FM3

INVERTER SENSOR WASHING MACHINE FIRMWARE

32-BIT MICROCONTROLLER
FM3 Family

USER MANUAL

SPANSION®

Publication Number AN706-00095-1v0-E Revision 1.0 Issue Date Jan 26, 2015

Target products

This user manual describes the following products:

Series	Product Number
FM3 Series	MB9AF111K, MB9AF312K,MB9BF121J

Table of Contents

Figures

Tables

1. Introduction

1.1 Purpose

This user manual describes SPANSION inverter sensor washing machine solution, and describes how to use inverter washing machine FW library.

This document will help you acquire a quick understanding of how to build a washing machine project and how to debug the motor with SPANSION inverter sensor washing machine FW library.

The document introduces the basic information of the washing machine solution including hardware, firmware, initial functions, basic motor setting functions and FOC drive modules. When you have understood these contents, you can get an overview of a whole washing machine project. And you can run a motor following the demo project steps.

1.2 Definitions, Acronyms and Abbreviations

API	-	Application Programming Interface
FOC	-	Field Oriented Control
FW	-	Firmware
HW	-	Hardware
I/O	-	Input and output
CW	-	Clockwise
CCW	-	Counter clockwise
WM	-	Washing Machine

1.3 Document Overview

The rest of document is organized as the following:

Chapter 2 explains System Hardware Environment

Chapter 3 explains Development Environment

Chapter 4 explains System Firmware Design

Chapter 5 explains System Function

Chapter 6 explains Event Function

Chapter 7 explains Driver Function

Chapter 8 explains Interrupt Function

Chapter 9 explains Demo System

1.4 Reference Documents

HW User Manual: MCU-UM-510115-E-12-WM-EVB_HW.pdf

2. System Hardware Environment

The following lists the MCUs used in washing machine inverter board.

CPU chip: Spansion MB9AF111K/ MB9AF312K

CPU Frequency: 40MHz

MCU pin number: 48pin

RAM Space: 16Kbytes

Code Space: 128Kbytes

Demo HW version: WM-MAINBORAD-V0.3.1

3. Development Environment

Table 3-1: MCU Development Environment

Name	Description	Part Number	Manufacturer	Remark
IAR bedded Workbench6.40	FW code edit , compile and debug	N/A	N/A	N/A
J-Link	Debug and Load FW by JTAG	N/A	N/A	N/A
SPANSION FLASH LOADER	Flash download program	N/A	N/A	N/A
Source Insight V3.50	Source code edit	N/A	N/A	Editor
Eclipse	Source code edit	N/A	N/A	Editor

4. System Firmware Design

This chapter introduces the FW structure of inverter washing machine project.

4.1 FW Feature

The features of the sensor inverter washing machine solution are shown in Table 4-1.

Table 4-1: Feature List of Sensor WM Solution

No	Feature	Description	Remark
1.	Hall Self-check	Hall status self-check	
		Hall phase angle self-check	
		Check whether the hall circuit in hardware part is normal.	
2.	Hardware Self -check	Check whether the current sample by A/D is right	
3.	Adjustable Carrier Frequency	Carrier frequency can be set by the corresponding variable in user interface	
4.	Hall Angle Control	Rotor electrical phase angle was corrected by hall edge jump interrupt	
5.	Motor Speed Calculate	Calculate speed by hall signals	
6.	Field Weaken Control	Run motor in field weaken area to raise speed	
7.	FOC Control	Using FOC control algorithm	
8.	Self-adaption Start Up	Adaptive to different load to start-up motor	
9.	Speed regulate	This function is used to speed up or slow down a motor by the command from host via UART or debugger	
10.	Brake	Stop motor by braking down	
		Speed Slow down by braking algorithm	
11.	Current Sample	Dual shunts sample	
		Single shunt sample algorithm	
12.	Protect	DC voltage protection	
		A/D offset protection	
		Lock rotor protection	
		Power protection	
		Hall single line lost protection	
		IPM temperature protection	
		Motor phase lost protection	
		Over Current Protection	
13.	OOB	Out of balance (OOB) load detection	
14.	Weight	The weight of the load detection	
15.	Un-Stop Running	Motor can switch running direction (CCW and CW) without stopping motor	
16.	UART	Receive and transform data to Host PC	

4.2 FW Structure

There are 5 layers in the FW structure of IAR, which are shown in Figure 4-1.

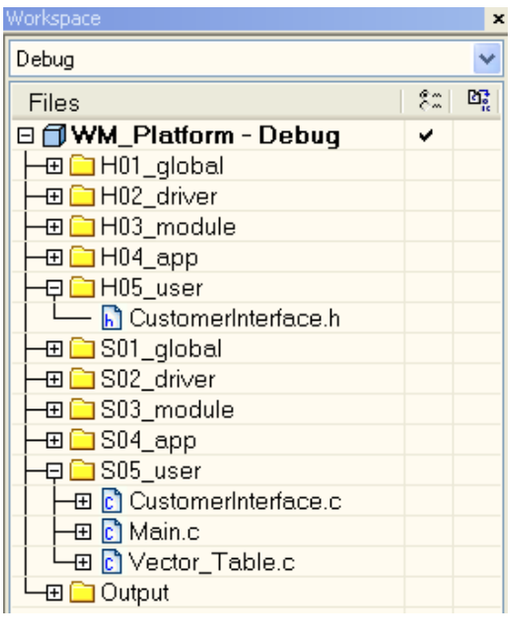

Figure 4-1: Structure of FW

The C source and Header files which are included in each layer are shown in Table 4-2

Table 4-2: Directory Description of Project

Layer	Folder	Description
global	H01_global, S01_global	MCU system file
driver	H02_driver, S02_driver	MCU register setting function such as GPIO, interrupt, MFT, AD
module	H03_module, S03_module	Algorithm folder for basic motor control such as FOC frame transform , SVPWM, math, PID, filter
app	H04_app, S04_app	Application folder for the files of application functions such as speed and position generator by hall sensor, protection, motor start-up, field weaken, brake, weight, OOB,UART, and etc.
user	H05_User, S05_User	Customer interface folder for the files for motor Configuration and HW setting

Note: if you want to quick start the motor, you can refer to the setting for user layer at 9.2.1FW Interface Configuration and chapter 5 ‘System Function’

The sub-files in each folder are shown in Figure 4-2, and the structure of header files is the same as C files.

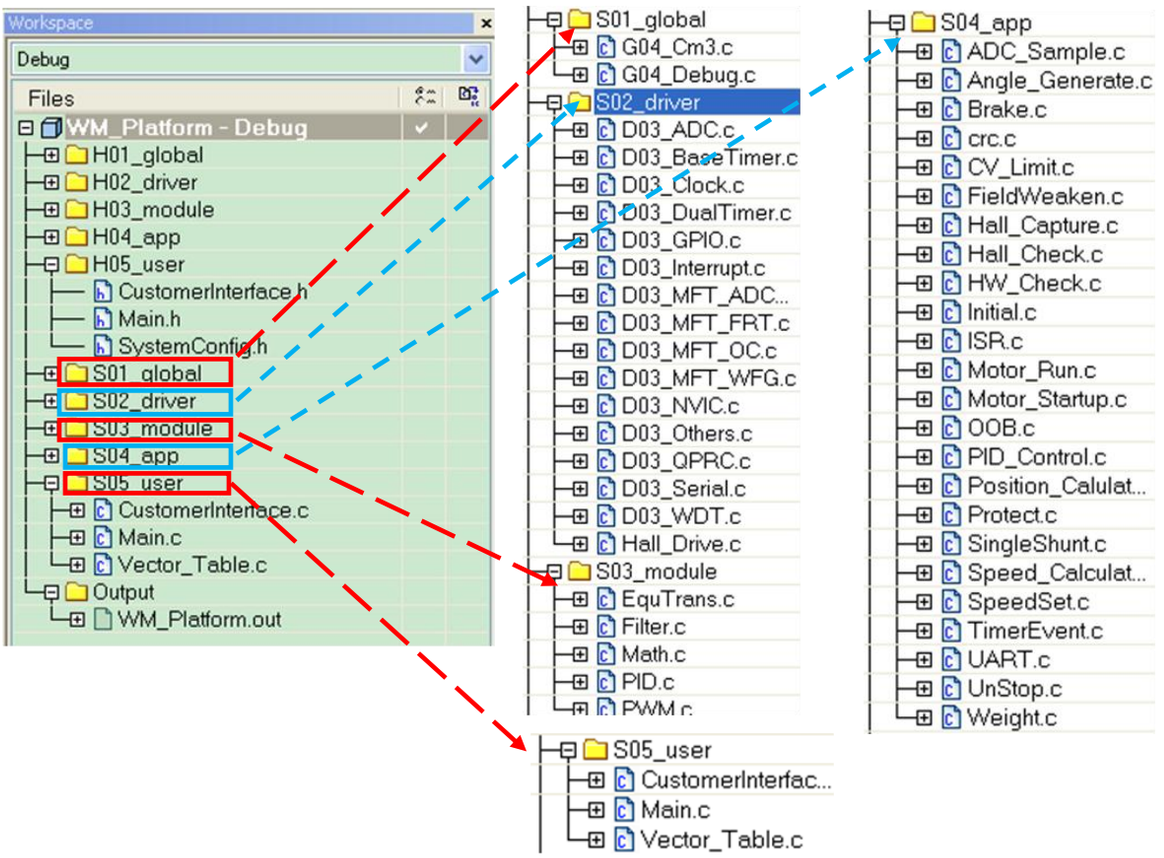

Figure 4-2: Sub-files in Each Layer

The relationship between each layer is shown as the diagram in Figure 4-3.

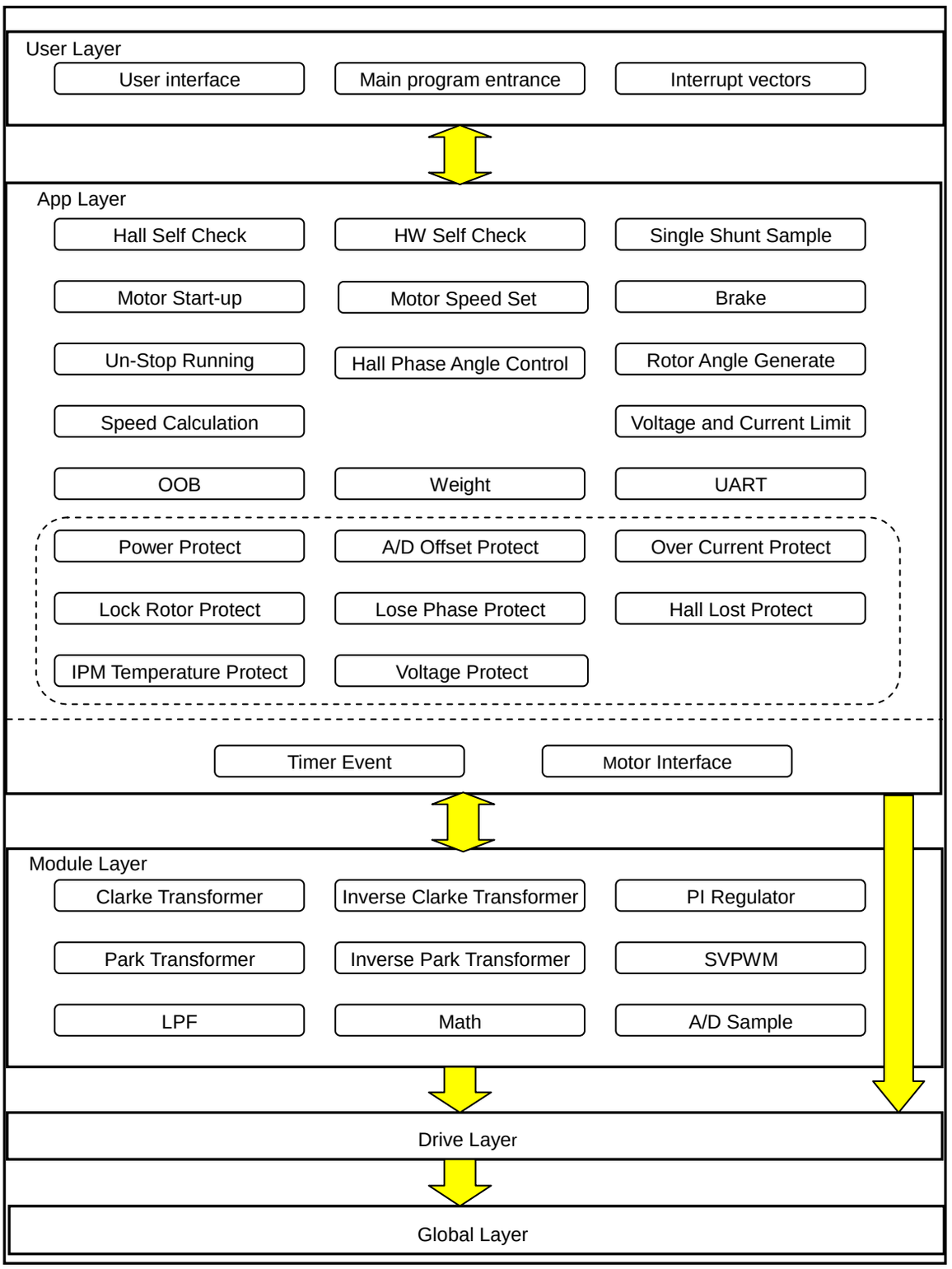

Figure 4-3: Sensor WM FW Architecture

4.3 Files Description

The detailed descriptions for each file are shown in Table 4-3.

Table 4-3: File Description of Project

Folder	File	Description
S01_global	G04_Cm3.c	The file for MCU driver
	G04_Debug.c	Debug information for MCU driver
S03_module	EquTrans.c	FOC axis convert
	Filter.c	One order low pass filter
	Math.c	The math module including the function such as SQRT,COS and SIN
	PID.c	The PID module for current and speed PI
	PWM.c	The SVPWM module
S04_app	ADC_Sample.c	The ADC process module based on the ADC ISR
	Angle_Generate.c	The rotor angle generation module
	Brake.c	The brake module including the speed down by brake
	CV_Limit.c	The FOC current and voltage limitation module
	FieldWeaken.c	The Field Weaken module
	Hall_Capture.c	Hall capture module
	Hall_Check.c	Hall check module
	HW_Check.c	HW check module
	Initial.c	MCU system initialization including interrupt priority list
	ISR.c	The ISR file for all of the interrupt routine of the MCU
	Motor_Run.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
	OOB.c	The OOB detect module
	PID_Control.c	The PID control module that including the Speed PI, current PI, PI parameter self-changing
	Position_Calulate.c	The position calculate module
	Protect.c	The protect module
	SingleShunt.c	The single shunt module
	Speed_Calculate.c	The speed calculate module
	SpeedSet.c	The speed setting module
	Timer_Event.c	Timer event module
	UART.c	The UART module
	UnStop.c	The Unstop running module
	Weight.c	The electrical weight module
S02_Driver	Ignored	
S05_User	CustomerInterface.c	The motor parameter setting
	Main.c	Main function
	Vector_Table.c	MCU interrupt vector list

4.4 FW Control Flow

The control flow for the motor is shown as Figure 4-4. There are 4 interrupts that are red highlighted for the motor FOC control, hall capture and AD converter. The timer events are executed in the end-less loop and the timers are generated in the zero detection interrupt ‘ISR_MFT_FRT’ of the free run timer 0.

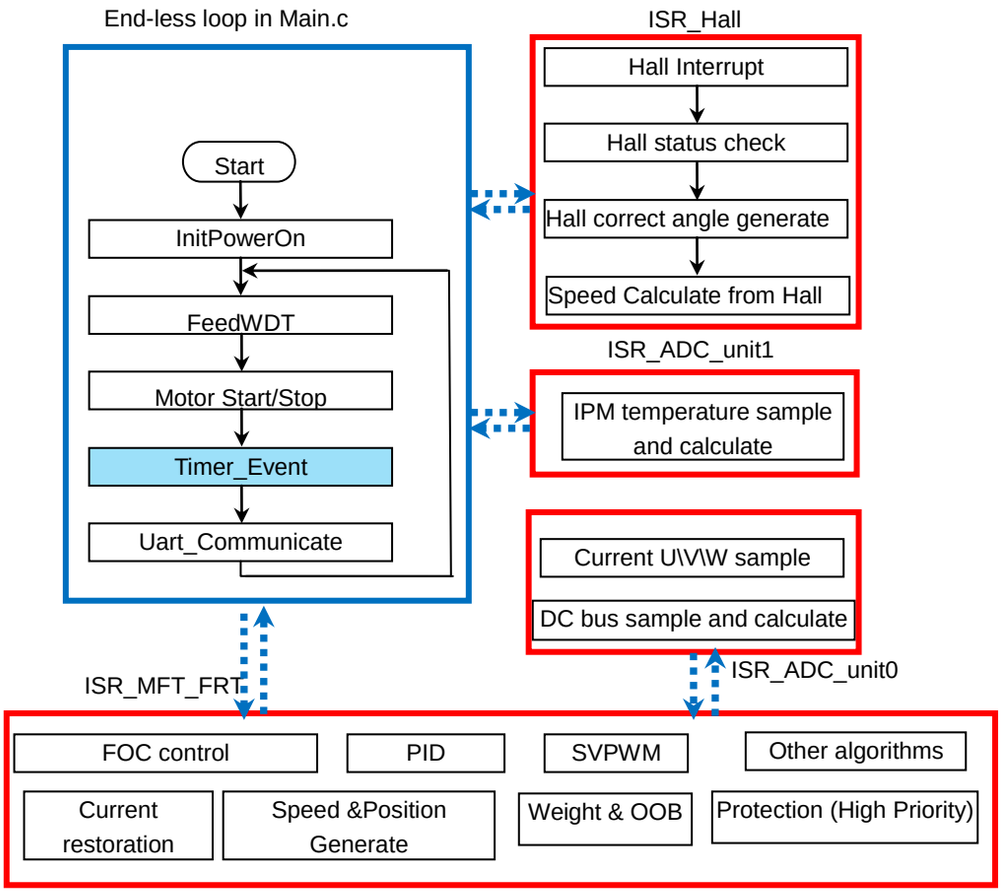

Figure 4-4: Diagram of the Control Flow

5. System Function

This chapter introduces the system function of the macro definition, global structure definition, and function definition in detail.

5.1 Macro Definition

The macro definition for the user will be described in section ‘9.2.1FW Interface Configuration’

5.2 Global Structure and Variable Definition

The variable for user interface can be found in section ‘9.2.1FW Interface Configuration’.
Any structure or variable that you want to watch can be pasted into the ‘Live Watch’ window of IAR as shown in Figure 5-1.

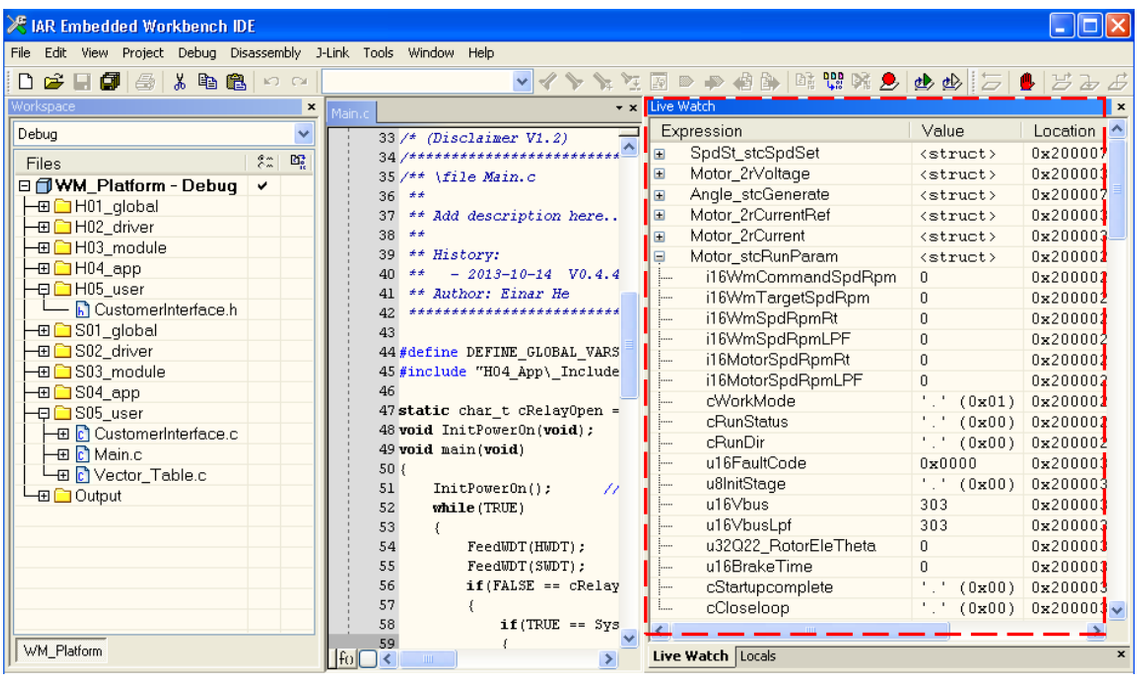

Figure 5-1: Diagram of Live Watch

5.2.1 Variables for Motor Running

Motor_stcRunParam

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, washing machine work mode, etc. Detailed information can be found in the comments for each variable.

```
typedefstruct
{
        int16_t i16WmCommandSpdRpm;//the command speed of drum from UART or debugger
online,unit:rpm
        int16_t i16WmTargetSpdRpm;//the middle speed for the reference speed of
speed PI,unit:rpm
        int16_t i16WmSpdRpmRt;  //the real-time drum speed of washing machine
        int16_t i16WmSpdRpmLPF; //the filtered drum speed of washing machine
        int16_t i16MotorSpdRpmRt;//the real-time motor speed of washing machine
        int16_t i16MotorSpdRpmLPF;//the filtered motor speed of washing machine
        char_tcWorkMode;        //wash or spin work mode
        char_tcRunStatus;       //run status: 0--stop,1--Run
        char_tcRunDir;          //run direction: CW or CCW
        uint16_t u16FaultCode;  //protection fault code
        uint8_t  u8InitStage;   //the start initial state machine
        uint16_t u16Vbus;       //the DC bus voltage, unit:V
        uint16_t u16VbusLpf;    //the DC bus voltage lpf value
        uint32_t u32Q22_RotorEleTheta;//the rotor position angle
        uint16_t u16BrakeTime;  //brake time, unit:1ms
        char_tcStartupcomplete;//flag for motor startup finish
        char_tcCloseloop;     //flag for the motor closed loop running
} stc_MotorRunParam_t;
externstc_MotorRunParam_t Motor_stcRunParam;
```

SpdSt_stcSpdSet

The structure is used to set the drum speed. It is the global structure for the SpeedSet module that is realized in file ‘S04_app/ SpeedSet.c’. Detailed information can be found in the comments for each variable, the variables in this structure are not recommended to modify.

```
typedefstruct stc_SpdSet
{
    int16_t      i16SetSpeed;    //setting speed of drum, unit:rpm
    int16_t      i16SetSpeedPre; //previous setting speed of drum, unit:rpm
    uint16_t     u16SpdChgTime;  //speed change time from spd A to B
    uint16_t     u16CommandSpeed; //the command speed of drum,unit:rpm
    char_t       cWorkMode;      //the WM working mode: wash or spin
    uint16_t     u16SpdChgCounter; //the speed regulate counter
    uint8_t      u8SpdChgStep;   //the speed change step for speed regulate
    char_t       cMotorStartFlag;//motor start flag
    char_t       cMotorStopFlag; //motor stop flag
    char_t        cRotateDir;    //motor running direction
    uint16_t     u16AcceLmt;     //the acceleration limit at speed up
    uint16_t     u16DeceLmt;     //the acceleration limit at speed down
    uint16_t     u16SpeedMax;    //the maximum speed limit of drum speed
    uint16_t     u16SpeedMin;    //the min speed limit of drum speed
} stc_SpdSet_t;
stc_SpdSet_t    SpdSt_stcSpdSet;
```

5.2.2 Variables for FOC

The variables for the FOC control are introduced in this section.

D&Q axis Current and Voltage

Motor_2rCurrentRef <struct> Q8_d 0 Q8_q 0 Q12_Cos 0 Q12_Sin 0	Reference current value on the 2 axis rotation frames Reference current on D-axis ‘Idref’ Reference current on Q-axis ‘Iqref’ Cosine value of the rotor position used for the frame transform Sine value of the rotor position used for the frame transform
Motor_2rCurrent <struct> Q8_d 0 Q8_q 0 Q12_Cos 2062 Q12_Sin 3538	current value on the 2 axis rotation frames Real-time current on D-axis ‘Id’ Real-time current on Q-axis ‘Iq’ Cosine value of the rotor position used for the frame transform Sine value of the rotor position used for the frame transform

Motor_2rVoltage <struct> Q8_d 0 Q8_q 0 Q12_Cos 2062 Q12_Sin 3538	Voltage value on the 2 axis rotation frames Real-time voltage on D-axis ‘Vd’ Real-time voltage on Q-axis ‘Vq’ Cosine value of the rotor position used for the frame transform Sine value of the rotor position used for the frame transform

Motor_Offset

The AD middle points of amplifier part on the HW are got in this structure. If the middle voltage of the amplifying circuit for the phase current is changed, the AD offset result will also be changed.

Motor_Offset <struct> U 2073 V 2040 W 0	Motor_Offset
	AD middle point for current Iu AD sample
	AD middle point for current Iv AD sample
	AD middle point for current Iw AD sample
	2048 = 2.5V, the offset error threshold is set by 'AD_OFFEST_MAX_VALUE'

Startup_stcCtrl

The structure is used for the motor start-up control. The detailed information can be found in the comments for each variable.

Startup_stcCtrl <struct> cStartComplete '.' (... cClosedLoop '.' (... cRunStage '.' (... cRunLevel '.' (...	
	Flag for motor startup complete,1→start finished
	Flag for motor closed loop running,1→speed closed loop
	Flag for the motor startup stage
	Flag for the motor startup and running level

Limit_stcCalc

The structure is used for the FOC current and voltage limitation to ensure the reliability of the inverter. The detailed information can be found in the comments for each variable.

Limit_stcCalc <struct> i32Q8_VdLmt 0 i32Q8_VqLmt 46192 i32Q8_IdLmt 0 i32Q8_IqLmt 0 i32Q8_IsLmt 0	
	D-axis voltage limit
	Q-axis voltage limit
	D-axis current limit, especially in field weaken
	Q-axis current limit
	Saturate phase current

FieldWeaken_stcCtrl

The structure is used for filed weaken control. The detailed information can be found in the comments for each variable.

FieldWeaken_stcCtrl <struct> cExeFlag '.' (0x00) u8ExeCnt '.' (0x00) u8ExeCycle '.' (0x00) u32BaseSpd 0 cForceOut '.' (0x00) u8ForceOutCycle '.' (0x00) u32BaseSpdRecord 0 u16DCVoltageRecord 0	
	Flag for the field weaken execution
	Counter for the field weaken PI
	The cycle of field weaken PI ,unit:1ms
	The base drum speed of motor without filed weaken
	Exit the field weaken by the load disturbance
	The cycle of field weaken PI ,unit:1ms
	The recorded base speed of drum speed
	The recorded DC bus voltage when enter the field weaken

5.2.3 Variables for Speed and Position

Angle_stcGenerate

The structure is used for rotor position generation. The detailed information can be found in the comments for each variable.

Angle_stcGenerate <struct> i32Q22_RotorAngle 3495000 i32Q22_RotorDtheta 0 i32Q22_RotorDthetaMin 830 i32Q26_RotorDthetaKts 1328 u8StartPassHallNumber '.' (0x00)	
	Rotor's output angle
	Rotor's forward angle every PWM
	Rotor's min forward angle every PWM
	Rotor's forward angle calculated factor
	Rotor pass hall number when start up

Spd_stcPar

The structure is used for outputting the rotor speed calculation result. The detailed information can be found in the comments for each variable.

Spd_stcPar	<struct>	
i32MotorRpmLpf	0	The output motor average speed
i32MotorRpmRt	0	The output motor real time speed
i32WmRpmLpf	0	The output WM average speed
i32WmRpmRt	0	The output WM real time speed
i32Q12_InvWMRatio	428	1/trans-ratio
SpdLpfParam	<struct>	
i32MotorEleSpd	0	Motor's real-time electrical speed

5.2.4 Variables for PID Control

The structures used for PID control are introduced in this part.

Pid_stcCtrl

The structure is used for PID control that enables or disables the corresponding PI regulator. The detailed information can be found in the comments for each variable.

Pid_stcCtrl	<struct>	
cIdEN	'.' (0x01)	Id PI Enable
cIqEN	'.' (0x01)	Iq PI Enable
cSpdEN	'.' (0x01)	Speed PI Enable
cFdWkEN	'.' (0x01)	Field weaken PI Enable
cFdWkPIExe	'.' (0x00)	Field weaken execution flag
cSpdPIExe	'.' (0x00)	Speed PI execution flag
u8IdPICyc	'.' (0x01)	Execute cycle of Id PI
u8IqPICyc	'.' (0x01)	Execute cycle of Iq PI
u16SpdPICyc	1	Execute cycle of speed PI, unit: ms
u16FdWkPICyc	5	Execute cycle of field weaken PI, unit: ms

Pid_stcSpdPI

The structure is used for the speed PI regulator. The detailed information can be found in the comments for each variable.

Pid_stcSpdPI	<struct>	
Q8_kp	3030	Kp parameter for speed PI, Q8 format
Q8_ki	25	Ki parameter for speed PI, Q8 format
Q8_kd	0	Kd parameter for speed PI, Q8 format
Q16_Pout	0	Pout of speed PI, Q16 format
Q16_Iout	13379	Iout of speed PI, Q16 format
Q16_Dout	0	Dout of speed PI, Q16 format
Q8_Error	0	Input error of speed PI, Q8 format
Q8_ErrorPre	0	Previous input error of speed PI, Q8 format
Q8_Out	52	Output of speed PI, Q8 format
Q8_Outmax	1696	Max output limit of speed PI, Q8 format
Q8_Outmin	0	Min output limit of speed PI, Q8 format

Pid_stcIqPI

The structure is used for the q-axis current 'Iq' PI regulator. The detailed information can be found in the comments for each variable.

Pid_stcIqPI	<struct>	
Q12_kp	45056	Kp parameter for Iq PI, Q12 format
Q12_ki	122	Ki parameter for Iq PI, Q12 format
Q20_Pout	−315392	Pout of Iq PI, Q20 format
Q20_Iout	21692088	Pout of Iq PI, Q20 format
Q8_Error	5	Input error of Iq PI, Q8 format
Q8_Out	5290	Output error of Iq PI, Q8 format
Q8_Outmax	46100	Max output limit of Iq PI, Q8 format
Q8_Outmin	0	Min output limit of Iq PI, Q8 format

Pid_stcIdPI

The structure is used for the d-axis current 'Id' PI regulator. The detailed information can be found in the comments for each variable.

Pid_stcIdPI	<struct>	
Q12_kp	45056	Kp parameter for Id PI, Q12 format
Q12_ki	122	Ki parameter for Id PI, Q12 format
Q20_Pout	90112	Pout of Id PI, Q20 format
Q20_Iout	1380918	Pout of Id PI, Q20 format
Q8_Error	4	Input error of Id PI, Q8 format
Q8_Out	353	Output of Id PI, Q8 format
Q8_Outmax	45061	Max output limit of Id PI, Q8 format
Q8_Outmin	−45027	Min output limit of Id PI, Q8 format

FieldWeaken_stcPiParam

The structure is used for field weaken PI regulator. The detailed information can be found in the comments for each variable.

FieldWeaken_stcPiParam	<struct>	
Q8_kp	0	Kp parameter for Field Weaken PI, Q8 format
Q8_ki	12	Ki parameter for Field Weaken PI, Q8 format
Q8_kd	0	Kd parameter for Field Weaken PI, Q8 format
Q16_Pout	0	Pout of Field Weaken PI, Q16 format
Q16_Iout	0	Iout of Field Weaken PI, Q16 format
Q16_Dout	0	Dout of Field Weaken PI, Q16 format
Q8_Error	0	Input error of Field Weaken PI, Q8 format
Q8_ErrorPre	0	Previous input error of Field Weaken PI, Q8 format
Q8_Out	0	Output of Field Weaken PI, Q8 format
Q8_Outmax	1433	Max output limit of Field Weaken PI, Q8 format
Q8_Outmin	−12	Min output limit of Field Weaken PI, Q8 format

5.2.5 Variables for Washing Machine Application

The variables for the advanced application of the washing machine are introduced in this section.

Weight_stcCtrl

The structure is used for the weight control. The detailed information can be found in the comments for each variable. The weight result and the inner data can be observed in this structure.

Weight_stcCtrl <struct>	
cStart '.' (0x00)	Weight start flag
cPowerDetectStart '.' (0x00)	Start detecting the power in weight
cReachSpdN2 '.' (0x00)	Flag for the speed acceleration finish
u8WtFinish '.' (0x00)	Weight finish flag, 1--finish 2--weight over time
u8WtStage '.' (0x00)	Weight stage
u32PowerN1 <array>	Average power in one drum cycle at stable running N1
u32PowerAcce 0	Sum power at weight speed up
u16AcceCycle 0	Drum cycle at weight speed up
u32WtValueTemp 0	Original weight result of the load
u32WtValue 0	Weight result of the load by the DC voltage compensation
u16LoadValue 0	Weight result of the load
u32WtTimeOut 960000	Max weight time, unit: s

OOB_stcCtrl

The structure is used for OOB detect. The detailed information can be found in the comments for each variable.

OOB_stcCtrl <struct>	
cOOBEn '.' (0x00)	OOB detect start flag
u8OOBStage '.' (0x00)	OOB detection stage, 4 means OOB finished
u32OobData 0	Original OOB data of the load
u16OOBValue 65535	OOB result to host

UnStop_stcParam

The structure is used for un-stop running. The detailed information can be found in the comments for each variable.

UnStop_stcParam <struct>	
cStart '.' (0x00)	Start unstop running
cStop '.' (0x00)	Stop unstop running
cForceRunning '.' (0x00)	Run in force status flag
cFirstCompose '.' (0x00)	
cAngleComposeStart '.' (0x00)	Angle compose start flag
i32Q22_AngleError 0	Angle error between rotor and hall
i32Q22_AngleComposeDth... 0	Compose angle speed

5.3 Function List

The functions for the system control are shown in Table 5-1.

Table 5-1: System Function List

Prototype	Description	Remark
void main(void)	Main function of the whole project	Main.c
InitPowerOn()	The initial function for all the MCU resource initial and key variable initial after the power is on	Main.c
Motor_RunInit(Motor_CARRY_FREQ)	The function for the motor start control but not for the motor start-up.	Motor_Run.c
Motor_StopControl()	The function for the motor stop control	Motor_Run.c
Uart_Communicate()	The main function for the UART communication	UART.c
static void Initial_Motor_RunPar(unsigned short sample_freq)	The key variable and the register initial at the motor start	Motor_Run.c
void Motor_Process(void)	The main function of the motor control that is called in each of the MFT zero detect ISR	Motor_Run.c
void Debug_Process(void)	The main function of the test mode for the hall and HW check ,and is also called in each of the MFT zero detect ISR	Motor_Run.c
void Debug_Watch(void)	The basic variable assignment for the motor running	Motor_Run.c
void Timer_Counter(void)	The 1ms/5ms/50ms timer generated by the MFT ISR	TimerEvent.c
void Timer_Event(void)	The timer event for the motor control or the advanced function	TimerEvent.c

6. Event Function

The primary functions for the motor inverter control are introduced in this chapter

6.1 Function List

The functions for the motor control that are called in the MFT ISR '*Motor_Process()'* and timer '*Timer_Event()'* are shown in Table 6-1 and Table 6-2.

Table 6-1: Event Function List by the 'Motor_Process()

Prototype	Description
Hall_CaptureOn();	The hall status check at the first electrical cycle of the motor by the query mode when the power is on
UnStop_Run()	The main function for the un-stop running
Spd_EstimateCalculate()	The speed calculate function by the estimator
Spd_Calculate()	The speed calculate function by the estimator and hall module
Motor_Sense()	The phase current restoration from ADC converter
ClarkeTransform(&Motor_3sCurrent, &Motor_2sCurrent)	The function of the Clarke frame transform
ParkTransform(&Motor_2sCurrent, &Motor_2rCurrent)	The function of the Park frame transform
Posi_Estimate(…)	The function of the rotor position estimator
Posi_Calculate()	The function of the rotor position calculation from the estimator and hall module
Angle_Generate()	The function of the rotor position generation
Current_PI(…)	The d/q current PI regulator
Startup_HallMotor()	The motor start-up function for the hall sensor motor
InvertParkTransform(…)	The function of the inverse Clarke frame transform
InvertClackeTransform(…)	The function of the inverse Park frame transform
SVPWM_Calc(…)	The SVPWM function
Write_MFT_OCCP(…)	The function for the OCCP register setting according to the SVPWM calculate result
Weight_LoadMeasure()	The function for the weight
OOB_Detect()	The function for the OOB
Protect_HallLost(…)	The protection function for the hall lost detect
Protect_OpenPhase(…)	The protection function for the open phase detect

Table 6-2: Event Function List by the ‘Timer_Event()’

Prototype	Description	Remark
SpdSt_SpeedSet(…)	The speed set function used for the motor speed acceleration or deceleration	1ms timer
SpdSt_SpeedRegulate(…)	The speed regulation function for the middle speed generation	
FieldWeaken_Control()	The main function for the field weaken	
Brake_SpeedDown()	The function of the speed down by brake	
PID_ParameterChange()	The function of the PID Parameter Change	
Speed_PI(…)	The function of the speed PI regulator	
Limit_Calculate()	The function of the FOC current and voltage limitation	5ms
Protect_LockRotor()	The function of the motor lock protection	
Protect_Power()	The function of the motor running power protection	
Protect_Voltage()	The function of the DC bus over and under protection	
Protect_IpmTemperature()	The function of the IPM temperature protection	
Debug_Watch();	The basic variable assignment for the motor running	50ms
Uart_Protect();	The function of the UART lost protection	

7. Driver Function

The MCU peripheral resources used for motor control are introduced in this chapter.

7.1 Function List

The MCU peripheral driver functions are mainly located in the file ‘S0’4_app/Initial.c

Table 7-1: Driver Function List

Prototype	Description	Remark
void InitNVIC(void)	The interrupt enable and the priority set that used in the motor control	Initial.c
void InitClock(void)	MCU clock initialization	Initial.c
void InitWDT(void)	Watch dog initialization	Initial.c
void InitGPIO(void)	The used GPIO initial, user can add the GPIO for other usage	Initial.c
void Motor_SVPWM_Init(void)	The SVPWM initial such as the FRT mode and cycle, AD trigger source, OCCP mode, etc.	Initial.c
void Motor_SVPWM_En(void)	Enable the SVPWM output	Initial.c
void Motor_SVPWM_Dis(void)	Disable the SVPWM output	Initial.c
void InitADC(unsigned short Motor_Sample_freq)	The AD initial such as the port setting, converter time setting, trigger point, etc.	Initial.c
void Motor_configPWM(void)	Configuration the PWM such as the dead time of the SVPWM, max duty	Initial.c
void Brake_On(void)	Porting setting for motor brake	Brake.c
void Brake_Off(void)	Release the port to finish the brake	Brake.c

8. Interrupt Function

8.1 Function List

Table 4-1: System Used Interrupt Function

Prototype	Description	Remark
__root void ISR_HardWatchdog(void)	The HW watch dog ISR	S04_app/ISR.c
__root void ISR_SoftWatchdog(void)	The software watch dog ISR	S04_app/ISR.c
__root void ISR_Hall(void)	Hall interrupt	S04_app/ISR.c
__root void ISR_MFT_FRT(void)	The MFT zero detect ISR for the motor control	S04_app/ISR.c
__root void ISR_MFT_WFG(void)	The HW over-current ISR	S04_app/ISR.c
__root void ISR_ADC_unit0(void)	The ADC unit0 ISR, trigger at the zero point for the 3 shunts	S04_app/ISR.c
__root void ISR_ADC_unit1(void)	The ADC unit1 ISR for the IPM temperature sample	S04_app/ISR.c
__root void Isr_UartRx(void)	UART receive interrupt by MFS3	S04_app/ISR.c
__root void Isr_UartTx(void)	UART transmit interrupt by MFS3	S04_app/ISR.c
__root void DefaultIRQHandler (void)	MCU exception interrupt	S04_app/ISR.c

8.2 Interrupt Priority Setting

Each interrupt priority can be set by the function ‘void InitNVIC(void)’ which is located at the file ‘S04_app/Initial.c’. Users are not recommended to modify it. The priority used for motor control is shown in Figure 8-1.

```
voidInitNVIC(void)
{
// INT priority
ConfPriorityForIRQ(16 + MFS3RX_IRQn, 4, PRI_LEVEL_6);   //UART receive
ConfPriorityForIRQ(16 + MFS3TX_IRQn, 4, PRI_LEVEL_6);   //UART Transmit
ConfPriorityForIRQ(16 + WFG_IRQn, 4, PRI_LEVEL_0);      //watchdog
ConfPriorityForIRQ(16 + EXINT0_7_IRQn, 4, PRI_LEVEL_0);//outside int
ConfPriorityForIRQ(16 + SWDT_IRQn, 4, PRI_LEVEL_1);//software watch dog
ConfPriorityForIRQ(16 + ADC0_IRQn, 4, PRI_LEVEL_2);     //adc0
ConfPriorityForIRQ(16 + ADC1_IRQn, 4, PRI_LEVEL_4);     //adc1
ConfPriorityForIRQ(16 + FRTIM_IRQn, 4, PRI_LEVEL_3);    //frt
ConfPriorityForIRQ(16 + OUTCOMP_IRQn, 4, PRI_LEVEL_6);  //outcompare
ConfPriorityForIRQ(16 + BTIM0_7_IRQn, 4, PRI_LEVEL_5);  //hall
}
```

Figure 8-1: Interrupt Priority Setting

8.3 Interrupt Generation

The diagram of the interrupt used for the motor control is briefly introduced in this section.

8.3.1 MFT

The multifunction timer is used to generate the interrupt for the motor control algorithm and trigger the AD sample at the zero point.

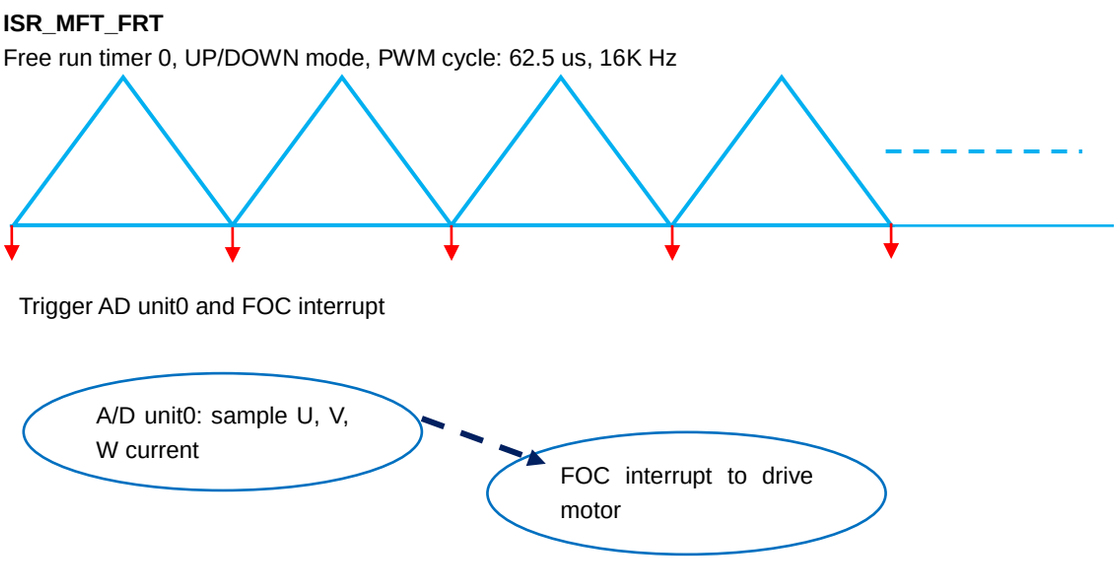

Figure 8-2: Free Run Timer Interrupt

8.3.2 Hall Capture

The PWC timer is used to capture the hall status change and the pulse of the edge of the hall signal.

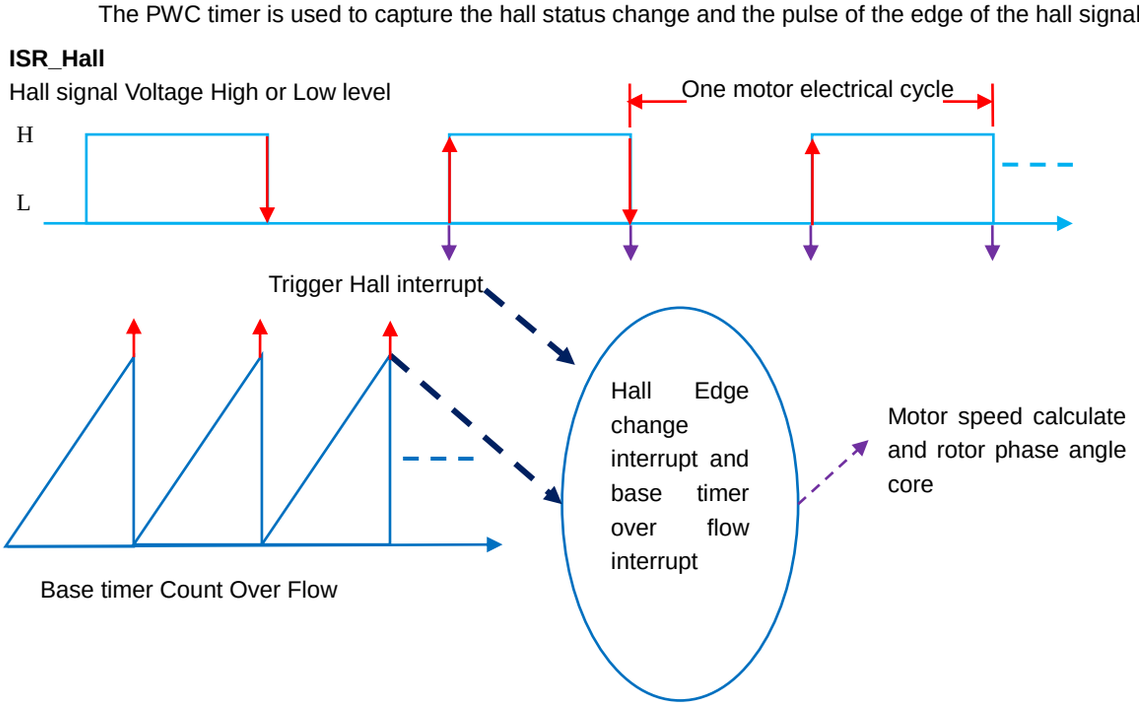

Figure 8-3: Base Timer Interrupt

8.3.3 DTTI

The DTTI0 is used to trigger the HW fault protection from the IPM. When the phase current is large enough to trigger the HW over-current fault, the interrupt is got and all of the drive signals for the motor control will shut off immediately.

ISR_MFT_WFG
IPM fault signal low voltage

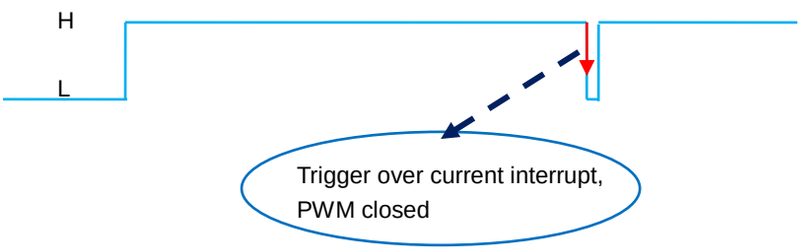

Figure 8-4: DTTI Interrupt

9. Demo System

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

9.1 Demo System Introduction

The sensor washing machine solution can be adaptive to any type of washing machine which uses the PMSM or BLDC motor. The connection diagram for debugger is shown in Figure 9-1.

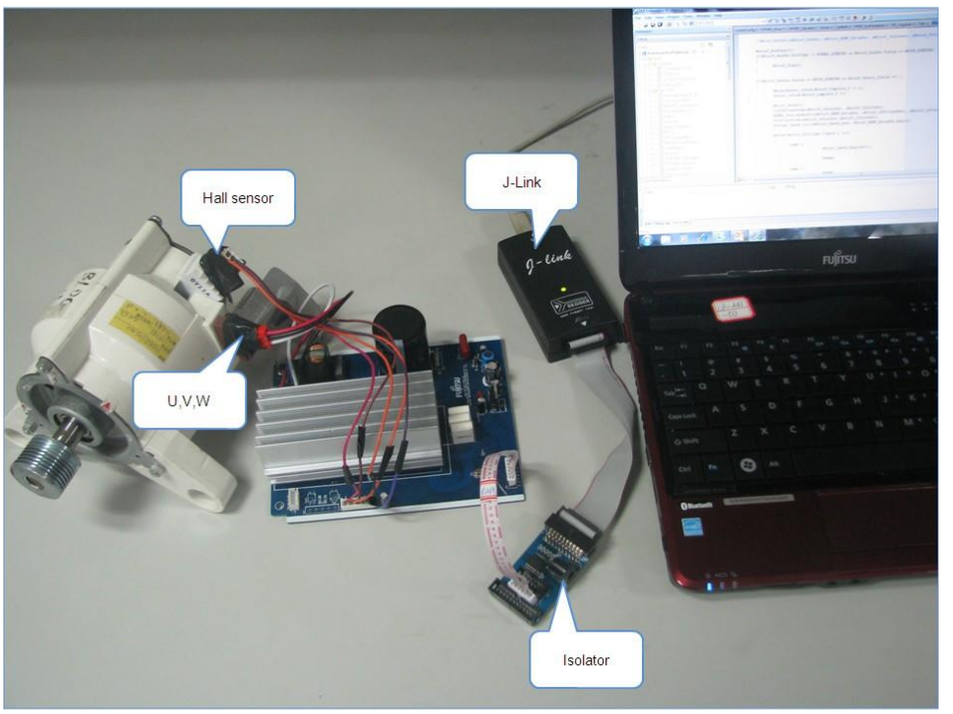

Figure 9-1: System Connection

9.1.1 Hardware Connection

It is necessary to connect below 4 lines:

1. Connect motor's hall signal to inverter board, shown as below.

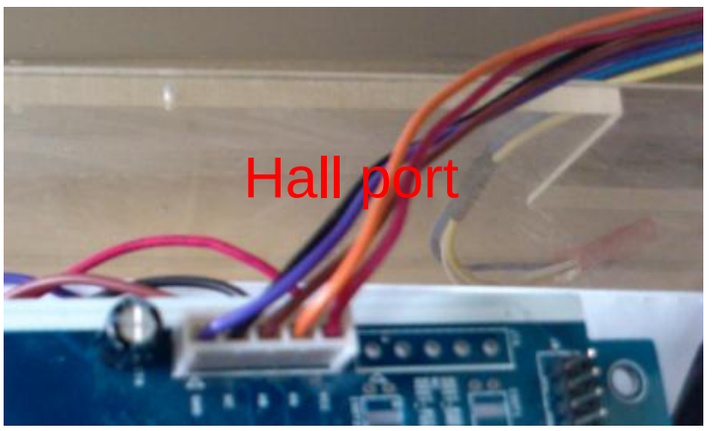

Figure 9-2: Hall Signal Line Connection

The Hall signal line connection is defined in the following table.

Table 9-1: Hall Connection

Motor's line	Inverter board circuit port
Hall A	Hall A
Hall B	Hall B
Hall C	Hall C
+5V	VCC
GND	GND

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 to the Hall C port on the board.
- VCC and GND must be connected rightly, otherwise the hall won't work properly and the motor will also not run.

2. Connect motor's U, V, W phase lines to inverter board, shown as below.

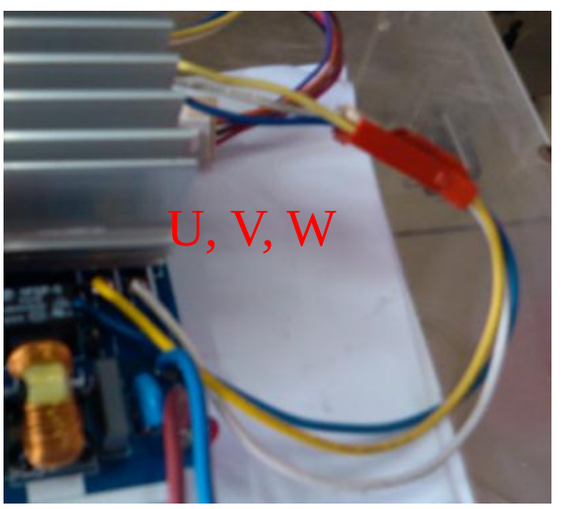

Figure 9-3: Motor Line Connection

Motor's U, V, W line can be optionally connected to Inverter's IPM's output U, V, W.

3. Connect JTAG to Inverter, shown as below.

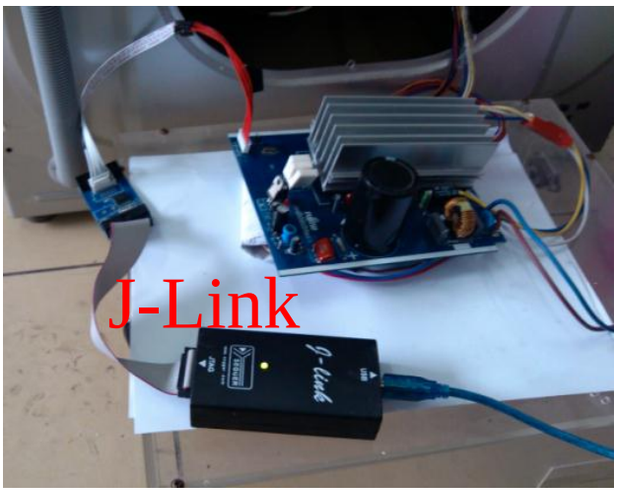

Figure 9-4: JTAG Line Connection

Note:

If there is no isolator between the J-link and the HW, you must unplug the AC power and use the battery of your note book.

4. Connect AC power to Inverter board, shown as below.

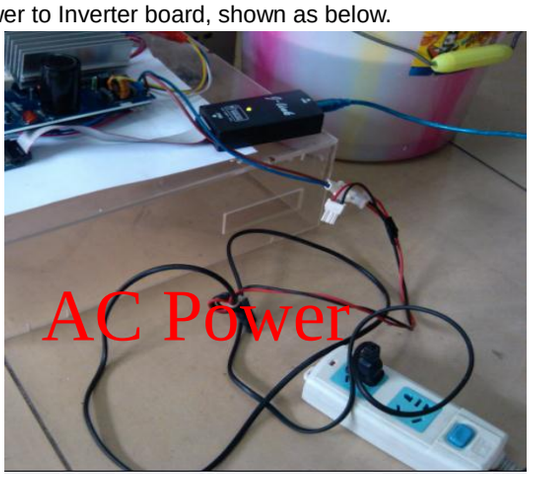

Figure 9-5: AC Plug

9.2 Motor Debug

The debug method on the new motor is described in this section when you finish the hardware connection with the motor.

Click the IAR program to open the IAR, and open the ‘EWW’ file of the inverter washing machine work space as shown in Figure 9-6.

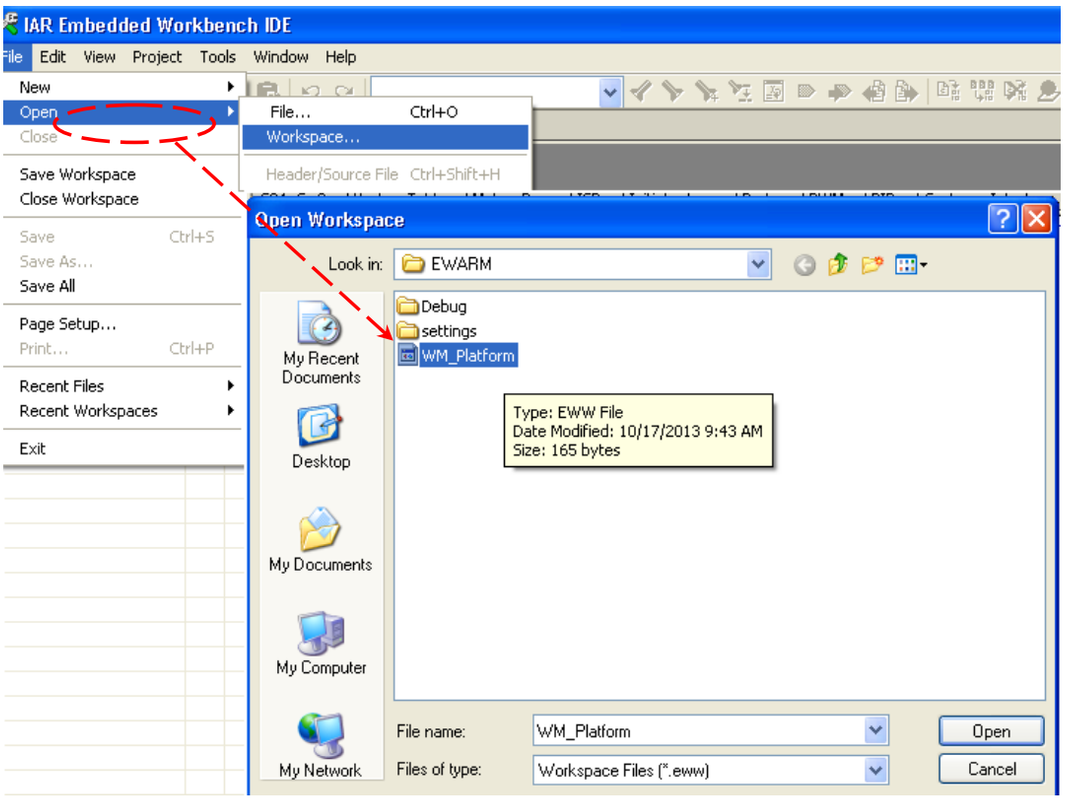

Figure 9-6: Open the Workspace

9.2.1 FW Interface Configuration

All of the variables reserved for the user interfaces are located in the file ‘S05_user/CustomerInterface.c’ and the macro definitions are located in the file ‘H05_user/CustomerInterface.h’. Both files are highlighted, as shown in Figure 9-7

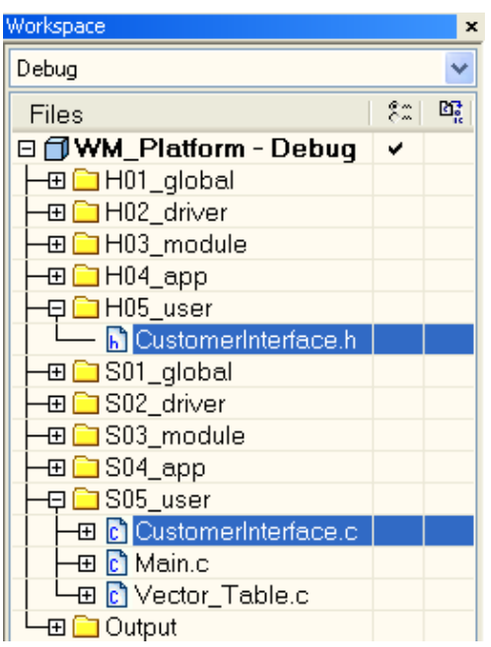

Figure 9-7: Interface File Diagram

9.2.1.1 Basic Setting

The motor can be started easily after basic setting. So the basic variables and macro definitions must be correctly set for the motor demo running.
All of the HW settings in this section must be based on Hardware User Manual.

A. Basic Variables Setting

The basic variables can be set in the c source file 'S05_user/CustomerInterface.c'.

Motor Parameter Configuration

The motor parameter must be correctly set except the hall related parameters that is red highlighted in Figure 9-8. When the hall related parameter is self-checked by hall check module, it must be set correctly according to 9.2.3.3 Motor Parameter.

```
/** UI_0101  configure motor parameter */
#define  MOTOR_ID    0      // motor ID number
                            // 0 --> new motor param,
                            // >=1 -->already debug motor,add at the end of this file
#if 0== MOTOR_ID            // new motor param -->LS BLDC
uint8_t  Motor_pole_pairs = 12;              // the pole pairs of rotor
uint8_t  Motor_HallNumber = THREE_HALL; // 3or 2 hall number of the motor
float    Motor_CurrentMax = 6.0;             //max peak phase current,unit,A
float    Motor_Rs         = 6;       // phase resistor of motor,unit:ohm
float    Motor_Ls         = 24;       // phase inductance of motor,unit:mH
uint8_t  Motor_HallStatuList[7] = {0,3,1,5,4,6,2};// hall status change sequence
int32_t  Motor_HallAngleCCW[7] =
{0,DEGREE(210),DEGREE(90),DEGREE(150),DEGREE(330),DEGREE(270),DEGREE(30)};

int32_t  Motor_HallAngleCW[7] = {0,DEGREE(90),DEGREE(330),DEGREE(30),\
                                    DEGREE(210),DEGREE(150),DEGREE(270)};
#endif
```

Figure 9-8: Motor Parameter Set

WM Parameter Setting

```
/** UI_0102  configure WM parameter */
char_t   WM_cType     = DD;  // washer type:DD,DDM,BLDC,BLDCMjust copy it
float    Wm_TransRate = 1;      // TransRate of washer,DD-->1, BLDC-->TBD
int32_t  WM_MinSpd    = 20;     // min speed of drum,unit:rpm
```

Figure 9-9: Washing Machine Parameter Setting

Note:

- The washing machine type must be correctly set, the drum running direction maybe reversed by the incorrect setting.
- The transmission ratio of the motor for the washing machine must be correctly set also. Otherwise the drum speed will be incorrect. Wm_TransRate = 1 for the DD and DDM washing machine.

Inverter Parameter Configuration

The inverter carrier frequency can be set by the reserved variable, but the variables in this part are not recommended to modify for the washing machine application.

```
/** UI_0103 configure inverter parameter*/
uint16_t Motor_CARRY_FREQ  = 16000;  //carrier frequency of motor driver,unit:Hz
uint32_t RelayDelayOnTms   = 2000;  // time delay for  relay switched on,unit:ms
```

Figure 9-10: Inverter Parameter Set

The carrier frequency of the washing machine is 16 KHz. The phase current AD sample frequency is 16 KHz. And the dead-time of the SVPWM is 2μs

B. Basic setting for HW

The basic settings for the HW can be set in the H file 'H05_user/ CustomerInterface.h'

ADC Port and Coefficient Setting

```
/** UI_0301 ADC port and coefficient set */
#define MOTOR_SHUNT_NUMBER              2       // current sample resistor
#define CURRENT_RS                     0.02  // Iuvw sample resistor,unit:ohm
#define Current_Amplifier_Multiple       10    // Iuvw calculation factor
#define VDC_Amplifier_Multiple           96.0  //Vdc calculation factor

#define DC_V_PIN                        ADC_CH_2// Vdc sample channel
#define MOTOR_U_PIN                     ADC_CH_0// Iu sample channel
#define MOTOR_V_PIN                     ADC_CH_1// Iv sample channel
#define MOTOR_W_PIN                            // Iw sample channel
#define IPM_TEMP_PIN                             ADC_CH_3// IPM temperature
sample channel
```

Figure 9-11: ADC Port Setting

The Demo Board's current sample resistor is 0.02Ω, current amplification factor is 10 times, DC Bus voltage sample factor is 96.

Relay and Other GPIO Setting

```
/** UI_0302 configure relay and other GPIO*/
// Relay port setting
#define RELAY_PORT PORT5
#define RELAY_PIN  PIN2
// other GPIO setting
```

Figure 9-12: GPIO Port Set

Other GIPO used by customer can be set in this part.

Hall I/O and Pin set

```
/** UI_0303 configure hall I/O and Pin    */
#define HALL_A_PORT             PORT4   //Port 4
#define HALL_A_PIN              PIN9    //P49
#defineHALL_A_TIMER             BT_CH_0//timer 0
#defineHALL_A_TIMER_CH  0       //timer 0_0

#define HALL_B_PORT             PORT4   //Port 4
#define HALL_B_PIN              PIN10   //P4A
#defineHALL_B_TIMER             BT_CH_1//timer 1
#defineHALL_B_TIMER_CH  0       //timer 1_0

#define HALL_C_PORT             PORT6   //Port 6
#define HALL_C_PIN              PIN1    //P61
#defineHALL_C_TIMER             BT_CH_2//timer 2
#defineHALL_C_TIMER_CH  2       //timer 2_2
```

Figure 9-13: Hall Port Set

The hall I/O is recommended to connect to the port the same as HW demo due to the special MCU peripheral usage.

Selectable Function Setting

```
/** UI_0304 Function set                                        */
char_t FW_TEST_MODE     FALSE// HW\Hall check set
                              //TRUE: work in debug mode for testing HW and Hall
                              //FALSE: work in normal mode and disable Hall check
```

Figure 9-14: Function Selection

The advanced functions are set in this part, if you want to run the hall and HW check functions, the variable: FW_TEST_MODE can be set to TRUE to run these functions.

9.2.1.2 Advanced Variables Setting

If the motor runs well in any working condition, the settings in this section do not need to be changed. The settings can be changed to improve the corresponding performance of the module.

Advanced Setting for MCU

These parts are not recommended to modify for the inverter washing machine solution in the file‘H05_user/CustomerInterface.h’

MCU Clock Setting

The MCU on the Demo Board is MB9AF111K. The maximum machine frequency is 40MHz.

```
/** UI_0401 MCU clock setting                                   */
#define FREQ_XTAL                       4L // MHz
#define SYS_CLOK                        Main_Frequency_40M
```

Figure 9-15: MCU Clock Setting

A/D Converter Setting

```
/** UI_0402 A/D sample setting                                    */
#define ADC_Digit12
#define ADC_MAX         (1 << ADC_Digit)
#define ADC_REF          5.0
#define MOTOR_ADC_FORWARD_TIME   2

#define AD_OFFEST_MAX_VALUE     200    //100
#define CURRENT_NORMAL_OFFEST    (1 << (ADC_Digit - 1))
#define CURRENT_OFFEST_MAX      (CURRENT_NORMAL_OFFEST +
AD_OFFEST_MAX_VALUE)
#define CURRENT_OFFEST_MIN (CURRENT_NORMAL_OFFEST - AD_OFFEST_MAX_VALUE)
```

Figure 9-16: A/D Converter Setting

Advanced Setting for FW

These variables in this part can be modified if the performance of corresponding module is not so good or you want to change the setting for a different washing machine, and you can find the file ‘S05_user/CustomerInterface.c’.

Motor Start-up and Start/stop Setting

The parameter for the motor start-up and the brake stop to end speed can be set in this part.

```
/** UI_0201  Motor start-up variables setting */
int32_t  Motor_StartSpd   = 20;        // start up drum speed,unit:rpm
float    Startup_InitCur  = 0.2;        //initial startup current, unit:A
float    Startup_IncCur   = 0.2;        //initial startup current, unit:A

/** UI_0202  Motor start/stop setting  */
uint16 t BrakeEndSpdRPM   = 0;          //the brake end speed of drum ,unit:rpm
```

Figure 9-17: Variables Setting for Motor Running

PI Parameter Setting

```
/** UI_0204  PI parameter setting*/
uint16_t PI_SPD_Doing_Cycle = 1;  //speed PI cycle, unit:1ms
float   PI_Spd_Kp_Min = 15;
float   PI_Spd_Kp_Max = 50;
float   PI_Spd_Ki_Min = 0.2;
float   PI_Spd_Ki_Max = 0.5;
float   PI_Idq_Kp_Wash = 11.0;
float   PI_Idq_Kp_Spin = 5.0;
float   PI_Idq_Ki_Wash = 0.03;
float   PI_Idq_Ki_Spin = 0.03;
uint16_t PI_Field_Doing_Cycle = 60; //Field weaken PI cycle, unit:1ms
float   PI_FieldWeaken_Ki_Init = 0.3;
float   PI_FieldWeaken_Kp_Init = 1.0;
```

Figure 9-18: PI Parameter Setting

Field Weaken and Limitation Setting

The minimum field weakening running current, the FOC current and the voltage limit can be set in this part.

```
/** UI_0205  Field Weaken variables setting*/
float    FieldWeaken_IsMin = -0.05; //min current in field weak,unit:A
        //IsMax = Motor_CurrentMax*Limit_IdUsage calculate on-line,unit:A
/** UI_0206  FOC limit setting */
float    Limit_VsUsage    = 1.00;   //DC voltage usage rate for FOC
float    Limit_VdUsage    = 0.98;  //d-axis voltage usage rate for FOC
float    Limit_IsUsage    = 0.95;  //d and q axis current usage rate for the FOC
float    Limit_IdUsage    = 0.8;   //d axis current usage rate in the field weaken
```

Figure 9-19: Field Weaken and Limitation Setting

UART Setting

```
/** UI_0207  UART setting                                          */
uint16_t u16Baudrate     = 2400;       // Baud rate of UART, unit:bps
char_t   cParityEn       = FALSE;      // TRUE -- Parity check enable,FALSE--Disable
char_t   cParitySel      = ODD_NUMBER_PARITY;//ODD_NUMBER_PARITY,EVEN_NUMBER_PARITY
uint16_t u16DataLen       = 8;         //data length, default 8bit
uint8_t  u8StopBitLen     = 1;         //stop bit,default 1bit
uint8_t  u8Direction      = LSB_FIRST; //bit direction
                              //LSB_FIRST -- low bit first,MSB_FIRST -- high bit first
uint8_t  Uart_u8CommErrTime= 6;//time delay for the comm error or resume,unit:s
uint8_t  Uart_u8CommDelay  = 0;//time delay between Rx and Tx switch,unit:ms
// PORT and other macro setting in UART.h
```

Figure 9-20: UART Setting

Speed Setting

```
/** UI_0208  Speed set parameter setting     */
int32_t  Wm_SpinSpd      = 70;  // switch drum speed between wash and spin
state,unit:rpm
float    SpdSet_BaseTime  = 0.1;   //the time unit of the speed change time from
UART,unit:s, range 0~1
uint16_t SpdSet_u16AcceLmt =100;   //maximum acceleration of drum speed,unit:rpm/s
uint16_t SpdSet_u16DeceLmt = 20;   //maximum deceleration of drum speed,unit:rpm/s
```

Figure 9-21: Speed Setting

OOB and Weight Setting

The OOB detect speed and the weight speed can be set in this part.

```
/** UI_0209  OOB parameter setting                        */
uint16_t OOB_u16OobSpd    = 89;     //OOB detect speed
uint16_t OOB_u16OobSpd1   = 90;     //the Second OOB detect speed
uint8_t  OOB_u8StableTime = 4;     //stable run time before OOBDetect stage,unit:s

/** UI_02010  weight parameter setting           */
int16_tWeight_i16WtSpdN1 = 90; //stable running at weight speed n1
int16_tWeight_i16WtSpdN2 = 130; //speed accelerate to n2
char_t   Weight_cEn       = TRUE;//weight function enable
float    Weight_fCoe      = 7.0;        //coefficient of the weight data with DC
bus(BLDC)default acceleration for debug mode
```

Figure 9-22: OOB and Weight Parameter Setting

Un-Stop Setting

```
/** UI_02011  UnStop parameter setting           */
uint8_t UnStopCCW_EleCycle = 10;  //configure the unstop CCW running ele-cycle
uint8_t UnStopCW_EleCycle = 10;//configure the unstop CW running ele-cycle
int16_t i16UnStopSpd = 45;             //configure the unstop running spd
```

Figure 9-23: Un-Stop Parameter Setting

Protection Setting

The protection setting is just the prompt. The detailed information can be found in the FW.

```
/** UI_02012  protect variable setting   ******/
char_t   LockRotorProtectEn = TRUE;
uint32_t LockMinSpd = 10;              //configure the locked min speed: 10r/min
uint32_t LockMaxTime = 4000;    //configure the check lock max time: 4000ms

char_t   DCVoltageProtectEn = TRUE;
uint16_t DCVoltageMax = 400; //configure the over voltage protect value: 400V
uint16_t DCVoltageMin = 200;//150; //configure the under voltage protect value: 150V
uint32_t OverVoltageProtectTime = 50;//200;    //configure the over voltage protect
max time 200ms
uint32_t UnderVoltageProtectTime = 30;//2000;  //configure the under voltage
protect max time 2000ms
uint32_t RecoverVoltageProtectTime = 2000;     //configure the voltage back normal
from error's time 2000ms
```

Figure 9-24: Protection Parameter Setting

9.2.2 HW Check

The HW performance can be self-checked by the HW check module.
If the HW has been already used for a long time, this module can be ignored. And the motor can be normally started as shown in section ‘9.2.4Run Motor.’

Note: The HW performance must be validated and the related FW setting must be also correctly set, otherwise the hall self- check and the motor may not run well.

9.2.2.1 FW Setting

Set the variable FW_TEST_MODE TRUE to 1 to make the control system run in debug mode.
Set the real DC bus that is measured by multi-meter between the PN points on the HW to the macro definition as following:

```
#define  DC_INPUT  310   //the DC input voltage to inverter board in test mode, unit: V
```

9.2.2.2 HW Check Run

Click the debugger button 

to connect the J-link, and paste the global structure '*HwCheck_stcPar'* into the Live Watch in the IAR debug online.
Enable the HW check function by the variable 'cStart' that is shown in Table 9-2, the HW performance such as DC sample and HW over-current point can be self-checked by this function.

Table 9-2: Global Structure for HW Check

HwCheck_stcPar <struct>	
cStart '.' (0x00)	HW check start command
cStop '.' (0x00)	HW check stop command
cOver '.' (0x00)	HW check finished flag
cError '.' (0x00)	Flag for HW check error
cDCError '.' (0x00)	Flag for DC voltage check error
cSampleError '.' (0x00)	Flag for current sample check error
i32Q8_OCPoint 0	Value of the HW over-current protection

When the HW check finished flag 'cOver' is set to '1', the HW check result is output by the global structure as shown in Table 9-2.

9.2.3 Hall Check

When the basic setting has been finished, the hall information can be self-checked by the hall check module if the hall information is not known, take one of the DD motor for top loading washing machine for example. And if the hall information has been known, this section can be ignored and the motor can be normally started and then take the reference at section 9.2.4Run Motor.

9.2.3.1 FW Setting

Set the variable FW_TEST_MODE TRUE according to Figure 9-25 to make the control system run in debug mode

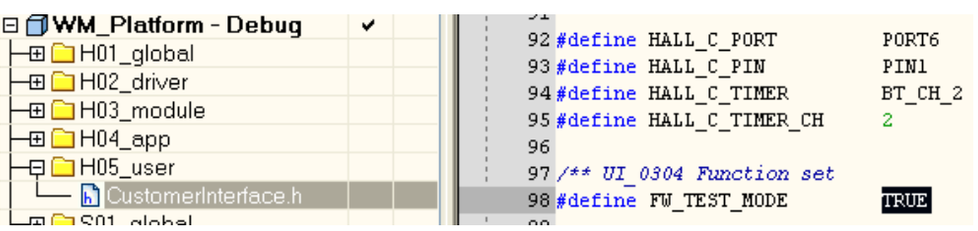

Figure 9-25: Macro Definition for the Test Mode

Set the 'motor pole pairs' according to the motor SPEC from manufacturer to the flowing variable.

```
/** UI_0101 Configuration motor parameter */
uint8_t   Motor_pole_pairs = 12;          // the pole pairs of rotor
float     Motor_CurrentMax = 6.0;         //max peak phase current,unit,A
float     Motor_Rs         = 6;           // phase resistor of motor,unit:ohm
float     Motor_Ls         = 24;           // phase inductance of motor,unit:mH
```

Motor_Rs: This parameter is used for the hall check validation and it can be measured by multi-meter.
Motor_Ls: This parameter is used for the hall check validation.

Set the washing machine parameters according to the following variables.

```
/** UI_0102Configuration WM parameter */
char_t    WM_cType = DD;      // washing machine type: DD, DDM, BLDC, BLDCM, just copy it
float     Wm_TransRate= 1;    // TransRate of washing machine, DD-->1, BLDC-->TBD
```

9.2.3.2 Hall Check Run

Click the debugger button 

to connect the J-link, and paste the global structure '*HallCheck_stcPar*' into the Live Watch in the IAR debug online.

Enable the hall check function by the variable 'cStart' as Table 9-3 and the hall information of the motor can be self-checked by this function. When the hall check finished flag 'cOver' is set to '1', the hall information is output by the global structure as Table 9-3 and Figure 9-26.

Note: the load in the drum must be empty for the hall check.

Table 9-3: Global Structure for Hall Check

HallCheck_stcPar <struct>	
cStart '.' (0x00)	Hall check start command
cStop '.' (0x00)	Hall check stop command
cOver '.' (0x00)	Hall check finished flag
cStage '.' (0x00)	Hall check stage
cError '.' (0x00)	Flag for Hall check error
cNumberError '.' (0x00)	Flag for Hall number error
cStatusError '.' (0x00)	Flag for Hall status error
cTimeOverError '.' (0x00)	Flag for Hall check time-out error
u8HallSensorNumber '.' (0x00)	The hall number of the motor
u8StatusTable ""	The status list table
i32Q22_AngleCCW <array>	The CCW angle table
i32Q22_AngleCW <array>	The CW angle table

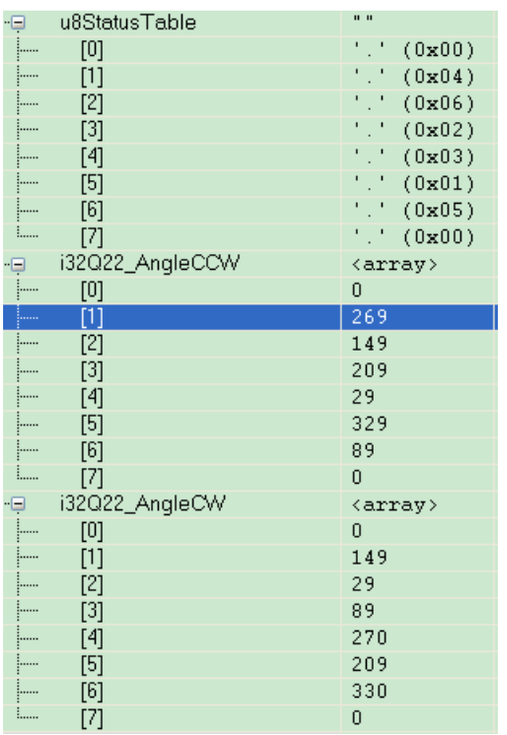

Figure 9-26: Hall Check Result

9.2.3.3 Motor Parameter Configuration

The data output by the hall check function in the Figure 9-26 must be filled into corresponding variables or array for the motor's normal running in the file 'S05 user/CustomerInterface.c' as follows.

```
/** UI_0101  configure motor parameter */
#define  MOTOR_ID    0   // motor ID number
                          // 0 --> new motor parameter,
                          // >=1 --> already debug motor,add at end of this file
#if 0== MOTOR_ID          // new motor param -->LS BLDC
uint8_t  Motor_pole_pairs = 12;              // the pole pairs of rotor
uint8_t  Motor_HallNumber = THREE_HALL; // 3or 2 hall number of the motor
float    Motor_CurrentMax = 6.0;             //max peak phasecurrent,unit,A
floatMotor_Rs        = 6;       // phase resistor of motor,unit:ohm
float    Motor_Ls        = 24;   // phase inductance of motor,unit:mH
uint8_t  Motor_HallStatuList[7] = {0,4,6,2,3,1,5};// hall status change sequence
int32_t  Motor_HallAngleCCW[7] =
{0,DEGREE(270),DEGREE(150),DEGREE(210),DEGREE(30),DEGREE(330),DEGREE(90)};
int32_t  Motor_HallAngleCW[7] = {0,DEGREE(150),DEGREE(30),DEGREE(90),\
                                 DEGREE(270),DEGREE(210),DEGREE(330)};
```

Figure 9-27: Configure Parameter to Test the Hall Phase Angle

MOTOR_ID: The motor ID for user, if the new motor is used for the debug, the motor can be set in the region '#if 0== MOTOR_ID ' and set the MOTOR_ID = 0. If the motor runs well with these motor parameters, these parameters can be fixed and added at the end of the 'S05 user/CustomerInterface.c''. And you can switch the motor debug more conveniently and quickly if you have the debugged parameter

Motor_pole_pairs: It must be got by the manufacturer

Motor_CurrentMax: It can be got by the manufacturer or determined by the phase peak current at the motor brake stable stage

Motor_Rs: The parameter of motor phase resistor, it can be measured by the multi-meter.

Motor_Ls: The parameter of motor phase inductance.

Motor_HallStatuList[7]: The hall status change sequence, it can be self-checked and filled sequentially according to buffer '*HallCheck_stcPar .u8StatusTable[8*]' that is shown in Figure 9-26

Motor_HallAngleCCW[7]: The hall angle matched with each hall status for CCW running, it can be self-checked and filled sequentially according to buffer '*HallCheck_stcPar .i32Q22_AngleCCW[8]*' that is shown in Figure 9-26.

Note:

- Due to the check error, the angle can be set to the integrate number nearby. If 'HallCheck_stcPar .i32Q22_AngleCCW[1]=269' is shown in Figure 9-26, we fill the buffer '*Motor_HallAngleCCW[1]=270*' as Figure 9-27.

Motor_HallAngleCW[7]: The hall angle matched with each hall status for CW running, it can be self-checked and filled sequentially according to buffer '*HallCheck_stcPar .i32Q22_AngleCW[8]*' that is shown in Figure 9-26

9.2.4 Run Motor

When the hall angle and status list have been checked and the HW performance check is correct by the FW TEST MODE, the motor can be started for the demo show.

(1) Reset the variable FW TEST MODE macro definition in 'S05_user/CustomerInterface.c' as following.

```
char_t    FW_TEST_MODE    TRUE      // HW\Hall check set
```

(2) Check the basic motor and HW parameter setting in the user interfaces. If the setting does not match the real HW and washing machine parameter, there will be an unexpected running error in the motor running.

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

①Click button **A** that is shown in Figure 9-28 to connect the J-link and download the FW into the MCU,

②Click button **B** to run the FW online.

③After two seconds the relay is switched on, you can enter none-zero speed value to start the motor in the structure that is shown as **C.**

For example, when the variable 'Motor_stcRunParam.i16WmCommandSpdRpm = 90' by your online input, the drum speed of the washing machine will CCW run to 90rpm.

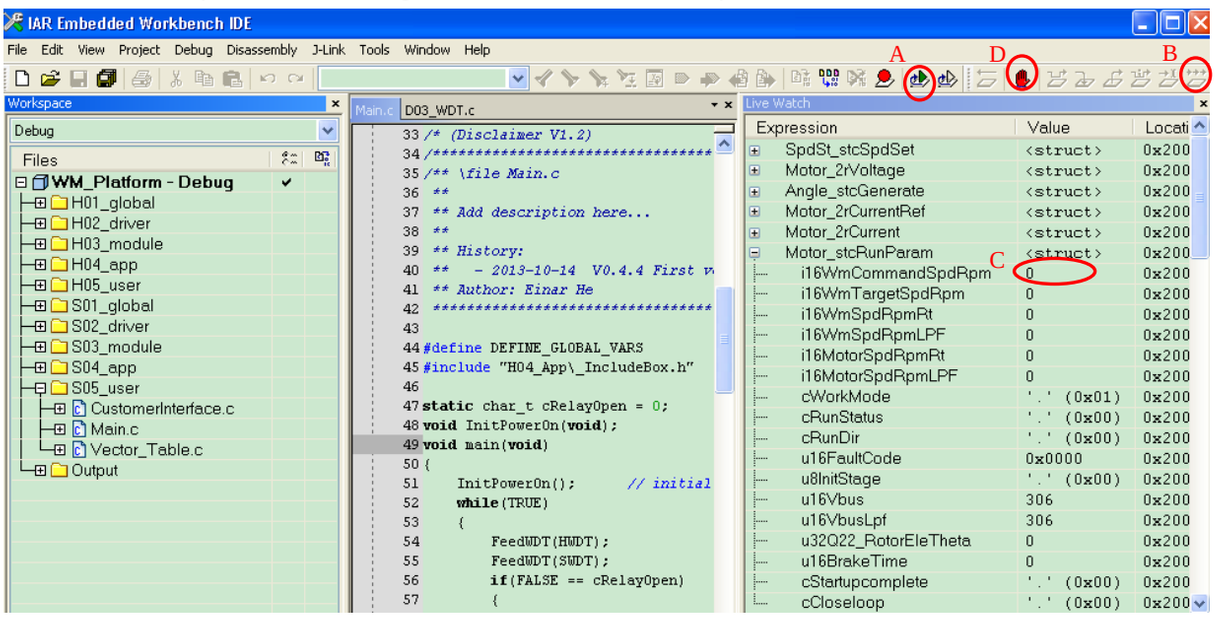

Figure 9-28: Motor Run by J-link

And you can take the Table 9-4 for your detailed reference for the speed command.

Table 9-4: Drum Running Status by the Command Speed

Motor_stcRunParam. i16WmCommandSpdRpm	Drum 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 drum 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.

(4) Watch the important variable to check the motor running performance such as whether the real motor is achieved the command speed and running speed is stable. Detailed meaning about the important variable is shown in the previous section '5.2Global Structure and Variable Definition' for your reference.

9.2.5 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 9-5

Table 9-5: Typical Running Status by the Command Speed

Motor_stcRunParam. i16WmCommandSpdRpm	Drum Direction	Description
30~50	CCW	The Drum speed runs at 30~50rpm for the wash mode
-30~-50	CW	The Drum speed runs at 30~50rpm for the wash mode
89	CCW	The Drum speed runs at 89rpm for the OOB detection before the spin mode
400	CCW	The Drum speed runs at 400rpm for the pre-spin
100	CCW	The Drum speed runs at 100rpm to drain away water by the host after the pre-spin
1000	CCW	The Drum speed runs at 1000rpm for the spin mode
1200	CCW	The Drum speed runs at 1200rpm for the spin mode
0	Stop motor	The motor will stop working

The default speed changing time is 10, which means 1s as shown in 9.2.1.2Advanced Variables Setting.
If you want to change the default acceleration, you can disable the UART macro definition ’UARTEN ’ in UART.h and set the default acceleration ‘*DefaultAcce*’ or the maximum acceleration ‘*SpdSet_u16AcceLmt’* as you want.
When the motor needs to reverse the running direction, you should stop the motor and then restart the motor to run in another direction.

9.3 Troubleshooting

9.3.1 Motor Start-up

When the motor can’t start-up normally, there may be 2 reasons:
(1) The Hall Angle found in debug mode is fault. Even if the washing machine’s load is empty, it also can’t Start-up.
(2) The startup parameter is not set correctly. User should change the parameter in ‘S05_user/CustomerInterface.c’ and refer to the section ‘Motor Start-up’

9.3.2 Protection

When the motor is stopped without the normal stop command, the protection fault may appear, you can see the value of the variable ‘Motor_stcRunParam.u16FaultCode’ in the watch window and the code is assigned by the bit OR operation. The fault codes for each protection are shown as below. You can match the value with these fault codes to find what protection is performed.

```
#define NORMAL_RUNNING          0x0000   //no error
#define OVER_VOLTAGE            0x0001   //DC bus over-voltage
#define UNDER_VOLTAGE           0x0002   //DC bus under-voltage
#define SW_OVER_CURRENT         0x0004   //over-current
#define MOTOR_OVER_CURRENT 0x0008   //over-current of HW
#define MOTOR_LOSE_PHASE        0x0010   //motor lose phase
#define NO_CONECT_COMPRESSOR 0x0020   //no motor connected
#define AD_MIDDLE_ERROR         0x0040   //current sample 2.5V offset error
#define SF_WTD_RESET            0x0080    //FW watch dog reset
#define MOTOR_LOCK              0x0100   //motor lock
#define UNDEFINED_INT           0x0200   //undefined interrupt
#define HW_WTD_RESET            0x0400   //HW watch dog reset
//washing machine fault code
#define POWER_OVER              0x0800   //motor over-power
#define IPM_TEMPOVER            0x1000   //IPM over current
#define HALL_LOST               0x2000   //Hall lost fault
#define COMM_ERROR              0x4000   //communicate error code
#define SINK_ERR                0x8000   //IPM circuit fault
```

There may be different processing logic about the protection.

The fault code may not be cleared except the DC bus voltage protection for the inverter DEMO. That is the FW may not run again when the protection fault happens. You can access the variable 'Motor_stcRunParam.u16FaultCode' to make your own protection processing logic.

9.3.3 Drum Direction Reversed

If the running direction of the drum does not match the requirement of washing machine, there are two possibilities for this trouble.

- The type of washing machine is not correctly set as section WM Parameter Setting. The CCW direction of motor and drum are different from DD or DDM for the belt drive washing machine.
- The U V W of the motor phase is not correctly connected on the corresponding port on the HW. If the motor phase is not correctly connected, the Hall status list and angle must be re-checked and modified to the motor parameter shown in section 9.2.3Hall Check.

9.3.4 High Power Consumption

If the power or the phase current is bigger than other solution, and you can watch the value of d-axis voltage 'Motor_2rVoltage.Q8_d'on the rotating frame at the speed of 45rpm.

If the variable's value is out the range of -10V to 10V, which indicates the rotor angle corrected by the hall may not be so accurate.

You can do as follows:

- The Hall status list and angle must be re-checked and modified to the motor parameter shown in section 9.2.3Hall Check.
- If the power consumption is still higher, you can make the same offset on correct angle for each hall status, the array is 'Motor_HallAngleCCW[7], Motor_HallAngleCW[7]' that is located at file 'S05_user\CustomerInterface.c'. Re-compile the projection and debug at the same working condition, and you can find the best angle list when the power is the best.

10. Additional Information

For more Information on Spansion semiconductor products, visit the following websites:
English version address:
http://www.spansion.com/Products/microcontrollers/

Chinese version address:
http://www.spansion.com/CN/Products/microcontrollers/

Please contact your local support team for any technical question
America: Spansion.Solutions@Spansion.com
China: mcu-ticket-cn@spansion.com
Europe: mcu-ticket-de@spansion.com
Japan: mcu-ticket-jp@spansion.com
Other: http://www.spansion.com/Support/SES/Pages/Ask-Spansion.aspx

AN706-00095-1v0-E

Spansion•Application Note

FM3 Family
32-BIT MICROCONTROLLER
Washing Machine 3-Phase BLDC FOC Control With Hall Sensor User Manual

Jan 2015 Rev. 1.0

Published: Spansion Inc.
Edited: Communications

Colophon

The products described in this document are designed, developed and manufactured as contemplated for general use, including without limitation, ordinary industrial use, general office use, personal use, and household use, but are not designed, developed and manufactured as contemplated (1) for any use that includes fatal risks or dangers that, unless extremely high safety is secured, could have a serious effect to the public, and could lead directly to death, personal injury, severe physical damage or other loss (i.e., nuclear reaction control in nuclear facility, aircraft flight control, air traffic control, mass transport control, medical life support system, missile launch control in weapon system), or (2) for any use where chance of failure is intolerable (i.e., submersible repeater and artificial satellite). Please note that Spansion will not be liable to you and/or any third party for any claims or damages arising in connection with above-mentioned uses of the products. Any semiconductor devices have an inherent chance of failure. You must protect against injury, damage or loss from such failures by incorporating safety design measures into your facility and equipment such as redundancy, fire protection, and prevention of over-current levels and other abnormal operating conditions. If any products described in this document represent goods or technologies subject to certain restrictions on export under the Foreign Exchange and Foreign Trade Law of Japan, the US Export Administration Regulations or the applicable laws of any other country, the prior authorization by the respective government entity will be required for export of those products.

Trademarks and Notice

The contents of this document are subject to change without notice. This document may contain information on a Spansion product under development by Spansion. Spansion reserves the right to change or discontinue work on any product without notice. The information in this document is provided as is without warranty or guarantee of any kind as to its accuracy, completeness, operability, fitness for particular purpose, merchantability, non-infringement of third-party rights, or any other warranty, express, implied, or statutory. Spansion assumes no liability for any damages of any kind arising out of the use of the information in this document.