

Technology  
实用技术

小型

# 交流伺服电机 控制电路 设计



YZLI0890167906

〔日〕石岛 胜 著  
薛 亮 祝建俊 译



科学出版社


( TN-1223.0101 )

责任编辑 杨 凯


责任制作 董立颖 魏 谨

封面设计 卢雪娇

**T**echnology  
实用技术



# 小型交流伺服电机 控制电路设计



## 小型直流电机 控制电路设计

www.sciencep.com

ISBN 978-7-03-035815-8



9 787030 358158 >

科学出版社 东方科龙公司  
联系电话: 010-82840399  
E-mail: boktp@mail.sciencep.com  
有关网址: <http://www.okbook.com.cn>

销售分类建议: 工业技术/电子技术

定 价: 36.00元

# 小型交流伺服电机 控制电路设计

〔日〕石岛 胜 著  
薛 亮 祝建俊 译



YZLI0890167905

科学出版社

北 京

图字: 01-2012-0416 号

## 内 容 简 介

本书主要介绍小型交流伺服电机的基本特性、设计方法及应用实例。主要内容包括交流伺服电机的基本原理、构造与特征,电机驱动电路,电机控制,反馈控制电路的设计,使用单片机控制交流伺服电机,驱动交流伺服电机的三相 PWM 控制回路,基于软件的伺服控制器的设计,基于汇编语言实现的伺服控制器高速化,交流伺服电机的控制实验等。书后还给出了与直流电机有关的专业名词解释,对于读者理解书中的内容有很大的帮助。

本书内容实用性强、结构清晰合理、言简意赅,对实际操作有很强的指导性和借鉴意义。

本书适合工科院校电子、电工等相关专业的师生参考阅读,同时适合作为广大电气从业人员的参考用书。

### 图书在版编目(CIP)数据

小型交流伺服电机控制电路设计/(日)石岛胜著;薛亮,祝建俊译.  
—北京:科学出版社,2012

ISBN 978-7-03-035815-8

I. 小… II. ①石…②薛…③祝… III. 小型电机-交流伺服电机-控制  
电路-电路设计 IV. TM383.402

中国版本图书馆 CIP 数据核字(2012)第 248424 号

责任编辑:杨 凯/责任制作:董立颖 魏 谨

责任印制:赵德静/封面设计:卢雪娇

北京东方科龙图文有限公司 制作

<http://www.okbook.com.cn>

**科 学 出 版 社** 出版

北京东黄城根北街 16 号

邮政编码:100717

<http://www.sciencep.com>

骏杰印刷厂 印刷

科学出版社发行 各地新华书店经销

\*

2013 年 1 月第 一 版 开本: B5(720×1000)

2013 年 1 月第一次印刷 印张: 15 1/4

印数: 1—5 000 字数: 225 000

定 价: 36.00 元

(如有印装质量问题,我社负责调换)

# 前言



本书是以月刊《晶体管技术》从 2005 年 12 月开始的特辑“电机的基础与实用控制电路设计”为基础,增添、修改完成的。希望能作为交流伺服电机的入门书与实用书奉献给读者。

控制电机需要广范围的知识与经验,不仅需要理解电机的原理与构造,更需要关于驱动方法以及控制电路的知识和经验。

鉴于上述原因,本书能够传达给读者的内容非常多,涵盖众多领域技术。我也考虑过是否应该限制一下涉及的领域,但是为了解说交流伺服电机的实际控制方法,还是保留了大部分必要内容。

本书主要涉及以下技术领域:

- (1) 电机的构造和原理。
- (2) 强电。
- (3) 微处理器的应用。
- (4) 数字信号处理。
- (5) 电机调整/控制。

书中引用的数学式与程序代码,大部分都是以中学与高中学习过的三角函数及微积分作为基础的。

本人是在设备制造厂工作的技术人员,在实际的设计中看到,与大学级别的高等数学相比,应用更多的是中学与高中级别的初等数学。希望各位能够通过本书理解这样的状况。

本书的读者对象如下:

- (1) 刚刚接触电子系统或者机械系统设计的工程师。
- (2) 亲自参与电机控制的非专业设计者。
- (3) 研究电机及机器人等领域的工科生。



若能够让年轻的技术者及研究者读到本书,则是作者的幸运。本书列举的电机控制主板是费了许多年月开发而已经商品化的东西。能够将电机驱动方法及控制回路的实践解说得如此详尽的书籍尚不多见,希望读者能感受到无法从教科书中得到的现实中的设计感觉。

最后,我要感谢给了我这次机会并且与我共同努力了两年的《晶体管技术》编辑部及制作部的全体同仁。

# 目 录

## 第 1 部分 基础篇

第 1 章 交流伺服电机的基础 .....	3
1.1 本书的重点课题 .....	3
1.2 本书中使用的交流伺服电机 .....	4
1.2.1 代表性的交流伺服电机 .....	4
1.2.2 交流伺服电机的内部构造 .....	5
1.2.3 交流伺服电机的用途 .....	5
1.3 电机的构造及原理 .....	7
1.3.1 直流电机的基本构造 .....	8
1.3.2 直流电机的转动原理 .....	9
1.3.3 无刷直流电机的基本构造 .....	10
1.3.4 无刷电机的旋转原理 .....	11
1.4 电机的分类与直流伺服电机在分类中所处的位置 .....	12
1.4.1 直流伺服电机属于旋转电机 .....	13
1.4.2 交流伺服电机是有逆变器电路的交流电机 .....	13
1.4.3 交流伺服电机是同步电机 .....	14
1.4.4 交流伺服电机的特征 .....	14
1.5 伺服是什么 .....	15
1.5.1 伺服机构与伺服电机 .....	15
1.5.2 直流伺服电机 .....	15
1.5.3 RC 伺服 .....	16
1.5.4 位置检出机构 .....	18
1.5.5 交流伺服电机的系统构成 .....	20

<b>第 2 章 交流伺服电机的构造与特征</b>	22
2.1 产生旋转力的原理	22
2.1.1 有关磁铁的知识	22
2.1.2 实际的电机	23
2.1.3 电机的恢复转矩和平衡点	25
2.2 旋转力与步进驱动	26
2.2.1 制造旋转磁场的方法	26
2.2.2 使其连续的旋转	26
2.2.3 决定位置机构的特征	28
2.2.4 控制位置	28
2.2.5 控制速度	29
2.3 无刷直流电机驱动	29
2.3.1 让电机不停止地转动	29
2.3.2 不会产生失步	30
2.3.3 检测转子的位置	30
2.3.4 控制速度	32
2.3.5 控制位置	33
2.4 交流电机驱动(正弦波驱动)	33
2.4.1 交流伺服电机驱动(正弦波驱动)的电流分配	34
2.4.2 交流伺服电机驱动(正弦波驱动)的转子位置检测	35
2.4.3 很容易实现高旋转/高效率的电机驱动的无刷直流电机 驱动	36
2.4.4 很难实现高旋转/高效率的电机驱动的交流伺服电机驱动	37
2.5 基于正弦波驱动的微步进驱动	40
2.5.1 三相正弦波驱动的稳定点	40
2.5.2 转子的步进角不总是正确的	40
2.5.3 抑制电机振动的效果	41
2.6 有刷直流电机	41



<b>第 3 章 电机驱动电路的基础</b>	43
3.1 电机驱动的基础知识	43
3.1.1 力与转矩的关系	43
3.1.2 转矩和功率的关系	44
3.1.3 电机驱动的 4 象限运行过程	45
3.1.4 直流电机的等价电路	46
3.1.5 直流电机的转数可以通过电压控制	47
3.1.6 线性放大器驱动和 PWM 放大器驱动	48
3.2 电机的 PWM 驱动方法	52
3.2.1 电桥电路 PWM 驱动的 MOSFET 栅极驱动器电路的例子	53
3.2.2 电桥 PWM 驱动方法的特征	54
3.2.3 线性 PWM 驱动的 MOSFET 栅极驱动器电路的例子	56
3.2.4 线性 PWM 驱动方法的特征	59
3.3 基于 PWM 驱动的电机电流的检测方法	61
3.3.1 直接测量电机线的方法(A 方法)	62
3.3.2 测量电压侧 MOSFET 的电源和 GND 的电流的方法 (B 方法)	63
3.3.3 测量电源电流的方法(C 方法)	63
3.3.4 测量电机电流的电路实例(A 方法)	63
<b>第 4 章 电机控制的基础</b>	65
4.1 电机控制的基础知识	65
4.1.1 拉普拉斯变换是什么?	65
4.1.2 传递函数是什么?	66
4.1.3 方框图是什么?	66
4.1.4 电机的传递函数	67
4.1.5 转矩控制放大器的传递函数	69
4.1.6 传感器的传递函数	70
4.1.7 电机驱动部的传递函数	70
4.2 伺服控制器的作用	71
4.2.1 位置控制的运作原理	71

4.2.2	速度控制的运作原理 .....	72
4.3	数字信号处理的基础知识 .....	73
4.3.1	延迟单元 $1/z$ .....	73
4.3.2	连续系统传递函数和离散系统传递函数 .....	73
4.3.3	数字滤波器的设计方法 .....	74
4.4	数字滤波器设计帮助软件(DSP.EXE) .....	76
4.4.1	模拟滤波器设计 .....	77
4.4.2	解析模拟滤波器频率特性(c3) .....	78
4.4.3	$s$ - $z$ 变换(c4) .....	78
4.4.4	解析数字滤波器频率特性(c5) .....	79
4.4.5	数字滤波器参数的保存(c6) .....	79
4.4.6	数字 PID 控制器的设计(w7) .....	80
4.4.7	数字位置/速度环控制器的设计(w8) .....	80
4.5	抓住滤波器设计的特征 .....	81
4.5.1	2 次 LPF 的特征 .....	81
4.5.2	2 次陷波滤波器的特征 .....	82
4.5.3	PID 控制器的特征 .....	83
4.5.4	位置/速度环控制器的特征 .....	85
4.6	伺服控制器的系统设计 .....	86
4.6.1	伺服控制器的构成 .....	86
4.6.2	目标轨迹生成器 .....	86
4.6.3	系统延迟 .....	87
4.6.4	反馈控制器 .....	88
4.6.5	前馈控制器 .....	88
4.6.6	输出滤波器 .....	88

## 第 2 部分 应用篇

第 5 章	使用单片机控制交流伺服电机的基础 .....	91
5.1	H8S/2367F 和 CoolRunner II 的主要特征 .....	91
5.1.1	16 bit 高速 H8S/2000 CPU .....	91

5.1.2	丰富的周边功能 .....	92
5.1.3	CoolRunnerII (XC2C256)的主要特征 .....	92
5.2	单片机的使用方法 .....	92
5.2.1	PWM 信号的产生方法 .....	93
5.2.2	DMA 的使用方法 .....	94
5.2.3	DTC 的使用方法 .....	97
5.2.4	环路计算用的时基的产生方法 .....	99
5.2.5	中断控制器的使用方法 .....	100
5.2.6	增计数的产生方法 .....	101
5.2.7	实时通信接口(SCI)的使用方法 .....	102
5.3	PLD 的使用方法 .....	103
5.3.1	CPLD 及 FPGA .....	103
5.3.2	CPLD 的设计工具 .....	103
5.3.3	设计中的注意点 .....	103
5.4	交流伺服电机控制回路的系统设计 .....	104
5.4.1	设计目标 .....	104
5.4.2	三相正弦波 PWM 驱动 .....	104
5.4.3	自增计数器 .....	105
5.4.4	PWM 频率 .....	106
5.4.5	电流环频率 .....	106
5.4.6	伺服环频率 .....	106
5.4.7	电机电流检出频率 .....	107

## 第 6 章 驱动交流伺服电机的三相 PWM 控制回路的 实践 .....

6.1	设计参数 .....	108
6.1.1	控制部分的参数 .....	108
6.1.2	驱动部分的参数 .....	108
6.2	使用部件及电路的选定 .....	109
6.2.1	MOSFET 的确定 .....	109
6.2.2	MOSFET 栅极驱动电路 .....	110

6.2.3 电机电流检测电路 .....	111
6.3 交流伺服电机驱动硬件电路 .....	112
6.3.1 使用微控制器内置的 TPU 生成三相 PWM 的方法 .....	112
6.3.2 非重叠 PWM 信号 .....	113
6.3.3 转子位置检测电路(CPLD) .....	115
6.4 向量控制的基础知识 .....	118
6.4.1 模拟的参数 .....	119
6.4.2 电机电流三相(U,V,W)的波形 .....	120
6.4.3 电机电流的三相(U,V,W)→二相( $\alpha, \beta$ )转换 .....	120
6.4.4 电机电流的二相( $\alpha, \beta$ )到 $d-q$ 的转换 .....	121
6.4.5 PI(比例积分)控制器 .....	123
6.4.6 PI 控制器 $d-q \rightarrow \alpha'-\beta'$ 变换 .....	125
6.4.7 二相( $\alpha', \beta'$ )→三相(U,V,W)变换 .....	126
6.5 基于微控制器的向量控制实验 .....	127
6.5.1 计算转子的位置,从 LUT(Look Up Table)中读取 sin 值和 cos 值 .....	128
6.5.2 U 相和 V 相的电机电流值的读取、偏差修正、增益修正 .....	129
6.5.3 对电机电流进行 $\alpha\beta$ 变换,再进行 $d-q$ 变换 .....	131
6.5.4 PI(比例积分)控制器 .....	132
6.5.5 PI 控制器 $d-q \rightarrow$ 二相( $\alpha', \beta'$ )→三相(U,V,W)变换 .....	135
6.5.6 三相(U,V,W)→PWM 变换 .....	135
6.5.7 对应各种各样的电机 .....	139
<b>第 7 章 基于软件的伺服控制器的设计</b> .....	141
7.1 伺服控制器的构成 .....	141
7.1.1 读取编码计数器的值,计算位置和速度 .....	141
7.1.2 计算反馈(PID 或者是位置/速度环) .....	143
7.1.3 前馈 & 输出滤波器 .....	145
7.1.4 伺服输出(转矩指令)处理 .....	146
7.1.5 标准值的生成 .....	148
7.2 反馈控制器 .....	149

7.2.1	PID 控制器的运算步骤 .....	149
7.2.2	位置/速度环控制器计算路径 .....	150
7.3	IIR 数字滤波器计算路径 .....	151
7.4	前馈计算路径 .....	154
7.5	目标轨迹生成器 .....	155
7.5.1	目标轨迹生成相关的基础知识 .....	155
7.5.2	目标位置轨迹的生成方法 .....	157
7.5.3	目标速度轨迹的生成方法 .....	158
<b>第 8 章</b>	<b>基于汇编语言实现的伺服控制器高速化 .....</b>	<b>160</b>
8.1	IEEE 标准的单精度浮点 .....	160
8.1.1	单精度浮点的位配置 .....	160
8.1.2	单精度浮点的表示方法 .....	161
8.2	16 位精度浮点 .....	162
8.2.1	16 位精度浮点的位的设置 .....	162
8.2.2	16 位精度浮点的表示方法 .....	163
8.2.3	16 位精度浮点的计算精度 .....	164
8.2.4	16 位精度浮点的计算方法 .....	164
8.3	PID 控制器 .....	180
8.3.1	寄存器的使用方法 .....	182
8.3.2	计算顺序和运行时间 .....	183
8.3.3	在积分项中设置极限 .....	184
8.4	位置/速度环控制器 .....	184
8.4.1	寄存器的使用方法 .....	187
8.4.2	计算顺序和运行时间 .....	187
8.4.3	在积分项中设置极限值 .....	188
8.5	IIR 数字滤波器 .....	188
8.5.1	寄存器和局部变量的使用方法 .....	190
8.5.2	计算顺序和运行时间 .....	192

<b>第 9 章 交流伺服电机的控制实验</b>	193
9.1 电机控制实验的设备	193
9.1.1 电机控制主板	194
9.1.2 使用的电机	195
9.1.3 实验中用的负荷	195
9.2 伺服调整方法	196
9.2.1 有关伺服调整的一些参数	196
9.2.2 伺服锁定的实现	197
9.2.3 伺服调整工具	200
9.2.4 PID 参数的决定方法	201
9.2.5 加速度极限和速度极限的确定方法	202
9.2.6 驱动极限(转矩控制放大器的电流极限)的确定方法	204
9.2.7 前馈的确定方法	204
9.2.8 数字滤波器的系数	204
9.3 电机控制实验(PID 控制器)	204
9.3.1 微小距离传送的定位性能	204
9.3.2 高速传送响应性能	207
9.3.3 速度控制性能	207
9.4 电机控制实验(位置/速度环控制器)	208
9.4.1 位置/速度环控制器是二重反馈控制器	209
9.4.2 伺服增益的决定	209
9.4.3 开环频率特性	210
9.5 电机的输出功率	213
9.5.1 电机转动圈数和输出功率的关系	213
9.5.2 电机转动是在做功	214
<b>附 录 交流/直流伺服电机驱动 MCG02</b>	215
专业术语解释	217
参考文献	229

第 **1** 部分

---

**基础篇**





# 第 1 章

## 交流伺服电机的基础

交流伺服电机一般是从电机制造商处把商品电机及伺服放大器作为一套而购入的。本人是在电机制造厂工作的技术人员,然而数年前我从未想过会由我自己来设计交流伺服电机的控制电路。

刚开始设计交流伺服电机的控制电路的理由是因为最近广受注目的人形机器人。人形机器人是模仿人类而制造的机器人,所以要使用非常多的电机。电机制造商制造的电机与伺服放大器,尺寸都过大,因此必须对电机及伺服放大器进行小型化。在这种背景下,我得到了在公司开发伺服放大器的机会。

刚刚开始开发交流伺服电机的伺服放大器的时候,我在书店与网络上寻找过相关的书籍,然而介绍电机的种类、特征、构造的书虽然多,但是关于交流伺服电机的实用控制方法及控制电路的书籍非常少。为了收集到设计所必需的资料我费了相当大的工夫。

### 1.1 本书的重点课题

电机的历史非常悠久,可以追溯到 19 世纪初。当时,电机本来是为了旋转而发明的一种设备。随着技术的发展,电机被用在各种各样的地方,电机的材料、构造及控制方法都得到了大幅度的提高,对电机的转动精度与效率等有了更多要求。近年因为地球温暖化问题,从保护环境观点来说,高效而清洁的电机技术将愈发重要。从控制层面来看,关键技术是矢量控制技术和高精度的

伺服控制技术。

交流伺服电机在所有的电机中是最昂贵的一种,用途仅限于 FA(Factory Automation)领域,但是在交流伺服电机上开发出来的矢量控制技术和高精度伺服控制技术正迎合了这样的时代要求。因此,在冰箱、空调这些家庭电器上,以及在工业机器、铁路、汽车等领域,开发低成本、高效率的电机控制技术成为最近非常热门的研究课题。作为结果,一部分家庭电器实现了不用电流传感器与速度传感器的无线传感器(sensorless)矢量控制,空调及冰箱的耗电量降低到了原来的  $1/4$ 。

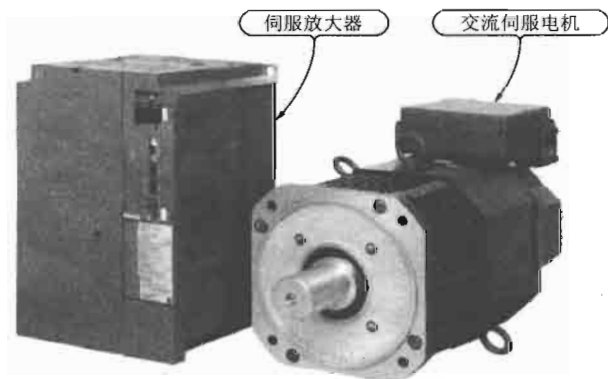
本书的重点课题是对交流伺服电机的矢量控制及伺服控制进行技术解说,目的是希望读者能理解这些控制技术。

## 1.2 本书中使用的交流伺服电机

交流伺服电机从某种意义上来说是特殊的电机,在家庭中较难看见。可能诸位有听说过交流伺服电机,但是实际上见过交流伺服电机的读者估计寥寥无几。那么就让我们看看交流伺服电机到底是什么样的吧。

### 1.2.1 代表性的交流伺服电机

照片 1.1 是典型的交流伺服电机(三菱电机)的外观。像这样,交流伺服电机由电机与伺服放大器组成,作为一个系统而工作。交流伺服电机中功率较小

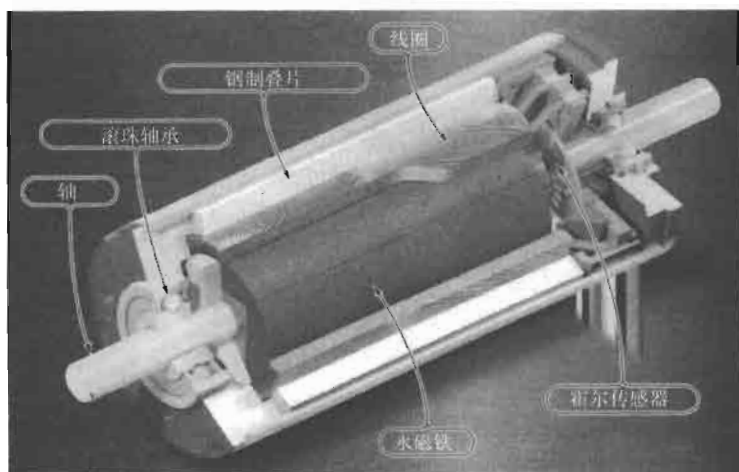


照片 1.1 典型的交流伺服电机(三菱电机)外观

的为 6W,功率较大的能达到 7kW 左右。一般设计伺服放大器输入端为交流电源,有 AC100V、AC200V 和 AC400V 等类型。

### 1.2.2 交流伺服电机的内部构造

照片 1.2 显示的是 maxon motor 公司生产的交流伺服电机的内部构造。交流伺服电机由永磁铁制成转子,线圈制成定子而构成无刷构造。一般来说,永磁转子通常是二极(也有四极乃至八极)。定子线圈是将导线卷起来再用树脂固定的杯状电枢线圈,是个无芯的线圈。



照片 1.2 交流伺服电机的内部构造(maxon motor 公司)

另外,与增量式编码器组合的电机还包含能够感应转子位置的霍尔传感器。

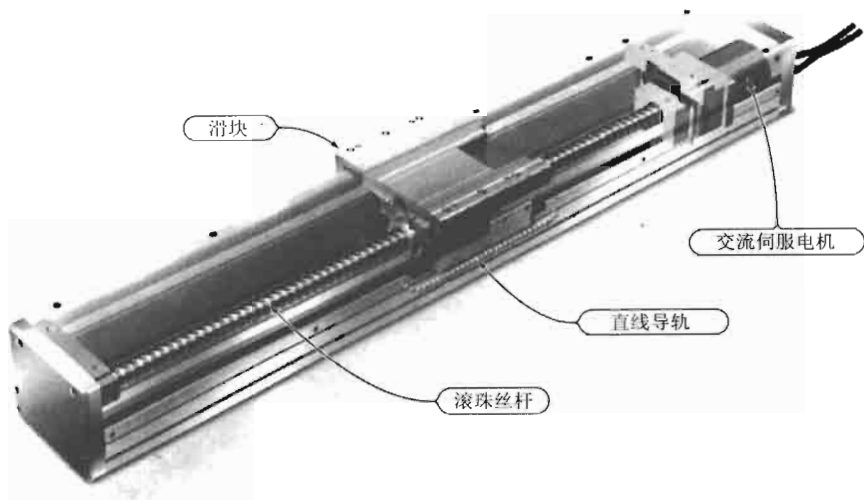
### 1.2.3 交流伺服电机的用途

交流伺服电机的特点在于精密的位置控制。因此,它的用途主要是精密测量台及工业机器人。而且,最近广受瞩目的交流伺服电机的用途是人形机器人。我们在电视上经常可以见到的人形机器人主要是用 RC 伺服的,而为企业开发的机器人主要使用交流伺服电机或者无刷直流电机。但是,这些机器人由于制作成本过高而大多数无法商品化。已经商品化的产品其价格也可以赶得上一台高级轿车。

以下是几个交流伺服电机的应用例子。

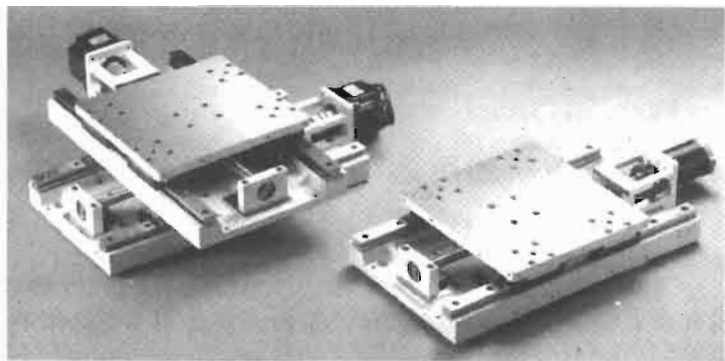
## 1. 精密测量台

照片 1.3 是 IAI 公司生产的单轴机器人。主要的构成部件除了交流伺服电机外,还有滚珠丝杆、滑块、直线导轨。像这样把单轴机器人组合起来就可以制成精密位置测量台。



照片 1.3 单轴机器人(IAI)

照片 1.4 是 THK 公司生产的精密 XY 测量台。这个 XY 测量台除了交流伺服电机外,还可以安装步进电机,最高精度可以达到  $6\mu\text{m}$ 。



照片 1.4 精密测量台(THK)

## 2. 工业机器人

照片 1.5 是 Epson 公司生产的 6 轴机械臂机器人。内部有 6 个交流伺服

电机,手臂的前端有螺丝刀和用于抓取物体的手,是工业用机器人。

### 3. 人形机器人

照片 1.6 是富士通制作的人形机器人。这个机器人有 26 个关节,除了交流伺服电机之外,还有一部分使用了 RC 伺服。此款机器人被作为研究双足步行的商品而开始销售,2009 年已经停止生产了,价格大约相当于一台高级汽车。



照片 1.5 机械臂(Epson)



照片 1.6 人形机器人(富士通)

## 1.3 电机的构造及原理

电机说到底是什么呢?在详细说明交流伺服电机之前,让我们来回顾一下基本的有刷直流电机及无刷电机的构造和转动原理吧。直流电机一般是指有刷直流电机,本书所说的直流电机即为有刷直流电机。

电机是能够把电能转换成动能的装置。在众多的电机中最熟悉的电机是直流电机,此种电机由电磁铁与永磁磁极组合而成,能够连续不断地转动。

磁铁有 S 极与 N 极,S 极与 N 极有相互吸引力,而同样的极之间则有斥力。电磁铁是由通过线圈的电流流动而制造出来的,因此可以根据电流的流向改变电磁铁的磁极,而由电流的大小可以改变磁力的大小。直流电机是通过控制流经线圈的电流的方向和大小,进而控制永磁体与电磁铁之间的引力和斥力,产生连续的运动。

作为交流伺服电机代表的无刷电机的基本原理和直流电机是一样的,其区别仅在于如何控制流经线圈的电流的大小和方向。

### 1.3.1 直流电机的基本构造

图 1.1 是 3 槽 2 极直流电机的基本构造图。在此要补充一下有关开槽转子的知识。开槽铁心是指包裹着线圈的铁心之类的总称,电机内部的铁心上如果缠绕着 3 个线圈就叫 3 槽铁心。直流电机根据其用途,有许多的极及槽数。多极、多槽的电机因为构造复杂,会导致价格变高,但是齿槽转矩(不通电时将电机的轴从外部转动时感受到的力矩)及转矩波动(力矩的波动)会比较小。

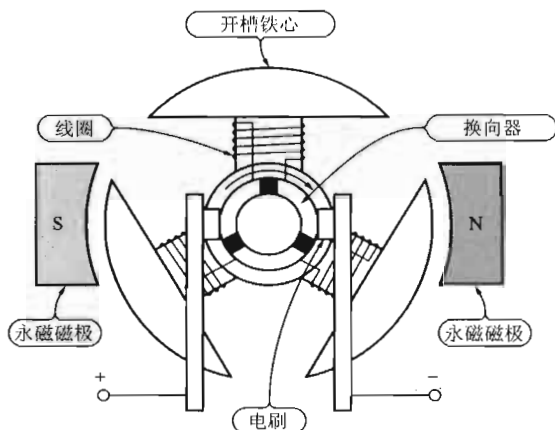


图 1.1 直流电机的基本构造(3 槽, 2 极)

另外,从照片 1.2 的交流伺服电机构造中可以看出,也有使用将导线卷起来用树脂固定的杯状电枢线圈的无心无槽的直流电机,这种电机一般被称为无心直流电机,在直流伺服电机中被广泛采用。对于电机来说,只要磁束通过铁心(照片 1.2 中是钢制叠片),没有必要与线圈一起旋转,把铁心固定而只旋转线圈即可。这样做的好处是转子的惯性量可以被降低,比较适合制作直流伺服电机。

让我们再一次看看图 1.1 吧。直流电机由作为定子的永磁磁极、电刷、开槽铁心、线圈、换向器等构成。直流电机正如其名,加在电机的电压是直流的,电机在转动时换向器和电刷起到了开关的作用,让电流流经的线圈一个个地进行切换。

### 1.3.2 直流电机的转动原理

图 1.2 表示了直流电机的转动原理。电机在转动时换向器和电刷起到了开关的作用,将电流流经的线圈按次序每  $60^\circ$  交换。

请看看图 1.2(a)。这里请关注直流电机的线圈的接线方法是  $\Delta$  连接。虽然直流电机一般不讨论线圈的接线方式,但本图中确实是  $\Delta$  连接。这个转子的位置将由电刷和换向器接触的状态开始,线圈 A,B,C 中有图中用粗线表示的电流流过。S 极被线圈 A 励磁,N 极被线圈 B 和线圈 C 励磁。

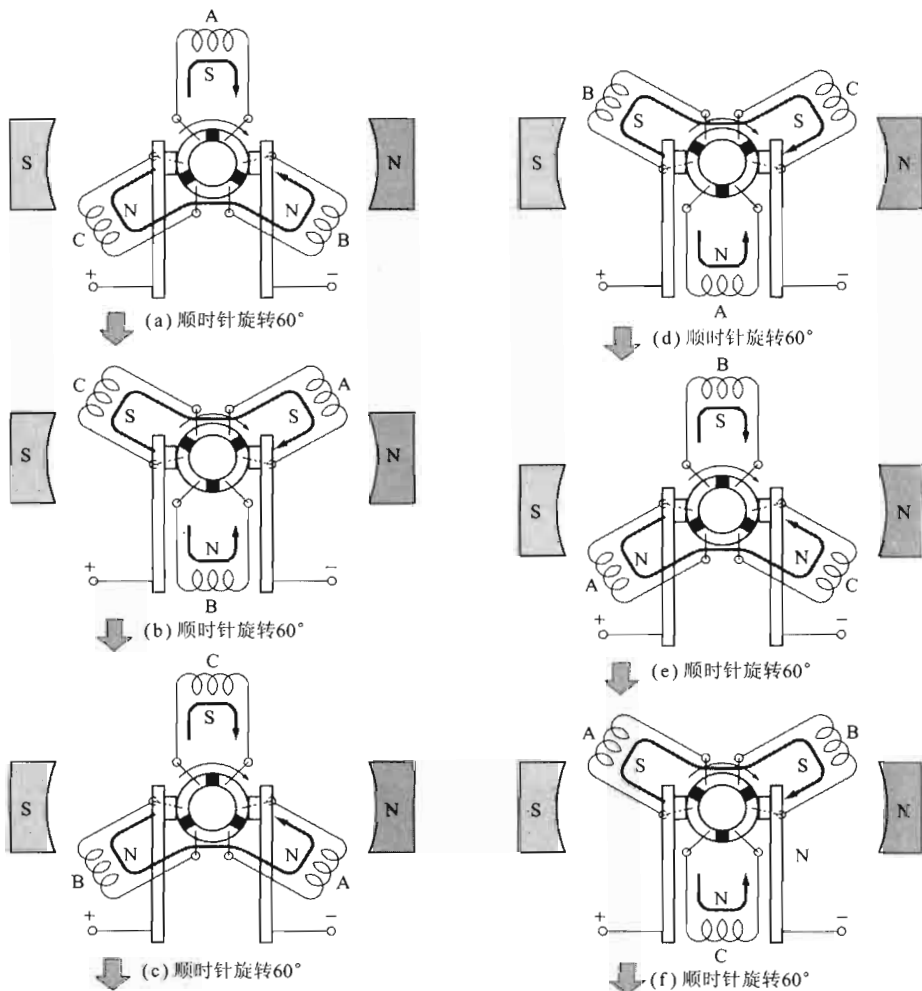


图 1.2 直流电机的旋转原理(3 槽,2 极)

这样的话,各线圈与定子的永磁磁极之间将产生引力和斥力,结果转子顺时针旋转。

图 1.2(b)是从图 1.2(a)的位置开始,转子顺时针转过  $60^\circ$  后的位置。从图中可以看出,该状态下线圈 A、B、C 有粗线所示的电流流过。S 极被线圈 A 及线圈 C 励磁,而 N 极被线圈 B 励磁。这样的话就和图 1.2(a)一样,每个线圈与定子的永磁磁极之间产生引力和斥力,导致转子继续顺时针旋转。

图 1.2(c)是从图 1.2(b)的位置开始,转子顺时针转过  $60^\circ$  后的位置。从图中可以看出,该状态下线圈 A、B、C 有粗线所示的电流流过。N 极被线圈 A 及线圈 B 励磁,而 S 极被线圈 C 励磁。这样的话就和图 1.2(b)一样,每个线圈与定子的永磁磁极之间产生引力和斥力,导致转子继续顺时针旋转。

图 1.2(d)、图 1.2(e)和图 1.2(f)的说明就此省略。

### 1.3.3 无刷直流电机的基本构造

交流伺服电机是无刷构造的电机中的一种。图 1.3 是一块带有两个极性的永磁磁极和 3 个槽的无刷电机的基本构造图。无刷电机是由定子的开槽铁心及线圈,以及转子的永磁磁极构成的。与图 1.1 的直流电机的基本构造图进行对比应该可以发现,本图中的无刷电机中转子和定子的关系是相反的。电机的构成(3 槽,2 极)是与图 1.1 中的直流电机的基本构造一样的。

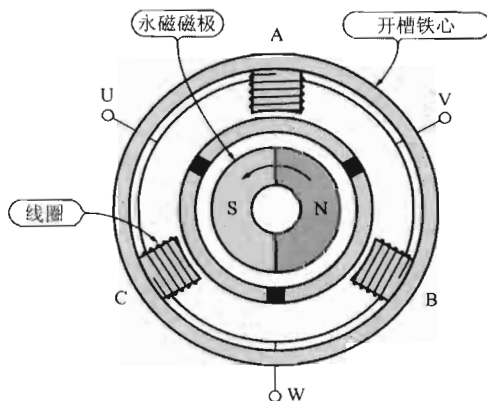


图 1.3 无刷电机的基本构造(3 槽, 2 极)



### 1.3.4 无刷电机的旋转原理

图 1.4 显示了无刷电机的旋转原理。电机在旋转的时候,通过不断改变电机端子(U,V,W)的电压,让电流流经的线圈按顺序每  $60^\circ$  切换。

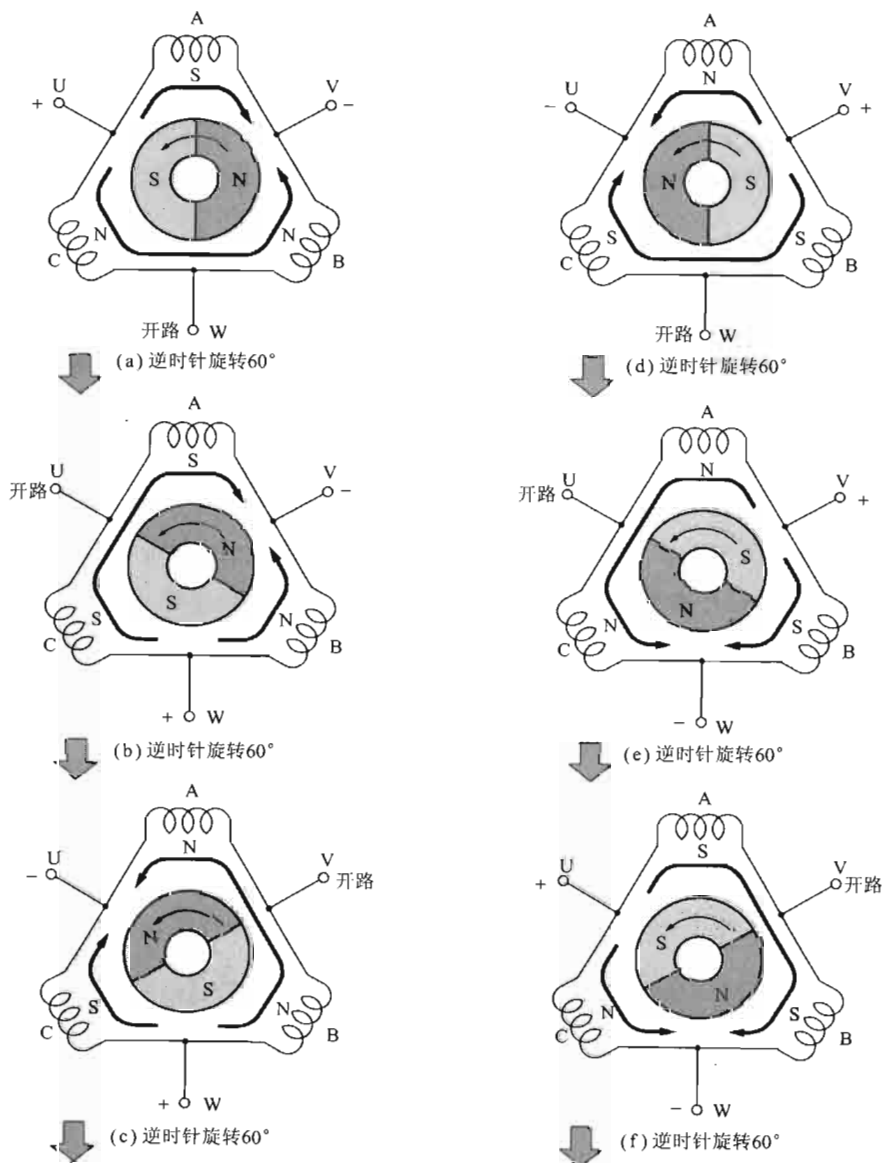


图 1.4 无刷电机的旋转原理(3 槽, 2 极)

请看看图 1.4(a)。线圈的接线方式与直流电机一样,是  $\Delta$  连接。无刷电机的线圈的接线方式有  $\Delta$  连接与 Y(星形)连接,Y 连接方式较多被采用。尤其是小型的电机几乎都是 Y 连接的。本图为了与直流电机进行对比而采用了  $\Delta$  连接。转子处于该位置时,电机的 U 端被加以正电压,电机端子 V 加以负电压,电机端子 W 则被开路。这样一来,线圈 A,B,C 有图中的粗线所示的电流流过,S 极被线圈 A 励磁,N 极被线圈 B 及 C 励磁。由此各线圈与转子的永磁磁极之间产生了引力与斥力。这个状态与图 1.2(a)中的直流电机是完全一致的,但是无刷电机与直流电机的转子与定子的关系是反的,因此这次是逆时针转动。

图 1.4(b)是从图 1.4(a)的位置开始,转子逆时针转过  $60^\circ$  的示意图。转子在该位置时,电机的端子 W 上加以正电压,端子 V 则加以负电压,端子 U 开路。线圈 A,B,C 中有图中粗线所示的电流流过,S 极被线圈 A 及线圈 C 励磁,N 极被线圈 B 励磁。这样的话与图 1.4(a)一样,线圈与转子的永磁磁极之间产生了引力与斥力,转子逆时针转动。

图 1.4(c)是从图 1.4(b)的位置开始,转子逆时针转过  $60^\circ$  的示意图。转子在该位置时,电机的端子 W 上加以正电压,端子 U 则加以负电压,端子 V 开路。线圈 A,B,C 中有图中粗线所示的电流流过,N 极被线圈 A 及线圈 B 励磁,S 极被线圈 C 励磁。这样的话与图 1.4(a)一样,线圈与转子的永磁磁极之间产生了引力与斥力,转子逆时针转动。

图 1.4(d)、图 1.4(e)和图 1.4(f)的说明就此省略。

直流电机在转子转动的时候换向器与电刷能够自动地改变电流,而无刷电机的话则需要驱动放大器主动地改变电流,此时需要霍尔传感器来感应转子的位置。

## 1.4 电机的分类与直流伺服电机在分类中所处的位置

现在,电机的用途非常广泛,根据不同的用途开发了各种各样的电机,因此,电机的分类变得越来越困难。我比较了几本书中对电机的分类,没有一个是相同的。表 1.1 是笔者根据自身经验总结的电机分类及其主要的用途。

表 1.1 电机的分类及其主要用途

按动作分类	按特性分类	按构造分类	按名称分类	主要用途
旋转电机	直流电机	有刷电机 (有整流子)	DC 电机	玩具/音响
			万用电机	音响
			直流伺服电机	电动轮椅/工业机器人
		无刷电机	无机电机	工业机器人
			无刷直流电机	DVD/硬盘
	交流电机	异步电机	感应电机	冰箱/空调
		同步电机	同步电动机	钟表/水泵
			交流伺服电机	工业机器人/精密仪器
	脉冲电机	可变电抗型(VR 型)		弹珠机
		永磁型(PM 型)		钟表
		混合型(HB 型)		精密测量台/软盘
线性电机	直流电机	无整流子	音圈电机	DVD/硬盘
	交流电机	异步电机	线性感应电机	
		同步电机	线性伺服电机	精密仪器
	脉冲电机	线性脉冲电机		

### 1.4.1 直流伺服电机属于旋转电机

按照电机的动作原理区分,有旋转电机与线性电机。

旋转电机正如其名,是进行旋转运动的电机。旋转电机的最大的特征是产生旋转力即转矩的机构在旋转中被连续重复利用。因此,相对于体积和质量而言,虽然直接的驱动力较小但能高效率利用,由于其高速旋转因而可以输出较大功率,利用齿轮组进行减速则可以对应负荷输出适当的转速和驱动力。

线性电机是进行直线运动的电机。线性电机的特征是电机产生的力就是测量台或者驱动器的推进力。电机产生的力直接作用于测量台或者驱动器的话在控制上较为有利,但无法对作用于测量台或驱动器的力进行放大。鉴于上述这些特征,在选定电机时应该多加注意。

### 1.4.2 交流伺服电机是有逆变器电路的交流电机

电机按照特性来分类,有直流电机、交流电机,以及脉冲电机(步进电机)。

直流电机是使用直流电源驱动的电机。直流电机与 DC 电机在语言上是

一样的,但是说 DC 电机的时候往往指模型用的有刷直流电机。按照本书的分类,无刷直流电机也属于直流电机。

交流电机是使用交流电源驱动的电机。但是,也有不符合这个分类法的情况。交流电机中有使用逆变器电路(利用直流电产生交流电的电源电路或者有该电路的电力变换装置)的,也有使用三极管开关来制造交流波形而驱动电机的。拥有逆变器电路的电机放大器中,有采用直流电源,也有采用交流电源的。交流伺服电机是有逆变器电路的交流电机。

脉冲电机(步进电机)的电源是直流电。而且,脉冲电机与交流伺服电机的逆变器电路不同,它拥有使用三极管开关来进行电机电流转换的电路。从驱动电源来分类的话属于直流电机,脉冲电机因为构造与性质和直流电机完全不同,因此另行分类。

### 1.4.3 交流伺服电机是同步电机

交流电机一般分为异步电机和同步电机。

异步电机的转子转动的频率比定子线圈中的交流电流的频率要低一些。异步电机的代表例子是感应电机,该电机被应用于冰箱及空调中。

同步电机的转子以定子线圈电流的频率转动。对于同步电机中以固定的商用电源频率(50Hz 或者 60Hz)转动的电机,为了与交流伺服电机等其他同步电机进行区别,特别命名为同步电动机。其代表性的用途为钟表及水泵。而在同步电机中,有逆变器电路能够自由改变转子的旋转速度的称为交流伺服电机。交流伺服电机的主要用途是工业机器人及精密测量台。

### 1.4.4 交流伺服电机的特征

交流伺服电机既有优点也有缺点。

优点:

- (1) 因为没有电刷的损耗,因此不需要保养。
- (2) 没有电刷磨损粉末,因此可以在清洁的环境下使用。
- (3) 使用高性能的永磁磁极可以达到高效率/高功率。
- (4) 因为使用逆变器进行交流驱动,所以低振动/低噪声。
- (5) 反馈的旋转精度及定位性能等基本性能很高。



(6) 电机线圈因为是无心的,所以没有齿槽转矩,转矩波动也很小。

缺点:

- (1) 需要根据负荷进行伺服调整。
- (2) 电机及伺服放大器的价格很高。
- (3) 因为需要位置感应机构,所以超小型电机很难制作。

## 1.5 伺服是什么

交流伺服电机最大的特征就是伺服机构。能够被称为伺服电机的电机除了交流伺服电机以外,还有直流伺服电机及 RC 伺服电机,等等。我们来说明一下它们的不同点和特征吧。而且,我还想简单介绍一下伺服机构不可或缺的位置/速度感应机构。

### 1.5.1 伺服机构与伺服电机

伺服是使物体的位置、方位、状态等输出被控量能够跟随输入目标(或给定值)的任意变化的自动控制系统。伺服(servo)一词是由拉丁语的“奴隶”的词根“servus”衍生出来的,意思是能让电机按照下达的命令而去行动的控制系统。

### 1.5.2 直流伺服电机

直流伺服电机是能适应控制用途的直流电机。一般而言,直流伺服电机与直流电机是被区别对待的。直流伺服电机的优点和缺点整理如下。

优点:

- (1) 体积小/重量轻,而且效率高。
- (2) 能在低电压下工作,制作简易。
- (3) 利用高性能永磁磁极可以达到高效率/高功率。
- (4) 电机与伺服放大器的系统价格比较便宜。
- (5) 反馈的旋转精度及定位性能等基本性能很高。

缺点:

- (1) 必须根据负荷的需要调整伺服。
- (2) 会出现电刷磨损的粉末,无法在干净的环境下使用。

(3) 电刷会有消耗,需要保养。

(4) 会产生机械及电机的噪声。

根据控制用途而特化的电机有在 1.3 节介绍过的无心直流电机。有许多制造企业采用了这种直流伺服电机。无心直流电机的特征归纳如下:

(1) 转子的惯量比较小。

(2) 电机线圈的电抗比较小,整流性很好。

(3) 不会产生齿槽转矩。

直流伺服电机虽然有电刷与换向器接触带来的缺点,但基本性能可以匹敌交流伺服电机,用途也与之十分接近。而且,电机及伺服放大器的知识和制造技术与交流伺服电机有很多共同点,很多制造商把直流伺服电机和交流伺服电机放在一起销售。

### 1.5.3 RC 伺服

RC 伺服在广义上也是一种直流伺服电机。但是该电机的特性及利用领域较为不同,和直流伺服电机有所区别。RC 伺服电机广泛使用在无线电控制小车上,用来操纵小车的舵。为了检测出位置,采用了可变电阻,旋转角度能达到  $180^\circ$ 。

图 1.5 是 RC 伺服的系统概略图。如图 1.5(a)所示,RC 伺服由控制器及电机构成,控制器里有位置偏移检测电路、电机驱动放大器及单稳多谐振荡器电路。电机部分有直流电机和电位计。

RC 伺服的工作原理使用图 1.5(b)的时序图来说明。目标位置由外部给定,是 PWM 信号,周期是 20ms,脉宽在 1~2ms 内能够指定位置(具体内容根据 RC 伺服的制造商而不同)。

接下来是关于位置检测,根据电机内电位计的电阻值,脉宽会变化。在这里单稳多谐振荡器电路将派上用场。检测位置的脉宽与目标位置相同,在 1~2ms 内。

位置偏移检测电路是当检测位置没有达到目标的时候输出顺时针脉冲,如果检测位置通过了目标位置则输出逆时针脉冲。

电机驱动放大器根据顺时针脉冲或者逆时针脉冲的位置偏移脉宽,决定直流电机的驱动电压的极性 & 驱动时间,并驱动直流电机。在此之下,位置的反

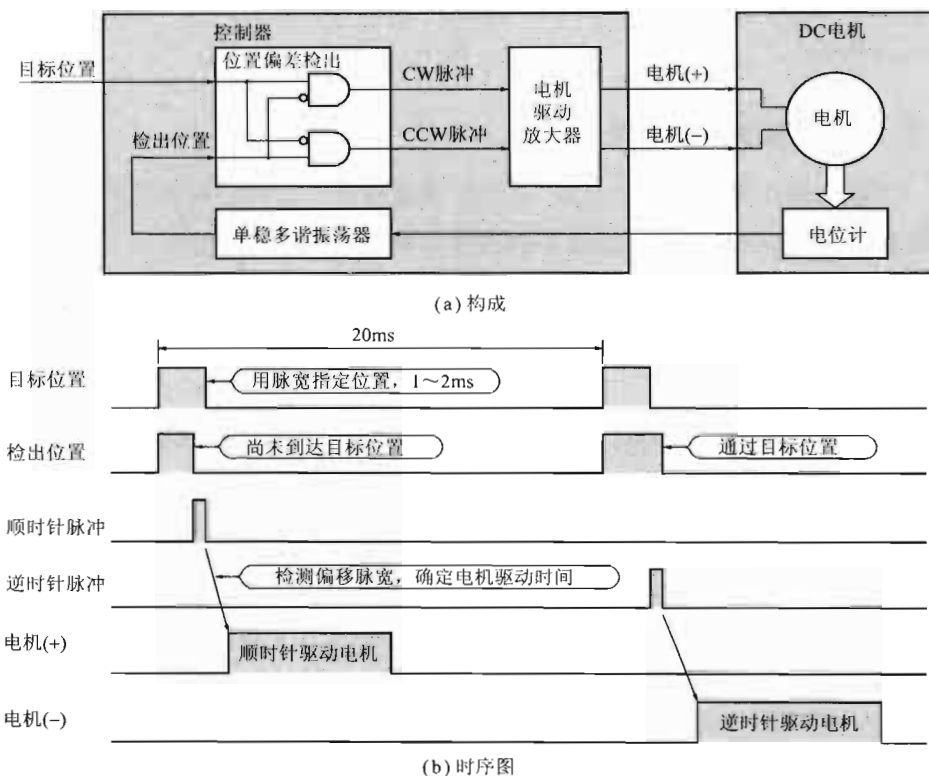


图 1.5 RC 伺服系统

馈能够收到,RC 伺服可以决定位置。再者,伺服增益的调整是由电机驱动放大器进行的。根据测量偏差脉冲,确定需要驱动多少时间。

下面把 RC 伺服的优点和缺点一一列出。

优点:

- (1) 用于玩具,所以廉价。
- (2) 设计上是为了用电池驱动,因此在低电压下能够工作。
- (3) 体积小/重量轻。

缺点:

- (1) 需要调整单稳多谐振荡器电路。
- (2) 会出现电刷磨损粉末,不能在干净的环境中使用。
- (3) 会有电刷消耗,因此需要保养。
- (4) 会产生机械/电机噪声。

(5) 伺服周期大约为 20ms, 略为粗糙, 振荡现象较为严重(交流伺服电机在 1ms 左右)。

### 1.5.4 位置检出机构

伺服机构因为是反馈式控制, 需要位置或者速度(转数)检测机构。伺服控制有硬件方式也有软件方式, 这些都非常依赖位置及速度检测机构。如果仅限于交流伺服电机的话, 交流伺服电机最擅长控制位置, 必须要有位置检测机构。下面让我介绍一下位置检测机构吧。

#### 1. 磁编码器

图 1.6 是磁编码器的概略构造图。电机的轴上附有多极的永磁磁极。这个永磁磁极不是驱动用而是编码器专用的。传感器模块的传感器有磁阻型(MR)与霍尔型两种。通过对从传感器中得来的正弦波状的二相信号(sin, cos)进行内插处理, 能得到很高分辨率的位置信息。以下总结出其特征:

- (1) 体积小。
- (2) MR 型可以实现很高的分辨率。
- (3) 霍尔型很难实现高分辨率(因为需要根据每个传感器去调整)。
- (4) 比较廉价。

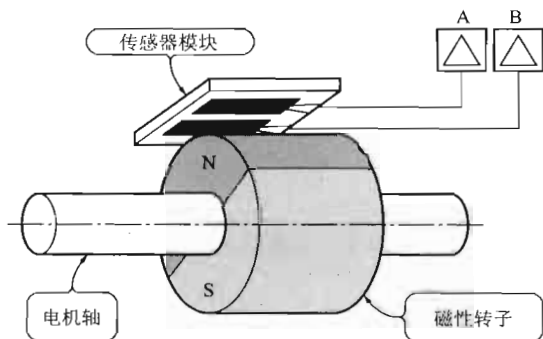


图 1.6 磁编码器的概略构造(资料提供: maxon motor)

#### 2. 光学编码器

图 1.7 是光学编码器的概略构造图。电机轴上高精度地固定了码盘, 码盘上面有许多光学的狭缝。让发光二极管(LED)的光通过检测光栅(mask)照射



在上面,再通过光电晶体管来检测出光线,因而可以把狭缝的位置转换成电信号。通过码盘,可以得到连续的脉冲列。本图显示的是增量式光学编码器,另外还有绝对式编码器。光学编码器的主要特征如下:

- (1) 能实现高分辨率。
- (2) 能准确测量位置(高精度)。
- (3) 增量式编码器相对比较便宜。
- (4) 绝对式编码器相对昂贵。

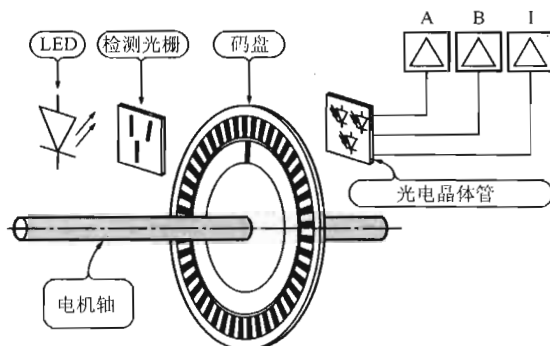


图 1.7 光学编码器的概略构造(资料提供: maxon motor)

### 3. 旋转变压器

图 1.8 是旋转变压器的概略构造图。旋转变压器转子与驱动轴直接相连,给励磁侧(一次侧)加交流电压,旋转变压器转子旋转,则相差  $90^\circ$  设置的输出侧(二次侧)可以得到  $\sin, \cos$  的信号。 $\sin, \cos$  的信号品质非常好,因而可以实现高分辨率的绝对值编码。其主要的特征如下:

- (1) 适合工业用的坚固的构造。
- (2) 没有主动的部件,很高的耐用性。

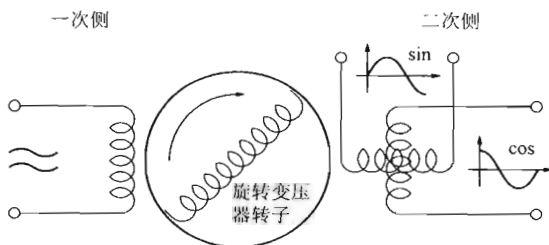


图 1.8 旋转变压器的概略构造(资料提供: maxon motor)

- (3) 能够检测转子的绝对值,不需要其他的检测器。
- (4) 伺服放大器侧需要有交流振动电路及内插处理电路。

### 1.5.5 交流伺服电机的系统构成

图 1.9 是交流伺服电机的系统构成图。交流伺服电机的系统可以分成上位控制器、伺服放大器,以及电机三部分。另外,交流伺服电机除了图 1.9 的构成以外,有的系统还在上位控制器中增加了位置反馈循环,用伺服放大器控制速度,又或者将位置检测放在电机外侧等。

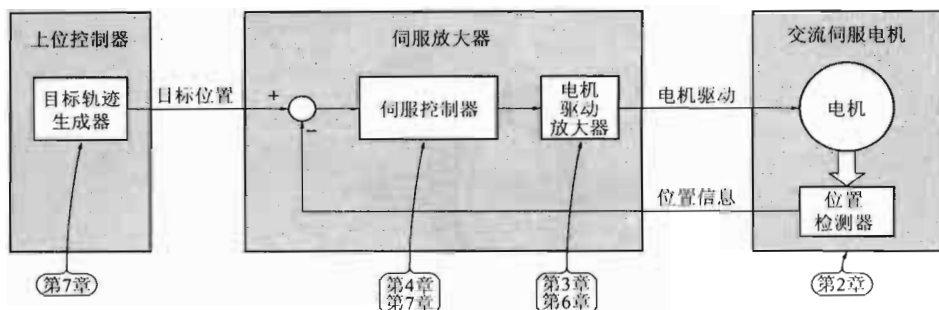


图 1.9 交流伺服电机的系统构成

#### 1. 上位控制器

产生电机的动作轨迹,向伺服放大器发出指令。这个接口一般都采用脉冲列,通过发送 1 个脉冲的指令,伺服电机和步进电机一样旋转 1 个角度。因此,上位控制器除了用于交流伺服电机外,也可以用于步进电机。接口在各个制造商之间已经统一化了,并不限于某个特定的制造商。但是本书所介绍的交流伺服驱动器已将上位控制器和伺服放大器设计在一起了,不需要单独的上位控制器。

#### 2. 伺服放大器

伺服放大器由伺服控制器与电机驱动放大器构成。交流伺服电机的伺服控制已经几乎都是数字控制,高性能的 DSP(Digital Signal Processor)或者微处理器已被广泛使用。电机驱动放大器则是采用矢量控制的三相正弦波 PWM 驱动方式。

理解以上控制技术是本书的重点课题。



### 3. 交流伺服电机

交流伺服电机是由电机与位置检测机构构成的。为了控制电机,首先需要理解电机到底是什么,是什么原理而产生力并旋转,也就是需要加深对电机的理解。

## 第 2 章

# 交流伺服电机的构造与特征

如果自己来进行交流伺服电机的驱动电路设计,必须充分理解电机的构造。当有电流流过电机的线圈时,到底有怎么样的力作用于电机上? 不理解这一点,不要说提高电机的效率,就连让电机转动都做不到。

本章虽然主要介绍交流伺服电机的构造与特征,但相同构造的无刷直流电机及 PM(Permanent Magnet)型步进电机也是有着相同的原理,为了加深理解,我们将交流伺服电机与无刷直流电机、PM 型步进电机进行对比说明。

### 2.1 产生旋转力的原理

电机是把电能转换为动能的装置。在这么多种类的电机中,转子使用永磁磁极的交流伺服电机基本上是依靠将电磁铁与永磁磁极集合,来产生连续的运动。

#### 2.1.1 有关磁铁的知识

磁铁有 S 极与 N 极,S 极与 N 极之间有引力,同极之间则会有斥力。电磁铁是通过将电流导入线圈中来产生磁场,磁场的方向可以根据电流的方向任意改变,磁场的强弱也可以根据电流的强弱改变。电机通过控制流入线圈的电流的方向与强弱,实现电机连续的运动。

## 2.1.2 实际的电机

图 2.1 显示了实际的电机上产生力的原理。定子是三相(U, V, W)Y 连接, 转子是 2 极的永磁磁极。这种构造在交流电机、无刷直流电机、PM 型步进电机等电机上较为常见。此图演示了电机的 U 相到 V 相有电流流过时, 根据转子的位置, 有怎么样的力量在作用, 导致电机旋转的过程。

转子在图 2.1(a) 中的位置时, 顺时针方向有转矩产生, 电机将旋转。图

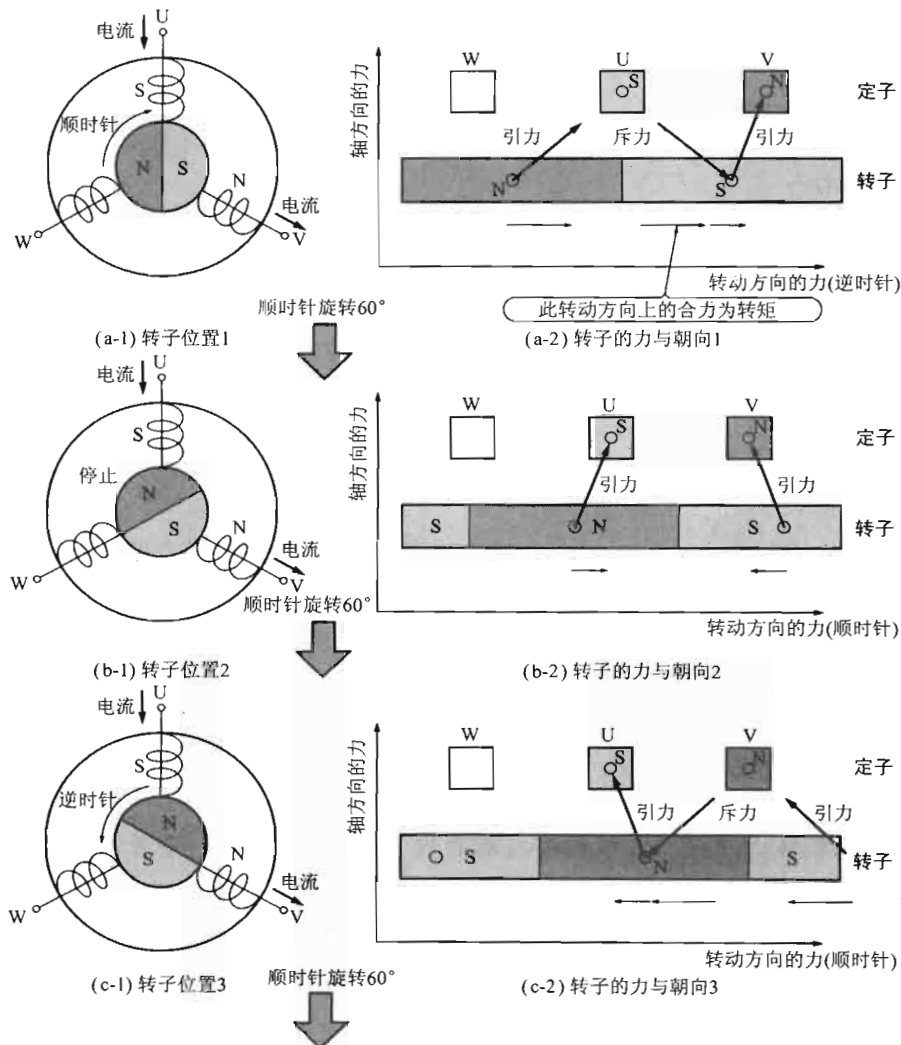
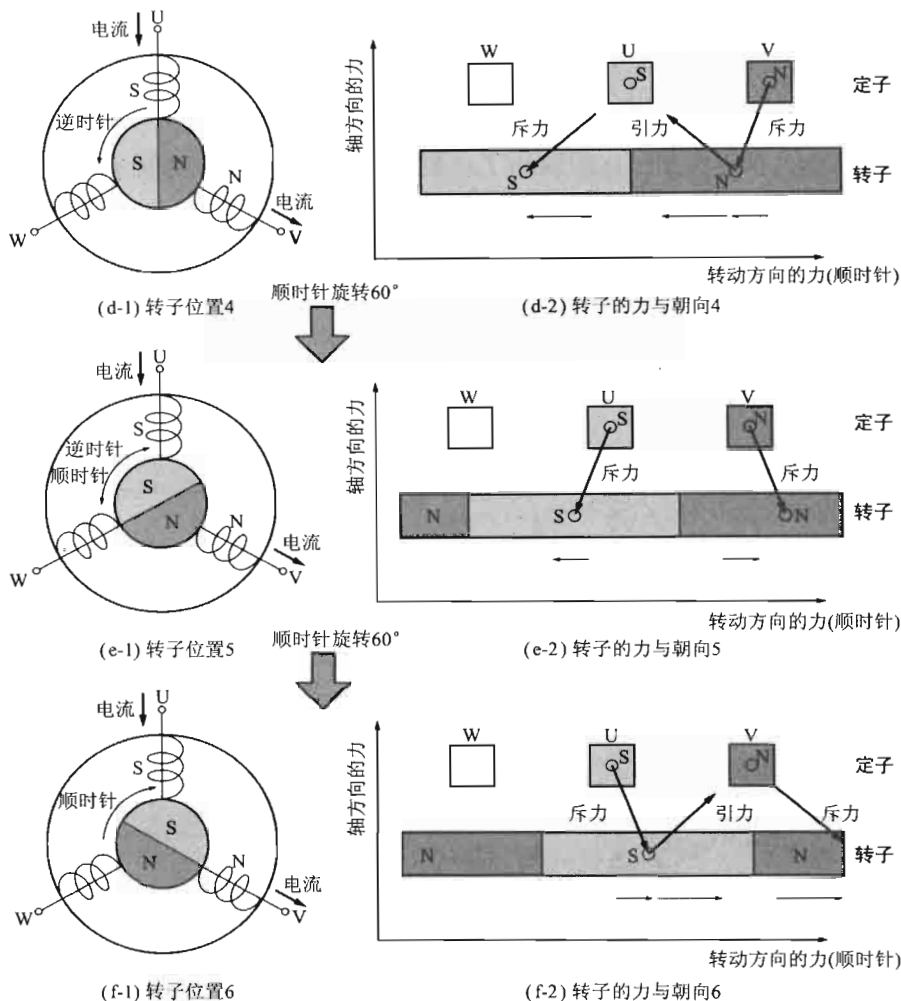


图 2.1 实际的电机中力的产生原理



续图 2.1

2.1(d)~图 2.1(f)是电机展开的时候,将力的发生用向量表示出来的图。横轴是电机的旋转力(转矩  $q$ ),纵轴是电机的轴与垂直方向的力(磁场  $d$ )。U 相线圈有磁场的引力和斥力的共同作用,产生巨大的顺时针转矩。V 相线圈是强大的引力,向量的方向向旋转轴垂直的方向倾斜,所以顺时针转矩并不大。

图 2.1(b)表示了从图 2.1(a)的位置开始,转子顺时针方向转了  $60^\circ$  的位置。这个转子位置不会有转矩,转动将停止。对照图 2.1(d)~图 2.1(f),U 相和 V 相有着强大的引力,但因为与旋转轴垂直,没有太大的转矩成分。而且,U

相的转矩与 V 相的转矩是反方向,这两个转矩将互相抵消。

图 2.1(c)是从图 2.1(b)的位置开始,转子顺时针方向旋转了  $60^\circ$  后的位置。这个转子的位置会产生逆时针方向的转矩,转子将会旋转。由向量图分析可以看出,这次是 V 相产生了很大的逆时针转矩,U 相线圈没有产生多大的逆时针转矩。

图 2.1(d)是从图 2.1(c)的位置开始,转子顺时针方向旋转了  $60^\circ$  的位置。这个位置下电机将会向逆时针方向旋转。由向量图可以看出,U 相线圈有很大的逆时针转矩。

图 2.1(e)是从图 2.1(d)的位置开始,转子顺时针方向旋转了  $60^\circ$  后的位置。这个位置下电机将向顺时针或者逆时针方向旋转。由向量图可以了解,U 相和 V 相都有很强的斥力,但是因为与旋转轴垂直,并没有太大的转矩。与图 2.1(b)相同,U 相线圈和 V 相线圈的转矩是相反的,将会互相抵消。但是,这个位置下转子不会停止。其理由将在下一节说明。

图 2.1(f)表示的是从图 2.1(e)的位置开始,转子顺时针方向旋转了  $60^\circ$  后的位置。这个位置下转子会向顺时针方向旋转,由向量图可以知道,V 相线圈有很大的顺时针转矩产生。

### 2.1.3 电机的恢复转矩和平衡点

图 2.2 是把转子的位置和转矩的关系表示出来的图。图的横轴是将转子的位置和之前的图一样,标注了(a)~(f)。图的纵轴是转矩的大小。

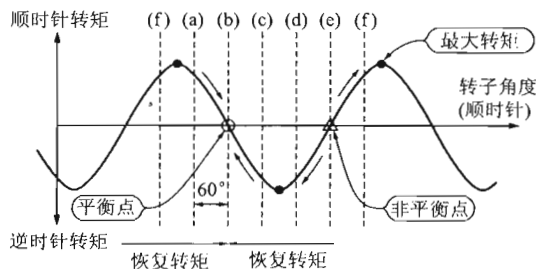


图 2.2 转子位置与转矩的关系

从图中 2.2(b)的位置开始,转子逆时针偏离的时候会产生顺时针转矩;反过来,从图中 2.2(b)的位置开始,顺时针方向偏离的时候会产生逆时针转矩。这个位置会有恢复转矩使得转子回到图 2.2(b)的位置。其结果就是转子将停

止,这个位置称为平衡点。

图 2.2(e)的转子位置和图 2.2(b)一样没有转矩,但是这个位置电机不会停止。这是因为转子逆时针方向偏离的时候会产生逆时针转矩;反过来,顺时针方向偏离时会产生顺时针转矩。也就是说,这个位置不能成为平衡点,只要稍微破坏平衡,转子就会开始转动。

## 2.2 旋转力与步进驱动

前节说明了当电流从 U 相流向 V 相时,根据转子位置,旋转转矩变化的原理。向其他相流入电流时也会同样导致旋转转矩发生变化。图 2.3 就是将它们总结起来的图。对于三相(U,V,W)Y 连接的线圈,二相励磁(向两个线圈通电流)的方法有 6 种。下面针对这 6 种情况,将转子的位置及转矩的发生状况进行总结。图 2.3 中(a)~(f)的转子位置与图 2.1 相同。

### 2.2.1 制造旋转磁场的方法

请看图 2.3 的最上面的(1)。电机电流从 U 相流向 V 相,电磁铁转子会因为恢复转矩而在(b)位置停止。接下来电机电流从(2)的 U 相切换到 W 相。电磁铁转子还在(b)位置,顺时针转矩产生作用,转子继续向顺时针方向旋转。转子到达(c)位置时又会有恢复转矩,将停止在(c)位置。然后接下来将电流切换成(3)中的 V 相到 W 相。此时转子还在(c)位置,顺时针转矩起作用,转子将向顺时针方向旋转。转子到达(d)位置,因为恢复转矩而停止在(d)位置。

像这样,电机电流从(1)开始按顺序切换到(6),转子就一步步地向顺时针方向旋转。想要让它向逆时针方向旋转,则只要从(6)到(1)的顺序流入电流即可。

虽然定子为三相(U,V,W)Y 连接的线圈、转子是 2 极的永磁磁极的步进电机较难见到,但它也算是正统的步进电机,每一步的角度是  $60^\circ$ 。

### 2.2.2 使其连续的旋转

2.1 节我们说电机电流是慢慢切换,让转子一步步地旋转。那么让我们考虑一下如果在短时间内切换电流,转子将会如何转动吧。

电机电流在(1)的情况下从 U 相流向 V 相。此时转子会因为恢复转矩在



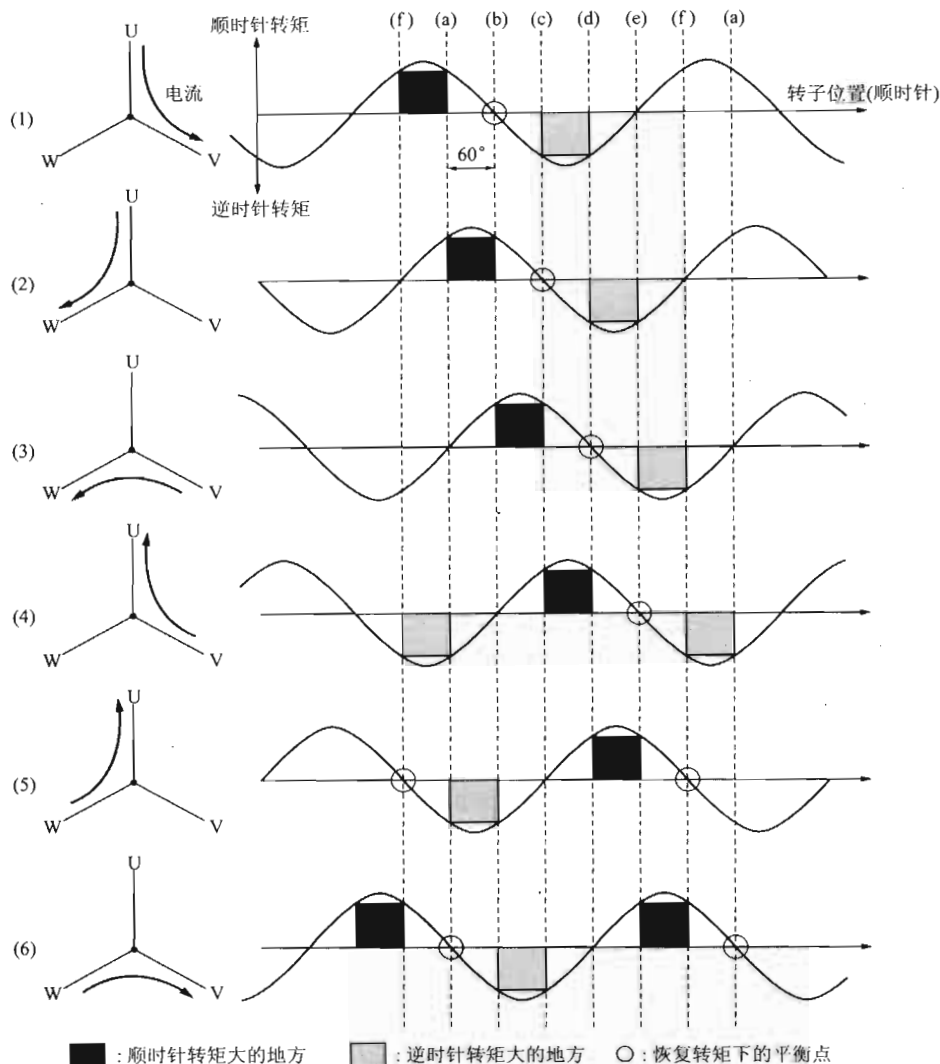


图 2.3 各相电流下的转子位置与转矩的关系

(b)位置停止。接下来,电机电流按照(2)的情况从 U 相切换到 W 相。此时转子还在(b)位置,在顺时针转矩的作用下,转子向顺时针方向转动。到此为止和上一节一样,转子有转动惯量,也有摩擦力,从(b)位置到(c)位置无法瞬间移动,在转子到达(c)位置之前,或者到达了(c)位置但还有旋转速度的时候,将电机的电流按照(3)的情况从 V 相切换到 W 相。

像这样,转子保持着速度,一个个地切换电机电流就能使转子连续转动。

让转子连续转动需要转矩,步进电机因为转子相对于旋转磁场会有所延迟,而得以保持转矩。这个转子的延迟角度称之为负荷角,惯性和摩擦力增大时,负荷角也将增大。

请看看图 2.3 最大转矩发生时的负荷角是  $90^\circ$ 。负荷角超过  $90^\circ$  时,惯性力和摩擦力将要高于电机的最大转矩,步进电机的转子将跟不上旋转磁场,这种现象被称为失步。

### 2.2.3 决定位置机构的特征

当步进电机转子到达步进临界点的位置(对于图 2.3 中的(1)来说,就是(b)位置)的时候将没有转矩。只有转子的位置产生偏移才会有转矩。如果电机有外力作用,会在与外力平衡的地方停止。因此,如果电机有外力作用的话,一定会发生位置偏移。

### 2.2.4 控制位置

图 2.4 是一般性的步进电机的控制器的构成图。步进电机的位置指示是依靠脉冲来传达的。1 个逆时针脉冲会使电机逆时针方向旋转 1 步。1 个顺时针脉冲则会使电机顺时针方向旋转 1 步。步进电机不需要反馈,其系统构成非常简单。与图 2.5 中的伺服电机的控制器对比就一目了然了。

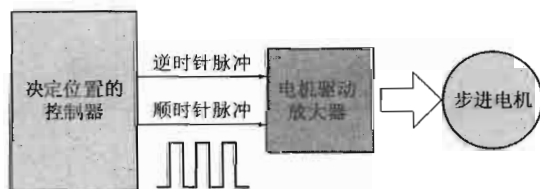


图 2.4 步进电机控制器的全体结构

让电机启动的时候,会产生惯性力和摩擦力。惯性力同加速度成正比关系,而摩擦力同速度成正比关系。当步进电机启动以后,会受到加速度的制约,惯性力不能超过电机的最大转矩。因此,对于决定步进电机位置的控制器来说,如图 2.6 所示,具有加速和减速控制的功能。

基于这些功能,对电机启动及制动时的加速度进行控制,可以避免失步的产生。

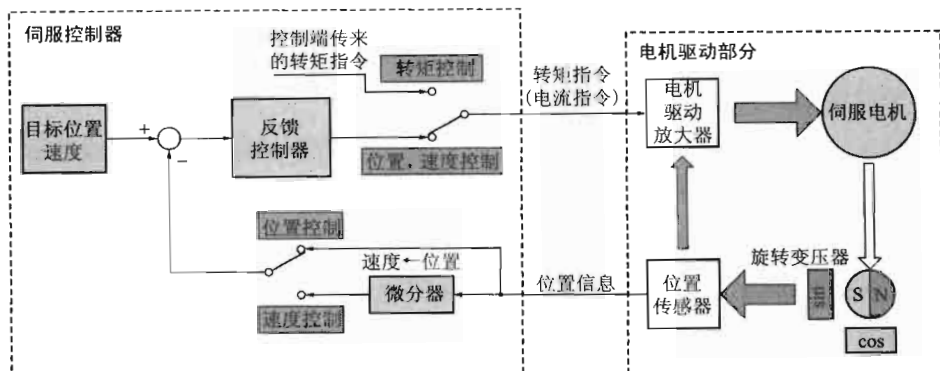


图 2.5 伺服电机控制器的全体构成

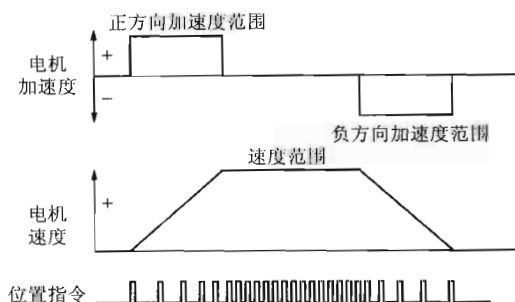


图 2.6 加速控制以及减速控制

### 2.2.5 控制速度

步进电机的速度控制如图 2.6 所示,是由位置指令脉冲的频率决定的。也就是说,由位置指令脉冲的频率对电机的旋转速度进行控制。

## 2.3 无刷直流电机驱动

下面我们从图 2.3 中各相位电流决定的转子位置及转矩的关系图开始,对无刷直流电机的动作进行解说。

### 2.3.1 让电机不停止地转动

我们在 2.2 节给大家解说过了,当电机电流同相流过时,转子会停止转动。那么,如果让转子一直不停止的话,应该怎么做呢? 我们这里试着考虑一下。

请看图 2.3 最上边的(1),转子位置(f)~(a)之间有很大的顺时针转矩产生。图 2.3(2)中表明转子(a)和(b)之间有很大的顺时针转矩产生。图 2.3(3)中表明转子位置(b)~(c)之间有很大的顺时针转矩产生。

对于(1)~(6)的各状态来说,顺时针方向产生转矩的转子位置可以很容易理解,在图中对应的是“■”,而逆时针转矩是以“□”表示的。

要想使电机不停地转动,首先要查看转子的位置,根据转子的位置会产生顺时针转矩或者逆时针转矩,这样电机中流过电流的线圈相位就能确定了。对于无刷直流电机来说,就是通过这样的反复过程来驱动电机的。

### 2.3.2 不会产生失步

从无刷直流电机的驱动方式来说,理论上是不会产生失步的。理由前文也说过,通过检测转子的位置,由转子的位置决定电机电流的相位。举例来说,如果负荷过重电机渐渐地转不动的时候,电机的旋转磁场(电机电流的切换)是不会顺利地进行反转的。

### 2.3.3 检测转子的位置

检测转子位置的方法有好多种,在这里我们给大家分别描述各自的特征。

#### 1. 霍尔元件及霍尔 IC 的使用方法

霍尔元件是一种基于霍尔效应的磁传感器,用它可以检测磁场及其变化。N 极有正的输出,S 极有负的输出。霍尔元件单体的输出有微小的差动信号,需要由差动放大器放大后再使用。但是,因为霍尔元件是很精密的元件,容易受到温度及转子磁铁的影响,在很多情况下,我们使用集成了霍尔元件、放大器及比较器的霍尔 IC 芯片。

图 2.7 中 3 个霍尔 IC( $S_A, S_B, S_C$ )以  $120^\circ$  的间隔配置。霍尔 IC 同电机线圈的相对位置在逆时针方向有  $60^\circ$  的差。如果转子的位置如图 2.7 所示顺时针方向旋转的话,就得到了图 2.7(b)所示的数字信号。图 2.7 中(a)~(f)的转子位置与同图 2.1 相同。对输出  $S_A, S_B, S_C$  进行解码的话,就可以得到电机步进  $60^\circ$  的状态,基于这个状态切换电机的电流流向。

#### 2. 感应电动势的利用方法(无刷方式)

因为三相 Y 连接的无刷直流电机通常的驱动方式是二相励磁,所以通常

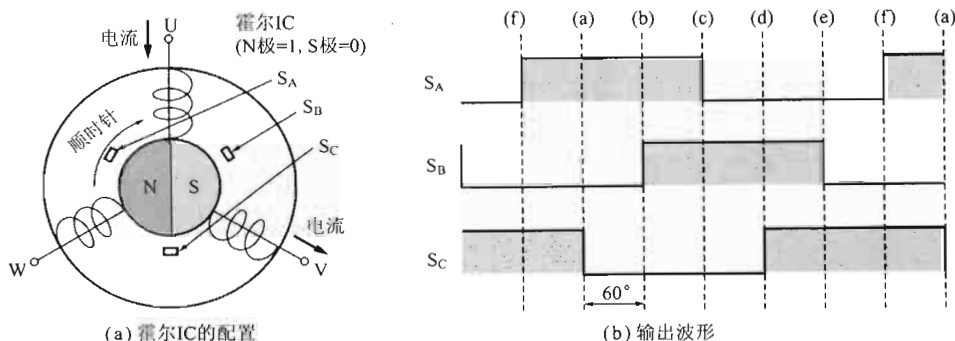


图 2.7 基于转子位置的霍尔 IC 的输出

会有一相余留下来。电机旋转一定的角度时,就会产生感应电动势,测定余下的一个线圈上的端子电压的话,就可以确定转子的位置,也就是所谓无刷方式。虽然到这里为止讲的东西都很简单,但是实行起来很难。虽然各大公司都在贩卖无刷方式的控制 IC 芯片,并且很多公司用这些芯片进行控制,但是从笔者自己的经验来讲,根据公司的不同,性能也有相当大的差别。因为对于这些差别的详细叙述并不是本书的目的,所以这里我们只给大家讲一下无刷方式的基本特征。

无刷方式的优点是,并不需要特别的位置传感器,所以成本可以削减很多,因此,常用于成本要求较为严格的硬盘等的主轴电机。

无刷方式的缺点是,因为使用了感应电动势,在电机启动时因为没有感应电动势,只能使用开环的旋转控制方式。也就是说,当电机启动时,要想像步进电机那样进行旋转控制,有可能会失步。

### 3. 旋转变压器

图 2.8 是旋转变压器的构成图。一次侧(励磁端)加交流电压,旋转变压器转子与驱动转子直接连接,旋转变压器转子如果转动的话,每  $90^\circ$  配置的二次侧(输出端)就得到了  $\sin$  和  $\cos$  信号。如果把从旋转变压器处得到的  $\sin$  和  $\cos$  信号如图 2.9 进行内插处理的话,就可以得到转子位置(角)。旋转变压器由于没有使用半导体元件,所以对环境的适应性很强。此外,由于旋转变压器比较坚固,所以对于冲击也有很强的抵抗力。

由于  $\sin$  和  $\cos$  信号的品质很好,所以作为高分辨率的绝对编码器使用的场合很多。这种情况下,一般备有作为电机电流切换用的霍尔传感器及伺服控

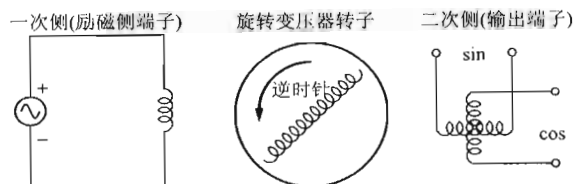


图 2.8 旋转变压器的构成

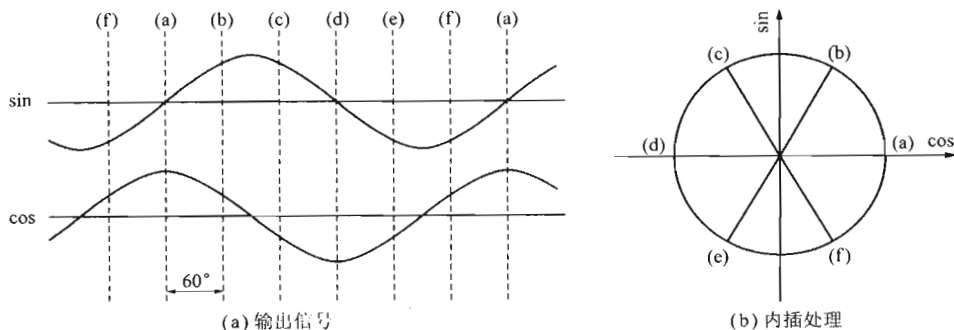


图 2.9 旋转变压器输出信号及内插处理

制用的位置传感器。其缺点是，由于进行了内插处理，所以需要 RD 转换器或者高速高分辨率的 A-D 转换器或者可以高速计算反正切函数的微处理器。

### 2.3.4 控制速度

无刷直流电机的特性和普通直流电机一样。基于这点，如果控制加在电机上的电压的话，某种程度上来说就可以控制电机的转速了。为什么这里要加上“某种程度上”呢，因为加在电机上的电压即使一样，根据负荷的不同转速也会不同。而所谓的速度控制，是基于速度传感器及上位控制器（伺服及 PLL）的反馈控制。

图 2.5 是伺服控制器的全体构成图，因为使用了旋转变压器，所以可以测量位置信息。也就是说，速度控制是通过对位置信息进行微分计算然后算出旋转速度，最后通过反馈控制来进行的。

无刷直流电机和步进电机有许多不同种类，且电机都商品化了，所以不能一概地讲哪个性能更好，单从动作原理上来讲，它们有以下不同点。

步进电机因为有失步的危险，所以电机转动过程中，电机及驱动电路中会流过允许的最大电流。这样，基于转子的位置，旋转转矩会有很大的不同，这样

就产生了振动。

另一方面,因为无刷直流电机没有失步的危险,所以电机旋转的时候,只流过和转速相对应的电流。此外,因为旋转转矩不会根据转子的位置产生很大的相位差,因此,无刷直流电机很少产生振动。

### 2.3.5 控制位置

无刷直流电机是通过改变流过电流的相位来产生旋转转矩的。电机中电流流过所以电机才能旋转。但是,只有电机及驱动回路的话并不能控制位置。位置的控制是通过位置传感器及基于上位控制器的反馈控制来进行的。前文提到的伺服控制器是通过使用位置传感器的旋转变压器检测到位置信息的。

无刷直流电机与步进电机在位置控制方面有所不同,同速度控制一样,由于两者都有很多的种类并且很多都已经形成了不同的商品,所以并不能一概而论地讲哪种更加优秀。单从动作原理上来讲,有以下的不同点。

当步进电机上作用外力时,转子就会产生转矩,与外力保持平衡,而步进电机也会在该平衡位置停止下来,所以,电机上不论加上什么样的外力时,必然会产生位置偏移,这种偏移不止是在有外力的情况下产生,即使在没有外力的情况下也会经常发生。

另一方面,无刷直流电机是当有外力作用时,虽然会一时地产生位置偏移,但是由于上位伺服控制器有关于位置偏差的积分反馈功能,所以可以修正位置偏差。但是,这只是外力的频率对于上位伺服控制器的伺服频域过低的情况。如果外力的频率在伺服的频域附近的话,振动就会被放大。

## 2.4 交流电机驱动(正弦波驱动)

“交流(伺服)电机同无刷直流电机的区别到底在哪里呢?”笔者经常听到这样的疑问。其实两者的基本构造是相同的,从驱动方式来说,不管是矩形波驱动还是正弦波驱动,两者都是基本相似的。事实上,就笔者而言,在交流电机上使用过矩形波驱动的方式,在无刷直流电机上也使用过正弦波驱动的方式。现在,不管是哪种电机都是采用正弦波驱动的方式来进行的。这些相关的细节我们将在下章详细解说。

回到我们前面的话题,交流电机同无刷直流电机的不同点是,交流电机是

以正弦波驱动为前提设计的电机,而无刷直流电机是以矩形波驱动为前提设计的电机。从结果来看,对于交流电机来说,无刷直流电机为了抑制齿槽转矩及转矩波动而有多槽、多极等多种电机。另外,两者有相同用途的例子也很多,更进一步来说,无刷直流电机中以恒速控制为目的的步进电机有很多。

本章中提到的交流电机是伺服电机。交流风扇与感应电机的电机的构成及驱动方法完全不同,在本章中我们不会接触。

### 2.4.1 交流伺服电机驱动(正弦波驱动)的电流分配

图 2.10 所示是在三相正弦波驱动下,产生转矩时各相的电流分配示意图。

图 2.10(a)的转子的位置,U 相电流正向峰值的时候顺时针转矩最大。电流的大小是由箭头的粗细来表现的。V 相、W 相中 U 相一半的电流反方向流过。

图 2.10(b)的转子的位置,W 相电流反向峰值的时候顺时针转矩最大。U 相、V 相中 W 相一半的电流正方向流过。

图 2.10(c)的转子的位置,V 相电流正向峰值的时候顺时针转矩最大。U

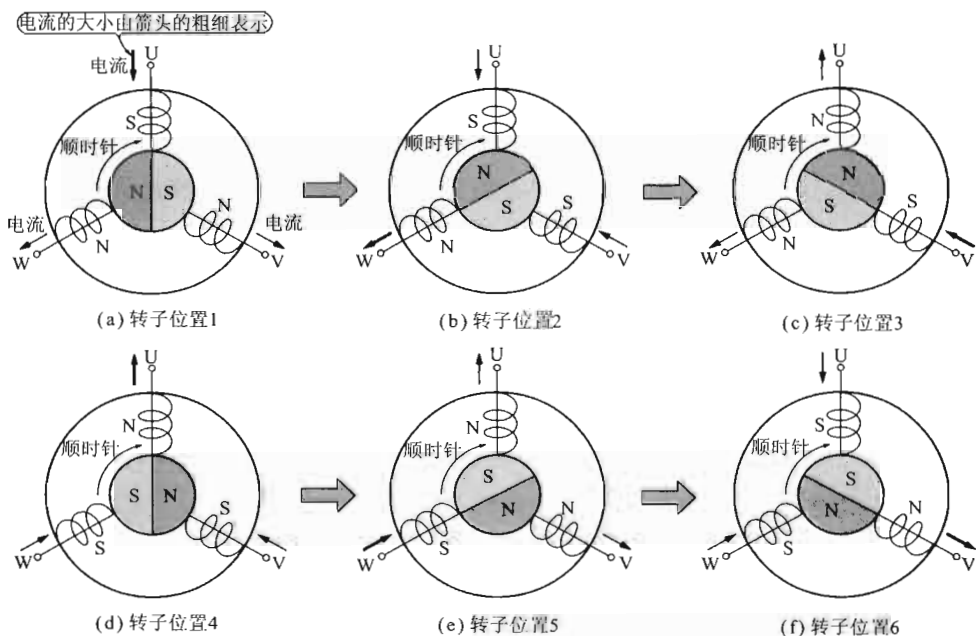


图 2.10 转子位置及三相正弦波驱动



相、W 相中 V 相一半的电流反方向流过。

图 2.10(d)的转子的位置,U 相电流负向峰值的时候顺时针转矩最大。U 相、W 相中 U 相一半的电流正方向流过。

图 2.10(e)的转子的位置,W 相电流正向峰值的时候顺时针转矩最大。U 相、V 相中 W 相一半的电流反方向流过。

图 2.10(f)的转子的位置,V 相电流反向峰值的时候顺时针转矩最大。U 相、W 相中 V 相一半的电流正方向流过。

图 2.11 是这些转子位置和各相电流的图例化。交流电机驱动时,其电流波形是三相交流。图 2.11 中表示了顺时针转矩产生时候的情况,同时也是逆时针转矩方向各层电流  $180^\circ$  反转以后的波形。

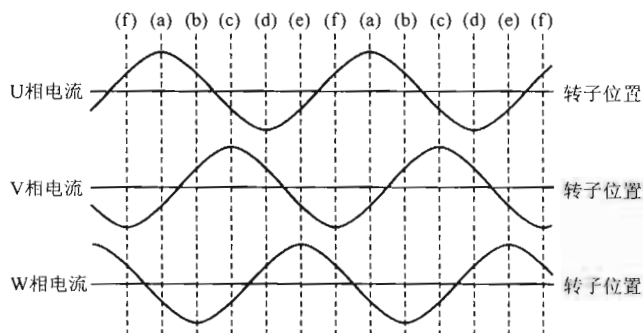


图 2.11 转子位置及三相正弦波的相位

## 2.4.2 交流伺服电机驱动(正弦波驱动)的转子位置检测

无刷直流电机驱动是通过检测六步的转子(2 极电机时是 60 步)的位置来实现的。交流电机驱动时因为是产生连续正弦波,能够检测出很细微的转子位置变动。这里我们介绍一下代表性的转子位置检测方法。

### 1. 使用霍尔传感器和增量式编码器的方法

当电源断电时,增量式编码器就会丢失位置信息,因此电机的位置也会丢失。所以,使用霍尔传感器初始化增量式编码器从而求出电机的位置。

### 2. 基于旋转变压器的绝对式编码器方法

基于旋转变压器的内插处理,能够细微地分割电机的位置,这些被分割的位置就是绝对式编码器。

### 2.4.3 很容易实现高旋转/高效率的电机驱动

#### 无刷直流电机驱动

无刷直流电机驱动的最大的特征是,即使是电压控制也能实现高转速/高效率的电机驱动。这里我们说明一下决定这种特征的理由。

#### 1. 基于转子位置的强制电流相位转换

现在为止我们已经说了好多次了,因为基于转子位置的强制电流相位转换,所以高转速时电机的效率也基本上不会降下来。

#### 2. 即使是矩形波电压驱动,电机电流也会连续流动

在这里稍稍说明一下,比较难理解的理论将要登场了。图 2.12 是安培右手定则。图 2.12 中电流从外向内流动的时候,磁通在导体的周围产生右旋转。磁通实质上不喜欢这样的状态变化,所以会对电流的变化产生妨碍作用。

图 2.13 是线圈的电压与电流的关系。线圈的电流表示了电压的积分特性,所以线圈中的电流很难流动,一旦流过后,电流不会很快消失。

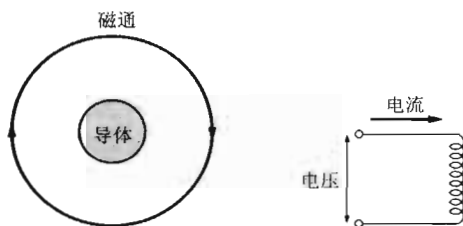


图 2.12 安培右手定则

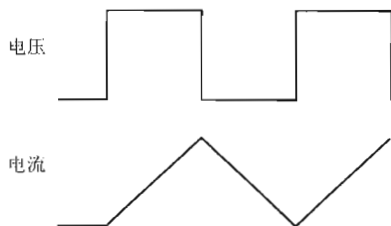


图 2.13 加在线圈上的电压和电流的关系

如前面所说的,“基于转子的位置强制电流相位转换”这样的事情会发生吗?实际上是完全会发生的。

图 2.14(a)中,一个磁体中绕了两个线圈,各自的线圈上连接了直流电源、电阻及开关。

如图 2.14(b)所示,一旦开关 1 开了以后,左边的线圈上就加上了电压,渐渐地流过电流 1。当电流流过后就会产生磁通,该磁通在磁体的内部转动。磁通与电一样具有通道(这里我们称为磁路通道),因为磁体的透磁率在空气中会达到百倍,所以磁通基本上都会通过磁体。

接下来,关闭开关 1,打开开关 2。这时候,因为回路变成了开路,所以电流 1 消失了。另一方面,开关 2 回路关闭,电流 2 流过时,如刚才说过的那样,因为

磁通具有不喜欢这样的状态变化的性质,这时候电流2不是0,而是接着刚才电流1的情况。这样一来,磁回路中没有变化的磁通,电流1加上电流2就形成了连续流动的电流。

无刷直流电机驱动也有同样的特征,因为各相电流瞬间进行了转化,综合的电机电流连续流动,所以不会产生高速旋转时的转矩急速回落。

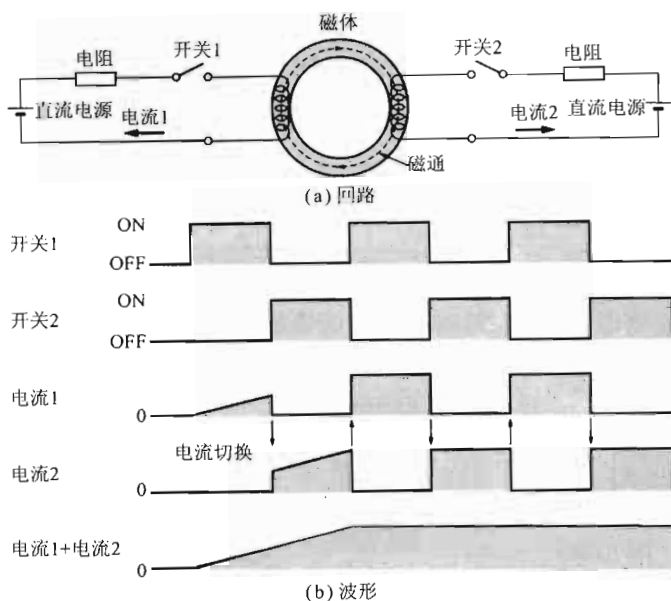


图 2.14 线圈上电流切换方向的情形

#### 2.4.4 很难实现高旋转/高效率的电机驱动的交流伺服电机驱动

交流伺服电机驱动的场所,如果仔细研究转子位置的话,就会发现,产生转矩的电流相位会驱动电机旋转。只看这点的话,基于无刷直流电机驱动的电机的效率也貌似很好,但实际上若要提高基于无刷直流电机驱动的电机效率的话,需要花费相当大的代价。

##### 1. 电机是基于电流运动的

电机从根本来说,是基于电流而动作的。交流伺服电机驱动由于使用了交流波形,所以产生了相位的概念。有电阻负荷的话,电压相位和电流相位就会一致,所以作为电压控制的话没有问题,但是作为电机的负荷来讲,电压和电

流并不一致。此外,基于电机的负荷及旋转速度,相位差也在发生变化,因此对交流伺服电机驱动进行电压控制的话,我们也不能指望电机的高效率化。

## 2. 转矩控制是有必要的

虽然控制电机的时候电流控制和转矩控制作为同样对待的例子很多,但是严格来说并不一样。电机中电流流过的时候,使电机运动的力包括旋转方向上的力(转矩  $q$ )及与轴垂直方向上的力(励磁  $d$ )。

因此,电流恒定场合控制电机时,转矩不一定恒定。对于交流伺服电机驱动来说,如果要实现高转速/高效率的电机驱动的话,控制转矩是必需条件。

## 3. 基于恒定电流控制的转矩控制方法

图 2.15 是基于恒定电流电路的转矩控制放大器的构成图。在这里我们用三相正弦波发生器产生的 U 相目标电流、V 相目标电流与其检出电流进行比较。目标电流是以图 2.11 的波形为基础做成的。

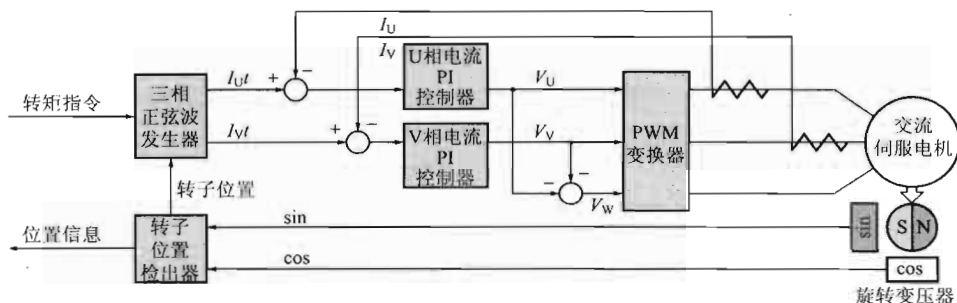


图 2.15 基于恒定电流电路的转矩控制放大器

这种方式的特征是把驱动电机的交流电源直接进行反馈,因此电机转速一旦进入高速旋转领域后电流的频率就会变高。此外,对于电机线圈来说,因为会产生很大的感应电动势,所以电路中的电流回路增益不足,从而一旦电流偏差变大,遵照指令的电流就会不再流动。为了避免这种现象发生,在可能的范围内电流的反馈回路的频率特性需要尽可能得高。

在笔者的经验看来,如果电机的旋转速度在 6000r/min 以上,要想实现和无刷直流电机驱动同等或者以上的效率,有必要按照图 2.15 那样搭建电路。并且,这种场合下不是使用 PWM 驱动方式,而是采用线性放大器的方式。

基于恒定电流控制的转矩控制在电机的转速为 6000r/min 以上时会发生驱动转矩低下的情况,这是因为在软件控制及 PWM 控制下,电流放大器的频

带只能延伸到 300~500Hz。如果是硬件控制的话,电流放大器-3dB 带宽如果能扩展到 10~100kHz 的话就没有问题了。

这就是恒定电流控制的特性。恒定电流控制时各线圈的电阻、电感及感应电动势的波动会被吸收,各相电流按照指令流过电路,最终实现最低振幅/高效率的电机控制。

#### 4. 基于矢量控制的转矩控制方法

矢量控制是最初为了提高感应电机(不使用永久磁铁的交流电机)的效率而考虑出来的。图 2.16 就是使用了矢量控制的交流伺服电机。因为交流伺服电机的转子是永久磁铁,磁场是由永久磁铁产生的,因此磁场同永久磁铁一块旋转。从与这个旋转的磁场平行的线圈(转子)出来的磁通,即使磁场变强或者变弱也不能成为旋转力,因此图 2.16 的磁场指令( $i_d$ )是零。关于矢量控制的详细说明会在第 6 章详细说明,这里只是简单描述一下特征。

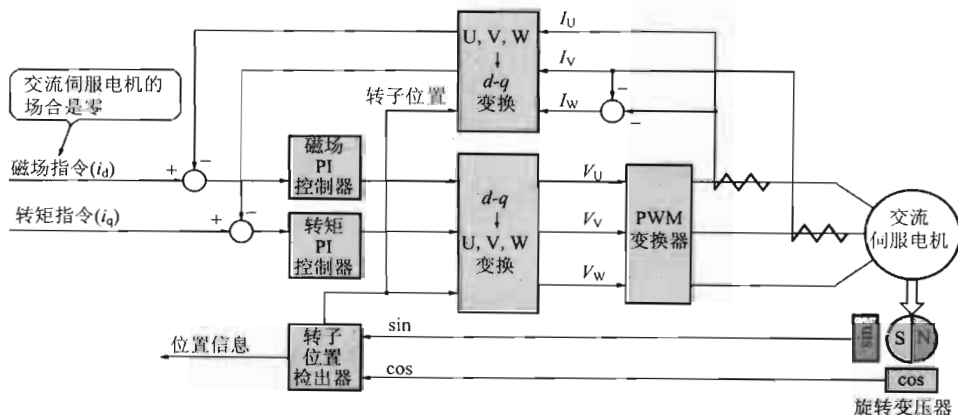


图 2.16 基于矢量控制的转矩控制放大器

矢量控制根据坐标变换( $\alpha\beta$ 变换,  $d-q$ 变换), 可以进行直流控制, 因此从低旋转区到高旋转区具有修正效果, 依据指令的电流也能流过。矢量控制一般来说需要基于高性能 CPU 的运算处理, 实际测试的结果是 32MHz 的 H8S/2000 的 CPU 可以进行演算控制, 可以实现非常高效率的交流伺服电机驱动。

矢量控制因为是各相转矩电流合计的控制, 所以会因各线圈内的电阻、电感及感应电动势波动, 而使电流的分配变得不稳定。因此, 从电机的振动特性来讲, 这种控制方式稍逊于恒定电流控制。

## 2.5 基于正弦波驱动的微步进驱动

2.2 节中说明了基于矩形波驱动的步进电机驱动方法,在这里我们对基于三相正弦波驱动的微步进驱动方法进行解说。同现在为止说明过的一样,还是以三相2极电机为例进行说明。

### 2.5.1 三相正弦波驱动的稳定点

同交流伺服电机驱动一样,我们试着用三相正弦波对电机进行驱动。但是,这次我们不用理会转子的位置。图2.17(a)是U相电流正向峰值的场合,V相和W相中U相一半的电流反方向流过。因为这个转子的位置是使顺时针转矩最大,所以转子向顺时针方向旋转。图2.17(b)是电流分配时转子位置的稳定时刻。像这样三相正弦波驱动与矩形波驱动一样,转子有一个稳定点。

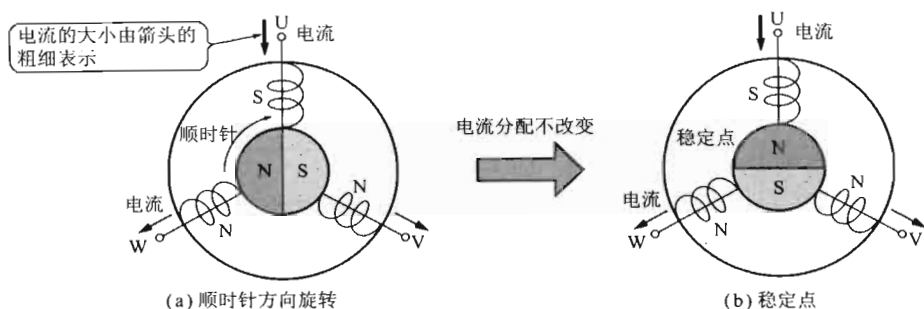


图 2.17 三相正弦波驱动时的稳定点

因为转子的稳定点是为了确保各相来的转矩保持平衡的点,所以三相正弦波的相位稍微起前一步的话,转子的稳定点也会移动,这点读者应该很容易理解吧。对于微步进驱动来说,由于驱动波形的正弦波被一步一步分割了,所以我们也将这种方式看作转子的步进角被细微分割的驱动方式。

### 2.5.2 转子的步进角不总是正确的

微步进驱动方式因为是电气方式分割的,不能完全确保转子的步进角是正确的。步进电机有 PM(Permanent Magnet)型、VR(Variable Reluctance)型、HB(Hybrid)型,只有转子使用永久磁铁的 PM 型,因为其磁场的分布是正弦波状的,所以转子的步进角是正确的。

### 2.5.3 抑制电机振动的效果

微步进驱动方式中如果把步进角细细地分开的话就会变成正弦波驱动。所以,电机转动的时候具有振动的抑制效果。电机的步进角如果细分的话,因为需要很多的脉冲指令,所以步进角不变,仅驱动波形变成正弦波状,从而抑制振动的产生。

## 2.6 有刷直流电机

在本章的最后我们加一些简单的对有刷直流电机的说明。如果是从开始一直读到这里,那么就应该很容易理解有刷直流电机的动作原理。

图 2.18(a)是三绕组直流电机的动作原理图。无刷直流电机的定子和转子关系是相反的,定子是永久磁铁,转子是电磁铁。

对于三个线圈来说,电刷和换向器来回切换。如图 2.18(b)所示,为了方便理解线圈中的电流,将线圈与换向器单独表示。上侧线圈流过电流成为 S 极。下侧两个线圈流过电流成为 N 极。下侧两个线圈对于转子来说是串联连接的,电流只有一半。上侧线圈会产生斥力和引力,在顺时针方向有很大的转矩产生。左下方的线圈中有引力动作,右下方的线圈有斥力动作,同时产生了顺时针转矩。这样,直流电机就可以考虑为不理睬转子在什么样的位置都会产

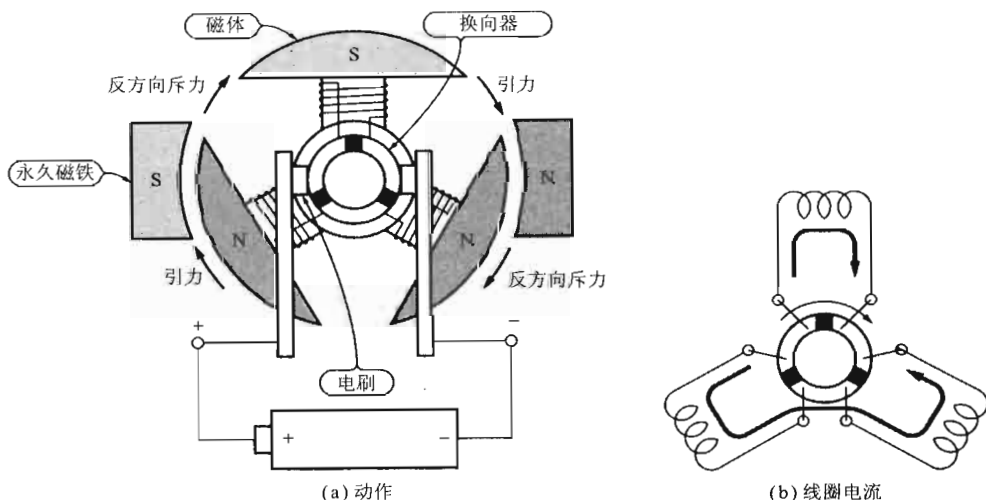


图 2.18 有刷直流电机的动作原理

生旋转转矩。

直流电机的特征如下：

- (1) 电机的效率很高。
- (2) 不用检测转子的位置,所以驱动回路很简单。
- (3) 电刷同整流子由于有摩擦所以需要经常维护。
- (4) 有很大的噪声。
- (5) 转动声音很大。



# 第 3 章

## 电机驱动电路的基础

电机的种类繁多,而且其驱动方法也是在考虑过各种各样的方式之后,才被实用化的,要把它们全部介绍出来是不可能的。因此,本章主要以笔者有过设计经验的直流电机、无刷直流电机及交流伺服电机为例,来解说电机的特征。虽然如此,电机驱动所需要的必要知识和诀窍,是任何一种驱动方式都共通的。

### 3.1 电机驱动的基础知识

在第 2 章中,关于电机的物理量(转矩、转数等)的使用方法和单位,并没有做出严格的定义,只是当做一般性的易于理解的表达方式来使用和说明的。但是,电机及其驱动方式的选择,或是到了实际设计的时候,就有必要理解关于电机的物理量的使用方法和单位了。在本章的最初,先来介绍电机驱动的基础知识。

#### 3.1.1 力与转矩的关系

所谓转矩,是指旋转力,如图 3.1 所示的那样,从距离旋转轴半径  $r=1\text{m}$  的位置,产生的力  $F$ ,即被规定为转矩。在国际单位体系(SI)中,转矩的单位是  $\text{N} \cdot \text{m}$ 。在这里,N 为牛顿,是表示力的单位。 $1\text{N}$ ,就被定义为“使质量为  $1\text{kg}$  的物体以  $1\text{m/s}^2$  的加速度运动时所需要的力”。

因为地球上的重力加速度是  $9.8\text{m/s}^2$ ,所以加在质量为  $1\text{kg}$  的物体上的力

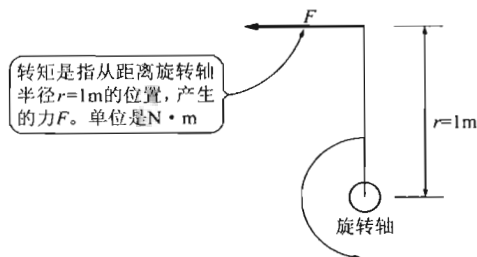


图 3.1 力与转矩的关系

就是 9.8N。此外,转矩和力的单位,除了国际单位体系之外,还有重力单位体系。地球表面质量为 1kg 的物体受到的重力就是 1 千克力(kgf)。这种场合下,转矩的单位就是  $\text{kgf} \cdot \text{m}$ 。

但是,从电机的数据表上来看,也会有转矩的单位是  $\text{kg} \cdot \text{m}$  和  $\text{kg} \cdot \text{cm}$  这样,把 f 去掉作为单位的情况。在这里,只是把 f 去掉了,还是作为重力单位体系来考虑的。

转矩的单位必须要注意的是,力的“N”和长度的“m”是相乘在一起的。比如说,在转矩为  $1\text{N} \cdot \text{m}$  的电机上,装置着一个半径是 10cm(0.1m)的齿轮,那么,这个齿轮所产生的力就是:

$$F = 1\text{N} \cdot \text{m} \div 0.1\text{m} = 10\text{N}$$

作为单位的 m,在计算过程中就被消去了。

### 3.1.2 转矩和功率的关系

正在旋转的电机的输出功率  $P_{\text{out}}$  (W) 为

$$P_{\text{out}} = T\omega$$

其中,  $T$  为转矩;  $\omega$  为角速度(rad/s)。

关于角速度  $\omega$ ,因为它也是在电气领域经常出现的单位,所以应该也是大家非常熟悉的一个单位。通常来说的话,只要记住  $\omega = 2\pi f$  ( $f$  为频率,单位为 Hz)就行了,而对于弧度的定义,则不需要怎么考虑。如图 3.2 所示的那样,弧度(radian),是把圆周上前进与半径  $r$  相等的长度,来定义为 1 弧度的。那么就有,圆的周长  $L = 2\pi r$ 。此外,频率  $f$  (Hz),指的是 1s 之内反复多少次,那么,角速度  $\omega$  (rad/s),就是 1s 之内前进了多少弧度。由此,就可以推导出,  $\omega = 2\pi f$  这个公式了。

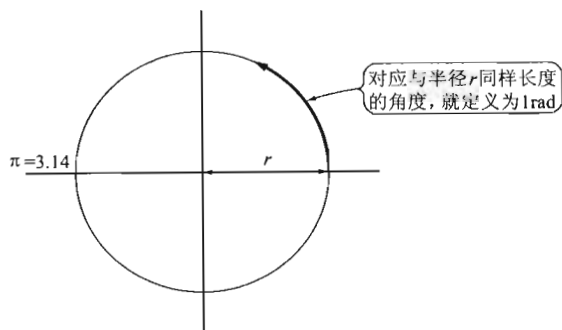


图 3.2 弧度的定义

顺便值得一提的是, 电机的输入功率  $P_{in} = \text{电压(V)} \times \text{电流(A)}$ , 因此, 电机的效率就是“输出功率/输入功率”了。

### 3.1.3 电机驱动的 4 象限运行过程

驱动电机的时候, 如图 3.3 所示的那样, 需要先来理解 4 象限运行过程。在设计或者选择电机、驱动程序的时候, 4 象限运行过程之中, 需要使之对应哪个运行过程, 或者正在对应哪个运行过程, 都是需要仔细考虑的问题。

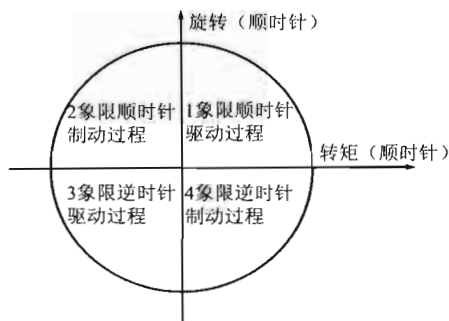


图 3.3 电机驱动的 4 象限运行过程

根据电机、驱动程序的驱动方法的不同, 也有完全无法对应的情况出现。这一点, 需要特别注意。

#### 1. 1 象限运行过程

转矩(顺时针)为正位置, 旋转(顺时针)也为正位置。因此, 1 象限运行过程就是顺时针方向驱动过程。

## 2. 2 象限运行过程

转矩(顺时针)为负位置,旋转(顺时针)为正位置。因此,2 象限运行过程就是顺时针方向旋转时的制动运行过程。旋转方向和转矩的产生方向,呈现互逆的关系。

## 3. 象限运行过程

转矩(顺时针)为负位置,旋转(顺时针)也为负位置。因此,3 象限运行过程就是逆时针方向驱动过程。

## 4. 4 象限运行过程

转矩(顺时针)为正位置,旋转(顺时针)为负位置。因此,4 象限运行过程就是逆时针方向旋转时的制动运行过程。旋转方向和转矩的产生方向,呈现互逆的关系。

### 3.1.4 直流电机的等价电路

电机有电阻  $R$ 、电感  $L$  及感应电动势等成分。实际上,这些成分是作为分散常数存在的。但是,分散常数的计算方程非常不好解,所以如图 3.4 那样,把它们作为集总常数,来表现直流电机的等价电路。

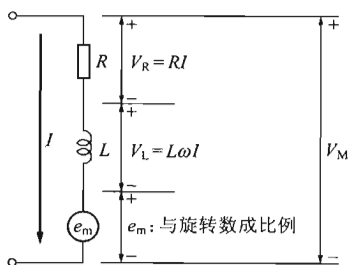


图 3.4 直流电机的等价电路

#### 1. 电阻 $R$

电阻  $R$  是指线圈的绕线电阻和电刷与换向器的接触电阻等。电阻  $R$  的电压  $V_R$  (V) 的推导式为

$$V_R = RI$$

#### 2. 电感 $L$

电感  $L$  是指线圈自身的电感。电感  $L$  的电压  $V_L$  (V) 的推导式为

$$V_L = L\omega I$$

其中,  $\omega$  是指电流  $I$  的角频率。直流电机在恒定旋转的时候,因为其电流是直流,所以  $V_L = 0$  V。但是,电感  $L$  是电机启动和制动时以及 PWM 驱动方式的重要组成部分。

### 3. 感应电动势 $e_m$

感应电动势  $e_m$  是指电机在旋转过程中所产生的电压。感应电动势  $e_m$  由于发电作用,会产生与电机的电流方向  $I$  反向的电流,并因此产生电压。因此,感应电动势也被称为反向电压。感应电动势常数用  $K_E$  来表示,感应电动势  $e_m$  (V)的推导式为

$$e_m = K_E \omega$$

其中,  $K_E$  为感应电动势常数 ( $V \cdot s/rad$ )。这时候的  $\omega$  (rad/s) 是电机旋转的角速度。

### 4. 电机电压 $V_M$

电阻  $R$  的电压  $V_R$  (V)、电感的电压  $V_L$  (V),以及感应电动势  $e_m$  (V)的合计电压,就是电机电压  $V_M$  (V)。

## 3.1.5 直流电机的转数可以通过电压控制

从现在开始的内容会变得稍稍有些困难。但是这对于理解电机的驱动原理是非常重要的内容。

直流电机的转数可以通过加在电机上的电压来控制。当然,电压升高的话,相应的电流也会变大,所以电机的转数也会上升……虽然一方面是这样子的,但对电机来说,还有别的原理在发挥作用。

试着考虑一下只有一个电机在旋转时候的情况。不过,这个电机是摩擦极小的电机,电机在恒定旋转的时候产生作用的力,在这个场合,就只有摩擦力了。不过,我们已经假定了这个电机是摩擦极小的电机,那么,维持旋转所需要的转矩就几乎可以忽略不计了。但并不是说,这个电机的转数就可以没有上限地上升。

请参考图 3.4 所示的直流电机的等价电路。电机的感应电动势  $e_m$  是与电机的转数成比例的电压。此外,感应电动势  $e_m$  是产生与电机电流  $I$  方向相反电流的电压。如果感应电动势  $e_m$  超过加在电机之上的电压的话,即便是低摩擦的电机,转数也是不可能维持的。这个场合,电机可以以感应电动势  $e_m$  与电机电压  $V_M$  几乎相等的转数来维持运转。通过以上内容,就能够很好理解利用电机电压  $V_M$  来控制电机转数这一原理了。

电机存在负荷和摩擦的时候,就需要与之相应的转矩。为了使电机产生转

矩,就需要电机电流,因此电阻电压  $V_R$ 、电感电压  $V_L$  及感应电动势  $e_m$  就会变低。结果就是,电机的旋转数比先前的例子低。

### 3.1.6 线性放大器驱动和 PWM 放大器驱动

电机的驱动方式,大致可以分为线性放大器驱动方式和 PWM 放大器驱动方式两种。下面就对这两种驱动方式的特征进行说明。

图 3.5 就是最简单的电机驱动电路的一个例子。图 3.5(a)是线性放大器驱动的例子,图 3.5(b)是 PWM 放大器驱动的例子。这两种方式在电路上的差别,主要在于电机是与并联二极管一起被放置在 PWM 放大器驱动电路中的。在这两种电路中,晶体管都是 N 沟道 MOSFET,而且这个 MOSFET 都是呈现理想的恒定电流特性的。

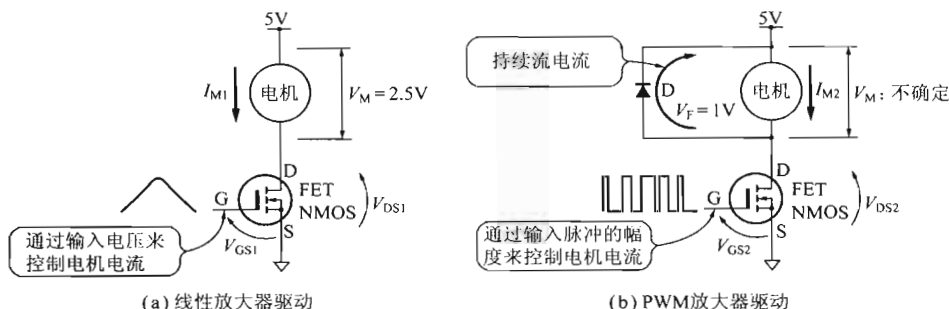


图 3.5 线性放大器驱动和 PWM 放大器驱动

图 3.6 所示是线性放大器驱动电路和 PWM 放大器驱动电路的时序图。其中,图 3.6(a)是线性放大器驱动电路的时序图,图 3.6(b)是 PWM 放大器驱动电路的时序图。我们假定电路的电源电压是 5V,在电机中流动的电流是 1A,电机的电阻是  $1\Omega$ ,电机的感应电动势是 1.5V, MOSFET 的线性运动时的 ON 电阻是  $0.1\Omega$ 。

#### 1. 栅极电压

线性放大器的栅极电压  $V_{GS1}$  是  $2.5V_{DC}$ ,这时, MOSFET 在饱和区域运转,电机中有 1A 的电流在流动。

PWM 放大器的栅极电压  $V_{GS2}$  就成了  $0\sim 5V$  的脉冲信号,这个信号的 ON/OFF 可以控制 MOSFET 的开关, MOSFET ON 表示在线性区域运转。PWM 放大器电路中的电机电流  $I_{M2}$  就是由于 MOSFET ON/OFF 而产生的电流,其

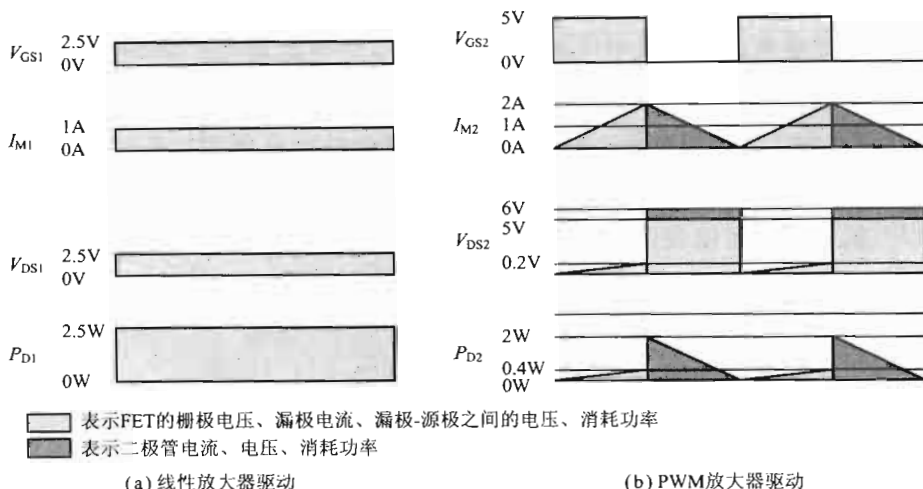


图 3.6 线性放大器驱动和 PWM 放大器驱动的时序图

平均电流是 1A。

因此,从电机的输出功率来看,线性放大器驱动电路和 PWM 放大器驱动电路是完全一样的。

## 2. 电机电流

线性放大器的电机电流是  $1A_{DC}$ 。

PWM 放大器的电机电流是通过 MOSFET 的 ON/OFF 而产生的脉冲,平均电流是 1A。PWM 驱动的电机电流会产生各种各样的复杂行为。虽然对于这一点,在文章中能不能准确地表达出来,但我感到有些不安,还是想要对其加以说明。

MOSFET 在 ON 的期间,电机中会有驱动电流流动。在电机上,我们使用线圈。因为线圈电流表现的是电压的积分特性,所以电机电流会随之逐渐上升。这个电流会从 5V 的电源开始,向接地(GND)的方向流动。这就是电机的驱动电流。

MOSFET 在 OFF 的时候,MOSFET 之中是没有电流流动的。但是电机的线圈还是在为了维持电流的流动而发挥着作用。这一点很重要,请一定要好好地理解。

图 3.7 表现了 MOSFET 从 ON 的状态变化为 OFF 的状态这一瞬间,电机电流随之发生改变的情况。其中,电阻是  $1\Omega$ ,这一瞬间,电机电流是 2A,所以,

$V_R = RI = 2V$ 。电机的感应电动势  $e_m = 1.5V$ 。截止到目前的内容,还是不难理解的。下面就要说说电机中电感的电压了。这才是问题点。MOSFET 从 ON 变化为 OFF 的瞬间,电机电流也随之从增加变化为减少。这样一来,对电感来说,就会产生反方向的电压。会发生怎样程度的电压变化呢?从结果上来说,变化的电压使二极管 ON,简单地说,就是因为二极管是处于反向连接状态的,所以二极管中电流就会像图中那样,变成从下往上流动的状态。二极管的正方向电压  $V_F = 1V$ ,所以电机电压  $V_M$  也与之相等。如果考虑电机电压的箭头符号的话,那么就是  $V_M = -1V$ 。由此,电感的电压  $V_L$  就是

$$V_L = V_M - V_R - e_m = -1V - 2V - 1.5V = -4.5V$$

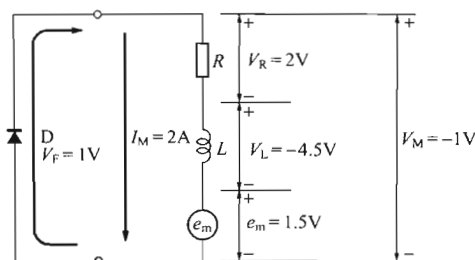


图 3.7 FET 从 ON 变化为 OFF 时的电机电流

MOSFET 在 OFF 状态下的电机电流也就成为通过二极管的回路电流了,这个电流也被称为持续电流。该电流的方向和驱动电流是一样的,所以这个电流也就成为了转矩。导致持续电流减少的原因,无非就是下面几点:因电阻而产生的发热损耗、因二极管而产生的发热损耗,以及电机旋转时所做的功。

### 3. 加在 MOSFET 上的电压

对于线性放大器电路来说,电机电流  $1A$ ,电阻  $1\Omega$ ,所以电阻电压  $V_R$  就是  $1V$ 。感应电动势  $e_m$  是  $1.5V$ ,所以 MOSFET 的漏极-漏极之间的电压  $V_{DS1}$  就是  $2.5V$ 。

对于 PWM 放大器电路来说,由于 MOSFET 的 ON/OFF 状态的改变,MOSFET 的电压状况也会随之改变。

MOSFET 在 ON 的时候,MOSFET 的 ON 电阻就是  $0.1\Omega$ ,所以电机电流就会从  $0A$  变化增加为  $2A$ ,于是加在 MOSFET 上面的电压  $V_{DS2}$  也就从  $0V$  变成了  $0.2V$ 。

MOSFET 在 OFF 的时候,还有一个很麻烦的事情,就是电机电流变成了





回路电流,电机电压  $V_M$  是  $-1V$ (请注意图 3.7 中的箭头符号),所以电机与 MOSFET 的接触点上,由于电源电压变大了  $1V$ ,使得加在 MOSFET 上的电压变成  $6V$ 。此外,在二极管上的电压与电机电压  $V_M$  一样,是  $-1V$ 。

对于这样的 PWM 驱动方式来说,由于加在 MOSFET 上的电压有时会超过电源电压,所以对于 MOSFET 的耐压是一个考验,需要引起足够的注意。此外,在这次的例子中,由于插入了二极管,持续电流变成了回路电流;如果没有二极管的话,电机电压  $V_M$  就会超过 FET 的源极-漏极耐压程度,从而引发击穿。因为也会有损坏 MOSFET 的情形发生,所以无论如何,请一定要在电路中加入续流二极管。

#### 4. MOSFET 的消耗功率

在线性放大器电路中,电机电流是  $1A$ ,加在 MOSFET 上的电压是  $2.5V$ ,所以 MOSFET 所消耗的功率为  $2.5W$ 。

PWM 放大器电路的情况呢? MOSFET 在 ON 的期间,电机电流从  $0A$  增加到了  $2A$ ,加在 MOSFET 上面的电压从  $0V$  增加到了  $0.2V$ ,所以 MOSFET 所消耗的功率也就随之从  $0W$  变成了  $0.4W$ 。在这一段时间中的平均功率消耗就是  $0.2W$ 。

MOSFET 在 OFF 的时候呢? 在 MOSFET 之中,并没有电流流动,所以 MOSFET 所消耗的功率也就是  $0W$ 。但是,注意,二极管是有功率损耗的。电机电流从  $2A$  减少到  $0A$ ,由于加在二极管上的电压是  $1V$ ,所以二极管所消耗的功率从  $2W$  减小到  $0W$ 。这一段时间内的平均消耗功率就成了  $1W$ 。

由此可知,MOSFET 在 ON 的时候以及在 OFF 的时候加在一起来看的话,PWM 放大器电路所消耗掉的整体功率就是  $(0.2W + 1W)/2 = 0.6W$ (除以 2 是因为 MOSFET 的 ON 的时间长度和 OFF 的时间长度是相等的)。

#### 5. 线性放大器驱动和 PWM 放大器驱动的特征

即使在电机的输出电压相同的情况下,PWM 放大器驱动方式的功率比线性放大器驱动方式的功率要低,这也就是说,驱动电路的效率要更好一些。

在这次的例子中,因为电源电压是  $5V$ ,所以还体现不出来特别大的差异性。如果电源电压变得非常巨大的话,那么自然而然地,从驱动电路这一方面来说,线性驱动方式和 PWM 驱动方式就会变得非常的不一样。因此,现在,除了一部分的特殊用途以外,其他情况下都是采用 PWM 驱动方式的。

## 3.2 电机的 PWM 驱动方法

这一部分,我们开始解说现在最主流的 PWM 驱动方法。图 3.8 表示的是基于 H 电桥电路的 PWM 放大器驱动方法。H 电桥电路,是在直流电机的驱动电路中最常被使用的一种电路,它是一种只通过单一电源就能够让直流电机正向旋转/反向旋转的电路。

请试着比较一下图 3.8(a)和图 3.8(b)的差别。看上去虽然感觉是完全相同的电路,但是差别之处在于高压侧的 MOSFET( $T_{R1}$ ,  $T_{R3}$ ),图 3.8(a)中使用的是 P 沟道 MOSFET,而在图 3.8(b)中则使用的是 N 沟道 MOSFET。虽然只有这么一点的差异,但却导致 MOSFET 的驱动方法变得完完全全的不一样了。而且对于电机的控制方法也就随之变得完全不一样了。

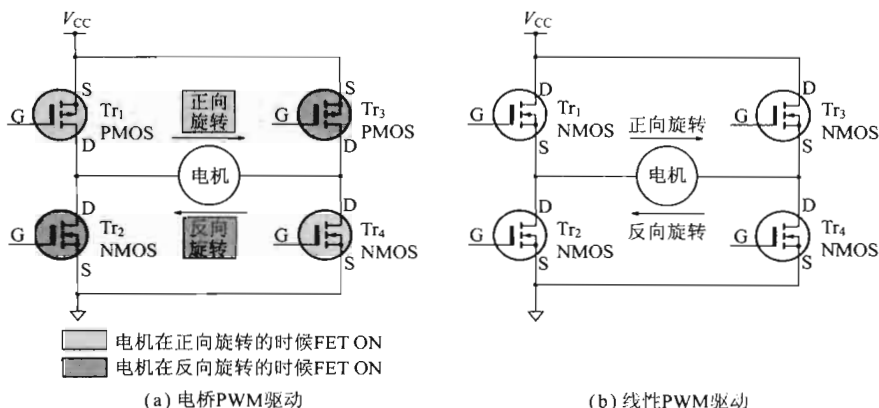


图 3.8 基于 H 电桥电路的 PWM 放大器驱动方法

从现在开始我们转到解说的部分上来。为了说明时候的方便,我们按照驱动方法的不同分别为图 3.8(a)和图 3.8(b)取了不同的名字。

对于前者,我们称之为电桥 PWM 驱动,而对于后者,我们称之为线性 PWM 驱动。但是要注意,这种称呼方法并不是一般广为通行的称呼方法,而仅仅是笔者自己比较习惯的称呼方法而已。此外,关于 PWM 驱动方法,除了这两种之外,还有各种各样多种驱动方法的提案,而在本章中所介绍的,仅仅是笔者自己有过实际设计经验的驱动方法。

前面已经说明过的续流二极管,在这里将担当 MOSFET 的体二极管。

### 3.2.1 电桥电路 PWM 驱动的 MOSFET 栅极驱动器电路的例子

图 3.9 表示的是电桥电路 PWM 驱动的 MOSFET 栅极驱动器电路的例子。在这个电路的例子中,我们使用安森美半导体公司生产的 MC33033 来作为 MOSFET 驱动器 IC。这个 IC 是无刷直流电机控制器,在直流电机上也可以应用。下面开始解说电桥 PWM 驱动方法的特征。

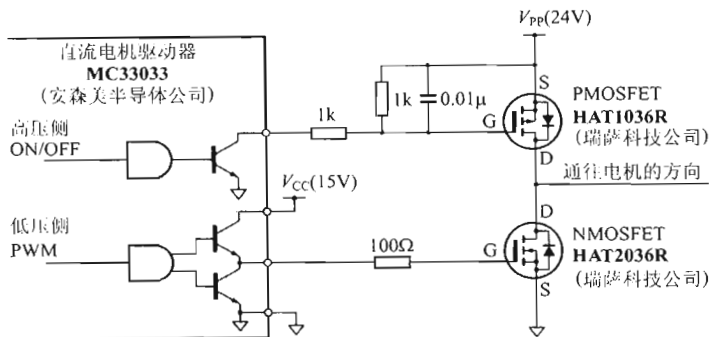


图 3.9 电桥电路 PWM 驱动的 MOSFET 栅极驱动器电路的例子

1. 如果在高压侧使用了 P 沟道 MOSFET 的话就不再需要使用浮动电源了

因为在高压侧使用了 P 沟道 MOSFET, MOSFET 的电源就是  $V_{PP}(+24V)$  了, 所以为栅极驱动器而准备的浮动电源就不需要了。

不再需要浮动电源, 意味着不再需要关于栅极驱动器的信号的电平转换, 所以 IC 的一体化也就变得更加容易了。作为直流电机驱动器和无刷直流电机控制器, 这种能够通过单一芯片来驱动的 IC, 现在各家公司都在销售。MC33033 也是这种 IC。

2. PWM 驱动只是在低压侧 MOSFET 进行的

PWM 驱动只是在低压侧 MOSFET 进行的。高压侧的 MOSFET, 电机在正向旋转的时候被设置为 ON, 电机在反向旋转的时候被设置为 OFF。

3. 短路电流必须引起重视

低压侧 MOSFET 在被 PWM 进行驱动的时候, 在高压侧的 MOSFET 必须设置为 OFF 状态。但是, 当低压侧的 MOSFET 从 OFF 状态切换为 ON 的

状态的时候,高压侧的 MOSFET 的漏极电压会从高电压(接近 24V 的电压)转变为 0V,再加上 MOSFET 在漏极-栅极之间也有电容的缘故,栅极电压像是被漏极拖动那样发生下降,于是就会在一瞬间产生出短路电流。在图中,作为对策,我们在高压侧的 MOSFET 的栅极上连接上了电容。

### 3.2.2 电桥 PWM 驱动方法的特征

现在,我们来说明一下基于电桥 PWM 驱动方法来驱动直流电机时候的一些特征。在下面的说明中,我们举了一些相对来说比较容易理解的例子。但是对于无刷直流电机来说,这些特征其实是完全一致的。

#### 1. 正向旋转/反向旋转的可能性

在图 3.10(a)中,我们来说明基于电桥 PWM 驱动方法的正向旋转和反向旋转的运行情况。

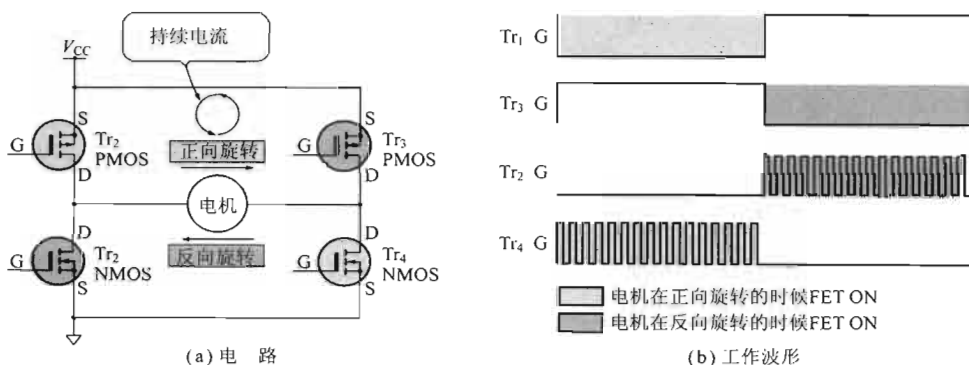


图 3.10 电桥 PWM 驱动的运行过程

在正向旋转的运转状况下,  $Tr_1$  是 ON 的状态,  $Tr_4$  被 PWM 所驱动。这个时候,  $Tr_2$  和  $Tr_3$  都是 OFF 的状态。在反向旋转的运转状况下,  $Tr_3$  是 ON 的状态,  $Tr_2$  被 PWM 所驱动。这个时候,  $Tr_1$  和  $Tr_4$  都是 OFF 的状态。像这样,通过切换 H 电桥电路的 MOSFET 的状态,就可以控制直流电机的正向旋转和反向旋转。

#### 2. 持续电流是高压侧的局部电流

持续电流正如图 3.10(a)所示的那样,是高压侧的局部电流。  $Tr_3$  被设置为 OFF 的状态, MOSFET 的体二极管就变成了续流二极管。

### 3. 电机运转的 4 象限运转过程中的第 2 象限运转过程和第 4 象限运转过程不能被对应

电桥 PWM 驱动方法决定了电机电流的流动方向。如果按照顺时针方向做正向旋转的话,由于只能产生出顺时针方向的转矩,所以无法对应上第 2 象限的运转过程。同样的,第 4 象限的运转过程也是无法被对应上的。第 2 象限的运转过程和第 4 象限的运转过程都是制动的过程,旋转方向和转矩呈相反的关系。

对于驱动电流和持续电流,从电机来看方向是相同的。即便是 PWM 的占空比是零的情况下,在电机中也不会产生出制动电流。PWM 的占空比如果是零的话,由于驱动电流会消失不见,所以电机就会因摩擦力的存在而使得转数下降。但是这种状态并不能看成电机正在被制动着。电机因为制动电流的流动,必须把 MOSFET 的开关从正向旋转变成反向旋转。这个时候,有可能会发生显著损害电机制动性能的情况。

图 3.11 就清楚地表明了这样一种情况。电机正按照顺时针(正向旋转)方向高速旋转。如果从这个状态开始,让电机停止的话,首先要使 PWM 的占空比缓缓下降,电机的驱动电流逐渐变少,转数下降。但因为这并不是制动电流的流动,所以不能如预期一样使转数下降。

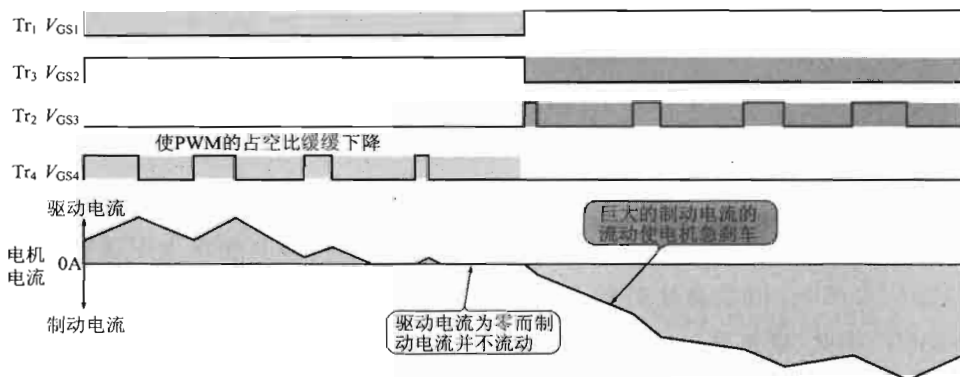


图 3.11 电桥 PWM 驱动的制动过程

然后把 MOSFET 的运转状态从顺时针(正向旋转)切换为逆时针(反向旋转)。因为电机正在高速旋转,所以会产生出很大的感应电动势,因此,现在用一个二极管使电机短路,巨大的制动电流作为高压侧的回路电流而流动起来。

在电桥 PWM 驱动方法中,由于高压侧的 MOSFET 不能被 PWM 所制动,所以这个制动电流无法被制动,而成为很大的电流。这样一来,电机就需要挂急刹车,从而给电机的制动性能造成很大的伤害。

#### 4. 并不是面向位置制动的驱动方法

电桥 PWM 驱动方法并不是面向位置制动的驱动方法。位置制动所要求的性能,并不只是位置的精确定位,电机的加速性能和减速性能也是非常重要的因素,必须被纳入考虑范围内。

由于没有办法很好地应对制动动作,所以电桥 PWM 驱动方法在减速性能上存在着问题。

因此,电桥 PWM 驱动方法主要用于追求一定转数(速度)性能,而不怎么过问减速性能的伺服电机和主轴电机。

为了不产生误解,这里再追加说明几句。不面向位置控制,并不是说完全无法进行位置控制。此外,即使对于制动动作来说,完全可以再另外设计一个制动动作专用的电路,来很好地实现制动。也有利用电阻和电容来实现再生制动的方法。

### 3.2.3 线性 PWM 驱动的 MOSFET 栅极驱动器电路的例子

图 3.12 表示的是一个线性 PWM 驱动的 MOSFET 栅极驱动器电路的例子。在这个电路中,使用国际整流器公司(IR)的 IR2106S 来作为 MOSFET 的驱动器 IC。此外,在这个电路中,虽然 N 沟道 MOSFET 作为电机的驱动晶体管来使用,但其实也存在着作为 NPN 晶体管和 IGBT 来使用的情况。

#### 1. 必须要有浮动电源

由于在高压侧的 MOSFET 上使用的是 N 沟道,所以其栅极电压就必须超过  $V_{PP}(+24V)$ 。但是高压侧的 MOSFET 的电源电压从 0V 开始变化为  $V_{PP}(+24V)$ ,因此,满足栅极驱动器的电源,就必须得是浮动电源了。

在交流伺服电机的驱动电路中,浮动电源必须得是 3 电路的。

#### 2. MOSFET 的栅极驱动器 IR2106S 的特征

这个 IC 的特征是,由于所消耗的电流非常小,因而并不需要使用特别的浮动电源,通过由二极管和电容组成的自举电源来运行,能够大幅度地削减其他周边关联的配件。此外,因为在接口上下了特别的工夫,所以对于 MCU 的

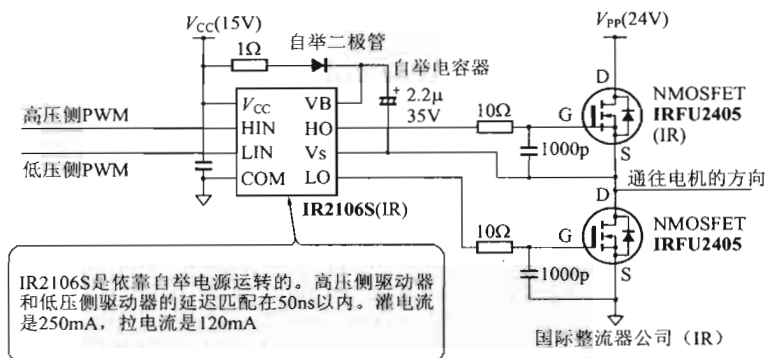


图 3.12 线性 PWM 驱动的 MOSFET 栅极驱动器电路的例子

3.3V 系列电源和自举电源来说,可以不使用光电耦合器而直接用接口进行连接。这一点,也是使周边关联配件大幅度减少的重要因素。

### 3. 自举电源

图 3.13(a)所显示的就是自举电源的充电路径。低压侧的 MOSFET 处在 ON 的状态的时候,通过二极管,可以给自举电容器进行充电。二极管电压每下降 1V 的时候,电容器的充电电压大约是 14V。

图 3.13(b)所显示的就是自举电源的放电路径。高压侧的 MOSFET 处在

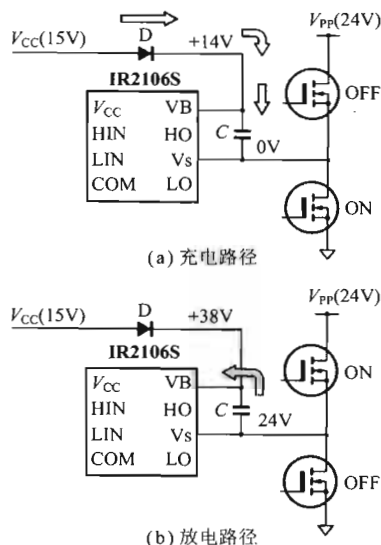


图 3.13 自举电源的充放电路径

#### 4. 自举电源的设计方法

$$C \geq \frac{2(2Q_g + I_{qbs}t_{ON} + Q_{ls} + I_{cbs}t_{ON})}{V_{CC} - V_F - V_{ls} - V_{min}}$$

$V_{CC}$ : 电源电压 = 15V;

 $I_{\text{qbs}}: C \text{ 的泄漏电流} = 30 \text{ nA};$ 

$Q_g$ : FET 的栅极电荷 = 70nC;

$V_{ls}$ : 低压侧 MOSFET 的 ON 电压 = 2.0V;

$V_F$ : 二极管的 ON 电压;

$$I_{chs}: \text{二次侧高压侧电流消耗} = 60 \mu\text{A}(\text{IR2106}) + 1\text{mA}(\text{IR2175});$$

$I_{ls}$ : 电平转换的充电电荷=5nC;

$V_{\min}$ : 高压侧电源的最小电压=10.8V。

用这些参数来计算,得出  $C=2.0\mu\text{F}$ 。因此,我们选择  $C=2.2\mu\text{F}/35\text{V}$ 。

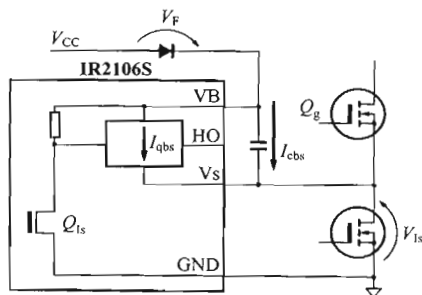


图 3.14 为算出自举电源的电容容量所需要的参数





### 5. 高压侧 MOSFET 和低压侧 MOSFET 共同的 PWM 驱动

在线性 PWM 驱动中,高压侧的 MOSFET 和低压侧的 MOSFET 双方,都是被 PWM 来控制的,是与电桥 PWM 驱动的根本性差异。这一点,对于电机的控制来说是有优势的。

图 3.12 中的高压侧的 MOSFET 和低压侧的 MOSFET 如果同时都是 ON 的状态的话,从  $V_{pp}(+24V)$  有短路电流经过 GND 从而导致 MOSFET 破损。所以,在那样设定 MOSFET 的 ON 和 OFF 的状态的栅极信号中,使用的是设定为同时不处于高的状态的时间(这一事件被称为停滞时间)的 PWM 信号。此外,线性 PWM 驱动中,我们希望高压侧的 MOSFET 和低压侧的 MOSFET 能具有开关特性,所以我们一般使用同样的 MOSFET。

### 3.2.4 线性 PWM 驱动方法的特征

下面我们来说明基于线性 PWM 驱动方法,来驱动直流电机时的特征。在下面的说明中,为了方便理解,我们用了直流电机来进行举例。但实际上,这些特征对于无刷直流电机和交流电机来说,也是完全一样的。

#### 1. 正向旋转和反向旋转的可能性

在图 3.15(a)中,我们来说明基于线性 PWM 驱动方法的直流电机的正向旋转和反向旋转运行状况。

在图 3.15(b)中,把这一运行状况表示为时间图。A 点的电压占空比是 60%,B 点的电压占空比就是 40%了。只有在 A 点的电压和 B 点的电压不同的时候,才会在电机中有电流流动。此外,如果 A 点的电压占空比和 B 点的电压占空比都是 50%的话,A 点和 B 点的电压就完全相同了,也就不会有驱动电流在流动了。反过来说,A 点的电压占空比和 B 点的电压占空比如果颠倒过来的话,电机的驱动电流的方向也就会颠倒过来了。这一点是不难理解的。

也就是说,如果我们能够控制 A 点侧的 PWM 占空比和 B 点侧的 PWM 占空比的话,那么我们就能够控制电机电流的大小和方向,从而使电机的正向旋转和逆向旋转变得可以实现。

#### 2. 持续电流在高压侧和低压侧成为局部电流

在线性 PWM 驱动方法中,由于高压侧的 MOSFET( $Tr_1, Tr_3$ )和低压侧

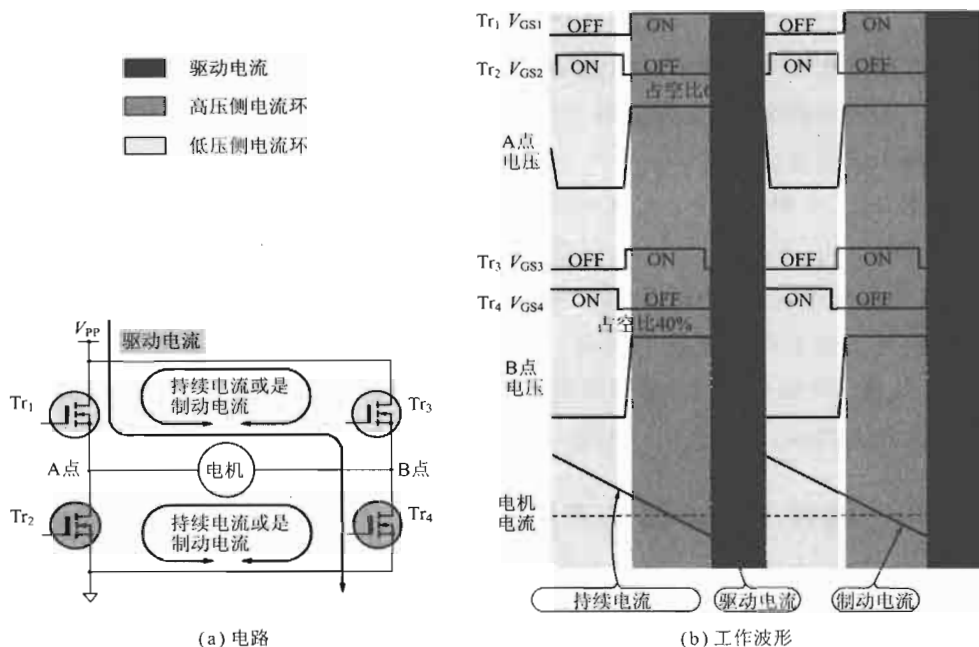


图 3.15 线性 PWM 驱动的运行过程

的 MOSFET ( $Tr_2, Tr_4$ ) 被设置为相互交错的 ON 的状态,也就形成了局部的电流回路。因此,电机电流的路径是不会断绝的(除了停止时间以外)。这就是说,PWM 驱动虽然一般被认为在 PWM 被设置成 OFF 的时候会使得电流的路径断绝,但是在线性 PWM 的情形下,由于经常在高压侧或者低压侧有封闭的回路,所以能够形成持续电流和制动电流两种电流同时按两个方向流动的状态。

这一点是与电桥 PWM 驱动方法的根本性差异之处。因此,这一点会对电机的控制性造成极大的影响。

### 3. 能够对应电机运转过程中的全部 4 个象限的运转过程

在线性 PWM 驱动方法中,并不存在像电桥 PWM 驱动方法那样的把 H 电桥的 MOSFET 作为“开关”来使用的概念。由于 PWM 的设定和感应电动势的关系,电机驱动电流和制动电流都可以被任意地控制。这样一来,就能够对应电机运转过程中的全部 4 个象限的运转过程。

在图 3.16 中,我们来说明基于线性 PWM 驱动的制动过程。电机正在做着高速的旋转。我们使 A 点电压的 PWM 的占空比缓缓下降,那么就会使 B

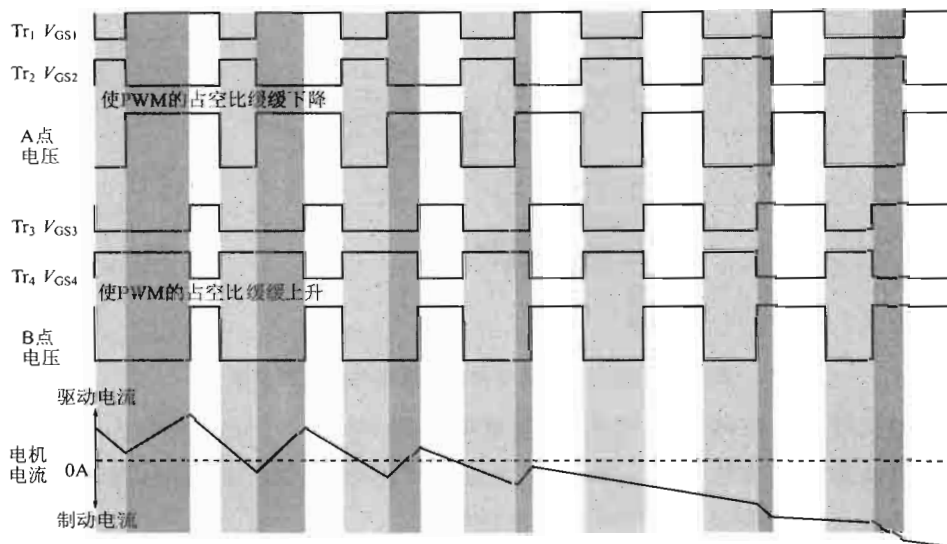


图 3.16 线性 PWM 驱动的制动过程

点电压的 PWM 的占空比缓缓上升,从而导致电机的驱动电流逐渐变少,而相反的制动电流逐渐增加。如果制动电流相比驱动电流更多的话,电机就被挂了刹车。

在线性 PWM 驱动方法中,由于我们可以像上述那样调整驱动电流和制动电流的比例,所以理所当然的可以实现 1 象限的运转过程和 3 象限的运转过程。对于旋转方向和转矩的产生方向相反的 2 象限的运转过程和 4 象限的运转过程,也是可以对应出来的。

#### 4. 能够适用于全部的控制方式,包括位置控制、速度控制及转矩控制

在电机的控制方式中,有位置控制、速度控制及转矩控制。线性 PWM 驱动方法对这些所有的控制方法都适用。

### 3.3 基于 PWM 驱动的电机电流的检测方法

下面我们来介绍基于 PWM 驱动的电机电流的主要的检测方法。此外,因为在本节中会有各种各样的检测方法,所以为了让实测波形更容易理解,我们以基于三相正弦波驱动的例子来加以说明。

在图 3.17 中,我们展现了主要的电机电流检测方法。在图中,我们把直接

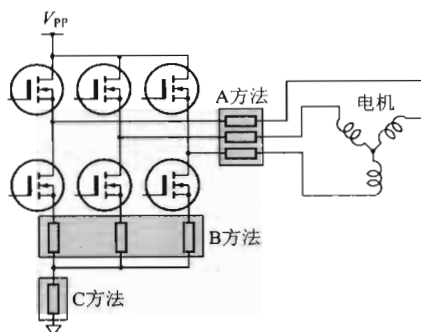


图 3.17 基于 PWM 驱动的电机电流的检测位置

测量电机线的方法称为 A 方法，把测量电压侧 MOSFET 的电源和 GND 的电流的方法称为 B 方法。进一步的，把测量电机驱动放大器的电源电流的方法称为 C 方法。下面来叙述各个测量方法的特征。

### 3.3.1 直接测量电机线的方法(A 方法)

#### 1. 能够测量出最准确的电流

因为是直接测量电机电流，所以能够测量出最准确的电流。但由于 PWM 驱动而产生的电流是脉冲电流，所以在经过平均化处理后，会得到图 3.18 的“A 方法”所示的实测波形。

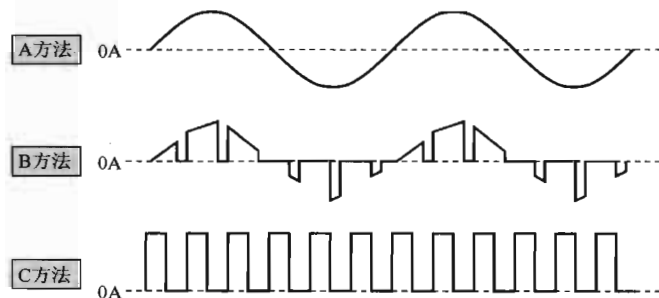


图 3.18 电机电流的主要检测方法的实测电流波形

#### 2. 测量过程很困难

由于 PWM 驱动而产生的电机电流会一直在高电压和低电压之间反复切换，所以通过电机线来测量电机电流是一件非常困难的事情。



### 3.3.2 测量电压侧 MOSFET 的电源和 GND 的电流的方法 (B 方法)

#### 1. 需要对测量的电流做加工

在这个点上的测量,由于无法测量高压侧的局部电流,所以得到的是图 3.18 的“B 方法”那样的实测波形。在求导电机电流的时候,需要对高压侧的局部电流进行预测。

#### 2. 易于测量

由于测定位置没有电压的变化,所以易于测量。

### 3.3.3 测量电源电流的方法(C 方法)

#### 1. 求导电机电流非常困难

在这个位置进行测量,得到的是三相电流的绝对值的总和,而且由于无法测量局部电流,就形成了图 3.18 的“C 方法”那样的实测电流。由于求导电机电流非常困难,因而采用检测电机放大器的过载电流的方法。

#### 2. 易于测量

由于测定位置没有电压的变化,所以易于测量。此外,由于测量位置只有一处,所以测量电路用很小规模就可以实现。

### 3.3.4 测量电机电流的电路实例(A 方法)

在过去的时候,想要检测出 PWM 控制的电机的电流是一件非常困难的事情。在以前的方法中,必须要有使用了高昂的大型电流传感器(霍尔元件)和绝缘放大器(线性光耦合器)之类的附属电路。

国际整流器公司(IR)的 IR2175S 是在这个用途方面非常便利的 IC。图 3.19 所表示的就是测量电机电流的电路实例。从图中可以清楚地看明白,只用了这一个 IC 及电阻和电容器各一个,就能够检测出电机的电流了。这个 IC 的输出是 130kHz 的 PWM 信号。由于输出的是数字信号,可以通过微型计算机之类的定时器端子来直接读取。但是,IR2175S 的电流检测 PWM 频率是 130kHz,是相当高性能的型号。微型计算机的时钟频率如果过低(需要有 32MHz 以上)的话,检测电流的分辨率就会过低,需要设计 PLD 之类的定时器电路。

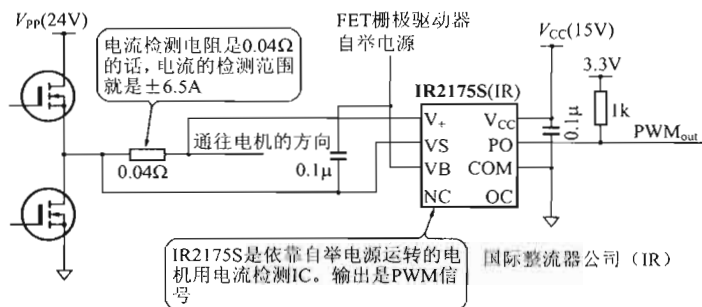


图 3.19 电机电流的检测电路的例子

# 第 4 章

## 电机控制的基础

到目前为止说明了电机的结构和驱动方法,本章将对电机的控制方法进行说明。笔者想在本章对伺服控制器设计的必要基础知识进行具体解说。

### 4.1 电机控制的基础知识

说到电机控制,那么就多多少少要理解控制理论。所谓的控制理论,给人的感觉好像是难以理解。实际上,要完全精通掌握现代控制理论,也确实需要花费很大的工夫。本章将对比较容易让人理解的经典控制理论,以及以经典控制理论为基础的数字信号处理方法进行解说。但是,即便说只是经典控制理论和数字信号处理方法,也是需要理解两三种专业书的。因为本章目的不是对那些进行说明,所以本章首先介绍笔者认为对电机控制很重要的基础知识。

#### 4.1.1 拉普拉斯变换是什么?

在读控制理论的专业书的时候,我想大家一开始就会对拉普拉斯变换产生疑问。拉普拉斯变换是把时间领域函数转变为频率领域函数。拉普拉斯变换一般在专业书上都有说明,本章就不对其进行详细说明了。如图 4.1 所示,下面对线圈  $L$  的电压和电流的求解公式进行说明。

求解线圈电压  $V_L$  的时候,如果把复数当做  $j$ ,角频率当做  $\omega$  的话,则

$$V_L = j\omega L I_L$$

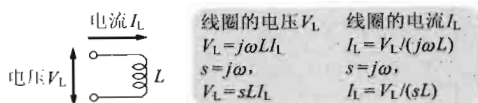


图 4.1 基于拉普拉斯变换,线圈电压及电流的求解

如果把上式及  $s = j\omega$  进行变换的话,则

$$V_L = s L I_L$$

其中,  $s$  为拉普拉斯变换因子。对拉普拉斯变换的详细解说就请读者阅读其他专业书籍,在本章中只是因为  $j\omega$  这个式子比较难以阅读,所以将其与  $s$  等换了而已。

同理,线圈的电流  $I_L$  的求解公式为

$$I_L = \frac{V_L}{sL}$$

怎么样,是不是感觉公式看起来好看了些呢?

### 4.1.2 传递函数是什么?

在控制理论的专业书籍中,读者大概也经常能看到传递函数出现吧。所谓的传递函数,是指输出特性和输入特性的关系。在使用拉普拉斯变换因子  $s$  的连续系统中,指的就是频率领域特性。线圈的电压公式为

$$V_L = s L I_L$$

如果用传递函数的方式表达的话,就变为了

$$\frac{V_L}{I_L} = sL$$

只是单纯地把电流  $I_L$  移到左边了而已,并不是很难的概念。

### 4.1.3 方框图是什么?

所谓方框图,就如字面意思那样,把传递函数用方框和线来表示。线圈的电压当做输出,电流当做输入,如果把传递函数用方框图来表示的话,就如图 4.2 所示。像这样,虽然表达方式不一样,但是传递函数和方框图其实是同一个东西。

方框图有很多种变换方法,在图 4.2 的例子中,只是把输入和输出置换了而已。方框中的式子也就在分子和分母中切换。在这里省略对其他变换方法



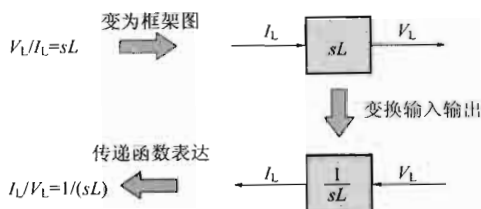


图 4.2 方框图是什么

的说明,其实和上述的变换方法没什么区别,只是如何让表达更简单易懂而已。

到目前为止已经说明过很多次了,线圈电流表示了线圈电压的积分特性,所以

$$\frac{I_L}{V_L} = \frac{1}{sL}$$

$1/s$  表示积分。

相反,线圈电压表示了线圈电流的微分特性,即

$$\frac{V_L}{I_L} = sL$$

$s$  表示微分。

#### 4.1.4 电机的传递函数

把转矩当做输入,旋转角度当做输出,试着基于方框图求解一般电机的传递函数。请参照图 4.3 电机的方框图。作用在电机上的转矩包括惯性转矩、摩擦转矩、轴的扭矩。

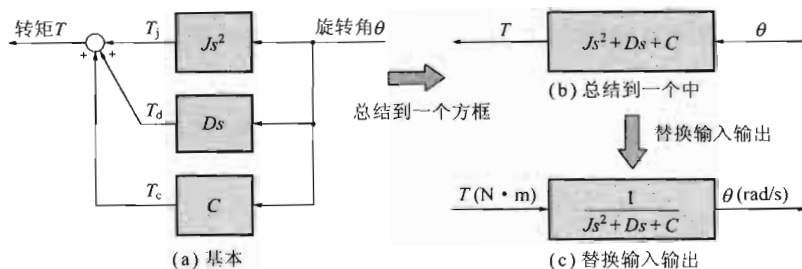


图 4.3 电机的方框图

##### 1. 基于转动惯量的转矩 $T_j$

转动惯量是指,在直线运动系中,以和物体质量相当的力进行旋转运动的

惯性量。转动惯量  $J$  的单位是  $\text{N} \cdot \text{m}^2$ 。直线运动系中,力  $F$ 、质量  $M$  及加速度  $A$  的关系式是

$$F=MA$$

将其放到旋转运动系统中的话,转动惯量  $T_j$ ,转动惯量  $J$ ,角加速度  $\alpha$  的关系式为

$$T_j=J\alpha$$

角加速度  $\alpha$  是旋转角弧度的 2 次微分。如果进行拉普拉斯变换的话,关系式就变为

$$T_j=Js^2\theta$$

方框图如图 4.3(a)的上部所示。

## 2. 基于摩擦的转矩 $T_d$

摩擦转矩  $T_d$  和角速度  $\omega$  成正比,和摩擦系数  $D$  的关系式为

$$T_d=D\omega$$

角速度  $\omega$  是旋转角弧度的 1 次微分。如果进行拉普拉斯变换的话,关系式就变为

$$T_d=Ds\theta$$

方框图如图 4.3(a)的中部所示。

## 3. 轴的扭矩 $T_c$

所谓轴的扭矩,是指在直线运动系统中和弹簧相当的物理量。请大家回想下弹簧秤。如果在弹簧秤上挂上重物的话,弹簧将伸长和重物重量成比例的长度。因为重力作用于重物上,所以力和弹簧的伸长量成比例。

在旋转运动系统中,轴的扭矩也是一样的,给电机一个转矩的话,虽然很微量,但是电机的轴也被扭转了。轴的扭矩  $T_c$  与轴的扭转系数  $C$  及旋转角弧度的关系式如下

$$T_c=C\theta$$

因为成比例,所以即使进行拉普拉斯变换,公式也不会变化,方框图如图 4.3(a)的下部所示。

## 4. 电机的传递函数

方框图的并联可以等价变换为和的形式。把惯性转矩  $T_j$ 、摩擦转矩  $T_d$  和

轴的扭矩  $T$ 。整合到一个方框中的话,就如图 4.3(b)所示。另外,输入输出替换,把转矩  $T$  当输入,旋转角当输出的话,就变成图 4.3(c)的方框图了,也就是一般电机的传递函数。

### 4.1.5 转矩控制放大器的传递函数

下面求解输入转矩电流指令,输出电机的实际发生转矩的传递函数。

#### 1. 放大器增益 $K_a$

转矩电流指令,因为是单片机内部处理的,所以是整数  $I(\text{count})$ 。放大器增益的单位是  $\text{A/count}$ 。通常,放大器增益  $K_a$  是电流检测的分辨率。放大器增益不是时间函数,而是比例元素。

#### 2. 转矩常量 $K_t$

转矩常量本来是和电机相关的参数,但是因为把转矩当做了电机传递函数的输入,所以把转矩常量放到放大器的传递函数中了。

#### 3. 转矩控制放大器的时间常数 $T_a$

转矩控制放大器中,对于转矩电流指令,电机的实际发生转矩会有一些延时。该延时可近似表达为 1 次延时元素。转矩控制放大器的时间常数定为  $T_a$ 。

1 次延时元素是控制领域的术语。在电气领域中,叫做 1 次 LPF (Low Pass Filter)。时间常数  $T_a$  和 1 次 LPF 的截至频率  $f_c$  的关系式为

$$f_c = \frac{1}{2\pi T_a}$$

1 次延时元素的传递函数是  $1/(T_a s + 1)$ ,所以如果把放大器增益  $K_a$ 、转矩常数  $K_t$  考虑进来的话,就会变成图 4.4(a)的方框图。方框图的串联可以等价变换为积的形式,所以整合到一个方框图的话,就如图 4.4(b)所示。

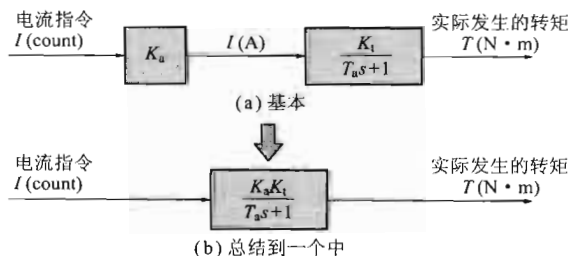


图 4.4 转矩控制放大器的方框图

如上述那样,如果把电机驱动电路当做转矩控制的话,驱动电路会吸收电机线圈的电阻成分、电感成分,以及感应电压,在控制上就可以不用考虑电阻、电感及感应电压了。

### 4.1.6 传感器的传递函数

试着考虑下含有编码器的传感器。电机的传递函数的输出是旋转角。另一方面,在编码器方式的位置检测器中,对脉冲进行计数。因此,传感器增益  $K_s$  的单位为 count/radian。因为旋转角和编码器计数都是位置信息,所以传感器增益  $K_s$  是比例元素。

### 4.1.7 电机驱动部的传递函数

图 4.5 中,以方框图的形式表示了电机驱动部全体的传递函数。把以下的参数代入这个传递函数中,再计算频率特性的话,就能得到图 4.6。这样,电机驱动部的基本特性就是 2 次积分的形式了。

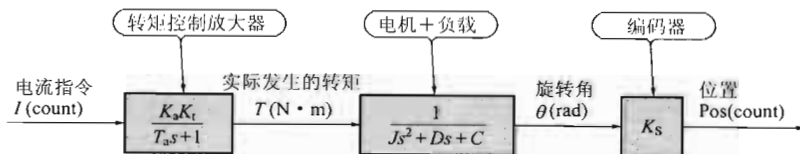


图 4.5 电机驱动部的方框图

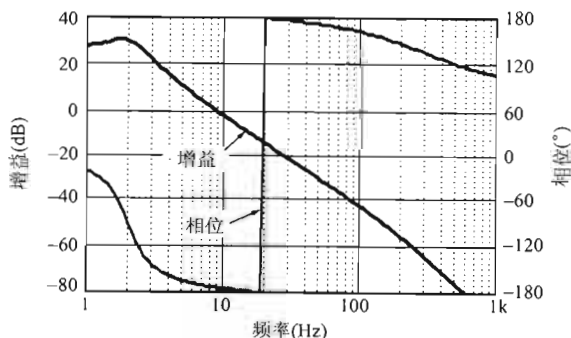


图 4.6 电机驱动部的频率特性

计算传递函数的频率特性的时候,把拉普拉斯变换因子  $s$  换回  $j\omega$  进行计算,这就是所谓的傅里叶变换。虽然拉普拉斯变换和傅里叶变换有  $s$  和  $j\omega$  的差异,但在本章中,基本上把两者当做相同的。本章的开头中就写到,拉普拉斯

变换是把时间领域函数变换为频率领域函数,意义就在于此。

$K_a=0.003802(\text{A/count})$	放大器增益
$K_t=20.5 \times 10^{-3}(\text{N} \cdot \text{m/A})$	转矩常量
$T_a=0.53 \times 10^{-3}(\text{s})$	转矩控制放大器时间常数
$J=82 \times 10^{-7}(\text{kg} \cdot \text{m}^2)$	转动惯量
$D=6.1827 \times 10^{-5}$	摩擦系数
$C=1.2949 \times 10^{-3}$	轴的扭转系数
$K_s=318.31(\text{count/rad})$	传感器增益

## 4.2 伺服控制器的作用

只根据电机的驱动电路,并不能决定直流电机和交流电机的位置。另外,虽然电机的机理上具备了速度控制性,但是因为负载和摩擦,旋转速度会随之变化。

伺服控制器的作用是,即便在负载变动和摩擦变动的系统中,也能使电机运转在目标位置或者目标旋转速度上。

图 4.7 是伺服控制器的全体构成图。到前面的章节为止,对图中右部的电机驱动部进行了相关说明。从本章开始将主要说明图中左部的伺服控制器。

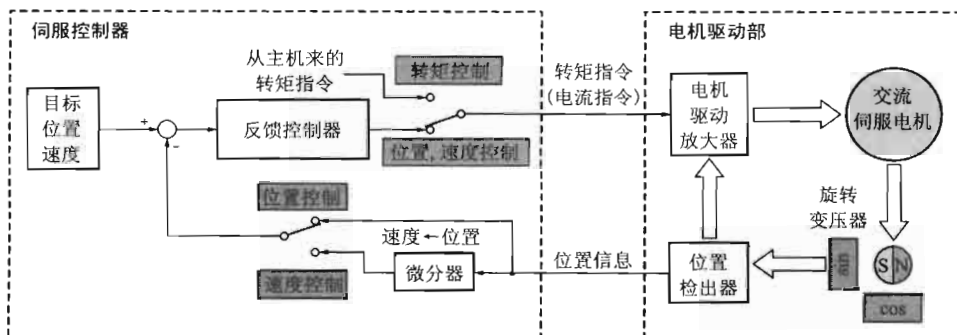


图 4.7 电机控制器的全体构成

### 4.2.1 位置控制的运作原理

#### 1. 电机的基本特性

在图 4.7 电机驱动部中,把转矩指令(电流指令)当做输入,位置信息当做

输出,计算频率特性的话,就能得到图 4.6 的增益特性和相位特性。这样,电机的基本特性就是 2 次积分的形式。因为 2 次积分的形式中,相位特性有  $180^\circ$  的滞后,所以在图 4.7 中,如果把比例放大器当做反馈控制器的话,就会发生振荡。也就是说,为了进行位置控制,增进相位的微分元素就变得很必要。

另外,虽然电机的特性是 2 次积分,但是也不是理想的特性。理想的积分特性的话,DC(频率为 0)的增益为无限大。但是实际的电机特性,因为摩擦的影响,增益特性在低频领域就饱和了。这样下去的话就会出现稳态偏差,这就是位置偏差的原因。因此,反馈控制器中,积分元素也很有必要。

## 2. PID 控制器

请参照图 4.8 的增益特性和相位特性。这些是在反馈控制器中使用了 PID 控制器的代表特性。在高频领域,基于微分,增进了相位。而在低频领域,基于积分,提高了增益。

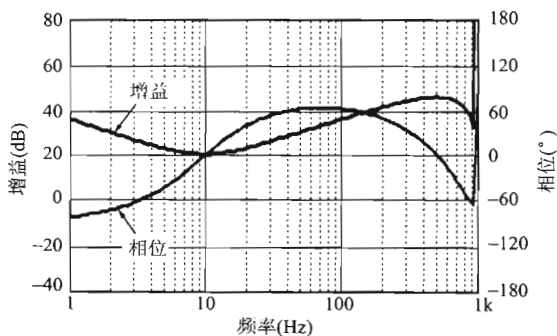


图 4.8 PID 控制器的频率特性

## 3. 开环特性

请参照图 4.9 的增益特性和相位特性。

把电机特性和 PID 控制器特性合在一起,就是我们通常说的开环特性。控制领域大概 50 Hz,相位裕度约  $55^\circ$ 。

### 4.2.2 速度控制的运作原理

速度控制如图 4.7 所示,对位置信息进行微分就是速度信息了。前面说过位置控制是 2 次积分,那么速度控制就是 1 次积分。

在 1 次积分中,相位滞后是  $90^\circ$ ,所以即使把比例放大器作为图 4.7 中的反

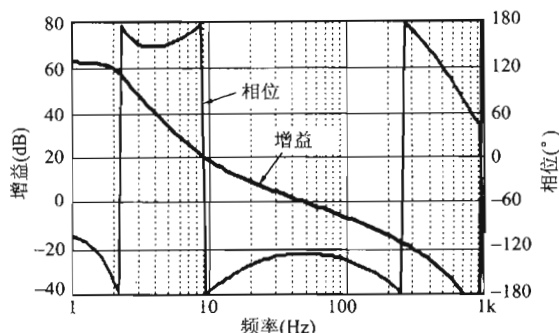


图 4.9 开环频率特性

馈控制器,也能够进行稳定控制。另外,和位置控制同理,速度控制也不是理想的积分特性,所以为了消除旋转中的速度偏差,在反馈控制器中加入积分元素。因此,反馈控制器中,PID 控制器的微分项  $D$  为 0,也就是 PI 控制。

### 4.3 数字信号处理的基础知识

本章讲述的伺服控制器是基于单片机软件的数字方式。在本章,将对笔者认为重要的数字信号处理的基础知识进行解说。

#### 4.3.1 延迟单元 $1/z$

打开数字信号处理的专业书籍的话,首先看到的就有  $1/z$ 。这就是延迟单元,表示 1 个样本的延迟。如图 4.10 所示,输入数据是  $X(n)$ 。 $n$  的意思是现在的数据。通过延迟单元  $1/z$  的话,就变为了  $X(n-1)$ 。 $n-1$  的意思是前一个样本的数据。

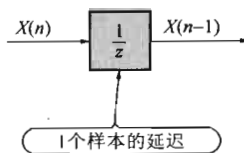


图 4.10 关于延迟单元

#### 4.3.2 连续系统传递函数和离散系统传递函数

下面解说下连续系统传递函数(模拟)和离散系统传递函数(数字)的异同。

在本章第 1 节中已经说明了,连续系统传递函数中使用了拉普拉斯变换因子  $s$ 。离散系统传递函数中使用的是延迟单元  $1/z$ 。请试着考虑下这两种函数有什么异同。

请回忆一下本章第 1 节的说明。拉普拉斯变换把时间领域函数变换到频

率领域函数。也就是说,连续系统传递函数表现的是频率领域,如果把  $s=j\omega$  置换的话,能够很容易求得系统的频率特性。

另一方面,延迟单元  $1/z$  表达的是时间,因此,离散系统传递函数表现的是时间领域,由传递函数能够很容易导出逐次计算式。

图 4.11 的方框图用传递函数表达的话,则

$$\frac{Y(n)}{X(n)} = 1 - \frac{1}{z}$$

这个式子转换为逐次计算式的话,则

$$Y(n) = X(n) - X(n) \cdot 1/z$$

$$Y(n) = X(n) - X(n-1)$$

因此,离散系统传递函数能够很容易地求得系统的阶跃响应等时间轴响应。

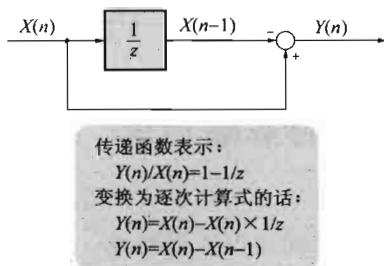


图 4.11 逐次计算式的变换方法

### 4.3.3 数字滤波器的设计方法

数字滤波器的设计方法有好几种,最常用的方法是,首先设计模拟滤波器,然后进行  $s-z$  变换,再求解数字滤波器的系数。下面以简单的例子来说明上述的设计方法(截止频率 100Hz 的 1 次 LPF)。

#### 1. 模拟滤波器的设计

设角频率  $\omega_n = 2\pi \times 100 = 628.3$ , 则传递函数为

$$\frac{Y}{X} = \frac{\omega_n}{s + \omega_n} = \frac{628.3}{s + 628.3}$$

#### 2. 预变换处理

从模拟滤波器变换到数字滤波器的话,会发生频率失真。预变换处理是指事先对失真的频率进行补偿。设补偿后的角频率为  $\omega_p$ , 数字取样周期  $T =$



0.001s, 则

$$\omega_p = \frac{2}{T} \tan \frac{\omega_n T}{2} = \frac{2}{0.001} \tan \frac{0.6283}{2} = 2000 \times 0.3249 = 649.8$$

### 3. $s$ - $z$ 变换

$s$ - $z$  变换的方法有好几种, 这次的设计例中使用双 1 次  $z$  变换。

$$s = \frac{2}{T} \cdot \frac{1-z^{-1}}{1+z^{-1}} = \frac{2(z-1)}{T(z+1)}$$

将  $s = \frac{2(z-1)}{T(z+1)}$  代入模拟滤波器的传递函数, 则

$$\frac{Y}{X} = \frac{649.8}{s+649.8} = \frac{649.8}{2(z-1)/T(z+1)+649.8}$$

乘以  $T(z+1)$ , 再将  $T=0.001$  代入, 则

$$\begin{aligned} \frac{Y}{X} &= \frac{649.8 \times T(z+1)}{2(z-1) + 649.8 \times T(z+1)} = \frac{0.6498 \times z + 0.6498}{2z - 2 + 0.6498 \times z + 0.6498} \\ &= \frac{0.6498 \times z + 0.6498}{2.6498 \times z - 1.3502} \end{aligned}$$

除以 2.6498 后得

$$\frac{Y}{X} = \frac{0.2452 \times z + 0.2452}{z - 0.5095}$$

设计好的模拟滤波器及数字滤波器的频率特性如图 4.12 所示。请注意数字滤波器中取样频率的 1/2 附近的频率和模拟滤波器的特性不同。

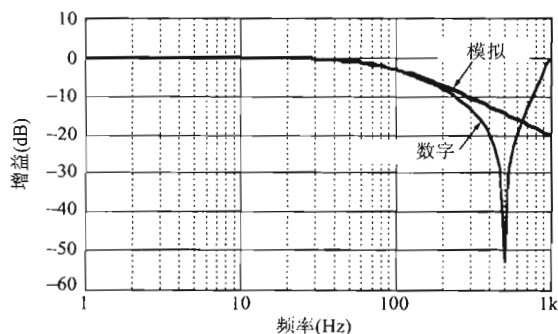


图 4.12 1 次 LPF 的频率特性

### 4. 变换为逐次加算式

$$\frac{Y(n)}{X(n)} = \frac{0.2452 \times z + 0.2452}{z - 0.5095}$$

除以  $z$  得

$$\frac{Y(n)}{X(n)} = \frac{0.2452 + 0.2452/z}{1 - 0.5095/z}$$

因此,

$$Y(n) \cdot (1 - 0.5095/z) = X(n) \cdot (0.2452 + 0.2452/z)$$

$$Y(n) - 0.5095 \cdot Y(n-1) = 0.2452 \cdot X(n) + 0.2452 \cdot X(n-1)$$

$$Y(n) = 0.2452 \cdot X(n) + 0.2452 \cdot X(n-1) + 0.5095 \cdot Y(n-1)$$

基于这个逐次计算式编写程序的话,就完成了数字滤波器。

## 4.4 数字滤波器设计帮助软件(DSP.EXE)

1次LPF的话,如前节介绍的那样,用手算也能够求得数字滤波器的常数。但是,如果是2次LPF的话,手算就比较困难了。

如果使用MATLAB等控制设计软件工具的话,能够比较容易地设计2次以上的数字滤波器。但是,如果想在自身的产品系统中,嵌入数字滤波器设计软件的话,因为MATLAB等软件工具的源代码和库等都没有公开,所以要自己事前制作。因此,这里介绍笔者制作的数字滤波器的设计帮助软件(DSP.EXE)。虽然没有市场上卖的软件那么有通用性,但是完全忠实前节介绍的设计顺序,很适用于2次滤波器。

DSP.EXE是用C语言描述的控制台应用程序。启动DSP.EXE的话,就能看到下述的菜单,执行输入的命令编号。因为没有图像表示的功能,所以如果想看频率特性的话,请把保存的文本文件复制到excel中,生成图像确认。

```
analog LPF design      : w1 digital PID design      : w7
analog NOTCH design    : w2 digital Multi FB control : w8
analog bode calculate  : c3
transfer to digital     : c4
digital bode calculate  : c5
save IIR parameter     : c6
Set : s Exit : e Menu : m
```

Select Command Number (ie. = c1) ;

### 4.4.1 模拟滤波器设计

设计方法和前节相同,首先,设计模拟滤波器。本软件有 LPF 和 NOTCH 的设计功能。

#### 1. 模拟 LPF 的设计(w1)

模拟 LPF 是固有频率为 100Hz、衰减系数(Dumping Factor)为 0.707 的 2 次 LPF(有下划线的部分,请从键盘输入)。

Select Command Number (ie. = c1) : w1

滤波器的次数(1 or 2) = 2

固有频率[hz] = 100

衰减系数 = 0.707

+ 0.000000e+000 \* s<sup>2</sup> + 0.000000e+000 \* s + 3.947841e+005

-----  
+ 1.000000e+000 \* s<sup>2</sup> + 8.884424e+000 \* s + 3.947841e+005

下面进行一些简单说明。

2 次 LPF 的传递函数一般是

$$\frac{Y}{X} = \frac{\omega_n^2}{s^2 + 2Z_n\omega_n s + \omega_n^2}$$

如果把固有角频率和衰减系数代入这个式中的话,就能得到刚才的结果。所谓衰减系数,指的是在电子工程中,表示共振锐度的  $Q$  值倒数的  $1/2$ ,即  $Q = 1/(2Z_n)$ 。

#### 2. 模拟陷波滤波器的设计(w2)

设模拟陷波滤波器的中心频率为 100Hz,衰减系数为 0.02,衰减量为 -20dB,则

Select Command Number (ie. = c1) : w2

中心频率[hz] = 100

衰减系数 = 0.02

振幅[dB] = -20

+ 1.000000e+000 \* s<sup>2</sup> + 2.513274e+001 \* s + 3.947841e+005

-----  
+ 1.000000e+000 \* s<sup>2</sup> + 2.513274e+002 \* s + 3.947841e+005

这里也进行一些简单说明。

2 次的陷波滤波器的传递函数一般是

$$\frac{Y}{X} = \frac{s^2 + 2Z_{n1}\omega_n s + \omega_n^2}{s^2 + 2Z_{n2}\omega_n s + \omega_n^2}$$

其中,  $\omega_n$  是中心角频率。分子中的衰减系数  $Z_{n1}$  只是把输入的值代入而已。那么问题就只剩下分母中的衰减系数  $Z_{n2}$  了。指定衰减量为  $-20\text{dB}$ 。在这个传递函数中, 衰减量由  $Z_{n1}$  与  $Z_{n2}$  的比来决定。关系式是

$$\text{衰减量 } Amp = 20\lg(Z_{n1}/Z_{n2})$$

为了求得  $Z_{n2}$ , 将上式换为指数函数, 则

$$Z_{n2} = Z_{n1} / 10^{(Amp/20)} = 0.002 / 10^{(-20/20)} = 0.02 / 10^{(-1)} = 0.2$$

那么分子中的衰减系数  $Z_{n1} = 0.02$  是什么意思呢? 这个值决定了陷波滤波器的锐度。值越小的话, 特性越锐利。

#### 4.4.2 解析模拟滤波器频率特性(c3)

解析前项中设计的模拟滤波器的频率特性。要解析的频率范围是由 DSP. EXE 启动时读入的文本文件 para. txt 决定的(S 命令, 再次读入)。频率解析的结果保存在文本文件 ccode. txt 中。

para. txt 的内容如下:

```
[param]
sample = 0.001;      //z 变换时的样本周期(s)
start = 1;           //解析开始频率(Hz)
stop = 1000;         //解析结束频率(Hz)
division = 200;      //频率分割数(最大 1024)
[end]
```

#### 4.4.3 s-z 变换(c4)

对 4.4.1 节中设计的模拟滤波器进行 s-z 变换, 求解数字模拟器的常数。s-z 变换时的数字取样周期由 para. txt 指定。下面的内容是前述陷波滤波器 s-z 变换后的结果。

Select Command Number (ie. = c1) : c4

```
+ 9.053280e-001 - 1.447831e+000 * Z^-1 + 8.842897e-001 * Z^-2
+ 1.000000e+000 - 1.447831e+000 * Z^-1 + 7.896178e-001 * Z^-2
```

#### 4.4.4 解析数字滤波器频率特性(c5)

解析前项设计的数字滤波器的频率特性。要解析的频率范围同模拟滤波器的时候一样,是由 DSP.EXE 启动时读入的文本文件 para.txt 决定的(S 命令,再次读入)。频率解析的结果保存在文本文件 ccode.txt 中。

#### 4.4.5 数字滤波器参数的保存(c6)

保存数字滤波器的参数到 iir.txt 中。iir.txt 的内容如下所示,对照的是下面的传递函数。

a2	b2	a2	b1	b0
- 7.896178e-001	8.842897e-001	1.447831e+000	- 1.447831e+000	9.053280e-001

图 4.13 的数字滤波器的方框图用传递函数表示的话,变为

$$\frac{Y(n)}{X(n)} = \frac{b_0 + b_1 \cdot z^{-1} + b_2 \cdot z^{-2}}{1 - a_1 \cdot z^{-1} - a_2 \cdot z^{-2}}$$

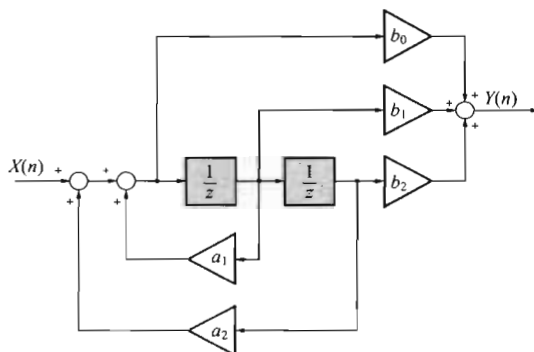


图 4.13 数字滤波器的方框图

4.4.3 节表示的传递函数将  $a_1$  及  $a_2$  符号反转了。如果变换为逐次计算式的话,理由就容易理解了。

$$Y(n) \cdot (1 - a_1 \cdot z^{-1} - a_2 \cdot z^{-2}) = X(n) \cdot (b_0 + b_1 \cdot z^{-1} + b_2 \cdot z^{-2})$$

$$Y(n) - a_1 \cdot Y(n-1) - a_2 \cdot Y(n-2) = b_0 \cdot X(n) + b_1 \cdot X(n-1) + b_2 \cdot X(n-2)$$

$$Y(n) = b_0 \cdot X(n) + b_1 \cdot X(n-1) + b_2 \cdot X(n-2) + a_1 \cdot Y(n-1) + a_2 \cdot Y(n-2)$$

$a_1$  和  $a_2$  变为正的符号。因为在软件中不需要进行符号处理,计算时速度

稍微变快了一些。

#### 4.4.6 数字 PID 控制器的设计(w7)

下面进行数字 PID 控制器的设计。设比例增益  $K_p = 1$ , 积分增益  $K_i = 0.1$ , 微分增益  $K_d = 10$ , 则

Select Command Number (ie. = c1) : w7

比例增益(Kp) = 1

积分增益(Ki) = 0.1

微分增益(Kd) = 10

+ 1.110000e+001 - 2.100000e+001 \* Z^-1 + 1.000000e-001 \* Z^-2

+ 1.000000e+000 - 1.000000e+000 \* Z^-1 + 0.000000e-000 \* Z^-2

图 4.14 是数字 PID 控制器的方框图。如果把这个方框图表达为传递函数的话,变为

$$\frac{Y(n)}{X(n)} = \frac{K_p + K_i + K_d - (K_p + 2.0f \cdot K_d) \cdot z^{-1} + K_d \cdot z^{-2}}{-1 \cdot z^{-1}}$$

因为数字 PID 控制器也是数字滤波器,所以用 c5 命令能够进行频率解析。

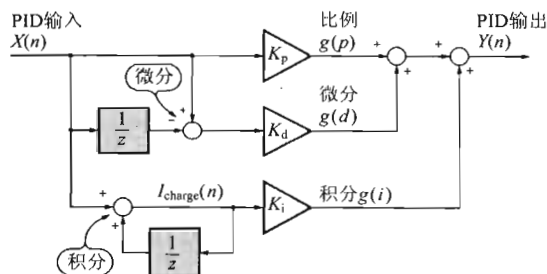


图 4.14 数字 PID 控制器的方框图

#### 4.4.7 数字位置/速度环控制器的设计(w8)

数字位置/速度环控制器是电机控制中常采用的控制器。作为位置控制环及其副回路,拥有速度控制环的反馈回路。即使接受的指令是阶跃状的位置指令,也不会有超调量。

设位置比例增益  $K_{pp} = 0.1$ , 速度比例增益  $K_{pv} = 10$ , 速度积分增益  $K_{iv} = 1$ , 则

Select Command Number (ie. = c1) : w8

位置比例增益(K<sub>pp</sub>) = 0.1

速度比例增益(K<sub>pv</sub>) = 10

速度积分增益(K<sub>iv</sub>) = 1

+ 1.210000e+001 - 2.200000e+001 \* Z^-1 + 1.000000e+001 \* Z^-2

+ 1.000000e+000 - 1.000000e+000 \* Z^-1 + 0.000000e+000 \* Z^-2

图 4.15 是数字位置/速度环控制器的方框图。对于这个方框图来说,位置指令 0 是输入  $X(n)$  的实际测量位置,把输出  $Y(n)$  作为电流指令时候的传递函数如下所示:

$$\frac{Y(n)}{X(n)} = \frac{(K_{pp} + 1) \cdot (K_{pv} + K_{iv}) - (2 \cdot K_{pv} + K_{iv} + K_{pp} \cdot K_{pv}) \cdot z^{-1} + K_{pv} \cdot z^{-2}}{-1 \cdot z^{-1}}$$

因为数字位置/速度环控制器也是数字滤波器,所以用 c5 命令对系统的频率进行分析。

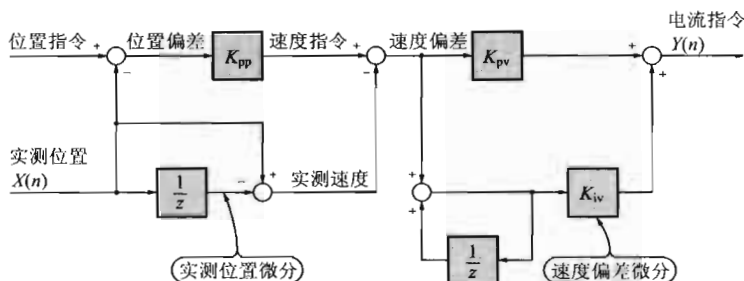


图 4.15 数字位置/速度环控制器的方框图

## 4.5 抓住滤波器设计的特征

设计数字滤波器及伺服调整时候最重要的事情是把握各自滤波器及伺服控制器的特征。改变各个参数的值的时候,如果能够把握特性是怎么变化的,那么设计工作就能很顺利地进行下去。下面通过使用数字滤波器帮助软件(DSP.exe),给大家解说前面一节介绍过的数字滤波器及伺服控制器的特征。

### 4.5.1 2次LPF的特征

2次LPF的特征是由固有角频率  $\omega_n$  及衰减系数  $Z_n$  决定的。在这里请大

家再看一次下面的传递函数：

$$\frac{Y}{X} = \frac{\omega_n^2}{s^2 + 2 \cdot Z_n \cdot \omega_n \cdot s + \omega_n^2}$$

对于 2 次数字 LPF, 固有角频率  $\omega_n = 2\pi \times 100 = 628.3$ , 衰减系数  $Z_n$  变化时, 可以从图 4.16 中清楚地看到频率特性的变化过程。由此可知, 衰减系数  $Z_n$  小的时候, 共振特性有增强的倾向。对于 LPF 而言, 一般使用滤波后的频率。滤波后的频率也就是振幅平坦时候衰减幅度小于  $-3\text{dB}$  的频率。同图中看到的一样, 同样固有角频率中衰减系数小的如果共振的话, 滤波掉的频率就会延伸。固有角频率同滤波掉的频率一致时候的衰减系数是  $0.7$ 。严格来说是  $1/\sqrt{2}$ , 即  $0.707106781$ 。

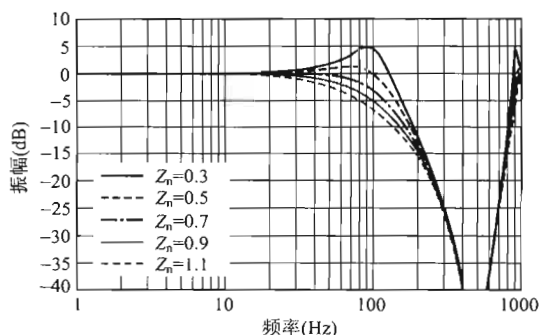


图 4.16 2 次数字 LPF 的频率特性

### 4.5.2 2 次陷波滤波器的特征

2 次陷波滤波器的特性是由固有角频率  $\omega_n$ 、衰减系数  $Z_{n1}$  及  $Z_{n2}$  决定的。我们再来看看以下传递函数：

$$\frac{Y}{X} = \frac{s^2 + 2 \cdot Z_{n1} \cdot \omega_n \cdot s + \omega_n^2}{s^2 + 2 \cdot Z_{n2} \cdot \omega_n \cdot s + \omega_n^2}$$

#### 1. 只改变振幅参数

陷波滤波器的衰减量(振幅)是由  $Z_{n1}$  及  $Z_{n2}$  的比值决定的。从图 4.17 可以看出,  $Z_{n1}$  及  $Z_{n2}$  的比值如果改变的话, 陷波滤波器的衰减量也会改变。图中  $20\text{dB}(+)$  及  $10\text{dB}(-)$  给了振幅项正的值, 同陷波滤波器的特性正好相反。按照这样的  $Z_{n1}$  及  $Z_{n2}$  的比率, 就可以任意设定陷波滤波器的衰减量了。



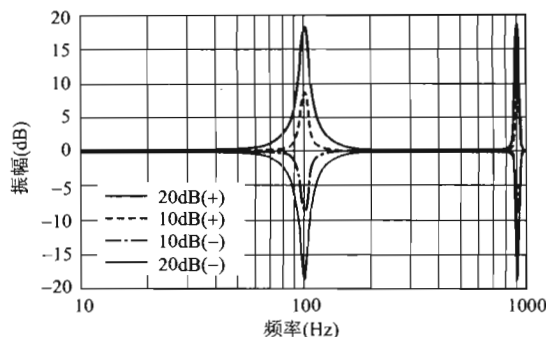


图 4.17 陷波滤波器的频率特性

## 2. 只改变衰减系数参数

如果这次固定振幅项为  $-20\text{dB}$ , 只改变衰减系数来观察的话, 可以从图 4.18 看出衰减系数从  $0.1$  到  $0.01$  变化时候的频率特性。陷波滤波器的中心频率及衰减量并没有变化, 只能看到特性的明显下降。

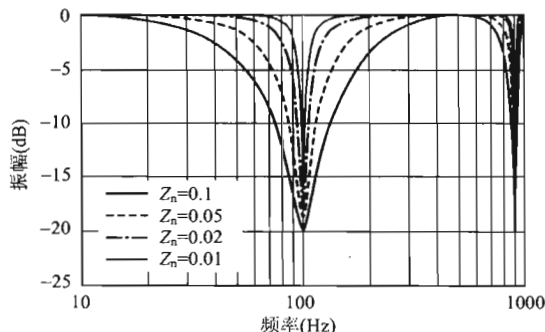


图 4.18 陷波滤波器的频率特性(改变衰减系数)

## 4.5.3 PID 控制器的特征

数字 PID 控制器的特性是由比例增益  $K_p$ 、积分增益  $K_i$  和微分增益  $K_d$  决定的。各自的系数变化的时候, 我们来看看频率是怎样变化的。

### 1. 只改变比例增益 $K_p$

图 4.19 表示了只改变比例增益  $K_p$  的例子。像这样如果只改变比例增益的话, 中间频率带的特性将发生变化。

### 2. 只改变积分增益 $K_i$

图 4.20 表示了只改变积分增益  $K_i$  的例子。像这样如果只改变积分增益

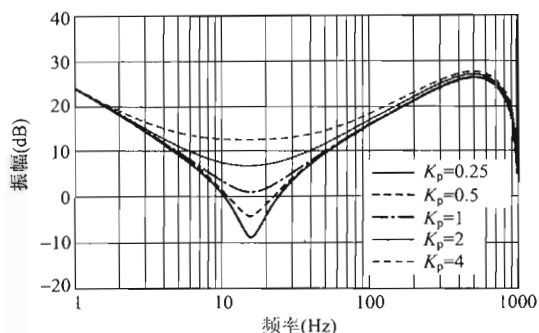


图 4.19 PID 控制器的频率特性(改变比例增益)

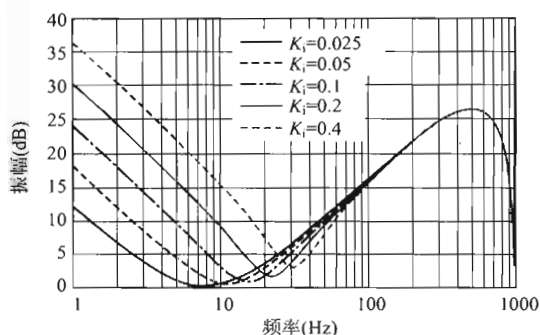


图 4.20 PID 控制器的频率特性(改变积分增益)

的话,低频率带的特性将发生变化。

### 3. 只改变微分增益 $K_d$

图 4.21 表示了只改变微分增益  $K_d$  的例子,像这样如果只改变微分增益的话,高频率带的特性将发生变化。

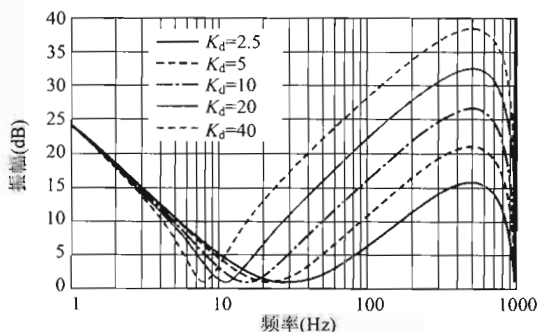


图 4.21 PID 控制器的频率特性(改变微分增益)

## 4. PID 控制器的特征

由于对影响比例增益、积分增益及微分增益的频带进行了区分,所以根据这些增益比例的变化,伺服系统的回路成形的自由度也会变高。

### 4.5.4 位置/速度环控制器的特征

位置/速度环控制器是由位置比例增益  $K_{pp}$ 、速度比例增益  $K_{pv}$ 、速度积分增益  $K_{iv}$  的特性决定的。根据每个增益的参数变化,我们来看看频率是怎么变化的。

#### 1. 只改变位置比例增益 $K_{pp}$

图 4.22 表示了只改变位置比例增益  $K_{pp}$  时候的例子。像这样,如果改变位置比例增益的话,位置/速度环控制器的低频带的特性将会发生变化。同 PID 控制器的积分项有同样的效果。

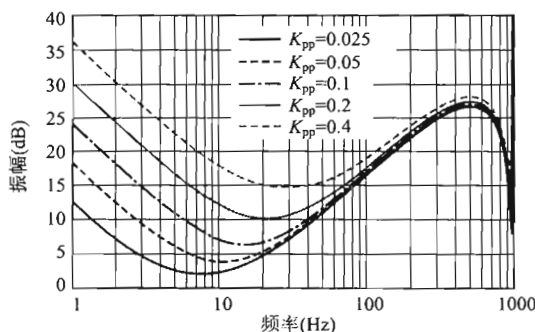


图 4.22 位置/速度环控制器的频率特性(改变位置比例增益)

#### 2. 只改变速度比例增益 $K_{pv}$

图 4.23 表示了只改变速度比例增益  $K_{pv}$  时候的例子。像这样,如果改变速度比例增益的话,位置/速度环控制器的高频率带的特性将会发生变化。同 PID 控制器的微分项有同样的效果。

#### 3. 只改变速度积分增益 $K_{iv}$

图 4.24 表示了只改变速度积分增益  $K_{iv}$  时候的例子。像这样,如果改变速度积分增益的话,位置/速度环控制器的低频率带的特性将会发生变化。同 PID 控制器的积分项有同样的效果。

#### 4. 位置/速度环控制器的特征

只关注频率特性的话,位置积分增益  $K_{pi}$  同速度积分  $K_{iv}$  有同样的特性。伺

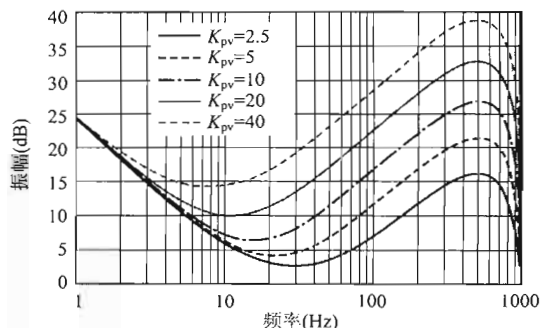


图 4.23 位置/速度环控制器的频率特性(改变速度比例增益)

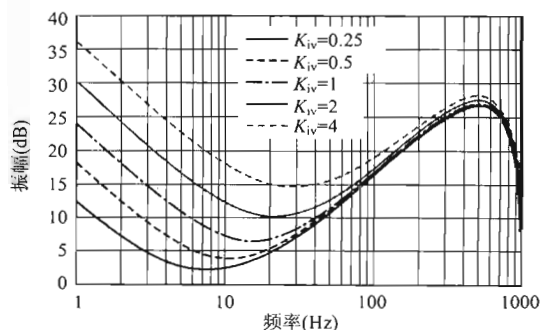


图 4.24 位置/速度环控制器的频率特性(改变速度积分增益)

服系统的回路成形的自由度比 PID 控制器的要差。即使不知道频率特性,时间轴响应也不容易产生超调量。因为本章的目的是说明数字滤波器设计帮助软件(DSP.exe)的使用方法,所以关于以上部分的详细说明将在本书的第 2 部分出现。

## 4.6 伺服控制器的系统设计

### 4.6.1 伺服控制器的构成

图 4.25 是伺服控制器的构成图。伺服控制器由目标轨迹生成器、反馈控制器、前馈控制器、输出滤波器,以及速度控制使用的微分器构成。

### 4.6.2 目标轨迹生成器

目标轨迹生成器的定义如下:在位置控制的时候,赋予新位置时,需要从现在位置找到新位置的轨迹,依照这个轨迹让电机旋转的时候,决定是否移动的模式。

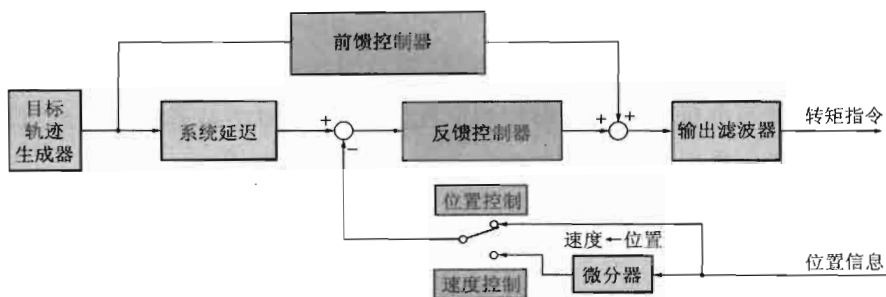


图 4.25 伺服控制器的构成

目标轨迹生成器的设计需要彻底理解加速度、速度，以及位置的关系。

图 4.26 表示了采用台形速度曲率的目标轨迹生成的过程，对加速度积分的话就得到了速度，对速度积分的话就得到位置，实际的目标轨迹生成器利用的就是这种演算过程。

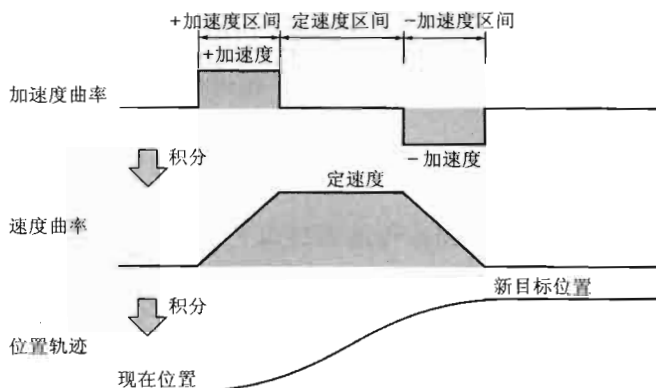


图 4.26 加速度、速度及位置的关系

### 4.6.3 系统延迟

系统延迟是把从目标位置生成器输出的信号通过伺服抽样延迟后，作为反馈的目标值。系统延迟比前馈系统更加具有效果。

在转矩控制放大器中，对于转矩电流指令来说，电机发生转矩会产生延迟。由于基于前馈电流的转矩产生也有延迟，所以电机的响应很慢。基于系统延迟，反馈的目标值如果与电机系统延迟一致的话，反馈偏差就会变小，比前馈系统的效果更好。

#### 4.6.4 反馈控制器

反馈控制器的定义如下:如果进行位置控制的话,比较目标轨迹生成器得到的目标位置与电机驱动部分得到的位置信息,由两者的差来决定电流值的大小。这部分我们采用了PID控制器与位置/速度环控制器。

#### 4.6.5 前馈控制器

前馈控制器是在反馈控制器之外工作的。反馈控制器是由于运动目标值的迟延产生了偏差,由此产生了电机的驱动力,所以一定会有延迟产生。而使用前馈的话,基于电机驱动电流偏差的反馈电流,再加上目标轨迹得到的反馈电流,就可以使运动中的跟随误差达到最小。

##### 1. 方框图

图4.27是前馈系统的方框图。对位置进行一次微分的话得到速度,用这个速度值同速度项增益  $K_{vel}$  进行积分演算。对速度进行一次微分得到加速度,同加速度增益  $K_{acc}$  进行积分演算。速度同加速度的积分演算的结果合起来就得到了前馈电流。

##### 2. 传递函数

图4.27的前馈系统方框图的传递函数如下

$$\frac{\text{FF 电流}}{\text{目标位置}} = (1 - z^{-1}) \cdot K_{vel} + (1 - 2 \cdot z^{-1} + z^{-2}) \cdot K_{acc}$$

前馈也是数字滤波器的一种,属于FIR(Finite Impulse Response)滤波器。

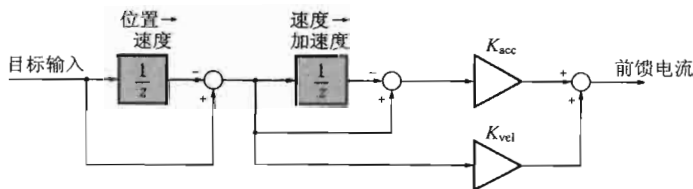


图 4.27 前馈系统方框图

#### 4.6.6 输出滤波器

使用数字滤波器的主要目的是为了抑制机械振动。数字滤波器使用了IIR(Infinite Impulse Response)滤波器,可以将双二次滤波器串联。

第 2 部分

---

应用篇





# 第 5 章

## 使用单片机控制交流 伺服电机的基础

在本章中,将会阐述使用单片机控制交流伺服电机的方法。电机的控制,并不一定要求使用类似于单片机或者 DSP 的微处理器。在家电里所使用的电机,绝大部分是用定制的硬件来控制的。但是,关于交流伺服电机的控制,绝大多数情况下,都要使用类似于单片机或者 DSP 的微处理器。

在本章中,所使用的电机控制处理器是瑞萨科技公司的 H8S/2367F,也就是所谓的 H8 处理器。在这个微处理器上所搭载的 H8S/2000 CPU 是拥有与 H8/300 CPU 及 H8/300H 向上兼容架构的 32bit 高速 CPU。

另外,如果只使用微处理器自带的功能是无法控制交流伺服传感器的全部功能,因此我们使用 CPLD CoolRunner II(XC2C256)作为外部的功能块。

### 5.1 H8S/2367F 和 CoolRunner II 的主要特征

#### 5.1.1 16 bit 高速 H8S/2000 CPU

H8S/200 CPU 的主要特征如下:

- (1) 与 H8/300 CPU 和 H8/300H CPU 在对象级别向上兼容。
- (2) 寄存器:16bit $\times$ 16(32bit $\times$ 8)。
- (3) 基本指令:65 种。
- (4) 频率:33MHz。

### 5.1.2 丰富的周边功能

主要的周边功能如下：

- (1) 内藏 flash ROM:384K Byte。
- (2) 内藏 RAM:24K Byte。
- (3) DMA 控制器(DMAC):4 通道。
- (4) 数据传输控制器(DTC):85 通道。
- (5) 16 位定时器脉冲单元(TPU):6 通道。
- (6) 程序脉冲产生器(PPG):1 通道。
- (7) 8 位定时器:2 通道。
- (8) 看门狗:1 通道。
- (9) I<sup>2</sup>C 总线接口:1 通道。
- (10) 10 位 A-D 转换器:10 通道。
- (11) 8 位 D-A 转换器:2 通道。
- (12) 频率发生器。

### 5.1.3 CoolRunnerII (XC2C256)的主要特征

赛灵思(Xilinx)公司的 CoolRunnerII 是最新的 CPLD,以消耗电力低和省空间著称。

- (1) 宏单元:256。
- (2) 核心电源电压:1.8V。
- (3) I/O 电源电压:1.5~3.3V。

## 5.2 单片机的使用方法

关于可以控制交流伺服电机的单片机的使用方法,我们将基于 H8S/2367F 进行阐述。

H8S/2367F 搭载了丰富的周边功能块。在交流伺服电机控制电路方面,如果巧妙地利用这些周边功能,可以大幅缩小电路的规模。

### 5.2.1 PWM 信号的产生方法

这里使用 16 位定时器脉冲单元(TPU)。在 TPU 里面拥有 PWM 工作模式,可以使用这个模式去产生 PWM 信号。

#### 1. PWM 的基本工作原理

PWM 是 Pulse Width Modulation 的简称,即脉宽调制。

图 5.1 描述了使用 TPU0 的情况下的 PWM 的基本工作原理。周期寄存器设置为 TRGA\_0,占空比寄存器设置为 TRGB\_0,然后,PWM 信号的输出设置为 TIOCB0,计数器 TCNT0 的时钟源为 CPU 时钟  $\Phi 1$ 。

计数器的动作非常简单,CPU 时钟  $\Phi 1$  增计数的话,同周期寄存器的比较匹配就会被清空,然后计数器增加一次。PWM 信号输出是在周期寄存器的比较匹配上设置为 low,同占空比寄存器的比较匹配输出设置为 High。

PWM 如图 5.1 所示,周期寄存器的周期一定的话,仅改变寄存器,即可改变输出脉宽。

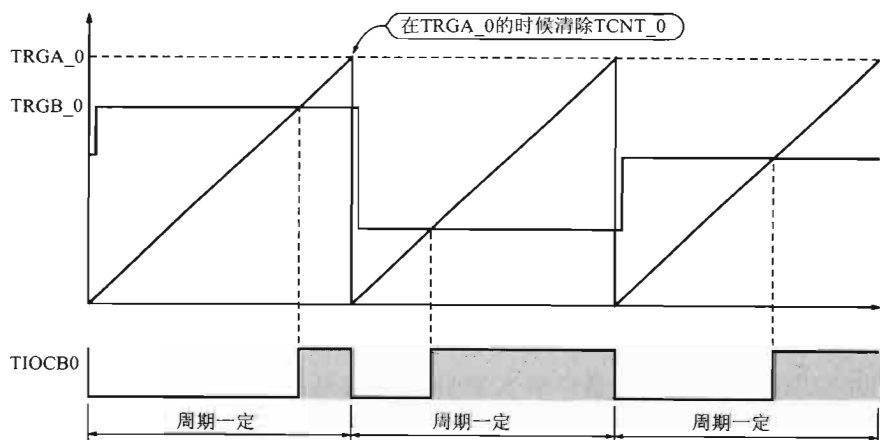


图 5.1 PWM 的基本工作原理

#### 2. 占空比寄存器更新时候需要注意的事项

在 PWM 上需要注意占空比寄存器的更新时刻。

图 5.2 所示的是占空比寄存器更新时刻不对,没有得到预期的 PWM 输出的例子。虽然在这个例子中占空比寄存器的设置同图 5.1 一样,但是图中,中央位置的 PWM 周期按照占空比寄存器设定的时候,计数器的值超过了占空比寄存器的值,没有发生比较匹配,所以就会得不到预期的 PWM 输出。

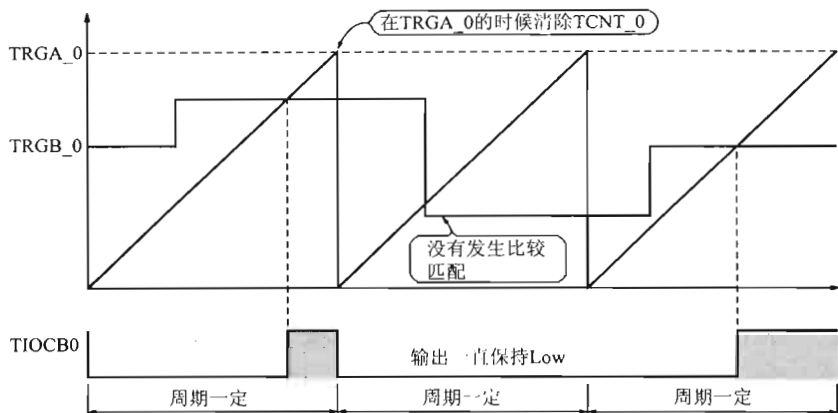


图 5.2 寄存器的更新时间很差的例子

### 3. 占空比寄存器的更新是同 PWM 周期同步的

在交流伺服电机的控制中, PWM 任务的值是由电流环的计算结果决定的。被决定的 PWM 任务值会马上保存在内存中, 一旦下次 PWM 周期寄存器的比较匹配的中断发生的话, 占空比寄存器本身也会更新。这个过程如果使用接下来要介绍的 DMA 以及 DTC 的话会非常方便。

### 4. PWM 的占空比规定了最大值和最小值

即使在占空比寄存器的更新中使用 DMA 或 DTC, 处理本身也需要花费 DMA 几个时钟或者 DTC 十几个时钟的时间。所以为了得到预期的 PWM 输出, 对于 PWM 的占空比我们规定了最大值和最小值。

## 5.2.2 DMA 的使用方法

H8S/2367F 在微处理器中确实被归类到高性能微处理器中去, 但是如果同 DPS 中的处理器比较的话, CPU 的能力相对低下。对于微处理器来说, 因为没有流水线及并行处理功能, 所以会感觉到时钟频率不同。但是, 因为微处理器搭载了丰富的周边功能, 如果充分利用这些功能和 DMA 以及接下来要说明的 DTC 的话, 可以设计出减轻 CPU 负荷的功能来。

### 1. 对于最优先的数据传送需要使用 DMA

因为 DMA (Direct Memory Access) 是完全的硬件处理, 所以可以非常高速地进行数据传输。如 PWM 占空比更新一样, 从内部 RAM 到内部寄存器的数据传输几个时钟就能完成。

## 2. DMA 的启动因素有制约

并不是说所有的 DMA 都是由中断引起的。表 5.1 中表示了中断原因、向量地址及中断优先度的顺序一览。H8S/2367F 的 DMA 中断原因是 SCi、TPU, 以及 A-D 转换器发来的中断信号。同时, 对于 TPU 来说, 只有从时钟寄存器 A 发来的中断才是唯一的原因, 因此, 5.2.1 节中 PWM 周期寄存器使用了 TRGA\_0。


表 5.1 中断原因、向量地址及中断优先顺序一览表

中断原因发生源	名 称	向量 编号	向量地址 高级模式	IPR	优先 顺序	DTC 启动	DMAC 启动
外部信号引脚	NMI	7	H'001C	—	高	—	—
	IRQ0	16	H'0040	IPRA14~IPRA12		○	—
	IRQ1	17	H'0044	IPRA10~IPRA8		○	—
	IRQ2	18	H'0048	IPRA6~IPRA4		○	—
	IRQ3	19	H'004C	IPRA2~IPRA0		○	—
	IRQ4	20	H'0050	IPRB14~IPRB12		○	—
	IRQ5	21	H'0054	IPRB10~IPRB8		○	—
	IRQ6	22	H'0058	IPRB6~IPRB4		○	—
	系统预约	23	H'005C	IPRB2~IPRB0	○	—	
		24	H'0060	IPRC14~IPRC12	—	—	
		25	H'0064	IPRC10~IPRC8	—	—	
		26	H'0068	IPRC6~IPRC4	—	—	
		27	H'006C	IPRC2~IPRC0	—	—	
		28	H'0070	IPRD14~IPRD12	—	—	
		29	H'0074	IPRD10~IPRD8	—	—	
		30	H'0078	IPRD6~IPRD4	—	—	
DTC	SWDTEND	31	H'007C	IPRD2~IPRD0		—	—
WDT	WOVI	32	H'0080	IPRE14~IPRE12		○	—
	系统预约	33	H'0084	IPRE10~IPRE8		—	—
刷新控制器	CM1	34	H'0088	IPRE6~IPRE4		—	—
	系统预约	35	H'008C	IPRE2~IPRE0		—	—
		36	H'0090	IPRF14~IPRF12		—	—
		37	H'0094			—	—
A-D 转换器	ADI	38	H'0098	IPRF10~IPRF8		○	○
	系统预约	39	H'009C		—	—	
TPU_0	TGI0A	40	H'00A0	IPRF6~IPRF4	○	○	
	TGI0B	41	H'00A4		○	—	
	TGI0C	42	H'00A8		○	—	
	TGI0D	43	H'00AC		○	—	
	TGI0V	44	H'00B0		—	—	
	系统预约	45	H'00B4		—	—	
		46	H'00B8		—	—	
		47	H'00BC		—	—	

续表 5.1

中断原因发生源	名 称	向量 编号	向量地址	IPR	优先 顺序	DTC 启动	DMAC 启动
			高级模式				
TPU_1	TGI1A	48	H'00C0	IPRF2~IPRF0	高 ↑	○	○
	TGI1B	49	H'00C4			○	—
	TGI1V	50	H'00C8			—	—
	TGI1U	51	H'00CC			—	—
TPU_2	TGI2A	52	H'00D0	IPRG14~IPRG12		○	○
	TGI2B	53	H'00D4			○	—
	TGI2V	54	H'00D8			—	—
	TGI2U	55	H'00DC			—	—
TPU_3	TGI3A	56	H'00E0	IPRG10~IPRG8		○	○
	TGI3B	57	H'00E4			○	—
	TGI3C	58	H'00E8			○	—
	TGI3D	59	H'00EC			○	—
	TGI3V	60	H'00F0			—	—
	系统预约	61	H'00F4			—	—
		62	H'00F8			—	—
		63	H'00FC			—	—
TPU_4	TGI4A	64	H'0100	IPRG6~IPRG4	○	○	
	TGI4	65	H'0104		○	—	
	TGI4	66	H'0108		—	—	
	TGI4	67	H'010C		—	—	
TPU_5	TGI5A	68	H'0110	IPRG2~IPRG0	○	○	
	TGI5B	69	H'0114		○	—	
	TGI5V	70	H'0118		—	—	
	TGI5U	71	H'011C		—	—	
TMR_0	CMIA0	72	H'0120	IPRH14~IPRH12	○	—	
	CMIB0	73	H'0124		○	—	
	OVI0	74	H'0128		—	—	
	系统预约	75	H'012C		—	—	
TMR_1	CMIA1	76	H'0130	IPRH10~IPRH8	○	—	
	CMIB1	77	H'0134		○	—	
	OVI1	78	H'0138		—	—	
	系统预约	79	H'013C		—	—	
DMAC	DMTEND0A	80	H'0140	IPRH6~IPRH4	○	—	
	DMTEND0B	81	H'0144		○	—	
	DMTEND1A	82	H'0148		○	—	
	DMTEND1B	83	H'014C		○	—	
	系统预约	84	H'0150	IPRH2~IPRH0	—	—	
		85	H'0154	IPRI14~IPRI12	—	—	
		86	H'0158	IPRI10~IPRI8	—	—	
		87	H'015C	IPRI6~IPRI4	—	—	

续表 5.1

中断原因发生源	名 称	向量 编号	向量地址	IPR	优先 顺序	DTC	DMAC
			高级模式			启动	启动
SCI_0	ERI0	88	H'0160	IPRI2~IPRI0		—	—
	RXI0	89	H'0164			○	○
	TXI0	90	H'0168			○	○
	TEI0	91	H'016C			—	—
SCI_1	ERI1	92	H'0170	IPRJ14~IPRJ12		—	—
	RXI1	93	H'0174			○	○
	TXI1	94	H'0178			○	○
	TEI1	95	H'017C			—	—
SCI_2	ERI2	96	H'0180	IPRJ10~IPRJ8		—	—
	RXI2	97	H'0184			○	—
	TXI2	98	H'0188			○	—
	TEI2	99	H'018C			—	—
SCI_3	ERI3	100	H'0190	IPRJ6~IPRJ4		—	—
	RXI3	101	H'0194			○	—
	TXI3	102	H'0198			○	—
	TEI3	103	H'019C			—	—
SCI_4	ERI4	104	H'01A0	IPRJ2~IPRJ0	—	—	
	RXI4	105	H'01A4		○	—	
	TXI4	106	H'01A8		○	—	
	TEI4	107	H'01AC		—	—	

### 5.2.3 DTC 的使用方法

DTC(Data Transfer Controller)的功能同 DMA 基本上一样。笔者也是在最初阅读数据表格时,产生了与 DMA 不同功能的单元到底在 DTC 中有没有的疑问,但是实际使用了以后才发现,确实是经过充分考虑以后才有的 DTC。DTC 的特征如下所示。

1. 对于 DTC 来说所有单元都可以作为启动中断的原因

表 5.2 所示是中断原因与 DTC 向量地址及对应的 DTCE。DTC 有自己的向量表。设定 DTC 的启动原因是由图中 DTCE 位设置为 1 后引起的。一旦发生了作为 DTC 的启动原因的中断,就会参照 DTC 的向量表格而不是软件中断的向量表格。

- ## 2. DTC 可以最多传输 85 通道的数据

对于 DTC 来说,一个硬件单元公用的话最多可以传输 85 个通道的数据。寄存器信息保存在特定的 RAM 位置(H'FFBC00~H'FFBFFF)。RAM 空间

大小为 1K Byte。因为 DTC 寄存器是 12Byte, 所以  $1024 \div 12 = 85$ , 最多可以传输 85 个通道的数据。

表 5.2 中断原因、DTC 向量地址及对应的 DTCE

启动原因的源头	启动原因	向量编号	DTC 向量地址	DTCE	优先顺序
软件	向 DTVECR 写入	DTVECR	H'0400+ (DTVECR[6:0]×2)	—	高
外部信号引脚	IRQ0	16	H'0420	DTCEA7	
	IRQ1	17	H'0422	DTCEA6	
	IRQ2	18	H'0424	DTCEA5	
	IRQ3	19	H'0426	DTCEA4	
	IRQ4	20	H'0428	DTCEA3	
	IRQ5	21	H'042A	DTCEA2	
	IRQ6	22	H'042C	DTCEA1	
	IRQ7	23	H'042E	DTCEA0	
A-D 转换器	ADI	38	H'044C	DTCEC6	
TPU_0	TGI0A	40	H'0450	DTCEC5	
	TGI0B	41	H'0452	DTCEC4	
	TGI0C	42	H'0454	DTCEC3	
	TGI0D	43	H'0456	DTCEC2	
TPU_1	TGI1A	48	H'0460	DTCEC1	
	TGI1B	49	H'0462	DTCEC0	
TPU_2	TGI2A	52	H'0468	DTCED7	
	TGI2B	53	H'046A	DTCED6	
TPU_3	TGI3A	56	H'0470	DTCED5	
	TGI3B	57	H'0472	DTCED4	
	TGI3C	58	H'0474	DTCED3	
	TGI3D	59	H'0476	DTCED2	
TPU_4	TGI4A	64	H'0480	DTCED1	
	TGI4B	65	H'0482	DTCED0	
TPU_5	TGI5A	68	H'0488	DTCEE7	
	TGI5B	69	H'048A	DTCEE6	
TMR_0	CMIA0	72	H'0490	DTCEE3	
	CMIB0	73	H'0492	DTCEE2	
TMR_1	CMIA1	76	H'0498	DTCEE1	
	CMIB1	77	H'049A	DTCEE0	
DMAC	DMTEND0A	80	H'04A0	DTCEF7	
	DMTEND0B	81	H'04A2	DTCEF6	
	DMTEND1A	82	H'04A4	DTCEF5	
	DMTEND1B	83	H'04A6	DTCEF4	
SCI_0	RXI0	89	H'04B2	DTCEF3	
	TXI0	90	H'04B4	DTCEF2	
SCI_1	RXI1	93	H'04BA	DTCEF1	
	TXI1	94	H'04BC	DTCEF0	低



续表 5.2

启动原因的源头	启动原因	向量编号	DTC 向量地址	DTCE	优先顺序
SCI_2	RXI2	97	H'04C2	DTCEG7	高 ↑ ↓ 低
	TXI2	98	H'04C4	DTCEG6	
SCI_3	RXI3	101	H'04CA	DTCEG5	
	TXI3	102	H'04CC	DTCEG4	
SCI_4	RXI4	105	H'04D2	DTCEG3	
	TXI4	106	H'04D4	DTCEG2	

### 3. DTC 的动作时序

图 5.3 表示了 DTC 的动作时序。一旦发生了作为 DTC 的启动原因的中断,就会参照对应于这个中断的向量表格。接下来,对于表格中展示的特定 RAM 位置中得到的 DTC 寄存器信息(图中读传送信息的部分)要在 DTC 单元中展开。虽然 DTC 寄存器是 12Byte,但是内部 RAM 是同 32Byte 宽的总线相连接的,所以传送信息读时间是 3 个时钟。根据 DTC 寄存器的内容,进行目的的数据传送。数据传送结束以后,DTC 寄存器的信息就会返回到原来 RAM 空间(图中传送信息写的部分)。

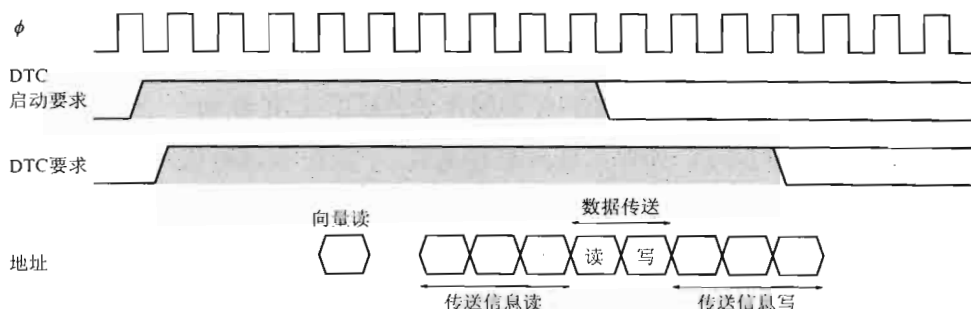


图 5.3 DTC 动作时序图

### 4. 与 DMA 的不同

与 DMA 的不同是,向量的参照及传输信息的读和写的部分的处理时间。但是基于软件中断的数据传输可以高速化。

## 5.2.4 环路计算用的时基的产生方法

对于交流伺服电机的控制来说,电流环计算用及伺服环计算用的两个时基是必要的。因为 8 位计时器(TMR)有 2 个通道,所以我们基于这两个时基

进行。

### 1. 电流环用时基

电流环用的时基是  $100\mu\text{s}$  ( $10\text{kHz}$ )，时钟计数器 TCNT\_0 的时钟源是  $\Phi/64$ ， $\Phi$  是  $32\text{MHz}$ ，所以  $32\text{MHz} \div 64 \div 10\text{kHz} = 50$ ，因此我们设定时间常数寄存器 TCORA\_0 为 49。

### 2. 伺服环用时基

伺服环用的时基是  $1\text{ms}$  ( $1\text{kHz}$ )，时钟计时器 TCNT\_1 的时钟源是 TCNT\_0 的比较匹配 A，所以  $10\text{kHz} \div 1\text{kHz} = 10$ ，因此我们设定时间常数寄存器 TCORA\_1 为 9。

这样设定的话，电流环的中断处理同伺服环的中断处理完全同步，对于电机的控制来说非常有利。

## 5.2.5 中断控制器的使用方法

H8S/2367F 微控制器的中断控制器是由两种中断控制模式支持的。这次使用了中断控制模式 2。这个模式除了 NMI 以外针对各中断原因设定了 8 级的优先顺序。

### 1. 中断控制器的优先顺序对 DMA 及 DTC 没有影响

对于 DMA 与 DTC 这样的硬件处理来说，中断控制器的优先顺序对他们没有影响。举例来说，即使把 DMA 的启动原因的中断的优先顺序设定为最低级别，DMA 也是最优先处理的。

### 2. 伺服环计算考虑到了周期失控

电流环计算比伺服环计算的中断优先顺序要高。电流环计算同伺服环计算的周期虽然不同，但是由于取得了完全同步，所以中断标志位在同一个过程中可以被设定。这样的话，就会先处理优先顺序高的电流环计算，保留伺服环计算。一旦电流环计算结束，伺服环计算的中断处理就会被执行。

也就是说，在伺服环计算之前必定先进行电流环计算，虽然发生了延迟，但是计算周期本身并没有失控。因为伺服环计算中包括了微分计算和速度计算，所以计算周期不会失控，对于电机的控制来说非常重要。

### 5.2.6 增计数的产生方法

16 位定时器脉冲单元 TPU 对应增量编码的一般接口使用的 2 相(A,B)信号。如图 5.4 所示,TPU 作为相位计数模式 1 的时候,A 信号与 TCLKA 相连接,B 信号与 TCLKB 相连接,这样就形成了增计数。相位计数模式 1 的增/减条件如表 5.3 所示。相位计数模式 1 因为包括了 A 信号和 B 信号的上升沿和下降沿的两个边沿,所以得到的计数值是编码器的时钟的 4 倍。

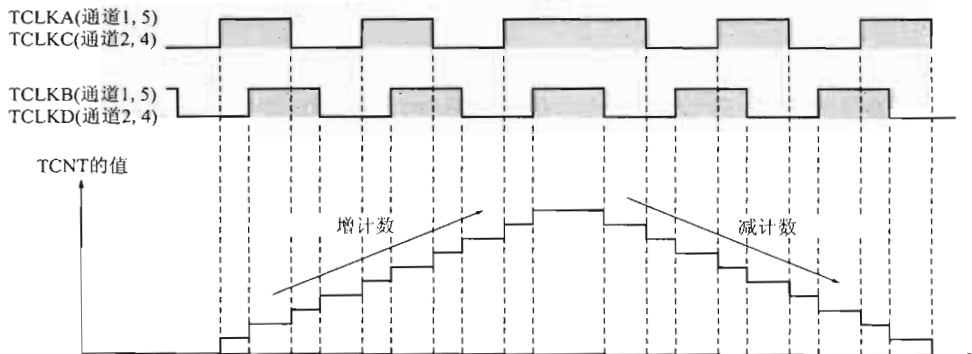


图 5.4 相位计数模式 1 的动作例子

表 5.3 相位计数模式 1 的增/减计数条件

TCLKA(通道 1,5) TCLKC(通道 2,4)	TCLKB(通道 1,5) TCLKD(通道 2,4)	动作内容
高电平	↓	增计数
低电平	↓	
↑	低电平	
↓	高电平	
高电平	↓	减计数
低电平	↑	
↑	低电平	
↓	高电平	

【符号说明】 ↓ : 上升沿, ↓ : 下降沿

TPU 是 16 位计数器,对于增计数来说可能会感觉稍显不足,这个时候我们会使用软件方式的 32 位扩展功能。

### 5.2.7 实时通信接口(SCI)的使用方法

H8S/2367F 微控制器搭载了 5 通道的实时通信接口(SCI)。这个接口是为了与上位主机通信及电机轴间的同步控制而使用的。H8S/2367F 的异步通信式 SCI 的最高传输速率是 1Mbps,用这个传输速度的话也会产生延迟收发通信。所以我们在这介绍 CPU 占用率低的几个方法。

#### 1. 由 DMA 来处理 SCI 的收发通信

传送率为 1Mbps 的收发通信是很困难的。通过软件而进行中断处理的情况下,因为每隔  $10\mu\text{s}$  发送信息以及接收信息都要产生中断,所以仅此就占用了 CPU。为了不占用 CPU,SCI 的接收信息及发送信息就完全地交给了 DMA。

对于 DMA 来说,发信息用及收信息用的两个通道功能分摊使用了顺序模式。DMA 设定的传送字节完成了的话,CPU 端就会产生中断。

#### 2. 有必要注意 SCI 接收信号的错误处理

SCI 的 DMA 传输容易出问题的是接收信号的错误处理。一旦发生了奇偶错误或者帧差错,将不能进行之后的接收信号处理。如果不进行适当的错误处理的话,系统将会陷入无法通信的状态。

#### 3. SCI 的接收信号错误处理例

使用 DMA 进行 SCI 的收发通信的时候,如果无法与传送字节数相匹配的话,将无法产生来自 DMA 的传送完毕的中断信号。SCI 的接收信号错误处理是指,放弃发生错误的指令数据,为了使得下一个指令可以正常地进行接收处理,有必要重新设定 DMA 的设定值。

(1) 清除错误标志。

(2) 读取 DMA 的传送计数器(ETCRA 或者 ETCRB),确定剩下的传送数。

(3) 无效 DMA。

在此之后接收信息的时候,会对 CPU 产生中断,计算接收的字节数,在第二步所确定的剩余传送数变为 -1 后,这个值会被再次初始化,等待下一个指令的到来。

## 5.3 PLD 的使用方法

PLD 是 Programmable Logic Device 的简称,即可编程逻辑器件。在这里要注意同 PLD 很大区别的 CPLD(Complex Programmable Logic Device)、FPGA(Field Programmable Gate Array)。

### 5.3.1 CPLD 及 FPGA

CPLD 是复杂可编程器件的简称,由 PAL 和 GAL 器件发展而来。与 FPGA 相比,构造非常简单,延迟时间也非常容易预测。此外,设备的构造是以 Flash 及 EEPROM 为基础的,因此不需要同外部电路内容保持一致的 ROM。鉴于以上特点,CPLD 常用于电路板的系统。

FPGA 是 Field Programmable Gate Array 的简称,主要在大规模电路中使用。FPGA 是由很小的逻辑模块组成,芯片内部自由连接部分非常多,所以自由度很高,可以实现大规模并且高速化的电路。

对于交流伺服电机的控制回路的设计来说,没有必要使用大规模电路,所以选择了 CPLD 的 Cool Runner II(XC2C256)。此外其他的选定理由是,FPGA 必须使用用来配置的 ROM,对于这个 ROM 来说在基板上必须有一定的空间,非常浪费,我们需要对制造商谴责的是,这样的设计使得价格非常高。

### 5.3.2 CPLD 的设计工具

CoolRunnerII(XC2C256)的设计工具 ISE WebPACK 是可以在 Xilinx 公司的主页上免费下载到的。最新的版本里电路图入口功能也具备了,对于笔者这种不使用 VHDL 的设计者来说非常有帮助。

### 5.3.3 设计中的注意点

Xilinx 公司的 CPLD(CoolRunnerII 以及 XC95xxxXL)可以同 BUS-Hold 及 Keeper 的这种小型三极管构成的门电路相连接。这种 BUS-Hold 的特点是即使在高阻抗状态下输入仍然稳定。以前,XC95144XL 的输入与集电极开路输出的上电复位电路相连接时,会产生不能解除复位的情况。当时使用的设计工具是 Foundation3.2,发生了不能禁用 BUS-Hold 的情况。此外,上电复位电路的上拉电阻也不能因确保复位时间而变小,也就是说电路进入了死机状态。

幸运的是无偿设计工具 WebPACK4.2 而使 BUS-Hold 禁用。

另外,最近我将 Altera 公司的 MAXII 与前面说过的同样的上电复位电路连接时,有过 20ms 的复位时间变成 4ms 的麻烦。调查原因才发现,MAXII 中虽然有 CPLD 的位置,但是芯片内部同有 ROM 的 FPGA 一样,接入电源的时候会进行配置。这个时候,I/O 引脚会产生  $20\Omega$  的上拉电阻,基于这个上拉电阻,复位时间变得失控了。

“怎么发生了这样低级的错误?!",给制造商打电话询问后才知道这个上拉电阻不能禁用,所以笔者只能挥泪把它去掉了。

CPLD 及 FPGA 的电路可以互换,产品线也很丰富。但是,缺少 I/O 引脚的选择功能的详细说明以及配置中的 I/O 引脚的动作说明,这样就会在设计中产生致命的错误,这点需要注意。

## 5.4 交流伺服电机控制回路的系统设计

虽然涉及基于微控制器的交流伺服电机控制回路的系统设计,但是在本章只着重于说明基于微控制器的控制部分。驱动电路的硬件部分会在下一章说明。

### 5.4.1 设计目标

现列举控制电路的设计概要:

- (1) 电机驱动方式:三相正弦波 PWM 驱动。
- (2) 自增计数器:1 通道。
- (3) PWM 频率:20kHz。
- (4) 电流环频率:10kHz。
- (5) 伺服环频率:1kHz。
- (6) 电机电流检出频率:130kHz。

### 5.4.2 三相正弦波 PWM 驱动

对于三相正弦波 PWM 驱动来说需要 6 根 PWM 信号。因为详细的说明已经在第 2 章讲解过了,所以如果想详细了解这方面的知识的话,请在读接下来的文章之前翻阅第 2 章的内容。

将 TPU 分配给三相正弦波 PWM 驱动。TP0 及 TPU3 的定时器通用寄存器有 4 个,将 TRGA 作为周期寄存器,余下来的 TRGB、TRGC、TRGD 作为占空比寄存器使用的话,各自可以制作 3 根 PWM 信号(合计 6 根)。微处理器使用的引脚如图 5.5 所示,有 35 脚(TIOCB0)、36 脚(TIOCC0)、37 脚(TIOCD0)、43 脚(TIOCB3)、44 脚(TIOCC3)、45 脚(TIOCD3)。

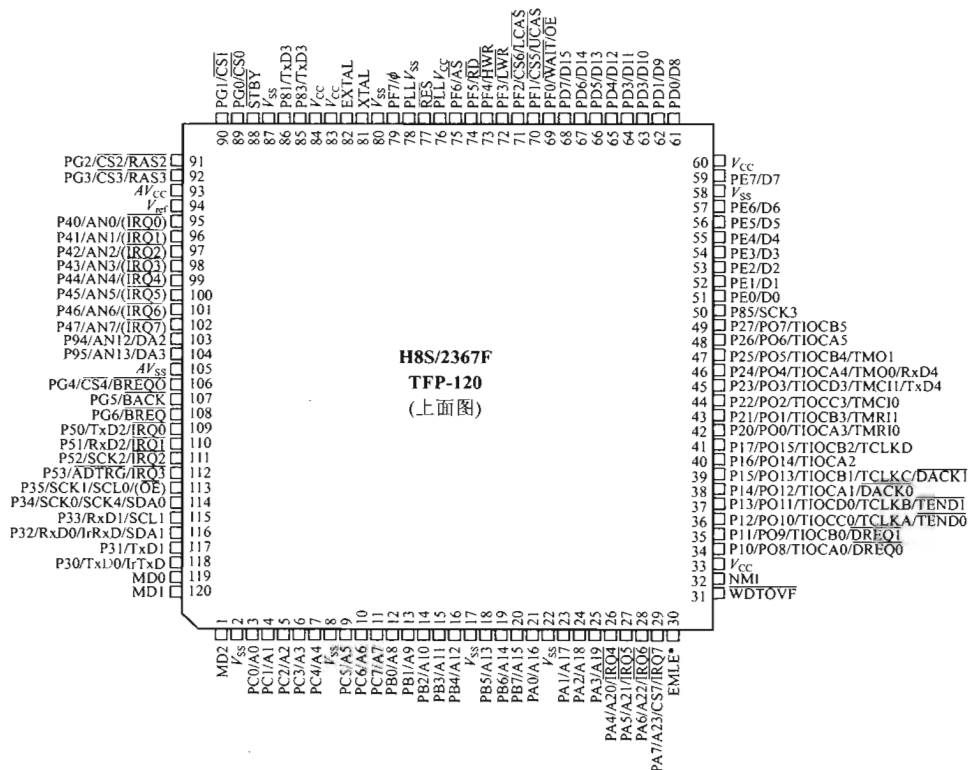


图 5.5 H8S/2367F 的引脚配置图

### 5.4.3 自增计数器

TPU 的相位计数模式的输入是 TCLKA 与 TCLKB 的组合,或者 TCLKC 与 TCLKD 的组合(参照表 5.3)。仔细看图 5.5 中 H8S/2367F 的配置图的话, TCLK0 同 TIOCC0, TCLKB 同 TIOCD0 共用,因为作为前项的 PWM 输出使用过了所以现在不能再使用了。自增计数器用的相位计数模式是 39 脚(TCLKC)、41 脚(TCLKD)。TPU 分配中,TPU2 或者 TPU5 哪一个都可以使用,这里我们使用 TPU2。

#### 5.4.4 PWM 频率

PWM 频率是比人类的可听到的范围更高的频率,一般来说是 20kHz。因为时钟计数器的时钟源是 32MHz,所以周期寄存器的值变为了  $32\text{MHz} \div 20\text{kHz} = 1600$ ,因为时钟计数器要同步清除,所以周期寄存器的值设定为 1599。

#### 5.4.5 电流环频率

电流环的频率是 10kHz,这个中断处理的 CPU 占用率要控制在 30% 之内。换成时间就是  $33\mu\text{s}$ 。但是,从最开始就去预测电流环计算的处理时间是一件很困难的事情。有必要明确计算内容,而且也有必要正确把握 CPU 的计算能力。笔者认为最开始控制电机的处理时间并不可读。因为对设备用途和成本要求不是那么严格,所以当时使用了最高速度的浮点 DSP。

电流必要的计算绝对不能少。对于单片机级别的 CPU 有点苛刻。但是如果通过各种手段实现的话,可以很大程度上节省成本,因此这次我们挑战性地使用单片机去控制。在这里,我们整理了电流环必要的计算:

(1) 算出转子位置,从 LUT(Look Up Table)里面取得  $\sin$  和  $\cos$  的值。

(2) 读入 U 相和 V 相的电机电流,移动平均处理后,进行增益和偏置的修正。

(3) 对于各相电机电流,进行  $\alpha\beta$  变换,再进行  $d-q$  变换。

(4) 把从伺服控制器得到的转矩指令和从(3)得到的转矩  $q$  进行对比,并进行转矩 PI 的控制计算。

(5) 把励磁指令(指令值 0)和(3)得到的励磁  $d$  进行比较。

(6) 对于转矩和励磁的 PI 计算结果,先进行  $d-q$  的变换,然后进行  $\alpha\beta$  的变换。

(7) 算出各相(U,V,W)的 PWM 设定值。

这些计算全部都是整数计算,所以即使是单片机(H8S/2367F)也可以实现。

#### 5.4.6 伺服环频率

伺服环计算和电流环计算是相同的,在这里,我们计划将中断处理的 CPU 占有率控制在 30% 以内。但是预测处理时间是比较困难的。下面整理伺服环



的处理内容如下：

- (1) 读取编码计数器,计算位置和速度。
- (2) 进行差分方程式(PID 或者位置速度环)的计算。
- (3) 进行前馈计算。
- (4) 输出滤波器的计算。
- (5) 轨迹计算。
- (6) 错误处理。

这些处理对于伺服控制器而言是全部的内容,但并不是所有的这些处理是必需的。比如说,伺服控制器和状态控制器连接之后,从外部获取位置指令和速度指令后,(5)的轨迹计算就没有必要了。还有(3)的计算以及(4)的输出滤波器的计算也并不是必要的,作为选择项而已。

伺服环的频率对系统的性能会产生很大的影响,因此对于自身的系统而言,有必要认真考虑哪些功能是必需的。交流伺服电机控制的情况下,伺服环频率是数千赫,在本书中,因为介绍一体化系统,所以安装了全部的功能并且设定伺服环的频率为 1kHz。

#### 5.4.7 电机电流检出频率

电流检出频率和使用的 IC 相关。在本书中,所介绍的电机电流检出 IC 是 IRF-Japan 公司的 IR2175S。这个 IC 大概可以测定 130kHz 的电机电流。因为要测定 U 相和 V 相的电机电流,所以需要使用 2 个 IC。IR2175S 的电流检测是使用了 PWM 的输出,在 CPLD 的内部设计时钟计数器。如果按照这个处理过程计算 CPU 占用率的话,就得到了以下的结果:

- (1) 电流检出频率: $1 \div 130\text{kHz} = 7.6723\mu\text{s}$ 。
- (2) DTC 数据传送所需要的时间: $15 \div 32\text{MHz} = 0.46875\mu\text{s}$ 。
- (3) CPU 的占有率: $0.46875\mu\text{s} \div 7.6723\mu\text{s} \times 2 \text{ 通道} = 12.2\%$

只是数据读取的话,对于单片机而言读取 130kHz 的 2 通道的数据是比较严峻的。如果可以的话,我们想使用 DMA,但是因为外部中断口(IRQ 端子)无法启动 DMA,所以在本章中我们决定使用 DTC。

# 第 6 章

## 驱动交流伺服电机的三相 PWM 控制回路的实践

电机的种类非常多,驱动方法也很多,全部的方法都介绍是不可能的,在这里,我们从具体的电路的例子着手主要对交流伺服电机的驱动方法进行介绍。从本质上来说,不管是哪种驱动方式,电机驱动必要的知识和经验都是共通的。

### 6.1 设计参数

三相 PWM 控制电路的参数如下所示。

#### 6.1.1 控制部分的参数

- (1) 控制微处理器:H8S/2367F。
- (2) PLD:CoolRunnerII(XC2C256)。
- (3) 控制部分电源:3.3V。

#### 6.1.2 驱动部分的参数

作为设计对象的电机是 maxon motor 公司的无刷电机 EC32 80W 48V (118890)。这个电机的额定电流是 1.6A,额定电压是 48V,瞬间最大电流是额定电流的 3 倍以上。

- (1) 驱动方式:三相正弦波 PWM 驱动。
- (2) 驱动部分的电源电压:最大 48V。

(3) 瞬间最大电机电流:6A。

## 6.2 使用部件及电路的选定

### 6.2.1 MOSFET 的确定

电机驱动电路中功率器件的代表是双极晶体管、MOSFET、IGBT(Insulated Gate Bipolar Transistor)。根据使用目的的不同,低电压/低输出时候用双极晶体管和 MOSFET,高电压/高输出时候使用 IGBT。这次我们介绍使用 MOSFET 的电机驱动电路。

MOSFET 的种类很丰富,看数据表格的话有很多的项目,很多人会为了不知道选哪个而烦恼。确定电机驱动电路时,MOSFET 的选择要点如下所示。

#### 1. 源漏耐压( $V_{dss}$ )

电机驱动电路中,考虑电机线圈的电感的浪涌电压及电机的感应电流,需要驱动电源电压的 1.5~2 倍的耐压。图 6.1 的驱动电路中驱动电压为 48V,使用  $V_{dss}=75V$  的 MOSFET。

#### 2. 漏极电流( $I_d$ )

虽然有必要设置大于电机平常使用的电流的耐压电流,但是具体需要多大的电流视情况而定。以图 6.1 的驱动电路为例,最大电机电流 6A 的话,应使用  $I_d=29A$ (周围温度  $100^{\circ}C$ ) 的 MOSFET。理由是为了设计电路板的方便起见,不能使用很大的散热片,所以为了抑制 MOSFET 的发热而使用该电流的 MOSFET。

#### 3. 输入电容( $C_{iss}$ )及反馈电容( $C_{rss}$ )

对于  $V_{dss}$  及  $I_d$ ,如果选定大容量 MOSFET 的话输入容量变大是必然的结果,所以作为选择 MOSFET 的要点,输入电容同反馈电容的比值作为选择大容量 MOSFET 的基准。图 6.1 中 IR 公司的 IRFU2407,  $C_{iss}=2400pF$ ,  $C_{rss}=77pF$ 。此外,栅极处外接了  $1000pF$  的电容,这种设计的目的是为了使高压侧 MOSFET 在 ON 的时候抑制低压侧栅极电压上升。

#### 4. 反向恢复时间( $t_{rr}$ )

电机驱动电路中的 MOSFET 内的体二极管起着续流二极管的作用,所以

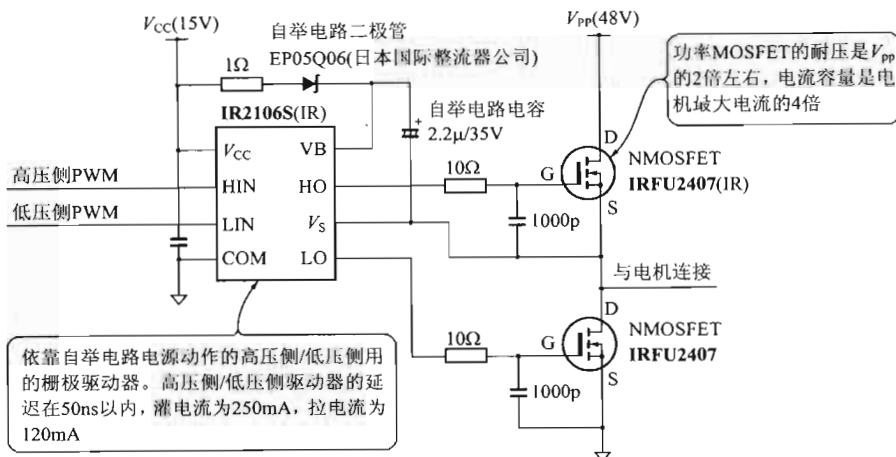


图 6.1 电机驱动电路

选择了反向恢复时间比较短的产品，能够抑制电机线圈中的浪涌电压。IRFU2407 的反向恢复时间是 100ns。

## 6.2.2 MOSFET 栅极驱动电路

图 6.1 的电路中作为 MOSFET 的栅极驱动 IC，使用 IR 公司的 IR2106S。这个 IC 可以驱动高压侧/低压侧的 N 沟道 MOSFET 的栅极。

### 1. 自举电路电源动作

IR2106S 的特征是当消耗电流很少的时候不需要特别的浮充电源，只需要以二极管与电容构成的自举电路电源。二极管使用了日本 Inter 公司的肖特基二极管。

### 2. 输入级别是 TTL

PWM 输入是 TTL 级别。基于自举电路的浮充电源，没有使用光电耦合器。高压侧/低压侧驱动管的延迟时间在 50ns 之内，灌电流是 250mA，拉电流是 120mA。

### 3. 空载时间是 500ns

本电路中空载时间设计为 500ns。高压侧 PWM 及低压侧 PWM 的空载时间都是 500ns 的非重叠 PWM 信号。

### 6.2.3 电机电流检测电路

图 6.2 的电路中电流检测 IC 使用了 IR 公司的 IR2175S。IR2175S 是在这种情况下使用非常方便的一款 IC。大家可能看到图就明白了,一个 IC 芯片里面只有很少几个电阻和电容就可以检测出电机电流。这个 IC 的输出是 130kHz 的 PWM 信号。因为输出是数字信号,不需要担心噪声的问题,可以直接从微处理器的计时器端子上进行信号控制。只是,这次为了方便使用微处理器内部的计时器,CPLD(CoolRunnerII)中设计了电流检测计时器电路。

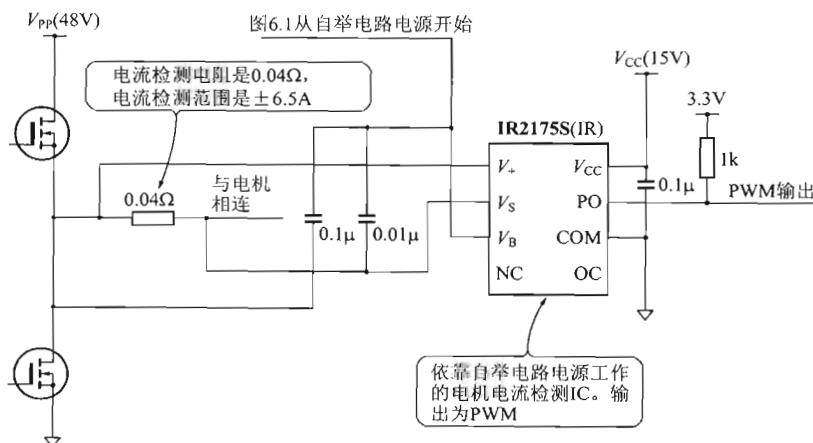


图 6.2 电机电流检测电路

#### 1. 电流检测范围是 ±6.5A

IR2175S 的电压测定范围是 ±0.26V, 电流检测电阻是 0.04Ω, 因此电流检测范围是  $\pm 0.26\text{V} / 0.04\Omega = \pm 6.5\text{A}$ 。

#### 2. 必须有偏置修正

IR2175S 电压测定的偏置误差是 ±0.01V, 电流检测电阻是 0.04Ω, 因此偏置误差是  $\pm 0.01\text{V} / 0.04\Omega = \pm 0.25\text{A}$ 。因为不能无视这个误差, 所以偏置修正是很必要的。

#### 3. 必须有增益修正

IR2175S 的电流检测输出是 PWM 信号。从数据表中看, PWM 频率是 130kHz(typ), 最小值是 95kHz, 最大值是 165kHz, 频率变化导致增益误差, 所以需要增益修正。

## 6.3 交流伺服电机驱动硬件电路

交流伺服电机的驱动硬件电路如图 6.3 所示。使用瑞萨公司的 H8S/2367F 单片机。CPLD 是使用了 Xilinx 公司的 CoolRunnerII(XC2C256)。设计的对象电机是 maxon motor 公司的无刷直流电机 EC32 80W 48V(118890)。这个电机内藏三相霍尔 IC 和增量式编码器。

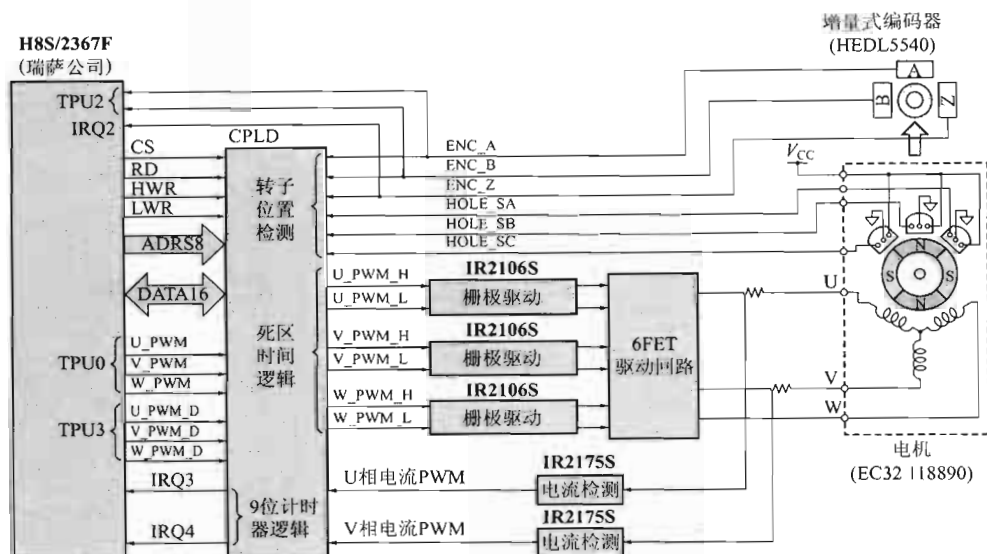


图 6.3

图中,相当于栅极驱动与 6FET 驱动电路的部分如图 6.1 所示,是三相正弦波 PWM 驱动。此外,电流检测是通过直接测量电机线路 U 相与 V 相得到的,相当于这部分的回路如图 6.2 所示。

### 6.3.1 使用微控制器内置的 TPU 生成三相 PWM 的方法

用 TPU(Timer Pulse Unit)产生驱动用的三相正弦波 PWM。由于 TPU0 和 TPU3 有四个通用计时寄存器,除去 TRGA 作为周期寄存器,TRGB、TRGC、TRGD 作为占空比寄存器,可以分别组成三个 PWM 信号(总共六个)。

图 6.4 是三相 PWM 的时序图。TPU0 与 TPU3 基本上是一起动作的。TRGA 在比较匹配之后清除计数器。另外同时将三相的 PWM 输出置为低电平。TRGB、TRGC、TRGD 分别比较匹配之后,分别对应的 PWM 输出设定为高电平。

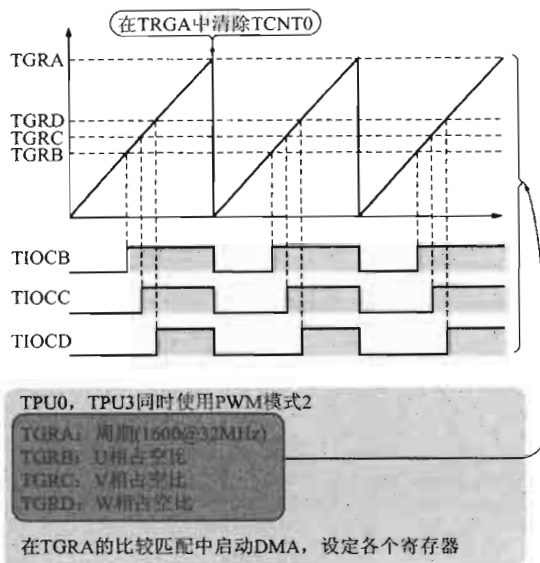


图 6.4 三相 PWM 的时序图

### 6.3.2 非重叠 PWM 信号

虽然 TPU0 和 TPU3 基本上是一起动作的,但是有一点是不一样的。TPU3 的计时器开始时间相对于 TPU0 来说慢了 500ns。因此,TPU3 的 PWM 输出相对于 TPU0 的输出有 500ns 的延迟,这个延迟称为死区时间。

如图 6.5(a)所示,用 CPLD 画出死区时间电路,生成高压侧 MOSFET 栅极用 PWM 信号和低压侧 MOSFET 栅极用 PWM 信号。图 6.5(b)是用时序图所表示出来的死区时间。死区时间是为了使高压侧和低压侧的三极管不会同时开启而设置的同时关闭的时间。由于微控制器内部生成的就是被延迟的 PWM 信号,所以 CPLD 电路简单了很多。

图 6.6(a)所示的是用于检测电机电流的电路构成。电流检测 IC 的 IR2175S 是输出 PWM 频率为 130kHz 的高速型,时钟频率为 32MHz,因此,  $32\text{MHz} \div 130\text{kHz} = 246$ 。

虽然电流的计时器有 8 位,足够计算用了,但是由于 IR2175S 的 PWM 频率随着温度或者电压等会有变动,所以设置了 9 位的计时器。MCU 的数据大小是 16 位,高位的 8 位设置为 PERIOD(周期),低位的 8 位设置为 DUTY(占空比)。PERIOD 数据向右移动一位,向 MCU 通知  $\frac{1}{2}$  PERIOD 数据。

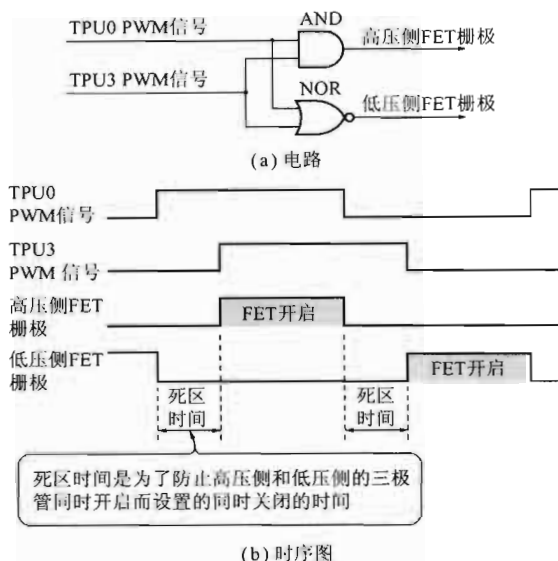


图 6.5 产生死区时间的电路

图 6.6(b)是电机电流检测用计时器电路的时序图。用于电流检测的 PWM 信号经过边沿检测电路之后,可以生成向上的脉冲和向下的脉冲。向下的脉冲用于 PERIOD 测定,向上的脉冲用于 DUTY 测定。

根据从电流检测中得到的 PWM 信号的向上脉冲,中断 MCU,进行数据的读取。

图 6.6(c)是针对 DTC(Data Transfer Control)动作和 MCU 的内部处理的说明。各个电流检测中执行的中断是在 130kHz 间隔内发生的。由于用软件进行中断处理的时候对 CPU 的占用率比较高,所以由 DTC 进行数据的转送。DTC 是利用重复的动作,读取电流检测计数器的 PERIOD/DUTY。一次数据传送需要的时间大概为  $0.4\mu\text{s}$ (15clock @ 32MHz)。MCU 内部准备有 13 个变量,用来保存新的 13 次的数据。在 MCU 内部的处理中,13 个变量用来表示各个相位的电机的电流。另外,由于 PERIOD 的数据也被保存起来了,用于计算电机电流值的周期数据也会被定期地更新。

用 PWM 驱动的电机电流随着 PWM 频率而波动。由于 PWM 的频率为 20kHz,将检测出的电流(130kHz)进行 13 次积分之后正好等于 10kHz,正好使电机电流的波动看不到。因此,将电流反馈周期定为  $100\mu\text{s}$ (10kHz)。



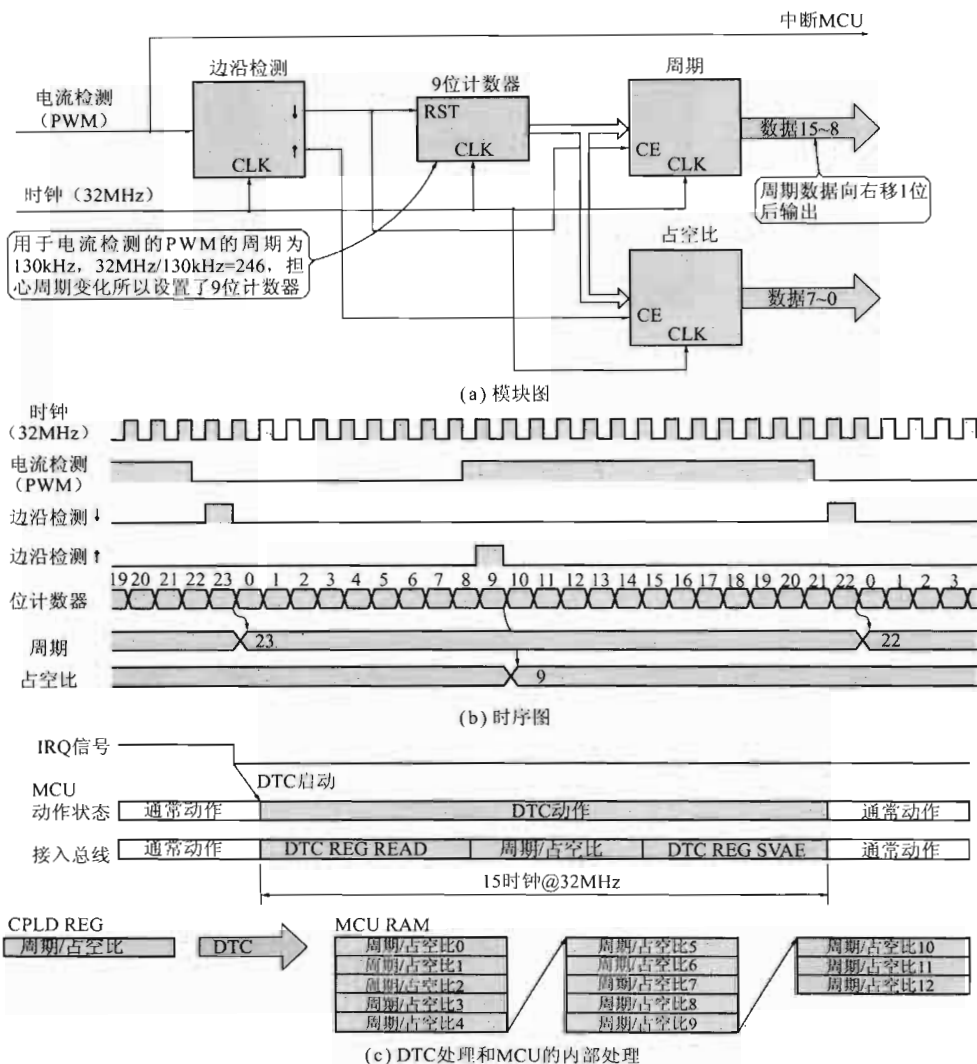


图 6.6 电流检测用的计时电路的组成和动作

### 6.3.3 转子位置检测电路(CPLD)

作为本电路的转子位置检测方法,采用了霍尔传感器和增量式编码器。增量式编码器会有断电之后存储的位置信息会消失的缺点,所以转子的位置就无法确定。在这里,使用霍尔传感器初始化增量式编码器,求得转子的位置。

图 6.7(a)表示的是转子位置检测电路的构成。转子位置检测电路是由判别转动方向的电路、转子位置计数器和转子原点检测电路组成的。

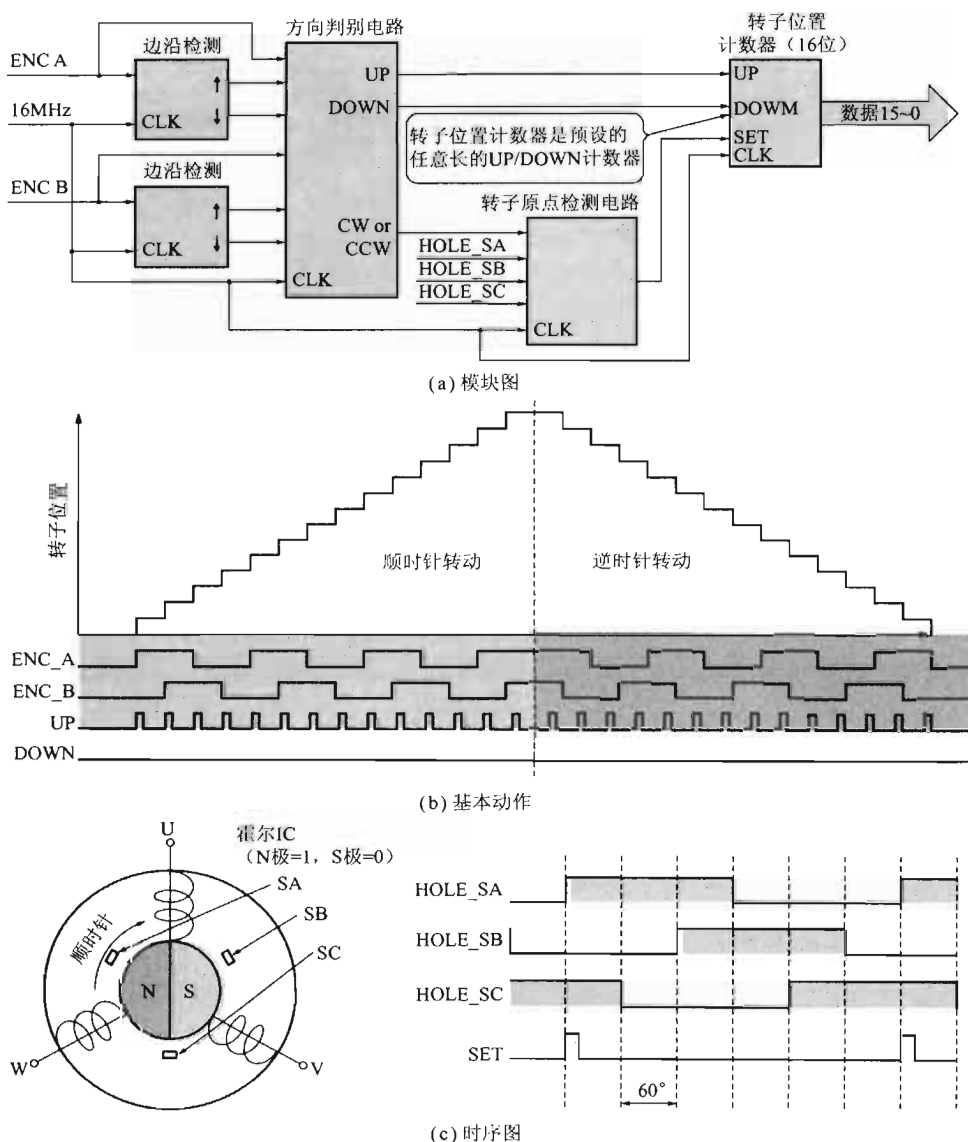


图 6.7 转子位置检测电路的构成和动作

### 1. 增量式编码器的输出信号

如图 6.7(b) 所示, 增量式编码器的输出信号是 A 相 (ENC\_A) 及 B 相 (ENC\_B) 的二相脉冲信号。输出的这个二相脉冲有  $90^\circ$  相位差。根据这二相脉冲的位置关系可以确定转动的方向。



图中所示例子是,顺时针转动的时候 A 相脉冲先行,逆时针转动的时候 B 相脉冲先行。虽然有很多的增量式编码器都是这样的关系,但是也有厂商的产品是相反设置的。另外,很多的增量式编码器是 Z 相或者 I 相作为标准脉冲输出的。这个标准脉冲是电机转动一周的时候,输出的一个脉冲。

## 2. 电机转动方向的判别电路

图 6.7(b)中所示的是电机转动方向的判别电路和转子位置计数器的基本动作。

根据 ENC\_A 和 ENC\_B 的信号来判断电机的转动方向,输出 UP 脉冲和 DOWN 脉冲。这里的脉冲是由 ENC\_A 和 ENC\_B 的上升与下降两边沿动作组成的,编码器信号可以得到 4 倍的脉冲。

## 3. 转子位置计数器

转子位置计数器如图 6.7(b)所示,是一种在 UP 脉冲中为增量计数,在 DOWN 脉冲中为减量计数的 UP/DOWN 计数器。

另外,每个电角度的编码器计数器都不同,电机位置计数器需要用任意长的计数器。

这里的电角度指的是电机驱动波形转动一周的时候电机的转动角度。2 极电机中为  $360^\circ$ ,4 极电机中为  $180^\circ$ 。

下面补充说明任意长的计数器。假设每个电角度的编码器计数器数为 2000,增量计数的时候和减量计数时的动作如下所示。

增量计数时	减量计数时
1997	3
1998	2
1999	1
0(清除)	0
1	1999(预设)
2	1998
3	1997

由此可知,增量计数和减量计数在计数器的处理中是有不同的,实际的电路稍微有点复杂。

## 4. 电机原点检测电路

图 6.7(c)中表示的是电机原点检测电路的时序图。电机的原点位置是由 HOLE\_SA 的两边沿检测出来的。在  $\text{HOLE\_SB}=0, \text{HOLE\_SC}=1$  的条件下输出电机的原点(SET)脉冲。

这种方法是连接电源后,在收到转子原点(SET)脉冲前不知道转子位置的时候,将转子位置由 HOLE\_SA、HOLE\_SB、HOLE\_SC 初始化成其中心值。如果电角度是  $360^\circ$  的话,最差的情况下有  $\pm 30^\circ$  左右的转子位置偏移,但是如果电机回到转子原点的话,就能将这个偏移消除。

## 6.4 向量控制的基础知识

在交流伺服电机的控制中,可以说向量控制是现在必须具有的功能。电机中电流流过的时候,作用在电机上的力是向量,可以分为旋转方向的力(转矩)和转动轴垂直方向的力(励磁)。在向量控制中也将转矩成分和励磁成分分开,为了使电机的效率最大化而分别进行控制。

图 6.8 表示的是基于向量控制的转矩控制放大器的整体构成。交流伺服电机的转子是永磁体,励磁是由永磁体形成的。因此,励磁和永磁体一起旋转。由于这个旋转的励磁和平行的电枢线圈磁通的励磁不论强弱,都不会改变其转动力,所以图 6.8 的励磁指令(ID)是 0。

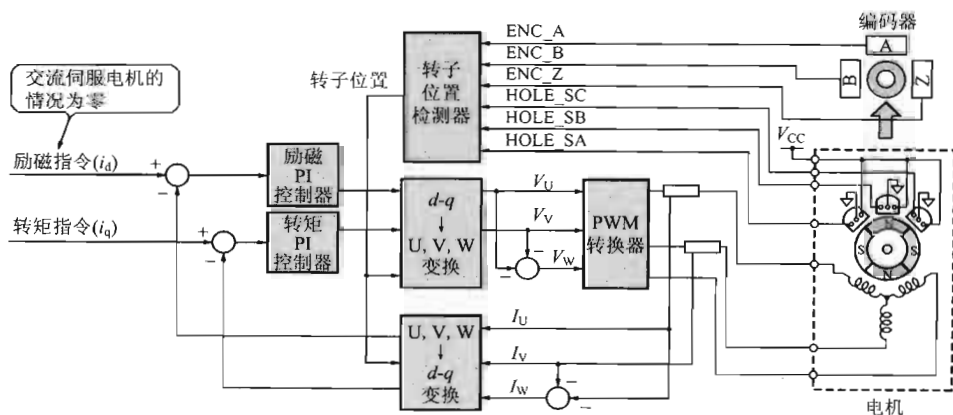


图 6.8 基于向量控制的转矩控制放大器的组成

本章在微控制器的软件安装之前,通过模拟(MATLAB)和计算(EXEL)整理得到向量控制所需要的一些演算,作为向量控制的基础知识。

### 6.4.1 模拟的参数

作为模拟对象的电机是 maxon motor 公司的无刷电机 EC32,80W,48V (118890)。

图 6.9 表示的是向量控制放大器的模拟模型。

#### 1. 控制参数

$T=0.0001$ ;	向量控制周期(s)(10kHz)
$F=100$ ;	转数 100(Hz)(6000r/min)
$K_p=0.5$ ;	比例
$K_i=0.25$ ;	积分
$G_i=0.004467$ ;	电流增益(A/count)
$G_v=0.03$ ;	电压增益(V/count)(48V/PWM 配分函数)

#### 2. 电机参数

$L_a=0.7 \times 10^{-3}$ ;	端口间的电感(H)
$R_a=5.5$ ;	端口间的电阻
$K_u=239$ ;	转数常数(r/min/V)

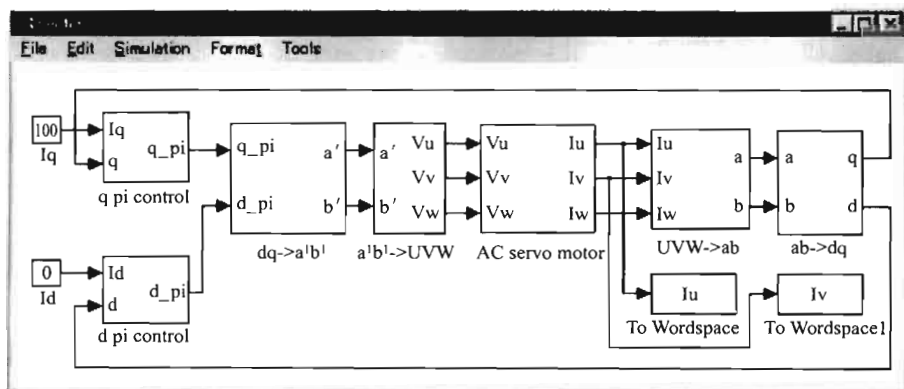


图 6.9 向量控制放大器的模拟模型

### 6.4.2 电机电流三相(U,V,W)的波形

设转矩指令值( $i_q$ )为 100,励磁指令值( $i_d$ )为 0,基于模拟形成三相的电机电流(U,V,W)波形。图 6.10 表示的是从模拟中得到的电机的三相电流图。另外,纵轴的电机电流是在微控制器内部使用的整数值(下同),横轴是时间(ms)。

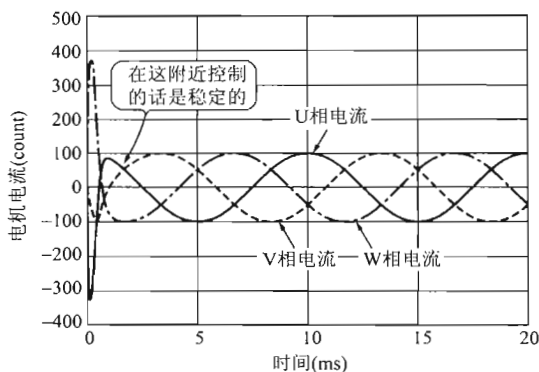


图 6.10 电机电流的三相模型(U,V,W)

从模拟的结果来看,由于转矩( $q$ )PI 控制器和励磁( $d$ )PI 控制器的积分项从 0 开始的,虽然一开始的电流比较乱,但是稳定 2ms 左右,就能够达到指令要求的电流强度。

### 6.4.3 电机电流的三相(U,V,W)→二相( $\alpha,\beta$ )转换

图 6.11 表示的是在 XY 坐标上三相(U,V,W)→二相( $\alpha,\beta$ )转换方法。用 XY 坐标表示三相正弦波(三相交流)的时候,每间隔  $120^\circ$  有个向量。U 相是 X 轴(cos)。像如图 6.11 表示的那样,将 W 相反转和 V 相合成的向量与 Y 轴一

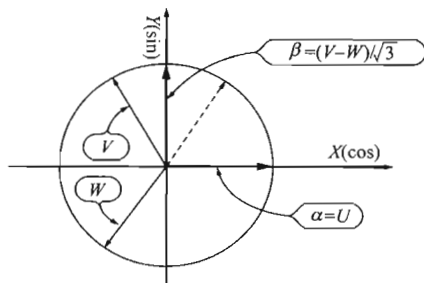


图 6.11 三相(U,V,W)到二相( $\alpha,\beta$ )变换的向量图

致。这个时候由于合成的向量是原来的 $\sqrt{3}$ 倍,所以这个三相(U,V,W)→二相( $\alpha,\beta$ )的转换式为

$$\alpha = U \text{ 相}$$

$$\beta = (V \text{ 相} - W \text{ 相}) / \sqrt{3}$$

图 6.12 中显示的是  $\alpha\beta$  转换之后的波形和  $\cos, \sin$  波形。图 6.12 是双 Y 轴的图,右侧是  $\cos/\sin$  的数值轴。参照左边的  $\alpha\beta$  轴的数值和看到的振幅进行缩放。图中,静止后的  $\cos\alpha$  和  $\sin\alpha\beta$  是完全一致的。可以证明以上的三相(U,V,W)→二相( $\alpha,\beta$ )转换式是正确的。

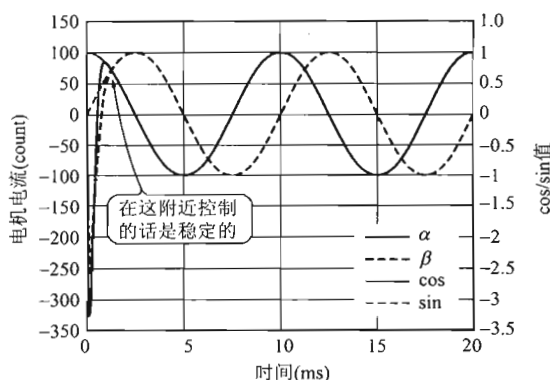


图 6.12 三相(U,V,W)→二相( $\alpha,\beta$ )变换后的波形

#### 6.4.4 电机电流的二相( $\alpha,\beta$ )到 $d-q$ 的转换

对于  $\alpha\beta$  变换后的波形来说,进行  $d-q$  变换,就可以把电机电流分离成  $d$  (励磁)成分和  $q$  (转矩)成分。 $d-q$  变换的基本式可以用以下的行列式来表示

$$\begin{bmatrix} q \\ d \end{bmatrix} = \begin{bmatrix} \cos\theta & +\sin\theta \\ \sin\theta & -\cos\theta \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \end{bmatrix} \quad (6.1)$$

一直到  $\alpha\beta$  变换为止都是静止坐标系。静止坐标系就像是以人的视角来观察电机的情况。电机转动的时候,其电枢线圈的电流是三相交流(U,V,W)或者二相交流( $\alpha,\beta$ )的正弦波。

$d-q$  变换的情况下,视角就发生改变了。就如之前说的那样,由于交流伺服电机的转子是永磁体,励磁是用永磁体产生的,因此,励磁和永磁体一起旋转。 $d-q$  变换将视角移动到了转动的永磁体中。考虑和永磁体一起转动的旋转坐标系,电枢线圈的电流振幅和永磁体的转动速度是一样的。这样的话,在静止坐标系中

看到的电枢线圈的电流在旋转坐标中看来就是直流的。取转子的永磁体的磁通方向为  $d$  (励磁) 轴, 取相位超前  $d$  轴  $90^\circ$  位置为  $q$  (转矩) 轴, 从各自的视角来看  $\alpha, \beta$  合成的向量电流, 可以分离抽出  $q$  (转矩) 成分和  $d$  (励磁) 成分。

### 1. 分离抽出 $q$ (转矩) 成分

图 6.13 表示的是二相  $(\alpha, \beta) \rightarrow q$  变换之后的波形。由式 (6.1) 可知,  $q$  (转矩) 成分的分离抽出方式为  $q = \cos\theta \times \alpha + \sin\theta \times \beta$ 。向量控制放大器的控制稳定时,  $q$  变成了只有第一项  $\cos\theta \times \alpha$  和第二项  $\sin\theta \times \beta$  之和的数据, 两个相位相差  $180^\circ$ 。因此, 第一项的  $\cos\theta \times \alpha$  和第二项  $\sin\theta \times \beta$  的和为直流, 与指定的转矩值 (指定值 = 100) 一致。

### 2. 分离抽出 $d$ (励磁) 成分

图 6.14 表示的是二相  $(\alpha, \beta) \rightarrow d$  之后的波形。由式 (6.1) 可知,  $d$  (励磁) 成分的分离抽出方式为  $d = \sin\theta \times \alpha - \cos\theta \times \beta$ 。

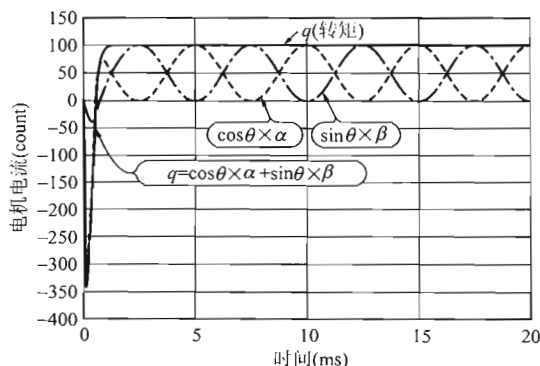


图 6.13  $q$  (转矩) 成分的分离抽出

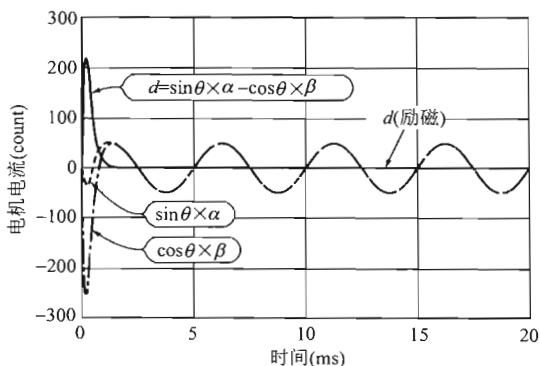


图 6.14  $d$  (励磁) 成分的分离抽出



向量控制放大器的控制稳定的时候,第一项的  $\sin\theta \times \alpha$  和第二项  $\cos\theta \times \beta$  的振幅及相位是一致的。因此第一项  $\sin\theta \times \alpha$  和第二项  $\cos\theta \times \beta$  的差分是直流的,与励磁指令(指令值=0)是一致的。

### 3. 二相( $\alpha, \beta$ ) $\rightarrow d$ - $q$ 变换的向量图

图 6.15(a)表示的是用  $XY$  坐标表示二相( $\alpha, \beta$ ) $\rightarrow q$  变换的样子。在  $\alpha\beta$  的直角坐标上,取永磁体的磁通方向为  $d$ (励磁)轴,取相位超前  $d$  轴  $90^\circ$  位置为  $q$ (转矩)轴。 $d$  轴及  $q$  轴取的是和永磁体一起旋转的旋转坐标。

永磁体旋转的话,如图 6.15(b)的  $d$ - $q$  向量轨迹所示, $d$  轴也旋转了,视角也改变了, $\alpha\beta$  的合成波形也偏移了。 $d$  轴的视角用  $\blacksquare$  来表示, $q$  轴的视角用  $\bigcirc$  来表示,分别表示其向量的轨迹。看  $d$  轴的向量轨迹的话是直流,其值为 0。另外, $q$  轴的向量轨迹也是直流,其值为 1(最大值)。

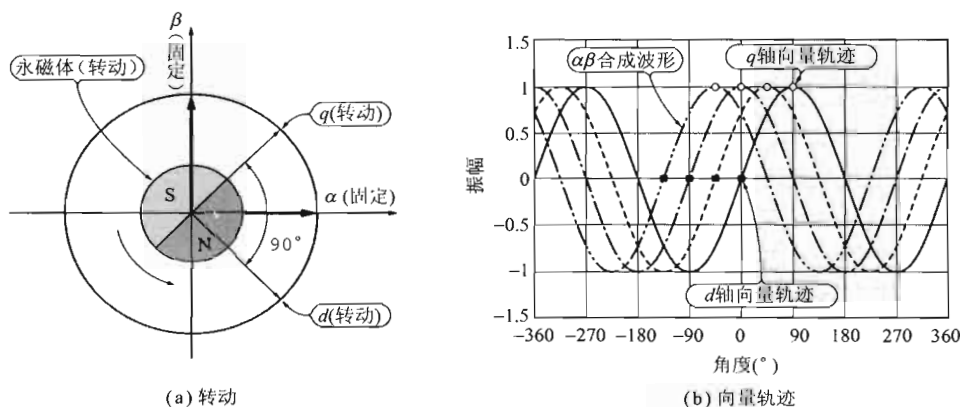
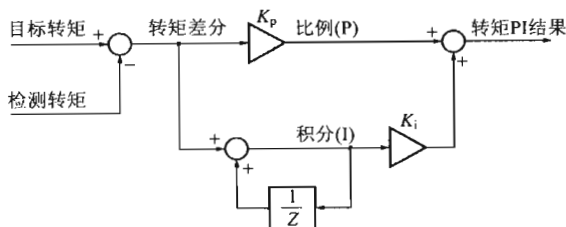
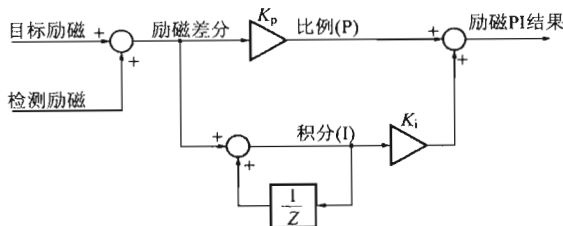
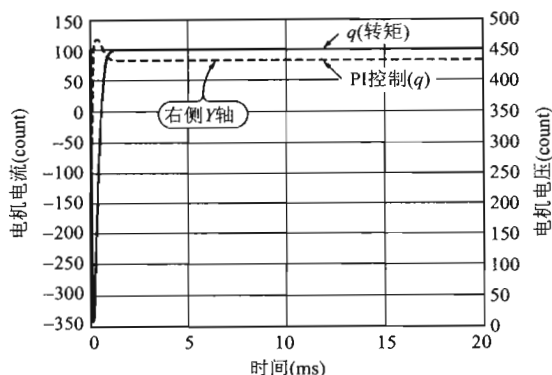


图 6.15 二相( $\alpha, \beta$ ) $\rightarrow d$ - $q$  变换的向量图

## 6.4.5 PI(比例积分)控制器

图 6.16 为  $q$ (转矩)的 PI 控制器方框图,图 6.17 为  $d$ (励磁)的 PI 控制器方框图。两者的 PI 控制器完全不同。 $d$ (励磁)的 PI 控制器中,与目标相加而不是相减。设比例常数  $K_P=0.5$ ,积分常数  $K_I=0.25$ ,分别计算其 PI 控制器的输出。图 6.18 为  $q$ (转矩)PI 控制器的计算结果。

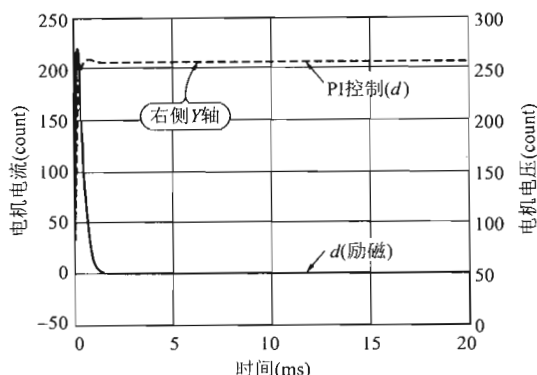
图 6.18 是双 Y 轴的图,输入是左边 Y 轴的  $q$ (转矩),输出是右边 Y 轴的 PI 控制( $q$ )。虽然 PI 控制器输入前的单位是观测的电机电流,但是 PI 控制器输出后变为了作为操作量的电机电压。向量控制放大器的控制稳定时,PI 控制


 图 6.16  $q$ (转矩)的 PI 控制器

 图 6.17  $d$ (励磁)的 PI 控制器

 图 6.18  $q$ (转矩)PI 控制器的特性

( $q$ )是 430 左右,转变成电压的话是  $430 \times \text{电压增益 } 0.03(\text{V/count}) = 12.9(\text{V})$ 。

图 6.19 是  $d$ (励磁)PI 控制器的计算结果。图 6.19 也是双 Y 轴的图,输入是左边 Y 轴的  $d$ (励磁),输出是右边 Y 轴的 PI 控制( $d$ )。同样的,PI 控制器输入前是作为观测量的电机电流,PI 控制器输出后变为了作为操作量的电机电压。向量控制放大器的控制稳定时,PI 控制( $d$ )是 255 左右,转变成电压的话是  $255 \times \text{电压增益 } 0.03(\text{V/count}) = 7.65(\text{V})$ 。

$d$ (励磁)PI 控制器的输出不为 0 的理由是,电机控制基本上是电感负荷的

图 6.19  $d$ (励磁)PI 控制器的特性

控制, 电机的电压相位超前电流相位。为了消除电机电流的相位延迟, 电机电压相位需要向前, 结果是,  $d$ (励磁)PI 控制器的输出不为 0。

#### 6.4.6 PI 控制器 $d-q \rightarrow \alpha'-\beta'$ 变换

对  $d$ (转矩)与  $d$ (励磁)的 PI 计算结果进行  $d-q$  变换的话, 可以得到  $\alpha'\beta'$ (二相)。 $d-q$  变换的基本公式可以用行列式表示:

$$\begin{bmatrix} \alpha' \\ \beta' \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & +\cos\theta \end{bmatrix} \begin{bmatrix} q \\ d \end{bmatrix} \quad (6.2)$$

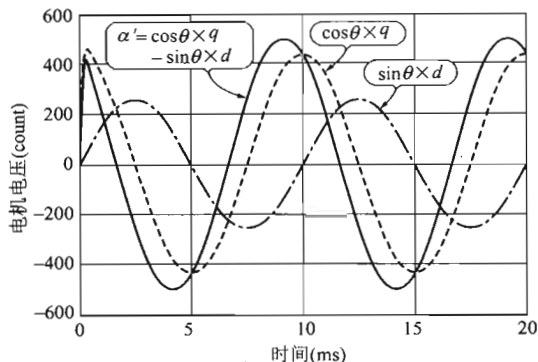
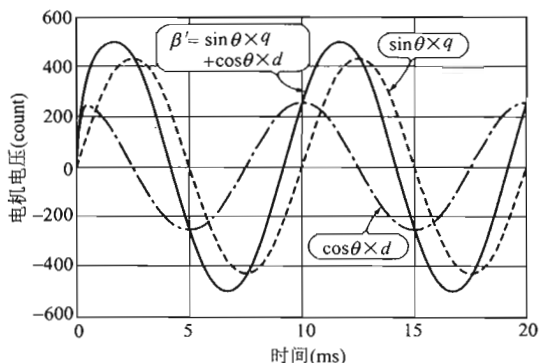
与式(6.1)的  $d-q$  变换的输入输出正好相反, 行列式里面的符号也变化了。基于这个符号, 可以从转动坐标系转变回静止坐标系。

##### 1. PI 控制器 $d-q \rightarrow \alpha'$ 变换

图 6.20 表示的是 PI 控制器  $d-q \rightarrow \alpha'$  变换后的波形图。PI 控制器的  $d-q \rightarrow \alpha'$  变换式可以从(6.2)中得到,  $\alpha' = \cos\theta \times q - \sin\theta \times d$ 。向量控制放大器的控制稳定时,  $d$  与  $q$  就是直流了,  $\cos\theta \times q$  是余弦波形,  $\sin\theta \times d$  是正弦波形,  $\alpha'$  是这两部分差分的合成波形。

##### 2. PI 控制器 $d-q \rightarrow \beta'$ 变换

图 6.21 表示的是 PI 控制器  $d-q \rightarrow \beta'$  变换后的波形图。PI 控制器的  $d-q \rightarrow \beta'$  变换式可以从(6.2)中得到。  $\beta' = \sin\theta \times q + \cos\theta \times d$ 。向量控制放大器的控制稳定时,  $d$  与  $q$  就是直流了,  $\sin\theta \times q$  是正弦波形,  $\cos\theta \times d$  是余弦波形,  $\beta'$  是这两部分的合成波形。

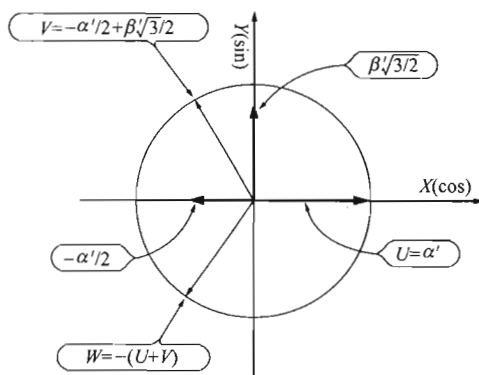
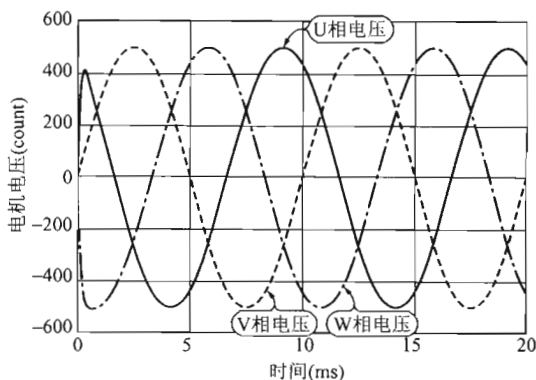

 图 6.20  $d-q \rightarrow \alpha'$  变换后的波形

 图 6.21  $d-q \rightarrow \beta'$  变换后的波形

#### 6.4.7 二相( $\alpha'$ , $\beta'$ ) $\rightarrow$ 三相(U, V, W)变换

这一小节中,进行的是和 6.4.3 节相反的二相( $\alpha'$ ,  $\beta'$ ) $\rightarrow$ 三相(U, V, W)的变换。这种变换中需要求得的是加载在电机上的电压。

图 6.22 表示的是在 XY 坐标系下的二相( $\alpha'$ ,  $\beta'$ ) $\rightarrow$ 三相(U, V, W)的变换。设 U 相在  $\cos$  轴上, U 相 =  $\alpha'$ ; V 相有点复杂,将 V 相放置在  $120^\circ$  的位置,  $V = -\alpha'/2 + \beta' \times \sqrt{3}/2$ 。W 相的公式中,电机电压和电流都同样的三项平衡,因此  $W = -(U + V)$ 。

图 6.23 表示的是电机电压的三相(U, V, W)波形。作为操作量的电机电压波形对于电机电流来说,在模拟的时候波形会超前  $30^\circ$ 。另外,其电压为  $\pm 500(\text{count}) \times 0.03(\text{V/count}) = \pm 15\text{V}$ 。

图 6.22 二相 $(\alpha', \beta')$ →三相 $(U, V, W)$ 变换图 6.23 电压的三相 $(U, V, W)$ 波形

## 6.5 基于微控制器的向量控制实验

下面在微控制器中进行基于向量控制的电流环的实验。对电流环进行如下必要的处理。

- (1) 计算出转子位置,从 LUT(Look Up Table)中读取  $\sin$  值和  $\cos$  值。
- (2) 读取 U 相及 V 相的电机电流,进行移动平均处理、偏差补偿、增益修正。
- (3) 针对各相的电机电流,进行  $\alpha\beta$  变换,之后进行  $d-q$  变换。
- (4) 进行  $q$ (转矩)控制计算和  $d$ (励磁)PI 控制计算。
- (5) 对转矩和励磁的 PI 计算结果进行  $d-q$  变换,然后进行  $\alpha\beta$  变换。
- (6) 计算各相 $(U, V, W)$ 的 PWM 设定值。

### 6.5.1 计算转子的位置,从 LUT(Look Up Table)中读取 sin 值和 cos 值

程序如程序清单 6.1 所示。

程序清单 6.1 从转子位置计数器的读取开始到 cos 值和 sin 值的读取为止

```

; *****
; ***** 转子位置检测 &cos 值(R6)&sin 值(E6)的读取 *****
; 将转子位置均匀分割成 1024 段
    MOV.W    @REG_LUT,R2      ; 读取转子位置计数器
    MOV.W    @_igRotMul,R1     ; 转子位置乘积值(4096 = 1 倍)
    MULXU.W  R1,ER2            ;
    SHLL.L   #2,ER2           ; × 4
    SHLL.L   #2,ER2           ; × 16
    MOV.W    E2,R3            ; ÷ 65536
    MOV.W    R3,@_igRotAgl     ; 转子位置更新
; 加上重叠角偏移补偿
    MOV.W    @_igSinPhase,R0   ; 读取相位修正数据
    ADD.W    R0,R3            ; 进行相位修正
    AND.W    #H'03FF,R3       ; 由于 LUT 是 1024 位的,隐藏 10 位以外
    MOV.W    R3,R2
; cos,sin LUT 对比
    SHLL.W   R2               ; 由于 LUT 的数据是 2 字节的
    EXTU.L   ER2              ; 扩展 32 位
    MOV.L    #COS_LUT,ER0     ; 得到指向 cos 值的基址
    MOV.L    #SIN_LUT,ER1     ; 得到指向 sin 值的基址
    ADD.L    ER2,ER0          ; 加上转子位置分量
    ADD.L    ER2,ER1          ; 加上转子位置分量
    MOV.W    @ER0,R6          ; 得到 cos 值(-32768~32767)
    MOV.W    @ER1,E6          ; 得到 sin 值(-32768~32767)
; igRotMul 的单位是
; 65535(FFFF) = 15.9997 倍 = 1024(LUT)/64(电角度)
; 65534(FFFE) = 15.9995 倍
;
; 4097(1001) = 1.0002 倍
; 4096(1000) = 1.0000 倍 = 1024(LUT)/1024(电角度)
; 4095(0FFF) = 0.9997 倍
;
; 64(0040) = 1/64 倍 = 1024(LUT)/65536(电角度)

```

首先读取 CPLD 的转子位置计数器。转子位置计数器根据编码器的分辨能力,可以作为任意长的 UP/DOWN 计数器。编码器分辨能力变化的时候,需要准备 LUT,很麻烦,所以将 LUT 固定在 1024。因此,转子位置用软件分成 1024 等分来处理。

然后,在转子位置中加上重叠角补偿。在这里用 $\theta$ 表示转子位置, $\cos\theta$ 和 $\sin\theta$ 参照 LUT(Look Up Table),求得其值。 $\cos\theta$ 和 $\sin\theta$ 参照值的范围为 $-32768\sim 32767$ 。

-32768	-1.0
-32767	-0.999969
-32766	-0.999939
⋮	⋮
-1	-0.0000305
0	0.0
1	0.0000305
2	0.0000610
⋮	⋮
32766	0.999939
32767	0.999969

### 6.5.2 U相和V相的电机电流值的读取、偏差修正、增益修正

程序如程序清单 6.2 所示。

程序清单 6.2 U相电机电流值的读取

; *****		MOV. B	@ER0, R2L	; 5 个
; *** U相电流的计算(保存在 R5 中) ***		INC. L	# 2, ER0	
; *****		ADD. W	R2, R1	
MOV. L	#_cgDutyIU, ER0	MOV. B	# ER0, R2L	; 6 个
流地址		INC. L	# 2, ER0	
SUB. W	R1, R1	ADD. W	R2, R1	
SUB. W	R2, R2	MOV. B	@ER0, R2L	; 7 个
; 求 13 个的电流的总值		INC. L	# 2, ER0	
MOV. B	@ER0, R1L	ADD. W	R2, R1	
INC. L	# 2, ER0	MOV. B	@ER0, R2L	; 8 个
MOV. B	@ER0, R2L	INC. L	# 2, ER0	
INC. L	# 2, ER0	ADD. W	R2, R1	
ADD. W	R2, R1	MOV. B	@ER0, R2L	; 9 个
MOV. B	@ER0, R2L	INC. L	# 2, ER0	
INC. L	# 2, ER0	ADD. W	R2, R1	
ADD. W	R2, R1	MOV. B	@ER0, R2L	; 10 个
MOV. B	@ER0, R2L	INC. L	# 2, ER0	
INC. L	# 2, ER0	ADD. W	R2, R1	
ADD. W	R2, R1	MOV. B	@ER0, R2L	; 11 个

续程序清单 6.2

INC. L    # 2,ERO		; 增益修正
ADD. W    R2,R1		MOV. W    E1,R1
MOV. B    @ERO,R2L	; 12 个	MOV. W    @_igGainIU,R0
INC. L    # 2,ERO		
ADD. W    R2,R1		
MOV. B    @ERO,R2L	; 13 个	
INC. L    # 2,ERO		
ADD. W    R2,R1		
; 和 period/2 比较		
MOV. W    @_igPer2IU,E1	; period/2	
SUB. W    R1,E1	; period/2-duty	
		MOV. W    E1,@_igCurU
		MOV. W    E1,R5

U 相电机电流时,基于 DTC 的最新的 13 次的数据在 MCU 的 RAM 中保存下来。

首先求得其总电流。其次,计算与周期/2(period/2)数据的差分。周期/2(period/2)数据,在和基本处理路径不同的线程中定期地(每 100ms)更新(修正补偿)。最后进行增益补偿,作为 U 相电流。

增益修正值(igGainIU)是  $16384=1$  倍的比例。最大倍数为 2 倍(32767)。

例:设电机电流 100,igGainIU=16384,则

```

MOV. W      @_igGainIU,R0      ; 1 倍 = 16384
MULXS. W    R0,ER1             ; 100 × 16384 = 1638400
SHAL. L     # 2,ER1            ; 1638400 × 4 = 6553600
MOV. W      E1,@_igCurU       ; 6553600 ÷ 65536 = 100

```

V 相电机电流的处理路径如程序清单 6.3 所示。处理内容和 U 相电机电流相同。

程序清单 6.3 V 相电机电流值的读取

; *****	INC. L    # 2,ERO	
; *** V 相电流的计算(保存在 E5 中)***	ADD. W    R2,R1	
; *****	MOV. B    @ERO,R2L	; 3 个
MOV. L    #_cgDutyIV,ERO	INC. L    # 2,ERO	
流地址	ADD. W    R2,R1	
SUB. W    R1,R1	MOV. B    @ERO,R2L	; 4 个
SUB. W    R2,R2	INC. L    # 2,ERO	
; 求 13 个的电流的总值	ADD. W    R2,R1	
MOV. B    @ERO,R1L	MOV. B    @ERO,R2L	; 5 个
INC. L    # 2,ERO	INC. L    # 2,ERO	
MOV. B    @ERO,R2L	ADD. W    R2,R1	



续程序清单 6.3

MOV. B    #ER0,R2L                    ; 6 个	INC. L    #2,ERO
INC. L    #2,ERO	ADD. W    R2,R1
ADD. W    R2,R1	MOV. B    @ER0,R2L                    ; 13 个
MOV. B    @ER0,R2L                    ; 7 个	INC. L    #2,ERO
INC. L    #2,ERO	ADD. W    R2,R1
ADD. W    R2,R1	; 和 period/2 比较
MOV. B    @ER0,R2L                    ; 8 个	MOV. W    @_igPer2IV,E1    ; period/2
INC. L    #2,ERO	SUB. W    R1,E1            ; period/2-duty
ADD. W    R2,R1	; 增益修正
MOV. B    @ER0,R2L                    ; 9 个	MOV. W    E1,R1
INC. L    #2,ERO	MOV. W    @_igGainIV,R0
ADD. W    R2,R1	; 1 倍 = 16384
MOV. B    @ER0,R2L                    ; 10 个	MULXS. W  R0,ER1           ; * _igGainIV
INC. L    #2,ERO	SHAL. L   #2,ER1           ; * _igGainIV * 4
ADD. W    R2,R1	
MOV. B    @ER0,R2L                    ; 11 个	MOV. W    E1,@_igCurU
INC. L    #2,ERO	; V 相检测出的
ADD. W    R2,R1	电流
MOV. B    @ER0,R2L                    ; 12 个	MOV. W    E1,R5

### 6.5.3 对电机电流进行 $\alpha$ - $\beta$ 变换,再进行 $d$ - $q$ 变换

程序如程序清单 6.4 所示。

程序清单 6.4 三相(U,V,W)→二相( $\alpha$ , $\beta$ )变换→ $d$ - $q$  变换

; *****	
; ***** 三相(U,V,W)→二相( $\alpha$ , $\beta$ )变换 *****	
; *****	
; W 相电流    R5 = U, E1 = V	
MOV. W    E1,E5	
ADD. W    R5,E1	; U + V
NEG. W    E1	; W = - (U;V)
; $\alpha$ - $\beta$ 变换 $\alpha$ = U $\beta$ = (V - W)/SQRT(3)	
MOV. W    E5,R0	; V
SUB. W    E1,R0	; V - W
MOV. W    #18918,R1	; 1/SQRT(3) = 0.57735
MULXS. W  R1,ERO	; (V - W)/SQRT(3)
SHAL. L    ERO	
MOV. W    E0,E5	; $\beta$ = E5
; *****	
; ***** 二相 $\alpha$ $\beta$ 变换→ $d$ - $q$ 变换 *****	
; *****	
; $d$ - $q$ 变换    R6 = $\cos\theta$ , R5 = $\alpha$ , E6 = $\sin\theta$ , E5 = $\beta$	
; q = ( $\cos\theta * \alpha + \sin\theta * \beta$ )	
MOV. W    R6,R0	; $\cos\theta$

```

        MULXS.W   R5,ER0           ;  $\cos\theta * \alpha$ 
        SHAL.L    ER0              ;  $\cos\theta * \alpha = E0$ 
        MOV.W     E6,R1            ;  $\sin\theta$ 
        MULXS.W   E5,ER1           ;  $\sin\theta * \beta$ 
        SHAL.L    ER1              ;  $\sin\theta * \beta = E1$ 
        ADD.W     E1,E0             ;  $q = (\cos\theta * \alpha + \sin\theta * \beta)$ 
        MOV.W     E0,R4             ;  $R4 = q$ 
        MOV.W     R4,@_igTerIU      ; 保存  $q$ (转矩)
; d = ( $\sin\theta * \alpha - \cos\theta * \beta$ )
        MOV.W     R6,R0            ;  $\sin\theta$ 
        MULXS.W   R5,ER0           ;  $\sin\theta * \alpha$ 
        SHAL.L    ER0              ;  $\sin\theta * \alpha = E0$ 
        MOV.W     R6,R1            ;  $\cos\theta$ 
        MULXS.W   E5,ER1           ;  $\cos\theta * \beta$ 
        SHAL.L    ER1              ;  $\cos\theta * \beta = E1$ 
        SUB.W     E1,E0             ;  $q = (\cos\theta * \alpha + \sin\theta * \beta)$ 
        MOV.W     E0,R4             ;  $E4 = d$ 
        MOV.W     E4,@_igTerIV      ; 保存  $d$ (励磁)
        BRA       Q_PI

```

首先计算出 W 相电流,进行三相(U,V,W)→二相( $\alpha,\beta$ )变换。程序中有以下指令:

```
MOV.W      #18918, R1           ;  $1 \div \text{SQRT}(3) = 0.57735$ 
```

$18918/32768 = 0.57733(1/\sqrt{3})$ 。

三相(U,V,W)→二相( $\alpha,\beta$ )变换后,再进行二相( $\alpha,\beta$ )→ $d-q$ 变换。

#### 6.5.4 PI(比例积分)控制器

图 6.24 中表示的是在微控制器中的  $q$ (转矩)PI 控制器的方框图。与图 6.16 的  $q$ (转矩)PI 控制器相比有很大的变动,理由是为了配合微控制器的 CPU。

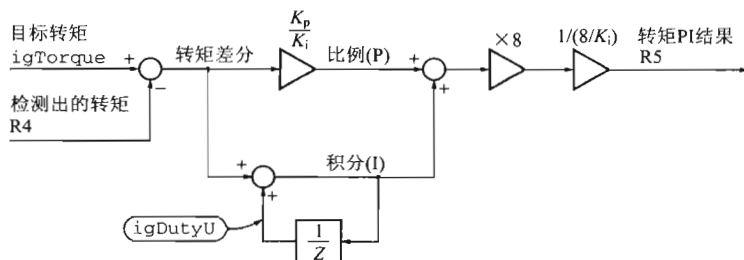


图 6.24  $q$ (转矩)成分的 PI(比例积分)控制器

为了避免积分项的保存值的溢出,使用 32 位的变量(ig dutyu)。这样一来的话,积分项的计算就需要  $32\text{bit} \times 32\text{bit}$  (或者是  $32\text{bit} \times 16\text{bit}$ ) 的乘法运算, H8S/2000CPU 中没有 32bit 的乘法指令,但却有  $32\text{bit} \times 16\text{bit}$  的除法命令(DIVXS.W),所以考虑了一个运用这个除法指令来进行 PI 控制的计算方法。H8S/2000CPU 的除法命令需要 21 个状态的执行时间,因此比起用  $32\text{bit} \times 32\text{bit}$  的乘法运算方法更能够高速地处理。

程序如程序清单 6.5 所示。

程序清单 6.5  $q$ (转矩)的 PI 控制计算

```

; *****
; *****  q(转矩)PI 计算 *****
; *****
; R4 = 检测出的转矩电流
; 与目标电流值比较
Q_PI:  MOV.W    @_igCurKpi,R0    ; 比例/积分增益( $K_p/K_i$ )
      MOV.W    @_igCurK8i,E0    ; 积分增益( $8/K_i$ )
      MOV.W    @_igTorque,R1    ; 目标电流值
      SUB.W     R4,R1           ; 电流差分 = 目标电流 - 检测出的电流
      MOV.W     R1,R3
      MULXS.W   R0,ER3          ; 电流差分 * ( $K_p/K_i$ )

; q(转矩)积分值的变更
      MOV.L     @_lgDutyU,ER2    ; 读取积分值
      EXTS.L    ER1
      ADD.L     ER1,ER2          ; 将电流差分积分
      BVS      PWMOVERU         ; 溢出?
SVDUTYU MOV.L     ER2,@_lgDutyU  ; 积分值更新
      ADD.L     ER3,ER2          ; 积分 + 比例
      SHAL.L    #2,ER2           ; (积分 + 比例) * 4
      SHAL.L    ER2             ; (积分 + 比例) * 8
      BVS      DIFOVERU         ; 溢出?
SVDIFFU  DIVXS.W  E0,ER2         ; (积分 + 比例) / ( $8/K_i$ )
      MOV.W     R2,R5           ; R5 = q(转矩)PI 计算结果
      BRA      D_PI             ; d(励磁)PI 计算

; igDutyU + 电流差分(积分)溢出的时候的处理
PWMOVERU:
      BCS      PWMMINU
      MOV.L     #H'7FFFFFFF,ER2
      BRA      SVDUTYU
PWMMINU MOV.L     #H'80000001,ER2
      BRA      SVDUTYU

; (积分 + 比例) * 8 溢出的时候的处理
DIFOVERU:

```

	BCS	DIFMINU
	MOV. L	# H'7FFFFFFF,ER2
	BRA	SVDIFFU
DIFMINU	MOV. L	# H'80000001,ER2
	BRA	SVDIFFU

首先读取 PI 计算的相关系数。一个是比例/积分增益( $K_p/K_i$ ),这是电流环增益  $K_i$ ,  $K_p$  在设定的时间预先计算出来保存在变量(igcurkpi)中的数值;另外一个积分增益( $8/K_i$ ),这也是预先计算出来保存在变量(igcurk8i)中的数值。这个 PI 控制模块的注意点是,禁止电流环积分增益  $K_i=0$ ,另外  $K_i=8$  是上限。

令转矩电流差分为  $X(n)$ ,PI 控制器输出为  $Y$ ,得到图 6.16 的传递函数为

$$\frac{Y(n)}{X(n)} = K_p + \frac{K_i}{1-z^{-1}}$$

另外,图 6.24 的传递函数为

$$\frac{Y(n)}{X(n)} = \left( K_p/K_i + \frac{1}{1-z^{-1}} \right) \times 8/(8/K_i)$$

$$\frac{Y(n)}{X(n)} = \left( K_p/K_i + \frac{1}{1-z^{-1}} \right) \times 8/(8/K_i)$$

$$\frac{Y(n)}{X(n)} = (K_p/K_i) \times K_i + \frac{1}{1-z^{-1}} \times K_i$$

$$\frac{Y(n)}{X(n)} = K_p + \frac{K_i}{1-z^{-1}}$$

图 6.24 和图 6.16 基本上是同样的结果。

$d$ (励磁)PI 控制器如程序清单 6.6 所示。只有目标值和实际测量值的比较(SUB 指令和 ADD 指令)有所不同,其他部分完全相同。

程序清单 6.6  $d$ (励磁)的 PI 控制计算

```

; *****
; ***** d(励磁)的计算 *****
; *****
; R4 = 检测出的转矩电流
D_PI:
; 和目标电流值的比较
MOV. W    @_igKaiji,R1    ; 目标励磁电流值
ADD. W    E4,R1           ; 电流差分 = 目标电流 + 检测出的电流
MOV. W    R1,R3
MULXS. W  R0,ER3          ; 电流差分 × (Kp/Ki)

```

续程序清单 6.6

```

; d(励磁)积分值的变更
MOV.L   @_lgDutyV,ER2
EXTS.L  ER1
ADD.L   ER1,ER2           ; 将电流差分积分
BVS     PWMOVERV         ; 溢出?
SVDUTYU MOV.L   ER2,@_lgDutyV ; 积分值更新
ADD.L   ER3,ER2           ; 积分 + 比例
SHAL.L  #2,ER2           ; (积分 + 比例) × 4
SHAL.L  ER2               ; (积分 + 比例) × 8
BVS     DIFOVERV         ; 溢出?
SVDIFFV DIVXS.W E0,ER2     ; (积分 + 比例)/(8/Ki)
MOV.W   R2,E5             ; E5 = d(励磁)PI 计算结果
BRA     DQTRANS

; igDutyV + 电流差分(积分)溢出的时候的处理
PWMOVERV:
BCS     PWMMINV
MOV.L   #H'7FFFFFFF,ER2
BRA     SVDUTYV
PWMMINV MOV.L   #H'80000001,ER2
BRA     SVDUTYV

; igDutyV + 电流差分(比例)溢出的时候的处理
DIFOVERV:
BCS     DIFMINV
MOV.L   #H'7FFFFFFF,ER2
BRA     SVDIFFV
DIFMINV MOV.L   #H'80000001,ER2
BRA     SVDIFFV

```

### 6.5.5 PI 控制器 $d-q \rightarrow$ 二相( $\alpha', \beta'$ ) $\rightarrow$ 三相(U,V,W)变换

程序如程序清单 6.7 所示。 $d$ (励磁)和  $p$ (转矩)对于各自的 PI 控制器输出,都进行  $d-q$  变换(式(6.2))的话,便可得到二相( $\alpha', \beta'$ )。程序中继续进行了二相( $\alpha', \beta'$ ) $\rightarrow$ 三相(U,V,W)的变换。但是如图 6.22 的向量图所表示的那样,由于  $\alpha'$  和 U 相配置在同一个向量里面,所以本程序中所进行的坐标变换只有 V 相。

### 6.5.6 三相(U,V,W) $\rightarrow$ PWM 变换

程序如程序清单 6.8~程序清单 6.10 所示。将三相(U,V,W)电压分别变换至 PWM 占空比的设定值。

程序清单 6.7  $d-q \rightarrow$  二相  $(\alpha', \beta')$   $\rightarrow$  三相  $(U, V, W)$  变换

```

; *****
; *****  d-q $\rightarrow$ 二相 $(\alpha, \beta)$ 变换 *****
; *****
; d-q 变换  R6 = cos $\theta$ , R5 = q, E6 = sin $\theta$ , E5 = d
DQTRANS;
;  $\alpha = (\cos\theta * q - \sin\theta * d)$ 
    MOV.W    R6, R0        ; cos $\theta$ 
    MULXS.W  R5, ER0       ; cos $\theta * q$ 
    SHAL.L   ER0           ; cos $\theta * q = E0$ 
    MOV.W    E6, R1        ; sin $\theta$ 
    MULXS.W  E5, ER1       ; sin $\theta * d$ 
    SHAL.L   ER1           ; sin $\theta * d = E1$ 
    SUB.W    E1, E0        ;  $\alpha = (\cos\theta * q - \sin\theta * d)$ 
    MOV.W    E0, R4        ; R4 =  $\alpha$ 

;  $\beta = (\sin\theta * q + \cos\theta * d)$ 
    MOV.W    E6, R0        ; sin $\theta$ 
    MULXS.W  R5, ER0       ; sin $\theta * q$ 
    SHAL.L   ER0           ; sin $\theta * q = E0$ 
    MOV.W    R6, R1        ; cos $\theta$ 
    MULXS.W  E5, ER1       ; cos $\theta * d$ 
    SHAL.L   ER1           ; cos $\theta * d = E1$ 
    ADD.W    E1, E0        ;  $\beta = (\sin\theta * q + \cos\theta * d)$ 
    MOV.W    E0, R4        ; E4 =  $\beta$ 

; *****
; *****  二相 $(\alpha, \beta) \rightarrow$ 三相变换 $(U, V, W)$  *****
; *****
; 二相 $(\alpha, \beta) \rightarrow$ 三相变换 $(U, V, W)$   R4 =  $\alpha$ , E4 =  $\beta$ 
;  $V = -\alpha/2 + \beta * \text{SQRT}(3)/2$ 
    MOV.W    #28378, R0
    MULXS.W  E4, ER0       ;  $\beta * \text{SQRT}(3)/2$ 
    SHAL.L   ER0           ;  $E0 = \beta * \text{SQRT}(3)/2$ 
    MOV.W    R4, R1        ;  $\alpha$ 
    NEG.W    R1            ;  $-\alpha$ 
    SHAR.W   R1            ;  $-\alpha/2$ 
    ADD.W    E0, R1        ;  $V = -\alpha/2 + \beta * \text{SQRT}(3)/2$ 
    MOV.W    R1, E4        ; E4 = V

```

程序清单 6.8 U 相 PWM 变换

```

; *****
; *****  U 相 PWM 变换 *****
; *****
; R4 =  $\alpha$ , PWMDUT = 799, PWMPER = 1599
AXIS_U  MOV.W  R4,R2                ; U =  $\alpha$ 
        ADD.W  # PWMDUT + PWMDUT,R2 ; PWM 变换(分辨能力 2 倍)
        CMP.W  # 128,R2             ; PWM duty  $\times 2 - 128$ 
        BLE    UZERO                ; if (duty < 4 %)
        CMP.W  # PWMPER + PWMPER - 128,R2 ; PWM duty  $\times 2 - (PWMPER \times 2 - 128)$ 
        BPL    UFULL                ; if (duty > 96 %)
; 保存 PWM duty
SVPWMU  MOV.W  R2,R6                ; 使用 W 相 PWM 变换
        SHAR.W R2                    ; PWM duty  $\times 2/2$ 
        BCS    PWMR5U
        MOV.W  R2,@_igTpuDutyU      ; 第一个 PWM
        MOV.W  R2,@_igTpuDutyU + 2 ; 第二个 PWM
        BRA    AXIS_V
PWMR5U  MOV.W  R2,@_igTpuDutyU      ; 第一个 PWM
        INC.W  # 1,R2
        MOV.W  R2,@_igTpuDutyU + 2 ; 第二个 PWM
        BRA    AXIS_V

; PWM DUTY 超过 96 % 的时候的处理
UFULL:  MOV.W  # PWMPER + PWMPER - 128,R2
        BRA    SVPWMU

; PWM DUTY 不超过 4 % 的时候的处理
UZERO:  MOV.W  # 128,R2
        BRA    SVPWMU

```

程序清单 6.9 V 相 PWM 变换

```

; *****
; *****  V 相 PWM 变换 *****
; *****
; E4 = V, PWMDUT = 799, PWMPER = 1599
AXIS_V:
        MOV.W  E4,R2                ; V
        ADD.W  # PWMDUT + PWMDUT,R2 ; PWM 变换(分辨能力 2 倍)
        CMP.W  # 128,R2             ; PWM duty  $\times 2 - 128$ 
        BLE    VZERO                ; if (duty < 4 %)
        CMP.W  # PWMPER + PWMPER - 128,R2 ; PWM duty  $\times 2 - (PWMPER \times 2 - 128)$ 
        BPL    VFULL                ; if (duty > 96 %)

```

; 保存 PWM duty		
SVPWMV	MOV. W R2,E6	; 使用 W 相 PWM 变换
	SHAR. W R2	; PWM duty $\times 2/2$
	BCS PWMR5V	
	MOV. W R2,@_igTpuDutyV	; 第一个 PWM
	MOV. W R2,@_igTpuDutyV + 2	; 第二个 PWM
	BRA AXIS_W	
PWMR5V	MOV. W R2,@_igTpuDutyV	; 第一个 PWM
	INC. W #1,R2	
	MOV. W R2,@_igTpuDutyV + 2	; 第二个 PWM
	BRA AXIS_W	
; PWM DUTY 超过 96 % 的时候的处理		; PWM DUTY 不超过 4 % 的时候的处理
VFULL;		VZERO;
	MOV. W #PWMPER + PWMPER - 128,R2	MOV. W #128,R2
	BRA SVPWMV	BRA SVPWMV

程序清单 6.10 W 相 PWM 变换

```

; *****
; ***** W 相 PWM 变换 *****
; *****
; R6 = U 相 PWM  $\times 2$ , E6 = V 相 PWM  $\times 2$ , PWMDUT = 799, PWMPER = 1599
AXIS_  ADD. W E6,R6          ; PWM U  $\times 2$  + PWM V  $\times 2$ 
      MOV. W R6,R2
      MOV. W #PWMPER + PWMPER + PWMPER - 3,E2
      SUB. W R2,E2          ; (PWMPER + PWMPER + PWMPER - 3)
                          ; - (PWM U  $\times 2$  + PWM V  $\times 2$ )
      MOV. W E2,R2
      CMP. W #128,R2        ; PWM duty  $\times 2$  - 128
      BLE WZERO             ; if (duty < 4 %)
      CMP. W #PWMPER + PWMPER = 128,R2 ; PWM duty  $\times 2$  - (PWMPER  $\times 2$  - 128)
      BPL WFULL            ; if (duty > 96 %)
; 保存 PWM duty
SVPWMV SHAR. W R2          ; PWM duty  $\times 2/2$ 
      BCS PWMR5W
      MOV. W R2,@_igTpuDutyW ; 第一个 PWM
      MOV. W R2,@_igTpuDutyW + 2 ; 第二个 PWM
      BRA RGSEL
PWMR5W MOV. W R2,@_igTpuDutyW ; 第一个 PWM
      INC. W #1,R2
      MOV. W R2,@_igTpuDutyW + 2 ; 第二个 PWM
      BRA RGSEL

```



```
; PWM DUTY 超过 96 % 的时候的处理
WFULL:
    MOV. W    # PWMPER + PWMPER - 128, R2
    BRA      SVPWMW
; PWM DUTY 不超过 4 % 的时候的处理
WZERO:
    MOV. W    # 128, R2
    BRA      SVPWMW
```

将三相(U, V, W)电压转换成 PWM 占空比的设定值的时候,虽然将电压 0 考虑成 PWM 占空比 50%,将 PWM 周期的一半的值加在三相(U, V, W)电压上就可以了,但是在程序中还要考虑到提高电机电流的控制性。电流环的频率是 10kHz,另外, PWM 的频率用的是 20kHz,这样的话,因为 PWM 的占空比值每 10 kHz 就更新了一次,作为 PWM 的话变成连续两次更新占空比值了。在这里,将 PWM 的第一次和第二次的占空比值改变的话,可以将 PWM 分辨能力提高两倍。另外, MOSFET 门驱动电路用的是自举电源, PWM 占空比变成 100% 的话,自举电容器便不带电,自举电源的电压会下降。因此为了防止这种情况发生,规定 PWM 占空比的设定范围为 4%~96%。

程序清单 6.3 中,这个方法的输入是 R4 寄存器的 U 相电压(虽然说是电压,单位却不是 V 的整数值)。假设这个值为 1,则在这个值中加上 PWM 周期的一般的值 $\times 2$ ,即  $1 + 799 + 799 = 1599$ 。之后,进行 PWM 占空比的设定范围的确认, PWM 占空比值除以 2,得到  $1599/2 = 799.5$ ,注意小数点后面的 0.5,第一次的 PWM 占空值为 799(50%),第二次的值是 800(50.0625)。这样就能实现与 PWM 分辨能力 2 倍的情况下一样的电机电流的控制性了。

### 6.5.7 对应各种各样的电机

交流伺服电机中有各种厂商生产的电机,即便是一家厂商生产的电机也有很多种类,因此有必要根据所要使用的电机的种类调整 6.5.4 节的 PI(比例积分)控制器的常数。

图 6.25 为其中一个例子。表示的是电机在位置控制的情况下转动一周时转矩电流图。使用的电机是 maxon motor 公司的无刷电机 EC32, 118890 (80W, 48V)。虽然相对于目标转矩电流,延迟了若干秒(2ms 左右),但是测定转矩电流跟随性很好。

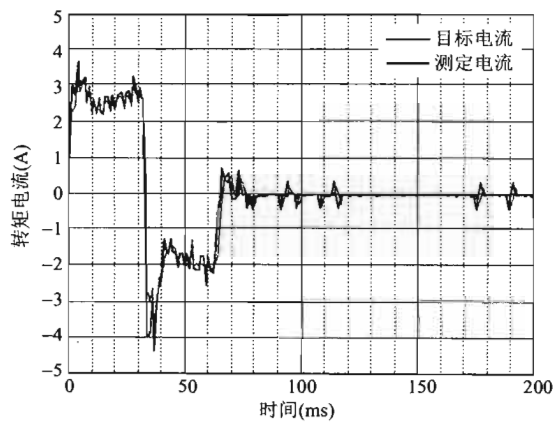


图 6.25 电机转矩电流的响应特性

# 第 7 章

## 基于软件的伺服控制器的设计

本章对采用了基于微控制器软件的数字伺服控制器进行解说。至本章为止主要是通过方框图和传递函数来介绍伺服控制器的,从本章开始,我们以基于软件的设计方法为主进行解说和分析。

### 7.1 伺服控制器的构成

下面整理出了用软件实现的伺服控制器的几点处理内容:

- (1) 读取编码计数器,算出位置和速度。
- (2) 计算反馈(PID 或者是位置/速度环)。
- (3) 前馈 & 输出滤波器。
- (4) 伺服输出(转矩指令)处理。
- (5) 生成目标值。
- (6) 错误处理等。

#### 7.1.1 读取编码计数器的值,计算位置和速度

源程序代码如程序清单 7.1 所示。编码计数器为 16 位。从最初的 16 位计数器扩展处理到 32 位计数器。这种处理的考虑方法为,预先保存前面一个采样的计数器值(lgRawPos),注意上面的 15,14 位的变化。比如,15,14 位从“11”变为“00”的时候,可以判断为进位,此时 32 位计数器的高位自增。相反,15,14 位从

“00”变为“11”的时候,可以判断为退位,此时 32 位计数器的高位自减。

程序清单 7.1 编码计数器值的读取与速度的计算

```

; *****
; *** 位置的读取和速度的计算 *****
; *****

      MOV.L   @lgRawPos,ERO           ; 1 个采样前的计数器值
      MOV.W   @TCNT_2,R1             ; 读取现在的计数器值
      MOV.W   R1,E1                  ; 保存在 E1 里
      AND.B   #H'CO,R0H              ; 调查 1 个采样前的 14,15 位的值
      BNE     IF3CHK
      AND.B   #H'CO,R1H              ; 调查现在的 14,15 位的值
      CMP.B   #H'CO,R1H              ; 判断 14,15 位是否同时为 1
      BNE     MKPOS
      DEC.W   #1,E0                  ; 高位计数器 - 1
MKPOS  MOV.W   E1,R0                  ; 将现在值复制到低位计数器中
      MOV.L   ERO,@_lgRawPos          ; 保存到新的位置中
      MOV.W   @_igDirection,R1        ; 判断传感器的方向
      BEQ     DIFPOS
      NEG.L   ERO                     ; 位置符号反转
      BRA     DIFPOS

IF3CHK CMP.B   #H'CO,R0H              ; 判断 1 个采样前的 14,15 位是否同时为 1
      BNE     MKPOS                  ; 不是 H'CO
      AND.B   #H'CO,R1H              ; 调查现在的 14,15 位的值
      BNE     MKPOS
      INC.W   #1,E0                  ; 高位计数器 + 1
      BRA     MKPOS

DIFPOS:
      MOV.L   @_lgPosition,ER1        ; 读取 1 个采样前的位置
      MOV.L   ERO,@_lgPosition        ; 保存现在的位置
      MOV.L   ERO,ER2
      SUB.L   ER1,ER2                 ; Position-OldPos
      MOV.L   ER2,ER5                 ; 为了控制位置/速度环
      ; 将现在速度(count/sample)保存在 ER5 中
      MOV.W   @_igSrvFreq,R4          ; 伺服采样频率
      MULXS.W R4,ER2                  ; 速度的计算
      MOV.L   ER2,@_lgVelocity        ; 保存现在速度(count/s)

; 速度控制模式的时候将位置微分
MODE1CHK:
      MOV.W   @_igProfile,R3          ; 读取 profile 模式
      BTST    #1,R3L                  ; 控制速度第一个位是 1
      BEQ     DIFEQN
      BTST    #2,R3L                  ; 控制速度第二个位是 0
      BNE     DIFEQN
      MOV.L   ER2,ERO                 ; 从 ERO 中读取速度

```

另外,由于位置计数器是由编码器设置的,如果有需要根据电机的转动方向而想改变位置计数器的增加的方向的情况的话,要设置成能使最终的位置计数器(lgPosition)的符号可以反转的形式。

ER5(count/sample)用于位置/速度环反馈,lgVelocity(count/sec)用于速度控制。

最初执行的中断服务处理是编码计数器的读取,也就是说,只让这个程序执行一次。要想从编码计数器中计算出位置和速度,保持其周期性对于伺服控制器来说是非常重要的。比如说,在编码控制器的读取周期的 10% 的时候发生了紊乱的话,速度误差会变为 10%,电机的控制也会变得紊乱。

### 7.1.2 计算反馈(PID 或者是位置/速度环)

程序代码如程序清单 7.2 所示。由于程序中的全局变量出现得比较多,所以需要对其内容做一些说明。

程序清单 7.2 反馈控制

```

; *****
; ***  伺服计算(ERO 中保存有现在位置或者是现在的速度)  ***
; *****
; R3 = _igProfile, R4 = _igSrvFreq
DIFEQN:
    MOV.W    @_igServoOn,r1        ; if (igServoOn)
    BEQ      NOSRV
    MOV.L    @_lgPosRef,ER1
    SUB.L    ERO,ER1                ; lgPosErr = lgPosRef-lgPosition
    BTST     #1,R3L                 ; 速度控制的第一位是 1
    BEQ      SVERR                  ; 位置控制至差分方程
    BTST     #2,R3L                 ; 速度控制的第二位是 0
    BNE      SVERR                  ; 位置控制至差分方程
; 进行速度控制
    DIVXS.W  R4,ER1                 ; lgPosErr/igSrvFreq
    SUB.L    ER5,ER5                 ; Vel(count/sample) = 0
; 差分方程式
SVERR    MOV.W  R1,@_igPosErr        ; 保存 igPosErr(只限低位的 16 位)
          MOV.W  R1,R0
          MOV.B  #0,R1H               ; signed
          JSR    @ITOF16              ; hgEqnIn = (float16) igPosErr

; ERO = PosErr(float16), ER5 = Vel(count/sample), R3 = _igProfile, R4 = _igSrvFreq
          MOV.L  ERO,ER4              ; 暂时保存 PosErr(float16)

```

```

MOV.B    #0,R1H          ; 执行 ITOF16 的时候,附上符号
MOV.L    ER5,ER0          ; 现在速度 -> ER0
JSR      @ITOF16          ; 将现在速度转换成 FLOAT
MOV.L    ER0,ER3          ; 将现在速度值输入(ER3)反馈控制器
MOV.L    ER4,ER2          ; 设置 hgEqnIn
MOV.W    @_igDifBlock,R1  ; 模块编号
MOV.B    #12,R4L          ;
MULXU.B  R4L,R1           ; 3 * 4 * igDifBlock
MOV.W    #0,E1            ;
MOV.L    #_hgDefCoef,ER5  ; 系数的基址
ADD.L    ER1,ER5          ; CoefPtr = hgDefCoef + igDifBlock
MOV.L    #_hgDifHist,ER4  ; HistPtr = DifHist

MOV.W    @_igDifEqn,R1    ; if (sgDifEqn == 0)
BNE      DIFPPI           ; 至位置/速度环

; PID 计算
DIFPID   JSR      @PIDEQN  ; 调用 PID 控制器
        BRA      ADDFF     ; 至前馈附加处理

; PI 计算
DIFPPI   JSR      @PPIEQN  ; 调用位置/速度环控制器

```

igProfile:控制模式(0 或者 1=位置控制,2 或者 3=速度控制,4=转矩控制)

lgSrvFreq:伺服周期(1000=1kHz)

igServoOn:伺服启用标记(0=伺服关闭,1=伺服启用)

igPosRef:基准位置(count)或者是基准速度(count/s)

igDifBlock:反馈系数的模块编号

hgDifCoef:保存反馈系数的数组变量

hgDifHist:反馈计算空间

igDifEqn:反馈的类别(0=PID,1=位置/速度环)

### 1. 关于浮点运算

在计算和基准值的偏差之前采用整数的计算,在将偏差输入到反馈之前变换为浮点型。由于浮点运算的例行程序实际上是使用了在汇编语言中所描述的快速例程,因此由整型向浮点型的变换则调用名为 ITOF16 的子例程来执行。由于本章中没有涉及在汇编语言中所描述的浮点运算例程,所以在这里认为已经变化成为了 float(单精度浮点)型。

## 2. 反馈系数的设置

表 7.1 给出了 PID 系数(模块 2 是位置/速度环的系数)设置情况。反馈是由 4 个系数模块组成,由 igDifBlock 变量指定的。

表 7.1 PID 的系数(模块 2 是位置/速度环的系数)

系数编号	内 容	系数编号	内 容	系数编号	内 容	系数编号	内 容
0	$K_P(\text{block0})$	3	$K_P(\text{block1})$	6	$K_{PP}(\text{block2})$	9	$K_P(\text{block3})$
1	$K_I(\text{block0})$	4	$K_I(\text{block1})$	7	$K_{PV}(\text{block2})$	10	$K_I(\text{block3})$
2	$K_D(\text{block0})$	5	$K_D(\text{block1})$	8	$K_{IV}(\text{block2})$	11	$K_D(\text{block3})$

## 3. 速度控制时的单位

在速度控制时的 lgPosRef 变量中有基准速度(count/s)。记录现在速度的寄存器 ER0 单位也是 count/s。在计算了和基准值的偏差之后,输入反馈之前将单位转换成 count/sample。

## 7.1.3 前馈 & 输出滤波器

程序代码如程序清单 7.3 所示。附加前馈值之后,计算出输出滤波器。

程序清单 7.3 前馈 & 输出滤波器

```

; 附加前馈值
ADDF    MOV. L    ER2, ER0
        MOV. L    @_hgFFVal, ER1
        JSR      @FADD16          ; EqnIn = hgEqnOut + hgFFVal

; 输出滤波器的计算
MOV. W   @_igOutDepth, E1          ; if (igOutDepth > 0)
BEQ      SRVOUT
MOV. W   @_igOutEqn, E1            ; if (igOutEqn == 1)
BEQ      SRVOUT
MOV. W   @_igOutBlock, R1          ; 模块编号
MOV. B   #40, R2L
MULXU. B R2L, R1V                  ; 10 * 4 * igDifBlock
ADD. W   #48, R1                    ; (10 * 4 * igDifBlock) + 12 * 4
MOV. W   #0, E1
MOV. L   #_hgDefCoef, ER5          ; 系数的基址
ADD. L   ER1, ER5                  ; CoefPtr = hgDefCoef + (10 * igDifBlock) + 12
MOV. L   #_hgOutHist, ER4          ; HistPtr = DifHist
MOV. L   ER0, ER2                  ; 设置 hgEqnIn
MOV. W   @_igOutDepth, R3          ; 设置环路数
JSR      @IIREQN                   ; IIR Equation
MOV. L   ER2, ER0

```

由于程序中有多个全局变量,在这里说明一下。

hgFFVal:前馈值(Float)

igOutDepth:输出滤波器的级联数(0=没有滤波器,1=一级,2=二级)

igOutEqn:输出滤波器的有效/无效(0=无效,1=有效)

igOutBlock:输出滤波器的系数模块编号

hgOutHist:输出滤波器的计算范围

输出滤波器系数的配置如表 7.2 所示。输出滤波器系数在反馈系数之后设置,可以同时拥有 4 个系数模块。被设置成可以由 igOutBlock 变量指定的形式。

表 7.2 IIR 系数(可以分成两段 CUSMATE)

系数编号	内 容	系数编号	内 容	系数编号	内 容	系数编号	内 容
12	$a_2(\text{block0})$	22	$a_2(\text{block1})$	32	$a_2(\text{block2})$	42	$a_2(\text{block3})$
13	$b_2(\text{block0})$	23	$b_2(\text{block1})$	33	$b_2(\text{block2})$	43	$b_2(\text{block3})$
14	$a_1(\text{block0})$	24	$a_1(\text{block1})$	34	$a_1(\text{block2})$	44	$a_1(\text{block3})$
15	$b_1(\text{block0})$	25	$b_1(\text{block1})$	35	$b_1(\text{block2})$	45	$b_1(\text{block3})$
16	$b_0(\text{block0})$	26	$b_0(\text{block1})$	36	$b_0(\text{block2})$	46	$b_0(\text{block3})$
17	$a_2(\text{block0})$	27	$a_2(\text{block1})$	37	$a_2(\text{block2})$	47	$a_2(\text{block3})$
18	$b_2(\text{block0})$	28	$b_2(\text{block1})$	38	$b_2(\text{block2})$	48	$b_2(\text{block3})$
19	$a_1(\text{block0})$	29	$a_1(\text{block1})$	39	$a_1(\text{block2})$	49	$a_1(\text{block3})$
20	$b_1(\text{block0})$	30	$b_1(\text{block1})$	40	$b_1(\text{block2})$	50	$b_1(\text{block3})$
21	$b_0(\text{block0})$	31	$b_0(\text{block1})$	41	$b_0(\text{block2})$	51	$b_0(\text{block3})$

#### 7.1.4 伺服输出(转矩指令)处理

程序清单 7.4 所示为伺服输出(转矩指令)的处理程序。伺服输出需要进行以下的处理:

##### 1. 由浮点型向整数型变换

由于给伺服输出(转矩指令)的是整数,所以在这里要将浮点型转换成整数型。

##### 2. 驱动极性的处理

因为有伺服输出的极性变化的情况,所以要进行伺服输出极性(驱动极性)的处理。



程序清单 7.4 伺服输出处理

```

; 这之后是伺服输出的更新
; 为了防止在加上驱动补偿的时候溢出,把数据扩展成 32 位型的
SRVOUT  MOV.B    #0,R1H                      ; signed
        JSR      @FTOI16                      ; 转换成整数
        MOV.W    @_igDrvPol,R1                ; 确认驱动极性
        BEQ      SKIPPOL
        NEG.W     R0                          ; 符号取反
        CMP.W    #H'8000,R0                  ; H'8000 就算是 NEG 也是 H'8000
        BNE      SKIPPOL
        NOT.W     R0

SKIPPOL:
        EXTS.L    ER0                          ; 扩展成 32 位型的
        MOV.W     @_igDrvOff,R1                ; 驱动补偿
        EXTS.L    ER1                          ; 扩展成 32 位型的
        ADD.L     ER1,ER0                      ; EqnOut + DrvOff
        BMI       MINLIM                      ; 转向负极限

; 正极限处理
        MOV.W     @_igDrvLim,R1                ;
        EXTS.L    ER1                          ;
        CMP.L     ER1,ER0                      ; (EqnOut + DrvOff) - (int)sgDrvLim
        BMT       OUTSRV
        MOV.L     ER1,ER0                      ; (int) sgDacVal = (int)sgDrvLim
        BRA       OUTSRV

; 负极限处理
MINLIM  MOV.W     @_igDrvLim,R1                ; 读取驱动极限
        EXTS.L    ER1                          ; 扩展成 32 位型的
        NEG.L     ER1                          ; 符号取反
        CMP.L     ER1,ER0                      ; (EqnOut + DrvOff) - (int) - sgDrvLim
        BPL       OUTSRV
        MOV.L     ER1,ER0                      ; (int) sgDacVal = (int) - sgDrvLim

; 伺服输出处理
OUTSRV  MOV.W     R0,@_igSrvOut                ; 伺服输出保存

```

### 3. 驱动补偿(转矩补偿)的加法

驱动补偿是为在垂直位置使用的情况而设置的。由于垂直位置不能忽视重力的影响,如果伺服停止了的话,位置就会下落,因此,驱动补偿是为了即使在伺服停止时候使得伺服输出(转矩指令)也能起作用的目的而设置的。例如在从位置控制转向速度控制的时候,需要暂时停止伺服,此时在电机中会产生和重力相平衡的转矩,位置就不会下落了。

## 4. 驱动极限(扭矩极限)的处理

在给电机加上转矩极限的情况下,由驱动极限来指定。驱动极限由整数的绝对值来指定。

## 7.1.5 标准值的生成

程序代码如程序清单 7.5 所示。标准值的生成在伺服输出处理之后进行。也就是说,在这里生成的标准值在下一次伺服取样中会被使用到。按照这样的处理顺序,可以减少伺服环上所浪费的时间。

程序清单 7.5 标准值的生成

```

; rebalance generation 的调用
RGSEL  MOV.W  @_igProfile,R1      ; PROFILE 模式
        BEQ    CALRG0              ; RG0
        CMP.W  # 21,R1             ; Profile - 2
        BEQ    CALRG2              ; RG2
        BMI    CALRG1              ; RG1
        CMP.W  # 4,R1              ; Profile - 4
        BEQ    CALRG4              ; RG4

; 外部转矩控制模式
CALRG4  MOV.W  @_igRg4Data,R1
        MOV.W  @_igDrvPol,R0      ; 驱动极性的确认
        BEQ    SETSRVOUT
        NEG.W  R1
SETSRVOUT:
        MOV.W  R1,@_igSrvOut      ; 直接设定伺服输出
        BRA    UPSTR

; 外部速度控制模式
CALRG3  JSR    @_Rg2              ; 标准速度的生成
        BRA    UPSTR

; 内部速度控制模式
CALRG2  JSR    @_Rg2              ; 标准速度的生成
        BRA    UPSTR

; 外部位置控制模式
CALRG1  JSR    @_Rg0              ; 标准位置的生成
        JSR    @_RG0FF            ; 前馈值的更新
        BRA    UPSTR

; 内部位置控制模式
CALRG0  JSR    @_Rg0              ; 标准位置的生成
        JSR    @_RG0FF            ; 前馈值的更新

UPSTR   ; 伺服状态的更新

```

## 7.2 反馈控制器

### 7.2.1 PID 控制器的运算步骤

#### 1. 方框图

图 7.1 是 PID 控制器的方框图。按照方框图所示,从上到下依次为比例(P)、微分(D)、积分(I)。

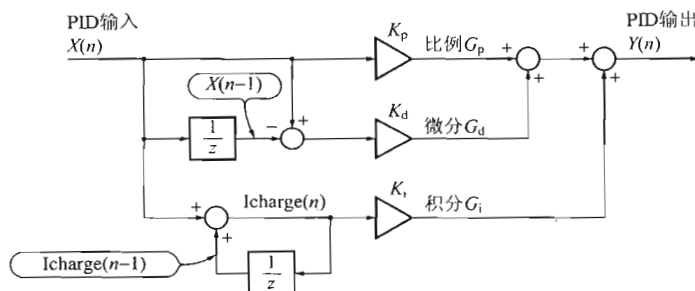


图 7.1 PID 控制器的方框图

#### 2. 传递函数的表达式

PID 控制器的传递函数表达式如下所示

$$\frac{Y(n)}{X(n)} = K_p + (1 - z^{-1}) \cdot K_d + \frac{K_i}{1 - z^{-1}}$$

#### 3. 软件程序清单

如果严格按照图 7.1 所示的方框图开发软件的话,结果就如程序清单 7.6 所示的那样。这个路径的输入由 fgEqnIn(float)给出,输出是这个路径的返回值(float)。计算使用的 PID 系数是 CoefPtr 的指针变量中预先保存的值。另外,计算中使用的计算区域使用的是 HistPtr 的指针变量。

程序清单 7.6 PID 控制器

```
float pidEqu (void)
{
    float R0;           //PID 输入输出数据
    float R1;           //计算用寄存器

    R0 = fgEqnIn;       //r0 = X(n)
    R1 = R0 - HistPtr[0]; //r1 = X(n) - X(n-1)
```

```

HistPtr[0] = R0;           //为了下次的计算保存 X(n)
R0 = CoefPtr[0] * R0;      //r0 = Gp = Kp * X(n)
R1 = CoefPtr[2] * R1;      //r1 = Gd = Kd * (X(n) - X(n-1))
R0 = R0 + R1;             //r0 = Gp + Gd
R1 = HistPtr[1];          //r1 = Icharge(n-1)
R1 = HistPtr[0] + R1;      //r1 = Icharge(n-1) + X(n)
HistPtr[1] = R1;          //为了下次的计算保存 Icharge(n)
R1 = CoefPtr[1] * R1;      //r1 = Gi = Ki * Icharge(n)
R0 = R1 + R0;             //r0 = Gp + Gi + Gd
return R0;
}
//全局变量
//fgEqnIn = PID 输入数据
//HistPtr[0] = 所保存的一个采样前的数据 X(n-1)
//HistPtr[1] = 所保存的一个采样前的积分项 Icharge(n-1)
//CoefPtr[0] = 所保存的比例增益 Kp
//CoefPtr[1] = 所保存的积分增益 Ki
//CoefPtr[2] = 所保存的微分增益 Kd
    
```

## 7.2.2 位置/速度环控制器计算路径

### 1. 方框图

图 7.2 是位置/速度环控制器的方框图。速度反馈环是位置反馈环的副回路。

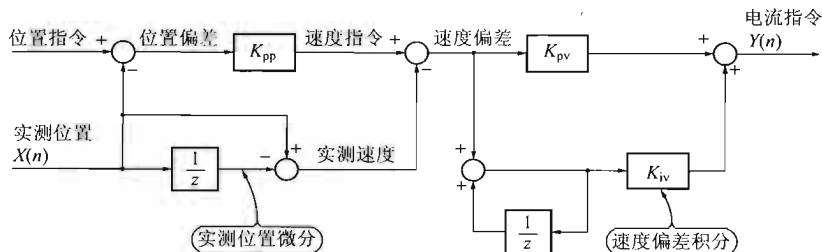


图 7.2 位置/速度环路控制器的方框图

### 2. 传递函数的表达式

在这个方框图中,如果设位置指令为 0,输入  $X(n)$  为实际测量位置,输出  $Y(n)$  为电流指令,则传递函数如下所示

$$\frac{Y(n)}{X(n)} = \frac{(K_{pp} + 1) \cdot (K_{pv} + K_{iv}) - (2 \cdot K_{pv} + K_{iv} + K_{pp} \cdot K_{pv}) \cdot z^{-1} + K_{pv} \cdot z^{-2}}{1 - z^{-1}}$$

### 3. 软 件

程序清单 7.7 是位置/速度环控制器的程序。与图 7.2 的方框图不同的地方是由于使用了实测速度计算伺服环中断的主路径,本程序程序清单中的输入是位置偏差  $fgPosErr(float)$  和实测速度  $fgVel(float)$ 。输出是这个路径的返回值(float)。计算时使用的系数是预先在  $CoefPtr$  的指针变量中保存着的。另外,计算中使用的计算区域是  $HistPtr$  的指针变量。

程序清单 7.7 位置/速度环控制器

```
float PpiEqn (void)
{
    float R0;           //输入数据
    float R1;           //计算用寄存器
    float R2;           //计算用寄存器

    //位置环
    R0 = fgPosErr;       //r0 = err(n)
    R1 = fgVel;          //r1 = vel(n)
    R0 = CoefPtr[0] * R0; //r0 = Gpp = Kpp * X(n)
    R2 = R0 - R1;        //r2 = verr(n) = Gpp * vel(n)
    //全局变量
    R0 = CoefPtr[1] * R2; //r0 = Gpv = Kpv * verr(n)
    R1 = HistPtr[0];      //r1 = Icharge(n-1)
    R1 = R1 + R2;         //R1 = Icharge(n-1) + verr(n)
    HistPtr[0] = R1;      //为了下次的计算所保存的 Icharge(n)
    R1 = CoefPtr[2] * R1; //r1 = Giv = Kiv * Icharge(n)
    R0 = R1 + R0;         //r0 = Gpv + Giv
    return R0;
}

//全局变量
//fgPosErr = 位置偏差
//fgVel = 实测速度
//HistPtr[0] = 所保存的一个采样前的积分项 Icharge(n-1)
//CoefPtr[0]所保存的位置比例增益  $K_{pp}$ 
//CoefPtr[1]所保存的速度比例增益  $K_{pv}$ 
//CoefPtr[2]所保存的速度积分增益  $K_{iv}$ 
```

## 7.3 IIR 数字滤波器计算路径

### 1. 方框图

输出滤波器使用图 7.3 所示的 IIR(Infinite impulse Response)数字滤波

器。图中,  $1/z$  表示一个采样时间的延迟。数字滤波器分为 IIR 和 FIR 两种, 在得到相同振幅特性前提下, IIR 的计算量相对于 FIR 要小。

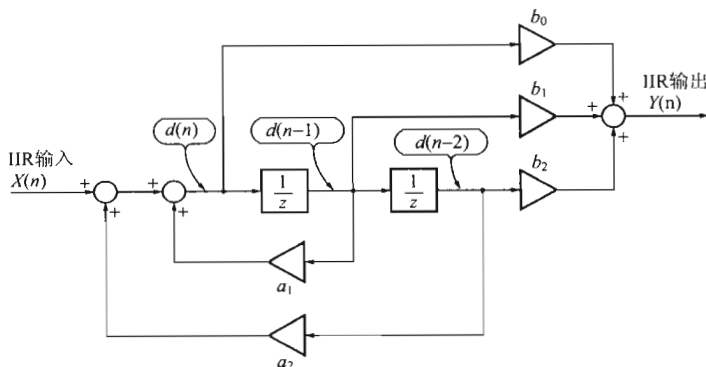


图 7.3 IIR 数字滤波器的方框图

## 2. 传递函数的表达

如下所示, 用传递函数来表示图 7.3 的 IIR 数字滤波器的方框图

$$\frac{Y(n)}{X(n)} = \frac{b_0 + b_1 \cdot z^{-1} + b_2 \cdot z^{-2}}{1 - a_1 \cdot z^{-1} - a_2 \cdot z^{-2}}$$

为了提高软件的运算速度, 相对于用专用术语表示的一般数字滤波器的传递函数, 这里用  $a_1$  和  $a_2$  来实现符号翻转。

## 3. 软 件

程序清单 7.8 所示的是 IIR 数字滤波器的程序。这个程序本来是用 C 语言写的 DSP 的计算路径, 但是后来在汇编语言的基础上有意识地使用 CPU 命令控制语句重新改写了程序。

这个路径的输入由 `fgEqnIn(float)` 给出。输出是这个路径的返回值。计算时使用的系数是在 `CoefPtr` 的指针变量里预先保存的。另外, 计算中使用的计算区域是 `HistPtr` 的指针变量。这个计算路径本身可以级联 8 级。保存级联数的变量是 `igN`。像这样在实现高次 (3 次以上) 的滤波器特性情况下, 在 IIR 数字滤波器中, 使用二次滤波器的级联是非常普遍的 (实用的)。

程序清单 7.8 IIR 数字滤波器

```

float IirEqn (void){
    int      i,n;
    float    y[9];
    float     * ar0,* ar1;
    float     ral,rb1,ra2,rb2;

    ar0 = CoefPtr;           //系数指针
    ar1 = HistPtr;           //节点指针
    y[0] = fgEqnIn;          //输入数据
    n = 0;
    for(i = 0; i < igN; i++){
        ral = ar0[0] * ar1[0];    //ral = a2 * d(n-2)
        ar0++;                  //将系数指针 + 1 a2→b2
        rb1 = ar0[0] * ar1[0];    //rb1 = b2 * d(n-2)
        ar0++;                  //系数指针 + 1 b2→a1
        ar1++;                  //节点指针 + 1 d(n-1)→d(n-1)
        ral += y[n];             //First sum term of d(n,n)
        ra2 = ar0[0] * ar1[0];    //ra2 = a1 * d(n-1)
        ar0++;                  //系数指针 + 1 a1→b1
        rb2 = ar0[0] * ar1[0];    //rb2 = b1 * d(n-1)
        ral += ra2;              //Second sum term of d(n,n)
        rb1 += rb2;
        ar0++;                  //系数指针 + 1 b1→b0
        ar1++;                  //节点指针 + 1 d(n-1)→d(n)
        rb2 = ar0[0] * ral;       //b0(n) * d(n)
        ar0++;                  //系数指针 + 1 b0→a2(为下个环路做准备)
        ar1++;                  //节点指针 + 1 d(n)→d(n-2)(为下个环路做
                                //准备)

        HistPtr[n*3+0] = HistPtr[n*3+1];    //d(n-2)更新
        HistPtr[n*3+1] = ral;               //d(n-1)更新
        y[n+1] = rb1 + rb2;                 //IIR 输出更新
        n++;
    }
    return y[n];
}

//全局变量
//fgEqnIn = IIR 输入数据
//igN = IIR 的级联数
//HistPtr[0] = d(n-2)保存(第一段)
//HistPtr[1] = d(n-1)保存(第一段)
//HistPtr[2] = 未使用
//HistPtr[3] = d(n-2)保存(第二段)
//HistPtr[4] = d(n-1)保存(第二段)
//HistPtr[5] = 未使用
// :
// :

```

```

//
//Coefptr[0] = IIR 系数 a2(第一段)
//Coefptr[1] = IIR 系数 b2(第一段)
//Coefptr[2] = IIR 系数 a1(第一段)
//Coefptr[3] = IIR 系数 b1(第一段)
//Coefptr[4] = IIR 系数 b0(第一段)
//Coefptr[5] = IIR 系数 a2(第二段)
//Coefptr[6] = IIR 系数 b2(第二段)
//Coefptr[7] = IIR 系数 a1(第二段)
//Coefptr[8] = IIR 系数 b1(第二段)
//Coefptr[9] = IIR 系数 b0(第二段)
// :
// :

```

## 7.4 前馈计算路径

### 1. 方框图

图 7.4 表示的是前馈方框图。将位置一次微分后的结果就是速度(fVel)，将此和速度项增益(fgFFVel)做乘积运算。将速度一次微分后的结果就是加速度(fAcc)，将此和加速度增益(fgFFAcc)做乘积运算。

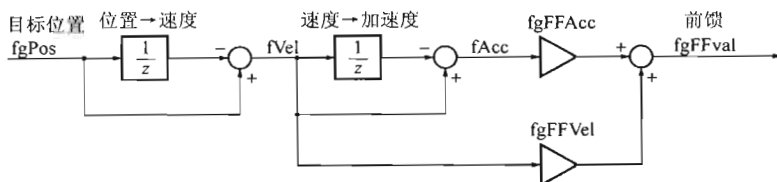


图 7.4 前馈方框图

速度和加速度的乘积结果的合成值即为前馈电流(fgFFVal)。

### 2. 传递函数的表达

如果将图 7.4 所示的前馈方框图改写成传递函数的话,则

$$fgFFVal/fgPos = (1 - z^{-1}) \cdot fgFFVel + (1 - 2 \cdot z^{-1} + z^{-2}) \cdot fgFFAcc$$

前馈也是数字滤波器的一种,属于 FIR(Finite Impulse Response)。

### 3. 软件

如果严格按照图 7.4 所示的方框图开发软件的话,结果就如程序清单 7.9 所示。这个路径的输入是 fgPos(float)变量,输出是 fgFFVal(float)。



程序清单 7.9 负反馈的计算

```
//前馈的计算
fVel = fgPos - fgPosOld;           //计算速度
fAcc = fVel - fgVelOld;           //计算加速度
fgFFval = fVel * fgFFVel + fAcc * fgFFAcc; //计算前馈

//为了下次采样,保存数据
fgPosOld = fgPos
fgVelOld = fVel;

//全局变量
//fgPos = 现在的目标位置
//fgPosOld = 一个采样前的目标位置
//fgVelOld = 一个采样前的目标速度
//fgFFVel = 速度项增益
//fgFFAcc = 加速度项增益
//fgFFVal = 前馈的计算结果
```

## 7.5 目标轨迹生成器

所谓目标轨迹生成器,是在位置控制的时候,赋予新位置时,需要从现在的位置找到新位置的轨迹,依照这个轨迹让电机旋转的时候,决定是否移动模块。

### 7.5.1 目标轨迹生成相关的基础知识

在用软件设计实际的目标轨迹生成的情况下,需要先解决几个问题。

#### 1. 单位

在伺服控制器中,需要用到加速度、速度及长度(位置)这几个物理量。一般的加速度的单位是  $\text{m/s}^2$ ,或者是设重力加速度  $9.8\text{m/s}^2$  为 1 个 G。由于交流伺服电机是转动的,用长度单位(m)比较困难。虽然在转动系统中有以 RAJIAN 为单位的,为了在软件的使用中的方便,我们采用编码计数器的值来作为长度(位置)单位。这样的话,加速度的单位就变成了  $\text{count/s}^2$ ,速度的单位就变成了  $\text{count/s}$ ,位置的单位则为 count。

#### 2. 基于恒定加速度来生成目标轨迹

在目标轨迹生成器的设计之中,需要很好地理解加速度、速度和位置的关

系。图 7.5 所表示的是采用恒定加速度生成目标轨迹的情况。将加速度积分的话就能得到速度,将速度积分的话就能得到位置,实际上的目标轨迹生成器也用的是同样的计算。这是一种台形速度曲线。

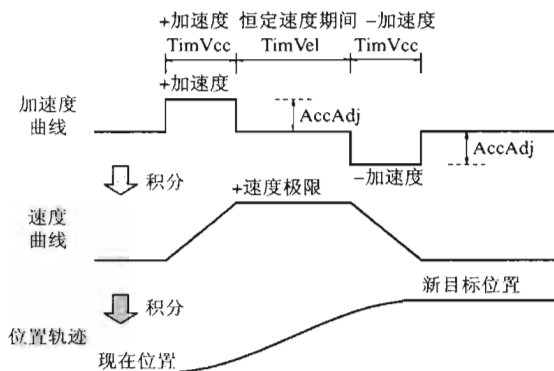


图 7.5 基于恒定加速度的目标位置轨迹

### 3. 积分运算

目标轨迹生成器是基于每个伺服采样周期进行积分计算,每个伺服采样的目标位置或者是目标速度算出来的。

那么,这个积分运算应该用软件如何实现呢。设加速度为  $a$ ,速度为  $v$ ,位置为  $x$ ,时间为  $t$ ,我们来简单写个 C 语言风格的程序,如下所示

```
i = 0;   v = 0;   x = 0
a = 1000                                //1000[count/s²]
t = 0.001;                             //0.001[s]
for(i = 0; i < 1000; i++){
    v + = a * t;                        //计算速度
    x + = v * t;                        //计算位置
}
```

基于图 7.6 的计算结果,1s 后的速度是 1000(count/s),位置是 500(count),实际的软件是按照每个伺服采样周期进行分段处理的,比起上面的程序可能稍微复杂一点,但是积分的计算方法是一样的。

另外,恒定加速度时的加速度和位置的关系如下所示

$$x = \frac{1}{2}at^2 = \frac{1}{2} \times 1000 \times 1^2 = 500$$

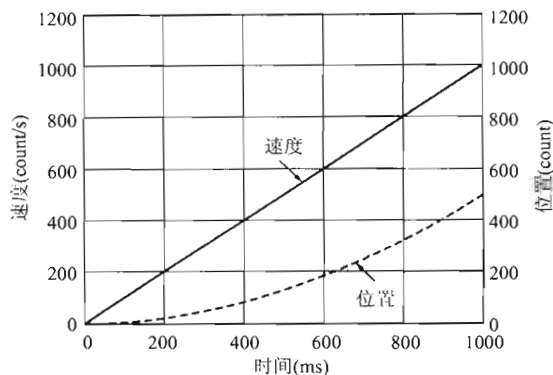


图 7.6 积分计算的例子

这和软件的计算结果一致。

### 7.5.2 目标位置轨迹的生成方法

位置控制是按照每个伺服周期分别生成标准位置的。但是这种计算不仅需要事先知道加速度和相乘的次数,还需要进行两次积分运算。如果加速度和相乘次数有一点误差的话,到最终的目标位置也会生成很大的误差。在位置控制中,目标位置误差过大的话就会本末倒置,所以在给出新目标位置的时候,需要预先进行速度和其相乘次数的修正。

在给定新目标位置的时候计算轨迹,进行加速度和相乘次数的修正。这种计算方法的特征为,在台形曲线的情况下,和移动的距离没有关系,在短时间内能够完成计算。

程序如程序清单 7.10 所示。计算从最初给定的加速度和速度极限开始到速度达到速度极限位置的时间。从算得的时间来计算出电机在加速和减速中所需的移动距离。虽然加速度  $a$  和移动距离存在以下的关系式

$$x = 1/2(at^2)$$

但是一起计算加速和减速时的总计移动距离的时候,  $1/2$  便可以消除。

其次,将加速减速时的移动距离和给定的目标移动距离进行比较的话,可以判断有无恒定速度的期间。在有恒定速度的情况下,达到的速度 = 速度极限;在没有恒定速度的情况下,达到的速度 < 速度极限。

程序清单 7.10 位置轨迹的计算

```

int TimVel;           //恒定速度的采样数
int TimAcc;           //加速度采样数
float AccAdj;         //修正后的加速度

float VelLim;         //速度极限
float AccLim;         //加速度
float Dest;           //目标移动距离
float Ts;             //伺服采样周期

float Time;           //用于加速度时间计算
float Pos, DelPos;    //用于移动距离计算
float Vel;            //用于速度计算

Time = VelLim/AccLim;
Pos = AccLim * Time * Time;           //到速度极限为止所花的时间
DelPos = Dest - Pos;                  //加速期间移动的距离
if(DelPos>0){                          //和目标移动距离的比较
    TimVel = (int)(DelPos/VelLim/Ts + 0.5); //有恒定速度期间的情况
}else{                                //恒定采样数
    TimVel = 0;                        //无恒定速度期间
    Time = sqrt(Dest/AccLim);          //恒定速度采样数 = 0
}
TimAcc = (int)(Time/Ts + 0.5);        //加速度采样数的计算

Vel = AccLim * ((float)TimAcc * Ts);  //达到的速度
Pos = Vel * ((float)TimAcc * Ts);     //加减速部分
Pos += Vel * ((float)TimVel * Ts);    //速度恒定部分
AccAdj = AccLim * (Dest/Pos);         //加速度极限修正

```

有加速度的时间和恒定速度的时间可以转换成伺服采样数,最后从给定的加速度及加速度伺服采样数和恒定速度下的伺服采样数计算出总计移动距离,将加速度修正为趋向于和目标移动距离一致的值。

### 7.5.3 目标速度轨迹的生成方法

目标速度轨迹采用图 7.7 所示的恒定加速度。在每个伺服采样周期中分别进行如图所示的积分计算,分别算出每个伺服采样周期的目标速度。但是,位置和轨迹相同的,这种计算必须要预先知道加速度的极限值和相乘次数。另外,由于所进行的是积分计算,即使加速度的极限值和相乘次数有一点误差,最后的计算出来的目标速度也会很不准确。

因此,在给定新的目标速度的时候,需要预先修正加速度和其相乘的次数,

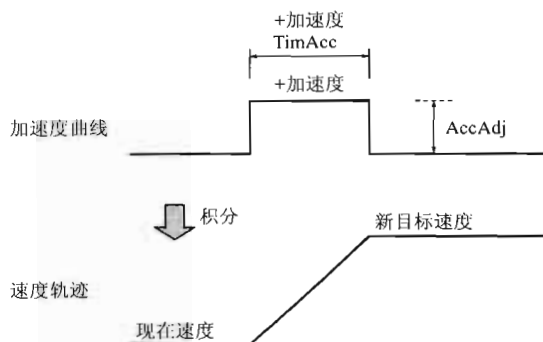


图 7.7 基于恒定加速度的目标速度轨迹

使其不至影响最终目标值的计算。

程序如程序清单 7.11 所示。首先,计算出在给定加速度的条件下达到给定目标速度的时间。将达到给定目标速度的时间变换成加速度伺服采样数。

程序清单 7.11 速度轨迹的计算

```
int TimAcc;           //加速度采样数
float AccAdj;         //修正后的加速度

float AccLim;         //加速度
float Dest;           //目标速度
float Ts;             //伺服采样周期

float Time;           //用于计算加速度的时间
float Vel;            //用于速度的计算

Time = Dest/AccLim;   //达到给定速度所花的时间
TimAcc = (int)(Time/Ts + 0.5); //计算加速度的采样数

Vel = AccLim * ((float)TimAcc * Ts); //达到的速度
AccAdj = AccLim * (Dest/Vel); //加速度修正
```

最后,从给定加速度和加速度伺服采样数计算出所达到的速度,修正加速度使得最后的速度趋近于目标速度。

# 第 8 章

## 基于汇编语言实现的 伺服控制器高速化

本章主要针对基于汇编语言实现伺服控制器的高速化进行讲解。用微控制器进行分类的处理器多数没有搭载浮点运算单元。由于在伺服控制器中需要用到浮点运算,所以也可能会降低伺服控制器的性能。下面结合 CPU 的情况来设计浮点格式,编写汇编语言实现浮点运算路径。

### 8.1 IEEE 标准的单精度浮点

#### 8.1.1 单精度浮点的位配置

图 8.1 表示的是 IEEE 标准的单精度浮点的位配置标准。单精度浮点用 4 字节来表示,从最高位开始分别由符号位(1 位)、指数位(1 位)、小数位(23 位)构成。这种结构的特征是将最高位的 31 位配置为符号位,因此指数位的 8 位就从 30 位开始配置到 23 位,如图 8.1 所示,以字节为单位来看的话,就相当于移动了一位,所以小数位也不再是 24 位了,变成了 23 位。

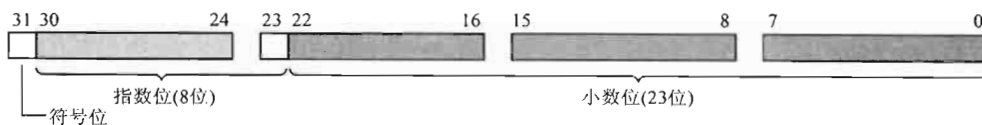


图 8.1 IEEE 标准的单精度浮点的位配置

## 8.1.2 单精度浮点的表示方法

符号位是 1 位数据。“0”表示正数，“1”表示负数的意思。

指数位是 8 位数据。用 2 的指数来表示,IEEE 中是用 127 来偏移的,比如  $2 \times 1$  的情况是用  $2^{(128-127)} = 2^1 = 2$  的方法来实现的。

小数位是 23 位数据。单精度浮点的精度是 24 位,有一位不足。小数位表示的是从 1.0 到 1.99999988 的数,最高位必须是“1”。因此可以默认最高位为“1”,从而可以节省一位。

图 8.2(a)表现的是 0.0 的单精度浮点。0.0 比较特殊,每个位都是“0”。

图 8.2(b)表示的是 +1.0 的单精度浮点。+1.0 是符号位,如果将指数位和小数位分开的话,便可以表示  $+2^{(127-127)} \times 1.0$ ,符号位为“0”,指数位为 127,小数位为 0。

图 8.2(c)表示的是 +2.0 的单精度浮点。+2.0 是符号位,如果将指数位和小数位分开的话,便可以表示  $+2^{(128-127)} \times 1.0$ ,符号位是“0”,指数位为“128”,小数位为 0。

图 8.2(d)表示的是 +3.1 的单精度浮点。+3.1 是符号位,如果将指数位和小数位分开的话,便可以表示  $+2^{(128-127)} \times 1.55$ ,符号位是“0”,指数位是 128。到这里还算挺简单的,但是到小数位就有点复杂了。在小数位的 1.55 中,1 是

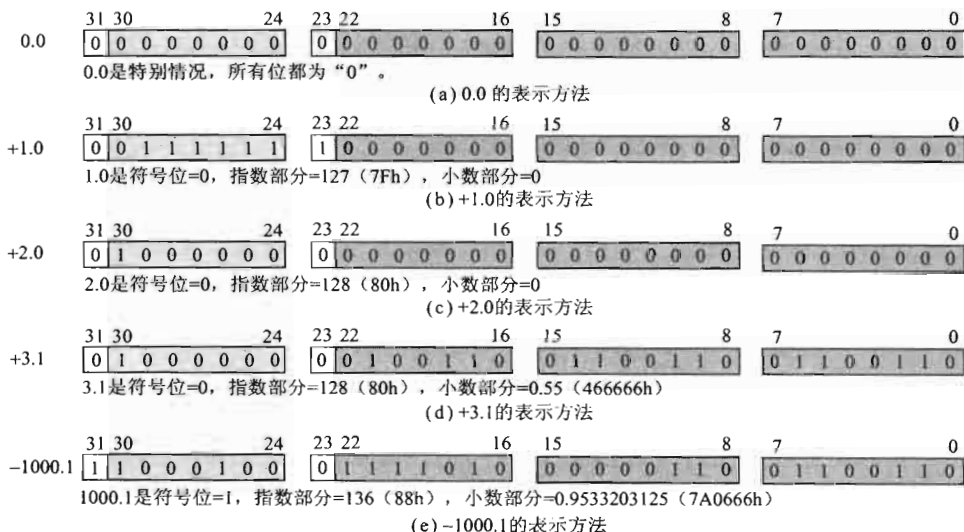


图 8.2 IEEE 单精度浮点的表示方法

默认的,小数位表示的是 0.55。由于小数位是 23 位的,  $0.55 \times (2^{23}) = 4613734.4$ , 在十六进制中表示为 466666h, 由于有余数, 所以产生了误差。

由于小数位是  $4613734 \div (2^{23}) = 0.5499999523162841796875$ , 所以可以得到

$$2^1 \times 1.5499999523162841796875 = 3.099999904632568359375$$

图 8.2(e)表示的是一1000.1单精度浮点。-1000.1是符号位,将指数部分和小数部分分开的话,可以表示  $-2^{(136-127)}$ , 符号位是“1”,指数部分是136,小数部分则是默认的1,所以其表示的是 0.9533203125。由于小数部分是 23 位的,  $0.9533203125 \times (2^{23}) = 7997030.4$ , 用十六进制表示为 7A0666h。在这里面产生了一些误差。

由于小数部分是  $7997030 \div (2^{23}) = 0.9533202648162841796875$ , 所以  $-2^9 \times 1.9533202648162841796875 = -1000.0999755859375$ 。

## 8.2 16 位精度浮点

H8S/2000 CP 是 16 位微处理器。这种 CPU 的乘法命令是 16 位  $\times$  16 位 = 32 位,除法命令是 32 位  $\div$  16 位 = 16 位,如果采用计算精度是 24 位的单精度浮点运算不是非常理想。小数部分需要计算很多次。

因此,设计了小数部分是 16 位的浮点结构,以提高伺服运算的计算速度。

### 8.2.1 16 位精度浮点的位的设置

16 位精度浮点与单精度浮点一样,都是由 4 个字节组成的。虽然和单精度浮点相比,精度上相差了 8 位,可以用 3 字节来实现,但是 3 字节的数值类型无论是在汇编语言还是在 C 语言中,所用的数值类型(long 或者 float 的意思)都不是 3 字节的,所以这里应该用 4 字节来表示。如图 8.3 所示,从第 32 位到第 16 位都没有被用到,全部置零。第 24 位是符号位。指数部分从第 23 位到

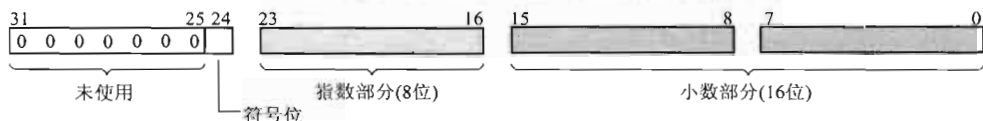


图 8.3 16 位精度浮点的位的配置



第 16 位,正好是从高位数下来第二个字节的位置。小数部分是从第 15 位开始到第 0 位的十六位。在 4 个字节当中低位分配 2 个字节。

## 8.2.2 16 位精度浮点的表示方法

符号位是 1 位的数据。“0”表示正数,“1”表示负数。

指数部分是 8 位的数据。虽然表示的是 2 的指数,但是与 IEEE 的结构不同,是带符号的整数。比如说  $2^{-1}$  的情况表示为  $2^{-1}$  (FFh)。

小数部分是 16 位的数据。并不是默认的 1,而是最高位必须为“1”。

图 8.4(a)表示的是 0.1 的 16 位精度浮点的表示方法。0.0 比较特别,所有位都是“0”。

图 8.4(b)表示的是 +1.0 的 16 位精度浮点的表示方法。+1.0 是符号位,将指数部分和小数部分分开的话,可以表示为  $+2^{(0)} \times 1.0$ 。其中符号位是 0,指数部分为 0,小数部分为 32768(8000h)。

图 8.4(c)表示的是 +2.0 的 16 位精度浮点的表示方法。+2.0 是符号位,将指数部分和小数部分分开的话,可以表示为  $+2^{(1)} \times 1.0$ 。其中符号位是 0,指数部分是 1,小数部分是 32768(8000h)。

图 8.4(d)表示的是 +3.1 的 16 位精度浮点的表示方法。+3.1 是符号位,

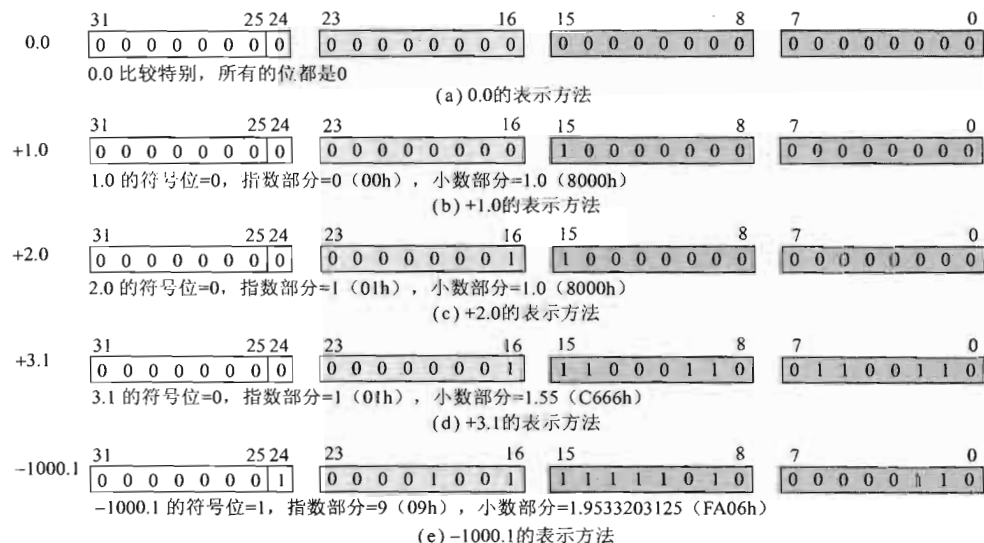


图 8.4 16 位精度浮点的表示方法

将指数部分和小数部分分开的话,可以表示为  $+2^{(1)} \times 1.55$ 。其中符号位是 0,指数部分是 1,小数部分是  $1.55 \times (2^{15}) = 50790.4$ ,用十六进制可以表示为 C666h。由于有余数,所以会产生一些误差。

由于小数部分是  $50790 \div (2^{15}) = 1.54998779296875$ ,所以  $2^1 \times 1.54998779296875 = 3.0999755859375$ 。

图 8.4(e)表示的是  $+1000.1$  的 16 位精度浮点的表示方法。 $-1000.1$  是符号位,将指数部分和小数部分分开的话,可以表示为  $-2^{(9)} \times 1.9533203125$ 。其中符号位是 1,指数部分是 9,小数部分是  $0.9533203125 \times (2^{15}) = 64006.4$ ,用十六进制可以表示为 FA06h。这里也会产生误差。

由于小数部分是  $46006 \div (2^{15}) = 1.95330810546875$ ,所以  $-2^9 \times 1.95330810546875 = -1000.09375$ 。

### 8.2.3 16 位精度浮点的计算精度

16 位精度浮点的有效位为 5 位,精度不是很好。虽然不能像计算器那样使用,不过作为反馈控制器来说,精度是足够了。

反馈控制器的输入是位置偏差或者是速度偏差,通常状态是数值计数器的整数。反馈控制器的输出是转矩电流指令,化成整数值,这个数据的范围也是取决于电机电流检测的分析能力的。第 2 章中介绍的电机驱动电路中最大是 1500count。所以,最终的结果中有效位数会减少,仅提高反馈控制器的计算精度的话,也没有什么意义。

### 8.2.4 16 位精度浮点的计算方法

下面针对 16 位精度的浮点运算方法进行介绍。因为所有的计算方法都是用汇编语言写的,为了让 C 语言也可以调用,ER0 和 ER1 以外的寄存器都避开不用。

#### 1. FADD16(浮点的加法运算)

程序如程序清单 8.1 所示。加法运算的输入数据是 ER0(参数 1)和 ER1(参数 2),运算结果保存在 ER0 中,在 R1L 中保存出错信息。

首先,确认输入数据是否为零。参数 1 或者参数 2 是 0 的情况下,没有必要进行加法运算,不为零的情况下就直接输出结果。

第二步是位对齐。与指数值大的位对齐。在小数部分移位的同时指数部分增加,使指数部分的位对齐。

位对齐后确认符号。判断是加法运算还是减法运算之后计算小数部分。

只有指数部分溢出时加法运算才会出错。

程序清单 8.1 浮点的加法运算

```

; *****
; *****
; ***** 浮点的加法运算 *****
; ***** Input:  ER0 = 4byte input data1 *****
; *****          ER1 = 4byte input data2 *****
; ***** Output: ER0 = 4byte output data *****
; *****          R1L = 0 no error, R1L<>0 error *****
; ***** Use:    R2 = 数据 1 的指数部分 *****
; *****          R3 = 数据 2 的指数部分 *****
; *****          R4 = 暂时保存 *****
; *****
; *****

FADD16:
    PUSH.W  R2          ; R2 入栈
    PUSH.W  R3          ; R3 入栈
    PUSH.W  R4          ; R4 入栈

; 确认开始的数据是否为零
    CMP.W   #0,R0        ; 确认小数部分的 16 位
    BNE     SNDCHK
    MOV.L   ER1,ER0      ; 直接输出第二个数据
    BRA     OKADD
SNDCHK    CMP.W   #0,R1    ; 确认小数部分的 16 位
    BNE     STADD        ; 跳转至加减运算的开始
    BRA     OKADD

; 开始浮点的加减运算
STADD     MOV.B   #0,R4H   ; 符号清零
          MOV.W   E0,R2    ; 复制数据 1 的指数部分
          MOV.W   E1,R3    ; 复制数据 2 的指数部分
          MOV.B   R2L,R4L  ; 将指数 1 保存在 R4L
          SUB.B   R3L,R2L  ; 比较指数
          BMI     SIFT1    ; 移动数据 1
          BNE     SIFT2    ; 移动数据 2
          MOV.B   R4L,R2L  ; 返回保存在 R4 中的指数 1
RETADD    BTST    #0,R2H   ; 确认数据 1 小数部分的 FLAG
          BNE     MIN1     ; 如果是负数的话
          BTST    #0,R3H   ; 确认数据 2 小数部分的 FLAG
          BNE     ADP1M2   ; 如果是负数的话

```

```

; 数据的符号一致的时候
ADP1P2  ADD.W   R1,R0           ; 小数 1 = 小数 1 + 小数 2
        BCC     SVADD           ; 没有进位的话
        ROTXR.W R0             ; 向右移位
        INC.B   R2L             ; 结果指数 + 1
        BVC     SVADD           ; 溢出?
        MOV.B   #1,R1L          ; 出错代码设置
        BRA     EXADD
SVADD   MOV.W   R2,E0
OKADD   MOV.B   #0,R1L          ; 清除错误信息
EXADD   POP.W   R4              ; R4 出栈
        POP.W   R3              ; R3 出栈
        POP.W   R2              ; R2 出栈
        RTS

; 对照数据 1
SIFT1   SHLR.W  R0              ; 小数部分 1 右移
        INC.B   R2L
        BNE     SIFT1           ; 0?
        MOV.B   R3L,R2L         ; 由于指数部分相同
        BRA     RETADD

; 对照数据 2
SIFT2   SHLR.W  R1              ; 小数部分 2 右移
        INC.B   R3L             ; 指数 2 + 1
        DEC.B   R2L
        BNE     SIFT2           ; 0?
        MOV.B   R4L,R2L         ; 返回保存在 R4 中的指数 1
        BRA     RETADD

MIN1    BTST     #0,R3H
        BEQ     ADP1P2          ; 数据 1 负数,数据 2 正数
        BRA     ADP1P2          ; 两个数值相加

; 数据的符号不相同的时候
ADM1P2  SUB.W   R0,R1           ; 小数 2 = 小数 2 - 小数 1
        MOV.W   R1,R0           ;
        BRA     CKADD
ADP1M2  SUB.W   R1,R0           ; 小数 1 = 小数 1 - 小数 2
CKADD   BEQ     ADDZERO         ; IF ZERO, SET ZERO
        BHI     ADPLS           ; IF HIGH or SAME, SKIP MINUS
        MOV.B   #1,R2H          ; SET MINUS FLAG
        NEG.W   R0              ; 取 2 的余数
SKMIN   BTST     #7,R0H          ; 确认 MSB 位
        BNE     SVADD           ; 计算结束
        SHLL.W  R0

```

续程序清单 8.1

```

        DEC. B   R2L                ; 结果指数 - 1
        BRA     SKMIN
ADPLS   MOV. B   # 0, R2H           SET PLUS FLAG
        BRA     SKMIN

ADDZERO MOV. W   # 0, E0            ; 结果指数清零
        MOV. B   # 0, R1L          ; 清除错误信息
        OPO. W   R4                ; R4 出栈
        POP. W   R3                ; R3 出栈
        POP. W   R2                ; R2 出栈
        RTS

```

实际运行的循环次数根据位对齐的移位量而变动,大致为插入几位的位对齐的 100 次( $3.5\mu\text{s}@32\text{MHz}$ )循环。

## 2. FSUB16(浮点的减法运算)

程序如程序清单 8.2 所示。减法运算的情况下将 ER1(参数 2)的符号反转,然后调用加法运算方法。

实际运行的循环次数根据位对齐的移位量而变动。大致为插入几位的位对齐的 112 次循环( $3.5\mu\text{s}@32\text{MHz}$ )。

程序清单 8.2 浮点的减法运算

```

; *****
; *****
; ***** 浮点的减法运算 *****
; ***** Input:  ER0 = 4byte input data1 *****
; *****          ER1 = 4byte input data2 *****
; ***** Output: ER0 = 4byte output data *****
; *****          R1L = 0 no error, R1L<>0 error *****
; ***** Use:    R2 = 暂时保存 *****
; *****
; *****

FSUB16:
        PUSH. W   R2                ; R2 入栈

; 反转第二个数据的符号,调用浮点的加法运算

        ADD. W    # 0, R1            ; 确认第二参数的小数部分
        BEQ       GOADD              ; if (2nd = 0)
        MOV. W    E1, R2             ; 复制指数部分 2
        ADD. B    # 0, R2H           ; 确认符号
        BEQ       MINSUB

```

续程序清单 8.2

```

MOV.B    #0,R2H          ; 变为正数
SVSUB    MOV.W    R2,E1    ; 返回指数部分
GOADD    JSR      @FADD16   ; 调用浮点的加法运算
          POP.W    R2       ; 复原 R2
          RTS
MINSUB   MOV.B    #1,R2H    ; 变为负数
          BRA      SVSUB

```

### 3. FDIV16(浮点的除法运算)

程序如程序清单 8.3 所示。除法运算的输入数据为 ER0(被除数)和 ER1(除数)。结果保存在 ER0 中,R1L 中保存出错信息。

程序清单 8.3 浮点的除法运算

```

; *****
; *****                                     *****
; ***** 浮点的除法运算                                     *****
; ***** Input:  ER0 = 4byte input data1                                     *****
; *****          ER1 = 4byte input data2                                     *****
; ***** Output: ER0 = 4byte output data                                     *****
; *****          R1L = 0 no error, R1L <> 0 error                             *****
; ***** Use:    R2 = 数据 1 的指数部分                                     *****
; *****          R3 = 数据 2 的指数部分                                     *****
; *****          R4 = 暂时保存                                             *****
; *****                                     *****
; *****
FDIV16:
    PUSH.W  R2          ; R2 入栈
    PUSH.W  R3          ; R3 入栈
    PUSH.W  R4          ; R4 入栈
    MOV.W   E0,R2       ; 保存指数 1
    MOV.W   E1,R3       ; 保存指数 2
    MOV.W   #H'0000,R4  ; 初始化指数部分的结果

; 指数部分(UNSIGNED 8BIT)的计算

    SUB.B   R3L,R2L     ; 指数 1(R2L) = 指数 1(R2L) - 指数 2(R3L)
    ADD.B   R2L,R4L     ; 指数的结果(R4L) = 指数的结果(R4L) + 指数 1(R2L)

; 符号的计算

    ADD.B   R3H,R2H     ; 符号 1(R2H) = 符号 1(R2H) + 符号 2(R3H)
    CMP.B   #1,R2H     ; 符号 1(R2H) - 1

```

续程序清单 8.3

	BNE	PLDIV	; 符号一致的情况下
	MOV.B	R2H,R4H	; 将得到的结果 1 写入 R4H
; 小数部分(16BIT)的除法运算			
PLDIV	MOV.W	R0,E0	; 将小数的 16 位复制到高位
	MOV.W	#0,R0	; 将低的 16 位置零
	SHLR.L	E0	; 右移一位
	DIVXU.W	R1,E0	; E0 = E0/R1
	BEQ	ERRDIV	; 如果除数为零则报错
	BTST	#7,ROH	; 确认 MSB
	BNE	SVDIV	; 确认是否要移动位数
	SHLL.W	R0	; 左移
	DEC.B	R4L	; 结果指数 - 1
SVDIV	MOV.W	R4,E0	; 复制结果指数
	MOV.B	#0,R1L	; 清除错误信息
	POP.W	R4	; R4 出栈
	POP.W	R3	; R3 出栈
	POP.W	R2	; R2 出栈
	RTS		
ERRDIV	MOV.B	#1,R1L	; 如果除数为 0 则报错
	POP.W	R4	; R4 出栈
	POP.W	R3	; R3 出栈
	POP.W	R2	; R2 出栈
	RTS		

首先计算指数部分。在除法运算当中,指数部分的计算是除法运算。举个例子,比如说  $1024 \div 254 = 4$  的计算,指数部分分别是 10,8,2,则除法运算为  $10 - 8 = 2$ 。

其次是符号的处理。符号一致的情况下,结果为正,不一致的情况下结果为负。

再次是小数部分的除法运算。与 CPU 的除法指令(32 位  $\div$  16 位)相符合,做以下处理。

(1) 被除数是最大值 1.99997(FFFFh),除数是最小值 1.0(8000h)的情况下,  $7FFF8000h \div 8000h = FFFFh(1.99997)$ ,由于最高位是 1,所以不需要位对齐。

(2) 被除数是最小值 1.0(8000h),除数是最大值 1.99997(FFFFh)的情况下,  $40000000h \div FFFFh = 4000h(0.5)$ ,由于最高位为 0,所以左移 1 位为 8000h(1.0)。

像这样,小数部分的最高位是 1 的情况下,在除法运算之后的位对齐,也只

需要 1 位就能实现。另外,只有除数是 0 的情况下除法运算才会产生错误。

实际运行的循环次数是固定的,大致为 82 次循环( $2.5625\mu\text{s}@32\text{MHz}$ )。

#### 4. FMUL16 (浮点的乘法运算)

程序如程序清单 8.4 所示。乘法运算的输入数据为 ER0(被乘数)和 ER1(乘数)。结果保存在 ER0 中。

首先计算指数部分。乘法运算的指数部分的计算是加法运算。在加法运算的结果之上加 1。这个原因与小数部分的计算有关。CPU 的乘法运算指令(16 位 $\times$ 16 位=32 位),在被乘数是最大值 1.99997(FFFFh)、乘数也是最大值 1.99997(FFFFh)的情况下,FFFFh $\times$ FFFFh=FFFE0001h。

小数部分的乘法运算是解的 32 位中取高位的 16 位。因此为 FFFEh(19999.4)。原本是  $1.99997 \times 1.99997 = 3.99988$  的,因此需要将小数部分的乘法运算结果右移一位。因此,计算指数部分的时候需要加上 1。

程序清单 8.4 浮点的乘法运算

```

; *****
; *****
; ***** 浮点的乘法运算 *****
; ***** Input: ER0 = 4byte input data1 *****
; ***** ER1 = 4byte input data2 *****
; ***** Output: ER0 = 4byte output data *****
; ***** Use: R2 = 数据 1 的指数部分 *****
; ***** R3 = 数据 2 的指数部分 *****
; ***** R4 = 暂时保存 *****
; *****
; *****

FMUL16:
    PUSH.W R2          ; R2 入栈
    PUSH.W R3          ; R3 入栈
    PUSH.W R4          ; R4 入栈
    MOV.W E0,R2        ; 保存指数 1
    MOV.W E1,R3        ; 保存指数 2
    MOV.W #H'0001,R4   ; 初始化指数部分的结果

; 指数部分(UNSIGNED 8BIT)的计算

    ADD.B R3L,R2L      ; 指数 1(R2L) = 指数 1(R2L) + 指数 2(R3L)
    ADD.B R2L,R4L      ; 结果指数(R4L) = 结果指数(R4L) + 指数 1(R2L)

; 符号的计算

```



续程序清单 8.4

	ADD. B	R3H, R2H	; 符号 1(R2H) = 符号 1(R2H) + 符号 2(R3H)
	CMP. B	# 1, R2H	; 符号 1(R2H) - 1;
	BNE	PLMUL	; 符号一致的话
	MOV. B	R2H, R4H	; 将结果 1 写入 R4H
; 小数部分(16BIT)的乘法运算			
PLMUL	MULXU. W	R1, ER0	; 乘法运算结果(ER0) = 小数 1(R0) * 小数 2(R1)
	ADD. W	# 0, E0	; 为了确认乘法运算的结果
	BMI	SVMUL	; 是否要移动位数?
	BEQ	EXFMUL	; 运算结果是否为零
	SHLL. L	ER0	; 左移
	DEC. B	R4L	; 指数的运算结果 - 1
SVMUL	MOV. W	E0, R0	; 将高位的 16 位复制到低位的 16 位
	MOV. W	R4, E0	; 复制指数部分的结果
EXFMUL	POP. W	R4	; R4 出栈
	POP. W	R3	; R3 出栈
	POP. W	R2	; R2 出栈
	RTS		

实际运行的循环次数是固定的, 大致为 80 次循环(2.5 $\mu$ s@MHz)。

### 5. FTOI16 (从浮点到整型的变换)

程序如程序清单 8.5 所示。输入数据是 ER0。R1H 是指定的输出数据类型。0 是带符号的整数, 1 是不带符号的整数。变换结果保存在 R0 中。R1H 存储的是错误信息。在输入数据超出整数范围时, 显示错误信息(R1H=1)。

变换的基本方法是, 如果指数部分是 15 的话, 直接将小数部分变成整数。比如说, 在 1000.1 的情况下, 将 1000.1 变换成浮点的话, 指数部分是 9, 小数部分是 1.9533203125(FA06h)。如果指数部分是 15, 则将小数部分右移 6 位, FA06h 变成了 3E8h, 在十进制中相当于 1000。

带符号的 16 位整数的范围是 -32768~32767, 不带符号的 16 位整数的范围是 0~65535。需要确认输出数据类型按照指定的数据范围输出。

实际运行的循环次数根据位对齐的移位量而变动。大致为 52 次循环(1.6 $\mu$ s@32MHz)~132 次循环(4.125 $\mu$ s@32Mhz)。

程序清单 8.5 从浮点到整型的变换

```

; *****
; *****                                     *****
; ***** 从浮点到整型的变换                                     *****
; ***** Input:  ERO = 4byte float                                     *****
; *****          R1H = 0 signed R1L<>0 unsigned                                     *****
; ***** Output: R0 = 2byte integer                                     *****
; *****          R1L = 0 no error, R1L<>0 error                                     *****
; ***** Use:    R2 = 指数部分                                     *****
; *****          R3 = 暂时保存                                     *****
; *****                                     *****
; *****
; *****

FTOI16:
    PUSH.W  R2                ; R2 入栈
    PUSH.W  R3                ; R3 入栈
    ADD.W   #0,R0              ; 确认小数部分
    BEQ     EXFTOI             ; IF ZERO,EXIT FTOI
    MOV.W   E0,R2              ; 复制指数部分
    MOV.B   #15,R3L            ; 如果指数部分是 15 的话,直接将小数变成整数
    SUB.B   R3L,R2L            ; 指数(R1L) = 指数(R1L) - 15
    BMI     SIFTR              ; 小数部分右移
    BNE     SIFTL              ; 小数部分左移

RETTOI  ADD.B   #0,R1H          ; SIGNED OR UNSIGNED
        BNE     EXFTOI          ; IF UNSIGNED,SKIP MSB CHECK
        BTST    #7,R0H          ; 确认小数部分的最高位
        BNE     SIGNOV          ; SIGNED 16bit OVER
        BTST    #0,R2H          ; 确认浮点的符号
        BEQ     EXFTOI          ; IF PLUS,SKIP NEG
        NEG.W   R0              ; 将小数变换为 2 的余数
EXFTOI  MOV.B   #0,R1L          ; 清除错误信息
        POP.W   R3              ; R3 出栈
        POP.W   R2              ; R2 出栈
        RTS

SIFTR   SHLR.W  R0              ; 小数右移
        BEQ     TOIZEWRO        ; 设零
        INC.B   R2L              ; 移位计数器 + 1
        BEN     SIFTR           ; 移位结束?
        BRA     RETTOI

; 指数部分比 15 大的情况下超过整数值
SIFTL:
        ADD.B   #0,R1H          ; SIGNED OR UNSIGNED
        BNE     TOIFULL         ; IF UNSIGNED,SET H'FFFF
        BRA     SIGNOV

; 置零

```

续程序清单 8.5

```

TOIZERO MOV.W  #0,R0
        BRA     EXFTOI

; Signed 16bit Over
SIGNOV  BTST    #0,R2H                ; 确认浮点的符号
        BEQ     SET7FFF
SET8000 MOV.W  #H'8000,R0            ; 负的最大值
        BRA     IOVER
SET7FFF MOV.W  #H'7FFF,R0            ; 正的最大值
IOVER   MOV.B   #1,R1L                ; 溢出错误
        POP.W   R3                    ; R3 出栈
        POP.W   R2                    ; R2 出栈
        RTS

; 超过 16bit 整数
TOIFULL ADD.B   #0,R1H                ; SIGNED OR UNSIGNED
        BEQ     SIGNOV                ; IF SIGNED,BRANCH SIGNED OVER
        MOV.W   #H'FFFF,R0            ; UNSIGNED 最大值
        BRA     IOVER

```

## 6. ITOF16(从整型到浮点的变换)

程序如程序清单 8.6 所示。输入数据是 R0。R1H 是指定的输入数据类型。

程序清单 8.6 从整型到浮点的变换

```

; *****
; *****
; ***** 从整型到浮点的变换 *****
; ***** Input:  R0 = 2byte integer *****
; *****          R1H = 0 signed R1H<>0 unsigned *****
; ***** Output: ER0 = 4byte float *****
; *****          R1L = 0 no error, R1L<>0 error *****
; ***** Use:    R2 = 指数部分 *****
; *****
; *****
ITOF16:
        PUSH.W  R2                    ; R2 入栈
        MOV.W   #15,R2                ; 初始化指数部分
        ADD.B   #0,R1H                ; CHECK SIGNED OR UNSIGNED
        BNE     UNSIGN
        ADD.W   #0,R0                ; 确认整数的内部
        BEQ     TOFZERO                ; 为零?
        BPL     SFITOF                ; 小数部分需要移位

```

	NEG. W	R0	; 计算 2 的余数
	MOV. B	#1, R2H	; 设置负标记
SFITOF	STSF	#7, R0H	; 确认 MSB
	BNE	SVITOF	; 移位结束?
	SHLL. W	R0	; 小数左移
	DEC. B	R2L	; 指数 - 1
	BRA	SFITOF	
SVITOF	MOV. W	R2, E0	; 复制指数部分
	MOV. B	#0, R1L	; 清除错误信息
	POP. W	R2	; R2 出栈
	RTS		
UNSIGN	ADD. W	#0, R0	; 确认整数的内部
	BEQ	TOFZERO	
	BRA	SFITOF	
TOFZERO	MOV. W	#0, R2	; 清除指数部分
	BRA	SVITOF	

0 表示带符号的整数。1 表示不带符号的整数。变换的结果保存在 ER0 中。虽然设定 R1H 保存错误信息,但是不可能出错。

变换的基本方法是,将整数部分复制到小数部分,指数部分设为 15。然后,将小数部分移位至使其最高位为 1,指数部分自减。因为小数部分是 unsigned,所以带符号的整数是负数的时候取 2 的余数作为小数部分的数据。

实际运行的循环次数根据位对齐的移位量而变动。循环次数大致为 44 次 ( $1.375\mu\text{s}@32\text{MHz}$ )~108 次 ( $3.375\mu\text{s}@32\text{MHz}$ )。

### 7. ITOF32(从长整型到浮点的变换)

程序如程序清单 8.7 所示。输入数据是 ER0(带符号的整数)。变换结果保存在 ER0 中。虽然设定 R1H 保存错误信息,但是这里也不会出现错误。

变换的基本方法是,将 32 位的整数全部考虑成 32 位的小数。这样的话指数部分就是 31 了,然后,将小数部分移位至 32 位的小数的最高位为 1 为止,指数部分自减。最高位变成 1 之后,将最高位的 16 位复制到 16 位的小数部分。因为小数部分是 UNSIGNED 的缘故,负数的时候取 2 的余数作为小数部分的数据。

实际运行的循环次数根据位对齐的移位量而变动。大致需要循环 172 次 ( $5.375\mu\text{s}@32\text{MHz}$ )~296 次 ( $9.25\mu\text{s}@32\text{MHz}$ )。

程序清单 8.7 从 32 位整型到浮点的变换

```

; *****
; *****                                     *****
; ***** 从 32 位整型到浮点的变换                                     *****
; ***** Input:  ER0 = 4byte integer                                     *****
; ***** Output: ER0 = 4byte float                                     *****
; *****          R1L = 0 no error, R1L <> 0 error                     *****
; ***** Use:     R2 = 指数部分                                         *****
; *****                                     *****
; *****                                     *****
; *****                                     *****

ITOF32:
    PUSH.W  R2                      ; R2 入栈
    MOV.W   #0,E1
    MOV.W   #31,R2                  ; 初始化指数部分
    ADD.L   #0,ER0                  ; 确认整数的内部
    BEQ     ITOF32Z                ; 为零?
    RPL     SFITF32                ; 小数部分需要移位
    NEG.L   ER0                    ; 计算 2 的余数
    MOV.B   #1,R2H                 ; 设置负标记
SFITF32 ADD.W  E1,E0                ; 确认 MSB
    BMI     SVITF32                ; 移位结束?
    SHLL.L  ER0                    ; 小数左移
    DEC.B   R2L                    ; 指数 - 1
    BRA     SFITF32
SVITF32 MOV.W  E0,R0                ; 复制小数部分
    MOV.W   R2,E0                  ; 复制指数部分
    MOV.B   #0,R1L                 ; 清除错误信息
    POP.W   R2                     ; R2 出栈
    RTS

ITOF32Z MOV.W   #0,R2              ; 清除指数部分
    BRA     SVITF32

```

## 8. FSQRT16(浮点的平方根)

程序如程序清单 8.8 所示。平方根的近似值运算用牛顿法。

输入数据是 ER0。R1H 是指定的平方根近似计算的环路次数。结果保存在 ER0 中。虽然 R1H 被用来设计成存储错误信息,但是不可能发生错误。

首先计算初始解  $\alpha_0$ 。将初始解的指数部分除以 2。输入数据为 X, 初始解为  $\alpha_0$ , 则牛顿法实现的近似计算方法为如下所示

$$\alpha_1 = \frac{\alpha_0 + \frac{X}{\alpha_0}}{2}$$

程序清单 8.8 浮点的平方根

```

; *****
; *****
; ***** 浮点的平方根 *****
; ***** Input: ER0 = 4byte float(X) *****
; ***** R1H = 平方根的环路次数 *****
; ***** Output: ER0 = 4byte float( $\sqrt{X}$ ) *****
; ***** R1L = 0 no error, R1L < > 0 error *****
; ***** Use: ER2 =  $\alpha$  *****
; ***** ER3 = X *****
; ***** R4H = 平方根的环路次数 *****
; *****
; *****
FSQRT16
; 计算初始解
    PUSH.L ER2          ; ER2 入栈
    PUSH.L ER3          ; ER3 入栈
    PUSH.W R4           ; R4 入栈
    MOV.B R1H,R4H       ; 保存平方根的环路次数
    MOV.W #H'8000,E2    ; 小数部分 1.000
    MOV.B #0,R2H        ; 加号
    MOV.W E0,R3         ; 复制输入的指数部分
    MOV.W R0,E3         ; 复制输入的小数部分
    MOV.B R3L,R2L       ; 将 X 的指数部分复制到  $\alpha$  的指数部分
    BMI MISQRT          ; 确认输入指数部分的符号
    SHLR.B R2L          ; 指数部分/2

; X/ $\alpha$  的计算
LPSQRT: MOV.W E3,R0      ; 复制输入的指数部分
        MOV.W R3,E0      ; 复制输入的小数部分
        MOV.W E2,R1      ;  $\alpha$  的小数部分
        MOV.W R2,E1      ;  $\alpha$  的指数部分
        JSR @FDIV16      ; X/ $\alpha$ 

;  $\alpha + (X/\alpha)$  的计算
        MOV.W E2,R1      ;  $\alpha$  的小数部分
        MOV.W R2,E1      ;  $\alpha$  的指数部分
        JSR @FADD16      ;  $\alpha + (X/\alpha)$ 

;  $(\alpha + (X/\alpha))/2$  的计算
        MOV.W E0,R2      ; 新  $\alpha$  的指数部分
        MOV.W R0,E2      ; 新  $\alpha$  的小数部分
        DEC.B R2L        ;  $(\alpha + (X/\alpha))/2$ 
        DEC.B R4H        ; COUNTER - 1
        BNE LPSQRT       ; 处理完毕?
        MOV.W R2,E0      ; 最终的  $\alpha$ 
        MOV.B #0,R1L     ; 清除错误信息
        POP.W R4         ; R4 出栈

```

续程序清单 8.8

```

        POP.L   ER3                ; ER3 出栈
        POP.L   ER2                ; ER2 出栈
        RTS

; 在指数为负数的时候计算初始解
MISQRT:
        SHLR.B   R2L                ; 指数/2
        OR.B     #H'80,R2L          ; 指数部分的负标记位设置
        BRA      LPSQRT

```

重复进行这个近似计算,可以求得更加近似的值。比如说,将 $\sqrt{3}$ 进行 4 次环路计算就变为 1.732025,而其真实值是 1.732050808,进行 4 次环路计算得到 16 位的精度。

实际运行的循环次数根据环路的次数改变而不同。4 次环路运算大致需要 832 次循环(26 $\mu$ s@32MHz)。

### 9. FTOF16(从单精度浮点到 16 位浮点的变换)

程序如程序清单 8.9 所示。执行的是从 IEEE 标准下的单精度浮点向 16 位浮点的变换。

程序清单 8.9 从单精度浮点到 16 位浮点的变换

```

; *****
; *****
; ***** 从单精度浮点到 16 位浮点的变换 *****
; ***** Input:  ER0 = 单精度浮点(4 字节) *****
; ***** Output: ER0 = 16 位浮点(4 字节) *****
; *****      R1L = 0 no error, R1L<>0 error *****
; ***** Use:    R2 = 指数部分 *****
; *****      R3 = 小数部分 *****
; ***** *****
; ***** *****
PTOF16:
        PUSH.W   R2
        PUSH.W   R3

; 参数为零?
        ADD.L     #0,ER0
        BEQ       EXFTOF

; 小数部分的变换
        MOV.W     E0,R2                ; float 的高位 16 位

```

```

MOV.W  E0,R3          ; float 的低位 16 位
MOV.B  R3H,R3L        ; 将小数部分 23 位转化成 16 位
MOV.B  R2L,R3H
OR.B   #H'80,R3H      ; 将小数部分的 MSB 设 1

; 指数部分的变换
SHLL.W  R2             ; 左移
MOV.B   #127,R2L
SUB.B   R2L,R2H        ; 指数部 - 127
MOV.B   R2H,R2L

; 符号的变换
ADD.W   #0,E0
BMI     FTOFMI          ; 负数?
MOV.B   #0,R2H          ; 正数

; 生成 16 位浮点
FTOFSV  MOV.W  R3,R0    ; 小数部分
        MOV.W  R2,E0    ; 指数部分
EXFTOF  MOV.B  #0,R1L    ; 清除错误信息
        POP.W  R3
        POP.W  R2
        RTS

FTOFMI  MOV.B  #1,R2H
        BRA    FTOFSV

```

输入输出寄存器为 ER0。虽然设定 R1H 保存错误信息,但是不可能出错。实际运行的循环次数是固定的,大致为 48 次循环( $1.5\mu\text{s}@32\text{MHz}$ )。

#### 10. FTOF24(从 16 位浮点到单精度浮点的变换)

程序如程序清单 8.10 所示。执行的是从 16 位浮点到 IEEE 标准下的单精度浮点的变换。

输入输出寄存器位 ER0。虽然设定 R1H 保存错误信息,但是不可能出错。实际运行的循环次数是固定的,大致为 58 次循环( $1.8\mu\text{s}@32\text{MHz}$ )。



程序清单 8.10 从 16 位浮点到单精度浮点的变换

```

; *****
; *****
; ***** 从 16 位浮点到单精度浮点的变换 *****
; ***** Input:  E0 = 16 位浮点型(4 字节) *****
; ***** Output: E0 = 单精度浮点型(4 字节) *****
; *****      R1L = 0 no error, R1L <> 0 error *****
; ***** Use:    R2 = 指数部分 *****
; *****      R3 = 小数部分 *****
; *****      R4 = 暂时保存 *****
; *****
; *****
FTOF24:
    PUSH.W  R2
    PUSH.W  R3
    PUSH.W  R4
; 参数为零?
    ADD.L   #0,E0
    BEQ     EXFTOF24

; 小数部分的变换
    MOV.W   E0,R2          ; float 高位的 16 位
    MOV.W   R0,R3          ; float 地位的 16 位
    MOV.B   R3H,R4L        ; 将小数部分的 16 位转化成 23 位
    MOV.B   R3L,R3H
    BCLR    #7,R4L         ; 小数部分没有第 23 位
    MOV.B   #0,R4H         ; 指数部分先置零
    MOV.B   #0,R3L         ; 最低 8 位为 0

; 指数部分的变换
    MOV.B   #127,R2H
    ADD.B   R2L,R2H        ; 指数部分 + 127
    SHLR.W  R2             ; 右移
    AND.W   #H'7F80,R2     ; 指数以外掩码
    OR.W    R4,R2          ; 生成指数部分和小数部分的高位部分

; 变换符号
    MOV.W   E0,R4
    BTST    #0,R4H         ; 符号
    BNE     FTOF24MI       ; 负数?
    BCLR    #7,R2H         ; 正数

; 生成 16 位浮点
FTOF24SV:
    MOV.W   R3,R0          ; 小数部分(低位的 16 位)
    MOV.W   R2,E0          ; 符号(1 位) + 指数部分(8 位) + 小数部分(高位的 7 位)
EXFTOF24:
    MOV.B   #0,R1L        ; 清除错误信息

```

```

        POP.W   R4
        POP.W   R3
        POP.W   R2
        RTS

FTOF24MI:
        BSET    #7,R2H
        BRA     FTOF24SV

```

### 8.3 PID 控制器

程序如程序清单 8.11 所示。在 PID 控制器中使用的是在 8.2 节中介绍的 16 位浮点的计算方法。请注意,在 PID 控制器中寄存器的名字改变了。

程序清单 8.11 PID 控制器

```

; *****
; *****
; ***** Title: PID Equation Program for H8S/2367. *****
; *****
; ***** Func:将需要输入的数据加入 PID(比例,积分,微分)的系数之后再输出 *****
; *****
; ***** Source File Name : PID.SRC *****
; *****
; ***** Object File Name : PID.OBJ *****
; *****
; *****
; .PROGRAM PID ; Program Name
; .CPU 2000A;24 ; CPU is H8S/2000 Advanced
; .SECTION ROM, CODE, ALIGN = 2 ; ROM Area Section

; 需要调用的外部子程序
; .IMPORT FADD16 ; 浮点的加法运算
; .IMPORT FSUB16 ; 浮点的减法运算
; .IMPORT FDIV16 ; 浮点的除法运算
; .IMPORT FMUL16 ; 浮点的乘法运算

; 需要调用的外部变量(全局变量是用 C 语言声明的)
; .IMPORT _hgDrvLim ; 驱动极限变量

; 外部定义的子程序
; .EXPORT PIDEQN

```

; 改变寄存器名

```

DATA0 .REG      (ER0)      ; 计算数据 0
DATA1 .REG      (ER1)      ; 计算数据 1
RA0 .REG        (ER2)      ; 输入输出数据
RA1 .REG        (ER3)      ; 通用
HISTPTR .REG    (ER4)      ; 计算域
COEFPTR .REG    (ER5)      ; 系数地址

```

PIDEQN;

```

PUSH.W  R6              ; R6 入栈
MOV.L   @HISTPTR, DATA1 ; err(n-1)
MOV.L   RA0, DATA0      ; err(n)
JSR     @FSUB16          ; r1 = err(n) - err(n-1)
MOV.L   ER0, RA1         ; r1 更新
MOV.L   RA0, @HISTPTR    ; err(n-1) = err(n)
MOV.L   RA0, DATA0      ; err(n)
MOV.L   @COEFPTR, DATA1 ; kp
JSR     @FMUL16          ; gp = kp * err(n)
MOV.L   DATA0, RA0      ; r0 更新
MOV.L   @(8:16, COEFPTR), DATA0 ; kd
MOV.L   RA1, DATA1      ; err(n) - err(n-1)
JSR     @FMUL16          ; gd = kd * (err(n) - err(n-1))
MOV.L   DATA0, RA1      ; r1 更新
MOV.L   RA0, DATA0      ; gp
MOV.L   RA1, DATA1      ; gd
JSR     @FADD16          ; r0 = gp + gd
MOV.L   DATA0, RA0      ; r0 更新
MOV.L   @(4:16, HISTPTR), DATA0 ; Icharge(n-1)
MOV.L   @(4:16, HISTPTR), DATA1 ; err(n)
JSR     @FADD16          ; r1 = Icharge(n-1) + err(n)
MOV.L   DATA0, RA1      ; r1 更新

```

; 加上积分项的极限

```

MOV.L   @(4:16, COEFPTR), DATA1 ; 读取 ki
BEQ     NOLIMIT          ; 如果 ki = 0, 则没有极限
MOV.L   @_hgDrvLim, DATA0      ; 读取驱动极限
JSR     @FDIV16            ; 计算 DrvLim/ki
PUSH.L  DATA0             ; DrvLim/ki 入栈
MOV.W   E3, R6            ; 复制 r1 的指数部分
ADD.B   #0, R6H           ; 确认符号
BEN     MINLIM            ; 负数?

```

; 积分项为正

```

MOV.L   DATA0, DATA1      ; DrvLim/ki
MOV.L   RA1, DATA0        ; r1
JSR     @FSUB16            ; r1 - DrvLim/ki
MOV.W   E0, R6            ; 复制指数部分

```

```

        POP.L    DATA1                ; DrvLim/ki 出栈
        ADD.B    #0,R6H                ; 确认符号
        BNE      NOLIMIT               ; 没有极限
        MOV.L    DATA1,RA1            ; r1 = DrvLim/ki
        BRA      NOLIMIT
; 积分项为负
MINLIM  MOV.L    DATA0,DATA1          ; DrvLim/ki
        MOV.L    RA1,DARA0             ; r1
        JSR      @FADD16               ; r1 + DrvLim/Ki
        MOV.W    E0,R6                ; 复制指数部分
        POP.L    DATA1               ; DrvLim/ki 出栈
        ADD.B    #0,R6H                ; 确认符号
        BEQ      NOLIMIT               ; 没有极限
        MOV.W    E1,R6                ; DrvLim/ki 的符号置为负数
        MOV.B    #1,R6H                ;
        MOV.W    R6,E1
        MOV.L    DATA1,RA1            ; r1 = - DrvLim/ki

NOLIMIT MOV.L    RA1,@(4:16,HISTPTR)    ; Icharge(n) = Icharge(n-1) + err(n)
        MOV.L    RA1,DATA0             ; r1
        MOV.L    @(4:16,COEFPTR),DATA1 ; ki
        JSR      @FMUL16               ; gi = ki * Icharge(n)
        MOV.L    DATA0,RA1            ; r1 更新
        MOV.L    RAO,DATA1             ; r0
        JSR      @FADD16               ; r0 = gp + gi + gd
        MOV.L    DATA0,RA0            ; r0 更新
        ADD.L    #0,RA1                ; if(r1 = 0)
        BNE      EXITPID               ; 结束
        MOV.L    RA1,@(4:16,HISTPTR)    ; HistPtr(1) = r1
EXITPID POP.W    R6                    ; R6 出栈
        RTS

```

### 8.3.1 寄存器的使用方法

#### 1. DATA0(ER0)

通用的计算寄存器。在浮点的各种运算方法中,DATA0(ER0)作为保存参数和计算结果的寄存器。

#### 2. DATA1(ER1)

通用的计算寄存器。在浮点的各种运算方法中,DATA1(ER1)作为保存参数的寄存器。



### 3. RA0(ER2)

PID 控制器中的输入输出寄存器。输入数据位伺服控制器的位置或者是速度偏差数据,输出数据则为所保存的 PID 控制器的计算结果。

### 4. RA1(ER3)

通用的计算寄存器。用于暂时保存计算结果的寄存器。

### 5. HISTPTR(ER4)

作为在 PID 控制器中使用的计算范围的指针来使用。

HISTPTR[0]:保存一个采样前的偏差数据。

HISTPTR[1]:保存积分项。

### 6. COEFPTR(ER5)

作为在 PID 控制器中使用的系数范围的指针来使用。

COEFPTR[0]:保存比例增益  $K_p$ 。

COEFPTR[1]:保存积分增益  $K_i$ 。

COEFPTR[2]:保存微分增益  $K_d$ 。

## 8.3.2 计算顺序和运行时间

计算顺序如下所示:

- (1) 偏差数据的微分计算(减法运算处理  $3.5\mu\text{s}$ )。
- (2) 乘以比例增益  $K_p$ (乘积运算处理  $2.5\mu\text{s}$ )。
- (3) 乘以微分增益  $K_d$ (乘法运算处理  $2.5\mu\text{s}$ )。
- (4) 将比例乘法运算结果和微分乘法运算结果相加(加法运算处理  $3.125\mu\text{s}$ )。
- (5) 将积分的保存值和偏差数据相加(加法运算处理  $3.125\mu\text{s}$ )。
- (6) 积分项极限的处理(除法运算处理  $2.5625\mu\text{s}$  + 加减法运算处理  $3.5\mu\text{s}$ )。
- (7) 乘以积分增益  $K_i$ (乘法运算处理  $2.5\mu\text{s}$ )。
- (8) 在(比例乘法运算结果 + 微分乘法运算结果)中加上积分乘法运算的结果( $3.125\mu\text{s}$ )。

整个浮点的计算时间需要  $26.43\mu\text{s}$ ,加上其他的处理的话,PID 控制器的运

行时间为  $34\mu\text{s}$ 。

### 8.3.3 在积分项中设置极限

在积分项中设置极限。比如在进行位置控制的时候,电机碰撞到限位器等时,位置偏差并不为零。在这种情况下,由于积分项中蓄积了非常大的值,如果是浮点的话,可能会产生溢出现象。

另外,即便没有产生溢出现象,由于积分项蓄积了非常大的值,想要使其限位器的反方向移动的时候,在积分项蓄积的保存值消除之前,电机的转矩都不能改变。这会导致电机控制中产生很大的迟延,从而引发低频振荡。因此,需要将PID控制器的积分项的极限设置为使其与转矩极限(驱动极限)趋向于相同的值。

比如说,转矩极限(驱动极限)=100,积分增益  $K_i=0.01$  的情况下,积分项中蓄积的值为  $100/0.01=10000$ ,这个值和转矩极限(驱动极限)相同。

## 8.4 位置/速度环控制器

程序如程序清单8.12所示。在位置/速度环控制器中使用的是在8.2节中介绍的16位精度浮点的计算方法。在位置/速度环控制器当中寄存器的名字也改变了,请注意。

程序清单 8.12 位置/速度环控制器

```

; *****
; *****
; ***** Title: Multi PI Equation Program for H8S/2367. *****
; *****
; ***** Func: 位置/速度环控制器 *****
; *****
; ***** Source File Name : PPI.SRC *****
; ***** Object File Name : PPI.OBJ *****
; *****
; *****
. PROGRAM PPIEQN ; Program Name
. CPU 2000A;24 ; CPU is H8S/2000 Advanced
. SECTION P, CODE, ALIGN = 2 ; ROM Area Section

; 外部调用子程序
. IMPORT FADD16 ; 浮点的加法运算

```

续程序清单 8.12

```

        .IMPORT  FSUB16          ; 浮点的减法运算
        .IMPORT  FDIV16         ; 浮点的除法运算
        .IMPORT  FMUL16         ; 浮点的乘法运算

; 需要调用的外部变量(全局变量是用C语言声明的)
        .IMPORT  _hgDrvLim      ; 驱动极限变量

; 外部定义的子程序
        .EXPORT   PPIEQN

; 改变寄存器名
DATA0   .REG      (ER0)        ; 计算数据 0
DATA1   .REG      (ER1)        ; 计算数据 1
RA0     .REG      (ER2)        ; 输入输出数据...指令值和现在值的偏差
RA1     .REG      (ER3)        ; 输入数据...现在的速度(速度控制的时候为 0)
HISTPTR .REG      (ER4)        ; 计算范围
COEFPTR .REG      (ER5)        ; 系数地址
TEMP    .REG      (ER6)        ; 暂时

; HISTPTR 是所指的系数
; HISTPTR + 0...Icharge(n)

; COEFPTR 是所指系数的构成
; COEFPTR + 0...Kpp(速度控制的时候设为 1)
; COEFPTR + 4...Kvp
; COEFPTR + 8...Kvi

; *****
; *          位置/速度环控制器的函数          *
; *****
PPIEQN: PUSH.L  TEMP            ; ER6 出栈
; - - - = 多重控制的前段部分 (RA1 作为前段部分的参数来使用)
        MOV.L   RA0, DATA0      ; err(n)
        MOV.L   @(0, 16, COEFPTR), DATA1 ; kpp
        JSR     @FMUL16          ; gpp = kpp * err(n)
        MOV.L   RA1, DATA1      ; vel(n)
        JSR     @FSUB16          ; verr(n) = gpp - vel(n)
        MOV.L   DATA0, TEMP     ; verr(n)保存

; - - - = 多重控制器的后段部分 (RA1 作为后段部分的工作范围来使用)
        MOV.L   @(4, 16, COEFPTR), DATA1 ; kvp
        JSR     @FMUL16          ; gvp = kvp * verr(n)
        MOV.L   DATA0, RA0      ; r0 更新
        MOV.L   @HISTPTR, DATA0 ; Icharge(n-1)
        MOV.L   TEMP, DATA1     ; verr(n)
        JSR     @FADD16          ; r1 = Icharge(n-1) + verr(n)
        MOV.L   DATA0, RA1     ; r1 更新

```

```

; 在积分项中加上极限
MOV.L    @(8:16,COEFPTR),DATA1    ; 读取 kvi
BEQ      NOLIMIT                    ; 如果 kvi = 0 的话,则没有极限
MOV.L    @_hgDrvLim,DATA0           ; 读取驱动极限
JSR      @FDIV16                     ; 计算 DrvLim/ki
PUSH.L   DATA0                     ; DrvLim/ki 入栈

MOV.W    E3,R6                      ; 复制 r1 的指数部分
ADD.B    #0,R6H                     ; 确认符号
BNE      MINLIM                      ; 负数?

; 积分项为正数
PLSLIM   MOV.L    DATA0,DATA1      ; DrvLim/ki
MOV.L    RA1,DATA0                  ; r1
JSR      @FSUB16                     ; r1 - DrvLim/ki
MOV.W    E0,R6                      ; 复制指数部分
POP.L    DATA1                     ; Drvlim/ki 出栈
ADD.B    #0,R6H                     ; 确认符号
BNE      NOLIMIT                    ; 没有极限
MOV.L    DATA1,RA1                 ; r1 = DrvLim/ki
BRA      NOLIMIT

; 积分项为负数
MINLIM   MOV.L    DATA0,DATA1      ; DrvLim/ki
MOV.L    RA1,DATA0                  ; r1
JSR      @FADD16                     ; r1 + DrvLim/ki
MOV.W    E0,R6                      ; 复制指数部分
POP.L    DATA1                     ; Drvlim/ki 出栈
ADD.B    #0,R6H                     ; 确认符号
BEQ      NOLIMIT                    ; 没有极限
MOV.W    E1,R6                      ; 将 DrvLim/ki 的符号设为负数
MOV.B    #1,R6H                     ;
MOV.W    R6,E1                      ;
MOV.L    DATA1,RA1                 ; r1 = - DrvLim/ki

NOLIMIT
MOV.L    RA1,@HISTPTR               ; Icharge(n) = Icharge(n-1) + verr(n)
MOV.L    @(8:16,COEFPTR),DATA0     ; kvi
MOV.L    RA1,DATA1                  ; Icharge(n)
JSR      @FMUL16                     ; gvi = kvi * Icharge(n)
MOV.L    DATA0,RA1                 ; r1 更新
MOV.L    RA0,DATA1                  ; r0
JSR      @FADD16                     ; r0 = gvp + gvi
MOV.L    DAT0,RA0                   ; r0 更新
ADD.L    #0,RA1                     ; if(r1 = 0)
BNE      EXITPID                    ; 结束
MOV.L    RA1,@HISTPTR               ; Icharge(n) = r1

EXITPID  POP.L    TEMP               ; ER6 出栈
RTE

```



### 8.4.1 寄存器的使用方法

#### 1. DATA0(ER0)

通用的计算寄存器。在浮点的各种运算方法中,DATA0(ER0)作为保存参数和计算结果的寄存器。

#### 2. DATA1(ER1)

通用的计算寄存器。在浮点的各种运算方法中,DATA1(ER1)作为保存参数的寄存器。

#### 3. RA0(ER2)

位置/速度环控制器中的输入输出寄存器。输入数据为伺服控制器的位置或者是速度偏差数据,输出数据则为所保存的位置/速度环控制器的计算结果。

#### 4. RA1(ER3)

位置/速度环控制器设有多重反馈,这个寄存器用于输入现在的速度的寄存器。

#### 5. HISTPTR(ER4)

作为在位置/速度环控制器中使用的计算范围的指针来使用。

HISTPTR[0]:保存速度积分项。

#### 6. COEFPTR(ER5)

作为在位置/速度环路控制器中使用的系数范围的指针来使用。

COEFPTR[0]:保存位置比例增益  $K_{pp}$ 。

COEFPTR[1]:保存速度比例增益  $K_{pv}$ 。

COEFPTR[2]:保存速度积分增益  $K_{iv}$ 。

#### 7. TEMP(ER6)

通用的计算寄存器。用于暂时保存计算结果的寄存器。

### 8.4.2 计算顺序和运行时间

计算顺序如下所示:

(1) 乘以位置比例增益  $K_{pp}$  (乘法运算处理  $2.5\mu\text{s}$ )。

(2) 计算速度偏差(减法运算处理  $3.5\mu\text{s}$ )。

- (3) 乘以速度比例增益  $K_{pv}$  (乘法运算处理  $2.5\mu s$ )。
- (4) 将积分的保存值和速度偏差数据相加 (加法运算处理  $3.125\mu s$ )。
- (5) 积分项极限的处理 (除法运算处理  $2.5625\mu s$  + 加减法运算处理  $3.5\mu s$ )。
- (6) 乘以速度积分增益  $K_{iv}$  (乘法运算处理  $2.55\mu s$ )。
- (7) 在速度比例乘法运算结果中加上速度积分乘法运算结果 ( $3.123\mu s$ )。

整个浮点的运算时间需要  $23.31\mu s$ , 加上其他的处理时间, 位置/速度环控制器的实际运行时间为  $30\mu s$ 。

### 8.4.3 在积分项中设置极限值

与 PID 控制器一样, 位置/速度环控制器也需要设置积分项的极限值。

## 8.5 IIR 数字滤波器

程序如程序清单 8.13 所示。在 IIR 数字滤波器中使用的是在 8.2 节中介绍的 16 位浮点的计算方法。注意在 IIR 数字滤波器中的寄存器名也会有改变。

程序清单 8.13 IIR 数字滤波器

```

; *****
; *****
; ***** Title: IIR Equation Program for H8S/2367. *****
; *****
; ***** Func: 在输入数据中通过 IIR 滤波器的计算之后再输出 *****
; *****
; ***** Source File Name : IIR.SRC *****
; *****
; ***** Object File Name : IIR.OBJ *****
; *****
; *****
; *****
PROGRAM IIR ; Program Name
CPU 2000A:24 ; CPU is H8S/2000 Advanced

; 外部调用子程序
IMPORT FADD16 ; 浮点的加法运算
IMPORT FSUB16 ; 浮点的减法运算
IMPORT FDIV16 ; 浮点的除法运算
IMPORT FMUL16 ; 浮点的乘法运算

; 需要调用的外部变量(全局变量是用 C 语言声明的)

```

```

; 外部定义的子程序
        .EXPORT      IIREQN

; 改变寄存器名
DATA0   .REG         (ER0)           ; 计算数据 0
DATA1   .REG         (ER1)           ; 计算数据 1
X        .REG         (ER2)           ; 输入输出
HISTPTR .REG         (ER4)           ; 计算范围
COEFPTR .REG         (ER5)           ; 系数地址
Y        .REG         (ER6)           ; 输出数组指针
n        .REG         (R3)           ; 环路计数器

; 确保数据的领域
        .SECTION     RAM,DATA,ALIGN = 2
IIRCALC .RES. L      9
RA1     .RES. L      1
RA1     .RES. L      1
RA2     .RES. L      1
RA2     .RES. L      1

        .SECTION     ROM,CODE,ALIGN = 2 ; ROM Area Section
IIREQN:
        MOV. L        # IIRCALC,Y
        MOV. L        X,@(0:16,Y)    ; Y[0] = X
IIRLOOP:
        MOV. L        @COEFPTR+ ,DATA0 ; a2
        MOV. L        @HISTPTR,DATA1   ; d(n-2)
        JSR           @FMUL16          ; ra1 = a2 * d(n-2)
        MOV. L        DATA0,@RA1      ; 保存 ra1
        MOV. L        @COEFPTR+ ,DATA0 ; b2
        MOV. L        @HISTPTR+ ,DATA1 ; d(n-2)
        JSR           @FMUL16          ; rb1 = b2 * d(n-2)
        MOV. L        DATA0,@RB1      ; 保存 rb1
        MOV. L        @RA1,DATA0       ; ra1
        MOV. L        @Y+ ,DATA1
        JSR           @FADD16          ; ra1 + = Y[n]
        MOV. L        DATA0,@RA1      ; 保存 ra1
        MOV. L        @COEFPTR+ ,DATA0 ; a1
        MOV. L        @HISTPTR+ ,DATA1 ; d(n-1)
        JSR           @FMUL16          ; ra2 = a1 * d(n-1)
        MOV. L        DATA0,@RA2      ; 保存 RA2
        MOV. L        @COEFPTR+ ,DATA0 ; b1
        MOV. L        @HISTPTR+ ,DATA1 ; d(n-1)
        JSR           @FMUL16          ; rb2 = b1 * d(n-1)
        MOV. L        DATA0,@RB2      ; 保存 rb2
        MOV. L        @RA1,DATA0

```

```

MOV.L   @RA2,DATA1
JSR     @FADD16           ; ra1 + = ra2
MOV.L   DATA0,@RA1      ; 保存 ra1
MOV.L   @RB1,DATA0
MOV.L   @RB2,DATA1
JSR     @FADD16           ; rb1 + = rb2
MOV.L   DATA0,@RB1      ; 保存 rb1
MOV.L   @COEFPTR+,DATA0  ; b0
MOV.L   @RA1,DATA1       ; d(n)
JSR     @FMUL16           ; rb2 = b0 * d(n)
MOV.L   DATA0,@RB2      ; 保存 rb2

; 准备下一次的环路计算
MOV.L   @(-4:16,HISTPTR),DATA0 ;
MOV.L   DATA0,(-8:16,HISTPTR) ; HistPtr[n * 3 + 0] = HistPtr[n * 3 + 1]
MOV.L   @RA1,DATA0
MOV.L   DATA0,@(-4:16,HISTPTR) ; HistPtr[n * 3 + 1] = ra1
MOV.L   @RB1,DATA0
MOV.L   @RB2,DATA1
JSR     @FADD16           ; y[n + 1] = rb1 + rb2
MOV.L   DATA0,@Y        ;
DEC.W   #1,n
BNE     IIRLOOP

; 终止环路
MOV.L   @Y,X
RTS

```

### 8.5.1 寄存器和局部变量的使用方法

#### 1. DATA0(ER0)

通用的计算寄存器。在浮点的各种运算方法中,DATA0(ER0)作为保存参数和计算结果的寄存器。

#### 2. DATA1(ER1)

通用的计算寄存器。在浮点的各种运算方法中,DATA1(ER1)作为保存参数的寄存器。

#### 3. x(ER2)

IIR 数字滤波器的输入输出寄存器。

#### 4. n(ER3)

指定 IIR 数字滤波器的环路次数。比如说 IIR 数字滤波器的级联当中,分



两段使用的情况下则为 2,最大的环路数为 8。

### 5. HISTPTR(ER4)

作为在 IIR 数字滤波器中使用的计算范围的指针来使用。

HISTPTR[0]:保存  $d(n-2)$ (第二段)。

HISTPTR[1]:保存  $d(n-1)$ (第一段)。

HISTPTR[2]:没有使用。

HISTPTR[3]:保存  $d(n-2)$ (第二段)。

HISTPTR[4]:保存  $d(n-1)$ (第一段)。

HISTPTR[5]:没有使用。

### 6. COEFPTR(ER5)

作为在 IIR 数字滤波器中使用的系数范围的指针来使用。

COEFPTR[0]:IIR 系数  $a_2$ (第一段)。

COEFPTR[1]: IIR 系数  $b_2$ (第一段)。

COEFPTR[2]: IIR 系数  $a_1$ (第一段)。

COEFPTR[3]: IIR 系数  $b_1$ (第一段)。

COEFPTR[4]: IIR 系数  $b_0$ (第一段)。

COEFPTR[5]: IIR 系数  $a_2$ (第二段)。

COEFPTR[6]: IIR 系数  $b_2$ (第二段)。

COEFPTR[7]: IIR 系数  $a_1$ (第二段)。

COEFPTR[8]: IIR 系数  $b_1$ (第二段)。

COEFPTR[9]: IIR 系数  $b_0$ (第二段)。

### 7. TEMP(ER6)

作为保存 IIT 数字滤波器中的各个输出结果的指针来使用。

Y[0]:X(输入数据)。

Y[1]:第一段的输出结果。

Y[2]:第二段的输出结果。

Y[3]:第三段的输出结果。

Y[4]:第四段的输出结果。

Y[5]:第五段的输出结果。

Y[6]:第六段的输出结果。

Y[7]:第七段的输出结果。

Y[8]:第八段的输出结果。

## 8. 局部变量

IIRCALC . RES. L 9 ; 保存 IIR 数字滤波器各个输出结果

RA1 . RES. L 1 ; 暂时保存计算结果

RB1 . RES. L 1 ; 暂时保存计算结果

RA2 . RES. L 1 ; 暂时保存计算结果

RB2 . RES. L 1 ; 暂时保存计算结果

## 8.5.2 计算顺序和运行时间

运算顺序如下所示:

(1) 系数  $a_2$  和节点  $d(n-2)$  相乘, 将计算结果保存于 RA1 中(乘法运算处理  $2.5\mu\text{s}$ )。

(2) 系数  $b_2$  和节点  $d(n-2)$  相乘, 将计算结果保存于 RB1 中(乘法运算处理  $2.5\mu\text{s}$ )。

(3) 系数  $a_1$  和节点  $d(n-1)$  相乘, 将计算结果保存于 RA2 中(乘法运算处理  $2.5\mu\text{s}$ )。

(4) 系数  $b_1$  和节点  $d(n-1)$  相乘, 将计算结果保存于 RB2 中(乘法运算处理  $2.5\mu\text{s}$ )。

(5) 将 RA1 和 RA2 相加, 计算结果为节点  $d(n)$ , 保存在 RA1 中(加法运算处理  $3.125\mu\text{s}$ )。

(6) 将 RB1 和 RB2 相加, 保存在 RB1 中(加法运算处理  $3.125\mu\text{s}$ )。

(7) 将系数  $b_0$  与节点  $d(n)$  相乘, 将计算结果保存在 RB2 中(乘法运算处理  $2.5\mu\text{s}$ )。

(8) 将 RB1 与 RB2 相加, 计算结果为滤波器的输出, 保存在  $Y(n)$  中(加法运算处理  $3.125\mu\text{s}$ )。

整个浮点的运算时间需要  $21.875\mu\text{s}$ , 加上其他的处理时间, IIR 数字滤波器一个环路的实际运行时间为  $30\mu\text{s}$ 。

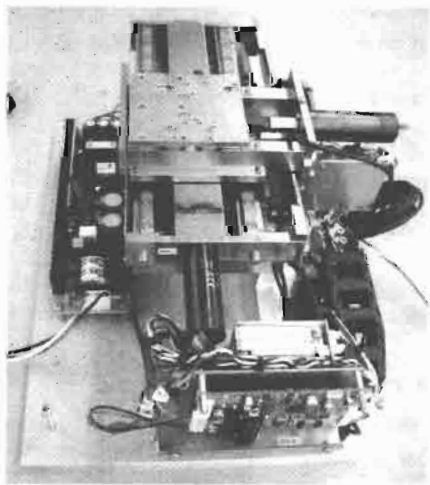
# 第 9 章

## 交流伺服电机的控制实验

本章中主要介绍基于微控制器的电机控制实验。用实验来验证前几章介绍的电机的驱动方法和电机的控制方法。深入浅出地介绍伺服的调整顺序和反馈控制以及前馈控制。

### 9.1 电机控制实验的设备

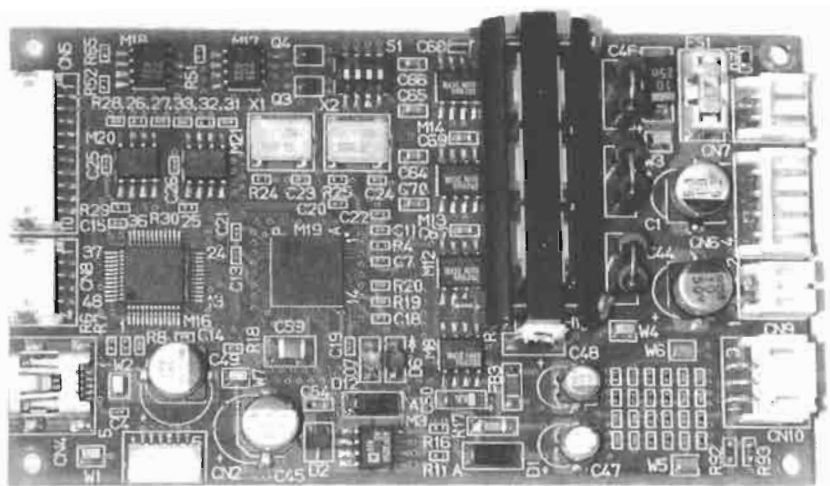
照片 9.1 所示的是在本章实验当中所要用到的线性工作台的外观。



照片 9.1 线性工作台的外观

### 9.1.1 电机控制主板

在照片 9.2 中展示的是在本次实验中所要用到的电机控制主板。电机控制主板是笔者设计的伺服电机驱动。这个主板的特征是采用了主机接口,只要有 PC 和电源就可以非常容易地进行电机控制实验。



照片 9.2 电机控制主板的外观

在一般工业用的伺服系统中,一般采用的都是脉冲控制,但是本次实验中使用的电机控制主板的位置指令不是脉冲,而是将最终目标位置由主机发送给 USB 的指令。每个伺服采样的电机动作轨迹是由搭载驱动的微控制器来计算的,所以不需要特别的脉冲发生器等一些特别的硬件。

下面是主板的一般组成:

- (1) 基板大小:长 90mm×宽 54mm×高 17.1mm。
- (2) 主机接口:USB1.0 标准(12Mbps)。
- (3) 供给电源:18~48V(用于驱动电机),18~24V(用于编码器),USB5V(用于控制)。
- (4) 控制的轴数:一轴电机。
- (5) 电机驱动方式:三相正弦 PWM 驱动。
- (6) 电流检测分辨率:0.004467A/count。
- (7) 使用的微控制器:H8S/2367F。



### 9.1.2 使用的电机

照片 9.3 中展示的是实验中所使用的电机的外观。是 maxon motor 公司生产的无刷电机(EC2250W)。这种电机是内部装有霍尔 IC 的交流伺服电机。这种电机的一般组成如下所示:

- (1) 电机种类:交流伺服电机。
- (2) 额定功率:50W。
- (3) 标准电压:32V。
- (4) 最大连续电流:2.82A。
- (5) 转矩常数:0.0136N·m/A。
- (6) 编码器:无。
- (7) 转子位置检测:HALL 三相(120°)。
- (8) 转子转动惯量: $4.2 \times 10^{-7} \text{ kg} \cdot \text{m}^2$ 。



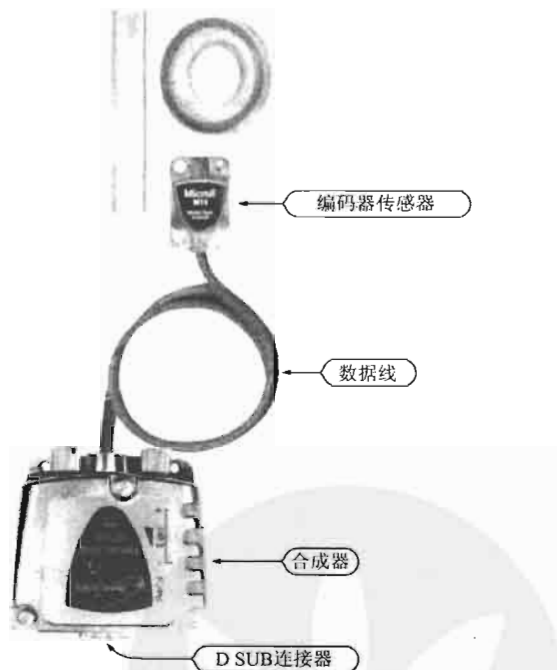
照片 9.3 电机的外观(maxon motor 公司,EC22 50W)

### 9.1.3 实验中用的负荷

实验用的负荷是为了开发电机控制主板而制作的线性工作台。这个工作台设置了  $0.1\mu\text{m}$  的线性编码器,构成了全闭环伺服机构。

照片 9.4 展示的是在本章的实验中所要用到的线性编码器的外观。线性编码器是 Microe Mercury2000。

- (1) 线性编码器: $0.1\mu\text{m}$ 。
- (2) 探头长度:1mm。
- (3) 负荷转动惯量: $82 \times 10^{-7} \text{ kg} \cdot \text{m}^2$ (包含转子的转动惯量)。
- (4) 极限传感器:光电传感器 $\times 2$ 。



照片 9.4 线性编码器的外观 (MicroE 公司, Mercury2000)

## 9.2 伺服调整方法

最近电机控制器发展迅速,加上(电机+负荷)特性的频率分析功能,使得伺服的自动调整功能的精度更加理想,变得谁都可以很简单地控制伺服电机了。由于本章中所用的电机控制主板用的是公开的伺服运算的传递函数,所以我们运用这个传递函数,从理论上进行伺服调整。首先用 PID 控制器进行伺服调整。PID 控制器是在反馈环中比较容易理解的一种控制器。

### 9.2.1 有关伺服调整的一些参数

虽然只是说伺服调整,但是与之相关联的参数也非常多。为了将电机、机械的最大能力发挥出来,非常有必要找出最适合的参数。在实验中所使用的电机控制主板中也记录了主要的伺服参数。到(7)为止都是与电机的性能相关的参数,(8)项以后是和安全使用电机相关的参数。



- (1) PID 参数(位置控制,速度控制)。
- (2) 加速度极限(计算轨迹用)。
- (3) 速度极限(计算轨迹用)。
- (4) 驱动极限(转矩控制放大器的电流极限)。
- (5) 前馈加速度系数。
- (6) 前馈速度系数。
- (7) 数字滤波器系数。
- (8) 电机电流总量规定的设定。
- (9) 速度超过的设定。
- (10) 稳定范围设定。
- (11) 稳定时间设定。
- (12) 动作范围的上限与下限的设定。

## 9.2.2 伺服锁定的实现

如果使用电机控制器的频率分析功能的话,就能分析机械(电机+负荷)的特性。但是使用这个功能需要至少实现伺服锁定功能。因此,这就需要解决如何实现伺服锁定的问题。

### 1. 决定 PID 参数(伺服增益)的初始值

最初为了实现伺服锁定,需要将 PID 参数值设低。另外,为了使其能够在更加广泛的频率范围内进行微分运算,积分增益也需要调低。试求使得伺服的控制频域为 20Hz 的 PID 参数。电机特性与纯粹的二次积分形状类似,传递函数的表达式如下所示

$$\frac{Y(n)}{X(n)} = \frac{K_a K_s K_t}{J s^2} \quad (9.1)$$

设频率为 20Hz,将以下参数代入上式,求其增益

$$K_a = 0.004467 [\text{A/count}] \quad (\text{放大增益})$$

$$K_t = 13.6 \times 10^{-3} [\text{N} \cdot \text{m/A}] \quad (\text{转矩常数})$$

$$J = 4.2 \times 10^{-7} [\text{kg} \cdot \text{m}^2] \quad (\text{转子转动惯量})$$

$$K_s = 1591.55 [\text{count/rad}] \quad (\text{传感器增益})$$

$$\frac{Y(n)}{X(n)} = \frac{0.004467 \times 1591.55 \times 13.6 \times 10^{-3}}{4.2 \times 10^{-7} \times (2 \times 3.141592 \times 20)^2} = \frac{0.09969}{0.006632} \approx 15.03$$

用 dB 表示的话是  $20\log_{10}(15.03) \approx 23.5(\text{dB})$ 。也就是说,在控制器的特性中如果将频率增益下调 23.5dB 的话,伺服频域应该正好为 20Hz。

图 9.1 和图 9.2 所示的是频率特性图。积分增益使用的是最小的  $K_i = 0.025$ 。这个振幅特性的 20Hz 增益是 3.6dB。若将 dB 的计算式  $Y = 20\log_{10}(X)$  指数化的话,则  $X = 10^{(Y/20)}$ 。

$$X = 10^{(-27.1/20)} = 10^{(-1.355)} = 0.04416$$

图 9.1 的 PID 控制器参数为  $K_i = 0.025, K_p = 1, K_d = 10$ 。

$$K_i = 0.025 \times 0.04416 = 0.001104$$

$$K_p = 1 \times 0.04416 = 0.04416$$

$$K_d = 10 \times 0.04416 = 0.4416$$

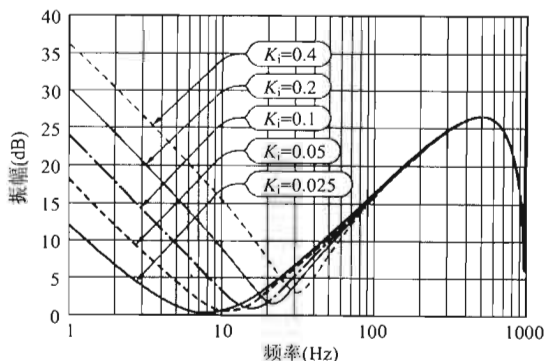


图 9.1 PID 控制器的频率特性(振幅)

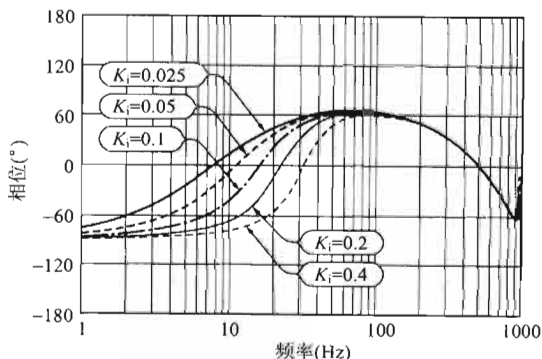


图 9.2 PID 控制器的频率特性(相位)

## 2. 惯性比

惯性比是负荷转动惯量(包含转子的转动惯量)与电机的转子转动惯量的比率。前一项的PID参数是从转子转动惯量中求得的,即电机无负荷工作的情况下求得的。

因为没有交流伺服电机单独工作的情况,所以在用电机驱动实际负荷的情况下,惯性比就变得非常有必要了。

## 3. 转动惯量的计算方法

图9.3是实验中使用的线性导轨的构造图。下面求联轴器和滚珠丝杆以及滑块的转动惯量。

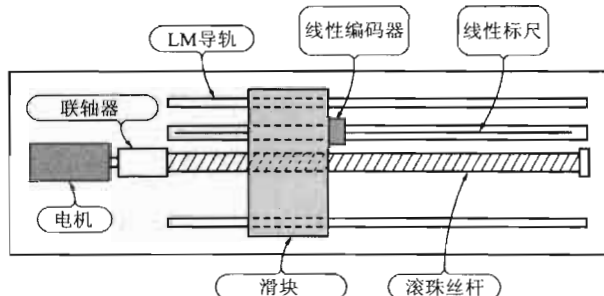


图 9.3 线性导轨的构造

联轴器和滚珠丝杆一类圆柱形的转动惯量  $J$  可以由以下公式求得

$$J(\text{kg} \cdot \text{m}^2) = \frac{\pi \gamma D^4 L}{32} \quad (9.2)$$

$\gamma$ : 材料的密度( $\text{kg}/\text{m}^3$ );  $D$ : 直径( $\text{m}$ ),  $L$ : 长度( $\text{m}$ )。

联轴器的转动惯量设为  $J_1$ , 将  $\gamma = 7800 \text{kg}/\text{m}^3$ ,  $D = 0.01 \text{m}$ ,  $L = 0.02 \text{m}$  代入式(9.2)中可得

$$J_1 = 3.13 \times 7800 \times 0.01^4 \times 0.013 \div 32 = 1.53 \times 10^{-7} \text{kg} \cdot \text{m}^2$$

下面, 设滚珠丝杆的转动惯量为  $J_2$ , 将  $\gamma = 7800 \text{kg}/\text{m}^3$ ,  $D = 0.008 \text{m}$ ,  $L = 0.25 \text{m}$  代入式(9.2)中可得

$$J_2 = 3.14 \times 7800 \times 0.008^4 \times 0.25 \div 32 = 7.84 \times 10^{-7} \text{kg} \cdot \text{m}^2$$

另外, 由滚珠丝杆驱动的移动体的转动惯量  $J$  可以由以下公式求得

$$J(\text{kg} \cdot \text{m}^2) = M \left( \frac{L}{2\pi} \right)^2 \quad (9.3)$$

$M$ :移动体的质量(kg), $L$ :导轨长度(m)。

滑块的转动惯量设为  $J_3$  的话,将  $M=1.0\text{kg}$ , $L=0.001\text{m}$  代入式(9.2)中可得

$$J_3 = 1 \times (0.001 \div (2 \times 3.14))^2 = 2.53 \times 10^{-8} \text{ kg} \cdot \text{m}^2$$

将联轴器、滚珠丝杆、滑块及转子转动惯量作为总负荷转动惯量,之后我们来求惯性比。

$$\begin{aligned} & 1.53 \times 10^{-7} + 7.84 \times 10^{-7} + 2.53 \times 10^{-8} + 4.2 \times 10^{-7} \\ & = 13.923 \times 10^{-7} \text{ kg} \cdot \text{m}^2 \end{aligned}$$

$$\begin{aligned} \text{惯性比} &= \text{负荷转动惯量} \div \text{转子转动惯量} = \frac{13.923 \times 10^{-7} \text{ kg} \cdot \text{m}^2}{4.2 \times 10^{-7} \text{ kg} \cdot \text{m}^2} \\ &= 3.315 \end{aligned}$$

#### 4. PID 参数(伺服增益)的修正

由于转动惯量在电机的传递函数中是分母部分的,惯性比为 3.315 的情况下,对于电机的特性来说则是  $1/3.315$ 。由于前一项的 PID 参数已经在只有电机的情况下算出来了,为了得到与负载相连时同样的频域,只需要将 PID 的参数设置成 3.315 倍即可。

$$K_i = 0.001104 \times 3.315 = 0.00366$$

$$K_p = 0.04416 \times 3.315 = 0.1464$$

$$K_d = 0.4416 \times 3.315 = 1.464$$

像这样,作为目的的伺服频域和惯性比已经决定的话,PID 的参数(伺服增益)就可以算出来了。负荷转动惯量的构造很复杂,由计算来得出的话比较困难,将最初伺服锁定时的惯性比作为参考值,为了使即便计算多少有点误差也不至于导致伺服电机无法正常工作,应选择有余度的伺服频域。

### 9.2.3 伺服调整工具

电机控制主板里带的伺服调整工具具有以下功能。在实际进行电机调整的时候,可以使用伺服调整工具。

#### 1. 伺服环频率特性分析功能

用标记扫描的方法来分析伺服的开环以及电机机械(设备特性)的频率特性。

#### 2. 跟踪功能

电机电流、位置、伺服内部计算等,在每个伺服采样中都有跟踪功能。

### 3. PID 控制器的设计功能

计算 PID 控制器的频率特性,加上电机机械(设备特性)的频率特性,预测开环特性。

### 4. 数字滤波器的设计功能

进行低通滤波器以及节点滤波器的数字滤波器的设计。加上 PID 控制器以及电机机械(设备特性)的频率特性,预测开环特性。

## 9.2.4 PID 参数的决定方法

由于 PID 控制器是反馈控制器,所以决定了电机机械的基本的动作特性。在实验中考虑到高速动作和稳定动作的平衡,设定伺服频率范围为 100Hz、相位余度  $45^\circ$  为目标值。

### 1. 机械(电机+负荷)特性的测定

因为已经实现了伺服锁定,所以可以测定机械(电机+负荷)特性了。测定频率从 1Hz 开始到 100Hz 结束。

图 9.4 和图 9.5 所示的为测定结果。请参照 plant 项目。从这个测定结果来求负荷的转动惯量  $J$  的话,由于 plant 的振幅的过零频率为 41Hz,从式(9.1)中的传递函数中可得

$$\begin{aligned}\frac{Y(n)}{X(n)} &= \frac{K_a K_s K_t}{J s^2} = \frac{0.004467 \times 1591.55 \times 13.6 \times 10^{-3}}{J \times (2 \times 3.141592 \times 41)^2} \\ &= \frac{0.09969}{J \times 66363.2} = 1\end{aligned}$$

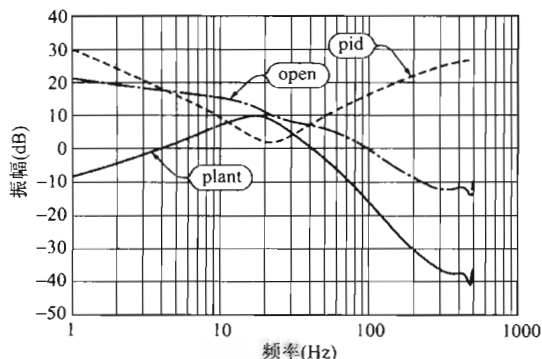


图 9.4 线性导轨的频率特性(振幅)

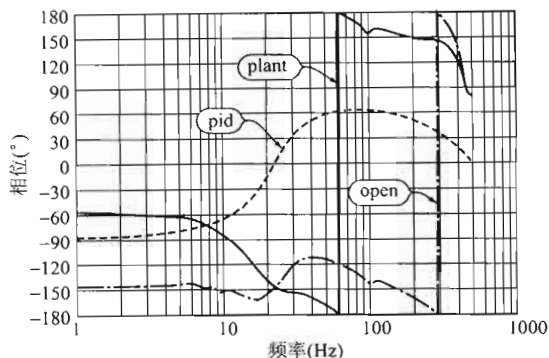


图 9.5 线性导轨的频率特性(相位)

$$J = 0.09969 \div 66363.2 = 15.02 \times 10^{-7} \text{ kg} \cdot \text{m}^2$$

惯性比为 3.5, 用转动惯量来计算的话, 惯性比为 3.315, 这个程度的误差是在允许范围内的。

另外, 参看图 9.4 和图 9.5 的 plant 项目的话, 低频范围的增益下降了, 这是由于考虑到了摩擦的影响。由于在实验中使用的线性导轨的摩擦很大, 所以在 PID 参数里面, 需要考虑到这个因素。

## 2. PID 参数的决定

使用伺服调整工具中的 PID 控制器的设计功能来设计 PID 参数。由于伺服调整工具是在对话框中的 PID 参数项中输入值, 点击图像显示按钮就马上生成开环的图, 所以将伺服频率范围设置在 100Hz 左右, 相位的余度设置在  $45^\circ$  左右作为 PID 的参数。

图 9.4 和图 9.5 是这个设计的例子。请参考 open 的项目。伺服频率范围在 100Hz, 这个时候的相位余度为  $45^\circ$ 。所设计的 PID 参数为  $K_i=0.2$ ,  $K_p=1$ ,  $K_d=10$ 。考虑到线性导轨的摩擦, 将积分增益  $K_i$  调大。

## 9.2.5 加速度极限和速度极限的确定方法

基于前面的 PID 参数, 基本的反馈特性已经确定了。接下来, 需要确定电机及机械的动作轨迹。考虑到这个轨迹反映的是整体的性能, 也非常重要。比如说, 给定超过电机或者是电机驱动的能力的加速度的话, 电机就完全不会响应这个给定的轨迹。在这种状态下使用的话, 可能会损坏电机或者是电机驱动器。



### 1. 最高转速(速度极限)

在确定最高转速时需要考虑负荷转动惯量和摩擦的影响。

在实验中所用的电机是高转速型电机。最大所允许的转速为 50000r/min。实验中用的负荷是基于滚珠丝杆的线性导轨,会产生摩擦,因此采用标准的交流伺服电机的额定转速 3000r/min。

角速度为  $3000\text{r/min} \div 60\text{s} \times 2 \times \pi = 314.16\text{rad/s}$

### 2. 给定电机的角加速度(加速度极限)

决定加速度的要素有很多,在这里给出由到达最高转速的时间求得的方法。假设到达最高转速的时间为 50ms,那么角加速度便可以由以下公式求得

$$\begin{aligned}\text{角加速度} &= (\text{角速度}) / (\text{到达时间}) = \frac{314.16\text{rad/s}}{0.05\text{s}} \\ &= 6283.2\text{rad/s}^2\end{aligned}$$

由这个角加速度来计算电机的电流。由于产生的转矩  $T$ 、转动惯量  $J$ ,以及角加速度  $\alpha$  的关系表达式  $T = J\alpha$ ,所以产生的转矩  $T$  为

$$T = 15.02 \times 10^{-7} \text{kg} \cdot \text{m}^2 \times 6283.2\text{rad/s}^2 = 9.437 \times 10^{-3} \text{N} \cdot \text{m}$$

另外产生的转矩  $T$ 、转矩常数  $K_t$  及电机电流  $I$  的关系式为  $T = K_t I$ ,所以可由下式求得  $I$

$$I = \frac{T}{K_t} = \frac{9.437 \times 10^{-3} \text{N} \cdot \text{m}}{13.6 \times 10^{-3} \text{N} \cdot \text{m/A}} = 0.694 \text{A}$$

由于电机驱动的最大驱动能力是 6A,即使考虑到摩擦的情况下,也有足够的余度。

### 3. 电机驱动的单位变换

由于给定电机驱动的加速度极限单位为  $\text{count/s}^2$ ,速度极限单位为  $\text{count/s}$ ,因此需要进行单位上的转换。

$$\begin{aligned}\text{加速度极限} &= 6283.2\text{radian/s}^2 \div 2 \div \pi \times 10000\text{count/圈} \\ &= 10000000\text{count/s}^2\end{aligned}$$

$$\begin{aligned}\text{速度极限} &= 314.16\text{radian/s} \div 2 \div \pi \times 10000\text{count/圈} \\ &= 500000\text{count/s}\end{aligned}$$

### 9.2.6 驱动极限(转矩控制放大器的电流极限)的确定方法

电机驱动器的最大驱动电流是 6A。这次实验中设定驱动的极限为 5A。

由于在从放大器增益中,电流的单位是 0.004467A/count,因此

$$\text{驱动极限} = 5\text{A} \div 0.004467\text{A/count} = 1119\text{count}$$

### 9.2.7 前馈的确定方法

前馈用实验的方法求得。一开始将前馈的速度项及前馈的加速度项都置零。

### 9.2.8 数字滤波器的系数

由于在本章的实验中,只用 PID 控制器便得到了令人十分满意的性能,没有使用到数字滤波器。

## 9.3 电机控制实验(PID 控制器)

那么,用实际实验时用的负荷,操作一下电机看看吧。首先,用 PID 控制器进行位置控制。

### 9.3.1 微小距离传送的定位性能

设定导轨的移动量为 0.1mm(电机转动 1/10 圈),测定在此传送距离下的定位性能。在前馈有效和无效的情况下进行对比和说明。

#### 1. 只有反馈(PID)控制下的定位性能

请参考图 9.6 的定位性能(pos),这是只有反馈(PID)控制的情况。导轨移动量在 0.1mm 的情况下,可以看到有 0.015mm 的超调量。另外,如果将移动后的稳定状态设定在以  $\pm 0.5\mu\text{m}$  以内的话,调整时间为 100ms 左右。

#### 2. 反馈(PID)控制+前馈控制下的定位性能

请参考图 9.7 的定位性能(pos)。现在为使用前馈控制的情况。超调情况基本没有,可以验证对电机响应性能的改善。如果将移动后的稳定状态设定为和前面相同的  $\pm 0.5\mu\text{m}$ ,调整时间为 0ms。

另外,使用前馈控制还能减少移动中位置的偏差,这在对前馈控制的理解

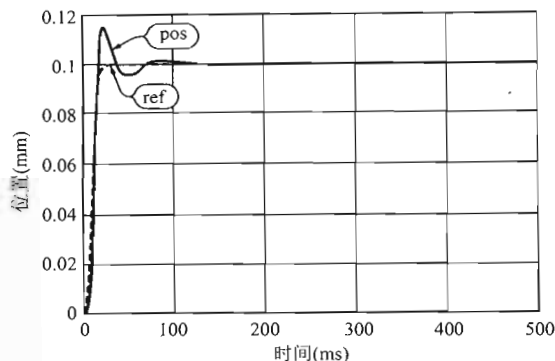


图 9.6 微小距离传送的定位性能(PID 控制)

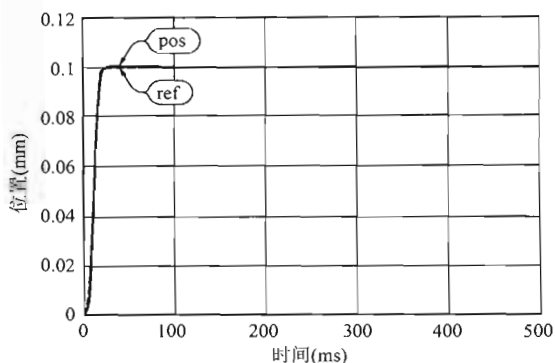


图 9.7 微小距离传送的定位性能(PID+前馈控制)

上,有着非常重要的意义。

### 3. 只有反馈(PID)控制下的电机电流特性

图 9.8 是只有反馈控制情况下的电机电流图。电机的加速电流约为  $+1.5\text{A}$ ,电机的减速电流约为  $-0.7\text{A}$ 。

在反馈控制中,位置偏差决定了电机电流的所有参数。由于电机加速/减速都需要电机电流,而电机电流是位置偏差通过伺服控制器放大之后得到的,所以相对于基准位置,电机的实际位置肯定会有延迟情况的发生。

### 4. 反馈(PID)控制+前馈控制情况下的电机电流特性

图 9.9 是使用前馈控制情况下的电机电流图。图中,反馈电流成分记为 FB 电流,前馈电流成分记为 FF 电流。

(1) 反馈电流成分:反馈电流是用 PID 控制器计算的位置偏差的结果。使

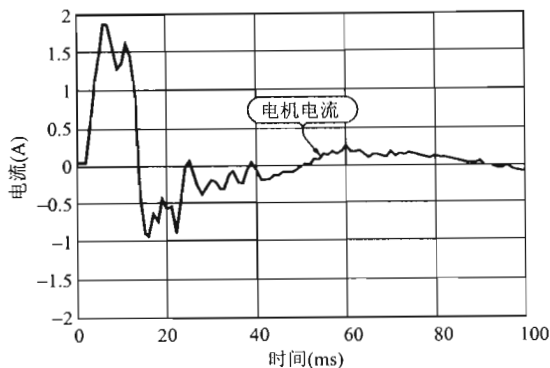


图 9.8 微小距离传送的电机电流(PID 控制)

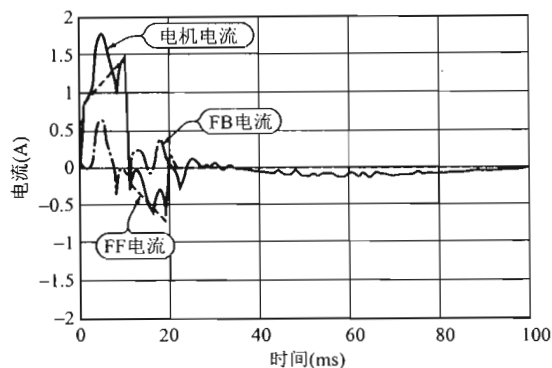


图 9.9 微小距离传送的电机电流(PID+前馈控制)

用前馈的情况下,由于移动中位置的偏差减小了,反馈电流成分也变小了。

(2) 前馈电流成分:前馈电流是通过目标位置生成的。由于前馈作用在反馈回路之外,不会影响电机的实际位置,因此,形成了没有噪声的完美波形。前馈电流在电机加速或者减速的时候有倾斜的原因是,加上了前馈加速度项和前馈的速度项。最终的前馈系数为:

前馈加速度增益:18.0。

前馈速度增益:1.5。

(3) 综合电机电流:综合电机电流是反馈电流成分和前馈电流成分的合成。由于像这样驱动电机加速或者减速的时候,所需要电流的大部分都是由前馈模块产生的,所以能够减少电机移动中的位置偏差,并能缩短移动后调整时间。

但是,前馈控制并不是万能的,由于电机及机械的特性(设备的特性)应与前馈系数一致,所以在负荷变动的系统中不能使用。

### 9.3.2 高速传送响应性能

设导轨的移动距离为 10mm(电机转动 10 圈),则导轨的移动速度达到极限 50mm/s(电机的转速为 3000r/min)。请参考图 9.10,图中实线为导轨的位置。

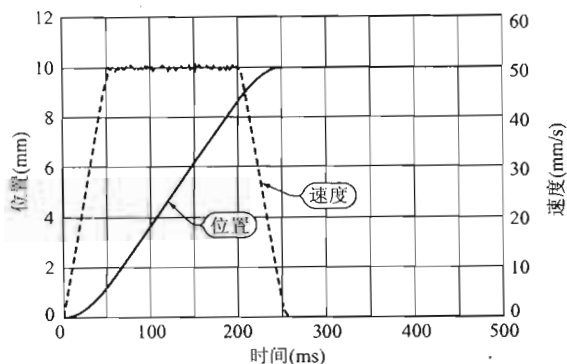


图 9.10 高速传送响应性能(PID 控制)

图中实线表示的是位置的数据,虚线表示的是速度。导轨的动作轨迹为台形的速度曲线。加速区间是 50ms,在 50ms 的时候达到了最高速度 50mm/s (3000r/min);恒定速度区间是 150ms,看图中的速度项可以发现,大部分是恒定的速度;减速区间是 50ms;由此可知,导轨移动 10mm 需要 250ms。

### 9.3.3 速度控制性能

至今为止进行的都是基于 PID 控制器的位置控制的说明,下面对速度控制进行实验。

#### 1. 速度控制的复习

在速度控制的情况下,根据编码器中的位置信息由电机控制主板内部进行微分运算,算出速度。因此,作为电机的基本特性,速度控制的情况下具有一次积分的特性。

因为一次积分的特性,相位延迟有  $90^\circ$ ,所以反馈控制器可以作为放大器稳定地进行控制。另外,即便是在速度控制当中,也不能像位置控制那样有理想

的积分特性,为了消除在恒定转动中产生的速度偏移,需要加上反馈控制器的积分因素。因此,将反馈控制器中 PID 控制器的微分项(D)置零,成为了 PI 控制。

## 2. 速度控制时候的 PID 参数

将位置控制时候的微分增益  $K_d$  作为速度控制时候的比例增益  $K_p$ 。速度控制的积分增益  $K_i$ ,使用伺服调整工具的跟踪功能来决定积分增益。由于在实验中使用的线性导轨是从图 9.4 的 plant 特性中判断出来有摩擦的,所以积分增益的设定应该调高。最终 PID 的参数为:

$$K_p = 10$$

$$K_i = 2$$

$$K_d = 0$$

## 3. 速度控制实验数据

图 9.11 中,将速度指令从 10mm/s、20mm/s、30mm/s、40mm/s 到 50mm/s 的时候各自的响应性能总结到图中。

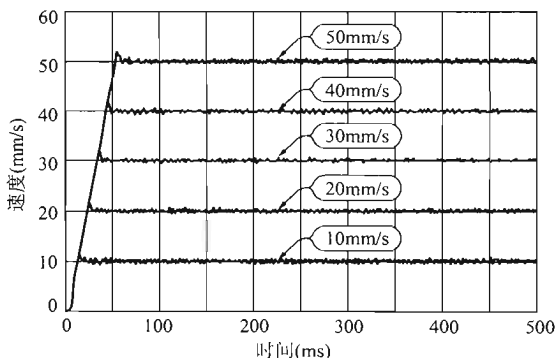


图 9.11 速度控制性能(PID 控制)

因为工作台的加速度都是一样的,所以速度的增加也都是一样的。从加速度区间到恒定速度区间转化的时候,虽然有一点过调的感觉,但所有的指令都很好地执行了。

## 9.4 电机控制实验(位置/速度环控制器)

接下来,试一下位置/速度环路控制器。由于位置/速度环控制器在位置控

制中不容易出现过调,所以在这里进行验证。

### 9.4.1 位置/速度环控制器是二重反馈控制器

图 9.12 表示的是速度位置环控制器的方框图。位置反馈环内侧有附带速度反馈环的二重反馈控制器。

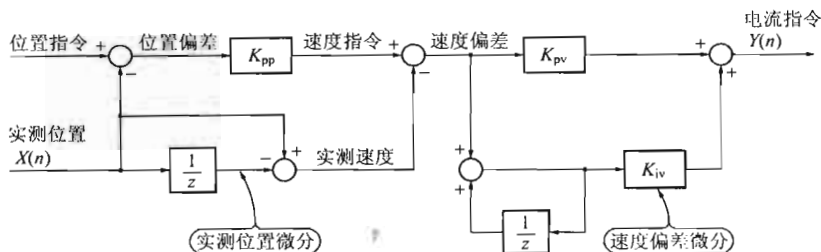


图 9.12 位置/速度环控制器的方框图

在位置反馈中,将位置指令与实际位置进行比较,将其偏差和位置比例增益  $K_{pp}$  进行比较。速度控制就是所谓的 PI 控制。因此,拥有速度比例增益  $K_{pv}$  和速度积分增益  $K_{iv}$  的参数。

位置/速度环控制器的特点是,在滑块渐渐地接近目标位置的情况下,位置偏差渐渐地缩小,速度指令也相应地成比例缩小。这个速度指令的变化动作是非常重要的,位置反馈环是速度反馈环的速度轨迹所生成的。比如说位置指令是阶梯状变化的情况下,滑块越接近目标位置,其速度指令变得越小,结果就能控制使其难以发生过调现象。

### 9.4.2 伺服增益的决定

这次的实验是和 PID 控制器边对比边进行的实验。位置/速度环控制器中的伺服频域和 PID 控制器的情况相同。PID 控制器的时候的伺服频域是 100Hz,只要符合 100Hz 情况下的振幅就可以了。

下面使用第 4 章中使用的数字滤波器的设计支援软件(DSP. EXE),进行位置/速度环控制器的设计。设计的参数如下所示:

位置比例增益  $K_{pp}=0.25$

速度比例增益  $K_{pv}=8.0$

速度积分增益  $K_{iv}=0.8$

图 9.13 是 PID 控制器(pid)和位置/速度环控制器(ppi)的振幅项比较。使得低频领域和 100Hz 的振幅一致。另外,图 9.14 是 PID 控制器(pid)和位置/速度环控制器(ppi)的相位项比较。

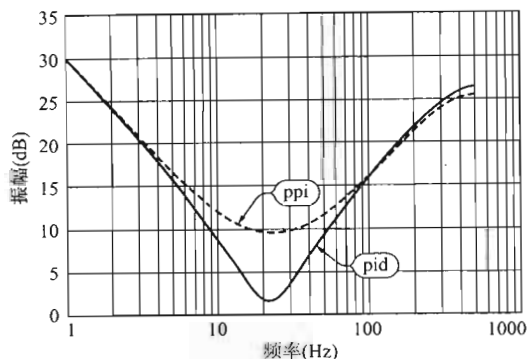


图 9.13 PID 与位置/速度环控制器的振幅项比较

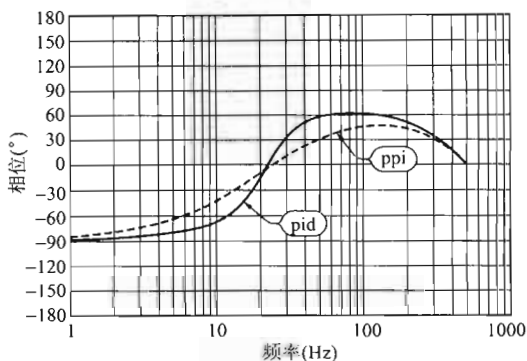


图 9.14 PID 与位置/速度环控制器的相位项比较

### 9.4.3 开环频率特性

图 9.15 和图 9.16 是位置/速度环控制时的线性导轨频率特性图。伺服频域大概在 100Hz,相位的余度在  $20^\circ$ 。将位置/速度环控制器的环路整形成与 PID 控制器频率特性类似的时候没有自由度。与 PID 控制器伺服频域相同的时候,不管如何都要牺牲相位的自由度。

但是,位置/速度环控制器也有其他的优点,下面针对这些优点进行说明。

#### 1. 微距离传送定位性能

在位置/速度环控制器中,基本上不需要前馈,只需要测定反馈控制。导轨



的移动量为 0.1mm(电机转动 1/10 圈)。

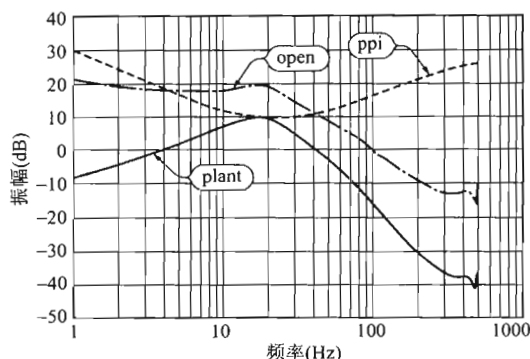


图 9.15 线性导轨的频率特性(振幅)

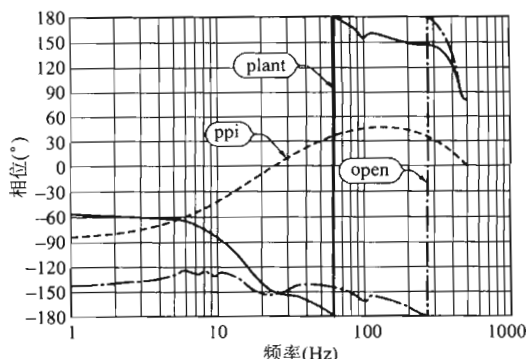


图 9.16 线性导轨的频率特性(相位)

## 2. 不会出现过调现象

在这次实验用的负荷线性导轨中,在微距离传送定位方面,对于伺服控制器来说控制非常难。不仅有滚珠丝杆的间隙,还有移动时间短的时候在反馈控制达到稳定之前到达目标位置的情况。因此,PID 控制器会产生较大的过调现象。如图 9.17 中所示的那样,在位置/速度环控制器中,完全没有发生过调现象。

虽然 PID+前馈控制也能抑制过调现象,但是由于前馈控制是环路外的控制,随着环境及所经过的时间的变化,导轨的状态发生变化的话,也会产生误差。因此虽然在进行伺服调整的时候是理想的数据,但是经过一段时间之后再拿出数据一看会发生“啊,怎么会变成这样”的情况。笔者也碰到过好几次这种情况。

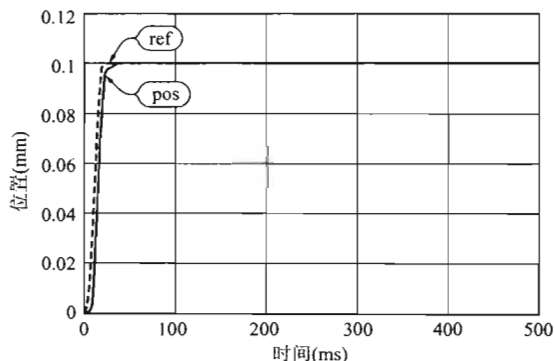


图 9.17 位置/速度环控制微距离传送定位性能

在这一点上,由于位置/速度环控制器只有反馈控制,导轨的些许变化都会被反馈环路吸收,能够得到和平常相同的响应性能。

### 3. 相对于基准位置出现的延迟

在位置/速度环控制器之中,位置环的偏差和位置比例增益的乘积是速度环中的指令速度。

因此,在导轨移动的时候,为了使其能够正确地移动,一定需要位置的偏差。因此,结果是相对于标准位置来说,实际位置产生了延迟。这种延迟,动作速度越快的时候,产生得越大。

### 4. 高速传送响应性能

如图 9.18 所示,位置/速度环控制也和 PID 控制的情况一样,使导轨移动 10mm(电机转动 10 圈),如图 9.18 所示,能得到和 PID 控制的情况(图 9.10)差不多相同的性能。

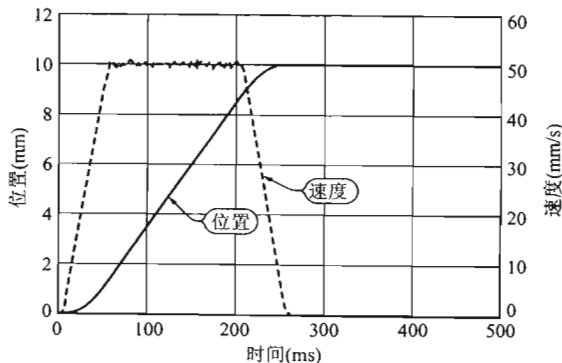


图 9.18 高速传送响应性能(位置/速度环控制)

## 9.5 电机的输出功率

虽然正确地测定转动着的电机所输出的功率需要通过功率检测仪等价格高昂的检测器,但是在这里我们也可以简单地通过电机的转动圈数和电机的电流来计算

$$P(W) = T\omega$$

$T$ : 转矩( $N \cdot m$ ),  $\omega$ : 角速度( $rad/s$ )。

另外,产生的转矩  $T$ 、转矩常数  $K_t$ ,以及电机电流  $I$  的关系式为

$$T(N \cdot m) = K_t I$$

$K_t$ : 转矩常数( $N \cdot m$ ),  $I$ : 电机电流(A)。

将这两个公式合并的话,则

$$P(W) = K_t I \omega$$

### 9.5.1 电机转动圈数和输出功率的关系

实验中使用的电机的额定输出功率是 50W,现在试着测定在导轨移动时电机的输出功率。图 9.19 是速度极限 50mm/s(3000r/min)时候的电机输出功率示意图。在恒定的速度区间,电机的输出功率约为 8W。

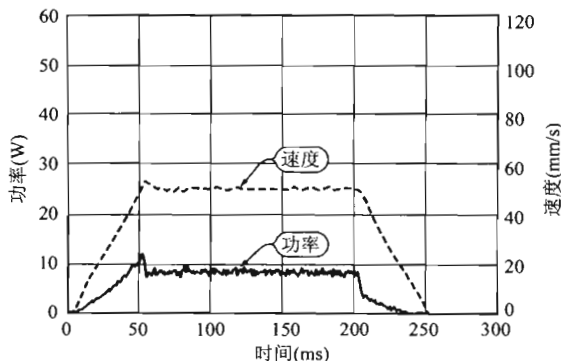


图 9.19 电机的输出功率(速度极限:50mm/s)

图 9.20 是速度极限改变到 100mm/s(6000r/min)时候的情况。在恒定速度区间的电机输出功率约为 26W。在任何一个条件下,从电机的额定输出来判断的话,在运用上都没有问题。电机在恒定转动中将负荷转矩只考虑成摩擦的话,由于摩擦转矩和转动圈数是成比例关系的,所以理论上电机的输出功率

与电机的转动圈数增加倍数的平方成比例。在这次的测定结果当中,将转动的圈数变成两倍的话,电机的输出功率是  $26\text{W}/8\text{W}=3.25$  倍。由于理论上电机的输出功率应该是 4 倍的,所以意外地出现了摩擦转矩没有增加的结果。

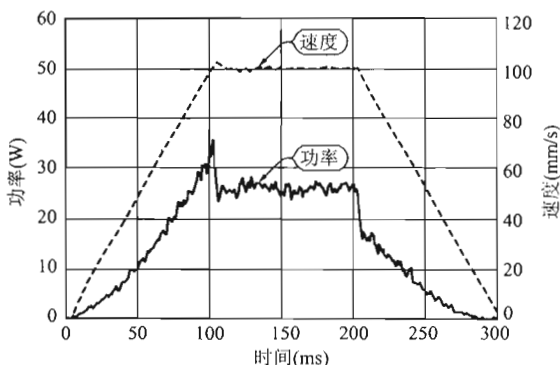


图 9.20 电机的输出功率(速度极限:100mm/s)

### 9.5.2 电机转动是在做功

电机正如其名,转动是其工作。电机的数据清单里记载的额定输出及额定转矩也是电机转动时候的值。比如说,对于导轨在垂直轴中有重力负荷的系统,在进行定位控制时,由于有重力的影响,所以电机需要产生相应的转矩。像这种情况,电机输入的功率全部被电机消耗了。这种功率大部分都被消耗成热了。即便电机产生的转矩都在额定转矩以内,电机坏掉的情况也有,需要留意电机的热容量。

# 附 录

## 交流/直流伺服电机驱动 MCG02

在第 9 章中所使用的交流伺服电机的驱动回路,已经被 MIYACHI(原富士通 AUTOMATION)商品化销售了。对于广大读者来说,非常适合实验等用途。

### 1. 特 征

电机控制器(MCG02)的特征是,采用了 USB 的超小型通用伺服驱动作为主机接口。虽然是超小型(名片大小),但它是一种搭载有动作控制器和伺服放大器的功能多合一系统。

### 2. 主板的一般组成

附表 1 表示了 MCG02 的一般组成。

附表 1 电机控制器 MCG02 的一般组成

基板大小	90mm(W)×54mm(D)×17.1mm(H)
主机接口	标准接口 USB1.0 标准(12Mbps)
供给电源	+18~+24V(用于驱动电机),+18~+24V(用于编码器),USB+5V(用于控制)。至 24V 位置,可以使用用于驱动电机的和编码器的单电源
控制轴数	电机轴 1 轴,电磁制动 1 轴
数字输入输出	输入 3ch(原点传感器 1,极限传感器 2),输出 2ch
模拟输入	输入 2ch(差动约为±400mV),电源是 4.5mA 的稳定电流源
电机驱动方式	三相正弦 PWM 驱动(无刷电机) H 桥 PWM 驱动(整流电机)

### 3. 电机的一般组成

附表 2 表示了适用于 MCG02 的电机的一般组成。

附表 2 适用于 MCG02 的电机的一般组成

电机形式	交流伺服电机, 直流伺服电机, 线性电机
平均最大输出	80W 级别(取决于电机的特性)
平均最大电流	4.0A(标准)
瞬间最大电流	6.0A(标准)
编码方式	增量式编码器(A, B, Z), 绝对式编码器(RS485)
编码合成数	4 倍递增(使用增量式编码器时)
编码接口	RS422, TTL, OC(使用增量式编码器) RS485(绝对式编码器)
最大解码能力	硬件 65536(count/电气角)(4 倍递增), 用软件可以扩展到 32 位
转子位置检测	HALL 三相(120°), HALLESS 控制

### 4. 咨询方式

MIYACHI SYSTEMS 公司([http://www.miyachi\\_sys.com](http://www.miyachi_sys.com))。

## 专业名词解释

### A

名 词	解 释
安培 (ampere)	法国物理学家,电磁学的创始人之一。发现了有电流流过时,以电流方向为轴的逆时针方向会产生磁场的现象,这被称为“右手法则”。电流的单位安培(A)是以安培来命名的

### B

名 词	解 释
饱和区 (saturated region)	源漏间电压上升的时候,载流子达到最大漂移速度,但电流不再上升。由于电子器件变得像电流源一样,所以输出阻抗变高了,这被称为饱和区。MOSFET 的动作和双极型三极管相比,双极型三极管的线性区及饱和区正好和 MOSFET 相反
步阶位置 (step position)	步进电机的停止位置(角度),也称为步进角
步进电机 (stepping motor)	在输入脉冲的时候,只转动输入脉冲数的角度的电机。其特征是开环位置控制

### C

名 词	解 释
采样频率 (sampling frequency)	将声音等模拟波形转变为数字数据的时候,每秒从连续信号中提取并组成离散信号的采样个数,单位为赫兹(Hz)。
齿槽转矩 (cogging torque)	由转子的永磁体磁场同定子铁心的齿槽相互作用,在圆周方向产生的转矩,与定子电流无关,它总是试图将转子定位在某位置上
传递函数 (transfer function)	可以表示输入输出特性。在连续系统中可以用拉普拉斯变换表示其频域特性。在离散系统的传递函数中,用延时单元 $1/z$ 表示时域特性
磁编码器 (magnetic encoder)	电机的转轴中加装编码器专用的永磁体,用磁场传感器进行检测的编码器。传感器元件有磁电阻元件和霍尔元件
磁电阻元件 (magneto resistance)	利用强磁体的磁电阻效应,阻抗值会随着磁场强度发生变化的传感器。在地磁场的情况下也能感应,使用的温度范围广泛,频率特性优良,具有在光学编码器使用困难的恶劣环境下也能使用的特点

## D

名 词	解 释
单精度浮点 (single precision floating point)	以 IEEE 为标准的单精度实数,符号部分 1 位,指数部分 8 位,小数部分 23 位,总计 32 位
单稳多谐振荡器 (one shot multivibrator)	输入触发脉冲的时候,输出一定时间间隔的单次脉冲的多谐振荡器,如 TTL 的 SN74121 等
电磁感应 (electromagnetic induction)	导体在磁场变化的环境下会产生电位差的现象
电动势 (induced electromotive force)	导体在有电流流过的时候产生磁通,磁通发生变化时候导体总会产生抵消其磁通变化方向的电动势
电 刷 (brush)	用于直流电机整流的电极。电刷是定子,与转动的换向器接触,切换电机线圈
电位计 (potentiometer)	一般来说指的是可变电阻。常用来将转动角或者移动量转换成电压
定 点 (fixed point)	定点运算基本上是整数运算。小数点的位置是由使用者(程序员)来决定的。根据小数点的位置决定 LSB 的加权。比如说,小数点的位置移动至 LSB 右边的话,LSB=1,变成了整数。小数点的位置移至 LSB 左边的话,LSB=0.5
定 子 (stator)	电机的固定机构部分。带刷直流电机中是永磁体,无刷电机中则为线圈
对数函数 (logarithm)	对数函数是指数函数的反函数。以 $e(2.71828\cdots)$ 为底的对数称为自然对数,以 10 为底的对数称为常用对数。
对 象 (object)	在软件中编译源代码的时候,指的是编译器生成的代码或者文件

## F

名 词	解 释
反 馈 (feedback)	将输出(结果)重新输入到输入端的系统。不仅用于放大器的特性改善、振荡、计算电路及自动控制电路等电子领域,在机械系统及生物系统等领域中也被广泛应用





<b>方框图</b> (block diagram)	用线和方框表示传递函数的图
<b>浮点 DSP</b> (floating point DSP)	DSP 的一种,内藏有浮点运算单元
<b>负荷角</b> (load angle)	由于步进电机转子相对旋转磁场(电机电流的转换)会有延迟,从而造成转矩的偏移。这个转子的延迟角被称为负荷角
<b>傅里叶变换</b> (Fourier transform)	用于函数变换的线性算法的一种,可以将时域函数变换成频域函数

## G

名 词	解 释
<b>感应电机</b> (induction motor)	是通过旋转的磁场和转子感应产生的电流(涡电流)而产生旋转转矩的电机
<b>古典控制理论</b> (classical control theory)	以被称为传递函数的线性输入输出系统所表示的控制对象为中心,以频率响应等评价预期动作的理论。1950 年被体系化,具有代表性的成果 PID 控制现在仍然是产业界的主力
<b>固有频率</b> (natural frequency)	用角频率( $\omega$ )表示固有振动。固有振动是指让物体自由振动的时候所检测出的特定的振动。任何物体都存在固有振动
<b>惯 性</b> (inertia)	物体在不受外力作用的时候,静止的物体会继续保持静止,运动的物体会继续保持运动的物理特性
<b>光学编码器</b> (optical encoder)	在旋转的码盘中有多个光栅,利用发光二极管发出的光通过固定的缝隙,通过感光三极管检测出的光,将缝隙的位置转换成电气信号的装置
<b>滚珠丝杆</b> (ball screw)	将旋转运动转化为直线运动或将直线运动转化为旋转运动的具有螺纹滚道的轴
<b>国际单位制</b> (International System of Units)	国际单位制(SI)是用被广泛使用的 MKS 单位制(长度单位米(m)、质量单位千克(kg)、时间单位秒(s))三种单位的组合而成表示各种量的单位的扩展。此单位制基于米制公约,在 1960 年被国际度量衡大会所采用

## H

名 词	解 释
<b>滑 块</b> (slider)	机构中与机架用移动副相连又与其他运动构件用转动副相连的构件

换向器 (commutator)	连接在和转子一起转动的圆筒电极的线圈上。电机旋转的时候和电刷接触,进行线圈的切换
霍尔传感器 (hall sensor)	利用霍尔效应的永磁体的磁极检测传感器。分为霍尔元件单体和内藏 OP 放大器的霍尔 IC

J

名 词	解 释
架 构 (architecture)	在计算机中表示基本设计、设计思想等基本设计概念。狭义的架构是指命令集架构
交流电机 (AC motor)	用交流电源驱动的电机电。分为同步电机和异步电机
交流伺服电机 (AC servo motor)	同步交流电机当中,有伺服机构和转换电路,转动圈数可以任意更改,而且可以实现高精度位置控制的电机
阶跃响应 (step response)	从输入信号为一定值开始突然变化为另一固定值时候的响应
截止频率 (cut off frequency)	在低通滤波器(Low Pass Filter)中,增益降低 3dB 的频率被称为截止频率
近似值 (approximate value)	在必要的误差范围内,即使用其他数值代替也没关系的数值。或者是删除一部分数值信息所得的值。也就是说,是被近似处理过的值
矩形波驱动 (rectangular wave drive)	直流电机的驱动方法的名称。虽然直流电机是直流电源驱动的,但是由于电机的电刷或者是驱动的放大电路(无刷直流电机)的整流作用的原因,驱动波形变成了矩形波
绝对式编码器 (absolute encoder)	输出绝对位置的光学编码器。旋转圆板中设有二进制码的光栅是其特征

K

名 词	解 释
开环传递函数 (open loop transfer function)	也称为循环传递函数。在闭环系统中,人为地断开系统的主反馈通路,将前向通道传递函数与反馈通路传递函数相乘所得到的函数

## L

名 词	解 释
拉普拉斯变换 (Laplace transform)	在函数解析学中,拉普拉斯变换是用积分定义函数空间的一种函数变换。使用拉普拉斯变换的话,可以将微分方程中的微分符号去掉,将微分方程变为代数方程,可以使式子变得简单
浪 涌 (surge)	在电气回路或者电气系统中超过通常电压,瞬间或者是断续地产生过压现象。这种浪涌电压会造成电气设备的绝缘破坏,机能停止,劣化等不良影响。浪涌发生的原因有自然现象的雷引起的浪涌(直接雷击浪涌,雷诱导浪涌),电气回路系统的过渡现象引起的开关浪涌,故障等引起的过压等
励磁系统 (field system)	直流电机或者交流电机作为电动机或者发电机使用的时候产生磁场的装置。一般用永磁体或者电磁体。一般来说励磁系统是用永磁体或者直流电激发的静磁场,也有用交流电产生的励磁系统
联轴器 (coupling)	电机轴和负荷轴的连接装置。在电机轴和负荷轴有偏差的时候,能够一定程度地吸收其偏差

## M

名 词	解 释
脉冲采集 (pulse collection)	在交流伺服驱动器中输入的指令脉冲和依照电机的转动量从交流伺服电机内藏的编码器输出的反馈脉冲的差
目标代码 (object code)	编译器在处理源代码的过程中产生的中间码。目标文件的内容是二进制的,是已经被机器解析过的代码。通过多个目标文件的链接,最终生成执行文件或者是库文件

## N

名 词	解 释
内插法 (interpolation)	计算 $\sin$ (正弦), $\cos$ (余弦) $^2$ 相信号的 $\text{atan}$ (反正切)值,将转动角细致分割的方法
逆变器 (inverter)	将直流转换成交流电源电路,也指内置以上电路的电力转换装置。在 FA 领域中也指可以控制任意旋转速度的感应电机的控制器称为逆变器

## P

名 词	解 释
偏移修正 (offset adjust)	修正偏移误差。偏移误差是最小尺度(通常为零)的误差。比如说,实际上虽然没有电流,但是检测电路却检测出了有电流

## Q

名 词	解 释
全闭环伺服 (full closed servo)	一般的伺服控制中,是利用电机轴中的编码器来进行反馈控制的,这被称为半闭环伺服,这种方式会产生执行机构的误差。执行机构的滚轴部分中用线性编码器来进行反馈控制的方式称为全闭环伺服,被应用在高精度的位置控制系统中

## S

名 词	解 释
闪存 (flash memory)	可以重写,在电源切断的时候数据也不会消失的不挥发性半导体存储器,被称为闪存 EEPROM 或者闪存 ROM。虽然是 EEPROM 的一种,但是与之前的 EEPROM 不同的是不能只重写一位的数据,需要预先将整个单位模块的数据消去以后再重新写入数据。为了消去和写入数据需要有作为另外的电源 $V_{pp}$ ,或者用单一电源驱动。最近使用 3.3V 单一电源的情况比较多
失步 (step out)	虽然步进电机是和脉冲同步转动的,但是由于剧烈的速度变化或者负荷过载的时候也会变得不同步。这种和输入脉冲不同步的状态称为失步状态
衰减系数 (dumping factor)	在电子工程领域表示共振强度的 $Q$ 值倒数的 $1/2$ 倍的值,关系表达式为 $Q=1/(2Z_n)$
双极型三极管 (bipolar transistor)	也可以称为接合型三极管,是三极管的一种。因为其与场效应型三极管(单极性三极管)不同,有两种载子,从而称为双极型。由于最初发明的三极管是双极型三极管,所以单说三极管常常指的是双极型三极管
死区时间 (dead time)	高压侧功率器件(MOSFET 或者 IGBT)和低压侧功率器件同时关闭的时间。为了防止高压侧功率器件和低压侧功率器同时开启的时候会产生直通电流而设置的
伺服电机 (servo motor)	指的是拥有能够按照指定的命令动作的控制,可以完成迅速高精度动作的电机
伺服放大器 (servo amplifier)	拥有伺服电机的驱动电路和伺服系统的伺服控制装置。也有制造商称其为伺服单元
伺服锁定 (servo lock)	与伺服电机的反馈有关,保持位置或者速度不变的状态
伺服系统 (servomechanism)	伺服系统是源于拉丁语中表示“奴隶”意思的“servus”,是能够按照指定的命令动作的控制系统

## T

名 词	解 释
同步电机 (synchronous motor)	和转子(线圈)的交流电流同步转动的电机。交流伺服电机是同步电机的一种

## W

名 词	解 释
微距驱动 (microstep drive)	在步进电机中,阶段性地控制驱动电流,从电气方面使得转动角精细地运动的驱动方法。不需要从机械角度就可以将转动角分割得更精细,还能有抑制振动的效果
位置/速度环控制 (cascade control)	也称为级联控制。在位置环的副回路中有速度环,是多重反馈环路控制。大部分的电机控制都采用这种控制方式
无刷直流电机 (brushless DC motor)	无电刷和换向器,采用非接触式的整流装置的电机。与转子是永磁体,定子是线圈的直流电机构造不同
无线电控制伺服系统 (radio control servo)	遥控汽车的手柄操作等中使用的伺服电机,位置检测使用的是电位计(可变电阻)
无心直流电机 (coreless DC motor)	定子只有胶膜塑封好了的铜线圈,转子采用永磁铁的直流电机。为了强调转子中没有铁心而称为无心电机

## X

名 词	解 释
现代控制理论 (modern control theory)	建立在状态空间法基础上的一种控制理论,是自动控制理论的一个主要组成部分。在现代控制理论中,对控制系统的分析和设计主要是通过系统的状态变量的描述来进行的,基本方法是时间域方法
线性编码器 (linear encoder)	检测直线方向移动量的位置传感器。被用在线性电机或者是全闭环伺服系统的位置检测中
线性导轨 (linear stage)	用于直线往复运动场合,拥有比直线轴承更高的额定负载,同时可以承担一定的转矩,可在高负载的情况下实现高精度的直线运动
线性放大器 (linear amplifier)	在电机驱动中,基本上采用的是 PWM 方式,线性放大器只用在特殊的用途中。主要用在 VCM(Voice Coil Motor)的精密位置控制或者是空气轴承中采用的主轴电机的高精度转动控制方面
线性区 (linear region)	源漏间电压不高的时候,MOSFET 的导通电阻基本恒定,这个状态称为线性区。MOSFET 的动作和双极性三极管比较的情况下,双极性的线性区及饱和区正好和 MOSFET 相反

陷波滤波器 (notch filter)	使电气信号在特定的频率成分衰减的滤波电路,也称为带阻滤波器
向量控制 (vector control)	改善交流电机的效率的电流控制方法。将电机电流分成 $d$ (励磁)成分和 $q$ (转矩)成分,分别进行控制的方法
续流二极管 (freewheel diode)	在电路中反向并联在继电器或电感线圈的两端,当电感线圈断电时其两端的电动势并不立即消失,此时残余电动势通过一个二极管释放,起这种作用的二极管就是续流二极管。在由 MOSFET 构成的驱动电路的情况下,将 MOSFET 的体二极管作为续流二极管的情况很多
旋转变压器 (resolver)	旋转变压器的原理类似于变压器。旋转变压器转子与电机轴相连,在一次侧(励磁端)加上交流电压,旋转变压器的转子转动的时候,在相差 $90^\circ$ 设置的二次侧(输出端)可以得到正弦和余弦信号。计算出正弦与余弦信号的反正切函数的话,便可以知道转动角
旋转磁场 (revolving field)	转子形成的磁场。因为是产生电机旋转力的原动力,所以这样命名

## Y

名 词	解 释
移动平均线 (moving average)	连续计算一定个数的数据的平均值,解析数据全体变化的倾向,在股票分析中经常被用到
异步电机 (asynchronous motor)	转子比定子的线圈的交流电流的频率低若干频率转动的电机。这被称为偏移。具有代表性的电机是感应电机,在冰箱和空调等中使用较多
异步通信方式 (asynchronous)	通信协议的一种,也称为非同步通信方式。这种方式是,在每一个字符的开始和结束的地方加上开始位和停止位进行传送,以便接收端能正确接收每一个字符
预 畸 (prewarping)	从模拟滤波器变换到数字滤波器的时候,会产生频率的偏移,预畸是预先将频率偏移修正的技术

## Z

名 词	解 释
增量计数器 (incremental counter)	Incremental 是增加的意思,Counter 是通过积累时钟脉冲数来进行数值处理的逻辑电路(数值电路)。在电机控制中,经常指的是记录增量式编码器的脉冲输出的电路

增量式编码器 (incremental encoder)	检测相对位置的装置。在旋转的码盘上有多个光栅,利用发光二极管发出的光通过固定的缝隙,通过感光三极管检测出的光,将缝隙的位置转换成电气信号的装置。旋转的码盘在转动的时候,可以得到连续的脉冲序列
增益修正 (gain adjust)	修正增益误差。增益误差主要是斜率变化引起的全尺寸误差。比如说,在电流检测电路中,电流检测电阻值混乱的话,斜率就会变化,就会产生增益误差
整数运算 (integer operation)	CPU 功能的一种。使用整数进行处理运算。是 CPU 最基本的功能的一种。除此之外,CPU 还可以进行浮点运算。另外定点运算从广义上来说也可以算是整数运算
正交坐标系 (rectangular coordinate system)	所有的坐标轴都互相正交的坐标系。正交坐标系中,所有的点都有唯一的坐标
正弦波驱动 (sine wave drive)	交流电机驱动方法的名称。由于交流电机的驱动波形是正弦波,所以称为正弦波驱动。虽然交流电机在 PWM 驱动的情况下,驱动电压为矩形,但是由于电机的电流为正弦波状,所以也称为正弦波 PWM 驱动
直流电机 (DC motor)	用直流电源驱动的电机电。分有刷直流电机和无刷直流电机
直线导轨 (linear guide)	精密地引导直线运动的机械构件。多数由导轨和滚珠轴承构成,也有称为 LM 导轨的
指数函数 (exponential)	将指数的幂作为变量,其定义域主要扩展到实数全体的初等超越函数的一种。是对数函数的反函数
主轴电机 (spindle motor)	主要以旋转控制为目的的电机的名称。经常被作为驱动 CD 或者硬盘装置的盘面转动的电机
转动惯量 (moment of inertia)	相当于直线运动系统的质量,用来表示转动系统的惯性的量。转动惯量的单位为 $N \cdot m^2$
转矩 (torque)	在离转轴半径 1m 的位置产生的力定为转矩。在国际单位系统(SI)中,转矩的单位是 $N \cdot m$ 。在这里 N 是牛顿,表示力的单位
转矩波动 (torque ripple)	在极低速阶段,额定电流时输出转矩的变动量和平均转矩的比率
转矩控制 (torque control)	控制电机产生任意指定转矩的控制方式。一般来说电机产生的转矩和电机电流是成比例关系的,所以一般先检测出电机电流再进行转矩控制

转 子 (rotor)	电机的转动机构部分(转子),有刷直流电机中是线圈,无刷电机中是永磁体
转子转动惯量 (rotor inertia moment)	为了达到伺服电机快速高精度的动作要求,需要将电机的转动惯量设计得尽量小

其 他

名 词	解 释
4 象限动作 (operating 4 quadrant)	电机的转动方向(正转,反转)和所产生的转矩(正方向,反方向)的组合有四种动作的模式,实现这四种动作模式的运转称为 4 象限动作
$\Delta$ 连接 (delta connection)	在电机线圈的连接方法中,线圈端分别邻接而成的接线方法。三相线圈的情况下看起来就像 $\Delta$ ,因此得名
$\alpha\beta$ 变换 ( $\alpha\beta$ axis transform)	在向量控制的坐标转换过程中,直接用三相交流比较困难,需要变换成两相交流,这种变换称为 $\alpha\beta$ 变换
CPLD (Complex Programmable Logic Device)	与 PAL 相当的逻辑模块,用多组开关矩阵排列组成的构造
DMA (Direct Memory Access)	不用 CPU 介入,直接在存储器与存储器,或者是存储器和 I/O 设备间通信,或者是指向硬件单元
DSP (Digital Signal Processor)	为了实现高速处理数字信号而发明的专门的大规模集成电路。内部有很多专门用于计算加法和乘法的硬件(高速乘法器、柱式位移器等),可以进行流程化处理和并列处理
$d$ - $q$ 变换 ( $d$ - $q$ axis transform)	在向量控制的情况下,将电机电流分成 $d$ (励磁)成分和 $q$ (转矩)成分,分别进行反馈控制。分离的过程称为 $d$ - $q$ 变换
FA (Factory Automation)	利用基于电脑控制的技术,用组装机器人、控制机器人实现工厂的自动化。另外也包含用计算机实现的工厂的在库管理、生产管理等业务方面的自动化
FET (Field Effect Transistor)	场效应晶体管的简称,是通过在栅极上加电压,用沟道电场在电子或者是空穴的流动中加设阀门的原理而制成的能够控制源极-漏极间的电流的三极管。因为只用了一种载流子,也称为单级三极管
FIR (Finite Impulse Response)	数字滤波器的一种,由于没有反馈环,所以是不发散的稳定的滤波器。为了使得 FIR 滤波器具有相同的振幅特性,相对于 IIR 滤波器来说,计算量比较大



<b>FPGA</b> (Field Programmable Gate Array)	主要用于由大规模电路组成的场合。FPGA 是由小的逻辑模块组合而成,在芯片内部可以自由连接各个配线区域,从而实现非常高自由度的大规模高速电路
<b>H 桥电路</b> (H bridge circuit)	直流电机用的由 4 个有源器件组成的驱动电路。电路的构成和“H”有点像。具有用单电源就能驱动直流电机正转或反转的特性
<b>IEEE</b> (the Institute of Electrical and Electronics Engineers)	本部在美国的电气电子技术学会。是非盈利的专门机构,在美国是最大的团体。在 1963 年由美国电气学会和无线学会合并组成。有多个学科分会,主要的活动内容为书籍的发行和标准化(规格的指定)等
<b>IGBT</b> (Insulated Gate Bipolar Transistor)	绝缘栅双极型晶体管的简称。IGBT 是在栅极加入 MOSFET 的双极型晶体管,用 MOSFET 来控制晶体管。MOSFET 具有高速动作的特性,驱动功率小,双极型晶体管的阻抗低
<b>IIR</b> (Infinite Impulse Response)	无限脉冲响应,数字滤波器的一种。具有反馈环,系数是发散的。为了使得 IIR 滤波器具有相同的振幅特性,相对于 FIR 滤波器来说,具有计算量较少的特点
<b>LUT</b> (Look Up Table)	在计算机中,以查看效率或者是进行变换为目的所设计的数组或者关联数组的数据结构
<b>MATLAB</b> (Matlab)	美国的 MathWorks 所开发的数值分析软件。在日本是由 Cybernet 公司进行销售的
<b>MOSFET</b> (Metal Oxide Semiconductor Field Effect Transistor)	金属氧化物半导体场效应晶体管的简称。MOS(金属-氧化物-半导体三种物质组成)组成的具有栅极的场效应晶体管。基于栅电压来控制源极-漏极间的电流,实现电气信号的放大和开关动作。其特点是动作的时候相对较小的耗电量,可以高密度集成。虽然不能达到双极型那么快,但是如果将器件缩小化的话,可以实现高速化。现在将其微小化以期待作为超大型集成电路的基本器件的研究非常盛行
<b>NMI</b> (Non Maskable Interrupt)	非可屏蔽中断的简称。用于在计算机中不可屏蔽的中断处理
<b>NMOS</b> (N channel Metal Oxide Semiconductor)	N 沟道金属氧化物半导体场效应晶体管的简称。NMOS 的载流子是电子,因此与同样大小的 PMOS 相比,有其 3 倍的电流容量
<b>PAL</b> (Programmable Array Logic)	与阵列可以被改写,而或阵列固定的器件。是小规模可编程逻辑器件

<b>PID 控制</b> (PID control)	PID 控制是反馈控制的一种,控制的输入值是基于输出值和目标值的偏差、积分及微分三要素而进行控制的方法。虽然作为控制理论一个分支的古典控制理论已经有很长的体系化的历史,但是作为反馈控制的基础,各种各样的控制方法还在持续被开发和提出来。直到现在,仍然作为产业界最主要的控制手法
<b>PLL</b> (Phase Locked Loop)	可以调节器件频率的电子回路。可以实现输出信号的频率和相位与输入端所设定的标准频率和输入信号一致。在电机控制中用于高精度转动控制
<b>PMOS</b> (P channel Metal Oxide Semiconductor)	P 沟道金属氧化物半导体场效应晶体管的简称。PMOS 的载流子是带正电的空穴。要达到与 NMOS 相同的电流容量需要大概 3 倍的大小,因此栅容量很大
<b>PM 型步进电机</b> (Permanent Magnet type stepping motor)	根据步进电机的电磁构造,大致可以分为 VR 型、PM 型、HB 型。PM 型是在转子的多极中装入永磁体的电机,被称为永磁型。步距角较大
<b>PWM</b> (Pulse Width Modulation)	变频方法的一种。通过变化脉冲波的占空比来进行变频。PWM 方式的放大用三极管的饱和动作(在 MOSFET 中为线性动作)来实现,放大端的损失比较小
<b>RAM</b> (Random Access Memory)	随机存储器的英文简写。是在计算机中使用的一种存储装置
<b>s-z 变换</b> (s-z transform)	从连续传递函数(模拟)转换成离散传递函数(数字)的变换。 $1/z$ 表示一个采样的延迟
<b>VHDL</b> (VHSIC Hardware Description Language)	是在数字电路设计中用的硬件描述语言(HDL)的一种。作为在 EDA 领域的一个标准,经常在 FPGA、ASIC 等设计中被用到
<b>Y(星形)连接</b> (star connection)	在电机线圈的连接方法中,将每个线圈的一端连接于一点,另外一端向外引出的接线方法。在三相线圈的情况下是 Y 字形的,因为与星形相像而得名