УДК 681.3

Наливайко А. М., Чебаненко А. В., Катрушенко Н. А.

МЕТОДИКА МОДУЛЬНОГО ПРОГРАММИРОВАНИЯ СИСТЕМЫ ВЕКТОРНОГО УПРАВЛЕНИЯ АСИНХРОННЫМ ДВИГАТЕЛЕМ В СРЕДЕ РАЗРАБОТКИ CCSV4 НА БАЗЕ УЧЕБНОГО СТЕНДА ФИРМЫ TEXAS INSTRUMENTS

Современная концепция разработки и отладки программного обеспечения микропроцессорных систем управления предполагает модульный подход, когда любая сложная задача разбивается на ряд программных модулей, разрабатываемых и отлаживаемых одновременно несколькими программистами, с последующим объединением наработок в один общий проект. При этом каждый файл, содержащий программный код (программный модуль), может быть написан независимо от других файлов и протестирован (отлажен) самостоятельно.

Программные модули могут быть написаны на языке Ассемблер конкретного процессора или на языке высокого уровня C/C++.

Основное преимущество программирования на языке высокого уровня состоит в универсальности языка – его аппаратной независимости от типа процессора. Функцию адаптации к конкретному процессору выполняет программа компилятор. Главный недостаток – более высокие требования к ресурсам процессора, как по производительности, так и по объему памяти программ/данных. В связи с ростом производительности современных микроконтроллеров, увеличением объема встроенной памяти, развитием двухъядерных архитектур центрального процессора с аппаратной поддержкой вычислений с фиксированной и плавающей точкой, специальных средств оптимизации выходного кода этот недостаток постепенно нивелируется [1–2].

Преимущества языка программирования C++ и специфика создания высокоуровневого программного кода позволили упростить процесс написания программ для систем автоматического управления электроприводом. Так, при использовании модульной структуры программного кода, увеличивается эффективность отладки проекта, сокращается время на разработку компонентов структурной схемы САУ.

Целью данной работы является нахождение оптимального метода построения системы автоматического управления, реализующего векторное управление асинхронным двигателем на базе сигнального контроллера C2000 фирмы Texas Instruments.

Для реализации систем управления и отладки их рабочих модулей существуют различные программные и аппаратные средства. В последнее время большую популярность на рынке платформ управления электроприводом приобретают сигнальные процессоры.

Основой сигнальных контроллеров является ядро процессора цифровой обработки сигналов (DSP).

Процессор цифровой обработки сигналов – специфическое устройство, специализированное для типичных математических операций манипулирования цифровыми данными, которые вырабатываются сигнальными преобразователями. Задача DSP – проводить технологическую операцию обработки данных как можно быстрее, чтобы быть способным обрабатывать входной поток данных в реальном времени.

Процессоры цифровой обработки сигналов подобны микропроцессорам, но содержат аппаратные модули для ускорения процесса вычисления сложных математических операций: дополнительный аппаратный умножитель; дополнительное арифметическое устройство; дополнительные системные шины параллельного доступа; дополнительный аппаратный сдвиг для масштабирования и/или умножения/деления на 2n.

Такими преимуществами сигнальных процессоров обладает контроллер TMS320F28335 фирмы Texas Instruments.

Процессоры семейства F28x (C2Sx) являются представителями платформы TMS320C2000. Данная платформа процессоров изначально создавалась для приложений управления электродвигателями и электроприводами.

Контроллеры TMS320F2833x обладают следующими техническими характеристиками: высокопроизводительный 32-битный DSP; 32×32 бита или два 16×16 бита MAC; IEEE модуль с плавающей запятой; атомарные инструкции чтения-модификации-записи; 8-уровневый полностью защищенный конвейер; менеджер прерываний; 256K слов Flash памяти на кристалле; модуль защиты кода (CSM); два менеджера событий; 12-битный модуль АЦП; прямой доступ в память; сторожевой таймер; коммуникационные устройства.

Но для полной реализации модульного управления электродвигателем, да и других задач, одного лишь сигнального процессора и соответствующего программного обеспечения не достаточно. Нужны также и периферийные устройства, которые помогут реализовать алгоритмы управления. Сигнальный процессор и периферийные устройства, а также частотный преобразователь для управления электродвигателями переменного тока, объединяет в себе учебный стенд TMDSHVMTRPFCKIT фирмы Texas Instruments, на базе контроллера серии C2000 F28335[2].

Учебный стенд TMDSHVMTRPFCKIT позволяет отладить работу микроконтроллеров Piccolo или Delfino и их аналоги в условиях высокого напряжения. Простой микроконтроллер позволяет управлять коррекцией коэффициента мощности (PFC) и управлять двигателем. При коррекции коэффициента мощности допустимое входное напряжения 110–240 В, мощность до 750 Вт, с замкнутой системой управления. Двигателем можно управлять и без коррекции коэффициента мощности, при этом входное напряжение может достигать 400 В, выходная мощность – до 1,5 кВт. С помощью данного набора можно управлять наиболее распространенными типами двигателей: асинхронным двигателем, бесщеточным двигателем постоянного тока и синхронным двигателей с постоянным магнитом. Микроконтроллер управляет каждым типом двигателя, используя различные методы управления с замкнутым контуром (трапециевидная, U/f, или FOC) с помощью датчика и без датчиков. Внешний вид стенда TMDSHVMTRPFCKIT представлен на рис. 1.

Рис. 1. Вид учебного стенда TMDSHVMTRPFCKIT

Для программирования DSP TMS320F28335 используется программное обеспечение Code Composer Studio.

Code Composer Studio – интегрированная среда разработки для создания кода для DSP и/или ARM процессоров семейства TMS320, и других процессоров, таких как MSP430, выпускаемых Texas Instruments. Code Composer Studio включает операционную систему реального времени DSP/BIOS. Также в состав продукта входят симуляторы и поддержка JTAG-ориентированной отладки.

С помощью Code Composer Studio создан программу векторного управления АД, разработанный фирмой Texas Instruments, прилагаемый к стенду TMDSHVMTRPFCKIT. Он включает блоки, которые являются составными частями структурной схемы векторного управления асинхронным двигателем. Таким образом, для отладки доступно шесть уровней проекта, в каждом из которых будут проверяться новые блоки. Тестируемые модули на каждом уровне приведены в табл. 1.

Таблица 1

Тестируемые модули на каждом уровне построения программы векторного управления

Модуль ПО	Этап 1	Этап 2	Этап 3	Этап 4	Этап 5	Этап 6
PWMDAC_MACRO	+	+	+	+	+	+
RC_MACRO	+	+	+	+		
RG_MACRO	+	+	+	+		
IPARK_MACRO	++	+	+	+	+	+
SVGEN_MACRO	++	+	+	+	+	+
PWM_MACRO	++	+	+	+	+	+
CLARKE_MACRO		++	+	+	+	+
PARK_MACRO		++	+	+	+	+
VOLT_MACRO		++	+	+	+	+
CAP_MACRO			++	+	+	
SPEED_PR_MACRO			++	+	+	
QEP_MACRO			++	+	+	
PEED_FR_MACRO			++	+	+	
PID_MACRO (IQ)			++	+	+	+
PID_MACRO (ID)			++	+	+	+
ACI_FE				++	+	+
ACI_SE				++	+	+
PID_MACRO (SPD)					++	++

В табл. 1 обозначено:

− «+» этот модуль используется на данном уровне тестирования программы;

− «++» этот модуль тестируется на данном уровне.

В табл. 1 используются модули программного обеспечения, расшифровка которых приведена в табл. 2.

При отладке каждого уровня используется следующий алгоритм выполнения программы:

1) инициализация S/W модулей;
2) инициализация временной базы;
3) включение входов ядра и ePWM time base CNT_zero;
4) инициализация других систем и параметров модулей;
5) начало замкнутого контура программы;
6) ePWM1_INT_ISR;
7) сохранение контекста и очистка входящих флагов;
8) выполнение аналого-цифрового преобразования (фазный ток и напряжение звена постоянного тока);
9) выполнение преобразования Кларка/Парка;

10) построение ПИД модулей (iq, id и скорость);

11) создание модулей ipark (обратное преобразование Парка) и svgen_dq (образование пространственного вектора);

12) выполнение расчета напряжения;

13) создание модулей ACI_FE (вычислители потока) и ACI_SE (вычислители скорости);

14) выполнение ШИМ-модуляции;

15) восстановление контекста;

16) возврат к началу замкнутого контура программы.

Таблица 2

Расшифровка модулей программного обеспечения

Название модуля Macro	Объяснение
CLARKE	Преобразование Кларка
PARK / IPARK	Прямое и обратное преобразование Парка
PID	ПИД-регулятор
RC	Рамп контроллер (ограничитель скорости нарастания)
RG	Генератор рамп / пилообразного напряжения
QEP / CAP	QEP и CAP приводы (опция для настройки контура скорости с датчиком скорости)
SPEED_PR	Измерение скорости (основанное на периоде сигнала датчика)
SPEED_FR	Измерение скорости (основанное на частоте сигнала датчика)
ACI_SE / ACI_FE	Вычислители скорости и потока для бесдатчиковых проектов
SVGEN	Пространственный вектор ШИМ с Quadrature контроллером (включая обратное преобразование Кларка)
VOLT	Вычисление фазного напряжения
PWM / PWMDAC	Приводы ШИМ и ШИМЦАП

Рассмотрим уровни построения программы векторного управления двигателем [3].

Уровень 1. На этом этапе двигатель должен быть отключен. В разделе уровень 1 описываются действия для «минимальной» системной проверки, которые включают работу системы прерываний, периферийного и системно-независимого I_PARK_MACRO (обратного преобразования Парка) и SVGEN_MACRO (генератор пространственного вектора) модулей и периферийно зависимого модуля PWM_MACRO (инициализация и обновление ШИМ). При отладке уровня переменная с именем «IsrTicker» будет, увеличивается, убедиться в этом можно наблюдая за переменной в окне просмотра. Это подтверждает, что система прерываний работает должным образом.

В программном обеспечении используются следующие ключевые переменные:

− SpeedRef (Q24) переменная изменения скорости вращения ротора в относительных единицах;

− VdTesting (Q24) переменная изменения напряжения оси d в относительных единицах;

− VqTesting (Q24) переменная изменения напряжения оси q в относительных единицах.

Модуль I_PARK_MACRO генерирует выходной сигнал для модуля SVGEN_MACRO. За тремя выходами модуля SVGEN_MACRO можно наблюдать на графиках при отладке, как показано на рис. 2, где T_a, T_b и T_c волны, сдвинутые относительно друг друга на 120°, причем T_b должна отставать от T_a на 120°, а T_c опережать T_a на 120°.

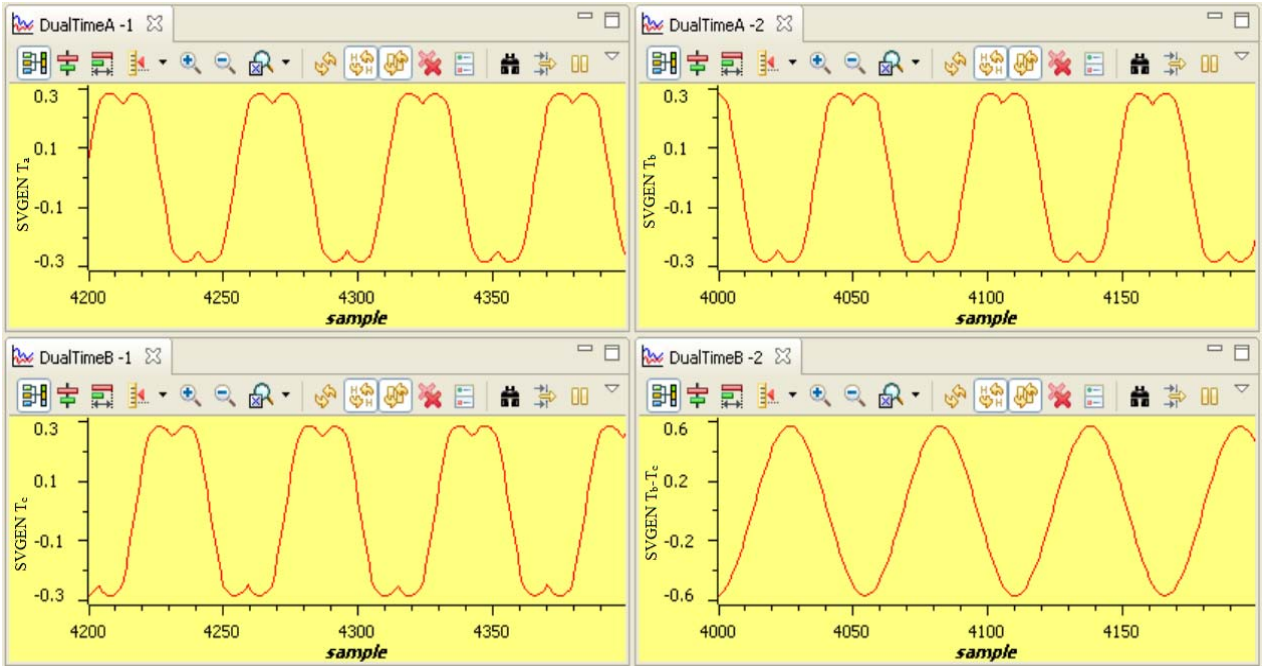

Рис. 2. Рабочий цикл выходов SVGEN T_a, T_b, T_c и T_b-T_c

Функциональная схема построения уровня 1 показана на рис. 3.

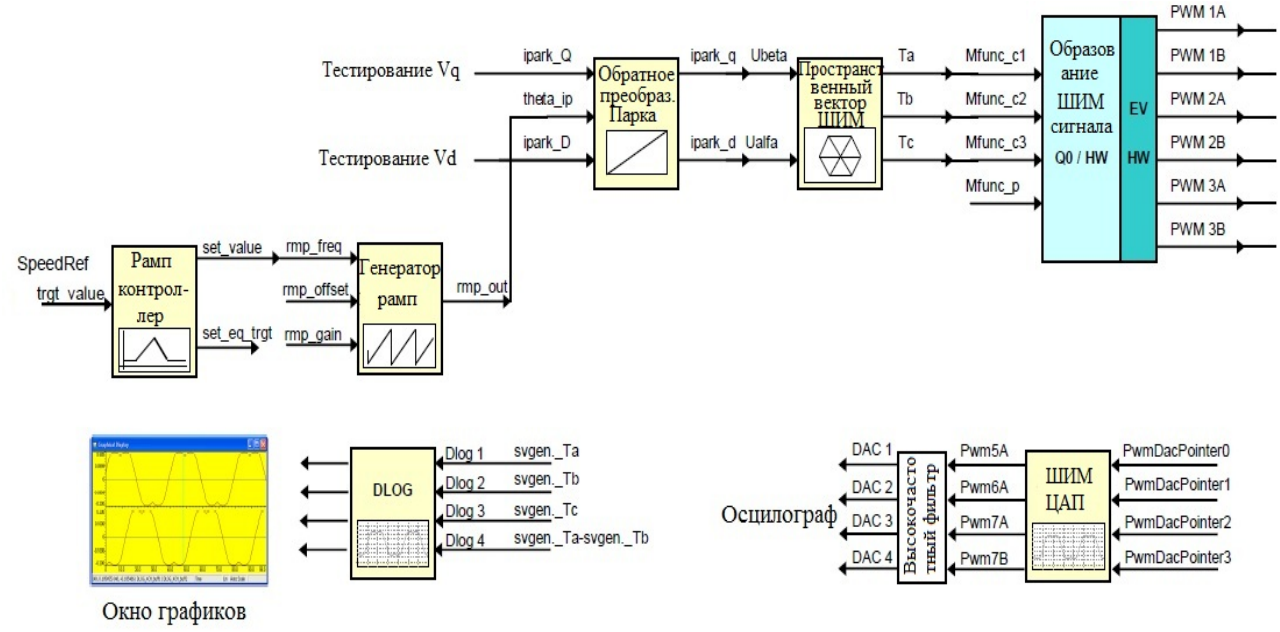

Рис. 3. Функциональная схема построения уровня 1

Уровень 2. На этом уровне проверяется аналогово-цифровой преобразователь, подключаются блоки преобразования Кларка и Парка, а также производится расчет фазного напряжения. На данном этапе двигатель подключается к преобразователю. В программном обеспечение будут использоваться такие же ключевые переменные, как и в уровне 1.

Для проверки модуля расчета фазного напряжения VOLT_MACRO, необходимо постепенно повышать напряжение в звене постоянного тока. Тогда при отладке уровня можно будет наблюдать кривые, показанные на рис. 4.

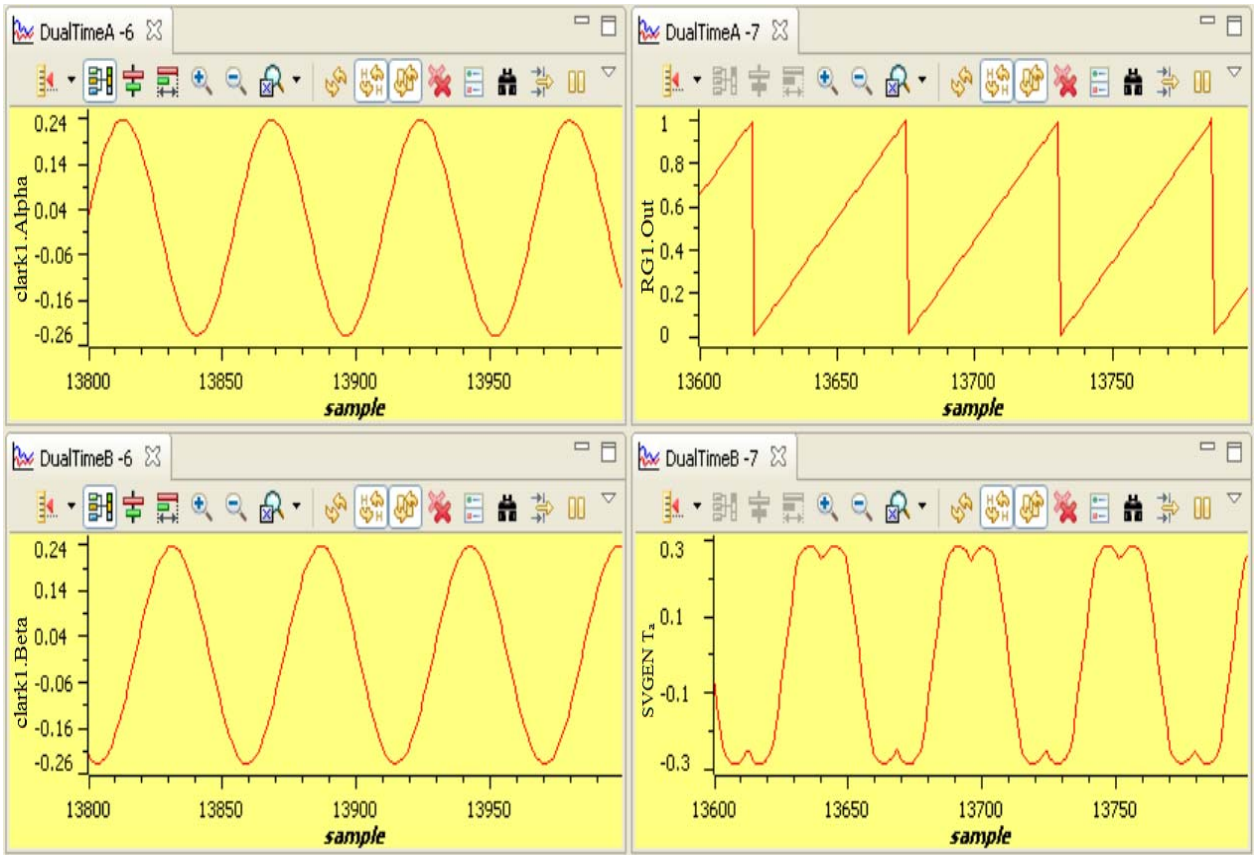

Рис. 4. Расчет фазного напряжения модулем volt1, модули rg1.Out и svgen_dq1.Ta

При преобразовании Кларка три измеренных линейных тока трансформируются в два фазных тока в системе координат dq. Выходы этого модуля можно проверить при отладке используя окна с графиками (рис. 5). Если процессы при отладке протекают правильно, то синусоида clark1.Alpha должна опережать синусоиду clark1.Beta на 90°, амплитуды синусоид должны быть одинаковыми.

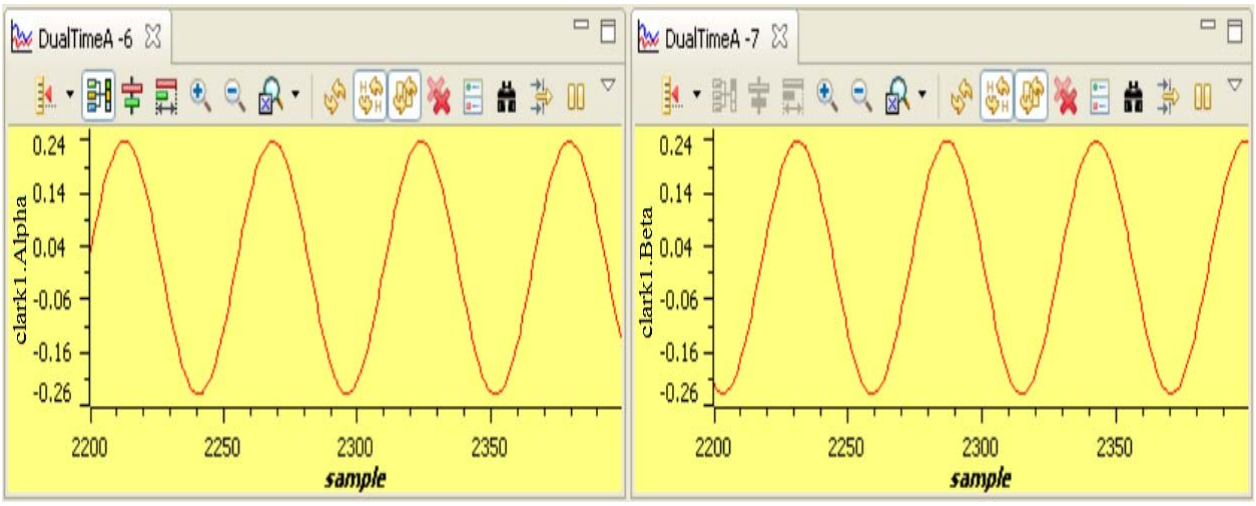

Рис. 5. Синусоиды преобразованного напряжения фаз альфа и бета

Функциональная схема построения уровня 2 показана на рис. 6.

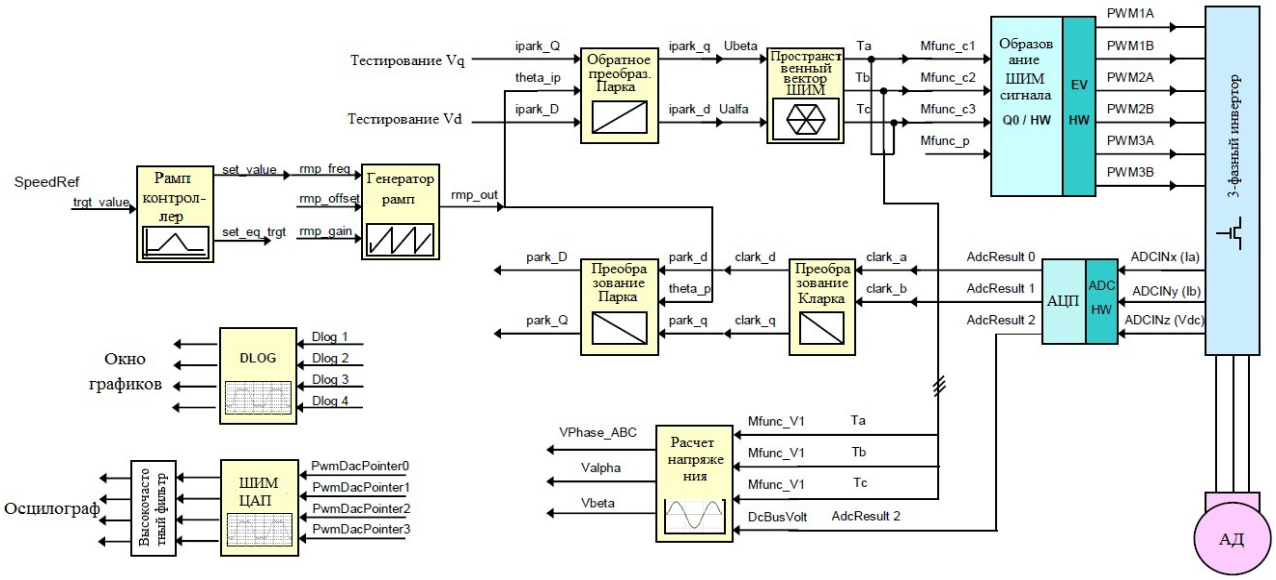

Рис. 6. Функциональная схема построения уровня 2

Уровень 3. В этом разделе осуществляется тестирование регулировки тока в осях dq, выполняемое модулем PID_REG3, и проверяется работа модуля измерения скорости. Для разрешения регулирования тока, выходы этих двух ПИД-регуляторов обязательно должны быть настроены для соответствующей работы. В программном используются ключевые переменные:

− SpeedRef (Q24) переменная изменения скорости вращения ротора в относительных единицах;

− IdRef (Q24) переменная изменения тока оси d в относительных единицах;

− IqRef (Q24) переменная изменения тока оси q в относительных единицах.

При построение на данном уровне, двигатель питается от переменного напряжения, и ток двигателя динамически регулируется с помощью модуля PID_REG3 путем преобразования Парка.

Таким образом, в графических окнах отображаются графики, показанные на рис. 7. Функциональная схема построения уровня 3 показана на рис. 8.

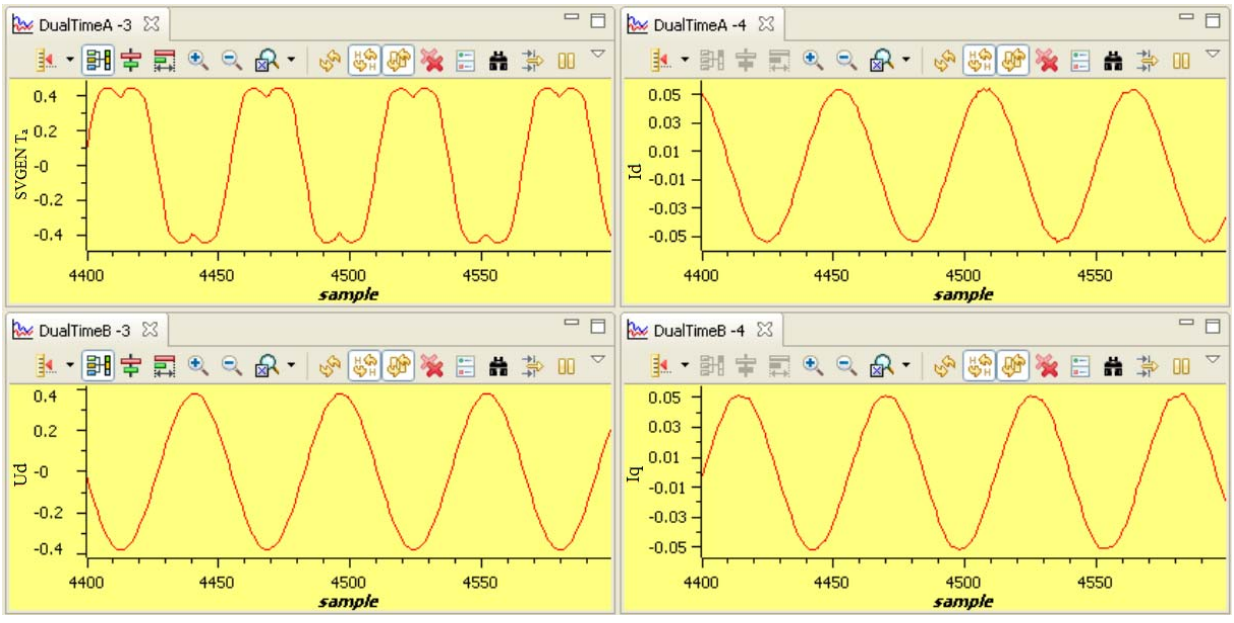

Рис. 7. Кривые токов осей d и q, напряжение оси d и Svgen_dq1.Ta

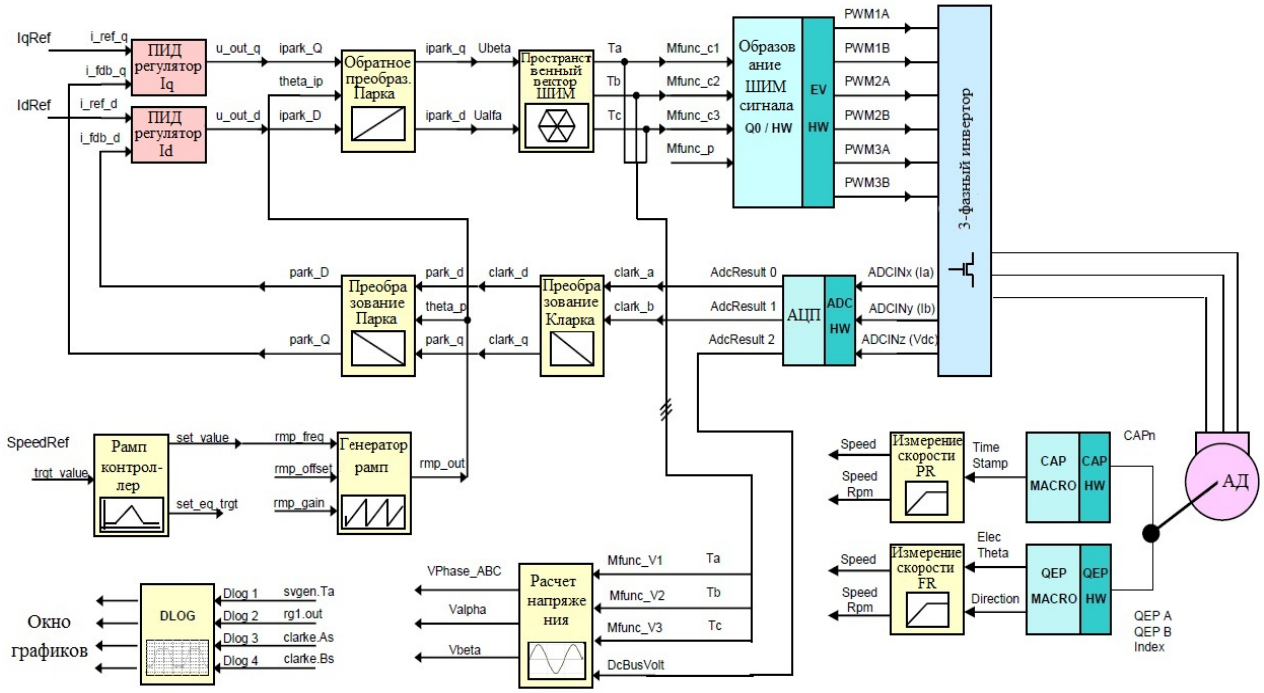

Рис. 8. Функциональная схема построения уровня 3

Уровень 4. На этом уровне проверяем правильность вычисления магнитного потока (ACI_FE) и скорости с разомкнутым контуром (ACI_SE), соответственно. В программном обеспечение будут использоваться такие же ключевые переменные, как и в уровне 3. Настройка пропорционального и интегрального коэффициентов усиления (K_{p_fe} и K_{i_fe}) внутри вычислителя магнитного потока может быть критична к режиму работы с очень низкой скоростью. Во время построения этого уровня в графических окнах Code Composer Studio должны отображаться графики, приведенные на рис. 9.

Функциональная схема построения уровня 4 показана на рис. 10.

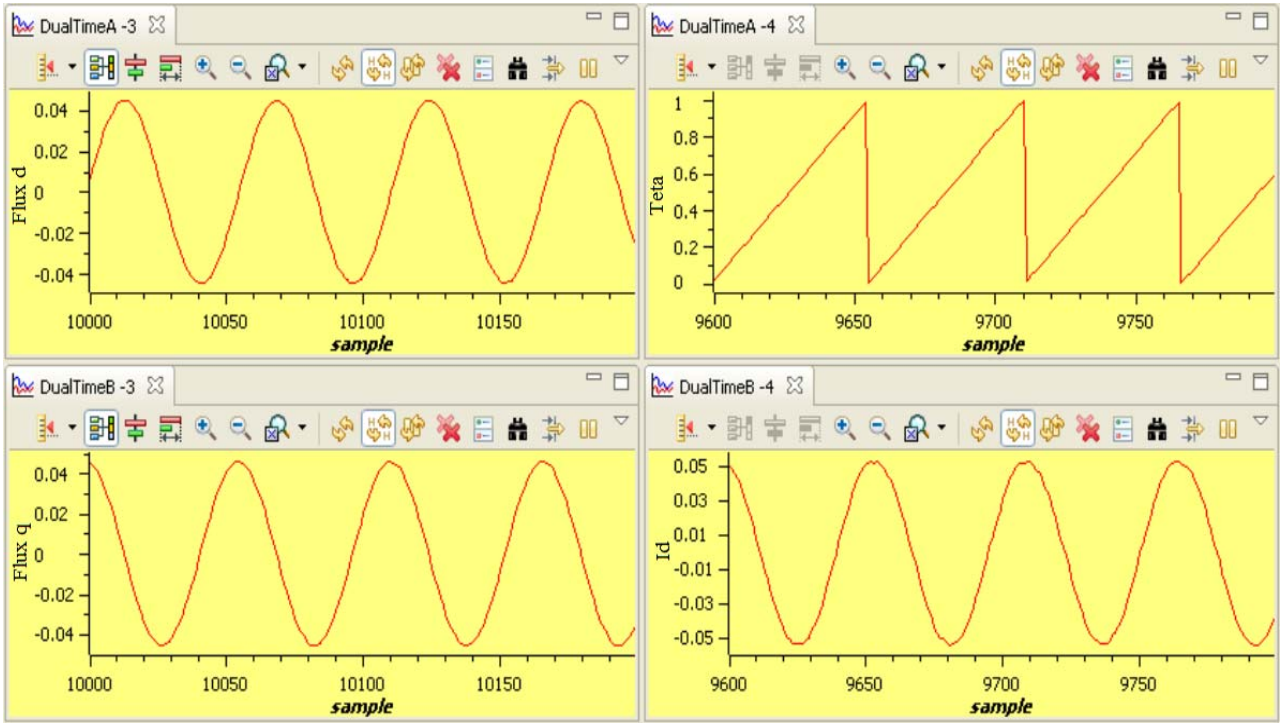

Рис. 9. Кривые расчетного магнитного потока осей dq, угол θ и ток оси d

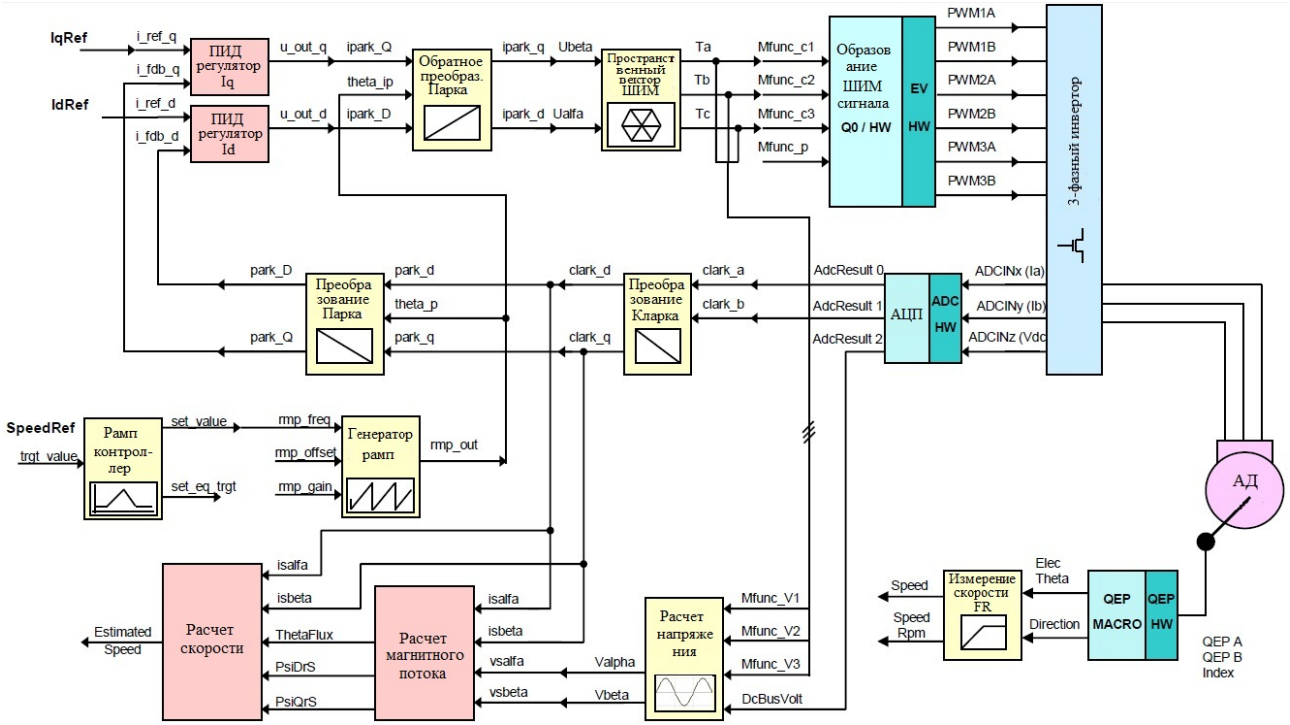

Рис. 10. Функциональная схема построения уровня 4

Уровень 5. В этом разделе проверяем регулятор скорости, основанный на модуле PID_REG3. Используем замкнутый контур с обратной связью по скорости. Следует подчеркнуть, что двигатель может вращаться только в одном направлении, когда измеренная скорость (с датчика положения) не дает информации о направлении, как это сделано при измерении скорости на основе QEP. Поэтому, если датчик скорости не инкрементный энкодер, в переменной SpeedRef необходимо указывать положительные значения.

Функциональная схема построения уровня 5 показана на рис. 11.

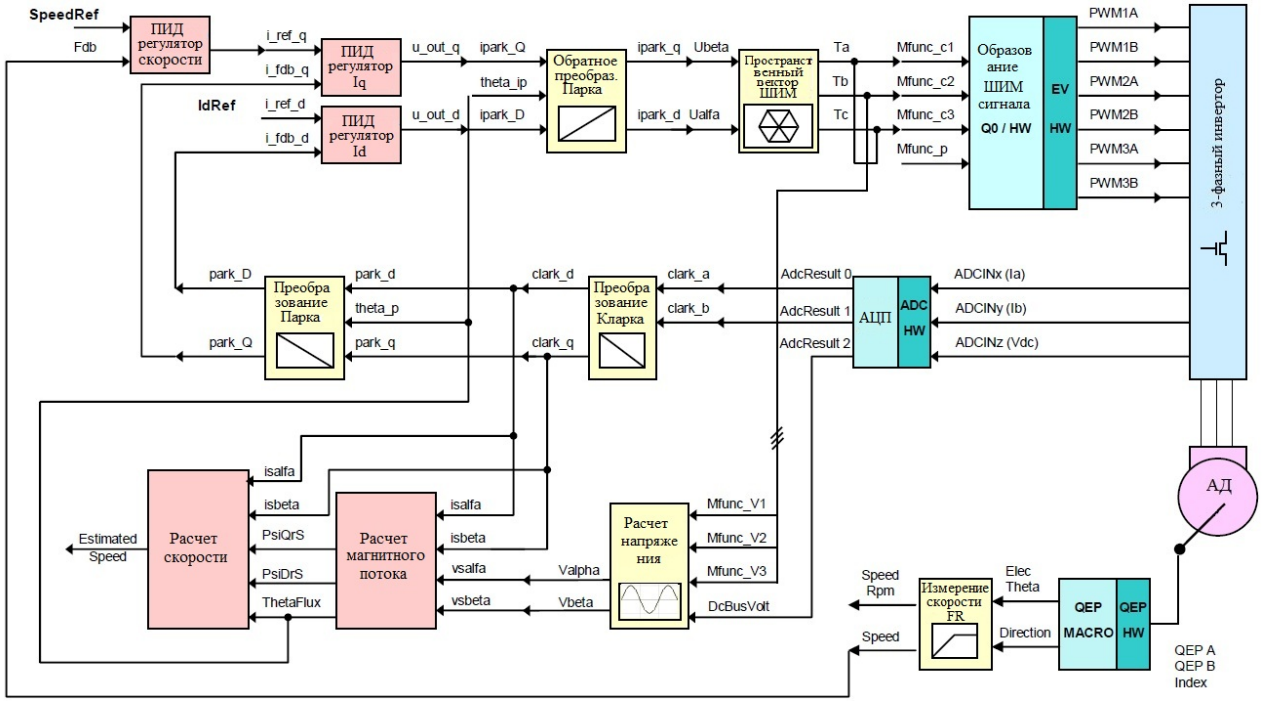

Рис. 11. Функциональная схема построения уровня 5

Уровень 6. В этом разделе проверяем регулятор скорости, выполненный модулем PID_REG3. Контур скорости замыкается с помощью расчета скорости (без датчика положения). В отличие от системы с датчиком в предыдущем уровне, двигатель может вращаться в обоих направлениях, потому что расчетная скорость (от блока расчета скорости) дает информацию о направлении. Таким образом, SpeedRef может быть положительным или отрицательным значением. Функциональная схема построения уровня 6 показана на рис. 12.

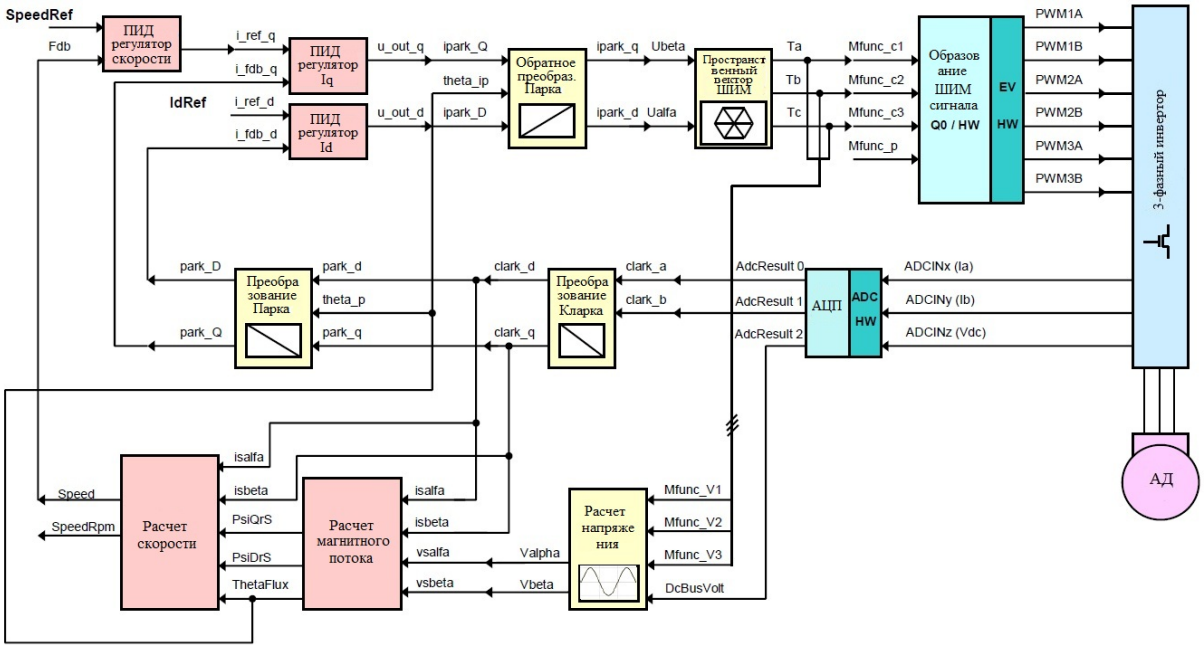

Рис. 12. Функциональная схема построения уровня 6

ВЫВОДЫ

И так, в процессе создания векторной системы управления был выбран метод модульной отладки программного кода. На каждом этапе создания программы исследована работа отдельных блоков структурной схемы, что позволило получить в конечном результате работоспособную систему управления асинхронным двигателем. В заключение можно сказать, что, во-первых, эта работа позволила создать лабораторную базу для изучения векторного управления асинхронным двигателем. Во-вторых, большим достоинством данного метода является то, что при обнаружение ошибок в программе, при поэтапной отладки, можно легко выявить блок, в котором она находится. Кроме того, программа, разбитая на модули, облегчает преобразование бесдатчиковой системы векторного управления в систему с использованием обратной связи по скорости и положению. И в-третьих, методика модульного построения адаптирована к возможностям реальной лабораторной базы и русскому языку для дальнейшего использования ее студентами.

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. Практический курс разработки и отладки программного обеспечения сигнальных микроконтроллеров TMS320x28xxx в интегрированной среде Code Composer Studio: учеб. пособие / А. С. Анучин, Д. И. Алямкин, А. В. Дроздов и др.; под общ. ред. В. Ф. Козаченко. – М. : Издательский дом МЭИ, 2010. – 270 с.

2. High Voltage Digital Motor Control Kit Quick Start Guide

3. Sensorless Field Oriented Control of 3-Phase Induction Motors

4. Горбань Р. Н. Современный частотно-регулируемый электропривод / Р. Н. Горбань, А. Т. Янукович; под ред. А. В. Гаврилова – С-Петербург, СПЭК. 2001.

5. Дартау В. А. Асинхронные электроприводы с векторным управлением / В. А. Дартау, В. В. Рудаков, И. М. Столяров – Л. : Энергоатомиздат, Л.О., 1987.

Статья поступила в редакцию 05.05.2015 г.