Vacuum Cleaner Motor Debugging Manual

3-phase Motor Control MCU
FU6861-Q2

Fortior Technology (Shenzhen) Co., Ltd

Content

Content **2**
1 Overview **4**
2 Hardware **5**
2.1 Hardware Schematic Diagram 5
2.1.1 Power Circuit 6
2.1.2 Chip Circuit 6
2.1.3 BEMF Detection Circuit 7
2.1.4 Power Drive Circuit 7
2.1.5 OP Amp Configuration Circuit 8
2.1.6 BUS Voltage Sampling Circuit 8
3 Software Architecture **9**
3.1 Motor State Machine Flowchart 9
3.2 Program Flowchart 11
3.3 Program Description 11
3.3.1 Main Function 11
3.3.2 1ms Timer Interrupt 11
3.3.3 FOC Interrupt 12
3.3.4 CMP3 Interrupt 12
3.3.5 Timer3 Interrupt 12
4 Debugging Steps **13**
4.1 Motor Parameter Configuration 13
4.1.1 Motor Parameters 13
4.1.2 Motor Parameter Measurement Method 13
4.1.3 Corresponding Program Code 14
4.2 Chip Internal Parameter Configuration 14
4.3 Hardware Parameter Configuration 14
4.4 Protection Parameter Configuration 15
4.5 Startup Parameter Configuration 16
4.6 Hardware Driver Circuit Detection 18
4.7 Current Loop Debugging 18
4.8 Power Loop Debugging 20
4.9 PWM Function Debugging 21
4.10 Other Functions 23
4.11 Reliability Test 23

4.11.1 Reliability of Function 23
4.11.2 Reliability of Protection 23
4.11.3 Stability of Startup 23
5 Function Introduction 24
5.1 Startup Debugging 24
5.1.1 Omega Startup 24
5.1.2 Common Issues & Solutions in Starting 26
5.2 Introduction to Protection Functions 26
5.2.1 Overcurrent Protection 26
5.2.2 Voltage Protection 27
5.2.3 Phase Loss Protection 27
5.2.4 Motor Locked Protection 28
5.2.5 Over Temperature Protection 29
5.2.6 Overspeed Protection 30
5.2.7 Bias Voltage Protection 30
5.2.8 Other Protections 30
6 Other Common Function Debugging 31
6.1 Speed Limiting Function 31
7 Key Issues and Solutions 32
8 Revision History 33
9 Copyright Notice 34

1 Overview

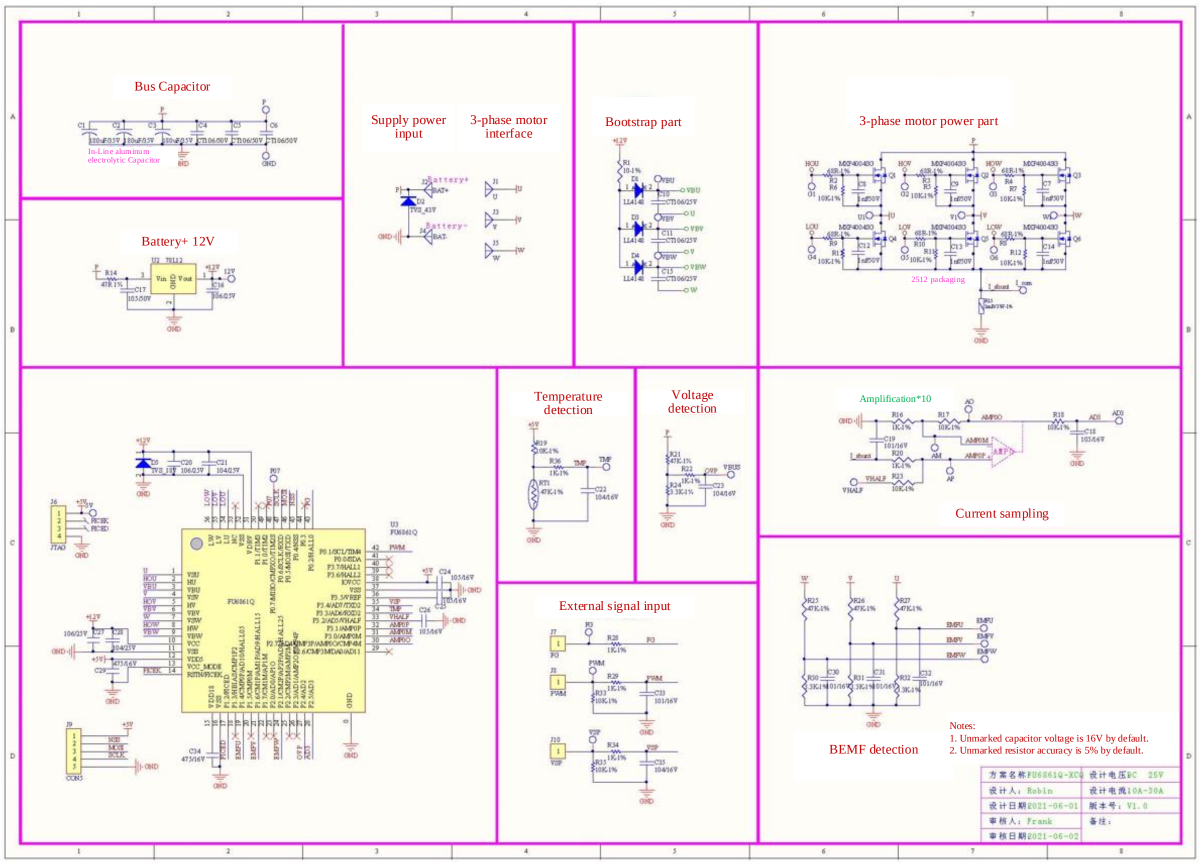
This debugging manual discusses how to use Fortior FU6861Q2 to conduct sensorless FOC drive control of a BLDC motor used in vacuum cleaner. The debugging is performed on a dedicated DEMO board for vacuum cleaner motor. Users can have a quick browse of hardware principles in Chapter 2 and software principles in Chapter 3 first, then focus on the debugging steps in Chapter 4.

Software and Hardware Involved

Software/ hardware	Name	Related Chapters	Notes
Software	FU-AM-FU6861-B-025-SW-V1.0.00-20220707	All	Debugging is conducted on this software
Hardware	FU-AM-FU6861-B-025-HW-V1.0.00-20210601	All	Debugging is conducted on this hardware

2 Hardware

2.1 Hardware Schematic Diagram



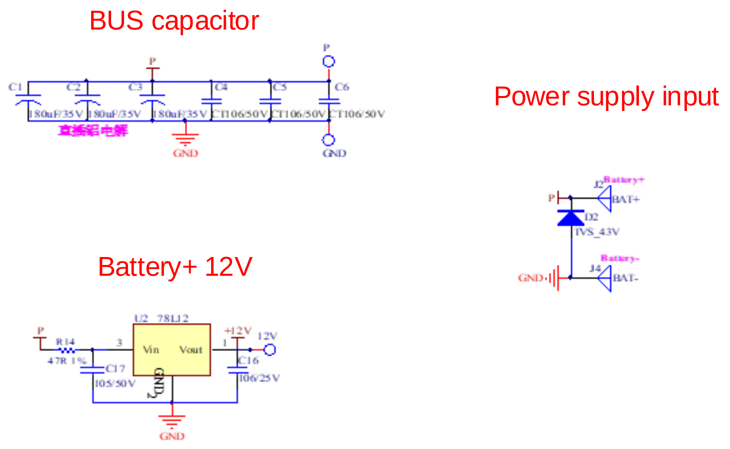
Hardware usage:

The board is a dedicated DEMO board for vacuum cleaner motor applications. Users can power on and use it directly.

Notes:

Users need to properly configure BUS voltage ratio, amplifier magnification, sampling resistance, and the voltage divider ratio of BEMF detection circuit, according to the amount of motor voltage and current.

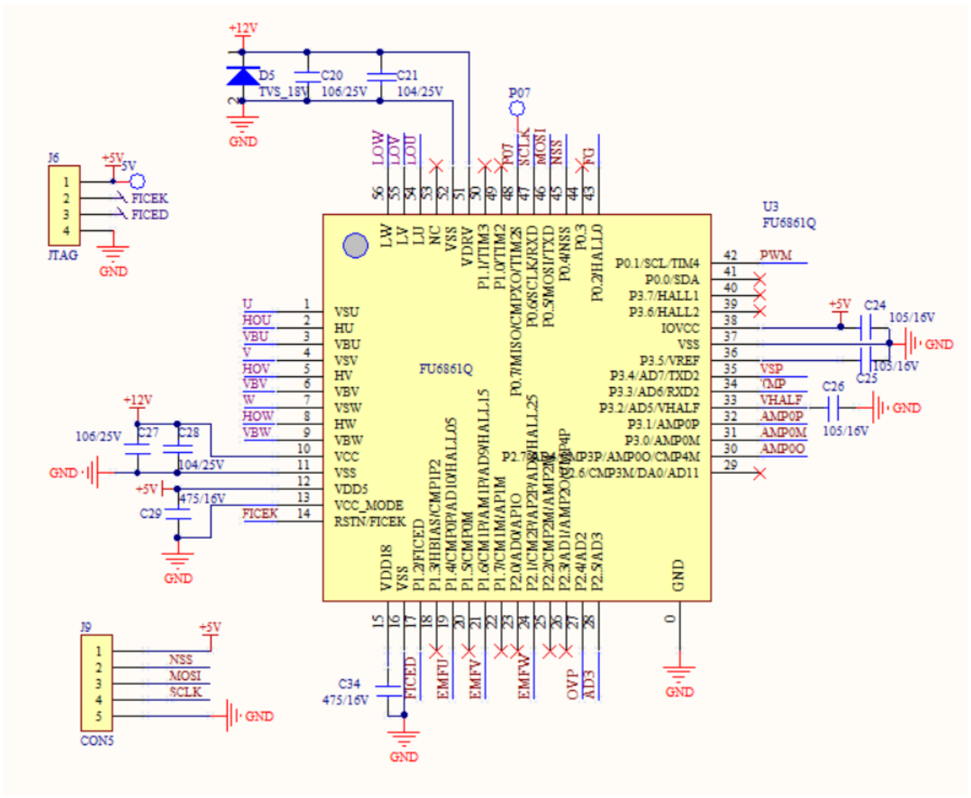
2.1.1 Power Circuit



Power Usage:

The positive terminal of DC power supply is connected to terminal P, and the negative is connected to GND.

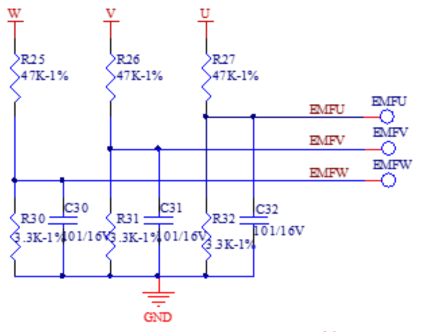
2.1.2 Chip Circuit



Chip Usage:

FU6861Q2 is applied in medium and low voltage based 6-NMOSFET drive applications. Chip pin J6 is programming interface, and J9 is SPI interface.

2.1.3 BEMF Detection Circuit



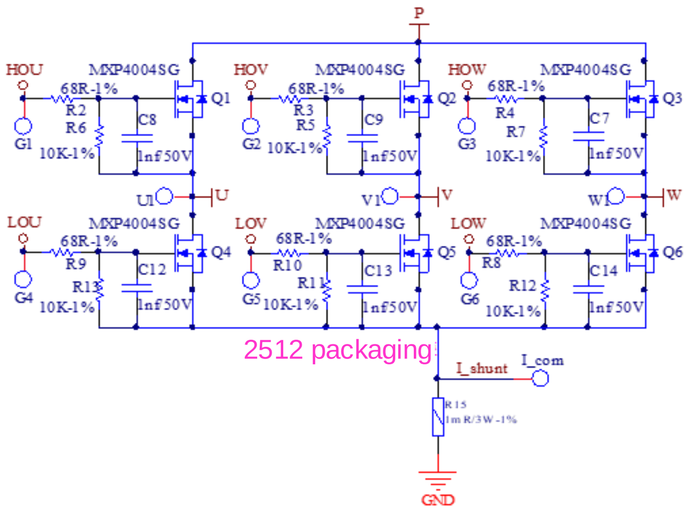
BEMF detection

Notes:
1. Unmarked capacitor voltage is 16V by default.
2. Unmarked resistor accuracy is 5% by default.

Notes:
Since the circuit is to detect BEMF, tailwind/headwind detection method can be selected as BEMF detection only.

2.1.4 Power Drive Circuit

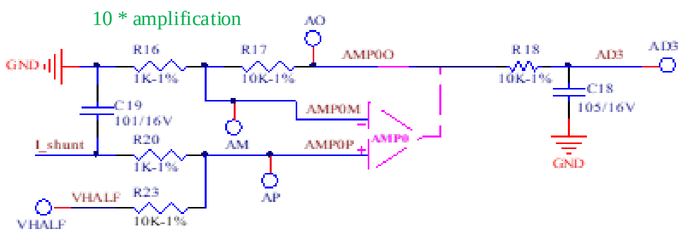
Power part of 3-phase motor



Notes:

In the case of maximum current, the power consumed by the sampling resistor cannot exceed 80% of the rated power.

2.1.5 OP Amp Configuration Circuit



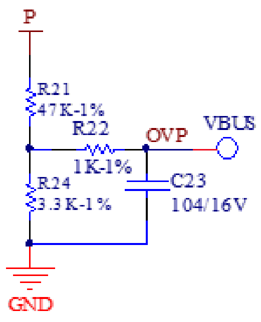
Current sampling

Notes:

1. C19 value shouldn't be adjusted, and should be with an accuracy of 10%;
2. R16, R17, R20 and R23 should apply resistors with an accuracy of 1%;
3. AD3 is for average BUS voltage sampling;
4. Magnification = R17/R16 = R23/R20;
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 BUS Voltage Sampling Circuit

Voltage detection



Notes:

1. R22, C23 shouldn't be adjusted;
2. R21, R24 should apply resistors with an accuracy of 1%;
3. Maximum sampling voltage = (R21 + R24)/(R24)*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 Architecture

3.1 Motor State Machine Flowchart

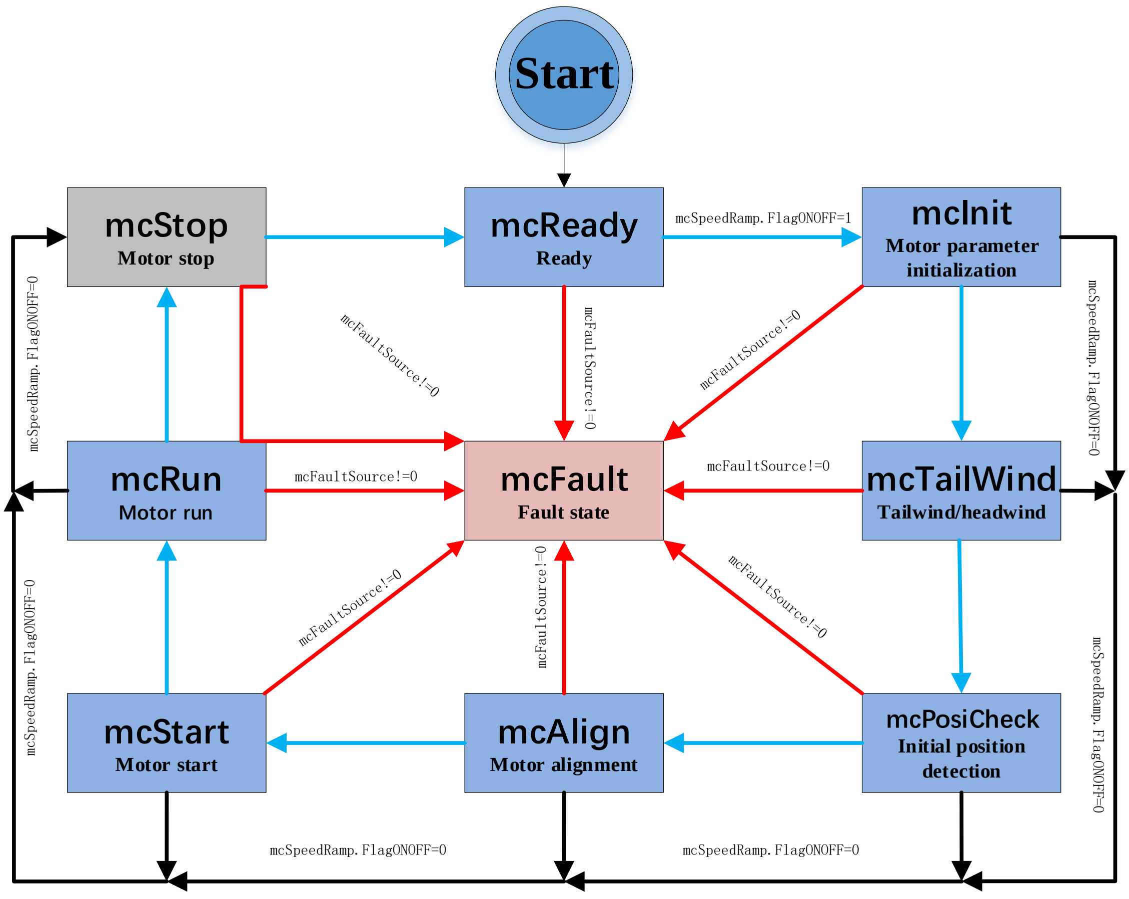


Figure 3-1 Motor State Machine Flowchart

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

1. Run: mcReady -> mcInit -> mcTailWind -> mcPosiCheck -> mcAlign -> mcStart -> mcRun;
2. Stop: In the states of mcInit, mcCharge, 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. mcTailWind: the state is to detect tailwind/headwind. When tailwind is detected, it enters mcRun state to run directly; when headwind is detected, it brakes first then proceeds (headwind is not applicable to vacuum cleaner motors); if neither tailwind nor headwind is detected, it proceeds then.
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, mcTailWind, 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 mcTailWind to mcStart directly.

3.2 Program Flowchart

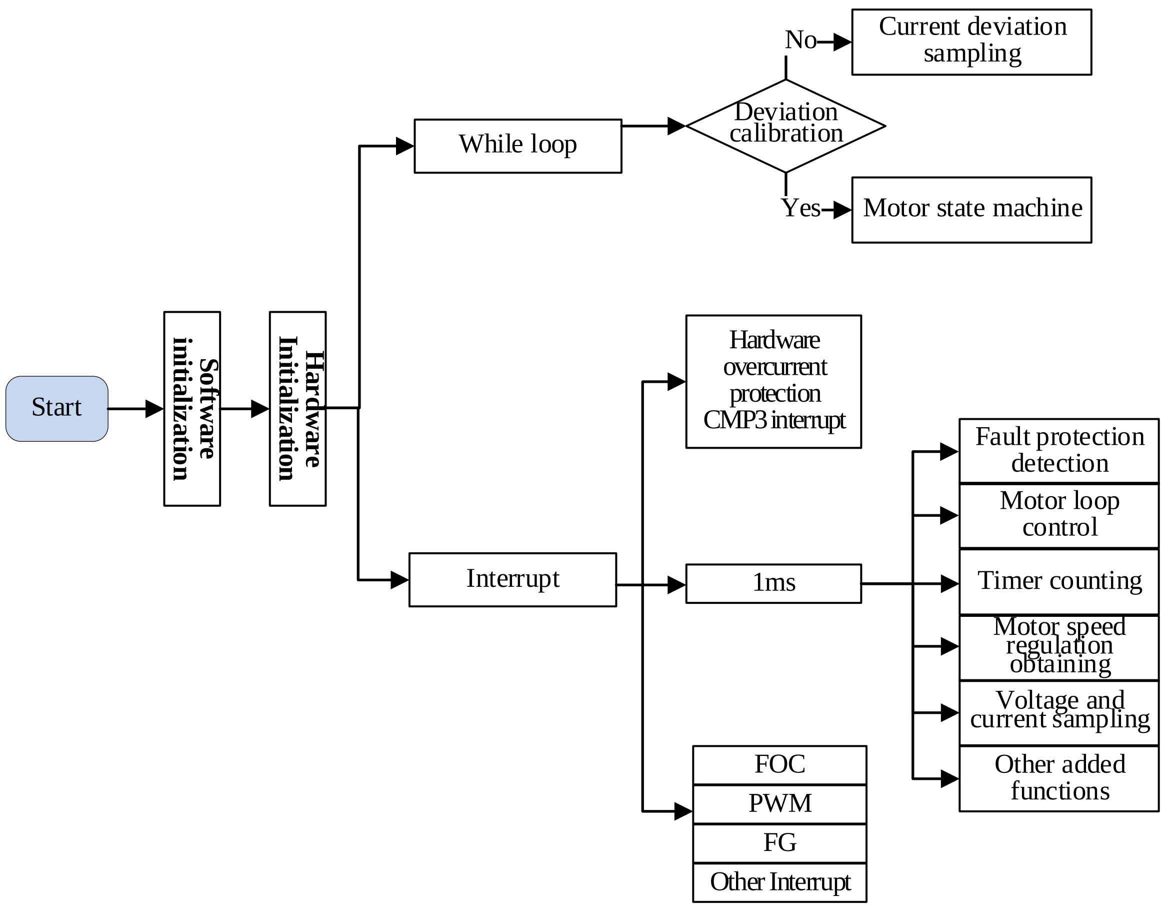


Figure 3-2 Program Flowchart

3.3 Program Description

3.3.1 Main Function

Program initialization -> GetCurrentOffset() for bias voltage detection + MC_Control() for motor control.

3.3.2 1ms Timer Interrupt

In the program, functions such as speed regulation, fault protection detection, BUS current sampling and BUS voltage sampling are all called in 1ms interrupt, with the following functions involved:

```
Speed_response();          // loop control function
PWMInputCapture();         // PWM speed regulation function
VSPSample();               // Analog voltage based speed regulation function
KeyScan();                 // Button based speed regulation function
ONOFF_Starttest();         // Motor on/off test to verify startup reliability
StarRampDealwith();        // ATO ramping control during motor startup
Fault_Detection();         // fault detection
```

3.3.3 FOC Interrupt

FOC interrupt, namely carrier interrupt, is majorly to handle relatively fast-timing programs such as calling a divider.

3.3.4 CMP3 Interrupt

Comparator 3 interrupt is majorly to handle hardware overcurrent protection. Please refer to Section 5.2.1 for more details.

3.3.5 Timer3 Interrupt

Timer3 interrupt is to obtain PWM duty cycle. PWM high level TIM3__DR and PWM period value TIM3__ARR are obtained through the interrupt, then PWM duty cycle is calculated accordingly.

4 Debugging Steps

4.1 Motor Parameter Configuration

4.1.1 Motor Parameters

1. The number of motor pole pairs: 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.

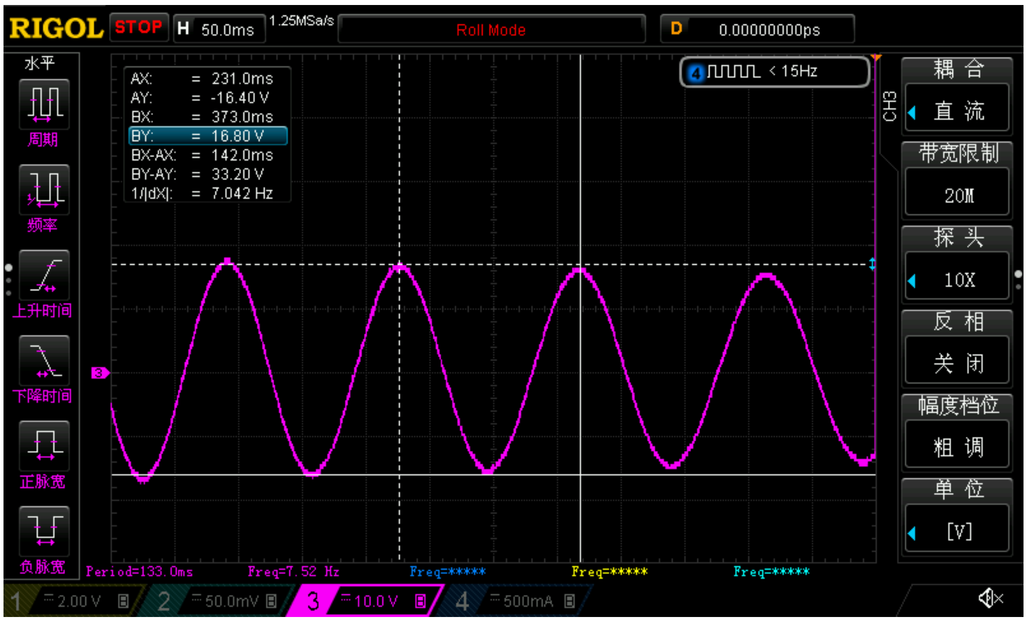
4.1.2 Motor Parameter Measurement Method

1. The number of pole pairs Pole_Pair: 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:

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

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

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



```
AX:      =  231.0ms
AY:      =  -16.40 V
BX:      =  373.0ms
BY:      =  16.80 V
BX-AX:   =  142.0ms
BY-AY:   =  33.20 V
1/|dX|:  =  7.042 Hz
```

```
Period=133.0ms   Freq=7.52 Hz   Freq=*****   Freq=*****   Freq=*****
```

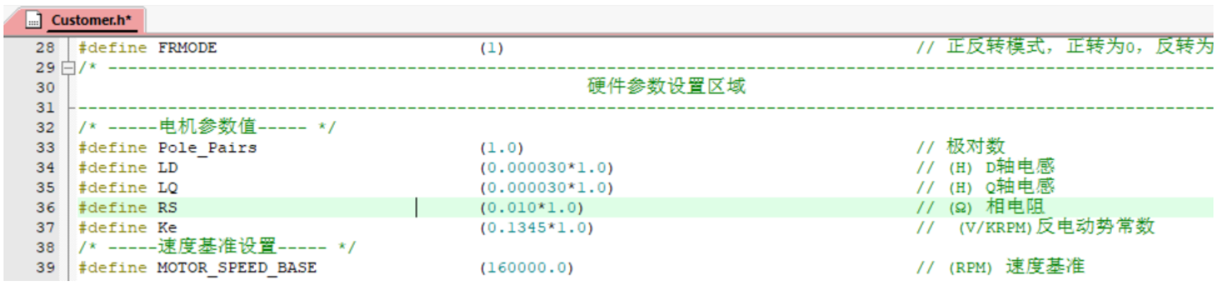
Figure 4-1 BEMF Waveform

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.

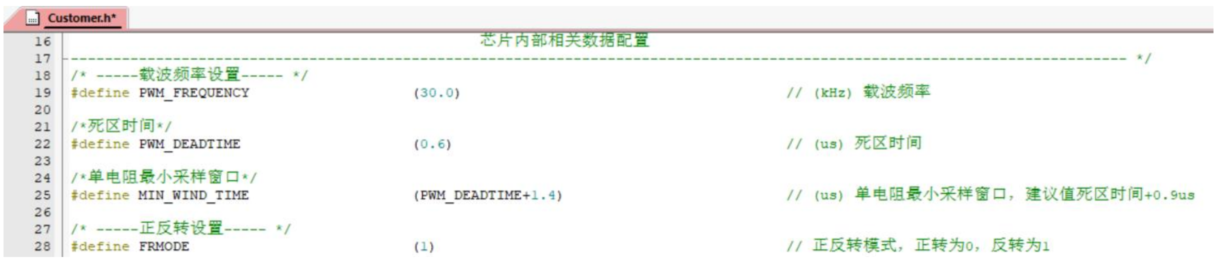
4.1.3 Corresponding Program Code



Customer.h*

```
#define FRMODE                        (1)                          // 正反转模式，正转为0，反转为
/* -----------------------------------------------------------------------------------------------
                                        硬件参数设置区域
-------------------------------------------------------------------------------------------------
/* -----电机参数值----- */
#define Pole_Pairs                    (1.0)                        // 极对数
#define LD                            (0.000030*1.0)               // (H) D轴电感
#define LQ                            (0.000030*1.0)               // (H) Q轴电感
#define RS                            (0.010*1.0)                  // (Ω) 相电阻
#define Ke                            (0.1345*1.0)                 //  (V/KRPM)反电动势常数
/* -----速度基准设置----- */
#define MOTOR_SPEED_BASE              (160000.0)                   // (RPM) 速度基准
```

4.2 Chip Internal Parameter Configuration



Customer.h*

```
                                        芯片内部相关数据配置
------------------------------------------------------------------------------------------- */
/* -----载波频率设置----- */
#define PWM_FREQUENCY                 (30.0)                       // (kHz) 载波频率

/*死区时间*/
#define PWM_DEADTIME                  (0.6)                        // (us) 死区时间

/*单电阻最小采样窗口*/
#define MIN_WIND_TIME                 (PWM_DEADTIME+1.4)           // (us) 单电阻最小采样窗口，建议值死区时间+0.9us

/* -----正反转设置----- */
#define FRMODE                        (1)                          // 正反转模式，正转为0，反转为1
```

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. Vacuum cleaner motors generally rotate at a relatively high speed, the default value 30K can be used to start debugging;
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. Forward and reverse rotation setting is configured according to actual wiring. High-frequency noise is caused when vacuum cleaner motors rotate reversely and airflow is much smaller in reverse rotation than in forward rotation. Set the bit reverse in the reverse rotation case.

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.

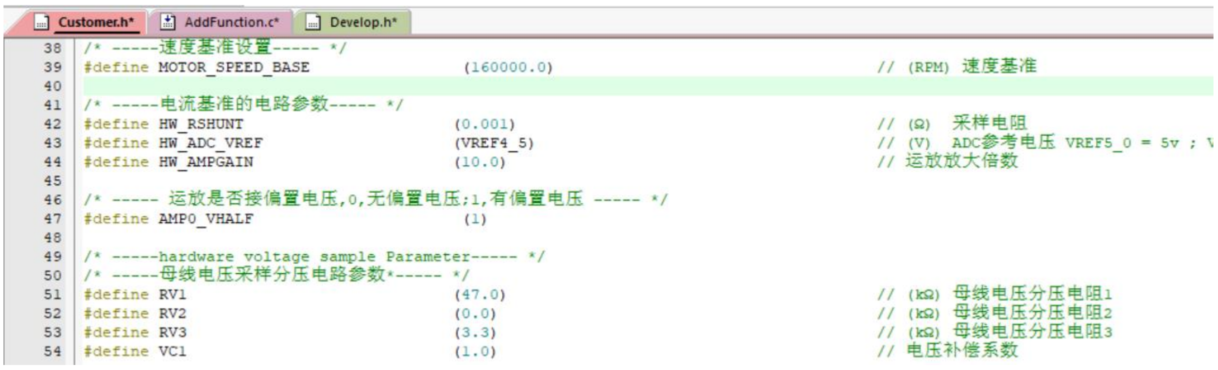
2) Sampling resistance and magnification:

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.

HW_RSHUNT represents sampling resistance and HW_AMPGAIN represents magnification.

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



```
/* -----速度基准设置----- */
#define MOTOR_SPEED_BASE              (160000.0)                              // (RPM) 速度基准

/* -----电流基准的电路参数----- */
#define HW_RSHUNT                    (0.001)                                 // (Ω)  采样电阻
#define HW_ADC_VREF                  (VREF4_5)                               // (V)  ADC参考电压 VREF5_0 = 5v ; V
#define HW_AMPGAIN                   (10.0)                                  // 运放放大倍数

/* ----- 运放是否接偏置电压,0,无偏置电压;1,有偏置电压 ----- */
#define AMP0_VHALF                    (1)

/* -----hardware voltage sample Parameter----- */
/* -----母线电压采样分压电路参数*----- */
#define RV1                          (47.0)                                  // (kΩ) 母线电压分压电阻1
#define RV2                          (0.0)                                   // (kΩ) 母线电压分压电阻2
#define RV3                          (3.3)                                   // (kΩ) 母线电压分压电阻3
#define VC1                          (1.0)                                   // 电压补偿系数
```

Where,

1) BUS voltage divider ratio = (RV1 + RV2 + RV3)/RV3;
2) VC1 is voltage compensation coefficient. It is used in startup stage only. Just leave it unchanged so far.

4.4 Protection Parameter Configuration

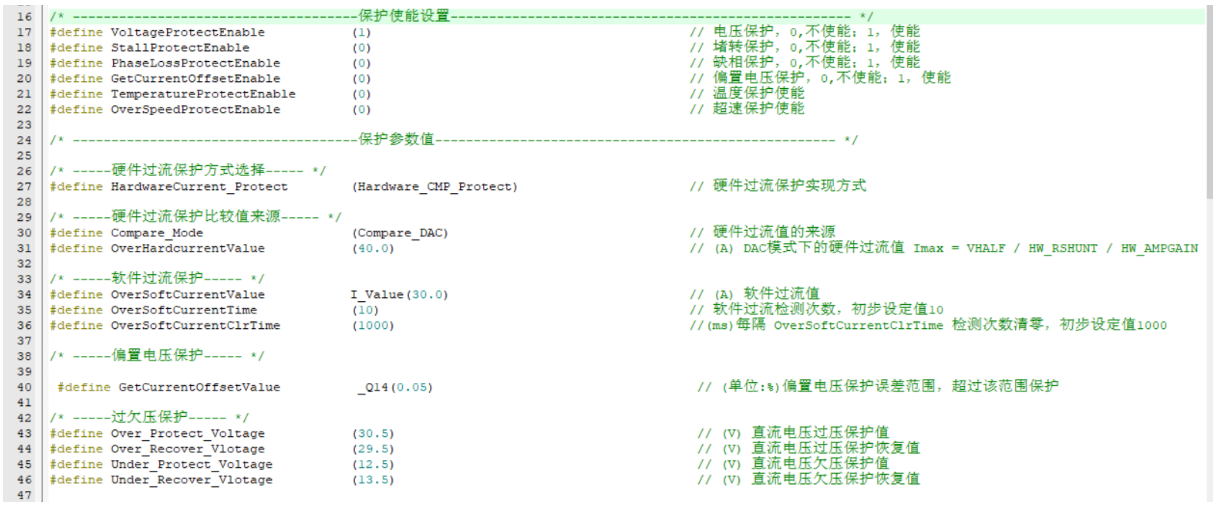
1. Current protection settings:

- 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.

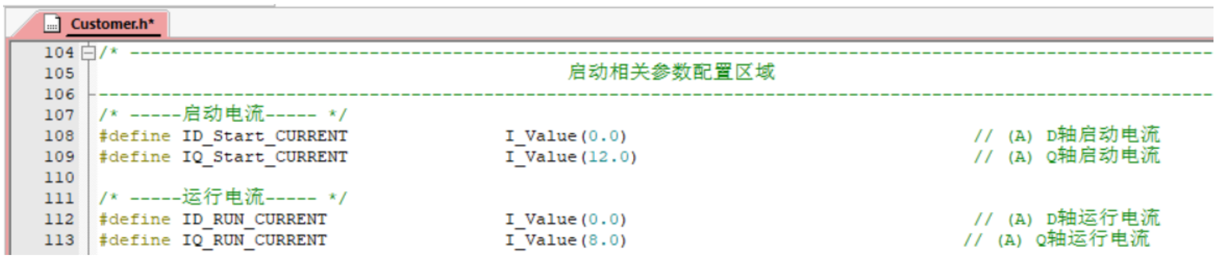
2. Set overvoltage/undervoltage protection and protection recovery parameters. Please refer to Section 5.2.2 for details;
3. Turn off all protections except the above to prevent false triggering during startup. Apply other protections later when they are required. As for overcurrent protection, it is always on with no enable bit.
4. Fill the parameters into the program code accordingly (in the Protect.h file).



```
/* -------------------------------------保护使能设置------------------------------------------------- */
#define VoltageProtectEnable            (1)                                     // 电压保护，0,不使能；1，使能
#define StallProtectEnable              (0)                                     // 堵转保护，0,不使能；1，使能
#define PhaseLossProtectEnable          (0)                                     // 缺相保护，0,不使能；1，使能
#define GetCurrentOffsetEnable          (0)                                     // 偏置电压保护，0,不使能：1，使能
#define TemperatureProtectEnable        (0)                                     // 温度保护使能
#define OverSpeedProtectEnable          (0)                                     // 超速保护使能

/* -------------------------------------保护参数值----------------------------------------------- */

/* -----硬件过流保护方式选择----- */
#define HardwareCurrent_Protect         (Hardware_CMP_Protect)                  // 硬件过流保护实现方式

/* -----硬件过流保护比较值来源----- */
#define Compare_Mode                    (Compare_DAC)                           // 硬件过流值的来源
#define OverHardcurrentValue            (40.0)                                  // (A) DAC模式下的硬件过流值 Imax = VHALF / HW_RSHUNT / HW_AMPGAIN

/* -----软件过流保护----- */
#define OverSoftCurrentValue            I_Value(30.0)                           // (A) 软件过流值
#define OverSoftCurrentTime             (10)                                    // 软件过流检测次数，初步设定值10
#define OverSoftCurrentClrTime          (1000)                                  //(ms)每隔 OverSoftCurrentClrTime 检测次数清零，初步设定值1000

/* -----偏置电压保护----- */

 #define GetCurrentOffsetValue           _Q14(0.05)                              // (单位:%)偏置电压保护误差范围，超过该范围保护

/* -----过欠压保护----- */
#define Over_Protect_Voltage            (30.5)                                   // (V) 直流电压过压保护值
#define Over_Recover_Vlotage            (29.5)                                   // (V) 直流电压过压保护恢复值
#define Under_Protect_Voltage           (12.5)                                   // (V) 直流电压欠压保护值
#define Under_Recover_Vlotage           (13.5)                                   // (V) 直流电压欠压保护恢复值
```

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 Section 5.1 for parameter adjustment details to settle down common startup issues.



Customer.h*

```
/* -----------------------------------------------------------------------------------------------
                                      启动相关参数配置区域
-------------------------------------------------------------------------------------------------
/* -----启动电流----- */
#define ID_Start_CURRENT              I_Value(0.0)                      // (A) D轴启动电流
#define IQ_Start_CURRENT              I_Value(12.0)                     // (A) Q轴启动电流

/* -----运行电流----- */
#define ID_RUN_CURRENT                I_Value(0.0)                      // (A) D轴运行电流
#define IQ_RUN_CURRENT                I_Value(8.0)                      // (A) Q轴运行电流
```

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

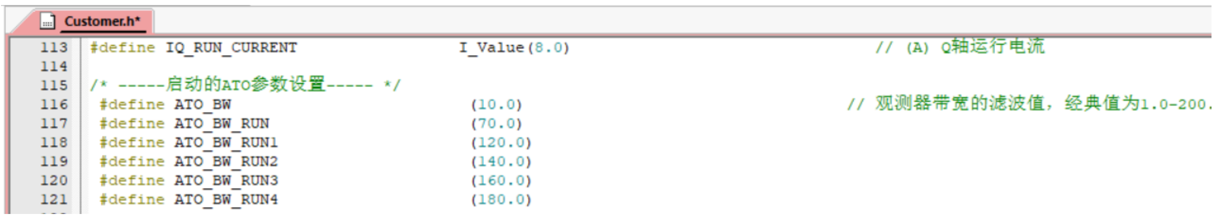
Notes:

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;

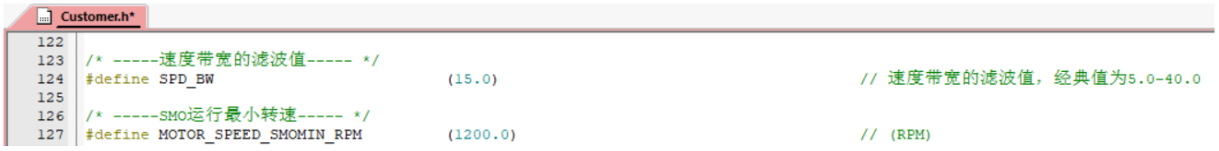


```
Customer.h*
#define IQ_RUN_CURRENT                  I_Value(8.0)                  // (A) Q轴运行电流

/* -----启动的ATO参数设置----- */
 #define ATO_BW                          (10.0)                       // 观测器带宽的滤波值，经典值为1.0-200.
 #define ATO_BW_RUN                      (70.0)
 #define ATO_BW_RUN1                     (120.0)
 #define ATO_BW_RUN2                     (140.0)
 #define ATO_BW_RUN3                     (160.0)
 #define ATO_BW_RUN4                     (180.0)
```

Notes:

For vacuum cleaner motors, the first three ATOs have obvious impact, which need to be adjusted per real situations. Since AO observer is turned on, make the first ATO_BW not too large.

4. Speed-bandwidth filtering value SPD_BW;

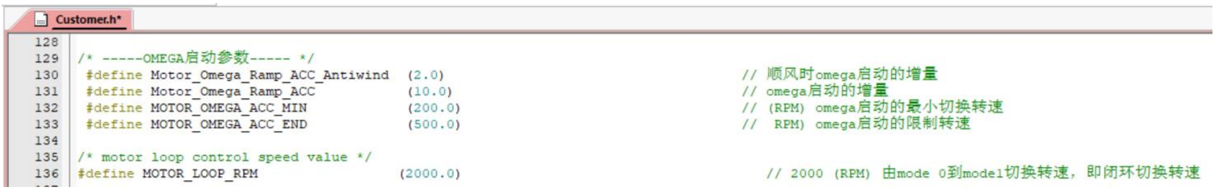


```
Customer.h*

/* -----速度带宽的滤波值----- */
#define SPD_BW                          (15.0)                        // 速度带宽的滤波值，经典值为5.0-40.0

/* -----SMO运行最小转速----- */
#define MOTOR_SPEED_SMOMIN_RPM          (1200.0)                      // (RPM)
```

Notes:

In general, it’s unnecessary to adjust SPD_BW.

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



```
Customer.h*

/* -----OMEGA启动参数----- */
 #define Motor_Omega_Ramp_ACC_Antiwind  (2.0)                         // 顺风时omega启动的增量
 #define Motor_Omega_Ramp_ACC           (10.0)                        // omega启动的增量
 #define MOTOR_OMEGA_ACC_MIN            (200.0)                       // (RPM) omega启动的最小切换转速
 #define MOTOR_OMEGA_ACC_END            (500.0)                       //  RPM) omega启动的限制转速

/* motor loop control speed value */
#define MOTOR_LOOP_RPM                 (2000.0)                        // 2000 (RPM) 由mode 0到mode1切换转速，即闭环切换转速
```

Notes:

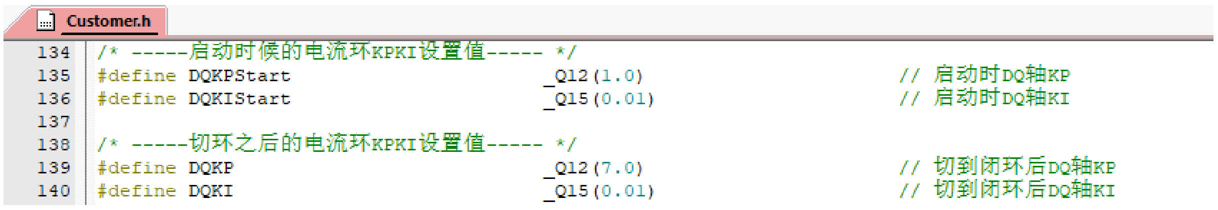
1) The reference value range of Motor_Omega_Ramp_ACC is 10 ~ 50;

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;



```
Customer.h
/* -----启动时候的电流环KPKI设置值----- */
#define DQKPStart                       _Q12(1.0)                     // 启动时DQ轴KP
#define DQKIStart                       _Q15(0.01)                    // 启动时DQ轴KI

/* -----切环之后的电流环KPKI设置值----- */
#define DQKP                            _Q12(7.0)                     // 切到闭环后DQ轴KP
#define DQKI                            _Q15(0.01)                    // 切到闭环后DQ轴KI
```

Notes:

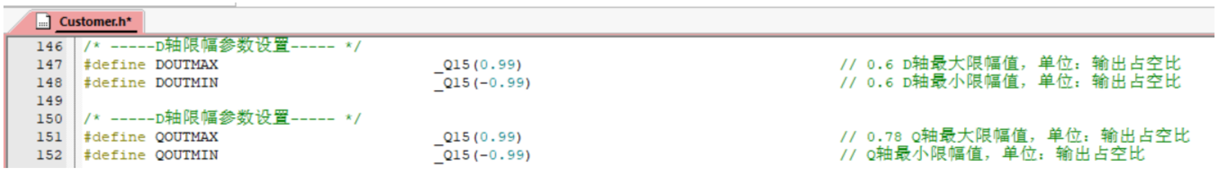
1) Startup-stage current loop PI affects motor startup;

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.

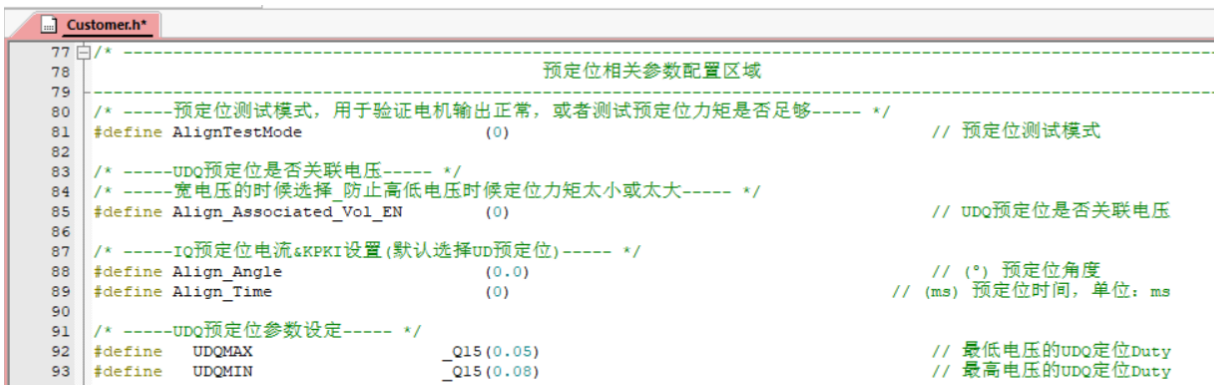

Customer.h*

```
/* -----D轴限幅参数设置----- */
#define DOUTMAX                     _Q15(0.99)                    // 0.6 D轴最大限幅值，单位：输出占空比
#define DOUTMIN                     _Q15(-0.99)                   // 0.6 D轴最小限幅值，单位：输出占空比

/* -----D轴限幅参数设置----- */
#define QOUTMAX                     _Q15(0.99)                    // 0.78 Q轴最大限幅值，单位：输出占空比
#define QOUTMIN                     _Q15(-0.99)                   // Q轴最小限幅值，单位：输出占空比
```

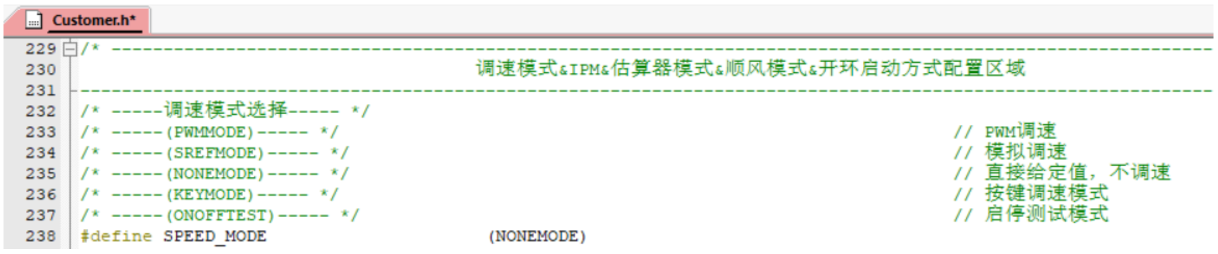
Notes:

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 Detection


Customer.h*

```
/* ---------------------------------------------------------------------------------------------------
                              预定位相关参数配置区域
-----------------------------------------------------------------------------------------------------
/* -----预定位测试模式，用于验证电机输出正常，或者测试预定位力矩是否足够----- */
#define AlignTestMode                (0)                          // 预定位测试模式

/* -----UDQ预定位是否关联电压----- */
/* -----宽电压的时候选择_防止高低电压时候定位力矩太小或太大----- */
#define Align_Associated_Vol_EN      (0)                          // UDQ预定位是否关联电压

/* -----IQ预定位电流&KPKI设置(默认选择UD预定位)----- */
#define Align_Angle                  (0.0)                        // (°) 预定位角度
#define Align_Time                   (0)                        // (ms) 预定位时间，单位：ms

/* -----UDQ预定位参数设定----- */
#define    UDQMAX                   _Q15(0.05)                    // 最低电压的UDQ定位Duty
#define    UDQMIN                   _Q15(0.08)                    // 最高电压的UDQ定位Duty
```

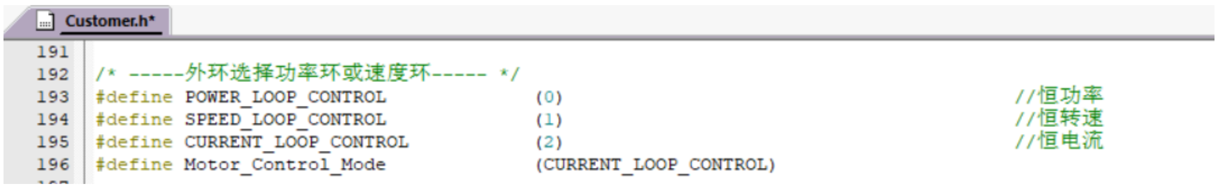

Customer.h*

```
/* ---------------------------------------------------------------------------------------------------
                   调速模式&IPM&估算器模式&顺风模式&开环启动方式配置区域
-----------------------------------------------------------------------------------------------------
/* -----调速模式选择----- */
/* -----(PWMMODE)----- */                                         // PWM调速
/* -----(SREFMODE)----- */                                        // 模拟调速
/* -----(NONEMODE)----- */                                        // 直接给定值，不调速
/* -----(KEYMODE)----- */                                         // 按键调速模式
/* -----(ONOFFTEST)----- */                                       // 启停测试模式
#define SPEED_MODE                    (NONEMODE)
```

Set AlignTestMode as 1 and select speed control mode as NONEMODE to start the motor. Then it goes to motor alignment state and the three phases UVW output fixed PWM waveforms, which indicates hardware drive circuit operates normally. Otherwise, if there is no output, users need check hardware problems.

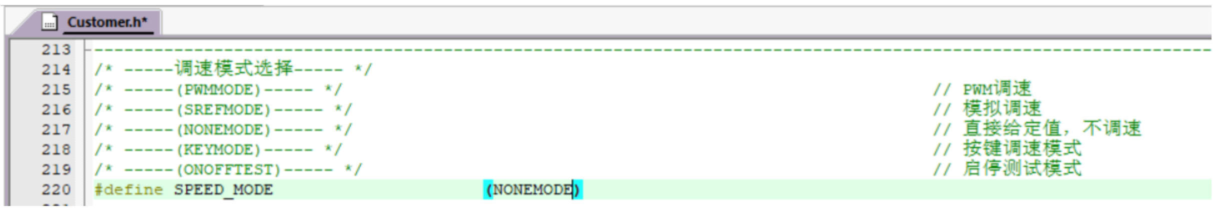
4.7 Current Loop Debugging

1. Select loop type as current loop;

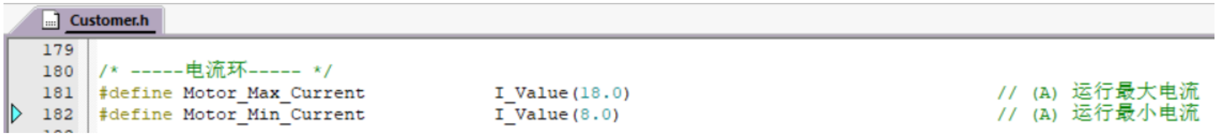


```
/* -----外环选择功率环或速度环----- */
#define POWER_LOOP_CONTROL              (0)                                         //恒功率
#define SPEED_LOOP_CONTROL              (1)                                         //恒转速
#define CURRENT_LOOP_CONTROL            (2)                                         //恒电流
#define Motor_Control_Mode              (CURRENT_LOOP_CONTROL)
```

2. Select speed regulation method as given value, and adjust the given value Motor_Min_Current to control the current through current loop (note that the given value is phase current value; since given value is chosen as speed regulation mode, the program recognizes Motor_Min_Current only, but not Motor_Max_Current);



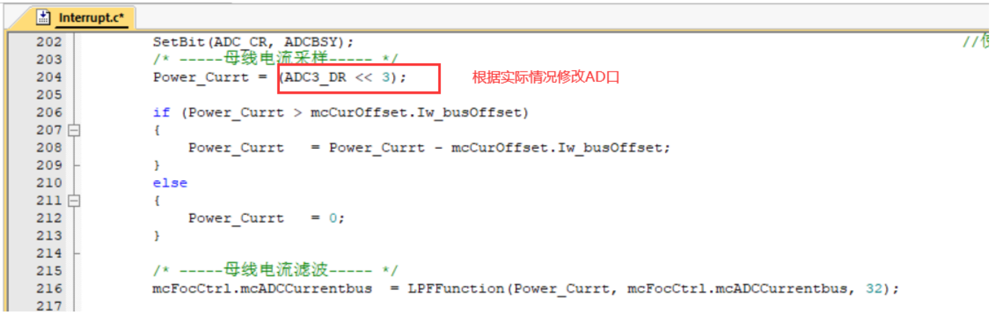
```
/*---------------------------------------------------------------------------------------------------------
/* -----调速模式选择----- */
/* -----(PWMMODE)----- */                                                    // PWM调速
/* -----(SREFMODE)----- */                                                   // 模拟调速
/* -----(NONEMODE)----- */                                                   // 直接给定值，不调速
/* -----(KEYMODE)----- */                                                    // 按键调速模式
/* -----(ONOFFTEST)----- */                                                  // 启停测试模式
#define SPEED_MODE                      (NONEMODE)
```



```
/* -----电流环----- */
#define Motor_Max_Current               I_Value(18.0)                              // (A)  运行最大电流
#define Motor_Min_Current               I_Value(8.0)                               // (A)  运行最小电流
```

3. Program the chip then power on and start the motor. If motor startup fails (usually, it starts up normally), adjust the startup settings below:
   - ■ Startup current: IQ_Start_CURRENT. A motor can't start up when the current is insufficient. Increase the current progressively with no sudden current rise.
   - ■ The current from startup stage to loop switching: IQ_RUN_CURRENT. Let the current a bit less than the startup current.
   - ■ Parameters affecting startup frequency such as ATO, Omega.
4. Power on and start up the motor, then increase the given value of current loop to reach the customer's target power;
5. Determine the maximum power and speed in current loop mode;
6. Record the calculated power value mcFocCtrl.Powerlpf (this value is the maximum given reference power value in power loop mode) at the maximum power, and the set phase current Motor_Min_Current at the time (this value can be applied as the reference value of outer loop SOUTMAX).

Notes: The AD port for BUS current sampling should be set according to actual hardware circuit. Where to set it is shown in the figure below.



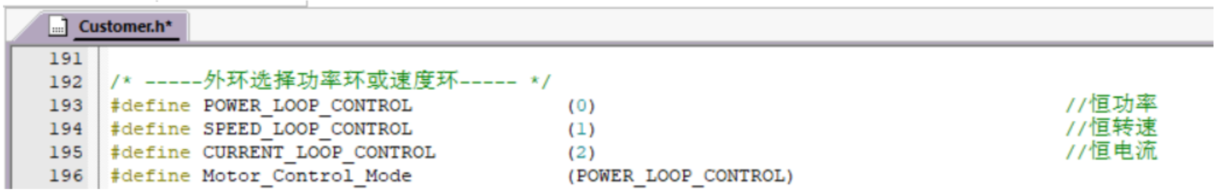
```
        SetBit(ADC_CR, ADCBSY);
        /* -----母线电流采样----- */
        Power_Currt = (ADC3_DR << 3);

        if (Power_Currt > mcCurOffset.Iw_busOffset)
        {
            Power_Currt    = Power_Currt - mcCurOffset.Iw_busOffset;
        }
        else
        {
            Power_Currt    = 0;
        }

        /* -----母线电流滤波----- */
        mcFocCtrl.mcADCCurrentbus  = LPFFunction(Power_Currt, mcFocCtrl.mcADCCurrentbus, 32);
```

Common issues and solutions:

1) The maximum power value required by a customer cannot be reached by the increase of given current.
   Solution: under the sinusoidal waveform condition, observe whether FOC__UQ is saturated. If so and if FOC__UD is quite a large value, adjust compensation angle FOC_THECOMP (try both positive value and negative value) to see whether it can meet customer requirements.
2) The recorded mcFocCtrl.Powerlpf is too large (e.g., exceeding 32767) or too small (e.g., just a few hundred at the maximum);
   Solution: since the value is obtained by bit-shifting the product of current value and voltage value, users can modify the value by modifying the bit shifting value to make it within a reasonable range (in general,10,000 ~ 20,000 is a reasonable range).
3) Overcurrent protection is triggered when the motor is operating;
   Solution: check whether phase current waveform has problem; check whether the set value is too small, which makes overcurrent protection be triggered accordingly. If no issue is found, check hardware wiring problems and so on.
4) Phase current waveforms encounter jitter.
   Solution: Adjust current loop PI value (i.e., DQKP, DQKI). The current loop PI and current sampling can affect current waveform stability obviously.

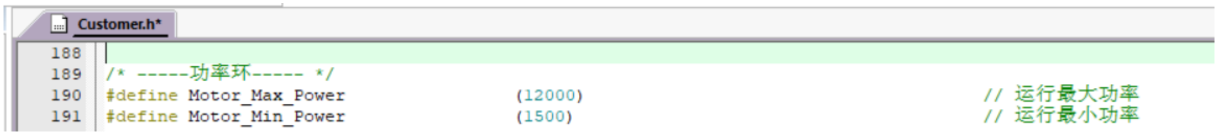
4.8 Power Loop Debugging

1. In general, vacuum cleaner motors apply power loop. Therefore, select loop type as power loop;

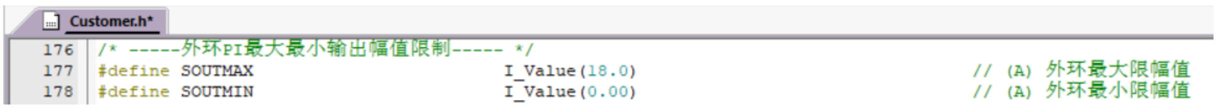

Customer.h*

```
/* -----外环选择功率环或速度环----- */
#define POWER_LOOP_CONTROL              (0)                         //恒功率
#define SPEED_LOOP_CONTROL              (1)                         //恒转速
#define CURRENT_LOOP_CONTROL            (2)                         //恒电流
#define Motor_Control_Mode              (POWER_LOOP_CONTROL)
```

2. Set the maximum value of the power curve and the value of SOUTMAX, which are recorded in the last step in Section 4.7. As for the value of SOUTMAX, it may need to increase a little based on the recorded value, to prevent current rising further while voltage drops. It needs to make enough margin to ascend. Motor_Min_Power is the minimum power value of the curve, which is set according to real customer requirements;

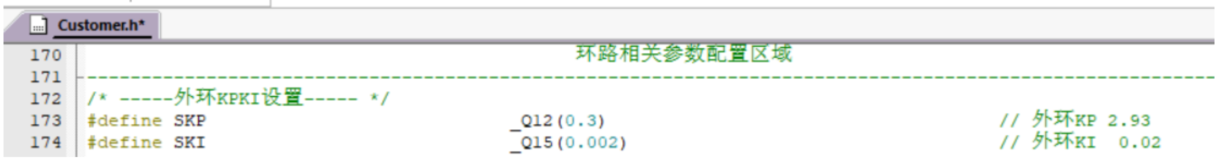

Customer.h*

```
/* -----功率环----- */
#define Motor_Max_Power                 (12000)                     // 运行最大功率
#define Motor_Min_Power                 (1500)                      // 运行最小功率
```

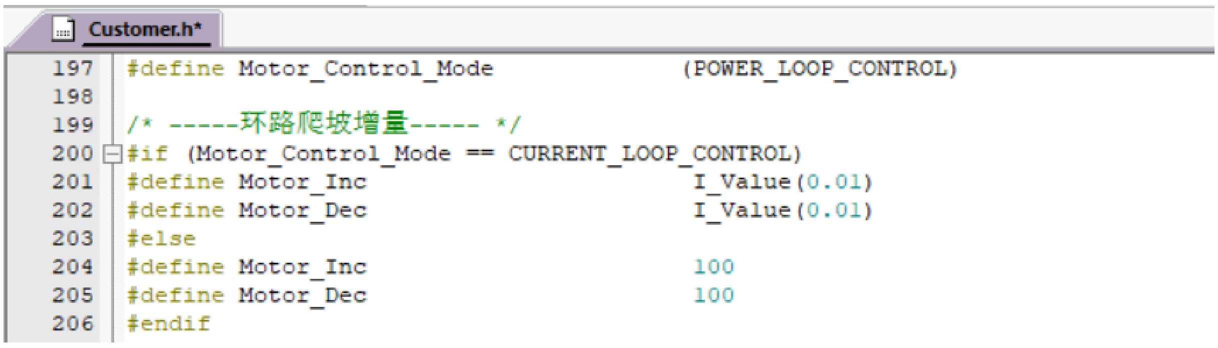

Customer.h*

```
/* -----外环PI最大最小输出幅值限制----- */
#define SOUTMAX                         I_Value(18.0)               // (A) 外环最大限幅值
#define SOUTMIN                         I_Value(0.00)               // (A) 外环最小限幅值
```

3. Adjust power loop PI (SKP and SKI) and the incremental ramping value of power loop, in order to ensure power loop stability, quick startup response, and avoid overshoot.


Customer.h*

```
                                         环路相关参数配置区域
------------------------------------------------------------------------------------------------
/* -----外环KPKI设置----- */
#define SKP                             _Q12(0.3)                          // 外环KP 2.93
#define SKI                             _Q15(0.002)                        // 外环KI  0.02
```


Customer.h*

```
#define Motor_Control_Mode              (POWER_LOOP_CONTROL)

/* -----环路爬坡增量----- */
#if (Motor_Control_Mode == CURRENT_LOOP_CONTROL)
#define Motor_Inc                        I_Value(0.01)
#define Motor_Dec                        I_Value(0.01)
#else
#define Motor_Inc                        100
#define Motor_Dec                        100
#endif
```

Notes:

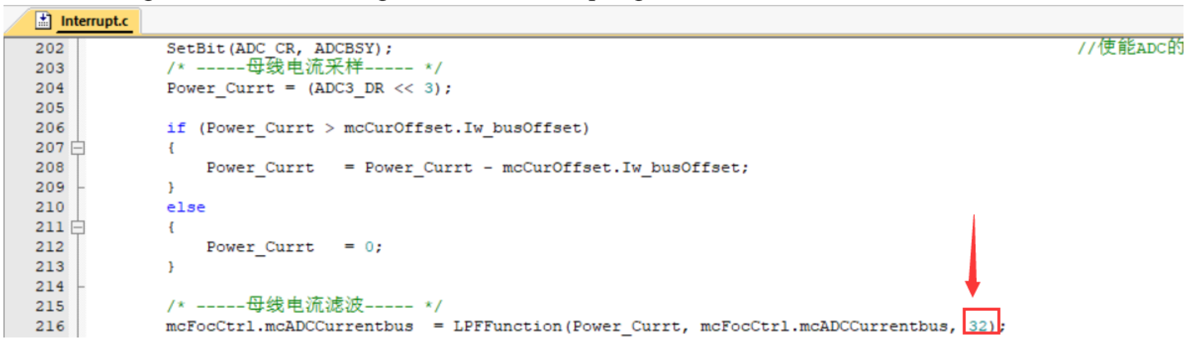
In general, the calculated value of mcFocCtrl.Powerlpf here is proportional to the real power. Thus, it is needed to measure a group of mcFocCtrl.Powerlpf and the real powers to calculate the power coefficient K.

E.g., when a motor operates at 100W, the mcFocCtrl.Powerlpf is 5000, then the power coefficient K = mcFocCtrl.Powerlpf/real power value = 50; If a target value is set as 500*50, it is equivalent to 500W.

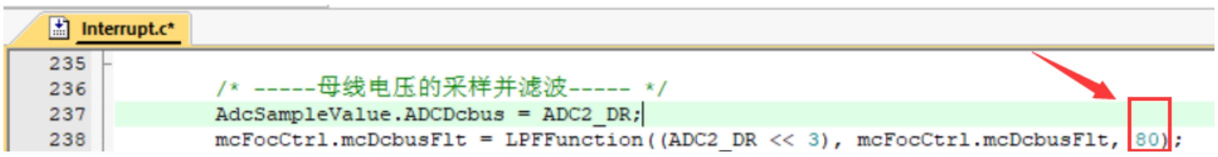
Common issues when power loop is selected:

1) The phase current of current loop is originally stable; however, it jitters after power loop is selected.

   Solution: In general, it's associated with power loop involved parameters. In this case, users can adjust power loop PI or the filter coefficient of power loop feedback value mcFocCtrl.Powerlpf. Since mcFocCtrl.Powerlpf is the product of current and voltage, to adjust the filter coefficient of mcFocCtrl. Powerlpf is to adjust the filtering coefficient of voltage and current sampling.


Interrupt.c

```
        SetBit(ADC_CR, ADCBSY);                                                  //使能ADC的
        /* -----母线电流采样----- */
        Power_Currt = (ADC3_DR << 3);

        if (Power_Currt > mcCurOffset.Iw_busOffset)
        {
            Power_Currt    = Power_Currt - mcCurOffset.Iw_busOffset;
        }
        else
        {
            Power_Currt    = 0;
        }

        /* -----母线电流滤波----- */
        mcFocCtrl.mcADCCurrentbus  = LPFFunction(Power_Currt, mcFocCtrl.mcADCCurrentbus, 32);
```

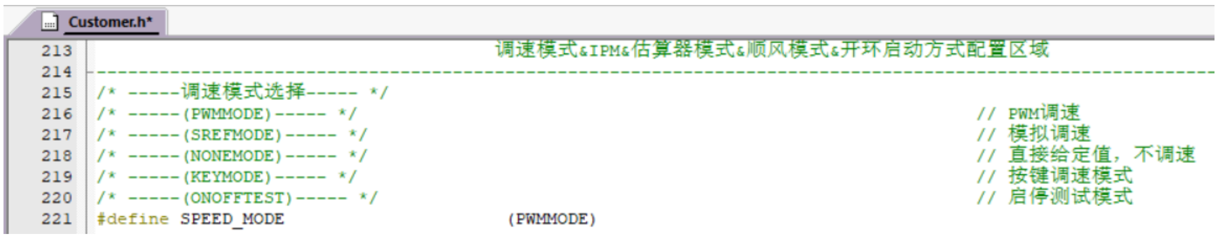

Interrupt.c*

```

        /* -----母线电压的采样并滤波----- */
        AdcSampleValue.ADCDcbus = ADC2_DR;
        mcFocCtrl.mcDcbusFlt = LPFFunction((ADC2_DR << 3), mcFocCtrl.mcDcbusFlt, 80);
```

4.9 PWM Function Debugging

In general, vacuum cleaner motors apply PWM based speed regulation. PWM debugging steps are as follows:

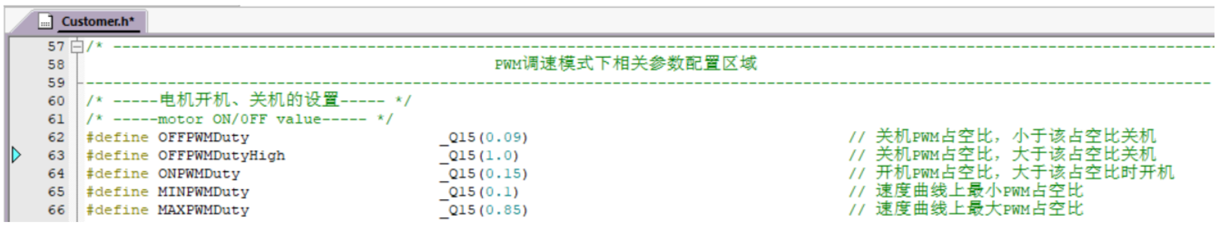
1) Select speed regulation mode as PWM speed regulation. Adjust minimum and maximum power values, the corresponding PWM on-off duty cycle, and minimum and maximum duty cycles, according to a customer's curve;



Customer.h*

```
                                        调速模式&IPM&估算器模式&顺风模式&开环启动方式配置区域
-----------------------------------------------------------------------------------------------------------
/* -----调速模式选择----- */
/* -----(PWMMODE)----- */                                                        // PWM调速
/* -----(SREFMODE)----- */                                                       // 模拟调速
/* -----(NONEMODE)----- */                                                       // 直接给定值，不调速
/* -----(KEYMODE)----- */                                                        // 按键调速模式
/* -----(ONOFFTEST)----- */                                                      // 启停测试模式
#define SPEED_MODE                          (PWMMODE)
```

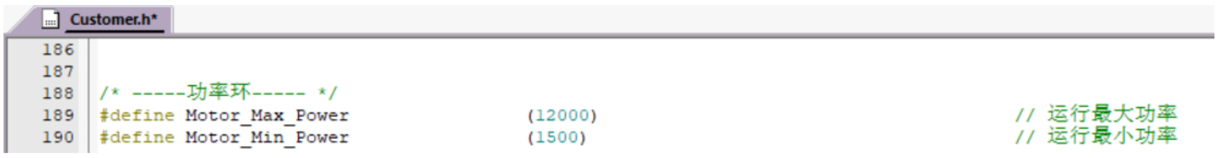
2) Adjust motor-on and motor-off duty cycles, maximum and minimum duty cycles according to the customer's PWM curve;



Customer.h*

```
/* ---------------------------------------------------------------------------------------------------------
                                        PWM调速模式下相关参数配置区域
-----------------------------------------------------------------------------------------------------------
/* -----电机开机、关机的设置----- */
/* -----motor ON/OFF value----- */
#define OFFPWMDuty                      _Q15(0.09)                      // 关机PWM占空比，小于该占空比关机
#define OFFPWMDutyHigh                  _Q15(1.0)                       // 关机PWM占空比，大于该占空比关机
#define ONPWMDuty                       _Q15(0.15)                      // 开机PWM占空比，大于该占空比时开机
#define MINPWMDuty                      _Q15(0.1)                       // 速度曲线上最小PWM占空比
#define MAXPWMDuty                      _Q15(0.85)                      // 速度曲线上最大PWM占空比
```

3) Adjust maximum and minimum power according to the ones provided by the customer;

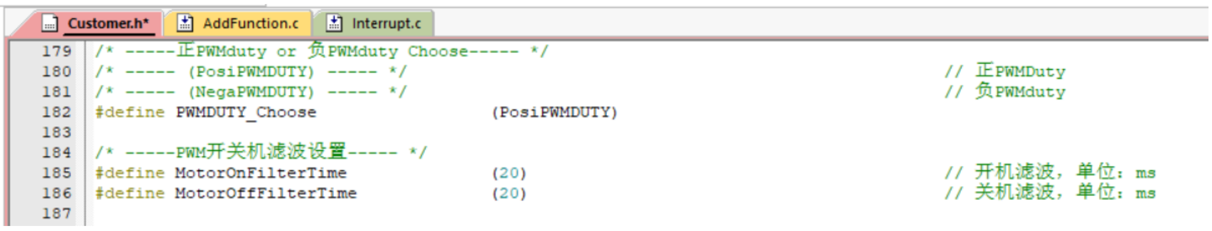


Customer.h*

```

/* -----功率环----- */
#define Motor_Max_Power                 (12000)                         // 运行最大功率
#define Motor_Min_Power                 (1500)                          // 运行最小功率
```

The lowest point of the curve is (MINPWMDuty, Motor_Min_Power), and the highest is (MAXPWMDuty, Motor_Max_Power).

4) Confirm whether the customer applies positive PWM speed regulation or negative PWM speed regulation. For positive PWM speed regulation, speed increases when PWM duty cycle increases; for negative PWM speed regulation, speed decreases when PWM duty cycle increases.



Customer.h* | AddFunction.c | Interrupt.c

```
/* -----正PWMduty or 负PWMduty Choose----- */
/* ----- (PosiPWMDUTY) ----- */                                                  // 正PWMDuty
/* ----- (NegaPWMDUTY) ----- */                                                  // 负PWMduty
#define PWMDUTY_Choose                  (PosiPWMDUTY)

/* -----PWM开关机滤波设置----- */
#define MotorOnFilterTime               (20)                                     // 开机滤波，单位：ms
#define MotorOffFilterTime              (20)                                     // 关机滤波，单位：ms

```

Notes:

1) Select an appropriate timer frequency division according to PWM frequency during Timer3 initialization;

2) Leave a hysteresis margin between motor-on and motor-off duty cycles, such as a motor-on duty cycle of 10% and a motor-off duty cycle of 8% with a hysteresis margin of 2%. The same motor-on and motor-off duty cycle would cause uncertainty in motor on/off operation, i.e., sometimes it operates as motor on and sometimes as off;

3) When getting incorrect PWM duty cycle, users should check if the PWM signal entering the chip pins has been distorted, which may be caused by filtering capacitance being too large;

4) When PWM signal interference is encountered, try turning on the filtering function of capture TIM port, or adjust the position of PWM hardware filter capacitor to place it as close as possible to the chip pin.

4.10 Other Functions

1. If other functions such as FG output are required, users can add them accordingly;
2. Add protection functions: Enable phase loss protection, motor locked protection, over-temperature protection, overspeed protection and so on according to customer needs. All other protections required but not contained in the program need to be implemented by customers. Refer to Section 5.2 for protection function details.

4.11 Reliability Test

4.11.1 Reliability of Function

After adding all functions, test the functions in the customer requirement list and make sure no abnormality occurs.

4.11.2 Reliability of Protection

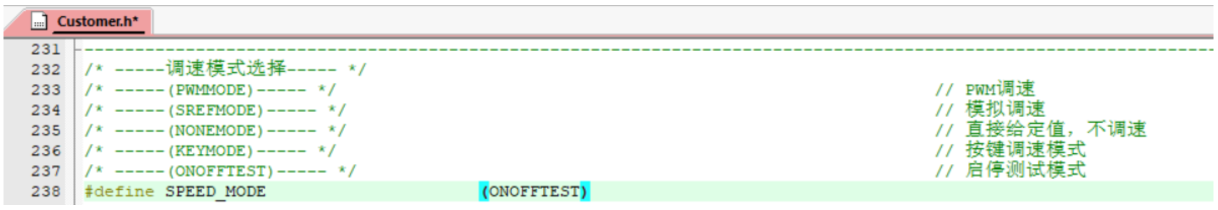
After adding protection functions, verify each protection can be triggered normally and no protection is triggered falsely while motor is in normal operation. E.g., upon improper motor locked protection settings, it may cause the motor to report locked protection falsely during normal operation; or it fails to report motor locked protection upon actual motor locked occurrence.

4.11.3 Stability of Startup

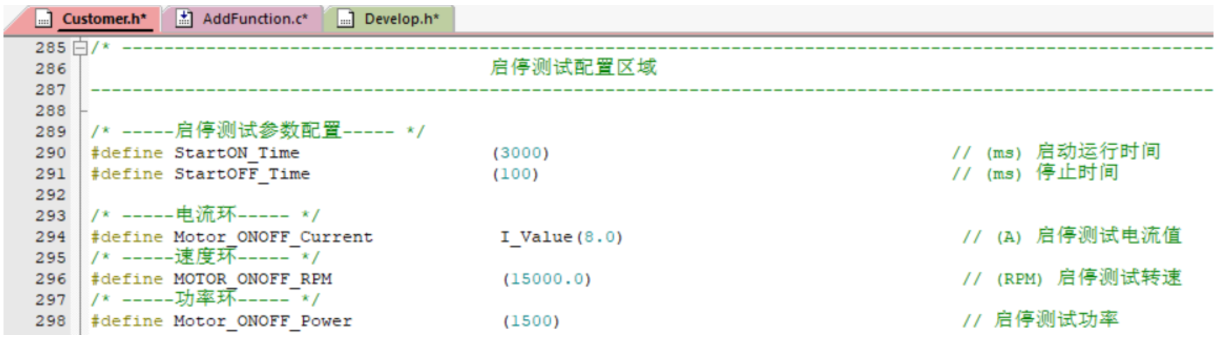
When finishing functional debugging, users can carry on startup reliability test. Manual test is performed first. Then perform aging test after manual test is passed.

Aging test steps:

1. Turn ONOFFTEST on;
2. Configure start on time StartON_Time and stop time StartOFF_Time according to real situations;
3. Users can adjust the power of motor on and motor off by tuning Motor_ONOFF_Power;
4. Block the motor using a tool then power up it. Check if a motor-locked protection is triggered normally, and no motor restart after the protection. It is to verify no restart occurs upon protection being triggered during motor startup or stop;
5. Power on again and perform aging test. Finally, judge startup failure by checking whether the motor remains stopped. Upon startup failure, the motor remains stopped with no more restart. In general, more than 3000 (the more the better if time allows) consecutively successful startup operations can secure startup reliability.


Customer.h*

```
----------------------------------------------------------------------------------------------------
/* -----调速模式选择----- */
/* -----(PWMMODE)----- */                                                         // PWM调速
/* -----(SREFMODE)----- */                                                        // 模拟调速
/* -----(NONEMODE)----- */                                                        // 直接给定值，不调速
/* -----(KEYMODE)----- */                                                         // 按键调速模式
/* -----(ONOFFTEST)----- */                                                       // 启停测试模式
#define SPEED_MODE                  (ONOFFTEST)
```



```
/* ----------------------------------------------------------------------------------------------------
                                    启停测试配置区域
   ----------------------------------------------------------------------------------------------------

/* -----启停测试参数配置----- */
#define StartON_Time                (3000)                       // (ms) 启动运行时间
#define StartOFF_Time               (100)                        // (ms) 停止时间

/* -----电流环----- */
#define Motor_ONOFF_Current          I_Value(8.0)                 // (A) 启停测试电流值
/* -----速度环----- */
#define MOTOR_ONOFF_RPM              (15000.0)                    // (RPM) 启停测试转速
/* -----功率环----- */
#define Motor_ONOFF_Power            (1500)                       // 启停测试功率
```

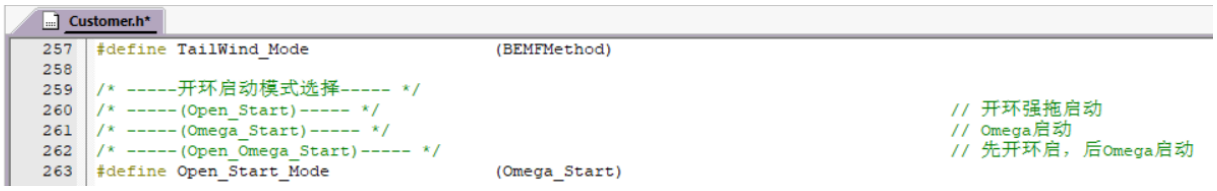
5 Function Introduction

When users get the original program, configure motor parameters and hardware parameters, then send start signal to a target motor, it usually can start normally. If encountering abnormal startup, users should check and settle down hardware problems first, then adjust startup settings.

5.1 Startup Debugging

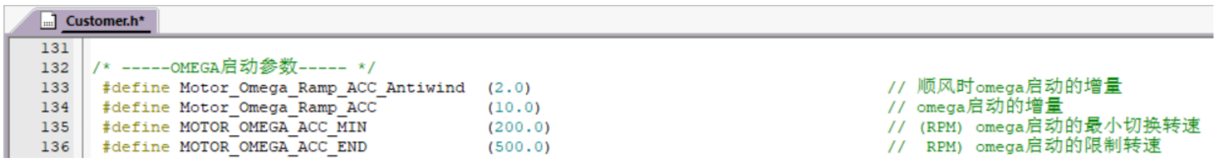
5.1.1 Omega Startup

Omega startup should be selected for vacuum cleaner motors, which is the default selection in the program.



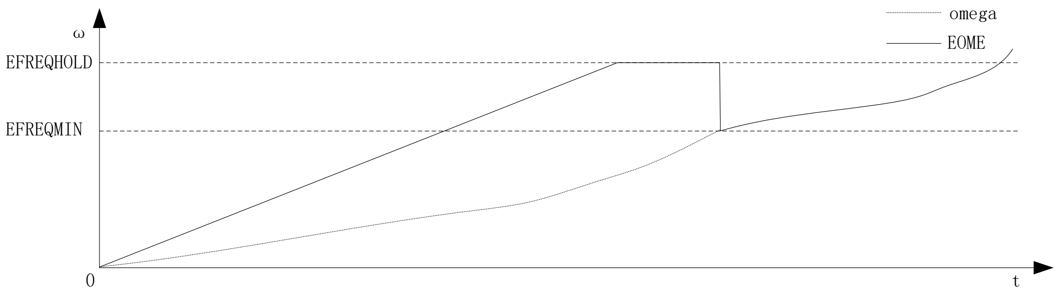
```
#define TailWind_Mode               (BEMFMethod)

/* -----开环启动模式选择----- */
/* -----(Open_Start)----- */                                 // 开环强拖启动
/* -----(Omega_Start)----- */                                // Omega启动
/* -----(Open_Omega_Start)----- */                           // 先开环启，后Omega启动
#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.



```
/* -----OMEGA启动参数----- */
 #define Motor_Omega_Ramp_ACC_Antiwind  (2.0)                 // 顺风时omega启动的增量
 #define Motor_Omega_Ramp_ACC           (10.0)                // omega启动的增量
 #define MOTOR_OMEGA_ACC_MIN            (200.0)               // (RPM) omega启动的最小切换转速
 #define MOTOR_OMEGA_ACC_END            (500.0)               //  RPM) omega启动的限制转速
```

Startup procedure is shown in the figure below.


omega
EOME
ω
EFREQHOLD
EFREQMIN
0
t

5.1.2 Common Issues & Solutions in Starting

Common Issues	Solutions
The motor rotates at an instant then stops with input 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.

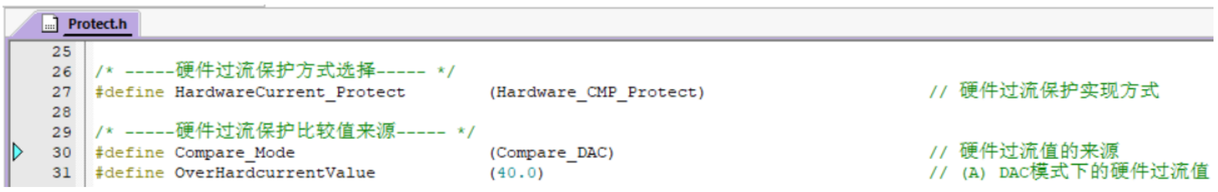
5.2 Introduction to Protection Functions

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;

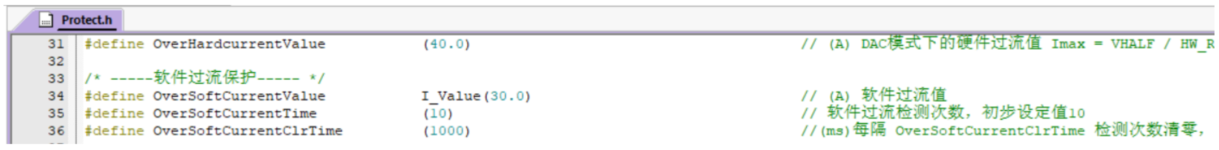
Hardware overcurrent protection is fulfilled through comparator 3 in the chip. The detection method is: The BUS current flows through the sampling resistor, and a voltage is formed on the sampling resistor; the voltage is amplified by the operational amplifier and sent to the positive input of the comparator. 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.



```
/* -----硬件过流保护方式选择----- */
#define HardwareCurrent_Protect        (Hardware_CMP_Protect)            // 硬件过流保护实现方式

/* -----硬件过流保护比较值来源----- */
#define Compare_Mode                   (Compare_DAC)                     // 硬件过流值的来源
#define OverHardcurrentValue           (40.0)                            // (A) DAC模式下的硬件过流值
```

2. Software overcurrent protection

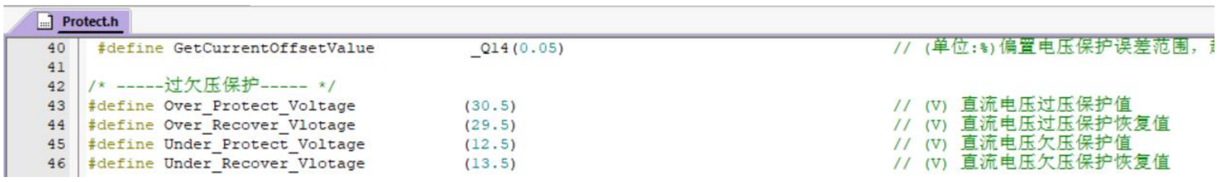
The program obtains the maximum current value of three phases. When the maximum current value exceeds a preset software overcurrent protection value OverSoftCurrentValue, it counts once. If the count value exceeds OverSoftCurrentTime within the OverSoftCurrentClrTime time, protection is triggered.



```
#define OverHardcurrentValue           (40.0)                  // (A) DAC模式下的硬件过流值 Imax = VHALF / HW_R

/* -----软件过流保护----- */
#define OverSoftCurrentValue           I_Value(30.0)           // (A) 软件过流值
#define OverSoftCurrentTime            (10)                    // 软件过流检测次数，初步设定值10
#define OverSoftCurrentClrTime         (1000)                  //(ms)每隔 OverSoftCurrentClrTime 检测次数清零，
```

5.2.2 Voltage Protection

The program detects the voltage through the AD2 port and reports overvoltage protection 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.

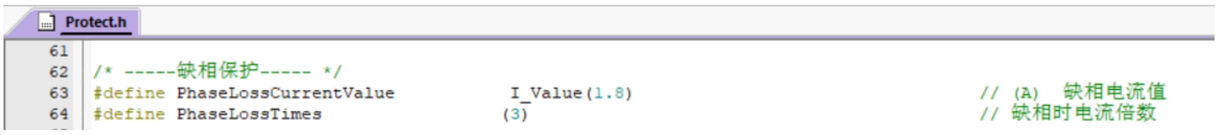


```
 #define GetCurrentOffsetValue          _Q14(0.05)                 // (单位:%)偏置电压保护误差范围，

/* -----过欠压保护----- */
#define Over_Protect_Voltage           (30.5)                     // (V) 直流电压过压保护值
#define Over_Recover_Vlotage           (29.5)                     // (V) 直流电压过压保护恢复值
#define Under_Protect_Voltage          (12.5)                     // (V) 直流电压欠压保护值
#define Under_Recover_Vlotage          (13.5)                     // (V) 直流电压欠压保护恢复值
```

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.


Protect.h
```
/* -----缺相保护----- */
#define PhaseLossCurrentValue           I_Value(1.8)                                  // (A)  缺相电流值
#define PhaseLossTimes                  (3)                                           // 缺相时电流倍数
```

Notes:

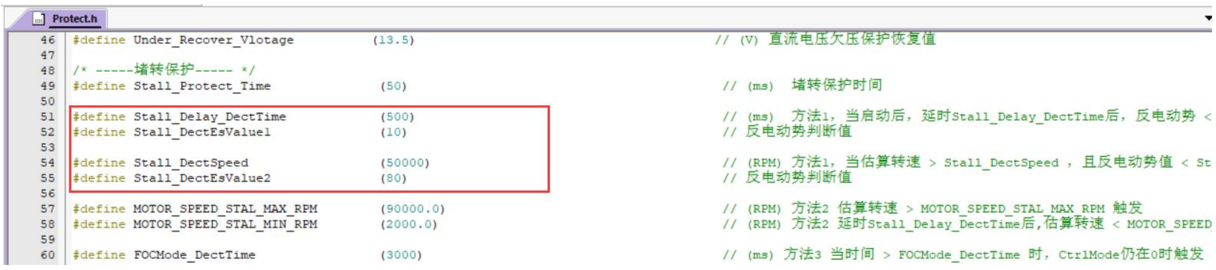
In some cases, when a phase is missing, the signal of the missing phase will have burrs, which may cause the maximum current value collected to be about the same as those of the other two phases, which may not be detected by the above method. The phase loss detection method discussed above may fail in 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

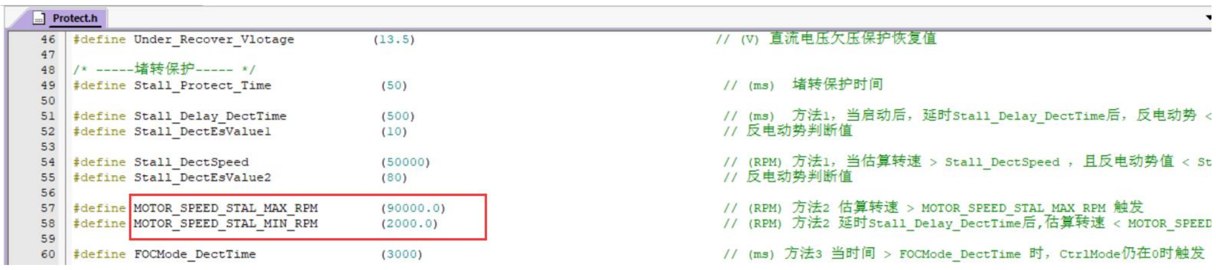
There are 3 detection methods for motor locked protection:

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. Thus, it can be used to detect motor being locked.

The specific method is: Start the motor with a delay time Stall_Delay_DectTime; judge whether the FOC_ESQU is still less than the set value Stall_DectEsValue1, or whether the value of FOC_ESQU is less than the set value Stall_DectEsValue2 when the estimated speed is higher than the set value Stall_DectSpeed. If so, the motor is judged as locked.

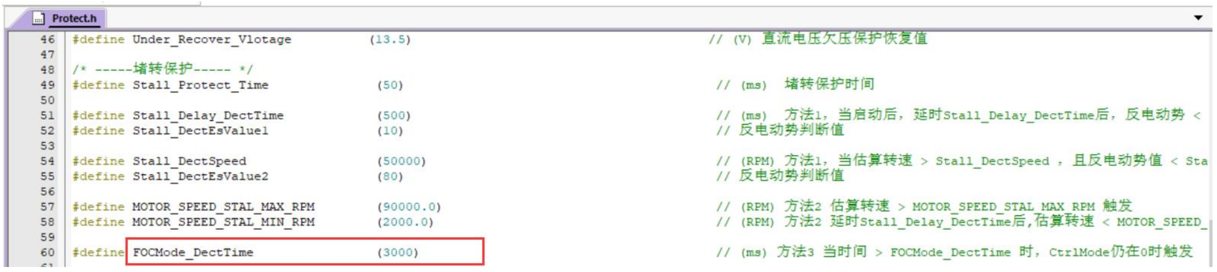

Protect.h
```
#define Under_Recover_Vlotage          (13.5)                          // (V) 直流电压欠压保护恢复值

/* -----堵转保护----- */
#define Stall_Protect_Time              (50)                            // (ms)  堵转保护时间

#define Stall_Delay_DectTime            (500)                           // (ms)  方法1, 当启动后, 延时Stall_Delay_DectTime后, 反电动势 <
#define Stall_DectEsValue1              (10)                            // 反电动势判断值

#define Stall_DectSpeed                 (50000)                         // (RPM) 方法1, 当估算转速 > Stall_DectSpeed , 且反电动势值 < St
#define Stall_DectEsValue2              (80)                            // 反电动势判断值

#define MOTOR_SPEED_STAL_MAX_RPM        (90000.0)                       // (RPM) 方法2 估算转速 > MOTOR_SPEED_STAL_MAX_RPM 触发
#define MOTOR_SPEED_STAL_MIN_RPM        (2000.0)                        // (RPM) 方法2 延时Stall_Delay_DectTime后,估算转速 < MOTOR_SPEED

#define FOCMode_DectTime                (3000)                          // (ms) 方法3 当时间 > FOCMode_DectTime 时, CtrlMode仍在0时触发
```

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.


Protect.h
```
#define Under_Recover_Vlotage          (13.5)                          // (V) 直流电压欠压保护恢复值

/* -----堵转保护----- */
#define Stall_Protect_Time              (50)                            // (ms)  堵转保护时间

#define Stall_Delay_DectTime            (500)                           // (ms)  方法1, 当启动后, 延时Stall_Delay_DectTime后, 反电动势 <
#define Stall_DectEsValue1              (10)                            // 反电动势判断值

#define Stall_DectSpeed                 (50000)                         // (RPM) 方法1, 当估算转速 > Stall_DectSpeed , 且反电动势值 < St
#define Stall_DectEsValue2              (80)                            // 反电动势判断值

#define MOTOR_SPEED_STAL_MAX_RPM        (90000.0)                       // (RPM) 方法2 估算转速 > MOTOR_SPEED_STAL_MAX_RPM 触发
#define MOTOR_SPEED_STAL_MIN_RPM        (2000.0)                        // (RPM) 方法2 延时Stall_Delay_DectTime后,估算转速 < MOTOR_SPEED

#define FOCMode_DectTime                (3000)                          // (ms) 方法3 当时间 > FOCMode_DectTime 时, CtrlMode仍在0时触发
```

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.



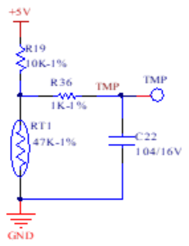
Protect.h

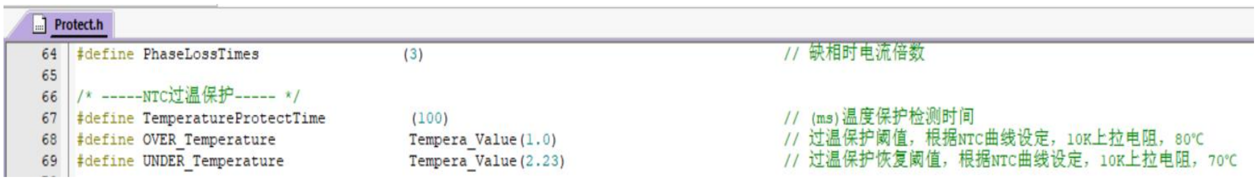
```
#define Under_Recover_Vlotage          (13.5)                                   // (V) 直流电压欠压保护恢复值

/* -----堵转保护----- */
#define Stall_Protect_Time              (50)                                    // (ms)  堵转保护时间

#define Stall_Delay_DectTime            (500)                                   // (ms)  方法1, 当启动后, 延时Stall_Delay_DectTime后, 反电动势 <
#define Stall_DectEsValue1              (10)                                    // 反电动势判断值

#define Stall_DectSpeed                 (50000)                                 // (RPM) 方法1, 当估算转速 > Stall_DectSpeed , 且反电动势值 < Sta
#define Stall_DectEsValue2              (80)                                    // 反电动势判断值

#define MOTOR_SPEED_STAL_MAX_RPM        (90000.0)                               // (RPM) 方法2 估算转速 > MOTOR_SPEED_STAL_MAX_RPM 触发
#define MOTOR_SPEED_STAL_MIN_RPM        (2000.0)                                // (RPM) 方法2 延时Stall_Delay_DectTime后,估算转速 < MOTOR_SPEED_

#define FOCMode_DectTime                (3000)                                  // (ms) 方法3 当时间 > FOCMode_DectTime 时, CtrlMode仍在0时触发
```

5.2.5 Over Temperature Protection

A typical circuit for over-temperature protection is shown in the figure below. The voltage divider usually adopts an NTC resistor, with the resistance value falling as temperature rising and each temperature value corresponding to a resistance value. TD is connected to an AD port of the chip. The program detects the voltage of this AD port. When the voltage is lower than the voltage at the set temperature, it indicates the temperature of the NTC resistor exceeds the set value. Protection is triggered then.

Temperature detection





Protect.h

```
#define PhaseLossTimes               (3)                              // 缺相时电流倍数

/* -----NTC过温保护----- */
#define TemperatureProtectTime        (100)                           // (ms)温度保护检测时间
#define OVER_Temperature             Tempera_Value(1.0)               // 过温保护阈值, 根据NTC曲线设定, 10K上拉电阻, 80℃
#define UNDER_Temperature            Tempera_Value(2.23)              // 过温保护恢复阈值, 根据NTC曲线设定, 10K上拉电阻, 70℃
```

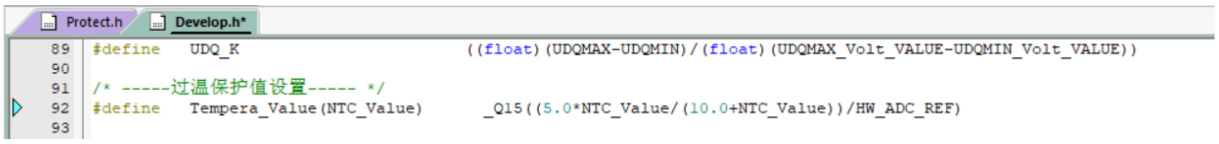
In the above diagram,

OVER_Temperature is the protection value set, and 1.0 represents the resistance value of the NTC resistor at 80°C is 1Ω;

UNDER_Temperature is recovery value, and 2.23 represents the resistance value of the NTC resistor at 70°C is 2.23Ω.

Notes:

If the pull-up resistor is not 10k and the pull-up voltage is not 5V, the definition formula should be changed accordingly.


Protect.h Develop.h*

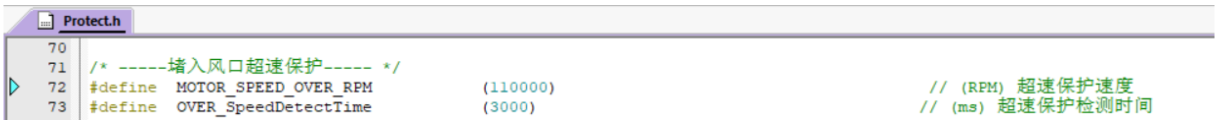
```
#define   UDQ_K                          ((float)(UDQMAX-UDQMIN)/(float)(UDQMAX_Volt_VALUE-UDQMIN_Volt_VALUE))

/* -----过温保护值设置----- */
#define   Tempera_Value(NTC_Value)        _Q15((5.0*NTC_Value/(10.0+NTC_Value))/HW_ADC_REF)
```

In the above diagram,

5.0 represents the voltage before division, namely 5V, as shown in the above circuit diagram. The value should be set according to real circuit; 10.0 represents pull-up resistance, the value should be set according to real circuit.

5.2.6 Overspeed Protection

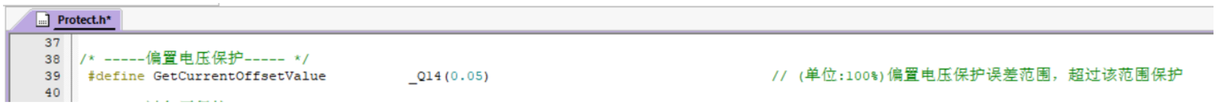
It detects motor speed. If motor speed continuously exceeds the set speed MOTOR_SPEED_OVER_RPM within the period of OVER_SpeedDetectTime, protection is triggered.


Protect.h

```
/* -----堵入风口超速保护----- */
#define  MOTOR_SPEED_OVER_RPM          (110000)                          // (RPM) 超速保护速度
#define  OVER_SpeedDetectTime          (3000)                          // (ms) 超速保护检测时间
```

5.2.7 Bias Voltage Protection

Before a motor starts, the bias voltage is sampled first. When VHALF is connected, the sampled value of bias voltage is 2048 in theory, being about 16383 after shifting 3 bits to the left. When VHALF is unconnected, the value is 0 in theory; when the difference between the sampled value and the theorical value exceeds the percentage specified in GetCurrentOffsetValue either upwards or downwards, the bias voltage is recognized as abnormal. In the diagram below, 0.05 represents 5%.


Protect.h*

```
/* -----偏置电压保护----- */
 #define GetCurrentOffsetValue          _Q14(0.05)                    // (单位:100%)偏置电压保护误差范围，超过该范围保护
```

5.2.8 Other Protections

Users can add other protections per customer needs.

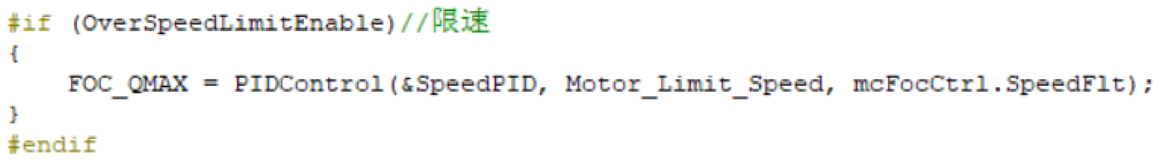
6 Other Common Function Debugging

6.1 Speed Limiting Function

When constant power control is applied, in case the air intaking port of a vacuum cleaner motor is blocked, the load declines and the motor could run at very high speed, which causes both bearing damage and motor damage due to poor heat dissipation. That's why motor speed limiting should to be conducted.

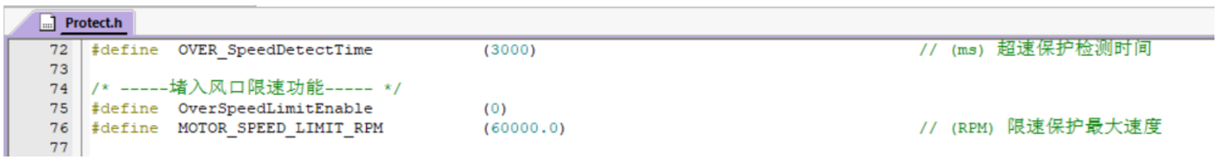
There are 3 speed limiting methods.

1. Limit the target value. When it is detected motor speed exceeds a protection threshold, the target value is limited in the ramping function to fulfill speed limiting. As this method is easy to cause oscillation, it is not detailed here.
2. Switch to different closed loops to fulfill speed limiting. When the air intaking port of a vacuum cleaner motor is blocked and overspeed is caused, it can be detected that the speed exceeds a limited value. The program switches to speed closed loop to fulfill speed limiting; When the block is removed, since the load recovers, the power at the current speed exceeds the target power value, then the program switches to power closed loop to fulfill speed limiting. This method needs to adjust speed loop PI, power loop PI and PI response period. Further, since the method is easy to cause oscillation upon loop switching, it is not detailed here.
3. Limit speed through dual-PI, i.e., hardware PI for power closed loop and software PI for speed limiting. Software PI outputs the speed limit FOC_QMAX. The code is as follows:



```
#if (OverSpeedLimitEnable)//限速
{
    FOC_QMAX = PIDControl(&SpeedPID, Motor_Limit_Speed, mcFocCtrl.SpeedFlt);
}
#endif
```

Speed limiting function is now available in the program. It can be used directly.



Protect.h

```
#define  OVER_SpeedDetectTime            (3000)                                     // (ms) 超速保护检测时间

/* -----堵入风口限速功能----- */
#define  OverSpeedLimitEnable            (0)
#define  MOTOR_SPEED_LIMIT_RPM           (60000.0)                                  // (RPM) 限速保护最大速度

```

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. Debug 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

版本号：第 1 位-原理 第 2 位-模块 第 3 和 4 位-细节

Version No.	Changes	Date	Owner	Reviewer
V1.0.00	Initial version	2022-06-24	Francis	
V1.0.01	Update description of hardware, debugging steps	2022-07-27	Francis	

9 Copyright Notice

Fortior Technology (Shenzhen) Co., Ltd.

Room203, 2/F, Building No.11, Keji Central Road2,

Software Park, High-Tech Industrial Park, Shenzhen, P.R. China 518057

Tel:0755-26867710

Fax: 0755-26867715

URL: http://www.fortiortech.com

峰岹科技（深圳）股份有限公司

ISO9001 质量管控文件

FU6861
吸尘器电机调试手册

文件版本管制

版本	修订者	主要修改内容	生效日期	审核/批准
V1.0.00	蔡坤旺	初稿	2022/06/24	
V1.0.01	蔡坤旺	修改硬件、调试步骤部分描述	2022/07/27	

会签单位	签核人员	签核时间
应用部		
品质部		