Guide: Interfacing with  
CubeMars AK Series Motors

Contents

[Introduction 2](#_Toc182939342)

[Resources 2](#_Toc182939343)

[CubeMars 2](#_Toc182939344)

[STM32 2](#_Toc182939345)

[Hardware Configuration 3](#_Toc182939346)

[Setup for CubeMars Software Control 3](#_Toc182939347)

[Setup for STM32 Control 3](#_Toc182939348)

[CubeMars Software 5](#_Toc182939349)

[General Usage 5](#_Toc182939350)

[Debug Panel 6](#_Toc182939351)

[For More Information 6](#_Toc182939352)

[STM32 7](#_Toc182939353)

[Conclusion 8](#_Toc182939354)

[Recommendations for Future Work 8](#_Toc182939355)

# Introduction

This document introduces the control of CubeMars AK series motors using CubeMars software and STM32 microcontrollers. Below is a collection of useful resources, and on the following pages, the hardware setup will be described. This is followed by a quick-start guide to using the CubeMars software and, finally, descriptions of the developed code for STM32 microcontrollers.

# Resources

## CubeMars

AK80-9 specs and graph: <https://www.cubemars.com/goods-982-AK80-9.html>

AK80-9 specs and graph: <https://uav-en.tmotor.com/html/2021/A_0106/640.html>

AK Series Q&A: <https://www.cubemars.com/article-262-AK+Series.html>

AK Series Module Driver User Manual (included in the *Motor Control* folder): <https://www.cubemars.com/images/file/20240611/1718085712815162.pdf>

Software and video guides: <https://www.cubemars.com/article.php?id=261>

Download “CubeMars upper computer” from the last link and follow the first few video guides to connect the motor. If you can’t connect, try installing “CH340 Driver” on your PC. To get the motor running, watch video 2, “MIT mode control”, and make sure you follow all the steps.

## STM32

NUCLEO-L476RG website: <https://www.st.com/en/evaluation-tools/nucleo-l476rg.html>

For easy access to convenient references, the following files have been collected in the “NUCLEO-L476RG” folder:

* Data brief for Nucleo-64 boards (nucleo-l476rg.pdf)
* User manual for Nucleo-64 boards (um1724-stm32-nucleo64-boards-mb1136-stmicroelectronics.pdf)
* Pinout for NUCLEO-L476RG (NUCLEO-L476RG Pinout.png)
* Screenshot of default pinout configuration (IOC Config Default.png)
* Screenshot of current pinout configuration (IOC Config 2024-10-28.png)
* Datasheet for STM32L476xx microcontrollers (stm32l476rg.pdf)
* Reference manual for STM32L4xxxx microcontrollers (rm0351-stm32l47xxx-stm32l48xxx-stm32l49xxx-and-stm32l4axxx-advanced-armbased-32bit-mcus-stmicroelectronics.pdf)

# Hardware Configuration

## Setup for CubeMars Software Control

To control the motor, the R-LINK is connected to the PC using a USB cable. The R-LINK is then connected to the

The CubeMars motor is connected to the PC using the R-LINK

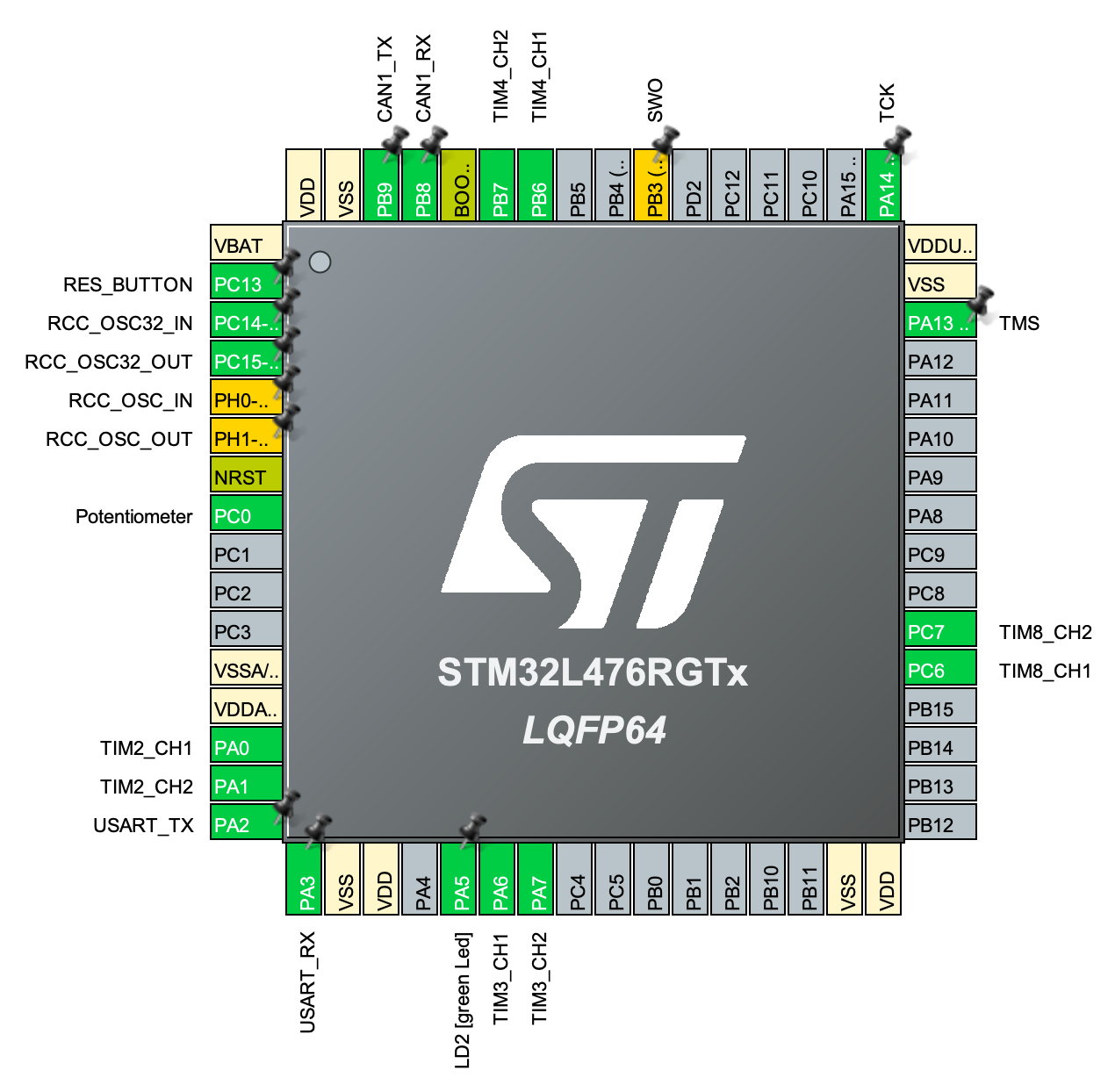
Power supply

Limits

## Setup for STM32 Control

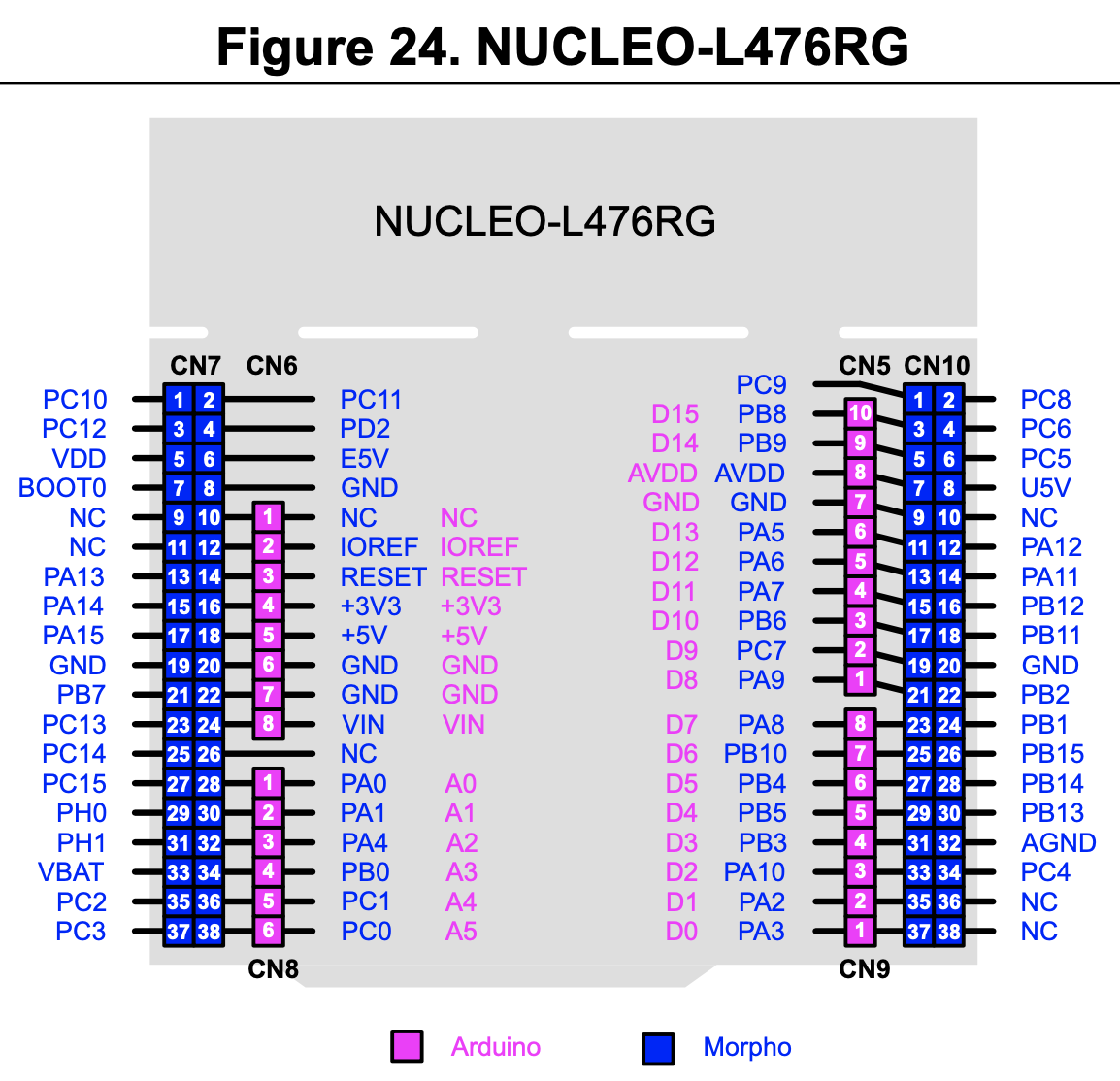
The NUCLEO-L476RG development boards are connected to the PC using USB Mini-B cables. When working with multiple MCUs, it can be helpful only to connect one of them to the PC using a data cable while the other one is just connected to a power source. This avoids confusing the MCUs with each other when debugging.

The current input pinout configuration in the initialisation and configuration (IOC) tool (referred to as STM32CubeMX) is shown below:



Of main interest are the CAN RX and TX (upper left) and Potentiometer (left).

The physical pinout of the Nucleo development board is shown below:



Referring to the physical pinout, the connections to be made are:

1. CAN transceiver Vcc: +5V (left)
2. CAN transceiver GND: GND (left)
3. CAN transceiver RX: PB8/D15 (upper right)
4. CAN transceiver TX: PB9/D14 (upper right)
5. Potentiometer Vcc (purple): +3V3 (left)
6. Potentiometer GND (grey): GND (left)
7. Potentiometer Signal (yellow): PC0/A5 (lower left)

To facilitate communication between two or more units, they should be connected CANH-CANH (blue) and CANL-CANL (white).

# CubeMars Software

The CubeMars software (referred to as the “upper computer”) can be downloaded from the link under *Resources* above. A screenshot of the software is shown below.

A screenshot of a computer

Description automatically generated

## General Usage

The software is a bit quirky and will likely take some getting used to. Start by maximising the window and changing the language to English using the switch below the colourful circle on the left.

Most controls are done in the panel on the right. To connect the motor, start by pressing “refresh”. This should display the relevant COM port in the dropdown which can then be connected by pressing the “Connect” button. The status in the lower right corner should then change accordingly.

There are two built-in control modes, “Servo Control” and “MIT Control”. Both can be used for position and/or speed control, but the difference lies in the control structure (refer to the user manual for block diagrams). In this guide, MIT mode will be used.

To control the motor in MIT mode, select the “Mit Control” tab on the right. First, set the motor’s CAN ID. (If you don’t know the motor’s CAN ID, see *Debug Panel* below.) Make sure all other values are zero. Press “Set Origin” to make sure the motor’s current position is zero before starting position control. Set the Kp (position control) and Kd (speed control) parameters. Then, press “RUN” to start CAN communication with the motor. At this point, you should see the graphs in the centre starting to display data received via CAN. You can then change the position (“P”) and speed (“S”) setpoints, and the motor should move accordingly. In the centre, switch between tabs to see graphs of position speed, temperature, duty cycle, DC current, FOC currents etc.

As a conservative starting point, Kp = 0.5, Kd = 0.2 and S = 0 can be used for position control. In this case, Kp controls the position and Kd limits the speed. For pure speed control, Kp = 0 and Kd = 0.2 can be used. In this case, the position reference has no effect, and Kd controls the speed.

Additionally, it is possible to command a feed-forward torque (“T”), which translates directly to a current without any associated feedback control.

## Debug Panel

When the serial connection has been established (through the COM port, as described above), the motor can be controlled using a CAN connection (though the CAN ID).

If the CAN ID is unknown, it can be found through the Debug panel. This panel can also be used to read or set other parameters, to monitor encoder data and other things.

To see the configuration of the connected motor, first go to the debug panel by pressing the “DEBUG” button in the MIT Control panel. This opens a command line-like interface for serial communication.

To read motor parameters, enter “setup”. This outputs a list of parameters, including the CAN ID. Other usable commands are displayed in the main debug menu. To return to the main debug menu, enter “exit”.

## For More Information

In the left panel, graphical menus not mentioned above can be opened, such as for settings, system parameters and exporting and importing settings.

For more information about using the CubeMars software, I recommend watching the video guides linked under *Resources* and referring to the user manual for detailed information about control modes and communication.

# STM32

The code developed for the NUCLEO-L476RG is based on template code from UTS Motorsports. Minimal changes have been made to the existing code and no changes to the IOC configuration. This section will describe the additions to the code.

For debugging purposes,

**Project Properties** under **C/C++ Build** > **Settings**: **Tool Settings**, expand **MCU GCC Linker** and select **Miscellaneous**: Add “-u \_printf\_float”

This may use significant flash memory, RAM and CPU, which could be problematic for critical real-time operation, so you should strongly consider removing the flag and limiting or eliminating the use of printf before doing critical testing.

## Constants and typedefs in cubemars\_control.h

In addition to function prototypes, cubemars\_control.h declares four special CAN commands used for entering and exiting motor control mode, setting the origin and reading the state (note that the command for reading the state is identical to the command for entering motor control mode). The function also declares limits for position, velocity, torque and control parameters. The values of the constants are assigned in cubemars\_control.c.

Additionally, a typedef for an enumeration (enum) defining different error codes received from the motor, which can be useful for monitoring.

## Functions in cubemars\_control.c

In cubemars\_control.c, values are assigned to the special CAN commands and limits mentioned above. Additionally, it contains functions used for communication with sending commands to CubeMars motors and receiving feedback. Most of these functions are based on examples from the user manual, many of them with modifications.

### float\_to\_uint and uint\_to\_float

Converts between floats and integers within specified

### pack\_cmd

### cubemars\_send\_can\_cmd

### cubemars\_get\_can\_msg

### unpack\_reply

### cubemars\_get\_can\_cmd4debug

### unpack\_cmd4debug

### print\_raw\_data, print\_motor\_data, print\_motor\_error and print\_cmd4debug

Four functions for printing data: print\_raw\_data prints 8 bytes as 8 integers; print\_motor\_data prints data received from motor; print\_motor\_error prints a fault description based on the received error code; and print\_cmd4debug prints the command received from another MCU for debugging purposes.

## Includes in main.h

Two lines have been added in “USER CODE BEGIN Includes” where math.h and stdio.h have been included.

## Structure of main.c

CAN\_SENDER\_ID and CAN\_LISTEN\_ID

Main loop

Enabling “Continuous Conversion Mode” would remove the need for repeated HAL\_ADC\_Start(&hadc1)

After receiving the message in the HAL\_CAN\_RxFifo0MsgPendingCallback function, the function cubemars\_get\_can\_msg is called.

# Conclusion

The motor can be successfully controlled using CubeMars software when connected to a PC using the R-LINK. The communication between the microcontrollers via the CAN transceivers also seems to work as expected. However, trying to send commands to the motor did not work, which could suggest that the current CAN configuration on microcontrollers could be incompatible with the motor.

## Recommendations for Future Work

Using the setup in *Hardware Configuration* and following the guide in *CubeMars Software* should allow for quickly getting the motor running using a PC connection.

The code described in *STM32* provides a foundation for further testing and the development of an interface for controlling CubeMars AK motors using STM32 microcontrollers, following adjustments to make the CAN communication compatible.