Motor Control Firmware

UG0692 User Guide

Microsemi
Power Matters.™

Table of Contents

1. Introduction

The motor control firmware resides in the microcontroller subsystem (MSS). The MSS communicates with the field programmable gate array (FPGA) fabric through the AMBA® APB bus, and exposes a USB-HID interface to communicate with SF2 Dual Axis Motor Control GUI on a host PC.

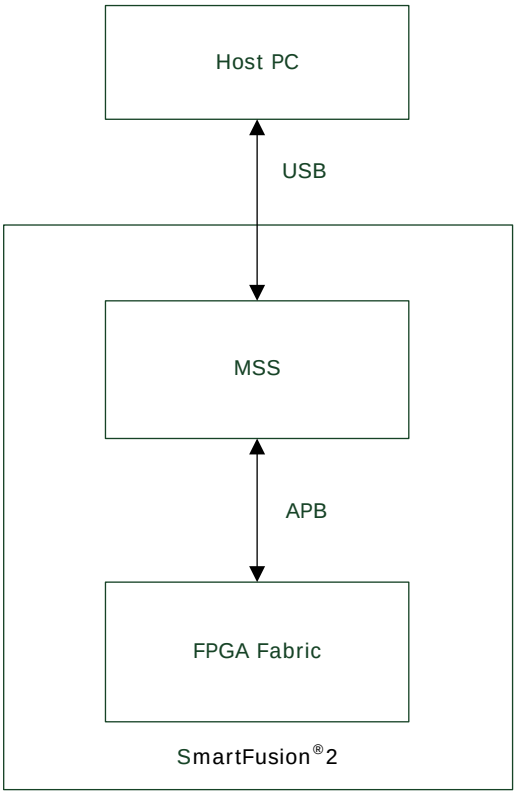

Figure 1 · MSS as an Interface Between External GUI and FPGA Fabric

The firmware plays two major roles in the motor control system – it initializes the FPGA fabric at reset, and acts as an interface between a GUI running on an external host and the FPGA fabric.

Table 1 lists the files present in the motor control firmware. The type definitions are discussed in detail in *Introduction*. The APIs in USBMsgHandler, MCCalc, and the MCFPGAIF modules are described in detail in the following sections.

Table 1. List of Files Present in the Motor Control Firmware

File	Function
main.c	Contains the main function.
USBMsgHandler.h, USBMsgHanlder.c	Declares and defines APIs to handle incoming USB messages.
MCCalc.h, MCCalc.c	Declares and defines APIs to initialize control parameters and compute derived parameters.
MCFPGAIF.h, MCFPGAIF.c	Declares and derives APIs to communicate with FPGA for various control functions.
MC_System.h	Contains APB addresses of various FPGA registers that can

File	Function
	be read or written.
config.h	Contains default configuration parameters.
TypeDef.h	Contains structures to hold motor or system data.
usb_user_descriptors.c	Contains USB HID parameters such as product ID and vendor ID.

2. Type Definitions

This section describes the type definitions that define the various motor parameters and the field oriented control (FOC) parameters. These type definitions are declared in `Typedef.h`.

2.1 MC_BLDC_FOC_type1

Table 2 describes the type definition used in the `MC_BLDC_FOC_type1` type definition.

Table 2. Specification of MC_BLDC_FOC_type1

Name	MC_BLDC_FOC_type1	

Type	typedef struct MC_BLDC_FOC_type1 mc_bldc_foc_type1;
	struct MC_BLDC_FOC_type1{
	int32_t speed_ramp;
	int32_t speed_ref;
	uint32_t speed_kp_gain;
	uint32_t speed_ki_gain;
	uint32_t i_kp_gain;
	uint32_t i_ki_gain;
	uint32_t angle_kp_gain;
	uint32_t angle_ki_gain;
	uint16_t open_loop_voltage;
	uint16_t open_loop_current;
	uint16_t closed_loop_speed;
	int8_t direction;
	int8_t run_status;
	uint8_t seq_controller_config;
	uint16_t encoder_resolution;
	mc_graph_type mc_graph;
	};	

Description	int32_t speed_ramp;	Speed ramp value (in RPM/s)
	int32_t speed_ref;	Reference speed input
	uint32_t speed_kp_gain;	Speed Kp gain
	uint32_t speed_ki_gain;	Speed Ki gain
	uint32_t i_kp_gain	Current Kp gain
	uint32_t i_ki_gain	Current Ki gain
	uint32_t angle_kp_gain;	Angle PI Kp gain
	uint32_t angle_ki_gain;	Angle PI Ki gain
	uint16_t open_loop_voltage;	Open loop voltages in per unit
	uint16_t open_loop_current;	Open loop motor current in per unit
	uint16_t closed_loop_speed;	Closed loop threshold speed in per unit

	int8_t direction;	Motor direction
	int8_t run_status	Motor running status
	uint8_t seq_controller_config	Sequence controller configuration
	uint16_t encoder_resolution	Encoder resolution
	mc_graph_type mc_graph	Structure for graph variables

2.2 bldc_motor_type

Table 3 describes the type definition used in bldc_motor_type.

Table 3. Specification of bldc_motor_type

Name	bldc_motor_type	

Type	typedef struct bldc_motor1 bldc_motor_type;
	struct bldc_motor1{
	uint32_t speed_RPM;
	uint32_t Npp;
	uint32_t Rs_mohm;
	uint32_t Ls_uhenry;
	uint32_t switching_freq_kHz;
	uint16_t dc_voltage_mV;
	uint16_t current_mA;
	};	

Description	int32_t speed_RPM;	Rated motor speed (in RPM)
	int32_t Npp;	Number of pole pairs
	uint32_t Rs_mohm;	Phase resistance (in mohm)
	uint32_t Ls_uhenry;	Phase inductance (in uhenry)
	uint32_t switching_freq_kHz	Switching frequency (in kHz)
	uint16_t dc_voltage_mV	DC Voltage (in mV)
	uint16_t current_mA;	Motor current (in mA)

2.3 stepper_type1

Table 4 describes the type definition used in stepper_type1.

Table 4. Specification of stepper_type1

Name	stepper_type1	
Type	typedef struct stepper_type1 stepper_type; struct stepper_type1{ int8_t mode; int8_t direction; int8_t run_status; int16_t microstep_res; int16_t speed_rpm; int16_t step_num; int32_t cmd_steps; int32_t i_ref; uint32_t i_kp_gain; uint32_t i_ki_gain; uint8_t seq_controller_config; mc_graph_type mc_graph; } ;	
Description	int8_t mode	Mode (Position/Speed)
	int8_t direction	Direction
	int8_t run_status	Running status
	int16_t microstep_res	Microstep resolution
	int16_t speed_rpm	Speed (in RPM)
	int16_t step_num	Step number of the motor
	int32_t cmd_steps	Command number of steps in position mode
	int32_t i_ref	Current reference
	uint32_t i_kp_gain	Current Kp gain
	uint32_t i_ki_gain	Current Ki gain
	uint8_t seq_controller_config	Sequence controller configuration
	mc_graph_type mc_graph	Structure for graph variables

2.4 mc_graph_type1

Table 5 describes the type definition used in mc_graph_type1.

Table 5. Specification of mc_graph_type1

Name	mc_graph_type	

Type	typedef struct mc_graph_type1 mc_graph_type;
	struct mc_graph_type1{
	int32_t variable1;
	int32_t variable2;
	int32_t variable3;
	int32_t variable4;
	int32_t rpm;
	int32_t current;
	
	};	

Description	int32_t variable1	Variable 1 to be plotted
	int32_t variable2	Variable 2 to be plotted
	int32_t variable3	Variable 3 to be plotted
	int32_t variable4	Variable 4 to be plotted
	int32_t rpm	Motor speed (in RPM)
	int32_t current	Motor current (in mA)

3. MCCalc Module

This section describes the various functions available in the MCCalc module. These functions initialize and calculate various constants in the system.

3.1 MC_StepperInit

This function initializes the values in stepper_type1 with default values from the `config.h` file. Table 6 lists the specifications of this function.

Table 6. Specification of API MC_StepperInit

Syntax	void MC_StepperInit()
Re-Entrancy	Non re-entrant
Parameters (Inputs)	None
Parameters (Output)	None
Return	None

3.2 MC_StepperConstCal

This function calculates stepper constants that are derived from other constant inputs. Table 7 lists the specifications of this function.

Table 7. Specification of API MC_StepperConstCal

Syntax	void MC_StepperConstCal()
Re-Entrancy	Non Re-entrant
Parameters (Inputs)	None
Parameters (Output)	None
Return	None

3.3 MC_BLDCInit

This function initializes the values in MC_BLDC_FOC_type1 and bldc_motor_type with default values from config.h. Table 8 lists the specifications of this function.

Table 8. Specification of API MC_BLDCInit

Syntax	void MC_BLDCInit()
Re-Entrancy	Non Re-entrant
Parameters (Inputs)	None
Parameters (Output)	None
Return	None

3.4 MC_BLDCConstCal

This function calculates BLDC constants that are derived from other constant inputs. Table 9 lists the specifications of this function.

Table 9. Specification of API MC_BLDCConstCal

Syntax	void BLDCConstCal()
Re-Entrancy	Non Re-entrant
Parameters (Inputs)	None
Parameters (Output)	None
Return	None

4. MCFPGAIF Module

This section describes the various functions available in the MCFPGAIF module. These functions provide interface to the FPGA fabric through the APB.

4.1 MotorControl_Init

This function initializes the system, and calculates brushless DC (BLDC) and Stepper parameters by calling functions from MCCalc module. This function passes the values of these two parameters to the FPGA fabric by calling the MC_BLDCSetFabReg and MC_StepperSetFabReg functions. Table 10 lists the specifications of this function.

Table 10. Specification of API MotorControl_Init

Syntax	void MotorControl_Init()
Re-Entrancy	Non re-entrant
Parameters (Inputs)	None
Parameters (Output)	None
Return	None
Algorithm	Call MC_BLDCInit from MCCalc Call MC_BLDCConstCal from MCCalc Call MC_StepperInit from MCCalc Call MC_StepperConstCal from MCCalc Call MC_BLDCSetFabReg from MCFPGAIF Call MC_StepperSetFabReg from MCFPGAIF

4.2 MotorControl_Start

This function starts the motor by setting appropriate value in the C_STOP_MOTOR_ADDR and C_START_MOTOR_ADDR addresses . This function changes the global variable g_base_address to BASE_ADDR_0 (BLDC) or BASE_ADDR_1 (Stepper) for motor axis. Table 11 lists the specifications of this function.

Table 11. Specification of API MotorControl_Start

Syntax	void MotorControl_Start ()
Re-Entrancy	Non Re-entrant
Parameters (Inputs)	None
Parameters (Output)	None
Return	None
Algorithm	Write zero into the C_STOP_MOTOR_ADDR Wait for 3 clock cycles Write one into the C_START_MOTOR_ADDR

4.3 MotorControl_Stop

This function stops the motor by setting appropriate values in the C_STOP_MOTOR_ADDR and C_START_MOTOR_ADDR addresses. This function changes the global variable g_base_address to BASE_ADDR_0 (BLDC) or BASE_ADDR_1 (Stepper) for motor axis. Table 12 lists the specifications of this function.

Table 12. Specification of API MotorControl_Stop

Syntax	void MotorControl_Stop()
Re-Entrancy	Non Re-entrant
Parameters (Inputs)	None
Parameters (Output)	None
Return	None
Algorithm	Write zero into the C_START_MOTOR_ADDR Wait for 3 clock cycles Write one into the C_STOP_MOTOR_ADDR

4.4 MC_WClearFault

This function clears a fault detected in the Sequence controller IP in FPGA fabric by setting appropriate values in the C_CLR_FAULT_ADDR address. Table 13 lists the specifications of this function.

Table 13. Specification of API MC_WClearFault

Syntax	void MC_WClearFault()
Re-Entrancy	Non Re-entrant
Parameters (Inputs)	None
Parameters (Output)	None
Return	None
Algorithm	Write one into the C_CLR_FAULT_ADDR Wait for 3 clock cycles Write zero into the C_CLR_FAULT_ADDR

4.5 MC_GetStatus

This function gets the sequence controller FSM status from FPGA fabric. This value provides the system status, and also enables fault detection. Table 14 lists the specifications of this function.

Table 14. Specification of API MC_GetStatus

Syntax	uint8_t MC_GetStatus()
Re-Entrancy	Non Re-entrant
Parameters (Inputs)	None
Parameters (Output)	uint8_t status
Return	None

Algorithm	Return FSM state read from Sequence controller

4.6 MC_BLDCSetDirn

This function assigns motor direction to the BLDC motor by setting appropriate values in the C_DIRECTION_CONFIG_ADDR address . Table 15 lists the specifications of this function.

Table 15. Specification of API MC_BLDCSetDirn

Syntax	void MC_BLDCSetDirn(uint32_t dirn)
Re-Entrancy	Non Re-entrant
Parameters (Inputs)	uint32_t dirn
Parameters (Output)	None
Return	None
Algorithm	Write dirn to C_DIRECTION_CONFIG_ADDR

4.7 MC_BLDCGetSpdI

This function gets motor speed and motor current, and converts these values from per unit to RPM and milliamperes. This function also displays a smoothened waveform on the GUI by filtering these values. Table 16 lists the specifications of this function.

Table 16. Specification of API MC_BLDCGetSpdI

Syntax	void MC_BLDCGetSpdI()
Re-Entrancy	Non Re-entrant
Parameters (Inputs)	None
Parameters (Output)	None
Return	None
Algorithm	Get speed from FPGA fabric Convert values from per unit to RPM Filter the speed Get current from FPGA fabric Convert values from per unit to milliamperes Filter the current

4.8 MC_BLDCSetFabReg

This function sets values in various FPGA fabric registers through the APB. This function also sets all the values related to BLDC operation. Table 17 lists the specifications of this function.

Table 17. Specification of API MC_BLDCSetFabReg

Syntax	void MC_BLDCSetFabReg()
Re-Entrancy	Non Re-entrant
Parameters (Inputs)	None
Parameters	None

(Output)	
Return	None
Algorithm	Write values into various FPGA fabric registers (corresponding to BLDC motor)

4.9 MC_StepperSetDirn

This function sets the motor direction by computing motor direction using command_steps and microstep_resolution. This function assigns the motor direction value to the C_THETA_GEN_CMD_STEP_NO_ADDR. Table 18 lists the specifications of this function.

Table 18. Specification of API MC_StepperSetDirn

Syntax	void MC_StepperSetDirn(uint32_t dirn)
Re-Entrancy	Non Re-entrant
Parameters (Inputs)	uint32_t dirn
Parameters (Output)	None
Return	None
Algorithm	Compute command_steps*microstep_resolution*dirn Write computed value into C_THETA_GEN_CMD_STEP_NO_ADDR

4.10 MC_StepperGetI

This function gets the motor current, and converts it from per unit to milliamperes. Table 19 lists the specifications of this function.

Table 19. Specification of API MC_StepperGetI

Syntax	void MC_StepperGetI()
Re-Entrancy	Non Re-entrant
Parameters (Inputs)	None
Parameters (Output)	None
Return	None
Algorithm	Obtain motor current from FPGA fabric Convert value from per unit to milliamperes.

4.11 BLDC_SetSpeedSRamp

This function sets the motor speed and speed ramp parameters after scaling these parameter values to per unit. Table 20 lists the specifications of this function.

Table 20 Specification of API BLDC_SetSpeedSRamp

Syntax	void BLDC_SetSpeedSRamp()
Re-Entrancy	Non Re-entrant
Parameters (Inputs)	None

Parameters (Output)	None
Return	None
Algorithm	Scale speed reference value from RPM to per unit. Write value into C_RATE_LIMIT_REF_ADDR Scale speed ramp value from RPM/s to per unit. Write value into C_RATE_LIMIT_RATE_CNT_ADDR

4.12 MC_StepperSetFabReg

This function writes parameters to FPGA fabric addresses corresponding to the stepper motor through the APB. Table 21 lists the specifications of this function.

Table 21. Specification of API MC_StepperSetFabReg

Syntax	void MC_StepperSetFabReg ()
Re-Entrancy	Non Re-entrant
Parameters (Inputs)	None
Parameters (Output)	None
Return	None
Algorithm	Write values into various FPGA fabric registers (corresponding to Stepper motor)

4.13 abs_function

This function gets absolute value of the input parameter. Table 22 lists the specifications of this function.

Table 22. Specification of API abs_function

Syntax	void abs_function (int32_t data)
Re-Entrancy	Non Re-entrant
Parameters (Inputs)	int32_t data
Parameters (Output)	None
Return	None
Algorithm	Determine the absolute value of input parameter data

5. USBMsgHandler Module

This section describes the various functions available in the USBMsgHandler module. These functions initialize and calculate various constants in the system.

5.1 USB_init

This function initializes the USB HID driver. Table 23 lists the specifications of this function.

Table 23. Specification of API USB_init

Syntax	void USB_init()
Re-Entrancy	Non re-entrant
Parameters (Inputs)	None
Parameters (Output)	None
Return	None

5.2 USBMsgHandler

This function acts as an interrupt handler for data received through the USB from GUI. Table 24 lists the specifications of this function.

Table 24. Specification of API USBMsgHandler

Syntax	void USBMsgHandler()
Re-Entrancy	Non Re-entrant
Parameters (Inputs)	None
Parameters (Output)	None
Return	None

command idSelect a particular GUI function using command ids. The first byte (byte[0]) of the USB packet received is used as the command id. In some cases, a response is sent to the GUI. Table 25 lists commands on Byte[0], and indicates if the command requires a response from the MSS.

Table 25. List of Commands and Functions

Byte[0]	Description	Response required by GUI
0x00	Start motor	No
0x01	Stop motor	No
0x02	Direction – Clockwise	No
0x03	Direction – Anticlockwise	No
0x04	Reset all parameters	No
0x05	Unused	
0x06	Get data (MSS to GUI)	Yes
0x07	Set data	No

Byte[0]	Description	Response required by GUI
0x08	Data plotting enabled	No
0x09	Test USB connection	Yes
0x0A	BLDC speed and current plot	Yes
0x0B	USB Loop back	Yes
0x0C	Unused	
0x0D	Clear fault (BLDC only)	Yes
0x0E	Set parameters in MSS	Yes
0x0F	Get parameters from MSS	Yes
0x10	Set BLDC parameters in MSS (BLDC only)	No
0x11	Get BLDC parameters from MSS (BLDC only)	Yes
0x12	Get fault status (BLDC only)	Yes

The second byte of the USB packet is used for axis selection. Table 26 lists axis selection on Byte[1].

Table 26. List of Values in Axis Selection (byte[1])

Byte[1]	Axis selection
0x00	BLDC motor
0x01	Stepper motor

List of Changes

The following table lists important changes made in this document for each revision.

Date and Revision	Change	Page
Revision 1 (March 2016)	Initial release.	NA

Product Support

Microsemi SoC Products Group backs its products with various support services, including Customer Service, Customer Technical Support Center, a website, electronic mail, and worldwide sales offices. This appendix contains information about contacting Microsemi SoC Products Group and using these support services.

Customer Service

Contact Customer Service for non-technical product support, such as product pricing, product upgrades, update information, order status, and authorization.

From North America, call **800.262.1060**
From the rest of the world, call **650.318.4460**
Fax, from anywhere in the world **408.643.6913**

Customer Technical Support Center

Microsemi SoC Products Group staffs its Customer Technical Support Center with highly skilled engineers who can help answer your hardware, software, and design questions about Microsemi SoC Products. The Customer Technical Support Center spends a great deal of time creating application notes, answers to common design cycle questions, documentation of known issues and various FAQs. So, before you contact us, please visit our online resources. It is very likely we have already answered your questions.

Technical Support

For Microsemi SoC Products Support, visit

http://www.microsemi.com/products/fpga-soc/design-support/fpga-soc-support

Website

You can browse a variety of technical and non-technical information on the Microsemi SoC Products Group home page, at http://www.microsemi.com/products/fpga-soc/fpga-and-soc.

Contacting the Customer Technical Support Center

Highly skilled engineers staff the Technical Support Center. The Technical Support Center can be contacted by email or through the Microsemi SoC Products Group website.

Email

You can communicate your technical questions to our email address and receive answers back by email, fax, or phone. Also, if you have design problems, you can email your design files to receive assistance. We constantly monitor the email account throughout the day. When sending your request to us, please be sure to include your full name, company name, and your contact information for efficient processing of your request.

The technical support email address is soc_tech@microsemi.com.

My Cases

Microsemi SoC Products Group customers may submit and track technical cases online by going to My Cases.

Outside the U.S.

Customers needing assistance outside the US time zones can either contact technical support via email (soc_tech@microsemi.com) or contact a local sales office. Visit About Us for sales office listings and corporate contacts.

ITAR Technical Support

For technical support on RH and RT FPGAs that are regulated by International Traffic in Arms Regulations (ITAR), contact us via soc_tech@microsemi.com. Alternatively, within My Cases, select **Yes** in the ITAR drop-down list. For a complete list of ITAR-regulated Microsemi FPGAs, visit the ITAR web page.

Microsemi Corporate Headquarters
One Enterprise, Aliso Viejo,
CA 92656 USA

Within the USA: +1 (800) 713-4113
Outside the USA: +1 (949) 380-6100
Sales: +1 (949) 380-6136
Fax: +1 (949) 215-4996

E-mail: sales.support@microsemi.com

Microsemi Corporation (MSCC) offers a comprehensive portfolio of semiconductor and system solutions for communications, defense & security, aerospace and industrial markets. Products include high-performance and radiation-hardened analog mixed-signal integrated circuits, FPGAs, SoCs and ASICs; power management products; timing and synchronization devices and precise time solutions, setting the world's standard for time; voice processing devices; RF solutions; discrete components; Enterprise Storage and Communication solutions, security technologies and scalable anti-tamper products; Ethernet Solutions; Power-over-Ethernet ICs and midspans; as well as custom design capabilities and services. Microsemi is headquartered in Aliso Viejo, Calif., and has approximately 4,800 employees globally. Learn more at **www.microsemi.com**.

50200692-1/03.16