Freescale Semiconductor
应用笔记

Document Number: AN4376
Rev. 0, 10/2011

基于 kinetis 和 MQX 的带霍尔传感器的无刷直流电机控制

作者： Ivan Lovas
Freescale Czech System Laboratories
Rožnov pod Radhoštem，捷克共和国

1 简介

本应用笔记介绍低功率三相 BLDC 电机驱动软件。目的是让设计人员简单、轻松地了解在时间关键的应用中使用 MQX 进行 BLDC 控制的方法。

本应用的控制方案基于使用霍尔效应位置传感器的速度闭环 BLDC 驱动。它可用作 BLDC 电机控制系统在低压电机应用中的示例。本应用在 Freescale Kinetis K60 微控制器上运行。该硬件基于 Freescale 塔式快速原型系统，它包含以下模块：

- TWR-Elevator
- TWR-K60N512
- TWR-MC-LV3PH
- TWR-SER

目录

可提供两个版本的应用软件。其中一个带 MQX 实时操作系统，另一个为裸机版。两者都使用相同的电机控制源代码。 MQX 版本包含一个 Web 服务器来展示 MQX 解决方案的优势。

两种应用版本均可通过 FreeMASTER 控制。

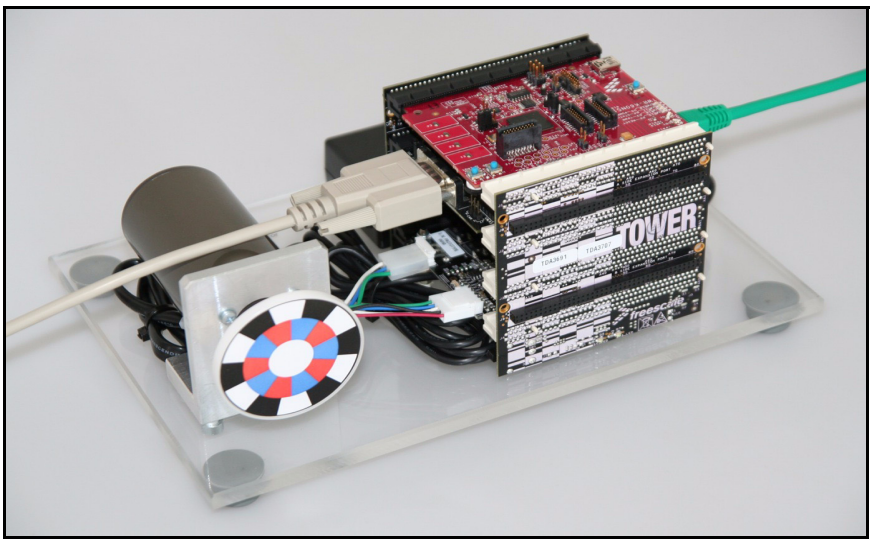

图 1. 使用 Kinetis 进行 BLDC 电机控制演示

1.1 应用特性

- MK60 Kinetis ARM Cortex-M4 微控制器
- 硬件构建于塔式快速原型系统上
- 使用霍尔效应传感器确定位置，实现 BLDC 电机控制
- 带速度测量和可调整参数的速度闭环
- 可调转速斜坡
- 多功能性，得益于小数算法
- MQX 允许轻松结合使用其他应用和进行快速原型设计
- FreeMASTER 软件，用于应用控制和监控
- 网页，用于通过以太网进行应用控制
- 两个旋转方向的电机模式
- 最小转速为 500 rpm
- 最大转速为 4000 rpm
- 过压、欠压及过流故障保护

1.2 内核特性

32 位 Kinetis MCU 是业界最具扩展能力的 ARM® Cortex™-M4 MCU。该产品组合先期推出的产品包括 5 个系列，200 多款引脚、外设和软件都兼容的 MCU，具有出色的性能、存储器和功能扩展能力。由于采用了创新的 90nm 薄膜存储器闪存技术，并带有独特的 FlexMemory（可配置嵌入式 EEPROM）， Kinetis 包含最新的低功耗创新技术和高性能、高精度的混合信号功能。 Kinetis MCU 还得到了飞思卡尔和 ARM 第三方生态体系提供的市场领先的实施工具包的支持。

2 通过 MQX RTOS 进行电机控制

2.1 什么是 MQX?

Freescale MQX 软件解决方案提供基于模块化架构的简单的 API，有助于微调定制应用，并可加以扩展，以支持多样化的应用需求。飞思卡尔将经过市场检验的 Freescale MQX 软件解决方案与芯片产品相结合，提供结构精简、功能强大的平台，满足硬件、软件、工具和服务的需求。

2.2 何时通过 MQX RTOS 进行电机控制

MQX 并非用于电机控制应用的典型操作系统。 MQX OS 适合于带高级功能的大型应用，如网页控制、 USB、显示以及在专用设备上读取和运行 SDHC 卡 （通常为单核）。 MQX 的主要优势在于其包含 RTCS、以太网、 USB 通信、 MFS 文件系统以及许多其他应用的库。

2.3 如何通过 MQX 进行电机控制

MQX RTOS 是一个具有动态分配和 POSIX 调度功能的复杂系统。其默认的系统节拍持续时间为 5 ms。该时长是大多数应用的理想之选。但是，这也意味着 MQX 的任务时间分辨率比电机控制要求所需的分辨率长 1000 倍以上。因此，很明显地，电机控制进程需要通过高优先级中断才能进行。

这些中断可由标准 MQX 中断程序提供。中断请求至执行服务程序之间的持续时间通常为几微秒（具体取决于 CPU 版本和时钟速度）。需要时，可以使用内核中断实施电机控制算法。内核中断为自然的 CPU 中断，无 MQX 开销和最小执行持续时间限制。内核中断的劣势是不支持诸如事件或信号量等 MQX 功能。

详情请参见 MQX 文档中章节 8 “参考文献” 3。

3 BLDC 电机理论

3.1 BLDC 基本信息

BLDC 电机是一种旋转电机，其定子为类似感应电机的传统三相定子，转子使用表面贴装的永磁体 （参见图 2）。

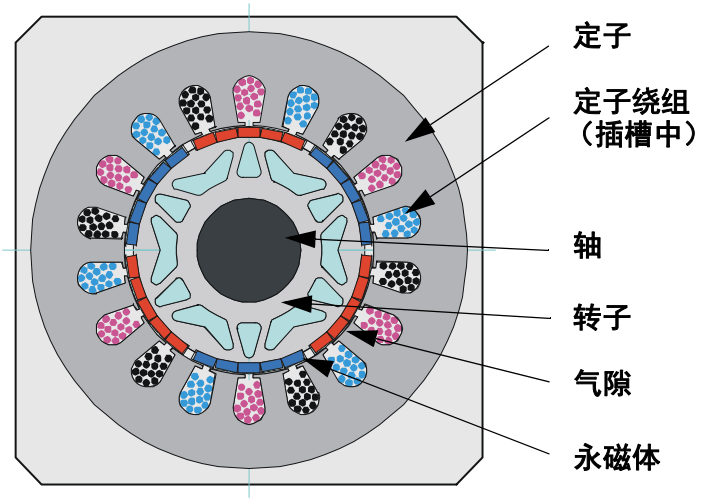

图 2. BLDC 电机 — 横截面

就这点来说，BLDC 电机与颠倒的直流换向器电机等效，BLDC 电机的磁体保持转动，导体保持静止。在直流换向器电机中，电流极性通过换向器和电刷改变。另一方面，在无刷直流电机中，极性反转则通过与转子位置同步的功率晶体管切换实现。因此，BLDC 电机一般都具有内部或外部位置传感器，以检测实际转子位置，或者不使用传感器进行位置检测。

3.2 BLDC 电机数字控制

BLDC 电机由矩形电压脉冲以及给定的转子位置驱动 （参见图 3）。产生的定子磁通与转子磁体产生的转子磁通相互作用，产生电机的扭矩，进行产生转速。必须将电压脉冲恰当地施加给三相绕组系统中的两相，使得定子磁通与转子磁通之间的角度保持接近 90°，以产生最大的扭矩。为此，电机需要采用电子控制装置来保证正常运行。

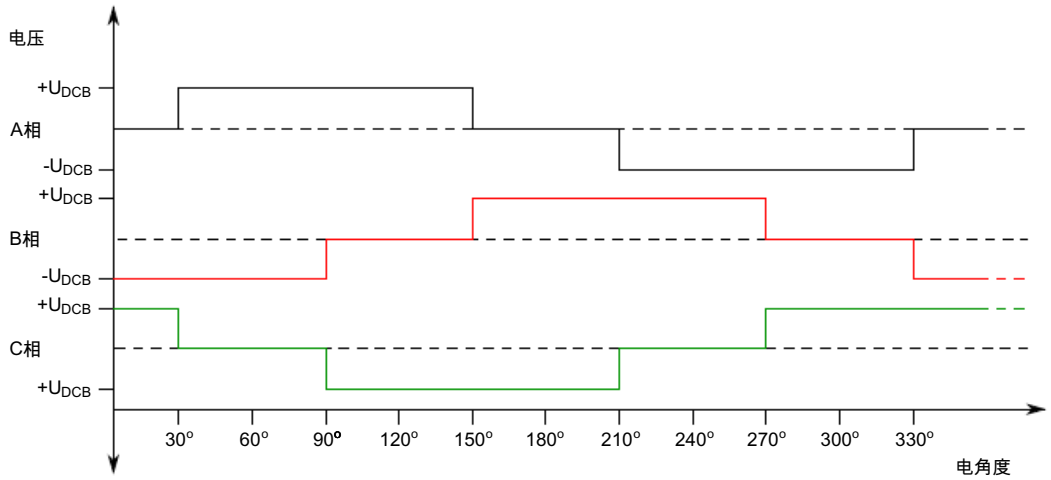

图 3. 施加到三相 BLDC 电机的电压脉冲

对于常见的三相 BLDC 电机，可以使用标准的三相功率级，如图 4 所示。该功率级使用 6 个功率晶体管。

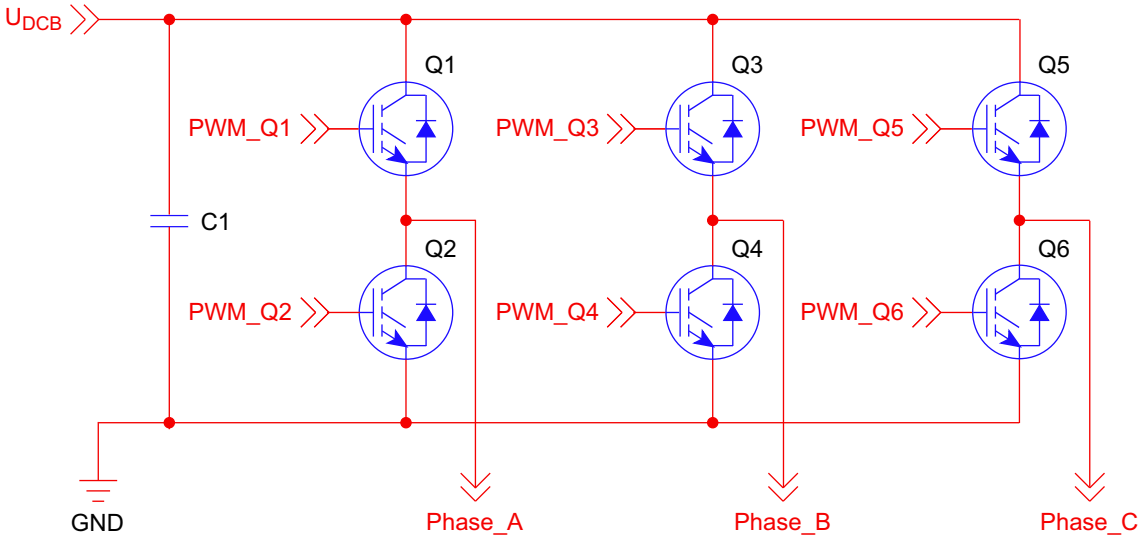

图 4. 三相 BLDC 功率级

在两种模式下，三相功率级可同时为电机的两相供电。第三相不通电 （参见图 3）。因此，我们通过 PWM 技术获得施加给 BLDC 电机的六种可能的电压向量。功率晶体管的切换有两种基本类型：互补切换和独立切换。

3.3 互补与独立切换

如果为互补切换，当电机某相连通电源时，两个晶体管导通。但在续流时存在差异。如果为独立切换，所有晶体管关闭，电流将继续按相同方向流经续流二极管，直到其降至零。与之相反，在互补切换时，互补晶体管将在续流时打开。因此，电流可能会按相反的方向流动。图 5 所示为互补切换。

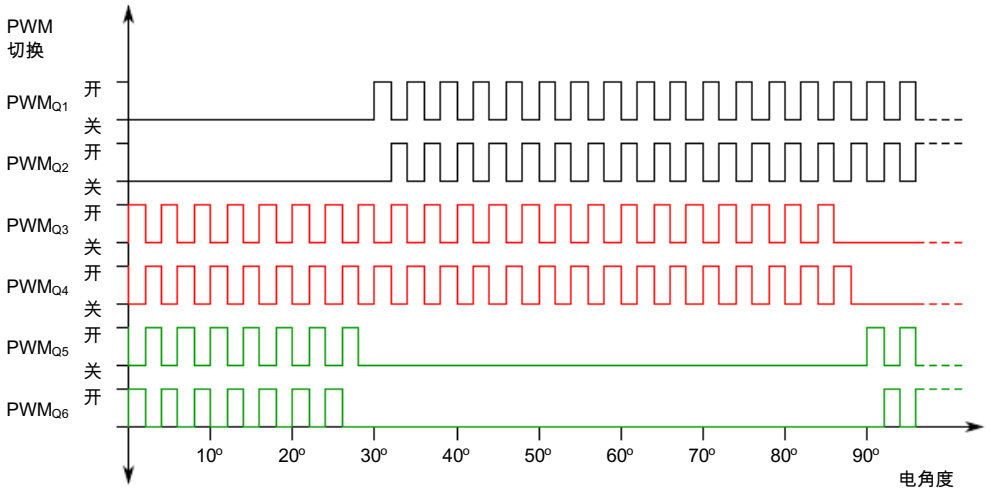

图 5. 功率晶体管互补切换

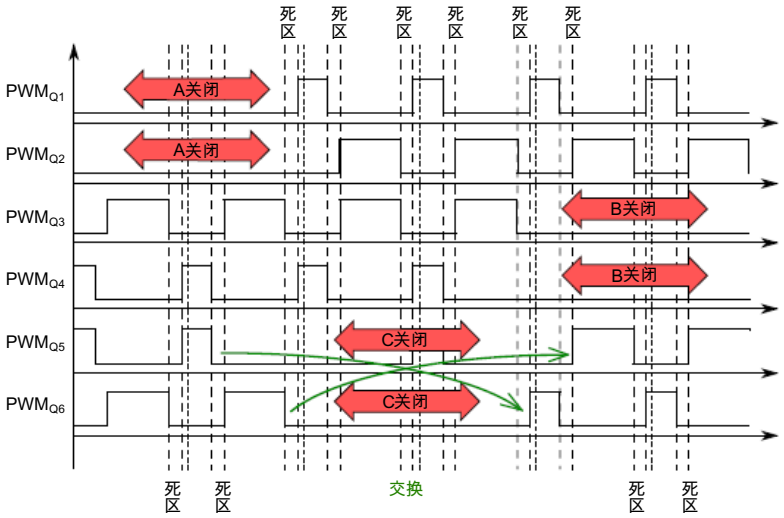

图 6. 双极性 PWM 切换 — 详细信息

技术详情如图 6 所示。在本示例中，我们可以了解双极性四象限（互补）切换的特征。双极性切换需要交换顶部和底部开关 PWM 信号。另一个需要详细说明的是在互补顶部和底部信号中进行死区插入。此死区插入是所有四象限功率级的典型操作。四象限操作通过顶部和底部开关的互补操作 （一个相的底部开关差不多为顶部开关的负序）实现。

这就需要插入死区，因为开关瞬态将会导致直流母线短路，对功率级造成致命损坏。

由于较严重的电磁辐射，双极性 PWM 开关并不像单极性开关一样常见。这是因为 PWM 纹波为直流母线电压的两倍。另一方面，该开关方法更适合于无传感器转子位置检测。

详情请参见文档中章节 8 “参考文献” 1 和 2。

3.4 换相

换相可以产生旋转场。我们解释过，为了使 BLDC 电机正确工作，必须将定子和转子磁通之间的角度保持为接近 90°。通过六步控制，我们总共可以获得六种可能的定子磁通向量。必须在特定的转子位置更改定子磁通向量。转子位置通常通过霍尔效应传感器检测。霍尔传感器生成三种信号，这三种信号也包含六种状态。每个霍尔传感器的状态对应特定的定子磁通向量。所有霍尔传感器状态及其对应的定子磁通向量如图 7 所示。相同数字如表 1 所示。

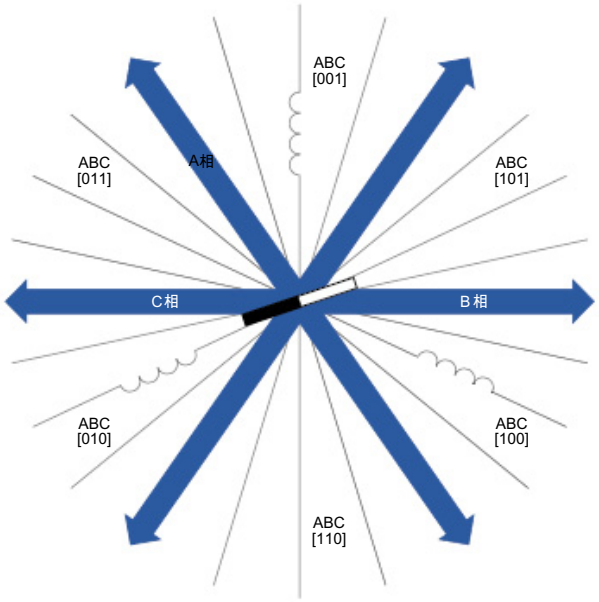

图 7. 带六步控制的定子磁通向量

可以看到，使用六步控制技术后，不可能将转子磁通和定子磁通之间的角度精确地保持在 90°。实际角度在 60° 至 120° 之间变化。

每六十电角度重复换相一次。换相事件对于其角度（时间）准确性非常重要。任何偏差都会导致扭矩波动，从而导致速度出现偏差。

表 1. 顺时针旋转换相序列

霍尔传感器 A	霍尔传感器 B	霍尔传感器 C	A 相	B 相	C 相
0	1	1	$-V_{DCB}$	$+V_{DCB}$	NC
0	1	0	NC	$+V_{DCB}$	$-V_{DCB}$
1	1	0	$+V_{DCB}$	NC	$-V_{DCB}$
1	0	0	$+V_{DCB}$	$-V_{DCB}$	NC
1	0	1	NC	$-V_{DCB}$	$+V_{DCB}$
0	0	1	$-V_{DCB}$	NC	$+V_{DCB}$

3.5 速度控制

换相可确保 BLDC 电机的转子正确旋转，而电机转速则仅取决于所施加电压的振幅。可使用 PWM 技术调整所施加电压的振幅。所需转速由速度控制器控制，速度控制器由比例 - 积分 (PI) 控制器

实现。实际转速与所需转速之差为 PI 控制器的输入，该控制器随后根据此差值控制 PWM 脉冲的占空比，并且此占空比与维持正确转速所需的电压振幅相对应。

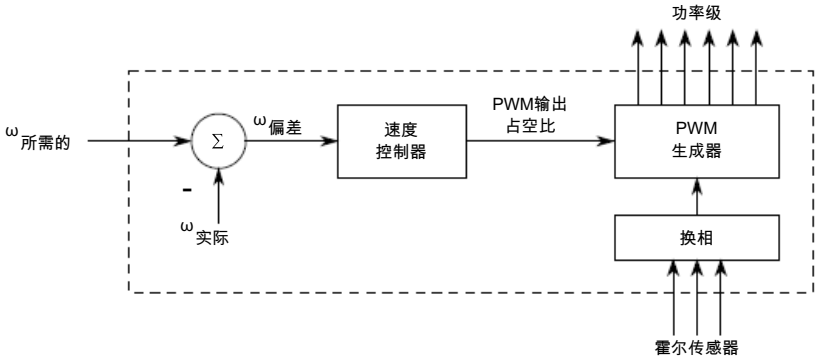

图 8. 速度控制器

速度控制器按照以下等式计算 PI 算法：

$$u(t) = K_c\left[e(t) + \frac{1}{T_I}\int_0^t e(\tau)d\tau\right] \quad \textit{等式 1}$$

使用积分逼近后退欧拉法将等式转化为离散时域之后，我们将会获得以下数字 PI 控制器计算等式：

$$u(k) = u_P(k) + u_I(k) \quad \textit{等式 2}$$

$$u_P(k) = K_c \cdot e(k) \quad \textit{等式 3}$$

$$u_I(k) = u_I(k-1) + K_c\frac{T}{T_I} \cdot e(k) \quad \textit{等式 4}$$

其中：

e(k)	=	第 k 步中的输入误差
w(k)	=	第 k 步中的所需值
m(k)	=	第 k 步中的测量值
u(k)	=	第 k 步中的控制器输出
$u_p(k)$	=	第 k 步中的比例输出部分
$u_I(k)$	=	第 k 步中的积分输出部分
$u_I(k–1)$	=	第 k–1 步中的积分输出部分
T_I	=	积分时间常数
T	=	采样时间
K_c	=	控制器增益

4 系统描述

4.1 应用概述

该系统设计用于驱动三相 BLDC 电机。该应用符合以下性能规范：

- 使用霍尔效应传感器对 BLDC 电机进行电压控制
- 带 TWR-K60N512 板的塔式系统解决方案
- 电源电压 +24 V_{DC}
- 控制技术包括：
  - 使用霍尔效应传感器信号进行位置检测
  - 使用速度闭环进行 BLDC 电机电压控制
  - 速度测量基于一个霍尔效应传感器
  - 两个旋转方向
  - 可从任何转子位置启动
  - 可在每次电机启动之前对 MOSFET 前置驱动自举预充电
  - 最小转速 500 RPM （取决于所使用的电机）
  - 最大转速 4000 RPM （取决于所使用的电机）
- FreeMASTER 接口 （输入速度、测得速度、速度误差、斜坡参数、过流 LED 指示）
- 以太网终端 （输入速度、以太网状态）
- 故障保护：
  - 直流母线过流故障保护 （硬件）
  - 电源反向极性保护电路 （硬件）

4.2 应用说明

MCU PK60N512VMD100 运行主控制算法。根据用户接口和反馈信号，它将会生成适用于三相逆变器的三相 PWM 输出信号。

整个应用在中断中运行，以便在 MQX 下使用。如果所需速度不为零，则应用启用霍尔传感器中断并强制首次调用霍尔中断程序。每个霍尔传感器的新边沿自动调用中断程序。在此程序中，将会扫描来自霍尔传感器的信号，并交换和屏蔽对应的 PWM 通道。此过程称为换相。霍尔传感器扫描独立于速度控制。速度控制循环由 PIT 0 定时调用。此周期循环中存在速度斜坡和应用状态机。主程序中仅存在一个带 FreeMASTER 轮询功能的无线循环，以控制此应用。

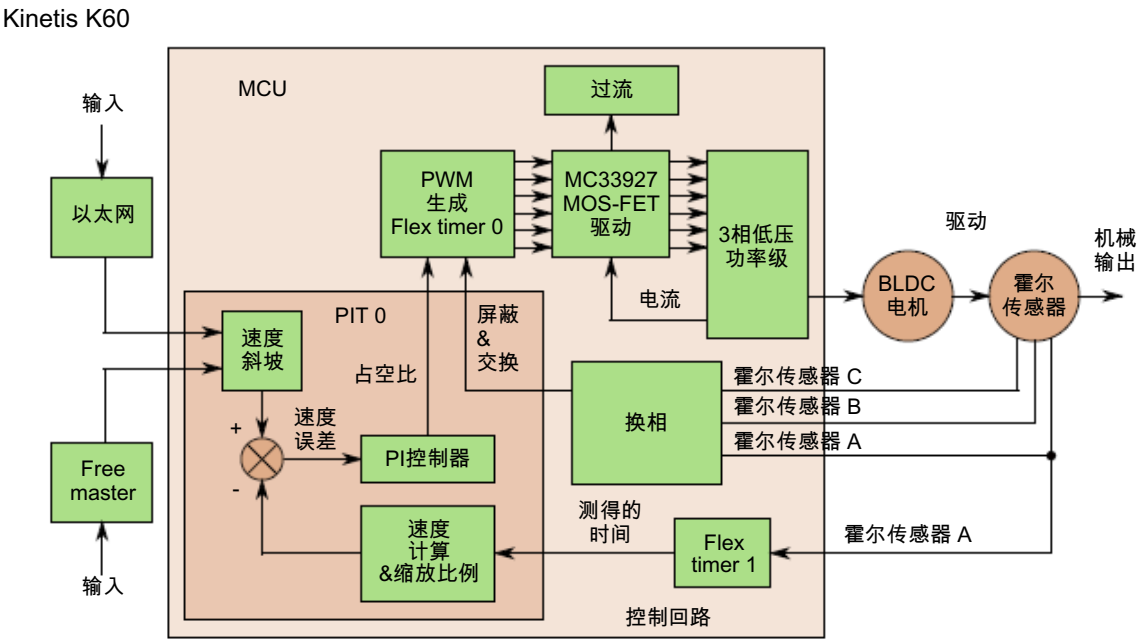

图 9. 应用框图

4.3 外设

附注

为了使此应用正确工作，必须使用以下外设。不允许将这些外设用于其他目的。

1. FTM0
   - 用于生成 PWM 信号
   - 在组合模式下运行
   - 开关频率为 19.2 kHz （48 MHz 的内核时钟）
   - 死区时间为 1 µs
2. FTM1
   - 用于速度测量
   - 在输入捕捉模式下运行
   - 预分频为 128
   - 模数为 0xFFFF
   - 溢出周期为 175 ms
3. PIT 0
   - 用于定时调用速度控制循环和应用状态机
   - 中断周期为 10 ms （48 MHz 内核时钟）
4. 端口 A

- 用于霍尔效应传感器中断
- 如果在此端口上施加任何其他信号，则在每个信号边沿调用此中断，程序不会正确运行

5. 端口 D
   - 用于霍尔效应传感器中断
   - 如果在此端口上施加任何其他信号，则在每个信号边沿调用此中断，程序不会正确运行
6. PTE26
   - 用于紧急停止按钮
7. PTA27
   - 用于读取 MC33937 MOS-FET 前置驱动的过流引脚
8. PTA10
   - 用于表示 MC33937 MOS-FET 前置驱动上的第一级过流
9. SPI 2
   - 用于与 MC33937 MOS-FET 前置驱动通信

4.4 数据流程图

驱动要求表明软件将提取用户接口和传感器中的部分数值进行处理，然后生成用于电机控制的三相 PWM 信号。闭环 BLDC 驱动的控制算法如 图 10 所述。

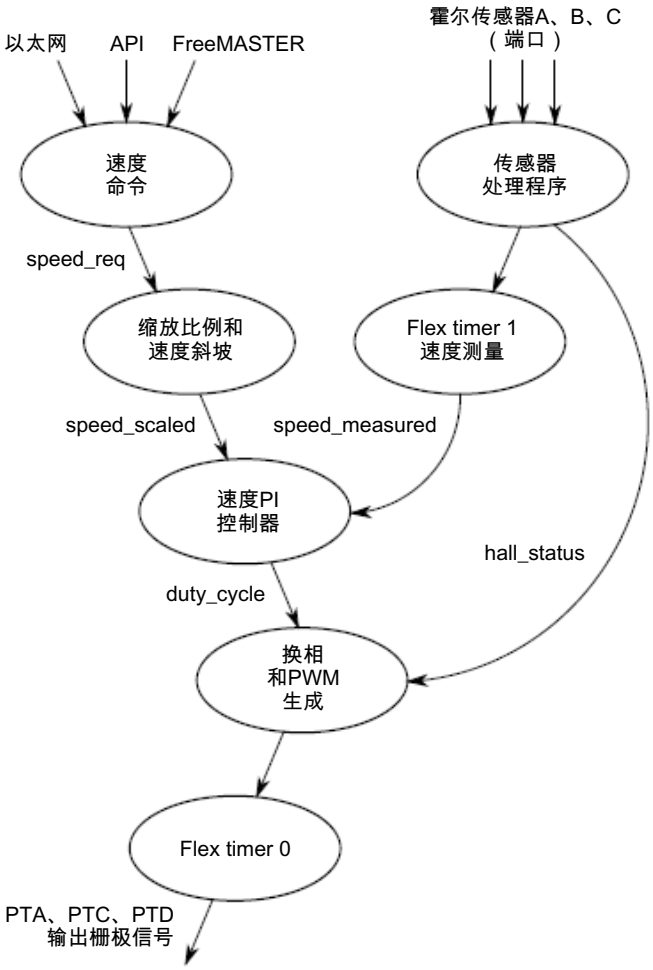

图 10. 数据流

数据流程图包含以下子章节中所述的流程：

- 速度命令
  每个速度更改命令均会使 speed_req 变量中的值改变。可以通过三种方式输入速度命令：
  — 通过以太网接口 （仅限 MQX 版本）
  — 通过 FreeMASTER 通信软件
  — 通过 API
- 缩放比例和速度斜坡
  提供 frac32 的比例更改和速度斜坡；请参见 5.3 “速度缩放”和 4.6 “速度斜坡”。运行于 PIT 中断处理程序中。
- 速度 PI 控制器
  它用于计算实际速度与所需速度之差并相应地补偿 PWM 模块的占空比；请参见 4.7 “PI 控制器”。
- 换相和 PWM 生成
  它用于根据霍尔传感器的信号创建旋转场。

- 传感器处理程序
  它用于生成换相向量。
- 速度测量
  请参见 4.5 “速度和位置测量”。

4.5 速度和位置测量

实际电机速度基于旋转周期 (*time_measured*) 计算得出，并与用户提供的 *speed_req* 进行对比。随后，通过速度斜坡算法对速度命令进行处理。从斜坡算法输出获得的实际速度命令与 *speed_measured* 进行比较，将生成一个 *speed_error*。

旋转周期通过霍尔传感器 A 和 flex timer 1 扫描，可在捕捉模式下对其进行配置。

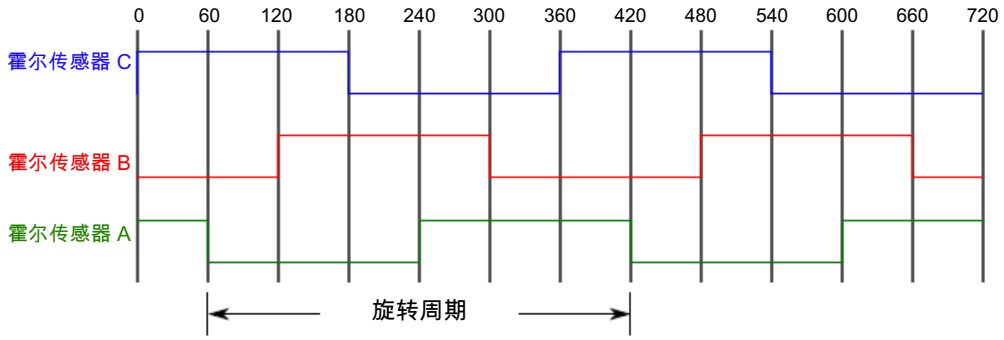

图 11. 用于速度测量的霍尔传感器信号

4.6 速度斜坡

由于整个应用系统具有很大惯性，因此，应用时必须细化速度命令，否则可能会导致系统过载。其中一种方法即为生成斜坡。此斜坡实施实际速度与速度命令之间的步骤。

在此应用中，所使用的斜坡来自库 GFLIB。此斜坡需要三个参数：

- s32RampUp
- s32RampDown
- s32State

前两个参数为斜坡的上下角度。最后一个参数为所需速度。这三个参数均为 32 位小数数据格式。

可以将 *variables_init.h* 中的前两个参数输入为以 [r/s^2] 为单位的整数，但它们将在应用初始化时重新计算为 frac32。参数名称为 *speed_ramp_down* 和 *speed_ramp_up*。

速度斜坡在PIT中断处理程序中调用。斜坡程序执行很大程度上取决于PIT周期。默认值为10 ms。如果 PIT 周期发生改变，则必须在 MAX_SCALED_SPEED_INV 宏中更改 PIT 周期。此宏用于将 *speed_ramp_down* 和 *speed_ramp_up* 从 int 重新计算回 frac32。

第二种输入速度斜坡参数的方法为使用 FreeMASTER。在 FreeMASTER 中，可在程序运行时将数据输入为以 [r/s^2] 为单位的整数。

详情请参见文档中章节 8 “参考文献”4。

4.7 PI 控制器

速度 PI 控制算法用于处理 *speed_req* 和 *speed_measured* 之间的 *speed_error*。PI 控制器输出传输至 PWM 发生器，以作为所施加电机电压的新校正值。

PI 控制器程序在 PIT 设备中断程序 PIT0_isr 中计算，它每 10 ms 被调用一次。此中断将在电机停止时禁用，从而使 PI 也禁用。 PI 控制器的积分部分在低速 （低于 499 RPM）时禁用，因为在此情况下，速度测量并不准确且 PI 控制器可能不稳定。要确定何时禁用，程序中存在两个宏：MIN_CW_SPEED_32 和 MIN_CWW_SPEED_32。

PI 控制器的输入为斜坡算法 *speed_scaled* 的输出，另一个输入为实际 *speed_measured*。另外两个输入为 PI 控制器参数 *trMyPI* 结构的指针。所有这些参数均供 PI 控制器函数 GFLIB_ControllerPIp 使用。

此函数输出为 *s32Output*。它将按照 PWM 比例缩放为 *delta_duty* 并添加到 *half_duty*。此处理结果为 *duty_cycle*，这将被加载至 Flex Timer 寄存器中。

每当速度比例更改或者电机改变时，必须配置 PI 速度控制器参数。

详情请参见文档中章节 8 “参考文献” 4。

5 软件实现

5.1 软件中的函数实现

整个电机控制算法由中断驱动。主函数仅用于MCU和应用初始化；请参见图 12。初始化终止时，程序将进入无限循环或其他应用处理程序 （web 服务器、 USB、 FreeMASTER 等）。

附注

此布局对于其他应用的正常工作非常重要。使用此布局时，可以通过 MQX 或裸机轻松实现。

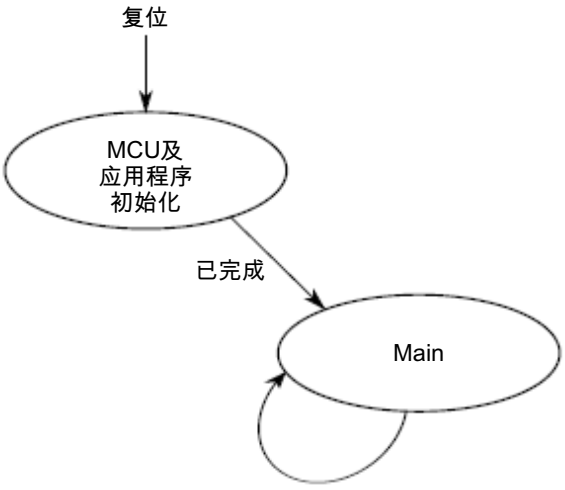

图 12. 软件中主函数的实现

为了使此电机应用正常工作，需要使用四个中断。

此中断处理程序提供以下服务：

- 溢出中断处理程序
  用于电机停止检测和速度测量。溢出处理程序用于复位速度斜坡。
- 输入捕捉中断处理程序
  用于读取两个霍尔传感器边沿之间的时间 （处理速度传感器的基本部分）。
- 周期中断
  用于定时调用速度控制器、速度斜坡和应用状态机。 PIT 中断在电机停止时禁用。
- 霍尔传感器中断
  用于扫描霍尔传感器的状态以及换相过程。换相过程生成适当的换相模式到六个栅极信号，而 PWM 发生过程则为选定栅极输出生成相应的 PWM 信号。可使用三位换相向量“hall_status”从换相表中选择换相模式。换相模式随后将载入微控制器的寄存器中。

在图 13 中，您可以查看这些中断在软件中是如何实现的。

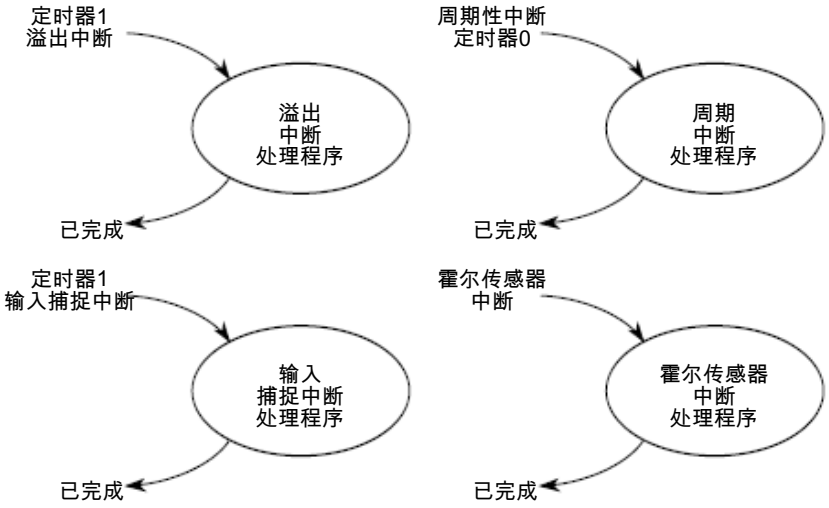

图 13. 电机控制应用中的中断

5.2 中断安装

裸机版本与 MQX 版本的差别仅在于中断安装方法。所用的方法取决于是否在 IAR 项目选项（选项 / C/C++ 编译器 / 预处理器 / 定义的符号）中预定义 MQX 或 Bare_Metal。

在裸机版本中，我们可以直接安装中断。您可以通过在 NVICISER 寄存器中设置正确位来轻松实现中断。可以使用寄存器 NVIC_IP 配置中断的优先级。为了更好地理解，请看以下示例。

中断安装：

```
NVICICPR2  = 0x4800010;   // 首先清除可能的挂起中断
NVICISER2  = 0x4800010;   // 启用中断
NVICICPR1 |= 0x80000000;  // 首先清除可能的挂起中断
NVICISER1 |= 0x80000000;  // 启用中断
```

设置中断的优先级：

```
NVIC_IP(63) = 0x40;    // 针对 FTM1 设置优先级
NVIC_IP(68) = 0x50;    // 针对 PIT0 设置优先级
NVIC_IP(87) = 0x40;    // 针对霍尔传感器设置优先级
NVIC_IP(90) = 0x40;    // 针对霍尔传感器设置优先级
```

如果在 MQX 中安装的中断与裸机版本中相同，则 MQX 不会正常工作。对于电机控制应用来说，MQX 自带的中断非常慢。高速应用的最佳解决方案为内核中断。内核中断为自然的 CPU 中断，无 MQX 开销和最小执行持续时间限制。内核中断的劣势是不支持诸如事件或信号量等 MQX 功能。为了更好地理解如何使用内核中断，请看以下示例。

中断安装：

```
_int_install_kernel_isr(INT_PIT0,PIT0_isr);
_int_install_kernel_isr(INT_FTM1,FTM1_isr);
_int_install_kernel_isr(INT_PORTA,Hall_Status_isr);
_int_install_kernel_isr(INT_PORTD,Hall_Status_isr);
```

设置中断的优先级：

```
_bsp_int_init((IRQInterruptIndex)INT_PIT0, 1, 0, 1);
_bsp_int_init((IRQInterruptIndex)INT_FTM1, 1, 0, 1);
_bsp_int_init((IRQInterruptIndex)INT_PORTA, 1, 0, 1);
_bsp_int_init((IRQInterruptIndex)INT_PORTD, 1, 0, 1);
```

5.3 速度缩放

本应用使用小数表示除时间以外的所有实数。 N 位带符号的小数格式用 1. [N-1] 格式 （1 个符号位， N-1 个小数位）。带符号的小数 (SF) 数值范围如下所示：

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

对于字和长字带符号小数，最小负数可以表示为 –1.0，其内部表示分别为 $8000 和 $80000000。对于字，最大正数为 $7FFF 或 $1.0 - 2^{-15}$，对于长字，最大正数为 $7FFFFFFF 或 $1.0 - 2^{-31}$。以下等式所示为实际表示与小数表示之间的关系：

等式 5

$$\text{Fractional Value} = \frac{\text{Real Value}}{\text{Real Quatity Range}}$$

如果为速度缩放：

小数值 = 速度量 [–] 的小数表示

实际值 = 实际速度量 （物理单位为 [rpm]）

实数范围 = 用于缩放的最大定义速度 （物理单位为 [rpm]）

在此应用中，缩放比例常数按以下等式计算：

等式 6

$$\text{Scale Constant} = \frac{1}{\text{MAX\_SCALED\_SPEED}} x \frac{30}{PP\frac{TPM\_P}{TPM\_C}} = \frac{1}{5000} x \frac{30}{2\frac{128}{48000000}} = 1125$$

在此应用中， SCALE_CONST 宏用于计算缩放比例：

SCALE_CONST (tFrac32)((30 / (PP*(TPM_P/TPM_C))) / MAX_SCALED_SPEED)

PP — 所使用的电机极对数。默认值为 2。

TPM_C— 定时器的输入时钟 (Hz)。默认值为 48e6。

TPM_P— 定时器输入时钟的预分频。默认值为 128。

MAX_SCALED_SPEED — 具有裕量时的最大速度。默认值为 5000。

如您在等式 6 中所见，计算的此比例常数仅适合于半个周期。这是为了更好地使用定时器。*Time_measured* 将被右移 （除以 2）以仅包含正数。将定时器模数设为 0xFFFF 并将预分频设为 128。在此情况下，该定时器的溢出周期为 174.8 ms，因此，我们可以检测到的最小速度为 171.6 rpm。

最后，按以下等式计算 *speed_measured*：

等式 7

$$\text{measured\_speed} = \frac{1}{\frac{\text{revolution\_period}}{2}} \cdot \text{scaling\_constant}$$

speed_measured = F32DivF32F16((SCALE_CONST), (time_measured >> 1));

详情请参见文档中章节 8 “参考文献” 6。

5.4 更改速度缩放比例

缩放常数通过宏计算，因此，仅当编译程序时才可计算该常数。您可以根据所用的电机在文件 *variables_init.h* 中更改缩放常数参数，但您必须重新编译程序。

可以在文件 *variables_init.h* 中设置这些参数：

PP — 所使用的电机极对数。此数值表示电机电气旋转和机械旋转之间的比率。默认值为 2。此参数可在电机数据手册中找到。

TPM_C— 定时器的输入时钟 (Hz)。默认值为 48e6。

TPM_P— 定时器输入时钟的预分频。默认值为 128。

MAX_SCALED_SPEED — 具有裕量时的最大速度。裕量为 20%。此参数必须大于电机的标称速度。

附注

每当速度比例更改或者电机改变时，必须配置 PI 速度控制器参数。

5.5 将驱动集成到其他应用中

两种版本的电机控制驱动均可作为其他应用的一部分。如果在其他应用中使用此应用，则必须在项目设置中定义 MQX 或 Bare_Metal。可以通过以下三个函数定义 API：

1. `void Set_speed(signed short, int)`

   第一个参数为带符号短整型数据格式的输入速度。输入值以 RPM 为单位。

   第二个参数为接收命令的电机序号。本演示仅有一个电机，因此输入 1。

   返回：void

```
/************************************************************
* 函数名称：Set_speed
* 参数：所需速度、电机序号
* 返回：测得的速度
* 描述：用于为每个电机输入所需速度
************************************************************/
void Set_speed(signed short speed_input, int motor_number)
{ speed_req = INT16TOF32((speed_input*SPEED_TO_RPM_SCALE)); }
```

2. `unsigned char Get_status(void)`

   返回：应用的状态

   0 — 处于空闲状态

1 — 处于停止状态
2 — 处于正在运行状态

```
/*****************************************************
* 函数名称：Get_speed
* 参数：电机序号
* 返回：测得的速度
* 描述：用于获取每个电机的速度
*****************************************************/
signed short Get_speed(int motor_number)
{
 signed long speed_temp;
 speed_temp = (speed_measured * MAX_SCALED_SPEED);
 return (speed_temp >> 15);
}
```

3. `signed short Get_speed(int)`

第一个参数为接收命令的电机序号。本演示仅有一个电机，因此输入 1。
返回：测得的速度 （带符号短整型数据格式）。此值以 RPM 为单位。

```
/*****************************************************
* 函数名称：Get_status
* 参数：无
* 返回：应用状态
*****************************************************/
unsigned char Get_status(void)
{ return App_state; }
```

6 应用配置

此应用配置的详情请参见文档中章节 8 “参考文献” 5。

7 定义和首字母缩略词

AN	应用笔记
API	应用程序接口
BLDC	无刷直流电机
DC	直流
MQX™	Freescale MQX 实时操作系统
节拍	操作系统时间单位 （也反映最小时间分辨率）
POSIX	便携式操作系统接口，由 IEEE 开发并由 ANSI 和 ISO 标准化。 MQX 符合 POSIX.4 （实时扩展）和 POSIX.4a （线程扩展）。
RTOS	实时操作系统
RTCS	嵌入式互联网协议栈为 MQX 平台提供 IP 网络。 RTCS 具有种类丰富的 TCP/IP 网络应用协议，并使用 MQX RTOS 驱动程序来实现以太网或串行连接。

8 参考文献

1. Freescale document DRM022, 3-*Phase BLDC Drive Control with Hall Sensors*, Jiri Ryba and Petr Stekl, 2003
2. Freescale document DRM117, *3-Phase Sensorless BLDC Motor Control Using MC9S08MP16,* Libor Prokop, 2009
3. Freescale document AN4254, *Motor Control Under the Freescale MQX Operating System,* Libor Prokop, 2011
4. Freescale document MCLIBCORETXM4UG, *Set of General Math and Motor Control Functions for Cortex M4 Core,* Jaroslav Musil and Pavel Rech, 2011
5. Freescale document BLDCK60UG, *3-Phase BLDC Motor Control on Kinetis,* Ivan Lovas, 2011
6. Freescale document AN1930, *3-Phase AC Induction Motor Vector Control Using a 56F80x, 56F8100 or 56F8300 Device,* Jaroslav Lepka and Petr Stekl, Freescale Semiconductors, 2005

此页有意留空

How to Reach Us:

Home Page:
freescale.com

Web Support:
freescale.com/support

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

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

Document Number: AN4376
Rev. 0, 10/2011

freescale™
semiconductor