文档编号：AN147

上海东软载波微电子有限公司

用户手册

电机开发套件 FOC 软件库 ESMCLIB

修订历史

版本	修订日期	修改概要
V1.0	2021.12.24	初版发布

地　址：中国上海市徐汇区古美路 1515 号凤凰园 12 号楼 3 楼
E-mail：support@essemi.com
电　话：+86-21-60910333
传　真：+86-21-60914991
网　址：http://www.essemi.com/

目 录

内容目录

第1章 概述

1.1 控制框图

ESMCLIB（ES Motor Control Library）向用户提供了 FOC 软件库，该软件库算法完善、层次分明以及模块众多，方便用户快速建立自己的 FOC 应用项目工程。ESMCLIB 软件库的 FOC 控制架构如下图所示。

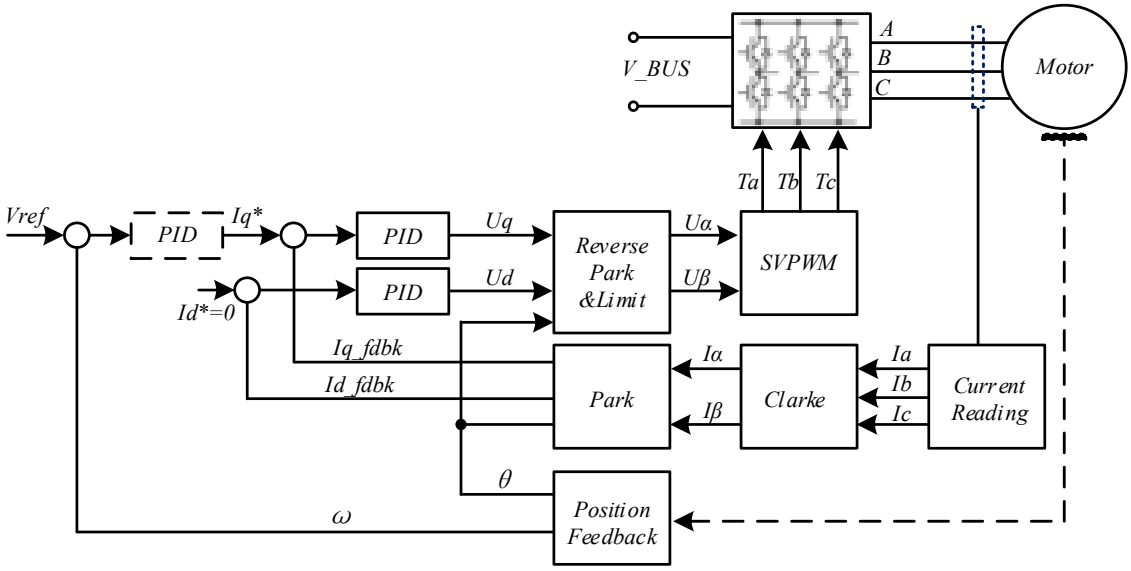

图 1.1 ESMCLIB FOC 软件库控制架构图

1.2 层次结构

ESMCLIB 软件库中涵盖的各功能组件层次结构概览如下图所示。

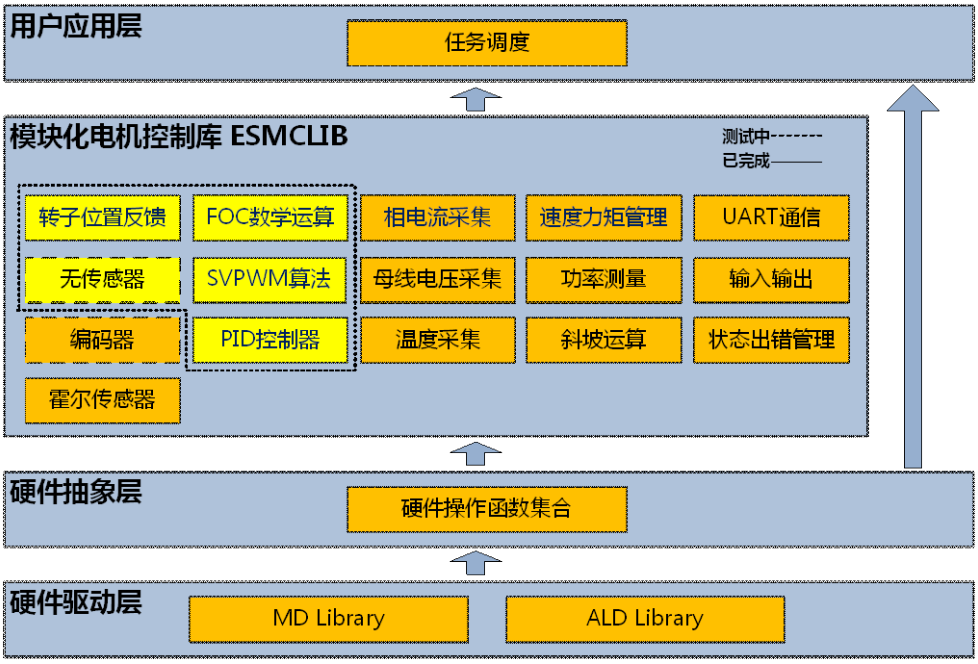

图 1.2 ESMCLIB 软件层次结构示意图

第2章 ESMCLIB软件接口

2.1 常用硬件操作

2.1.1 概况

软件库向应用层用户提供常用的硬件操作接口函数，概况如下表所示：

函数名	形式参数	功能简述
HW_Abstr_SwitchOffPwm	HWOption_Struct_t *	关闭 PWM
HW_Abstr_SwitchOnPwm	HWOption_Struct_t *	打开 PWM
HW_Abstr_OpenLowSideMos	HWOption_Struct_t *	打开下桥 MOS
HW_Abstr_CloseLowSideMos	HWOption_Struct_t *	关闭下桥 MOS
HW_Abstr_OpenHighSideMos	HWOption_Struct_t *	打开上桥 MOS
HW_Abstr_CloseHighSideMos	HWOption_Struct_t *	关闭上桥 MOS
HW_Abstr_ReadAdcNormChConv	HWOption_Struct_t *,uint8_t	采集 ADC 标准组转换值

2.1.2 核心接口使用

常用硬件操作中，用户较常使用的接口主要有两个，一个是 HW_Abstr_OpenLowSideMos()，另一个是 HW_Abstr_SwitchOnPwm()。HW_Abstr_OpenLowSideMos()输入参数是 HWOption_Struct_t 结构体指针变量，无返回值。它给应用层用户提供一种硬件操作的抽象接口，功能是打开下桥 MOS 管，一般用于系统上电时自举充电阶段；HW_Abstr_SwitchOnPwm()的输入参数是 HWOption_Struct_t 结构体指针变量，无返回值，它给应用层用户提供一种硬件操作的抽象接口，其功能是打开 MCU 管脚的 PWM 脉冲输出。至于如何具体操作，由硬件实例化源文件的特定函数实现。使用示例如下图所示。

```
        case SYS_POWER:
            //打开下桥
            HW_Abstr_OpenLowSideMos(&gVarHw3shOpt.super);
            //设置自举充电时间
            TB_Set_Bootcap_Timeout(20);
            //获取下一个状态
            gVarTsk.SysState = SYS_BOOT;

        break;
```

```
case SYS_INIT:
    gVarTsk.SysState = SYS_START;
    HW_Abstr_SwitchOnPwm(&gVarHw3shOpt.super);
break;
```

图 2.1 应用示例 1

2.2 位置反馈

2.2.1 概况

软件库向应用层用户提供转子位置和速度等相关的接口函数，概况如下表所述：

函数名	形式参数	功能简述
Posfd_GetRotorElecAngle	PosFdbk_Struct_t *pHdl	获取转子电角度
Posfd_GetRotorMecAngle	PosFdbk_Struct_t *pHdl	获取转子机械角度
Posfd_GetRotorMechSpeedRPM	PosFdbk_Struct_t *pHdl	获取机械转速 rpm
Posfd_GetRotorElecSpeedRPM	PosFdbk_Struct_t *pHdl	获取电转速 rpm
Posfd_GetRotorMechSpeed0_1RPS	PosFdbk_Struct_t *pHdl	获取机械转速 0.1rpm
Posfd_GetRotorElecSpeed0_1RPS	PosFdbk_Struct_t *pHdl	获取电转速 0.1rpm
Posfd_GetRotorMechSpeedS16	PosFdbk_Struct_t *pHdl	获取机械转速 s16 格式
Posfd_GetRotorElecSpeedS16	PosFdbk_Struct_t *pHdl	获取电转速 s16 格式
Posfd_GetElecAngleIncPerPwm	PosFdbk_Struct_t *pHdl	获取角速度（每 PWM 周期，角度增量）
Posfd_SpeedFaultCheck	PosFdbk_Struct_t *pHdl	速度反馈错误检查
Posfd_IsMechSpeedReliable	PosFdbk_Struct_t *pHdl	机械速度可靠性检查
Posfd_GetPolePairsNum	PosFdbk_Struct_t *pHdl	获取电机极对数
SetPolePairsNum	PosFdbk_Struct_t *pHdl	设置电机极对数

2.2.2 核心接口使用

位置反馈功能中，最常使用的接口是 Posfd_GetRotorElecAngle()，输入参数为 PosFdbk_Struct_t 结构体指针变量，返回值为电角度（单位：S16 格式数字量），可在应用层中调用。其功能是获取转子电角度，一般用于 FOC 算法中的坐标运算。使用示例如下图所示：

```
Posfd_GetRotorElecAngle(gVarTsk.pPosfd);
GetThreePhaseCurrent_ByThreeShunt(&gVarCurrSmp,&gVarSvpwm);
gVarCurrSmp.Ialfa_beta      =   Clarke(gVarCurrSmp.Ia_b);
gVarCurrSmp.I_q_d           =   Park(gVarCurrSmp.Ialfa_beta,gVarHallAngle.Common.s16ElecAngle);
gVarSvpwm.Uq_d.s16U_Member1 =   PID_Regulators(gVarTsk.s16TargetTorque,gVarCurrSmp.I_q_d.s16I_Member1, &gVarIqCurrentPid);
gVarSvpwm.Uq_d.s16U_Member2 =   PID_Regulators(0,gVarCurrSmp.I_q_d.s16I_Member2, &gVarIdCurrentPid);
gVarSvpwm.Uq_d              =   VoltQDLimitation(gVarSvpwm.Uq_d);
gVarSvpwm.Ualfa_beta        =   Rev_Park(gVarSvpwm.Uq_d);
ThreeShuntSevenSec_SvpwmGenerator(&gVarSvpwm);
```

图 2.2 应用示例 2

2.3 相电流采集

2.3.1 概况

软件库向应用层用户提供相电流采集等相关的接口函数，概况如下表所述：

函数名	形式参数	功能简述
GetThreePhaseCurrent_ByThreeShunt	CurrSmp_Struct_t*,Svpwm_Struct_t *	三电阻方式采集电流
ReadCurrentOffsetValue	CurrSmp_Struct_t*	读取相电流零点偏置值
GetPhaseACurrentOffset	CurrSmp_Struct_t*	获取 A 相零点偏置值

函数名	形式参数	功能简述
GetPhaseBCurrentOffset	CurrSmp_Struct_t*	获取 B 相零点偏置值
GetPhaseCCurrentOffset	CurrSmp_Struct_t*	获取 C 相零点偏置值
GetPhaseACurrentValue	CurrSmp_Struct_t*	获取 A 相相电流值
GetPhaseBCurrentValue	CurrSmp_Struct_t*	获取 B 相相电流值
GetAlphaCurrentValue	CurrSmp_Struct_t*	获取 Alpha 轴电流值
GetBetaCurrentValue	CurrSmp_Struct_t*	获取 Beta 轴电流值
GetIdCurrentValue	CurrSmp_Struct_t*	获取 D 轴电流值
GetIqCurrentValue	CurrSmp_Struct_t*	获取 Q 轴电流值

2.3.2 核心接口使用

相电流采集功能中，常用的接口是 GetThreePhaseCurrent_ByThreeShunt()和 ReadCurrentOffsetValue()。函数 GetThreePhaseCurrent_ByThreeShunt()的输入参数为 CurrSmp_Struct_t 结构体指针变量和 Svpwm_Struct_t 结构体体指针变量，其作用是通过三电阻采样的方式采集电机三相相电流并存储到 CurrSmp_Struct_t 结构体变量。三电阻采样需要按扇区切换特定的 ADC 通道，扇区号信息来源于另一个输入参数 Svpwm_Struct_t 结构体中的成员变量；函数 ReadCurrentOffsetValue()的输入参数是 CurrSmp_Struct_t 结构体指针变量，无返回值，在电机启动前调用该函数以获取相电流的 ADC 通道零点偏置值。由于相电流是一个交流信号，应用电路会将交流信号增加一个直流偏置供 ADC 采集。在电机运行前就必须获取相电流零点值才能进行 FOC 运算，该函数内部采样 1024 次并进行了均值计算。使用示例如下图所示：

```
case SYS_BOOT:
    if(TB_Bootcap_Timeout_IsElapsed()== 1){
        //开始读取电流偏移值
        ReadCurrentOffsetValue(&gVarCurrSmp);
        gVarTsk.SysState = SYS_IDLE;
    }
break;
```

```
Posfd_GetRotorElecAngle(gVarTsk.pPosfd);
GetThreePhaseCurrent_ByThreeShunt(&gVarCurrSmp,&gVarSvpwm);
gVarCurrSmp.Ialfa_beta      =   Clarke(gVarCurrSmp.Ia_b);
gVarCurrSmp.I_q_d           =   Park(gVarCurrSmp.Ialfa_beta,gVarHallAngle.Common.s16ElecAngle);
gVarSvpwm.Uq_d.s16U_Member1 =   PID_Regulators(gVarTsk.s16TargetTorque,gVarCurrSmp.I_q_d.s16I_Member1, &gVarIqCurrentPid);
gVarSvpwm.Uq_d.s16U_Member2 =   PID_Regulators(0,gVarCurrSmp.I_q_d.s16I_Member2, &gVarIdCurrentPid);
gVarSvpwm.Uq_d              =   VoltQDLimitation(gVarSvpwm.Uq_d);
gVarSvpwm.Ualfa_beta        =   Rev_Park(gVarSvpwm.Uq_d);
ThreeShuntSevenSec_SvpwmGenerator(&gVarSvpwm);
```

图 2.3 应用示例 3

2.4 FOC数学运算

2.4.1 概况

软件库向应用层用户提供转子位置和速度等相关的接口函数，概况如下表所述：

函数名	形式参数	功能简述
Clarke	Curr_Struct_t	CLARKE 坐标计算
Park	Curr_Struct_t, int16_t	PARKE 坐标计算
VoltQDLimitation	Volt_Struct_t	电压限幅处理
Rev_Park	Volt_Struct_t	反 PARKE 坐标计算

2.4.2 核心接口使用

FOC 数学运算接口函数作为 FOC 控制的核心部件，一般在高频任务中进行调用。本固件中在 TSK_FOCMotorControl()任务中由用户使用。使用示例如下图所示：

```
gVarCurrSmp.Ialfa_beta      =   Clarke(gVarCurrSmp.Ia_b);
gVarCurrSmp.I_q_d           =   Park(gVarCurrSmp.Ialfa_beta,gVarHallAngle.Common.s16ElecAngle);
gVarSvpwm.Uq_d.s16U_Member1 =   PID_Regulators(gVarTsk.s16TargetTorque,gVarCurrSmp.I_q_d.s16I_Member1, &gVarIqCurrentPid);
gVarSvpwm.Uq_d.s16U_Member2 =   PID_Regulators(0,gVarCurrSmp.I_q_d.s16I_Member2, &gVarIdCurrentPid);
gVarSvpwm.Uq_d              =   VoltQDLimitation(gVarSvpwm.Uq_d);
gVarSvpwm.Ualfa_beta        =   Rev_Park(gVarSvpwm.Uq_d);
```

图 2.4 应用示例 4

2.5 PID控制器

2.5.1 概况

软件库向应用层用户提供 PI 控制器相关的接口函数，概况如下表所述：

函数名	形式参数	功能简述
PID_SetKP	Pid_Struct_t*，int16_t	设置比例项系数
PID_SetKI	Pid_Struct_t*，int16_t	设置积分项系数
PID_GetKP	Pid_Struct_t*	获取比例项系数
PID_GetKI	Pid_Struct_t*	获取积分项系数
PID_SetIntegralTerm	Pid_Struct_t*，int16_t	设置积分值
PID_GetKPDivisor	Pid_Struct_t*	获取比例项缩放因子
PID_SetKPDivisorBS	Pid_Struct_t*，int16_t	设置比例项缩放因子移位数
PID_GetKIDivisor	Pid_Struct_t*	获取积分项缩放因子
PID_SetKIDivisorBS	Pid_Struct_t*，int16_t	设置积分项缩放因子移位数
PID_SetLowerIntegralTermLimit	Pid_Struct_t*，int16_t	设置积分项累加下限边界值
PID_SetUpperIntegralTermLimit	Pid_Struct_t*，int16_t	设置积分项累加上限边界值
PID_SetLowerOutputLimit	Pid_Struct_t*，int16_t	设置控制器输出下限边界值
PID_SetUpperOutputLimit	Pid_Struct_t*，int16_t	设置控制器输出上限边界值
PID_Regulators	int16_t , int16_t, Pid_Struct_t *	PID 运算

2.5.2 核心接口使用

函数 PID_Regulators()作为 FOC 控制的核心部件之一，一般在高频任务中进行调用。本固件中在 TSK_FOCMotorControl()任务中由用户使用。使用示例如下图所示：

```
gVarSvpwm.Uq_d.s16U_Member1 =   PID_Regulators(gVarTsk.s16TargetTorque,gVarCurrSmp.I_q_d.s16I_Member1, &gVarIqCurrentPid);
gVarSvpwm.Uq_d.s16U_Member2 =   PID_Regulators(0,gVarCurrSmp.I_q_d.s16I_Member2, &gVarIdCurrentPid);
gVarSvpwm.Uq_d              =   VoltQDLimitation(gVarSvpwm.Uq_d);
gVarSvpwm.Ualfa_beta        =   Rev_Park(gVarSvpwm.Uq_d);
ThreeShuntSevenSec_SvpwmGenerator(&gVarSvpwm);
```

图 2.5 应用示例 5

2.6 空间矢量脉宽调制

2.6.1 概况

软件库向应用层用户提供空间矢量脉宽调制相关的接口函数，概况如下表所述：

函数名	形式参数	功能简述
ThreeShuntSevenSec_SvpwmGenerator	Svpwm_Struct_t	SVPWM 算法
Svpwm_GetCurrentSectorID	Svpwm_Struct_t	获取当前电压矢量所在的扇区区号
Svpwm_GetPhaseADutyValue	Svpwm_Struct_t	获取当前 A 相占空比数值
Svpwm_GetPhaseBDutyValue	Svpwm_Struct_t	获取当前 B 相占空比数值
Svpwm_GetPhaseCDutyValue	Svpwm_Struct_t	获取当前 C 相占空比数值
Svpwm_GetQaxisVoltage	Svpwm_Struct_t	获取旋转坐标系下 Q 轴定子电压
Svpwm_GetDaxisVoltage	Svpwm_Struct_t	获取旋转坐标系下 D 轴定子电压
Svpwm_GetALPHAaxisVoltage	Svpwm_Struct_t	获取静止坐标系下 Alpha 轴定子电压
Svpwm_GetBETAaxisVoltage	Svpwm_Struct_t	获取静止坐标系下 Beta 轴定子电压

2.6.2 核心接口使用

函数 ThreeShuntSevenSec_SvpwmGenerator()作为 FOC 控制的核心部件之一，一般在高频任务中进行调用。本固件在 TSK_FOCMotorControl()任务中由用户使用。使用示例如下图所示：

```
GetThreePhaseCurrent_ByThreeShunt(&gVarCurrSmp,&gVarSvpwm);
gVarCurrSmp.Ialfa_beta      =   Clarke(gVarCurrSmp.Ia_b);
gVarCurrSmp.I_q_d           =   Park(gVarCurrSmp.Ialfa_beta,gVarHallAngle.Common.s16ElecAngle);
gVarSvpwm.Uq_d.s16U_Member1 =   PID_Regulators(gVarTsk.s16TargetTorque,gVarCurrSmp.I_q_d.s16I_Member1, &gVarIqCurrentPid);
gVarSvpwm.Uq_d.s16U_Member2 =   PID_Regulators(0,gVarCurrSmp.I_q_d.s16I_Member2, &gVarIdCurrentPid);
gVarSvpwm.Uq_d              =   VoltQDLimitation(gVarSvpwm.Uq_d);
gVarSvpwm.Ualfa_beta        =   Rev_Park(gVarSvpwm.Uq_d);
ThreeShuntSevenSec_SvpwmGenerator(&gVarSvpwm);
```

图 2.6 应用示例 6

2.7 速度力矩管理

2.7.1 概况

软件库向应用层用户提供速度和力矩管理相关的接口函数，概况如下表所述：

函数名	形式参数	功能简述
TorqSpdMng_Component_Init	TorqSpdMng_Struct_t *,Pid_Struct_t *,PosFdbk_Struct_t *)	组件初始化
TorqSpdMng_Clear	TorqSpdMng_Struct_t *	组件变量清零
TorqSpdMng_SetSpdFdbkMethod	TorqSpdMng_Struct_t *	设置组件速度反馈指针
TorqSpdMng_SteupRampCurve	TorqSpdMng_Struct_t *	建立爬坡曲线任务
TorqSpdMng_RefTorqueCalc	TorqSpdMng_Struct_t *	计算爬坡曲线点值
TorqSpdMng_IsRampCompleted	TorqSpdMng_Struct_t *	判断爬坡曲线任务是否结束
TorqSpdMng_StopSpdRampTsk	TorqSpdMng_Struct_t *	立刻停止爬坡曲线任务
TorqSpdMng_RefSpeedEqualToActualSpeed	TorqSpdMng_Struct_t *	将参考速度设置为当前实际速度

2.7.2 核心接口使用

速度力矩管理中，常用接口是 TorqSpdMng_SteupRampCurve()和 TorqSpdMng_RefTorqueCalc()。TorqSpdMng_SteupRampCurve()输入参数为 TorqSpdMng_Struct_t 结构体指针变量，uint32_t 型变量和 int16_t 型变量，返回值为 bool 型，第一个参数是组件结构体指针，第二个参数是用户定义的时间，单位 ms，即上升或者下降到目标值所需要的时间，第三个参数是用户定义的终值，可以是速度值也可以是力矩值，具体取决于用户设定的控制模式。在应用层调用该函数后，系统生成一个符合用户定义的爬坡曲线任务，配合爬坡曲线点累加计算函数 TorqSpdMng_RefTorqueCalc()可达到目的；TorqSpdMng_RefTorqueCalc()输入参数为 TorqSpdMng_Struct_t 结构体指针变量，返回值为 int16_t 型，输入参数是组件结构体指针，返回值为当前斜坡曲线点值。用户在低频任务中调用该函数后，可按照固定频率依据前文 TorqSpdMng_SteupRampCurv()定义的斜坡参数进行处理，函数返回值输入到力矩电流环 PID 参考值。使用示例如下图所示：

```
case SYS_START:
    TorqSpdMng_SteupRampCurve(&gVarTorSpd,6000,2000);//建立斜坡曲线任务
    gVarTsk.SysState = SYS_RUN;
break;

case SYS_RUN:
    //计算参考电流
    gVarTsk.s16TargetTorque = TorqSpdMng_RefTorqueCalc(&gVarTorSpd);
```

图 2.7 应用示例 7

第3章 ESMCLIB硬件平台

3.1 硬件构成

为配合固件调试及使用，我司可向用户提供低压驱动开发板 ES-GMB-MOTOLV1，该驱动开发板由电源电路、功率驱动电路、位置检测电路、电流采样电路、通信电路以及保护电路等模块构成，其结构框图和实物图如图所示。

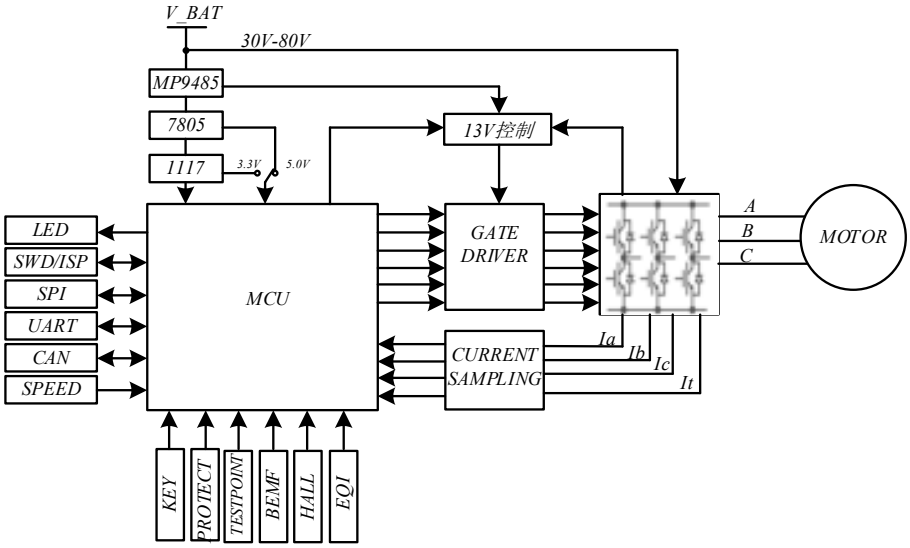

图 3.1 ES-GMB-MOTOLV1 电路结构图

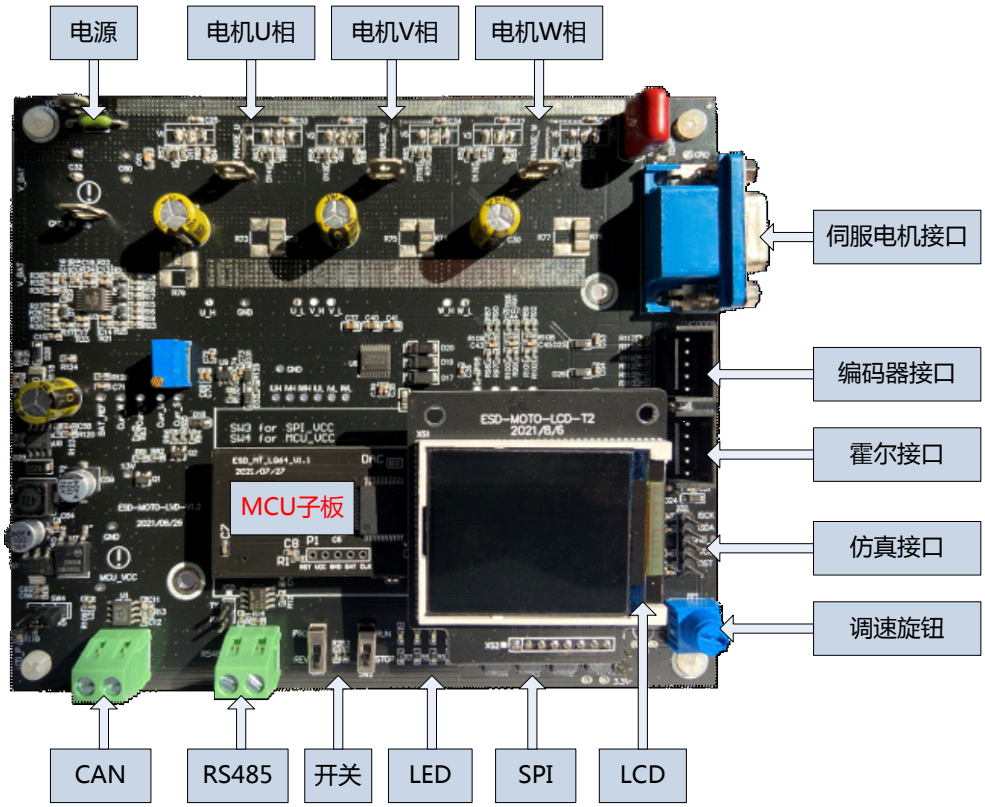

图 3.2 实物图

3.2 方案特性

ES-GMB-MOTOLV1 低压驱动开发版有如下特点：

支持多种电机控制方式：

- HALL/ENCODER 传感器 FOC 控制
- 无传感器 BLDC 控制
- 无传感器 FOC 控制

通信接口：

- 支持 SPI
- 支持 UART，RS485
- 支持 CAN
- 支持 SWD 调试协议

保护功能

- 软硬件过流保护
- 欠压保护
- 过压保护
- 过热保护

控制与显示

- 调速旋钮
- 正反转、启刹车开关
- 按键 x 4
- LED x 4
- LCD

其他特性：

- 额定值为 100V/192A 的三相逆变桥
- 电压输入范围 30V-80V
- 采用子母板方式，可灵活搭配 ES32 系列 MCU
- 抗高 dv/dt 共模干扰和瞬态负电压的三相栅极驱动器
- 多达 22 个信号测试点

3.3 注意事项

- MCU 控制子板使用前确认与母板完全贴合压紧；
- 用户输入电压一定要控制在 30-80V 范围内；
- 供电后发现驱动板各测点输出电压不对，需要调整电位器电压。如图 3.3 所示；
- 电机相线和霍尔线请按照图 3.4 所示连接，连接方法请参考 4.1 节所示。

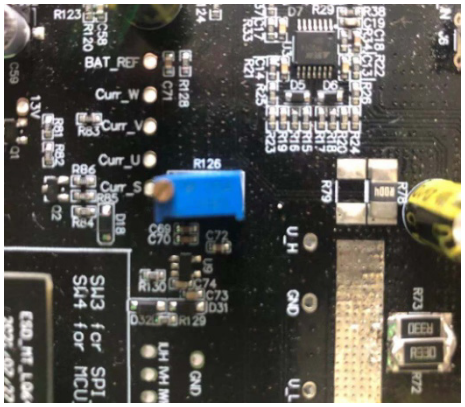

图 3.3　电位器

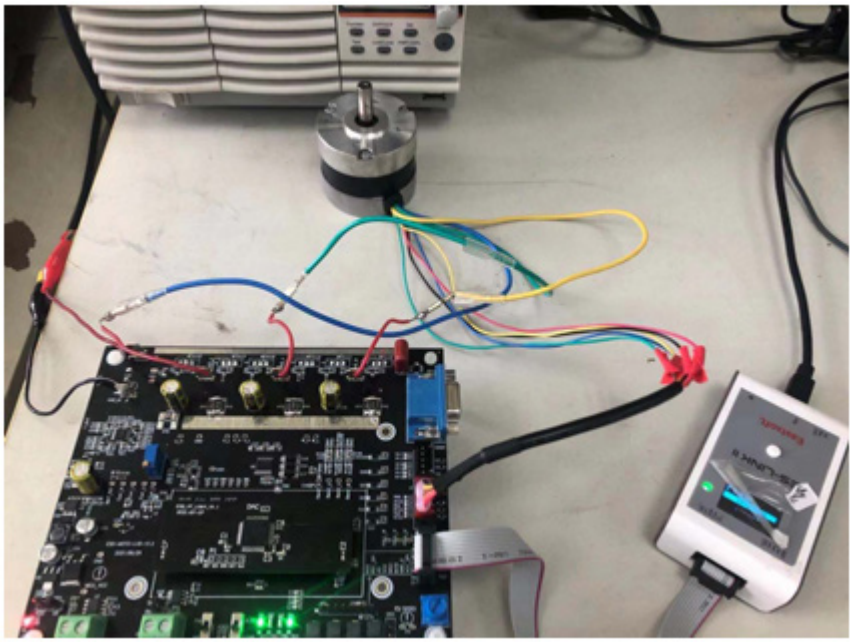

图 3.4　实物连接示意图

第4章 ESMCLIB调试

用户熟悉我司提供的硬件平台后或者使用自己开发的硬件系统，在初次使用该固件库时可按照下文步骤依次展开调试。一是方便用户尽快掌握固件特性；二是能及早发现每个环节存在的问题以提高开发效率。

4.1 虚拟电角度开环调试

在工程中头文件“esmc_method.h”中找到宏“OPEN_LOOP_DEBUG”，取消对其的注释，即开始虚拟电角度开环调试环节。在虚拟电角度开环调试阶段，按照如下步骤进行：

1. 先将电机相线接到电机驱动板 U,V,W 相线接线柱；
2. 在 esmc_method.h 中调整 PHASE_INC 和 UQ_DEBUG（UQ_DEBUG 的值一般取电机额定电压幅值的%10~20%，PHASE_INC 取正整数，用户根据实际电机参数进行调整）；
3. 编译下载到目标板中，同时给驱动板接上电源。观察电机能否平稳运转，同时规定电机旋转正反向。如果与规定的正反向相反，则调整电机相线顺序，直至调整到与规定正方向一致。
4. 如果电机运转不平稳或者电流过大，返回第 2 步适当调整 PHASE_IN 和 UQ_DEBUG，当电压幅值 UQ_DEBUG 与虚拟电角度转速 PHASE_IN 相匹配时，电机是可以平稳运转的而且电流较小。如果用户无论怎么调整都无法实现电机平稳运行，请检查驱动板 PWM 部分是否正常。
5. 执行上述步骤调整到电机能按照用户规定正方向平稳运行，接入电机霍尔传感器线。使用 ESLINK 在线调试，观察正方向运转时，霍尔状态变量是否能按照 5->1->3->2->6->4 变化。否则，先用示波器检测霍尔信号是否正常，如果霍尔信号正常请调整霍尔传感器接线顺序，调整到霍尔状态在电机正方向运行时能按照 5->1->3->2->6->4 变化。
6. 当电机开环稳定运行时，可以通过示波器观察各个信号变化情况，进而可以发现电路或者程序问题，比如用户可以观察相电流实际波形与运放输出对比情况进而判断采样电路是否存在问题，如图 4.1。

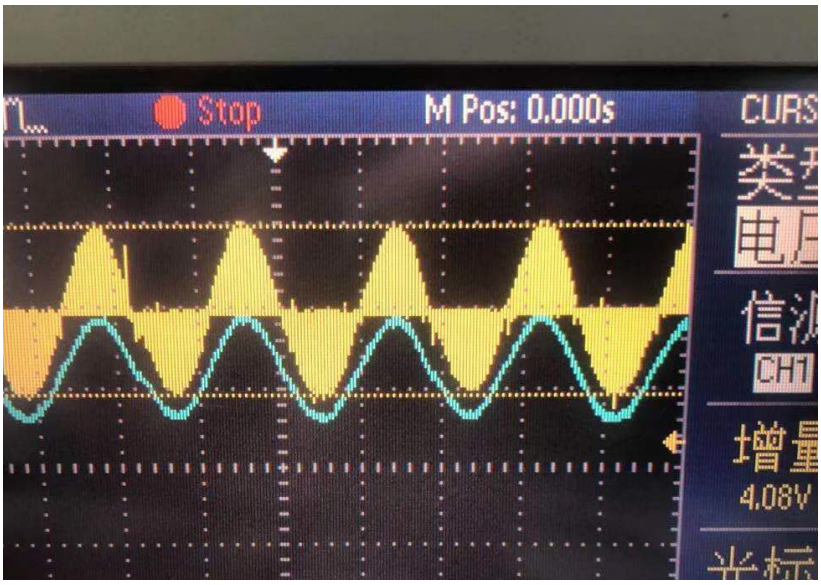

图 4.1 相电流测试波形

设计此调试环节目的：

1.便于用户逐步检查硬件环节是否存在问题，通过该调试环节用户可以确认 MCU 芯片、PWM 驱动电路、PWM 死区时间、HALL 信号及其接口电路、电机以及相电流采样电路是否正常；

2.确定了电机相线和霍尔线连接顺序，同时保证了电机运行正方向和电角度正向相匹配，以避免在后续的开发工作遭遇到方向逻辑混乱的局面；

3.确认软件算法模块的的正确性。

4.2 霍尔电角度开环调试

经过 3.1 的调试环节，可以确认系统运行的基础条件基本正常。由于 3.1 采用的是虚拟电角度，本环节将逐渐引入霍尔电角度代替虚拟电角度参与开环调试。在工程中头文件“esmc_method.h”中找到宏“HALL_CLOSELOOP_DEBUG，并取消对其的注释，即开始霍尔传感器电角度开环调试环节。在霍尔电角度调试阶段，按照如下步骤进行：

1.按照 3.1 步骤方法确认好电机相线和霍尔线连接正确并且工作正常；

2. 用户在头文件 esmc_paramconfig.h 中找到“Hall Sensor parameters”模块中的霍尔角度补偿宏“PHASE_BIAS”，该宏的数值调整范围是 0~360。不断调整该值编译下载，观察电机运行状况。此时，最好需要对电源进行限流，限流值可设置为额定电流的%20 以下，防止出现角度调整不合适导致堵转。

3.用户在调整 PHASE_BIAS 时，可每隔 30°依次进行粗调，编译下载后直到电机能平稳且电流较小。记住这个值，然后结合电流大小和转速进行微调。

4.微调后，直观上看电机运行电流非常小而且转速高，此时基本就差不多可以了。条件允许的前提下，可测量电流波形和电角度波形进行微调。理想状态下，空载时电角度 0 点对应 A 相电流正弦波形峰值点附近，如图 4.2 所示。

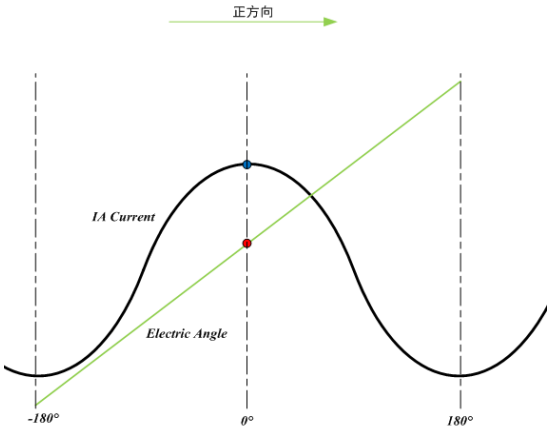

图 4.2 电角度与反电动势位置关系

5.通过上述步骤反复调试，用户可以方便地找到合适霍尔角度补偿值 PHASE_BIAS。电机能平稳运行，如有必要在此期间用户可以利用 DAC 通道观察其他重要变量，如相电流反馈、电角度变化、转矩电流和磁通电流等等。

设计此调试环节目的：

1.通过调试正确的引入霍尔角度反馈，找到合适的霍尔补偿角度；

2.观察各个关键变量的变化情况，能及时反映出硬件系统存在的问题；

3.确认软件霍尔估算算法模块的的正确性。

4.3 简易正弦速度闭环调试

经过 3.1 和 3.2 的调试步骤，已经初步具备引入闭环控制的基础条件。在此之前都是开环调试，尚未引入闭环控制，为了更好的帮助用户掌握本文在速度电流双闭环调试前设计了一个简易正弦控制电机的调试步骤。其目的是帮助用户更好地掌握固件特性以及及时发现隐藏问题。在工程中头文件“esmc_method.h”中找到宏“SIMPLE_FOC_METHOD_DBUG，并取消对其的注释，即开始简易正弦速度闭环调试环节，可按照如下步骤进行：

1.在 esmc_task.c 中找到系统任务调度函数 TSK_SystemScheduler()，在 SYS_START 状态下引入函数 TorqSpdMng_SteupRampCurve(&gVarTorSpd,6000,2000)建立斜坡速度曲线任务。用户可根据具体电机特性，设置该速度爬坡曲线。这里 6000 是爬升时间，单位 ms，2000 是目标速度，单位 rpm。

2. 在头文件 esmc_paramconfig.h 中找到“Pid Controller Parameters”部分，调整速度 PI 控制器比例系数宏 KP_SPD_DEFAULT 和速度 PI 控制器积分系数宏 KI_SPD_DEFAULT。先比例后积分，在简易正弦控制模式下调整速度闭环。

3.调整到合适参数，转速经过爬坡以后可以稳定在用户设置的速度值。此时，除了未引入电流 PI 控制闭环，所有控制部分均已完成调试，实际调试波形参照图 4.3。在当前运行条件下，可以通过 DAC 观察相电流反馈的正确性。确认相电流反馈基本正常后，就可以逐步引入 Id 电流闭环和 Iq 电流闭环控制。

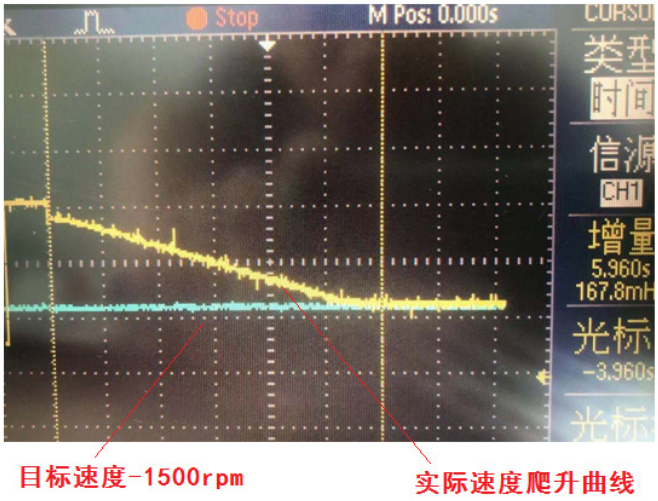

图 4.3 简易正弦速度闭环调试

设计此调试环节目的：

1.相对于电流环，速度环更容易调试，更直观。让用户初步调试速度闭环，理解固件特性和掌握调试方法；

2.在调试效果良好的条件下观察各个关键变量的变化情况，能及时反映出系统可能存在的问题；

3.确认软件 PI 算法模块的的正确性；

4.某些低端要求不高的电机应用产品即可采用简易正弦控制。

4.4 FOC速度电流双闭环调试

经过 3.1，3.2 和 3.3 的调试步骤，可以确认硬件系统正常、接线无误、霍尔补偿角度合适、相电流反馈正常以及闭环控制有效，排除了引入电流反馈之前的所有不确定性，此时可以正式引入电流闭环控制。在工程中头文件“esmc_method.h”中找到宏“FOC_METHOD_DBUG，并取消对其的注释，即开始 FOC 速度电流双闭环调试环节，可按如下步骤进行：

1.取消对 FOC_METHOD_DBUG 的注释后，系统在 3.4 简易速度闭环调试的基础上引入 Id 电流闭环，此时暂不引入 Iq 环。由于 3.3 步骤后，电机能进行稳定调速，此时让电机以速度闭环稳定运行起来；

2.确认电机能稳定运行后，设置 Id 的 PI 控制器参考目标值为 0。在头文件 esmc_paramconfig.h 中找到“Pid Controller Parameters”部分，调整 Id 电流 PI 控制器比例系数宏 KP_FLX_DEFAULT 和速度 PI 控制器积分系数宏 KI_FLX_DEFAULT，采取先比例后积分的原则。编译下载反复调试后直到 Id 反馈值能被稳定锁定在 0，且 PI 控制器输出波动维持在 3000 以内；

3.将调整完成的 Id 环的 PI 参数拷贝一份到 Iq 环，并将 gVarSvpwm.Uq_d.s16U_Member1 赋值为 Iq 环 PI 控制器输出。此时即引入了 Iq 电流环控制，然后给系统稍加负载对 Iq 环的 PI 参数进行微调。如有必要，在 Id 和 Iq 电流环调节完毕后，再对速度环 PI 参数进行微调。注意速度环的输出要进行限幅，限幅值是宏 IQ_MAX。

4.5 总结

1.通过 3.1 和 3.2 的调试，可以确认系统硬件是否存在问题、线路连接是否存在问题以软件算法是否存在问题。用户依次按步骤操作，可及时发现问题并进行处理，防止问题累积到后面不便处理。

2.通过 3.3 和 3.4 的调试，用户可逐步掌握系统闭环调试方法，继而应用到自己的产品开发工作中，同时也有利于用户快速掌握并使用本软件库。