

MDSF40

Low Voltage Ceiling Fan

Application Note

Sensorless Based High-Speed PMSM Solution

Catalog

CATALOG..... 2

1. BASIC SPECIFICATION PARAMETER CONFIGURATION 3

 1.1 MOTOR PARAMETER CONFIGURATION..... 3

 1.2 MOC AND HARDWARE PARAMETER CONFIGURATION..... 4

 1.3 PROTECTION PARAMETER CONFIGURATION 6

2. LOW VOLTAGE CEILING FAN ADJUSTMENT PROCESS 10

 2.1 START PARAMETER ADJUSTMENT..... 10

 2.2 PERIPHERAL CONTROL FUNCTION..... 15

 2.3 FAIRWIND AND HEADWIN START-UP PARAMETER ADJUSTMENT..... 27

 2.4 PROTECTION PARAMETER ADJUSTMENT 29

 2.5 OTHER PARAMETER CONFIGURATION..... 30

3. PROGRAM STRUCTURE..... 33

 3.1 PROGRAM FLOW 33

4. REFERENCE CIRCUIT DESIGN 34

 4.1 POWER INPUT CIRCUIT 34

 4.2 CORE UNIT AND PERIPHERAL CIRCUITS 34

 4.3 THREE-PHASE FULL-BRIDGE INVERTER..... 35

 4.4 TWO-PHASE SAMPLING CIRCUIT 35

 4.5 BUS CURRENT SAMPLING CIRCUIT 35

 4.6 BEMF FEEDBACK CIRCUIT 36

 4.7 EXTERNAL PORT 36

 4.8 OTHER PERIPHERALS 36

1. Basic specification parameter configuration

1.1 Motor parameter configuration

■ Configuration file : MOTOR.h <Configuration Wizard>

■ Set motor parameters

Parameter	Value range/option	Description	Default value
Motor Pole	0~30(multiple of 2)	Number of motor poles	14
Motor SMO_G	0~32767	$G = Ts / Ls/2$	15000
Motor SMO_F	0~32767	$F = 1 - (G * Rs/2)$	23000
Motor SMO_Kslf	0~32767	Low Pass Filter Gain	8000
Motor SMO_Gain	0~32767	$=Kslide1/MaxSmcError$	32767
MaxSmcError	0~32767	Judgment value of SMO current tolerance	32767

Suggestions for adjusting direction :

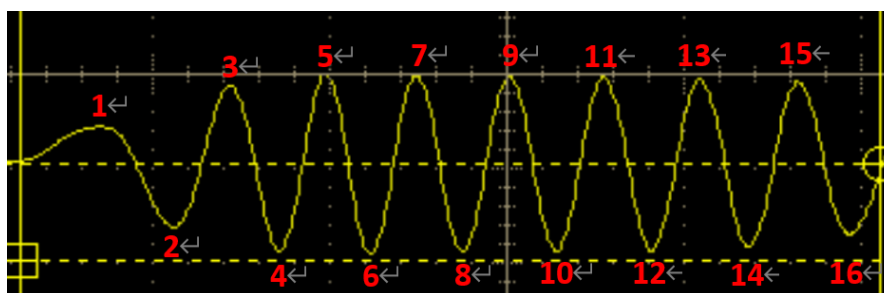
Measure the Motor poles :

When the exact number of poles of the motor cannot be known, the number of poles can be obtained by measuring the back EMF, methods as below:

Step1. Oscilloscope carbon rod randomly selects between any two phases of the motor (ex: U to V or V to W...), the motor does not need to be connected to the control board

Step2. Make the motor rotate one circle by hand or external force

Step3. According to the figure below, calculate the total number of positive and negative half cycles of the counter electromotive force, that is, the number of poles of the motor (The number of poles is a multiple of 2)



SMO:

Sliding-Mode Current Observer Mainly used to estimate the motor rotor position , A certain degree of toughness to parameter change disturbances , So the SMO_G & SMO_F parameters do not need to be calculated by the R and L parameters.

When the motor runs to the rated power , if the current diverges abnormally, it will cause shutdown protection , dSMO parameters can be tuned by try and error (**G must be less than F**).

SMO_Kslf is the filter coefficient of the sliding mode current observer when estimating the rotor position , This parameter is linear with the electrical frequency of the motor , also indirectly affects the phase relationship between the actual rotor position and the estimated rotor position .

When the motor runs to the rated power , if the current diverges abnormally, it will cause shutdown protection , SMO_Kslf parameters can be tuned by try and error .

$SMO_Gain = Kslide1/MaxSmcError$, Kslide is SMO linear gain, SMO can be adjusted appropriately_ Gain to indirectly change Kslide1.

1.2 MOC and hardware parameter configuration

- **Configuration file : Pwm.h <Configuration Wizard>**
- **Set MPWMDATA**

Parameter	Value range/option	Description	Default value	Unit
Set PWM Frequency	12000~40000(multiple of 10)	Carrier frequency	15000	Hz

- **Set MPWMDDB**

Parameter	Value range/option	Description	Default value	Unit
Set Deadband Time	1us/1.5us/2us/3us/4us/5us	Set dead-time	1us	--

Suggestions for adjusting direction :

PWM output frequency :

The higher the PWM frequency setting , although it will increase the switching loss of the MOSFET and the heat of the motor , affecting the system efficiency , but it can effectively suppress the current/power/speed ripple, vibration and noise when the motor is running in high-speed applications , and improve the overall performance of the system . And the frequency setting is low, it is easy for the human ear to hear (below 16K) , the maximum speed of the motor can be increased , It is recommended that a motor with a small inductance value should be set to a higher carrier frequency .

Set dead-zone :

Determine the required dead zone range according to the measured output waveform of the upper and lower arms , excessive dead zone will affect the speed characteristics of the motor , may not increase to rated top speed.

- **Configuration file : Motor.h <Configuration Wizard>**
- **Set Rshunt and OPA Gain**

Parameter	Value range/option	Description	Default value	Unit
Rshunt	--	Set the sampling phase resistance value	100	0.1mR
OPA GAIN	1 Gain /2.5 Gain /5 Gain /10 Gain	Internal OPA gain	5 Gain	--

Suggestions for adjusting direction :

MDSFxx Internal OPA Gain have 1 、 2.5 、 5 、 10 four Amplification Gain Values to Choose , the matching principle depends on the current sampling resolution and the rated phase current of the motor .

ShuntR * motor maximum current (I_{Peak}) < 0.5V

Both ends of the differential will be limited to less than 0.5V because the maximum voltage across the internal OPA is 0.5V.

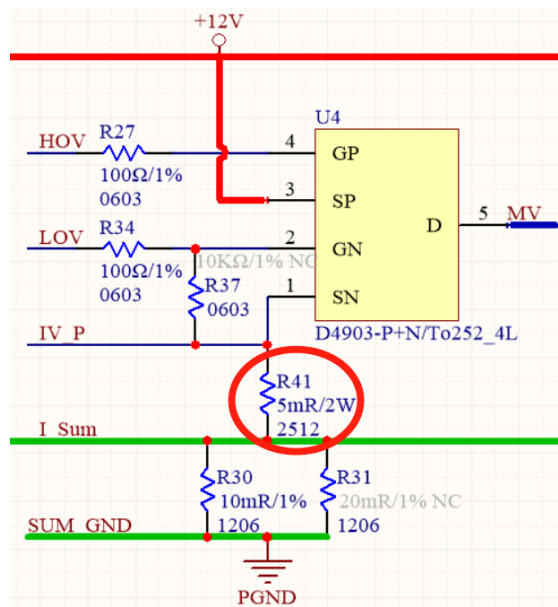
OPA Gain * ShuntR * motor maximum current (I_{Peak}) < 2.5V

The 2.5V is limited because the internal A/D is offset by 2.5V . Phase current has positive/negative, 0~2.5V is negative current, 2.5V~5V is positive current.

Suggestions for Sampling Resistor Selection :

The recommended maximum value of the general current sampling A/D corresponds to twice the motor I_{peak}(max), for example :

DC24 V motor , Phase current at maximum rated speed I_{peak} = 2.7A , choose 3A , then sample twice the maximum current to 6A . R_{Shunt} = 2.5V / Gain / 6A , Gain choose 5 first , then Rshunt = 83mR , take 0.1R for use .



The larger the resistance value of the sampling resistor, the higher the sampling accuracy. However, the temperature rise caused by the power dissipation of the sense resistor also needs to be considered. In the case of ensuring that the temperature rise of the resistance meets the requirements, the resistance value should be increased as much as possible.

■ **Configuration file : Pwm.h <Configuration Wizard>**

■ **Gate driver output mode**

Parameter	Value range/option	Description	Default value
Set MPWM SWAP	MDSF05/ MDSF40 MDSFA0	Gate driver output mode	MDSF05/MDSF40

Suggestions for adjusting direction :

When use MDSF05(MCU : need external Gate-Driver 、MOSFET or IPM) and MDSF40 (MCM : built-in P/N-type Gate-Driver , need external MOSFET) , select” **MDSF05/MDSF40**” , conversely , when use MDSFA0 (MCM : built-in N/N-type Gate-Driver , need external MOSFET) , select” **MDSFA0**” .(**Note: This option must not be selected wrongly, otherwise it may cause MOSFET damage**)

■ **Configuration file : Moc.h <Configuration Wizard>**

■ **Space Vector Pulse Width Modulation (SVPWM) configuration**

■ **Set MOTOR_CONT2**

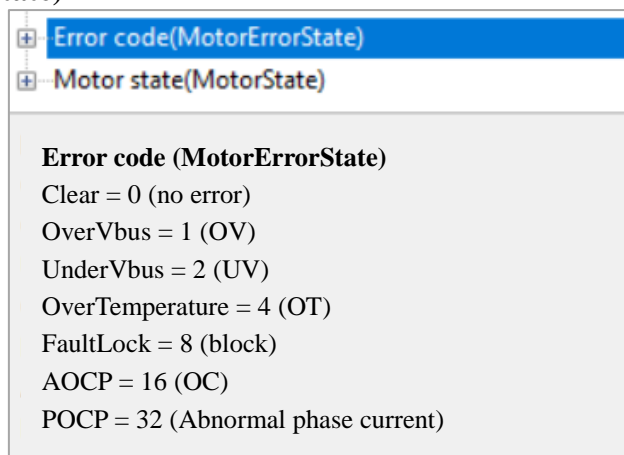
Parameter	Value range/option	Description	Default value
SVPWMMODE	7-SVPWM 5-SVPWM	SVPWM mode	5-SVPWM

Suggestions for adjusting direction :

(a). When product or system prioritizes efficiency , optional " **5-SVPWM** " , then SVPWM will switch the output in 5 segments , reduce MOSFET switching loss and heat generation , conversely, when the product or system takes silence, vibration and interference as the appeal and key indicators , optional " **7-SVPWM** " .

1.3 Protection parameter configuration

- Configuration file : Motor.h <Configuration Wizard>
- Protection / Error status identification (ErrorStatus)
- Error code(MotorErrorState)



Suggestions for adjusting direction :

When the system stops abnormally or fails to start , protection status flag can be used , it is known that currently Error_Code output code name , determine the protection status of the system . Available in main.c file (line 179) , look for if(UartFlag == 1) this judgment , below line 195 the Debug_(A~F) any parameter make it equal to " **MotorErrorState** " this variable , the example is as follows :

```
if(UartFlag == 1){  
    SFR_PAGE = 0; Debug_A = eprom_data;  
    SFR_PAGE = 0; Debug_B = PI_OUT;  
    SFR_PAGE = 1; Debug_C = PI_OUT;  
    SFR_PAGE = 2; Debug_D = PI_OUT;  
    SFR_PAGE = 2; Debug_E = PI_UI;  
    SFR_PAGE = 3; Debug_F = MotorErrorState;  
  
    Uart_Package_Build();  
    UartFlag = 0;  
}
```

Add this variable

In UartSystem.h <Configuration Wizard> UART function Enable , and open " DataLogger.exe " software can monitor this variable .

Configuration file : Motor.h <Configuration Wizard>

- Set motor protection function
- Overvoltage/Undervoltage protection (OVP/UDP) (✓)

Parameter	Value range/option	Description	Default value	Unit
Set Vbus A/D Channel	CH0~CH7	Sampling V-bus voltage A/D channel	CH6	--
Set Vbus rate parameter	0~65535	Sampling V-bus voltage calibration parameters = (Vbus_avg / Input voltage)*1000	27500	--
OVP Values	0~4000	Overvoltage value	230	0.1V
OVP recovery Values	0~4000	Overvoltage recovery value	200	0.1V
UVP recovery Values	0~4000	Undervoltage recovery value	75	0.1V
UVP Values	0~4000	Undervoltage value	60	0.1V
OVP/UDP judgment times	0~3000	OVP/UDP judgment times (every 10ms)	50	times

Suggestions for adjusting direction :

Vbus rate parameter :

Confirm the sampling Vbus AD channel first , and fill in V_BUS_CH , later " Vbus_avg " this variable put UART Debug monitor (as above) . Divide the obtained Vbus_avg by the current input voltage after power-on , multiplied by 1000, the value obtained is the **Vbus rate parameter** , the example is as follows :

Suppose the working voltage is DC12V , the resulting Vbus_avg is 330 , $330 / 12 = 27.5$
 $27.5 * 1000 = 27500$, **Vbus rate parameter** = 27500 .

OVP /UDP setting :

Set the required overvoltage protection value and undervoltage protection value , fill in separately **OVP Values** 、**UVP Values** parameter , overvoltage recovery value and undervoltage recovery value , fill in separately **OVP recovery Values** 、**UVP recovery Values** .

OVP/UDP judgment times the parameter is judged OVP/LVP Debounce times , the number of judgments is judged every 10ms .

- Locked-rotor protection (LRP) (✓)

Parameter	Value range/option	Description	Default value	Unit
Motor speed abnormally high value	--	Abnormally high value of motor speed	400	rpm
Motor speed abnormally low value	--	Abnormally low value of motor speed	20	rpm
LRP judgment cycle	10~30000	Motor blocked judgment cycle	2000	ms

Suggestions for adjusting direction :

LRP setting :

According to the highest/lowest rated speed of the motor , add or subtract a value , to judge that the motor is blocked . Higher than **Motor speed abnormally high value** is over speed protection , lower than **Motor speed abnormally low value** is under speed protection .

LRP judgment cycle parameter is the cycle of judging LRP , it is recommended not to set too small to avoid misjudgment .

■ Over temperature protection (OTP) (✓)

Parameter	Value range/option	Description	Default value	Unit
Set OTP A/D Channel	CH0~CH7	Sampling OTP A/D channel	CH3	--
OTP A/D Values	0~1023	Overheating A/D value	900	--
OTP recovery A/D Values	0~1023	Over temperature recovery A/D value	800	--
OTP judgment times	0~3000	OTP judgment times (every 10ms)	5	times

Suggestions for adjusting direction :

OTP setting :

Confirm the OTP AD channel first , and fill in **Set OTP A/D Channel** • **OTP A/D Values** : is the A/D value of over-temperature protection. **OTP recovery A/D Values** : is the A/D value to clear the over-temperature protection

If you can know the R/T table of NTC, you can directly use the partial pressure formula to make calculations.

OTP judgment times parameter is judging OTP Debounce times , judged every 10ms.

■ Lack-Phase Protection(LPP) (✓)

Reserved; pending verification of necessity

■ Phase current protection (✓)

Reserved; pending verification of necessity

- Phase Ia 、 Ib 、 Ic parameter (unit : mA) (700)
- PHASE_OCP_DURATION parameter (unit : ms) (50)

Suggestions for adjusting direction :

(a). Abnormal phase current protection , the three-phase current is greater than the set **Phase Ia & Ib & Ic parameter** considered abnormal current protection , Set the direction to place IaFb, IbFb on UART for observation . Record the average value when the motor is running at the rated speed, and fill in this parameter by about 1 times.

(b). **PHASE_OCP_DURATION parameter** : Judgment phase current delay time

- **Set Protection to retry**
 - AOCN_Retry_Enable (✓)
 - POCP_Retry_Enable (✓)
 - FaultLock_Retry_Enable (✓)
 - MotorLackPhase_Run_Retry_ENABLE (✓)

Parameter	Value range/option	Description	Default value	Unit
Set the number of retries	0~255	Number of restarts after error	5	times
Set retry delay time	0~32767	restart delay time	1000	ms

Suggestions for adjusting direction :

After the corresponding protection function Retry Enable is triggered , then when the protection occurs, it will automatically restart, otherwise, it needs to be powered off again .

Set the number of retries : It is the number of restarts of the motor when an error occurs. Once this value is exceeded, it needs to be powered off and restarted.

Set retry delay time : It is the delay time between restart after an error occurs.

- **Configuration file : Ocp.h <Configuration Wizard>**
- **Hardware over-current protection setting**
- **Set AOCPCONT**

Parameter	Value range/option	Description	Default value
I_SHORT	0.15V/0.2V/0.25V/0.3V/0.35V/0.4V/0.45V/0.5V	Overcurrent reference voltage	0.15V
AOCPEN	Enable/Disable	Analog overcurrent function	Enable
DOCPEN	Enable/Disable	Digital overcurrent function	Disable

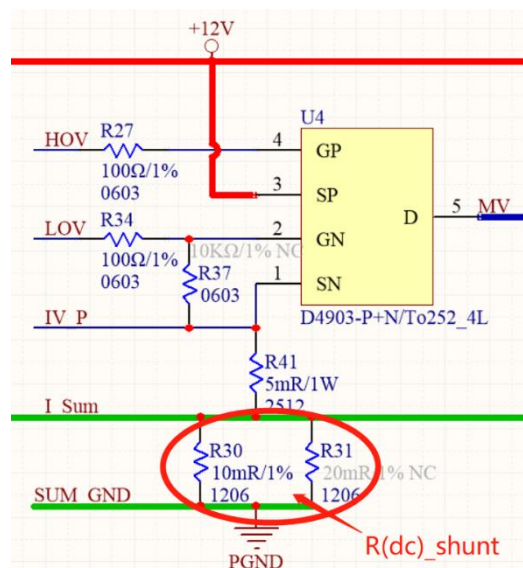
Suggestions for adjusting direction :

MDSF40 provide 8 channel of equivalent overcurrent voltage gears for configuration , the calculation of the bus current protection point is : $I_SHORT = I_{ocp} * R(dc_shunt)$

AOCPEN is analog input for OCP · DOCPEN is digital input for OCP . If the plug-in Driver is IPM, generally is set DOCPEN , otherwise is set AOCPEN.

OCP setting :

The overcurrent value is set at about **1.5 to 2 times the maximum motor current (I_Peak)**.



2. Low Voltage Ceiling Fan adjustment process

2.1 Start parameter adjustment

Step1. Rotor alignment (Parking)

- Configuration file : Motor.h <Configuration Wizard>
- Set the motor tuning process
 - DCR Start Strategy (✓)

Parameter	Value range/option	Description	Default value	Unit
Set DCR_V max values	0~100	DCR final voltage	15	%
Set DCR_V min values	0~100	DCR initial voltage	5	%
Set DCR time	0~10000	DCR overall time	1000	ms
Set DCR Zone1 percentage of time	0~100	0~180 duty	80	%
Set DCR Zone2 percentage of time	0~100	181~359 duty	20	%

- Square Wave Parking Start Strategy (✓)

Parameter	Value range/option	Description	Default value	Unit
Set Startup_duty	0~32767	Startup duty value	256	--
Set Square Parking Time	0~32767	Parking time	500	ms

Suggestions for adjusting direction :

When the motor starts, there will be a specific position that is likely to cause startup failure. Either DCR Start Strategy or Square Wave Parking Start Strategy can be turned on.

In the DCR strategy, the DCR_V value is reasonably adjusted according to the different sizes of the load fan blades. Generally, the heavier the load, the higher the DCR_V value required, and the longer the DCR_time, the longer the positioning oscillation time.

It is not recommended to set too long a time here to avoid too long a start-up time. The parameters can be adjusted according to the trial-and-error method to find the suitable starting parameters for the motor.

The DCR Zone1 percentage of time and DCR Zone2 percentage of time parameters should add up to 100, and generally Zone1 is set larger than Zone2.

Square Wave Parking Start Strategy : This strategy is not recommended for the time being; Startup duty is to set the duty amount, this value is recommended not to set too big at first to avoid too much starting current. Square Parking Time is to set the overall positioning time, it can be increased or decreased according to the positioning situation.

Step2. Starting

- Configuration file : Motor.h <Configuration Wizard>
- Set the motor tuning process

Parameter	Value range/option	Description	Default value	Unit
Set IQ parking duration	1-32767	IQ parking duration	1	ms

- Set FOC LOOP Parameter
 - IQ
 - ✓ Set IQ Current parameter

Parameter	Value range/option	Description	Default value	Unit
Set IQ Initial current	--	IQ initial current	4000	mA
Set IQ Starting current	--	IQ starting current	3000	mA

Suggestions for adjusting direction :

Due to the high inertia of the ceiling fan rotor, the starting current/torque during start-up needs to be larger. The load is appropriately adjusted according to the size of the load, so as to effectively complete the start-up, reduce the vibration and the chance of reversal.

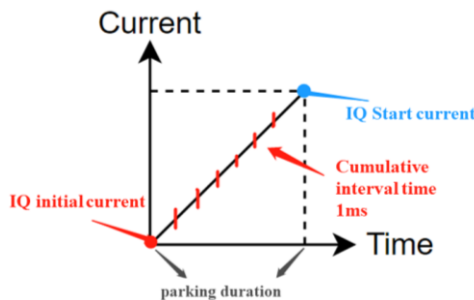
Moderate increase in time can improve start-up failure rate.

The slope and time of acceleration of the bit current are described below:

As shown in the figure on the right, the positioning duration is the total time from the initial current to the start current, and the cumulative interval is the Timer0 time (1ms), so the accumulated current per 1ms is:

(IQ Starting current - IQ Initial current) / parking duration

parking duration is at least 1ms, if parking is not required, the initial current = Starting current



Step3. Position open loop operation (Open loop)

- Configuration file : Motor.h <Configuration Wizard>
- Set the motor tuning process

Parameter	Value range/option	Description	Default value	Unit
FOC_Control_Stage	Standby/ OpenLoop/ CloseLoop	FOC control Stage	CloseLoop	--
Set SMO_RAMP acceleration slope	1~32767	PLL acceleration slope	28	ms
Set PLL accumulation	1~100	PLL accumulation	1	--
Set SMO_PLL initial speed	--	PLL initial speed	5	rpm
Set SMO_PLL end speed	--	PLL end speed	35	rpm

Suggestions for adjusting direction :

FOC Control Stage :

Standby : This stage does not perform any FOC function and can be used for initial calibration of IA and IB parameters.

OpenLoop : During open loop, it is current closed loop and position open loop . When adjusting for open loop at the initial stage of adjustment, it is recommended to first adjust **FOC_Control_Stage** to **OpenLoop** .

In this way, the motor will only run in an open loop state , when an irreversible phenomenon occurs during the operation of the motor . **FOC_Control_Stage** can also be used to analyze the problem of open loop or closed loop.

CloseLoop : Optional speed outer ring , or power outer ring .

PLL :

The reduction of **SMO_RAMP** can effectively improve the convergence speed of the position observer and reduce the open-loop startup time . But if it is too small, it may cause the startup too fast to fail . It should be noted that this variable value also includes the PLL acceleration slope under the downwind or headwind.

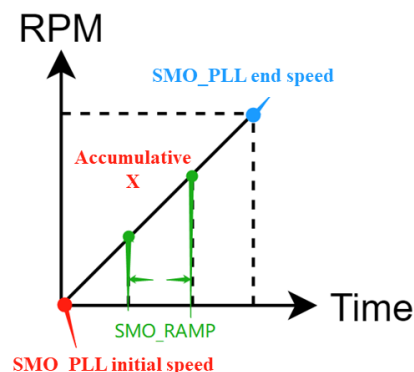
The open-loop acceleration slope and time description are as follows:

As shown on the right , it is the slope from the initial speed of SMO_PLL to the end speed of SMO_PLL

Every SMO_RAMP time to accumulate a PLL amount
(for the accumulation number X in the figure)

Therefore, **the total time of PLL open loop is calculated as:**

$$\frac{[(\text{SMO_PLL end speed} - \text{SMO_PLL initial speed}) / (\text{PLL accumulation} / \text{POLE PAIRS})] * \text{SMO_RAMP}}$$



Step4. Open loop to closed loop

- **Configuration file : Motor.h <Configuration Wizard>**
- **Set the motor tuning process**

Parameter	Value range/option	Description	Default value	Unit
Set SMO_DELAY Delay time	1-32767	Open loop into closed loop delay time	10	ms

SMO_DELAY is the delay time for the inner loop to prepare to enter the closed loop, and this parameter can be adjusted appropriately. Parameters are reserved and not used for now.

- **Set FOC LOOP Parameter**

- **IQ>**

✓ **Set IQ Current parameter**

Parameter	Value range/option	Description	Default value	Unit
Set IQ End current	--	Initial value of closed loop current	2500	mA

Suggestions for adjusting direction :

The current after the end of the open loop is the IQ end current , after this value enters the closed loop, it will be directly filled in the integral value and output value of the closed loop . When the position open loop (current closed loop) cuts into the closed loop, since the integrator of the initial outer loop PI controller is zero . In order to avoid the abnormal ripple of the speed during the connection process, it is necessary to provide an initial value for the integrator of the outer loop PI controller.

Step5. Closed loop operation (Close loop)

- Configuration file : Motor.h <Configuration Wizard>
- Set motor control program
 - Set the main control loop
 - ✓ Constant Phase Current control Enable/Disable(✓)
 - ✓ Constant Speed control Enable/Disable(✓)
 - ✓ Constant Power control Enable/Disable(✓)

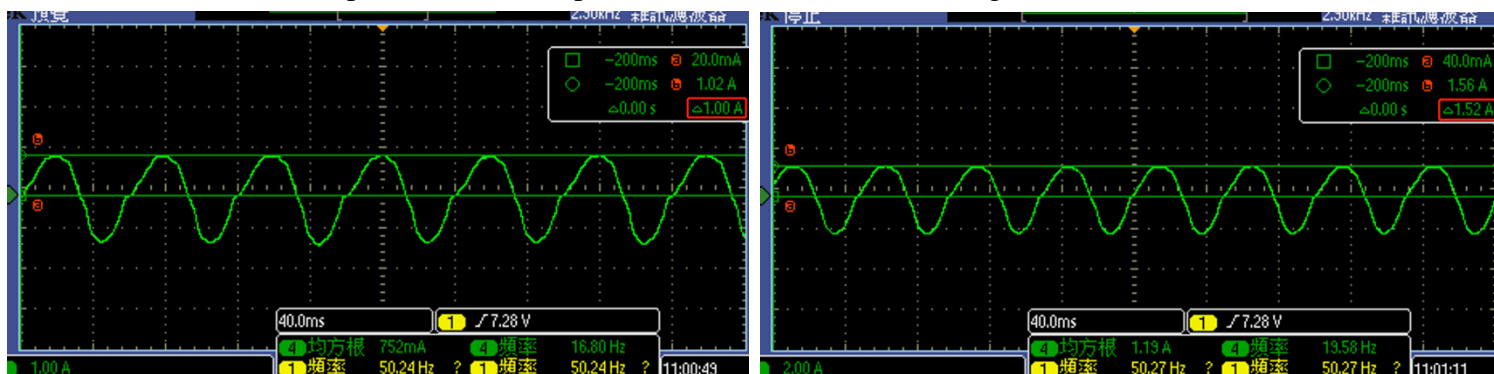
There are three control modes in the closed-loop control, which are **Phase Current control** & **Speed control** & **Power control**, described below.

Phase Current control

Parameter	Value range/option	Description	Default value	Unit
Set the rated output phase current (max)	0~30000	Output phase current rated value (max)	4000	mA
Set PHASE CURRENT_RAMP slope	1~32767	Phase current control cycle	20	ms
Set PHASE CURRENT_CMD accelerate RAMP	0~32767	Accumulated amount of phase current	5	--
Set PHASE CURRENT_CMD decelerate RAMP	0~32767	Decelerate of phase current	10	--

Suggestions for adjusting direction :

Fill in the setting value according to the maximum rated phase current of the motor. The current controlled here is the peak value of the phase current, as shown in the figure below:



Set the rated output phase current (max) = 1A

Set the rated output phase current (max) = 1.5A

The current control slope can be adjusted according to **CURRENT_RAMP** & **CURRENT_CMD accelerate RAMP** & **CURRENT_CMD decelerate RAMP**. Direction is every **CURRENT_RAMP** time, to add or subtract the amount of **CURRENT_CMD**.

The conversion formula of accumulated subtracted phase current and actual accumulated/subtracted phase current is:

$$\text{Actual accumulative or subtractive phase current} = (\text{Accumulation or decrement phase current}) / [(R_SHUNT * OPA_GAIN) * (1023/5) * 64]$$

Speed control

Parameter	Value range/option	Description	Default value	Unit
Set the rated output speed (max)	0~(32767/ Motor Pole/2)	Rated speed (max)	380	rpm
Set SPEED_RAMP slope	1~32767	Speed control cycle	20	ms
Set SPEED_CMD accelerate RAMP	0~32767	Accumulated speed	15	--
Set SPEED_CMD decelerate RAMP	0~32767	Decrement speed	5	--

Suggestions for adjusting direction :

Fill in the setting value according to the maximum rated speed of the motor, it is recommended to reserve some margin . The maximum speed command control can be set by **Set the rated output speed (max)**. The speed control slope can be adjusted according to **SPEED_RAMP & SPEED_CMD accelerate RAMP & SPEED_CMD decelerate RAMP** . Direction is every **SPEED_RAMP** time , to add or subtract the amount of **SPEED_CMD** . **The addition and subtraction here are divided by Pole pairs to get rpm.**

The maximum output value of the internal angular velocity is 32767 . So fill in the upper limit of the maximum value of the rated speed is 32767 divide by number of pole pairs . If the rated speed of the motor exceeds this upper limit, please go to **Pwm.h <Configuration Wizard>** . Fill in the parameter of Set Base Angular Velocity = rated maximum speed * Pole Pairs.

Power control

Parameter	Value range/option	Description	Default value	Unit
Set the rated output power (max)	0~10000	Rated power (max)	5100	0.01W
Set power magnification parameters	--	Power correction value	1840	--
Set I_BUS A/D Channel	CH0~CH7	Sampling Ibus A/D channel	CH2	--

■ Set Ibus Limit

Parameter	Value range/option	Description	Default value	Unit
Ibus limit values	0~100	Limit bus current value	35	0.1A
Ibus OPA GAIN	0~100	Internal OPA gain	11	--
Ibus Correction factor	0~1000	Bus current correction factor	140	--

Suggestions for adjusting direction :

Fill in the set value according to the rated power of the motor . The power command control of the motor can be set by Set the rated output power (max).

If there is a gap between the actual power value and the filled power output , you can use the power correction value to make fine adjustments . Fine adjustments method is:

If the target power is lower than the actual power >> Increase the power correction value

If the target power is higher than the actual power >> Reduce the power correction value

Limit bus current value :

It will be necessary to set this item under wide voltage conditions to do additional settings, as suming that the working voltage is 8v~20v, the rated power needs to be controlled at 51W. The 51W current difference between 8v and 20v (8v/6.375A; 20v/2.55A), provided that the motor c an still output such a high current at low voltage, will additionally increase the function of limiting the bus current.

As with power correction, if there is a difference between the limiting bus current and the actual bus current, then fine-tune the bus current correction value, the correction method can first turn down the limiting bus current, and then do the current correction fine-tuning after making sure the limiting current function has been triggered. **Regardless of whether you are doing power trimming or current trimming, the writer needs to use an isolated type, or remove the writer temporarily after firing in, and then do calibration observation.**

2.2 Peripheral Control Function

Remote Control

■ Configuration file : Motor.h <Configuration Wizard>

- Set motor control program
 - ✓ Set Peripheral Control
 - Remote control commands Enable/Disable(✓)

Parameter	Value range/option	Description	Default value	Unit
Remote control type	RF/IR	Remote type	IR	--

Suggestions for adjusting direction :

Choose according to the type of remote control. If you can't tell whether the remote control is RF or IR, you can see if there is an LED-like component at the transmitter end of the remote control. As shown in the picture on the right.



● Power Level

Parameter	Value range/option	Description	Default value	Unit
Set IRF Button Power1	0~6000	Power 1	500	mW
Set IRF Button Power2	0~6000	Power 2	1000	mW
...It by analogy	0~6000	By analogy	2000	mW

● Speed Level

Parameter	Value range/option	Description	Default value	Unit
Set IRF Button SPEED1	0~ Set the rated output speed (max)	Speed 1	100	rpm
Set IRF Button SPEED2	0~ Set the rated output speed (max)	Speed 2	150	rpm
...It by analogy	0~ Set the rated output speed (max)	By analogy	250	rpm

✓ Observe Speed_UI parameters Enable(✓)

Parameter	Value range/option	Description	Default value	Unit
Set IRF Button SPEED_UI1	0~32767	Speed integral 1	500	--
Set IRF Button SPEED_UI2	0~32767	Speed integral 2	700	--
...It by analogy	0~32767	By analogy	900	--

● Current Level

Parameter	Value range/option	Description	Default value	Unit
Set IRF Button Current1	0~10000	Current 1	800	mA
Set IRF Button Current2	0~10000	Current 2	1000	mA
...It by analogy	0~10000	By analogy	2000	mA

According to **Set the main control loop** external ring control mode set **Power Level** or **Speed Level** or **Current Level** . Each Level corresponds to the numeric function on the remote control button.

If you don't need so many functions, you can close it in **RF_Decode.h/IR_Decode.h** (remote decoding will be explained separately later), and Level is not enough to increase by yourself, as the following example .

Add the following codes in the **red box** according to the control mode (as shown below).

■ Configuration file : Motor.h <Configuration Wizard>

```

378 // <o> Set IRF Button Power7 (unit : mW)<0-6000>
379 #define IRF_Power_7 5100
380
381 // <o> Set IRF Button Power8 (unit : mW)<0-6000>
382 #define IRF_Power_8 5100
383 </h>

431 // <o> Set IRF Button SPEED_UI7 <0-32767>
432 #define IRF_SPEED_UI_7 1400
433
434 // <o> Set IRF Button SPEED_UI8 <0-32767>
435 #define IRF_SPEED_UI_8 1400
436 /h>

404 // <o> Set IRF Button SPEED7 (unit : rpm)
405 #define IRF_SPEED_7 (short)((float) 380 * 32767 / (BASE_RPM))
406
407 // <o> Set IRF Button SPEED8 (unit : rpm)
408 #define IRF_SPEED_8 (short)((float) 380 * 32767 / (BASE_RPM))

463 // <o> Set IRF Button Current7 (unit : mA)<0-10000>
464 #define IRF_Current_7 (float)4500/1000
465 #define IRF_Current_7_VALUE (int16)((float) IRF_Current_7 * I_AMPLIFIER) // unit : A * Gain
466
467 // <o> Set IRF Button Current8 (unit : mA)<0-10000>
468 #define IRF_Current_8 (float)4500/1000
469 #define IRF_Current_8_VALUE (int16)((float) IRF_Current_8 * I_AMPLIFIER) // unit : A * Gain
470 </h>

1309 enum Remote_Control_Speed_Level{
1310 R_1_SPEED = 1,
1311 R_2_SPEED = 2,
1312 R_3_SPEED = 3,
1313 R_4_SPEED = 4,
1314 R_5_SPEED = 5,
1315 R_6_SPEED = 6,
1316 R_7_SPEED = 7,
1317 R_8_SPEED = 8
1318 };

```

■ Configuration file : Motor.c

```

178 #if (IRF_TRI == 1)
179 typedef struct{
180 int tableData[9]; // Fill in the Level number + 1
181 int tableDataUI[9];
182 }ConstIRF;
183
184 #if (POWER_CONTROL == 1)
185 code const ConstIRF rtConstPwr = {
186 {0, IRF_Power_1, IRF_Power_2, IRF_Power_3, IRF_Power_4, IRF_Power_5, IRF_Power_6, IRF_Power_7, IRF_Power_8}
187 };
188 #endif
189 #endif

184 #if (SPEED_CONTROL == 1)
185 code const ConstIRF rtConstSpd = {
186 {0, IRF_SPEED_1, IRF_SPEED_2, IRF_SPEED_3, IRF_SPEED_4, IRF_SPEED_5, IRF_SPEED_6, IRF_SPEED_7, IRF_SPEED_8},
187 {Spd_MinLimit+1, IRF_SPEED_UI_1, IRF_SPEED_UI_2, IRF_SPEED_UI_3, IRF_SPEED_UI_4, IRF_SPEED_UI_5, IRF_SPEED_UI_6, IRF_SPEED_UI_7, IRF_SPEED_UI_8}
188 };
189 #endif

```



```
197 #if (POWER_CONTROL == 1)
198     code const ConstIRF rtConstPwr = {
199         {0, IRF_Power_1, IRF_Power_2, IRF_Power_3, IRF_Power_4, IRF_Power_5, IRF_Power_6, IRF_Power_7, IRF_Power_8}
200     };
201 #endif
```

The following code is added in the red box according to IR control or RF control respectively.

■ Configuration file : RF Decode.c

```
141 #if(e_RF_SPEED7 == Enable)
142     case RF_DATA_SPEED7: IRF_Channel = R_SPEED; IRF_SPEED_Level = R_7_SPEED; break;
143 #endif
144
145 #if(e_RF_SPEED8 == Enable)
146     case RF_DATA_SPEED8: IRF_Channel = R_SPEED; IRF_SPEED_Level = R_8_SPEED; break;
147 #endif
```

■ Configuration file : RF Decode.h <Text Editor>

```
253 //<e> Enable Speed7 Button
254 #define e_RF_SPEED7 0
255 #if(e_RF_SPEED7 == 1)
256     //<o> Button speed 7<0-65535>
257     #define RF_DATA_SPEED7 65533
258 #endif
259 //</e>
260
261 //<e> Enable Speed8 Button
262 #define e_RF_SPEED8 0
263 #if(e_RF_SPEED8 == 1)
264     //<o> Button speed 8<0-65535>
265     #define RF_DATA_SPEED8 65533
266 #endif
267 //</e>
```

■ Configuration file : IR Decode.c

```
86 #if(e_IR_SPEED7 == 1)
87     case IR_DATA_SPEED7: IRF_Channel = R_SPEED; IRF_SPEED_Level = R_7_SPEED; break;
88 #endif
89
90 #if(e_IR_SPEED8 == 1)
91     case IR_DATA_SPEED8: IRF_Channel = R_SPEED; IRF_SPEED_Level = R_8_SPEED; break;
92 #endif
```

■ Configuration file : IR Decode.h <Text Editor>

```
217 //<e> Enable Speed7 Button
218 #define e_IR_SPEED7 0
219 #if(e_IR_SPEED7 == 1)
220     //<o> Button speed 7<0-65535>
221     #define IR_DATA_SPEED7 51255
222 #endif
223 //</e>
224
225 //<e> Enable Speed8 Button
226 #define e_IR_SPEED8 1
227 #if(e_IR_SPEED8 == 1)
228     //<o> Button speed 8<0-65535>
229     #define IR_DATA_SPEED8 51255
230 #endif
231 //</e>
```

RF Decode

■ Configuration file : RF_Decode.h <Configuration Wizard>

- Customize
 - ✓ Set RF_DEC_CTRL

Parameter	Value range/option	Description	Default value	Unit
RFIN_SEL	P0.5(IR1) / P3.3(IR2)	Select the receive code input pin	P3.3(IR2)	--

✓ Set RF_DEC_SET

Parameter	Value range/option	Description	Default value	Unit
CLK_DIV_SEL	24MHz/16MHz/8MHz/6MHz/3MHz/2MHz	Data decoding clock frequency	2MHz	--
DIN_DB_SEL	0ns /250ns /500ns 1000ns	Data bounce time	500ns	--
DIN_TYP	Data Type1/Data Type2	Data type	Data Type1	--
DOUT_REV	First input data is LSB/ First input data is MSB	Decoding output data reversal	First input data is LSB	--
HEADER_EN	No HEADER / With HEADER	Headcode selection	No HEADER	--

Suggestions for adjusting direction :

First confirm the RF DATA connection pin, and select the input pin in RFIN_SEL (as shown in the figure on the right).

CLK_DIV_SEL Setting : The upper limit of the internal Counter is 32767, and the clock frequency is set without exceeding this value, the formula is as follows :

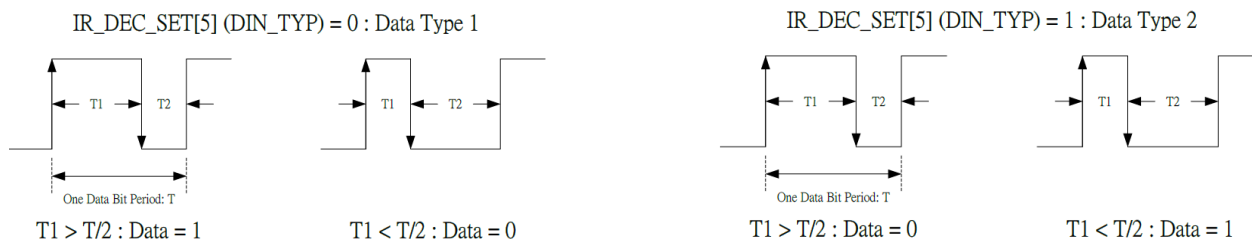
CLK_DIV_SEL <= 32767 / (Data Time Width)

For example, the maximum RF data width is Stop(10ms), then **CLK_DIV_SEL** <= 32767 / 10ms

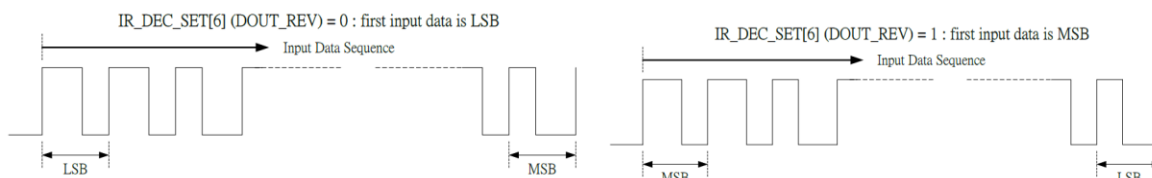
>> **CLK_DIV_SEL** <= 3.2MHz

IR1/CH3/P0.5	33
CAP1/CH4/P0.6	32
CAP2/CH5/P0.7	31
INT0/CH6/P3.2	30
IR2/INT1/CH7/P3.3	29

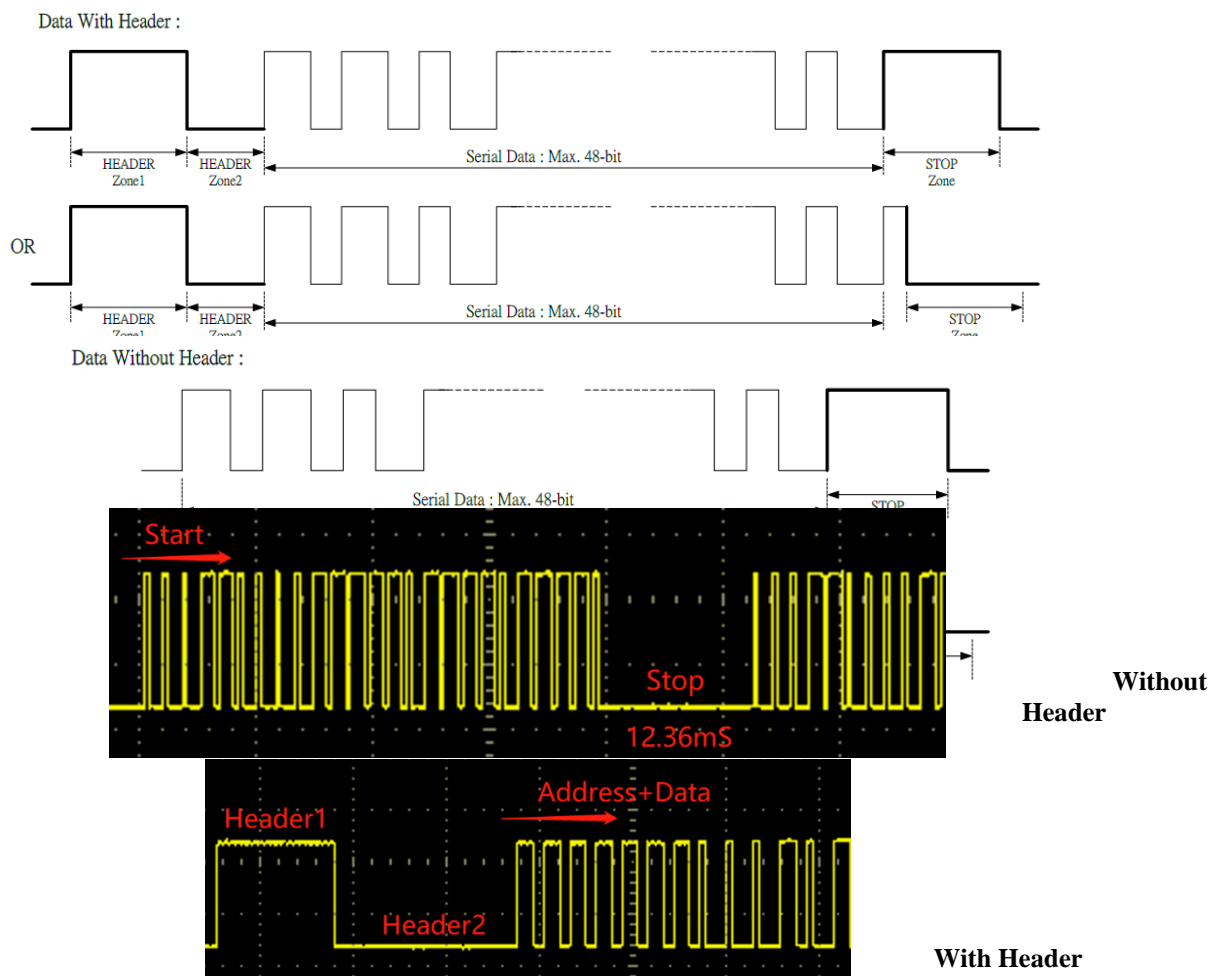
DIN_TYP Data Type : Data 0 and 1 are judged in the following format :



DOUT_REV Set : Define the first Bit as MSB or LSB.



HEADER_EN Set : Choose whether the header needs to be judged. Note here that if you choose With Header, Zone1 and Zone2 of Header must exist at the same time and cannot be set to 0 to avoid dropping.



- Remote control code set
 - ✓ Head code set

Parameter	Value range/option	Description	Default value	Unit
Set Header Zone1 time width	0~32767	Header 1(High)	5000	us
Set Header Zone2 time width	0~32767	Header 2(Low)	7800	us

- ✓ Stop code set

Parameter	Value range/option	Description	Default value	Unit
Set Stop time width	0~32767	Stop Code	5000	us

- ✓ RF decoding to UR Enable(✓)
- ✓ Learning function (✓)

Parameter	Value range/option	Description	Default value	Unit
Learning data values set	0~65535	Learning data values	243	--
Set learn time	0~20	Learn time	10	sec
Judge the times of successful learning	0~20	Judge the times of successful learning	1	10times

Parameter	Value range/option	Description	Default value	Unit
RF address value set	0~65535	Address value	54399	--

- ✓ RF data values set
- Enable OFF Button(✓)

Parameter	Value range/option	Description	Default value	Unit
Button OFF	0~65535	Off button value	13503	--

- Enable Speed1 Button(✓)

Parameter	Value range/option	Description	Default value	Unit
Button speed 1	0~65535	Speed 1 button value	15903	--

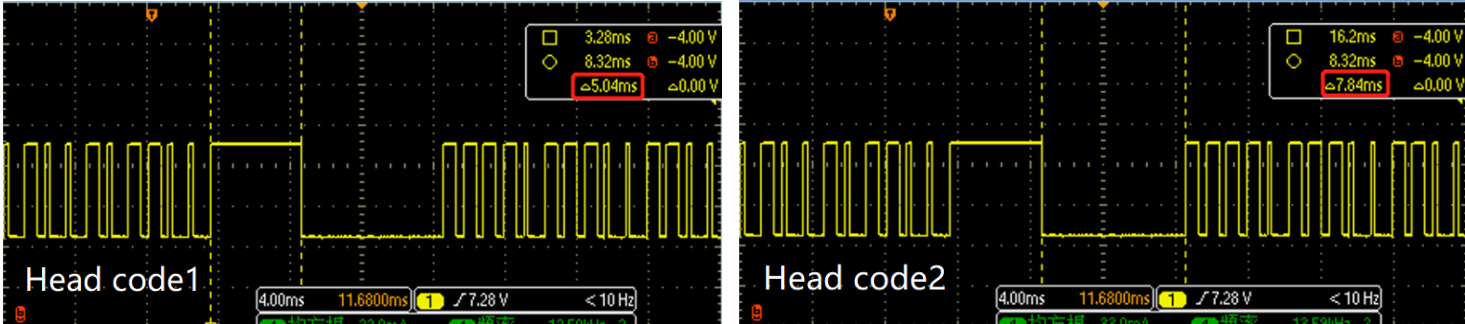
- Enable Speed2 Button(✓)

Parameter	Value range/option	Description	Default value	Unit
Button speed 2	0~65535	Speed 2 button value	13983	--

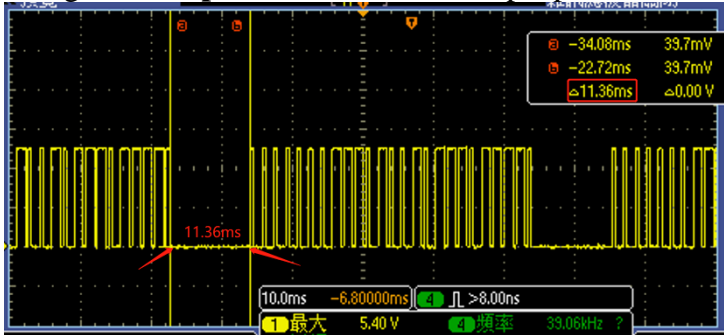
It by analogy...

Suggestions for adjusting direction :

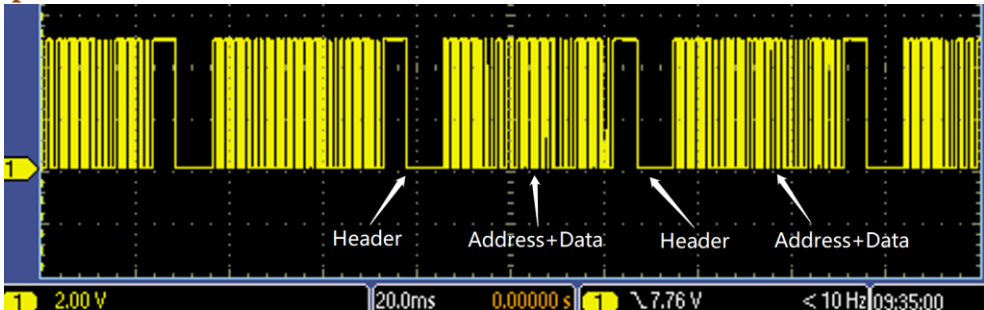
Header Set: Please measure the waveform dropped from RF TX on the oscilloscope, the part with wider width at the beginning of the signal is the header code, the width of High is the **Header Zone1 time width**; the width of Low is the **Header Zone2 time width**, the example picture is as follows.



Stop Code Set: Please measure the waveform from RF TX on the oscilloscope, the ending code part with wider width at the end of the signal is **Stop time width**, the example picture is as follows.



If it encounter a remote control without an end code (as shown below).
It can set the **Stop time width** to be the same as the **Header Zone1 time width**.



Address/data code decoding : After the header code and end code are set, the address code and data code length of the remote control should be determined first.

The code has a default value of 16bits address + 16bits data

If different from the default value, you need to do your own changes in the code, the location as shown in the right figure (RF_Decode.c),
each register is 8bits, please do the displacement according to the remote control the number of bits.

```

83  IR_DEC_CTRL = IR_DEC_CTRL_REGS;
84  }
85  #endif
86  //***** RF_Decode_Read
87  //*****
88  #if (Remote_control_type == dRF_Control)
89  void RFDecode_Read(void){
90  if((IR_DEC_CTRL & 0x10) == 0x10){
91  IR_DEC_CTRL |= 0x01;
92  RF_Decode.RF_address = IR_DOUT0;
93  RF_Decode.RF_address += (IR_DOUT1 * 256);
94  RF_Decode.RF_data = IR_DOUT2;
95  RF_Decode.RF_data += (IR_DOUT3 * 256);
96  #if (RF_decoding_UR_Enable == 1)
97  RF_Decode.RF_address_ur = RF_Decode.RF_address;
98  RF_Decode.RF_data_ur = RF_Decode.RF_data;
99  #endif
100  #if (Remote_control_learning_function == Enable)
101  if(FLAG_RF_Learning_sw==0)
102  RFLearning_mode();
103  if(RF_Decode.RF_address==RF_Decode.RF_Learning_Address[0]+(RF_Decode.RF_
104  #else

```

Judgment of address code/data code bits : Under the same remote control, press different keys and record them separately, the first few same bits are address code, the last few different bits are data code, for example: (The following 0 and 1 are converted according to Page18 **DIN_TYP** data category).

OFF button : 1111 1110 0010 1011 1111 1101 0010 1100

Speed1 button : 1111 1110 0010 1011 1111 1000 0111 1100

Speed2 button : 1111 1110 0010 1011 1111 1001 0110 1100

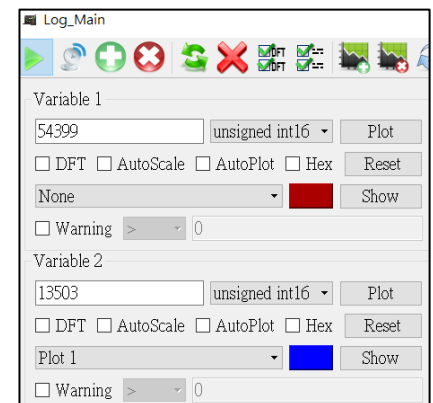
The blue part is the address code, the remaining part is the data code, so it is 16bits address + 16bits data. Although 20bits address + 12bits data is also common in this example, it is recommended to use a multiple of 8 for data processing and writing.

Note : When checking the number of address codes and data code bits, it is recommended to record more than 3 keys to avoid misjudgment.

Check the box **RF decoding to UR Enable** and open Log_Main to observe the decoded hex value. (Note: Please remember to check UART Enable in **UartSystem.h <Configuration Wizard>** before you do this, otherwise the UART function will not work).

As shown on the right, press any key to observe the Variable1 and Variable2 values, 1 for the **RF address value set** and 2 for the **data values** of the key function.

Note : If the **Learning function** is turned on, the **RF address value set** can not be filled in.



Learning function (pair of codes) : Open the **Learning function** and fill in the **Learning data values set** according to the Uart observation method described earlier.

Set learn time for learning time, from the moment of MCU work, after that time will no longer learn.

Judge the times of successful learning is to judge the number of times of successful learning, every 10 times is a unit, meaning that when the MCU judges the number of times the learning button is pressed, the address code of the remote control will be saved to EEPROM when the learning is considered successful.

Remember to turn off **RF decoding to UR Enable** after decoding.

Note : If you need to enable the learning function, please remember to get **Eeprom.h <Configuration Wizard>** and check the EEPROM Enable box.

IR Decode

■ Configuration file : IR_Decode.h <Configuration Wizard>

- Customize
 - ✓ Decoding with Capture(✓)
 - IR decoding to UR Enable(✓)

Parameter	Value range/option	Description	Default value	Unit
Set the number of data to observe	1~50	Observe the first few Bit values	1	--

Parameter	Value range/option	Description	Default value	Unit
Set IR High Head counter_min	0~65535	Headcode minimum	11100	--
Set IR High Head counter_MAX	0~65535	Headcode maximum	15000	--
Set IR High Head counter_error	0~10000	Headcode error	2000	--

✓ Set read IR data format

Parameter	Value range/option	Description	Default value	Unit
IR_data_format	Low_level/ High_level	IR value judgment level	High_level	--

Parameter	Value range/option	Description	Default value	Unit
Set IR 0 data counter	0~65535	Data 0 counter	1681	--
Set IR 1 data counter	0~65535	Data 1 counter	4955	--
Set IR data counter_error	0~10000	Data counter error	400	--
Set the number of bits of IR data	0~16	Data number bit value	16	Bits
Set the number of bits of the IR address	0~16	Address number bit value	16	Bits
Set IR address value	0~32767	Address number value	7038	--

✓ IR data values set

● Enable OFF Button(✓)

Parameter	Value range/option	Description	Default value	Unit
Button OFF	0~65535	Off button value	4335	--

● Enable Speed1 Button(✓)

Parameter	Value range/option	Description	Default value	Unit
Button speed 1	0~65535	Speed 1 button value	32895	--

● Enable Speed2 Button(✓)

Parameter	Value range/option	Description	Default value	Unit
Button speed 2	0~65535	Speed 2 button value	16575	--

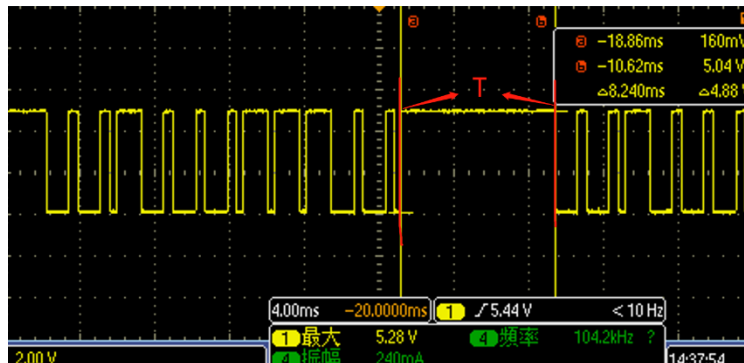
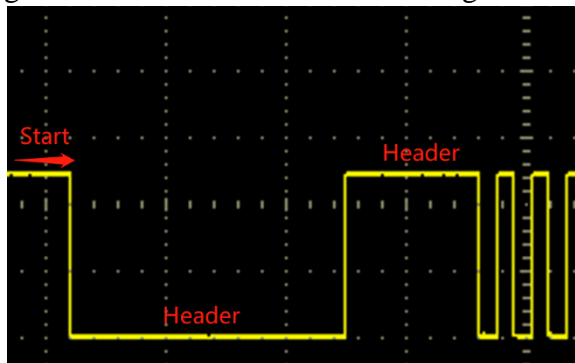
It by analogy...

Suggestions for adjusting direction :

If you are using Capture as a decoder, please first go to **Capture.h <Configuration Wizard>** , and check the External Capture Enable/Disable box . And go to **Interrupt.h <Configuration Wizard>** and check the CAPTURE Interrupt box. Check the pin used for Capture and set it to CAPPINSEL, and finally check the **Decoding with Capture** box.

Decode headcode: Measure the signal of any key and record the time width of the header code on the oscilloscope (as shown in the bottom right). **Because internal Capture can only use positive edge triggering, the following header widths are based on positive half period signals.**

However, if the headcode has both negative and positive half period signals (as shown in the figure below left) . There are two solutions: 1. skip the negative half period signal and judge only the positive half period signal. 2. use TIMER+INT decoding instead.



IR High Head counter value : Measured time width * E_CAPCKS (set from Capture.h)

For example, the top right figure is measured as 8.24ms , CAPCKS set as 48MHz/16

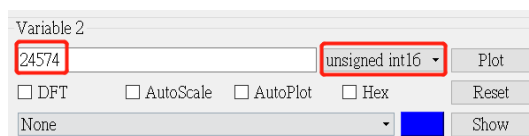
And IR High Head counter = 8240us * 3MHz = 24720.

(Here should pay attention , if the Counter value is too small (for example, are less than 1000), that represents E_CAPCKS to be set further up. It is generally recommended that the headcode Counter should be greater than 10000. If the header Counter is too small, in the later value of the Counter judgment, it is easy to have misjudged the situation)

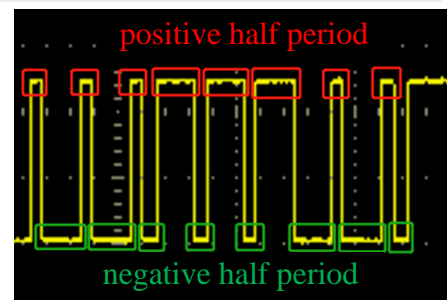
The difference between counter min/MAX is that some remotes have a slight difference in the width of the headcode for each key. So the minimum width measured is entered as min and the maximum is entered as MAX. While if there is not much difference in the width of the headcode for each key, both values are entered as the same.

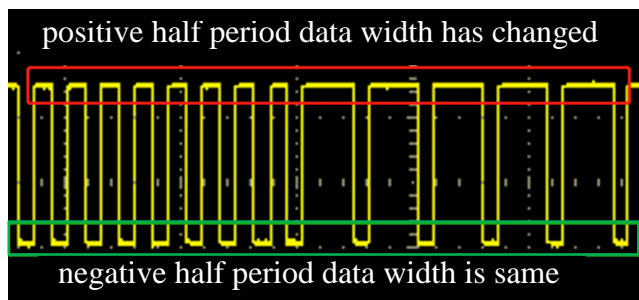
IR High Head counter_error generally fill in IR High Head counter * 10% value (if not a whole number, can be raised up to a whole number). For example, IR High Head counter = 24720; 24720*10% = 2472 . Then IR High Head counter_error fill in 3000 (more estimate to catch some error value)

Turn IR decoding to UR Enable on, when pressing any button, if the value of Variable1 jumps and is similar to the IR High Head counter filled in, it means the head code is decoded correctly.

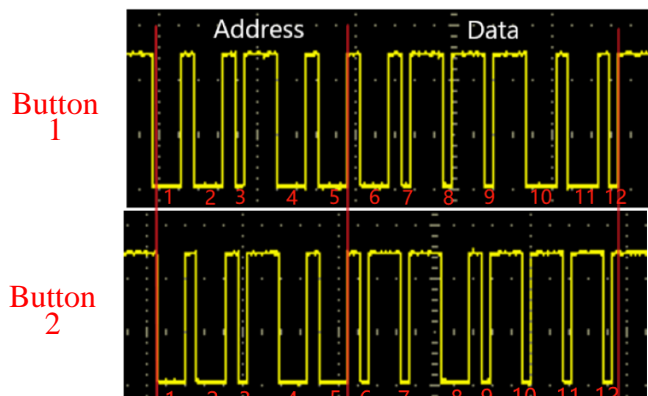


Set IR_data_format: This setting is the way to capture data (including address), Low_level is to capture negative half period data, High_level is to capture positive half period data (as shown on the right) When the data code as shown below appears, it is recommended to set it to High_Level, or use TIMER+INT to decode it instead. (It is obvious to judge 0 and 1 as the main)





Measure the data waveform to confirm the number of address bits and data bits (**the same remote control, different keys, the same code is the address code, different code is the data code**), the example is as follows.



The picture on the left shows the address + data code of the two buttons

It can be observed that from the 6th bit onwards it is different (IR_data_format is set to Low_Level), so the Address bit number is 5 and , the Data bit number is 7.

Fill in 7 Set the number of bits of IR data

Fill in 5 Set the number of bits of the IR address

(It is recommended that when differentiating between Address and Data, please observe more than 3 buttons respectively to avoid misunderstanding when judging only 2 buttons)

Set the definition of data0 and 1. Generally, the short time width is defined as 0; the long time width is defined as 1 .

As shown above, Address is 11011 (set to Low_Level, only see negative half period width), 11011 is decimal "27" . Fill in 27 **Set IR address value**.

Set IR 0 data counter 、Set IR 1 data counter : Determine the Counter number of Data0/1, turn on **IR decoding to UR Enable** . Set the number of data to observe this parameter should be combined with the Variable2 value in UR, for example, the above figure : For example, if I know that the first bit is 1; the third bit is 0, first determine the Counter of Data=1, then fill in 1 in Set the number of data to observe .

And press the button to observe the value of Variable2 and fill the value into the **Set IR 1 data counter**. On the contrary, the Counter of Data=0 is in Set the number of data to observe fill in 3, observe the value of Variable2, fill its value into **Set IR 0 data counter** . **(Note: The judgment sequence must start from the first bit, and fill in the corresponding Counter, then find another data position from left to right, if bit1=0, then find the position of bit=1, and vice versa, but if you jump the bit directly at the beginning, the program may jump the Error directly.)** If the value is the same as IR_address_value, it means the decoding is OK so far.

IR data values set :Finally, put the Variable4 of UR into the corresponding button name, for example, if I press the Off function button and the Variable4 is 198, then put 198 in Button OFF, it by analogy...

VSP control

■ Configuration file : Motor.h <Configuration Wizard>

- Set motor control program
 - ✓ Set Peripheral Control
 - VSP control commands Enable/Disable (✓)

Parameter	Value range/option	Description	Default value	Unit
Set VSP_CH	CH0~CH7	VSP A/D	CH4	--

■ Slope control

Parameter	Value range/option	Description	Default value	Unit
Set VSP srart voltage	0~50	VSP srart voltage	5	0.1V
Set VSP max voltage	0~50	VSP max voltage	45	0.1V
Set VSP stop voltage	0~50	VSP stop voltage	3	0.1V

■ Stage control

Parameter	Value range/option	Description	Default value	Unit
Set VSP SPEED1 speed	--	VSP First section speed	100	rpm
Set VSP SPEED2 speed	--	VSP Second section speed	150	rpm

It by analogy...

◆ Observe Speed_UI parameters Enable(✓)

Parameter	Value range/option	Description	Default value	Unit
Set VSP_SPEED_UI_1	--	VSP First section speed	350	--
Set VSP_SPEED_UI_2	--	VSP Second section speed	800	--

It by analogy...

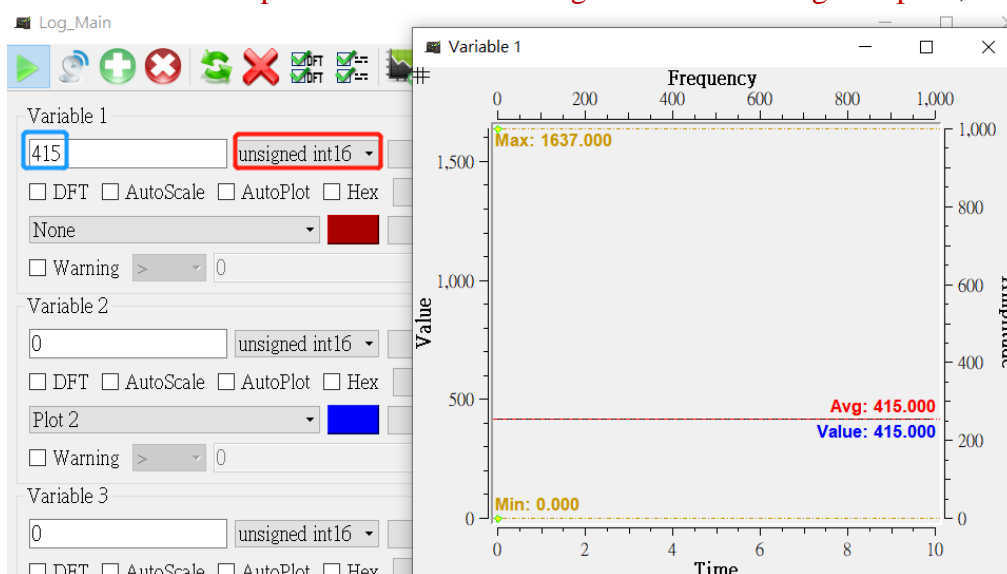
Suggestions for adjusting direction :

For VSP external control function, first confirm the A/D Channel of VSP and fill in VSP_CH . VSP has two control modes, Slope control and Stage control. **VSP is currently only available for testing, and Stage control is currently only available for speed control.** We will update it later if there is any demand.

Supplement : Speed UI

Under the speed external loop, whether it is remote control or VSP control, there is a UI setting parameter that can be used, and this value is set for quick response of acceleration/deceleration, and its adjustment is explained as follows :

After filling in the desired speed values for each segment, turn on **Observe Speed_UI parameters Enable** and observe the **Variable1** parameter under each segment of the ceiling fan speed, as shown below.



Assuming that the **SPEED1 speed** button is pressed, when the ceiling fan speed is stabilized and the value of Variable1 is 416 at this time, fill in 416 **Button1_SPEED_UI** . SPEED2_speed corresponds to Button2_SPEED_UI, it by analogy....

If the actual measurement of the phenomenon of overshoot, this parameter can be fine-tuned downward .

Note that this function is limited to the speed control setting only.

Remember to turn off **Observe Speed_UI parameters Enable** after Speed_UI adjustment.

2.3 Fairwind and Headwind start-up parameter adjustment

- **Configuration file : Motor.h <Configuration Wizard>**
- **Set Fairwind and Headwind judgment function**
 - **BEMF Fairwind/Headwind judgment (resistance) Enable/Disable (✓)**

Parameter	Value range/option	Description	Default value	Unit
BEMF Fairwind/Headwind adjustment process	LEVEL_1/LEVEL2	Fairwind/Headwind adjustment process	LEVEL_2	--

✓ Set BEMF A/D Channel

Parameter	Value range/option	Description	Default value	Unit
Set BEMF_V_CH	CH0~CH7	BEMF_V A/D channel	CH3	--
Set BEMF_W_CH	CH0~CH7	BEMF_W A/D channel	CH5	--

✓ Set Speed for judging Fairwind/Headwind

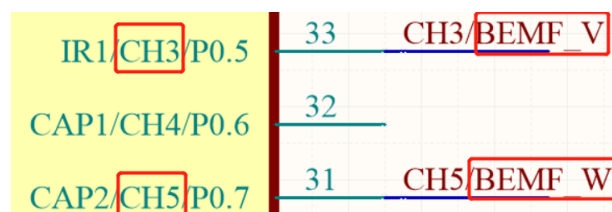
Parameter	Value range/option	Description	Default value	Unit
Set the minimum connection speed for judging Fairwind	--	Determine the minimum speed of fairwind	23	rpm
Set the maximum connection speed for judging Fairwind	--	Determine the maximum speed of fairwind	180	rpm
Set the maximum connection speed to judge the headwind	--	Determine the maximum speed of headwind	100	rpm
Set the duration of Fairwind start judgment	--	Judgment of fairwind/headwind cycles	100	ms
Set Headwind_high_speed_start_Strategy	PLL/ABS	High speed headwind activation strategy	ABS	--

Suggestions for adjusting direction :

First confirm the AD Channel of the two phases of the voltage dividing resistor connection, fill in **Set BEMF_V_CH** and **Set BEMF_W_CH** respectively.

At the early stage of wind adjustment, you can select LEVEL_1 for the BEMF tuning process. At this time, you can test the fairwind/headwind activation status by hand-pulling the fan blade to make the motor rotate while the motor is stationary.

Judgmental articulation fairwind speed range : The condition mechanism of articulating to the wind, between the minimum speed to the maximum speed, can be articulated to the wind, can be fine-tuned according to the actual ceiling fan articulation condition, will limit the minimum speed is because, when the speed is too small, will not be able to judge to the BEMF, so can not know the motor position, so limit a lower limit value.



Judgmental articulation headwind speed range : The following speed values are set here (**Set the maximum connection speed to judge the headwind**). If the speed is greater than this, the motor will be free run until it is lower than this speed, which can be fine-tuned according to the actual motor condition.

Headwind_high_speed_start_Strategy : There are PLL strategy and ABS strategy for backwind convergence, and ABS strategy is currently recommended.

Parameter	Value range/option	Description	Default value	Unit
Set Fairwind/Headwind angle(CW_CCW=0)	0~383	CW_CCW=0 Fairwind/Headwind initial angle	0	--
Set Fairwind/Headwind angle(CW_CCW=1)	0~383	CW_CCW=1 Fairwind/Headwind initial angle	64	--
Set IQ Fairwind/Headwind starting current(IQ_CMD)	--	Fairwind/Headwind starting Iq current	2000	mA
Set ID Fairwind/Headwind starting current(ID_CMD)	--	Fairwind/Headwind starting Id current	0	mA
Set the initial value of Fairwind/Headwind current(IQ_UI)	0~32767	Fairwind/Headwind starting initial inner ring value	8000	--
Set Headwind to Fairwind target speed	--	Headwind articulation into the outer ring speed	45	rpm
Set the judgment counter for the static BEMF value of the motor	0~10000	Judgment motor standstill delay time	300	ms

✓ Observe BEMF_V_W Parameters (✓)

Suggestions for adjusting direction :

Fairwind/Headwind articulation judgment angle set : This parameter (**Fairwind/Headwind angle(CW_CCW=0/1)**) recommended no to change.

Fairwind/Headwind articulation IQ set : Fine tuning according to the articulation into fairwind/headwind conditions. If the motor makes a loud banging sound after articulation, you can fine tune the articulated IQ current downward.

Fairwind/Headwind articulation IQ_UI initial set : This parameter is set so that IQ_OUT will have an initial value when articulating into the fairwind/headwind, and will not start from 0.

Headwind to Fairwind target speed : This value is the speed value that determines the articulation to the close loop when the wind is reversed, too small may cause the articulation failure.

Set the judgment counter for the static BEMF value of the motor : Determine the motor standstill time when the motor speed is too low. It is impossible to rely on the BEMF to determine whether the motor is currently operating at low speed or stationary. Therefore, when BEMF_V-W is below a certain value (the default code is 5, you can go to Motor.c line 765 to make changes) . The set judgment time is the delay time to avoid misjudgment of noise.

2.4 Protection parameter adjustment

■ Configuration file : Motor.h <Configuration Wizard>

■ Set motor protection function

■ Over/under voltage protection

- The parameter configuration is shown in section 1.3 (protection parameter configuration)

Suggestions for adjusting direction :

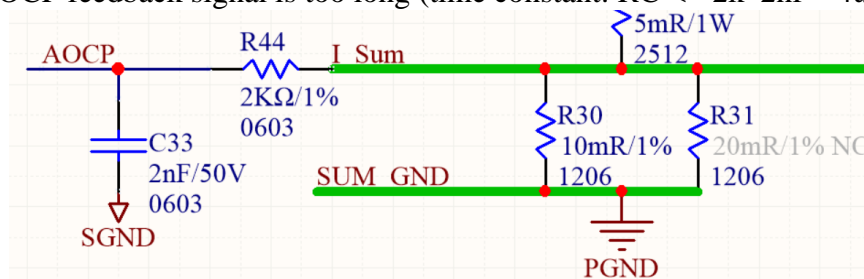
1. Set according to the overvoltage, undervoltage and recovery voltage conditions provided by the customer.
2. Overvoltage and recovery test : After starting the ceiling fan, increase the voltage of the power supply to the overvoltage level . Confirm whether the ceiling fan enters shutdown protection. The voltage of the power supply is then reduced to the overvoltage recovery level , determine whether the ceiling fan is running normally .
3. Undervoltage and recovery test : After starting the ceiling fan, increase the voltage of the power supply to the undervoltage level . Confirm whether the ceiling fan enters shutdown protection. Then raise the voltage of the power supply to the undervoltage recovery level , determine whether the ceiling fan is running normally .

■ **Overcurrent protection - hardware test (key test)**

- The parameter configuration is shown in section 1.3 (protection parameter configuration)

Suggestions for adjusting direction :

1. Verify hardware . Short-circuit the AOCP point and 5V first , the motor can be unconnected. Use UART to monitor whether there is OCP occurs. After confirming that there is no problem, do the following short circuit test.
2. After starting the ceiling fan, short-circuit the two-phase (UV or UW or VW) output at will. Judgment whether the ceiling fan enters shutdown protection. Then turn on the power again to confirm whether the fan can work normally. And repeat this step several times to ensure that the MOSFET is adequately protected.
3. If the ceiling fan cannot be restarted after the short circuit test , check that the specifications of the MOSFET meet or not , whether the AOCP feedback signal has interference on the wiring path , the time constant of the AOCP feedback signal is too long (time constant: $RC \leq 2k \cdot 2nf = 4\mu s$).



■ Motor blocked protection

- The parameter configuration is shown in section 1.3 (protection parameter configuration)

Suggestions for adjusting direction :

Can use hands to hold the fan blades to confirm that the ceiling fan enters the out-of-step/high-frequency state, and whether the blocked protection works normally.

2.5 Other parameter configuration

- Configuration file : Motor.h <Configuration Wizard>
- Set motor control program
 - Set other functions

Parameter	Value range/option	Description	Default value	Unit
Set CW/CCW steering	CW/CCW	Steering direction setting	CW	--
Set Stop_Fun stop speed	--	PWM stop output	60	rpm

Suggestions for adjusting direction :

Set CW/CCW steering is modified according to the desired blade orientation

Stop_Fun the stop speed here refers to when the remote control presses the stop button and the motor speed is lower than this parameter , PWM will stop output . When the speed value is higher than this parameter, PWM is still out put. This is set for a smooth connection with the wind. Its value can be set at the connection point where the tailwind speed cannot be judged at low speed.

✓ CW/CCW test Enable/Disable(✓)

Parameter	Value range/option	Description	Default value	Unit
Running time after changing direction	--	Running time after changing direction	10000	ms
Stop this time before catching the wind	--	Stop this time before catching the wind	2000	ms
Operating time after connecting downwind	--	Operating time after connecting downwind	5000	ms

Suggestions for adjusting direction :

This function is for testing. The test procedure is : Motor start operation (Initially set the direction according to **Set CW/CCW steering**). Transit time is **Running time after changing direction** . Then the motor stopped running , time is **Stop this time before catching the wind** . Then articulate the wind to start , transit time is **Operating time after connecting downwind** . Last change of direction to start , time is **Running time after changing direction** . Continuous cycle...

✓ IPD Enable/Disable(✓)

- Automatic cycle test of initial position Enable/Disable(✓)

Suggestions for adjusting direction :

This IPD function needs to be adjusted with the configuration file **IPD.h <Configuration Wizard>** .

Automatic cycle test of initial position this function can be used automatically when the motor is stationary. Continuous cycle of output IPD, used to do IPD functional accuracy. Can pull this parameter of IPDPattern to UART to observe. The pattern of the fixed position is the same at different angles of the motor rotor.

- Configuration file : IPD.h <Configuration Wizard>
- Set IPD LEVEL

Parameter	Value range/option	Description	Default value	Unit
I_SHORT	0.15V/0.2V/0.25V 0.3V/0.35V/0.4V 0.45V/0.5V	IPD OCP LEVEL	0.15V	--
AOCPEN	Disable/Enable	Analog OCP function	Enable	--
DOCPEN	Disable/Enable	Digital OCP function	Disable	--
IPD Path Select	IPD Current Compare from AOC Path IPD Current Compare from OPA Path	IPD OCP judgment path	AOC Path	--

■ Set IPD IAECYC

Parameter	Value range/option	Description	Default value	Unit
IAECYC	48MHz/24MHz/12MHz/6MHz	IPD Counter Frequency	24MHz	--

Suggestions for adjusting direction :

The sound produced by IPD is proportional to I_SHORT. If it is set to 0.15V, the sound is still loud, and the resistance of Shunt R can only be increased (pay attention to the wattage of the resistance).

AOCPEN、DOCPEN、IPD Path Select : not to change recommended. **IAECYC** : This parameter is an important parameter of IPD. Used to count the OCP dt value, the IPD performed by the steering at different positions, take the minimum Counter as the Pattern of the position. The maximum value of its Counter is 65535, according to the measured OCP dt width. Know the required IAECYC, the maximum dt width that can be determined by different IAECYC is calculated as follows:

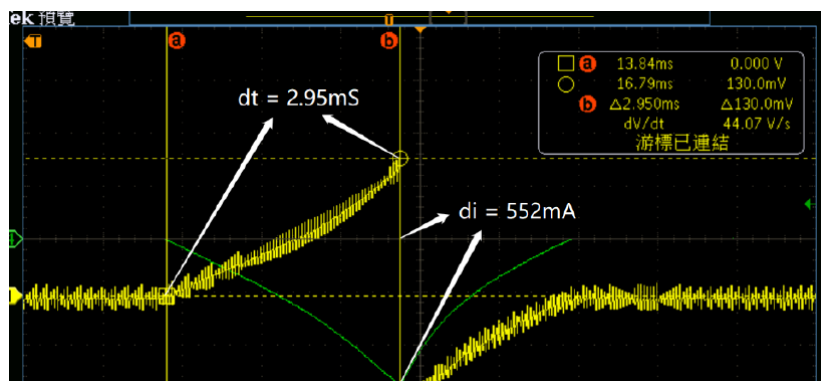
$$48\text{MHz} : (1/48\text{MHz}) * 65535 \approx 1.3\text{ms}$$

$$24\text{MHz} : (1/24\text{MHz}) * 65535 \approx 2.6\text{ms}$$

$$12\text{MHz} : (1/12\text{MHz}) * 65535 \approx 5.2\text{ms}$$

$$6\text{MHz} : (1/6\text{MHz}) * 65535 \approx 10.4\text{ms}$$

The picture on the right as an example, the yellow waveform is the OCP signal, the dt is 2.95ms. Assuming that IAECYC selects 48MHz or 24MHz at this time, because dt is higher than the maximum value of Counter and overflow. You will find that the motor will still reverse when the motor is started, so only dt is only suitable for choosing 12MHz.



✓ Brake control Enable/Disable(✓)

Parameter	Value range/option	Description	Default value	Unit
Set braking force	0~100	Brake Duty	99	%

Suggestions for adjusting direction :

Braking force, it is used by Diode structure or internal estimator against the wind, not used for the time being.

■ Configuration file : Ipwm.h <Configuration Wizard>

■ IPWM Enable/Disable(✓)

Parameter	Value range/option	Description	Default value	Unit
IPWM Frequency SET	--	IPWM frequency	4000	Hz
IPWM MODE SET	Force Low/Force High/Active Low/Active High	IPWM mode	Active High	--

● Set buzzer working time

Parameter	Value range/option	Description	Default value	Unit
IPWM Times_Init	1~255	IPWM execution times	1	times
IPWM Enable time_Init	1~10000	IPWM execution times	100	ms
IPWM Disable time_Init	1~10000	IPWM interval	50	ms

It by analogy...

Suggestions for adjusting direction :

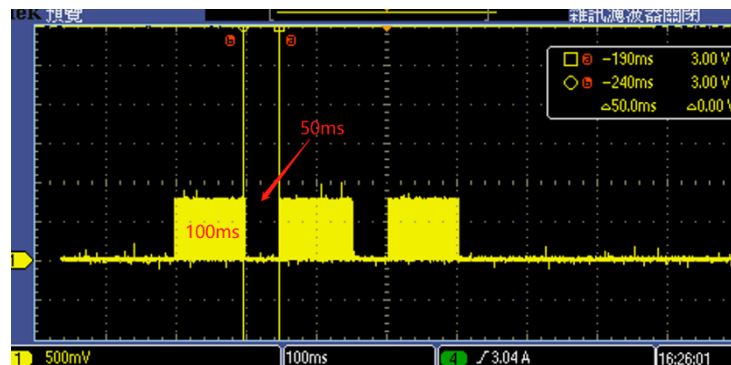
This function needs to be used with Buzzer and connected to MCU internal IPWM pin. The Buzzer operating frequency fill in IPWM Frequency.

IPWM MODE : If the Buzzer is externally excited (passive) , recommended setting is Active High .

If the self-excited (active) Buzzer , recommended setting is Force High .

Buzzer working underneath is divided into power on setting, power off setting, error setting . The number and duration of buzzer chirps can be adjusted according to the customer's needs. In this case, the number of times represented is IPWM Times. Chirping time is Enable time . Close time is Disable time.

For example , Times=3; Enable time = 100ms ; Disable time = 50ms . The IPWM waveform is as follows.



3. Program structure

3.1 Program flow

■ Configuration file : main.c

■ Main Function

- System initialization settings
- while(1){
 - ✓ RfDecode_Read(); // RF decode_read
 - ✓ IRDecode_Read(); // IR decode_read
 - ✓ EEP_Storage(); // EEPROM storage
 - ✓ Judgment of zero-crossing point of idle start voltage and estimation of speed
 - ✓ UART output data
- }

■ Configuration file : Interrupt.c

■ Timer 0 ISR(Time Based : 1ms)

- Correction_Current_AD_offset(); // Power control - V-bus current A/D initial calibration
- Motor_Control(); // Motor error judgment, forward and reverse control, motor start
- Buzzer_Fun(); //Buzzer.

■ Timer 1 ISR(Time Based : 10ms)

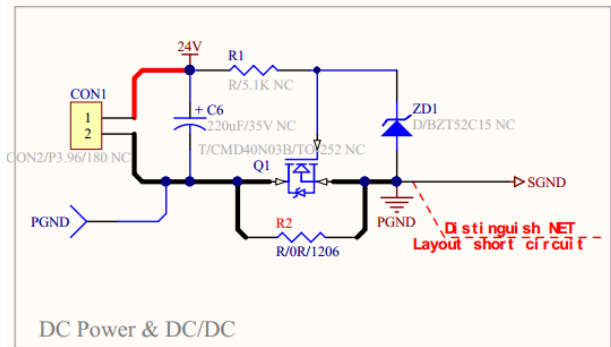
- Vsp_Fun(); //External VSP command reception
- FaultLock_Fun (); //System stall, over/low speed protection
- Vbus_Protect_Fun (Vbus_avg); //V-bus voltage detection and protection judgment
- Temperture_Protect_Fun (Temperture_avg); //System over temperature protection
- Phase_OCP_Protect_Fun() //System phase current protection
- LACK_OF_PHASE_Function(); // Motor still deficiency judgment
- MotorLackPhase_Run_Fun(); // Motor operation deficiency judgment

■ PWM MAX ISR(Time Based : Fpwm)

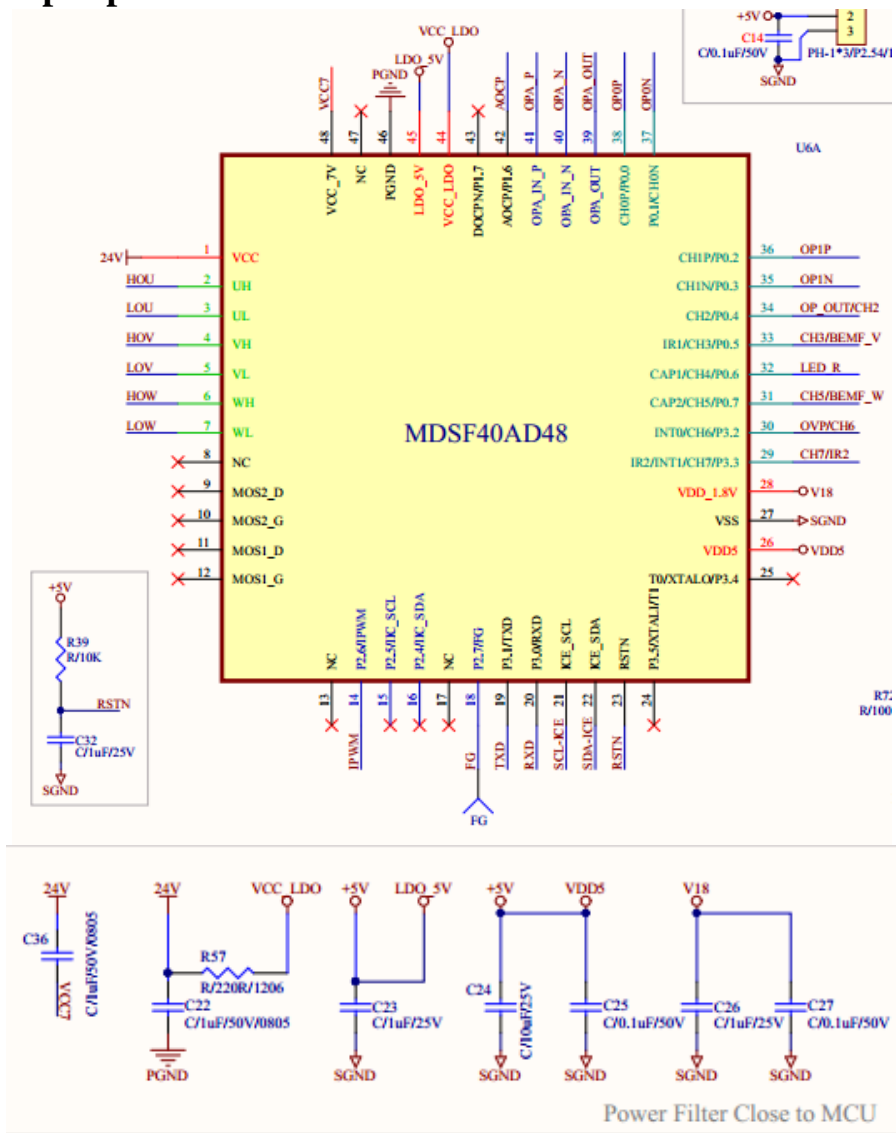
- Judgment of zero-crossing point of idle start voltage and estimation of speed , flag is on

4. Reference circuit design

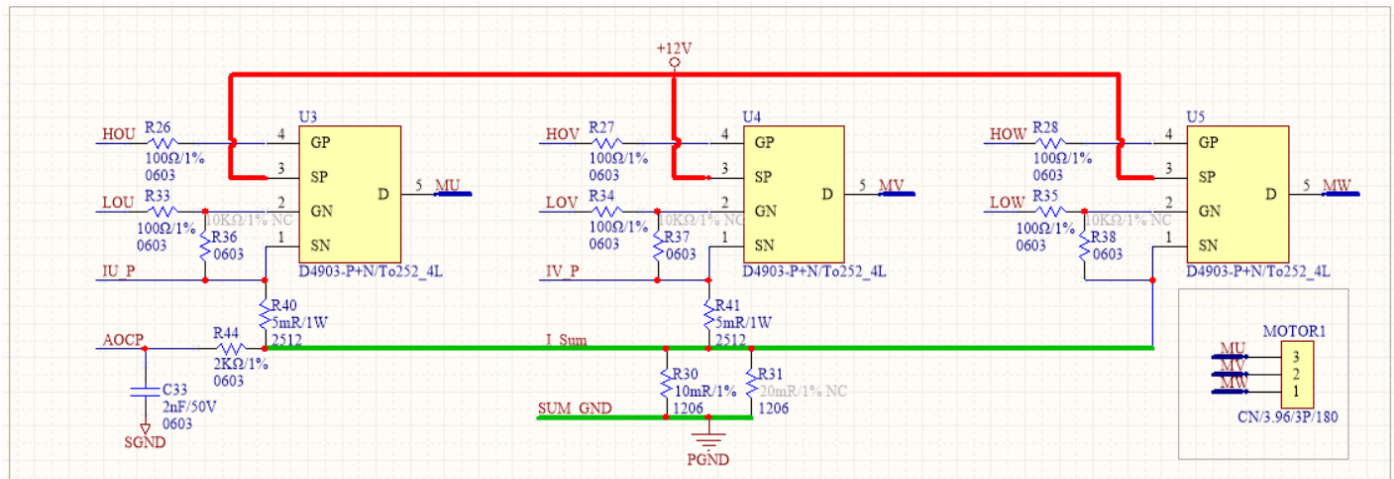
4.1 Power input circuit



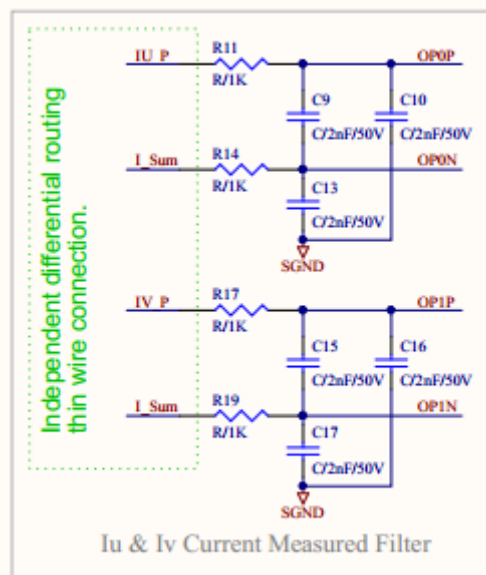
4.2 Core unit and peripheral circuits



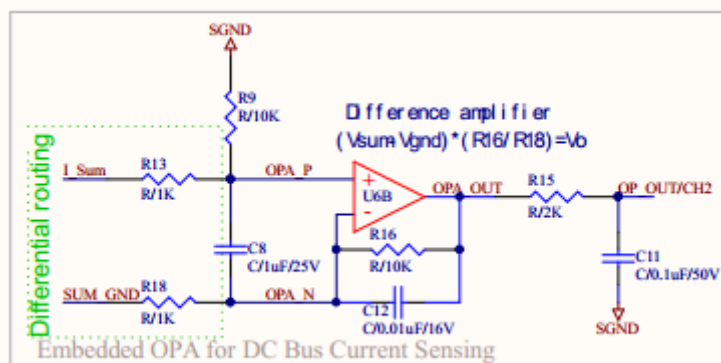
4.3 Three-phase full-bridge inverter



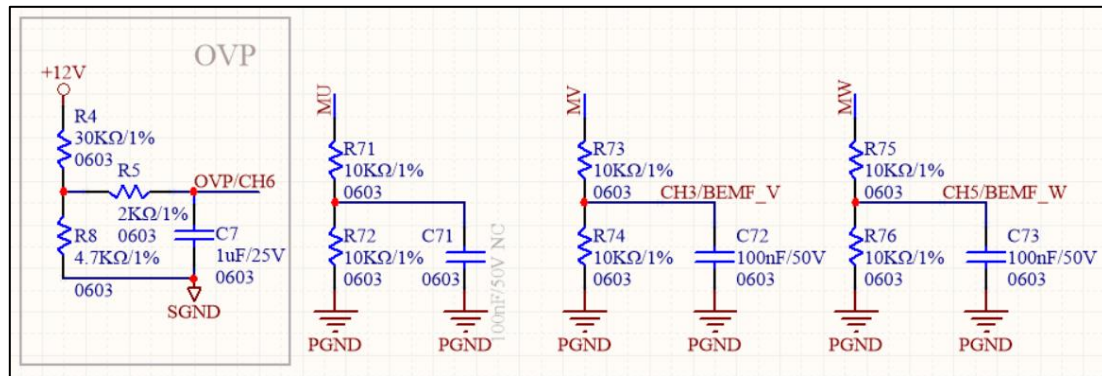
4.4 Two-phase sampling circuit



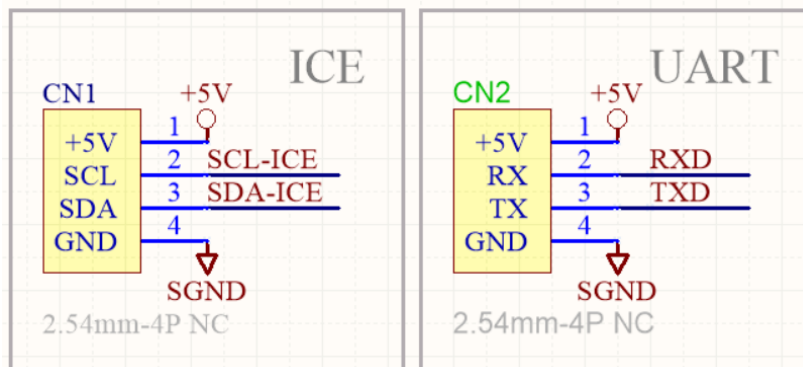
4.5 Bus current sampling circuit



4.6 BEMF feedback circuit



4.7 External port



4.8 Other peripherals

