Application Note
BLDC Driver for Ceiling Fan

Date: May 27, 2025

Version: V1.0.1

Version	Date	Modifier	Remarks
1.0.0	May 13, 2025	Xu Guiqi	Document editing
1.0.1	May 27, 2025	Xu Guiqi	1, Add function expansion 2, Add remote control programming

Catalogue

1. Summary

This application note is designed to guide users through the debugging and simulation of motor control systems, as well as the selection of operating modes and parameter optimization. The debugging section covers the entire process, including establishing communication connections via upper-level software, reading key parameters, and fault diagnosis.

Operating modes can be selected as manual configuration or automatic recognition. With automatic recognition, there are three options: single recognition, initial power-on recognition, and start-up recognition. Users can switch flexibly based on their specific application scenarios.

Parameter settings should be adjusted according to the motor's physical characteristics and the actual application scenario, such as motor voltage protection, maximum power limit, preset positioning and starting current.

2. Debug Environment

2.1 Engineering environment

2.1.1 System wiring

As shown in Figure 2-1, the three-phase power supply lines U, V and W can be connected to the three phase lines of the fan at will (without strictly distinguishing the order). If the rotation direction of the fan is not consistent with the direction marked on the shell, just swap the connection position of any two phase lines to achieve forward rotation.

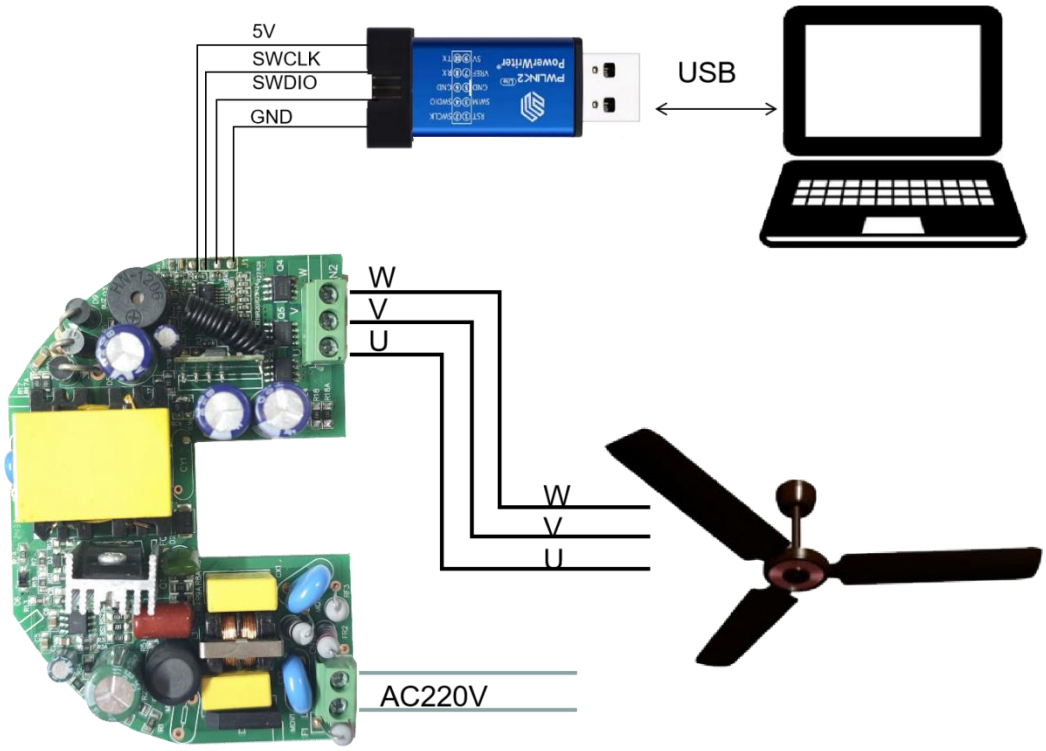


Fig. 2-1

2.1.2 Compilation environment setting

Step 1: Install the compiled software according to the file “MDK Installation Tutorial”.

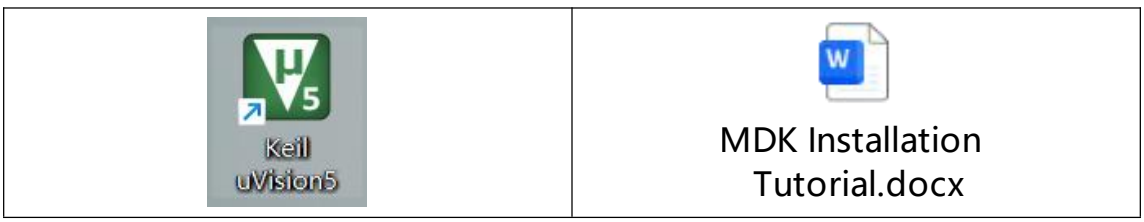


Fig. 2-2

Step 2: Open project XXX (Projects Name) from “Projects” file.

Fig. 2-3

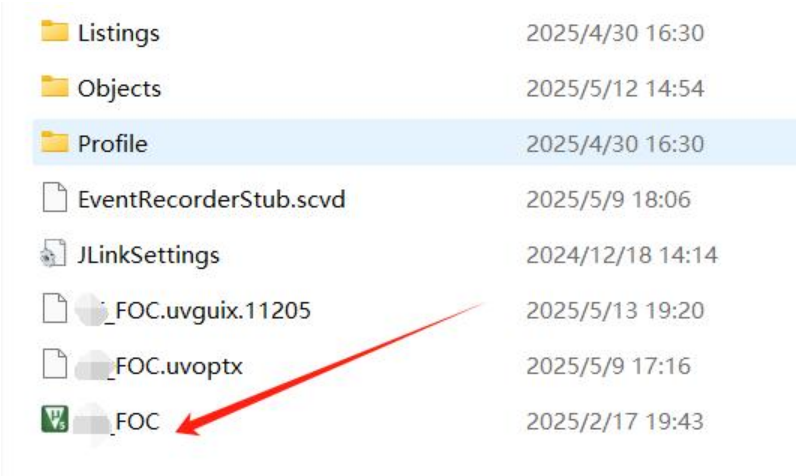


Fig. 2-4

Step 3: Compile “Build” or rebuild “Rebuild” the project, and the system prompts error 0 and warning 0.

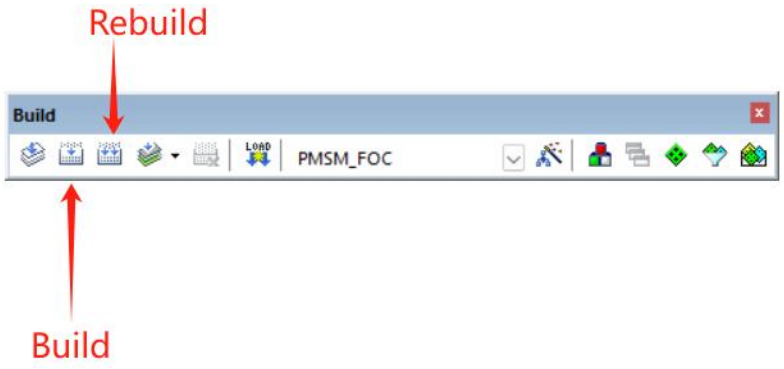


Fig. 2-5

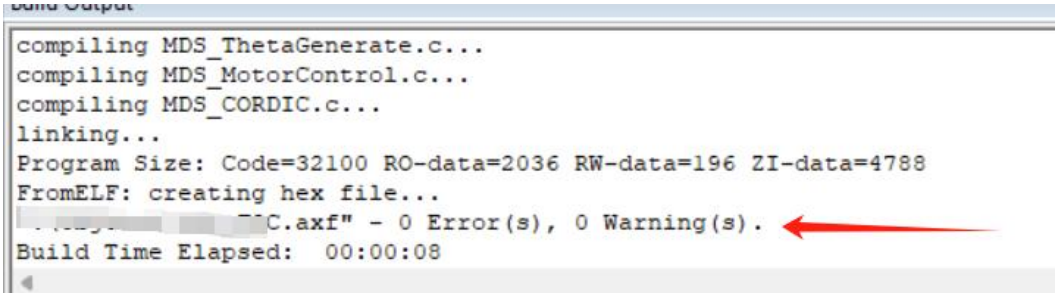


Fig. 2-6

Step 4: Configuration options

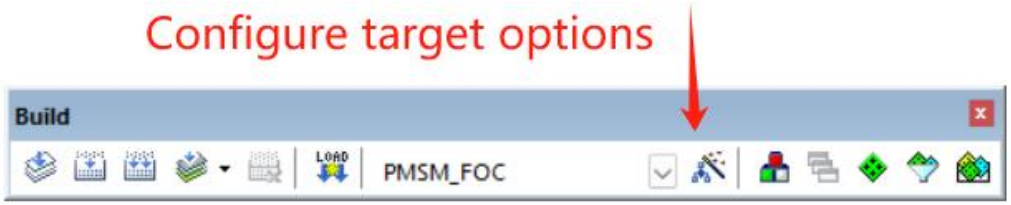


Fig. 2-7

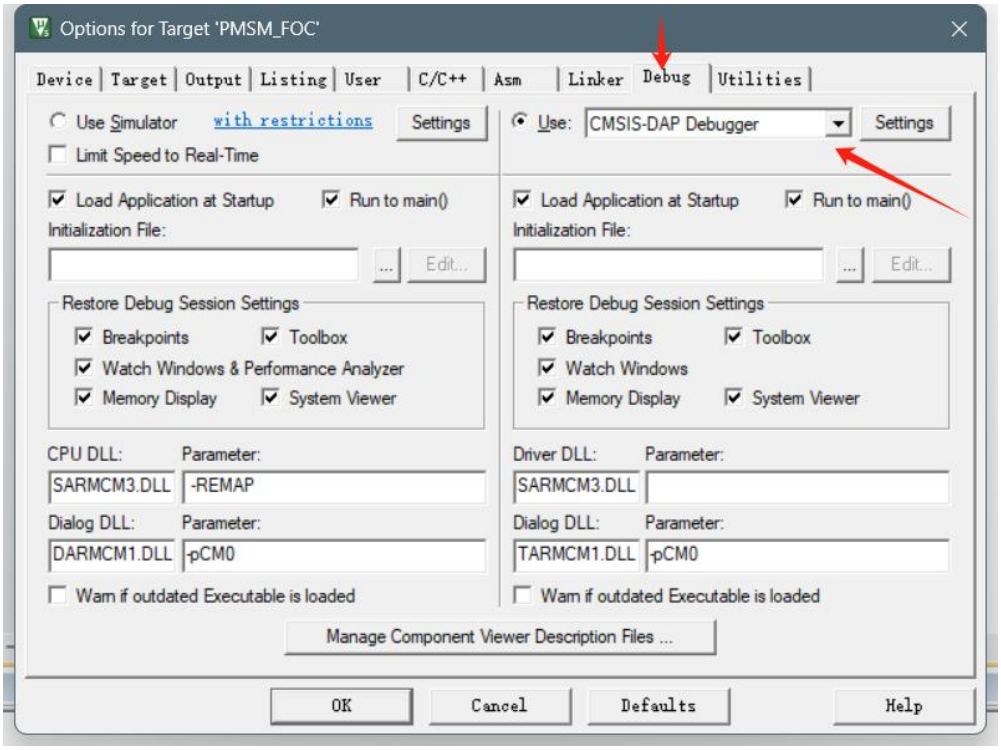


Fig. 2-8

Step 5: Simulation debugging

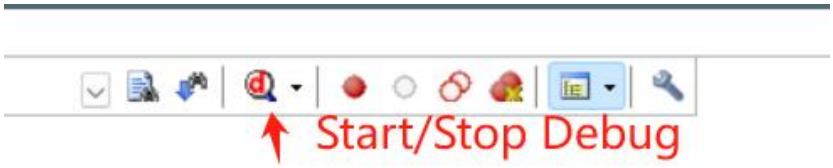


Fig. 2-9

Fig. 2-10

2.2 Remote controller

2.2.1 Remote controller

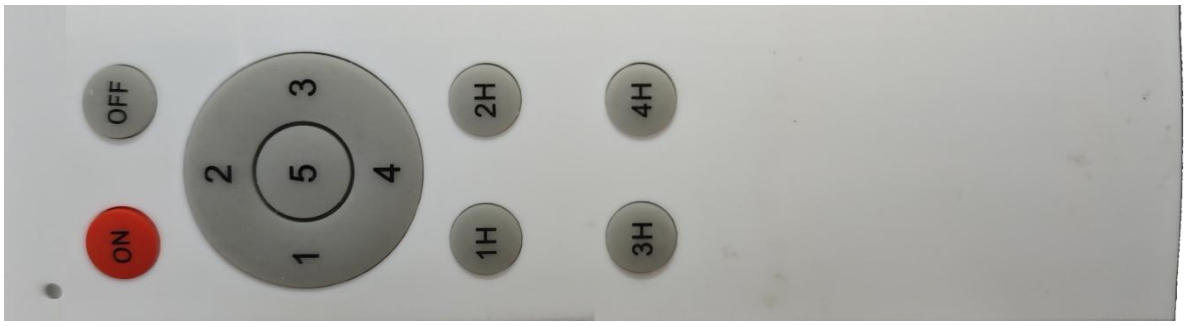


Fig. 2-11

2.2.2 Functions description of remote control button

KEY-ON	Start the fan
KEY-OFF	Turn off the fan
KEY-1	Wind speed 1
KEY-2	Wind speed 2
KEY-3	Wind speed 3
KEY-4	Wind speed 4
KEY-5	Wind speed 5
KEY-1H	Turn off the fan one hour later, and KEY-OFF cancels the timing setting
KEY-2H	Turn off the fan two hours later, and KEY-OFF cancels the timing setting
KEY-3H	Turn off the fan 3 hours later,, and KEY-OFF cancels the timing setting
KEY-4H	Turn off the fan 4 hours later,, and KEY-OFF cancels the timing setting

2.2.3 Code operation

When the AC220V is turned on, the timer starts. Within 10 seconds, the user performs the code matching operation. If the system does not recognize the remote control in the user's hand, the user can press the keys in the following order: KEY-ON →KEY-OFF→KEY-1 to perform the code matching operation. Upon successful code matching, the buzzer will sound 'Di' twice. After successfully matching the codes, the remote control can be used to control the motor operation.

3. Parameters Configure Mode

3.1 Manual configure mode

In the manual configuration mode, users need to set "MOTOR_PARAMETER_IDENT" to "UPDS_DISABLE" in the file of "User_ParaSetDefineTemp.h". The key parameters of the motor need to be set by users themselves, such as phase resistance RS, D/Q axis inductance, amplitude of permanent magnet flux and number of poles.

User_ParaSetDefineTemp.h

```
75  //
76
77  //电机参数辨识
78  //Motor parameter identification
79  #define MOTOR_PARAMETER_IDENT           (UPDS_DISABLE)//Motor parameter identification
80
```

Fig. 3-1

Users can get the correct and all the necessary parameters from the "Design Sheet for Motor Parameters_A01.xlsx" by filling the tested basic parameters of the motor.

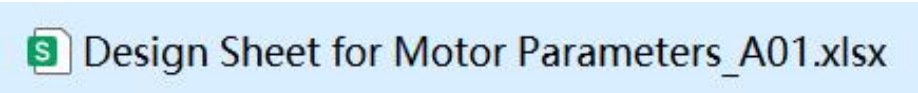


Fig. 3-2

3.1.1 Measure the resistance (UPDS_RS_REAL)

Phase resistance Rs: Measure the two-phase line resistance R of the motor under the DCR range of the bridge.

3.1.2 Measure the axial inductance (UPDS_LD_REAL/UPDS_LQ_REAL)

D/Q axis inductance Ld, Lq: the inductance L of two phase lines measured by the bridge at 1kHz frequency. Slowly rotate the rotor and record the maximum and minimum values of inductance which are Lmax and Lmin respectively.

For SPMSM, it is generally not necessary to rotate the rotor; the measured inductance value L is directly recorded, then Ld = Lq = L/2.

If it is an embedded permanent magnet synchronous motor (IPMSM), then Ld = Lmin/2 and Lq = Lmax/2.

3.1.3 Measure the magnetic flux (UPDS_PSI_REAL)

The oscilloscope probes are clamped to any one of the motor phase lines, and the earth clamp is clamped to any of the rest of the other two phase lines. By rotating the motor, the back EMF waveform can be obtained. As shown in Figure 3-3, in one cycle, the peak to peak value Vpp of the back EMF waveform is 9.76V, and the frequency F is 827.8Hz.

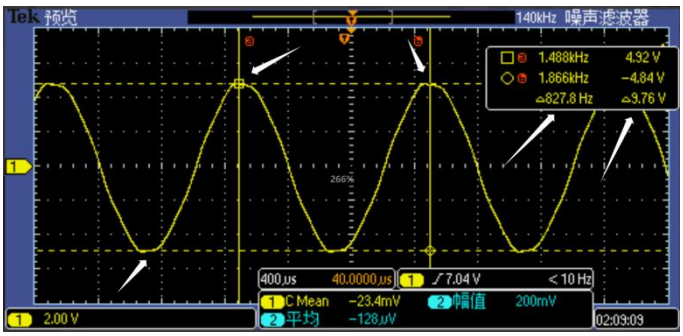


Fig. 3-3

3.1.4 Measure the motor logarithm of poles (UPDS_PAIRS)

Motor logarithm of poles: in a three-phase AC motor, each coil group will produce N and S poles, and each phase contains the number of poles is the logarithm of poles which can be tested like following: Step 1: connect any two of the three phase lines of the motor to a DC power supply (that is the positive and negative terminals of the power supply are limited to about 0.5A, and the DC voltage is recommended to gradually increase from 0); Step 2: turn on the power supply and slowly rotate the motor by hand (if it does not turn, reduce the limiting current), feel the impact produced; Step 2: the number of times the impact is generated after one whole rotation is the motor logarithm of poles.

3.1.5 Parameter update

The parameters calculated by the “Design Sheet for Motor Parameters_A01.xlsx” can be written into the file of “User_ParaSetDefine.h” as shown in Figure 3-4.

User_ParaSetDefine.h

```
58
59 //==================电机参数设置=====================
60 //==================Motor Parameter Setting=====================
61 #define UPDS_RS_REAL                    (1.276000f)    // 定子相电阻，欧姆 // Stator phase resistance, Ohm
62 #define UPDS_LD_REAL                    (0.003079f)    // d轴电感，H// d-axis inductance, H
63 #define UPDS_LQ_REAL                    (0.003079f)    // q轴电感，H// q-axis inductance, H
64 #define UPDS_PSI_REAL                   (0.037320f)    // 永磁体磁链幅值(V)/(rad/s)// Permanent magnet flux linkage (
65 #define UPDS_PAIRS                      (6u)            // 极对数// Number of pole pairs
66
```

Fig. 3-4

3.2 Automatic configure mode

When the user enables the automatic configure mode, he only needs to fill in the logarithmic parameters, and the system will automatically identify the key parameters such as motor resistance, inductance and flux within the setting time. The automatic configure mode simplifies the debugging process compared with manual configure mode, reduces the development cost and shortens the development cycle for users.

3.2.1 Enable motor parameter identification

In the automatic configuration mode, users need to set "MOTOR_PARAMETER_IDENT" to "UPDS_ENABLE" in the file of "User_ParaSetDefineTemp.h".

And then for "MOTOR_PARAMETER_IDENT_TIMES" configuration, customers can fill in the corresponding parameters to select different application scenarios.

Every startup includes started from remote controller identifies the parameters like "MOTOR_PARAMETER_IDENT_EVERY_Time";

Every startup from AC power on identifies the parameters like "MOTOR_PARAMETER_IDENT_FIRST_POWER_ON";

Only once for ever after startup from AC power on identifies the parameters like "MOTOR_PARAMETER_IDENT_ONCE".

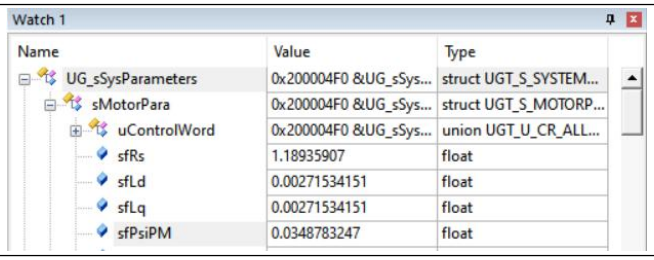
User_ParaSetDefineTemp.h

```
75  //
76
77  //电机参数辨识
78  //Motor parameter identification
79  #define MOTOR_PARAMETER_IDENT           (UPDS_ENABLE)//Motor parameter identification
80
81  #if (MOTOR_PARAMETER_IDENT)
82  #define MOTOR_PARAMETER_IDENT_EVERY_Time      0//每次启动都识别参数 // Identify parameters every time on startup
83  #define MOTOR_PARAMETER_IDENT_FIRST_POWER_ON  1//初次上电启动识别参数// Identify parameters on the first power-on
84  #define MOTOR_PARAMETER_IDENT_ONCE            2//只读一次参数 // Identify parameters only once
85  #define MOTOR_PARAMETER_IDENT_TIMES           (MOTOR_PARAMETER_IDENT_EVERY_Time)
86  #define UPDS_RS_IDENTIFY_EN_SEL         (UPDS_ENABLE)
87  #define UPDS_LS_IDENTIFY_EN_SEL         (UPDS_ENABLE)
88  #define UPDS_PHI_IDENTIFY_EN_SEL        (UPDS_ENABLE)
```

Fig. 3-5

3.2.2 Key parameter reading

In Debug mode, the user can view the automatically recognized motor parameters in the Watch1 observation window. The recorded key parameters can be used for manual configuration mode.



SfRs:Stator phase resistance
SfLd: D-axis inductance
SfLq: Q-axis inductance
sfPsiPM: Permanent magnet flux

Fig. 3-6

3.2.3 Fan speed reading (u32FanSpeed)

In debug mode, users can view the current fan speed in the observation window Watch1, which greatly facilitates the user to test the fan function when external test tools are lacking.

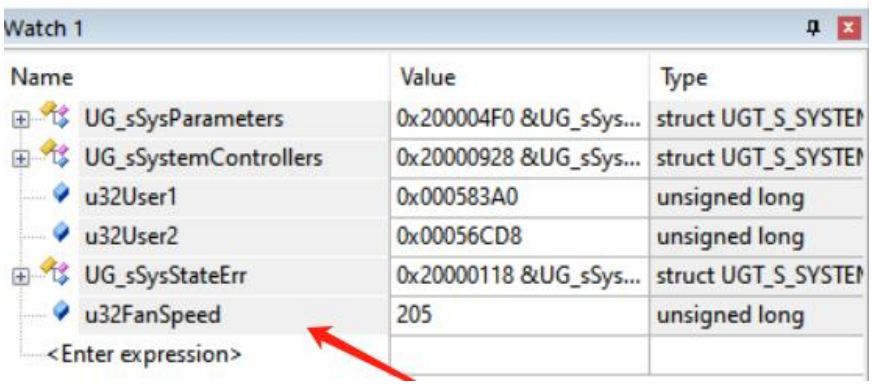


Fig. 3-7

4. Control Mode (UPDS_CONTROL_MODE)

There are two control modes: 1) Constant Speed Mode “UPDS_SPEED_MODE” and Constant Power Mode “UPDS_POWER_MODE”. In the Constant Speed Mode, if the power does not exceed the limit, each gear maintains a constant speed and the power may fluctuate with the load. If the power exceeds the limit, the speed is restricted to keep the power at its maximum point. In the Constant Power Mode, the output power of each gear remains constant, and the motor power is set to its maximum speed within the gear's power limit, ensuring that the maximum speed is less than or equal to the rated speed.

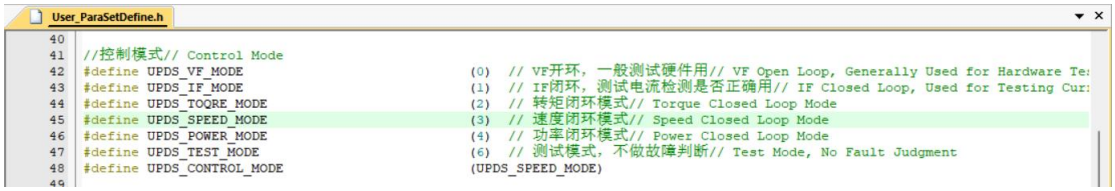


Fig. 4-1

4.1 Rated speed (UPDS_RATED_SPEED)

“UPDS_RATED_SPEED” can set the maximum rated speed of the motor.

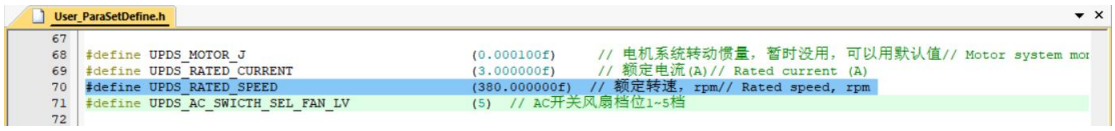


Fig. 4-2

4.2 Remote control setting under constant speed mode

Speed commands for each gear are set in the form of the rated speed's nominal value. For instance, consider the expression “_IQ (200/UPDS RATED SPEED)” for gear 1: the numerator 200 represents the user-defined speed of gear 1 (unit: rpm), while the denominator “UPDS RATED SPEED” is the motor's rated speed (380 rpm in this example). The result is 0.526, corresponds to 52.6% of the rated speed.

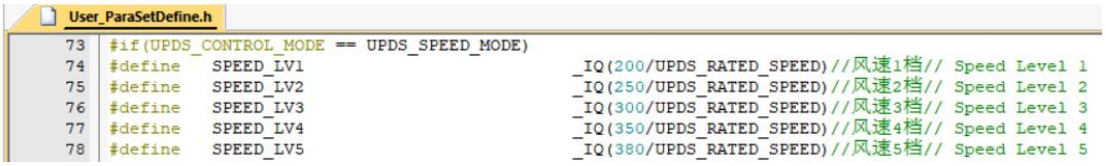


Fig. 4-3

4.3 Maximum output power (UPDS_POWER_MAX)

The system estimates the current output power through sampling parameters and the relationship between input power and output power:

Input power =UPDS_POWER_MAX*1.2

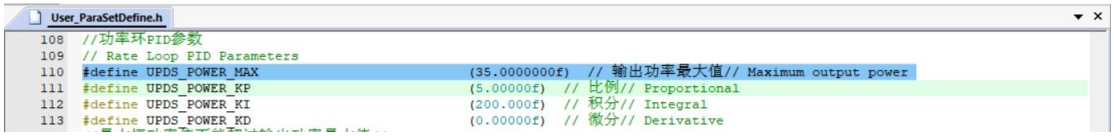


Fig. 4-4

4.4 Remote control setting under constant power mode

The power command of each gear is set in the form of the maximum output power nominal value.

For example, the expression _IQ(15/UG_sSysParameters.sMotorPara.sfPowerB) of 1 gear: the numerator 15 represents the output power set by the user for 1 gear, and the denominator “UG_sSysParameters.sMotorPara.sfPowerB” is the maximum output power “UPDS_POWER_MAX” like 35W. The calculation result is 0.428, which corresponds to 42.8% of the maximum output power.

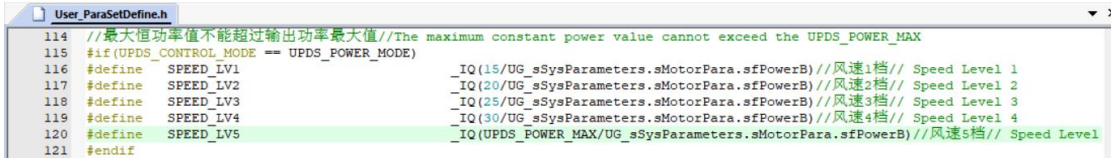


Fig. 4-5

4.5 Application scenario setting

Users may encounter a variety of application scenarios in the process of use. The program sets some possible application scenarios for customers to choose. For example, whether to start the fan when the mains is connected; how to start the fan after the remote control is lost and what speed the fan runs.

4.5.1 Start the fan after power on

This parameter is mainly used for scenarios where the fan needs to be powered by mains electricity to start running. The user only needs to enable this parameter and the fan will run after mains electricity is on. Configure “UPDS_ENABLE” as

shown in Fig. 4-6 to “UPDS_POWER_ON_OPERATE_THE_MOTOR” in “User_ParaSetDefineTemp.h”.

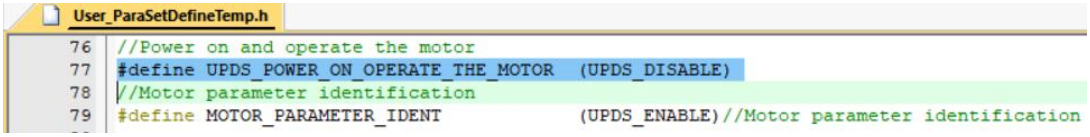


Fig. 4-6

4.5.2 Remote control is lost

If the user loses the remote control, just switch the mains power on and off three times, and the system will set the state as the remote control is lost. In this state, the fan starts when the mains power is on and runs at the preset position. If the remote control command is received again, the system will automatically reset the lost state.

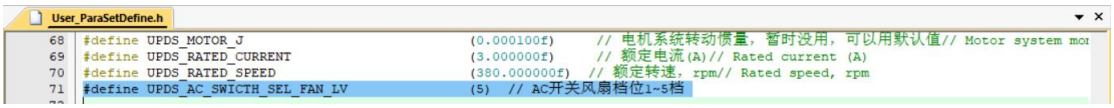


Fig. 4-7

5. Parameter Optimization

In the actual application process, users may encounter a variety of motors. Adjusting the corresponding parameter configuration according to different application scenarios can optimize the using experience, such as large or small blades, motor start vibration, motor reverse start, etc.

5.1 Startup current

The configuration of startup current in the program we offered can match most of the blades. Customer can also optimize this parameter to do some performance balance by adjusting the parameter in "UPDS_INJECT_D_CURRENT_A_PU" larger or smaller.

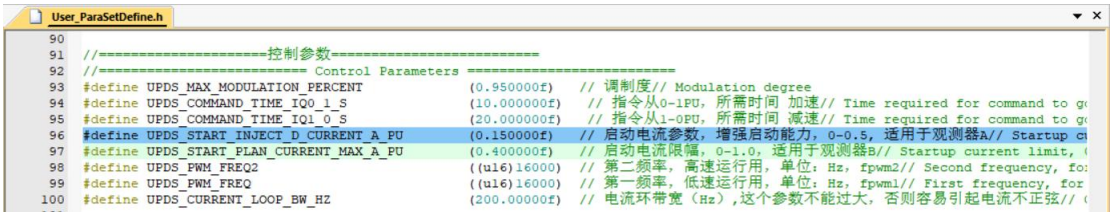


Fig. 5-1

5.2 Command time

"UPDS_COMMAND_TIME_IQ0_1_S" is the time required from 0-1PU. The smaller the value, the faster the acceleration; the larger the value, the slower the acceleration. If there is a jitter situation, this parameter can be appropriately increased or decreased.

Fig. 5-2

5.3 Speed and time for judging the failure of startup

When the load is too heavy and the start fails, the judgment speed can be appropriately reduced and the judgment time can be increased.

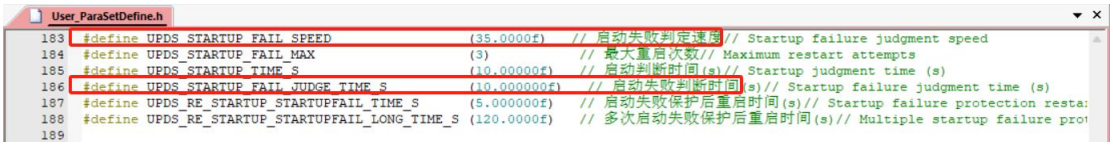


Fig. 5-3

5.4 Open loop I/F starting current

The I/F starting current can be increased or decreased appropriately according to the loading.

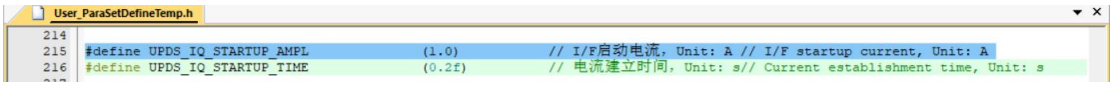


Fig. 5-4

5.5 Pre-positioning

Enabling the pre-positioning function can improve the reversal rotate phenomenon.

Fig. 5-5

6. Protection Mechanism

The protection functions of this scheme include start blockage protection, over current protection (hardware over current and software over current), current limiting protection, phase loss protection, over voltage protection and under voltage protection. “US_vSysErrorCheck()” is the protection execution function which is in “User_Interrupt.c”.

The protection function is performed in the Systick interrupt, which has a lower priority than the DMA interrupt. After successful judgment on the protections (except for current limiting protection), system will enter the vFault state machine for processing. First protection action will be PWM output turned off, then set up a wait for restart or direct shutdown as required.

User_Interrupt.c

```
247   if (UG_sSysStateErr.uControlReg.Bits.ControlMode != UGT_E_CM_TEST)
248   {
249     if ((UG_sSysStateErr.eSystemState < UGT_E_SS_WAIT) && (UG_sSysStateErr.eSystemState >= UGT_E_SS_START))
250     {
251       US_vSysErrorCheck();
252     }
253   }
254 }
```

Fig. 6-1

6.1 Over voltage and under voltage protection

In the running state, if the voltage is lower than the set under voltage or over voltage limits, it will enter the vFault state after judgment.

Shut down, enter the vWait state, wait for a certain time to enter the initialization vInit state, ready to restart.

User_ParaSetDefine.h

```
159  #define UPDS_REOVERVOLTAGE_V                (30.00000f)   // 母线过压保护恢复值(V) // Bus overvoltage protection recovery
160  #define UPDS_OVERVOLTAGE_JUDGE_TIME_MS      (10.00000f)   // 母线过压判断时间(ms)// Bus overvoltage judgment time (ms)
161  #define UPDS_RE_STARTUP_OVERVOLT_TIME_S     (1.000000f)   // 母线过压保护后重启时间(s)// Bus overvoltage protection resta
162
163  #define UPDS_UNDERVOLTAGE_V                 (18.00000f)   // 母线欠压保护值(V)// Bus undervoltage protection value (V)
164  #define UPDS_REUNDERVOLTAGE_V               (19.0000f)   // 母线欠压保护恢复值(V)// Bus undervoltage protection recovery
165  #define UPDS_UNDERVOLTAGE_JUDGE_TIME_MS     (10.00000f)   // 母线欠压判断时间(ms)// Bus undervoltage judgment time (ms)
166  #define UPDS_RE_STARTUP_UNDERVOLT_TIME_S    (1.000000f)   // 母线欠压保护后重启时间(s)// Bus undervoltage protection resta
```

Fig. 6-2

6.2 Over current protection

6.2.1 Hardware over current protection

Set the comparator interrupt service subroutine with the highest priority. The PWM braking function is triggered by an event from the MCU's internal comparator

(ACMP0/1). The principle of operation is as follows: the bus current (I_bus) flows through a sampling resistor (R), generating a voltage V_bus. This voltage is then amplified by the operational amplifier (OPA0/1/2) and enters the positive terminal of the comparator (or directly to the positive terminal). The negative terminal of the comparator is connected to the output of the DAC0/1, which can be configured based on the reference voltage Vref. When the bus current exceeds a certain threshold, the positive terminal voltage of the comparator becomes higher than the negative terminal voltage, triggering an interrupt in the comparator. This interrupt activates the PWM braking function, thereby providing over current protection. By continuously monitoring the three-phase current of the motor, if the maximum current value exceeds the set protection threshold, the system triggers hardware over current protection.

6.2.2 Software over current protection

By detecting the three-phase current of the motor in real time, the software is triggered when the maximum value of the current exceeds the set protection value.

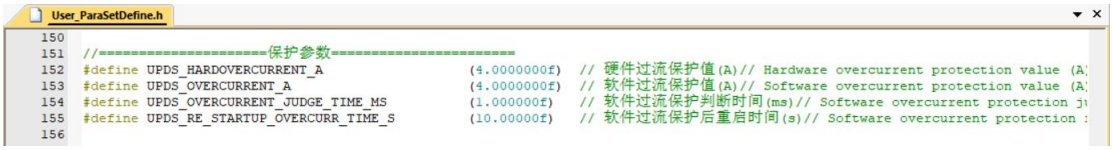


Fig. 6-3

6.3 Phase loss protection

Phase deficiency protection refers to the condition in which, during the operation of a motor, one or two phases of the three-phase power are missing, and the current value of that phase should be zero, the phase current of a certain phase is always 0 within a period of detection (the time to trigger the phase loss protection) and the phase loss protection mechanism is triggered.

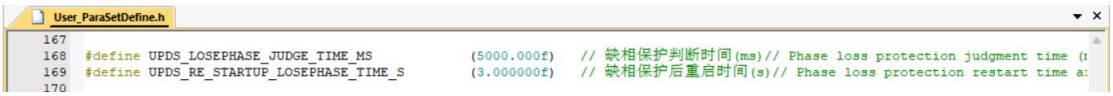


Fig. 6-4

6.4 Protection against blockage and failure to start

Detect whether the motor's operating speed is below the preset protection threshold, and trigger the stall protection mechanism. Set the stall duration as needed. Once the stall duration is reached, first turn off the PWM, then wait for a reboot period before restarting. If the maximum number of stalls is exceeded, the system must be powered off and then re-powered.

```
#define UPDS_STARTUP_FAIL_SPEED                 (35.0000f)   // 启动失败判定速度// Startup failure judgment speed
#define UPDS_STARTUP_FAIL_MAX                   (3)           // 最大重启次数// Maximum restart attempts
#define UPDS_STARTUP_TIME_S                     (10.00000f)   // 启动判断时间(s)// Startup judgment time (s)
#define UPDS_STARTUP_FAIL_JUDGE_TIME_S          (10.000000f)   // 启动失败判断时间(s)// Startup failure judgment time (s)
#define UPDS_RE_STARTUP_STARTUPFAIL_TIME_S      (5.000000f)   // 启动失败保护后重启时间(s)// Startup failure protection resta
#define UPDS_RE_STARTUP_STARTUPFAIL_LONG_TIME_S (120.0000f)   // 多次启动失败保护后重启时间(s)// Multiple startup failure pro

#define UPDS_STALLCURRENT_A_PU                  (0.700000f)   // 堵转电流判断值，标幺 // Stall current judgment value, per un
#define UPDS_STALL_JUDGE_TIME_MS                (5000.000f)   // 堵转判断时间(ms)// Stall judgment time (ms)
#define UPDS_RE_STARTUP_STALL_TIME_S            (1.000000f)   // 堵转保护后重启时间(s)// Stall protection restart time (s)
```

Fig. 6-5

7. Fault Diagnosis

During debugging, the Watch1 can be used to monitor the UG_sSysStateErr parameter in real time to find out the cause of failure, such as over voltage, under voltage, phase loss, and over current.

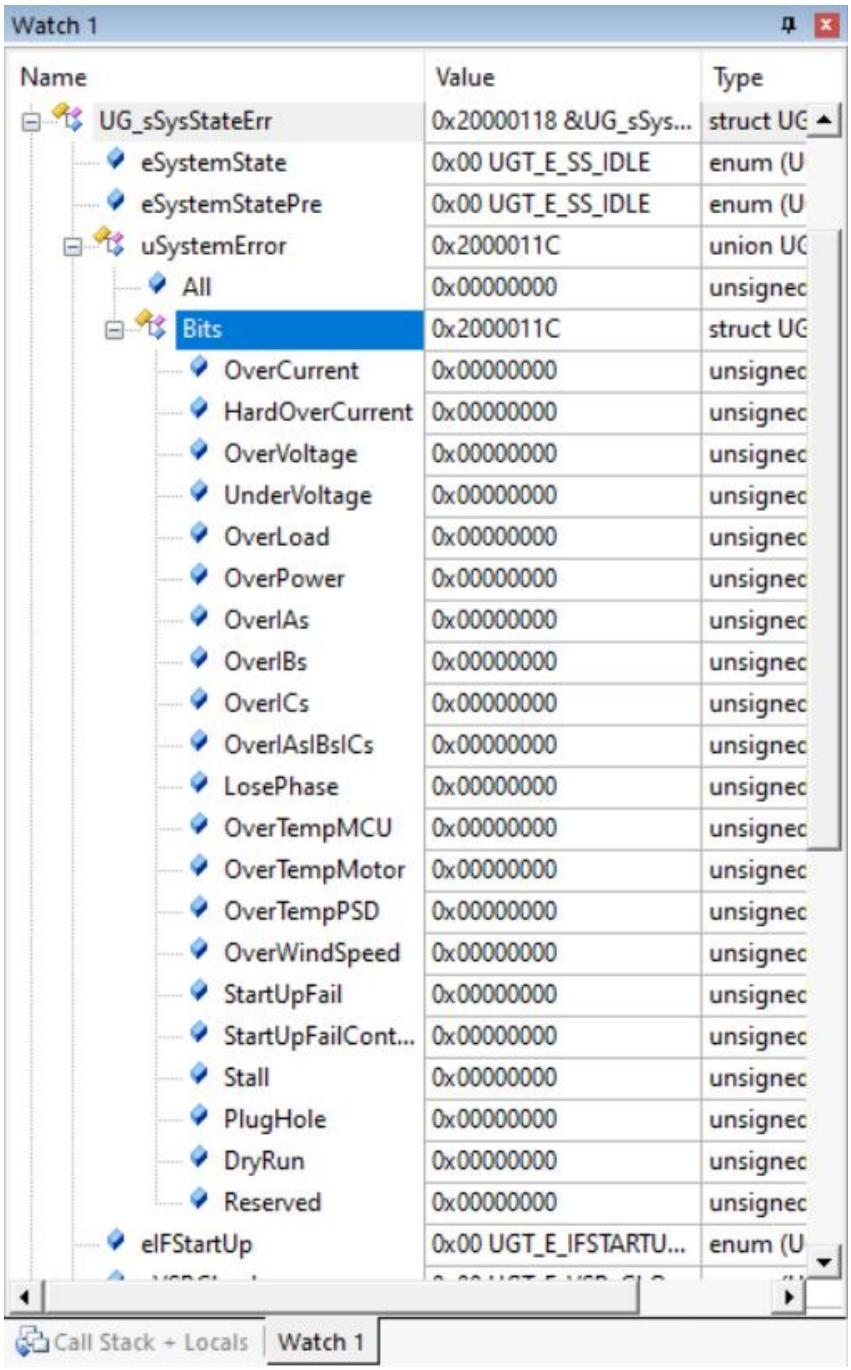


Fig. 7-1

8. Function Expansion

8.1 Increase the fan speed levels

If users need to increase or decrease the speed levels, they only need to add or reduce the definition of the levels on the existing framework.

Note: The key code of each button is unique. If the key codes of two buttons are the same, the compiler will report an error.

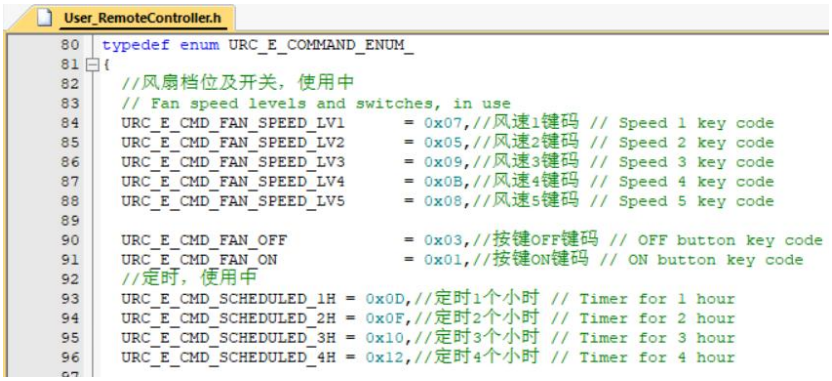


Fig. 8-1

8.1.1 Add wind speed gear button

Assuming that the fan has 1-8 key positions, the key codes of 1-5 are consistent with the original setting, and the key codes of 6-8 are 0x11, 0x12, 0x13, then the following three settings need to be added:

URC_E_CMD_FAN_SPEED_LV6 = 0x11

URC_E_CMD_FAN_SPEED_LV7 = 0x12

URC_E_CMD_FAN_SPEED_LV8 = 0x13

The modified result is shown in figure 8-2.

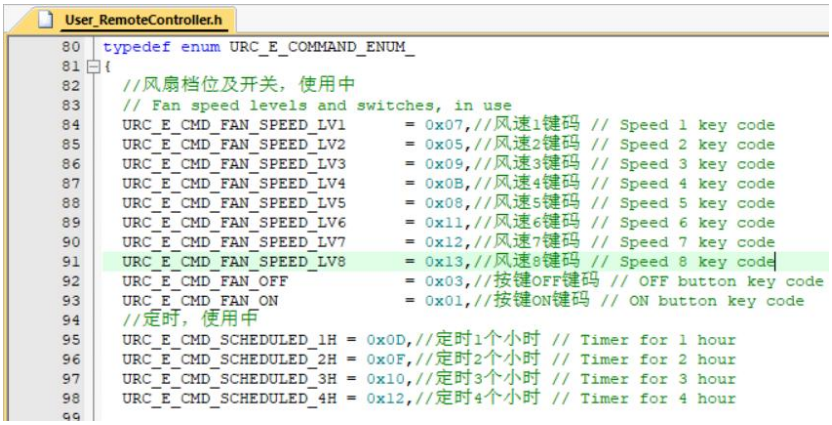


Fig. 8-2

8.1.2 Add wind speed gear function

Add corresponding gear function to “void URC_vRemoteCtrlHandle(void)” as shown in Figure 8-3:

In the figure, “case URC_E_CMD_FAN_SPEED_LVx” is the function of the key, and “sRemoteCtrl.ucSpeedLevel” is the gear. If set sRemoteCtrl.ucSpeedLevel = 6, then the wind speed is set to gear 6.

User_RemoteController.c

```
185  u8 u8FanTimer;
186  void URC_vRemoteCtrlHandle(void)
187  {
188    switch (sRemoteCtrl.uFrame.sFrame.u32Command)
189    {
190      case URC_E_CMD_FAN_SPEED_LV1:
191      {
192        if(sRemoteCtrl.ubFanRun)
193        {
194          sRemoteCtrl.ucSpeedLevel = 1;
195          u8BeepTime = 1;
196        }
197        break;
198      }
199      case URC_E_CMD_FAN_SPEED_LV2:
200      {
201        if(sRemoteCtrl.ubFanRun)
202        {
203          sRemoteCtrl.ucSpeedLevel = 2;
204          u8BeepTime = 1;
205        }
206        break;
207      }
```

Fig. 8-3

8.1.3 Add fan gear switching execution function

P_SPEED_LVx is the forward speed;

N_SPEED_LVx is the reverse speed;

To increase the execution function, you only need to modify the parameter pointed by the red arrow.

MDS_GeneralFunctions.c

```
138         }
139         else if((sRemoteCtrl.ucSpeedLevel == 7) && (sRemoteCtrl.ubFanRun == 1))    //7档
140         {
141           if(ul6FanDirection)
142           {
143             UG_sSystemControllers.slCommand           = P_SPEED_LV7;
144           }
145           else
146           {
147             UG_sSystemControllers.slCommand           = N_SPEED_LV7;
148           }
149           UG_sSysStateErr.sSoftClose.ssSystemSwitch = UPDS_SYSTEM_START + 1;     //启动
150         }
151         else if((sRemoteCtrl.ucSpeedLevel == 8) && (sRemoteCtrl.ubFanRun == 1))    //8档
152         {
153           if(ul6FanDirection)
154           {
155             UG_sSystemControllers.slCommand           = P_SPEED_LV8;
156           }
157           else
158           {
159             UG_sSystemControllers.slCommand           = N_SPEED_LV8;
160           }
161           UG_sSysStateErr.sSoftClose.ssSystemSwitch = UPDS_SYSTEM_START + 1;     //启动
162         }
```

Fig. 8-4

8.1.4 Increase the fan speed or power setting

User_ParaSetDefine.h*

```
73  #if(UPDS_CONTROL_MODE == UPDS_SPEED_MODE)
74  //FORWARD
75  #define   P_SPEED_LV1                     _IQ(200/UPDS_RATED_SPEED)//风速1档// Speed Level 1
76  #define   P_SPEED_LV2                     _IQ(250/UPDS_RATED_SPEED)//风速2档// Speed Level 2
77  #define   P_SPEED_LV3                     _IQ(300/UPDS_RATED_SPEED)//风速3档// Speed Level 3
78  #define   P_SPEED_LV4                     _IQ(350/UPDS_RATED_SPEED)//风速4档// Speed Level 4
79  #define   P_SPEED_LV5                     _IQ(380/UPDS_RATED_SPEED)//风速5档// Speed Level 5
80  #define   P_SPEED_LV6                     _IQ(380/UPDS_RATED_SPEED)//风速6档// Speed Level 6
81  #define   P_SPEED_LV7                     _IQ(380/UPDS_RATED_SPEED)//风速7档// Speed Level 7
82  #define   P_SPEED_LV8                     _IQ(380/UPDS_RATED_SPEED)//风速8档// Speed Level 8
83
84  //REVERSE
85  #define   N_SPEED_LV1                     _IQ(-200/UPDS_RATED_SPEED)//风速1档// Speed Level 1
86  #define   N_SPEED_LV2                     _IQ(-250/UPDS_RATED_SPEED)//风速2档// Speed Level 2
87  #define   N_SPEED_LV3                     _IQ(-300/UPDS_RATED_SPEED)//风速3档// Speed Level 3
88  #define   N_SPEED_LV4                     _IQ(-350/UPDS_RATED_SPEED)//风速4档// Speed Level 4
89  #define   N_SPEED_LV5                     _IQ(-380/UPDS_RATED_SPEED)//风速5档// Speed Level 5
90  #define   N_SPEED_LV6                     _IQ(-380/UPDS_RATED_SPEED)//风速6档// Speed Level 6
91  #define   N_SPEED_LV7                     _IQ(-380/UPDS_RATED_SPEED)//风速7档// Speed Level 7
92  #define   N_SPEED_LV8                     _IQ(-380/UPDS_RATED_SPEED)//风速8档// Speed Level 8
93  #endif
```

Fig. 8-5

User_ParaSetDefine.h

```
127 #define UPDS_POWER_KD                     (0.00000f)  // 微分// Derivative
128 //最大恒功率值不能超过输出功率最大值//The maximum constant power value cannot exceed the UPDS_POWER_MAX
129 #if(UPDS_CONTROL_MODE == UPDS_POWER_MODE)
130 //FORWARD
131 #define   P_SPEED_LV1                     _IQ(15/UG_sSysParameters.sMotorPara.sfPowerB)//风速1档// Speed Level 1
132 #define   P_SPEED_LV2                     _IQ(20/UG_sSysParameters.sMotorPara.sfPowerB)//风速2档// Speed Level 2
133 #define   P_SPEED_LV3                     _IQ(25/UG_sSysParameters.sMotorPara.sfPowerB)//风速3档// Speed Level 3
134 #define   P_SPEED_LV4                     _IQ(30/UG_sSysParameters.sMotorPara.sfPowerB)//风速4档// Speed Level 4
135 #define   P_SPEED_LV5                     _IQ(35/UG_sSysParameters.sMotorPara.sfPowerB)//风速5档// Speed Level 5
136 #define   P_SPEED_LV6                     _IQ(35/UG_sSysParameters.sMotorPara.sfPowerB)//风速6档// Speed Level 6
137 #define   P_SPEED_LV7                     _IQ(35/UG_sSysParameters.sMotorPara.sfPowerB)//风速7档// Speed Level 7
138 #define   P_SPEED_LV8                     _IQ(35/UG_sSysParameters.sMotorPara.sfPowerB)//风速8档// Speed Level 8
139 //REVERSE
140 #define   N_SPEED_LV1                     _IQ(-15/UG_sSysParameters.sMotorPara.sfPowerB)//风速1档// Speed Level 1
141 #define   N_SPEED_LV2                     _IQ(-20/UG_sSysParameters.sMotorPara.sfPowerB)//风速2档// Speed Level 2
142 #define   N_SPEED_LV3                     _IQ(-25/UG_sSysParameters.sMotorPara.sfPowerB)//风速3档// Speed Level 3
143 #define   N_SPEED_LV4                     _IQ(-30/UG_sSysParameters.sMotorPara.sfPowerB)//风速4档// Speed Level 4
144 #define   N_SPEED_LV5                     _IQ(-35/UG_sSysParameters.sMotorPara.sfPowerB)//风速5档// Speed Level 5
145 #define   N_SPEED_LV6                     _IQ(-35/UG_sSysParameters.sMotorPara.sfPowerB)//风速6档// Speed Level 6
146 #define   N_SPEED_LV7                     _IQ(-35/UG_sSysParameters.sMotorPara.sfPowerB)//风速7档// Speed Level 7
147 #define   N_SPEED_LV8                     _IQ(-35/UG_sSysParameters.sMotorPara.sfPowerB)//风速8档// Speed Level 8
148 #endif
```

Fig. 8-6

8.2 Fan forward and reverse rotation

8.2.1 Add forward and reverse keys

Assuming that the key code of the forward button is 0xF8 and the key code of the reverse button is 0xF9, the key setting is shown in Figure 8-7:

User_RemoteController.h

```
typedef enum URC_E_COMMAND_ENUM_
{
  //风扇档位及开关，使用中
  // Fan speed levels and switches, in use
  URC_E_CMD_FAN_SPEED_LV1      = 0x07,//风速1键码 // Speed 1 key code
  URC_E_CMD_FAN_SPEED_LV2      = 0x05,//风速2键码 // Speed 2 key code
  URC_E_CMD_FAN_SPEED_LV3      = 0x09,//风速3键码 // Speed 3 key code
  URC_E_CMD_FAN_SPEED_LV4      = 0x0B,//风速4键码 // Speed 4 key code
  URC_E_CMD_FAN_SPEED_LV5      = 0x08,//风速5键码 // Speed 5 key code
  URC_E_CMD_FAN_SPEED_LV6      = 0x11,//风速6键码 // Speed 6 key code
  URC_E_CMD_FAN_SPEED_LV7      = 0x12,//风速7键码 // Speed 7 key code
  URC_E_CMD_FAN_SPEED_LV8      = 0x13,//风速8键码 // Speed 8 key code
  URC_E_CMD_FAN_OFF            = 0x03,//按键OFF键码 // OFF button key code
  URC_E_CMD_FAN_ON             = 0x01,//按键ON键码 // ON button key code
  URC_E_CMD_FAN_FORWARD        = 0xF8,//按键FORWARD键码 // Forward button key code
  URC_E_CMD_FAN_REVERSE        = 0xF9,//按键REVERSE键码 // Reverse button key code
```

Fig. 8-7

8.2.2 Add forward and reverse button functions

u16FanDirection = TRUE; // fan is running forward

u16FanDirection = FALSE; // fan reversed

User_RemoteController.c

```
LxBool FBREEZER_DISABLE;
u16 u16FanDirection;
void URC_vRemoteCtrlHandle(void)
{
  switch (sRemoteCtrl.uFrame.sFrame.u32Command)
  {
     case URC_E_CMD_FAN_FORWARD:
    {
      if(sRemoteCtrl.ubFanRun)
      {
        u16FanDirection = TRUE;
        u8BeepTime = 1;
      }
      break;
    }

    case URC_E_CMD_FAN_REVERSE:
    {
      if(sRemoteCtrl.ubFanRun)
      {
        u16FanDirection = FALSE;
        u8BeepTime = 1;
      }
      break;
    }

```

Fig. 8-8

8.3 Timing function

8.3.1 Add timing keys

Assuming that the remote control has 1, 2, 3, 4 hours of scheduled shutdown function, and the key codes are 0x21, 0x22, 0x23 and 0x24, then the key setting is shown in Figure 8-9.

User_RemoteController.h

```
77   * @enum URC_E_COMMAND_ENUM
78   * @brief 遥控模块帧指令类型
79   */
80  typedef enum URC_E_COMMAND_ENUM_
81  {
82    //风扇档位及开关，使用中
83    // Fan speed levels and switches, in use
84    URC_E_CMD_FAN_SPEED_LV1      = 0x07,//风速1键码 // Speed 1 key code
85    URC_E_CMD_FAN_SPEED_LV2      = 0x05,//风速2键码 // Speed 2 key code
86    URC_E_CMD_FAN_SPEED_LV3      = 0x09,//风速3键码 // Speed 3 key code
87    URC_E_CMD_FAN_SPEED_LV4      = 0x0B,//风速4键码 // Speed 4 key code
88    URC_E_CMD_FAN_SPEED_LV5      = 0x08,//风速5键码 // Speed 5 key code
89    URC_E_CMD_FAN_SPEED_LV6      = 0x11,//风速6键码 // Speed 6 key code
90    URC_E_CMD_FAN_SPEED_LV7      = 0x12,//风速7键码 // Speed 7 key code
91    URC_E_CMD_FAN_SPEED_LV8      = 0x13,//风速8键码 // Speed 8 key code
92    URC_E_CMD_FAN_OFF            = 0x03,//按键OFF键码 // OFF button key code
93    URC_E_CMD_FAN_ON             = 0x01,//按键ON键码 // ON button key code
94    URC_E_CMD_FAN_FORWARD        = 0xF8,//按键FORWARD键码 // Forward button key code
95    URC_E_CMD_FAN_REVERSE        = 0xF9,//按键REVERSE键码 // Reverse button key code
96    //定时，使用中
97    //Timer,in use
98    URC_E_CMD_SCHEDULED_1H = 0x21,//定时1个小时 // Timer for 1 hour
99    URC_E_CMD_SCHEDULED_2H = 0x22,//定时2个小时 // Timer for 2 hour
100   URC_E_CMD_SCHEDULED_3H = 0x23,//定时3个小时 // Timer for 3 hour
101   URC_E_CMD_SCHEDULED_4H = 0x24,//定时4个小时 // Timer for 4 hour
102
```

Fig. 8-9

8.3.2 Add the timing button function

User_RemoteController.c

```
399     case URC_E_CMD_SCHEDULED_1H:
400     {
401       if(sRemoteCtrl.ubFanRun)
402       {
403         u8BeepTime = 1;
404         u8FanTimer = 1;
405       }
406       break;
407     }
408
409
410     case URC_E_CMD_SCHEDULED_2H:
411     {
412       if(sRemoteCtrl.ubFanRun)
413       {
414         u8BeepTime = 1;
415         u8FanTimer = 2;
416       }
417       break;
418     }
419     case URC_E_CMD_SCHEDULED_3H:
420     {
421       if(sRemoteCtrl.ubFanRun)
422       {
423         u8BeepTime = 1;
424         u8FanTimer = 3;
425       }
```

Fig. 8-10

8.3.3 Timer function execution function

The "URC_vFanTimer(u8FanTimer)" function is responsible for executing the timing function, where u8FanTimer represents the timing duration, u16TimerCnt is the time count in seconds and TIMER_1H is set to 1 hour. All these specific parameters are detailed in the header file. If the user presses the URC_E_CMD_SCHEDULED_1H button, the system will shut down when the time count u16TimerCnt exceeds 1 hour.

User_RemoteController.c

```
618
619  void URC_vFanTimer(u8 TimerVal)
620  {
621
622    static u16 u16Timer1Ms;
623    if(TimerVal > 0)
624    {
625      u16Timer1Ms++;
626      if(u16Timer1Ms > 1000)
627      {
628        u16Timer1Ms = 0;
629        u16TimerCnt++;
630
631      }
632      if(TimerVal == 1)
633      {
634          if(u16TimerCnt > TIMER_1H)
635          {
636            u16TimerCnt = 0;
637            sRemoteCtrl.ubFanRun = 0;
638            u8FanTimer = 0;
639            sRemoteCtrl.ubStateChanged = 1;
640          }
641      }
```

Fig. 8-11

```
#define TIMER_1H      (3600*1)        // 1 hour is equal to 3600 seconds
#define TIMER_2H      (TIMER_1H*2)  // 2 hours is twice the TIMER_1H
#define TIMER_3H      (TIMER_1H*3)  // 3 hours is three times the TIMER_1H
#define TIMER_4H       (TIMER_1H*4)// 4 hours is four times the TIMER_1H
```

User_RemoteController.h

```
59  #define TIMER_1H                    (3600*1)
60  #define TIMER_2H                    (TIMER_1H*2)
61  #define TIMER_3H                    (TIMER_1H*3)
62  #define TIMER_4H                    (TIMER_1H*4)
```

Fig. 8-12

8.4 Modify the code button

The default code step of the current project is KEY_ON→KEY_OFF→KEY_1. Users can change the code button according to their needs.

User_RemoteController.c

```
117         else
118         {
119           if(FPowerOn)
120           {                                                                    1
121             if(psHandle->uFrame.sFrame.u32Command == URC_E_CMD_FAN_ON && MatchStep == 0)
122             {
123               MatchStep = 1;                                                   2
124             }
125             else if(psHandle->uFrame.sFrame.u32Command == URC_E_CMD_FAN_OFF && MatchStep == 1)
126             {
127               MatchStep = 2;                                                   3
128             }
129             else if(psHandle->uFrame.sFrame.u32Command == URC_E_CMD_FAN_SPEED_LV1 && MatchStep == 2)
130             {
131               MatchStep = 3;
132               u8BeepTime = 2;
133               FSaveUser = TRUE;
134               u32User2 = u32User1;
135               u32User1 = psHandle->uFrame.sFrame.u32Address;
136               FPowerOn = FALSE;
137             }
138             else
139             {
140               MatchStep = 0;
141             }
```

Fig. 8-13

9. Configure the Remote Controller

There are two kinds of situations to change the remote control. The first is that the user does not know the synchronization code (starting code), Bit0 and Bit1 parameters and key value of the remote control protocol; the second is that the user knows the synchronization code (starting code), Bit0 and Bit1 parameters and key value of the remote control protocol.

9.1 The user has a remote control protocol

For example, the user has the following parameters for the remote control.

	Signal length (us)	high level (us)	low level (us)
SYN code	12400	400	12000
Bit1	1620	1220	400
Bit0	1620	400	1220

For these parameters, we only need to pay attention to the width of the low level, fill in the corresponding maximum and minimum values of the low level, and include the target level, as shown in figure 9-1.

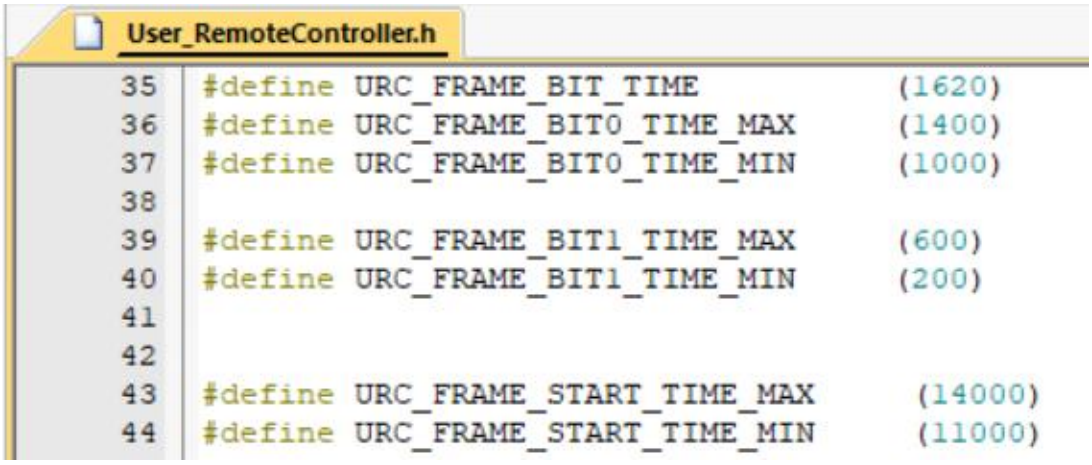


Fig. 9-1

Fill in the corresponding key value into the setting, as shown in figure 9-2:

User_RemoteController.h

```
/**
 * @enum URC_E_COMMAND_ENUM
 * @brief 遥控模块帧指令类型
 */
typedef enum URC_E_COMMAND_ENUM_
{
  //风扇档位及开关，使用中
  // Fan speed levels and switches, in use
  URC_E_CMD_FAN_SPEED_LV1      = 0x07,//风速1键码 // Speed 1 key code
  URC_E_CMD_FAN_SPEED_LV2      = 0x05,//风速2键码 // Speed 2 key code
  URC_E_CMD_FAN_SPEED_LV3      = 0x09,//风速3键码 // Speed 3 key code
  URC_E_CMD_FAN_SPEED_LV4      = 0x0B,//风速4键码 // Speed 4 key code
  URC_E_CMD_FAN_SPEED_LV5      = 0x08,//风速5键码 // Speed 5 key code
  URC_E_CMD_FAN_SPEED_LV6      = 0x11,//风速6键码 // Speed 6 key code
  URC_E_CMD_FAN_SPEED_LV7      = 0x12,//风速7键码 // Speed 7 key code
  URC_E_CMD_FAN_SPEED_LV8      = 0x13,//风速8键码 // Speed 8 key code
  URC_E_CMD_FAN_OFF            = 0x03,//按键OFF键码 // OFF button key code
  URC_E_CMD_FAN_ON             = 0x01,//按键ON键码 // ON button key code
  URC_E_CMD_FAN_FORWARD        = 0xF8,//按键FORWARD键码 // Forward button key code
  URC_E_CMD_FAN_REVERSE        = 0xF9,//按键REVERSE键码 // Reverse button key code
  //定时，使用中
  //Timer,in use
  URC_E_CMD_SCHEDULED_1H = 0x21,//定时1个小时 // Timer for 1 hour
  URC_E_CMD_SCHEDULED_2H = 0x22,//定时2个小时 // Timer for 2 hour
  URC_E_CMD_SCHEDULED_3H = 0x23,//定时3个小时 // Timer for 3 hour
  URC_E_CMD_SCHEDULED_4H = 0x24,//定时4个小时 // Timer for 4 hour
```

Fig. 9-2

After completing the above steps, you can use the new remote control to control the fan.

9.2 User has no remote control protocol

If the user does not have the relevant parameters of the remote control protocol, the user needs to measure the relevant parameters by himself.

9.2.1 Measure the level width of the data

The measurement requires one 433 receiving module, one oscilloscope and one terminal user remote control. After power on, the signal of the remote control is captured.

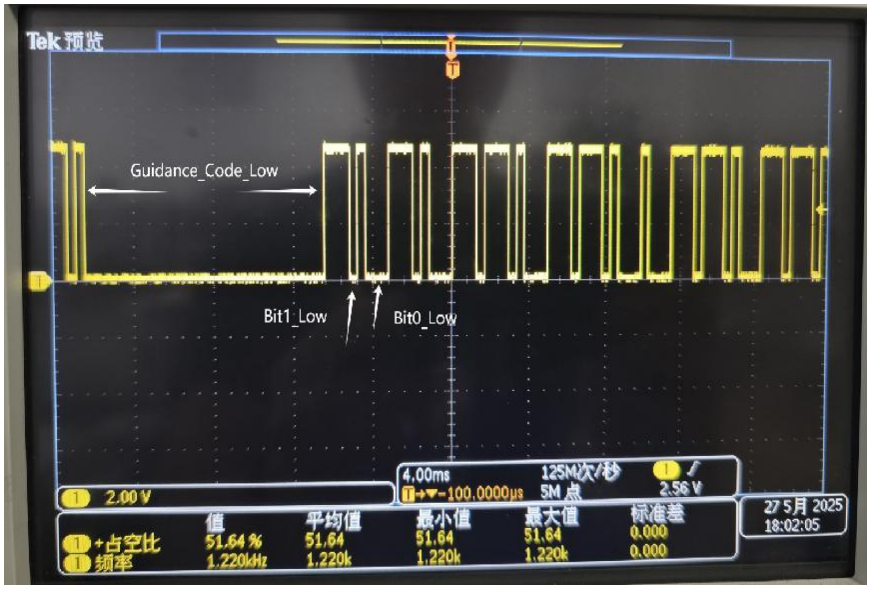


Fig. 9-3

The maximum and minimum values of the measured data are filled in figure 9-1.

9.2.2 Debug to check the corresponding key value

After setting the maximum and minimum values of the pulse, the project can decode the 433 signal normally. Star debug, using Watch1 to read u32PreCommand can obtain the key value and fill in the corresponding key value into Figure 9-2.

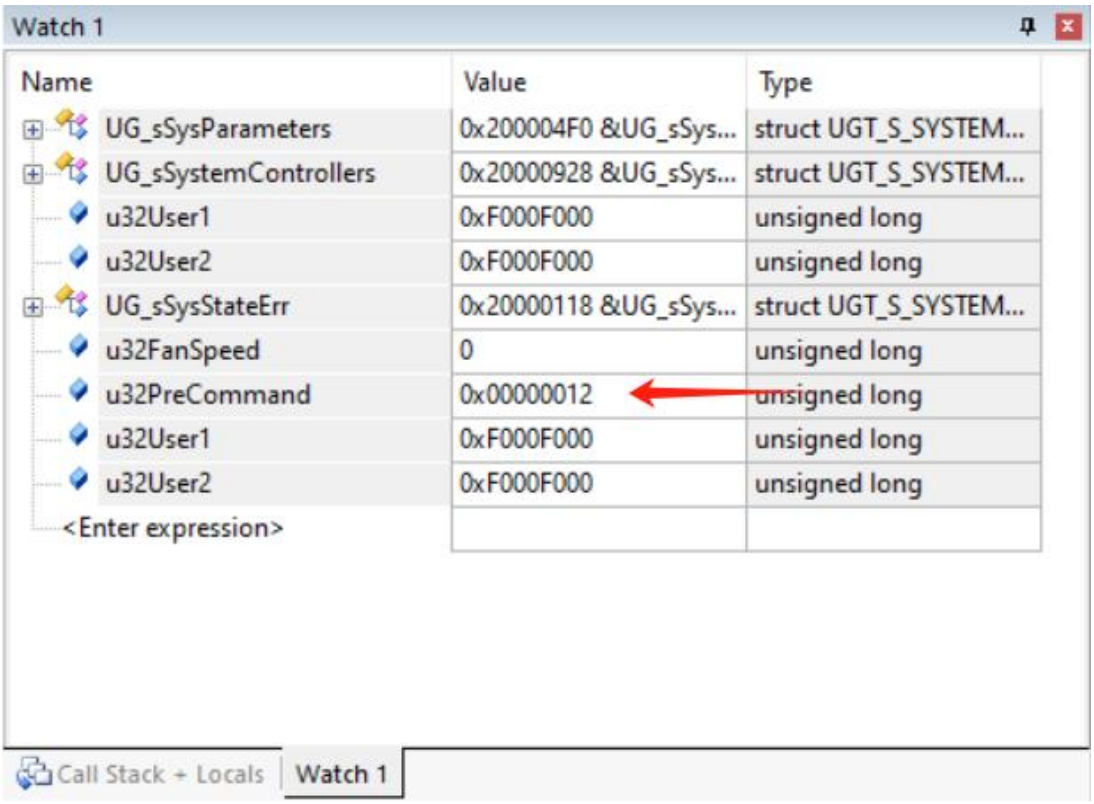


Fig. 9-4