



---

## Getting Started Guide – CM408 FlexMC Platform

*Rev1.0*

---

Last Modified: Oct-14

---

# 1 Contents

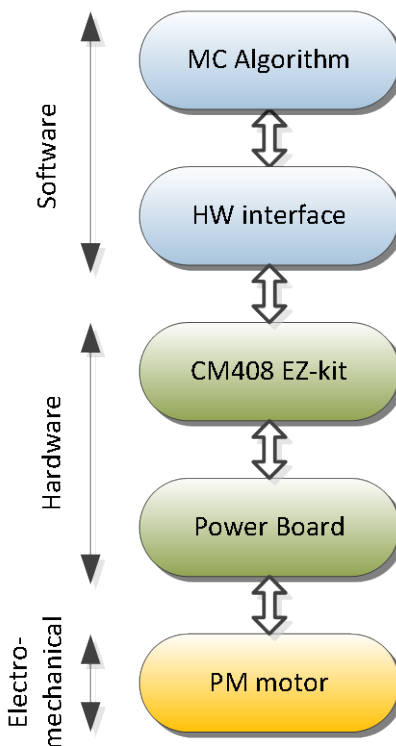
---

1	Contents .....	2
2	Overview .....	3
2.1	System requirements .....	4
3	HW setup.....	4
3.1	Low Voltage Board.....	5
3.1.1	Encoder connector.....	5
3.1.2	Motor and Power Connector .....	7
3.2	High Voltage Board .....	8
3.2.1	Encoder and motor connector .....	8
4	SW setup .....	10
4.1	SW Enablement Package.....	10
4.2	Matlab Environment Setup.....	11
4.3	EWB Environment setup.....	11
5	Build Process.....	13
5.1	Building Simulink Model.....	15
5.2	Building IAR project .....	15
6	Download and programming .....	16
7	Running the Motor .....	17
8	Data Visualization .....	20
9	Support.....	25

## 2 Overview

This document will give a high level introduction to the CM408F motor control development platform and it will provide information on how to install and configure tools, embedded SW and HW. The purpose of the document is to provide a step-by-step approach that will get a motor up running quickly but without delving into the details.

The CM408F development platform consists of software-, hardware and electro-mechanical components; see Figure 1.



*Figure 1 Components of Motor Control platform.*

### Software components

- The Motor Control algorithm performs field oriented control of a 3-phase PM motor. The algorithm relies on current feedback, incremental rotor position and absolute rotor position. A model based approach and auto-code generation is utilized to implement the control algorithm.
- The Hardware interface is an abstraction layer that ties the generic algorithm code to the embedded platform. Relying on device drivers, the HW interface handles system setup, scheduling and access to peripherals.

### Hardware components

- The CM408F EZ-kit is a general purpose processor evaluation board. It has all the I/Os and glue circuits needed to run the CM408F ASSP.

- The Power Board is a 3 phase switching inverter that interfaces to the EZ-kit though a digital- and an analog connector. A high voltage- as well as a low voltage variant of the Power Board exists. This document will cover both platforms.

#### Electro-mechanical components

- The PM motor is a 3-phase permanent magnet motor that is matched to the power board. The low voltage platform uses a low voltage motor and the high voltage platform uses a high voltage motor.

The same SW is used to run the high- and low voltage platform.

## 2.1 System requirements

Before you start working on the motor control platform, please make sure you have the hardware and software listed below.

### Required Hardware

- ADSP-CM408F EZ-KIT rev 0.2.
- 24V power board rev 2 or 110V/220V power board rev 2.
- Anaheim Automation BLY171D-24V-6000 motor if using 24V board.
- Bearing Engineering M-2311S-LN-02D motor if using 110V/220V board.
- Segger J-link Ultra+ debugger or J-Link Lite debugger
- Isolated 24V power supply powering 24V board.
- 9-pin D-SUB serial cable (or USB to serial if PC has no serial port)

### Required Software

- Matlab version 2014a. Both 32bit and 64 bit version supported.
- IAR Embedded Workbench® revision 6.60 (or later)
- Segger J-Link driver version 4.76a
- CM40x SW Enablement Package version 1.1.0
- Motor control SW package version 0.2.0

## 3 HW setup

This section will describe how to setup the HW. This only has to be performed once, when bringing up a new platform. Low- and High Voltage boards are configured differently and will be described in separate sections.

### 3.1 Low Voltage Board

Connect the EZ-kit to the Power Board as shown in Figure 2. Make sure both Samtec connectors mate completely. Also, note the location of Encoder, Power and Motor connectors.

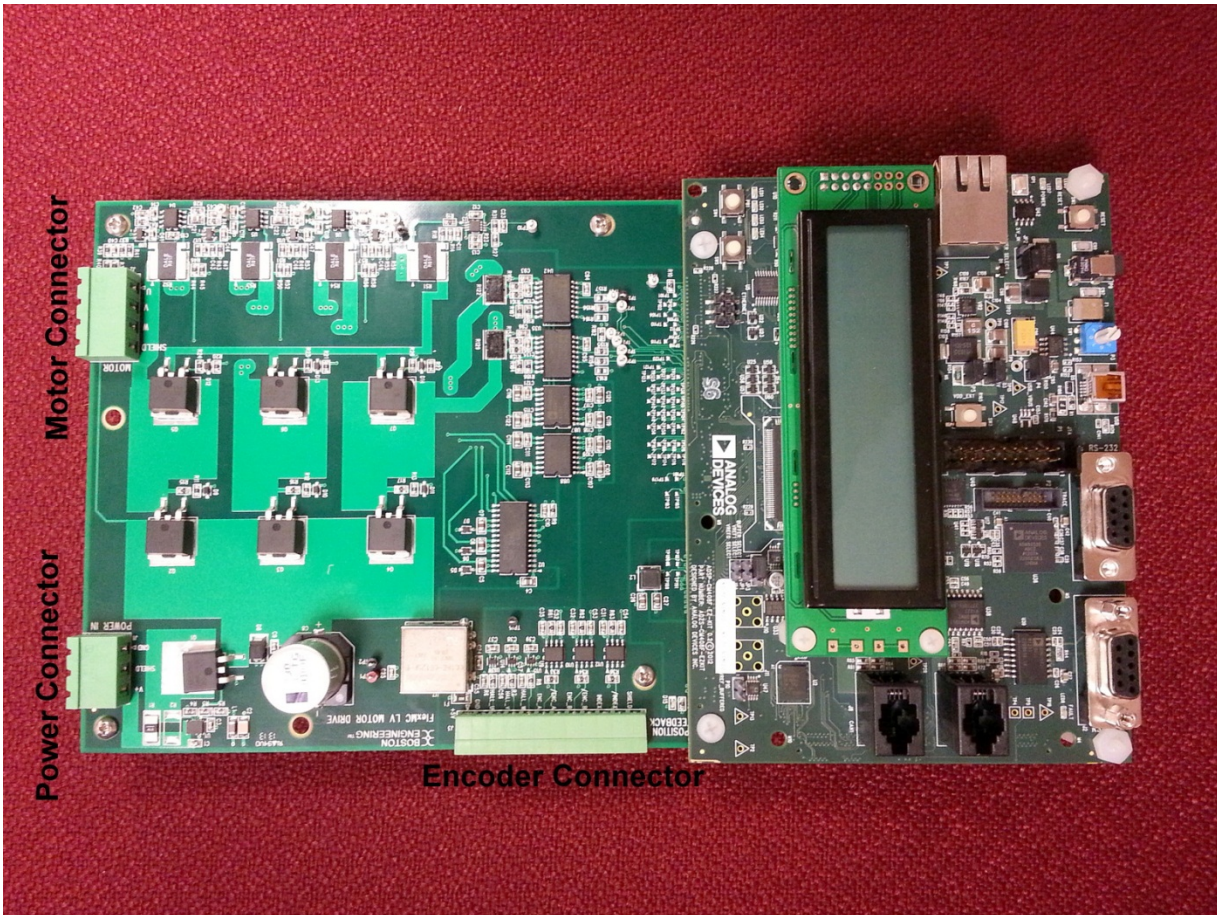


Figure 2 Low Voltage Power Board connected to CM408F EZ-kit.

#### 3.1.1 Encoder connector

Wire the Encoder connector as shown in Figure 3 and Table 1.

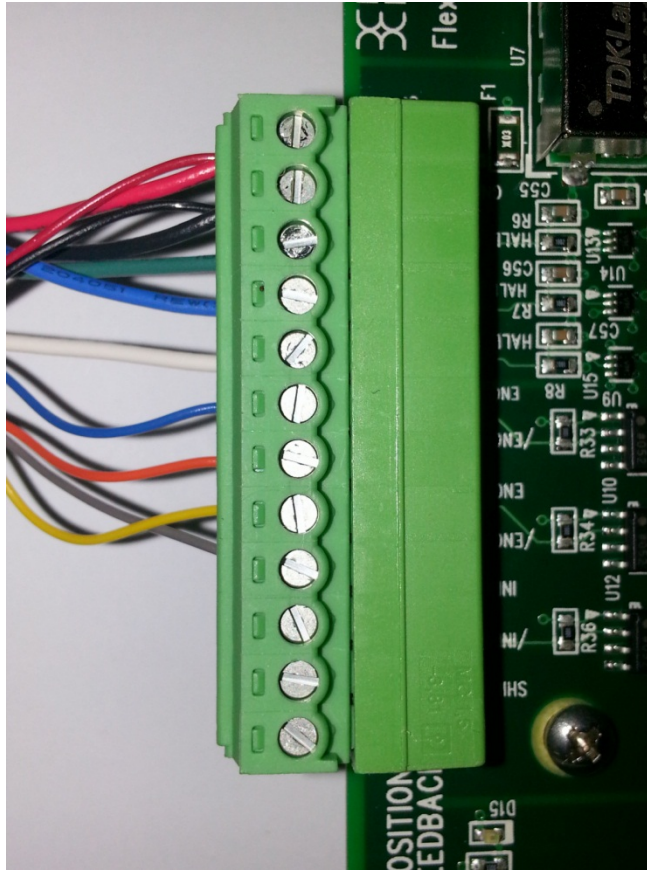


Figure 3 Wiring of Encoder Connector.

Pin	Color	Signal
1	Red (thick and thin wire)	+5V
2	Black (thick and thin wire)	GND
3	Green	HALL_U
4	Blue (thick)	HALL_V
5	White	HALL_W
6	Blue (thin)	ENC_A+
7	Orange	ENC_A-
8	Yellow	ENC_B+

9	Grey	ENC_B-
10	NC	INDEX+
11	NC	INDEX-
12	NC	Shield

Table 1 Encoder Connector

### 3.1.2 Motor and Power Connector

Wire the Motor and Power Connector as shown in Figure 4, Table 2 and Table 3.

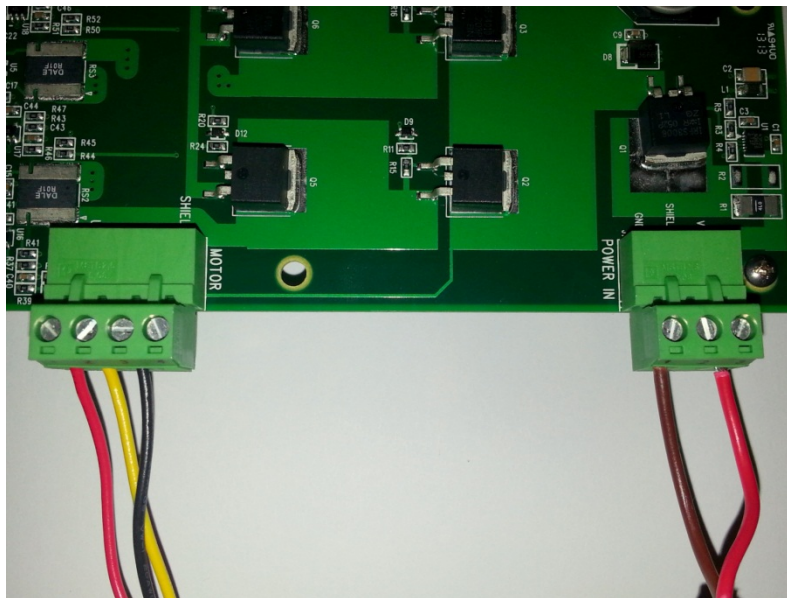


Figure 4 Wiring of Motor- and Power Connector.

Pin	Color	Signal
1	Red	Motor phase U
2	Yellow	Motor phase V
3	Black	Motor phase W
4	NC	Shield

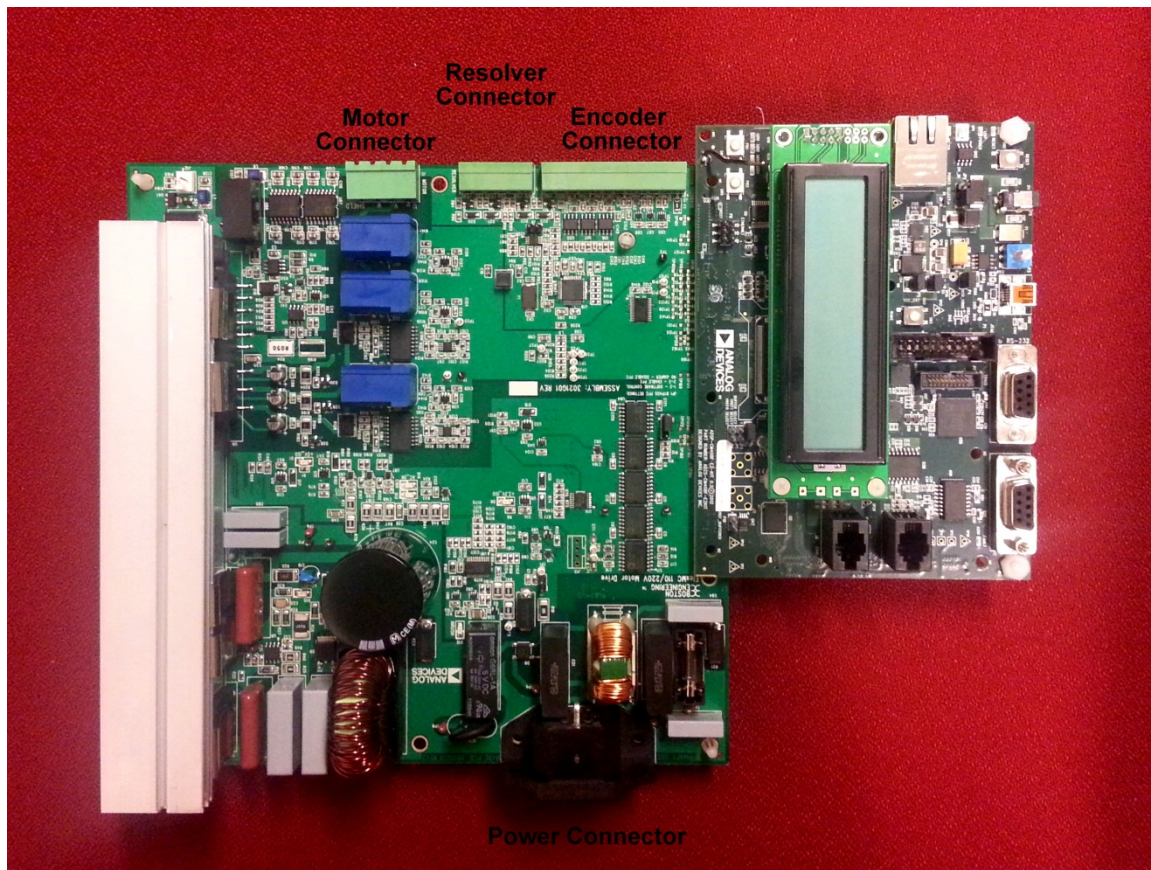
Table 2 Motor Connector.

<i>Pin</i>	<i>Color</i>	<i>Signal</i>
1	Brown	GND
2	NC	Shield
3	Red	+24V

*Table 3 Power Connector*

### 3.2 High Voltage Board

Connect the EZ-kit to the Power Board as shown in *Figure 5*. Make sure both Samtec connectors mate completely. Also, note the location of Encoder, Power and Motor connectors.



*Figure 5 High Voltage Power Board connected to CM408F EZ-kit.*

#### 3.2.1 Encoder and motor connector

Wire the Motor- and Encoder Connector as shown in *Figure 6*, *Table 4* and *Table 5*.

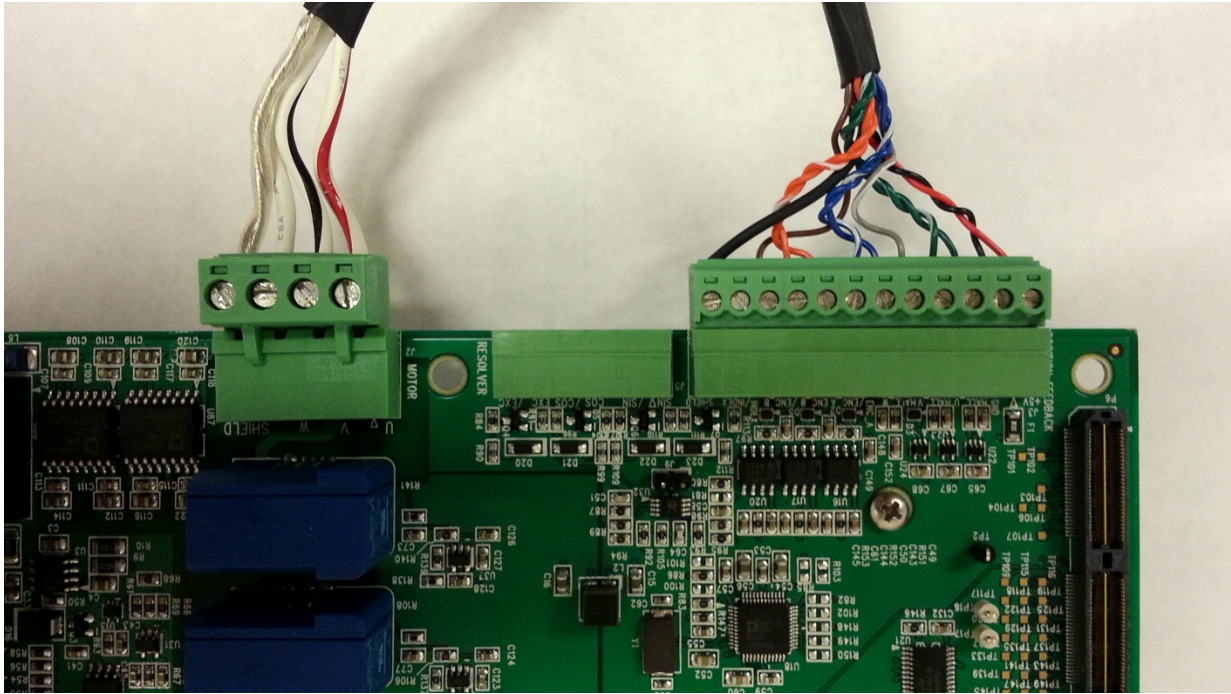


Figure 6 Wiring of Motor- and Encoder Connector.

Pin	Color	Signal
1	Red	+5V
2	Black	GND
3	Green	HALL_U
4	Green w. white stripe	HALL_V
5	Grey	HALL_W
6	Blue	ENC_A+
7	Blue w. white stripe	ENC_A-
8	Orange	ENC_B+
9	Orange w. white stripe	ENC_B-
10	Brown	INDEX+
11	NC	INDEX-
12	Black	Shield

Table 4 Encoder Connector.

<i>Pin</i>	<i>Color</i>	<i>Signal</i>
1	White w. red stripe	Motor phase U
2	White w. black stripe	Motor phase V
3	White	Motor phase W
4	Clear	Shield

Table 5 Motor Connector.

## 4 SW setup

First, get the Motor Control SW package copy it to a local disk drive. The path of the MC project will be referenced as \$MC\_DIR\$ in the following. For example, MC\_DIR could be “C:\CM408Demo\”

After copying, the folder structure must look like the one shown in Figure 7.

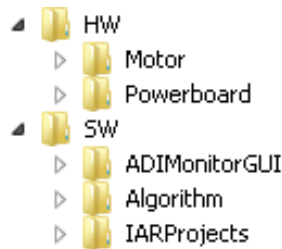


Figure 7 Folder structure.

### 4.1 SW Enablement Package

Before proceeding, download and install CM40x SW Enablement package. Version number is specified under SW Requirements at the top of this document.

To enable all debug functionality it is highly recommended to rebuild the device driver libraries on your own computer. To do so, follow the steps below:

1. Go to *C:\Analog Devices\ADSP-CM40x\CM403F\_CM408F\_EZ-KIT\lib\iar\debug\* and delete old library files (all files in this folder)
2. Open *C:\Analog Devices\ADSP-CM40x\CM403F\_CM408F\_EZ-KIT\lib\iar\lib40z.eww* in IAR EWB.
3. In the workspace window, right-click on libdrv40z and select rebuild all.
4. Right-click on libosal40z\_noos and select rebuild all.
5. Right-click on libssl40z and select rebuild all.

Note: Do not build libosal40z\_ucos3

## 4.2 Matlab Environment Setup

To prepare the Matlab environment, follow the following steps.

1. Open Matlab.
2. Install a compiler for Matlab to use by typing `mex -setup` in the Matlab Command Window. Matlab can search for available compilers – select the one you want to use.
3. Add `$MC_DIR$\SW\Algorithm\Library\` **including sub-folders** to Matlab path.
4. Add `$MC_DIR$\SW\Algorithm\Pictures\` to Matlab path.
5. Add `$MC_DIR$\SW\ADIMonitorGUI\` to path.
6. If Matlab path contains paths of previous installed versions of the demo SW, make sure to remove them.
7. Open `$MC_DIR$\SW\Algorithm\CM40xPMSM\InitCM40xPMSM.m` and set `MOTOR_CFG` to the motor to be used in the demo. E.g. `MOTOR_CFG = MOTOR.ANAHEIM_24V;`
8. Open `$MC_DIR$\SW\Algorithm\CM40x\CfgCM40xPMSM.m` and set `USE_SINC = 0` if SAR ADC is used and `USE_SINC = 1` if Sigma-Delta ADC is used.

Step 1 to 6 only needs to be done once. Step 7 must be done when switching between LV and HV platforms. Step 8 must be done when switching between SAR ADC and Sigma-Delta ADC.

## 4.3 EWB Environment setup

To prepare the IAR Embedded Workbench environment, follow the steps below.

1. Open `$MC_DIR$\SW\IARProjects\PMSM_FOC\iar\pmsm_foc.eww` in IAR EWB (Open Workspace)
2. Go to “*Project*→*Options*→*General Options*→*Target*” and ensure that a floating point unit (FPU) is selected. In the “*Library Configuration*” tab, ensure that “Use CMSIS” is selected.
3. Go to “*Project*→*Options*→*C/C++ Compiler*→*Preprocessor*→*Define Symbols*”, see Figure 8, and check if the correct precompiler options are selected for the demo platform used. Possible configurations for the preprocessor defines are shown in Table 6.
4. Ensure that the rebuilt library files are listed as shown in Figure 9.

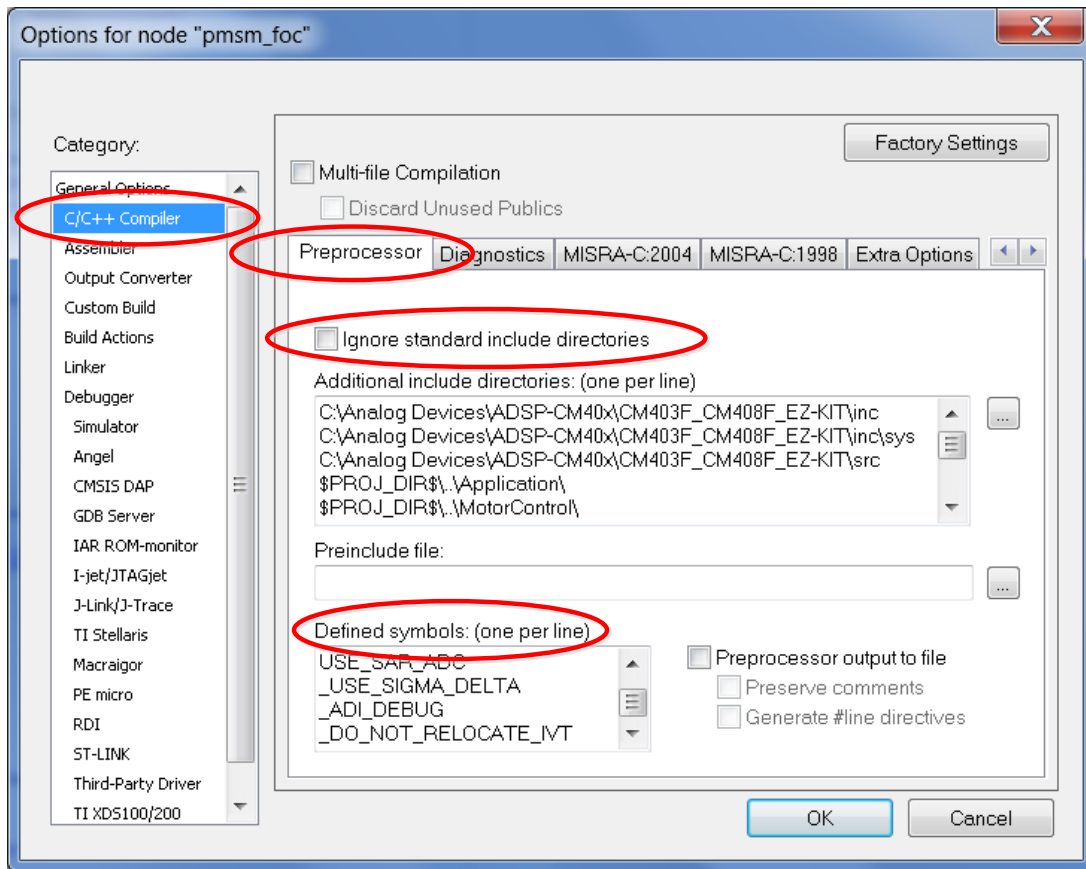


Figure 8 EWB Compiler settings.

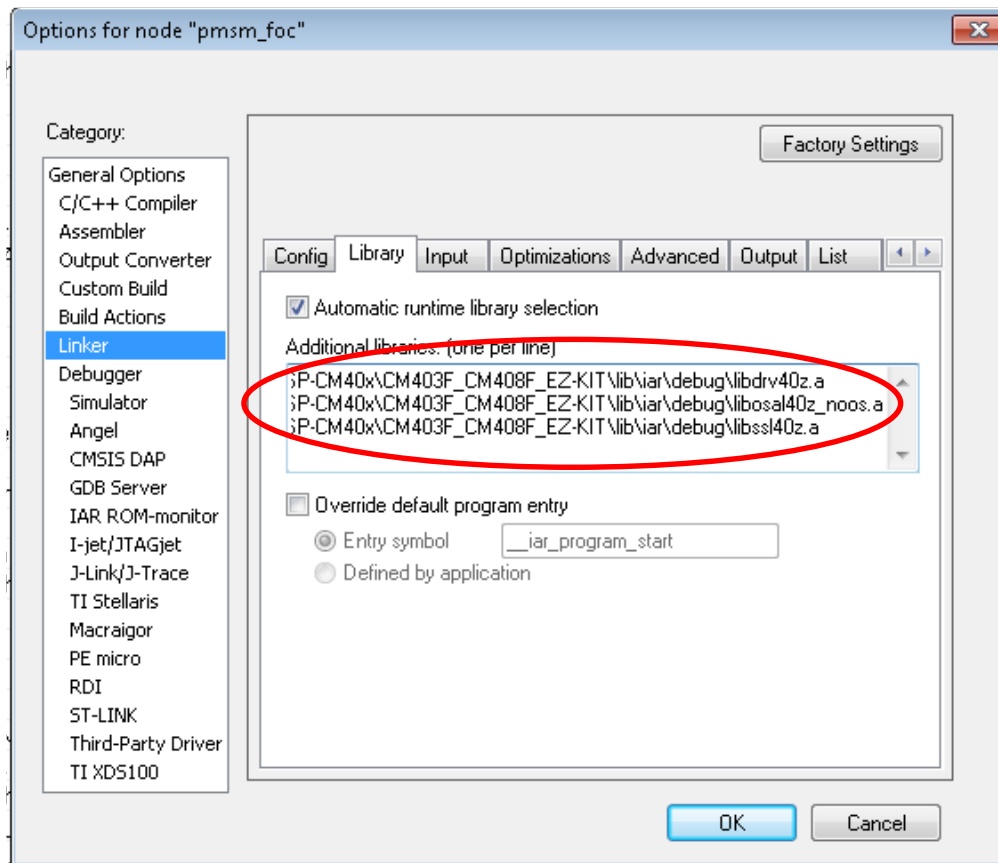


Figure 9 EWB Linker Libraries

## 5 Build Process

The build process consists of two steps. The first is to build the motor control application into c-code which is done in the Matlab/Simulink. The second step is to build and link the motor control code together with the HW abstraction layer, device drivers and system setup. This second step is performed in IAR Embedded Workbench. The end-result of the build process is an executable that can be deployed to the target. A high level view of the build is shown in Figure 10.

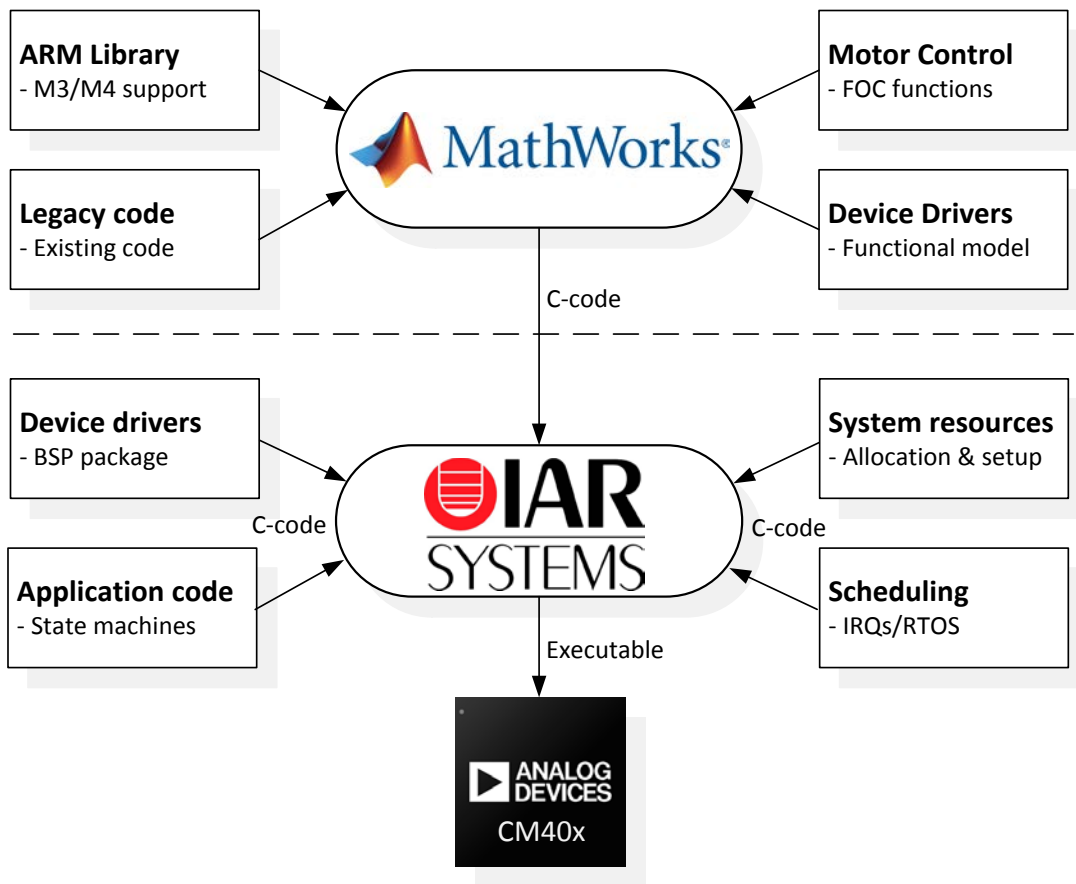


Figure 10 Overview of build process.

With the release comes prebuild motor control code for the Low Voltage platform. It is recommended always to confirm the right configuration of the Simulink model and to rebuild to make sure the design is up to date. If Low Voltage board/motor is not used, the motor control must be rebuilt.

A number of different system configurations are supported by the Motor Control SW package. The source code is always the same but configuration variables and pre-compiler defines vary. All supported configurations are listed in Table 6.

Configuration	Simulink setting	EWB setting	GUI setting
Low voltage board	In InitCM40xPMSM.m: MOTOR_CFG = MOTOR. ANAHEIM_24V	-	-
High voltage board	In InitCM40xPMSM.m: MOTOR_CFG = MOTOR.TEKNIC_110V	In pfc.c #defines DC_BUS_VOLTAGE Set this to >ac line peak.	-
SAR ADC	In CfgCM40xPMSM.m USE_SINC = uint8(0);	Preprocessor define: USE_SAR_ADC	-
Sigma-delta ADC	In CfgCM40xPMSM.m USE_SINC = uint8(1);	Preprocessor define: USE_SIGMA_DELTA	-
Open-loop V/f	-	-	VF_CTRL=1 VF_GAIN as needed

PFC <sup>1</sup>	-	PFC	-
------------------	---	-----	---

Table 6 Supported configurations.

<sup>1</sup> It is OK to have PFC defined on the Low Voltage board. Commands over i2c are sent but have no effect.

## 5.1 Building Simulink Model

The Simulink model is built using the following steps:

1. Open `$MC_DIR$\SW\Algorithm\CM40xPMSM\ CM40xPMSM.slx` and go to the top level view as shown in Figure 11.
2. Right-click on the `PMSMctrl` sub-system (black box with ADI logo) and select `C/C++ code` → `Build This Subsystem`.
3. When a dialog box regarding tunable parameters pop up, click Build.
4. When Code Generation Report pops up the build is done. Click OK. Build status can be obtained from the Matlab prompt.

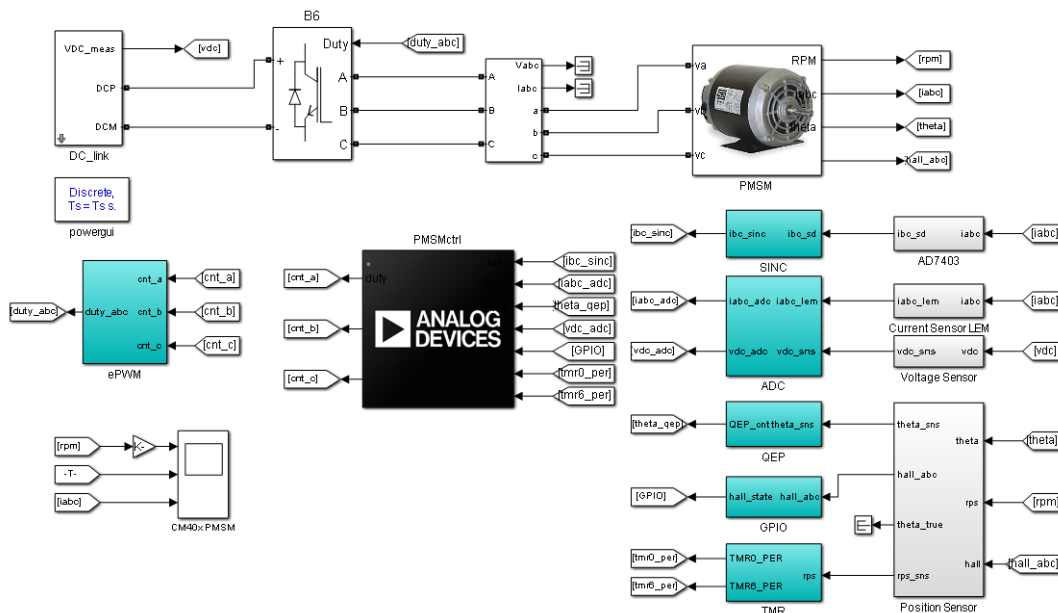


Figure 11 Simulink design.

## 5.2 Building IAR project

The IAR project now needs to be rebuilt on your machine.

1. Open `$MC_DIR$\SW\IARProjects\PMSM_FOC\iar\pmsm_foc.eww` in IAR EWB (Open Workspace)
2. Click `Project` → `Rebuild All`
3. Build/link status can be obtained from the message window.

## 6 Download and programming

Two different programming/boot load methods are supported:

1. Program embedded flash. At power up CM40x load code from flash into SRAM and executes code from SRAM. In this mode it is only necessary to program memory when the program is changed.
2. Load program directly to SRAM and execute from SRAM. In this mode there is no boot load and memory is cleared at power off.

There is a number of settings and that need to be configured to select the preferred download and programming method. They are all summarized in Table 7. It is recommended to program flash and let the device boot from flash.

To program the device, follow the steps below

1. Setup preferred programming and boot load method according to Table 7.
2. Connect Segger J-Link debugger between a USB port on the PC and the Debug connector on the EZ-kit
3. Turn on the 5V power supply to the EZ-kit.
4. Go to IAR EWB and download program to target by clicking *Project*→*Download and Debug*
5. In IAR EWB, start the program by clicking *Debug*→*Go*

Load and run from SRAM option requires IAR EWB+debugger for all downloads.

Once Flash has been programmed, Flash boot runs stand-alone and does not require IAR EWB+debugger

	Load and run from SRAM	Load from FLASH, run from SRAM
Boot mode switch P3 on EZ-kit	0 (do not boot)	1 (boot from flash)
Device <i>Options</i> → <i>General</i> <i>Options</i> → <i>Target</i>	<i>AnalogDevices ADSP-CM40z-384_2048</i>	<i>AnalogDevices ADSP-CM40z-384_2048</i>
Preprocessor <i>Options</i> → <i>C/C++ Compiler</i> → <i>Preprocessor</i>	Do NOT move IVT <i>DO_NOT_RELOCATE_IVT</i>	Allow IVT to be moved <i>_DO_NOT_RELOCATE_IVT</i>
Linker file <i>Options</i> → <i>Linker</i> → <i>Config</i>	Check <i>Override default</i> and select <i>CM40z – NoFlash.icf</i>	Uncheck <i>Override default</i>
Setup Macro	Check “Use macro file(s)” and point	Uncheck “Use macro file(s)”


Options → Debugger → Setup	to:  \$TOOLKIT_DIR\$\CONFIG\debugger\AnalogDevices\NoFlash_CM40z.mac  	
Flash loader Options → Debugger → Download	Uncheck Use flash loader(s)	Check Use flash loader(s)
J-link reset Options → Debugger → J-Link/J-Trace → Setup	Select <i>Connect during reset</i>	Select <i>Connect during reset</i>
J-link connection Options → Debugger → J-Link/J-Trace → Connection	JTAG or SWD	SWD

Table 7 Configuration of download and bootload

## 7 Running the Motor

This section covers the final steps in getting a motor up running.

1. If not done earlier, connect EZ-kit to PC via serial cable
2. Apply power to power board.
3. In Matlab Command window, type *ADIMonitor*
4. In the parameter pane of ADIMonitor, open COM port by clicking Connect, see Figure 12. The target is running at 57600baud (default value). If you do not know the number of the COM port go to Device Manager on the PC.
5. In the parameter pane of ADIMonitor click *Run* → *Command GUI*, see Figure 13.
6. In the command pane of ADIMonitor Press *Configure*, see Figure 14.
7. To start the motor press Start. Note that if using the high voltage board, an audible click should be heard on pressing Start (due to the inrush relay being turned on). Set the speed reference by sliding the bar or enter a numeric number. See Figure 15

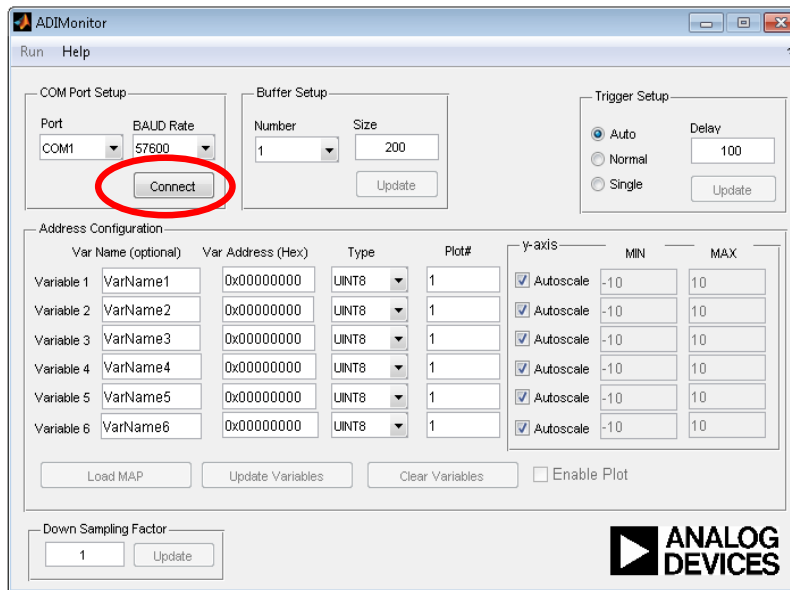


Figure 12 ADIMonitor, parameter pane.

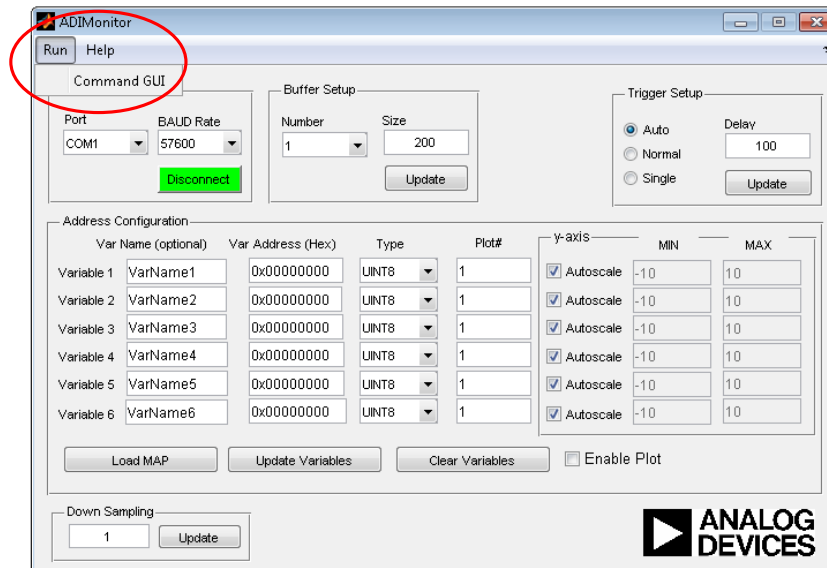


Figure 13 ADIMonitor, opening Command GUI.

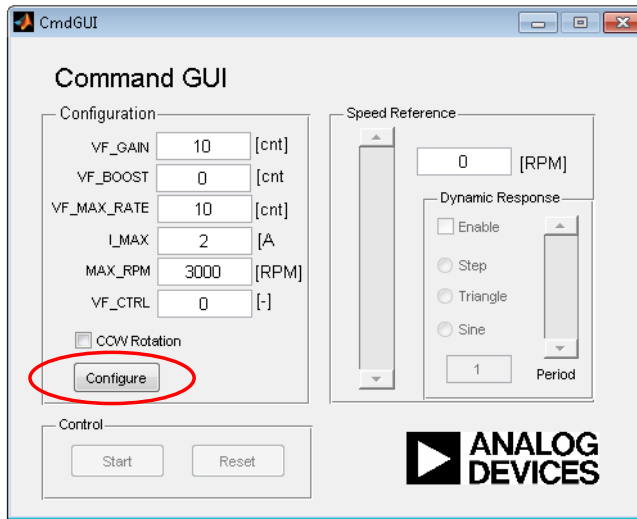


Figure 14 ADIMonitor, command pane.

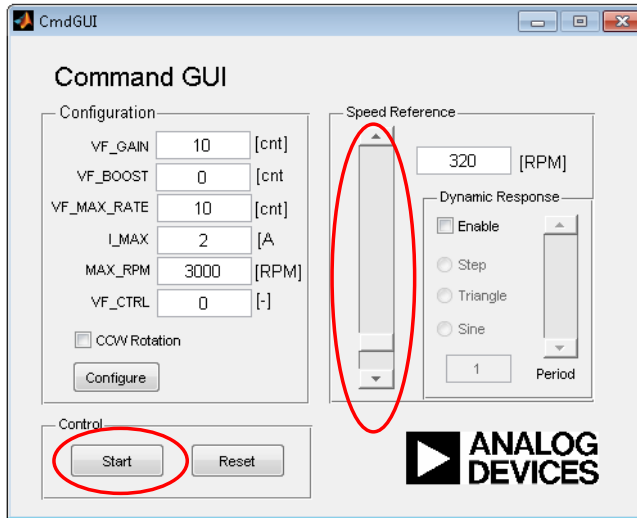


Figure 15 ADIMonitor, start and speed reference.

The Configuration parameters found in the Command pane are listed in Table 8.

Parameter	V/f control	FOC
VF_GAIN	Volt per Hz	Not used
VF_BOOST	Boost voltage	Not used
VF_MAX_RATE	Frequency change per sample	Not used
I_MAX	Not used	
MAX_RPM	Max rpm	
VF_CTRL	Set to 1 for V/f	Set to 0 for FOC
CCW rotation	Check for CCW rotation. Uncheck for CW rotation. When looking into shaft end of	

	motor.
Speed reference	Speed reference
Reset	Stop motor and reset speed reference

Table 8 Commands and tunable parameters in Command Pane.

For dynamic response testing, the motor speed reference can be changed to a square wave, triangle wave, or sine wave, by enabling the Dynamic Response checkbox and selecting the appropriate settings, including the period, as shown in Figure 16. These selections generate a motor speed reference with an envelope from zero to a peak level of the value selected by the main speed reference slider. It should be noted that, since the timing of the periodic waveforms is generated from the PC, it is not intended to be accurate with respect to the period.

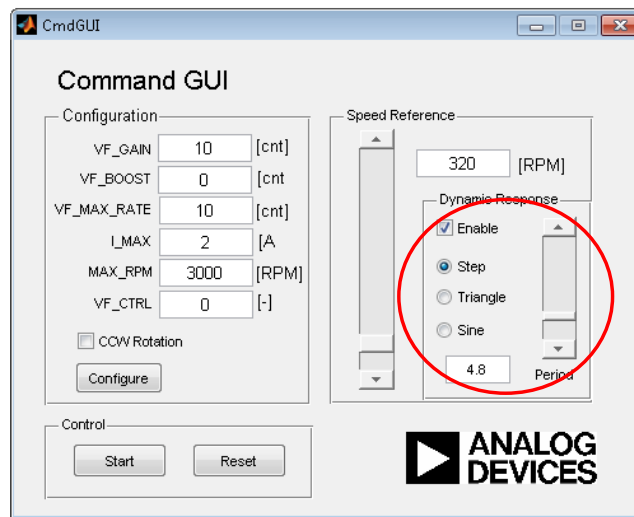


Figure 16: ADIMonitor, dynamic response selection

## 8 Data Visualization

The ADIMonitor tool can also be used to visualize ‘time slice’ buffers of data stored in the ADSP-CM40x memory. Specific data variables are accessed by pressing Load Map in the ADIMonitor GUI. This opens an additional window containing selected contents of the linker map file generated by the IAR EWB project, as shown in Figure 17. Also shown are the steps to select an alternative map file, or to select the correct map file, if the default map file (which is preloaded if it exists) does not exist. Listed in this table are all data variables listed in the map file, of size 1-4 bytes, with text filters applied to the map file search as indicated. These are intended for use either to select specific variables, or to limit the variable list to those associated with particular object files (e.g.” measure.o”) Arrays are not included. In order to list all of the 1-4 byte variables in the map file, simply delete all of the text filters.

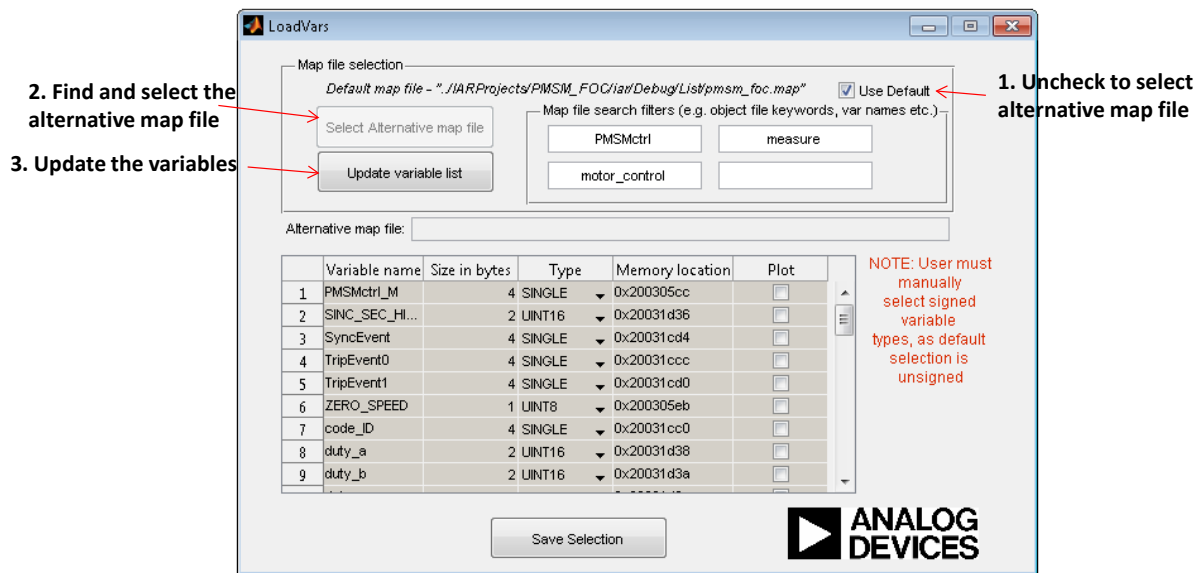


Figure 17 ADIMonitor, visualization variable selection

The next step is to select the required variables for plotting, as shown in Figure 18. Two points are important here:

1. The GUI attempts to guess the variable type based on its size, however it cannot distinguish between sign and unsigned variables, so the polarity of the variables must be manually selected in the type column.
2. The program will limit the total buffer size to 1.7k bytes. So if the combination of the buffer length, selected in the main ADIMonitor pane, and the total size of the variables selected is greater than this, the program will only select the first number of variables that gives a total buffer size <1.7kB. The number of variables plotted can be increased by reducing the buffer length, which is maximum 200, although a total maximum selected variable size of 8 bytes is allowed (e.g. 2xSINGLE or 8xINT8, or 1xSINGLE+2xINT16 etc.)

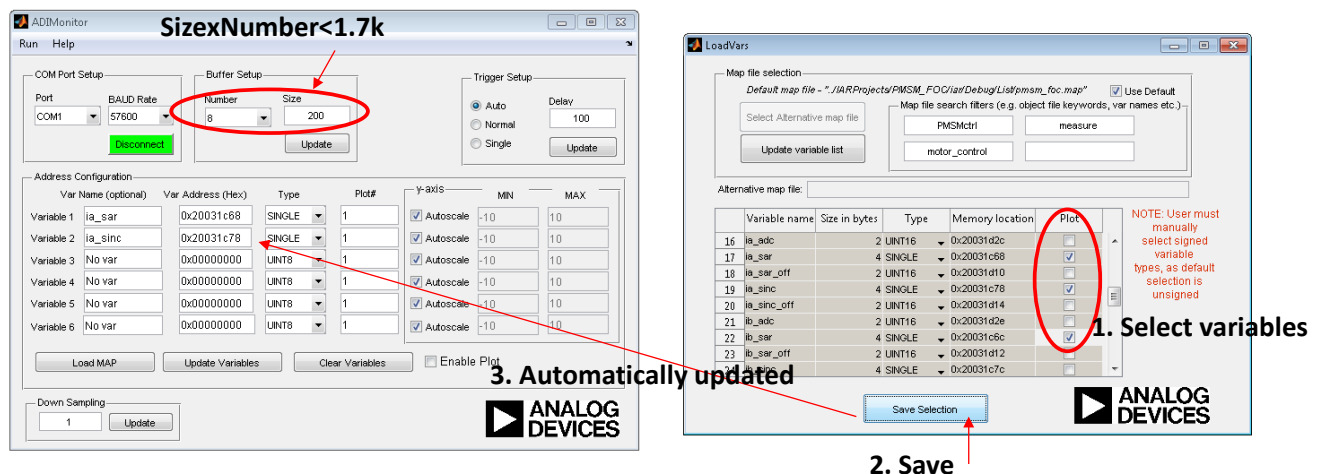


Figure 18 Loading variables

Notice that in Figure 18, three variables of type SINGLE are chosen in the LoadVars window. However, the program only updates the main window with the first 2 variables, since the total buffer size (i.e.  $3 \times 4 \times 200$  bytes) would be  $> 1.7\text{kB}$ . If the buffer size was reduced to say a length of 100 samples, then the 3 variables would be updated as the total buffer size would be  $= 3 \times 4 \times 100 = 1.2\text{kB}$ .

The next steps to prepare for plotting of variables are:

1. Select how many plots, i.e. all on the same plot, or multiple sub-plots
2. Y-axis scaling – default is autoscaling, but this does not work well for, say a step response
3. Down Sampling factor – the sampling window equals the PWM switching period x Down Sampling ratio, so for a down sampling ratio of 1, the sampling period will be  $100\mu\text{s}$ , and with a buffer length of 200, the plot time slices will be of 20ms length. To look at longer time slices, increase the down sampling factor.
4. Update the Buffer Setup
5. Update the Down Sampling factor (if it has been changed)
6. Press Update Variables
7. Check Enable Plot

At this point, the plot should appear with a periodic update of the plotted data. Note that the Trigger Setup buttons are not functional. Default triggering is Auto mode. These steps are illustrated in Figure 19.

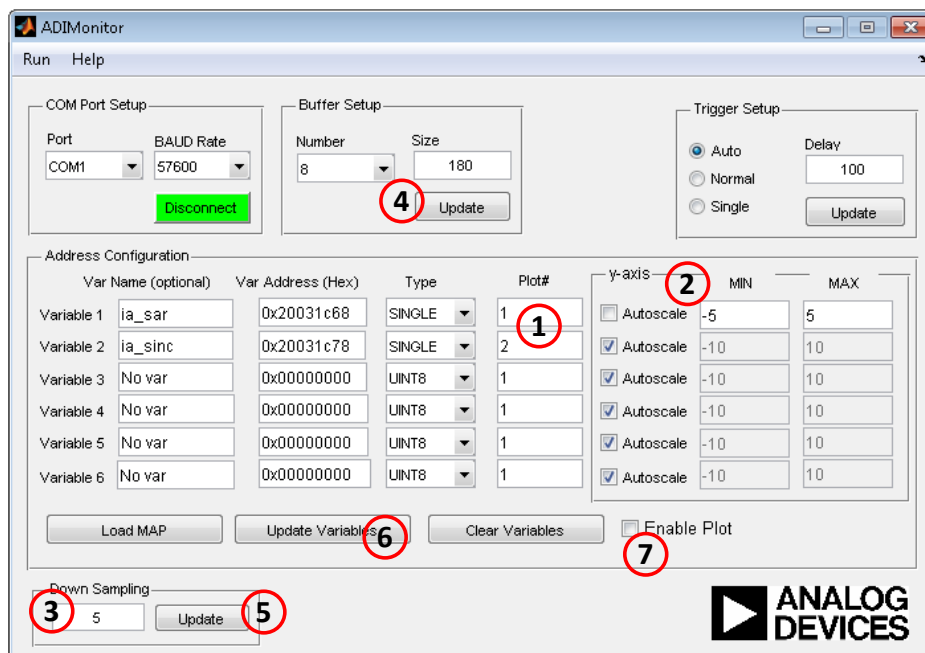


Figure 19 Update steps for data visualization

After the plot is enabled, it needs to be disabled for any changes to be made to the plotted variables, and these buttons will be greyed out while the plot is enabled. To clear the variable selection in both the

main ADIMonitor window and in the Load Vars window, press ‘Clear Variables’. Then follow the previously outlined steps in order to select a new set of variables. Some typical plot examples are shown in Figure 20.

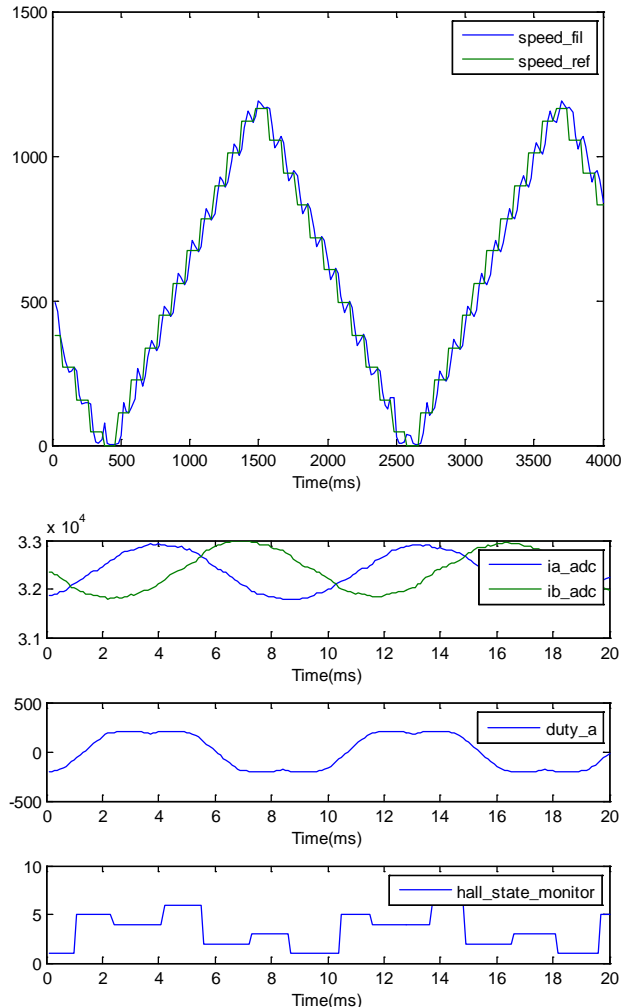


Figure 20: Example of single (top) and multiple (bottom) plots

Two additional GUI features related to data visualization are available in the motor control SW release version 0.2.0. These are single triggering and data logging. Single trigger is useful for examining a specific time slot of data around a user-defined trigger event. Single trigger is activated by selecting this option in the Trigger Setup field as illustrated in Figure 21, and by entering the trigger delay in the corresponding box. Trigger delay is specified in data samples, so, for instance if the buffer size is selected as 200 (the maximum), a delay setting of 100 will mean that the data window will contain 100 samples before the trigger event, and 100 after. In similar manner, a trigger delay of 0 will place all of the data samples immediately after the trigger event etc.

The trigger event itself must be defined in the main project C code. The template of the trigger event code is:

```
if( var > LEVEL and var_old < LEVEL) // Trigger condition - here positive going var, level= LEVEL
```

```
TrigAdiMonitor(); // Call TrigAdiMonitor to let ADIMonitor know trig has happened
var_old = var; // Update for edge detection
```

This code should be placed at the location where “var” is being updated, e.g. in the ADC ISR if the data is an ADC sampled quantity. A new set of data is then collected once “Arm Trigger” in Figure 21 is clicked, and the trigger event occurs. “Arm Trigger” can be clicked as many times as desired, with a new data set being gather each time (assuming a trigger event occurs each time).

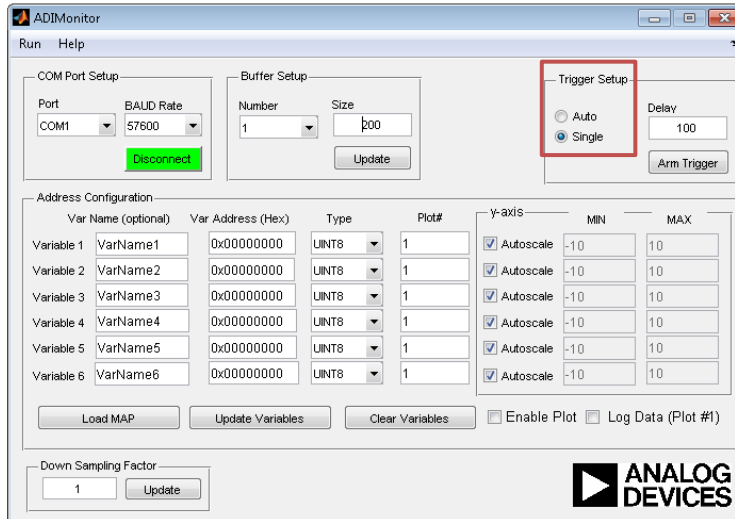


Figure 21: Single trigger in ADIMonitor

For easy post-processing of ADIMonitor variables, data sets can now be logged and saved to a MATLAB workspace file. This is only possible for variables with a Plot#=1, so any variables being logged should have this Plot# set accordingly, and “Log Data” selected as illustrated in Figure 22.

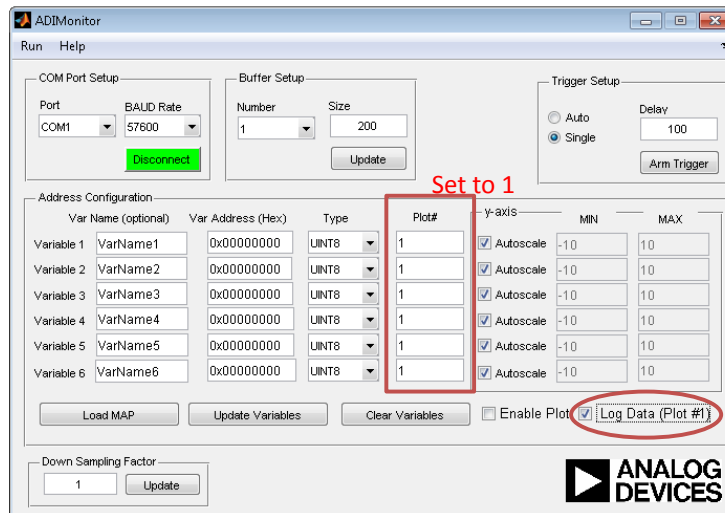


Figure 22: Data logging selection in ADIMonitor

This will result in all of the variables specified being stored in an “output.mat” workspace file. If single triggering is selected, then this file contains the triggered data set. If triggering is continuous, then the file will be constantly updated with the most recent data set. The .mat file contains a data vector/array called “plot1\_buffer” and a time stamp vector “plot1\_buffer\_t” as illustrated in Figure 23, which can be

extracted and saved for post-processing.

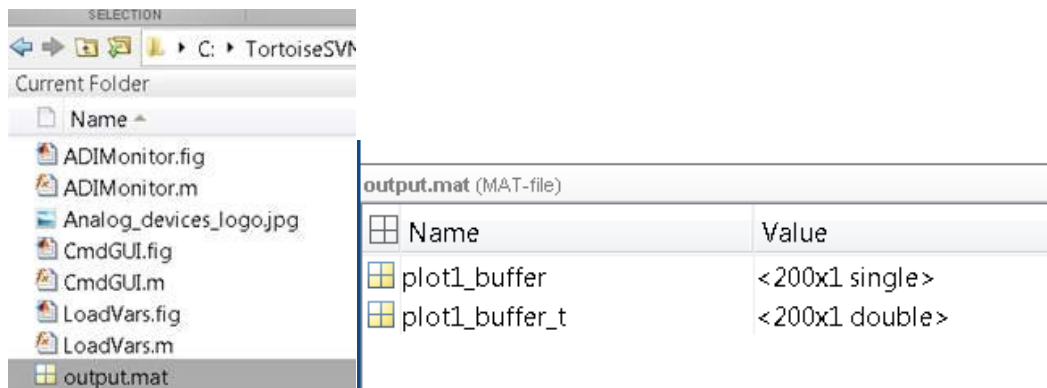


Figure 23: Data logging file and vectors

## 9 Support

You can reach Boston Engineering for FlexMC support at:

- <http://www.boston-engineering.com/services/technologies-expertise/motion-controls/flexmc/>
- E-mail: [flexmc@boston-engineering.com](mailto:flexmc@boston-engineering.com)

Boston Engineering, the FlexMC Motor Control Development Platform, and the FlexMC Kit are trademarks of Boston Engineering Corporation.

Analog Devices and Analog Devices logo are registered trademarks of Analog Devices, Inc.