AC78xx Motor App Development Guide

文档版本：2.3

发布日期： 2023-03-09

修订信息

版本	日期	作者	修订说明
1.0	2019-03-05	ATC	文档初版
2.0	2021-05-06	ATC	优化文档结构，增加参考文档
2.1	2021-08-02	ATC	软件工程结构调整；支持 IAR EWARM
2.2	2022-02-22	ATC	更新电机控制算法，增加参考资料链接
2.3	2023-03-09	ATC	增加 AC7840x 系列介绍及外设配置说明

版权声明

本文档包含 AutoChips 公司的机密信息。禁止未经授权使用或披露本文档包含的信息。对因未经 AutoChips 公司授权而全部或部分披露此文档内容而给 AutoChips 公司带来的任何损失或损害，AutoChips 公司将追究责任。

AutoChips 公司保留对此处任何信息进行更改的权利，此处的信息如有变更，恕不另行通知。AutoChips 公司对使用或依赖此处包含的信息不承担任何责任。

本文档的所有信息均“按原样”提供，不提供任何形式的明示，暗示，法定或其他形式的保证。AutoChips 公司明确拒绝对适销性，非侵权性和针对特定用途的适用性方面的的所有暗示保证。AutoChips 公司对本文档可能使用、包含或提供的任何第三方软件不提供任何担保，并且用户同意仅向该等第三方寻求与此相关的任何担保索赔。AutoChips 公司对于根据用户规格或为符合特定标准或公开论坛而产生的任何交付物，也不承担任何责任。

文档目录

插图目录

表格目录

1 简介

1.1 本文目的

本文主要目的是为了描述 Demo 板的适配 MotorApp 软件工程基本组成，并简要介绍基本软件模块初始化，根据实际电机的技术参数需适配修改的宏定义变量等，让用户能够快速在现有 MotorApp 工程上进行修改适配，满足项目的应用需求。

1.2 Motor_App 工程组成

以 AC780x 系列为例，Motor_App 工程组成部分如图 1-1 所示。

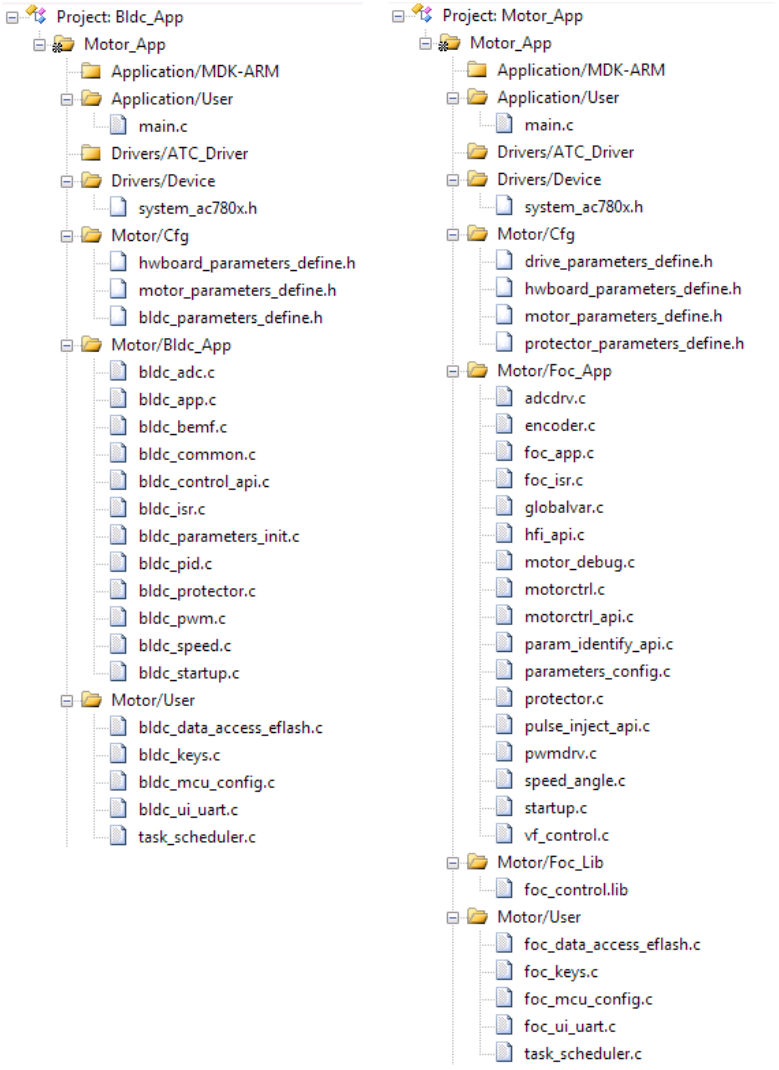

图 1-1 Motor_App 工程模块图

1.3 Motor_App 工程简介

1.3.1 Motor_App 功能

ATC 电机控制算法应用层软件（Motor_App）主要功能包括以下几个部分：

(1) **User**：

a. 系统时钟初始化

b. 各外设模块的初始化(GPIO、PWM、ADC、TIMER、CTU 等)

c. 控制板按钮控制接口实现

d. 电机控制算法任务调度

e. Motor Studio 上位机工具通信

f. 参数在 flash 中的固化/恢复

(2) **Bldc_App**：

a. 霍尔信号采集及电机转子速度解算

b. BLDC 带霍尔控制算法

c. BLDC 无感控制算法

d. BLDC 速度环调速实现

e. 电机保护策略

(3) **Foc_App**：

a. 速度斜坡

b. 电机保护策略

c. 无感电机启动策略

d. 相电流采样

e. ADC 中断，在 ADC 中断中调用 FOC 算法

f. 编码器速度估算

g. 电机控制 Debug 变量输出

(4) **Foc_Lib**：

a. FOC 算法矢量变换实现

b. SVPWM(Space Vector Pulse Width Modulation)空间矢量脉宽调制实现

c. HALL 传感器估算转子初始位置及霍尔安装方式辨识

d. 滑模观测器

e. PID 自整定（根据电机参数计算最佳 PID 参数）。

f. 脉冲注入法（识别电机初始位置）。

g. 高频注入算法（实现电机全速度范围内转速闭环）。

h. 快速刹车制动。

i. 磁链观测器

j. 模型参考自适应无感算法

k. 弱磁控制

l. 电机参数辨识

m. 单电阻采样电流重构（* AC781x 系列 MCU 不支持）

n. 节能控制算法

o. 功率与母线电流估算

p. 数学库

(5) **Config define：**

a. Demo 板硬件参数配置

b. 电机控制算法驱动参数设置

c. 电机硬件参数设置

d. 电机控制保护参数设置

1.3.2 Motor_App 硬件背景

基于 AC78xx 进行 BLDC 或 PMSM 电机控制，其原理框图如图 1-2 所示。

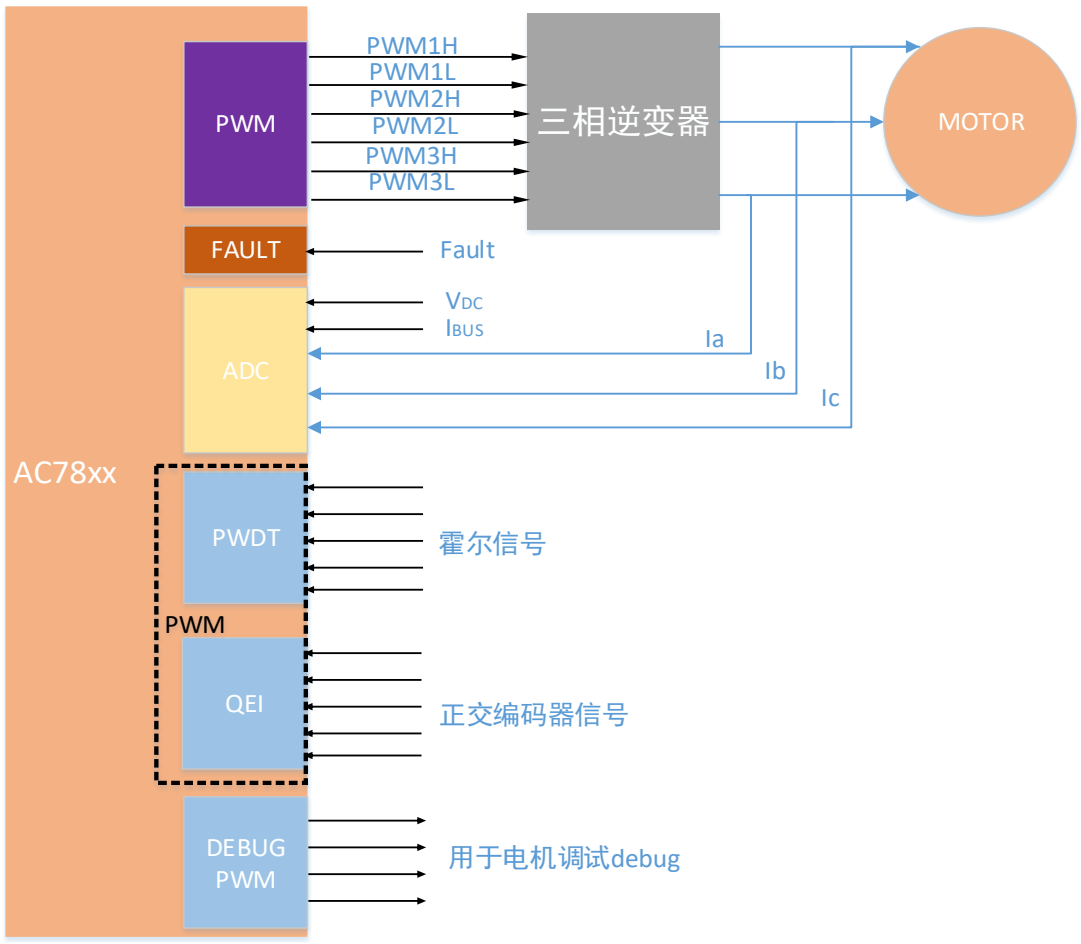

图 1-2 电机控制系统框图

由图 1-2 可知，电机控制系统需使用以下 MCU 外设资源，包括：

1、 PWM

AC780x 与 AC781x 系列中仅用于控制六路 PWM 输出，经逆变器后驱动三相电机。AC7840x 系列中集成了电机传感器信号捕获功能，可扩展用于霍尔，正交编码器等信号输入捕获。

2、 FAULT

检测电机过电流故障，便于进行电机保护。

3、 ADC

5 路 ADC 分别采集 a、b、c 三相电流，母线电压和母线电流，采集任意两相电流即可还原三相电流，满足 FOC 的 Clark 变换需求；5 路 ADC 信号均进行采集，则可实现更复杂的控制算法。

AC780x 与 AC7840x 系列 MCU 支持单电阻采样重构三相电流。

4、 PWDT

AC780x 与 AC781x 系列中用于 HALL 位置传感器采集，获取电机转子扇区位置。

5、 ACMP

无传感 BLDC 控制中进行相电压采集比较，获取电机转子扇区位置。

6、 QEI

AC780x 与 AC781x 系列中用于正交编码位置传感器采集，获取电机转子位置。

7、 Debug PWM

用于电机开发调试，产品量产时不使用，可与 CAN、UART、SPI 等外设模块复用。

1.3.3 代码结构及文件描述

Motor_App 的代码结构及文件路径描述如表 1-1 所示。

表 1-1 代码结构及描述表

文件名	文件路径	文件描述
bldc_adc.c	Motor_App\Motor-Master\Bldc\Bldc_App\src	ADC 采样直流母线
bldc_app.c	Motor_App\Motor-Master\Bldc\Bldc_App\src	Bldc 电机控制参数初始化及主函数
bldc_bemf.c	Motor_App\Motor-Master\Bldc\Bldc_App\src	Bldc 电机控制反电动势过零检测,换相
bldc_common.c	Motor_App\Motor-Master\Bldc\Bldc_App\src	Bldc 电机控制相关结构体
bldc_control_api.c	Motor_App\Motor-Master\Bldc\Bldc_App\src	Bldc 电机控制功能接口
bldc_isr.c	Motor_App\Motor-Master\Bldc\Bldc_App\src	Bldc 电机控制相关中断回调任务
bldc_parameters_init.c	Motor_App\Motor-Master\Bldc\Bldc_App\src	Bldc 电机控制相关参数初始化
bldc_pid.c	Motor_App\Motor-Master\Bldc\Bldc_App\src	Bldc 电机控制 PID 回路运算
bldc_protector.c	Motor_App\Motor-Master\Bldc\Bldc_App\src	Bldc 电机控制故障保护

文件名	文件路径	文件描述
bldc_pwm.c	Motor_App\Motor-Master\Bldc\Bldc_App\src	Bldc 电机控制 PWM 发波驱动
bldc_speed.c	Motor_App\Motor-Master\Bldc\Bldc_App\src	Bldc 中速度指令给定运算
bldc_startup.c	Motor_App\Motor-Master\Bldc\Bldc_App\src	Bldc 电机启动控制
adcdrv.c	Motor_App\Motor-Master\Foc\Foc_App\src	ADC 采样
encoder.c	Motor_App\Motor-Master\Foc\Foc_App\src	编码器位置传感器的 Foc 控制
foc_app.c	Motor_App\Motor-Master\Foc\Foc_App\src	Foc 电机控制主函数
foc_isr.c	Motor_App\Motor-Master\Foc\Foc_App\src	Foc 电机控制相关中断函数
globalvar.c	Motor_App\Motor-Master\Foc\Foc_App\src	Foc 电机控制相关结构体
hfi_api.c	Motor_App\Motor-Master\Foc\Foc_App\src	高频注入 app 接口文件
motor_debug.c	Motor_App\Foc\Foc_App\src	Foc 调试中 DAC 输出送显
motorctrl.c	Motor_App\Motor-Master\Foc\Foc_App\src	正交编码器位置传感器的 Foc 控制
motorctrl_api.c	Motor_App\Motor-Master\Foc\Foc_App\src	Foc 中电机控制接口文件
param_identify_api.c	Motor_App\Motor-Master\Foc\Foc_App\src	电机参数识别接口文件
parameters_config.c	Motor_App\Motor-Master\Foc\Foc_App\src	Foc 电机控制参数配置初始化
protector.c	Motor_App\Motor-Master\Foc\Foc_App\src	Foc 中电机保护策略

文件名	文件路径	文件描述
pulse_inject.c	Motor_App\Motor-Master\Foc\Foc_App\src	脉冲注入 app 接口文件
pwmdrv.c	Motor_App\Motor-Master\Foc\Foc_App\src	PWM 模块驱动
speed_angle.c	Motor_App\Motor-Master\Foc\Foc_App\src	Foc 中速度指令及速度环运算
startup.c	Motor_App\Motor-Master\Foc\Foc_App\src	无感 Foc 启动策略控制
vf_control.c	Motor_App\Motor-Master\Foc\Foc_App\src	电机 V/f 控制
bldc_parameters_define.h	Motor_App\Inc\Bldc_Cfg	Bldc 电机算法控制参数定义头文件
hwboard_parameters_define.h	Motor_App\Inc\Common_Cfg	Foc 硬件电路板参数定义头文件
motor_parameters_define.h	Motor_App\Inc\ Common _Cfg	Foc 电机参数定义头文件
drive_parameters_define.h	Motor_App\Inc\Foc_Cfg	Foc 电机算法控制参数定义头文件
protector_parameters_define.h	Motor_App\Inc\Foc_Cfg	Foc 保护策略参数定义头文件
autocur_regulator.h	Motor_App\Motor-Master\Foc\Foc_Lib\inc	Foc 电流调节器相关头文件
autospd_regulator.h	Motor_App\Motor-Master\Foc\Foc_Lib\inc	Foc 速度调节器相关头文件
coordnt.h	Motor_App\Motor-Master\Foc\Foc_Lib\inc	Foc 矢量变换相关头文件
cur_restruct.h	Motor_App\Motor-Master\Foc\Foc_Lib\inc	Foc 采样电流处理相关头文件
energy_save.h	Motor_App\Motor-Master\Foc\Foc_Lib\inc	节能控制算法相关头文件
flux_observer.h	Motor_App\Motor-Master\Foc\Foc_Lib\inc	Foc 磁链观测器相关头文件

文件名	文件路径	文件描述
flux_weakening.h	Motor_App\Motor-Master\Foc\Foc_Lib\inc	Foc 弱磁控制相关头文件
hall_sensor.h	Motor_App\Motor-Master\Foc\Foc_Lib\inc	Foc Hall 信号处理相关头文件
highfreqinj.h	Motor_App\Motor-Master\Foc\Foc_Lib\inc	高频注入法相关头文件
iqmath.h	Motor_App\Motor-Master\Foc\Foc_Lib\inc	电机算法数学库函数头文件
maxtorque_per_ampere.h	Motor_App\Motor-Master\Foc\Foc_Lib\inc	MTPA 算法相关头文件
mras.h	Motor_App\Motor-Master\Foc\Foc_Lib\inc	模型参考自适应算法相关头文件
param_identify.h	Motor_App\Motor-Master\Foc\Foc_Lib\inc	电机参数辨识相关头文件
param_init.h	Motor_App\Motor-Master\Foc\Foc_Lib\inc	Foc 电机控制参数标幺相关头文件
pid_regulator.h	Motor_App\Motor-Master\Foc\Foc_Lib\inc	Foc PID 控制相关头文件
power_dccur_estimate.h	Motor_App\Motor-Master\Foc\Foc_Lib\inc	功率及母线电流估算相关头文件
pulse_inject.h	Motor_App\Motor-Master\Foc\Foc_Lib\inc	无感 Foc 初始位置识别相关头文件
short_brake.h	Motor_App\Motor-Master\Foc\Foc_Lib\inc	快速刹车功能相关头文件
smopos.h	Motor_App\Motor-Master\Foc\Foc_Lib\inc	Foc 滑模观测器相关头文件
svgen.h	Motor_App\Motor-Master\Foc\Foc_Lib\inc	FOC SVPWM 相关头文件
version.h	Motor_App\Motor-Master\Foc\Foc_Lib\inc	电机 Foc 控制开发库版本相关头文件

文件名	文件路径	文件描述
vf_compensate.h	Motor_App\Motor-Master\Foc\Foc_Lib\inc	vf 控制辅助功能相关头文件
foc_control.lib foc_control.a	Motor_App\Motor-Master\Foc\Foc_Lib\src	电机 Foc 控制开发库

1.4 Motor_App 软件设计

1.4.1 Motor_App 架构

Motor_App 软件工程包括 Motor_App，MotorLib 等模块，其基本架构如图 1-3 所示。

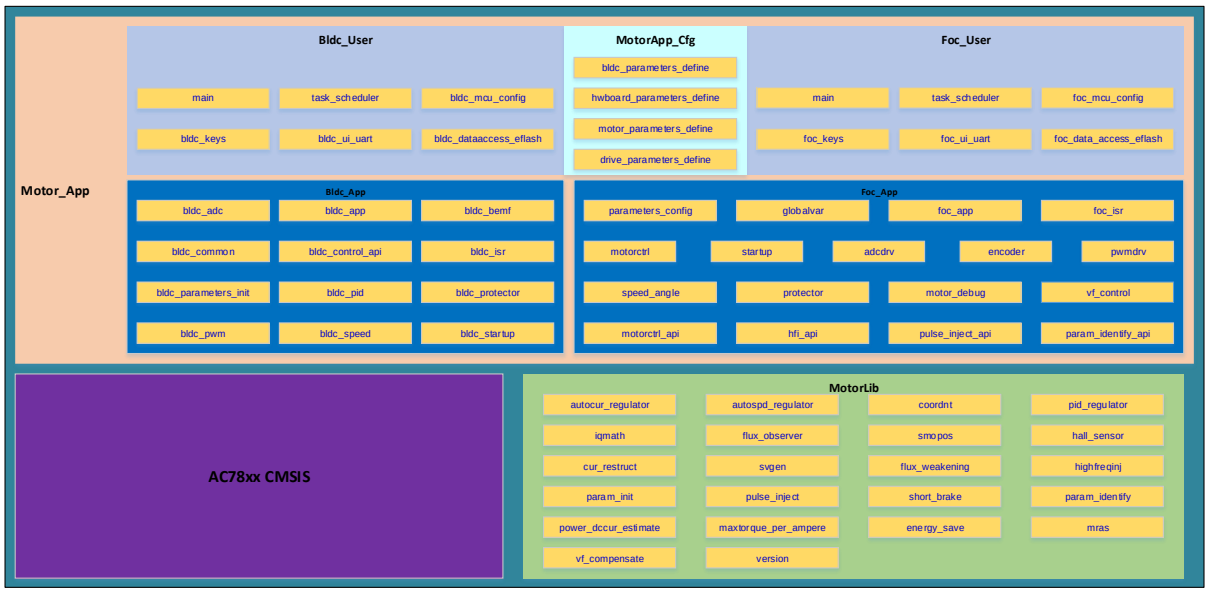

图 1-3 Motor_App 架构图

1.4.2 Motor_App 流程图

Motor_App 主程序流程如图 1-4 所示：

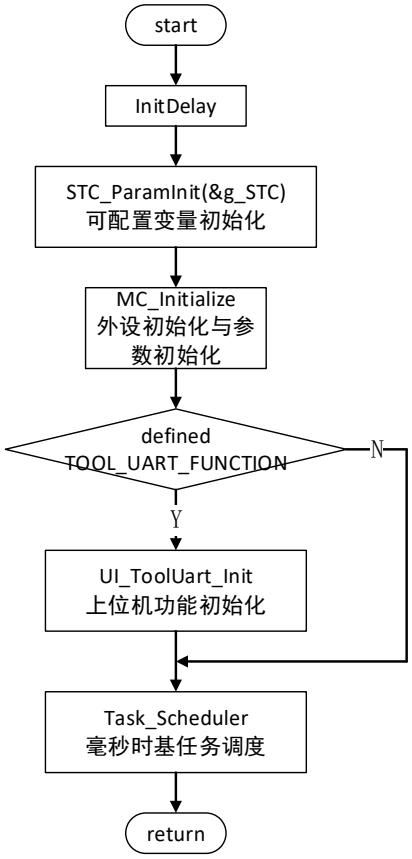

图 1-4 Motor_App 主程序流程图

1.4.3 while 主循环

while 主循环中任务流程图如图 1-5 所示：

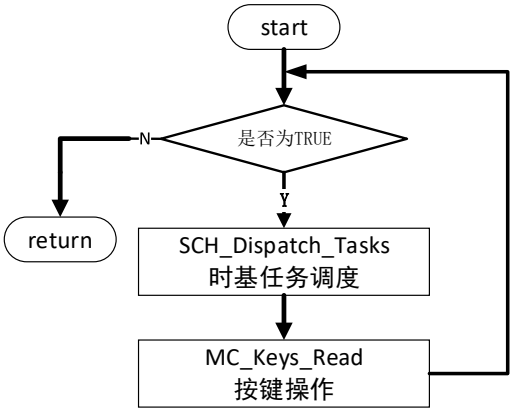

图 1-5 while 主循环任务流程

1.5 Motor_App 接口描述

1.5.1 Motor_App(API)

Motor_App 模块中 API 接口如表 1-2 和表 1-3 所示。

表 1-2 Foc_App 模块 API 接口表

API	说明
FOC_Parameters_Initialize	电机 FOC 控制全局变量初始化
FOC_Peripheral_Initialize	电机 FOC 控制外设初始化
FOC_PWM_FaultCallback	电机 1 硬件过流保护中断
FOC_PWM_FaultCallback_M2	电机 2 硬件过流保护中断
Timer0_Callback	定时器 0 中断回调函数，产生时基
Timer1_Callback	定时器 1 中断回调函数，用于电机 1 编码器 M_T 测速法计算任务
Timer2_Callback	定时器 2 中断回调函数，用于电机 2 编码器 M_T 测速法计算任务
FOC_PWDT_Callback	FOC 控制 PWDT 外设中断
Motor_Ctrl_CallBack	电机 1 FOC 控制 ADC 中断
Motor_Ctrl_CallBack_M2	电机 2 FOC 控制 ADC 中断
FOC_1ms_Task	电机 FOC 控制 1ms 时基任务，主状态机及速度环运算
FOC_2ms_Task	电机 FOC 控制 2ms 时基任务，上位机功能
MainStateMachine	电机 FOC 控制状态机
MC_Keys_Read	电机控制按键识别
Task_Scheduler	时基任务调度管理
ASR_SetSpeedTarget_Pu	设定速度环速度目标值
ASR_SpeedTargetRampCalc	速度斜坡计算
ASR_SpeedLoop_Process	速度环计算

API	说明
ASR_SpeedFbk_Update	获取速度环反馈
Get_ElecAngle	获取电机电角度
Debug_Dac_Watch	电机调试 DAC 输出送显

表 1-3 Bldc_App 模块 API 接口表

API	说明
BLDC_Parameters_Initialize	BLDC 控制参数初始化
BLDC_Peripheral_Initialize	BLDC 控制外设初始化
BLDC_Base_TimerCallBack	定时器 0 中断回调函数，产生方波控制时基
Timer1_Callback	定时器 1 中断回调函数，用于电机 1 无感反电动势检测与换相任务
Timer2_Callback	定时器 2 中断回调函数，用于电机 2 无感反电动势检测与换相任务
BLDC_Hall_PWDTCallback	BLDC 有感控制 PWDT 外设中断
BLDC_PWM_FaultCallback	电机 1 方波控制 PWM 硬件过流中断
BLDC_PWM_FaultCallback_M2	电机 2 方波控制 PWM 硬件过流中断
BLDC_ADC_Callback	电机 1 方波控制 ADC 中断
BLDC_ADC_Callback_M2	电机 2 方波控制 ADC 中断
BLDC_1ms_Task	电机 BLDC 控制 1ms 时基任务，主状态机及速度命令给定等
BLDC_2ms_Task	电机 BLDC 控制 2ms 时基任务，上位机功能
BldcStateMachine	电机 BLDC 控制状态机
BLDC_CurrentLoop_Calculate	电机 BLDC 控制电流环计算
BLDC_SpeedLoop_Calculate	电机 BLDC 控制速度环计算
BLDC_ASR_SetTargetPu	电机 BLDC 控制设定速度环目标值

API	说明
BLDC_ASR_FbkUpdate	电机 BLDC 控制获取速度环反馈
MC_Keys_Read	电机控制按键识别

说明：

具体函数介绍请参考 AC78xx 电机工程介绍文档《AC78xx Motor Driver》。

1.5.2 数据结构

Motor_App 工程中定义的结构体如表 1-4 所示。

表 1-4 Motor_App 中结构体

数据结构	说明
MOTOR_PARAM	电机参数结构体
MOTOR_PARAM_PU	电机参数标幺化结构体
HW_BOARD	电机控制器硬件电路参数结构体
FOC_VARS_CFG	FOC 配置参数结构体
FOC_VARS_CTRL	FOC 控制参数结构体
FOC_VARS	FOC 控制功能结构体
STC_PARAM	速度-转矩控制参数结构体
PROTECTOR	保护相关结构体
SPEED_RAMP_HANDLE	速度环速度斜坡参数结构体
START_UP_CFG	FOC 起动功能配置参数结构体
START_UP_CTRL	FOC 起动功能控制参数结构体
START_UP	FOC 起动控制参数结构体

数据结构	说明
BLDC_VARS_CFG	BLDC 配置参数结构体
BLDC_VARS_CTRL	BLDC 控制参数结构体
BLDC_VARS	BLDC 控制功能结构体
SPEED_RAMP_TYPE	BLDC 控制速度环速度斜坡参数结构体

说明：

具体函数介绍请参考 AC78xx 电机工程介绍文档《AC78xx Motor Driver》。

2 外设初始化设置

2.1 AC781x 系列 MCU 外设初始化设置

2.1.1 GPIO 初始化

2.1.1.1 PWM GPIO 初始化

AC781x 的 GPIO 具有 Multi-Function 功能，可以根据具体的 pinmux 表格将相应的 GPIO 设置为 PWM 输出。本文使用的 AC781x 支持 80-PIN 封装类型，表 2-1 给出这种封装类型下 GPIO 的设置参考。

表 2-1 AC781x 系列 80-PIN 封装 PWM2 模块 GPIO 定义

Module	PAD Name	BGA Ball Name	Function1	GPIO
PWM	PAD_PWM_FAULT1	PWM_FAULT1	PWM_FAULT1(I)	58
PWM	PAD_PWM2_CH4	PWM2_CH4	PWM2_CH4(I/O)	61
PWM	PAD_PWM2_CH5	PWM2_CH5	PWM2_CH5 (I/O)	62
PWM	PAD_PWM2_CH0	PWM2_CH0	PWM2_CH0(I/O)	31
PWM	PAD_PWM2_CH1	PWM2_CH1	PWM2_CH1(I/O)	32
PWM	PAD_PWM2_CH2	PWM2_CH2	PWM2_CH2(I/O)	33
PWM	PAD_PWM2_CH3	PWM2_CH3	PWM2_CH3(I/O)	34

目前 MotorApp 中所支持的所有方波 BLDC 和正弦波 FOC 模式，最终均通过 PWM2 模块发波来驱动控制电机，故 PWM2 模块的 GPIO 初始化一样，即使用 GPIO 的 Funtion1 将 GPIO 设置为 PWM 输出模式，在 PWM2 功能初始化起始部分配置，进行 PWM 功能初始化。

2.1.1.2 ADC GPIO 初始化

AC781x 有一个精度为 12bit 的 16 通道的 ADC，其引脚定义如表 2-2 所示。在使用的时候，需要结合实际的硬件来使能 GPIO 的模拟输入功能。

表 2-2 AC781x 电机 Demo 板 ADC 模块 GPIO 定义

Module	PAD Name	BGA Ball Name	Function1	GPIO
ADC	PAD_ADC_IN10	ADC_IN10	ADC_IN10(I)	6
ADC	PAD_ADC_IN12	ADC_IN12	ADC_IN12(I)	54
ADC	PAD_ADC_IN0	ADC_IN0	ADC_IN0(I)	7
ADC	PAD_ADC_IN1	ADC_IN1	ADC_IN1(I)	8
ADC	PAD_ADC_IN2	ADC_IN2	ADC_IN2(I)	9

Module	PAD Name	BGA Ball Name	Function1	GPIO
ADC	PAD_ADC_IN3	ADC_IN3	ADC_IN3(I)	10
ADC	PAD_ADC_IN4	ADC_IN4	ADC_IN4(I)	11
ADC	PAD_ADC_IN5	ADC_IN5	ADC_IN5(I)	12
ADC	PAD_ADC_IN6	ADC_IN6	ADC_IN6(I)	13
ADC	PAD_ADC_IN7	ADC_IN7	ADC_IN7(I)	14
ADC	PAD_ADC_IN8	ADC_IN8	ADC_IN8(I)	15
ADC	PAD_ADC_IN9	ADC_IN9	ADC_IN9(I)	16
ADC	PAD_ADC_IN11	ADC_IN11	ADC_IN11(I)	17
ADC	PAD_ADC_IN13	ADC_IN13	ADC_IN13(I)	55
ADC	PAD_ADC_IN14	ADC_IN14	ADC_IN14(I)	56
ADC	PAD_ADC_IN15	ADC_IN15	ADC_IN15(I)	57

在 ATC 的 AC781x 电机 Demo Board 上，FOC 算法使用的是 ADC_IN4，ADC_IN5，ADC_IN6 和 ADC_IN7 来采样 3 相的电流和母线电流，使用 ADC_IN8 采样母线电压；BLDC 无霍尔传感器算法需使用的是 ADC_IN0，ADC_IN1， ADC_IN2 和 ADC_IN3 来采样反电动势进行比较；因此 MotorApp 中需将以上使用的 ADC 各通道 GPIO 口进行初始化配置。在 MCU 上电后，根据配置的电机工作模式设置相应的 GPIO 口为 ADC 功能，进行 ADC 功能初始化。

2.1.1.3 PWDT GPIO 初始化

AC781x 的 PWDT 可用来捕获霍尔信号。本文中使用 GPIO24，GPIO20，GPIO19 作为 PWDT 输入通道。在带霍尔的 BLDC 和 FOC 模式下均需对 PWDT 的 GPIO 进行配置。在 Demo 板支持的相应电机工作模式启动初始化阶段，将他们设置为特定 PWDT 功能口，进行 PWDT 功能初始化。

2.1.1.4 Debug PWM GPIO 初始化

AC781x 的 PWM 可用来作为 DAC 输出观察变量，默认采用 PWM0 作为 Debug 通道。与 PWM GPIO 初始化同理，需在 Demo 板支持的各电机工作模式启动初始化阶段，将对应的 GPIO25，GPIO26 设置为 PWM 功能。

2.1.2 PWM 初始化

可以设置 AC781x 的 PWM 为组合(Combine)输出模式，此时，PWM2_CH0~PWM2_CH5 被分为 3 对 PWM，其中 PWM2_CH(n)~PWM2_CH(n+1)为 1 对(n = 0，2，4)。

每对 PWM 的输出可以设置为互补输出和非互补输出。

- 当设置为互补输出时候，Channel(n)和 Channel(n+1)输出的电平相反。
- 当互补输出模式禁用的时候，Channel(n)和 Channel(n+1)输出的电平相同。

Channel(n)的初始电平由 ELSR1：ELSR0 决定。

- 当 ELSR1：ELSR0 = 1:0 时候，当计数器计数到 CHnVR 时候，Channel(n)的输出变高，当计数器计数到 CH(n+1)VR 时，Channel(n)的输出变低。
- 当 ELSR1：ELSR0 = X:1 时候，当计数器计数到 CHnVR 时候，Channel(n)的输出变低，当计数器计数到 CH(n+1)VR 时，Channel(n)的输出变高。

PWM 的频率由 MCVR 决定，Duty 占空比由 CH(n+1)VR － CH(n)VR 决定。其配置输出 PWM 波形如图 2-1 所示。

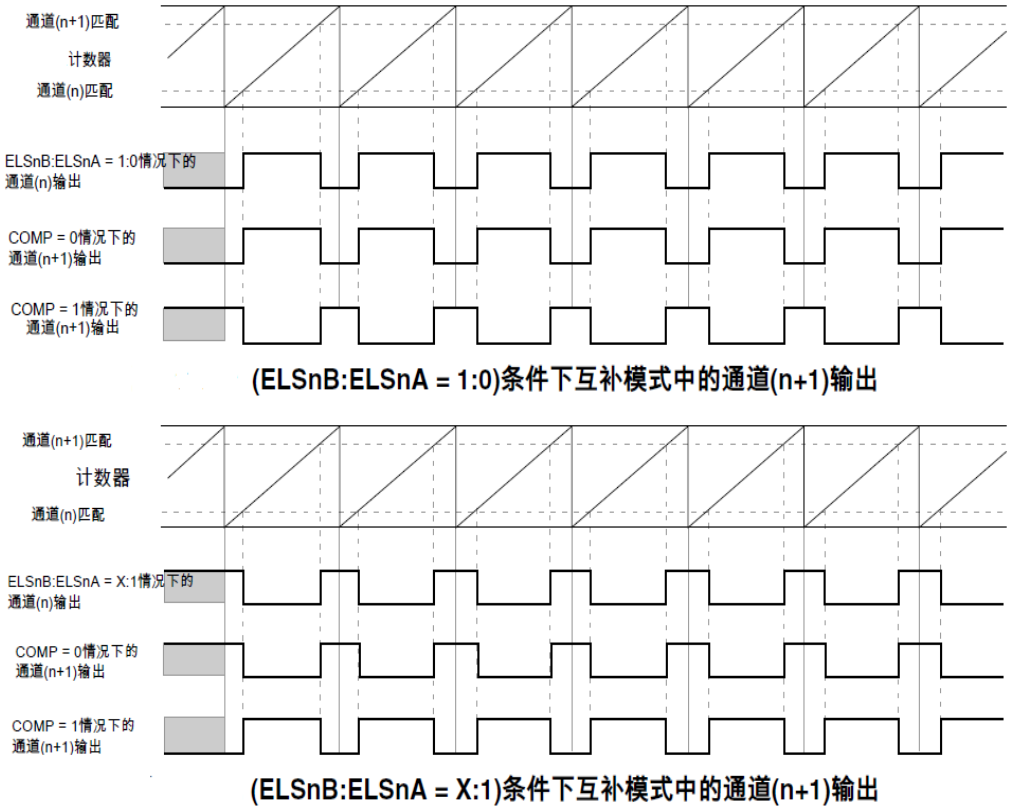

图 2-1 AC781x 电机 Demo 板 PWM 模块典型配置示意图

(1) 是否需要使用 PWM 的互补输出模式取决于实际的硬件电路，默认 Demo 板情况下 PWM 的互补模式输出是使能的。

以正弦波 FOC 算法为例，调用 FOC 算法开始后，即需要对 PWM 进行初始化配置，其初始化 API 接口为 void FOC_PWM_Initialize(void)函数。主要是将 PWM2 模块设置为互补模式输出，设置 PWM 产生周期，进行死区时间设置，使能 PWM2 模块工作，设置 PWM2 模块中断源及中断优先级，并调用中断回调函数。

Demo 板硬件环境为 GATE_DRIVER_HIGH_HIGH，即上、下开关管均为高电平有效。PWM 互补输出即可保证上、下桥臂不同时有效导通。但当硬件环境为 GATE_DRIVER_HIGH_LOW 或

GATE_DRIVER_LOW_HIGH 时，则需要禁止 PWM 互补输出功能，防止上、下桥臂同时有效导通，损坏系统硬件环境。

(2) 当电机根据指令停止或启动时，需对 PWM 输出进行重新初始化和更新，可使用 PWM 的输出屏蔽功能来使能/禁止 PWM 的输出。使能 PWM 的输出 API 为 void PWM_MaskAllChannels(void)，禁止 PWM 的输出 API 为 void PWM_ClearAllOutput(void)，详细代码可参考 MotorApp 工程。

2.1.3 ADC 初始化

AC781x 的 ADC 分为规则组和注入组，使用规则组来进行母线电压的采样，使用注入组来实现母线电流和相电流的采样。

规则组采用软件触发模式，AC781x ADC 的数据支持 DMA 传输，在规则组采样完后，可以使用 DMA 将数据从 ADC 寄存器搬到 Memory 中。

注入组采用外部触发模式，在 PWM 下桥臂导通的中心开始 ADC 采样，采集相电流。注入组 AD 采集完后，触发 ADC 中断，在 ADC 中断中根据当前采样的相电流执行 FOC 算法。

FOC 算法中需用 ADC 采样三相电流，因此在 MotorApp 配置为 FOC 模式，由按键启动电机运转执行 FOC 算法时，需先对 ADC 模块进行初始化配置，其初始化 API 接口为 void FOC_ADC_Initialize (void) 函数。其初始化主要包括使能 ADC 模块，配置 ADC 中断回调函数，根据硬件电路设置 ADC 注入组通道，采样延迟设置，配置 ADC 中断源及中断优先级，使能 ADC 中断等。

当 MotorApp 配置为带霍尔 BLDC 模式，执行 BLDC 方波控制时，BLDC 必须进行 ADC 模块进行初始化使能，用于母线电流的采集，其初始化 API 接口为 void BLDC_ADC_Init (void) 函数。在该接口函数中进行使能 ADC，设置母线电流采样 ADC 通道及采样延迟时间，配置 ADC 中断回调函数，配置 ADC 中断源及中断优先级等操作。

2.1.4 PWM 触发 ADC 功能初始化

AC781x 的 ADC 有 8 个可用的外部触发源，其配置寄存器定义见表 2-3 所示。其中 PWM2 的 Init 和 Match 都能够产生触发源触发 ADC 转换。

表 2-3 AC781x 电机 Demo 板 ADC 外部触发源配置表

ADHWT1	ADC hardware trigger1
000	RTC overflow
001	PWM0 init trigger
010	PWM2 init trigger with 8-bit programmable delay
011	PWM0 match trigger with 8-bit programmable delay
100	TIMER ch0 overflow

ADHWT1	ADC hardware trigger1
101	TIMER ch1 overflow
110	ACMP0 out
111	ACMP1 out

在 FOC 运算中，由于 PWM 的产生方式为边沿对齐，因此需要采用 PWM2 的 Init 来触发 ADC 转换，在下半桥导通的中心位置开始 ADC 的采样，如图 2-2 所示。

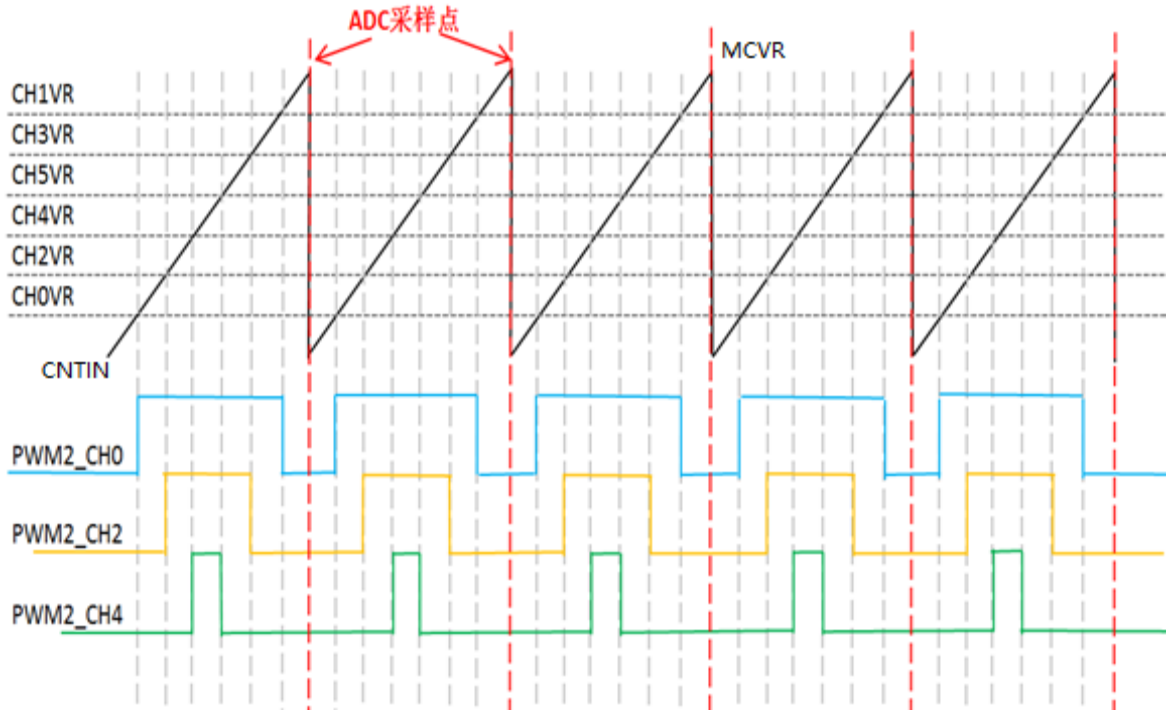

图 2-2 AC781x 电机 Demo 板 FOC 模式 ADC 中断触发示意图

因此需要做以下设置：

```
CTU_ModuleEnable();                          /*CTU 使能*/
CTU_TriggerAdcInjectGroupByPwm2Init();       /*配置 PWM2 触发 ADC 中断*/
CTU_SetBusDivide(0);                         /*根据算法执行需要，配置 PWM 分频系数*/
CTU_SetPwm2Trig2AdcDelayCnt(0);              /*选择 PWM 触发 ADC 中断采集延迟时间*/
PWM_SetExternalTrigger(PWM2, (1 << 6));      /*使能 PWM 的 Init 触发源*/
```

2.1.5 TIM 初始化

在电机控制中，还会用到 Timer 定时器来定时处理一些任务。这时候，就需要用的 AC781x 的 Timer 模块，AC781x 有 8 个 Timer 通道，其中 Timer0~Timer1 为 32-bit 的计数器，Timer2~Timer7 为 16-bit 的计数器，可以根据实际的定时时间来选择合适的 Timer。

Motor_App 软件工程中采用 Timer0 定时作为 BLDC 方波有感控制模式的时基，初始化接口 API 为 void BLDC_TIM_Initialize(void)，中断回调接口为 void BLDC_Base_TimerCallback (void)；而在无感 BLDC 方波控制中，Timer1 被初始化为无感换相定时器使用，初始化接口 API 为 void BLDC_Bemf_TimerInit (void)。Motor_App 采用 Timer0 定时作为 FOC 控制模式的时基，初始化接口 API 为 void FOC_TIM_Initialize(void)，中断回调接口为 void Timer0_Callback (void)；Timer1 初始化为正交编码器 M_T 测速法运算定时器，初始化接口 API 为 void Encoder_TIM_Initialize (void)。

2.1.6 PWDT 初始化

在带霍尔传感器的电机控制中， AC781x 通过 PWDT 模块来检测霍尔脉冲信号，并同时记录正负脉冲宽度计算电机转子速度。作为电流环转子位置和速度环转子速度的测量手段，在电机控制正式闭环运算之前必须初始化 PWDT 模块保证工作正常。BLDC 方波和带霍尔 FOC 控制中均用到了 PWDT 模块，他们对 PWDT 的初始化设置略有差异，主要体现在两种控制模式下对 PWDT 的时钟频率需求不一样。BLDC 方波有感控制中对 PWDT 的初始化接口 API 为 void BLDC_Hall_PWDT_Initialize(void)，BLDC 方波无感控制中对 PWDT 的初始化接口 API 为 void BLDC_Bemf_PWDT_Initialize(void)，带霍尔 FOC 控制中对 PWDT 的初始化接口 API 为 void FOC_PWDT_Initialize(void)，均在电机开始启动之前进行初始化设置。

2.1.7 ACMP 初始化

在无感 BLDC 的电机方波控制中， AC781x 无法通过 PWDT 模块直接检测霍尔脉冲信号，需先进行 ACMP 模块的初始化配置，使能 ACMP 模块正常工作后，ACMP 会对输入的三相反电动势与电机中性点电压进行轮询模拟比较，得到各相反电动势过零点，进而模拟霍尔信号通过 CTU 输出至 PWDT 采集。带无感 BLDC 的电机方波控制中对 ACMP 的初始化接口 API 为 void BLDC_Bemf_Acmp_Init (void)，需在无感 BLDC 电机控制启动之前进行初始化设置。

2.1.8 QEI 初始化

对于使用光电编码器作为位置反馈信号的 FOC 控制应用，可以使用 AC781x 的 QEI 模块对码盘输出的正交编码信号进行捕捉采集，通过捕捉电机转动过程中码盘输出脉冲数来测量电机转子位置。AC781x 库函数中已定义了正交编码器的底层函数，用户在使用过程中可直接调用编码器的初始化库函数 void PWM_QEIInit (PWM_Type *PWMx, QEI_ConfigType *qeiconfig)对正交编码器接口进行初始化。正

交编码 FOC 控制模式中对 QEI 的初始化接口 API 为 void FOC_Encoder_Initialize(void)，需在电机开始 FOC 闭环运算之前进行初始化设置。

2.1.9 DEBUG PWM 初始化

在电机控制中，需要通过 DAC 来观察变量，可以通过 AC781x 的 PWM 来模拟，是 PWM 作为 DAC 输出观察变量，本文中 PWM0 作为 debug 通道，其初始化 API 为 void MC_DEBUG_Initialize(void)，详细代码可参考 MotorApp 工程。

2.2 AC780x 系列 MCU 外设初始化设置

2.2.1 GPIO 初始化

2.2.1.1 PWM GPIO 初始化

AC780x 的 GPIO 具有 Multi-Function 功能，可以根据具体的 pinmux 表格将相应的 GPIO 设置为 PWM 输出。以 AC780x 的 48-PIN 封装类型为例，表 2-4 给出这种封装类型下 GPIO 的设置参考。

表 2-4 AC780x 系列 48-PIN 封装 PWM1 模块 GPIO 定义

Module	PAD Name	BGA Ball Name	Function1	GPIO
PWM	PAD_PB4	PWM1_PB4	PWM1_CH1(I/O)	23
PWM	PAD_PB5	PWM1_PB5	PWM1_CH0(I/O)	24
PWM	PAD_PB6	PWM1_PB6	PWM1_FLT0(I/O)	33
PWM	PAD_PB7	PWM1_PB7	PWM1_CH3(I/O)	35
PWM	PAD_PB8	PWM1_PB8	PWM1_CH2(I/O)	36
PWM	PAD_PB9	PWM1_PB9	PWM1_CH5(I/O)	43
PWM	PAD_PB10	PWM1_PB10	PWM1_CH4(I/O)	44

目前 MotorApp 中所支持的所有方波 BLDC 和正弦波 FOC 模式，最终均通过 PWM1 模块发波来驱动控制电机，故 PWM1 模块的 GPIO 初始化一样，使用 GPIO 的 Funtion1 将 GPIO 设置为 PWM 输出，在 PWM1 功能初始化起始部分配置，进行 PWM 功能初始化。

2.2.1.2 ADC GPIO 初始化

AC780x 有一个精度为 12bit 的 14 通道的 ADC，其引脚定义如表 2-5 所示。在使用的时候，需要结合实际的硬件来使能 GPIO 的模拟输出功能。

表 2-5 AC780x 电机 Demo 板 ADC 模块 GPIO 定义

Module	PAD Name	BGA Ball Name	Function2	GPIO
ADC	PAD_PA2	PA2	ADC_IN8(I)	2
ADC	PAD_PA3	PA3	ADC_IN7(I)	3
ADC	PAD_PA4	PA4	ADC_IN6(I)/ACMP_IN6(I)	4
ADC	PAD_PA5	PA5	ADC_IN5(I)/ACMP_IN5(I)	5
ADC	PAD_PC0	PC0	ADC_IN10(I)	32
ADC	PAD_PC1	PC1	ADC_IN9(I)	33
ADC	PAD_PA7	PA7	ADC_IN4(I)/ACMP_IN4(I)	7

在 ATC 电机 Demo Board 上，FOC 算法使用的是 ADC_IN4，ADC_IN5, ADC_IN6 和 ADC_IN7 来采样 3 相的电流和母线电流，使用 ADC_IN8 采样母线电压；MotorApp 中将使用的 ADC 各通道的 GPIO 口为 ADC 功能，进行 ADC 功能初始化

2.2.1.3 PWDT GPIO 初始化

AC780x 的 PWDT 可用来捕获霍尔信号。本文中使用 GPIO38, GPIO39, GPIO40 作为 PWDT 输入通道。在带霍尔的 BLDC 和 FOC 模式下均需对 PWDT 的 GPIO 的初始化。当电机采用有感控制且传感器为霍尔时，需在 Demo 板支持的相应电机工作模式启动初始化阶段调用执行，将他们设置为特定 PWDT 功能口。

2.2.1.4 Debug PWM GPIO 初始化

AC780x 的 PWM 可用来作为 DAC 输出观察变量，本文中使用 PWM0 作为 debug 通道。只需在 Demo 板支持的所有电机工作模式启动初始化阶段调用一次即可。

2.2.2 PWM 初始化

可以设置 AC780x 的 PWM 为组合(Combine)输出模式，此时，PWM1_CH0~PWM1_CH5 被分为 3 对 PWM，其中 PWM1_CH(n)~PWM1_CH(n+1)为 1 对(n = 0，2，4)。

配置 pwmConfig.countMode = PWM_UP_DOWN_COUNT 时为向上-向下计数模式。

每对 PWM 的输出可以设置为互补输出和非互补输出。

- 当设置为互补输出时候，Channel(n)和 Channel(n+1)输出的电平相反。
- 当互补输出模式禁用的时候，Channel(n)和 Channel(n+1)输出的电平相同。

Channel(n)的初始电平由 ELSR1：ELSR0 决定。

- 当 ELSR1：ELSR0 = 1:0 时候，为 High-true 脉冲（在 channel(n)匹配时清除，在 channel(n+1)匹配时置位）。
- 当 ELSR1：ELSR0 = X:1 时候，为 Low-true 脉冲（在 channel(n)匹配时清除，在 channel(n+1)匹配时置位）。

PWM 的频率由 MCVR 决定，Duty 占空比由 CH(n+1)VR － CH(n)VR 决定：

周期= 2 * (MCVR - CNTIN) * PWM 计数器时钟周期

脉冲宽度（占空比）= 2 * (|CH(n+1)V – CH(n)V|) * PWM 计数器时钟周期

其配置输出 PWM 波形如图 2-3 所示。

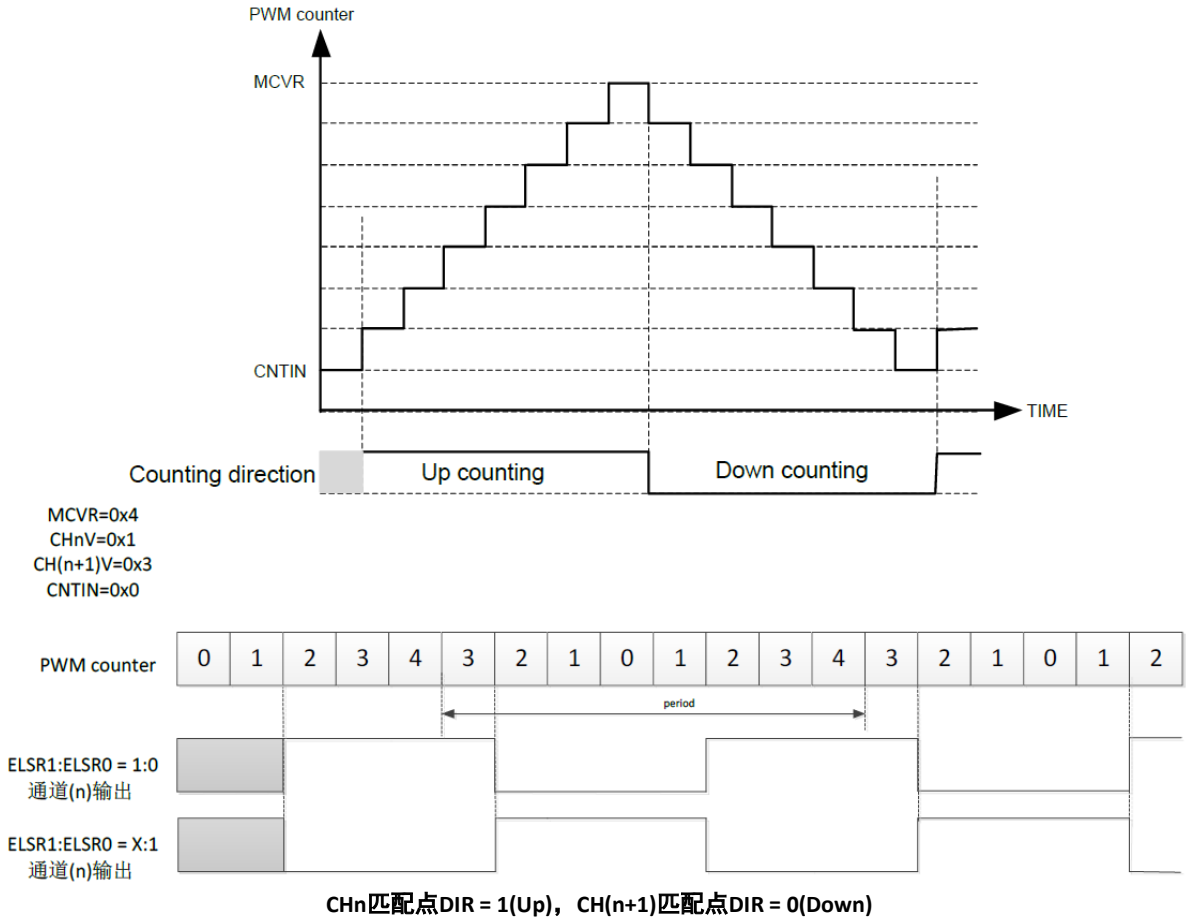

图 2-3 AC780x 电机 Demo 板 PWM 模块典型配置示意图

(1) 是否需要使用 PWM 的互补输出模式取决于实际的硬件电路，默认 Demo 板情况下 PWM 的互补模式输出是使能的。

以正弦波 FOC 算法为例，调用 FOC 算法开始后，即需要对 PWM 进行初始化配置，其初始化 API 接口为 void FOC_PWM_Initialize(void)函数。主要是将 PWM1 模块设置为互补模式输出，设置

PWM 产生周期，进行死区时间设置，使能 PWM1 模块工作，设置 PWM1 模块中断源及中断优先级，并调用中断回调函数。

Demo 板硬件环境为 GATE_DRIVER_HIGH_HIGH，即上、下开关管均为高电平有效。PWM 互补输出即可保证上、下桥臂不同时有效导通。但当硬件环境为 GATE_DRIVER_HIGH_LOW 或 GATE_DRIVER_LOW_HIGH 时，则需要禁止 PWM 互补输出功能，防止上、下桥臂同时有效导通，损坏系统硬件环境。

(2) 当电机根据指令停止或启动时，需对 PWM 输出进行重新初始化和更新，可以使用 PWM 的输出屏蔽功能来使能/禁止 PWM 的输出。使能/禁止 PWM 的输出 API 为 void PWM_Output(uint8_t enable)，详细代码可参考 MotorApp 工程。

2.2.3 ADC 初始化

AC780x 的 ADC 分为规则组和注入组，使用规则组来进行母线电压的采样，使用注入组来实现母线电流和相电流的采样。

规则组采用软件触发模式，AC780x ADC 的数据支持 DMA 传输，在规则组采样完后，可以使用 DMA 将数据从 ADC 寄存器搬到 Memory 中。

注入组采用外部触发模式，在 PWM 下桥臂导通的中心开始 ADC 采样，采集相电流。注入组 AD 采集完后，触发 ADC 中断，在 ADC 中断中根据当前采样的相电流执行 FOC 算法。

FOC 算法中需用 ADC 采样三相电流，因此在 MotorApp 配置为 FOC 模式，由按键启动电机运转执行 FOC 算法时，需先对 ADC 模块进行初始化配置，其初始化 API 接口为 void FOC_ADC_Initialize (void) 函数。其初始化主要包括使能 ADC 模块，配置 ADC 中断回调函数，根据硬件电路设置 ADC 注入组通道，采样延迟设置，配置 ADC 中断源及中断优先级，使能 ADC 中断等。

当 MotorApp 配置为带霍尔 BLDC 模式，执行 BLDC 方波控制时，BLDC 必须进行 ADC 模块进行初始化使能，用于母线电流的采集，其初始化 API 接口为 void BLDC_ADC_Init (void) 函数。在该接口函数中进行使能 ADC，设置母线电流采样 ADC 通道，配置 ADC 中断回调函数，配置 ADC 中断源及中断优先级等操作。

2.2.4 PWM 触发 ADC 功能初始化

AC780x 的 ADC 有 8 个可用的外部触发源，其配置寄存器定义如表 2-6 所示。其中 PWM1 的初始化和匹配都能够产生触发源触发 ADC 转换。

表 2-6 AC780x 电机 Demo 板 ADC 外部触发源配置表

ADHWT1	ADC 注入组硬件触发源
000	RTC 溢出作为 ADC 硬件触发源
001	PWM0 初始化触发，具有 8 位可编程计数器延迟

ADHWT1	ADC 注入组硬件触发源
010	PWM0 匹配触发，具有 8 位可编程计数器延迟
011	PWM1 初始化触发，具有 8 位可编程计数器延迟
100	PWM1 匹配触发，具有 8 位可编程计数器延迟
101	TIMER 通道 0 溢出作为 ADC 硬件触发
110	TIMER 通道 1 溢出作为 ADC 硬件触发
111	ACMP0 输出作为 ADC 硬件触发

说明：

选择 ADC 注入硬件触发源，所有触发源在上升沿开始转换。

在 FOC 运算中，由于 PWM 的产生方式为边沿对齐，因此需要采用 PWM1 的 Init 来触发 ADC 转换，在下半桥导通的中心位置开始 ADC 的采样，如图 2-4 所示。

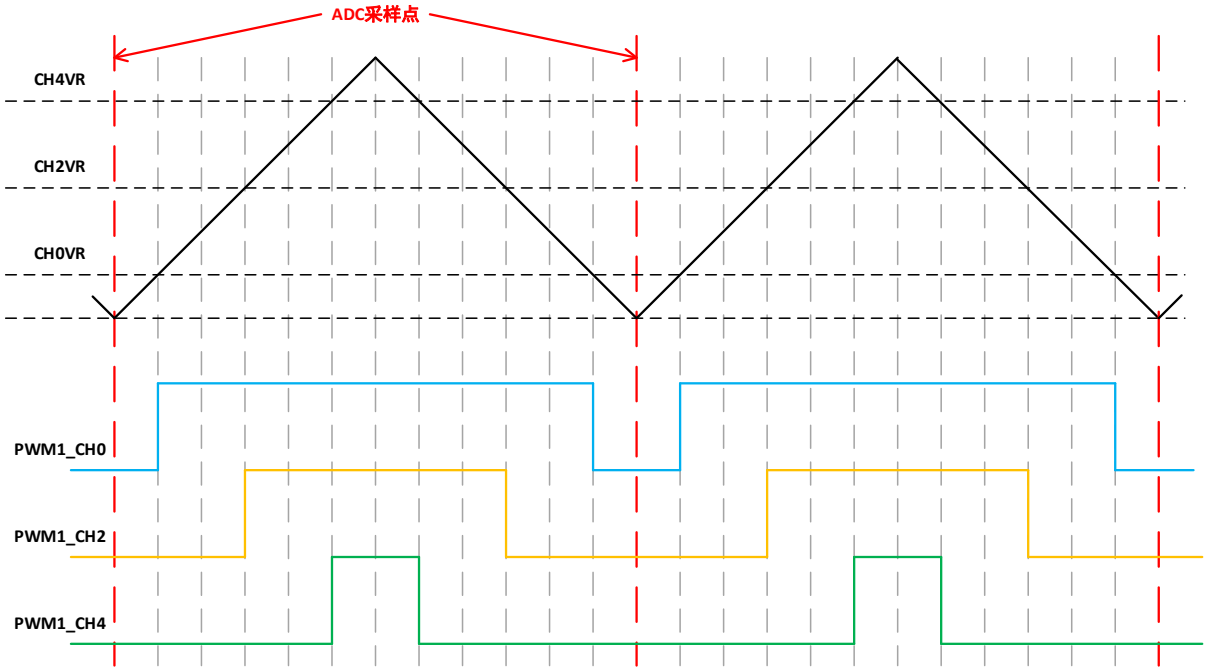

图 2-4 AC780x 电机 Demo 板 FOC 模式 ADC 中断触发示意图

需要做以下设置：

(1) 配置 PWM 模块：

```
PWM_CombineChConfig     chConfig[4];
PWM_ModulationConfigType pwmConfig;
PWM_ConfigType          config;
///< 3相上下桥臂配置
chConfig[0].pairChannel = PWM_CH_0;          ///< 设置输出通道：上臂PWM_CH_0，下臂PWM_CH_1
chConfig[0].ch1stValue = 1;
chConfig[0].ch2ndValue = 1;
chConfig[0].levelMode = PWM_HIGH_TRUE;    ///< PWM电平模式
chConfig[0].deadtimeEn = ENABLE;              ///< 使能死区补偿
chConfig[0].complementEn = ENABLE;          ///< 使能互补模式
chConfig[0].ch1stMatchDir = PWM_MATCH_DIR_UP;       ///< 通道1匹配方向
chConfig[0].ch2ndMatchDir = PWM_MATCH_DIR_DOWN; ///< 通道2匹配方向
……
chConfig[2].ch2ndMatchDir = PWM_MATCH_DIR_DOWN;

pwmConfig.initTriggerEn = ENABLE;             ///< 使能InitTrigger
pwmConfig.countMode = PWM_COUNT_UP_DOWN_COUNT;  ///< 上升下降计数
pwmConfig.combineChannelNum = 3;          ///< 3组通道组合输出
pwmConfig.combineChConfig = chConfig;
pwmConfig.deadtime = PWM_DEAD_TIME;    ///< 设置死区时间
pwmConfig.deadtimePsc = PWM_DEADTIME_DIVID_1;   ///< 死区时间分频系数
///< 设置通道输出极性
pwmConfig.combineChConfig->ch1stPolarity = PWM_OUTPUT_POLARITY_ACTIVE_HIGH;
pwmConfig.combineChConfig->ch2ndPolarity = PWM_OUTPUT_POLARITY_ACTIVE_LOW;
```

其中，3 相上下桥臂配置内容一致，只需参考 chConfig[0]的配置，将 chConfig 数组中元素累加至 2。

(2) 配置 CTU 模块：

```
CTU_ConfigType          ctuConfig;                  ///< CTU模块配置
memset(&ctuConfig, 0, sizeof(CTU_ConfigType));  ///< 清零
ctuConfig.clkPsc = CTU_CLK_PRESCALER_1;         ///< 时钟分频系数
ctuConfig.adcInjectTriggerSource = CTU_TRIGGER_ADC_PWM1_INIT;  ///< ADC注入组触发源
ctuConfig.delay1Time = 0;                       ///< 设置延时
CTU_Init(&ctuConfig);                           ///< CTU初始化
```

将 CTU 模块中 ADC 注入组触发源设置为 ADC_PWM1_INIT 初始触发，即是在 PWM1 模块上臂的计数值为 0 时触发 ADC 模块的注入组转换。

(3) 配置 ADC 模块：

```
ADC_ConfigType          tempAdcConfig;              ///< ADC模块配置
ADC_ConfigType*         adcConfig;
adcConfig = &tempAdcConfig;
memset(adcConfig, 0x0, sizeof(ADC_ConfigType)); ///< 清零
ckgen_Enable(CLK_ADC0, ENABLE);                  ///< 设置时钟
adcConfig->scanModeEn = ENABLE;                  ///< 扫描模式使能
adcConfig->continousModeEn = DISABLE;            ///< 禁用连续模式
adcConfig->regularDiscontinousModeEn = DISABLE; ///< 禁用规则组中止模式
adcConfig->injectDiscontinousModeEn = DISABLE;   ///< 禁用注入组间隔模式
adcConfig->injectAutoModeEn = DISABLE;           ///< 禁用注入组自动模式
……
```

(4) 输出 PWM 计数值：

```
PWM_SetChannelCountValue(PWM1, PWM_CH_0, PWM_PERIOD_VALUE - Ta);   ///< 设置U相上臂PWM计数值
PWM_SetChannelCountValue(PWM1, PWM_CH_1, PWM_PERIOD_VALUE - Ta);   ///< 设置U相下臂PWM计数值
PWM_SetChannelCountValue(PWM1, PWM_CH_2, PWM_PERIOD_VALUE - Tb);   ///< 设置V相上臂PWM计数值
PWM_SetChannelCountValue(PWM1, PWM_CH_3, PWM_PERIOD_VALUE - Tb);   ///< 设置V相下臂PWM计数值
PWM_SetChannelCountValue(PWM1, PWM_CH_4, PWM_PERIOD_VALUE - Tc);   ///< 设置W相上臂PWM计数值
PWM_SetChannelCountValue(PWM1, PWM_CH_5, PWM_PERIOD_VALUE - Tc);   ///< 设置W相下臂PWM计数值
```

将电机控制 FOC 算法 SVPWM 功能产生的三相桥臂 PWM 计数值更新到 PWM 模块寄存器。在 PWM_HIGH_TRUE 高有效条件下，上臂在 PWM 计数值高于寄存器中写入的通道值(PWM_PREIOD_VALUE – Tx)时产生高电平，下臂与上臂互补，产生低电平。

2.2.5 TIM 初始化

在电机控制中，还会用到 Timer 定时器来定时处理一些任务，这时候，就需要用的 AC780x 的 Timer 模块，AC780x 有 4 个 32-bit 的 Timer 通道，可以根据实际的定时时间来选择合适的 Timer。

Motor_App 软件工程中采用 Timer0 定时作为 BLDC 方波控制模式的时基，初始化接口 API 为 void BLDC_TIM_Initialize(void)，中断回调接口为 void BLDC_Base_TimerCallBack (void *device, uint32_t wpara, uint32_t lpara)；在无感 BLDC 方波控制中， Timer1 作为换相定时器，初始化接口 API 为 void BLDC_Bemf_TimerInit (void)。

Motor_App 采用 Timer0 定时作为 FOC 控制模式的毫秒任务时基，初始化接口 API 为 void FOC_TIM_Initialize(void)，中断回调接口为 void Timer0_Callback (void *device, uint32_t wpara, uint32_t lpara)。Timer1 初始化为正交编码器 M_T 测速法运算定时器，初始化接口 API 为 void Encoder_TIM_Initialize (void)。

2.2.6 PWDT 初始化

在带霍尔传感器的电机控制中， AC780x 通过 PWDT 模块来检测霍尔脉冲信号，并同时记录正负脉冲宽度计算电机转子速度。作为电流环转子位置和速度环转子速度的测量手段，在电机控制正式闭环运算之前必须初始化 PWDT 模块保证工作正常。

BLDC 方波和带霍尔 FOC 控制中均用到了 PWDT 模块，他们对 PWDT 的初始化设置略有差异，主要体现在两种控制模式下对 PWDT 的时钟频率需求不一样。

- BLDC 方波有感控制中对 PWDT 的初始化接口 API 为 void BLDC_Hall_PWDT_Initialize(void)
- BLDC 方波无感控制中对 PWDT 的初始化接口 API 为 void BLDC_Bemf_PWDT_Initialize (void)；
- 带霍尔 FOC 控制中对 PWDT 的初始化接口 API 为 void FOC_PWDT_Initialize(void)，均在电机开始启动之前进行初始化设置。

2.2.7 ACMP 初始化

在无感 BLDC 的电机方波控制中， AC780x 无法通过使能 ACMP 模块正常工作后，由 ACMP 对输入的相反电动势与电机中点电压进行模拟比较，得到各相反电动势过零点。带无感 BLDC 的电机方波控制中对 ACMP 的初始化接口 API 为 void BLDC_Bemf_ACMP_Init (void)，需在无感 BLDC 电机控制启动之前进行初始化设置。

2.2.8 QDI 初始化

对于使用光电编码器作为位置反馈信号的 FOC 控制应用，可以使用 AC780x 的 QDI 模块对码盘输出的正交编码信号进行捕捉采集，通过捕捉电机转动过程中码盘输出脉冲数来测量电机转子位置。AC780x 正交编码 FOC 控制模式中对 QDI 的初始化接口 API 为 void FOC_Encoder_Initialize(void)，需在电机开始 FOC 闭环运算之前进行初始化设置。

2.2.9 DEBUG PWM 初始化

在电机控制中，需要通过 DAC 来观察变量，可以通过 AC780x 的 PWM 来模拟，使 PWM 作为 DAC 输出观察变量，本文中 PWM0 作为 debug 通道，其初始化 API 为 void MC_DEBUG_Initialize(void)，详细代码可参考 MotorApp 工程。

2.3 AC7840x 系列 MCU 外设初始化设置

2.3.1 GPIO 初始化

与 AC780x、AC781x 系列相同，AC7840x 系列中各模块 GPIO 的初始化均在模块初始化起始部分配置，在 MCU 上电后，分别设置各模块 IO 口功能，详细代码可参考 MotorApp 工程。

2.3.2 PWM 初始化

AC7840x 系列 PWM 模块集成了霍尔，编码器等电机传感器信号捕获采集功能，本章节统一说明。

2.3.2.1 发波 PWM 初始化

AC7840x 的发波 PWM 初始化基本配置与 AC780x 系列一致，此处不再赘叙。AC7840x 的发波 PWM 初始化配置与 AC780x 系列的区别在于 AC7840x 支持双电机控制应用。在 AC7840x 单电机 Demo 板中，PWM3 的 Channel 0~5 作为电机控制发波 PWM 功能使用；在 AC7840x 双电机 Demo 板中，PWM2、PWM3 的 Channel 0~5 分别作为两台电机控制发波 PWM 功能使用。

- 单电机应用 BLDC 方波控制中对发波 PWM 的初始化接口为 void BLDC_PWM_Initialize(void)；双电机应用 BLDC 方波控制中对发波 PWM 的初始化接口为 void BLDC_PWM_Initialize(PWM_Type *PWMx)。
- 单电机应用 FOC 控制中对发波 PWM 的初始化接口 API 为 void FOC_PWM_Initialize(void)；双电机应用 FOC 控制中对发波 PWM 的初始化接口 API 为 void FOC_PWM_Initialize(void)和 void FOC_PWM_Initialize_M2(void)。
- 双电机应用中，若两个发波 PWM 模块之间计数同步，则会同时触发 ADC 中断导致两个中断任务都无法正常执行。因此，采用系统延时 udelay(unit32_t us)将两个发波 PWM 模块计数时序错开，确保分时执行 PWM 触发的 ADC 中断任务。

2.3.2.2 Hall 捕获 PWM 初始化

AC7840x 的 PWM 模块集成了双边沿捕获模式，用于支持 Hall 信号的捕获测量。在 AC7840x 单电机 Demo 板中，PWM0 作为霍尔捕获 PWM 功能使用；在 AC7840x 双电机 Demo 板中，PWM0、PWM1 分别作为两台电机霍尔捕获 PWM 功能使用。Hall 捕获 PWM 模块初始化时，只初始化 Channel 0~3 作为霍尔信号输入接口。PWM 工作模式配置为输入捕获，通道捕获模式需配置为双边沿捕获模式，同时使能霍尔测量模式，以便于支持 Hall 脉宽和频率的捕获和测量。

- 单电机应用 BLDC 方波控制中对 Hall 捕获 PWM 的初始化接口 API 为 void BLDC_Hall_Initialize(void)；双电机应用 BLDC 方波控制中对 Hall 捕获 PWM 的初始化接口为 void BLDC_Hall_Initialize(void)和 void BLDC_PWM_Initialize_M2(void)。
- 单电机应用 FOC 控制中对 Hall 捕获 PWM 的初始化接口 API 为 void FOC_Hall_Initialize(void)；双电机应用 FOC 控制中对 Hall 捕获 PWM 的初始化接口 API 为 void FOC_Hall_Initialize(void)和 void FOC_Hall_Initialize_M2(void)。

2.3.2.3 Encoder 捕获 PWM 初始化

AC7840x 的 PWM 模块集成了正交编码捕获模式，用于支持 Encoder 信号的捕获测量。在 AC7840x 单电机 Demo 板中，PWM1 的 Channel 0~3 作为正交编码捕获功能使用；在 AC7840x 双电机 Demo 板中，PWM0、PWM1 分别作为两台电机正交编码器捕获功能使用。PWM 正交编码捕获功能的初始化过程与 AC780x 系列相同，详细代码可参考 AC780x 系列。

单电机应用 FOC 控制中对 Encoder 捕获 PWM 的初始化接口为 void FOC_Encoder_Initialize(void)；双电机应用 FOC 控制中对 Encoder 捕获 PWM 的初始化接口分别为 void FOC_Encoder_Initialize(void)和 void FOC_Encoder_Initialize_M2(void)。

2.3.2.4 Debug PWM 初始化

在电机控制中，需要通过 DAC 来观察变量，可以通过 AC7840x 的 PWM 来模拟，使 PWM 作为 DAC 输出观察变量。在 AC7840x 单电机 Demo 板中，PWM2 的 Channel 0~1 作为 PWM 输出 Debug 功能使用；在 AC7840x 双电机 Demo 板中，PWM4、PWM5 的 Channel 0~1 分别作为两台电机 PWM 输出 Debug 功能使用。Debug PWM 功能的初始化过程与 AC780x 系列相同，详细代码可参考 AC780x 系列。BLDC 方波控制和 FOC 控制使用公用的 Debug PWM 初始化接口 API 为 void MC_DEBUG_PWM_Initialize(PWM_Type *PWMx)。

2.3.3 ADC 初始化

AC7840x 拥有两个 ADC 单元(ADC0 和 ADC1)，每个 ADC 可分为规则组和注入组，使用规则组来进行母线电压的采样，使用注入组来实现母线电流和相电流的采样。

规则组采用软件触发模式，AC7840x ADC 的数据支持 DMA 传输，在规则组采样完后，可以使用 DMA 将数据从 ADC 寄存器搬到 Memory 中。

注入组采用外部触发模式，在 PWM 下桥臂导通的中心开始 ADC 采样，采集相电流。注入组 ADC 转换完后，触发 ADC 中断，在 ADC 中断中根据当前采样的相电流执行 FOC 算法。

FOC 算法中需用 ADC 采样三相电流，因此在 MotorApp 配置为 FOC 模式，由按键启动电机运转执行 FOC 算法时，需先对 ADC 模块进行初始化配置。ADC 模块初始化主要包括使能 ADC 模块，配置 ADC 中断回调函数，根据硬件电路设置 ADC 注入组通道，采样延迟设置，配置 ADC 中断源及中断优先级，使能 ADC 中断等。

- 单电机应用 FOC 控制中对 ADC 的初始化接口 API 为 void FOC_ADC_Initialize(void)；
- 双电机应用 FOC 控制中对 ADC 的初始化接口 API 为 void FOC_ADC_Initialize_M1(void)和 void FOC_ADC_Initialize_M2(void)。

当 MotorApp 配置为带霍尔 BLDC 模式，执行 BLDC 方波控制时，BLDC 必须进行 ADC 模块进行初始化配置，用于母线电流的采集。ADC 模块初始化主要包括 ADC 模块使能，设置母线电流采样 ADC 通道，配置 ADC 中断回调函数，配置 ADC 中断源及中断优先级等操作。双电机应用 BLDC 控制比单电机应用 BLDC 控制多采样了一个母线电流通道，用于支持两台电机的 BLDC 控制电流环运算；BLDC 控制的共同 ADC 初始化 API 接口为 void BLDC_ADC_Init (void) 函数。

2.3.4 PWM 触发 ADC 功能初始化

AC7840x 的每个 ADC 模块都可以配置独立的规则组触发源和注入组触发源。相较于 AC780x 系列的主要触发源 PWM_INIT 和 PWM_MATCH，AC7840x 系列增加了 PWM_MAX 触发源。因此，AC7840x 系列 PWM 除法 ADC 转换更加灵活，用户可根据实际应用情况在 CTU 中进行有效的触发路径选择配置。

以 FOC 运算双电阻采样算法为例，由于 PWM 的产生方式为边沿对齐，而相电流采样时机在下半桥导通的中心位置，因此需要采用 PWM_INIT 来触发 ADC 采样转换，详细采样时序可参考图 2-4 所示。

单电机应用 FOC 控制 CTU 配置的 API 接口为 void FOC_CUT_Initialize (void) 函数，配置如下：

```
CTU_DRV_DeInit(0U);                                    ///< CTU模块清除初始化
CTU_DRV_Init(0U);                                      ///< CTU模块初始化
/*配置PWM2模块的PWM_INIT作为触发源,触发ADC1的注入组转换*/
TRGMUX_DRV_SetTrigSourceForTargetModule(0U, TRGMUX_TRIG_SOURCE_PWM2_INIT_TRIG, TIRGMUX_TARGET_MODULE_ADC1_INJECTION0);
```

双电机应用 FOC 控制 CTU 配置的 API 接口为 void FOC_CUT_Initialize (void) 函数，配置如下：

```
CTU_DRV_DeInit(0U);                                    ///< CTU模块清除初始化
CTU_DRV_Init(0U);                                      ///< CTU模块初始化
/*配置PWM2模块的PWM_INIT作为触发源，触发ADC1的注入组转换；*/
TRGMUX_DRV_SetTrigSourceForTargetModule(0U, TRGMUX_TRIG_SOURCE_PWM2_INIT_TRIG, TIRGMUX_TARGET_MODULE_ADC1_INJECTION0);
/*配置PWM3模块的PWM_INIT作为触发源，触发ADC0的注入组转换；*/
TRGMUX_DRV_SetTrigSourceForTargetModule(0U, TRGMUX_TRIG_SOURCE_PWM3_INIT_TRIG, TIRGMUX_TARGET_MODULE_ADC0_INJECTION1);
```

2.3.5 TIM 初始化

AC7840x 有 1 个四通道的 32-bit 定时器模块，用户可以根据实际的定时需求来选择合适的 Timer。

Motor_App 软件工程中采用 Timer0 定时作为 BLDC 方波控制模式的时基，初始化接口 API 为 void BLDC_TIM_Initialize(void)，中断回调接口为 void BLDC_Base_TimerCallBack (void *device, uint32_t wpara, uint32_t lpara)；在无感 BLDC 方波控制中， Timer1 作为电机 1 换相定时器，初始化接口 API 为 void BLDC_Bemf_TimerInit (void)；Timer2 作为电机 2 换相定时器，初始化接口 API 为 void BLDC_Bemf_TimerInit_M2 (void)。

Motor_App 采用 Timer0 定时作为 FOC 控制模式的毫秒任务时基，中断回调接口为 void Timer0_Callback (void *device, uint32_t wpara, uint32_t lpara)。Timer1 初始化为电机 1 正交编码器 M_T 测速法运算定时器，Timer2 初始化为电机 2 正交编码器 M_T 测速法运算定时器，FOC 控制模式 Timer 初始化接口 API 为 void FOC_TIM_Initialize(void)。

3 中断服务程序

Demo 板的适配 MotorApp 软件工程根据电机类型分为 BLDC 方波控制与 PMSM FOC 控制，下面介绍各控制类型中的中断服务程序。

3.1 FOC 控制中断服务程序

3.1.1 电机过电流保护中断

发生硬件过电流时触发，由外部比较器电路产生过流故障信号，芯片检测到有效的故障电平信号后，触发 PWM Fault 中断。关闭 PWM 脉冲宽度调制波输出，保护电路板。该中断优先级一般设置为 0，即最高优先级。

在 MotorApp 工程中，FOC 模式中加入了电机过流保护机制，其执行 API 为 void FOC_PWM_FaultCallback(void *device, uint32_t wpara, uint32_t lpara)，发生过流触发中断时，将电机运行状态设置为 FAULT，给出过流标志位并禁止 PWM 脉冲宽度调制波输出。

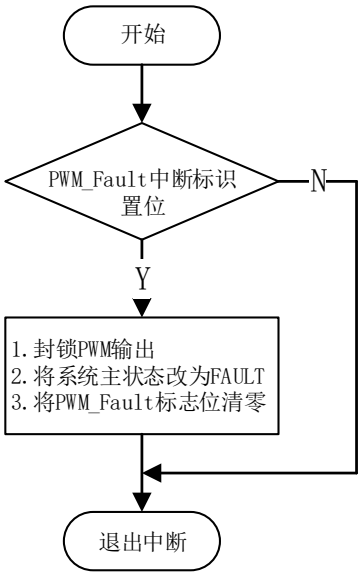

图 3-1 硬件过流中断服务程序

3.1.2 FOC 计算 ADC 采样完成中断

在 ADC 初始化函数中，已设置了相应的 ADC 中断回调函数，因此在相应 ADC 通道完成 ADC 转换触发 ADC 中断后，即调用相应的回调函数。

FOC 模式执行时 ADC 中断调用 void Motor_Ctrl_CallBack(void *device, uint32_t wpara, uint32_t lpara)函数，因为 ADC 中断后即可完成采集三相电流。故在回调函数 Motor_Ctrl_CallBack 中，调用 Get_CurrentSample ， AdcResultTransmit ， Foc_Execution_AngleProcess 和 Foc_Execution_CurrentLoopProcess 进行 FOC 运算。

ADC 中断服务程序如图 3-2 所示。

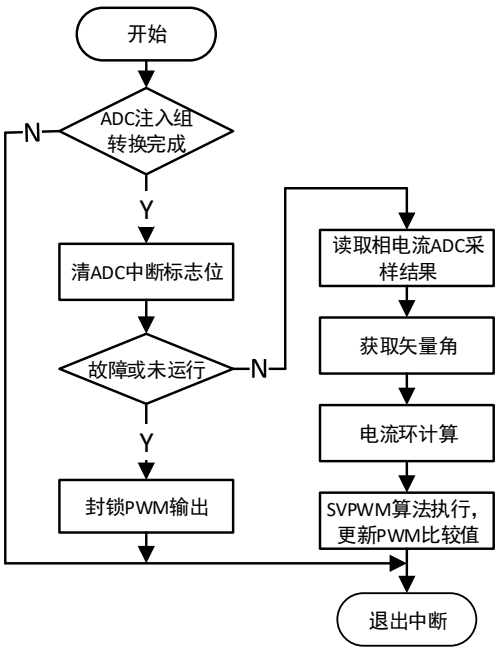

图 3-2 FOC 运算 ADC 中断服务程序

3.1.3 Timer 中断

FOC 模式执行时 Timer0 中断调用 void Timer0_Callback(void *device, uint32_t wpara, uint32_t lpara)函数，在回调函数中执行毫秒时基任务调度管理；Timer1 中断调用 void Timer1_Callback(void *device, uint32_t wpara, uint32_t lpara)函数，在回调函数中执行 M_T 测速法运算。

Timer 中断流程图如图 3-3 所示。

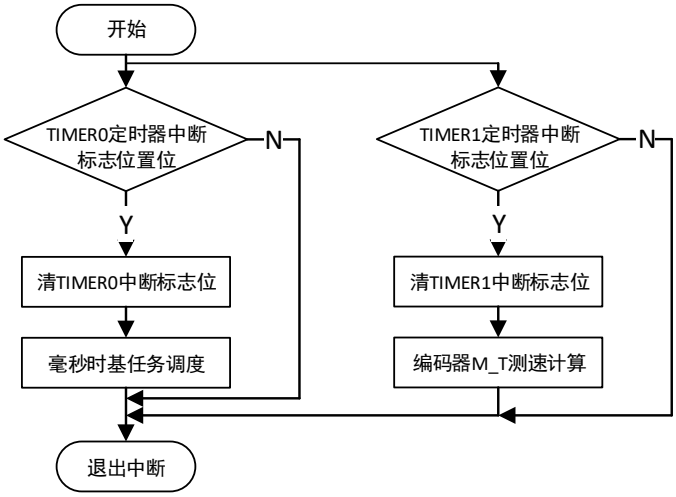

图 3-3 定时器中断服务程序

3.1.4 Hall 中断

在 Hall 传感器 FOC 模式执行会触发 Hall 中断，并在中断服务中通过中断回调函数执行 Hall 传感器自学习与 Hall 角度计算。AC780x 和 AC781x 系列中，Hall 中断通过 PWDT 外设检测，PWDT 中断回调 API 为 void FOC_PWDT_Callback(void *device, uint32_t wpara, uint32_t lpara)函数；AC7840x 系列中，Hall 中断通过 PWM 外设双边沿捕获模式捕获 Hall 输入信号，PWM 溢出中断回调 API 为 void FOC_Hall_OverCallback(uint8_t instance, uint32_t status, viod *userData)函数，PWM 捕获通道事件中断回调 API 为 void FOC_Hall_Callback(uint8_t instance, uint32_t status, viod *userData)函数。

Hall PWDT 中断流程图如图 3-4 所示，Hall PWM 中断流程图如图 3-5 所示。

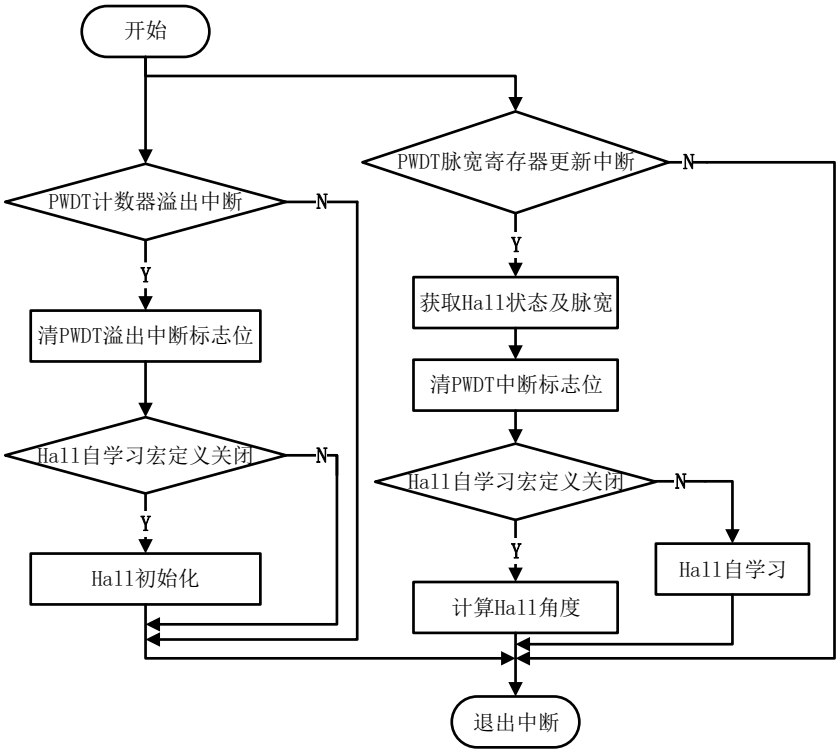

图 3-4 FOC Hall PWDT 中断服务程序

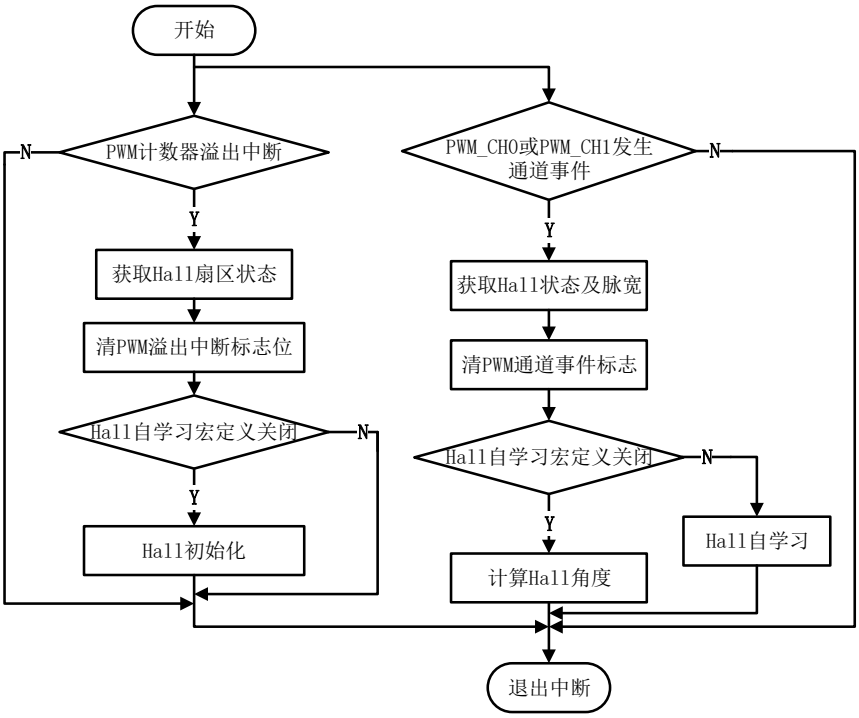

图 3-5 FOC Hall PWM 中断服务程序

3.1.5 Encoder 中断

在 Encoder FOC 模式执行时 PWM 溢出中断调用 void FOC_Encoder_Callback(void *device, uint32_t wpara, uint32_t lpara)函数，在回调函数中执行编码器计算电机旋转圈数功能。

Encoder 中断流程图如图 3-6 所示。

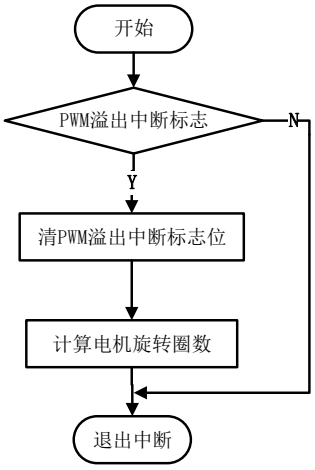

图 3-6 编码器 PWM 中断服务程序

3.1.6 PulseInject 脉冲注入 ADC 中断

FOC 模式使用脉冲注入预定位功能时 ADC 中断调用 void InitPos_AdcSubCallback(void *device, uint32_t wpara, uint32_t lpara)函数，在回调函数中执行脉冲注入预定位功能。

PulseInject 中断流程图如图 3-7 所示。

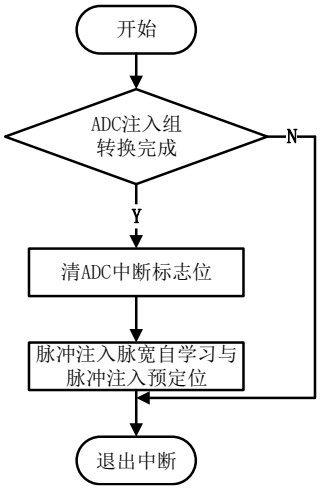

图 3-7 脉冲注入 ADC 中断服务程序

3.1.7 ParamIdentify 参数辨识 ADC 中断

FOC 模式执行电机参数辨识时 ADC 中断调用 void ParamIdenfity_AdcSubCallBack(void *device, uint32_t wpara, uint32_t lpara)函数，在回调函数中执行电机参数辨识子功能。

ParamIdentify 中断流程图如图 3-8 所示。

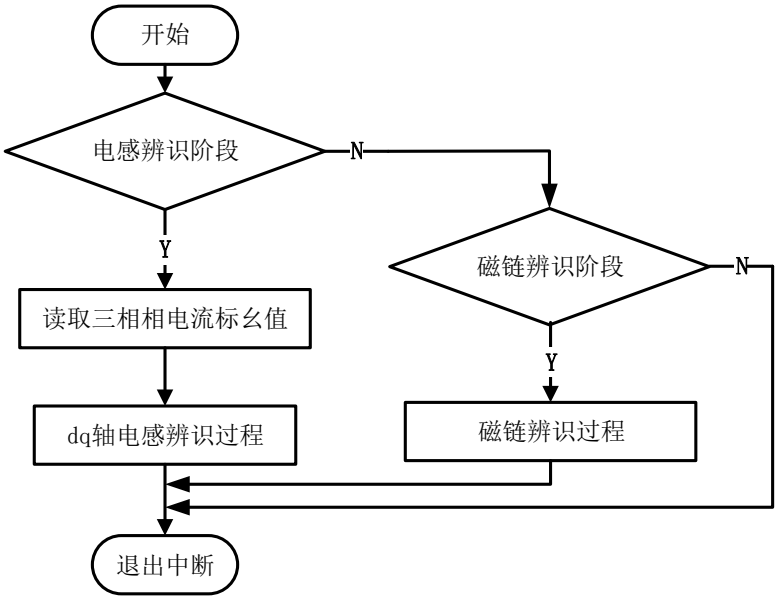

图 3-8 参数辨识 ADC 中断服务程序

3.2 方波控制中断服务程序

3.2.1 电机过电流保护中断

发生硬件过电流时触发，由外部比较器电路产生过流故障信号，芯片检测到有效的故障电平信号后，触发 PWM Fault 中断。关闭 PWM 脉冲宽度调制波输出，保护电路板。该中断优先级一般设置为 0，即最高优先级。

在 MotorApp 工程中，BLDC 模式中加入了电机硬件过流保护机制，其执行 API 为 void BLDC_PWM_FaultCallback(void *device, uint32_t wpara, uint32_t lpara)，发生过流触发中断时，将电机运行状态设置为 FAULT，给出过流标志位并将 PWM 脉冲宽度调制波输出设置为无效。

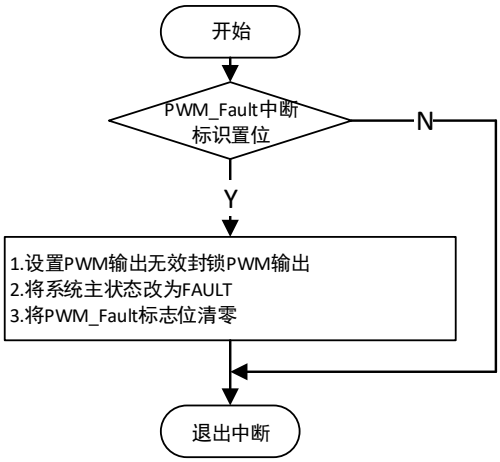

图 3-9 BLDC 控制硬件过流中断服务程序

3.2.2 ADC 中断

在 ADC 初始化函数中，已为 BLDC 模式设置了相应的 ADC 中断回调函数，在 ADC 注入组受 PWM_MATCH 触发 ADC 转换完成后，中断标志位置位并进入相应的中断回调函数。

BLDC 模式执行时 ADC 中断调用 void BLDC_ADC_Callback(void *device, uint32_t wpara, uint32_t lpara)函数，在回调函数中读取母线电流 AD 值，用于电流环闭环运算。

ADC 中断服务程序如图 3-10 所示。

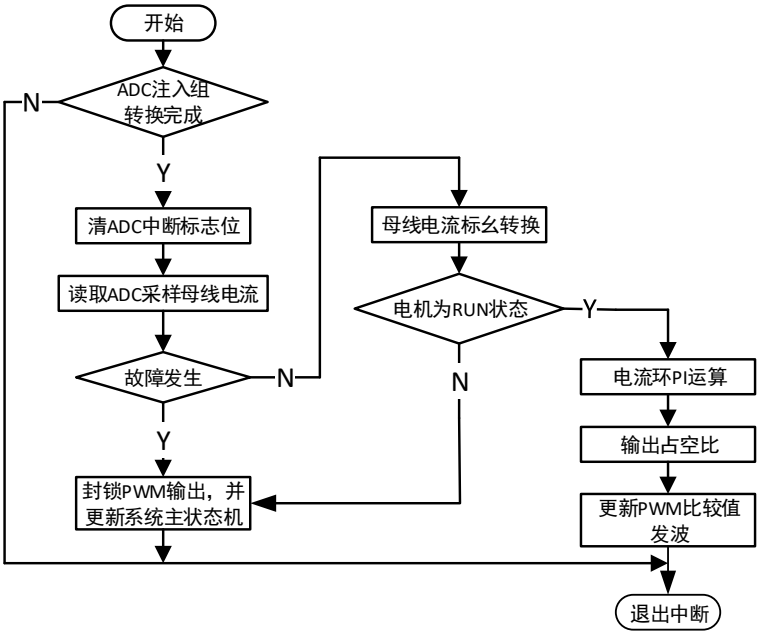

图 3-10 BLDC 控制 ADC 中断服务程序

3.2.3 Timer 中断

在 Timer 初始化函数中，已为 BLDC 模式设置了相应的 Timer 中断回调函数，因此在相应 Timer 定时到出发中断后，即调用相应的回调函数。

BLDC 采用 Timer0 定时作为 BLDC 方波控制模式的时基。

BLDC 无感模式执行时 Timer1 作为换相定时器，初始化接口 API 为 void BLDC_Bemf_TimerInit (void)函数，在该中断中进行无感方波控制运行阶段的换相控制。

Timer0 中断流程图如图 3-11 所示，Timer1 中断流程图如图 3-12 所示。

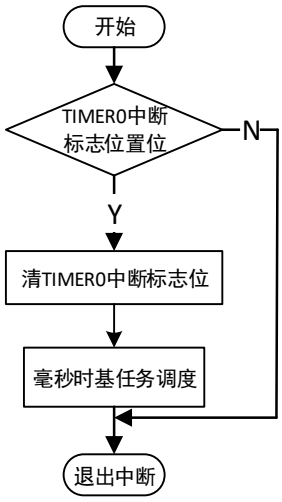

图 3-11 BLDC 控制定时器 0 中断服务程序

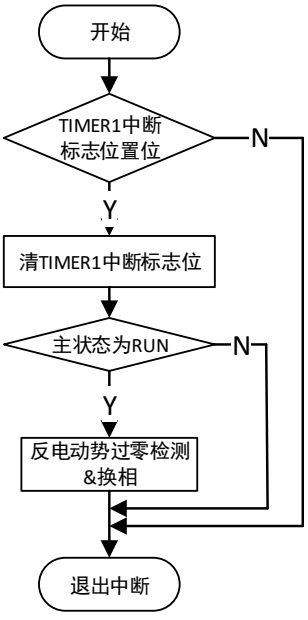

图 3-12 BLDC 控制定时器 1 中断服务程序

3.2.4 Hall 中断

在 Hall 传感器 BLDC 模式执行会触发 Hall 中断，并在中断服务中通过中断回调函数执行 Hall 传感器扇区读取和查表换相任务。AC780x 和 AC781x 系列中，Hall 中断通过 PWDT 外设检测，PWDT 中断回调 API 为 void BLDC_Hall_ PWDTCallback (void *device, uint32_t wpara, uint32_t lpara)函数；AC7840x 系列中，Hall 中断通过 PWM 外设双边沿捕获模式捕获 Hall 输入信号，PWM 溢出中断回调 API 为 void BLDC_Hall_OFCallback(uint8_t instance, uint32_t status, viod *userData)函数，PWM 捕获通道事件中断回调 API 为 void BLDC_Hall_ChannelCallback(uint8_t instance, uint32_t status, viod *userData)函数。

Hall PWDT 中断流程图如图 3-13 所示，Hall PWM 中断流程图如图 3-14 所示

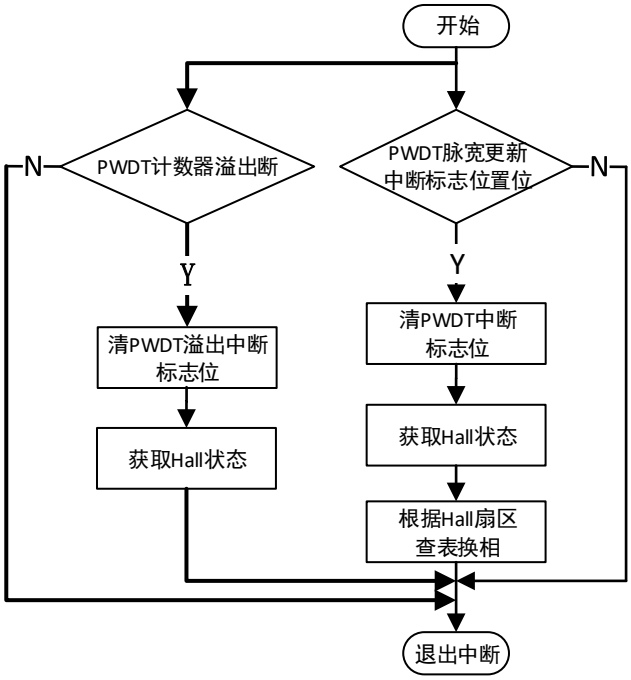

图 3-13 BLDC Hall PWDT 中断服务程序

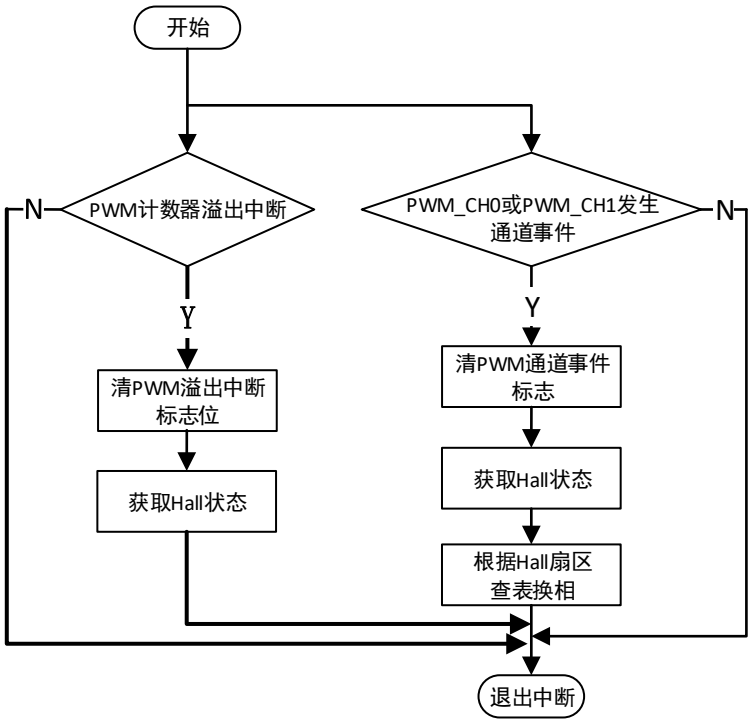

图 3-14 BLDC Hall PWM 中断服务程序

4 电机控制算法

4.1 BLDC 方波控制

BLDC 的控制电路对电机转子位置信号进行逻辑变换后产生脉宽调制 PWM 信号，驱动逆变器的功率开关管，从而控制 BLDC 电机各相绕组按一定顺序工作，在电机气隙中产生跳跃式旋转磁场。BLDC 转子旋转时，电角度每转过 60°，逆变器开关管换流一次、定子磁场状态改变一次，因此 BLDC 共有 6 个磁场状态，三相各导通 120°，相电流为方波。BLDC 的方波控制框图如图 4-1 所示。

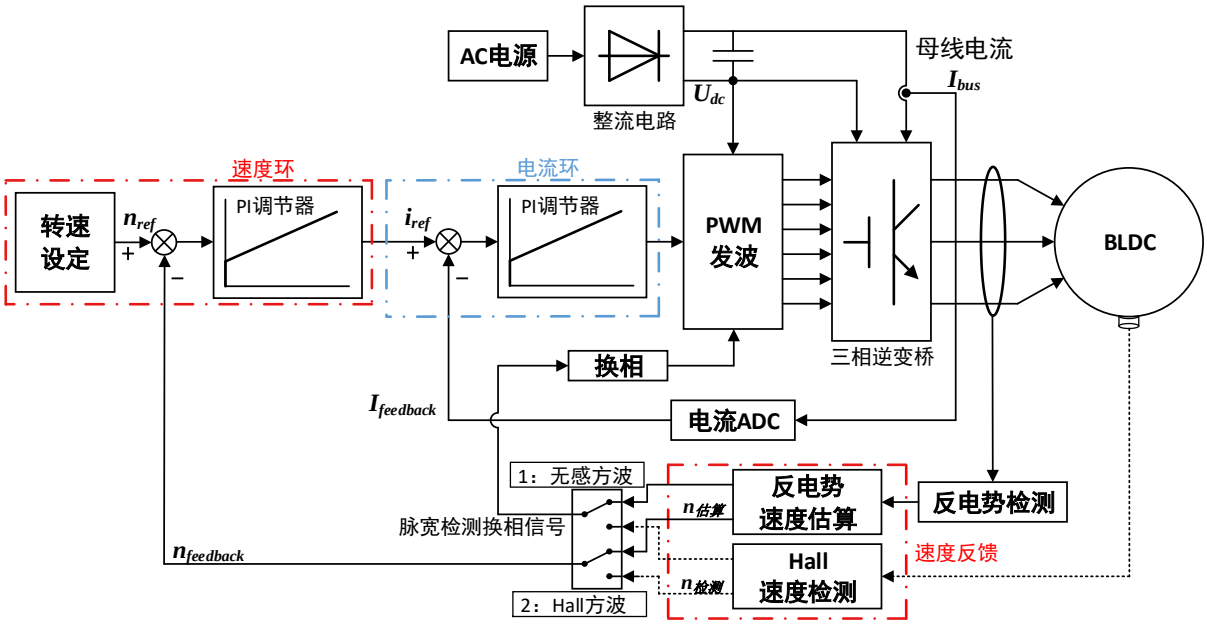

图 4-1 BLDC 方波控制框图

BLDC 电机控制支持带 Hall 传感器的 Hall 方波控制与基于反电势检测的无感方波控制。

表 4-1 BLDC 方波控制参数对应表

编号	名称	框图变量名	程序变量名	备注
1	速度参考值	n_{ref}	g_bldc_speedCmd.speedTargetRamp	来源于速度斜坡
2	速度反馈值	$n_{feedback}$	g_bldc_speedCommand.speedFbk	根据控制方式有不同来源
3	电流给定值	i_{ref}	g_bldcVarsCtrl.ibusRefPu	来自速度环输出
4	电流反馈值	$I_{feedback}$	g_bldcVarsCtrl.ibusFdkPu	来自母线电流采样结果

4.2 PMSM 矢量控制

PMSM 的矢量控制也称为磁场定向控制（Field Oriented Control，FOC）。在 FOC 中，电机的定子电流被分解为用于产生磁场的直轴电流（励磁电流）与用于控制转矩的交轴电流（转矩电流），通过对转速和电流的双环控制实现电机的高性能运行。根据转速信号来源可以分成带传感器 FOC 以及无传感器 FOC，其中速度传感器通常有 Hall 传感器、编码器等类型。PMSM 的 FOC 控制框图如图 4-2 所示。

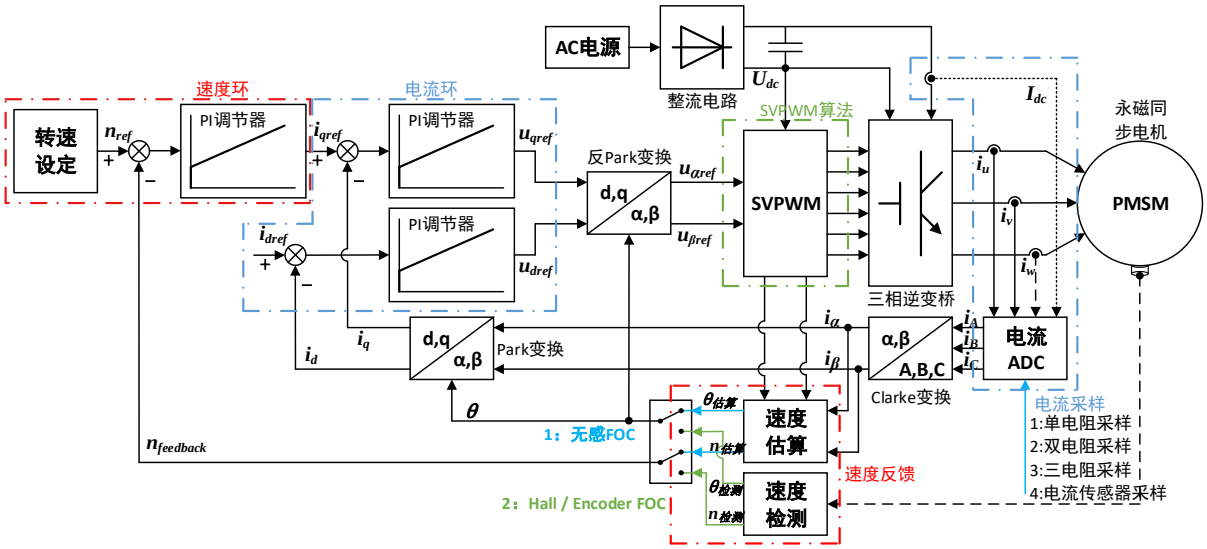

图 4-2 PMSM 矢量控制框图

PMSM 矢量控制算法中，除角度变量为 Q16（0~65535）格式外，其余算法中控制参数均为 Q15（-32768~32767）格式。

PMSM 矢量控制支持带速度/位置传感器的有感（Hall 或 Encoder）FOC，以及无传感器 FOC。电流采样支持电阻采样与电流传感器采样，其中电阻采样模式支持单电阻采样重构三相电流（*AC781x 系列不支持）。

表 4-2 PMSM 矢量控制参数对应表

编号	名称	框图变量名	程序变量名	备注
1	速度参考值	n_{ref}	g_speedCommand.speedTargetRamp	来自速度斜坡
2	速度反馈值	$n_{feedback}$	g_speedCommand.speedFbk	根据控制方式有不同来源
3	矢量角	θ	g_focVarsCtrl.elecAngle	根据控制方式有不同来源

编号	名称	框图变量名	程序变量名	备注
4	三相电流	iA， iB，iC	g_adcTransform.iaPu g_adcTransform.ibPu g_adcTransform.icPu	根据电流采样结果进行标幺化处理
5	d 轴电流参考值	i_{dref}	g_focVarsCtrl.idRef	根据控制方式有不同来源
6	d 轴电流反馈值	i_d	g_focVarsCtrl.idFdb	来自电流 park 变换
7	q 轴电流参考值	i_{qref}	g_focVarsCtrl.iqRef	来自速度环的输出
8	q 轴电流反馈值	i_q	g_focVarsCtrl.iqFdb	来自电流 park 变换

4.3 异步电机矢量控制

异步电机的矢量控制框图与图 4-2 相同，区别仅在于 d 轴电流给定方式不一致，以及异步电机的转子转速不能反映定子电流的频率，其矢量角的计算方式与 PMSM 不同。

5 电机运行保护功能

5.1 故障 ID 标志

Motor_App 软件工程中包含 32 个保护故障状态，ID 对应 0-31，在 protector.h 中的定义参见表 5-1。

表 5-1 保护功能故障 ID 对应表

故障 ID 号	故障宏定义	故障名称
0	OVER_CURRENT_PHASE_FAILURE	相电流过流故障
1	OVER_CURRENT_IBUS_FAILURE	母线电流过流故障
2	OVER_CURRENT_IS_FAILURE	矢量电流过流故障
3	VBUS_OVER_VOLTAGE_FAILURE	过电压故障
4	VBUS_UNDER_VOLTAGE_FAILURE	欠电压故障
5	LOSE_SPEED_FAILURE	失速故障
6	ZERO_SPEED_FAILURE	零速故障
7	STARTUP_FAILURE	启动故障
8	MOTOR_STALL_FAILURE	电机堵转故障
9	PHASEA_CURRENT_MIDPOINT_FAILURE	A 相电流中点故障
10	PHASEB_CURRENT_MIDPOINT_FAILURE	B 相电流中点故障
11	PHASEC_CURRENT_MIDPOINT_FAILURE	C 相电流中点故障
12	LOSE_PHASEA_FAILURE	A 相缺相故障
13	LOSE_PHASEB_FAILURE	B 相缺相故障
14	LOSE_PHASEC_FAILURE	C 相缺相故障
15	HALL_STATUS_FAILURE	霍尔传感器状态故障
16	HALL_STATUS_CHANGE_FAILURE	霍尔传感器状态转换故障
17	PHASE_CURRENT_SHORT_FAILURE	相电流短路故障
18	MOSFET_LOWERBRIDGEA_SHORT_FAILURE	A 相下桥臂短路故障

故障 ID 号	故障宏定义	故障名称
19	MOSFET_LOWERBRIDGEB_SHORT_FAILURE	B 相下桥臂短路故障
20	MOSFET_LOWERBRIDGEC_SHORT_FAILURE	C 相下桥臂短路故障
21	MOSFET_UPPERBRIDGEA_SHORT_FAILURE	A 相上桥臂短路故障
22	MOSFET_UPPERBRIDGEB_SHORT_FAILURE	B 相上桥臂短路故障
23	MOSFET_UPPERBRIDGEC_SHORT_FAILURE	C 相上桥臂短路故障
24	MOSFET_LOWERBRIDGEA_OPEN_FAILURE	A 相下桥臂开路故障
25	MOSFET_LOWERBRIDGEB_OPEN_FAILURE	B 相下桥臂开路故障
26	MOSFET_LOWERBRIDGEC_OPEN_FAILURE	C 相下桥臂开路故障
27	MOSFET_UPPERBRIDGEA_OPEN_FAILURE	A 相上桥臂开路故障
28	MOSFET_UPPERBRIDGEB_OPEN_FAILURE	B 相上桥臂开路故障
29	MOSFET_UPPERBRIDGEC_OPEN_FAILURE	C 相上桥臂开路故障
30	HALL_SELF_LEARN_FAILURE	霍尔传感器自学习故障
31	HARDWARE_OC_FAILURE	硬件过流故障

5.2 电机保护功能 API

5.1 节描述的电机的保护功能中，判断上下桥臂短路、通路，需与硬件电路相匹配，此处不作详细介绍，仅对常规使用的电机的保护功能接口进行描述，如表 5-2 所示。

表 5-2 故障保护功能 API

故障保护 API 接口	功能
Fault_Report	故障置位
Fault_Clear	故障清除
Phase_Over_Current_Check	相电流过流保护
Bus_Over_Current_Check	母线电流过流保护

故障保护 API 接口	功能
Vbus_OverVoltage_Check	过压保护
Vbus_UnderVoltage_Check	欠压保护
LoseSpeed_Check	失速保护
ZeroSpeed_Check	零速保护
Stall_Check_Sensor	堵转保护（有位置传感器）
Stall_Check_Sensorless	堵转保护（无位置传感器）
Current_Sensor_Check	相电流中点检测保护
LosePhase_Check	缺相保护（动态）
Phase_Short_Dynamic_Check	相线短路保护（动态）
Motor_Protection	电机保护主函数

5.3 电机保护功能说明

5.3.1 相电流过流保护

电机相电流超过设定阈值，且连续达到设定次数（防误报）时，上报相电流过流故障（OVER_CURRENT_PHASE_FAILURE）。

通过以下进行相电流保护设置：

```
#define PHASE_CURRENT_PROTECTION_ENABLE          /*!< Phase current overcurrent protection enable */
#define MAX_IPHASE_THRESHOLD      Math_IQ(0.95)  /*!< Protection threshold, refer to current base value */
#define IPHASE_DBC                (10)           /*!< Phase current overcurrent protection operation period value in ms unit, 10 refer to 10ms */
```

- 宏定义 PHASE_CURRENT_PROTECTION_ENABLE，定义时打开相电流保护功能，屏蔽则关闭相电流保护功能。
- 宏定义 MAX_IPHASE_THRESHOLD 设置保护电流阈值，保护电流阈值对应真实值 = 0.95* MAX_CURRENT(在 motor_parameters_define.h 中定义)。

- 宏定义 IPHASE_DBC 设置保护电流判断次数，连续判断过流超过此次数上报过流故障。
- 验证相电流过流保护时可先将宏定义 MAX_IPHASE_THRESHOLD 的值设置为较小值，提前模拟相电流过流保护的发生。

5.3.2 母线电流过流保护

电机母线电流超过设定阈值，且连续达到设定次数（防误报）时，上报母线电流过流故障（OVER_CURRENT_IBUS_FAILURE）。

通过以下进行母线电流保护设置：

```
#define BUS_OVERCURRENT_PROTECTION_ENABLE                /*!< Bus current overcurrent protection enable */
#define BUS_OVERCURRENT_THRESHOLD          (20)          /*!< Bus current overcurrent protection threshold,
10 timers larger than true value, 20 means 2A*/
#define BUS_OVERCURRENT_DBC                (10)          /*!< Bus current overcurrent protection operation
period value in ms unit, 10 refer to 10ms */
```

- BUS_OVERCURRENT_PROTECTION_ENABLE，定义时打开母线电流保护功能，屏蔽则关闭母线电流保护功能。
- 宏定义 BUS_OVERCURRENT_THRESHOLD 设置保护电流阈值，保护电流阈值对应真实值 = BUS_OVERCURRENT_THRESHOLD /10。
- 宏定义 BUS_OVERCURRENT_DBC 设置保护电流判断次数，连续判断过流超过此次数上报母线过流故障。
- 母线电流若采用估计值，需在 drive_parameters_define.h 中打 POWER_DCCUR_EST_ENABLE 宏定义，使能母线电流估算功能。(注意估算的母线电流准确性取决于当前电机三相电流准确度)
- 验证母线过流保护功能时，可将保护阈值 BUS_OVERCURRENT_ THRESHOLD 减小来提前模拟保护的发生。

5.3.3 过压保护

母线电压超过设定阈值，且连续达到 10 次时上报过电压故障（VBUS_OVER_VOLTAGE_FAILURE），低于设定恢复值且连续达到 10 次时清除过电压故障（VBUS_OVER_VOLTAGE_FAILURE）并恢复。

通过以下进行过电压保护设置：

```
#define VBUS_OVERVOLTAGE_PROTECTION_ENABLE                    /*!< Bus voltage overvoltage fault protection enable */
#define OVERVOLTAGE_RECOVERY_VOLTAGE    (MAX_BUS_VOLTAGE – 2)    /*!<Bus voltage overvoltage fault recovery voltage, 2 means to recover from the fault when the bus voltage is 2V lower than the overvoltage point */
#define VBUS_OVERVOLTAGE_THRESHOLD        (MAX_BUS_VOLTAGE * 10)  /*!< Real vbus value enlarges 10 times, action threshold of overvoltage protection */
#define VBUS_OVERVOLTAGE_RECOVERY_THRESHOLD    (OVERVOLTAGE_RECOVERY_VOLTAGE * 10)  /*!< Real value enlarges 10 times, action threshold of overvoltage recovery */
```

- VBUS_OVERVOLTAGE_PROTECTION_ENABLE，定义时打开过电压保护功能，屏蔽则关闭过电压保护功能。
- 宏定义 VBUS_OVERVOLTAGE_THRESHOLD 设置过电压保护阈值，过电压保护阈值对应真实值 = VBUS_OVERVOLTAGE_THRESHOLD / 10。
- 宏定义 OVERVOLTAGE_RECOVERY_VOLTAGE 设置过电压保护恢复阈值，过电压保护恢复阈值对应真实值 = OVERVOLTAGE_RECOVERY_VOLTAGE / 10。
- 验证母线过压保护时，可先将外部电压源电压设置高于 VBUS_OVERVOLTAGE_THRESHOLD，然后 START 电机，可报母线过压故障；将电压源电压设置低于 OVERVOLTAGE_RECOVERY_VOLTAGE，故障标志可清零。

5.3.4 欠压保护

母线电压低于设定阈值，且连续达到 10 次时上报欠电压故障（VBUS_UNDER_VOLTAGE_FAILURE），高于设定恢复值且连续达到 10 次时清除欠电压故障（VBUS_UNDER_VOLTAGE_FAILURE）并恢复。

通过以下进行过电压保护设置：

```
#define VBUS_UNDERVOLTAGE_PROTECTION_ENABLE                    /*!< Bus voltage undervoltage fault protection enable */
#define UNDERVOLTAGE_RECOVERY_VOLTAGE    (MIN_BUS_VOLTAGE + 2)    /*!<Bus voltage undervoltage fault recovery voltage, 2 means to recover from the fault when the bus voltage is 2V higher than the undervoltage point */
#define VBUS_UNDERVOLTAGE_THRESHOLD        (MIN_BUS_VOLTAGE * 10)  /*!< Real vbus value enlarges 10 times, action threshold of undervoltage protection */
#define VBUS_UNDERVOLTAGE_RECOVERY_THRESHOLD    (UNDERVOLTAGE_RECOVERY_VOLTAGE * 10)  /*!< Real value enlarges 10 times, action threshold of undervoltage recovery */
```

- VBUS_UNDERVOLTAGE_PROTECTION_ENABLE，定义时打开欠电压保护功能，屏蔽则关闭欠电压保护功能。
- 宏定义 VBUS_UNDERVOLTAGE_THRESHOLD 设置欠电压保护阈值，欠电压保护阈值对应真实值 = VBUS_UNDERVOLTAGE_THRESHOLD / 10。

- 宏定义 UNDERVOLTAGE_RECOVERY_VOLTAGE 设置欠电压保护恢复阈值，欠电压保护恢复阈值对应真实值 = UNDERVOLTAGE_RECOVERY_VOLTAGE / 10。
- 验证母线欠压保护时，可将外部电压源电压设置低于 VBUS_UNDERVOLTAGE_THRESHOLD，然后 START 电机，可报母线欠压故障；将电压源电压设置高于 UNDERVOLTAGE_RECOVERY_VOLTAGE，故障标志可清零。

5.3.5 失速保护

转速超过设定阈值或当前转速与上一次转速差值大于设定阈值，且连续达到 10 次时上报失速故障（LOSE_SPEED_FAILURE）。

```
#define LOSE_SPEED_PROTECTION_ENABLE                /*!< Lose speed protection enable */
#define LOSE_SPEED_THRESHOLD        (Math_IQ(0.99)) /*!< Frequency threshold of lose speed protection */
#define LOSE_SPEED_ERR_THRESHOLD    (Math_IQ(0.1))  /*!< Frequency error threshold of lose speed protection */
#define LOSE_SPEED_DBC              (2)             /*!< Lose speed protection operation period value in ms unit, 2 refer to 2ms */
```

- LOSE_SPEED_PROTECTION_ENABLE，定义时打开失速保护功能，屏蔽则关闭失速保护功能。
- 宏定义 LOSE_SPEED_THRESHOLD 设置失速保护阈值，失速保护阈值电气频率对应真实值 =（0.99 * BASE_FREQ）。
- 宏定义 LOSE_SPEED_ERR_THRESHOLD 设置失速保护速度差值阈值，差值大小对应转速加速度，与失速保护接口调用周期相关。
- 宏定义 LOSE_SPEED_DBC 设置失速保护判断次数，连续判断超过此次数上报失速故障。
- 验证失速保护时可将 LOSE_SPEED_THRESHOLD 设置为较小值，提前模拟失速保护的发生。

5.3.6 零速保护

转速低于设定阈值，且连续达到 2 次时上报零速故障（ZERO_SPEED_FAILURE）。

```
#define ZERO_SPEED_PROTECTION_ENABLE                /*!< Zero speed protection enable */
#define ZERO_SPEED_THRESHOLD        Math_IQ(0.07)   /*!< Zero speed protection frequency threshold */
#define ZERO_SPEED_DBC              (2)             /*!< Zero speed protection operation period value in ms unit, 2 refer to 2ms */
```

- ZERO_SPEED_PROTECTION_ENABLE，定义时打开零速保护功能，屏蔽则关闭零速保护功能。

- 宏定义 ZERO_SPEED_THRESHOLD 设置零速保护阈值，零速保护阈值电气频率对应真实值 =（0.07 * BASE_FREQ）。
- 宏定义 ZERO_SPEED_DBC 设置零速保护判断次数，连续判断超过此次数上报失速故障。

5.3.7 堵转保护（有位置传感器）

转速低于设定阈值且矢量电流大小超过设定保护阈值，连续达到设定时间，上报堵转保护故障（MOTOR_STALL_FAILURE）。

```
#define STALL_PROTECTION_SENSOR_ENABLE          /*!< Hall sensor control stall protection enable */
#define STALL_SPD_THRESHOLD          (20)       /*!< Stall judge speed, true */
#define STALL_CUR_THRESHOLD          (Math_IQ(0.3)) /*!< Stall judge current, pu, refer to current base value*/
#define STALL_DBC          (500)     /*!< Hall sensor stall protection operation period value, 500 refer to 500ms */
```

- STALL_PROTECT_SENSOR_ENABLE，定义时打开（有位置传感器）堵转保护功能，屏蔽则关闭堵转保护功能。
- 宏定义 STALL_SPD_THRESHOLD 设置堵转保护转速阈值，堵转保护转速阈值对应 RPM 为单位的转速值。
- STALL_CUR_THRESHOLD 设置堵转保护电流阈值，保护电流阈值对应真实值 = 0.3* MAX_CURRENT(在 motor_parameters_define.h 中定义)。
- 宏定义 STALL_DBC 设置堵转保护判断时间，以 ms 为单位（取决于堵转保护调用周期）。

5.3.8 堵转保护（无位置传感器）

可通过反电动势幅值与当前估算转速的比例超过设定阈值判断无传感模式下发生堵转情况：

```
#define STALL_PROTECTION_SENSORLESS_ENABLE          /*!< Sensorless control stall protection enable */
#define BEMF_CONSISTENT_THRESHOLD          (5)      /*!< Maxiumum accepted back-EMF on speed estimates (percentage) */
#define BEMF_DBC                           (500)    /*!< Sensorless stall protection operation period value on back-EMF method in ms unit, 500 refer to 500ms*/
```

- STALL_PROTECT_SENSORLESS_ENABLE，定义时打开（无位置传感器）堵转保护功能，屏蔽则关闭堵转保护功能。
- 宏定义 BEMF_CONSISTENT_THRESHOLD 设置反电动势幅值方式保护判断阈值，需实际调试确认，从 1 开始往上设置。
- 宏定义 BEMF_DBC 设置堵转保护反电动势判断时间，以 ms 为单位（取决于堵转保护调用周期）。

5.3.9 相电流中点检测保护

上电初始化阶段调用，获取三相电流中点偏移大小，超过 5%中点偏移电压，根据三相中点电压情况，上报相电流中点故障，分别为 PHASEA_CURRENT_MIDPOINT_FAILURE 、PHASEB_CURRENT_MIDPOINT_FAILURE、PHASEC_CURRENT_MIDPOINT_FAILURE。

5.3.10 缺相保护（动态）

电机运行状态下，当相电流小于设定缺相判断阈值，连续达到设定次数时，根据三相中点电流情况，上报缺相故障，分别为 LOSE_PHASEA_FAILURE 、 LOSE_PHASEB_FAILURE 、LOSE_PHASEC_FAILURE。

```
#define LOSE_PHASE_PROTECTION_ENABLE                /*!< Lose phase protection enable */
#define LOSE_PHASE_THRESHOLD    (Math_IQ(0.055))    /*!< Lose phase protection operation threshold value, refer to current base value */
#define LOSE_PHASE_DBC          (20)      /*!< Lose phase protection operation period value, 20 refer to 20ms */
#define LOSE_PHASE_ADD_CNT  (6)       /*!< Lose phase protection operation addition counter, 6 means 2^6 = 64 */
#define LOSE_PHASE_JUDGE_THRESHOLD    (Math_IQ(3.0))    /*!< Lose phase protection operation (Imax / Imin) ratio judge threshold value */
```

- LOSE_PHASE_PROTECTION_ENABLE，定义时打开动态缺相保护功能，屏蔽则关闭缺相保护功能。
- 宏定义 LOSE_PHASE_THRESHOLD 设置缺相电流保护阈值，缺相保护电流阈值对应真实值 = 0.055* MAX_CURRENT(在 motor_parameters_define.h 中定义)。
- 宏定义 LOSE_PHASE_DBC 设置保护电流判断次数，连续判断过流超过此次数上报过流故障。
- 宏定义 LOSE_PHASE_ADD_CNT 设置保护电流判断采样点数。
- 宏定义 LOSE_PHASE_JUDGE_THRESHOLD 设置过流保护最大/最小电流比值。
- 验证缺相保护功能时，在电机运行过程中拔掉某一相电机线，可上报某相缺相故障。

5.3.11 相线短路保护（动态）

电机运行状态下，当电机转速高于判断阈值时，判断两相之间电流偏差大小，小于设定阈值且连续达到设定次数时，根据三相中点电流情况，上报相线短路故障（PHASE_CURRENT_SHORT_FAILURE）。

```
#define PHASE_SHORT_PROTECTION_ENABLE        /*!< Phase current short-circuit protection enable */
#define SPEED_THRESHOLD_SHORT        (Math_IQ(1.0 / BASE_FREQ))    /*!< Phase current short-circuit protection speed judge threshold value, refer to base frequency */
#define JUDGE_THRESHOLD_SHORT    (5)  /*!< Phase current short-circuit protection current judge threshold value */
#define IPHASE_SHORT_DBC        (10)   /*!< Phase current short-circuit protection operation period value in ms unit, 10 refer to 10ms */
```

- 宏定义 SPEED_THRESHOLD_SHORT 设置相线短路保护转速阈值，转速高于此值进行相线电流阈值判断。
- 宏定义 JUDGE_THRESHOLD_SHORT 设置相线短路电流保护阈值，两相电流差值小于此值判断为相线短路故障。
- 宏定义 IPHASE_SHORT_DBC 设置相线保护电流判断次数，连续判断大于此次数上报相线短路故障。

5.4 电机保护调用主函数

各个独立的电机功能在保护主函数 Motor_Protection()中调用，在 1ms 周期中执行。函数内部会进行分频得到 10ms loop 和 100ms loop，用于执行周期较慢的相关保护。

保护信息通过变量 g_protector.ErrFlag.all 查询，当无任何故障发生时，g_protector.ErrFlag.all 为零。当发生故障后，相应保护位会被置位。通过判断 g_protector.ErrFlag.all 对应 16 进制数置位情况判断故障发生情况。

举例如下：

g_protector.ErrFlag.all = 17（十进制） = 0x11 （十六进制）=b10001（二进制）

第 0 位和第 4 位被置位，查询故障 ID 可知发生了过流保护 OVER_CURRENT_PHASE_FAILURE 和 VBUS_UNDER_VOLTAGE_FAILURE。

6 使用说明和注意事项

6.1 电流环控制

1. 修改 drive_parameters_define.h 文件中的宏定义，为速度环和电流环 PID 参数设置默认值：

```
#define PID_TORQUE_KP_DEFAULT       (Math_IQ(0.2))
#define PID_TORQUE_KI_DEFAULT       (Math_IQ(0.004))
#define PID_TORQUE_KD_DEFAULT       (0)
#define PID_TORQUE_MAX_DEFAULT      (Q15_MAX / 2)
#define PID_TORQUE_MIN_DEFAULT      (Q15_MIN / 2)

#define PID_FLUX_KP_DEFAULT         (Math_IQ(0.2))
#define PID_FLUX_KI_DEFAULT         (Math_IQ(0.004))
#define PID_FLUX_KD_DEFAULT         (0)
#define PID_FLUX_MAX_DEFAULT        (Q15_MAX / 2)
#define PID_FLUX_MIN_DEFAULT        (Q15_MIN / 2)

#define PID_SPEED_KP_DEFAULT        (Math_IQ(0.3))
#define PID_SPEED_KI_DEFAULT        (Math_IQ(0.003))
#define PID_SPEED_KD_DEFAULT        (0)
#define PID_SPEED_MAX_DEFAULT       (Q15_MAX)
#define PID_SPEED_MIN_DEFAULT       (Q15_MIN)
```

速度环在仅电流环控制时作速度限幅使用，可保护电机避免飞车。

2. 修改 drive_parameters_define.h 文件中的宏定义，设定电机控制模式为扭矩控制模式：

```
#define MOTOR_CONTROL_MODE          TORQUE_CONTROL_WITH_SPDLIMIT
```

3. 修改 drive_parameters_define.h 文件中的宏定义，设定最大和最小限幅电流、电流变化步长：

```
#define CONTROL_MODE_IQ_ADD_STEP        (3000.0 / 1000)
#define CONTROL_MODE_IQ_CMD_MAX         (Math_IQ(0.95))
#define CONTROL_MODE_IQ_CMD_MIN         (Math_IQ(0.1))
```

4. 通过 g_torqueCommand.iqTarget 设定电流大小，范围由 3 中电流限幅决定。

6.2 速度环控制

1. 修改 drive_parameters_define.h 文件中的宏定义，为速度环和电流环 PID 参数设置默认值：

```
#define PID_TORQUE_KP_DEFAULT        (Math_IQ(0.2))
#define PID_TORQUE_KI_DEFAULT        (Math_IQ(0.004))
#define PID_TORQUE_KD_DEFAULT        (0)
#define PID_TORQUE_MAX_DEFAULT       (Q15_MAX / 2)
#define PID_TORQUE_MIN_DEFAULT       (Q15_MIN / 2)

#define PID_FLUX_KP_DEFAULT          (Math_IQ(0.2))
#define PID_FLUX_KI_DEFAULT          (Math_IQ(0.004))
#define PID_FLUX_KD_DEFAULT          (0)
#define PID_FLUX_MAX_DEFAULT         (Q15_MAX / 2)
#define PID_FLUX_MIN_DEFAULT         (Q15_MIN / 2)

#define PID_SPEED_KP_DEFAULT         (Math_IQ(0.3))
#define PID_SPEED_KI_DEFAULT         (Math_IQ(0.003))
#define PID_SPEED_KD_DEFAULT         (0)
#define PID_SPEED_MAX_DEFAULT        (Q15_MAX)
#define PID_SPEED_MIN_DEFAULT        (Q15_MIN)
```

2. 修改 drive_parameters_define.h 文件中的宏定义，为电机设定转速加减速斜率：

```
#define ACCELERATION_RPS             (Math_IQ((600.0 * 10) / (MOTOR_MAX_SPEED_RPM * 100)))
 /*!< Acceleration of motor speed up in unit RPM/s, motor mechanical rotation speed increase per second.
600.0: 600rpm increase per second, must be float type. Can be modified to change the acceleration;
MOTOR_MAX_SPEED_RPM: motor speed max value, can not be modified;
100: 1s = 100 * (10ms), the speed increase every 10ms, can not be modified. */
#define DECELERATION_RPS             (Math_IQ((600.0 * 10) / (MOTOR_MAX_SPEED_RPM * 100)))
 /*!< Deceleration of motor speed up in unit RPM/s, motor mechanical rotation speed decrease per second.
600.0: 600rpm decrease per second, must be float type. Can be modified to change the deceleration;
MOTOR_MAX_SPEED_RPM: motor speed max value, can not be modified;
100: 1s = 100 * (10ms), the speed decrease every 10ms, can not be modified. */
```

3. 修改 drive_parameters_define.h 文件中的宏定义，设定电机控制模式为速度控制模式：

```
#define MOTOR_CONTROL_MODE          SPEED_CONTROL_MODE
```

5. 通过 g_speedCommand.speedTarget 设定转速大小，范围-32768~32767。
6. 可通过观察 g_speedCommand. speedFbkRpm 得到电机当前运转速度。

7 用户定制化

以 AC7840x 电机算法为例，Motor_ App 软件工程的文件结构如图 7-1 所示。

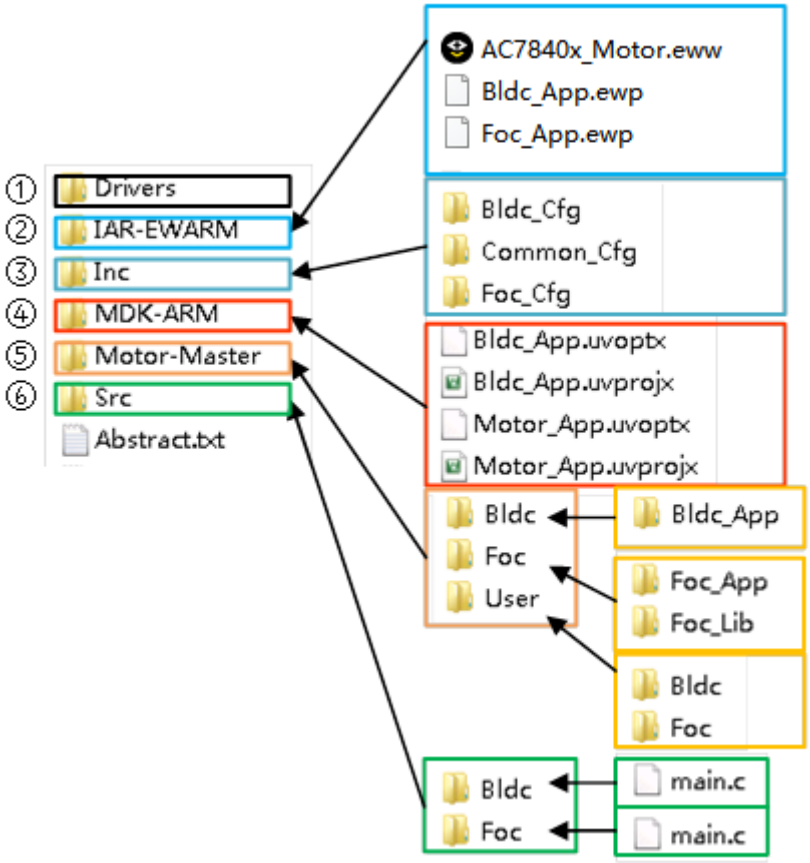

图 7-1 Motor App 文件结构

其中：

①：ATC AC78xx 系列 MCU 驱动包，用户不可编辑。

②：Motor App 的 IAR EW for ARM 工程。

③：Motor App 算法参数及硬件参数配置宏定义文件。

④：Motor App 的 MDK Keil 工程。

⑤：Motor App 的电机控制算法源文件。

⑥：Motor App 的 main 源文件。

用户只使用方波控制算法时，将②~⑥各文件路径下的 Bldc 部分保留，将 Foc 部分删除即可。

用户只使用矢量控制算法时，将②~⑥各文件路径下的 Foc 部分保留，将 Bldc 部分删除即可。

Motor_ App 软件工程的代码结构如图 7-2 所示（以 MDK Keil 工程为例，IAR EWARM 工程结构与此相同*）。

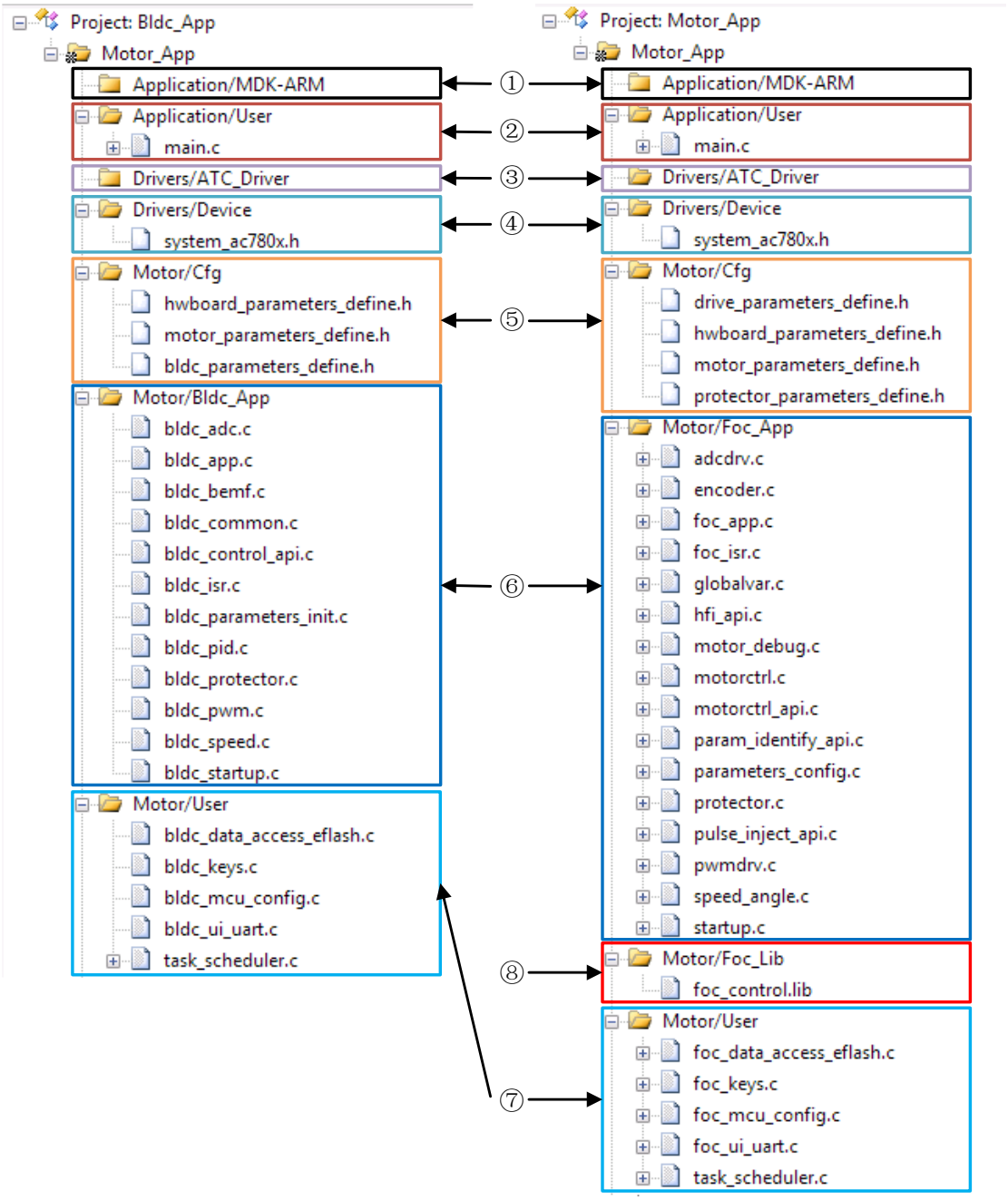

图 7-2 Motor_App 定制化分类

其中：

①：ATC AC78xx 系列 MCU 编译器支持文件。当用户安装使用了 AC78xx 系列 MCU 的 CMSIS pack，并在 MDK Keil 的 Manage Run-Time Environment 中配置选择了相应 Component，则此文件组无需

添加文件；否则需要将图 7-1 中①所内含的 ac78xx_debugout.c、startup_ac78xx.s、system_ac78xx.c 等文件链接到此文件组中。

②：Motor App 的 main.c。

③：ATC AC78xx 系列 MCU 驱动包。当用户安装使用了 AC78xx 系列 MCU 的 CMSIS pack，并在 MDK Keil 的 Manage Run-Time Environment 中配置选择了相应 Component，则此文件组无需添加文件；否则需要将图 7-1 中①所内含的 ac78xx_adc.c、ac78xx_pwm.c、ac78xx_spm.c 等文件链接到此文件组中。用户可根据实际使用 MCU 外设情况链接对应的驱动文件到此文件组中。

④：MCU 系统配置文件，规定了 MCU 的系统时钟、晶振频率等参数。用户一般无需修改此文件。

⑤：Motor App 算法参数及硬件参数配置宏定义文件组。用户需要根据硬件电路的实际参数更新此文件组中的 hwboard_parameters_define.h 文件；根据实际电机参数更新此文件组中的 motor_parameters_define.h 文件；根据使用的控制算法类型及调试情况更新此文件组中的 bldc_parameters_define.h（方波算法）或 drive_parameters_define.h（FOC 算法）文件。当使用 FOC 算法时，根据具体应用需求更新此文件组中的 protector_parameters_define.h 文件来调整保护策略的参数。

⑥：Motor App 的电机控制算法文件组。实现电机驱动的方波控制算法或 FOC 算法。

⑦：Motor App 的用户层接口文件组。提供 MCU 外设配置、上位机通信、时基任务调度等功能接口。

⑧： ATC 电机 FOC 算法库。提供三角函数、Hall 传感器角度反馈、无传感矢量控制角度估算、PID 调节器、SVPWM 算法、电机控制算法参数标幺等功能接口，方便客户根据应用需求自由开发电机控制算法。

使用不同应用硬件进行开发需要做一些定制修改适配，例如：

- **MCU 外设的配置**

修改图 7-2 中 MotorApp 软件工程框图⑦中 xx_mcu_config.c，使软件中的 GPIO 定义符合客户当前的电路板设置。

- **六步方波控制**

图 7-2 中 MotorApp 软件工程左侧框图⑥为 ATC 电机六步方波控制算法，提供六步方波换相点检测和转速指令控制功能。

- **电机控制算法**

修改图 7-2 中 MotorApp 软件工程右侧框图⑥中 speed_angle.c，使软件中的控制指令符合客户的应用需求。

- **ATC 电机算法库**

图 7-2 中 MotorApp 软件工程框图⑧为 ATC 电机算法库，提供三角函数、Hall 传感器角度反馈、无传感矢量控制角度估算、PID 调节器、SVPWM 算法、电机控制算法参数标幺等功能接口，方便客户根据应用需求自由开发电机控制算法。

- **适配多种硬件电路参数设置**

修改图 7-2 中 MotorApp 软件工程框图⑤中 hwboard_parameters_define.h，使软件中的电路板设置符合客户当前的电路板参数。

- **适配电机参数设置**

修改图 7-2 中 MotorApp 软件工程框图⑤中 motor_parameters_define.h，使软件中的电机参数设置符合客户当前的电机。

- **用户时基任务**

修改图 7-2 中 MotorApp 软件工程框图⑦中 task_scheduler.c，可自定义不同时基的任务调度

8 缩略语

表 8-1 术语缩写

缩写	全称	描述
BLDC	Brushless Direct Current Motor	直流无刷电机
PMSM	Permanent Magnet Synchronous Motor	永磁同步电机
ADC	Analog-to-Digital Converter	模拟数字转换器
FOC	Field Oriented Control	磁场定向控制
PWM	Pulse Width Modulation	脉冲宽度调制波
MRAS	Model Reference Adaptive System	模型参考自适应系统

9 参考资源

可参考的资源有：

表 9-1 相关资源简介

资源类型	资源名	资源简介
文档	ATC_AC78xx_Motor_Lib_Development_Guide_CH	AC78xx 电机算法库说明文档
文档	AC78xx Motor Driver_v1.1	AC78xx 系列电机 App 代码说明文档
文档	ATC 电机主要参数介绍及测量方法说明	电机主要参数介绍及测量方法说明