Application Note

Document No.: AN1098

APM32F035_MOTOR EVAL Senseless Square Wave Control Scheme

Version: V1.1

Contents

1 General Introduction

1.1 Project Overview

APM32F035 is a specialized chip launched by Geehy Semiconductor Co., Ltd. for motor control. Based on APM32F035, this design provides the senseless square wave control scheme and adopts the ADC detection back electromotive force angle estimation scheme. The detailed design specifications are shown in the table below:

Table 1 Design Specifications

Control mode	Senseless square wave six-step commutation
PWM modulation mode	HPWM_LON
Angle acquisition	ADC detection back electromotive force
PWM frequency	8KHz
Number of pole pairs	2 pairs of poles
Motor speed	0~3000RPM
Starting mode	6 Step
Protection function	Overvoltage, undervoltage, overcurrent, locked rotor
Code size	<10Kbytes
Development software	Keil C (V5.23 version and above)

1.2 APM32F035 Chip Resources

APM32F035 is a high-performance special MCU for motor control which is based on the Arm Cortex-M0+ core, integrates the mathematical operation accelerators (Cordic, Svpwm, hardware divider, etc.) commonly used in FOC algorithms, and integrates such analog peripherals as amplifiers and comparators, as well as CAN controllers.

Table 2 Functions and Peripherals of APM32F035 Series Chip

Product		**APM32F035**	
Model		**C8T7**	**K8T7**
Package		LQFP48	LQFP32
Core and maximum working frequency		Arm® 32-bit Cortex®-M0+@72MHz	
M0CP Co-processor		1	
Flash memory (KB)		64	
SRAM(KB)		10	
Timer	32 bit/16 bit universal	1/2	
	16-bit advanced	1	
	16-bit basic	2	

Product		**APM32F035**	
Model		**C8T7**	**K8T7**
	24-bit counter	1	
	Watchdog (WDT)	2 (1 independent watchdog +1 window watchdog)	
	Real-time clock	1	
Communication interface	USART	2	
	SPI/I2S	1/1	
	I2C	1	
	CAN	1	
12-bit ADC	Unit	1	
	External channel	16	12
	Internal channel	3	
Comparator (COMP)		2	
Operational amplifier (OPA)		4	2
GPIOs		42	27
Operating temperature		Ambient temperature: -40℃ to 105℃ Junction temperature: -40℃ to 125℃	
Working voltage		2.0~3.6V	

2 Hardware Introduction

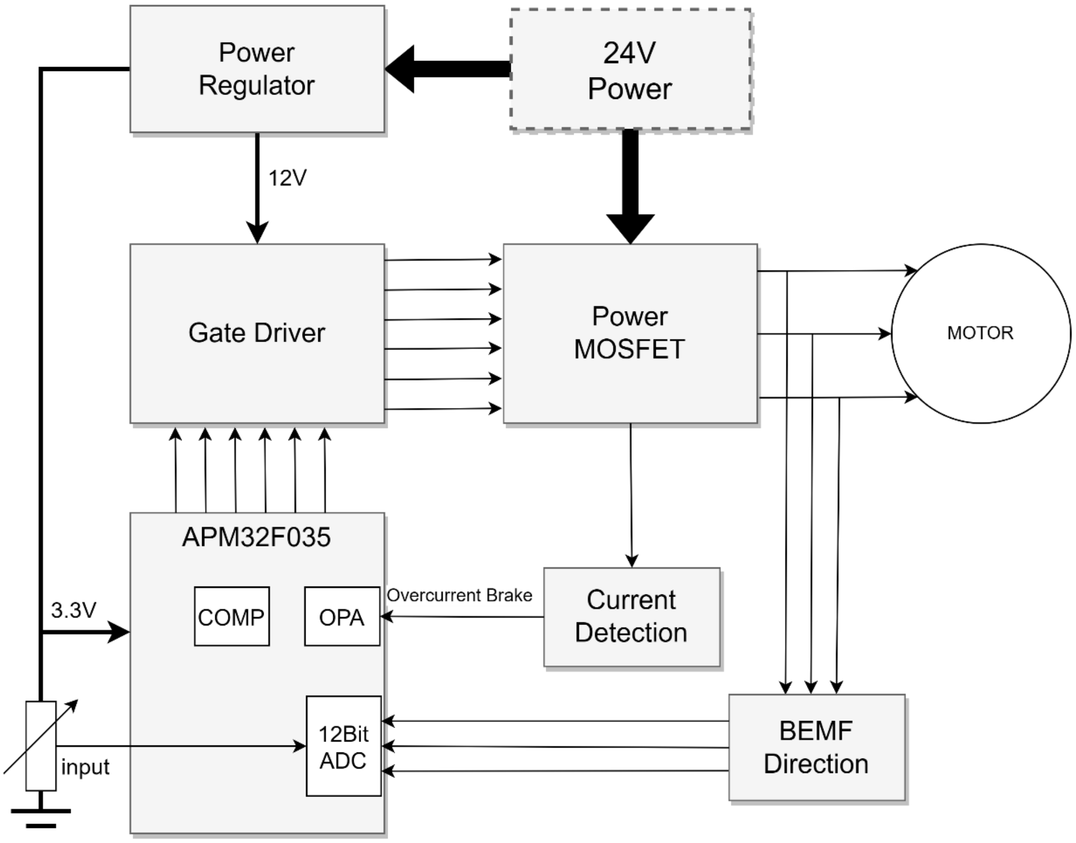
2.1 Overall Hardware Circuit

The overall hardware system is powered by an external 24V power supply and after conversion through the corresponding power step-down circuit, it outputs stable 12V, 5V, and 3.3V voltages. The 12V voltage is output to the Gate driver IC, the 3.3V voltage is output to the APM32F035 series microprocessor, and the power switch tube is directly connected to the 24V power supply. At the same time, this scheme uses a variable resistance knob to adjust the voltage input of 0~3.3V as the input end of the speed command, in order to adjust the motor speed. Users can directly adjust the input voltage by turning the variable resistor knob in actual use. When the input voltage value exceeds the starting threshold, the motor will start running, and when the voltage value is below the threshold, the motor will stop running.

After the motor is started, the internal ADC of the APM32F035 processor takes samples through an external circuit to detect and confirm the zero crossing point signal of the back electromotive force, and achieve six-step commutation of the BLDC motor according to six different inverter MOS transistor driving sequences.

The hardware block diagram is shown in the figure.

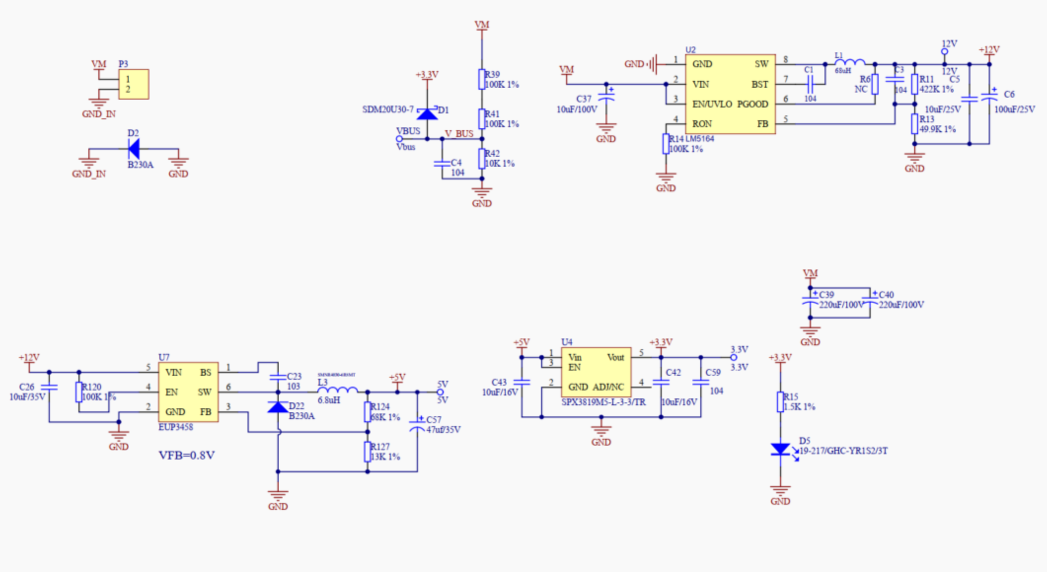
Figure 1 Hardware System Block Diagram



2.2 Interface Circuits and Settings

2.2.1 Power circuit

Figure 2 Power Circuit



As shown in the figure, supply voltage V_BUS =VM/((100K+100K+10K)/10K)=VM/21

A 12-bit ADC is adopted, and the sampling range 0-3.3V corresponds to 0-4096

Then the maximum sampling voltage corresponding to 3.3V is: VM= 3.3 *21 =69.3V

2.2.2 Back electromotive force sampling circuit

Figure 3 MOSFET Circuit

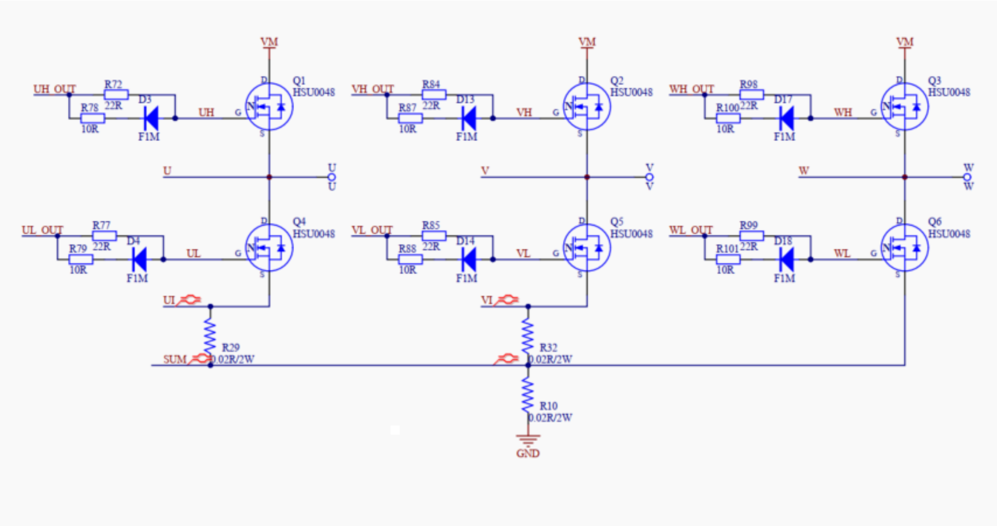
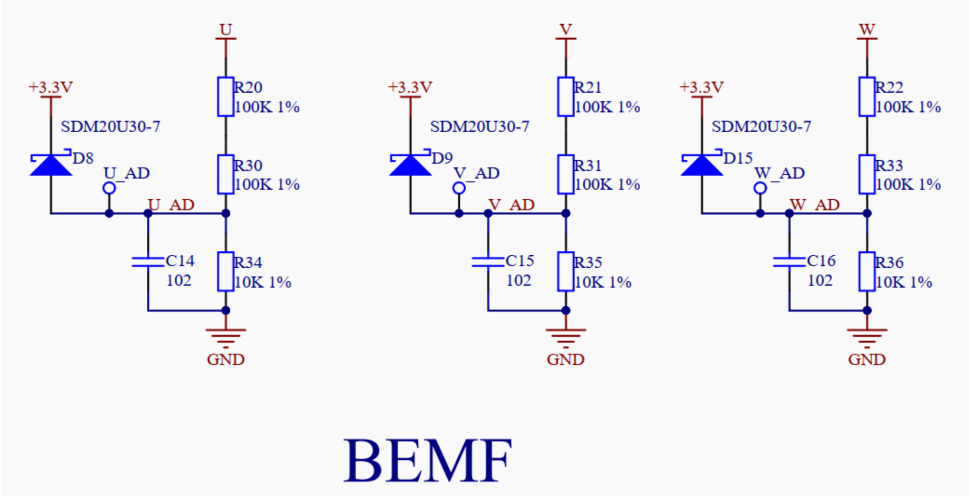


Figure 4 Back Electromotive Force Sampling Circuit

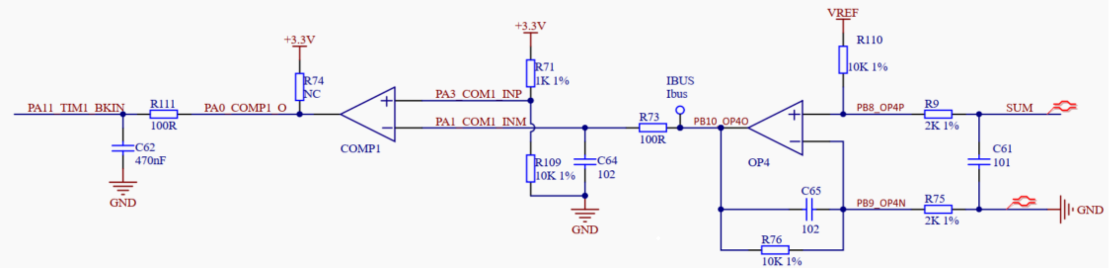


As shown in the figure, the voltage division sampling method is adopted,

Back electromotive force voltage U_AD=U/((100K+100K+10K)/10K)=U/21

2.2.3 Overcurrent protection circuit

Figure 5 Overcurrent Protection Circuit



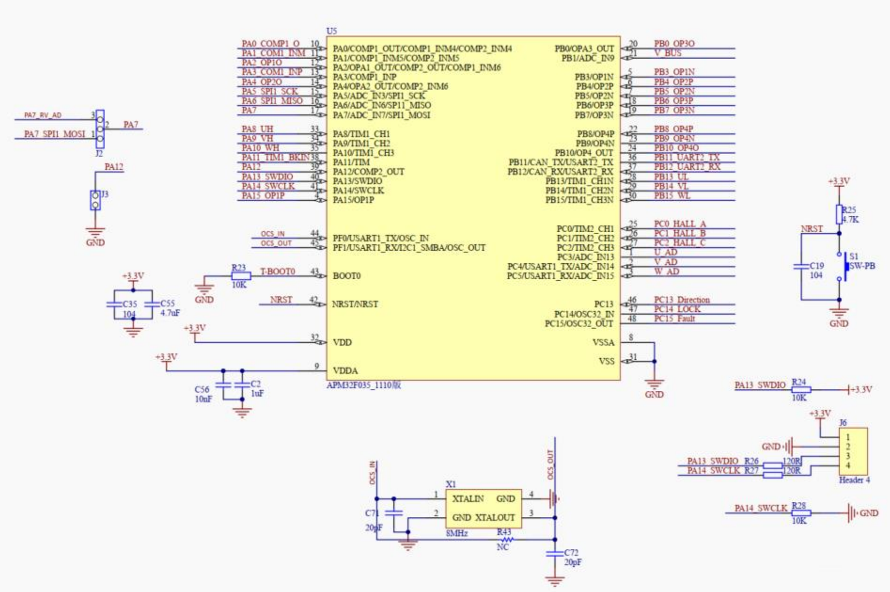
As shown in the figure, a built-in operational amplifier OPA4 is used to sample the bus current. A 12-bit ADC is adopted with a sampling range of 0-3.3V corresponding to 0-4096. From the figure, it can be seen that the sampling resistance is 0.02R,

the output end of OPA4 is used as the reverse input end of COMP1, and resistance voltage division is adopted at the forward input end. Through simple calculation, it can be concluded that when the input is 3V,

the maximum current corresponding to 3V is (3-1.65)/5/0.02=13.5A

2.2.4 Minimum system circuit

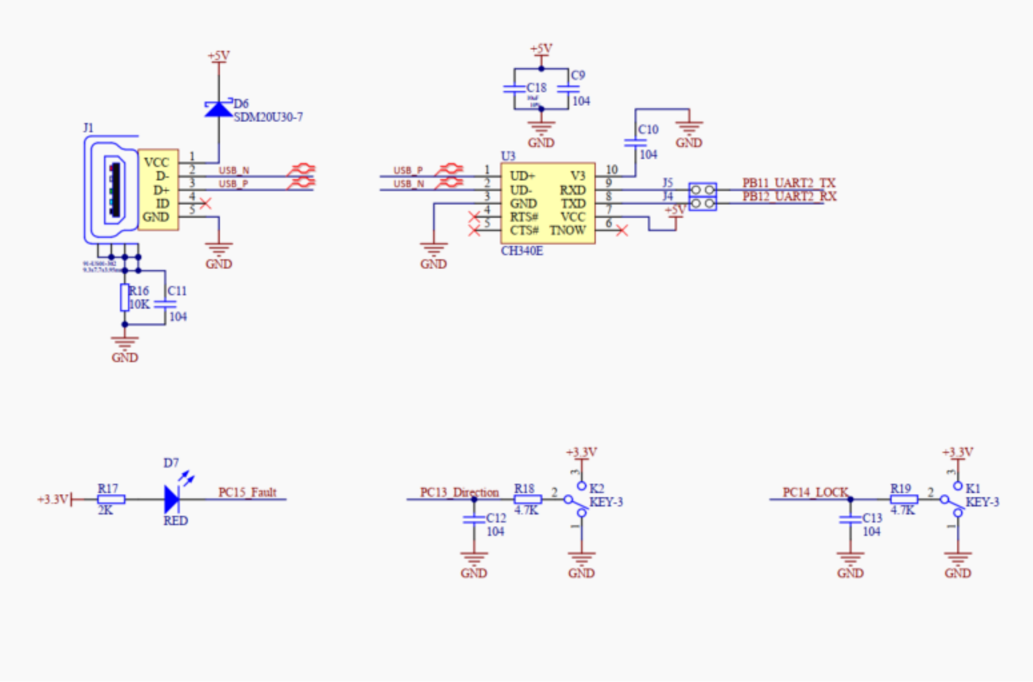
Figure 6 Minimum System Circuit



As shown in the figure, the utilization of APM32F035 MOTOR EVAL V1.0 board hardware interface resources is described in the above figure. The external crystal oscillator input of HSE is 8MHz, and SWD burning interface is adopted for burning.

2.2.5 Communication Interface and Button Circuit

Figure 7 Communication Interface and Button Circuit



As shown in the figure, a USB-to-serial port and a fault indicator light are reserved in the APM32F035 MOTOR EVAL V1.0 board hardware for debugging by developers; the two buttons are responsible for implementing the functions of controlling the running direction of the motor and locking.

2.3 Physical System Hardware

The picture of the system is shown in the figure, and it mainly includes the following four interfaces:

(1) Power input interface (connect to 24V; pay attention to positive and negative poles)

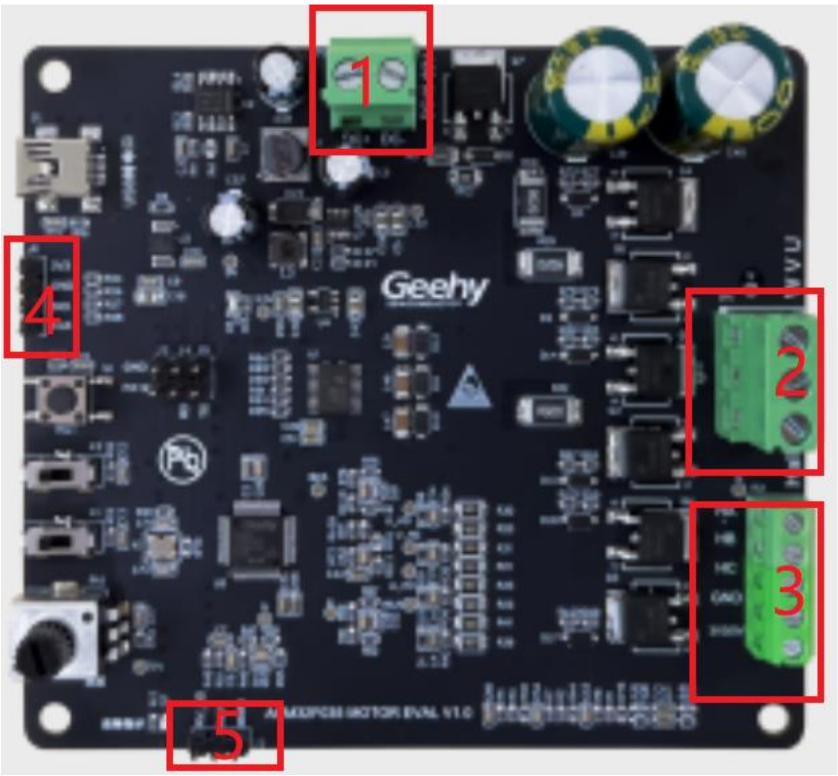
(2) Three phase motor interface (phase sequence only affects the direction of rotation)

(3) HALL input interface

(4) SWD debugging interface

(5) The jumper cap port needs to be connected

Figure 8 Hardware Picture



3 Software Introduction

3.1 Overall Program Architecture

The overall code architecture of this project can be divided into four layers: user layer, peripheral driver layer, motor control driver layer, and motor algorithm layer. The specific functional descriptions are as follows:

3.1.1 USER Layer

main.c: The main function entry is responsible for switching of motor initialization parameters, underlying peripherals, interrupt priority, while cycle, and low-speed state machine loop;

apm32f035_int.c: All interrupt handling functions, mainly including TMR1 interrupt function and ADC interrupt handler function;

user_function.c: Includes initialization configuration, parameter reset, and other handler functions of motor parameters;

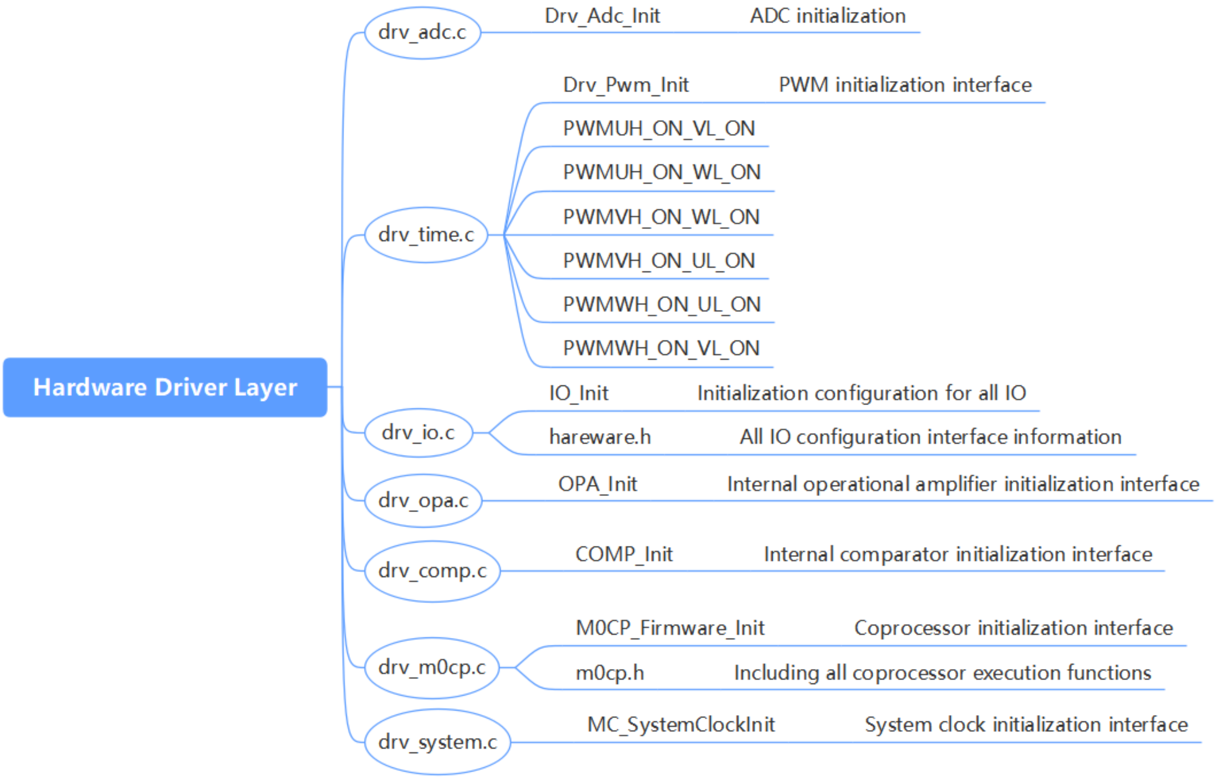
parameter.h: Includes all required configuration parameter information;

board.c: Includes initialization configuration functions of board-level underlying peripheral.

3.1.2 Peripheral Driver Layer (HARDWARE Layer)

The peripheral driver layer is mainly responsible for the peripheral driver functions and configuration of the APM32F035 chip, mainly including GPIO, PWM, ADC, OPA, COMP and M0CP coprocessors, as shown in the following figure.

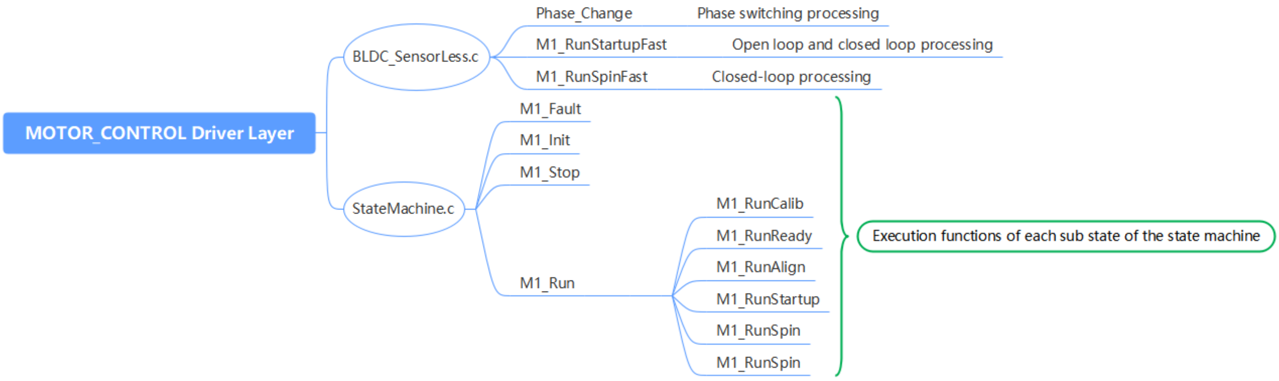
Figure 9 Peripheral Driver Layer



3.1.3 Motor Control Drive Layer (MOTOR_CONTROL Layer)

The motor control driver layer is mainly responsible for the control run logic and core processing algorithm call of the motor, as shown in the following figure.

Figure 10 Motor Control Driver Layer



3.1.4 Geehy Motor Algorithm Layer (Geehy_MCLIB Layer)

The motor algorithm layer includes zero crossing point detection function, math library and other library functions.

3.2 Introduction to State Machine

In this case, the structure of embedding the sub-state machine into the main state machine is adopted, as shown below:

Four main states: INIT, STOP, FAIL, and RUN;

The six RUN sub-states of the main state are **run calib, run-ready, run-align, run-startup, run-spin, and run-freewheel**.

The main state machine is described below:

Fault: When an error occurs in the system, it will remain in this state until the error flag bit is cleared;

Then after delay for a period of time, it will jump from the Fault state to the STOP state and wait for the start command

Init: This main state executes variable initialization;

Stop: The system waits for the speed command after completing initialization. In this state, the PWM output is turned off;

Run: When the system is in a running state and there is a Stop command or a Fault exception, the system will stop running;

When the main state machine is in the Run state, the Run sub-state machine will be called, as described below.

Calib: After this state is executed, the system will switch to the Ready state and disable the PWM output;

Ready: Clear the closed-loop flag bit, determine whether the speed instruction is greater than the set range, and if the set range is reached switch to the Align state;

Align: Execute the call pre-positioning, adjust the startup duty cycle, adjust the rotor position, execute the state within the specified time, and the system will switch to the Startup sub-state;

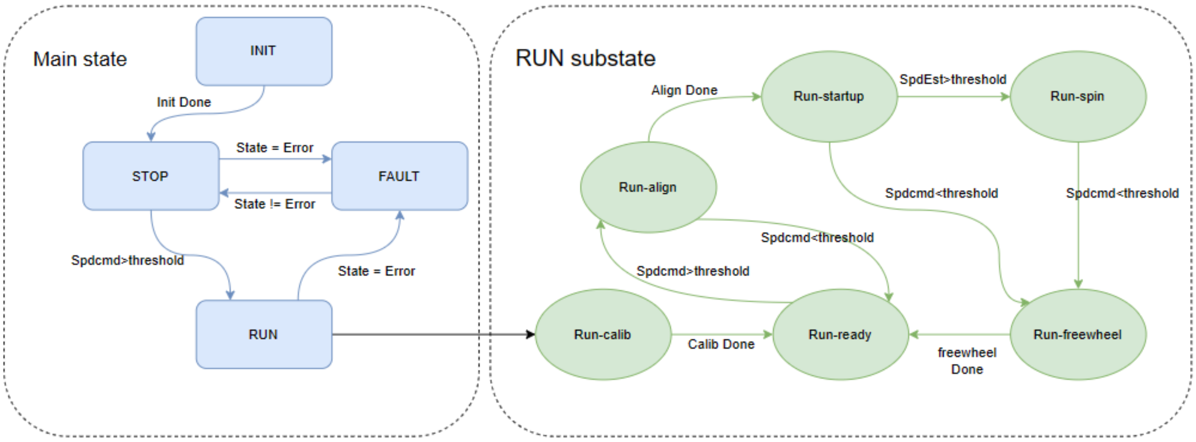
Startup: Start the motor with an open loop and call the zero crossing point algorithm of the back electromotive force to confirm the rotor position. If the motor is started successfully, the system will spin the sub-state;

Spin: Call the zero crossing point algorithm of the back electromotive force to estimate the rotor speed and position, update the PWM, and the motor starts to switch to the closed-loop operation;

Freewheel: Enable PWM output and stop the machine through shorting the brake. Due to rotor inertia, the state can be switched only after the motor stops running and further switched to Ready state. If an error occurs, the system will enter the Fault state.

To sum up, the state machine flowchart of the system is shown in the figure below.

Figure 11 State Machine Flowchart



3.3 Top-layer Peripheral Configuration

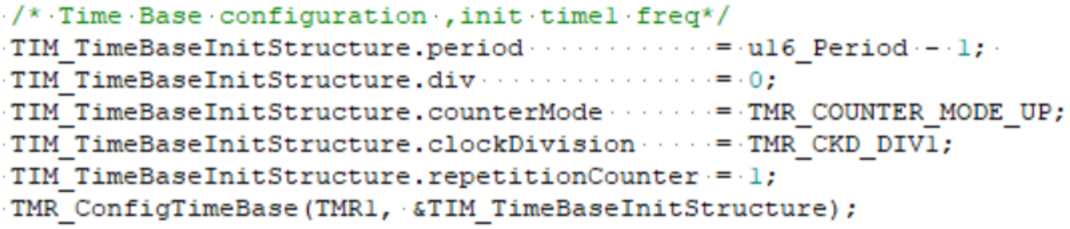
3.3.1 PWM Output Configuration

void Drv_Pwm_Init(uint16_t u16_Period,uint16_t u16_DeadTime)

(1) The general configuration of PWM is as follows:

Set the PWM clock frequency division to 1, select the count-up mode, and set the repeat counter to 1, as shown in the figure below.

Figure 12 General Configuration of PWM



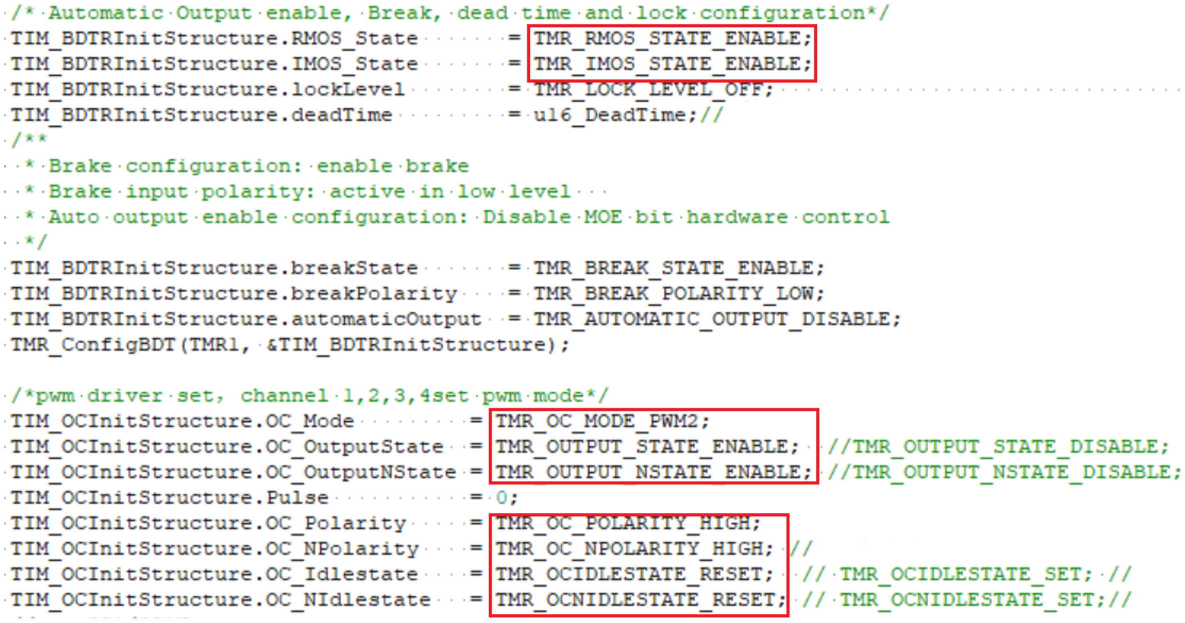
```
 /* Time Base configuration ,init timel freq*/
 TIM_TimeBaseInitStructure.period            = u16_Period - 1;
 TIM_TimeBaseInitStructure.div               = 0;
 TIM_TimeBaseInitStructure.counterMode       = TMR_COUNTER_MODE_UP;
 TIM_TimeBaseInitStructure.clockDivision     = TMR_CKD_DIV1;
 TIM_TimeBaseInitStructure.repetitionCounter = 1;
 TMR_ConfigTimeBase(TMR1, &TIM_TimeBaseInitStructure);
```

(2) PWM Output Status Configuration

Set the output status of upper and lower tubes of PWM and enable the configuration of PWM output of the upper and lower tubes to be effective,

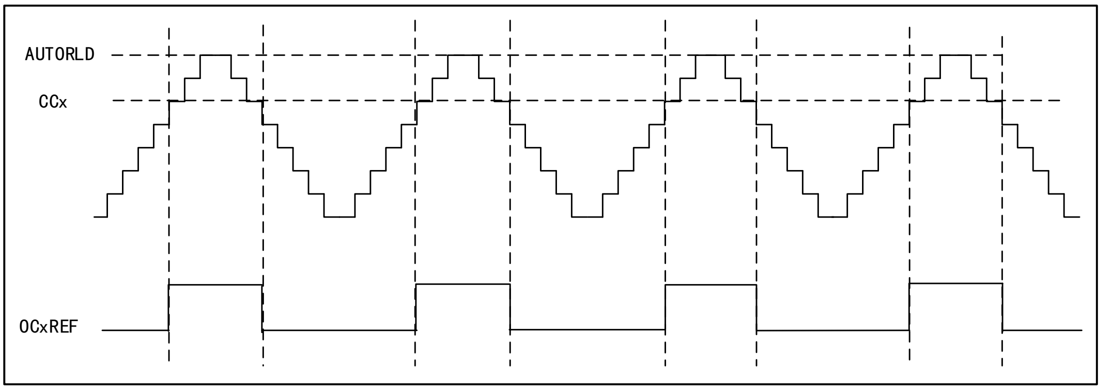
Configure the enabled brakes, configure the brake input polarity, and disable automatic recovery of brake hardware;

Figure 13 PWM Output Status Configuration



```
/* Automatic Output enable, Break, dead time and lock configuration*/
TIM_BDTRInitStructure.RMOS_State        = TMR_RMOS_STATE_ENABLE;
TIM_BDTRInitStructure.IMOS_State        = TMR_IMOS_STATE_ENABLE;
TIM_BDTRInitStructure.lockLevel         = TMR_LOCK_LEVEL_OFF;
TIM_BDTRInitStructure.deadTime          = u16_DeadTime;//
/**
  * Brake configuration: enable brake
  * Brake input polarity: active in low level
  * Auto output enable configuration: Disable MOE bit hardware control
  */
TIM_BDTRInitStructure.breakState        = TMR_BREAK_STATE_ENABLE;
TIM_BDTRInitStructure.breakPolarity     = TMR_BREAK_POLARITY_LOW;
TIM_BDTRInitStructure.automaticOutput   = TMR_AUTOMATIC_OUTPUT_DISABLE;
TMR_ConfigBDT(TMR1, &TIM_BDTRInitStructure);

/*pwm driver set, channel 1,2,3,4set pwm mode*/
TIM_OCInitStructure.OC_Mode          = TMR_OC_MODE_PWM2;
TIM_OCInitStructure.OC_OutputState   = TMR_OUTPUT_STATE_ENABLE;   //TMR_OUTPUT_STATE_DISABLE;
TIM_OCInitStructure.OC_OutputNState  = TMR_OUTPUT_NSTATE_ENABLE;  //TMR_OUTPUT_NSTATE_DISABLE;
TIM_OCInitStructure.Pulse            = 0;
TIM_OCInitStructure.OC_Polarity      = TMR_OC_POLARITY_HIGH;
TIM_OCInitStructure.OC_NPolarity     = TMR_OC_NPOLARITY_HIGH; //
TIM_OCInitStructure.OC_Idlestate     = TMR_OCIDLESTATE_RESET;  // TMR_OCIDLESTATE_SET; //
TIM_OCInitStructure.OC_NIdlestate    = TMR_OCNIDLESTATE_RESET; // TMR_OCNIDLESTATE_SET;//
```

Figure 14 Timing Diagram of PWM2 Center-aligned Mode



In count-up mode, when TMR1_CNT<TMR1_CCR1, Channel 1 is invalid level; otherwise it is valid level;
In count-down mode, when TMR1_CNT>TMR1_CCR1, Channel 1 is valid level; otherwise it is invalid level.

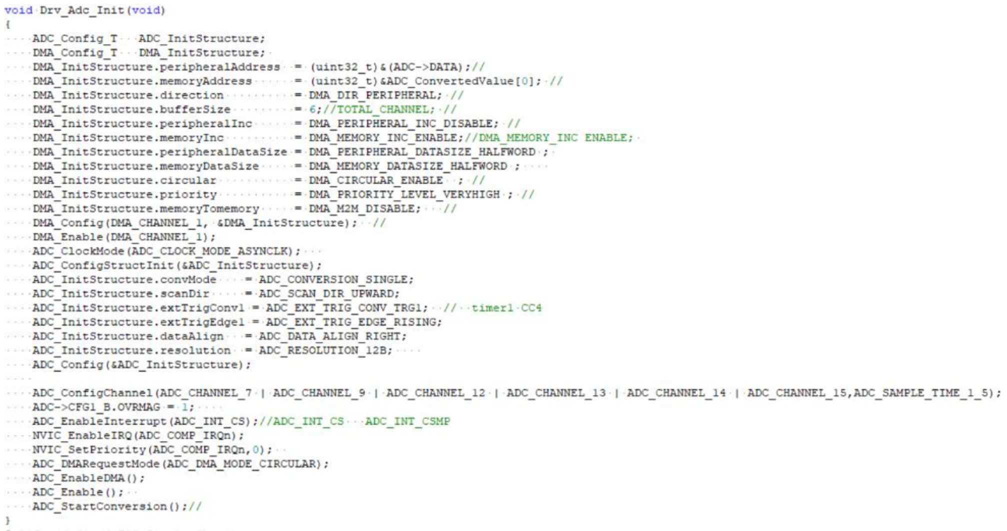
3.3.2 ADC Configuration

void Drv_Adc_Init(void)

(1) ADC underlying configuration

DMA mode is adopted, and the quantized data of ADC is directly transported to the ADC_ConvertedValue array for storage. The ADC trigger condition uses CC4 of TMR1 as the trigger source, to enable ADC and configure ADC interrupt priority and its enable. Details are shown below:

Figure 15 ADC Underlying Configuration



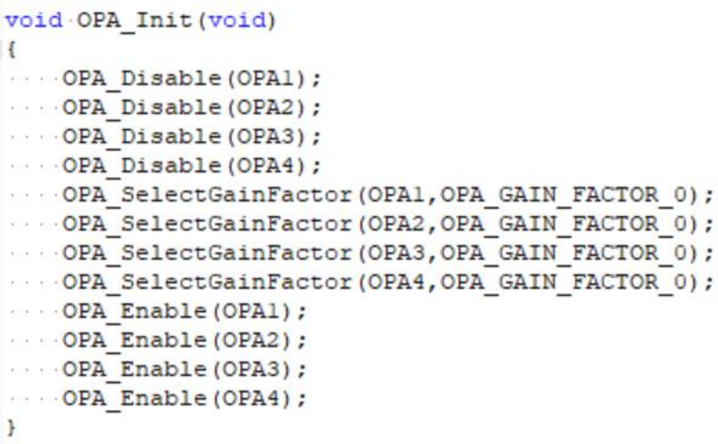
```
void Drv_Adc_Init(void)
{
    ADC_Config_T   ADC_InitStructure;
    DMA_Config_T   DMA_InitStructure;
    DMA_InitStructure.peripheralAddress  = (uint32_t)&(ADC->DATA);//
    DMA_InitStructure.memoryAddress      = (uint32_t)&ADC_ConvertedValue[0]; //
    DMA_InitStructure.direction          = DMA_DIR_PERIPHERAL; //
    DMA_InitStructure.bufferSize         = 6;//TOTAL_CHANNEL; //
    DMA_InitStructure.peripheralInc      = DMA_PERIPHERAL_INC_DISABLE; //
    DMA_InitStructure.memoryInc          = DMA_MEMORY_INC_ENABLE;//DMA_MEMORY_INC ENABLE;
    DMA_InitStructure.peripheralDataSize = DMA_PERIPHERAL_DATASIZE_HALFWORD ;
    DMA_InitStructure.memoryDataSize     = DMA_MEMORY_DATASIZE_HALFWORD ;
    DMA_InitStructure.circular           = DMA_CIRCULAR_ENABLE  ; //
    DMA_InitStructure.priority           = DMA_PRIORITY_LEVEL_VERYHIGH ; //
    DMA_InitStructure.memoryTomemory     = DMA_M2M_DISABLE;   //
    DMA_Config(DMA_CHANNEL_1, &DMA_InitStructure);  //
    DMA_Enable(DMA_CHANNEL_1);
    ADC_ClockMode(ADC_CLOCK_MODE_ASYNCLK);
    ADC_ConfigStructInit(&ADC_InitStructure);
    ADC_InitStructure.convMode    = ADC_CONVERSION_SINGLE;
    ADC_InitStructure.scanDir     = ADC_SCAN_DIR_UPWARD;
    ADC_InitStructure.extTrigConv1 = ADC_EXT_TRIG_CONV_TRG1;  //  timer1 CC4
    ADC_InitStructure.extTrigEdge1 = ADC_EXT_TRIG_EDGE_RISING;
    ADC_InitStructure.dataAlign   = ADC_DATA_ALIGN_RIGHT;
    ADC_InitStructure.resolution  = ADC_RESOLUTION_12B;
    ADC_Config(&ADC_InitStructure);

    ADC_ConfigChannel(ADC_CHANNEL_7 | ADC_CHANNEL_9 | ADC_CHANNEL_12 | ADC_CHANNEL_13 | ADC_CHANNEL_14 | ADC_CHANNEL_15,ADC_SAMPLE_TIME_1_5);
    ADC->CFG1_B.OVRMAG = 1;
    ADC_EnableInterrupt(ADC_INT_CS);//ADC_INT_CS   ADC_INT_CSMP
    NVIC_EnableIRQ(ADC_COMP_IRQn);
    NVIC_SetPriority(ADC_COMP_IRQn,0);
    ADC_DMARequestMode(ADC_DMA_MODE_CIRCULAR);
    ADC_EnableDMA();
    ADC_Enable();
    ADC_StartConversion();//
}
```

3.3.3 OPA and COMP Underlying Configuration

(1) OPA underlying configuration

To configure the underlying configuration of OPA, first configure the OPA pin, DISABLE the operational amplifier OPA, configure to use an external resistor network, and then ENABLE it, as shown in the figure below;

Figure 16 OPA Underlying Configuration

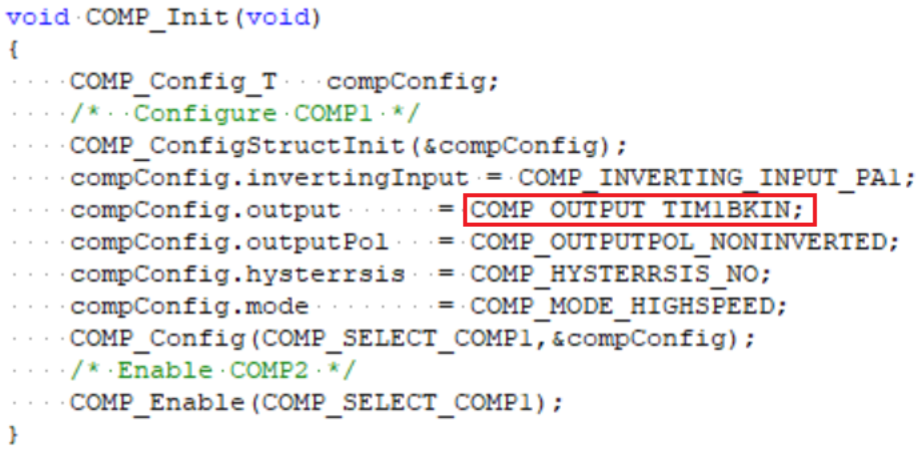


```
void OPA_Init(void)
{
    OPA_Disable(OPA1);
    OPA_Disable(OPA2);
    OPA_Disable(OPA3);
    OPA_Disable(OPA4);
    OPA_SelectGainFactor(OPA1,OPA_GAIN_FACTOR_0);
    OPA_SelectGainFactor(OPA2,OPA_GAIN_FACTOR_0);
    OPA_SelectGainFactor(OPA3,OPA_GAIN_FACTOR_0);
    OPA_SelectGainFactor(OPA4,OPA_GAIN_FACTOR_0);
    OPA_Enable(OPA1);
    OPA_Enable(OPA2);
    OPA_Enable(OPA3);
    OPA_Enable(OPA4);
}
```

(2) COMP underlying configuration

COMP is used for overcurrent anomaly detection. To configure the underlying configuration of COMP, first configure the COMP pin, set the COMP output to the BKIN connected to TMR1, set the output reverse, and trigger the BKIN of TMR1 at a low level, as shown in the following figure;

Figure 17 COMP Underlying Configuration



```
void COMP_Init(void)
{
    COMP_Config_T   compConfig;
    /*  Configure COMP1 */
    COMP_ConfigStructInit(&compConfig);
    compConfig.invertingInput  = COMP_INVERTING_INPUT_PA1;
    compConfig.output          = COMP_OUTPUT_TIM1BKIN;
    compConfig.outputPol       = COMP_OUTPUTPOL_NONINVERTED;
    compConfig.hysterrsis      = COMP_HYSTERRSIS_NO;
    compConfig.mode            = COMP_MODE_HIGHSPEED;
    COMP_Config(COMP_SELECT_COMP1,&compConfig);
    /* Enable COMP2 */
    COMP_Enable(COMP_SELECT_COMP1);
}
```

3.4 Settings of Key Parameters

All parameters in this system are configured in parameter.h of the user layer, mainly including system parameters, related parameters of backplane, related parameters of state machine, and related parameters of motor, as follows:

3.4.1 System Parameters

Table 3 System Parameters

Parameter name	Parameter description	Set value
SYS_REFV	Supply voltage of the system	3.3 (V)
SYSCLK_HSE_72MHz	Main frequency of the system	72000000 (Hz)
PWMFREQ	PWM frequency	8000 (Hz)
DEAD_TIME	PWM dead band time	1.0 (μs)
SLOWLOOP_FREQ	Control frequency of slow loop	1000 (Hz)

3.4.2 Backplane Hardware Parameters

Table 4 Parameters of Backplane Hardware

Parameter name	Parameter description	Set value
ADC_REFV	ADC reference voltage	3.3 (V)
R_SHUNT	Sampling resistance value	0.02 (Ω)
CURRENT_OPA_GAIN	Amplification factor of operational amplifier	5.0

Parameter name	Parameter description	Set value
I_MAX	Current standardization reference value	16.5 (A)
UDC_MAX	Voltage standardization reference value	69.0 (V)
U_MAX	Phase voltage standardization reference value	39.83 (V)

3.4.3 Parameters of State Machine

Table 5 Parameters of State Machine

Parameter name	Parameter description	Set value
FAULTRELEASE_TIME	Fault detection transition time	5 (s)
ALIGN_PWMVALUE	PWM duty in Align state	460
ALIGN_TIME	Align state time	0.1 (s)
FREEWHEEL_SPEED	Stop after the speed command is below the threshold	300 (rpm)
STARTUPSPEED_RPM	Rated speed of open-loop rotation	300 (rpm)
STARTUP_SPEED_RAMP	Slope value of speed command under open loop	500 (rpm/s)
STARTUP_TIME	Startup open-loop time	1.0 (s)

3.4.4 Motor Related Parameters

Table 6 Motor Related Parameters

Parameter name	Parameter description	Set value
POLEPAIRS	Number of motor pole-pairs	2 (unit)
SPEED_MAX	Speed calibration value	5000 (rpm)
MAX_DUTY	Maximum PWM duty	9000
RAMP_UP	Speed rising slope	1000 (rpm/s)
RAMP_DOWN	Speed descending slope	1000 (rpm/s)
M1_SPEED_KP_Q10	Speed loop KP parameter Q10 format	2048
M1_SPEED_KI_Q10	Speed loop KI parameter Q10 format	300

3.5 Debugging method

(1) Check hardware connections: Check if the connections between the motor, motor driver and controller are correct, and if the power supply is stable, and ensure that all interfaces are inserted and tightened and free of damage or short circuits.

(2) Configuration parameters: Configure the parameters of the controller according to the specifications of the motor and driver, such as the rated current of the motor, the motor parameters, the number of pole pairs of the motor, and the maximum speed of the motor. Ensure that all parameters are set correctly.

(3) Perform senseless calibration: The senseless square wave needs to measure the back electromotive force of the motor to confirm the zero crossing point and calculate the position and speed of the motor on the basis of it. Conducting senseless calibration can help the system measure the back electromotive force. In the actual debugging process, the waveform of the back electromotive force voltage can be confirmed through an oscilloscope.

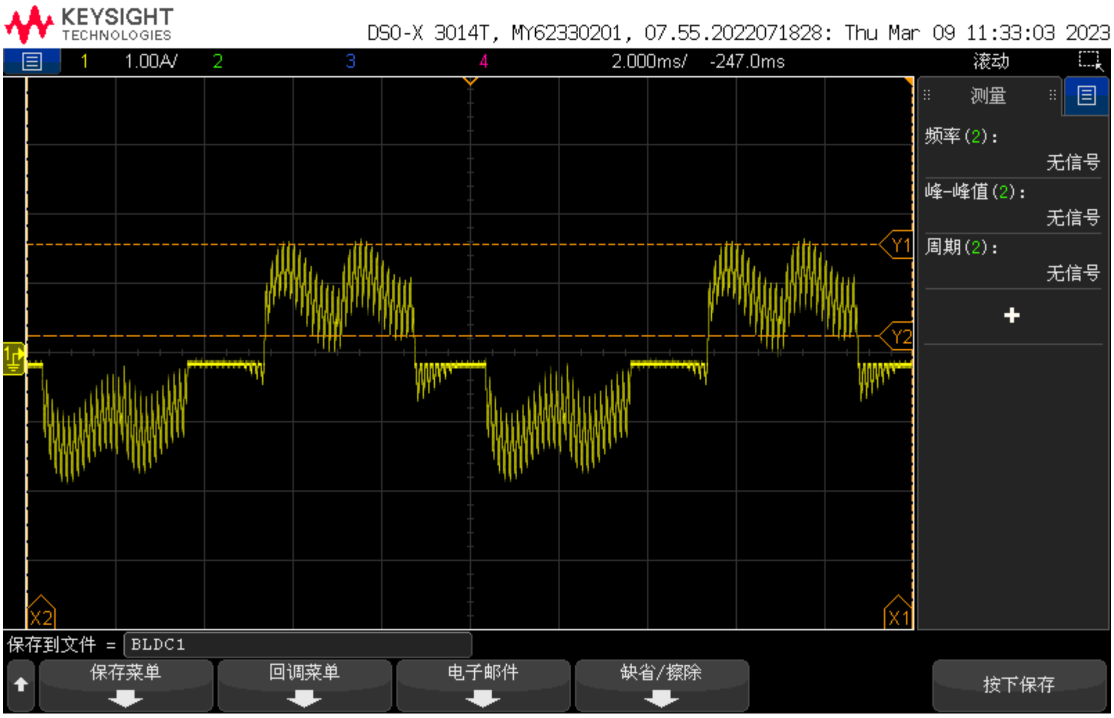
(4) Debugging of motor operation: First adopt the open-loop debugging method and check whether the operation sequence of the motor is correct. If it is not consistent, it can be modified in the BLDC_SensorLess.h file. Then check whether the open-loop operation of the motor is normal. An oscilloscope can be used to observe the output waveform and check whether they live up to the expectation (in this process, the motor can be modified and debugged by adjusting the open-loop current and setting the rated open-loop speed); confirm that the closed-loop debugging is conducted after the open-loop operation is normal.

(5) Conduct speed PID debugging: Use a PID regulator to control the response speed of the motor. The response and stability of the motor can be optimized by changing the PID parameters. During debugging, the experimental results should be recorded for future reference.

(6) Conduct performance tests: After the above steps are completed, some performance tests can be conducted, such as measuring the maximum speed, maximum torque, and efficiency of the motor. Some testing equipment can be used for measurement, such as tachometer, load tester, and power meter.

4 Actual test waveform

Figure 18 Actual Test Waveform



DSO-X 3014T, MY62330201, 07.55.2022071828: Thu Mar 09 11:33:03 2023

频率(2):
无信号
峰-峰值(2):
无信号
周期(2):
无信号

保存到文件 = BLDC1

5 Revision History

Table 7 Document Revision History

Date	Revision	Revision History
July 26, 2023	1.0	New
August 14, 2023	1.1	（1） Modified the production information form （2） Modified the format

Statement

This document is formulated and published by Geehy Semiconductor Co., Ltd. (hereinafter referred to as "Geehy"). The contents in this document are protected by laws and regulations of trademark, copyright and software copyright. Geehy reserves the right to make corrections and modifications to this document at any time. Please read this document carefully before using Geehy products. Once you use the Geehy product, it means that you (hereinafter referred to as the "users") have known and accepted all the contents of this document. Users shall use the Geehy product in accordance with relevant laws and regulations and the requirements of this document.

1. Ownership

This document can only be used in connection with the corresponding chip products or software products provided by Geehy. Without the prior permission of Geehy, no unit or individual may copy, transcribe, modify, edit or disseminate all or part of the contents of this document for any reason or in any form.

The "极海" or "Geehy" words or graphics with "®" or "TM" in this document are trademarks of Geehy. Other product or service names displayed on Geehy products are the property of their respective owners.

2. No Intellectual Property License

Geehy owns all rights, ownership and intellectual property rights involved in this document.

Geehy shall not be deemed to grant the license or right of any intellectual property to users explicitly or implicitly due to the sale or distribution of Geehy products or this document.

If any third party's products, services or intellectual property are involved in this document, it shall not be deemed that Geehy authorizes users to use the aforesaid third party's products, services or intellectual property, unless otherwise agreed in sales order or sales contract.

3. Version Update

Users can obtain the latest document of the corresponding models when ordering Geehy products.

If the contents in this document are inconsistent with Geehy products, the agreement in thesales order or the sales contract shall prevail.

4. Information Reliability

The relevant data in this document are obtained from batch test by Geehy Laboratory or cooperative third-party testing organization. However, clerical errors in correction or errors caused by differences in testing environment may occur inevitably. Therefore, users should understand that Geehy does not bear any responsibility for such errors that may occur in this document. The relevant data in this document are only used to guide users as performance parameter reference and do not constitute Geehy's guarantee for any product performance.

Users shall select appropriate Geehy products according to their own needs, and effectively verify and test the applicability of Geehy products to confirm that Geehy products meet their own needs, corresponding standards, safety or other reliability requirements. If loses are caused to users due to the user's failure to fully verify and test Geehy products, Geehy will not bear any responsibility.

5. Legality

USERS SHALL ABIDE BY ALL APPLICABLE LOCAL LAWS AND REGULATIONS WHEN USING THIS DOCUMENT AND THE MATCHING GEEHY PRODUCTS. USERS SHALL UNDERSTAND THAT THE PRODUCTS MAY BE RESTRICTED BY THE EXPORT, RE-EXPORT OR OTHER LAWS OF THE COUNTIRIES OF THE PRODUCTS SUPPLIERS, GEEHY, GEEHY DISTRIBUTORS AND USERS. USERS (ON BEHALF OR ITSELF, SUBSIDIARIES AND AFFILIATED ENTERPRISES) SHALL AGREE AND PROMISE TO ABIDE BY ALL APPLICABLE LAWS AND REGULATIONS ON THE EXPORT AND RE-EXPORT OF GEEHY PRODUCTS AND/OR TECHNOLOGIES AND DIRECT PRODUCTS.

6. Disclaimer of Warranty

THIS DOCUMENT IS PROVIDED BY GEEHY "AS IS" AND THERE IS NO WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, TO THE EXTENT PERMITTED BY APPLICABLE LAW.

GEEHY WILL BEAR NO RESPONSIBILITY FOR ANY DISPUTES ARISING FROM THE SUBSEQUENT DESIGN OR USE BY USERS.

7. Limitation of Liability

IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL GEEHY OR ANY OTHER PARTY WHO PROVIDE THE DOCUMENT "AS IS", BE LIABLE FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, DIRECT, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE DOCUMENT (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY USERS OR THIRD PARTIES).

8. Scope of Application

The information in this document replaces the information provided in all previous versions of the document.

Geehy Semiconductor Co.,Ltd.