BLDC Debugging manual

SNR8503M

Ver: V1.1

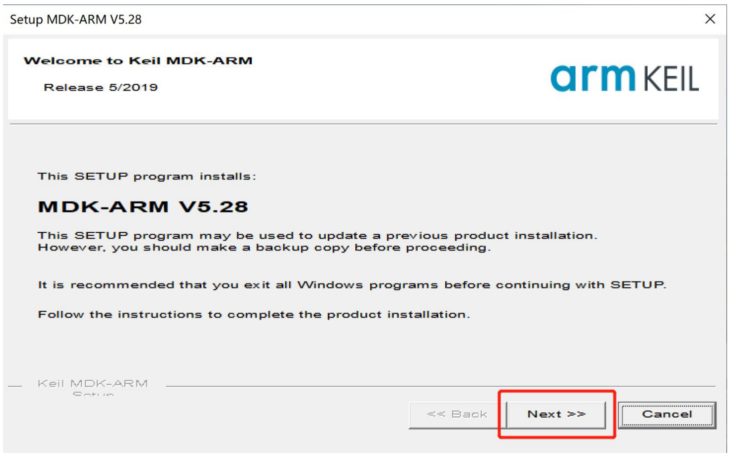
catalog

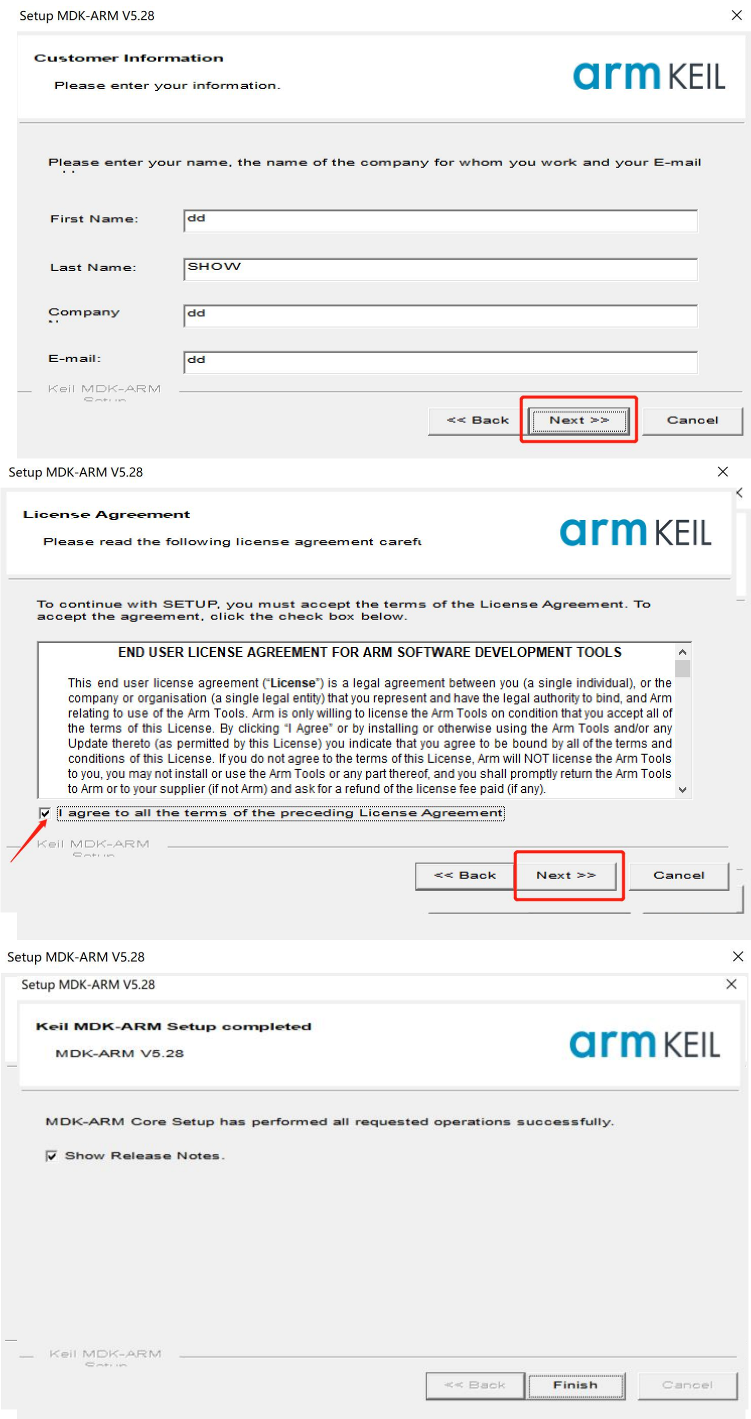
1. Development environment setup

1. 1Install compiler KEIL MDK

The user has installed KEIL MDK software, please skip this chapter.

If it has not been installed before, please follow the<mdk528. exe>installation package provided by our company and follow the prompts to complete the installation. The steps are shown in the following figure：



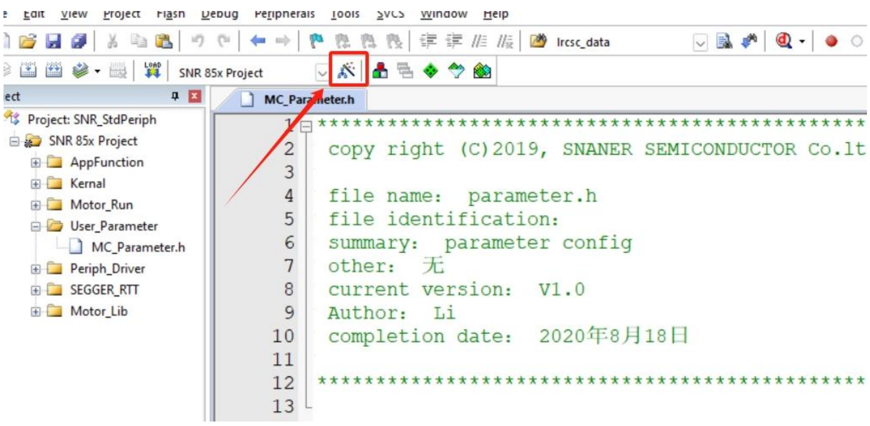


After completion, close all the pop-up pop ups, and KEIL MDK has been installed.

1.2 Engineering burning settings

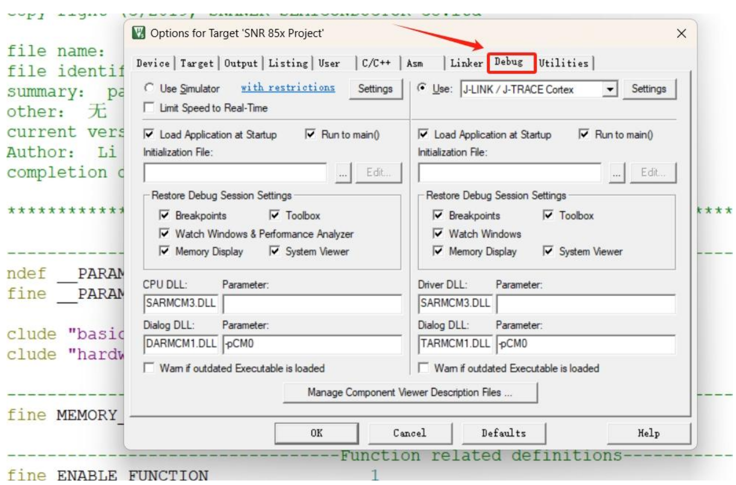
File<SNR8503x FLM>Copy to KEIL installation path, for example C: \ Keil_ In v5 \ ARM \ Flash, the following file:

Next, open the project and double-click the red box icon in the following image:

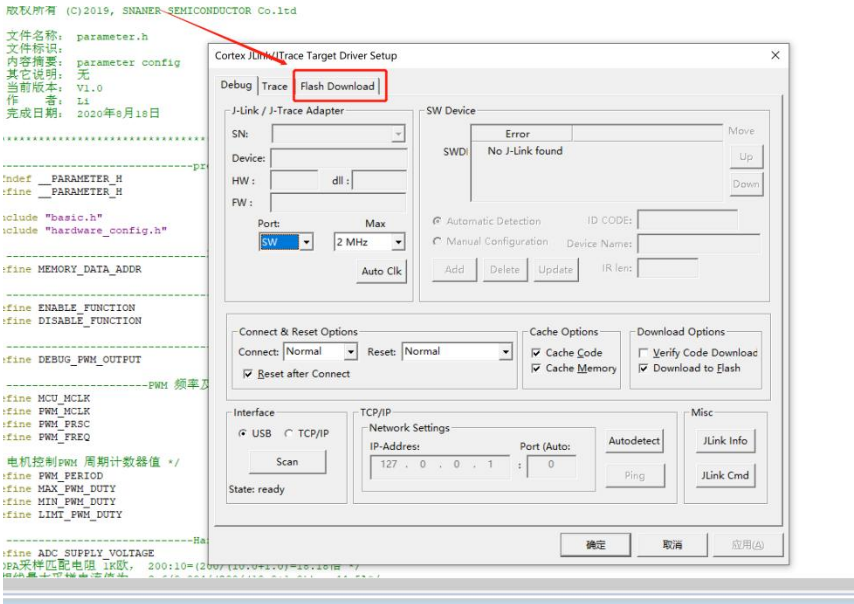


Click on the red box icon in the following image:

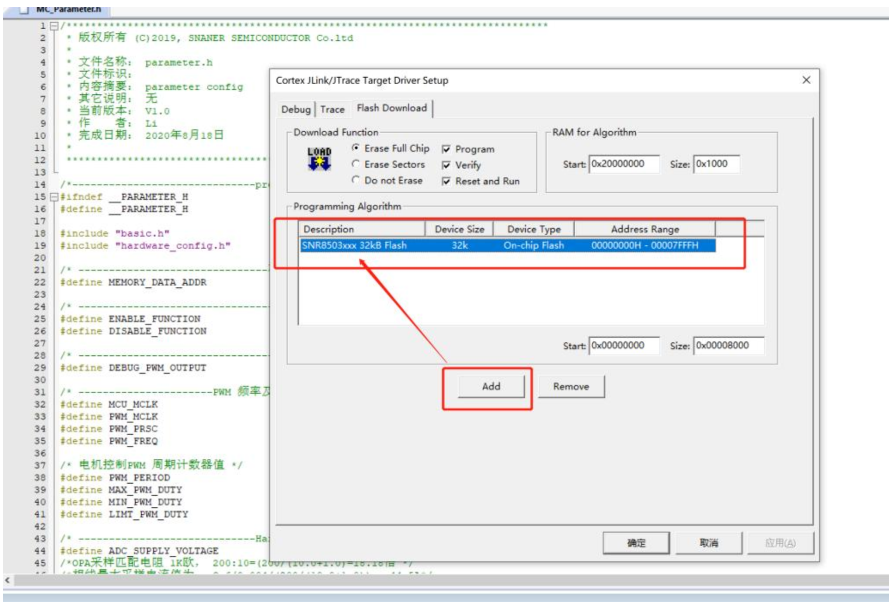
Select the red box tab in the following image:



Select the following tab:



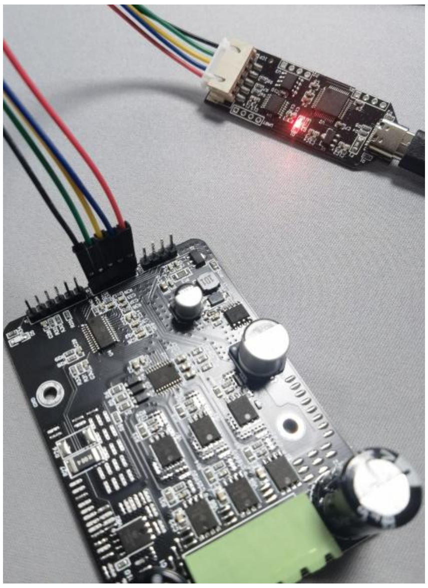
Load the options shown in the following picture. If they already exist, there is no need to add them again. Click OK to exit.



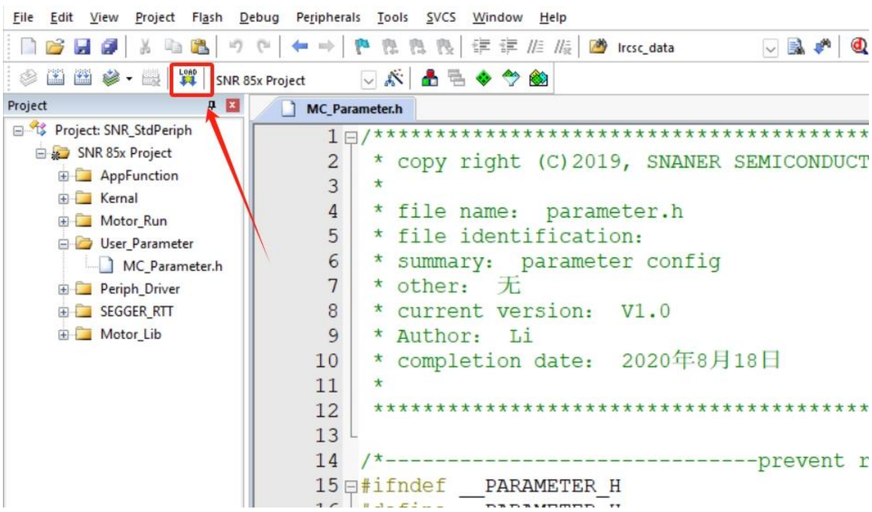
At this point, the project burning has been completed.

1.3 Engineering burning demonstration

Prepare our company's dedicated BLDC burning simulator, connect the burner to the module, with a total of 5 wires. The red wire of the burner is connected to 5V, the blue wire is connected to ICPCK, the yellow wire is connected to ICPDA, the green wire is connected to ICPCS, and the black wire is connected to GND, as shown in the following figure:

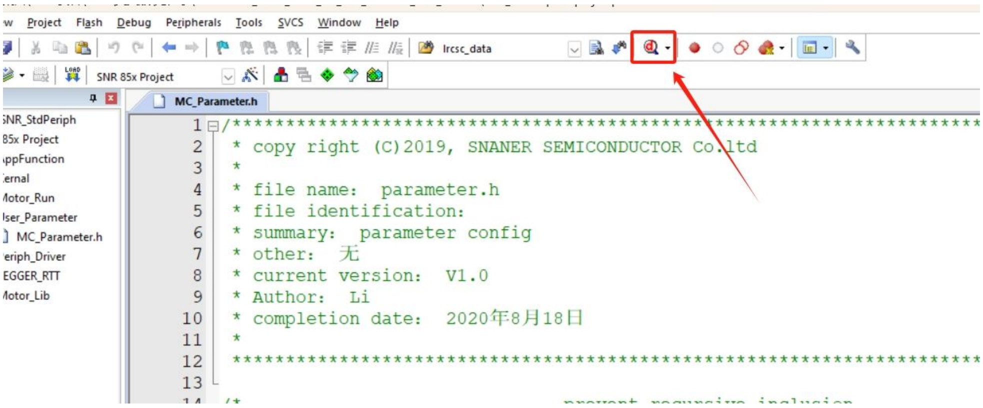


After connecting, click the button in the following picture to start burning. If no error window pops up, it indicates that the burning is complete:



1.4 Engineering simulation demonstration

The BLDC burning simulator supports online simulation function. After successfully burning according to the above operation, click the red button in the figure below to enter the online simulation debugging page.

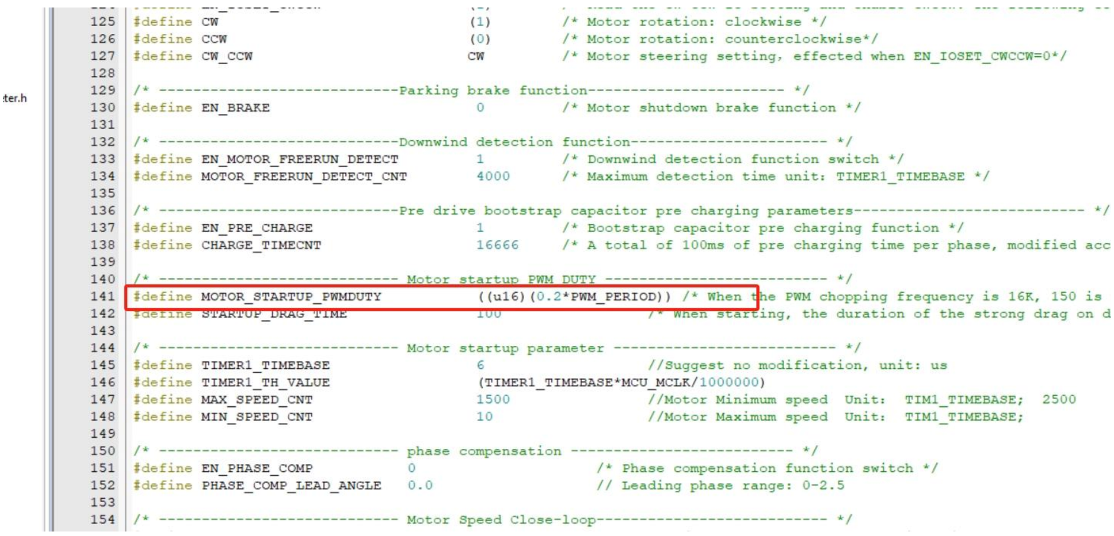


2. Motor startup debugging

The DC brushless motor without induction/Hall effect can affect the starting effect due to different commutation angles, motor phase resistance/inductance, and starting load sizes. If the motor fails to start, users should debug it according to the following methods.

2. 1 Strong drag start parameters

As shown in the red box parameter in the figure below, it is the forced phase change time for motor startup. Based on motor debugging experience, for high-speed motors and small motors with light loads, the parameter value should be adjusted down a bit, such as in the range of 300~2000; For large or heavy-duty motors, increase the parameter values slightly, such as in the range of 1000~5000; If the startup is smooth and normal, it indicates that the parameter has been debugged properly.



2. 2 Starting torque

```
#define CW                          (1)         /* Motor rotation: clockwise */
#define CCW                         (0)         /* Motor rotation: counterclockwise*/
#define CW_CCW                      CW          /* Motor steering setting, effected when EN_IOSET_CWCCW=0*/

/* ----------------------------Parking brake function----------------------- */
#define EN_BRAKE                     0          /* Motor shutdown brake function */

/* ----------------------------Downwind detection function----------------------- */
#define EN_MOTOR_FREERUN_DETECT      1          /* Downwind detection function switch */
#define MOTOR_FREERUN_DETECT_CNT     4000       /* Maximum detection time unit: TIMER1_TIMEBASE */

/* ----------------------------Pre drive bootstrap capacitor pre charging parameters---------------------------
#define EN_PRE_CHARGE                1          /* Bootstrap capacitor pre charging function */
#define CHARGE_TIMECNT               16666      /* A total of 100ms of pre charging time per phase, modified a

/* ---------------------------- Motor startup PWM DUTY ------------------------- */
#define MOTOR_STARTUP_PWMDUTY        ((u16)(0.2*PWM_PERIOD)) /* When the PWM chopping frequency is 16K, 150 i
#define STARTUP_DRAG_TIME            100                  /* When starting, the duration of the strong drag on

/* ---------------------------- Motor startup parameter -------------------------- */
#define TIMER1_TIMEBASE              6                    //Suggest no modification, unit: us
#define TIMER1_TH_VALUE              (TIMER1_TIMEBASE*MCU_MCLK/1000000)
#define MAX_SPEED_CNT                1500                 //Motor Minimum speed  Unit:  TIM1_TIMEBASE;  2500
#define MIN_SPEED_CNT                10                   //Motor Maximum speed  Unit:  TIM1_TIMEBASE;

/* ---------------------------- phase compensation -------------------------- */
#define EN_PHASE_COMP                0                    /* Phase compensation function switch */
#define PHASE_COMP_LEAD_ANGLE        0.0                  // Leading phase range: 0-2.5

/* ---------------------------- Motor Speed Close-loop--------------------------- */
#define EN_MOTOR_SPEED_CLOSELOOP      (0)                  /* Speed closed-loop function */
#define MOTOR_POLES                   2                    /* Number of motor poles 10 */
#define MOTOR_SPEED_X                 ((60*1000000/MOTOR_POLES)/TIMER1_TIMEBASE)  /* Speed calculation coeffic
```

As shown in the red box parameter in the figure below, it is used to set the starting torque of the motor. In fact, it is used to adjust the starting duty cycle. The larger the duty cycle, the greater the starting torque, and the range is generally between 0.05 and 0.3. Please note that<MOTOR_ STARTUP_ PWMDUTY>cannot be less than<MIN_ PWM_ DUTY>。

In addition, assuming that the motor load is particularly heavy, it is necessary to increase the starting and dragging time as follows, with a range of 100-500ms. This parameter generally does not need to be adjusted separately, and can be set to 100ms by default.

Special attention: When the user has debugged the motor to operate and turned the potentiometer at a lower speed, if there is a false alarm of stalling, simply increase the starting torque until the situation no longer occurs.

At this point, the motor startup debugging has been completed.

3. Adjust the overvoltage and undervoltage protection value

Due to the different rated voltages of different motors, the program defaults to setting undervoltage protection at 6V and overvoltage protection at 78V. Users need to modify the overvoltage and undervoltage protection parameters for their own motors.

3. 1 Undervoltage protection parameters

As shown in the red box below, the parameters are undervoltage protection parameters, which include<first undervoltage value>,<second undervoltage value>, and<undervoltage recovery value>。

- The first undervoltage value: the voltage value that triggers the undervoltage protection, which needs to be smaller than the second undervoltage value. Users can adjust it according to their needs, and the default continuous response time is 500ms.
- The second undervoltage value: the voltage value that triggers the undervoltage protection, which needs to be larger than the first undervoltage value. Users can adjust it according to their needs, and the default continuous response time is 50ms.
- Undervoltage recovery value: The voltage value to restore normal operation, which needs to be larger than the first and second undervoltage values, generally smaller than the rated voltage. Users can debug according to their needs, and the default continuous recovery time is 500ms.

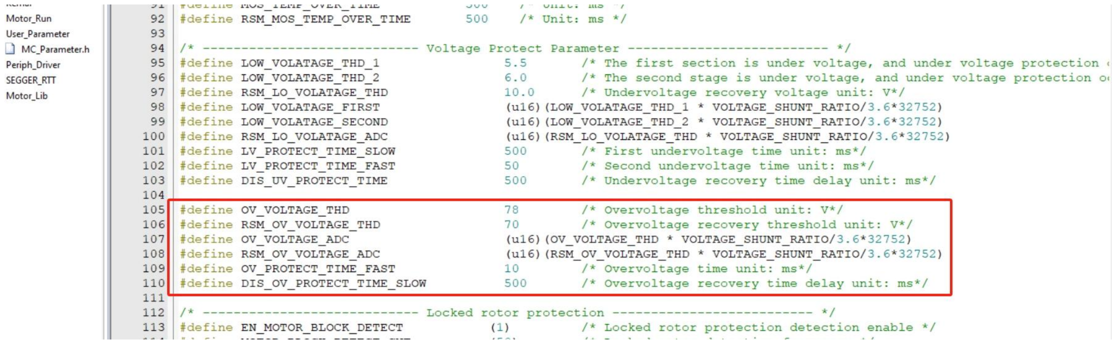
```
#define RSM_MOS_TEMP_OVER_TIME       500    /* Unit: ms */

/* ---------------------------- Voltage Protect Parameter -------------------------- */
#define LOW_VOLATAGE_THD_1                    5.5        /* The first section is under voltage, and under voltage protection
#define LOW_VOLATAGE_THD_2                    6.0        /* The second stage is under voltage, and under voltage protection
#define RSM_LO_VOLATAGE_THD                   10.0       /* Undervoltage recovery voltage unit: V*/
#define LOW_VOLATAGE_FIRST                    (u16)(LOW_VOLATAGE_THD_1 * VOLTAGE_SHUNT_RATIO/3.6*32752)
#define LOW_VOLATAGE_SECOND                   (u16)(LOW_VOLATAGE_THD_2 * VOLTAGE_SHUNT_RATIO/3.6*32752)
#define RSM_LO_VOLATAGE_ADC                   (u16)(RSM_LO_VOLATAGE_THD * VOLTAGE_SHUNT_RATIO/3.6*32752)
#define LV_PROTECT_TIME_SLOW                  500        /* First undervoltage time unit: ms*/
#define LV_PROTECT_TIME_FAST                  50         /* Second undervoltage time unit: ms*/
#define DIS_UV_PROTECT_TIME                   500        /* Undervoltage recovery time delay unit: ms*/

#define OV_VOLTAGE_THD                        78         /* Overvoltage threshold unit: V*/
#define RSM_OV_VOLTAGE_THD                    70         /* Overvoltage recovery threshold unit: V*/
#define OV_VOLTAGE_ADC                        (u16)(OV_VOLTAGE_THD * VOLTAGE_SHUNT_RATIO/3.6*32752)
#define RSM_OV_VOLTAGE_ADC                    (u16)(RSM_OV_VOLTAGE_THD * VOLTAGE_SHUNT_RATIO/3.6*32752)
#define OV_PROTECT_TIME_FAST                  10         /* Overvoltage time unit: ms*/
#define DIS_OV_PROTECT_TIME_SLOW              500        /* Overvoltage recovery time delay unit: ms*/

/* ---------------------------- Locked rotor protection -------------------------- */
```

3. 2 Overvoltage protection parameters

The parameters highlighted in red in the following figure are overvoltage protection parameters, including<overvoltage threshold>and<overvoltage recovery threshold>.

- Overvoltage threshold: The voltage value that triggers overvoltage protection. Users can adjust it according to their needs, and the default continuous response time is 10ms.
- Overvoltage recovery threshold: The voltage value required to restore normal operation, which needs to be larger than the overvoltage threshold value.

Users can adjust it according to their needs, and it is generally larger than the rated voltage. The default continuous response time is 500ms.



```
#define RSM_MOS_TEMP_OVER_TIME       500    /* Unit: ms */

/* --------------------------- Voltage Protect Parameter ------------------------- */
#define LOW_VOLATAGE_THD_1            5.5       /* The first section is under voltage, and under voltage protection
#define LOW_VOLATAGE_THD_2            6.0       /* The second stage is under voltage, and under voltage protection o
#define RSM_LO_VOLATAGE_THD           10.0      /* Undervoltage recovery voltage unit: V*/
#define LOW_VOLATAGE_FIRST            (u16)(LOW_VOLATAGE_THD_1 * VOLTAGE_SHUNT_RATIO/3.6*32752)
#define LOW_VOLATAGE_SECOND           (u16)(LOW_VOLATAGE_THD_2 * VOLTAGE_SHUNT_RATIO/3.6*32752)
#define RSM_LO_VOLATAGE_ADC           (u16)(RSM_LO_VOLATAGE_THD * VOLTAGE_SHUNT_RATIO/3.6*32752)
#define LV_PROTECT_TIME_SLOW          500       /* First undervoltage time unit: ms*/
#define LV_PROTECT_TIME_FAST          50        /* Second undervoltage time unit: ms*/
#define DIS_UV_PROTECT_TIME           500       /* Undervoltage recovery time delay unit: ms*/

#define OV_VOLTAGE_THD                78        /* Overvoltage threshold unit: V*/
#define RSM_OV_VOLTAGE_THD            70        /* Overvoltage recovery threshold unit: V*/
#define OV_VOLTAGE_ADC                (u16)(OV_VOLTAGE_THD * VOLTAGE_SHUNT_RATIO/3.6*32752)
#define RSM_OV_VOLTAGE_ADC            (u16)(RSM_OV_VOLTAGE_THD * VOLTAGE_SHUNT_RATIO/3.6*32752)
#define OV_PROTECT_TIME_FAST          10        /* Overvoltage time unit: ms*/
#define DIS_OV_PROTECT_TIME_SLOW      500       /* Overvoltage recovery time delay unit: ms*/

/* --------------------------- Locked rotor protection ------------------------- */
#define EN_MOTOR_BLOCK_DETECT        (1)        /* Locked rotor protection detection enable */
```

4. Current protection value

Due to the different rated currents of different motors and the maximum current that MOS transistors can withstand, users need to make adjustments accordingly. The default short-circuit current protection value of the program is 50A, with a primary current limit of 20A and a secondary current limit of 21A.

4.1 Short circuit current protection

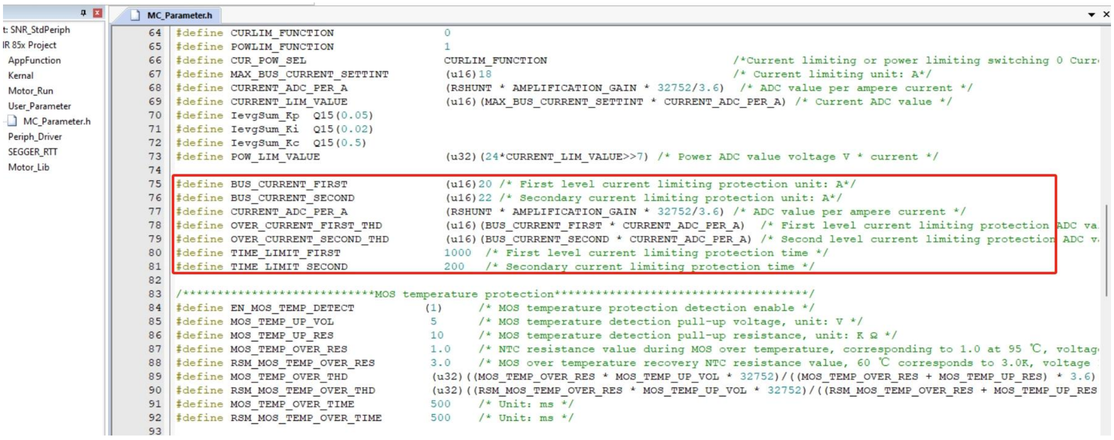
As shown in the red box parameter in the figure below, it is the protection value for short-circuit current, which can generally refer to the selected MOS transistor ID@TC =100 ℃, set slightly lower than this value to ensure timely protection during high or short circuit currents.

4.2 First and second level current limiting protection

As shown in the red box parameter in the figure below, it is the protection value for the first and second level current limiting. Generally, it is set according to the maximum current of the motor to ensure timely protection when operating beyond the load current, and to avoid motor damage.

- First level overcurrent protection: The current value that triggers the protection needs to be slightly smaller than the second level overcurrent protection setting. Users can adjust it according to their needs, and the default continuous response time is 1000ms.

- Secondary overcurrent protection: The current value that triggers the protection needs to be slightly higher than the setting of the primary overcurrent

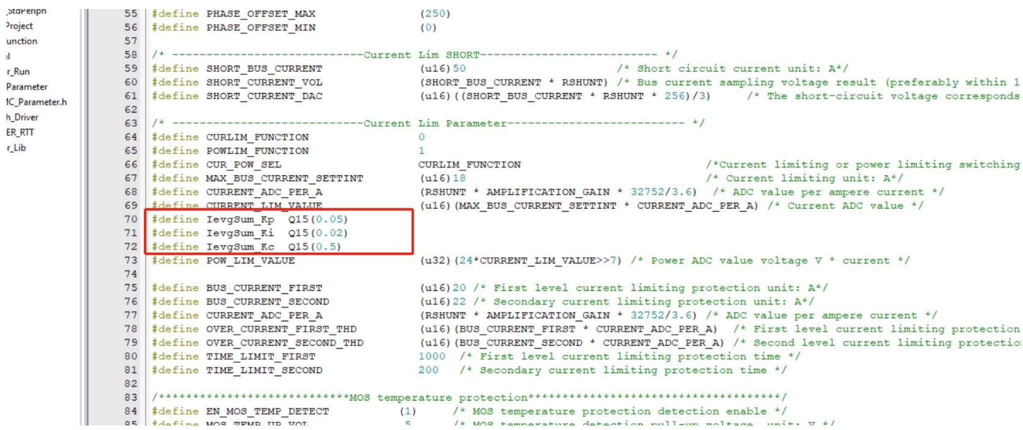


protection. Users can adjust it according to their needs, and the default continuous response time is 200ms.

4.3 Current limiting operation

As shown in the red box parameter in the figure below, it is the value for current limited operation. The program defaults to a maximum limit of 18A current operation. If it exceeds the limit, the current will remain constant at this current value. Users can modify the settings according to their needs to ensure that the maximum current value for motor operation does not exceed.

Please note that if the test is in a current limited state and runs unstable, it is necessary to reset the PID parameters in the red box in the figure below. Users can adjust them themselves and generally do not need to modify them.



5. MOS over temperature protection

To protect the MOS from stopping operation in case of high temperature and returning to normal operation in case of low temperature, NTC components have been added to the circuit to be placed near the MOS for real-time detection of the temperature situation of the MOS.

It should be noted that when users replace NTC components themselves, they should pay attention to modifying the corresponding program parameter configuration, as shown in the following figure:

```
#define OVER_CURRENT_FIRST_THD          (u16)(BUS_CURRENT_FIRST * CURRENT_ADC_PER_A) /* First level current limiting protection A
#define OVER_CURRENT_SECOND_THD         (u16)(BUS_CURRENT_SECOND * CURRENT_ADC_PER_A) /* Second level current limiting protection A
#define TIME_LIMIT_FIRST                1000  /* First level current limiting protection time */
#define TIME_LIMIT_SECOND               200   /* Secondary current limiting protection time */

/****************************MOS temperature protection************************************/
#define EN_MOS_TEMP_DETECT           (1)      /* MOS temperature protection detection enable */
#define MOS_TEMP_UP_VOL               5       /* MOS temperature detection pull-up voltage, unit: V */
#define MOS_TEMP_UP_RES               10      /* MOS temperature detection pull-up resistance, unit: K Ω */
#define MOS_TEMP_OVER_RES             1.0     /* NTC resistance value during MOS over temperature, corresponding to 1.0 at 95 ℃, vo
#define RSM_MOS_TEMP_OVER_RES         3.0     /* MOS over temperature recovery NTC resistance value, 60 ℃ corresponds to 3.0K, volt
#define MOS_TEMP_OVER_THD             (u32)((MOS_TEMP_OVER_RES * MOS_TEMP_UP_VOL * 32752)/((MOS_TEMP_OVER_RES + MOS_TEMP_UP_RES) *
#define RSM_MOS_TEMP_OVER_THD         (u32)((RSM_MOS_TEMP_OVER_RES * MOS_TEMP_UP_VOL * 32752)/((RSM_MOS_TEMP_OVER_RES + MOS_TEMP_UP
#define MOS_TEMP_OVER_TIME            500     /* Unit: ms */
#define RSM_MOS_TEMP_OVER_TIME        500     /* Unit: ms */
```

According to the above diagram, it is tested that when the MOS temperature exceeds 95 ℃, it will be protected. When it is below 60 ℃, it will resume operation. Users can directly use the same NTC model components as our company, and the components should be placed as close to the MOS as possible without modifying software parameters。

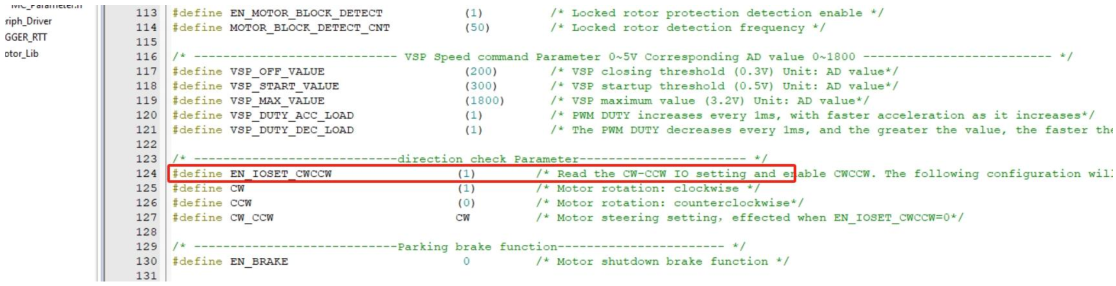
NTC component link： https://item.szlcsc.com/266512.html

Manufacturer：Sunlord

model：SDNT1608X103F3950FTF

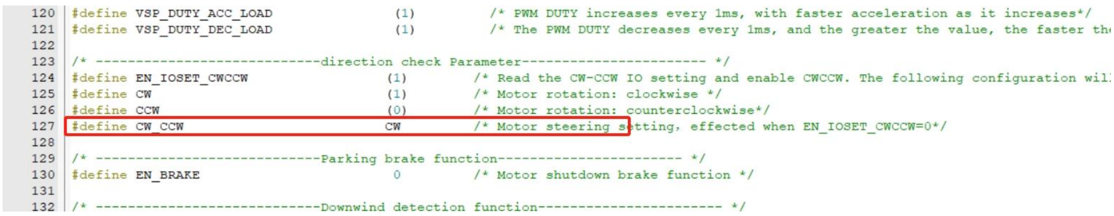
6. REVERSE Servo Reversing

The program defaults to reading the first pin P0 of the chip_ 9 levels are used to set the motor forward or reverse, with high levels indicating CW forward and low levels indicating CCW reverse. Users can disable this function in the program and set<EN_ IOSET_ CWCCW>set to 0, as shown in the figure below:



```
#define EN_MOTOR_BLOCK_DETECT          (1)        /* Locked rotor protection detection enable */
#define MOTOR_BLOCK_DETECT_CNT         (50)       /* Locked rotor detection frequency */

/* --------------------------- VSP Speed command Parameter 0~5V Corresponding AD value 0~1800 ------------------------- */
#define VSP_OFF_VALUE                  (200)      /* VSP closing threshold (0.3V) Unit: AD value*/
#define VSP_START_VALUE                (300)      /* VSP startup threshold (0.5V) Unit: AD value*/
#define VSP_MAX_VALUE                  (1800)     /* VSP maximum value (3.2V) Unit: AD value*/
#define VSP_DUTY_ACC_LOAD              (1)        /* PWM DUTY increases every 1ms, with faster acceleration as it increases*/
#define VSP_DUTY_DEC_LOAD              (1)        /* The PWM DUTY decreases every 1ms, and the greater the value, the faster th

/* ----------------------------direction check Parameter---------------------- */
#define EN_IOSET_CWCCW                (1)       /* Read the CW-CCW IO setting and enable CWCCW. The following configuration wil
#define CW                            (1)       /* Motor rotation: clockwise */
#define CCW                           (0)       /* Motor rotation: counterclockwise*/
#define CW_CCW                        CW        /* Motor steering setting, effected when EN_IOSET_CWCCW=0*/

/* ----------------------------Parking brake function---------------------- */
#define EN_BRAKE                      0         /* Motor shutdown brake function */
```

At this point, the parameter<CW can be modified through the program_ If CCW>is CW or CCW, the software changes the motor direction as shown in the following figure:



```
#define VSP_DUTY_ACC_LOAD              (1)        /* PWM DUTY increases every 1ms, with faster acceleration as it increases*/
#define VSP_DUTY_DEC_LOAD              (1)        /* The PWM DUTY decreases every 1ms, and the greater the value, the faster th

/* ----------------------------direction check Parameter---------------------- */
#define EN_IOSET_CWCCW                (1)       /* Read the CW-CCW IO setting and enable CWCCW. The following configuration wil
#define CW                            (1)       /* Motor rotation: clockwise */
#define CCW                           (0)       /* Motor rotation: counterclockwise*/
#define CW_CCW                        CW        /* Motor steering setting, effected when EN_IOSET_CWCCW=0*/

/* ----------------------------Parking brake function---------------------- */
#define EN_BRAKE                      0         /* Motor shutdown brake function */

/* ----------------------------Downwind detection function---------------------- */
```

7. Speed open-loop/closed-loop debugging

The program defaults to speed open-loop, and the output PWM duty cycle is proportional to the voltage value of VSP, thereby achieving proportional adjustment of motor speed with parameters<EN_ MOTOR_ SPEED_ When CLOSELOOP>0, it is in an open-loop state of speed, as shown in the following figure:

```
5x Project
MC_Parameter.h

/* --------------------------- Motor startup PWM DUTY -------------------------- */
#define MOTOR_STARTUP_PWMDUTY          ((u16)(0.2*PWM_PERIOD)) /* When the PWM chopping frequency is 16K, 150 is the sta
#define STARTUP_DRAG_TIME              100                /* When starting, the duration of the strong drag on duty is

/* --------------------------- Motor startup parameter ------------------------- */
#define TIMER1_TIMEBASE                6                  //Suggest no modification, unit: us
#define TIMER1_TH_VALUE                (TIMER1_TIMEBASE*MCU_MCLK/1000000)
#define MAX_SPEED_CNT                  1500               //Motor Minimum speed  Unit:  TIM1_TIMEBASE;  2500
#define MIN_SPEED_CNT                  10                 //Motor Maximum speed  Unit:  TIM1_TIMEBASE;

/* --------------------------- phase compensation -------------------------- */
#define EN_PHASE_COMP          0                  /* Phase compensation function switch */
#define PHASE_COMP_LEAD_ANGLE  0.0                // Leading phase range: 0-2.5

/* --------------------------- Motor Speed Close-loop-------------------------- */
#define EN_MOTOR_SPEED_CLOSELOOP       (0)                    /* Speed closed-loop function */
#define MOTOR_POLES                    2                      /* Number of motor poles 10 */
#define MOTOR_SPEED_X                  ((60*1000000/MOTOR_POLES)/TIMER1_TIMEBASE)  /* Speed calculation coefficient */
#define MOTOR_SPEED_MAX_RPM            50000                  /* unit: RPM, Closed loop maximum target speed */
#define MOTOR_SPEED_MIN_RPM            200                    /* unit: RPM, Closed loop minimum target speed */
#define SPEED_ACC_MS                  (float)(5.0)            //Speed loop ramp acceleration RPM/MS
#define SPEED_DEC_MS                  (float)(5.0)            //Speed loop climbing deceleration RPM/MS
#define SPEED_PI_PRC                  (2)                     //Speed loop pre division, this macro needs to be manuall
#define SSum_Kp                        Q15(0.05)
#define SSum_Ki                        Q15(0.01)
#define SSum_Kc                        Q15(0.5)

/* --------------------------- Motor IPD-------------------------- */
#define EN_MOTOR_ROTOR_DETECT          0
#define UART0_FUNCTION                DISABLE_FUNCTION        /* Motor Control UART0 Serial Port Control Communication
```

When the user needs to set the speed closed-loop/constant speed control, the following steps are set:

- Accurately input the pole number parameter of the motor<MOTOR_ POLES>, unit pairs
- Input the maximum target speed parameter of the motor<MOTOR_ SPEED_ MAX_ RPM>, unit RPM
- Input motor minimum target speed parameter<MOTOR_ SPEED_ MIN_ RPM>, unit RPM
- Enable speed closed-loop control, set<EN_ MOTOR_ SPEED_ CLOSELOOP>is 1

As shown in the following diagram, the motor has 5 pairs of poles and a speed range of 1000~5000RPM

```
#define PHASE_COMP_LEAD_ANGLE  0.0                // Leading phase range: 0-2.5

/* --------------------------- Motor Speed Close-loop-------------------------- */
#define EN_MOTOR_SPEED_CLOSELOOP       (0)                    /* Speed closed-loop function */
#define MOTOR_POLES                    2                      /* Number of motor poles 10 */
#define MOTOR_SPEED_X                  ((60*1000000/MOTOR_POLES)/TIMER1_TIMEBASE)  /* Speed calculation coefficient */   //
#define MOTOR_SPEED_MAX_RPM            50000                  /* unit: RPM, Closed loop maximum target speed */
#define MOTOR_SPEED_MIN_RPM            200                    /* unit: RPM, Closed loop minimum target speed */
#define SPEED_ACC_MS                  (float)(5.0)            //Speed loop ramp acceleration RPM/MS
#define SPEED_DEC_MS                  (float)(5.0)            //Speed loop climbing deceleration RPM/MS
#define SPEED_PI_PRC                  (2)                     //Speed loop pre division, this macro needs to be manually a
#define SSum_Kp                        Q15(0.05)
```

The user needs to adjust the PID parameters for the stability of the motor's

```
#define EN_MOTOR_SPEED_CLOSELOOP       (0)                    /* Speed closed-loop function */
#define MOTOR_POLES                    2                      /* Number of motor poles 10 */
#define MOTOR_SPEED_X                  ((60*1000000/MOTOR_POLES)/TIMER1_TIMEBASE)  /* Speed calculation coeffi
#define MOTOR_SPEED_MAX_RPM            50000                  /* unit: RPM, Closed loop maximum target speec
#define MOTOR_SPEED_MIN_RPM            200                    /* unit: RPM, Closed loop minimum target speec
#define SPEED_ACC_MS                  (float)(5.0)            //Speed loop ramp acceleration RPM/MS
#define SPEED_DEC_MS                  (float)(5.0)            //Speed loop climbing deceleration RPM/MS
#define SPEED_PI_PRC                  (2)                     //Speed loop pre division, this macro needs tc
#define SSum_Kp                        Q15(0.05)
#define SSum_Ki                        Q15(0.01)
#define SSum_Kc                        Q15(0.5)

/* --------------------------- Motor IPD-------------------------- */
#define EN_MOTOR_ROTOR_DETECT          0
```

operating speed, as shown in the following figure:

At this point, the speed closed-loop/constant speed control has been completed.

8. Phase compensation

This function is not enabled by default. When the phase voltage waveform lags behind or cannot reach the rated speed, the user can enable this function to achieve phase compensation advance control and set the parameter<EN_ PHASE_ COMP>set to 1, as shown in the following figure

```
#define EN_PRE_CHARGE                  1          /* Bootstrap capacitor pre charging function */
#define CHARGE_TIMECNT                 16666      /* A total of 100ms of pre charging time per phase, modif

/* ---------------------------- Motor startup PWM DUTY -------------------------- */
#define MOTOR_STARTUP_PWMDUTY          ((u16)(0.2*PWM_PERIOD)) /* When the PWM chopping frequency is 16K,
#define STARTUP_DRAG_TIME              100                  /* When starting, the duration of the strong dr

/* ---------------------------- Motor startup parameter -------------------------- */
#define TIMER1_TIMEBASE                6                    //Suggest no modification, unit: us
#define TIMER1_TH_VALUE                (TIMER1_TIMEBASE*MCU_MCLK/1000000)
#define MAX_SPEED_CNT                  1500                 //Motor Minimum speed  Unit:  TIM1_TIMEBASE;  2
#define MIN_SPEED_CNT                  10                   //Motor Maximum speed  Unit:  TIM1_TIMEBASE;

/* ---------------------------- phase compensation -------------------------- */
#define EN_PHASE_COMP            0                      /* Phase compensation function switch */
#define PHASE_COMP_LEAD_ANGLE    0.0                    // Leading phase range: 0-2.5

/* ---------------------------- Motor Speed Close-loop-------------------------- */
#define EN_MOTOR_SPEED_CLOSELOOP       (0)                      /* Speed closed-loop function */
#define MOTOR_POLES                    2                        /* Number of motor poles 10 */
#define MOTOR_SPEED_X                  ((60*1000000/MOTOR_POLES)/TIMER1_TIMEBASE)  /* Speed calculation cc
#define MOTOR_SPEED_MAX_RPM            50000                    /* unit: RPM, Closed loop maximum target sp
#define MOTOR_SPEED_MIN_RPM            200                      /* unit: RPM, Closed loop minimum target sp
#define SPEED_ACC_MS                  (float)(5.0)              //Speed loop ramp acceleration RPM/MS
#define SPEED_DEC_MS                  (float)(5.0)              //Speed loop climbing deceleration RPM/MS
#define SPEED_PI_PRC                  (2)                       //Speed loop pre division, this macro needs
#define SSum_Kp                        Q15(0.05)
#define SSum_Ki                        Q15(0.01)
#define SSum_Kc                        Q15(0.5)
```

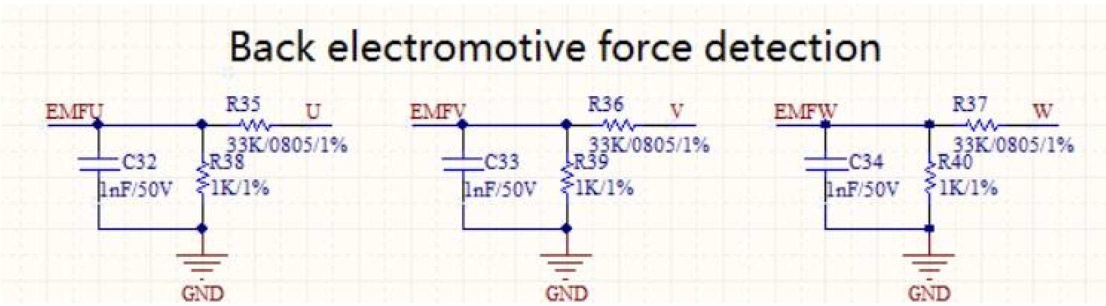
After enabling, adjust the parameter<PHASE_ COMP_ LEAD_ ANGLE>, with a range of 0~2.5. The larger the value, the more obvious the phase leading effect. However, it is important to note whether there will be any abnormalities during motor startup when the set value is large.

9. Hardware debugging

When customers use our company's modules, the hardware circuit is standard and does not need to be adjusted. When more precise circuit tuning is needed, please refer to the following methods.

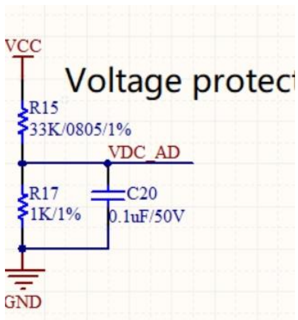
9. 1 Back electromotive force detection circuit

The operation of a non inductive motor relies on the back electromotive force detection circuit for position observation, which is very important. After the back electromotive force is divided, it is recommended to enter the chip pin voltage below 2.5V and above 0.5V, and the current below 5mA and above 1mA, with appropriate margin reserved. Our standard module circuit is shown below, considering compatibility with 80V voltage, with a voltage division ratio set to (R38/(R38+R35)). Users can adjust the circuit parameters based on the rated voltage of the motor. Assuming the rated voltage of the motor is 24V, R35, R36, and R37 can be adjusted to 20K to increase the input voltage after voltage division, enhance the signal-to-noise ratio, and improve observation accuracy.



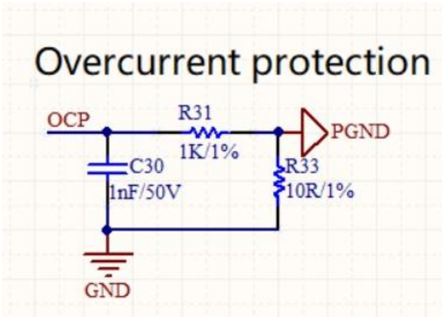
9. 2 Voltage sampling

The method of sampling resistance for bus voltage is shown in the following figure. The voltage division ratio of bus voltage is (R17/(R17+R15)). The bus voltage sampling channel corresponds to ADC_ CHANNEL_ 3. Please note that the voltage after voltage division should not exceed 3.6V, as the reference voltage of the chip ADC defaults to 3.6V.



9. 3 Hardware overcurrent protection

The hardware overcurrent protection adopts comparator and DAC processing, as shown in the following figure, with OCP connected to CMP1_ IP0, configure CMP1 positive terminal to CMP1 in the program_ IP0, negative terminal connected to internal DAC of the chip.



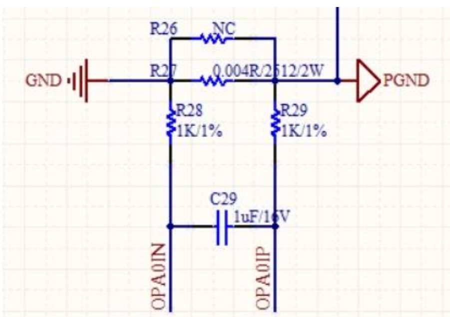
Set 50A hardware overcurrent protection, software as shown in the following figure:

```
#define PHASE_OFFSET_MAX              (250)
#define PHASE_OFFSET_MIN              (0)

/* ----------------------------Current Lim SHORT------------------------- */
#define SHORT_BUS_CURRENT             (u16)50                         /* Short circuit current unit: A*/
#define SHORT_CURRENT_VOL             (SHORT_BUS_CURRENT * RSHUNT) /* Bus current sampling voltage result
#define SHORT_CURRENT_DAC             (u16)((SHORT_BUS_CURRENT * RSHUNT * 256)/3)      /* The short-circuit

/* ----------------------------Current Lim Parameter------------------------- */
#define CURLIM_FUNCTION               0
#define POWLIM_FUNCTION               1
```

9. 4 Current sampling

Due to the chip's support for differential sampling, the current sampling circuit is very simple. The chip is equipped with four sets of operational amplifier feedback resistors, namely 200K/10K, 190K/20K, 180K/30K, and 170K/40K. Our standard module has a sampling resistance of 0.004 Ω, an external feedback resistance of 1K, and an internal feedback resistance of 200K/10K. The designed phase line has a maximum sampling current of 3.6V/0.004R/(200/(10.0+1.0))=44.5A.

In practical projects, it is important to pay attention to setting the maximum sampling current value reasonably. Generally, the design is based on three times the overload. For certain applications, the maximum sampling current value can be appropriately reduced to improve the accuracy of current sampling.



Bus current detection