

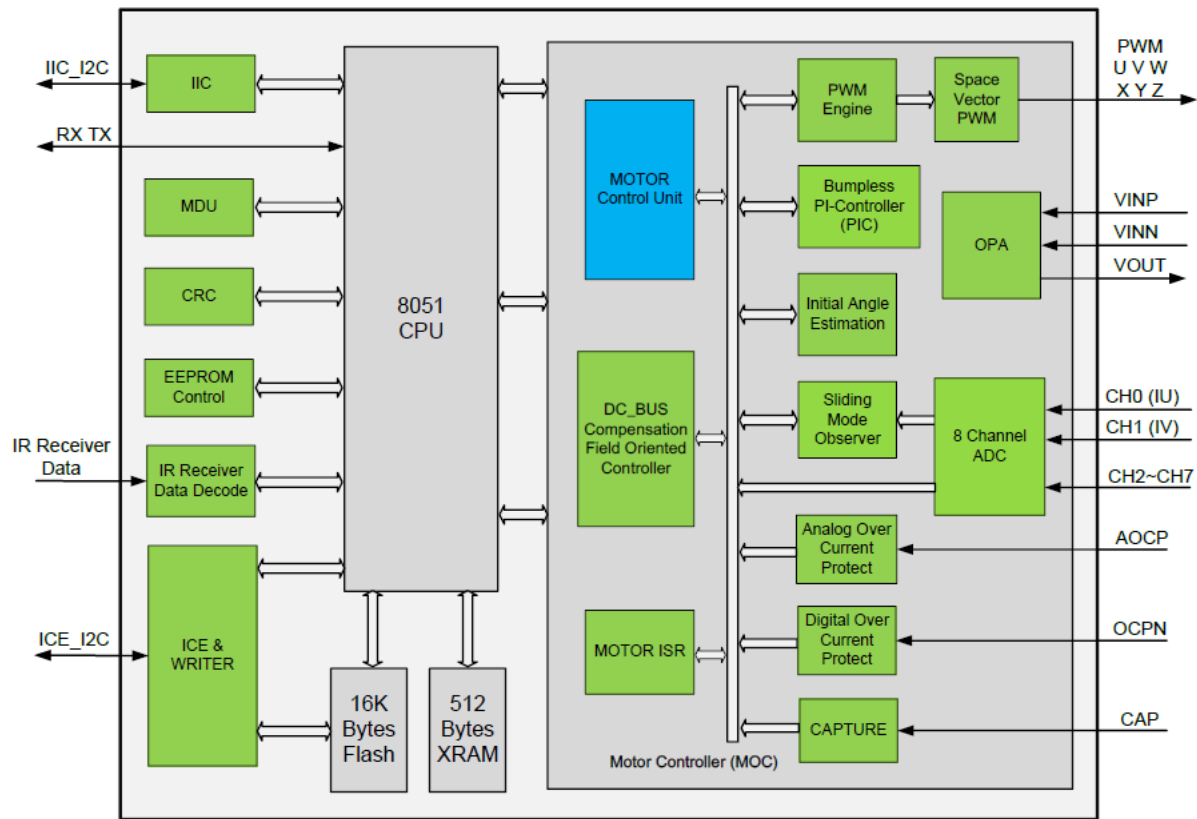
MDSF40

Sample Code Sensorless Introduce

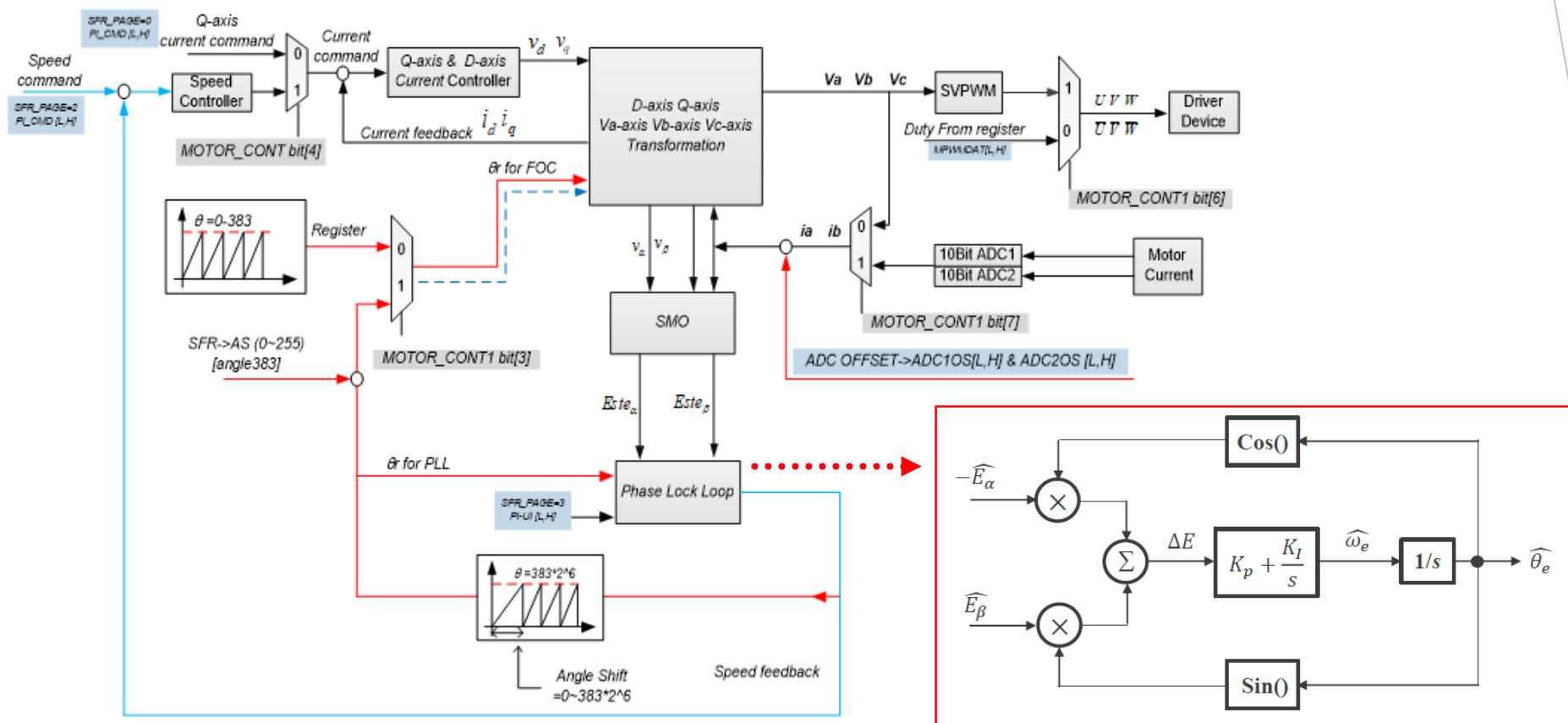


MDSF架构

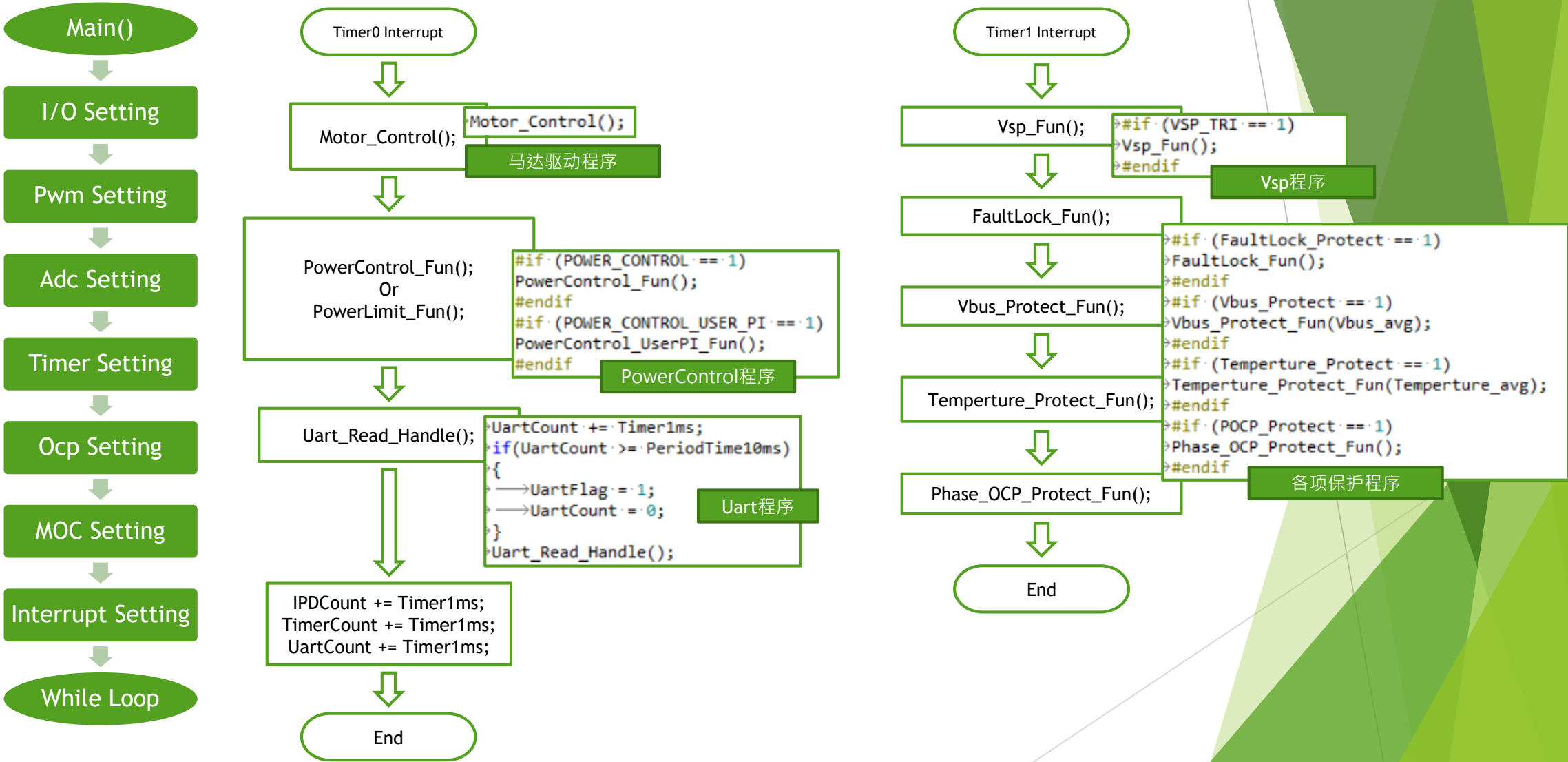
MDSF架构



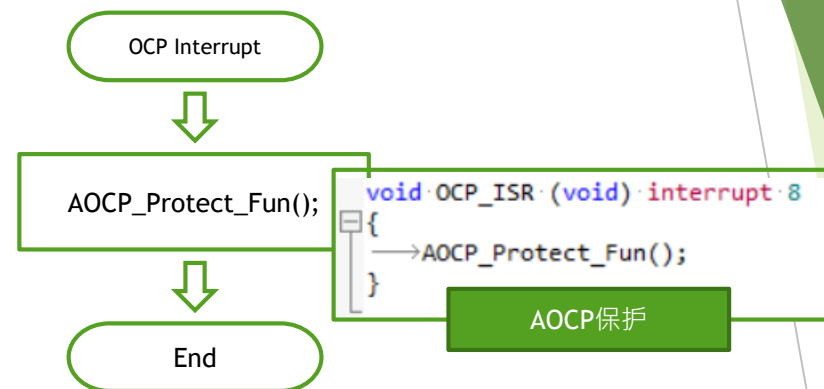
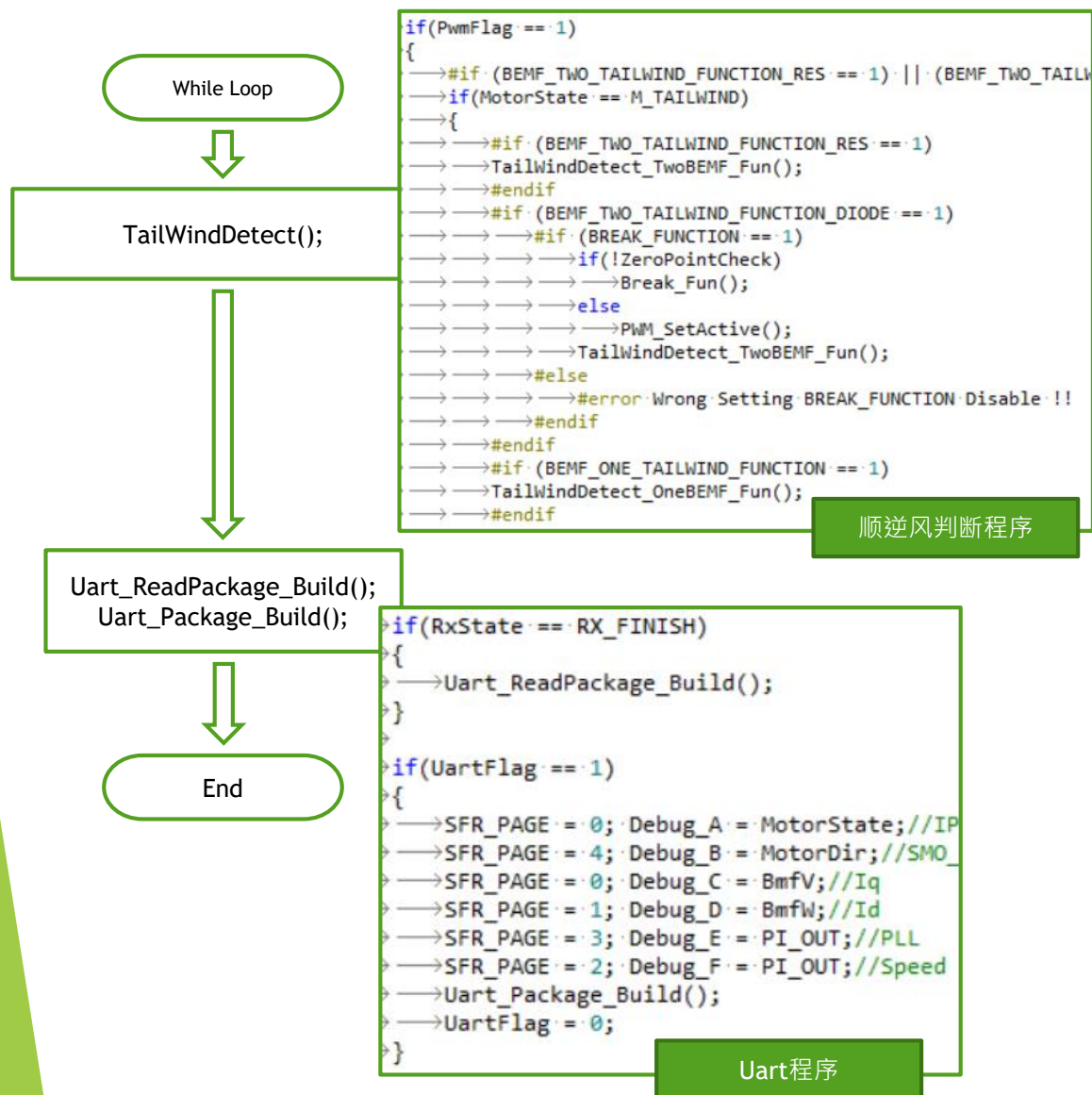
MOC架构



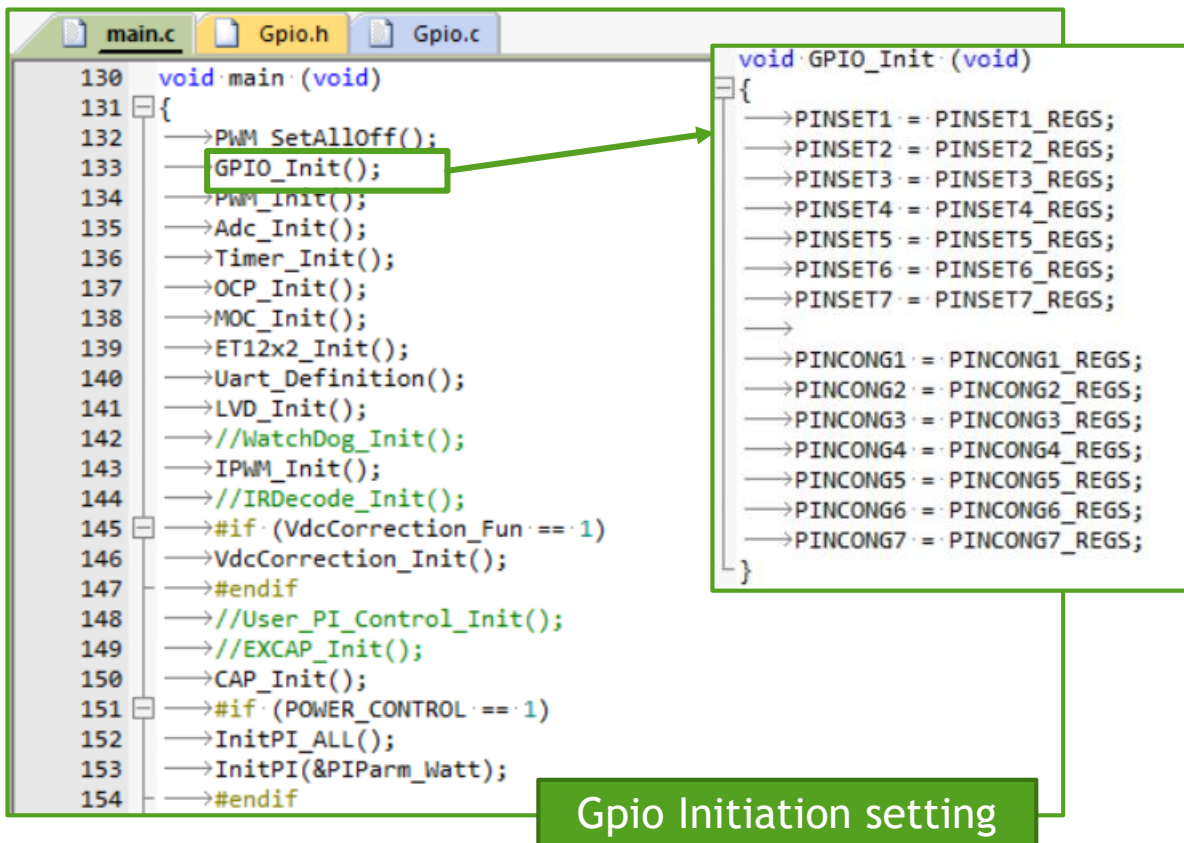
程式流程图



程式流程图



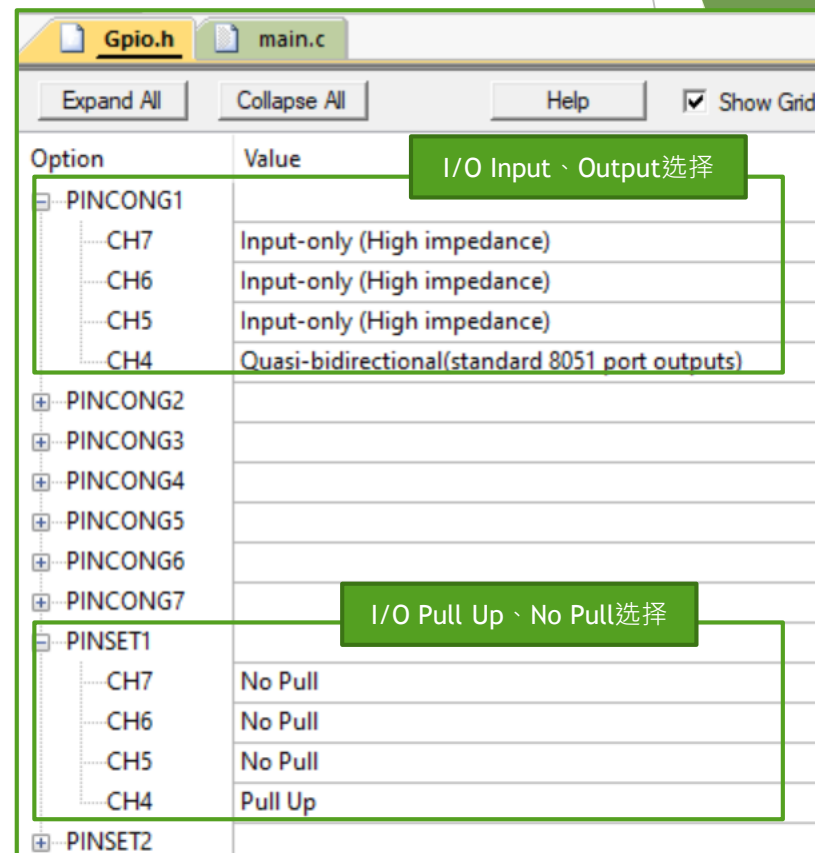
I/O Initiation setting



```
130 void main(void)
131 {
132     →PWM_SetAllOff();
133     →GPIO_Init();
134     →PWM_Init();
135     →Adc_Init();
136     →Timer_Init();
137     →OCP_Init();
138     →MOC_Init();
139     →ET12x2_Init();
140     →Uart_Definition();
141     →LVD_Init();
142     →//WatchDog_Init();
143     →IPWM_Init();
144     →//IRDecode_Init();
145     →#if (VdcCorrection_Fun == 1)
146     →VdcCorrection_Init();
147     →#endif
148     →//User_PI_Control_Init();
149     →//EXCAP_Init();
150     →CAP_Init();
151     →#if (POWER_CONTROL == 1)
152     →InitPI_ALL();
153     →InitPI(&PIParm_Watt);
154     →#endif
}
```

```
void GPIO_Init(void)
{
    →PINSET1 = PINSET1_REGS;
    →PINSET2 = PINSET2_REGS;
    →PINSET3 = PINSET3_REGS;
    →PINSET4 = PINSET4_REGS;
    →PINSET5 = PINSET5_REGS;
    →PINSET6 = PINSET6_REGS;
    →PINSET7 = PINSET7_REGS;
    →
    →PINCONG1 = PINCONG1_REGS;
    →PINCONG2 = PINCONG2_REGS;
    →PINCONG3 = PINCONG3_REGS;
    →PINCONG4 = PINCONG4_REGS;
    →PINCONG5 = PINCONG5_REGS;
    →PINCONG6 = PINCONG6_REGS;
    →PINCONG7 = PINCONG7_REGS;
}
```

Gpio Initiation setting



Option	Value
PINCONG1	
CH7	Input-only (High impedance)
CH6	Input-only (High impedance)
CH5	Input-only (High impedance)
CH4	Quasi-bidirectional(standard 8051 port outputs)
PINCONG2	
PINCONG3	
PINCONG4	
PINCONG5	
PINCONG6	
PINCONG7	
PINSET1	
CH7	No Pull
CH6	No Pull
CH5	No Pull
CH4	Pull Up
PINSET2	

I/O Input、Output选择

I/O Pull Up、No Pull选择



Pwm Initiation setting

```
main.c  Gpio.h  Gpio.c
130 void main(void)
131 {
132     →PWM_SetAllOff();
133     →GPIO_Init();
134     →PWM_Init();
135     →Adc_Init();
136     →Timer_Init();
137     →OCP_Init();
138     →MOC_Init();
139     →ET12x2_Init();
140     →Uart_Definition();
141     →LVD_Init();
142     →//WatchDog_Init();
143     →IPWM_Init();
144     →//IRDecode_Init();
145     →#if (VdcCorrection_Fun == 1)
146     →VdcCorrection_Init();
147     →#endif
148     →//User_PI_Control_Init();
149     →//EXCAP_Init();
150     →CAP_Init();
151     →#if (POWER_CONTROL == 1)
152     →InitPI_ALL();
153     →InitPI(&PIParm_Watt);
154     →#endif
```

```
void PWM_Init(void)
{
    →SFR_PAGE = 0;
    →MPWMDATA = MPWMDATA_REGS;
    →SYNC = 0x55;
    →MPWMINV = MPWMINV_REGS;
    →SYNC = 0x55;
    →MPWMDB = MPWMDB_REGS;
    →SYNC = 0x55;
    →PWM_SetAllOff();
}
```

Pwm Initiation setting

注意PWM SWAP配置，需要设定正确!!

Pwm.h	
Expand All	Collapse All
Help	Show Grid
Option	Value
Set MPWM SWAP	MDSF40
Set MPWMDATA	
Set PWM Frequency (unit : Hz)	27000
Set MPWMINV	
U INV	Non-Inverse
X INV	Non-Inverse
V INV	Non-Inverse
Y INV	Non-Inverse
W INV	Non-Inverse
Z INV	Non-Inverse
Set MPWMDB	
Deadband Time	Deadband Time 1.5us
BASE_RPM (unit : rpm)	32767

BASE_RPM : 代表PLL_OUT 32767对应的实际转速的标么值



Adc Initiation setting

```

main.c  Gpio.h  Gpio.c
130 void main(void)
131 {
132     →PWM_SetAllOff();
133     →GPIO_Init();
134     →PWM_Init();
135     →Adc_Init();
136     →Timer_Init();
137     →OCP_Init();
138     →MOC_Init();
139     →ET12x2_Init();
140     →Uart_Definition();
141     →LVD_Init();
142     →//WatchDog_Init();
143     →IPWM_Init();
144     →//IRDecode_Init();
145     →#if (VdcCorrection_Fun == 1)
146     →VdcCorrection_Init();
147     →#endif
148     →//User_PI_Control_Init();
149     →//EXCAP_Init();
150     →CAP_Init();
151     →#if (POWER_CONTROL == 1)
152     →InitPI_ALL();
153     →InitPI(&PIParm_Watt);
154     →#endif
    
```

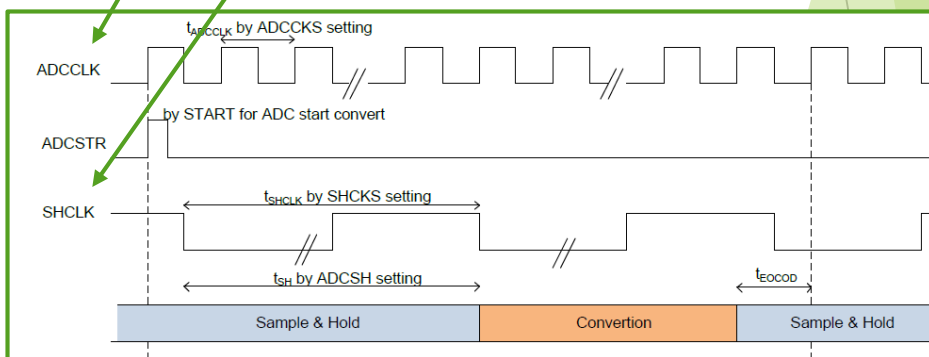
Adc Initiation setting

```

void Adc_Init(void)
{
    →ADCCONT = ADCCONT_REGS;
    →ADCSTR = ADCSTR_REGS | OPA_GAIN_REGS;
    →
    →SFR_PAGE = 0;
    →ADCOFST = 512; //ADCOFST_Init: 512
    →SFR_PAGE = 1;
    →ADCOFST = 512; //ADCOFST_Init: 512;
}
    
```

AdcCKS、SHCKS 设定

Option	Value
ADCCONT	
ADCCH	CH0
ADCCKS	24MHz
ADCDS	ADCD2 LSB
ADCSH	1 clock
ADCPD	Normal
ADCSTR	
SHCKS	6MHz



Timer Initiation setting

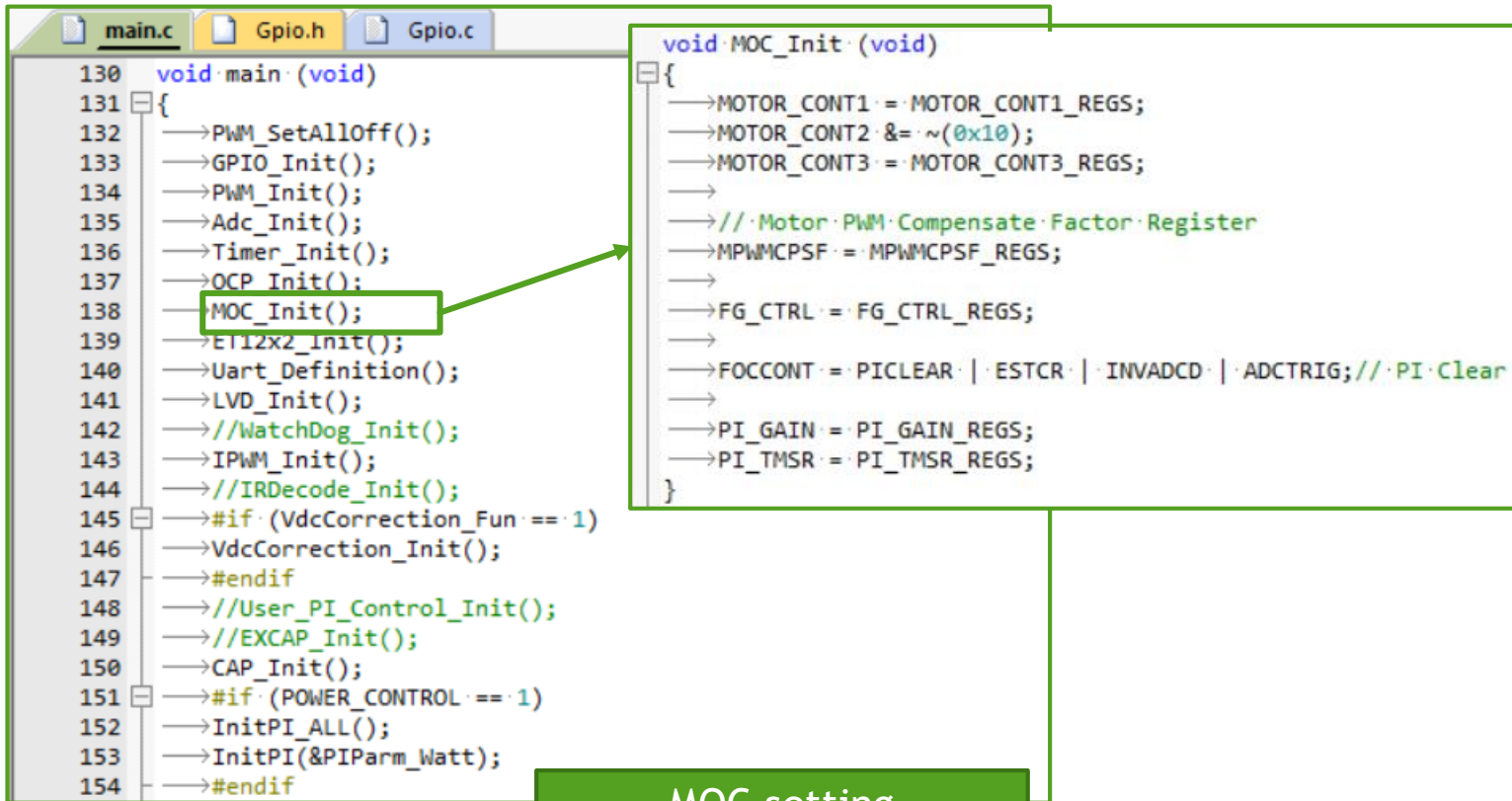
The image displays a code editor on the left and a peripheral configuration tool on the right. In the code editor, the `main.c` file is open, showing a `main` function. A green box highlights the `Timer_Init();` call on line 136. An arrow points from this box to the `Timer_Init` function definition in a separate window. This window shows the function's implementation, which initializes the PFCON, TMOD, and T2CON registers, and enables the timers. A green box highlights the `Timer_Init` function definition. An arrow points from this box to the peripheral configuration tool. The tool shows the configuration for the timer module. The `Timer.h` file is selected. The configuration table shows the settings for the timer module, including the PFCON, TMOD, and T2CON registers. The `Interrupt TIMER0_FREQ (unit : Hz)` is set to 1000, and the `Interrupt TIMER1_FREQ (unit : Hz)` is set to 100. A green box highlights the `Interrupt TIMER0_FREQ (unit : Hz)` setting, with a label "Timer 中断频率设定" (Timer interrupt frequency setting). A red arrow points to the bottom right corner of the configuration tool.

```
130 void main(void)
131 {
132     →PWM_SetAllOff();
133     →GPIO_Init();
134     →PWM_Init();
135     →Adc_Init();
136     →Timer_Init();
137     →OCP_Init();
138     →MOC_Init();
139     →ET12x2_Init();
140     →Uart_Definition();
141     →LVD_Init();
142     →//WatchDog_Init();
143     →IPWM_Init();
144     →//IRDecode_Init();
145     →#if (VdcCorrection_Fun == 1)
146     →VdcCorrection_Init();
147     →#endif
148     →//User_PI_Control_Init();
149     →//EXCAP_Init();
150     →CAP_Init();
151     →#if (POWER_CONTROL == 1)
152     →InitPI_ALL();
153     →InitPI(&PIParm_Watt);
154     →#endif
}
```

```
void Timer_Init(void)
{
    →PFCON = PFCON_REGS;
    →TMOD = TMOD_REGS;
    →TH0 = TIMER0_TH;
    →TL0 = TIMER0_TL;
    →TR0 = TIMER0_ENABLE;
    →
    →TH1 = TIMER1_TH;
    →TL1 = TIMER1_TL;
    →TR1 = TIMER1_ENABLE;
    →
    →T2CON = T2CON_REGS;
    →TH2 = TIMER2_TH;
    →TL2 = TIMER2_TL;
    →TR2 = TIMER2_ENABLE;
}
```

tion	Value
PFCON	
T0PS	F_PER/12 (2MHz)
T1PS	F_PER/12 (2MHz)
SRELPS	F_PER/64
TMOD	
T0 Mode	16-bit Counter/Timer (Not auto-reload)
C/T0	Timer
GATE0	--
T1 Mode	16-bit Counter/Timer (Not auto-reload)
C/T1	Timer
GATE1	--
T2CON	
T2PS	F_PER/12 (2MHz)
T2 Mode	16-bit Counter/Timer (Not auto-reload)
TIMER0	<input checked="" type="checkbox"/>
Interrupt TIMER0_FREQ (unit : Hz)	1000
TIMER1	<input checked="" type="checkbox"/>
Interrupt TIMER1_FREQ (unit : Hz)	100
TIMER2	<input type="checkbox"/>
IT0	1 : External interrupt is activated at falling edge on input pin
IT1	1 : External interrupt is activated at falling edge on input pin

MOC setting



```
main.c  Gpio.h  Gpio.c

130 void main(void)
131 {
132     →PWM_SetAllOff();
133     →GPIO_Init();
134     →PWM_Init();
135     →Adc_Init();
136     →Timer_Init();
137     →OCP_Init();
138     →MOC_Init();
139     →EI12x2_Init();
140     →Uart_Definition();
141     →LVD_Init();
142     →//WatchDog_Init();
143     →IPWM_Init();
144     →//IRDecode_Init();
145     →#if (VdcCorrection_Fun == 1)
146     →VdcCorrection_Init();
147     →#endif
148     →//User_PI_Control_Init();
149     →//EXCAP_Init();
150     →CAP_Init();
151     →#if (POWER_CONTROL == 1)
152     →InitPI_ALL();
153     →InitPI(&PIParm_Watt);
154     →#endif

void MOC_Init(void)
{
    →MOTOR_CONT1 = MOTOR_CONT1_REGS;
    →MOTOR_CONT2 &= ~(0x10);
    →MOTOR_CONT3 = MOTOR_CONT3_REGS;
    →
    →// Motor PWM Compensate Factor Register
    →MPWMCPSF = MPWMCPSF_REGS;
    →
    →FG_CTRL = FG_CTRL_REGS;
    →
    →FOCCONT = PICLEAR | ESTCR | INVADCD | ADCTRIG; // PI Clear
    →
    →PI_GAIN = PI_GAIN_REGS;
    →PI_TMSR = PI_TMSR_REGS;
}
```

MOC setting



Interrupt Setting

Interrupt setting

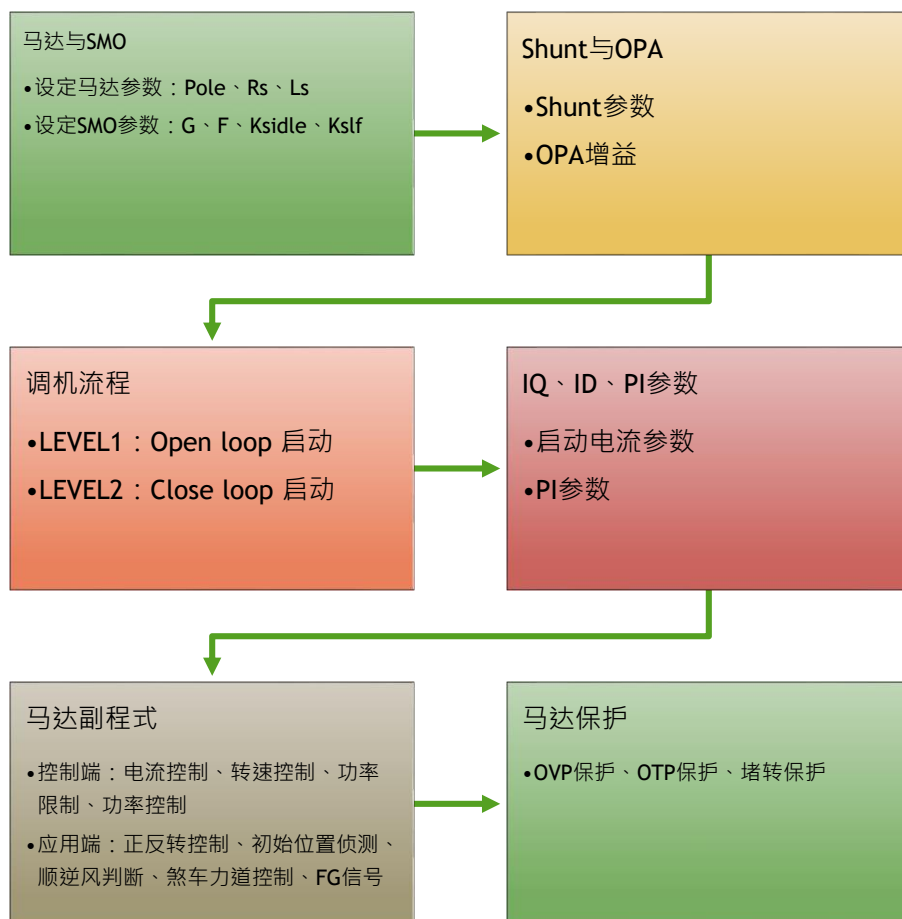
Interrupt setting

```
main.c
75 void main(void){
76     →PWM_SetAllOff();
77     →GPIO_Init();
78     →PWM_Init();
79     →Adc_Init();
80     →Timer_Init();
81     →OCP_Init();
82     →MOC_Init();
83     →Interrupt_Init();
84     →
85     →#if (Uart_Debug == 1)
86     →Uart_Definition();
87     →#endif
88     →
89     →#if (Uart_Debug == 0)
90     →led_a_OUTPUT;
91     →led_b_OUTPUT;
92     →led_c_OUTPUT;
93     →led_a = 0;
94     →led_b = 0;
95     →led_c = 0;
96     →#endif
97     →
98     →//LVD_Init();
99     →WatchDog_Init();
100    →CAP_Init();
101    →
102    →#if (CONTROL_MODE == Power_Control) || (CONTROL_MODE == 2)
103    →InitPI_ALL();
104    →InitPI(&PIParm_Watt);
105    →#endif
```

```
{
→EX0 = 0; //External0_ISR interrupt 0
→ET0 = 1; //Timer0_ISR interrupt 1
→EX1 = 0; //External1_ISR interrupt 2
→ET1 = 1; //Timer1_ISR interrupt 3
→ESP = 1; //Uart_ISR interrupt 4
→ET2 = 0; //Timer2_ISR interrupt 5
→OCPSIE = 1; //OCP_ISR interrupt 8
→ADCIE = 1; //ADC_ISR interrupt 9
→MPWMMINIE = 0; //PwmMin_ISR interrupt 10
→MPWMMAXIE = 1; //PwmMax_ISR interrupt 11
→IICIE = 0; //IIC_ISR interrupt 12
→LVDIIE = 0; //LowVoltage_ISR interrupt 13
→WDIIE = 0; //WatchDog_ISR interrupt 14
→CAPIE = 0; //Cap_ISR interrupt 15
→//IP0 = 0x0C; //Interrupt Priority
→//IP1 = 0x06; //Group 2 > 3 > 1 > 0
→EA = 1; //Allow interrupt
}
```

main.h	
Expand All	Collapse All
Help	
<input checked="" type="checkbox"/> Show Grid	
Option	Value
Group0 - LVDIF IE0	Level_0
Group1 - WDTIF TF0	Level_0
Group2 - OCPSIF ADCIF IE1	Level_3
Group3 - MPWMMINIF MPWMMAXIF TF1	Level_0
Group4 - SPIF(TI, RI)	Level_0
Group5 - CAPIF TF2	Level_0

Motor Control 调机流程 (1)



Motor.h	
Expand All	Collapse All
Help	Show Grid
Option	Value
+ Set motor parameters	
+ Set Rshunt and OPA_Gian	
+ Set the motor tuning process	
+ Set FOC LOOP Parameter	
+ Set motor control program	
+ Set Fairwind and Headwind judgment function	
+ Set motor protection function	
TailWind Determine Time(unit : ms)	200
Stop_Fun Time (unit : ms)	200
+ Set Protection to retry	
+ Error code(MotorErrorState)	



Motor Control 调机流程 (2)

Motor.h	
Expand All Collapse All Help	
Option	Value
[-] Set motor parameters	
Motor_Pole	2
Motor SMO_G	16000
Motor SMO_F	32346
Motor SMO_Kslf	8000
Motor SMO_Z-Corr	32767
Motor SMO Kslide (Sat)	16000
Motor SMO MaxSmcError (Limit)	32767

设定马达极数(必填)。

马达参数不熟悉情况下，SMO参数建议初始设定如下：

SMO参数G：16000

SMO参数F：32000

SMO参数Kslf：8000

SMO参数ZCorr：32767

SMO参数Kslide：32767

SMO参数MaxSmcError：32767

SMO参数SMOGain：327677



Motor Control 调机流程 (4)

OpenLoop

- SmoPLL强制角度启动，需手动衔接闭回路

CloseLoop

- 衔接闭回路时，系统稳定运转，代表整体马达参数正确，之后即可设定LEVEL2

Set the motor tuning process	
FOC_Control_Stage	CloseLoop
Set IQ parking duration(unit : ms)	10
Set SMO_PLL initial speed (unit : 10rpm)	1
Set SMO_PLL end speed (unit : 10rpm)	300
Set PLL accumulation	2
Set SMO_RAMP acceleration slope (unit : m...	1
Set SMO_DELAY Delay time (unit : ms)	10

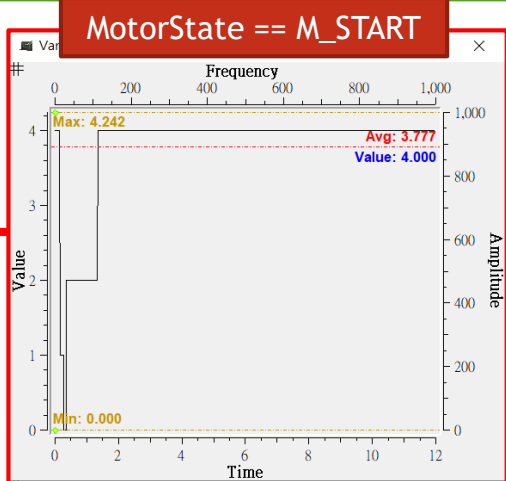
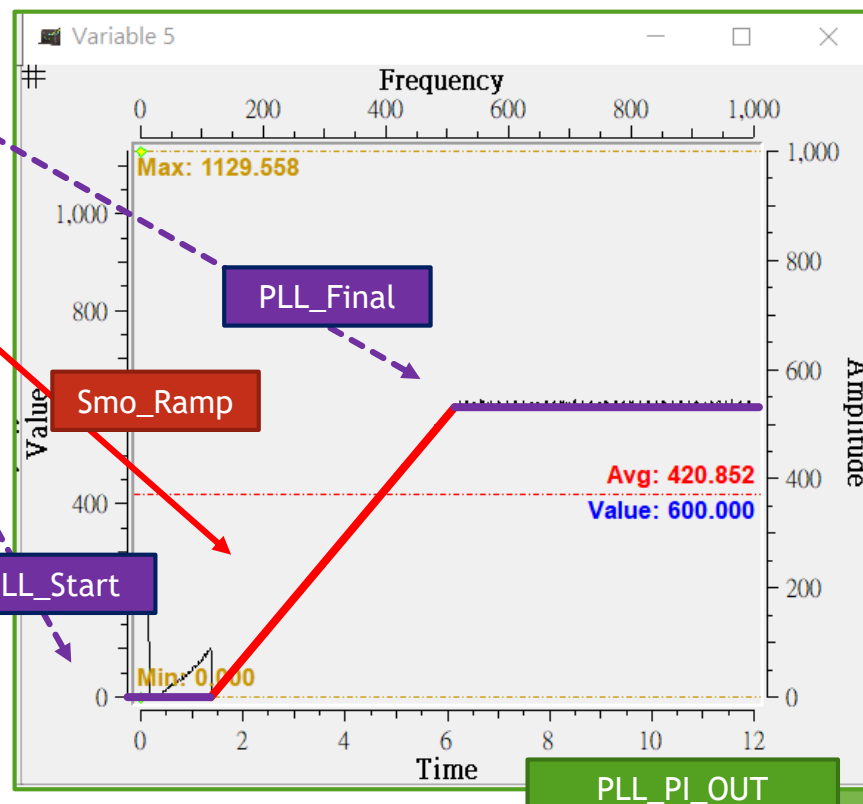
Set FOC LOOP Parameter	
IQ	
Set IQ Current parameter	
Set IQ Initial current (unit : mA)	0
Set IQ Starting current (unit : mA)	440
Set IQ End current (unit : mA)	450
IQ_TailWind Value (unit : mA)	500



Motor Control 调机流程 (5)

Set the motor tuning process		
FOC_Control_Stage	1.	OpenLoop
Set IQ parking duration(unit : ms)	10	
Set SMO_PLL initial speed (unit : 10rpm)	1	2.
Set SMO_PLL end speed (unit : 10rpm)	300	
Set PLL accumulation	2	
Set SMO_RAMP acceleration slope (unit : ms)	1	
Set SMO_DELAY Delay time (unit : ms)	10	
Set FOC LOOP Parameter		
IQ		
Set IQ Current parameter		
Set IQ Initial current (unit : mA)	0	
Set IQ Starting current (unit : mA)	440	
Set IQ End current (unit : mA)	450	
IQ_TailWind Value (unit : mA)	500	

1. 设定调机流程 OpenLoop
2. 设定开回路”启动 - 结束”转速
3. 设定开回路”启动 - 结束”电流

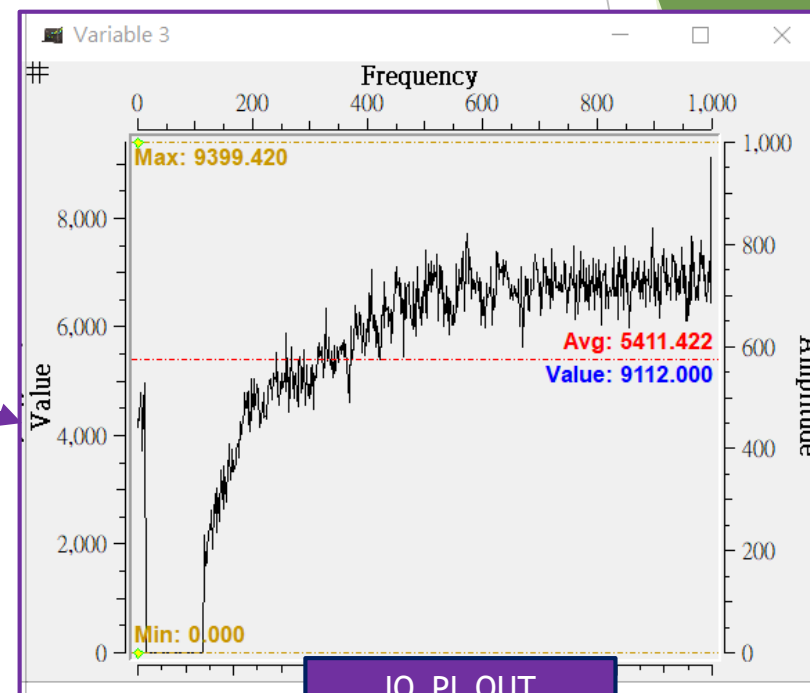


```
enum MotorStatus
{
    →M_OFF = 0,
    →M_INIT = 1,
    →M_TAILWIND = 2,
    →M_IPD = 3,
    →M_START = 4,
    →M_RUN = 5,
    →M_STOP = 6,
    →M_BREAK = 7,
    →M_ERROR = 8,
    →M_BMF_BREAK = 9
};
```

Motor Control 调机流程 (6)

Set the motor tuning process	
FOC_Control_Stage	OpenLoop
Set IQ parking duration(unit : ms)	10
Set SMO_PLL initial speed (unit : 10rpm)	1
Set SMO_PLL end speed (unit : 10rpm)	300
Set PLL accumulation	2
Set SMO_RAMP acceleration slope (unit : ms)	1
Set SMO_DELAY Delay time (unit : ms)	10
Set FOC LOOP Parameter	
IQ	
Set IQ Current parameter	
Set IQ Initial current (unit : mA)	0
Set IQ Starting current (unit : mA)	440
Set IQ End current (unit : mA)	450
IQ_TailWind Value (unit : mA)	

设定Openloop启动电流



IQ_PI_OUT

定位时间

启动电流

结束电流

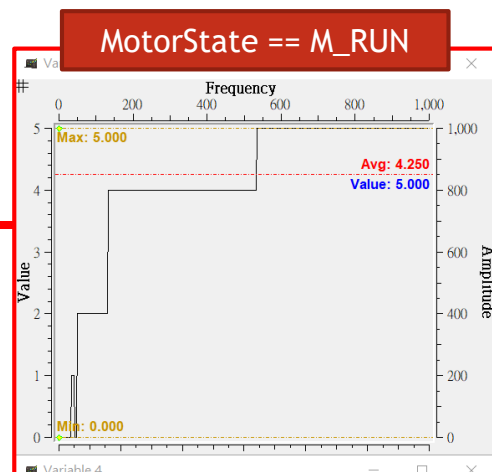
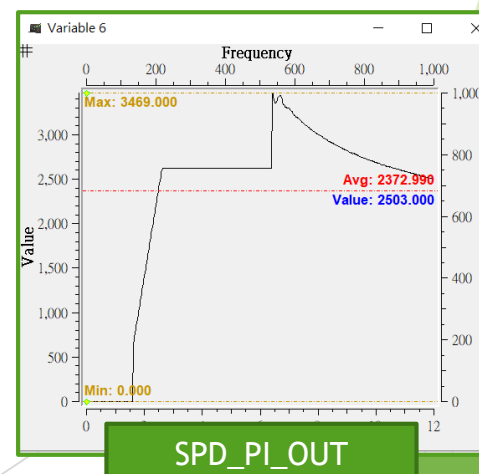
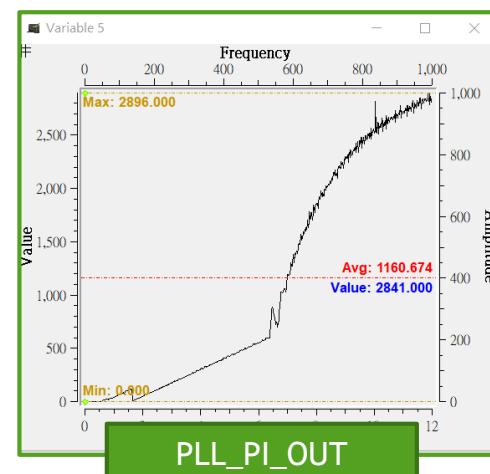
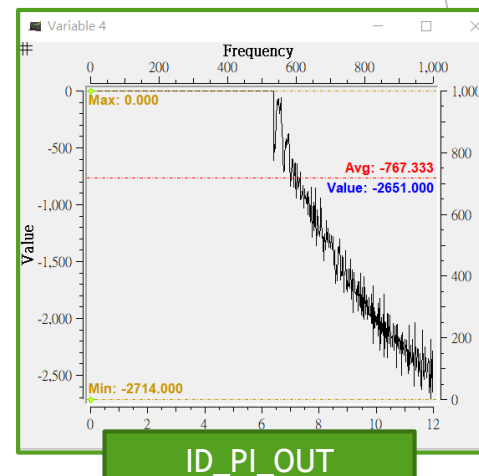
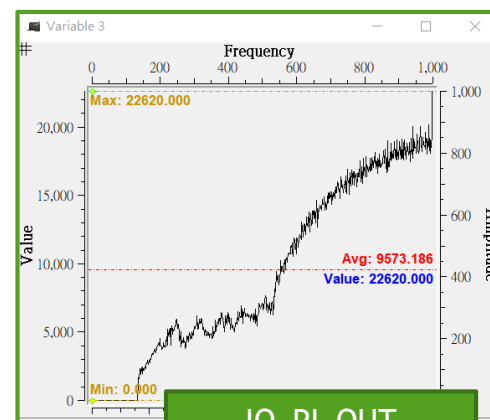
顺逆风启动电流



Motor Control 调机流程 (7)

1. 开回路运转调完成
2. 设定调机流程 CloseLoop
3. 进入闭回路阶段 · 调机完成

Set the motor tuning process	
FOC_Control_Stage	CloseLoop
Set IQ parking duration(unit : ms)	10
Set SMO_PLL initial speed (unit : 10rpm)	1
Set SMO_PLL end speed (unit : 10rpm)	300
Set PLL accumulation	2
Set SMO_RAMP acceleration slope (unit : ms)	1
Set SMO_DELAY Delay time (unit : ms)	10



```
enum MotorStatus
{
    →M_OFF = 0,
    →M_INIT = 1,
    →M_TAILWIND = 2,
    →M_IPD = 3,
    →M_START = 4,
    →M_RUN = 5,
    →M_STOP = 6,
    →M_BREAK = 7,
    →M_ERROR = 8,
    →M_BMF_BREAK = 9
};
```



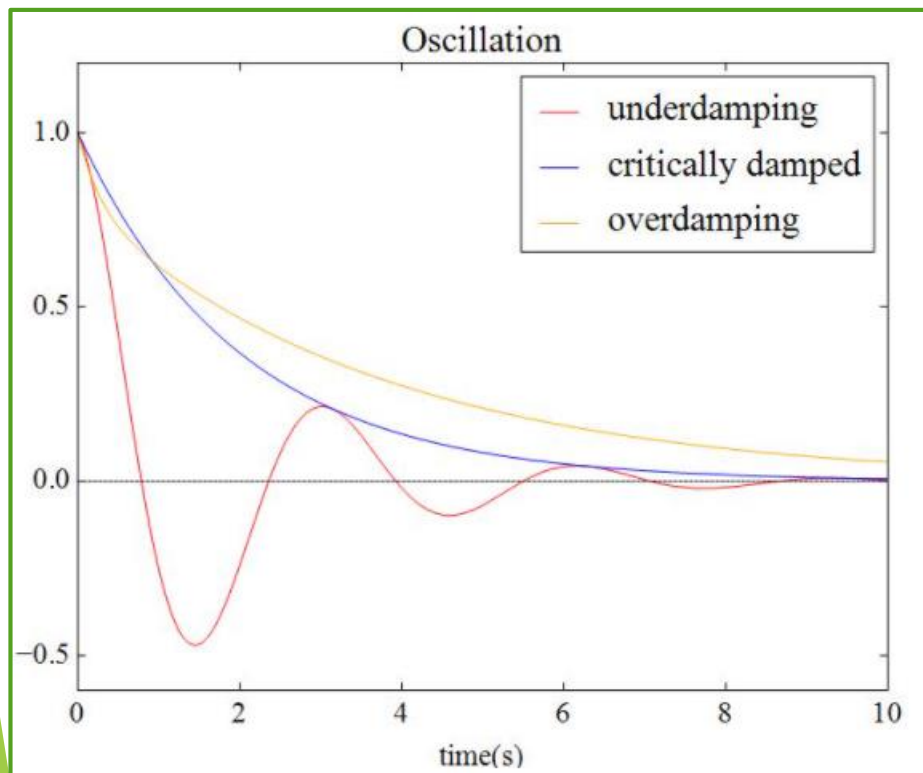
Motor Control 调机流程 (8)

各项PI控制器参数，依造不同马达参数微调。

在自动控制理论中，系统响应分类如下：

1. 过阻尼：PI 控制器调整方向， $K_p \uparrow$ 、 $K_i \downarrow$ 。
2. 欠阻尼：PI 控制器调整方向， $K_p \downarrow$ 、 $K_i \uparrow$ 。
3. 临界阻尼： K_p 、 K_i 为理想值。

K_t 为反积分终结参数，预设32767 即可。



Set IQ PI parameters	
Kp parameters	28000
Ki parameters	160
Kt parameters	32767
MaxLimit parameters	32767
MinLimit parameters	32767

Set ID PI parameters	
Kp parameters	28000
Ki parameters	160
Kt parameters	32767
MaxLimit parameters	32767
MinLimit parameters	32767

Set SPEED PI parameters	
Start Kp	0
Final Kp	6000
Ki parameters	130
Kt parameters	32767
MaxLimit (unit : mA)	550
MinLimit (unit : mA)	0
Speed Cycle parameters	20

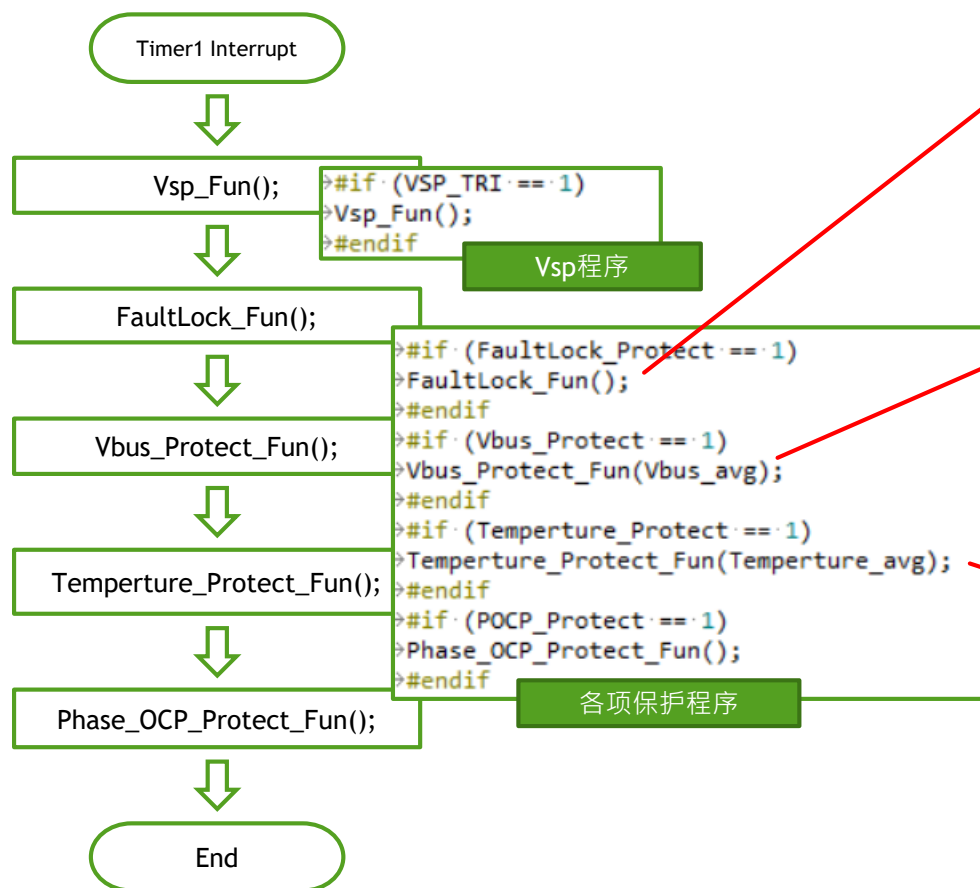
PLL	
Start Kp	1000
Final Kp	1000
Ki parameters	50
HeadWind/TailWind	
Kt parameters	32767
MaxLimit parameters	32767
MinLimit parameters	32767

PI_GAIN	
PLL KI Gain x16	Enable
PLL KP Gain x16	Enable
SPEED KI Gain x16	Disable
SPEED KP Gain x16	Enable
ID KI Gain x16	Disable
ID KP Gain x16	Disable
IQ KI Gain x16	Disable
IQ KP Gain x16	Disable



Motor Control 调机流程 (9)

马达各项保护功能



Locked-rotor protection (LRP)	<input checked="" type="checkbox"/>
Motor speed abnormally high value (unit : 10rpm)	13000
Motor speed abnormally low value (unit : 10rpm)	600
LRP DURATION (unit : ms)	500

Overvoltage/Undervoltage protection (OVP/UVP)	<input checked="" type="checkbox"/>
Set Vbus A/D Channel	CH2
Set Vbus rate parameter	2160
OVP Values (unit : 0.1V)	3800
OVP recovery Values (unit : 0.1V)	3750
UVP recovery Values (unit : 0.1V)	1450
UVP Values (unit : 0.1V)	1400
BUS_VOLT_DURATION (unit : ms)	50

1.选择OVP_CH

2.需调整正确倍率

3.调整OVP保护

Over temperature protection(OTP)	<input checked="" type="checkbox"/>
Set OTP A/D Channel	CH5
OTP A/D Values (unit : Val)	670
OTP recovery A/D Values (unit : Val)	620
OVER_TEMPERATURE_LOAD_REDUCE_VALUE (unit : Val)	670
TEMPERATURE_DURATION (unit : ms)	500



Motor Control 调机流程 (10)

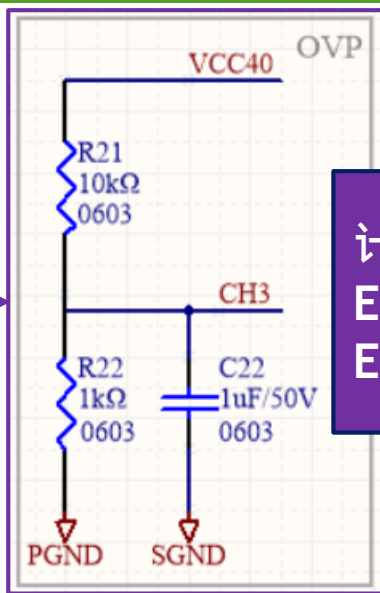
马达各项保护功能

Overvoltage/Undervoltage protection (OVP/UVP)	<input checked="" type="checkbox"/>
Set Vbus A/D Channel	CH2
Set Vbus rate parameter	2160
OVP Values (unit : 0.1V)	3800
OVP recovery Values (unit : 0.1V)	3750
UV recovery Values (unit : 0.1V)	1450
UV Values (unit : 0.1V)	1400
BUS_VOLT_DURATION (unit : ms)	50

1. 选择OVP_CH

2. 需填写OVP分压正确倍率

3. 调整OVP保护



计算Vbus倍率参数, $V_{cc}=36V$, $AD_Val=669$

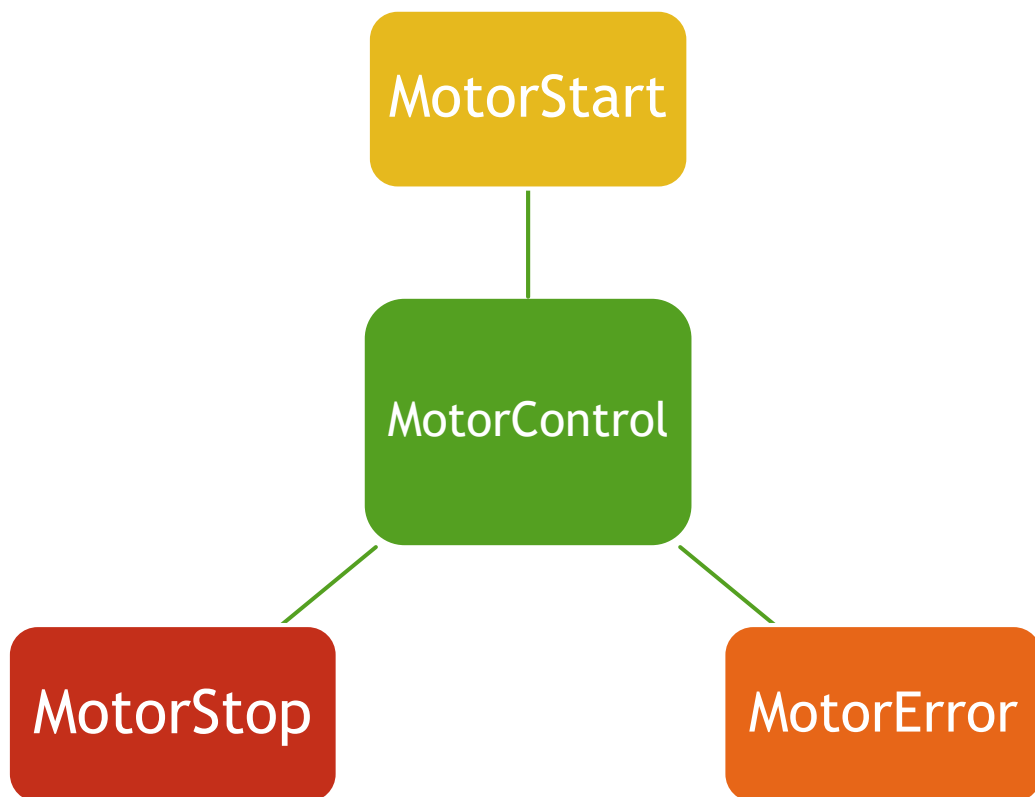
Ex1. Vbus倍率参数= $669/36=18.5833$

Ex2. Vbus倍率参数= $((R22/(R21+R22))/5)*1023=18.6$



马达控制程序流程 (1)

马达驱动程序流程



马达驱动程序

```
void Motor_Control(void)
{
    → #if (CW_CCW_FUNCTION == 1)
    → if (CCWFlag != CCWFlagOld) // 正反转控制
    → → MotorState = M_INIT;
    → → CCWFlagOld = CCWFlag;
    → #endif

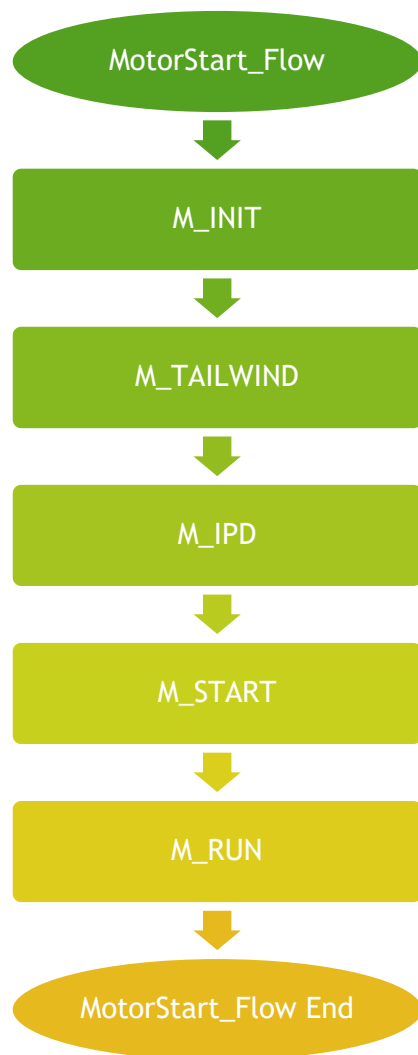
    → if (MotorErrorState)
    → {
    → → if (MotorState != M_OFF)
    → → → ResetMOC();
    → → → PWM_SetAllOff();
    → → → MotorStartRetry_Flow();
    → → → if ((FG_CTRL_REGS & 0x80) == 0x80)
    → → → FG_DISABLE;
    → → #endif
    → }

    → else if ((SystemState & 0x04) == 0x04) // 啟動
    → {
    → → MotorStart_Flow();
    → }

    → else
    → {
    → → if (MotorState != M_OFF)
    → → → ResetMOC();
    → → → PWM_SetAllOff();
    → → → MotorStartRetryCount = 0;
    → → → MotorErrorState = Clear;
    → → → MotorState = M_OFF;
    → → → if ((FG_CTRL_REGS & 0x80) == 0x80)
    → → → FG_DISABLE;
    → → #endif
    → }
}
```



马达启动程序流程 (1)



马达启动程序

```
void MotorStart_Flow(void)
{
    switch (MotorState)
    {
        case M_INIT:
            MotorInit_Fun();
            if (CW CCW FUNCTION == 1) // 正反转控制
                break;
        case M_TAILWIND:
            if (TAILWIND FUNCTION == 1)
                break;
        case M_IPD:
            if (IPD FUNCTION == 1)
                break;
        case M_START:
            if (StartUpState == S_IPD) // IPD Start Up
                IPDStart_Fun();
            if (StartUpState == S_TAILWIND) // Tailwind Start Up
                TailwindStart_Fun();
            break;
        case M_RUN:
            if (CURRENT CONTROL == 1)
            if (SPEED CONTROL == 1)
                break;
```



顺逆风启动机制 (1)

目前有实现出顺逆风启动的方法：

1. 顺逆风判断(Two BEMF分压)
2. 顺逆风判断(Diode BEMF分压)

目前有实现出顺风启动的方法：

1. 顺风判断(One BEMF分压)

至少需要**两相BEMF**才能做**顺逆风判断**，
做**顺风判断**则**一相BEMF**即可。

每个区块都有规画顺逆风调整流程：

LEVEL_1：将程序停止至**M_TAILWIND**。

LEVEL_2：将程序执行至**M_START**、**M_RUN**。

```
enum MotorStatus
{
    → M_OFF = 0,
    → M_INIT = 1,
    → M_TAILWIND = 2,
    → M_IPD = 3,
    → M_START = 4,
    → M_RUN = 5,
    → M_STOP = 6,
    → M_BREAK = 7,
    → M_ERROR = 8,
    → M_BMF_BREAK = 9
};
```

Set Fairwind and Headwind judgment function	
BEMF Fairwind/Headwind judgment (resistance) Enable/Di...	<input checked="" type="checkbox"/>
BEMF Fairwind/Headwind judgment (Diode) Enable/Disable	<input checked="" type="checkbox"/>
BEMF TailWind Fun (One BEMF) Enable/Disable	<input checked="" type="checkbox"/>

BEMF Fairwind/Headwind judgment (resistance) Enable/Disable	<input checked="" type="checkbox"/>
BEMF_TAILWIND_IQ_OUT_VALUE (unit : Val)	8000
BEMF_V_CH	CH4
BEMF_W_CH	CH5
BEMF_TAILWIND_SOP	LEVEL 1
BEMF_TAILWIND_SPEED_MAX (unit : 10rpm)	1200
BEMF_TAILWIND_SPEED_MIN (unit : 10rpm)	300
BEMF_HEADWIND_SPEED_MAX (unit : 10rpm)	1200
BEMF_HEADWIND_SPEED_MIN (unit : 10rpm)	300

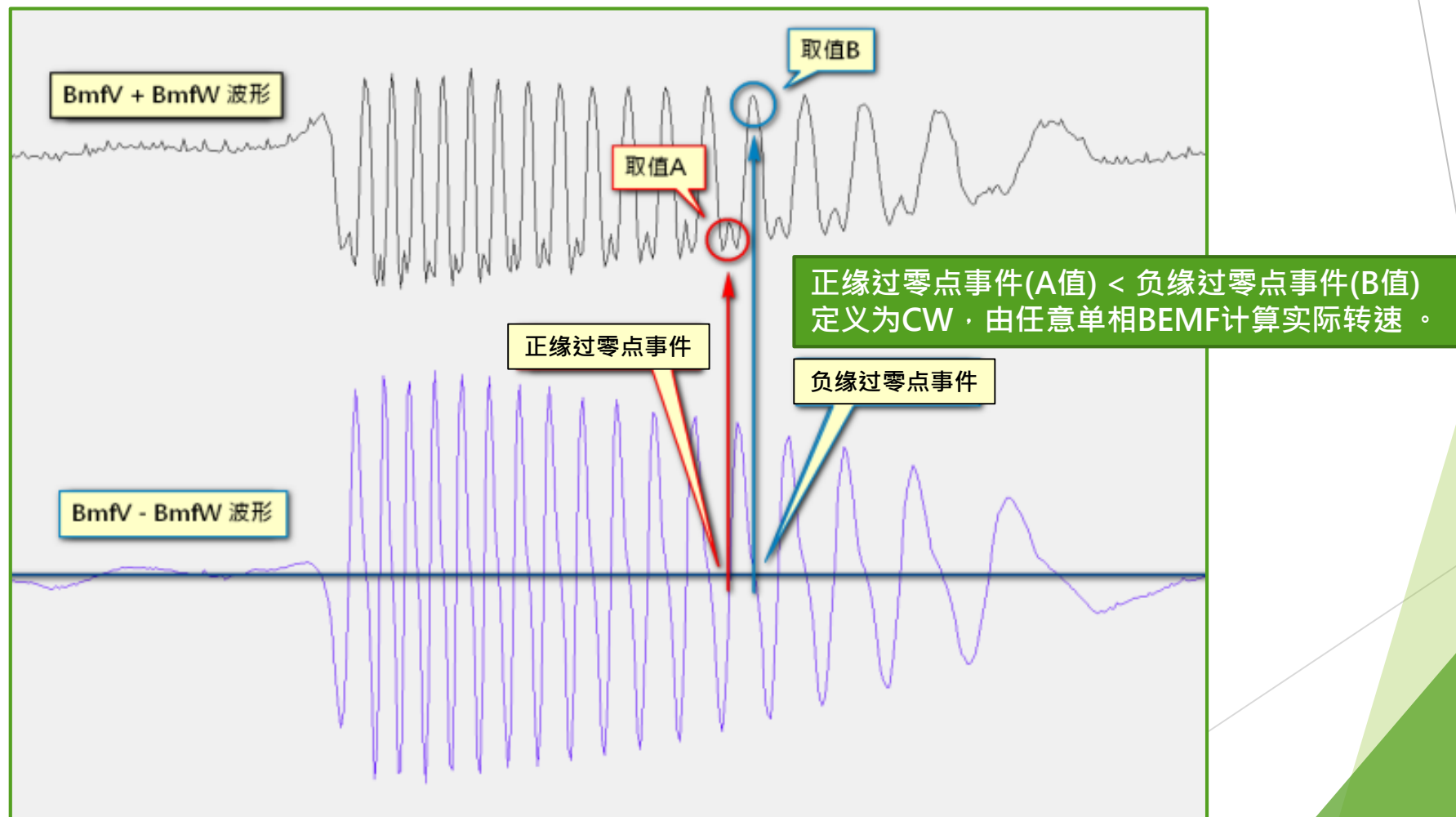
BEMF Fairwind/Headwind judgment (Diode) Enable/Disable	<input checked="" type="checkbox"/>
BEMF_TAILWIND_IQ_OUT_VALUE (unit : Val)	8000
BEMF_V_CH	CH4
BEMF_W_CH	CH5
BEMF_TAILWIND_SOP	LEVEL 2
BEMF_TAILWIND_SPEED_MAX (unit : 10rpm)	1200
BEMF_TAILWIND_SPEED_MIN (unit : 10rpm)	300
BEMF_HEADWIND_SPEED_MAX (unit : 10rpm)	600
BEMF_HEADWIND_SPEED_MIN (unit : 10rpm)	300

BEMF TailWind Fun (One BEMF) Enable/Disable	<input checked="" type="checkbox"/>
BEMF_TAILWIND_IQ_OUT_VALUE (unit : Val)	8000
BEMF_CH	CH4
BEMF_CH_LATEST_THETA	120Deg
BEMF_TAILWIND_SOP	LEVEL 2
BEMF_TAILWIND_SPEED_MAX (unit : 10rpm)	600
BEMF_TAILWIND_SPEED_MIN (unit : 10rpm)	180



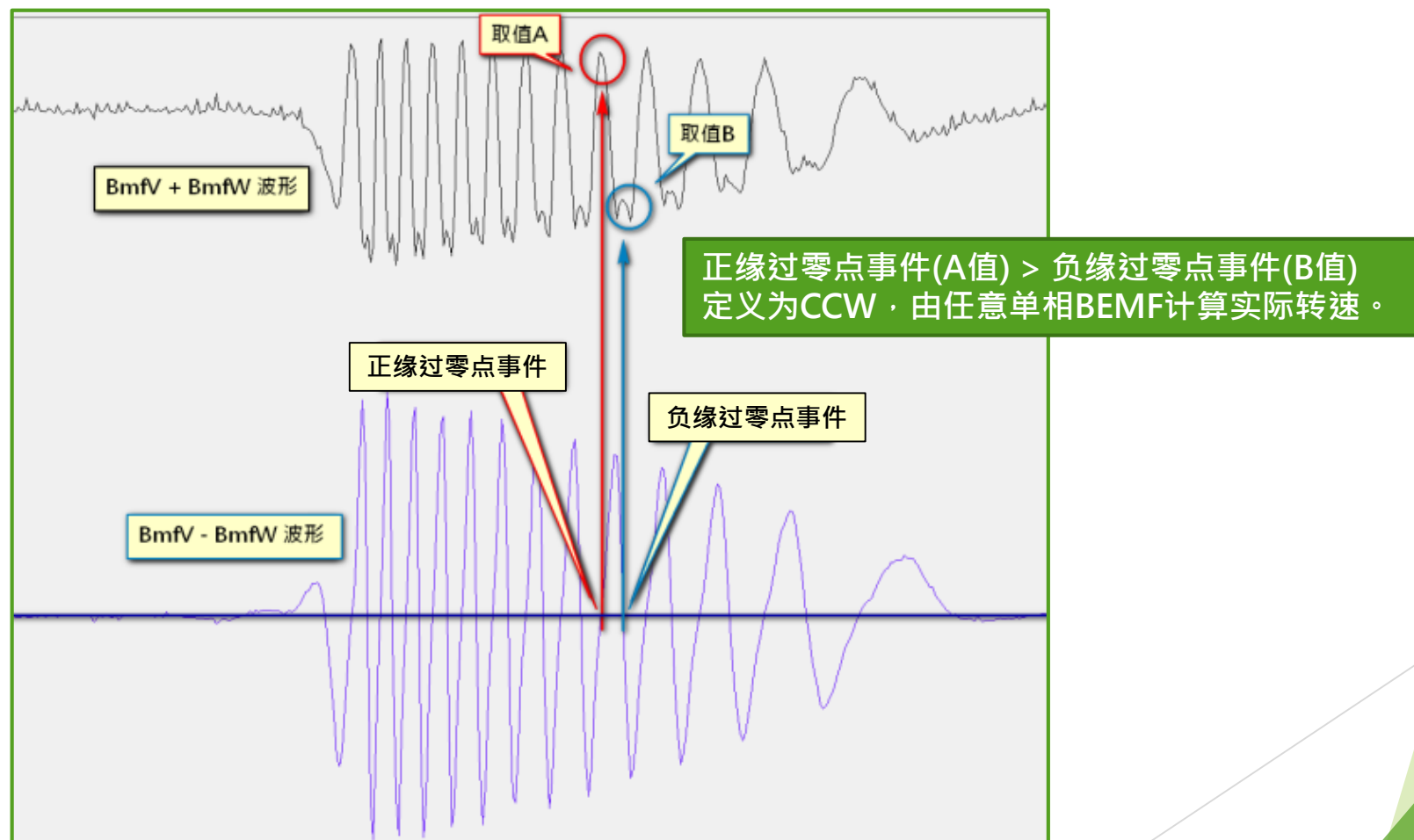
顺逆风启动机制 (2)

顺逆风机制 - 任意两相BEMF回授方式



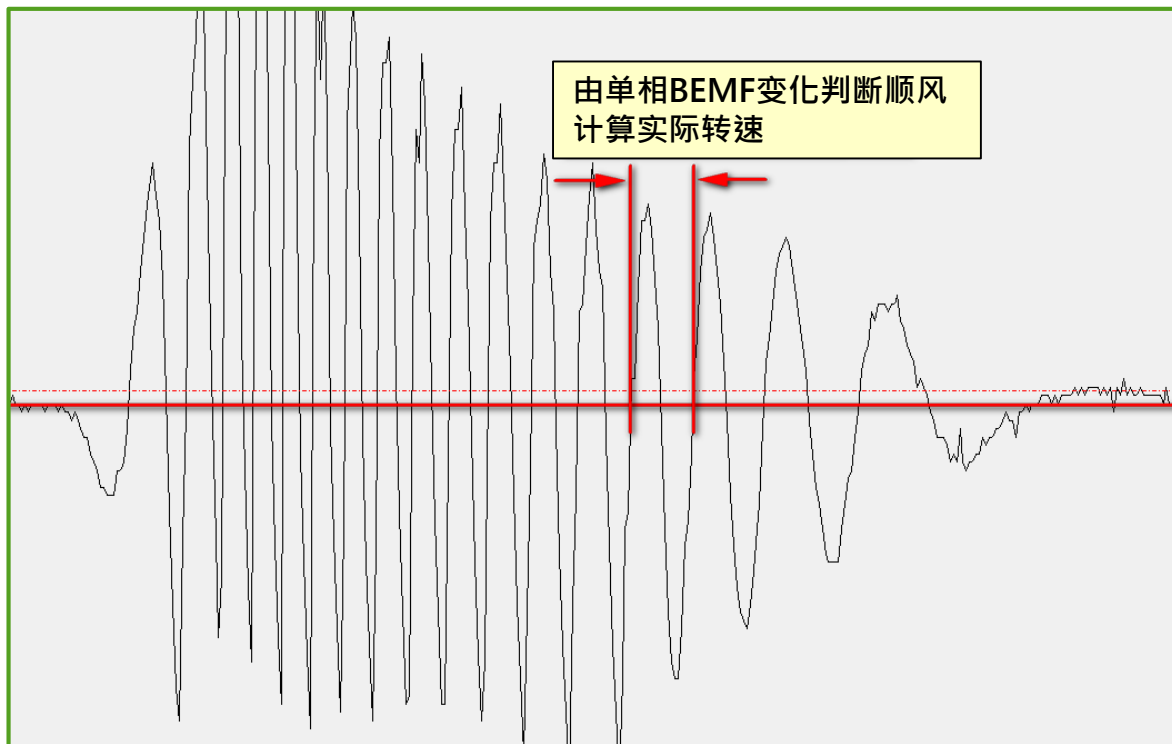
顺逆风启动机制 (3)

顺逆风机制 - 任意两相BEMF回授方式



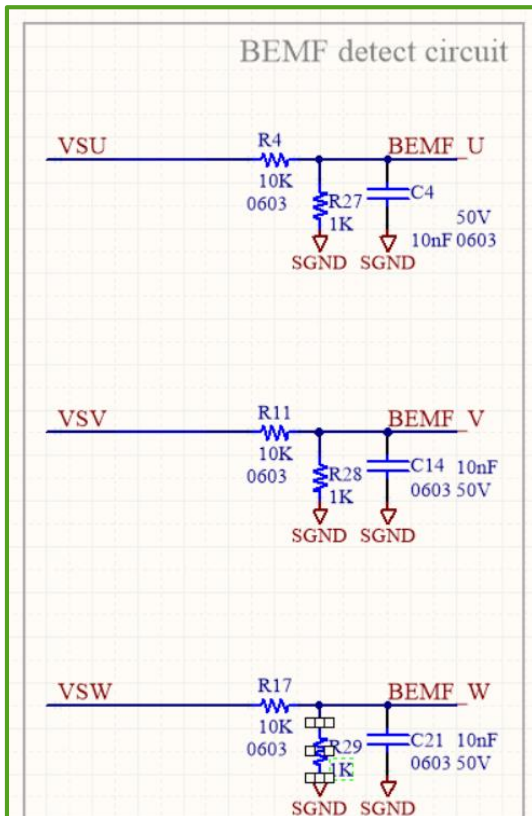
顺逆风启动机制 (4)

顺逆风机制 - 单相BEMF回授方式

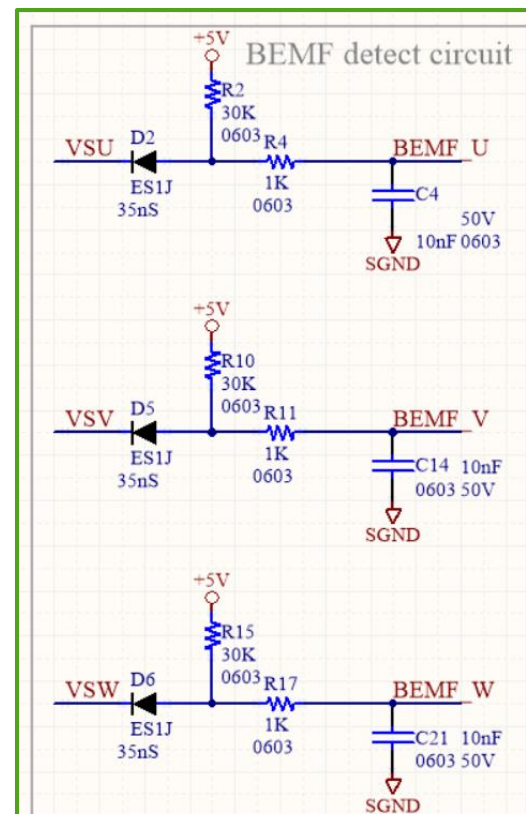


顺逆风启动机制 (5)

BEMF回授电路



分压电阻：此线路可读取到 各相BEMF的变化，但须注意分压电阻的匹配，避免BEMF过大，导致ADC脚位损坏。



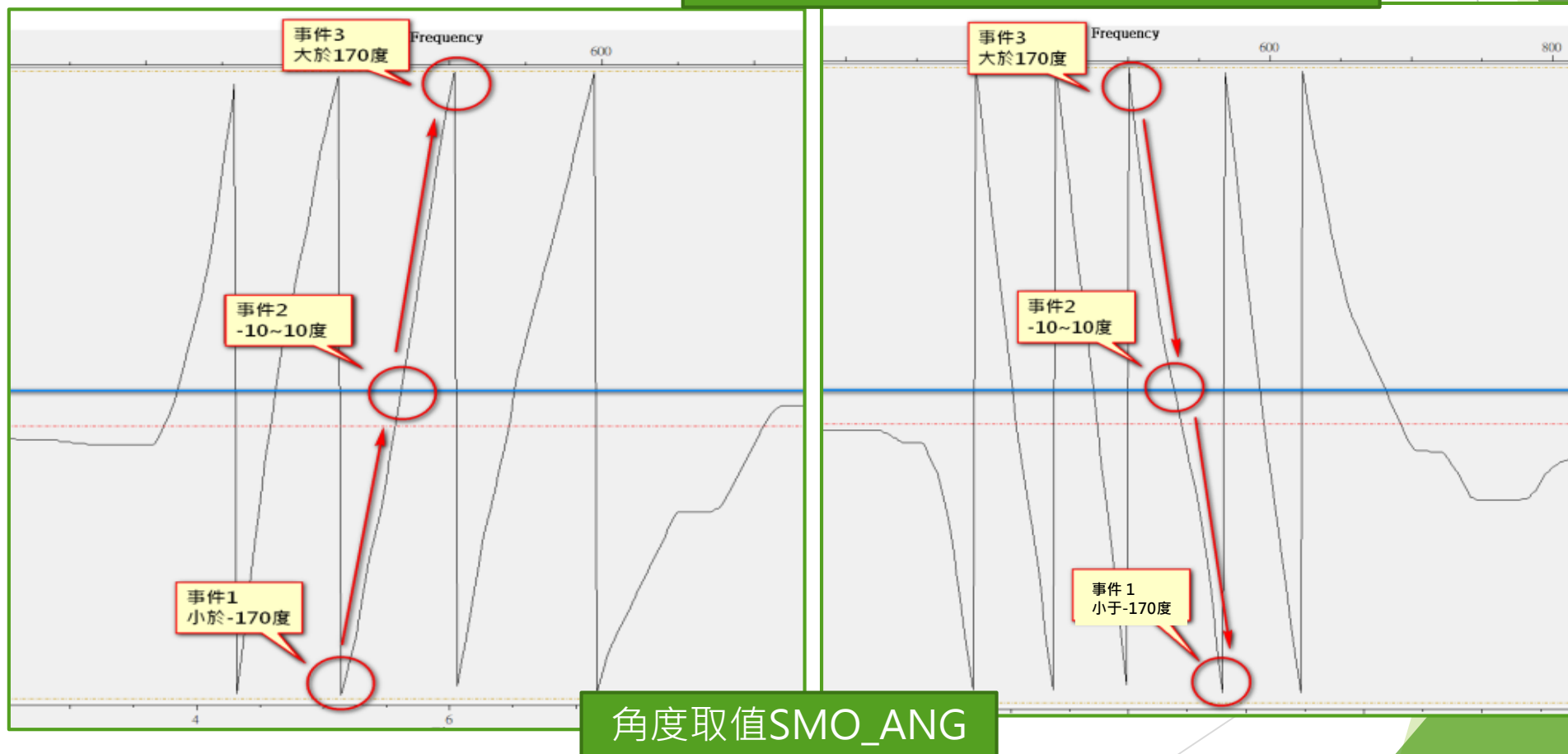
Diode分压电阻：线路需要下臂使用PWM煞车，才能读取到各相BEMF的变化。



顺逆风启动机制 (6)

顺逆风机制 - 煞车回授电流方式

1. 事件1 > 事件2 > 事件3 > 定义为 CW
2. 事件3 > 事件2 > 事件1 > 定义为 CCW



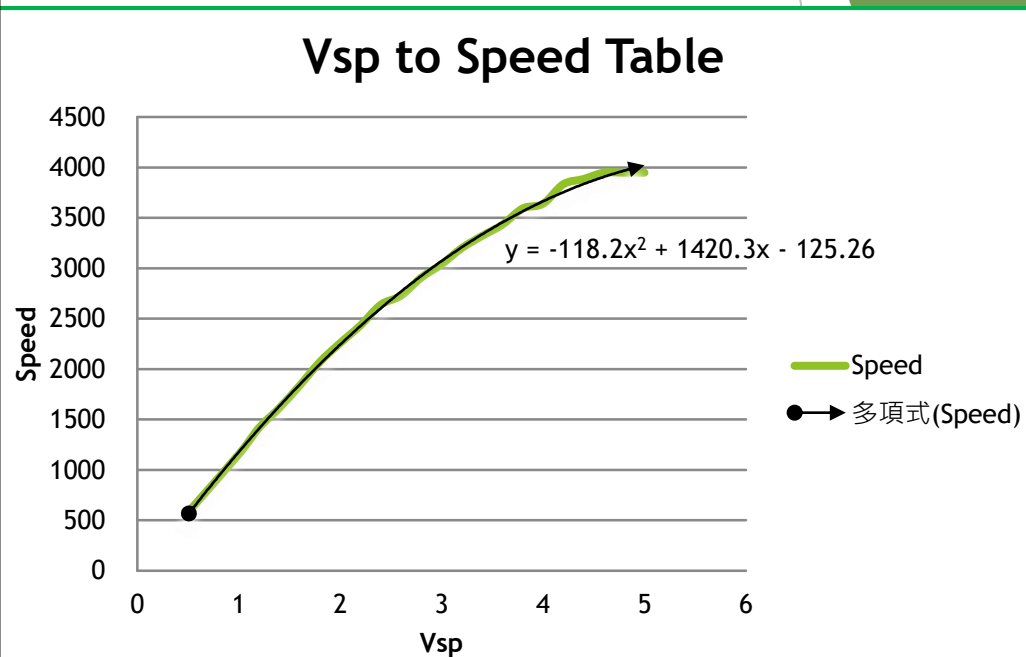
LookUpTable

LookupTable.h LookupTable_Fun main.c

Expand All Collapse All Help Show

Option	Value
Output : LookupTable_Data[1] (unit : Iq_Cmd)	327
Output : LookupTable_Data[2] (unit : Iq_Cmd)	1300
Output : LookupTable_Data[3] (unit : Iq_Cmd)	3750
Output : LookupTable_Data[4] (unit : Iq_Cmd)	6100
Output : LookupTable_Data[5] (unit : Iq_Cmd)	8300
Output : LookupTable_Data[6] (unit : Iq_Cmd)	9300
Input : Vsp_Data[1] (unit : Vsp_Val)	102
Input : Vsp_Data[2] (unit : Vsp_Val)	192
Input : Vsp_Data[3] (unit : Vsp_Val)	400
Input : Vsp_Data[4] (unit : Vsp_Val)	602
Input : Vsp_Data[5] (unit : Vsp_Val)	804
Input : Vsp_Data[6] (unit : Vsp_Val)	1002

Vsp(V)	I(mA)	Speed
0.51	40	590
1	70	1153
1.2	100	1417
1.4	130	1613
1.6	170	1835
1.8	220	2071
2	280	2258
2.2	340	2433
2.4	410	2638
2.6	460	2728
2.8	550	2908
3	620	3040
3.2	720	3201
3.4	800	3321
3.6	890	3433
3.8	1010	3594
4	1070	3640
4.2	1230	3834
4.4	1290	3886
4.6	1360	3949
4.8	1360	3949
5	1360	3949



```

→ if(Vsp_avg >= 102) // 204 = 1V
→ {
→   SystemState |= 0x04;
→   #if 1
→   CurrentCmd = look1_binlx(Vsp_avg, rtConstP.uDLookupTable_bp01Data, rtConstP.uDLookupTable_tableData, 5);
→   #else
→   CurrentCmd = (int)((float)Vsp_avg * IQ_GAIN);
→   if(CurrentCmd > IQ_MAX_LIMIT_VALUE)
→   CurrentCmd = IQ_MAX_LIMIT_VALUE;
→   else if(CurrentCmd < IQ_MIN_LIMIT_VALUE)
→   CurrentCmd = IQ_MIN_LIMIT_VALUE;
→   #endif
→ }

```

“6笔资料”，宣告LookupTable[5]

“宣告LookupTable[5]” >> 填“5”

利用差分建表方式，将Vsp - Speed Table完整建立对应曲线出来。

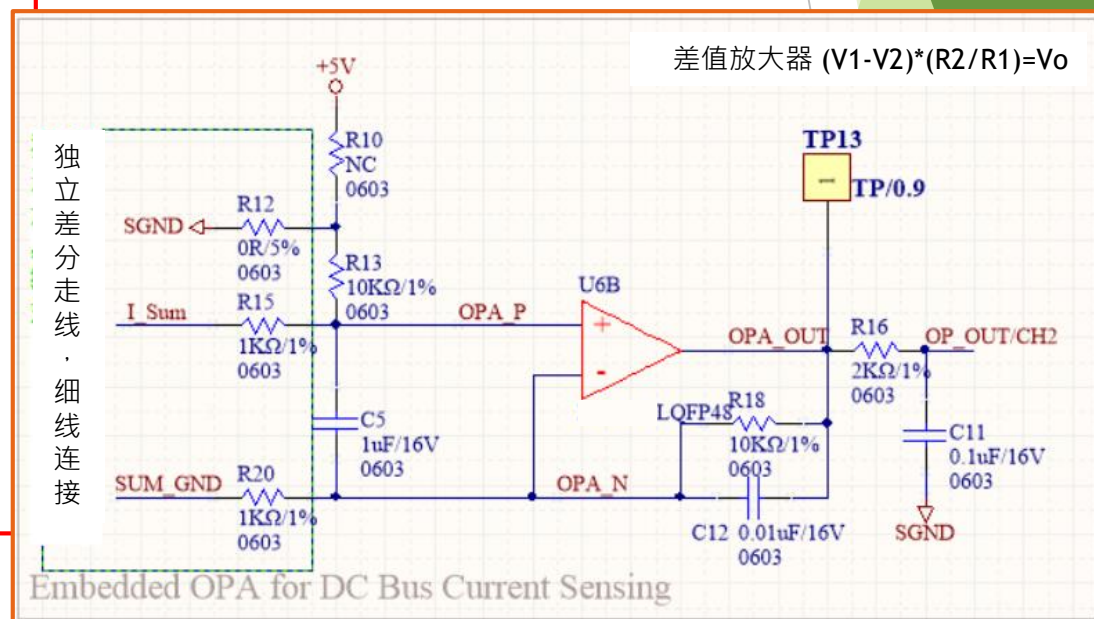
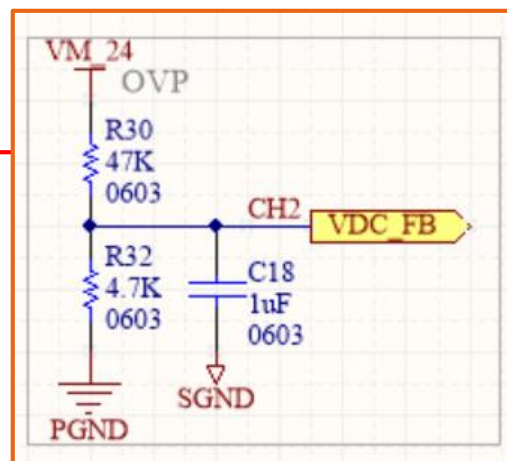


PowerControl (1)

PowerControl_Fun

取样Vbus、Ibus

```
void ADC_ISR (void) interrupt 9
{
    → AdcFlag = 1;
    → #if (Vbus_Protect == 1)
    → Vbus_avg = Adc_Channel(V_BUS_CH);
    → #endif
    → #if ((POWER_CONTROL == 1) || (POWER_LIMIT == 1) || (POWER_CONTROL_USER_PI == 1))
    → Ibus_avg = Adc_Channel(I_BUS_CH);
    → #endif
    → #if (BEMF_TAILWIND_FUNCTION_4 == 1)
    → if (MotorState == M_TAILWIND) {
    → → BmfU = Adc_Channel(BEMF_CH);
    → }
    → #endif
    → #if (BEMF_TAILWIND_FUNCTION_1 == 1) || (BEMF_TAILWIND_FUNCTION_3 == 1)
    → if (MotorState == M_TAILWIND) {
    → → BmfV = Adc_Channel(BEMF_V_CH);
    → → BmfW = Adc_Channel(BEMF_W_CH);
    → }
    → #endif
    → #if (VSP_TRI == 1)
    → Vsp_avg = Adc_Channel(4);
    → Vsp_avg = 1023 - Vsp_avg;
    → #endif
    → #if (Temperature_Protect == 1)
    → Temperature_avg = Adc_Channel(TEMPERATURE_CH);
    → #endif
}
```



PowerControl (2)

PowerControl_Fun

```

if(MotorState == M_RUN)
{
    #if (POWER_CONTROL_USER_PI_SOP == 2)
    if(UserPI_PowerControlDelayCount > POWER_CONTROL_DELAY_DURATION)
    {
        UserPI_PowerControlDelayCount = 0;
        Watt = (int)((float)Vbus_avg * Ibus_avg * dPOWER_GAIN);
        USER_PI_ACTIVE;
        USER_CMD_MACRO(WATT_LIMIT_VALUE);
        USER_FB_MACRO(Watt);
        GET_USER_OUT_MACRO(CurrentCmd);
        CurrentCmdTemp = CurrentCmd;
        IQ_CMD_MACRO(CurrentCmdTemp);
    }
    else
    {
        UserPI_PowerControlDelayCount++;
    }
}
else
{
    Watt = (int)((float)Vbus_avg * Ibus_avg * dPOWER_GAIN);
    CurrentCmdTemp = CurrentCmd;
    IQ_CMD_MACRO(CurrentCmdTemp);
}
#endif
}

```

Power_Control & Power_Limit	
Set the rated output power (max) (unit : 0.01W)	1800
Set power magnification parameters (unit : 10 ⁻⁵)	100
Set I_BUS A/D Channel	CH2
POWER_SOP	LEVEL 2
Power_LookUpTable Enable/Disable	<input type="checkbox"/>

LEVEL_1 : 不执行 PowerControl_Fun
LEVEL_2 : 执行 PowerControl_Fun

```

#define POWER_CONTROL 1
#if (POWER_CONTROL == 1)
#define WATT_LIMIT_VALUE (unsigned long) 800
#define POWER_CONTROL_DELAY_DURATION 2
#define I_BUS_CH 2
#define POWER_PI_OUT (float) 300/1000
#define POWER_PI_OUT_VALUE (signed short)((float) POWER_PI_OUT * I_AMPLIFIER)
#define dPOWER_GAIN ((float) 594/100000)
#define POWER_CONTROL_SOP 2
#if (CURRENT_CONTROL == 0)
#error Wrong setting POWER_CONTROL and CURRENT_CONTROL !!!
#endif

```

只有在CURRENT_CONTROL，才可以使用POWER_CNOTROL。
#error Wrong setting POWER_CONTROL and CURRENT_CONTROL !!!



IPD程式流程 (1)

AOCPCONT		Address = EEH				Reset Value = 0xE7H		
Analog OCP Control Register								
Bit	DOCPNEN	AOCPEN	OPAPD	IPD	----	I_SHORT[3:0]		
	7	6	5	4	3	2	1	0
Type	R	R	----	----	----	R/	R/W	R/W
DOCPNEN		Digital OCPN enable:						
[7]		0 : Disable						
		1 : Enable						
AOCPEN		Analog OCP enable:						
[6]		0 : Disable						
		1 : Enable						
OPAPD		OPA Power Down						
[5]		0 : Normal						
		1 : OPA Power Down						
IPD		IPD (Initial Position Detect) Path Select						
[4]		0 : IPD Current Compare from AOCPP Path						
		1 : IPD Current Compare from OPA Path						
I_SHORT		Analog OCP SHORT level select : (OCP interrupt : OCPIF)						
[2:0]		000 : 0.15V						
		001 : 0.2V						
		010 : 0.25V						
		011 : 0.3V						
		100 : 0.35V						
		101 : 0.4V						
		110 : 0.45V						
		111 : 0.5V(default)						

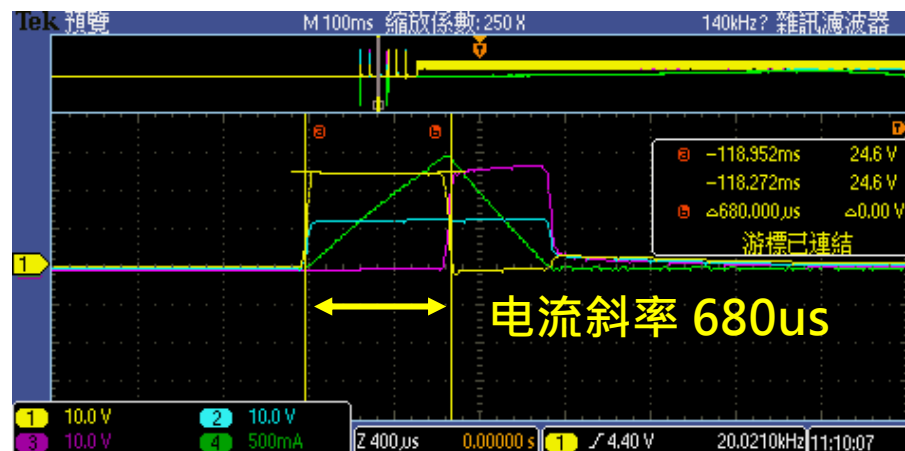
设定 AOCPCONT : IPD_LEVEL

Set IPD LEVEL	
I_SHORT	0.15V
AOCPEN	Enable
DOCPEN	Disable
IPD Path Select	IPD Current Compare from AOCPP Path
Set IPD IAECYC	
IAECYC	24MHz

选择IPD路径

当马达电流斜率太缓慢时，需调低IPD_CYC：

1. 电流斜率时间 < 1.3ms : IPD_CYC 设定48MHz
2. 电流斜率时间 < 2.6ms : IPD_CYC 设定24MHz
3. 电流斜率时间 < 5.2ms : IPD_CYC 设定12MHz
4. 电流斜率时间 < 10.4ms : IPD_CYC 设定 6MHz



IPD程式流程 (2)

IPD程式流程建议分步骤执行

AOCPCONT : IPD_LEVEL
需要预先设定

P+N、N+N Gate Driver 需要下臂预充电 20 ~ 30ms

当启用IPD时，需先WatchDog Disable，避免IPD过程中触发WatchDog

LatestTheta初始角度，因马达不同可作微调，目前参数为预设建议值。
IPDAdvanceAng作为进相角调整。

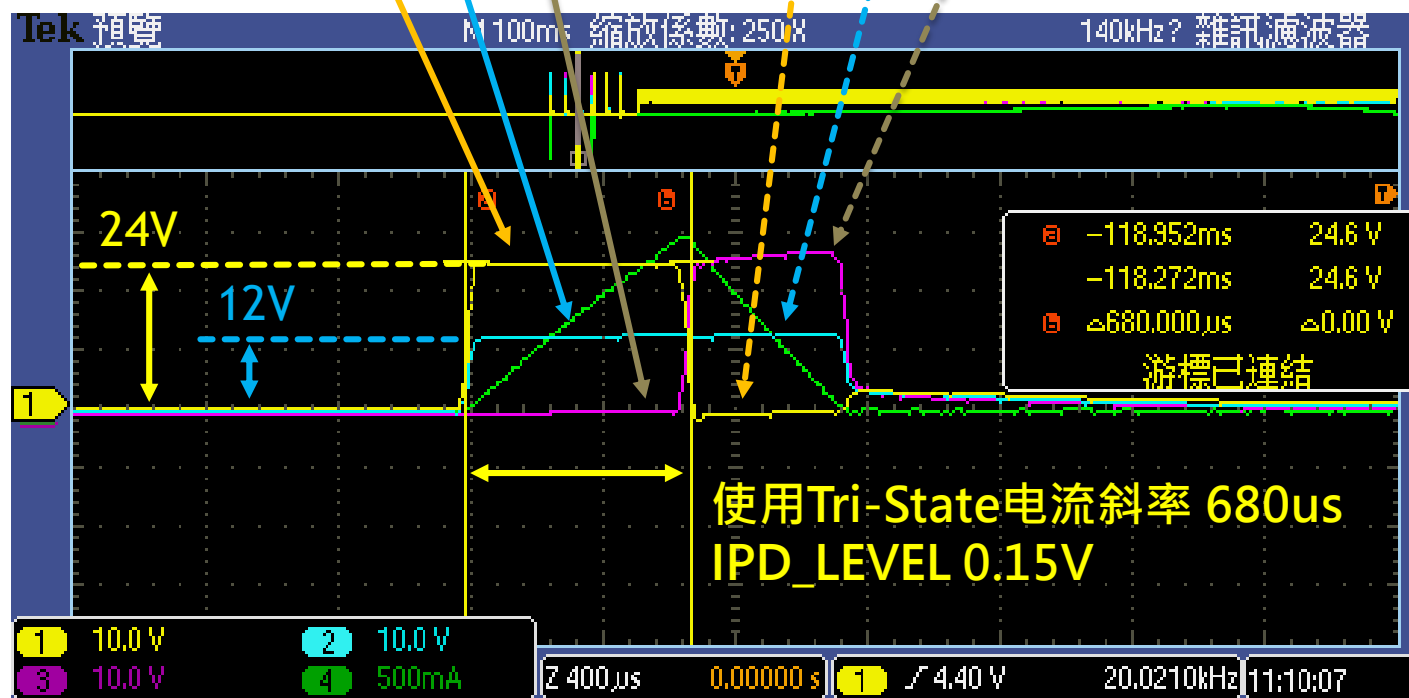
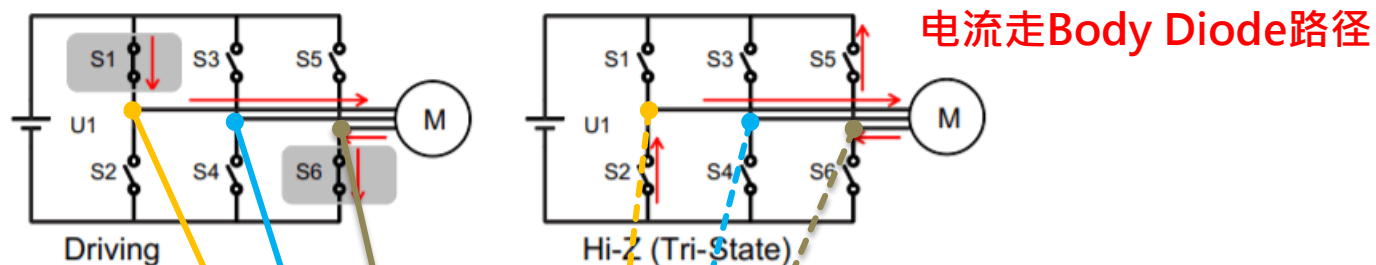
```
void IPDDetect_Fun(void)
{
    #define IPDAdvanceAng 30
    switch (IPD_Detect_State)
    {
        case 0:
            AOCPCONT = IPD_LEVEL;
            Break_Fun(); // 注：当 IPD 时，N+N Gate Driver 需要下臂预充电
            IPD_Cnt++;
            if (IPD_Cnt > 20)
            {
                IPD_Detect_State = 1;
                IPD_Cnt = 0;
            }
            break;
        case 1:
            WatchDog_Disable(); // (注：当 IPD 时，会触发 WatchDog Reset, Disable WatchDog)
            IPD_Init();
            WatchDog_Init();
            IPD_Detect_State = 2;
            break;
        case 2:
            if (IPDPattern == 4) LatestTheta = (64+IPDAdvanceAng)<<6;
            else if (IPDPattern == 5) LatestTheta = (128+IPDAdvanceAng)<<6;
            else if (IPDPattern == 2) LatestTheta = (192+IPDAdvanceAng)<<6;
            else if (IPDPattern == 3) LatestTheta = (256+IPDAdvanceAng)<<6;
            else if (IPDPattern == 6) LatestTheta = (320+IPDAdvanceAng)<<6;
            else if (IPDPattern == 1) LatestTheta = (383+IPDAdvanceAng)<<6;
            else LatestTheta = (383+IPDAdvanceAng)<<6;
            MotorErrorState &= ~(AOCPCONT);
            MotorState = M_START;
            IPD_Detect_State = 0;
            break;
        default:
            break;
    }
}
```

Set IPD LEVEL	
I_SHORT	0.15V
AOCPCEN	Enable
DOCPEN	Disable
IPD Path Select	IPD Current Compare from AOCP Path

Set IPD IAECYC	
IAECYC	24MHz

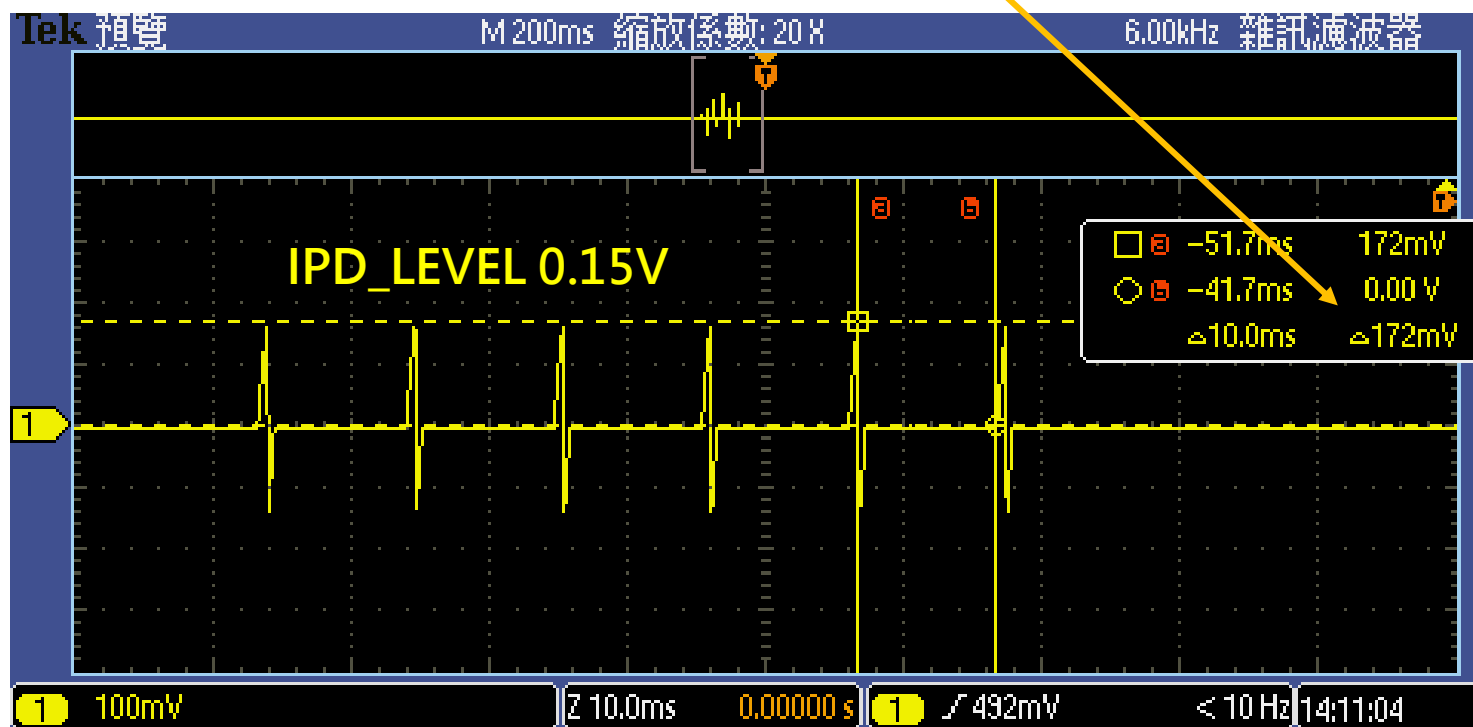


IPD程式流程 (3)



IPD程式流程 (4)

Set IPD LEVEL	
I_SHORT	0.15V
AOCPEN	Enable
DOCPEN	Disable
IPD Path Select	IPD Current Compare from AOC Path
Set IPD IAECYC	
IAECYC	24MHz



MDRFD0应用程式支援项目

项目	MDRFxx	MDSFxx	备注
Vsp控制	○	○	
电流控制	○	○	
转速控制	○	○	
功率控制	○	○	MDRFD0使用上，注意Code Size
功率限制	○	○	MDRFD0使用上，注意Code Size
正反转控制	○	○	
初始位置侦测	○	○	
顺逆风侦测	○	○	MDSF40利用Cap计算转速 MDRFD0利用Pwm中断计算转速
IR Decoder	×	○	MDRFD0需要用Cap计算解码
断电记忆	×	○	MDRFD0需要外部EEPROM

