BLDC/DD Washer Debugging Manual

3-phase Motor Control MCU
EU6815L

Fortior Technology Co., Ltd

Content

1 Overview

This debugging manual discusses how to use Fortior EU6815L chip to control BLDC/DD washing machine senseless FOC drive through Fortior dedicated DEMO board. When reading the manual, the second chapter of hardware principles and the third chapter of software principles can be roughly first explored, focusing on the fourth chapter debugging steps.

Software and Hardware Involved

Software/ hardware	Name	Related Chapters	Notes
Software	FU-AM-EU6815-B-095-SW-V1.0.00-20221215	All	Debugging is conducted on this software
Hardware	FU-AM-EU6815-B-095-HW-V1.0.00-20221215	All	Debugging is conducted on this hardware

2 Hardware Schematics and Parameters Configuration

2.1 Hardware Physical Drawing and Schematic Diagram

Figure 2-1 Hardware Physical Drawing

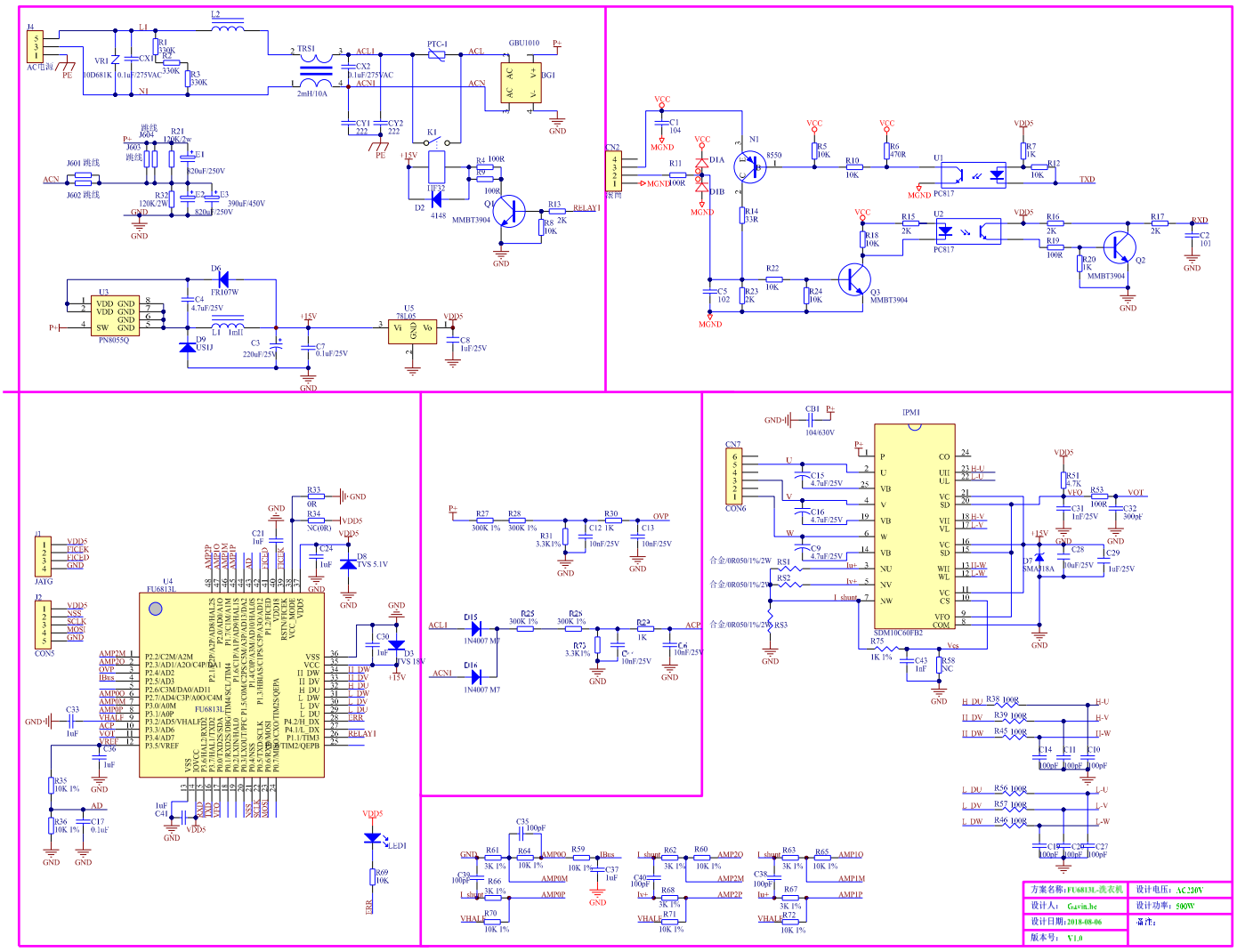


Figure 2-2 Hardware Schematic Diagram

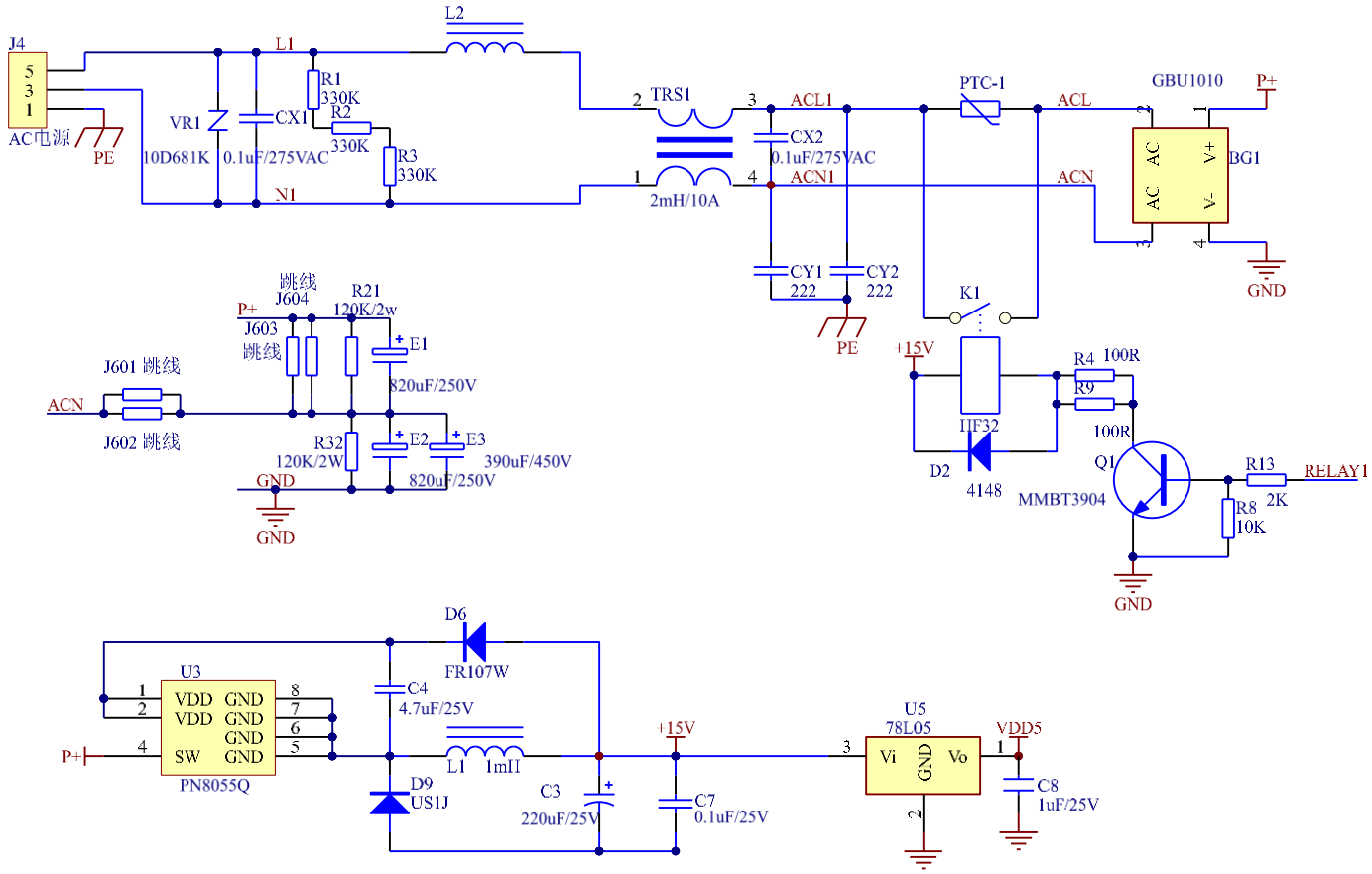
Directions for use:

It is a DEMO board dedicated for washing machine applications, which can be operational after power on.

Notes:

The bus voltage divider ratio, Op-amp gain and shunt resistance should be selected properly, depending on the motor type.

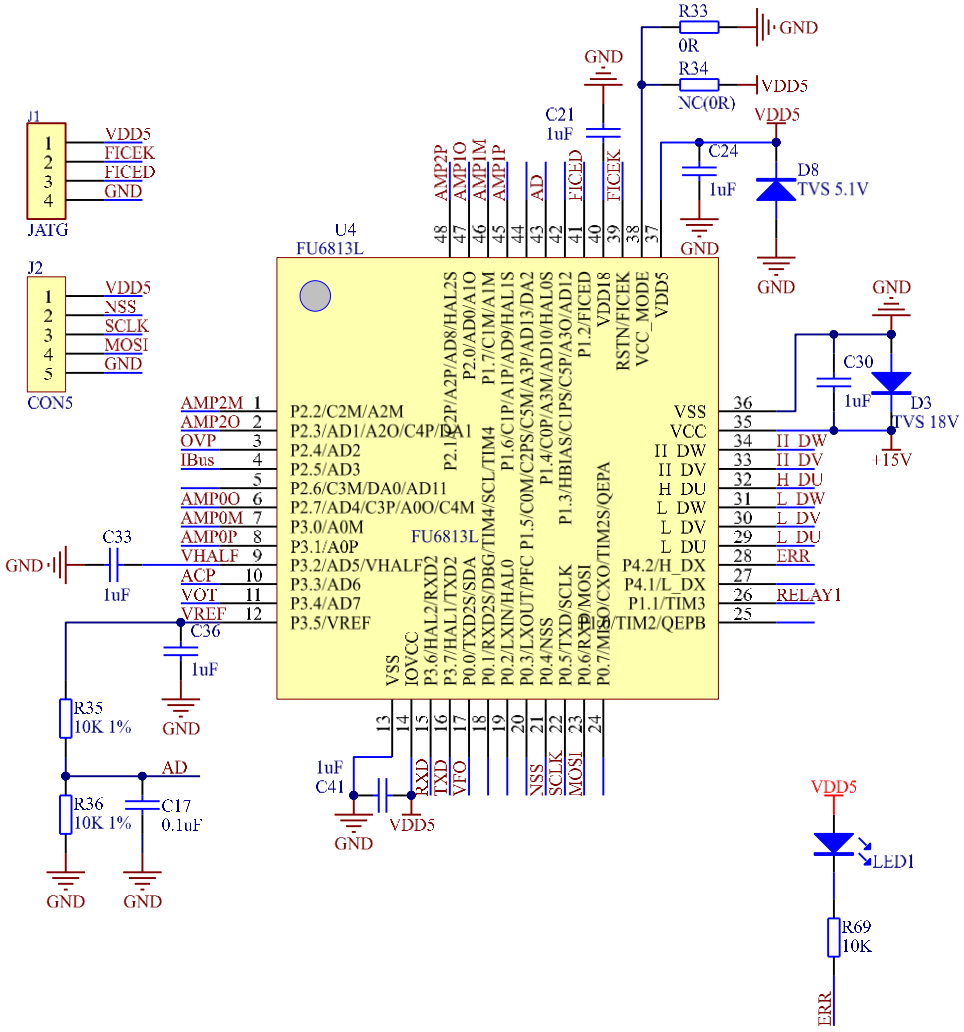
2.1.1 Power Circuit



Power Usage:

Connect DC source positive cable to AC input connector P and negative cable to connector GND.

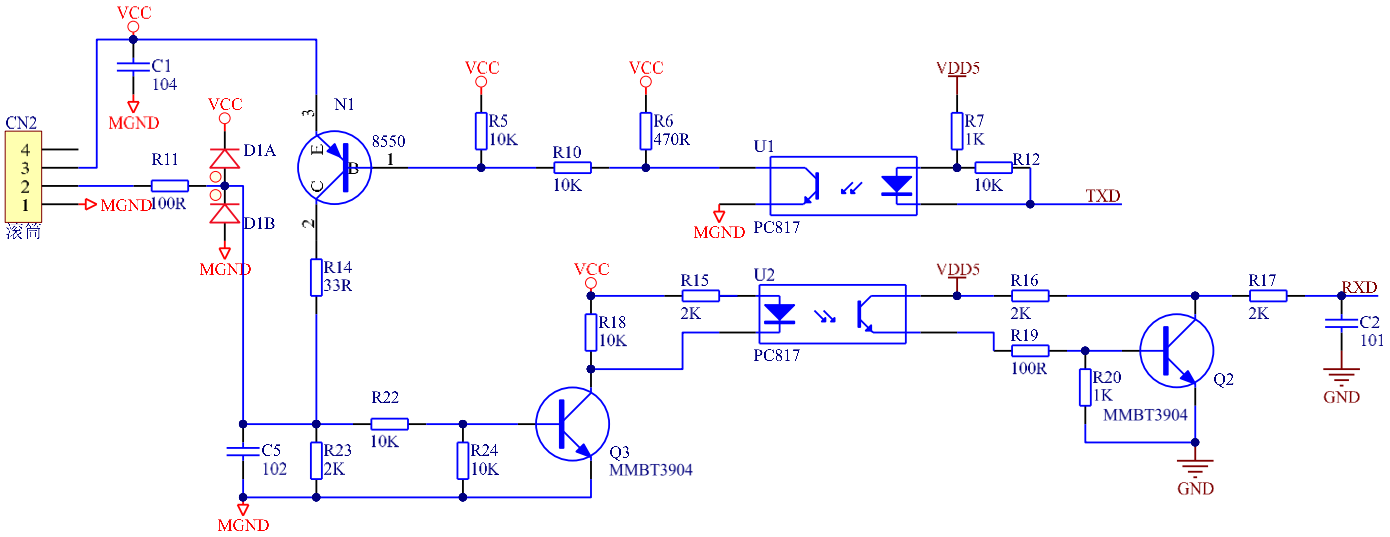
2.1.2 Chip Circuit



Chip Usage:

The EU6815L chip drives BLDC/DD washing machine using an IPM. J1 is a programming cable interface, and J2 is a data monitoring interface.

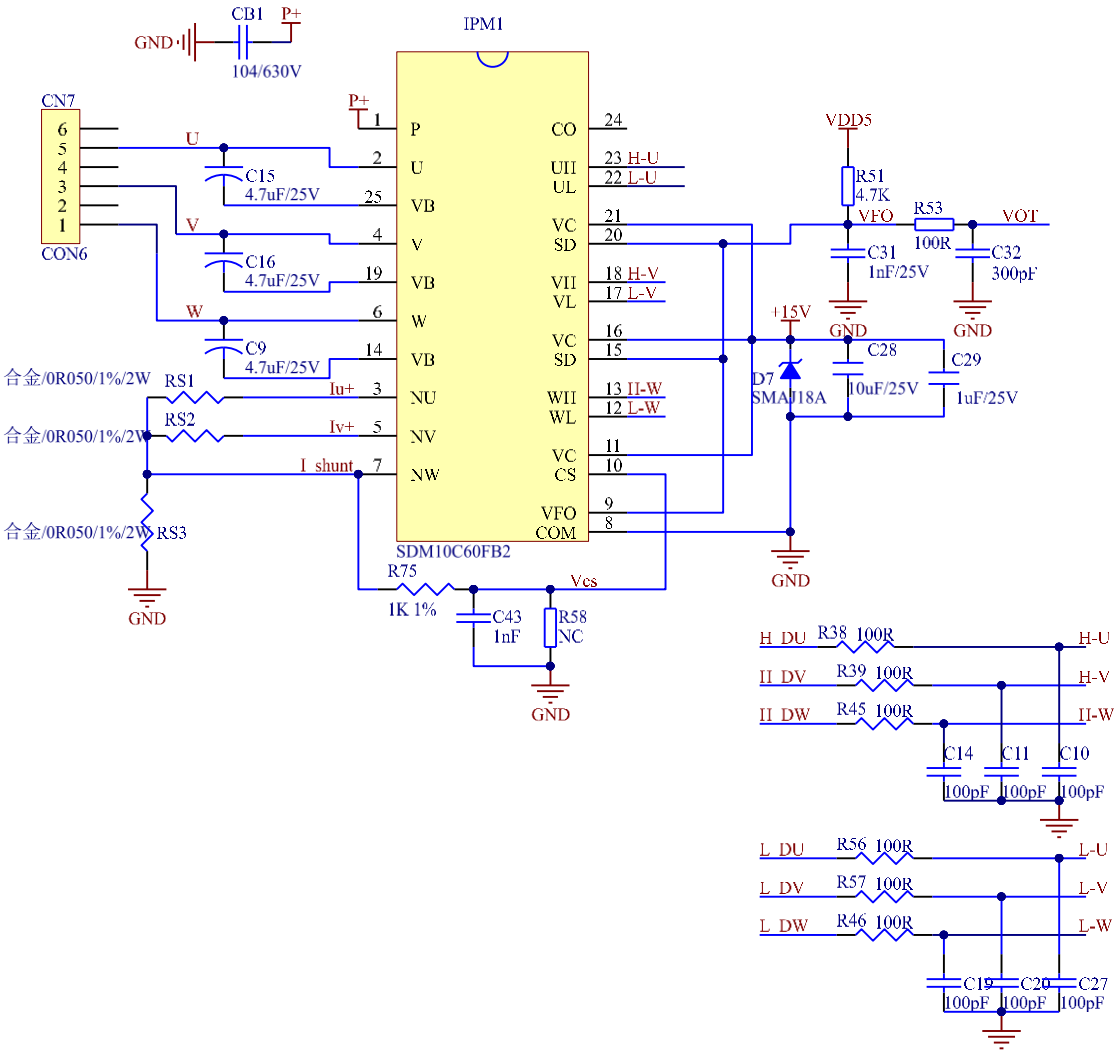
2.1.3 Communication Circuit



Notes:

The circuit communicates with the master computer which sends speed commands to control BLDC/DD operations.

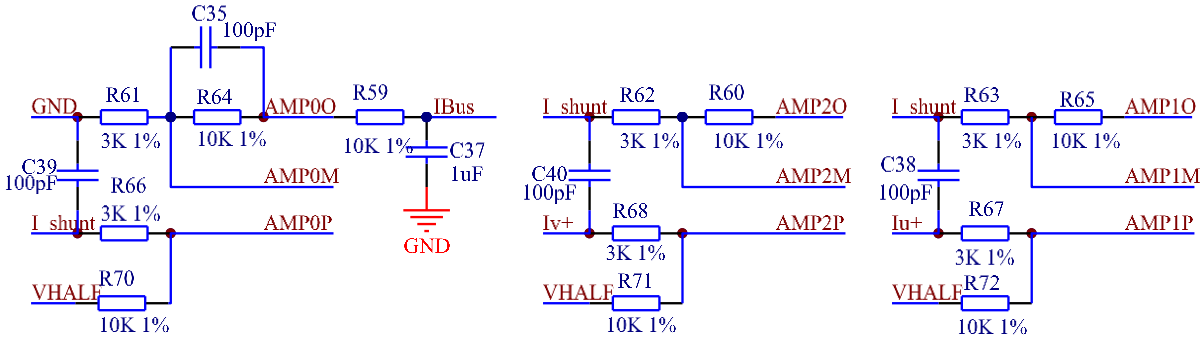
2.1.4 Power Driver Circuit



Notes:

1. The power of shunt resistor may not exceed 80% of the power rating at the maximum current.
2. Note the FO hardware filtering problem.

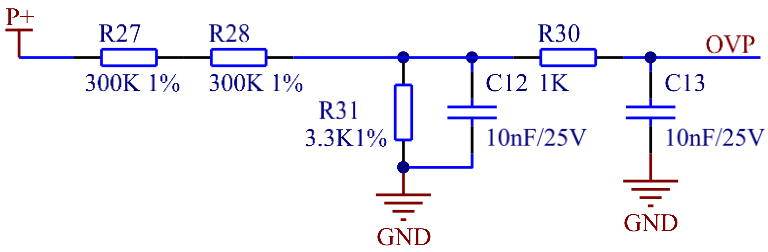
2.1.5 Op-Amp Configuration Circuit



Notes:

1. C38、C39、C40 value shouldn't be adjusted, and should be with an accuracy of 10%;
2. Op amp resistors should apply resistors with an accuracy of 1%;
3. AD3 is for average BUS voltage sampling;
4. Magnification = R60/R62 = R71/R68;
5. Maximum sampling current = (VREF - VHALF)/magnification/sampling resistance;
6. In general, maximum sampling current is set as about 4 times maximum BUS current.

2.1.6 DC Bus Voltage Sampling Circuit



Notes:

1. R30、C13 shouldn't be adjusted;
2. R27、R28、R31 should apply resistors with an accuracy of 1%;
3. Maximum sampling voltage = (R27 + R28 + R31)/(R31)*VREF;
4. In general, maximum sampling voltage is set as twice maximum application voltage, and the voltage at OVP should be lower than 0.8*VREF.

3 Software Flowcharts

3.1 Motor State Machine Flowchart

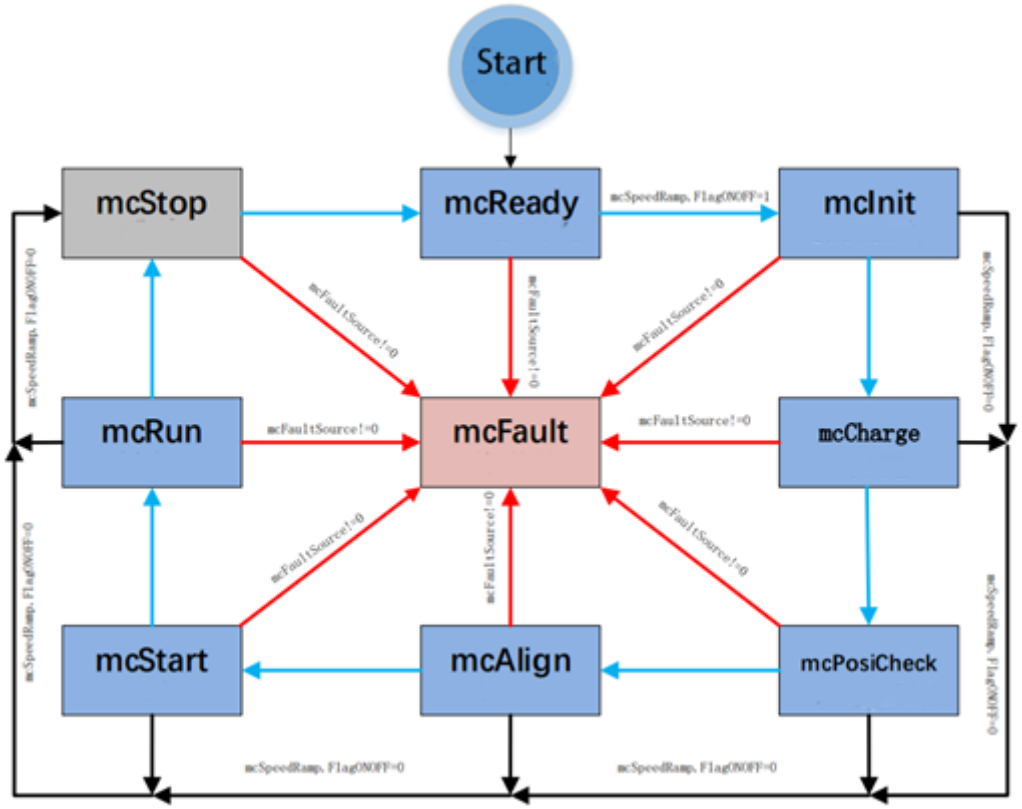


Figure 3-1 Motor State Machine Flowchart

Motor state machine is broken down into three paths as shown in the above figure:

1. Run: mcReady -> mcInit -> mcCharge -> mcPosiCheck -> mcAlign -> mcStart -> mcRun;
2. Stop: In the states of mcInit, mcCharge, mcPosiCheck, mcAlign, mcStart or mcRun, once a shutdown signal is detected, it shifts to mcStop state to slow down then shutdown the motor;
3. Fault: if a fault occurs in any status, it jumps to mcFault state. As fault detection is not performed in mcFault state, concurrent reporting of multiple faults is unavailable.

Notes:

1. mcReady: ready state, waiting for start command. Upon start signal, it shifts to mcInit state;
2. mcInit: the state is to initialize related variables and PI, in which current and external ADC triggering for BUS sampling are switched off. It shifts to the next state once the operation is done.
3. mcCharge: Charge the bootstrap capacitor.
4. mcPosiCheck: initial position detection state, which is to detect the initial position of a motor. This operation is done before normal startup procedure;
5. mcAlign: motor alignment state, in which controller outputs a constant current to drag the motor forcedly to a fixed angle. It shifts to mcStart once the operation is done.

6. mcStart: start state, which is majorly to configure motor startup code. Once the configuration of related registers and variables is done, it enters the next state mcRun. Motor startup process is fulfilled in ME Core.
7. mcRun: run state covers both motor startup stage and running stage. Motor speed control is performed in this state.
8. mcStop: stop state, in which motor stop operation is conducted. It brakes first at high speed, then shuts off output when speed slows down, and enters mcReady state to wait for the next start command.
9. mcFault: fault state. Upon protection occurrence, the program records the error source and shifts state machine to fault state to perform shutdown protection. When the error source is cleared, it enters mcReady state to wait for the next start command.

Notes:

1. The motor state machine supports 8 states, allowing only fixed transition among them. E.g., mcReady state can only switch to mcInit state and mcFault state;
2. In particular, the three states, mcCharge, mcPosiCheck and mcAlign, all support enable bits. When they are not enabled, it skips to the next state directly. E.g., when neither mcPosiCheck nor mcAlign is enabled, it switches from mcCharge to mcStart directly.

3.2 Program Flowchart

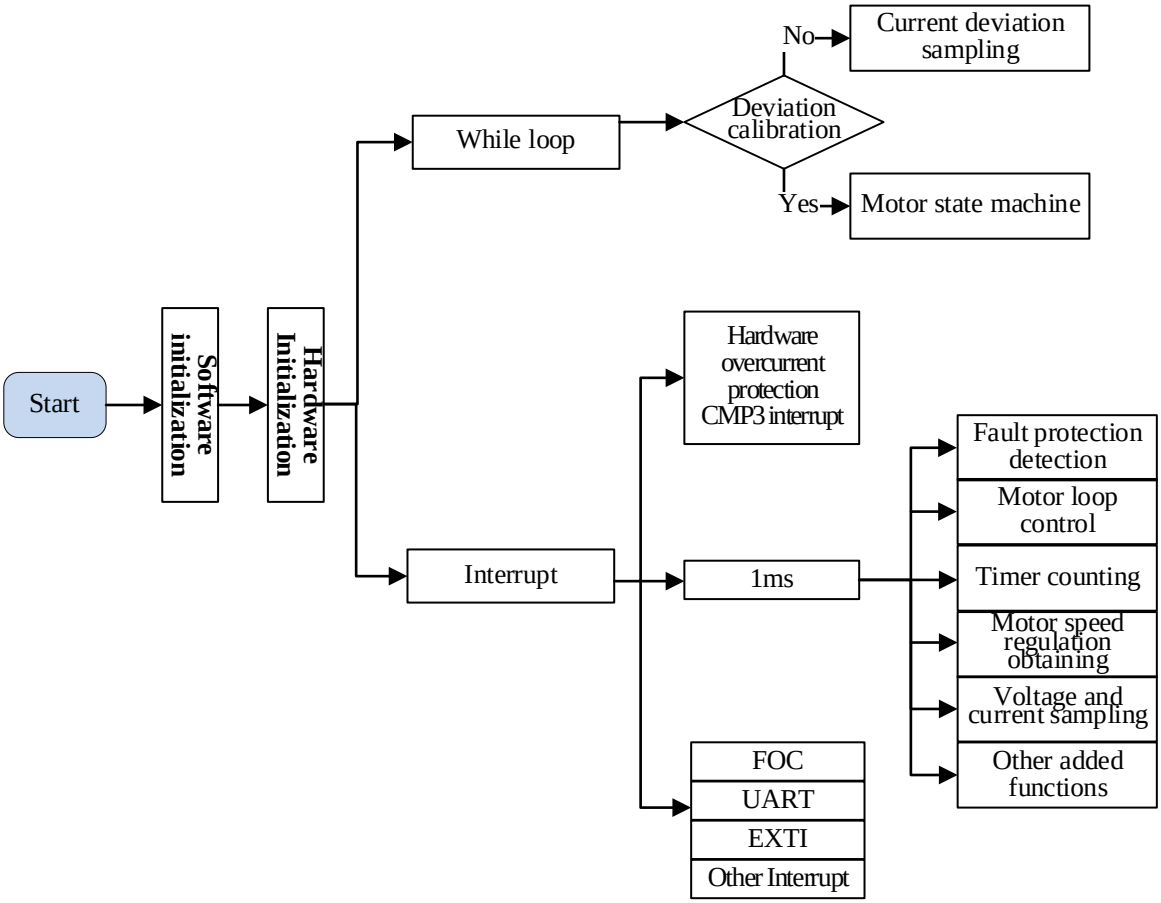


Figure 3-2 Program Execution Flowchart

3.3 Program Specification

3.3.1 Main Function:

Program initialization -> bias voltage sensing GetCurrentOffset() + motor running control MC_Control().

3.3.2 1ms Timer Interrupt

The built-in speed regulation, fault protection detection, bus current and voltage sampling features are called with 1ms Timer Interrupt, including the following functions:

```
Speed_response();          // loop control
StarRampDealwith();        // startup ATO ramp control
Fault_Detection();         // fault detection
Led_Display();             // fault indicator with LED display
IpmTempCal();              // temperature detection
```

3.3.3 FOC Interrupt

FOC interrupt, i.e. carrier interrupt, mainly deals with some fast-timing programs such as AC Power Detector, Wash Angle Calculator, Cradle Wash for Delicates, etc.

3.3.4 CMP3 Interrupt

CMP3 interrupt is mainly designed for hardware overcurrent protection (OCP), which is further explained in 5.2.1.

3.3.5 UART Interrupt

UART Interrupt mainly communicates with the master computer of the washing machine. After instructions are received from the master computer, the master computer RPM, query and other instructions are loaded for execution or upload. During debugging, please note whether the baud rate, parity, stop bit and data bit are consistent with those of the master computer.

```
void USART_INT(void)  interrupt 14
{
    if (UT2TI)
    {
        UT2TI = 0;
        UartSend();
    }

    if (UT2RI)
    {
        UT2RI = 0;
        UartRecive();
    }
}
```

4 Debugging Procedures

4.1 Motor Parameter Configuration

4.1.1 Motor Parameter

1. The number of motor pole pairs.
2. Motor phase resistance RS, phase inductance LD and LQ, and BEMF constant Ke;
3. Motor speed base MOTOR_SPEED_BASE = 2*rated motor speed.

4.1.2 Motor Parameter Measurement Method

1. The number of pole pairs : the parameter value is given in design;
2. Phase resistance Rs: the 2-phase line resistance RL of a motor is measured through a multimeter or LCR; phase resistance Rs = RL/2;
3. Phase inductance Ls: the 2-phase line inductance LL at 1KHz frequency is measured through LCR; phase inductance Ls = LL/2; LD = LQ = Ls;
4. BEMF constant Ke: connect an oscilloscope probe to one phase of a motor, and connect ground to one of the other two terminals of the motor; rotate the load, and measure the BEMF waveform. Take a sine wave in the middle and measure the peak-to-peak Vpp and frequency f. The calculation is as follows:

$$Ke = 1000 * P * \frac{Vpp}{2 * 1.732 * 60 * f}$$

Where, P is the number of pole pairs of the motor.

As an example, the measured BEMF waveform is as follows:

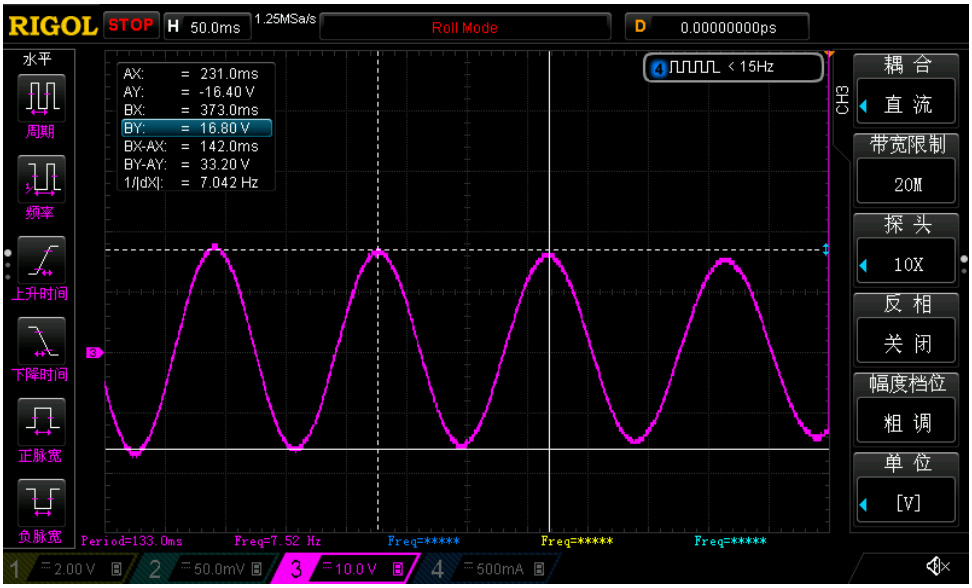


Figure 4-1 BEMF Waveform

The measured peak-to-peak Vpp is 33.2V, the frequency f is 7.042Hz, and the number of pole pairs P is 4, then:

$$\text{BEMF Ke} = 1000 * 4 * \frac{33.2}{2*1.732*60*7.042} = 90.73$$

5. Speed base MOTOR_SPEED_BASE: In general, speed base is set as about 2 times maximum motor speed. As this value affects startup performance and so on, it should be determined beforehand and better unchanged later.

4.1.3 Corresponding Program Code

```
#define Pole_Pairs                  (24)                                    ///< Pole pairs
#define LD                          (0.047)                                 ///< (H) d-Axis inductance
#define LQ                          (0.047)                                 ///< (H) q-Axis inductance
#define RS                          (6.8)                                   ///< (Ω) Phase resistance
#define KeVpp                       (143.0)
#define KeF                         (41.5)
#define Ke                          (Pole_Pairs * KeVpp * 1000 / KeF / 207.84)   ///< (V/kRPM) Back EMF constant
#define MOTOR_SPEED_BASE            (1600.0)                                ///< (RPM) Motor speed base
```

4.2 Chip Internal Parameter Configuration

```
/**********************Basic Parameter*******************************
1. Motor PWM Parameter
2. Motor Parameter
3. Current Sampling Parameter
4. Bus Voltage Sampling Parameter
 ************************************************************/
/* ------------------------------------------------------------------------------------------------------------------
                                        1.Motor PWM Parameter
 ------------------------------------------------------------------------------------------------------------------ */

#define PWM_FREQUENCY               (16.0)                  ///< (kHz) Motor PWM frequency
#define PWM_DEADTIME                (1.5)                   ///< (μs) Deadtime
#define MIN_WIND_TIME               (PWM_DEADTIME+0.9)      ///< (μs) Minimum sampling window of single-resistance current sampling mod
```

Notes:

1. In general, carrier frequency needs to be set as about 10 times maximum electrical cycle. As carrier frequency affects startup, MOS temperature rising and so on, users need to select a proper carrier frequency before debugging. Washer motors generally use 16K of carrier frequency;
2. Dead zone value is set according to actual MOS on/off speed to ensure no risk of shoot-through;
3. Minimum sampling window should be greater than 2 times dead zone and less than 1/16 of carrier cycle, i.e., 1000/16/PWM_FREQUENCY > MIN_WIND_TIME > 2*PWM_DEADTIME;

4.3 Hardware Parameter Configuration

1. Figure out BUS voltage divider ratio, sampling resistance value, and magnification according to the voltage and power ranges of a motor.
2. Resistance and magnification selection rules:
   1) BUS voltage divider resistance:
      - Voltage divider ratio should not be too small: In general, suggest maximum sampling voltage is 0.8*VREF. E.g., for a motor with maximum voltage being 30V and ADC reference VREF being 4.5V, suggest the voltage divider ratio is no less than 30/0.8/4.5 = 8.33; If the voltage divider ratio is too small a value like 5, when the voltage is 30V, the voltage at the AD port is 6V after voltage division, then it overflows.
      - Voltage divider ratio should not be too large: If so, the AD sampling voltage accuracy is insufficient. E.g.,

If voltage divider ratio is 40, when maximum voltage is 30V, the voltage at AD port is 30V/40V = 0.75V; when maximum voltage is 28V, the voltage at AD port is 0.7V. Thus, the accuracy is quite low. Moreover, the AD still has a margin of 4.5 - 0.75 = 3.75V.

2) Sampling resistance and magnification:

Maximum sampling current = VREF/HW_RSHUNT/HW_AMPGAIN; it should be noted that maximum sampling current is not the current displayed on power supply (i.e., the current after filtering), but the current through a sampling resistor.

- Sampling resistance should not be too large: If so, it is easy to cause sampling overflow or resistor power out of range; sampling resistors of 2512 packaging commonly support 1W or 2W power; sampling resistors of 1206 packaging commonly support 1/4W; users should make sure the power I^2R through a sampling resistor does not exceed the corresponding power value.
- Sampling resistance should not be too small: If so, the accuracy is insufficient.
- Magnification is adjusted according to sampling resistance. Determine sampling resistance first, then adjust magnification.

3. Fill the values of BUS voltage divider ratio, sampling resistance and magnification into the program code (in the Customer.h file) accordingly.

```
56                                      3.Current Sampling Parameter
57  ------------------------------------------------------------------------------------------------------------------------ */
58
59  #define HW_RSHUNT                   (0.05)                      ///< (Ω) Current sampling resistance
60  #define HW_ADC_REF                  (4.5)                       ///< (V) ADC voltage reference
61  #define HW_AMPGAIN                  (3.3)                       ///< AMP amplification gain
62  #define VREF_OUT_EN                 (1)                         ///< VREF_OUT_Enable
63  #define VHALF_OUT_EN                (1)                         ///< VHALF_OUT_Enable
64
65  /* ------------------------------------------------------------------------------------------------------------------------
66                                      4.Bus Voltage Sampling Parameter
67  ------------------------------------------------------------------------------------------------------------------------ */
68
69  #define RV1                         (300.0)                     ///< (kΩ) Bus voltaeg divider resistor1
70  #define RV2                         (300.0)                     ///< (kΩ) Bus voltaeg divider resistor2
71  #define RV3                         (3.3)                       ///< (kΩ) Bus voltaeg divider resistor3
72  #define RV                          ((RV1 + RV2 + RV3) / RV3)   ///< (V) The maximum sampling bus voltage
73
74  #define ACRV1                       (300.0)                     ///< (kΩ) AC voltaeg divider resistor1
75  #define ACRV2                       (300.0)                     ///< (kΩ) AC voltaeg divider resistor2
76  #define ACRV3                       (3.3)                       ///< (kΩ) AC voltaeg divider resistor3
77  #define ACRV                        ((ACRV1 + ACRV2 + ACRV3) / ACRV3)
78  #define AC_VOLT_MAX                 (HW_ADC_REF * ACRV)          ///  Maximum AC voltage measured by ADC
79
```

Where,

1) BUS voltage divider ratio = (RV1 + RV2 + RV3)/RV3;

4.4 Protection parameter configuration

1. Current protection settings:

- Hardware overcurrent: Set hardware overcurrent protection value according to the maximum current value of a power device. In general, set hardware overcurrent protection value OverHardcurrentValue larger than maximum BUS current, and less than the maximum current value of the power device.
- Software overcurrent: In general, set OverSoftCurrentValue a little smaller than hardware overcurrent. Software overcurrent is triggered by software, and protection time is less than that of hardware overcurrent.

2. Set overvoltage/undervoltage protection and protection recovery parameters. Please refer to 5.2.2 for

details;

3. Turn off all protections except the above to prevent false triggering during startup. Apply other protections later when they are required. As for overcurrent protection, it is always on with no enable bit.
4. Fill the parameters into the program code accordingly (in the Protect.h file).

```
33
34  /* ------------------------------------Protection Enable---------------------------------------------------- */
35
36  #define SWCurrentProtectEn                (1)                   ///< Software overcurrent  protection. 0 is disable, 1 is enable.
37  #define VoltageProtectEn                  (1)                   ///< Bus voltage protection. 0 is disable, 1 is enable.
38  #define StallProtectEn                    (1)                   ///< motor stall protection. 0 is disable, 1 is enable
39  #define PhaseLossProtectEn                (1)                   ///< Phase loss protection. 0 is disable, 1 is enable
40  #define OverTPProtectEn                   (1)                   ///< Temperature protection. 0 is disable, 1 is enable
41  #define ComErrorProtectEn                 (1)                   ///< communication protection. 0 is disable, 1 is enable
42
43  /* -----Overvoltage/Undervoltage protection----- */
44
45  #define Over_Protect_ACVoltage            (282)                 ///< (V) AC voltage overvoltage threshold
46  #define Over_Recover_ACVlotage            (256)                 ///<(V)  AC voltage overvoltage recover value
47
48  #define Under_Protect_ACVoltage1           (80)                 ///< (V) AC voltage undervoltage threshold
49  #define Under_Protect_ACVoltage2           (100)                 ///< (V) AC voltage undervoltage threshold
50  #define Under_Protect_ACVoltage3           (120)                 ///< (V) AC voltage undervoltage threshold
51
52  #define Under_Recover_ACVlotage           (150)                 ///< (V) AC voltage undervoltage recover value
53
54  #define Over_Protect_Voltage              (400)                 ///< (V) Bus voltage overvoltage threshold
55  #define Over_Recover_Vlotage              (360)                 ///< (V) Bus voltage overvoltage recover value
56  #define Under_Protect_Voltage             (220)                 ///< (V) Bus voltage undervoltage threshold
57  #define Under_Recover_Vlotage             (240)                 ///< (V) Bus voltage undervoltage recover value
58
59
60  /* ----- motor stall protection----- */
61
62  #define STALL_MAX_SPEED                   S_Value(17000)        ///< motor stall MAX Speed
63  #define STALL_MIN_SPEED                   S_Value(200)          ///< motor stall Min Speed
64
65  #define EsThresholdValue0                 (60.0)                ///< motor stall Back EMF parameters0
66  #define EsThresholdValue1                 (80.0)                ///< motor stall Back EMF parameters1
67  #define EsThresholdValue2                 (500.0)               ///< motor stall Back EMF parameters2
68
```

4.5 Startup Parameter Configuration

Users can take all startup default settings first then adjust the values when encountering startup problems or difficulties. Please refer to 5.1 for parameter adjustment details to settle down common startup issues.

```
132
133 /* -----Startup current----- */
134
135 #define ID_Start_CURRENT              I_Value(0.0)                  ///< (A) d-Axis startup current
136 #define IQ_Start_CURRENT              I_Value(4.5)                  ///< (A) q-Axis startup current
137 #define IQ_Start_CURRENT_End          I_Value(4.5)                  ///< (A) q-Axis startup current end
138
139 /* -----Current when the controller switch to model----- */
140
141 #define ID_RUN_CURRENT                I_Value(0.0)                  ///< (A) d-Axis running current
142 #define IQ_RUN_CURRENT                I_Value(2.0)                  ///< (A) d-Axis running current
143 #define IQ_TailWind_CURRENT           I_Value(0.75)                 ///< (A) d-Axis TailWind running current
144
```

1. Startup current: In general, ID_Start_CURRENT is fixed to 0 and IQ_Start_CURRENT is set according to actual motor settings;

Notes:

IQ_Start_CURRENT should not be too small. If so, the starting torque gets too small to start the motor normally.

IQ_Start_CURRENT should not be too large. If so, overshoot in startup is encountered and startup noise is introduced.

2. Switching current: IQ_RUN_CURRENT determines transient current. By observing actual phase current upon IO port reverse, users can figure out whether current is smooth at loop switching moments. Tune IQ_RUN_CURRENT accordingly if required.
3. Startup ATO: Since inaccuracy output of FOC estimator in the case of lower speed, it is necessary to set ATO_BW (speed bandwidth filtering value) to limit the maximum speed output of FOC estimator;

```
144
145 /* -----ATO bandwidth----- */
146
147 #define ATO_BW                          (60.0)                          ///< (Hz) ATO bandwidth, the recommended value is 1.0 ~ 200.0. In AC
148 #define ATO_BW_RUN                      (60.0)
149 #define ATO_BW_RUN1                     (80.0)
150 #define ATO_BW_RUN2                     (120.0)
151 #define ATO_BW_RUN3                     (120.0)
152 #define ATO_BW_RUN4                     (120.0)
153 #define ATO_BW_RUN5                     (120.0)
154
```

Notes:

Since AO observer is turned on, make the first ATO_BW not too large.

4. Speed-bandwidth filtering value SPD_BW;

```
154
155 /* -----Speed filter bandwidth----- */
156
157 #define SPD_BW                          (8.0)                           ///< (Hz) Speed filter bandwidth, the recommended value is 5.0 ~ 40.0
158 #define ATT_COEF                        (0.85)                          ///< Damping factor of ATO
159
160 /* -----OMEGA startup parameter----- */
161
162 #define Motor_Omega_Ramp_ACC            (5)                             ///< Forced speed increment step in standstillstate
163 #define MOTOR_OMEGA_ACC_MIN             (10.0)                          ///< (RPM) The minimum speed to switch form forced speed to estimated s
164 #define MOTOR_OMEGA_ACC_END             (20.0)                          ///< (RPM) The maximum speed of forced speed
165
```

Notes:

In general, it's unnecessary to adjust SPD_BW.

5. Omega startup settings affect startup current frequency, namely motor startup acceleration;

```
159
160 /* -----OMEGA startup parameter----- */
161
162 #define Motor_Omega_Ramp_ACC            (5)                             ///< Forced speed increment step in standstillstate
163 #define MOTOR_OMEGA_ACC_MIN             (10.0)                          ///< (RPM) The minimum speed to switch form forced speed to estimated
164 #define MOTOR_OMEGA_ACC_END             (20.0)                          ///< (RPM) The maximum speed of forced speed
165
166 /* The motor speed to switch from mode0 to model */
167
168 #define MOTOR_LOOP_RPM                  (20.0)                          ///< (RPM) The motor speed to switch from mode0 to model. i.e., The m
169
170 /* motor run speed value */
```

Notes:

1) The reference value range of Motor_Omega_Ramp_ACC is 10 ~ 50;
2) The reference value range of MOTOR_OMEGA_ACC_MIN is 200 ~ 500;
3) The reference value range of MOTOR_OMEGA_ACC_END is 500 ~ 3000;
4) MOTOR_LOOP_RPM should be greater than MOTOR_OMEGA_ACC_END. The reference value range of MOTOR_LOOP_RPM is 2000 ~ 4000.

6. Current loop PI: It is divided into startup-stage current loop PI and run-stage current loop PI;

```
/* -----Current loop coefficient after controller switch to model----- */

#define DKP                             _Q12(2.0)                       ///< DKP of current loop after controller switch to model
#define DKI                             _Q15(0.02)                      ///< DKI of current loop after controller switch to model
#define QKP                             _Q12(2.0)                       ///< QKP of current loop after controller switch to model
#define QKI                             _Q15(0.02)                      ///< QKI of current loop after controller switch to model

#define FiledWeakenCompEnable             (1)                           ///< Filed Weaken CompEnable
```

Notes:

1) Startup-stage current loop PI affects motor startup;
2) Run-stage current loop PI affects current stability and efficiency;
3) The recommended range of DQKP is 3.0 ~ 0.1.
4) The recommended range of DQKI is 0.05 ~ 0.001

7. The maximum output limit of DQ axis: D axis affects motor magnetic flux and Q axis affects motor torque.

```
#define FiledWeakenCompEnable           (1)                             ///< Filed Weaken CompEnable

#define MotorFiledWeakenUs                _Q15(0.70)                    ///< Motor Filed Weaken reference voltage

/* -----d-Axis voltage reference limit value----- */

#define DOUTMAX                           _Q15( 0.85)                   ///< d-Axis voltage reference maximum value. The recommended value is _
#define DOUTMIN                           _Q15(-0.85)                   ///< d-Axis voltage reference minimum value. The recommended value is _

/* -----q-Axis voltage reference limit value----- */

#define FWQOUTMAX                         _Q15( 0.55)                   ///< q-Axis voltage reference minimum value. The recommended value is _
#define FWQOUTMIN                         _Q15(-0.55)                   ///< q-Axis voltage reference minimum value. The recommended value is _
#define QOUTMAX                           _Q15( 0.75)                   ///< q-Axis voltage reference maximum value. The recommended value is _
#define QOUTMIN                           _Q15(-0.75)                   ///< q-Axis voltage reference minimum value. The recommended value is _
```

Notes:

1) FOC__UQ feedbacks whether motor output is saturated.
2) The greater the positive value of FOC__UD, the greater the lead angle. Users can increase motor lead angle by increasing compensation angle (FOC_THECOMP), thus maximum motor speed is lifted. FOC__UD is a positive value.
3) An excessive lead angle causes current overshoot during motor shutdown, which can be settled down by low-voltage warning in shutdown, or by fast under-voltage protection.
4) An excessive lead angle causes efficiency decline. The phase current amplitude becomes larger at the same power. So that compensation angle should be set properly.

4.6 Hardware Driver Circuit Configuration

```
251
252 #define IPMState                        (NormalRun)                    ///< Operation mode selection
253
254 /* -----Estimator Mode Selection----- */
255
256 #define SMO                            (0)                            ///< SMO Observer
257 #define PLL                            (1)                            ///< PLL Observer
258 #define MF                             (2)                            ///< AO Observer
259 #define EstimateAlgorithm              (MF)
260
261 /* -----Current sampling mode selection----- */
262
263 #define Single_Resistor                (0)                            ///< Single-resistance current sampling mode
264 #define Double_Resistor                (1)                            ///< Double-resistance current sampling mode
265 #define Three_Resistor                 (2)                            ///< Triple-resistance current sampling mode
266 #define Shunt_Resistor_Mode            (Double_Resistor)
267
268 #define OverModulation                 (0)                            ///< UDQ will be multiplied by 1.15 after OverModulation is enabled
269
```

Set the motor running mode to IPMState. If constant PWM waveform outputs are generated on U, V and W phases, it suggests that the hardware driver circuit functions properly, otherwise you need to locate the hardware faults.

4.7 Speed Loop and Voltage Loop Debugging

1. Generally, the washing machine controls speed through the speed loop. The motor speed range is configured depending on the specific motor. The speed of a BLDC motor ranges from 350RPM to 17000RPM, and the speed of a DD motor ranges from 30RPM to 1400RPM.
2. Set the speed loop output limit SPEEDLOOP_FWIQMAX which is a function of the washing machine capacity and motor demagnetization current. The washing machine capacity should be so designed that it can be started properly when partially or fully loaded at motor temperature of 120°C.
3. Considering that overshoot in DC bus voltage may damage the DEMO board, we need a voltage loop to control the DC bus voltage in a desired range and prevent large capacitance causing the MOSFET to break down.
4. Adjust the speed loop PI and ramp increment to guarantee stability of the speed loop and quick startup response without overshoot. If a set of PI parameters cannot meet the requirements of the entire speed segment, you may configure different PI parameters for different speed segments.

```
209
210 #define SPEEDLOOP_FWIDMAX               I_Value( 4.0)           ///< Speed loop D-axis current Max limit
211 #define SPEEDLOOP_FWIDMIN               I_Value(-4.0)           ///< Speed loop D-axis current Min limit
212 #define SPEEDLOOP_FWIQMAX               I_Value( 5.8)           ///< Speed loop Q-axis current Max limit
213 #define SPEEDLOOP_FWIQMIN               I_Value(-5.8)           ///< Speed loop Q-axis current Max limit
214
215 #define DQPISWSPEEDH                    S_Value(900)            ///< High speed PI parameter switching speed
216 #define SPEEDLOOP_IDQSKPH               _Q10(4.5)               ///< High speed Speed loop KP
217 #define SPEEDLOOP_IDQSKIH               _Q15(0.0001)            ///< High speed Speed loop KI
218
219 #define SPEEDLOOP_IDQUKPH               _Q10(16.0)              ///< High speed Voltage loop KP
220 #define SPEEDLOOP_IDQUKIH               _Q15(0.02)              ///< High speed Voltage loop KI
221
222 #define CURRENTLOOP_DQKPH               _Q12(2.0)               ///< High speed Current loop KP
223 #define CURRENTLOOP_DQKIH               _Q15(0.002)             ///< High speed Current loop KI
224
225 #define DQPISWSPEEDM                    S_Value(250)            ///< Medium speed PI parameter switching speed
226 #define SPEEDLOOP_IDQSKPM               _Q10(8.0)               ///< Medium speed Speed loop KP
227 #define SPEEDLOOP_IDQSKIM               _Q15(0.002)             ///< Medium speed Speed loop KI
228
229 #define SPEEDLOOP_IDQUKPM               _Q10(16.0)               ///< Medium speed Voltage loop KP
230 #define SPEEDLOOP_IDQUKIM               _Q15(0.02)               ///< Medium speed Voltage loop KI
231
232 #define CURRENTLOOP_DQKPM               _Q12(2.0)                ///< Medium speed Current loop KP
233 #define CURRENTLOOP_DQKIM               _Q15(0.01)               ///< Medium speed Current loop KI
234
235 #define DQPISWSPEEDL                    S_Value(100)             ///< Low speed PI parameter switching speed
236 #define SPEEDLOOP_IDQSKPL               _Q10(24.0)               ///< Low speed Speed loop KP
237 #define SPEEDLOOP_IDQSKIL               _Q15(0.005)              ///< Low speed Speed loop KI
238
239 #define SPEEDLOOP_IDQUKPL               _Q10(16.0)               ///< Medium speed Voltage loop KP
240 #define SPEEDLOOP_IDQUKIL               _Q15(0.02)               ///< Medium speed Voltage loop KI
241
242 #define CURRENTLOOP_DQKPL               _Q12(4.0)                ///< Medium speed Current loop KP
243 #define CURRENTLOOP_DQKIL               _Q15(0.1)                ///< Medium speed Current loop KI
244
```

5 Function Description

So far, motors can basically start properly when the Start signal is detected after running the initial release of the program and configuring the motor parameters and hardware parameters as required. If motor cannot start as expected, you should adjust the startup parameters after hardware faults are cleared.

5.1 Startup Debugging

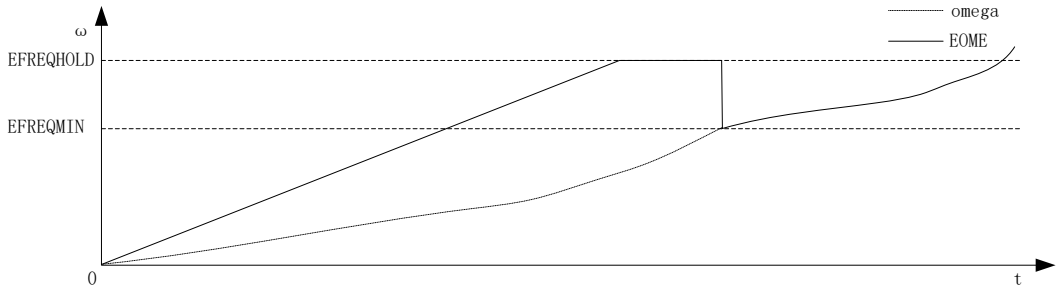
5.1.1 Omega Startup

Select Omega Startup, the program corresponds to this startup method by default

```
260 /* -----(Open_Start)----- */
261 /* -----(Omega_Start)----- */
262 /* -----(Open_Omega_Start)----- */
263 #define Open_Start_Mode                (Omega_Start)
```

When the estimated speed OMEGA of FOC estimator is less than the minimum value FOC_EFREQMIN (corresponding to the parameter MOTOR_OMEGA_ACC_MIN) set by user, the forced speed starts from 0. It is added up with the incremental speed value FOC_EFREQACC (the parameter Motor_Omega_Ramp_ACC) and meanwhile it is limited by the maximum value FOC_EFREQACC (corresponding to the parameter Motor_Omega_Ramp_AC) in each operation cycle. The forced speed is output as the final speed EOME, which is applied by angle calculation module to calculate estimator angle ETHETA; when the estimated speed OMEGA of FOC estimator is greater than or equal to EFREQMIN, the estimated speed OMEGA is output as the final speed EOME.

Startup procedure is shown in the figure below:



5.1.2 Common Issues & Solutions in Starting

Common Issues	Solutions
The motor rotates at an instant then stops with output current all the time.	1. A possible reason is startup current being too small, which cannot drive the motor to the next commutation. The solution is to increase IQ_Start_CURRENT; 2. Another possible reason is the output speed of FOC estimator being too small, which cannot drive the motor to the next commutation. If Item 1 is ruled out, then increase ATO_BW, ATO_BW_RUN, ATO_BW_RUN1 and ATO_BW_RUN2 sequentially; 3. If item 1 and item 2 are ruled out, check whether the hardware circuit of AMP0 encounters problems, which leads to inaccurate current sampling and the false estimation of FOC estimator; 4. It's also possible that the frequency of omega acceleration is too high. The solution is to reduce Motor_Omega_Ramp_ACC.
The motor rotates at an instant then stops and keeps jittering.	1. It's most probably due to ATO_BW being too large, which causes the output speed of FOC estimator being high, and makes the motor run out during startup. The solution is to reduce ATO_BW, ATO_BW_RUN, ATO_BW_RUN1 and ATO_BW_RUN2 sequentially; 2. Another possible reason is improper Omega startup settings.
The motor starts and rotates forward for a certain angle, then is stuck and locked at an instant, then rotates normally.	1. Users can estimate the time from startup to freezing, and then set ATO_BW accordingly. E.g., the motor starts and operates for 1s then freezes for at an instant then operates normally. The period 1s is corresponding to ATO_BW_RUN1 and ATO_BW_RUN2. The issue is caused by ATO_BW being relatively small, which limits motor speeding up. The solution is to increase ATO_BW. 2. Omega acceleration being too small can cause the stuck and locked as well. The solution is to increase Motor_Omega_Ramp_ACC.
The motor starts and rotates reversely, then when turning to rotate forward, it gets to jitter constantly.	1. It takes a long time for the motor to rotate reversely at an instant upon startup then rotate forward. At the time, ATO_BW is already increased to a relatively large value. The solution is to reduce ATO_BW; 2. Another possible reason is the frequency of omega acceleration being too high. The solution is to adjust Motor_Omega_Ramp_ACC.

5.2 Introduction to Protection Functions

Protection values vary in different projects, motors and boards. Parameters for various protection functions need to be adjusted according to real projects. Upon the occurrence of expected motor locked protection or fault protection not being reported, or protection being falsely triggered in normal operation, it indicates improper setting of protection parameters, users need to adjust them accordingly.

5.2.1 Overcurrent Protection

1. Hardware overcurrent protection;

Hardware overcurrent protection is fulfilled through comparator 3 in the chip. The detection method is: The BUS current flows through the sampling resistor, and a voltage is formed on the sampling resistor; the voltage is amplified by the operational amplifier and sent to the positive input of the comparator. A reference voltage generated by a DAC or by an external voltage divider (DAC is used currently) is input to the negative input of the comparator. When the BUS current is increasing to a certain value where the voltage of the comparator's positive input is higher than that of the negative input, a comparator interrupt in MCU is triggered. Upon this interrupt, MCU turns off the MOE automatically (whether it is automatic or not is configurable and it is automatic by default) to fulfill overcurrent protection. For hardware overcurrent protection, users only need to adjust OverHardcurrentValue.

```
23
24  #define Compare_DAC_MODE                    (0)                     ///< DAC setting hardware overcurrent value
25  #define Compare_HW_MODE                     (1)                     ///< Hardware setting hardware overcurrent value
26  #define Compare_Mode                        (Compare_DAC_MODE)      ///< Source of hardware overcurrent value
27  #define OverHardcurrentValue                (8.5)                   ///< (A) Hardware overcurrent value in DAC mode
28
29  /* ----Software overcurrent protection parameter setting---- */
30
31  #define SW_OC_CurrentVal                    I_Value(8.5)
32
33
34  /* -------------------------------------Protection Enable------------------------------------------------------ */
35
36  #define SWCurrentProtectEn                  (1)                     ///< Software overcurrent  protection. 0 is disable, 1 is enable.
37  #define VoltageProtectEn                    (1)                     ///< Bus voltage protection. 0 is disable, 1 is enable.
38  #define StallProtectEn                      (1)                     ///< motor stall protection. 0 is disable, 1 is enable
39  #define PhaseLossProtectEn                  (1)                     ///< Phase loss protection. 0 is disable, 1 is enable
40  #define OverTPProtectEn                     (1)                     ///< Temperature protection. 0 is disable, 1 is enable
41  #define ComErrorProtectEn                   (1)                     ///< communication protection. 0 is disable, 1 is enable
42
```

2. Software overcurrent protection

The program obtains the maximum current value of the three phases, and counts it once when the maximum current value exceeds the SW_OC_CurrentVal of the set software overcurrent protection value; Protection is triggered when the three-phase current trigger count reaches the set count.

5.2.2 Voltage Protection

Voltage protection is performed by detecting AC and DC voltages through the AD6 and AD2 GPIO pins when the detected voltage exceeds a set value. Then when the voltage becomes lower than the overvoltage recovery value again, the overvoltage protection fault is cleared. When the voltage is lower than a set undervoltage value, an undervoltage protection is reported. Then when the voltage becomes higher than the undervoltage recovery value again, the undervoltage protection fault is cleared.

```
42
43  /* -----Overvoltage/Undervoltage protection----- */
44
45  #define Over_Protect_ACVoltage              (282)                   ///< (V) AC voltage overvoltage threshold
46  #define Over_Recover_ACVlotage              (256)                   ///<(V)  AC voltage overvoltage recover value
47
48  #define Under_Protect_ACVoltage1             (80)                   ///< (V) AC voltage undervoltage threshold
49  #define Under_Protect_ACVoltage2             (100)                   ///< (V) AC voltage undervoltage threshold
50  #define Under_Protect_ACVoltage3             (120)                   ///< (V) AC voltage undervoltage threshold
51
52  #define Under_Recover_ACVlotage             (150)                   ///< (V) AC voltage undervoltage recover value
53
54  #define Over_Protect_Voltage                (400)                   ///< (V) Bus voltage overvoltage threshold
55  #define Over_Recover_Vlotage                (360)                   ///< (V) Bus voltage overvoltage recover value
56  #define Under_Protect_Voltage               (220)                   ///< (V) Bus voltage undervoltage threshold
57  #define Under_Recover_Vlotage               (240)                   ///< (V) Bus voltage undervoltage recover value
58
```

5.2.3 Phase Loss Protection

3-phase current is asymmetrical in case of motor phase loss. Based on this, phase loss protection function detects the maximum values of 3-phase current within a certain period, and judges whether the maximum values of the 3-phase current are asymmetric.

The specific procedure is: When it is detected the maximum current of a phase is greater than PhaseLossTimes times the maximum current of another phase, and its maximum current is greater than the set PhaseLossCurrentValue value, it is determined that phase loss occurs.

Notes:

In some cases, when a phase is missing, the signal of the missing phase will have burrs, which may cause the maximum current value collected to be about the same as those of the other two phases, which may not be detected by the above method. The phase loss detection method discussed above may fail in the case. Solution: Phase loss can be judged by comparing the accumulated current value within a certain period through integration method.

5.2.4 Motor Locked Protection

There are 3 detection methods for motor locked protection:

1. Judge by detecting the FOC_ESQU value calculated by FOC estimator (the square of BEMF calculated by FOC estimator). In normal case, the higher the motor speed, the greater the FOC_ESQU. Upon motor being locked, the motor runs out, and the estimated speed is very high, but the FOC_ESQU is very small. Therefore, it can be judged whether it is a stall rotation by the back EMF and rotational speed;

```
64
65 #define EsThresholdValue0                (60.0)                ///< motor stall Back EMF parameters0
66 #define EsThresholdValue1                (80.0)                ///< motor stall Back EMF parameters1
67 #define EsThresholdValue2                (500.0)               ///< motor stall Back EMF parameters2
68
69
70 #define EsThresholdSpeed1                S_Value(350)          ///< motor stall speed1
71 #define EsThresholdSpeed2                S_Value(4600.0)       ///< motor stall speed2
72
```

2. Detect whether estimated speed exceeds the set speed MOTOR_SPEED_STAL_MAX_RPM, or is lower than the set speed MOTOR_SPEED_STAL_MIN_RPM. If so, the motor is judged as locked.

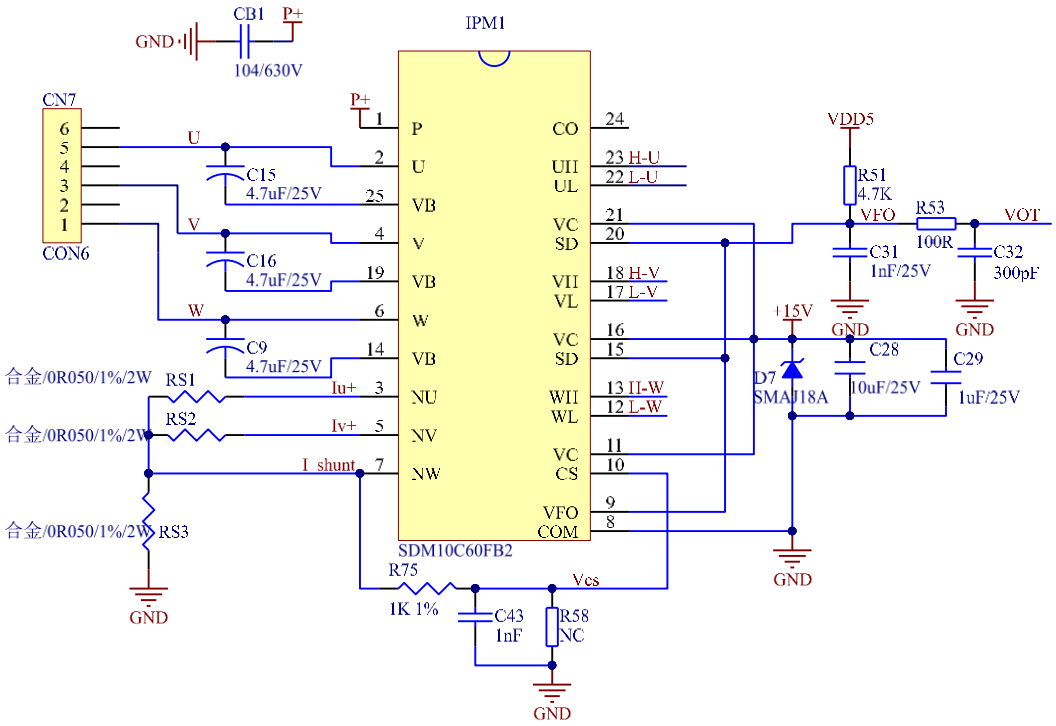
```
59
60 /* ----- motor stall protection----- */
61
62 #define STALL_MAX_SPEED                  S_Value(17000)        ///< motor stall MAX Speed
63 #define STALL_MIN_SPEED                  S_Value(200)          ///< motor stall Min Speed
64
```

3. During motor startup, when determining estimated speed being greater than MOTOR_LOOP_RPM, the program sets the mode from 0 to 1 to start the motor with a fixed current, then enter normal loop. At the time point, the mode value can be used to judge whether the motor is locked. If the mode value is still 0 after a time period of FOCMode_DectTime since the motor starts, it can be regarded as motor startup failure, namely it is judged as locked.

```
277         if (mcFocCtrl.CtrlMode == 0)
278         {
279             Fault.Stall.Mode0DectCnt++;
280
281             if (Fault.Stall.Mode0DectCnt >= 3000)
282             {
283                 Fault.Stall.Mode0DectCnt = 0;
284                 mcFaultSource            = FaultStall;
285                 Fault.Stall.Type         = 31;
286             }
287         }
288         else
289         {
290             Fault.Stall.Mode0DectCnt = 0;
291         }
292     }
293 }
```

5.2.5 Thermal Shutdown (TSD)

The figure below provides a common schematic representation of thermal shutdown, where an IPM with a built-in NTC thermistor is used, the resistance of which decreases gradually with increasing temperature. A resistance value corresponds to each given temperature. The program senses the voltage applied to the AD port. When the voltage becomes less than the voltage at the given temperature, it indicates that the NTC resistor temperature exceeds the setpoint and thus TSD is triggered.




```
64
65  #define TempProtValue                105                 // Unit: 1 degree centigrade
66  #define TempWarningOn                95                  // Unit: 1 degree centigrade
67  #define TempWarningOff               80                  // Unit: 1 degree centigrade
68  #define TempRecoverValue             80                  // Unit: 1 degree centigrade
69
```

5.2.6 Bias Voltage Protection

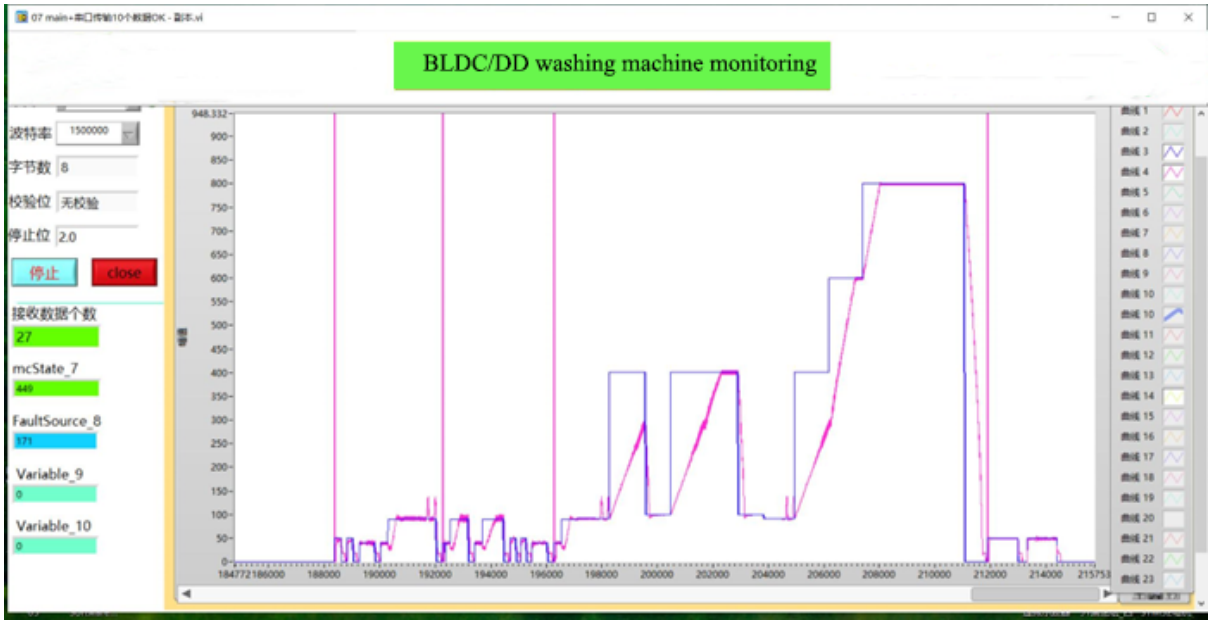
Bias voltage is measured before the motor starts up. When VHALF is attached, the theoretical value of bias voltage is 2048 which converts to 16383 after being moved left by 3 bits; when VHALF is not connected, the theoretical value of bias voltage is 0; when the error of the measured bias voltage exceeds GetCurrentOffsetValue of its theoretical value, the bias voltage is considered abnormal.

```
542         if (mcCurOffset.OffsetCount > Calib_Time)
543         {
544             //mcCurOffset.OffsetCount = 0;
545             mcCurOffset.OffsetFlag = 1;
546 
547             if(((mcCurOffset.IuOffset > 17480)  || (mcCurOffset.IuOffset < 15200))  ||
548                 ((mcCurOffset.IvOffset > 17480) || (mcCurOffset.IvOffset < 15200))  ||
549                 ((mcCurOffset.Iw_busOffset > 17480) || (mcCurOffset.Iw_busOffset < 15200)))     // The bias voltage is not within the normal range
550             {
551                     mcState = mcStop;                                                           // Perform the bias current detection again
552                     mcCurOffset.OffsetFlag = 0;
553                     mcFaultSource = FaultIbusOffset;
554 
555 
556 
557             }
558         }
559     }
```

6 Debugging of Other Common Functions

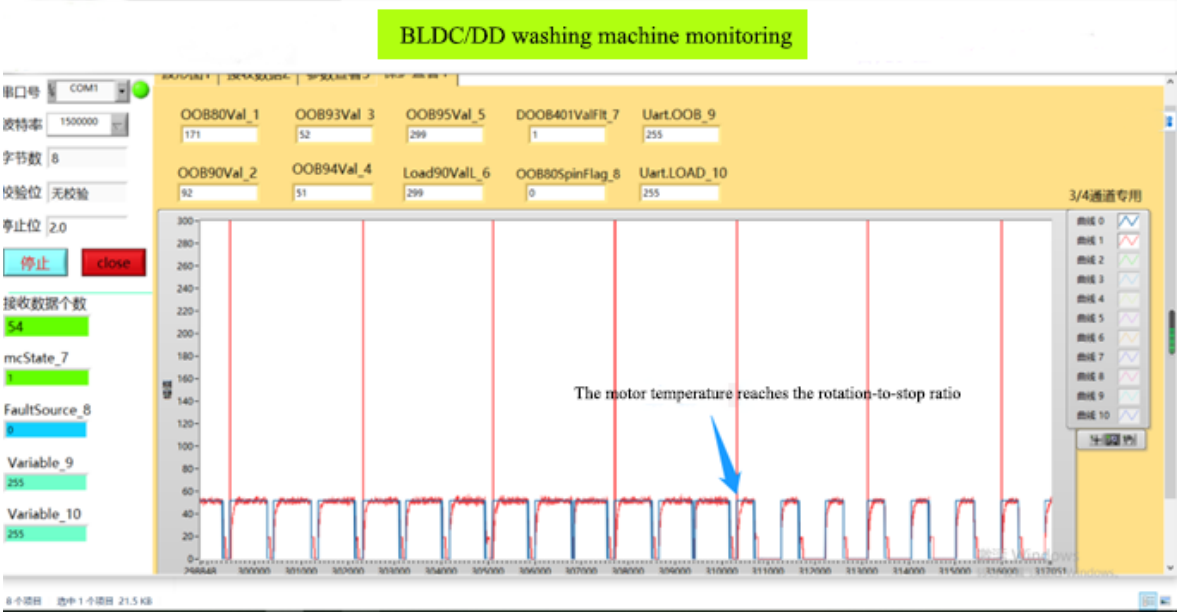
6.1.1 Dehydration Curve

The dehydration curve of a washing machine is explained as follows: Constant spin speed → OOB detection before weighing → OOB detection and Load detection → Two double-peak pre-dehydration cycles → Re-weighing to determine if the conditions for applying the maximum spin speed are met.



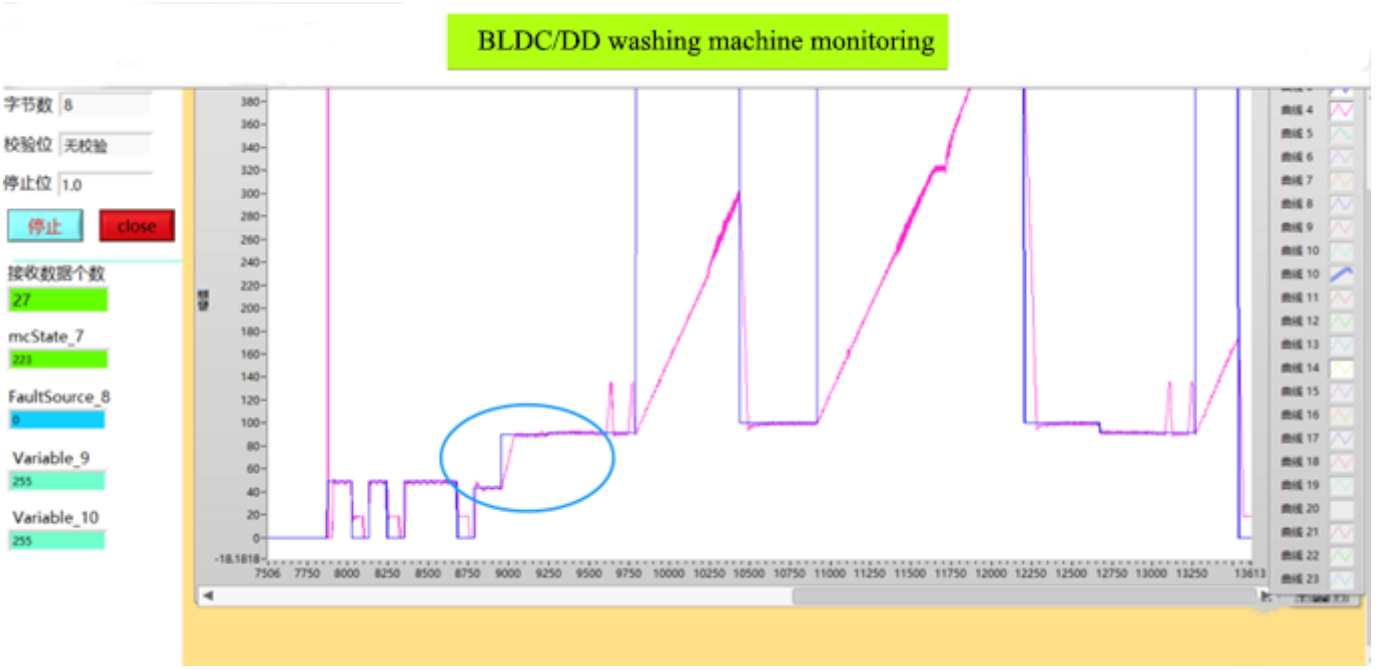
6.1.2 Motor Temperature Detection

Motor temperature detection works by measuring FOC__VALP and FOC__IA after injecting d-q-frame currents during alignment. The detection duration is set according to the customer's demand, during which period the mcFocCtrl.MotorRes value is read to match with the real temperature sensor reading. If the motor temperature detected is too high, the master computer or inverter will adjust the run-stop ratio automatically.



6.1.3 OOB and Load Detection

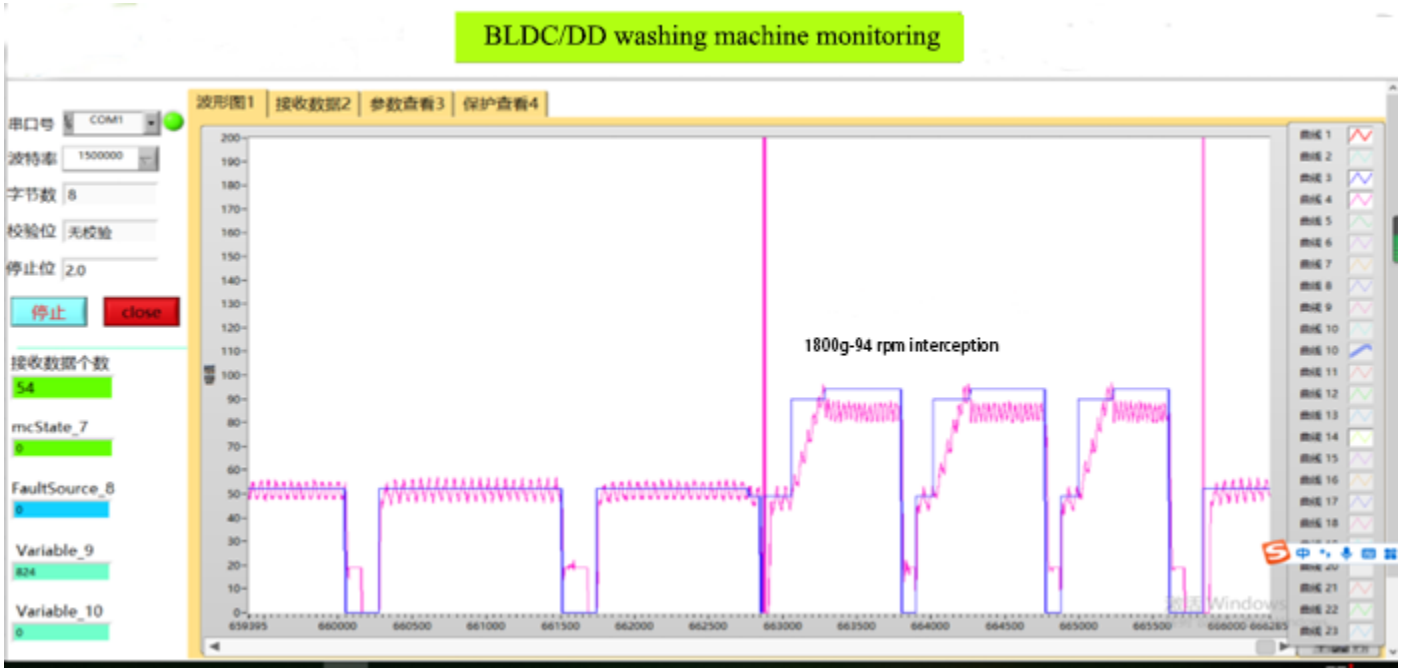
1. Protection at 80rpm: to measure power fluctuation for at least 3 seconds. This command is executed before wet laundry weighing in order to prevent drum bumping during wet laundry weighing.
2. OOB detection at 90rpm: to measure power fluctuation which is the basis for determining whether the conditions are ready for high-speed dehydration.
3. Load detection at 90rpm: to measure power fluctuation. Record the power (static friction) when the motor is running at 90rpm for OOB detection. And then accelerate the drum rotation to 130 ~ 150rpm, and calculate the power in acceleration mode. Figure out the difference between the two power values to get the power needed for acceleration. Thus, the inertia value can be inferred. In practice, standard eccentric blocks are often used for calibration.



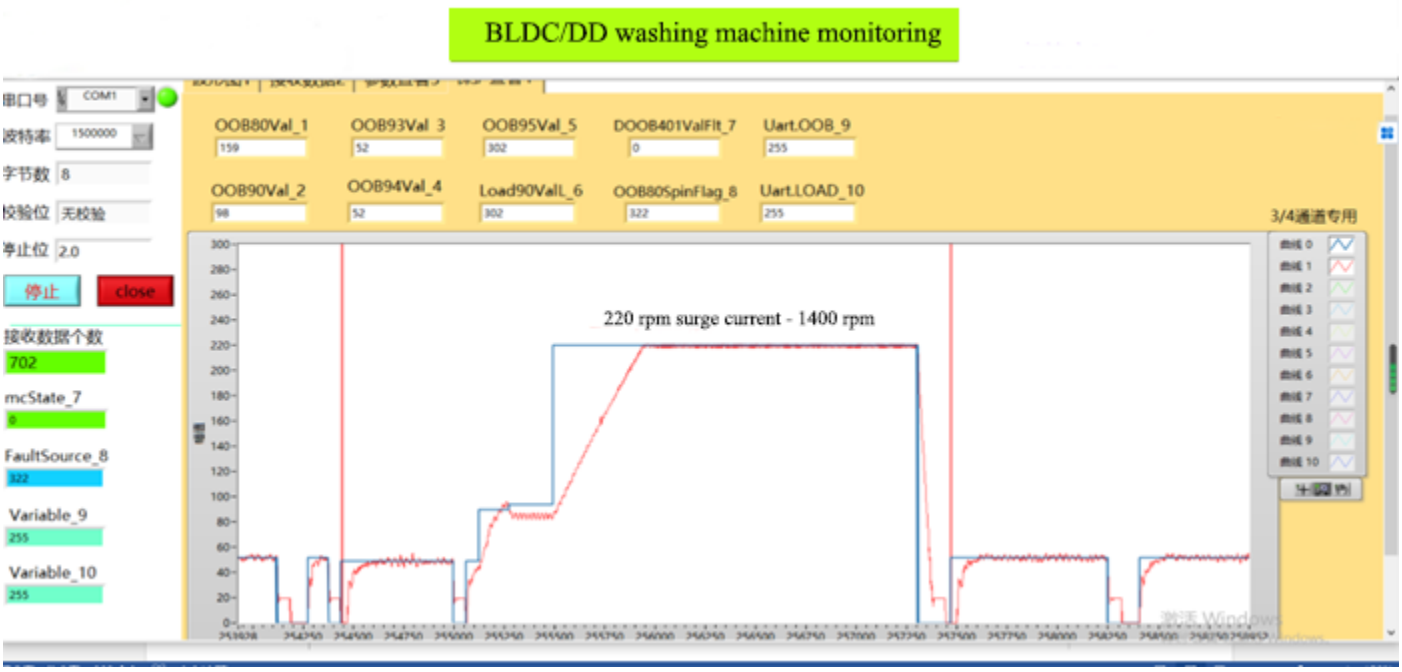
6.1.4 Hydraulic Agitation Test

Principle: to measure the power and speed fluctuations for one revolution of the inner drum. The range of speed tested should be determined and communicated to the customer. In general, the threshold can be appropriately magnified in the case of no load.

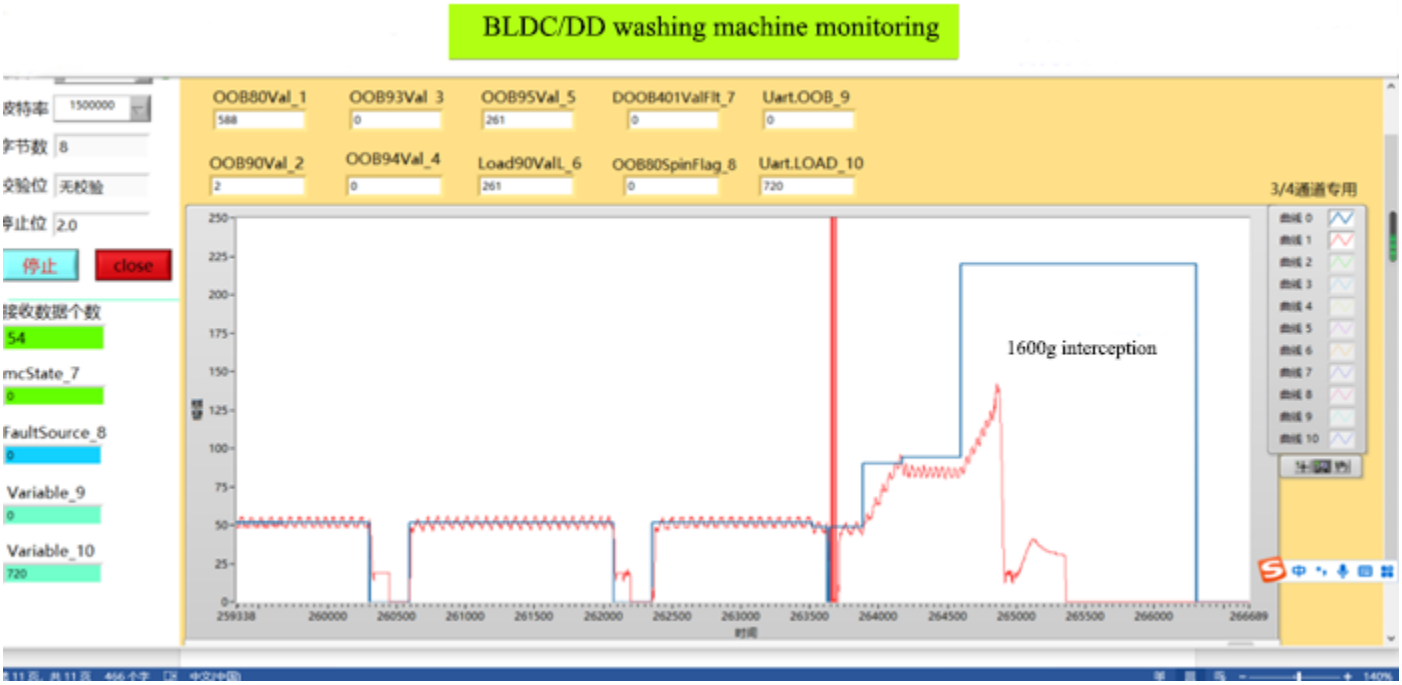
1. Low speed interception



2. High speed passing



3. High speed interception

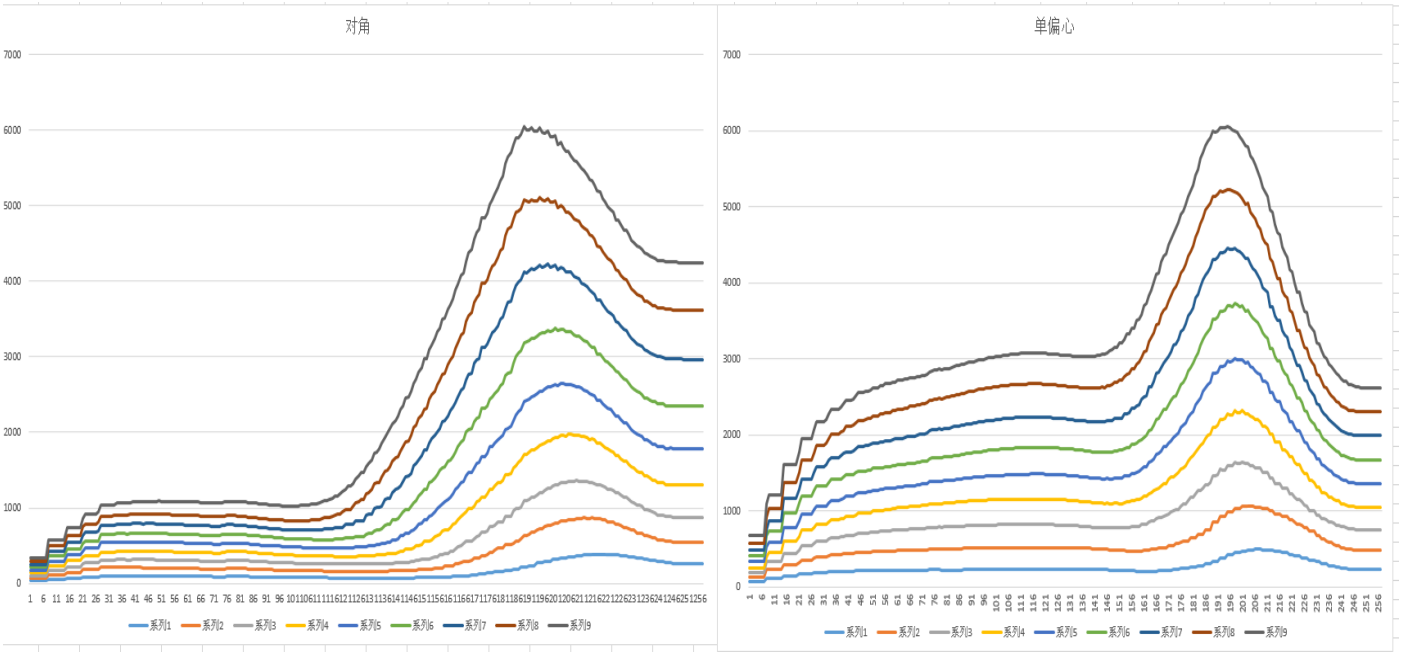


6.1.5 DOOB Detection

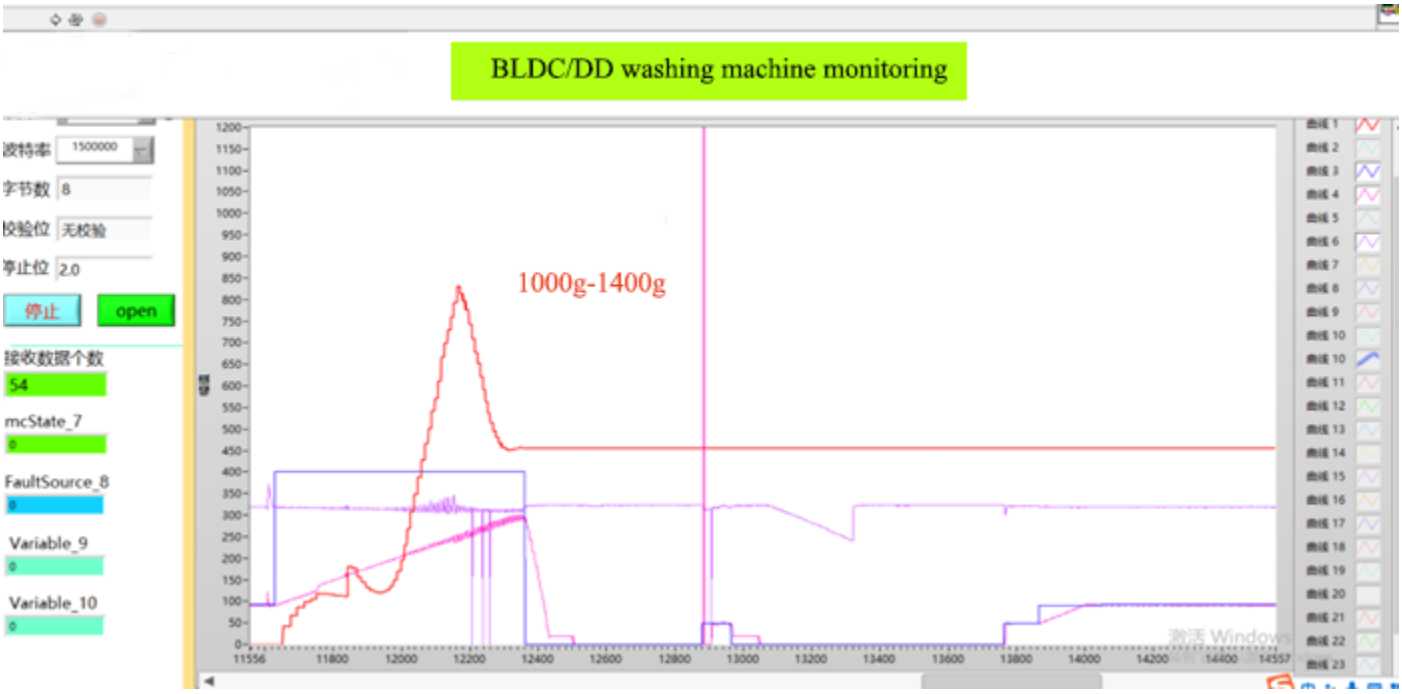
Principle: to measure the power and speed fluctuations for one revolution of the inner drum, figure out the difference between each two adjacent calculated values, and then integrate each difference. Here, the range of speed tested should be established.

The diagonal test procedures are explained as follows:

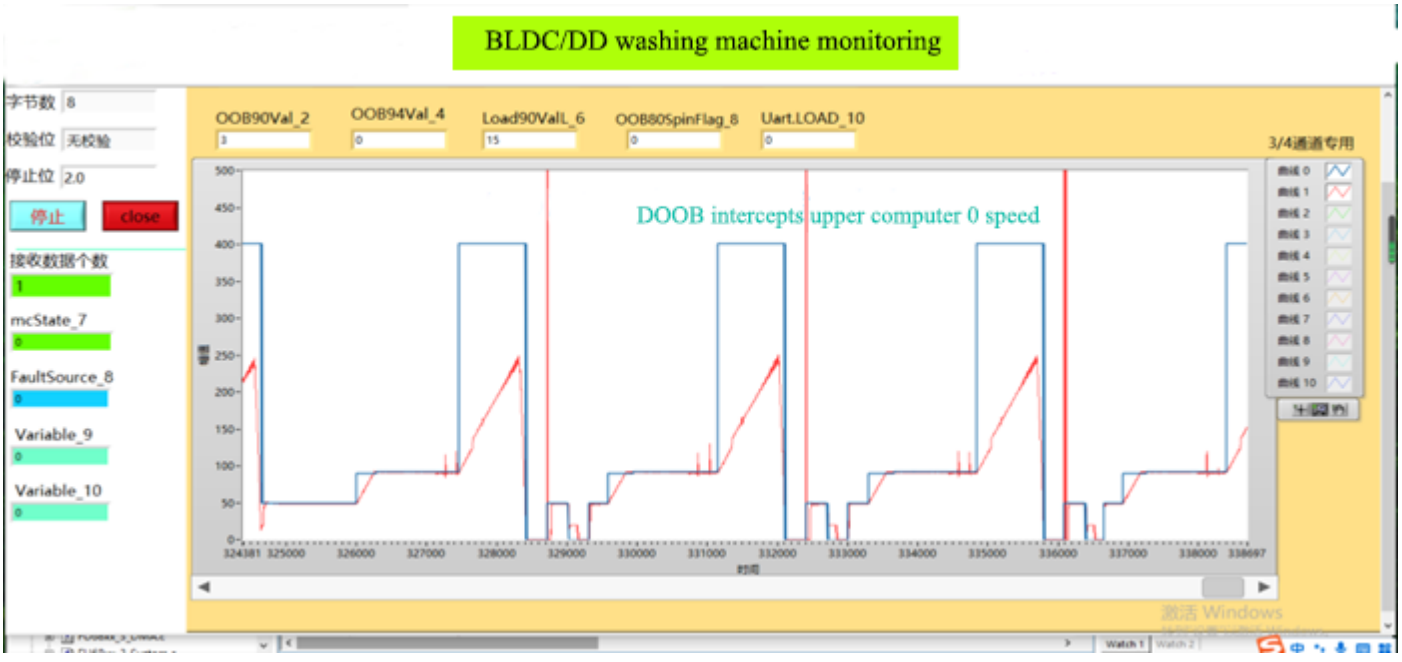
1. When placing eccentric blocks diagonally, you need to mount the eccentric blocks of the same weight to the cylinder wall, making it eccentrically symmetrical which is indicated by the measured OOB value falling within 0 ~ 10;
2. When placing eccentric blocks, you should take the measurements from eccentric blocks of 800g placed diagonally, and then increase the weight by 100g each time up to 1300g where the eccentric blocks must be held on by tape;
3. Read the DOOB.DOOBValFlt value;
4. The data of single eccentricity and diagonal eccentricity scenarios using 500*500g ~ 1300*1300g eccentric blocks is illustrated below.



As can be seen from the above data, each scenario has distinctive intervals, so a cutoff threshold of DOOB can be set according to the customer requirements in order to effectively solve the problem of high-speed drum bumping caused by low eccentricity.



When the OOB value is very high in relative to a low DOOB value, the signals will be blocked. In such case, the inverter will upload the OOB or LOAD value as 254. This value will be received by the master computer which will then send an OFF command, as shown in the following figure:

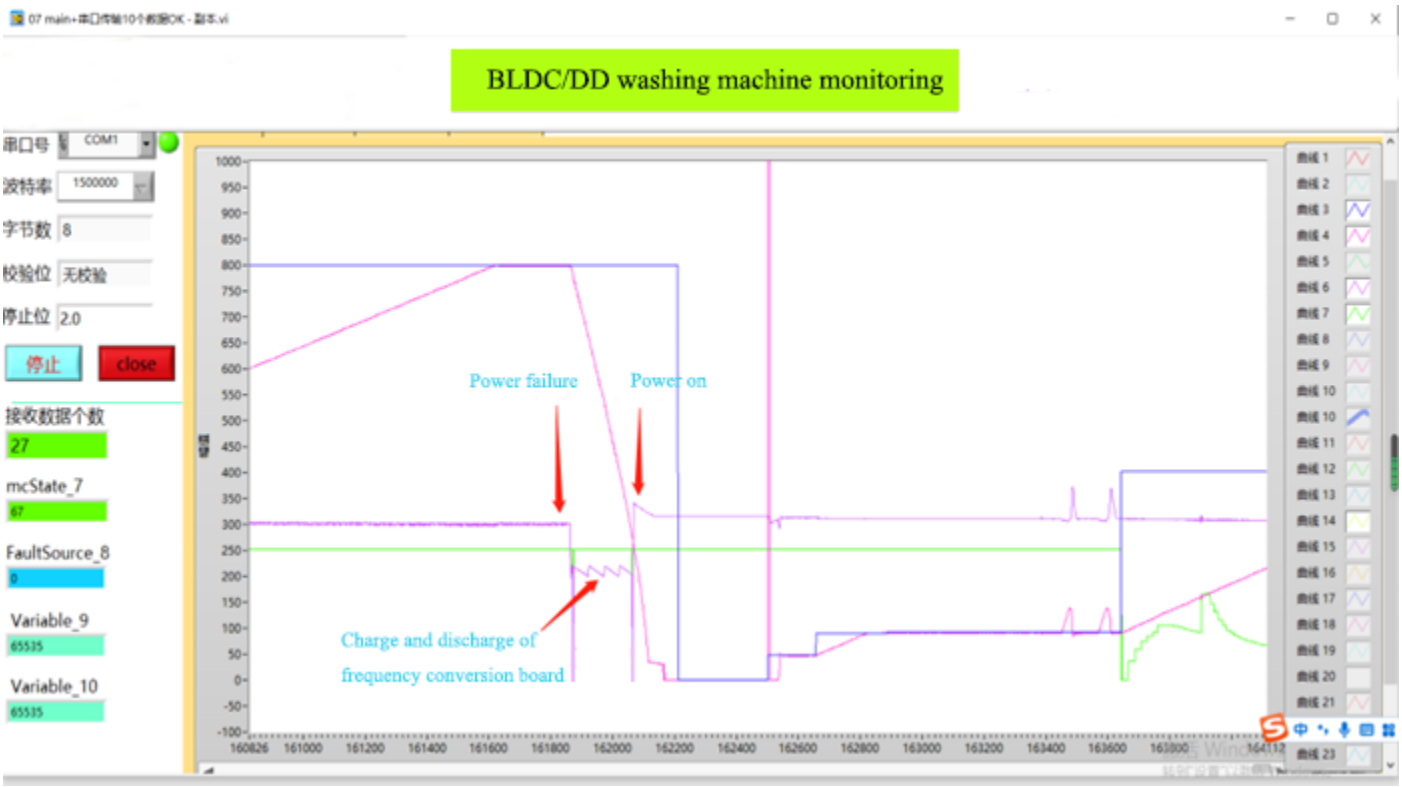


6.1.6 High Speed Weak Magnetism

In case of power failure during the weak magnetism period, the controller outputs will be turned off, and the rotor field of BLDC motor and the flux will be restored. As a result, a very high BEMF will be produced in the motor windings. This high voltage, which is connected to the controller, will damage the controller.

Solution:

1. Maintain the control system voltage to keep the MCU operating;
2. When the voltage is high, the brushes can be shorted to apply braking, where the control system consumes the energy stored in the bus capacitor. When the voltage is low, the duty cycle can be adjusted for braking.



7 Key Issues and Solutions

Constant Power Debugging	
Common Issues	**Solutions**
Startup problems are unsettled for a long time.	Users is blocked by startup issue debugging. When software issues are ruled out, users should check hardware problems such as sampling and layout.
The motor starts up abnormally in tailwind and detection is inaccurate.	Check whether the ground trace layout of BEMF detection circuit is properly arranged. Inaccurate detection usually is due to interferences caused by improper ground traces and so forth.
The response of the motor speed regulation is slow.	1. Adjust the SKP and SKI of outer loop; 2. Adjust SPEED_LOOP_TIME; 3. If only the acceleration and deceleration are relatively slow, tune the incremental value of acceleration and deceleration.
Block the air intaking port with a tool then remove it quickly. The current gets large and overcurrent occurs.	In general, it is caused by slow response of inner current loop. Users can increase the PI of current loop, namely DQKP and DQKI.
Speed or power cannot meet customer requirements.	1. When current is sine waveform, observe whether FOC__UQ is saturated. 2.If FOC_UQ is saturated and FOC__UD is relatively large, adjust compensation angle FOC_THECOMP (try both positive angle and negative angle); see whether customer needs are met. 3.If customer needs are not met yet by above solutions, overmodulation can be considered. However, it is not suggested doing so. If the target power is unreachable due to motor issues, recommend discussing with the customer to adjust the winding design of the motor.
When the motor runs to a high speed, it's easy to cause large current.	1. Adjust compensation angle FOC_THECOMP; 2. Shift the sampling point, that is, change the sampling point delay time FOC_TRGDLY.
There is sine distortion in current waveforms.	1.Check whether the sampling bias reference is normal; 2.Adjust the PI of current loop, namely DQKP, DQKI; 3.Adjust the sampling point delay time FOC_TRGDLY; 4.Adjust carrier frequency (note that the modification could affect both startup and run operations).
Notes: In general, all the parameter adjustment affects performance of startup and run. Users need to re-test and double check after problems are resolved.	

8 Revision History

Revision	Changes	Effective Date	Revised by
V1.2	First release of EU version, converted from English version V1.2	2023/03/01	Bobsare

Copyright Note

Fortior Technology Co., Ltd.
(Singapore)： 1003 Bukit Merah Central, #04-22, INNO Center,(s)159836
Customer service:info@fortiortech.com
URL: http://www.fortiortech.com/global/