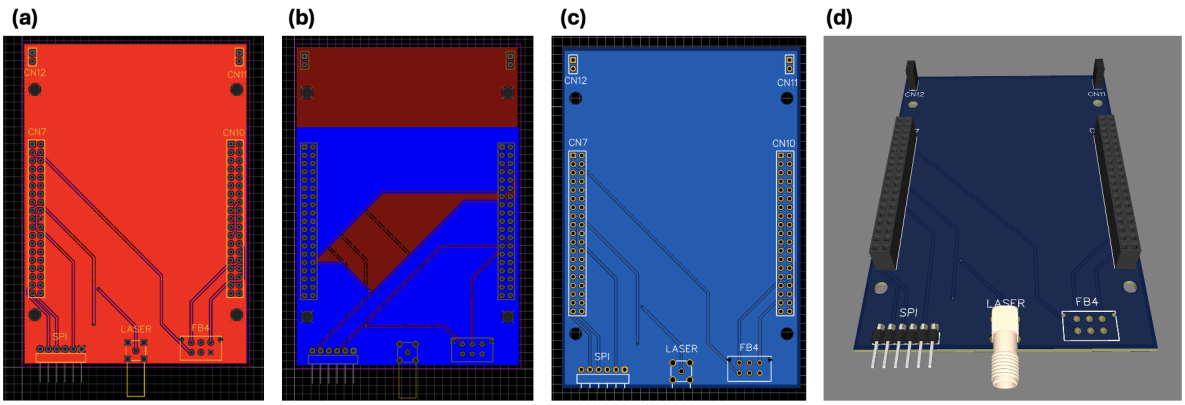

Figure 6: (a) Top-layer copper routing for the digital galvanometer interface PCB. (b) Combined top and bottom copper layers, illustrating ground (blue) and signal/power (red) planes. (c) Silkscreen layer view showing connector and label placement. (d) 3D rendered model of the fully assembled board with key headers and SMA connector for laser trigger.

Table 3: Bill of Materials

#	Part Number	Description	Qty	Unit Price ($)
1	SAM8856–ND	SMA-J-P-H-RA 50 ohm PCB jack (Samtec)	1	4.66
2	S7000–ND	2-pos 0.1 PCB female header (Sullins)	2	0.26
3	S7122–ND	38-pos 0.1 PCB gold header (Sullins)	2	2.05
4	732-5319–ND	6-pos 2.54 mm vertical header (Wurth)	1	0.31
5	43025-0600	Molex Micro-Fit 3.0 receptacle, 6 pos	1	0.48

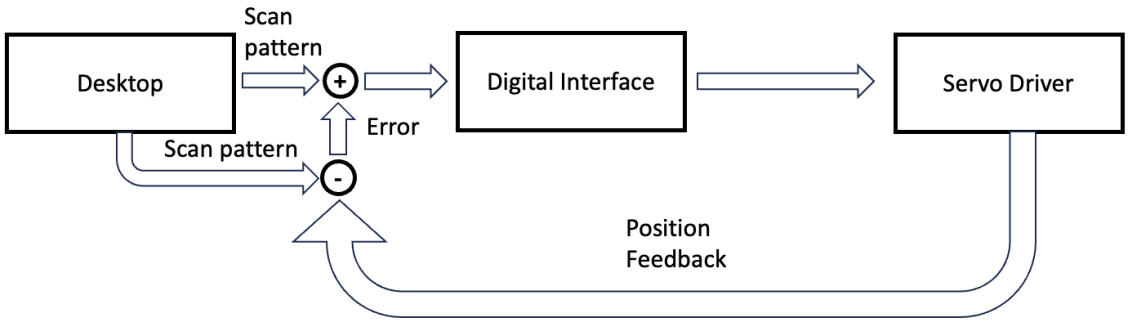

Figure 7: Overview of the Feedforward Control system: Scan pattern is updated by negatively adding the value of error calculated from previous scan pattern.

7. Initiate the OCT scan in Vortex. You can stream / stop / ping / stream with feedforward control solely using keyboard.

References

[1] M. Draelos, "Vortex – An open-source library for building real-time OCT engines in C++ or Python," 2022. [Online]. Available: https://www.vortex-oct.dev/

[2] FTDI, *AN_178 User Guide for LibMPSSE-SPI*, 2020, accessed: January 28, 2025. [Online]. Available: https://ftdichip.com/wp-content/uploads/2020/08/AN_178_User-Guide-for-LibMPSSE-SPI.pdf

[3] ScannerMAX. (2025) Scannermax software downloads and information. Accessed: January 28, 2025. [Online]. Available: https://scannermax.com/pages/software?srsltid=AfmBOop4pK1auB09N9fPVOuDXyOJJ64bFQdkP6SalPbLLaqEDNzurnbn

[4] STMicroelectronics, *STM32 Nucleo-64 boards (MB1136)*, https://www.st.com/resource/en/user_manual/um1724-stm32-nucleo64-boards-mb1136-stmicroelectronics.pdf, 2023, UM1724 User Manual, Revision 15. [Online]. Available: https://www.st.com/resource/en/user_manual/um1724-stm32-nucleo64-boards-mb1136-stmicroelectronics.pdf