Freescale Semiconductor
设计参考手册

Document Number: DRM155
Rev. 0, 12/2014

基于MC56F84789 DSC单芯片同时控制两个增量式编码器的三相永磁同步电机的伺服驱动

1 概述

1.1 参考方案简介

本文档描述了基于单颗控制器芯片设计的同时驱动两个带增量式光电编码器的三相永磁同步电机的矢量控制伺服驱动器。本参考设计主要面向消费和工业市场应用。得益于飞思卡尔电机控制专用的数字信号控制器MC56F84789独一无二的片内资源和特性，伺服驱动器具有定位准确、静态刚度好、节约成本等优点。

目录

1.2 方案的构成和特点

本方案设计的目的是用来驱动两个带增量式光电编码器的三相永磁同步电机。以下是本方案系统的主要特点：

- 带增量式光电编码器的三相永磁同电机的矢量控制
- 三电阻电机相电流检测技术
- 直流母线电压检测，以补偿母线纹波
- 采用飞思卡尔MC56F84789数字信号控制器
- 单块功率驱动板搭配一块MC56F84789子卡的硬件架构
- 无霍尔传感器的初始转子位置搜索算法

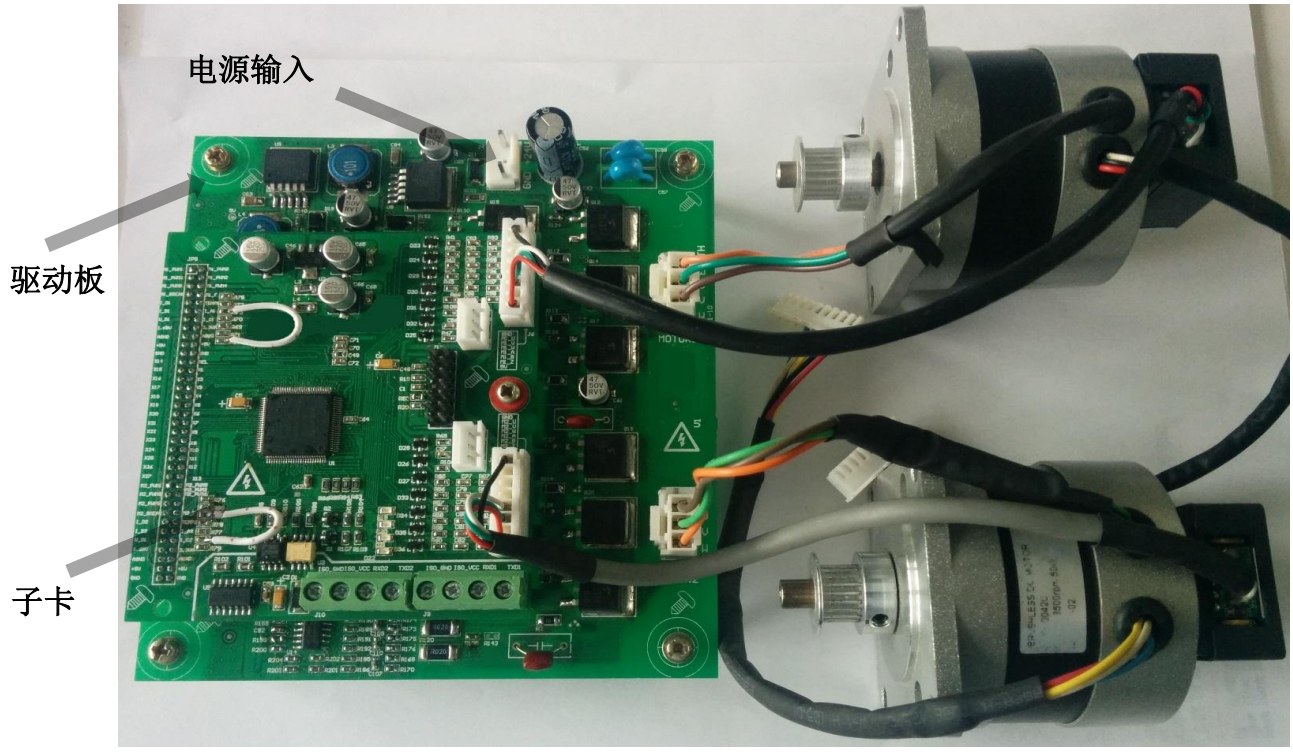

图 1 方案系统实物图

1.3 正弦永磁同步电机应用概述

在电机应用领域，正弦永磁同步电机正在替代传统的有刷直流电机、通用电机和其它类型电机成为最广泛应用的电机类型。其主要原因是永磁同步电机具有高可靠性（没有电刷），高能效，低噪声和其它电子控制优点。当然，永磁同步电机驱动的劣势是需要更为复杂的电子电路。但在今天，绝大部分的电机应用都需要电子电路来进行电机速度或转矩控制，以及其它功能控制。一旦电机控制应用中使用了电子电路，那么就可以附加很少的系统硬件成本，使用数字控制的逆变器

实现更先进的诸如正弦永磁同步电机驱动系统。飞思卡尔MC56F84789数字信号控制器是一款高性价比、具有强劲运算性能的电机控制专用控制器，非常适合正弦永磁同步电机控制应用。

相比于交流感应电机，永磁同步电机具有很多优点。由于转子带有永磁材料，转子磁场直接由转子永磁体产生，这样永磁同步电机可以获得比交流感应电机更高的效率。基于这一优势，永磁同步电机被广泛地应用于需要高可靠性和高能效的产品中，如白色家电（冰箱、洗衣机和洗碗机等）、泵、压缩机、风机和其它电器。

三相永磁同步电机通常有两种最常见的类型：正弦永磁同步电机和方波无刷直流电机（BLDC）。正弦永磁同步电机与方波无刷直流电机非常相似，但有以下两个主要差别：

- 电机结构
    - 定子感应的反电动势：正弦永磁同步电机为正弦波，方波无刷直流电机为梯形波。
- 控制方式
    - 正弦永磁同步电机采用三相正弦波控制，而方波无刷直流电机采用方波六步换向控制。

由于恒定的转矩控制，通常正弦永磁同步电机比方波无刷直流电机具有更好的性能，然而方波无刷直流电机更易于控制。方波无刷直流电机的广泛使用有其历史原因，主要因为在当时先进的控制技术和高性能的控制器还未成熟，而方波无刷直流电机因其总有一相处于非导通状态，使得使用简单的算法便可实现无位置传感器的方波六步换向控制。正弦永磁同步电机虽然需要复杂的控制技术，但却能提供更平滑的转矩、更低的噪声等好处。正如本文档所描述的，飞思卡尔MC56F84789数字信号控制器能提供两个带增量式光电编码器的三相永磁同步电机矢量控制必需的所有功能模块。

使用飞思卡尔的MC56F84789数字信号控制器，在几乎不增加系统硬件成本的情况下，可直接用三相永磁同步电机控制系统替代方波无刷直流电机控制系统。正弦永磁同步电机常见的控制算法有以下两种：

- 标量控制：V/Hz恒量控制是非常流行的标量控制技术
- 矢量控制或磁场定向控制：相比于标量控制，矢量控制带来整体驱动性能的提升，如高能效、全转矩控制、励磁和转矩解耦控制、和优异的动态性能等。

本文档所描述的永磁同步电机控制技术是更为高级的矢量控制。

对于永磁同步电机矢量控制系统，必需获得准确的电机转子初始位置，才能实现闭环启动。而本方案采用一种转子初始位置搜索法，仅利用增量式编码器，无需霍尔UVW传感器，进而降低系统成本。本文档内容包含基本电机理论、系统设计概念、软件设计、以及FreeMASTER可视化软件工具。

1.4 飞思卡尔数字信号控制器的优势和特点

飞思卡尔MC56F84789数字信号控制器不仅带有兼具DSP和MCU优点的内核，同时还集成了诸多如脉宽调制器（PWM）、模数转换器（ADC）、定时器、DMA、内部模块互联单元（XBAR），通信外设（SCI，SPI，IIC），和片内Flash和RAM存储器等专用外设模块，非常适用于数字电机控制应用。

MC56F84789集成以下功能模块：

- 100 MHz内核
  - 单周期32位乘以32位结果为32位或64位乘 - 累加指令，支持一个可选的32位并行数据搬移指令
  - 单周期16位乘以16位结果为16位或32位乘 - 累加指令，支持两个可选的16位并行数据搬移指令
  - 四个包含扩展位的36位累加器
- 片内256 KB Flash
- 片内32 KB RAM
- COP看门狗
- 中断控制器
- 系统集成模块
- 8通道高精度脉宽调制器
- 8通道标准脉宽调制器
- 定时器
- 两个16通道高速12位ADC
- 一个16通道16位SAR ADC
- 一个12位DAC
- 四个带6位参考DAC的模拟比较器
- 串行通信接口：IIC、SCI、SPI和CAN
- 四通道DMA
- 两个内部模块互联单元（XBAR）
- 与/或/非模块
- 低功耗控制模块

- 循环冗余校验码产生器

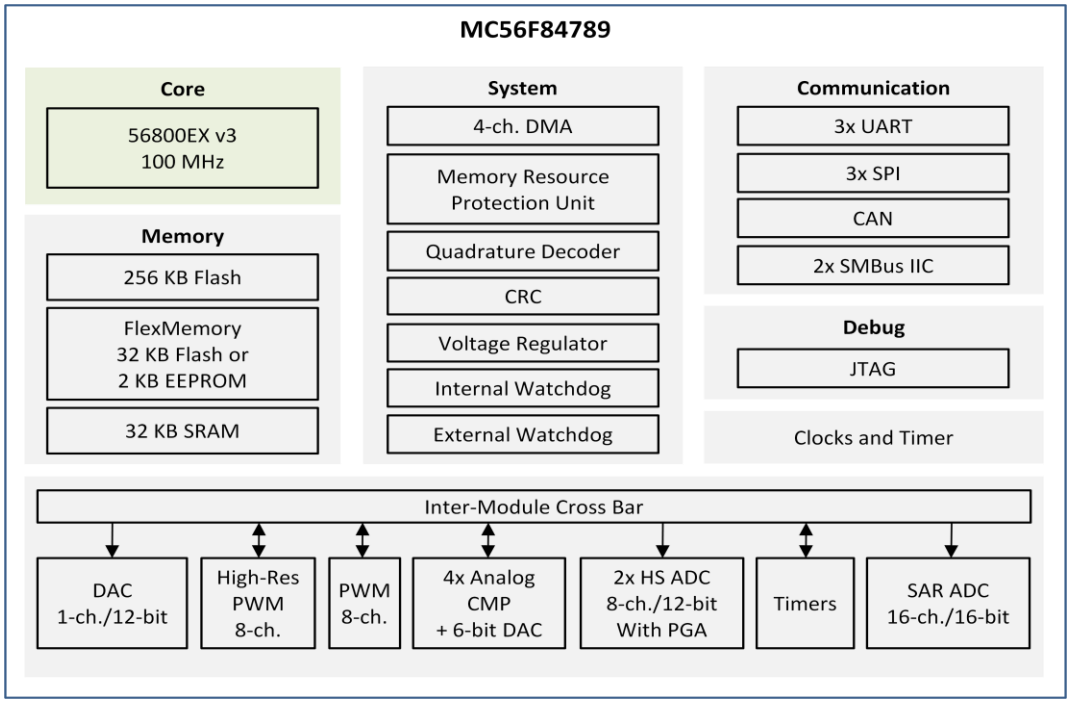

图 2 MC56F84789简化块图

永磁同步电机矢量控制和PFC控制对PWM和ADC模块有特殊的需求。MC56F84789的eFlexPWM模块提供了非常灵活的配置功能，可以实现高效的三相永磁同步电机和PFC的控制。更为甚者，该eFlexPWM模块还能在中心对齐模式下产生非对称的PWM输出。

eFlexPWM模块有以下主要特点：

- 16位分辨率，支持中心对齐、边沿对齐和非对称PWM输出
- 高精度小数延时功能，同时支持高精度PWM频率和占空比
- 改善分辨率的抖频功能
- 支持互补或独立PWM输出
- 支持符号数PWM生成
- PWM两个边沿的独立控制
- 支持与外部硬件或其它PWM模块同步功能
- 双缓冲PWM寄存器
- 每个PWM周期多个硬件触发输出
- 支持双开关PWM输出

- 每个故障输入能控制多个PWM模块输出
- 可编程的故障引脚输入滤波器
- 独立可编程的PWM输出极性
- 独立的上下管死区时间插入
- 每个互补PWM信号对有自己独立的PWM频率和死区时间
- 每个PWM输入支持独立的软件控制
- 通过FORCE_OUT事件，能同时更改所有PWM输出状态
- PWM_X能配置成为每个PWM模块第三个PWM输出信号
- 未作为PWM输出使用的通道能配置为带缓冲的比较输出功能
- 未作为PWM输出使用的通道能配置为输入捕获功能
- 增强的双边沿捕获功能
- 每个PWM互补对的源信号可以来自以下信号：
    - 外部数字引脚
    - 内部定时器通道
    - 内部模块互联单元（XBAR）输出
    - 经过高低限幅寄存器处理的外部ADC输入

12位ADC模块有以下主要特点：

- 12位分辨率
- 支持最高20 MHz输入时钟
- 高达6.67 MSPS采样率
- 单次转换时间：8.5个ADC时钟周期
- 顺序转换时间：6个ADC时钟周期
- 并行同时转换模式下，完成8个通道转换仅需26.5个ADC时钟周期
- 通过内部模块互联单元，能实现与如eFlexPWM等模块的同步操作
- 顺序扫描模式可存储多达16个测量结果
- 并行扫描模式每个ADC模块可存储多达8个测量结果
- 支持扫描暂停并在新的触发输入来临时继续原来扫描序列

- 可配置输入信号增益：x1、x2和x4
- 如果转换结果超过限幅或有过零发生，可配置在转换完成后产生中断事件
- 当扫描结束或转换结果已经就绪，可触发DMA功能搬移转换结果数据
- 通过减去预编程的偏移值，可实现采样纠正功能
- 支持有符号或无符号结果输出
- 支持单端或差分输入
- 三个模拟输入支持滞环PWM输出

16位ADC模块有以下主要特点：

- 达到16位分辨率，基于线性逐次逼近算法
- 多达24个外部单端输入通道
- 输出模式：
  - 单端16位、12位、10位或8位
  - 单端无符号右对齐格式输出
- 单次转换完成后自动返回空闲态
- 可配置的采样时间和转换速度/功耗
- 支持转换完成/硬件平均完成生成标志和中断
- 支持四个可选的时钟源输入
- 为降低噪声，可运行在低功耗模式
- 适合低噪声运行的异步时钟源，可配置为时钟输出
- 可配置的硬件通道选择和硬件转换触发信号
- 支持小于、大于、等于、在范围内或超出范围的自动比较和中断功能
- 温度传感器
- 硬件平均功能
- 可选择的参考电源：外部或内部
- 自校正模式

本方案使用ADC与PWM同步功能。此配置可以在所需的时间内完成对所需的逆变器电流和直流母线电压等模拟信号的同时转换。

1.5 参考文档

用户可从freescale.com官网上下载以下文档：

- *DSP56800E and DSP56800EX Reference Manual*，Rev.3（DSP56800ERM）
- *MC56F847xx Reference Manual*，Rev.1 （MC56F847XXRM）
- *MC56F84789 Peripherals Synchronization for Interleaved PFC Control* （AN4583）
- *Use of PWM and ADC on MC56F84789 to Drive Dual PMS Motor FOC* （AN4608）
- *FreeMASTER Software User's Manual*，freescale.com/FreeMaster
- *Freescale's Embedded Software Libraries*，freescale.com/fslesl
- 飞思卡尔电机控制，freescale.com/motorcontrol

有关最新文档版本，请访问freescale.com。

1.6 缩略语和缩写

表 1 缩略语

术语	含义
AC	交流电
ADC	模数转换器
API	应用程序接口
FSLESL	飞思卡尔嵌入式软件函数库
BEMF	反电动势
BLDC	无刷直流电机
CCW	逆时钟方向
COP	看门狗
CW	顺时钟方向
DAC	数模转换器
DC	直流电
DMA	直接存储器访问模块
DRM	设计参考手册
DT	死区时间
GPIO	通用目的输入/输出
HSCMP	高速比较器模块
I/O	计算机系统与外部世界之间的输入/输出接口

ISR	中断服务例程
IIC	集成电路总线
LED	发光二极管
PFC	功率因数校正
DSC	数字信号控制器
PDB	可编程延时模块
PLL	锁相环
PWM	脉冲宽度调制器
RPM	每分钟转速
XBAR	内部模块互联单元
SCI	串行通信接口模块
SPI	串行外设接口模块

1.7 符号列表

表 2 符号索引列表

符号	定义
d, q	正交旋转坐标系
g_ω	自适应速度控制增益
u_γ, u_δ	Alpha/Beta 反电动势观测器误差
i_{sa}, i_{sb}, i_{sc}	a、b、c 相定子电流
$i_{sd,q}, i_{s(d,q)}$	d, q坐标系定子电流
i'_q	i_q 电流的一阶导数
$i_{s(d,q)^*}, i_{\gamma,\delta}$	估计d, q坐标系定子电流
$i_{s\alpha,\beta}, i_{s(\alpha,\beta)}$	α, β坐标系定子电流
$\hat{i}_{sg}$	通用坐标系的定子电流空间矢量
i_{sx}, i_{sy}	通用坐标系的定子电流空间矢量分量
$\hat{i}_{rg}$	通用坐标系的转子电流空间矢量
i_{rx}, i_{ry}	通用坐标系的转子电流空间矢量分量
J	机械惯量
K_M	电机常数
k_e	反电动势常数
L_s	定子相电感

L_{sd}, L_D	d轴定子电感
L_{sq}, L_Q	q轴定子电感
p	电机极对数
R_s	定子相电阻
s	导数算子
T_L	负载转矩
$u_{s\alpha,\beta}, u_{s(\alpha,\beta)}$	α, β坐标系定子电压
$u_{sd,q}, u_{s(d,q)}$	d, q坐标系定子电压
a, b	定子正交坐标系
$\Psi_{s\alpha,\beta}$	α, β坐标系定子磁通
$\Psi_{sd,q}$	d, q坐标系定子磁通
Ψ_M	转子磁通
θ_r	α, β坐标系转子位置角
$\omega, \omega_r, \omega_{el}/\omega_F$	转子电角速度/磁场角速度

2 控制原理

2.1 三相永磁同步电机

永磁同步电机是一种类似于感应电机带有三相定子绕组的旋转电机，通常其转子永磁体为表贴式，如图 3所示。

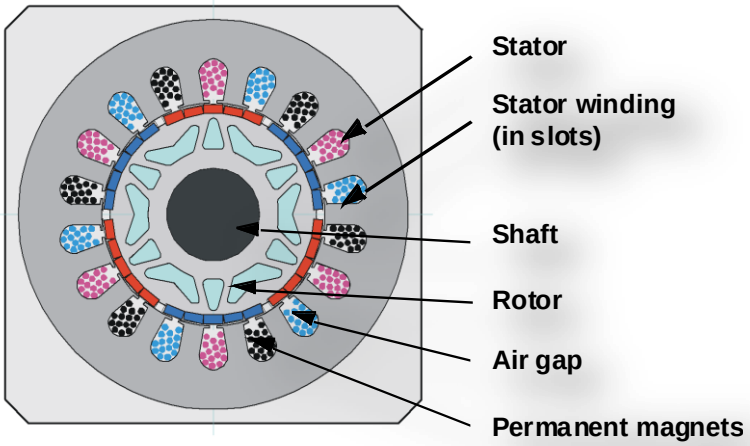

图 3 永磁同步电机-剖面图

从某种角度来说，永磁同步电机可以等效于感应电机，其合成气隙磁场中的转子磁场由永磁体产生，且转子磁场是恒定的。在设计高性能的运动控制系统中永磁同步电机有很多优势。使用永磁体产生气隙磁通可设计出非常高效的永磁电机。

2.2 永磁同步电机的数学模型

永磁同步电机有很多数学模型。用于矢量控制的模型可以从空间矢量理论的角度去分析。三相电机物理量（诸如电压，电流，磁通等）可以用综合空间矢量来表示。这个模型适用于任何电压电流瞬变状态，可以同时表征电机在稳态和瞬态的运行性能。综合空间矢量可仅使用两个正交坐标轴来表示。因此我们可以将电机看成两相电机。使用该两相电机模型可以减少电机方程数量，从而简化控制设计分析。

2.2.1 空间矢量的定义

我们假设i_{sa}, i_{sb}, i_{sc} 是对称的瞬态三相定子电流：

$$i_{sa} + i_{sb} + i_{sc} = 0$$

等式 1

这时我们可以定义如下的定子电流空间矢量：

$$\overline{i_s} = k(i_{sa} + ai_{sb} + a^2 i_{sc})$$

等式 2

这里的a 和a^2 是空间算子$a = e^{j2\pi/3}$, $a^2 = e^{j4\pi/3}$, k 基于磁势不变的变换常数，$k=2/3$。图 4所示为定子电流空间矢量投影。

等式 2定义的空间矢量可以使用两相坐标理论表示。空间矢量的实部等价表示瞬态直轴定子电流分量$i_{s\alpha}$， 而虚部等价表示瞬态交轴定子电流分量$i_{s\beta}$，因此在建立在定子上的静止参考坐标系里的定子电流空间矢量可以表示为：

$$\overline{i_s} = i_{s\alpha} + ji_{s\beta}$$

等式 3

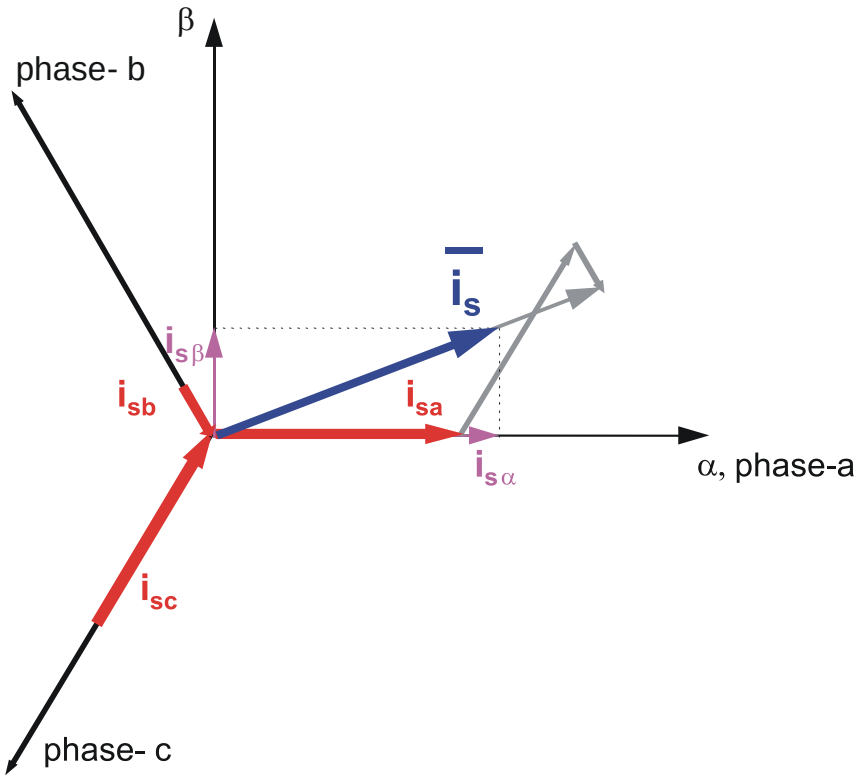

图 4 定子电流空间矢量与投影

在三相对称电机里，交直轴定子电流分量是虚拟的两相电流分量，可以用如下三相定子电流表示：

$$i_{s\alpha} = k\left(i_{sa} - \frac{1}{2}i_{sb} - \frac{1}{2}i_{sc}\right)$$

等式 4

$$i_{s\beta} = k\frac{\sqrt{3}}{2}(i_{sb} - i_{sc})$$

等式 5

这里的*k=2/3* 是基于磁势不变的变换常数，所以最终的方程为：

$$i_{s\alpha} = \frac{1}{3}(2i_{sa} - i_{sb} - i_{sc})$$

等式 6

$$i_{s\beta} = \frac{1}{\sqrt{3}}(i_{sb} - i_{sc})$$

等式 7

其它的电机物理量的空间矢量（电压，电流，磁链等）都可以用如上定子电流空间矢量的方法表示。

2.2.2 永磁同步电机数学模型

在对永磁电机数学模型分析之前，我们假设电机气隙均匀，绕组产生的磁场按正弦规律分布，那么定子电压动态方程可以表示为：

$$u_{SA} = R_S i_{SA} + \frac{d}{dt}\psi_{SA}$$

等式 8

$$u_{SB} = R_S i_{SB} + \frac{d}{dt}\psi_{SB}$$

等式 9

$$u_{SB} = R_S i_{SC} + \frac{d}{dt}\psi_{SC}$$

等式 10

这里的u_{SA}，u_{SB}，u_{SC} 定子电压的瞬态值，i_{SA}，i_{SB}，i_{SC} 是定子电流的瞬态值，ψ_{SA}，ψ_{SB}，ψ_{SC} 定子磁链的瞬态值，以上分别对应A，B，C三相。

由于在等式 8，等式 9，等式 10中含有大量的瞬态变量比较复杂，所以使用两相坐标系理论（Clarke变换）重构电机瞬态方程会更简单实用。那么永磁同步电机在两相静止坐标系下可表示为：

$$u_{S\alpha} = R_S i_{S\alpha} + \frac{d}{dt}\psi_{S\alpha}$$

等式 11

$$u_{S\beta} = R_S i_{S\beta} + \frac{d}{dt}\psi_{S\beta}$$

等式 12

$$\psi_{S\alpha} = L_{S\alpha} i_{S\alpha} + \psi_M \cos\Theta_r$$

等式 13

$$\psi_{S\beta} = L_{S\beta} i_{S\beta} + \psi_M \sin \Theta_r$$

等式 14

$$\frac{d\omega}{dt} = \frac{p}{J}\left[\frac{3}{2} p\left(\psi_{S\alpha} i_{S\beta} - \psi_{S\beta} i_{S\alpha}\right) - T_L\right]$$

等式 15

相关符号术含义请语参见表 2。

等式 11到等式 15就是基于α, β 定子静止坐标系表示的永磁同步电机的数学模型。

除了基于定子静止坐标系，电机电压矢量方程还可以基于一种更通用的旋转坐标系来表示，其旋转速度为 ω_g。如果基于旋转坐标系，直轴交轴分量 *x,y* 以瞬时速度$\omega_g = d\theta_g/dt$旋转，如图 5所示，这里的角度θ_g 就是定子静止坐标系α轴分量与通用坐标系 *x*轴分量的夹角，等式 16定义了通用坐标系下定子电流空间矢量方程：

$$\overline{i_{sg}} = \bar{i}_s e^{-j\Theta_g} = i_{sx} + j i_{sy}$$

等式 16

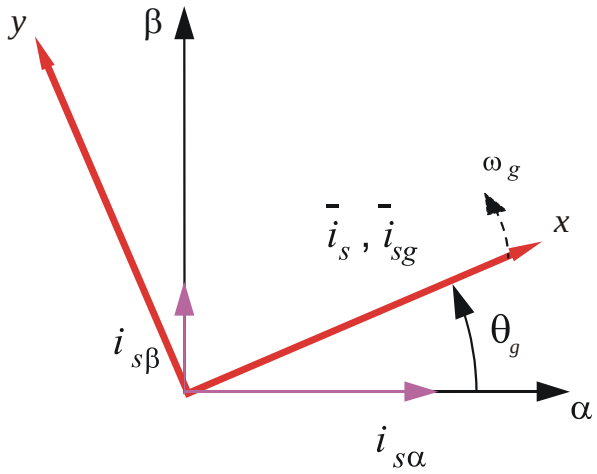

图 5 通用坐标的应用

同样，定子电压和磁链空间矢量也可以用通用坐标系表示。

同样地，转子电压、电流、磁链空间矢量也可以用类似的分析方法。定位于转子的坐标系实轴与定子坐标系直轴夹角为θ_r。我们可以看到，转子坐标系的实轴与通用坐标系的实轴x的夹角为θ_g-θ_r，所以在通用坐标系中，转子电流空间矢量可以表示为：

$$\overline{i_{rg}} = \bar{i}_r e^{-j(\Theta_g - \theta_r)} = i_{rx} + j i_{ry}$$

等式 17

这里的$\bar{i}_r$ 是在转子参考坐标系里的转子电流空间矢量。

同样，转子电压和磁链空间矢量也可以用通用坐标系表示。

通用坐标系电机电压方程可以利用介绍过的坐标变换从静止坐标系旋转变换到通用坐标系来推导。矢量控制经常使用永磁同步电机模型。矢量控制的目的就是使用类似直流电机的控制方案实现高性能的交流电机动态控制。为实现这个目标，可以把参考坐标系定位于定子磁链空间矢量，转子磁链空间矢量，或者气隙空间矢量。最常见的就是定向在转子磁链空间矢量，其直轴为d，交轴为q。

经过dq坐标变换，电机模型可以表示如下：

$$u_{Sd} = R_S i_{Sd} + \frac{d}{dt}\psi_{Sd} - \omega_F \psi_{Sq}$$

等式 18

$$u_{Sq} = R_S i_{Sq} + \frac{d}{dt}\psi_{Sq} + \omega_F \psi_{Sd}$$

等式 19

$$\psi_{Sd} = L_S i_{Sd} + \psi_M$$

等式 20

$$\psi_{Sq} = L_S i_{Sq}$$

等式 21

$$\frac{d\omega}{dt} = \frac{p}{J}\left[\frac{3}{2}p\left(\psi_{Sd} i_{Sq} - \psi_{Sq} i_{Sd}\right) - T_L\right]$$

等式 22

考虑到下面内容都是基于 i_{sd}=0，等式 22可以简化为如下形式：

$$\frac{d\omega}{dt} = \frac{p}{J}\left[\frac{3}{2}p\left(\psi_M i_{Sq}\right) - T_L\right]$$

等式 23

从等式 23可以看到转矩是完全解耦的，只受控于电流i_{sq} 。

2.3 永磁同步电机的矢量控制

2.3.1 矢量控制基本原理

高性能的电机控制表现为整个速度范围运行平稳，零速下全转矩控制，和快速的加减速特性。三相交流电机一般使用矢量控制来实现上述目标。矢量控制技术又称磁场定向控制（FOC）。FOC最基本的思想就是将定子电流解耦成一个控制磁场的电流分量和一个控制转矩的电流分量。经过解耦后，两个电流分量独立受控，互不干扰。这时电机的控制器结构就和他励直流电机控制器一样简单。

图 6展示了永磁同步电机基本的矢量控制算法结构。为实现矢量控制，需要执行下列步骤：

1. 检测电机物理量（相电压和相电流）
2. 使用Clarke变换将三相定子电流变换到两相坐标系(α，β)
3. 计算转子磁通空间矢量的幅值和相角
4. 使用Park变换将αβ轴定子电流旋转变换到dq坐标系
5. 分别独立控制转矩电流（i_{sq}）分量和磁场电流（i_{sd}）分量
6. 使用解耦模块计算输出定子电压空间矢量
7. 定子电压空间矢量经过反Park变换从dq坐标系变换到αβ两相坐标系
8. 使用空间矢量调制，产生三相电压输出

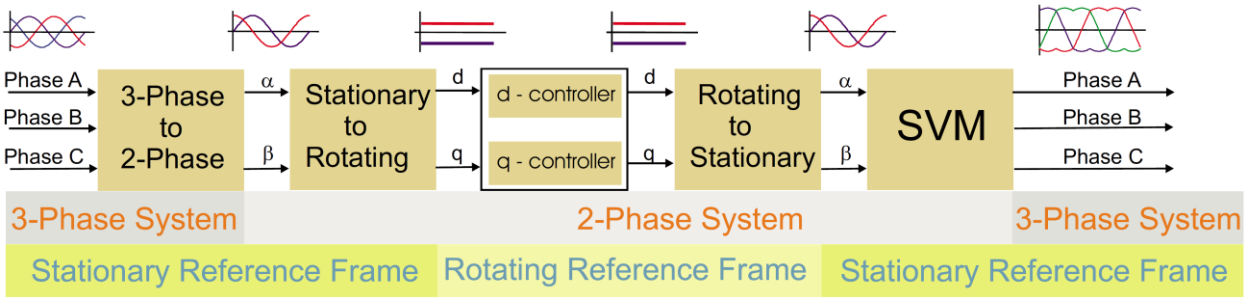

图 6 矢量控制的变换关系图

为了将定子电流分解为转矩分量和磁通分量，必需知道电机励磁磁通的位置，这需要精确检测转子的位置和速度信息。在矢量控制系统中，通常使用安装在转子上的增量式编码器或解析器来作为转子位置传感器。

2.3.2 矢量控制算法

图 7所示的就是实现矢量控制算法系统框图。其它矢量定向控制方案和其类似，可以实现电机磁场和转矩的独立控制。控制的目标是调节电机的转速。速度命令值由上层控制设定。具体算法由两个控制环路实现。快速内环62.5微秒执行一次。慢速外环0.5毫秒执行一次。

为实现精确的永磁同步电机速度控制，需要采集反馈信号。必需的反馈信号是：三相定子电流和电压。对于定子电压，使用电流环的输出电压。为实现准确运行，现有的控制架构需要一个位置和速度传感器。

快速控制环执行两个独立的电流环控制。他们就是直轴和交轴(i_{sd}, i_{sq}) PI控制器。直轴电流用于控制转子励磁磁通。交轴电流则控制电机转矩。电流PI控制器输出需要再加上相应的解耦电压部分，这就得到了施加于电机上理想的定子电压空间矢量。快速控制环执行以下必要的任务来完成对定子电流分量的独立控制：

- 三相电流重构

- Clarke变换
- Park变换和反Park变换
- 定子电压解耦
- 直流母线电压谐波补偿
- 空间矢量调制（SVM）

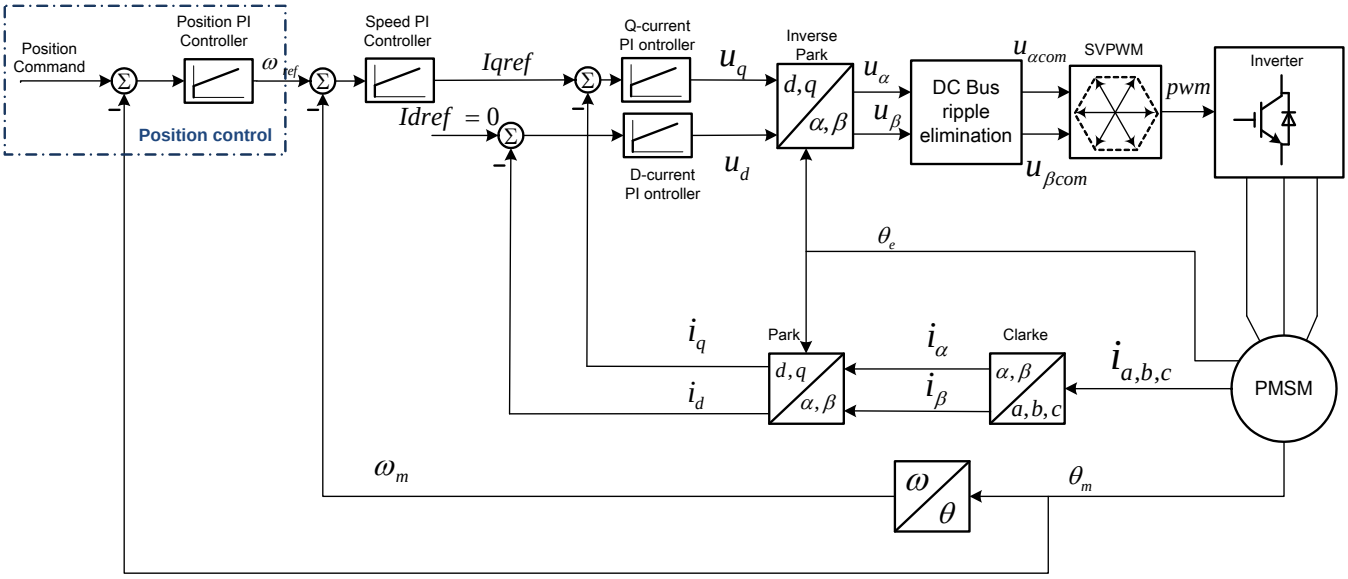

图 7 PMSM矢量控制算法架构

慢速控制环执行速度控制器和低级别的控制任务。位置控制器的输出作为速度参考值，速度控制器输出作为产生转矩的交轴电流（i_{sq}）的参考值。

2.3.3 空间矢量调制

空间矢量调制（SVM）可以直接将两相α,β-坐标系电压转化为脉宽调制（PWM）信号（占空比）。

标准的输出电压产生方法是使用反Clarke变换得到三相电压值。得到三相电压值就可以计算相应的用于功率开关的占空比。尽管上述做法效果很好，但是空间矢量调制更简单直接（直接从 α,β 坐标系变换得到占空比）。

标准空间矢量调制技术的基本原理可以用如图 8所示的功率回路原理图来解释。图 8中所示的三相功率回路可以产生8种开关状态（矢量）。这些矢量由相应的功率开关组合形成。**Error! Reference source not found.**所示是由所有的开关状态组合而形成的六边形。8个矢量由定义在 α,β 坐标系6个运动矢量U_0, U_{60}, U_{120}, U_{180}, U_{240}, U_{300}和2个零矢量 O_{000} , O_{111}组成。

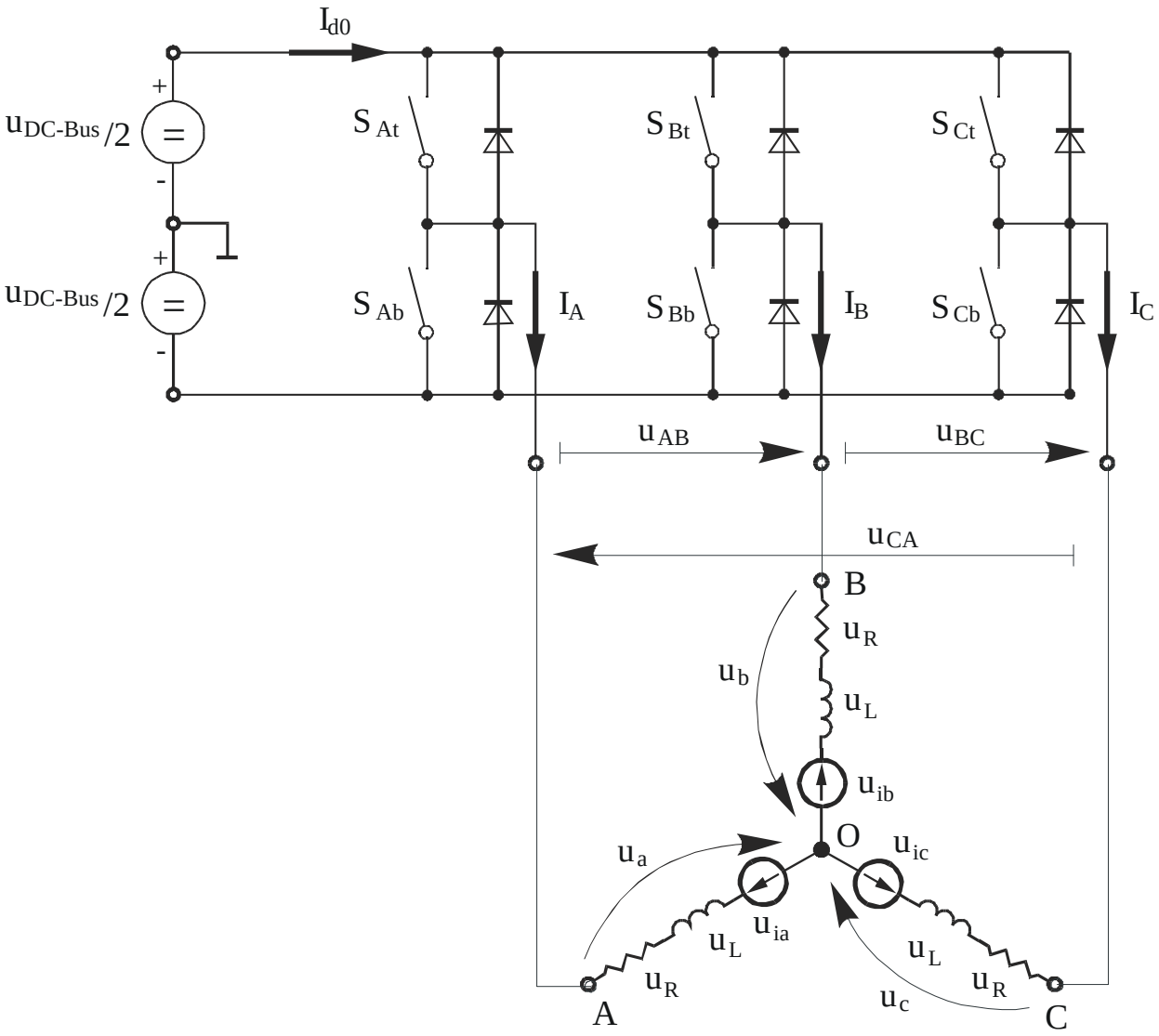

图 8 功率回路原理图

每个电压矢量所对应的功率回路开关的开通关断状态将以图 9括号里的数字所示进行编码。一个数字对应一相开关状态。对每一相来说，1代表上管开通，下管关断。0代表上管关断，下管开通。表 3列出了这些状态组合，相应的瞬态线电压输出，相电压输出和电压矢量。

表 3 开关模态和对应的定子电压

a	b	c	U_a	U_b	U_c	U_{AB}	U_{BC}	U_{CA}	Vector
0	0	0	0	0	0	0	0	0	O_{000}
1	0	0	$2U_{DC\text{-}Bus}/3$	$-U_{DC\text{-}Bus}/3$	$-U_{DC\text{-}Bus}/3$	$U_{DC\text{-}Bus}$	0	$-U_{DC\text{-}Bus}$	U_0
1	1	0	$U_{DC\text{-}Bus}/3$	$U_{DC\text{-}Bus}/3$	$-2U_{DC\text{-}Bus}/3$	0	$U_{DC\text{-}Bus}$	$-U_{DC\text{-}Bus}$	U_{60}
0	1	0	$-U_{DC\text{-}Bus}/3$	$2U_{DC\text{-}Bus}/3$	$-U_{DC\text{-}Bus}/3$	$-U_{DC\text{-}Bus}$	$U_{DC\text{-}Bus}$	0	U_{120}
0	1	1	$-2U_{DC\text{-}Bus}/3$	$U_{DC\text{-}Bus}/3$	$U_{DC\text{-}Bus}/3$	$-U_{DC\text{-}Bus}$	0	$U_{DC\text{-}Bus}$	U_{240}
0	0	1	$-U_{DC\text{-}Bus}/3$	$-U_{DC\text{-}Bus}/3$	$2U_{DC\text{-}Bus}/3$	0	$-U_{DC\text{-}Bus}$	$U_{DC\text{-}Bus}$	U_{300}
1	0	1	$U_{DC\text{-}Bus}/3$	$-2U_{DC\text{-}Bus}/3$	$U_{DC\text{-}Bus}/3$	$U_{DC\text{-}Bus}$	$-U_{DC\text{-}Bus}$	0	U_{360}
1	1	1	0	0	0	0	0	0	O_{111}

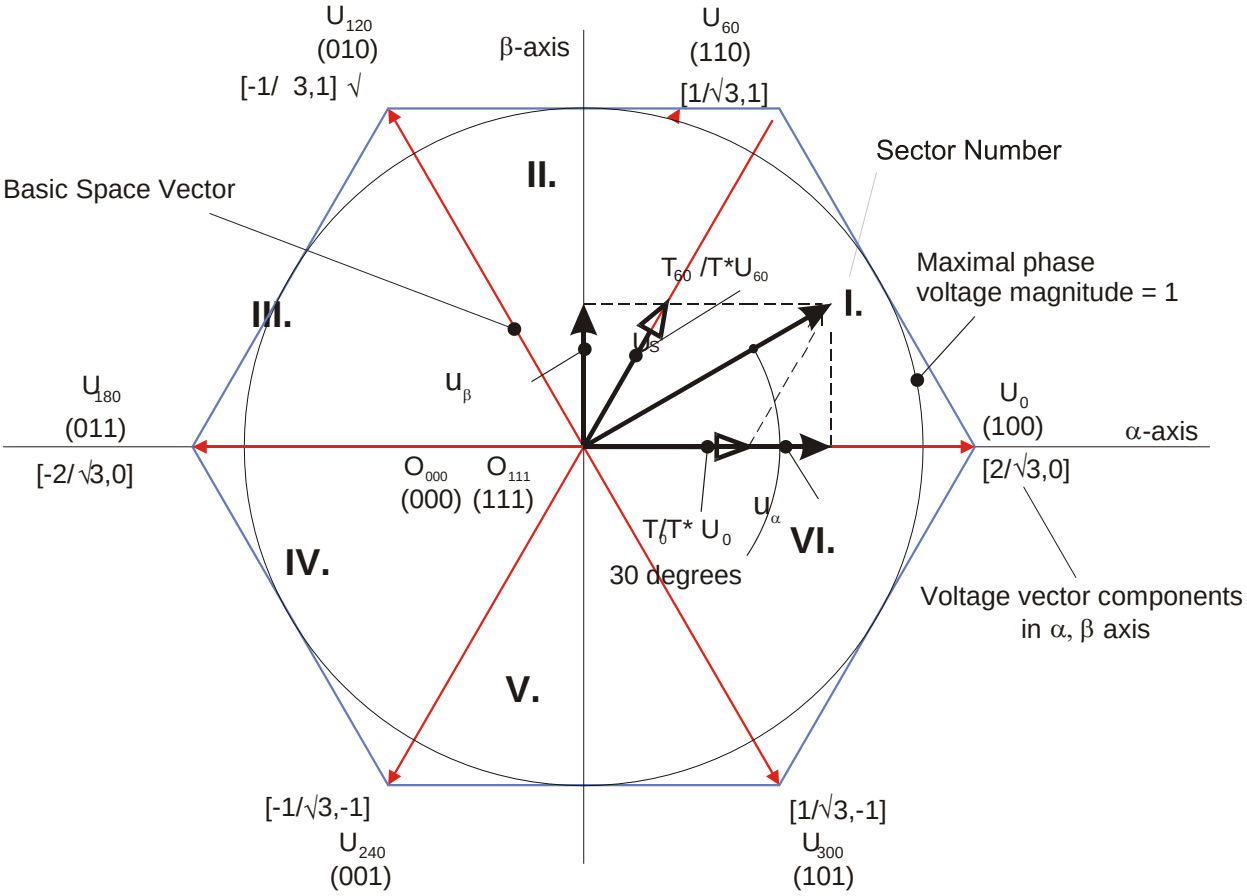

图 9 基本的空间矢量和电压矢量投影关系

SVM其实就是作为矢量控制（电压空间矢量）和PWM之间的一座桥梁。

SVM技术由以下几步组成：

1. 扇区判断
2. 空间矢量分解为扇区对应的两个相邻电压矢量 U_x，$U_{x\pm60}$
3. PWM占空比计算

SVM的原理就是利用固有基本电压矢量U_{XXX}和O_{XXX}在一个PWM周期T_{PWM} 内合成的平均矢量来等效期望的电压矢量。

这个方法在一个PWM周期内可以灵活配置零矢量和运动矢量。可以选择以降低开关损耗为目的的配置，也可以为达到不同的效果选择其它的配置，例如中心对齐的PWM，边沿对齐的PWM，最少的开关切换的PWM等。

对于SVM，我们定义以下规则：

- 期望的电压空间矢量由所在扇区的固有基本电压矢量合成：运动矢量 (U_x, $U_{x\pm60}$) 和零矢量 (O_{000} or O_{111}).

  SVM原理可以用下面的表达式表示：

$$T_{PWM} \cdot U_{S[\alpha,\beta]} = T_1 \cdot U_X + T_2 \cdot U_{X\pm60} + T_0 \cdot (O_{000} \vee O_{111})$$

等式 24

$$T_{PWM} = T_1 + T_2 + T_0$$

等式 25

  为算出T_0，T_1，T_2时间，就必须将空间电压矢量$U_{S[\alpha,\beta]}$ 分解到运动电压矢量U_x, $U_{x\pm60}$上。等式 24可以分解为等式 26和等式 27。

$$T_{PWM} \cdot U_{SX} = T_1 \cdot U_X$$

等式 26

$$T_{PWM} \cdot U_{S(X\pm60)} = T_2 \cdot U_{X\pm60}$$

等式 27

  通过计算这些方程，我们就可以计算出电压矢量所在扇区固有基本电压矢量在一个PWM周期内的作用时间，以此来产生正确的定子电压。

$$T_1 = \frac{|U_{SX}|}{|U_X|} T_{PWM}$$

等式 28 运动矢量U_X作用时间

$$T_2 = \frac{|U_{SX}|}{|U_{X\pm60}|} T_{PWM}$$

等式 29 运动矢量$U_{X\pm60}$作用时间

$$T_0 = T_{PWM} - (T_1 + T_2)$$

等式 30 零矢量 O_{000} 和 O_{111}作用时间

2.4 转子位置与速度的检测

MC56F84789芯片内部有两组16-bit定时器模组：Quad Timer A与Quad Timer B，其中每个模组里含有4个Timer。本应用中Quad Timer A中的4个Timer用于电机1的位置和转速的检测，Quad Timer B中的4个Timer用于电机2的位置和转速的检测。本章节中仅描述电机1的位置与转速的检测，电机2的位置与转速检测与电机1相同。Quad Timer A模组中有4个定时器，分别命名为TMRA0、TMRA1、TMRA2与TMRA3。每个定时器都有两个输入端：primary input以及secondary input，同时还有一个输出信号OFLAG。在Crossbar的作用下，primary和secondary的输入信号可以从芯片引脚或者内部信号而来，OFLAG也可以灵活的给芯片内部的其他模块使用或者输出至芯片引脚。有关Quad Timer和Crossbar的相关描述请参考MC56F847xx Reference Manual。

- TMRA0工作于Quadrature count mode，用于检测转子位置；
- TMRA1工作于Count mode，结合TMRA0，通过捕捉TMRA0/1的计数器值来检测转速；
- TMRA2工作于Count mode，用于检测转子旋转的圈数；
- TMRA3工作于Count mode，用于提高低转速检测精度。

利用Quad Timer A模组检测位置、转速以及旋转圈数如图 10所示。

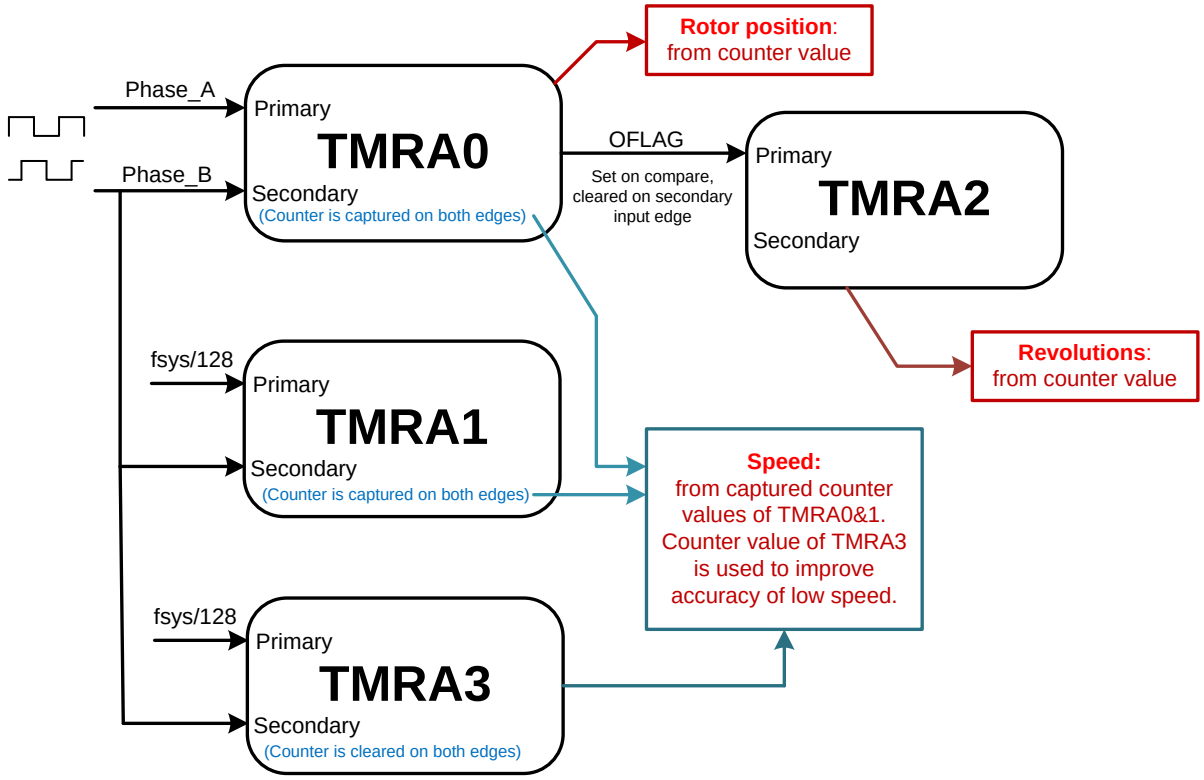

图 10 用Quad Timer A检测转子位置、转速和旋转圈数

2.4.1 位置的检测

电机的转轴上安装有增量式光电脉冲编码器，随着转子旋转，增量式编码器输出两组脉冲序列，即A相和B相，AB两相脉冲序列之间相差90°相位。将TMRA0配置为Quadrature count模式，A、B两相信号分别与其primary和secondary输入端连接。A、B信号的每个边沿都将触发其计数器计数。A相超前B相时，计数器增计数，反之减计数，如图 11所示。

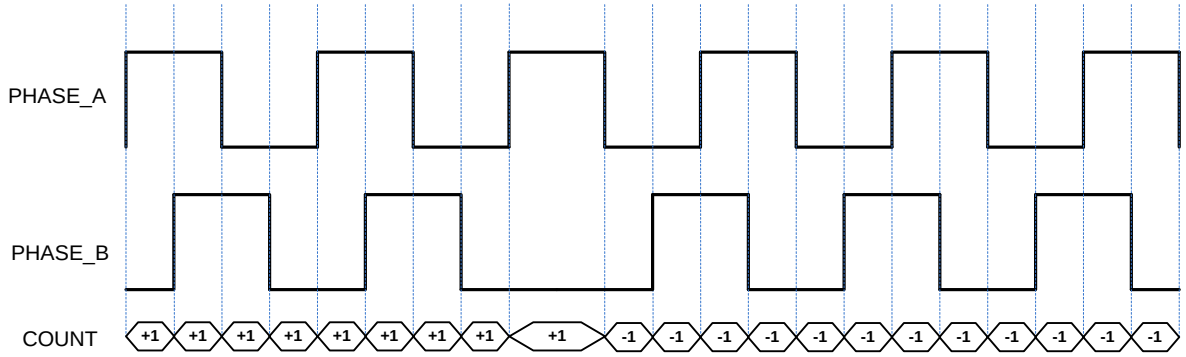

图 11 Timer工作于正交编码器信号解码模式

由于A、B信号的每个沿都触发计数器计数，那么TMRA0相当于在输入信号的四倍频下计数。每个Timer都有两个比较寄存器COMP1和COMP2：

- 计数器向上增计数时，计数器值与COMP1的值比较，匹配时计数器值将从LOAD寄存器中重载；
- 计数器向下减计数时，计数器值与COMP2的值比较，匹配时计数器值亦从LOAD寄存器中重载。

本应用中，编码器为1000线，将COMP1设置为3999，COMP2设置为0xF061（即-3999），LOAD设置为0。假设编码器A相信号超前B相为正向旋转，那么电机正转时，TMRA0计数器从0开始计数增至3999，然后发生比较匹配，于是从LOAD寄存器中载入0开始计数，每次比较匹配代表电机旋转了一个机械周期；电机反向旋转时，TMRA0计数器从0开始向下减计数至-3999，然后发生比较匹配，于是从LOAD寄存器中载入0开始计数，每次比较匹配代表电机旋转了一个机械周期。通过读取TMRA0计数器的值可以换算得到转子的位置。

图 12给出了如何由TMRA0的计数器值得到转子机械位置的示意图。TMRA0工作在Quadrature count模式下，横轴为时间，纵轴为计数器值。电机正向旋转时，计数器从0增计数至3999，发生比较匹配后重装为0；反向旋转时，计数器减计数至0时，由于计数器本身只有16位，故在下一个输入边沿的作用下减至0xFFFF，电机继续反向旋转，计数器从0xFFFF向下减计数，减至0xF061时，发生比较匹配后重装为0（下一个输入边沿减至0xFFFF）。

- 若把计数器值看做无符号数，图 12中最上端的波形给出了这一过程的示意图。
- 若程序中读出计数器值，并将其参与运算，那么其会被当作有符号数（补码）进行计算，见图 12中间的波形。如此配置之下，电机正向旋转时，计数器在0~3999之间增计数；反向旋转时，计数器在0~ -3999之间减计数。
- 若将此计数器值（看作补码）乘上16.38，由于乘法结果限制在16位并禁止了结果饱和功能，电机正向运行时得到从-32768~32767增计数的位置变量值、反向运行时得到从32767~ -32768减计数的位置变量值，见图 12中最下端波形。若将该位置变量看做Q1.15格式，那么其值在-1~1之间变化，将其与-pi~pi对应便得到转子机械位置。具体实现见章节4“软件设计”。

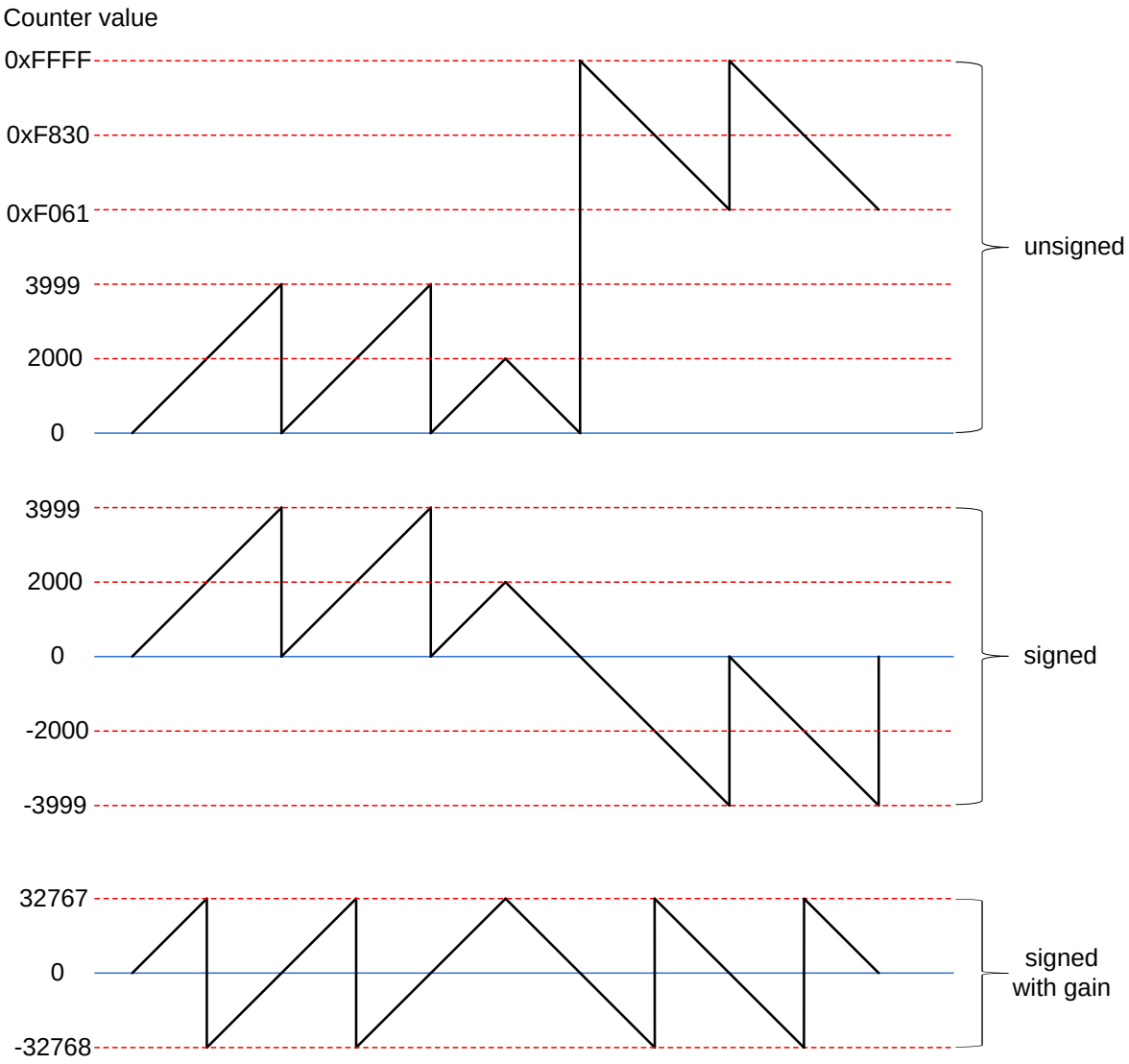

图 12 由TMRA0计数器值得到转子机械位置（quadrature count mode）

本应用使用的电机是2对极，故只需要将转子机械角度乘以2便得到转子的电角度位置。

2.4.2 Z信号处理

一般增量式光电脉冲编码器除了输出 AB 两组脉冲序列，还输出一路Z信号。Z信号是电机转子每转一圈输出的一个零位脉冲，表示转过一圈。输出 Z 信号的点称为零位。Z信号可以方便的用来判断转子转过的圈数。

有些低成本的编码器没有Z信号，本方案中针对不带Z信号的增量式编码器，在工作于quadrature count模式的TMRA0的基础上再添加一个定时器TMRA2可生成一个模拟 Z 信号，节约硬件成本，如图 10所示。

2.4.2.1 由Quad Timer模拟Z信号

Quad Timer模组中每个定时器内部都有一个output flag信号，称之为OFLAG。这个信号可以输出至芯片的引脚上，或给内部其他模块使用，或禁止输出。TMRA0如2.4.1节所述中配置，其OFLAG信号作为TMRA2的primary输入。

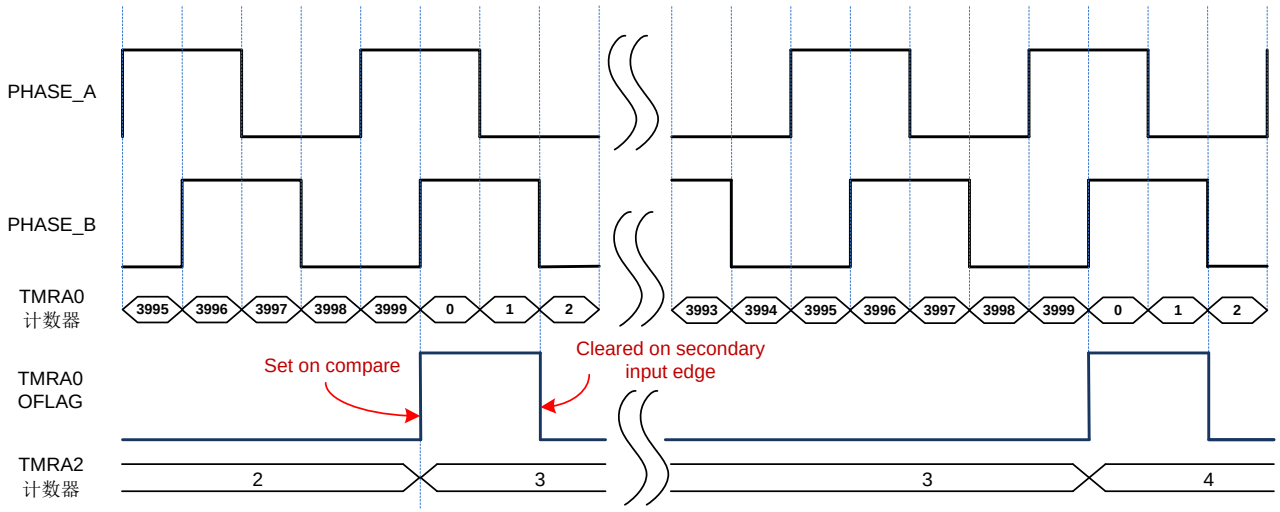

图 13 由Quad Timer模拟Z信号

以 1000线增量式光电编码器为例，TMRA0的COMP1设置为3999，COMP2设置为-3999，OFLAG的输出方式设置为“Set on compare, cleared on secondary input edge”，即比较匹配时OFLAG信号置1，然后由secondary输入信号的边沿清零。以编码器A相超前B相为例，如图 13所示，当TMRA0计数器值与COMP1匹配后，OFLAG置1，由于编码器B相信号是其secondary输入，故下一个B相信号的边沿使得OFLAG清零。TMRA0的OFLAG信号上的脉冲作为TMRA2的primary输入，驱动其计数器向上计数。对于TMRA0，电机每转一圈其OFLAG信号输出一个脉冲，即模拟Z信号，故TMRA2计数器值可以反映电机旋转的圈数。

2.4.2.2 模拟Z信号的正反转判断

在一次位移过程中可能有正转一圈也可能有反转一圈的情况，但模拟Z信号始终触发TMRA2计数器向上累加，故需要软件对其进行修正，具体思路为：

1. 定义一个全局变量RealZCnt，作为记录转子旋转圈数的计数器，而不是直接使用TMRA2计数器值。
2. 第k次执行位置环/速度环时，记录TMRA0的计数器值ENC(k)和TMRA2的计数器值REV(k)。计算ENC(k)-ENC(k-1)，从结果的符号得知当前电机是正转还是反转。若电机正转，RealZCnt += [REV(k)-REV(k-1)]；反之RealZCnt -= [REV(k)-REV(k-1)]。最后更新ENC(k-1)和REV(k-1)的值。

该方法流程如图 14所示：

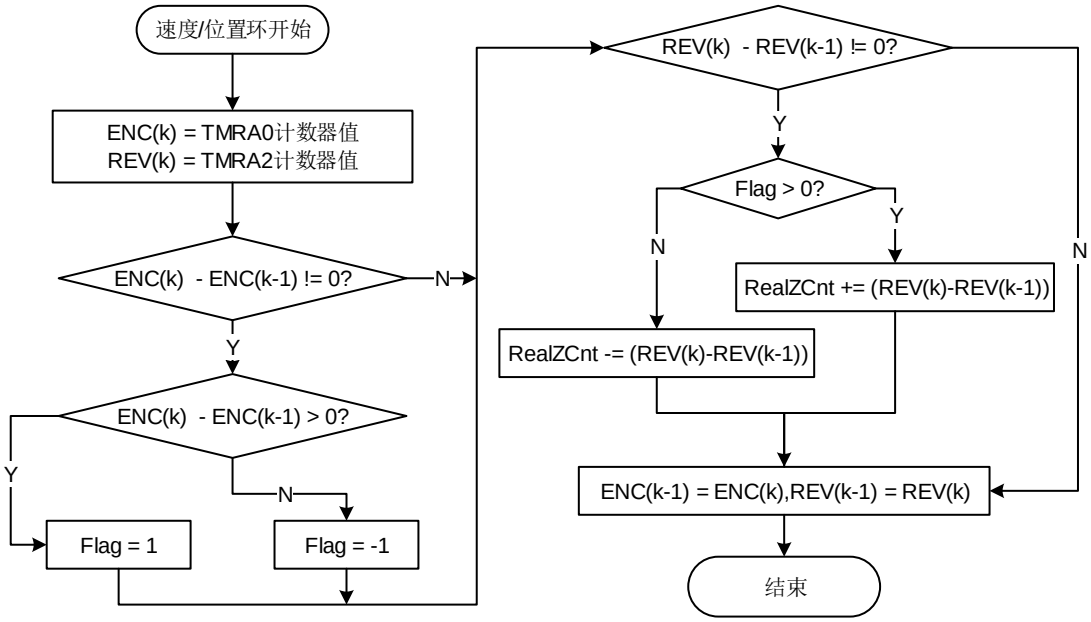

图 14 模拟Z信号的处理

2.4.3 转子初始位置检测

2.4.3.1 基于增量式编码器的初始位置搜索算法

增量式光电编码器的精度远高于霍尔信号，但是编码器脉冲信号只能提供转子相对位置，并不能直接反应转子的绝对位置。

由矢量控制原理可知，坐标变化所需的转子位置是转子直轴（d轴）与A相绕组轴线之间的夹角。在空间某个位置输出一个电流矢量，如果这个电流矢量足够大且与转子呈一定角度，就会引起转子向该电流矢量方向转动。增量式编码器可以检测到转子微小的偏移量以及偏移方向，体现在TMRA0计数器数值的变化。根据不同的偏移方向，可以将转子所在角度范围分为两个区间，这样就将转子存在的角度范围缩小到二分之一，即二分法。只要输出的电流矢量与转子不重合，转子就会发生微小偏移，不断的改变输入电流矢量的方向，便能通过二分法缩小估计的转子所在角度范围，经过逐次迭代直到得到所需的精度。

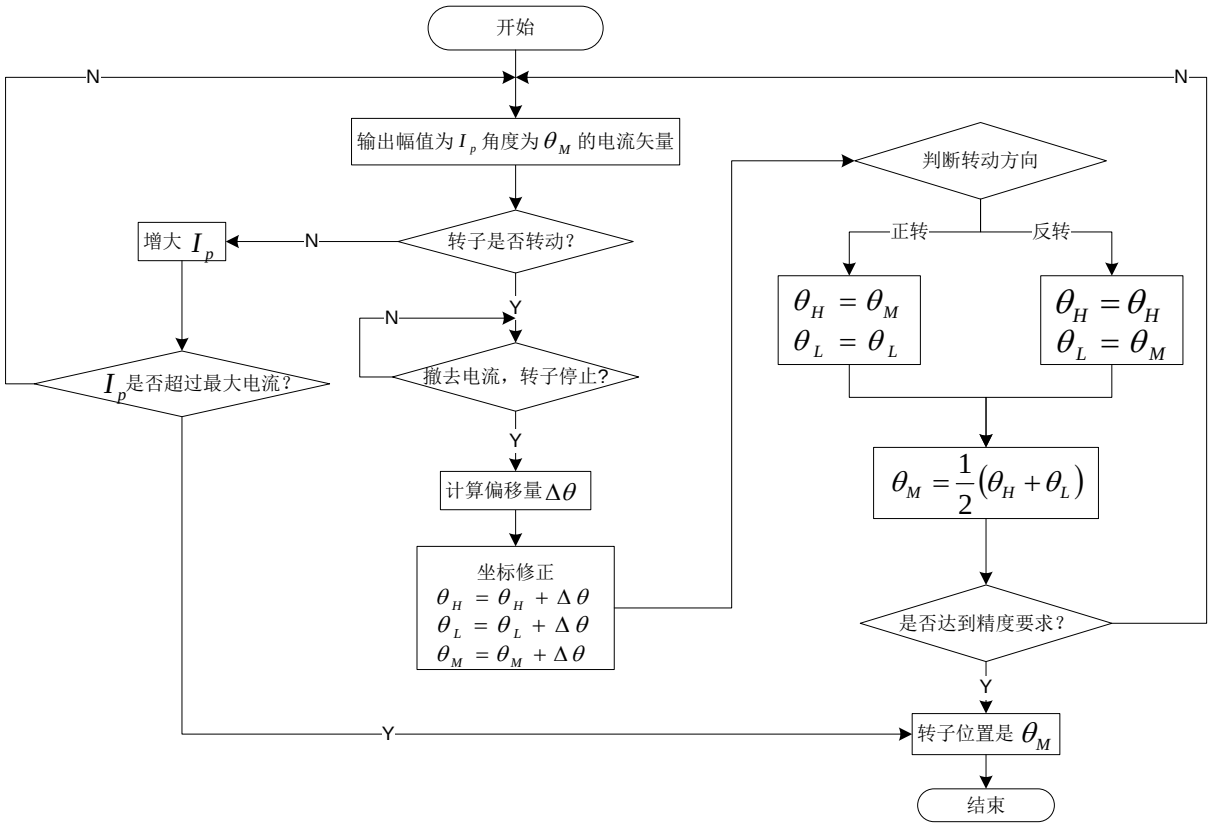

图 15 引入坐标修正的转子初始位置搜索算法

图 15中 I_p 是所施加的电流矢量幅值，若轴上负载较大，则在算法迭代过程中不断累加。

- $\Delta\theta$ 是每一次迭代过程中转子移动的角度。
- θ_H 是估计的转子位置角度范围上限，初始值是 $+180^o$。
- θ_L 是估计的转子位置角度范围下限，初始值是 -180^o。
- θ_M 是估计的转子位置角度范围的中心值，也即所施加电流矢量位置，初始值是 0^o。

算法中设置了两个退出迭代的条件：θ_H 与 θ_L 之间的距离已经达到了需求的精度，或者电流矢量幅值已超过了最大输出电流限幅，转子仍然没有转动。这两种情况下，转子实际位置非常接近 θ_M。

随着迭代次数增多，θ_H 和 θ_L 之间的角度范围越来越小，转子在位于 θ_M 的电流矢量的作用下，可能被拉出 θ_H 和 θ_L 组成的角度范围之外，从而造成后续迭代测试的结论错误。为了避免这种现象，每次迭代后，利用当前转子转过的位置偏移量 $\Delta\theta$ 修正角度范围 θ_H 和 θ_L，这样即使在位置检测过程中出现较大转子偏移，也不影响算法的正常运行和检测精度。详细算法如图 15所示。

2.4.3.2 实验数据

电机采用1000线的增量式光电编码器，TMRA0工作于quadrature count模式，其计数器值的变化范围如图 12所示。计数器值每变化1代表机械角度0.09°，或电角度0.18°（电机为2对极）。实验中先将电机转子置于电角度30°位置，TMRA0计数器值清零，然后运行搜索算法，实验结果如图 16所示。

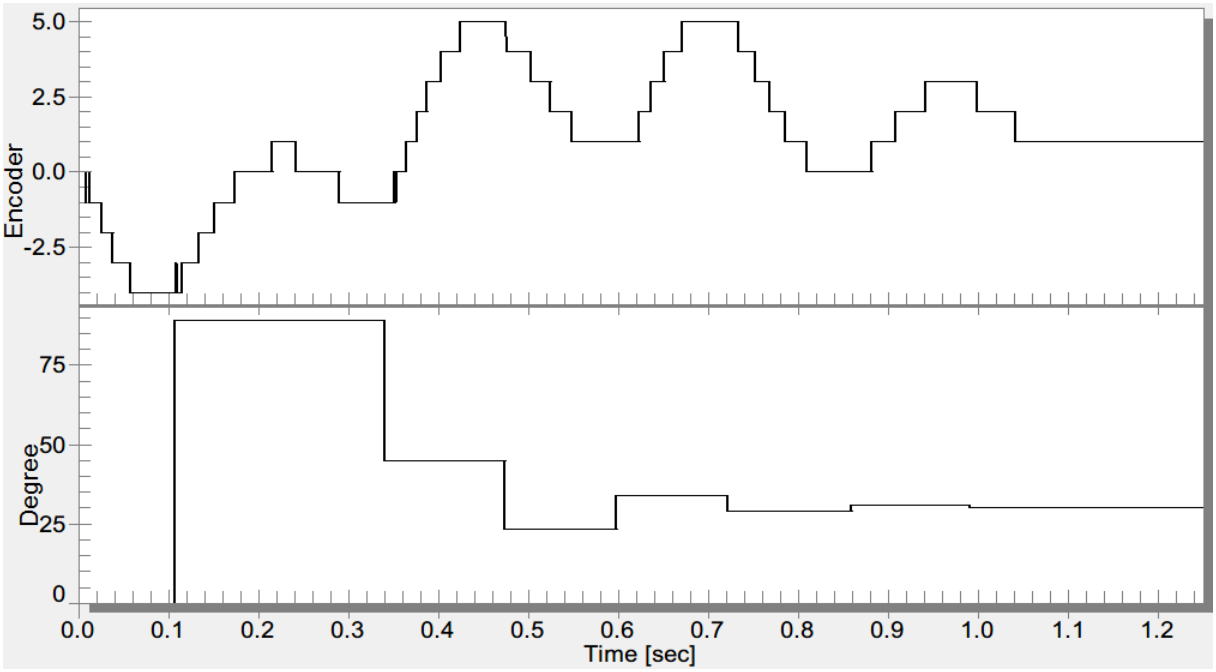

图 16 引入坐标修正的转子初始位置搜索算法的实验结果

图 16中上图是TMRA0计数器值变化曲线，纵坐标为原始的计数器值。下图是θ_M（电角度）的变化曲线，即输出的定子电流矢量的角度。上图中进行了8次搜索，整个搜索算法在1.2秒内完成，最终θ_M的值就是检测到的转子电角度位置。由于电机是2对极，将检测到的转子电角度右移一位得到转子的机械位置，并参照图 12，修改TMRA0计数器为相应的值。

2.4.4 测速原理

一般伺服控制器中，测速的方法有M法、T法、M/T法，下面逐一进行介绍。

- M法：在固定定时时间T_c（以秒为单位）内，统计编码器脉冲数M_1，通过等式 31可以计算得到角速度ω。C为常数，代表电机旋转机械360度编码器输出的脉冲数。由于C和T_c均为常数，角速度ω与M_1成正比，低速时时间T_c内M_1数值变小，量化误差变大，所以M法在低速时的测速精度低。

$$\omega = \frac{2\pi M_1}{CT_c}$$

等式 31

- T法：在编码器两个脉冲的间隔时间T_t之内，激活一个频率为f_0的高频脉冲并进行计数，计数值为M_2，角速度ω通过等式 32计算。与M法相反，由于T法中C和f_0为常数，角速度ω与M_2反比，高速时T_t减小，M_2数值随之变小，量化误差变大，所以T法在高速时的测速精度低。

$$\omega = \frac{2\pi}{CT_t} = \frac{2\pi f_0}{CM_2}$$

等式 32

- M/T法：综合M法和T法各自优势，在一个相对固定的时间间隔内，计数编码器脉冲数M_1，同时计数频率为f_0的高频脉冲数M_2从而得到这个时间间隔的精确长度，角速度ω通过等式 33计算。在M/T法中，C和f_0为常数。当高速时，M_1变大而M_2基本不变，相当于M法，满足高速的测速精度；当低速时，M_1变小而M_2变大，相当于T法，满足低速的测速精度。

$$\omega = \frac{2\pi M_1}{CT_t} = \frac{2\pi M_1 f_0}{CM_2}$$

等式 33

M/T 法可以通过 TMRA0 和 TMRA1 的捕捉功能实现。Timer 的 secondary 输入的上升和下降沿都可以用来触发捕捉（capture），将对应的计数器值捕捉到 CAPT 寄存器中。当捕捉发生后，SCTRL 寄存器中的 IEF 标志会置为 1，当此标志为 1 时无法再次触发捕捉。由图 10 所示，编码器的 B 相信号同时连接至 TMRA0 和 TMRA1 的 secondary 输入，TMRA0 的 primary 输入是编码器的 A 相信号，且工作于 quadrature count 模式，故 TMRA0 的计数器是对 A/B 信号的每个沿计数；TMRA1 的 primary 输入是系统时钟的 128 分频，即 100MHz/128=781.25KHz，这是一个相对高频的时钟信号。图 17 给出了 M/T 法的具体实现。

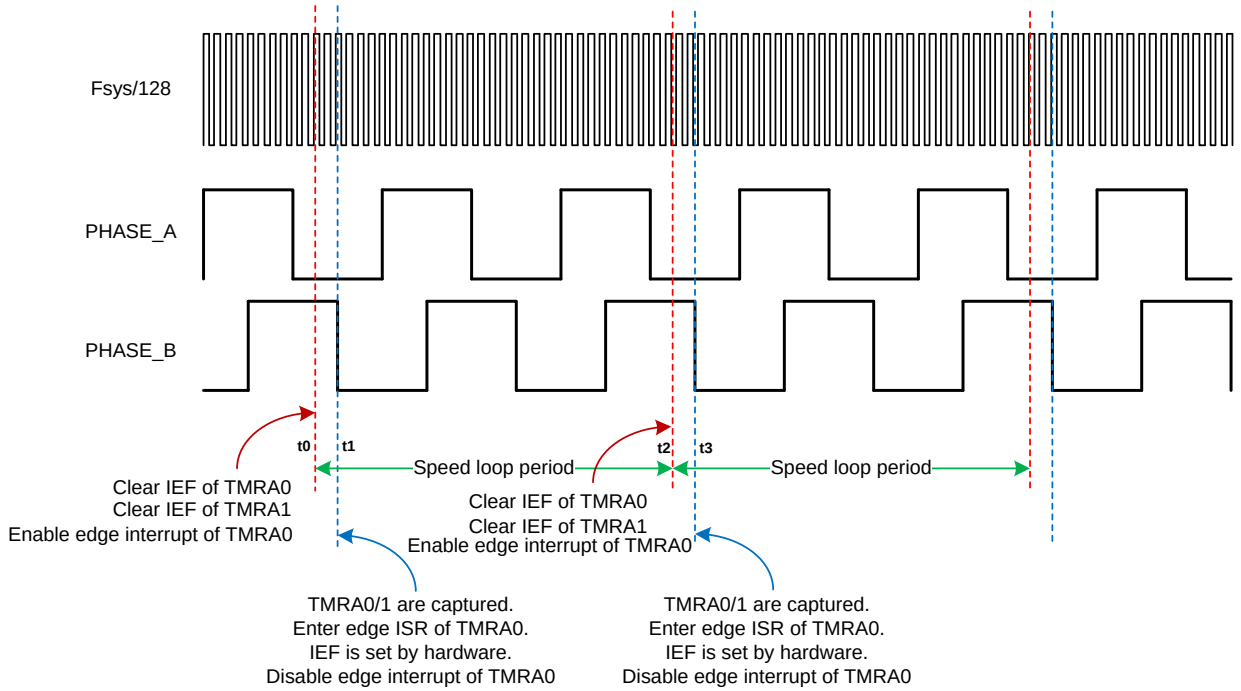

图 17 M/T法的实现

速度环每0.5 ms执行一次。t0时刻进入速度环，清除TMRA0/1的IEF标志以使能捕捉功能，同时使能TMRA0的secondary边沿中断。t1时刻编码器B相信号出现下降沿，于是TMRA0和TMRA1的计数器值捕捉至对应的CAPT寄存器并进入TMRA0的边沿中断，在中断函数中读取捕捉值，根据上次和本次的捕捉值计算转速，并禁止TMRA0的边沿中断。假设t1时刻TMRA0的捕捉值为M1(n-1)，TMRA1的捕捉值为M2(n-1)。t2时刻再次进入速度环，执行与t0时刻一样的操作，t3时刻编码器B相信号出现下降沿，得到TMRA0和TMRA1的捕捉值M1(n)和M2(n)，在TMRA0的边沿中断内计算转速，并禁止TMRA0边沿中断。编码器为1000线，转速的计算如下：

$$speed = \frac{2\pi * [M1(n) - M1(n-1)] * (f_{sys}/128)}{4000 * [M2(n) - M2(n-1)]} rad/s$$

2.4.5 极低速情况下M/T法的失真问题

极低速性能对一个伺服驱动的运动控制精度而言至关重要，目前考虑到成本和可靠性，伺服控制器的速度和位置检测传感器绝大多数采用增量式编码器，利用M/T法计算平均速度。

传统的M/T法结合了M法和T法各自优势，充分利用了增量式编码器的分辨率。当电机运行在中高转速时，在每个速度采样周期内能够获得足够的编码器脉冲数（即边沿数），满足测速精度。然而在极低速的情况下，每个速度采样周期内获得的编码器脉冲数极少，甚至不到一个脉冲，即编码器脉冲间隔时间超过了速度采样周期。图 18是极低速时M/T法示意图。

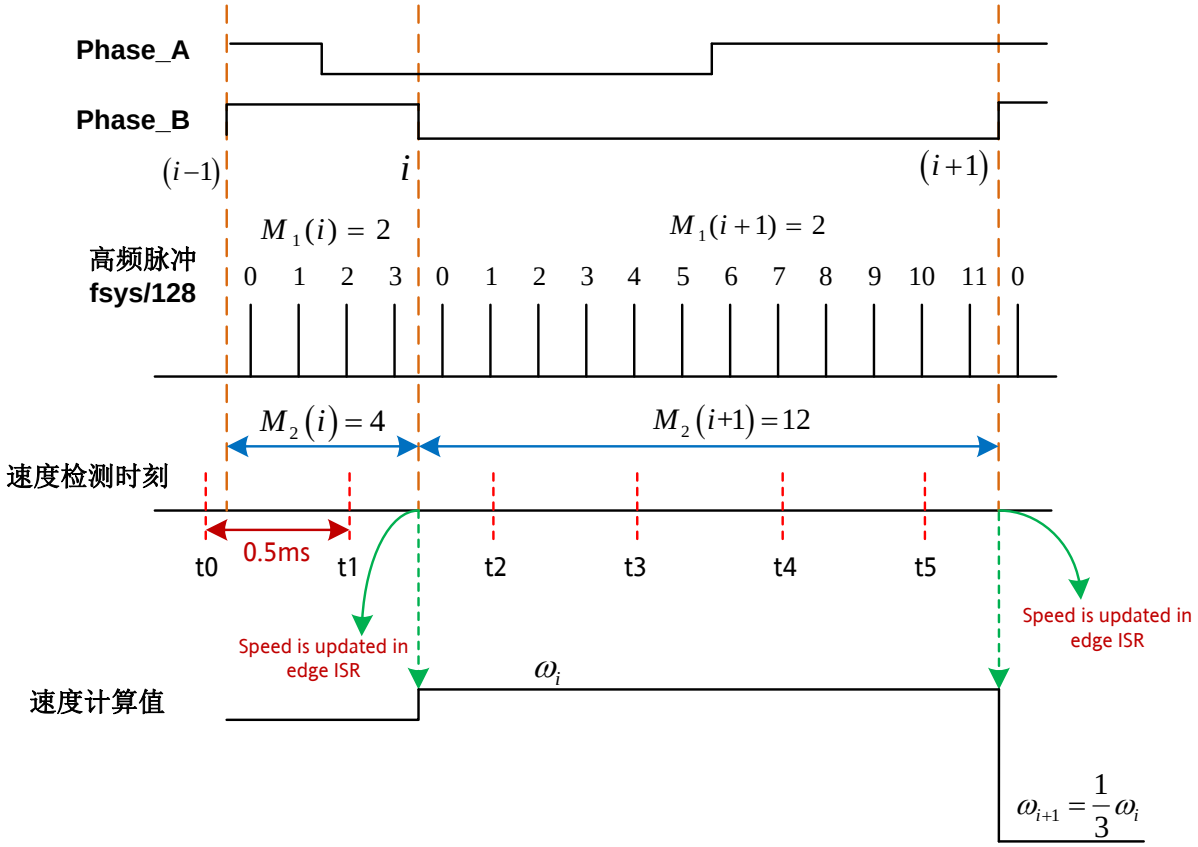

图 18 极低速时M/T法示意图

在图 18中，第一行是编码器A/B波形，ω_i 和 ω_{i+1} 是B相两个边沿间计算出的平均速度。第二行是用于计时的高频脉冲，即 $f_{sys}/128$。第三行给出了速度检测时刻，即每个速度环的入口处执行速度检测。$M_1(i)$ 和 $M_2(i)$ 与等式 33中的 M_1 和 M_2 对应。由图 18和等式 33可知，ω_{i+1} 小于 ω_i，编码器B相信号第i个边沿到第（i+1）个边沿的间隔时间超过了速度检测周期，t2、t3和t4时刻虽然使能了TMRA0的边沿捕捉和中断功能，但由于没有编码器脉冲，在第（i+1）个边沿到来之前，速度采样值一直保持 ω_i，然而实际的速度已经降低为 ω_i 到 ω_{i+1} 之间的某一数值。直到t5时刻之后，第（i+1）个脉冲触发捕捉和边沿中断才将速度更新为 ω_{i+1}。这段实际速度已经降低而速度检测仍保持前一次数值的时间，称为速度检测停滞时间。在速度检测停滞时间内，速度检测值偏离真实值，导致速度失控，这是一般伺服驱动在极低速运行时稳态性能差的重要原因。遇到这种情况，在实际操作中一般通过减小速度控制器增益来减小振荡。在极低速情况下，编码器线数越高，伺服控制器的速度环带宽可以设置越大，速度的动态响应可以做得更好，但是这样成本会提高。所以如何在不提高编码器成本的前提下，减小或者消除速度检测停滞时间，从而提高伺服控制器极低速性能是一个难点。

2.4.6 增强型M/T测速法

为了最小化速度检测停滞时间，本方案使用一种增强型M/T测速法，该方法在每个速度检测时刻都判断转速是否变慢了，从而通过补偿的方式来更新转速。具体原理如图 19所示。

由图 10可见，编码器的B相信号同时与TMRA3的secondary输入连接，TMRA3工作在count模式，对 f_{sys} /128 计数。使能TMRA3的捕捉功能，并在此基础上使能“Reload on capture”功能，这样的话每个B相信号的边沿在触发捕捉的同时还将TMRA3的计数器清零了，如图 19所示。

每次进入速度环计算转速时，首先读取TMRA3的当前计数器值，并与上次捕捉计算得到的 M_2 值进行比较。图 19中，在t1和t2时刻之间触发了一次B相边沿中断，从而更新了 M_2 值： $M_2(i)=4$，并按照等式 33计算得到了转速 ω_i，同时TMRA3的计数器自动清零。

1. 在t2时刻读取TMRA3的计数器值，为2，$2<M_2(i)$，不补偿转速，保持前一拍计算的转速值，$\overline{\omega}_i=\omega_i$。
2. 在t3时刻读取TMRA3的计数器值，为5，$5>M_2(i)$，补偿转速，更新转速值为 $\frac{4}{5}\overline{\omega}_i$。
3. 在t4时刻读取TMRA3的计数器值，为8，$8>M_2(i)$，补偿转速，更新转速值为 $\frac{4}{8}\overline{\omega}_i$。
4. 在t5时刻读取TMRA3的计数器值，为11，$11>M_2(i)$，补偿转速，更新转速值为 $\frac{4}{11}\overline{\omega}_i$。
5. 紧接着B相信号出现上升沿，触发捕捉和边沿中断并更新 $M_2(i+1)=12$，并在边沿中断ISR中按照等式 33计算出转速为 $\frac{1}{3}\omega_i$，同时TMRA3计数器值自动清零。

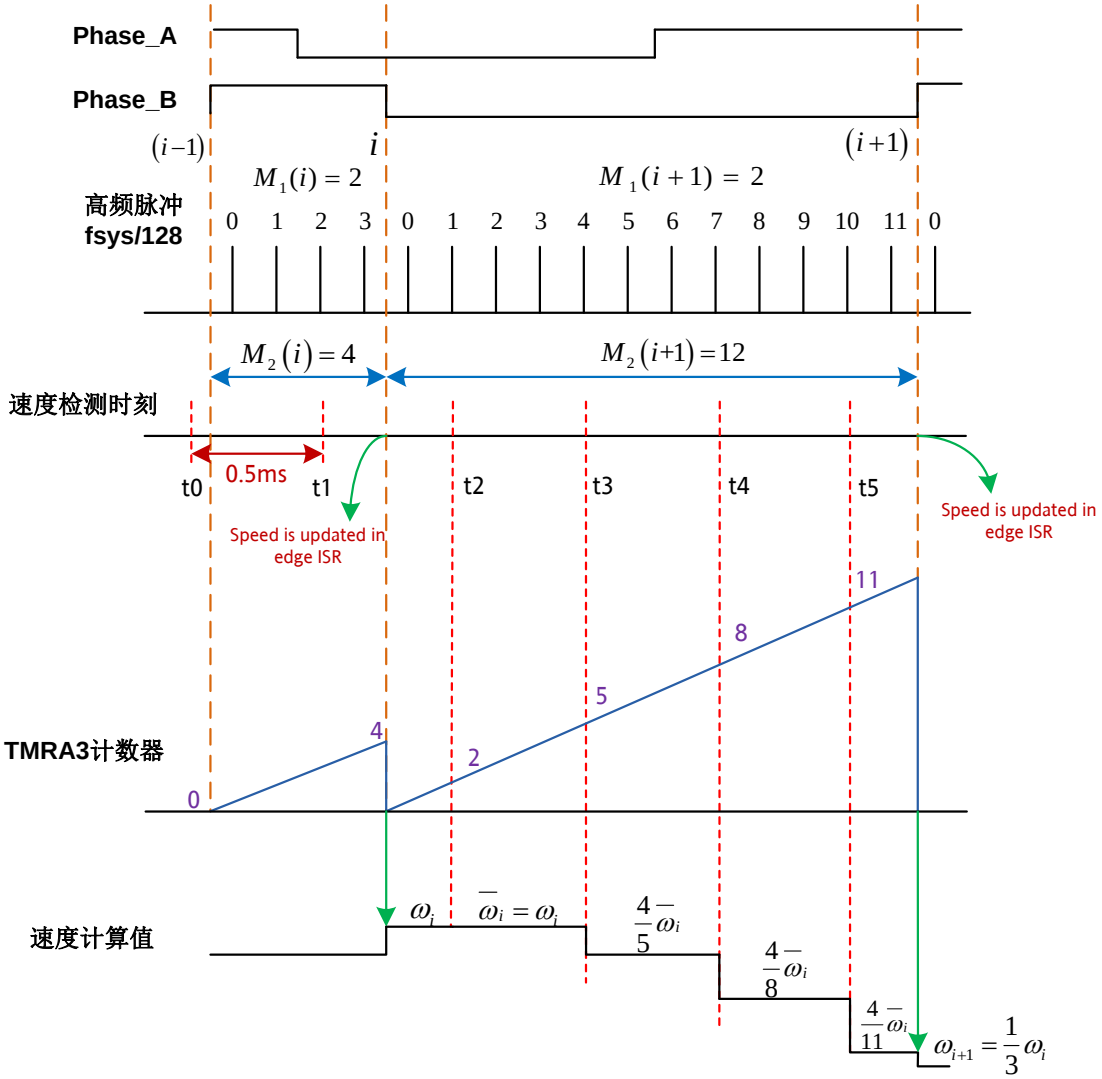

图 19 增强型M/T法原理图

可以看到t2~t5时刻对转速进行了补偿，芯片中的Quad Timer模块可以方便的实现这一功能。

2.5 速度PI控制器设计

2.5.1 速度PI控制器的退饱和超调现象

在实际的伺服控制器中，考虑到数字运算存在溢出问题和硬件输出电流的能力，需要对速度PI控制器的输出值实行限幅处理。在数字控制算法中，要对速度PI控制器输出值u进行限幅，首先确定电流最大值I_{max}，然后进行限幅判断，如果u的绝对值大于I_{max}，则将I_{max}代替u的值作为输出，否则u值不变。表达式如下：

$$\begin{cases} u = I_{\max} & if\ (u > I_{\max}) \\ u = u & if\ (-I_{\max} \le u \le I_{\max}) \\ u = -I_{\max} & f\ (u < -I_{\max}) \end{cases}$$

等式 34

按照等式 34的限幅方法，当速度PI控制器进入饱和状态后，积分项大于限幅后的输出项，那么在退饱和时要先将积分项超过限幅值的那部分消除后，才能离开限幅状态进入PI控制状态。这样速度PI控制器的输出在退饱和时会长时间保持在限幅值，从而引起速度超调，进而影响伺服控制器动态性能。

2.5.2 抗积分饱和的速度PI控制器原理

为了解决速度PI控制器的退饱和超调问题，本方案利用一种基于抗积分饱和的速度PI控制器。所谓抗积分饱和，就是将限幅前的PI控制器输出值u_{presat}与限幅后的输出值u_{out}之间的差值作为积分项的一部分进行计算，达到快速退饱和的目的，抗积分饱和的速度PI控制器原理如图 20所示：

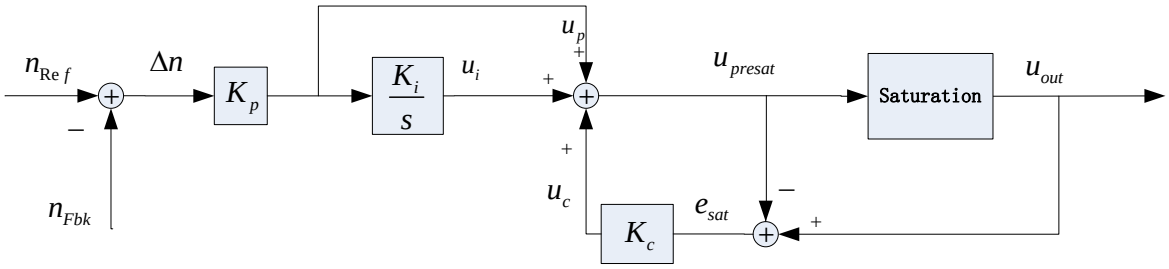

图 20 抗积分饱和的速度PI控制器原理图

在图 20中n_{Ref}是转速参考值，n_{Fbk}是转速反馈值，Δn是转速偏差值，u_p是比例项，u_i是积分项，u_c是抗积分饱和项，u_{presat}是限幅前速度PI控制器输出值，u_{out}是限幅后输出值，e_{sat}是限幅前的PI控制器输出值u_{presat}与限幅后的输出值u_{out}之间的差值，K_c是抗积分饱和系数。

在伺服控制器的数字控制算法中，使用向后差分方法设计抗积分饱和速度PI控制器，第k时刻的比例项$u_p(k)$、积分项$u_i(k)$、抗积分饱和项$u_c(k)$分别为：

$$\begin{cases} u_p(k) = K_p \cdot e(k) \\ u_i(k) = u_i(k-1) + K_p \cdot K_i \cdot T_s \cdot e(k) \\ u_c(k) = K_c \left[u_{out}(k-1) - u_{presat}(k-1) \right] \end{cases}$$

等式 35

其中$e(k)$是第k时刻的速度偏差，T_s是速度采样时间。

第k时刻限幅前速度PI控制器输出值$u_{presat}(k)$为：

$$u_{presat}(k)=u_p(k)+u_i(k)+u_c(k)$$

等式 36

第k时刻限幅后的输出值$u_{out}(k)$为：

$$\begin{cases} u_{out}(k)=I_{\max} & if\left[u_{presat}(k)>I_{\max}\right] \\ u_{out}(k)=u_{presat}(k) & if\left[-I_{\max}\le u_{presat}(k)\le I_{\max}\right] \\ u_{out}(k)=-I_{\max} & if\left[u_{presat}(k)<-I_{\max}\right] \end{cases}$$

等式 37

2.5.3 抗积分饱和的速度PI控制器的实验效果

实验时，伺服在速度闭环模式下空载运行，速度给定信号是幅值500 rpm，频率为20 Hz的正弦信号，观察速度和q轴电流的动态响应。为了凸显抗积分饱和的效果，将q轴电流输出限幅设置为1 A，同时保持速度PI控制器的比例项系数和积分项系数不变，分别对抗积分饱和速度PI控制器和一般速度PI控制器进行对比实验，结果分别如图 21和图 22所示。

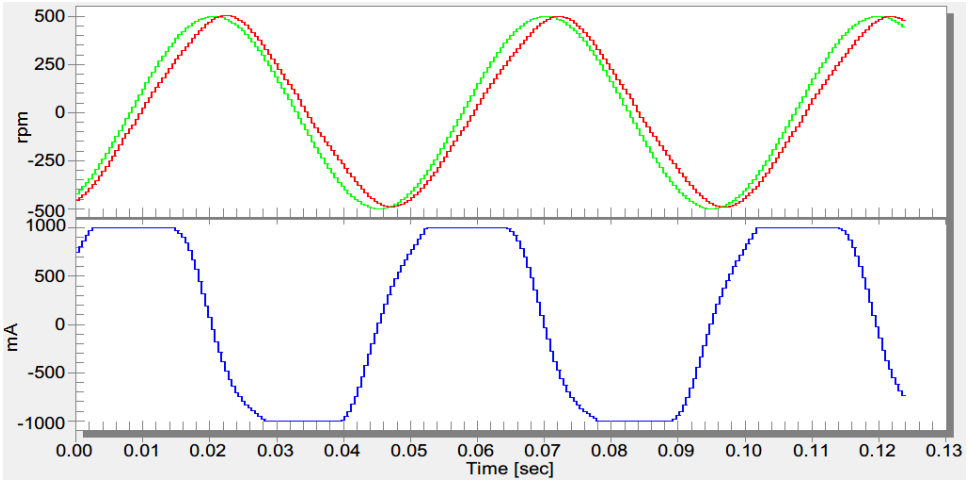

图 21 抗积分饱和速度PI控制器实验效果

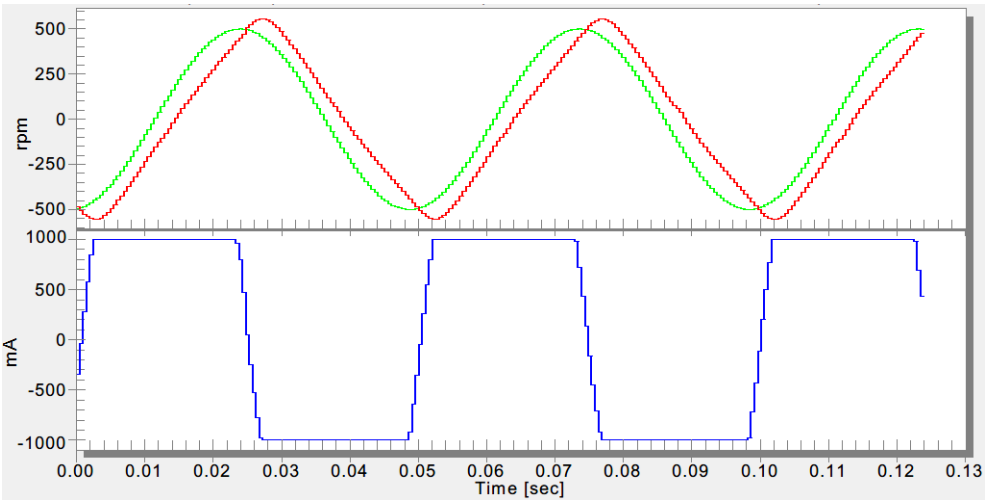

图 22 一般速度PI控制器实验效果

在图 21和图 22中，绿色是正弦速度给定，红色是速度反馈，蓝色是速度控制器输出，即q轴电流给定。

分析图 21抗积分饱和速度PI控制器实验效果可知，在加速过程中，当速度偏差（速度给定与速度反馈之间的差值）不断加大，q轴电流给定达到限幅值并保持。一旦速度偏差减小，由于抗积分饱和项的存在，速度控制器能够迅速退饱和，消除退饱和超调。速度反馈能够较好地跟踪速度正弦给定信号，几乎无超调，相位滞后较小。

分析图 22一般速度PI控制器实验效果可知，当速度偏差开始减小时，速度控制器仍处于饱和状态，直到速度偏差改变符号时才退出饱和状态，出现退饱和超调。速度反馈曲线跟踪速度正弦给定信号效果较差，有明显超调和相位延迟。

有上述分析可知，抗积分饱和速度PI控制器可以有效消除速度控制的退饱和超调，显著提高伺服控制器速度闭环动态响应性能。MC56F84789所具备的DSP指令能够方便有效的实现此控制器。

2.6 基于二阶数字滤波器的位移路径优化

2.6.1 位移路径分析

伺服电机的位移路径由位置给定信号决定。由牛顿运动定律可知，位移路径一旦确定，每一时刻的速度和加速度就随之确定。理想的伺服电机运动过程中，应极力避免速度和加速度的突变，否则会产生过大的电流和电压变化，对伺服驱动的稳定造成不利影响，所以需要对位移路径给定进行优化。

本方案利用二阶滤波器将斜坡型的位移曲线变为平滑的S型曲线。对二阶滤波器的位移路径优化效果进行仿真，图 23是二阶滤波器路径优化效果的位移曲线，相对应的速度曲线。

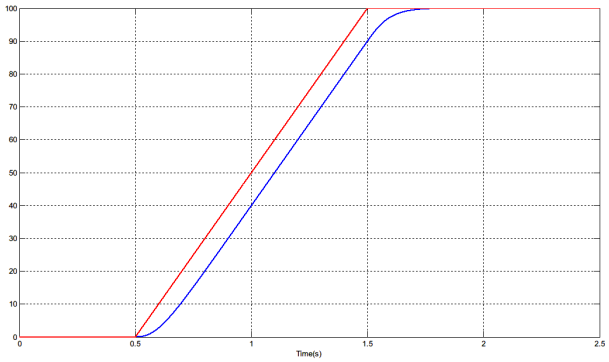

图 23 二阶滤波器路径优化效果的位移曲线

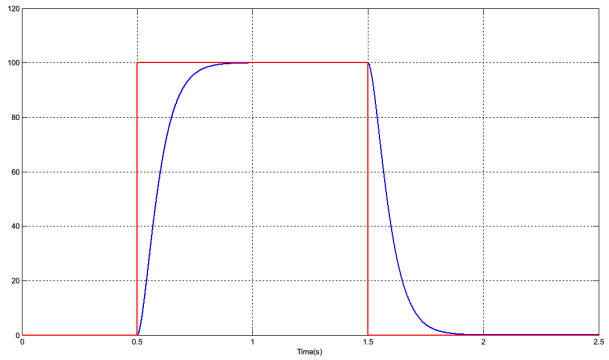

图 24 对应的速度曲线

图 23中红色的是斜坡型位移路径曲线，蓝色的是二阶滤波后的S型位移路径曲线，可见二阶滤波器可以使位移路径的起始点和终止点变得平滑。图 24中红色的是斜坡型位移路径所对应的速度曲线，蓝色的是二阶滤波后的S型位移路径所对应的速度曲线，可见斜坡型位移路径给定导致速度给定在起始点和终止点处发生阶跃变化，而S型位移路径给定使得速度给定在起始点和终止点处变化平缓。

理论上，二阶滤波器可以有效优化位移路径，提高伺服控制器的位置控制稳定性。

2.6.2 二阶数字滤波器的设计

二阶滤波器的原型是一个阻尼比等于1二阶系统（即临界阻尼的情况）：

$$G(s)=\frac{\omega_n^2}{s^2+2\omega_n\cdot s+\omega_n^2}$$

等式 38

其中 ω_n 是无阻尼自振角频率，单位 rad/s 。

在数字信号处理中，积分环节是最容易程序实现的环节，所以将等式 38转换为以积分环节为基础的形式：

$$G_s(s)=\frac{\frac{\omega_n^2}{s^2}}{1+\frac{2\omega_n}{s}+\frac{\omega_n^2}{s^2}}$$

等式 39

按照等式 39设计二阶滤波器结构图：

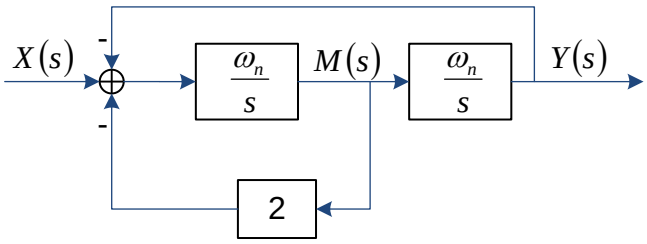

图 25 二阶数字滤波器结构图

图 25中 $M(s)$ 是中间变量。

利用后项差分法，得到二阶数字滤波器的差分方程：

$$\begin{cases} m(k)=m(k-1)+\omega_n\cdot T_s\cdot[x(k)-2m(k-1)-y(k-1)] \\ y(k)=y(k-1)+\omega_n\cdot T_s\cdot m(k) \end{cases}$$

等式 40

其中 T_s 是二阶数字滤波器的采样时间。

2.7 位置控制

2.7.1 位置环具体实现

图 7中示出的“Position control”部分有所简化，具体实现如图 26所示。

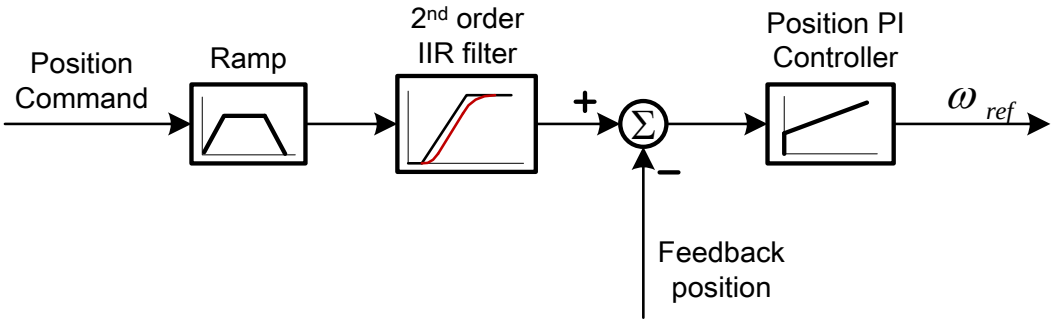

图 26 位置环功能模块结构图

2.7.2 位置信号的数据格式

按照2.4.3.1节描述的转子初始位置检测算法检测出转子的初始电角度位置之后，TMRA0的计数器值修改为对应的数值，此后的运行中，TMRA0的计数器值就可以反映转子的真实电角度位置和机械位置了。反馈的位置信号是相对于检测到的转子初始机械位置转子转过的圈数，以P.Q格式表示，其中P代表机械圈数，Q代表小数部分。比如3.64代表转子相对于其初始机械位置向正方向转过了3.64圈；-5.79代表转子相对于其初始机械位置向反方向转过了5.79圈。

初始位置检测之后，得到转子初始电角度位置，将其转换为TMRA0的计数器值EncCntInit，并修改TMRA0计数器。每次进入位置环时读取当前TMRA0的计数器值EncCnt，并按照2.4.2.2节所述计算转子转过的机械圈数RealZCnt，然后计算反馈的位置:

$$\text{Feedback position} = \text{RealZCnt} + (\text{EncCnt} - \text{EncCntInit})/(4 * \textit{EncoderLines})$$

等式 41

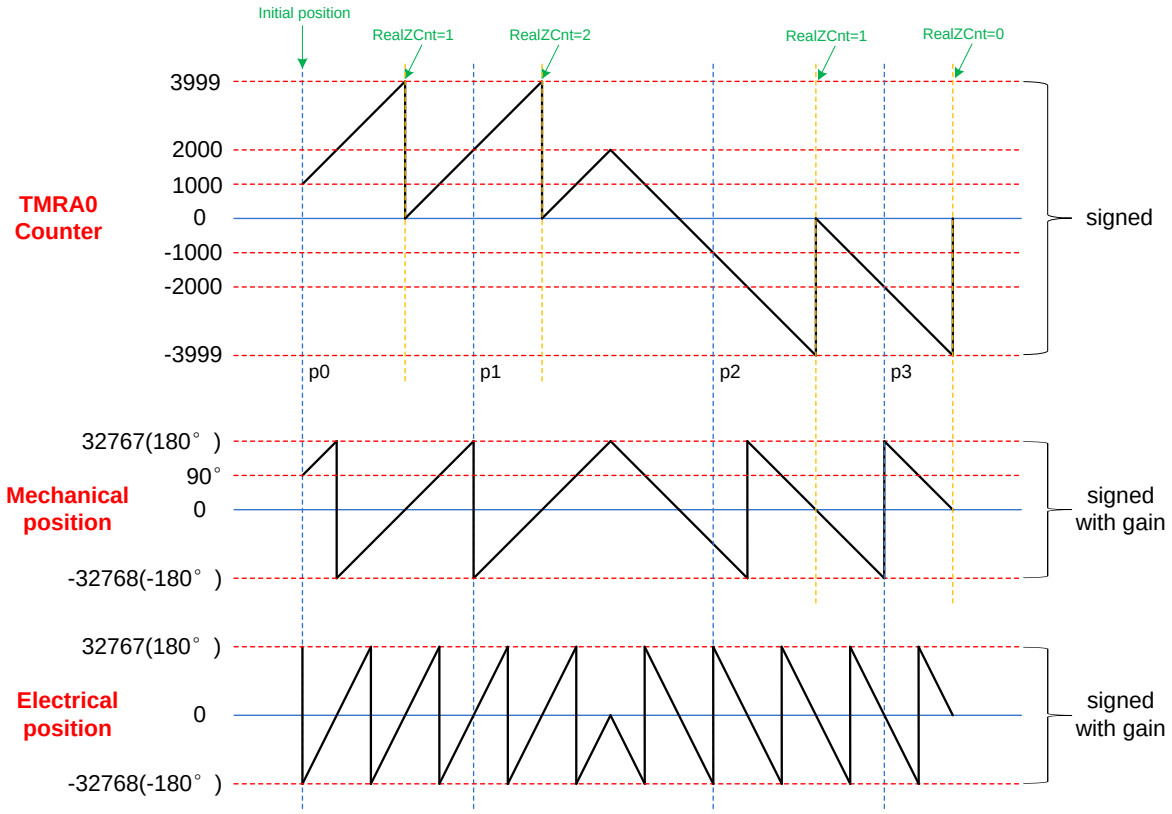

图 27 反馈位置检测示意图

图 27给出了反馈位置计算的示意图。图中最上的波形为TMRA0的计数器值，中间为转子的机械位置，最下为转子电角度位置，由于电机为2对极，故电角度位置是机械位置的两倍。初始位置检测中得到转子处于电角度180°，即机械角度90°，换算为计数器值为1000，如图 27中Initial position所示，此后电机开始正向旋转。编码器为1000线。

- 在p1时刻，TMRA0计数器值为2000，由等式 41计算得反馈的转子位置为 $RealZCnt + (EncCnt - EncCntInit)/(4 * EncoderLines) = 1 + (2000 - 1000)/4000 = 1.25$，即此时转子处于相对其初始位置1.25圈的位置。
- 在p2时刻，TMRA0计数器值为-1000，由等式 41计算得反馈的转子位置为 $RealZCnt + (EncCnt - EncCntInit)/(4 * EncoderLines) = 2 + (-1000 - 1000)/4000 = 1.5$，即此时转子处于相对其初始位置1.5圈的位置。
- 在p3时刻，TMRA0计数器值为-2000，由等式 41计算得反馈的转子位置为 $RealZCnt + (EncCnt - EncCntInit)/(4 * EncoderLines) = 1 + (-2000 - 1000)/4000 = 0.25$，即此时转子处于相对其初始位置0.25圈的位置。

2.7.3 位置控制器的设计

由于位置环是伺服控制器的最外环，位置环的设计应当以速度环的设计为基础。根据速度环的设计中得到的速度环单位负反馈开环传递函数式可以得到速度闭环传递函数，这是一个高阶结构，如果依照此模型作为控制对象来设计位置环的话，则得到的位置控制系统就是一个高阶控制系统，相应的位置控制器的设计会变得复杂。为了解决这个问题，本方案采用等效模型的方法，来处理位置控制对象。

根据电机机械运动方程式和速度与位移之间的物理关系，可知电机转速的变化快于位置的变化，速度环的截止频率应大于位置环的截止频率，所以在设计位置控制系统时可以将速度环控制对象近似等效为一阶惯性环节：

$$W_N = \frac{1}{T_N s + 1}$$

等式 42

其中 T_N 是速度闭环等效惯性环节的时间常数。电机空载情况下，以额定电磁转矩将电机加速到额定速度，根据电机机械运动方程可以推导出 T_N：

$$T_N = \frac{N_{rd} \frac{\pi}{30} J_0}{T_{rd}}$$

等式 43

其中 N_{rd} 是电机额定转速，T_{rd} 是电机额定转矩，J_0 是电机本体转动惯量。$\pi / 30$ 是将分钟每转的速度单位转换为角速度单位的系数。

在伺服应用中，应极力避免位置跟踪出现超调和振荡，所以位置控制器仅包括比例项，将位置环整定为典型Ⅰ型系统。位置环数学模型结构图如下：

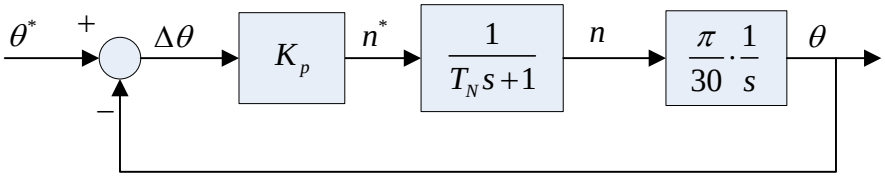

图 28 位置环数学模型结构图

由图 28可知位置环开环传递函数：

$$W_{opp} = \frac{K_p \frac{\pi}{30}}{s(T_N s + 1)}$$

等式 44

按照针对典型Ⅰ型系统的“最佳整定法”折中选取 $K_p \cdot (\pi / 30) T_N = 0.5$，则

$$K_p = \frac{15}{\pi \cdot T_N}$$

等式 45

由等式 45可知位置控制器比例项系数 K_p 与速度闭环等效惯性环节的时间常数 T_N 有关。但是按“最佳整定法”设计出的比例项系数 K_p 只能作为实际调试时的参考值，应用中往往以此为参考并根据性能需求调整比例项系数 K_p。

3 硬件设计

3.1 电机控制系统概述

本应用中，电机控制系统由两部分构成，电机驱动板和Freescale MC56F84789 DSC控制核心板，可以同时完成两个三相电机的控制。驱动板上电机1和电机2的驱动电路完全一致，使用分立IGBT实现逆变桥。在驱动电路中在三个下桥臂中串入电流采样电阻，再经过差分放大电路，连接到核心板上MC56F84789 的ADC通道引脚实现电流采样。驱动板将24V直流电源转换为15 V、5 V、3.3 V，供其他模块使用。

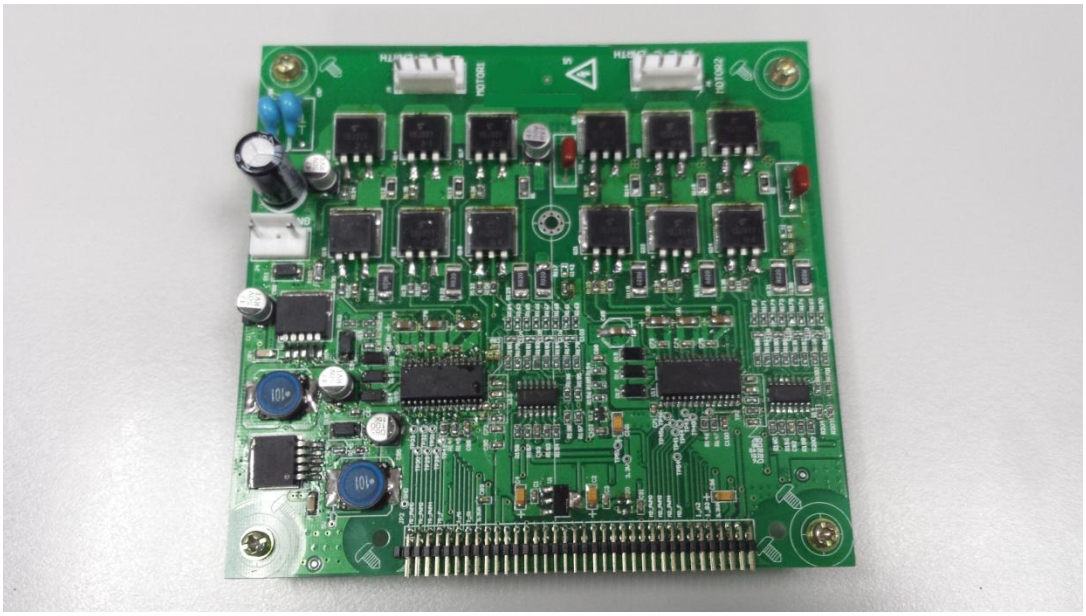

图 29 驱动板实物图

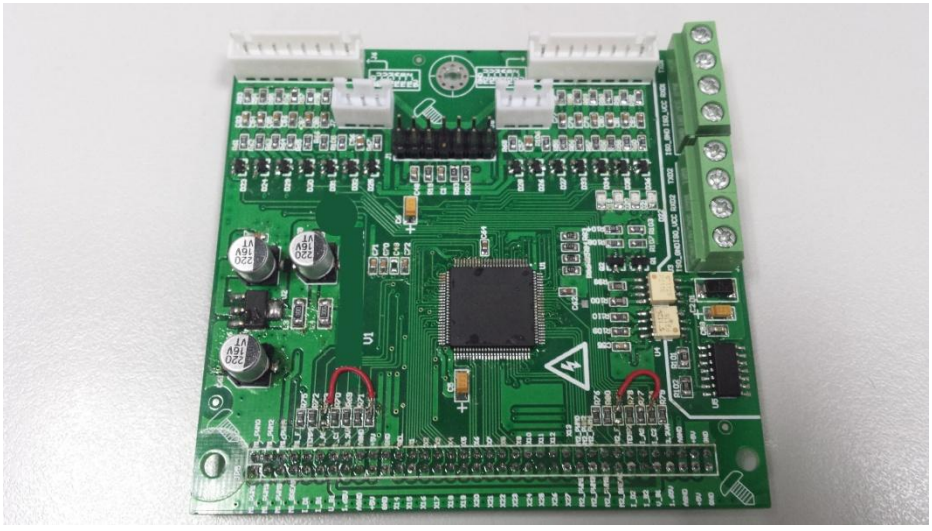

图 30 核心板实物图

3.2 电机控制系统规格参数

电机驱动板：

- 直流24V输入
- 两路三相电机驱动电路，每路均包括以下部分：
  - 由分立IGBT组成的三相逆变桥
  - 具有过流、欠压保护的预驱动芯片
  - 电机三相电流检测
  - 电机直流母线电流检测
- 直流母线电压检测
- 1.65V基准电压输出，供核心板使用
- 板载DC-DC电源，输出+15 V，+5 V，+3.3 V和+3.3 VA，供预驱动芯片和电流放大电路使用
- 板级通讯接口,与控制核心板连接

DSC控制核心板：

- LQFP100封装的DSC MC56F84789
- 与电机驱动板连接用的板级通讯接口
- JTAG 调试接口
- 板载DC-DC电源，输出 +3.3 V，+3.3 VA
- 隔离SCI通讯
- 编码器/Hall接口

4 软件设计

本应用的软件开发环境使用飞思卡尔的集成开发工具CodeWarrior10.5，整个软件是用C语言实现的，并调用了飞思卡尔的嵌入式软件库（FSLESL）。该软件库包括各种数学运算库、电机控制算法库、观测器库、信号滤波器库等。这些库和对应的文档可以从飞思卡尔网站freescale.com/fslesl直接下载。想获得更多有关如何在CodeWarrior的工程中使用这些库的信息，请参考文档*Inclusion of DSC Freescale Embedded Software Libraries in CodeWarrior 10.2* (AN4586)。

本章主要描述系统的软件设计。首先讨论如何用归一化的定点小数格式来表示控制器中的各个物理量，然后介绍系统中电机的控制时序、ADC同步问题，最后简单介绍控制软件的实现。

4.1 定点小数表示

应用程序中涉及到的电压、电流、角度和速度等物理量都是用归一化的定点小数表示的，即Q数据格式。对于一个N比特位的补码数据D，可以将其看做Qx.y格式的小数，其中y=N-x。x表示整数的位数，y表示小数的位数。Qx.y代表的数值为 $D/2^y$，D为该补码数据本身的值。显然，当y=0时，Qx.y代表的值就是D，不再有小数部分，此时，N比特位的补码数据D的数值范围为：

$$-2^{[N-1]} \le D \le +2^{[N-1]} - 1$$

当x=1时，定点小数表示形式为Q1.[N-1]。其中整数位只有1位，即符号位。剩余N-1位是小数位。它表示的有符号小数值（SF）的范围为：

$$-1.0 \le SF \le +1.0 - 2^{-[N-1]}$$

对于字型和长字型的变量，其可以表示的最小负数为-1，对应的数据分别为0x8000和0x80000000。字型变量可以表示的最大正数是$1.0 - 2^{-15}$，对应的数据为0x7FFF；长字型变量可以表示的最大正数是$1.0 - 2^{-31}$，对应的数据为0x7FFFFFFF。

飞思卡尔的数字信号控制器的CPU具有直接支持Q1.15/Q1.31格式数运算的指令。为了充分利用变量的精度以及方便运算，程序中将所有物理量都归一化为Q1.15或Q1.31格式的小数。

4.2 物理量的归一化表示

物理量的实际值和归一化的小数表示之间满足下面等式：

$$\text{小数值} = \frac{\text{实际值}}{\text{实际量化范围}}$$

其中：

- 小数值=物理量的归一化小数表示
- 实际值=用物理单位表示的实际物理量的值
- 实际量化范围=系统定义的用物理单位表示的物理量的最大值

下面几个小节分别举例介绍如何用归一化的定点小数来表示电压、电流和角度。

4.2.1 电压的归一化表示

母线电压值一般是通过ADC采样分压电阻上的电压获得的。所以被采样电压的最大值和ADC输入电压的范围成正比。在本应用的分压电阻网络的作用下，当ADC输入为3.3 V（AD输入所能识别的最大值）时，对应的母线电压是36.3 V。很明显，若母线电压超过36.3 V，那么ADC也无法识别其真实电压值。因此用36.3 V作为母线电压归一化的基底，当母线实际电压为24 V时，有：

电压最大值: $V_{max} = 36.3\ V$

被采样的电压: $V_{measured} = 24\ V$

$$\text{voltage\_norm} = \frac{V_{measured}}{V_{max}} = \frac{24V}{36.3V} = 0.661157$$

等式 46

将这个小数化为Q1.15格式：

$$\text{voltage\_variable} = \text{voltage\_norm} \cdot 2^{15} = 0.661157 \cdot 2^{15} = 21665$$

等式 47

程序中用voltage_variable变量表示母线电压。

上面的归一化过程可以用下面的图 31表示。

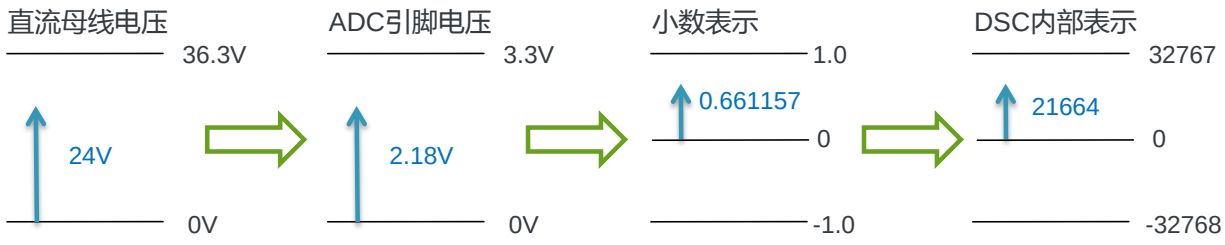

图 31 母线电压的测量

4.2.2 电流的归一化表示

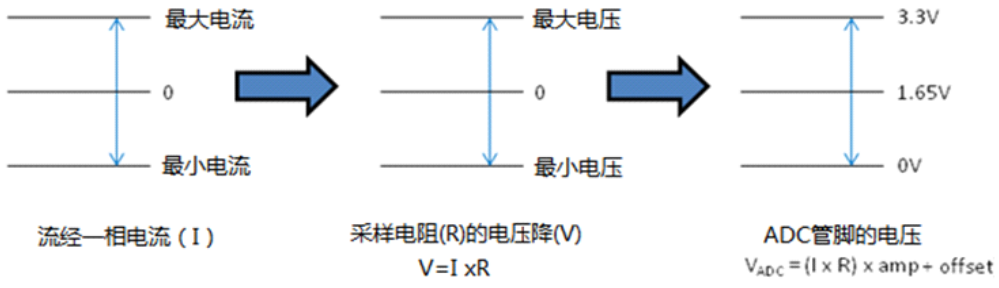

图 32 相电流测量

电流采样电阻的压降经过运算放大器输出到ADC进行采样，获得对应的电流值。如果电流的方向是双向的，则运放的输出信号还要有个偏置量。通过硬件电路设计，将正负电流的检测范围设置为一样，则偏置量的大小等于ADC输入范围的一半。这种情况下，可检测电流的最大值和ADC输入最大电压的一半值成正比。如图 32所示。电流变量的小数表示过程和电压变量类似。

4.2.3 角度的归一化表示

转子位置的角度也是用Q1.15格式表示的，其范围为[-1, 1)，对应的角度范围是[−π, π)。例如用Q1.15格式的字表示−π和π分别为0x8000和0x7FFF。

4.3 控制时序的设计

本参考设计基于单颗控制器芯片以矢量控制的方式同时驱动两个带增量式光电编码器的三相永磁同步电机。这两个控制对象要用一颗数字信号控制器芯片来同时控制，彼此间的时序相互关联，因此整个系统的控制需要精准的时序安排。

4.3.1 PWM模块的设置

本参考设计所用的数字信号控制器MC56F84789运行的时钟频率是100 MHz。控制两个电机的PWM模块的运行频率都是16 KHz。为了分时控制两个电机的电流环和速度环，两个电机的PWM波形相差180度相位，如图 34所示。图中分别给出了电机1和电机2的A相桥臂的上下管PWM驱动波形。

MC56F84789中有两个eFlexPWM模块：PWMA和PWMB。关于eFlexPWM模块的详细描述，请参考MC56F847xx Reference Manual。PWMA模块中子模块SM0~SM2控制电机1，PWMB模块中子模块SM0~SM2控制电机2。为了使两组PWM波形产生180度的相位差，PWMA中SM1和SM2的同步信号使用Master sync信号，即SM1和SM2的计数器与SM0同步，三个计数器在同一时刻的值完全相同。PWMB中SM0~SM2使用来自EXT_SYNC的同步信号，以PWMA中SM0子模块的计数器与其VAL0匹配时产生的trigger信号通过内部模块互连单元XBAR连接到PWMB的EXT_SYNC端。由于该trigger信号是在PWMA波形的中心时刻产生，故PWMA和PWMB有180度相位延迟。如图 33所示。

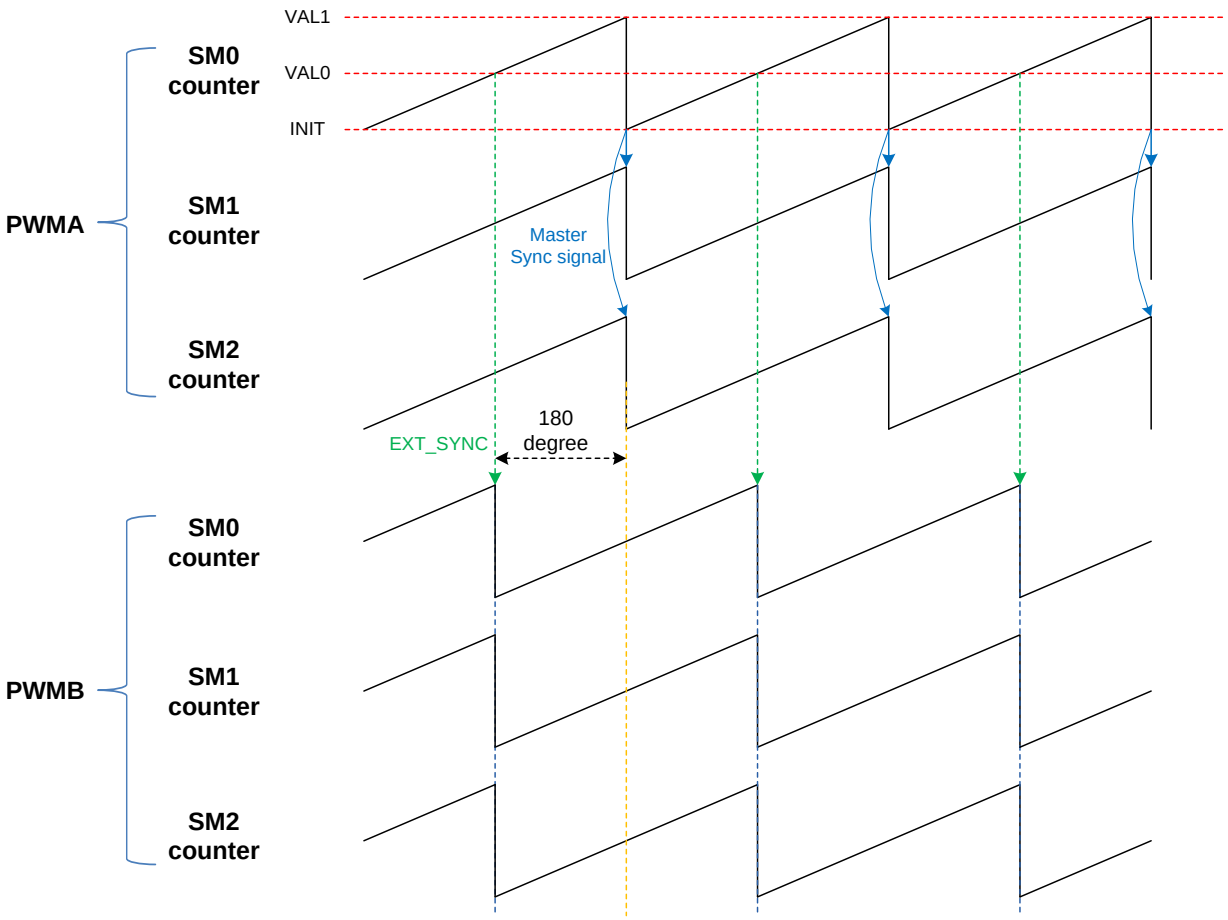

图 33 PWMA与PWMB相位互差180度的产生

4.3.2 PWM与ADC之间的同步关系

PWMA、PWMB模块产生12bit ADC的触发信号，同步关系如图 34所示。

ADC触发信号：

- PWMA模块中子模块SM1的计数器与其VAL4寄存器比较匹配后产生一个ADC触发信号；
- PWMB模块中子模块SM0的计数器与其VAL4寄存器比较匹配后产生一个ADC触发信号；

以上两个触发信号通过XBAR模块“或(OR)”在一起送给ADC的触发输入端。

ADC工作在“触发并行转换模式(Triggered parallel mode)”，当有触发信号到来时，预先设定好的转换序列将在ADC时钟的作用下自动顺序转换完成并最终触发ADC中断。该ADC模块中还有“扫描控制(Scan Control)”功能：当ADC被触发一次扫描序列后，可以指定在任意sample处暂停，等待下一个触发信号到来以继续该序列的转换，而不是触发一次就把序列中所有sample转换完成。有关12bit ADC的具体描述，请参考MC56F847xx Reference Manual。本应用中，ADC的转换序列设置如图 35所示。

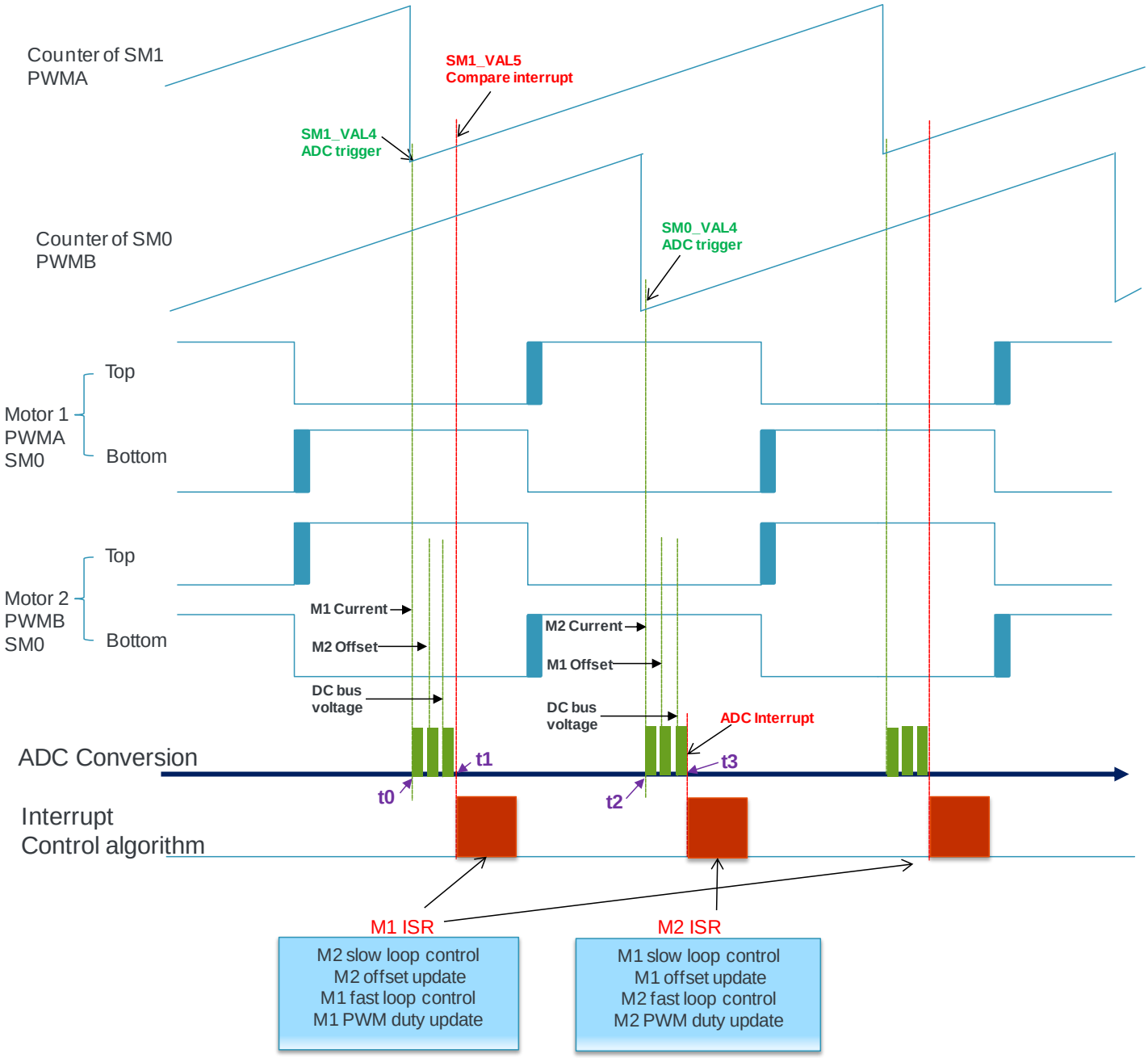

图 34 电机1与电机2的控制时序图

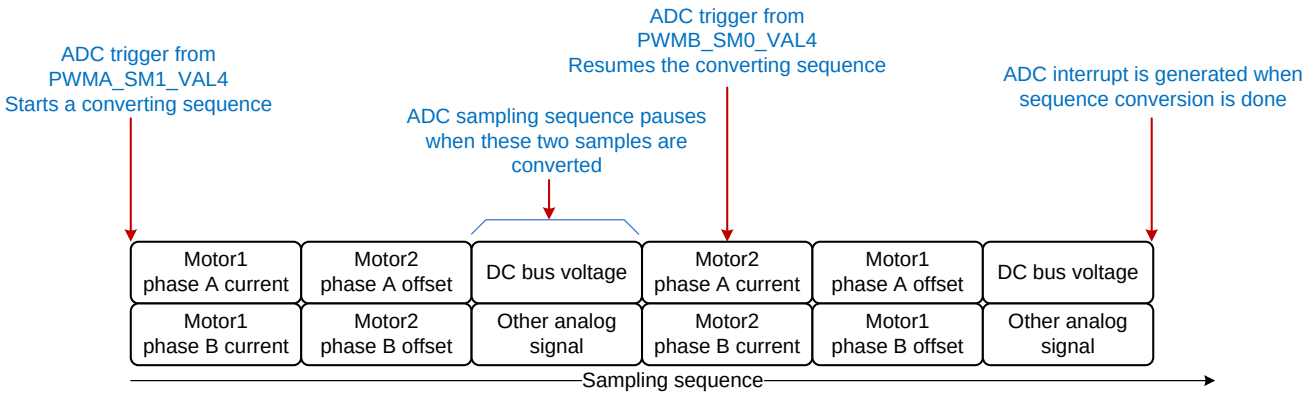

图 35 ADC转换序列的控制

电机1和电机2协同运行，在控制时序上有4个重要的时间点，按时间顺序循环运行，如图 34所示：

- t0时刻：电机1（PWMA控制）的SM1子模块中计数器与VAL4比较匹配触发ADC转换，整个转换序列有6次并行采样转换，如图 35所示。ADC结果寄存器（ADC12_RSLTn）中获得电机1的两相电流值、电机2的两相电流偏移值和母线电压值。3对通道转换完成后，ADC在Scan Control的控制下进入暂停状态，并等待下一次触发。
- t1时刻：电机1的SM1子模块中计数器与VAL5比较匹配触发PWM比较中断，进入M1 ISR。在M1 ISR中先运行电机2的慢速环（包括速度、位置的控制），并更新电机2的相电流偏移值，然后运行电机1的快速环（包括电流环控制），并更新电机1的PWM占空比。要谨慎设置VAL5的值，保证足够的时间供ADC三次采样转换。
- t2时刻：电机2的SM0子模块中计数器与VAL4比较匹配触发ADC继续转换序列，ADC结果寄存器（ADC12_RSLTn）中获得电机2的两相电流值、电机1的两相电流偏移值和母线电压值。ADC序列转换完成后，触发ADC转换完成中断，进入M2 ISR。
- t3时刻：在M2 ISR中先运行电机1的慢速环，并更新电机1的相电流偏移值，然后运行电机2的快速环，并更新电机2的PWM占空比。最后清除ADC转换完成标志。

PWMA与PWMB都设置为半周期重载（计数器与VAL0匹配时），在下桥臂导通中心时刻采样对应的两相相电流，在上桥臂导通中心时刻采样对应两相相电流偏差值（0A电流对应的采样值）。

4.3.3 时序的启动

外设初始化时，先将电机1（PWMA）的SM1VAL4比较触发、SM1VAL5的比较中断、电机2（PWMB）的SM0VAL4比较触发分别禁用。配置PWMA的SM0VAL5比较中断，这个中断只运行一次。在它的中断服务函数中，执行如下操作：

1. 清除PWMA的SM1VAL5的比较匹配标志。
2. 使能PWMA的SM1VAL4的比较匹配触发功能。
3. 使能PWMA的SM1VAL5的比较中断。

4. 使能PWMB的SM0VAL4的比较匹配触发功能。
5. 清除PWMA的SM0VAL5比较匹配标志，并禁用PWMA的SM0VAL5比较中断。

执行上述操作之后，就进入了图 34所示的t0~t3时刻的循环。

4.4 状态机

本应用使用状态机来实现双伺服电机控制，每个电机都由一套独立的状态机控制。该状态机由四个主状态组成：

- Fault状态：系统遇到过压、欠压、过流等错误所处的状态。
- Init状态：系统初始化。
- Stop状态：系统初始化完成后，进入Run之前所处的状态。电机在此状态下停止运行。
- Run 状态：系统运行状态。电机在此状态下运行。

这些主状态内还有如下的过渡函数：

- Init -> Stop：系统完成初始化后向Stop状态切换。
- Stop -> Run：在Stop状态中，当Run命令生效时，系统将进入Run状态。
- Run -> Stop：在Run状态中，当Stop命令生效时，系统将进入到Stop状态。
- Fault -> Init：错误标志标被清除时，系统将切换到初始化状态。
- Init, Stop, Run -> Fault：状态中发生错误时，系统将切换到Fault状态。

主状态机使用如下的标志实现各个主状态之间的切换：

- SM_CTRL_INIT_DONE：Init状态中，当此标志置位时，系统将从Init状态切换到Stop状态。
- SM_CTRL_FAULT：当此标志置位时，系统将从当前状态（Init，Stop或Run）切换到Fault状态。
- SM_CTRL_FAULT_CLEAR：当此标志置位时，系统将从Fault状态切换到Init状态。
- SM_CTRL_START：Stop状态中，这个标志将通知系统将从Stop状态切换到Run状态，并执行Stop -> Run的过渡函数。调用这个过渡函数的原因是系统切换状态之前可以做一些必要的处理。
- SM_CTRL_RUN_ACK：Stop状态中，若SM_CTRL_START标志置起，且该标志也置起，那么系统就从Stop状态切换到Run状态。这个标志就是承认系统可以从Stop状态切换到Run状态，一般在Stop -> Run的过渡函数中所有事物处理结束之后置起。
- SM_CTRL_STOP：Run状态中，这个标志将通知系统将从Run状态切换到Stop状态，并执行Run -> Stop的过渡函数。调用这个过渡函数的原因是系统切换状态之前可以做一些必要的处理。

- SM_CTRL_STOP_ACK：Run状态中，若SM_CTRL_STOP标志置起，且该标志也置起，那么系统就从Run状态切换到Stop状态。这个标志就是承认系统可以从Run状态切换到Stop状态，一般在Run -> Stop的过渡函数中所有事物处理结束之后置起。

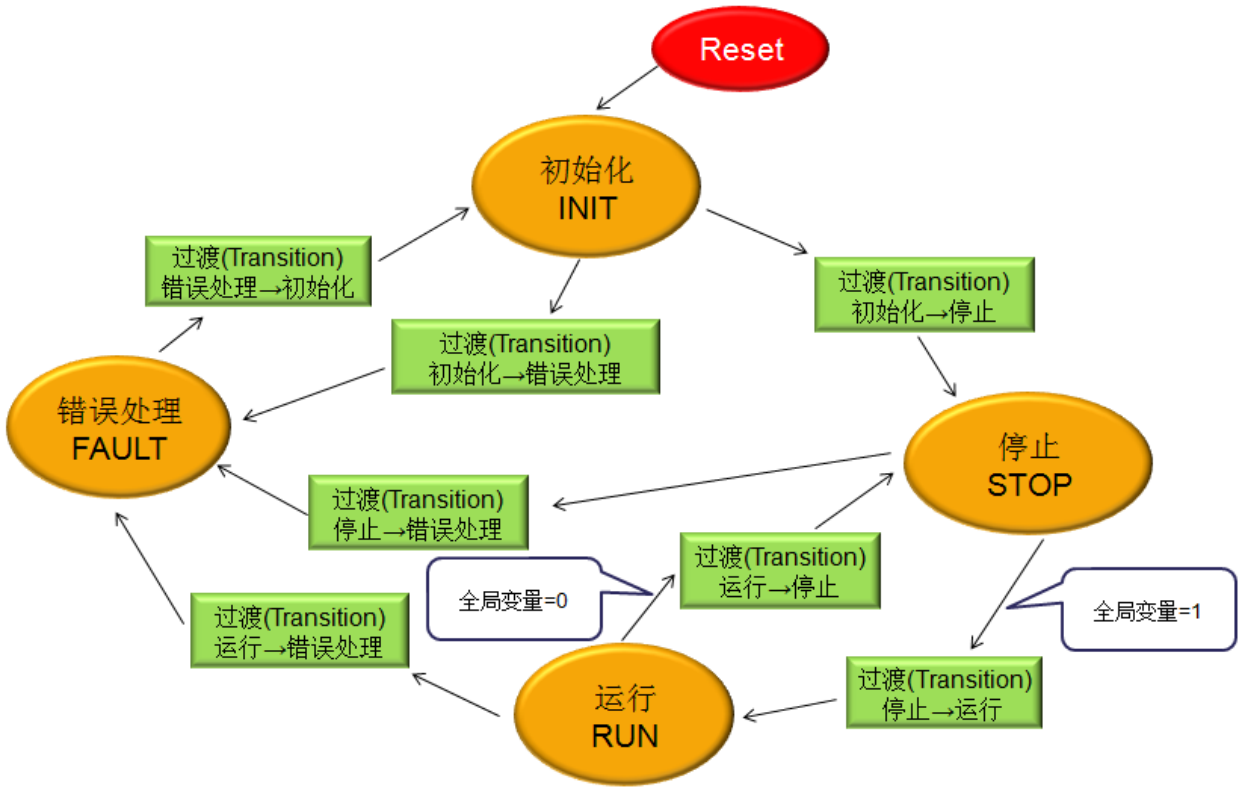

图 36 主状态机框图

此状态机数据结构定义在文件 state_machine.c和state_machine .h中。该状态机描述如下：

```
/* State machine control structure */
typedef struct
{
    __pmem SM_APP_STATE_FCN_T const*      psState; /* State functions */
    __pmem SM_APP_TRANS_FCN_T const*      psTrans; /* Transition functions */
    SM_APP_CTRL                           uiCtrl;  /* Control flags */
    SM_APP_STATE_T                        eState; /* State */
} SM_APP_CTRL_T;
```

状态机结构体有四个组成部分：

- psState：指向客户端状态机函数。当系统运行在特定的状态时，将调用客户端指定的状态机函数。
- psTrans：指向客户端状态过渡函数。当系统准备切换到另一个特定状态时，将调用指定的过渡函数。
- uiCtrl： 基于上面讲到的多种标志，这个变量用于控制状态机行为。

- eState：这个变量指示当前状态机所处的实际状态。

客户端状态机函数定义如下所示结构：

```
/* User state machine functions structure */
typedef struct
{
      PFCN_VOID_VOID    Fault;
      PFCN_VOID_VOID    Init;
      PFCN_VOID_VOID    Stop;
      PFCN_VOID_VOID    Run;
} SM_APP_STATE_FCN_T;
```

客户端过渡状态机函数定义见如下结构：

```
/* User state-transition functions structure*/
typedef struct
{
      PFCN_VOID_VOID    FaultInit;
      PFCN_VOID_VOID    InitFault;
      PFCN_VOID_VOID    InitStop;
      PFCN_VOID_VOID    StopFault;
      PFCN_VOID_VOID    StopInit;
      PFCN_VOID_VOID    StopRun;
      PFCN_VOID_VOID    RunFault;
      PFCN_VOID_VOID    RunStop;
} SM_APP_TRANS_FCN_T;
```

控制标志的变量见如下定义：

```
typedef unsigned short SM_APP_CTRL;

/* State machine control command flags */
#define SM_CTRL_NONE           0x0
#define SM_CTRL_FAULT          0x1
#define SM_CTRL_FAULT_CLEAR    0x2
#define SM_CTRL_INIT_DONE      0x4
#define SM_CTRL_STOP           0x8
#define SM_CTRL_START          0x10
#define SM_CTRL_STOP_ACK       0x20
#define SM_CTRL_RUN_ACK        0x40
```

状态标识变量见如下定义：

```
/* Application state identification enum */
typedef enum {
    FAULT          = 0,
    INIT           = 1,
    STOP           = 2,
    RUN            = 3,
} SM_APP_STATE_T;
```

状态机必须使用如下的内联函数在程序中周期性地调用。这个函数输入就是指向上述状态机结构体变量的指针。此结构在程序调用的地方被声明和初始化。

```
/* pointer to function with a pointer to state machine control structure */
typedef void (*PFCN_VOID_PSM)(SM_APP_CTRL_T *sAppCtrl);

/* State machine functions field (in pmem) */
__pmem const PFCN_VOID_PSM gSM_STATE_TABLE[4] = {SM_StateFault, SM_StateInit, SM_StateStop,
SM_StateRun};

/* State machine function */
extern inline void SM_StateMachine(SM_APP_CTRL_T *sAppCtrl)
{
      gSM_STATE_TABLE[sAppCtrl -> eState](sAppCtrl);
}
```

如何初始化并使用状态机结构的例子将在电机状态机中展示。

4.5 电机子状态机

电机子状态机包含在主状态机的Run状态中。首先，对主状态机的具体功能描述如下：

- Fault：当系统有错误发生时一直处于此状态，直到错误的标志被清除。在此状态内，采样直流母线电压并滤波处理。
- Init：此状态执行变量初始化。
- Stop：系统完成初始化等待Run命令。此状态内PWM输出被禁止。直流母线电压被采样并滤波处理。
- Run：系统处于运行状态，当有Stop命令时可以停止系统的运行。Run的子状态在此状态内被调用。

在这些主状态内还有如下的过渡函数:

- Init -> Stop：在这个函数中执行空任务。
- Stop -> Run：占空比被初始化为50%；PWM输出被使能。电流ADC通道初始化。Calib子状态被设置为Run子状态机的初始状态。
- Run -> Stop：当Stop命令生效时，系统将进入Stop状态。如果系统在特殊的Run子状态时，系统不会直接进入Stop状态，而是先过渡到自由停车（Freewheel）状态。
- Fault -> Init： 在这个函数中执行空任务。
- Init -> Stop：禁止PWM输出。
- Run -> Fault：电流和电压变量被清零。禁止PWM输出。

当主状态机在Run状态时，Run子状态机就会被调用，Run子状态机如中各状态描述如下：

- Calib：相电流偏置ADC校准。执行完此状态后系统将切换到Ready子状态。在此状态内，采样直流母线电压并滤波处理。PWM占空比设为50%且禁止输出。
- Ready：PWM占空比设为50%且使能输出。采样电流，配置ADC通道。初始化相关变量。
- Align：采样电流，配置ADC通道。调用初始转子位置搜索算法。在搜索算法完成之后，系统将切换到Spin子状态。采样直流母线电压并滤波处理。采样相电流偏置并滤波。
- Spin：采样电流并配置ADC通道。调用FOC算法。电机开始旋转，采样直流母线电压并滤波处理。采样电流偏置并滤波处理。调用位置环/速度环/电流环控制器。
- Freewheel：PWM占空比设为50%且禁止输出。采样电流并配置ADC通道。采样直流母线电压并滤波处理。采样电流偏置并滤波处理。由于转子惯性，系统将在此子状态等待一段时间，即等到转子静止为止。然后系统将评估现有条件，以确定切换到Align或者Ready子状态。如果有错误发生，系统将进入Fault状态。

Run子状态机也有对应的过渡函数，在子状态相互切换前调用。过渡子状态函数描述如下：

- Calib -> Ready：自校准完成，进入Ready状态。
- Ready -> Align：在Ready状态中出现非零速度命令时，将准备进入Align子状态。初始化一些变量（电压，速度，位置）。
- Align -> Ready：在Align状态中出现零速度命令，将进入Ready子状态。电压和电流相关的变量清零。PWM占空比设为50%。
- Align -> Spin：转子初始位置检测完成后系统将进入Spin子状态。PWM占空比设为50%。
- Spin -> Freewheel：在Spin状态中，当出现零速度命令时，将进入Freewheel子状态。初始化一些变量（电压，速度，位置）。确定自由停车的时间。
- Freewheel -> Ready：在Freewheel子状态，速度命令还是零时，并且自由停车时间到了，系统将进入Ready子状态。
- Freewheel -> Align：在Freewheel子状态，当速度命令非零时，系统将进入Align子状态。禁止PWM输出。初始化一些变量（电压，速度，位置）。

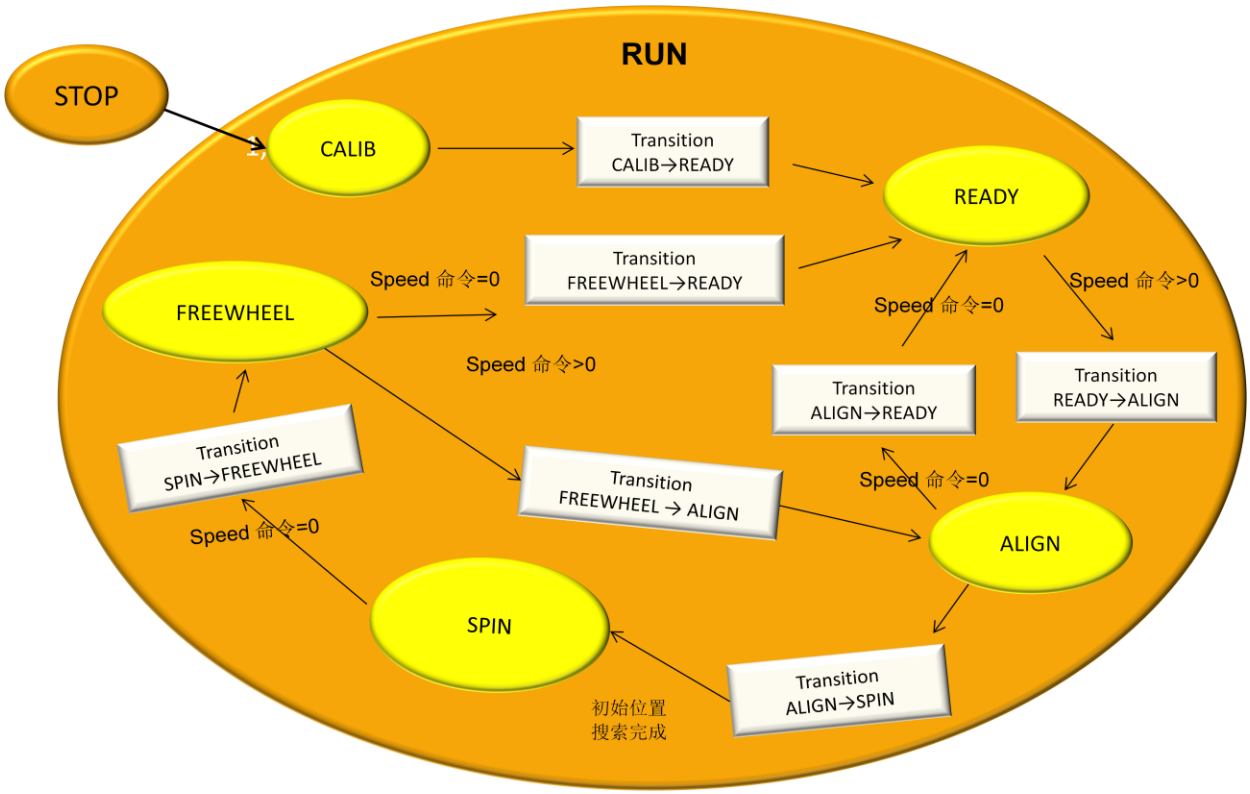

图 37 电机运行的子程序框图

在M1_statemachine.c .h和M2_statemachine.c .h两个头文件中定义了上述电机子状态机的应用。电机1的主状态机结构描述如下：

主状态机客户端函数原型：

```
static void M1_StateFault(void);
static void M1_StateInit(void);
static void M1_StateStop(void);
static void M1_StateRun(void);
```

主状态机客户端切换函数原型：

```
static void M1_TransFaultInit(void);
static void M1_TransInitFault(void);
static void M1_TransInitStop(void);
static void M1_TransStopFault(void);
static void M1_TransStopInit(void);
static void M1_TransStopRun(void);
static void M1_TransRunFault(void);
static void M1_TransRunStop(void);
```

主状态机函数表初始化：

```
/* State machine functions field (in pmem) */
__pmem static const SM_APP_STATE_FCN_T msSTATE = {M1_StateFault, M1_StateInit,
M1_StateStop, M1_StateRun};
```

主状态机过渡函数初始化：

```
/* State-transition functions field (in pmem) */
__pmem static const SM_APP_TRANS_FCN_T msTRANS = {M1_TransFaultInit, M1_TransInitFault,
M1_TransInitStop, M1_TransStopFault, M1_TransStopInit, M1_TransStopRun,
M1_TransRunFault, M1_TransRunStop};
```

最后介绍一下主状态机结构变量定义：

```
/* State machine structure declaration and initialization */
SM_APP_CTRL_T gsM1_Ctrl =
{
      /* gsM1_Ctrl.psState, User state functions  */
      &msSTATE,

      /* gsM1_Ctrl.psTrans, User state-transition functions */
      &msTRANS,

      /* gsM1_Ctrl.uiCtrl, Deafult no control command */
      SM_CTRL_NONE,

      /* gsM1_Ctrl.eState, Default state after reset */
      INIT
};
```

子状态机也是类似的声明。因此Run子状态机状态变量有如下定义：

```
typedef enum {
    CALIB     = 0,
    READY     = 1,
    ALIGN     = 2,
    SPIN      = 3,
    FREEWHEEL = 4,
} M1_RUN_SUBSTATE_T;          /* Run sub-states */
```

对于Run子状态机有两套函数定义。一套供快速环使用，另一套供慢速环使用。因此对于客户端状态函数原型如下：

```
static void M1_StateRunCalib(void);
static void M1_StateRunReady(void);
static void M1_StateRunAlign(void);
static void M1_StateRunSpin(void);
static void M1_StateRunFreewheel(void);

static void M1_StateRunCalibSlow(void);
static void M1_StateRunReadySlow(void);
static void M1_StateRunAlignSlow(void);
static void M1_StateRunSpinSlow(void);
static void M1_StateRunFreewheelSlow(void);
```

子状态机客户端过渡函数原型如下：

```
static void M1_TransRunCalibReady(void);
static void M1_TransRunReadyAlign(void);
static void M1_TransRunAlignReady(void);
static void M1_TransRunStartupFreewheel(void);
static void M1_TransRunSpinFreewheel(void);
static void M1_TransRunFreewheelAlign(void);
static void M1_TransRunFreewheelReady(void);
```

子状态函数表初始化：

```
/* Sub-state machine functions field (in pmem) */
__pmem static const PFCN_VOID_VOID mM1_STATE_RUN_TABLE[6][2] =
{
      {M1_StateRunCalib, M1_StateRunCalibSlow},
      {M1_StateRunReady, M1_StateRunReadySlow},
      {M1_StateRunAlign, M1_StateRunAlignSlow},
      {M1_StateRunSpin, M1_StateRunSpinSlow},
      {M1_StateRunFreewheel, M1_StateRunFreewheelSlow}
};
```

如上文所述，系统状态机在中断服务例程中调用。在中断服务例程里调用的状态机有两个： 一个是用于快速环控制，另一个用于慢速环控制。如何调用快速环的方法如下：

```
/* Fast loop calculation */
geM1_StateRunLoop = FAST;

/* StateMachine call */
SM_StateMachine(&gsM1_Ctrl);
```

同样地慢速环状态机的调用方法如下：

```
/* Slow loop calculation */
geM1_StateRunLoop = SLOW;

/* StateMachine call */
SM_StateMachine(&gsM1_Ctrl);
```

在Run状态机里，电机子状态机函数调用如下：

```
/* Run sub-state function */
mM1_STATE_RUN_TABLE[meM1_StateRun][geM1_StateRunLoop]();
```

第一维变量里先区分Run的子状态，然后才是快速环和慢速环的区分。对于电机2，代码也是以同样的方式进行。只是不再用M1命名，而是M2命名。

4.6 外设的使用情况

对于应用中特定的功能，需要使用下列外设（表 4）。这些外设具有特定的用途。

表 4 外设使用情况

Module	Purpose
PWM A	• 电机 1的 3相 PWM • 系统同步和时序控制 • 电机1过流保护
PWM B	• 电机 2 的 3 相 PWM • 电机2过流保护
ADC A & B	• 电机 1 & 2相电流 • 直流母线电压
TMR A0/B0	• 电机1/2的编码器脉冲计数
TMR A1/B1	• 电机1/2的M/T法测速
TMR A2/B2	• 电机1/2的Z信号处理
TMR A3/B3	• 电机1/2的增强型M/T法
Cross bar A	• PWMA到PWMB之间同步信号
Cross bar B & AOI	• PWM到ADC硬件触发信号
JTAG	• 上位机的通信，连接 FreeMASTER

4.7 示波器时序图

图 38显示的是电机1和电机2同时运行时的波形，其中：

- 第一个通道（蓝色）是电机1的A相下桥臂对应的PWM波形。其频率是16 kHz。
- 第二个通道（青色）是电机2的A相下桥臂对应的PWM波形。其频率是16 kHz。
- 第三个通道（红色）是GPIO的输出信号。该GPIO在进入电机1控制中断设置为1，出中断时清零。电机我们可以看到其频率和电机的PWM频率一样的。
- 第四个通道（绿色）是GPIO的输出信号。该GPIO在进入电机2控制中断设置为1，出中断时清零。电机我们可以看到其频率和电机的PWM频率一样的。

图 38和图 34的算法同步时序是对应关系，从图 38可以看出：

- 电机1和电机2的PWM信号有180度相位差。
- 电机1的PWM周期开始时，进入M1 ISR中。在电机1的半个PWM周期到来之前，完成占空比更新，退出M1 ISR。

- 电机2的PWM周期开始时，进入M2 ISR中。在电机2的半个PWM周期到来之前，完成占空比更新，退出M2 ISR。
- 实际的同步时序符合本文4.3.2节的设计要求。

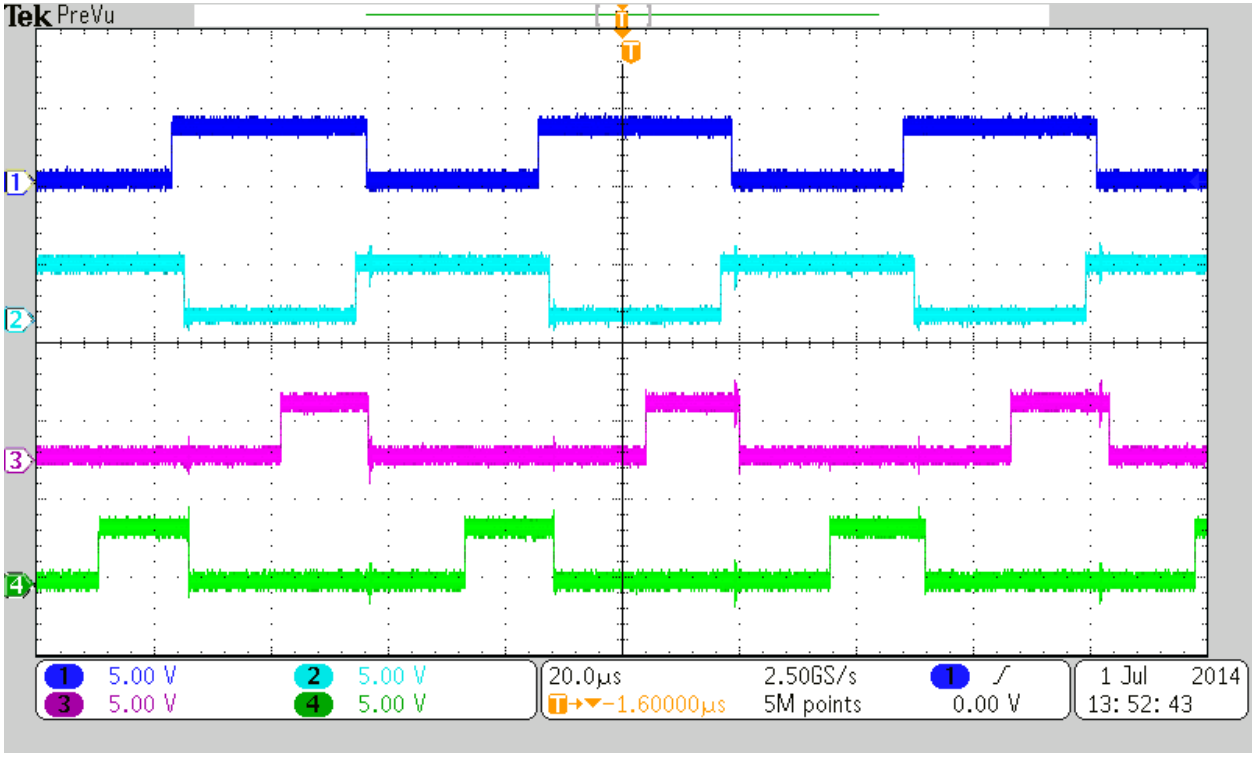

图 38 示波器时序图

How to Reach Us:

Home Page:
freescale.com

Web Support:
freescale.com/support

本文档中的信息仅供系统和软件实施方使用 Freescale 产品。本文并未明示或者暗示授予利用本文档信息进行设计或者加工集成电路的版权许可。 Freescale 保留对此处任何产品进行更改的权利，恕不另行通知。

Freescale 对其产品在任何特定用途方面的适用性不做任何担保、表示或保证，也不承担因为应用程序或者使用产品或电路所产生的任何责任，明确拒绝承担包括但不局限于后果性的或附带性的损害在内的所有责任。Freescale 的数据表和/或规格中所提供的“典型”参数在不同应用中可能并且确实不同，实际性能会随时间而有所变化。 所有运行参数，包括“经典值”在内，必须经由客户的技术专家对每个客户的应用程序进行验证。Freescale 未转让与其专利权及其他权利相关的许可。 Freescale 销售产品时遵循以下网址中包含的标准销售条款和条件：freescale.com/SalesTermsandConditions

Document Number: DRM155
Rev. 0
12/2014