Векторное управление с библиотекой MC SDK от STMicroelectronics

Валентин Юрзин (yuvalmid@rambler.ru)

В статье рассматривается реализация векторного управления с помощью комплекта разработки STM32 MC SDK программного обеспечения управления синхронным двигателем MotorControl Workbench 5.3.3 компании STMicroelectronics. Особое внимание уделяется поле-ориентированному управлению и его важнейшему компоненту – наблюдателю, цифровому вычислителю координат вращения тока статора.

Краткий обзор STM32 MC SDK

Микроконтроллеры STM32 обеспечивают высокую производительность для стандартных ядер ARM Cortex M, что позволяет использовать их в управлении приводами трёхфазного напряжения, работающими в режимах векторного (VC) или поле-ориентированного управления (FOC) [1].

Комплект разработки программного обеспечения (SDK) для управления двигателем включает в себя:

- библиотеку программ ST MC FOC для управления синхронным двигателем с постоянными магнитами в роторе;
- среду разработки (IDE) с графическим интерфейсом MotorControl Workbench;
- ST Motor Profiler (MP) – инструмент определения параметров двигателя, подключённого к типовой плате STM32.

Данный комплект позволяет сократить время проектирования и трудозатраты при создании приложений для управления двигателем. Через графический интерфейс комплекта разработки пользователь может генерировать все файлы, необходимые для приложения, а также отслеживать и корректировать некоторые переменные. В библиотеке реализована поддержка интерфейса UART, который позволяет удобно настраивать систему управления двигателем. Проект может быть выполнен в нескольких популярных IDE:

- IAR Embedded Workbench® для ARM® версии 7.80.4 или выше;
- Keil® MDK версии 5.24.2 или выше;
- TrueSTUDIO for STM32.

Для установки и настройки программного обеспечения следует обратиться к руководству пользователя [2]. На момент написания статьи доступна последняя версия MotorControl Workbench 5.3.3.

Библиотека ST MC FOC позволяет организовать поле-ориентированное управление трёхфазного двигателя с постоянными магнитами в роторе PMSM, SM-PMSM (ротор расположен поверх статора) и I-PMSM (ротор расположен внутри статора). Библиотека STM32 FOC PMSM SDK построена на принципах объектно-ориентированного программирования (ООП), которые обеспечивают создание объектов, классов на языке С и задач управления двигателем.

Благодаря использованию ООП улучшаются переносимость и эффективность кода, возможность доступа к определённым аппаратным адресам, снижаются требования к системным ресурсам. Язык программирования С широко используются во встроенных системных приложениях, но, в отличие от более сложных языков, таких как C++ и Java, не является объектно-ориентированным языком программирования. В связи с этим компанией STMicroelectronics принципы ООП для языка С были реализованы в библиотеке STM32 PMSM FOC FW [3].

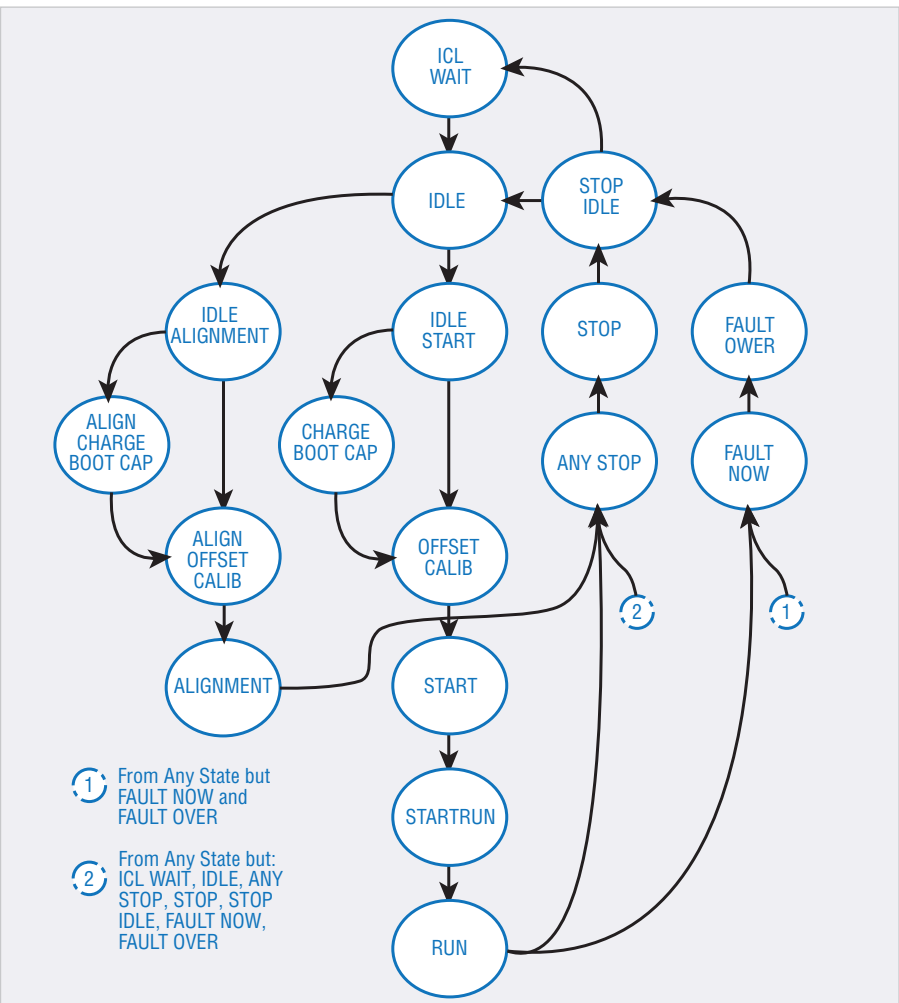

Рис. 1. Автомат управления двигателем

Управление состоянием двигателя

Подсистема микропрограммы MC поддерживает автомат управления для каждого из двух двигателей [4]. На рисунке 1 приведена структурная схема автомата управления двигателем. В синих кружках показаны состояния, стрелками – возможные переходы между ними. Некоторые API, вызываемые из приложения, приводят к изменению состояния:

- MC_StartMotor1(), MC_StartMotor2(): запуск двигателя – переключение с IDLE в IDLE_START;
- MC_StopMotor1(), MC_StopMotor2(): остановка двигателя – переключение в ANY_STOP;
- AcknowledgeFaultMotor1(), AcknowledgeFaultMotor2(): подтверждение ошибки и готовность двигателя к запуску – переключение в STOP_IDLE.

Управление двигателями и изменение их состояний осуществляется с помощью функций TSK_MediumFrequencyTaskM1() и TSK_MediumFrequencyTaskM2(). В таблице представлены описания состояний автомата.

Состояния автомата управления двигателем

Состояние	Описание
ICL WAIT	Ожидание ограничителя пускового тока
IDLE	Двигатель не вращается, но готов к запуску или выравниванию энкодера
IDLE ALIGNMENT	Переходное состояние после команды выравнивания
ALIGN CHARGE BOOT CAP	Зарядка запускающих конденсаторов инвертора перед калибровкой
ALIGN OFFSET CALIB	Калибровка смещения токов перед выравниванием энкодера
ALIGNMENT	Энкодер правильно выровнен
IDLE START	Проходное состояние после команды старта двигателя
CHARGE BOOT CAP	Зарядка запускающих конденсаторов инвертора
OFFSET CALIB	Калибровка смещения токов
START	Состояние начала процедуры запуска
START RUN	Переходное состояние перед запуском двигателя
RUN	Работа двигателя
ANY STOP	Переходное состояние перед остановкой двигателя
STOP	Переходное состояние после остановки двигателя
STOP IDLE	Переходное состояние между STOP и IDLE
FAULT NOW	Состояние ошибки, переход к состоянию FAULT OVER
FAULT OVER	Состояние ошибки, ожидание подтверждения пользователем или сброс ошибки и переход к STOP IDLE

API управления двигателями

API управления двигателем является основным интерфейсом в библиотеке управления двумя двигателями с помощью одного STM32 MCU. API предлагает один набор функций для каждого из двигателей. В функции определён дескриптор, указывающий на номер двигателя, которому она предназначена.

Основными задачами API являются запуск, остановка и контроль вращения двигателей. Контроль вращения двигателя достигается путём программирования тока, крутящего момента или задания скорости с помощью пропорционально-интегральных регуляторов (ПИД-регуляторов) подсистемы управления двигателем. Эти параметры должны быть установлены до запуска двигателя. Значения крутящего момента или скорости программируются как линейные изменения от текущих до заданных показателей. Запрограммированное задание или линейное изменение выполняется сразу, если двигатель вращается и находится в устойчивом состоянии (машина находится в состоянии RUN). В противном случае команда буферизуется до тех пор, пока автомат двигателя не перейдёт в состояние RUN. Одновременно может быть запрограммирован только один темп, т.е. последний темп заменяет предыдущий. Предпочтительным методом для привода двигателей является метод управления скоростью или крутящим моментом. Контроль крутящего момента является режимом управления по умолчанию. Также в API есть функции для получения значений различных параметров и переменных состояния подсистемы MC. Дескрипторы, через которые можно получить доступ к параметрам MC, определены в файлах *mc_config.c* и *mc_config.h*.

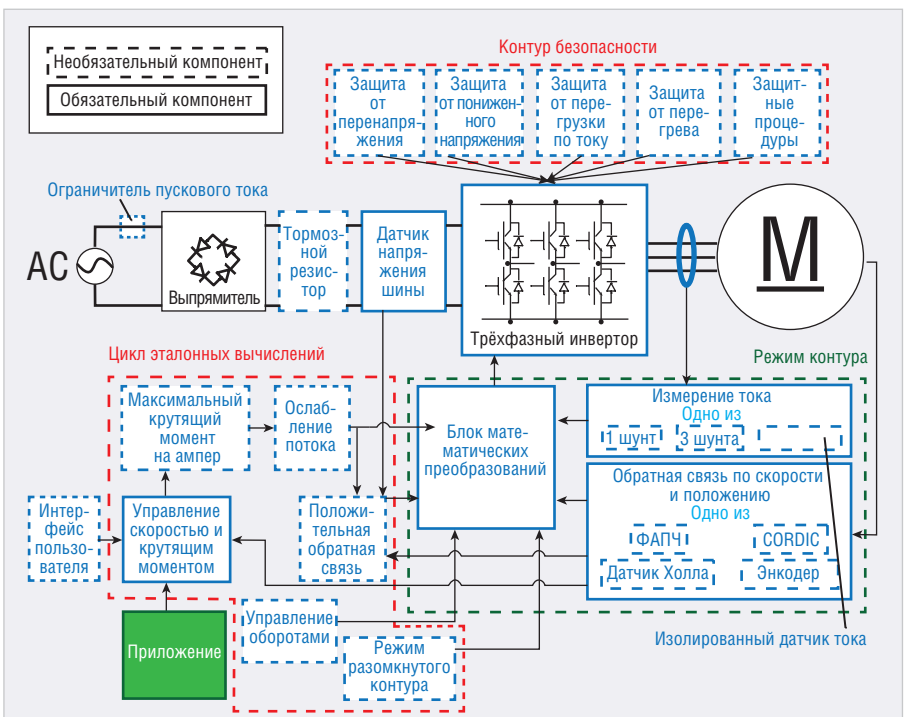

Рис. 2. Подсистема управления двигателем

Подсистема управления двигателем

Подсистема управления двигателем – это библиотека микропрограмм, которая создаётся в результате конфигурации и генерирования проекта в MotorControl Workbench 5.3.3. Затем пользователи создают свой собственный код с помощью функций интерфейса API. Подсистема программного обеспечения управления двигателем показана на рисунке 2. На схеме представлены основные и дополнительные функциональные блоки, а также отражено их взаимодействие. Выделяются три основных функциональных блока.

Блок цикла FOC является ядром алгоритма FOC. Его цель – рассчитать фазные напряжения и получить рабочие циклы широтно-импульсной модуляции (ШИМ) для управления транзисторами инвертора. Блок выполняет все математические преобразования, необходимые для перехода от измеренных фазных токов I_a, I_b, I_c к токам I_q, I_d и затем обратно – от напряжения U_q, U_d к созданию рабочих циклов ШИМ. Значения I_q, I_d, сопоставленные со значения-

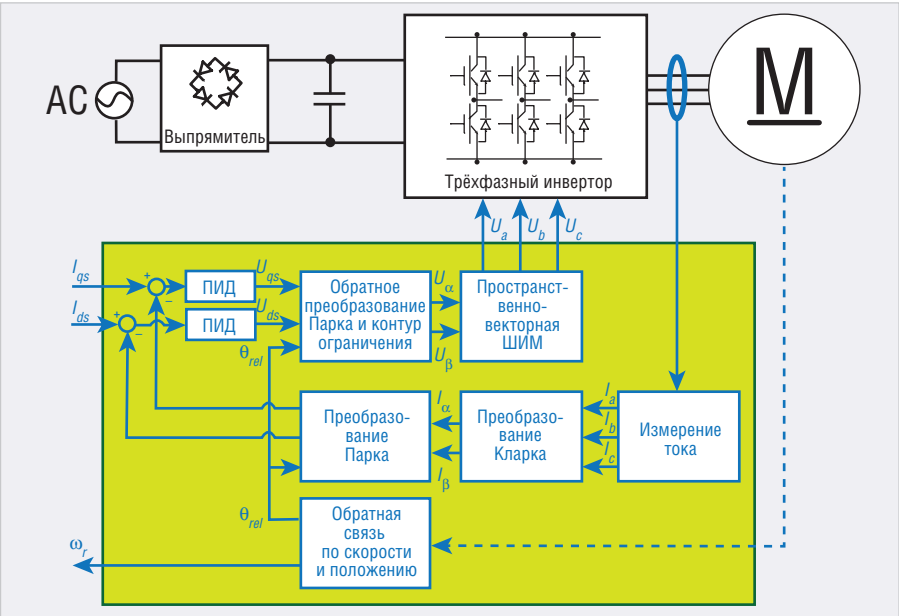

Рис. 3. Базовая структура алгоритма FOC

ми U_q, U_d с помощью ПИД-регуляторов, преобразуются в U_a, U_b.

Цикл FOC реализован в функции TSK_HighFrequencyTask(), которая выполняется на частоте ШИМ (один раз в каждый период ШИМ). Частота ШИМ является самой высокой частотой в подсистеме управления двигателем. Функция вызывается в обработчике прерывания, которое возникает, когда периферийные АЦП используются для захвата значения тока фазы. Основной задачей этой функции является вычисление рабочих циклов ШИМ, которые должны быть запрограммированы в каналах таймера TM1. Следовательно, время работы этой функции ограничено, т.к. она должна завершиться до следующего события обновления таймера, когда будут создаваться новые рабочие циклы ШИМ, – в противном случае вычисления приводят к ошибке алгоритма FOC.

Блок цикла эталонных вычислений отвечает за вычисление значений I_q^*, I_d^* на основе команд, поступающих из приложения, которое посылает заданные значения скорости, крутящего момента или постепенного нарастания скорости до заданного значения. Цикл вычисления сначала преобразует задание приложения в исходные значения I_q^*, I_d^*, которые затем могут пропускаться через один или несколько алгоритмов оптимизации привода двигателя: оптимизация максимального крутящего момента на ампер (MTPA) или ослабление потока. Затем результирующие значения I_q^*, I_d^* используются в цикле FOC. Этот процесс выполняется, когда подсистема управления двигателем работает в режиме замкнутого контура (статус RUN).

В зависимости от выбранной скорости и состояния привода может потребоваться фаза настройки, которая продолжается до тех пор, пока оценка программой положения ротора не станет надёжной. Кроме того, в некоторых случаях может потребоваться, чтобы управление двигателем оставалось в разомкнутом контуре, например в процессе разгона. Этот случай обрабатывается компонентом Open Loop Control, который выполняется вместо нормального процесса регулирования FOC.

Циклы управления реализуются в двух функциях, по одной на мотор: TSK_MediumFrequencyTaskM1(), TSK_MediumFrequencyTaskM2() для двигателей 1 и 2 соответственно. Эти функции должны вызываться периодически с частотой, как правило, ниже, чем TSK_HighFrequencyTask(). В подсистеме управления MC функции вызываются по прерыванию SysTick.

Последний набор функциональных блоков – это контур безопасности. Все действия в данном цикле направлены на реагирование на события, которые могут поставить под угрозу аппаратную часть системы: перенапряжение или понижение напряжения, перегрев, перегрузка. Для защиты от перегрузки по току микропрограмма STM32 MC использует аппаратные механизмы, реализованные в микроконтроллерах STM32, такие как вход Timer Break – аппаратный вход для отключения таймера TM1. Эти механизмы ускоряют реакцию системы на сложившуюся ситуацию. Цикл безопасности выполняется с той же скоростью, что и цикл эталонных вычислений, со средней скоростью по прерыванию SysTick.

Цикл безопасности реализован в функции TSK_SafetyTask(), которая вызывается из TSK_SafetyTask_PWMOFF(), TSK_SafetyTask_RBRK() или TSK_SafetyTask_LSON(), в зависимости от выбранного метода защиты по перенапряжению.

Управление с ориентацией по полю FOC

Библиотека программного обеспечения PMSM FOC предлагает производительный алгоритм FOC для управления синхронным двигателем. При таком подходе можно создавать электромагнитный момент регулирования и, в некоторой степени, ослаблять поле, контролируя два компонента тока, которые являются математическим преобразованием токов статора. Это напоминает управление двигателем постоянного тока, где ток якоря взаимодействует с токами полюсов возбуждения двигателя, позволяя хорошо управлять скоростью и моментом.

Таким образом, можно сказать, что принцип FOC состоит в управлении и ориентации по фазе тока статора, который взаимодействует с потоком ротора. Из этого определения становится ясно, что необходимы средства измерения токов статора и угла ротора.

На рисунке 3 представлена базовая структура алгоритма FOC управления скоростью или крутящим моментом. Значения I_{qs}^* и I_{ds}^* – идентификаторы значений электромагнитного момента и тока. Алгоритм состоит из следующих блоков:

- блок пространственно-векторной ШИМ (SVPWM) реализует усовершенствованный способ модуляции, который уменьшает гармоники тока и оптимизирует эксплуатацию шины постоянного тока;
- блок измерения тока (Current Reading) позволяет системе корректно рассчитывать ток статора, используя резисторы шунта, датчики Холла или изолированные датчики тока;
- блок обратной связи по скорости и положению ротора (Rotor Speed & Position Feedback) позволяет системе обрабатывать датчик Холла или инкрементные сигналы энкодера для распознавания углового перемещения ротора;
- блоки ПИД-регуляторов осуществляют вычисления обратной связи (регулирование тока);

- блоки Clarke, Park, Reverse Park и Circle реализуют математические преобразования, требуемые для FOC.

Библиотека встроенного программного обеспечения MC оптимизирована для машин SM-PMSM и I-PMSM. Регулируя токи двигателя через их компоненты I_{qs} и I_{d}, FOC управляет крутящим моментом и током.

Бессенсорный алгоритм наблюдателя CORDIC

Измерения положения и скорости ротора играют решающую роль в FOC-управлении. Бессенсорные алгоритмы определения положения и скорости находят широкое применение по причине их невысокой стоимости. Библиотека микропрограмм MC предлагает эффективное решение для обнаружения положения без датчика обратной связи по положению (например, энкодера), которое основано на теории наблюдателя состояния. В библиотеке реализован алгоритм, применимый к синхронным двигателям SM-PMSM и I-PMSM [5].

Наблюдатель состояния в теории управления представляет собой систему, которая обеспечивает оценку внутреннего состояния системы с учётом её входных и выходных параметров. В данном случае внутренними состояниями двигателя являются обратные ЭДС и фазные токи. Наблюдаемые состояния анализируются на предмет соответствия реальной системе через фазу тока, а результат измерений используется для настройки модели через вектор усиления (K_1, K_2). Обратные ЭДС двигателя определяются следующим образом:

$$e_{\alpha}=\Phi_m \rho\omega_r \times \cos(\rho\omega_r \times t),$$
$$e_{\beta}=-\Phi_m \rho\omega_r \times \sin(\rho\omega_r \times t).$$

Как видно, они содержат информацию об угле ротора. Обратная ЭДС передаётся в блок, который восстанавливает электрический угол и скорость вращения ротора. Данный блок может быть системой фазовой автоподстройки частоты (ФАПЧ) или цифровым вычислителем координат вращения (CORDIC), в зависимости от выбора пользователя. Кроме того, модуль обрабатывает выходные данные и тем самым реализует функцию безопасности, которая обнаруживает состояние заблокированного ротора или его неисправность.

На рисунке 4 показана общая блок-схема захвата полезного сигнала во время работы двигателя с FOC-управлением. На рисунке 5 жёлтым и красным цветом показаны синусоиды обратных ЭДС e_{α} и e_{β}. Осциллограмма

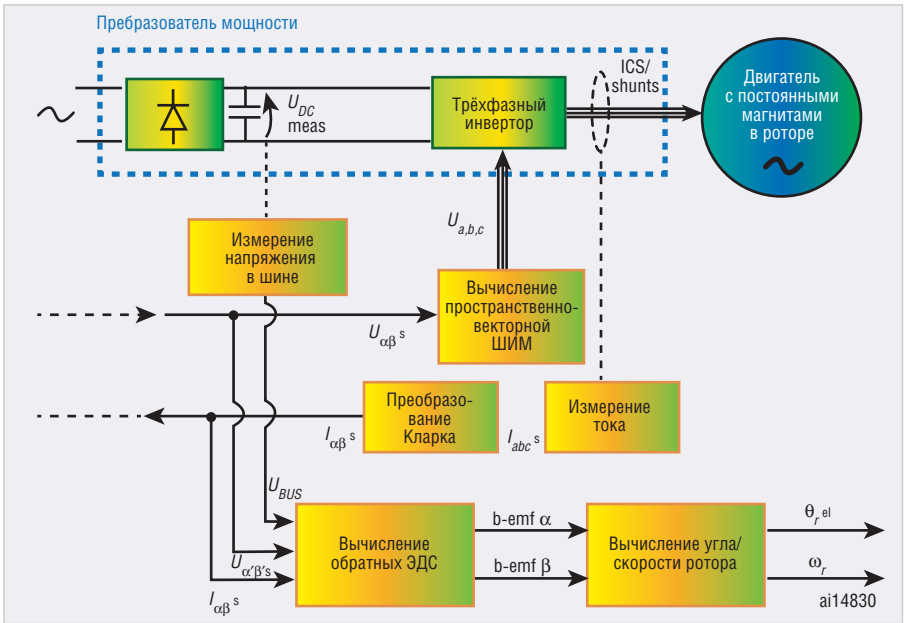

Рис. 4. Общая блок-схема алгоритма без датчика

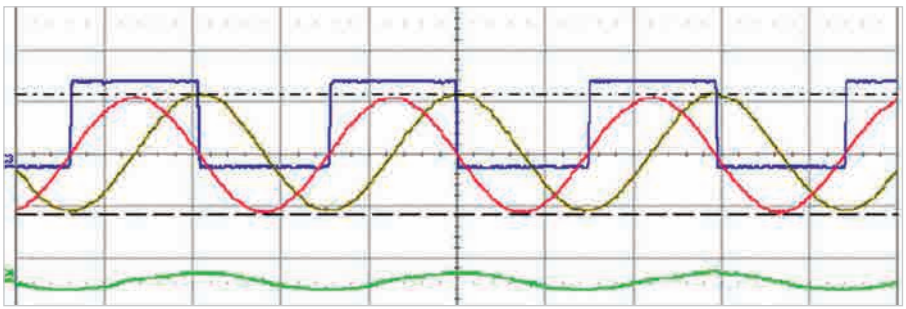

Рис. 5. Обратные ЭДС, которые обнаруживает алгоритм наблюдателя без датчика

синего цвета – сигнал от датчика Холла. Зелёная синусоида – измеренный ток.

Важными для расчёта значений модели двигателя являются следующие его параметры: сопротивление обмотки R_s, индуктивность обмотки L_s, время выборки алгоритма без датчика T, которое совпадает с выборкой FOC и токов статора, и число пар полюсов, влияющее на конечную скорость двигателя. Собственные значения модели могут быть рассчитаны как:

$$e_1=1-(R_s \times T/L_s),$$
$$e_2=1.$$

Собственные значения наблюдателя:

$$e_{1obs}=e_1/f,$$
$$e_{2obs}=e_2/f.$$

Как правило, f=4. Начальные значения K_1 и K_2 могут быть рассчитаны как:

$$K_1=(e_{1obs}+e_{2obs}-2/T)-R_s/L_s,$$
$$K_2=(L_s \times (1-e_{1obs}-e_{2obs}+e_{1obs} \times e_{2obs}))/T^2.$$

За правильность расчёта отвечает графический интерфейс ST MC. Эти значения можно также изменить после генерации проекта в файле *drive_parameters.h*. Классы, которые реализованы в алгоритме без датчиков, находятся в библиотеке как скомпилированные объектные файлы (файлы с расширением **.lib*). Их содержимое невозможно посмотреть или изменить.

На низких оборотах, от 0 до 250 об./мин (0–8 Гц для двигателей с номинальной частотой 50 Гц), величины обратных ЭДС e_{α} и e_{β} недостаточны для захвата наблюдателем и, соответственно, для определения положения ротора. Поэтому необходимо разогнать двигатель, используя виртуальный сенсор (VIRTUAL) в системе управления, до скорости более 250 об./мин, когда генерация обратных ЭДС становится достаточной для определения положения ротора. В функциях автомата управления двигателем отслеживается надёжность работы наблюдателя, и, как только расчётные данные скорости становятся стабильны, происходит переключение с виртуального сенсора (режим разомкнутого контура регулирования) на CORDIC (режим замкнутого контура регулирования), а состояние автомата изменяется на START_RUN (проходное состояние) и последующее RUN (стойкое состояние – двигатель запущен).

Проиллюстрируем работу сенсоров Virtual и CORDIC с помощью осцилло-

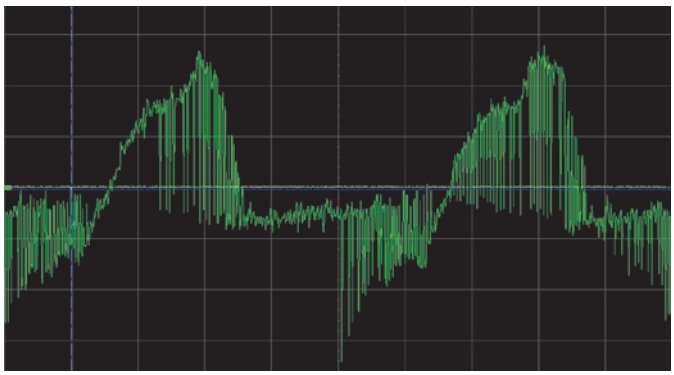

Рис. 6. Выход датчика тока фазы B (нагрузка – синхронный двигатель)

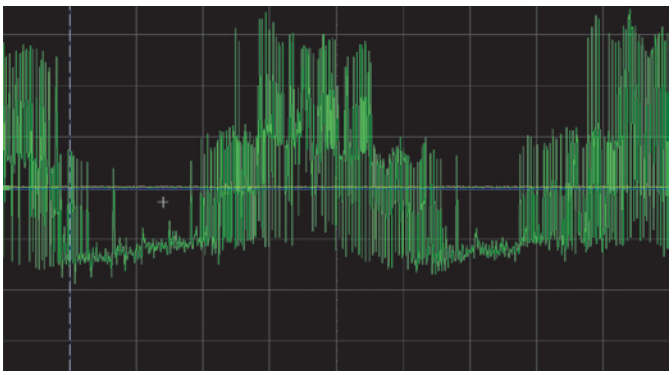

Рис. 7. Выход датчика тока фазы B (нагрузка – резисторы)

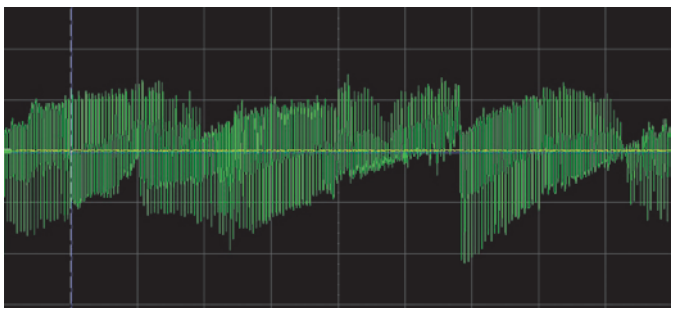

Рис. 8. Выход датчика тока фаз ABC (управление – Virtual-сенсор)

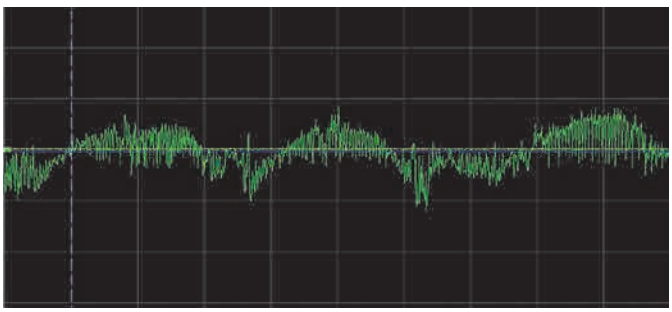

Рис. 9. Выход датчика тока фаз ABC (управление – CORDIC-сенсор)

грамм. Вид формы тока с датчика тока при нагрузке инвертора на резисторы и синхронный двигатель показан на рисунках 6 и 7. На рисунке 6 представлена осциллограмма фазы B датчика тока платы инвертора. Выход датчика тока не подключён к плате управления, т.е. регулирование от обратной связи по току отсутствует. Данный способ включения является экспериментальным и кратковременным, т.к. при отсутствии обратной связи по току возникают максимальное напряжение на выходе инвертора и максимальный момент. Программа опрашивает сенсор Virtual. Частота ШИМ-модуляции составляет 16 000 Гц. Нагрузкой инвертора является синхронный двигатель, который вращается со скоростью 300 об./мин. На осциллограмме наблюдаются значения ЭДС e_α и e_β, необходимые для захвата наблюдателями типа CORDIC или PLL.

На рисунке 7 показана осциллограмма фазы B с такими же параметрами управления, но нагрузкой в данном случае являются резисторы. В составляющей тока видна только ШИМ-модуляция. Наблюдатель не может захватить обратные ЭДС при данной нагрузке в инверторе.

Теперь подключим выход датчика тока инвертора ко входу платы управления. В программе управления двигателем в качестве сенсора управления установлен виртуальный датчик. Нагрузкой инвертора является синхронный двигатель, который вращается со скоростью 300 об./мин. На рисунке 8 показана осциллограмма датчика тока фаз ABC (конфигурация с одним шунтом) при управлении от виртуального датчика.

При таких же параметрах нагрузки изменим в программе управления двигателем сенсор управления на CORDIC – соответствующая осциллограмма представлена на рисунке 9.

В файлах проекта, генерируемых MotorControl Workbench 5.3.3, есть возможность добавлять пользовательский код. Для этого необходимо разместить свой фрагмент кода в окружении специальных комментариев:

```
/* USER CODE BEGIN XXX */
Код пользователя…
/* USER CODE END XXX */
```

Заключение

Библиотека STM MC обеспечивает эффективное управление скоростью и крутящим моментом. Замечательным свойством библиотеки является алгоритм измерения тока наблюдателем на одном шунте. При очень зашумлённом выходе трёх фаз ABC на один шунт датчика тока происходит уверенный захват полезного сигнала. Управление полем от наблюдателя CORDIC вызывает автоматическую коррекцию ШИМ фаз ABC и исключает потери энергии, но это происходит только при благоприятных условиях во время работы наблюдателей. При работе двигателя в реальных условиях возникают различные ситуации, когда, например, резко изменяется нагрузка, происходят скачки, просадки питающего напряжения, что вызывает сбой в работе наблюдателя и остановку привода в данной реализации. Алгоритмы обхода таких ситуаций в библиотеке не предусмотрены – более того, если программно разрешить работу во время сбоя захвата сигнала наблюдателем, то результат управления может быть непредсказуемым. Именно такие задачи по созданию мягких переходов от работы по управлению полем в неблагоприятных условиях к управлению виртуальным сенсором и обратно, когда условия для захвата наблюдателем становятся благоприятными, предстоит решать пользователю библиотеки STM32 MC SDK от STMicroelectronics.

Литература

1. https://www.st.com/resource/en/user_manual/dm00486148.pdf
2. https://www.st.com/resource/en/application_presentation/stm32_pmsm_foc_sdk_getting_started.pdf
3. https://www.st.com/resource/en/user_manual/cd00298482.pdf
4. https://www.st.com/resource/en/user_manual/dm00490980.pdf
5. https://www.st.com/resource/en/user_manual/cd00298474.pdf

ВНЕСЕНЫ В ГОСРЕЕСТР СИ

Ceyear

Высококлассные измерительные приборы микроволнового и миллиметрового диапазона без экспортных ограничений

Анализатор спектра серии 4024

- Диапазон частот от 9 кГц до 44 ГГц
- Полоса разрешения: 1 Гц~ 10 МГц
- Отображаемый средний уровень шума: -163 дБм при RBW 1Гц (тип. значение)
- Функции измерения: напряженности поля, мощности канала, занимаемой полосы частот, мощности в соседнем канате, аудио демодуляция, отношения мощности несущей к шуму, маски излучения и др.

Анализатор сигнала/спектра серии 4051

- Диапазон частот от 3 Гц до 67 ГГц
- Максимальная полоса анализа: 550 МГц
- Полоса разрешения: 1 Гц ~ 20 МГц
- Однополосный фазовый шум (несущая 1 ГГц): -125 дБн/Гц при 10 кГц
- Чувствительность измерений: -135 дБм/Гц (типовое значение 67 ГГц)
- Многочисленные функциональные опции
- Гибкие аналоговые и цифровые выходные интерфейсы

Векторный анализатор цепей серии 3672

- Диапазон частот от 10 МГц до 67 ГГц
- Гибкие типы калибровки, совместимость с несколькими комплектами для калибровки
- Высокая скорость при реализации комплексных решений по тестированию
- Форматы отображения: логарифмическая амплитуда (Log Mag), линейная амплитуда (LinMag), КСВ, диаграмма Смита и др.
- Функции: измерение импульсных S-параметров, измерения по временной области, измерения смесителей, двухмерные измерения амплитудных искажений, расширение спектра волн миллиметрового диапазона, измерение параметров антенн и калибровочной сферы РЛС (RCS) и т.д.

ЮЕ / ИНТЕРНЕЙШНЛ
ГРУППА ЮЕ

ОФИЦИАЛЬНЫЙ ДИСТРИБЬЮТОР

Санкт-Петербург (812) 313-34-40
Москва (495) 150-52-21
Екатеринбург (343) 365-90-40
Новосибирск (383) 319-17-09
Нижний Новгород (831) 220-59-64

www.yeint.ru
rf@yeint.ru