Freescale Semiconductor
应用笔记

Document Number: AN4704
Rev. 1 , 2013/08

采用S12 MagniV MC9S12ZVM的三相无传感器BLDC电机控制套件

作者：Branislav Zigmund、 Petr Staszko和Anita Maliverney

1 简介

本应用笔记介绍采用无传感器算法的三相无刷直流电机(BLDC)控制驱动的设计。该设计针对汽车应用。这种经济高效的解决方案基于飞思卡尔半导体专用于汽车电机控制的MC9S12ZVML128芯片。

该设计展现出MC9S12ZVML128微控制器对于电机控制的适用性和优势。这是一个使用飞思卡尔16位S12 MagniV混合信号MCU设计BLDC控制的示例。

2 系统概念

该系统设计用于驱动三相BLDC电机。应用符合以下系统概念：

- 以MC9S12ZVML128微控制器为目标。

目录

- 采用S12 MagniV MC9S12ZVM的三相无传感器BLDC电机控制套件运行(有关详细信息，请参见编号为MTRCKTSBNZVM128的文档)。有关MC9S12ZVML128 评估板(EVB)的详细信息，请参见MC9S12ZVM128MCBUG （*MC9S12ZVML128评估板(EVB)用户手册*）。
- 该BLDC无传感器控制技术整合了：
  - 三相BLDC电机的6步换向控制
  - 通过BEMF过零点检测技术实现位置感应
  - 速度闭环控制
  - 电机电流限制
  - 启动时对齐
- FreeMASTER软件控制接口(电机启动/停止、速度设置)
- DC母线过压、欠压、过流、过载保护

3 BLDC无传感器控制

3.1 无刷直流电机概述

BLDC电机是一种旋转电机，采用与感应电机相似的传统三相定子。定子内的三相绕组，根据连接方式的不同，可以分为Y型连接或Δ型连接。转子采用表面贴装永磁体。电机的每一相下可以有多个极对。每相极对数就定义了电转速和机械转速的比率。

BLDC电机相当于把有刷直流电机定转子进行交换，使得永磁体转动而绕组保持静止。在有刷直流电机中，换向器和电刷会改变电流的方向，从而使得定子磁场和转子磁场正交。但在无刷直流电机中，是靠功率晶体管 （必须与转子位置同步切换）来实现极性的变换。此过程也称为电子换向。

在转子旋转时，转子永磁体的转动会产生梯形波反电动势。忽略高阶谐波项，电机每相的BEMF(e_a,e_b,e_c)如图 1所示。每相BEMF都具有120° 电角度的恒定幅值，随后在每个半周期内都有60° 的电角度转换。以下公式给出了任何实例上产生的转矩：[1]

公式 1

$$T = \frac{e_a i_a + e_b i_b + e_c i_c}{\omega_{el}}$$

其中：

- ω_{el}为电角速度

在每半个周期中，每一相的理想电流波形(i_a,i_b,i_c)都应该是120° 电子导通角的准方波型。每一相的电流导通必须与BEMF波形的平顶部分相符；这能保证形成的转矩始终恒定或无波动。为了将每一相的电流导通部分与BEMF的平顶部分对齐，必须要知道转子的位置。

1.Rashid， Muhammad H. ed. *《电子器件手册》*，第2版，伦敦：Academic Press， 2007。

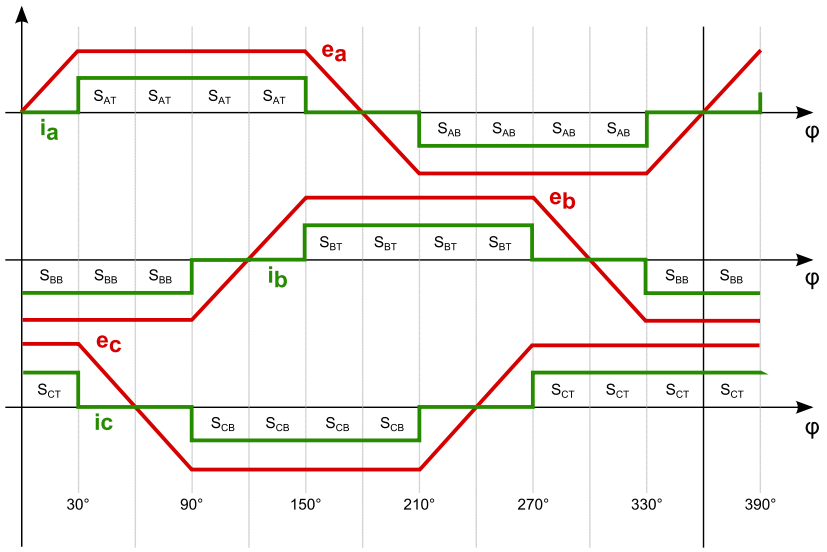


图 1. BLDC电机的三相BEMF电压和相电流

可以通过位置传感器或无传感器算法获取转子位置。可以使用的位置传感器多种多样，但由于转子为永磁体，因此使用简单、可靠、价格低廉的霍尔效应传感器确定物理磁极边缘非常容易。[1]

以下技术常用在BLDC电机无传感器控制的应用中预测转子位置：

- BEMF过零检测法
- 磁通检测法
- 各种系统状态观测器法
- 信号注入法

从控制角度来看，必须用到两种控制过程：

- *换向控制*，使用准方波电流波形，根据转子位置给相位通电。
- *速度/转矩控制*，通过控制准方波相电流波形的幅值，实现所需速度/转矩性能。

以下几节讨论了BEMF过零点检测方法的概念，还介绍了其正确评估方法和条件。

3.1.1 电子换向控制

换向过程提供了利用准方波电流波形根据转子位置给对应绕组通电的机制。由于每个电气周期仅需要6路分立输出(如图 1所示)，因此6个半导体电源开关足以为每一相创建准方波电流波形。 6个半导体开关，使用IGBT或MOSFET，构成三相全桥电路。系统由直流母线电压U_{DCB}供电。在图 2中，半导体开关和二极管视为理想器件。

1.Ibid.

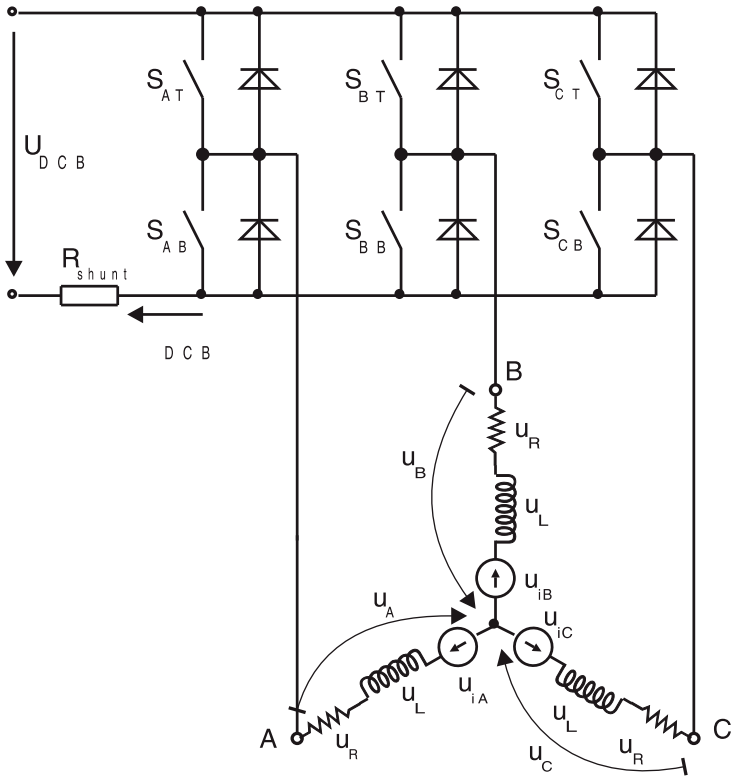


图 2. 功率部分和电机拓扑

*6步换向*是一种驱动三相Y型连接BLDC电机的常用方法。在6步换向控制中， BLDC采用两相工作模式。任意时刻都是两相导通，另一相断开，无传感器算法就是通过检测断开相的反电动势过零来换向的。究竟选择哪两相绕组导通是由位置传感器或者位置观测器来决定的。表 1展示了每个电周期里六步所对应的开关状态和电流方向与转子位置的关系。

表 1. 六步切换顺序

转子位置	扇区编号	开关闭合		相电流		
				A	B	C
330°–30°	1	S_{AT}	S_{BB}	+	–	关
30°–90°	2	S_{AT}	S_{CB}	+	关	–
90°–150°	3	S_{BT}	S_{CB}	关	+	–
150°–210°	4	S_{BT}	S_{AB}	–	+	关
210°–270°	5	S_{CT}	S_{AB}	–	关	+
270°–330°	6	S_{CT}	S_{BB}	关	–	+

3.1.2 速度/转矩控制

BLDC根据其转子位置进行换向可以确保相电流方向正确性，而电机转矩/速度仅取决于准方波电流波形幅值。连续对电机每一相准方波电流幅值的控制是由PWM控制来确保的。

PWM控制常用于采用微控制器的应用。 PWM调制器的占空比由速度PI控制器获取。速度PI控制器将放大所需速度与实际速度之间的误差，其输出经过恰当调整后送到PWM调制器。

实际机械速度可根据轴位置φ_{mech}的时间导数计算得出。

公式 2

$$\omega_{mech} = \frac{d\varphi_{mech}}{dt} = \frac{1}{p}\frac{d\varphi_{el}}{dt} \approx \frac{1}{p}\frac{\Delta\varphi_{el}}{\Delta T}$$

由于在两次换向之间，轴的移动距离恰好为一个电周期(2π弧度)的1/6，因此以上公式可改写为以下形式：

公式 3

$$\omega_{mech} \approx \frac{1}{p}\frac{\Delta\varphi_{el}}{\Delta T} = \frac{1}{p}\frac{\frac{360°}{6}}{T_{CM}}$$

$$= \frac{1}{p}\frac{360°}{T_{(330°\to 30°)} + T_{(30°\to 90°)} + T_{(90°\to 150°)} + T_{(150°\to 210°)} + T_{(210°\to 270°)} + T_{(270°\to 330°)}}$$

$$= \frac{360°}{p\sum_{n=1}^{6} {T_{CM}}^{n}}$$

其中：

- p是极对数
- T_{CM}是两次连续换向间的间隔时间
- ${T_{CM}}^{n}$是扇区n = 1, 2, 3, 4, 5, 6中换向间的间隔时间
- φ_{el}是电角度位置

3.2 互补/独立单极性PWM调制技术

有多种绕组通电和开关的方法。单极性PWM控制技术结合了换向控制和转矩控制。换向控制决定了开关状态， PWM占空比决定了转矩。 BLDC控制的应用中采用单极性PWM控制技术可以降低MOSFET的开关损耗以及提升系统EMC可靠性。

单极性PWM控制意味着电机相线上看只能看到正电压。为实现单极性PWM模式，其中一相采用互补PWM模式，第二相接地，第三相不通电，如图 3所示。电角度每变换60° 就可以观察到这种PWM模式，仅有相序不同。换向控制根据轴位置确定相序。

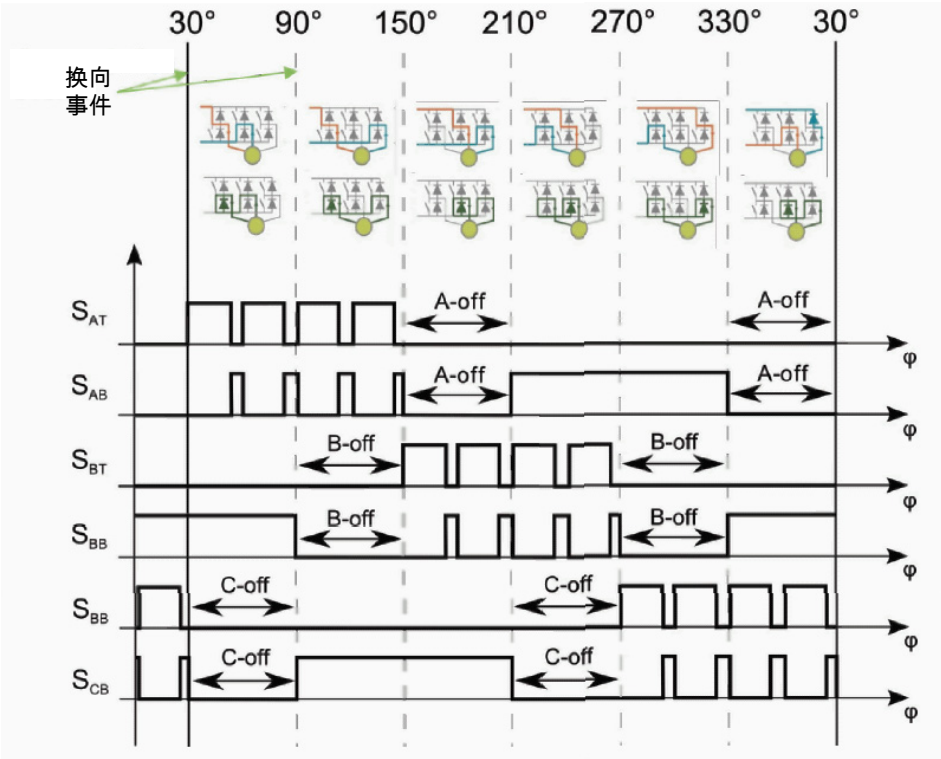


图 3. 互补/独立单极性PWM开关

例如，在第一个周期中， A相由互补PWM信号供电，而B相的底部晶体管接地， C相未通电。在90° 电角度处发生换向事件后， A相仍然由互补PWM信号供电， B相断电， C相转为接地。

本应用说明中描述的控制基于互补/独立单极性PWM调制技术。

以下部分介绍了为实现换向控制，通过BEMF过零点检测估计无传感器位置的方法。

3.3 基于BEMF过零点检测的位置估计

图 4显示了理想BEMF波形，描述了BEMF过零点出现位置之后的30° 电角度处发生的换向事件。BEMF过零点出现在最后一次换向事件所在位置之后的30° 电角度处。假设电机正在以恒定速度转动；在这种情况下，电机从最后一次发生换向事件的位置移动到BEMF过零点所在位置与从BEMF过零点所在位置移动到后续换向事件所在位置的时间是完全相同的。在时域内， BEMF过零点恰好位于两个换向事件的中间。因此，借助定时器， BEMF过零点事件可以轻松预测正确的换向点以及转子速度。

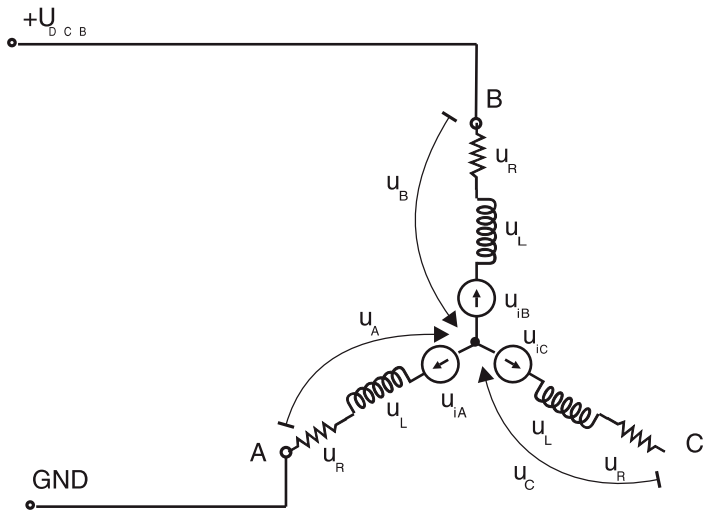


图 4. 过零点检测和换向图

3.3.1 BEMF过零点原则

为解释和模拟BEMF传感技术的概念，本文档提供了基于基本电路拓扑的简化数学模型(请参见图 4)。该数学模型的目标是确定可测量电机波形与BEMF过零点之间的依赖关系。 BEMF过零点又有助于确定换向事件。

该数学模型基于这样一个事实：电机的两相通电，第三个相断开。整个模型的中性点电压级别可以参照母线电压的一半，这样可以简化数学表达式。该数学模型假设电机三相是对称的(请参见图 4)。

公式 4

$$\left.\begin{aligned} u_N &= U_{DCB} - Ri_b - L\frac{di_b}{dt} - e_b \\ u_N &= Ri_a + L\frac{di_a}{dt} - e_a \end{aligned}\right\} \xrightarrow[i_a = i_b]{} u_N = \frac{U_{DCB}}{2} - \frac{e_b + e_a}{2}$$

对于对称三相电机，所有BEMF电压的总和为零，因此：

公式 5

$$e_c + e_b + e_c = 0 \rightarrow e_c = -(e_b + e_a)$$

由于无电流，因此未通电相的电压公式如下：

公式 6

$$u_N = u_C - e_c$$

将公式 5和公式 6代入公式 4即可得出未通电相的相电压，具体如下：

公式 7

$$u_C = \frac{u_{DCB}}{2} + \frac{3}{2}e_c$$

在出现BEMF过零点时， BEMF电压(本例中为e_c)为零，正如其名称所表示的那样。因此，通过测量未通电相的电压(e_c)，并将其与直流母线电压($\frac{U_{DCB}}{2}$)的一半进行对比，即可准确确定BEMF过零点。

3.3.2 BEMF过零点事件检测、相电流测量和互补/独立单极性PWM开关

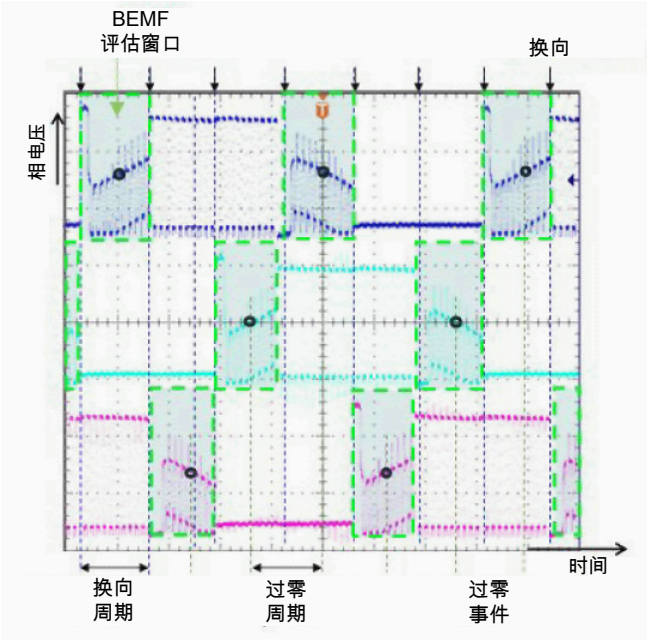


图 5. BEMF过零点事件、换向及其与互补/独立单极性PWM开关的关系

可通过测量未通电相中旋转永磁体所产生的BEMF电压来感应转子的准确位置。

在图 5中，绿色窗口标明了对应相位未通电的时间段。此时间窗口中测量的电压为BEMF电压。在BEMF过零点事件处，永磁体恰好位于线圈前方，转子磁场与定子磁场呈90° 角。此事件在换向周期中间发生，标为绿色BEMF窗口中的黑色圆圈。此时，相电压等于直流母线电压的一半，如 3.3.1节，“BEMF过零点原则”中所示。如果轴速度为恒定，两次连续过零点事件之间的周期将等于换向周期。

图 6 进一步放大观察一个PWM周期。图片顶端是PWM模式，其中A相由PWM控制， C相在整个PWM周期内接地。在PWM On周期中， A相的顶部开关开启， C相的底部开关接地。电流从直流母线流入A相，经由C相和直流母线采样电阻流回。在此周期中，电机路径点显示电压级别 $\frac{U_{DCB}}{2}$ 。未通电相的BEMF电压相对于 $\frac{U_{DCB}}{2}$ 向正方向和反方向变化，这表示在未通电相的相电压等于 $\frac{U_{DCB}}{2}$ 时可以检测到过零点。此外，可以在直流母线采样电阻上测量相电流。

在PWM周期的OFF周期内， A相和C相的底部开关均为开启。因此，相电流将循环通过A相绕组、 B相绕组，然后通过两个底部开关流回。在此周期内，相电流无法流经直流母线采样电阻，相电流无法测量。电机星形点同样接地，在此周期内无法准确测量过零点。

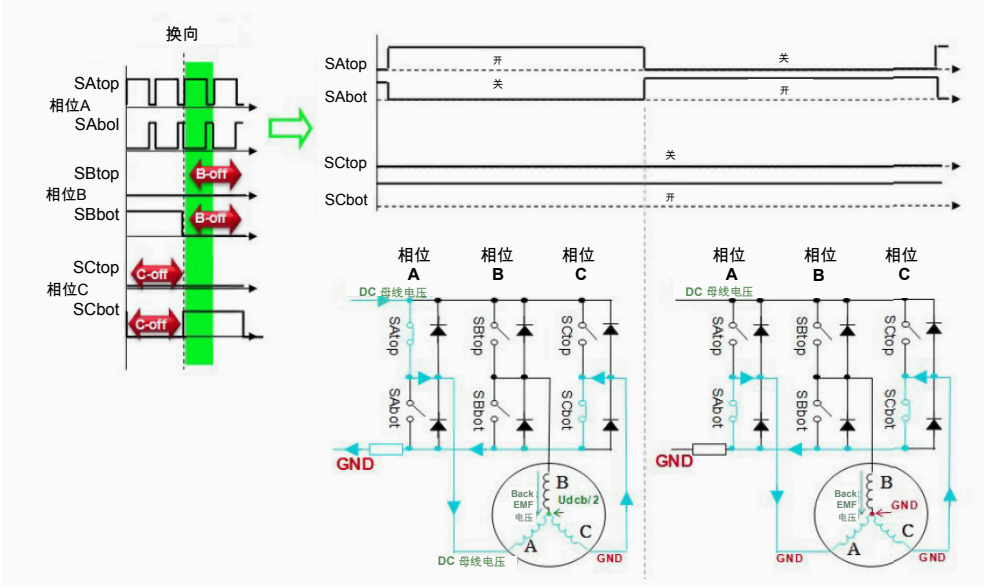


图 6. BEMF过零点检测及互补/独立单极性PWM开关

延续上文的讨论，相电流和BEMF电压测量必须在PWM周期的有效相位中执行。

此外，在换向事件发生后，将立即出现可在相电压上测量到的换向瞬态脉冲，这是由于反向二极管导电时的电流续流产生的。此瞬态的持续时间取决于电机电感、相电流和速度。不应在此时测量BEMF电压。软件必须考虑到此间隔。

3.3.3 BEMF电压测量

如前文所述， BEMF电压仅可在PWM有效相位期间测量。务必注意，由于存在开关噪声，因此要一直测量到有效周期结束。在图 7中，标为绿色的区域显示了应该测量BEMF的时间段。

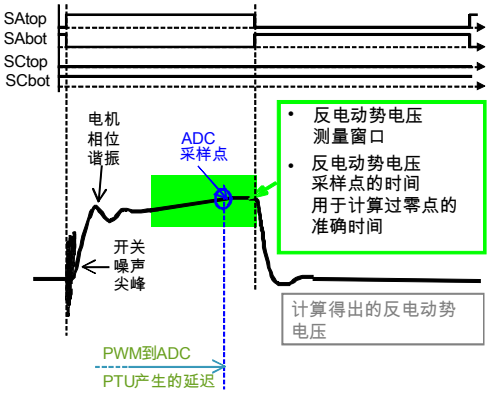


图 7. BEMF测量

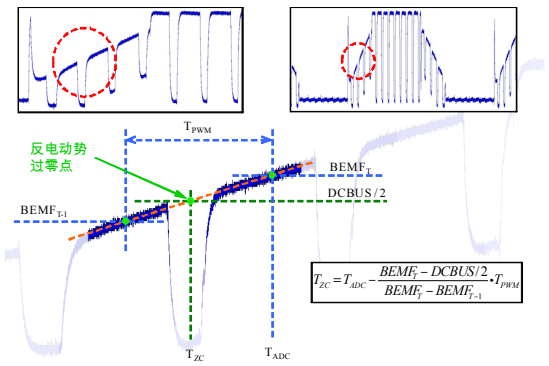


图 8. 精确的BEMF过零点识别

应该说明，根据电机和功率部分参数的不同，电压振荡的幅值、周期和阻尼均有所不同。因此，建议您在测试窗口将近结束时测量BEMF电压。此采样点的时间也需要存储下来，用于增强过零点检测。

如果再次放大观察BEMF电压周期(请参见图 8)，即可看到BEMF电压的过零点和$\frac{U_{DCB}}{2}$级别可能在连续两次BEMF电压测量之间的任意位置出现。为准确估计位置，必须确定准确的过零点。这种准确的过零点是通过基于连续两次BEMF测量值插值的近似结果值确定的。

假设轴未加速，在时间T_{ADC}测量实际BEMF电压，电压级别为e_T，先前的测量值在时间$T_{ADC} - T_{PWM}$测得，电压级别为e_{T-1}，则计算过零点事件准确发生时间的公式应修改为如下形式：

公式 8

$$\frac{e_T - e_{T-1}}{T_{PWM}} = \frac{e_T - U_{DCB}/2}{T_{ADC} - T_{ZC}} \Rightarrow T_{ZC} = T_{ADC} - \frac{e_T - U_{DCB}/2}{e_T - e_{T-1}} T_{PWM}$$

如果在换向期间，将BEMF电压与直流母线电压的一半连续两次进行对比，得出相反符号，就会计算此公式。

为了进一步提高过零点检测的准确性，需要同时测量直流母线电压和BEMF电压。

3.4 直流母线电流测量

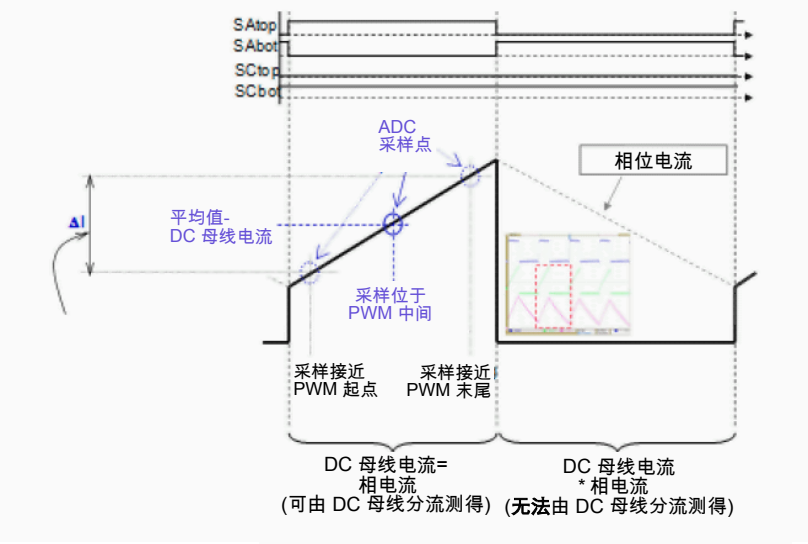


图 9. 直流母线电流测量

如上一节所述，由于直流母线电流仅在有效周期内才等于相电流，因此直流母线电流必须在PWM周期的有效周期内测量，如图 9所示。

在PWM周期的有效周期内，相电流会升高。上升电流的斜率由电机相电感确定，相电感越小，上升电流的梯度就越大。务必在PWM有效周期中间测量电流，以获取平均相电流。

3.5 基于BEMF过零点检测的BLDC无传感器控制的几个阶段

为了启动并运行BLDC电机，控制算法必须经由以下状态：

- 对齐(初始位置设置)
- 启动(强制换向或开环模式)
- 运行(无传感器，运行时使用BEMF采集和过零点检测)

3.5.1 对齐

如前文所述， BLDC电机无传感器控制的主要任务就是位置估计。但在启动电机之前，转子位置为未知。对齐状态的目的在于将转子对齐到已知位置。这个已知位置支持按照所需方向开始旋转轴，并在启动过程中生成最大的转矩。在对齐状态期间，全部三相绕组都导通，这是为了在转轴旋转的任意方向上都获得最好的性能。 A相 和B相连接到直流母线电压的正极， C相接地。对齐时间取决于电机的机械常量，包括负载以及电机所施加的电流。在此状态下，电机电流(转矩)由PI控制器控制。

3.5.2 启动

在启动状态下，电机换向在开环模式下控制，没有任何转子位置反馈。换向周期由开环启动曲线控制。仅在轴速度足够高(大约为电机额定转速的5%)，足以产生可辨识的BEMF电压时，开环启动才结束。

3.5.3 运行

图 10展示了运行状态的结构框图，其中包含BEMF采集及过零点检测，旨在控制换向。电机速度根据过零点时间段预测。所需速度与预计速度之间的差异将馈送给速度PI控制器。速度PI控制器的输出与BLDC电机所施加的电压成比例。电机电流在BEMF过零点事件期间进行测量和滤波，并作为反馈传输给电流控制器。电流PI控制器的输出限制速度PI控制器的输出。限制速度PI控制器输出能够保护电机，避免其超出允许的最大电机电流。

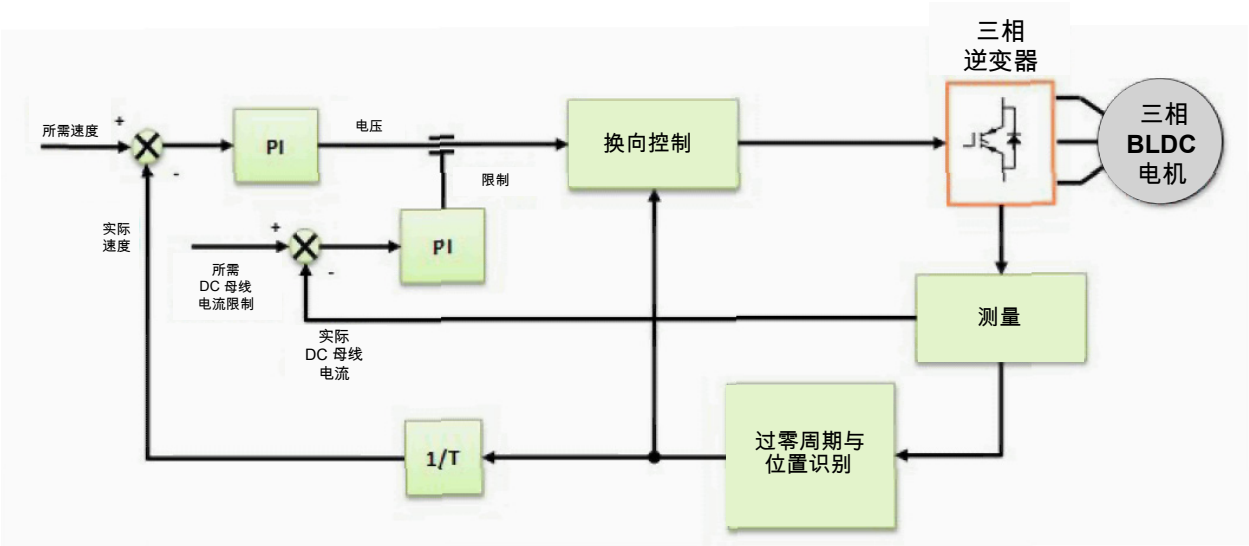


图 10. 带有电流限制的速度控制

4 MC9S12ZVML128上的软件实现

4.1 MC9S12ZVM配置

BLDC无传感器应用框架设计用于满足以下技术规格：

- 针对MC9S12ZVML128评估板(EVB)(有关更多信息，请参见MC9S12ZVM128MCBUG，*MC9S12ZVML128评估板(EVB)用户手册*以及MTRCKTSBNZVM128QSG，*采用MagniV MC9S12ZVML128 MCU的三相无传感器BLDC套件快速指南*)
- PWM输出频率 = 20 kHz
- 电流环路采样周期为1 ms
- 速度环路采样周期为1 ms
- 用于准确的BEMF过零点检测的ADC采样线性近似值。 MC9S12ZVML128的内部分压电阻网络和多路选择器用于相电压测量。
- 利用MC9S12ZVML128内部的电阻分压， ADC模块可以在HD脚测得母线电压。
- 为直流母线电流测量使用内部运算放大器AMP0。

MC9S12ZVML128器件包括多个模块，例如带故障保护功能的脉宽调制器(PMF)、一个可编程触发器单元(PTU)、一个模数转换器(ADC)、一个定时器模块(TIM)和一个适合控制应用(特别是电机控制应用)的栅极驱动单元(GDU)。这些模块直接相连，可配置为满足不同的电机控制应用需求。图 11展示了典型BLDC无传感器应用的模块互连。下面介绍了这些模块，可以在MC9S12ZVMRMV1， *MC9S12ZVM系列参考手册*中找到详细介绍。

4.1.1 模块互连

参与输出执行、数据获取和执行与获取同步的模块构成了所谓的*控制环*。这种控制环包括PMF、GDU、 ADC和PTU模块。控制环的操作极为灵活，可以支持静态、动态或异步时序。

PTU 和 ADC 按照存储在存储器中的列表操作。这些列表定义了PTU的触发点、 ADC的命令和ADC的结果。

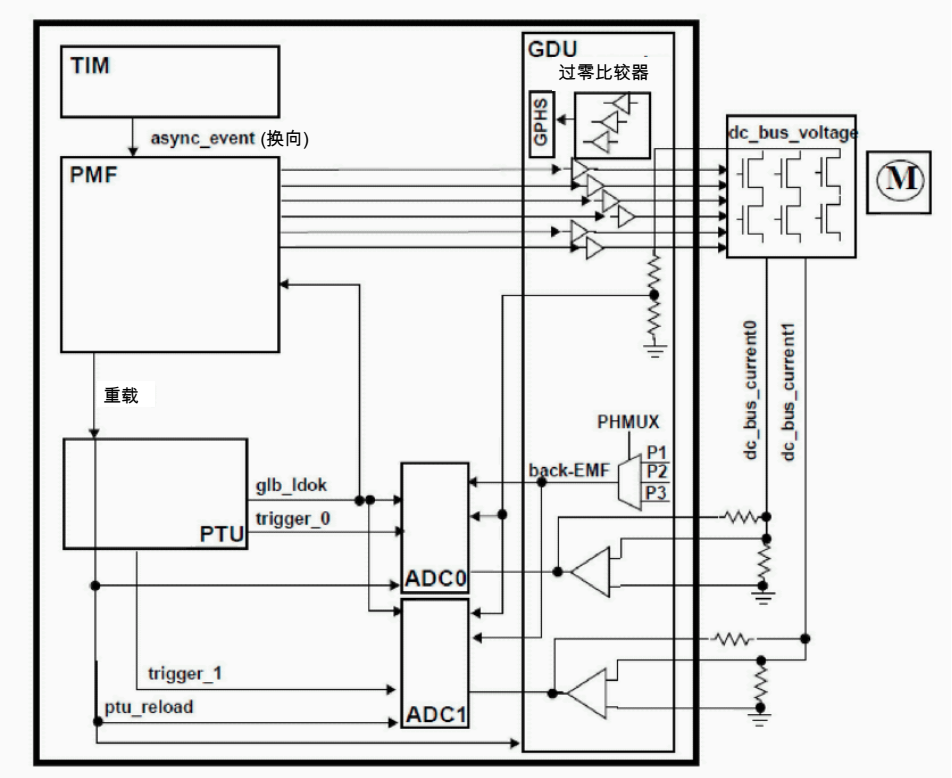


图 11. 无传感器BDLC配置的结构框图

PMF重载事件启动每一个控制环周期。 PMF重载事件重新启动PTU时基。如果启用PTU，则重载将立即作为ptu_reload事件传递到ADC和GDU模块。

PMF在所需的PWM重载频率中生成重载事件。为了从列表中获取首个触发时刻，并为ADCx生成ptu_reload信号，以便开始从命令顺序列表(CSL)载入ADC转换命令， PMF重载事件将导致PTU时基重新启动。

当触发时刻到来时，对应的PTU触发器将为相关ADC生成trigger_x信号。对于同时采样， PTU将生成两个同步trigger_x信号，每个信号对应于一个ADC。确认是trigger_x信号， ADC开始启动CSL中下一个转换序列中的第一项转换(第一个A/D命令已下载)。如需了解更多信息，请参阅MC9S12ZVMRMV1， *MC9S12ZVM系列参考手册*。

通过这种方式， PTU模块即可作为延迟单元，用于调度相对于PWM周期的起点和一个PWM周期以内的状态变量的多次获取。

4.1.2 BLDC无传感器软件控制环中涉及到的模块

本节讨论MC9S12ZVM128L测量以及支持BEMF电压和相电流测量的内部硬件功能。

每个换向事件最初都是由定时器中断触发的。这样的中断可以定义为PWM重载信号。 PWM重载信号本身也会重载PTU模块中的触发器列表，并重启PTU计数器。在PTU计数器达到触发器列表中预定义的值(T1和T2)时， PTU将触发ADC测量。达到T1时触发直流母线电流测量。 T2时将触发另外两个同步测量，一个测量BEMF电压，另一个测量直流母线电压。 ADC转换结果将自动存储到存储器中预定义的队列内。3.3.3节，“BEMF电压测量”和 3.4节，“直流母线电流测量”介绍了相电压、直流母线电压和直流母线电流测量原理。

CPU由ADC转换完成中断触发。它将根据所存储的ADC值计算过零点事件。根据过零点周期计算下一次换向事件的时间。

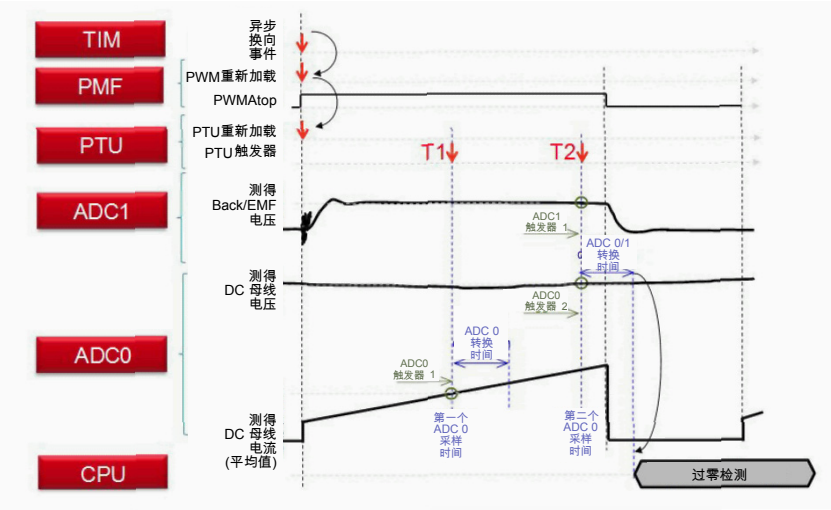


图 12. BLDC无传感器软件控制环中涉及到的模块

TIM、 PTU、 GDU和ADC外设均基于母线时钟。为实现更高的PWM分辨率， PMF模块由内核时钟提供。内核时钟提供的分辨率是母线时钟的二倍。时钟模块使用4 MHz外部晶振，配置为可生成12.5 MHz的母线时钟和25 MHz的内核时钟。

4.1.3 PMF

带故障保护功能的脉宽调制器(PMF)模块配置为以20 kHz的频率(PMFMODA = 1250)生成边沿对齐(PMFCFG0_EDGEx = 1) PWM。为保护逆变器同一桥臂内的MOSFET器件，将死区时间设置为大约0.5 µs (PMFDTMA = 13)。 PWM生成器A作为主模块运行，并产生重载信号作为其他子模块的同步信号(PMFCFG2_REV[0:1] = 1)。每个PWM(PMFFQCA = 0)都会生成重载信号。A对、 B对和C对PWM同步到PWM生成器A (PMFCFG0_MTG = 0)。全部三对均使用值寄存器零(PMFCFG3_VLMODE = 1，写入值寄存器零的值也会写入到值寄存器1到5)设置占空比。每次发生换向事件时(PMFENCA_RSTRTA = 1)， PWM发生器都会重启。

为实现单极性PWM模式，一相采用互补PWM模式，第二相由软件控制模式(PMFOUTC、PMFOUTB)接地，第三相通过屏蔽输出(PMFCFG2[MSK5:MSK0])保持不通电，如图 3中所示。用于控制单极性PWM模式的寄存器(PMFOUTC、 PMFOUTB、 PMFCFG2[MSK5:MSK0])采用双缓冲模式(PMFCFG1_ENCE = 1)。双缓冲信号在换向事件发生时交换。PWM脉宽PMFVAL寄存器也会采用双缓冲，并在GLDOK置位、 PWM重载信号发生时交换。 GLDOK是PTU模块生成的外部信号。 GLDOK在PWM模块中启用(PMFENCA_GLDOKA = 1)

例如，在第一个周期中， A相由互补PWM信号供电，而B相的底部晶体管接地(PMFOUTC = 0x0C，PMFOUTB = 0x2A)， C相未通电(PMFCFG2[MSK5:MSK0] = 0x34)。在90° 电角度处发生换向事件后， A相仍然由互补PWM信号供电， B相断电(PMFCFG2[MSK5:MSK0] = 0x1C)， C相转为接地(PMFOUTC = 0x30， PMFOUTB = 0x2A)。

4.1.4 PTU

可编程触发器装置(PTU)的目标是在控制周期内对于与时序有关的状态变量的采集时，可以完全避免CPU的干预。

PTU模块包含2个触发生成器(TG)。每个TG分别有自己的使能位，因此两个TG可以独立使能。TG0连接到ADC0， TG1连接到ADC1。 PTU模块的触发的产生与到来的重载事件同步。此重载事件将重置并重启内部时基计数器，并确保载入实际触发器列表中的首个触发值。此外，对应的ADC将被告知新控制周期已开始。

如果计数器值与当前的触发值相匹配，则将生成触发器事件。由当前触发值定义的母线时钟周期数将通过这种方式延迟重载事件。所有采集时间值均存储在全局存储器映像内；也就是说，以三维整数数组(PTUTriggerEventList[][][])的形式存储在系统存储内。采集时间值(PTUTriggerEventList[][][])在系统存储器内的准确位置由链接器命令文件指定，并在初始化阶段链接到PTU模块。

```
PTUPTRL = (uint8_t)((long)PTUTriggerEventList);
PTUPTRM = (uint8_t)(((long)PTUTriggerEventList) >> 8);
PTUPTRH = (uint8_t)(((long)PTUTriggerEventList) >> 16);
```

每个TG均仅使用一个列表从存储器载入触发值。主列表(TG0L0IDX/TG1L0IDX)的指针和备用列表(TG0L1IDX/ TG1L1IDX)的指针完全相同。

```
TG0L1IDX = (uint8_t)(((long)&PTUTriggerEventList[0][0][0] - (long)PTUTriggerEventList) >> 1);
TG1L0IDX = (uint8_t)(((long)&PTUTriggerEventList[1][0][0] - (long)PTUTriggerEventList) >> 1);
TG1L1IDX = (uint8_t)(((long)&PTUTriggerEventList[1][0][0] - (long)PTUTriggerEventList) >> 1);
```

即便TG逻辑是在两个指针之间切换， TG也仅使用一个触发器事件的物理列表。触发器列表末尾处启用了TG1的中断(PTUIEL_TG1DIE = 1)。此中断用于存储相应相电压的ADC测量时间。 PTU模块生成LDOK信号，用于通知其他模块软件已经对双缓冲的寄存器进行了更新。

4.1.5 TIM

定时器模块(TIM)是一种基础可缩放定时器，包括由灵活的可编程预分频器驱动的16位软件可编程计数器。 BLDC无传感器算法在输出比较模式下利用两个定时器通道(TIM0TIOS_IOS0 = 1；TIM0TIOS_IOS3 = 1)。

定时器通道0用于识别换向事件。输出比较信号作为async_event在内部路由到PMF模块，以执行PWM对的换向。定时器计数器达到输出比较通道的通道寄存器中的值时，定时器将切换(TIM0TCTL2_OL0 = 1；TIM0TCTL2_OM0 = 0)通道输出。此通道还会在某些应用标志受控的位置安排中断(TIM0TIE_C0I = 1)。

定时器通道3用于通过软件任务控制电机转矩。定时器通道3安排周期性中断(TIM0TIE_C3I = 1)，周期为1 ms (TIM0TC3 = 781)。

定时器的预分频等于4 (TIM0TSCR2_PR = 4)，所有定时器输出比较引脚均断开连接(TIM0OCPD = 0xff)。

4.1.6 GDU

栅极驱动单元(GDU)是一种场效应晶体管(FET)预驱动器，专为三相电机控制应用而设计。 BLDC无传感器控制中采用了以下GDU功能：

- *电荷泵：*用于在PWM以100%占空比运行时，保持高压侧驱动器栅源电压VGS。电荷泵的时钟设置为 $\frac{f_{bus}}{32}$ (GDUCLK2_GCPCD = 2)
- *去饱和错误：*集成三个适用于低压侧FET预驱动器的去饱和比较器，以及三个适用于高压侧FET预驱动器的去饱和比较器。低压侧和高压侧FET的去饱和级别均设置为1.35 V (GDUDSLVL = 0x77)。应用此功能时需要用到FET瞬态期间的消隐时间。消隐时间设置为大约8 µs (GDUCTR = 0x13)。
- *相位复用：*用于选择要在内部路由到ADC1通道2的相电压(GDUPHMUX)。
- *电流感应放大器：*内部电流感应放大器0 (GDUE_GCSE0 = 1)用于测量电机相电流。电流感应放大器0的输出在内部路由到ADC0通道0。

4.1.7 ADC

MC9S12ZVML128使用两个独立的模数转换器(ADC)。两个ADC均为*n*通道多路复用输入连续近似模数转换器。基于列表的架构(LBA)提供灵活的转换序列定义以及灵活的过采样。两个ADC转换命令列表均存储在全局存储器映像内；即以二维字节数组的形式(ADC0CommandList[][]，ADC1CommandList[][])存储在系统存储器内。 ADC转换命令在系统存储器内的准确位置由链接器命令文件指定，并在初始化阶段链接到对应的ADC模块。 ADC结果也采用了相同的策略。转换结果存储在系统存储器内的short型数组中(ADC0ResultList[]， ADC1ResultList[])。

```
// ADC0 Command Base Pointer
ADC0CBP_0 = (uint8_t)(((long)ADC0CommandList) >> 16);
ADC0CBP_1 = (uint8_t)(((long)ADC0CommandList) >> 8);
ADC0CBP_2 = (uint8_t)((long)ADC0CommandList);

// ADC0 Command Base Pointer
ADC0RBP_0 = (uint8_t)(((long)ADC0ResultList) >> 16);
ADC0RBP_1 = (uint8_t)(((long)ADC0ResultList) >> 8);
ADC0RBP_2 = (uint8_t)((long)ADC0ResultList);

// ADC1 Command Base Pointer
ADC1CBP_0 = (uint8_t)(((long)ADC1CommandList) >> 16);
ADC1CBP_1 = (uint8_t)(((long)ADC1CommandList) >> 8));
ADC1CBP_2 = (uint8_t)(long)ADC1CommandList);

// ADC1 Result Base Pointer
ADC1RBP_0 = (uint8_t)(((long)ADC1ResultList) >> 16);
ADC1RBP_1 = (uint8_t)(((long)ADC1ResultList) >> 8);
ADC1RBP_2 = (uint8_t)((long)ADC1ResultList);
```

ADC转换时钟设置为6.25 MHz (ADC0TIM = 0； ADC1TIM = 0)。结果以12位(ADC0FMT_SRES = 4；ADC1FMT_SRES = 4)左对齐数据(ADC0FMT_DJM = 0； ADC1FMT_DJM = 0)的形式存储在存储器内。

两个ADC的转换流均由内部信号(由PTU生成)和数据母线(ADC0CTL_0_ACC_CFG = 3；ADC1CTL_0_ACC_CFG = 3)控制。即便在转换过程中发生换向(ADC0CTL_0_STR_SEQA = 1；ADC1CTL_0_STR_SEQA = 1)，结果也会存储在系统存储器内。

ADC0将安排列表结束中断(ADC0CONIE_1_EOL_IE = 1)，计算BEMF过零点算法。

BLDC无传感器算法使用ADC0测量电机相电流和直流母线电压。 ADC1用于测量对应的电机相电压。

4.2 软件体系结构

本节介绍BLDC无传感器基于过零点算法的软件设计。图 13显示了系统结构框图。

本应用针对MC9S12ZVM电机控制外设进行了优化，以尽可能减少内核在状态变量采集和输出操作应用中的参与。电机控制外设(PMF、PTU、ADC、TIM和GDU模块)在内部相互链接/设置在一起，独立于内核工作，并实现确定时刻模拟量的采集和定子磁场的准确换向。应用的软件部分包括不同模块，详见下文所述。整个应用行为由一台PC使用FreeMASTER工具控制。有关更多详细信息，请参阅MTRCKTSBNZVM128QSG，*采用MagniV MC9S12ZVML128 MCU的三相无传感器BLDC套件快速指南*。

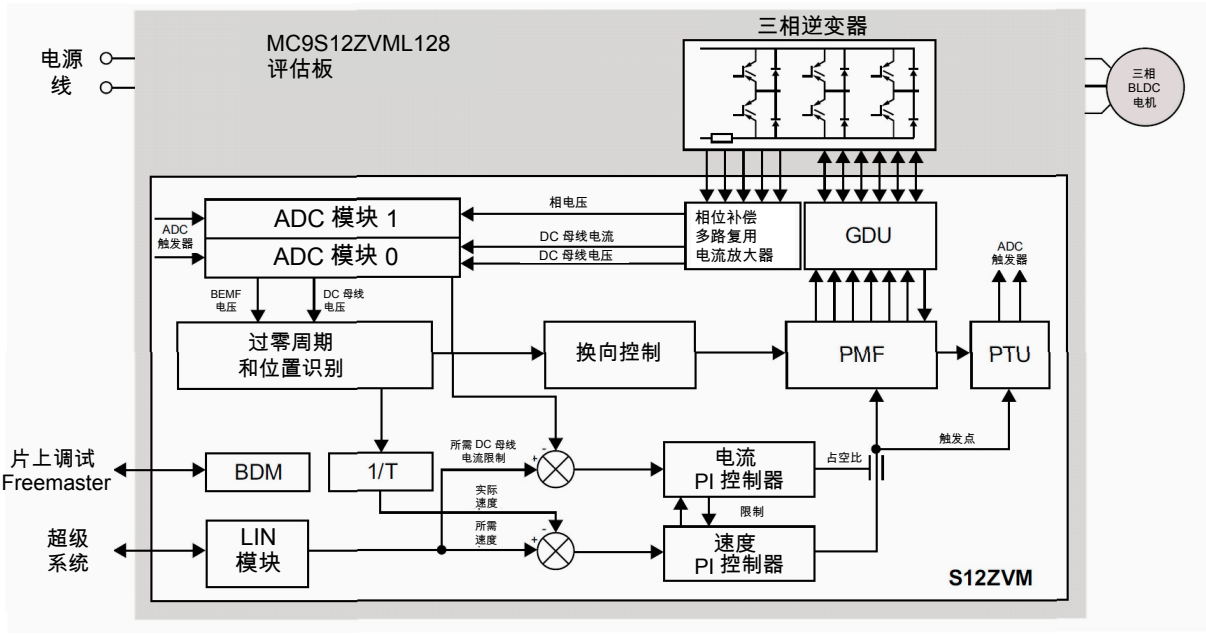


图 13. 系统结构框图

图 13显示的结构框图是控制模块的概述。顶端的框显示了功率部分；图片的下半部分展示了S12ZVM中实现的功能。功率部分采用基于单电阻的系统测量相电流。

浅绿色模块在S12ZVM器件上使用硬件实现；浅蓝色模块使用软件实现，利用电机控制库(MCLib)(有关详细信息，请参阅MC9S12ZVMMCLUG，*MC9S12ZVML128数学和电机控制库用户指南*)。控制环的输入为功率部分测得电压和电流，特别是相电压、直流母线电流和直流母线电压。

作为栅极驱动单元(GDU)一部分的电流感应放大器将放大直流母线电流，随后与直流母线电压一同路由到两个ADC之一处，以采集测量值。相电压由GDU多路复用，随后由第二个ADC使用。

在控制视图中，结构框图分为两个逻辑部分：

- *换向控制*，相电压和直流母线电压用于计算轴的实际位置。根据得出的位置，即可准备下一次换向事件。
- *速度/转矩控制*，所需轴速度将与实际测得的速度对比，并由PI控制器进行调节。速度PI控制器的输出即为占空比。占空比由电流PI控制器限制，并施加到PWM。

应用由一种实时调试工具FreeMASTER控制。 FreeMASTER通过BDM或SCI外设的方式与S12ZVM器件通信。

4.2.1 应用流

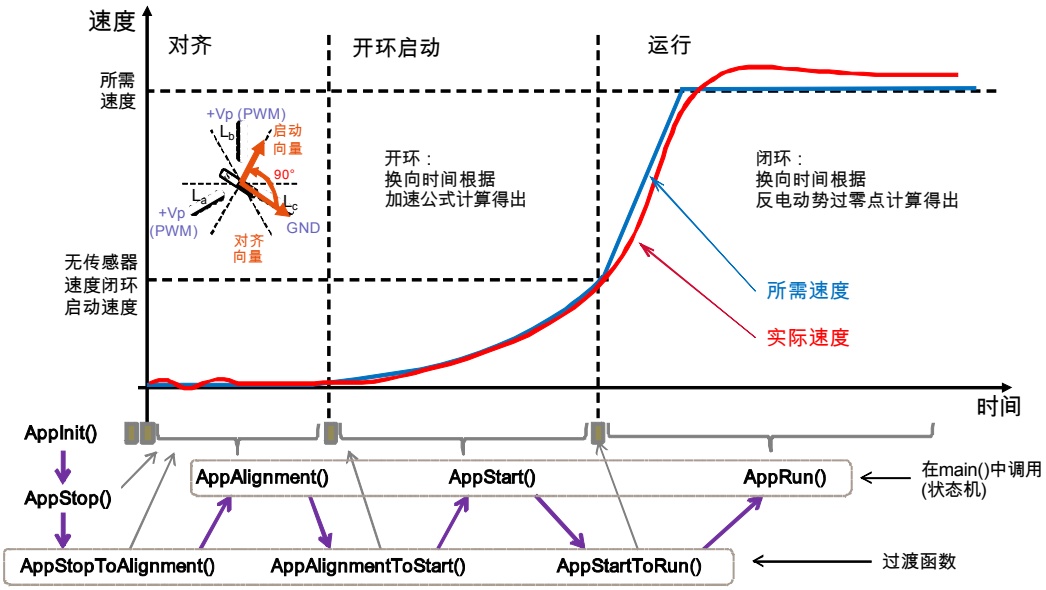


图 14. 应用状态流

图 14解释了不同的应用状态。图中包含两个互连的部件：

- 速度随时间的变化特性
- 图片中下半部分的模块显示了应用状态以及相应状态之间的过渡

应用软件有三种主要状态：*对齐*状态、*开环启动*状态和*运行*状态。在运行状态中， BLDC电机完全在闭环无传感器模式中控制。

外设模块初始化完成后，软件将进入对齐状态。在对齐过程中，转子位置稳定到已知位置，以在两个旋转方向创建相同的启动转矩。这是通过向A相和B相应用PWM实现的。占空比由对齐PI控制器计算。此外还会为C相分配等于零的占空比；也就是说， C相连接到直流母线的负极。 A相和B相的占空比值取决于电机惯性和应用于轴的负载。这样的技术会将轴与C相对齐， C相与定子绕组生成的两个磁通矢量呈正交关系，因此可以确保两个转动方向上均具有相同的启动转矩。对齐状态的持续时间取决于电机的电气和机械常量、应用的电流(表示占空比)以及机械负载。

对齐超时到了后，应用软件将转为开环启动状态。在极低的轴速度下， BEMF电压过低，无法可靠地检测过零点。因此，电机必须在开环模式下控制，并保持一定的时间段。定子绕组生成的第一个向量需要设置为与转子上安装的磁体生成的磁通向量呈90° 的位置。图 14中显示了对齐和第一个启动向量。开环启动状态的持续时间由开环换向次数决定。开环换向次数取决于电机的机械时间常数(包括负载在内)以及施加的电机电流。在开环启动之后，轴速度大约为额定速度的5%。在大约为额定速度5%的速度下， BEMF电压足够高，足以可靠地检测到过零点。

经过既定数量的换向周期后，状态从开环启动状态转变为运行状态。此后就会发生基于BEMF过零点测量的换向过程，控制进入闭环模式。

4.2.2 应用时序和中断

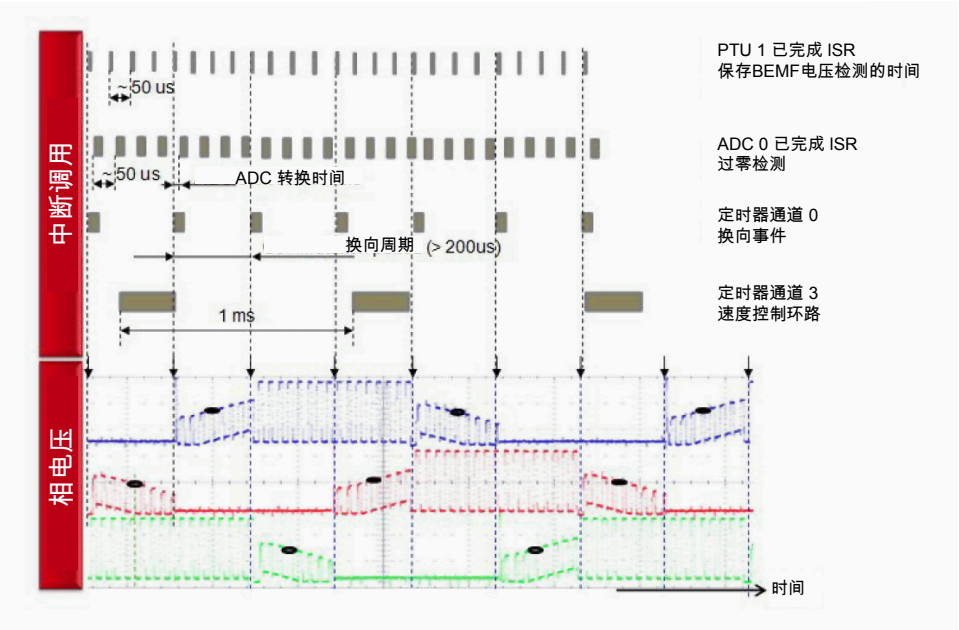


图 15. 应用时序和中断

图 15展示了用于换向、过零点和速度控制的应用时序和相关中断。灰色框显示了所执行的中断例程和相电压测量值。

图中最上面的行显示，在PTU内的所有触发器均已执行时，中断将被激活，这是在每次有效PWM周期结束时触发器单元触发BEMF测量时发生的。此中断内将会保存定时器值，并将其作为BEMF测量参考点。

图中的第二行显示， PWM周期内的所有ADC测量均已执行时激活的中断。此时， ADC转换器已经自动将过零点检测需要的所有值存储到存储器中，随后即可开始执行过零点事件检测和过零点周期的计算。过零点检测完成后，相电压测量的多路复用器切换到在后续换向周期中感应BEMF的那一相。

第三行显示了实际触发换向事件(由定时器通道中断触发)的中断。各次调用之间的时间取决于电机的实际速度。

最后一行显示了用于速度控制环的中断例程。这是在每毫秒设置的定时器通道中断中处理的。实际速度信息与所需速度的变量是速度控制器的输入值。

4.2.3 过零点检测处理

4.2.3.1 获取BEMF测量时间

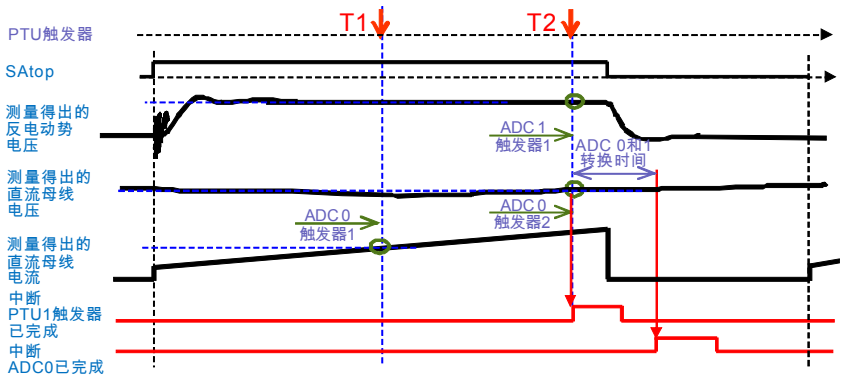
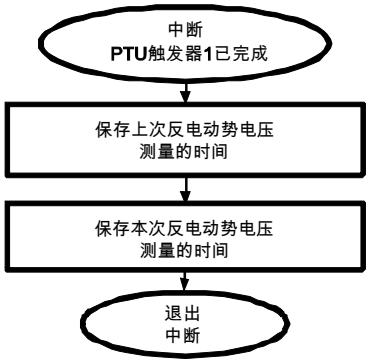


图 16. 直流母线电流、电压和BEMF电压测量值



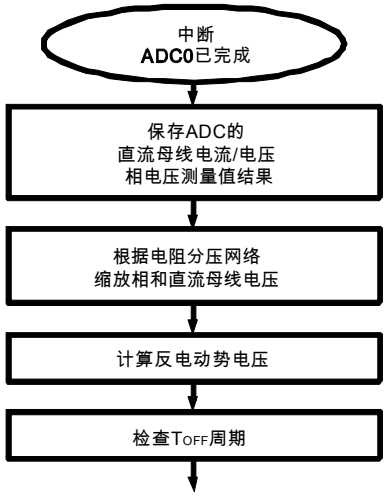

```
INTERRUPT void PTUTrigger1Done_ISR(void)
{
        timeOldBEMF = timeBEMF;
        timeBEMF = TIM0TCNT;

        PTUIFL = PTUIFL_TG1DIF_MASK;// Clear flag
}
```

图 17. 获取PTU触发器完成ISR中的BEMF测量时间

在PTU触发器列表中的所有触发器均已完成调度时， PTU触发完成中断服务程序(ISR)将被激活。PTU、 PTU触发器列表和ADC命令列表配置为在BEMF测量发生时(图 16)，即执行PTU触发完成中断服务程序。在ISR中，上一个周期内获取的BEMF测量时间和自由运行定时器的计数器值将会存储下来，如图 17所示。要计算过零点，必须同时使用两个时间戳。

4.2.3.2 状态变量采集和过零点检测处理

在状态变量采集和过零点检测处理中将使用ADC0完成中断。在ADC0和ADC1达到ADC命令列表末尾处时，将会执行中断服务程序(ISR)；ADC1和ADC0命令列表中的最后一条命令分别是测量相电压和直流母线电压。 ADC1和ADC0将同时测量相电压和直流母线电压。图 18中的流程图和中断例程描述了处理测量的步骤。



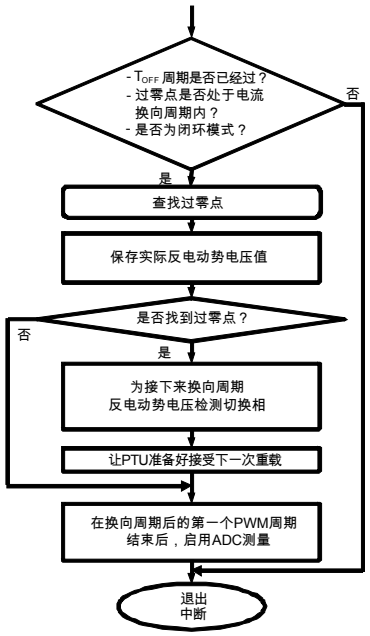

```
INTERRUPT void ADC0done_ISR(void)
{
  DCBusCurrent    = ADC0ResultList[0] - DCBusCurrentOffset;
  DCBusVoltageHalf = MLIB_Mul(ADC0ResultList[1]>>1,13653,F16);
  phaseVoltage = ADC1ResultList[0]>>1;
  bemfVoltage = phaseVoltage - DCBusVoltageHalf;

  if (driveStatus.bit.AfterCMT == 1) {
    if ((timeBEMF - timeCommutation) > timeZCToff){
        driveStatus.bit.AfterCMT = 0;
    }
  }
…
```

图 18. ADC完成ISR中的测量处理

在执行ADC0完成ISR例程之前， ADC会利用DMA，将结果存储到预定义的存储器空间中(也称为ADCx结果列表)。直流母线电流和直流母线电压测量值存储在ADC0结果列表中，相电压存储在ADC1结果列表中。

在ADC0完成ISR例程中，ADC结果根据硬件设置缩放，并保存到适当的变量中。BEMF电压计算为相电压与 $\frac{U_{DCB}}{2}$ 之差。 BEMF电压值为有符号数值。

软件将检查换向瞬态时间是否已经通过(请参见 3.3.2节，“BEMF过零点事件检测、相电流测量和互补/独立单极性PWM开关”)。在软件代码设置中，换向瞬态时间称为T_{off} (timeZCToff)(图 19)。




```
…
  if ((driveStatus.bit.AfterCMT == 0) &&
     (driveStatus.bit.NewZC == 0) &&
     (driveStatus.bit.Sensorless == 1)){

     ZCdetectionAdc[ActualCmtSector]();

     bemfVoltageOld = bemfVoltage;
     driveStatus.bit.AdcSaved = 1;

     if (driveStatus.bit.NewZC) {
       GDUPHMUX_GPHMX = BemfPhase[NextCmtSector];
       PTUC_PTULDOK = 1;
       DCBusCurrentZC = DCBusCurrent;
       DCBusCurrentFiltered = (tFrac16)((long)
                              ((long)DCBusCurrentFiltered +
                               (long)DCBusCurrentFiltered +
                               (long)DCBusCurrentFiltered +
                               (long)DCBusCurrentZC) >> 2);
     }
  }

  if (driveStatus.bit.DisableAdc == 1){
     driveStatus.bit.DisableAdc = 0;
     driveStatus.bit.NewZC = 0;
  }

  ADC0CONIF = 1;       // Clear flag
}
```

图 19. 换向瞬态处理

在换向瞬态时间T_{Off}尚未到期时，将不会执行过零点计算。如果过零点已经在电流换向期间确定，或者应用在开环模式下运行，那么也不会执行此项计算。

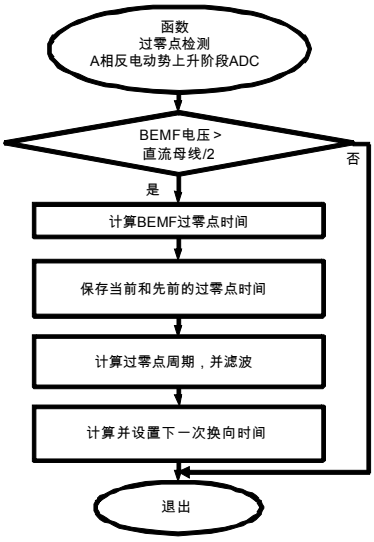
如果不满足上述条件，过零点检测程序就会执行。根据当前换向扇区，代码执行将会略有差异，因此要向将扇区编号0至5传递给算法。

完成过零点位置计算时， BEMF电压值将作为旧值存储，并在下一个PWM周期中再次引用。

确定过零点时，读取3个相电压之一的多路复用器将会切换到与下一个换向周期内的相位电压读数相关的那一相。

此后， PTU的glb_ldok置位，准备接收下一个重载信号，以更改GDU模块中的相位多路复用器。

图 20中的流程图和代码清单描述了前文所述中断内调用的过零点检测例程。根据实际检测到的相和有效扇区，为上升或下降BEMF电压检测函数调用例程。这里的示例展示了A相的BEMF上升时调用的例程。




```
void ZCdetectPhAraisingAdc(void)
{
  tFrac16 delta;

  if (bemfVoltage >= 0){
    // Raising approximation
    delta = bemfVoltage - bemfVoltageOld;
    if ((driveStatus.bit.AdcSaved == 1) &&
        (delta > bemfVoltage)) {
      timeBEMF -= MLIB_Mul(MLIB_Div(bemfVoltage, delta, F16),
                           timeBEMF - timeOldBEMF, F16);
    }
    else{
      // middle of previous ADC sensing events
      timeBEMF -= ((timeBEMF - timeOldBEMF) >> 1);
    }

    lastTimeZC = timeZC;
    timeZC = timeBEMF;

    periodZC_R_PhA = timeZC - lastTimeZC;
    actualPeriodZC = (actualPeriodZC + periodZC_R_PhA) >> 1;

    NextCmtPeriod = MLIB_Mul(actualPeriodZC,
                             advanceAngle, F16);
    TIM0TC0 = timeZC + NextCmtPeriod;

    driveStatus.bit.NewZC = 1;
  }
}
```

图 20. 过零点检测例程

在BEMF电压为负时，尚未通过过零点，因此也无法检测到过零点。软件退出过零点检测例程，过零点状态位仍然未予标记。如果BEMF电压为正数，已经通过过零点，因此用公式 8来计算，即BEMF电压除以两个测量点的增量，并乘以测量得出的PWM周期(BEMF测量周期)。此计算完成后，旧过零点时间和新过零点时间将保存到适当的变量中。随后根据计算得出的过零点时间和上一个换向周期中的过零点时间计算过零点周期。过零点周期还会进行过滤，以提高可靠性。

例程结尾处将会计算新换向时间。这里已经考虑到了部分电机特征。因此并没有将一半的过零点周期加到实际过零点时间之上，而是考虑了所谓的提前角因数，此因数用于比计算时间略提前地激活换向。该因数通常为常数，与电机特征相关。

最后，过零点标为已找到，这样既可保证不在当前换向周期内执行计算。

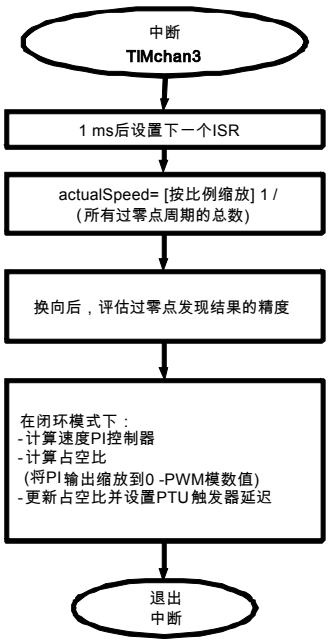
4.2.3.3 速度评估和控制

图 21中的速度控制器在每隔1 ms设置的定时器中断中执行。首先，通过最后 6 个过零点周期计算得出实际速度，并采用折算格式，将其作为实际速度存储。

过零点周期的质量通过函数(StallCheck())评估，如果超出限值，错误将会被存储。

在闭环模式下，将会计算实际速度与所需速度之间的速度误差。这种计算得出的速度误差将代入PI控制器函数。 PI控制器是飞思卡尔电机控制库的一部分(如需了解更多信息，请参见MC9S12ZVMMCLUG， *MC9S12ZVML128数学和电机控制库用户指南*)。 PI控制器函数的输入包括速度误差和 PI 控制器的各种参数，例如比例和积分常数。 PI控制器的输出为占空比，与PWM分辨率呈比例。

在速度控制函数末尾处，占空比将指派给PMF模块，此外还会根据占空比计算新触发点并在PTU模块中更新。




```
INTERRUPT void TIMchan3_ISR(void)
{
  tFrac16 PIOut;
  EnableInterrupts;
  TIM0TC3 = TIM0TC3 + TIMER_1MS;

  if (driveStatus.bit.StallCheckReq == 1) {
      driveStatus.bit.StallCheckReq = 0;
      StallCheck();
  }

  if (driveStatus.bit.Sensorless == 1)  {
    period6ZC = periodZC_F_PhA + periodZC_R_PhA +
                periodZC_F_PhB + periodZC_R_PhB +
                periodZC_F_PhC + periodZC_R_PhC;

    actualSpeed = SPEED_CALC_NUMERATOR / period6ZC;
    speedErr = requiredSpeed - (tFrac16) actualSpeed;
    speedPIOut = GFLIB_ControllerPIrAW(speedErr,
                                       &speedPIPrms, F16);

    duty_cycle = MLIB_Mul(PIOut, PWM_MODULO, F16);
    UpdateDutycycle();
  }

  CheckManualInterface();
  UpdateCurrentLimitingLed();

  TIM0TFLG1 = TIM0TFLG1_C3F_MASK;
}
```

图 21. 速度评估ISR软件流

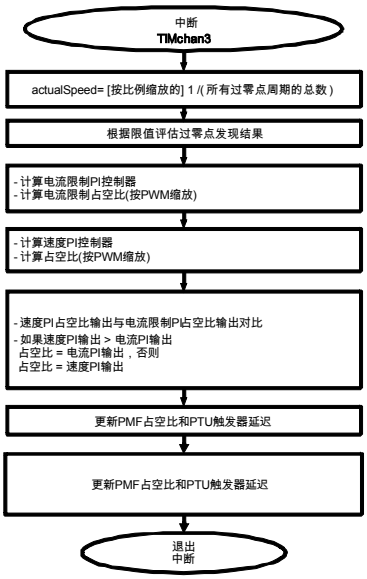
4.2.4 电流限制控制器

电流限制控制器与速度控制器位于同一个1 ms定时器中断内，这是因为两个控制器的输入和输出相互链接。

计算得出实际速度时，可以通过代入实际电流与电机最大允许电流之间的差值来调用电流限制PI控制器。 PI控制器的输出将缩放到与PWM周期成比例的值。在电流PI控制器计算出占空比后，两个占空比输出值将相互比较。

如果速度PI控制器占空比输出高于电流限制PI控制器输出，速度PI控制器占空比输出值将限制为电流限制PI控制器的值；否则，速度PI占空比输出将用作占空比更新值。占空比值将用于更新PMF模块和PTU单元中的触发延迟值。

最后，两个PI控制器的积分部分值需要进行同步，避免其中之一的内部值增加到上限。如果占空比受电流PI占空比输出的限制，则电流PI控制器的积分部分将复制到积分控制器的积分部分，反之亦然。图 22中也描述了上述过程。




```
INTERRUPT void TIMchan3_ISR(void)
{
  tFrac16 PIOut;
  EnableInterrupts;
  TIM0TC3 = TIM0TC3 + TIMER_1MS;

  if (driveStatus.bit.StallCheckReq == 1) {
      driveStatus.bit.StallCheckReq = 0;
      StallCheck();
  }

  if (driveStatus.bit.Sensorless == 1)  {
    period6ZC = periodZC_F_PhA + periodZC_R_PhA +
                periodZC_F_PhB + periodZC_R_PhB +
                periodZC_F_PhC + periodZC_R_PhC;

    actualSpeed = SPEED_CALC_NUMERATOR / period6ZC;
    speedErr = requiredSpeed - (tFrac16) actualSpeed;
    speedPIOut = GFLIB_ControllerPIrAW(speedErr,
                                       &speedPIPrms, F16);
 currentErr = DCBusCurrentLimit - DCBusCurrentFiltered;
    currentPIOut = GFLIB_ControllerPIrAW(currentErr,
                                         &currentPIPrms, F16);

    if (currentPIOut >= speedPIOut) {
      currentPIPrms.f32Acc = ((tFrac32) speedPIOut) << 16;
      currentPIPrms.f16InErrK1 = 0;
      PIOut = speedPIOut;
      driveStatus.bit.CurrentLimiting = 0;
    }
    else{
      speedPIPrms.f32Acc = ((tFrac32) currentPIOut) << 16;
      speedPIPrms.f16InErrK1 = 0;
      PIOut = currentPIOut;
      driveStatus.bit.CurrentLimiting = 1;
    }
  duty_cycle = MLIB_Mul(PIOut, PWM_MODULO, F16);
    UpdateDutycycle();
  }
  else if ((appState == APP_START) ||
           (appState == APP_ALIGNMENT))
  {
    currentErr = ALIGN_CURRENT_SCALED - DCBusCurrent;
    PIOut = GFLIB_ControllerPIrAW(currentErr,
                                  AlignCurrentPIPrms, F16);

    duty_cycle = MLIB_Mul(PIOut, PWM_MODULO, F16);
    UpdateDutycycle();
  }
CheckManualInterface();
  UpdateCurrentLimitingLed();

  TIM0TFLG1 = TIM0TFLG1_C3F_MASK;
}
```

图 22. 速度评估和相电流限制

4.2.5 MC库

应用源代码使用面向MC9S12ZVML128微控制器的飞思卡尔电机控制库(参见MC9S12ZVMMCLUG，*MC9S12ZVML128数学和电机控制库用户指南*)。这个库中包含三个独立库模块：GFLIB、GDFLIB和GMCLIB。GFLIB包含基本数学函数(例如正弦、余弦、斜坡等)。高级滤波器函数是常规数字滤波器库的一部分，标准电机控制算法是常规电机控制库的一部分。

5 FreeMASTER用户界面

FreeMASTER调试工具用于在运行时实时控制应用并监测某一变量。 (有关更多详细信息，请参阅freescale.com/FREEMASTER和文档编号MTRCKTSBNZVM128QSG，*采用MagniV MC9S12ZVML128 MCU的三相无传感器BLDC套件快速指南*)。与主机PC之间的通信通过USB传递。但由于FreeMASTER支持RS232通信，因此通过USB创建虚拟COM端口的主机PC上必须安装物理USB接口CP2102的驱动程序。可以通过silabs.com安装驱动程序。应用将对MC9S12ZVML128的SCI模块进行配置，其波特率为9600 。因此， FreeMASTER用户界面也需要进行相应的配置。

6 结语

本文档中介绍的设计体现了使用MC9S12ZVML128微控制器支持无传感器BLDC电机控制的简单和高效优势，此外也表明这是取代汽车领域中多种低成本应用的理想选择。

7 参考

文档编号	标题	位置
MC9S12ZVMRMV1	*MC9S12ZVM系列参考手册*	freescale.com
MC9S12ZVM128MCBUG	*MC9S12ZVML128评估板(EVB)用户手册*	
MTRCKTSBNZVM128	*采用MagniV MC9S12ZVM的三相无传感器BLDC电机控制套件*	freescale.com/AutoMCDevKits
MTRCKTSBNZVM128QSG	*采用MagniV MC9S12ZVML128 MCU的三相无传感器BLDC套件快速指南*	
MC9S12ZVMMCLUG	*MC9S12ZVML128数学和电机控制库用户指南*	freescale.com/AutoMCLib
—	*FREEMASTER实时调试工具*	freescale.com/FREEMASTER

How to Reach Us:

Home Page:

Freescale.com

Web Support:

Freescale.com/support

Document Number: AN4704
Rev 1, 2013/08