Application Note

AN2356/D
11/2002

Sensorless BLDC Motor Control on MC68HC908MR32 Software Porting to Customer Motor

By Libor Prokop and Leos Chalupa
Roznov System Application Laboratory
Roznov, Czech Republic

General Description

This application note describes using the software for 3-phase sensorless brushless dc motor (BLDC) control with back-EMF zero crossing. The software is designed for Motorola's MC68HC908MR32 microcontroller unit (MCU) and dedicated for motor control; however, it could be easily ported to other derivatives of the MC68HC(9)08MRx Family.

This application note serves as a guide for application software use and parameter configuration for a customer motor. It also helps to decide if the software and control method is suitable for the specific customer application.

For a more detailed view of brushless dc motor control, application system concepts, control techniques, and software description refer to the complementary application note entitled *Sensorless BLDC Motor Control on MC68HC908MR32 — Software Description* (Freescale document order number AN2355/D).

System Concept

The application block diagram is shown in **Figure 1**. The sensorless rotor position technique detects the zero crossing points of back-EMF induced in the motor windings. The phase back-EMF zero crossing points are sensed while one of the three phase windings is not powered. The information obtained is processed in order to commutate the energized phase pair and control the phase voltage, using pulse width modulation.

The back-EMF zero crossing detection enables position recognition. The resistor network is used to step down sensed voltages to a 0–3.3 V level. Zero crossing detection is synchronized with the middle of center aligned PWM signals by the software, in order to filter high voltage spikes produced by switching the IGBTs (MOSFETs). The software selects by MUX command the phase comparator output that corresponds to the current commutation step. The multiplexer (MUX) circuit selects this signal, which is then transferred to the MCU input.

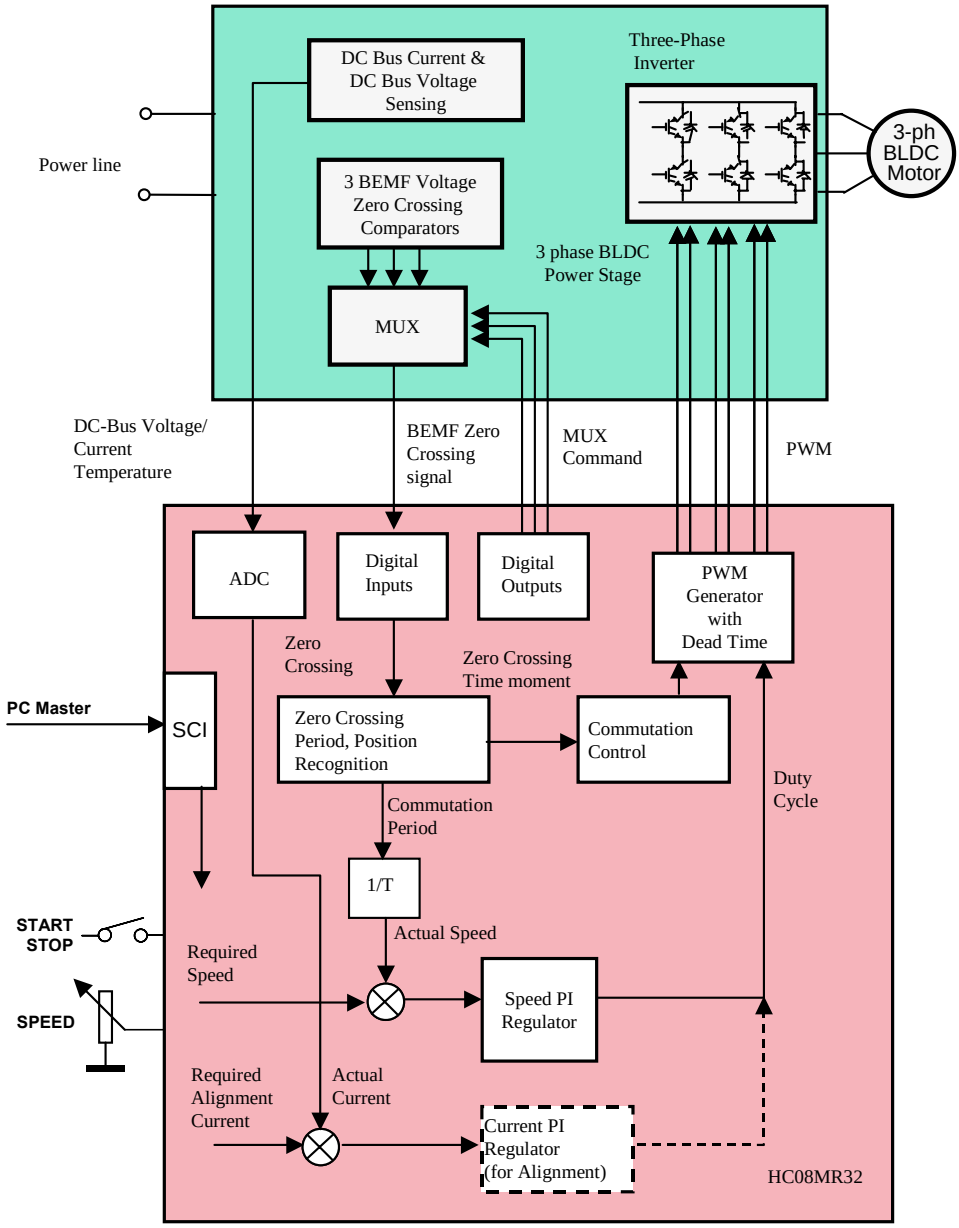

Figure 1. System Concept

The voltage drop resistor is used to measure the dc-bus current which is chopped by the pulse-width modulator (PWM). The signal obtained is rectified and amplified (0–3.3 V with 1.65 V offset). The internal MCU analog-to-digital (A/D) converter and zero crossing detection are synchronized with the PWM signal. This synchronization avoids spikes when the IGBTs (or MOSFETs) are switched and simplifies the electric circuit.

During the rotor alignment state, the dc-bus current is controlled by the current PI regulator. In the other states (motor running), the phase voltage (PWM duty cycle) is controlled by the speed PI regulator.

The A/D converter is also used to sense the dc-bus voltage and the drive temperature. The dc-bus voltage is stepped down to a 3.3-V signal level by a resistor network.

The six IGBTs (copack with built-in fly back diode), or MOSFETs, and gate drivers create a compact power stage. The drivers provide the level shifting that is required to drive the high side switch. The PWM technique is used to control motor phase voltage.

Drive Specification

The concept of the application is that of a speed-closed loop drive using back-EMF zero crossing technique for position detection. It serves as an example of a sensorless BLDC motor control system using Motorola's MC68HC908MR32 MCU. It also illustrates the usage of dedicated motor control on-chip peripherals.

The system for BLDC motor control consists of hardware and software. The application uses universal modular motion control development hardware boards, which are provided by Freescale for customer development support. For a description of these hardware boards refer to **References 3.**, **4.**, **5.**, **6.**, **7.**, and the World Wide Web at:

http://www.freescale.com

There are three board and motor hardware sets for the application:

1. High-Voltage Hardware Set — For variable line voltage 115–230 Vac and medium power (phase current < 2.93 A)
2. Low-Voltage Evaluation Motor Hardware Set — For automotive voltage (12 V) and very low power (phase current < 4 A)
3. Low-Voltage Hardware Set — For automotive voltage (12 V or possibly 42 V) and medium power (phase current < 50 A)

Software Specification

The application software is practically the same for all three hardware platforms. The only modification needed is to include one of three constants that customize the hardware and motor parameter settings.

The software (written in C language) specifications are listed in **Table 1**. A useful feature of the software is serial communication with PC master software protocol via RS232. The PC master software is PC computer software which allows reading and setting of all the system variables, and can also run html script pages to control the application from the PC. Another feature of the BLDC control software, is on-line parameter modification with PC master software, which can be used for software parameter tuning to a customer motor.

Table 1. Software Specifications

Control Algorithm	3-phase trapezoidal BLDC motor control star or delta! connected
	Sensorless, with back-EMF zero crossing commutation control
	Speed closed loop control
	Motoring mode
Target Processor	MC68HC908MR32
Language	C-language with some arithmetical functions in assembler
Compiler	Metrowerks ANSI-C/cC++ Compiler for HC08
Application Control	Manual interface (start/stop switch, speed potentiometer control, LED indication)
	PC master software (remote) interface (via RS232 using PC computer)
MCU Oscillator Frequency	4 MHz (with default software setting)
MCU Bus Frequency	8 MHz (with default software setting)
Minimal BLDC Motor Commutation Period (Without PC Master Software Communication)	333 μs (with default software setting and COEF_HLFCMT = 0.450)
Minimal BLDC Motor Commutation Period (with PC Master Software Control)	520 μs (with default software setting and COEF_HLFCMT = 0.450)
Targeted Hardware	Software is prepared to run on three optional board and motor hardware sets: • High-voltage hardware set at variable line voltage 115–230 Vac (software customizing file const_cust_hv.h) • Low-voltage evaluation motor hardware set (software customizing file const_cust_evmm.h) • Low-voltage hardware set (software customizing file const_cust_lv.h)
Software Configuration and Parameters Setting	Configuration to one of the three required hardware sets is provided by inclusion of dedicated software customizing files. The software pack contains the files const_cust_hv.h, const_cust_lv.h, and const_cust_evm.h with predefined parameter settings for running on one of the optional board and motor hardware sets. The required hardware must be selected in code_fun.c file by one of these files #include.
	Where software is configuration for different customer motors, the software configuration for any motor is provided in the dedicated customizing file, according to the hardware board used.
	PWM frequency 15.626 kHz with default software setting, possibly changeable in const.h file

Hardware and Drive Specifications

The other system specifications are determined by hardware boards and motor characteristics. The boards and their connections are shown in **Hardware Configuration**. The hardware set specifications are discussed in the following subsections.

High-Voltage Hardware Set Specification

This hardware set is dedicated for medium power (phase current < 2.93 A) and main voltage. The specifications for a high-voltage hardware and motor set are listed in **Table 2**. The hardware operates on both 230 Vac and 115 Vac mains.

Table 2. High Voltage Hardware Set Specifications

Hardware Boards Characteristics	Input voltage:	230 Vac or 115 Vac
	Maximum dc-bus voltage:	407 V
	Maximal output current:	2.93A
Motor Characteristics	Motor type:	6 poles, three phase, star connected, BLDC motor, EM Brno SM40V
	Speed range:	2500 rpm (at 310 V)
	Maximum electrical power:	150 W
	Phase voltage:	3*220 V
	Phase current:	0.55 A
Drive Characteristics	Speed range:	< 2500 rpm (determined by motor used)
	Maximum dc-bus voltage:	380 V
	Optoisolation:	Required
	Protection:	Over-current, over-voltage, and under-voltage fault protection
Load Characteristic	Type:	Varying

Low-Voltage Evaluation Hardware Set Specification

This hardware set is dedicated for 12 V voltage and very low power (phase current < 4 A). The specifications for a low-voltage evaluation hardware and motor set are listed in **Table 3**. It is targeted first of all to software evaluation with small motors.

Table 3. Low Voltage Evaluation Hardware Set Specifications

Hardware Boards Characteristics	Input voltage:	12 Vdc
	Maximum dc-bus voltage:	16.0 V
	Maximal output current:	4.0 A
Motor Characteristics	Motor type:	4 poles, three phase, star connected, BLDC motor
	Speed range:	< 5000 rpm (at 60 V)
	Maximal line voltage:	60 V
	Phase current:	2 A
	Output torque:	0.140 Nm (at 2 A)
Drive Characteristics	Speed range:	< 1400 rpm
	Input voltage:	12 Vdc
	Maximum dc-bus voltage:	15.8 V
	Protection:	Over-current, over-voltage, and under-voltage fault protection
Load Characteristic	Type:	Varying

Low-Voltage Hardware Set Specification

This hardware set is dedicated for medium power (phase current < 50 A) and automotive voltage. The specifications for a low-voltage hardware and motor set are listed in **Table 4**. The hardware power stage board is dedicated for 12 V, but can be simply configured to a 42 V supply (described in documentation for the ECLOVACBLDC board). The supplied motor is targeted for 12 V.

Table 4. Low Voltage Hardware Set Specifications

Hardware Boards Characteristics	Input voltage:	12 Vdc or 42 V
	Maximum dc-bus voltage:	16.0 V or 55.0 V
	Maximal output current:	50.0 A
Motor Characteristics	Motor type:	6 poles, three phase, star connected, BLDC motor, EM Brno SM40N
	Speed range:	3000 rpm (at 12 V)
	Maximum electrical power:	150 W
	Phase voltage:	3*6.5 V
	Phase current:	17 A
Drive Characteristics	Speed range:	< 2500 rpm
	Input voltage:	12 Vdc
	Maximum dc-bus voltage:	15.8 V
	Protection:	Over-current, over-voltage, and under-voltage fault protection
Load Characteristic	Type:	Varying

Software Suitability Guide for Customer Application and Motor

Minimal Application Speed

As it is known, the back-EMF voltage is proportionally dependent on motor speed. Since the sensorless back-EMF zero crossing sensing technique is based on back-EMF voltage, it has some minimal speed limitations! The motor start-up is solved by starting (back-EMF acquisition) state, but minimal operation speed is limited.

The minimal speed depends on many factors of the motor and hardware design, and differs for any application. This is because the back-EMF zero crossing is disturbed and effected by the zero crossing comparator threshold as explained below and in the **Effect of Mutual Inductance** and **Effect of Mutual Phase Capacitance** sections of the complementary application note entitled *Sensorless BLDC Motor Control on MC68HC908MR32 — Software Description* Freescale document order number AN2355/D).

NOTE: *Usually, the minimal speed for reliable operation is from 7% to 20% of the motor's nominal speed.*

Maximal Application Speed

The maximal motor speed is limited by the minimal commutation period:

$$\text{maximal speed[rpm]} = \frac{60(10^6)}{\text{min. commutation period [us]}*\text{COMMUT\_REV}} \quad \text{Equation 1}$$

COMMUT_REV — commutations per motor revolution, must be set according to rotor poles:

$$\text{COMMUT\_REV} = \frac{6p}{2} \quad \text{Equation 2}$$

where: p = rotor poles

The minimal commutation period is determined by execution time of the software. With default software settings and COEF_HLFCMT = 0.450 it is 333 μs, as shown in **Table 1**. So for example, the 4 pole (3-phase) motor can be controlled up to the maximal speed of 15,015 rpm.

NOTE: *Using PC master software in the application increases the minimal commutation time. This is due to the execution of PC master software routine. In this case, the minimal execution time is 520 ms. The minimal commutation period could be decreased using pure assembler code instead of C coding.*

Voltage Close Loop

As shown in **Application Hardware and Software Configuration**, the speed control is based on voltage close loop control. This should be sufficient for most applications.

Motor Suitability

Back-EMF zero crossing sensing is achievable for most of BLDC motors with a trapezoidal back-EMF. However, for some BLDC motors the back-EMF zero crossing sensing can be problematic since it is affected by unbalanced mutual phase capacitance and inductance. It can disqualify some motors from using sensorless techniques based on the back-EMF sensing.

Effect of Mutual Phase Capacitance

As described in the complementary application note entitled *Sensorless BLDC Motor Control on MC68HC908MR32 — Software Description* (Freescale document order number AN2355/D), the mutual phase capacitances play an important role in the back-EMF sensing. Usually the mutual capacitance is very small. Its influence is only significant during the PWM switching when the system experiences very high *du/dt*.The effect of mutual capacitance is shown in **Figure 2**. Channel 1 shows an example of the disturbed “branch” voltage caused by unbalanced mutual capacitance. The other phase (channel 2) is not affected because it faces balanced mutual capacitance.

An important conclusion from the complementary application note entitled *Sensorless BLDC Motor Control on MC68HC908MR32 — Software Description* (Freescale document order number AN2355/D) is the fact that only the unbalance of the mutual capacitance (not the capacitance itself) disturbs the back-EMF sensing! With both capacities equal (they are balanced) the disturbances disappear.

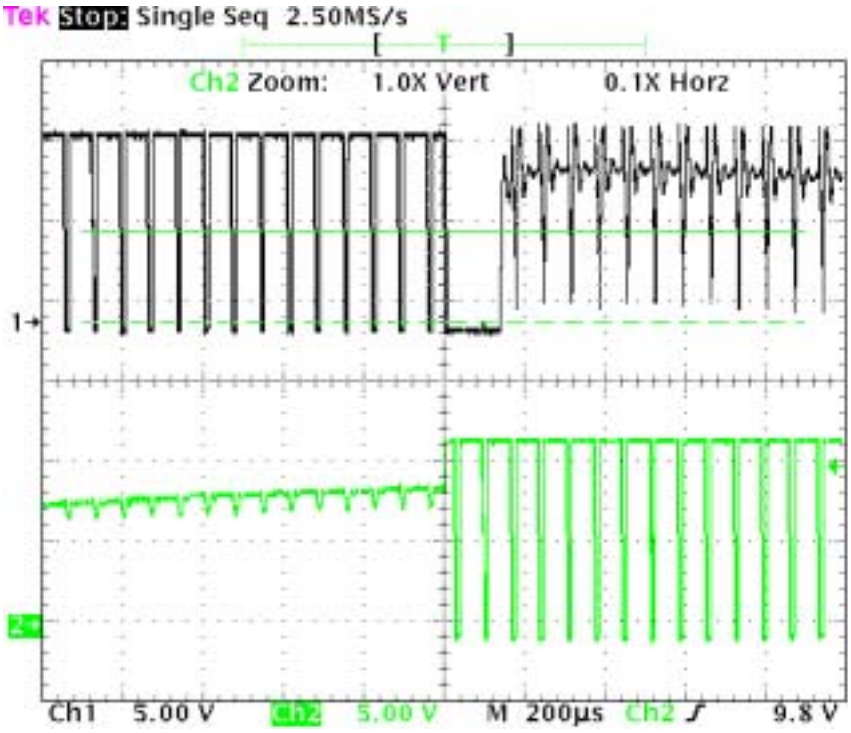

Figure 2. Disturbed Back-EMF by Unbalanced Capacity Coupling

NOTE: *Note that the configuration of the end-turns of the phase windings has a significant impact. Therefore, it must be properly managed to preserve the balance of the mutual capacity. This is especially important for prototype motors that are usually hand-wound.*

Channel 1 in **Figure 2** shows an example of the disturbed “branch” voltage which can cause impossible back-EMF zero crossing sensing.

CAUTION: *Failing to maintain balance of the mutual capacitance can easily disqualify such motors from using sensorless techniques based on the back-EMF sensing. Usually the BLDC motors with windings wound on separate poles show minor presence of the mutual capacitance. Thus, the disturbance is insignificant.*

Effect of Mutual Inductance

The negative effect on back-EMF sensing of mutual inductance, is not to such a degree as unbalanced mutual capacitance. However, it can be noticed on the sensed phase.

The difference of the mutual inductances between the coils which carry the phase current and the coil used for back-EMF sensing, causes the PWM pulses to be superimposed onto the detected back-EMF voltage. **Figure 3** shows the real measured “branch” voltage. The red curves highlight the effect of the difference of the mutual inductances. This difference is not constant.

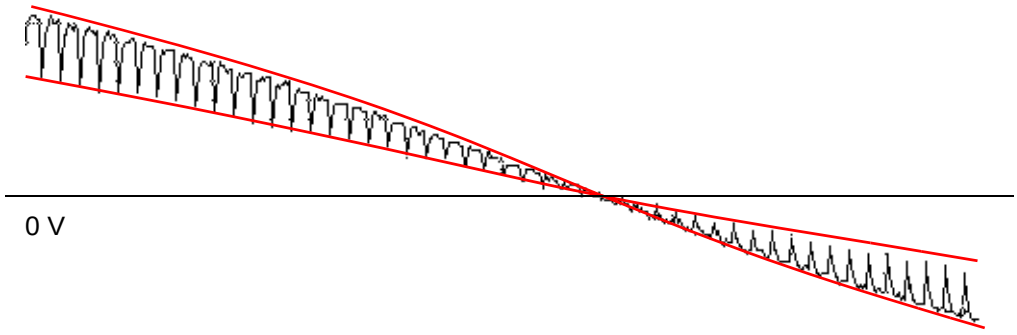

Figure 3. Mutual Inductance Effect

Due to the construction of the BLDC motor, both mutual inductances vary and they are equal at the position that corresponds to the back-EMF zero crossing detection.

NOTE: *The BLDC motor with stator windings distributed in the slots has technically higher mutual inductances than other types. Therefore, this effect is more significant. On the other hand, the BLDC motor with windings wound on separate poles, shows minor presence of the effect of mutual inductance.*

CAUTION: *However noticeable this effect, it does not degrade the back-EMF zero crossing detection, because it is cancelled at the zero crossing point. Additional simple filtering helps to reduce ripples further.*

Application Hardware and Software Configuration

Hardware Configuration

As mentioned, the software can be configured to run on one of the three hardware and motor platforms:

- **High-Voltage Hardware Set Configuration**
- **Low-Voltage Evaluation Motor Hardware Set Configuration**
- **Low-Voltage Hardware Set Configuration**

The hardware setups are shown in **Figure 4**, **Figure 5**, and **Figure 6**. These setups are described in following subsections (see **Hardware and Drive Specifications** for each platform characteristics).

The supplied controller boards for MC68HC908MR32 (ECCTRMR32) allows two possibilities for software execution:

1. MMDS Evaluation Board (KITMMDSMR32)
   Using a real-time debugger (supplied with the Metrowerks compiler) the evaluation board is connected to the controller board (ECCTRMR32) via an emulator connector. This solution is recommended for software evaluation.
2. Programmed MCU (MC68HC908MR32)
   Where a daughter board module, with MC68HC908MR32 processor, is plugged into the controller board (ECCTRMR32) instead of the emulator cable, the processor must be programmed in an external programmer. This solution is recommended for demonstration purposes and final tests.

Figure 4, **Figure 5**, and **Figure 6** show the configuration with MMDS evaluation board.

NOTE: *A detailed description of the individual boards can be found in comprehensive user's manuals belonging to each board (see **References 3.**, **4.**, **5.**, **6.**, **7.**, **8.**). Each user's manual incorporates the schematic of the board, description of individual function blocks, and bill of materials. The individual boards can be ordered from Freescale as a standard product.*

High-Voltage Hardware Set Configuration

The system configuration for a high-voltage hardware set is shown in **Figure 4**.

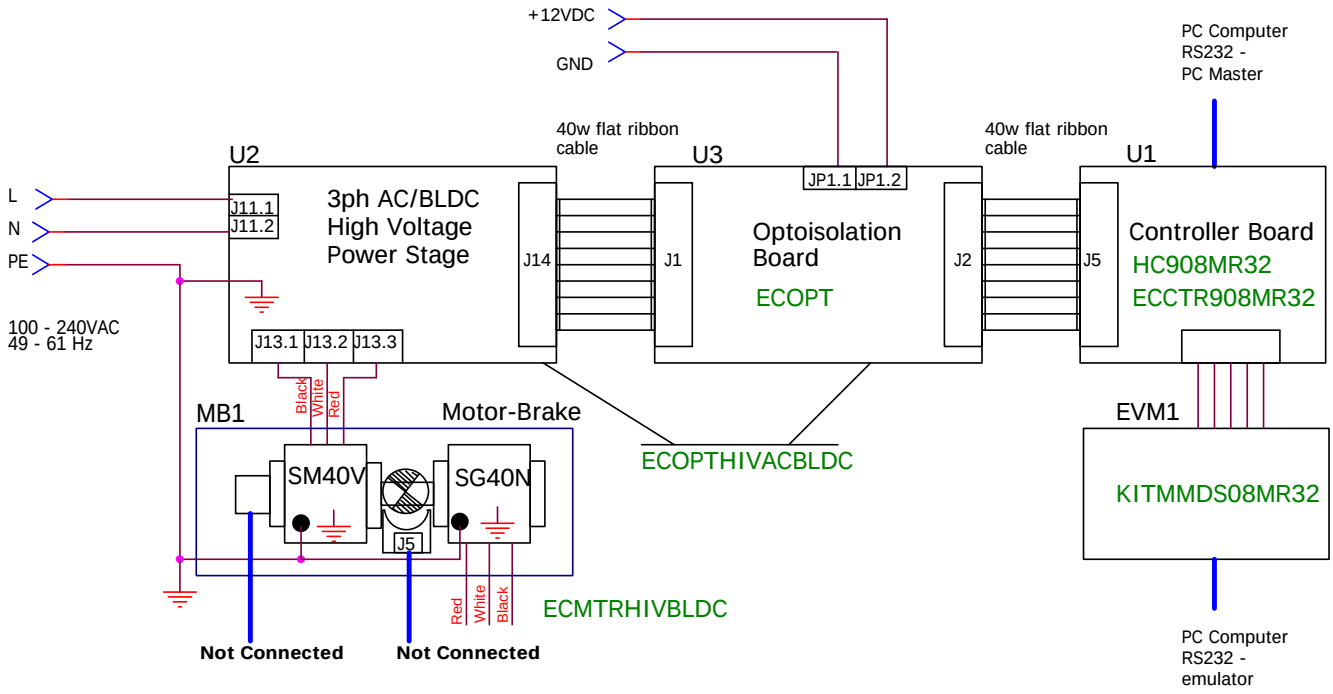

Figure 4. High-Voltage Hardware System Configuration

All the system parts are supplied and documented according to the following references:

- EVM1 — Modular Development System with EM08MR32 Daughter Board:
  - Supplied as: KITMMDS08MR32
  - Described in: Manual supplied with kit
- U1 — Controller Board for MC68HC908MR32:
  - Supplied as: ECCTR908MR32
  - Described in: *MC68HC908MR32 Control Board — User's Manual* (Freescale document order number MEMCMR32CBUM/D), see **References 3.**
- U2 — 3-Phase AC/BLDC High Voltage Power Stage:
  - Supplied in kit with optoisolation board as: ECOPTHIVACBLDC
  - Described in: *3-Phase AC Brushless DC High Voltage Power Stage User's Manual* (Freescale document order number MEMC3PBLDCPSUM/D), see **References 4.**

- U3 — Optoisolation Board
  - Supplied with 3-phase AC/BLDC high voltage power stage as: ECOPTHIVACBLDC
  - Or, supplied alone as: ECOPT–ECOPT optoisolation board
  - Described in: *Optoisolation Board User's Manual* (Freescale document order number MEMCOBUM/D), see **References 5.**

NOTE: *It is strongly recommended to use opto-isolation (optocouplers and optoisolation amplifiers) during development time to avoid any damage to the development equipment.*

- MB1 — Motor-Brake SM40V + SG40N
  - Supplied as: ECMTRHIVBLDC

For a description of the hardware boards see **References 3.**, **4.**, **5.** which can be found on the World Wide Web at:

http://www.freescale.com

Low-Voltage Evaluation Motor Hardware Set Configuration

The system configuration for a low-voltage evaluation motor hardware set is shown in **Figure 5**.

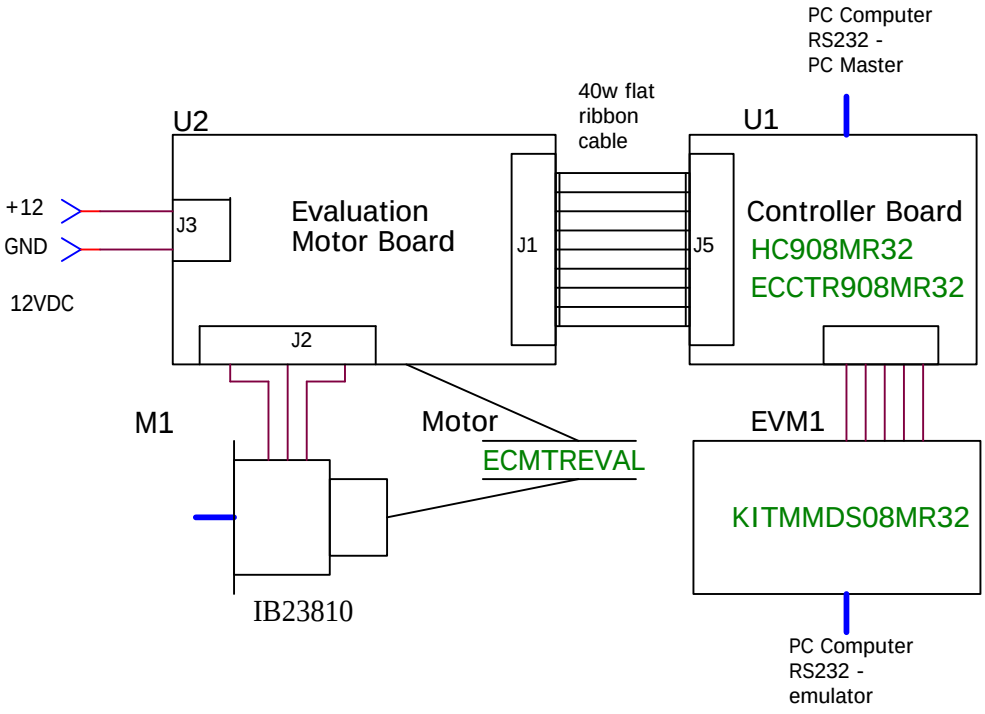

Figure 5. Low-Voltage Evaluation Motor Hardware System Configuration

All the system parts are supplied and documented according to the following references:

- EVM1 — Modular Development System with EM08MR32 Daughter Board:
  - Supplied as: KITMMDS08MR32
  - Described in: Manual supplied with kit
- U1 — Controller Board for MC68HC908MR32:
  - Supplied as: ECCTR908MR32
  - Described in: *MC68HC908MR32 Control Board — User's Manual* (Freescale document order number MEMCMR32CBUM/D), see **References 3.**
- M1 — IB23810 Motor
  - Supplied in kit with IB23810 Motor as: ECMTREVAL — Evaluation Motor Board Kit
- U2 — 3-Phase AC/BLDC Low Voltage Power Stage:
  - Supplied in kit with IB23810 Motor as: ECMTREVAL — Evaluation Motor Board Kit
  - Described in: *Freescale Embedded Motion Control Evaluation Motor Board User's Manual* (Freescale document order number MEMCEVMBUM/D) see **References 6.**

For a description of the hardware boards see **References 3.**, **6.** which can be found on the World Wide Web at:

http://www.freescale.com

Low-Voltage Hardware Set Configuration

The system configuration for low-voltage hardware set is shown in **Figure 6**.

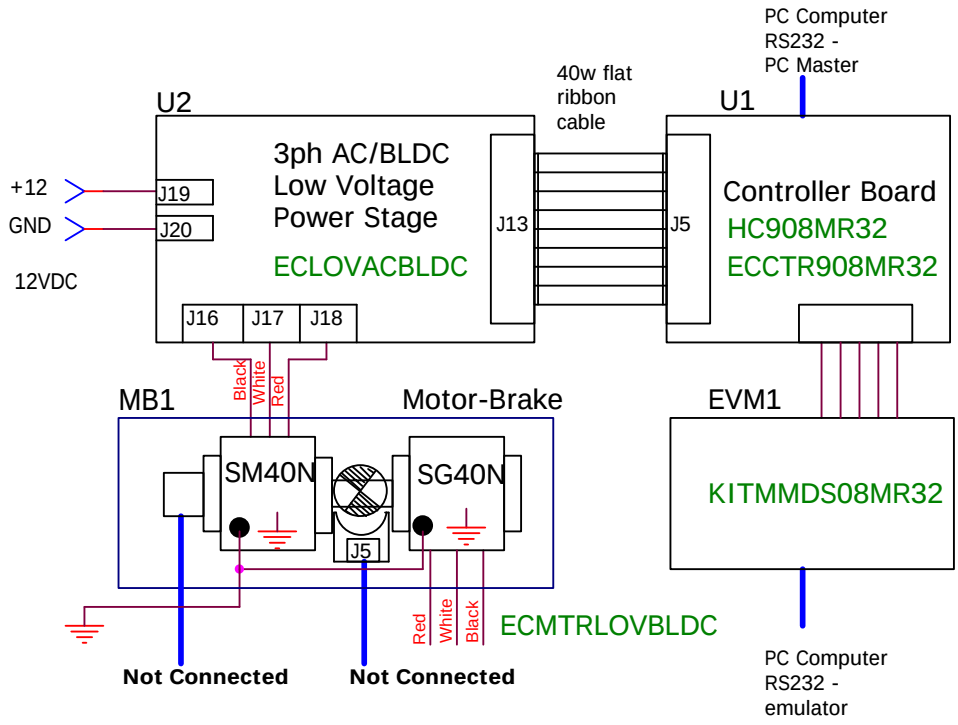

Figure 6. Low-Voltage Hardware System Configuration

All the system parts are supplied and documented according to the following references:

- EVM1 — Modular Development System with EM08MR32 Daughter Board:
  - Supplied as: KITMMDS08MR32
  - Described in: Manual supplied with kit
- U1 — Controller Board for MC68HC908MR32:
  - Supplied as: ECCTR908MR32
  - Described in: *MC68HC908MR32 Control Board — User's Manual* (Freescale document order number MEMCMR32CBUM/D), see **References 3.**
- U2 — 3-Phase AC/BLDC Low Voltage Power Stage
  - Supplied as: ECLOVACBLDC
  - Described in: *Freescale Embedded Motion Control 3-Phase BLDC Low-Voltage Power Stage User's Manual* (Freescale document order number MEMC3PBLDCLVUM/D3), see **References 7.**
- MB1 — Motor-Brake SM40N + SG40N
  - Supplied as: ECMTRLOVBLDC

For a description of the hardware boards see **References 3.**, **7.** which can be found on the World Wide Web at:
http://www.freescale.com

Controller Board Settings

Controller board settings are the same for all hardware platforms.

Jumpers JP3 and JP7 must be connected with the other jumpers disconnected. See **Figure 7**.

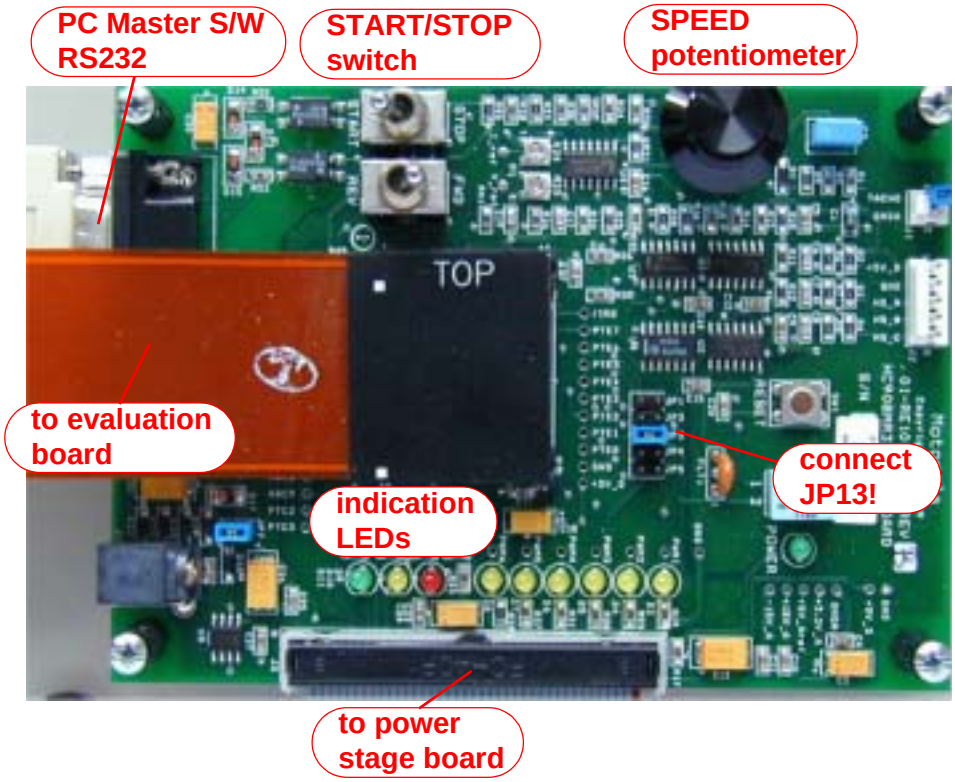

Figure 7. Controller Board

EVM Board Settings

EVM board settings are the same for all hardware platforms.

CAUTION:

Remember, the MMDS MCU clock must be set to 4 MHz. Change the crystal oscillator, or set oscillator W1 to MMDS, and set for 4 MHz in the debugger!

Software Setup

In order to run the sensorless BLDC application the following software is needed:

- Metrowerks compiler for HC08 — installed on your PC computer
- Sensorless BLDC application HC08 software files (located in bldc_zerocros08MR32 directory)

For application PC master software (remote) control, the following software is needed:

- PC master software for PC — installed on your PC computer
- Sensorless BLDC application PC master software control files (located in bldc_zerocros08MR32\pc_master directory)

Both the HC08 and PC master software control files for the sensorless BLDC application are delivered together in the **bldc_zerocros08MR32** directory. It consists of files listed in **Application HC08 Software Files**.

Application HC08 Software Files

The application HC08 software files are:

- **...\bldc_zerocros08MR32\bldc_zerocross.mcp**, application project file
- **...\bldc_zerocros08MR32\sources\const_cust_hv.h**, definitions for software customizing for high voltage (230/115 Vac) power board
- **...\bldc_zerocros08MR32\sources\const_cust_evm.h**, definitions for software customizing for EVM motor board (12 V low power)
- **...\bldc_zerocros08MR32\sources\const_cust_lv.h**, definitions for software customizing for low voltage (12 Vdc) power board

NOTE: *Change the code_fun.c file to include (#include) one of const_cust_hv.h, const_cust_evmm.h, or const_cust_lv.h files according to the hardware platform used! See **Table 5**.*

Table 5. Required Software Configuration for Dedicated Hardware Platform

Hardware Platform	Dedicated Customizing File	Required Software Configuration
High-voltage hardware	const_cust_hv.h	#include const_cust_hv.h into code.fun.c (done with default software setting)
Low-voltage evaluation motor hardware	const_cust_evmm.h	#include const_cust_evmm.h into code.fun.c
Low-voltage hardware	const_cust_lv.h	#include const_cust_lv.h into code.fun.c

- **...\bldc_zerocros08MR32\sources\code_fun.c**, program C language functions
- **...\bldc_zerocros08MR32\sources\code_fun.h**, program C language functions header
- **...\bldc_zerocros08MR32\sources\const.h**, main program definitions
- **...\bldc_zerocros08MR32\sources\mr32io.h**, MC68HC908MR32 registers definitions file

- **...\bldc_zerocros08MR32\sources\mr32_bit.h**, MCHC908MR32 register bits definitions file
- **...\bldc_zerocros08MR32\sources\bldc08.c**, main program
- **...\bldc_zerocros08MR32\sources\code_start.c**, motor alignment and starting (back-EMF acquisition) state functions
- **...\bldc_zerocros08MR32\sources\code_start.h**, motor alignment and starting (back-EMF acquisition) state function header
- **...\bldc_zerocros08MR32\sources\code_run.c**, motor running state function
- **...\bldc_zerocros08MR32\sources\code_run.h**, motor running state function header
- **...\bldc_zerocros08MR32\sources\code_isr.c**, program interrupt functions
- **...\bldc_zerocros08MR32\sources\code_isr.h**, program interrupt functions header
- **...\bldc_zerocros08MR32\sources\ram.c**, general RAM definitions
- **...\bldc_zerocros08MR32\sources\ram.h**, general RAM declarations header
- **...\bldc_zerocros08MR32\sources\ram_bit.h**, general RAM bits definitions header
- **...\bldc_zerocros08MR32\sources\ram_cust_param.c**, RAM variables for software customizing definitions
- **...\bldc_zerocros08MR32\sources\ram_cust_param.h**, RAM variables for software customizing header declarations
- **...\bldc_zerocros08MR32\sources\tab_cust.c**, constants/tables definitions
- **...\bldc_zerocros08MR32\sources\tab_cust.h**, constants/tables definitions header
- **...\bldc_zerocros08MR32\sources\pcmaster.c**, PC master software communication subroutines
- **...\bldc_zerocros08MR32\sources\pcmaster.h**, PC master software communication subroutines header
- **...\bldc_zerocros08MR32\sources\code_asm.asm**, program assembler functions
- **...\bldc_zerocros08MR32\sources\code_asm.h**, program assembler functions header
- **...\bldc_zerocros08MR32\prms\default.prm**, linker command file

Application PC Master Software Control Files

The application PC master software control files are:

- **...\bldc_zerocros08MR32\pc_master\BLDC.pmp**, PC master software project file
- **...\bldc_zerocros08MR32\pc_master\source**, directory with PC master software control page files

Software Execution

Build

To build the BLDC sensorless with the back-EMF zero crossing application, open the **bldc_zerocross.mcp** project file and execute the *Make* command, as shown in **Figure 8**. This will build and link the application and all needed Metrowerks libraries.

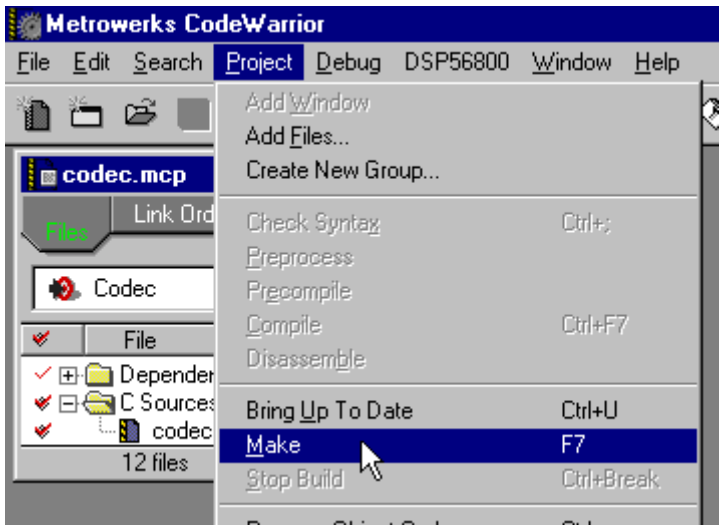

Figure 8. Execute *Make* Command

Execute from Evaluation Board

To execute the application from MMDS evaluation board (KITMMDSMR32), choose the *Project/Debug* command in the Code Warrior IDE. This will start real-time debugger, load firmware, and application software to evaluation board MMDS.

The application should then be started from the real-time debugger IDE by the *Run/Start* command. For more help with these commands, refer to the CodeWarrior tutorial documentation located in the CodeWarrior installation directory.

NOTE: *Remember, the MMDS MCU clock must be set to 4 MHz. Change the crystal oscillator, or set oscillator W1 to MMDS, and set 4 MHz in the real-time debugger — MMDS0508/target signals/4 MHz!*

Once the application is running, move the RUN/STOP switch to the RUN position and set the required speed with the SPEED potentiometer. If successful, the BLDC motor will be spinning.

NOTE: *If the RUN/STOP switch is set to the RUN position when the application starts, toggle the RUN/STOP switch between the STOP and RUN positions to enable motor spinning. This is a protection feature that prevents the motor from starting when the application is executed from CodeWarrior.*

Execute from Pre-programmed MCU

When the software is built, the S-record file **bldc_zerocros08mr32_MMDS.sx** is generated in:

...\bldc_zerocros08MR32\bin\bldc_zerocros08mr32_MMDS.sx

NOTE: *The software must be built (see **Build**) to generate by linker the **bldc_zerocros08mr32_MMDS.sx** file (last update)*

This S-record file can be used for programming of MC68HC908MR32 MCU devices. An external programmer (e.g., Freescale M68HC08 serial programmer) must be used to program the device!

The programmed MCU, with MC68HC908MR32 daughter board module, can then be plugged into the controller board (ECCTRMR32) instead of the emulator cable.

Application Control

This BLDC sensorless motor control application can operate in two modes:

1. **Manual Operating Mode**
2. **PC Master Software (Remote) Operating Mode**

Manual Operating Mode

In the manual operating mode, the drive is controlled by the RUN/STOP switch and the required speed is set by the SPEED potentiometer. The RUN/STOP switch enables/disables motor spinning. The yellow LED will light whenever the application software correctly executes (so, it will also light when motor spinning is disabled or at a fault state).

- When motor spinning is enabled and starts spinning (alignment or starting/back-EMF acquisition state), only the yellow LED lights.
- When motor rotation is enabled and the motor runs with speed close loop (running state), the green LED lights (yellow LED also lights).
- If an over-current or over-voltage fault occurs, the internal fault logic is asserted and the application enters a fault state indicated by a red LED blinking (yellow LED lights).This state can be exited only by an application RESET or setting RUN/STOP switch to STOP.

NOTE: *It is strongly recommended that you inspect the entire application to locate the source of the fault, before starting again.*

PC Master Software (Remote) Operating Mode

In the PC master software (remote) operating mode, the drive is controlled remotely from a PC through the serial communication interface (SCI) channel of the MCU device via an RS-232 physical interface. The drive is enabled by the RUN/STOP switch, which can be used to safely stop the application at any time.

For the PC master software (remote) control it is necessary to have PC master software installed on your PC computer!

Start the PC master software application:

...\bldc_zerocros08MR32\pc_master\BLDC.pmp

After you start the PC master software, press “control page” to make the control window visible! **Figure 9** illustrates the PC master software Control Window.

NOTE: *After you start the PC master software, the algorithm block description window appears instead of the PC master control window; therefore, press “control page”. If the PC master software project (**..pmp** file) is unable to control the application, it is possible that the wrong load map (**..\bin\bldc_zerocros08mr32_MMDS.map** file) has been selected. PC master software uses the load map to determine addresses for global variables being monitored. Once the PC master project has been launched, this option may be selected in the PC master window under Project/Select Other Map FileReload.*

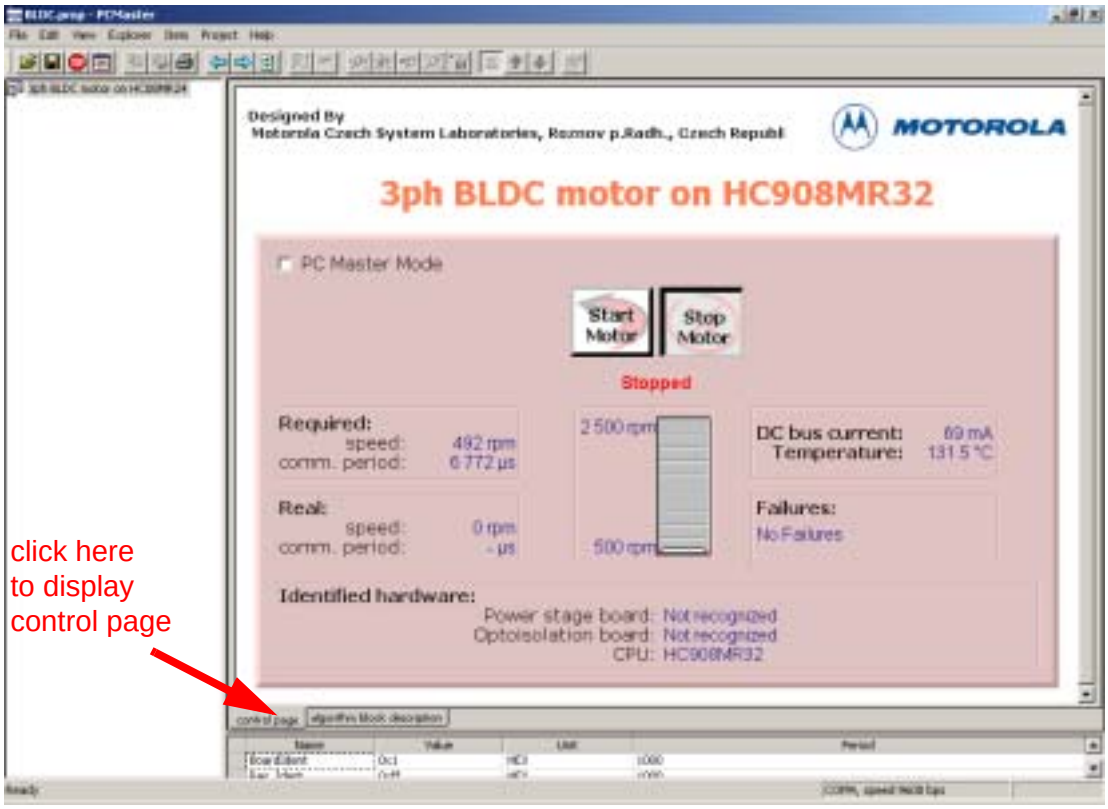

Figure 9. PC Master Software Control Window

The following control actions are supported:

- Setting PC master software/manual control mode (PC Master Mode Radio button)

NOTE: *Application control from PC master software requires that PC master software control mode must be set. Before changing PC master software/manual control mode (by PC Master Mode Radio button) the controller board START/STOP switch must be set to STOP. This is a protection feature that prevents the motor from unexpected starts!*

- Start the motor (Start Motor button)

NOTE: *To start the motor with PC master software control mode, two conditions must be fulfilled. START/STOP switch on the controller board must be set to START and Start Motor button on PC master software control page must be "pressed down".*

- Stop the motor (Stop Motor button)
- Set the required speed of the motor (via bar graph)
- Clear failures (Clear Failures button)

PC master software displays the following information:

- Required speed of the motor
- Actual speed of the motor
- dc-bus current
- Temperature of the power stage
- Fault status (no fault, over-voltage, under-voltage, or over-current in dc-bus)
- Motor status — running/stopped

NOTE: *Hardware board identification is not implemented in the software. Therefore, the PC master software control window displays Power Stage board, Optoisolation board: Not Recognized.*

If the fault status is different from the no faults (when over-current, over-voltage, or under-voltage fault), the red LED blinks and the motor is stopped. This state can be exited by application RESET or Clear Failures button on the PC master software control page.

NOTE: *It is strongly recommended that you inspect the entire application to locate the source of fault before starting it again.*

Introduction to Software Parameters Setting and Tuning for Customer Motor

This section describes how to modify the software parameters for any BLDC motor and some hardware adaptations. The software parameters can be evaluated from a PC computer using PC master software, so the first subsection describes tuning the PC master software project file.

- A follow-up for software customizing to a customer motor is shown in **Figure 10**.
- Before starting the software modification for a customer motor and application, it is recommended that you check the application and motor suitability. This is explained in **Software Suitability Guide for Customer Application and Motor**.
- The **Parameters File Selection** must be made according to **Hardware Configuration** used.
- If a modified hardware power stage is used, the appropriate constants in const_cust_x.h file must be set as described in **Software Customizing to Power Stage**.
- If a low-voltage board with a modification for 42 V is used, the constants **VOLT_HW_MAX** and **VOLT_MAX_FAULT_V** must be changed.
- If one of the three standard power stages is used, the software customizing to power stage is not needed.
- For software customizing to customer motor and application, a setting must be made as explained in:
  - **Software Customizing to Motor — Voltage and Current Settings**
  - **Alignment Current and Current Regulator Setting**
  - **Software Customizing to Motor — Commutation and Start-up Control Setting**
  - **Software Customizing to Motor — Speed Control Setting**.

Accomplishing the above steps should be sufficient for most applications. However, in some cases there may be a need for advanced software customizing (see **Figure 11**) with changes to motor PWM frequency or to the current regulator sampling period as explained in **PWM Frequency and Current Sampling Period Setting**.

If there's still a problem running the motor, check if the motor is suitable for sensorless control with back-EMF zero crossing (see **Figure 12**) as described in **Motor Suitability**. Then, again check the application suitability.

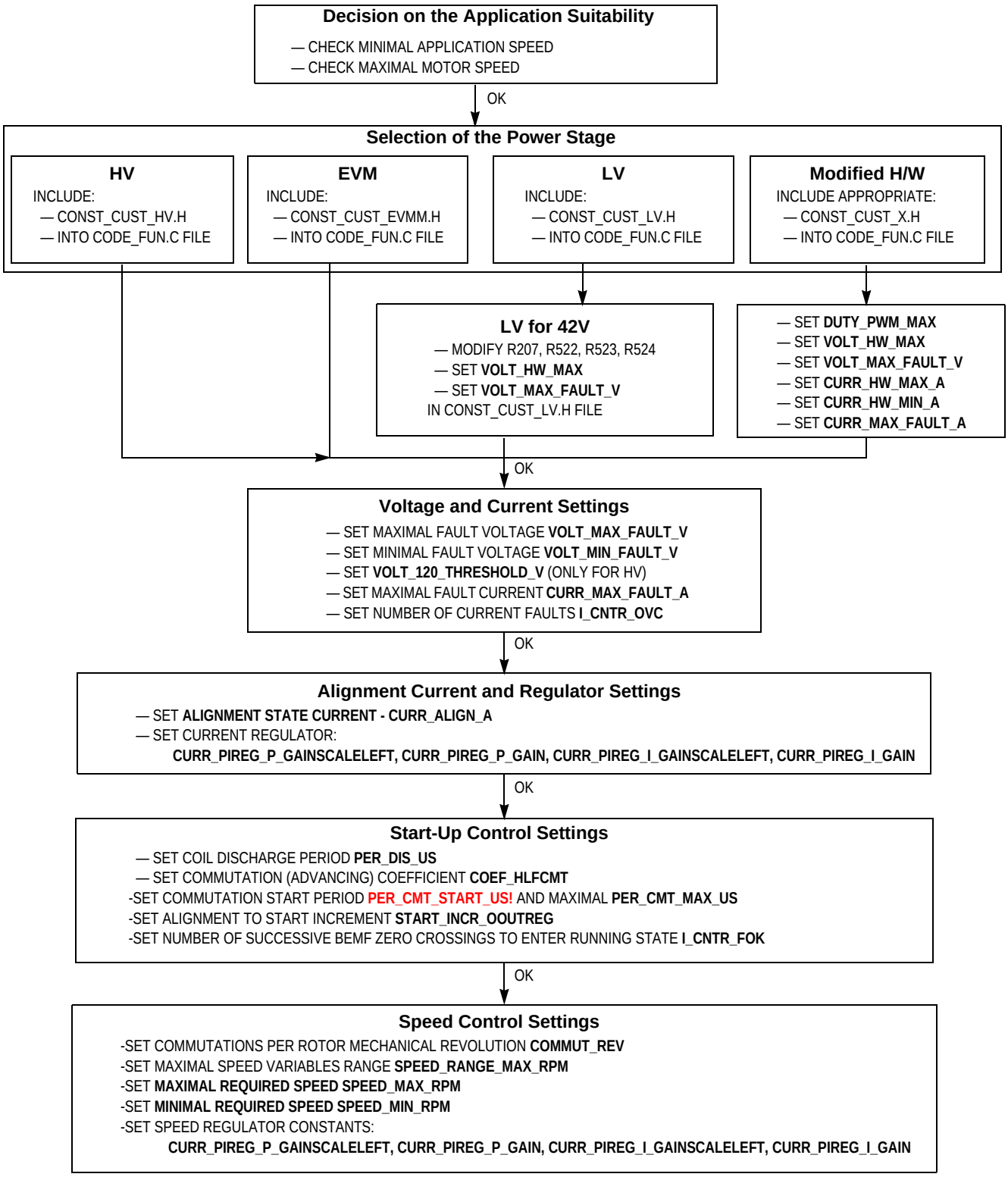

Figure 10. Follow-up for Software Customizing to Customer Motor

PWM Frequency and Current Sampling Period Setting

— SET PWM FREQUENCY **SET_PER_PWM**
— SET CURRENT SAMPLING PERIOD **SET_PER_CS**
— SET PERIOD FROM PWM RELOAD TO CURRENT SAMPLING **SET_PER_CS**

Figure 11. Follow-up for Advanced Software Customizing

Decision on the Motor Suitability

— MEASURE FREE PHASE BACK-EMF VOLTAGE FOR THE MOTOR MUTUAL CAPACITANCE EFFECT
— MEASURE FREE PHASE BACK-EMF VOLTAGE FOR THE MOTOR INDUCTANCE CAPACITANCE EFFECT

Decision on the Application Suitability

— CHECK MINIMAL APPLICATION SPEED
— CHECK MAXIMAL MOTOR SPEED

Figure 12. Follow-up for Software Customizing Trouble Shouting

Software Parameters Tuning with PC Master Software Project File

Sensorless BLDC software is provided with a PC master software project file for on-line software parameters tuning (see **Figure 13**). This file supports:

- Remote application control
- Key software parameters modification for:
  - Current parameters tuning
  - Start-up parameters tuning
  - Speed parameters tuning
- PC master software “oscilloscope” windows with required variables

The remote application control uses the same control page as described in **PC Master Software (Remote) Operating Mode**. Moreover, the tuning file incorporates subprojects for a dedicated system variables setting, and PC master software “oscilloscope“ windows for watching dedicated parameters (variables).

NOTE: *For software parameter tuning with PC master software, it is necessary to have PC master software installed on your PC computer!*

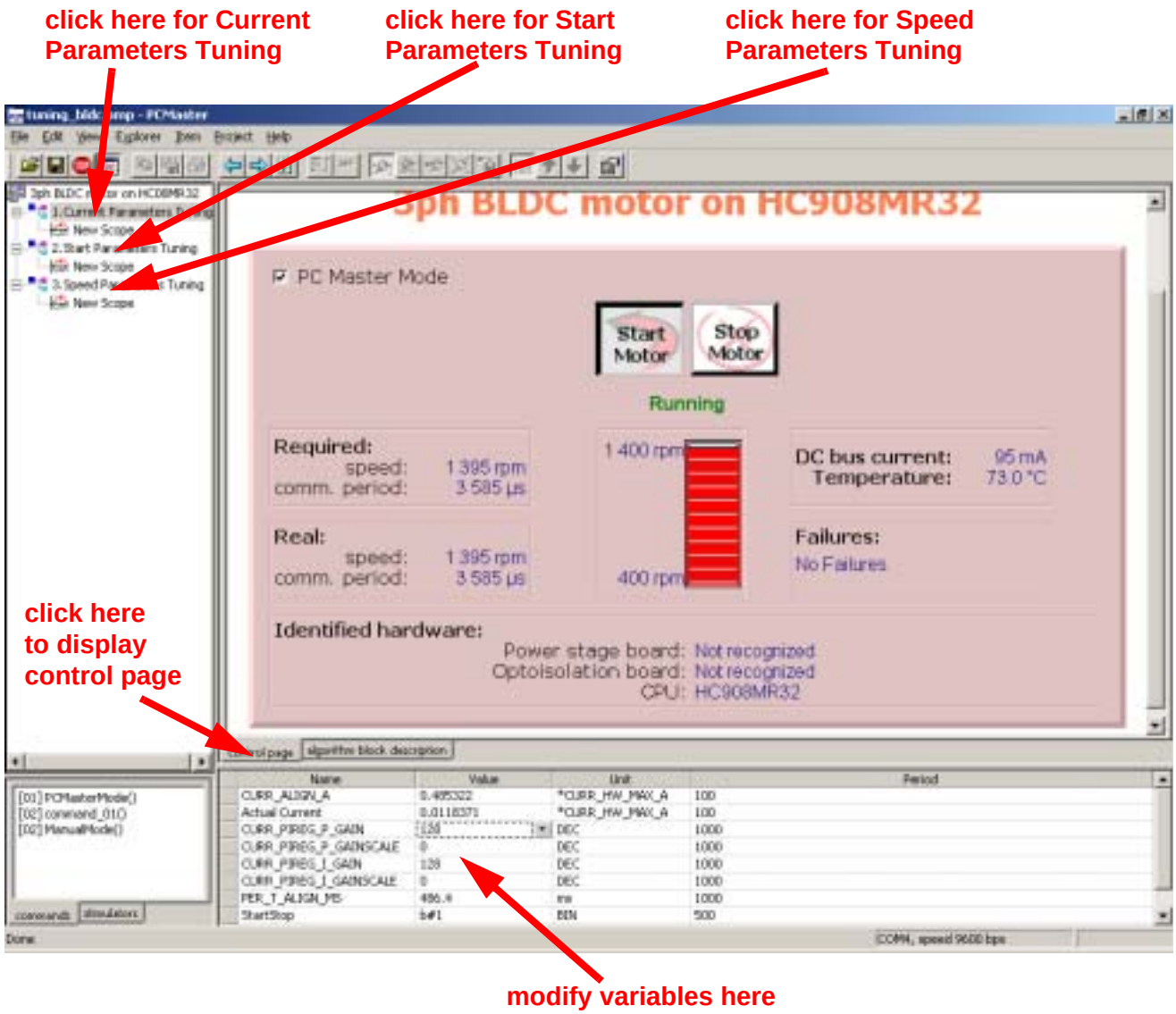

Figure 13. PC Master Software Parameters Tuning Control Window

Start the PC master software parameters tuning application:

...\bldc_zerocros08MR32\pc_master\tuning_bldc.pmp

After you start the PC master software, you can choose which parameters you are going to tune (current, start-up, speed parameters — see **Figure 13**). Then you can press “control page” to make the control window visible (and provide control in the same way as in **PC Master Software (Remote) Operating Mode**). Or, you can display the oscilloscope window (see **Figure 14**). You can then modify the variable values in the variable window (**Figure 14**), which is visible for both control page or oscilloscope page turned on. The variables can be modified according to their defined limits.

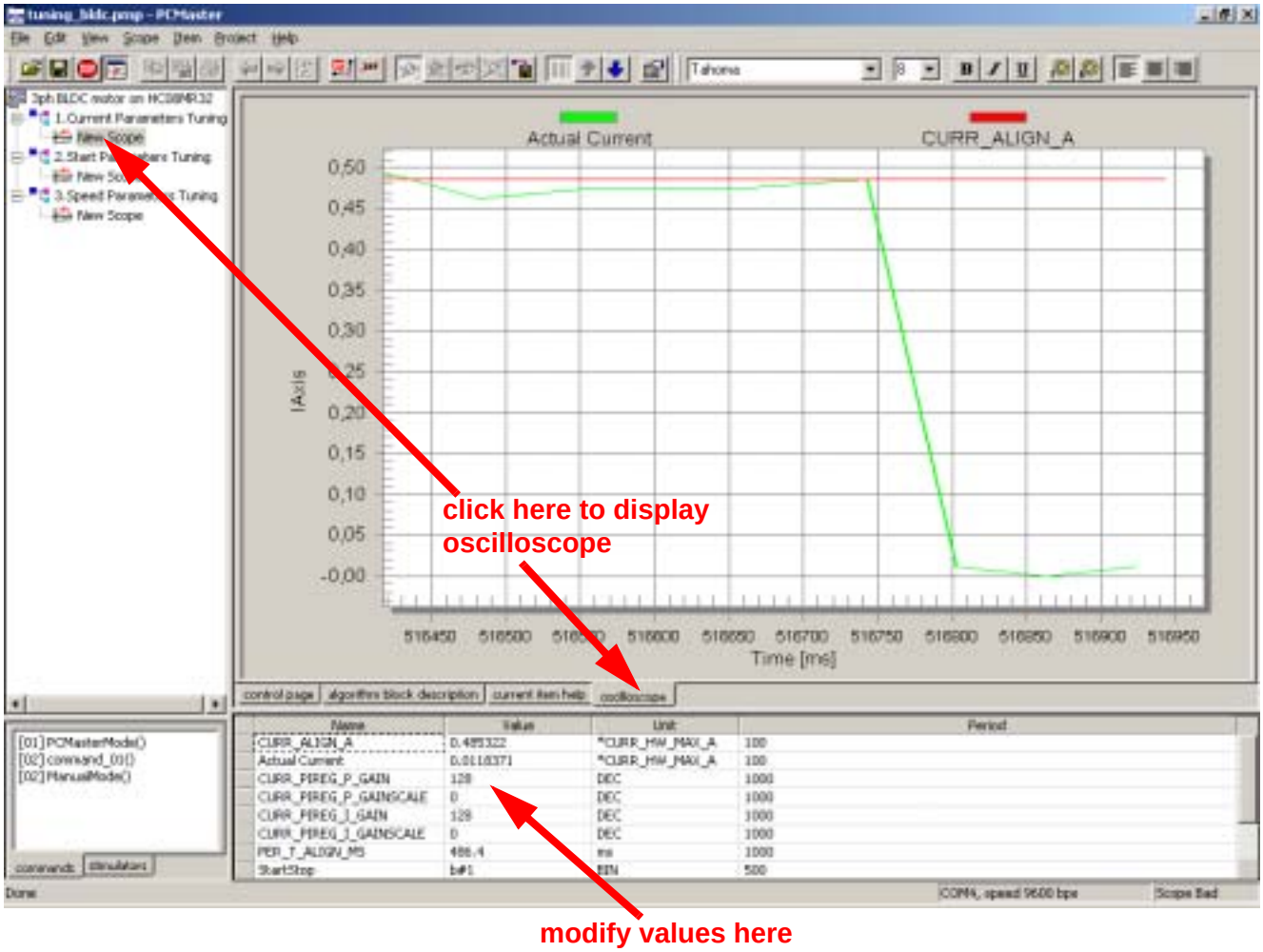

Figure 14. PC Master Software Parameters Tuning Control Window

NOTE: The software parameters can be temporarily modified and evaluated using the PC master software tuning file. But, the parameter settings are not stored in the non-volatile memory (after reset the software loads parameters from const_cust_x.h file). When you finish the software parameters evaluation, you must open one of the **const_cust_hv.h**, **const_cust_evmm.h**, and **const_cust_lv.h** files and manually modify the parameters according to the final variable values evaluated (otherwise, you will get default setting after the MCU reset!).

Software Parameters Setting Follow-up

The software is provided with three parameters sets (files const_cust_hv.h, const_cust_evmm.h, and const_cust_lv.h) configured for three hardware and motor kits (HV, LV, and EVM), as was described before. But, the software can be configured for other 3-phase trapezoidal BLDC motors (or possibly other hardware board parameters).

The motor control drive usually needs setting/tuning for:

- Current/voltage parameters
- Dynamic parameters

The parameter configurations must be set in source code before compilation. However, some parameters can also be temporarily changed using PC master software (experimental setting). Finally, when an appropriate parameter value is found, it can be set in the source code. The software parameters settings are described in the subsections below and in the software code by comments.

You should proceed with some steps to customize the software according to your motor (or hardware) characteristics. The source code is commented with descriptive labels to simplify the process.

Labels in the Files const_cust_hv.h, const_cust_evm.h, const_cust_lv.h

Most of the software parameter settings are provided in one of these files: const_cust_hv.h, const_cust_evmm.h or const_cust_lv.h. The required steps are marked:

`/* MUST_CHANGE_nn: */`
: Label for changes which must be set (changed) when adapting software for a motor

`/* MUST_CHANGE_nn_EXPER: */`
: Label for changes which must be set (changed) when adapting software for a motor — the setting can be done experimentally

`/* MUST_IF_HW_CHANGE_nn */`
: Label for changes which must be set (changed) when a power stage board different from high voltage power board is used

`/* CAN_CHANGE_nn */`
: Label for changes which can be set (changed) when adapting software for a motor, but usually the setting is not needed

`/* CAN_CHANGE_nn_EXPER */`
: Label for changes which can be set (changed) when adapting software for a motor, but usually the setting is not needed — the setting can be done experimentally

Labels in the File const.h

The other parameters, like motor PWM frequency and current sampling period can be set in the file const.h. The required steps are marked:

`/* CAN_CHANGE_FPWM_n */`

Label for definitions which should be modified, when changing PWM frequency

`/* CAN_CHANGE_PERCURSAMP_n */`

lLabel for definitions which should be corrected, when changing current sampling period

Let's follow the next sections, or the labels in the source code to customize the software.

Parameters File Selection

As explained before (see **Software Setup**) one of the following files is used for most of software parameters configuration:

...\bldc_zerocros08MR32\sources\const_cust_hv.h, definitions for software customizing, for high-voltage (230/115 Vac) power board

...\bldc_zerocros08MR32\sources\const_cust_evm.h, definitions for software customizing for EVM motor board (12 V low power)

...\bldc_zerocros08MR32\sources\const_cust_lv.h, definitions for software customizing for low-voltage (12 Vdc) power board

According to hardware used, the designated file (see **Table 5**) must be selected by including:

...\bldc_zerocros08MR32\sources\code_fun.c, program c language functions

NOTE: *The following parameter settings will be provided in the selected file. Therefore, it will be referred to as* ***const_cost_x.h*** *in the following sections.*

Software Customizing to Power Stage

The hardware boards parameters customizing is provided in the **const_cust_x.h** file.

NOTE: *Skip this section, when standard modular motion control development hardware boards are used without any changes. When low-voltage power stage* ECLOVACBLDC *is configured for 42 V, some* ***const_cust_x.h*** *changes are needed.*

For setting, follow the labels **MUST_IF_HW_CHANGE_nn** in file ***const_cust_x.h*** from nn = 1. Detailed description starts here.

An example of software customizing to power stage is shown in **Example of Software Customizing to Hardware**.

Maximal PWM Duty Cycle

Maximal PWM duty cycle [-]:

```
/* MUST_IF_HW_CHANGE_1: */
#define DUTY_PWM_MAX 0.96
```

Range: <0,1>

Proportional value of maximal PWM duty cycle is determined by power stage boards used.

DUTY_PWM_MAX must be set for any hardware customizing. Some hardware boards need maximal duty cycle<1, in order to charge high side drivers for power inverters.

Voltage Setting Hardware Customizing

Maximal measurable voltage determined by hardware voltage sensing [V]:

```
/* MUST_IF_HW_CHANGE_2: */
#define VOLT_HW_MAX  407.0
```

Range: <0,infinity)

VOLT_HW_MAX must be changed when voltage sensing range is different from default hardware.

Maximum limit of dc-bus voltage allowable for the hardware [V]:

```
/* MUST_IF_HW_CHANGE_3 */
#define VOLT_MAX_FAULT_V 380.0
```

Range: <0,VOLT_RANGE_MAX>

VOLT_MAX_FAULT_V determines the maximal voltage when the drive fault state should be entered. So the constant **VOLT_MAX_FAULT_V** must be set according to maximal voltage limit of the power stage or the motor, using the lower value. Therefore, setting this constant is also mentioned in **Maximal and Minimal Voltage Limits Setting** under a different label (**CAN_CHANGE_1**).

Current Setting Hardware Customizing

Maximal measurable current determined by hardware current sensing [A]:

```
/* MUST_IF_HW_CHANGE_4: */
#define CURR_HW_MAX_A  2.93
```

Range: <0,infinity)

Minimal measurable current determined by hardware current sensing [A]:

```
/* MUST_IF_HW_CHANGE_5: */
#define CURR_HW_MIN_A (-2.93)
```

Range: (-infinity,0>

CURR_HW_MAX_A and **CURR_HW_MIN_A** must be changed when current sensing range is different from default hardware.

Maximal limit of dc-bus current allowable for the hardware [A]:

```
/* MUST_IF_HW_CHANGE_6: */
#define CURR_MAX_FAULT_A 1.5
```

Range: <0,CURRENT_RANGE_MAX_A>

CURR_MAX_FAULT_A determines the maximal current when the drive fault state should be entered. So, it must be set to the maximum current allowed for the power stage or the motor (see also **Maximal and Minimal Current Limits Setting**)

Example of Software Customizing to Hardware

Let's have low-voltage power stage ECLOVACBLDC modified to 42 V (from 12 V) as described in its documentation. So, software must be customized for hardware changes. Because of low-voltage set, the **const_cust_lv.h** must be modified

1. Maximal PWM duty cycle remains the same:
   ```
   #define DUTY_PWM_MAX 0.942
   ```
2. Modified maximal measurable voltage is 55 V, so set:
   ```
   #define VOLT_HW_MAX 55.0
   ```
3. Maximum limit of dc-bus voltage should be set according to motor or application requirements, but VOLT_MAX_FAULT_V > 42V
   ```
   #define VOLT_MAX_FAULT_V 63.0
   ```
4. Board modified to 42 V has maximal measurable current unchanged
   ```
   #define CURR_HW_MAX_A  2.93
   #define CURR_HW_MIN_A (-2.93)
   ```
5. Maximum limit of dc-bus current should remain unchanged or set according to motor or application requirements:
   ```
   #define CURR_MAX_FAULT_A 45.0
   ```

When the software parameters are set for the hardware, you should follow the settings in **Software Customizing to Motor — Voltage and Current Settings**.

Software Customizing to Motor — Voltage and Current Settings

The software parameter settings according to customer motor are described in this section.

NOTE: *First of all, voltage and current settings need to be done. For settings which must be done, follow the labels **MUST_CHANGE_nn** and **MUST_CHANGE_EXPER_nn** in file **const_cust_x.h** where nn = 1.*

*For changes which can be done (but usually are not necessary), follow the labels **CAN_CHANGE_nn** and **CAN_CHANGE_EXPER_nn** in file **const_cust_x.h***

Detailed description starts here.

Maximal and Minimal Voltage Limits Setting

Most of voltage limit settings do not necessarily need to be done:

Maximal limit of dc-bus voltage [V]:

```
/* CAN_CHANGE_1: */
#define VOLT_MAX_FAULT_V 380.0
```

Range: <0,VOLT_RANGE_MAX>

VOLT_MAX_FAULT_V determines the maximal voltage when the drive fault state should be entered. So, the constants **VOLT_MAX_FAULT_V** must be set according to maximal voltage limit of the motor or the power stage, using the lower value. Therefore, the setting of this constants is also mentioned in **Voltage Setting Hardware Customizing** under a different label **MUST_IF_HW_CHANGE_3**. It should be changed when there are problems with over-voltage.

Minimal limit of dc-bus voltage [V]:

```
/* CAN_CHANGE_2_EXPER: */
#define VOLT_MIN_FAULT_V 100.0
```

Range: <0,VOLT_RANGE_MAX>

VOLT_MIN_FAULT_V determines the minimal voltage when the drive fault state should be entered. So the constants **VOLT_MIN_FAULT_V** must be set according to minimal voltage limits of the motor application. It should be changed when there are problems with under-voltage.

dc-bus voltage threshold mains 120V/230V [V]:

```
/* CAN_CHANGE_10: */
#define VOLT_120_THRESHOLD_V 150
```

Range: <0,VOLT_RANGE_MAX>

120 V voltage threshold setting should only be used for high-voltage hardware. It determines if 120 or 230 V mains voltage will be detected by software. But the VOLT_120_THRESHOLD_V detection has no importance for the software functionality! For low-voltage hardware the VOLT_120_THRESHOLD_V should be set to 0.

Maximal and Minimal Current Limits Setting

Most of current limits settings do not necessarily need to be done.

Current offset limit for fault during calibration (initialization) [V]:

```
/* CAN_CHANGE_4: */
#define OFFSET_MAX_CURR_V (1.65+0.225)
```

Range: <0,5>

When PWM is off, the default hardware determined offset should be 1.65 V.The actual offset is checked during current calibration. The fault offset limit should be:

```
OFFSET_MAX_CURR_V = Default h/w offset
+ minimal allowed offset error                Equation 3
```

OFFSET_MAX_CURR_V should only be changed if there are over-current problems during current offset calibration (at MCU initialization).

Maximal limit of dc-bus current [A]:

```
/* CAN_CHANGE_3: */
#define CURR_MAX_FAULT_A 1.5
```

Range: <0,CURRENT_RANGE_MAX_A>

CURR_MAX_FAULT_A should be changed for a motor with maximal allowable current lower than the board.

Initial value for OVer-Current Counter [-]:

```
/* CAN_CHANGE_5: */
#define I_CNTR_OVC   0x04
```

Range: <0,255>

I_CNTR_OVC determines the number of current samples with current value > **CURR_MAX_FAULT_A** needed before entering the drive fault state. The current sampling period is **PER_CS_T1_US** = 128 μs at default software and PWM frequency setting. **I_CNTR_OVC** should normally not be changed. Lower value of **I_CNTR_OVC** secures a fast, safer over-current switch-off. High value of **I_CNTR_OVC** secures an unexpected over-current switch-off.

Alignment Current and Current Regulator Setting

The current during alignment state (before motor starts) [A]:

```
/* MUST_CHANGE_1_EXPER: */
#define CURR_ALIGN_A 0.55
```

Range: <0,CURRENT_RANGE_MAX_A>

It is recommended that nominal motor current value be set. Sometimes when power source is not able to deliver the required current, it is necessary to set a lower value then nominal motor current.

NOTE: *CURR_ALIGN_A can be evaluated with PC master software tuning file tuning_bldc.pmp.*

It might also be necessary to set the current PI regulator constants:

```
/* MUST_CHANGE_2_EXPER: */
#define CURR_PIREG_P_GAINSCALELEFT 0
```

Range: <0,8>

```
/* MUST_CHANGE_3_EXPER: */
#define CURR_PIREG_P_GAIN  128
```

Range: <0,255>

where the current regulator proportional gain is:

$$KP = CUR\_PIREG\_P\_GAIN * 2^{CURR\_OUREG\_P\_GAINSCALELEFT} \quad \text{Equation 4}$$

```
/* MUST_CHANGE_4_EXPER: */
#define CURR_PIREG_I_GAINSCALELEFT 0
```

Range: <0,8>

```
/* MUST_CHANGE_5_EXPER: */
#define CURR_PIREG_I_GAIN  64
```

Range: <0,255>

where the current regulator integral gain is:

$$KP = CUR\_PIREG\_I\_GAIN * 2^{CURR\_PIREG\_I\_GAINSCALELEFT} \quad \text{Equation 5}$$

These constants can be calculated according to regulators theory. The current sampling (regulator execution) period is **PER_CS_T1_US** = 128 μs, at the default software setting. Normally it does not need to be changed (if change is required see **PWM Frequency and Current Sampling Period Setting**). Another recommended solution is an experimental setting.

NOTE: *CURR_PIREG_P_GAINSCALELEFT, CURR_PIREG_P_GAIN, CURR_PIREG_I_GAINSCALELEFT, CURR_PIREG_I_GAIN can be evaluated with PC master software tuning file tuning_bldc.pmp.*

We suggest using PC master software with tuning file tuning_bldc.pmp for regulator parameters evaluation. You can use this procedure:

1. Set **const_cust_x.h**:
   CURR_PIREG_P_GAINSCALELEFT 0
   CURR_PIREG_P_GAIN 0
   CURR_PIREG_I_GAINSCALELEFT 0
   CURR_PIREG_I_GAIN 0
2. Temporarily change the software: in **code_start.c** file, label TUNING_1 enable goto Align (it will cause infinite time for alignment state, where the current is tuned)
3. Build and run the code (see **Software Execution**, **Build**, **Execute from Evaluation Board**)
4. Start the PC master software tuning project
5. Select Current Parameters Tuning subproject (see **Software Parameters Tuning with PC Master Software Project File**) in order to be able to modify the current regulator
6. You can see the actual current (and required alignment current) on the Current Parameters Tuning\New Scope, or measure the powered motor coil current on real oscilloscope
7. Set PC master software control mode, and start the motor (see **Application Control** and **PC Master Software (Remote) Operating Mode**)

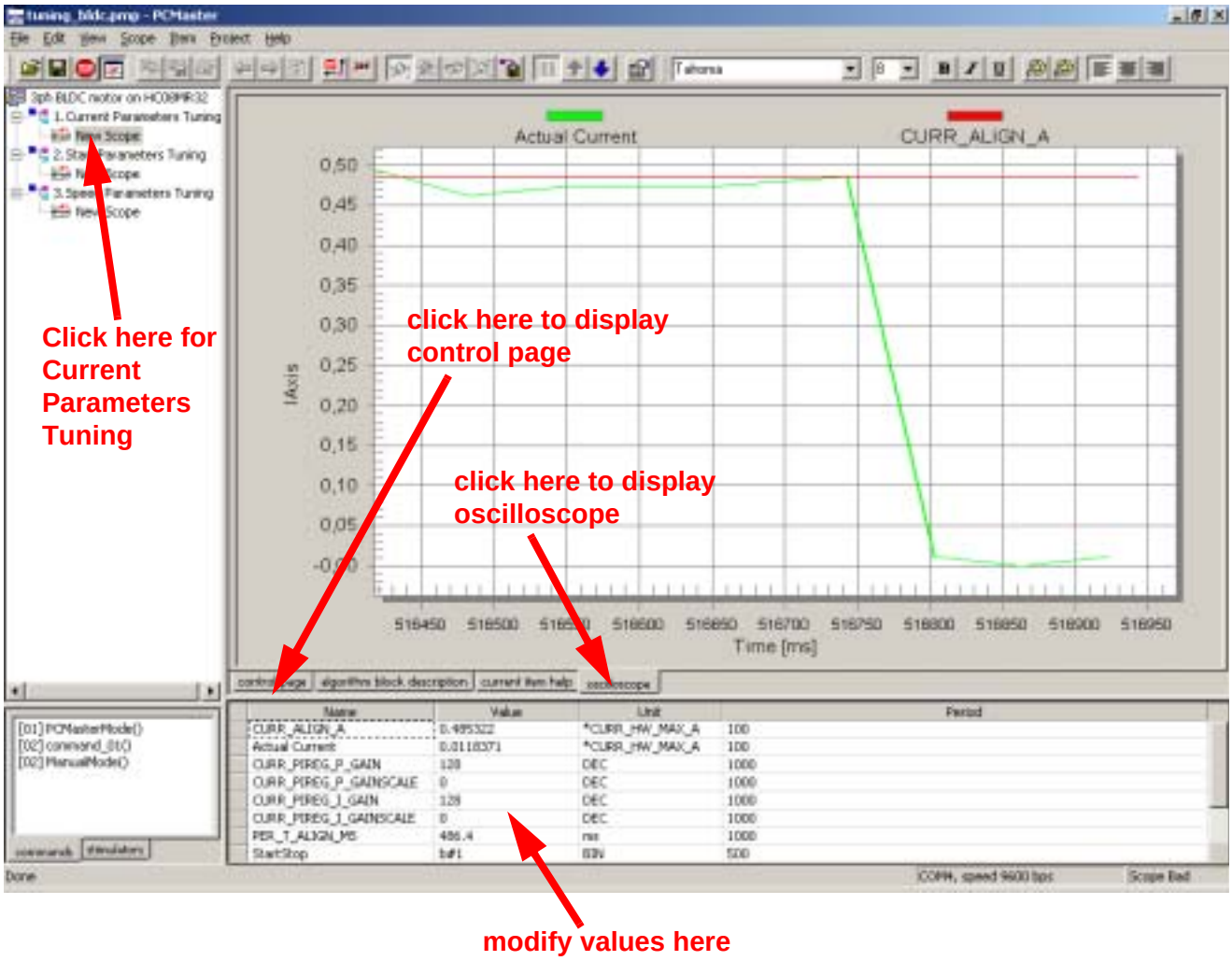

Figure 15. PC Master Software Current Parameters Tuning Window

8. Increase, step by step, the proportional gain CURR_PIREG_P_GAIN with PC master software, until current noise or oscillation appears, or up to 128
9. If CURR_PIREG_P_GAIN is set to 128, do further proportional gain increase CURR_PIREG_P_GAINSCALELEFT with PC master software, steps 0, 1, 2...8, otherwise leave CURR_PIREG_P_GAINSCALELEFT as 0
10. Increase, step by step, the integral gain CURR_PIREG_I_GAIN with PC master software, up to current oscillation or noise, or up to 128
11. If CURR_PIREG_I_GAIN is set to 128, do further integral gain increases to CURR_PIREG_I_GAINSCALELEFT with PC master software, steps 0, 1, 2...8; otherwise, leave CURR_PIREG_I_GAINSCALELEFT as 0

12. You can further evaluate the setting of the regulator parameters in order to get a smoother current waveform, or until the regulation seems to be performing well
13. Open **const_cust_x.h** and modify the regulator parameters with the final variable values evaluated with PC master software
14. Change the software back to normal: in code_start.c file, label TUNING_1 remove goto (modify as comment): /* goto Align */ (it will allow finishing Alignment state when alignment period ends)
15. Build the code (see **Software Execution**, **Build**)
16. You can also tune regulator dynamic characteristics of current transients (steps **17.** to **26.**) or finish the regulators tuning
17. Run the code (see **Software Execution** and **Execute from Evaluation Board**)
18. Start the PC master software tuning project
19. Select Current Parameters Tuning subproject (see **Software Parameters Tuning with PC Master Software Project File**) in order to be able to modify the current regulator
20. You can see the actual current (and required alignment current) on the Current Parameters Tuning\New Scope, or measure the powered motor coil current on real oscilloscope
21. Set PC master software control mode and start motor (see **Application Control** and **PC Master Software (Remote) Operating Mode**)
22. Observe the current transient at Alignment start, then stop motor (or reset software)
23. Then modify the regulator parameters with PC master software as in steps **8.**, **9.**, **10.**, and **11.**
24. Repeat steps **21.** to **23.** until regulation is improved
25. Open **const_cust_x.h** and modify the regulator parameters with the final variable values evaluated with PC master software
26. Build the code (see **Software Execution** and **Build**)

The last Alignment setting constant is Alignment Time period [ms]:

```
/* MUST_CHANGE_6_EXPER: */
#define PER_T_ALIGN_MS  1000.0
```

Range: <0,PER_BASE_T3_ALIGN_US/1000/255>

This period can be set experimentally. This constant can also be evaluated using PC master software tuning file. This period must be high enough to let the rotor stabilize during Alignment state. It is recommended that you begin with large values such as 20,000 ms, then it can be lowered. The period should be set to ensure that the rotor (and, therefore, also the current) is stabilized at the end of Alignment state.

Software Customizing to Motor — Commutation and Start-up Control Setting

When all voltage and current settings are done, the motor commutation and start-up parameters need to be set.

For settings which must be done, follow the labels **MUST_CHANGE_nn**, **MUST_CHANGE_EXPER_nn** in file ***const_cust_x.h***.

For changes, which can be done (but usually are not necessary), follow the labels **CAN_CHANGE_nn**, **CAN_CHANGE_EXPER_nn** in file ***const_cust_x.h***

NOTE: *Thanks to the Freescale patented start-up technique, the start parameters setting is quite simple and reliable. However, in order to reliably start the motor, the commutation control constants must be properly set.*

Detailed description starts here.

Commutation Parameters

Commutation time period to discharge coil current [μs]

```
/* MUST_CHANGE_7: */
#define PER_DIS_US 300.0
```

Range: <0,minimal commutation period*COEF_TOFF>

It is the maximal allowed current decay period, determined by motor winding and maximal current.

Must be:

PER_DIS_US < minimal motor commutation period[μs]***COEF_TOFF**

where: **COEF_TOFF** is commutation Toff period coefficient from const.h file explained in the complementary application note entitled *Sensorless BLDC Motor Control on MC68HC908MR32 — Software Description* (Freescale document order number AN2355/D).

NOTE: *If **PER_DIS_US** is too high, it can cause commutation errors at high speed*

Half Commutation (advancing) Coefficient [-]:

```
/* CAN_CHANGE_9: */
#define COEF_HLFCMT  0.375
```

Range: <0,1>

COEF_HLFCMT, multiplied by commutation period, determines the time from back-EMF zero crossing to motor commutation. So, it sets the electrical angle from back-EMF zero crossing to motor commutation step. The software controls BLDC motor with a 6-step commutation (six commutations per one electrical rotation), which means 60° between commutations. For ideal commutation with no advancing (no field weakening), the back-EMF zero crossing should be just in the middle between commutations, which

means that the electrical angle (ZC-Cmt angle), between back-EMF zero crossing and commutation, is 30°.

$$COEF\_HLFCMT = \frac{ZC\text{-}Cmt\ angle}{60} \qquad \text{Equation 6}$$

ZC-Cmt angle = 15° for COEF_HLFCMT = 0.25

ZC-Cmt angle = 22.5° for COEF_HLFCMT = 0.375

ZC-Cmt angle = 30° for COEF_HLFCMT = 0.5

In the real system, the ZC-Cmt angle is a little bit greater than the theoretical calculation. This is due to the response time of the hardware back-EMF zero crossing sensing. Therefore, the default software setting is **COEF_HLFCMT** = 0.375

Normally, **COEF_HLFCMT** should only be changed if you need a different commutation angle (time from back-EMF zero crossing to commutation). For example, for motor field weakening.

The relation between **COEF_HLFCMT** and the commutation can also be defined by Advance_angle, which is the electrical angle shift from ideal commutation.

$$Advance\_angle = 30\ Deg - ZC\text{-}Cmt\ angle \qquad \text{Equation 7}$$

$$COEF\_HLFCMT = \frac{1}{2} - \frac{Advance\_angle}{60} \qquad \text{Equation 8}$$

Advance_angle = 15° for COEF_HLFCMT = 0.25

Advance_angle = 7.5° for COEF_HLFCMT = 0.375

Advance_angle = 0° for COEF_HLFCMT = 0.5

The relation between back-EMF zero crossing and the commutation is explained in the **Running — Commutation Time Calculation** section of the complementary application note entitled *Sensorless BLDC Motor Control on MC68HC908MR32 — Software Description* (Freescale document order number AN2355/D).

Start-up Constants and Maximal Commutation Period

Constants defining start-up need to be changed according to the drive dynamics.

Start Commutation Period [μs]:

```
/* MUST_CHANGE_8_EXPER: */
#define PER_CMT_START_US 4000.0
```

Range: <0,PER_CMT_MAX_US/2>

PER_CMT_START_US is the period used to calculate the first (start) commutation period.

PER_CMT_START_US period must be changed for any motor accommodation. It can be set experimentally. If the motor displays errors during Starting (Back-EMF Acquisition) state, beginning Running state, or has a low start-up torque, DO decrease or increase the **PER_CMT_START_US** value. **Table 6** shows typical setting examples.

Must be:

PER_CMT_START_US<=PER_CMT_MAX_US/2

NOTE: *Setting this constant is an empirical process. It is difficult to use a precise formula, because there are many factors involved which are difficult to obtain in the case of a real drive (motor and load mechanical inertia, motor electromechanical constants, and sometimes also the motor load). So they need to be set with a specific motor.*

***PER_CMT_START_US** can be evaluated with PC master software tuning file tuning_bldc.pmp.*

Table 6. Start-up Period

Motor Size	Typical PER_CMT_START_US	First-to-Second Commutation Step Period	Second-to-Third Commutation Step Period
Slow motor/high load and motor mechanical inertia	8000.0 μs	8 ms	8–16 ms
Fast motor/high load and motor mechanical inertia	2000.0 μs	2 ms	2–4 ms

Maximal commutation period limit [μs]:

```
/* CAN_CHANGE_6_EXPER: */
#define PER_CMT_MAX_US 65536.0
```

Range: <0,65535*UNIT_PERIOD_T2_US>

Usually it is not recommended to change **PER_CMT_MAX_US**. The change is only necessary if the commutation period at start-up is to long.

Alignment to Start Increment of the regulators output [-]:

```
/* CAN_CHANGE_7_EXPER: */
#define START_INCR_OOUTREG 20.0
```

Range: <-128,127>

START_INCR_OOUTREG should not necessarily be changed for a motor accommodation. It can be set experimentally. If the motor has a low torque, increase the value. If the motor starts with a high speed, then slows down by regulator, decrease the value.

NOTE: ***START_INCR_OOUTREG** can be evaluated with PC master software tuning file tuning_bldc.pmp.*

Number of successive feedbacks necessary to enter the Running state [-]:

```
/* CAN_CHANGE_8_EXPER: */
#define I_CNTR_FOK 0x03
```

Range: <0,255>

The motor starts spinning with Starting (Back-EMF Acquisition) state. The software enters regular Running state with speed regulation after **I_CNTR_FOK** back-EMF successive commutation steps are done.

Usually it is not recommended to change **I_CNTR_FOK**, but it can be evaluated when there are problems with motor start up.

NOTE: ***I_CNTR_FOK*** *can be evaluated with PC master software tuning file uning_bldc.pmp.*

We suggest using PC master software with tuning file tuning_bldc.pmp for start-up parameters evaluation. You can use this procedure:

1. Ensure that the Alignment current and regulator were properly set (**Alignment Current and Current Regulator Setting**) in **const_cust_x.h**
2. Ensure that **PER_DIS_US** and **COEF_HLFCMT** are properly set in **const_cust_x.h**
3. Set #define **PER_CMT_START_US** in **const_cust_x.h** according to **Table 6**.
4. Ensure **PER_CMT_START_US**<=**PER_CMT_MAX_US**/2
5. Set #define **START_INCR_OOUTREG 20.0** in **const_cust_x.h**
6. In order to disable speed regulator, temporarily change the software by clearing speed regulator parameters:

   ```
   #define SPEED_PIREG_P_GAIN   0 /* 64 */
   #define SPEED_PIREG_I_GAIN 0
   ```

   in **const_cust_x.h** file
7. Build and run the code (see **Software Execution**, **Build**, and **Execute from Evaluation Board**)
8. Start the PC master software tuning project
9. Select Start Parameters Tuning subproject (see **Software Parameters Tuning with PC Master Software Project File**) in order to be able to modify the start parameters
10. You can see the actual zero-crossing (commutation) period on the Start Parameters Tuning\New Scope, or measure the phase a, b, and c voltages on a real oscilloscope

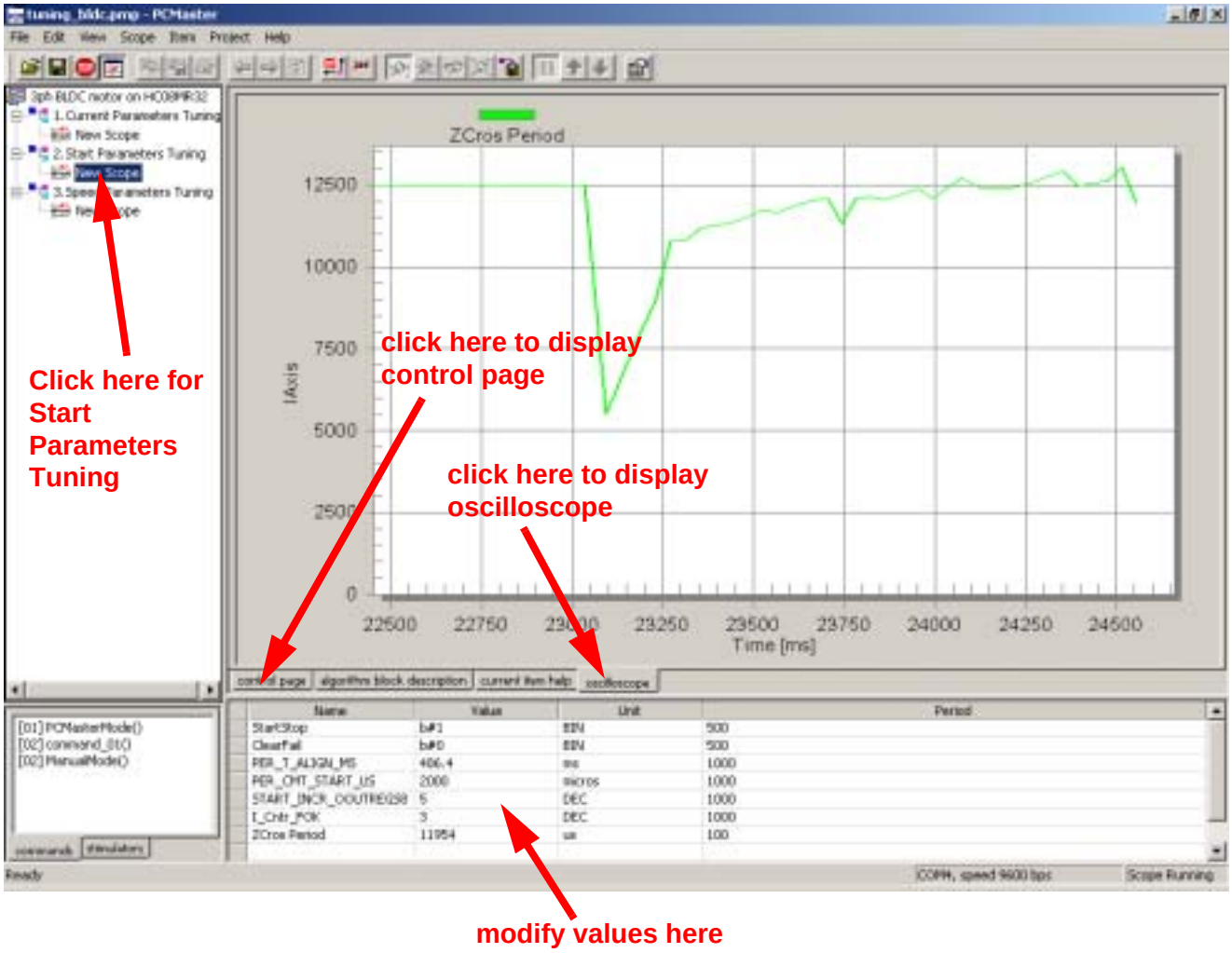

Figure 16. PC Master Software Start Parameters Tuning Window

11. Set PC master software control mode (see **Application Control** and **PC Master Software (Remote) Operating Mode**)
12. Start motor (see **Application Control** and **PC Master Software (Remote) Operating Mode**)
13. If the software signalizes errors (usually commutation error), clear the errors, stop the motor, and change **PER_CMT_START_US** (increase or decrease!) by PC master software
14. Repeat step **13.** until the motor starts well. If the motor starts against a high start-up torque, or if Alignment state current is low, it is recommended to change **START_INCR_OOUTREG** by PC master software. (If it is a problem to start the motor, then **I_CNTR_FOK** can also be changed from default 0x03, but not recommended!)

15. If the motor starts and continues running, after you repeatedly start/stop, the start-up parameters are set properly
16. Open **const_cust_x.h** and modify parameters with the final variable values **PER_CMT_START_US, START_INCR_OOUTREG**, evaluated with PC master software.
17. Change the software back to normal, set speed regulator parameters to:
    #define SPEED_PIREG_P_GAIN 64
    #define SPEED_PIREG_I_GAIN 0
    in **const_cust_x.h** file to enable speed regulation
18. Build the code (see **Software Execution** and **Build**)

Software Customizing to Motor — Speed Control Setting

When the motor commutation setting is done, the speed control parameters need to be set.

For settings which must be done, follow the labels **MUST_CHANGE_nn**, **MUST_CHANGE_EXPER_nn** in file ***const_cust_x.h***.

For changes which can be done (but usually are not necessary), follow the labels **CAN_CHANGE_nn**, **CAN_CHANGE_EXPER_nn** in file ***const_cust_x.h***

Number of commutations per motor revolution:

```
/* MUST_CHANGE_9: */
#define COMMUT_REV   18.0
```

Range: <0,255>

COMMUT_REV period must be changed for any motor accommodation. Set the number of commutations according to the number of rotor poles (there are six commutations for one electrical angle revolution). Therefore:

$$COMMUT\_REV = \frac{6 * motor\_poles}{2}$$

Equation 9

Maximal speed range [rpm]:

```
/* MUST_CHANGE_10: */
#define SPEED_RANGE_MAX_RPM 3000.0
```

Range: <0,infinity>

Determines scaling of speed variables. **SPEED_RANGE_MAX_RPM** must be changed for any motor accommodation as the software calculates the internal speed variables using this constant. For proper speed control it is important to set **SPEED_RANGE_MAX_RPM** higher then maximal actual motor speed (even during speed transient).

Maximal speed of the drive [rpm]:

```
/* MUST_CHANGE_11: */
#define SPEED_MAX_RPM 2500.0
```

Range: <0,SPEED_RANGE_MAX_RPM>

SPEED_MAX_RPM determines the maximal desired speed. It must be changed for any motor accommodation.

The software calculates the internal speed variables using **SPEED_RANGE_MAX_RPM** constant. For proper speed control it is important that **SPEED_MAX_RPM** and **SPEED_RANGE_MAX_RPM** constants relation must be set according to the following equation:

$$SPEED\_MAX\_RPM < SPEED\_RANGE\_MAX\_RPM \quad \text{Equation 10}$$

Minimal speed of the drive [rpm]:

```
/* MUST_CHANGE_12_EXPER: */
#define SPEED_MIN_RPM 500.0
```

Range: <0,SPEED_RANGE_MAX_RPM>

SPEED_MIN_RPM determines the minimal desired speed. It must be changed for any motor accommodation. The minimal speed is also determined by the back-EMF zero crossing technique. Usually:

$$SPEED\_MIN\_RPM = (0.07 \text{ to } 0.5)SPEED\_MAX\_RPM \quad \text{Equation 11}$$

Therefore, for low speed requirements minimal speed SPEED_MIN_RPM must be evaluated experimentally.

NOTE: ***SPEED_MIN_RPM*** *can be evaluated with PC master software tuning file tuning_bldc.pmp.*

Minimal PWM Duty cycle limit [-]:

```
/* CAN_CHANGE_11: */
#define DUTY_PWM_MIN 0.250
```

Range: <0,1>

DUTY_PWM_MIN determines minimal PWM duty cycle limit, and in this way it restricts minimal voltage on the motor.

Therefore, **DUTY_PWM_MIN** must be changed if its default setting creates a higher voltage than is physically necessary to run with a speed close to **SPEED_MIN_RPM**.

CAUTION: *If the motor is unable to run down to the speed set in* **SPEED_MIN_RPM***, then decrease* **DUTY_PWM_MIN** *constant.*

Speed PI regulator constants:

```
/* MUST_CHANGE_13_EXPER: */
#define SPEED_PIREG_P_GAINSCALELEFT 0
```

Range: <0,8>

```
/* MUST_CHANGE_14_EXPER: */
#define SPEED_PIREG_P_GAIN  128
```

Range: <0,255>

where the current regulator proportional gain is:

$$KP = SPEED\_PIREG\_P\_GAIN * 2^{SPEED\_PIREG\_P\_GAINSCALELEFT} \quad \text{Equation 12}$$

```
/* MUST_CHANGE_15_EXPER: */
#define SPEED_PIREG_I_GAINSCALERIGHT 0
```

Range: <0,8>

```
/* MUST_CHANGE_16_EXPER: */
#define SPEED_PIREG_I_GAIN  64
```

Range: <0,255>

where the current regulator integral gain is:

$$KI = SPEED\_PIREG\_I\_GAIN * 2^{(-SPEED\_PIREG\_I\_GAINSCALERIGHT)} \quad \text{Equation 13}$$

These constants can be calculated according to regulators theory. The speed sampling (regulator execution) period is **PER_T3_RUN_US** = 2.560 ms at default software setting. Another recommended solution is experimental setting.

NOTE: *SPEED_PIREG_P_GAINSCALELEFT, SPEED_PIREG_P_GAIN, SPEED_PIREG_I_GAINSCALERIGHT, SPEED_PIREG_I_GAIN can be evaluated with PC master software tuning file tuning_bldc.pmp.*

We suggest using PC master software with tuning file tuning_bldc.pmp for regulator parameters evaluation. You can use this procedure:

1. Ensure that the start-up and commutation parameters were set properly (**Start-up Constants and Maximal Commutation Period**) in **const_cust_x.h**
2. Set **const_cust_x.h**:
   SPEED_PIREG_P_GAINSCALELEFT 0
   SPEED_PIREG_P_GAIN 0
   SPEED_PIREG_I_GAINSCALERIGHT 7
   SPEED_PIREG_I_GAIN 0
3. Ensure that **COMMUT_REV**, **SPEED_RANGE_MAX_RPM**, **SPEED_MAX_RPM** are set properly in **const_cust_x.h**
4. Set **SPEED_MIN_RPM** as required (should be **SPEED_MIN_RPM** > **SPEED_MAX_RPM/5** for reliable commutation at low speed)

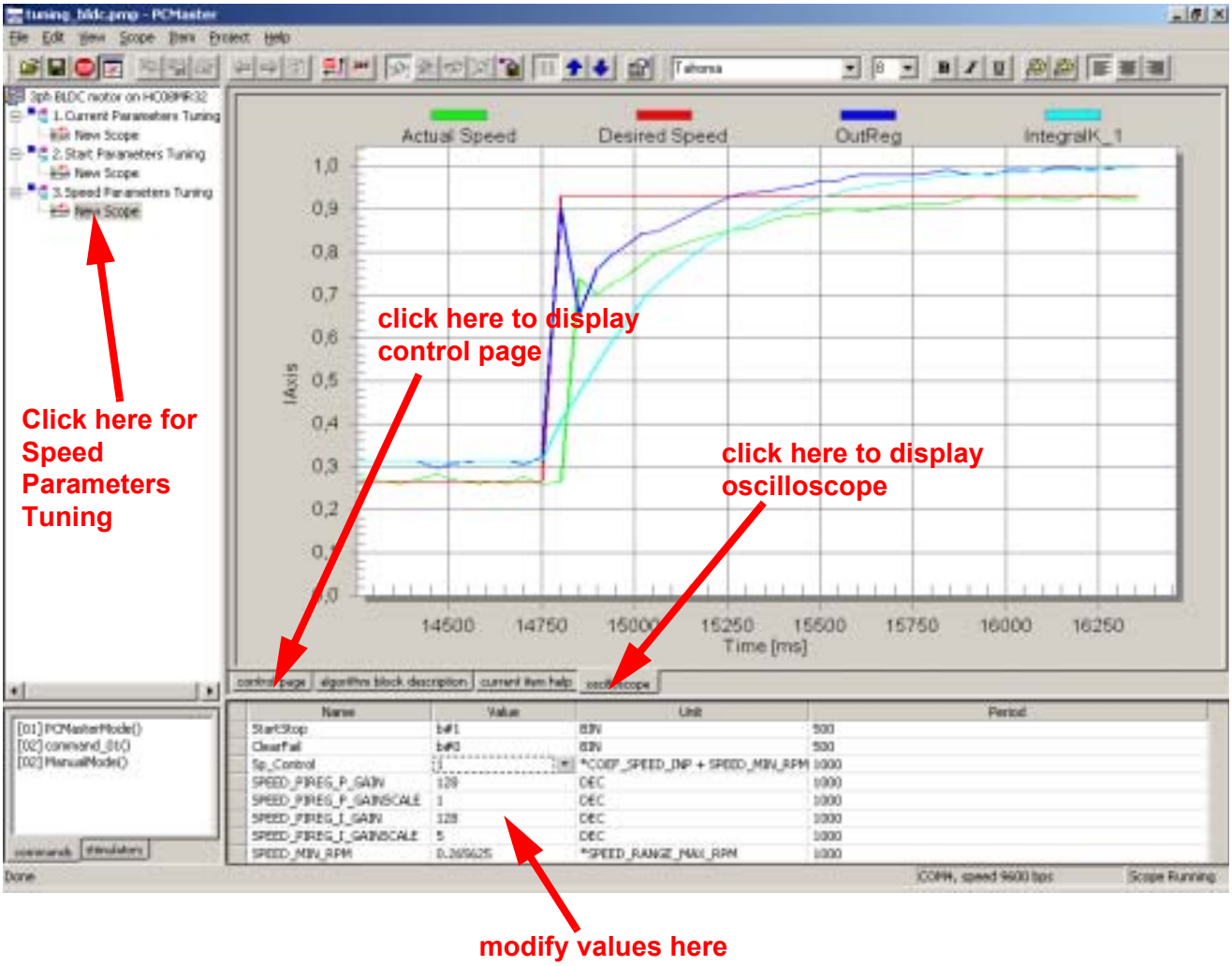

Figure 17. PC Master Software Speed Parameters Tuning Window

5. Build and run the code (see **Software Execution**, **Build**, and **Execute from Evaluation Board**)
6. Start the PC master software tuning project
7. Select Speed Parameters Tuning subproject (see **Software Parameters Tuning with PC Master Software Project File**) in order to be able to modify the current regulator
8. You can see the actual speed (and desired speed) on the Speed Parameters Tuning\New Scope, or measure the phase voltage period on real oscilloscope
9. Set PC master software control mode and start motor (see **Application Control** and **PC Master Software (Remote) Operating Mode**)
10. Set the speed to the middle of minimal and maximal speed

11. Increase, step by step, the proportional gain SPEED_PIREG_P_GAIN with PC master software, until speed noise or oscillation appears, or up to 128
12. If SPEED_PIREG_P_GAIN is set to 128, increase proportional gains in SPEED_PIREG_P_GAINSCALELEFT further with PC master software, steps 0, 1, 2 ... 8 otherwise leave SPEED_PIREG_P_GAINSCALELEFT as 0
13. Increase, step by step, the integral gain SPEED_PIREG_I_GAIN with PC master software, up to current oscillation or noise, or up to 128
14. If SPEED_PIREG_I_GAIN is set to 128, do further integral gain increases to SPEED_PIREG_I_GAINSCALELEFT with PC master software, steps 6, 5, 4...0; otherwise, leave SPEED_PIREG_I_GAINSCALELEFT as 7
15. You can further evaluate the setting of the regulator parameters in order to get a smoother current waveform until the regulation seems to be performing well
16. Open **const_cust_x.h** and modify the regulator parameters with the final variable values evaluated with PC master software
17. Start motor (see **Application Control** and **PC Master Software (Remote) Operating Mode**)
18. Set minimal desired speed
19. If the displayed real speed is not able to go down to the desired minimal speed, it is necessary to decrease minimal PWM duty cycle DUTY_PWM_MIN in **const_cust_x.h**!
20. Then, you can tune dynamic characteristics of speed regulators (steps **18.** to **29.**) or finish tuning the regulators
21. Start motor (see **Application Control** and **PC Master Software (Remote) Operating Mode**)
22. Set minimal speed
23. Set maximal speed and observe the speed transient
24. Set minimal speed and observe the speed transient
25. Then, modify the regulator parameters with PC master software as in steps **11.** to **14.**
26. Change **SPEED_MIN_RPM** if problems occur at low speed
27. Repeat steps **21.** to **25.** until regulation is improved
28. Open **const_cust_x.h** and modify the regulator parameters with the final variable values evaluated with PC master software
29. Build the code (see **Software Execution** and **Build**)

Most important software settings are described in previous sections, but for some applications, PWM frequency must be modified. It is described in **PWM Frequency and Current Sampling Period Setting**.

Once you set the speed control and the motor is running in all start, speed up, and slow down conditions, the software parameters are set for the motor. Remember that all parameters are set in **const_cust_x.h**. Then, it is possible to program the FLASH memory of the MC68HC908MR32 device.

PWM Frequency and Current Sampling Period Setting

PWM frequency and current sampling period settings are not usually needed. The PWM frequency also affects the current sampling period. Consequently, the current regulation setting should be done, while understanding their mutual dependency. Therefore, the PWM frequency setting is provided in the file **const.h**, instead of **const_cust_x.h**.

PWM Frequency

For the PWM frequency setting, follow the label **CAN_CHANGE_FPWM_n** in **const.h** file.

The PWM frequency setting is provided by:

```
/* CAN_CHANGE_FPWM_1: */
#define SET_PER_PWM     32.0
```

Range: <1,255>

The final PWM period is defined by setting **SET_PER_PWM**.

The PWM period [μs] is:

$$\text{PWM period} = \text{PERIOD\_PWM\_US} = \text{SET\_PER\_PWM}*2 \qquad \text{Equation 14}$$

With default software setting (oscillator clock, etc.).

The final PWM frequency [Hz] is:

$$\text{PWMfrequency} = \frac{10^6}{2\text{SET\_PER\_PWM}} \qquad \text{Equation 15}$$

With default software setting.

Settings for some important PWM frequencies are listed in **Table 7**.

Table 7. PWM Frequency Setting

SET_PER_PWM	PWM Frequency (FREQUENCY_PWM)	PWM Period (PERIOD_PWM_US)
16.0	31.250 Hz	32 μs
25.0	20.000 Hz	50 μs
32.0 (default)	15.625 Hz	64 μs
128.0	3.90625 Hz	256 μs

CAUTION: *Current measurement sampling period is synchronized with PWM. Therefore, changing PWM frequency automatically changes the current sampling period. This, apart from other things, effects the current regulator. Therefore, after changing PWM frequency, changing (or checking) current sampling period is strongly recommended.*

Current Sampling Period

Current sampling period should usually be changed in two cases:

1. When PWM frequency is changed
2. Motors with externally low electrical constant

If the motor electrical constant is lower than default current sampling period of 128 μs, the current regulator may not work properly.

For current sampling period setting follow the label **CAN_CHANGE_PERCURSAMP_n** in **const.h** file.

Current sampling period setting is provided by:

```
/* CAN_CHANGE_FPWM_n: */
/* CAN_CHANGE_PERCURSAMP_n: */
#define SET_PER_CS      2.0
```

Range: <1,->

The final current sampling period [μs] is:

$$\text{Current sampling period} = \text{PWM period} * \text{SET\_PER\_CS} \qquad \text{Equation 16}$$

current sampling period = PWM period * **SET_PER_CS** [μs]

PER_CS_T1_US = **PERIOD_PWM_US** * **SET_PER_CS** [μs]

Current Sampling Instant

Time period from a PWM reload event (middle of central aligned PWM) to current sampling (time shift of A/D conversion with PWM) [μs]:

```
/* CAN_CHANGE_PERCURSAMP_n: */
#define PER_PWM_CS_US   5.0
```

Range: <-PERIOD_PWM_US/2,PERIOD_PWM_US/2>

Usually it is not recommended to change **PER_PWM_CS_US**, but it can be evaluated when there are problems with back-EMF zero crossing noise.

It is necessary to set **SET_PER_CS** according to the following equation:

$$-\frac{\text{PERIOD\_PWM\_US}}{2} < \text{PER\_PWM\_CS\_US} < \frac{\text{PERIOD\_PWM\_US}}{2} \qquad \text{Equation 17}$$

Conclusion

If all the points in **Introduction to Software Parameters Setting and Tuning for Customer Motor** are done, the software should be customized to customer motor.

If the software customizing of your motor was not successful, it is recommended that you read **Software Suitability Guide for Customer Application and Motor**, since the software may not be suitable for some applications. Some important recommendations can also be found under the ***Caution*** and ***Note*** labels in this Application Note.

References

1. *Sensorless BLDC Motor Control on MC68HC908MR32 — Software Description* (Freescale document order number AN2355/D)
2. Motion Control Development Tools found on the World Wide Web at:
   http://e-www.freescale.com
3. *Freescale Embedded Motion Control MC68HC908MR32 Control Board User's Manual*, (document order number MEMCMR32CBUM/D), Freescale 2000
4. Freescale *Embedded Motion Control 3-Phase AC BLDC High-Voltage Power Stage User's Manual* (document order number MEMC3PBLDCPSUM/D), Freescale 2000
5. Freescale *Embedded Motion Control Optoisolation Board* (document order number MEMCOBUM/D), Freescale 2000
6. Freescale *Embedded Motion Control Evaluation Motor Board User's Manual* (document order number MEMCEVMBUM/D), Freescale 200
7. Freescale *Embedded Motion Control 3-Phase BLDC Low-Voltage Power Stage User's Manual* (document order number MEMC3PBLDCLVUM/D), Freescale 2000
8. User's Manual for PC Master Software, Freescale 2000, found on the World Wide Web at:
   http://e-www.freescale.com
9. *68HC908MR32, 68HC908MR16 Advance Information* (document order number MC68HC908MR32/D), Freescale
10. *Low Cost High Efficiency Sensorless Drive for Brushless DC Motor using MC68HC(7)05MC4* (document order number AN1627), Freescale

How to Reach Us:

Home Page:
www.freescale.com

E-mail:
support@freescale.com

USA/Europe or Locations Not Listed:
Freescale Semiconductor
Technical Information Center, CH370
1300 N. Alma School Road
Chandler, Arizona 85224
+1-800-521-6274 or +1-480-768-2130
support@freescale.com

Europe, Middle East, and Africa:
Freescale Halbleiter Deutschland GmbH
Technical Information Center
Schatzbogen 7
81829 Muenchen, Germany
+44 1296 380 456 (English)
+46 8 52200080 (English)
+49 89 92103 559 (German)
+33 1 69 35 48 48 (French)
support@freescale.com

Japan:
Freescale Semiconductor Japan Ltd.
Headquarters
ARCO Tower 15F
1-8-1, Shimo-Meguro, Meguro-ku,
Tokyo 153-0064
Japan
0120 191014 or +81 3 5437 9125
support.japan@freescale.com

Asia/Pacific:
Freescale Semiconductor Hong Kong Ltd.
Technical Information Center
2 Dai King Street
Tai Po Industrial Estate
Tai Po, N.T., Hong Kong
+800 2666 8080
support.asia@freescale.com

For Literature Requests Only:
Freescale Semiconductor Literature Distribution Center
P.O. Box 5405
Denver, Colorado 80217
1-800-441-2447 or 303-675-2140
Fax: 303-675-2150
LDCForFreescaleSemiconductor@hibbertgroup.com

freescale™
semiconductor

AN2356/D
Rev. 0
11/2002