Charles Dong

APPLICATION NOTE

PAC52xx BLDC

PAC 配置及固件流程

Power Application Controllers

目录

1 PAC52xx 配置

针对 BLDC 算法，PAC52xx 芯片外设资源配置及初始化讲解，分为模拟部分和数字部分，以下以 PAC5223 为例，架构框图如下，橙色为数字核，绿色为模拟核；

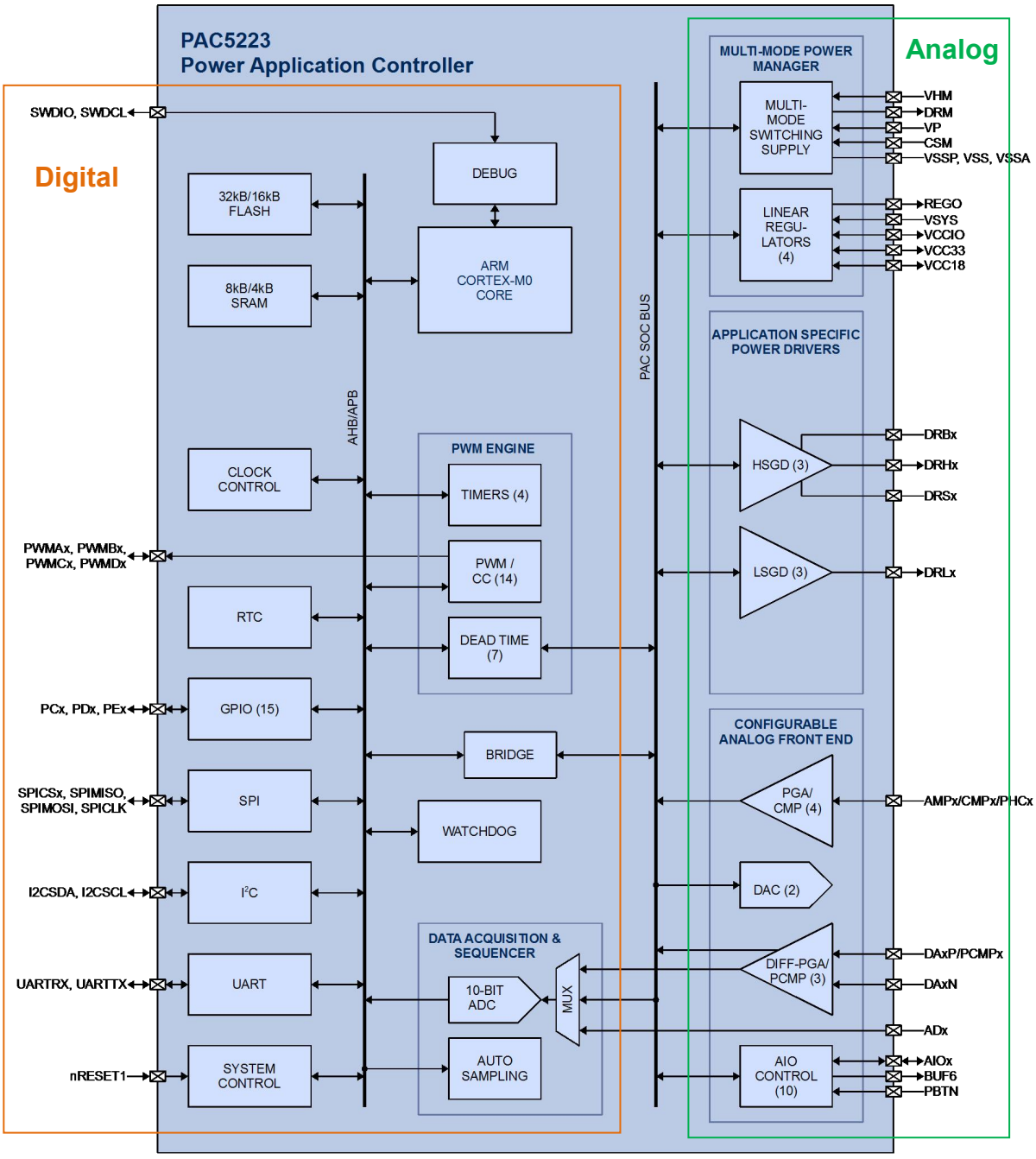


图 1-1：架构框图

1.1 模拟部分

电源管理模块(可参考文档 DS_PAC5223)

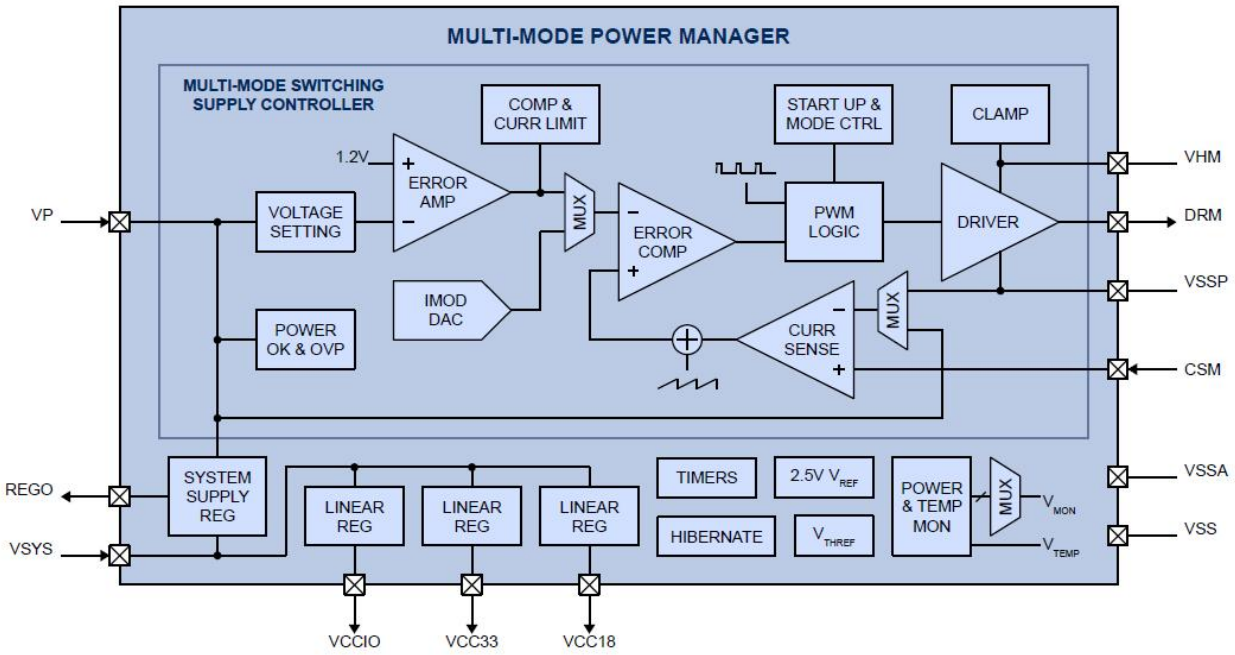


图 1-2 电源管理模块框图

Buck 模式，降压方式产生供电电源 VP（可程序设定为 5V、9V、12V、15V）。

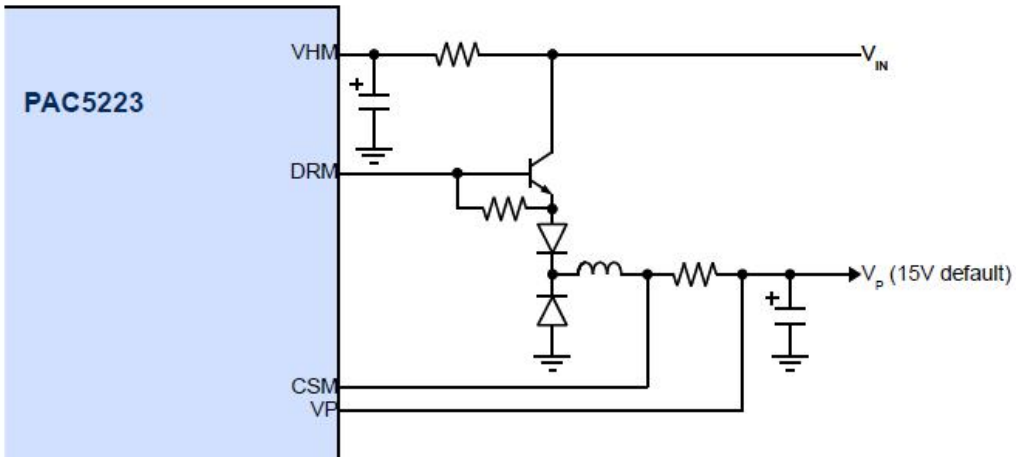


图 1-3 Buck 模式

反激模式，降压方式产生供电电源 VP（可程序设定为 5V、9V、12V、15V）。

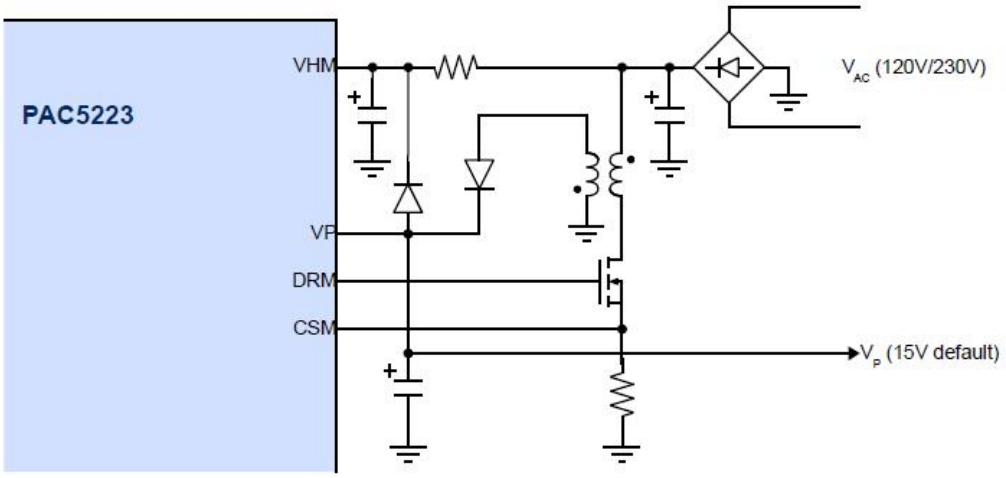


图1-4 反激模式

线性稳压电源，由VP经过LDO降压产生5V(VSYS)，可提供200mA电流。在REGO和VSYS引脚之间串联一个适当的功率电阻，可以分担内部稳压电源的功耗从而降低芯片温度。当VSYS 超过4V 后，另外三个附加的40mA输出能力的线性电源VCCIO（默认3.3V，如需要配置成5V，只需将VCCIO与VSYS短接即可）、VCC33（3.3V）和VCC18（1.8V）即开始工作。

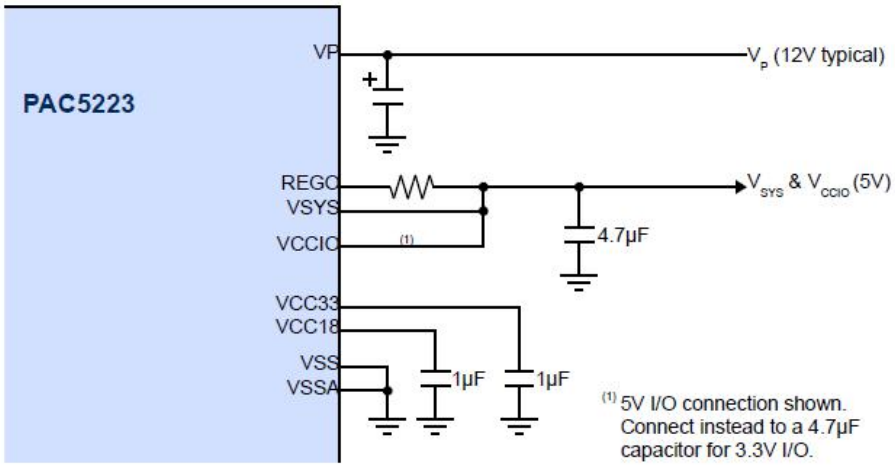


图1-5 线性稳压

注意：如果外部能提供一个稳定的 9V 到 12V 的电源，用户也可以直接将其连接到 Vp 上，这时所有线性电源都可以脱离开关电源管理模块系统工作。在这样的条件下，VHM 脚可以直接与 Vp 连接，并且 MCU 可以在工作时将开关电源管理模块禁用，可以

减少功耗。PAC52xx 有两种低功耗模式,一种是让数字核工作在很低主频下的睡眠模式，一种是通过多功能电源模块关断各电源部分的冬眠模式，在冬眠模式下，仅仅在 VHM 上有很小的电流，冬眠模式可由 AIO6 引脚保持 32ms 低电平而唤醒，唤醒后，芯片重新上电、复位，重新初始化。芯片内部 ADC 参考电压为 2.5V，差分运算放大器可设 1.25V 电压偏置。

模拟前端模块

模拟部分初始化在程序初始化函数里模拟前端初始化部分“**void cafe_init(void)**”中，要对模拟前端进行配置前，必须保证片上系统总线桥是使能的，如下语句：

```
      pac5xxx_tile_socbridge_config(1, 0);

      pac5xxx_tile_register_write(ADDR_DEVID,
0x55);
if (pac5xxx_tile_register_read(ADDR_PWRSTAT))
      pac5xxx_tile_register_write(ADDR_PWRSTAT,
0xFF);

      pac5xxx_tile_register_write(ADDR_PWRCTL,
0x40);
      pac5xxx_tile_register_write(ADDR_PSTATSET
, 0x80);
      pac5xxx_tile_register_write(ADDR_SCFG,
0x10);
      pac5xxx_tile_register_write(ADDR_IMOD,
0xFF);
      pac5xxx_tile_register_write(0x15, 0x80);

//0x15对应的寄存器是配置VP电压的，
0xC0→15V,0x80→12V,0x40→9V,0x20→5V
```

//使能 SOC 总线桥

ADDR_DEVID、
ADDR_PWRSTAT 、
ADDR_PWRCTL、
ADDR_PSTATSET、
ADDR_SCFG、
ADDR_IMOD、

这些寄存器的配置这里不用更改，如左边配置就可，寄存器各位的含义可查阅“PAC5250_UG”文件

模拟前端，运算放大器、比较器部分，与 ADC 配合使用(可参考文档 DS_PAC5250)，如下图

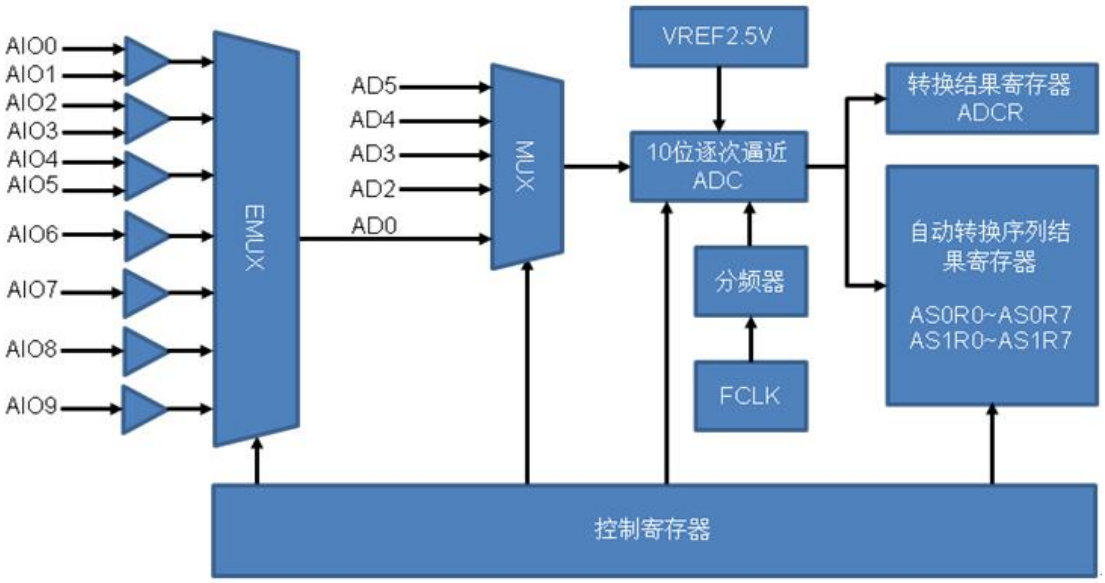
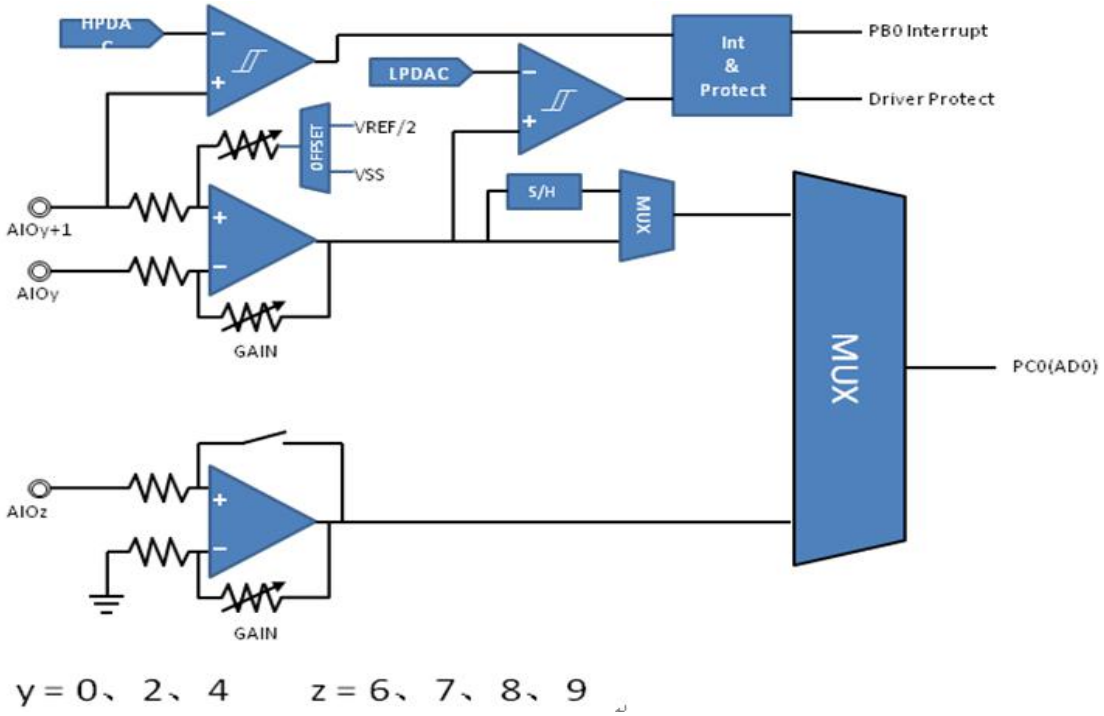


图1-6 ADC采样序列结构框图

AIO 与比较器：



y = 0、 2、 4　　z = 6、 7、 8、 9

图1-7　AIO与比较器框图

在模拟信号进 ADC 转换器前置了 3 路差分运算放大器（AIO10、AIO32、AIO54），其中在差分运算放大器的同相输入端（AIO1，AIO3，AIO5）和输出端上各加了比较器，用以作为信号保护；4 路同相运算放大器（AIO6、AIO7、AIO8、AIO9），其中 AIO7、AIO8、AIO9 带相位比较器功能，AIO6 可作为冬眠模式唤醒触发引脚。这里将模拟前端（AIO）配置结合 ADC 一起说明，以下是差分运算放大器（AIO10、AIO32、AIO54）的配置代码，

```
      #ifdef LP_AOI_PROTECT

            pac5xxx_tile_register_write(ADDR_CFGAIO4, 0x5B);     //Charles PAC5250
LPROT45 Blanking time 2us

            pac5xxx_tile_register_write(ADDR_CFGAIO5, 0x30);     //Charles PAC5250
LPROT45

            pac5xxx_tile_register_write(ADDR_PROTINTM, 0x04);    //LPROT45

      #else

            pac5xxx_tile_register_write(ADDR_CFGAIO4, 0x58);     //Charles PAC5220
HPROT10  //0x48-->X1  0x58-->X4

            pac5xxx_tile_register_write(ADDR_CFGAIO5, 0xC3);     //Charles PAC5220
LPROT10-->0x30  HPROT10-->0xC3

            pac5xxx_tile_register_write(ADDR_PROTINTM, 0x40);    //LPROT10-->0x01
HPROT10-->0x10

      #endif
```

以上 *ADDR_CFGAIO0*、*ADDR_CFGAIO2*、*ADDR_CFGAIO4* 寄存器各位含义如下表：

BIT	I/O MODE	DIFFAMP MODE	RFU	RFU
MODEy<1>	0	0	1	1
MODEy<0>	0	1	0	1
OPTy<1>	Option Setting: 00b = Input, 01b = Output, 10b = Open-Drain, 11b = Wk OD	**GAINy<2:0>**: 0 = 1x, 1=1x, 2 = 2x, 3 = 4x, 4 = 8x, 5 = 16x, 6 = 32x, 7 = 48x	RFU	RFU
OPTy<0>			RFU	RFU
POLy	Polarity Setting: 0 = Active High, 1 = Active Low		RFU	RFU
MUXy<2>	Digital Mux: 000b = DATAIOy, 111b..001b = DB<7:1>.	RFU	RFU	RFU
MUXy<1>		**LPOPTy<1:0>**: 00b = LPCOMP Disabled, 01b = LPCOMP Enabled w/ 0.5µs Blanking, 10b = w/ 1µs, 11b = w/ 2µs	RFU	RFU
MUXy<0>			RFU	RFU

以上 *ADDR_CFGAIO1*、*ADDR_CFGAIO3*、*ADDR_CFGAIO5* 寄存器各位含义如下表：

BIT	I/O MODE	DIFFAMP MODE	RFU	RFU
-	-	**nHP(y+1)yPR1M**: HPROT PR1 Mask	-	-
-	-	**nHPy+1)yPR2M**: HPROT PR2 Mask	-	-
OPT(y+1)<1>	Option Setting: 00b = Input, 01b = Output, 10b = Open-Drain, 11b = Wk OD	**nLP(y+1)yPR1M**: LPROT PR1 Mask	RFU	RFU
OPT(y+1)<0>		**nLP(y+1)yPR2M**: LPROT PR2 Mask	RFU	RFU
POL(y+1)	Polarity Setting: 0 = Active High, 1 = Active Low	**ENOSy**: Enable Differential Amp Offset	RFU	RFU
MUX(y+1)<2>	Digital Mux: 000b = DATAIO(y+1), 111b..001b = DB<7:1>.	**CALy**: Differential Amp Calibration Mode	RFU	RFU
MUX(y+1)<1>		**HPOPTy<1:0>**: 00b = HPCOMP Disabled, 01b = HPCOMP Enabled w/ 0.5µs Blanking, 10b = w/ 1µs, 11b = w/ 2µs	RFU	RFU
MUX(y+1)<0>			RFU	RFU

以下是 AIO6、AIO7、AIO8、AIO9 的配置代码：

```
pac5xxx_tile_register_write(ADDR_CFGAIO6, 0x44);

pac5xxx_tile_register_write(ADDR_CFGAIO7, 0xD0);

pac5xxx_tile_register_write(ADDR_CFGAIO8, 0xD0);

pac5xxx_tile_register_write(ADDR_CFGAIO9, SLCOMP7
| 0x80);
```

AIO6 被配置到模拟总线 AB4 上；AIO7、AIO8、AIO9 都被配置为特殊模式，作为相比较器使用,三个比较器可分别找三相相电压的过零点

以上 *ADDR_CFGAIO6*、*ADDR_CFGAIO7* 、*ADDR_CFGAIO8*、*ADDR_CFGAIO9* 寄存器各位含义如下表：

BIT	I/O MODE	GAIN MODE	COMPARATOR MODE	SPECIAL MODE
MODEx<1>	0	0	1	1
MODEx<0>	0	1	0	1
OPTx<1>	Option Setting: 00b = Input, 01b = Output, 10b = Open-Drain, 11b = Wk OD	Gain Setting: 0 = Direct Mux; 1= 1x, 2 = 2x, 3 = 4x, 4 = 8x, 5 =16x, 6 = 32x, 7 = 48x	Reference Select: 00b = VTHREF, 01b = AB1, 10b = AB2, 11b = AB3.	TBD
OPTx<0>				TBD
POLx	Polarity Setting: 0 = Active High, 1 = Active Low		Polarity Setting: 0 = Active High, 1 = Active Low	Polarity Setting: 0 = Active High, 1 = Active Low
MUXx<2>	Digital Mux: 000b = DATAIOx, 111b..001b = DB<7:1>.	Analog Mux: 000 = ABx, 111..001 = AB<7:1>.	Digital Mux: 000b = DATAIOx, 001b = DBx, 111b...001b = DB<7:1>.	
MUXx<1>				
MUXx<0>				

以下是 ADC 序列配置代码（详细说明请参考 ADC 应用文档）：

```
        pac5xxx_adc_as0_config_pwm(ADCCTLX_AS1D_DEPTH6,
ADCCTLX_TRIGEDGE_HITOLOW, ADCCTLX_TRIGPWM_PWMA0);

        pac5xxx_adc_as0_sequence_config(0, ADCCTL_ADMUX_AD0,
ASSEQ_MSPI_AFTER_SH, SIGMGR_MSPI(ADC_SEQ_P_UIN_EDATA_REG), ASSEQ_DELAY_0);

        pac5xxx_adc_as0_sequence_config(1, ADCCTL_ADMUX_AD0,
ASSEQ_MSPI_AFTER_SH, SIGMGR_MSPI(ADC_SEQ_P_VIN_EDATA_REG), ASSEQ_DELAY_0);

        pac5xxx_adc_as0_sequence_config(2, ADCCTL_ADMUX_AD0,
ASSEQ_MSPI_AFTER_SH, SIGMGR_MSPI(ADC_SEQ_P_WIN_EDATA_REG), ASSEQ_DELAY_0);

        pac5xxx_adc_as0_sequence_config(3, ADCCTL_ADMUX_AD0,
ASSEQ_MSPI_AFTER_SH, SIGMGR_MSPI(ADC_SEQ_VOUT_EDATA_REG), ASSEQ_DELAY_0);

        pac5xxx_adc_as0_sequence_config(4, VBUS_ADC_SAMPLE_CHANEL,
ASSEQ_MSPI_AFTER_SH, 0x40, ASSEQ_DELAY_0);

        pac5xxx_adc_as0_sequence_config(5, ADCCTL_ADMUX_AD5,
ASSEQ_MSPI_AFTER_SH, SIGMGR_MSPI(ADC_SEQ_I_TOTAL_EDATA), ASSEQ_DELAY_0);
```

注意：PC 引脚及 AIO 引脚作为 ADC 输入时，电压不要超过 2.5V，作为普通 IO 是电压不要超过 3.3V。

下是过流保护配置代码（详细说明请参开 ADC 应用文档）：

```
#ifdef LP_AOI_PROTECT

      pac5xxx_tile_register_write(ADDR_LPDAC0, (uint8_t)((0X3FF >> 2) & 0xFF));

      pac5xxx_tile_register_write(ADDR_LPDAC1, (uint8_t)((0X3FF & 0x3)));

#else

      pac5xxx_tile_register_write(ADDR_HPDAC, 0XF0);

#endif
```

过流保护的配置是在差分运算放大器的配置（寄存器 *ADDR_CFGAIO1*、*ADDR_CFGAIO3*、*ADDR_CFGAIO5*）中设置的，这里设置的是 HP 和 LP 的保护阀值点，也就是保护比较器的反相输入端电压（连接到芯片内部 DAC）。

HP 为 8 位 DAC（由寄存器 *ADDR_HPDAC* 设置），LP 为 10 为 DAC（由寄存器 *ADDR_LPDAC0*（高八位）、*ADDR_LPDAC1*（低两位）设置）。

模拟部分其他配置可按参考方案配置，详细内容可参考 PAC5250_UG 文件。

1.2 数字部分

ADC 的配置与其他 MCU 有很大的差别，PAC52xx 内部有一个 10 位的 ADC 转换器，但 ADC 输入通道有多路，且可分为两个序列，以上结合 AIO 模拟前端部分已有简单说明，这里简单说明 ADC 的时钟源，如下代码，详细请参考 AIO 应用文档。

```
pac5xxx_adc_emux_config(ADCEMUXCTL_EMUXC_SEQ,
ADCEMUXCTL_EMUXDIV_DIV2);
pac5xxx_adc_config_emux_io();

pac5xxx_adc_config(ADCCTL_ADSTART_AUTO_01_IND_TRIG,
ADCCTL_ADCDIV_DIV4, 1);
pac5xxx_adc_config_io(ADC_CHANNEL_MASK);

adc_sequence_config();
pac5xxx_timer_cctrl_int_enable(TimerA, 0, 1);

NVIC_EnableIRQ(TimerA_IRQn);
NVIC_SetPriority(TimerA_IRQn,0);
pac5xxx_adc_int_enable_as0(1);
NVIC_EnableIRQ(ADC_IRQn);

NVIC_SetPriority(ADC_IRQn,1);
pac5xxx_adc_enable(1);
```

有两个时钟需配置，一个是多路选择通道 EMUX 的时钟，一个是自动转换序列的时钟。按以上代码配置就可行。以上 AIO 部分讲过，选择 PWMA0 的下降沿作为触发源，这里需对 PWMA0 配置占空比，需注意触发 ADC 转化的时间点要避过开关噪音。以上还配置了 ADC 中断。

GPIO 有上拉、下拉、推挽等模式，时钟系统这里用内部时钟 PLL 到 50MHz,UART 等详细资料可参考 DS_PAC5250 和 PAC5250_UG;

这里重点说明定时器 PWM 与预驱动的配合使用，如下代码，详细请参考“*PAC SDK Training*” PPT 文件。

```
pac5xxx_timer_a_clear_assert();

pac5xxx_timer_io_select_pwma4_pa3();//定时器pwma4对应到pa3引脚，而pa6对应的是DRH3
pac5xxx_timer_io_select_pwma5_pa4();//定时器pwma5对应到pa4引脚，而pa7对应的是DRH4
pac5xxx_timer_io_select_pwma6_pd5();//定时器pwma6对应到pa5引脚，而pd7对应的是DRH5
pac5xxx_timer_io_select_pwma0_pa0();//定时器pwma0对应到pa0引脚，而pa0对应的是DRL0
pac5xxx_timer_io_select_pwma1_pa1();//定时器pwma1对应到pa1引脚，而pa1对应的是DRL1
pac5xxx_timer_io_select_pwma2_pa2();//定时器pwma2对应到pa2引脚，而pa2对应的是DRL2

m1_disable();
//以下配成三组死区互补模式，并设置前沿、后延死区时间
pac5xxx_dtg_config(DTGA0, MOTOR1_LED_TICKS, MOTOR1_TED_TICKS, 0, 0, 0, 0);
pac5xxx_dtg_config(DTGA1, MOTOR1_LED_TICKS, MOTOR1_TED_TICKS, 0, 0, 0, 0);
pac5xxx_dtg_config(DTGA2, MOTOR1_LED_TICKS, MOTOR1_TED_TICKS, 0, 0, 0, 0);
//以下是选择TimerA时钟源及计数方式，和PWM频率
pac5xxx_timer_clock_config(TimerA, TxCTL_CS_ACLK, TxCTL_PS_DIV1);
pac5xxx_timer_base_config(TimerA, TIMERA_TICKS, 0, TxCTL_MODE_UPDOWN, 0);
//以下分别设置pwma4、pwma5 、pwma6占空比， pwma0、pwma1 、pwma2分别是其带死区互补
pac5xxx_timer_cctrl_config(TimerA, 4, (TIMERA_TICKS>>1), 0);

pac5xxx_timer_cctrl_config(TimerA, 5, (TIMERA_TICKS>>1), 0);

pac5xxx_timer_cctrl_config(TimerA, 6, (TIMERA_TICKS>>1), 0);
//使能输出

pac5xxx_gpio_out_enable_a(0x3F);
```

计数模式、死区方式、PWM 与引脚对应关系图：

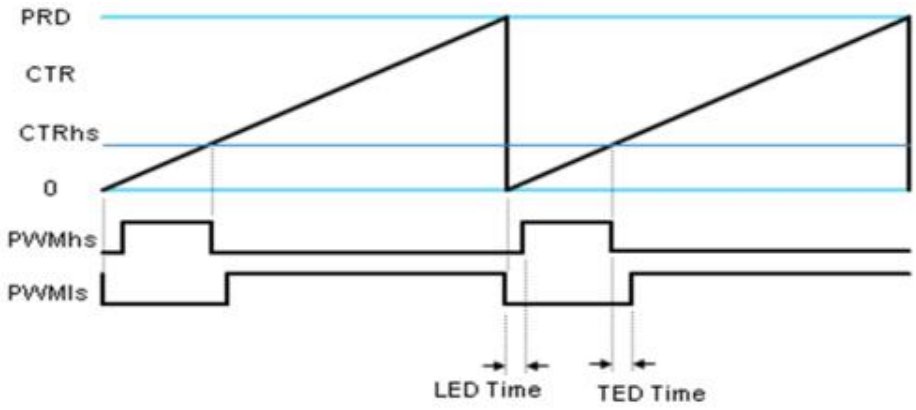


图 1-9 TIMER 配置成 UP 模式

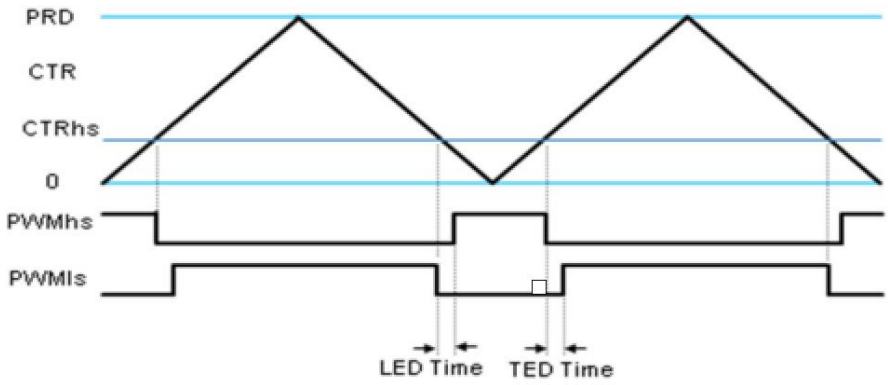


图 1-10 TIMER 配置成 UP-DOWN 模式

GPIO	PA0	PA1	PA2	PA3	PA4	PA5	PA6	PA7	PD7
PWM	PWMA0	PWMA1	PWMA2	PWMA3/ PWMA4/ PWMB0	PWMA5/ PWMC0	PWMA6/ PWMD0	PWMA4/ PWMB0	PWMA5/ PWMA7/ PWMC1	PWMA6/ PWMD0
PAC5210									
PAC5220	DRL0	DRL1	DRL2	DRH3	DRH4	DRH5			
PAC5230	DRL0	DRL1	DRL2	OP3	OP4	OP5			
PAC5240	DRL0	DRL1	DRL2	DRH3	OP4	DRH5			
PAC5250	DRL0	DRL1	DRL2	DRL3	DRL4	DRL5	DXH0	DXH1	DXH2

图 1-11 PWM 与驱动口对应关系

	TIMERA				TIMERB	TIMERC	TIMERD
DTG	DTGA0	DTGA1	DTGA2	DTGA3	DTGB	DTGC	DTGD
PWMHS	PWMA4	PWMA5	PWMA6	PWMA7	PWMB1	PWMC1	PWMD1
PWMLS	PWMA0	PWMA1	PWMA2	PWMA3	PWMB0	PWMC0	PWMD0

图 1-12 PWM 互补死区对应关系

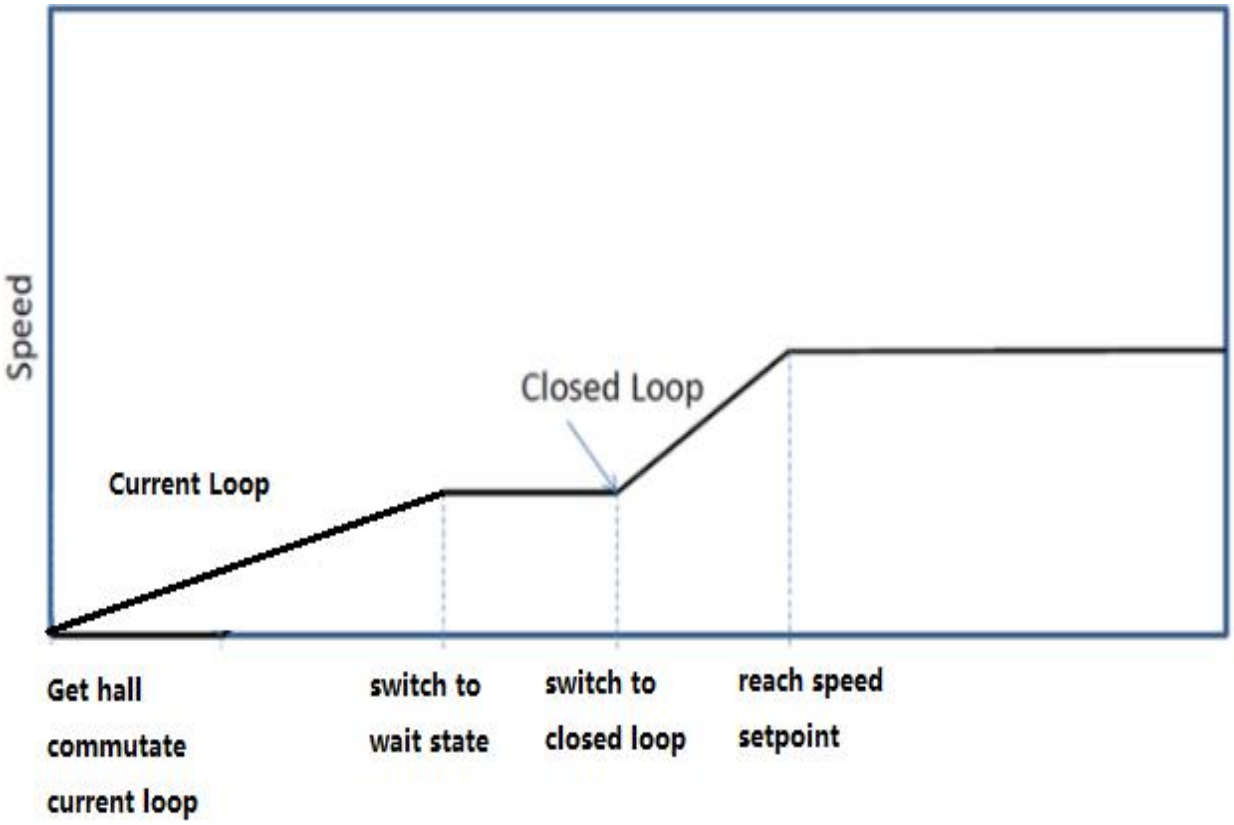
为了方便调试，将数字量转化为模拟量，可以用示波器实时观察，这里配置了 2 路 PWM 输出,经过 RC 低通滤波器得到连续的模拟量,以方便示波器观测，代码如下，

```
void debug_dac_init()

{

	PAC5XXX_GPIOD->OUTEN.P2 = 1;

	PAC5XXX_GPIOD->OUTEN.P3 = 1;



	PAC5XXX_GPIOD->PU.P2 = 1;

	PAC5XXX_GPIOD->PU.P3 = 1;



	pac5xxx_timer_clock_config(TimerB, TxCTL_CS_HCLK, TxCTL_PS_DIV1);

	pac5xxx_timer_base_config(TimerB, TIMERB_TICKS_DEBUG_DAC_45KHZ, 0,
TxCTL_MODE_UP, 0);  // Configure Timer

	pac5xxx_timer_cctrl_config(TimerB, 0, 0, 1);

	pac5xxx_timer_cctrl_config(TimerB, 1, 0, 1);

	pac5xxx_timer_io_select_pwmb0_pd2();

	pac5xxx_timer_io_select_pwmb1_pd3();

}
```

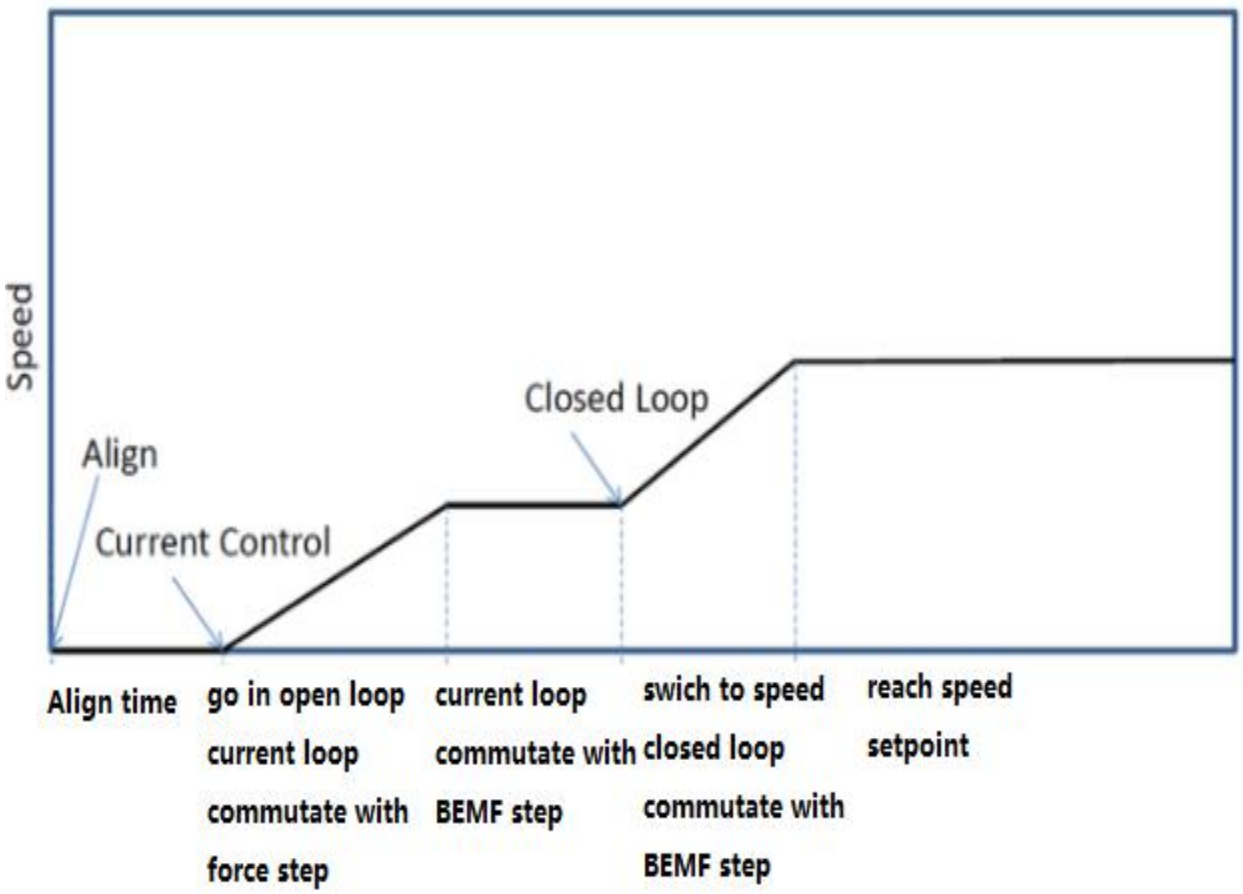
2 方波控制流程说明

2.1 方波控制整体框架及流程概览

Hall 方式：



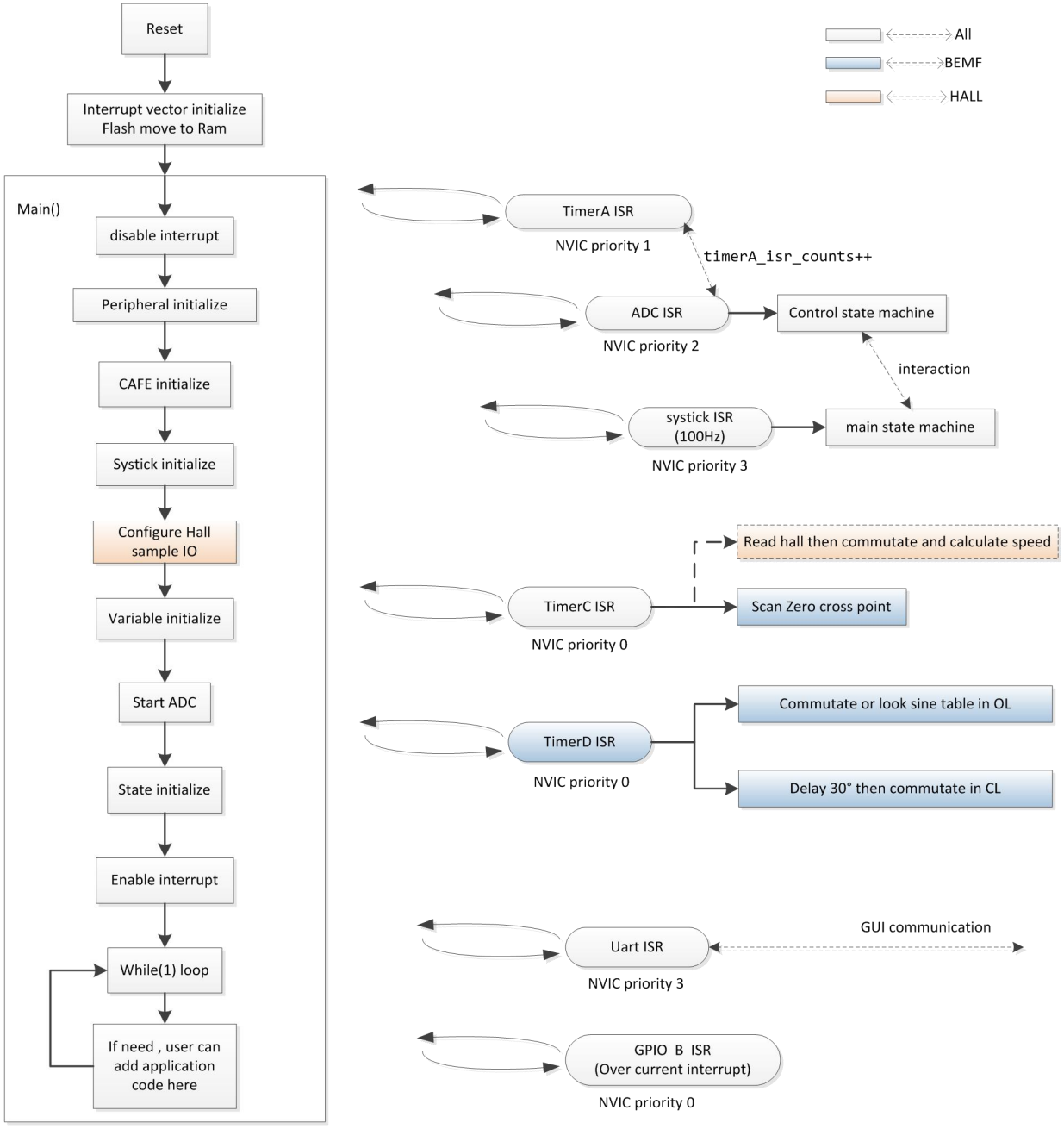
BEMF 方式：



PAC52xx 在整套方波控制流程中加了许多的附加功能，以便用户利用 Demo 工程，包括电机运动中启动（SIM）、过流保护、缺相保护、堵转保护、过欠压保护、限速保护（可设最大值和最小值）等等，从结构框架图可以看到，这套系统是配有人机界面（GUI）方便用户对电机参数进行调试的。

2.2 固件架构

BLDC 固件整体框架如下图所示，在 main( )函数里完成初始化后进入空循环等待中断触发，主状态机在 100Hz 的 Systick ISR 中执行，方波控制流程在 ADC 中断中执行，TimerC 配置为 100KHz 用来扫描过零点（或读取 hall 状态换相），TimerD 用来在找到过零点后延时换相，Uart 中断用来与 GUI 通讯，GPIOB 中断用来处理过流保护。



Main

在 main()函数里主要是做芯片配置和变量初始化：

芯片配置：数字核--> 系统时钟 PPL 为 50MHz；

--> TimerA 配置 updown 计数模式，并作为 PWM 载波定时器，配置互补死区，将 PWMAx 映射到驱动引脚；

--> TimerC 配置 up 计数模式，频率为 100KHz 中断，用来扫描反电动势过零点（或者读 hall 状态换相）；

--> TimerD 配置 up 计数模式，频率为最慢（128 分频，且配置周期寄存器值为 65535），反电动势过零点后开始计时等待中断换相；

--> 配置 GPIOB0 中断，当发生过流事件时触发中断；

--> 配置 UART 中断，用来与 GUI 通讯；

--> 配置 ADC 采样序列；

模拟核--> 配置开关电源，设置 VP 电压；

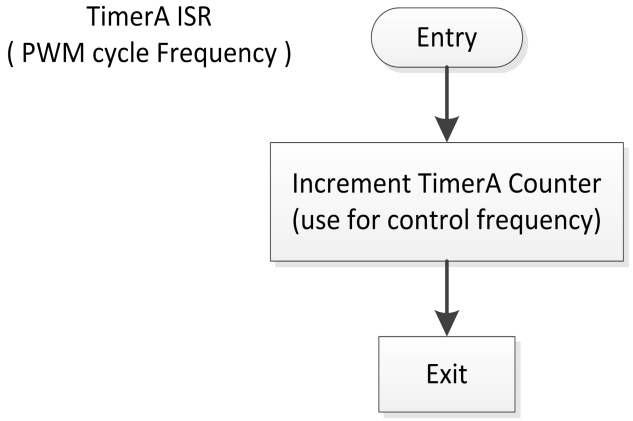
--> 配置差分运算放大器；

--> 配置相比较器；

配置 systick 时钟为 100Hz；

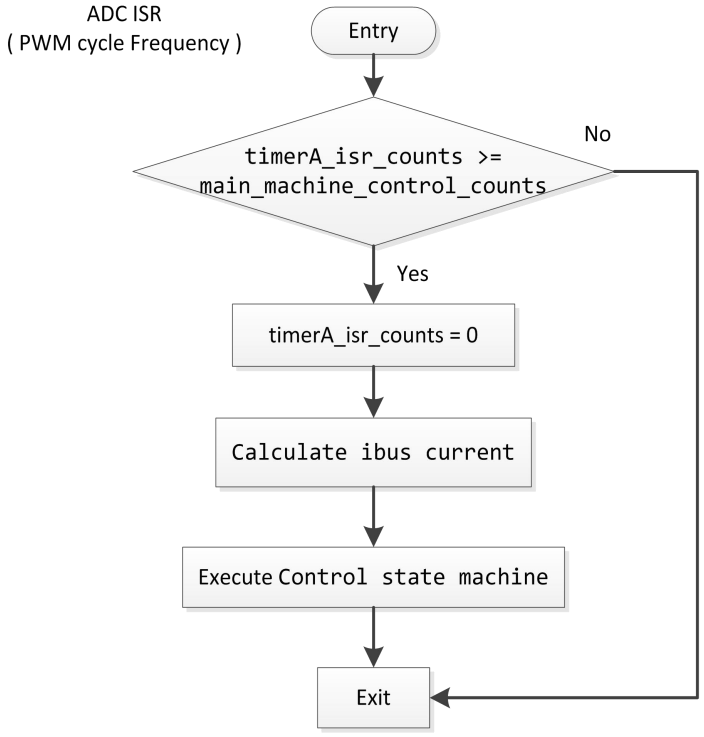
TimerA ISR

TimerA 配置的是 PWMA1 的中断，在 TimerA 计数到接近最高点时产生中断，在中断对 timerA_isr_counts 加 1，timerA_isr_counts 用在 ADC 中断中，用来分频控制频率。



ADC ISR

ADC 中断是在 ADC 序列采样完成后触发的，在中断里会计数采样电流，执行“control_state_machine()”，更新 PWM_DAC_debug 的变量。

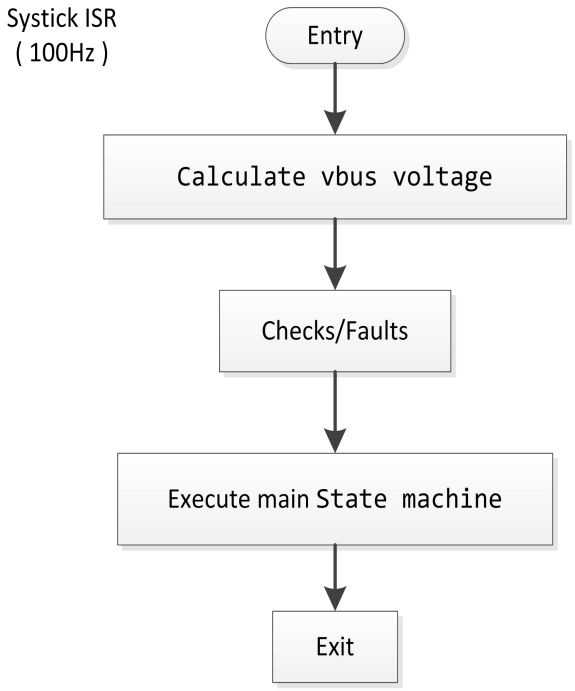


Systick ISR

Systick 中断被配置为 100Hz，用来执行一些对频率要求不高的事情，比如一些保护检测和 main_state_machine。

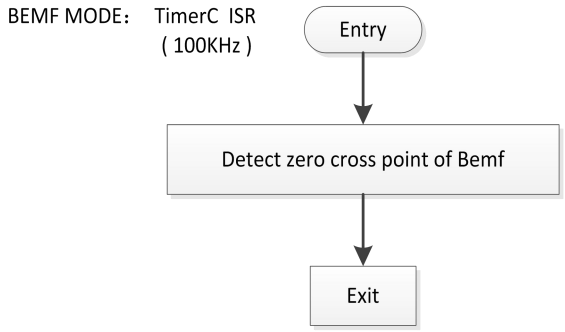
保护检测有：

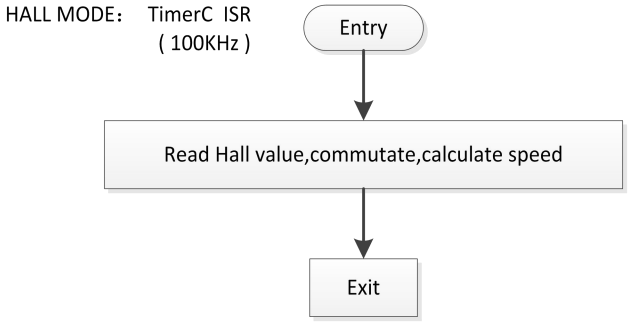
- 母线电压检测
- 过压欠压保护
- 速度指令控制
- 上电检测电压启停



TimerC ISR

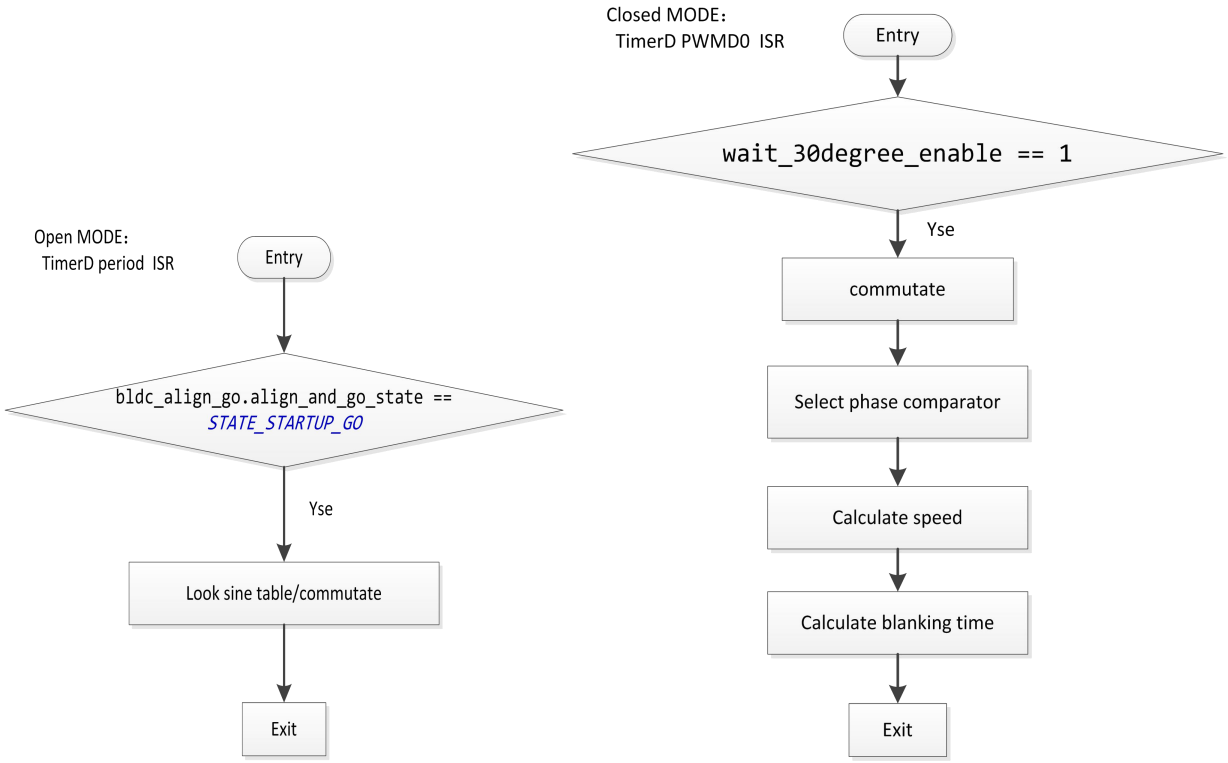
TimerC 中断被配置为 100KHz，用来检测反电动势过零点，如果是 Hall 模式，则用来读取 Hall 值，当 Hall 值发生变化是就换相并计算转速。





TimerD ISR

TimerD 配置了周期中断和 PWMD0 中断，周期中断用作开环加速计时；PWMD0 中断用作闭环过零点后延时换相，在找过零点后读取 TimerD 计时值，然后重新配 TimerD，等待 PWMD0 中断，在中断中换相，并重新选择相比较器，计算转速，计算消磁时间。



UART ISR

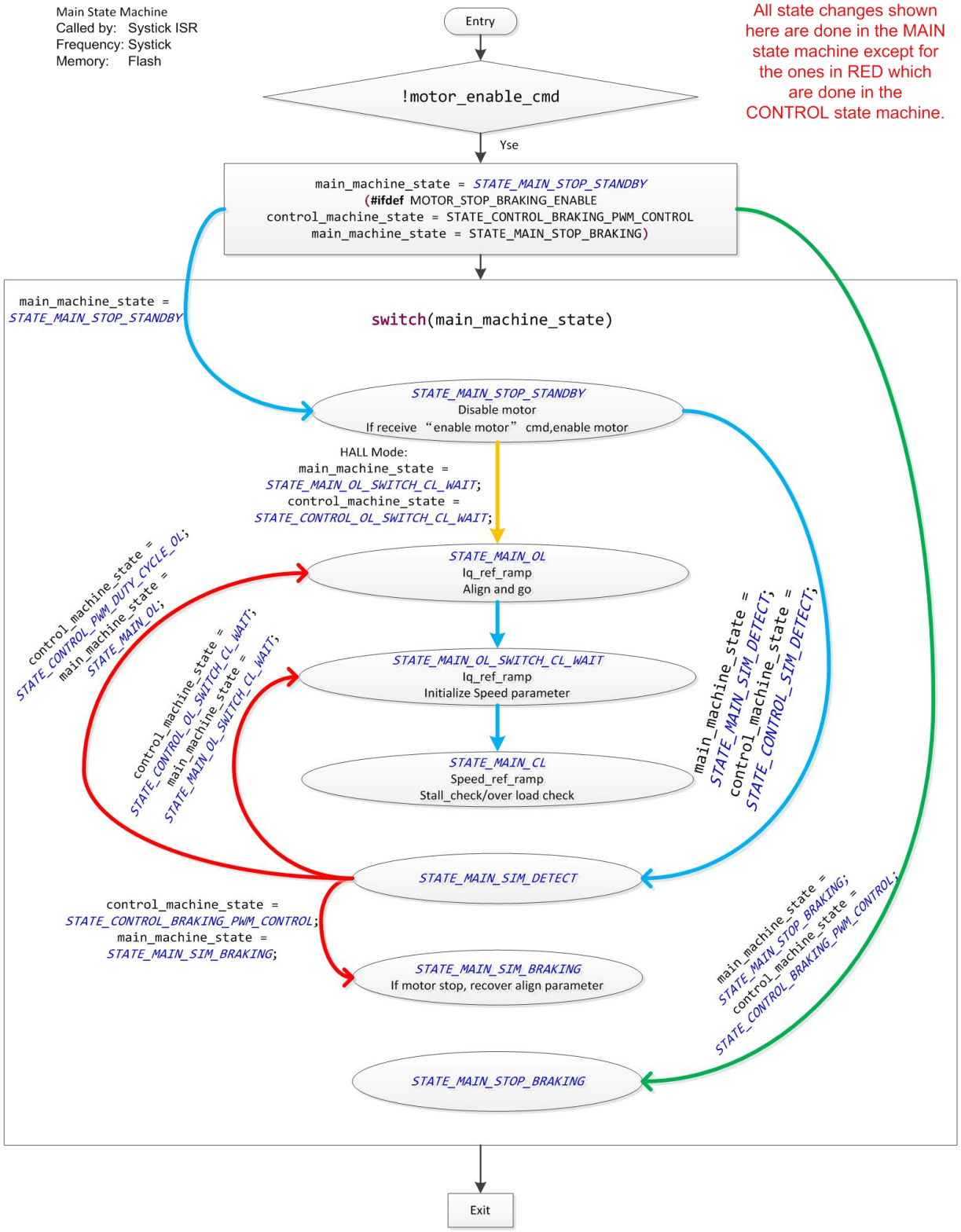
UART 中断用来与 GUI 通讯，接收和发送都是用的 9 个 byte 的协议，接收的包头为 0x89，发送的包头为 0xEE，有 2byte 功能码，4byte 为数据，1byte 为校验和，有 byte 保留。当接收完成，就切换到发送；当发送完成，就切换到接收。UART 的中断优先级配置为 3，如果其他较高优先级的中断占用了所有时间，则 UART 不能正常通讯，芯片将连接不上 GUI。

GPIOB ISR

PAC 系列芯片的模拟前端有硬件过流保护功能，在发生过流保护事件同时可触发 GPIOB0 中断，在此中断了可以置过流标志位，用处理过流事件。当发生过流保护后，可通过 GUI 的“OC_Reset”命令清过流保护。

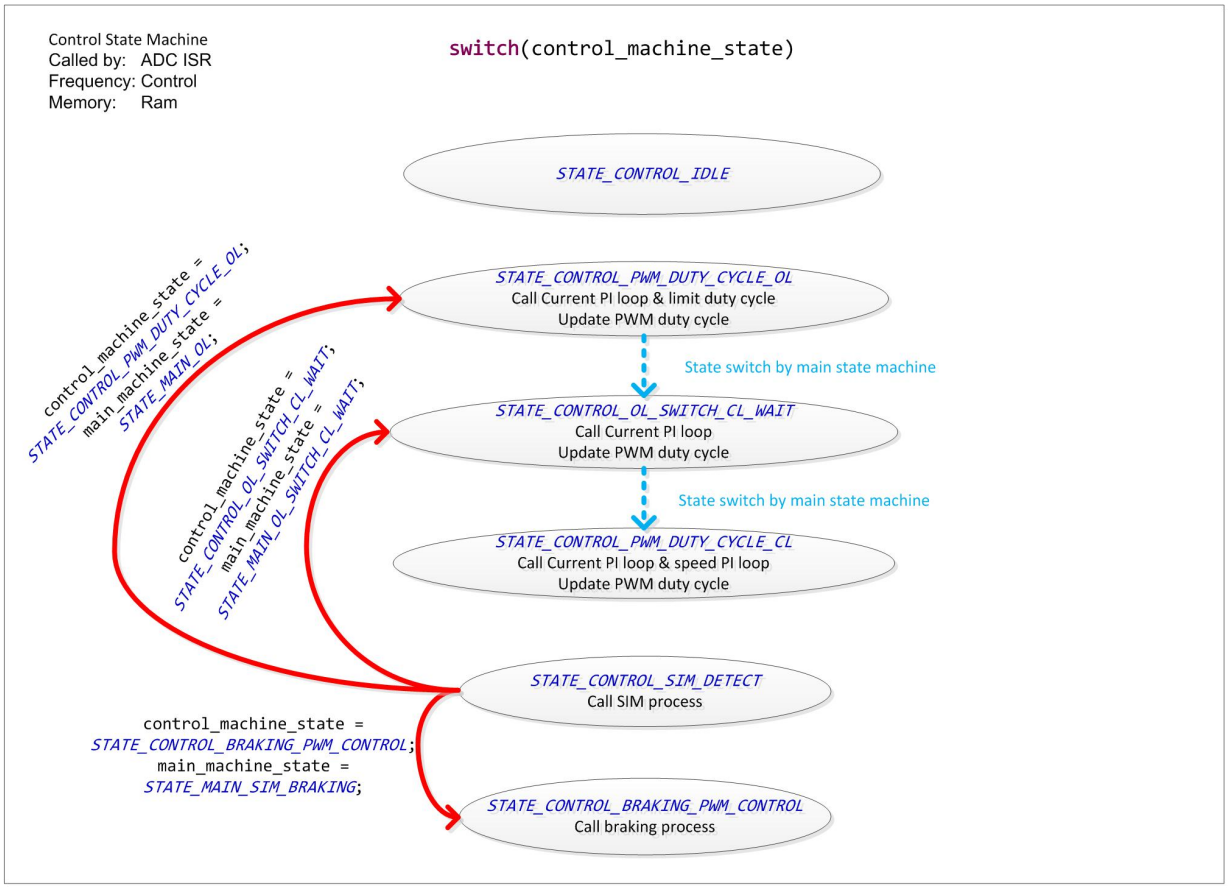
Main State Machine

Main State Machine 是最顶层的主状态机，用来控制电机的运行状态，比如，电机启停，何时进入 SIM，何时进入 OL，何时进入 CL；主状态机主要是做状态的切换，包括对 Control State Machine 状态的切换。



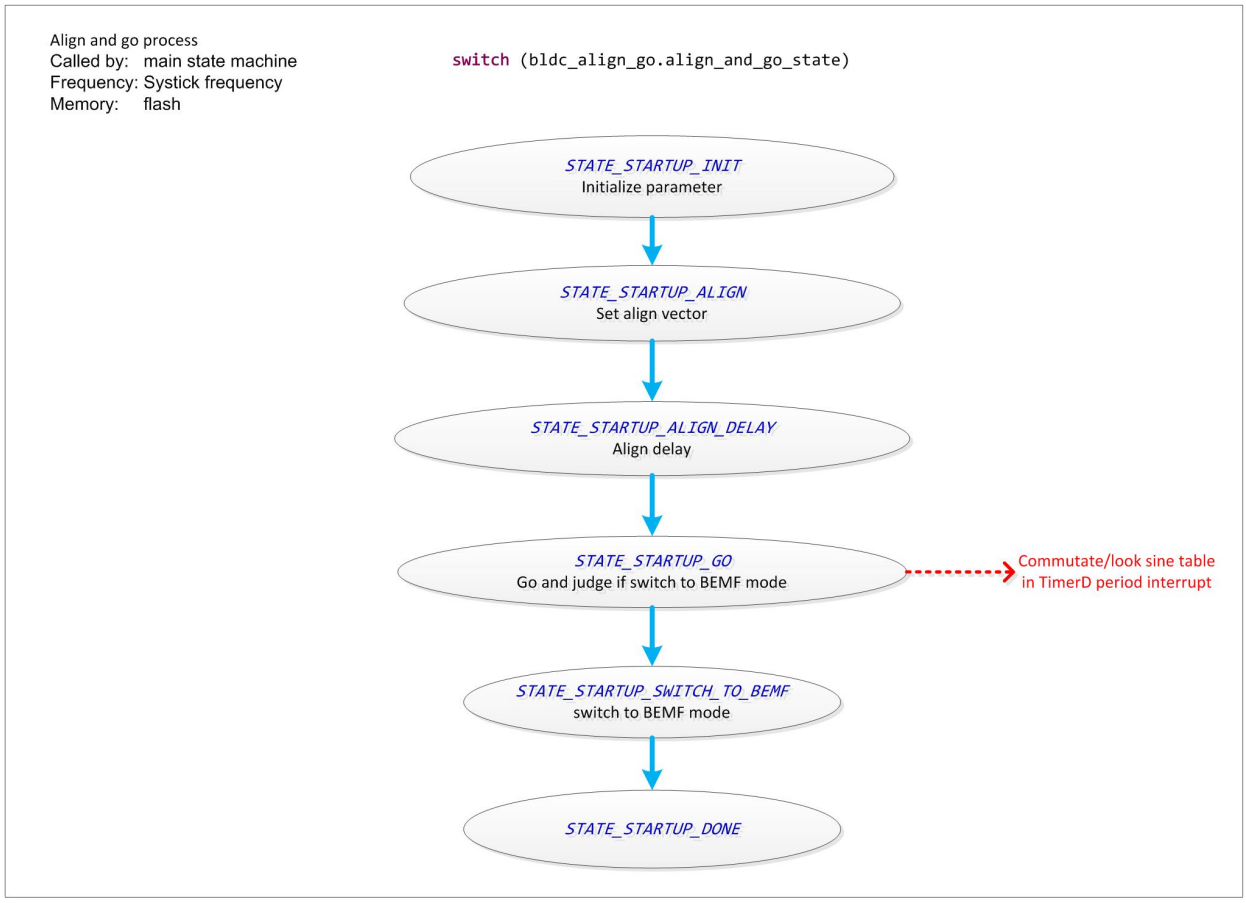
Control State Machine

Control State Machine 是在 ADC 中断里执行，主要运行电流环 PI，速度环 PI，SIM，Braking 控制，需要较高的频率。



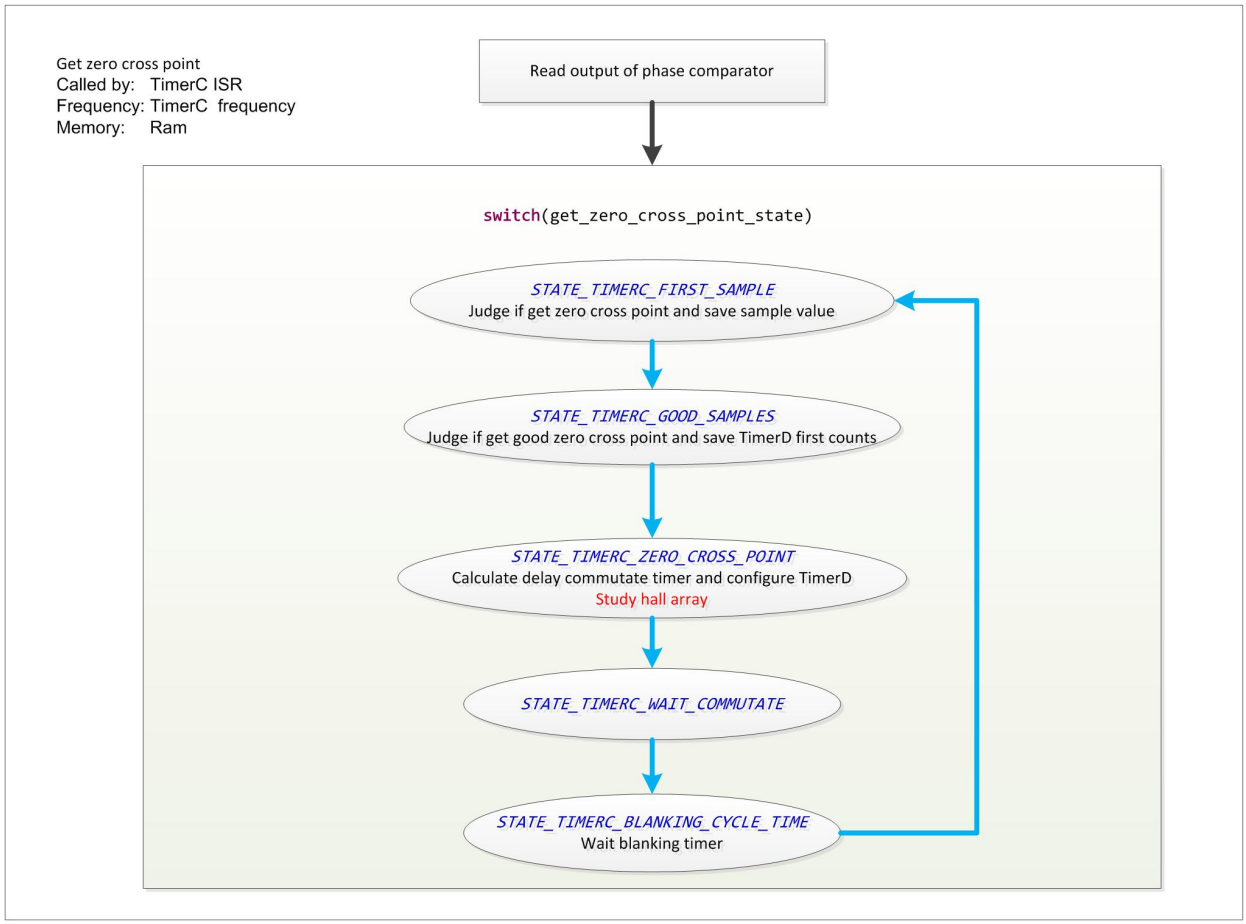
Align and go process

在无传感 BLDC 启动时，由于不知道转子位置，需要用外同步模式强行拖动电机转动，当电机转起来有了一定反电动势后，就可以切入反电动势模式。



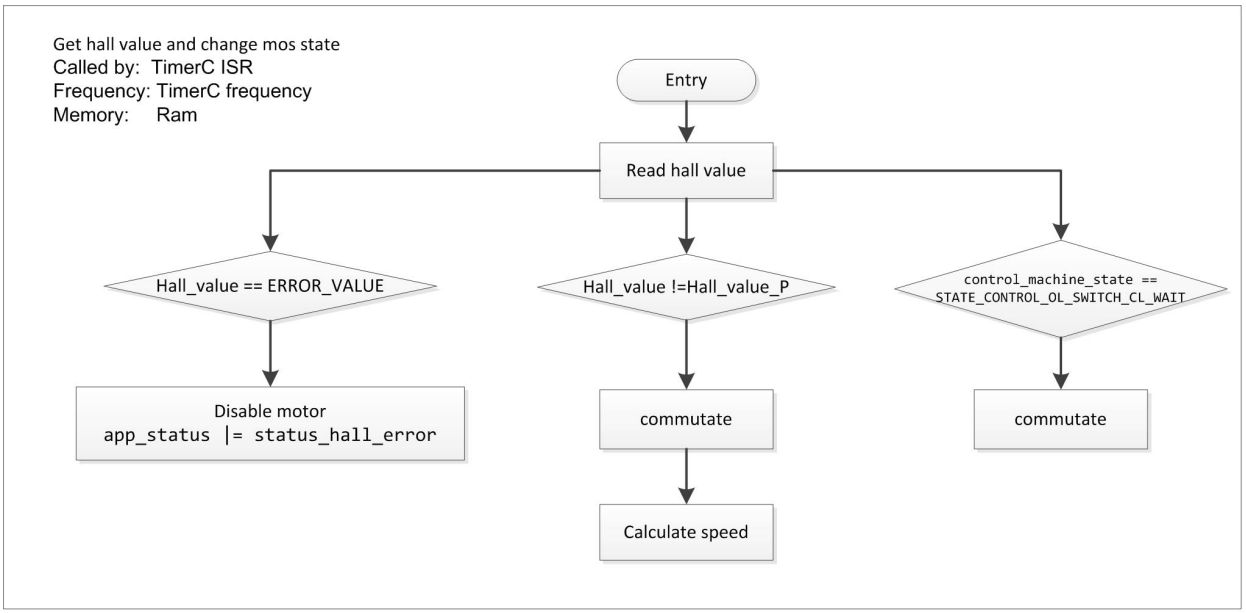
Get zero cross point

每次进 TimerC 中断会读取相比较器的输出电平，若判断有电平跳变，且是符合要求的电平跳变，则认为找到有效反电动势过零点，找到过零点后，保存 TimerD 计数值，计算延时换相的时间，重新配置 TimerD，等待 TimerD 的 PWMD0 中断。在换相完成之后，会有一段消磁时间，在这段时间不去读相比较器的输出电平，等过了这段时间再去读取。



Get hall value and change mos state

如果定义了 Hall 方式，则每次进 TimerC 中断会读取 Hall 值，若判断 Hall 值有跳变，且是符合要求的电平跳变，则换相，读取 TimerD 计数值，计算转速。



3 更改履历

（日期）　　（更改事项）