



基于双电阻电流采样的电机无感 FOC 调速系统应用

基于双电阻电流采样的电机无感 FOC 调速系统应用

文档版本 06

发布日期 2024-01-05

版权所有 © 海思技术有限公司2024。保留一切权利。

非经本公司书面许可，任何单位和个人不得擅自摘抄、复制本文档内容的部分或全部，并不得以任何形式传播。

商标声明



HISILICON、海思和其他海思商标均为海思技术有限公司的商标。

本文档提及的其他所有商标或注册商标，由各自的所有人拥有。

注意

您购买的产品、服务或特性等应受海思公司商业合同和条款的约束，本文档中描述的全部或部分产品、服务或特性可能不在您的购买或使用范围之内。除非合同另有约定，海思公司对本文档内容不做任何明示或默示的声明或保证。

由于产品版本升级或其他原因，本文档内容会不定期进行更新。除非另有约定，本文档仅作为使用指导，本文档中的所有陈述、信息和建议不构成任何明示或暗示的担保。

海思技术有限公司

地址：上海市青浦区虹桥港路2号101室 邮编：201721

网址：<https://www.hisilicon.com/cn/>

客户服务邮箱：support@hisilicon.com



前言

概述






本文以Pmsm Sensorless 2shunt Foc电机控制例程软件为例，介绍基于GBM2804H-100T云台电机的FOC（Field-Oriented Control）控制应用流程。

读者对象

- 软件工程师

符号约定

在本文中可能出现下列标志，它们所代表的含义如下。

符号	说明
 危险	表示如不避免则将会导致死亡或严重伤害的具有高等级风险的危害。
 警告	表示如不避免则可能导致死亡或严重伤害的具有中等级风险的危害。
 注意	表示如不避免则可能导致轻微或中度伤害的具有低等级风险的危害。
 须知	用于传递设备或环境安全警示信息。如不避免则可能会导致设备损坏、数据丢失、设备性能降低或其它不可预知的结果。 “须知”不涉及人身伤害。
 说明	对正文中重点信息的补充说明。 “说明”不是安全警示信息，不涉及人身、设备及环境伤害信息。



修改记录

修订日期	版本	修订说明
2022-12-12	01	第1次正式版本发布。
2023-01-12	02	第2次正式版本发布。 刷新1.3章节的图片：IDE工程生成界面2。 刷新1.4章节的图片：软件架构图。
2023-02-28	03	第3次正式版本发布。 补充缩略语。
2023-06-21	04	第4次正式版本发布。 刷新3.1章节的图片：系统模型框图角度参数多余输入。
2023-10-13	05	第4次正式版本发布。 刷新图1、图2图片内容。 刷新表1、表2参数变量列表。 刷新程序主流程图操作流程。
2024-01-05	06	第4次正式版本发布。 刷新文档支持芯片型号，将文档内容和芯片型号解绑



目 录

前言..... i

1 系统概述..... 1

 1.1 应用概述..... 1

 1.2 开发环境..... 1

 1.2.1 硬件开发环境..... 1

 1.2.2 软件开发环境..... 1

 1.3 系统环境搭建..... 1

 1.4 软件架构..... 3

 1.5 软件规格..... 3

2 无传感器磁场定向控制（FOC）..... 4

 2.1 Clark 变换..... 5

 2.2 Park 变换..... 5

 2.3 PI 控制器..... 6

 2.4 Park 逆变换..... 6

 2.5 空间矢量脉宽调制（SVPWM）..... 6

 2.6 滑模观测器..... 7

 2.6.1 滑模观测器设计..... 7

 2.6.2 角度及速度估计--PLL 锁相环..... 9

 2.7 IF 启动控制..... 10

3 双电阻电流采样调速系统工程介绍..... 11

 3.1 电机调速系统模型框图介绍..... 11

 3.2 电机控制应用功能介绍..... 11

 3.2.1 电机启停/调速控制方法介绍..... 11

 3.2.2 双电阻电流采样方法介绍..... 12

 3.2.3 相电流过流保护功能介绍..... 12

 3.3 软件目录及架构流程介绍..... 12

 3.3.1 软件目录安排..... 13

 3.3.2 主流程介绍..... 17

 3.3.2.1 APT 高级 PWM 定时器..... 20

 3.3.2.2 ACMP 模拟比较器..... 23

 3.3.2.3 PGA 可编程增益放大器..... 24

 3.3.2.4 ADC 模数转换器..... 26



3.3.2.5 TIMER 基本定时器.....27

3.3.2.6 UART 串口收发器..... 28

3.3.3 载波控制流程介绍..... 29

3.3.4 定时器中断执行流程.....30

3.3.5 电位器调速流程介绍.....32

4 参考文献.....34

5 缩略语..... 35



插图目录

图 1-1 IDE 工程生成界面 1..... 2

图 1-2 IDE 工程生成界面 2..... 2

图 1-3 软件架构图..... 3

图 2-1 Clark 变换示意图..... 5

图 2-2 Park 变换示意图..... 5

图 2-3 dq 轴作用矢量示意图..... 6

图 2-4 符号函数示意图..... 8

图 2-5 饱和函数估计框图..... 8

图 2-6 反电动势估计模型框图..... 9

图 2-7 PLL 原理图..... 9

图 3-1 系统模型框图..... 11

图 3-2 电流采样时刻示意图..... 12

图 3-3 程序主流程图..... 19

图 3-4 波形生成示意图..... 21

图 3-5 例程波形生成示意图..... 21

图 3-6 波形输出组合示意图..... 22

图 3-7 ACMP 与其他模块连接关系..... 24

图 3-8 PGA 外围电路示意图..... 25

图 3-9 触发 ADC 采样示意图..... 26

图 3-10 载波中断执行流程图..... 30

图 3-11 定时器中断执行流程图..... 31

图 3-12 电位器调速执行流程图..... 33



表格目录

表 1-1 软件规格表..... 3

表 3-1 工程目录安排..... 13

表 3-2 参数变量..... 14

表 3-3 宏定义..... 16



1 系统概述

1.1 应用概述

本例程适配低压12V电源环境，被控电机型号Gimbal GBM2804H -100T，使用双电阻采样方法实现电机相电流采样，通过滑模观测器实现电机无传感器FOC控制。可通过板载按键和电位器实现电机的启停和调速控制。

1.2 开发环境

开发环境包括程序运行所依托的硬件开发环境、软件开发环境及系统环境。

1.2.1 硬件开发环境

请参见当前MCU型号《Hi306XX通用生态板用户手册》。

1.2.2 软件开发环境

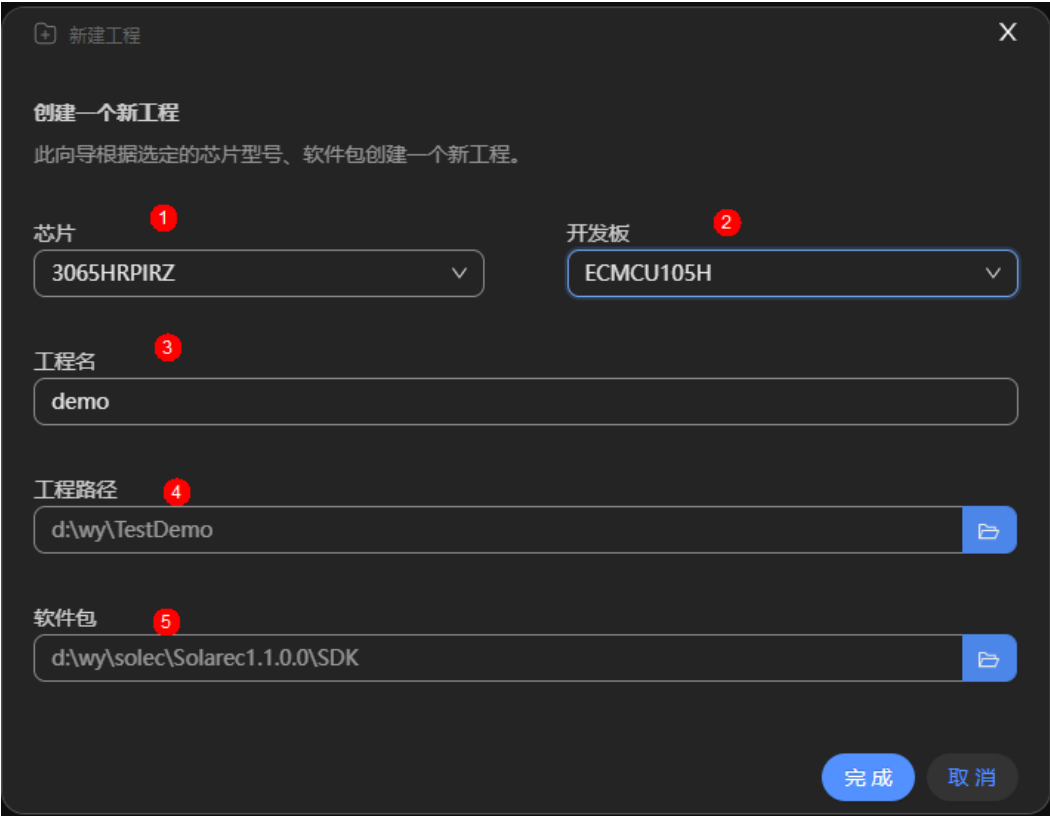
请参见当前MCU型号《Hi306XX通用生态板用户手册》。

1.3 系统环境搭建

步骤1 安装Huawei DevEco Device Tool代码开发与调试工具，调试工具安装过程参照《IDE使用指南》说明文档，安装成功并创建基于当前MCU型号控制板的应用工程（图中以3065HRPIRZMCU芯片型号为例），如图1-1所示。

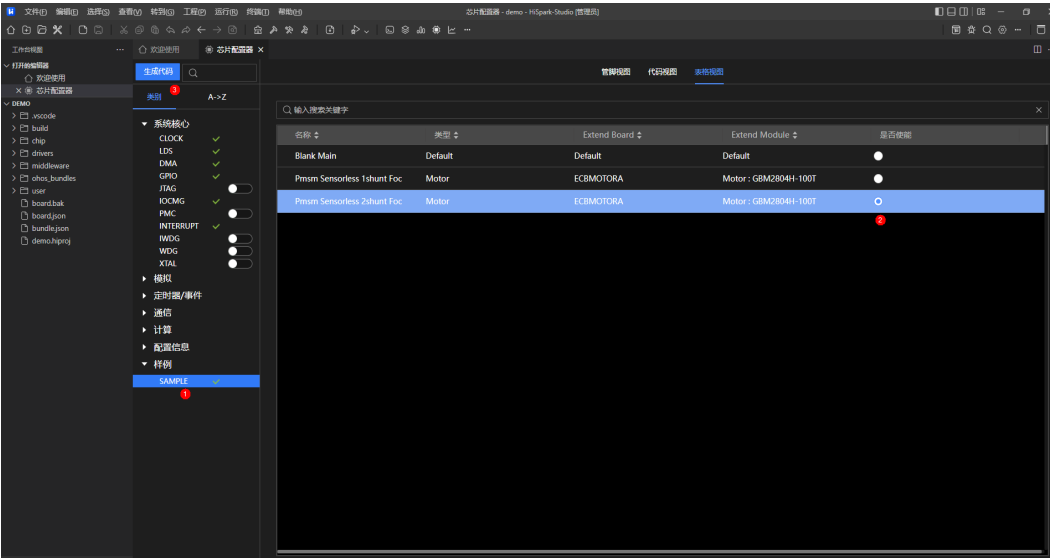


图 1-1 IDE 工程生成界面 1



步骤2 生成单电机双电阻电位器调速应用sample工程，如图1-2所示。

图 1-2 IDE 工程生成界面 2



步骤3 编译调试sample应用工程，本应用采用openocd在线调试方式烧录，烧录调试方式详见《IDE 使用指南》说明文档。

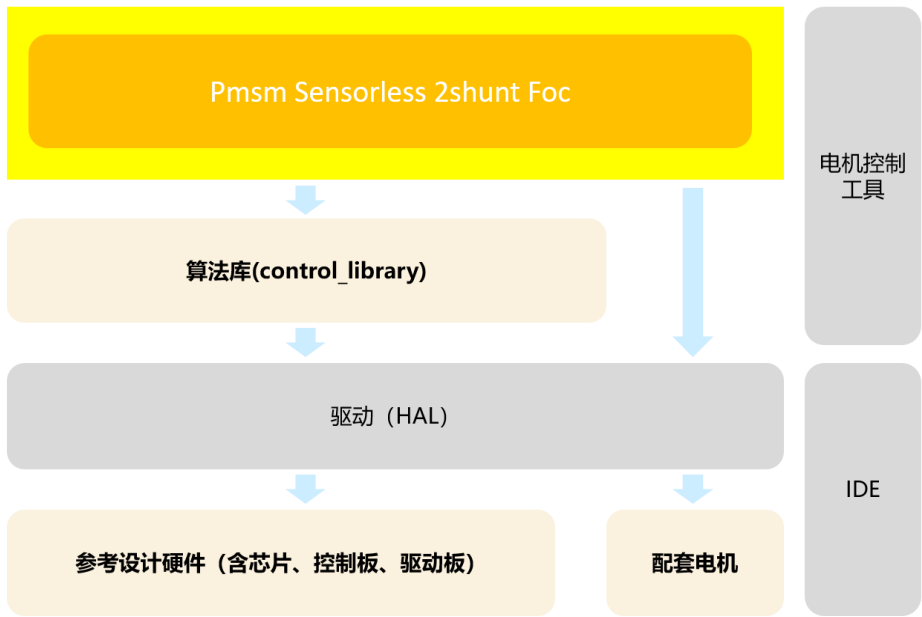
----结束



1.4 软件架构

软件架构可分为三层，以应用例程为主的应用层、支撑应用层功能实现的算法库，以及HAL驱动层。软件架构运行依托于外围硬件，配合IDE调试编译功能，通过层层嵌套完成电机控制。

图 1-3 软件架构图



1.5 软件规格

控制参考例程支持按键启停及电位器旋钮调速，其具体软件规格表如表1-1所示。

表 1-1 软件规格表

序号	规格
1	支持无感FOC电机控制。
2	支持双电阻电流采样方式。
3	单板上电初始化后电机处于停止状态。
4	通过按键进行电机启动、停止状态切换。
5	通过旋钮进行电机调速，调速范围： 300RPM~1545RPM。
6	支持单板复位，复位后电机初始化为停止状态。
7	过流保护。



2 无传感器磁场定向控制 (FOC)

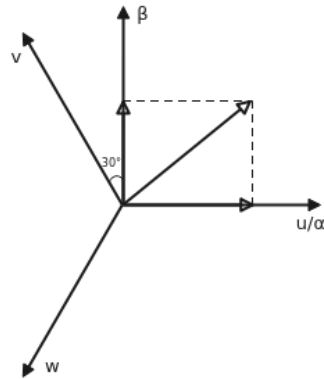
磁场定向控制技术可用于实现BLDC (Brushless Direct Current Motor) 和PMSM (Permanent Magnet Synchronous Motor) 的精准控制, 它具有效率高、运行转矩平稳、噪声小等控制优势。这些优势使其在家电领域如洗衣机、空调、高速风机等场合得到了广泛应用。无传感器磁场定向控制过程可总结为如下步骤:

- 步骤1** 读取电机三相绕组电流, 双电阻采样方式可获取其中两相电流, 第三相电流值可通过基尔霍夫电流定律 $I_u + I_v + I_w = 0$ 计算得到。
- 步骤2** Clark变换, 通过Clark基坐标投影变换可将三相间隔 120° 的基向量坐标系转化为 α - β 静止坐标系, 分别以 I_α 、 I_β 轴表示, 可见经过变换后的被控对象减少了一个维度。
- 步骤3** Park变换, 通过Park变换将上述 α - β 静止坐标系转化为可跟随转子角度实时转动的旋转坐标系, 分别以 I_d 、 I_q 轴表示, 其中d轴方向与转子磁通方向保持平行, q轴方向垂直于转子磁通方向。可见经Park变换将作用于 α - β 静止坐标系下的非线性控制变量转化为了d-q轴坐标系下的线性变量。
- 步骤4** 电流环PI控制, 在d-q轴坐标系下, 将目标电流值与实际采样并经变换后的电流值做差作为电流环的输入, 经过PI计算将输出值赋给 U_d 、 U_q , 目的是将被控对象转化为单板可以调节的变量。
- 步骤5** Park逆变换, 经过逆变换后将控制对象还原到 α - β 静止坐标系下的 U_α 、 U_β 分量, 为下一步的SVPWM (Space Vector Pulse Width Modulation) 解算做准备。
- 步骤6** SVPWM调制, U_α 、 U_β 分量经过SVPWM调制, 最终可计算出对应每个位置时刻所需的电压空间矢量大小及作用时间, 将作用时间转化为控制器比较寄存器的计数值便可输出期望的PWM (Pulse Width Modulation) 波形。
- 步骤7** 速度环PI控制, 速度环作为控制器的外环对电机速度进行调节, 采用滑模观测器进行电机角度和速度的估算。

----结束

2.1 Clark 变换

图 2-1 Clark 变换示意图



如图2-1所示，将uvw坐标系与α/β坐标系原点重合，根据三角函数定理及投影变换，可推知：

$$i_{\alpha} = i_u - i_v \cos 60^{\circ} - i_w \cos 60^{\circ} = i_u - \frac{1}{2} * i_v - \frac{1}{2} * i_w$$

$$i_{\beta} = i_v \sin 60^{\circ} - i_w \sin 60^{\circ} = \sqrt{3}/2 * (i_v - i_w)$$

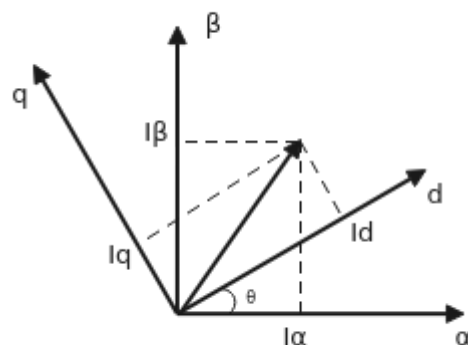
对上式进行化简并对电流值进行等幅值变换：

$$i_{\alpha} = i_u$$

$$i_{\beta} = (i_u + 2 * i_v) / \sqrt{3}$$

2.2 Park 变换

图 2-2 Park 变换示意图



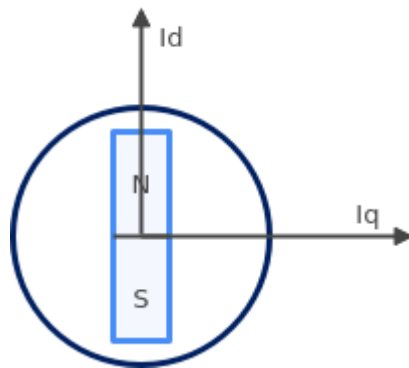
如图2-2所示，将 α - β 静止坐标系做投影变换转化为d-q轴旋转坐标系，可表示为：

$$\begin{aligned} i_d &= i_\alpha \cos\theta + i_\beta \sin\theta \\ i_q &= -i_\alpha \sin\theta + i_\beta \cos\theta \end{aligned}$$

2.3 PI 控制器

例程中PI控制器应用了双环串级控制，电流环作为内环，速度环作为外环，速度环的输出作为电流环的输入。其中 i_d 、 i_q 分别代表沿转子旋转的径向和切向两个方向的变量。从图2-3中可以看出 i_q 方向的力使转子产生旋转加速度，而 i_d 方向的力对转子旋转方向并不产生有效作用，因此在电流环PI控制中应尽量增加 i_q 而使 i_d 控为0。

图 2-3 dq 轴作用矢量示意图



2.4 Park 逆变换

将d-q轴坐标系信息还原到 α - β 轴坐标系，可表示为：

$$v_\alpha = v_d \cos\theta - v_q \sin\theta$$

$$v_\beta = v_d \sin\theta + v_q \cos\theta$$

2.5 空间矢量脉宽调制 (SVPWM)

SVPWM的输入为 U_α 、 U_β ，输出为控制器比较寄存器的计数值，控制器可分如下步骤进行处理：

步骤1 扇区判断。

步骤2 计算各扇区矢量作用时长合成目标矢量所需的作用时长。

步骤3 将作用时长转化为定时器比较寄存器计数值，输出预期PWM并最终作用在控制逆变器中。

----结束

2.6 滑模观测器

滑模观测器在无传感器永磁同步电机矢量控制调速系统软件架构中处于关键环节。滑模观测器是级联控制结构中外环反馈链路中的一环，用于向内环电流控制回路的两次坐标变换提供转子角度估计信息，以及向外环速度控制回路提供转速估计信息。以一阶滑模观测器为例，观测器系统包括如下三个模块：

1. 电流估计

电流估计由两个电流微分估计子模块以及两个离散积分器组成，输出电流估计值。

2. 反电动势估计

反电动势估计由两个滑模函数子模块和一个反电动势滤波器模块组成，输出滤波后的反电动势和滤波产生的角度补偿量。电流估计和反电动势估计组成滑模观测器。

3. 角度及速度估计

角度及速度估计由一个PLL、一个角度补偿计算和一个速度滤波器组成，输出角度估计和速度估计。下面以一阶滑模观测器为例对其应用做详细说明。

2.6.1 滑模观测器设计

1. 电流估计

永磁同步电机扩展反电动势模型下电压方程表示为：

$$\begin{bmatrix} u_\alpha \\ u_\beta \end{bmatrix} = \begin{bmatrix} R_s + sL_d & \omega_e(L_d - L_q) \\ -\omega_e(L_d - L_q) & R_s + sL_d \end{bmatrix} \begin{bmatrix} i_\alpha \\ i_\beta \end{bmatrix} + \begin{bmatrix} e_\alpha \\ e_\beta \end{bmatrix}$$

其中， U_α 和 U_β 为电机控制器向电机施加矢量控制的激励电压， i_α 和 i_β 为电机反馈电流， R_s 为定子电阻， L_d 和 L_q 为d、q轴电感， s 为拉普拉斯算子， e_α 和 e_β 为电机反电动势。

将上式表示为状态空间方程，如下所示：

$$s \begin{bmatrix} i_\alpha \\ i_\beta \end{bmatrix} = \begin{bmatrix} -\frac{R_s}{L_d} & -\frac{\omega_e(L_d - L_q)}{L_d} \\ \frac{\omega_e(L_d - L_q)}{L_d} & -\frac{R_s}{L_d} \end{bmatrix} \begin{bmatrix} i_\alpha \\ i_\beta \end{bmatrix} + \frac{1}{L_d} \begin{bmatrix} u_\alpha \\ u_\beta \end{bmatrix} - \frac{1}{L_d} \begin{bmatrix} e_\alpha \\ e_\beta \end{bmatrix}$$

其中， e_α 和 e_β 表示为：

$$\begin{bmatrix} e_\alpha \\ e_\beta \end{bmatrix} = \left((L_d - L_q) \left(\omega_e i_d - \frac{di_q}{dt} \right) + \omega_e \psi_f \right) \begin{bmatrix} -\sin \theta_e \\ \cos \theta_e \end{bmatrix}$$

其中， ψ_f 为永磁体磁链幅值。

根据上式，观测器状态方程应表示为：

$$s \begin{bmatrix} \hat{i}_\alpha \\ \hat{i}_\beta \end{bmatrix} = \begin{bmatrix} -\frac{R_s}{L_d} & -\frac{\hat{\omega}_e(L_d - L_q)}{L_d} \\ \frac{\hat{\omega}_e(L_d - L_q)}{L_d} & -\frac{R_s}{L_d} \end{bmatrix} \begin{bmatrix} \hat{i}_\alpha \\ \hat{i}_\beta \end{bmatrix} + \frac{1}{L_d} \begin{bmatrix} u_\alpha \\ u_\beta \end{bmatrix} - \frac{1}{L_d} \begin{bmatrix} \hat{e}_\alpha \\ \hat{e}_\beta \end{bmatrix}$$

其中，上标[^]表示观测器估计值。

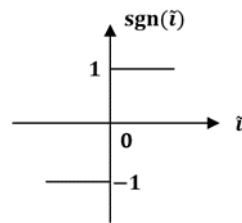
2. 反电动势估计

上式中反电动势估计值通过滑模函数进行估计，滑模函数的趋近律取符号函数，其表达式为：

$$\begin{bmatrix} \hat{e}_\alpha \\ \hat{e}_\beta \end{bmatrix} = \begin{bmatrix} K_{SMO} \cdot \text{sgn}(\hat{i}_\alpha - i_\alpha) \\ K_{SMO} \cdot \text{sgn}(\hat{i}_\beta - i_\beta) \end{bmatrix}$$

函数图像如[图2-4](#)所示。

图 2-4 符号函数示意图

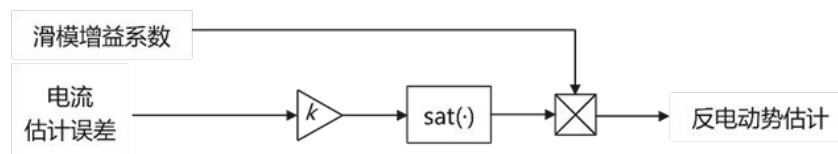


趋近律采用符号函数的滑模观测器，当估计误差在零附近时会出现较为剧烈的抖振现象，从而造成观测结果中出现抖振，影响观测效果，将直接影响电机FOC控制性能。通过将符号函数零点附近进行平滑化处理，可以一定程度上减缓抖振现象，使输出结果更为平滑。饱和函数是一种实现容易，线性化的平滑处理函数，函数方程如下所示：

$$\text{sat}(x) = \begin{cases} 1, & \text{当 } x > 1 \\ x, & \text{当 } -1 \leq x \leq 1 \\ -1, & \text{当 } x < -1 \end{cases}$$

在上式中趋近律使用饱和函数代替符号函数来设计滑模函数，可以有效削减符号函数在零点附近的抖振，使得观测结果更加平滑，使用饱和函数的估计过程如[图2-5](#)所示。

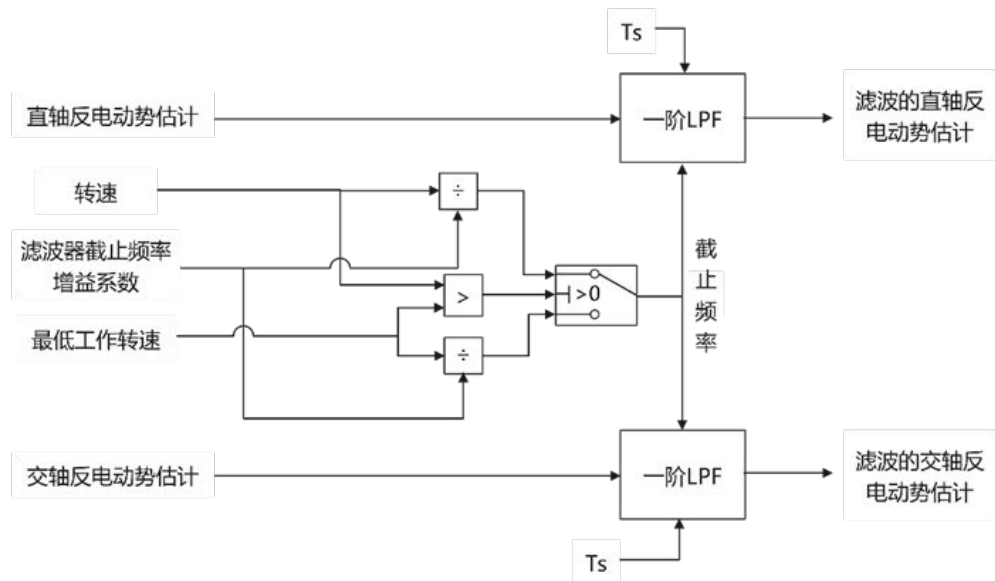
图 2-5 饱和函数估计框图



反电动势估计采用了有饱和函数的滑模估计进行了平滑处理，但是通常仍会有部分抖振传递到后级环节，还需要将反电动势进行一次低通滤波，如[图2-6](#)所示。



图 2-6 反电动势估计模型框图



为保证能容易的对滤波造成的相移进行补偿，需要设计滤波器截止频率。设电机基频为 ω_e ，控制周期为 T_s ，滤波器截止频率为 ω_c ，滤波器设计为：

$$y = \frac{\omega_c}{s + \omega_c} u$$

采用后向差分进行离散，离散结果为：

$$y[k] = \frac{1}{1 + \omega_c T_s} (\omega_c T_s u[k] + y[k-1])$$

设计 $\omega_c = \omega_e * \lambda$ ，则滤波器在基频处相移为：

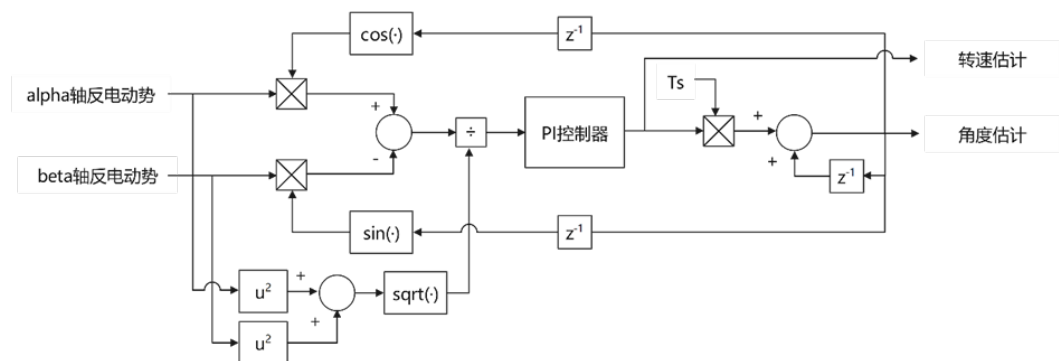
$$\phi = \arctan(\lambda)$$

故只需要在最终辨识结果处补偿 ϕ 角度即可消除低通滤波器造成的误差。

2.6.2 角度及速度估计--PLL 锁相环

PLL原理图如图2-7所示。

图 2-7 PLL 原理图





PLL闭环传递函数可以表示为：

$$G_{PLL} = \frac{\hat{\theta}_e}{\theta_e} = \frac{kK_{p_pll}s + kK_{i_pll}}{s^2 + kK_{p_pll}s + kK_{i_pll}}$$

其中， k_{p_pll} 、 k_{i_pll} 为PLL锁相环PI参数。 k 为误差增益系数，在不同转速下其取值会发生变化，导致PLL的PI参数也需要相应发生变化。通过将误差标么化（将误差除以反电动势幅值），可以使得PI参数不用跟随转速变化取不同的值，使PLL性能保持稳定。

标么化的PLL闭环传递函数表示为：

$$G_{PLL} = \frac{\hat{\theta}_e}{\theta_e} = \frac{K_{p_pll}s + K_{i_pll}}{s^2 + K_{p_pll}s + K_{i_pll}}$$

根据上式，PLL状态方程可描述为：

$$\dot{\hat{\omega}}_e = K_{i_pll}\Delta\theta_e$$

$$\dot{\hat{\theta}}_e = \hat{\omega}_e + K_{p_pll}\Delta\theta_e$$

设计PLL传递函数为如下二阶系统，其中 ρ 为系统带宽：

$$\frac{\hat{\theta}_e}{\theta_e} = \frac{2\rho s + \rho^2}{s^2 + 2\rho s + \rho^2}$$

即：

$$K_{p_pll} = 2\rho$$

$$K_{i_pll} = \rho^2$$

2.7 IF 启动控制

滑模观测器依靠反电动势信息估算角度和速度，在低速时电机反电动势很小，估算结果存在较大误差，往往需要先将电机拖动起来，常用的有V-F和I-F启动法。I-F启动通过控制电流内环的电流幅度和给定转子电角度实现对电机的电流矢量控制，从而启动电机。

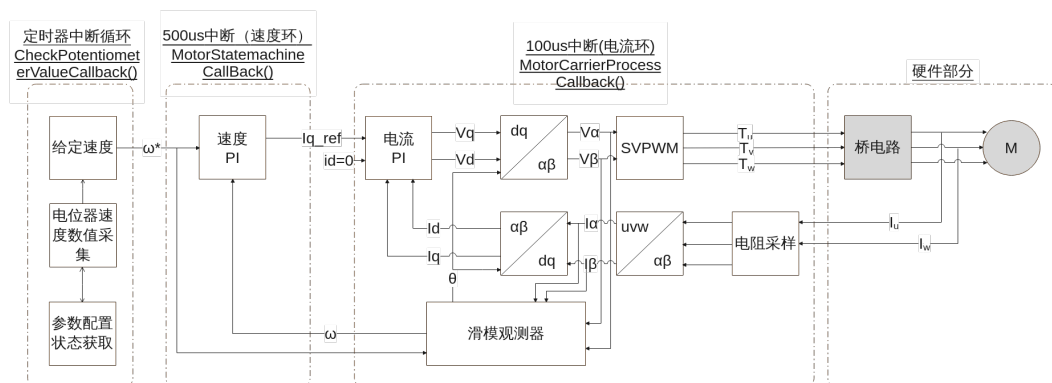
I-F方法在电机启动加速阶段，给q轴逐步注入电流至目标值，随后电机加速启动，速度开环加速期间角度信息由给定的速度积分获得，电枢绕组中产生具有固定幅度和恒定频率的旋转电流矢量，以加速转子转动。当速度达到设定的阈值后，由速度开环切换至速度闭环运行，该阈值根据电机特性进行设定。

3 双电阻电流采样调速系统工程介绍

3.1 电机调速系统模型框图介绍

系统模型框图如图3-1所示，主要由以FOC流程为核心的电流环（相关函数：MCS_CarrierProcess()）、以滑模观测器为核心的速度环（相关函数：TSK_Systiclslr()）、以及负责外部速度指令输入的中断环路（相关函数：CheckPotentiometerValueCallback()）组成。

图 3-1 系统模型框图



3.2 电机控制应用功能介绍

3.2.1 电机启停/调速控制方法介绍

电机启停信号由按键中断产生，每检测到一次按钮点击事件，便产生一次电机启停状态切换。电机调速信号由电位器产生，通过旋钮改变电位器阻值的大小以改变对应的电机目标速度，电位器数值读取及电机目标值设定在定时器（TIMER0）中断中循环更新。

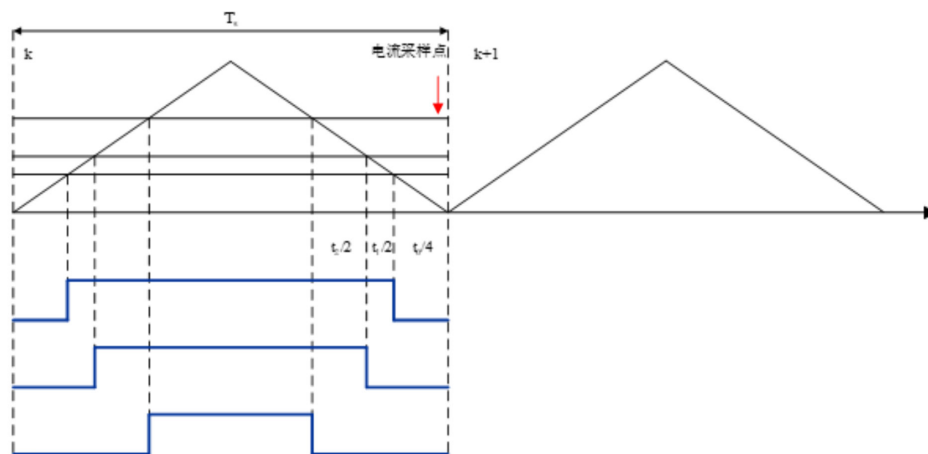
3.2.2 双电阻电流采样方法介绍

电机三相电流参数是永磁同步电机进行坐标变换的关键，采样获取电流值后在通过 **SVPWM** 实现电机转子磁场和定子磁场的同步转动。电流采样通常有三种方案：单电阻采样、双电阻采样和三电阻采样。采样方式关系到整体系统的成本，算法的复杂程度和电机最终运行的效果，因此需要根据项目的具体需求进行选择。

相对于单电阻采样方案，双电阻采样成本相对较高，但在算法实现上要简便许多，例程中双电阻采样算法将电流采样点设置在波谷。可以根据SVPWM当前所在象限，进行分类，只需要采集其中不受窗口时间限制的两相电流，然后根据基尔霍夫电流定律进行电流重构，电流值相对更加准确，后面相关算法的实现也是最好实现的。

例程中，在初始化流程中首先通过函数ReadInitCurrUvwCp()对ADC电流采样偏置值进行校准，通过采集20次偏置数值并取平均的方法确定校准值，如果初始化校准值超出设定阈值范围，则会触发D13位号指示灯常亮并打印复位提示符，此时请复位单板以重新启动程序。

图 3-2 电流采样时刻示意图



3.2.3 相电流过流保护功能介绍

过流保护信号由硬件比较器触发，当过流信号产生后触发APT0事件中中断回调函数MotorSysErrCallback()，在回调函数中立即执行关闭APT的PWM输出操作，以保护电路。并设置电机状态为异常，过流保护灯及D13位号指示灯同时点亮。

3.3 软件目录及架构流程介绍

软件目录主要分为三部分，/middleware/control_library文件夹中主要存放电机所依赖的通用算法库；/user/generatecode/文件夹中主要存放应用外设配置；/pmsm_sensorless_2shunt_foc/inc文件夹主要存放软件定义的电机参数变量及宏定义等电机控制变量；/pmsm_sensorless_2shunt_foc/src文件夹存放了电机载波中断流程。软件架构流程主要由main入口函数等函数的初始化流程、100us载波中断的电流环执行流程、500us定时器中断的速度环执行流程和过流保护处理流程等构成。其软件工程目录安排及模块执行流程如下详示。



3.3.1 软件目录安排

表 3-1 工程目录安排

文件名	子文件夹	子文件	描述
user/ pmsm_sensorless_2 shunt_foc	inc	mcs_carrier.h	存放电机控制数据结构体。
		mcs_user_config.h	存放控制算法应用层接口参数宏定义
		mcs_motor_process.h	枚举变量。
		mcs_chip_config.h	存放不同芯片型号下的底层模块配置接口。
		mcs_status.h	电机运行状态控制函数。
		mcs_ctlmode_config.h	枚举电机控制模式。
	src	mcs_carrier.c	电机角度同步、FOC执行流程、电流环执行函数等。
		mcs_motor_process.c	电机功能函数的封装、软件参数初始化、模块初始化、速度环执行函数。
	protection	mcs_curr_prot.c/.h ...	电机过压、欠压、过流、过温、堵转检测。
	user_interface	cust_process.c/.h ...	电机工具与下位机通信接口。
user/generatecode	-	system_init.c	板载模块初始化配置。
		feature.h	相关MCU配置文件。
		main.h	驱动初始化函数定义。
middleware	control_library (算法库)	mcs_curr_ctrl.c/.h	PI电流环函数封装。
		mcs_filter.c/.h	低通滤波器函数封装。
		mcs_fosmo.c/.h	一阶滑模函数封装。
		mcs_if_ctrl.c/.h	IF启动相关函数封装。
		mcs_pll.c/.h	锁相环函数封装。
		mcs_svpwm.c/.h	svpwm函数封装。



文件名	子文件夹	子文件	描述
		mcs_smo_4th.c/ .h	四阶滑模函数封装。
	

表 3-2 参数变量

序号	文件目录及参数	成员变量	描述说明
1	文件目录：user/ pmsm_sensorless_ 2shunt_foc/inc/ mcs_carrier.h 句柄结构体： MTRCTRL_Handle -- 电机控制句柄	unsigned char motorStateFlag	电机启动/停止状态标志。
		float spdCmdHz	用户配置电机目标电转速。
		float axisAngle	同步坐标系下的转子角度。
		float spdRefHz	速度指令值。
		float currCtrlPeriod	电流环控制周期。
		float adc0Compensate	adc0通道电流采样偏置校准值。
		float adc1Compensate	adc1通道电流采样偏置校准值。
		float udc	单板输入电压
		float powerBoardTemp	功率板温度
		unsigned short aptMaxcntCmp	APT配置的最大计数值。
		float adcCurrCofe	电流采样系数
		unsigned short sysTickCnt	系统定时器计数值。
		unsigned short capChargeTickNum	自举电容充电时间。
		volatile unsigned int msTickCnt	ms定时计数值。
		unsigned short msTickNum	1ms对应的定时器计数次数。
		char obserType	观测器类型
		char controlMode	控制模式
		char spdAdjustMode	速度调节模式



序号	文件目录及参数	成员变量	描述说明
		char uartConnectFlag	串口连接成功标志
		char uartHeartDetCnt	串口连接状态心跳检测
		float uartTimeStamp	串口通信时间戳
		SysStatusReg statusReg	系统当前状态：启动命令、停止命令…
		volatile FsmState stateMachine	电机控制状态：电机运行、电机故障…
		SampleMode sampleMode	电流采样模式设置：单电阻、双电阻。
		MtrParamHandle mtrParam	电机电气参数。
		FoSmoHandle smo	一阶滑模观测器句柄。
		Smo4thHandle smo4th	四阶滑模观测器句柄。
		IfHandle ifCtrl	电机启动阶段的I/F控制句柄。
		SvpwmHandle sv	双电阻采样SVPWM句柄。
		R1SvpwmHandle r1Sv	单电阻采样SVPWM句柄。
		RmgHandle spdRmg	速度斜坡管理句柄。
		SpdCtrlHandle spdCtrl	速度环控制器句柄。
		CurrCtrlHandle currCtrl	电流环控制器句柄。
		StartupHandle startup	启动阶段开闭环切换句柄。
		FwCtrlHandle fw	弱磁控制结构体句柄。
		DqAxis idqRef	dq轴电流指令值。
		UvwAxis currUvw	三相电流采样值。
		AlbeAxis iabFbk	$\alpha\beta$ 轴电流值。
		DqAxis idqFbk	dq轴电流值。
		AlbeAxis vabRef	电流环输出的 $\alpha\beta$ 轴电压值。
		DqAxis vdqRef	电流环输出的dq轴电压值。
		UvwAxis dutyUvw	UVW三相占空比值。



序号	文件目录及参数	成员变量	描述说明
		UvwAxis dutyUvwLeft	APT左侧计数值。
		UvwAxis dutyUvwRight	APT右侧计数值。
		MCS_ReadCurrUvwCb readCurrUvwCb	读取三相电流函数回调接口。
		MCS_SetPwmDutyCb setPwmDutyCb	设置PWM占空比函数回调接口。
		MCS_SetADCTriggerTime Cb setADCTriggerTimeCb	设置采样时间点的回调接口。
		MotorProtStatus_Handle prot	电机错误状态检测标志位句柄。
2	文件目录：user/ pmsm_sensorless_ 2shunt_foc/inc/ mcs_user_config.h 宏定义： MOTORPARAM_D EFAULTS -- 电机电气参数	unsigned int mtrNp	极对数。
		float mtrRs	定子电阻。
		float mtrLd	D轴电感。
		float mtrLq	Q轴电感。
		float mtrPsif	磁链（本例程暂时不用该参数）。
		float mtrJ	转动惯量（本例程暂时不用该参数）。
		float maxElecSpd	最大电转速。
		float maxCurr	最大电流。

表 3-3 宏定义

序号	宏定义	数值	描述说明
1	SMO4TH	无	定义4阶滑模开关，预留参数。
2	SYSTICK_PERIOD_US	500	系统定时器计数值标志。
3	INV_CAP_CHARGE_MS	3	电容充电时间3ms。
4	INV_VOLTAGE_BUS	12	母线电压12V。
5	CTRL_CURR_PERIOD	0.000 1	电流环周期100us。
6	CTRL_SYSTICK_PERIOD	0.000 5	速度环周期500us。
7	SAMPLE_WINDOW_DUTY	0.06	电流ADC采样窗时间，提供单电阻电流采样方案接口，该定义本例程不涉及。



序号	宏定义	数值	描述说明
8	SAMPLE_POINT_SHIFT	0.008	电流采样相移，提供单电阻电流采样方案接口，该定义本例程不涉及。
9	SPEED_FILTER_CUTOFF_FREQUENCY	40	一阶速度低通滤波器截止频率40Hz，计算方法详见2.6 滑模观测器章节。
10	FOSMO_PLL_BDW	30.0	一阶滑模观测器PLL带宽。
11	ADC_CURR_COFFI	0.001 3295	电流采样转化因子 = 3.3V/4096/3.03/0.2ohm，其中3.03为PGA放大倍数，0.2ohm为采样电阻值。
12	FOSMO_GAIN	4.0	一阶滑模增益。
13	CTRL_IF_CURR_AMP_A	0.07	IF启动PI控制目标电流0.07A。
14	USER_TARGET_SPD_HZ	100	自定义目标速度值100Hz。
15	USER_SWITCH_SPDBEGIN_HZ	30	电机从速度开环切至闭环开始点30Hz。
16	USER_SWITCH_SPDEND_HZ	33	电机从速度开环切至闭环结束点33Hz。
17	USER_MAX_SPD_HZ	180.2 5	设定电机最大速度值180.25Hz = 1545RPM/60*7。
	USER_MIN_SPD_HZ	35.0	设定电机最小速度值。
18	USER_SPD_SLOPE	50	开环启动速度加速斜坡。
19	USER_CURR_SLOPE	0.07 *10	开环启动电流注入斜坡。
20	SMO4TH_PLL_BDW	30.0	四阶滑模观测器PLL带宽。
21	SMO4TH_KD	300.0	四阶滑模观测器KD参数。
22	SMO4TH_KQ	2000. 0	四阶滑模观测器KQ参数。
23	SMO4TH_SPD_FILTER_CUTOFF_FREQ	40.0	四阶滑模观测器速度滤波器截止频率。
24	FOSMO_LAMBDA	2.0	一阶低通滤波器截止频率系数。
25	FOSMO_EMF_CUTOFF_FREQ		一阶滑模观测器反电动势滤波器截止频率。
26	SPEED_FILTER_CUTOFF_FREQUENCY	40.0	一阶滑模观测器速度势滤波器截止频率。

3.3.2 主流程介绍

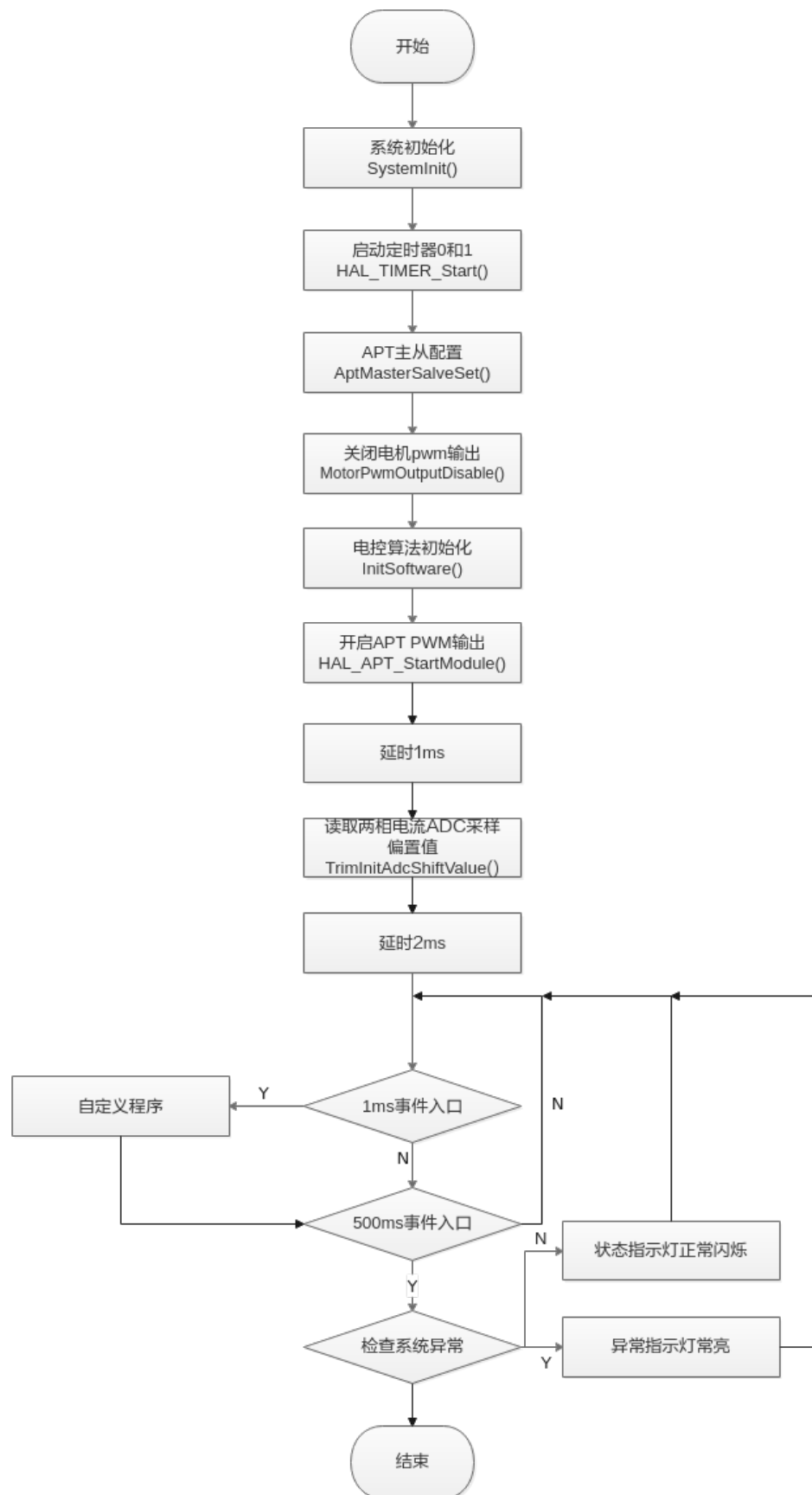
入口函数处理流程包括相关硬件模块和软件参数的初始化、定时器的启动以及一些放在主循环中的处理动作，如电机控制动作执行、过流保护处理和电机正常或异常运行



状态时的LED灯指示等，其中电机调速在定时器TIMER0中断中完成，电机启停信号在按键中断中发生。其执行流程如[图3-3](#)所示。



图 3-3 程序主流程图





在电机系统初始化流程中涉及到了软件各外设模块的初始化配置，包括APT模块、ACMP模拟比较器模块、PGA可编程增益放大器模块、ADC模数转化模块、TIMER基本定时器模块、UART串口通信模块，下面对各模块配置做详细说明（以3065HRPIRZMCU芯片型号为例）。

3.3.2.1 APT 高级 PWM 定时器

APT用于产生双路互补的带死区的PWM，同时支持触发ADC采样、保护事件和同步等功能。本例程中载波中断由APT0模块产生，下面以APT0_Init()函数为例详细讲解APT的初始化及中断初始化配置过程。

- **APT产生PWM配置**

APT产生PWM的原理为在计数器计数到指定的时间点输出指定的电平动作，这个指定时间点在本文中称之为计数参考值点。在一个周期内做两次相反的电平动作即可产生一个周期内的PWM波。

波形控制的动作有：

- 禁止动作：表示忽略该波形控制点，波形不做任何动作。
- 低电平：波形输出低电平。
- 高电平：波形输出高电平。
- 翻转电平：波形进行翻转。

波形生成子模块在以下的时机触发波形控制动作：

- 软件强制，由 PG_ACT_FRC寄存器配置。
- 计数器等于或者周期值。
- 计数器计数到达计数参考值（每个APT有共4个）。
- 外部输入事件有效时（请参见《Hi306xH系列技术参考指南》中“15.3.4 波形生成”的内容）。

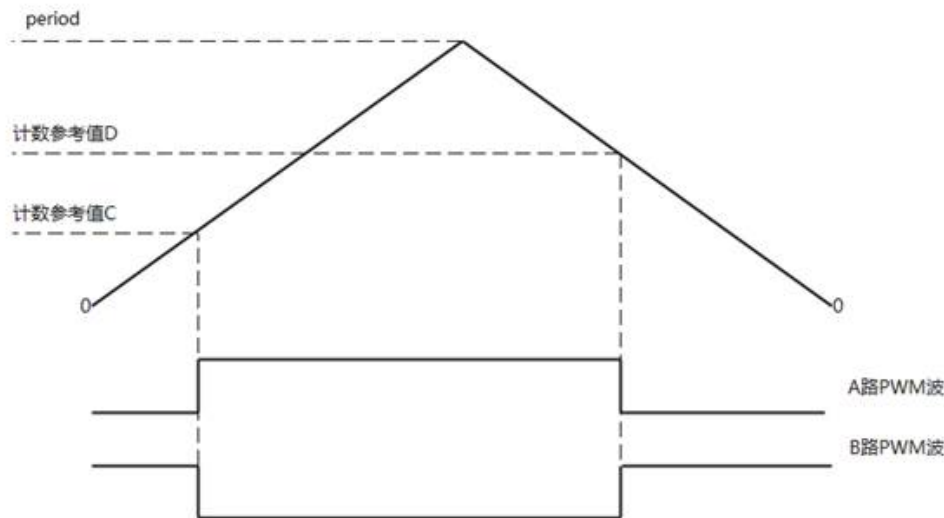
在产生基础的PWM波应用中通常只使用在计数器计数到达参考值点时触发波形控制动作。每个APT支持设置A/B/C/D一共4个计数参考值，计数参考值通过TC_REFA、TC_REFB、TC_REFC、TC_REFD寄存器进行设置。

计数参考值的动作在PG_ACT_A和PG_ACT_B寄存器中进行配置，其中PG_ACT_A指定APT输出A路的动作，PG_ACT_B指定APT输出B路的动作。

如#ZH-CN_TOPIC_0000001426060336/fig71021619142820所示，在参考值C点指定A路PWM波的动作作为高电平，在参考值D点指定A路PWM波的动作作为低电平，即可产生一个周期内的A路PWM波。如计数参考值点C/D不发生改变，在每个周期内都会执行同样的波形控制动作，从而产生持续的PWM波输出。



图 3-4 波形生成示意图



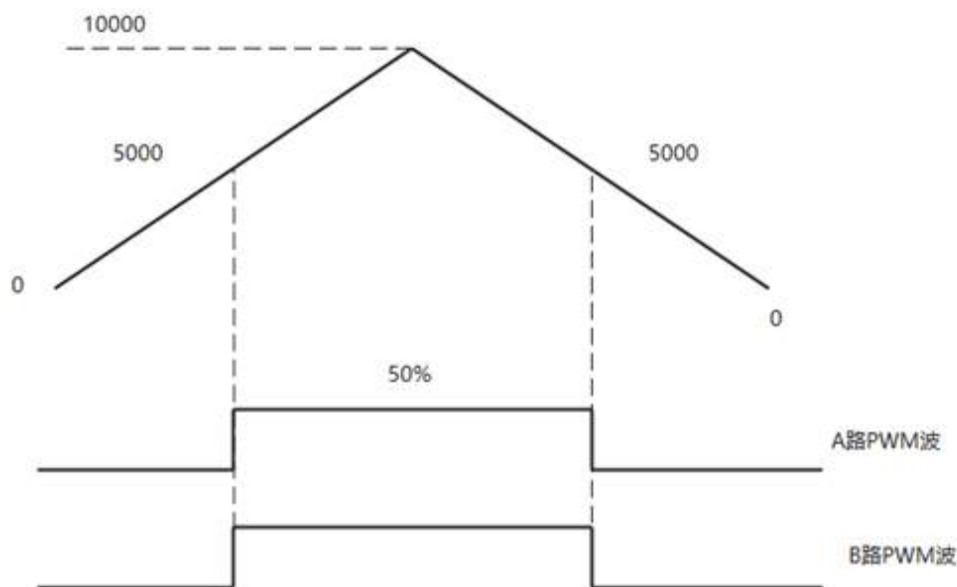
本应用中的配置。

本应用中使用计数参考值C和D用于波形的生成，其中：

- 配置参考值C初始值配置为5000。
- 配置参考值D初始值配置为5000。
- 配置A路在计数器递增到参考值C时的动作为高电平，在计数器递减到D点的动作为低电平。
- 配置B路在计数器递增到参考值C时的动作为高电平，在计数器递减到D点的动作为低电平。

生成的波形如#ZH-CN_TOPIC_0000001426060336/fig109018191260所示，频率为10kHz，占空比为50%。

图 3-5 例程波形生成示意图





通过A/B两路不同的波形变换动作可以组合成多种波形的输出方式，在HAL层接口中使用了basicType来定义这两路波形的关系。

- AHBL类型：表示在第一个翻转电平动作是：A路输出为低电平翻转为高电平，B路为输出高电平翻转到低电平。此时为两路反向的波形；
- ALBH类型：表示在第一个翻转电平的动作是：A路输出为高电平翻转为低电平，B路输出为低电平翻转为高电平。此时为两路反向的波形；
- AHBH类型：表示在第一个翻转电平的动作是：A路输出为低电平翻转为高电平，B路输出为低电平翻转为高电平。此时为两路同向的波形；
- ALBL类型：表示在第一个翻转电平的动作是：A路输出为高电平翻转为低电平，B路输出为高电平翻转为低电平。此时为两束同向的波形。

组合的输出方式如#ZH-CN_TOPIC_0000001426060336/fig1584212386303所示。

图 3-6 波形输出组合示意图



须知

此处仅介绍初始化的配置，算法应用运行的过程中会进行修改。

● 中断配置

每个APT都有2个中断事件，APT保护事件触发中断和APT的计数器触发的中断。

- 配置中断号。
- 配置计数器中断使能。
- 配置计数器触发中断的时机，本应用中配置为在计数为0时触发。
- 配置计数器触发中断的缩减比例，本应用中配置缩减比列为1。
- 注册中断回调函数。
- 设置中断优先级。
- 注册中断服务函数。
- 使能对应的中断。

```
static void APT0_Init(void)
{
    HAL_CRG_IpEnableSet(APT0_BASE, IP_CLK_ENABLE); /* 使能APT时钟 */
    g_ap0.baseAddress = APT0; /* 设置APT0寄存器地址 */
    g_ap0.irqNumEvt = IRQ_APT0_EVT; /* 使能APT0事件中断 */
    g_ap0.irqNumTmr = IRQ_APT0_TMR; /* 使能APT0定时器中断 */
    /* 设置APT0时钟分频系数1，APT时钟计数频率 = 200Mhz */
    g_ap0.waveform.dividerFactor = 1 - 1;
    /* 设置APT0时钟周期10K = 200Mhz / 10000 / 2 计数模式先增后减 */
    g_ap0.waveform.timerPeriod = 10000;
    g_ap0.waveform.cntMode = APT_COUNT_MODE_UP_DOWN; /* 设置APT0为上下计数模式 */
    /* Wave Form */
    g_ap0.waveform.basicType = APT_PWM_BASIC_A_HIGH_B_LOW; /* 波形类型为AHBL类型 */
    g_ap0.waveform.chAOutType = APT_PWM_OUT_BASIC_TYPE; /* A路输出PWM波 */
    g_ap0.waveform.chBOutType = APT_PWM_OUT_BASIC_TYPE; /* B路输出PWM波 */
    g_ap0.waveform.divInitVal = 0; /* 分频初始值为0 */
}
```



```
g_apt0.waveform.cntInitVal = 0; /* 计数器初始值为0 */
/* 初始化APT0计数器左比较值为5000，即初始化50%占空比 */
g_apt0.waveform.cntCmpLeftEdge = 5000;
/* 初始化APT0计数器右比较值为5000，即初始化50%占空比 */
g_apt0.waveform.cntCmpRightEdge = 5000;
/* 比较点值寄存器独立加载 */
g_apt0.waveform.cntCmpLoadMode = APT_BUFFER_INDEPENDENT_LOAD;
/* 比较值在计数为0时加载 */
g_apt0.waveform.cntCmpLoadEvt = APT_COMPARE_LOAD_EVENT_ZERO;
/* 设置死区时间1.5us，防止上下管直通，1.5 * 200 000 000 / 1000 000 = 300 */
g_apt0.waveform.deadBandCnt = 300;
/* ADC Trigger SOCA */
g_apt0.adcTrg.trgEnSOCA = BASE_CFG_ENABLE; /* 使能SOC的A路触发ADC采样 */
g_apt0.adcTrg.cntCmpSOCA = 1; /* 设置参考值A点的值为1 */
/* 递减计数到达参考值A点时触发ADC采样 */
g_apt0.adcTrg.trgSrcSOCA = APT_CS_SRC_CNTR_CMPA_DOWN;
g_apt0.adcTrg.trgScaleSOCA = 1; /* 触发SOC的A路的缩减比例为1，不进行缩减 */
/* ADC Trigger SOCB */
g_apt0.adcTrg.trgEnSOCB = BASE_CFG_ENABLE; /* 使能SOC的B路 */
g_apt0.adcTrg.cntCmpSOCB = 1; /* 设置参考值B点的值为1 */
/* 计数触发模式为递减计数到达参考值B点时触发 */
g_apt0.adcTrg.trgSrcSOCB = APT_CS_SRC_CNTR_CMPB_DOWN;
g_apt0.adcTrg.trgScaleSOCB = 1; /* 触发SOC的B路的缩减比例为1，不进行缩减 */
/* 触发ADC采样的参考值A/B的寄存器值的加载方式为独立加载 */
g_apt0.adcTrg.cntCmpLoadMode = APT_BUFFER_INDEPENDENT_LOAD;
/* 加载的时机为当计数等于0时 */
g_apt0.adcTrg.cntCmpLoadEvt = APT_COMPARE_LOAD_EVENT_ZERO;
/* Timer Trigger */
g_apt0.tmrInterrupt.tmrInterruptEn = BASE_CFG_ENABLE; /* 使能APT定时中断 */
/* 计数器计数到0时触发APT定时中断 */
g_apt0.tmrInterrupt.tmrInterruptSrc = APT_INT_SRC_CNTR_ZERO;
g_apt0.tmrInterrupt.tmrInterruptScale = 1; /* 缩减系数为1，不缩减 */
APT0_ProtectInit(); /* APT0过流保护初始化 */
HAL_APT_PWMInit(&g_apt0); /* PWM初始化 */
/* 注册保护事件中断回调函数，该事件中断配置为过流保护 */
HAL_APT_RegisterCallBack(&g_apt0, APT_EVENT_INTERRUPT, MotorSysErrCallback);
IRQ_SetPriority(g_apt0.irqNumEvt, 7); /* 设置保护事件中断的优先级 */
HAL_APT_IRQService(&g_apt0); /* 注册保护事件中断的服务函数 */
IRQ_EnableN(g_apt0.irqNumEvt); /* 使能保护事件中断 */
/* 注册计数器中断回调函数 */
HAL_APT_RegisterCallBack(&g_apt0, APT_TIMER_INTERRUPT, MotorCarrierProcessCallback);
IRQ_SetPriority(g_apt0.irqNumTmr, 6); /* 设置计数器中断中断优先级 */
HAL_APT_IRQService(&g_apt0); /* 注册计数器中断的服务函数 */
IRQ_EnableN(g_apt0.irqNumTmr); /* 使能计数器中断 */
}
```

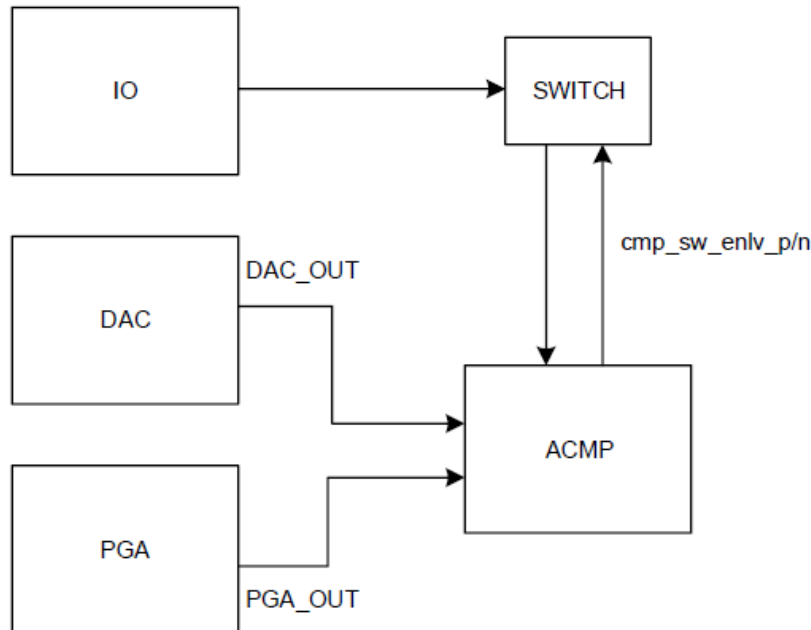
须知

当APT初始化完成后需要执行HAL_APT_StartModule(RUN_APT0 | RUN_APT1 | RUN_APT2)函数启动APT模块运行。

3.3.2.2 ACMP 模拟比较器

ACMP为模拟电压比较器，选择两个输入源进行电压比较，ACMP的比较信号有三个来源即管脚输出、DAC输出或PGA输出。具有端口选择、比较极性选择、迟滞电压选择功能，端口选择示意图如图3-7所示。

图 3-7 ACMP 与其他模块连接关系



1. 本应用中的例程配置。

- 步骤1** 配置比较器输入输出属性，配置CMP_CTRL寄存器0bit和1bit以及CMP_SW和CMP_CTRL1寄存器0bit~8bit对应位域。
- 步骤2** 配置输入信号滤波，配置CMP_CTRL寄存器8bit~141bit 和CMP_QUALI2寄存器的0bit~5bit和9bit~14bit对应位域。
- 步骤3** 配置迟滞电压，配置CMP_CTRL寄存器的3bit~4bit对应位域。

----结束

2. HAL层中相关参数的配置代码。

```
static void ACMP1_Init(void)
{
    HAL_CRG_IpEnableSet(ACMP1_BASE, BASE_CFG_ENABLE); /* 使能ACMP1时钟 */
    g_acmp1.baseAddress = ACMP1_BASE;
    g_acmp1.enable = true; /* 模块使能 */
    g_acmp1.syncEn = false; /* 同步功能关闭 */
    g_acmp1.inOutConfig.vinNNum = ACMP_VIN_MUX3; /* 配置输入负极来自3通道，即来自管脚 */
    g_acmp1.inOutConfig.vinPNum = ACMP_VIN_MUX3; /* 配置输入正极来自3通道，即来自管脚 */
    g_acmp1.inOutConfig.swVinPNum = ACMP_SW_VIN3; /* 配置复选开关为通道3的外部IO */
    g_acmp1.inOutConfig.swVinNNum = ACMP_SW_VIN3; /* 配置复选开关为通道3的外部IO */
    g_acmp1.inOutConfig.polarity = ACMP_OUT_NOT_INVERT; /* 输出结果不反相 */
    g_acmp1.filterCtrl.filterMode = ACMP_FILTER_NONE; /* 不加滤波 */
    g_acmp1.hysteresisVol = 0; /* 0: without hysteresis */
    HAL_ACMP_Init(&g_acmp1);
}
```

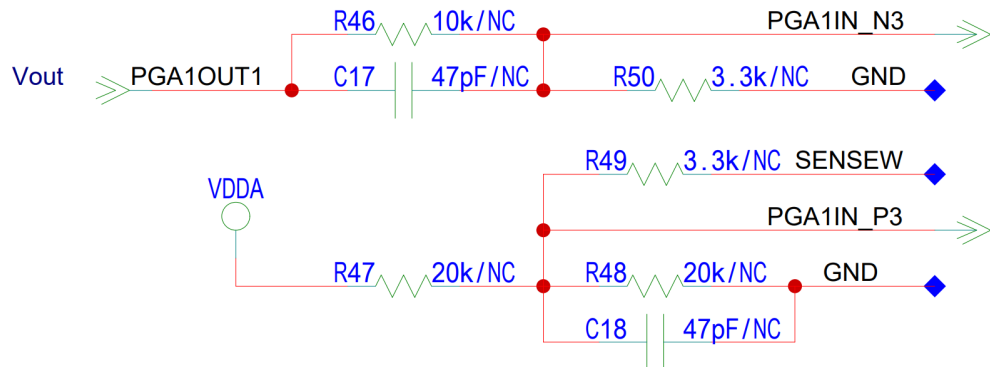
3.3.2.3 PGA 可编程增益放大器

PGA为可编程增益放大器，在本应用中用于作为ADC信号输入的前级放大器。PGA的放大增益可选择由外部电阻或内部电阻决定放大系数。使用内部电阻时可选择



1/2/4/8/16倍增益，使用外部电阻时PGA可视为普通的运放。在本应用中使用外部电阻如图3-8所示，放大系数由电路设计决定。

图 3-8 PGA 外围电路示意图



$$V_{out}=1.65V+3.03V_{in}$$

1. 本应用中的例程配置。

步骤1 使能PGA，配置PGA_CTRL0寄存器中的对应位域。

步骤2 设置PGA的增益为外部电阻决定，配置PGA_CTRL2.BIT.pga_smux寄存器。

步骤3 设置PGA的输入通道，配置PGA_CTRL3.BIT.pga_sw_enlv_n和PGA_CTRL3.BIT.pga_sw_enlv_p寄存器。

（如选用内部电阻进行放大，还需要配置PGA_CTRL2.BIT.pga_gain指定放大增益。）

----结束

须知

通道选择请参见《Hi306xx系列技术参考指南》中表24-2 PGA连接关系。

2. HAL层中相关参数的配置。

```
static void PGA0_Init(void)
{
    HAL_CRG_IpEnableSet(PGA0_BASE, IP_CLK_ENABLE); /* 使能PGA的时钟 */
    g_pga0.baseAddress = PGA0_BASE;
    g_pga0.enable = BASE_CFG_ENABLE; /* 使能PGA */
    g_pga0.extLoopbackEn = BASE_CFG_DISABLE; /* 不使能，通常不需要使用 */
    g_pga0.pgaMux = PGA_EXT_RES_VI0; /* 使用外部电阻 */
    g_pga0.pgaSwVinN = PGA_SW_VIN0; /* N端输入选择通道0 */
    g_pga0.pgaSwVinP = PGA_SW_VIN0; /* P端输入选择通道0 */
    g_pga0.gain = PGA_GAIN_1X; /* 使用外部电阻时不需要 */
    HAL_PGA_Init(&g_pga0); /* 调用PGA初始化接口 */
}
```

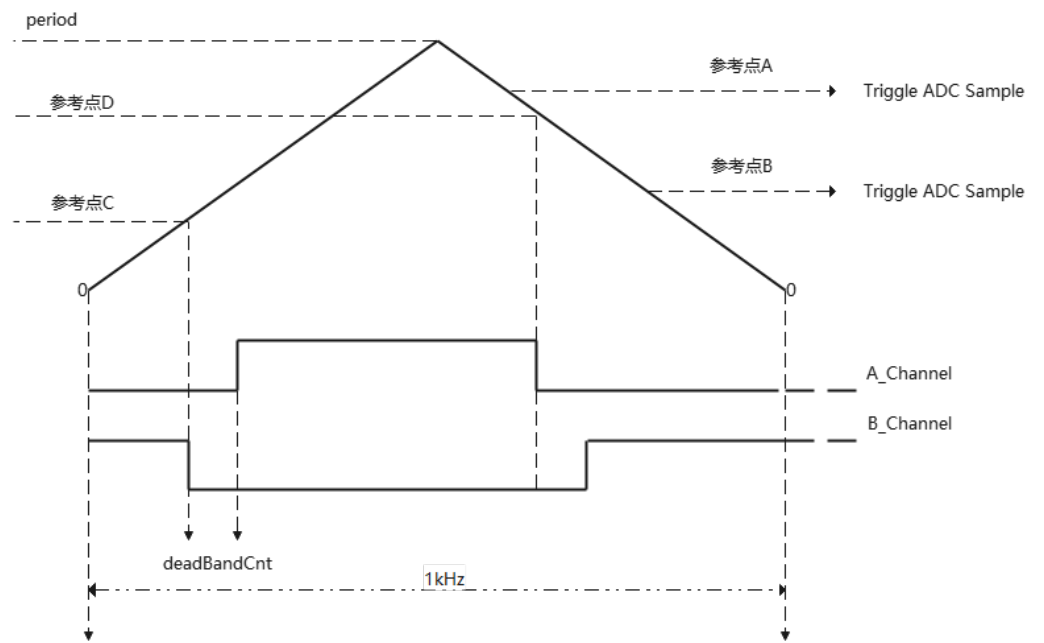


3.3.2.4 ADC 模数转换器

APT模块触发ADC采样的信号称为SOC（Start Of Conversion）信号，每个APT拥有A/B两路SOC信号（在下文中简称SOCA和SOCB信号）。

触发ADC采样的方式与触发波形生成动作的方式类似，也是计数器按照指定的计数方向计数到特定的参考值点时触发。

图 3-9 触发 ADC 采样示意图



- 1. 本应用中的例程配置。
指定TC_REFA为SOC的A路，TC_REFB为SOC的B路进行采样。

- 步骤1 使能SOC信号，A路的寄存器为CS_TMR_SELA.BIT.rg_csa_en_cs和B路的寄存器为CS_TMR_SELB.BIT.rg_csb_en_cs。
- 步骤2 配置计数参考值寄存器TC_REFA、TC_REFB、TC_REFC、TC_REFD。（在HAL层中指定TC_REFA为SOC的A路，TC_REFB为SOC的B路，如使用DCL层或直接配置寄存器可灵活配置）。
- 步骤3 配置触发条件选择寄存器CS_TMR_SELA.BIT.rg_csa_tmr_sel和CS_TMR_SELB.BIT.rg_csb_tmr_sel。
- 步骤4 配置缩减比例寄存器CS_PRSCA_CFG.BIT.rg_csa_prsc_prd和CS_PRSCB_CFG.BIT.rg_csb_prsc_prd。

----结束

- 2. HAL层中相关参数的配置代码。

```
static void APT0_Init(void)
{
    /* 省略代码在APT载波中断小节说明，此处仅对APT触发ADC采样的方式进行说明*/
    /* ADC Trigler SOCA */
    g_apUcp.adcTrg.trgEnSOCA = BASE_CFG_ENABLE; /* 使能SOC的A路*/
    g_apUcp.adcTrg.cntCmpSOCA = 1; /* 设置参考值A点的值为1 */
}
```



```
/* 递减计数到达参考值A点时触发*/  
g_apUcp.adcTrg.trgSrcSOCA = APT_CS_SRC_CNTR_CMPA_DOWN;  
g_apUcp.adcTrg.trgScaleSOCA = 1; /* 触发SOC的A路的缩减比例为1，不进行缩减*/  
/* ADC Triggler SOCB */  
g_apUcp.adcTrg.trgEnSOCB = BASE_CFG_ENABLE; /* 使能SOC的B路*/  
g_apUcp.adcTrg.cntCmpSOCB = 1; /* 设置参考值B点的值为1 */  
/* 计数触发模式为递减计数到达参考值B点时触发*/  
g_apUcp.adcTrg.trgSrcSOCB = APT_CS_SRC_CNTR_CMPB_DOWN;  
g_apUcp.adcTrg.trgScaleSOCB = 1; /* 触发SOC的B路的缩减比例为1，不进行缩减 */  
/* 触发ADC采样的参考值A/B的寄存器值的加载方式为独立加载 */  
g_apUcp.adcTrg.cntCmpLoadMode = APT_BUFFER_INDEPENDENT_LOAD;  
/* 加载的时机为当计数等于0 时 */  
g_apUcp.adcTrg.cntCmpLoadEvt = APT_COMPARE_LOAD_EVENT_ZERO;  
}
```

3.3.2.5 TIMER 基本定时器

TIMER为基本定时器，定时方式有自由计数、周期计数和一次计数。定时器1中断由TIMER1模块产生，下面以TIMER1_Init()函数为例详细讲解定时器的初始化配置及中断初始化过程：

1. 本应用中的例程配置。

在本应用中使用了TIMER的周期计数功能，用于每隔500us执行一次速度环。

步骤1 通过HAL_CRG_IpEnableSet()使能TIMER的时钟。

步骤2 配置TIMERx_LOAD寄存器，设置TIMER载入的计算初值。

步骤3 TIMER在周期计数模式下，若需要更改计数初值，需配置TIMERx_BGLOAD 寄存器，设置TIMER新的计数次数。

步骤4 TIMER模块内部支持分频，可通过TIMERx_CONTROL中的timerpre位配置预分频因子。

步骤5 调用IRQ_SetPriority设置TIMER的中断优先级，调用IRQ_EnableN使能中断。

步骤6 通过配置TIMERx_CONTROL中的timeren位使能TIMER。

步骤7 调用HAL_TIMER_Init完成TIMER初始化。

步骤8 编写一个中断处理函数，函数类型为 void (*callBack)(void *param)。

步骤9 调用HAL_TIMER_RegisterCallback将该中断处理函数注册到中断向量表中。

步骤10 需要使用TIMER计数时，要调用HAL_TIMER_Start启动TIMER。

----结束

须知

TIMER的周期计数模式：计数周期结束后，会触发HAL_TIMER_RegisterCallback中绑定的中断处理函数，在此函数中需清除TIMER的中断。

2. HAL层中相关参数的配置代码。

```
static void TIMER1_Init(void)  
{  
    /* TIMER计数周期为500us */  
    unsigned int load = HAL_CRG_GetIpFreq((void *)TIMER1) / 1000000u * 500;
```



```
HAL_CRG_IpEnableSet(TIMER1_BASE, IP_CLK_ENABLE);    /* 使能TIMER的时钟 */
HAL_CRG_IpClkSelectSet(TIMER1_BASE, CRG_PLL_NO_PREDV);
g_sysTickTimer.baseAddress = TIMER1;
g_sysTickTimer.irqNum = IRQ_TIMER1;
g_sysTickTimer.load = load - 1;    /* 设置TIMER的全局计数次数 */
g_sysTickTimer.bgLoad = load - 1;    /* 设置TIMER的重新加载计数次数 */
g_sysTickTimer.mode = TIMER_MODE_RUN_PERIODIC; /* 设置TIMER为周期计数模式 */
g_sysTickTimer.prescaler = TIMERPRESCALER_NO_DIV; /* 设置TIMER内分频为1 */
g_sysTickTimer.size = TIMER_SIZE_32BIT; /* 设置TIMER的计数器大小为32bit */
g_timer1.dmaAdcSingleReqEnable = BASE_CFG_DISABLE; /* 失能DMA */
g_timer1.dmaBurstReqEnable = BASE_CFG_DISABLE;
g_sysTickTimer.interruptEn = BASE_CFG_ENABLE;    /* 开启TIMER的中断 */
HAL_TIMER_Init(&g_sysTickTimer);
/* 注册中断回调函数 ISR_Systick 为注册的中断回调函数 每次周期计数完成之后 就会回调此函数 */
/* 在ISR_Systick 中用户需要调用HAL_TIMER_IrqClear()函数，清除中断 */
HAL_TIMER_RegisterCallback(&g_sysTickTimer, ISR_Systick);
IRQ_SetPriority(g_sysTickTimer.irqNum, 1);    /* 设置中断优先级 */
IRQ_EnableN(g_sysTickTimer.irqNum);    /* 使能TIMER的中断 */
}
```

须知

当定时器初始化完成后需要执行HAL_TIMER_Start(&g_sysTickTimer)函数启动定时器模块运行。

3.3.2.6 UART 串口收发器

UART为通用异步收发器，UART可作为程序信息的输入或输出，收发数据的形式有阻塞、中断和DMA模式。

1. 本应用中的例程配置。

在本应用中使用UART阻塞形式输出程序的信息。UART支持配置传输速率、FIFO深度、校验方式和停止位宽等。主要的配置的过程如下：

- 步骤1** 配置UART的时钟，HAL_CRG_IpEnableSet()使能UART的时钟。UART的主频默认配置为100MHz。
- 步骤2** 配置UART的波特率，通过配置寄存器UART_IBRD和UART_FBRD设置UART的工作波特率。波特率的具体配置方法请参见《Hi306xH技术参考指南》中“20.4 典型配置”的内容。
- 步骤3** 配置UART的寄存器UART_LCR_H，设置字长、停止位、奇偶校验和FIFO使能。
- 步骤4** 通过配置寄存器UART_IFLS来设置FIFO的深度，确定接收FIFO和发送FIFO的阈值。
- 步骤5** 调用HAL_UART_Init()接口，对UART进行初始化。

----结束



须知

使用DBG_PRINTF函数输出数据之前，必须初始化UART。

UART内部提供了阻塞发送数据和阻塞接收数据的HAL接口，分别为HAL_UART_WriteBlocking()和HAL_UART_ReadBlocking()，初始化完成之后，用户可以使用这两个接口发送或接收数据。

2. HAL层中相关参数的配置代码。

```
static void UART0_Init(void)
{
    HAL_CRG_IpEnableSet(UART0_BASE, IP_CLK_ENABLE); /* 使能UART的时钟 */
    HAL_CRG_IpClkSelectSet(UART0_BASE, CRG_PLL_NO_PREDV);
    g_uart0.baseAddress = UART0;
    g_uart0.irqNum = IRQ_UART0; /* 配置UART的中断号 */
    g_uart0.baudRate = UART0_BAND_RATE; /* 配置UART的波特率为
    UART0_BAND_RATE */
    g_uart0.dataLength = UART_DATALENGTH_8BIT; /* 设置UART的字长 */
    g_uart0.stopBits = UART_STOPBITS_ONE; /* 设置UART为1个停止位 */
    g_uart0.parity = UART_PARITY_NONE; /* 配置UART为无校验 */
    g_uart0.txMode = UART_MODE_BLOCKING; /* 设置UART发送数据为阻塞模式 */
    g_uart0.rxMode = UART_MODE_BLOCKING; /* 设置UART的接收数据为阻塞模式 */
    g_uart0.fifoTxThr = BASE_CFG_ENABLE; /* 使能UART的接收和发送FIFO */
    g_uart0.fifoRxThr = UART_FIFOFULL_ONE_TWO; /* 设置UART的发送FIFO阈值为1/2full */
    g_uart0.fifoRxThr = UART_FIFOFULL_ONE_TWO; /* 设置UART的接收FIFO阈值为1/2full */
    g_uart0.hwFlowCtr = BASE_CFG_DISABLE; /* 关闭UART的硬件流控模式 */
    HAL_UART_Init(&g_uart0); /* 初始化UART */
}
```

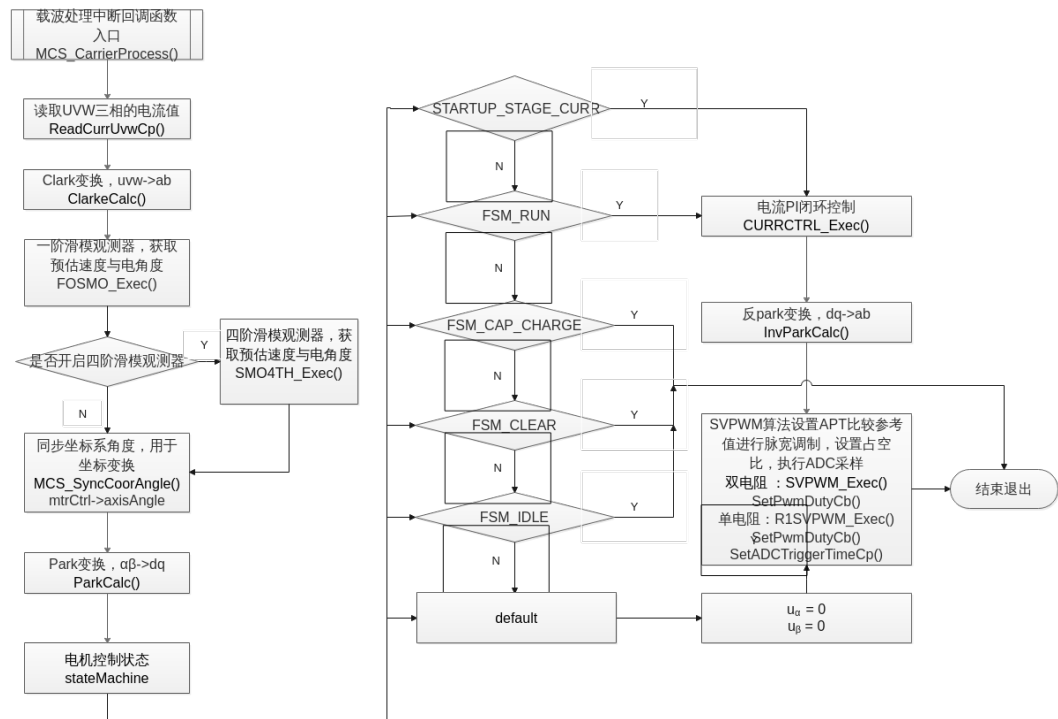
3.3.3 载波控制流程介绍

载波中断函数MotorCarrierProcessCallback()在APT初始化中配置并使能，设置在每个APT载波的波谷触发中断函数，中断执行周期为100us。中断执行函数包含了坐标系变换、滑模状态观测器、不同状态观测角度的切换、电流环执行以及对SVPWM计算出的PWM占空比数值更新等。

在本函数中首先获取三相电流值进行ClarkeCalc()变换，将变换之后的 i_α 、 i_β 电流数据信息送入滑模观测器用于角度和速度值的估算以及后续的ParkCalc()变换，从而将变换后得到的 i_d 、 i_q 值送入电流环PI控制器进行电流闭环调节。值得注意的是，例程中提供了一阶和四阶滑模估测算法，默认采用四阶滑模估测结果，用户也可通过注释"SMO4TH"宏定义的方式切换为一阶滑模估算数据进行电机控制。

当电机经过充电、clark变换、角度计算、park变换、PI电流环、反park变换和svpwm处理后，最终得出需要的PWM计数值信息并更新到相应的APT执行模块中。其具体处理流程指导请参见2 无传感器磁场定向控制（FOC）章节。载波中断执行流程介绍如图3-10所示。

图 3-10 载波中断执行流程图



3.3.4 定时器中断执行流程

定时器1中断处理函数MotorStatemachineCallBack()在TIMER1初始化中配置并使能，配置其中断周期为500us。在该中断中轮询电机状态机，包含电机启动前的启动信号获取、电容充电过程、电机开环启动、电机速度开/闭环切换运行以及速度闭环运行后的过流等错误状态检测。

- 步骤1** 状态机FSM_IDLE状态：程序烧录运行后默认为停止状态，在该状态时中断函数轮询电机启动信号，当接收到启动信号后状态机设置为FSM_CAP_CHARGE状态，并设置三相驱动桥路下桥臂导通，使能APT的PWM输出为电容充电做准备。
- 步骤2** 状态机FSM_CAP_CHARGE状态：在该状态中利用定时器中断的计数功能对计时变量sysTickCnt自加，当该数值等于设定的充电时长目标值后切换为FSM_CLEAR状态，电容充电时长为3ms。
- 步骤3** 状态机FSM_CLEAR状态：该状态下对电机控制所涉及的变量参数进行初始化，完成后进入FSM_STARTUP状态。
- 步骤4** 状态机FSM_STARTUP状态：在该状态中完成电机的IF拖动，速度开环闭环动作。根据2.3 PI控制器以及2.7 IF启动控制小节所述，首先给电机q轴逐步充电至目标电流值，随后设定速度斜坡上升RMG_Exec()，此时的电机角度由速度积分离散化获得，角度值的获取见MCS_SyncCoorAngle()函数。当拖动电机至大于等于设定目标值后，进入下一阶段。进入下一阶段后执行速度闭环函数SPDCTRL_Exec()，并将d轴电流逐步控为零。当速度大于等于设定的闭环切入结束点后将d轴控为零并进入状态机FSM_RUN状态。
- 步骤5** 状态机FSM_RUN状态：在该状态中速度同样以斜坡的方式增加至设定速度目标值，并实时保持速度闭环运行。
- 步骤6** 状态机FSM_STOP状态：电机停止状态，关闭pwm输出并将状态机设置为初始状态。

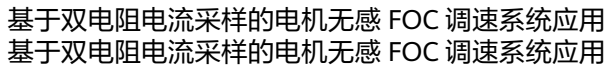
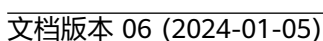


图 3-11 定时器中断执行流程图



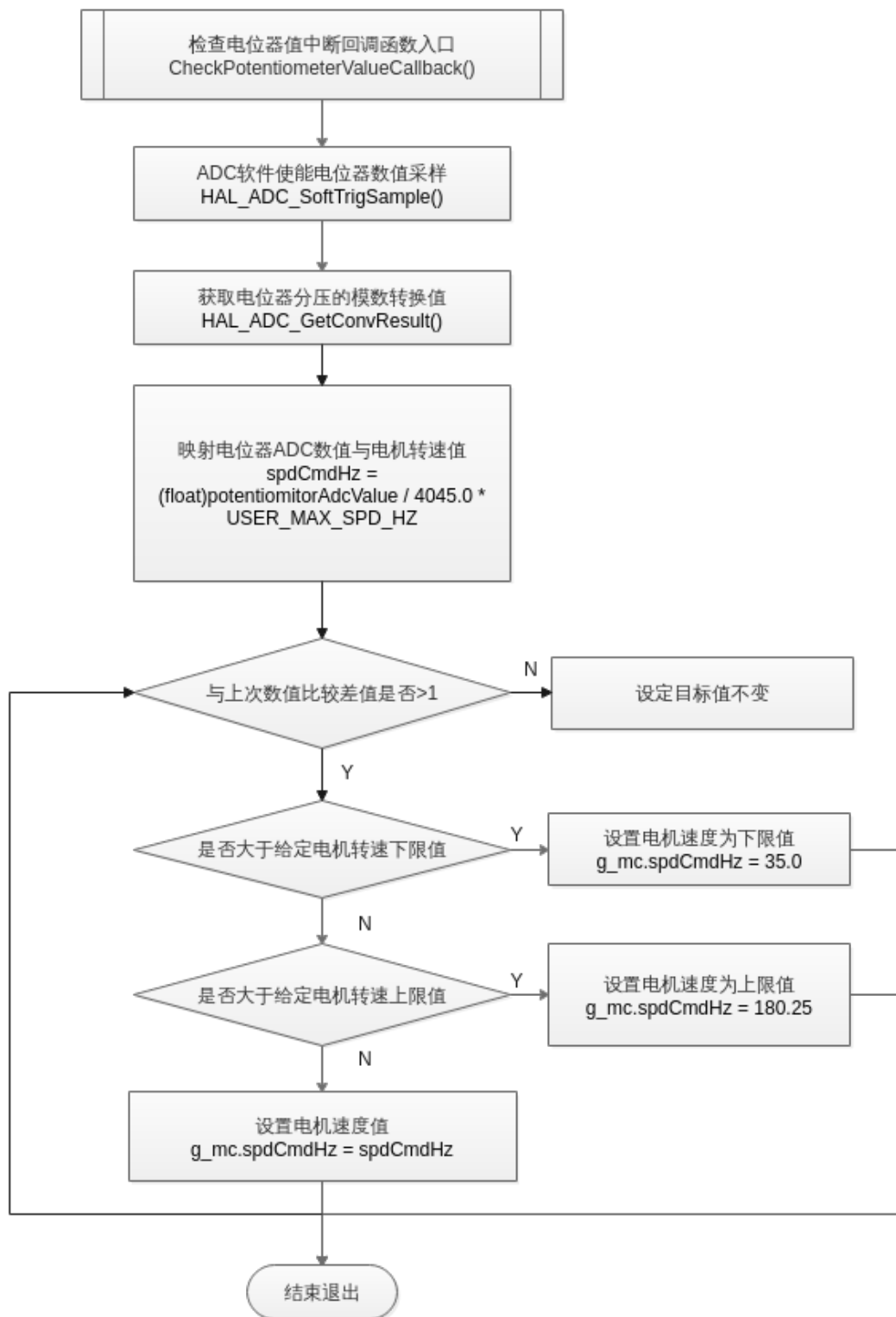


3.3.5 电位器调速流程介绍

电位器数值的读取在CheckPotentiometerValueCallback()中断函数中完成，由TIMER0初始化配置并使能，中断时间1秒。通过ADC检测滑动变阻器的分压值并做等比例缩放，从而设定电机转速。电位器的量程为0~4096，为了维持一定的裕量，将程序中的电位器最大值设定为4045并对转化后的速度目标值做限幅处理，限制幅度为35Hz~180.25Hz电频率，由于电机极对数为7，对应电机机械转速为 $(35-180.25)\text{Hz} \times 60/7 = 300 - 1545\text{RPM}$ ，流程如图3-12所示。



图 3-12 电位器调速执行流程图





4 参考文献

1. IDE工具使用说明文档：《调试器 使用指南》、《IDE 使用指南》。
2. 306xx MCU（Microcontroller Unit）数据手册和技术文档：《Hi306xx系列数据手册》、《Hi306xx系列技术参考指南》。
3. 《Hi306xx通用生态板用户手册》。



5 缩略语

缩略语	英文	中文
BLDC	Brushless Direct Current Motor	无刷直流电机
FOC	Field-Oriented Control	磁场定向控制
PMSM	Permanent Magnet Synchronous Motor	永磁同步电机
MCU	Microcontroller Unit	微控制器单元
PWM	Pulse Width Modulation	脉冲宽度调制
SVPWM	Space Vector Pulse Width Modulation	空间矢量脉宽调制