



电机控制基础算法库 API 说明

文档版本 03

发布日期 2024-09-14

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






本文档描述了SolarA²电控基础算法库API使用的数据结构定义，以及控制模块和保护模块的参数和接口。

读者对象

本文档（本指南）主要适用于电机控制开发工程师。

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。



修订记录

修订日期	版本	修订说明
2022-06-21	00B01	第1次临时版本发布。
2023-07-24	00B02	第2次临时版本发布。 刷新2.3章节故障检测与保护模块的内容； 刷新5.2章节API接口； 刷新18.2.1章节电机故障状态定义联合体说明； 刷新18.2.2章节电机故障状态和保护等级枚举说明。
2023-10-13	00B03	第3次临时版本发布。 对照新算法库，刷新所有文档描述。
2023-12-07	00B04	第4次临时版本发布。 观测器文件夹由observe更名为observer。 添加FOC控制框图图示和刹车图示说明。 根据专家评审意见进行相关模块的修改，如API的返回参数，角度与弧度的对应关系等。
2024-01-08	00B05	第5次临时版本发布。 新增 参数校验 章节。
2024-02-22	00B06	第6次临时版本发布（算法TR4A）。
2024-03-12	00B07	第7次临时版本发布。 将 bool SysStartCmdSet 更正为 void SysStartCmdSet。
2024-03-21	00B08	第8次临时版本发布。 修改了前后文名称不统一的情况。
2024-4-23	01	第1次正式版本发布。 修改弱磁Handle成员变量udcThreshPer描述。 修改弱磁初始化入参thr描述。
2024-05-27	02	第2次正式版本发布。 新增 位置控制器 描述。
2024-09-14	03	第3次正式版本发布。 新增 pid控制器模块 描述。新增抗反馈值扰动PID接口数据结构和接口描述。



目 录

前 言..... i

1 概述..... 1

2 文件目录说明..... 2

 2.1 通用数据类型定义..... 2

 2.2 控制模块..... 2

3 物理量单位说明..... 4

 3.1 角度..... 4

 3.2 速度..... 4

4 utilities 通用模块..... 5

 4.1 参数校验..... 5

 4.2 电机控制坐标系结构定义..... 5

 4.3 电机属性..... 6

 4.3.1 电机属性 handle 结构体说明..... 6

 4.3.2 API 接口..... 7

 4.3.2.1 MtrParamInit..... 7

 4.3.2.1.1 API 形式..... 7

 4.3.2.1.2 API 功能描述..... 7

 4.3.2.1.3 Example..... 7

 4.3.2.1.4 注意事项..... 7

 4.4 系统状态..... 8

 4.4.1 系统状态 handle 结构体说明..... 8

 4.4.2 API 接口..... 8

 4.4.2.1 SysGetCmdStart..... 8

 4.4.2.1.1 API 形式..... 8

 4.4.2.1.2 API 功能描述..... 8

 4.4.2.1.3 Example..... 9

 4.4.2.1.4 注意事项..... 9

 4.4.2.2 SysCmdStartSet..... 9

 4.4.2.2.1 API 形式..... 9

 4.4.2.2.2 API 功能描述..... 9

 4.4.2.2.3 Example..... 9

 4.4.2.2.4 注意事项..... 9



4.4.2.3 SysCmdStartClr..... 9

4.4.2.3.1 API 形式..... 9

4.4.2.3.2 API 功能描述..... 10

4.4.2.3.3 Example..... 10

4.4.2.3.4 注意事项..... 10

4.4.2.4 SysGetCmdStop..... 10

4.4.2.4.1 API 形式..... 10

4.4.2.4.2 API 功能描述..... 10

4.4.2.4.3 Example..... 10

4.4.2.4.4 注意事项..... 10

4.4.2.5 SysCmdStopSet..... 10

4.4.2.5.1 API 形式..... 11

4.4.2.5.2 API 功能描述..... 11

4.4.2.5.3 Example..... 11

4.4.2.5.4 注意事项..... 11

4.4.2.6 SysCmdStopClr..... 11

4.4.2.6.1 API 形式..... 11

4.4.2.6.2 API 功能描述..... 11

4.4.2.6.3 Example..... 11

4.4.2.6.4 注意事项..... 12

4.4.2.7 SysIsRunning..... 12

4.4.2.7.1 API 形式..... 12

4.4.2.7.2 API 功能描述..... 12

4.4.2.7.3 Example..... 12

4.4.2.7.4 注意事项..... 12

4.4.2.8 SysRunningSet..... 12

4.4.2.8.1 API 形式..... 12

4.4.2.8.2 API 功能描述..... 13

4.4.2.8.3 Example..... 13

4.4.2.8.4 注意事项..... 13

4.4.2.9 SysRunningClr..... 13

4.4.2.9.1 API 形式..... 13

4.4.2.9.2 API 功能描述..... 13

4.4.2.9.3 Example..... 13

4.4.2.9.4 注意事项..... 13

4.4.2.10 SysIsError..... 13

4.4.2.10.1 API 形式..... 13

4.4.2.10.2 API 功能描述..... 14

4.4.2.10.3 Example..... 14

4.4.2.10.4 注意事项..... 14

4.4.2.11 SysErrorSet..... 14

4.4.2.11.1 API 形式..... 14



4.4.2.11.2 API 功能描述.....	14
4.4.2.11.3 Example.....	14
4.4.2.11.4 注意事项.....	14
4.4.2.12 SysErrorClr.....	15
4.4.2.12.1 API 形式.....	15
4.4.2.12.2 API 功能描述.....	15
4.4.2.12.3 Example.....	15
4.4.2.12.4 注意事项.....	15
5 adc_calibr ADC 校准模块.....	16
5.1 ADC 校准.....	16
5.1.1 ADC 校准 handle 结构体说明.....	16
5.1.2 API 接口.....	17
5.1.2.1 ADCCALIBR_Init.....	17
5.1.2.1.1 API 形式.....	17
5.1.2.1.2 API 功能描述.....	17
5.1.2.1.3 Example.....	17
5.1.2.1.4 注意事项.....	17
5.1.2.2 ADCCALIBR_Exec.....	17
5.1.2.2.1 API 形式.....	18
5.1.2.2.2 API 功能描述.....	18
5.1.2.2.3 Example.....	18
5.1.2.2.4 注意事项.....	18
5.1.2.3 ADCCALIBR_IsFinish.....	18
5.1.2.3.1 API 形式.....	18
5.1.2.3.2 API 功能描述.....	18
5.1.2.3.3 Example.....	19
5.1.2.3.4 注意事项.....	19
6 brake 刹车模块.....	20
6.1 刹车.....	20
6.1.1 刹车 handle 结构体说明.....	21
6.1.2 API 接口.....	22
6.1.2.1 BRAKE_Init.....	22
6.1.2.1.1 API 形式.....	22
6.1.2.1.2 API 功能描述.....	23
6.1.2.1.3 Example.....	23
6.1.2.1.4 注意事项.....	23
6.1.2.2 BRAKE_Clear.....	23
6.1.2.2.1 API 形式.....	23
6.1.2.2.2 API 功能描述.....	23
6.1.2.2.3 Example.....	23
6.1.2.2.4 注意事项.....	23
6.1.2.3 BRAKE_Exec.....	24



6.1.2.3.1 API 形式.....	24
6.1.2.3.2 API 功能描述.....	24
6.1.2.3.3 Example.....	24
6.1.2.3.4 注意事项.....	24
7 filter 滤波器模块.....	25
7.1 一阶低通滤波器.....	25
7.1.1 一阶滤波器 handle 结构.....	25
7.1.2 API 接口.....	25
7.1.2.1 FOLPF_Init.....	26
7.1.2.1.1 API 形式.....	26
7.1.2.1.2 API 功能描述.....	26
7.1.2.1.3 Example.....	26
7.1.2.1.4 注意事项.....	26
7.1.2.2 FOLPF_Clear.....	27
7.1.2.2.1 API 形式.....	27
7.1.2.2.2 API 功能描述.....	27
7.1.2.2.3 Example.....	27
7.1.2.2.4 注意事项.....	27
7.1.2.3 FOLPF_Exec.....	27
7.1.2.3.1 API 形式.....	27
7.1.2.3.2 API 功能描述.....	27
7.1.2.3.3 Example.....	28
7.1.2.3.4 注意事项.....	28
7.1.2.4 FOLPF_SetTs.....	28
7.1.2.4.1 API 形式.....	28
7.1.2.4.2 API 功能描述.....	28
7.1.2.4.3 Example.....	28
7.1.2.4.4 注意事项.....	28
7.2 四阶龙格-库塔离散化一阶低通滤波器.....	28
7.2.1 四阶龙格-库塔低通滤波器 handle 结构体说明.....	28
7.2.2 API 接口.....	29
7.2.2.1 LPFRK4_Clear.....	29
7.2.2.1.1 API 形式.....	29
7.2.2.1.2 API 功能描述.....	29
7.2.2.1.3 Example.....	29
7.2.2.1.4 注意事项.....	29
7.2.2.2 LPFRK4_Exec.....	29
7.2.2.2.1 API 形式.....	30
7.2.2.2.2 API 功能描述.....	30
7.2.2.2.3 Example.....	30
7.2.2.2.4 注意事项.....	30
7.3 PLL 锁相环.....	30



7.3.1 锁相环 handle 结构.....	30
7.3.2 API 接口.....	31
7.3.2.1 PLL_Init.....	31
7.3.2.1.1 API 形式.....	31
7.3.2.1.2 API 功能描述.....	31
7.3.2.1.3 Example.....	32
7.3.2.1.4 注意事项.....	32
7.3.2.2 PLL_Reset.....	32
7.3.2.2.1 API 形式.....	32
7.3.2.2.2 API 功能描述.....	32
7.3.2.2.3 Example.....	32
7.3.2.2.4 注意事项.....	32
7.3.2.3 PLL_Clear.....	32
7.3.2.3.1 API 形式.....	32
7.3.2.3.2 API 功能描述.....	33
7.3.2.3.3 Example.....	33
7.3.2.3.4 注意事项.....	33
7.3.2.4 PLL_Exec.....	33
7.3.2.4.1 API 形式.....	33
7.3.2.4.2 API 功能描述.....	33
7.3.2.4.3 Example.....	33
7.3.2.4.4 注意事项.....	34
7.3.2.5 PLL_SetTs.....	34
7.3.2.5.1 API 形式.....	34
7.3.2.5.2 API 功能描述.....	34
7.3.2.5.3 Example.....	34
7.3.2.5.4 注意事项.....	34
7.3.2.6 PLL_ParamUpdate.....	34
7.3.2.6.1 API 形式.....	34
7.3.2.6.2 API 功能描述.....	35
7.3.2.6.3 Example.....	35
7.3.2.6.4 注意事项.....	35
8 foc_loop_ctrl FOC 控制模块.....	36
8.1 电流控制器.....	36
8.1.1 电流控制器 handle 结构体说明.....	36
8.1.2 API 接口.....	37
8.1.2.1 CURRCTRL_Init.....	37
8.1.2.1.1 API 形式.....	37
8.1.2.1.2 API 功能描述.....	37
8.1.2.1.3 Example.....	37
8.1.2.1.4 注意事项.....	38
8.1.2.2 CURRCTRL_Reset.....	38



8.1.2.2.1 API 形式.....	38
8.1.2.2.2 API 功能描述.....	38
8.1.2.2.3 Example.....	38
8.1.2.2.4 注意事项.....	38
8.1.2.3 CURRCTRL_Clear.....	38
8.1.2.3.1 API 形式.....	38
8.1.2.3.2 API 功能描述.....	39
8.1.2.3.3 Example.....	39
8.1.2.3.4 注意事项.....	39
8.1.2.4 CURRCTRL_Exec.....	39
8.1.2.4.1 API 形式.....	39
8.1.2.4.2 API 功能描述.....	39
8.1.2.4.3 Example.....	40
8.1.2.4.4 注意事项.....	40
8.1.2.5 CURRCTRL_SetTs.....	40
8.1.2.5.1 API 形式.....	40
8.1.2.5.2 API 功能描述.....	40
8.1.2.5.3 Example.....	40
8.1.2.5.4 注意事项.....	40
8.2 电流环前馈控制.....	40
8.2.1 API 接口.....	41
8.2.1.1 CURRFF_Exec.....	41
8.2.1.1.1 API 形式.....	41
8.2.1.1.2 API 功能描述.....	41
8.2.1.1.3 Example.....	41
8.2.1.1.4 注意事项.....	41
8.3 弱磁控制.....	41
8.3.1 弱磁控制器 handle 结构体说明.....	42
8.3.2 API 接口.....	42
8.3.2.1 FW_Init.....	42
8.3.2.1.1 API 形式.....	43
8.3.2.1.2 API 功能描述.....	43
8.3.2.1.3 Example.....	43
8.3.2.1.4 注意事项.....	43
8.3.2.2 FW_Exec.....	43
8.3.2.2.1 API 形式.....	43
8.3.2.2.2 API 功能描述.....	44
8.3.2.2.3 Example.....	44
8.3.2.2.4 注意事项.....	44
8.3.2.3 FW_SetTs.....	44
8.3.2.3.1 API 形式.....	44
8.3.2.3.2 API 功能描述.....	44



8.3.2.3.3 Example..... 45

8.3.2.3.4 注意事项..... 45

8.3.2.4 FW_Clear.....45

8.3.2.4.1 API 形式..... 45

8.3.2.4.2 API 功能描述..... 45

8.3.2.4.3 Example..... 45

8.3.2.4.4 注意事项..... 45

8.4 IF 控制器..... 45

8.4.1 IF 控制器 handle 结构体说明.....45

8.4.1.1 IF 控制器 handle 结构..... 45

8.4.2 API 接口..... 46

8.4.2.1 IF_Init..... 46

8.4.2.1.1 API 形式..... 46

8.4.2.1.2 API 功能描述..... 47

8.4.2.1.3 Example..... 47

8.4.2.1.4 注意事项..... 47

8.4.2.2 IF_Clear..... 47

8.4.2.2.1 API 形式..... 47

8.4.2.2.2 API 功能描述..... 47

8.4.2.2.3 Example..... 47

8.4.2.2.4 注意事项..... 47

8.4.2.3 IF_CurrAmpCalc.....48

8.4.2.3.1 API 形式..... 48

8.4.2.3.2 API 功能描述..... 48

8.4.2.3.3 Example..... 48

8.4.2.3.4 注意事项..... 48

8.4.2.4 IF_CurrAngleCalc.....48

8.4.2.4.1 API 形式..... 48

8.4.2.4.2 API 功能描述..... 49

8.4.2.4.3 Example..... 49

8.4.2.4.4 注意事项..... 49

8.4.2.5 IF_SetAngleTs..... 49

8.4.2.5.1 API 形式..... 49

8.4.2.5.2 API 功能描述..... 49

8.4.2.5.3 Example..... 49

8.4.2.5.4 注意事项..... 49

8.5 位置控制器..... 49

8.5.1 位置控制器 handle 结构体说明.....50

8.5.2 API 接口..... 51

8.5.2.1 POSCTRL_Clear..... 51

8.5.2.1.1 API 形式..... 51

8.5.2.1.2 API 功能描述..... 52



8.5.2.1.3 Example..... 52

8.5.2.1.4 注意事项..... 52

8.5.2.2 POSCTRL_Init..... 52

8.5.2.2.1 API 形式..... 52

8.5.2.2.2 API 功能描述..... 52

8.5.2.2.3 Example..... 52

8.5.2.2.4 注意事项..... 53

8.5.2.3 POSCTRL_PidExec..... 53

8.5.2.3.1 API 形式..... 53

8.5.2.3.2 API 功能描述..... 53

8.5.2.3.3 Example..... 53

8.5.2.3.4 注意事项..... 53

8.5.2.4 POSCTRL_ModeSelect..... 53

8.5.2.4.1 API 形式..... 53

8.5.2.4.2 API 功能描述..... 54

8.5.2.4.3 Example..... 54

8.5.2.4.4 注意事项..... 54

8.5.2.5 POSCTRL_SetSlope..... 54

8.5.2.5.1 API 形式..... 54

8.5.2.5.2 API 功能描述..... 54

8.5.2.5.3 Example..... 55

8.5.2.5.4 注意事项..... 55

8.5.2.6 POSCTRL_SetTarget..... 55

8.5.2.6.1 API 形式..... 55

8.5.2.6.2 API 功能描述..... 55

8.5.2.6.3 Example..... 55

8.5.2.6.4 注意事项..... 55

8.5.2.7 POSCTRL_Exec..... 55

8.5.2.7.1 API 形式..... 56

8.5.2.7.2 API 功能描述..... 56

8.5.2.7.3 Example..... 56

8.5.2.7.4 注意事项..... 56

8.5.2.8 POSCTRL_AngleExpand..... 56

8.5.2.8.1 API 形式..... 56

8.5.2.8.2 API 功能描述..... 57

8.5.2.8.3 Example..... 57

8.5.2.8.4 注意事项..... 57

8.5.2.9 POSCTRL_SetKp..... 57

8.5.2.9.1 API 形式..... 57

8.5.2.9.2 API 功能描述..... 57

8.5.2.9.3 Example..... 57

8.5.2.9.4 注意事项..... 57



8.5.2.10 POSCTRL_SetKi.....	57
8.5.2.10.1 API 形式.....	58
8.5.2.10.2 API 功能描述.....	58
8.5.2.10.3 Example.....	58
8.5.2.10.4 注意事项.....	58
8.5.2.11 POSCTRL_SetKd.....	58
8.5.2.11.1 API 形式.....	58
8.5.2.11.2 API 功能描述.....	58
8.5.2.11.3 Example.....	59
8.5.2.11.4 注意事项.....	59
8.5.2.12 POSCTRL_SetNs.....	59
8.5.2.12.1 API 形式.....	59
8.5.2.12.2 API 功能描述.....	59
8.5.2.12.3 Example.....	59
8.5.2.12.4 注意事项.....	59
8.6 速度控制器.....	59
8.6.1 速度控制器 handle 结构体说明.....	59
8.6.2 API 接口.....	60
8.6.2.1 SPDCTRL_Init.....	60
8.6.2.1.1 API 形式.....	60
8.6.2.1.2 API 功能描述.....	61
8.6.2.1.3 Example.....	61
8.6.2.1.4 注意事项.....	61
8.6.2.2 SPDCTRL_Reset.....	61
8.6.2.2.1 API 形式.....	61
8.6.2.2.2 API 功能描述.....	61
8.6.2.2.3 Example.....	61
8.6.2.2.4 注意事项.....	61
8.6.2.3 SPDCTRL_Clear.....	61
8.6.2.3.1 API 形式.....	62
8.6.2.3.2 API 功能描述.....	62
8.6.2.3.3 Example.....	62
8.6.2.3.4 注意事项.....	62
8.6.2.4 SPDCTRL_Exec.....	62
8.6.2.4.1 API 形式.....	62
8.6.2.4.2 API 功能描述.....	62
8.6.2.4.3 Example.....	63
8.6.2.4.4 注意事项.....	63
8.7 启动过程开闭环切换控制器.....	63
8.7.1 启动过程开闭环切换控制器 handle 结构体说明.....	63
8.7.1.1 启动过程枚举结构.....	63
8.7.1.2 切换过程控制器 handle 结构.....	63



8.7.2 API 接口..... 64

8.7.2.1 STARTUP_Init..... 64

8.7.2.1.1 API 形式..... 64

8.7.2.1.2 API 功能描述..... 64

8.7.2.1.3 Example..... 64

8.7.2.1.4 注意事项..... 64

8.7.2.2 STARTUP_Clear..... 65

8.7.2.2.1 API 形式..... 65

8.7.2.2.2 API 功能描述..... 65

8.7.2.2.3 Example..... 65

8.7.2.2.4 注意事项..... 65

8.7.2.3 STARTUP_CurrCal..... 65

8.7.2.3.1 API 形式..... 65

8.7.2.3.2 API 功能描述..... 65

8.7.2.3.3 Example..... 66

8.7.2.3.4 注意事项..... 66

9 math 模块..... 67

9.1 基础数学方法..... 67

9.1.1 三角函数计算 handle 结构体说明..... 67

9.1.2 API 接口..... 67

9.1.2.1 GetSin..... 68

9.1.2.1.1 API 形式..... 68

9.1.2.1.2 API 功能描述..... 68

9.1.2.1.3 Example..... 68

9.1.2.1.4 注意事项..... 68

9.1.2.2 GetCos..... 68

9.1.2.2.1 API 形式..... 69

9.1.2.2.2 API 功能描述..... 69

9.1.2.2.3 Example..... 69

9.1.2.2.4 注意事项..... 69

9.1.2.3 TrigCalc..... 69

9.1.2.3.1 API 形式..... 69

9.1.2.3.2 API 功能描述..... 69

9.1.2.3.3 Example..... 70

9.1.2.3.4 注意事项..... 70

9.1.2.4 ParkCalc..... 70

9.1.2.4.1 API 形式..... 70

9.1.2.4.2 API 功能描述..... 70

9.1.2.4.3 Example..... 70

9.1.2.4.4 注意事项..... 70

9.1.2.5 InvParkCalc..... 70

9.1.2.5.1 API 形式..... 71



9.1.2.5.2 API 功能描述..... 71

9.1.2.5.3 Example..... 71

9.1.2.5.4 注意事项..... 71

9.1.2.6 ClarkeCalc..... 71

9.1.2.6.1 API 形式..... 71

9.1.2.6.2 API 功能描述..... 72

9.1.2.6.3 Example..... 72

9.1.2.6.4 注意事项..... 72

9.1.2.7 Abs..... 72

9.1.2.7.1 API 形式..... 72

9.1.2.7.2 API 功能描述..... 72

9.1.2.7.3 Example..... 72

9.1.2.7.4 注意事项..... 73

9.1.2.8 Clamp..... 73

9.1.2.8.1 API 形式..... 73

9.1.2.8.2 API 功能描述..... 73

9.1.2.8.3 Example..... 73

9.1.2.8.4 注意事项..... 73

9.1.2.9 Max..... 73

9.1.2.9.1 API 形式..... 73

9.1.2.9.2 API 功能描述..... 74

9.1.2.9.3 Example..... 74

9.1.2.9.4 注意事项..... 74

9.1.2.10 Min..... 74

9.1.2.10.1 API 形式..... 74

9.1.2.10.2 API 功能描述..... 74

9.1.2.10.3 Example..... 74

9.1.2.10.4 注意事项..... 75

9.1.2.11 Sqrt..... 75

9.1.2.11.1 API 形式..... 75

9.1.2.11.2 API 功能描述..... 75

9.1.2.11.3 Example..... 75

9.1.2.11.4 注意事项..... 75

9.1.2.12 AngleSub..... 75

9.1.2.12.1 API 形式..... 75

9.1.2.12.2 API 功能描述..... 76

9.1.2.12.3 Example..... 76

9.1.2.12.4 注意事项..... 76

9.1.2.13 Sat..... 76

9.1.2.13.1 API 形式..... 76

9.1.2.13.2 API 功能描述..... 76

9.1.2.13.3 Example..... 77



9.1.2.13.4 注意事项.....	77
9.1.2.14 Mod.....	77
9.1.2.14.1 API 形式.....	77
9.1.2.14.2 API 功能描述.....	77
9.1.2.14.3 Example.....	77
9.1.2.14.4 注意事项.....	77
9.1.2.15 Atan2.....	77
9.1.2.15.1 API 形式.....	77
9.1.2.15.2 API 功能描述.....	78
9.1.2.15.3 Example.....	78
9.1.2.15.4 注意事项.....	78
9.2 数学常数宏定义.....	78
10 modulation 调制模块.....	80
10.1 空间矢量脉宽调制.....	80
10.1.1 SVPWM 控制器参数描述.....	80
10.1.1.1 空间矢量脉宽调制 handle 结构.....	80
10.1.1.2 空间矢量脉宽调制计算变量结构体.....	81
10.1.2 API 接口.....	81
10.1.2.1 SVPWM_Init.....	82
10.1.2.1.1 API 形式.....	82
10.1.2.1.2 API 功能描述.....	82
10.1.2.1.3 Example.....	82
10.1.2.1.4 注意事项.....	82
10.1.2.2 SVPWM_SectorCalc.....	82
10.1.2.2.1 API 形式.....	82
10.1.2.2.2 API 功能描述.....	83
10.1.2.2.3 Example.....	83
10.1.2.2.4 注意事项.....	83
10.1.2.3 SVPWM_CompareValCalc.....	83
10.1.2.3.1 API 形式.....	83
10.1.2.3.2 API 功能描述.....	83
10.1.2.3.3 Example.....	83
10.1.2.3.4 注意事项.....	83
10.1.2.4 SVPWM_IndexConvert.....	83
10.1.2.4.1 API 形式.....	84
10.1.2.4.2 API 功能描述.....	84
10.1.2.4.3 Example.....	84
10.1.2.4.4 注意事项.....	84
10.1.2.5 SVPWM_Exec.....	84
10.1.2.5.1 API 形式.....	84
10.1.2.5.2 API 功能描述.....	85
10.1.2.5.3 Example.....	85



10.1.2.5.4 注意事项.....	85
10.2 单电阻 SVPWM 调制和电流重构方法.....	85
10.2.1 单电阻空间矢量脉宽调制 handle 结构.....	85
10.2.2 API 接口.....	86
10.2.2.1 R1SVPWM_Init.....	86
10.2.2.1.1 API 形式.....	86
10.2.2.1.2 API 功能描述.....	86
10.2.2.1.3 Example.....	86
10.2.2.1.4 注意事项.....	86
10.2.2.2 R1SVPWM_Clear.....	87
10.2.2.2.1 API 形式.....	87
10.2.2.2.2 API 功能描述.....	87
10.2.2.2.3 Example.....	87
10.2.2.2.4 注意事项.....	87
10.2.2.3 R1SVPWM_Exec.....	87
10.2.2.3.1 API 形式.....	87
10.2.2.3.2 API 功能描述.....	88
10.2.2.3.3 Example.....	88
10.2.2.3.4 注意事项.....	88
10.2.2.4 R1SVPWM_PhaseShift.....	88
10.2.2.4.1 API 形式.....	88
10.2.2.4.2 API 功能描述.....	89
10.2.2.4.3 Example.....	89
10.2.2.4.4 注意事项.....	89
10.2.2.5 R1CurrReconstruct.....	89
10.2.2.5.1 API 形式.....	89
10.2.2.5.2 API 功能描述.....	89
10.2.2.5.3 Example.....	89
10.2.2.5.4 注意事项.....	90
11 observer 观测器模块.....	91
11.1 一阶滑模位置观测器.....	91
11.1.1 一阶滑模位置观测器 handle 结构体说明.....	91
11.1.2 API 接口.....	92
11.1.2.1 FOSMO_Init.....	93
11.1.2.1.1 API 形式.....	93
11.1.2.1.2 API 功能描述.....	93
11.1.2.1.3 Example.....	93
11.1.2.1.4 注意事项.....	93
11.1.2.2 FOSMO_Exec.....	93
11.1.2.2.1 API 形式.....	94
11.1.2.2.2 API 功能描述.....	94
11.1.2.2.3 Example.....	94



11.1.2.2.4 注意事项.....	94
11.1.2.3 FOSMO_ParamUpdate.....	94
11.1.2.3.1 API 形式.....	94
11.1.2.3.2 API 功能描述.....	95
11.1.2.3.3 Example.....	95
11.1.2.3.4 注意事项.....	95
11.1.2.4 FOSMO_Clear.....	95
11.1.2.4.1 API 形式.....	95
11.1.2.4.2 API 功能描述.....	95
11.1.2.4.3 Example.....	96
11.1.2.4.4 注意事项.....	96
11.1.2.5 FOSMO_SetTs.....	96
11.1.2.5.1 API 形式.....	96
11.1.2.5.2 API 功能描述.....	96
11.1.2.5.3 Example.....	96
11.1.2.5.4 注意事项.....	96
11.1.2.6 FOSMO_SetLambda.....	96
11.1.2.6.1 API 形式.....	97
11.1.2.6.2 API 功能描述.....	97
11.1.2.6.3 Example.....	97
11.1.2.6.4 注意事项.....	97
11.2 四阶滑模位置观测器.....	97
11.2.1 四阶滑模位置观测器 handle 结构体说明.....	97
11.2.2 API 接口.....	98
11.2.2.1 SMO4TH_Init.....	98
11.2.2.1.1 API 形式.....	99
11.2.2.1.2 API 功能描述.....	99
11.2.2.1.3 Example.....	99
11.2.2.1.4 注意事项.....	99
11.2.2.2 SMO4TH_Clear.....	99
11.2.2.2.1 API 形式.....	99
11.2.2.2.2 API 功能描述.....	100
11.2.2.2.3 Example.....	100
11.2.2.2.4 注意事项.....	100
11.2.2.3 SMO4TH_ParamUpdate.....	100
11.2.2.3.1 API 形式.....	100
11.2.2.3.2 API 功能描述.....	100
11.2.2.3.3 Example.....	100
11.2.2.3.4 注意事项.....	101
11.2.2.4 SMO4TH_Exec.....	101
11.2.2.4.1 API 形式.....	101
11.2.2.4.2 API 功能描述.....	101



11.2.2.4.3 Example.....	101
11.2.2.4.4 注意事项.....	101
11.2.2.5 SMO4TH_SetTs.....	101
11.2.2.5.1 API 形式.....	101
11.2.2.5.2 API 功能描述.....	102
11.2.2.5.3 Example.....	102
11.2.2.5.4 注意事项.....	102
12 pfc 功率因数校正模块.....	103
12.1 PFC 电流控制.....	103
12.1.1 PFC 电流控制器 handle 结构体说明.....	103
12.1.2 API 接口.....	104
12.1.2.1 PFC_CurrCtrlInit.....	104
12.1.2.1.1 API 形式.....	104
12.1.2.1.2 API 功能描述.....	104
12.1.2.1.3 Example.....	104
12.1.2.1.4 注意事项.....	105
12.1.2.2 PFC_CurrCtrlClear.....	105
12.1.2.2.1 API 形式.....	105
12.1.2.2.2 API 功能描述.....	105
12.1.2.2.3 Example.....	105
12.1.2.2.4 注意事项.....	105
12.1.2.3 PFC_CurrCtrlExec.....	105
12.1.2.3.1 API 形式.....	105
12.1.2.3.2 API 功能描述.....	106
12.1.2.3.3 Example.....	106
12.1.2.3.4 注意事项.....	106
12.2 PFC 电压控制.....	106
12.2.1 PFC 电压控制器 handle 结构体说明.....	106
12.2.2 API 接口.....	106
12.2.2.1 PFC_VoltCtrlInit.....	106
12.2.2.1.1 API 形式.....	107
12.2.2.1.2 API 功能描述.....	107
12.2.2.1.3 Example.....	107
12.2.2.1.4 注意事项.....	107
12.2.2.2 PFC_VoltCtrlExec.....	107
12.2.2.2.1 API 形式.....	107
12.2.2.2.2 API 功能描述.....	108
12.2.2.2.3 Example.....	108
12.2.2.2.4 注意事项.....	108
12.2.2.3 PFC_VoltCtrlClear.....	108
12.2.2.3.1 API 形式.....	108
12.2.2.3.2 API 功能描述.....	108



12.2.2.3.3 Example.....	108
12.2.2.3.4 注意事项.....	108
13 pid 控制器模块.....	109
13.1 PID 控制器.....	109
13.1.1 PID 控制器 handle 结构.....	109
13.1.2 API 接口.....	111
13.1.2.1 PID_Reset.....	111
13.1.2.1.1 API 形式.....	111
13.1.2.1.2 API 功能描述.....	111
13.1.2.1.3 Example.....	112
13.1.2.1.4 注意事项.....	112
13.1.2.2 PID_Clear.....	112
13.1.2.2.1 API 形式.....	112
13.1.2.2.2 API 功能描述.....	112
13.1.2.2.3 Example.....	112
13.1.2.2.4 注意事项.....	112
13.1.2.3 PI_Exec.....	112
13.1.2.3.1 API 形式.....	112
13.1.2.3.2 API 功能描述.....	113
13.1.2.3.3 Example.....	113
13.1.2.3.4 注意事项.....	113
13.1.2.4 PID_Exec.....	113
13.1.2.4.1 API 形式.....	113
13.1.2.4.2 API 功能描述.....	113
13.1.2.4.3 Example.....	113
13.1.2.4.4 注意事项.....	114
13.1.2.5 PID_ExecDiffWithFbk.....	114
13.1.2.5.1 API 形式.....	114
13.1.2.5.2 API 功能描述.....	114
13.1.2.5.3 Example.....	114
13.1.2.5.4 注意事项.....	114
13.1.2.6 PID_SetKp.....	115
13.1.2.6.1 API 形式.....	115
13.1.2.6.2 API 功能描述.....	115
13.1.2.6.3 Example.....	115
13.1.2.6.4 注意事项.....	115
13.1.2.7 PID_SetKi.....	115
13.1.2.7.1 API 形式.....	115
13.1.2.7.2 API 功能描述.....	116
13.1.2.7.3 Example.....	116
13.1.2.7.4 注意事项.....	116
13.1.2.8 PID_SetKd.....	116



13.1.2.8.1 API 形式.....	116
13.1.2.8.2 API 功能描述.....	116
13.1.2.8.3 Example.....	116
13.1.2.8.4 注意事项.....	116
13.1.2.9 PID_SetNs.....	116
13.1.2.9.1 API 形式.....	117
13.1.2.9.2 API 功能描述.....	117
13.1.2.9.3 Example.....	117
13.1.2.9.4 注意事项.....	117
13.1.2.10 PID_SetTs.....	117
13.1.2.10.1 API 形式.....	117
13.1.2.10.2 API 功能描述.....	117
13.1.2.10.3 Example.....	118
13.1.2.10.4 注意事项.....	118
13.1.2.11 PID_SetLimit.....	118
13.1.2.11.1 API 形式.....	118
13.1.2.11.2 API 功能描述.....	118
13.1.2.11.3 Example.....	118
13.1.2.11.4 注意事项.....	118
14 power 模块.....	119
14.1 功率管理.....	119
14.1.1 功率计算 handle 结构体说明.....	119
14.1.2 API 接口.....	119
14.1.2.1 MotorPowerInit.....	120
14.1.2.1.1 API 形式.....	120
14.1.2.1.2 API 功能描述.....	120
14.1.2.1.3 Example.....	120
14.1.2.1.4 注意事项.....	120
14.1.2.2 MotorPowerCalc.....	120
14.1.2.2.1 API 形式.....	120
14.1.2.2.2 API 功能描述.....	121
14.1.2.2.3 Example.....	121
14.1.2.2.4 注意事项.....	121
14.1.2.3 MotorPowerClear.....	121
14.1.2.3.1 API 形式.....	121
14.1.2.3.2 API 功能描述.....	121
14.1.2.3.3 Example.....	121
14.1.2.3.4 注意事项.....	121
15 protection 故障检测模块.....	122
15.1 开相检测.....	122
15.1.1 开相检测 handle 结构体说明.....	122
15.1.2 API 接口.....	123



15.1.2.1 OPD_Init..... 123

15.1.2.1.1 API 形式..... 123

15.1.2.1.2 API 功能描述..... 123

15.1.2.1.3 Example..... 124

15.1.2.1.4 注意事项..... 124

15.1.2.2 OPD_Exec..... 124

15.1.2.2.1 API 形式..... 124

15.1.2.2.2 API 功能描述..... 124

15.1.2.2.3 Example..... 124

15.1.2.2.4 注意事项..... 124

15.1.2.3 OPD_Clear..... 124

15.1.2.3.1 API 形式..... 124

15.1.2.3.2 API 功能描述..... 125

15.1.2.3.3 Example..... 125

15.1.2.3.4 注意事项..... 125

15.2 堵转检测..... 125

15.2.1 堵转检测与保护 handle 结构..... 125

15.2.2 API 接口..... 126

15.2.2.1 STD_Init..... 126

15.2.2.1.1 API 形式..... 126

15.2.2.1.2 API 功能描述..... 126

15.2.2.1.3 Example..... 126

15.2.2.1.4 注意事项..... 126

15.2.2.2 STD_Exec_ByCurrSpd..... 127

15.2.2.2.1 API 形式..... 127

15.2.2.2.2 API 功能描述..... 127

15.2.2.2.3 Example..... 127

15.2.2.2.4 注意事项..... 127

15.2.2.3 STD_Clear..... 127

15.2.2.3.1 API 形式..... 127

15.2.2.3.2 API 功能描述..... 128

15.2.2.3.3 Example..... 128

15.2.2.3.4 注意事项..... 128

15.3 电流不平衡检测..... 128

15.3.1 电流不平衡检测 handle 结构体说明..... 128

15.3.2 API 接口..... 129

15.3.2.1 UNBAL_Init..... 129

15.3.2.1.1 API 形式..... 129

15.3.2.1.2 API 功能描述..... 129

15.3.2.1.3 Example..... 129

15.3.2.1.4 注意事项..... 130

15.3.2.2 UNBAL_Det..... 130



15.3.2.2.1 API 形式.....	130
15.3.2.2.2 API 功能描述.....	130
15.3.2.2.3 Example.....	130
15.3.2.2.4 注意事项.....	130
15.3.2.3 UNBAL_Clear.....	130
15.3.2.3.1 API 形式.....	130
15.3.2.3.2 API 功能描述.....	131
15.3.2.3.3 Example.....	131
15.3.2.3.4 注意事项.....	131
16 ramp 斜坡管理模块.....	132
16.1 斜坡管理控制器.....	132
16.1.1 斜坡管理控制器 handle 结构体说明.....	132
16.1.2 API 接口.....	132
16.1.2.1 RMG_Init.....	133
16.1.2.1.1 API 形式.....	133
16.1.2.1.2 API 功能描述.....	133
16.1.2.1.3 Example.....	133
16.1.2.1.4 注意事项.....	133
16.1.2.2 RMG_Clear.....	133
16.1.2.2.1 API 形式.....	134
16.1.2.2.2 API 功能描述.....	134
16.1.2.2.3 Example.....	134
16.1.2.2.4 注意事项.....	134
16.1.2.3 RMG_Exec.....	134
16.1.2.3.1 API 形式.....	134
16.1.2.3.2 API 功能描述.....	134
16.1.2.3.3 Example.....	135
16.1.2.3.4 注意事项.....	135
16.1.2.4 RMG_SetTs.....	135
16.1.2.4.1 API 形式.....	135
16.1.2.4.2 API 功能描述.....	135
16.1.2.4.3 Example.....	135
16.1.2.4.4 注意事项.....	135
16.1.2.5 RMG_SetSlope.....	136
16.1.2.5.1 API 形式.....	136
16.1.2.5.2 API 功能描述.....	136
16.1.2.5.3 Example.....	136
16.1.2.5.4 注意事项.....	136
17 vf 控制模块.....	137
17.1 恒压频比控制器.....	137
17.1.1 vf 控制器 handle 结构体说明.....	137
17.1.2 API 接口.....	138



17.1.2.1 VF_Init.....	138
17.1.2.1.1 API 形式.....	138
17.1.2.1.2 API 功能描述.....	138
17.1.2.1.3 Example.....	139
17.1.2.1.4 注意事项.....	139
17.1.2.2 VF_Exec.....	139
17.1.2.2.1 API 形式.....	139
17.1.2.2.2 API 功能描述.....	139
17.1.2.2.3 Example.....	140
17.1.2.2.4 注意事项.....	140
17.1.2.3 VF_Clear.....	140
17.1.2.3.1 API 形式.....	140
17.1.2.3.2 API 功能描述.....	141
17.1.2.3.3 Example.....	141
17.1.2.3.4 注意事项.....	141
17.1.2.4 VF_SetTs.....	141
17.1.2.4.1 API 形式.....	141
17.1.2.4.2 API 功能描述.....	141
17.1.2.4.3 Example.....	141
17.1.2.4.4 注意事项.....	141
17.1.2.5 VF_SetSpdSlope.....	141
17.1.2.5.1 API 形式.....	142
17.1.2.5.2 API 功能描述.....	142
17.1.2.5.3 Example.....	142
17.1.2.5.4 注意事项.....	142
17.1.2.6 VF_SetDRatio.....	142
17.1.2.6.1 API 形式.....	142
17.1.2.6.2 API 功能描述.....	142
17.1.2.6.3 Example.....	143
17.1.2.6.4 注意事项.....	143
A 缩略语.....	144



插图目录

图 1-1 FOC 控制框图..... 1

图 6-1 主动刹车示意图..... 20

图 9-1 饱和函数示意图..... 76

图 10-1 UVW 三相比较值与 PWM 波形示意图..... 80

图 17-1 API 功能描述图..... 140



表格目录

表 2-1 通用数据类型定义文件目录说明..... 2

表 2-2 控制模块文件目录说明..... 2

表 4-1 三相静止(UVW)坐标系结构体说明..... 5

表 4-2 两相静止($\alpha\beta$)坐标系结构体说明..... 6

表 4-3 两相旋转(dq)坐标系结构体说明..... 6

表 4-4 电机参数结构体 MOTOR_Param 说明..... 6

表 4-5 电机参数初始化 API..... 7

表 4-6 获取系统启动状态 API..... 8

表 4-7 设置系统启动状态 API..... 9

表 4-8 清除系统启动状态 API..... 9

表 4-9 获取系统停止状态 API..... 10

表 4-10 设置系统停止状态 API..... 11

表 4-11 清除系统停止状态 API..... 11

表 4-12 获取系统运行状态 API..... 12

表 4-13 设置系统运行状态 API..... 12

表 4-14 清除系统运行状态 API..... 13

表 4-15 获取系统错误状态 API..... 13

表 4-16 设置系统错误状态 API..... 14

表 4-17 清除系统错误状态 API..... 15

表 5-1 ADC_CALIBR_State 校准状态枚举定义..... 16

表 5-2 ADC_CALIBR_Handle 校准结构体说明..... 16

表 5-3 接口列表..... 17

表 5-4 ADC 校准初始化 API..... 17

表 5-5 设置系统启动状态 API..... 18

表 5-6 清除系统启动状态 API..... 18

表 6-1 电机主动刹车参数结构体 BRAKE_Param 说明..... 21

表 6-2 BRAKE_Handle 校准结构体定义..... 21

表 6-3 BRAKE_Status 刹车状态枚举定义..... 22

表 6-4 BRAKE_Stage 刹车阶段枚举定义..... 22

表 6-5 接口列表..... 22

表 6-6 刹车制动初始化 API..... 22

表 6-7 刹车制动清除历史值 API..... 23

表 6-8 刹车制动执行 API..... 24



表 7-1 FOFLT_Handle 结构体说明..... 25

表 7-2 接口列表..... 25

表 7-3 一阶低通滤波器初始化 API..... 26

表 7-4 一阶滤波器清除历史值 API..... 27

表 7-5 执行一阶低通滤波器计算 API..... 27

表 7-6 更新低通滤波器的控制周期 API..... 28

表 7-7 LPF_RK4_Handle 结构体定义..... 29

表 7-8 接口列表..... 29

表 7-9 清除滤波器历史值 API..... 29

表 7-10 执行一阶低通滤波 API..... 30

表 7-11 PLL_Handle 结构体说明..... 30

表 7-12 接口列表..... 31

表 7-13 初始化锁相环 API..... 31

表 7-14 重置锁相环 API..... 32

表 7-15 清除锁相环 PID 历史值 API..... 32

表 7-16 执行锁相环计算功能 API..... 33

表 7-17 设置 PLL 的控制周期 API..... 34

表 7-18 设置 PLL 的带宽 API..... 34

表 8-1 CURRCTRL_Handle 成员变量..... 36

表 8-2 接口列表..... 37

表 8-3 电流控制器初始化 API..... 37

表 8-4 电流控制器变量复位 API..... 38

表 8-5 电流控制器历史值清除 API..... 38

表 8-6 电流控制器执行方法 API..... 39

表 8-7 设置电流控制器的控制周期 API..... 40

表 8-8 接口列表..... 41

表 8-9 电流环前馈控制计算 API..... 41

表 8-10 FW_Handle 结构体说明..... 42

表 8-11 接口列表..... 42

表 8-12 初始化弱磁控制器 API..... 43

表 8-13 弱磁控制执行 API..... 43

表 8-14 设置弱磁控制器控制周期 API..... 44

表 8-15 弱磁控制器清除历史值 API..... 45

表 8-16 IF_Handle 成员变量..... 46

表 8-17 接口列表..... 46

表 8-18 IF 控制器变量初始化 API..... 46

表 8-19 IF 控制器历史值清除 API..... 47

表 8-20 IF 控制器电流矢量幅值计算方法 API..... 48

表 8-21 IF 控制器电流矢量相角计算方法 API..... 48

表 8-22 IF 控制器更新角度控制周期 API..... 49

表 8-23 POSCTRL_Handle 成员变量..... 50

表 8-24 接口列表..... 51



表 8-25 位置控制器历史值清除 API.....	51
表 8-26 位置控制器初始化 API.....	52
表 8-27 位置控制器 PID 执行 API.....	53
表 8-28 位置控制器模式选择 API.....	53
表 8-29 设置位置控制器的斜坡管理模块的斜率 API.....	54
表 8-30 位置控制器设置目标位置 API.....	55
表 8-31 位置控制器执行方法 API.....	56
表 8-32 位置控制器的绝对位置计算 API.....	56
表 8-33 位置控制器设置 kp 参数 API.....	57
表 8-34 位置控制器设置 ki 参数 API.....	58
表 8-35 位置控制器设置 kd 参数 API.....	58
表 8-36 位置控制器设置 PID 微分环节低通滤波的截止频率 API.....	59
表 8-37 SPDCTRL_Handle 成员变量.....	60
表 8-38 接口列表.....	60
表 8-39 速度控制器变量初始化 API.....	60
表 8-40 速度控制器变量复位 API.....	61
表 8-41 速度环 PI 控制器历史值清除 API.....	62
表 8-42 速度环控制器执行方法 API.....	62
表 8-43 STARTUP_Stage 枚举类型.....	63
表 8-44 STARTUP_Handle 成员变量.....	63
表 8-45 接口列表.....	64
表 8-46 切换过程控制器变量初始化 API.....	64
表 8-47 切换过程控制器历史值清除 API.....	65
表 8-48 切换过程控制器电流指令值计算方法 API.....	65
表 9-1 TrigVal 三角函数值结构体说明.....	67
表 9-2 接口列表.....	67
表 9-3 三角函数 sin 计算 API.....	68
表 9-4 三角函数计算 API.....	69
表 9-5 三角函数计算 API.....	69
表 9-6 Park 变换 API.....	70
表 9-7 逆 Park 变换 API.....	71
表 9-8 Clarke 变换 API.....	71
表 9-9 取绝对值 API.....	72
表 9-10 限幅运算 API.....	73
表 9-11 两个数中取较大值 API.....	73
表 9-12 两个数中取较小值 API.....	74
表 9-13 开方运算 API.....	75
表 9-14 角度相减运算 API.....	75
表 9-15 饱和函数 API.....	76
表 9-16 取模运算 API.....	77
表 9-17 带符号的反正切运算 API.....	77
表 9-18 数学常数宏定义.....	78



表 10-1 SVPWM_Handle 结构体说明..... 80

表 10-2 SVPWM_CALC_Handle 结构体说明..... 81

表 10-3 接口列表..... 81

表 10-4 SVPWM_Handle 初始化 API..... 82

表 10-5 电压矢量扇区和作用时间计算 API..... 82

表 10-6 电压矢量扇区和作用时间计算 API..... 83

表 10-7 基于扇区变换的三相占空比数据索引 API..... 84

表 10-8 SVPWM 执行 API..... 84

表 10-9 R1SVPWM_Handle 成员变量..... 85

表 10-10 接口列表..... 86

表 10-11 R1SVPWM_Handle 初始化 API..... 86

表 10-12 R1SVPWM_Handle 历史值清除 API..... 87

表 10-13 单电阻空间矢量脉宽调制执行 API..... 87

表 10-14 单电阻非观测区移相控制 API..... 88

表 10-15 单电阻电流重构方法 API..... 89

表 11-1 FOSMO_Handle 成员变量..... 91

表 11-2 FOSMO_Param 一阶滑模用户配置参数..... 92

表 11-3 接口列表..... 92

表 11-4 一阶滑模位置观测器初始化 API..... 93

表 11-5 滑模位置观测器执行方法 API..... 94

表 11-6 滑模位置观测器参数更新 API..... 94

表 11-7 清除滑模位置观测器变量历史值 API..... 95

表 11-8 设置滑模位置观测器控制周期 API..... 96

表 11-9 设置滑模位置观测器滤波系数 API..... 97

表 11-10 SMO4TH_Handle 四阶滑模位置观测器结构体说明..... 97

表 11-11 SMO4TH_Param 四阶滑模用户配置参数..... 98

表 11-12 接口列表..... 98

表 11-13 四阶滑模位置观测器初始化 API..... 99

表 11-14 四阶滑模位置观测器历史值清除 API..... 99

表 11-15 四阶滑模位置观测器参数更新 API..... 100

表 11-16 四阶滑模位置观测器执行方法 API..... 101

表 11-17 设置四阶滑模位置观测器控制周期 API..... 101

表 12-1 PFC_CURRCTRL_Handle 成员变量..... 103

表 12-2 接口列表..... 104

表 12-3 PFC 电流环初始化 API..... 104

表 12-4 清除 PFC 电流环历史值 API..... 105

表 12-5 PFC 执行电流环控制 API..... 105

表 12-6 PFC_VOLTCTRL_Handle 成员变量..... 106

表 12-7 接口列表..... 106

表 12-8 PFC 电压环初始化 API..... 107

表 12-9 执行 PFC 电压环控制 API..... 107

表 12-10 PFC 清除电压环历史值 API..... 108



表 13-1 PID_Handle 结构体说明.....	109
表 13-2 PI 控制器用户参数 PI_Param.....	110
表 13-3 PID 控制器用户参数 PID_Param.....	110
表 13-4 接口列表.....	111
表 13-5 PID 控制器重置 API.....	111
表 13-6 清除 PID 控制器历史值 API.....	112
表 13-7 执行 PID 控制器 PI 计算功能 API.....	112
表 13-8 执行 PID 控制器 PID 计算功能 API.....	113
表 13-9 执行 PID 控制器 PID 计算功能 API.....	114
表 13-10 PID 比例增益系数设置 API.....	115
表 13-11 PID 积分增益系数设置 API.....	115
表 13-12 PID 微分增益系数设置 API.....	116
表 13-13 PID 差分过滤系数设置 API.....	117
表 13-14 PID 控制周期设置 API.....	117
表 13-15 PID 上、下限幅值设置 API.....	118
表 14-1 POWER_Handle 结构体说明.....	119
表 14-2 接口列表.....	119
表 14-3 功率计算初始化 API.....	120
表 14-4 平均功率计算 API.....	120
表 14-5 平均功率计算清除历史值 API.....	121
表 15-1 OPD_Handle 结构体说明.....	122
表 15-2 开相检测的电流方向 OPD_Index 枚举定义.....	122
表 15-3 接口列表.....	123
表 15-4 开相检测初始化 API.....	123
表 15-5 开相检测执行 API.....	124
表 15-6 清除开相检测历史值 API.....	124
表 15-7 STD_Handle 成员变量.....	125
表 15-8 接口列表.....	126
表 15-9 电机堵转故障检测初始化 API.....	126
表 15-10 堵转检测 API.....	127
表 15-11 清除堵转故障历史值 API.....	127
表 15-12 UNBAL_Handle 结构体说明.....	128
表 15-13 接口列表.....	129
表 15-14 电流不平衡检测初始化 API.....	129
表 15-15 电流不平衡故障检测 API.....	130
表 15-16 清除电流不平衡检测的历史值 API.....	130
表 16-1 RMG_Handle 结构体说明.....	132
表 16-2 接口列表.....	132
表 16-3 初始化斜坡管理控制器 API.....	133
表 16-4 清零斜坡管理控制器历史值 API.....	134
表 16-5 执行斜坡管理控制器计算功能 API.....	134
表 16-6 设置斜坡控制器的控制周期 API.....	135



表 16-7 设置斜坡管理控制器斜率 API..... 136

表 17-1 VF_Handle 结构体说明..... 137

表 17-2 接口列表..... 138

表 17-3 初始化 vf 控制器 API..... 138

表 17-4 执行 vf 控制计算 API..... 139

表 17-5 清除 vf 控制历史值 API..... 140

表 17-6 设置 vf 控制器的控制周期 API..... 141

表 17-7 设置 vf 控制器的速度斜率 API..... 142

表 17-8 设置 vf 控制器 d 轴电压比例 API..... 142

表 A-1 缩略语..... 144

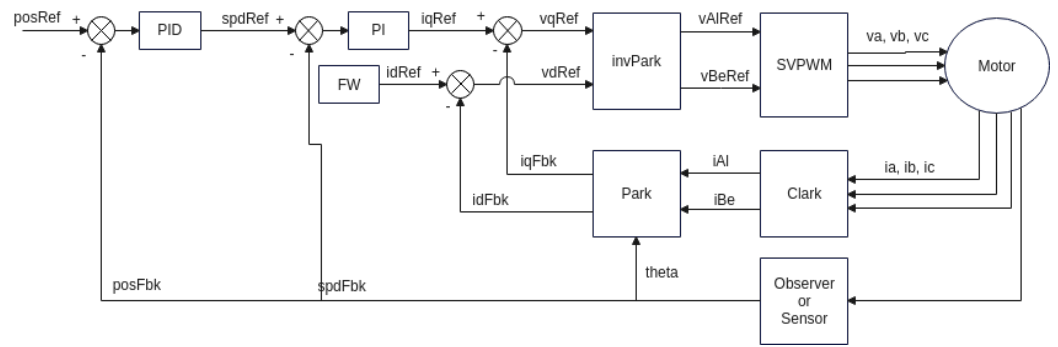


1 概述

SolarA²基础算法库是针对具有浮点运算单元（FPU）的MCU开发的浮点版本电机控制基础算法库，旨在帮助用户开发电机控制产品，减少开发工作量，提高开发效率。算法库包括：

- 电机控制使用的数据结构和数学常数定义，包括电机控制算法需要使用的不同坐标系中数据结构的定义、电机参数数据结构的定义等。
- 电机控制算法模块，包括例如低通滤波器、PID控制器等通用控制模块以及例如电流控制器、位置观测器等电机专用控制模块。
- 以FOC控制为例，整体框图和使用的算法模块如[图1-1](#)所示。

图 1-1 FOC 控制框图





2 文件目录说明

2.1 通用数据类型定义

为了方便数据管理和在函数间高效传递数据，将电机控制算法所使用的数据结构在[表 2-1](#)所示的文件中进行定义，控制模块和用户源文件通过包含对应头文件来使用类型定义。文件子目录如下：

..\middleware\control_library\utilities

表 2-1 通用数据类型定义文件目录说明

文件目录	文件说明
mcs_typedef.h	定义不同坐标系下的控制变量的数据结构。
mcs_mtr_param.h/.c	定义被控电机属性的数据结构。
mcs_sys_status.h	电机系统状态的数据结构定义和API实现。
mcs_assert.h	断言的定义。

2.2 控制模块

控制模块按照功能类型进行划分，文件目录如[表 2-2](#)所示。h文件中定义控制模块结构体，并提供对外开放的接口API，用户源文件通过包含对应控制模块的头文件来使用控制方法，c文件中为对应控制方法的实现。文件子目录如下：

..\middleware\control_library\

表 2-2 控制模块文件目录说明

模块名	模块目录	模块说明
ADC校准模块	control_library/ adc_calibra	用于ADC偏置校准。



模块名	模块目录	模块说明
刹车控制模块	control_library/ brake	电机主动刹车，使速度快速降为0。
滤波器	control_library/ filter	使信号中特定的频率成分通过，而极大地衰减其他频率成分。
FOC控制模块	control_library/ foc_loop_ctrl	包含电流环控制、电流前馈控制、弱磁控制、if控制、速度环控制和启动切换控制。
通用数学函数和常数宏模块	control_library/ math	包含三角函数、开方、取模和常用数学宏等常用的数学计算功能。
调制模块	control_library/ modulation	包含单电阻、双电阻和三电阻SVPWM调制功能。
观测器模块	control_library/ observer	包含一阶滑模观测器和四阶滑模观测器，提供电机的速度和角度信息。
PFC控制模块	control_library/pfc	包含PFC电压控制和电流控制。
PID控制器模块	control_library/ pid_controller	结合比例、积分和微分三种环节于一体的控制算法。
功率计算模块	control_library/ power	计算电机平均功率的算法。
故障检测模块	control_library/ protection	包含堵转、开相和电流不平衡的检测算法。
斜坡管理模块	control_library/ ramp	生成斜坡指令，例如速度斜坡增长。
通用数据类型定义模块	control_library/ utilities	包含电机参数、电控系统状态和参考坐标系的通用数据类型定义。
v/f控制模块	control_library/vf	提供恒压频比控制算法。



3 物理量单位说明

3.1 角度

电机转子角度的取值范围为 $-180^{\circ}\sim+180^{\circ}$ 。在电机控制程序中为了直观体现三角函数的运算，使用浮点变量进行表示，即用 $[-3.1415926f, 3.1415926f]$ 弧度对应真实的电机转子角度 $[-180^{\circ}, +180^{\circ}]$ ，换算关系为：角度 = (弧度 / π) * 180。

3.2 速度

电机速度分为电角速度和机械角速度，在本算法库中电机速度统一指电角速度，单位为Hz。



4 utilities 通用模块

4.1 参数校验

参数校验默认开启。参数检查作为代码安全的一道重要防线，对不合法的参数（如空指针、入参范围、返回值异常等），会让程序停止在第一个不合法的校验处，防止程序产生未定义行为；本文定义的参数检查类型共分为以下三类。

- 1. 函数入参检查，包含范围检查，条件检查，非空指针等，支持&，|，！，&&，||等逻辑运算符。
- 2. 函数无返回值校验。
- 3. 函数存在返回值校验。

说明

为减少中断资源占用，建议关闭参数校验。关闭方法为“芯片配置器/配置信息/MACRO，取消勾选MCS_PARAM_CHECK”。

建议：对中断资源要求较高的例程（如高速风机例程），建议关闭参数校验，提高程序执行效率。

4.2 电机控制坐标系结构定义

在电机控制领域内，通常会在3个基本坐标系（三相静止坐标系、两相静止坐标系和两相旋转坐标系）下进行数据运算，为了方便数据管理和函数间传递，在文件mcs_typedef.h中分别定义这三个坐标系的结构体，具体定义如下。

三相静止坐标系结构体 UvwAxis：

表 4-1 三相静止(UVW)坐标系结构体说明

成员类型	成员名	成员描述
float	u	U相分量
float	v	V相分量
float	w	W相分量



两相静止($\alpha\beta$)坐标系结构体 AlbeAxis:

表 4-2 两相静止($\alpha\beta$)坐标系结构体说明

成员类型	成员名	成员描述
float	alpha	alpha轴分量
float	beta	beta轴分量

两相旋转(dq)坐标系结构体 DqAxis:

表 4-3 两相旋转(dq)坐标系结构体说明

成员类型	成员名	成员描述
float	d	d轴分量
float	q	q轴分量

4.3 电机属性

4.3.1 电机属性 handle 结构体说明

在电机控制中需要使用被控电机的各种属性，为了方便数据管理和函数间传递，在 mcs_mtr_param.h 中定义电机参数结构体，具体定义如表 4-4 所示。

表 4-4 电机参数结构体 MOTOR_Param 说明

成员类型	成员名	成员描述
unsigned short	mtrNp	电机极对数。
float	mtrRs	定子电阻，单位： Ω 。
float	mtrLd	d轴电感，单位：H。
float	mtrLq	q轴电感，单位：H。
float	mtrLs	d、q轴电感平均值，单位：H。
float	mtrPsif	永磁体磁链，单位：Wb。
float	mtrJ	电机惯量，单位： $\text{kg}\cdot\text{m}^2$ 。
float	maxElecSpd	最大电角速度，单位：Hz。
float	maxCurr	最大电流，单位：A。



成员类型	成员名	成员描述
float	maxTrq	最大输出转矩，单位：N·m。
float	busVolt	母线电压，单位：V。
unsigned int	mtrPPMR	编码器每圈电角度对应的脉冲数。
unsigned int	zShift	编码器z脉冲对应电机零度的安装误差偏移。

4.3.2 API 接口

4.3.2.1 MtrParamInit

4.3.2.1.1 API 形式

表 4-5 电机参数初始化 API

标题	描述
接口形式	void MtrParamInit(MOTOR_Param *handle, const MOTOR_Param motorTable)
输入参数	handle：电机参数结构体指针。 motorTable：电机参数表。
返回参数	handle：MOTOR_Param指针类型，初始化电机参数结构体成员变量。
返回值	无。

4.3.2.1.2 API 功能描述

初始化电机参数，如极对数、相电阻、d轴电感、q轴电感等。

4.3.2.1.3 Example

```
/* 定义电机参数表 */
MOTOR_Param motorTable = {3, 2.5f, 12.5f, 15.0f, 13.75f, 0.112f, 0.04f, 500.0f, 4.0f, 3.2f, 24.0f, 0.0f, 0.0f};
MOTOR_Param paramHandle = {0};
MtrParamInit(&paramHandle, motorTable); /* 初始化电机参数 */
```

4.3.2.1.4 注意事项

无。



4.4 系统状态

4.4.1 系统状态 handle 结构体说明

在电机控制中需要控制系统的状态，在mcs_sys_status.h中定义系统状态结构体 SysStatusReg，具体定义如下。

```
typedef union {
    unsigned short all;
    struct {
        unsigned short cmdStart : 1; /**< Indicates that a start system command has been
received. */
        unsigned short cmdStop : 1; /**< Indicates that a stop system command has been
received. */
        unsigned short isRunning : 1; /**< Indicates that the system is running (enable signal) */
        unsigned short sysError : 1; /**< Indicates that the system reports an error. */
        unsigned short poweron : 1; /**< Indicates that the power-on initialization phase is
complete. */
        unsigned short capcharge : 1; /**< Indicates that the bootstrap capacitor charging phase
is complete. */
        unsigned short adczero : 1; /**< The current sampling point is reset to zero after power-
on. */
    } Bit;
} SysStatusReg;
```

4.4.2 API 接口

4.4.2.1 SysGetCmdStart

4.4.2.1.1 API 形式

表 4-6 获取系统启动状态 API

标题	描述
接口形式	static inline bool SysGetCmdStart(SysStatusReg *sysStatus)
输入参数	sysStatus：系统状态结构体指针。
返回参数	无。
返回值	系统的启动状态。

4.4.2.1.2 API 功能描述

通过读取系统状态结构体中的cmdStart位，获取系统启动状态。

0：电机未启动；

1：电机启动。



4.4.2.1.3 Example

```
SysStatusReg sysStatus = {0};
bool startState = SysGetCmdStart(&sysStatus);
```

4.4.2.1.4 注意事项

无。

4.4.2.2 SysCmdStartSet

4.4.2.2.1 API 形式

表 4-7 设置系统启动状态 API

标题	描述
接口形式	static inline void SysCmdStartSet(SysStatusReg *sysStatus)
输入参数	sysStatus：系统状态结构体指针。
返回参数	sysStatus：设置系统的启动状态，将cmdStart位设置为1。
返回值	无。

4.4.2.2.2 API 功能描述

设置系统状态结构体中的cmdStart位，将其设置为1，表示电机状态为启动状态。

4.4.2.2.3 Example

```
SysStatusReg sysStatus = {0};
SysCmdStartSet(&sysStatus);
```

4.4.2.2.4 注意事项

无。

4.4.2.3 SysCmdStartClr

4.4.2.3.1 API 形式

表 4-8 清除系统启动状态 API

标题	描述
接口形式	static inline void SysCmdStartClr(SysStatusReg *sysStatus)
输入参数	sysStatus：系统状态结构体指针。
返回参数	sysStatus：清除系统的启动状态，将cmdStart位设置为0。
返回值	无。



4.4.2.3.2 API 功能描述

清除系统状态结构体中的cmdStart位，将其设置为0，表示电机状态为不启动状态。

4.4.2.3.3 Example

```
ysStatusReg sysStatus = {0};
SysCmdStartClr(&sysStatus);
```

4.4.2.3.4 注意事项

无。

4.4.2.4 SysGetCmdStop

4.4.2.4.1 API 形式

表 4-9 获取系统停止状态 API

标题	描述
接口形式	static inline bool SysGetCmdStop(SysStatusReg *sysStatus)
输入参数	sysStatus：系统状态结构体指针。
返回参数	无。
返回值	系统的停止状态。

4.4.2.4.2 API 功能描述

通过读取系统状态结构体中的cmdStop位，获取系统停止状态。

0：没有停止；

1：停止。

4.4.2.4.3 Example

```
SysStatusReg sysStatus = {0};
bool stopState = SysGetCmdStop(&sysStatus);
```

4.4.2.4.4 注意事项

无。

4.4.2.5 SysCmdStopSet



4.4.2.5.1 API 形式

表 4-10 设置系统停止状态 API

标题	描述
接口形式	static inline void SysCmdStopSet(SysStatusReg *sysStatus)
输入参数	sysStatus：系统状态结构体指针。
返回参数	sysStatus：设置系统的停止状态，将cmdStop位设置为1。
返回值	无。

4.4.2.5.2 API 功能描述

设置系统状态结构体中的cmdStop位，将其设置为1，表示电机状态为停止状态。

4.4.2.5.3 Example

```
SysStatusReg sysStatus = {0};
SysCmdStopSet(&sysStatus);
```

4.4.2.5.4 注意事项

无。

4.4.2.6 SysCmdStopClr

4.4.2.6.1 API 形式

表 4-11 清除系统停止状态 API

标题	描述
接口形式	static inline void SysCmdStopClr(SysStatusReg *sysStatus)
输入参数	sysStatus：系统状态结构体指针。
返回参数	sysStatus：清除系统的停止状态，将cmdStop位设置为0。
返回值	无。

4.4.2.6.2 API 功能描述

清除系统状态结构体中的cmdStop位，将其设置为0，表示电机状态为不停止状态。

4.4.2.6.3 Example

```
SysStatusReg sysStatus = {0};
SysCmdStopClr(&sysStatus);
```



4.4.2.6.4 注意事项

无。

4.4.2.7 SysIsRunning

4.4.2.7.1 API 形式

表 4-12 获取系统运行状态 API

标题	描述
接口形式	static inline bool SysIsRunning(SysStatusReg *sysStatus)
输入参数	sysStatus：系统状态结构体指针。
返回参数	无。
返回值	系统的运行状态。

4.4.2.7.2 API 功能描述

通过读取系统状态结构体中的isRunning位，获取系统运行状态。

0：没有运行；

1：运行。

4.4.2.7.3 Example

```
SysStatusReg sysStatus = {0};
bool runState = SysIsRunning(&sysStatus);
```

4.4.2.7.4 注意事项

无。

4.4.2.8 SysRunningSet

4.4.2.8.1 API 形式

表 4-13 设置系统运行状态 API

标题	描述
接口形式	static inline void SysRunningSet(SysStatusReg *sysStatus)
输入参数	sysStatus：系统状态结构体指针。
返回参数	sysStatus：设置系统的运行状态，将isRunning位设置为1。
返回值	无。



4.4.2.8.2 API 功能描述

设置系统状态结构体中的isRunning位，将其设置为1，表示电机状态为运行状态。

4.4.2.8.3 Example

```
SysStatusReg sysStatus = {0};
SysRunningSet(&sysStatus);
```

4.4.2.8.4 注意事项

无。

4.4.2.9 SysRunningClr

4.4.2.9.1 API 形式

表 4-14 清除系统运行状态 API

标题	描述
接口形式	static inline void SysRunningClr(SysStatusReg *sysStatus)
输入参数	sysStatus：系统状态结构体指针。
返回参数	sysStatus：清除系统的运行状态，将isRunning位设置为0。
返回值	无。

4.4.2.9.2 API 功能描述

清除系统状态结构体中的isRunning位，将其设置为0，表示电机状态为不运行状态。

4.4.2.9.3 Example

```
SysStatusReg sysStatus = {0};
SysRunningClr(&sysStatus);
```

4.4.2.9.4 注意事项

无。

4.4.2.10 SysIsError

4.4.2.10.1 API 形式

表 4-15 获取系统错误状态 API

标题	描述
接口形式	static inline bool SysIsError(SysStatusReg *sysStatus)
输入参数	sysStatus：系统状态结构体指针。



标题	描述
返回参数	无。
返回值	系统的错误状态。

4.4.2.10.2 API 功能描述

通过读取系统状态结构体中的sysError位，获取系统错误状态。

- 0：没有错误；
- 1：错误。

4.4.2.10.3 Example

```
SysStatusReg sysStatus = {0};
bool errState = SysIsError(&sysStatus);
```

4.4.2.10.4 注意事项

无。

4.4.2.11 SysErrorSet

4.4.2.11.1 API 形式

表 4-16 设置系统错误状态 API

标题	描述
接口形式	static inline void SysErrorSet(SysStatusReg *sysStatus)
输入参数	sysStatus：系统状态结构体指针。
返回参数	sysStatus：设置系统的错误状态，将sysError位设置为1。
返回值	无。

4.4.2.11.2 API 功能描述

设置系统状态结构体中的sysError位，将其设置为1，表示电机状态为错误状态。

4.4.2.11.3 Example

```
SysStatusReg sysStatus = {0};
SysErrorSet(&sysStatus);
```

4.4.2.11.4 注意事项

无。



4.4.2.12 SysErrorClr

4.4.2.12.1 API 形式

表 4-17 清除系统错误状态 API

标题	描述
接口形式	static inline void SysErrorClr(SysStatusReg *sysStatus)
输入参数	sysStatus：系统状态结构体指针。
返回参数	sysStatus：清除系统的错误状态，将sysError位设置为0。
返回值	无。

4.4.2.12.2 API 功能描述

清除系统状态结构体中的sysError位，将其设置为0，表示电机状态不是错误状态。

4.4.2.12.3 Example

```
SysStatusReg sysStatus = {0};
SysErrorClr(&sysStatus);
```

4.4.2.12.4 注意事项

无。



5 adc_calibr ADC 校准模块

5.1 ADC 校准

ADC校准的目的是消除ADC采样偏差，其原理是在电机启动前，读取50次ADC的值，并对它们求平均，得到零电流下实际ADC的读数，用于后续电流计算。使用方法如下：

- 1. 初始化ADC校准API：ADCCALIBR_Init；
- 2. 在载波中断中执行ADC校准API：ADCCALIBR_Exec；
- 3. 在Timer中断中，执行ADC校准是否完成的API：ADCCALIBR_IsFinish，如果完成校准，记录此时ADC的校准值，并跳转至下一状态。

5.1.1 ADC 校准 handle 结构体说明

ADC校准结构体定义如下。

表 5-1 ADC_CALIBR_State 校准状态枚举定义

成员类型	成员名	成员描述
ENUM	ADC_CALIBR_NOT_FINISH	未完成ADC校准。
ENMU	ADC_CALIBR_FINISH	完成ADC校准。

表 5-2 ADC_CALIBR_Handle 校准结构体说明

成员类型	成员名	成员描述
unsigned int	adcShiftAccu	ADC读数的累计值。
unsigned int	cnt	ADC校准时采样次数。
ADC_CALIBR_State	state	ADC校准状态。



5.1.2 API 接口

表 5-3 接口列表

函数名称	功能描述
ADCCALIBR_Init	初始化ADC校准。
ADCCALIBR_Exec	执行ADC校准。
ADCCALIBR_IsFinish	询问ADC校准是否完成。

5.1.2.1 ADCCALIBR_Init

5.1.2.1.1 API 形式

表 5-4 ADC 校准初始化 API

标题	描述
接口形式	void ADCCALIBR_Init(ADC_CALIBR_Handle *adcCalibr)
输入参数	adcCalibr: ADC校准结构体指针。
返回参数	无。
返回值	无。

5.1.2.1.2 API 功能描述

ADC校准模块初始化。

5.1.2.1.3 Example

```
ADC_CALIBR_Handle adcCalibr;  
ADCCALIBR_Init(&adcCalibr);
```

5.1.2.1.4 注意事项

可以通过修改ADC_CNT_POINTS，修改ADC校准时的采集次数。

5.1.2.2 ADCCALIBR_Exec



5.1.2.2.1 API 形式

表 5-5 设置系统启动状态 API

标题	描述
接口形式	unsigned int ADCCALIBR_Exec(ADC_CALIBR_Handle *adcCalibr, ADC_Handle *adcHandle, unsigned int soc)
输入参数	adcCalibr: ADC校准结构体指针。 adcHandle: 需要被校准的ADC结构体指针。 soc: 数据存放的SOC号。
返回参数	校准的累计值和完成状态。
返回值	ADC校准值。

5.1.2.2.2 API 功能描述

通过累计采样50（默认）次ADC的读数，并取平均后得到ADC的校准读数，执行ADC校准功能。

5.1.2.2.3 Example

```
ADC_CALIBR_Handle adcCalibr;  
ADC_Handle adc1;  
unsigned int soc = ADC_SOC_NUM1;  
unsigned int adcVal = ADCCALIBR_Exec(&adcCalibr, &adc1, ADC_SOC_NUM1);
```

5.1.2.2.4 注意事项

只有在ADC校准完成后，才能获得正确的校准值，否则都返回0。

5.1.2.3 ADCCALIBR_IsFinish

5.1.2.3.1 API 形式

表 5-6 清除系统启动状态 API

标题	描述
接口形式	bool ADCCALIBR_IsFinish(ADC_CALIBR_Handle *adcCalibr)
输入参数	adcCalibr: ADC校准结构体指针。
返回参数	无。
返回值	是否完成校准。

5.1.2.3.2 API 功能描述

调用该API，判断ADC校准是否完成。



0: 未完成校准;

1: 完成校准。

5.1.2.3.3 Example

```
ADC_CALIBR_Handle adcCalibr;  
bool adcFinish = ADCCALIBR_IsFinish(&adcCalibr);
```

5.1.2.3.4 注意事项

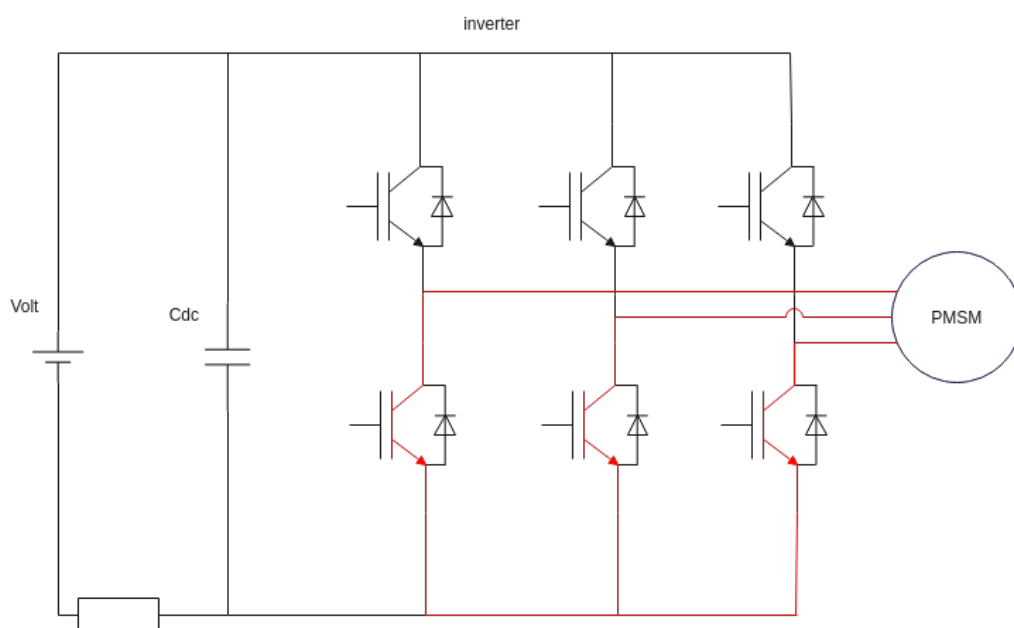
调用该函数，以此判断ADC校准是否完成，只有当ADC校准完成才能获得正确的校准值，否则校准值是0。

6 brake 刹车模块

6.1 刹车

本文的刹车功能采用主动短路（ASC，active short circuit）的形式，其原理是上三桥或下三桥全开（导通），使电机定子绕组形成闭合回路，是一种电机的安全保护机制，防止控制器系统产生损坏。如图6-1所示。

图 6-1 主动刹车示意图



电机转子旋转会产生反电势，转速越高反电势越大。反电势小于母线电压时，电流不会经过续流二极管流入电源，对系统无影响；反电势大于母线电压时，则电流反方向流动，由电机经续流二极管流入电源，发生电流倒灌，相当于对稳压电容Cdc充电，两端电压升高，若长时间充电，电压过高，有击穿稳压电容的风险。所以此时需控制逆变器进入ASC模式，即下（上）三桥导通，与电机U V W三项形成短路，通过电机定子绕组将产生的反电势耗散掉。



6.1.1 刹车 handle 结构体说明

在一些场合下，需要做到电机的快速制动，因此需要刹车制动功能，刹车结构体定义如下。

表 6-1 电机主动刹车参数结构体 BRAKE_Param 说明

成员类型	成员名	成员描述
float	brkTime	刹车时间，s。
float	maxBrkCurr	最大刹车电流，A。推荐值：额定电流。
float	blindDutyThr	进入开环刹车状态时的刹车占空比。
float	blindBrkDutyStep	最大刹车占空比增量。
float	fastBrkCurrCoeff	快速刹车的电流系数，当电流小于最大刹车电流*fastBrkCurrCoeff 时，开始快速刹车制动。推荐值：0.5f。
float	fastBrkDutyStep	快速刹车占空比增量。
float	slowBrkDutyStep	慢速刹车占空比增量。
unsigned short	aptMaxCmp	APT最大数量。

表 6-2 BRAKE_Handle 校准结构体定义

成员类型	成员名	成员描述
float	ts	刹车控制周期，s。
float	brkDuty	刹车占空比，0~1。
float	brkCurr	刹车电流，A。
unsigned int	tickCnt	刹车制动过程的计数值。
unsigned int	tickNum	完成刹车制动的总计数值，值 = brkTime / ts。
unsigned int	brkFinished	刹车完成标志，有0，1两种状态。
BRAKE_Stage	stage	刹车阶段，分为BRAKE_INIT、BRAKE_EXEC、BRAKE_FINISHE三个阶段。
BRAKE_Status	status	刹车状态分为BRAKE_CLOSED_LOOP、BRAKE_OPEN_LOOP两种状态。
BRAKE_Param *	brkParam	刹车参数。



表 6-3 BRAKE_Status 刹车状态枚举定义

成员类型	成员名	成员描述
enum	BRAKE_CLOSED_LOOP	闭环刹车。
enum	BRAKE_OPEN_LOOP	开环刹车。

表 6-4 BRAKE_Stage 刹车阶段枚举定义

成员类型	成员名	成员描述
enum	BRAKE_INIT	刹车初始化，包括APT配置。
enum	BRAKE_EXEC	刹车执行，包括刹车电流采样和刹车占空比调整。
enum	BRAKE_FINISHE	刹车完成。

6.1.2 API 接口

表 6-5 接口列表

函数名称	功能描述
BRAKE_Init	初始化刹车制动结构体。
BRAKE_Clear	清除刹车制动历史值。
BRAKE_Exec	执行刹车制动。

6.1.2.1 BRAKE_Init

6.1.2.1.1 API 形式

表 6-6 刹车制动初始化 API

标题	描述
接口形式	void BRAKE_Init(BRAKE_Handle *brake, BRAKE_Param *brkParam, float ts)
输入参数	brake：刹车制动结构体指针。 BRAKE_Param：刹车参数结构体指针。 ts：刹车周期。



标题	描述
返回参数	brake结构体中刹车相关参数。
返回值	无。

6.1.2.1.2 API 功能描述

刹车制动模块初始化，包括控制周期、刹车时间、最大刹车电流等。

6.1.2.1.3 Example

```
BRAKE_Param brkParam = {0.0004f, 0.5f, 0.00004f, 3.0f, 0.001f, 0.005f, 0.5f, 0.0001f, 0.2f};
float ts = 0.0001;
BRAKE_Handle brake = {0};
BRAKE_Init(&brake, &brkParam, ts);
```

6.1.2.1.4 注意事项

无。

6.1.2.2 BRAKE_Clear

6.1.2.2.1 API 形式

表 6-7 刹车制动清除历史值 API

标题	描述
接口形式	void BRAKE_Clear(BRAKE_Handle *brake)
输入参数	brake：刹车制动结构体指针。
返回参数	brake结构体中清除刹车过程中的历史值。
返回值	无。

6.1.2.2.2 API 功能描述

清除刹车历史值。

6.1.2.2.3 Example

```
BRAKE_Handle brake = {0};
...
BRAKE_Clear(&brake);
```

6.1.2.2.4 注意事项

无。



6.1.2.3 BRAKE_Exec

6.1.2.3.1 API 形式

表 6-8 刹车制动执行 API

标题	描述
接口形式	void BRAKE_Exec(BRAKE_Handle *brake, APT_RegStruct **aptAddr, float brkCurr)
输入参数	brake: 刹车制动结构体指针。 aptAddr: APT地址指针。 brkCurr: 刹车时的电流。
返回参数	brake结构体指针，刹车完成后，更新刹车状态。
返回值	无。

6.1.2.3.2 API 功能描述

刹车制动执行函数。

6.1.2.3.3 Example

```
BRAKE_Handle brake = {0};
float brkCurr = 0;
APT_RegStruct *aptAddr[3] = {APT0_BASE, APT1_BASE, APT2_BASE};
...
/* APT地址数组已初始化 */
BRAKE_Exec(&brake, aptAddr, brkCurr);
```

6.1.2.3.4 注意事项

无。



7 filter 滤波器模块

7.1 一阶低通滤波器

7.1.1 一阶滤波器 handle 结构

用于保存一阶滤波器参数及历史值的结构体，具体定义如表7-1所示。

表 7-1 FOFLT_Handle 结构体说明

成员类型	成员名	成员描述
float	yLast	一阶滤波器输出历史值。
float	uLast	一阶滤波器输入历史值。
float	fc	一阶滤波器截止频率，单位：Hz。
float	ts	一阶滤波器计算周期，单位：s。
float	a1	一阶滤波器内部计算中间系数。
float	b1	一阶滤波器内部计算中间系数。
float	b2	一阶滤波器内部计算中间系数。

7.1.2 API 接口

表 7-2 接口列表

函数名称	功能描述
FOLPF_Init	初始化一阶低通滤波器。
FOLPF_Clear	清零一阶滤波器历史值。



函数名称	功能描述
FOLPF_Exec	进行一阶低通滤波器计算。
FOLPF_SetTs	设置一阶低通滤波器的控制周期。

7.1.2.1 FOLPF_Init

7.1.2.1.1 API 形式

表 7-3 一阶低通滤波器初始化 API

标题	描述
接口形式	void FOLPF_Init(FOFLT_Handle *lpfHandle, float ts, float fc)
输入参数	lpfHandle, FOFLT_Handle类型指针, 指向一阶滤波器结构体变量。 ts: 一阶滤波器计算周期, 单位: s; fc: 一阶滤波器截止频率, 单位: Hz。
返回参数	一阶滤波器采样时间和截止频率。
返回值	无。

7.1.2.1.2 API 功能描述

初始化一阶低通滤波器结构体变量, 计算内部滤波器系数: a1和b1, 计算方法如下:

$$a_1 = \frac{1}{1 + 2 * \pi * fc * ctrlPeriod}$$
$$b_1 = 1 - a_1$$

7.1.2.1.3 Example

```
FOFLT_Handle lpf = {0}; /* 定义一阶滤波器变量 */
float ts = 0.0001f; /* 定义一阶滤波器计算周期 */
float fc = 100.0f; /* 定义一阶滤波器截止频率 */
FOLPF_Init(&lpf, ts, fc); /* 初始化一阶滤波器结构体变量 */
```

7.1.2.1.4 注意事项

ts和fc均为正浮点数。



7.1.2.2 FOLPF_Clear

7.1.2.2.1 API 形式

表 7-4 一阶滤波器清除历史值 API

标题	描述
接口形式	void FOLPF_Clear(FOFLT_Handle *lpfHandle)
输入参数	lpfHandle: FOFLT_Handle 类型指针，指向一阶滤波器结构体变量。
返回参数	无。
返回值	无。

7.1.2.2.2 API 功能描述

清除一阶滤波器结构体具有历史信息的变量。

7.1.2.2.3 Example

```
FOFLT_Handle lpf = {0}; /* 定义一阶滤波器变量 */
FOLPF_Clear(&lpf); /* 清除一阶滤波器结构体变量内部历史值 */
```

7.1.2.2.4 注意事项

无。

7.1.2.3 FOLPF_Exec

7.1.2.3.1 API 形式

表 7-5 执行一阶低通滤波器计算 API

标题	描述
接口形式	float FOLPF_Exec(FOFLT_Handle *lpfHandle, float u)
输入参数	lpfHandle: FOFLT_Handle 类型指针，指向一阶滤波器结构体变量。 u: 待滤波的输入值。
返回参数	无。
返回值	对输入信号进行一阶低通滤波后的结果。

7.1.2.3.2 API 功能描述

执行一阶低通滤波计算，返回滤波结果。



7.1.2.3.3 Example

```
FOFLT_Handle lpf = {0}; /* 定义一阶低通滤波器变量 */
float ts = 0.0001f; /* 定义一阶低通滤波器计算周期 */
float fc = 100.0f; /* 定义一阶低通滤波器截止频率 */
FOLPF_Init(&lpf, ts, fc); /* 初始化一阶低通滤波器结构体变量 */
FOLPF_Clear(&lpf); /* 清除一阶低通滤波器历史值 */
float u; /* 待滤波信号变量 */
float ret; /* 滤波结果变量 */
ret = FOLPF_Exec(&lpf, u); /* 执行低通滤波计算，结果返回到ret中 */
```

7.1.2.3.4 注意事项

无。

7.1.2.4 FOLPF_SetTs

7.1.2.4.1 API 形式

表 7-6 更新低通滤波器的控制周期 API

标题	描述
接口形式	void FOLPF_SetTs(FOFLT_Handle *lpfHandle, float ts)
输入参数	lpfHandle: FOFLT_Handle 类型指针，指向一阶滤波器结构体变量。 ts: 控制周期。
返回参数	控制周期。
返回值	无。

7.1.2.4.2 API 功能描述

更新低通滤波器的控制周期。

7.1.2.4.3 Example

```
FOFLT_Handle lpf = {0};
float ts = 0.0002f;
FOLPF_SetTs(&lpf, ts);
```

7.1.2.4.4 注意事项

无。

7.2 四阶龙格-库塔离散化一阶低通滤波器

7.2.1 四阶龙格-库塔低通滤波器 handle 结构体说明

用于保存四阶龙格-库塔离散法的一阶滤波器参数及历史值的结构体，具体定义如[表 7-7](#)所示。



表 7-7 LPF_RK4_Handle 结构体定义

成员类型	成员名	成员描述
float	y1	基于四阶龙格-库塔离散方法的一阶滤波器的滤波值。

7.2.2 API 接口

表 7-8 接口列表

函数名称	功能描述
LPFRK4_Clear	清除历史值。
LPFRK4_Exec	进行基于RK4离散法的一阶低通滤波器计算。

7.2.2.1 LPFRK4_Clear

7.2.2.1.1 API 形式

表 7-9 清除滤波器历史值 API

标题	描述
接口形式	void LPFRK4_Clear(LPFRK4_Handle *lpf)
输入参数	lpf: LPFRK4_Handle结构体指针。
返回参数	无。
返回值	无。

7.2.2.1.2 API 功能描述

清除一阶滤波器结构体具有历史信息的变量。

7.2.2.1.3 Example

```
LPFRK4_Handle lpf = {0};
LPFRK4_Clear(&lpf);
```

7.2.2.1.4 注意事项

无。

7.2.2.2 LPFRK4_Exec



7.2.2.2.1 API 形式

表 7-10 执行一阶低通滤波 API

标题	描述
接口形式	float LPFRK4_Exec(LPF_RK4_Handle *lpf, float u, float freq, float ts)
输入参数	lpf: LPF_RK4_Handle结构体指针。 u: 被滤波的数据。 freq: 截止频率, Hz。 ts: 滤波控制周期, s。
返回参数	无。
返回值	滤波后的结果。

7.2.2.2.2 API 功能描述

基于四阶龙格-库塔离散化方法，对一阶滤波器离散化，并返回滤波后的结果。

7.2.2.2.3 Example

```
LPF_RK4_Handle lpf = {0};
float freq = 100.0f;
float ts = 0.0001f;
float u;
float ret;
ret = LPFRK4_Exec(&lpf, u, freq, ts);
```

7.2.2.2.4 注意事项

无。

7.3 PLL 锁相环

7.3.1 锁相环 handle 结构

用于保存锁相环参数及历史值的结构体，具体定义如[表7-11](#)所示。

表 7-11 PLL_Handle 结构体说明

成员类型	成员名	成员描述
PID_Handle	pi	锁相环内部PI控制器。
float	minAmp	输入信号归一化最小幅值。
float	ts	计算周期，单位：s。



成员类型	成员名	成员描述
float	ratio	频率积分折算系数。
float	freq	锁相环提取的频率，单位：Hz。
float	angle	锁相环提取的角度值，单位：rad。
float	pllBdw	锁相环的带宽，单位：Hz。

7.3.2 API 接口

表 7-12 接口列表

函数名称	功能描述
PLL_Init	初始化锁相环。
PLL_Reset	锁相环重置。
PLL_Clear	锁相环清零历史值。
PLL_Exec	执行锁相环计算。
PLL_SetTs	设置锁相环控制周期。
PLL_ParamUpdate	更新锁相环的带宽。

7.3.2.1 PLL_Init

7.3.2.1.1 API 形式

表 7-13 初始化锁相环 API

标题	描述
接口形式	void PLL_Init(PLL_Handle *pllHandle, float ts, float bdw)
输入参数	pllHandle: PLL_Handle类型指针，指向锁相环结构体变量。 ts: 控制周期，单位：s。 bdw: PLL带宽，单位：Hz。
返回参数	初始化相关参数。
返回值	无。

7.3.2.1.2 API 功能描述

初始化锁相环参数。



7.3.2.1.3 Example

```
PLL_Handle pll = {0}; /* 定义锁相环结构体变量 */
float ts = 0.0001f;
float bdw = 40.0f
PLL_Init(&pll, ts, bdw); /* 锁相环结构体变量重置 */
```

7.3.2.1.4 注意事项

无。

7.3.2.2 PLL_Reset

7.3.2.2.1 API 形式

表 7-14 重置锁相环 API

标题	描述
接口形式	void PLL_Reset(PLL_Handle *pllHandle)
输入参数	pllHandle: PLL_Handle类型指针，指向锁相环结构体变量。
返回参数	无。
返回值	无。

7.3.2.2.2 API 功能描述

清零锁相环内部变量历史值。

7.3.2.2.3 Example

```
PLL_Handle pll = {0}; /* 定义锁相环结构体变量 */
PLL_Reset(&pll); /* 锁相环结构体变量重置 */
```

7.3.2.2.4 注意事项

无。

7.3.2.3 PLL_Clear

7.3.2.3.1 API 形式

表 7-15 清除锁相环 PID 历史值 API

标题	描述
接口形式	void PLL_Clear(PLL_Handle *pllHandle)
输入参数	pllHandle: PLL_Handle类型指针，指向锁相环结构体变量。
返回参数	无。



标题	描述
返回值	无。

7.3.2.3.2 API 功能描述

清零锁相环内部PID控制器历史值。

7.3.2.3.3 Example

```
PLL_Handle pll = {0}; /* 定义锁相环结构体变量 */
PLL_Clear(&pll); /* 锁相环结构体变量清零历史值 */
```

7.3.2.3.4 注意事项

无。

7.3.2.4 PLL_Exec

7.3.2.4.1 API 形式

表 7-16 执行锁相环计算功能 API

标题	描述
接口形式	void PLL_Exec(PLL_Handle *pllHandle, float sinVal, float cosVal)
输入参数	pllHandle: PLL_Handle类型指针，指向锁相环结构体变量。 sinVal: 输入的正弦信号值。 cosVal: 输入的余弦信号值。
返回参数	无。
返回值	无。

7.3.2.4.2 API 功能描述

执行锁相环计算功能，提取输入的正交正弦信号的相位和频率信息，储存于锁相环结构体变量中。

7.3.2.4.3 Example

```
PLL_Handle pll = {0}; /* 定义锁相环结构体变量 */
float ts = 0.0001f;
float bdw = 100.0f;
float sinVal; /* 输入正弦信号 */
float cosVal; /* 输入余弦信号 */
PLL_Reset(&pll); /* 锁相环结构体变量重置 */
PLL_Clear(&pll);
PLL_Init(&pll);
PLL_Exec(&pll); /* 执行锁相环计算功能 */
```




```
float freq = pll.freq; /* 将锁相环计算提取的频率传出 */
float angle = pll.angle; /* 将锁相环计算提取的角度传出 */
```

7.3.2.4.4 注意事项

无。

7.3.2.5 PLL_SetTs

7.3.2.5.1 API 形式

表 7-17 设置 PLL 的控制周期 API

标题	描述
接口形式	void PLL_SetTs(PLL_Handle *pllHandle, float ts)
输入参数	pllHandle: PLL_Handle类型指针，指向锁相环结构体变量。 ts: 控制周期，单位：s。
返回参数	PLL的控制周期。
返回值	无。

7.3.2.5.2 API 功能描述

设置PLL计算的控制周期。

7.3.2.5.3 Example

```
PLL_Handle pll = {0};
float ts = 0.0001f;
PLL_SetTs(&pll, ts);
```

7.3.2.5.4 注意事项

合法的ts为正浮点数。

7.3.2.6 PLL_ParamUpdate

7.3.2.6.1 API 形式

表 7-18 设置 PLL 的带宽 API

标题	描述
接口形式	void PLL_ParamUpdate(PLL_Handle *pllHandle, float bdw)
输入参数	pllHandle: PLL_Handle类型指针，指向锁相环结构体变量。 bdw: 要设置的带宽，单位Hz。
返回参数	设置的带宽。



标题	描述
返回值	无。

7.3.2.6.2 API 功能描述

设置PLL计算的带宽。

7.3.2.6.3 Example

```
PLL_Handle pll = {0};  
float bdw = 40.0f;  
PLL_ParamUpdate(&pll, bdw);
```

7.3.2.6.4 注意事项

合法的bdw为正浮点数。



8 foc_loop_ctrl FOC 控制模块

8.1 电流控制器

实现dq轴电流闭环反馈控制。

8.1.1 电流控制器 handle 结构体说明

用于保存电流控制器参数的结构体，具体的定义如[表8-1](#)所示。

表 8-1 CURRCTRL_Handle 成员变量

成员类型	成员名	成员描述
DqAxis *	idqRef	dq轴电流指令值指针。
DqAxis *	idqFbk	dq轴电流反馈值指针。
DqAxis	idqFf	dq轴电流控制器前馈值。
PID_Handle	dAxisPi	d轴电流环PI控制器。
PID_Handle	qAxisPi	q轴电流环PI控制器。
MOTOR_Param *	mtrParam	电机参数结构体指针。
float	outLimit	电流控制器输出电压限幅值。
float	ts	电流控制器执行周期，单位： s。



8.1.2 API 接口

表 8-2 接口列表

函数名称	功能描述
CURRCTRL_Init	电流控制器初始化。
CURRCTRL_Reset	电流控制器所有成员复位为0。
CURRCTRL_Clear	电流环PI控制器历史值清除为0。
CURRCTRL_Exec	电流控制器执行方法。
CURRCTRL_SetTs	设置电流控制周期。

8.1.2.1 CURRCTRL_Init

8.1.2.1.1 API 形式

表 8-3 电流控制器初始化 API

标题	描述
接口形式	void CURRCTRL_Init(CURRCTRL_Handle *currHandle, MOTOR_Param *mtrParam, DqAxis *idqRef, DqAxis *idqFbk, const PI_Param dAxisPi, const PI_Param qAxisPi, float ts)
输入参数	currHandle: 指向电流控制器的指针。 mtrParam: 指向电机参数的指针。 idqRef: 指向dq轴参考电流的指针。 idqFbk: 指向dq轴反馈电流的指针。 dAxisPi: d轴PI参数。 qAxisPi: q轴PI参数。 ts: 控制周期, 单位: s。
返回参数	currHandle: 电流环控制结构体指针。
返回值	无。

8.1.2.1.2 API 功能描述

此API用于初始化电流控制器成员变量。

8.1.2.1.3 Example

```
CURRCTRL_Handle currHandle = {0};
MOTOR_Param mtrParam = {...}; /* 由用户配置参数 */
PI_Param dAxisPi = {...}; /* 由用户配置参数 */
```



```
PI_Param qAxisPi = {...}; /* 由用户配置参数 */
DqAxis *idqRef;
DqAxis *idqFbk;
float ts = 0.0001f;
CURRCTRL_Init(&currHandle, &mtrParam, idqRef, idqFbk, dAxisPi, qAxisPi, ts); /* 初始化电流控制器 */
```

8.1.2.1.4 注意事项

电流控制器中含有指针变量，在实例化时需要对其赋0。

8.1.2.2 CURRCTRL_Reset

8.1.2.2.1 API 形式

表 8-4 电流控制器变量复位 API

标题	描述
接口形式	void CURRCTRL_Reset(CURRCTRL_Handle *currHandle)
输入参数	currHandle：电流控制器结构体指针。
返回参数	currHandle：电流环控制器结构体指针。
返回值	无。

8.1.2.2.2 API 功能描述

此API用于MCU启动后复位电流控制器的所有成员变量。

8.1.2.2.3 Example

```
CURRCTRL_Handle currHandle = {0};
/* 前置条件：用户已完成电流控制器初始化 */
CURRCTRL_Reset(&currHandle); /* 复位电流控制器 */
```

8.1.2.2.4 注意事项

电流控制器中含有指针变量，在初始化时需要对其赋值。

8.1.2.3 CURRCTRL_Clear

8.1.2.3.1 API 形式

表 8-5 电流控制器历史值清除 API

标题	描述
接口形式	void CURRCTRL_Clear(CURRCTRL_Handle *currHandle)
输入参数	currHandle：电流控制器结构体指针。



标题	描述
返回参数	currHandle：电流控制器结构体指针。
返回值	无。

8.1.2.3.2 API 功能描述

此API用于在电机启动前清除dq轴电流环PI控制器历史信息。

8.1.2.3.3 Example

```
CURRCTRL_Handle currHandle = {0};
/* 前置条件：用户已完成电流控制器初始化 */

CURRCTRL_Clear(&currHandle); /* 清除电流控制器的历史信息 */
```

8.1.2.3.4 注意事项

在每次电机启动前需要执行该函数，清除历史变量的影响。

8.1.2.4 CURRCTRL_Exec

8.1.2.4.1 API 形式

表 8-6 电流控制器执行方法 API

标题	描述
接口形式	void CURRCTRL_Exec(CURRCTRL_Handle *currHandle, DqAxis *vdqRef, float spdRef, int ffEnable)
输入参数	currHandle：电流控制器结构体指针。 vdqRef：电流控制器输出的dq轴电压指令，单位：V。 spdRef：参考速度，用于前馈计算，单位：Hz。 ffEnable：电流前馈使能标志。 0：不使能； 1：使能。
返回参数	currHandle：电流控制器结构体指针。 vdqRef：电流控制器输出的dq轴电压指令值。
返回值	无。

8.1.2.4.2 API 功能描述

此API用于按照固定的频率执行电流控制器的计算。



8.1.2.4.3 Example

```
CURRCTRL_Handle currHandle = {0};
/* 前置条件：用户已完成电流控制器初始化 */

DqAxis vdqRef;
float spdRef = 35.0f; /* 35是示例数值，实际使用时由用户自行确定。 */
int ffEnable = 0;
CURRCTRL_Exec(&currHandle, &vdqRef, spdRef, ffEnable); /* 执行电流控制器的计算 */
```

8.1.2.4.4 注意事项

- 电流控制器的计算需要在固定周期的中断服务程序中执行。
- 在执行电流控制器计算之前，需要对电流控制器的对象进行初始化。

8.1.2.5 CURRCTRL_SetTs

8.1.2.5.1 API 形式

表 8-7 设置电流控制器的控制周期 API

标题	描述
接口形式	void CURRCTRL_SetTs(CURRCTRL_Handle *currHandle, float ts)
输入参数	currHandle：电流控制器结构体指针。 ts：电流环控制周期，单位：s。
返回参数	currHandle：电流控制器结构体指针。
返回值	无。

8.1.2.5.2 API 功能描述

设置电流环的控制周期。

8.1.2.5.3 Example

```
CURRCTRL_Handle currHandle = {0};
/* 前置条件：用户已完成电流控制器初始化 */
float ts = 0.0001f;
CURRCTRL_SetTs(&currHandle, ts);
```

8.1.2.5.4 注意事项

无。

8.2 电流环前馈控制



8.2.1 API 接口

表 8-8 接口列表

函数名称	功能描述
CURRFF_Exec	执行电流环前馈控制计算。

8.2.1.1 CURRFF_Exec

8.2.1.1.1 API 形式

表 8-9 电流环前馈控制计算 API

标题	描述
接口形式	void CURRFF_Exec(DqAxis *vdqRef, DqAxis idqFbk, MOTOR_Param *param, float spdRef, int enable)
输入参数	vdqRef: 指向dq轴参考电压的结构体指针。 idqFbk: dq轴电流反馈。 param: 指向电流参数的结构体指针。 spdRef: 参考速度, 单位Hz。 enable: 使能电流前馈控制的开关。0: 不使能; 1: 使能。
返回参数	vdqRef: dq轴参考电压的结构体指针。
返回值	无。

8.2.1.1.2 API 功能描述

执行电流环前馈控制计算。

8.2.1.1.3 Example

无。

8.2.1.1.4 注意事项

此API已在CURRCTRL_Exec()接口内部调用, 用户无需直接调用此接口。

用户通过控制CURRCTRL_Exec()接口的第三个入参ffEnable, 控制此接口是否被执行。ffEnable配置为0: CURRCTRL_Exec()内部不执行此API; ffEnable配置为1: CURRCTRL_Exec()内部执行此API。

8.3 弱磁控制



8.3.1 弱磁控制器 handle 结构体说明

用于保存弱磁参数及历史值的结构体，具体定义如表8-10所示。

表 8-10 FW_Handle 结构体说明

成员类型	成员名	成员描述
bool	enable	弱磁使能标志。
float	udcThreshPer	进入弱磁的电压百分比 * 1/ sqrt(3)。
float	ts	计算周期，单位：s。
float	idSlope	d轴电流变化斜坡，单位：A/s。
float	idRef	d轴电流指令，单位：A。
float	idMaxAmp	能注入的最大d轴电流，单位：A。
float	currMax	最大相电流，单位：A。
float	currMaxSquare	最大相电流的平方。
float	idDemag	消磁时的d轴电流，单位：A。

8.3.2 API 接口

表 8-11 接口列表

函数名称	功能描述
FW_Init	初始化弱磁控制器。
FW_Exec	执行弱磁控制器计算。
FW_SetTs	设置弱磁控制器的计算周期。
FW_Clear	清除弱磁控制器历史值。

8.3.2.1 FW_Init



8.3.2.1.1 API 形式

表 8-12 初始化弱磁控制器 API

标题	描述
接口形式	void FW_Init(FW_Handle *fw, float ts, bool enable, float currMax, float idDemag, float thr, float slope)
输入参数	fw: FW_Handle类型指针，指向弱磁结构体变量。 ts: 控制周期，单位：s。 enbale: 弱磁使能标志。 currMax: 最大相电流，单位：A。 idDemag: 退磁电流，单位：A。 thr: 进入弱磁控制时的电压使用率，单位：%。 slope: d轴电流斜坡，单位：A/s。
返回参数	fw: 弱磁控制器fw结构体指针。
返回值	无。

8.3.2.1.2 API 功能描述

初始化弱磁控制器参数。

8.3.2.1.3 Example

```
FW_Handle fw = {0}; /* 定义弱磁控制器变量 */  
  
float ts = 0.0001f;  
bool enable = 1;  
float currMax = 4.0f;  
float idDemag = -2.0f;  
float percent = 0.85f;  
float idSlope = 0.5f;  
FW_Init(&fw, ts, enable, currMax, idDemag, percent, idSlope);
```

8.3.2.1.4 注意事项

无。

8.3.2.2 FW_Exec

8.3.2.2.1 API 形式

表 8-13 弱磁控制执行 API

标题	描述
接口形式	void FW_Exec(FW_Handle *fw, DqAxis udqRef, float udc, DqAxis *idqRefRaw)



标题	描述
输入参数	fw: FW_Handle类型指针，指向弱磁控制器结构体变量。 udqRef: dq轴电压指令。 udc: 母线电压。 idqRefRaw: dq轴电流指令。
返回参数	fw: 弱磁控制器fw结构体指针。 idqRefRaw: dq轴电流指令结构体指针。
返回值	无。

8.3.2.2.2 API 功能描述

执行弱磁控制器计算，根据实际母线电压值，调整d轴电流指令。

8.3.2.2.3 Example

```
FW_Handle fw = {0}; /* 定义弱磁结构体变量 */
/* 前置条件: 已完成弱磁控制器初始化 */
DqAxis udqRef; /* dq轴电压指令。 */
float udc; /* 母线电压实时采样值。 */
DqAxis idqRefRaw; /* dq轴电流指令。 */
FW_Exec(&fw, udqRef, udc, &idqRefRaw);
```

8.3.2.2.4 注意事项

无。

8.3.2.3 FW_SetTs

8.3.2.3.1 API 形式

表 8-14 设置弱磁控制器控制周期 API

标题	描述
接口形式	void FW_SetTs(FW_Handle *fw, float ts)
输入参数	fw: FW_Handle类型指针，指向弱磁控制器结构体变量。 ts: 需要设置的控制周期。
返回参数	fw: 弱磁控制器fw结构体指针。
返回值	无。

8.3.2.3.2 API 功能描述

设置弱磁控制器的控制周期。



8.3.2.3.3 Example

```
FW_Handle fw = {0};
/* 前置条件：已完成弱磁控制器初始化 */
float ts = 0.0001f;
FW_SetTs(&fw, ts);
```

8.3.2.3.4 注意事项

无。

8.3.2.4 FW_Clear

8.3.2.4.1 API 形式

表 8-15 弱磁控制器清除历史值 API

标题	描述
接口形式	void FW_Clear(FW_Handle *fw)
输入参数	fw: FW_Handle类型指针，指向弱磁控制器结构体变量。
返回参数	无。
返回值	无。

8.3.2.4.2 API 功能描述

弱磁控制器清除控制历史值。

8.3.2.4.3 Example

```
FW_Handle fw = {0};
/* 前置条件：已完成弱磁控制器初始化 */
FW_Clear(&fw);
```

8.3.2.4.4 注意事项

无。

8.4 IF 控制器

实现电机在无位置传感器条件下以IF控制方式从零低速启动。

8.4.1 IF 控制器 handle 结构体说明

8.4.1.1 IF 控制器 handle 结构

用于IF启动控制器参数的结构体，具体的定义如[表8-16](#)所示。



表 8-16 IF_Handle 成员变量

成员类型	成员名	成员描述
float	anglePeriod	电流矢量角计算执行周期，单位：s。
float	curAmpPeriod	电流矢量幅值计算执行周期，单位：s。
float	targetAmp	电流矢量幅值目标值，单位：A。
float	curAmp	当前周期内电流矢量幅值，单位：A。
float	stepAmp	电流矢量幅值变化步长，单位：A。
float	angle	电流矢量角，单位：rad。

8.4.2 API 接口

表 8-17 接口列表

函数名称	功能描述
IF_Init	IF控制器结构体参数初始化。
IF_Clear	IF控制器历史值清除为0。
IF_CurrAmpCalc	电流矢量幅值计算方法。
IF_CurrAngleCalc	电流矢量相角计算方法。
IF_SetAngleTs	设置IF角度的控制周期。

8.4.2.1 IF_Init

8.4.2.1.1 API 形式

表 8-18 IF 控制器变量初始化 API

标题	描述
接口形式	void IF_Init(IF_Handle *ifHandle, float targetAmp, float currSlope, float stepAmpPeriod, float anglePeriod)
输入参数	ifHandle: IF控制器结构体指针。 targetAmp: IF启动的电流目标值，单位：A。 currSlope: IF启动电流从0增加到目标值的斜率，单位：A/s。 stepAmpPeriod: 电流从0增加到目标值的控制周期，使用速度环控制周期，单位：s。 anglePeriod: IF角度计算周期，使用电流环控制周期，单位s。



标题	描述
返回参数	ifHandle：IF控制器结构体指针。
返回值	无。

8.4.2.1.2 API 功能描述

此API用于电机启动前初始化IF控制器的所有成员变量。

8.4.2.1.3 Example

```
IF_Handle ifHandle = {0};
float tarAmp = 0.8f;
float currSlope = 0.1f;
float stepAmpPeriod = 0.0005f;
float anglePeriod = 0.0001f;
IF_Init(&ifHandle, tarAmp, currSlope, stepAmpPeriod, anglePeriod); /* 初始化IF控制器 */
```

8.4.2.1.4 注意事项

无。

8.4.2.2 IF_Clear

8.4.2.2.1 API 形式

表 8-19 IF 控制器历史值清除 API

标题	描述
接口形式	void IF_Clear(IF_Handle *ifHandle)
输入参数	ifHandle：IF控制器结构体指针。
返回参数	ifHandle：IF控制器结构体指针。
返回值	无。

8.4.2.2.2 API 功能描述

此API用于在电机启动前清除IF控制器的所有具有历史信息的成员变量，包括：curAmp，angle。

8.4.2.2.3 Example

```
IF_Handle ifCtrl = {0};
/* 前置条件：已完成IF控制器初始化 */
IF_Clear(&ifCtrl); /* 清除IF控制器的历史信息 */
```

8.4.2.2.4 注意事项

在每次电机启动前需要执行该函数，清除历史变量的影响。



8.4.2.3 IF_CurrAmpCalc

8.4.2.3.1 API 形式

表 8-20 IF 控制器电流矢量幅值计算方法 API

标题	描述
接口形式	float IF_CurrAmpCalc(IF_Handle *ifHandle)
输入参数	ifHandle: IF控制器结构体指针。
返回参数	ifHandle: IF控制器结构体指针。
返回值	定子电流幅值。

8.4.2.3.2 API 功能描述

此API用于按照固定的计算周期计算IF控制器中电流矢量幅值的指令值。

8.4.2.3.3 Example

```
DqAxis currRefDq;  
IF_Handle ifCtrl = {0};  
/* 前置条件: 已完成IF控制器初始化 */  
  
currRefDq.q = IF_CurrAmpCalc(&ifCtrl); /* 执行IF控制器电流矢量幅值的计算 */
```

8.4.2.3.4 注意事项

- 计算需要在固定周期的中断服务程序中执行。
- 在执行计算之前, 需要初始化IF控制器对象。

8.4.2.4 IF_CurrAngleCalc

8.4.2.4.1 API 形式

表 8-21 IF 控制器电流矢量相角计算方法 API

标题	描述
接口形式	float IF_CurrAngleCalc(IF_Handle *ifHandle, float spdRef)
输入参数	ifHandle: IF控制器结构体指针; spdRef: 速度指令值, 单位: Hz。
返回参数	ifHandle: IF控制器结构体指针。
返回值	定子电流相角, 单位: rad。



8.4.2.4.2 API 功能描述

此API用于按照固定的计算周期计算IF控制器中电流矢量的相角。

8.4.2.4.3 Example

```
float axisAngle;
float spdRefHz = 50.0f;
IF_Handle ifCtrl = {0};
/* 前置条件：已完成IF控制器初始化 */

axisAngle = IF_CurrAngleCalc(&ifCtrl, spdRefHz); /* 执行IF控制器电流矢量相角的计算 */
```

8.4.2.4.4 注意事项

- 计算需要在固定周期的中断服务程序中执行。
- 在执行计算之前，需要初始化IF控制器对象。

8.4.2.5 IF_SetAngleTs

8.4.2.5.1 API 形式

表 8-22 IF 控制器更新角度控制周期 API

标题	描述
接口形式	void IF_SetAngleTs(IF_Handle *ifHandle, float ts)
输入参数	ifHandle：IF控制器结构体指针。 ts：控制周期。
返回参数	ifHandle：IF控制器结构体指针。
返回值	无。

8.4.2.5.2 API 功能描述

此API用于设置IF模式下角度计算的控制周期。

8.4.2.5.3 Example

```
float ts = 0.0002f;
IF_Handle ifCtrl = {0};
/* 前置条件：已完成IF控制器初始化 */
IF_SetAngleTs(&ifCtrl, ts);/* 设置if控制周期 */
```

8.4.2.5.4 注意事项

无。

8.5 位置控制器



8.5.1 位置控制器 handle 结构体说明

用于保存位置控制器参数的结构体，具体的定义如表8-23所示。

表 8-23 POSCTRL_Handle 成员变量

成员类型	成员名	成员描述
PID_Handle	posPid	位置PID控制器。
float	posTarget	位置输入指令值，单位：rad。
float	posTargetBk	位置输入指令值备份，单位：rad。
RMG_Handle	posRmg	位置控制器斜波管理。
float	ts	位置控制器执行周期，单位：s。
int	angFbkLoop	位置控制周期计数。
float	angFbkPrev	上一控制周期中的反馈位置，单位：rad。
float	posFbk	绝对位置反馈值，单位：rad。
float	posIncRef	位置控制增量指令值，单位：rad。
float	posIncRefPrev	位置控制上一周期增量指令值，单位：rad。
float	spdRef	位置控制器输出速度指令值，单位：Hz。
float	posRef	位置控制器位置控制参考值，单位：rad。
float	posFbkPrev	位置控制器上一控制周期位置反馈值，单位：rad。
float	posErr	位置控制器位置误差，单位：rad。
POSCTRL_Mode	mode	位置控制器控制模式。
float	posTargetShadow	位置输入指令值影子变量，单位：rad。
float	runTime	单次轨迹规划控制持续时间，单位：s。
float	timeTick	位置控制器内部计数器，单位：s。
float	deltaTime	每段位置控制持续时间，单位：s。
float	accMax	加速度最大值，单位：m/s ² 。
float	jerk	加加速度值，单位：m/s ³ 。
int	targetUpdateBlockFlag	位置目标更新控制。



成员类型	成员名	成员描述
float	deltaTimeSq	每段位置控制持续时间平方值，单位： s^2 。
float	deltaTimeCu	每段位置控制持续时间三次方值，单位： s^3 。
float	timeStg[7]	轨迹控制每段控制的时间值，单位：s。

8.5.2 API 接口

表 8-24 接口列表

函数名称	功能描述
POSCTRL_Clear	位置控制器清除历史值并设置为0。
POSCTRL_Init	位置控制器初始化。
POSCTRL_PidExec	位置控制器PID执行方法。
POSCTRL_ModeSelect	位置控制模式设置。
POSCTRL_SetSlope	位置控制位置变化率设置。
POSCTRL_SetTarget	位置控制器位置目标值设置。
POSCTRL_Exec	位置控制器执行方法。
POSCTRL_AngleExpand	位置控制器绝对位置计算。
POSCTRL_SetKp	位置控制器设置kp参数。
POSCTRL_SetKi	位置控制器设置ki参数。
POSCTRL_SetKd	位置控制器设置kd参数。
POSCTRL_SetNs	位置控制器设置ns参数。

8.5.2.1 POSCTRL_Clear

8.5.2.1.1 API 形式

表 8-25 位置控制器历史值清除 API

标题	描述
接口形式	void POSCTRL_Clear(POSCTRL_Handle *posHandle);
输入参数	posHandle：位置控制器结构体指针。



标题	描述
返回参数	posHandle：位置控制器结构体指针。
返回值	无。

8.5.2.1.2 API 功能描述

此API用于清除位置控制器成员变量的历史值。

8.5.2.1.3 Example

```
POSCTRL_Handle posHandle = {0};
POSCTRL_Clear(&posHandle); /* 清除位置控制器历史值 */
```

8.5.2.1.4 注意事项

无。

8.5.2.2 POSCTRL_Init

8.5.2.2.1 API 形式

表 8-26 位置控制器初始化 API

标题	描述
接口形式	void POSCTRL_Init(POSCTRL_Handle *posHandle, const PID_Param *pidCtrlTable, float ts);
输入参数	posHandle：位置控制器结构体指针。 pidCtrlTable：位置控制器PID控制参数。 ts：位置控制器执行周期，单位：s。
返回参数	posHandle：位置控制器结构体指针。
返回值	无。

8.5.2.2.2 API 功能描述

此API用于初始化位置控制器成员变量。

8.5.2.2.3 Example

```
POSCTRL_Handle posHandle = {0};
PID_Param pidCtrlTable = {0.0f, 0.0f, 0.06f, ..., 100}; /* 示例参数，实际使用由用户配置 */
float ts = 0.0025;
POSCTRL_Init(&posHandle, &pidCtrlTable, ts); /* 初始化位置控制器*/
```



8.5.2.2.4 注意事项

无。

8.5.2.3 POSCTRL_PidExec

8.5.2.3.1 API 形式

表 8-27 位置控制器 PID 执行 API

标题	描述
接口形式	float POSCTRL_PidExec(POSCTRL_Handle *posHandle, float posErr);
输入参数	posHandle: 位置控制器结构体指针。 posErr: 位置误差, 单位: rad。
返回参数	posHandle: 位置控制器结构体指针。
返回值	速度指令, 单位: rad/s。

8.5.2.3.2 API 功能描述

此API是位置控制器PID执行函数, 可获得速度指令值。

8.5.2.3.3 Example

```
POSCTRL_Handle posHandle = {0};  
float posErr = 0.0025; /* 示例参数, 实际使用由用户配置 */  
/* 前置条件: 位置控制器已初始化。 */  
float spdRefRad = POSCTRL_PidExec(&posHandle, posErr); /* 位置控制器PI执行函数 */
```

8.5.2.3.4 注意事项

无。

8.5.2.4 POSCTRL_ModeSelect

8.5.2.4.1 API 形式

表 8-28 位置控制器模式选择 API

标题	描述
接口形式	void POSCTRL_ModeSelect(POSCTRL_Handle *posHandle, POSCTRL_Mode mode);



标题	描述
输入参数	posHandle：位置控制器结构体指针。 mode：位置控制模式。 POSCTRL_MODE_CONTINUOUS：连续模式； POSCTRL_MODE_TRAJ：轨迹控制模式。
返回参数	posHandle：位置控制器结构体指针。
返回值	无。

8.5.2.4.2 API 功能描述

此API可用于设置位置控制器的模式。

8.5.2.4.3 Example

```
POSCTRL_Handle posHandle = {0};
POSCTRL_Mode mode = POSCTRL_MODE_CONTINUOUS; /* 示例参数，实际使用由用户配置 */
/* 前置条件：位置控制器已初始化。 */
POSCTRL_ModeSelect(&posHandle, mode);
```

8.5.2.4.4 注意事项

无。

8.5.2.5 POSCTRL_SetSlope

8.5.2.5.1 API 形式

表 8-29 设置位置控制器的斜坡管理模块的斜率 API

标题	描述
接口形式	void POSCTRL_SetSlope(POSCTRL_Handle *posHandle, float slope);
输入参数	posHandle：位置控制器结构体指针。 slope：位置变化率，单位：Hz。
返回参数	posHandle：位置控制器结构体指针。
返回值	无。

8.5.2.5.2 API 功能描述

此API可用于设置位置控制器的斜坡管理模块的斜率。



8.5.2.5.3 Example

```
POSCTRL_Handle posHandle = {0};
float slope = 30.0f; /* 示例参数，实际使用由用户配置 */
/* 前置条件：位置控制器已初始化。 */
POSCTRL_SetSlope(&posHandle, slope);
```

8.5.2.5.4 注意事项

无。

8.5.2.6 POSCTRL_SetTarget

8.5.2.6.1 API 形式

表 8-30 位置控制器设置目标位置 API

标题	描述
接口形式	void POSCTRL_SetTarget(POSCTRL_Handle *posHandle, float posTarget);
输入参数	posHandle：位置控制器结构体指针。 posTarget：目标位置，单位：rad。
返回参数	posHandle：位置控制器结构体指针。
返回值	无。

8.5.2.6.2 API 功能描述

此API可用于设置位置控制器的目标位置。

8.5.2.6.3 Example

```
POSCTRL_Handle posHandle = {0};
float posTarget = 0.0f; /* 示例参数，实际使用由用户配置 */
/* 前置条件：位置控制器已初始化。 */
POSCTRL_SetTarget(&posHandle, posTarget);
```

8.5.2.6.4 注意事项

无。

8.5.2.7 POSCTRL_Exec



8.5.2.7.1 API 形式

表 8-31 位置控制器执行方法 API

标题	描述
接口形式	float POSCTRL_Exec(POSCTRL_Handle *posHandle, float posTarget, float posFbk);
输入参数	posHandle：位置控制器结构体指针。 posTarget：目标位置，单位：rad。 posFbk：位置反馈值，单位：rad。
返回参数	posHandle：位置控制器结构体指针。
返回值	速度指令，单位：Hz。

8.5.2.7.2 API 功能描述

此API是位置控制器的执行方法，可获得速度指令（Hz）。

8.5.2.7.3 Example

```
POSCTRL_Handle posHandle = {0};
float posTarget = 0.0f; /* 示例参数，实际使用由用户配置 */
float posFbk = 0.5f; /* 示例参数，实际使用由用户配置 */
/* 前置条件：位置控制器已初始化。 */
float spdRefHz = POSCTRL_Exec(&posHandle, posTarget, posFbk);
```

8.5.2.7.4 注意事项

无。

8.5.2.8 POSCTRL_AngleExpand

8.5.2.8.1 API 形式

表 8-32 位置控制器的绝对位置计算 API

标题	描述
接口形式	float POSCTRL_AngleExpand(POSCTRL_Handle *posHandle, float angFbk);
输入参数	posHandle：位置控制器结构体指针。 angFbk：反馈角度，单位：rad。
返回参数	posHandle：位置控制器结构体指针。
返回值	位置反馈值，单位：rad。



8.5.2.8.2 API 功能描述

此API是位置控制器的绝对位置（电机转子相对初始位置转过的累计角度，该角度不受限于0~2*pi范围内，是转动累计角度值）计算方法，可获得位置反馈值。

8.5.2.8.3 Example

```
POSCTRL_Handle posHandle = {0};
float angFbk = 1.5f; /* 示例参数，实际使用由用户配置 */
/* 前置条件：位置控制器已初始化。 */
float posFbk = POSCTRL_AngleExpand(&posHandle, angFbk);
```

8.5.2.8.4 注意事项

无。

8.5.2.9 POSCTRL_SetKp

8.5.2.9.1 API 形式

表 8-33 位置控制器设置 kp 参数 API

标题	描述
接口形式	void POSCTRL_SetKp(POSCTRL_Handle *posHandle, float kp);
输入参数	posHandle：位置控制器结构体指针。 kp：比例增益系数目标设置值。
返回参数	posHandle：位置控制器结构体指针。
返回值	无。

8.5.2.9.2 API 功能描述

此API是位置控制器设置PID控制器的比例增益系数。

8.5.2.9.3 Example

```
POSCTRL_Handle posHandle = {0};
float kp = 1.0f; /* 示例参数，实际使用由用户配置 */
/* 前置条件：位置控制器已初始化。 */
POSCTRL_SetKp(&posHandle, kp);
```

8.5.2.9.4 注意事项

无。

8.5.2.10 POSCTRL_SetKi



8.5.2.10.1 API 形式

表 8-34 位置控制器设置 ki 参数 API

标题	描述
接口形式	void POSCTRL_SetKi(POSCTRL_Handle *posHandle, float ki);
输入参数	posHandle：位置控制器结构体指针。 ki：积分增益系数目标设置值。
返回参数	posHandle：位置控制器结构体指针。
返回值	无。

8.5.2.10.2 API 功能描述

此API是位置控制器设置PID控制器的积分增益系数。

8.5.2.10.3 Example

```
POSCTRL_Handle posHandle = {0};
float ki = 1.0f; /* 示例参数，实际使用由用户配置 */
/* 前置条件：位置控制器已初始化。 */
POSCTRL_SetKi(&posHandle, ki);
```

8.5.2.10.4 注意事项

无。

8.5.2.11 POSCTRL_SetKd

8.5.2.11.1 API 形式

表 8-35 位置控制器设置 kd 参数 API

标题	描述
接口形式	void POSCTRL_SetKd(POSCTRL_Handle *posHandle, float kd);
输入参数	posHandle：位置控制器结构体指针。 ki：微分增益系数目标设置值。
返回参数	posHandle：位置控制器结构体指针。
返回值	无。

8.5.2.11.2 API 功能描述

此API是位置控制器设置PID控制器的微分增益系数。



8.5.2.11.3 Example

```
POSCTRL_Handle posHandle = {0};
float kd = 1.0f; /* 示例参数，实际使用由用户配置 */
/* 前置条件：位置控制器已初始化。 */
POSCTRL_SetKd(&posHandle, kd);
```

8.5.2.11.4 注意事项

无。

8.5.2.12 POSCTRL_SetNs

8.5.2.12.1 API 形式

表 8-36 位置控制器设置 PID 微分环节低通滤波的截止频率 API

标题	描述
接口形式	void POSCTRL_SetNs(POSCTRL_Handle *posHandle, float ns);
输入参数	posHandle：位置控制器结构体指针。 ns：微分环节低通滤波的截止频率，单位：rad/s。
返回参数	posHandle：位置控制器结构体指针。
返回值	无。

8.5.2.12.2 API 功能描述

此API是位置控制器设置PID控制器的微分环节低通滤波的截止频率。

8.5.2.12.3 Example

```
POSCTRL_Handle posHandle = {0};
float ns = 1.0f; /* 示例参数，实际使用由用户配置 */
/* 前置条件：位置控制器已初始化。 */
POSCTRL_SetNs(&posHandle, ns);
```

8.5.2.12.4 注意事项

无。

8.6 速度控制器

实现速度闭环反馈控制。

8.6.1 速度控制器 handle 结构体说明

用于保存速度控制器参数的结构体，具体的定义如[表8-37](#)所示。



表 8-37 SPDCTRL_Handle 成员变量

成员类型	成员名	成员描述
PID_Handle	spdPi	速度环PI控制器。
float	outLimit	速度环输出指令限幅值。
MOTOR_Param *	mtrParam	电机参数结构体指针。
float	ts	速度控制器执行周期，单位：S。

8.6.2 API 接口

表 8-38 接口列表

函数名称	功能描述
SPDCTRL_Init	速度控制器初始化。
SPDCTRL_Reset	速度控制器所有成员复位为0。
SPDCTRL_Clear	速度环PI控制器清除历史值并设置为0。
SPDCTRL_Exec	速度控制器执行方法。

8.6.2.1 SPDCTRL_Init

8.6.2.1.1 API 形式

表 8-39 速度控制器变量初始化 API

标题	描述
接口形式	void SPDCTRL_Init(SPDCTRL_Handle *spdHandle, MOTOR_Param *mtrParam, const PI_Param piParam, float ts)
输入参数	spdHandle：速度控制器结构体指针。 mtrParam：电机参数结构体指针。 piParam：PI控制器参数的结构体变量。 ts：控制周期。
返回参数	spdHandle：速度控制器结构体指针。
返回值	无。



8.6.2.1.2 API 功能描述

此API用于初始化速度控制器的所有成员变量。

8.6.2.1.3 Example

```
MOTOR_Param g_mtrParam;
/* 此处省略：对电机参数初始化已完成。 */

SPDCTRL_Handle spdCtrl = {0};
MOTOR_Param *mtrParam = &g_mtrParam;
PI_Param piParam = {...}; /* 此处省略，由用户自定义配置。 */
float ts = 0.0005f;
SPDCTRL_Init(&spdCtrl, mtrParam, piParam, ts); /* 初始化速度控制器 */
```

8.6.2.1.4 注意事项

无。

8.6.2.2 SPDCTRL_Reset

8.6.2.2.1 API 形式

表 8-40 速度控制器变量复位 API

标题	描述
接口形式	void SPDCTRL_Reset(SPDCTRL_Handle *spdHandle)
输入参数	spdHandle：速度控制器结构体指针。
返回参数	spdHandle：速度控制器结构体指针。
返回值	无。

8.6.2.2.2 API 功能描述

此API用于MCU启动后复位速度控制器的所有成员变量。

8.6.2.2.3 Example

```
SPDCTRL_Handle spdCtrl = {0};
/* 前置条件：已完成速度控制器初始化 */
SPDCTRL_Reset(&spdCtrl); /* 复位速度控制器 */
```

8.6.2.2.4 注意事项

速度控制器中含有指针变量，在实例化时需要对其赋0。

8.6.2.3 SPDCTRL_Clear



8.6.2.3.1 API 形式

表 8-41 速度环 PI 控制器历史值清除 API

标题	描述
接口形式	void SPDCTRL_Clear(SPDCTRL_Handle *spdHandle)
输入参数	spdHandle：速度控制器结构体指针。
返回参数	spdHandle：速度控制器结构体指针。
返回值	无。

8.6.2.3.2 API 功能描述

此API用于在电机启动前清除速度环PI控制器的所有具有历史信息的成员变量。

8.6.2.3.3 Example

```
SPDCTRL_Handle spdCtrl = {0};
/* 前置条件：已完成速度控制器初始化 */
SPDCTRL_Clear(&spdCtrl); /* 清除速度环PI控制器的历史信息 */
```

8.6.2.3.4 注意事项

在每次电机启动前需要执行该函数，清除历史变量的影响。

8.6.2.4 SPDCTRL_Exec

8.6.2.4.1 API 形式

表 8-42 速度环控制器执行方法 API

标题	描述
接口形式	float SPDCTRL_Exec(SPDCTRL_Handle *spdHandle, float spdTarget, float spdFdbk)
输入参数	spdHandle：速度环控制器结构体指针。 spdTarget：速度指令值，单位：Hz。 spdFdbk：速度反馈值，单位：Hz。
返回参数	spdHandle：速度控制器结构体指针。
返回值	速度控制器输出。

8.6.2.4.2 API 功能描述

此API用于按照固定的计算周期执行速度控制器的计算。



8.6.2.4.3 Example

```
float spdRefHz = 50.0f; /* 速度指令值为50Hz */
float spdEstHz = GetSpd(); /* 调用速度获取方法得到速度反馈值 */
SPDCTRL_Handle spdCtrl = {0};
/* 前置条件：已完成速度控制器初始化 */

float spdOut = SPDCTRL_Exec(&spdCtrl, spdRefHz, spdEstHz); /* 执行速度控制器的计算 */
```

8.6.2.4.4 注意事项

- 速度控制器的计算需要在固定周期的中断服务程序中执行。
- 在执行速度控制器计算之前，需要对速度控制器的对象进行初始化。

8.7 启动过程开闭环切换控制器

实现电机从基于IF的速度开环控制方式切换到速度闭环控制。

8.7.1 启动过程开闭环切换控制器 handle 结构体说明

8.7.1.1 启动过程枚举结构

用于定义电机启动过程各阶段的枚举类型，具体的定义如表8-43所示。

表 8-43 STARTUP_Stage 枚举类型

枚举值	变量描述
STARTUP_STAGE_CURR	开环启动过程中定子电流矢量幅值增加阶段。
STARTUP_STAGE_SPD	开环启动过程中定子电流矢量频率调节阶段。
STARTUP_STAGE_SWITCH	从开环控制切换到闭环控制阶段。
STARTUP_STAGE_DETECT	闭环切开环控制阶段。

8.7.1.2 切换过程控制器 handle 结构

用于保存切换过程控制器参数的结构体，具体的定义如表8-44所示。

表 8-44 STARTUP_Handle 成员变量

成员类型	成员名	成员描述
STARTUP_Stage	stage	启动过程阶段。
float	spdBegin	速度切换区间起点，单位：Hz。
float	spdEnd	速度切换区间终点，单位：Hz。



成员类型	成员名	成员描述
float	regionInv	速度切换区间长度的倒数。
float	initCurr	切换开始时电流值。

8.7.2 API 接口

表 8-45 接口列表

函数名称	功能描述
STARTUP_Init	切换过程控制器结构体参数初始化。
STARTUP_Clear	切换过程控制器历史值清除为0。
STARTUP_CurrCal	切换过程电流指令值计算。

8.7.2.1 STARTUP_Init

8.7.2.1.1 API 形式

表 8-46 切换过程控制器变量初始化 API

标题	描述
接口形式	void STARTUP_Init(STARTUP_Handle *startHandle, float spdBegin, float spdEnd)
输入参数	startHandle: 切换过程控制器结构体指针。 spdBegin: 速度切换区间起点, 单位: Hz。 spdEnd: 速度切换区间终点, 单位: Hz。
返回参数	startHandle: 切换过程控制器结构体指针。
返回值	无。

8.7.2.1.2 API 功能描述

此API用于电机启动前初始化切换控制器的所有具有历史信息的成员变量。

8.7.2.1.3 Example

```
STARTUP_Handle start = {0};
STARTUP_Init(&start, 30.0f, 50.0f); /* 初始化切换过程控制器，速度切换区间为[30.0, 50.0]Hz */
```

8.7.2.1.4 注意事项

速度切换区间参数为正数，当电机速度指令为负数时，需要取其绝对值。



8.7.2.2 STARTUP_Clear

8.7.2.2.1 API 形式

表 8-47 切换过程控制器历史值清除 API

标题	描述
接口形式	void STARTUP_Clear(STARTUP_Handle *startHandle)
输入参数	startHandle：切换过程控制器结构体指针。
返回参数	startHandle：切换过程控制器结构体指针。
返回值	无。

8.7.2.2.2 API 功能描述

此API用于在电机启动前将stage初始化为STARTUP_STAGE_CURR。

8.7.2.2.3 Example

```
STARTUP_Handle start = {0};  
/* 前置条件：已完成启动过程开闭环切换控制器初始化 */  
STARTUP_Clear(&start); /* 清除切换过程控制器的历史信息 */
```

8.7.2.2.4 注意事项

在每次电机启动前需要执行该函数，清除历史变量的影响。

8.7.2.3 STARTUP_CurrCal

8.7.2.3.1 API 形式

表 8-48 切换过程控制器电流指令值计算方法 API

标题	描述
接口形式	float STARTUP_CurrCal(STARTUP_Handle *startHandle, float spdRef)
输入参数	startHandle：切换过程控制器结构体指针。 spdRef：速度指令值，单位：Hz。
返回参数	startHandle：切换过程控制器结构体指针。
返回值	电流指令值。

8.7.2.3.2 API 功能描述

此API用于按照固定的计算周期计算切换过程中电流指令值。



8.7.2.3.3 Example

```
DqAxis currRefDq;  
STARTUP_Handle start = {0};  
/* 前置条件：已完成启动过程开闭环切换控制器初始化 */  
  
float refHz = GetSpdRef(); /* 获取速度指令值 */  
currRefDq.d = STARTUP_CurrCal(&start, refHz); /* 执行切换过程控制器中电流指令值的计算 */
```

8.7.2.3.4 注意事项

- 计算需要在固定周期的中断服务程序中执行。
- 在执行计算之前，需要初始化切换过程控制器对象。



9 math 模块

9.1 基础数学方法

9.1.1 三角函数计算 handle 结构体说明

用于保存正余弦函数值的结构体，具体定义如[表9-1](#)所示。

表 9-1 TrigVal 三角函数值结构体说明

成员类型	成员名	成员描述
float	sin	正弦函数值。
float	cos	余弦函数值。

9.1.2 API 接口

表 9-2 接口列表

函数名称	功能描述
GetSin	采用泰勒级数计算sin值。
GetCos	采用泰勒级数计算cos值。
TrigCalc	泰勒级数展开法计算三角函数的sin和cos值。
ParkCalc	Park变换。
InvParkCalc	逆Park变换。
ClarkeCalc	Clarke变换。
Abs	取绝对值。



函数名称	功能描述
Clamp	限幅运算。
Max	取两个数中的较大值。
Min	取两个数中的较小值。
Sqrt	开方运算。
AngleSub	计算角度差值。
Mod	取模运算。
Sat	饱和函数计算。
Atan2	带符号的反正切运算。

9.1.2.1 GetSin

9.1.2.1.1 API 形式

表 9-3 三角函数 sin 计算 API

标题	描述
接口形式	float GetSin(float angle)
输入参数	angle: 计算使用的角度值, 单位: rad。
返回参数	无。
返回值	angle的sin值。

9.1.2.1.2 API 功能描述

使用泰勒级数展开, 计算angle的sin值。

9.1.2.1.3 Example

```
float angle = 1.57f;
float sinVal = GetSin(angle); /* 90°的sin值为1 */
```

9.1.2.1.4 注意事项

无。

9.1.2.2 GetCos



9.1.2.2.1 API 形式

表 9-4 三角函数计算 API

标题	描述
接口形式	float GetCos(float angle)
输入参数	angle: 计算使用的角度值, 单位: rad。
返回参数	无。
返回值	angle的cos值。

9.1.2.2.2 API 功能描述

使用泰勒级数展开, 计算angle的cos值。

9.1.2.2.3 Example

```
float angle = 1.57f;
float cosVal = GetCos(angle); /* 90°的cos值为0 */
```

9.1.2.2.4 注意事项

无。

9.1.2.3 TrigCalc

9.1.2.3.1 API 形式

表 9-5 三角函数计算 API

标题	描述
接口形式	void TrigCalc (TrigVal *val, float angle)
输入参数	val: 指向TrigVal结构体的指针。 angle: 计算使用的角度值, 单位: rad。
返回参数	val: TrigVal类型指针, 指向对应的三角函数值结构体。
返回值	无。

9.1.2.3.2 API 功能描述

使用泰勒级数展开, 同时获取输入角度值的sin值和cos值。



9.1.2.3.3 Example

```
TrigVal localTrigVal = {0}; /* 定义保存返回结果的变量 */
float angle = 3.1415926f;
TrigCalc(&localTrigVal, angle); /* 求解180°对应的正余弦值并存放在localTrigVal中 */
```

9.1.2.3.4 注意事项

无。

9.1.2.4 ParkCalc

9.1.2.4.1 API 形式

表 9-6 Park 变换 API

标题	描述
接口形式	void ParkCalc(const AlbeAxis *albe, float angle, DqAxis *dq)
输入参数	albe: AlbeAxis类型指针，指向αβ轴下变量。 angle: 计算使用的角度值，为从α轴为起点旋转至当前d轴位置所经过的角度，单位：rad。
返回参数	dq: DqAxis类型指针，指向dq轴下变量。
返回值	无。

9.1.2.4.2 API 功能描述

根据Park变换公式，将变量从两相静止坐标系（αβ轴）变换到两相旋转坐标系（dq轴）。Park变换公式：

$$\begin{bmatrix} d \\ q \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \end{bmatrix}$$

9.1.2.4.3 Example

```
AlbeAxis albe = {1.0f, 1.0f}; /* 定义αβ轴下的变量 */
DqAxis dq = {0}; /* 定义dq轴下的变量 */
ParkCalc(&albe, 3.1415926f, &dq); /* 表示α轴逆时针旋转至d轴位置所经过的角度（180°）。函数返回αβ轴下的变量变换到dq轴下的结果 */
```

9.1.2.4.4 注意事项

与InvParkCalc互为逆运算。

9.1.2.5 InvParkCalc



9.1.2.5.1 API 形式

表 9-7 逆 Park 变换 API

标题	描述
接口形式	void InvParkCalc(const DqAxis *dq, float angle, AlbeAxis *albe)
输入参数	<p>dq: DqAxis类型指针，指向dq坐标系下的变量。</p> <p>angle: 计算使用的角度值，为从α轴为起点旋转至当前d轴位置所经过的角度，单位：rad。</p>
返回参数	albe: AlbeAxis类型指针，指向αβ轴下变量。
返回值	无。

9.1.2.5.2 API 功能描述

根据Park变换公式，将变量从两相旋转坐标系（dq轴）变换到两相静止坐标系（αβ轴）。逆Park变换公式：

$$\begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} d \\ q \end{bmatrix}$$

9.1.2.5.3 Example

```
DqAxis dq = {1.0f, 1.0f}; /* 定义dq轴下的变量 */
AlbeAxis albe = {0}; /* 定义αβ轴下的变量 */
InvParkCalc(&dq, 3.1415926f, &albe); /* 表示α轴逆时针旋转至d轴位置所经过的角度（180°）。函数返回dq轴下的变量变换到αβ轴下的结果 */
```

9.1.2.5.4 注意事项

与ParkCalc互为逆运算。

9.1.2.6 ClarkeCalc

9.1.2.6.1 API 形式

表 9-8 Clarke 变换 API

标题	描述
接口形式	void ClarkeCalc(const UvwAxis *uvw, AlbeAxis *albe)
输入参数	uvw: UvwAxis类型指针，指向UVW自然坐标系下变量。
返回参数	albe: AlbeAxis类型指针，指向αβ轴下变量。
返回值	无。



9.1.2.6.2 API 功能描述

根据Clarke变换公式，将变量从三相静止坐标系（UVW轴）变换到两相静止坐标系（αβ轴）。Clarke变换公式：

$$\begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \frac{2}{3} \begin{bmatrix} 1 & -1/2 & -1/2 \\ 0 & \sqrt{3}/2 & -\sqrt{3}/2 \end{bmatrix} \begin{bmatrix} u \\ v \\ w \end{bmatrix}$$

9.1.2.6.3 Example

```
UvwAxis uvw = {1.0f, 1.0f, 1.0f}; /* 定义UVW坐标系下的变量 */
AlbeAxis albe = {0}; /* 定义αβ坐标系下的变量 */
ClarkeCalc(&uvw, &albe); /* 函数返回UVW轴下的变量变换到αβ轴下的结果 */
```

9.1.2.6.4 注意事项

无。

9.1.2.7 Abs

9.1.2.7.1 API 形式

表 9-9 取绝对值 API

标题	描述
接口形式	float Abs(float val)
输入参数	val：待计算绝对值的数值。
返回参数	无。
返回值	val的绝对值。

9.1.2.7.2 API 功能描述

计算输入浮点数值的绝对值。

9.1.2.7.3 Example

```
float val = -5.0f;
float ret;
ret = Abs(val); /* 函数返回val的绝对值，为5.0f */
```



9.1.2.7.4 注意事项

无。

9.1.2.8 Clamp

9.1.2.8.1 API 形式

表 9-10 限幅运算 API

标题	描述
接口形式	float Clamp(float val, float upperLimit, float lowerLimit)
输入参数	val: 待限幅的数值。 upperLimit: 限幅的上限值。 lowerLimit: 限幅的下限值。
返回参数	无。
返回值	限幅钳位后的结果值。

9.1.2.8.2 API 功能描述

根据输入函数上限值和下限值，对输入浮点数值进行限幅。

9.1.2.8.3 Example

```
float val = -5.0f;  
float ret;  
ret = Clamp(val,2.0f,-2.0f); /* 函数将输入值-5.0f进行限幅，限幅上限为2.0f，下限为-2.0f，因输入  
值-5.0f超出下限值，函数返回值为-2.0f */
```

9.1.2.8.4 注意事项

合法的限幅值需要满足upperLimit大于等于lowerLimit，输入数值的有效位数不应超过6位。

9.1.2.9 Max

9.1.2.9.1 API 形式

表 9-11 两个数中取较大值 API

标题	描述
接口形式	float Max(float val1, float val2)
输入参数	val1: 待比较数值1。 val2: 待比较数值2。



标题	描述
返回参数	无。
返回值	两个输入数值中较大的数值。

9.1.2.9.2 API 功能描述

对输入的两个浮点数进行比较，返回其中较大的一个数值。

9.1.2.9.3 Example

```
float val1 = -5.0f;
float val2 = 3.0f;
float ret;
ret = Max(val1, val2); /* 比较输入数值-5.0f和3.0f的大小，函数返回其中较大的一个数值，为3.0f */
```

9.1.2.9.4 注意事项

float类型精度为7位，能绝对保证的为6位，参与比较的两个数值不应超过6位有效数字。

9.1.2.10 Min

9.1.2.10.1 API 形式

表 9-12 两个数中取较小值 API

标题	描述
接口形式	float Min(float val1, float val2)
输入参数	val1：待比较数值1。 val2：待比较数值2。
返回参数	无。
返回值	两个输入数值中较小的数值。

9.1.2.10.2 API 功能描述

对输入的两个浮点数进行比较，返回其中较小的一个数值。

9.1.2.10.3 Example

```
float val1 = -5.0f;
float val2 = 3.0f;
float ret;
ret = Min(val1, val2); /* 比较输入数值-5.0f和3.0f的大小，函数返回其中较小的一个数值，为-5.0f */
```



9.1.2.10.4 注意事项

float类型精度为7位，能绝对保证的为6位，参与比较的两个数值不应超过6位有效数字。

9.1.2.11 Sqrt

9.1.2.11.1 API 形式

表 9-13 开方运算 API

标题	描述
接口形式	float Sqrt(float val)
输入参数	val：待进行开方运算的数值。
返回参数	无。
返回值	对输入数值开方运算的结果。

9.1.2.11.2 API 功能描述

对输入的浮点数进行开方计算。

9.1.2.11.3 Example

```
float val = 4.0f;
float ret;
ret = Sqrt(val); /* 函数返回2.0f */
```

9.1.2.11.4 注意事项

合法的输入应为非负的浮点数。

9.1.2.12 AngleSub

9.1.2.12.1 API 形式

表 9-14 角度相减运算 API

标题	描述
接口形式	float AngleSub(float angle1, float angle2)
输入参数	angle1：被减数（第一个角度）。 angle2：减数（第二个角度）。
返回参数	无。
返回值	angle1~angle2的角度差，限制在[-pi,pi]。



9.1.2.12.2 API 功能描述

对输入的浮点角度进行角度差值计算，计算结果在 $[-\pi,\pi]$ 内。

9.1.2.12.3 Example

```
float angle1 = 3.0f;
float angle2 = 2.0f;
float ret;
ret = AngleSub(angle1, angle2); /* 函数返回1.0f */
```

9.1.2.12.4 注意事项

无。

9.1.2.13 Sat

9.1.2.13.1 API 形式

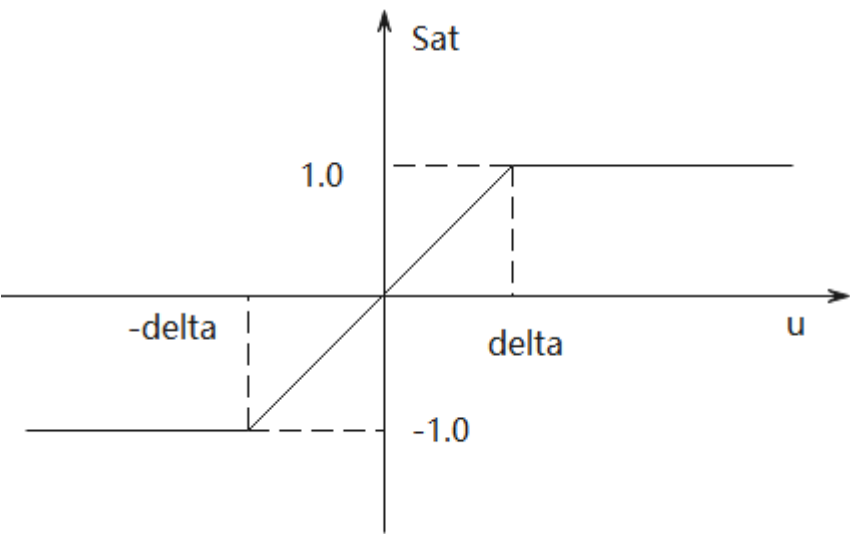
表 9-15 饱和函数 API

标题	描述
接口形式	float Sat(float u, float delta)
输入参数	u：需要被饱和限幅的数值。 delta：饱和输出点。
返回参数	无。
返回值	返回-1~+1之间的结果。

9.1.2.13.2 API 功能描述

对输入的浮点数进行饱和输出。当 $u>\delta$ 时输出1.0，当 $u<-\delta$ 时，输出-1.0，否则输出 u/δ ，其中 $\delta>0$ 。

图 9-1 饱和函数示意图





9.1.2.13.3 Example

```
float u = 0.0f;
float delta = 1.0f;
float ret = Sat(u,delta);
```

9.1.2.13.4 注意事项

合法的delta应为非负的浮点数。

9.1.2.14 Mod

9.1.2.14.1 API 形式

表 9-16 取模运算 API

标题	描述
接口形式	float Mod(float val1, float val2)
输入参数	val1：待取模运算的数值。 val2：模数。
返回参数	无。
返回值	对输入数值取模运算的结果。

9.1.2.14.2 API 功能描述

对输入的浮点数进行取模运算。

9.1.2.14.3 Example

```
float val1 = 5.4f;
float val2 = 2.0f;
float ret;
ret = Mod(val1, val2); /* 函数返回1.4f */
```

9.1.2.14.4 注意事项

合法的输入val2应为非负的浮点数。

9.1.2.15 Atan2

9.1.2.15.1 API 形式

表 9-17 带符号的反正切运算 API

标题	描述
接口形式	float Atan2(float x, float y)



标题	描述
输入参数	x: x轴坐标; y: y轴坐标。
返回参数	无。
返回值	对输入数值反正切计算结果。

9.1.2.15.2 API 功能描述

对输入的浮点数进行反正切运算。

9.1.2.15.3 Example

```
float x = -1.0f;  
float y = 1.0f;  
float ret;  
ret = Atan2(x, y); /* 函数返回2.3562弧度(即为135°) */
```

9.1.2.15.4 注意事项

无。

9.2 数学常数宏定义

在电机控制中需要使用一些数学常数，为了方便管理，将其定义为宏，具体定义如表 9-18 所示。

表 9-18 数学常数宏定义

宏名称	数值	描述
ONE_DIV_THREE	0.3333333f	1/3。
TWO_DIV_THREE	0.6666667f	2/3。
ONE_PI_DIV_SIX	0.5235988f	$\pi/6$ 。
ONE_PI_DIV_THREE	1.047197f	$\pi/3$ 。
DOUBLE_PI_DIV_THREE	2.094395f	$2*\pi/3$ 。
ONE_PI	3.141593f	π 。
DOUBLE_PI	6.283185f	$2*\pi$ 。
SQRT3_DIV_TWO	0.8660254f	$\text{sqrt}(3)/2$ 。
ONE_DIV_SQRT3	0.5773503f	$1/\text{sqrt}(3)$ 。
ONE_DIV_DOUBLE_PI	0.1591549f	$1/(2*\pi)$ 。



宏名称	数值	描述
RAD_TO_DEG	57.29578f	$180/\pi$ ，从弧度制转换为角度制的转换系数。
RAD_TO_DIGITAL	10430.06f	$32767/\pi$ ，从弧度制转换为Q15格式角度的转换系数。
DIGITAL_TO_RAD	0.00009587673f	$\pi/32767$ ，从Q15格式角度转换为弧度制的转换系数。
HALF_PI	1.5707963f	$\pi/2$ 。
THREE_PI_DIV_TWO	4.7123890f	$3*\pi/2$ 。
ONE_DIV_SIX	0.16666667f	$1/6$ 。
SEVEN_DIV_SIX	1.16666667f	$7/6$ 。
SIXTY_FIVE_DIV_SIX	10.8333333f	$65/6$ 。
SEVENTY_ONE_DIV_SIX	11.8333333f	$71/6$ 。
ONE_DIV_NINE	0.11111111f	$1/9$ 。
ONE_DIV_TWELVE	0.08333333f	$1/12$ 。
SQRT2	1.41421356f	$\text{sqrt}(2)$ 。
SMALL_FLOAT	0.00000001f	较小的浮点数。
LARGE_FLOAT	10000.0f	较大的浮点数，用于P限幅。

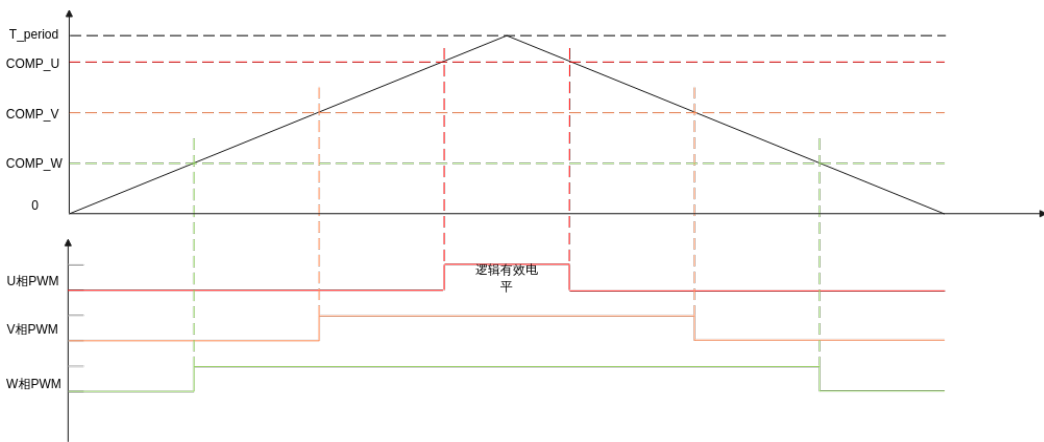


10 modulation 调制模块

10.1 空间矢量脉宽调制

空间矢量脉宽调制算法将控制器需要输出的静止坐标系下的定子电压矢量 $U(u_\alpha, u_\beta)$ 转变为UVW三相PWM信号的比较值。本模块基于七段式、PWM增减计数方式、有效电平在PWM周期中央等特性实现如图10-1所示的PWM波形。

图 10-1 UVW 三相比较值与 PWM 波形示意图



10.1.1 SVPWM 控制器参数描述

10.1.1.1 空间矢量脉宽调制 handle 结构

用于保存空间矢量脉宽调制算法所需参数的结构体，具体定义如表10-1所示。

表 10-1 SVPWM_Handle 结构体说明

成员类型	成员名	成员描述
float	voltPu	定子电压标么基准值。



成员类型	成员名	成员描述
float	oneDivVoltPu	voltPu的倒数，用于减少除法运算。

10.1.1.2 空间矢量脉宽调制计算变量结构体

用于保存空间矢量脉宽调制算法计算过程中间变量的结构体，具体定义如表10-2所示。

表 10-2 SVPWM_CALC_Handle 结构体说明

成员类型	成员名	成员描述
float	vAlpha	电压矢量α轴分量。
float	vBeta	电压矢量β轴分量。
float	t1	第一个基本电压矢量的作用时间，标么值。
float	t2	第二个基本电压矢量的作用时间，标么值。
unsigned short	indexU	U相占空比索引。
unsigned short	indexV	V相占空比索引。
unsigned short	indexW	W相占空比索引。
short int	sectorIndex	电压矢量扇区号。
float	volt[SVPWM_VOLT_TOTAL]	计算电压矢量作用时间的临时变量。SVPWM_VOLT_TOTAL值定义为3。
float	comp[SVPWM_COMP_VAL_TOTAL]	三相比较值，按照小中大进行排序。SVPWM_COMP_VAL_TOTAL值定义为3。

10.1.2 API 接口

表 10-3 接口列表

函数名称	功能描述
SVPWM_Init	SVPWM参数配置。
SVPWM_SectorCalc	SVPWM算法电压矢量扇区判定。
SVPWM_CompareValCalc	SVPWM算法UVW三相比较值计算。
SVPWM_IndexConvert	SVPWM算法扇区索引转换。



函数名称	功能描述
SVPWM_Exec	执行SVPWM算法。

10.1.2.1 SVPWM_Init

10.1.2.1.1 API 形式

表 10-4 SVPWM_Handle 初始化 API

标题	描述
接口形式	void SVPWM_Init(SVPWM_Handle *svHandle, float voltPu)
输入参数	svHandle: SVPWM_Handle结构体指针。 voltPu: 定子电压标么基准值。
返回参数	SVPWM结构体的标么电压。
返回值	无。

10.1.2.1.2 API 功能描述

设置定子电压标么基准值，并计算其倒数。

10.1.2.1.3 Example

```
SVPWM_Handle sv = {0};
SVPWM_Init(&sv, 179.0f); /* 设置SvpwmHandle的标么基准值为179V */
```

10.1.2.1.4 注意事项

电压标么基准值通常设置为0.5773倍母线电压。

10.1.2.2 SVPWM_SectorCalc

10.1.2.2.1 API 形式

表 10-5 电压矢量扇区和作用时间计算 API

标题	描述
接口形式	void SVPWM_SectorCalc(SVPWM_CALC_Handle *svCalc)
输入参数	svCalc: SVPWM_CALC_Handle结构体指针。
返回参数	无。
返回值	无。



10.1.2.2.2 API 功能描述

根据输入的 $\alpha\beta$ 坐标系下的定子电压矢量计算电压矢量的扇区号和作用时间。

10.1.2.2.3 Example

```
SVPWM_CALC_Handle svCalc = {0};
svCalc.vAlpha = 0.1f; /* 电压矢量 $\alpha$ 轴分量相对母线电压基值的比例为0.1 */
svCalc.vbeta = 0.5f; /* 电压矢量 $\beta$ 轴分量相对母线电压基值的比例为0.5 */
SVPWM_SectorCalc(&svCalc); /* 计算输入电压的扇区号和电压矢量作用时间 */
```

10.1.2.2.4 注意事项

无。

10.1.2.3 SVPWM_CompareValCalc

10.1.2.3.1 API 形式

表 10-6 电压矢量扇区和作用时间计算 API

标题	描述
接口形式	void SVPWM_CompareValCalc(SVPWM_CALC_Handle *svCalc)
输入参数	svCalc: SVPWM_CALC_Handle结构体指针。
返回参数	无。
返回值	无。

10.1.2.3.2 API 功能描述

根据电压矢量的扇区号和作用时间计算按照小中大进行排序的比较值占PWM半周期的比例。

10.1.2.3.3 Example

```
SVPWM_CALC_Handle svCalc = {0};
svCalc.vAlpha = 0.1f; /* 电压矢量 $\alpha$ 轴分量相对母线电压基值的比例为0.1 */
svCalc.vbeta = 0.5f; /* 电压矢量 $\beta$ 轴分量相对母线电压基值的比例为0.5 */
SVPWM_SectorCalc(&svCalc); /* 计算输入电压的扇区号和电压矢量作用时间 */
SVPWM_CompareValCalc(&svCalc); /* 计算UVW三相比较值占PWM半周期的比例 */
```

10.1.2.3.4 注意事项

无。

10.1.2.4 SVPWM_IndexConvert



10.1.2.4.1 API 形式

表 10-7 基于扇区变换的三相占空比数据索引 API

标题	描述
接口形式	void SVPWM_IndexConvert(SVPWM_CALC_Handle *svCalc)
输入参数	svCalc: SVPWM_CALC_Handle结构体指针。
返回参数	无。
返回值	无。

10.1.2.4.2 API 功能描述

根据电压矢量的扇区号设定UVW三相比较值占PWM半周期的比例与比较值的对应关系。

10.1.2.4.3 Example

```
SVPWM_CALC_Handle svCalc = {0};
svCalc.vAlpha = 0.1f; /* 电压矢量α轴分量相对母线电压基值的比例为0.1 */
svCalc.vbeta = 0.5f; /* 电压矢量β轴分量相对母线电压基值的比例为0.5 */
SVPWM_SectorCalc(&svCalc); /* 计算输入电压的扇区号和电压矢量作用时间 */
SVPWM_CompareValCalc(&svCalc); /* 计算按大小排序的比较值 */
SVPWM_IndexConvert(&svCalc); /* 根据扇区号设定UVW三相比较值与排序比较值的对应关系 */
```

10.1.2.4.4 注意事项

无。

10.1.2.5 SVPWM_Exec

10.1.2.5.1 API 形式

表 10-8 SVPWM 执行 API

标题	描述
接口形式	void SVPWM_Exec(const SVPWM_Handle *svHandle, const AlbeAxis *uAlbe, UvwAxis *dutyUvw)
输入参数	svHandle: SVPWM_Handle结构体指针。 uAlbe: 待输出定子电压。
返回参数	dutyUvw: 三相PWM比较值相对PWM计数周期值的占空比。
返回值	无。



10.1.2.5.2 API 功能描述

将电流控制器输出的静止坐标系下的定子电压矢量调制为三相PWM比较值相对PWM最大计数值的占空比。

10.1.2.5.3 Example

```
SVPWM_Handle sv = {0};
SVPWM_Init(&sv, 179.0f);
AlbeAxis vab = {0};
UvwAxis dutyUvw;
vab.alpha = 50.0f;
vab.beta = 10.0f
SVPWM_Exec(&sv, &vab, &dutyUvw)
```

10.1.2.5.4 注意事项

- 实际应用中需要注意PWM波形的产生方式与本模块默认方式是否一致。
- APT模块的比较值等于dutyUvw某一项分量乘以APT计数器周期值。

10.2 单电阻 SVPWM 调制和电流重构方法

单电阻采样模式下，在通用空间矢量脉宽调制方法的基础上增加移项特性和电流重构方法。

10.2.1 单电阻空间矢量脉宽调制 handle 结构

用于保存单电阻采样方式下空间矢量脉宽调制方法中所需控制参数的结构体，具体的定义如表10-9所示。

表 10-9 R1SVPWM_Handle 成员变量

成员类型	成员名	成员描述
float	voltPu	定子电压标么基准值。
float	oneDivVoltPu	voltPu的倒数，用于减少除法运算。
float	sampleWindow	电流采样窗口大小。
float	samplePointShift	电流采样点偏移值。
unsigned int	voltIndex	电压矢量扇区号。
unsigned int	voltIndexLast	上一拍电压矢量扇区号。
float	samplePoint[R1_ADC_SAMPLE_NUMS]	两次电流采样点。 R1_ADC_SAMPLE_NUMS值定义为2。



10.2.2 API 接口

表 10-10 接口列表

函数名称	功能描述
R1SVPWM_Init	单电阻R1SVPWM参数初始化配置。
R1SVPWM_Clear	R1SVPWM历史值清除为0。
R1SVPWM_Exec	单电阻R1SVPWM调制方法执行。
R1SVPWM_PhaseShift	单电阻调制非观测区移相控制。
R1CurrReconstruct	单电阻电流采样重构方法。

10.2.2.1 R1SVPWM_Init

10.2.2.1.1 API 形式

表 10-11 R1SVPWM_Handle 初始化 API

标题	描述
接口形式	void R1SVPWM_Init(R1SVPWM_Handle *r1svHandle, float voltPu, float samplePointShift, float sampleWindow)
输入参数	r1svHandle: R1SVPWM_Handle结构体指针。 voltPu: 定子电压标么基准值。 samplePointShift: 电流采样点偏移值占空比, [0,1]。 sampleWindow: 电流采样窗口大小占空比, [0,1]。
返回参数	无。
返回值	无。

10.2.2.1.2 API 功能描述

此API用于电机启动前初始化R1SVPWM_Handle的部分成员变量。

10.2.2.1.3 Example

```
R1SVPWM_Handle r1Sv = {0};
R1SVPWM_Init(&r1Sv, 179.0f, 0.008f, 0.06f); /* 设置母线电压基值为179V，采样窗口占空比为0.06，采样点偏移值占空比为0.008 */
```

10.2.2.1.4 注意事项

- 电压标么基准值通常设置为0.5773倍母线电压。
- 移相窗口大小和电流采样点的设置依赖硬件条件。



10.2.2.2 R1SVPWM_Clear

10.2.2.2.1 API 形式

表 10-12 R1SVPWM_Handle 历史值清除 API

标题	描述
接口形式	void R1SVPWM_Clear(R1SVPWM_Handle *r1svHandle)
输入参数	r1svHandle：R1SVPWM_Handle结构体指针。
返回参数	无。
返回值	无。

10.2.2.2.2 API 功能描述

此API用于电机启动前清除R1SVPWM_Handle的所有具有历史信息的变量。

10.2.2.2.3 Example

```
R1SVPWM_Handle r1Sv = {0};
R1SVPWM_Clear(&r1Sv); /* 清除R1SVPWM_Handle的历史信息 */
```

10.2.2.2.4 注意事项

无。

10.2.2.3 R1SVPWM_Exec

10.2.2.3.1 API 形式

表 10-13 单电阻空间矢量脉宽调制执行 API

标题	描述
接口形式	void R1SVPWM_Exec(R1SVPWM_Handle *r1svHandle, const AlbeAxis *uAlbe, UvwAxis *dutyUvwLeft, UvwAxis *dutyUvwRight)
输入参数	r1svHandle，R1SVPWM_Handle结构体指针。 uAlbe：αβ坐标系下定子电压矢量。 dutyUvwLeft：三相左边沿比较值相对PWM计数周期值的占空比。 dutyUvwRight：三相右边沿比较值相对PWM计数周期值的占空比。



标题	描述
返回参数	r1svHandle，R1SVPWM_Handle结构体指针。 uAlbe：αβ坐标系下定子电压矢量。 dutyUvwLeft：三相左边沿比较值相对PWM计数周期值的占空比。 dutyUvwRight：三相右边沿比较值相对PWM计数周期值的占空比。
返回值	无。

10.2.2.3.2 API 功能描述

此API用于单电阻采样时将定子电压矢量转化为三相PWM信号的比较值。

10.2.2.3.3 Example

```
R1SVPWM_Handle r1Sv = {0};
AlbeAxis vab = {0};
UvwAxis dutyUvwA = {0};
UvwAxis dutyUvwB = {0};
vab.alpha = 50.0f;
vab.beta = 10.0f
R1SVPWM_Exec(&r1Sv, &vab, &dutyUvwA, &dutyUvwB);
```

10.2.2.3.4 注意事项

- 实际应用中需要注意PWM波形的产生方式与本模块默认方式是否一致。
- APT模块左右比较值不一定相同，比较值等于dutyUvwA某一项分量乘以APT计数器周期值。

10.2.2.4 R1SVPWM_PhaseShift

10.2.2.4.1 API 形式

表 10-14 单电阻非观测区移相控制 API

标题	描述
接口形式	void R1SVPWM_PhaseShift(R1SVPWM_CALC_Handle *r1SvCalc, float sampleWindow)
输入参数	r1SvCalc，R1SVPWM_CALC_Handle结构体指针。 sampleWindow：移相采样窗口。
返回参数	无。
返回值	无。



10.2.2.4.2 API 功能描述

此API用于单电阻采样在非观测区时通过移相创造采样窗口，从而实现相电流的有效采样。

10.2.2.4.3 Example

```
R1SVPWM_Handle r1Sv = {0};
R1SVPWM_CALC_Handle r1SvCalc = {0};
r1Sv.sampleWindow = 0.06f;
R1SVPWM_PhaseShift(&r1SvCalc, r1Sv.sampleWindow);
```

10.2.2.4.4 注意事项

最小采样窗口可以表示为 $T_{min}=T_{set}+T_{conv}+T_d$ 。功率开关管在开通关断时刻电流会产生振荡，需要等待电流稳定后再进行采样，电流振荡时间为 T_{set} ；ADC采样不是瞬间完成，需要一定时间完成采样保持，采样时间为 T_{conv} ；为避免上下管直通，需要设置一定的死区时间，死区时间为 T_d 。采样窗口单位为占空比。

10.2.2.5 R1CurrReconstruct

10.2.2.5.1 API 形式

表 10-15 单电阻电流重构方法 API

标题	描述
接口形式	void R1CurrReconstruct(unsigned int sectorIndex, float currSocA, float currSocB, UvwAxis *curr)
输入参数	sectorIndex：上一拍电压矢量扇区号。 currSocA：SocA点电流采样值。 currSocB：SocB点电流采样值。 curr：三相定子电流重构结果。
返回参数	curr：三相定子电流重构结果。
返回值	无。

10.2.2.5.2 API 功能描述

此API用于将分时采样得到的母线电流重构为电机三相定子电流。

10.2.2.5.3 Example

```
R1SVPWM_Handle r1Sv = {0};
float socA = GetBusCurrA();
float socB = GetBusCurrB();
UvwAxis currUvw = {0};
R1CurrReconstruct(r1Sv.voltIndexLast, socA, socB, &currUvw);
```




10.2.2.5.4 注意事项

- 在执行电流重构计算之前，需要对速度控制器的对象进行初始化。
- 需要配合R1SVPWM_Exec方法使用。



11 observer 观测器模块

11.1 一阶滑模位置观测器

基于U-I模型观测电机转子位置和速度。

11.1.1 一阶滑模位置观测器 handle 结构体说明

用于保存一阶滑模观测器参数的结构体，具体的定义如[表11-1](#)所示。一阶滑模观测器的用户配置参数如[表11-2](#)所示。

表 11-1 FOSMO_Handle 成员变量

成员类型	成员名	成员描述
float	ts	位置观测器执行周期，单位：s。
float	a1	SMO微分方程系数1。
float	a2	SMO微分方程系数2。
float	kSmo	滑模增益（应满足李雅普诺夫稳定性定理）。
float	lambda	滤波系数（低通滤波器截止频率/电频率）。
float	emfLpfMinFreq	低通滤波器最小截止频率，单位：Hz。
float	pllBdw	PLL的带宽，单位：Hz。
float	fcLpf	速度滤波器的截止频率，单位：Hz。
float	filCompAngle	低通滤波器相位补偿角度，单位：rad。
float	elecAngle	SMO算法得到的电角度，单位：rad。



成员类型	成员名	成员描述
float	spdEst	SMO算法观测到的速度，单位：Hz。
MOTOR_Param *	mtrParam	电机参数结构体指针。
AlbeAxis	emfEstUnFil	基于SMO微分方程得到的反电动势。
AlbeAxis	ialbeEst	$\alpha\beta$ 轴电流估计值。
AlbeAxis	ialbeEstLast	上一拍 $\alpha\beta$ 轴电流估计值。
AlbeAxis	emfEstFil	经过低通滤波器后的反电动势。
PLL_Handle	pll	锁相环结构体。
FOFLT_Handle	spdFilter	速度信号的低通滤波器。

表 11-2 FOSMO_Param 一阶滑模用户配置参数

成员类型	成员名	成员描述
float	gain	滑模增益。
float	lambda	滤波系数（低通滤波器截止频率/电频率）。
float	fcEmf	反电动势的滤波截止频率，单位：Hz。
float	pllBdw	PLL带宽，单位：Hz。
float	fcLpf	速度滤波器的截止频率，单位：Hz。

11.1.2 API 接口

表 11-3 接口列表

函数名称	功能描述
FOSMO_Init	初始化滑模观测器。
FOSMO_Clear	滑模位置观测器历史值清除为0。
FOSMO_Exec	滑模位置观测器执行方法。
FOSMO_ParamUpdate	更新滑模观测器的参数。
FOSMO_SetTs	设置滑模观测器的计算周期。



函数名称	功能描述
FOSMO_SetLambda	设置滑模观测器的截止频率系数。

11.1.2.1 FOSMO_Init

11.1.2.1.1 API 形式

表 11-4 一阶滑模位置观测器初始化 API

标题	描述
接口形式	void FOSMO_Init(FOSMO_Handle *fosmo, FOSMO_Param foSmoParam, MOTOR_Param *mtrParam, float ts)
输入参数	fosmo: 滑模位置观测器结构体指针。 foSmoParam: 滑模观测器参数。 mtrParam: 电机参数结构体指针。 ts: 控制周期, 单位: s。
返回参数	fosmo: 滑模位置观测器结构体指针。
返回值	无。

11.1.2.1.2 API 功能描述

此API用于电机启动前初始化滑模位置观测器的所有成员变量。

11.1.2.1.3 Example

```
FOSMO_Handle fosmo = {0};
FOSMO_Param foSmoParam = {0}; /* 此处省略为0, 需用户进行参数配置 */
MOTOR_Param mtrParam = {0}; /* 此处省略为0, 需用户进行参数配置 */
float ts = 0.0001f;
FOSMO_Init(&fosmo, foSmoParam, &mtrParam, ts); /* 初始化滑模位置观测器 */
```

11.1.2.1.4 注意事项

在每次电机启动前需要执行该函数，进行参数初始化。

11.1.2.2 FOSMO_Exec



11.1.2.2.1 API 形式

表 11-5 滑模位置观测器执行方法 API

标题	描述
接口形式	void FOSMO_Exec(FOSMO_Handle *fosmo, const AlbeAxis *ialbeFbk, const AlbeAxis *valbeRef, float refHz)
输入参数	fosmo: 滑模位置观测器结构体指针。 ialbeFbk: $\alpha\beta$ 坐标系下定子电流矢量。 valbeRef: $\alpha\beta$ 坐标系下定子电压矢量。 refHz: 速度指令值, 单位: Hz。
返回参数	fosmo: 滑模位置观测器结构体指针。
返回值	无。

11.1.2.2.2 API 功能描述

此API用于获取电机转子位置和速度信息。

11.1.2.2.3 Example

```
float spdRefHz = GetSpdRef(); /* 获取速度指令值 */
AlbeAxis curr; /* 用户需配置电流值 */
AlbeAxis volt; /* 用户需配置电压值 */
FOSMO_Handle fosmo = {0};
/* 前置条件: 已完成观测器初始化 */
FOSMO_Exec(&fosmo, &curr, &volt,
spdRefHz
); /* 执行滑模位置观测器对的计算 */
```

11.1.2.2.4 注意事项

- 滑模位置观测器的计算需要在固定周期的中断服务程序中执行。
- 在执行滑模位置观测器计算之前, 需要对滑模位置观测器的对象进行初始化。

11.1.2.3 FOSMO_ParamUpdate

11.1.2.3.1 API 形式

表 11-6 滑模位置观测器参数更新 API

标题	描述
接口形式	void FOSMO_ParamUpdate(FOSMO_Handle *fosmo, float gain, float pllBdw, float fc);



标题	描述
输入参数	fosmo: 滑模位置观测器结构体指针。 gain: 滑模增益。 pllBdw: PLL带宽, 单位: Hz。 fc: 速度滤波器截止频率, 单位: Hz。
返回参数	fosmo: 滑模位置观测器结构体指针。
返回值	无。

11.1.2.3.2 API 功能描述

此API用于更新一阶滑模观测器的滑模增益、PLL带宽、速度滤波器截止频率等参数。

11.1.2.3.3 Example

```
FOSMO_Handle fosmo = {0};  
/* 前置条件: 已完成观测器初始化 */  
  
float gain = 2.0f;  
float pllBdw = 100.0f;  
float fc = 40.0f;  
FOSMO_ParamUpdate(&fosmo, gain, pllBdw, fc);
```

11.1.2.3.4 注意事项

更新参数一定要使用该API, 切勿手动修改FOSMO_Handle结构体内的成员变量值, 否则会导致中间计算值错误。

11.1.2.4 FOSMO_Clear

11.1.2.4.1 API 形式

表 11-7 清除滑模位置观测器变量历史值 API

标题	描述
接口形式	void FOSMO_Clear(FOSMO_Handle *fosmo)
输入参数	fosmo: 滑模位置观测器结构体指针。
返回参数	fosmo: 滑模位置观测器结构体指针。
返回值	无。

11.1.2.4.2 API 功能描述

此API用于电机启动前清除滑模位置观测器的所有成员变量历史值。



11.1.2.4.3 Example

```
FOSMO_Handle smo = {0};  
/* 前置条件：已完成观测器初始化 */  
  
FOSMO_Clear(&smo); /* 清除滑模位置观测器的历史信息 */
```

11.1.2.4.4 注意事项

在每次电机启动前需要执行该函数，清除历史变量的影响。

11.1.2.5 FOSMO_SetTs

11.1.2.5.1 API 形式

表 11-8 设置滑模位置观测器控制周期 API

标题	描述
接口形式	void FOSMO_SetTs(FOSMO_Handle *fosmo, float ts)
输入参数	fosmo：滑模位置观测器结构体指针。 ts：控制周期。
返回参数	fosmo：滑模位置观测器结构体指针。
返回值	无。

11.1.2.5.2 API 功能描述

此API用于设置滑模观测器控制周期。

11.1.2.5.3 Example

```
FOSMO_Handle fosmo = {0};  
/* 前置条件：已完成观测器初始化 */  
  
float ts = 0.0001f;  
FOSMO_SetTs(&fosmo, ts);
```

11.1.2.5.4 注意事项

无。

11.1.2.6 FOSMO_SetLambda



11.1.2.6.1 API 形式

表 11-9 设置滑模位置观测器滤波系数 API

标题	描述
接口形式	void FOSMO_SetLambda(FOSMO_Handle *fosmo, float lambda)
输入参数	fosmo：滑模位置观测器结构体指针。 lambda：滤波系数。
返回参数	fosmo：滑模位置观测器结构体指针。
返回值	无。

11.1.2.6.2 API 功能描述

此API用于设置滑模观测器滤波截止频率，其值 = lambda * 观测的电角速度。

11.1.2.6.3 Example

```
FOSMO_Handle fosmo = {0};
/* 前置条件：已完成观测器初始化 */
float lambda = 2.5f;
FOSMO_SetLambda(&fosmo, lambda);
```

11.1.2.6.4 注意事项

合法的lambda值应为非负浮点数。

11.2 四阶滑模位置观测器

观测电机转子位置和速度。

11.2.1 四阶滑模位置观测器 handle 结构体说明

用于保存四阶滑模观测器参数的结构体，具体的定义如表11-10所示。四阶滑模观测器的用户配置参数如表11-11所示。

表 11-10 SMO4TH_Handle 四阶滑模位置观测器结构体说明

成员类型	成员名	成员描述
float	ts	位置观测器执行周期，单位：s。
float	fcLpf	速度一阶LPF的截止频率（Hz）。
MOTOR_Param *	mtrParam	电机参数结构体指针。
AlbeAxis	ialbeEst	αβ轴电流估计值。
AlbeAxis	ealbeEst	αβ轴反电势估计值。



成员类型	成员名	成员描述
PLL_Handle	pll	锁相环结构体。
float	elecAngle	四阶SMO算法得到的电角度，float格式，单位：rad/s。
FOFLT_Handle	spdFilter	速度信号的低通滤波器。
float	spdEst	四阶SMO算法得到的观测速度，单位：Hz。
float	kd	观测器d轴增益。
float	kq	观测器q轴增益。
float	pllBdw	锁相环带宽，单位：Hz。

表 11-11 SMO4TH_Param 四阶滑模用户配置参数

成员类型	成员名	成员描述
float	kd	d轴滑模增益。
float	kq	q轴滑模增益。
float	pllBdw	PLL带宽，单位：Hz。
float	fcLpf	速度滤波器的截止频率，单位：Hz。

11.2.2 API 接口

表 11-12 接口列表

函数名称	功能描述
SMO4TH_Init	四阶滑模位置观测器初始化。
SMO4TH_Clear	四阶滑模位置观测器历史值清零。
SMO4TH_ParamUpdate	四阶滑模位置观测器参数更新。
SMO4TH_Exec	四阶滑模位置观测器执行方法。
SMO4TH_SetTs	四阶滑模位置观测器控制周期设置。

11.2.2.1 SMO4TH_Init



11.2.2.1.1 API 形式

表 11-13 四阶滑模位置观测器初始化 API

标题	描述
接口形式	void SMO4TH_Init(SMO4TH_Handle *smo4th, SMO4TH_Param smo4thParam, MOTOR_Param *mtrParam, float ts)
输入参数	smo4th: 四阶滑模位置观测器结构体指针。 smo4thParam: 四阶滑模用户配置参数。 mtrParam: 电机参数结构体指针。 ts: 滑模观测器执行周期, 单位: s。
返回参数	smo4th: 四阶滑模位置观测器结构体指针。
返回值	无。

11.2.2.1.2 API 功能描述

此API用于电机启动前初始化四阶滑模位置观测器参数。

11.2.2.1.3 Example

```
SMO4TH_Handle smo4th = {0};  
  
SMO4TH_Param smo4thParam = {...}; /* 此处省略, 用户自行配置参数 */  
MOTOR_Param mtrParam = {...}; /* 此处省略, 由用户自行配置参数 */  
float ts = 0.0001f;  
SMO4TH_Init(&smo4th, smo4thParam, &mtrParam, ts);
```

11.2.2.1.4 注意事项

无。

11.2.2.2 SMO4TH_Clear

11.2.2.2.1 API 形式

表 11-14 四阶滑模位置观测器历史值清除 API

标题	描述
接口形式	void SMO4TH_Clear(SMO4TH_Handle *smo4th)
输入参数	smo4th: 四阶滑模观测器结构体指针。
返回参数	smo4th: 四阶滑模观测器结构体指针。
返回值	无。



11.2.2.2.2 API 功能描述

此API用于在电机启动前清除四阶滑模位置观测器具有历史信息的成员变量。

11.2.2.2.3 Example

```
SMO4TH_Handle smo4th = {0};
/* 前置条件：已完成观测器初始化 */
SMO4TH_Clear(&smo4th); /* 清除四阶滑模位置观测器的历史信息 */
```

11.2.2.2.4 注意事项

在每次电机启动前需要执行该函数，清除历史变量的影响。

11.2.2.3 SMO4TH_ParamUpdate

11.2.2.3.1 API 形式

表 11-15 四阶滑模位置观测器参数更新 API

标题	描述
接口形式	void SMO4TH_ParamUpdate(SMO4TH_Handle * smo4th float kd, float kq, float pllBdw, float fc)
输入参数	smo4th：四阶滑模位置观测器结构体指针。 kd：d轴增益。 kq：q轴增益。 pllBdw：锁相环带宽。 fc：速度低通滤波器截止频率。
返回参数	smo4th：四阶滑模位置观测器结构体指针。
返回值	无。

11.2.2.3.2 API 功能描述

此API用于更新四阶滑模位置观测器参数，包括：d轴增益、q轴增益、锁相环带宽和速度低通滤波器截止频率。

11.2.2.3.3 Example

```
SMO4TH_Handle smo4th = {0};
/* 前置条件：已完成观测器初始化 */

float kd = 300.0f; /* d轴增益 */
float kq = 2000.0f; /* q轴增益 */
float pllBdw = 30.0f; /* 锁相环带宽30.0rad/s */
float fc= 40.0f; /* 截止频率40.0Hz */
SMO4TH_ParamUpdate(&smo4th, kd, kq, pllBdw, fc); /* 设置四阶滑模位置观测器参数 */
```



11.2.2.3.4 注意事项

无。

11.2.2.4 SMO4TH_Exec

11.2.2.4.1 API 形式

表 11-16 四阶滑模位置观测器执行方法 API

标题	描述
接口形式	void SMO4TH_Exec(SMO4TH_Handle *smo4th, const AlbeAxis *ialbeFbk, const AlbeAxis *valbeRef)
输入参数	smo4th: 四阶滑模位置观测器结构体指针。 ialbeFbk: $\alpha\beta$ 轴电流反馈值。 valbeRef: $\alpha\beta$ 轴电压参考值。
返回参数	smo4th: 四阶滑模位置观测器结构体指针。
返回值	无。

11.2.2.4.2 API 功能描述

此API为四阶滑模执行函数，用于获取电机转子位置和速度信息。

11.2.2.4.3 Example

```
SMO4TH_Handle smo4th = {0};  
/* 前置条件：已完成观测器初始化 */  
  
AlbeAxis ialbeFbk; /*  $\alpha\beta$ 轴电流反馈值 */  
AlbeAxis valbeRef; /*  $\alpha\beta$ 轴电压参考值 */  
SMO4TH_Exec(&smo4th, &ialbeFbk, &valbeRef); /* 执行四阶滑模位置观测器的计算 */
```

11.2.2.4.4 注意事项

- 四阶滑模位置观测器的计算需要在固定周期的中断服务程序中执行。
- 在执行四阶滑模位置观测器计算之前，需要对滑模位置观测器的对象进行初始化。

11.2.2.5 SMO4TH_SetTs

11.2.2.5.1 API 形式

表 11-17 设置四阶滑模位置观测器控制周期 API

标题	描述
接口形式	void SMO4TH_SetTs(SMO4TH_Handle *smo4th, float ts)



标题	描述
输入参数	smo4th：四阶滑模位置观测器结构体指针。 ts：控制周期，单位：s。
返回参数	smo4th：四阶滑模位置观测器结构体指针。
返回值	无。

11.2.2.5.2 API 功能描述

此API为设置四阶滑模控制周期。

11.2.2.5.3 Example

```
SMO4TH_Handle smo4th = {0};  
/* 前置条件：已完成观测器初始化 */  
  
float ts = 0.0002f;  
SMO4TH_SetTs(&smo4th, ts);
```

11.2.2.5.4 注意事项

无。



12 pfc 功率因数校正模块

12.1 PFC 电流控制

12.1.1 PFC 电流控制器 handle 结构体说明

用于保存PFC电流控制器参数的结构体，具体定义如表12-1所示。

表 12-1 PFC_CURRCTRL_Handle 成员变量

成员类型	成员名	成员描述
float	iacRef	电流指令，单位：A。
float	iacFbk	电流反馈，单位：A。
float	vacFbk	电压反馈，单位：V。
float	unitVacFbk	单位化电压反馈。
float	unitCoeff;	输入电压幅值的倒数。
float	maxCurr	最大电流，单位：A。
float	startCurr;	PFC启动电流。
float	stopCurr	PFC停止电流。
float	pwmDuty	PFC电流环输出占空比。
PID_Handle	currPi	电流环PI控制器。



12.1.2 API 接口

表 12-2 接口列表

函数名称	功能描述
PFC_CurrCtrlInit	PFC电流环控制器初始化。
PFC_CurrCtrlClear	PFC电流环控制器清除历史值。
PFC_CurrCtrlExec	PFC电流环控制器执行。

12.1.2.1 PFC_CurrCtrlInit

12.1.2.1.1 API 形式

表 12-3 PFC 电流环初始化 API

标题	描述
接口形式	void PFC_CurrCtrlInit(PFC_CURRCTRL_Handle *currCtrl, PI_Param *piParam, float startCurr, float stopCurr, float vacAmp, float ts)
输入参数	currCtrl: PFC电流控制器结构体指针。 piParam: 电流环PI控制参数。 startCurr: PFC启动电流。 stopCurr: PFC停止电流。 vacAmp: 输入电压幅值。 ts: 控制周期, 单位: s。
返回参数	currCtrl: PFC电流控制器结构体指针。
返回值	无。

12.1.2.1.2 API 功能描述

此API用于PFC电流控制器初始化所有成员变量。

12.1.2.1.3 Example

```
PFC_CURRCTRL_Handle currCtrl = {0};
PI_Param piParam = {1.0f, 1.0f, 0.99f, -0.99f}; /* 电流环PI控制示例参数 */
float startCurr = 10.0f;
float stopCurr = 0.5f;
float vacAmp = 310.0f;
float ts = 0.00001f;
PFC_CurrCtrlInit(&currCtrl, &piParam, startCurr, stopCurr,vacAmp,ts);
```



12.1.2.1.4 注意事项

无。

12.1.2.2 PFC_CurrCtrlClear

12.1.2.2.1 API 形式

表 12-4 清除 PFC 电流环历史值 API

标题	描述
接口形式	void PFC_CurrCtrlClear(PFC_CURRCTRL_Handle *currCtrl)
输入参数	currCtrl: PFC电流控制器结构体指针。
返回参数	currCtrl: PFC电流控制器结构体指针。
返回值	无。

12.1.2.2.2 API 功能描述

此API用于清除PFC电流控制器所有具有历史值的成员变量。

12.1.2.2.3 Example

```
PFC_CURRCTRL_Handle currCtrl = {0};
/* 前置条件：已完成PFC电流控制器初始化。*/

PFC_CurrCtrlClear(&currCtrl);
```

12.1.2.2.4 注意事项

在每次启动PFC前需要执行该函数，清除历史变量的影响。

12.1.2.3 PFC_CurrCtrlExec

12.1.2.3.1 API 形式

表 12-5 PFC 执行电流环控制 API

标题	描述
接口形式	void PFC_CurrCtrlExec(PFC_CURRCTRL_Handle *currCtrl, float iacFbk)
输入参数	currCtrl: PFC电流控制器结构体指针。 iacFbk: 电流反馈值。
返回参数	currCtrl: PFC电流控制器结构体指针。
返回值	无。



12.1.2.3.2 API 功能描述

此API用于PFC电流环的执行，使网侧电流跟随电压同相位正弦波动。

12.1.2.3.3 Example

```
PFC_CURRCTRL_Handle currCtrl = {0};
/* 前置条件：已完成PFC电流控制器初始化。 */

float iacFbk = 1.5f; /* 1.5示例参数，使用时请使用电感电流值。 */
PFC_CurrCtrlExec(&currCtrl, iacFbk);
```

12.1.2.3.4 注意事项

在执行该函数前，需要先对PFC电流控制进行初始化，初始化代码需要用户自定义在应用层。

12.2 PFC 电压控制

12.2.1 PFC 电压控制器 handle 结构体说明

用于保存PFC电压控制器参数的结构体，具体定义如表12-6所示。

表 12-6 PFC_VOLTCTRL_Handle 成员变量

成员类型	成员名	成员描述
float	vdcRef	电压指令，单位：V。
float	vdcFbk	电压反馈，单位：V。
float	iamp	电压环输出，即为电流幅值，单位：A。
PID_Handle	voltPi	电压环PI控制器。

12.2.2 API 接口

表 12-7 接口列表

函数名称	功能描述
void PFC_VoltCtrlInit	PFC电压环初始化。
PFC_VoltCtrlExec	PFC电压环控制执行。
PFC_VoltCtrlClear	PFC电压环控制清除历史值。

12.2.2.1 PFC_VoltCtrlInit



12.2.2.1.1 API 形式

表 12-8 PFC 电压环初始化 API

标题	描述
接口形式	void PFC_VoltCtrlInit(PFC_VOLTCTRL_Handle *voltCtrl, PI_Param *piParam, float vdcRef, float ts)
输入参数	voltCtrl: PFC电压控制结构体指针; piParam: PFC电压环PI控制器参数指针; vdcRef: PFC的指令电压; ts: 控制周期。
返回参数	voltCtrl: PFC电压环控制结构体指针。
返回值	无。

12.2.2.1.2 API 功能描述

此API用于PFC电压环的初始化。

12.2.2.1.3 Example

```
PFC_VOLTCTRL_Handle voltCtrl = {0}; /* 示例 */
PI_Param piParam = {1.0f, 1.0f, 5.0f, 0.0f}; /* 示例 */
float vdcRef = 370.0f;
float ts = 0.00001f;
PFC_VoltCtrlInit(&voltCtrl, &piParam, vdcRef, ts);
```

12.2.2.1.4 注意事项

无。

12.2.2.2 PFC_VoltCtrlExec

12.2.2.2.1 API 形式

表 12-9 执行 PFC 电压环控制 API

标题	描述
接口形式	void PFC_VoltCtrlExec(PFC_VOLTCTRL_Handle *voltCtrl, float vdcFbk)
输入参数	voltCtrl: PFC电压控制结构体指针; vdcFbk: 母线电压。
返回参数	voltCtrl: PFC电压控制结构体指针。
返回值	无。



12.2.2.2.2 API 功能描述

此API用于PFC电压环的执行，通过PI控制器使母线电压升高到指定值。

12.2.2.2.3 Example

```
PFC_VOLTCTRL_Handle voltCtrl = {0};
float vdcFbk = 360.0f; /* 根据实际采样值传入 */
PFC_VoltCtrlExec(&voltCtrl, vdcFbk);
```

12.2.2.2.4 注意事项

在执行该函数前，需要先对PFC电压控制进行初始化，初始化代码需要用户自定义在应用层。

12.2.2.3 PFC_VoltCtrlClear

12.2.2.3.1 API 形式

表 12-10 PFC 清除电压环历史值 API

标题	描述
接口形式	void PFC_VoltCtrlClear(PFC_VOLTCTRL_Handle *voltCtrl)
输入参数	voltCtrl：PFC电压控制结构体指针。
返回参数	voltCtrl：PFC电压控制结构体指针。
返回值	无。

12.2.2.3.2 API 功能描述

此API用于清除PFC电压控制器所有具有历史值的成员变量。

12.2.2.3.3 Example

```
PFC_VOLTCTRL_Handle voltCtrl = {0};
PFC_VoltCtrlClear(&voltCtrl);
```

12.2.2.3.4 注意事项

在每次启动PFC前需要执行该函数，清除历史变量的影响。



13 pid 控制器模块

13.1 PID 控制器

通用PID控制器，用于对指令信号的闭环计算，结构如下所示（ s 为微分算子）。

$$PID=k_p*error+\frac{k_i}{s}*error+k_d*s*error$$

13.1.1 PID 控制器 handle 结构

用于保存PID控制器参数及历史值的结构体，具体定义如[表13-1](#)所示。

其中，用户可以根据使用控制器类型调用不同的用户参数结构体，如[表13-2](#)和[表13-3](#)所示。

表 13-1 PID_Handle 结构体说明

成员类型	成员名	成员描述
float	error	控制误差值。
float	errorLast	控制误差的历史值。
float	fbkVal	当前反馈值。
float	fbkValLast	上一次的反馈值。
float	feedforward	控制器前馈输入值。
float	integral	积分器历史值。
float	saturation	饱和值历史值，饱和值为限幅前输出值与限幅后输出值之差。
float	differ	微分器历史值。
float	kp	比例增益系数。



成员类型	成员名	成员描述
float	ki	积分增益系数。
float	kd	微分增益系数。
float	ns	微分滤波系数。
float	ka	抗饱和增益系数。
float	ts	控制周期。
float	upperLimit	输出限幅的上限值。
float	lowerLimit	输出限幅的下限值。

表 13-2 PI 控制器用户参数 PI_Param

成员类型	成员名	成员描述
float	kp	比例增益系数。
float	ki	积分增益系数。
float	upperLim	输出限幅的上限值。
float	lowerLim	输出限幅的下限值。

表 13-3 PID 控制器用户参数 PID_Param

成员类型	成员名	成员描述
float	kp	比例增益系数。
float	ki	积分增益系数。
float	kd	微分增益系数。
float	ns	微分滤波系数。
float	ka	抗饱和增益系数。
float	saturation	饱和值历史值，饱和值为限幅前输出值与限幅后输出值之差。
float	upperLim	输出限幅的上限值。
float	lowerLim	输出限幅的下限值。



13.1.2 API 接口

表 13-4 接口列表

函数名称	功能描述
PID_Reset	重置控制器。
PID_Clear	清零控制器历史值。
PI_Exec	执行PID控制器中PI计算功能。
PID_ExecDiffWithFbk	执行带抗反馈值扰动的PID控制器计算功能。
PID_Exec	执行PID控制器中PID计算功能。
PID_SetKp	设置kp参数。
PID_SetKi	设置ki参数。
PID_SetKd	设置kd参数。
PID_SetNs	设置ns参数。
PID_SetTs	设置ts参数。
PID_SetLimit	设置upper Limit和lower Limit参数。

13.1.2.1 PID_Reset

13.1.2.1.1 API 形式

表 13-5 PID 控制器重置 API

标题	描述
接口形式	void PID_Reset(PID_Handle *pidHandle)
输入参数	pidHandle: PID_Handle类型指针，指向PID控制器结构体变量。
返回参数	pidHandle结构体。
返回值	无。

13.1.2.1.2 API 功能描述

清零PID控制器所有参数。



13.1.2.1.3 Example

```
PID_Handle pid = {0}; /* 定义PID控制器结构体变量 */
PID_Reset(&pid); /* 重置PID结构体变量 */
```

13.1.2.1.4 注意事项

无。

13.1.2.2 PID_Clear

13.1.2.2.1 API 形式

表 13-6 清除 PID 控制器历史值 API

标题	描述
接口形式	void PID_Clear(PID_Handle *pidHandle)
输入参数	pidHandle: PID_Handle类型指针，指向PID控制器结构体变量。
返回参数	pidHandle结构体。
返回值	无。

13.1.2.2.2 API 功能描述

清零PID结构体变量内部历史值。

13.1.2.2.3 Example

```
PID_Handle pid = {0}; /* 定义PID控制器结构体变量 */
PID_Clear(&pid); /* 清零PID结构体变量内部历史值 */
```

13.1.2.2.4 注意事项

无。

13.1.2.3 PI_Exec

13.1.2.3.1 API 形式

表 13-7 执行 PID 控制器 PI 计算功能 API

标题	描述
接口形式	float PI_Exec(PID_Handle *pidHandle)
输入参数	pidHandle: PID_Handle类型指针，指向PID控制器结构体变量。
返回参数	pidHandle结构体。



标题	描述
返回值	PID控制器输出的控制量。

13.1.2.3.2 API 功能描述

执行PID控制器的PI计算功能，包含比例环节、积分环节和输出限幅环节，返回控制器输出的控制量。

13.1.2.3.3 Example

```
PID_Handle pidHandle = {0};
pidHandle.ts = 0.0005f;
pidhandle.kp = 0.1f;
pidhandle.ki = 0.5f;
pidhandle.upperLim = 5.0f;
pidhandle.lowerLim = -5.0f;
pidhandle.error = 1.0f;
float ret = PI_Exec(&pidHandle);
```

13.1.2.3.4 注意事项

合法的kp和ki是正浮点数，ki中没有包含PID控制器的计算周期信息。

13.1.2.4 PID_Exec

13.1.2.4.1 API 形式

表 13-8 执行 PID 控制器 PID 计算功能 API

标题	描述
接口形式	float PID_Exec(PID_Handle *pidHandle)
输入参数	pidHandle: PID_Handle 类型指针，指向PID控制器结构体变量。
返回参数	pidHandle结构体。
返回值	PID控制器输出的控制量。

13.1.2.4.2 API 功能描述

执行PID控制器的PID计算功能，包含比例环节、积分环节、微分环节、抗饱和环节和输出限幅环节，返回控制器输出的控制量。

13.1.2.4.3 Example

```
PID_Handle pid = {0}; /* 定义PID控制器结构体变量 */
PID_Reset(&pid); /* 重置PID结构体变量所有参数 */
pid.kp = 10.0f; /* 设置kp参数 */
pid.ki = 20.0f; /* 设置ki参数 */
```




```
pid.kd = 1.0f; /* 设置kd参数 */
pid.ns = 100.0f; /* 设置ns参数 */
pid.ka = 0.1f; /* 设置ka参数 */
pid.upperLimit = 10.0f; /* 设置upperLimit参数 */
pid.lowerLimit = -10.0f; /* 设置lowerLimit参数 */
pid.error = 1.0f; /* 向PID控制器输入误差信号 */
float ref; /* 定义存放控制量的变量 */
ref = PID_Exec(&pid); /* 执行PID控制器的PI计算功能，返回控制量 */
```

13.1.2.4.4 注意事项

合法的kp、ki、kd、ns和ka是正浮点数，ki中没有包含PID控制器计算周期信息。

13.1.2.5 PID_ExecDiffWithFbk

13.1.2.5.1 API 形式

表 13-9 执行 PID 控制器 PID 计算功能 API

标题	描述
接口形式	float PID_ExecDiffWithFbk(PID_Handle *pidHandle)
输入参数	pidHandle: PID_Handle 类型指针，指向PID控制器结构体变量。
返回参数	pidHandle结构体。
返回值	PID控制器输出的控制量。

13.1.2.5.2 API 功能描述

执行带抗反馈值扰动的PID控制器计算功能，包含比例环节、积分环节、微分环节、抗饱和环节和输出限幅环节，微分环节输入为反馈值的偏差，用于抗反馈值扰动，计算结果返回控制器输出的控制量。

13.1.2.5.3 Example

```
PID_Handle pid = {0}; /* 定义PID控制器结构体变量 */
PID_Reset(&pid); /* 重置PID结构体变量所有参数 */
pid.kp = 10.0f; /* 设置kp参数 */
pid.ki = 20.0f; /* 设置ki参数 */
pid.kd = 1.0f; /* 设置kd参数 */
pid.ns = 100.0f; /* 设置ns参数 */
pid.ka = 0.1f; /* 设置ka参数 */
pid.upperLimit = 10.0f; /* 设置upperLimit参数 */
pid.lowerLimit = -10.0f; /* 设置lowerLimit参数 */
pid.error = 1.0f; /* 向PID控制器输入误差信号 */
pid.fbkVal = 1.0f; /* 向PID控制器输入当前反馈值信号 */
float ref; /* 定义存放控制量的变量 */
ref = PID_ExecDiffWithFbk(&pid); /* 执行PID控制器的PI计算功能，返回控制量 */
```

13.1.2.5.4 注意事项

合法的kp、ki、kd、ns和ka是正浮点数，ki中没有包含PID控制器计算周期信息，微分环节不是输入误差，而是两次反馈结果的差值，用于抗反馈值扰动。



13.1.2.6 PID_SetKp

13.1.2.6.1 API 形式

表 13-10 PID 比例增益系数设置 API

标题	描述
接口形式	void PID_SetKp(PID_Handle *pidHandle, float kp)
输入参数	pidHandle: PID_Handle 类型指针，指向PID控制器结构体变量。 kp: 比例增益系数目标设置值。
返回参数	比例增益系数。
返回值	无。

13.1.2.6.2 API 功能描述

设置PID控制器的比例增益系数。

13.1.2.6.3 Example

```
PID_Handle pid = {0}; /* 定义PID控制器结构体变量 */
PID_SetKp(&pid, 1.0f); /* 将PID控制器的比例增益系数设置为1.0f */
```

13.1.2.6.4 注意事项

合法的kp是正浮点数。

13.1.2.7 PID_SetKi

13.1.2.7.1 API 形式

表 13-11 PID 积分增益系数设置 API

标题	描述
接口形式	void PID_SetKi(PID_Handle *pidHandle, float ki)
输入参数	pidHandle: PID_Handle 类型指针，指向PID控制器结构体变量。 ki: 积分增益系数目标设置值。
返回参数	积分增益系数。
返回值	无。



13.1.2.7.2 API 功能描述

设置PID控制器的积分增益系数。

13.1.2.7.3 Example

```
PID_Handle pid = {0}; /* 定义PID控制器结构体变量 */
PID_SetKi(&pid, 1.0f); /* 将PID控制器的积分增益系数设置为1.0f */
```

13.1.2.7.4 注意事项

合法的ki是正浮点数。

13.1.2.8 PID_SetKd

13.1.2.8.1 API 形式

表 13-12 PID 微分增益系数设置 API

标题	描述
接口形式	void PID_SetKd(PID_Handle *pidHandle, float kd)
输入参数	pidHandle: PID_Handle 类型指针，指向PID控制器结构体变量。 kd: 微分增益系数目标设置值。
返回参数	微分增益系数。
返回值	无。

13.1.2.8.2 API 功能描述

设置PID控制器的微分增益系数。

13.1.2.8.3 Example

```
PID_Handle pid = {0}; /* 定义PID控制器结构体变量 */
PID_SetKd(&pid, 1.0f); /* 将PID控制器的微分增益系数设置为1.0f */
```

13.1.2.8.4 注意事项

合法的kd是非负浮点数。

13.1.2.9 PID_SetNs



13.1.2.9.1 API 形式

表 13-13 PID 差分过滤系数设置 API

标题	描述
接口形式	void PID_SetNs(PID_Handle *pidHandle, float ns)
输入参数	pidHandle: PID_Handle 类型指针，指向PID控制器结构体变量。 ns: 差分项过滤系数。
返回参数	差分过滤系数。
返回值	无。

13.1.2.9.2 API 功能描述

设置PID控制器差分项参数ns的过滤参数。

13.1.2.9.3 Example

```
PID_Handle pid = {0}; /* 定义PID控制器结构体变量 */
PID_SetNs(&pid, 1.0f); /* 将PID控制器的差分过滤系数设置为1.0f */
```

13.1.2.9.4 注意事项

合法的ns是非负浮点数。

13.1.2.10 PID_SetTs

13.1.2.10.1 API 形式

表 13-14 PID 控制周期设置 API

标题	描述
接口形式	void PID_SetTs(PID_Handle *pidHandle, float ts)
输入参数	pidHandle: PID_Handle 类型指针，指向PID控制器结构体变量。 ts: 微分增益系数目标设置值。
返回参数	控制周期。
返回值	无。

13.1.2.10.2 API 功能描述

设置PID控制器的控制周期。



13.1.2.10.3 Example

```
PID_Handle pid = {0}; /* 定义PID控制器结构体变量 */
PID_SetTs(&pid, 0.0002f); /* 将PID控制器的控制周期设置为0.0002f */
```

13.1.2.10.4 注意事项

合法的Ts是非负浮点数。

13.1.2.11 PID_SetLimit

13.1.2.11.1 API 形式

表 13-15 PID 上、下限幅值设置 API

标题	描述
接口形式	void PID_SetLimit(PID_Handle *pidHandle, float limit)
输入参数	pidHandle: PID_Handle 类型指针，指向PID控制器结构体变量。 limit: 控制器上、下限幅目标设置值。
返回参数	PID控制器上、下限幅值。
返回值	无。

13.1.2.11.2 API 功能描述

设置PID控制器的PID上、下限幅值。

13.1.2.11.3 Example

```
PID_Handle pid = {0}; /* 定义PID控制器结构体变量 */
PID_SetLimit(&pid, 10.0f); /* 将PID控制器的输出上、下限幅值设置为10.0f 和-10.0f */
```

13.1.2.11.4 注意事项

合法的limit是大于零的浮点数。



14 power 模块

14.1 功率管理

该模块使用电机d、q轴电流和电压，对电机的平均功率进行计算。

14.1.1 功率计算 handle 结构体说明

用于保存功率管理的结构体，具体定义如[表14-1](#)所示。

表 14-1 POWER_Handle 结构体说明

成员类型	成员名	成员描述
float	avgPower	平均功率。
float	fltCoeff	功率滤波系数。
DqAxis *	idqFbk	dq轴电流。
DqAxis *	vdqRef	dq轴电压。

14.1.2 API 接口

表 14-2 接口列表

函数名称	功能描述
MotorPowerInit	电机功率计算初始化。
MotorPowerCalc	电机平均功率计算。
MotorPowerClear	清除电机功率计算的历史值。



14.1.2.1 MotorPowerInit

14.1.2.1.1 API 形式

表 14-3 功率计算初始化 API

标题	描述
接口形式	void MotorPowerInit(POWER_Handle *avgPower, DqAxis *vdqRef, DqAxis *idqFbk, float fltCoeff)
输入参数	avgPower: 平均功率计算的结构体指针。 vdqRef: dq轴电压。 idqFbk: dq轴电流。 fltCoeff: 滤波系数。
返回参数	平均功率avgPower结构体。
返回值	无。

14.1.2.1.2 API 功能描述

初始化平均功率计算的相关参数。

14.1.2.1.3 Example

```
POWER_Handle avgPower;  
DqAxis vdqRef;  
DqAxis idqFbk;  
float fltCoeff = 0.01f;  
MotorPowerInit(&avgPower, &vdqRef, &idqFbk, fltCoeff);
```

14.1.2.1.4 注意事项

功率计算只能在有dq轴坐标系的控制方法中使用。

14.1.2.2 MotorPowerCalc

14.1.2.2.1 API 形式

表 14-4 平均功率计算 API

标题	描述
接口形式	float MotorPowerCalc(POWER_Handle *avgPower)
输入参数	avgPower: 平均功率计算的结构体指针。
返回参数	平均功率avgPower结构体。
返回值	电机的平均功率，单位：W。



14.1.2.2.2 API 功能描述

利用电机的dq轴电流和电压计算电机平均功率。

14.1.2.2.3 Example

```
POWER_Handle avgPower;  
float avgPower = MotorPowerCalc(&avgPower);
```

14.1.2.2.4 注意事项

功率计算只能在有dq轴坐标系的控制方法中使用。

14.1.2.3 MotorPowerClear

14.1.2.3.1 API 形式

表 14-5 平均功率计算清除历史值 API

标题	描述
接口形式	void MotorPowerClear(POWER_Handle *avgPower)
输入参数	avgPower：平均功率计算的结构体指针。
返回参数	平均功率avgPower结构体。
返回值	无。

14.1.2.3.2 API 功能描述

清除平均功率计算的历史值。

14.1.2.3.3 Example

```
POWER_Handle avgPower;  
MotorPowerClear(&avgPower);
```

14.1.2.3.4 注意事项

无。



15 protection 故障检测模块

15.1 开相检测

15.1.1 开相检测 handle 结构体说明

用于保存开相检测的结构体，具体定义如表15-1所示。

表 15-1 OPD_Handle 结构体说明

成员类型	成员名	成员描述
float	minOpenPhsCurr	开相检测的最小电流，单位：A。
bool	isOpenPhsU	U相是否开相。
bool	isOpenPhsV	V相是否开相。
bool	isOpenPhsW	W相是否开相。

六个电流方向的枚举变量定义如表15-2所示。

表 15-2 开相检测的电流方向 OPD_Index 枚举定义

成员类型	成员名	成员描述
enum	OPD_U_V	开相检测的电流方向，由U流向V。
enum	OPD_V_U	开相检测的电流方向，由V流向U。
enum	OPD_V_W	开相检测的电流方向，由V流向W。



成员类型	成员名	成员描述
enum	OPD_W_V	开相检测的电流方向，由W流向V。
enum	OPD_W_U	开相检测的电流方向，由W流向U。
enum	OPD_U_W	开相检测的电流方向，由U流向W。
enum	OPD_END	无。

15.1.2 API 接口

表 15-3 接口列表

函数名称	功能描述
OPD_Init	开相检测初始化。
OPD_Exec	开相检测执行。
OPD_Clear	清除开相检测历史值。

15.1.2.1 OPD_Init

15.1.2.1.1 API 形式

表 15-4 开相检测初始化 API

标题	描述
接口形式	void OPD_Init(OPD_Handle *opd, float minOpenPhsCurr)
输入参数	opd：指向OPD_Handle的结构体指针。 minOpenPhsCurr：判断开相的最小电流。
返回参数	开相检测opd结构体。
返回值	无。

15.1.2.1.2 API 功能描述

初始化开相检测的相关参数。



15.1.2.1.3 Example

```
OPD_Handle opd = {0};
float minCurr = 0.1f;
OPD_Init(&opd, minCurr);
```

15.1.2.1.4 注意事项

无。

15.1.2.2 OPD_Exec

15.1.2.2.1 API 形式

表 15-5 开相检测执行 API

标题	描述
接口形式	bool OPD_Exec(OPD_Handle *opd, const float *iuvw)
输入参数	opd: 指向OPD_Handle的结构体指针。 iuvw: 指向6个方向的相电流指针。
返回参数	开相检测opd结构体。
返回值	是否发生开相。

15.1.2.2.2 API 功能描述

通过检测六个方向的电流，判断电机是否发生开相，以及哪一相开相。

15.1.2.2.3 Example

```
OPD_Handle opd = {0};
float minCurr[6] = {1};
...
/* minCurr数组需保证已获取到电流值 */
bool isOpenPhs = OPD_Exec(&opd, minCurr);
```

15.1.2.2.4 注意事项

使用该方法检测是否开相需要传入6个方向的相电流。

15.1.2.3 OPD_Clear

15.1.2.3.1 API 形式

表 15-6 清除开相检测历史值 API

标题	描述
接口形式	void OPD_Clear(OPD_Handle *opd)



标题	描述
输入参数	opd：指向OPD_Handle的结构体指针。
返回参数	开相检测opd结构体。
返回值	无。

15.1.2.3.2 API 功能描述

清除开相检测的历史值。

15.1.2.3.3 Example

```
OPD_Handle opd = {0};
...
OPD_Clear(&opd);
```

15.1.2.3.4 注意事项

无。

15.2 堵转检测

基于电机速度与电流的综合关系执行电机堵转检测。

15.2.1 堵转检测与保护 handle 结构

用于保存电机堵转检测相关参数的结构体，具体的定义如表15-7所示。

表 15-7 STD_Handle 成员变量

成员类型	成员名	成员描述
float	currAmpLimit	电机堵转检测的电流阈值，单位：A。
float	spdLimit	电机堵转检测的速度阈值，单位：Hz。
float	timeLimit	堵转检测的时间阈值，单位：s。
float	timer	执行堵转保护的时间累计，单位：s。
float	ts	控制周期，单位：s。



15.2.2 API 接口

表 15-8 接口列表

函数名称	功能描述
STD_Init	电机堵转故障检测初始化。
STD_Exec_ByCurrSpd	通过电流和速度进行电机堵转故障检测。
STD_Clear	电机堵转故障历史值状态清零。

15.2.2.1 STD_Init

15.2.2.1.1 API 形式

表 15-9 电机堵转故障检测初始化 API

标题	描述
接口形式	void STD_Init(STD_Handle *stall, float currLimit, float spdLimit, float timeLimit, float ts)
输入参数	stall: 堵转故障检测结构体指针。 currLimit: 触发堵转故障的电流幅值, 单位: A。 spdLimit: 触发堵转的速度幅值, 单位: Hz。 timeLimit: 电流幅值超过限制的阈值时间 (s)。 ts: 控制周期, 单位: s。
返回参数	堵转检测stall结构体。
返回值	无。

15.2.2.1.2 API 功能描述

此API用于电机堵转检测的成员变量的初始化。

15.2.2.1.3 Example

```
STD_Handle stall = {0};
float currLim = 1.0f;
float spdLim = 30.0f;
float timeLim = 0.5f;
float ts = 0.0001f;
STD_Init(&stall, currLim, spdLim, timeLim, ts); /* 初始化堵转检测的成员变量信息。*/
```

15.2.2.1.4 注意事项

调用该函数，初始化堵转检测参数。



15.2.2.2 STD_Exec_ByCurrSpd

15.2.2.2.1 API 形式

表 15-10 堵转检测 API

标题	描述
接口形式	bool STD_Exec_ByCurrSpd(STD_Handle *stall, float spdFbk, float currAmp);
输入参数	stall: 堵转故障与保护结构体指针。 spdFbk: 速度反馈, 单位: Hz。 currAmp: 电流反馈, 单位: A。
返回参数	堵转检测stall结构体。
返回值	是否发生堵转。

15.2.2.2.2 API 功能描述

此API根据电机运行时的速度和电流检测电机是否发生堵转故障。

15.2.2.2.3 Example

```
STD_Handle stall = {0};
float spd = 100.0f;
float currAmp = 1.0f;
bool isStall = STD_Exec_ByCurrSpd(&stall, spd, currAmp);
```

15.2.2.2.4 注意事项

- 堵转检测需要在电流环中断服务程序中执行。
- 在执行堵转检测前, 需要对堵转检测对象进行初始化。

15.2.2.3 STD_Clear

15.2.2.3.1 API 形式

表 15-11 清除堵转故障历史值 API

标题	描述
接口形式	void STD_Clear(STD_Handle *stall)
输入参数	stall: 堵转故障与保护结构体指针。
返回参数	堵转检测stall结构体。
返回值	无。



15.2.2.3.2 API 功能描述

此API用于清除堵转故障历史值。

15.2.2.3.3 Example

```
STD_Handle stall = {0};
STD_Clear(&stall); /* 清除堵转故障历史值。 */
```

15.2.2.3.4 注意事项

无。

15.3 电流不平衡检测

15.3.1 电流不平衡检测 handle 结构体说明

用于保存电流不平衡检测的结构体，具体定义如表15-12所示。

表 15-12 UNBAL_Handle 结构体说明

成员类型	成员名	成员描述
unsigned int	detCnt	电流不平衡检测的计数值。
unsigned int	detCntLimit	判断发生不平衡故障的计数阈值。
unsigned int	integralCnt	累计积分计数值。
unsigned int	timeCnt	堵转检测的时间计数值。
unsigned int	startupCnt	从电机启动到开始检测不平衡的计数间隔。
bool	startFlag	对电流积分开始的标志。
bool	startFlagLast	对电流积分开始的上一次标志。
bool	zeroFlag	电流过零标志。
bool	calFlag	对电流进行有效值计算的标志。
float	unbalDegree	电流不平衡度。
float	unbalDegreeLimit	发生电流不平衡故障的不平衡度阈值。
float	delta	电流过零点判断，当电流小于delta时，判断电流过零。
float	ts	控制周期。
float	ia	A相电流。
float	ib	B相电流。



成员类型	成员名	成员描述
float	ic	C相电流。

15.3.2 API 接口

表 15-13 接口列表

函数名称	功能描述
UNBAL_Init	电机功率计算初始化。
UNBAL_Det	电机平均功率计算。
UNBAL_Clear	清除电机功率计算的历史值。

15.3.2.1 UNBAL_Init

15.3.2.1.1 API 形式

表 15-14 电流不平衡检测初始化 API

标题	描述
接口形式	void UNBAL_Init(UNBAL_Handle *unbal, float currDelta, float timeThr, float unbalDegreeLim, float ts)
输入参数	unbal: 电流不平衡检测的结构体指针。 currDelta: 判断电流过零的阈值, 单位: A。 timeThr: 电流不平衡的检测时间, 单位: s。 unbalDegreeLim: 电流不平衡度阈值。 ts: 检测周期, 单位: s。
返回参数	电流不平衡检测unbal结构体。
返回值	无。

15.3.2.1.2 API 功能描述

初始化电流不平衡检测相关参数。

15.3.2.1.3 Example

```
UNBAL_Handle unbal = {0};
float currDelta = 0.05f;
float timeThr = 1.0f;
float unbalDegreeLim = 0.04f;
```




```
float ts = 0.0001f;
UNBAL_Init(&unbal, currDelta, timeThr, unbalDegreeLim, ts);
```

15.3.2.1.4 注意事项

无。

15.3.2.2 UNBAL_Det

15.3.2.2.1 API 形式

表 15-15 电流不平衡故障检测 API

标题	描述
接口形式	bool UNBAL_Det(UNBAL_Handle *unbal, UvwAxis *iuvwFbk, float unbalFltCoeff)
输入参数	unbal：电流不平衡检测的结构体指针。 iuvwFbk：UVW三相电流。 unbalFltCoeff：不平衡度的滤波系数。
返回参数	电流不平衡检测unbal结构体。
返回值	电机是否发生电流不平衡现象。

15.3.2.2.2 API 功能描述

检测电机相电流是否发生电流不平衡故障。

15.3.2.2.3 Example

```
UNBAL_Handle unbal = {0};
UvwAxis iuvwFbk = {0};
...
float unbalFltCoeff = 0.5f;
bool isUnbal = UNBAL_Det(&unbal, &iuvw, unbalFltCoeff);
```

15.3.2.2.4 注意事项

无。

15.3.2.3 UNBAL_Clear

15.3.2.3.1 API 形式

表 15-16 清除电流不平衡检测的历史值 API

标题	描述
接口形式	void UNBAL_Clear(UNBAL_Handle *unbal)



标题	描述
输入参数	unbal：电流不平衡检测的结构体指针。
返回参数	电流不平衡检测unbal结构体。
返回值	无。

15.3.2.3.2 API 功能描述

清除电流不平衡检测的历史值。

15.3.2.3.3 Example

```
UNBAL_Handle unbal = {0};  
...  
UNBAL_Clear(&unbal);
```

15.3.2.3.4 注意事项

无。



16 ramp 斜坡管理模块

16.1 斜坡管理控制器

产生斜坡信号的控制器，使控制器内部历史值按照设置的斜率变化，以计算目标指令值，本控制器使得阶跃变化的目标指令值经过处理，呈现为按照一定斜率变化的斜坡指令值。

16.1.1 斜坡管理控制器 handle 结构体说明

用于保存斜坡管理控制器参数及历史值的结构体，具体定义如表16-1所示。

表 16-1 RMG_Handle 结构体说明

成员类型	成员名	成员描述
float	delta	斜坡每次输出更新的变化量。
float	yLast	输出历史值。
float	ts	计算周期，单位：s。
float	slope	斜坡斜率值，单位：Hz/s。

16.1.2 API 接口

表 16-2 接口列表

函数名称	功能描述
RMG_Init	初始化斜坡管理控制器。
RMG_Clear	清零斜坡管理控制器历史值。



函数名称	功能描述
RMG_Exec	执行斜坡管理控制器计算。
RMG_SetTs	设置斜坡控制器的控制周期。
RMG_SetSlope	设置斜坡控制器的斜率。

16.1.2.1 RMG_Init

16.1.2.1.1 API 形式

表 16-3 初始化斜坡管理控制器 API

标题	描述
接口形式	void RMG_Init(RMG_Handle *rmg, float ts, float slope)
输入参数	rmg: RMG_Handle类型指针，指向斜坡管理控制器结构体变量。 ts: 斜坡管理控制器计算周期，单位：s。 slope: 斜坡管理控制器斜率，单位：Hz/s。
返回参数	斜坡控制rmg结构体。
返回值	无。

16.1.2.1.2 API 功能描述

初始化斜坡管理控制器参数，包括：清零yLast，设置ts和slope，根据ts和slope设置delta。

16.1.2.1.3 Example

```
RMG_Handle rmg; /* 定义斜坡管理控制器结构体变量 */
RMG_Init(&rmg, 0.0001f, 1.0f); /* 初始化斜坡管理控制器，控制器计算周期为0.0001秒，斜率为1.0f */
```

16.1.2.1.4 注意事项

合法的ts为正浮点数。

16.1.2.2 RMG_Clear



16.1.2.2.1 API 形式

表 16-4 清零斜坡管理控制器历史值 API

标题	描述
接口形式	void RMG_Clear(RMG_Handle *rmg)
输入参数	rmg：RMG_Handle类型指针，指向斜坡管理控制器结构体变量。
返回参数	斜坡控制rmg结构体。
返回值	无。

16.1.2.2.2 API 功能描述

清零斜坡管理控制器历史值。

16.1.2.2.3 Example

```
RMG_Handle rmg; /* 定义斜坡管理控制器结构体变量 */
RMG_Clear(&rmg); /* 斜坡管理控制器清除历史值 */
```

16.1.2.2.4 注意事项

无。

16.1.2.3 RMG_Exec

16.1.2.3.1 API 形式

表 16-5 执行斜坡管理控制器计算功能 API

标题	描述
接口形式	float RMG_Exec(RMG_Handle *rmg, float targetVal)
输入参数	rmg：RMG_Handle类型指针，指向斜坡管理控制器结构体变量。 targetVal：斜坡目标值。
返回参数	斜坡控制rmg结构体。
返回值	斜坡管理控制器输出的斜坡信号值。

16.1.2.3.2 API 功能描述

执行斜坡管理控制器计算功能，使控制器内部历史值按照设置的斜率计算输入值。斜坡管理计算公式：



$$y = \begin{cases} yLast + delta, & y < yLast \\ yLast, & y = yLast \\ yLast - delat, & y > yLast \end{cases}$$

16.1.2.3.3 Example

```
RMG_Handle rmg; /* 定义斜坡管理控制器结构体变量 */
RMG_Init(&rmg, 0.0001f, 1.0f); /* 初始化斜坡管理控制器，控制器计算周期为0.0001秒，斜率为1.0f */
float targetVal = 10.0f; /* 定义目标值为10.0f */
float ret; /* 定义斜坡管理控制器输出值 */
ret = RMG_Exec(&rmg, targetVal); /* 目标值输入到rmg中进行斜坡信号计算，并返回为当前应该所在的斜坡位置 */
```

16.1.2.3.4 注意事项

无。

16.1.2.4 RMG_SetTs

16.1.2.4.1 API 形式

表 16-6 设置斜坡控制器的控制周期 API

标题	描述
接口形式	void RMG_SetTs(RMG_Handle *rmg, float ts)
输入参数	rmg: RMG_Handle类型指针，指向斜坡管理控制器结构体变量。 ts: 要设置的控制周期。
返回参数	控制周期。
返回值	无。

16.1.2.4.2 API 功能描述

设置斜坡控制器的控制周期。

16.1.2.4.3 Example

```
RmgHandle rmg; /* 定义斜坡管理控制器结构体变量 */
float ts = 0.002f;
RMG_SetTs(&rmg, ts);
```

16.1.2.4.4 注意事项

合法的ts应为正浮点数。



16.1.2.5 RMG_SetSlope

16.1.2.5.1 API 形式

表 16-7 设置斜坡管理控制器斜率 API

标题	描述
接口形式	void RMG_SetSlope(RMG_Handle *rmg, float slope)
输入参数	rmg: RMG_Handle类型指针，指向斜坡管理控制器结构体变量。 slope: 斜坡的斜率。
返回参数	斜坡控制器斜率。
返回值	无。

16.1.2.5.2 API 功能描述

设置斜坡控制器的斜率。

16.1.2.5.3 Example

```
RMG_Handle rmg;  
float slope = 2.0f  
RMG_SetSlope(&rmg, slope);
```

16.1.2.5.4 注意事项

无。



17 vf 控制模块

17.1 恒压频比控制器

根据指令速度，产生dq轴电压指令和电流矢量角度信号，用于电机v/f开环控制。

17.1.1 vf 控制器 handle 结构体说明

用于保存vf控制器参数及历史值的结构体，具体定义如表17-1所示。

表 17-1 VF_Handle 结构体说明

成员类型	成员名	成员描述
float	spdCmd	速度目标值，单位：Hz。
float	spdRef	速度指令值，单位：Hz。
float	vfAngle	vf角度，单位：rad。
float	ts	控制周期，单位：s。
float	spdThr[2]	最小速度阈值和最大速度阈值。
float	voltThr[2]	最小电压阈值和最大电压阈值。
float	slope	斜坡斜率值，单位：Hz/s。
DqAxis	ratio	dq轴的电压指令分配比例。
DqAxis	vdqRef	dq轴电压指令。
RMG_Handle	rmg	斜坡管理控制器。



17.1.2 API 接口

表 17-2 接口列表

函数名称	功能描述
VF_Init	初始化vf控制器。
VF_Exec	执行vf控制。
VF_Clear	清除vf控制器历史值。
VF_SetTs	设置vf控制器的控制周期。
VF_SetSpdSlope	设置速度斜坡控制器的斜率。
VF_SetDRatio	设置d轴的电压指令分配比例。

17.1.2.1 VF_Init

17.1.2.1.1 API 形式

表 17-3 初始化 vf 控制器 API

标题	描述
接口形式	void VF_Init(VF_Handle *vf, const float *spdThr, const float *voltThr, float ts, float spdCmd, float spdSlope)
输入参数	vf: 指向VF_Handle结构体的指针。 spdThr: spdThr[0]: vf控制的初始速度; spdThr[1]: vf控制的最大速度。 voltThr: voltThr[0]: vf控制的初始电压矢量; voltThr[1]: vf控制的最大电压矢量。 ts: 控制周期。 spdCmd: 速度目标值, 单位: Hz。 spdSlope: 速度斜坡, 单位Hz/s。
返回参数	VF控制vf结构体。
返回值	无。

17.1.2.1.2 API 功能描述

初始化vf控制器参数。



17.1.2.1.3 Example

```
VF_Handle vf = {0};
float spdThr[2] = {5.0f, 50.0f}; /* vf控制的初始速度为5Hz，目标速度为50Hz */
float voltThr[2] {10.0f, 24.0f}; /* vf控制的初始电压矢量为10V，最大电压矢量为24V */
float ts = 0.0001f;
float spdCmd = 40.0f;
float spdSlope = 1.0f
VF_Init(&vf, spdThr, voltThr, ts, spdCmd, spdSlope);
```

17.1.2.1.4 注意事项

无。

17.1.2.2 VF_Exec

17.1.2.2.1 API 形式

表 17-4 执行 vf 控制计算 API

标题	描述
接口形式	void VF_Exec(VF_Handle *vf, DqAxis *vdqRef)
输入参数	vf: 指向VF_Handle结构体的指针。 vdqRef: vf控制的dq轴电压指令。
返回参数	VF控制vf结构体中dq轴电压指令。
返回值	无。

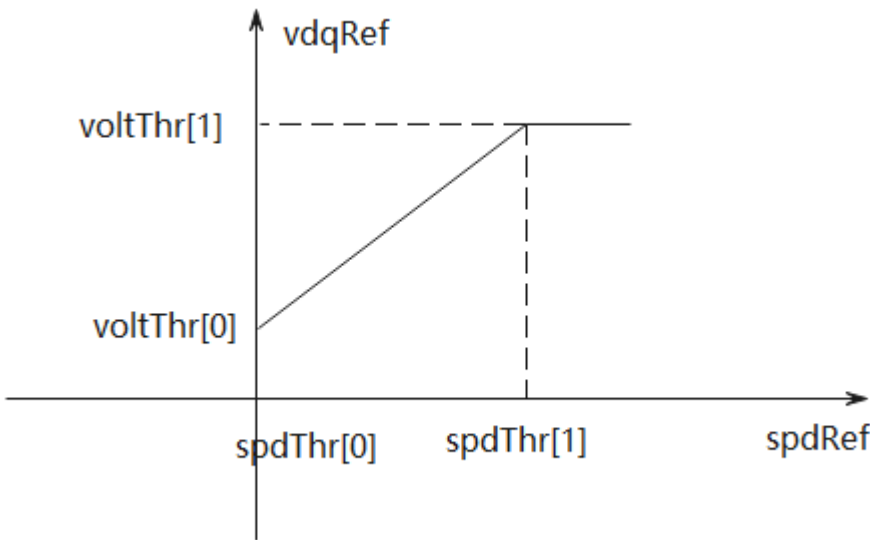
17.1.2.2.2 API 功能描述

vf控制器执行，根据参考速度生成角度信息和dq轴电压指令，如图17-1所示。

- 当vf参考速度小于最小速度阈值spdThr[0]时，电压设置为最小电压阈值voltThr[0]。
- 当vf参考速度大于最大速度阈值spdThr[1]时，电压设置为最大电压阈值voltThr[1]。



图 17-1 API 功能描述图



17.1.2.2.3 Example

```
VF_Handle vf = {0};
DqAxis vdqRef = {0};
float spdRef = 0;
float vfAngle = 0;
...
VF_Exec(&vf, &vdqRef);
spdRef = vf.spdRef;
vfAngle = vf.vfAngle;
```

17.1.2.2.4 注意事项

无。

17.1.2.3 VF_Clear

17.1.2.3.1 API 形式

表 17-5 清除 vf 控制历史值 API

标题	描述
接口形式	void VF_Clear(VF_Handle *vf)
输入参数	vf: 指向VF_Handle的结构体指针。
返回参数	VF控制vf结构体。
返回值	无。



17.1.2.3.2 API 功能描述

清除vf控制器历史值。

17.1.2.3.3 Example

```
VF_Handle vf = {0};
...
VF_Clear(&vf);
```

17.1.2.3.4 注意事项

无。

17.1.2.4 VF_SetTs

17.1.2.4.1 API 形式

表 17-6 设置 vf 控制器的控制周期 API

标题	描述
接口形式	void VF_SetTs(VF_Handle *vf, float ts)
输入参数	vf: 指向VF_Handle的结构体指针。 ts: 要设置的控制周期。
返回参数	vf结构体中控制周期。
返回值	无。

17.1.2.4.2 API 功能描述

设置vf控制器的控制周期。

17.1.2.4.3 Example

```
VF_Handle vf = {0};
float ts = 0.002f;
VF_SetTs(&vf, ts);
```

17.1.2.4.4 注意事项

合法的ts应为正浮点数。

17.1.2.5 VF_SetSpdSlope



17.1.2.5.1 API 形式

表 17-7 设置 vf 控制器的速度斜率 API

标题	描述
接口形式	void VF_SetSpdSlope(VF_Handle *vf, float spdSlope)
输入参数	vf: 指向VF_Handle的结构体指针。 spdSlope: 要设置的速度斜坡值，单位：Hz/s。
返回参数	vf结构体中速度斜率。
返回值	无。

17.1.2.5.2 API 功能描述

设置vf控制器的速度斜率。

17.1.2.5.3 Example

```
VF_Handle vf = {0};
float spdSlope = 2.0f;
VF_SetSpdSlope(&vf, spdSlope);
```

17.1.2.5.4 注意事项

无。

17.1.2.6 VF_SetDRatio

17.1.2.6.1 API 形式

表 17-8 设置 vf 控制器 d 轴电压比例 API

标题	描述
接口形式	void VF_SetDRatio(VF_Handle *vf, float dRatio)
输入参数	vf: 指向VF_Handle的结构体指针。 dRatio: d轴的电压比例，取值0~1，默认为1。
返回参数	vf结构体中d轴电压比例。
返回值	无。

17.1.2.6.2 API 功能描述

设置vf控制器的d轴电压比例，那么q轴电压比例为 $qRatio = \sqrt{1 - dRatio * dRatio}$ 。



17.1.2.6.3 Example

```
VF_Handle vf = {0};  
float dRatio = 0.8f; /* dq轴电压比例的平方和为1，因此q轴的电压比例为0.6 */  
VF_SetDRatio(&vf, dRatio);
```

17.1.2.6.4 注意事项

无。



A 缩略语

表 A-1 缩略语

缩略语	英文	中文解释
Det	Dedect	检测
Exec	Execute	执行