Table of Content

Introduction to Model-Based Design:

A Comprehensive Guide from A to Z

Author: Ahmed Abdelhafez
Review: Dr. T. Lahlou

1. Introduction to Model-Based Design

Model-Based Design is the systematic use of models throughout the development process that improves how you deliver complex systems. You can use Model-Based Design with MATLAB and Simulink to shorten development cycles and reduce your development time by 50% or more.

1.1. Terms used in Model-Based Design (MBD)

Modelling:
Create mathematical or graphical models to represent system behaviour.
Simulation:
Test system performance in a virtual environment to identify issues early.
Verification and Validation:
Ensure the design meets requirements (verification) and intended purpose (validation).
Automatic Code Generation:
Generate consistent, error-free code (e.g., C, C++) directly from models.
Rapid Prototyping: Develop and test prototypes quickly, reducing physical prototyping costs.
Iteration and Optimization: Continuously refine models to improve performance and address new requirements.
Documentation and Collaboration: Models act as a central reference, simplifying communication and teamwork.
Lifecycle Management: Update and maintain designs efficiently by integrating models across the product lifecycle.
Testing and Feedback: Automate testing and use real-world data to refine models for better accuracy.

1.2. Importance of MBD in Embedded Systems

Faster Prototyping:
Speeds up virtual prototyping before hardware development.
Early Issue Detection:
Identifies design problems early, saving time and costs.
Manages Complexity: Visual models simplify the design of intricate systems.
Consistency: Ensures alignment of design and requirements.
Iterative Refinement: Allows continuous testing and improvement.
Team Collaboration: Provides a shared platform for hardware and software engineers.
MBD is essential in industries like aerospace, automotive, and embedded systems where precision, reliability, and efficiency are critical

2. Hardware Preparation

2.1. Required Hardware

- • Two microcontrollers MPC5744P RevE.

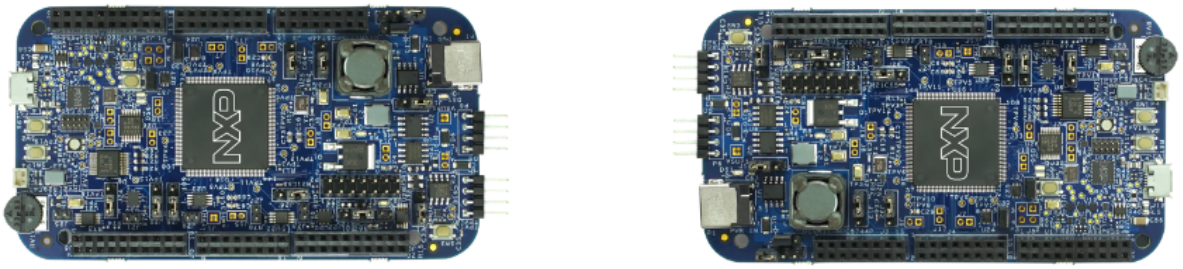


Fig 2.1: Two microcontrollers MPC5744P RevE.

- • Two inverter boards DEVKIT-MOTORGD.

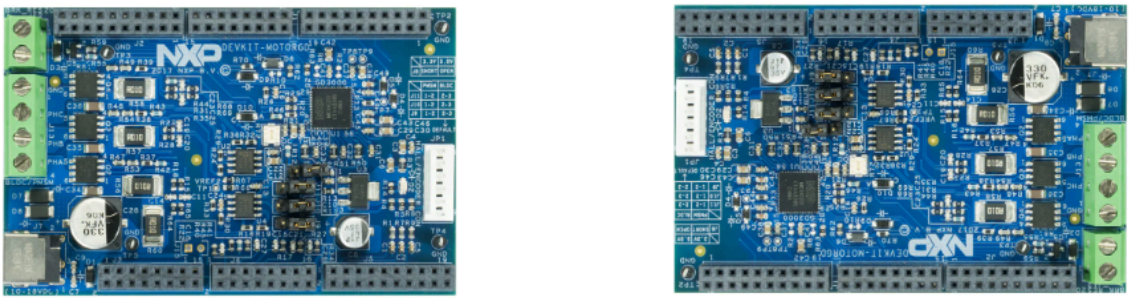


Fig 2.2: Two inverter boards DEVKIT-MOTORGD.

- Two Adapter PCBs

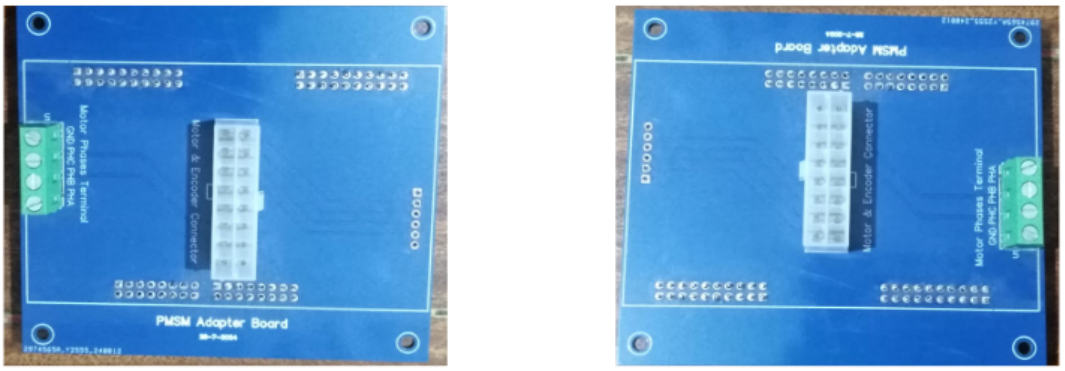


Fig 2.3: Two Adapter PCBs.

- • Two ferrite clamps.

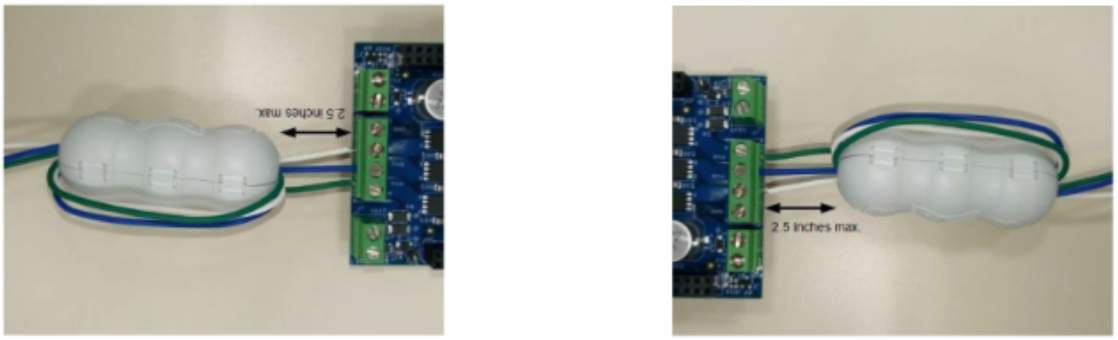


Fig 2.4: Two ferrite clamps.

- Two USB cables.
- Power supply that can supply 12V and 5A.
- Two 3-ph PMSM Teknic M-2310P-LN-04K PMSM with coupling part.

Fig 2.5: Two 3-ph PMSM Teknic M-2310P-LN-04K PMSM with coupling part.

2.2. Steps of setting up the Hardware Testbench

1-Attach each Inverter board above each MCU board to get two couple of boards.Then connect three power cables in the green terminal as shown below..

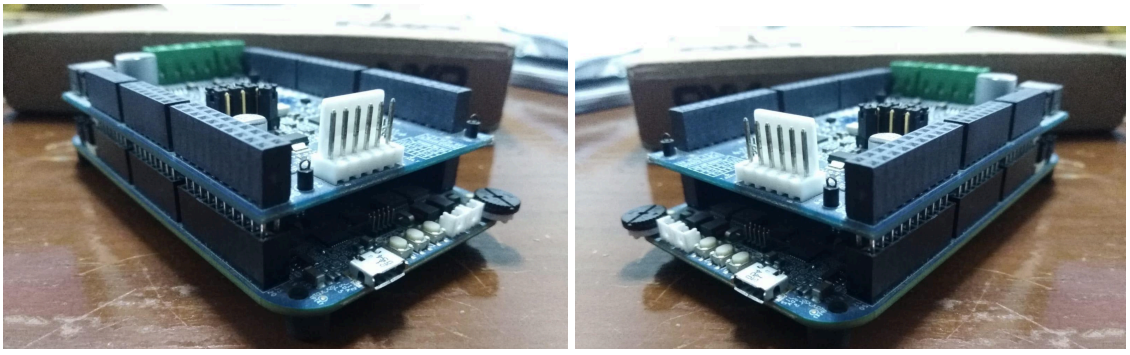


Fig 2.6: MCU & Inverter Boards after attachment

2-Adjust the jumpers j9,j10 & j11 in the inverter boards on position 1-2 for PMSM Running as shown

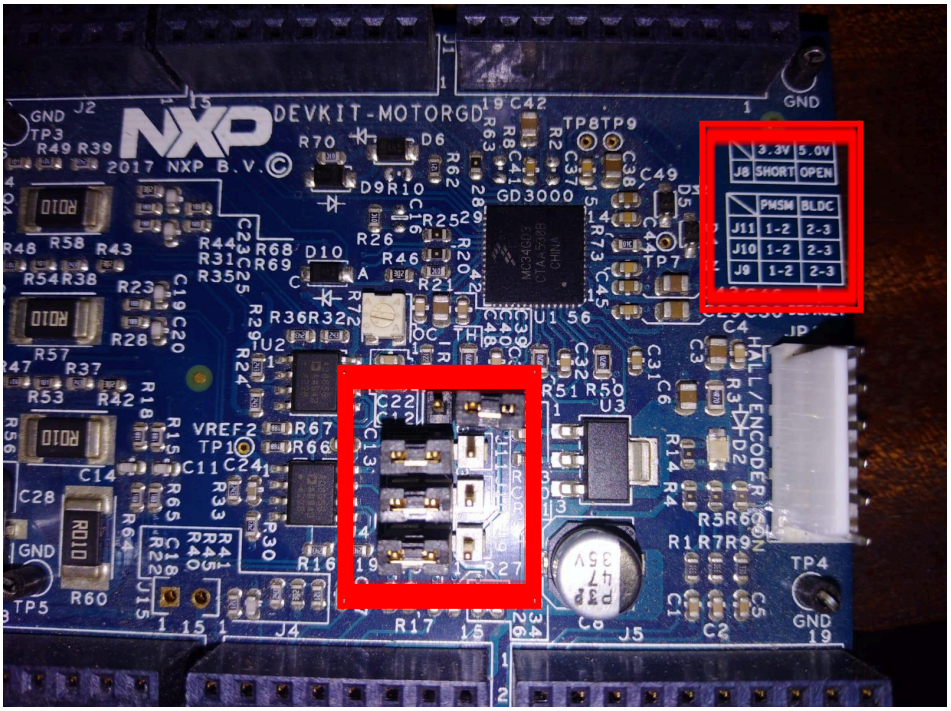


Fig 2.7: Adjusting Jumpers on the Inverter Boards

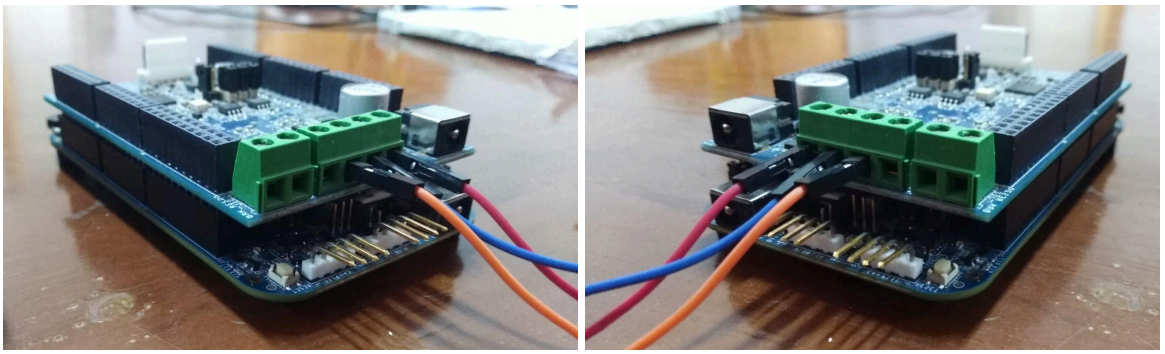


Fig 2.8: MCU & Inverter Boards after connecting cables

3-Attach each adapter PCB above each couple of boards and connect between the terminals of Adapter PCB & Inverter board as shown

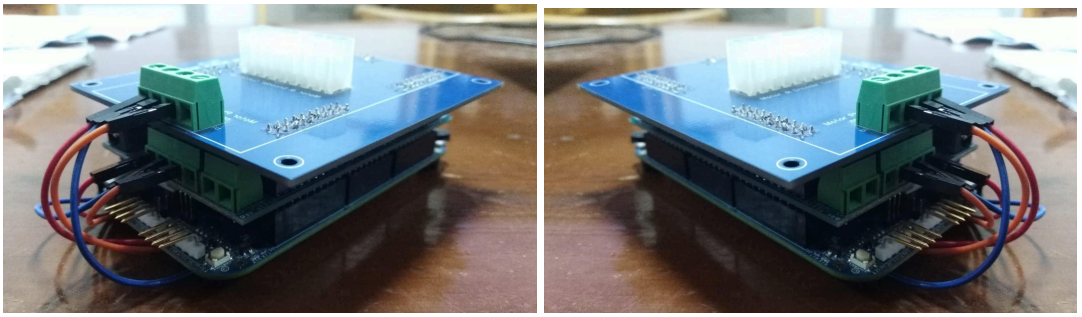


Fig 2.9: MCU Board, Inverter Board & Adapter PCB after attachment

4-Fix the Two motors using holders and coupling as shown

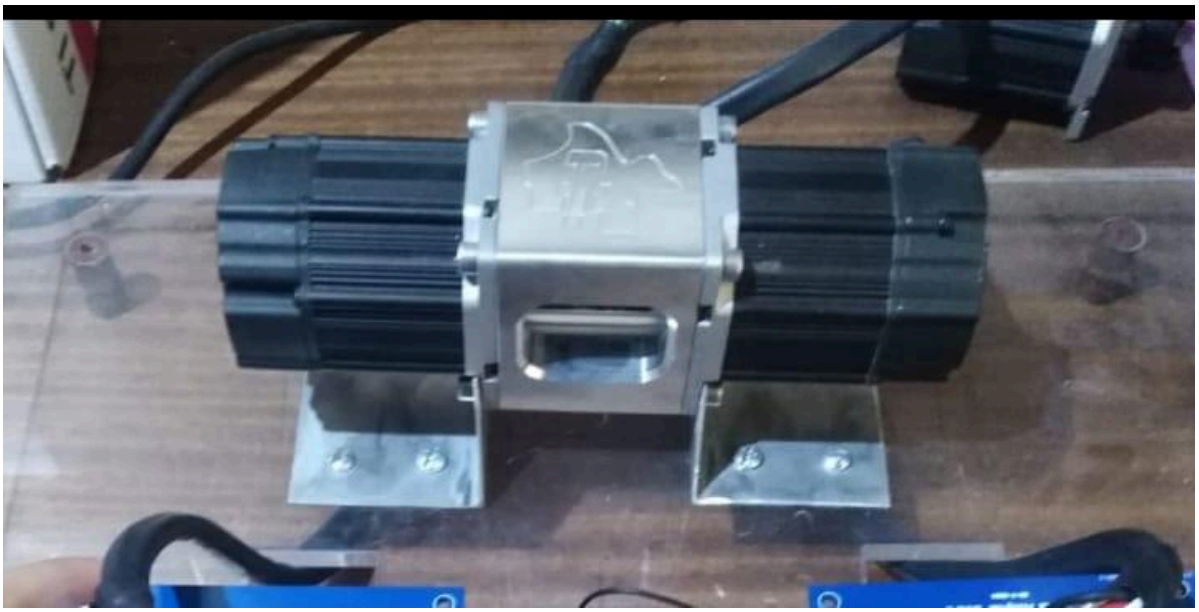


Fig 2.10: Motors coupling and Fixation

5-Connect the jack of motor terminals to the adapter PCB [Each motor to one Board] as shown

Fig 2.11: Connecting Motors to the Adapter PCBs

6-Connect the DC Supply to the two Inverter boards using jacks as shown below

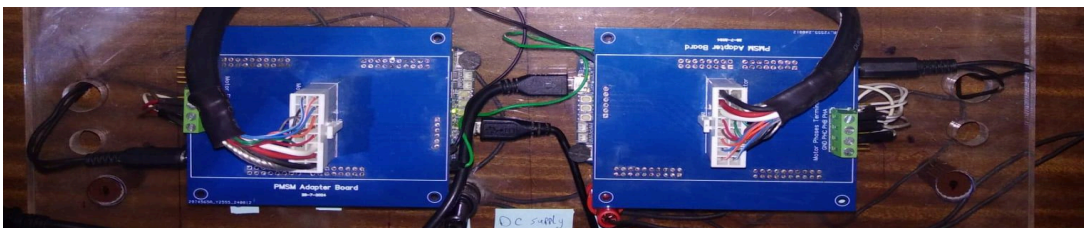


Fig 2.12: Connecting the The two inverter boards to the DC Supply

7-You will end up with the testbench as shown:

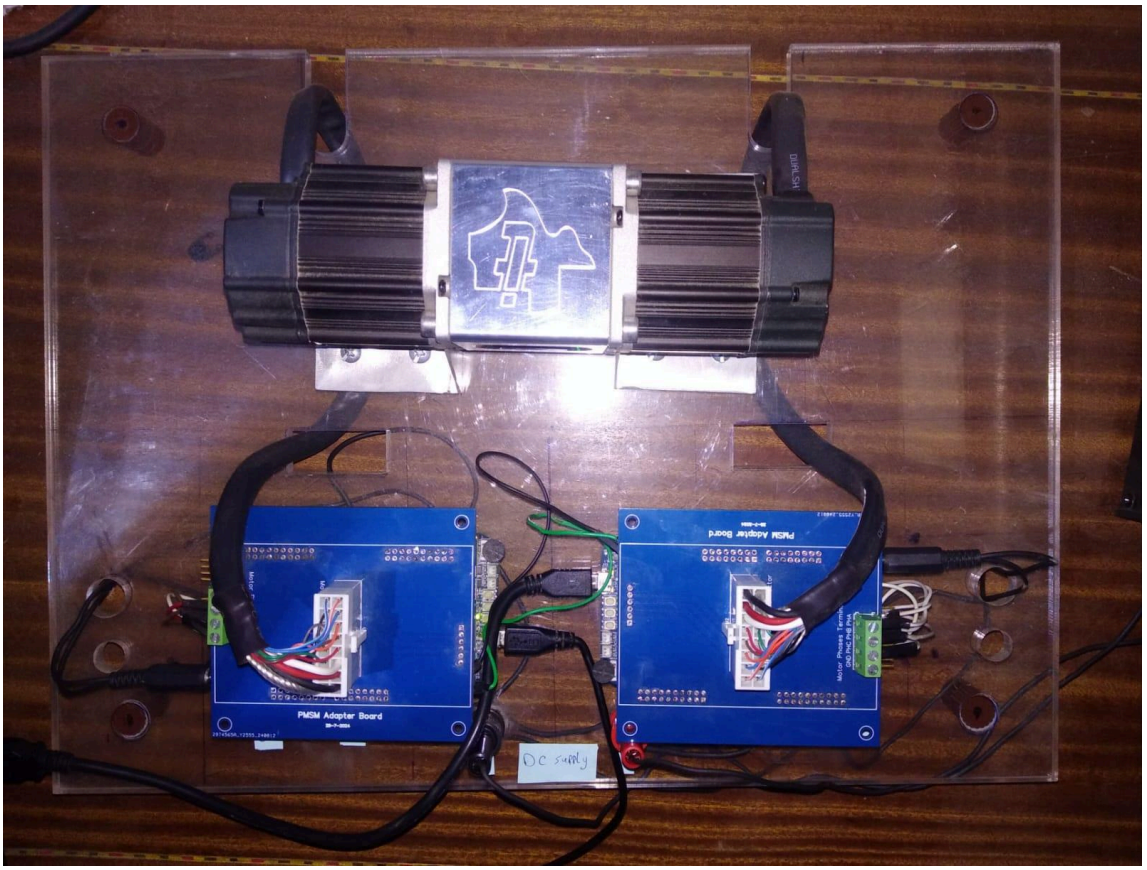


Fig 2.13: Plane View of the TestBench

3. Software Preparation

[All the files needed in this chapter can be found in this folder Files of Chapter 3]

3.1. Steps of Running the testbench for the first time

3.1.1. Connect The MCU board to PC

Use a USB Cable to connect the MCU Board to the PC or Laptop

Note:

If you're using this board for the very first time (i.e., it has never been used before), you'll need to flash the .rbf file onto it. You can follow the instructions provided in the next section. However, if you've already used the board previously, you can skip the next section and go to step 3.1.6

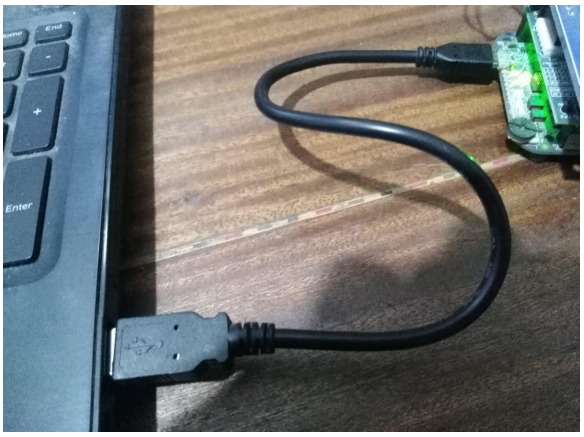


Fig 3.1: Connecting MCU to Laptop

3.1.2. Download the bootloader RAppID

- Download and install the bootloader to use it in uploading the code to the board.
  - Download the zip file from here: RAppID_Boot_Loader
  - Unzip it
  - Install: using setup file

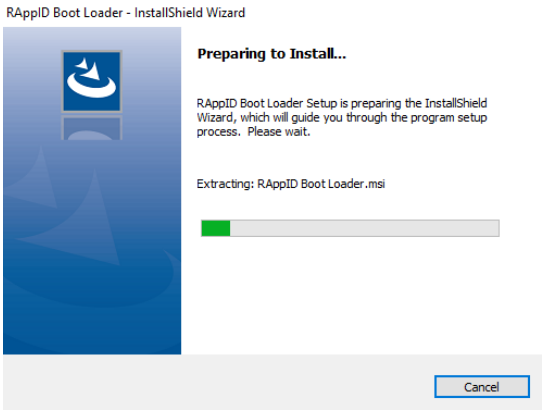


Fig 3.2: Bootloader Installing

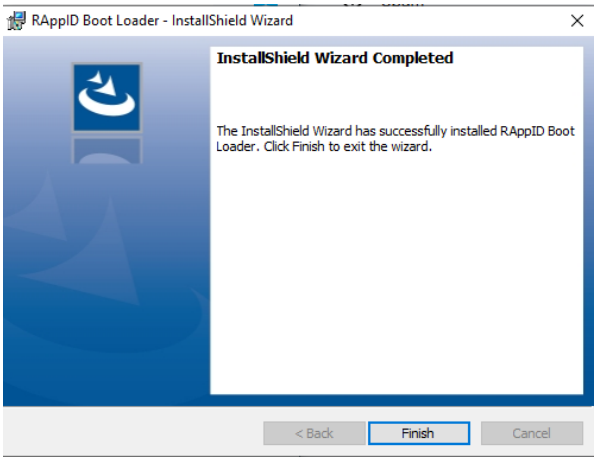


Fig 3.3:Finishing Bootloader Installation

3.1.3. Flashing The .rbf file on the board for the first time

- When you buy a new MPC5744P board from NXP if you connect it to your laptop by USB, the led will emit white light and if you try loading your code [.mot file] on the board by RAppID Bootloader Utility you won't succeed in this. This is because, for the first time only, you should flash on the board the .rbf file of this board using S32 Design studio for Power Architecture [S32DS for PA] and then you can use RAppID Bootloader Utility.
- Follow these steps to flash the new board:

3.1.3.1. Download the [S32DS for PA] if your board is from family MPC57xx

From this link: Downloading S32DS

Fig 3.4: S32DS Downloading

For installation steps refer to the file in this link

3.1.3.2. Open S32DS

After completing the installation of S32DS, open it and you can add any updates or extensions you need from the window that will appear while opening the software.

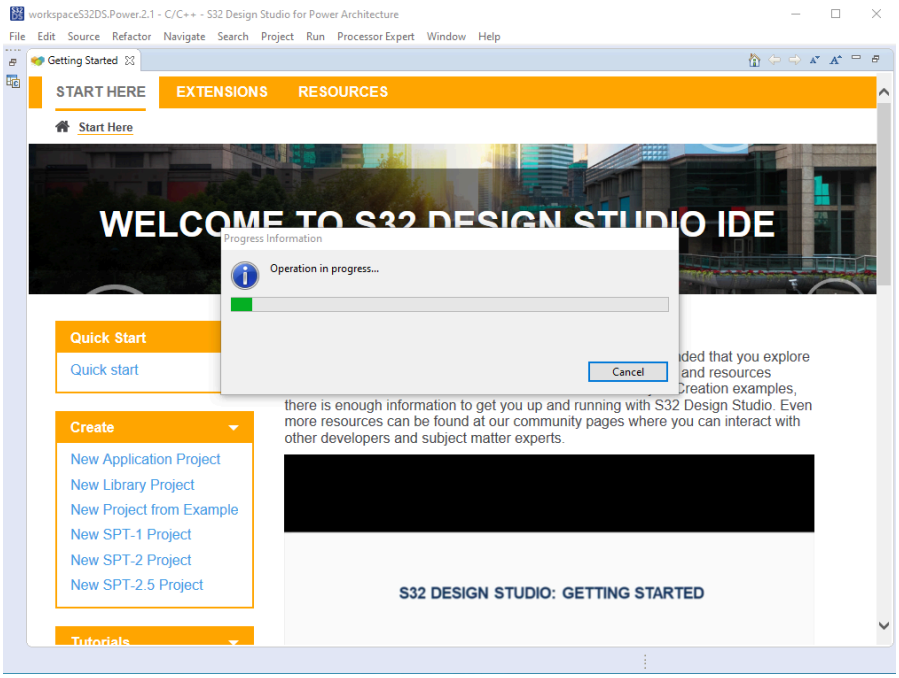


Fig 3.4:Entry Screen of Design Studio Software

It takes several minutes to finish

3.1.3.3. Add the .rbf file

you will be asked for the name of the board and then in the field of C/C++ application , choose browse and find on your device [in the folder of the RAppID Bootloader Utility folder → RBF folder] the .rbf file of this board and then flash it as shown below.

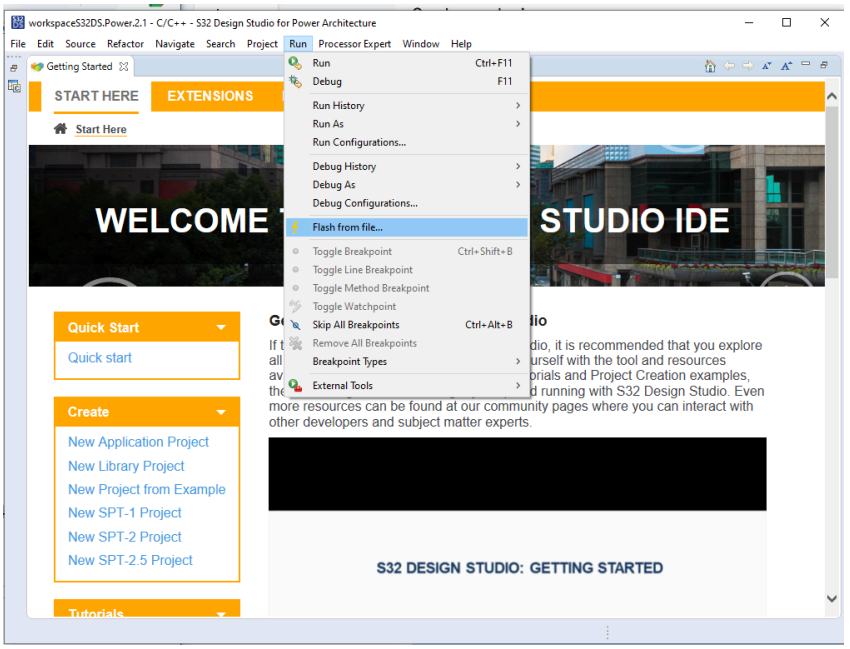


Fig 3.5:Flashing from file

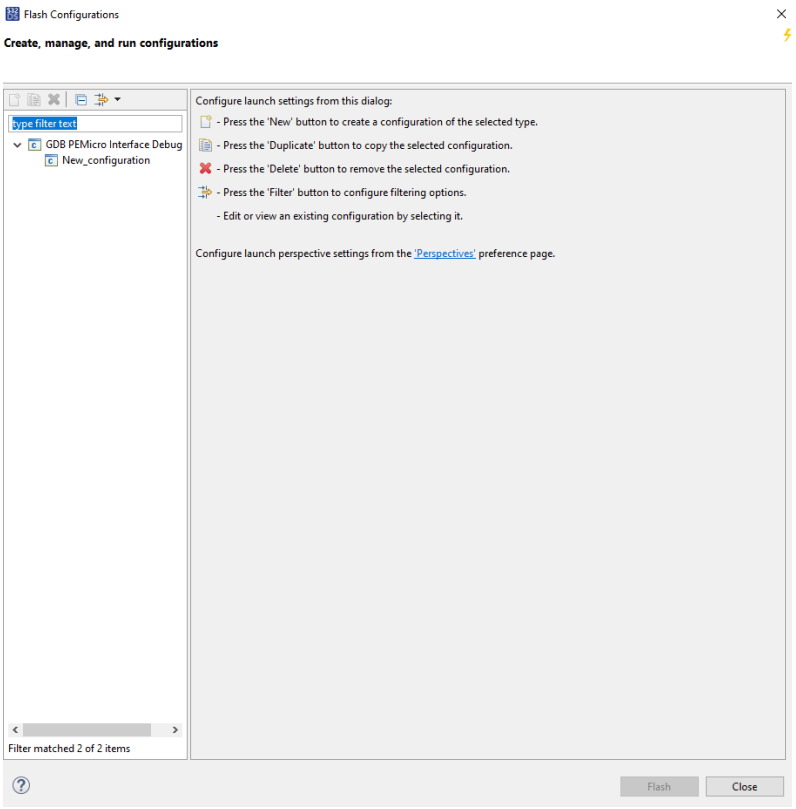


Fig 3.6:Flash Configuration

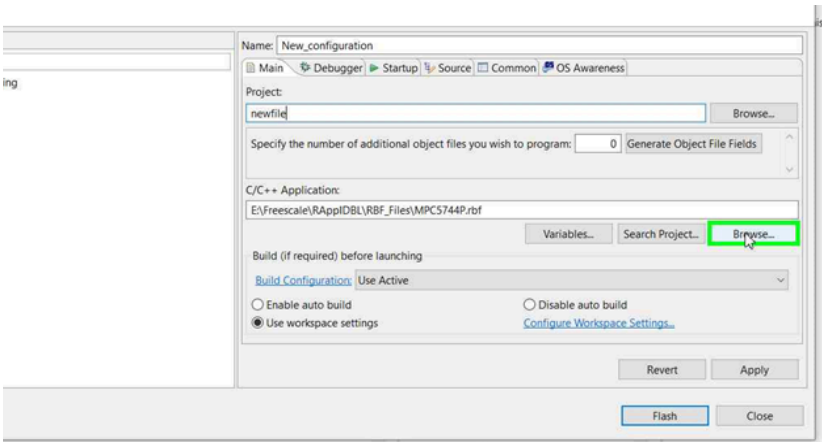


Fig 3.7:browsing for the .rbf file

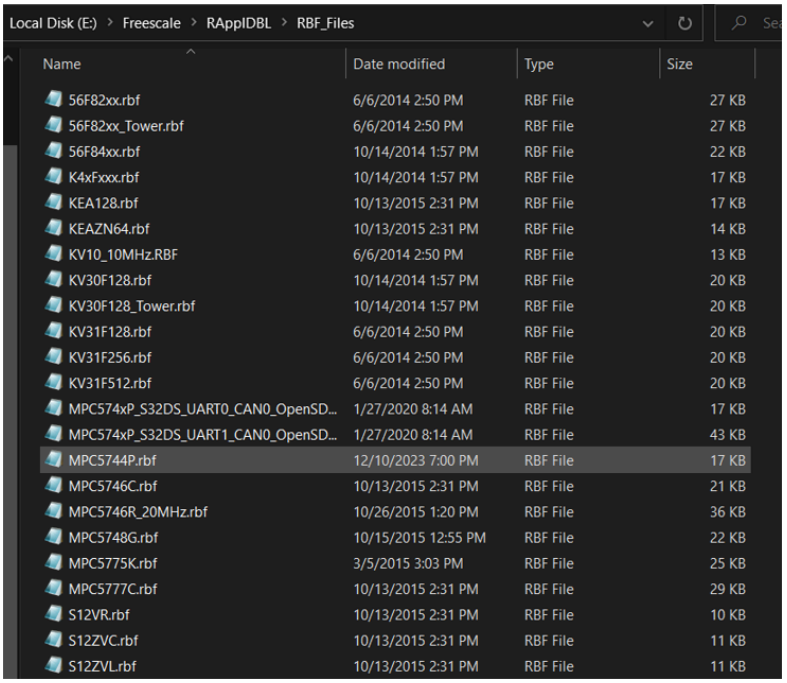


Fig 3.8:Choosing the .rbf file

3.1.3.4. Adjust Configuration

From tab of Debugger, adjust these configurations as shown.

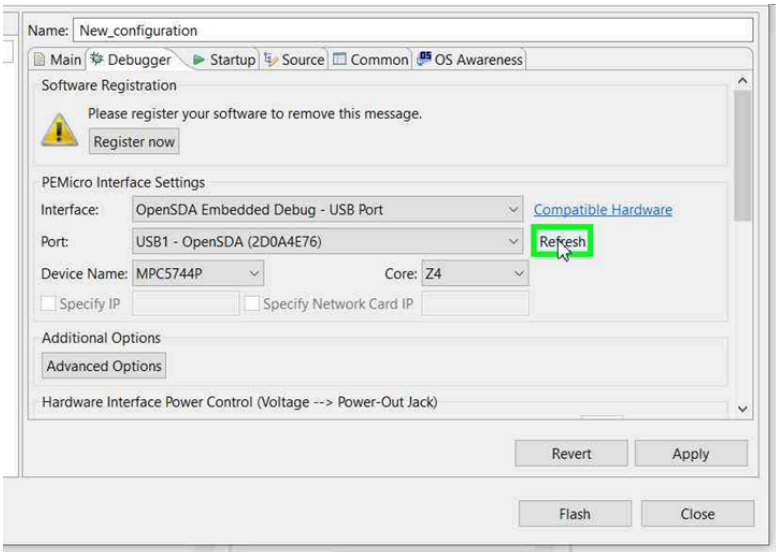


Fig 3.9:Debugger Tab

change tab debugger What does this mean?

3.1.3.5. Press Apply and then Flash

You will notice that the white led is turned off and Now, the flashing is done and the board is ready for loading the code on it using RAppID Bootloader Utility.

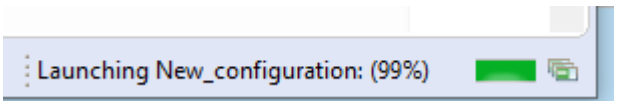


Fig 3.10:Flashing...

3.1.4. Open the Bootloader

- Open the BootLoader and configure the setting as shown in the figure bellow:
- The RAppID was installed in previous steps. Start it from the desktop.

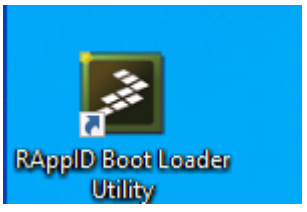


Fig 3:11:icon of the bootloader

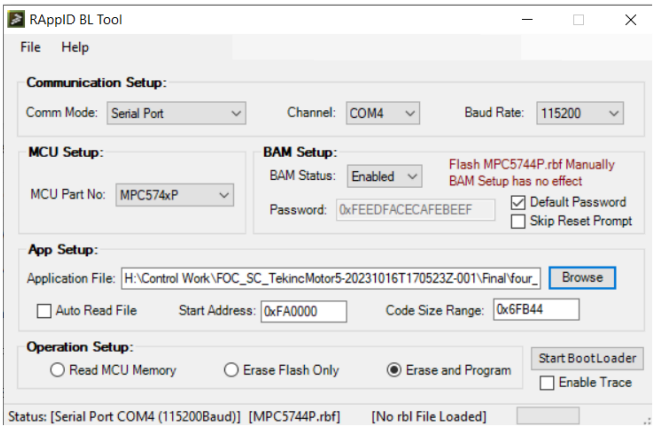


Fig 3.12:Entry Screen of the bootloader

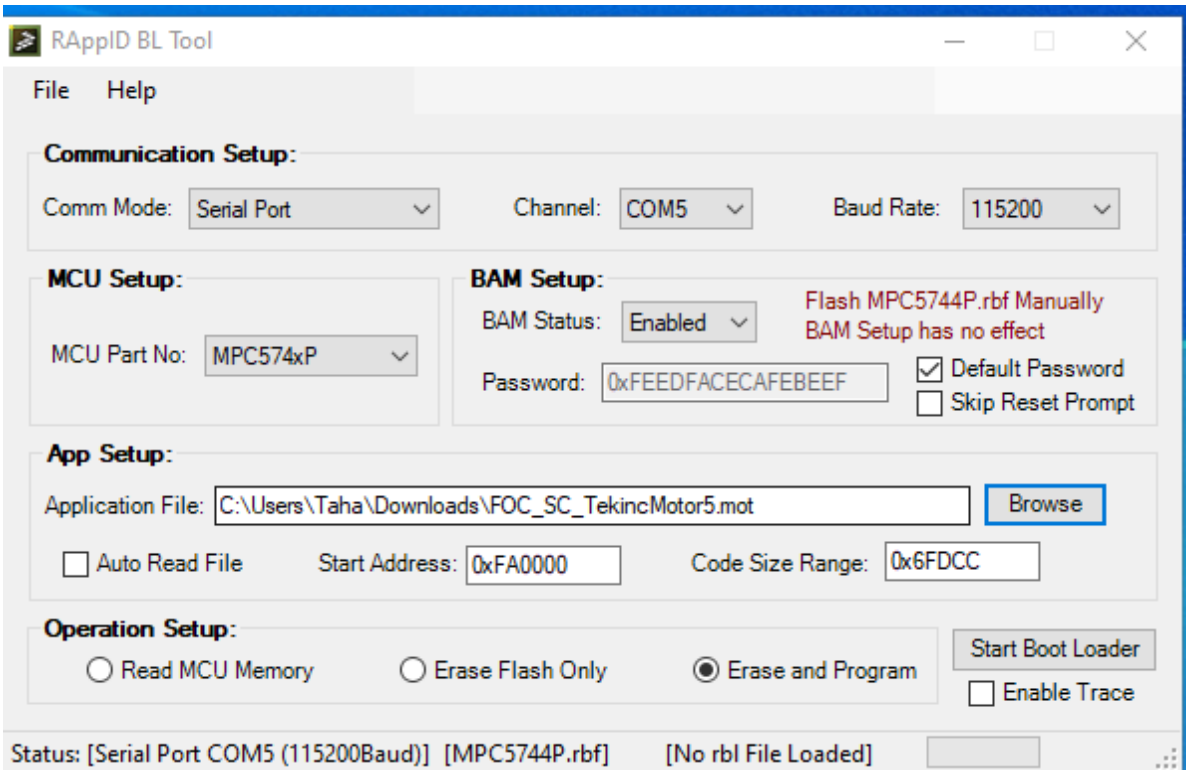


Fig 3.13: Configuration of Bootloader

- Adjust the channel field according to the COM of the board on your device and in the field of Application file browse for the .mot file. You will find it in the folder in this link : [The .mot File]
- **To allow the flashing, the MCU must be disconnected and reconnected**
- The flash should be done in the first 5 seconds after reconnecting the Board to the pc
- Press Start Bootloader

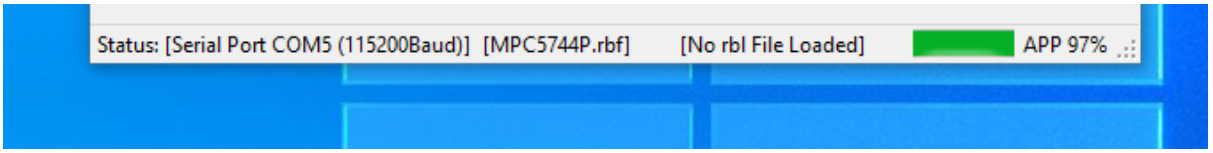


Fig 3.14:Uploading the .mot file to the MCU board

Note: If you fail on flashing the MCU :-
If It gives this Reply:

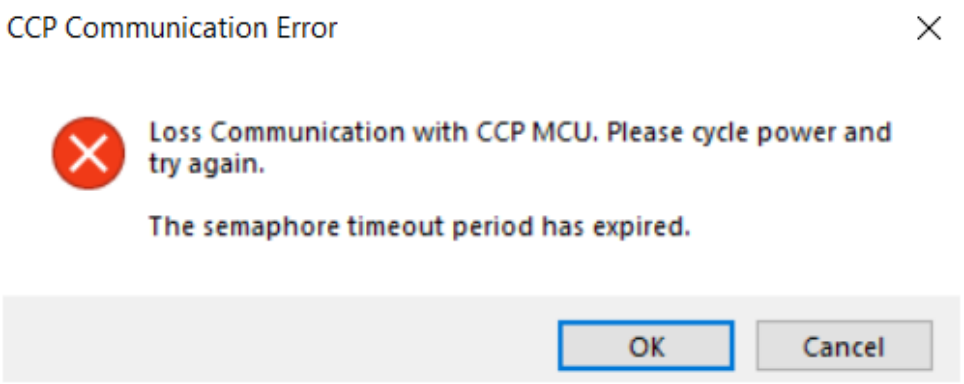


Fig 3.15:Communication Error

- **Remove the usb cable and close this window and reconnect the usb cable again and press startBootloader again within 5 seconds. It will start.**
- If after doing this, it doesn't start, refer to this link Problem with Bootloader

Note:Another way to upload the .mot file

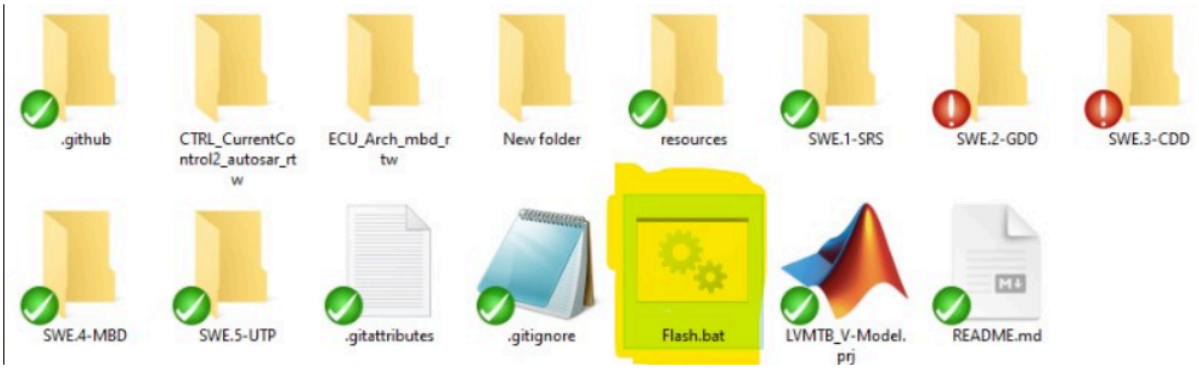


Fig 3.16: the .bat file used

a windows script that generated by the matlab in the rtw folder

this batch file is used to make the Simulink able to flash the code after the build. a script has been written to select the COM port and flash the

code without complex details or any knowledge about RAPPID.exe application

you need only 4 files :

1. Flash.bat
2. our_code.mot
3. our_code.rbl
4. our_code_bam.bat

the last 3 file already generated by matlab and you can get them from the rtw folder

you can connect with the user send the files to hem and flash the code without any extra steps only connect the board and double click.

after running this script :

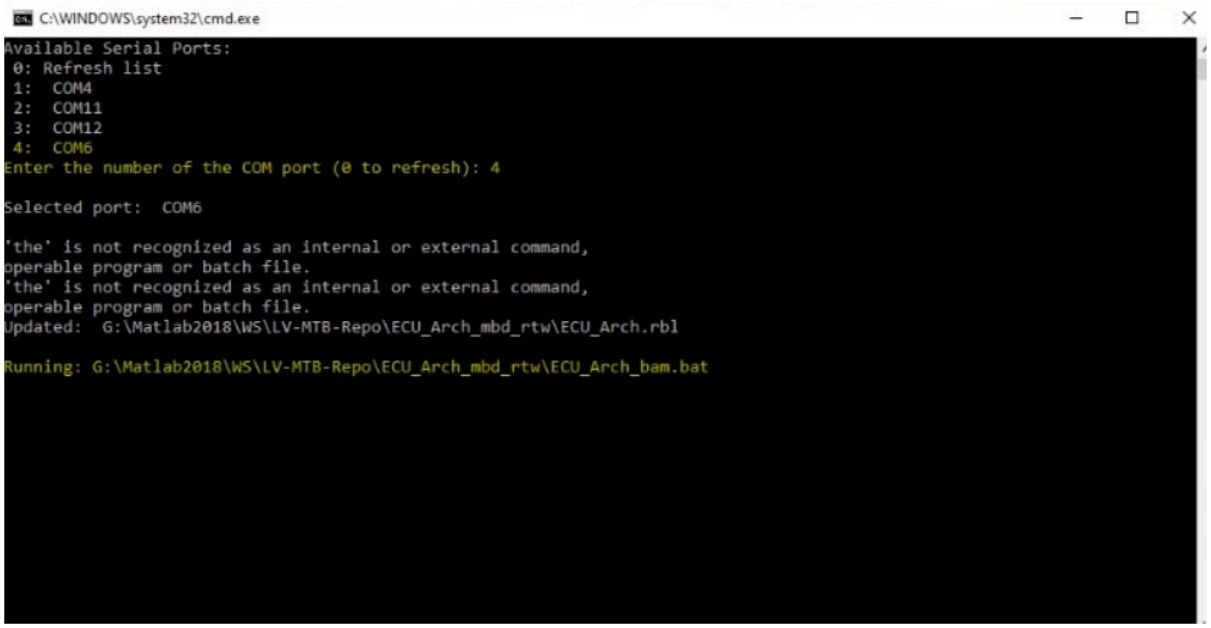


Fig 3.17:After Running the Script

the RAppID will automatically open to flash the mot file:

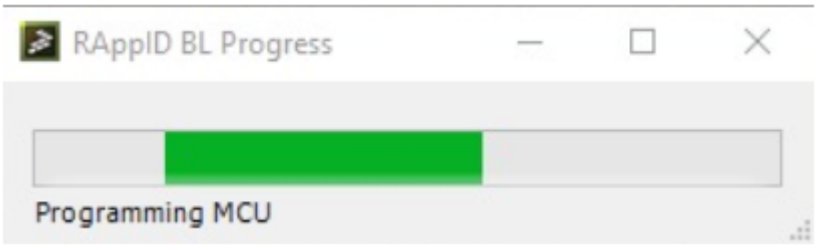


Fig 3.18:Flashing the .mot file automatically

3.1.5. Install FreeMaster Software:

3.1.5.1. System requirements

- The FreeMASTER application requirements can be easily met by almost any Windows OS-based host PC available today. It was
- tested with the latest versions of Windows OS (7, 8, and 10).
- Operating system: Microsoft Windows 7 or later
- Required software: Internet Explorer 10 or later
- Hard drive space: 400 MB (additional 500 MB while installing)
- Other hardware requirements: mouse, serial RS-232 or USB port for communication with the target board, network access for remote operation.

3.1.5.2. Installation Steps

- Download FreeMaster software from this link:Download FreeMaster and run the installer.

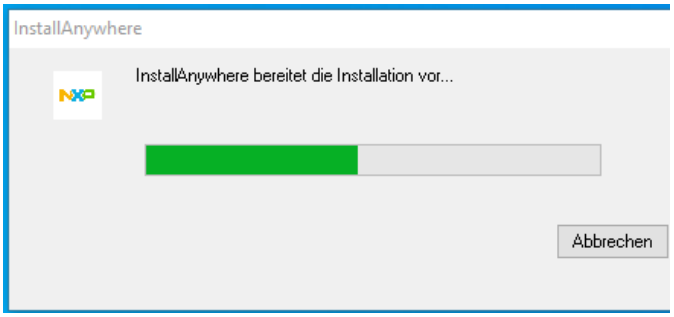


Fig 3.19 :Running the FreeMaster Installer

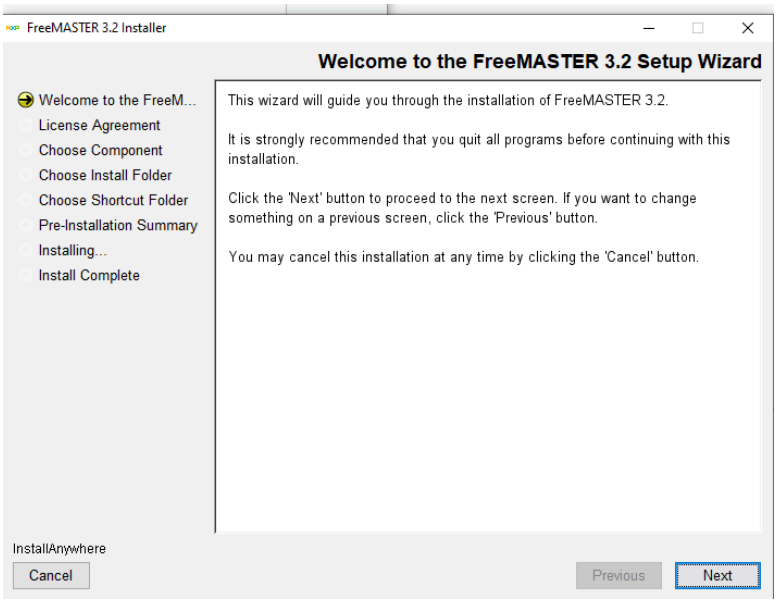


Fig 3.20 :FreeMaster Setup Wizard

Next next next

- Uncheck the option of installing FreeMaster Lite while setting up the package as shown in the figure below

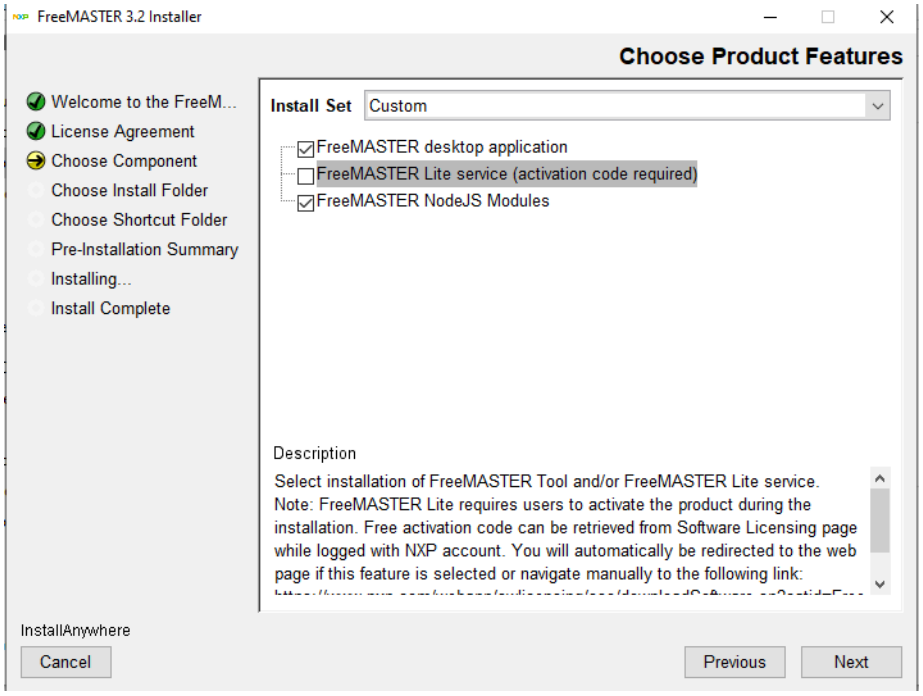


Fig 3.21 :Uncheck the option of freemaster lite

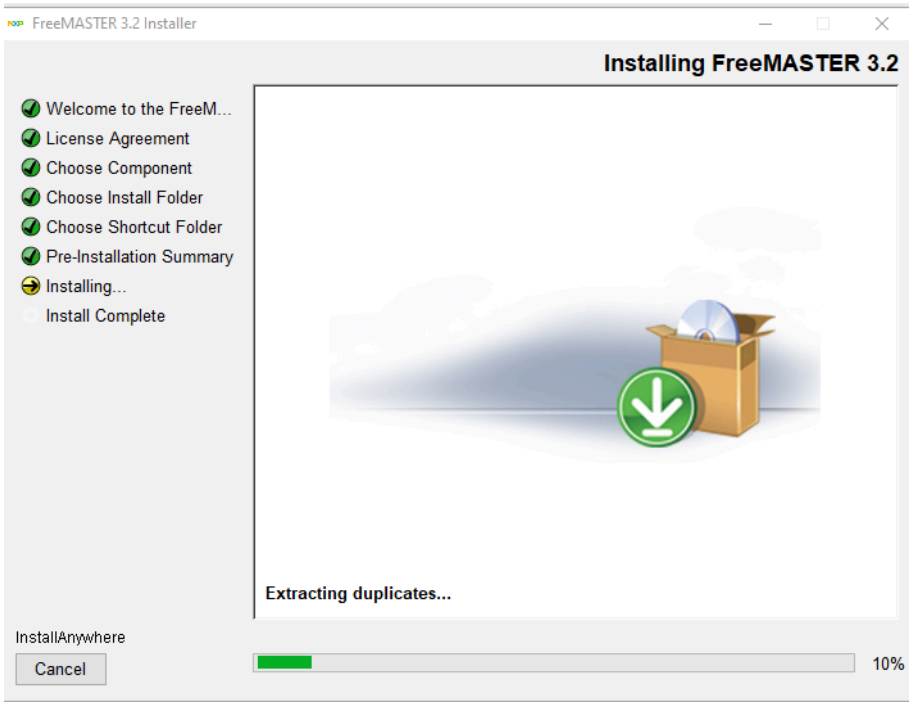


Fig 3.22 :Installing

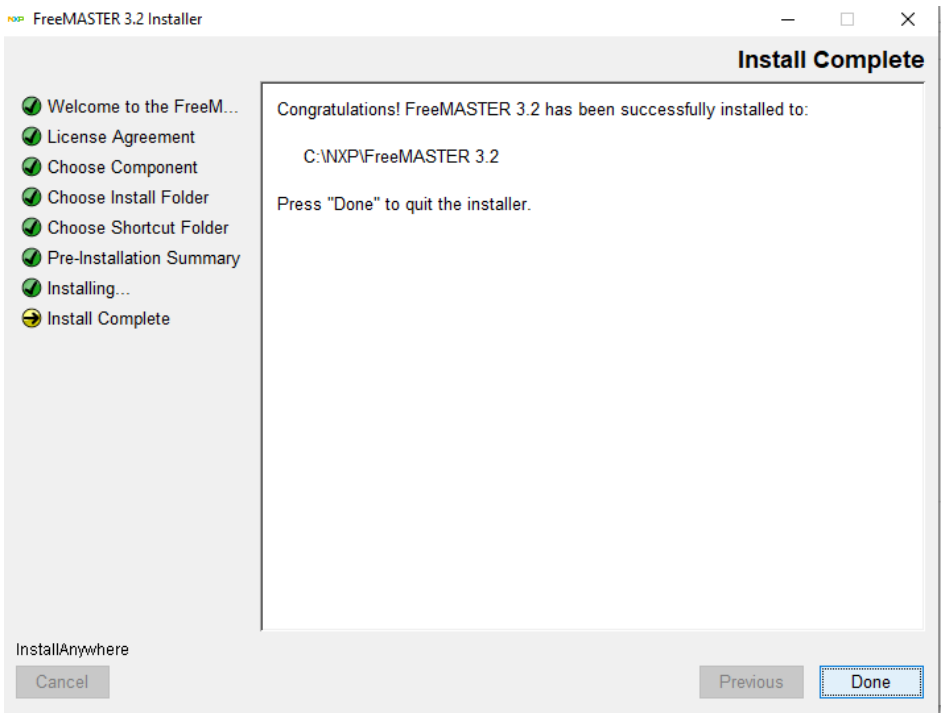


Fig 3.23 :Install Complete

- You can find FreeMaster User Guide in this link :[FreeMaster User Guide]

3.1.6. Repeat for the second MCU Board

- Repeat the steps described before on the other MCU Board
- Now you have the code uploaded on the MCU Board and you can start the verification process found in the next chapter

4. Initial Hardware Setup Verification using one Motor

4.1. Connect one MCU Board:

- Use USB cable to connect the MCU board to your PC or laptop.
- You should see a green light on the board.

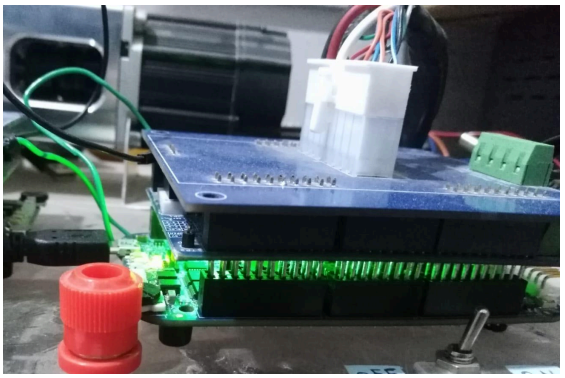


Fig 4.1: green light on the board

4.2. Inverter Board Preparation:

- Connect one inverter board to a DC power supply set to 12V and 5A.
- Keep the power supply switched off for now.

4.3. Openlopp test - Taha - to be placed

The test using open loop model
C:\Users\Taha\Downloads\VF_control_MP574xP_Bassel_02_TeknicMotor.mot
Leads to a rotatin motor at 500rpm and 2.5A and and AC voltage measured was 2.3V line to line (the jumper cable was used and is went melt (daouaban)

4.4. FreeMaster Project Preparation:

4.4.1. Open FreeMaster Project:

- Open FreeMaster project file (SpdControl) from this link. [.pmp files].

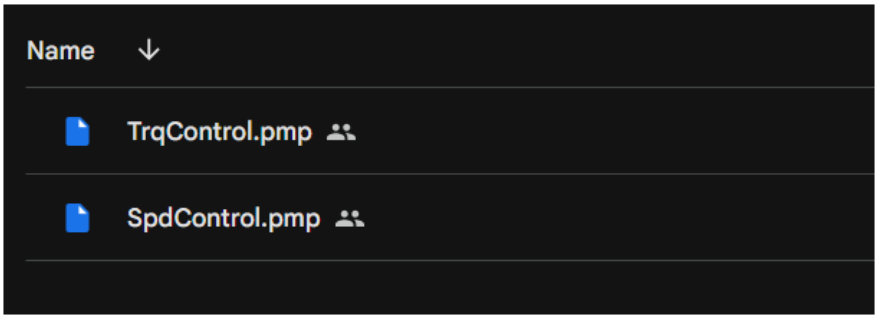


Fig4.2: FreeMaster project files

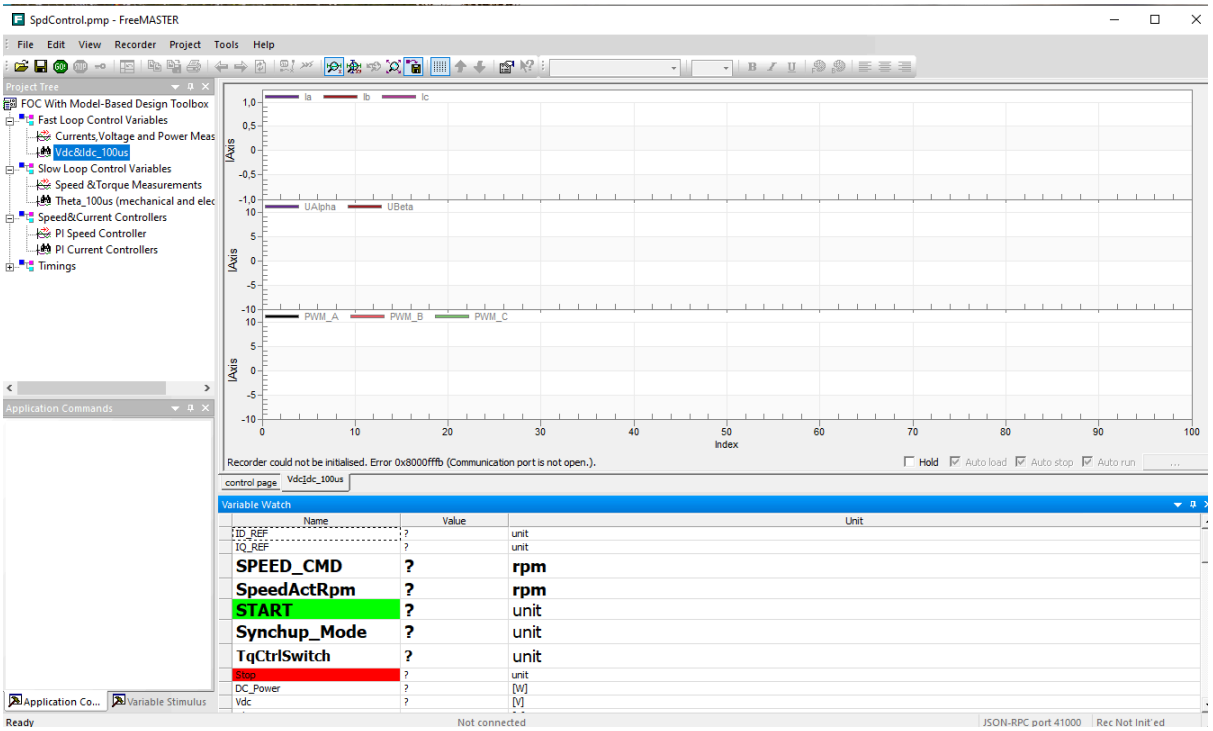


Fig 4.3: Entry Screen of FreeMaster

- Adjust FreeMaster Communication Settings:
  - In the FreeMaster instance, go to the "Project" menu, then from it choose “Options”
  - From “Options” window, open the "Comm" tab.

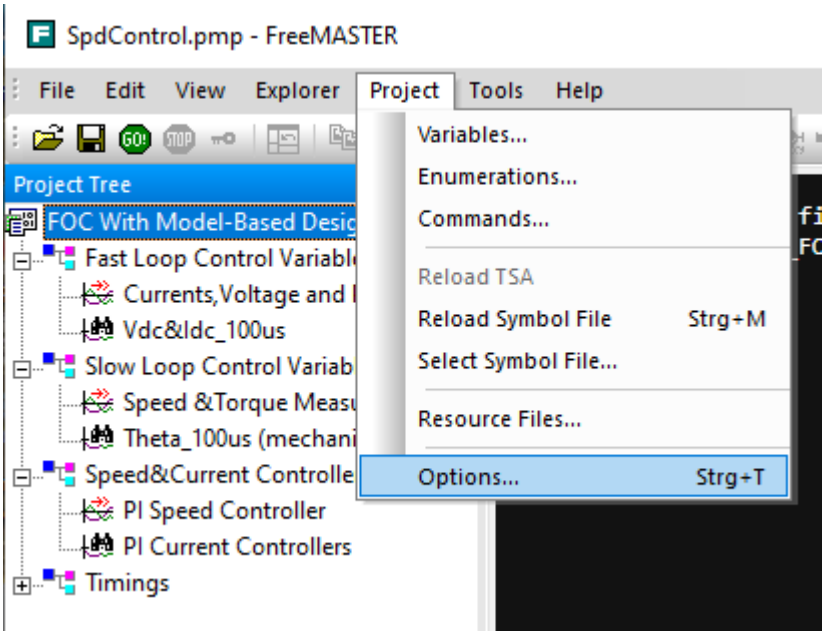


Fig 4.4: Project menu

Set the port number for the board based on its connection to your device, as shown in the figures below.

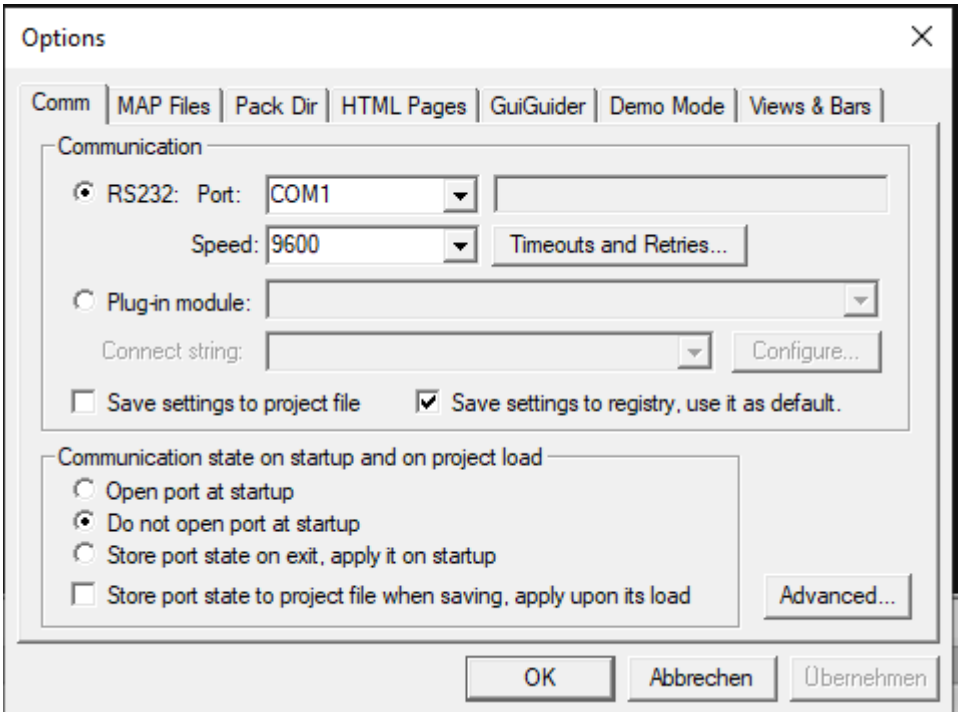
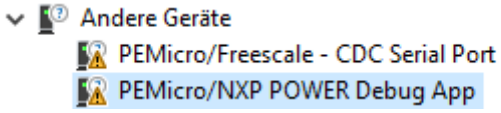


Fig 4.5: Options window from Project menu

4.4.2. Open Question from taha



- Should I change some info in the usb setting.
- You should download this driver from this link and install it

4.4.3. Add The .elf file

- In the freemaster instance, from the same “Options” window shown above choose the “MAP files” tab and browse for the . elf file found On This Link and add it as shown in the figure below.

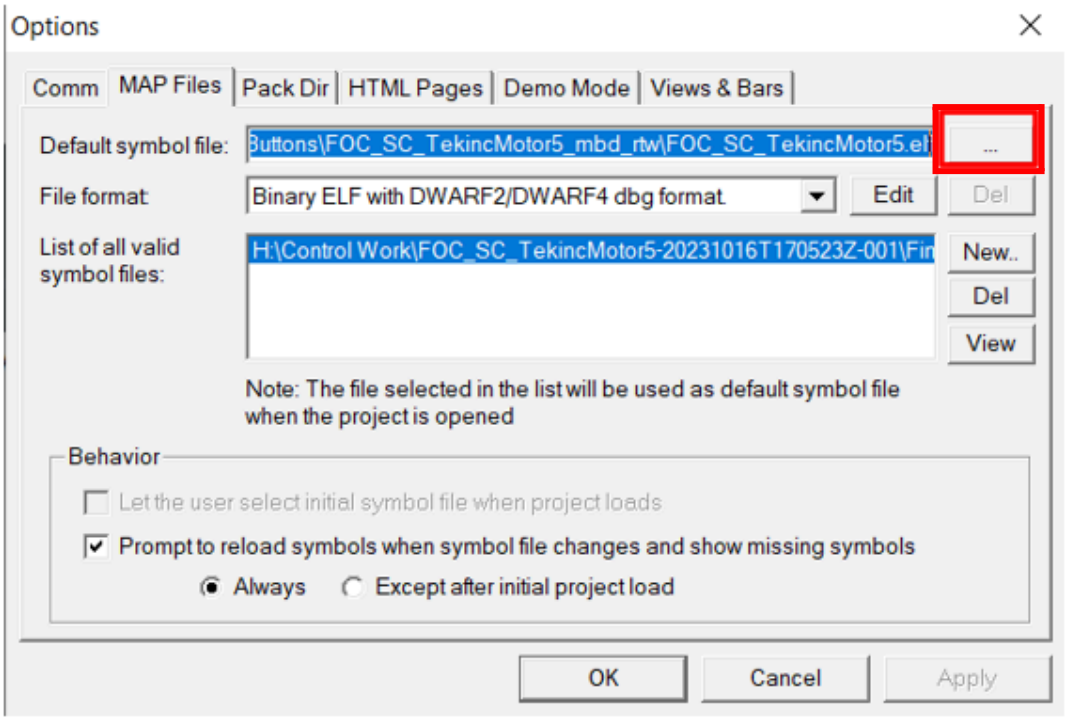


Fig 4.6: MAP file tab in Options window from Project menu

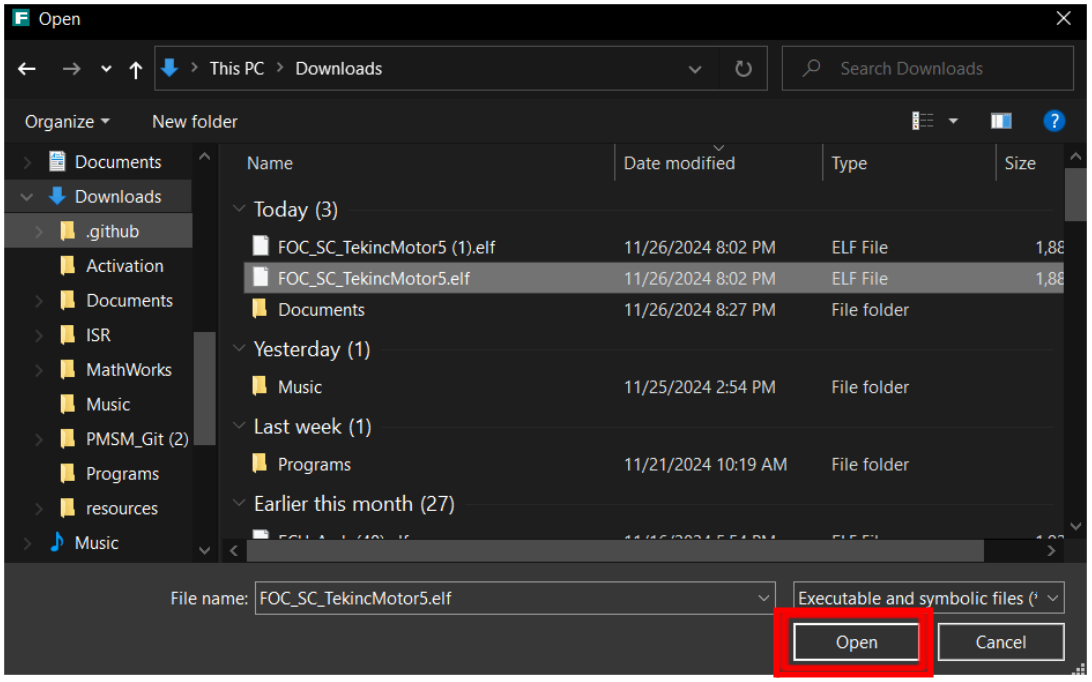


Fig 4.7: Browsing for the .elf file

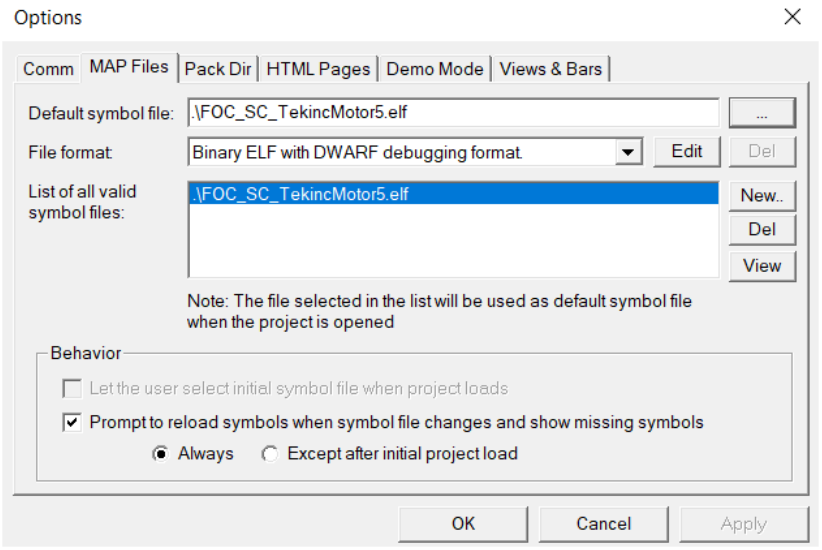


Fig 4.8: Adding .elf file in Options window from Project menu

4.4.4. Add The GUI File

- Download the .RAR file [in this link.]
- Extract the .RAR File to get a folder called “build”
- In the freemaster instance, from the same “Options” window shown above choose “HTML Pages” tab and browse for the .html file inside the “build” folder as shown and add it as shown in the figure below and then press OK.

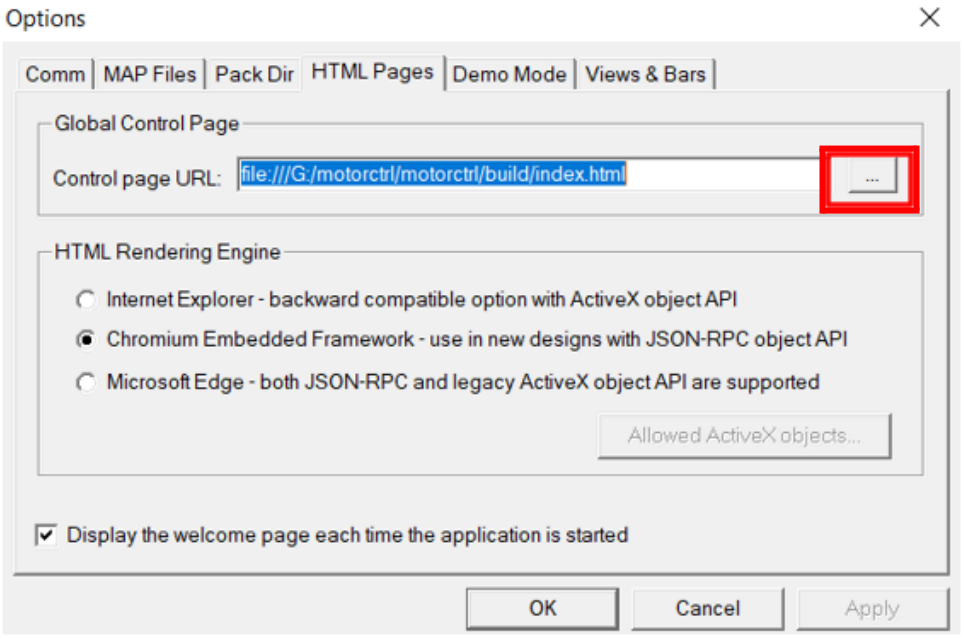


Fig 4.9: HTML Pages Tab in Options window from Project menu

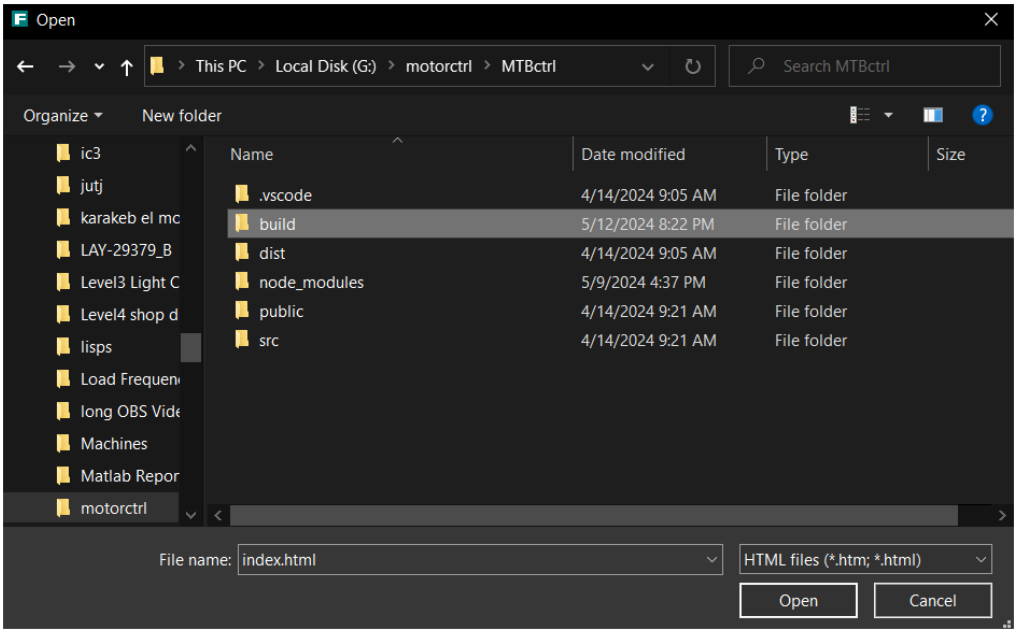


Fig 4.10: Browsing for the .html page

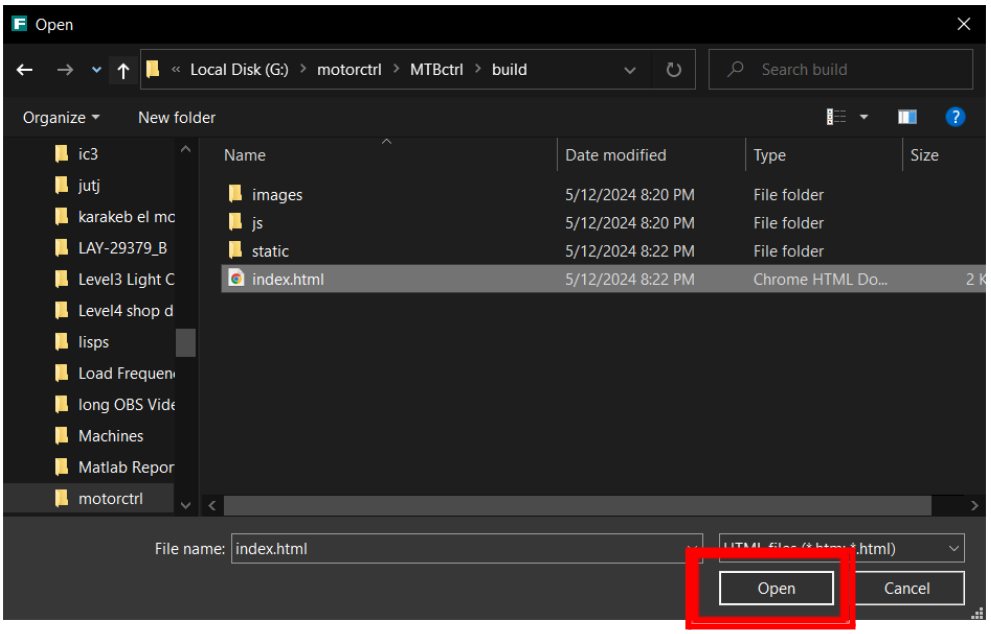


Fig 4.11: Continue Browsing for the .html page

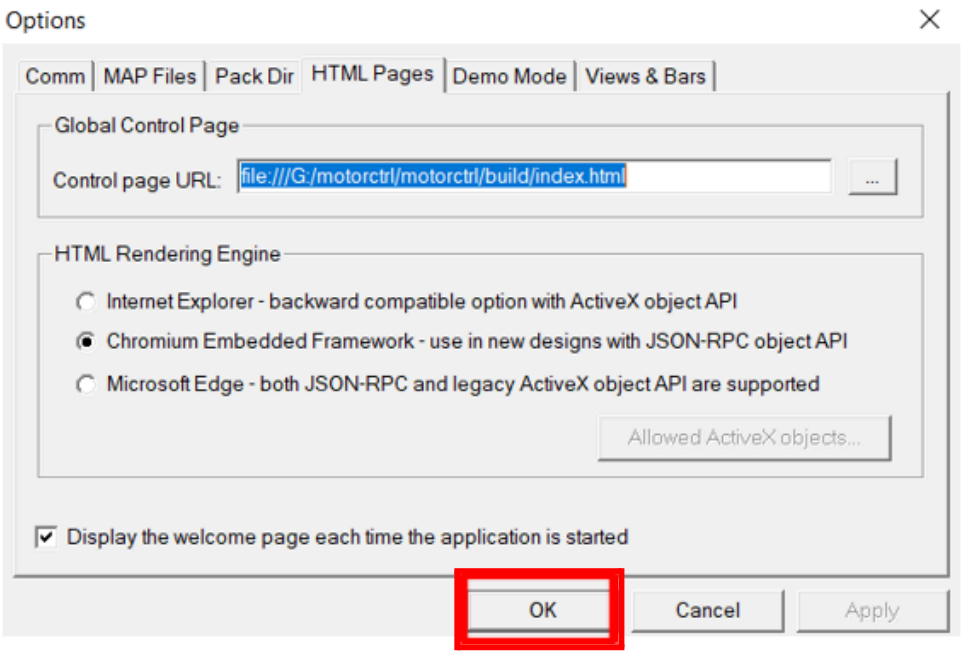


Fig 4.12: Adding the .html page

- When this menu shown below appears check the second option and press Continue as shown below

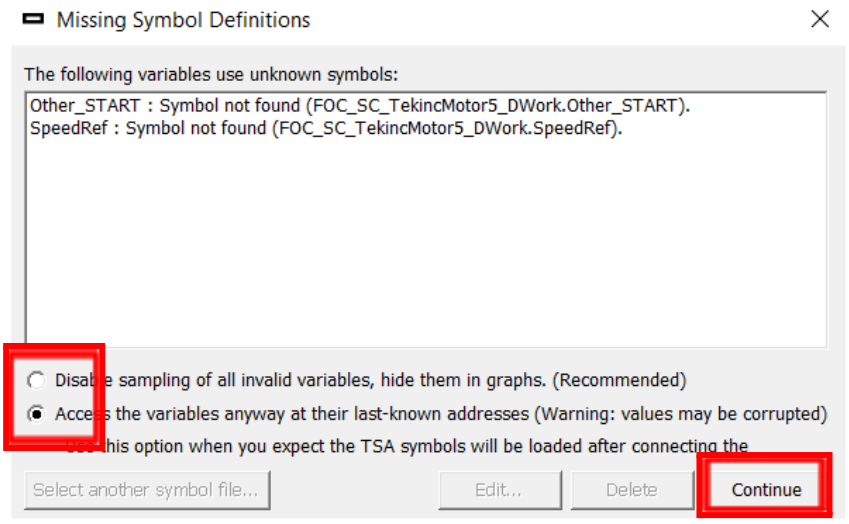


Fig 4.13: Missed Symbol Definitions

4.4.5. Save the configurations

- so as not to repeat these steps next time, save the configurations

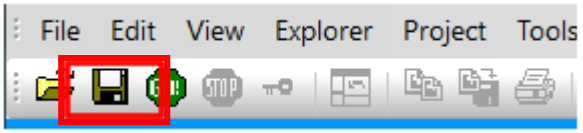
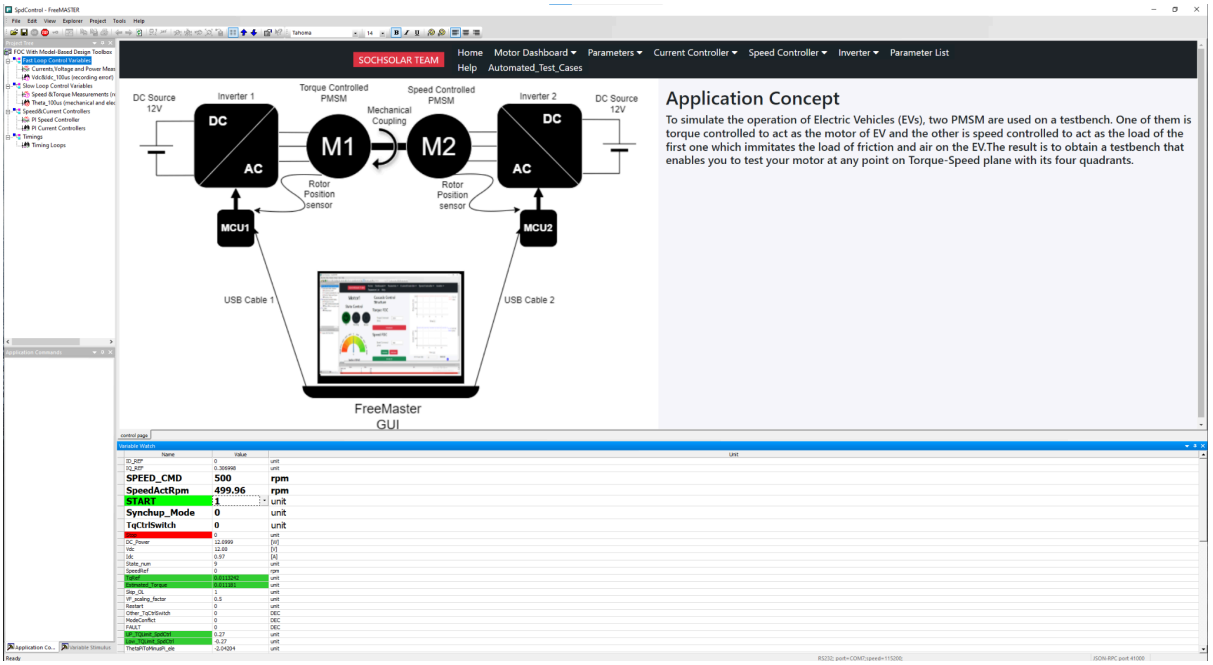


Fig 4.14: Saving Configurations

4.5. Spinning the Motor

4.5.1. Freemaster



Open Freemaster
Top left open project
Use project xxx
(this is how user manual is done)

4.5.2. Press “GO” button

In the top left corner of the screen.

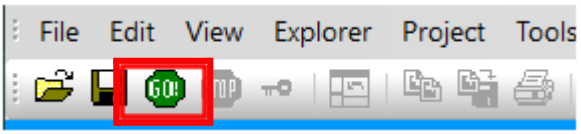


Fig 4.15: GO Button

<u>Note:</u>Click on “Go” only if the green led on the inverter is on.
If you clicked the “Go” button before the green led is lighting, then you get the error shown below and you have to click “close Port”, unplug the usb cable and plug it again to restart.

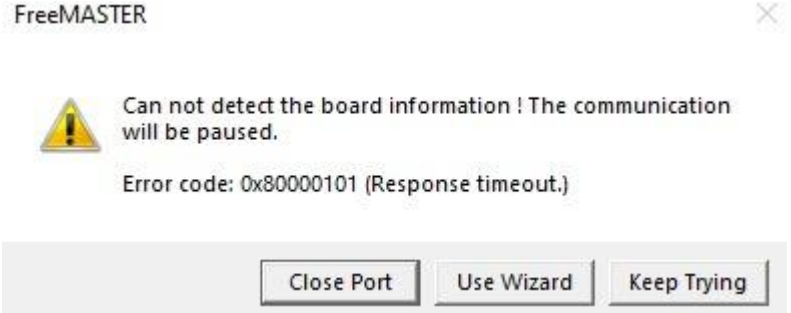


Fig 4.16: Board Detection Error

4.5.3. Adjust the Synchup mode

- Assign the variable called “Synchup_Mode” to 0

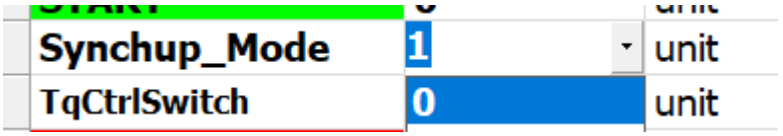


Fig 4.17: Sunchup_Mode

4.5.4. Adjust the Control Mode

- Adjust the motor to be speed controlled by either:
  - Making the button under speed command enabled in the GUI
  - Giving value of 0 to the variable called “TqCtrlSwitch” in FreeMaster as shown in the figure below.

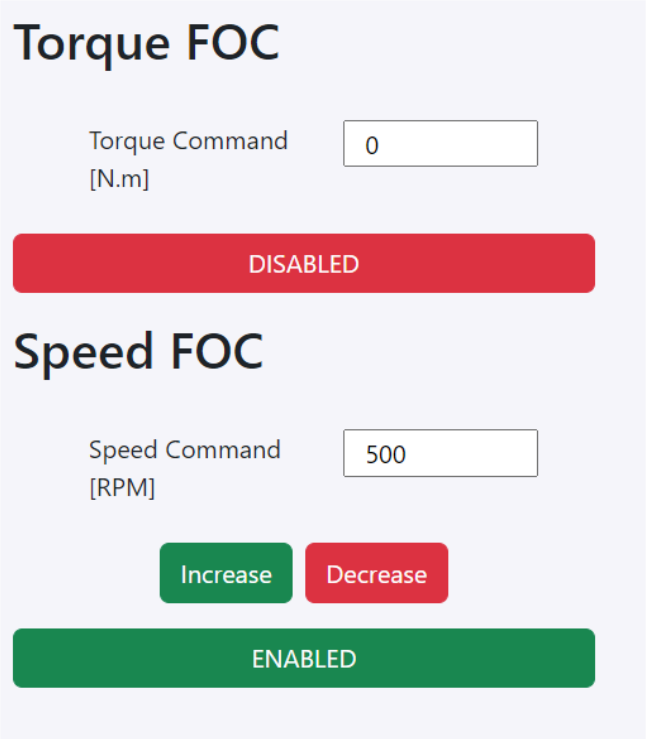


Fig 4.18: Choosing Speed Control Mode In GUI

Fig 4.19: Choosing Speed Control Mode In FreeMaster Interface

4.5.5. Enter the desired command

- For the speed controlled motor, enter the desired speed command

4.5.6. Switch on the power supply

When you switch on the power supply, you will see a yellow light from a small led on the inverter board].

4.5.7. Push START button

- In the GUI, push start buttons and the operation will start.
- You can also put the variable called “Start” =1 in the FreeMaster interface as shown.

Fig 4.20:Starting the operation In GUI

Fig 4.21:Starting the operation In FreeMaster Interface

4.6. Observation

- After pressing “START”, you will pass through the following stages:
  - You will see a blue light for moments as shown

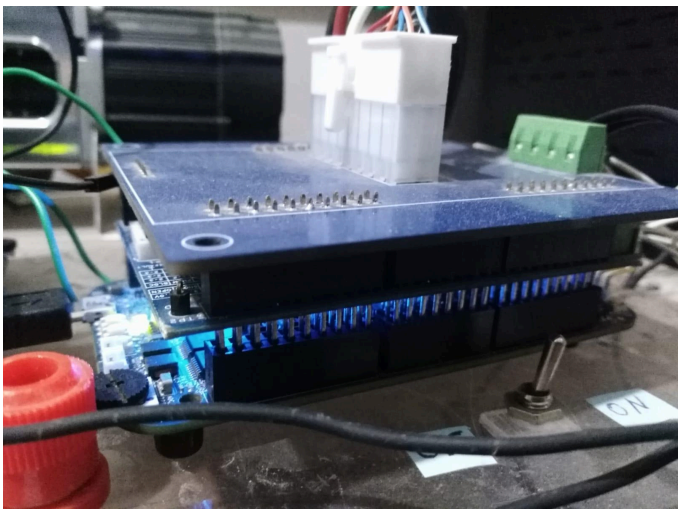


Fig 4.22:Blue Light after pressing START

- Then you will see a blinking green light which indicates the **Alignment state** and the shaft of the motor will be attracted and aligned

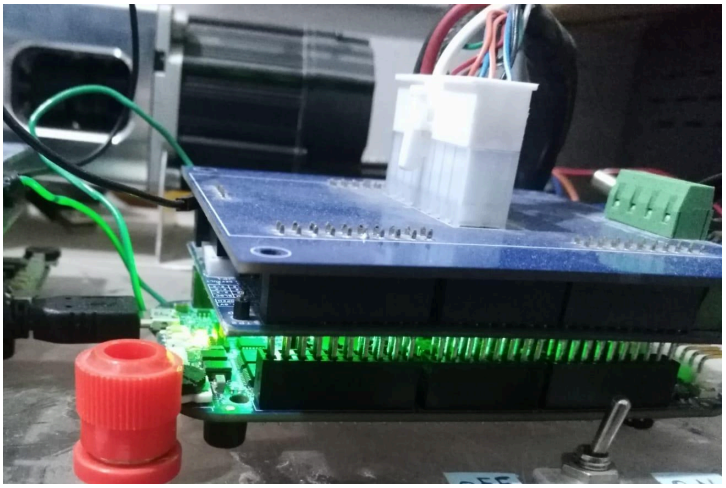


Fig 4.23:Blinking Green Light during Alignment

- Then you will find the blue light again for **Normal Operation State** and the motor will rotate at the commanded speed which is 500 rpm by default.

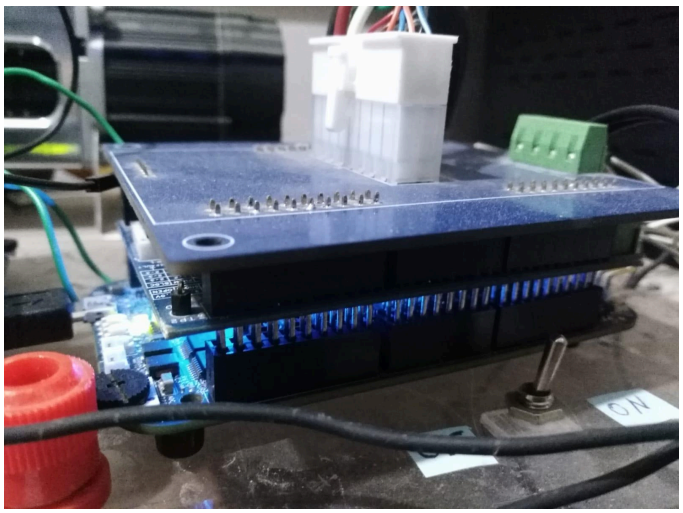


Fig 4.24: Blue Light during Normal Operation

- Now you can vary the Command Speed Either in GUI or in FreeMaster as shown and watch the motor following the new commanded speed
  - In GUI Write the Commanded speed in the field of speed command as shown

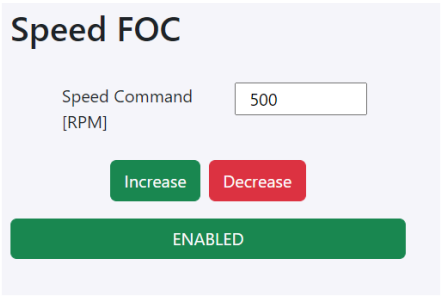


Fig 4.25:Controlling speed from GUI

  - In FreeMaster Interface assign the required speed value to the variable called “SPEED_CMD” as shown

Fig 4.26:Controlling speed from FreeMaster Interface

4.7. If you want to repeat the test:

- If the motor is rotating, first, activate the stop variable in FreeMaster as shown to stop the motor rotation gradually:

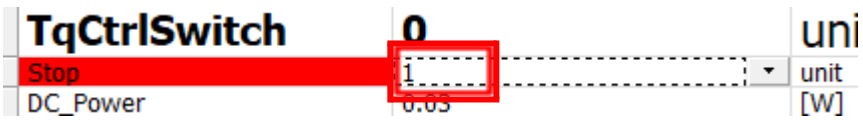


Fig 4.27:STOP Variable in FreeMaster

- Remove the USB cable

- Turn off the power supply
- Connect the USB cable again
- Click stop in FreeMaster as shown

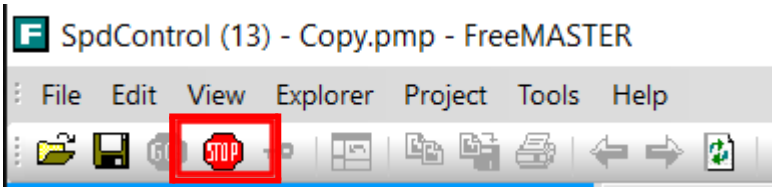


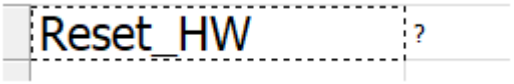
Fig 4.28:STOP Button in FreeMaster toolbar

- Repeat steps from 4.4

4.8. Repeat The Previous Steps for the other board

NOTE: RESET the SW without unplug the USB cable:

To reset the board without removing the usb cable and replugging it again, you can use the feature of software reset by activating the variable Reset [give it 1], you will see the board restarting in the same behaviour of replugging the usb cable and you won't even need to push the "GO" button in FreeMaster again.



[This feature can be used-till now- only if the elf file is the uploaded file on the MCU not the mot file. ELF file can be uploaded using "Design Studio DS32" from NXP.]
[Steps of installation of DS32 can be found in this document
S32DS software installation steps , steps of uploading the elf file using DS32 can be found in this document……]

5. Running Test Bench with Two Motors

5.1. Connect the MCU Boards:

- Use USB cables to connect the two MCU boards to your PC or laptop.
- You should see a green light on both boards.

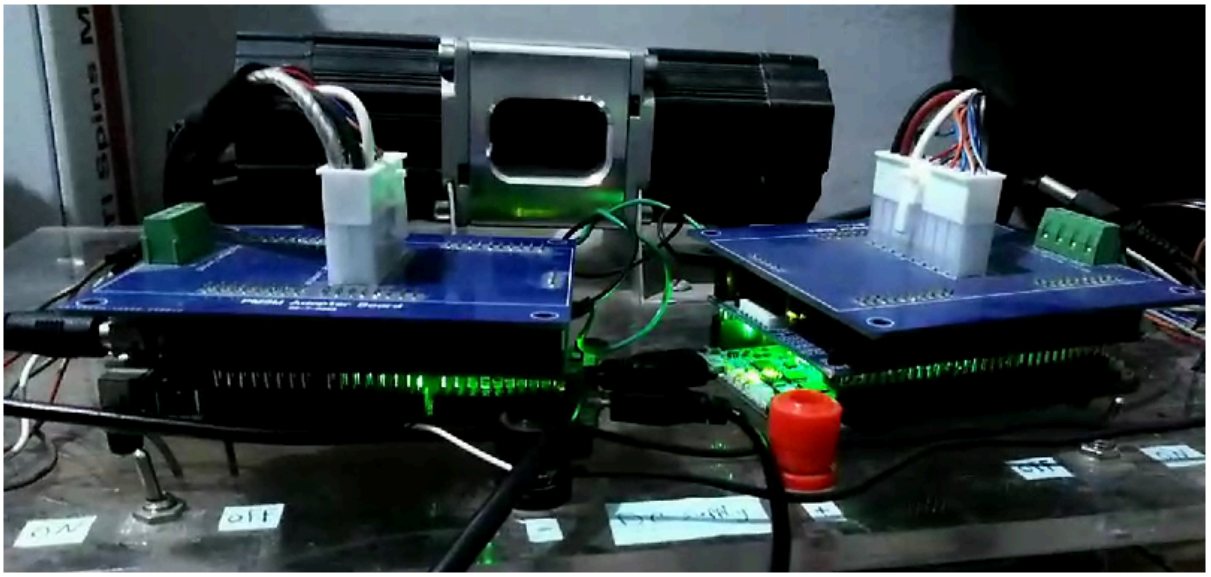


Fig 5.1: green light on the two boards

5.2. Inverter Boards Preparation:

- Connect the two inverter boards to a DC power supply set to 12V and 5A.
- Keep the power supply switched off for now.

5.3. FreeMaster Projects Preparation:

5.3.1. Open FreeMaster Projects:

- Open the two FreeMaster project files (.pmp) from this link. [.pmp files] Each project controls and monitors one motor.

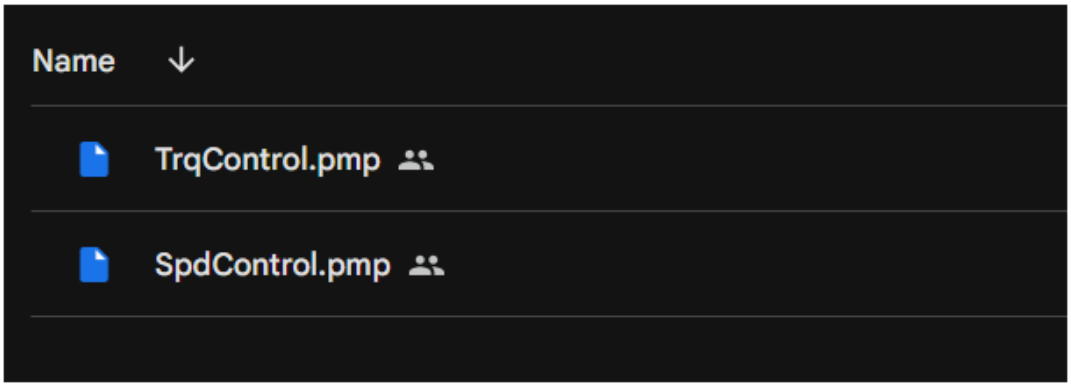


Fig 5.2:FreeMaster project files

- Adjust FreeMaster Communication Settings:
    - In each FreeMaster instance, go to the "Project" menu, then open the "Comm" tab.

Set the port number for each board based on its connection to

your device, as shown in the figures below.

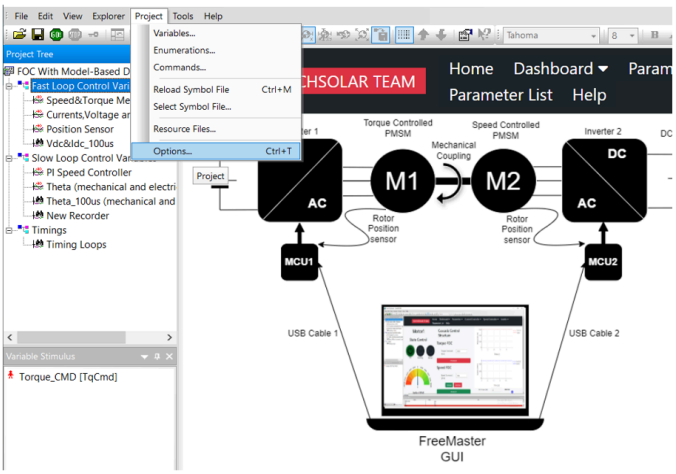


Fig 5.3: Entry Screen of FreeMaster

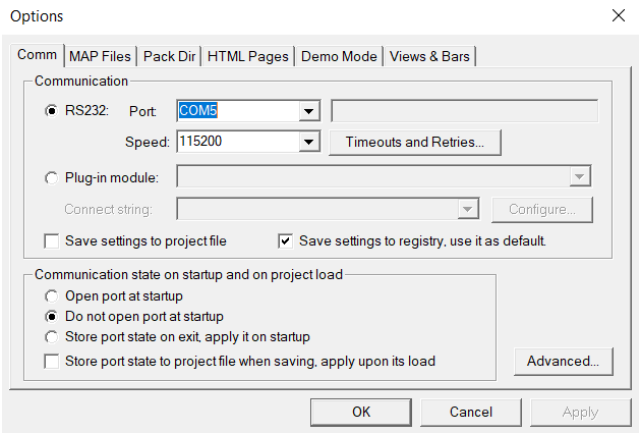


Fig 5.4: Options window from Project menu

5.3.2. Add The .elf file

- Download the .elf file found On This Link
- In each freemaster instance, from the same “Options” window shown above choose the “MAP files” tab.
- From the three points shown in the figure below, browse for the .elf file and add it as shown.

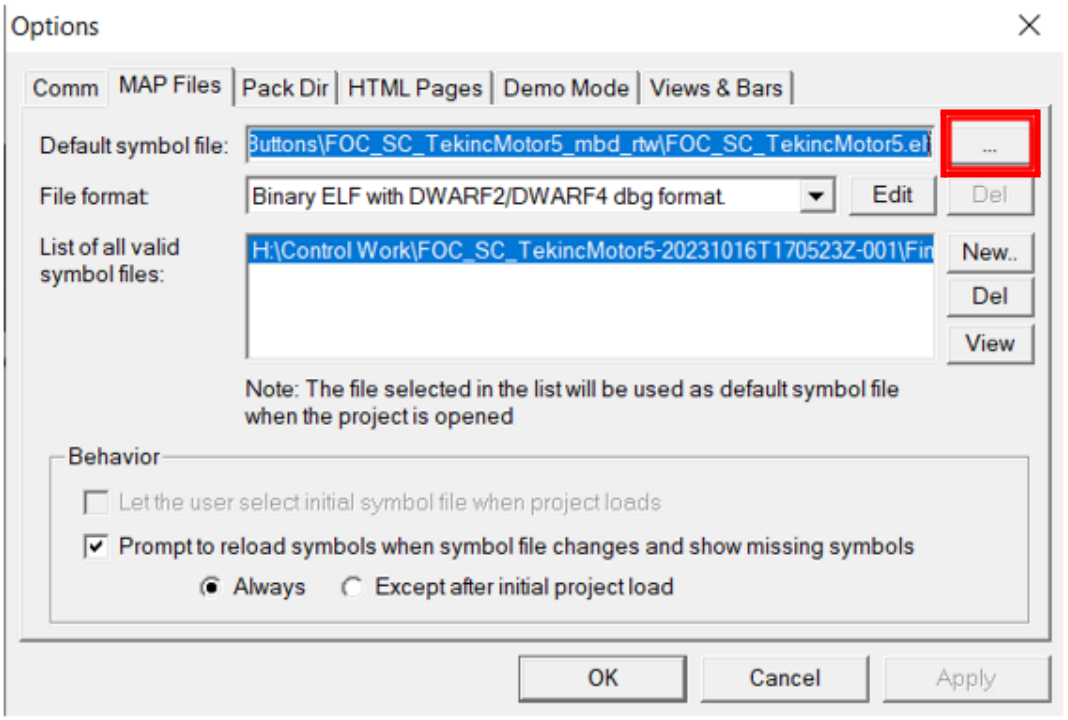


Fig 5.5: Adding .elf file in Options window from Project menu

5.3.3. Add The GUI File

- In each freemaster instance, from the same “Options” window shown above choose “HTML Pages” tab and browse for the .html file (On This Link) and add it as shown in the figure below and then press OK.

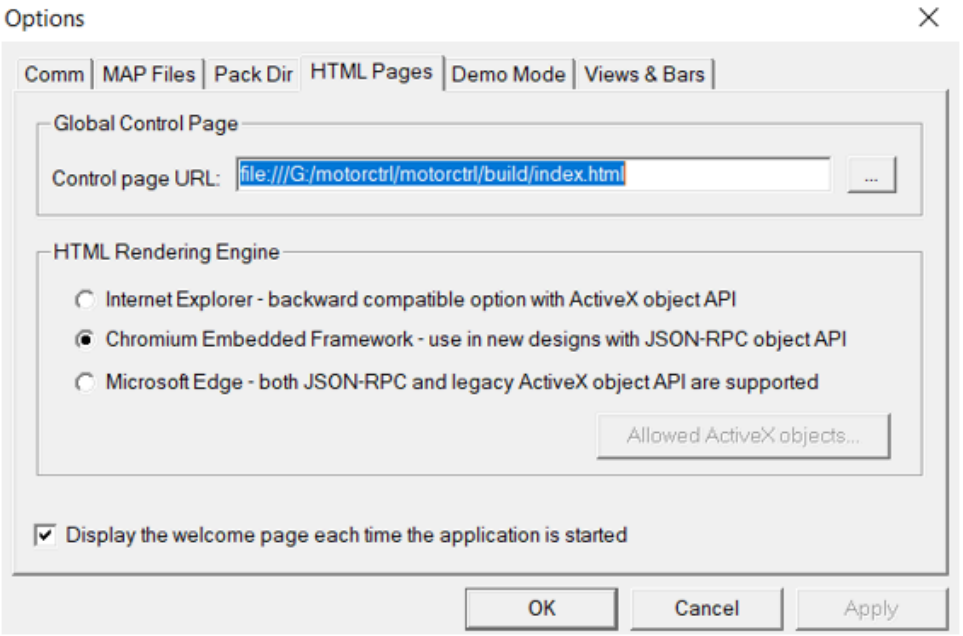


Fig 5.6: Adding GUI file in Options window from Project menu

5.3.4. Save the configurations

So as not to repeat these steps next time, save the configurations.

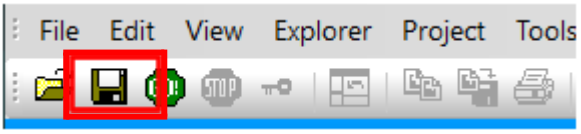


Fig 5.7: Saving Configurations

5.4. Physical Connections

- Make two physical connections between pin A14 in each board to pin G10 in the other board before starting the B2B operation so as to send and receive the values of Align_dn-out and Align_dn_in variables respectively.
- Find the places of the mentioned pins in the figure below in Rev E columns

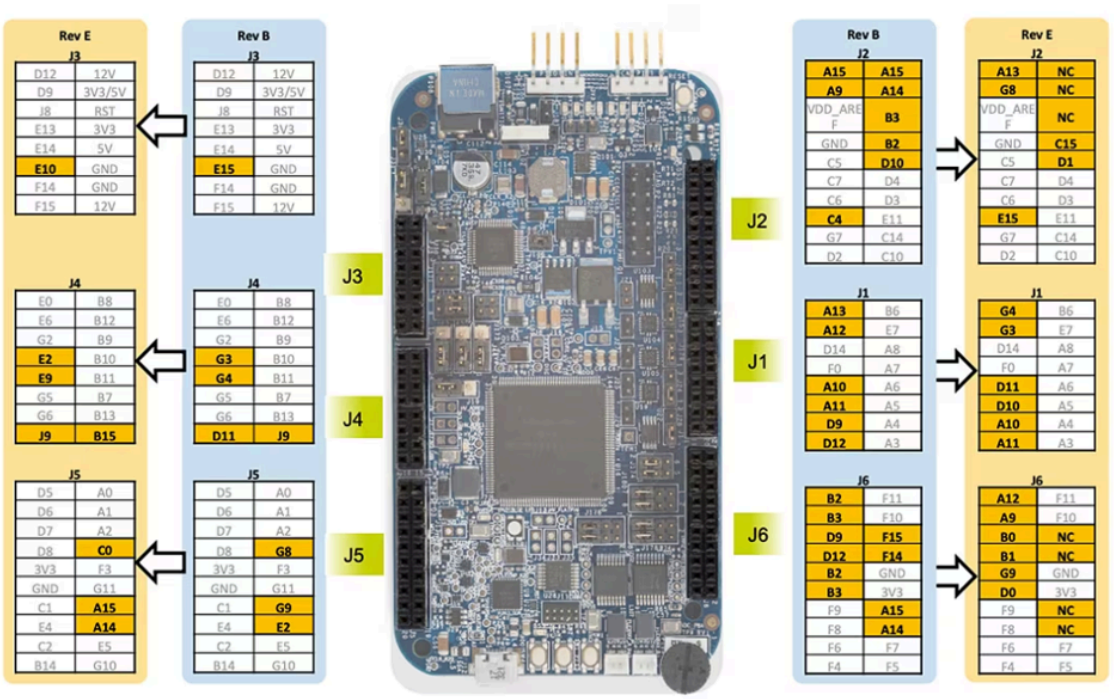


Fig 5.8: Pinout of the MPC5744P Board

5.5. Spinning the Two Motors

5.5.1. Press “GO” button

In the top left corner of the screen in the two freemaster instances as in the figure below.

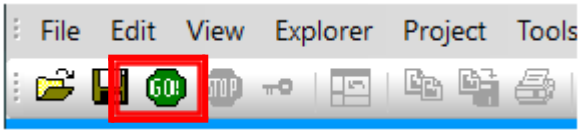


Fig 5.9: GO Button

5.5.2. Adjust the Control Mode

- Adjust one of them to be torque controlled [by making the button under torque command enabled in the GUI or by giving 1 to TqCtrlSwitch variable in FreeMaster as shown in the figure below.]

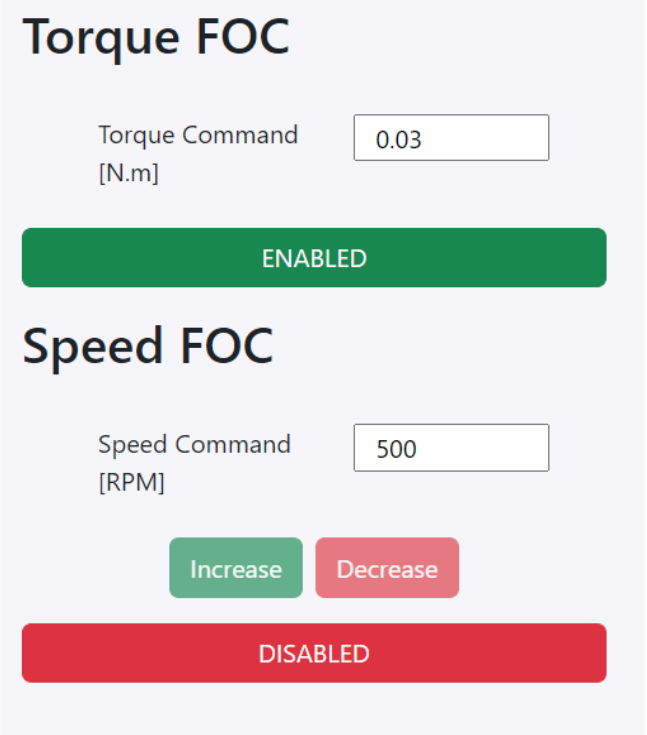


Fig 5.10: Choosing Torque Control Mode In GUI

Fig 5.11: Choosing Torque Control Mode In FreeMaster Interface

- Adjust the other one to be speed controlled [by making the button under speed command enabled in the GUI or by giving 0 to TqCtrlSwitch variable in FreeMaster as shown in the figure below.]

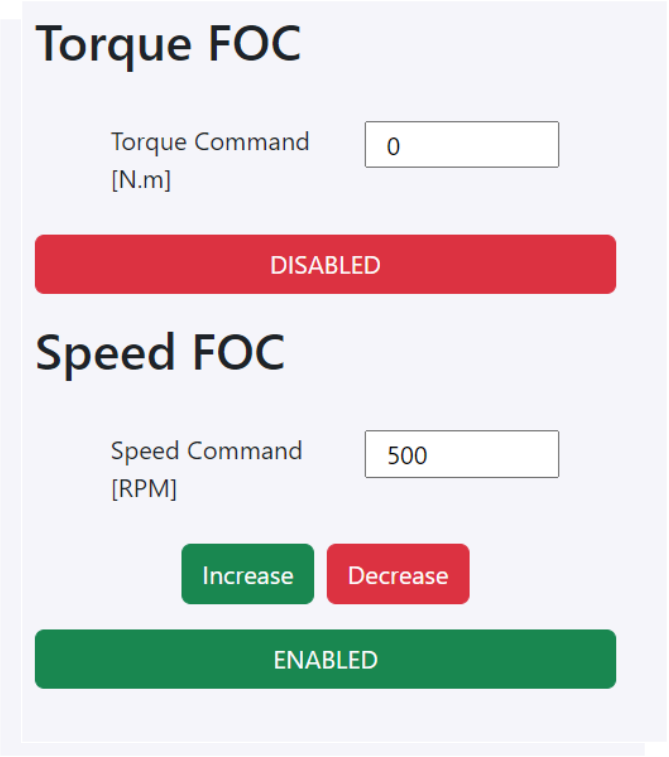


Fig 5.12: Choosing Speed Control Mode In GUI

Fig 5.13: Choosing Speed Control Mode In FreeMaster Interface

- **<u>Note:</u> You can't run both motors in the same mode. This will be considered as a Mode Conflict and a warning message will appear on the GUI and the motors won't rotate.**

5.5.3. Enter the desired command

- For the speed controlled motor, enter the desired speed command and for the torque controlled motor.
  Enter the desired torque command

5.5.4. Switch on the power supply

- After this, you will see a yellow light from a small led on each inverter board.

5.5.5. Push START button

- In both GUIs, push start buttons and the operation will start.
- You can also put the Start variable =1 in both FreeMaster instances as shown.

Fig 5.14:Starting the operation In FreeMaster Interface

5.6. Observation

- After pressing “START”, you will pass through the following stages:
  - You will see a blue light for moments from the two boards as shown

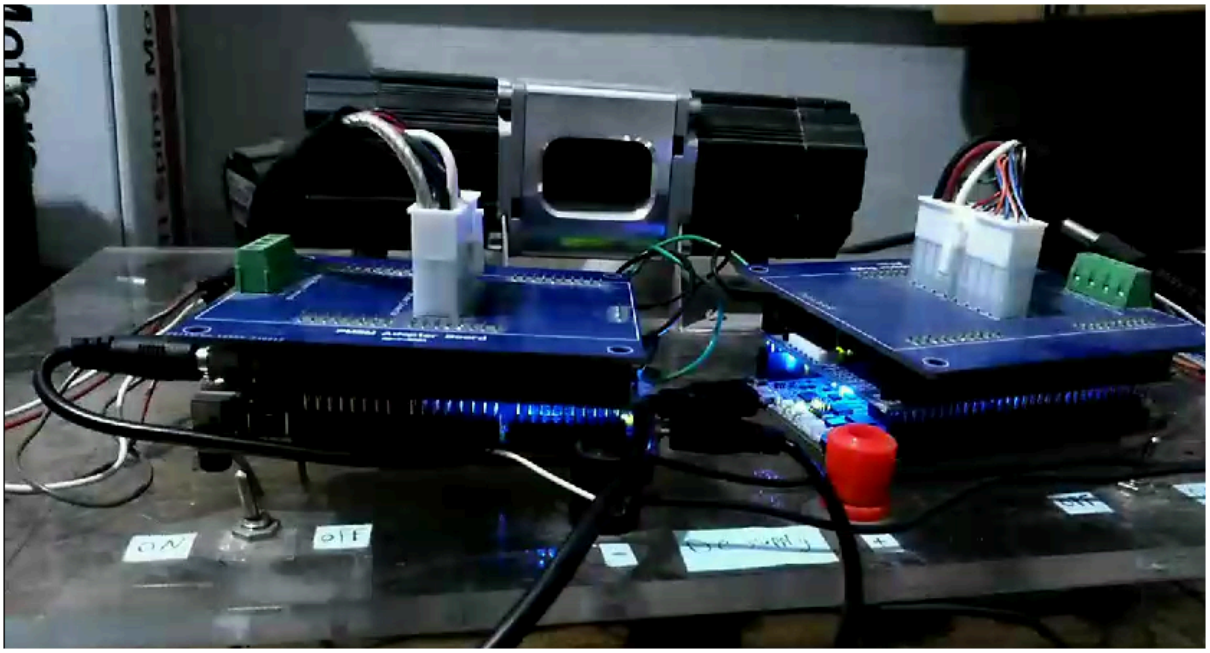


Fig 5.15:Blue Light after pressing START

  - Then you will see a blinking green light from one board which indicates the **Alignment state** and the shaft of the motor will be attracted and aligned while the other board still blue

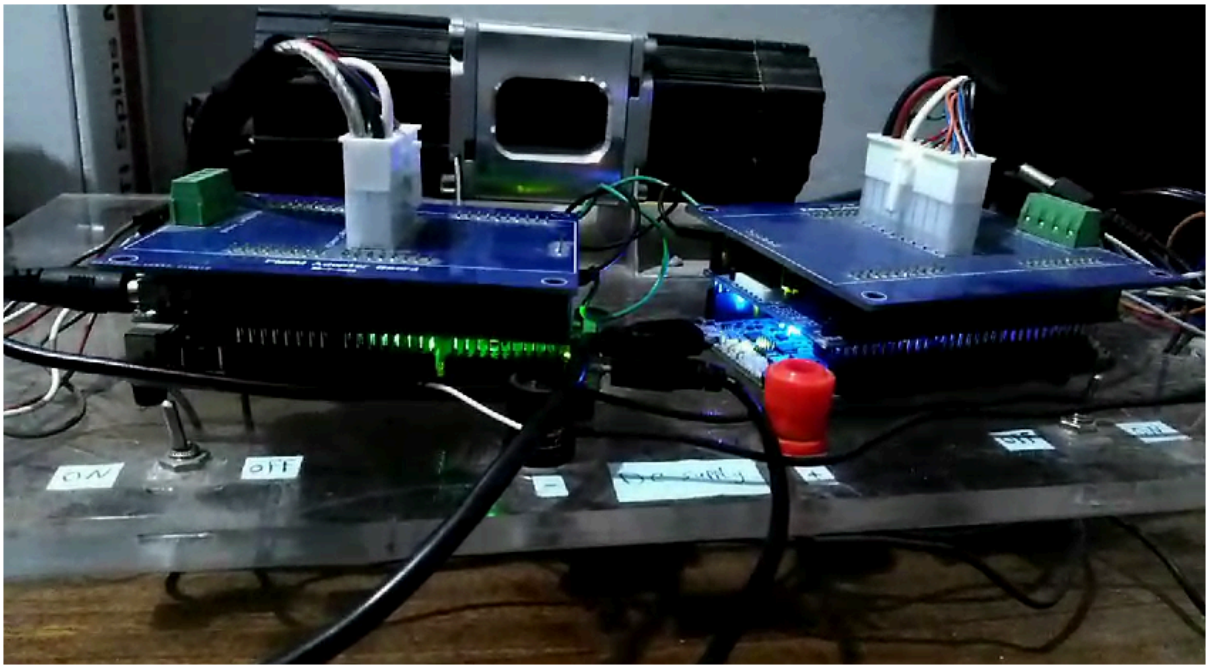


Fig 5.16:Blinking Green Light during Alignment of one of the two motors

- Then you will see a blinking green light from the other board which indicates the **Alignment state for the other motor** and the shaft of the motor will be attracted and aligned.

Fig 5.17:Blinking Green Light during Alignment of the other motor

- Then you will find the blue light again from the two boards for **Normal Operation State** and the motors will rotate at the commanded speed which is 500 rpm by default and at the commanded torque 0.03 N.m.

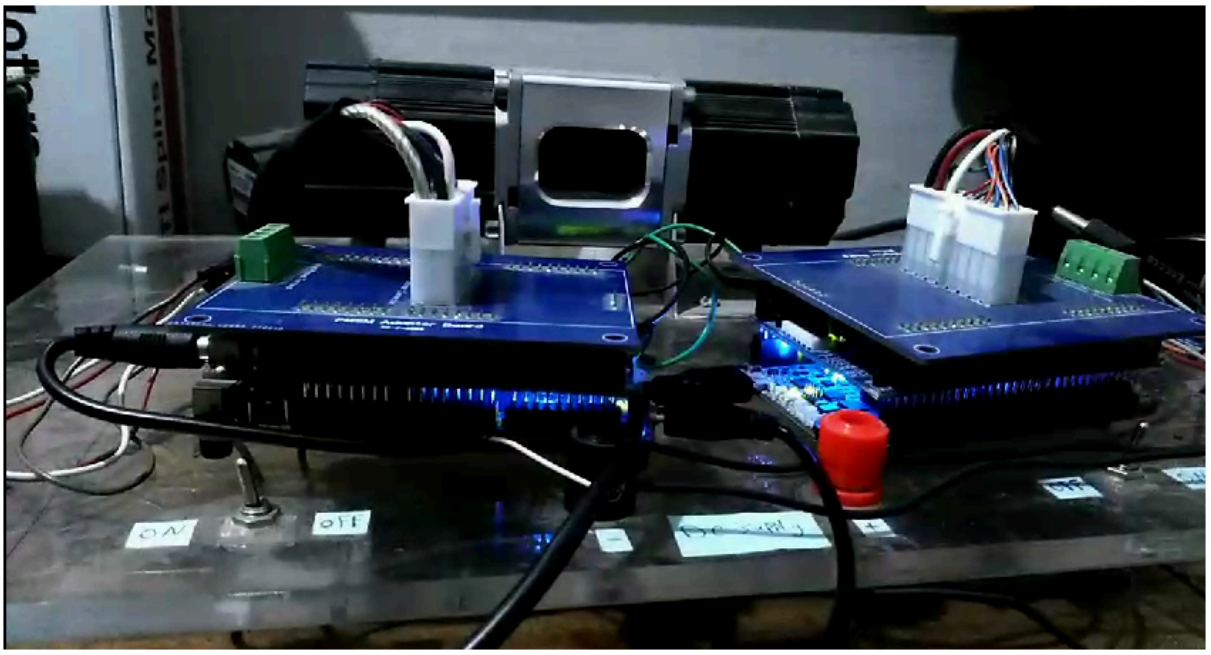


Fig 5.19: Blue Light from the two boards during Normal Operation

- Now you can vary the Command Speed Either in GUI or in FreeMaster as shown and watch the motor following the new commanded speed
  - In GUI Write the Commanded speed in the field of speed command as shown

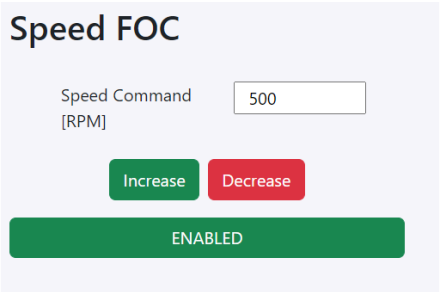


Fig 5.20:Controlling speed from GUI

  - In FreeMaster Interface assign the required speed value to the variable called “SPEED_CMD” as shown

Fig 5.21:Controlling speed from FreeMaster Interface

5.7. If you want to repeat the test:

- If the motors are rotating, first, activate the stop variable in both FreeMaster instances as shown to stop the motors rotation gradually:

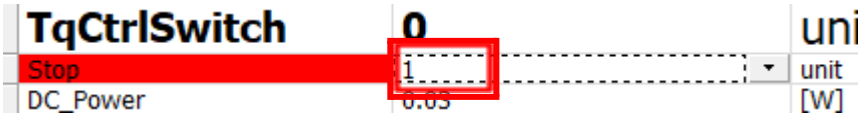


Fig 5.22:STOP Variable in FreeMaster

- Remove the two USB cables
- Turn off the power supply
- Connect the two USB cables again
- Click stop in both FreeMaster instances as shown

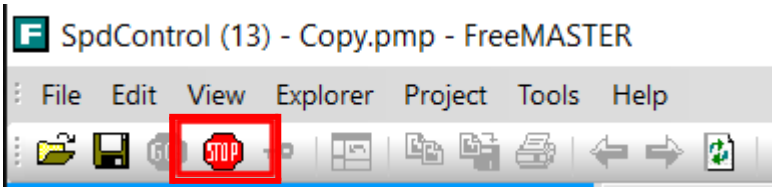


Fig 5.23:STOP Button in FreeMaster toolbar

- Repeat steps from 5.5

6. Watched Variables via FreeMASTER

6.1. Most Used variables in the attached FreeMaster project

Variable Name in FreeMASTER	Description	Editable (E) or Measur-ement (M)
SpeedActRpm	Actual speed measured in RPM	M
SPEED_CMD	Speed Command [used only if the motor is in Speed Control mode]	E
TqCmd	Torque Command [used only if the motor is in Torque Control mode]	E
START	Starts the operation of the motor based on the specified control mode and the commanded values.	E
Synchup_Mode	Identify whether the operation will be B2B motors(1) or only one motor will operate(0).	E
TqCtrlSwitch	Identify the control mode: (1) for Torque Control mode and (0) for speed Control mode.	E
DC_Power	The power transmitted from the supply towards the motor over the DC link. (+) indicates the Motoring mode. (-) indicates the Generating mode.	M
Vdc	The DC link Voltage.	M
Idc	The current drawn from the supply towards the motor over the DC link. (+) indicates the Motoring mode. (-) indicates the Generating mode.	M
IdcFiltered_After_Sw	The filtered current drawn from the supply towards the motor over the DC link.[Same as Idc but with filter in the way of the signal]	M

	(+) indicates the Motoring mode. (-) indicates the Generating mode.	
PWM_A	The PWM of phase A	M
TrgTime	The Time during which the GD3000 predriver is being triggered.	E
STATE_READY	Indicates the triggering of GD3000 predriver so it is the ready state for starting the application when having value (1).	M
SpeedRef	Reference Speed that takes the value of speed command to compare it with the actual speed [has an effect only if the motor is in speed control mode].	M
TqRef	Reference Torque that takes the value of torque command to compare it-in terms of current- with the measured iq [has an effect only if the motor is in torque control mode].	M
PWM_Enable	Allowness of the PWM reaching to the concerned blocks. [(1):allow - (0):prevent]	E & M
theta_to_parkInv	Measured electrical angle of Rotor position whose sine and cosine are directed to Park and Inverse Park Transformations.	M
VF_scaling_factor	Scaling the voltage when operating in Open Loop.	E
State_num	Identifies the current operating state of the application.	M
OL_timer	Calculates the time that the application spends inside the open loop operation state.	M
End_OL	Indicates the end of open loop operation state and transition into closed loop operation. (1):End of OL operation. (0):Still in OL operation.	M
Skip_OL	Skipping the open loop operation state before the	E

	closed loop operation. (1):Skip OL and go directly to the CL operation. (0):Go through the OL state for an adjustable time and then transit to CL operation.	
Indx_pulse	Indicates the index pulse of the encoder.	M
Align_dn_in	Indicates that the other motor finishes its alignment state (its alignment is done) and that it is the time now to get out from the waiting state and continue the algorithm. [Received from pin G7]	M
Align_dn_out	Indicates that the motor finishes its alignment state (its alignment is done) and that it is the time now to tell the other motor to get out from the waiting state and continue the algorithm. [Transmitted through pin C10]. **<u>Note</u>:Two physical connections between pin A14 in each board to pin G10 in the other board must be made before starting the B2B operation so as to send and receive the values of Align_dn-out and Align_dn_in variables respectively.**	M
Offset1	Stores the offset of Motor1 [An offset from the initial position of the rotor occurs during the alignment state so this offset is stored in this variable to use it to calibrate the measured angle when the rotor rotates]	M
Offset 2	Stores the offset of Motor2 [An offset from the initial position of the rotor occurs during the alignment state so this offset is stored in this variable to use it to calibrate the measured angle when the rotor rotates]	M
Execute_ALign	Indicates the Alignment state execution. (1):Still inside the alignment state. (0):Out of (before or after) the alignment state.	M
OL_IDC	The maximum allowed DC current during the OL	E

Overcurrent	operation.	
IDcOvercurrentAlignment	The maximum allowed DC current during the alignment state.	E
IDcOvercurrent	The maximum allowed DC current during the normal operation.	E
Estimated_Torque	The Output Torque [calculated not measured]	M
UP_TQLimit_TqCtrl	The upper limit of torque of the motor when operating in the Torque Control mode.	E
Low_TQ Limit_TqCtrl	The lower limit of torque of the motor when operating in the Torque Control mode.	E
UP_TQLimit_SpdCtrl	The upper limit of torque of the motor when operating in the Speed Control mode.	E
Low_TQ Limit_SpdCtrl	The lower limit of torque of the motor when operating in the Speed Control mode.	E
PMflux	Rotor’s Permanent Magnet Flux	E [but constant for this motor type]
AlignmentIdRef	Reference Id during the alignment state [this reference Id indicates the reference flux during the alignment state]	E
Restart	Restarting the application. (1):Return back to the beginning of the operation.	E

	(0):Continue the operation.	
Stop	Stopping the motor gradually according to the specified decreasing step.	E
FAULT	Indicates the occurrence of any type of faults: Over Current, OverVoltage, UnderVoltage or even no storing of offset value. (1):There is a fault. (0):There is no fault.	M
ThetaPiToMinusPi_elec	Measured electrical angle of rotor position normalised to be from -pi to pi.	M
IQ_REF	It is the TqRef in terms of current.	M

6.2. Important Variables for Protection

These variables can be changed [Editable] but be careful while assigning the suitable values to them.

Variable Name	Value	Description
OL_IDcOvercurrent	5.5 A	Maximum Allowed Current Limit during Open Loop State.
IDcOvercurrent	5 A	Maximum Allowed Current Limit during Normal State
IDcOvercurrentAlignment	6 A	Maximum Allowed Current Limit during Alignment State.
VDC_OVERVOLTAGE	18 V	Maximum Allowed Voltage Limit
VDC_UNDERVOLTAGE	8 V	Minimum Allowed Voltage Limit
MaxSpdLimit	199 Rad/s = 2000 RPM	Maximum Allowed Speed Limit

AlignmentIdRef	0.3	IdRef given for attracting the rotor during Alignment State