FM3 Family

32-bit Microcontroller

Math and Motor Control Library Quick Start Guide

Doc. No. 002-04766 Rev. *A

Cypress Semiconductor
198 Champion Court
San Jose, CA 95134-1709
Phone (USA): 800.858.1810
Phone (Intnl): +1 408.943.2600
www.cypress.com

Contents

1. Demo System

This quick start guide mainly describes how to use the sample project which includes the library functions provided by the Math and Motor Control Library.

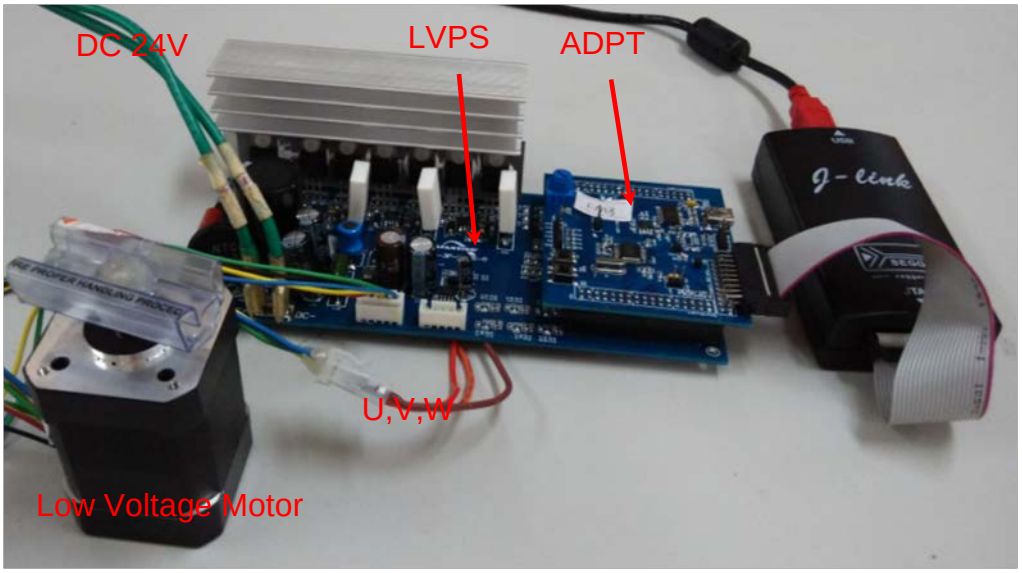
The MMCL is based on the Cypress Cortex-M3 microcontrollers, all of the modules in the MMCL are used in the sample project that will be described in detail in this guide. You can use this sample project to control the low voltage PMSM or BLDC.

This chapter introduces one example of inverter motor control project and help you run a motor quickly.

1.1 Demo System Introduction

The connection diagram for the MMCL sample project which is used to control the low voltage inverter motor is shown in Figure 1-1. The sample project can be adaptive to the any type of low voltage PMSM or BLDC motor.

Figure 1-1. Demo Connection

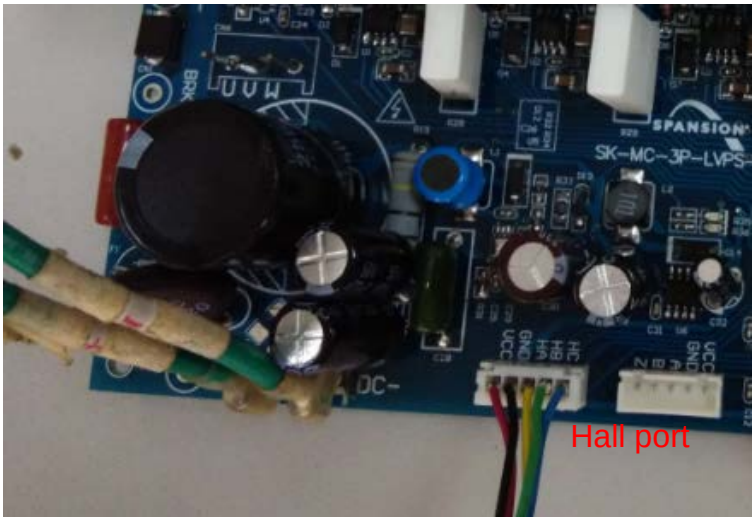


1.2 Hardware Connect

There are 4 lines to be connected to the demo board.

1. Plug the hall signal of motor to inverter board, shown as below.

Figure 1-2. Hall Signal Line Connection



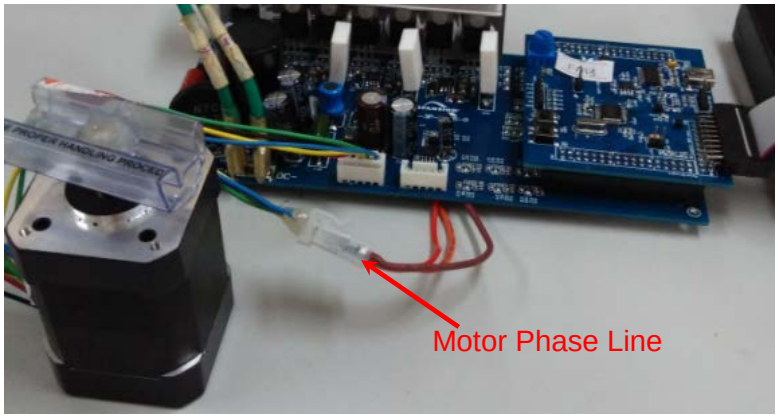
Note:

If there are only 2 hall signals on the motor, the hall A and B line can be only connected to the inverter's Hall A and Hall B port. Don't connect on the Hall C port on the board.

VCC and GND must be connected rightly, otherwise the hall won't work right and the motor will also not run.

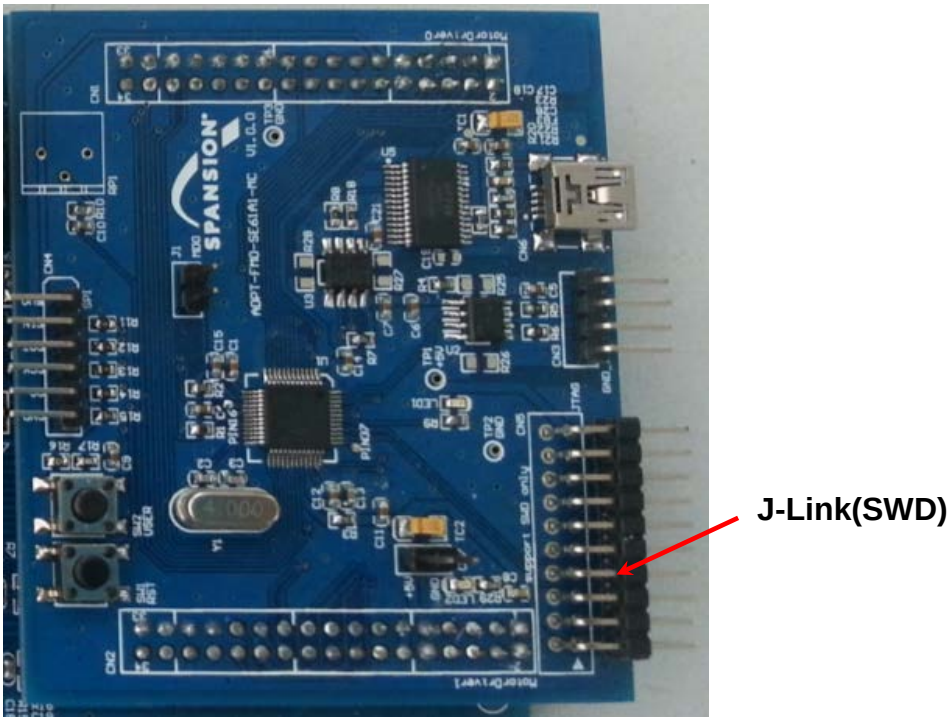
2. Plug the U, V, W phase lines to inverter board, shown as below.

Figure 1-3. Motor Line Connection



3. Connect JTAG to adapter, shown as below.

Figure 1-4. JTAG Line Connection

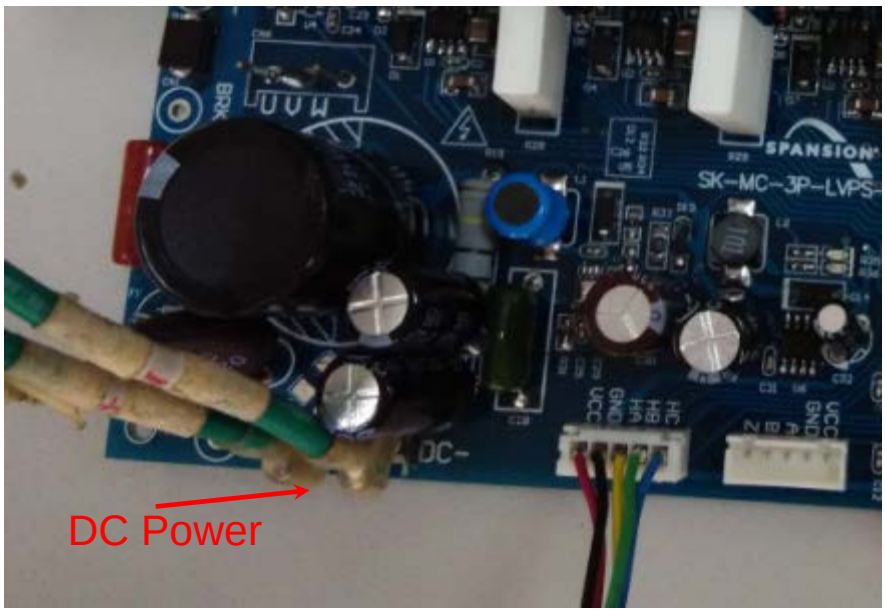


Note:

There is no isolator between the J-link and the HW, it is recommended to unplug the AC power and use the battery of your note book.

4. Connect DC power 24V to Inverter board, shown as below. The DC voltage can be from 24~48V according to the rated voltage of the motor and the current of the board must be lower than 2A.

Figure 1-5. AC Plug



1.3 Run the Motor

This section introduces how to the make the motor running as the setting parameter.

1.3.1 Motor Start/Stop

When the parameters for the firmware and hardware are correctly set, the motor can be started for the normal running. You can follow the steps to run to motor.

1). Check the basic motor and HW parameter setting in the user interfaces. If the setting is not matched with the real HW and motor parameter, there will be the unexpected running error in the motor running.

2). Compile project and download program to inverter board by the J-link.

① Set the j-link interface to the SWD mode: `Project →Option → J-Link/J-Trace → Connection →SWD`

② Click the button **A** that is shown in Figure 1-6 to connect the J-link and download the FW into the MCU,

③ Cick the button **B** to execute the sample project. You can enter the none-zero speed value to start the motor in the structure that is shown as **C** after the firmware is executed about 2s later.

For example, when the variable '`Motor_stcRunPar.i32TargetSpeedRpm= 1000'` by your online input, the drum speed of the washer will CCW run to 1000rpm.

And you can take the Table 1-1 for your detailed reference for the speed command.

Table 1-1. Drum Running Status by the Command Speed

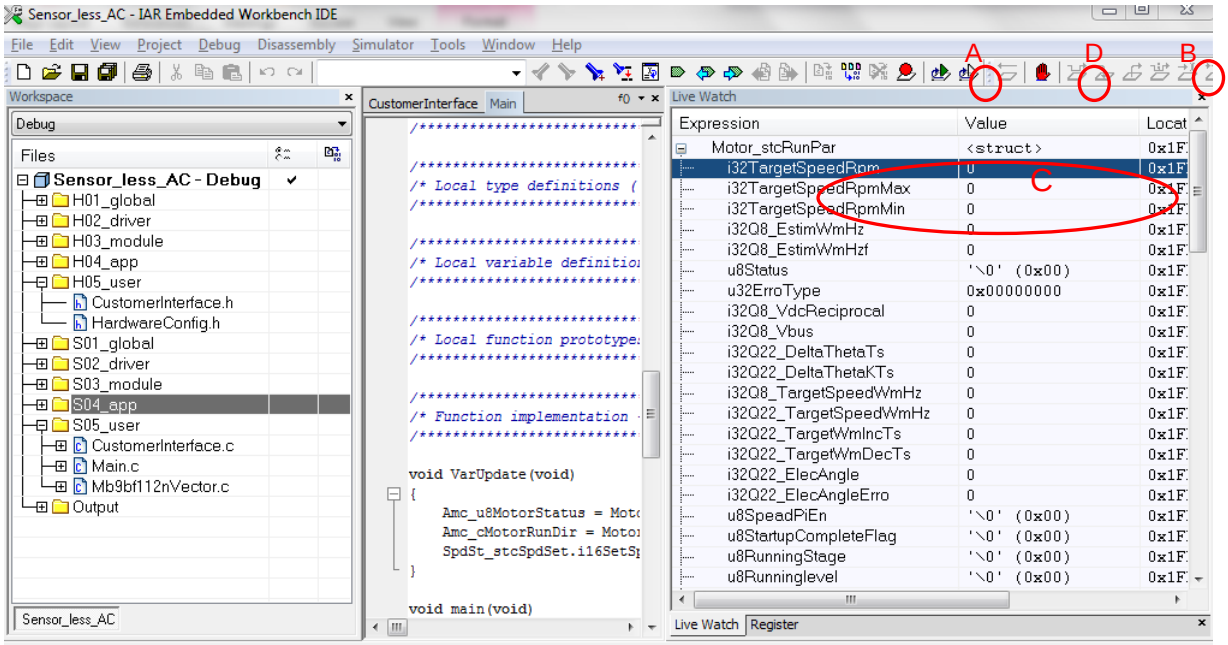
Motor_stcRunPar. i32TargetSpeedRpm	Running Direction	Motor's status
>0	CCW	Running
<0	CW	Running
=0	Stop	Stop

Note:

All of the command speed from the debugger or UART is defined as the motor mechanical speed.

Do not click the button D to break the FW running, the HW over-current or DC over fault may appear and damage the HW if you do that.

Figure 1-6. Motor Run by J-link



3). Watch the important variable in the live watch window of IAR as Figure 1-6 to check the motor running performance such as whether the real motor is achieved the command speed and running speed is stable.

1.3.2 Speed Acceleration and Deceleration

After run motor normally, you can run motor in any speed, and the type speed of the drum for front loading washing machine can be taken for the reference at Table 1-2

Table 1-2. Typical Running Status by the Command Speed

Motor_stcRunPar. i32TargetSpeedRpm	Drum Direction	Description
300~500	CCW	The motor runs at 300~500rpm
-300~-500	CW	The motor runs at 300~500rpm
1000	CCW	The motor runs at 1000rpm
2000	CCW	The motor runs at 2000rpm
4000	CCW	The motor runs at 1000rpm
0	Stop motor	The motor will stop running

If you want to change the default acceleration, you can set the default acceleration '`Motor_f32SpdAccelerationHz`' in the file of '`CustomerInterface.c`' as you want.

The motor can be changed to the hall sensor mode by the macro definition '**`HALLSENSOR`**', you can take section A.1.1 Hardware Setting for your reference.

2. Additional Information

For more information on MMCL, visit our website:

http://www.cypress.com/applications/home-appliances/math-and-motor-control-library

For more Information on Cypress semiconductor products, visit the following website:

http://www.cypress.com/cypress-microcontrollers

Please contact your local support team for any technical question

Appendix

A.1 Common Used Structure

The common used structures for the motor control are shown in the following table, you can use these structures to observe the running performance of the motor.

Item	Structure	Description
Motor Control	`Motor_stcRunPar`	The structure is used to control motor run or stop and the basic running information for the motor such as real running speed, DC bus voltage,
Rotor position and speed	`Angle_stcGenerate`	The structure is used for rotor position generate
	`Motor_stcPll`	The structure for the PLL rotor estimator
	`Spd_stcPar`	The structure is used for rotor speed calculation output
	`Motor_stcAverSpd`	The structure for the average speed calculation
	`Hall_stcCapture`	The structure for the hall sensor capture
	`Spd_stcCalHall`	The structure for the rotor speed calculation for the hall sensor capture
	`Posi_stcCalHall`	The structure for the rotor position calculation for the hall sensor capture
Motor voltage and current	`Motor_stcIuvwSensed`	3 phase stationary axis current
	`Motor_stcIabSensed`	Alpha-beta axis current
	`Motor_stcIdqSensed`	D&Q axis current
	`Motor_stcIdqRef`	D&Q axis reference current
	`Motor_stcVdqRef`	D&Q axis voltage
	`Motor_stcVabRef`	Alpha-beta axis voltage
PID	`Motor_stcWmPidReg`	The structure is used for the speed PI regulator
	`Motor_stcIqPidReg`	The structure is used for the q-axis current ‘Iq’ PI regulator
	`Motor_stcIdPidReg`	The structure is used for the d-axis current ‘Id’ PI regulator

A.2 Firmware Parameter Setting

All of the variables reserved for the user interfaces are located the file ‘`S05_user/ CustomerInterface.c`’ and the macro definitions for the hardware setting are located the file ‘`H05_user/ HardwareConfig.h`’, they are highlighted shown in Figure A 1.

When the motor can’t be started or run well, the variable for the user interface can be found in this section. and all of the files in the sample project can also be found in Figure A 1.

Figure A 1. Interface File Diagram

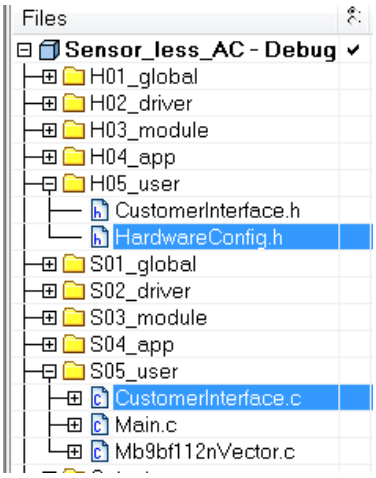


Table A-1. File Description of Project

Folder	File	Description	Remark
S01_global	*Ignored*		
S03_module	*AMCLIB.c*	Advanced Motor Control Library	
	BFLIB.asm	Basic Function Library	
	BMC.c	C file for Basic Motor Control Library	
	BMCLIB.asm	Basic Motor Control Library	
	UDFLIB.asm	Universal Digital Filter Library	
	UMFLIB.asm	Universal Math Function Library	
	UMOLIB.asm	Universal Math Operation Library	
S04_app	*FieldWeaken.c*	The Field Weaken module	
	MMCLApp.c	The position and speed calculation function with hall sensor	
	AdcSample	The ADC process module based on the ADC ISR	
	InitMcu.c	MCU system init include interrupt priority list	
	ISR.c	The ISR file for all of the interrupt routine of the MCU	
	MotorCtrl.c	The main file of the motor control including the main function of FOC process of motor and the start/stop function of motor	
	Motor_Startup.c	The motor start-up module with hall sensor or sensor-less	
	Protection.c	The Protect module	
	SingleShunt.c	The Single Shunt module	
S02_Driver	*Ignored*		
S05_User	*CustomerInterface.c*	The motor parameter setting	
	Main.c	Main function	
	Vector_Table.c	MCU interrupt vector list	

2.1.1 Firmware Setting

All of the variables reserved for the user interfaces are located the file '`S05_user/ CustomerInterface.c`'. Each part will be described in detail as the following sections.

Motor Parameter Configure

The motor parameter such as motor resistance, inductance, and hall angle are set at this part.

```
/** UI_0101  configure motor parameter */
#if 0== MOTOR_ID                    // FXD42BL-2460-002
int32_t   Motor_PolePairs        =  2;// the pole pairs of rotor
float32_t Motor_f32Ld            =  0.65;// the d axis reductance
float32_t Motor_f32Lq            =  0.85;// the q axis reductance
float32_t Motor_f32Res           =  0.94;// the resistance between two phases
float32_t Motor_f32Ke            =  2.86;// inductive voltage constant between two phases
float32_t Motor_f32Flux          =  12;// wb
#define   Motor_f32IsMax         2//3.6
float32_t Motor_CurrentMax       = Motor_f32IsMax;
uint8_t  Motor_HallNumber = THREE_HALL;
float    Wm_TransRate     = 1;       // TransRate of Motor
uint8_t  Motor_HallStatuList[7] = {0,2,6,4,5,1,3};
int32_t  Motor_HallAngleCCW[7] = {0,DEGREE(270-6),DEGREE(30-6),DEGREE(330-6),
                        DEGREE(150-6),DEGREE(210-6),DEGREE(90-6)};
int32_t  Motor_HallAngleCW[7] = {0,DEGREE(150),DEGREE(30),DEGREE(90),
```

Note: The transmission ratio of the motor must be also correctly set, otherwise the motor speed will be incorrect. Wm_TransRate =1 for the DD and DDM motor.

ADC and coefficient Configure

```
/** UI_0102 ADC port and coefficient set */
float32_t Motor_f32IuvwSampleResistor    = 0.02;// Iuvw sample resistor (ohm)
float32_t Motor_i32IuvwAmplifierFactor   = 5.0; // Iuvw calculation factor


int32_t   Motor_i32IuvwOffsetNormal      = 2048;   // the middle value of 12-bits ADC
int32_t   Motor_i32IuvwOffsetRange       = 100;    // ADC offset range of Iuvw sampling
int32_t   Motor_i32IuvwOffsetCheckTimes  = 64;     // Iuvw ADC sample offset check times
```

The Demo Board's current sample resistor is 0.02Ω, current OP's 5 times

Carrier Frequency and Speed Acceleration

The inverter carrier frequency can be set by the reserved variable, it's recommended to set the carrier frequency corresponding to the motor.

```
/** UI_0103 configure Deadtime, carrier frequency, direction and speed*/
float32_t Motor_f32DeadTimeMicroSec      = 2.0f; // Dead timer us
uint16_t  Motor_u16CarrierFreq           = 16000; // motor carry frequency (hz)
                                                  // Range: [5(khz), 8(khz)]
float32_t Motor_f32SpdAccelerationHz     = 10;//1.0f; // acceleration speed
```

The system carrier frequency of demo board is16 KHz. The current sample frequency is 16 KHz. And the dead-time of the SVPWM is 2us.

PID Parameter

```
/** UI_0104 PID Parameter */
float32_t Motor_f32LowSpdKi = 0.0002f;//0.0003f; //speed PI regulator integral constant
float32_t Motor_f32LowSpdKp = 0.043f;//0.15f;    //speed PI regulator proportion constant
float32_t Motor_f32Dki       = 0.02;  //d axis current PI regulator integral constant
float32_t Motor_f32Dkp       = 1;   //d axis current PI regulator proportion constant
float32_t Motor_f32Qki       = 0.02;  //q axis current PI regulator integral constant
float32_t Motor_f32Qkp       = 1;   //q axis current PI regulator proportion constant
float32_t Motor_f32Skp        = 0.05f;    //speed PI regulator proportion constant
float32_t Motor_f32Ski         = 0.0001f; //speed PI regulator integral constant
uint16_t  Motor_u16ChgPiSpdHz = 10;      //PID parameters change at this speed
```

The PID parameter would be changed when the motor is not running stable at someone speed.

Filter Parameter

```
/** UI_0105 Filter */
float32_t Motor_f32BemfLpfkMinSpd       = 6;      // Hz
float32_t Motor_f32BemfLpfkMaxSpd       = 120;     // Hz
int32_t   Motor_i32Q12_BemfLpfkMin      = Q12(0.05);
int32_t   Motor_i32Q12_BemfLpfkMax      = Q12(0.8);
```

These parameters are used for the sensor-less rotor observer that is PLL estimator.

Hall Motor Start-up Parameter

```
/** UI_0106 Start-up param for hall motor*/
int32_t  Motor_StartSpd   = 280;       // start up drum speed,unit:rpm
float    Startup_InitCur  = 0.3;      //initial startup current, unit:A
float    Startup_IncCur   = 0.08;     //startup current increase step, unit:A
float    Startup_SwitchCur = 0.3; //Switch current at closeloop
```

If the sensor motor does not start well, these parameter can be modified to make the motor start smooth.

Sensor-less Motor Start-up Parameter

The parameters for sensor-less motor start-up are shown as below. When the sensor-less motor do not start well, the parameter can be modified to improve the start-up performance.

```
/** UI_0107 running control of open loop*/
float32_t Motor_f32BemfLpfkMinSpd = 6;      // Hz
uint8_t   Motor_u8RunLevel        = 4;     // 1->orientation, 2->open loop running,
                                                // 3->closed loop running, 4->change speed enable
uint32_t  Motor_u32PreheatTimeSec       = 0;
uint16_t  Motor_u16PreheatSpdRpm        = 1;     //the rpm of prehot
int16_t   Motor_i16Q8_PreheatCurrent = Q8(Motor_f32IsMax*0.7); //the voltage for prehot
int16_t   Motor_i16Q8_OrientIqRef = Q8(Motor_f32IsMax*0.7); // orientation current
uint16_t  Motor_u16OpenLoopSpdIncHz     = 1;
uint16_t  Motor_u16OpenLoopSpdInitHz    = 0;
uint16_t  Motor_u16OpenLoopSpdEndHz     = 1;
int16_t   Motor_i16Q8_OpenLoopIqRef     = Q8(Motor_f32IsMax*1.0); // q axis current referencein open
loop stage
/** running control of closeloop */
uint16_t  Motor_u16ColseLoopTargetSpdHz = 1;
int16_t   Motor_i16Q8_CloseLoopIsMax    = Q8(Motor_f32IsMax);
```

Motor Protection Parameter

The speed range and the phase current of the motor and other protection parameter for the motor control can be modified at this part.

```
/** UI_0108 Protection Parameter */
uint16_t  Motor_u16SpdMax             = 4000;   // motor run maximum speed rpm 9000
uint16_t  Motor_u16SpdMin             = 180;    // motor run minimum speed rpm 1
int16_t   Motor_i16Q8CurrentMax       = Q8(Motor_f32IsMax); // motor phase current peak A
uint16_t  Motor_u16OverCurrentTimeSec = 1;      // motor over-current timer PWM
uint16_t  Motor_u16VbusMax            = 27;    // the maximum value of DC voltage
uint16_t  Motor_u16VbusMin            = 21;     // the minimum value of DC voltage
uint8_t  Motor_ErrorTime              = 10;     //s
```

A.1.1 Hardware Setting

The Hardware settings for the HW can be set in the H file ‘H05_user/ HardwareConfig.h’

Clock and ADC Set

The MCU clock and the ADC port and relevant coefficient are set at this part.

```
/** UI_0201 ADC port and coefficient set */
#define CLK_XTAL_FREQ   4 // MHz
#define CLK_SYS_FREQ    9 // System clock frequency = (CLK_XTAL_FREQ * (CLK_SYS_FREQ + 1))
#define APB0_CLK_DIV         APBx_CLK_DIV0     //
#define APB1_CLK_DIV         APBx_CLK_DIV0     //
#define APB2_CLK_DIV         APBx_CLK_DIV0     //
#define ADC_VDC_FACTOR      21.4
#define ADC_VOLT_REF        5.0f     // Reference voltage for ADC
#define ADC_VALUE_MAX       4096.0f     // 12-bits ADC
#define ADC_CH_VDC          ADC_CH_2    // ADC channel - Vdc
#define ADC_CH_IU           ADC_CH_1    // ADC channel - Iu
#define ADC_CH_IV           ADC_CH_0    // ADC channel - Iv
```

The DC Bus voltage sample factor of Demo Board is 21.4.

Hall I/O and Pin set

The macro definition '**HALLSENSOR**' must be set to '**FALSE**' when the motor is sensor-less.

```
/** UI_0202 configure hall I/O and Pin    */
#define HALLSENSOR     TRUE    //HALL sensor correct
/** UI_0303 configure hall I/O and Pin    */
#define HALL_A_PORT    PORT4   //Port 4
#define HALL_A_PIN     PIN9    //P49
#define HALL_A_TIMER   BT_CH_0 //timer 0
#define HALL_A_TIMER_CH        0       //timer 0_0
#define HALL_B_PORT    PORT4   //Port 4
#define HALL_B_PIN     PIN10   //P4A
#define HALL_B_TIMER   BT_CH_1 //timer 1
#define HALL_B_TIMER_CH        0       //timer 1_0
#define HALL_C_PORT    PORT6   //Port 6
#define HALL_C_PIN     PIN1    //P61
```

A.3 Motor Parameter

The motor parameters used for the sample project are shown in the following table.

Pole pairs	2
D-axis Inductance	0.65mH
Q-axis Inductance	0.85mH
Resistance(line to line)	0.94 ohm
Inductive voltage constant(line to line)	2.86 V/krpm
Saturation current	2A
Speed range	360rpm~4000rpm
Hall Number	3
Hal line definition	Red(Vcc),Black(GND) Yellow(Hall A),Green(Hall B),Blue(Hall C)
Motor phase definition	Yellow(U),Green(V),Blue(W)

Revision History

Document Revision History

Document Title: FM3 Family 32-bit Microcontroller Math and Motor Control Library Quick Start Guide Document Number: 002-04766				
Revision	**ECN#**	**Issue Date**	**Origin of Change**	**Description of Change**
**	—	09/19/2014	BOZH	Initial release
*A	5275832	08/29/2016	BOZH	Migrated to Cypress format