DEVELOPMENT OF A MODULAR MICROCONTROLLER-BASED SCANNER FOR OPTICAL COHERENCE TOMOGRAPHY

by

Malcolm Brown

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF BACHELOR OF APPLIED SCIENCE

in the School of Engineering Science

SIMON FRASER UNIVERSITY

JUNE 2021

APPROVAL

Name: Malcolm Brown

Degree: Bachelor of Applied Science (Honours)

Title of Thesis: Development of a Modular Microcontroller-Based Scanner for Optical Coherence Tomography

Dr. Glenn Chapman, P.Eng
Director, School of Engineering Science

Examining Committee:

Dr. Pierre Lane, P.Eng (Technical Supervisor and Chair)
Associate Professor of Professional Practice, School of Engineering Science
Senior Scientist, BC Cancer Research Centre

Dr. Anita Tino, P.Eng (Academic Supervisor)
Lecturer, School of Engineering Science

Dr. Ash Parameswaran, P.Eng (Committee Member)
Professor, School of Engineering Science

Date Approved: June 7, 2021

Abstract

The purpose of this thesis was to develop a scanner for an existing optical coherence tomography (OCT) system at the BC Cancer Research Center. Work included adding features, reducing response times, increasing reliability, and modularizing the current imaging system's design. The system is normally used in fiber optic imaging using a Rotary Pullback Device (RPD) and other scanning geometries can be implemented with these improvements.

Scanning methods including the RPD generally involve two motors, which allow for scanning in two axes. This project's goal was to design and validate microcontroller (MCU) firmware for the scanner's motor control and peripheral systems. The MCU receives commands from the imaging system's main software and translates them to control the motors and command the rest of the system.

In the previous design, the scanner's RPD motors were controlled through software and a data acquisition board (DAQ), which generated signals to control the RPD motors and acquire sample data. However, this implementation was not ideal. This project replaces the DAQ's control of the scanner motors, leaving it to only acquire sample data.

Modularity was the main motivation for design changes, but there are other benefits. The RPD hardware requires prompt, reliable responses from software. Previously, if the imaging system Windows software crashed, it could damage hardware if the scanning motors continued to move unintentionally. Using a dedicated MCU allows for faster response times and if software fails, the MCU still functions as intended.

An MCU and peripherals were selected for this task, with the MCU firmware being the main development step. The firmware was developed to facilitate multiple lines of communication and replaces the existing interface while adding to existing functionality and robustness. Testing and validation of the design followed in order to ensure the device operates safely and reliably. These steps are intended to allow for implementation of a custom printed circuit board (PCB) in the project's future.

Table of Contents

Lists of Figures

List of Tables

List of Acronyms

ADC – Analog to Digital Converter

CCP – Compare/Capture/PWM

CV – Coefficient of Variation

DAQ – Data Acquisition Board

EEPROM – Electrically Erasable Programmable Read Only Memory

GPIO – General Purpose Input Output

FORJ – Fiber Optic Rotary Joint

HSM – Hierarchical State Machine

IC – Integrated Circuit

IR LED – Infrared Light Emitting Diode

ISR – Interrupt Service Routine

MCU – Microcontroller Unit

OCT – Optical Coherence Tomography

OC4 – Output Compare (fourth module)

ODV – Object Dictionary Value

PCB – Printed Circuit Board

PWM – Pulse Width Modulation

RPD – Rotary Pullback Device

SD – Standard Deviation

SPI – Serial Peripheral Interface

TCLKIA – Timer Clock Input (module A)

UART – Universal Asynchronous Receiver-Transmitter

1 Introduction

The goal of this thesis project was to upgrade an existing imaging system with an improved design, focusing on modularity and increased reliability. The previous system used a National Instruments Data Acquisition Board (DAQ) to scan images in two axes and acquire image data. This new design replaces the DAQ's scan functionality with a dedicated microcontroller unit (MCU), custom firmware, and other components. The new MCU not only allows more reliable control of the current Rotary Pullback Device (RPD) scanner, but also enables other scanning methods to be implemented in the future with its modular design.

A high-level block diagram of the old scanning system is shown below in Figure 1.

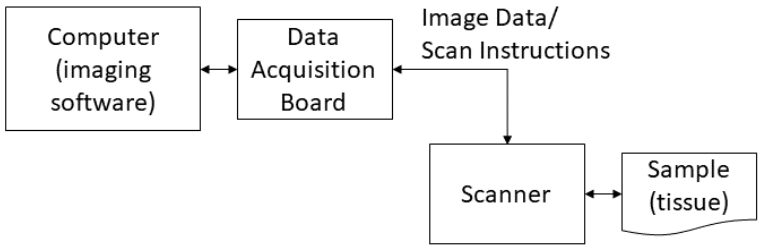


FIGURE 1. OLD SYSTEM BLOCK DIAGRAM

The imaging software communicates scan instructions to the DAQ, which uses those instructions to control the scanner and collect data at the same time. Redevelopment included separating the scanner into two parts as shown in Figure 2. Now the scanner control is built into the scanner itself and takes instructions directly from imaging software. The DAQ remains in the imaging system only to serve as an Analog to Digital Converter (ADC) collecting image data for processing in software.

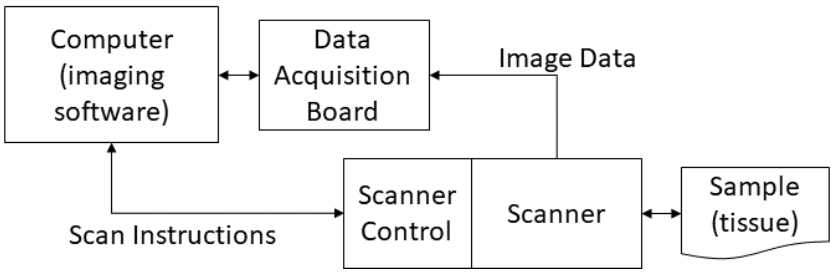


FIGURE 2. NEW SYSTEM BLOCK DIAGRAM

This new design allows the scanner block to function as the current RPD or other types of two-axis scanners. Modular implementation of various scanners is enabled by designing the scanner control block with a dedicated MCU, which takes commands from imaging software and translates them to appropriate signals used to control hardware for each type of scanner.

1.1 Requirements

The end goal of this thesis was to design a fully functional and tested scanning system. This involved writing firmware for a MCU that controls various peripherals at once and integrates them. This project works with an RPD, but the design also had to allow other scanning methods to be implemented easily without changing the framework of the rest of the system.

1.2 Scope and Contributions

Work involved in this thesis project consisted primarily of coding firmware for slow axis motor control. There was also work done to integrate hardware and firmware components to ensure everything would work together as intended. The existing RPD hardware was provided. This project involved replacing the DAQ as the RPD hardware controller with the microcontroller and firmware described within. The microcontroller is a Microchip PIC24F1024GB610 and included an Explorer 16/32 development board for pinout and debugging purposes.

The thesis proposal included collaboration with a third party to have the printed circuit board (PCB) designed. Due to time constraints, this step has not been reached and will have to be included in future work instead. All hardware functionalities can be implemented and tested without the custom PCB, but more work will be required to complete this final step in the future.

1.3 Scanning Geometries

Scanners work in general by sweeping a laser beam across the sample being imaged. For this type of imaging system, there is a slow axis and a fast axis in which the scanner moves the beam. They are termed as such because of the speed at which they scan along the axis. One fast axis sweep is performed for each step along the slow axis. The following sections outline three examples of scanners which could be controlled with this modular system: a 2D galvanometer, the RPD, and a micro-motor scanner.

1.3.1 2D Galvanometer Scanner

A 2D galvanometer scanner is illustrated in Figure 3 below [1]. Galvanometers scan two dimensions in a row-by-row fashion. This is also known as a raster scan and is commonly seen in a Cathode Ray Tube TV's scan pattern. The fast axis scan is controlled by Galvanometer Scanner 2, which sweeps across the range of ϕ_2 one time for each increment of ϕ_1 in the slow axis. When the entire range of ϕ_1 has been covered, a single scan of the sample is complete.

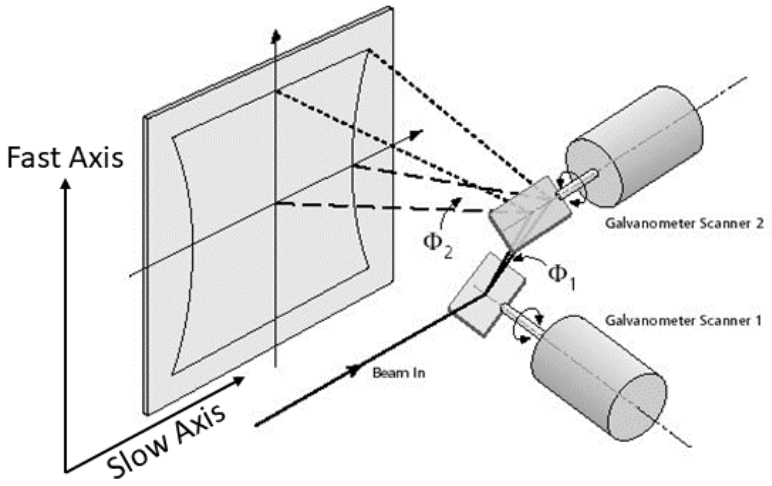


FIGURE 3. 2-AXIS GALVANOMETER

The 2D galvanometer is useful for acquiring images of 2D surfaces as we see them. However, the design is not suited to be miniaturized to the same scale as the following RPD and micro-motor scanners.

1.3.2 Rotary Pullback Device

Rotary-pullback scanners rotate an optical fiber in the fast axis (azimuthal direction) while "pulling back" the entire fiber in the linear slow axis. This can be thought of as tracing the inside of a cylinder as circular slices (fast axis) from top to bottom (slow axis). Again, one scan of the fast axis (rotation of the fiber) is performed for each step along the slow axis. The distance of each increment in the slow axis determines the slow axis resolution, while the sample rate of the DAQ and rotational velocity of the optical fiber determine fast axis resolution.

An example can be seen in Figure 4, where a laser is directed through the fiber and the light recaptured within the same fiber to create Optical Coherence Tomography (OCT) images. This produces a cylindrical scan, usually of lumenal tissue such as the inside of lung airways. Motors controlling rotation and pullback are mounted at the proximal end of the fiber nearest the laser source while imaged tissue is at the distal end. RPDs offer excellent miniaturization, with some being ~0.5 mm in diameter, allowing for imaging of very narrow passages. The window tube allows the fiber to spin without contacting the tissue itself.

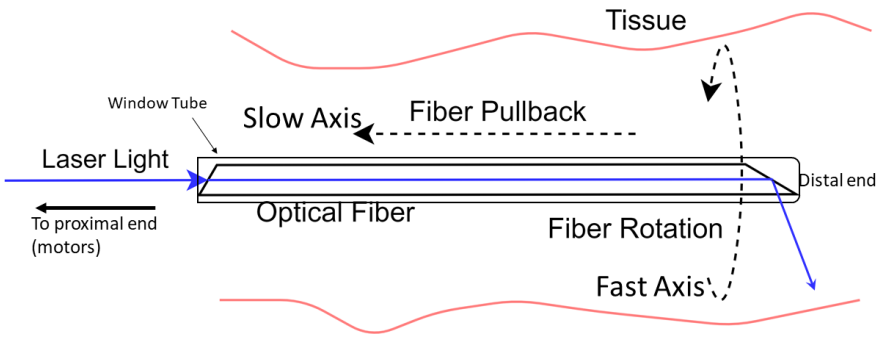


FIGURE 4. ROTARY PULLBACK DEVICE

1.3.3 Micro-motor Scanner

Micro-motor scanning seen in Figure 5 [2] below is similar to an RPD, but rotation in the fast axis (azimuthal scan) is done with a motor within the distal end of the fiber. This can solve some issues that come with an RPD, like non-uniform rotational velocity, but the minimum diameter of the scanning end is limited by the size of the micromotor. Also, this method requires active electrical components to be inside the sampled tissue, which is potentially less safe than the RPD.

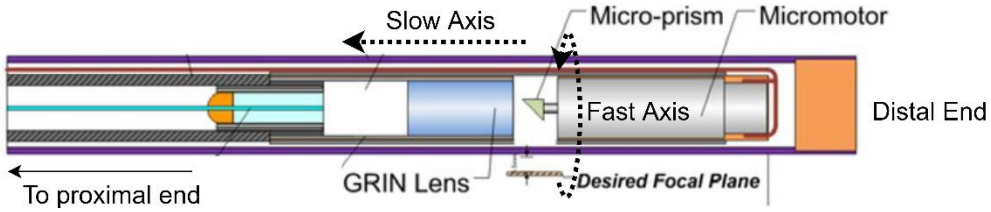


FIGURE 5. MICRO-MOTOR SCANNER

2 Design

Initial work on this project started during a Fall 2019 Co-op semester and included development of rotary fast-axis scanning, simple PC to MCU communication, and an electrically erasable programmable

read-only memory (EEPROM) module for storing configuration settings. This thesis continues development and validation of the MCU firmware, including:

1. Motor and limit switch operation for slow axis operation.
2. System integration using a state machine within the MCU.
3. Testing and validation of the entire system.

A block diagram of the proposed hardware system is shown in Figure 6. The main changes from the current system included replacing most DAQ board functionality with the MCU, an additional limit switch, and the EEPROM module. While this description is specific to the current system's RPD, fast and slow axis hardware could be replaced with various types of two-axis scanning methods with minor revisions.

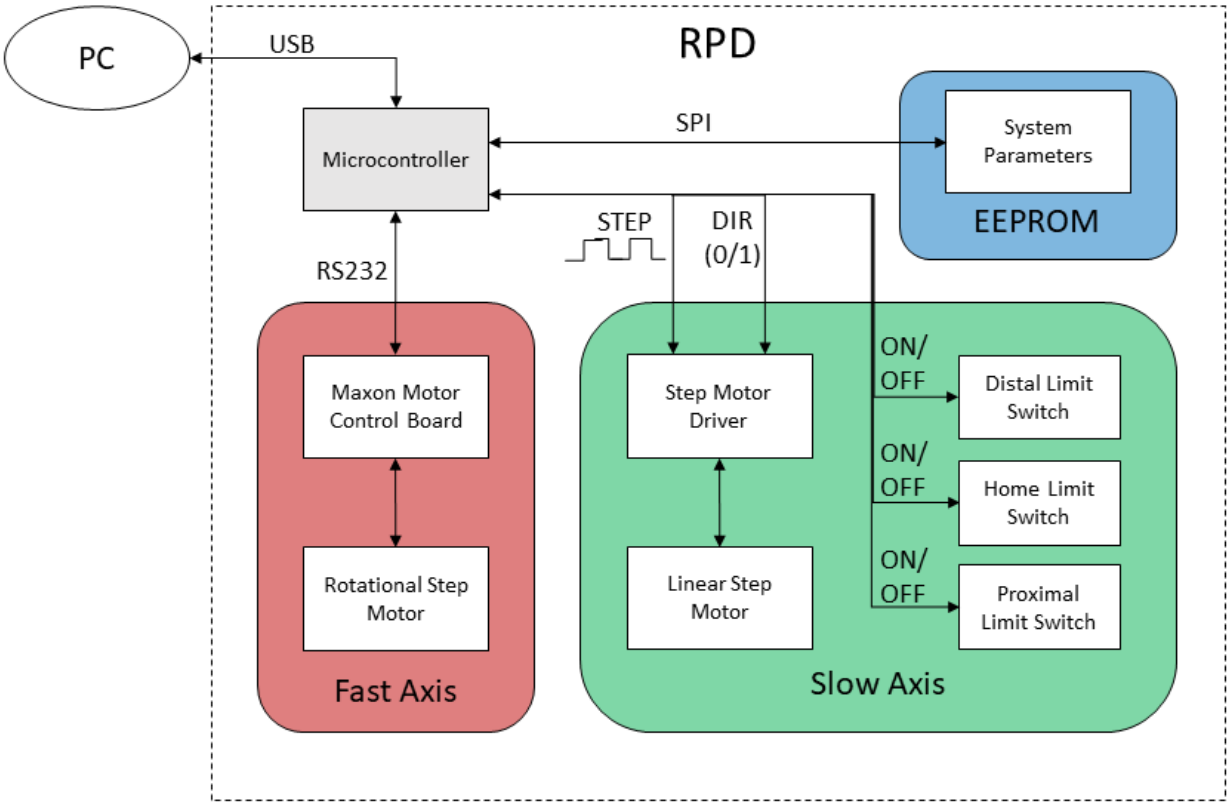


FIGURE 6. RPD HARDWARE BLOCK DIAGRAM

The MCU receives commands from the PC imaging software through a USB connection. Also, the MCU is responsible for processing these commands and controlling the rest of the system. Linear translation along the slow axis is done with a lead-screw step motor and motor driver. The motor's speed depends on the frequency of a square wave generated by the MCU and its direction is set with a binary signal. Distal and proximal limit switches are used to prevent over-ranging the slow axis. An additional home limit switch gives the axis a 'zero' reference point for starting each scan from. Fast axis rotation uses a precise Maxon step motor and control board, which receives commands via a RS232 serial

communication scheme. An EEPROM acts as a memory card, storing system parameters loaded each time the machine boots. The parameters are read or written over a Serial Peripheral Interface (SPI), which is another form of serial communication.

To integrate these functions, a state machine was designed to improve firmware writing, maintenance, and modularity. The state machine simplifies code by eliminating the need for nested conditional statements, which can become unwieldy and difficult to debug in complex applications.

Selection of hardware was not in the scope of this project because all but the MCU and EEPROM were being used in the DAQ-based imaging system. The hardware was all selected and provided by this undergrad thesis' supervisor Dr. Lane, who designed the original imaging system. The MCU chosen is from the Microchip brand (PIC24FJ1024GB610). A development board (Microchip Explorer 16/32 [3]) was also provided. This board offers many peripheral features for testing including 256kb of EEPROM, pinout for the 100-pin MCU, and mikroBus connections. The mikroBus provides pins for an RS232 Click [4], enabling serial communication. Fast axis scanning for the RPD uses a Maxon EPOS2 24/2 control board and EC-max step-motor [5]. The slow axis uses an AMP STR2 step motor driver, Haydon lead-screw motor and three optical switches.

2.1 State Machine Overview

The MCU is configured to run as a state machine. The device can only be in one state at a time and various inputs called 'triggers' cause the device to transition to other states:. For example, when the system is in the 'Idle' state shown in Figure 7, it is waiting for a new command from the imaging system software. When the 'Start Scan' command is received over USB, the MCU changes states to 'Scanning' and carries out the scanning process. Upon scan completion, the MCU itself initiates the 'Stop Scan' trigger and transitions itself back to the 'Idle' state until it is given a new command. These states and state transitions form the core logic of the MCU's firmware and function regardless of what happens to the imaging system software between commands.

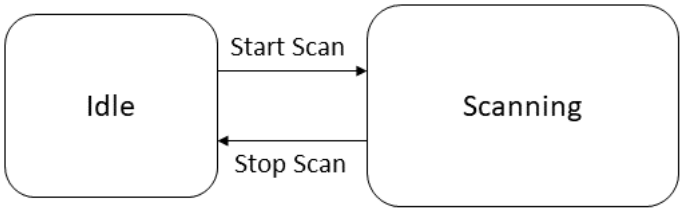


FIGURE 7. SIMPLIFIED STATE MACHINE

The MCU firmware in this project operates in a specific type of state machine called a Hierarchical State Machine (HSM). An HSM contains higher level states known as 'superstates' that can contain lower level 'substates'. As the machine transitions between any two states, functions called entry and exit actions are called. This is the basis for the HSM model and is illustrated in Figure 8 below.

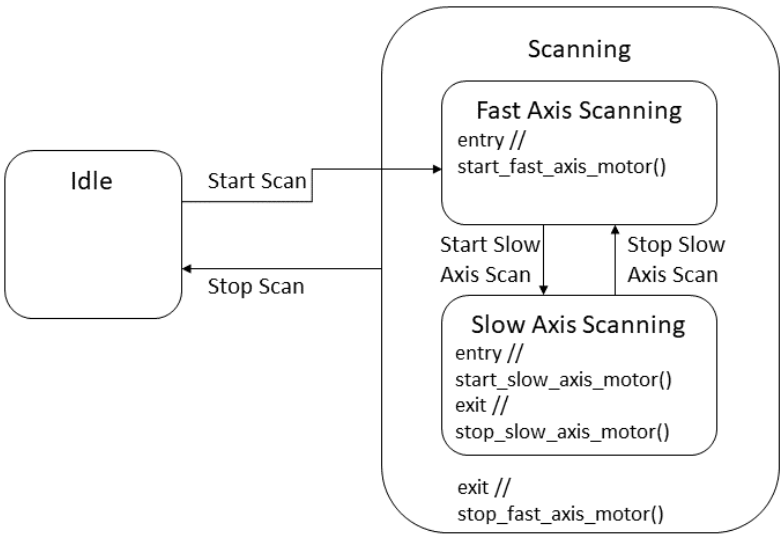


FIGURE 8. HIERARCHICAL STATE MACHINE

The figure shows a simplified version of the states, triggers, and entry/exit actions involved in a typical scan. When a trigger causes the machine to leave one state, that state's exit actions are called. Entering the next state then causes the new state's entry actions to be called. For example, when entering the 'Fast Axis Scanning' state in the HSM figure above, if the machine is within a substate such as 'Slow Axis Scanning' and the 'Stop Scan' trigger occurs, the 'Slow Axis Scanning' state's exit actions are called. Next, the superstate 'Scanning' exit actions are performed before the machine transitions back to the idle state. This HSM style ensures no essential actions such as stopping a motor are missed upon leaving a state.

State machine firmware is written in C and at its core, operates within a while loop. This allows new information sent and received by the device to be processed continuously. Real time systems can use flags and other dynamic variables to trigger actions and are useful in simple designs. However, they can become very cumbersome code to read and debug in complex programs where each new flag can necessitate another nested level of conditional logic. State machines are suitable solutions to this problem because they break the system down into finite states for the code to transition between.

For this device, Quantum Leap's QM Model-Based Design Tool (QM) was used to design the state machine. QM generates state machine code in C from a diagram similar to Figure 8 above, which is then imported into the MCU firmware's code.

2.2 Fast Axis

The firmware for fast axis control was developed during a Fall 2019 co-op semester, but still needed to be integrated with the rest of the system. The fast axis is responsible for rotating the optical fiber at a very specific velocity throughout the scanning process so that the image can be reconstructed from sample data accurately. The selected Maxon EC-Max step motor uses a Maxon EPOS 24/2 control board in the current RPD scanner and meets specifications for precise velocity control during imaging. Communication with the control board is done via RS232, a type of serial communication. An overview of the hardware for fast axis operation is shown in Figure 9 below.

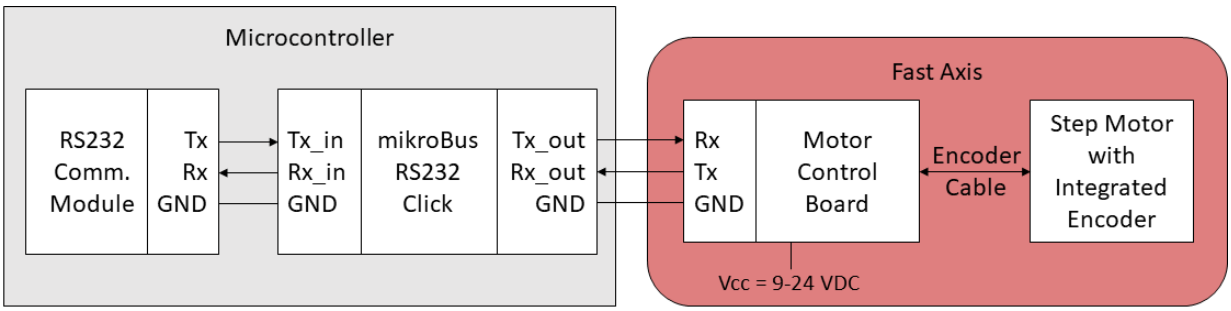


FIGURE 9. FAST AXIS OVERVIEW

2.3 Fast Axis Communication

As illustrated in Figure 9, the EPOS motor control board receives universal asynchronous receiver transmitter (UART) commands from the MCU via the mikroBus RS232 Click. The Click translates UART logic levels from the MCU to appropriate voltage levels for the motor control board. UART uses high and low logic levels to translate single characters into 8-bit binary sequences which are digitally transmittable. The previous work included programming the MCU firmware to send and receive specific commands compatible with the Maxon motor control board such as starting and stopping the motor along with many other controls.

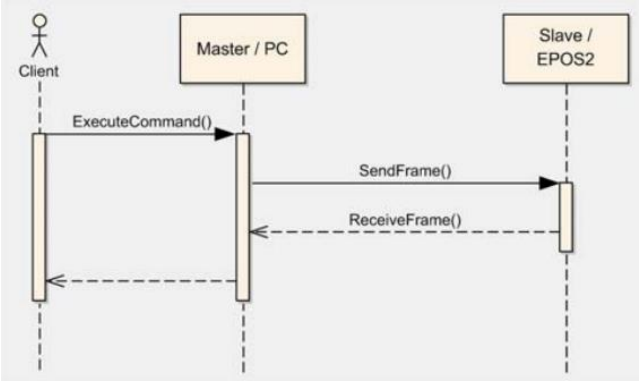
2.3.1 RS232 Click and mikroBus Connection

The mikroBus provides a pinout for the MCU which allows many different peripherals like the RS232 Click. The selected MCU, mikroBus, and the RS232 Click are all designed to work together. The RS232 Click, developed by mikroBus, simply snaps into place on the mikroBus pinout. The mikroBus headers will be on the PCB with the MCU when it is designed. The RS232 Click then has an RS232 socket which wires directly to the Maxon control board.

2.3.2 MCU to Maxon Control Board Communication

The basic communication scheme is described in Maxon's documentation [6] and is shown in Figure 10. Here, the imaging system software is the client, the MCU the master, and the control board the slave. Each frame of data is composed of data also seen in Figure 10. The Operational Code byte (OpCode) is sent first. This byte lets the control board know what type of command is being issued.

OpCodes used in this project allow the MCU to read and write object dictionary values (ODVs). The control board's object dictionary maintains the state of the device. Depending on what value is in each location, the motor will be controlled in different ways. For example, one of the key ODVs used to control the motor sets the movement velocity. By writing to the motor velocity ODV, the desired velocity of the motor can be set. Reading ODVs works similarly to writing, but is more important for things like reading the true real time velocity of the motor.



OpCode (8-bit)	Len-1 (8-bit)	Data[0] (16-bit)	…	Data[Len-1] (16-bit)	CRC 16-bit
HEADER		DATA			CRC

FIGURE 10. COMMUNICATION SCHEME (TOP), FRAME FORMAT (BOTTOM)

After the OpCode, the number of bytes to be written is sent along with the data. At the end of each frame, a CRC checksum is sent. The checksum is two bytes which are a value calculated based on the rest of the bytes composing the frame. The MCU and control board both calculate the checksum. If it does not match, a transmission error has occurred. The Microchip MCU includes a CRC module capable of CRC generation for a few different algorithms. In this case, the EPOS motor control board requires CCITT-16, which is a 16-bit CRC standard defined by the Consultative Committee for International Telephony and Telegraphy.

The imaging system software already uses these commands provided by Maxon's C library and transmits to the control board through USB. The library written for this project translates any commands used in the previous system so they may be used by the MCU. Commands include reading and writing ODVs, starting motor movement, and setting the motor's velocity. Other commands can be seen in a brief API in Appendix A.

2.4 Slow Axis

The slow axis is used to pull the distal end of the optical fiber proximally along the slow axis while scanning. In other words, the slow axis controls the length of the cylindrical scan. Hardware used in the slow axis includes an AMP STR2 step motor driver, Haydon Inc. AMP 3540M lead-screw step motor, and limit switches along the slow axis pullback path as shown in Figure 11.

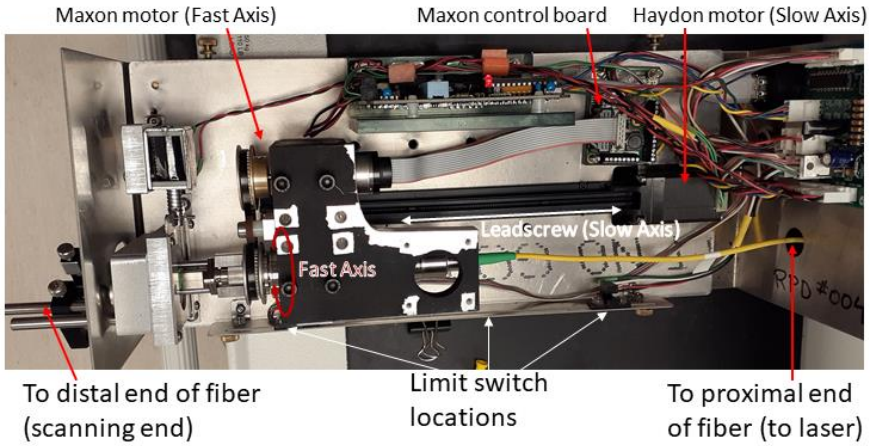


FIGURE 11. RPD VIEW FROM ABOVE

The motor's rotational velocity is translated to linear movement with a leadscrew system. The screw can be seen extending from the Haydon motor body out to the left along the slow axis. As it rotates, the screw pushes or pulls the stage holding the fast axis motor and fiber optic rotary joint (FORJ). The FORJ allows the distal end of the fiber to be rotated by the Maxon motor while the proximal end leading to the laser stays still. As the stage moves along the slow axis, the distal end of the fiber also moves. This controls the fiber's pullback motion while collecting data for each scan.

2.4.1 PWM and Compare Modules

The Maxon motor control board has a built-in encoder which manages velocity, position, and many other parameters internally for precise fast axis rotation. However, the slow axis motor driver is much simpler, so the MCU has to manage those responsibilities itself. A block diagram of the slow axis is shown here in Figure 12:

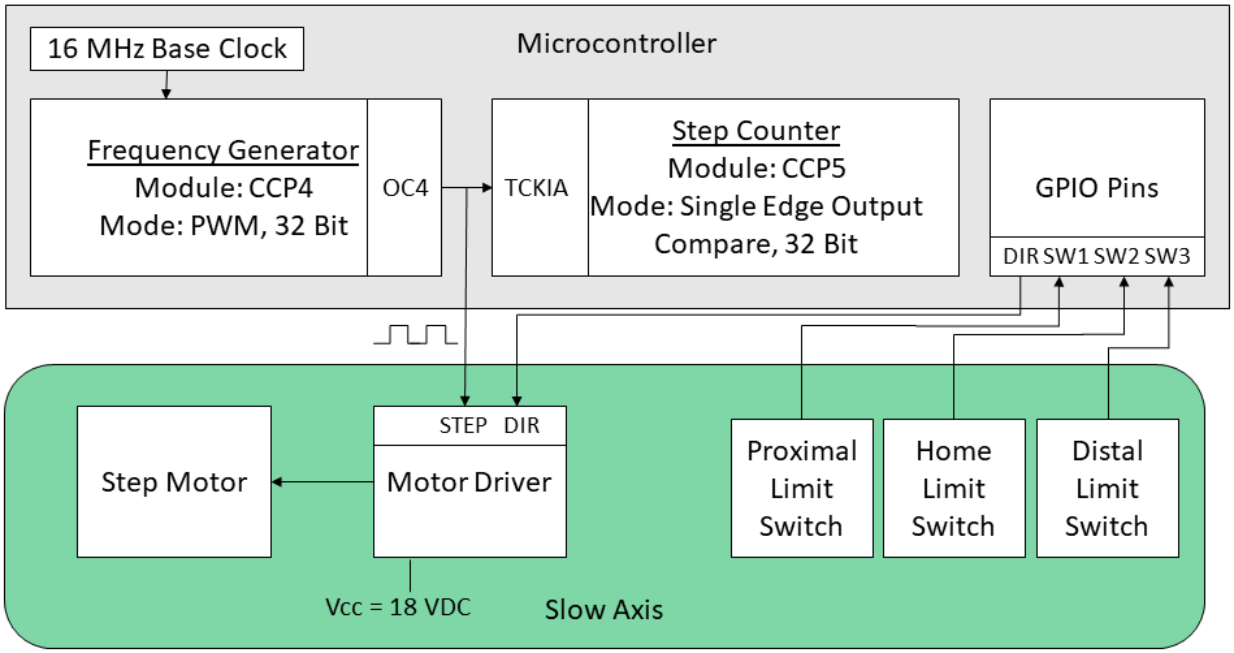


FIGURE 12. SLOW AXIS OVERVIEW

The Microchip MCU has several Capture/Compare/PWM (CCP) peripheral modules, two of which were used here to implement the slow axis drive. They can be configured in a variety of ways, but this project uses the Compare and Pulse Width Modulation (PWM) modes. The Frequency Generator block is configured in PWM mode, allowing it to generate a PWM signal used as the STEP signal in the motor driver. The Step Counter block keeps track of the number of steps sent to the motor driver to control the distance traveled and current position along the fast axis for each pullback.

The Frequency Generator block sends a square wave into the motor driver's STEP pin with a specific frequency required to move the slow axis motor at any speed dictated by the imaging software. To accomplish this, it uses a 16-bit counter that increments for every rising edge of the MCU's 16MHz base clock. The counter is compared after each increment to another 16-bit register with a set value. When the registers match, the frequency generator pin alternates between 0 and 1 and resets the counter to zero, thereby forming a square wave. The step motor moves 0.00127mm per pulse of the square wave.

Therefore, to move at a given speed, the required frequency is found using the formula: $f = \frac{v}{0.00127}$, where v is the desired speed in mm/s.

This square wave is also sent from OC4 (output compare #4) to be used as signal source TCLKIA (Timer Clock Input A) in the Step Counter block. Here, a 32-bit register tracks the slow-axis position throughout each scan. For every step sent to the motor driver, the register increments as well. The value in the register then corresponds to the distance traveled along the slow axis and is used to track the position of the stage shown in Figure 11. To define a repeatable zero position, a homing routine is used which sets the 32-bit position register at zero on the home limit switch. The microcontroller's documentation refers to and defines these pins (OC4, TCLKIA) along with many others specific to this specific controller [7].

A general purpose in/out (GPIO) pin is used to set the direction of the lead-screw motor. When the pin is set to 0, the motor moves the stage forward and the step counter increments. When the pin is set to zero, the motor moves back and the counter decrements.

Decrementation is not normally possible with the CCP modules in this MCU. To decrement the count, the counter register is first read, then rewritten with its one's complement value. It then can be incremented as usual, which effectively decrements the count when read back as the one's complement again.

2.4.2 Limit Switches

General purpose input output (GPIO) pins in the MCU are used to receive single digit binary signals from the three limit switches. The three optical limit switches are places along the slow axis. The home limit switch is used to zero the step counter, providing a reference point so that each scan can start from the exact same position. The distal and proximal switches are used to ensure the stage does not go beyond the intended slow axis range, which could damage the hardware.

When a proximal or distal limit switch is triggered, the binary value read by the MCU changes and is read during an Interrupt Service Routine (ISR). ISRs cause the MCU to stop whatever it was doing and execute ISR code. So, when a limit switch is triggered, the MCU can stop any movement immediately to prevent potential damage to equipment. The ISR is configured to run on a timer which triggers once every millisecond.

One of the optical limit switches used is shown in Figure 13 below. An infrared light emitting diode (IR LED) produces light which is detected on the other side of gap seen in the figure. A tab moves along the slow axis with the scanning fiber and moves into the gap, occluding the light from the IR LED. The detector's voltage is inversely proportional to the amount of IR light detected, so as the tab blocks out light, the voltage rises.

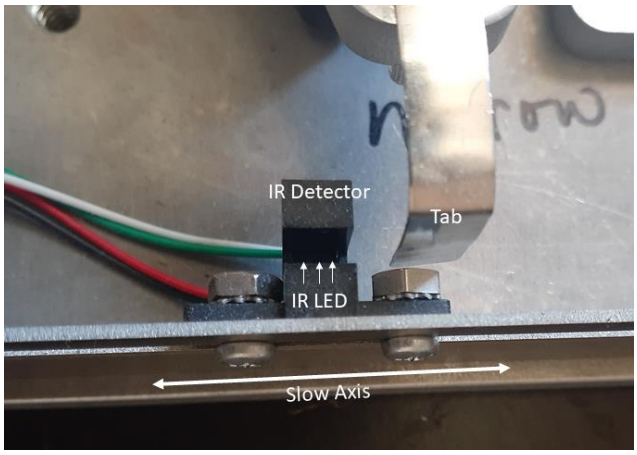


FIGURE 13. LIMIT SWITCH HARDWARE

Due to the width of the tab and gap width in the limit switch, there is an active range in the slow axis where the switch is considered 'on' as shown in Figure 14 with the layout of the three switches along the slow axis. This range is important in RPD operations discussed in a later chapter.

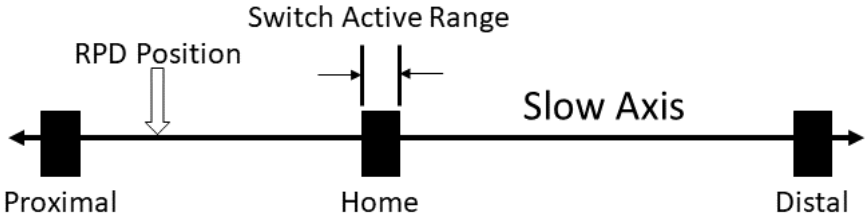


FIGURE 14. LAYOUT OF LIMIT SWITCHES ON SLOW AXIS

2.5 Persistent Memory

Memory for various start-up parameters such as movement mode, velocity, and many others has been implemented in the form of an EEPROM integrated circuit (IC). The EEPROM communicates over a Serial Peripheral Interface (SPI), which has been configured for the MCU in this project. Figure 15 shows a schematic of the hardware configuration of the system built so far.

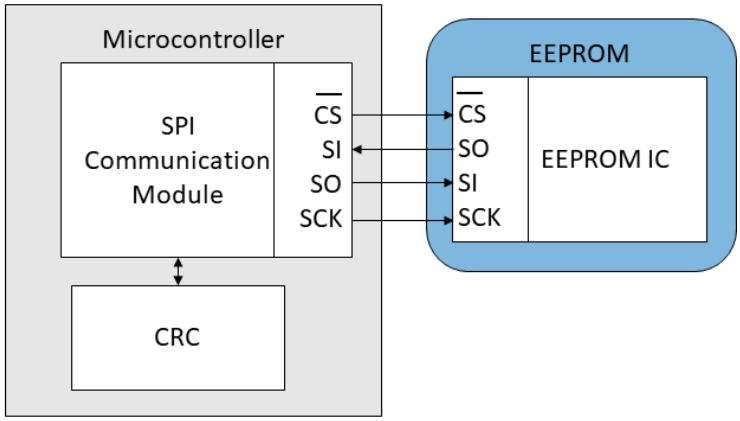


FIGURE 15. EEPROM OVERVIEW

Data can be written or read on the MCU side over the SO and SI pins, respectively. The SCK line is a clock the MCU provides to time reading and writing data. CS_not is used to “activate” the EEPROM chip. When the line is high, data can be read and written to the EEPROM.

The communication protocol for this EEPROM is similar to read/write functions used with the Maxon control board. Read and write OpCodes are first sent to the EEPROM. Bytes to be written or read are sent next, depending on the OpCode. To ensure loading and storing of parameters is successful, a CRC is used for each operation. The MCU contains only one CRC generation module, so the CCITT-16 algorithm required for communication with the EPOS motor control board is used here as well.

3 System Integration

System integration is done within the MCU using a previously described HSM. A high-level diagram of the state machine used is shown here in Figure 16:

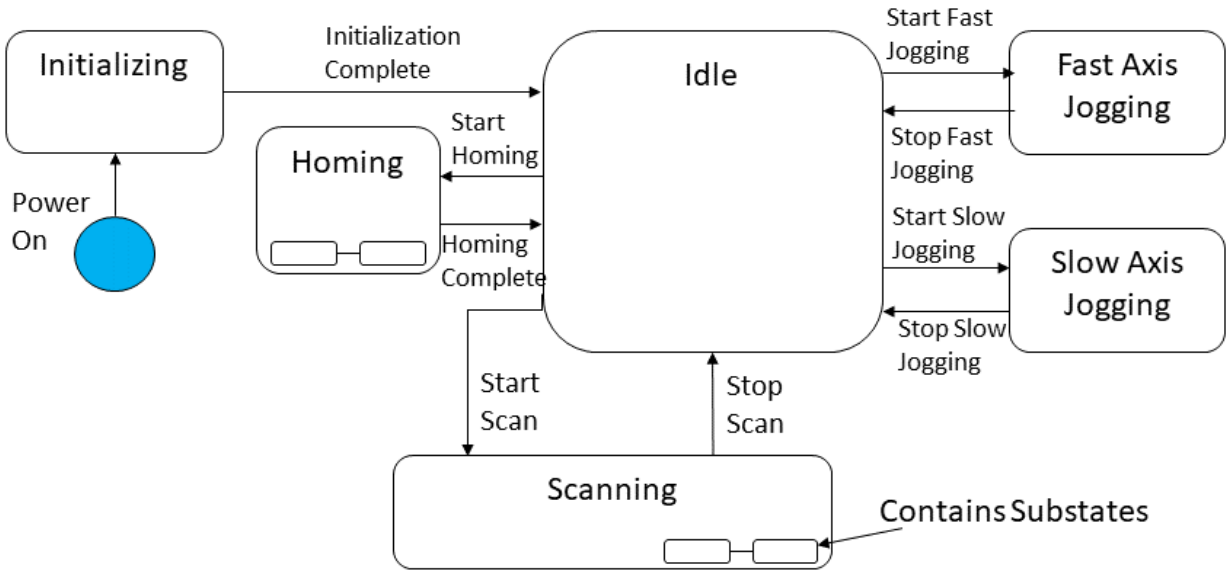


FIGURE 16. HSM OVERVIEW

The system takes care of initialization steps after powering on including connecting to the PC over USB, loading parameters from EEPROM and other tasks. When idle, the jogging states can be entered temporarily to move the probe to any desired position in the slow or fast axes. Scanning involves bringing the fast axis motor up to speed, followed by pullback in the slow axis while the DAQ acquires image data. A summary of these states is shown in Table 1.

State	Description
Initializing	MCU loads parameters from EEPROM, starts connecting with imaging system and other start-up tasks.
Idle	The state the machine is in when waiting for commands. No motor movement.
Jogging	These allow the machine to move forward/backward in the slow axis or rotate in the fast axis for calibration.
Homing	Used to zero the RPD so each scan starts in the same position.
Scanning	Used to perform pullback while the imaging system acquires data.

TABLE 1. HIGH LEVEL STATES

To ensure the system works properly after integration, several tests were carried out to measure important RPD movement parameters during the homing and scanning states. These parameters included RPD speed and positions throughout the homing and scanning processes. In the future, more work on the PCB discussed earlier will be required including further testing of the final system.

3.1 Imaging System Commands

The imaging system software ultimately commands the RPD by sending triggers to be processed by the state machine. It tells the MCU which operations to perform along with relevant parameters. USB communication is used to transmit data and the MCU acts as device to the imaging software host. The host determines when to initiate a command and sends its own commands to the MCU for various tasks. Primary commands expected from the imaging system are seen in Table 2.

Command	Description
Start Scan	Starts rotation of the fast axis motor to ensure required fast axis velocity is met before starting a scan
Scan	Performs a scan, moving the slow axis motor a distance and at a velocity commanded by the imaging software
Home	Commands the RPD to automatically find the home position and reset the step counter there.
Fast Axis Jog	Rotates the fiber in the fast axis at a velocity determined by the imaging system.
Slow Axis Jog	Moves the fiber forward or backward.

TABLE 2. IMAGING SYSTEM COMMANDS

3.1.1 Homing

Homing is used to reset the slow axis position to precisely the same place. This is important for imaging the same sample multiple times. The Homing superstate contains several substates and a logical condition to determine what movement is required. The Homing state diagram and description of substates are shown in Figure 17 and Table 3 below.

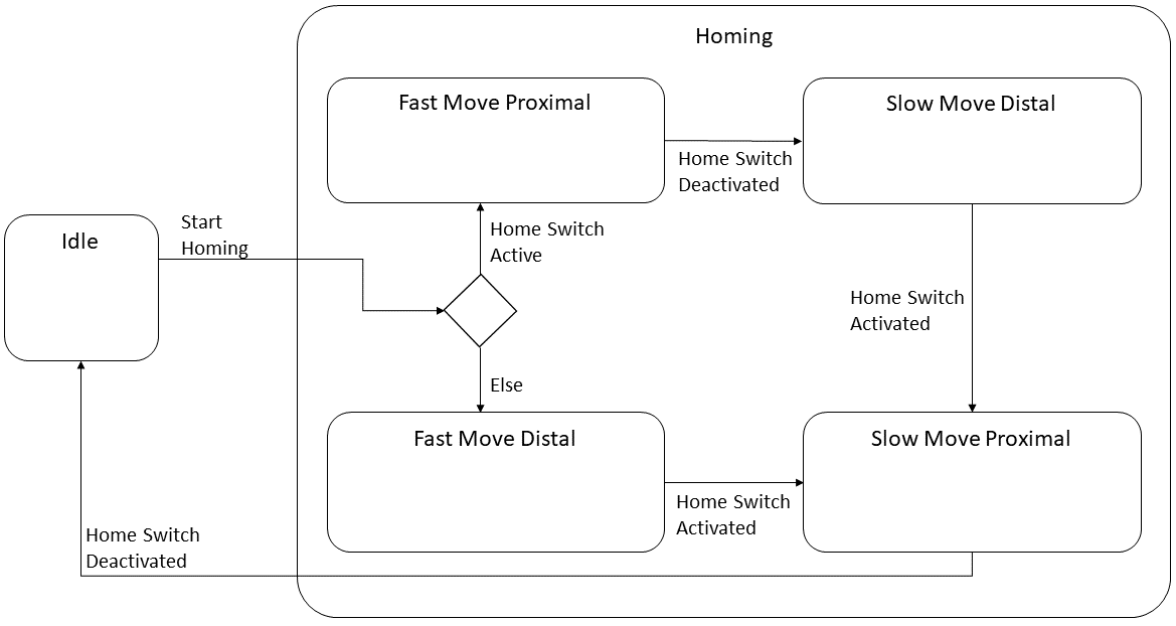


FIGURE 17. HOMING SUPERSTATE

Homing moves through a series of substates shown in Table 3 that depend on the RPD's position on the slow axis when homing is initiated. These steps are required to find the same home position for each homing sequence due to the limit switch active range discussed earlier. During homing, the RPD moves along the slow axis quickly at first to find the first point of reference. It then moves slowly to zero in on the same position for each homing sequence. Slow movement in the final step gives the MCU more time to read the status of the home limit switch.

State	Actions
Fast Move Proximal	Moves the RPD proximally (towards laser) in the slow axis at a high speed.
Fast Move Distal	Moves the RPD distally (towards sample) in the slow axis at a high speed.
Slow Move Distal	Moves the RPD distally in the slow axis at a low speed.
Slow Move Proximal	Moves the RPD proximally in the slow axis at a low speed.

TABLE 3. HOMING SUBSTATES

The sequence of steps for each condition of the homing routine is summarized in Figure 18. The goal is to carry out the final step of each sequence in the exact same way to minimize the difference in final position regardless of starting position. In the Home Switch Inactive case, the RPD could start distally or proximally to the home switch. The slow axis step counter always tracks position, so it is checked to determine which side the RPD starts on. In either situation, the first step is to move quickly towards the home position. The following steps will always be the same.

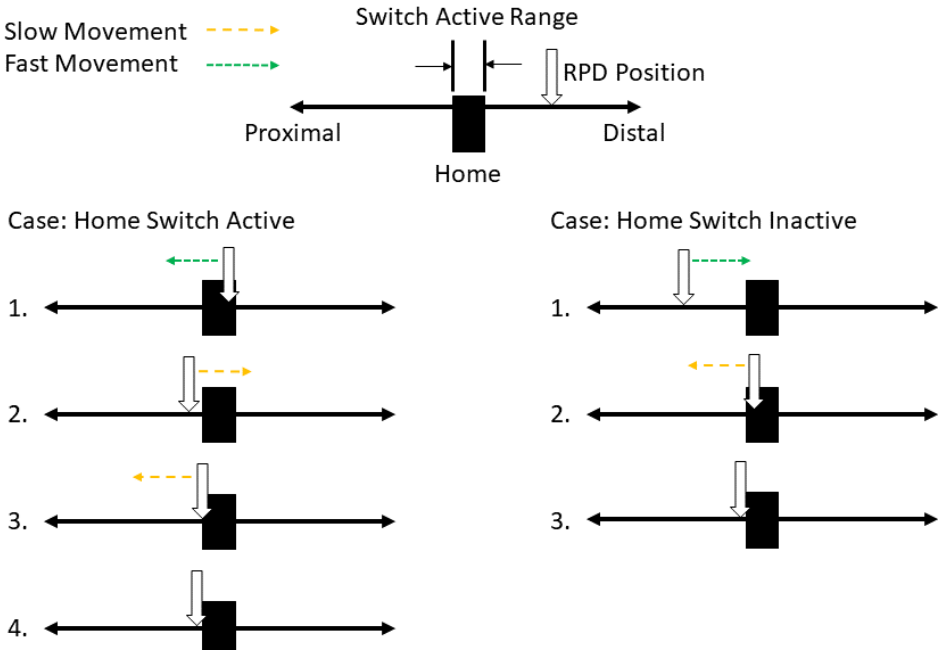


FIGURE 18. HOMING ROUTINE CASE SEQUENCES

3.1.2 Scanning

A detailed state diagram corresponding to the 'Scanning' superstate is shown in Figure 19:

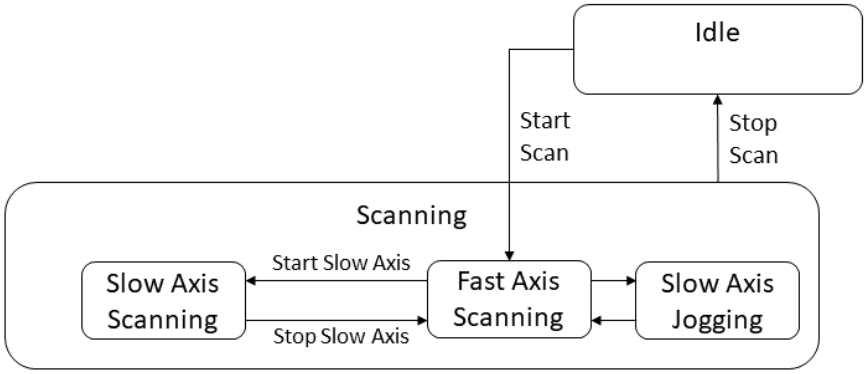


FIGURE 19. SCANNING SUPERSTATE

From the Idle state, a 'Start Scan' trigger sent by the imaging system software causes the RPD to transition to start scanning in the fast axis. The Maxon motor rotates the optical fiber, and a step signal is sourced from the laser used for image acquisition. The motor's velocity is controlled by the frequency of this signal. The RPD then waits for a trigger to start scanning. The slow axis motor then pulls the fiber back along the sample as the imaging system acquires samples. Once complete, the slow axis motor moves the fiber back to the home position.

3.2 Validation

Proving validity of the design involved running the RPD through several tests similar to the current applications of the imaging system. The device underwent multiple scanning and homing routines where the expected and measured distances were compared to ensure the scanner was operating as expected. Measurements were performed with a linear digital caliper with 0.01mm precision, positioned between a fixed part of the RPD frame and the linear axis stage as shown in Figure 20. The RPD is in the home position and the measurements D1 and D2 correspond to the home distance and pullback distance, respectively.

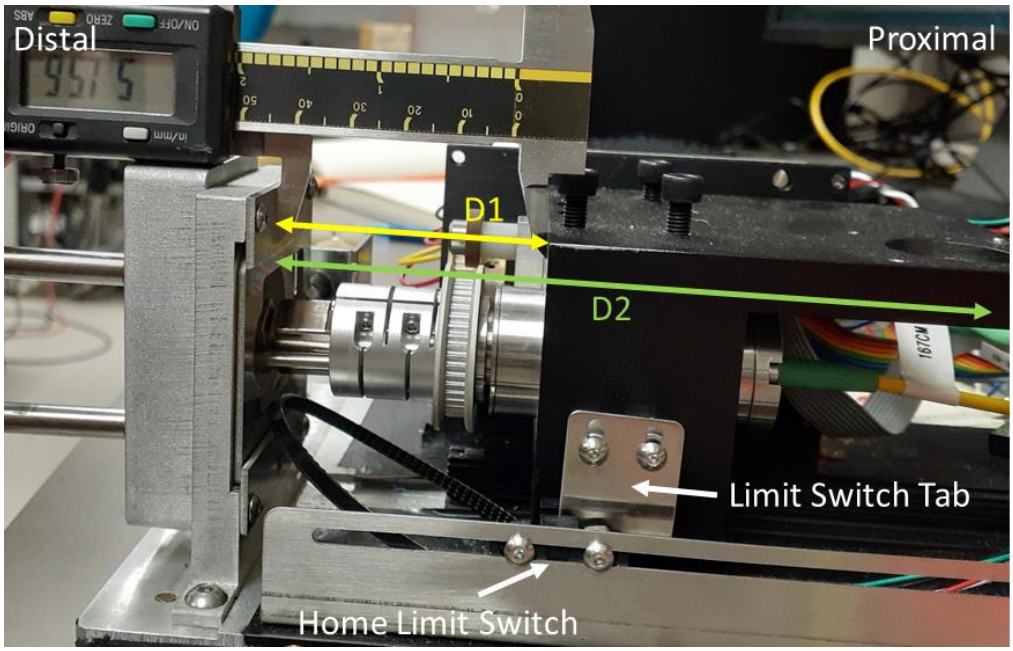


FIGURE 20. MEASUREMENT METHOD

When measuring the same distance D1 multiple times without any slow axis movement, it was found that any given measurement could vary by ±0.05mm due to measurement error. The results from the homing and scanning tests are shown in the following sections.

3.2.1 Homing

Homing tests were started with the limit switch tab on the proximal side of the home switch for the inactive home switch condition and at random points with the tab blocking the home limit switch for the active home switch case. The distance D1 was measured once after each homing routine and compared to the average.

The most important parameter in this test was the repeatability of arriving at the same final position once complete, or having the difference between the average starting position and D1 be as close to zero as possible. The results are shown in Table 4 along with the average, standard deviation (SD), and coefficient of variation (CV).

Test #	D1 (mm)	μ – D1 (mm)
1	51.62	-0.013
2	51.58	0.027
3	51.59	0.017
4	51.64	-0.033
5	51.60	0.007
6	51.61	-0.003
Avg	μ = 51.607	N/A
SD	σ = 0.0216	N/A
CV (σ/μ)	0.042%	N/A

TABLE 4. HOMING TEST

3.2.2 Scanning

The first scanning test measured the overall distance traveled from the starting point to the return position during pullback (D2 - D1). The RPD pullback distance was set to 40 mm with pullback speed 5 mm/s. Results can be seen in Table 5.

Test #	D1 (mm)	D2 (mm)	L = D2 - D1 (mm)	μ_L - L (mm)	μ_{D1} - D1 (mm)
1	51.69	91.75	40.06	-0.046	0.022
2	51.71	91.74	40.03	-0.016	0.002
3	51.69	91.70	40.01	0.004	0.022
4	51.74	91.72	39.98	0.034	-0.028
5	51.73	91.72	39.99	0.024	-0.018
Avg	μ_{D1} = 51.712	μ_{D2} = 91.726	μ_L = 40.046	N/A	N/A
SD	σ_{D1} = 0.0228	σ_{D2} = 0.0195	0.0321	N/A	N/A
CV (σ/μ)	0.044%	0.021%	0.080%	N/A	N/A

TABLE 5. PULLBACK DISTANCE AND RETURN POSITION TEST

The second test measured the average speed during pullback using a stopwatch. Pullback speed was set to 5, 10, and 20 mm/s. These tests were performed separately from the pullback distance tests to make measurement easier to manage. The results are shown in Tables 6, 7, and 8 below.

Test #	Measured Time (s)	Speed (mm/s)
1	8.38	4.77
2	8.42	4.75
3	8.43	4.74
4	8.42	4.75
5	8.37	4.78
Avg	μ = 8.404	μ = 4.758
SD	σ = 0.024	σ = 0.015

TABLE 6. PULLBACK SPEED TEST (40MM @ 5 MILLIMETERS PER SECOND)

Test #	Measured Time (s)	Speed (mm/s)
1	4.44	9.01
2	4.40	9.10
3	4.45	8.99
4	4.42	9.05
5	4.37	9.15
Avg	μ = 4.416	μ = 9.060
SD	σ = 0.029	σ = 0.059

TABLE 7. PULLBACK SPEED TEST (40MM @ 10 MILLIMETERS PER SECOND)

Test #	Measured Time (s)	Speed (mm/s)
1	2.42	16.53
2	2.42	16.53
3	2.36	16.95
4	2.42	16.53
5	2.36	16.95
Avg	μ = 2.396	μ = 16.698
SD	σ = 0.029	σ = 0.206

TABLE 8. PULLBACK SPEED TEST (40MM @ 20 MILLIMETERS PER SECOND)

Owing to the deviation of the measured speeds from those expected, a second set of speed measurements were performed using an internal timer on the MCU. The timer increments a 32 bit register at 16Mhz and the register value was taken just before and after each scan. The difference is then divided by 16MHz to find the total time per scan. Two sets of measurements were taken at 5 and 10 mm/s with a distance of 40mm for this test and the results are shown in Tables 9 and 10. The timer runs continuously, so the start and end counts are dependent on when the test is started after powering on the MCU.

Test #	Start Count	End Count	End - Start	Time (s)	Speed (mm/s)
1	58,575,557	188,747,953	130,172,396	8.135775	4.92
2	44,991,572	175,163,975	130,172,403	8.135775	4.92
3	38,895,557	169,067,953	130,172,396	8.135775	4.92
4	55,407,585	185,579,983	130,172,398	8.135775	4.92
5	67,311,572	197,483,975	130,172,403	8.135775	4.92
Avg	N/A	N/A	N/A	μ = 8.135775 ,	μ = 4.92
SD	N/A	N/A	N/A	σ = 0	σ = 0

TABLE 9. INTERNAL TIMER TEST (40MM @ 5 MILLIMETERS PER SECOND)

Test #	Start Count	End Count	End - Start	Time (s)	Speed (mm/s)
1	2,107,912,249	2,174,078,558	66,166,309	4.135394	9.67
2	564,367,585	630,534,017	66,166,432	4.135402	9.67
3	2,245,359,545	2,311,525,854	66,166,309	4.135394	9.67
4	97,631,572	163,798,011	66,166,439	4.135402	9.67
5	1,250,335,557	1,316,501,993	66,166,436	4.135402	9.67
Avg	N/A	N/A	N/A	$\mu = 4.135399$	$\mu = 9.67$
SD	N/A	N/A	N/A	$\sigma = 3.919E\text{-}6$	$\sigma = 0$

TABLE 10. INTERNAL TIMER TEST (40MM @ 10 MILLIMETERS PER SECOND)

3.2.3 Discussion

During the homing measurement tests, the maximum displacement from the average home position (μ – D1) was 0.033mm. The CV of all displacements was 0.042%, indicating very little variation between measurements. When practicing the measurements, there was an error range of ±0.05mm when measuring D1. Therefore, it is not possible to tell if the error is due to machine or measurement error.

Distance measurements during the scanning tests yielded similar results. The intended 40mm pullback distance (D2-D1) was measured with a maximum 0.06mm overshoot, with all values falling in a -0.02 to 0.06mm range of the desired distance.

The distance measurements were all very close to expected ranges, especially when accounting for the difficulty in acquiring measurement values accurately using the digital caliper.

Speed tests for scanning were first performed with a stopwatch and therefore were very susceptible to user error. The average time exceeding all expected stop-watch measurements was 0.405 seconds, with a standard deviation of 0.029 seconds.

More timing tests were performed using the MCU's internal timer and were also inconclusive. At both 5 and 10 mm/s rates, the measured time was off by around 0.136 seconds, which was surprising using this method.

Some extra time is expected as the count was started before and after a couple of operations required to start and end a scan, but over one hundred milliseconds is significant. The MCU had to be operated in a debug mode to acquire the timer's register values, so it is possible that operations took longer than usual. The additional time could indicate that the slow axis frequency generator is sending too many steps to the motor drive, but does not appear to be the case.

If the frequency generator ran for 0.136 seconds too long at 5 mm/s as indicated in Table 9, the pullback would be expected to move 0.68mm farther than intended. The pullback distance test in Table 5 contradicts this though, showing that the average pullback traveled only 0.046mm farther than expected. Therefore, it is more likely that the timing errors are mostly due to reaction time and measurement methods.

One other very small source of pullback speed error is due to MCU limitations. The slow axis signal generator's frequency is calculated in firmware with some rounding. For example, to move at 10mm/s, the slow axis motor driver requires a signal of exactly 7874.015748Hz to move the motor at a rate of 10mm/s. However, 7874Hz is used instead because floating point calculations take a significant amount

of processing time in the MCU. This rounded frequency moves the slow axis at 9.99998m/s, which is negligible for the requirements of the RPD system.

This rounding error will be similar when aiming for exact pullback distances as the step counter also only deals with integer values. The exact error will depend on the amount rounded. In the worst case, 0.5 steps will be rounded off. Each step moves the linear axis 0.00127mm, so the maximum error in distance due to firmware limitations will be 635nm. It is also possible that some step signals are missed by the motor driver, but this is not detectable with the existing test setup. Missed steps could add up over time, but the homing routine allows the system to reset back at a reference point to zero out potential miscounts.

3.3 Future Work

Once the firmware is complete to the point that no other modules or pins may be required in the future, steps will be taken to implement everything in hardware. This will involve creating an in-depth schematic which can communicate the requirements of a PCB to a Vancouver-based design company.

Information will need to be conveyed regarding maximum current ratings, MCU pins required, line traces, and more. At that point, the designer will take over and a PCB prototype will be manufactured.

Once all hardware has been acquired, the final stages of testing can begin. This will involve running the setup through tests to ensure that everything works as expected. Undoubtedly there will be bugs to fix, but the MCU firmware will always be able to be updated. The intention is that any issues will be able to be fixed with firmware only rather than updating the PCB.

Conclusion

The results of this project are steps towards a reworked scanner control that improves upon the existing RPD system's design and provides options for other scanning methods. The modular aspect is important in allowing other scanner types to easily replace the RPD.

Testing and debugging of the system was important to ensure the final product worked as intended and can be operated safely, without putting patients or operators at risk. Implementation of a state machine simplified this process by breaking the MCU firmware down into finite states and repeatable action sequences.

More work is needed in the future to have a custom PCB designed for the system. Once the hardware is in place, more testing and debugging will be required to further validate the design before using it in studies.

4 References

[1] “galvanometer scanner” *diy cup*, 07-Oct-2015. [Online]. Available: https://m.blog.naver.com/PostView.nhn?blogId=wanghoosoo&logNo=220502250621&proxy. (Accessed: Mar. 26, 2020).

[2] Proceedings Volume 8927, Endoscopic Microscopy IX; and Optical Techniques in Pulmonary Medicine; 89270T (2014) https://doi-org.proxy.lib.sfu.ca/10.1117/12.2040417

[3] Microchip Technology Inc. *Explorer 16/32 Development Kit Documents and Software* https://www.microchip.com/DevelopmentTools/ProductDetails/dm240001-2 (Accessed: Apr. 17, 2020).

[4] “RS232 CLICK.” mikroe.com, https://www.mikroe.com/rs232-click (Accessed: Apr. 17, 2020).

[5] “Maxon EC-Max.” maxongroup.com, https://www.maxongroup.com/maxon/view/content/ec-max-motors (Accessed: Apr. 17, 2020).

[6] Maxon Motor AG, Sachseln, Switzerland. *EPOS2 Communication Guide.* (2016). [Online]. Available: https://www.maxongroup.com/medias/sys_master/root/8834321252382/EPOS2-Communication-Guide-En.pdf (Accessed: Apr. 17, 2020).

[7] Microchip PIC24F1024GB610 Documentation. [Online]. Available: https://www.microchip.com/wwwproducts/en/PIC24FJ1024GB610 (Accessed: Jun. 8, 2021).

5 Appendix A - Maxon Communication Library

This library comprises the main commands used to control the Maxon motor, which drives fast axis scanning for the RPD used in the imaging system.

Command	Description
SetOperationMode	Switches between Profile Velocity and Step Direction modes for scanning and returning to home position.
SetProfileVelocity	Sets the desired velocity while in Profile Velocity mode
GetObject	Retrieves a specific object dictionary value
SetObject	Sets a specific object dictionary value
ActivateStepDirectionMode	Causes the motor to start moving in Step Direction Mode
ActivateProfileVelocityMode	Causes the motor to start moving in Profile Velocity mode
ClearFault	Clears a fault flag if present
SetState	Allows the state to be set ie. QuickStop, Enabled, Disabled
GetDeviceErrorCode	Gets an error code if error is present
GetVelocityIs	Gets the current motor velocity
SetStepDirectionParameter	Sets the ratio of input signal rising edges to rotation steps for Step Direction Mode

TABLE 11. MAXON COMMUNICATION LIBRARY

6 Appendix B – MCU Debug Board to RPD Component Connections

The Explorer 16/32 development board has a Plug-In Module (PIM) which allows many different Microchip MCUs to be programmed and tested. The development board contains pin-out for the PIM including the mikroBus, GPIO, EEPROM, and many other features. Figure 21 shows the relevant pins and connections currently used for controlling the RPD:

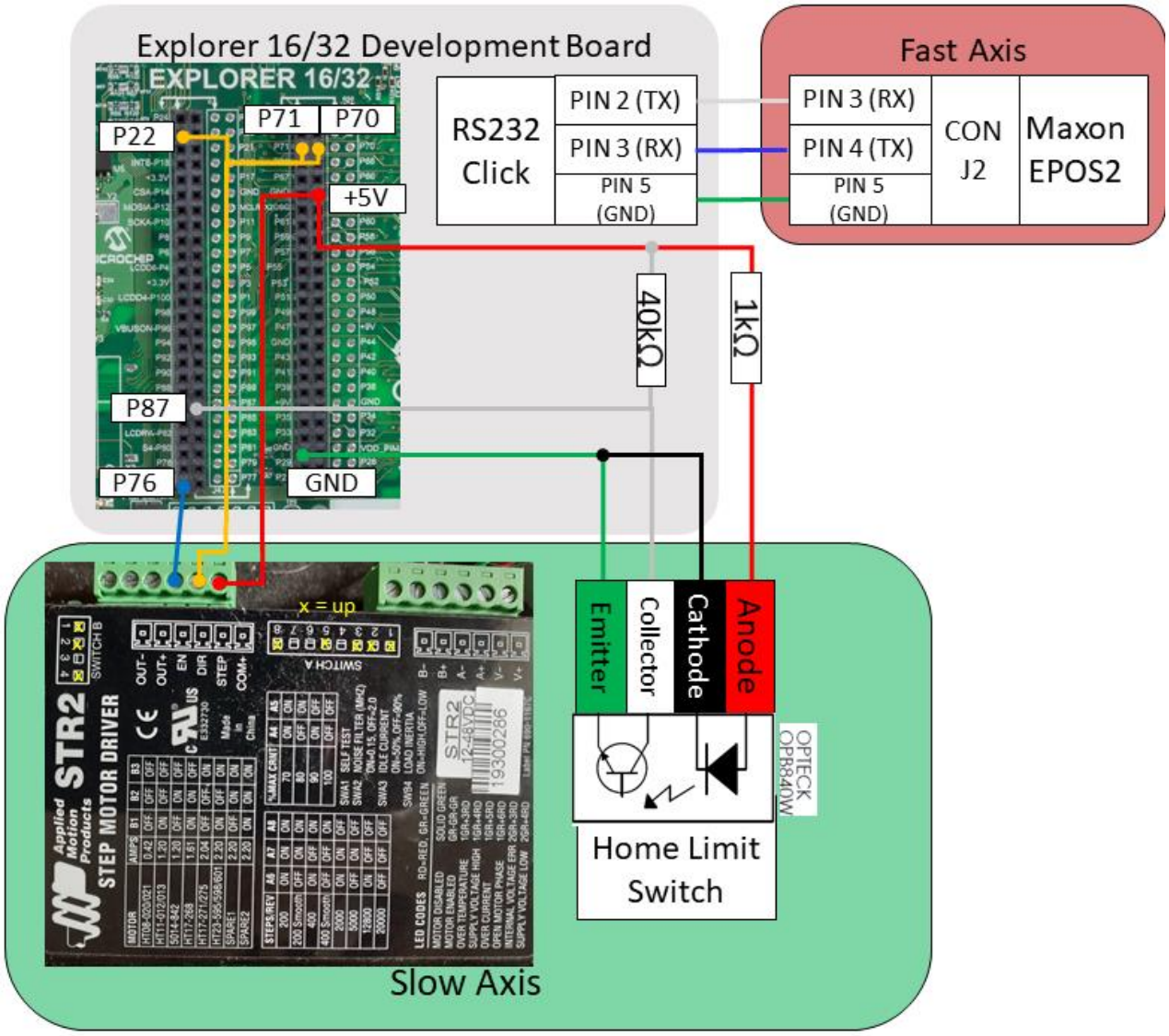


FIGURE 21. EXPLORER 16/32 BOARD TO RPD CONNECTIONS