Prosthesis v1 Application Notes

Table of Contents

[1. General Notes 2](#_Toc203390118)

[1.1 Wiring 2](#_Toc203390119)

[1.2 Software Usage 2](#_Toc203390120)

[2. Flowchart of Code 4](#_Toc203390121)

[3. State Machine 5](#_Toc203390122)

[4. CubeMonitor 6](#_Toc203390123)

[4.1 GUI Description 6](#_Toc203390124)

[4.2 Variable Output to .csv 6](#_Toc203390125)

[5. User Defined Options 7](#_Toc203390126)

[5.1 Initialization Settings 7](#_Toc203390127)

[5.1.1 Joint 7](#_Toc203390128)

[5.1.2 Side 7](#_Toc203390129)

[5.2 Test Programs 7](#_Toc203390130)

[5.2.1 None 8](#_Toc203390131)

[5.2.2 Read Only 8](#_Toc203390132)

[5.2.3 Impedance Control 8](#_Toc203390133)

[6. Future Improvements 9](#_Toc203390134)

1. General Notes

1. **IMPORTANT:** A new encoder bias position must be found and defined whenever the magnet is reassembled into the device. The bias can be found by logging data in CubeMonitor and averaging the encoder values.
2. There is a GitHub release (rev0) for the working version of firmware. This may be helpful if others make future changes and want to return to this version.
3. Variables with prefix CM\_ are used in CubeMonitor for visualization, logging, or both.
4. Positive/negative Knee angles are flexion/extension, respectively. Positive/negative Ankle angles are dorsiflexion / plantar flexion, respectively.
5. IMU coordinates are x = forward, y = up, z = right. IMU variables follow the right-hand rule.
6. Control loop runs at 512 Hz.
7. Executed code in control loop (shown below) completes in ?? ms.
8. The nominal current for the motors is, to the best of my knowledge, not published. The EPOS4 documentation states that the controller will drop current if it pulls 2X the nominal current. So, I chose the current limit divided by 2 to be the nominal current to avoid this issue. No problems have occurred as of writing this note.

# 1.1 Wiring

A black background with a black square

AI-generated content may be incorrect.

**Figure 1.** Wiring of Prosthesis v1.

# 1.2 Software Usage

1. All files were built with STM32CubeIDE 1.10.1 with no errors or warnings. Using newer versions of STM32CubeIDE (which may have newer versions of GCC and/or Eclipse) may be stricter on builds yielding errors/warnings.
2. STM32CubeMonitor 1.8.0 (later versions *probably* work without issues).

2. Flowchart of Code

A screenshot of a chat

AI-generated content may be incorrect.

**Figure 2.** Flowchart of code.

3. State Machine

A black background with a black square

AI-generated content may be incorrect.

**Figure 3.** State machine.

4. CubeMonitor

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

# 4.1 GUI Description

A screenshot of a computer

AI-generated content may be incorrect.

**Figure 4.** CubeMonitor GUI with descriptions (fitting whatever I can from my screen).

Fig. 4 shows the GUI with descriptions for each section. The four charts will plot the shown values as well as a “staircase” of the state machine sized appropriately for each chart. The lowest “stair” is Early Stance and the highest “stair” is Swing Descension as shown in Fig. 3.

Thresholds can be adjusted on the fly. Once I had these dialed in I did not need to adjust for following experiments. However, for every new participant these values are likely to be unique.

State machine control parameters only have an effect when test program is *None*.

Impedance control parameters only have an effect when test program is *ImpedanceControl*.

LED will turn green when program starts. If any errors occur, whether during initializations or normal operation, LED will turn red and the corresponding errors will be displayed. A description of the EPOS errors (and aborts) can be found in the EPOS4 Firmware Specification documentation.

# 4.2 Variable Output to .csv

All CM\_ variables are logged in a .csv file whenever the data acquisition is stopped. These files can be found in C:\Users\<user name>\log.

# 4.3 Updating after Builds

A screenshot of a computer

AI-generated content may be incorrect.

**Figure 5.** Steps for updating CubeMonitor after builds.

CubeMonitor works off memory addresses. After builds memory addresses can change. Fig. 5 shows steps for updating CubeMonitor to grab the current memory addresses. Detailed instructions are discussed below.

1. Double click *Pv1 Variables* block.
2. Click *Edit* button.
3. Check all CM\_ variables are checked (you may now notice why having CM\_ in front of these variables is very helpful).
4. Click *Update* button.
5. Click *Done* button.
6. Repeat for all remaining variable blocks (including those not shown in Fig. 5). If all CM\_ variables in Step 3 were checked, then you can skip Step 3 for all remaining variable blocks.

5. User Defined Options

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

# 5.1 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 following sections.



**Figure 6.** User defined initialization settings.

## 5.1.1 Joint

Enter a value below for Prosthesis\_Init.Joint shown in Fig. 6 to control either the Ankle, Knee, or Combined (both) joints.

*Ankle*

*Knee*

*Combined*

## 5.1.2 Side

Enter a value below for Prosthesis\_Init.Side shown in Fig. 6 to control either the Left or Ride side of the participant.

*Left*

*Right*

# 5.2 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.



**Figure 7.** User defined test program.

## 5.2.1 None

Enter value *None* as the argument for the RequireTestProgram() function shown in Fig. 7. This program runs the full firmware. Control parameters for both joints are set to initial values for every state. At the time of writing this section, Ankle parameters are all set to values that the participant is comfortable standing. Knee parameters are set to zero for Stance and tuned values for Swing. All control parameters can be adjusted in CubeMonitor.

## 5.2.2 Read Only

Enter value *ReadOnly* as the argument for the RequireTestProgram() function as shown in Fig. 7. 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).

## 5.2.3 Impedance Control

Enter value *ImpedanceControl* in the argument for the RequireTestProgram() function as shown in Fig. 7. Works identically as *None* but the state machine has no affect on the control parameters. Instead a single set of parameters are used which can be set from CubeMonitor. This can be helpful for bench testing purposes where it is difficult to have the state machine trigger correctly.

6. Future Improvements

1. IMU sensitivities can probably be better than 8 g and 1000 °/sec. They were adopted from previous firmware.
2. 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.