Prosthesis v1 Application Notes

Table of Contents

[1. General Notes 2](#_Toc186367847)

[1.1 Wiring 2](#_Toc186367848)

[1.2 Software Usage 2](#_Toc186367849)

[2. Flow of Code 3](#_Toc186367850)

[3. CubeMonitor 4](#_Toc186367851)

[3.1 GUI Description 4](#_Toc186367852)

[3.2 Variable Output to .csv 4](#_Toc186367853)

[4. User Defined Options 5](#_Toc186367854)

[4.1 Initialization Settings 5](#_Toc186367855)

[4.1.1 Joint 5](#_Toc186367856)

[4.12.2 Side 5](#_Toc186367857)

[4.2 Test Programs 5](#_Toc186367858)

[4.2.1 None 6](#_Toc186367859)

[4.2.2 Read Only 6](#_Toc186367860)

[4.2.3 Encoder Bias 6](#_Toc186367861)

[4.2.4 Impedance Control 6](#_Toc186367862)

[5. Future Improvements 7](#_Toc186367863)

1. General Notes

1. **IMPORTANT:** A new magnetic encoder bias position must be found and defined whenever the magnet is reassembled into the device. A test program is provided to find the bias. See chapter 4 for more information. Number correct??
2. Variables with prefix CM\_ are used in CubeMonitor.
3. Coordinates are x = forward, y = up, z = right. CM\_ variables follow the right-hand rule.
4. Knee and ankle angle figure.
5. Timing for control loop is based on LPTIM2 set for a 512 Hz interrupt.
6. Executed code in control loop (shown below) completes in XXX.



Executed code in control loop (found in main.c). Lines correct??

# 1.1 Wiring

XXX

# 1.2 Software Usage

1. STM32CubeIDE 1.10.1 (later version probably work as well)
2. STM32CubeMonitor 1.8.0 (later versions probably work as well)

2. Flow of Code

XXX

3. CubeMonitor

CubeMonitor serves as the interface between the device and the user. It performs data acquisition, plotting and displaying of variables, and user defined inputs to variables.

# 3.1 GUI Description

XXX

# 3.2 Variable Output to .csv

XXX

4. User Defined Options

The firmware contains some user defined options that must be changed within the code itself (not in CubeMonitor).

# 4.1 Initialization Settings

The user can modify the code in main.c to initialize the Prosthesis device with the desired settings. The settings can be found as shown below. The options for initialization settings are described in the follow sections.



User defined initialization settings. Lines correct??

## 4.1.1 Joint

Enter a value below for Prosthesis\_Init.Joint shown above to control either the Ankle, Knee, or Combined (both) joint(s).

*Ankle*

*Knee*

*Combined*

## 4.12.2 Side

Enter a value below for Prosthesis\_Init.Side shown above to control either the Left or Ride side of the participant.

*Left*

*Right*

# 4.2 Test Programs

Various test programs are provided to check functionality at my desk. A test program is selected in main.c as shown below. The options for test programs are described in the follow sections.



User defined test program. Lines correct??

## 4.2.1 None

Enter value *None* as the argument for the RequireTestProgram() function shown above. This program runs the full firmware.

## 4.2.2 Read Only

Enter value *ReadOnly* as the argument for the RequireTestProgram() function as shown above. This program allows all sensors to be read and the state machine to be deployed, but no power will be provided to the motor(s).

## 4.2.3 Encoder Bias

Enter value *EncoderBias* in the argument for the RequireTestProgram() function as shown above. This program allows all sensors to be read and the state machine to be deployed while calculating the average encoder reading of every 10 consecutive samples. No power is provided to the motor(s). This is helpful if the encoder’s relative position changes in the device and a new bias must be found and programmed in prosthesis\_v1.c. The Ankle encoder bias can be found as shown below.



Ankle encoder bias. Lines correct??

The Knee encoder bias can be found below.



Knee encoder bias. Lines correct??

My approach for ensuring some confidence in the encoder reading is shown below. With the device attached to a fixture I used a digital level (currently found in the toolbox drawer labeled “MEASURES”)

Add figure.

## 4.2.4 Impedance Control

Enter value *ImpedanceControl* in the argument for the RequireTestProgram() function as shown above. This program allows all sensors to be read and initially calculates equilibrium of the current motor position(s). The control gains are initially XXX for Ankle and XXX for Knee. The values can be changed from the CubeMonitor interface. State machine is not deployed??

5. Future Improvements

1. IMU sensitivities can probably be better than 8 g and 1000 °/sec. They were adopted from previous firmware.
2. Nominal current for motor is most likely not accurate. This was adopted from previous firmware. Should probably ask a forum to help out.
3. It would be nice if CubeMonitor had a START, STOP, and RESET *program* button (different than the acquisition buttons). This would allow low- and high-level power to be turned on asynchronously and require less power cycling during testing.
4. Having Initialization Settings and Test Programs selected via CubeMonitor instead of changing the actual firmware. This was attempted, but it only worked by passing variable addresses in CubeMonitor. When the firmware is rebuilt, the variable addresses can (and most often do) change. However, others who are smarter than me might be able to figure out a way to get it to work, or perhaps CubeMonitor will make this approach more robust in the future.