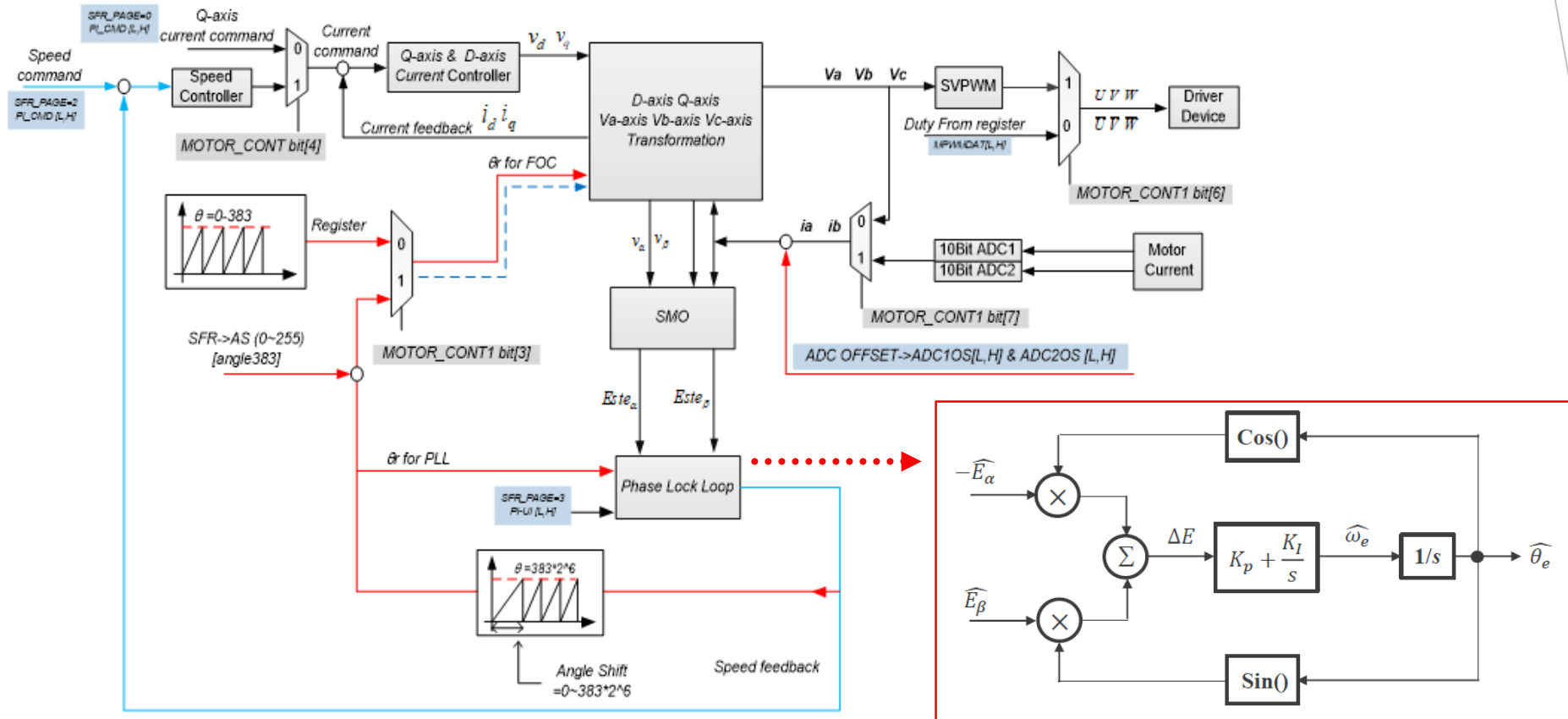


MDRFD0

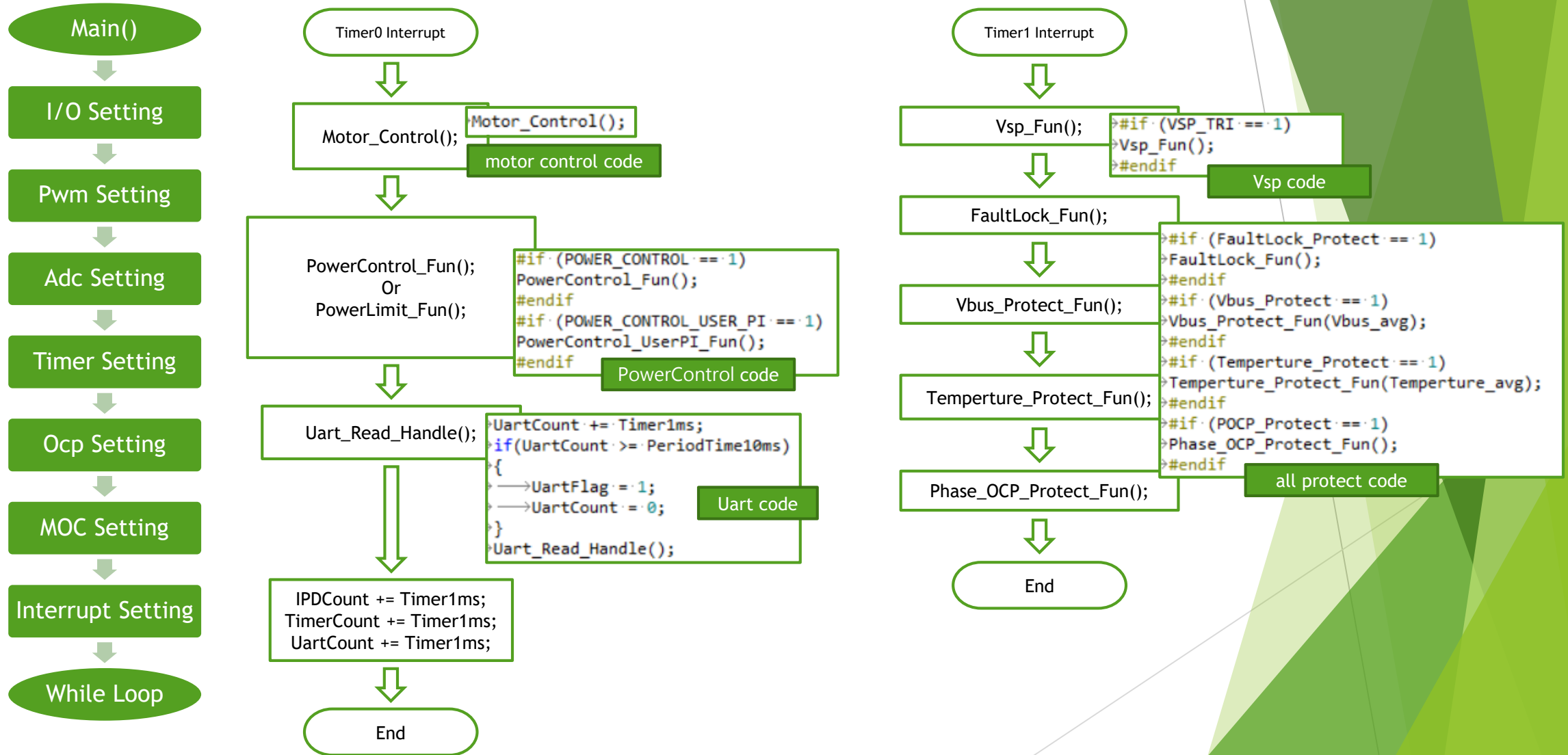
Sample Code Sensorless Introduce



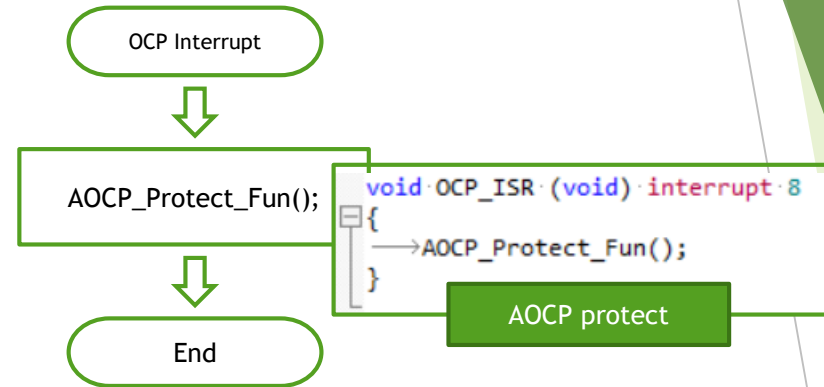
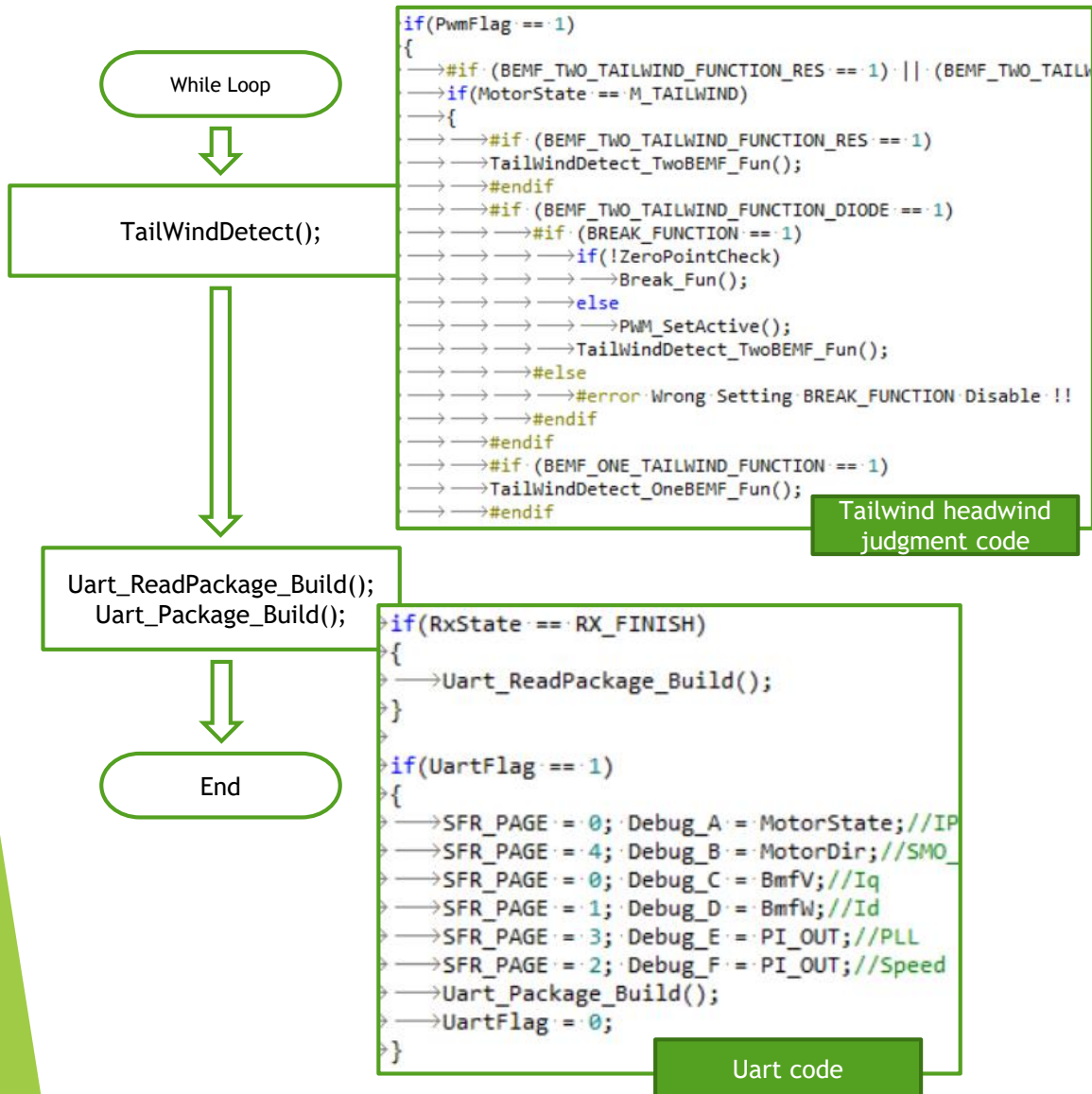
MOC block diagram



Program flowchart



Program flowchart



I/O Initiation setting

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

```
Gpio.c

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

Gpio.h		main.c
Expand All		Collapse All
Help		✓ Show Grid
Option	Value	I/O Input - Output option
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		I/O Pull Up - No Pull option
CH7	No Pull	
CH6	No Pull	
CH5	No Pull	
CH4	Pull Up	
PINSET2		



Pwm Initiation setting

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

```
void PWM_Init(void)
{
    SFR_PAGE = 0;
    MPWMDATA = MPWMDATA_REGS;
    SYNC = 0x55;
    MPWMINV = MPWMINV_REGS;
    SYNC = 0x55;
    MPWMDDB = MPWMDDB_REGS;
    SYNC = 0x55;
    PWM_SetAllOff();
}
```

Pwm Initiation setting

BASE_RPM : represent PLL_OUT 32767 the actual speed corresponding to the unit value

Notice PWM SWAP option , need to be correct

Pwm.h	
Expand All	Collapse All
Help	Show Grid
Option	Value
Set MPWM SWAP	MDSFD0
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 MPWMDDB	
Deadband Time	Deadband Time 1.5us
BASE_RPM (unit : rpm)	32767

PWM SWAP option

PWM frequency

PWM Dead Time

Adc Initiation setting

```

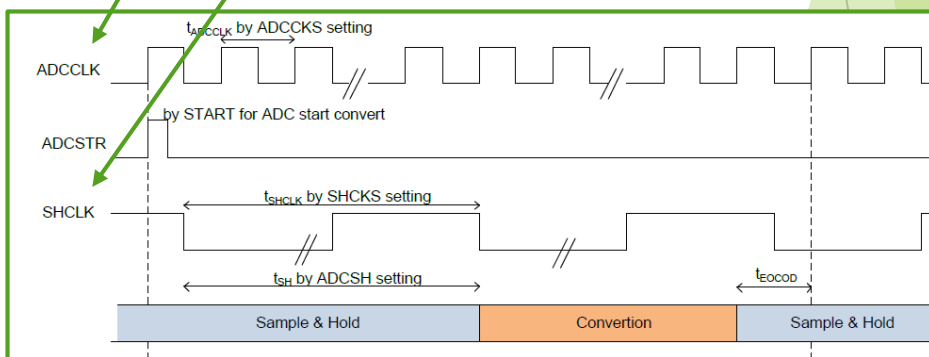
main.c  Gpio.h  Gpio.c
75 void main (void){
76     PWM_SetAllOff();
77     GPIO_Init();
78     PWM_Init();
79     Adc_Init();
80     Timer_Init();
81     OCP_Init();
82     MOC_Init();
83     Interrupt_Init();
84
85     #if (Uart_Debug == 1)
86         Uart_Definition();
87     #endif
88
89     #if (Uart_Debug == 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 }

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;
}
    
```

Adc Initiation setting

AdcCKS、SHCKS option

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



Timer Initiation setting

```
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 }
```

Timer Initiation setting

```
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;
}
```

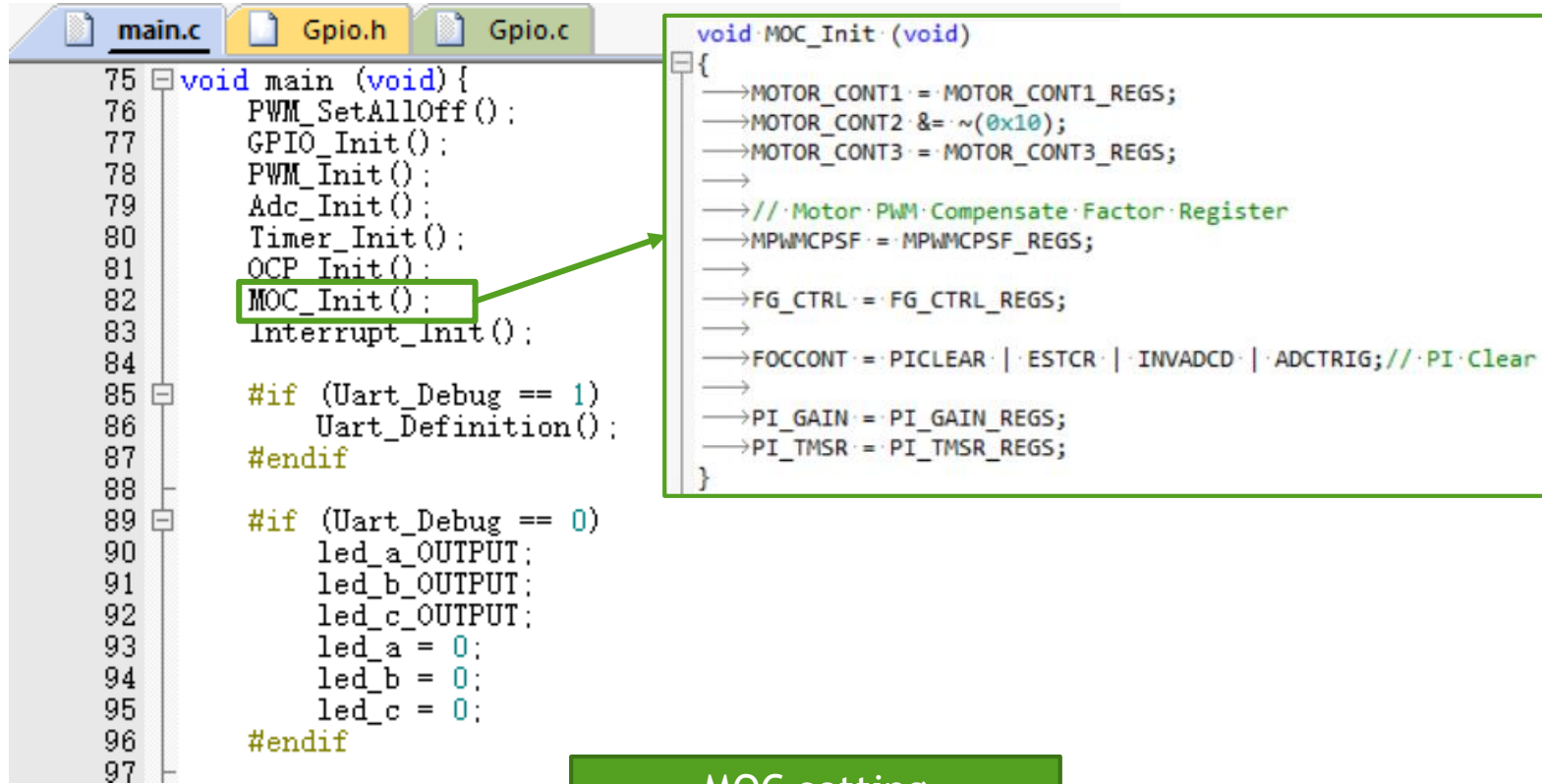
Timer.h Adc.h Pwm.h

Expand All Collapse All Help ☒ Show Grid

tion	Value
PFCON	
TOPS	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

Timer Interrupt frequency setting

MOC setting



The image shows a code editor with two files open: `main.c` and `Gpio.c`. In `main.c`, the `MOC_Init();` function call on line 82 is highlighted with a green box. A green arrow points from this box to the `MOC_Init` function definition in `Gpio.c`. The `Gpio.c` file shows the implementation of `MOC_Init`, which initializes various motor control registers.

```
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 }
```

```
Gpio.c
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

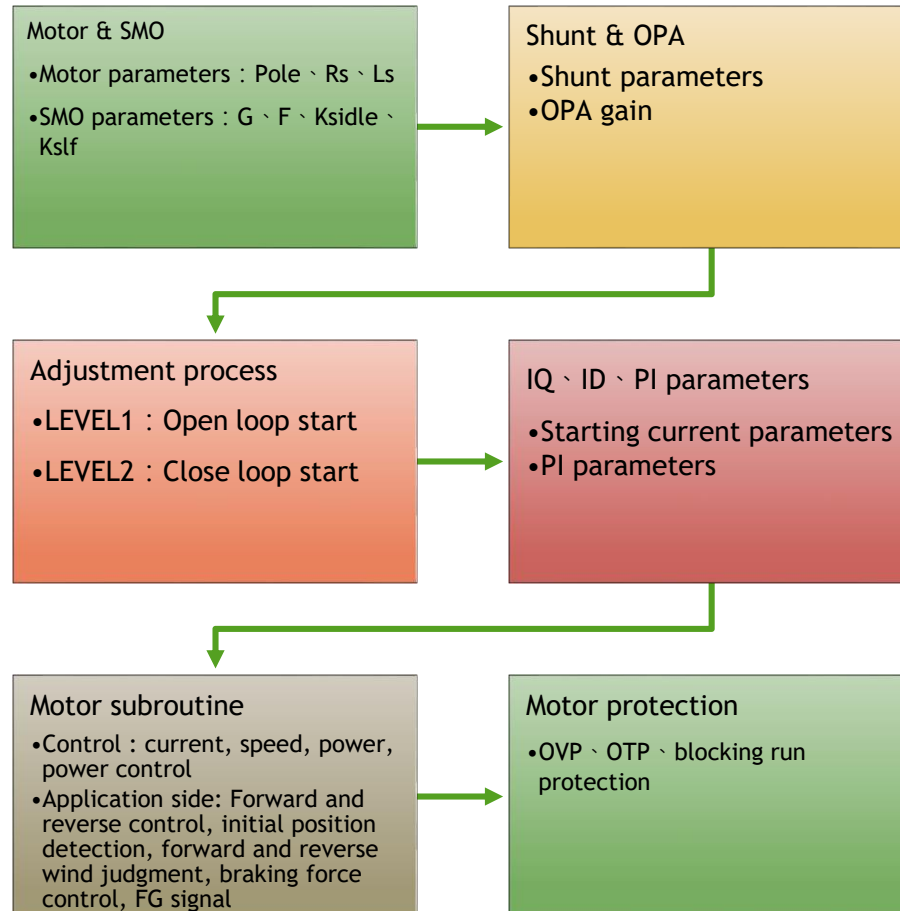
Interrupt setting

```
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 }
```

```
{
  → 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
  → WDTIE = 0; // WatchDoag_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

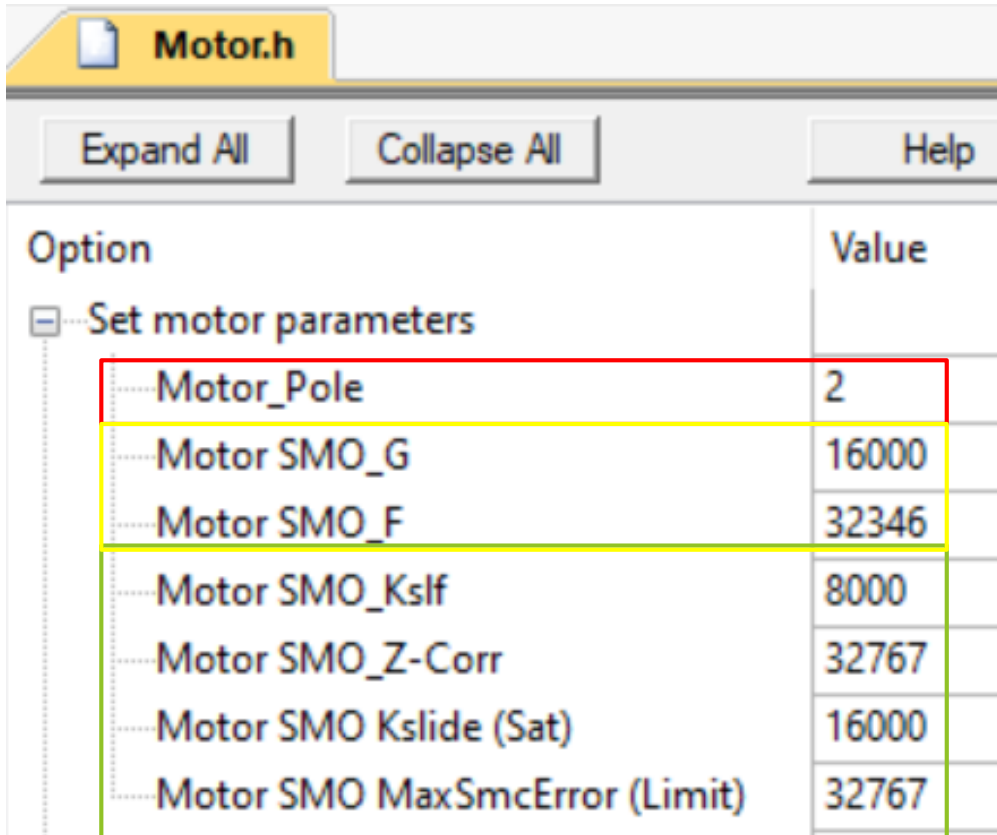
Motor Control adjustment process (1)



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 adjustment process (2)



Option	Value
<input checked="" type="checkbox"/> 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

Set the motor poles (required).

If you are not familiar with the motor parameters, the recommended initial setting of the SMO parameters is as follows:

SMO_G : 16000

SMO_F : 32000

SMO_Kslf : 8000

SMO_ZCorr : 32767

SMO_Kslide : 32767

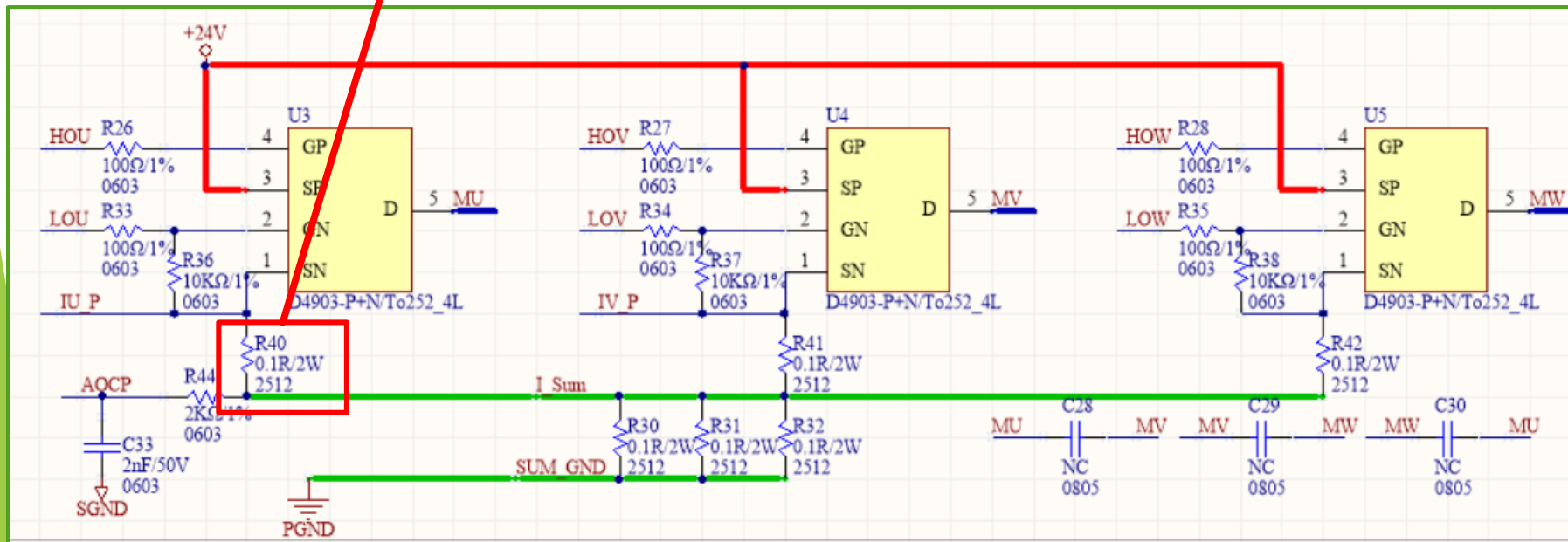
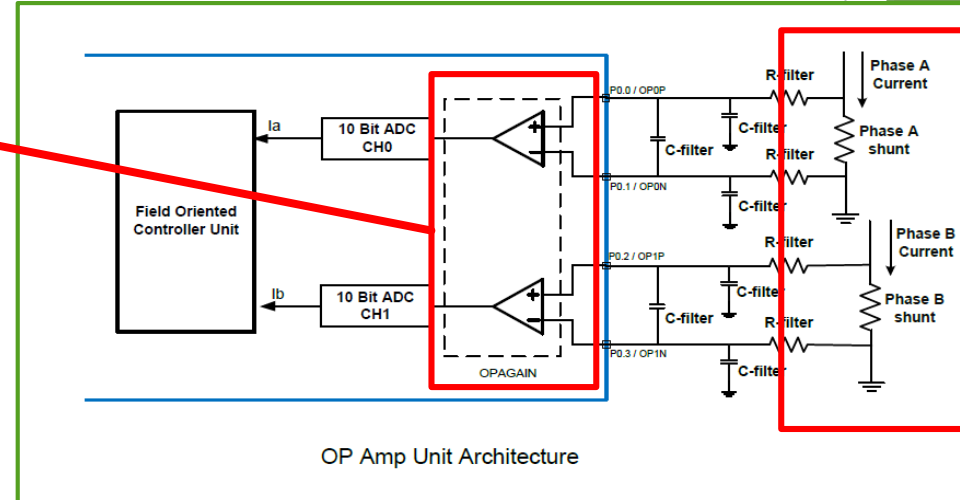
SMO_MaxSmcError : 32767

SMO_SMOGain : 327677

Motor Control adjustment process (3)

Set Rshunt and OPA_Gian	
Rshunt (unit : 0.1m Ohm)	1000
OPA_Gian	5 Gain

Fill in the Shunt resistor, and the amplifier magnification
Pay attention to the voltage difference of the Shunt
resistance in the design, and it must not exceed : $\pm 0.5V$



Motor Control adjustment process (4)

OpenLoop

- SmoPLL Forced angle start, need to manually connect the close-loop

CloseLoop

- When the close loop is connected, the system runs stably, which means that the motor parameters are correct, and can be set 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

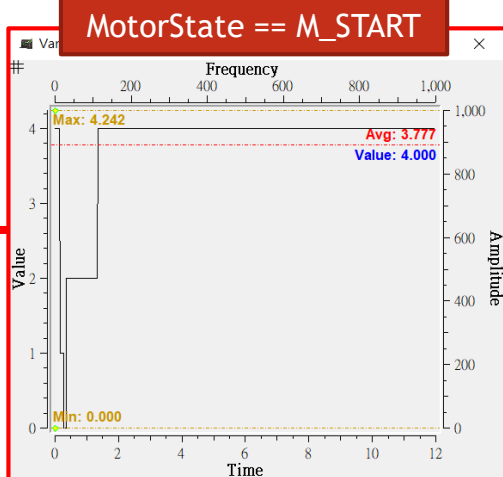
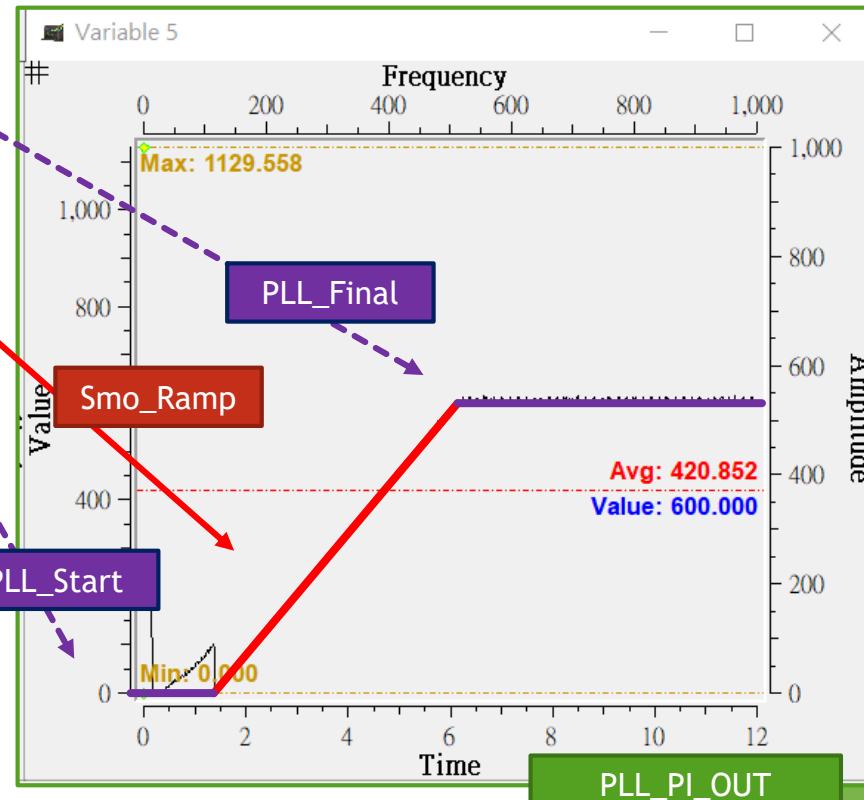
Set FOC LOOP Parameter	
IQ	
Set IQ Current parameter	
Set IQ Initial current (unit : mA)	0
Set IQ Starting current (unit : mA)	440
Set IQ End current (unit : mA)	450
IQ_TailWind Value (unit : mA)	500



Motor Control adjustment process (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. Setup Adjustment Process OpenLoop
2. Set open loop "start-end" speed
3. Set open loop "start-end" current

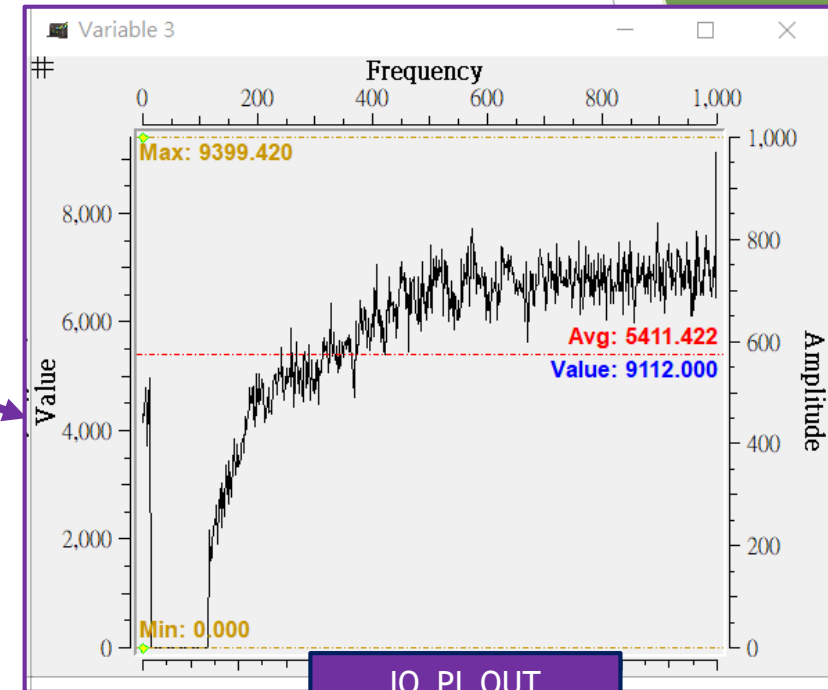


```
enum MotorStatus
{
    →M_OFF = 0,
    →M_INIT = 1,
    →M_TAILWIND = 2,
    →M_IPD = 3,
    →M_START = 4,
    →M_RUN = 5,
    →M_STOP = 6,
    →M_BREAK = 7,
    →M_ERROR = 8,
    →M_BMF_BREAK = 9
};
```

Motor Control adjustment process (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)	

Setup Openloop Starting current



IQ_PI_OUT

positioning time

Starting current

end current

Tailwind and
headwind starting
current

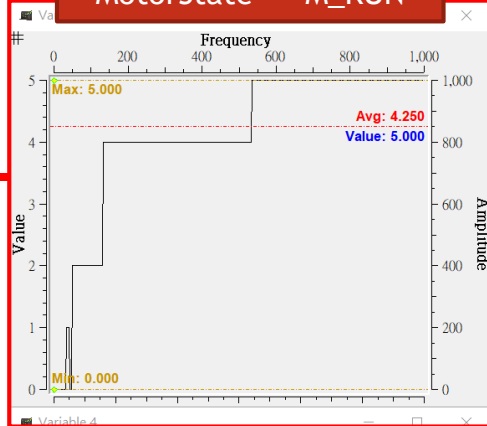


Motor Control adjustment process (7)

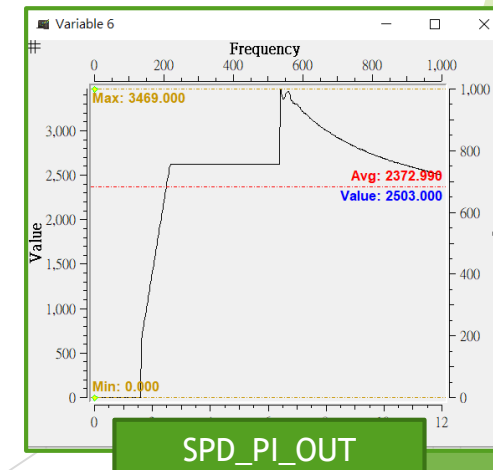
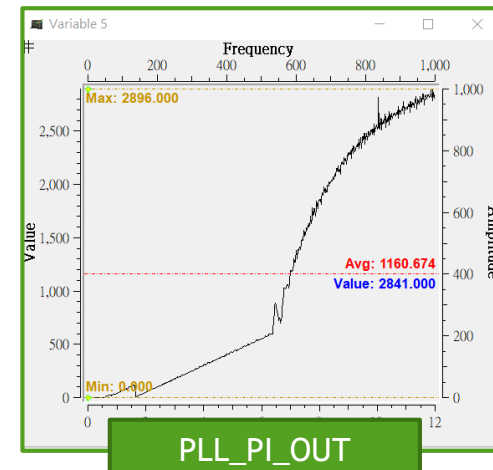
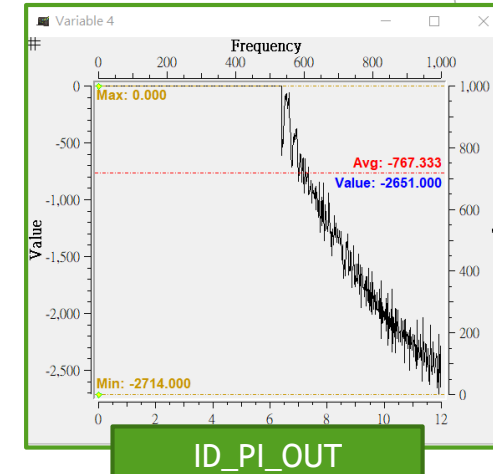
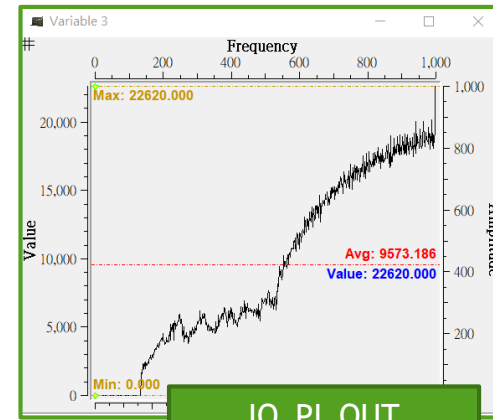
1. Open loop operation is completed
2. Setup Adjustment Process CloseLoop
3. Enter the closed-loop stage, and the machine adjustment is completed

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

MotorState == 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
};
```



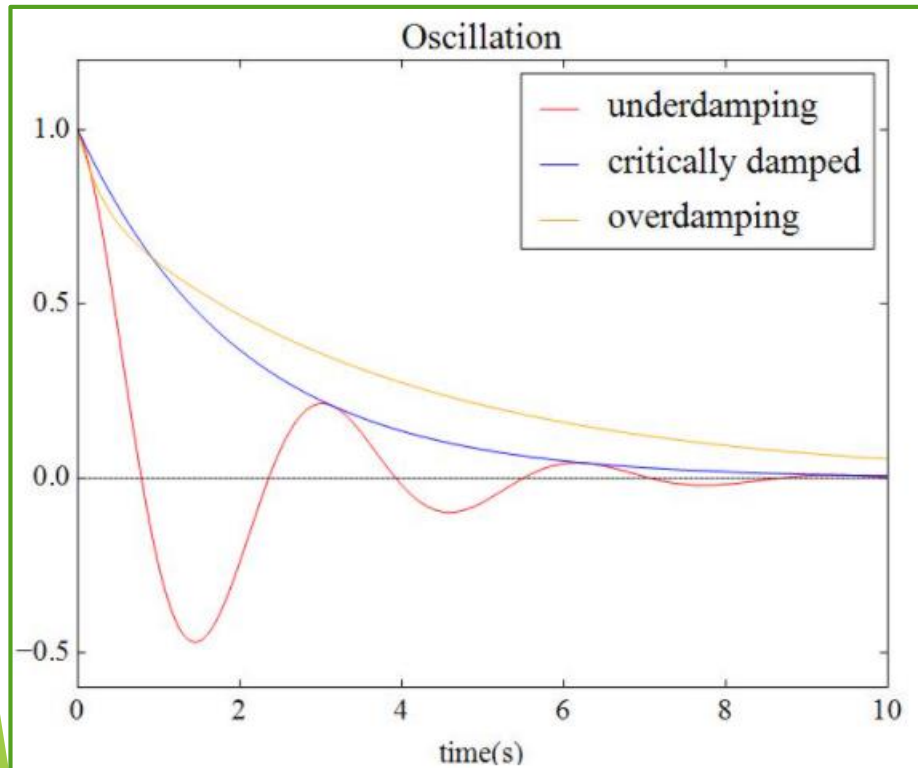
Motor Control adjustment process (8)

All PI controller parameters, fine-tuning according to different motor parameters

In automatic control theory, system responses are classified as follows:

1. **overdamping** : PI controller adjust direction · $K_p \uparrow$ · $K_i \downarrow$.
2. **Underdamping** : PI controller adjust direction · $K_p \downarrow$ · $K_i \uparrow$.
3. **critically damped** : K_p · K_i is ideal value .

K_t is the end parameter for inverse integration , default is 32767.



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

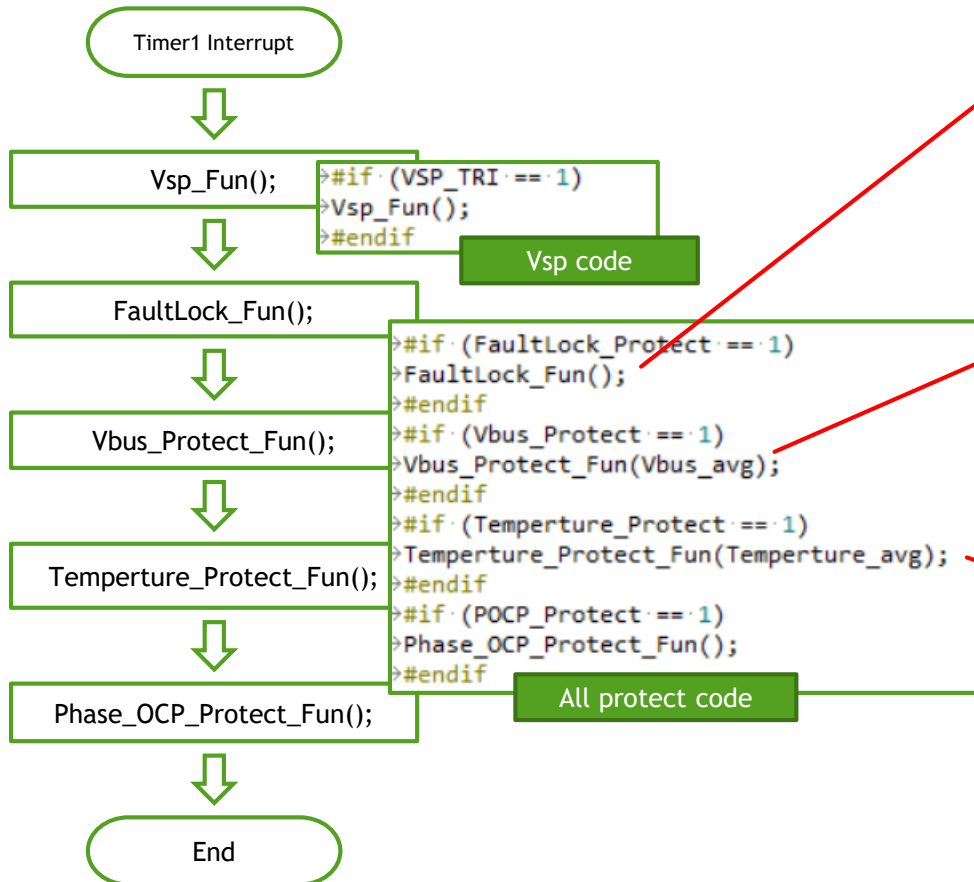
PLL	
Start Kp	1000
Final Kp	1000
Ki parameters	50
HeadWind/TailWind	
Kt parameters	32767
MaxLimit parameters	32767
MinLimit parameters	32767

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 adjustment process (9)

Motor protection functions



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

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

1. choose OVP_CH
2. Adjust the correct magnification
3. Adjust OVP



Motor Control adjustment process (10)

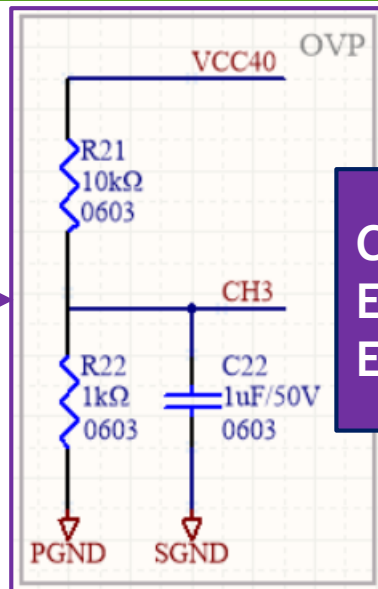
Motor protection functions

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. choose OVP_CH

2. Adjust the correct magnification

3. Adjust OVP



Calculate the Vbus magnification parameter, $V_{cc}=36V$, $AD_Val=669$

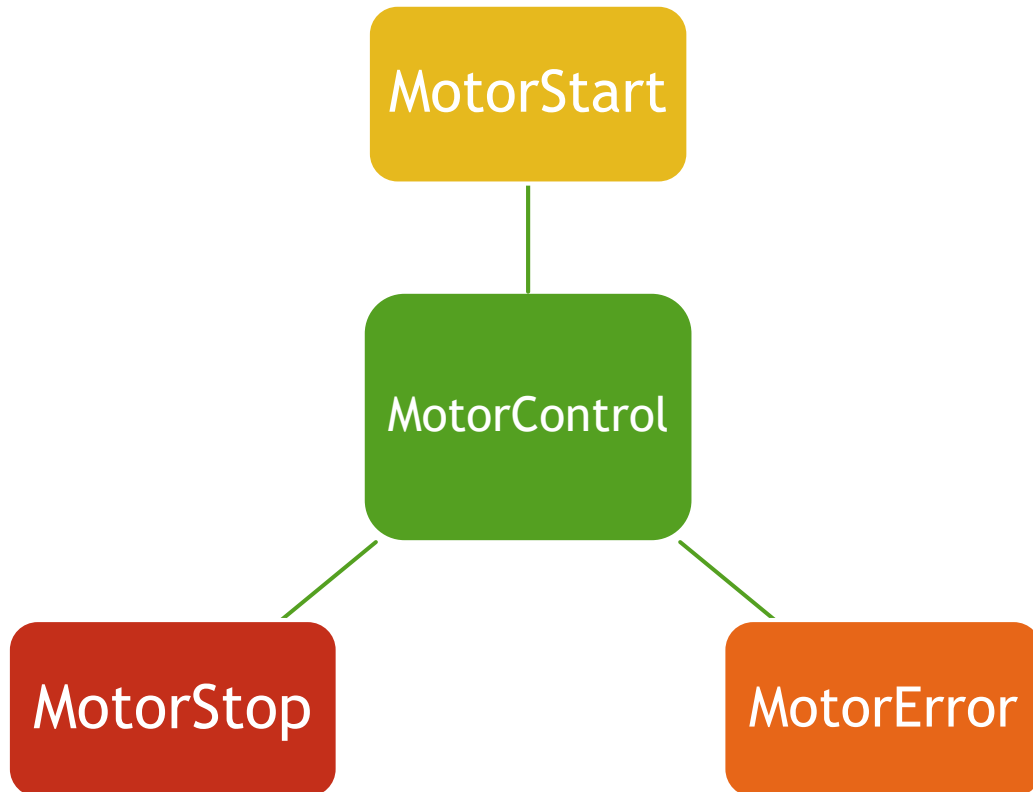
Ex1. Vbus magnification parameter $= 669/36 = 18.5833$

Ex2. Vbus magnification parameter $= ((R22/(R21+R22))/5)*1023 = 18.6$



Motor Control Program Flow (1)

Motor Driver Process



Motor Driver Process

```
void Motor_Control(void)
{
    → #if (CW_CCW_FUNCTION == 1)
    → if (CCWFlag != CCWFlagOld) // Forward and reverse control
    → → MotorState = M_INIT;
    → → CCWFlagOld = CCWFlag;
    → #endif

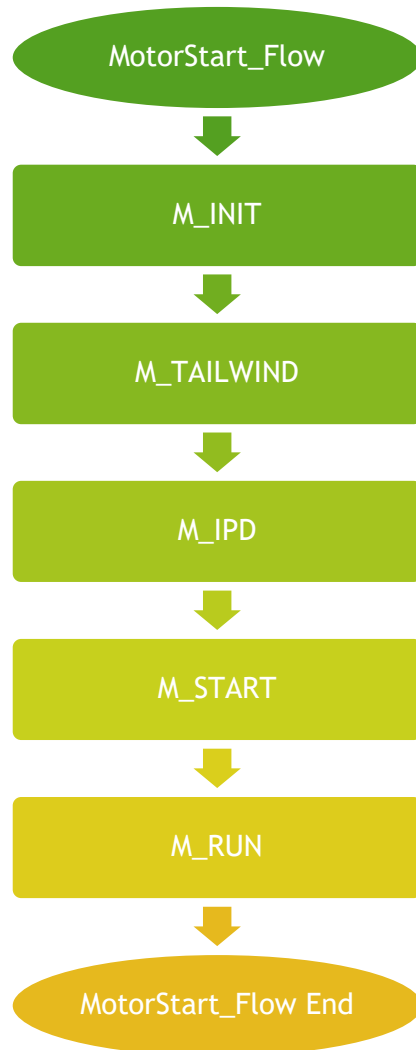
    → if (MotorErrorState)
    → {
    → → if (MotorState != M_OFF)
    → → → ResetMOC();
    → → → PWM_SetAlloff();
    → → → MotorStartRetry_Flow();
    → → → if ((FG_CTRL_REGS & 0x80) == 0x80)
    → → → FG_DISABLE;
    → → → #endif
    → }

    → else if ((SystemState & 0x04) == 0x04) // Start
    → {
    → → 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
    → }
}
```



Motor start code flow (1)



Motor start code

```
void MotorStart_Flow(void)
{
    switch (MotorState)
    {
        case M_INIT:
            MotorInit_Fun();
            if (CW CCW FUNCTION == 1) // Forward and reverse control
                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;
```



Tailwind and headwind start (1)

Accomplish tailwind and headwind start method :

1. Judgment (Two BEMF divided voltage)
2. Judgment (Diode BEMF divided voltage)

Accomplish tailwind start method :

1. Judgment (One BEMF divided voltage)

Need at least **two phase BEMF** can judgment **tailwind and headwind** , and judgment tailwind need **one phase**

Each block has a planning tailwind and headwind adjustment process :

LEVEL_1 : code stop to M_TAILWIND

LEVEL_2 : code run to 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

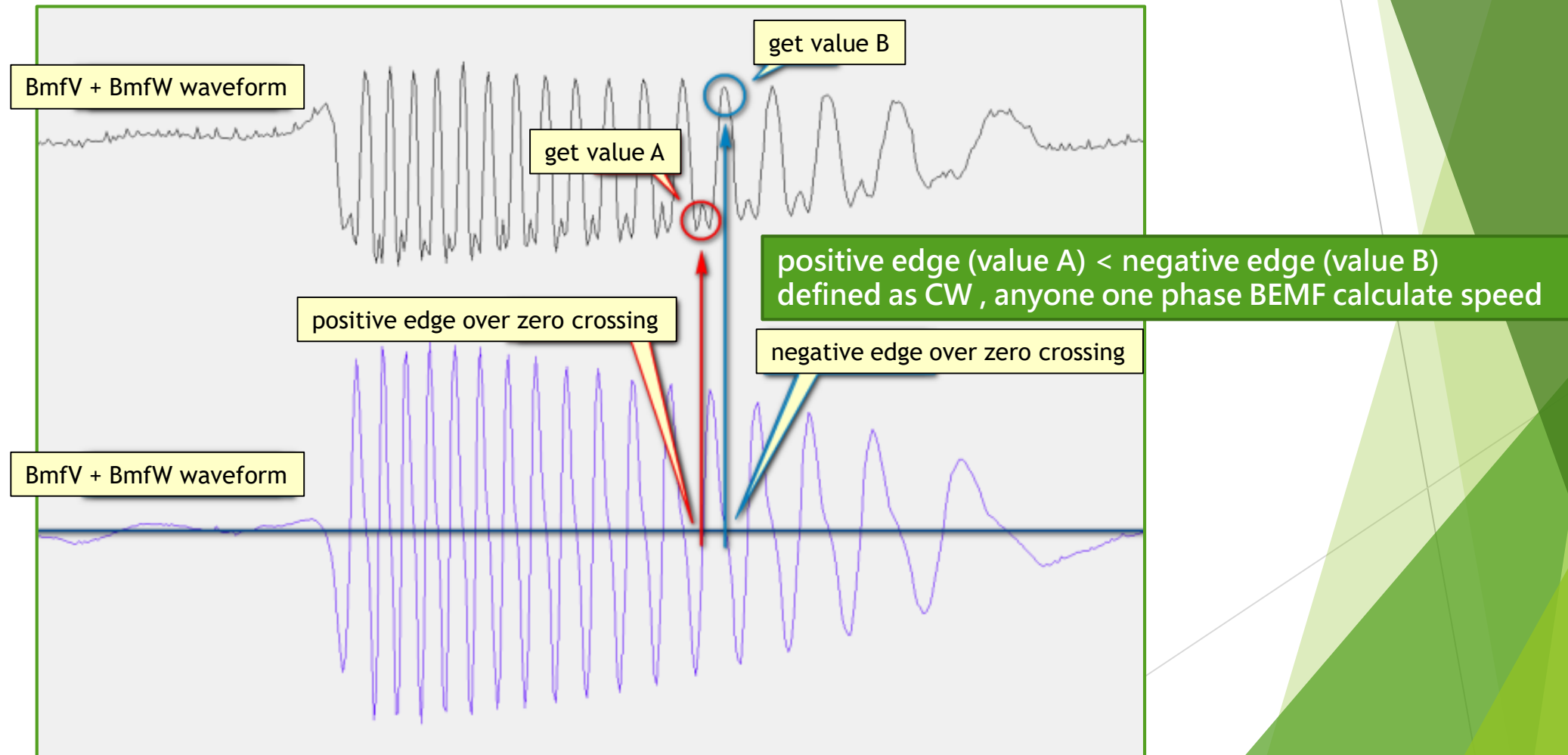
BEMF Fairwind/Headwind judgment (Diode) Enable/Disable	<input checked="" type="checkbox"/>
BEMF_TAILWIND_IQ_OUT_VALUE (unit : Val)	8000
BEMF_V_CH	CH4
BEMF_W_CH	CH5
BEMF_TAILWIND_SOP	LEVEL 2
BEMF_TAILWIND_SPEED_MAX (unit : 10rpm)	1200
BEMF_TAILWIND_SPEED_MIN (unit : 10rpm)	300
BEMF_HEADWIND_SPEED_MAX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



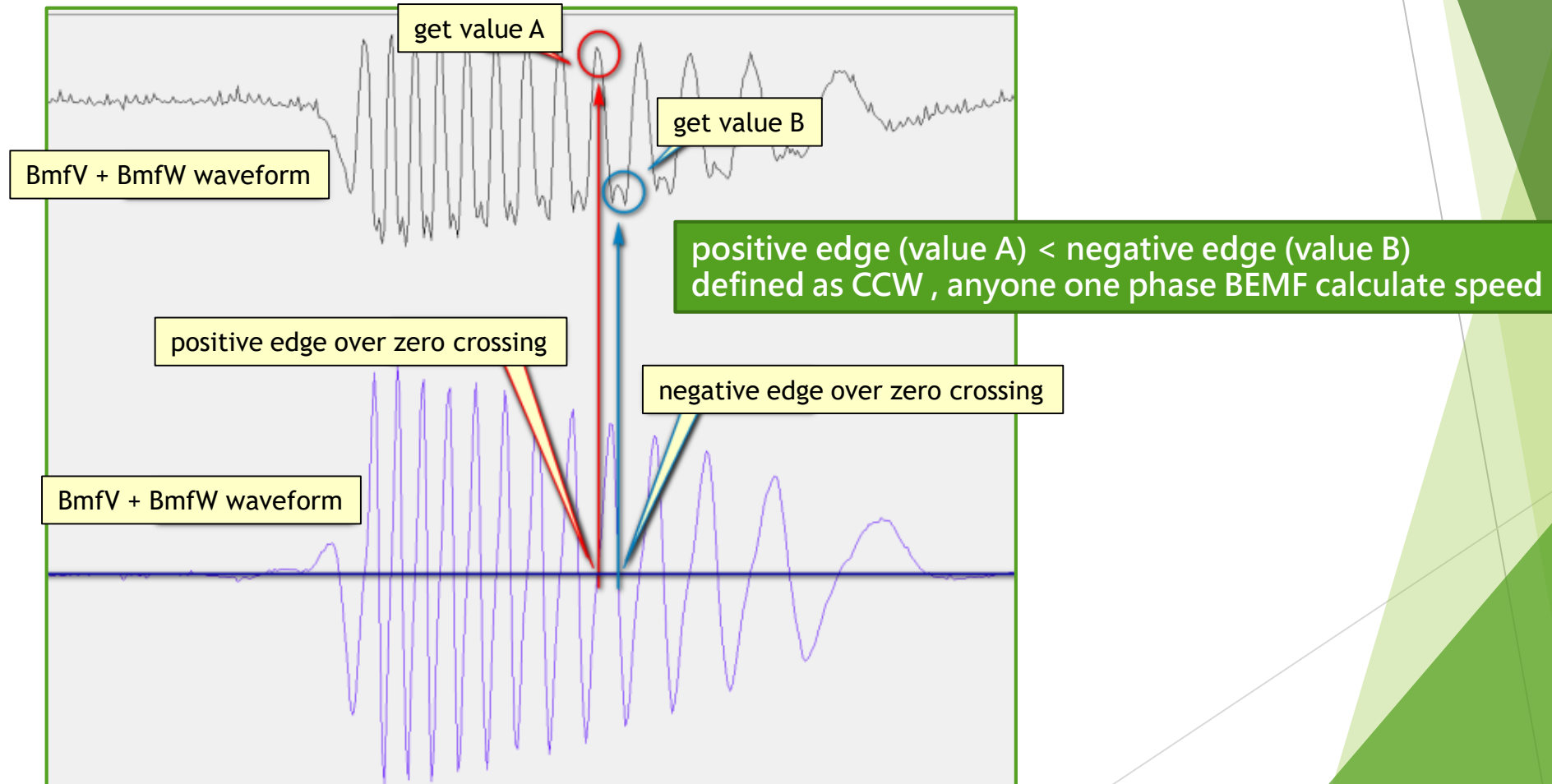
Tailwind and headwind start (2)

anyone two phase BEMF feedback method



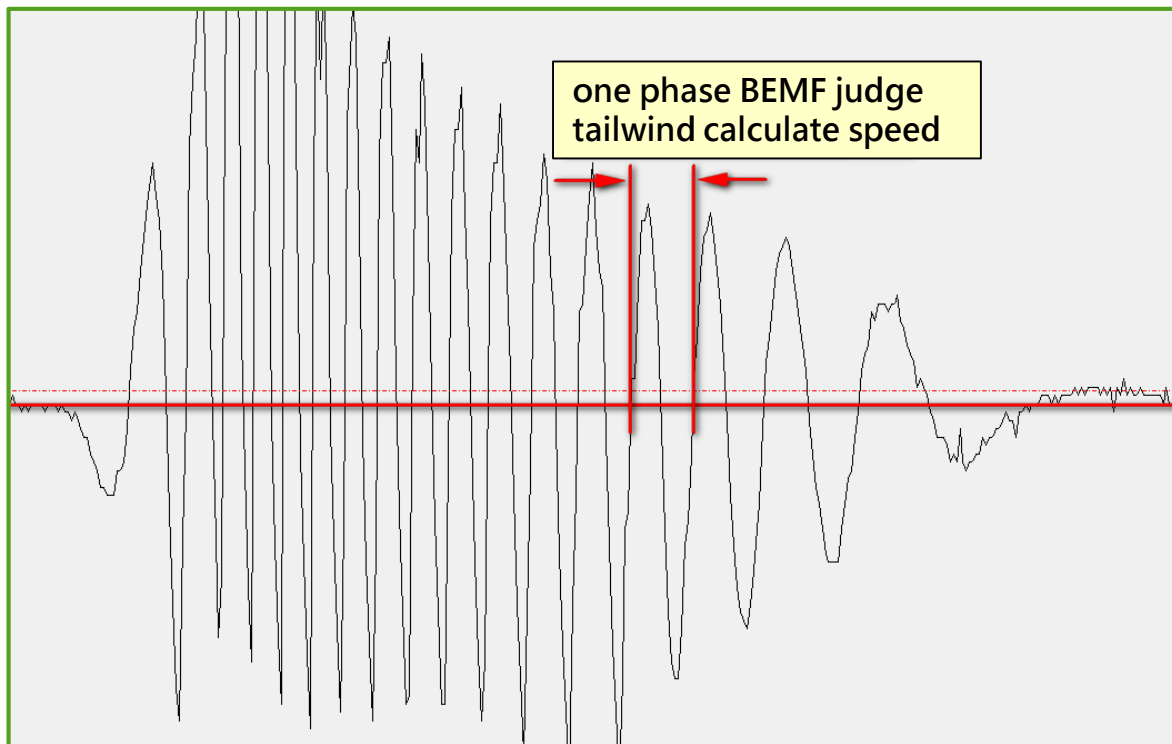
Tailwind and headwind start (3)

anyone two phase BEMF feedback method



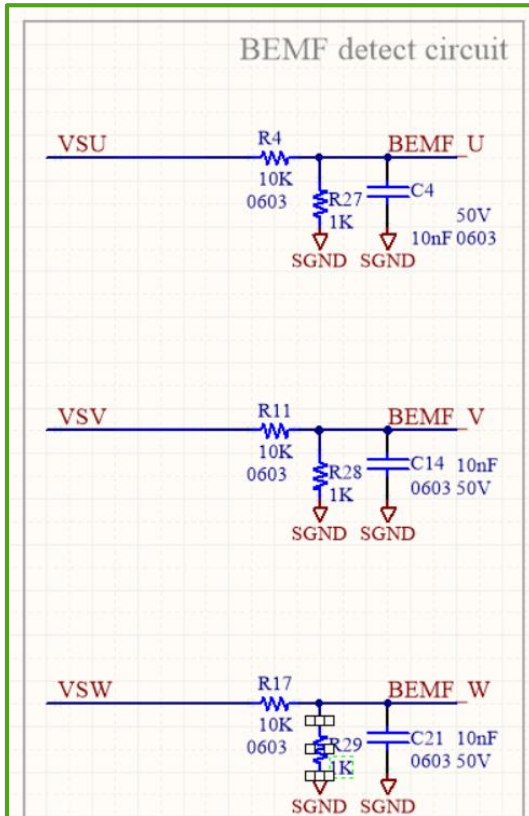
Tailwind and headwind start (4)

anyone one phase BEMF feedback method

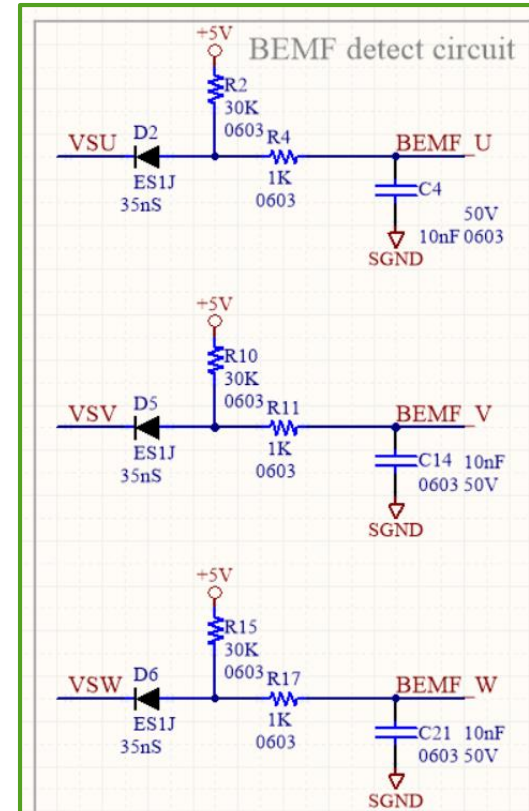


Tailwind and headwind start (5)

BEMF feedback circuit



Divider resistor : Readable each phase BEMF , but pay attention to the matching of divider resistor , avoid BEMF is too high , cause damage to the ADC pin .



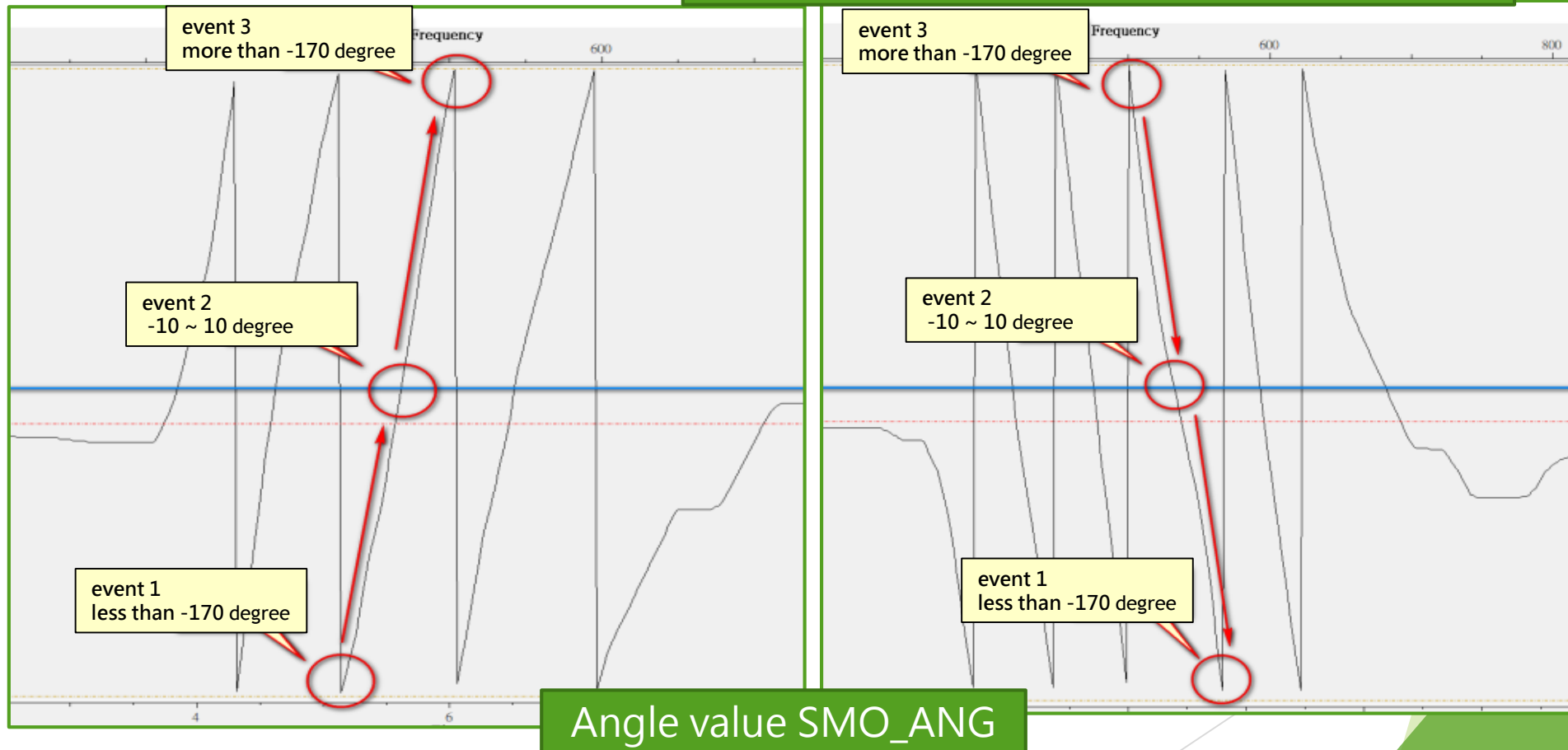
Diode divider resistor : circuit needs lower arm to use PWM brakes , can readable each phase BEMF .



Tailwind and headwind start (6)

Brake feedback circuit method

1. event 1 > event 2 > event 3 > defined as CW
2. event 3 > event 2 > event 1 > defined as CCW



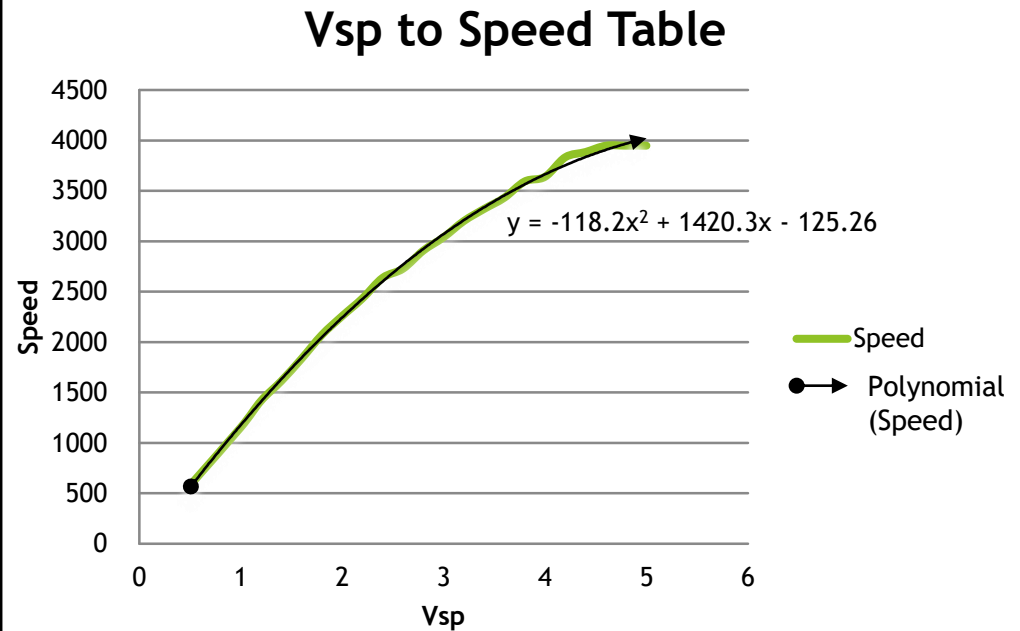
LookUpTable

LookUpTable_Fun

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 records”, proclaim LookupTable[5]

“proclaim LookupTable[5]” >> fill in “5”

Create table using differencing, Vsp - Speed Table completely construct the corresponding curve and display



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 : Not run PowerControl_Fun
LEVEL_2 : run 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

```

Only in CURRENT_CONTROL , can use POWER_CNOTROL .
#error Wrong setting POWER_CONTROL and CURRENT_CONTROL !!!



IPD code flow (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 AOCP 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)							

設定 AOCPCONT : IPD_LEVEL

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

choose IPD path

When the motor current slope is too slow , IPD_CYC need to lower :

1. current slope time < 1.3ms : IPD_CYC set 48MHz
2. current slope time < 2.6ms : IPD_CYC set 24MHz
3. current slope time < 5.2ms : IPD_CYC set 12MHz
4. current slope time < 10.4ms : IPD_CYC set 6MHz



IPD code flow (2)

IPD code flow recommended to implement step by step

```
void IPDDetect_Fun(void)
{
    #define IPDAdvanceAng 30
    switch (IPD_Detect_State)
    {
        case 0:
            AOCPCONT = IPD_LEVEL;
            Break_Fun(); // When used as IPD , N+N Gate Driver need lower arm pre-charge
            IPD_Cnt++;
            if (IPD_Cnt > 20)
            {
                IPD_Detect_State = 1;
                IPD_Cnt = 0;
            }
            break;
        case 1:
            //WatchDog_Disable(); // When used as IPD , it will trigger 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;
    }
}
```

AOCPCONT : IPD_LEVEL
Need to pre-set

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

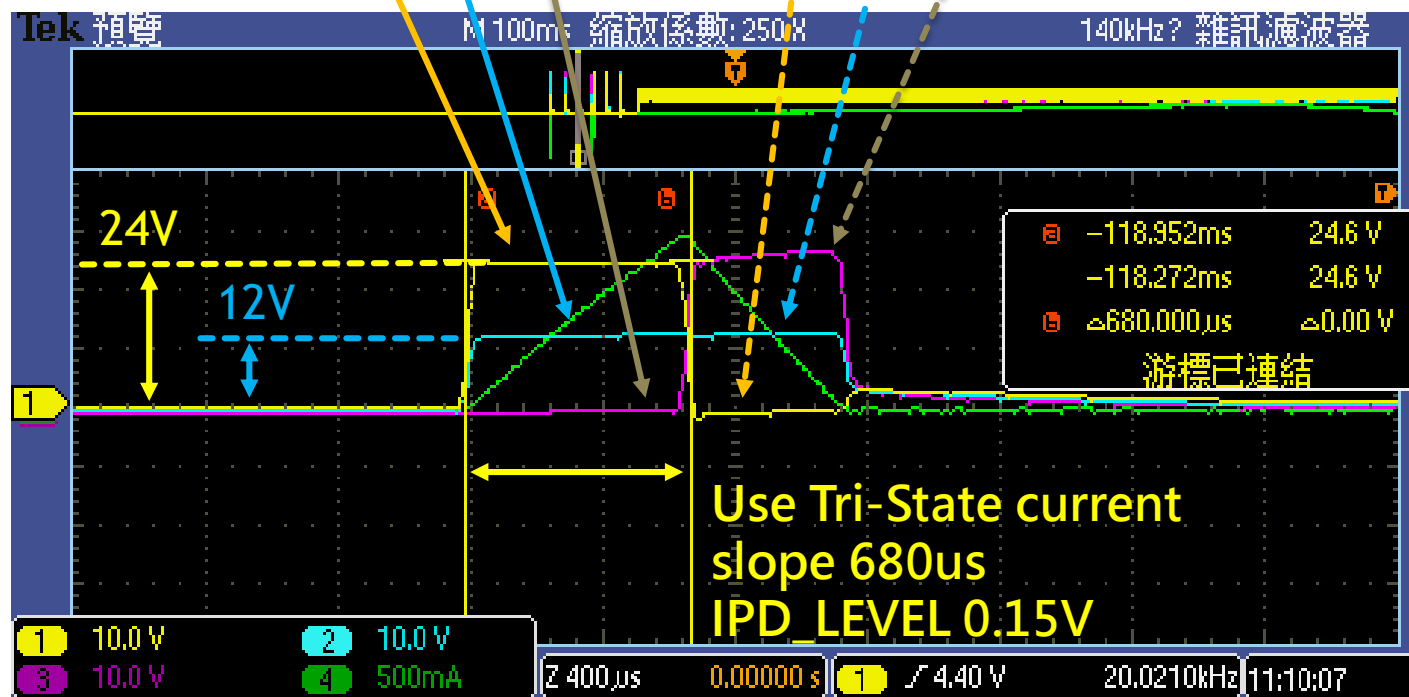
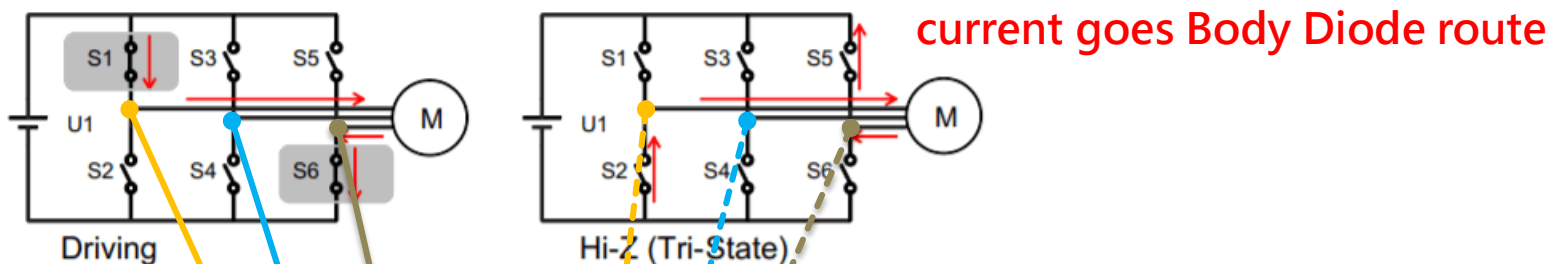
P+N 、N+N Gate Driver need lower arm pre-charge 20 ~ 30ms

When enabled IPD , need WatchDog Disable first , avoid triggering WatchDog during IPD

LatestTheta initial angle , can be fine-tuned for different motor
The current parameters are preset suggested values .
IPDAdvanceAng as phase advance adjustment .

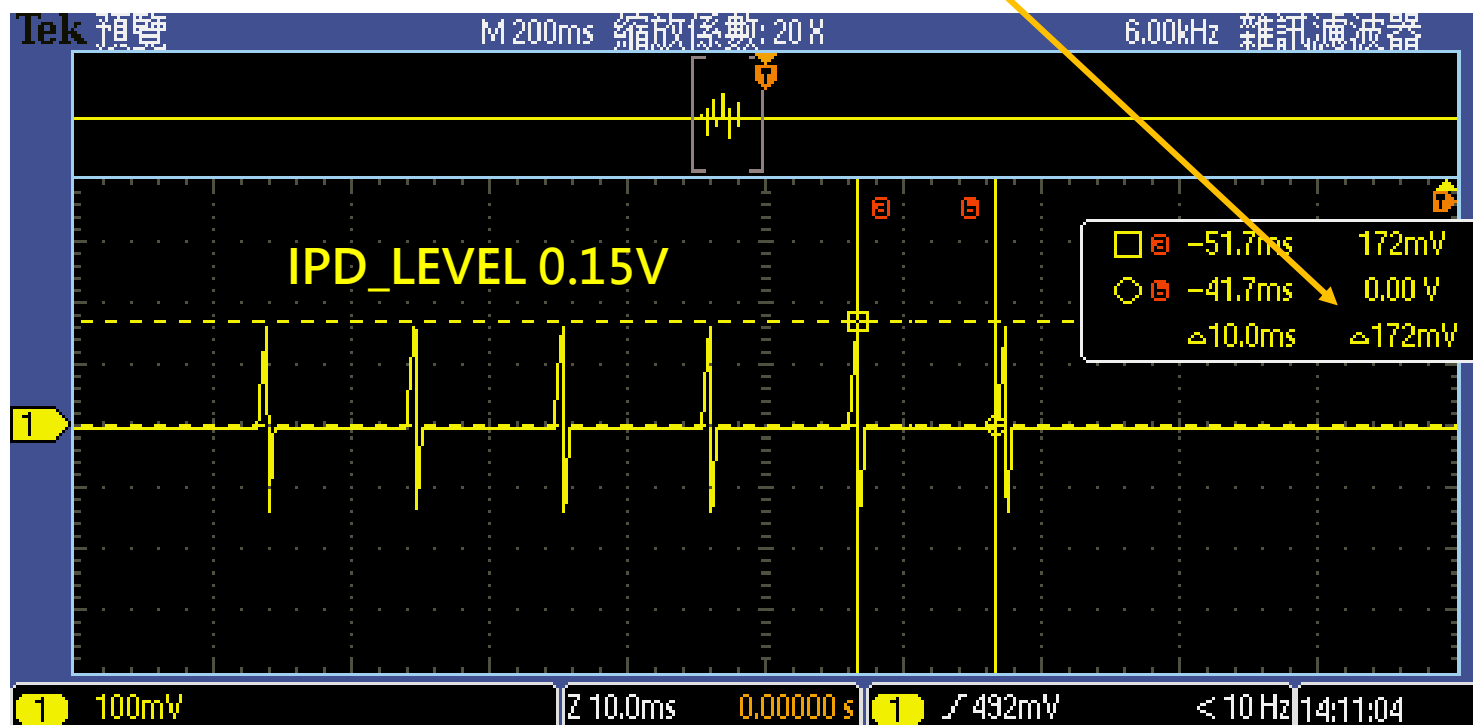


IPD code flow (3)



IPD code flow (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 application code support item

item	MDRFxx	MDSFxx	Note
Vsp control	○	○	
current control	○	○	
rotating speed control	○	○	
power control	○	○	Pay attention Code Size to use MDRFD0
power limit	○	○	Pay attention Code Size to use MDRFD0
forward and reverse control	○	○	
initial position detection	○	○	
Tailwind and headwind detection	○	○	MDRFD0 use Pwm interrupt calculation speed
IR Decoder	×	○	MDRFD0 need use Cap calculation decoding
Power off memory	×	○	MDRFD0 need external EEPROM

