FM3

SENSOR-LESS WASHING MACHINE FIRMWARE USER MANUAL

32-BIT MICROCONTROLLER
FM3 Family

USER MANUAL

Publication Number FM3_AN706-00096   Revision 1.0   Issue Date Feb 26, 2015

Target products

This application note describes the following products:

Series	Product Number
FM3 Series	MB9AF111K, MB9AF312K

Table of Contents

Figures

Tables

1. Introduction

1.1 Purpose

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

The chapter 2 and chapter 3 describe the hardware and software work environment, which the project should work with IAR 6.4 or an upper version tool. Chapter 4 and chapter 5 introduce the firmware structure and function calling in system. After you have an overall understanding on this system, then you can study more through chapter 5~7 which introduce the timer event function and interrupt time flowchart. In the last chapter, there is a demo show to help user handle a new case when run this system.

1.2 Definitions, Acronyms and Abbreviations

HW	Hardware, at this document it means Inverter platform hardware board
FW	Firmware
FOC	Field Oriented Control
FEE	Fast Back-EMF Estimator
WM	Washing Machine
HFI	High frequency injection
CW	Clockwise
CCW	Counter clockwise

1.3 Document Overview

The rest of document is organized as the following:

Section 2 explains System Scope.

Section 3 explains System Firmware Design.

Section 4 explains System Function.

Section 5 explains Event Function.

Section 6 explains Interrupt Function.

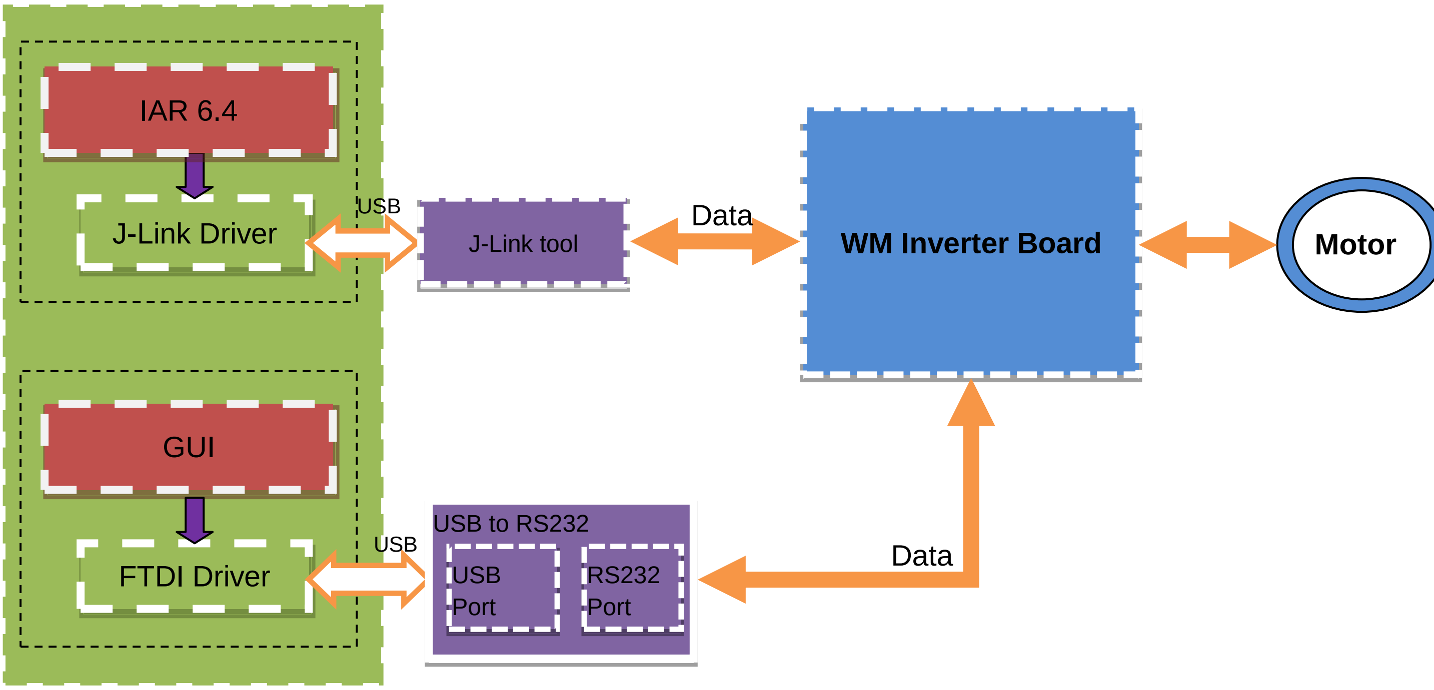
Section 7 explains Demo Show.

2. System Scope

2.1 System Structure

Figure 2-1 shows the whole overview of running system. IAR 6.4 is the main tool to debug and edit FW for your project. GUI is also provided to make debug more easily. When build a new project, you must prepare the IAR tool, J-Link and the motor driving board.

Figure 2-1: System Structure



2.2 System Hardware Environment

Below shows the brief information list of MCU used in wash 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

2.3 System Development Environment

Table 2-1: 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

3. System Firmware Design

3.1 FW Feature

The features of the sensor-less inverter washing machine solution are shown in Table 3-1. All the functions can be found in the demo project. But some core algorithms are made into library. User can set the corresponding variables to enable or disable the function, which will be described in detail in the demo show chapter.

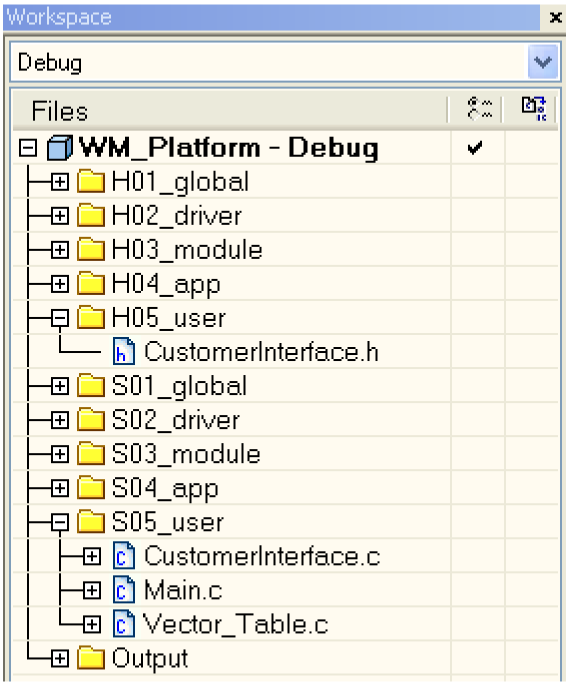
Table 3-1: Feature List of Sensor-less WM Solution

No	Feature	Description	Remark
1.	Adjustable Carrier Frequency	Carrier frequency can be set by the corresponding variable in user interface	
2.	Rotor Position Estimator	Rotor electrical phase angle was corrected by the FEE estimator	
3.	Motor Speed Calculate	Calculate speed by the FEE estimator	
4.	Field Weaken Control	Run motor in field weaken area to raise speed	
5.	FOC Control	Using FOC control algorithm	
6.	Self-adaption Start Up	Adaptive to different load to start-up motor	
7.	High Frequency Injection	The rotor initial position can be checked by High Frequency Injection algorithm which could shorten the start-up time	
8.	Parameter Self Check	Motor's stator resistor can be measured during the startup process and d/q inductor can be measured in the debug process.	
9.	Speed regulate	This function is used to accelerate motor speed and decelerate motor speed by the command from host via UART or debugger	
10.	Brake	Stop motor by brake down	
		Down motor's speed by brake function	
11.	Current Sample	Dual shunts sample	
		Single shunt sample algorithm	
12.	Protect	DC voltage protect	
		A/D offset protect	
		Lock rotor protect	
		Power protect	
		IPM temperature protect	
		Motor phase lost protect	
		Over Current Protect	
13.	OOB	Out of balance (OOB) load detect	
14.	Weight	The weight of the load detect	
15.	Un-Stop Running	Motor can switch running direction (CCW and CW) without stopping motor	
16.	UART	Receive and transform data to Host PC	

3.2 FW Structure

There are 5 layers in the FW structure of IAR, which is shown in Figure 3-1.

Figure 3-1: Structure of FW



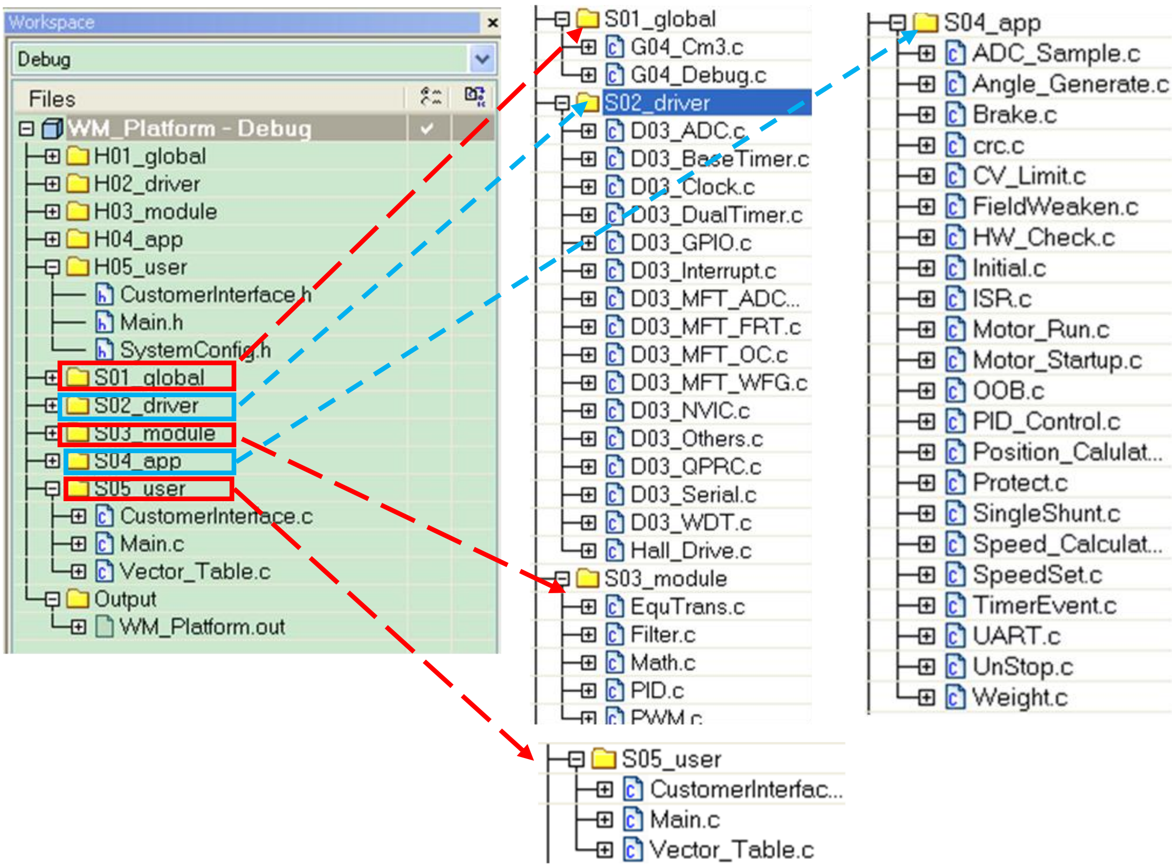
The C source and Header files which are included in each layer are shown in Table 3-2

Table 3-2: 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 , SVM, math, PID, filter
app	*H04_app, S04_app*	Application folder for the files of application function such as speed and position generator by FEE, protection, motor start-up, filed weaken, brake, weight, OOB, UART, etc.
user	*H05_User, S05_User*	Customer interface folder for the files for motor configure and HW setting

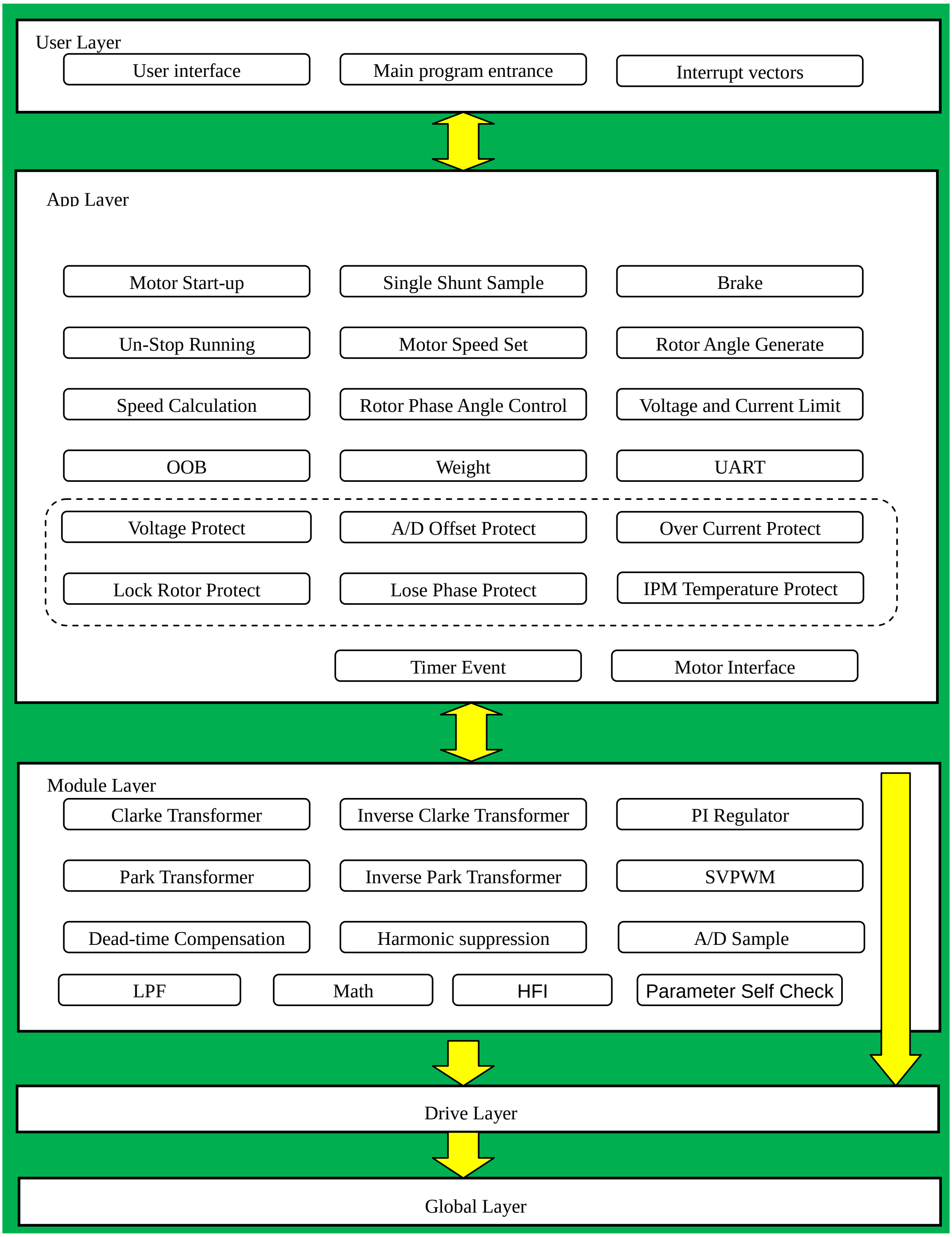
The sub-files in each folder are shown in Figure 3-2, and the structure of header files is the same as C files.

Figure 3-2: Sub-files in Each Layer



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

Figure 3-3: Sensor-less WM FW Architecture



3.3 Files Description

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

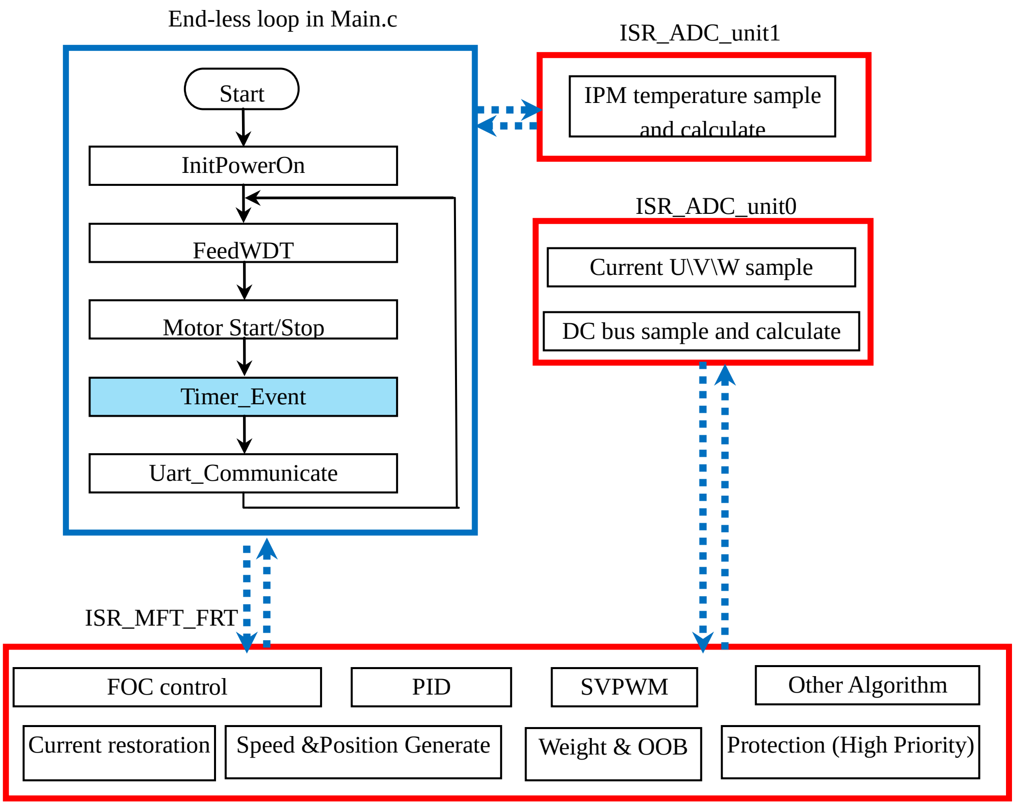
Table 3-3: File Description of Project

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,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 generate 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
	HW_Check.c	HW Check module
	Initial.c	MCU system initialization include 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 set module
	Timer_Event.c	Timer event module
	UART.c	The UART module
	UnStop.c	The Unstop running module
	Weight.c	The electrical weighing module
S02_Driver	*Ignored*	
S05_User	*CustomerInterface.c*	The motor parameter setting
	Main.c	Main function
	Vector_Table.c	MCU interrupt vector list

3.4 FW Control Flow

The control flow for the motor control is shown as Figure 3-4. 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 3-4: Diagram of the Control Flow



4. System Function

This chapter will introduce the system function of the macro definition, global structure definition, and function definition

4.1 Macro Define

The macro definition for the user will detailed describe in the last section '7.2.1FW Interface '

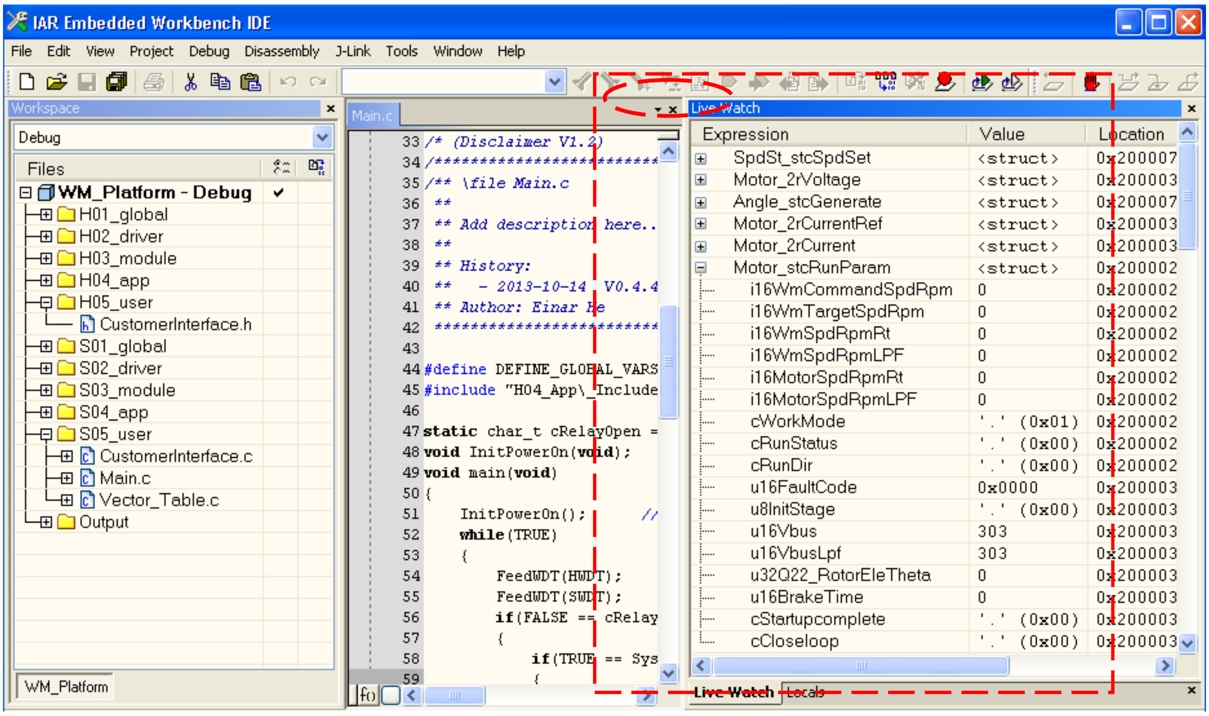
4.2 Global Structure and Variable Define

Common used structure and variables that can be used for the motor running status debug will be detailed listed in this section.

The variable for user interface can be found in section '7.2.1FW Interface

Any structure or variable that you want to watch can be pasted into the 'Live Watch' window of IAR as shown in Figure 4-1.

Figure 4-1: Diagram of Live Watch



4.2.1 Variable 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. The detailed information can be found in the comments for each variable.

```
typedef struct
{
        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 washer
        int16_t i16WmSpdRpmLPF; //the filtered drum speed of washer
        int16_t i16MotorSpdRpmRt;  //the real-time motor speed of washer
        int16_t i16MotorSpdRpmLPF;//the filtered motor speed of washer
        char_t   cWorkMode;     //wash or spin work mode
        char_t   cRunStatus;    //run status: 0--stop,1--Run
        char_t   cRunDir;       //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_t   cStartupcomplete; //flag for motor startup finish
        char_t   cCloseloop;      //flag for the motor closed loop running
} stc_MotorRunParam_t;
extern stc_MotorRunParam_t Motor_stcRunParam;
```

SpdSt_stc

The structure is used to the drum speed set. 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.

```
typedef struct 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
```

4.2.2 Variables for FOC

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

D&Q axis Current and Voltage

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

Motor_2rVoltage	<struct>	Voltage value on the 2 axis rotation frames
Q8_d	0	Real-time voltage on D-axis 'Vd'
Q8_q	0	Real-time voltage on Q-axis 'Vq'
Q12_Cos	2062	Cosine value of the rotor position used for the frame transform
Q12_Sin	3538	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 be also changed at same direction.

Motor_Offset	<struct>	Motor_Offset
U	2073	AD middle point for current Iu AD sample
V	2040	AD middle point for current Iv AD sample
W	0	AD middle point for current Iw AD sample
		2048 = 2.5 V, 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	'.' (...	Flag for motor startup complete,1→start finished
cClosedLoop	'.' (...	Flag for motor closed loop running,1→speed closed loop
cRunStage	'.' (...	Flag for the motor startup stage
cRunLevel	'.' (...	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	D-axis voltage limit
i32Q8_VqLmt	46192	Q-axis voltage limit
i32Q8_IdLmt	0	D-axis current limit, especially in field weaken
i32Q8_IqLmt	0	Q-axis current limit
i32Q8_IsLmt	0	Saturate phase current

FieldWeaken_stcCtrl

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

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

4.2.3 Variables for Speed and Position

Angle_stcGenerate

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

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

Spd_stcPar

The structure is used for rotor speed calculation output. 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 ele-speed

4.2.4 Variables for PID Control

The structures used for PID control are introduced at 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

4.2.5 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—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

4.3 Function List

The functions for the system control are shown in Table 4-1.

Table 4-1: System Function List

Prototype	Description	Remark
void main(void)	Main function of the whole projection	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

5. Event Function

The functions for the motor control that are called in the MFT interrupt 'Motor_Process()' and timer 'Timer_Event()' are shown in Table 5-1 and Table 5-2 ,

5.1 Motor FOC Run Process Function

Table 5-1: Event Function List in the 'Motor_Process()

Prototype	Description	Remark
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_SensorLessMotor ()	The motor start-up function for the sensor-less 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_OpenPhase(…)	The protection function for the open phase detect	

5.2 System Timer Event

Table 5-2: Event Function List in the 'Timer_Event()'

Prototype	Description	Remark
SpdSt_Function(…)	The speed set function used for the motor speed acceleration or deceleration	1ms timer
SpdSt_PIReg(…)	The speed regulation function for the middle speed generation	
FieldWeaken_Control()	The main function for the field weaken	
SpeedDownControl()	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_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	

6. Interrupt Function

6.1 Interrupt Function List

Table 6-1: 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_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

6.2 Interrupt Priority Set

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 the file. The priority used for motor control is shown as below.

```
void InitNVIC(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
}
```

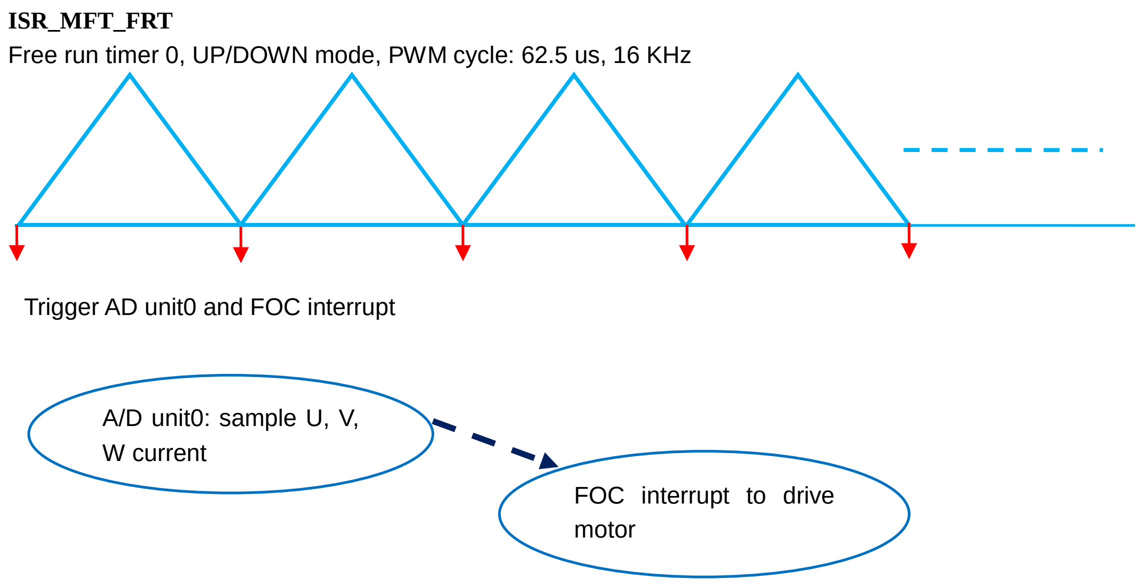
6.3 Interrupt Generate Timer Flow

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

6.3.1 MFT & A/D Interrupt Generate Flow

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

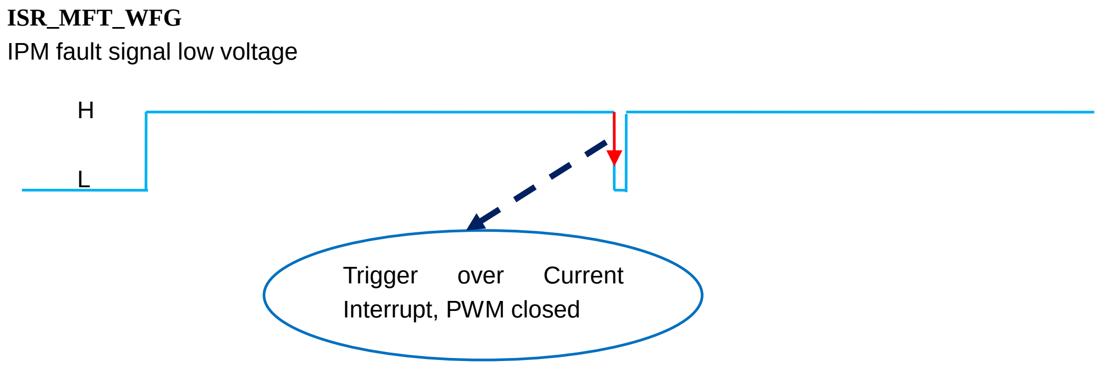
Figure 6-1: Free Run Timer Interrupt



6.3.2 DTTI Generate Flow

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.

Figure 6-2: DTTI Interrupt



7. Demo Show

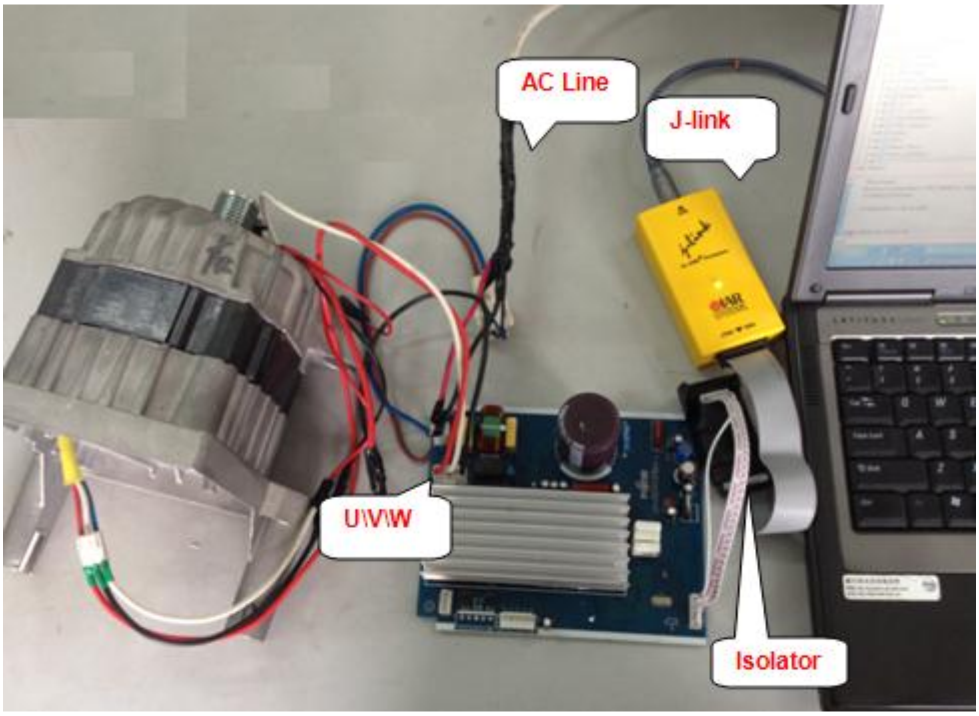
The primary steps are shown as following:

- Hardware Connect
- FW Interface
-
-
- HW Check
- Run Motor
- Speed Acceleration and Deceleration

7.1 Demo System Introduction

The sensor-less wash 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 7-1.

Figure 7-1: System Connection

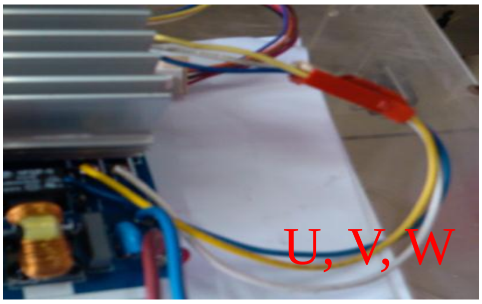


7.1.1 Hardware Connection

It is necessary to connect the 3 lines shown as following:

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

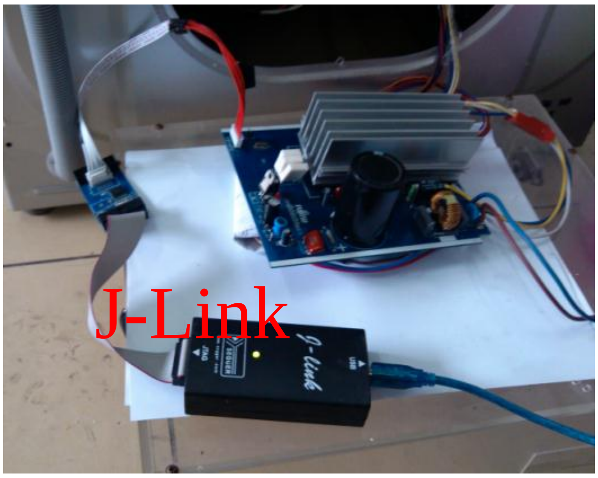
Figure 7-2: Motor Line Connection



Motor's U, V, W lines can be connected to Inverter's IPM's output U, V, W port. And it is also recommended to connect the U, V, W lines according to the definition of the motor.

2. Connect JTAG to Inverter, shown as below.

Figure 7-3: 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.

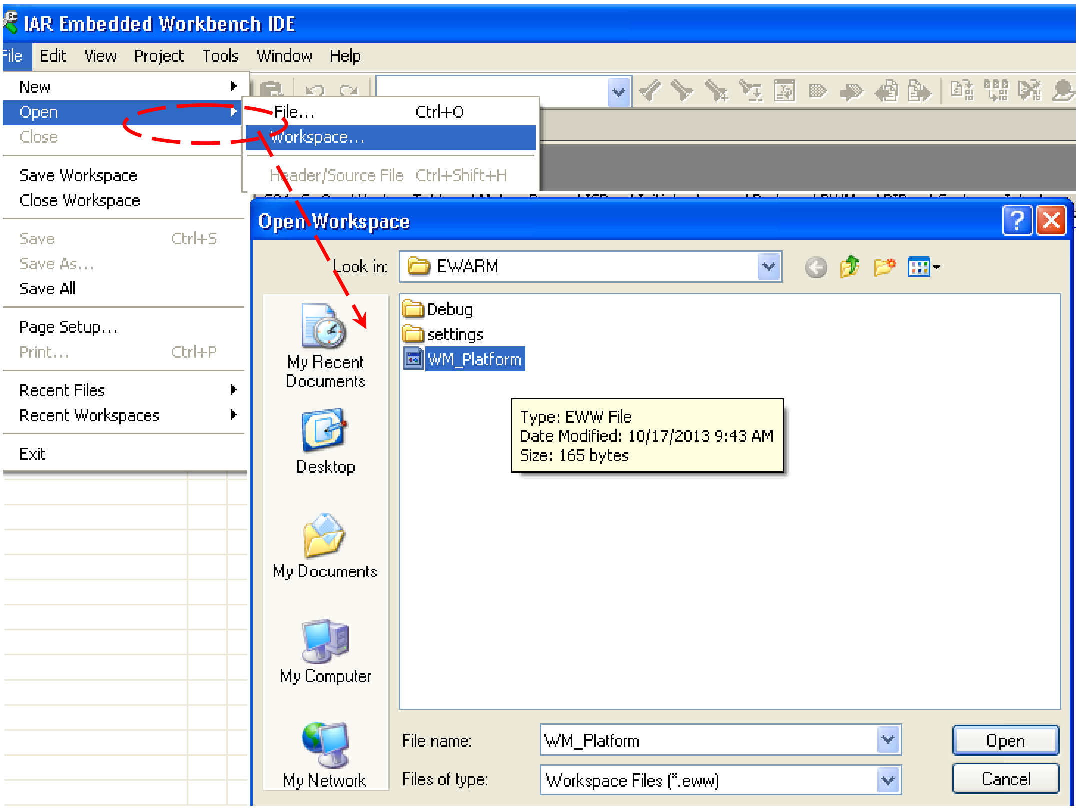
3. Connect the AC line for the inverter board as shown in Figure 7-4 .

Figure 7-4: AC Plug

7.2 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 workspace at the location you've stored on your computer as shown in Figure 7-5.

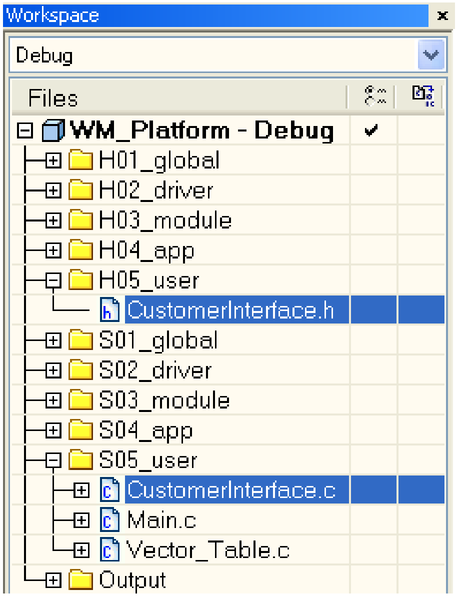
Figure 7-5: Open the Workspace



7.2.1 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 7-6.

Figure 7-6: Interface File Diagram



7.2.1.1 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 the HW design of HW user manual.

A. Basic Variables Setting

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

Figure 7-7: Motor Parameter Set

```
/** UI_0101  configure motor parameter */
#define  MOTOR_ID    0   // motor ID number
                                // 0 --> new motor param,
                                // >=1 -->already debugged motor.
#if 0== MOTOR_ID                        // new motor param -->LS BLDC
uint8_t  Motor_pole_pairs = 12;         // the pole pairs of rotor
float    Wm_TransRate  = 1;     // TransRate of washer,DD-->1, BLDC-->TBD
float    Motor_CurrentMax = 6.0;        //max peak phase current,unit,A
float    Motor_Rs        = 2.1;  // phase resistor of motor,unit:ohm
float    Motor_Ld        = 17.5;
float    Motor_Lq         = 22.5;
float    Motor_EsMin      = 6.0;  //the most min spd may be at 20r/min
#endif
```

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 parameters.

Motor_pole_pairs: it must be got by the motor 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: phase resistor of motor, unit: ohm. It can be measured by the multi-meter.

Motor_Ld: d-axis inductance of the motor, unit: mH.

Motor_Lq: q-axis inductance of the motor, unit: mH.

Wm_TransRate: The transmission ratio of the motor for the washing machine must be also correctly set, It is recommended to set the Wm_TransRate =10 if the max running ele-frequency of motor >1000Hz. That means the motor mechanical speed is reduced by 10 times to make other configuration parameters more robust.

The WM Parameter can be configured as Figure 7-8.

Figure 7-8: Washing machine Parameter Setting

```
/** UI_0102  configure WM parameter */
char_t   WM_cType     = DD;             // wahser type:DD,DDM,BLDC,BLDCM
int32_t  WM_MinSpd     = 30;            // min speed of drum,unit:rpm
int32_t  WM_MaxSpd     = 2000;          // max speed of drum,unit:rpm
```

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.

Figure 7-9: Inverter Carrier Frequency Setting

```
/** 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
```

The carrier frequency for washing machine motor on the demo Board's is 16 KHz. The current sample frequency is 16 KHz. And the dead-time of the SVPWM is 2us.

B. Basie Setting for HW

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

Figure 7-10: ADC Port 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
```

The Demo Board's current sample resistor is 0.02Ω, current OP's 10 times, DC Bus voltage sample factor is 96. Relay and other GPIO settings are shown in Figure 7-11.

Figure 7-11: GPIO Port Setting

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

Firmware can work in debug mode to check whether the hardware works properly. This macro is defined in CustomerInterface.h, as shown in Figure 7-12.

Figure 7-12: Function Select

```
/** UI_0304 Function set                                       */
#define 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
```

The advanced functions are set at this part, if you want run the HW check function, the macro FW_TEST_MODE can be set to TRUE to run these functions.

7.2.1.2 Advanced Variables Setting

If the motor runs well in any working condition, the settings in this section do not need to change.

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.

Figure 7-13: MCU Clock Setting

```
/** UI_0401 MCU clock setting                                                    */
#define FREQ_XTAL                              4L // MHz
#define SYS CLOK                               Main Frequency 40M
```

A/D Converter Setting

Figure 7-14: A/D Converter Setting

```
/** UI_0402 A/D sample setting                                                  */
#define ADC_Digit                             12
#define ADC_MAX                       (1 << ADC_Digit)
#define ADC_REF                    5.0
#define MOTOR_ADC_FORWARD_TIME    2

#define AD_OFFEST_MAX_VALUE      200    //100
```

Advanced Setting for FW

These variables in this parts 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 them in the file'S05_user/ CustomerInterface.c'.

Motor Start-up and Start/stop Setting

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

Figure 7-15: Variables Setting for Motor Running

```
/** UI_0201  Motor start-up variables setting */
int32_t  Motor_StartSpd   = 25;        // start up drum speed,unit:rpm
float    Startup_InitCur  = 0.2;        //initial startup current, unit:A
float    Startup_IncCur  = 0.08;        //initial startup current, unit:A
float    Startup_SwitchCur= 0.5;        //the initial current of at close loop
                        //the times of Motor_CurrentMax,range 0~1, unit:A
uint16_t Startup_PreOrtTime = 100;      //pre-oriente time,unit:ms
uint16_t Startup_OrtTime = 500;         //oriente time,unit:ms
uint16_t Startup_ForceTime = 512;       //force running time,unit:ms
uint16_t Startup_StableTime = 100; //stable running time after the force
running,unit:ms
uint8_t  Startup_RunLevel = CHANGE_SPEED;
```

PI Parameter Setting

Figure 7-16: PI Parameter Setting

```
/** UI_0204  PI parameter setting*/
uint16_t PI_SPD_Doing_Cycle_Wash = 8;  // 8*62.5us
uint16_t PI_SPD_Doing_Cycle_Spin = 16;  // 1ms
float   PI_Spd_Kp = 5;
float   PI_Spd_Kp_Min = 10;
float   PI_Spd_Kp_Max = 55;
float   PI_Spd_Ki_Min = 0.02;
float   PI_Spd_Ki_Max = 0.5;
float   PI_Spd_Kp_Spin = 4;
float   PI_Spd_Ki_Spin = 0.02;

float   PI_Idq_Kp_Wash = 20.0;
float   PI_Idq_Kp_Spin = 10.0;
float   PI_Idq_Ki_Wash = 0.03;///0.03;
float   PI_Idq_Ki_Spin = 0.03;
```

Field Weaken and Limitation Setting

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

Figure 7-17: Field Weaken and Limitation Setting

```
/** 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
```

UART Setting

Figure 7-18: 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
```

Speed Setting

Figure 7-19: 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
```

OOB and Weight Setting

Figure 7-20: OOB and Weight Parameter Setting

```
/** 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 OOB Detect stage, unit:s

/** UI_02010  weight parameter setting           */
int16_t  Weight_i16WtSpdN1 = 90;   //stable running at weight speed n1
int16_t  Weight_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
```

Un-Stop Setting

Figure 7-21: Un-Stop Parameter 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
```

Protection Setting

Figure 7-22: Protection Parameter Setting

```
/** 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; //configure the under voltage protect value: 150V
uint32_t OverVoltageProtectTime = 50; //configure the over voltage protect max time
200ms
uint32_t UnderVoltageProtectTime = 30;//configure the under voltage protect max time
2000ms
uint32_t RecoverVoltageProtectTime = 2000; //configure the voltage back normal from
error's time 2000ms
```

7.2.2 HW Check

The HW performance can be self-checked by the HW check module.

If the HW has been used for a long time, this module can be ignored. And the motor can be normally started as shown in section 7.2.2.3 Run 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.

7.2.2.1 FW Setting

Set the macro '#define FW_TEST_MODE TRUE' to make the control system run as 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

7.2.2.2 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 7-1. The HW performance such as DC sample and HW over-current point can be self-checked by this function.

Table 7-1: 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 7-1.

7.2.2.3 Run Motor in Normal Mode

When the setting parameter especially the Basic Setting parameters have been finished. The motor can be started for the demo show.

(1) Reset the FW TEST MODE macro definition in 'H05_user/ CustomerInterface.h' as following:

#define FW_TEST_MODE FALSE // HW¥Hall check set

(2) Check the basic motor and HW parameter setting in the user interfaces. If the setting does not match with 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.

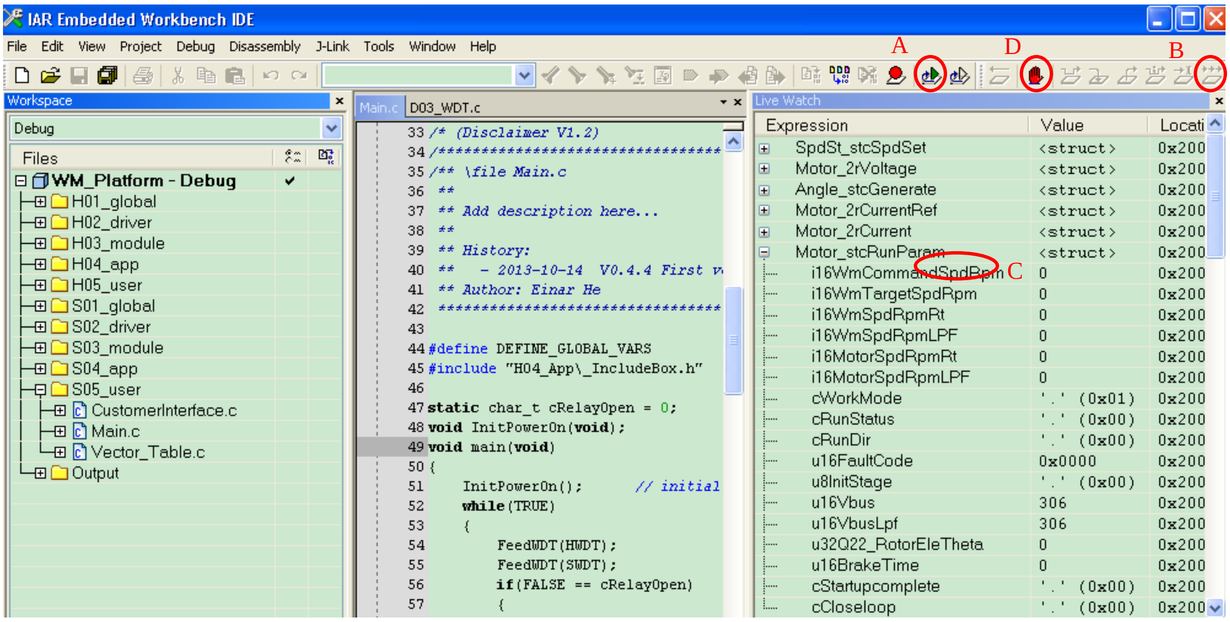
①Click button A that is shown in Figure 7-23 to connect the J-link and download the FW into the MCU,

②Click button B to run the FW online.

③When the relay is switched on about 2 seconds later, you can enter the 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 7-23: Motor Run by J-link



And you can take the Table 7-2 for your detailed reference for the speed command.

Table 7-2: 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 for your reference.

7.2.3 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 7-3.

Table 7-3: 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 7.2.1.2 Advanced 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.

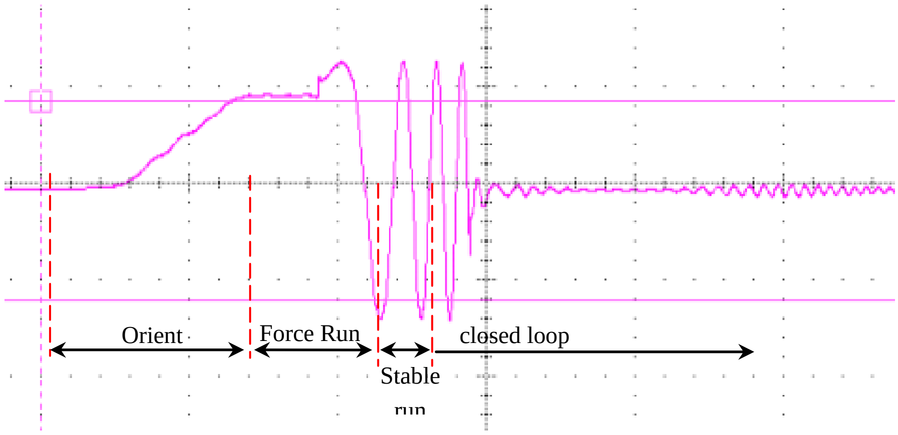
When the motor needs to reverse the running direction, you should stop motor and then restart the motor to run in another direction.

7.3 Troubleshooting

7.3.1 Motor Start-up

When the motor can't start-up normally, you can modify the related interface to improve the start-up performance, the reference is in section 'Motor Start-up'

Figure 7-24: Motor Start-up Diagram



The common abnormal start-up is shown as below:

● The orient time is too long or short, you can modify the interface 'Startup_PreOrtTime' and ' Startup_PreOrtTime' till the performance meets your requirement.

● The force running time is too long or short, you can modify the interface 'Startup_ForceTime ' till the performance meets your requirement.

●The rotor speed and the phase current over-shoot greatly at the closed loop, you can modify the interface 'Startup_SwitchCur' till the performance meets your requirement. The initial current at close-loop is Startup_SwitchCur*Motor_CurrentMax, so 'Startup_SwitchCur' is between 0~1.

● If you want to change the force running speed at start-up, you can modify the interface 'Motor_StartSpd' as you want.

7.3.2 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 code for each protection is shown as below and it is located at file ''H05_user/ CustomerInterface.h''. You can match the value with these fault codes to find which protection happened.

```
#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
#define IPM_TEMPOVER              0x1000   //IPM over current
#define COMM_ERROR                0x4000   //communicate error code
```

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.

7.3.3 Drum Direction Reversed

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

The running direction of the motor is different from the belt drive washing machine. You can change the value of this variable to make the motor run the right direction as you want.

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, you can change any one of the phase line with another.

7.3.4 PI Parameter

If the speed can't be stable at the command speed, all of the PI parameters and the cycles of the PI regulator can be modified in the 'CustomerInterface.c'.

Each of the PI parameters can be modified on line due to the PI parameter changeable function 'void PID_ParameterChange(void)' that is located in file 'PID_Control.c' has been masked in 1ms timer event at file 'Timer_Event'. The PI parameters can be fixed into this function when the PI parameters are fine tuned.

8. 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

9. Reference Documents

[1]. AN_104_FTDI_Drivers_Installation_Guide_for_WindowsXP(FT_000093).pdf: FTDI device driver installation guide for Windows XP.

[2]. AN_119_FTDI_Drivers_Installation_Guide_for_Windows7.pdf: FTDI device driver installation guide for Windows 7.

[3]. AN_234_FTDI_Drivers_Installation_Guide_for_Windows_8.pdf: FTDI device driver installation guide for Windows 8.

AN706-00096-1v0-E

Spansion • Application Note

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

Feb 2015 Rev. 1.0

Published: Spansion Inc.
Edited: Communications