

ID502 - Back-EMF Motor Control Lab

Description: Lab to familiarize users with Motor Control Algorithm using BEMF commutation and on how to set parameters for different motors.

Objectives

After the presentation the attendee will be able to:

1. Run the YMCRPR8C25 BEMF Code.
2. Use the LEM GUI to evaluate performance
3. Understand where BEMF Commutation can be applied
4. Be able to change the parameters for different BLDC motors.

Lab Materials:

Please verify you have the following materials at your lab station.

- (1) MCRP25 Motor Control Board
- (1) Emerson BLDC motor
- (1) E8A On-Chip Debugging Emulator
- (1) 24VDC Power Supply
- HEW + NC30WA V.5.43 + E8A Debugger Component V2.10 or later

Skill Level: Familiarity with Embedded development tools

Total Time to Complete Lab: 60 minutes

Lab Sections

1	Set up Motor Control BEMF-Mode Demo Time to complete task: 10 minutes
2	BEMF Speed Analysis Time to complete task: 10 minutes
3	Motor Alignment Time to complete task: 10 minutes
4	Open Loop Ramp Time to complete task: 10 minutes
5	BEMF Ramp Time to complete task: 10 minutes
6	Mask Time Adjustment Time to complete task: 10 minutes

1 Set up Motor Control BEMF-Mode Demo

Time to complete task: 10 minutes

Overview:

Setup and Run the Back-EMF Motor demo, and learn how the BEMF version is different from the Hall version (hopefully, you had a chance to try it in ID 501).

Procedural Steps:

1.2) Visually confirm that the jumpers for BEMF mode are correct according to Figure 1.

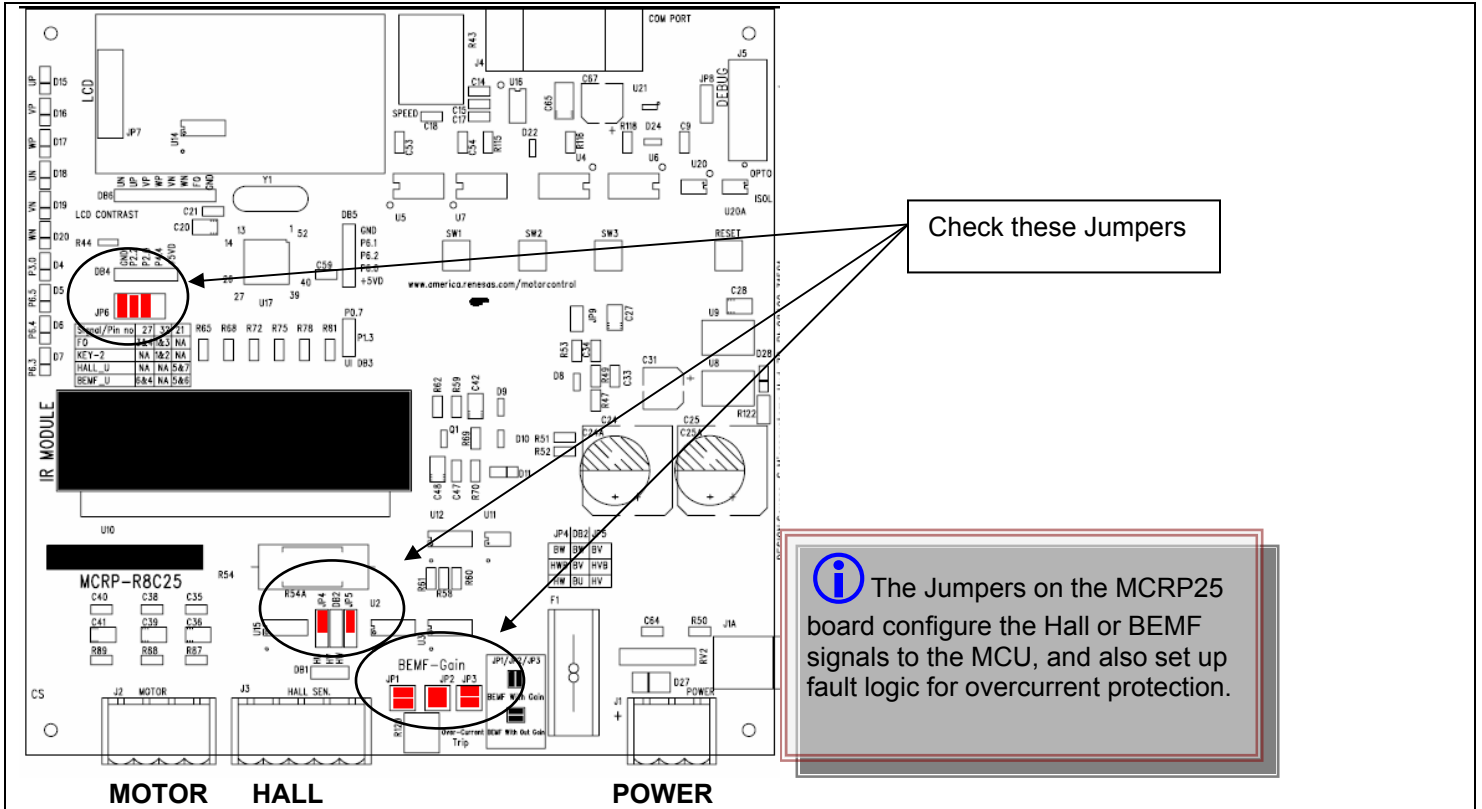
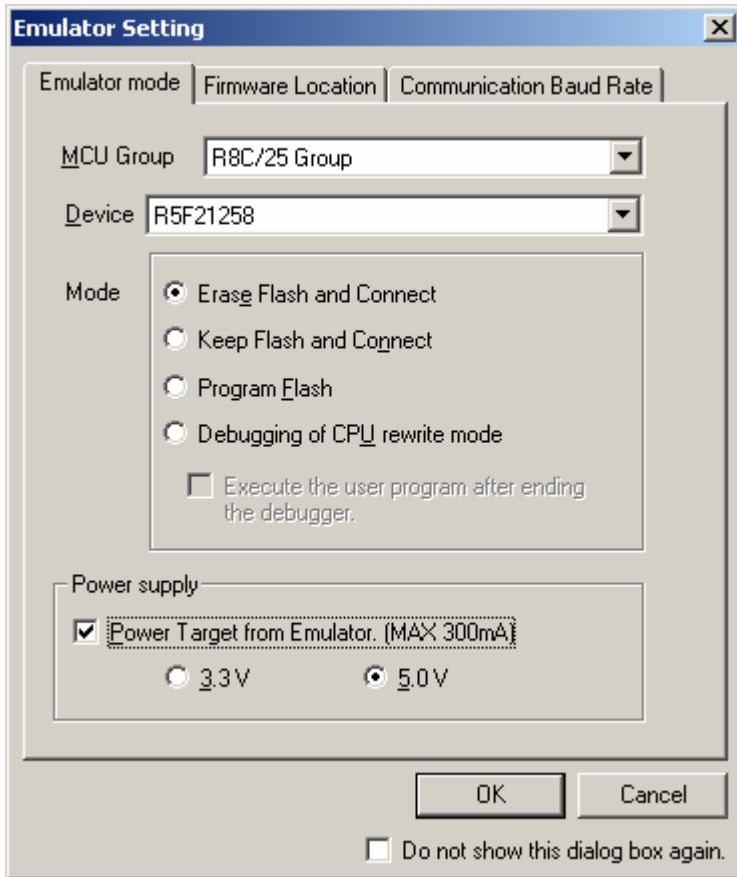




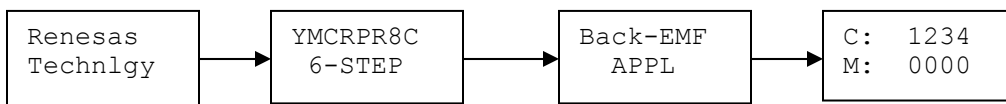
Figure 1. MCRP25 Board BEMF Jumper Configuration

1.3) Load the BEMF-Mode Demo project, found in the workspace directory-(C:\Workspace\YMCRPR8C25\BEMF_App.hws).

- 1.4) HEW will pop up an Emulator window, set the Mode to “Erase Flash and Connect” and select the “Power Target from Emulator” option, at 5.0 V. Hit OK. The E8a emulator will then connect to the board. Note: The E8a emulator port on the YMCRPR8C25 is optically isolated from the MCU and power electronics, so it needs to be powered by the E8a Emulator.



- 1.5) **Build** the project within HEW: [menu] Build >>  Build All. This should return a “No Errors” message, returning with one warning is OK, if the version of HEW is a demo version. This will also download the program to the board.
- 1.6) Start the program running with HEW: [menu] Debug >>  Reset Go. The LCD on the YMCRPR8C25 board should show a splash message and then show Commanded (C) and Measured (M) RPM, in this format:



The Commanded RPM value will change when rotating the speed control Potentiometer, R43.

- 1.7) Hit the Forward and Reverse buttons (SW3 and SW1), Rotate the Speed Control Potentiometer, R43, and observe the Commanded and Measured Speed in RPM displayed on the LCD. The speed range available here is 500 to 3000 RPM.
- 1.8) Set the Commanded Speed Pot, so the display reads close to 1500 RPM (C: 1500), then load the motor by hand and note the measured speed variation.


2
BEMF Speed Analysis

Time to complete task: 10 minutes

Overview:

Evaluate BEMF Commutation with Motor Load. Get a feel for how BEMF performs under load, especially at low speeds.

Procedural Steps:

- 2.1) While the BEMF demo project is open/running, stop the motor by using the Stop button (SW2) on the MCRP25 board.
- 2.2) Launch the GUI program by clicking on ICON in Quick Launch  . Follow On screen Prompts.
- 2.3) Change to OScope Panel.
- 2.4) Verify that plots selected are Speed and Motor Current
- 2.5) Click RUN button to begin the scope scanning.
- 2.6) Click in the RPM box and type in 1500, click Set RPM Button
- 2.7) Note speed plot ramps to 1500 RPM
- 2.8) Click the RUN button twice to stop and restart PLOT, Note scale on RPM contracts (re-scales the window).
- 2.9) Note the apparent speed variation.
- 2.10) Slide speed to minimum, around 500 RPM.
- 2.11) Apply additional Load to motor. Is it easy to stall?

Questions:

- 2.1) Is the apparent speed variation more or less than the HALL sensor code? Why?
- 2.2) Is the BEMF at low speeds easier to stall than HALL Sensor based code? Why?

NOTE: Answers are to be written on the last page of this lab.

3
Motor Alignment

Time to complete task: 10 minutes

Overview:

Changing Alignment Parameters for Large Inertial Loads. Large loads, such as fans and blowers, need longer time to allow the rotor position to settle. We will triple this time then observe it on the motor and on the GUI.

Procedural Steps:

- 3.1) Open the file named `common_def.h` for editing. There are two parameters that determine the length of this ramp, the `MAX_ALIGN_DUTY_CYCLE`, which is set at 10%, and `ALIGN_DELTA_DUTY`, which we will change to make the ramp up slower and so take longer.
- 3.2) Add the 1/2" drill chuck to the shaft of the motor, this will triple the inertia and make it more difficult to get started.
- 3.3) Try to start the motor with the standard parameters several times to see how reliable it is.
- 3.4) Commenting out the original parameters for later reference, change the `ALIGN_DELTA_DUTY` value by using the handy macro `DUTY_TO_COUNT()`. Set the value to `DUTY_TO_COUNT(0.3)`. This will reduce the amount added to the PWM on each cycle and so increase the alignment period by a factor of three.
- 3.5) Save the file and re-compile. This will also download the new hex file to the chip.
- 3.6) Click Reset-Go.
- 3.7) Restart LEM, and start the motor moving by sliding the speed bar.
- 3.8) Note in the LEM OScope the time it takes for alignment. The slower acceleration is needed for starting a motor with a large inertia load, like a fan blade or blower.

Questions:

- 3.1) What is relationship of alignment time to motor/load inertia?
- 3.2) Why does the alignment period use a ramping current instead of a simple square pulse?
- 3.3) What could happen if the alignment period was too short? Too long?

NOTE: Answers are to be written on the last page of this lab.

Bonus steps, if you have time:

- 3.9) Try setting the `ALIGN_DELTA_DUTY` to its original value, but increase the `MAX_ALIGN_DUTY_CYCLE` to see if that will make the extra inertia start reliably. Does it? Why or why not? What risk do we take in increasing the open loop duty cycle?
- 3.10) Calculate the `MAX_ALIGN_DUTY_CYCLE` (in percent of bus voltage) that you would recommend for a motor with a continuous current rating of 6A, bus voltage of 36V, and a line-to-line resistance of 0.6 ohms.

4
Open Loop Ramp

Time to complete task: 10 minutes

Overview:

Setting up STARTING_RAMP Parameter, also important for dealing with large inertial loads.

Procedural Steps:

- 4.1) Exit LEM, if it is still running and halt the code operation.
- 4.2) The motor should still have the drill chuck on the shaft for added inertia.
- 4.3) Open the file named common_def.h for editing, if it isn't already opened.
- 4.4) Commenting out the original parameter, change the STARTING_RAMP value to 40. This will reduce the rate at which the speed increases while in the open-loop mode.
- 4.5) Save the file and compile. This will also download the new hex file to the chip.
- 4.6) Click Reset-Go.
- 4.7) Restart LEM, and start the motor moving by sliding the speed bar.
- 4.8) Note the time it takes for the open-loop acceleration. This is another critical parameter for driving a motor with a large inertia.

Questions:

- 4.1) What is the relationship between the STARTING_RAMP and acceleration time?
- 4.2) What would happen if the STARTING_RAMP parameter was set to zero?
- 4.3) How can you tell when the parameter is set correctly?

NOTE: Answers are to be written on the last page of this lab.

Bonus steps, if you have time:

- 4.9) Try removing the inertia (drill chuck) and see how the motor starts. Any problems with the lower inertia?
- 4.10) Try setting the STARTING_RAMP parameter to zero, but be ready to halt the code.
- 4.11) If a new load represented 10 times the inertia of a known load, how would the new STARTING_RAMP parameter be set?
- 4.12) What other factors, besides inertia, can affect the STARTING_RAMP parameter setting? (Hint, imagine a blower is being started and is ramped up to 20% of its full speed.)

5

BEMF Ramp

Time to complete task: 10 minutes

Overview:

Setting up ACCEL_TIME Parameter, used during normal speed control operation. This parameter also affects deceleration.

Procedural Steps:

- 5.1) Exit LEM and stop the code, if they are still running.
- 5.2) Open the file named common_def.h, if it isn't already opened.
- 5.3) Change the parameter near the bottom called ACCEL_TIME from 2 to 10. This will lengthen the time the speed ramp takes to go from the present speed to the new set speed.
- 5.4) Save the file and compile. This will also download the new hex file to the chip.
- 5.5) Click Reset-Go.
- 5.6) Restart LEM, and start the motor moving by sliding the speed bar.
- 5.7) Note the time it takes to go from minimum speed to maximum speed in the GUI scope.

Questions:

- 5.1) What is the relationship between the ACCEL_TIME and top speed?
- 5.2) What would happen if the ACCEL_TIME parameter was set to zero?
- 5.3) ACCEL_TIME affects deceleration, too. What can happen to the bus voltage if a motor decelerates quickly?

NOTE: Answers are to be written on the last page of this lab.

Bonus steps, if you have time:

- 5.8) Try setting the ACCEL_TIME to zero, but be ready to stop the code if necessary.
- 5.9) If the ACCEL_TIME is reduced, what happens to the motor current when the speed is changed?

6

Mask Time Adjustment

Time to complete task: 10 minutes

Overview:

Setting up DI/DT Filter Parameter, used to filter affects caused by motor inductance onto the Back-EMF voltage. If too small, the low speed performance is affected. If too large, the high speed is affected.

Procedural Steps:

- 6.1) Exit LEM and stop the code, if they are still running.
- 6.2) Open the file named common_def.h, if it isn't already opened.
- 6.3) Change the parameter near the bottom called MASK_CYCLES from 6 to 20. This will lengthen the time the BEMF Comparators are ignored after a commutation step change.
- 6.4) Save the file and compile. This will also download the new hex file to the chip.
- 6.5) Click Reset-Go.
- 6.6) Restart LEM, and start the motor moving by sliding the speed bar a little bit at a time (unclick the mouse at each increment).
- 6.7) Continue sliding the speed bar up until the motor speed reaches a maximum.

Questions:

- 6.1) Is there a relationship between the MASK_CYCLES and top speed?
- 6.2) What would happen if the MASK_CYCLES parameter was set to zero?
- 6.3) What particular motor characteristic establishes the need for this filter?

NOTE: Answers are to be written on the last page of this lab.

Bonus steps, if you have time:

- 6.8) Try reducing the MASK_CYCLES value from the original, and note what happens to the lower speed operation. It might be necessary to load the motor to note the change.
- 6.9) It is not as easy to add inductance to the motor as it was to add inertia, so this lab is just for practice. What other motor parameter might affect the MASK_CYCLES value? (hint: what constitutes the electrical time constant?)

Answer Page

2.1) Is the apparent speed variation more or less than the HALL sensor code? Why?

2.2) Why was it easier to stall the motor in BEMF mode?

3.1) What is relationship of alignment time to motor/load inertia?

3.2) Why does the alignment period have a ramping current instead of a simple square pulse?

3.3) What could happen if the alignment period is too short? Too long?

4.1) What is the relationship between the STARTING_RAMP and acceleration time?

4.2) What would happen if the STARTING_RAMP parameter was set to zero?

4.3) How can you tell when the parameter is set correctly?

5.1) What is the relationship between the ACCEL_TIME and top speed?

5.2) What would happen if the ACCEL_TIME parameter was set to zero?

5.3) ACCEL_TIME affects deceleration, too. What can happen to the bus voltage if a motor decelerates quickly?

6.1) Is there a relationship between the MASK_CYCLES and top speed?

6.2) What would happen if the MASK_CYCLES parameter was set to zero?

6.3) What particular motor characteristic establishes the need for the Di/Dt filter?
