

无感无刷直流电机之 电调设计全攻略

前 言	1
1. 无刷直流电机基础知识.....	2
1.1 三个基本定则.....	2
1. 左手定则.....	2
2. 右手定则（安培定则一）	3
3. 右手螺旋定则（安培定则二）	3
1.2 内转子无刷直流电机的工作原理.....	3
1. 磁回路分析法.....	4
2. 三相二极内转子电机结构.....	5
3. 三相多绕组多极内转子电机的结构.....	7
1.3 外转子无刷直流电机的工作原理.....	8
1. 一般外转子无刷直流电机的结构	8
2. 新西达 2212 外转子电机的结构	8
1.4 无刷直流电机转矩的理论分析.....	14
1. 传统的无刷电机绕组结构.....	14
2. 转子磁场的分布情况.....	15
3. 转子的受力分析.....	16
4. 一种近似分析模型.....	18
1.5 换相与调速.....	19
1. 换相基本原理.....	19
2. 新西达 2212 电机的换相分析.....	24
3. 调速.....	28
2. 无感无刷电调的驱动电路设计.....	30
2.1 电池电压监测电路.....	30
2.2 换相控制电路.....	30
1. 六臂全桥驱动电路原理.....	31
2. 功率场效应管的选择.....	33
2.3 电流检测电路.....	45
2.4 反电势过零检测电路.....	49
2.5 制作你自己的电调线路板.....	50
3. 无感无刷电调的软件设计.....	52
3.1 电流检测.....	52
3.2 定时器延时与PWM信号	53
1. 定时器初始化.....	54
2. 定时器T0 的溢出中断服务程序	54
3. 利用T0 延时（毫秒级）	54
4. 利用T0 延时（微秒级）	55
5. PWM信号的产生	55
3.3 过零事件检测与电机换相.....	56
1. BLMC.h中定义的宏	56
2. 过零检测与换相代码分析.....	59

3.4 启动算法.....	63
1. 函数Anwerfen启动流程分析.....	63
2. 启动算法机理探究.....	65
3.5 上电时的MOSFET自检.....	68
1. 函数Delay和DelayM.....	68
2. 函数MotorTon自检流程分析.....	68
3.6 让你的电机演奏音乐.....	70
3.7 通信模块.....	72
1. PPM解码.....	72
2. TWI总线通信.....	74
3. 串口通信.....	74
4. 指令的收入函数SollwertErmittlung.....	75
4. 德国MicroKopter项目BL-Ctrl电调程序主程序代码流程分析（V0.41 版本）.....	77
5.1 全局变量列表.....	78
5.2 main主函数流程分析.....	80
1. 进入while(1)前的准备工作.....	80
2. while(1)主循环内容分析.....	81
5. 高级话题.....	86
5.1 电机的控制模型.....	86
5.2 四轴上的校正策略.....	87
附录一.....	88
附录二.....	89
附录三.....	93
附录四.....	94

前言

关注开源四轴项目也有近一年了，前期都以潜水为主，业余时间主要是在啃那些控制和导航的理论书籍。最近开始动手做了，打算先从电调开始，发现真要做起来问题还真是一大堆。所幸有论坛这么好一个交流平台，很多问题其实前人都已经碰到过了，参考前人的经验，让我少走了很多弯路。在此要感谢论坛各位前辈大侠和阿莫的 ourdev。:-)

前人种树、后人乘凉，既然受惠于前人，怎好意思独享，当然也应该帮助一下新入门的开发者。由于四轴分论坛的帖子数量已经很多了，光搜一下无刷电机和电调也有近百来篇帖子，次序和深浅程度不一，想要看完并完全理解这些帖子对新人来说不啻是一个艰巨的任务。而且很多帖子的发帖时间都比较久远了，回帖提问也未必能得到原作者的回答。我写这篇文档的目的，就在于做一个整理和汇编，把很多零散的、前人已解答过的问题分门别类整理出来，并添加一些自己制作电调时的经验和总结。

在参考一些关于无刷电机驱动的书籍和帖子的时候，发现高手或是大师好像都比较惜字如金，一些问题往往点到为止或者一笔带过，有些看似简单的问题会让像我这样的电调DIYer困惑很久。所以在本文行文时，笔者力图把问题以大白话的形式说明白，如果各位有觉得哪里看得不明不白的，可以回帖提出（时限一个月，呵呵），我会修改文档以试图将问题讲清楚。如果有些问题我无法回答，我会老老实实跟你说我也没搞清楚，还要请高手来解惑啊。如果发现我哪些内容讲错了，也请不吝指正。

最后还将附上德国 MK 项目电调代码（V0.41 版本）的全代码分析，这件事可能以前没人做过吧，我就来揭晓一下答案好了^^。同时我也参照他的程序，自己写了一个可供 mega8 和 mega32 使用的电调驱动程序，将一些结构作了优化，所有变量名都从德语改成了英语，添加了比较完备的中文注释，通讯规约也做了一些整理和改动，并附带上位机调试程序。也希望大家能多多把自己的一些心得体会和经验拿出来，建立好一个基础的知识平台后，可以让后来的开发者少走很多初期摸索的弯路，而专心于攻克我们未能解决的难点。衷心希望后来的开发者能站在我们的肩膀上，走得比我们更远。

timegate 墨鸢

2010 年 7 月

1. 无刷直流电机基础知识

关于无刷直流电机的驱动的基本原理，很多教材和文档都已经讲得很清楚了，特别是坛上网友提供的：《无刷直流（BLDC）电机基础》（MicroChip 公司，编号 AN885）、《Brushless DC Motors Made Easy》（Freescall 公司，编号 PZ104）和 Atmel 公司的编号为：AVR194、AVR491、AVR492 的几篇文档，都写得很不错，深入浅出，很适合入门的初学者学习。稍后我会给出它们的下载链接（见附录一）。

不过一上来就让读者自己去看文档，貌似不太厚道，那我这里还是辛苦一下，把各篇文档的精华部分抽取出来，重新组织一下，给大家一个关于无刷电机的比较概要的认识。

1.1 三个基本定则

首先要搞清楚一件基本的事情：我们只是来搞电调的，而不是去设计电机的。所以不要被一些无刷电机教材一上来那些林林总总的关于什么磁路、磁导率、气隙饱和、去磁曲线等基础知识给吓倒，那些东西是给设计电机的人看的，对我们这种仅仅以弄出一个电调为目标的人来讲，意义不大（不过你如果打算以此为职业的话，这些东西还是建议深入学习一下的）。

对于入门开发者来说，只需要记牢三个基本定则：左手定则，右手定则，右手螺旋定则。

1. 左手定则

位于磁场中的载流导体，会受到力的作用，力的方向可按左手定则确定，如右图所示：伸开左手，使大拇指和其余四指垂直，把手心面向 N 极，四指顺着电流的方向，那么大拇指所指方向就是载流导体在磁场中的受力方向。

力的大小为： $F = BIL \sin \theta$

其中： B 为磁感应强度（单位 T）， I 为电流大小（单位 A）， L 为导体有效长度（单位 m）， F 为力的大

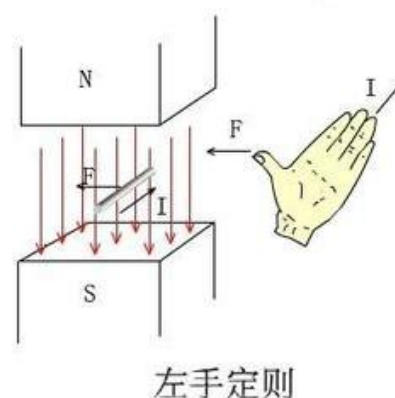


图 1-1 左手定则

小（单位 N）， θ 为： B 和 I 的夹角。

2. 右手定则（安培定则一）

在磁场中运动的导体因切割磁力线会感生出电动势 E ，其示意图见图：

$$E = vBL \sin \theta$$

其中： v 为导体的运动速度（单位 m/s）， B 为磁感应强度（单位 T）， L 为导体长度（单位 m）， θ 为： B 和 L 的夹角。

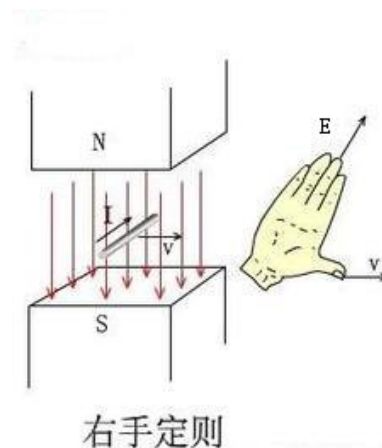


图 1-2 右手定则

3. 右手螺旋定则（安培定则二）

用右手握住通电螺线管，使四指弯曲与电流方向一致，那么大拇指所指的那一端就是通电螺线管的 N 极。



图 1-3 右手螺旋定则

1.2 内转子无刷直流电机的工作原理

一般的教材或是文档，介绍的多半都是内转子无刷电机的工作原理。按理说，资料已经这么多了，学习起来不应该有什么困难，其实不然。以笔者亲身经历，无刷电机的资料看得多了，反而会产生困惑。究其原因，是因为它们分别采用了两种不同的方法进行描述，同样是比较简单的三相二极无刷电机，这两种描述方法所采用的绕组结构其实是不太一样的。

1. 磁回路分析法

在 MicroChip, Freescale 和 Atmel 三家公司的文档中,都不约而同地采用了这种方法来说明无刷电机的工作原理,其原理说明见图 1-4:

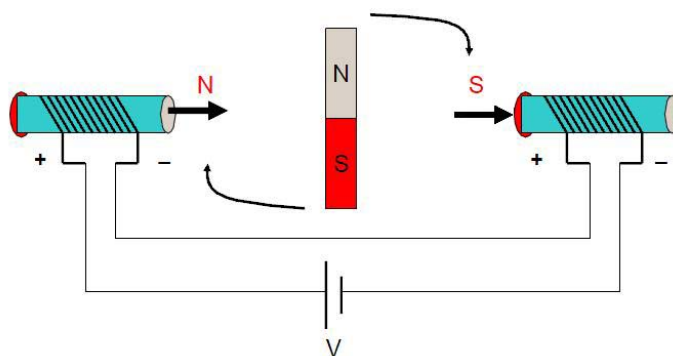


图 1-4 （摘自 Freescale PZ104 文档）

在图 1-4 中,当两头的线圈通上电流时,根据右手螺旋定则,会产生方向指向右的外加磁感应强度 B (如粗箭头方向所示),而中间的转子会尽量使自己内部的磁力线方向与外磁力线方向保持一致,以形成一个最短闭合磁力线回路,这样内转子就会按顺时针方向旋转了。

顺便提一句,有网友曾经提到说不太理解这句话的含义:“当转子磁场方向与外部磁场方向垂直时,转子所受的转动力矩最大”。注意这里说的是“力矩”最大,而不是“力”最大。诚然,在转子磁场与外部磁场方向一致时,转子所受磁力最大,但此时转子呈水平状态,力臂为 0,当然也就不会转动了。

当转子转到水平位置时,虽然不再受到转动力矩的作用,但由于惯性原因,还会继续顺时针转动,这时若改变两头螺线管的电流方向,如下图所示,转子就会继续顺时针向前转动,见图 1-5 所示:

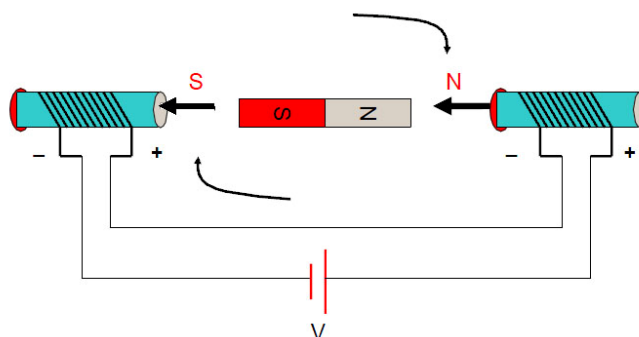


图 1-5 （摘自 Freescale PZ104 文档）

如此不断改变两头螺线管的电流方向,内转子就会不停转起来了。改变电流方向的这一动作,就叫做**换相** (commutation)。注意:何时换相只与转子的位置有关,而与转速无关。

这一点是初学者比较容易混淆的概念，应当注意。

以上是最简单的两相两级无刷电机的工作原理，仅仅用来说明概念用，下面我们来看比较普遍的三相两极无刷电机的构造。

2. 三相二极内转子电机结构

一般来说，定子的三相绕组有星形联结方式和三角联结方式，而“三相星形联结的二二导通方式”最为常用，故这里只对这种情况作详细分析。

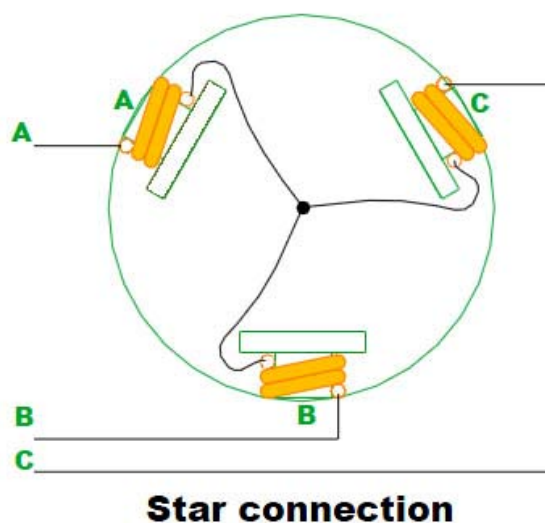


图 1-6 （修改自 Freescale PZ104 文档）

图 1-6 显示了定子绕组的联结方式（转子未画出），三个绕组通过中心的连接点以“Y”型的方式被联结在一起。整个电机就引出三根线 A, B, C。当它们之间两两通电时，有 6 种情况，分别是 AB, AC, BC, BA, CA, CB，图 1-7(a)~(f)分别描述了这 6 种情况下每个通电线圈产生的磁感应强度的方向（红、兰色表示）和两个线圈的合成磁感应强度方向（绿色表示）。

在图(a)中，AB 相通电，中间的转子（图中未画出）会尽量往绿色箭头方向对齐，当转子到达图(a)中绿色箭头位置时，外线圈换相，改成 AC 相通电，这时转子会继续运动，并尽量往图(b)中的绿色箭头处对齐，当转子到达图(b)中箭头位置时，外线圈再次换相，改成 BC 相通电，再往后以此类推。当外线圈完成 6 次换相后，内转子正好旋转一周（即 360° ）。**再次重申一下：**何时换相只与转子位置有关，而与转速无关。

图 1-8 中画出了换相前和换相后合成磁场方向的比较与转子位置的变化。一般来说，换相时，转子应该处于，比与新的合成磁力线方向垂直的位置不到一点的钝角位置，这样可以使产生最大的转矩的垂直位置正好处于本次通电的中间时刻。

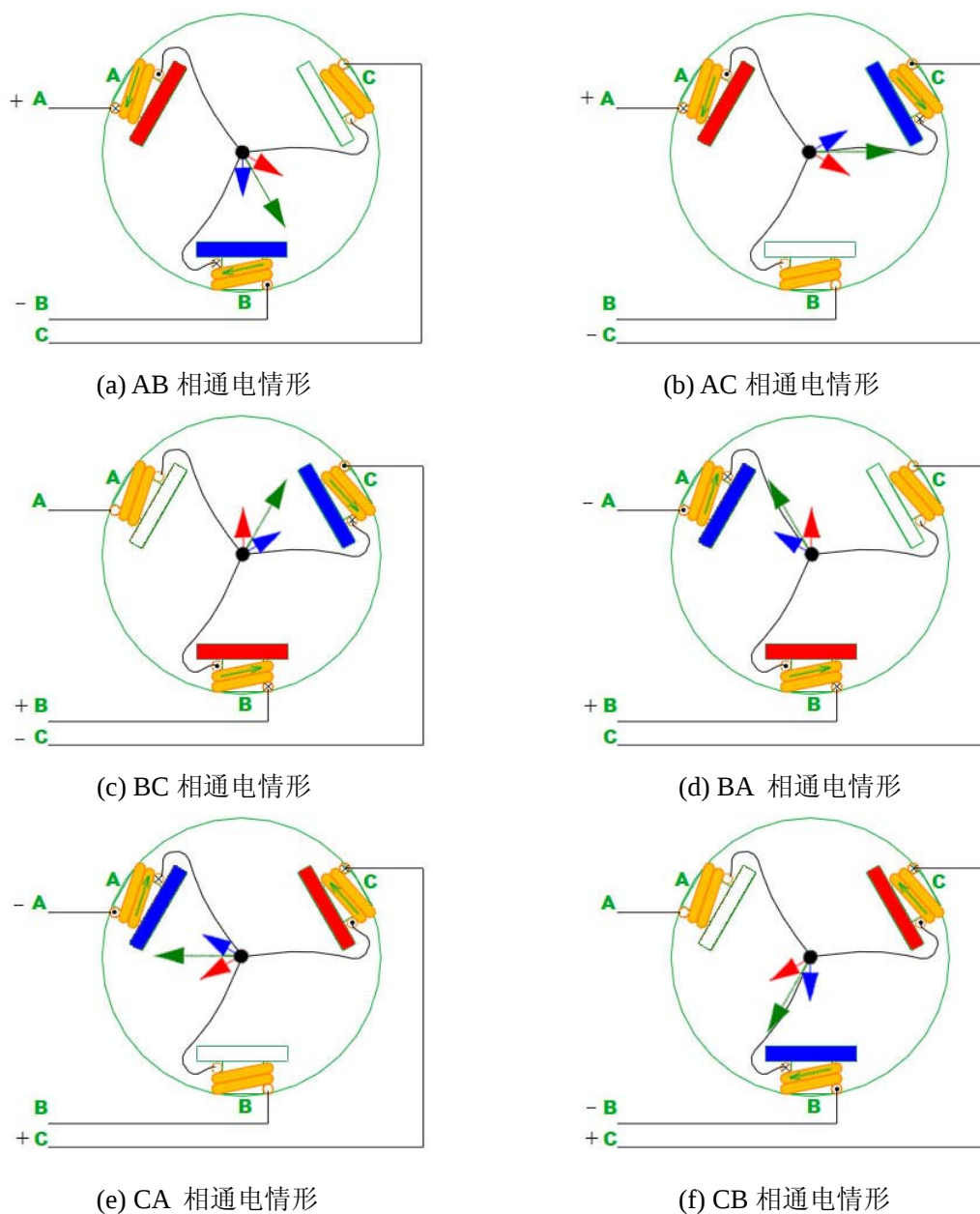


图 1-7 星形绕组两两通电的 6 种情形

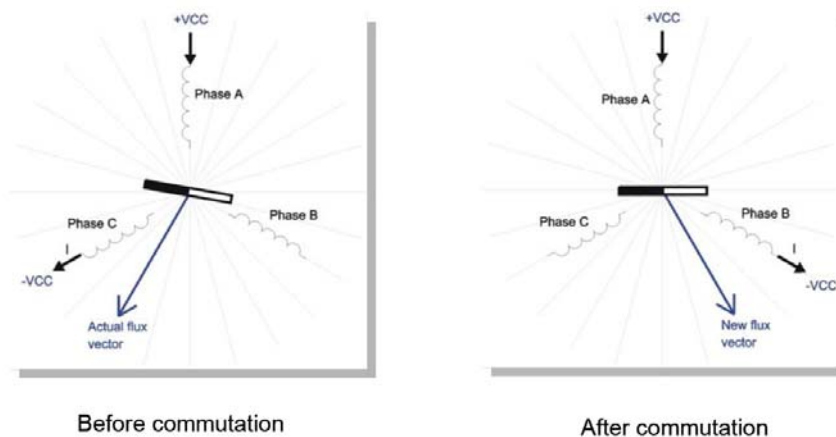


图 1-8 换相前和换相后的情形（摘自 Freescale PZ104 文档）

3. 三相多绕组多极内转子电机的结构

搞清了最简单的三相三绕组二极电机，我们再来看一个复杂点的，图 1-9(a)是一个三相九绕组六极（三对极）内转子电机，它的绕组连线方式见图 1-9(b)。从图(b)可见，其三相绕组也是在中间点连接在一起的，也属于星形联结方式。一般而言，电机的绕组数量都和永磁极的数量是不一致的（比如用 9 绕组 6 极，而不是 6 绕组 6 极），这样是为了防止定子的齿与转子的磁钢相吸而对齐，产生类似步进电机的效果，此种情况下转矩会产生很大波动。

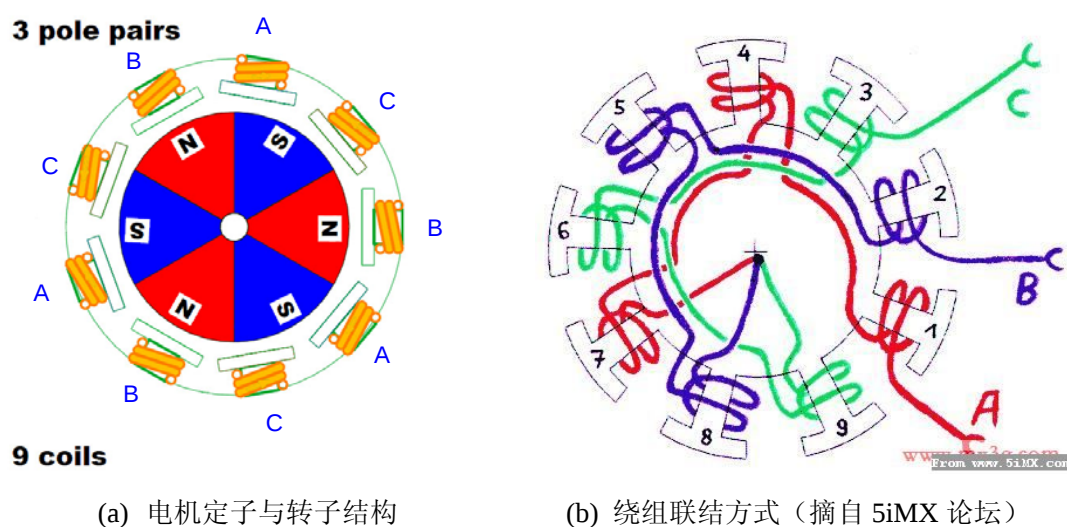


图 1-9 三相 9 绕组 3 对极内转子无刷直流电机结构

其二二导通时的 6 种通电情况读者可自行分析，原则是转子的 N 极与通电绕组的 S 极有对齐的运动趋势，而转子的 S 极与通电绕组的 N 极有对齐的运动趋势。为便于读者理解，图 1-10 给出了一个对齐的运动趋势的图例。

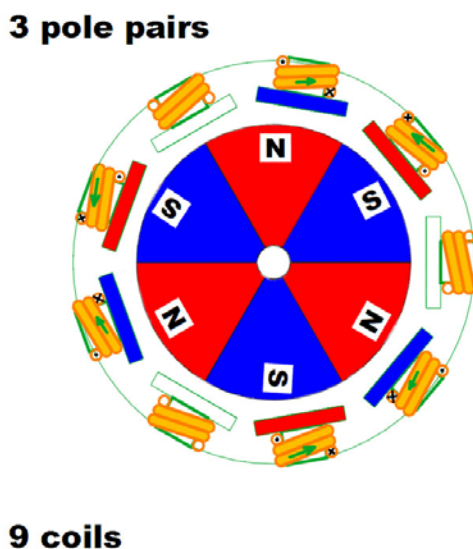


图 1-10 某 2 相通电时的转子磁极和定子磁极对齐运动的最终位置

1.3 外转子无刷直流电机的工作原理

看完了内转子无刷直流电机的结构，我们来看外转子的。其区别就在于，外转子电机将原来处于中心位置的磁钢做成一片片，贴到了外壳上，电机运行时，是整个外壳在转，而中间的线圈定子不动。外转子无刷直流电机较内转子来说，转子的转动惯量大很多（因为转子的主要质量都集中在外壳上），所以转速较内转子电机要慢，通常 KV 值在几百到几千之间，用在航模上可以直接驱动螺旋桨，而省去了机械减速机构。

噢，这里顺便解释一下 KV 值的含义，网上其实一搜一大把啦，这里为了文档的完整性，也啰嗦一下吧。无刷电机 KV 值定义为：转速/V，意思为输入电压每增加 1 伏特，无刷电机空转转速增加的转速值。比如说，标称值为 1000KV 的外转子无刷电机，在 11 伏的电压条件下，最大空载转速即为： $11 \times 1000 = 11000 \text{rpm}$ （rpm 的含义是：转/分钟）。

同系列同外形尺寸的无刷电机，根据绕线匝数的多少，会表现出不同的 KV 特性。绕线匝数多的，KV 值低，最高输出电流小，扭力大；绕线匝数少的，KV 值高，最高输出电流大，扭力小。

1. 一般外转子无刷直流电机的结构

下面是一些常见的外转子无刷电机的结构：

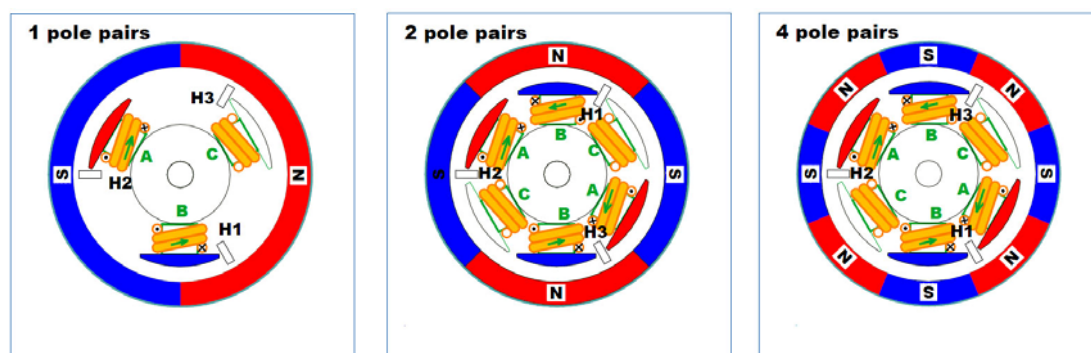


图 1-11 一些常见外转子无刷电机结构（摘自 Freescale PZ104 文档）

其分析方法也和内转子电机类似，这里再唐僧一遍吧：转子永磁体的 N 极与定子绕组的 S 极有对齐的趋势，转子永磁体的 S 极与定子绕组的 N 极有对齐的趋势。

2. 新西达 2212 外转子电机的结构

坛子里做四轴用得比较多的是新西达的 KV 值为 1000 的 XXD2212 电机。其结构为

12 绕组 14 极（即 7 对极），见图 1-12。其结构如下：定子绕组固定在底座上，转轴和外壳固定在一起形成转子，插入定子中间的轴承。由于各种资料上很少有描述 12 绕组的线圈是怎么绕的，为此笔者专门破坏性地拆解了一个 XXD2212 电机（见图 1-13），终于搞清楚了其绕组是怎么绕的，看在损失一个电机的份上，阿莫也该给个酷字，呵呵。图 1-14 画出了 XXD2212 电机的绕组绕法，跟我们想象的都不太一样，是吧？（注意圆心处三根导线是互相绝缘的，并不像普通星形方式是连在一起的）



图 1-12 XXD2212 电机结构（摘自网友 liuliu443 所发帖子）

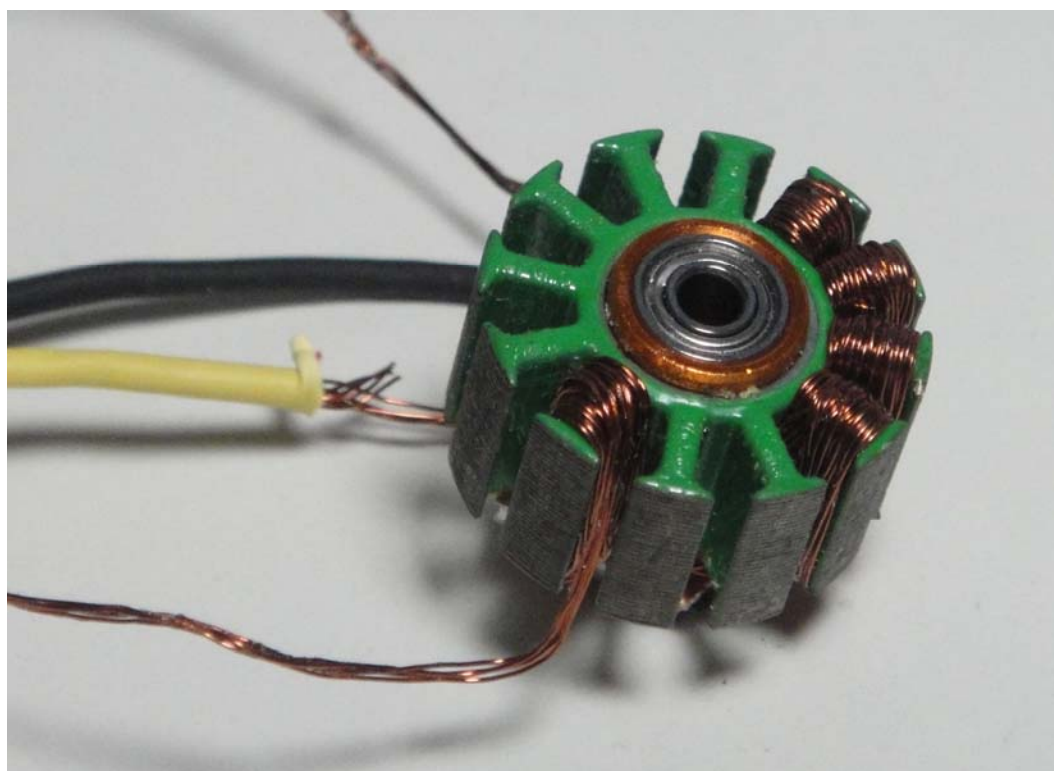


图 1-13 被笔者拆解的定子绕组

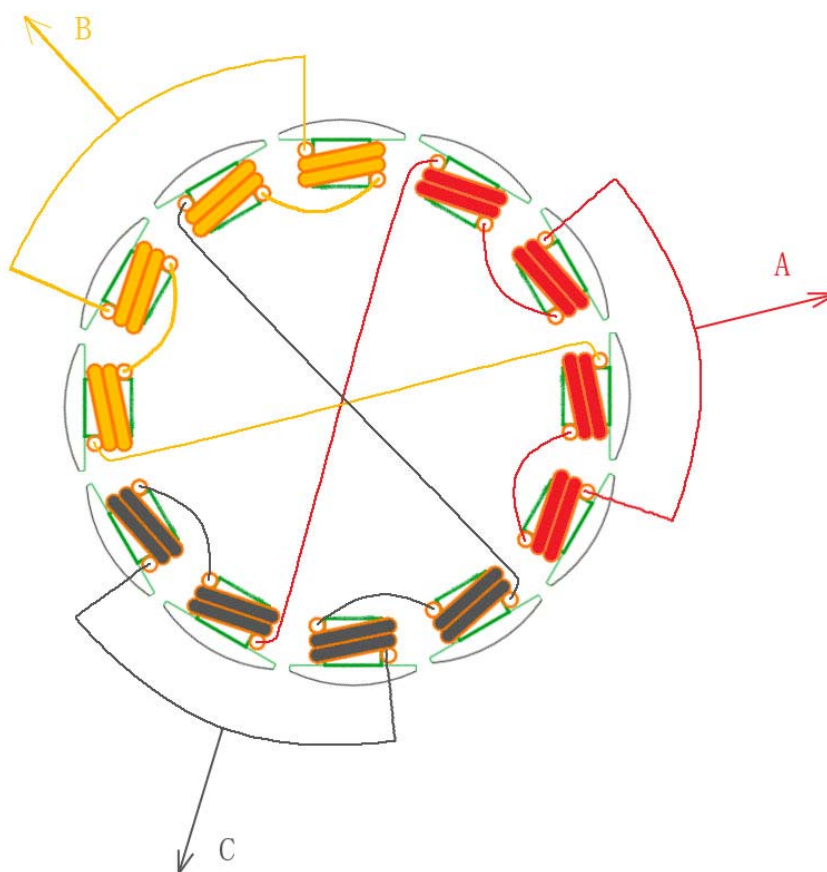
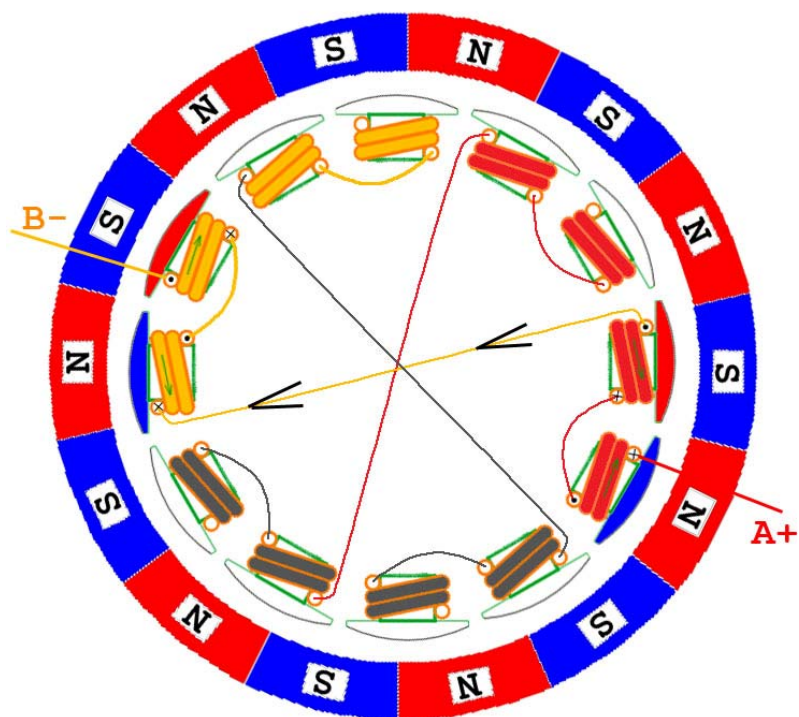
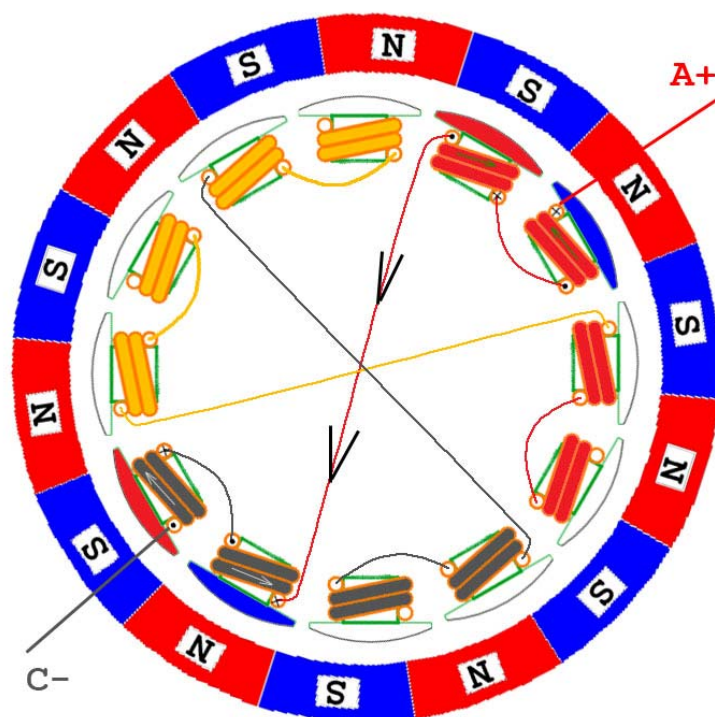


图 1-14 XXD2212 电机的绕线方式（注意圆心处三根线是互相绝缘的）

图 1-15 详细画出了 6 种两相通电的情形，可以看出，尽管绕组和磁极的数量可以有多种变化，但从电调控制的角度看，其通电次序其实是相同的，也就是说，不管外转子还是内转子电机，都遵循 $AB \rightarrow AC \rightarrow BC \rightarrow BA \rightarrow CA \rightarrow CB$ 的顺序进行通电换相。当然，如果你想让电机反转的话，可以按倒过来的次序通电：）。要说明一下的是，由于每根引出线同时接入两个绕组，所以电流是分两路走的。这里为使问题尽量简单化，下面几个图中只画出了主要一路的电流方向，还有一路电流未画出，另一路电流的具体情况放到 1.5 小节再作详细分析。

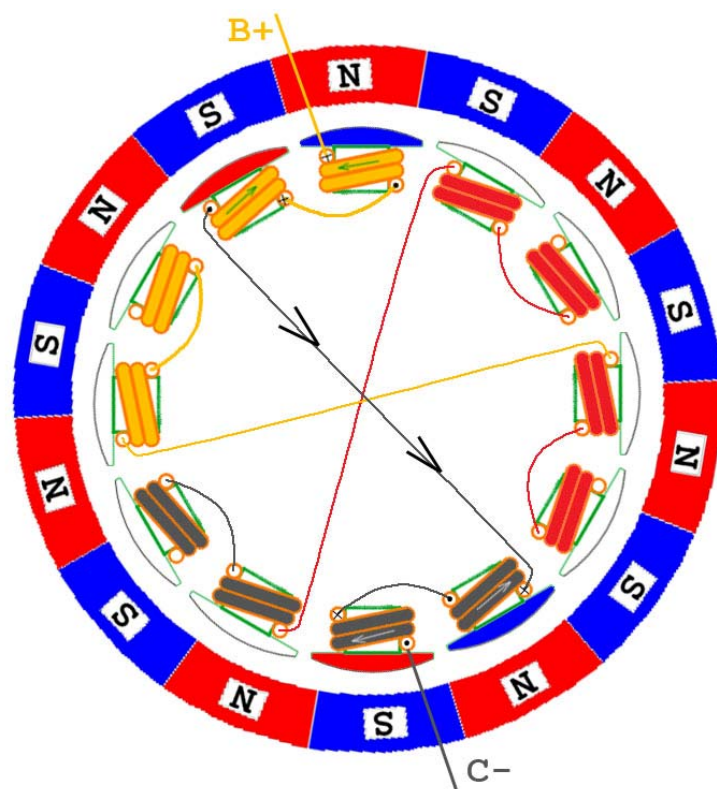


(a) AB 相通电情形

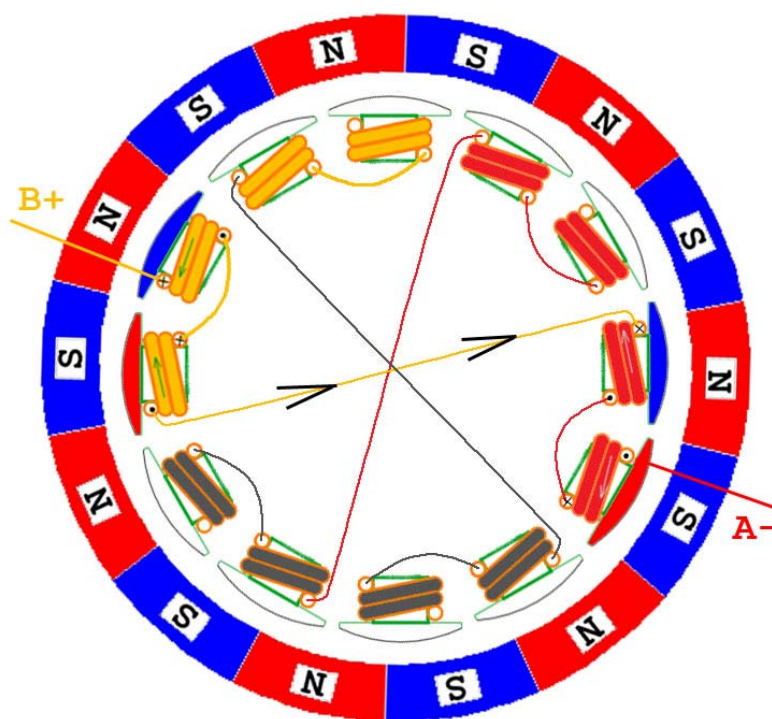


(b) AC 相通电情形

图 1-15 XXD2212 电机两两通电的 6 种情形

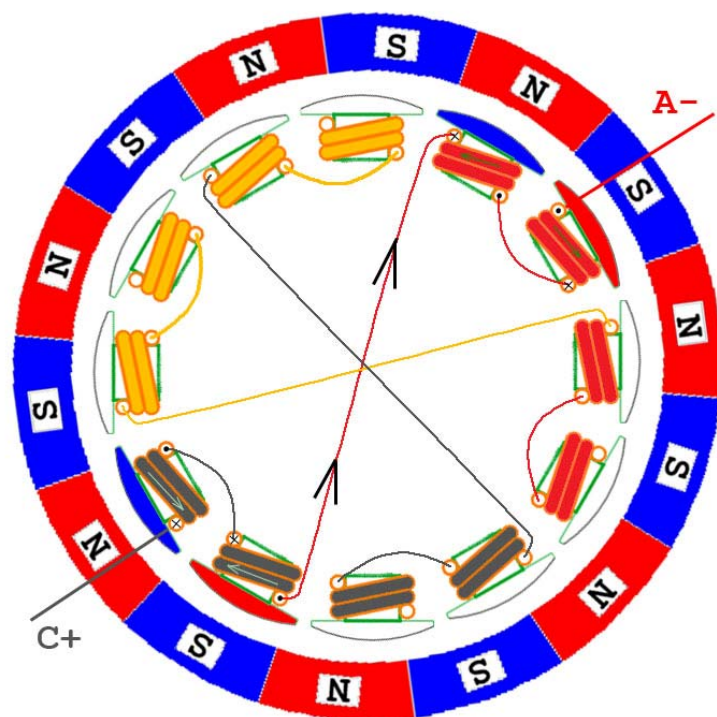


(c) BC 相通电情形

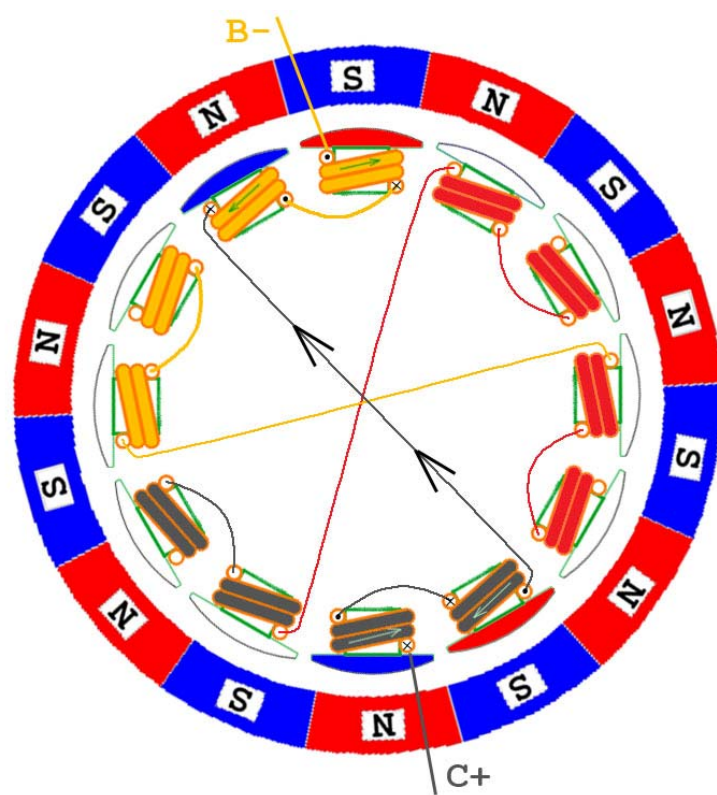


(d) BA 相通电情形

图 1-15 XXD2212 电机两两通电的 6 种情形（续）



(e) CA 相通电情形



(f) CB 相通电情形

图 1-15 XXD2212 电机两两通电的 6 种情形（续）

1.4 无刷直流电机转矩的理论分析

我们再回到最简单的三相二极内转子电机。以上的磁回路分析方法对于一般的感性认识来讲是足够了，但如果你翻阅的无刷电机的教材书够多的话，你会发现，几乎没有哪本教材是采用上面这种结构来说明无刷电机的工作原理的，这些教材中用的都是类似图 1-17 所示的结构来研究无刷电机的。究其原因，是因为上两小节示例的那种电机绕组结构，从严格上来说，并不是传统的经典的工业用无刷直流电机的结构，而是属于一种叫做“开关磁阻电机”（Switched Reluctance Motor）结构的变种（原始的开关磁阻电机的转子上是没有永磁体的）。由于它的控制方式很类似于无刷直流电机的 6 步二二导通控制方式，所以直接把它当无刷直流电机来用也没问题。真正的工业用无刷直流电机的定子绕组实物图见图 1-18。顺便说一句，笔者遍查还施水阁中关于无刷电机的典藏古今中外约十来本（80 年代的书应该算古了吧，呵呵），愣是没找到专门分析上两节那种电机结构和原理的著作，憾甚。如果哪位高人知道有相关的文献，还请指点一二，不胜感激。

1. 传统的无刷电机绕组结构

其线圈形状见图 1-16，线圈包围整个转子。电机三相绕组示意图见图 1-17。

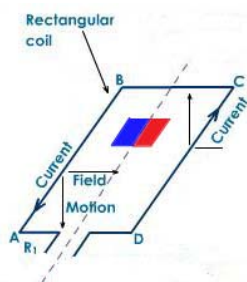


图 1-16 磁场中的线圈

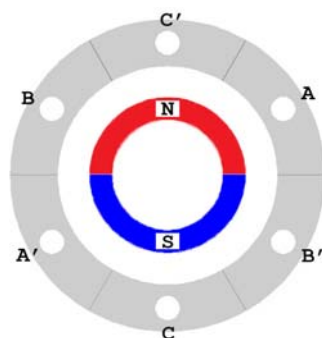


图 1-17 电机绕组和转子抽象示意图

图 1-17 中为简略示意起见，每相只画出了一个线圈，其实每相应有 N 匝线圈。其绕组联结方式为： A' 、 B' 、 C' 端通过星形联结在一起， A 、 B 、 C 为电机的三根引出线。其实物外形见图 1-18。（注意辨别图 1-18 和图 1-12 的绕组形式的区别）

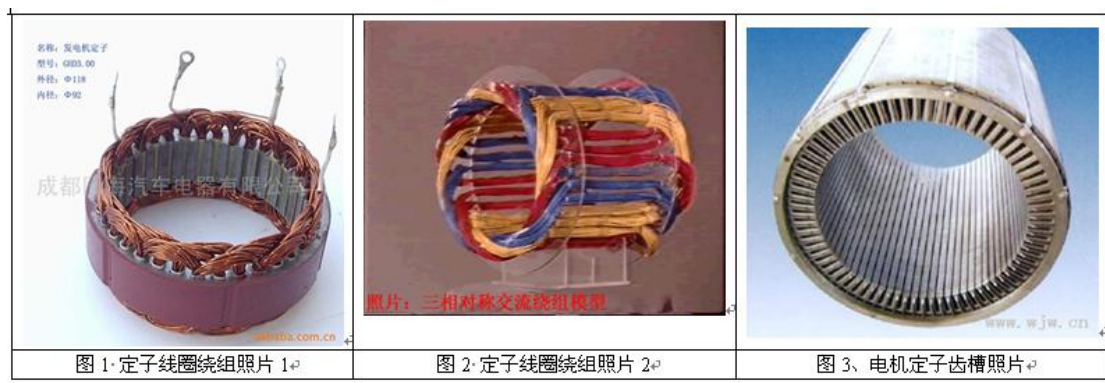


图 1-18 无刷直流电机定子绕组结构（摘自松晓的blog）

2. 转子磁场的分布情况

绕组形式变成这个样子后，就可以用“左手定则”来分析啦。不过在此之前，还要搞清楚一件事情，就是在这种绕组结构下，磁感应强度 B 的分布情况。

关于这个问题，在夏长亮的《无刷直流电机控制系统》一书中，开门见山地就讲清楚了，在此小赞一下，呵呵。现将这段论述摘抄如下：（不要问偶要电子书，没有，本书是从市立图书馆借的，各位想省点银子的也可以去图书馆借：）

“目前，国内外对无刷直流电机（Brushless DC motor, BLDCM）定义一般有两种：一种定义认为只有梯形波/方波无刷直流电机才可以被称为无刷直流电机，而正弦波无刷电机则被称为永磁同步电机（permanent magnet synchronous motor, PMSM）；另一种定义则认为梯形波/方波无刷电机和正弦波无刷电机都是无刷直流电机。迄今为止，还没有一个公认的统一标准对无刷直流电机进行准确的分类和定义。本书将采用第一种定义，把具有串励直流电机启动特性和并励直流电机调速特性的梯形波/方波无刷直流电机称为无刷直流电机。”

好了，现在来解释一下上面说的“梯形波/方波”是什么意思。图 1-19 展示了内转子磁极的磁感应强度 B 的分布情况。我们预定义磁感应强度方向向外为正，从图中可以看出，在 0° 的时候，处于正反方向交界处，磁感应强度为零，然后开始线性增加，在 A 点时达到最大，然后一直保持恒定值不变，直到 B 点开始下降，到 180° 的时候下降到零。然后开始负向增长，在 C 点处达到负值最大，然后保持恒定负值不变，直到 D 点强度开始减弱，到

0° 时又回到零。至于 A 点到底在几度的位置，不同的电机不一样。如果 A 非常接近 0° 的位置，上升和下降直线就会非常陡峭，“梯形波”就变成了“方波”。根据右手定则 $E=BLV$ 的公式，在匀速转动下，各绕组产生的反电动势波形也呈梯形波/方波。

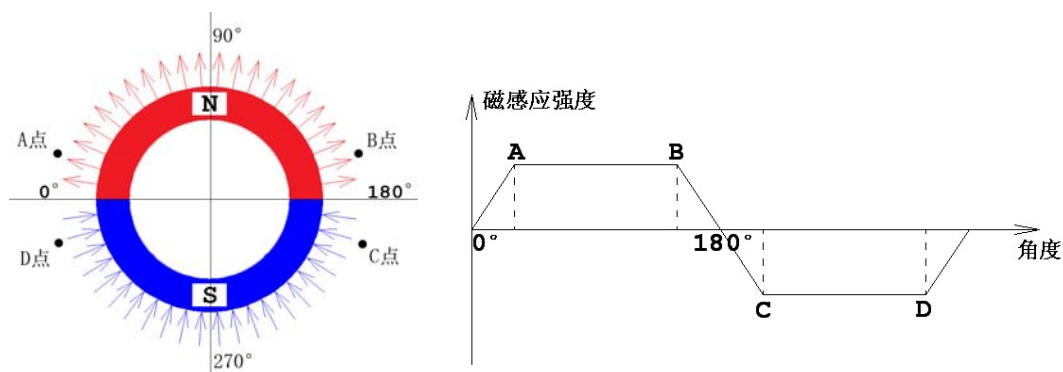


图 1-19 转子磁感应强度分布情况

与此类似，上文提到的另一种“正弦波”电机就是一种磁感应强度呈正弦波图形分布的直流无刷电机，也叫永磁同步电机。这种电机的绕组结构和我们的梯形波电机的绕组结构不太相同，进而驱动方式也不太相同，需要用到矢量分析法，由于本文只关注于梯形波的无刷直流电机，故对这种正弦波电机就不展开讨论了。需要研究的朋友可以查看专门文献。

3. 转子的受力分析

同样，我们仿照前面的做法，画出 6 种通电方式情形下，转子的受力情况，这里只用“左手定则”作一个定性分析。至于定量的计算，我们放到第三章的“启动算法”一小节中讨论。

在下面的图 1-20 中，除了画出了 6 种通电情形外，还画出了 6 个中间过程，这是为了更清楚地说明问题，同时也与下一节将要讨论的换相内容作一个衔接。

在图 1-20(a)中，AB 相通电，电流处于转子产生的磁场内，根据左手定则，我们判断线圈 AA' 中的上半部导线 A 受到一个顺时针方向的电磁力，而 AA' 的下半部导线 A' 也受到一个顺时针方向的电磁力。由于线圈绕组在定子上，定子是固定不动的，故根据作用力与反作用力，定子绕组 AA' 会施加给转子一个逆时针方向的反作用力，转子在这个力的作用下，就转起来了。同理，与 AA' 的情况类似，BB' 也会对转子产生一个逆时针的反作用力。

当转子逆时针转过 60° 后，到达图 1-20(b)的位置，这时线圈 BB' 已经到达转子磁极的边缘位置了，再转下去就要产生反方向的力了，所以这时就要换相，换成 AC 相通电，见图 1-20(c)。这样，每过 60° 换相通电，转子就可以一直转下去了。

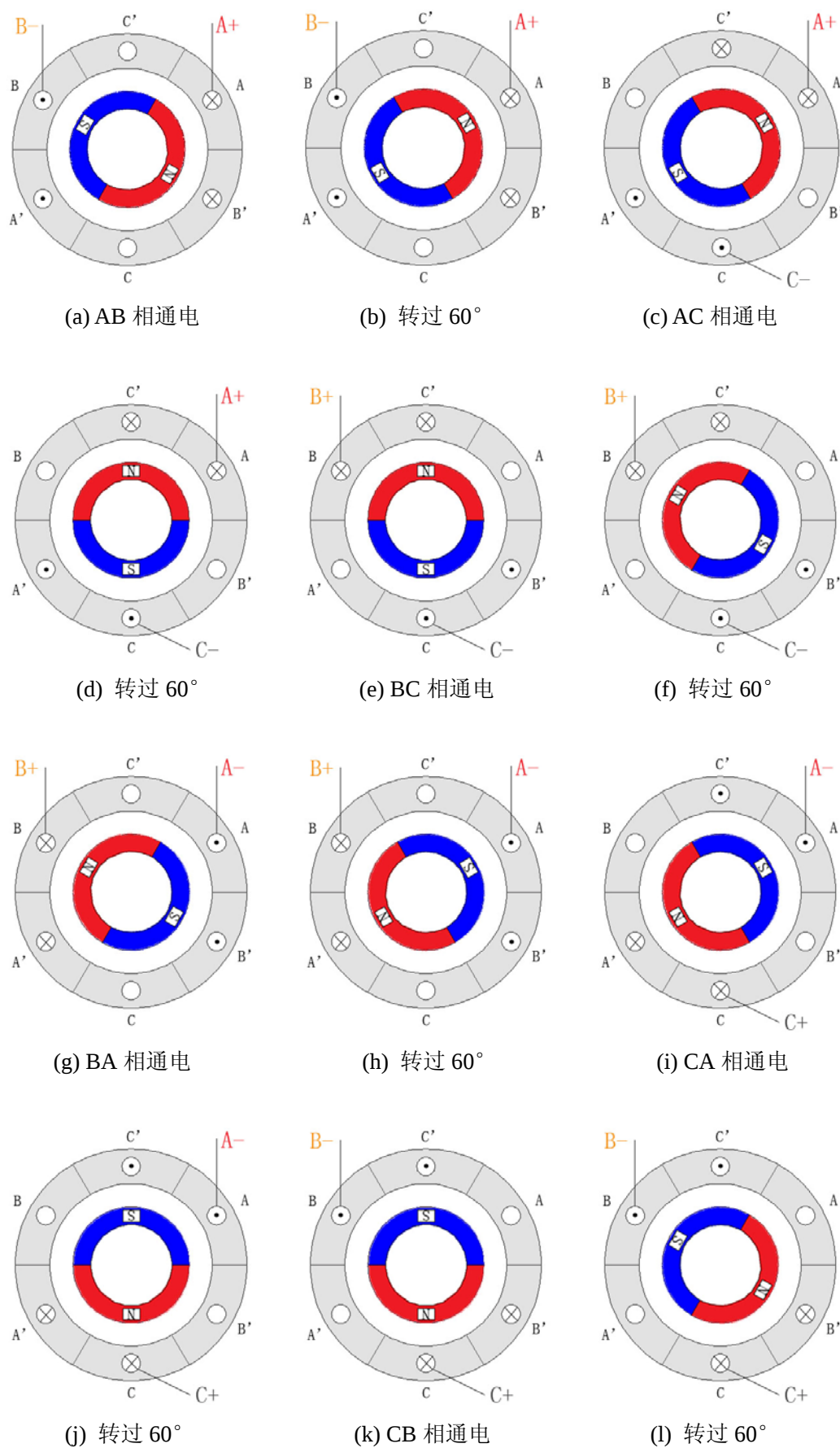


图 1-20 转子位置与换相的关系（参考自《电动机的单片机控制》王晓明著）

4. 一种近似分析模型

刚才的讨论全都基于一个假设，就是转子磁场的磁力线是垂直穿过绕组的导线的。但事实上，磁力线总是倾向于沿磁阻最小的路径前进，其实并不穿过导线，见下图。

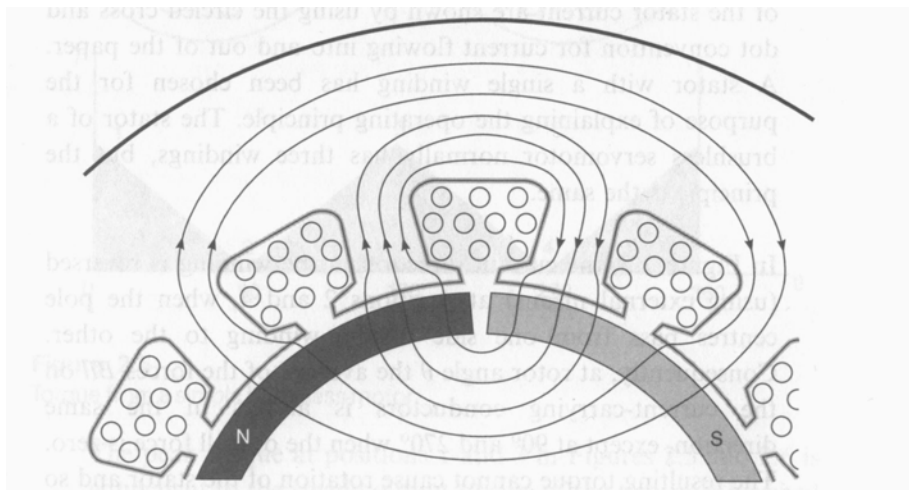


图 1-21 磁力线分布（摘自《Industrial Brushless Servomotors》）

如果要分析这种情况下转子的受力情况，要用到复杂的磁链路分析理论。不过，事实上不用这么麻烦，实验证明，用高深的磁链路分析方法所得到的结果，和我们上面假设磁力线穿过导线的分析方法所得到的结果，基本吻合。这句话可不是我说的，这是一本名为《Industrial Brushless Servomotors》的书中提到结论的。也就是说，我们现在可以放心地用左手定则和右手定则去对绕组作近似分析了。顺便提一句，这本书写得不错，篇幅也不大，就 180 多页，想更全面研究无刷电机的朋友可以看看。（老样子，没有电子书，图书馆借的）

好，有了这柄利器在手，再来看看我们能对 1.3 节的那种绕组结构作些什么简化和假设。毕竟，现在我们做四轴用的大多数电机都是以那种结构绕的。现仍以新西达 2212 电机为例，为了方便说明问题，每个绕组的 N 匝线圈现都简化成了一个，而且我们对所有绕组和磁极都做一了个编号，见图 1-22。

AB 相通电时，A1-1 导线处在 N 极下，根据左手定则，受到一个顺时针方向的作用力，即同时施加给转子一个逆时针方向的反作用力。同时，A1-2 导线处于 S 极下，但电流方向与 A1-1 相反，所以还是会施加给转子一个逆时针方向的作用力。与此类似，A2-1，A2-2，B3-1，B3-2，B4-1，B4-2 都会施加给转子一个逆时针方向的作用力，读者可自行分析。至于其换相和反电动势的情况，将放在下一小节详细分析。

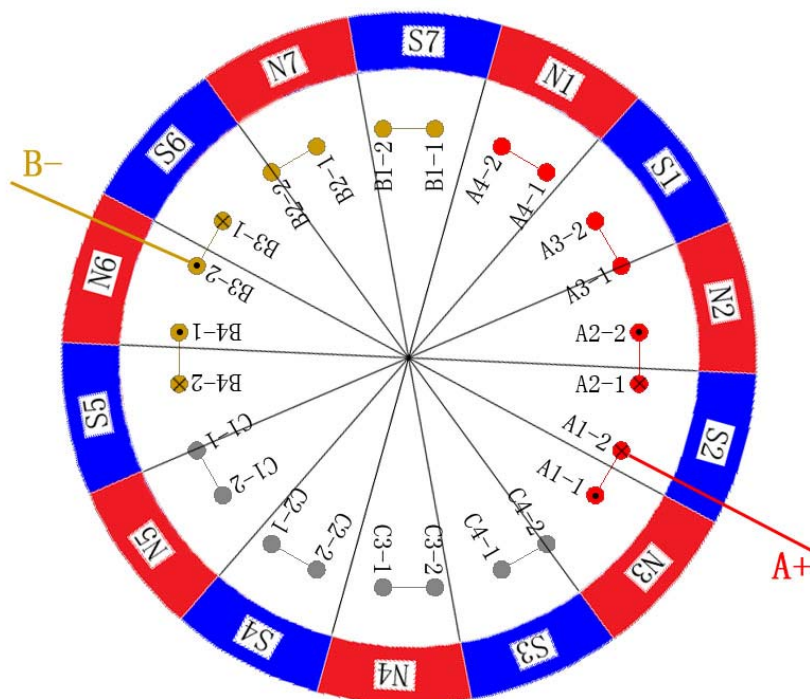


图 1-22 新西达 2212 电机 AB 相通电时情形

1.5 换相与调速

1. 换相基本原理

(1) 转子位置与过零检测

前面已经唧唧歪歪过很多遍了，换相的时机只取决于转子的位置，那顺理成章的问题就是：转子的位置怎么测？

一种比较简单的方式是用光电编码盘，这个东西在工业上用得比较多。不过由于其价格比较贵，而且还要接联轴器等一堆乱七八糟的东西，分量也不轻，显然不适合我们做四轴用。

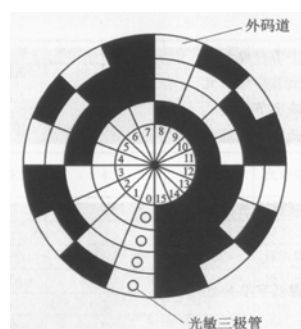


图 1-23 一种 4 位二进制编码盘（摘自《电动机的单片机控制》）

其次是用霍尔效应器件来测，简单来讲，霍尔效应测量器件可以根据转子不同位置时的不同磁场方向分布情况，而给出 1 或 0 的输出，一般在电机的不同位置上装三个霍尔传感器，就可测出转子的位置。这就是所谓的“有感无刷电机的驱动”。这种方法在很多文献中多有论述，汗牛充栋，这里我也不展开讲了，需要的朋友可参看相关文献和教材。值得一提的是，车模和船模中的电调多是使用“有感”方式，因为其电机需要频繁启动、停止、反转，而且对整套动力系统的重量也不是十分讲究，故用有感无刷电机电调是比较合适的。

接下来就是我们本文要主讲的“无感”测量方式。无传感器怎么测量？答：利用第三相的感生电动势。无感驱动方式的优点在于省略了三个霍尔传感器，整套系统分量更轻，结构更简单。其缺点在于启动比较麻烦（这个在后文会具体分析），启动的时候可控性较差，要达到一定转速后才变得可控。不过这对航模来说倒不是个问题，航空发动机一旦转起来后，在空中是不需要停车的。

现在我们来具体分析无感无刷直流电机如何利用第三相的感生电动势去测量转子的位置。回过头再去看图 1-20，先看图(a)和图(b)，在 AB 通电期间，你会发现线圈 CC' 的 C 边在图(a)中切割 N 极的磁力线并产生一个正向的感生电动势，在图(b)中确是切割 S 极的磁力线而产生一个反向的感生电动势了；C' 边的情况也类似。（这里我们定义：在转子逆时针旋转时，C 边切割 N 极磁力线和 C' 边切割 S 极磁力线产生的感生电动势为正；AA' 和 BB' 也用类似的定义）。这说明，在 AB 相通电期间，如果我们去测量线圈 CC' 上的电压，会发现其间有一个从正到负的变化过程。与此类似，图(c)~图(l)中的情况也可以用相同的方法分析出来，如图 1-24 所示（图在下页）。

这里需要说明一下的是，在 AB 相通电期间，不只是线圈 CC' 上产生感生电动势，其实 AA' 和 BB' 也在切割磁力线，也都会产生感生电动势，其电动势方向与外加的 12V 电源相反，所以叫“反向感生电动势”（BEMF）。其等效电路图见图 1-25。

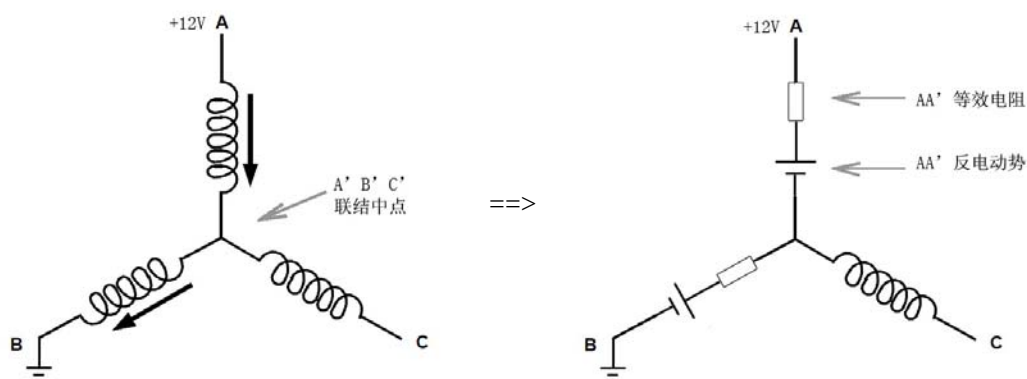


图 1-25 AB 相通电期间线圈 AA' 和 BB' 的等效电路（修改自 Microchip AN885 文档）

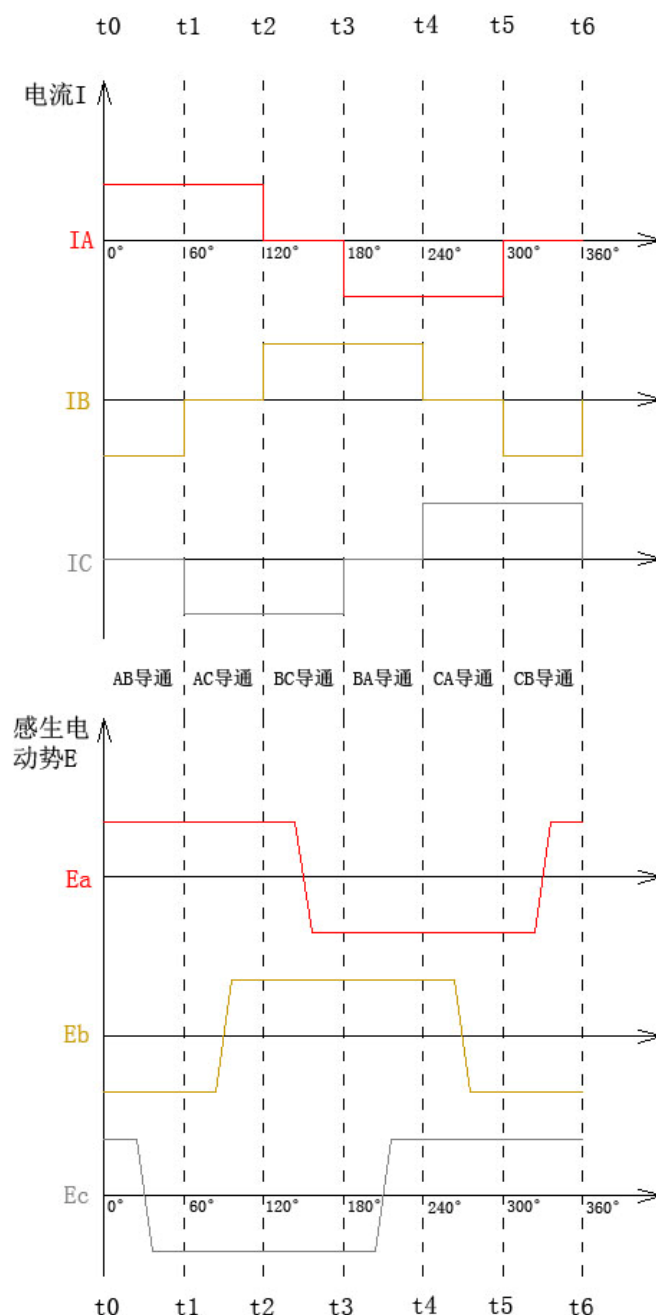


图 1-24 六种通电情形下各绕组的电流和感生电动势

从图 1-25 可以看出，线圈绕组 AA'和 BB'上产生的反电动势是很大的，两个加起来几乎略小于 12V。为什么呢，因为线圈绕组本身的等效电阻很小（约 0.1 欧左右），如果反电动势不大的话，端电压加载在线圈绕组等效电阻上，会产生巨大的电流，线圈非烧掉不可。为方便理解，我们姑且假设在额定转速下 AA'和 BB'各产生 5.7V 的反电动势，那么它们串联起来就产生 11.4V 的反电动势，结合图 1-25 看，那么加载在等效电阻上的电压就为 $12 - 11.4 = 0.6\text{ V}$ ，最终通过绕组 AB 的电流就是 $0.6 / (2 \times 0.1) = 3\text{ A}$ ，看来这个假设还是比较

合理的。同理，由于各绕组的结构是相同的，切割磁力线的速度也是相同的，所以线圈 CC' 也应该会产生一个大小约为 $5.7V$ 的感生电动势；不同的是：在 AB 相通电期间， CC' 的感生电动势会整个换一个方向，也即所谓的“过零点”。

在图 1-24 的 t_0 时刻（即图 1-20(a)的位置），为 AB 相通电刚开始时的情况， CC' 产生的感生电动势的等效电路图如图 1-26(a)所示；而在图 1-24 的 t_1 时刻（即图 1-20(b)的位置），为 AB 相通电快结束时的情况， CC' 产生的感生电动势的等效电路图如图 1-26(b)所示。

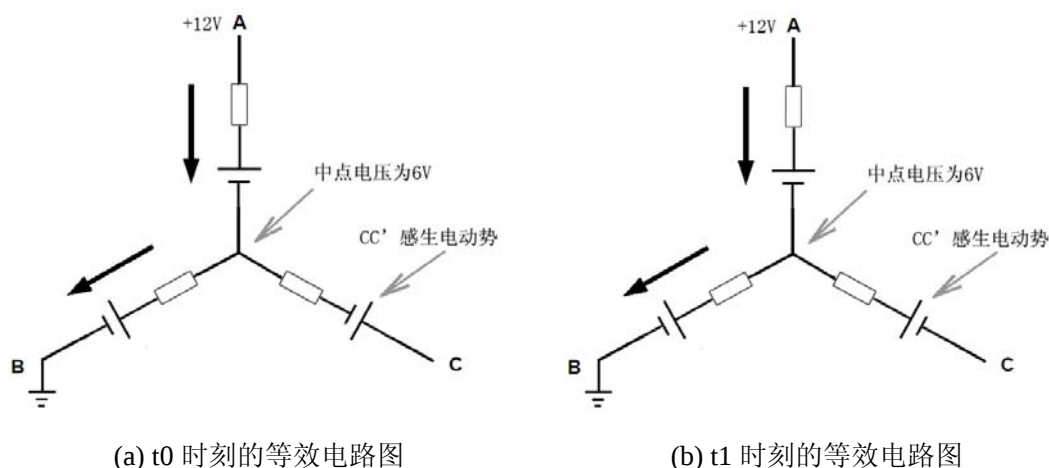


图 1-26 AB 相通电期间 CC' 的感生电动势

由于中点电势值始终为 $6V$ ， CC' 的线圈产生的感生电动势只能在以中点 $6V$ 电势为基准点的基础上叠加，仍旧假设在额定转速下 CC' 上会产生 $5.7V$ 的感生电动势，那么在 t_0 时刻，如果我们去测量 C 点的电压，其值应为 $6 + 5.7 = 11.7V$ ；在 t_1 时刻， C 点的电压值应为 $6 - 5.7 = 0.3V$ 。

电压都为正

也就是说，在 AB 相通电期间，只要一直监测电机的 C 引线的电压，一旦发现它低于 $6V$ ，就说明转子已转过 30° 到达了 t_0 和 t_1 中间的位置，只要再等 30° 就可以换相了。如果电调的 MCU 足够快的话，可以采用连续 AD 采样的方式来测量 C 点电压，不过貌似有点浪费，因为大部分采到的 AD 值都是没用的，我们只关心它什么时候低于 $6V$ 。这时候模拟比较器的作用就来了，它天生就是干这个活的料。比较器的联结电路图见图 1-27。一旦 C 相输出电压低于 $6V$ ，比较器马上可以感知并在输出端给出一个下降沿。同理，当电机处于 AC 相通电时，监测的是 B 相输出电压；当电机处于 BC 相通电时，监测的是 A 相输出电压。继续往前，当电机开始进入 BA 相通电时， C 相输出电压一开始会处于一个较低的状态（ $0.3V$ ），过零事件发生时， C 相输出电压会超过 $6V$ ，也就是说，这时比较器会感知并输出一个上跳沿。接下来的 CA ， CB 相通电情况也类似，不再赘述。

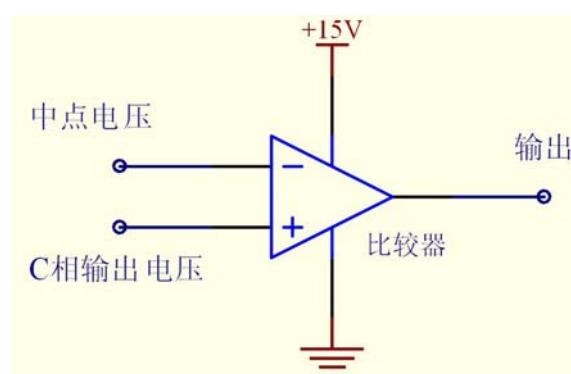


图 1-27 比较器的电路图

可能有人会说，这可是 15V 的比较器哪，单片机自带的比较器一般只支持最高 5V 的比较啊。事实上，上面这个电路图只是为了方便说明问题，在真正的实用中，会对 C 相输出电压和 6V 中点电压再加个分压电路，而且中点电压也不总是等于 6V，这个留待第二章再作详细分析，这里只要建立个概念就可以了。另外，这种检测方式中还有一个消磁问题，这个留待第三章软件部分予以分析。

(2) 换相策略

另一个问题是，就算检测到了 C 相的过零点，那还要等转子转过 30° 才可以换相，转这剩下的 30° 究竟要花多少时间？

一种比较简单的做法是近似认为转子转速在这 $0^\circ \sim 60^\circ$ 的小范围区间内基本是恒定的：从 AB 相开始通电到检测出 C 相过零的前半段时间，基本等于后半段的时间。所以只要记录下前半段的时间间隔 T1，等过零事件出现后再等待相同的时间，就可以换相了。

另一种比较暴力的做法是检测到过零事件后，也不再等转子再转 30° 了，立马就换相，事实上德国 MK 项目的 BL-Ctrl 电调程序就是这么干的。我们来看看这样做会有什么后果：

图 1-28(a)同图 1-20(a)，为 AB 刚开始通电时的情况。转过 30° 后，到达图(b)的位置时，检测到 C 相过零，如果此时立刻换相为 AC 导通，将成为图(c)的状态。这时，CC'线圈还处于 NS 极的交界处，此时穿过 CC'的磁感应强度为零，CC'上将不产生电磁力。也就是说此时只有线圈 AA'在出力，CC'处于出力不出力的状态。不过这个情况只是瞬时的，只要转子稍微向前再转一点，穿过 C'和 C 的磁感应强度就会开始增加，CC'就会开始出力。回忆一下图 1-19，如果梯形波电机工艺做得比较好，磁感应强度上升和下降直线比较陡峭的话，穿过 CC'的磁感应强度将很快达到最大值，期间损失的效率很小。如果电机的工艺做得一般般，上升和下降直线比较平缓的话，就会多损失一点效率，电机输出转矩的波动也会大一点。

接着往下看，当转子继续转过 30° 到达图(d)的位置时，一切都好，相安无事。当转子再转过 30° 到达图(e)的位置时，会检测到 B 相的过零事件，此时如果立刻换相成 BC 相通电，将成为图(f)的状态，刚导通的 BB' 线圈照例会处于“星期一综合症”的状态，效率很低、出工不出力，要再过一会儿才能进入最佳工作状态。

综上所述，暴力换相的方法也是可以用的，只不过损失一点效率。除了首次换相是间隔 30° 外，以后的每次的换相间隔也都是 60° ，转子旋转一周也是换 6 次相。如果有时间的话，我会做一个测试实验，比较采用以上两种不同换相策略时的电机功耗情况，测试结果将放于后续附录中。

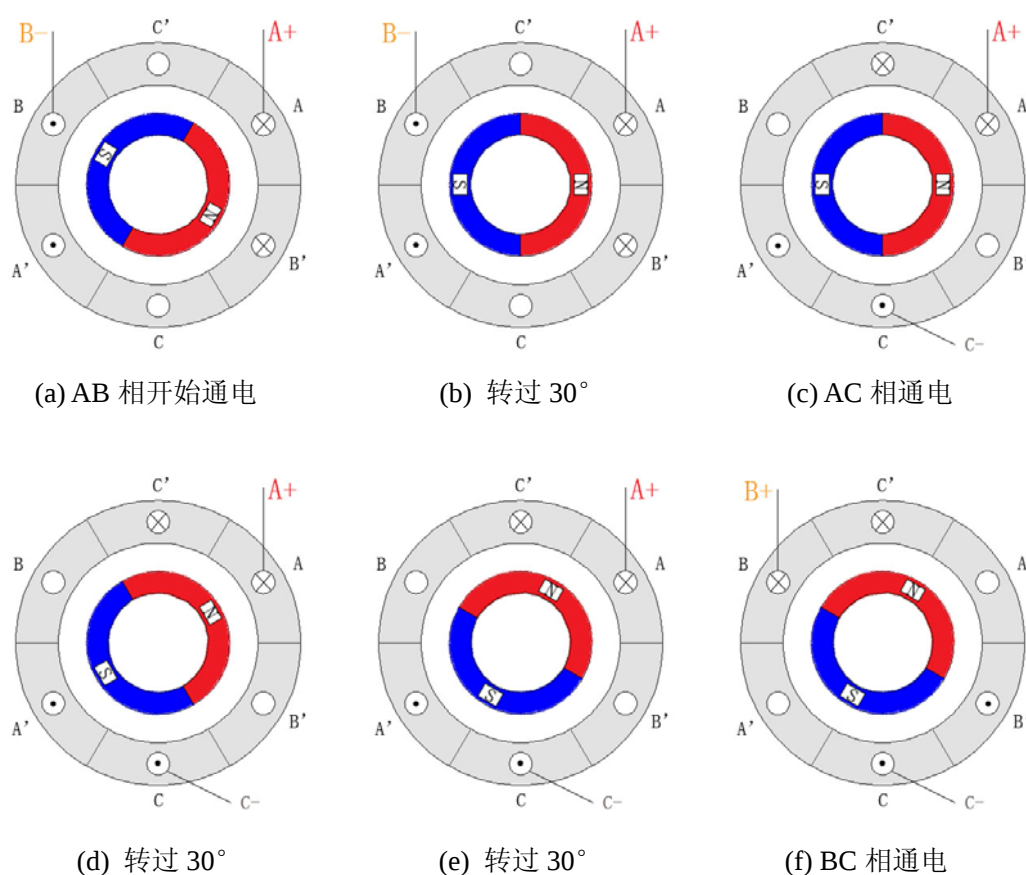


图 1-28 暴力换相时各情景的分析

2. 新西达 2212 电机的换相分析

分析完了上面最基本的三绕组二极无刷直流电机的换相情况，再来看看我们的老朋友，新西达 2212 电机的换相过程。不过在此之前，先要介绍一个概念：电角度。

图 1-29(a)显示了新西达 2212 电机 AB 相通电时情形，在图中可以看出，A1-2 绕组边和

A2-1 绕组边都处于磁极 S2 极下。现假设转子逆时针转过了一个角度，使得 A1-2 和 A2-1 都处于磁极 S3 之下，见图 1-29(b)。从物理上讲，每对磁极都是相同的，不同的磁极编号是人为加上去的；所以在定子绕组看来，图(a)和图(b)的磁场情况是完全相同的，故而这即是一个完整周期，期间应该完成全部的 6 次换相。

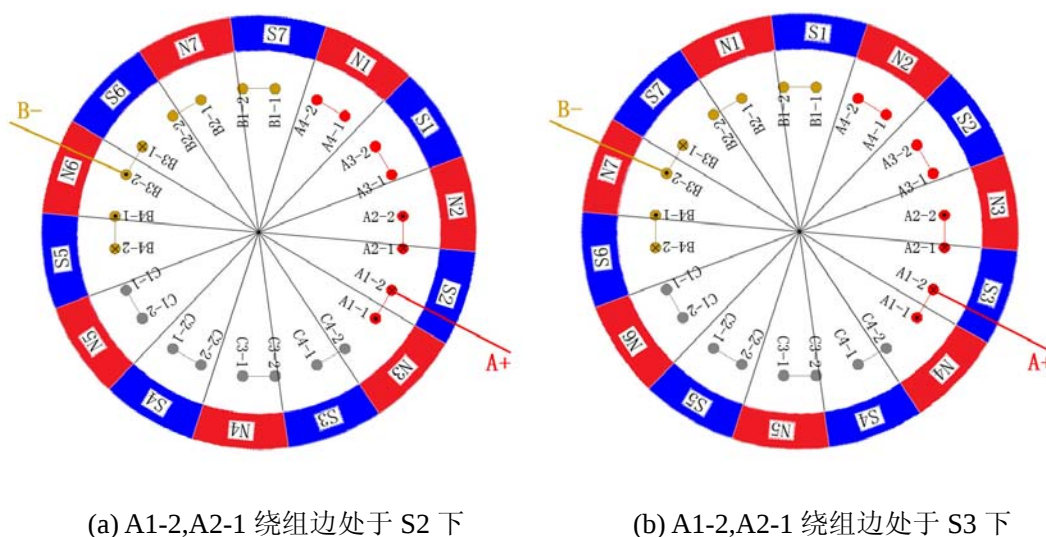


图 1-29 新西达 2212 电机的一个电周期

由于新西达 2212 电机共有 7 对磁极，故上述这个完整周期内，转子转过的机械角应为： $360/7 = 51.43$ 度，而其中的 6 步换相状态，每步所转过的机械角为 $51.43/6 = 8.57$ 度。由于结果中带了小数，对于说明问题很不方便，于是在电工技术中，就产生了电角度（简称电角）的概念。虽然每对磁极占圆周空间的机械角为 $360^\circ / (\text{极对数})$ ，但规定其电角度总为 360° 。这样在上图中，从 S2 极转到 S3 极，就等于转过了 360° 电角度，每次换相间隔仍为 60° 电角度。

好，现在来分析运行和换相过程。为方便理解，假设电机的梯形波上升下降沿十分陡峭，近似于方波，转子呈逆时针旋转。

在图 1-30(a)中，N2 和 S2 磁极的分界线刚划过 A1-2 绕组边，AB 相开始导通。由于其定子绕组诡异的绕线方式，电流其实是分两路走的（在图 1-15、图 1-22 和图 1-29 中为便于读者理解，没有画出第二路电流，这里特此说明）。参照图 1-14，第一路电流按照 $A+ \rightarrow A1-2 \rightarrow A1-1 \rightarrow A2-1 \rightarrow A2-2 \rightarrow B4-2 \rightarrow B4-1 \rightarrow B3-1 \rightarrow B3-2 \rightarrow B-$ 的路径走完；第二路电流按 $A+ \rightarrow A3-2 \rightarrow A3-1 \rightarrow A4-1 \rightarrow A4-2 \rightarrow C2-2 \rightarrow C2-1 \rightarrow C1-1 \rightarrow C1-2 \rightarrow C3-2 \rightarrow C3-1 \rightarrow C4-1 \rightarrow C4-2 \rightarrow B2-2 \rightarrow B2-1 \rightarrow B1-1 \rightarrow B1-2 \rightarrow B-$ 的路径走完。图中的点和 X 描述了电流方向。

由左手定则分析可知，第一路电流所经过的绕组边 A1-1, A1-2, A2-1, A2-2, B3-1,

B3-2, B4-1, B4-2 都会对转子产生一个逆时针方向的电磁力。且每个绕组边产生的反电动势为：略小于 $12/8 = 1.5\text{ V}$ 。其等效电路图见图 1-30(b)。

比较麻烦的是分析此时 C 相的感生电动势电动势，我们将第二路电流路径中的每个绕组边逐个分析，这里先预定义面向读者的感生电动势方向为正。结合绕组情况来看，A3-1 和 A3-2 都处于 S1 磁极下，产生的感生电动势方向相同，所以互相抵消。同理，B1-1 和 B1-2 的互相抵消，C1-1 和 C1-2 的互相抵消，C2-1 和 C2-2 的互相抵消，C3-1 和 C3-2 的互相抵消。只有 B2-1, B2-2, C4-1, C4-2 会产生有效的反向感生电动势，其等效电路图见图 1-30(c)。

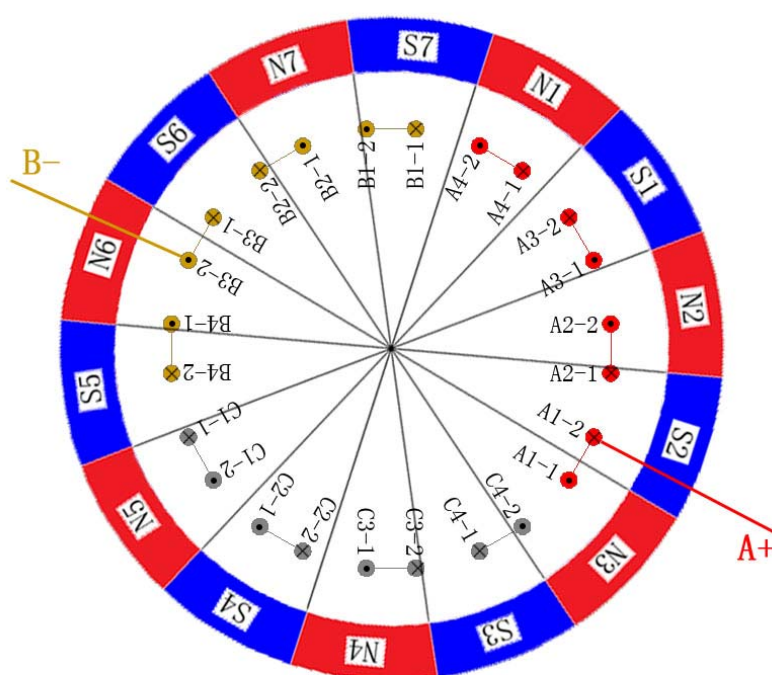


图 1-30(a)

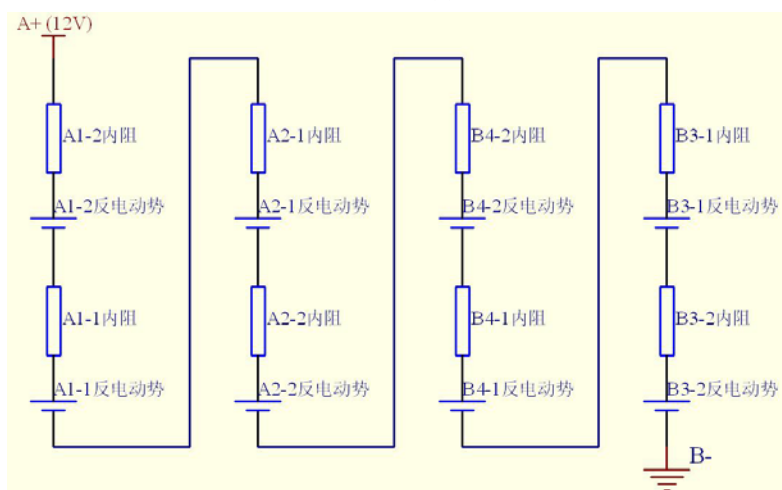


图 1-30(b)

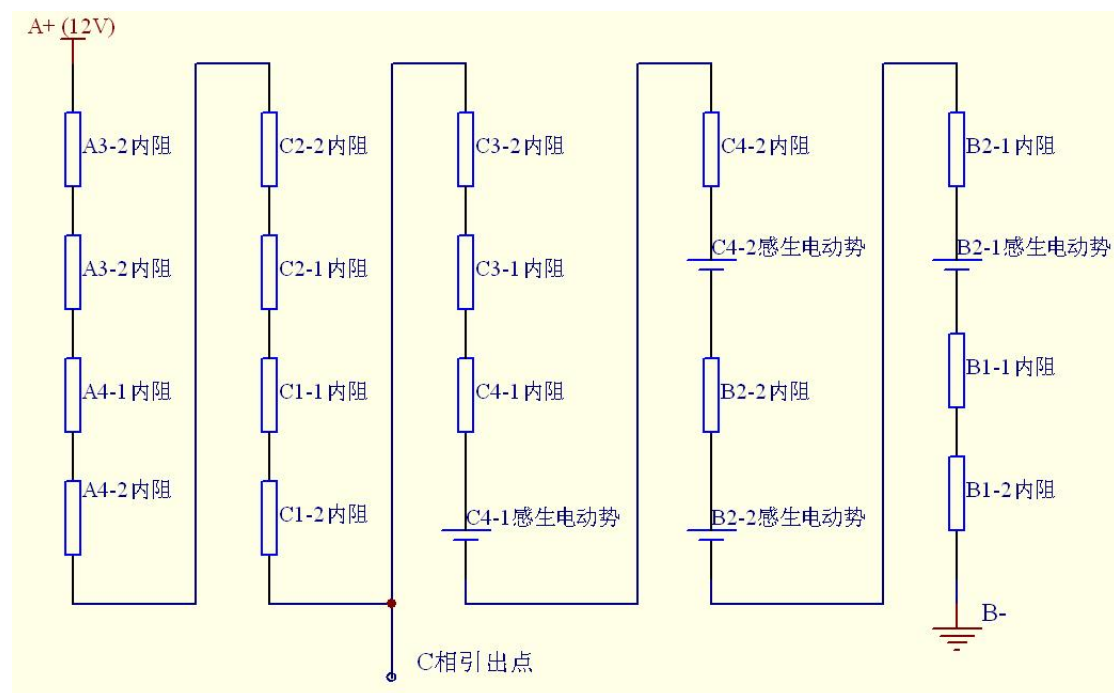


图 1-30(c)

根据前面每个绕组边产生略小于 1.5V 反电动势的结论，可算得 C 相引出点此时的电压值约为 8V，大于绕组中点电压 6V。

用相同的方法，我们再分析 AB 相导通快要结束时的情况。

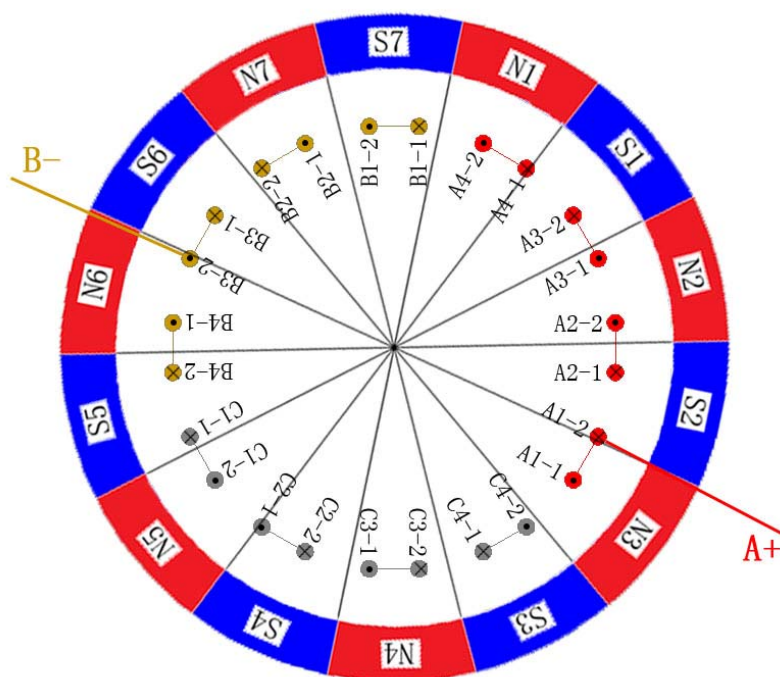


图 1-31(a)

在图 1-31(a)中, S2 和 N3 磁极的分界线将要(还未)划过绕组边 A1-2。第一路电流的情况同图 1-30(a), 这里就不重复了。要分析的是第二路电流中的反电动势情况: 从图中可以看出, A4-1 和 A4-2 的反电动势相互抵消, B1-1 和 B1-2 的相互抵消, B2-1 和 B2-2 的相互抵消, C2-1 和 C2-2 的相互抵消, C3-1 和 C3-2 的相互抵消, C4-1 和 C4-2 的相互抵消, 只有 A3-1, A3-2, C1-1, C1-2 能产生有效的反向感生电动势, 其等效电路图见图 1-31(b)。

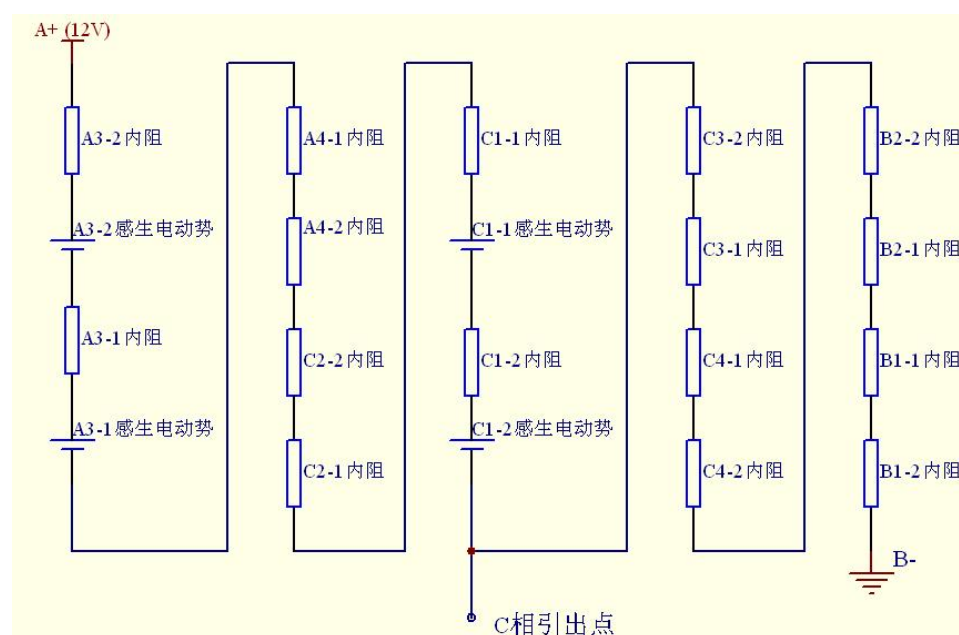


图 1-31(b)

根据这个等效电路图, 可算得 C 点此时的输出电压约为 4V, 小于绕组中点的电压 6V。
可见, 在 AB 相通电期间, C 相的输出电压穿越了 6V 的中点; 也就是说, 期间发生了一次过零事件, 可以被比较器检测到, 并以此作为换相的依据。这里还可以看出的是, 在第二路电流中, 产生有效反向感生电动势 A3-2, A3-1, C1-1, C1-2 绕组边, 其实对转矩也是有贡献的。由于这种电机非常特别的绕线方式, 在任何一种通电情况下, 电流都会流过所有绕组, 足可见其定量分析之复杂, 怪不得鲜有教材对这种电机结构做详细讲解。

接着换为 AC 相通电, 其第一路电流路径可见图 1-15(b), 第二路就自己分析啦, 偶实在是在画不动了……

3. 调速

无刷直流电机, 无论其换相模式多么复杂, 一些控制方式和交流同步电机多么相似, 但

从本质上来讲，还是属于直流电机。只不过将原来有刷直流电机的机械换向器，改成了现在的电子换相器。

直流电机怎么调速？当然是用直流电压来控制。电压越高，转得越快；电压越低，转得越慢。不过遗憾的是，单片机并不能输出可调的直流电压，于是只好变通一下，用脉宽调制（PWM）方式来控制电机的输入电压。PWM 占空比越高，等效电压就越高，占空比越低，等效电压就越低。

当然，单片机给出的 PWM 波形只是控制信号，而且最高电压也只有 5V，其能量并不足以驱动无刷直流电机，所以必须要再接一个功率管来驱动电机。功率管可以是 MOSFET（场效应管），也可以是 IGBT（绝缘栅双极晶体管）。关于驱动电路的结构，我们将在下一章详细讲述。

2. 无感无刷电调的驱动电路设计

这里主要参考的是德国 MicroKopter 项目的 V1.2 版本的电调电路，完整的电路原理图见附录二，建议各位还是把它的电路图打印下来看吧，页面翻上翻下很累人的说。

附录二中还给出了一个商业电调的电路（凤凰 25A 电调），虽然主控芯片一个用的是 MEGA8，一个用的是 C8051F，但通过比较外围功能电路可以看出，这两者是还是很相似的。下面我们将分模块地一部分一部分地分析其外围功能电路。

2.1 电池电压监测电路

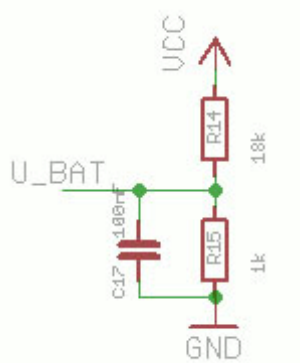


图 2-1 电池电压监测电路（摘自 MK 项目电调原理图 V1.2 版本）

先来简单的，热热身。图 2-1 是一个电阻分压网络，其中 VCC 接电源锂电池的正极，GND 接电源锂电池负极，U_BAT 接 MEGA8 的 ADC7 通道，电容 C17 用来消除电源中的一些高频波纹的影响。一节标准锂电池的电压为 3.7V，一般航模用锂电池都是三节串联，也就是 11.1V。若电池即将用尽，VCC 会下降，相应的 U_BAT 测得的电压也会下降。

不过在 MK 项目的电调程序 V0.41 版本中，作者把监控电压的工作放在了主控板上，所以在整个电调程序中没有任何测量 ADC7 通道的代码，本模块基本属于鸡肋。:)

2.2 换相控制电路

换相控制电路主要由 6 个功率场效应管和一些外围电阻和三极管构成，虽然原理不复杂，但设计起来还是比较复杂的。~~~~~

杂，但涉及到的相关知识还是蛮多的，所以要分几个部分讲。

1. 六臂全桥驱动电路原理

为了清楚地说明问题，我们先将原图作一些简化。

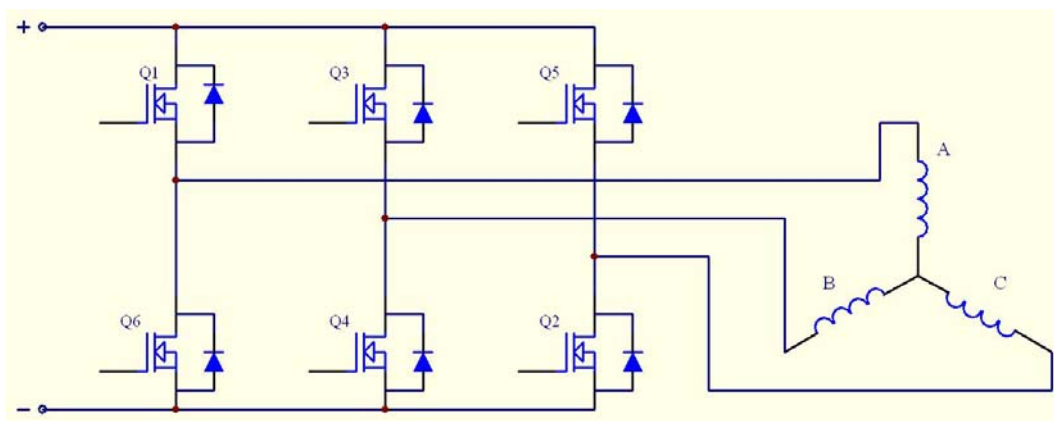


图 2-2 六臂全桥式驱动电路

Q1 到 Q6 为功率场效应管，当需要 AB 相导通时，只需要打开 Q1, Q4 管，而使其他管保持截止。此时，电流的流经途径为：正极→Q1→线圈 A→绕组 B→Q4→负极。这样，六种相位导通模式：AB, AC, BC, BA, CA, CB 分别对应的场效应管打开顺序为 Q1Q4, Q2Q2, Q3Q2, Q3Q6, Q5Q6, Q5Q4。

很简单么？好，接下来说一些稍微复杂一点的问题。不知道各位有没有注意到 Q1~Q6 的每个场效应管旁边还并联着一个二极管，这是干什么用的，画蛇添足的设计么？

非也。我们在第一章曾经提到，无刷直流电机的调速是用 PWM 波形的占空比来调，图 2-2 中，采用的是 H_PWM--L_ON 方式来驱动的，也就是上臂采用 PWM 信号控制，而下臂常开的一种驱动方式。比如在 AB 相导通时，单片机给 Q1 的栅极是 PWM 信号，而给 Q4 的栅极是常开信号，这样你可以通过控制 Q1 输入端的 PWM 信号占空比来控制驱动电机的有效电压。此时 A 端和 B 端的电压波形如图 2-3 的圆圈中所示（我们等会儿再讲 C 相电压是怎么回事）。现在问题来了，A 相的电压是可以突变的，但是由于电感的作用，流经 AB 线圈的电流是不能突变的。你不给人家一条活路，人家是要造反的，这里所谓的造反就是线圈由于自身电感的作用产生极高的瞬时反电动势（回忆一下： $U = L \frac{di}{dt}$ ）而击穿元器件。所以这时候二极管的作用就来啦，在 PWM 信号的低电平期间，电流是按照图 2-4 所示的箭头路径续流的。由于负极端电位强制为零，二极管有一个正向压降，A 点的电压就可以在瞬间

降到比零略小的值，与图 2-3 的实验结果相吻合。

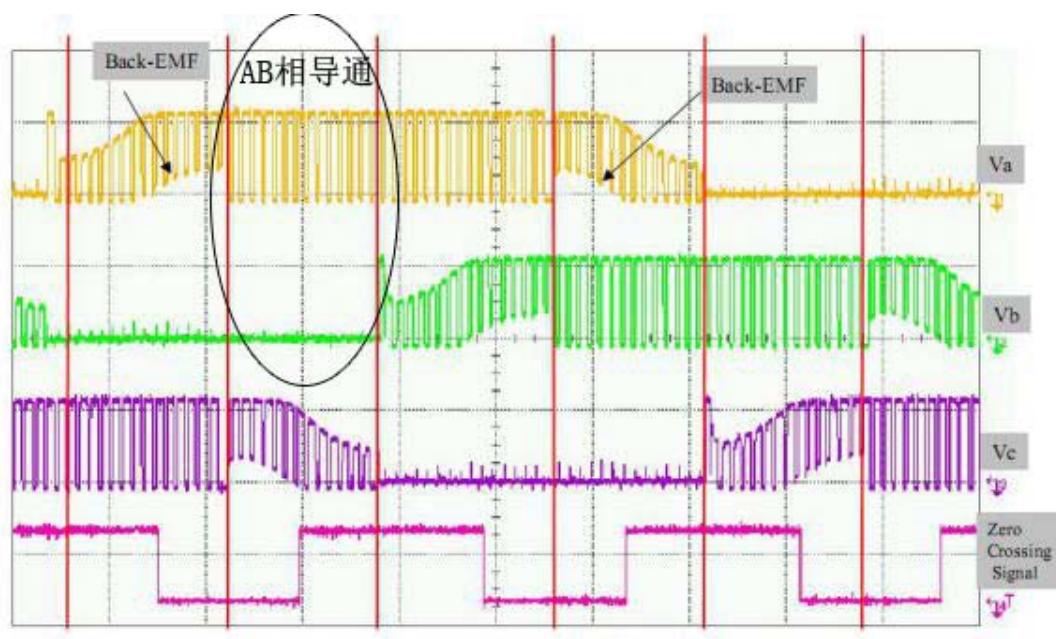


图 2-3 各相电压波形（摘自网友 lijieamd 实验结果）

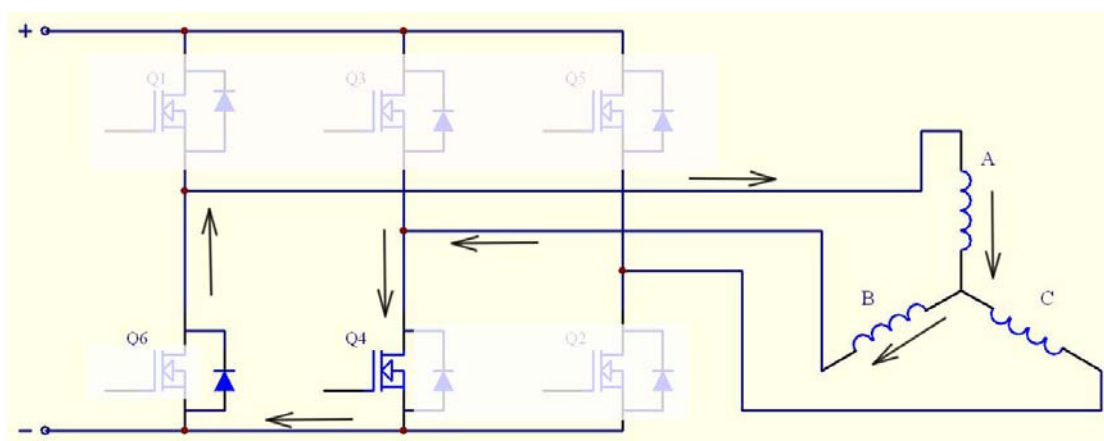


图 2-4 AB 相续流期间电流方向

由于 A 点电位忽上忽下的变化，会导致 ABC 线圈的中点电位也会忽上忽下的变化（中点电位总是等于 A 点和 B 点电位的平均值），我们来看看这样会对我们采样 C 点的反电动势有什么影响。当 PWM 处于高电平期间，A 点的电压值接近 12V，中点的电压值接近 6V，根据我们在第一章的分析，C 线圈产生的感生电动势叠加在中点上，会在 C 点产生接近于 12V 的电压值。然后 PWM 进入低电平期间，A 点电位迅速降到略小于零，中点电位也会迅速降到略小于零，这时 C 线圈的感生电动势就会以零为基点往上叠加，此时 C 点的电压就是略小于 6V，这个也可以在图 2-3 中得到验证。虽然 C 点电压向下穿越了 6V，但是回忆一

下比较器的结构（见图 2-5），由于中点电压和 C 点电压同时降低和升高，所以不管中点电位如何变化，只要 C 线圈本身的感生电动势不过零，比较器输出就不会产生跳变。有人也许会问，这个悬浮的中点电压是怎么测得的呢，又不能从中点引根线出来。其实这是通过一个设计很巧妙的分压电路根据 A 点和 B 点的电压值估计出来的，这个放到下面的“反电势过零点检测电路”一小节详讲。

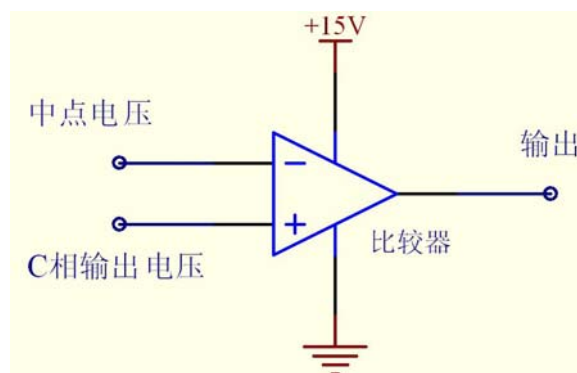


图 2-5 比较器电路图

随着转子继续旋转，C 线圈的感生电动势终将由正变负，而被比较器给感知到。至于图 2-3 的波形图中为什么没有 C 电压为负值的点，因为 C 端电压如果负得太厉害，Q2 的二极管就会导通，而将 C 端电位钳制在 -1V 左右。C 点理想的电压波形我想应该是这样的：

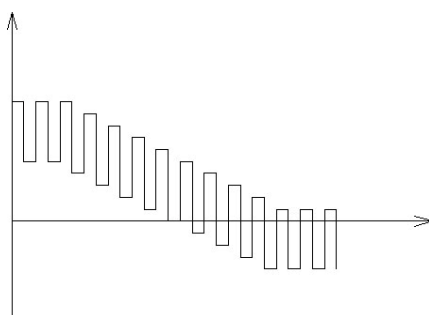


图 2-6 C 点电压波形（想象图）

图 2-3 中其实还有一个知识点，是关于消磁事件的，这个放在第三章软件部分讲。

2. 功率场效应管的选择

(1) N 型和 P 型 MOSFET

上面的图 2-2 电路图对于理解换相原理来说，是可以用的，但在实际的电路中，是不能用的。为什么呢？问题于 N 型场效应管的门限开启电压 V_{GS} 。

先来复习一下场效应管的基本知识，图 2-7(a)是一个 N 沟道型场效应管，图 2-7(b)是一个 P 沟道型场效应管。N 沟道场效应管有点类似于 NPN 三极管，只要栅源极间加一个正向电压，并且其值超过数据手册上的 V_{GS} 阈值电压时，场效应管的 D 极和 S 极就会导通。一般 N 型功率型场效应管的 V_{GS} 阈值电压都会在 3~20V 之间。

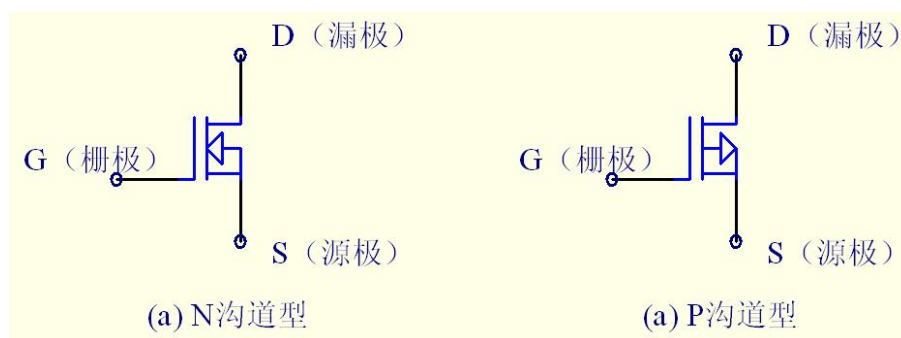


图 2-7 N 型和 P 型场效应管

现假设 AB 相通电（见图 2-8），Q1 和 Q4 管导通。一般场效应管的导通电阻 R_{DS} 都在毫欧级，所以可近似忽略场效应管的压降 V_{DS} 。这样，A 点的电位就近似为 12V，B 点的电位近似为 0V。为了要导通 Q4，Q4 的栅极电压必须大于 3V，这个靠单片机的 I/O 输出是可以办到的。但如果要导通 Q1，则在 Q1 的栅极必须至少加载 $12 + 3 = 15$ V 的电压，这个已经超过了电源电压，纯靠单片机加三极管的电路是办不到的。

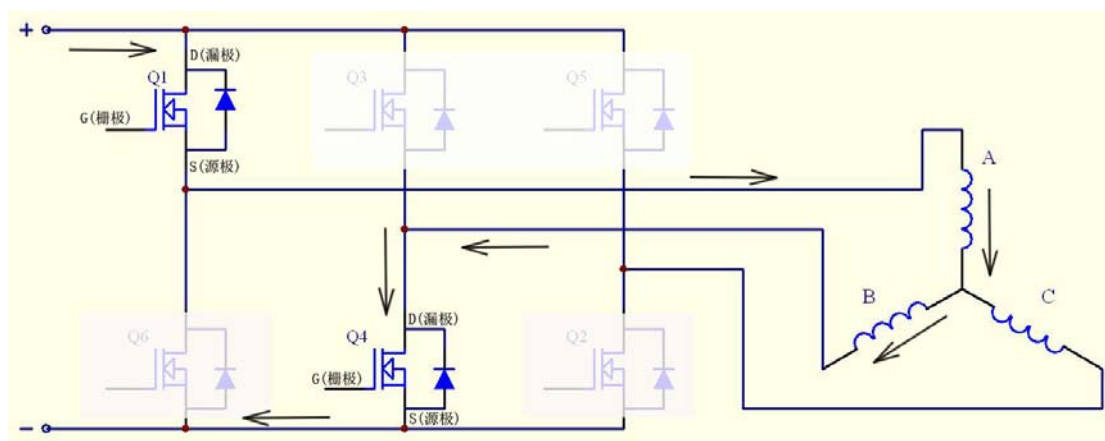


图 2-8 AB 相导通时情形

解决的方法有两个：一是采用自举升压电路，“hn_ny_dxs 夏风”同学给出了一个完整的解决方案，其采用 6 个 N 型场效应管，并配以自举升压电路，可为上臂的驱动管 Q1 的栅极提供 $2 \times 11 = 22$ V 的电压，足以导通 Q1。附录二中给出了夏风同学的帖子的链接地址，有图有真相，在那个帖子的 26 楼还有他对升压电路的工作原理的说明。

另一种方法就是采用 3 个 N 型场效应管和 3 个 P 型场效应管，这样可避开驱动电压的问题，这也是德国 MK 项目的电调采用的方案。P 型场效应管有点类似于 PNP 三极管（见图 2-7(b)），只要栅极电压小于源极电压（ V_{GS} 为负值），并且其值小于某一负的阈值电压，场效应管的 S 极和 D 极就会导通，电流从 S 极流向 D 极。一般 P 型功率型场效应管的 V_{GS} 阈值电压都会在 $-3 \sim -20V$ 之间。

下面来分析图 2-9，这是德国 MK 的电调的换相驱动电路部分，下臂用的是 IRLR7843 的 N 型 MOSFET，如果在 STEUER_A- 端给以 5V 的栅极电压，场效应管 NA- 就会导通，所以这个端口可以直接用单片机的 I/O 口驱动。上臂用的是 FDD6637 的 P 型 MOSFET，当 STEUER_A+ 端给出高电平时，三极管导通，FDD6637 的栅极被拉低，这样在 FDD6637 的栅源极之间就会形成一个负电压，而导致场效应管 NA+ 导通。图中 P 管的外围电阻 R2,R3 和 N 管的外围电阻 R16,R17 都是有作用的，我们会在本小节的第三部分试图给出一种分析。

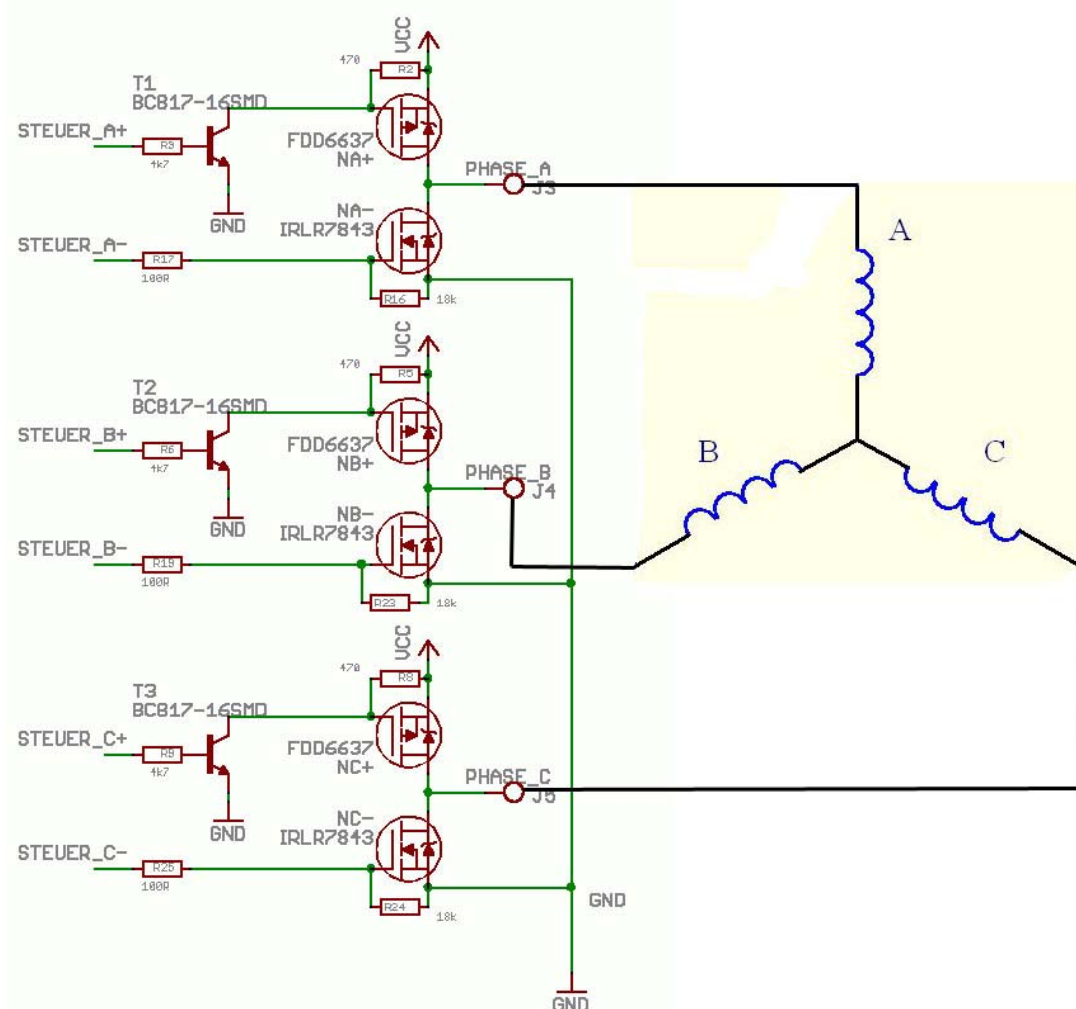


图 2-9 换相控制电路 (摘自 MK 项目电调原理图 V1.1 版本)

(2) MOSFET 的选型

下面来谈谈功率场效应管的选型。在 MK 项目的电调电路 V1.1 版本中,用的是 Fairchile 公司的 FDD6637 (P 管) 和 IR 公司的 IRLR7843 (N 管); 而在 V1.2 版本的电路中, 却都换成了 IR 公司的 IRFR5305 (P 管) 和 IRFR1205 (N 管), 两者到底有什么不同? 稍后我们将逐个参数地比较一下两者的区别。另外, 关于 MOSFET 的驱动设计, 有一篇很好的文档《Matching MOSFET Drivers to MOSFETs》(Microchip 公司, 编号 AN799), 大家有时间的话可以看一下, 附录二中给出了其下载地址。下面我们来看 MOSFET 选型主要考虑的参数:

➤ 1. 额定参数 (Absolute Maximum Ratings)

N 管比较: (表 2-1)

参数名称	符号	IRLR7843 (N 管)	IRFR1205 (N 管)
漏源极击穿电压	V_{DS} 或 $V_{(BR)DSS}$	30 V	55 V
栅源极击穿电压	V_{GS}	± 20 V	± 20 V
通态漏极电流	I_D @ $T_C = 25^\circ\text{C}$	161 A	44 A
漏极峰值电流	I_{DM}	620 A	160 A
漏源极雪崩击穿电流	I_{AR}	12 A	25 A
总功耗	P_D @ $T_C = 25^\circ\text{C}$	140 W	107 W
额定结温和储存温度	T_J, T_{STG}	$-55\sim+175^\circ\text{C}$	$-55\sim+175^\circ\text{C}$

P 管比较: (表 2-2)

参数名称	符号	FDD6637 (P 管)	IRFR5305 (P 管)
漏源极击穿电压	V_{DS} 或 $V_{(BR)DSS}$	-35 V	-55 V
栅源极击穿电压	V_{GS}	± 25 V	± 20 V
通态漏极电流	I_D @ $T_C = 25^\circ\text{C}$	-55A	-31 A
漏极峰值电流	I_{DM}	-100 A	-110 A
漏源极雪崩击穿电流	I_{AR}	-14A	-16 A

总功耗	$P_D @ T_C = 25\text{ }^{\circ}\text{C}$	57 W	110 W
额定结温和储存温度	T_J, T_{STG}	-55 ~ +175 $^{\circ}\text{C}$	-55 ~ +175 $^{\circ}\text{C}$

选择功率型场效应管首先要考虑参数的是漏源极的额定电压 V_{DS} 和额定电流 I_D 。从上两个表格中可以看出，V1.2 版的 MOSFET 的漏源极耐压性普遍要好于 V1.1 版本，但电流特性稍微弱了一点。而且上表中给出的漏源极电流，只是特定实验条件下的参考值，真正的漏源极额定电流还和栅源极电压 V_{GS} 有关，图 2-10 为 IRFR1205 的 $V_{GS} - I_D$ 曲线，称为转移特性，它表示了 MOSFET 的放大能力，用跨导表示。跨导 g_{FS} 定义为： $g_{FS} = \Delta I_D / \Delta V_{GS}$ ，单位为 S（西门子），它表明了转移特性的斜率（ g_{FS} 值在后表中）。可以看到，当栅极电压为 5V 时，允许的导通电流仅为 25A 左右，虽然比不过商业电调，但对四轴的电机来说，是足够了。而且较小的允许 I_D 一定程度上能保护电流不至于过大。

还有一个比较重要的参数就是开启阈值电压 V_{GS} ，这个参数在表 2-5 和表 2-6 中。

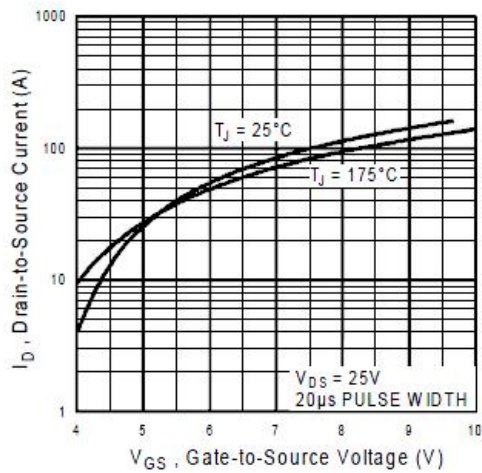


Fig 3. Typical Transfer Characteristics

图 2-10 IRFR1205 的转移特性（摘自 IR1205 datasheet）

➤ 2. 热传导特性（Thermal Characteristics）

N 管比较：（表 2-3）

参数名称	符号	IRLR7843 (N 管)	IRFR1205 (N 管)
结-壳热阻	$R_{\theta JC}$	1.05 $^{\circ}\text{C/W}$	1.4 $^{\circ}\text{C/W}$

结-空气环境热阻(PCB 贴片散热)	$R_{\theta JA}$	50 °C/W	50 °C/W
结-空气环境热阻	$R_{\theta JA}$	110 °C/W	110 °C/W

P 管比较：（表 2-4）

参数名称	符号	FDD6637（P 管）	IRFR5305（P 管）
结-壳热阻	$R_{\theta JC}$	2.2 °C/W	1.4 °C/W
结-空气环境热阻(PCB 贴片散热)	$R_{\theta JA}$	40 °C/W	50 °C/W
结-空气环境热阻	$R_{\theta JA}$	96 °C/W	110 °C/W

热阻简单地说就是导热性能。有一个形象的比喻：热阻就像是电阻，热流就像是电流，温度就好比是电压。Junction temp（结温）即是半导体的核心温度。一般来说，结-壳热阻是给定的，结-空气热阻也是给定的，可以通过在空气和金属壳之间并联一个较小的热阻（即散热器）来降低温差。从表中也可以看到，第二行通过 PCB 散热后的热阻要明显小于第三行。不过这些参数我们一般看看就行了，只要不是大的太离谱就可以。

➤ 3. 电气特性（Electrical Characteristics）

N 管比较：（表 2-5）

参数名称	符号	IRLR7843（N 管）		IRFR1205（N 管）	
		测试条件	典型值	测试条件	典型值
零栅极电压下的漏极漏电流	I_{DSS}	$V_{DS} = 24V$ $V_{GS} = 0V$	1 uA	$V_{DS} = 55V$ $V_{GS} = 0V$	25 uA
栅极至源极漏电流	I_{GSS}	$V_{GS} = \pm 25V$ $V_{DS} = 0V$	± 100 nA	$V_{GS} = \pm 20V$ $V_{DS} = 0V$	± 100 nA
栅极开启阈值电压	$V_{GS(th)}$	$V_{DS} = V_{GS}$ $I_D = -250uA$	1.5 ~ 2.3V	$V_{DS} = V_{GS}$ $I_D = -250uA$	2 ~ 4V
静态漏源极电阻	$R_{DS(on)}$	$V_{GS} = 10V$ $I_D = 15A$	2.6m Ω	$V_{GS} = 10V$ $I_D = 26A$	27m Ω

V_{GS} 与 I_D 正向跨导	g_{FS}	$V_{DS} = 15V$ $I_D = 12A$	37 S	$V_{GS} = 25V$ $I_D = 25A$	17 S
-----------------------	----------	-------------------------------	------	-------------------------------	------

P 管比较：（表 2-6）

参数名称	符号	FDD6637（P 管）		IRFR5305（P 管）	
		测试条件	典型值	测试条件	典型值
零栅极电压下的漏极漏电流	I_{DSS}	$V_{DS} = -28V$ $V_{GS} = 0V$	-1 uA	$V_{DS} = -28V$ $V_{GS} = 0V$	-25 uA
栅极至源极漏电流	I_{GSS}	$V_{GS} = \pm 25V$ $V_{DS} = 0V$	± 100 nA	$V_{GS} = \pm 20V$ $V_{DS} = 0V$	± 100 nA
栅极开启阈值电压	$V_{GS(th)}$	$V_{DS} = V_{GS}$ $I_D = -250uA$	-1 ~ -3V	$V_{DS} = V_{GS}$ $I_D = -250uA$	-2 ~ -4V
静态漏源极电阻	$R_{DS(on)}$	$V_{GS} = -10V$ $I_D = -14A$	9.7m Ω	$V_{GS} = -10V$ $I_D = -16A$	65 m Ω
V_{GS} 与 I_D 正向跨导	g_{FS}	$V_{DS} = -5V$ $I_D = -14A$	35 S	$V_{GS} = -25V$ $I_D = -16A$	8 S

从上两表的比较中可以看到，V1.2 版本的导通时漏源极电阻要明显大于 V1.1 版本，可能会在栅源极导通大电流时产生更多的热量（ $P = I^2 R$ ），并且吃掉更多的压降。另外 V1.2 版本的正向跨导也要明显低于 V1.1 版本，这个也印证了我们前面关于“V1.2 版本的电流能力稍弱”的判断。

➤ 4. 动态特性（Dynamic Characteristics）

N 管比较：（表 2-7）

参数名称	符号	IRLR7843（N 管）		IRFR1205（N 管）	
		测试条件	典型值	测试条件	典型值

输入电容	C_{iss}	$V_{DS} = 15V$	4380pF	$V_{DS} = 25V$	1300pF
反向传输电容	C_{oss}	$V_{GS} = 0V$	940pF	$V_{GS} = 0V$	410pF
输出电容	C_{rss}	$f = 1.0MHz$	430pF	$f = 1.0MHz$	150pF

P 管比较：（表 2-8）

参数名称	符号	FDD6637（P 管）		IRFR5305（P 管）	
		测试条件	典型值	测试条件	典型值
输入电容	C_{iss}	$V_{DS} = -20V$	2370pF	$V_{DS} = -25V$	1200pF
反向传输电容	C_{oss}	$V_{GS} = 0V$	470pF	$V_{GS} = 0V$	520pF
输出电容	C_{rss}	$f = 1.0MHz$	250pF	$f = 1.0MHz$	250pF

关于这三个电容的位置和关系，可见图 2-11。半导体电路设计中一个很麻烦的问题就是无处不在的小电容，这些电容在低频的应用中不存在什么问题，但在高频电路中，就成为影响电路性能的主要杀手。从表中数值中可以看出，一般 $C_{iss} \gg C_{rss}$ ，但由于密勒（Miller）效应，使栅源极电容增大了许多倍，成为影响放大器带宽的主要因素。稍后我们将看到，由于这几个电容的存在，MOSFET 的外围驱动电路不得不增加一些电阻，以消除它们的影响。

从上两表的比较可以看出，V1.2 版本的极间电容的综合情况，要明显好于 V1.1 版本，故动态特性也会优于 V1.1 版本，这可能是其换型的主要原因。

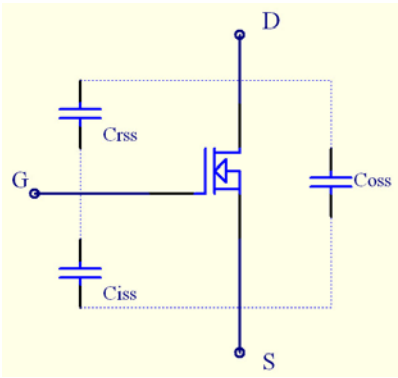


图 2-11 MOSFET 的极间电容

➤ 5. 开关特性（Swich Characteristics）

N 管比较：（表 2-9）

参数名称	符号	IRLR7843 (N 管)		IRFR1205 (N 管)	
		测试条件	典型值	测试条件	典型值
开通延迟时间	$t_{d(on)}$	$V_{DD} = 15V$	25 ns	$V_{DD} = 28V$	7.3 ns
上升时间	t_r	$I_D = 12A$	42 ns	$I_D = 25A$	69 ns
关断延迟时间	$t_{d(off)}$	$V_{GS} = 4.5V$	34 ns	$V_{GS} = 5V$	47 ns
下降时间	t_f	$R_G = 0\Omega$	19 ns	$R_G = 12\Omega$	60 ns
总栅极充电量	Q_g	$I_D = 12A$	34 nC	$I_D = 25A$	65 nC
栅源极充电量	Q_{gs}	$V_{DS} = 15V$	9.1 nC	$V_{DS} = 44V$	12 nC
栅漏极 (Miller 效应) 充电量	Q_{gd}	$V_{GS} = 4.5V$	12 nC	$V_{GS} = 10V$	27 nC

P 管比较: (表 2-10)

参数名称	符号	FDD6637 (P 管)		IRFR5305 (P 管)	
		测试条件	典型值	测试条件	典型值
开通延迟时间	$t_{d(on)}$	$V_{DD} = -20V$	18 ns	$V_{DD} = -28V$	14 ns
上升时间	t_r	$I_D = -1A$	10 ns	$I_D = -16A$	66 ns
关断延迟时间	$t_{d(off)}$	$V_{GS} = -10V$	62 ns	$V_{GS} = -10V$	39 ns
下降时间	t_f	$R_G = 6\Omega$	36 ns	$R_G = 6.8\Omega$	63 ns
总栅极充电量	Q_g	$I_D = -14A$	45 nC	$I_D = -16A$	63 nC
栅源极充电量	Q_{gs}	$V_{DS} = -20V$	7 nC	$V_{DS} = -44V$	13 nC
栅漏极 (Miller 效应) 充电量	Q_{gd}	$V_{GS} = -10V$	10 nC	$V_{GS} = -10V$	29 nC

先来看看时间参数, 图 2-12 显示了四个时间参数之间的关系。以 P 管为例, 一般从 V_{GS} 开始化到 V_{DS} 开始变化有一定的延迟, 而且方向相反。所以 $t_{d(on)}$ 定义为: 从 V_{GS} 上升到 10%~ V_{DS} 下降到 90% 的延迟。而上升时间 t_r 则定义为: V_{DS} 从 90% 下降到 10% 所用的时间。开通时间 t_{on} 即为延迟时间 $t_{d(on)}$ 与上升时间 t_r 之和。后面的关断时间 t_{off} 的定义也类似。(注意一

下 FDD6637 的 t_r 的测试条件, I_D 只有 1A, 厂家玩了个小 trick, 呵呵)

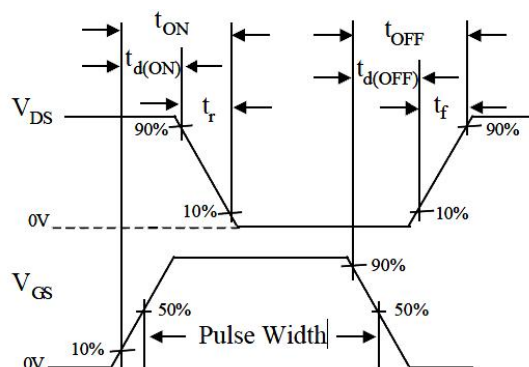


Figure 19. Switching Time Waveforms

图 2-12 开关特性参数 (摘自 FDD6637 datasheet)

再看看参数“总栅极充电量 Q_g ”的作用, Q_g 的单位是 C (库伦), 可结合图 2-13 和图 2-12 一起看以便于理解。当 MOSFET 开通时, 由于有栅源极电容 C_{iss} 的存在, 在其充电到阈值电压 $V_{GS(th)}$ 以前, I_D 和 V_{DS} 都基本不变, 这段时间就是上面所说的开通延迟时间 $t_{d(on)}$ 。当 V_{GS} 上升到开启阈值电压 $V_{GS(th)}$ 时, I_D 才明显上升, 同时 V_{DS} 下降, 进入上升时间。在上升时间里, 由于 V_{DS} 下降, 使栅极电流向漏极与栅极之间的“传输反向电容 C_{rss} ”(C_{rss} 先反向放电然后再充电), 在这段时间内栅极电流基本不给 C_{iss} 充电, 所以在 V_{GS} 曲线上出现一个平台, 这段时间内 V_{DS} 持续下降而 V_{GS} 基本保持不变, 这就是上面所说的上升时间 t_r 。等到 t_r 之后, 栅极电流才继续向 C_{iss} 充电, 最后 V_{GS} 才上升到最高。

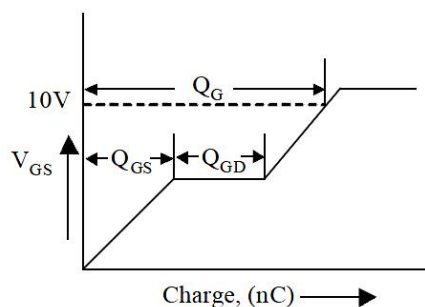


图 2-13 栅极充电波形 (摘自 FDD6637 datasheet)

另外, 在《Matching MOSFET Drivers to MOSFETs》一文中还提到, Q_g 对于计算场效应管的耗热和栅极尖峰充值电流也是相当重要的一个参数, 对于更大功率的场效应管的应用来说 (电动车等), 是一个不得不考虑的问题。想拓展一下知识的朋友可以阅读一下此文档。

➤ 6. 二极管特性 (Drain-Source Diode Characteristics)

N 管比较: (表 2-11)

参数名称	符号	IRLR7843 (N 管)		IRFR1205 (N 管)	
		测试条件	典型值	测试条件	典型值
二极管正向压降	V_{SD}	$V_{GS} = 0V$ $I_S = 12A$	1.0 V	$V_{GS} = 0V$ $I_S = 22A$	1.3 V
二极管反向恢复时间	trr	$I_F = 12A$	39 ns	$I_F = 25A$	65 ns
二极管反向充电量	Qrr	$di / dt = 100A/us$	36 nC	$di / dt = 100A/us$	160 nC

P 管比较: (表 2-12)

参数名称	符号	FDD6637 (P 管)		IRFR5305 (P 管)	
		测试条件	典型值	测试条件	典型值
二极管正向压降	V_{SD}	$V_{GS} = 0V$ $I_S = -14A$	-0.8 V	$V_{GS} = 0V$ $I_S = -16A$	-1.3 V
二极管反向恢复时间	trr	$I_F = -14A$	28 ns	$I_F = -16A$	71 ns
二极管反向充电量	Qrr	$di / dt = 100A/us$	15 nC	$di / dt = 100A/us$	170 nC

前面曾经提到, 这些功率型场效应管的应用中, 需要在漏源极之间并一个二极管, 厂家已经很贴心地帮你在芯片中把二极管集成进去了, 并且是稳压二极管, 可以保护 V_{DS} 不至于过大而击穿 MOSFET。

(3) 外围电阻的作用

一些比较具有专研精神的网友往往会在这四个电阻的作用与阻值的选取感到疑惑(见图 2-14), 关于它们的作用在论坛里讨论的不多, 不过也不是没有, 在网友 rei1984 的一篇帖子里, cancer, lwglxilixi 和 jnstsean 网友给出了一些很精彩的回答, 其中 lijieamd 兄还做了一些具有探索性的实验, 对笔者启发很大, 在此一并感谢。下面笔者将对这些电阻的作用做个整理并给出一些定性的分析, 若有不正确之处, 欢迎批评指正。

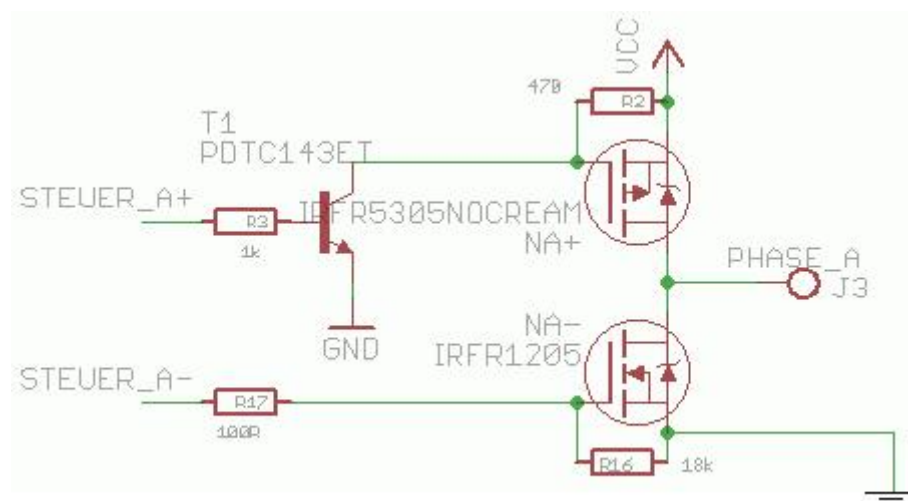


图 2-14 电调 A 相的上臂和下臂（摘自 MK 项目 V1.2 版本）

首先要明确一个概念，模拟电路不是软件编程、非 0 即 1，而是一个连续变化的过程。无论是电容上的电压还是电感上的电流，都不能突变，否则将产生灾难性的后果。上跳沿和下降沿并不是越陡峭越好，有时候在设计中甚至故意添加一些电阻让上升下降沿变得平缓以保护元器件。当然，如果你能在确保安全的情况下，尽可能提高上升下降沿的频率的话，你就是模电设计高手了。

先来看 R17，它有三个作用，其一是防止震荡，其二是减小栅极充电峰值电流，其三是保护 NA-场效应管的 D-S 极不被击穿。先来看第一点，一般单片机的 I/O 输出口都会带点杂散电感，在电压突变的情况下可能和栅极电容形成 LC 振荡，当它们之间串上 R17 后，可增大阻尼而减小振荡效果。第二，当栅极电压拉高时，首先会对栅极电容充电，充电峰值电流可大致计算为：
$$I = \frac{Q_g}{t_{d(on)} + t_r} = \frac{65nC}{(7.3 + 69)ns} = 0.85A$$
，可见已经超过了单片机的 I/O 输出能力，串上 R17 后可放慢充电时间而减小栅极充电电流。第三，当栅极关断时，NA-管的 D-S 极从导通状态变为截止状态时，漏源极电压 V_{DS} 会迅速增加，如果 dV_{DS}/dt 过大，就会击穿器件，所以添加 R17 可以让栅极电容慢慢放电，而不至于使器件击穿，100 欧是比较通用的做法。

接下来看 R16，其作用是下拉型抗干扰电阻。当单片机刚上电时，I/O 口一般都处于高阻态，如果没有 R18，栅极电压就处于悬浮状态，可能意外使场效应管导通，R16 的选值范围没啥特别的讲究，笔者曾经用 1.8k 和 180k 都仿真过，看不出对输出波形有明显影响。

再接下来看 R2，它的作用是上拉 NA+场效应管的栅极，其阻值不能太小，太小会造成三极管导通时承受过大的电流；同样，阻值也不能太大，太大会导致场效应管的栅极电压上

升缓慢，而影响开关性能。关于这点再多说两句，如果阻值太大，这个时候的影响场效应管开关速度的瓶颈不在 MOSFET 的栅极电容上，而是在三极管的 BC 极电容上！当 STEUER_A+ 从低电平变高电平后，VCC 首先会通过 R2 给三极管的 BC 极电容充电，如果 R2 太大会导致充电速度缓慢，从而导致 R2 两端电压变化缓慢，进而影响场效应管的栅源极电压上升速度。网友 lijieamd 兄做了个很有意义的实验：在三极管的 BC 极并了个钳位二极管，强制不给 BC 极电容充电，然后发现场效应管的上升下降沿都有明显的改善。此君现在正在研究导航算法，在论坛诸位四轴达人中实力不俗。lijieamd 写的关于调试无刷电机的两篇心得帖子都很有分量，链接在附录二中给出，大家有兴趣的话可以去看一看。

最后看 R3，是三极管的基极电阻，这个没啥特别的讲究，只要确保三极管能正常工作在放大区就可以了。

以上就是对各电阻作用的定性分析，如果有时间的话，笔者会对各电阻阻值和开关时间再作一个定量分析计算，以完善本篇文档，计算过程、仿真结果和电路实测结果将放于后续附录中。

2.3 电流检测电路

这是坛子里讨论比较多的话题，基本已有定论，Shunt 相当于一个 0.01 欧姆左右的小电阻，不过关于这值怎么来的好像没人提到，我这里就解释一下吧。

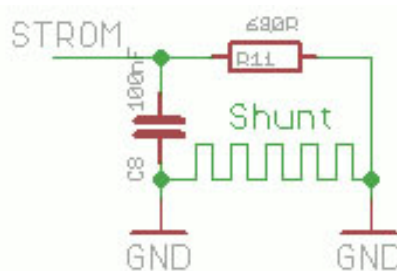


图 2-15 电流检测电路（摘自 MK 项目电调原理图 V1.2 版本）

Shunt 在德语中是“分流”的意思，STROM 在德语中是“电流”的意思（跟英文的 Stream 很像，呵呵，看来西语之间都是相通的呀）。其实在这里叫“分流”也不恰当，它在 PCB 板上是一段蛇形走线，相当于一个阻值很小的电阻，见图 2-16。经过 MOSFET 和电机的电流全都从它那里过，其实是“总流”，而不是分流。如果一定要将其以原理图的形式表现出来的话，应该是如图 2-17 所示。

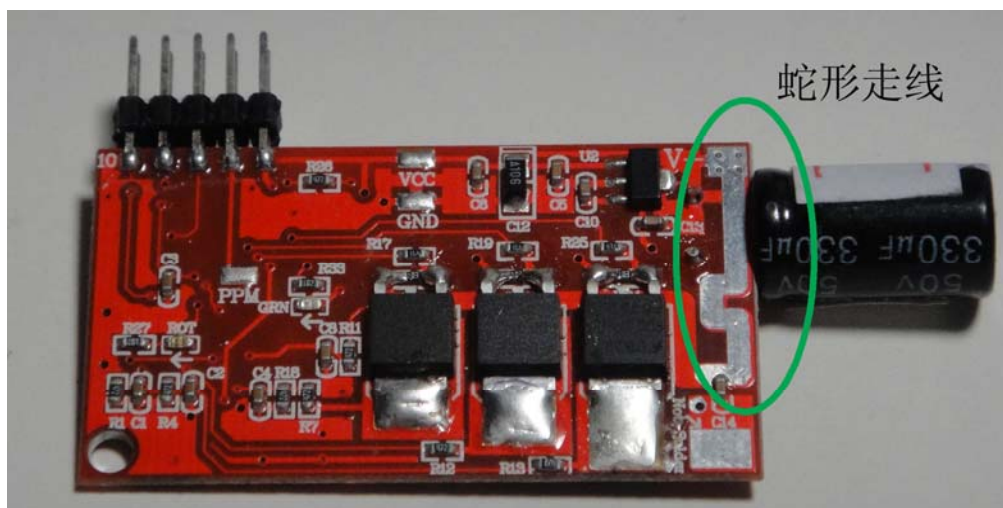


图 2-16 电路 V1.2 版本的成品电调（购自淘宝 西子书店）

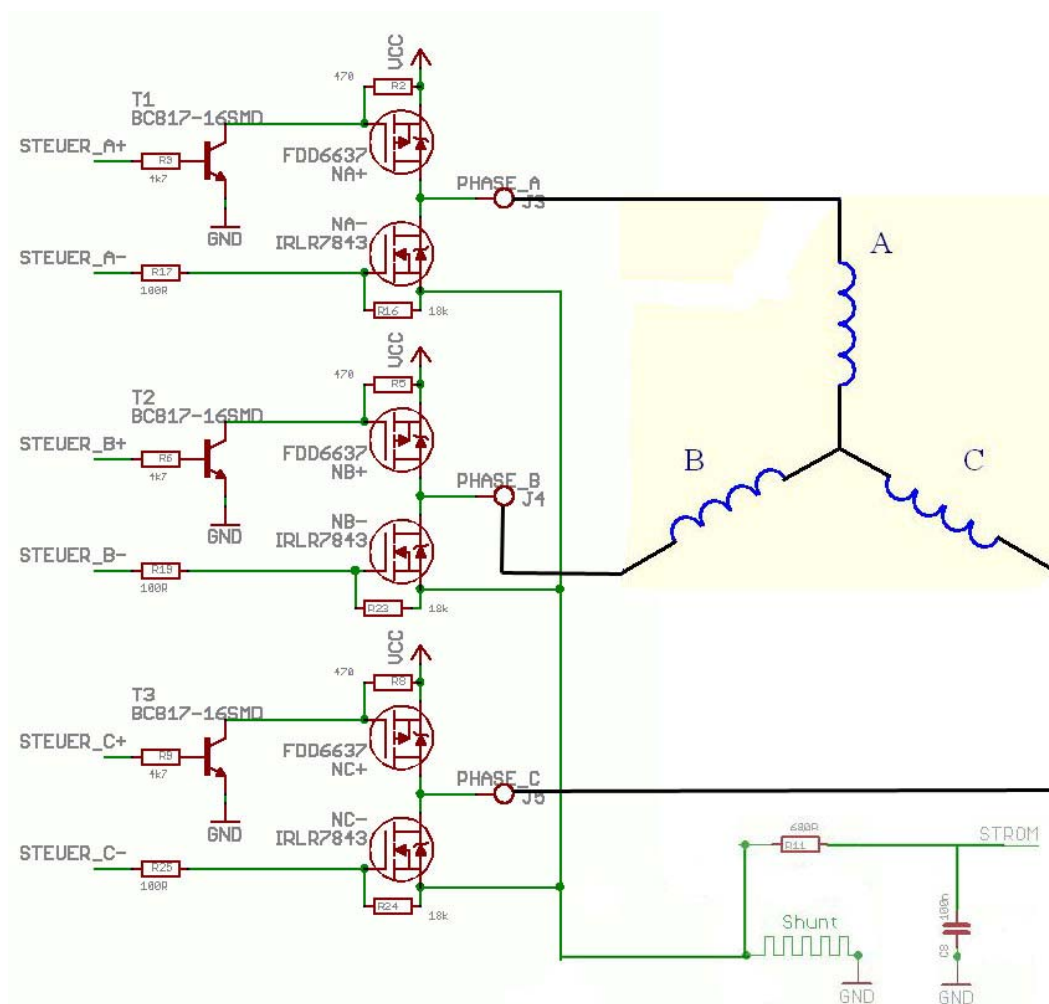


图 2-17 Shunt 走线的原理图

图 2-17 中，经过电机的总电流经过 Shunt 而流向 GND，尽管 Shunt 的阻值很小，但如

果电流够大的话，应该会在 Shunt 的左端产生一个小的电压，经过 R11 和 C8 一阶低通滤波，最后接入单片机的 ADC6 通道。那作者为什么在原理图上画得那么令人费解？目的是为了原理图生成 PCB 图时方便，因为 shunt 本质上是一段地线，而不是个元件。

那么这个 Shunt 的阻值怎么算？根据导体的电阻率公式： $R = \rho L / S$ ，其中 ρ 是特定导体的电阻率，L 是导线长度，S 是导体截面积。

先来看 ρ ，PCB 导线上一般敷的是铜，铜的电阻率查表可得： 1.75×10^{-8} 。再来看 L，蛇形走线不太好量，笔者用直尺大致量了一下，约为 4.5 厘米左右，也就是 4.5×10^{-2} m。

最后是横截面 S，等于线宽乘以线高，一般 PCB 板上敷铜的厚度都是 35um（不信你可以去问问你们公司的 PCB 打板供应商：），线宽这里我量了一下，大概为 2mm 左右。

好了，接下来就可以算了：

$$R = \frac{1.75 \times 10^{-8} \times 4.5 \times 10^{-2}}{3.5 \times 10^{-5} \times 2 \times 10^{-3}} = 0.011 \Omega。$$

有人也许会问，你这样算出来的值正确不正确啊，当然正确，且有诗为证，不对，是有程序为证：在电调程序（V0.41 版本）main.h 的第 26 行，有如下宏定义：

```
#define MAX_STROM          200 // ab ca. 20A PWM ausschalten
```

我们从 MK 作者的注释可以看到，若电流为 20A 时，在程序内部用 200 表示。再看 analog.c 文件的 void AdConvert(void) 函数的第 33 行，ADC6 通道采样到的原始值要乘以 4 才作为最终程序内部的电流表示值（后面的 35 行的作用是数字滤波，这个我们放待第三章软件分析时再讲）。

假设通过 shunt 的电流为 20A，根据计算的阻值，会在 shunt 上会产生 $20 \times 0.011 = 0.22$ V 的电压，单片机 ADC 采到到的原始值为： $(0.22 / 5) \times 1024 = 45$ ，再乘以 4 为： $45 \times 4 = 180$ 。基本接近程序内部 200 这个值的定义。考虑到前面测量蛇形走线的长度时产生的误差和其他因素，这个电阻的计算方法应该还是比较合理的。

至于后面的一阶 RC 低通滤波电路，是为了滤去一些偶尔产生的瞬时高频分量。V1.2 版本 R11 的阻值为 680 Ω ，而 V1.1 版本 R11 的阻值为 4.7k Ω 。相比之下，V1.2 版本放宽了截止频率。其 Bode 图比较见图 2-18，V1.2 版的截止频率约为 2341Hz，V1.1 版的为 339Hz。注意图中的频率单位为 rad/s，换算成 Hz 时还要除以 2π 。

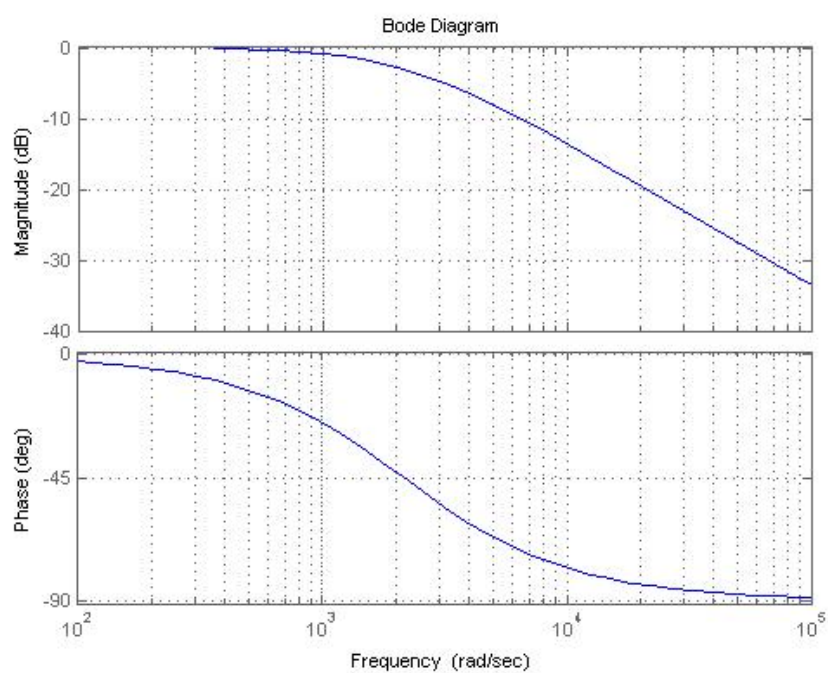


图 2-18(a) 电路 V1.1 版本 RC 低通滤波 Bode 图

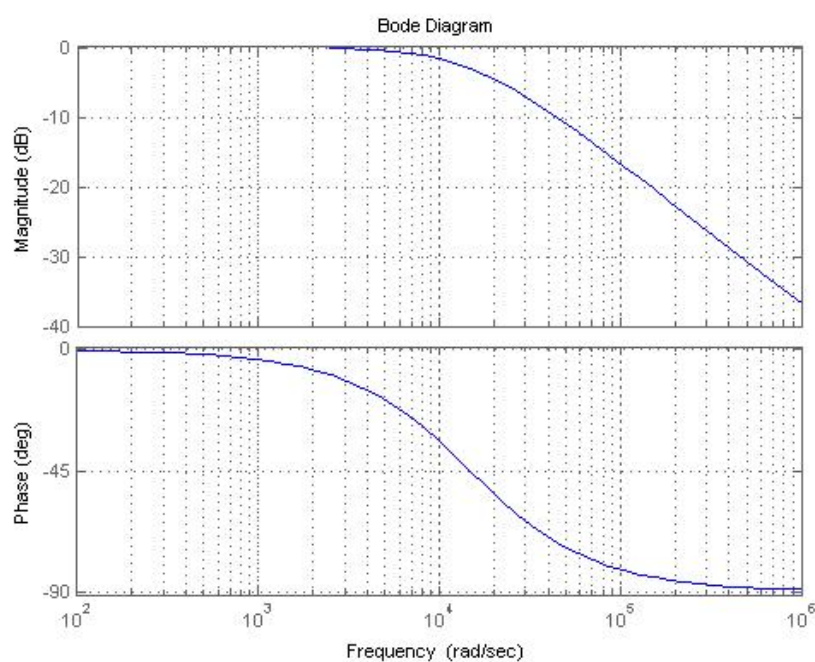


图 2-18(b) 电路 V1.2 版本 RC 低通滤波 Bode 图

2.4 反电势过零检测电路

前面说过，反电动势检测，需要不停比较中点电压和 A 相 B 相 C 相三个端点的电压，以截获每相感生电动势的过零事件。一般的无刷电机教材上都会给出一个三个比较器过零检测电路，其实，由于这三个过零事件产生的时间不同，如果能在比较器的输入端不断地切换这三个端点电压，那其实只要复用一個比较器就可以了，而 MEGA8 自带的模拟比较器正好提供了复用功能。当复用功能启动时，模拟比较器的正向输入端为 AIN0 引脚，负向输入端可以根据 ADMUX 寄存器的配置而选择 ADC0~ADC7 任意一个管脚。而这个外下面我们来看 MK 电调的。

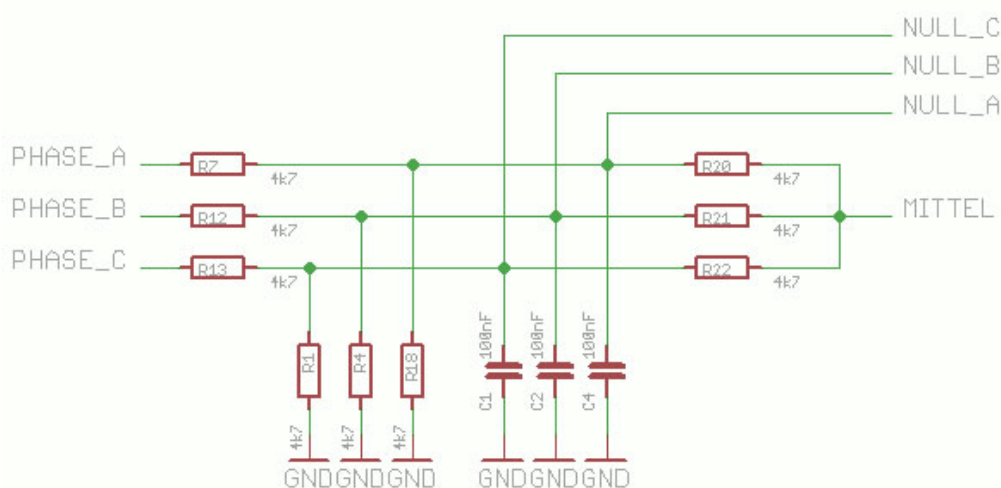


图 2-19 反电势过零检测电路（摘自 MK 项目电调原理图 V1.2 版本）

图中，PHASE_A, PHASE_B, PHASE_C 分别接电机的 A, B, C 线，经过一个分压网络后分别为 NULL_A, NULL_B, NULL_C，再连接到单片机的 ADC0, ADC1, ADC2 引脚。而 MITTEL 为估测的变形后的中点电压，接单片机的 AIN0 引脚。只要在 AB 通电期间开通 NULL_C 和 MITTEL 的比较；AC 通电期间开通 NULL_B 和 MITTEL 的比较；BC 通电期间开通 NULL_A 的比较，就可以成功检测出各相的过零事件。

下面来说说这个分压网络到底是怎么回事，这是个比较经典的反电动势采集电路，很多无刷电机的教材上都有照搬，我想应该也不是 MK 项目那个德国人的原创。现在我们来分析，先无视图中这几个电容，把它当成纯电阻网络。假设 AB 相开始通电的时候，PHASE_A 的电压约为 12V，PHASE_B 的电压约为 0V，C 线圈此时产生 6V 的反向感生电动势，叠加在绕组中点上后，在 PHASE_C 输出的电压应为 12V 左右，问：此时 MITTEL 点电压值是多少？NULL_C 点电压值又是多少？

基尔霍夫定律学得好的朋友可以手算，学得不好的朋友，呃……可以用电路软件仿真。答案是：MITTEL=4V，NULL_C=5.3V。此时 NULL_C 点电压高于 MITTEL 点电压，模拟比较器输出 AC0 为 0。接着转子继续向前转动，C 线圈的感生电动势逐渐减小，当减小到零时，PHASE_C 端输出的电压就是绕组中点电压 6V，此时可以算得：MITTEL=3V，NULL_C=3V，可见此时模拟比较器处于零界状态。随着转子继续转动 C 线圈产生的电动势方向变反，PHASE_C 端测得的电压会小于中点 6V，假设现在 C 线圈产生-1V 的电动势，则 PHASE_C 端测得的电压为 5V，放到这个电阻网络里一计算，结果得：MITTEL=2.8V，NULL_C=2.6V。此时 MITTEL 点电压高于 NULL_C 点电压，说明刚才模拟比较器的输出已发生了一次跳变，过零事件被检测到。

再来看图中的几个电容，基本作用就是用来滤除 NULL_A, B, C 三点的高频分量的，坛子里有网友认为它们会造成被测电压一定程度的滞后，笔者倒认为这几个电容加不加关系都不大，一是后面软件里面有滤除高频杂波电压的措施，二是参考附录二中凤凰商业电调的设计，也是没有这几个电容的。

2.5 制作你自己的电调线路板

为了更好地继承前人的经验，笔者花了一周时间翻看了四轴分论坛十几页的旧帖。感觉在 2008 年前后，那时候的四轴论坛端的热闹。一时间，各路高手达人群集于此，纷纷使出解数，重画原理图的重画原理图、PCB 打板的打板、改程序的改程序，豪气云天，颇有当年大炼钢铁之势，呵呵。不过，调侃归调侃，在下可丝毫没有贬低前人的意思；相反，笔者对这些四轴的先行者还是保持相当的敬意的，没有他们提供的初期摸索的宝贵经验，笔者是不可能写出这篇文档的。

说了前面这么多，笔者想说的意思是，如果你想比较彻底地掌握电调技术，可以一开始不用急着画 PCB 线路板，而是先用万用电路板（俗称“洞洞板”）搭接一个电调出来，体积可以做得比较大也没关系。这样，一方面元器件若在实验中烧坏比较容易更换；二来，一些比较关键的点可以引出接线柱方便测量观察。如果你纯粹是想以研究飞控算法为主，也没必要专门自画 PCB，淘宝上有很多仿 MK 项目的电路板出售，而且价格也还算合理，一般它们都自带烧程序的接口和 bootloader，这样你就可以专心把精力放在研究软件算法方面而不必关注硬件问题。不过如果你打算 DIY 出来卖……呃……还是自己画板好了。

自己在万用板做电调时候，可能唯一碰到的难题就是那个 shunt 电阻的问题了，其实也不难解决，可以用一段比较细且长的康铜丝或锰铜丝（其他极细导线也可以），直接并在直流电源两端，然后慢慢调高电压，并注意观察电流，一般可以根据欧姆定律算出这段导线的电阻，然后再按比例剪短就是需要的阻值了。

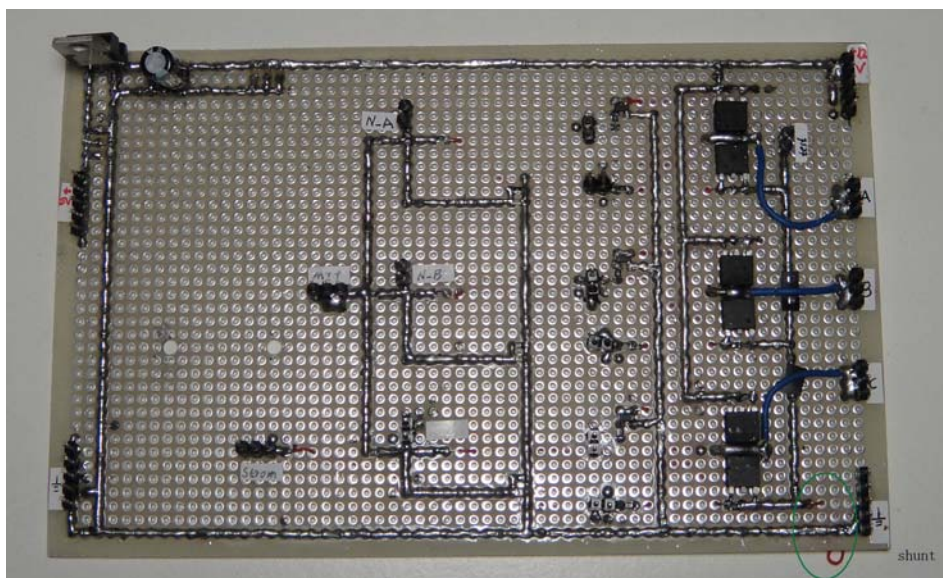


图 2-20 笔者自行焊接的电调实验板（右下角绿圈中为 shunt 导线）

关于场效应管的选择，最好用 V1.2 版本的 IRFR1205 和 IRFR5305 而不要 V1.1 版本的 FDD6637，因为据网友反映，市场上被 Remark 过的假 FDD6637 较多，大家注意甄别。

最后要强调的一点是：机电一体化实验不是电子实验，**是有一定危险性的**！高速旋转的螺旋桨无异于两把利刃，千万要注意安全。网友 leoli321 在测试电机在运行中堵转时，食指不小心碰到电机的风叶，被螺旋桨利刃割开了一道长约 10mm 的伤口，他在帖子中很善意地提醒大家，一定要注意安全，lijieamd 网友也曾专门发帖提醒大家注意安全。另外，玩模型的都知道，如果螺旋桨质量不好，在高速旋转中可能发生断裂，这就是所谓可怕的“射桨”。被巨大的离心力甩出来的桨叶，就是一柄飞刀，如果碰到人（不管是自己还是房间内的其他人），必定会产生伤害事故。所以有条件的话，最好在螺旋桨和人之间，放置一块有机玻璃；并且在调试时应带上防护眼镜（如图 2-21 所示），这种眼镜不贵，也就二三十左右，好比安全带一样，关键时候能帮你大忙。再次重申一下：DIY 时千万不要忽视安全问题！



图 2-21 防护眼镜

3. 无感无刷电调的软件设计

上面看完了电调的硬件设计，下面我们来看看电调的软件设计。同样还是以开源的德国 MK 项目的电调为蓝本，程序版本以 V0.41 为准，附录三给出了其下载地址。为使分析层次更加清晰，本章将把 MK 电调程序中关于电机驱动的部分单独抽出来，分几个主题进行讲解。而第四章则更侧重于程序的整体流程分析和其他一些功能模块的描述。小提示：在阅读本章时，最好旁边打开一个程序编辑窗口和本文并排显示在同一屏幕中，这样可省去窗口来回切换的麻烦：)

3.1 电流检测

还是老规矩，先来简单的模块热身。电流检测的程序模块和整个软件的其他部分关联较小，所有代码都在 `analog.c` 文件中，比价适合于单独抽出来讲解。其中主要有四个函数，下面分别说明：

1. `void ADC_Init(void)`

这个函数位于 `analog.c` 文件的第 9 行，比较简单，功能为初始化单片机的 ADC 模块。唯一可能令人感到比较困惑的是函数中第 2 行的 `IntRef` 变量。这个其实涉及到 V1.2 版本电路和 V1.1 版本电路的不同设计。对比原理图你会发现，V1.1 版本采用外部 5V 接在 `AREF` 引脚做参考电压，而 V1.2 版本则采用了片内的 2.56V 的基准电压，所以 `AREF` 引脚是悬空的。所以在 `main.c` 中，主程序先会判断本电路是 V1.1 版本还是 V1.2 版本。若是 V1.1 版本，则全局变量 `IntRef` 的值为 0，若是 V1.2 版本电路，则主程序会将 `IntRef` 的值设为 0xC0。至于如何判断是 V1.1 电路还是 V1.2 电路，这个我们放到下一章再讲。

2. `uint MessAD(uchar channel)`

这个函数位于 `analog.c` 文件的第 45 行，功能是测量某一 ADC 通道的电压值并将其返回。入参 `channel` 为要测量的通道号。本函数流程中规中矩，没什么特别难的。需要注意一下的

是,第 49 行先暂时保存原来的通道号,最后在 58 行还要恢复原来通道号。为什么这样做呢?这是由于 MEGA8 的 AD 转换器(以下简称 ADC)和模拟比较器(以下简称 AC)是复用某些管脚的,所以这个 ADMUX 寄存器不单是给 ADC 用的,有可能在调用本函数的时候,AC 也在用 ADMUX,所以在本次 ADC 测量完成后,还要把原来的通道号恢复回去。第 50 行 IntRef 的作用前面已讲过。第 57 行,这里是用轮询的方式来等待 ADC 转换结束的,而不是用中断的方式,虽然效率低了一点,不过问题也不大。第 59 行关闭 ADC,第 60 行恢复 AC 的管脚复用功能,第 61 行返回测量值。

3. void FastADConvert(void)

本函数位于 analog.c 文件的第 66 行,本函数的功能是快速测量 shunt 上的电流大小(shunt 作用见第二章)。在第 70 行可以看到,调用 MessAD(6)得到的返回值是要乘以 4 的,接着将其放入全局变量 Strom。这里 Strom 的值被限制为最大为 200,全局变量 Strom_max 总是等于历次所测到的最大 Strom 值。最后第 75,76 两行有点画蛇添足。

4. void ADConvert(void)

本函数位于 analog.c 文件的第 19 行,功能为测量 shunt 上的电流大小。可以看到,本函数中的绝大部分语句和 FastADConvert + MessAD 是相同的,唯一不同的就是第 35 行:

```
Strom = (I + Strom * 7) / 8;
```

这句话的作用是数值滤波,减弱一些尖峰电流对测量值的影响。其含义为:将本次测量的值(即 i 值)仅取 1/8 权重,而将以往的 Strom 值取 7/8 权重,最后结果放入 Strom。

3.2 定时器延时与 PWM 信号

定时与延时在整个电调程序中是比较基础的功能,后面的很多功能模块都会用到定时或延时功能,包括 PWM 信号的产生等等也都和定时器有关,所以有必要先建立一个认识。关于 MEGA8 的三个定时器,MK 的作者是这么分配的,T0 作为系统时钟定时器用,T1 和 T2 都用作 PWM 的发生器用。

1. 定时器初始化

在 timer0.c 文件的第 35 行的 `void Timer0_Init(void)` 中，一共就三行，言简意赅。第一句设 T0 为 clk 的 8 分频（也即定时计数器的计数频率为 1MHz）。第二句使能 T0 的溢出中断功能，第三句使能 T2 的溢出中断功能。事实上，作者后来并没有用到 T2 的溢出功能，T2 的溢出中断服务程序只是个空架子。

2. 定时器 T0 的溢出中断服务程序

在 timer0.c 文件的第 20 行的 `SIGNAL(SIG_OVERFLOW0)` 中。查看 MEGA8 的数据手册可知，T0 每次从 0 计数到 255 产生中断，然后再从零开始计数。故每次溢出中断产生的周期为 256us。第 23 行是个没用的全局变量，这里不管它。由于溢出周期为 256us，那每溢出 4 次就相当于大约 1ms，时间，此时全局变量 `CountMilliseconds` 自加 1，说明已过了 1ms。后面的 `I2C_Timeout`、`PPM_Timeout`、`SIO_Timeout` 分别是给 I2C、PPM、串口通信用的，这里也暂时不管，等讲到通信模块时再回过来看。

3. 利用 T0 延时（毫秒级）

如果要进行毫秒级的延时，主要是利用 `uint SetDelay(uint t)` 和 `char CheckDelay(uint t)` 这两个函数来完成，分别在 timer0.c 的第 44 行和第 49 行。具体要怎么做呢，下面我们来看一个情景分析。假设我现在要延时 100ms，就应该这么写：

```
unsigned int a;
a = SetDelay(100);
while (!CheckDelay(a));
```

当前的全局变量 `CountMilliseconds` 有可能为任意值，我们就假设当前的 `CountMilliseconds` = 12345 好了。先调用 `SetDelay(100)`，返回 12345+100-1 到 a 中，a 的值就为 12444。然后再在 while 中反复调用 `CheckDelay(a)`，返回值为 `(12444 - CountMilliseconds) & 0x8000 >> 8`，现在 `CountMilliseconds` = 12345，故返回值为 0，继续在 while 里循环。当时间渐渐过去，`CountMilliseconds` 会渐渐变大，到过了 100ms 后，`CountMilliseconds` 的值会走到 12445，由于无符号整形数减法的特性，`(12444 - CountMilliseconds)` 会借位而产生一个非常大的数，然

后这时候就会返回非零，退出 `while` 循环。而且由于无符号数加法的特性，即便在调用 `SetDelay` 时 `CountMilliseconds` 已接近 65535 了也没有关系，读者可自行理解。第 54 行的 `void Delay_ms(unsigned int w)` 函数帮你把这三句话都集成到一个函数中了。

4. 利用 T0 延时（微秒级）

在 `main.c` 文件第 140 行的 `void Wait(uchar dauer)` 函数可以完成微秒级延时。前面曾经说过，`TCNT0` 的计数频率为 1MHz，也就是每过 1us 会自加 1。在 `Wait` 这个函数中同样是利用无符号整形数的减法特性来实现延时的，原理和上面类似，读者可自行分析。延时极限为 255us。

5. PWM 信号的产生

前面说过，A, B, C 的三路 PWM 信号主要靠 T1 与 T2 产生，在 `main.c` 文件的第 127 行的 `void PWM_Init(void)` 函数中完成了 T1 与 T2 的初始化。

在函数中，`PWM_OFF` 是一个宏，这个宏我们后面会分析，其中的 `TCCR2 = 0x41`；完成了对 T2 的初始化：`WGM2[1:0]=1` {T2 工作于“相位修正 PWM 模式”，`COM2[1:0]=0` {`OC2`(接场效应管 A+)作为普通端口，暂时不作 PWM 输出}，`CS2[2:0]=1` {T2 的时钟源为 `clk/1`，不预分频}。所以 T2 的 PWM 频率就为： $8\text{MHz} / 512 = 16\text{kHz}$ 。如果使用外部 16M 的晶振，T2 的 PWM 频率就为 32kHz。

这个宏中的 `TCCR1A = 0x01`；与函数中第 131 行的 `TCCR1B=.....`；两句话完成了对 T1 的初始化，T1 的初始化内容读者可参照 MEGA8 数据手册自行分析，这里不再赘述，T1 的 PWM 频率也为 16kHz。

当这个函数运行完后，T1 和 T2 就开始工作计数了，只是这时候 `OC1A`, `OC1B`, `OC2` 都是 0，PWM 信号还并未产生。

只有当运行 `void SetPWM(void)` 函数后（`main.c` 文件第 84 行），MEGA8 内部才会产生真正有效的 PWM 信号，但此信号仅限于芯片内部，还不会输出到管脚，只有再当运行 `PWM_A_ON` 宏或 `STEUER_A_H` 宏后，才将这个 PWM 信号真正接通到管脚，在管脚产生 PWM 波形。这些宏在下一小节会详细分析。

SetPWM 函数比较简单，功能是将全局变量“PWM”的值设置到 OC1A, OC1B 与 OC2 中。PWM 变量取值范围为 0~255，0 表示占空比 0%，255 表示占空比 100%，中间的按比例折合。第 89~93 行判断将要设定的 PWM 值是否超过最大值，这里全局变量 MaxPWM 的值为 255，若超过则红灯亮，这里 MaxPWM 的值定义为 255，故从来不会超的。第 94~112 行判断当前的电流值 Strom 是否大于 MAX_STROM 宏，若大于则关掉所有 MOSFET，红灯亮。这里关于 MAX_STROM 宏的定义比较有意思，参看 main.h 的第 25~31 行。假设当前是以 16kHz 的 PWM 频率运行，看第 25 行，#if FDD_IRLR == 1，我们从第二章的电路知道电调电路的 V1.1 版本设计是用 FDD6637 与 IRLR7843 场效应管的，故 MAX_STROM 定义成 200，如果用的是 V1.2 版的设计，MAX_STROM 就定义成第 29 行的 130，再次印证了我们关于“V1.2 版本电流驱动能力比较弱”的结论。若一切都每哟问题，则运行第 102 行 else 中的内容，将全局变量 PWM 中的值，设置进 OCR1A, OCR1B 和 OCR2。

3.3 过零事件检测与电机换相

无感无刷电机控制的主要内容就是检测悬浮相的感生电动势的过零点，当模拟比较器发生中断时说明过零事件产生，然后准备换相。在第一章已经说过了，德国 MK 电调的换相策略是一过零就换相，算法简单，缺点是稍微损失一点效率，不过如果电机的极对数够多，损失的效率会很小。关于过零检测和换相的代码都在 BLMC.c 文件中，其中用到了大量的宏，这些宏都定义在 BLMC.h 文件中，我们先把这些宏的功能都看一遍（注意要对照电路看）。由于笔者用的是 16kHz 的 PWM 频率，故关于 32kHz 的定义都会略过，两者的代码都是类似的，自己看看应该可以触类旁通。

1. BLMC.h 中定义的宏

➤ #define PWM_C_ON {TCCR1A=0xA1; TCCR2=0x61; DDRB=0x02;}

功能：使 OC1A 引脚（接场效应管 NC+）输出 PWM 信号。TCCR1A 和 TCCR2 的定义查一下手册便可，理解起来应该不难，这里 T1 和 T2 都采用的是“相位修正 PWM 模式”。注意最后一句 DDRB=0x02; 会在 STEUER_C+ 输出 PWM 信号的同时将 STEUER_A+ 和 STEUER_B+ 都关掉。后面的 PWM_B_ON 和 PWM_A_ON 结构基本相

同，读者可自行分析。

- `#define PWM_OFF {OCR1A=0;OCR1B=0;OCR2=0;TCCR1A=0x01;TCCR2=0x41; DDRB = 0x0E; PORTB &= ~0x0E;}`

功能：关掉 STEUER_A+, STEUER_B+, STEUER_C+三路 PWM 信号。而且关得比较狠，连输出比较寄存器 OCR1A, OCR1B, OCR2 都一起清零了。最后将 OC1A, OC1B, OC2 三个引脚都变成普通 GPIO 口，并输出低电平。

- `#define STEUER_A_H {PWM_A_ON}`

功能同 PWM_A_ON，没啥好说的。

- `#define STEUER_A_L {PORTD&=~0x30; PORTD|=0x08;}`

功能：STEUER_A-路输出高电平（即打开 NA-场效应管），同时关断 NB-和 NC-场效应管。

- `#define STEUER_OFF {PORTD&=~0x38; PWM_OFF;}`

功能：关断所有的 6 个场效应管，并终止内部的 PWM 信号。

- `#define FETS_OFF {PORTD&=~0x38; PORTB&=~0x0E; }`

功能：临时关断所有的 6 个场效应管，但内部的 PWM 信号不停。

- `#define SENSE_A ADMUX = 0|IntRef;`

功能：将 ADC0 管脚复用为模拟比较器的反向输入端。其中 IntRef 已经在前面解释过了，后面的 SENSE_B, SENSE_C 也类似。

- `#define ClrSENSE ACSR|=0x10`

功能：清零模拟比较器的中断标志，不过这个宏在整个程序中没用到。

- `#define SENSE ((ACSR & 0x10))`

和前面的动作宏不同，这是个判断宏，若模拟比较器中断标志置位，则此表达式的值为

非零，否则表达式的值为零。

➤ **#define SENSE_L** (!(ACSR & 0x20))

这也是个判断宏，若比模拟比较器的正向输入端（也就是 MITTEL）电压高于被复用的负向输入端（NULL_A 或 NULL_B 或 NULL_C），则模拟比较器输出（ACO）为高电平，这个表达式再行取非，整个表达式的值为零。反之，当 MITTEL 的电压低于 NULL_X 时整个表达式的值为真。

➤ **#define SENSE_H** ((ACSR & 0x20))

和刚才的表达式相反，当 MITTEL 电压高于 NULL_X 时，表达式的值为真；反之当 MITTEL 的电压低于 NULL_X 时，表达式的值为零。

➤ **#define ENABLE_SENSE_INT** {CompInterruptFreigabe=1; ACSR|=0x0A;}

功能：使能模拟比较器中断。这里的 ComInterruptFreigabe 是一个全局变量，不过在程序其他地方都没有用到。后一句 ACSR=0x0A，笔者认为应该换成 ACSR=0x08，更为妥当，不过两者都是可以工作的。

➤ **#define DISABLE_SENSE_INT** {CompInterruptFreigabe=0; ACSR&=~0x08}

功能：禁能模拟比较器中断。

➤ **#define SENSE_FALLING_INT** ACSR &= ~0x01

功能：设置模拟比较器为下降沿产生中断。

➤ **#define SENSE_RISING_INT** ACSR |= 0x03

功能：设置模拟比较器为上升沿产生中断。

➤ **#define SENSE_TOGGLE_INT** ACSR &= ~0x03

功能：设置模拟比较器为任何输出变化都可触发中断。

2. 过零检测与换相代码分析

在 BLDC.c 文件中有两个函数例程，一个是模拟比较器的中断服务例程（第 74 行），另一个是 Manuell 函数。其中 Manuell 函数我们放在后面的“启动算法”一小节讨论，这里主要分析模拟比较器的中断服务例程：SIGNAL(SIG_COMPARATOR)。

现假设电机已处于正常旋转状态，每次检测到悬浮相的过零事件后，就会触发进入本段代码。下面我们来假设一些情景进行分析，假设目前电机正处于 AB 相通电状态，转到一半时 C 相的感生电动势会过零（参考图 1-24），此时 NULL_C 端的电压会低于 MITTEL 中点电压，模拟比较器的输出端产生一个上跳沿而触发中断，进入中断服务程序。

进入中断服务程序后，首先会碰到一个 do-while 循环，这个是用来过滤消磁事件的，我们放到后面讲，这里先不管。再下去到第 80 行会判断 SENSE_H 宏的值，此时为真，故局部变量 sense=1。然后进入 Switch(Phase)条件判断语句，Phase 是一个全局变量，表示当前通电相位，0~5 分别表示：AB, AC, BC, BA, CA, CB 相通电。当前是 AB 相通电，Phase 的值为 0，故进入 case 0 条件。

进入 case 0 后，先在第 84 行对 STEUER_A+输出 PWM 信号，接着第 85 行有一个 if 语句判断 sense，这个 if 判断是为了消除消磁事件的影响，等会儿细讲。现在 sense 值为 1，故进入 if 语句。再在第 87 行打开 NC-场效应管（STEUER_C_L 宏会同时关断 NA-和 NB-管），如此便完成了从 AB 到 AC 的换相过程。接着在第 88 行，判断全局变量 ZeiZumAdWandelen 的值，若为非零，则调用 AdConver()函数采样一次电流大小。在第四章我们会看到，在主程序中每隔一定的时间就会将 ZeiZumAdWandelen 置 1，以达到定期监控电流的目的。

然后在第 89 行和 90 行将模拟比较器设置成监控比较 B 相电压 NULL_B 和中点电压 MITTLE，监控方式为下跳沿产生中断。参看图 1-24，由于在 AC 相通电期间，B 相电压应该从低于中点电压变为超过中点电压，而且由于 NULL_B 是接在比较器的负端，故在 B 相电压超过中点时，模拟比较器会产生一个下跳沿。

接着在第 91 行将变量 Phase 加 1，第 92 行将全局变量 CntKommutterungen 也加 1，这个变量是用来累计换相次数的。然后 break 跳出 switch 语句，转到 186 行的 while 语句处。此时 sense 仍旧为 1，而此时的模拟比较器的负向输入端接的已是 NULL_B 了，故 SENSE_L 会比较 NULL_B 和 MITTLE 的电压，此时 NULL_B 电压低于 MITTLE 电压，故 SENSE_L 表达式的值为零，退出 do-while 循环。再将全局变量 ZeitZumAdWandeln 的值设成 0，退出中断服务程序。过零检测加换相事件完成，一气呵成。

现在先来看看为什么要前面这个 if 判断。图 1-24 表示的是理想的电流和感生电动势，只考虑了导线切割磁力线产生的感生电动势，而没有考虑线圈自身的电感。在实际的 AB 到 AC 的换相过程中，由于 B 相电流突然减小（不会立即消失，还会续流一段时间，直到本身能量耗尽），其线圈的自身电感会在续流期间，成为一个电动势产生者，而且方向和原来相反，并叠加在中点之上，见图 3-1，此时 B 端电位是高于中点电位的。注意(a)图中 B 线圈的感生电动势是导体切割磁力线产生的，而(b)图中的续流电动势是 B 线圈自身的电感产生的（其大小要高于切割磁力线的感生电动势大小）。

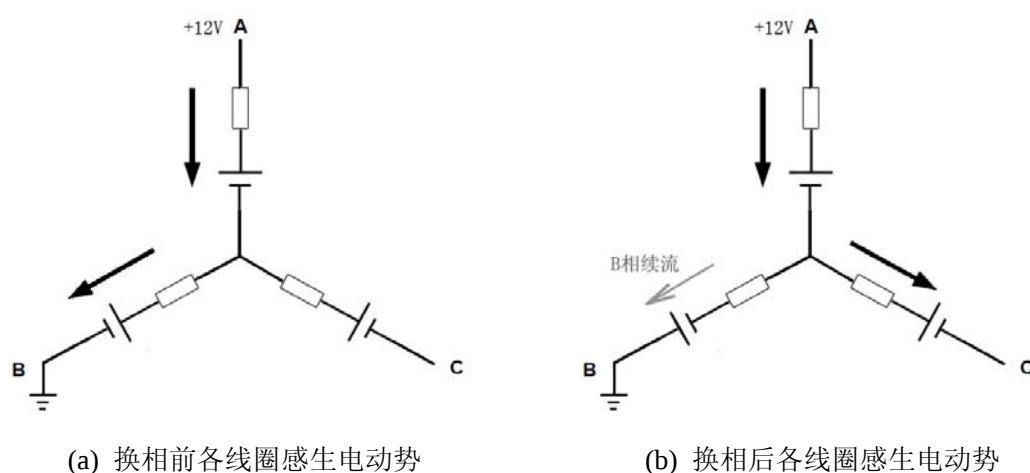


图 3-1 从 AB 到 AC 换相过程中各线圈的感生电动势

当 B 线圈能量耗尽后，切割磁力线的行为再次成为主导 B 相感生电动势的主要因素，所以 B 端电位此时又会低于中点电位，这就是所谓的“消磁事件”。如果要看 B 端波形的话，应该是像图 3-2 这个样子的（图中为在 100% 占空比驱动下，类似直流电压，而非 PWM）。

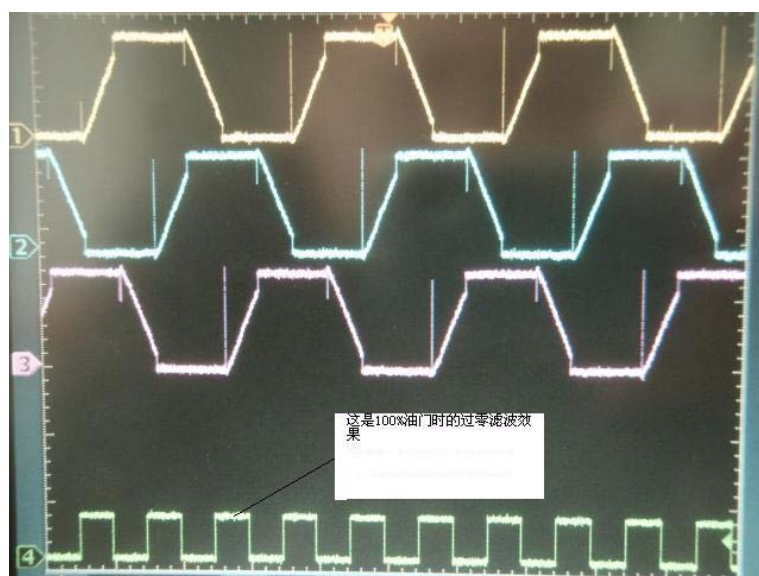


图 3-2 各相感生电动势的实际波形（摘自网友 dasuantou 的实验结果）

当用 PWM 信号驱动时，同样会产生消磁事件，网友 lijieamd 兄也在自己的实验中观测到了这一现象，见图 3-3。

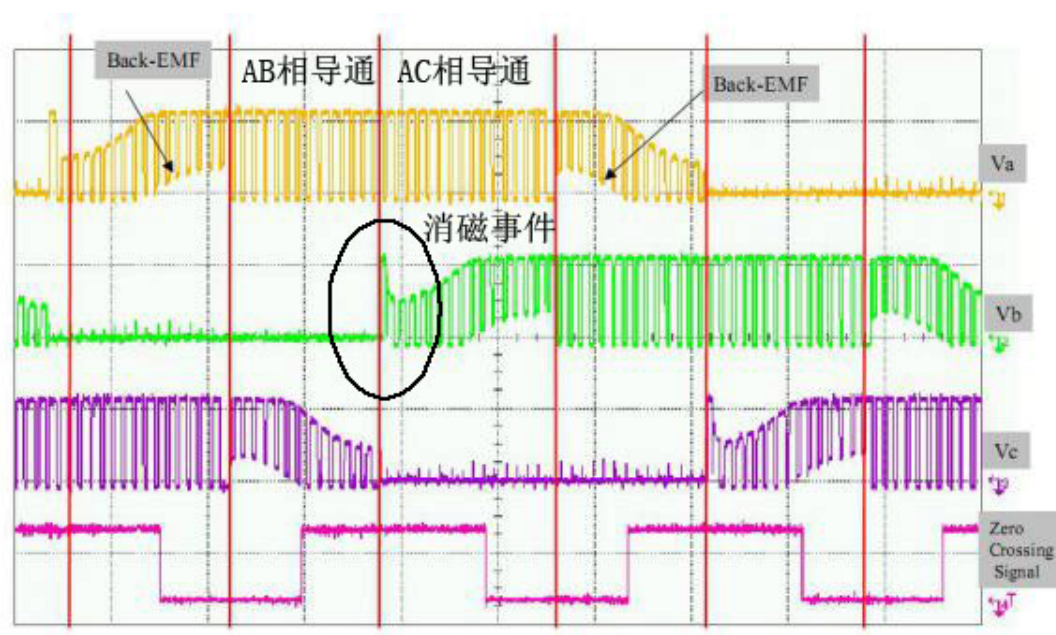
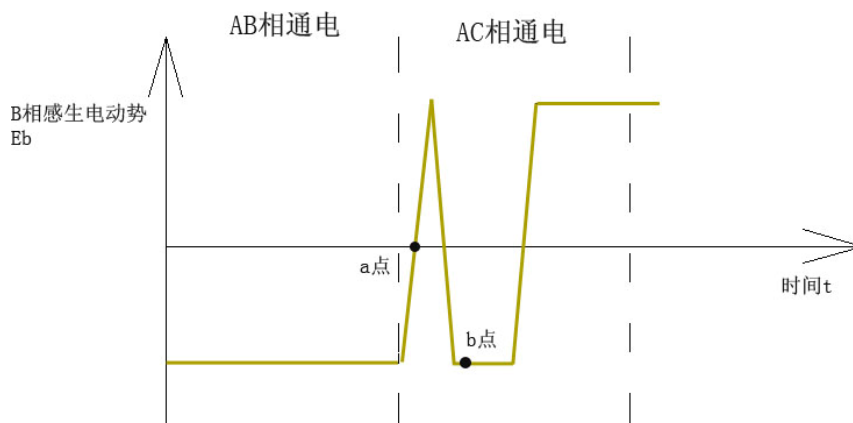


图 3-3 PWM 信号驱动时的消磁波形（摘自网友 lijieamd 实验结果）

由于消磁事件的存在，B 端会在 AC 相通电期间，产生两次上冲过零事件（也即模拟比较器会捕捉到两次下跳沿），前一次是假的，后一次才是真的。由于这第一个上冲事件的脉冲十分短暂（应该在 ns 级），假设在 AC 相通电期间，模拟比较器捕捉到 B 相一次上冲事件，位于图 3-4 的 a 点，然后单片机还要做一些保存堆栈指针啊、PC 跳转啊之类的乱七八糟的事，等到运行到中断服务程序的第 80 行的时候，实际的波形已经到图 3-4 的 b 点了，这时 SENSE_H 的值为真，故 sense=1。由于现在是 AC 相通电，Phase 为 1，故跳到第 99 行的 case 1 条件，因为此时 NC-管已经打开了，故第 100 行没起什么作用。然后在第 101 行的 if 判断会进入 else 部分，故仍旧保持 AC 相通电状态，不换相。之后 break，到 186 行的 while 表达式，此时 SENSE_L 值为 0，退出 do-while 循环。可见，由于 if 和 sense 的存在，这次短暂过零上冲事件并没有对换相造成影响。而且对于 B 相检测期间其他的瞬时过零干扰脉冲也有很好的抑制作用。

图 3-4 B 相感生电动势 E_b 的波形

下面再来看看这个 `do-while` 有什么用，还是以刚才的 B 相检测为例，在 B 相真正过零的时候，可能波形并没有理想中那么漂亮，而是存在一些波动，见图 3-5 所示。假设在 a 点模拟比较器检测到过零事件，但中断服务程序运行到第 80 行的时候，实际的波形已到了 b 点，按照我们刚才的分析，`if` 判断会把它当成一种干扰事件而忽略，程序继续往下运行到 186 行的时候，实际波形已到了 c 点。如果此时没有 `while` 的判断语句，接着就会退出中断服务程序。但此时波形已到 c 点，真正的过零事件已成为过去时，检测不到了。但是如果在 c 点的时候有 `while` 语句，此时 `SENSE_L` 的判断值为 1，`sense` 变量也为 1，就会跳回到上面第 80 行再来一遍，此时 `SENSE_H` 判断值为 0，`sense=0`，然后再进入第 101 行的 `if` 语句就可以顺利换相了，而不会漏掉此次过零事件。

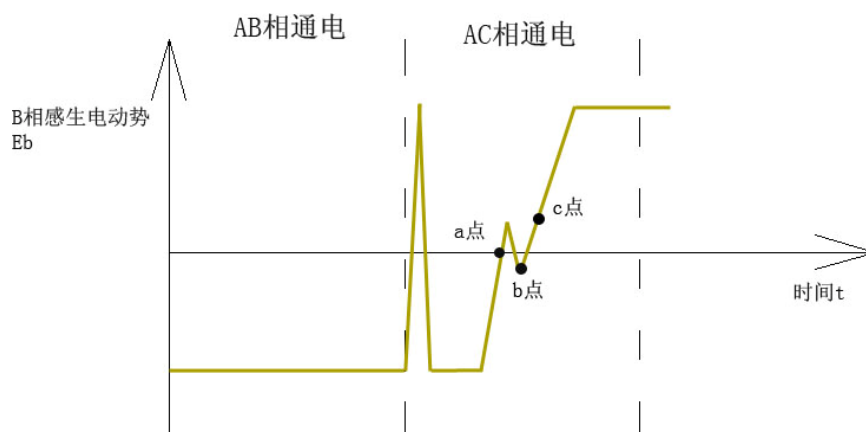


图 3-5 B 相过零时的波动

3.4 启动算法

无感无刷直流电机的启动比较麻烦，其原因就在于反电动势检测法只有在电机正常转起来之后才能正常工作，在电机静止状态，你是无法知道电机转子当前的实际位置的，从而也无法判断到底该通哪两相电进行驱动。而且在低速状态下，反电动势输出极低，波形也比较乱，从而也无法用反电动势过零检测法进行换相。你必须确保在两眼一摸黑的情况下，使电机运行到一个比较快的速度，等反电动势波形输出正常了，然后才能利用刚才一小节的过零检测换相法使电机正常工作和换相。

关于无感无刷直流电机的启动算法是一个难点，也是个学术热点话题，关于它的论文多如牛毛，笔者也参阅了一些，可惜全是泛泛而谈，都只给出了些概念，并没有给出实际的可操作的方法。看来看去，还是 MK 项目的那个德国哥们比较厚道，在程序中给出了一个实际的可用的启动程序范例。代码不长，只有几十行，但其中包含的原理还是需要颇费一番脑筋才能理解的。记得坛子里有位海绵宝宝网友说过，这些东西只是江湖一层纸，戳穿不值半文钱，那现在笔者就来试着戳戳看，戳错了还请大家指正：)

在 MK 的电调程序中，关于启动算法的所有内容都在 `char Anwerfen(uchar pwm)` 这个函数中（`main.c` 文件的第 162 行），下面我们将分两步讲，前半部分关注于启动算法程序的流程，后半部分探究其中的物理原理。

1. 函数 Anwerfen 启动流程分析

Anwerfen 在德语中就是“启动”的意思，入参为启动时打算施加的电压 PWM 占空比，取值范围为 0~255，按比例折合，0 表示占空比 0%，255 表示占空比 100%。在主程序中，使用的是 `Anwerfen(10)` 来调用本函数的。

进入函数后，首先在 166 行关掉模拟比较器中断，接着先给一个较小的 PWM 值（`PWM=5`）来试运行看看。第 168 行的 `SetPWM` 函数的功能是将这个全局变量“PWM”设到 `OCR1A`, `OCR1B` 和 `OCR2` 中去。接着在第 169 行调用 `Manuell` 函数，`Manuell` 函数不难，我们可以进到 `Manuell` 函数里看看。`Manuell` 函数的定义在 `BLMC.c` 文件的第 192 行。进入 `Manuell` 后，当前 `Phase` 的值是 0。故跳到第 198 行，先对 A 相上臂的 MOS 管输出 PWM 信号，再打开 B 相的下臂场效应管。接着在第 200、201 行启动 C 相的上升沿比较器，不过前

面模拟比较器中断已经关了，故这两句话在这里没起啥作用。可见，**Manuell** 函数的功能就是根据全局变量 **Phase** 的值对电机通以不同的相位，前面说过，0~5 分别代表 AB, AC, BC, BA, CA, CB 相通电。

然后回到 **Anwerfen** 函数，接着是启动前的一个自检功能。第 171~181 句的作用是延时 300ms，在期间不断地去测量电流，如果发现电流大于 12A，则关掉所有 MOSFET，红灯闪 10 下（**RotBlink(10)** 函数），然后退出 **Anwerfen** 启动函数，返回值为 0，说明启动失败。其中延时 300ms 的功能是通过 **SetDelay** 和 **CheckDelay** 函数来完成的，其延时原理后面会具体分析，这里只要知道是在延时过扯中不断循环运行 **while** 中的内容就可以了。

如果电流没问题，则检测通过，在 182 行将 PWM 值设成入参的 10，接着就进入下面的 **while(1)** 启动算法主循环。

首先在 185 行看到有一个 **for** 循环，这里 **timer** 的初始值为 300，故循环 300 次，每个循环里面都有一个 **Wait(100)** 函数，其功能是延时 100us。至于那个上面的那个 **if-else** 语句，这是给串口输出一些调试信息的，跟我们这里的启动算法无关，故暂时略去不考虑。不过还是忍不住多说一句，呵呵，全局变量 **UebertragungAbgeschlossen** 是用来表示串口当前是否正忙，0 表示不忙，1 表示忙。这 300 次 **for** 循环的 100us，加起来就是 30ms，跟这段时间比起来，下面的一些语句的运行时间都可以忽略不计了。

延时完后，在 191 行的 **DebugAusgaben** 函数的功能是将一些当前信息写到调试结构中去，这里也暂时不考虑。下面 192~198 行再次测试电流（此时虽然 PWM 全局变量等于 10，但 **OCxx** 寄存器内部的 PWM 还是 5），如果大于 6A 则关断所有 MOSFET，红灯闪 10 下，退出启动函数，返回值为 0，说明启动失败。

若电流正常，则跳到 200 行，这句话的实质是 **timer** 值减小为原来的 14/15，加 1 是为了抵消 C 语言除法时截断小数行为，而使 **timer** 的收敛更快。之后在这个 **while(1)** 的主循环内，每循环一次，**timer** 值都会减小为原来的 14/15，直到 **timer** 减小到小于 25，如果真的循环若干次后 **timer** 减小到 25 以下了，在 201 行的 **if** 判断会退出启动函数，返回值为 1，说明启动成功。这里 **TEST_MANUELL** 是一个调试时用的宏，正常时定义为 0，故不考虑。之后在 202 行重复调用一次 **Manuell**，还是 AB 相通电不变，在 203~204 行使 **Phase** 的值每次自加 1，并在 0~5 之间轮回。205 行再测一次电流。206 行再设一次 PWM（这句好像有点多余）。直到 207 行才把 10 这个值真正设到 **Ocxx** 寄存器中）。

之后在 208 行用 **SENSE** 宏看看 C 相电压有没有变负，虽然这里模拟比较器中断被禁掉

了，但看看状态总还是可以的吧，如果按正常情况变负了，绿灯就亮，给你看看电机已正常输出反电动势了。

之后回到上面的 `while(1)` 主小循环，开始第二次循环，此时 `timer` 的值已变为 $300-(300/15+1)=279$ 了，故 `for` 循环 279 次，总共延时 $279*100\mu s=27.9ms$ 。后面的流程也是一样，到 202 行会换成 AC 相通电，之后的流程都是类似的，每循环一次换一次相，直到 `timer` 值减小到 25 以下。

纵观整个启动算法，其核心就是先让 AB 相通电足够长的时间，以使转子在这个位置固定下来。然后换 AC 相、BC 相……，每步通电时间都为上步时间的 $14/15$ ，换句话说，就是随着转子速度的加快，每步的通电时间越来越少，换相若干次后（实测为 32 次），则视为启动成功，退出启动函数。如果一定要写成公式的话，启动算法中每步的通电时间可以写成：

$t_n = A^{n-1}t_1$ ，其中 $A = \frac{14}{15}$ ， $t_1 = 30ms$ 。这是个指数函数，下面我们将看看这个计算式是怎么来的。

2. 启动算法机理探究

阅读本小节你可能需要一点高数和大学物理的知识，如果看到数学比较头疼的朋友可跳过本小节不看，不影响后面章节的理解；想深究的朋友欢迎探讨交流。

在高中物理里我们学过，对于直线运动物体： $F = ma$ ，如果写成微分形式就是 $F = m\dot{v} = m\ddot{s}$ 。同样，对于定轴旋转物体，也有一个类似的表达式： $M = J\dot{\omega} = J\ddot{\beta}$ ，其中 M 是外力矩， J 是物体的惯量， ω 是转动角速度， β 是转动的角度。这就是“定轴转动刚体的角动量定理”。

下面为便于理解，将转子的圆周运动展平成一条直线，在图 3-6 中可以看到，每一步通电期间，转过的角度是相等的， $\beta_1 = \beta_2 = \dots = \beta_6$ ，但由于转子是处于加速运动状态，转过每一个 β_n 的时间是逐渐减小的。如果转子是处于匀加速运动状态的，那倒好办了，高中物理那点知识就可轻易把它拿下，关键就在于，转子是处于变加速状态的，难就难在这里。

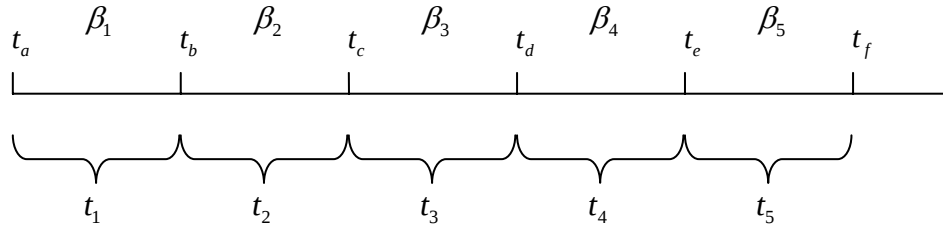


图 3-6 转子启动过程示意图

下面我们分析转子在启动时的受力情况。根据最简单的左手定则： $F = nBLI$ ，其中 n 为线圈匝数。而转动物体需要的是力矩，故 $M = Fr = nBLIr$ ，其中 r 为转子半径。

在上式中 n, B, L, r 都是定值，只有 I 是变化值，那么 I 等于多少呢？有人说， $I = \frac{U}{R}$ ，错啦，真要这么想的话，前面那么多电动势/反电动势都白看啦。真正的 $I = \frac{U - E}{R}$ ，其中 U 是外加电压， E 是线圈切割磁力线产生的反电动势。那么 E 又等于多少？根据右手定则， $E = nBLV$ ，由于这里是圆周运动，故 V 要化成角速度，则 $E = nBL\omega r$ ，将这个 E 代入上面的 I 再代入上面的 M 得：

$$M = \frac{nBLrU}{R} - \frac{(nBLr)^2}{R} \omega \quad (3.1)$$

由于上式中 n, B, L, r, U 都是常量，我们可设 $k_1 = \frac{nBLrU}{R}$ ， $k_2 = \frac{(nBLr)^2}{R}$ ，这样上式就可简写成：

$$M = k_1 - k_2 \omega \quad (3.2)$$

怎么样，是不是看上去清爽很多？后面我们还会多次用到这样的简写法。根据上面给出的角动量定理： $M = J\dot{\omega} = J\ddot{\beta}$ ，式(3.2)可写成： $J\dot{\omega} = k_1 - k_2 \omega$ ，再整理一下成：

$$\dot{\omega} + \frac{k_2}{J} \omega = \frac{k_1}{J} \quad (3.3)$$

这是一个典型的一阶线性非齐次微分方程，再用我们刚才简写方法，设 $k_4 = \frac{k_2}{J}$ ， $k_3 = \frac{k_1}{J}$ ，上式又可简写成：

$$\dot{\omega} + k_4 \omega = k_3 \quad (3.4)$$

接着我们解(3.4)，由于这个方程比较典型，它的解还是比较容易的，先求对应的齐次方程的通解，再用常数变易法求非齐次方程的通解，最后解得：

$$\omega = Ce^{-\int k_4 dt} + e^{-\int k_4 dt} \int k_3 e^{\int k_4 dt} dt = Ce^{-k_4 t} + \frac{k_3}{k_4} \quad (3.5)$$

最后用初始条件求常数 C ，当 $t=0$ 时 $\omega=0$ ，代入式(3.5)可解得， $C = -\frac{k_3}{k_4}$ ，再次用

简写方法，设 $k_5 = \frac{k_3}{k_4}$ ，则式(3.5)可简写为：

$$\omega = k_5 - k_5 e^{-k_4 t} \quad (3.6)$$

好了，现在 ω 的精确表达式已经得到了，可以求图 3-6 中的 $t_1, t_2, t_3 \dots$ 的关系了，

利用角度等于角速度的积分这一关系，可写出下列各式：

$$\beta_1 = \int_0^{t_1} (k_5 - k_5 e^{-k_4 t}) dt \quad (3.7a)$$

$$\beta_2 = \int_{t_1}^{t_2} (k_5 - k_5 e^{-k_4 t}) dt \quad (3.7b)$$

$$\beta_3 = \int_{t_2}^{t_3} (k_5 - k_5 e^{-k_4 t}) dt \quad (3.7c)$$

.....

再利用 $\beta_1 = \beta_2$ 的关系，可得到 t_2 和 t_1 的关系式；再利用 $\beta_2 = \beta_3$ 的关系，可得到 t_2 和 t_3

的关系式。下面我们来解一下：

$$\beta_1 = \int_0^{t_1} (k_5 - k_5 e^{-k_4 t}) dt = k_5 t \Big|_0^{t_1} - \left(-\frac{k_5}{k_4} e^{-k_4 t} \Big|_0^{t_1} \right) = k_5 t_1 + \frac{k_5}{k_4} e^{-k_4 t_1} - \frac{k_5}{k_4} \quad (3.8a)$$

$$\beta_2 = \int_{t_1}^{t_2} (k_5 - k_5 e^{-k_4 t}) dt = \dots = k_5 t_2 - k_5 t_1 + \frac{k_5}{k_4} e^{-k_4 t_2} - \frac{k_5}{k_4} e^{-k_4 t_1} \quad (3.8b)$$

$$\beta_3 = \int_{t_2}^{t_3} (k_5 - k_5 e^{-k_4 t}) dt = \dots = k_5 t_3 - k_5 t_2 + \frac{k_5}{k_4} e^{-k_4 t_3} - \frac{k_5}{k_4} e^{-k_4 t_2} \quad (3.8c)$$

将 β_1 和 β_2 的右边等起来得到：

$$k_5 t_1 + \frac{k_5}{k_4} e^{-k_4 t_1} - \frac{k_5}{k_4} = k_5 t_2 - k_5 t_1 + \frac{k_5}{k_4} e^{-k_4 t_2} - \frac{k_5}{k_4} e^{-k_4 t_1} \quad (3.9)$$

式(3.9)为一个超越方程，手算是解不出来滴，那怎么办呢？所幸我们生在一个伟大的时代，有了计算机和 MATLAB 之类的工具，以前算不出来的问题，都可迎刃而解。用 MATLAB 去计算数值解，进而得到 $t_1, t_2, \dots, t_n$ 之间的关系，然后比较其曲线形状和上面 MK 用的指数

函数的形状是否相似。这个笔者还在进一步研究中，为了文档能早一点放上来，这里先放一放，待以后补全，呵呵，各位不要扔砖头啊。

3.5 上电时的 MOSFET 自检

一般的电路板在上电后都要自检，这里电调板自检的主要内容就是 MOSFET 及其驱动电路的好坏，关于自检的代码都在 `mian.c` 文件第 265 行的 `void MotorTon(void)` 函数中。在这个函数中，会用到两个仅供本函数使用的私用延时函数：`uchar Delay(uint timer)` 和 `uchar DelayM(uint timer)`，跟定时器 T0 提供的公用延时函数无关，下面我们先来看一下这两个私用延时函数：

1. 函数 Delay 和 DelayM

`Delay` 函数在 `main.c` 文件的第 244 行，关于它究竟会延时多久，没人知道，取决于不同的编译器和编译选项（背景音乐：Nobody Nobody but you …… 汗_-#）。笔者在 AVRGCC 编译环境下查看了这个函数的反汇编代码，得到如下结果：在 `-O2` 编译条件下运行时间为 $(5 * \text{入参} + 3) / 8$ 微秒，在 `-O0` 编译条件下运行时间为 $(10 * \text{入参} + 22) / 8$ 微秒。由此可见，这个 `Delay` 函数仅仅用于不需要精确的随便延时一下的场合。

`DelayM` 函数的机理基本同 `Delay` 函数，只不过在每次 `while` 的循环中调用一下 `FastADConvert()` 测一下电流，看看是否会超过 $(\text{TEST_STROMGRENZE} + \text{RuheStrom})$ ，若超过则所有场效应管全关，返回 1。其中 `TEST_STROMGRENZE` 宏的定义为 13A，`RuheStrom` 是一个全局变量，意义是当所有场效应管全关时通过各管的漏电流总和。

2. 函数 MotorTon 自检流程分析

`MotorTon` 函数大致可分为五个部分：①第 273~333 行做一些准备工作；②第 334~346 行测试六个 MOSFET 的短路特性；③第 383~426 行测试 A+, B+, C+ 管的导通特性；④第 431~463 行测试 A-, B-, C- 管的导通特性；⑤第 467~491 行做一些收尾工作。

(1) 第 273~333 行：一些准备工作

在第 273 行先关掉红灯，然后在第 274 行延时一段时间，由于四轴的四个电调地址并不相同，故这句延时时间也会不一样，后面发出的自检音效也会各自错开。第 275 行关模拟比较器中断，第 276 行关总中断，第 277 行给串口发一个换行字符。第 278~281 行的用意是这样的：宏 STEURE_OFF 先关断所有场效应管，然后调用 DelayM(50)测 50 次电流，其中测到的最大一个电流会被放入全局变量 Strom_max，然后让 RuheStrom = Strom_max，这个 RuheStrom 就是测得的总漏电流。这时候若接了上位机调试软件，上位机会在刚才收到单片机串口发上来的换行符后，回发给单片机一个 ‘_’ 字符，这里我们为方便分析，假设电调此时并未接上位机，故 332 行 t=1000，然后 333 行将全局变量 Strom 清零。

(2) 第 334~346 行：测试六个 MOSFET 的短路特性

第 334~346 行有两个用意：一是看看是否有管子已被击穿，二是看看场效应管的开关特性是否足够陡峭。现分析如下：第 336,337 行打开 A 相的下单臂（即 NA-管），并测一下电流。此时，电流应该是极小，但是如果测到大电流，则说明 NA+管已是击穿状态。然后第 338~340 行先关断所有管，然后延时一下，再打开 A 相的上单臂，在第 341 行测电流时也应该极小，如果测到大电流，说明可能有两种情况：1. NA-场效应管已是击穿状态，2. NA-场效应管的下降沿过于缓慢，以至于 NA+都打开了，NA-还没彻底关断。如果测到大于 0.5A 的大电流，不管哪种情况，都会在 343 行将局部变量 anz 设为 4，表明故障点，并通过串口发给上位机。

之后 365~363 行测试 B 相的两个 MOSFET，365~376 行测试 C 相的两个管，原理亦相同，如果有管子损坏亦会设 anz 为相应的值，以记录故障点。第 378 行若 anz 不为零，说明有管子损坏，进入死循环，红灯不停地闪烁 anz 下。

(3) 第 383~426 行：测试 A+, B+, C+管的导通特性

第 383~426 行的主要用意是：分别测试 A+, B+, C+三个上臂管是否能正常导通。第 383 行设置 ADMUX 为 ADC 通道 0，IntRef 变量的作用上面已经讲过了。384~387 行打开 B-和 C-管，第 388~392 行为一些宏定义，在下一小节我们将知道，它们的作用是确定电机发出音调的频率，这里先把它当作普通宏来看好了。第 394~401 行，循环 $40000/330=121$ 次，每次先

在 396 行打开 A+管，现在的电流方向应为从 VCC 流过 A 线圈，在中点一分为二，一半经 B 线圈流向地，另一边经 C 线圈流向地。第 397 行延时一段时间，第 398 行测 NULL_A 的电压，从图 2-19 可以算得：此时 PHASE_A 点电压约为 12V，NULL_A 点电压约为 5V。故局部变量 MosfetOkay 的第 0 位会置 1，表示 A+管良好。但如果 A+管损坏而不能导通，PHASE_A 的值为 0，自然 NULL_A 的值也为 0，则第 398 行会把 MosfetOkay 的第 0 位置零，以表示 A+损坏。之后在 399~400 行关掉 A+管，再延时一会儿。然后又回到 394 行重新循环。其实要测量 A+管是否导通只要循环一次就够了，为什么要循环 121 次呢，下一小节我们会看到，这是为了使电机发出音调。121 次循环完成后，在 402 行关掉所有场效应管。

下面 B+管的检测和 C+管的检测也与 A+管相同，如果某个管有问题会置位 MosfetOkay 变量的相应位。

(4) 第 431~463 行测试 A-, B-, C-管的导通特性

第 431~463 行的主要用意是：分别测试 A-, B-, C-三个下臂管是否能正常导通。测量方法也是类似，不过这次要被测电压小于某个值才算正常。若某个管正常，同样会置位 MosfetOkay 的相应位；若不正常，则清零相应位。

(5) 第 467~491 行：一些收尾工作

完成以上的 MOSFET 自检工作后，在 467 行开中断，480 行再延时一下下，481~486 行根据 MosfetOkay 的值判断哪个管损坏，若有损坏，则置 anz 为相应的值，并通过串口将相应损坏管代码发出去。第 489 行若 anz 不为零，则进入死循环，红灯不停地闪 anz 下。第 490 行再延时 1 秒钟，491 行向串口发送 ‘.’ 字符，说明自检完成。

3.6 让你的电机演奏音乐

关于商业电调上电自检时发音机理，在四轴论坛讨论的不多，在 5iMX 上倒是讨论过很多次了。一种理论认为，是由于线圈上通电后，定子铁芯中的硅钢片受到磁场的作用力，会向一个方向运动；当电断掉后，硅钢片又会回到原来的位置，如此以一定的频率通电断电、硅钢片就会往复振动，当振动频率在 20Hz~20000Hz 内时，就会发出人耳能听到的音频了。还有另一种理论认为，发出声音的是振动的转子：我们在用手去转动无刷电机转子的时候，

是不是有明显的一格一格的感觉？从第一章的分析知道，磁路总是倾向与走磁阻最小的通路，所以转子在平时静止时的位置，肯定是磁阻最小的位置，如果你用手去拨动转子，等于打破了原来的平衡，肯定要受到阻力，转子总是倾向于回到它原来磁阻最小时的位置。因此，如果对线圈通电，转子受力会向一个方向运动；若通电时间很短，当断电时，转子又会回到它原来的平衡位置，如此也能造成振动效果而发出声音。

那这个声音究竟是定子发出的，还是转子发出的，很简单，做个实验就知道了：把转子拆掉，再运行程序，看看还有没有声音。这个……因为笔者已经拆掉一个电机了，不能再烧钱了，所以这个实验还是请阁下自己做一做吧，做完了别忘了把结果发到论坛上来噢。

不管是定子发声还是转子发声，对程序来讲都是一样的，现在来看能发声的程序，在 main.c 的第 388~402 行。其中 388~392 行的这些宏定义了发声的频率，在 394 行进入 for 循环。当 396 行打开了 A+管后，硅钢片就开始往一个方向运动了，之后的 Delay(1)和 MessAD(0)要花多少时间呢，只能用反汇编看，笔者在-O2 编译条件下粗看了一下，大约为 20us 左右。接着关掉 A+后，Delay(300)根据前面的计算公式，延时约为 180us 多一点。因此这一开一关的周期约为 20+180=200us，也就是说能发出 $(1/200\text{us})=5000\text{Hz}$ 频率的声音。那这个声音持续多久呢，看循环次数。这里的 for 循环 121 次，即声音应该持续 $200\text{us} \times 121 = 25\text{ms}$ 左右。最后 402 行关掉所有 MOSFET，测试 A+管的发声过程结束。后面的测试 B 管 C 管的原理也类似。

知道了发声原理后，要编一个音乐程序就是易如反掌的事情啦，只要控制好第 400 行 Delay 的入参就可以了。在笔者自编的可供 MEGA32 和 MEGA8 使用的电调程序中，自检 MOSFET 从 A+管到 C-管分别会发出 do re mi fa so la 的音调，程序下载地址在本帖中给出。图 3-7 给出了一个各音调的频率表。

音符	C	D	E	F
名称	1	2	3	4
	dou	rai	mi	fa
频率 (Hz)	256	288	320	$341\frac{1}{3}$

音符	G	A	B	Ci
名称	5	6	7	I
	sou	la	si	dou
频率 (Hz)	384	$426\frac{2}{3}$	480	512

图 3-7 各阶音符频率表

3.7 通信模块

MK 电调与外界的信息交流主要通过三个途径：①PPM 信号；②TWI 总线；③串口通信，可以用它们中的任何一个来控制电调的 PWM 信号的占空比，它们的代码分别位于 PPM_Deccode.c、twislave.c 和 uart.c 三个文件中。笔者个人认为 MK 电调关于通信部分的代码写得比较 ugly，如果说电机驱动部分不得不参考 MK 的代码的话，通信部分各位就可以大显身手啦，我相信坛内在座的各位高手达人一定有能力写出比 MK 更为优雅和简洁的代码，所以关于这部分模块会分析得比较简略，以叙述原理为主。

1. PPM 解码

先讲一下 PPM 的原理，这是遥控模型中比较通用的一种信号格式，其原理是通过检测给定频率的 PWM 信号的占空比来获取指令信号。一般遥控模型中通用的伺服舵机、电调等都可以接收 PPM 的编码信号。

其原理见图 3-8，信号频率为 50Hz，一个周期为 20ms。对于电调来讲，脉宽为 1ms 表示停转，脉宽为 2ms 表示满油门运转，其间的各点按比例换算：比如脉宽为 1.5ms 就表示 50%油门等等。对于伺服舵机来讲，脉宽为 1ms 表示转到左极限位置，脉宽为 2ms 表示转到右极限位置，中间的各点也同样按比例折算。

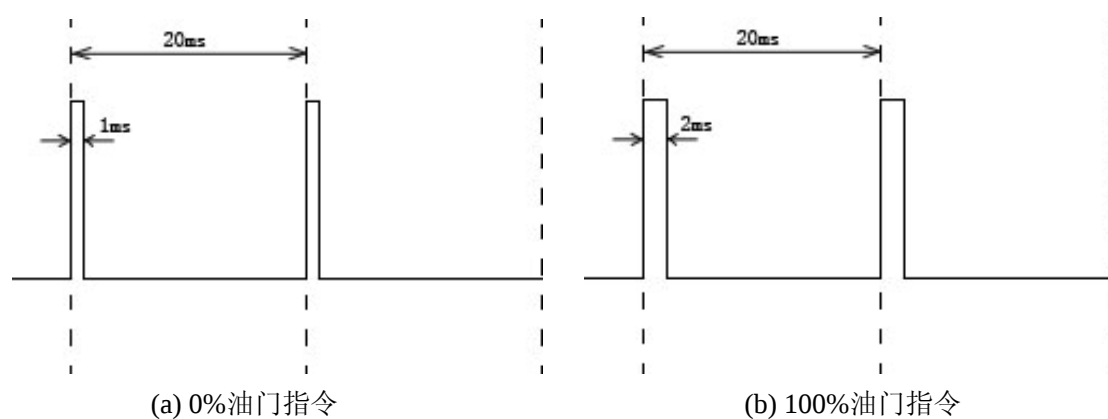


图 3-8 PPM 信号图解

在程序中，主要是利用定时器 T1 的输入捕捉功能来实现 PPM 的解码功能，程序代码主要在 PPM_Deccode.c 文件中的 `SIGNAL(SIG_INPUT_CAPTURE1)` 中断服务程序中，我们来粗略的看一下：

在主程序中，首先会调用 `InitPPM` 函数来初始化 T1 的输入捕捉模块，其功能是设置捕

捉模式为上升沿触发，并使能输入捕捉中断。当 ICP 引脚上有上升沿来到时，会进入第 37 行的中断服务程序。if 先判断，如果此时是上升沿触发，则先把全局变量 Timer1Overflow 清零，记录一下当前的 ICR1，然后改为下降沿触发，最后将全局变量 PPM_Timeout 设为 100（这个变量的作用我们后面再讲），退出中断服务程序。当之后的下降沿来到后，再次进入中断服务程序后会进入 else 分支。然后再把触发方式改回上升沿触发，之后计算这个脉冲宽度。下面来说说这个脉冲宽度是怎么算出来的。

我们知道，当输入捕获事件发声时，MEGA8 会将 16 位的 T1 计数器内的值自动放到 ICR1 寄存器中。那么根据一上一下两次中断时记录的不同的 ICP1 的差值，就可算出其间的时间差。如果期间 T1 溢出了若干次，则每次溢出，Timer1Overflow 全局变量都会自加 1（见 SIGNAL(SIG_OVERFLOW1) 中断服务程序），这样就不会遗漏了。由于 T1 工作于“8 位相位修正 PWM 模式”，其计数方式为从 0 自加 1 计到 0xFF 再从 0xFF 自减 1 慢慢记回到 0，故其每次的溢出时累计会计数 $256 \times 2 = 512$ 次。明白了这点，第 56 行：

```
ppm = (ICR1 - tim_alt + (int) Timer1Overflow * 512) / 32;
```

就很好理解了。那么为什么要除以 32 呢，由于 T1 的计数频率为 8M，如果不除以 32，算出的 ppm 值会相当大（假设 2ms 的间隔时间，ppm 大约会等于 $2m \times 8M = 16k$ ，没必要要这么高的分辨率）。故上式中除以 32 后，ppm 变量内数值的单位为 4us。下面第 58 行可以看到，ppm 的下限是 $280 \times 4us = 1.12ms$ ，之后在第 59 行再减去这个基数，变为一个纯信号。我们大致可以算一下，如果 ICP 捕获到的是 2ms 的间隔，那第 58 行算出的 ppm 大约应该是 500，再在 59 行减去 280 后为 220。

之后的第 60~63 行也是个数字滤波，由于 PPM 为周期性信号，新的脉冲会源源不断地被收到，ppm 会突变，但 PPM_Signal 只会平缓地上下变动，最后在第 62 行再来一次权重分配，平缓稳重的 PPM_Signal 会占 7/8 权重、每次新来而轻佻的 ppm 只占 1/8 权重，综合后的值再放入 PPM_Signal 全局变量。

第 64 行的 anz_ppm_werte 全局变量也很有意思，第一个单词为英语 analyze 的缩写，中间是 ppm，最后的 werte 在德语中是“价、值”的意思（感觉有点像英文的 worth）。而纵观整个程序代码可以发现，以严谨著称的德国人一般变量名都会用全称，很少用缩写。而这个变量既用到了简写，又是英语德语混用，足以证明这是个比较妖的变量。至于它的作用，光看这里是看不清楚的，要讲到后面的 SollwertErmittlung 函数时才能明白它的作用。这里只要理解为，每次有下降沿来到时，不断把这个变量往上加就可以了，封顶值为 255。而且这

个变量还和刚才没有展看讲解的第 47 行的 PPM_Timeout 有点关系，每次有新的上升沿来到时，都会把 PPM_Timeout 的值给推到 100。但是在 timer0.c 第 29 行的 T0 溢出中断服务程序中，定时器每过 1ms 后，都会把 PPM_Timeout 减 1，当减到 0 时，anz_ppm_werte 变量就会被清零。可见，其综合效果是，如果已经有 100ms 没有收到新的上跳沿了，那 anz_ppm_werte 的值就会被置 0。

最后的 ZeitZumAdWadeln 全局变量我们在换相一小节提到过，将它置 1 的作用是通知换相中断服务程序程序，该检测电流啦。

2. TWI 总线通信

这里的 I2C 总线通讯程序写得还是比较简单的，通讯规约也简单，兄弟们就自己分析一下啦。我这里就简单给个脉络吧：本 MEGA8 的 TWI 工作于从机模式，当主机发送数据时，MEGA8 会将新收到的有效数据放入全局变量 I2C_RXBuffer 中，并且将 I2C_Timeout 推到 500。这个数据的意义就是新的 PWM 信号值。T0 定时器每过 1ms 都会将 I2C_Timeout 减 1，若减到 0，则说明已经有 500ms 没有收到主机新的 TWI 数据了。如果主机要求数据，MEGA8 会将 Mittlestrom（平均电流）和 MaxPWM（最大 PWM 值）发上去。

这里要注意一下的是第 17 行，虽然入参 adr 的值为 0x50，但由于 TWAR 寄存器特殊的格式，实际的 TWI 从机地址其实是：0x28+MotorAdresse，这个笔者曾经中过招，大家小心。

3. 串口通信

关于串口通信的代码都在 uart.c 文件中，虽然林林总总一大堆函数，但如果你对单片机编程还算比较熟的话，自己分析一下应该是没有问题的。况且做四轴主要是用 I2C 控制电调，串口仅仅用作调试用。如果你想要更彻底了解和折腾电调，最好这部分串口代码和上位机对应的调试软件都自己写，然后去调试，这样你对电调机理的理解会得到极大的提高。不过如果实在回帖说看不懂的人太多，笔者还是会考虑将这部分的内容解释一下以补全这份文档的，不过已是下下策啦。

这部分代码的关键有两个，一是所有收到的数据都会放入全局变量 RxdBuffer 数组中，每次的命令报文都会以字符 ‘#’ 开头，和 ‘\r’ 结尾。每当成功收到一条报文后，都会将
~~~~~  
By: timegate 墨鸢 技术交流 QQ:1181733110 email:moyuan2000@163.com 第 74 页

SIO\_Timeout 的值推到 500。同理，T0 定时器每过 1ms 都会将 SIO\_Timeout 减 1，若减到 0，则说明已经有 500ms 没有收到新的串口数据了。另一个是当串口收到一个关于调节 PWM 占空比的指令时，这个指令会被换算成 0~255 的值，并放入全局变量 SIO\_Sollwert 中。

## 4. 指令的收入函数 SollwertErmittlung

这个函数在 main.c 文件的第 496 行，Sollwert 在德语中是“标称值”的意思，Ermittlung 是“查明”的意思。合在一起，这个函数的功能就是查明外部到底给电调的是什么 PWM 值，查明这个值后，将其作为返回值返回。这个值有可能是 I2C 给的，也有可能是串口给的，也有可能是 PPM 给的；查看的优先级为 I2C > 串口 > PPM 信号。

我们来看程序，第 501 行首先判断 I2C 有没有收到指令，前面说过，当 I2C 收到数据时，会将 I2C\_Timeout 的值给推到 500 的，假设现在电调工作于 I2C 通信模式下，这里会直接跳转到 530 行的 else 中，第 532 行将 I2C\_RXBuffer 中的值取出放入本地变量 sollwert，再将全局变量 PPM\_Betrieb 的值置 0（PPM\_Betrieb 在主程序的其他地方都没有用到，属于废弃不用的变量），将红灯关掉，最后在 535 行禁用输入捕捉中断，也就是说，如果 I2C 有数据进来，电调将不再接收 PPM 信号。537 行限定一下 sollwert 的最大值，538 行将这个值返回。

我们再进一步，看看若四轴的主控板损坏或死机，已经有超过 500ms 没有给电调发 I2C 数据了会怎样。这时 I2C\_Timeout 为 0，而且 SIO\_Timeout 也为零，由于前面关掉了 PPM 功能，再下面的 anz\_ppm\_werte 也为零，所以程序会直接跳到 526 行运行。这里 TEST\_SCHUB 是个宏，只有在测试电机推力的时候才会定义成 1，一般都定义为 0，所以最终会运行那句 sollwert--;的语句，注意这里 sollwert 是静态变量，当本函数被调用若干次后，sollwert 的值就会慢慢变为 0，并且返回 0。

下面来看如果电调未接 I2C 总线而只接了串口信号会怎样，第 503 行开始进入串口数据处理流程，先在 505 行根据 SIO\_Sollwert 值算出一个对应的 PWM 值，然后赋到 sollwert 中，后面 506~508 行与前面类似，禁用了 PPM 模式，最后 537~538 返回这个 sollwert 值。于 I2C 类似，如果若干时间没有收到串口信号了，sollwert 值也会慢慢变 0。

最后来看，如果电调未接 I2C 与串口而只接了 PPM 信号，会怎么处理。进入函数后，由于 I2C\_Timeout 和 SIO\_Timeout 都是 0，会立刻跳到 511 行，由于 PPM 脉冲是源源不断地过来的，anz\_ppm\_werte 这个值自然肯定会大于 20，然后会运行 if 中的语句。前面说过，

PPM\_Betrieb 是废变量，没用。然后 514~517 规范一下取出的 PPM\_Signal 值，使它不要太大也不要太小，最后在 520 行将当前 ppm 的值按区间[10, 190]的范围，按比例折算成区间[3, 255]中的对应位置，算出的值就是当前要设定的 PWM 值，最后在 522 行将红灯关掉，并在 537~538 将这个值规范一下返回。

哦，最后差点忘了提醒一下，MEGA8 如果用片内 RC 作为时钟的话，需要在程序的一开始校正一下频率(设置 OSCCAL 寄存器)，以消除由于生产工艺所带来的振荡器频率偏差。否则串口通讯若报文长了可能出问题，MK 的电调在 Bootloader 中把这件事做掉了，所以在主程序中看不到相关代码。

## 4. 德国 MicroKopter 项目 BL-Ctrl 电调程序 主程序代码流程分析（V0.41 版本）

看完了第三章各功能模块的软件设计，本章再来总体把握一下程序的流程。不过貌似只剩下一个 `main(void)` 主函数了，其他该讲的都讲了，那这里先扯点别的，再进入正题。

工欲善其事，必先利其器，先讲一下读源代码要用哪些工具。虽然 `AvrStudio` 是一款优秀的 IDE 软件，但并不是个很好的阅读代码工具，如果要阅读包含很多个文件的大型代码群，一个比较好的工具软件是 `Source Insight`，其强大的交叉索引功能和变色功能无疑是阅读源代码的一柄利器。想当年笔者是用它来阅读 `Linux` 和 `U-boot` 源代码的，现在用它来分析这个小小的电调程序，有点杀鸡用牛刀的感觉，呵呵。第二个要用好的是 `Excel` 工具，有人说没搞错吧，这不是 `Office` 里面的电子表格么？没错，你在阅读源代码的时候，必然会遇到大量的宏，有的时候还宏套宏。即便有 `Source Insight` 这么强大的回溯工具，只要一碰到宏就返回去翻宏的定义，必然会在无穷尽的前翻后翻中耗光你阅读的激情。这时，制作一张赏心悦目的宏功能列表，你会觉得是多么受用，而 `Excel` 无疑是制表的最佳工具。第三个要介绍的工具是 `Notepad++`，虽然在这里没用到，不过提一下还是有好处的。这也是一款开源软件，其最大的作用是能将 `Makefile` 文件中的关键字变色，从此 `Makefile` 不再是枯燥的黑白脚本啦，而变成了跃动的音符。

如果你想提高自己的源代码阅读能力，又苦于没有好的教材的话，笔者推荐你可以阅读一下毛德操老先生著的《`Linux` 源代码情景分析》，虽说是厚厚的两本砖头，但如果你能耐得寂寞把它啃完的话（或者只啃掉一半也行），你的源代码阅读能力必然会获得本质的提高，这时候你看代码的角度都不一样了，你自己都不好意思再说自己是菜鸟了：)

还有，如果你打包下载德国 `MK` 的电调程序的话，你会发现除了 `.c` 和 `.h` 文件外还有一堆乱七八糟的文件，这是因为德国人用了 `svn` 版本控制工具，该工具软件的详细用法 `pitolan` 版主曾经发了个教学贴，链接在附录四中给出。

好了，废话就说到这里，下面看程序。

## 5.1 全局变量列表

这里将所有全局变量做成一个表格，并注明功能，可以方便大家在阅读后面的程序中随时查看用到的全局变量，很多全局变量在第三章中已经出现过，这里还会用到。

表 5-1 main.c 文件全局变量列表

| 变量名                  | 参考翻译     | 用途               |
|----------------------|----------|------------------|
| PWM                  | PWM      | 当前的 PWM 值（0~255） |
| Strom                | 电流       | 当前电流值            |
| RuheStrom            | 静止电流     | 场效应管全关时的漏电流      |
| Strom_max            | 最大电流     | 最大电流             |
| Mittlestrom          | 平均电流     | 平均电流             |
| Drehzahl             | 转速       | 转速               |
| KommutierDelay       | 换向器延时    | 未使用              |
| I2C_Timeout          | I2C 到点时间 | 指示 I2C 是否最近收到数据  |
| SIO_Timeout          | 串口到点时间   | 指示串口是否最近收到数据     |
| SollDrehzahl         | 定额转速     | 未使用              |
| IstDrehzahl          | 实际转速     | 未使用              |
| DrehZahlTabelle[256] | 转速数组表    | 未使用              |
| ZeitFuerBerechnungen | 估计时间     | 指示一些操作的优先级用      |
| MotorAnwerfen        | 马达启动     | 指示马达是否处于启动状态     |
| MotorGestoppt        | 马达停止     | 指示马达是否处于停止状态     |
| MaxPWM               | 最大 PWM   | 最大 PWM 值         |
| CntKommutierungen    | 换相计数     | 累计的换相次数          |
| SIO_Drehzahl         | 串口转速     | 向串口输出的当前转速       |
| ZeitZumAdWandeln     | 时间到直到改变  | 指示电流测量标志         |
| MotorAdresse         | 马达地址     | 本电调的地址           |
| PPM_Betrieb          | PPM 运行   | 指示当前是否为 PPM 控制   |
| HwVersion            | 硬件版本     | 指示电路板版本号         |
| IntRef               |          | ADC 参考电压寄存器掩码    |

|             |             |           |
|-------------|-------------|-----------|
| MinUpmPulse | 最小 Rpm 转速脉冲 | 作延时定时器变量用 |
|-------------|-------------|-----------|

表 5-2 BLMC.c 文件全局变量列表

| 变量名                   | 参考翻译      | 用途          |
|-----------------------|-----------|-------------|
| Phase                 | 相位        | 表示当前相位（0~5） |
| ShadowTCCR1A          | TCCR1A 掩码 | 未使用         |
| CompFreigabeTimer     | 比较器使能定时器  | 未使用         |
| CompInterruptFreigabe | 比较器中断使能标志 | 没有实际用途      |

表 5-3 PPM\_Decode.c 文件全局变量

| 变量名            | 参考翻译     | 用途              |
|----------------|----------|-----------------|
| PPM_Signal     | PPM 信号   | 由脉冲宽度换算成的 PPM 值 |
| Timer1Overflow | T1 溢出    | T1 累计溢出次数       |
| PPM_Timeout    | PPM 到点时间 | 指示 PPM 最近是否收到脉冲 |
| anz_ppm_werte  | 分析 ppm 值 | 指示 PPM 最近是否收到脉冲 |

表 5-4 timer.c 文件全局变量

| 变量名               | 参考翻译  | 用途       |
|-------------------|-------|----------|
| CountMilliseconds | 毫秒计数  | 用于当前系统时间 |
| Timer0Overflow    | T0 溢出 | 没有实际用途   |

表 5-5 twislave.c 文件全局变量

| 变量名          | 参考翻译     | 用途             |
|--------------|----------|----------------|
| I2C_RXBuffer | I2C 接收缓冲 | 用于保存收到的 I2C 数据 |
| Byte_Counter | 字节数量     | 发送 I2C 数据的字节计数 |

表 5-6 uart.c 文件全局变量

| 变量名          | 参考翻译  | 用途           |
|--------------|-------|--------------|
| SIO_Sollwert | 串口标称值 | 串口收到的 PWM 指令 |

~~~~~

SioTmp	串口临时值	临时保存串口收到的字节
SendeBuffer[100]	发送缓冲区	串口发送缓冲区
RxdBuffer[100];	接收缓冲区	串口接收缓冲区
NeuerDatensatzEmpfangen	新的数据记录接收	没有实际用途
UebertragungAbgeschlossen	传送完成	指示当前发送是否忙
MeineSlaveAdresse	我的从机地址	没有实际用途
MotorTest[4]	马达测试	用于保存一些中间结果
AnzahlEmpfangsBytes	一些接收字节	表示接收到多少字节
DebugOut	调试输出	调试信息结构体

5.2 main 主函数流程分析

main 函数在 main.c 文件的第 545 行。进入 main 函数后，会先做一些各模块初始化和准备工作，然后进入单片机伟大的 while(1)主循环。

1. 进入 while(1)前的准备工作

第 553~558 行初始化 I/O 端口，这个比较简单，自己看。然后在 560~577 行设置本电调的地址。早期 1.0 版本的电路中，电调程序的地址是不可变的，是在宏 MOTORADRESSE 中定义的。而从 1.1 版电路起，电调地址可在 PCB 板上用跳线动态设定，这段代码也比较简单，一看电路图就可明白是怎么回事了。若是 1.1 版本及以上电路，HwVersion 变量设为 11；若是 1.0 版本电路，HwVersion 则设为 10。

后面比较有意思的是第 578 行，不知道大家在看原理图的时候有没有注意到那个绿色 LED 的电路，1.1 版本是正的，PD7 管脚输出电流；而 1.2 版本是反的，PD7 管脚吸收电流。其实这就给程序判断是 V1.1 电路还是 V1.2 电路提供了依据。程序一开始将 PD7 置于输入状态，然后去测 PD7 的电平，如果是高电平则为 V1.2 电路，将 HwVersion 设为 12，IntRef 设为 0xC0，IntRef 的作用第三章第一节就已讲过，这里不再赘述。当判断完电路版本后，579 行再将 PD7 置为输出模式。

下面的 597~600 行，都是各功能模块的初始化函数，或已讲过，或比较简单，读者可自

行阅读。唯一要注意一下的是第 582 行，已开总中断。

接着 602~607 行会设一堆定时器，马上会用到的是 MinUpmPulse，第 608~611 行的作用是延时 103ms，但延时中的每次循环都会调用 SollwertErmittlung()函数，一旦发现信号输入立刻停止延时，继续往下运行。如果没有信号输入也没关系，103ms 延时过后也会继续往下运行。

后面为叙述方便，我们都假设电调工作在 I2C 控制模式下！

之后 613~623 在做一些杂事：613 行打开绿灯。614~616 行将PWM设为 0 并写入寄存器。618~619 行使能模拟比较器中断，并将负极输入设为通道 1。621 行再次将MinUpmPulse设为未来10ms的时刻。622 行填写一些调试信号。623 行将PPM_Signal设为 0。

第 625 行，如果此时电调还未收到主机的信号（SollwertErmittlung 函数返回 0），就调用 MotorTon 启动 MOSFET 自检程序，并弄点声音给你听听。

由于刚才的 MOSFET 自检，把 PB 口的方向和上拉都弄乱了，所以在 627 行再行设定一遍 PORTB。

第 630 行，若定义了 TEST_MANUELL 宏，则会立刻运行 Anwerfen 函数启动电机，而且会以定义的 TEST_MANUELL 宏的值为大小来启动电机，这个只在作者调试时用，正式程序中 TEST_MANUELL 只定义成 0，故这里的 Anwerfen 不会运行。

2. while(1)主循环内容分析

while(1)主循环里的内容其实也分为两段，第 640~670 行为第一段，第 672~749 行（即 if(!ZeitFuerBerechnungen++)中的内容）为第二段。为什么要这么分呢，这里作者使用了一些优先级的概念。第一段的内容优先级比较高，是每次 while 循环都要运行的，故这段的运行频率比较高。第二段 if 里的内容，优先级比较低，每次当 ZeitFuerBerechnungen 自加 1 直到 255 翻转到 0 时，才会运行第二段 if 里的内容。所以运行频率为：第一段每运行 255 次，第二段才运行一次。当如果在第一段中发生一些紧急事件，需要立刻运行第二段时，在第一段中会把 ZeitFuerBerechnungen 设为 0，这样到第二段时就会直接运行 if 里的内容，而不必等加到 255 翻转了。

由于 while 主循环内的分支比较多，下面我们将假设一些情景，进行逐步抽丝剥茧的分析，分析的方式和以前会有些不一样，注意区别。

➤ 情景一：假设开机后主控板静默，并没有发任何 I2C 信号过来

先看第一段：由于没有任何信号过来，故第 640 行 PWM 收到的值为 0。643 行 Phase 和 altPhase 此时也都是 0，故会跳过这段 if 的内容。再下面到 651 行，由于此时 PWM 的值为 0，故会运行 if 中的内容。在 if 中，先将 MotorAnwerfen 设为 0 和 ZeitFuerBerechnungen 设为 0，故等会儿到 672 行第二段时会先运行一次 if 中的内容。接着看 656 行，由于 MotorGestopptimer 是个未初始化的局部变量，故这里 if 里的内容有可能运行，也有可能不运行。不管运不运行，其实效果是一样的：比如，第 658 行禁用比较器中断，本来比较器中断就是关着的；第 659 行将 MotorGestoppt 设为 1，而 MotorGestoppt 本来就被初始化为 1；第 660 行关所有 MOSFET，本来就都是关着的。之后到 669 行，此时 MotorGestoppt 为 1，故 PWM 还是设成 0，并在 670 行写入 MEGA8 的寄存器。

再来看第二段：由于刚才把 ZeitFuerBerechnungen 设成了 0，故在第一次循环中就会先运行一次第二段中的内容。我们一句句来看：

第 674~678 行比较简单，开绿灯，测电流。第 681 行会看看 MittelstromTimer 定时器时间到没到，前面在 605 行 MittelstromTimer 被设成了 254ms，这里显然不会这么快到，故跳过 if 中内容不运行。第 698 行看看 DrehzahlMessTimer 定时器到没到时间，前面 DrehzahlMessTimer 设的延时更大，为 1005ms，自然也不会这么快运行。

接下来的 709~748 行 if 块中的内容都是和电机启动或关闭有关的，我们来看这句话：
`if((CheckDelay(MinUpmPulse) && SIO_Drehzahl == 0) || MotorAnwerfen)`
要进入这段 if 程序有两个条件，满足任何一个都可。① 如果 MotorAnwerfen 在前面被置 1，说明需要启动电机，会进入本 if 块内运行。② 如果 MinUpmPulse 时间到且 SIO_Drehzahl 为 0，则会进入本段 if 块程序。MinUpmPulse 在第一段的 621 行被设为延时 10ms，等会儿总归会到的，但现在还没到；而 SIO_Drehzahl 我们等会儿可以看到，是当前转速值，现在在停机状态下当然为 0。后面我们会看到，这其实是电机意外停转的判别条件。故这里在第一次 while 主循环中不会进入这块 if 代码。

最后跳到 750 行，返回到前面去继续 while 循环，接下去的若干遍循环会和上述分析的过程一样，不会有任何变化。

➤ 情景二：假设主机继续一直保持静默，不发送任何 I2C 信号过来

循环若干次以后，刚才“情景一”中的那些定时器会一个个到期，所以会运行一些刚才情景一中没运行过的代码。假设时间过去了 10ms，第一个到期的是MinUpmPulse定时器，所以会在某次循环中，进入第 711 行。由于MotorGestoppt本来就是 1，且比较器中断本来就是禁用状态的，故此时第 711, 712 行没起啥作用。第 713 行再次把MinUpmPulse更新为未来 100ms。可见MinUpmPulse这个定时器会不停地被更新、到期、被更新、到期……。后面我们将会看到，这 3 行代码其实是给电机意外停转时用的，而不是给启动用的。真正的是否要启动判别在 714 行，由于这里MotorAnwerfen为 0，故不会进入下面这段启动算法。跳过这段程序后，又会回到前面的while继续循环。

时间继续过，到了 254ms的时候，第 681 行的MittelstromTimer定时器会到期，然后进入if中的语句：在 683 行再将MittelstromTimer定时器更新为未来50ms；再后面 684~685 行使Mittelstrom平缓化；第 686 行如果测得的Strom太大，则将全局变量MaxPWM减小 1/32（下次你再调用SetPWM函数时，新的PWM值是不能大于这个MaxPWM的）。第 687~695 行会根据动态Mittelstrom的情况，向上或向下不断修正MaxPWM的值。由于现在电机还没开始运行，故这个段落目前不起什么作用。

时间继续过，到了 1005ms的时候，第 698 行的DrehzahlMessTimer定时器会到期，然后进入if中的语句，在 700 行再将DrehzahlMessTimer定时器更新为未来10ms；第 701 行使SIO_Drehzahl等于CntKommutierungen的值。我们在第三章的换相一小节曾经提到过CntKommutierungen，其作用是电机每换一次相自加 1，作为累计换相次数用。看看这行后面注释掉的代码很有意思，作者本来是想将累计换相次数CntKommutierungen的值和前后两次进入DrehzahlMessTimer的时间差，换算成转速的，但是出于某种原因，放弃了这一企图，仅仅将SIO_Drehzahl=CntKommutierunge完事。后面第 702 行清零CntKommutierungen，703 行将ZeitZumAdWandeln置 1，提醒换相程序该测电流了。由于现在电机还没开始运行，故这个段落目前不起什么作用。

在接下去的 while 循环中，程序会根据定时器的时间，不断反复进入 MinUpmPulse, MittelstromTimer, DrehzahlMessTimer 三个定时器中的内容运行。

➤ 情景三：假设主机现在通过 I2C 发过来一个字节“50”，通知电调该启动啦

这个 30 是主机期望的 PWM 值，也就是说，希望电调 A+,B+,C+引脚发出占空比为 50/255=19%的占空比来驱动电机。

在while主循环里，第一个感受到这件事情的是 640 行的PWM=SollwertErmittlung()，这句话会把PWM变量的值设为 50。然后到 651 行，此时PWM的值终于不为 0 了，直接跳到 665 运行else中的内容，将MotorAnwerfen变量设为 1，通知下面该启动了。第 666 行将MotorGestopptTimer定时器设为未来1500ms。

第 669~670 行，由于现在 MotorGestoppt 仍是 1，故 PWM 又被改回 0，并写入寄存器。读者看到这里可能纳闷，好不容易才等到主机发过来的启动信号“50”，怎么这里又把它清零了？其实不必着急，这个启动信号已经起过作用了，刚才将 MotorAnwerfen 变量置 1 就是证明。这里电机还没启动呢，第三章讲过，启动时会用一个默认的值为 10 的 PWM 信号来启动，所以这里写任何 PWM 值进去都无所谓。

接下去与刚才“情景二”最大的不同就是运行到 709 行时的情况，此时 MotorAnwerfen 为 1，铁定进入 if 里的内容运行。711~713 行刚才讲过，不看，看 714 行。终于进入激动人心的启动流程了：第 716 行关红灯；717 行清零 Strom_max；718 行清零启动标志 MotorAnwerfen；然后在 719 行以参数 10 调用 Anwerfen 函数，在这个函数的运行过程中，模拟比较器中断是关闭的，故不会自动换相，而且此函数运行的时间有点长，要带动电机走 32 步（即换 32 次相），如果启动成功，会返回 1；若启动失败，则返回 0。

接着会遇到两种情景：启动失败或是启动成功。若是失败，则跳转到 746 行，再次看一看有没有外来信号，若外来信号不为 0，则将 MotorAnwerfen 置 1，下次循环到 709 行的时候，再尝试重新启动，屡败屡战！

若启动成功，第 721 行会打开绿灯，表示启动成功；然后 722 行将电机停止标志 MotorGestoppt 清零，说明电机现在是非静止状态；723 行 Phase--后实测值为 1；724~725 行给一个极小的 PWM 值并写入寄存器，使电机此时处于无激励的自由旋转状态；然后在 726~727 行使能比较器中断，从此之后，换相的事情就交给模拟比较器中断服务程序自动去完成了，和主程序已经无关了。728~729 行延时 20ms，让电机的转动和换相进入正轨。730~731 行再用一个稍大点的 PWM 值 15 去驱动电机，让电机加速一点；732~741 行再延时 300ms，并且在延时过程中，不断地去检测电流，若电流超过 1/2 额定值就说明有问题，有问题则关所有 MOSFET，红灯闪 10 下，将 MotorAnwerfen 置 1，下次再屡败屡战。

若这 300ms延时下来没有问题，则转到 743 行，重新设定一下DrehzahlMessTimer定时器为未来50ms（因为刚启动嘛，晚点再测转速也不迟）。第 744 行将altPhase设成 7。

之后再回到上面的 while，此时 640 行再探测一下 PWM 因该还是 50，没这么快消失的。

由于 Phase 的取值范围为 0~5，如论如何和不会等于刚才 altPhase 的值 7，故进入 643 行的 if 语句段落。里面的前三句没多大用处，自己看看也不难，最后一句 altPhase = Phase 可使下次再循环不会再进入本段落。之后到了 669 行，此时的 MotorGestoppt 标志已是 0，故不会再把 PWM 清零，第 670 行将这个 50 写入寄存器。

至此，电机就按照给定的占空比正常运转起来了。

➤ 情景四：假设在电机正常运转中，主控板给出停转信号

第一个感受到的还是 640 行，PWM 将被设为 0。之后到 670 行，单片机会输出 0% 占空比的 PWM 信号，电机失去了驱动电压，立刻会慢下来。再过 1500ms 后，会进入 656 行，真正把电机关掉（比较器和 6 个 MOSFET 全关，MotorGestoppt 标志置 1）。之后又是无尽的 while 循环啦。

呼……终于写完了，main 函数里大概就这些东西了，可能第一遍看你还是会觉得有点乱，不要紧，静下心来画一张程序流程图（你可能需要一张很大的纸，不过用 AutoCAD 画也行），等到你能够把图画出来，这个流程你就基本了然于胸了。

5. 高级话题

本章是一个开放性话题，讨论在四轴的无刷电机应用中，怎样才算闭环控制。鉴于笔者自身水平有限，有些问题可能描述得并不正确，欢迎大家前来指正和讨论。阅读本章你需要预先学过《自控原理》或是《信号与系统》之类的课程，否则你可能会对本章内容感到不知所云。不过也没关系，笔者之后会 DIY 设计一个二轴平衡系统，如果能侥幸做成的话，笔者将结合这个实例，再写一篇攻略，以试图将一些控制理论的基本概念讲清楚。

之后再做四轴平衡，接着再放开 yaw 和 z 轴方向自由度，最后再放开 xy 轴方向自由度做到真正的平衡悬停。我想，如果大家能学懂控制理论，基于数学模型而讨论一些问题，做四轴将不再是在黑暗中摸索，论坛的整体技术实力也将达到一个新的高度。

5.1 电机的控制模型

前面说过，无刷直流电机的本质还是直流电机，无论换相机制多么复杂，控制其速度的输入量还是电压，所以可建立以下推导：

1. 电流与转矩的关系：

假定电机转子的磁感应强度 B 为常量，则转矩和电流成正比： $T_m = K_m i_a(t)$

然后进行拉普拉斯变换得： $T_m(s) = K_m I_a(s)$

2. 电流与控制电压的关系：

电压与电流的拉氏变换关系式为： $V_a(s) = (R_a + L_a s) I_a(s) + V_b(s)$ ，其中 R_a 是线圈电阻， L_a 为线圈电感， $V_b(s)$ 为线圈上的反电动势，其大小为： $V_b(s) = K_b \omega(s)$ 。将其代入

上式，得电流为：
$$I_a(s) = \frac{V_a(s) - K_b \omega(s)}{(R_a + L_a s)}$$

3. 负载转矩和转速的关系：

转矩和负载的关系为： $T_m(s) = T_L(s) + T_d(s)$ ，其中 T_L 为负载转矩， T_d 为扰动转矩，通常可以忽略。由转动惯量引起的负载转矩可以写作： $T_L(s) = Js^2\theta(s) + bs\theta(s)$ ，其中 J 是转

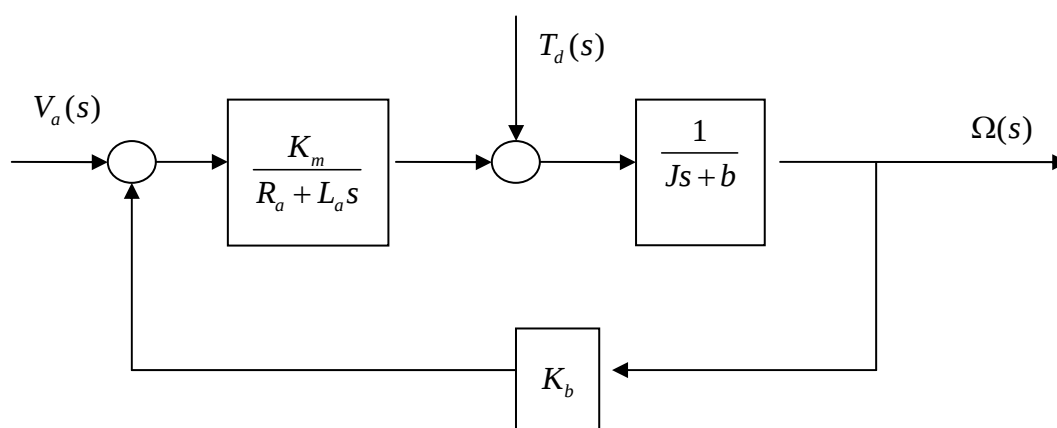
动惯量， b 是和转速成比例的粘滞摩擦系数。

4. 最终传递函数：

结合上几式的内容，再使扰动转矩 $T_d(s) = 0$ ，我们可得到传递函数：

$$G(s) = \frac{s\theta(s)}{V_a(s)} = \frac{\Omega(s)}{V_a(s)} = \frac{K_m}{(R_a + L_a s)(Js + b) + K_b K_m}$$

画成系统框图，应该是这样的：



从上图可以看到，其实这本身已经是个负反馈系统了，转速会以反电动势的形式反馈到电压输入端，所以即便产生扰动，这也是个自稳定系统。

由于一般 L_a 很小，所以 $R_a + L_a s \approx R_a$ ，上面的传递函数可进一步简化为一个一阶系统：

$$G(s) = \frac{\Omega(s)}{V_a(s)} = \frac{K}{\tau_1 s + 1}$$

5.2 四轴上的校正策略

至于 MK 的作者为什么不在电调里加校正环节，一方面可能是因为 MEGA8 的运算能力所限，做了换相等那么多事后，剩下的资源不足以支撑 PID 运算。其二是就算转速精确能精确控制，转速到螺旋桨产生的升力也不是呈严格的对应关系的，存在相当的非线性，而由升力再到整机姿态，也有一定的非确定性，所以与其单独校正转速，不如测得整机姿态后，一起校正。当然，这个部分笔者也正在研究中，这个结论也未必正确，不妥之处欢迎各位讨论。

附录一

MicroChip AN885 文档下载地址:

http://www.ourdev.cn/bbs/bbs_content.jsp?bbs_sn=3228280&bbs_page_no=1&bbs_id=1025

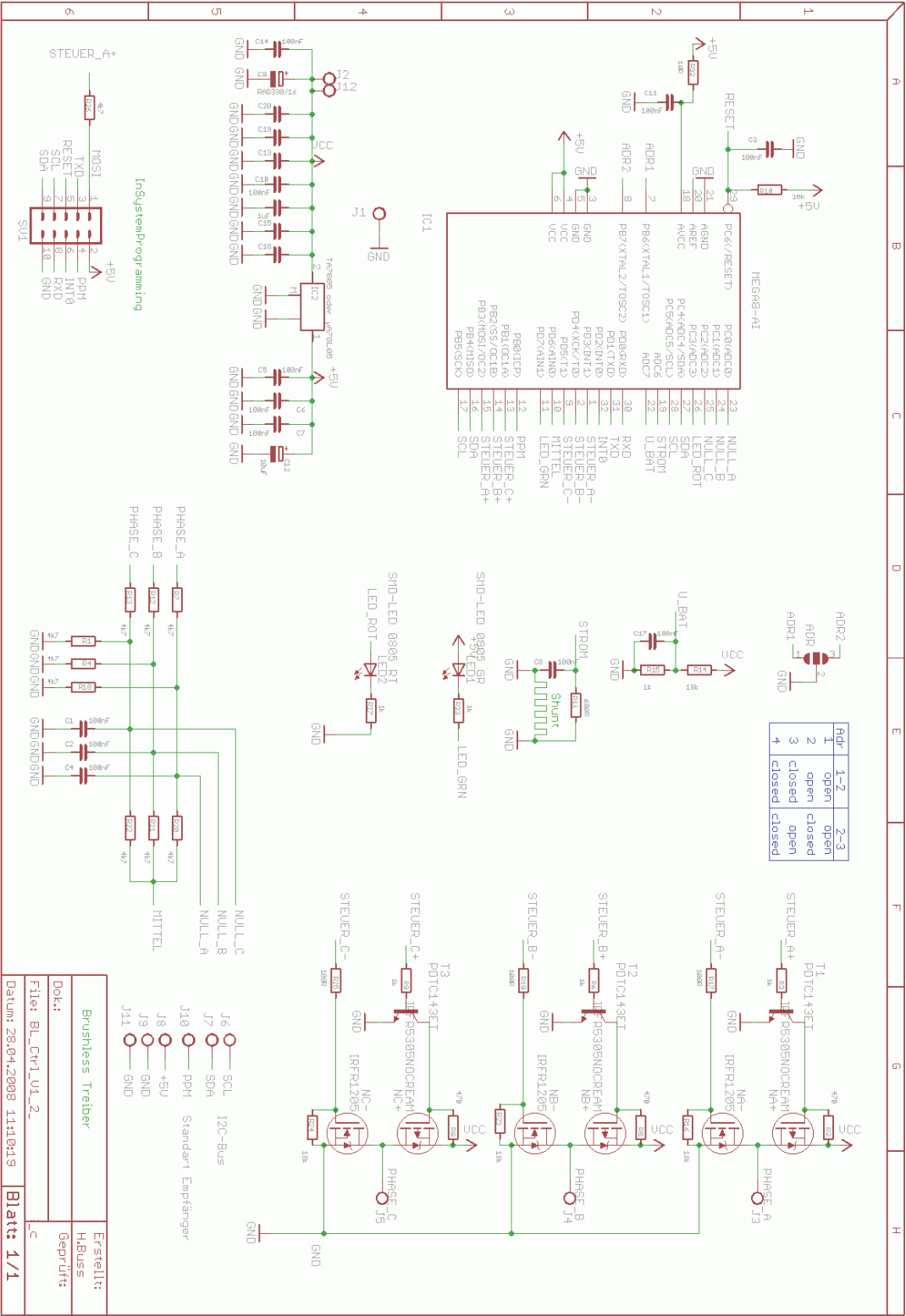
FreeScale PZ104 文档下载地址:

http://www.ourdev.cn/bbs/bbs_content.jsp?bbs_sn=4076156&bbs_page_no=1&bbs_id=1025

Atmel AVR194、AVR492、AVR493 文档下载地址:

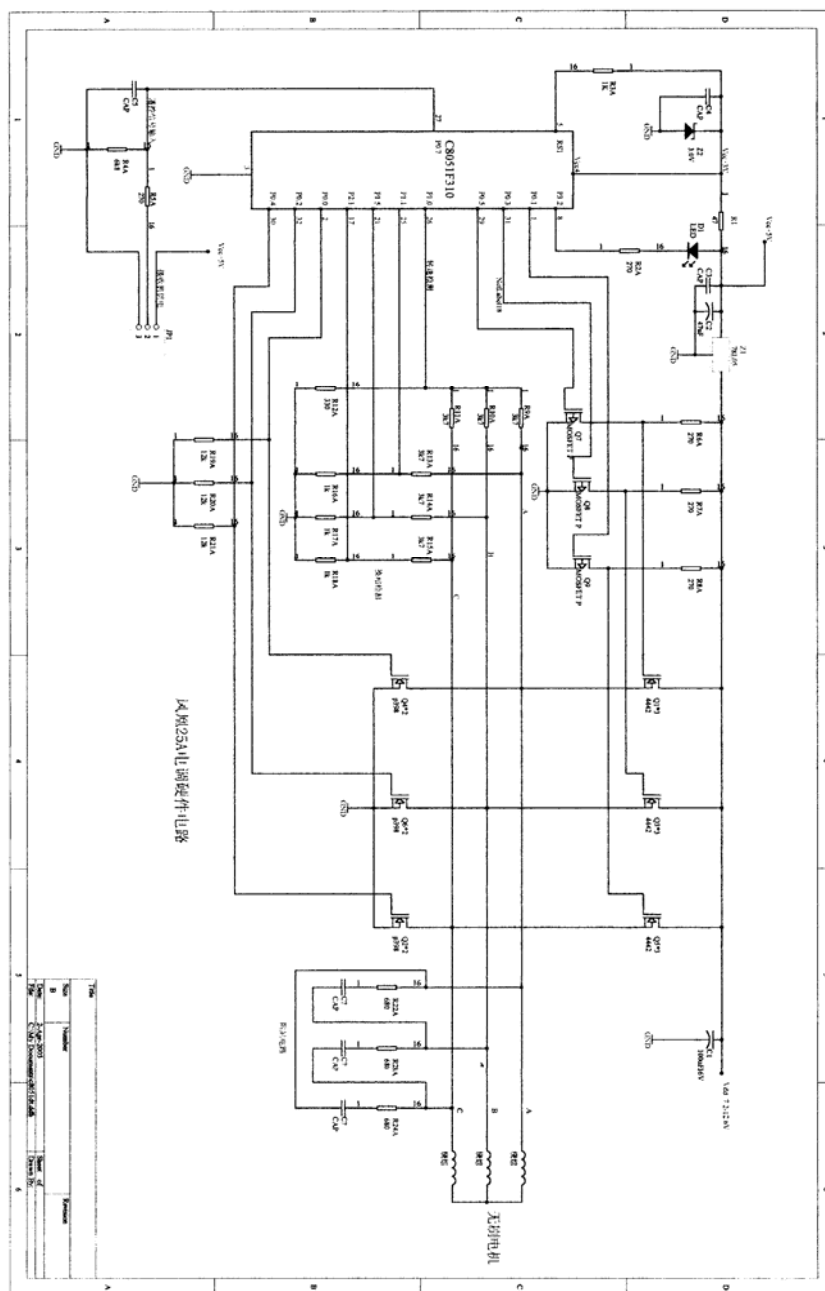
http://www.atmel.com/dyn/products/tools_card_v2.asp?tool_id=3764

附录二



德国 MK 项目 BLDC 电调电路 (V1.2)





凤凰 25A 电调电路 (摘自 5iMX 论坛)

驱动你的无刷电机---关于无刷电机和电调的基本原理（作者：hn_ny_dxs 夏风）

http://www.ourdev.cn/bbs/bbs_content.jsp?bbs_sn=1985447&bbs_page_no=2&bbs_id=1025

DIY 电调的细节以及算法讨论（作者：lijieamd）

http://www.ourdev.cn/bbs/bbs_content.jsp?bbs_sn=3883794&bbs_page_no=2&bbs_id=1025

关于 DIY 电调的一些心得(希望大家踊跃发表自己的心得)（作者：lijieamd）

http://www.ourdev.cn/bbs/bbs_content.jsp?bbs_sn=3671105&bbs_page_no=1&search_mode=3&search_text=lijieamd&bbs_id=1025

德国人电调原理图问题（楼主：rei1984）

http://www.ourdev.cn/bbs/bbs_content.jsp?bbs_sn=3217836&bbs_page_no=1&search_mode=3&search_text=rei1984&bbs_id=1025

附录三

MK 电调程序 V0.41 版本下载地址:

<http://svn.mikrokopter.de/listing.php?repname=BL-Ctrl&path=/tags/V0.41/&#Ab296432c951ee0d2ea8fa11d73430623>

附录四

MikroKopter 飞控、无刷电调源代码(winavr 可编译) 完整的项目开发包（作者： pitolan）

http://www.ourdev.cn/bbs/bbs_content.jsp?bbs_sn=3245280&bbs_page_no=1&search_mode=3&search_text=pitolan&bbs_id=1025