Программное обеспечение VectorCARDServoInverter

Руководство программиста/пользователя

версия документа от 02.08.2016

Оглавление

1. Введение ... 3
2. Список файлов проекта ... 4
3. Структура программного обеспечения (ПО) ... 7
3.1. Общий обзор компоновки проекта ... 7
3.2. Прерывания и их обслуживание ... 7
3.3. Введение в CANopen ... 8
3.1. Целочисленные вычисления и относительные единицы ... 10
4. Начало работы ... 12
4.1. Программирование и запуск контроллера ... 12
4.2. Подключение к ПК через UniCON ... 15
4.3. Подключение датчиков положения двигателя ... 17
4.4. Настройка АЦП ... 17
4.5. Настройка модуля защит ... 18
4.6. Режимы работы (структуры системы управления) ... 18
4.7. Запуск привода (Быстрый старт) ... 22
5. Модули ПО и их настройка ... 31
5.1. ПИД регулятор ... 31
5.2. Кривая U/f ... 31
5.3. Задатчик интенсивности (ЗИ) ... 32
5.4. Модуль фазных преобразований ... 32
5.5. Модуль координатных преобразований ... 33
5.6. Модуль ШИМ ... 33
5.1. Модуль АЦП ... 37
5.2. Модуль ДПР на элементах Холла ... 39
5.3. Модуль инкрементального энкодера ... 40
5.4. Модуль цифрового осциллографа (даталоггер) ... 41
5.5. Модуль защит ... 42
5.6. Модуль часов реального времени ... 44
5.7. Модуль работы с драйвером ключей DRV8301 ... 44
5.8. Отображение текущих параметров привода ... 44
5.9. Различные настройки системы ... 44
5.1. Переменные для отладки ... 44
5.2. Счетчики ... 45
6. О модульности программ ... 46
7. Соглашения по оформлению и структуре программы ... 48
7.1. Наименование переменных, функций и т.д. ... 48
7.2. Наименование типов ... 48
7.3. Комментарии к коду программы ... 48
8. Заключение ... 49

1. Введение

Данное руководство предназначено для пользователей отладочного комплекта управления электродвигателем на базе платы VectorCARD производства ООО «НПФ Вектор». Оно описывает основные функции и организацию демонстрационного программного обеспечения VectorCARDServoInverter микроконтроллера, поставляемого вместе с комплектом. Программное обеспечение предназначено для микроконтроллера К1921ВК01Т фирмы ОАО «НИИЭТ», отладочной платы VectorCARD и инвертора из комплекта DRV8301-HC-EVM или DRV8312-C2-KIT Texas Instruments. ПО микроконтроллера создано в среде разработки VectorIDE (Eclipse+GCC+OpenOCD, загрузить можно по адресу http://motorcontrol.ru/production/soft/vector-ide/). Для настройки системы управления, задания уставок, переключения режимов и наблюдения осциллограмм переходных процессов ПО микроконтроллера использует драйвер CANopen (в виде библиотеки), который обеспечивает связь по интерфейсу CAN с персональным компьютером. Со стороны персонального компьютера используется ПО UniCON для Windows (http://motorcontrol.ru/production/soft/softcanopen/), которое при помощи переходника CAN-USB или встроенного в VectorCARD USB-UART конвертора взаимодействует через CANopen с ПО микроконтроллера и обеспечивает интерфейс пользователя.

2. Список файлов проекта

Каталог ***asm***

startup_MCP_gcc.S – стартовый файл, с которого начинается выполнение программы. После небольшой инициализации передает управление функции main.

Каталог ***cmd***

build.ld – файл компоновки памяти проекта. Служит лишь для удобства переключения двух конфигураций, описанных в файлах **niiet_flash.ld** и **niiet_ram.ld**.

niiet_flash.ld – файл компоновки памяти проекта, предназначенный для работы программы из flash памяти микроконтроллера.

niiet_ram.ld – файл компоновки памяти проекта, предназначенный для работы программы полностью из оперативной памяти микроконтроллера (flash не прошивается).

Каталог ***include***

core_cm4_simd.h, **core_cm4.h**, **core_cmFunc.h**, **core_cmInstr.h** – заголовочные файлы для работы с ядром микроконтроллера.

K1921VK01T.h – заголовочный файл для работы с периферией микроконтроллера. Содержит объявления регистров всех периферийных устройств.

Каталог ***lib***

libc.a, **libgcc.a**, **libm.a** – стандартные библиотеки GCC для работы с микроконтроллером (версия float point).

libCANOpen_drv_LIB.a – библиотека драйвера CANopen. Реализует соответствующий протокол связи при помощи интерфейса CAN.

libV_DPR_eCAP_LIB.a – библиотека функций датчика абсолютного положения ротора с использованием модулей CAP.

libV_PWM_Vect_LIB.a – библиотека функций векторной ШИМ.

libV_UserMemory_LIB.a – библиотека для работы с пользовательской энергонезависимой памятью MC01.

Каталог ***src***

K1921VK01T_init.c – файл предварительной инициализации периферии микроконтроллера. Содержит функции включения и подготовки микроконтроллера к работе.

Каталог ***Vinclude***

build.h – конфигурационный заголовочный файл. Выбирает тип аппаратной части.

can_macros.h – макросы для удобства настройки и работы с модулем CAN.

CANBlockTransferInterface.h – заголовочный файл интерфейса блочной передачи данных.

CANOpen_drv.h – заголовочный файл драйвера CANopen.

clarke.h – заголовочный файл модуля фазных преобразований.

co_ODvars.h – объявление служебных структур драйвера CANopen.

cood2.h – объявление служебных переменных, которые должны быть определены в файле cood2.c.

DRV8301_SPI.h – заголовочный файл для работы с микросхемой драйвера ключей DRV8301.

DSP.h - файл подключения заголовочных файлов периферии и объявления системных констант.

DSP2833x_EPwm_defines.h – макросы для удобства настройки и работы с модулем ШИМ.

filter.h – заголовочный файл инерционного фильтра.

ipark.h – заголовочный файл модуля обратных координатных преобразований.

main.h – главный заголовочный файл проекта.

park.h – заголовочный файл модуля координатных преобразований.

SM_cmd_logic.h – заголовочный файл обработки команд и заданий.

SM_ctrl.h – заголовочный файл основного ДА переключения систем управления.

SM_NET.h – заголовочный файл модуля общего назначения для работы с сетью.

SM_protect.h – заголовочный файл модуля защит.
SM_SYS.h – заголовочный файл модуля-обертки для расчета остальных модулей.
V_adc.h – заголовочный файл модуля обработки АЦП.
V_AutoOffset.h – заголовочный файл модуля автоподстройки смещения датчиков тока каналов АЦП.
V_bits_to_enum_numbers.h – заголовочный файл модуля "листания" битовых переменных.
V_CANtoRS.h – драйвер для работы CANopen драйвера через RS (UART).
v_common.h – заголовочный файл с объявлением структур данных различного назначения, которые еще слишком малы для того, чтобы организовывать их в отдельный модуль.
V_CurPar.h – заголовочный файл расчета наблюдаемых текущих параметров.
v_data_log.h – заголовочный файл 4-х канального логгера для осциллографирования в реальном времени.
V_DPR_eCAP.h – заголовочный файл модуля расчёта скорости и положения по трём датчикам на элементах Холла.
V_UserMemory.h – заголовочный файл работы с энергонезависимой памятью.
V_IQmath.h – заголовочный файл библиотеки функций целочисленной математики.
V_pid_reg3.h – заголовочный файл ПИД-регулятора.
V_PWM_6_12.h – заголовочный файл модуля реализации ШИМ.
V_QEP.h – заголовочный файл модуля оценки скорости и положения при помощи eQEP.
V_rmp_ctrl.h – заголовочный файл задатчика интенсивности.
V_RTC_Clock.h – заголовочный файл модуля работы с часами реального времени.
V_SSI_Encoder.h – заголовочный файл модуля работы с датчиком положения ротора с SSI интерфейсом.
vhzprof.h – заголовочный файл криваой U(f).
Vled.h – заголовочный файл управления светодиодной индикацией.

Каталог ***Vsrc***

clarke.c – модуль фазных преобразований.
cood2.c – файл словаря объектов CANOpen второго модуля CAN.
DRV8301_SPI.c – файл работы с микросхемой драйвера ключей DRV8301.
filter.c – файл инерционного фильтра.
ipark.c – модуль инверсных координатных преобразований координат.
main.c – главный файл проекта. Содержит main(), а также обработчики прерываний.
park.c – **м**одуль координатных преобразований.
SM_cmd_logic.c – обработка команд и заданий.
SM_ctrl.c – основной ДА переключения систем управления.
SM_NET.c – модуль общего назначения для работы с сетью.
SM_protect.c – модуль защит.
SM_SYS.c – модуль-обертка для расчета остальных модулей.
V_adc.c – модуль обработки АЦП.
V_AutoOffset.c – модуль автоподстройки смещения датчиков тока каналов АЦП.
V_bits_to_enum_numbers.c – модуль "листания" битовых переменных.
v_call_back.c – файл функций, вызываемых драйвером CANopen.
V_CANtoRS.c – драйвер для работы CANopen драйвера через RS (UART)
V_CurPar.c – расчет наблюдаемых текущих параметров.
v_data_log.c – 4-х канальный логгер для осциллографирования в реальном времени.
V_IQmath.c – библиотека функций целочисленной математики.
V_pid_reg3_pos.c – ПИД-регулятор для контура положения.
V_pid_reg3.c – ПИД-регулятор общего назначения.
V_PWM_6_12.c – модуль реализации ШИМ.
V_QEP.c – модуль оценки скорости и положения при помощи eQEP.
v_rmp_ctrl.c – задатчик интенсивности.

V_RTC_Clock.c – модуль работы с часами реального времени.
V_SSI_Encoder.c – модуль работы с датчиком положения ротора с SSI интерфейсом.
vhzprof.c – кривая U(f).
Vled.c – управление светодиодной индикацией.

3. Структура программного обеспечения (ПО)

3.1. Общий обзор компоновки проекта

Большая часть программного обеспечения (ПО) написана в виде модулей. Модуль обычно состоит из двух файлов – заголовочного файла с расширением *.h и одноименного файла с расширением *.c. Большинство новых модулей имеют префикс “V_” в именах файлов. Ряд модулей имеет префикс “SM_”, что расшифровывается как State Machine (дискретный автомат, машина состояний): эти модули в своей основе не являются законченным блоком для реализации какой-либо одной функции, а представляют собой логику, построенную на принципах дискретных автоматов, и служат для отслеживания состояний работы, смены режимов и управляют остальными модулями, т.е. реализуют так называемую «бизнес – логику».

В проект также добавлены стандартные библиотечные файлы для работы с микроконтроллером и его периферией.

Все стандартные *.h и *.c файлы от находятся в директориях include и src соответственно. Все самостоятельно написанные файлы аналогично размещены в Vinclude и Vsrc.

Главный файл проекта, main.c, является первым файлом, с которого начинается выполнение программы (после ассемблерного startup_MCP_gcc.S). В нем определяются экземпляры всех модулей, а также он содержит функции-обработчики, вызываемые практически по всем прерываниям в системе (исключение – драйвер CANopen, который содержит обработчик прерывания в своем файле). В файле main.h подключаются все заголовочные файлы всех модулей, а также объявляются экземпляры всех модулей, определенных в main.c. Таким образом, если внутри какого-либо модуля активно используется ряд других модулей (например, так делают модули SM_ *), то достаточно подключить в нем main.h, чтобы сделать доступной работу со всеми другими модулями проекта.

3.2. Прерывания и их обслуживание

В проекте ПО данного проекта используется ряд прерываний контроллера. Все прерывания системы (за исключением, опять же CAN), конфигурируются в модуле SM_SYS. Существует два главных прерывания, в которых выполняется большая часть программного кода и ряд остальных, служащих для какой-либо одной специализированной функции. Эти два прерывания включают:

1. **Прерывание «10кГц».** Таймер общего назначения NT_TIMER0используется для создания прерывания с частотой 10кГц. В обработчике этого прерывания рассчитывается бОльшая часть системы управления: модуль ШИМ, задатчик интенсивности, кривая U/f, модуль защит и т.п. .
2. **Прерывание «1кГц».** Аналогично прерыванию «10кГц», использует один из таймеров микроконтроллера. Нужно для задач, не требующих большой скорости выполнения, однако требующих строгую дискретизацию по времени. Здесь обрабатываются различные сетевые интерфейсы и другая логика. Используется таймер общего назначения NT_TIMER1.

Третий таймер общего назначения NT_TIMER2 используется для расчета длительности обработки прерываний. Значения счетчика таймера в начале и конце процедуры вычитаются, в результате получается длительность обработки прерывания в тактах.

Кроме этих прерываний в системе присутствует прерывание, возникающее при аппаратной аварии инвертора, три прерывания модулей захвата, прерывание модуля квадратурного энкодера и два прерывания драйвера CANopen.

Следует учитывать приоритет прерываний и их маскирование. Например, если во время обработки прерывания «10кГц» возникнет прерывание «1кГц», то во все время выполнения программного кода из обработчика «1кГц» прерывание 10кГц будет заблокировано, т.е. ключи инвертора «заморозятся», модуль защит прекратит работать и т.п. Поэтому все прерывания распределены по приоритетам, и более высокоприоритетное прерывание не может быть приостановлено более низкоприоритетным. Приоритеты задаются в виде констант в файле DSP.h. Прерывания в порядке убывания приоритета представлены в списке:

1. Прерывание по аппаратной аварии инвертора
2. Прерывание модуля захвата или модуля квадратурного энкодера (имеют одинаковый приоритет)
3. Прерывание с частотой 10 кГц
4. Прерывание модуля CAN
5. Прерывание с частотой 1 кГц

3.3. Введение в CANopen

В ПО данного контроллера используется протокол CANopen для связи с ПК (программа UniCON) или другими устройствами. Связь может осуществляться штатно через CAN интерфейс, однако также в данном ПО существует драйвер, осуществляющий взаимодействие по CANopen через последовательный интерфейс за счет микросхемы преобразователя USB-UART CP2102, установленной на VectorCARD. Протокол CANopen, используемый в данном ПО, расширен по сравнению с общепринятым стандартом и обладает следующими характеристиками:

- Позволяет соединить и адресовать в одной сети до 127 устройств;
- Реализует сервис SDO – подтверждаемые посылки с 8ю байтами данных. С помощью них можно как читать, так и записывать данные.
- Реализует сервис PDO – не подтверждаемые посылки с 8ю байтами данных. С помощью них можно только записывать данные.
- Реализует сервис HeartBeat – не подтверждаемые посылки, сигнализирующие о наличии устройства в сети и передающие его статус.
- Реализует ряд других, не используемых в данном ПО сервисов.
- Посылки SDO можно послать любому узлу в сети. Посылки содержат номер адресата, данные, контрольную сумму, адрес объекта (индекс, подындекс) и ряд служебных данных.
- Каждое устройство содержит словарь объектов – переменных, к которым имеется доступ через протокол CANopen. Посылки могут адресоваться к любому из объектов. SDO может адресовать только один объект, PDO может адресовать до 4х объектов.
- Каждый объект имеет свой адрес в словаре: индекс и подындекс (двухуровневое дерево). Объект-индекс может как содержать, так и не содержать подындексы. Если он их содержит, он является корневым элементом и не содержит данных пользователя. Объекты - подындексы могут содержать любые данные пользователя. Т.е. индекс либо корневой объект и его подындексы – объекты словаря с данными пользователя, либо он не содержит подындексов и содержит только данные пользвателя.
- У каждого объекта имеется формат данных. Например, в нем содержится информация о том, редактируемый это параметр или только для чтения, тип данных: числовой, битовый и т.п., визуализация: числовой параметр, текстовое отображение и т.п., разрядность данных, масштабирующий коэффициент и другое.
- У каждого объекта, в зависимости от формата, могут иметься дополнительные атрибуты (помимо значения и самого формата): максимум, минимум, значение по умолчанию, номер текста. Эти атрибуты хранятся в структуре словаря объектов и принадлежат данному объекту.
- У объекта хранится только идентификационный (уникальный) номер текста. Сами тексты должны храниться в памяти устройства управления данным устройством (пульт управления (далее ПУ), компьютер со специальной программой (UniCON) и т.п.). Это сделано для экономии свободной памяти контроллера. Сам текст - это «название» объекта, понятное пользователю и отображаемое на пульте управления (ПУ) как имя параметра.
- Пользователю не удобно отображение на ПУ или ПК словаря в виде индексов и подындексов. Поэтому были введены группы параметров. Каждая группа имеет свое

название. Каждый индекс словаря принадлежит той или иной группе. Если индекс содержит подындексы, то они автоматически попадают в данную группу. Например, группы параметров «Задатчик интенсивности», «Температуры» и т.п. Внутри этих групп, соответственно, будут все объекты словаря, им принадлежащие. Принадлежность определяется в словаре CANopen.

- Вся структура словаря объектов содержится в памяти контроллера. Чтобы ПУ или ПК мог адекватно управлять им, необходимо, чтобы в нем также была структура этого словаря. Поэтому был создан специальный интерфейс, позволяющий скачивать (при подключении к устройству) структуру словаря устройства (в демо-версии драйвера эта функция отключена).
- После загрузки словаря объектов он сохраняется в энергонезависимую память ПУ или на диске ПК, чтобы не требовалась повторная загрузка после.
- Так как вся система управления построена с использованием относительных единиц, а пользователю требуется отображать реальные величины (вольты, амперы), введена идеология масштабирующих коэффициентов. Масштабирующие коэффициенты – отдельные индексы словаря объектов, который обязательно должны присутствовать в словаре объектов. Значение коэффициента – число, на которое требуется умножить число в относительных единицах, чтобы получить реальные единицы. Так, если номинальному току соответствует 100 А, в относительных единицах переменная равна 0.5, то отобразится значение 50А. Кроме того, масштабирующий коэффициент задает отображаемые префикс и размерность: амперы (А), миллисекунды (мс), вольты (В) и т.п. Каждому объекту можно задать любой масштабирующий коэффициент. Всего масштабирующих коэффициентов в настоящий момент 31.
- Кроме текстов, символизирующих имена параметров, существует такое понятие, как перечисление. Перечисление – массив текстов. Они используются, если значение объекта должно быть текстовым. Например, тип торможения может быть: Выбег, Динамическое торможение, Частотное торможение. На самом деле переменная, отвечающая за тип торможения, будет меняться от 0 до 2, где 0 – Выбег, 1 - Динамическое торможение и т.п. А на ПУ будут отображаться соответствующие тексты перечисления. То, что для отображения переменной необходимо использовать перечисление, также задается в формате объекта.
- Драйвер CANopen несет также функцию загрузки и восстановления параметров из энергонезависимой памяти. Все параметры, которые входят в словарь объектов и не имеют тип «только для чтения» в функции инициализации драйвера CANopen загружаются из энергонезависимой пользовательской flash памяти. После какой-либо перенастройки привода пользователь с помощью ПУ должен дать команду сохранения, после которой все значения словаря объектов сохраняются на flash. Кроме того, существует команда загрузки параметров по умолчанию, при которой драйвер CANopen каждому элементу словаря присваивает значение, заданное у него в атрибутах как «значение по умолчанию». Технически эти команды от ПУ передаются через объекты интерпретатора команд.
- В первый момент времени, после соединения устройств в сеть (например, ПК и контроллера) происходит следующее. Каждое из устройств каждую секунду посылает сообщение сервиса HeartBeat (о том, что оно в сети). Как только специальная программа на ПК (UniCON http://motorcontrol.ru/production/soft/softcanopen/unicon/) принимает HeartBeat от контроллера, она запрашивает информацию о нем с помощью сервиса SDO. А именно читает данные с двух объектов (которые обязательно должны присутствовать в словаре), в которых закодирована информация о названии устройства и версии его ПО. После этого либо происходит скачивание словаря объектов, либо, если он уже был скачан для этого устройства, ПК становится готов к работе с устройством.
- При включении контроллера при загрузке параметров из внешней энергонезависимой памяти драйвер CANopen проверяет общую контрольную сумму загруженных дан-

ных. Если она не сходится, то драйвер загружает параметры по умолчанию. Не сходиться она может по двум причинам: произошла какая-то ошибка (маловероятно), в результате прошивки контроллера структура словаря изменилась. Таким образом, при добавлении, удалении или другом изменении параметров словаря объектов остальные значения параметров после включения контроллера не смогут загрузиться из энергонезависимой памяти и «сбросятся» на стандартные настройки.

- CANopen драйвер имеет возможность вызова заданной пользовательской функции при обращении (чтение или запись с помощью SDO) к любым заданным индексам словаря (или подындексам, в них входящим). Например, если есть необходимость выполнять какую-либо переинициализацию некоторого модуля при изменении его настроек, возможен вызов заданной пользователем функции при обращении ПУ к его параметрам. Функции сообщается, произошло событие чтения или записи. Такие пользовательские функции, вызываемые драйвером, называются callback-функциями.
- Словари объектов для первого и второго CAN определяются в файлах cood1.c и cood2.c. Определяются они в виде нескольких таблиц (массивов), в которых задаются: адреса переменных контроллера, которые соответствуют объектам словаря; форматы, номера текстов, максимумы, минимумы и др.; адреса для сохранения словаря в энергонезависимую память; принадлежность к группам; таблица callback функций. Для генерирования этих файлов в ООО «НПФ Вектор» разработано специальное ПО CoodEdit (http://motorcontrol.ru/production/soft/softcanopen/coodedit/): редактор словарей объектов CANopen, приобретаемый отдельно.

Если требуется найти в ПО контроллера переменную, соответствующую какому-либо параметру на пульте управления (или программе UniCON), это легко сделать с использованием файла cood1.c или cood2.c. Напротив каждой переменной, выводимой в словарь CANopen, в данном файле стоит комментарий с названием параметра. Таким образом, для нахождения соответствия переменной и параметра нужно выполнить поиск в данном файле текста имени параметра.

Для уверенного использования программы UniCON рекомендуется ознакомиться с соответствующим руководством пользователя, расположенному в корневой директории с программой в формате pdf. Без изучения этого документа некоторая часть возможностей программы останется «за кадром», так как не все функции работают интуитивно-понятным образом.

3.1. Целочисленные вычисления и относительные единицы

Весь проект построен на использовании целочисленных вычислений с фиксированной точкой в формате IQ8.24, где 8 разрядов отводится под целую часть, 24 под дробную. Для обеспечения таких вычислений реализована библиотека IQmath, содержащая функции умножения, деления, синуса и т.п. в указанном формате. Библиотека реализована по аналогии с такой же библиотекой Texas Instruments для микроконтроллеров этой фирмы. Функциональность и производительность библиотеки в поставляемом ПО VectorCARDServoInverter ограничена под задачи этого проекта. Полная версия библиотеки может быть приобретена отдельно.

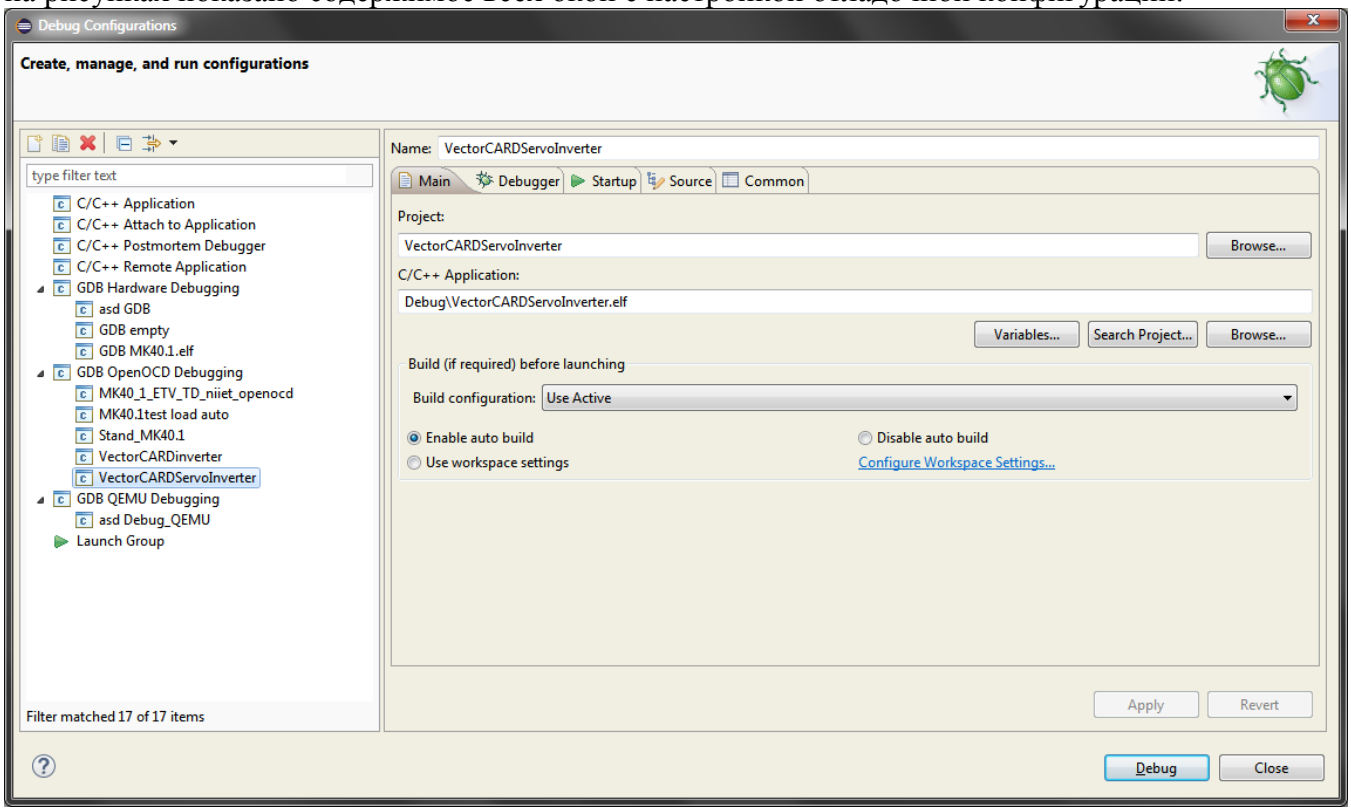
Весь проект реализован с использованием относительных величин. Это означает, что для каждой физической величины – тока, частоты вращения, напряжения и т.п. выбирается некоторая базовая величина, относительно которой нормируются все значения переменных в программе. Например, базовый фазный амплитудный ток в данном проекте равен 5А. Таким образом, если переменная, в которой хранится значение тока в программе равна единице, это обозначает ток в 5А. Такой подход является классическим для встраиваемых систем управления и выбран для исключения проблем переполнения переменных или нехватки точности. Программа UniCON, которая осуществляет управление приводом по CANopen, автоматически преобразует относительные единицы в физические и обратно. Так, прочитав из параметра величины тока значение, равное 0,5, программа автоматически умножит его на базовую величину для тока 5А и отобразит пользователю 2,5А. При вводе значения параметра пользователем UniCON, наоборот, поделит введенное значение на базовую величину и отправит данные в микроконтроллер. Список базовых значений можно посмотреть в группе параметров UniCON **Базов. парам**. Изменить

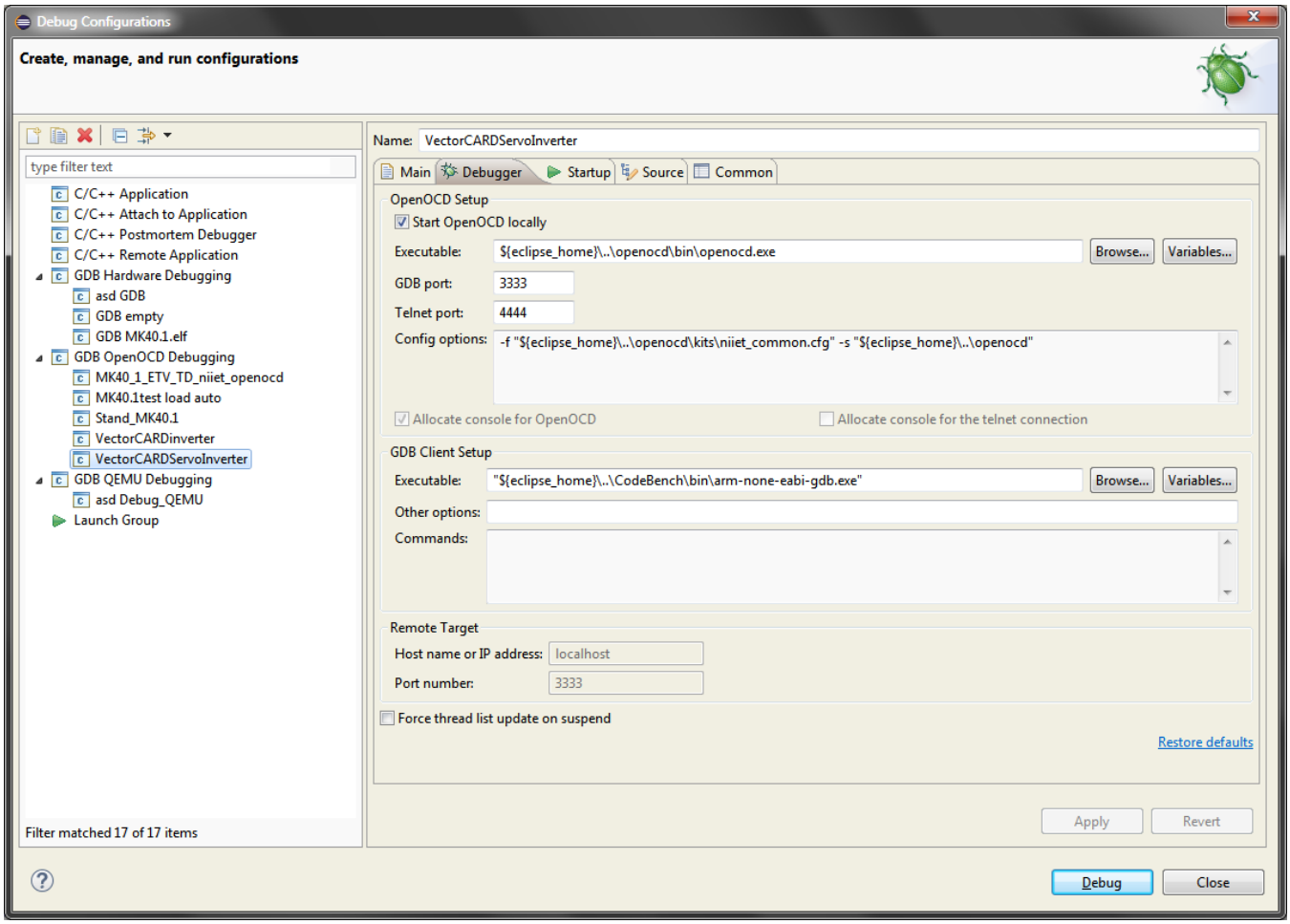
базовые параметры для данного отладочного комплекта нельзя (демонстрационная версия программы UniCON). Посмотреть истинное значение в IQ формате можно выбрав пункт «Инфо» через контекстное меню интересующего параметра в UniCON.

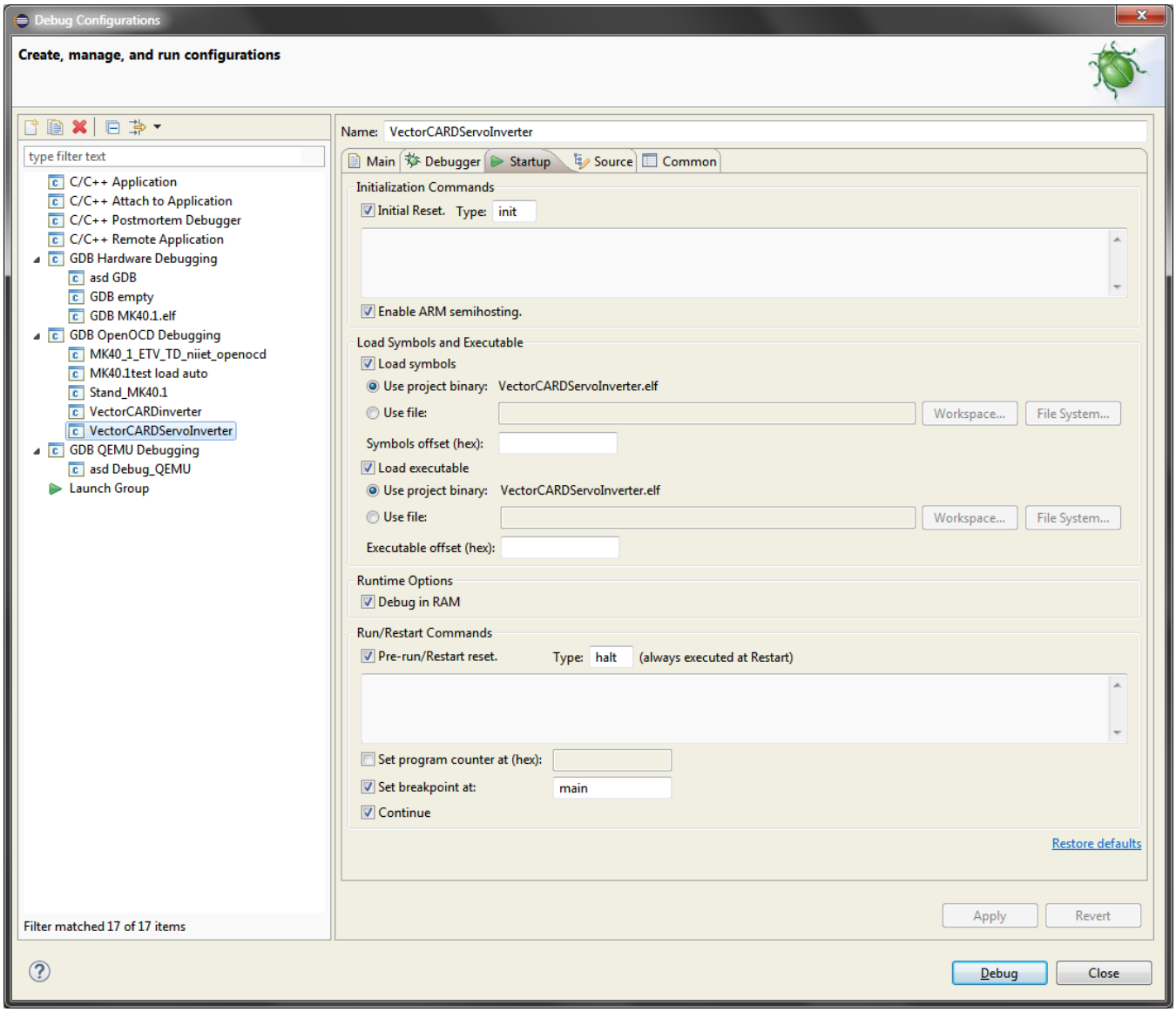
4. Начало работы

4.1. Программирование и запуск контроллера

Перед прошивкой необходимо выполнить компиляцию исходного кода проекта VectorCARDServoInverter в VectorIDE (http://motorcontrol.ru/production/soft/vector-ide/) или другой сборке Eclipse+GCC. Для этого необходимо импортировать проект в свое рабочее пространство Eclipse (через File->Import->General->Existing Projects into Workspace, выбрать проект VectorCARDServoInverter). Затем необходимо в файле Vinclude/build.h выбрать версию аппаратной части: для инвертора на базе DRV8301 раскомментировать константу DRV8301_EVM, а для DRV8312 константу DRV8312_EVM, после чего запустить сборку проекта. Убедившись, что процесс завершился без ошибок, необходимо выполнить программирование (прошивку) контроллера. Для этого нужно создать отладочную конфигурацию, нажав на угловую стрелочку с пиктограммой жука, там выбрав «Debug Configurations…», там в разделе «GDB OpenOCD Debugging» найти готовую конфигурацию от другого проекта и скопировать её. В скопированной конфигурации изменить на вкладке Main название проекта и путь до выходного elf файла. Если по какой-то причине готовой конфигурации для проекта нет, необходимо создать новую. Ниже на рисунках показано содержимое всех окон с настройкой отладочной конфигурации.




Debug Configurations
Create, manage, and run configurations
type filter text
C/C++ Application
C/C++ Attach to Application
C/C++ Postmortem Debugger
C/C++ Remote Application
GDB Hardware Debugging
asd GDB
GDB empty
GDB MK40.1.elf
GDB OpenOCD Debugging
MK40_1_ETV_TD_niiet_openocd
MK40.1test load auto
Stand_MK40.1
VectorCARDinverter
VectorCARDServoInverter
GDB QEMU Debugging
asd Debug_QEMU
Launch Group
Filter matched 17 of 17 items
Name: VectorCARDServoInverter
Main
Debugger
Startup
Source
Common
OpenOCD Setup
Start OpenOCD locally
Executable: ${eclipse_home}\..\openocd\bin\openocd.exe
Browse...
Variables...
GDB port: 3333
Telnet port: 4444
Config options: -f "${eclipse_home}\..\openocd\kits\niiet_common.cfg" -s "${eclipse_home}\..\openocd"
Allocate console for OpenOCD
Allocate console for the telnet connection
GDB Client Setup
Executable: "${eclipse_home}\..\CodeBench\bin\arm-none-eabi-gdb.exe"
Browse...
Variables...
Other options:
Commands:
Remote Target
Host name or IP address: localhost
Port number: 3333
Force thread list update on suspend
Restore defaults
Apply
Revert
Debug
Close


Debug Configurations
Create, manage, and run configurations
type filter text
C/C++ Application
C/C++ Attach to Application
C/C++ Postmortem Debugger
C/C++ Remote Application
GDB Hardware Debugging
asd GDB
GDB empty
GDB MK40.1.elf
GDB OpenOCD Debugging
MK40_1_ETV_TD_niiet_openocd
MK40.1test load auto
Stand_MK40.1
VectorCARDinverter
VectorCARDServoInverter
GDB QEMU Debugging
asd Debug_QEMU
Launch Group
Filter matched 17 of 17 items
Name: VectorCARDServoInverter
Main
Debugger
Startup
Source
Common
Initialization Commands
Initial Reset. Type: init
Enable ARM semihosting.
Load Symbols and Executable
Load symbols
Use project binary: VectorCARDServoInverter.elf
Use file:
Workspace...
File System...
Symbols offset (hex):
Load executable
Use project binary: VectorCARDServoInverter.elf
Use file:
Workspace...
File System...
Executable offset (hex):
Runtime Options
Debug in RAM
Run/Restart Commands
Pre-run/Restart reset. Type: halt (always executed at Restart)
Set program counter at (hex):
Set breakpoint at: main
Continue
Restore defaults
Apply
Revert
Debug
Close

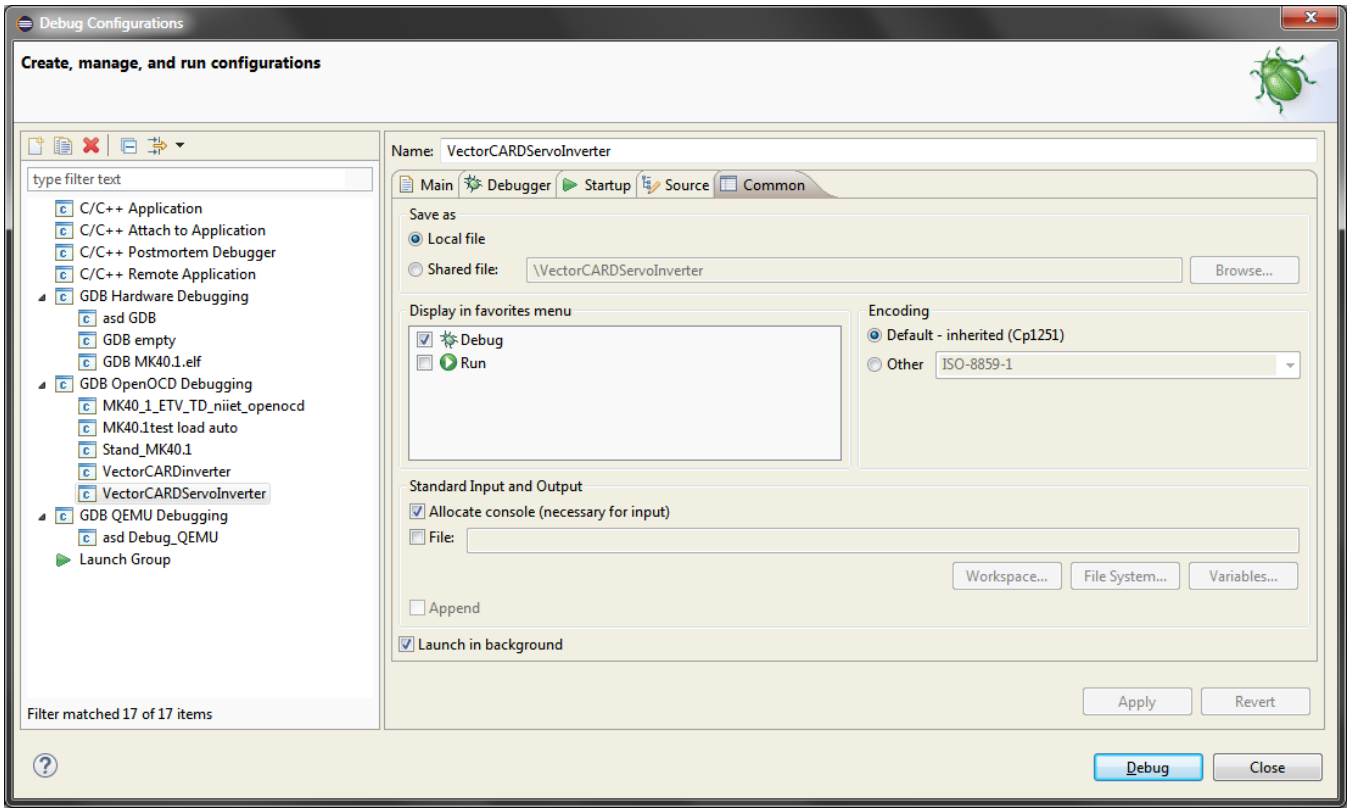


Рис. 1 Содержимое окон настройки отладочной конфигурации

Содержимое текстовых полей:

```
${eclipse_home}\..\openocd\bin\openocd.exe
-f "${eclipse_home}\..\openocd\kits\niiet_common.cfg" -s "${eclipse_home}\..\openocd"
"${eclipse_home}\..\CodeBench\bin\arm-none-eabi-gdb.exe"
```

Далее нужно подключить JTAG-эмулятор, установить на него драйвера, запустить процесс отладки, щелкнув в VectorIDE по стрелочке рядом с кнопкой Debug (с зеленым жучком) и выбрав Debug-конфигурацию VectorCARDServoInverter Debug. Если с подключением проблемы, изучите инструкцию на VectorIDE или обратитесь с вопросом к специалистам ООО «НПФ Вектор».

4.2. Подключение к ПК через UniCON

Подключите VectorCard к ПК при помощи переходника CAN-USB (разъем J7 на плате DRV8301-HC-EVM Texas Instruments, полярность можно определить опытным путем, перевернув разъем, если связи не будет). Опционально связь с ПК может осуществляться через последовательный интерфейс за счет микросхемы преобразователя USB-UART CP2102, установленной на VectorCARD и программного драйвера микроконтроллера V_CANtoRS, передающего CAN посылки через RS (UART) интерфейс.

Подайте питание на инвертор, плата VectorCARD и датчики положения двигателя запитаются от него, дополнительного питания не требуется. Рекомендуемое напряжение питания инвертора 24В – это безопасно для экспериментов «на столе», для вращения двигателя *ACM601V36-01-1000* с небольшой нагрузкой достаточно источника на ток около 1А.

Запустите программу UniCON (предварительно установив драйвера на usb-can переходник). Если вы используете CAN интерфейс с переходником CAN-USB фирмы марафон, то выберите в меню параметров модуля связи программы UniCON модуль связи «MARATHON». Для связи по последовательному интерфейсу подключите USB кабель в VectorCARD в боковой разъем X2, а в параметрах модуля связи UniCON выберите «USB RS» драйвер, задайте необходимый номер виртуального COM порта. Различие в результирующей скорости обмена данными: через CAN интерфейс осциллограммы загружаются на порядок быстрее.

Если все сделано верно, отобразится узел в сети CANopen с названием «ARM_MCP», станет доступна кнопка «Подключиться». Нужно нажать на кнопку, после можно приступать к работе. На Рис. 2 показано окно программы UniCON при успешном подключении.

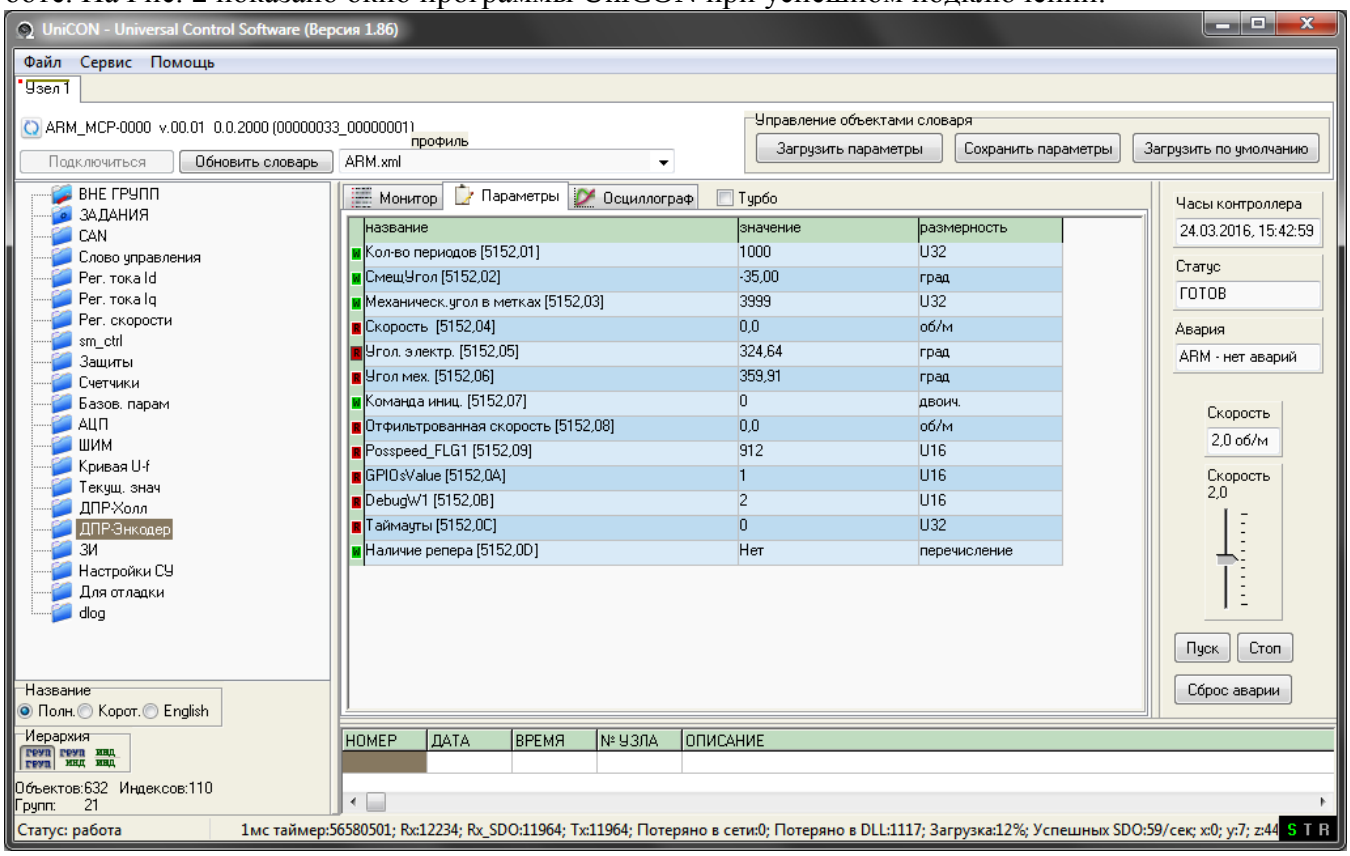


Рис. 2 Окно программы UniCON после подключения к устройству

В левой части окна программы расположен список объектов, в центре – значения объектов или осциллограф, а в правой – панель управления устройства. Для простоты последующего указания параметра будет применяться такое именование: **Группа.Параметр**. Например, **АЦП.Ток фазы А**. В данном случае параметр – это объект словаря CANopen.

Рекомендуется изучить руководство пользователя на UniCON (файл «Руководство UniCON.pdf» в каталоге с программой).

Если подключиться не удается, то проверьте следующее:

1. Запущена ли программа проекта VectorCARDServoInverter в микроконтроллере? Светодиоды в верхней части должны мигать в виде «бегущий огонек».
2. При подключении через CAN проверьте, установлены ли драйвера на переходник usb-can фирмы Марафон? Световой индикатор «USB» на переходнике должен светиться. Можно запустить служебную утилиту canwize (поставляется с драйверами), открыть нулевой канал, выбрать Baudrate 125 Kbit/s, нажать start. Успешное запуск переходника говорит о корректности работы драйверов. На вкладке CANmonitor должны быть видны посылки от контроллера при верном подключении CAN.
3. Запустив программу UniCON (нужно закрыть canwize, одновременно программы не работают) увидеть, что слева снизу окна программы написан «Статус: работа» (как на рисунке). Это обозначает, что программа успешно открыла usb-can переходник. Если в верхней левой части окна программы не появляется вкладки с устройством в сети, то проверьте корректность подключения CAN кабеля. Также откройте в программе UniCON Сервис-Параметры модуля связи. Убедитесь, что выбран модуль связи MARATHON, скорость передачи 125 kBit/s.
4. При подключении через USB-RS проверьте, установлены ли драйверы на микросхему CP2102, отображается ли в диспетчере устройств Windows виртуальный COM порт? Убедитесь, что это именно тот COM порт, который соответствует плате VectorCARD: отключите USB кабель, убедитесь, что виртуальный COM порт

пропал. Проверьте, верно ли задан номер COM порта в настройках UniCON, выбран ли USB-RS модуль связи. Параметры COM порта менять не следует: UniCON автоматически настраивает параметры COM порта при работе с ним.

5. Если вместо одного узла в сети с названием «ARM_MCP» появилось два с названием «Эмулятор», это значит в настройках UniCON вместо переходника связи CAN выбран эмулятор. Откройте в программе UniCON Сервис-Параметры модуля связи. Выберите модуль связи MARATHON, скорость передачи 125 kBit/s.
6. Если ничего из перечисленного не помогает, обратитесь в поддержку.

4.3. Подключение датчиков положения двигателя

Двигатель *Leadshine ACM601V36-01-1000* из комплектации «motor control-полная» оснащен двумя датчиками положения ротора: инкрементальным энкодером и ДПР на элементах Холла. Питание датчиков общее и может быть подключено к контактам 4 (+5В) и 5 («Земля») любого из разъемов J4 или J10 на плате инвертора DRV8301-HC-EVM Texas Instruments: инвертор питает датчики, дополнительного питания не требуется. Выходы датчика на элементах Холла подключаются к контактам 1-3 разъема J10: 1 – контакт HallU+, 2 – контакт HallV+, 3 – контакт HallW+. Энкодер подключается к контактам 1-3 разъема J4: 1 – контакт EA+, 2 – контакт EB+, 3 – контакт EZ+. Однако индексная метка энкодера в данном двигателе отсутствует, а соответствующий провод зарезервирован для других конфигураций.

Двигатель DT4260-24-005-04H-PT из комплектации «motor control-упрощенная» имеет только датчики Холла. Питание датчиков (черный минус, красный плюс) может быть подключено к контактам 4 (+5В) и 5 («Земля») любого из разъемов J4 или J10 на плате инвертора DRV8312-C2-KIT Texas Instruments. Выходы датчика на элементах Холла подключаются к контактам 1-3 разъема J10: 1 – контакт HallU+ (белый), 2 – контакт HallV+ (зеленый), 3 – контакт HallW+ (синий).

4.4. Настройка АЦП

Если АЦП настроено неверно, то большинство защит инвертора, не говоря уже об отображении величин, работать не будут. Каждая измеряемая величина представлена, как минимум, тремя объектами:

- Коэффициент усиления. Для каждой величины он задается в соответствующих единицах измерения. Его можно либо найти подбором, либо рассчитать: он равен тому, что должен измерять датчик, когда АЦП измеряет максимальное значение (на контроллер приходит максимальное измеряемое напряжение). Другими словами, если вход АЦП 0 – 3.3В, на него подается сигнал с датчика напряжения на звене постоянного тока и «0» на входе контроллера соответствует 0 В на ЗПТ, 3.3 вольта соответствует 66В на ЗПТ, то коэффициент равен 66.
- Смещение. Если измеряемая величина равна нулю, а реально измеряется не нулевое значение, можно установить смещение нулевого уровня сигнала.
- Измеряемое значение. Отображает мгновенное значение измеряемой величины. Так, если датчик измеряет фазный ток двигателя, то в процессе работы из-за быстрого изменения сигнала измеряемое значение будет «дрожать», что связано с периодичностью оцифровки мгновенной величины («стробоскопический эффект»). При отключенном инверторе в значениях токов фаз должны быть ненулевые величины, колеблющиеся в районе нуля. Они возникают из-за погрешностей измерения и косвенно свидетельствуют о работе АЦП.

Настройку АЦП необходимо начинать с напряжения на звене постоянного тока. Рассчитайте или подберите такой коэффициент усиления, чтобы отображаемое измеряемое значение соответствовало реальному напряжению на ЗПТ (параметры по умолчанию должны быть почти верными).

Настройку датчиков тока можно произвести разными способами. Проще всего ввести преобразователь в работу и по амперметру в фазах двигателя настроить коэффициенты. Так как без настроенных датчиков есть риск не зафиксировать максимально-токовую аварию, или,

наоборот, зафиксировать её ложно, желательно предварительно рассчитать коэффициенты усиления согласно описанию (см. выше). После этого произвести пуск двигателя (см. последовательность пуска в «4.7 Запуск привода») и наблюдать на экране ПК ток в фазах двигателя. Если значения в программе UniCON и на амперметре отличаются, подстроить коэффициенты. Не следует производить настройку на очень малых токах: постарайтесь создать ток не меньше ¼ номинала, иначе настройка может оказаться не очень точной. После окончания настройки сохраните все значения в энергонезависимую память, нажав кнопку «сохранить параметры». По умолчанию датчики токов для инвертора Texas Instruments настроены верно, однако может потребоваться подстройка нулевого уровня – отредактируйте смещения токов фаз, чтобы в среднем индицировался нулевой уровень тока, а затем дайте команду на сохранение параметра. Настройка смещения может выполняться автоматически состоянии останова. Для этого нужно включить модуль автосмещения АЦП – вписать «1» в параметр **Настройки СУ.АЦП авто смещ.**. Если автосмещение активно, то в остановленном состоянии привода смещение АЦП для токов фаз будет подстраиваться так, чтобы отображался нулевой ток. Во включенном состоянии смещения пересчитываться не будут.

4.5. Настройка модуля защит

Модуль защит позволяет отслеживать все критические ситуации и принимать меры для предотвращения аварии. Проверьте все уставки защит на соответствие ожидаемым значениям. В данном ПО предусмотрены защиты по максимальному току в фазе (например, на уровне 10 А), минимальному напряжению на ЗПТ (примерная уставка 12 В), максимальному напряжению на ЗПТ(примерная уставка 48 В) и максимальной скорости вращения (примерная уставка 3000 об/мин). Уставки могут быть изменены, однако следует чётко понимать, для чего это делается и к чему приведет. Несмотря на аппаратные защиты в инверторе Texas Instruments, его по неосторожности можно вывести из строя, например, «задрав» защиту по максимальному напряжению и создав рекуперативный режим электропривода.

4.6. Режимы работы (структуры системы управления)

В зависимости от технологических требований управление двигателем может осуществляться по-разному. Система управления принимает сигналы обратных связей (в большинстве случаев), рассчитывает и выдает управляющие воздействия на ключи инвертора, от которого питается двигатель. Набор обратных связей, состав и тип регуляторов зависят от того, какую величину и как требуется регулировать.

В данном ПО реализовано четыре основных вида систем управления:

1. Скалярная СУ обеспечивает вращение вектора напряжения с заданной частотой и амплитудой, определяемой в зависимости от частоты по закону U(f).
2. Частотно-токовая СУ обеспечивает вращение вектора тока с заданной частотой и поддержание его на заданном уровне амплитуды посредством регуляторов тока.
3. Система векторного управления имеет контуры тока и скорости. Внешний контур скорости выдает задание внутреннему контуру тока оси q, которая отвечает за момент двигателя. Вектор тока ориентируется по показаниям датчика положения ротора.
4. Система с контуром положения использует в основе векторную структуру управления и обеспечивает отработку приводом заданного положения. В итоге образуется трёхконтурная система подчиненного регулирования с последовательной коррекцией.

Все структуры систем управления реализованы из отдельных блоков в файле sm_ctrl.c, который организован в виде машины состояний (дискретного автомата). Каждой структуре соответствует определенное состояние дискретного автомата. Желаемое состояние дискретного автомата, в которое он переходит после запуска привода, задается в переменной (**Настройки СУ.режим работы**), что на в ПО соответствует переменной sm_ctrl.state.

Режим скалярного управления (**Настройки СУ.режим работы** = 3)

Служебный режим, предназначенный для проверки преобразователя при работе на асинхронный двигатель, резистивную нагрузку, холостом ходе и т.п. Используется в процессе тести-

рования и наладки преобразователя. Обеспечивает традиционное скалярное управление АД/СД с регулированием напряжения по заранее заданной кривой *U/f*. Настройка кривой ограничена двумя опорными точками через следующие параметры группы **КриваяU-f**: **Нулевая опорная частота**, **Нулевое опорное напряжение**, **Первая опорная частота**, **Первое опорное напряжение**. Разгон производится от задатчика интенсивности ЗИ (подробнее рассмотрен в п. 5.3). Структура скалярной СУ представлена на Рис. 3

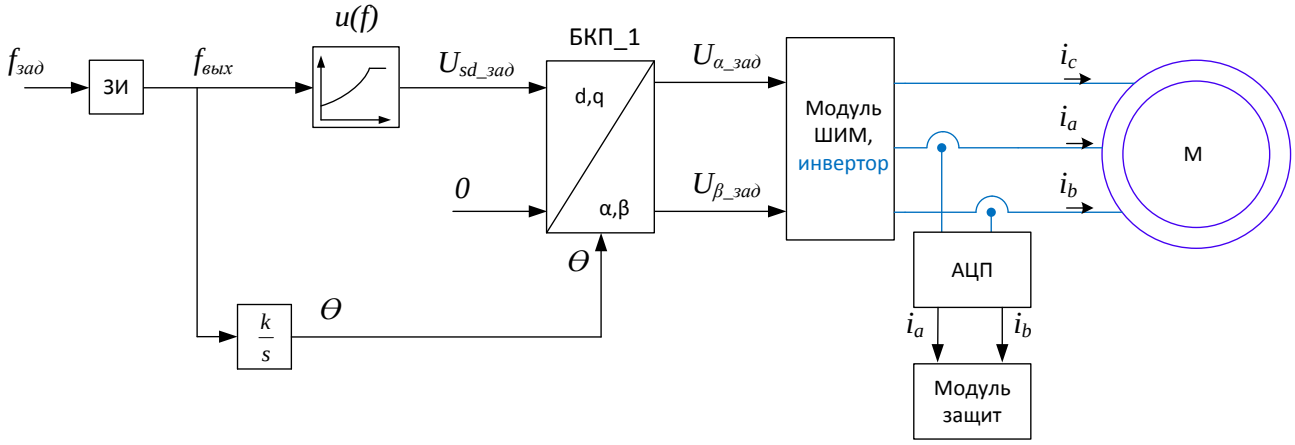


Рис. 3 Структура системы скалярного управления

Режим частотно-токового управления (**Настройки СУ.режим работы** = 4)
Служебный режим, в котором происходит вращение вектора тока статора (поля двигателя) с фиксированной амплитудой и нарастающей частотой. Система является разомкнутой по положению ротора, т.е. угловое положение вектора тока не зависит от реального положения ротора, а скорость вращения определяется задатчиком интенсивности. Ротор машины при этом должен вращаться синхронно с полем статора. При неверной настройке амплитуды тока и темпа задатчика интенсивности возможны колебательные процессы, вплоть до «выпадения» машины из синхронизма. Этот режим используется преимущественно при наладке привода, настройке регуляторов тока, проверке работы привода без обратной связи по положению и т.п.

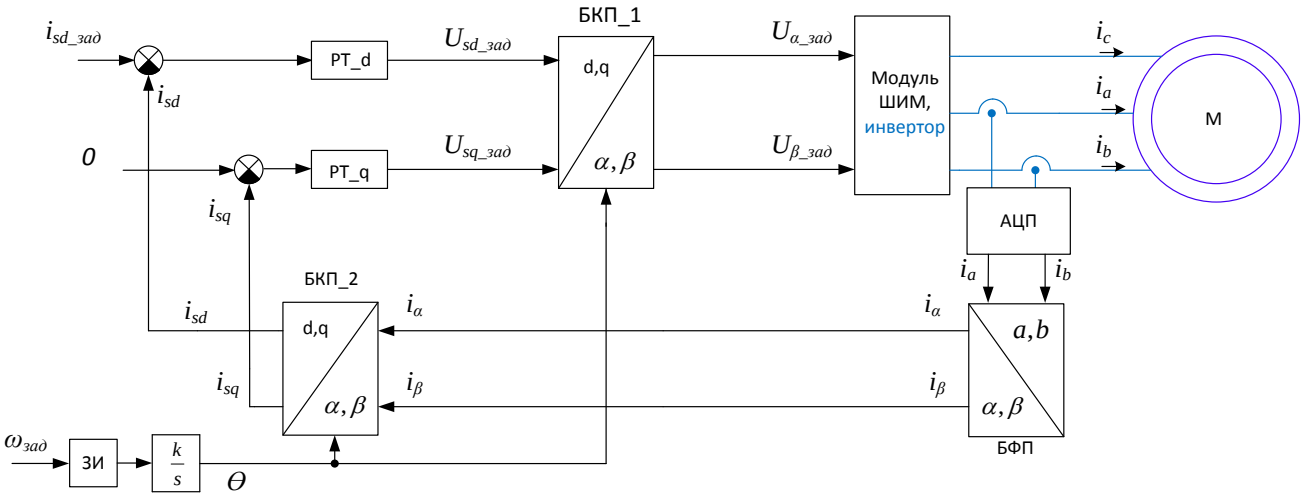


Рис. 4 Структура системы частотно-токового управления

Задание частоты вращения вектора тока двигателя **ЗАДАНИЯ.Скорость** поступает через блок ограничения скорости на задатчик интенсивности ЗИ, который обеспечивает формирование линейной тахограммы разгона двигателя. С его выхода сигнал скорости поступает на вход интегрирующего элемента, на выходе которого формируется задание углового положения вектора тока θ.

Измеренные значения токов двух фаз двигателя i^*_a, i^*_b поступают на вход блока фазных преобразований БФП. Его задача - преобразовать токи из трехфазной системы координат в двух-

фазную декартову систему координат α, β. Ток фазы c вычисляется по первому закону Кирхгофа (обмотки двигателя соединены в звезду).

Токи фаз в координатах α, β (i^*_α, i^*_β) поступают на блок координатных преобразований БКП_2, обеспечивающий преобразование токов из неподвижной системы координат к вращающейся системе координат d,q. Положение оси d будет определяться текущим значением угла θ. Токи i^*_q, i^*_d поступают на ПИ-регуляторы токов, выходы которых после обратного координатного преобразования в блоке БКП_1 (поворот в обратную сторону на угол θ) заводятся на инвертор напряжения через блок ограничения минимальной скважности.

Задание на амплитуду тока (**ЗАДАНИЯ.Ток удержания**) подается на один из регуляторов, в данном случае на регулятор тока по оси d. Таким образом, регуляторы поддерживают постоянный ток согласно заданию $i^*_{sd_зад}$, но так как система координат d, q, в которой работают регуляторы, вращается, то вращается и результирующий вектор тока в осях a, b, c, связанных с фазами двигателя. Работа регуляторов тока во вращающихся осях позволяет исключить статическую ошибку воспроизведения амплитуды трехфазного синусоидального тока, которая присутствовала бы, если бы регуляторы токов работали в осях α, β и на их вход подавалось бы синусоидальное задание токов.

Необходимо заметить, что ортогональные оси d,q в данном случае не связаны с фактическим положением ротора, как это обеспечивается при векторном управлении, а лишь поворачиваются с заданной скоростью независимо от состояния машины.

Результатом работы такой системы должен стать синусоидальный ток в фазах двигателя, практически не зависящий от различных нелинейностей, влияния "мертвого" времени инвертора, колебаний механической части и напряжения в звене постоянного тока.

Режим удержания (**Настройки СУ.режим работы** = 2)

Этот режим соответствует режиму фиксации двигателя под током и необходим перед переходом в режим частотно-токового управления (вращение вектора тока с фиксированной амплитудой и нарастающей частотой) для того, чтобы ротор двигателя предварительно повернулся и зафиксировался в определенном механическом состоянии. При этом последующее начало вращения вектора тока в режиме частотно-токового управления выполняется с заданной фазы и сопровождается меньшими колебаниями ротора.

Структура системы управления в данном режиме аналогична структуре частотно-токового управления за одним исключением: угол положения вектора тока Θ неизменен и задается нулевым.

Режим векторного управления (**Настройки СУ.режим работы** = 5 или 6)

Настройки СУ.режим работы = 5 соответствует векторной системе с датчиком пооженя на элементах Холла, а **Настройки СУ.режим работы** = 6 – с инкрементальным энкодером.

Основной режим работы привода. Для реализации векторного управления необходимо определять угловое положение вектора потокосцепления ротора двигателя, которое связано с положением ротора и определяется по датчику положения ротора (ДПР). Ортогональные подвижные оси d,q направляются так, что ось d совпадает с направлением вектора потокосцепления ротора. Тогда ток статора по оси q, контролируемый напряжением по этой оси, определяет момент, развиваемый двигателем. Ток по оси d (возбуждение) в данном случае равен нулю, так как возбуждение у машины от постоянных магнитов. В этом режиме работы СДПМ подобен двигателю постоянного тока, так как амплитуда тока по оси q напрямую задаёт момент двигателя (аналог якорной обмотки коллекторного двигателя постоянного тока).

На Рис. 5 показана структура системы датчикового векторного управления. Задание частоты **ЗАДАНИЯ.Скорость** ($\omega^*_{зад}$ на Рис. 5) поступает на задатчик интенсивности ЗИ, который обеспечивает линейный разгон. С его выхода задание частоты поступает на сумматор ПИД регулятора скорости, в который также заводится обратная связь по скорости с датчика положения ротора. Выходом регулятора скорости является задание на ток по оси q. Это значение пропорционально электромагнитному моменту машины. Оно поступает на сумматор ПИ-регулятора тока по оси q, выходом которого является задание напряжения по оси q. Аналогично функционирует регулятор тока по ортогональной оси d, задание на входе которого нулевое. Это связано с тем, что ось d ответственна за возбуждение машины, а оно в СДПМ внешнее.

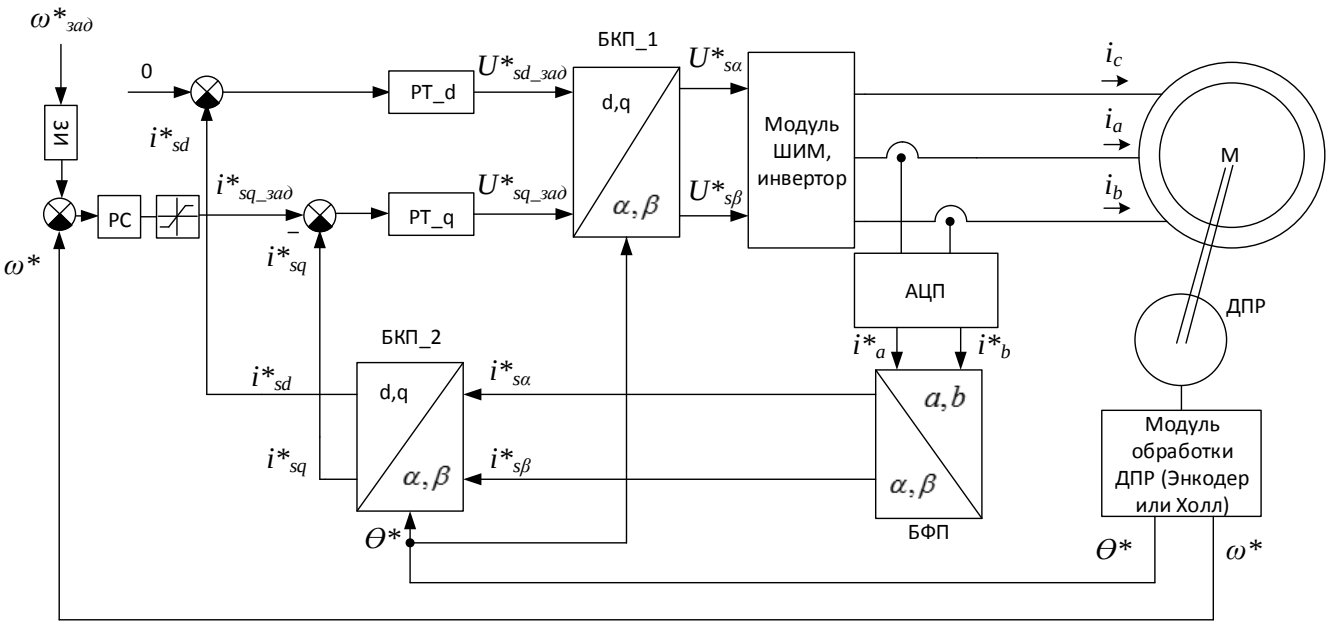


Рис. 5 Структура системы векторного управления

Напряжения с регуляторов токов поступают на блок обратного координатного преобразования (поворот системы координат в обратную сторону на угол θ) БКП_1, а затем заводятся на инвертор напряжения. Измеренные значения токов фаз двигателя i^*_a, i^*_b поступают на вход блока фазных преобразований БФП. В этом блоке они преобразуются из трехфазной системы координат в двухфазную декартовую систему координат α, β. Ток фазы c вычисляется по первому закону Кирхгофа (обмотки двигателя соединены в звезду). Токи i^*_α, i^*_β поступают на блок координатных преобразований БКП_2, поворачивающих систему координат на угол θ. На выходе получаем фактические значения токов в системе координат d,q, ориентированной по потокосцеплению ротора. Эти значения используются в качестве сигналов обратных связей на входах регуляторов токов по осям d,q.

Следует отметить, что выход регулятора скорости (задание на ток/момент) имеет ограничения: **Рег. скорости.Максимум выхода** и **Рег. скорости.Минимум выхода**. При этом по умолчанию минимум установлен нулем, чтобы исключить отрицательный момент двигателя (условие отсутствия рекуперации энергии при положительной частоте вращения), а максимум устанавливается равным заданию тока для векторного управления **ЗАДАНИЯ.Макс. ток ВУ**. Установка минимума и максимума происходит автоматически.

Существует возможность включения режима рекуперации установкой флага **Настройки СУ.Рекуперация**. В этом случае минимум регулятора скорости получит значение минус **ЗАДАНИЯ.Макс. ток ВУ**.

Осторожно! Этот режим приведет к повышению напряжения в звене постоянного тока, при отсутствии рассеивания энергии торможения в резисторе сработает защита по превышению напряжения звена постоянного тока.

Внимание! Перед запуском этого режима необходимо произвести настройку ДПР. Либо вручную, как описано в главе 4.7, либо воспользоваться *режимом автонастройки смещений ДПР* (**Настройки СУ.режим работы** = 29).

Примечание. Если читатель плохо знаком с принципами векторного управления электродвигателями, то рекомендуется к прочтению вводная статья «Векторное управление электродвигателем «на пальцах», доступная по ссылке https://geektimes.ru/company/npf_vektor/blog/269486/ .

Режим позиционирования (**Настройки СУ.режим работы** = 10)

Структура позиционирования (Рис. 6) при помощи дополнительного внешнего контура положения на основе структуры векторного управления, описанной выше, реализует поддержание приводом заданного пользователем положения ротора $\theta^*_{зад}$. Регулятор положения (РП) имеет

тип ПИД, хотя дифференциальная часть может быть опущена. В качестве регулятора скорости (РС) рекомендуется П-регулятор (Ки и Кд необходимо обнулить).

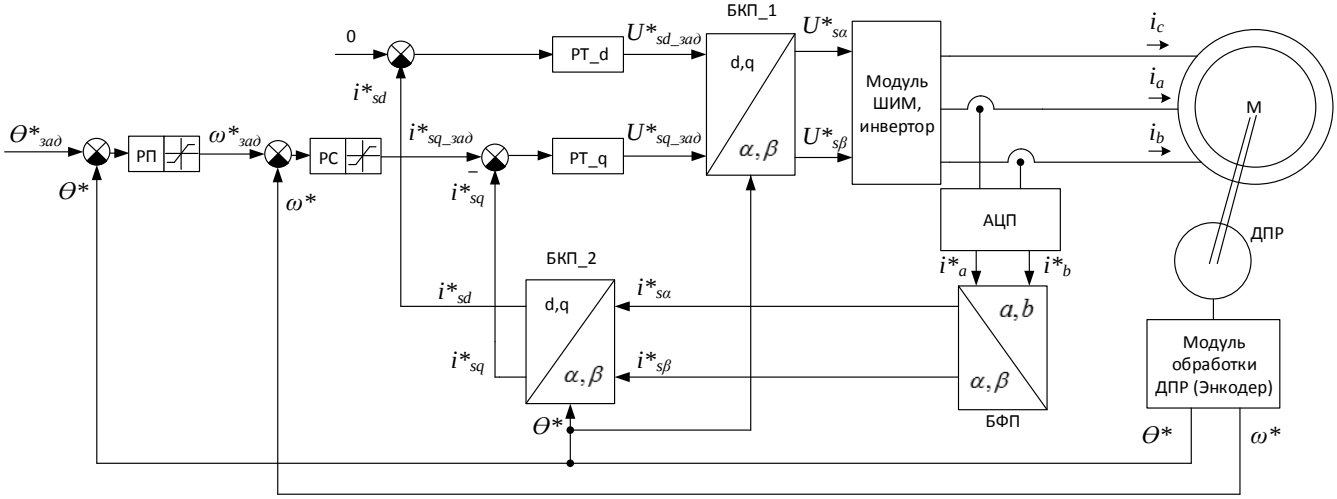


Рис. 6 Структура для поддержания заданного положения на основе векторного управления: трехконтурная система подчиненного регулирования

Структура работает при использовании ДПР типа «инкрементальный энкодер», так как точности ДПР на элементах Холла недостаточно для данного режима. Перед запуском структуры необходимо настроить ДПР, как описано в главе 4.7, либо воспользоваться *режимом автонастройки смещений ДПР* (**Настройки СУ.режим работы** = 29). Пользователь задает желаемое положение вала ротора в электрических градусах в параметре **ЗАДАНИЯ.theta_elec**. Параметр **ЗАДАНИЯ.Скорость** в данном режиме ограничивает максимальную частоту вращения (выход регулятора положения). Параметр **ЗАДАНИЯ.Макс. ток ВУ** в данном режиме ограничивает максимальный моментообразующий ток статора (выход регулятора скорости). В качестве обратной связи по контуру положения используется электрическое угловое положение, учитывающее совершенное валом двигателя число оборотов.

Режим автонастройки смещений ДПР (**Настройки СУ.режим работы** = 29)

Режим работы по структуре полностью аналогичен режиму 4 (частотно-токовое управление), однако в данный режим добавлена логика подстройки смещений датчиков положения ротора (параметры **ДПР-Холл.СмещУгол** и **ДПР-Энкодер. СмещУгол**). Смещения подстраиваются таким образом, чтобы максимуму тока фазы А соответствовало нулевое показание обоих датчиков положения. Этот режим корректно настраивает датчики положения ротора при условии отсутствия нагрузки на валу и относительно невысокой частоте вращения.

4.7. Запуск привода (Быстрый старт)

У преобразователя существует несколько различных структур управления. Например, классическая U/f для асинхронного двигателя, частотно-токовая система управления, реализующая источник тока и т.п. Желаемая структура управления задается в параметре **Настройки СУ.режим работы**. Режиму работы «останов», т.е. отсутствию выбранной структуры соответствует число 0. Режиму работы U/f соответствует номер 3. Перед запуском двигателя необходимо убедиться, что в данном параметре записано это число.

Перед первым пуском установите максимальный ток фаз в модуле защит (**Защиты.Imax**) на сравнительно низкий уровень: около того значения тока, который вы ожидаете увидеть, т.е. на границе срабатывания. Это позволит проверить срабатывание защит, убедиться, что токи соответствуют ожидаемым значениям и произвести более раннее отключение, если что-то пойдет не по плану. В дальнейшем, при корректной работе можно вернуть уставку на прежний уровень.

Убедившись, что АЦП работоспособно (в напряжении ЗПТ адекватное значение, значения токов фаз колеблются вблизи нуля), в модуле защит заданы адекватные значения, задана верная

структура управления и нет проблем в аппаратной части, можно дать команду пуска. Примерные настройки для тестового запуска двигателя *ACM601V36-01-1000* при питании инвертора от источника 24В следующие: **Защиты.Максимальный ток** = 5 А, **ШИМ.Частота ШИМ** = 20 кГц, **Кривая U-f.Нулевая опорная частота** = 0 Гц, **Кривая U-f.Нулевое опорное напряжение** = 0,7 В, **Кривая U-f.Первая опорная частота** = 3000 об/мин, **Кривая U-f.Первое опорное напряжение** = 5 В. Для запуска задайте частоту вращения 2-5 об/мин (**Задания.Скорость**) и нажмите кнопку Пуск на панели управления в правой части окна UniCON. JTAG эмулятор при этом допустимо оставлять подключенным с запущенной сессией отладки, однако желательно не останавливать микроконтроллер во время работы ШИМ.

Чтобы добавить параметр в осциллограф, нужно щелкнуть по параметру правой кнопкой, выбрать пункт «Добавить в осциллограф» и любой из четырех каналов.

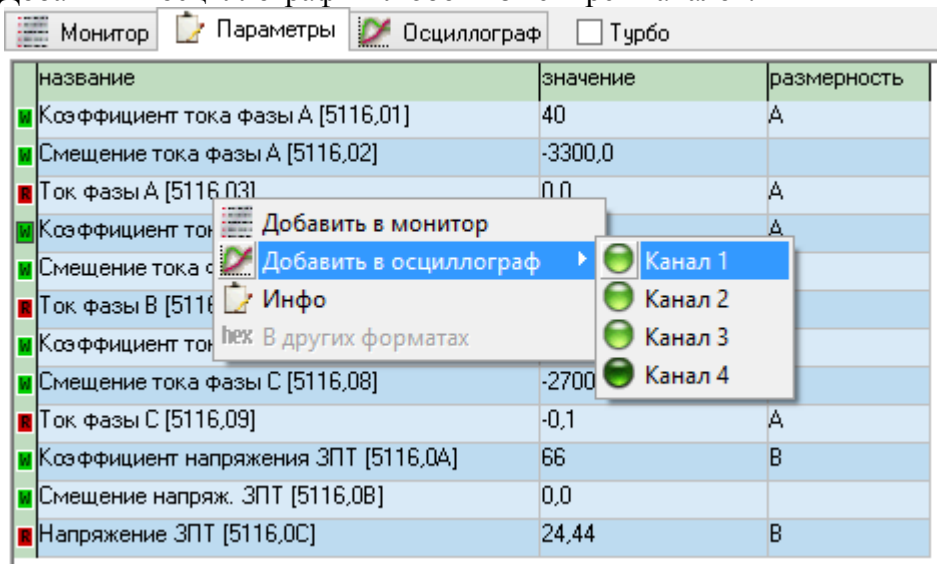


Рис. 7 Добавление параметра в осциллограф

Примерные осциллограммы токов фаз (**АЦП.Ток фазы А, В, С**) и заданного фазного напряжения (**ШИМ.UаЗад**) для таких настроек показаны на Рис. 8.

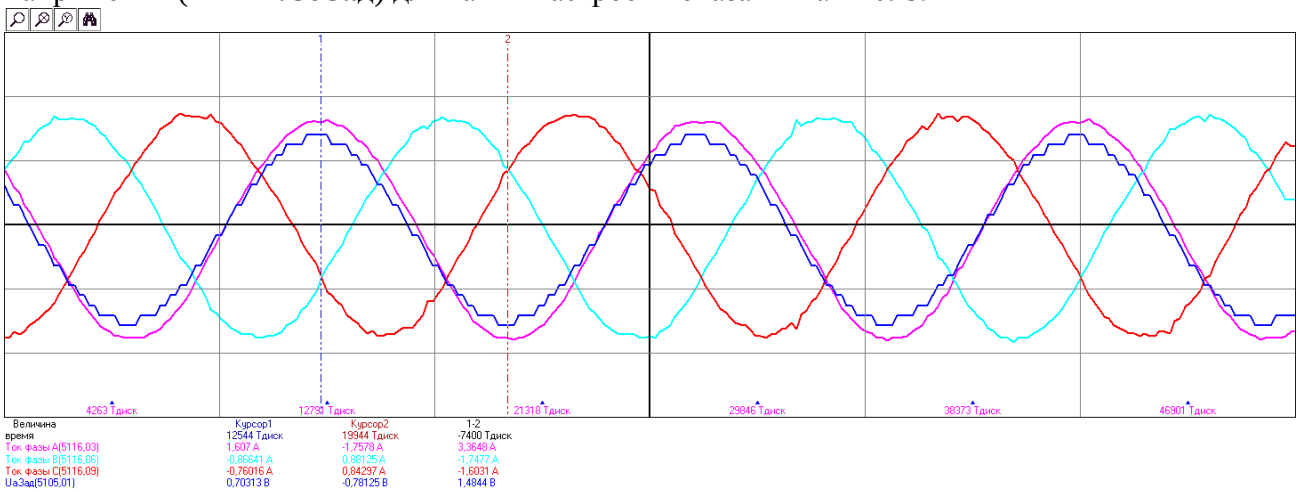


Рис. 8 Осциллограмма тестового запуска

Если привод работает, а осциллограммы токов похожи на приведенные, подайте команду останова (кнопка Стоп на панели управления в правой части окна UniCON).

Теперь перейдем к настройке регуляторов тока. Для начала выберем режим удержания – в **Настройки СУ.режим работы** введем значение «2». В осциллограф добавим токи фаз из группы АЦП. Также может быть интересно наблюдать заданное напряжение (**ШИМ.UаЗад**).

Для правильного отображения процесса пуска в параметр **dlog.StartBitL** вводим значение «00000010», в **dlog.StartBitH** – «00000000», в параметре **dlog.Записывать сначала** выбираем «Да». Параметры **dlog.StartBitL** и **dlog.StartBitH** задают триггеры срабатывания осциллографа, в данном случае по событию перехода машины состояний модуля sm_ctrl в состояние 2 (структура

постоянного тока). Подробнее о работе модуля осциллографа читайте в руководстве пользователя к программе UniCON, а также в главе 5.4.

Нажимаем кнопку «Пуск» на панели управления в правой части окна UniCON, заходим во вкладку «Осциллограф» и нажимаем кнопку «Обновить» (Рис. 9).

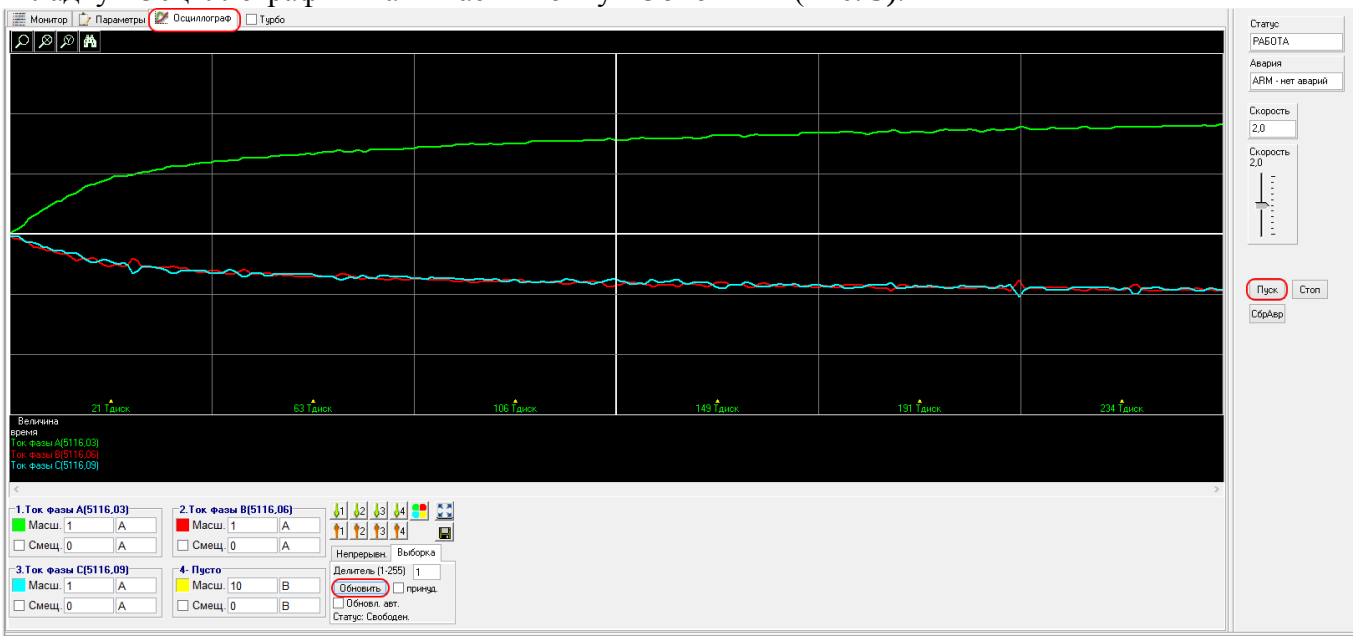


Рис. 9 Окно осциллографа

Если при нажатии кнопки «Обновить» появляется окно Рис. 10, значит по одному из активных триггеров произошел захват осциллограммы и её можно загрузить. Однако если это первый запуск осциллографа после изменения параметров осциллографирования, то необходимо отказаться от загрузки осциллограммы, так как там «мусор». Осциллографирование по триггеру может быть полезно также, например, при срабатывании защиты («1» в младшем разряде **dlog.StartBitH** активирует триггер по переходу привода в состояние «Стоп») можно посмотреть форму интересующих величин во время аварийного события. Чтобы увидеть «предысторию» срабатывания триггера, следует в **dlog.Записывать сначала** выбрать «Нет».

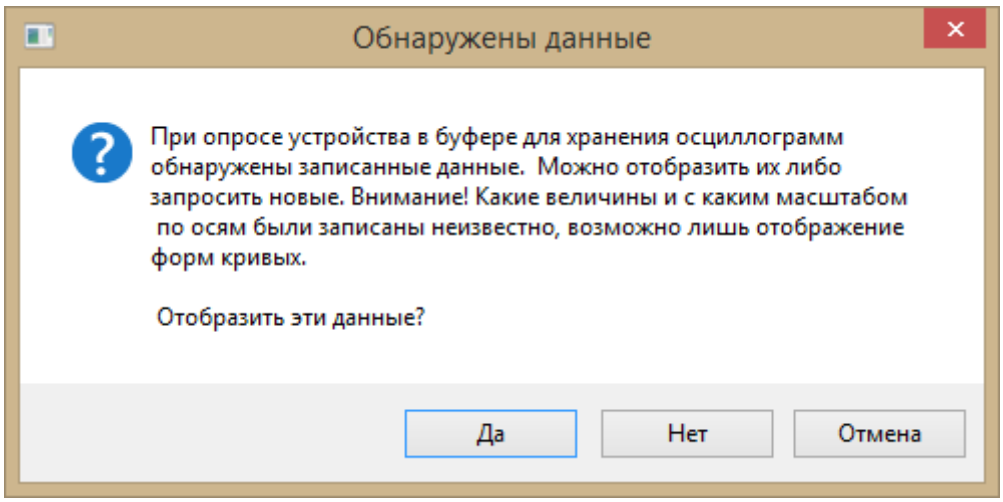


Рис. 10 Диалоговое окно загрузки захваченных осциллограмм

Для отображения осциллограммы запуска в появившемся окне нужно щелкнуть по кнопке «Да». Переходной процесс при ненастроенных регуляторах может выглядеть подобно Рис. 11.

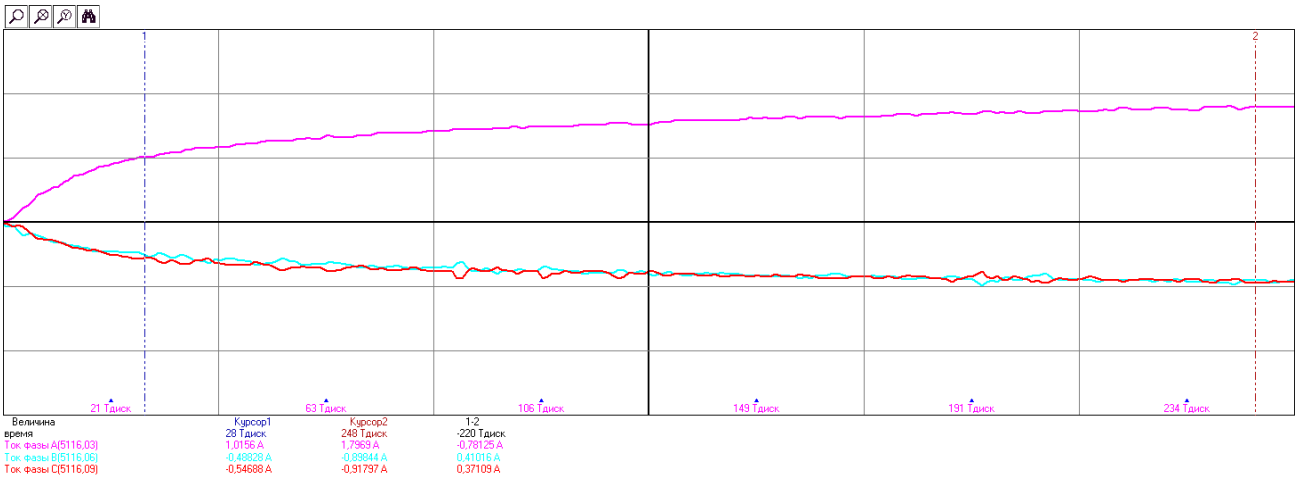


Рис. 11 Переходной процесс пуска при ненастроенных регуляторах

Процесс подбора коэффициентов регуляторов тока рассмотрен на рисунках ниже. Сначала интегральный коэффициент обнуляется, а пропорциональный постепенно повышается до появления колебательности переходного процесса. Коэффициенты для обоих регуляторов тока (Iq,Id) должны быть равны и изменяются одновременно. Частота ШИМ в данном опыте была задана 20кГц (параметр **ШИМ.Частота ШИМ**), тип ШИМ – векторная ШИМ (параметр **ШИМ.Тип ШИМ** равен нулю).

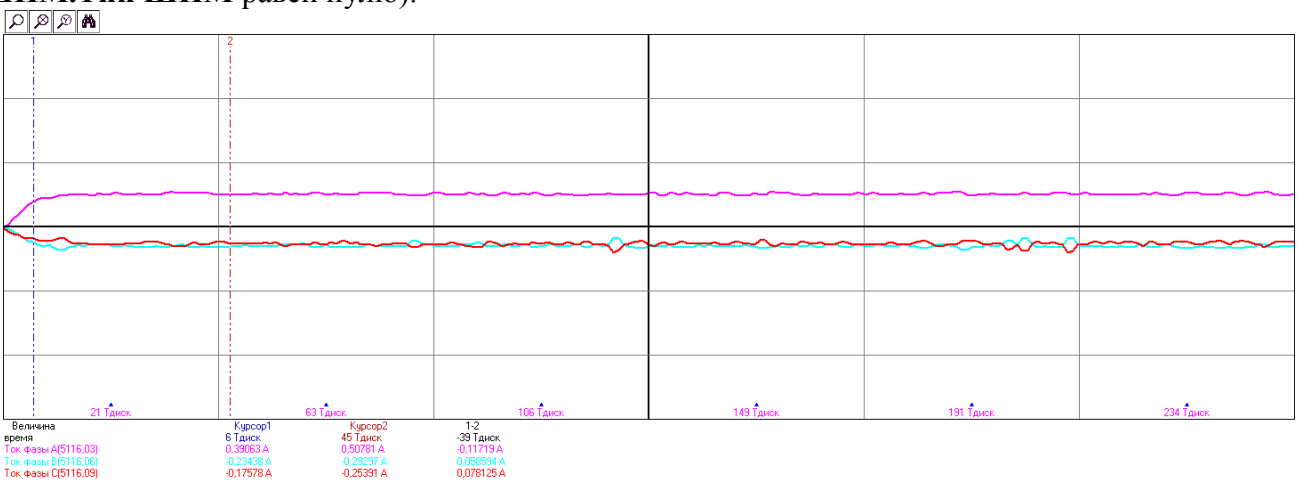


Рис. 12 Кп=0,04

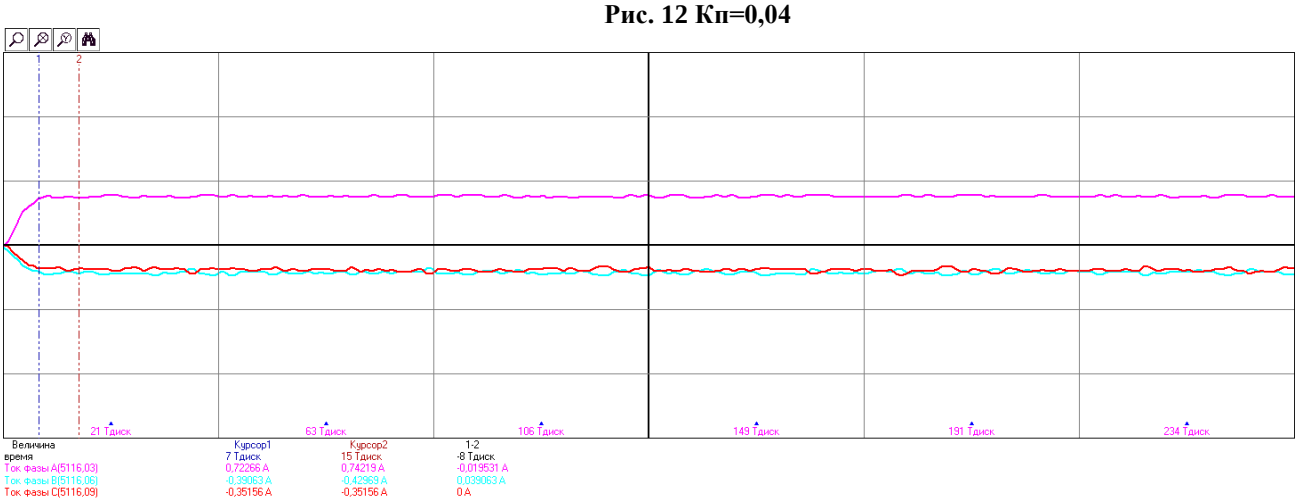


Рис. 13 Кп=0,07

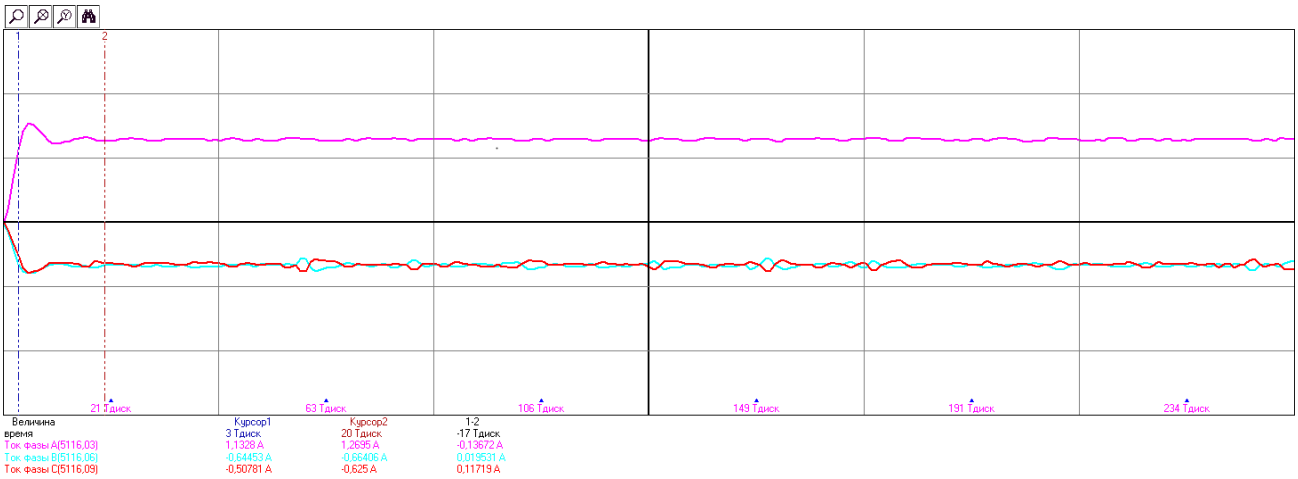


Рис. 14 Кп=0,2

Для снижения колебательности немного уменьшим Кп.

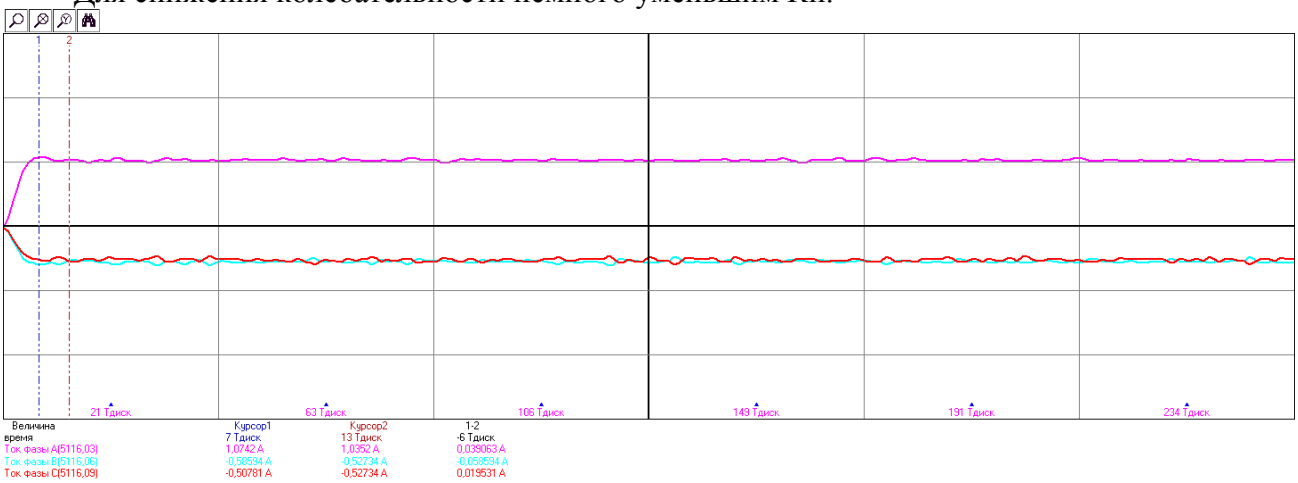


Рис. 15 Кп=0,12

Как видно, в контуре тока есть статическая ошибка (задание равно 2А, а фактическое значение меньше – около 1А). Устранить её поможет интегральная часть регулятора. Подбор Ки представлен на рисунках далее.

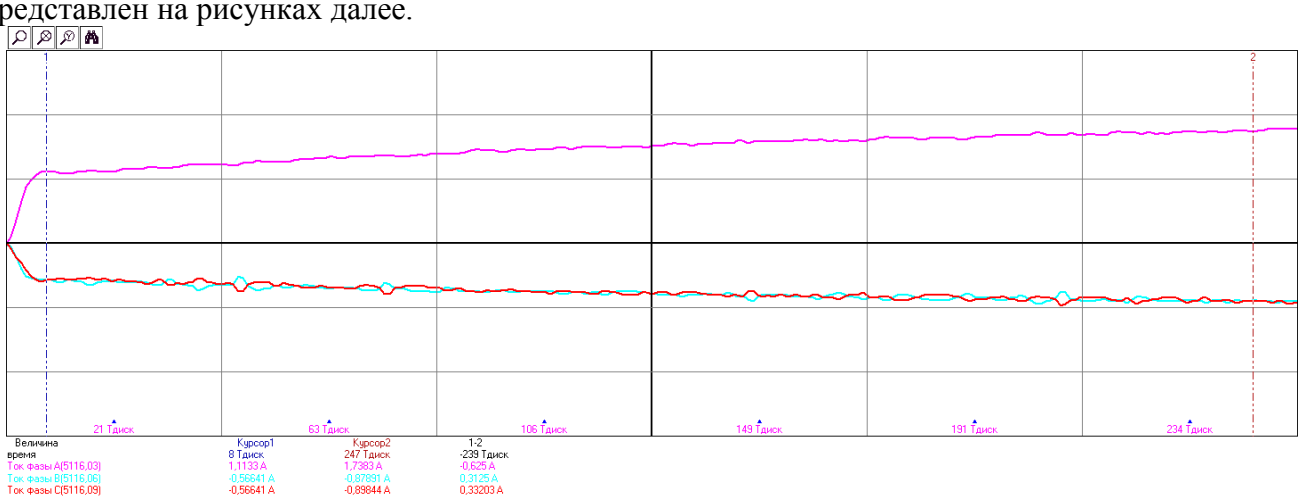


Рис. 16 Ки=0,01

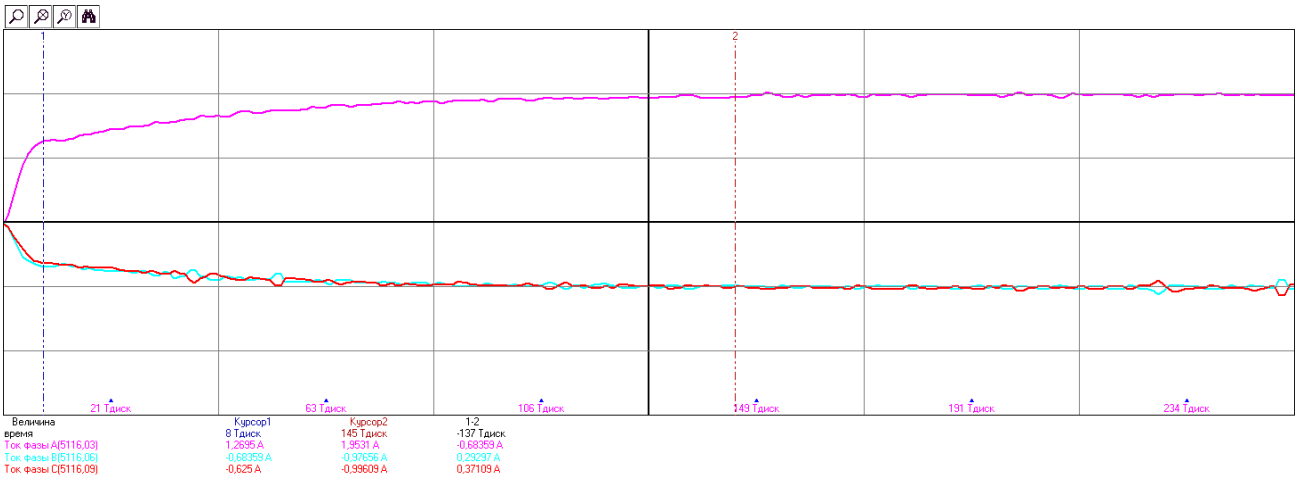


Рис. 17 Ки=0,04

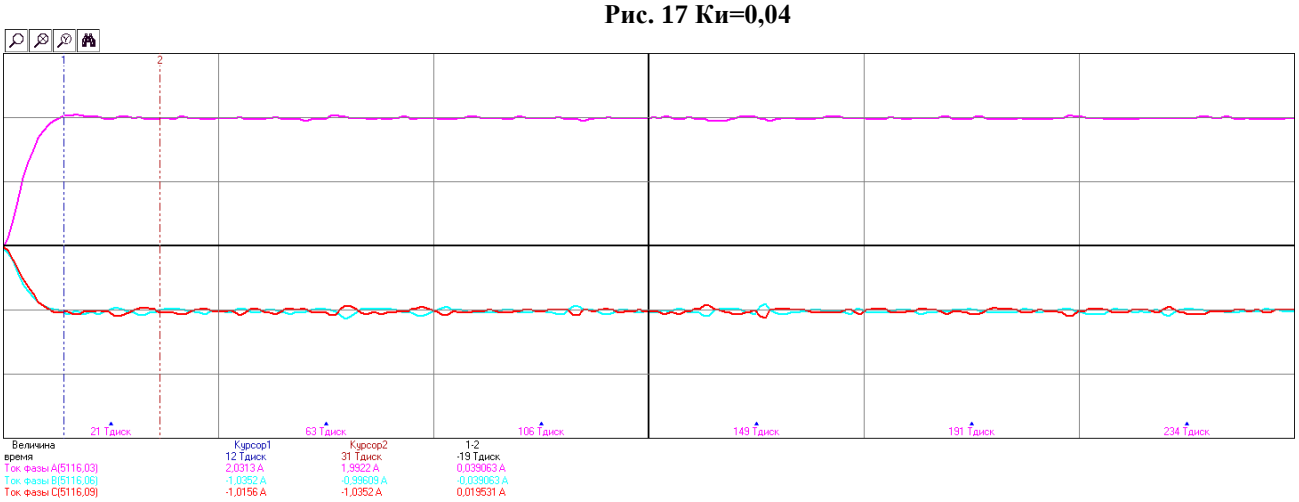


Рис. 18 Ки=0,16

Регуляторы тока настроены. Для проверки вращения двигателя запустим его в частотно-токовой системе управления (вращение вектора тока с заданной амплитудой), **Настройки СУ.режим работы** = «4». Двигатель должен вращаться со скоростью (частотой вращения), заданной в **ЗАДАНИЯ.Скорость**, форма токов – синусоидальная (Рис. 19). Начать работу следует с небольшой частоты вращения, например, 2-5 об/мин.

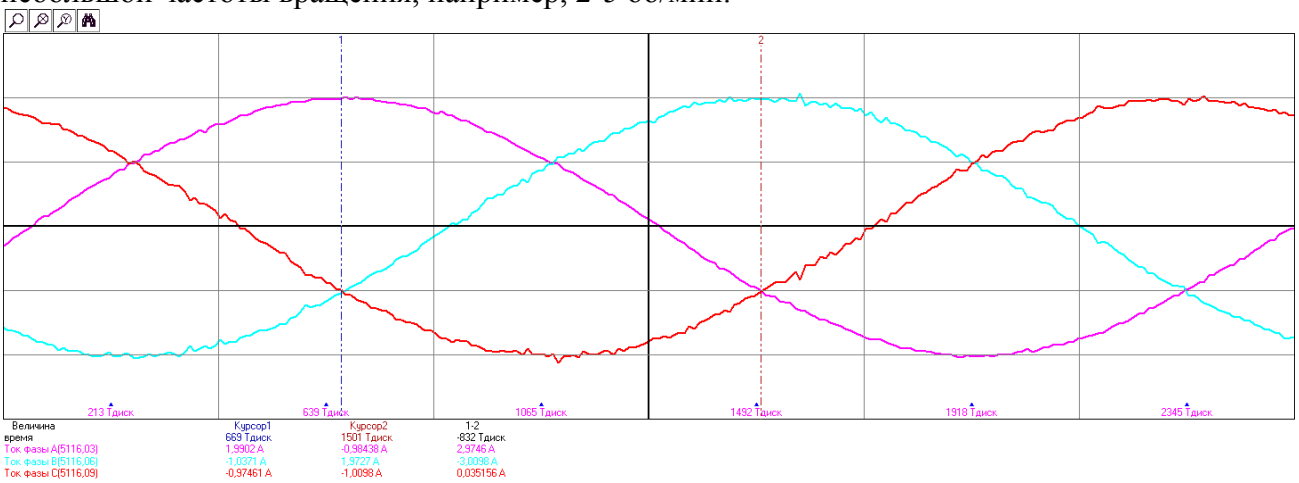


Рис. 19 Форма токов

Перед тем как запустить векторную СУ, нужно проверить работу датчиков положения. Для этого следует запустить двигатель в частотно-токовой системе управления (режим работы 4), задать небольшую скорость, к примеру, 2 об/мин и наблюдать осциллограммы сигналов токов и сигналов датчиков положения: **ДПР-Холл.ЭлУгол** (с датчика на элементах Холла) и **ДПР-Энкодер.Угол. электр.** (угловое положение с квадратурного энкодера в электрических граду-

сах). Обратите внимание, что энкодер доступен не во всех комплектациях отладочных комплектов.

Корректный сигнал с ДПР для датчика на элементах Холла представлен на Рис. 20, а для инкрементального энкодера – на Рис. 21. Тип датчика выбирается в зависимости от выбранной структуры системы управления в параметре **Базов. парам.Тип датчика положения**. Можно активировать оба датчика одновременно, но это увеличивает нагрузку на вычислительные мощности микроконтроллера и в некоторых случаях может привести к выходу времени обработки прерывания за границу периода этого прерывания (сработает защита).

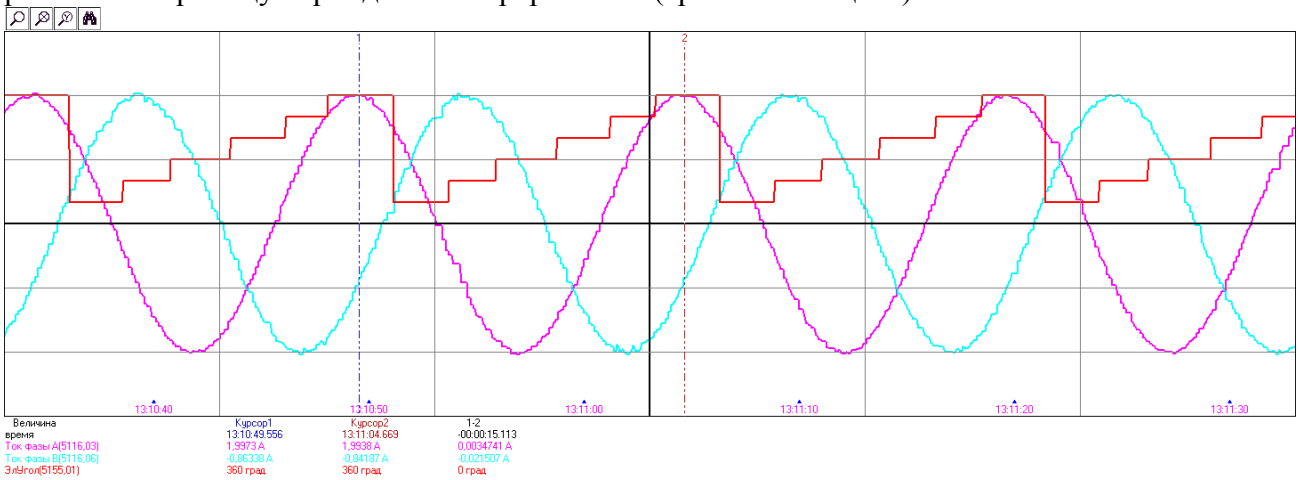


Рис. 20 Осциллограмма сигнала ДПР на элементах Холла

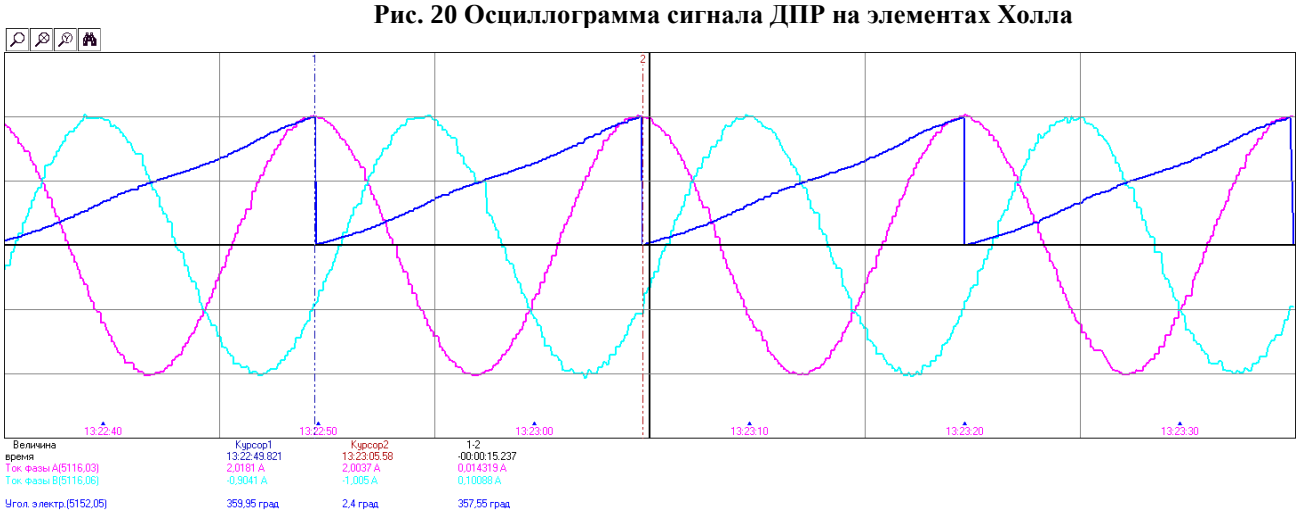


Рис. 21 Осциллограмма сигнала инкрементального энкодера

Сигнал с ДПР любого типа должен иметь возрастающий характер, его период должен совпадать с периодом тока в фазах, а обнуление сигнала ДПР должно происходить в районе максимума тока в фазе А. В этом случае датчик считается настроенным. Если сигнал ДПР отсутствует или «совсем не похож» на представленные, проверьте правильность подключения проводов ДПР к разъемам платы инвертора. Если характер сигнала ДПР спадающий, а не нарастающий, поменяйте местами любые две фазы питания двигателя или датчика положения. Если сигнал по одному датчику нарастает, а по другому спадает – проверьте или поменяйте сигналы одного из датчиков местами. Если период сигнала ДПР отличается от периода тока, проверьте **параметр Базов. парам.Число пар полюсов**: если период ДПР больше, чем тока, число пар полюсов следует увеличить, в противном случае – уменьшить. После изменения числа пар полюсов нужно сохранить параметры (кнопка «Сохранить параметры») и перезагрузить отладочную плату, сбросив питание или вписав «1» в параметр **Настройки СУ.Перезагрузка**. Если обнуление сигнала ДПР не совпадает по времени с максимумом тока в фазе А, настройте смещение датчика положения параметрами **ДПР-Холл.СмещУгол** и **ДПР-Энкодер. СмещУгол** так, чтобы эти два события происходили одновременно. Для инкрементального энкодера такую настройку смещения необходимо делать каждый раз при включении комплекта в работу, так как процедура причаливания и сигнал индексной метки в демонстрационном ПО не используются. Для ДПР на

элементах Холла настройку можно провести один раз для данного двигателя и сохранить параметры в энергонезависимую пользовательскую память устройства.

Настройка датчиков положения «вручную» полезна как обучающий пример. Чтобы экономить время при каждом запуске, можно запускать автоматическую настройку. Она реализована в виде отдельного режима работы. Для запуска автоматической настройки смещений обоих ДПР необходимо задать **Настройки СУ.режим работы** = 29 и запустить привод без нагрузки на частоте вращения 2-5 об/мин. Структура системы управления в режиме 29 аналогична частотно-токовой структуре управления (режим 4), однако в этом состоянии добавлена логика автоматической подстройки смещений обоих типов датчиков. Необходимо запустить этот режим работы и подождать около минуты, смещения датчиков настроятся автоматически. Для визуализации процесса автонастройки можно добавить параметры смещений датчиков в осциллограф (параметры **ДПР-Холл.СмещУгол** и **ДПР-Энкодер. СмещУгол**).

Теперь можно запускать двигатель в векторной системе управления с регулятором скорости (**Настройки СУ.режим работы** = 5 или 6). Следует проверить соответствие выбранного типа ДПР в параметре **Базов. парам.Тип датчика положения** и режима работы. **Настройки СУ.режим работы** = 5 – для ДПР на элементах Холла, а **Настройки СУ.режим работы** = 6 – для инкрементального энкодера.

Векторная СУ с настройками по умолчанию из-за малого момента инерции двигателя неустойчиво работает на низких скоростях (менее 500 об/мин), поэтому задавать следует бóльшие значения. Рекомендуемый диапазон скоростей – 500..2000 об/мин.

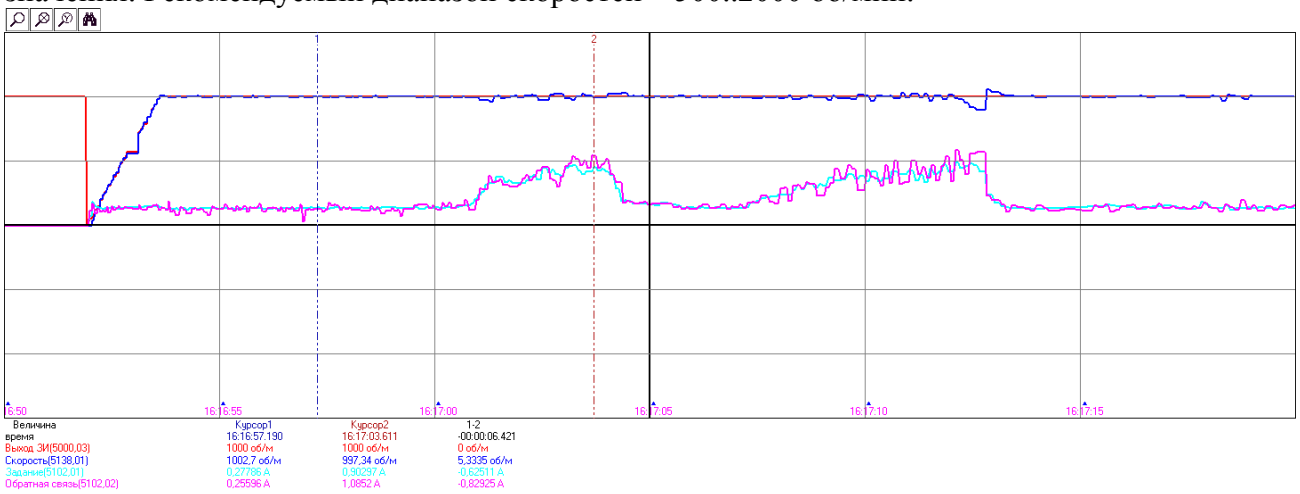


Рис. 22 Осциллограмма работы векторной системы управления

На Рис. 22 показана работа векторной системы управления. Выход задатчика интенсивности является заданием скорости. Задание и обратная связь (голубой и розовый графики) - задание регулятора тока по оси q и фактическое значение тока по оси q. Показан пуск и приложение нагрузки к двигателю. Как видно из осциллограммы, при повышении момента нагрузки ток по оси q возрастает, чтобы поддержать задание скорости (частоты вращения). В данном опыте разрешенный максимальный ток векторной системы управления (ограничение выхода регулятора скорости) был задан равным 1А (параметр Задания.Макс. ток ВУ).

Регулятор скорости векторной СУ, как и регуляторы тока, пропорционально-интегральный. Его настройки так же можно изменять для получения требуемой динамики и точности регулирования скорости.

Для запуска структуры, замкнутой по положению (**Настройки СУ.режим работы** = 10), рекомендуется обнулить параметр **ДПР-Энкодер.Число оборотов**, если он не нулевой, чтобы привод не начал «отматывать их назад». Параметры регулятора положения по умолчанию настроены на достаточно жесткое поддержание положения. Необходимо качественно настроить регуляторы токов, чтобы исключить возможные автоколебания. Также рекомендуется обнулить интегральную часть (Ки) в регуляторе скорости. С поставляемым в комплекте двигателем *АСМ601V36-01-1000* такие настройки обеспечат достаточно качественное позиционирование. Первый раз рекомендуется запускать привод с небольшим разрешенным током (**ЗАДА-**

НИЯ.Макс. ток ВУ), равным, например, 1А, а также ограничить частоту вращения на 1000об/мин. В параметре **ЗАДАНИЯ.theta_elec** пользователем задается желаемое положение в электрических градусах. Качественная настройка контура положения достаточно сложная процедура и выходит за рамки данного руководства. Пользователю предлагается изучить данный вопрос самостоятельно, экспериментируя с разными параметрами регулятора скорости и положения на данном комплекте. Простейшее правило настройки систем подчиненного регулирования такое – начиная со вложенного контура повышать коэффициенты регуляторов до тех пор, пока не начнутся колебания, затем уменьшить коэффициенты в два раза и перейти к вышестоящему контуру.

После завершения настройки можно сохранить все параметры привода в энергонезависимую память. Это делается в программе UniCON путем нажатия на кнопку «Сохранить параметры», расположенной справа сверху. При этом значения всех параметров CANopen, доступных на запись, сохранятся в пользовательской flash памяти микроконтроллера и будут в дальнейшем восстанавливаться при перезагрузке микроконтроллера в процедуре инициализации драйвера CANopen.

5. Модули ПО и их настройка

5.1. ПИД регулятор

ПИД-регулятор устроен по схеме, показанной на рисунке:

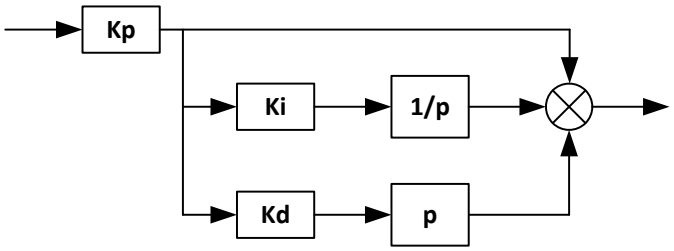


Рис. 23 Структура ПИД-регулятора

ПИД регулятор реализован в файле V_pid_reg3.c в виде модуля TPidReg3. В проекте модуль ПИД регулятора используется в качестве регуляторов тока, скорости (частоты вращения).

Наиболее простой и часто применяемый способ настройки таких регуляторов – подбор коэффициентов. К примеру, чтобы настроить коэффициенты ПИ-регулятора, можно сначала обнулить Ki и задать маленький Kp. Повышать Kp до тех пор, пока переходной процесс реакции на скачок задания не станет колебательным. Несколько снизить Kp до устранения колебательности. Аналогично повышать Ki до достижения приемлемого быстродействия регулятора или до появления колебательности.

Регуляторы токов по осям d и q настраиваются одновременно, с одинаковыми коэффициентами. Сначала настраиваются внутренние контура системы управления, а затем внешние.

Более подробное описание структуры данного ПИД регулятора можно изучить в документе pid_reg3.pdf Texas Instruments, который можно легко найти в поисковой системе.

5.2. Кривая U/f

Кривая U/f – соответствие напряжения и частоты, прикладываемых к двигателю. Данный блок служит для расчета величины выходного напряжения в зависимости от выходной частоты (т.н. закон управления двигателем). Расчет осуществляется методом линейной интерполяции между двумя опорными точками:

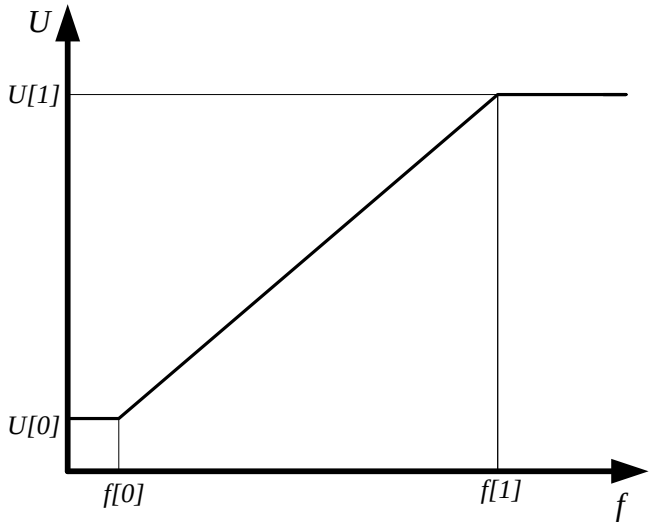


Рис. 24 Вид кривой U/f

Настройки модуля содержатся в группе **Кривая U-f**. Напряжение и частота каждой точки могут быть заданы пользователем.

Кривая U/f симметрична для положительной и отрицательной скорости.

Модуль применяется в структуре скалярного управления. Реализован в файле vhzprof.c в виде модуля TVhzProf.

5.3. Задатчик интенсивности (ЗИ)

Данный блок обеспечивает ограничение темпов разгона/торможения (ускорение) двигателя.

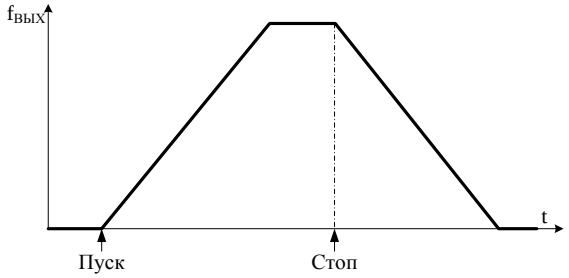


Рис. 25 Пуск и торможение по задатчику интенсивности

Настройки темпа разгона/торможения задается параметром **ЗИ.Т разгона до номинала**.

Темп ускорения и замедления это время, за которое частота с нуля достигает номинальной. Т.е. темп разгона «5 секунд» значит, что с 0 Гц до 100Гц частота будет нарастать 5 секунд. Соответственно, например, с 20 Гц до 40 Гц разгон будет идти 1 секунду. Торможение, соответственно, наоборот. Номинальная частота рассчитывается исходя из параметров **Базов. параметры. Номинальная скорость** и **Базов. параметры. Число пар полюсов** и отображается в параметре **Базов. параметры. Номинальная частота**.

Задатчик интенсивности применяется при разгоне и торможении двигателя в большинстве структур управления. Реализован в файле v_rmp_ctrl.c в виде модуля TRMPCtrl.

5.4. Модуль фазных преобразований

Фазные преобразования переводят трехфазную систему величин A,B,C со сдвигом на 120° в двухфазную систему α,β со сдвигом 90°. В зарубежной литературе называется преобразованием Кларк (Clarke).

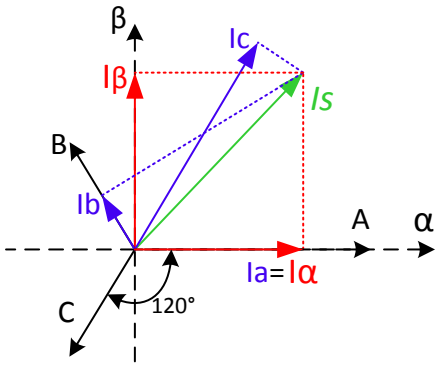


Рис. 26 Принцип фазных преобразований

Преобразование осуществляется с сохранением амплитуды тока фазы, т.е. а $I_A = I_\alpha$.

В ПО используется обозначение, принятое за рубежом. Входы модуля clarke: as – фаза A, bs – фаза B. Выходы: ds – фаза α, qs – фаза β. Формулы, по которым осуществляется преобразование в данном модуле, следующие:

$$ds = as;$$

$$qs = \frac{as + 2 \cdot bs}{\sqrt{3}}$$

В ПО данный модуль используется для преобразования токов статора из трехфазной системы координат в двухфазную. Реализован в файле clarke.c в виде модуля TClarke.

5.5. Модуль координатных преобразований

Координатные преобразования поворачивают двухфазную систему со сдвигом 90° на заданный угол. В зарубежной литературе называется преобразованием Парка (Park). В основном используется для преобразования стационарных осей α,β к вращающимся d,q и обратно (обратно часто используется инверсное преобразование, ipark).

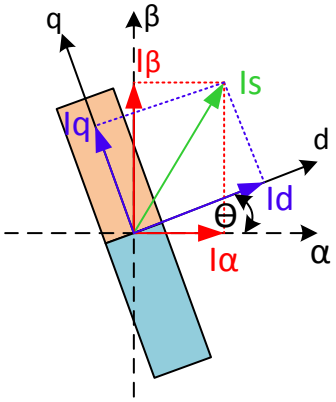


Рис. 27 Принцип координатных преобразований

В ПО обозначения используются следующие. Входы модуля park: ds – обычно фаза α, qs – обычно фаза β, ang – угол поворота осей. Выходы: de – обычно фаза d, qe – обычно фаза q. Для модуля ipark входные и выходные оси названы наоборот (входы – de и qe, а выходы ds и qs).

Формулы, используемые в ПО в модулях park и ipark представлены ниже.

$$de = ds \cdot \cos(ang) + qs \cdot \sin(ang);$$

$$qe = qs \cdot \cos(ang) - ds \cdot \sin(ang).$$

$$ds = de \cdot \cos(ang) - qe \cdot \sin(ang);$$

$$qs = qe \cdot \cos(ang) + de \cdot \sin(ang).$$

Данный модуль используется практически во всех структурах системы управления для обеспечения вращения координат. Реализован в файле park.c в виде модуля TPark (прямое преобразование координат) и в файле ipark.c в виде модуля TIPark (обратное преобразование координат).

5.6. Модуль ШИМ

Модуль ШИМ (широтно-импульсной модуляции) предназначен для управления ключами инвертора и отработки заданного вектора напряжения. На вход модуля поступает задание в виде напряжений в ортогональной системе координат с осями альфа, бета (в двухфазной системе координат). Если модуль ШИМ включен, то на следующем периоде ШИМ заданное напряжение будет отработано. Модуль отрабатывает постоянное напряжение, никакой эволюции вектора во времени он не осуществляет: «синусоидальность» выходного напряжения должен обеспечивать модуль более высокого уровня путем соответствующего изменения задания.

Модуль ШИМ в данном проекте рассчитывается на фиксированной частоте 10кГц, на этой частоте рассчитываются и обновляются уставки сравнения таймеров ШИМ микроконтроллера. Однако частота ШИМ может быть задана любой. При этом отработка ШИМ и изменение его задания происходит асинхронно. Такой подход имеет ряд достоинств и недостатков, на описание которых выходит за рамки данного документа.

Модуль позволяет «на ходу» изменять частоту ШИМ, мёртвое время, тип алгоритма ШИМ.

Алгоритмов управления ключами два: синусоидальная и векторная 6-ти секторная модуляция.

Скважности синусоидальной ШИМ в относительных единицах рассчитываются по формулам:

$$\gamma_A = \frac{1}{2} - U_{\alpha.зад.};$$

$$\gamma_B = \frac{1}{2} - \left(U_{\beta.зад.} \cdot \frac{\sqrt{3}}{2} - U_{\alpha.зад.} \cdot \frac{1}{2} \right);$$

$$\gamma_C = \frac{1}{2} - \left(-U_{\beta.зад.} \cdot \frac{\sqrt{3}}{2} - U_{\alpha.зад.} \cdot \frac{1}{2} \right).$$

Скважности в относительных единицах умножаются на период ШИМ и загружаются в регистры сравнения модулей ШИМ.

На Рис. 28 показана общая структура модуля ШИМ. Заданные напряжения по перпендикулярным осям альфа (Ualpha) и бета (Ubeta) поступают на вход модуля из более высокоуровневой структуры управления двигателем (из sm_ctrl). Их мгновенные значения доступны пользователю для чтения в переменных **ШИМ.UaЗад** и **ШИМ.UbЗад**. Далее следует блок расчета их амплитуды, результат которого помещается в **ШИМ.U_ампл**. После этого напряжения Ualpha и Ubeta поступают на блок коррекции выходного напряжения в зависимости от напряжения на звене постоянного тока. Так, если напряжение на звене постоянного тока понижено, то требуется пропорционально этому повысить задание на выходное напряжение. На блок поступает флаг разрешения работы (параметр **ШИМ.Компенсация Ud**) и напряжение Ud с соответствующего датчика. Кроме того, для более точной подстройки компенсации может использоваться коэффициент **ШИМ.Коэф. компенсации Ud**, использующийся в формуле компенсации:

Ualpha_после = Ualpha_до *[1+(Uном-Uизмер)* Квых/ Uном],

Ubeta_после = Ubeta_до*[1+(Uном-Uизмер)* Квых/ Uном], где

Квых - коэффициент компенсации **ШИМ. Коэф. компенсации Ud**;

Ualpha_до, Ubeta_до – напряжения по осям альфа, бета до блока компенсации пульсаций;

Uном – номинальное напряжение на звене постоянного тока;

Uизмер – измеренное напряжение на звене постоянного тока.

Следует заметить, что компенсация никак не сказывается на отображаемой амплитуде выходного напряжения **ШИМ.U_ампл**, так как реальное напряжение на выходе инвертора при изменении напряжения Ud и работе блока компенсации также меняться не должно. Кроме того, существуют ограничения на возможности компенсации связанные с максимальным напряжением, реализуемым инвертором. Так, если привод работает на максимальной частоте с максимальным напряжением, и в это время напряжение на звене постоянного тока снижается, то у модуля ШИМ нет резерва, чтобы повысить задание на амплитуду выходного напряжения (за исключением работы «по граням шестигранника», что описано ниже). Правильно коррекция может работать только или при повышении напряжения на звене постоянного тока, или при запасе в амплитуде задания выходного напряжения.

Далее следует блок расчета амплитуды, но уже скорректированных Ualpha и Ubeta. Их амплитуда поступает на блок ограничения выходного напряжения, который обеспечивает коллинеарное ограничение заданного вектора напряжения. Ниже будет дано более подробное описание работы данного блока. Далее, скорректированные и ограниченные напряжения, которые для отличия названы UalphaNorm и UbetaNorm, поступают непосредственно на блок, реализующий алгоритм переключения ключей в соответствии с алгоритмом выбранного типа ШИМ. Блок управляется параметрами:

- **ШИМ.Тип ШИМ**: выбирает синусоидальный или 6-ти секторный алгоритм;
- **ШИМ.Частота ШИМ**: задает частоту ШИМ в кГц;
- **ШИМ.Мертвое время (мкс)**: задает величину мертвого времени в мкс.

Результатом работы блока являются скважности для каждой из стоек инвертора GammaA, GammaB, GammaC. Под скважностью понимается отношение времени открытого состояния верхнего ключа стойки к периоду ШИМ. Скважности стоек поступают на блок ограничения минимальной

скважности. Так как прикладывать к ключам управляющий сигнал, сравнимый с временем их коммутации нежелательно (для мощных инверторов), можно искусственно ограничить минимально-допустимые скважности. Так, если скважность любой из стоек оказываются меньше заданного в параметре **ШИМ.Ограничение мин. скважн.** времени в мкс, происходит обнуление этой скважности. Если **ШИМ.Ограничение мин. скважн.** равен нулю, блок не используется. После всех описанных преобразований скважности GammaA, GammaB, GammaC попадают непосредственно на выход контроллера в виде ШИМ-сигнала и управляют 6ю ключами инвертора.

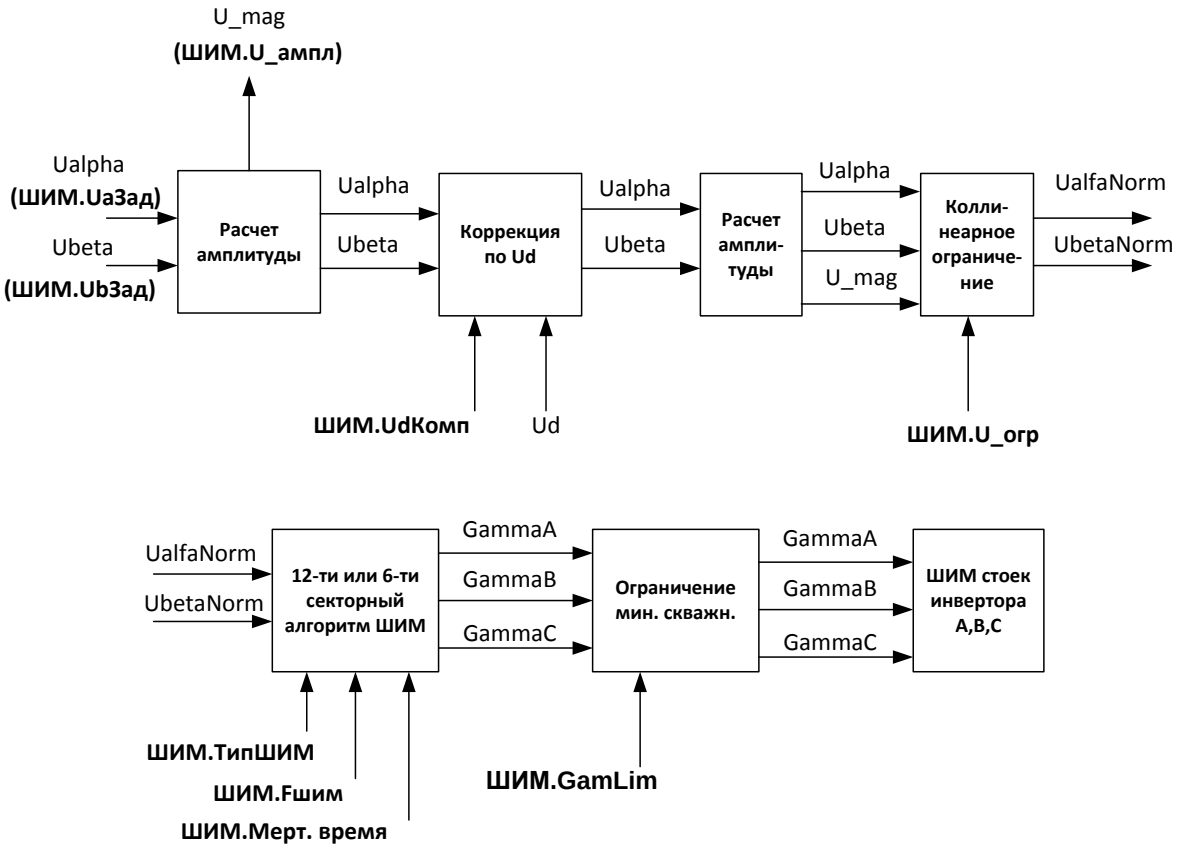


Рис. 28 Схема алгоритма ШИМ

Для более детального понимания работы блока коллинеарного ограничения вектора напряжения обратимся к рисунку ниже (справедливо только для векторной ШИМ).

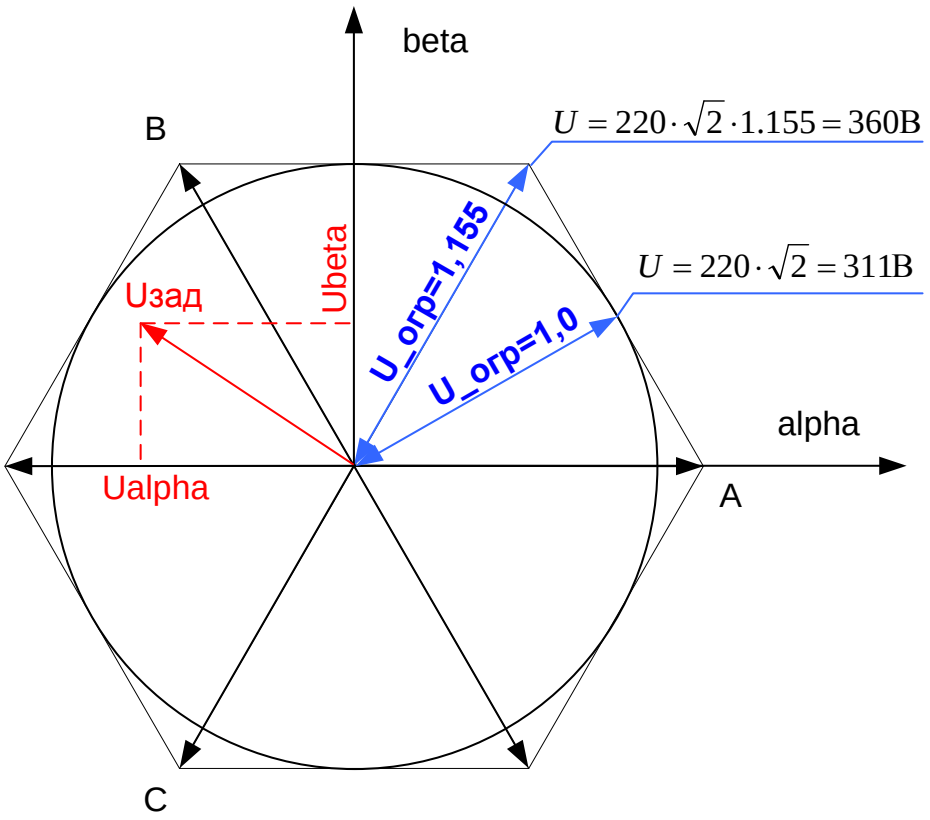


Рис. 29 Графическое представление работы векторной ШИМ

На нем изображены шесть базовых векторов инвертора напряжения, расходящихся из центра координат. Это максимальные фазные амплитудные напряжения, которые могут быть приложены к двигателю при тех или иных комбинациях ключей инвертора. Например, базовый вектор, направленный в сторону оси фазы А, может быть получен открытием верхнего ключа стойки А инвертора и открытием нижних ключей стоек В и С. Все напряжения, которые могут быть получены в результате использования широтно-импульсной модуляции, лежат внутри образованного базовыми векторами шестиугольника и не могут превосходить его. На данном рисунке отмечены сразу две системы координат: образованная перпендикулярными осями alpha и beta, в которых происходит работа модуля векторной ШИМ и остальной системы управления, а также соответствующие им трехфазные оси A,B,C. Если система управления задает вектор напряжения небольшой амплитуды, например Uзад, как показано на рисунке, то блок ограничения не используется. Если же Uзад превысит шестиугольник (что может быть, если, например, пользователь задаст в модуле кривой U/f слишком большое напряжение), то требуется ограничение вектора. Используется так называемое коллинеарное ограничение, при котором угловое положение вектора (и, соответственно, направление) сохраняются, а амплитуда уменьшается до необходимой величины. Если ограничивать вектор напряжения вписыванием в шестиугольник, то при таком ограничении выходное фазное напряжение будет не синусоидальным (похожим на трапецию), поэтому чаще всего используется ограничение заданного вектора окружностью, вписанной в шестиугольник. Окружность показана на рисунке. Параметр **ШИМ.U_огр** управляет амплитудой окружности, которой ограничивается вектор напряжения. Когда этот параметр равен 1.0, то вектор будет ограничен максимально-допустимым напряжением, при котором еще не происходит искажения синусоидальности (справедливо для векторных типов ШИМ). Если задавать этот параметр больше 1.0, то начнет автоматически работать ограничение вектора шестиугольником там, где окружность превысила его. Так, если задать **ШИМ.U_огр=1,155**, то окружность будет описана вокруг шестиугольника, в результате чего ограничение будет происходить по граням шестиугольника. Это является предельным выходным напряжением для инвертора. При напряжении на звене постоянного тока 540 В (представим, что это «обычный» преобразователь частоты), т.е. когда фазное действующее напряжение на входе 220 В, параметру **ШИМ.U_огр=1,0** соответствует фазное действующее напряжение на двигателе 220В (или 311 В амплитудное фазное)

без учета падений напряжения на ключах и мертвого времени. При питании от источника 48 В предельное действующее фазное напряжение составит 19,5 В. Иногда требуется любыми способами повысить выходное напряжение, даже ценой искажения синусоидальности. Для этого необходимо задать **ШИМ.U_огр=1,155** (и соответствующие напряжения в модуле кривой U/f). В этом случае выходное напряжение повысится до 360 В амплитудного фазного напряжения (при 540 В на ЗПТ). Кроме повышения напряжения такое ограничение дает также уменьшение потерь в ключах инвертора (при работе на максимальном напряжении). Это связано с конкретным алгоритмом реализации векторной ШИМ. В этом случае количество коммутаций на периоде ШИМ сокращается в два раза (в векторной ШИМ).

Особо следует отметить, что в инверторе TI питание драйверов ключей «бутстрепное». Это значит, что для питания драйвера верхних ключей инвертора используется конденсатор, который заряжается в то время, когда открыт нижний ключ инвертора. Поэтому в любой схеме ШИМ необходимо, чтобы на какой-то промежуток времени нижние ключи инвертора обязательно открывались на периоде ШИМ. Иначе ёмкость разрядится и верхние ключи перестанут управляться. Необходимо использовать ограничение максимальной скважности инвертора в модуле ШИМ, чтобы скважность по каждой стойке никогда не превышала, например, 90%. Для этого используется параметр **ШИМ. U_lim**. Если скважность приблизится к 100%, то в микросхеме драйверов ключей сработает защита, что в ПО UniCON отобразится как аппаратная авария.

При этом параметр **ШИМ. U_lim** по-разному работает для векторной и скалярной ШИМ. Для векторной ШИМ его можно задать равным 90% (хотя при этом пострадает измерение тока, но работоспособность остается, см. подробнее главу 5.1). Для синусоидальной ШИМ предельным работоспособным значением является ограничение 70% в силу более низкого выходного напряжения в данном типе модуляции.

Модуль ШИМ реализован в файле V_PWM_6_12.c в виде модуля TPWM_6_12. В файле находится только реализация синусоидальной ШИМ, реализация векторной ШИМ вынесена в библиотеку libV_PWM_Vect_LIB.a.

5.1. Модуль АЦП

Модуль реализован в файле V_adc.c в виде модуля с названием TAdcDrv. Модуль измеряет токи фаз двигателя (ток фазы A и B, ток фазы C рассчитывается как сумма двух других) и напряжение звена постоянного тока инвертора. Параметры отображаются в группе АЦП. О настройке параметров написано в главе 4.4.

Отдельно стоит описать способ измерения токов фаз. В инверторе используются шунтовые датчики токов фаз. Ток фазы пропорционален падению напряжения на шунтовом калиброванном резисторе. Эти измерительные резисторы установлены последовательно с нижними ключами инвертора – R80, R81, R82 – смотрите принципиальную схему инвертора по ссылке http://motorcontrol.ru/production/controlcards/controlcard_nt32m4f1/ в разделе документация. Сигнал с резисторов подается на операционные усилители (двух разных типов, встроенные в драйвер ключей и внешние «образцовые»), после чего сигнал подается в контроллер. Таким образом, измерить корректно ток в фазе можно только в тот момент, когда включен нижний ключ инвертора – в иной ситуации ток фазы может протекать по другому контуру, в обход измерительного резистора.

Существуют разные методы организации такого измерения тока. В текущем проекте измерение производится строго в тот момент, когда во время работы ШИМ открываются нижние ключи инвертора, т.е. в момент приложения нулевого вектора напряжения. Это происходит, когда таймеры микроконтроллера, формирующие ШИМ, достигают максимума при счете вверх-вниз. В этот момент по событию запускается измерение токов АЦП – см. Рис. 30.

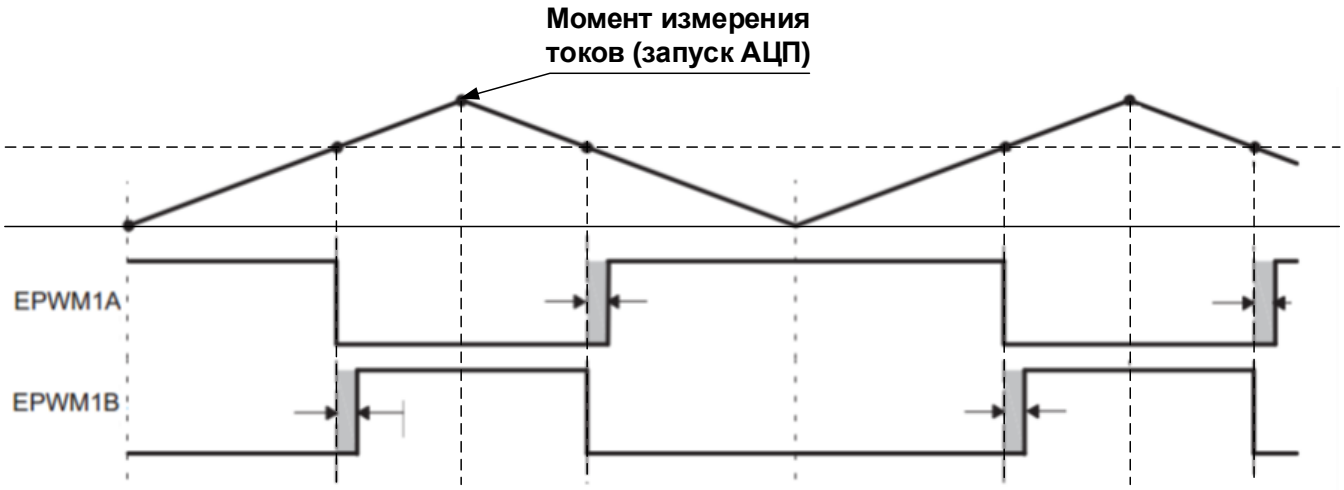


Рис. 30 Синхронизация ШИМ и измерения токов АЦП

Проблема этого способа измерения токов в том, что при высокой частоте ШИМ переходной процесс в аналоговых цепях может не успеть закончиться к тому моменту, когда приходит время измерять величину тока. Также ситуация ухудшается с ростом скважности, когда всё меньшее время остается между включением нижнего ключа и моментом измерения. На данном отладочном комплекте время переходного процесса в аналоговой цепи между коммутацией ключа и стабилизацией сигнала на входе АЦП микроконтроллера составляет около 3-4 мкс. Это необходимо учитывать при повышении частоты ШИМ и скважности управления ключами. Это время больше, чем на оригинальном комплекте TI с их микроконтроллерами, так как в данном комплекте используются дополнительные операционные усилители на плате VectorCARD, преобразующие входной сигнал АЦП от диапазона 0-3В к необходимому для К1921ВК01Т диапазону 0-1,5В. Дополнительные аналоговые цепи затягивают время переходного процесса.

Понижать частоту ШИМ для «улучшения» измерения АЦП также следует осторожно, так как индуктивность фазы электродвигателя ACM601V36-1000 небольшая: при напряжении питания инвертора 24В пульсации тока на частоте ШИМ 20кГц на периоде ШИМ составляют около 0.3А. Однако модуль ШИМ построен таким образом, что при приложении нулевого вектора напряжения (момент измерения тока) фактический ток в фазе двигателя проходит через свое среднее значение. Поэтому, несмотря на пульсацию, измерение тока осуществляется корректно. Наиболее оптимальная частота ШИМ 10-20кГц – при этой частоте достигается оптимум между качеством измерения тока и пульсацией тока на периоде ШИМ. Максимальная погрешность измерения тока при этом может составлять до 0.15А.

На рисунке Рис. 31 показана осциллограмма тока фазы, снятая двумя разными датчиками: штатным шунтовым датчиком тока (красный и голубой сигналы) и внешним датчиком тока на эффекте Холла компенсационного типа, сигнал которого был заведен на свободный вход микроконтроллера (фиолетовый сигнал). Также синим показана скважность на ключах фазы А из модуля векторной ШИМ. Осциллограмма получена в скалярной системе управления при частоте ШИМ 20кГц, тип ШИМ – векторная. Видно, что пока был включен нижний ключ и не коммутировался, ток фазы измерялся корректно (показания шунтового датчика тока и внешнего образцового совпадали). Когда ключи фазы начали коммутироваться, появилась ошибка в измерении тока порядка 0.15А.

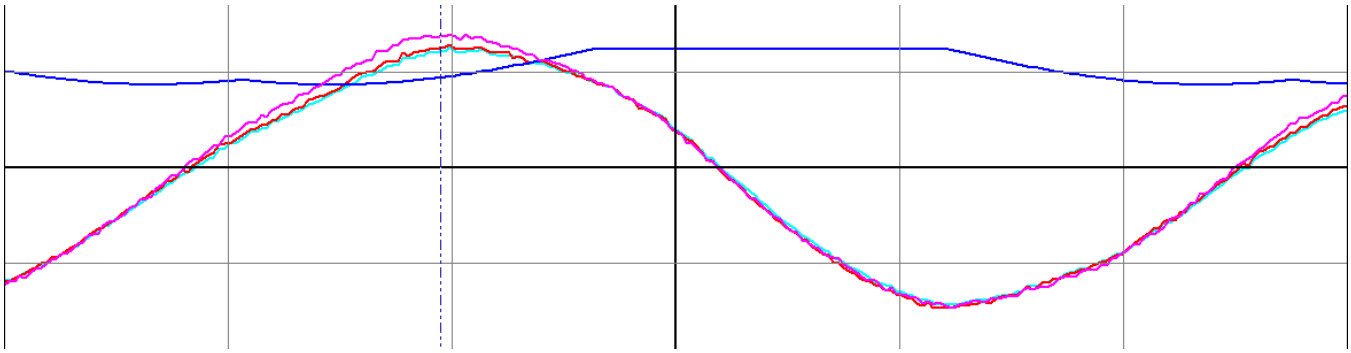


Рис. 31 Ток фазы, измеренный внешним датчиком тока (фиолетовый), измеренный через шунт (голубой и красный) и скважность на ключах фазы при работе векторной ШИМ при частоте 20кГц. В одной клетке 1,5А.

При частоте ШИМ 10кГц совпадение токов лучше, однако появляются слышимые шумовые эффекты от двигателя из-за пульсации тока на периоде ШИМ.

Также для улучшения точности измерения (уменьшения шума АЦП) применяется усреднение измерений, полученных на разных периодах ШИМ. Так, основной цикл расчета контуров тока выполняется на частоте 10кГц. При частоте ШИМ 20кГц образуется два измерения токов, которые помещаются в аппаратное FIFO модуля АЦП микроконтроллера, а затем из них вычисляется среднее значение, которое уже используется в системе управления. При большей частоте ШИМ большее количество точек будет применяться для усреднения.

5.2. Модуль ДПР на элементах Холла

ДПР состоит из трех дискретных элементов Холла, которые меняют свое выходное электрическое состояние в зависимости от положения ротора двигателя. На Рис. 20 показаны временные диаграммы работы датчика. Из дискретных сигналов, поступающих на вход контроллера, программным драйвером формируется дискретное значение углового положения с шагом в 60 градусов (доступно для наблюдения в параметре **ДПР-Холл.ЭлУгол6**). На основе этого углового положения и текущей частоты вращения формируется экстраполированное значение углового положения, непрерывное на электрическом периоде двигателя (доступно для наблюдения в параметре **ДПР-Холл.ЭлУгол**). Именно это значение используется далее для векторной системы управления. Также к параметру **ДПР-Холл.ЭлУгол** прибавляется задаваемое пользователем значение смещения датчика положения ротора, используемое для программной юстировки (калибровки) положения датчика (параметр **ДПР-Холл.СмещУгол**). Помимо углового положения модуль вычисляет частоту вращения двигателя (скорость), засекая время между метками датчика. Значение скорости используется для регулятора скорости двигателя в векторной систему управления и доступно для наблюдения в параметре **ДПР-Холл.Скорость**.

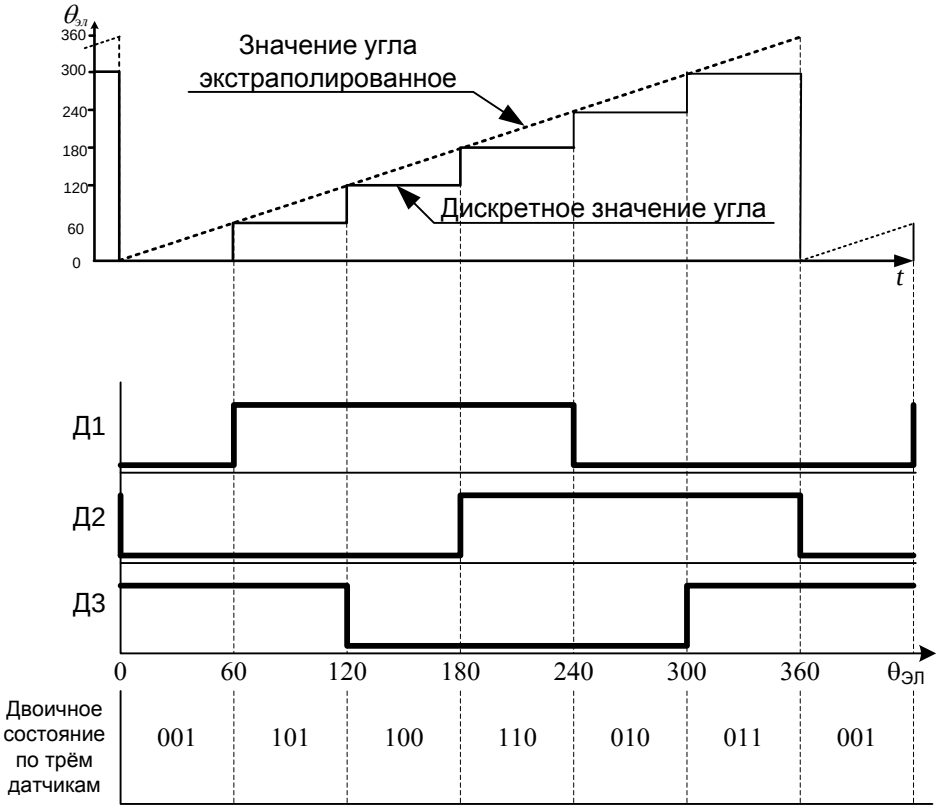


Рис. 32 Временные диаграммы работы датчика положения ротора

Интерполяцию можно отключить параметром **ДПР-Холл.Инт-полятор вкл**. Она также отключается автоматически при скорости ниже уставки **ДПР-Холл.Уставка скорости**. Скорость обнуляется, если состояние датчика не изменялось более **ДПР-Холл.Уставка обнуления скорости** в миллисекундах. Смещение угла настраивается параметром **ДПР-Холл.СмещУгол**.

В параметре **ДПР-Холл.Дискретные входы** можно наблюдать текущее состояние дискретных входов микроконтроллера, используемых для приёма сигнала датчика. В параметре **ДПР-Холл.Счётчик прерываний по приему** видно срабатывание прерываний по изменению фронтов сигнала датчика.

Модуль реализован в библиотеке libV_DPR_eCAP_LIB.a в виде модуля с названием TDPReCAP, кроме того имеются обработчики прерываний CAP в файле main.c.

5.3. Модуль инкрементального энкодера

Модуль реализован в файле V_QEP.c. Инкрементальный энкодер, в отличие от датчика на элементах Холла, не выдаёт абсолютного положения ротора, а лишь показывает, на какой угол и в каком направлении поворачивается двигатель. Разрешение энкодера двигателя *ACM601V36-01-1000* составляет 1000 меток на механический оборот. Каждая метка на оптическом диске энкодера создает 4 фронта, поэтому на механический оборот датчик сформирует 4000 фронтов (суммарно по обоим каналам A и B). В модуле обработки механический угол преобразуется в электрический. Электрический оборот – период изменения магнитного состояния двигателя, т.е. минимальный угол, пройдя который следующий северный (к примеру) полюс окажется точно в том месте, где был предыдущий северный полюс. Электрический оборот равен механическому, разделенному на число пар полюсов. Для двигателя *ACM601V36-01-1000* число пар полюсов равно 4 и электрический оборот составляет 90 геометрических градусов. Индексная метка (репер) в энкодере *ACM601V36-01-1000* отсутствует.

Угол положения ротора (электрический и механический) отображается в параметрах **ДПР-Энкодер.Угол. электр.** и **ДПР-Энкодер.Угол мех.**. Частота вращения, рассчитываемая по энкодеру, отображается в **ДПР-Энкодер.Скорость**. Т.к. энкодер показывает только относительное смещение ротора, ему требуется задать начальное смещение угла, чтобы при нулевом угле поворота ротора он показывал нулевой электрический угол. Смещение задается параметром **ДПР-Энкодер.СмещУгол**. Это смещение необходимо настраивать каждый раз заново при за-

пуске системы управления, так как система управления не знает, в каком в данный момент положении находится вал двигателя после включения и что считать нулевым положением. Процедура настройки смещения описана в главе 4.7. В дальнейшем пользователь может самостоятельно реализовать необходимый ему способ автоматической юстировки энкодера, например, по сигналу с датчика на элементах Холла, по реперной (индексной) метке (если она имеется в используемом датчике), по концевым выключателям и т.п.

Модуль реализован в файле V_QEP.c в виде модуля с названием TposspeedEqep.

5.4. Модуль цифрового осциллографа (даталоггер)

Реализован в файле V_data_log.c в виде модуля TDataLog. Осуществляет цифровое осциллографирование (запись в четыре массива) значений из четырех выбранных пользователем переменных. Модуль работает совместно с режимом «выборка» осциллографа ПО UniCON.

Пользователь выбирает в словаре объектов до четырех любых переменных, которые он хочет осциллографировать, и добавляет их в осциллограф UniCON. Затем переходит на вкладку осциллографа «выборка». В этот момент UniCON записывает адреса (индекс, подындекс) желаемых для осциллографирования объектов словаря CANopen в параметры dlog.ind_subind1-4 группы dlog. Далее функция dlog_background_analizer в ПО МК по известным адресам словаря CANopen вычисляет адреса этих переменных в оперативной памяти микроконтроллера, настраивая на них указатели dlog_iptr1-4. Также та функция обрабатывает поле команд control, через которое UniCON управляет осциллографом (когда записывать данные, спрашивает, готова ли осциллограмма и т.п.).

В функции data_log_update осциллограф заполняет четыре 16ти разрядных массива по 256 точек значениями переменных, которые выбрал пользователь. Функция вызывается с частотой вычисления системы управления (10кГц) и записывает в массив значения очередной точки. Запись можно делать не каждый такт расчета системы управления, а с прореживанием, которое задается через UniCON на вкладке «выборка» в поле ввода «делитель»: например, каждый пятый такт расчета или каждый сотый.

Запуск осциллограммы может производиться как в от команды UniCON, так и самостоятельно по триггеру в ПО микроконтроллера, чтобы можно было получить осциллограмму, соответствующую какому-либо событию. Запуск осциллографа внутри ПО осуществляется функцией dlog.set_mode(1, &dlog); Эта функция запускает запись осциллограммы с того момента, когда она вызвана. Эта функция размещена во многих участках кода программы микроконтроллера – при запуске системы управления в том или ином режиме, при переходе в останов и так далее. Чтобы выбрать, когда конкретно нужно в данном опыте запустить осциллограф, сделана битовая переменная dlog.StartBits, где пользователь через UniCON устанавливает тот или иной бит, соответствующий тому или иному событию в системе управления. Так, например, запуск осциллографа по событию включения векторной системы управления выполняется следующими строками:

```
if (dlog.StartBits & (1 << 4))
      dlog.set_mode(1, &dlog);
```

Что означает «запустить осциллограф, если пользователь установил 4й бит начиная с нулевого». Пользователь должен установить в параметрах **dlog.StartBitL** и **dlog.StartBitH** (нижняя и верхняя части переменной StartBits) желаемые биты, соответствующие наступлению желаемого события в программе. Какие биты каким событиям соответствуют можно узнать, выполнив поиск строки dlog.set_mode в проекте ПО.

UniCON по нажатию кнопки «обновить» считывает текущий статус осциллографа. Если есть уже записанная осциллограмма, UniCON предложит отобразить её. Если текущей осциллограммы не записано, в момент нажатия кнопки «обновить» запишется и отобразится новая осциллограмма.

Чтобы иметь возможность видеть предысторию, а именно осциллограмму до и во время наступления определенного события, применяется следующая схема. После окончания загрузки осциллограммы из микроконтроллера UniCON запускает осциллограф в режим постоянной кольцевой записи выбранных данных. Далее пользователь создает условия эксперимента, чтобы запись осциллограммы внутри МК произошла снова по заданному им событию. Когда внутри

ПО микроконтроллера вызывается функция записи осциллограммы dlog.set_mode, то есть два возможных варианта поведения. Первый – осциллограмма запишется сначала, т.е. начальному моменту времени будет соответствовать срабатывания триггера, т.е. предыстории не будет. Второй вариант – по вызову dlog.set_mode осциллограф допишет массив до конца и остановится (осциллограф постоянно пишет данные ещё с прошлого раза). Поведение регулируется пользователем заданием переменной **dlog. Записывать сначала**. Если значение установлено «нет», то получится осциллограмма с предысторией. Однако размер предыстории в таком способе задать невозможно, это элемент «удачи» – зависит от того, на каком элементе массива счетчик записи осциллограммы окажется на момент срабатывания триггера и вызова dlog.set_mode. При желании пользователь может доработать модуль осциллографа для его более интеллектуального поведения.

Загрузка записанной осциллограммы в UniCON производится через передачу массива данных драйвера CANopen (функция блочной передачи), или, если эта функция по какой-то причине не работает, поэлементным вычитыванием данных через служебную переменную **dlog.next_value_var**.

Особо стоит отметить, что осциллограф 16ти разрядный. При добавлении в осциллограф 32х разрядной переменной в осциллографе записывается её старшая часть. Для переменных в формате IQ8.24, на которых основана вся целочисленная математика данного проекта, это приемлемо. После усечения переменной получается формат 8.8, что достаточно для наблюдения большинства процессов. Таким образом следует помнить, что в режиме работы «выборка» осциллографа переменные будут иметь соответствующее усечение формата 8.8 (в отличие от «непрерывного» режима, где переменные запрашиваются в реальном времени, без усечения формата, но медленно). Если добавить в осциллограф какой-либо 32х разрядный счетчик, то в режиме «выборка» не будет видно изменения его младших 16ти разрядов, что иного неприемлемо. Необходимо для отображения такого счетчика копировать его значение внутри программы МК в 16ти разрядную переменную, например, в отладочные переменные Debug3,4.

Дополнительную информацию о работе функции осциллографирования можно прочитать в руководстве пользователя на программу UniCON.

5.5. Модуль защит

Модуль защит является одним из наиболее важных элементов системы управления, так как обеспечивает быстрое распознавание некоторого числа аварийных событий. Это, в свою очередь, позволяет системе управления среагировать и предпринять необходимые меры по прекращению развития аварийной ситуации и безопасному отключению оборудования. Модуль обеспечивает не только защиту электрооборудования инвертора (защиту силовых ключей и т.п.), но и защиту непосредственно двигателя (защиту от коротких замыканий и т.п.). Фиксирование и отработка защит реализуется как программными (например, максимальный ток) средствами контроллера, так и аппаратными средствами драйвера (защита ключей инвертора от сверхтока и перегрева обрабатывается на аппаратном уровне, что позволяет диагностировать аварию и отключать оборудование с минимально возможными задержками).

Характеристики модуля:

- частота дискретизации (для программно-реализуемых защит) – 10 кГц;
- возможность фиксации до 32 аварийных событий;
- возможность оперативной настройки списка разрешенных и запрещенных аварий;
- протоколирование возникающих аварий в соответствующем «журнале аварий» (сохраняется 50 последних аварий).

Список идентифицируемых аварий представлен в таблице.

№	Тип аварии	Описание	Параметр порога срабатывания в группе «Защиты»	Бит маскирования аварии

№	Тип аварии	Описание	Параметр порога срабатывания в группе «Защиты»	Бит маскирования аварии
1.	**Понижение Udc**	Понижение напряжения на ЗПТ ниже уставки	**Минимальное напряжение**	mask1L.b1 1111 1110
2.	**Повышение Udc**	Повышение напряжения на ЗПТ выше уставки	**Максимальное напряжение**	mask1L.b2 1111 1101
3.	**Резерв**			mask1L.b3 1111 1011
4.	**Резерв**			mask1L.b4 1111 0111
5.	**Резерв**			mask1L.b5 1110 1111
6.	**Резерв**			mask1L.b6 1101 1111
7.	**Резерв**			mask1L.b7 1011 1111
8.	**Резерв**			mask1L.b8 0111 1111
9.	**Сбой прогр-ы 1кГц**	Программная ошибка 1кГц (превышение времени выполнения программного кода в прерывании 1кГц)	-	mask1H.b1 1111 1110
10.	**Сбой прогр-ы 10кГц**	Программная ошибка 10кГц (превышение времени выполнения программного кода в прерывании 10кГц)	-	mask1H.b2 1111 1101
11.	**Резерв**			mask1H.b3 1111 1011
12.	**Резерв**			mask1H.b4 1111 0111
13.	**АППАРАТНАЯ**	Аппаратная авария инвертора	-	mask1H.b5 1110 1111
14.	**Резерв**			Mask1H.b6 1101 1111
15.	**Резерв**			mask1H.b7 1011 1111
16.	**Резерв**			mask1H.b7 0111 1111
17.	**Резерв**			mask2L.b2 1111 1101
18.	**Резерв**			mask2L.b3 1111 1011
19.	**Резерв**			mask2L.b4 1111 0111
20.	**Резерв**			mask2L.b5 1110 1111
21.	**Макс. ток фазы А**	Превышение тока фазы А;	**Максимальный ток**	mask2L.b6 1101 1111
22.	**Макс. ток фазы В**	Превышение тока фазы В	**Максимальный ток**	mask2L.b7 1011 1111
23.	**Макс. ток фазы С**	Превышение тока фазы С	**Максимальный ток**	mask2L.b8 0111 1111
24.	**Резерв**			mask2H.b1 1111 1110
25.	**Резерв**			mask2H.b2 1111 1101
26.	**Макс. скорость**	Превышение уставки защиты по скорости	**Максимальная скорость**	mask2H.b3 1111 1011

Все уставки защит можно настроить в группе **Защиты**. При возникновении любой незамаскированной аварии преобразователь мгновенно отключается (отключаются ключи). Каждая авария может быть замаскирована. При её маскировании ПО контроллера не обрабатывает событие возникновения аварии и преобразователь продолжит работу (если это возможно…). Для маски-

рования служат параметры с **Защиты.Маска аварий 1L** по **Защиты. Маска аварий 2H**. Установленный ноль в соответствующем бите маскирует аварию. Соответствие номера бита и маскируемой аварии см. в таблице (нумерация бит с единицы).

Модуль реализован в файле SM_protect.c в виде модуля с названием TSMProtect.

5.6. Модуль часов реального времени

Модуль реализован в файле V_RTC_Clock.c в виде модуля TRTCClock. Представляет из себя интерфейс между часами реального времени микроконтроллера и ПО UniCON. Запаковывает текущее время в переменную packed_time, доступную в параметре **Текущ. знач. Часы реального времени**. В распакованном виде время доступно на панели UniCON справа сверху. Для установки времени используется функция RTC_Clock_Set_Time, которая вызывается через функцию обратного вызова драйвера CANopen (в файле V_call_back.c) при записи в параметр **Текущ. знач. Часы реального времени**.

5.7. Модуль работы с драйвером ключей DRV8301

Модуль реализован в файле DRV8301_SPI.c в виде модуля Tdrv8301interface. Служит для конфигурирования микросхемы драйвера ключей DRV8301 через интерфейс SPI (для комплекта на основе DRV8301). Настраивает, по скольким каналам управляются ключи, как работает реакция на аварию, коэффициент усиления внутренних операционных усилителей для шунтов и т.п.

5.8. Отображение текущих параметров привода

На главном экране ПУ (группа **Текущ. знач**) отображаются наиболее интересные текущие параметры привода. Например, скорость, ток статора и т.п. В данном разделе описано, как происходит вычисление этих величин.

- **Скорость** – текущая частота вращения двигателя в оборотах в минуту. Вычисляется по датчику положения ротора (в зависимости от выбранного типа ДПР). Для скалярной и частотно-токовой системы управления отображается заданная скорость после ЗИ ($f_{вых}$ на Рис. 3).
- **Ток статора –** амплитудное значение тока статора двигателя. Вычисляется при помощи координатных и фазных преобразований и равно току по оси q, т.к. по оси d ток задается нулевым.
- **ЭлУголЗад –** заданный угол положения ротора. Вычисляется при расчете системы управления.

5.9. Различные настройки системы

В группе **Настройки СУ** собраны различные настройки, не вошедшие в отдельную тематическую группу параметров.

Настройки СУ.РежРаб –режим работы инвертора. 2 – режим удержания (задание постоянного тока), 3 – скалярная СУ, 4 – частотно-токовая СУ, 5 – векторная СУ с ОС по датчику Холла, 6 – векторная СУ с ОС по инкрементальному энкодеру.

Настройки СУ.Рекуперация – разрешенные рекуперации. Влияет на нижнее ограничение регуляторов тока.

5.1. Переменные для отладки

В демонстрационной версии драйвера CANopen возможность добавлять свои переменные в словарь объектов и наблюдать их в UniCON отключена. Однако в словарь выведены несколько служебных отладочных переменных, в которые можно внутри ПО микроконтроллера присваивать любые данные и наблюдать их затем в ПО UniCON. Эти переменные доступны в группе параметров «Для отладки», а внутри ПО представлены как Debug1- Debug4 (доступны в словаре

только на чтение), DebugW1- DebugW4 доступны и для записи. Стоит отметить, что модуль цифрового осциллографа data_log 16ти разрядный. И в режиме работы осциллографа UniCON «Выборка», когда данные записываются во внутренний массив контроллера, в каждую точку записываются только 16 разрядов. Если переменная 32-х разрядная, то записывается содержимое её верхней части (для формата IQ8.24 приемлемо). Поэтому для просмотра 16ти разрядных переменных (таких как счетчики, состояние ножек и т.п.) переменные Debug3, Debug4 сделаны 16ти разрядными.

5.2. Счетчики

В группе параметров **Счетчики** содержатся параметры со счетчиков прерываний. По их инкрементированию их можно судить о работоспособности прерываний и фонового цикла.

Также в группе параметров **Счетчики** присутствуют параметры, отображающие длительность расчета программного кода в каждом из прерываний в тактах микроконтроллера. T_10кГц и T_1кГц показывают время расчета в прерывания 10кГц и 1кГц соответственно. По этим данным удобно оценивать процент загрузки ядра микроконтроллера. В ПО микроконтроллера это время высчитывается таймером общего назначения NT_TIMER2.

6. О модульности программ

Очевидно, что в большом проекте требуется разделение программного кода на логические блоки. "Встроенной" модульности языка Си на основе заключения кода в функции становится недостаточно. Выходом из ситуации стали так называемые объектно-ориентированные языки программирования, например Java, Си++ и т.п. Они позволяют связать в единый модуль – "класс" целый ряд функций, переменных; позволяют ограничить область видимости членов этого класса по отношению ко всему проекту, сделав возможным обращение только через имя самого класса (объекта), например:

```
ClassDisplay.Print("my string");
```

Внутри же самого класса обращаться к переменным этого класса можно непосредственно по их имени, что позволяет, назвав переменные короткими именами, сделать программу более наглядной.

VectorIDE и GCC поддерживает Си++, но его использование приводит (или приводили ранее) к возникновению ряда трудностей при применении для встраиваемых систем. Однако, можно реализовать аналогичную Си++ функциональность непосредственно на языке Си. "Модуль" представляет собой структуру, внутри которой размещаются необходимые переменные, а также указатели на функции модуля, предназначенные для "внешнего" использования. Сама реализация функций находится в отдельном файле, а имена функций должны быть с префиксами. В саму же структуру указатели на эти функции добавляются с короткими и простыми именами. Пример:

```
//Заголовочный файл модуля:

struct SDataLog{ //Определение структуры SDataLog (не экземпляра, а просто символьного имени)

        int BufNum; //какая-то переменная

        void (*Update)(struct SDataLog *); //указатель с именем "Update" на любую функцию, в которую передается
указатель на структуру SDataLog.

} ;

//в имени SDataLog префикс "S" имеет смысл Struct, структура.


typedef struct SDataLog TDataLog; //объявляем тип с именем TDataLog,

//который теперь является эквивалентом "struct SDataLog".

//В-принципе, его можно не вводить, но тогда во всем проекте придется пользоваться данной структурой с ключевым

//словом struct, так, как это сделано выше при объявлении указателя. Теперь же можно пользоваться

//типом TDataLog и слово struct опускать, как будет показано ниже.

//в имени TDataLog префикс "T" имеет смысл Type, тип.

//При создании экземпляра структуры в инициализационной части программы ему должны быть присвоены значения

//по-умолчанию. Для этого в каждом модуле предусматривается специальный define:

#define DATALOG_DEFAULTS { 0,\

DataLogUpdate,\

}

//тут DataLogUpdate - имя конкретной функции


//теперь при инициализации модуля достаточно написать

//TDataLog Dlog = DATALOG_DEFAULTS;
```

```
//после чего, можно, например, вызывать функцию DataLogUpdate:

//Dlog.Update(&Dlog);

//В конце заголовочного файла - прототипы функций модуля.

void DataLogUpdate(TDataLog *);


//В *.c файле модуля, фактически, находится только реализация самих функций.

void DataLogUpdate(TDataLog *p)

{

        int myBuf=p->BufNum; //какие-то операции

        ....

}
```

Передавать в функцию модуля указатель на сам экземпляр структуры обязательно – через нее функция модуля может обращаться к "своим" переменным и вызывать другие функции модуля. Использовать в функции модуля непосредственно сам экземпляр структуры нельзя (в данном случае он был назван dlog). Это нарушит модульность программы, не позволит сделать несколько экземпляров одновременно. Например, может потребоваться использовать модуль TDataLog для «логирования» сразу нескольких процессов и записывать в разные буферы. Если модуль написан абстрактно, то можно просто создать различные экземпляры структур dlog0 и dlog1 и присвоить им различные BufNum.

Соответственно, работать с ними нужно так:

```
dlog0.update(&dlog0);
dlog1.update(&dlog1);
```

При этом одна и та же функция DataLogUpdate будет работать с различными экземплярами структур и использовать разные номера BufNum. Если в функции DataLogUpdate обращаться напрямую к dlog, то, очевидно, сделать два разных экземпляра и работать с двумя буферами сразу нельзя.

Такая система модульности применена в проекте VectorCARDServoInverter. Язык Си++ не используется в основном по историческим причинам и причинам совместимости уже используемых готовых программных модулей на Си.

7. Соглашения по оформлению и структуре программы

7.1. Наименование переменных, функций и т.д.

Имена переменных и функций, как правило, пишутся строчными буквами, каждое новое слово с заглавной буквы (за исключением, возможно, первой). Другой вариант – с использованием подчеркивания для разделения слов. Например:

```
long DebugW1;
TSMCtrl sm_ctrl;
void AdcDrv_init(TAdcDrv *p)
```

Константы всегда объявляются заглавными буквами с разделением слов подчеркиванием:

```
#define CTRL_RUN_VECTOR_HALL                    5
```

Для установки принадлежности функции или константы к какому-либо модулю используются префиксы:

```
void DPReCAP_Init(TDPReCAP*);
void AdcDrv_slow_calc(TAdcDrv*);
#define PROT_FAIL                   2
```

7.2. Наименование типов

Типы данных, таких как типы структур, именуются аналогично переменным и функциям, но с префиксом "T". Например:

```
TSMProtect
```

7.3. Комментарии к коду программы

Большая часть ПО документирована в системе doxygen. Придерживаясь синтаксиса комментариев doxygen при помощи соответствующей утилиты (https://ru.wikipedia.org/wiki/Doxygen) возможна автоматическая генерация документации на основе комментариев к программе.

В терминах doxygen, чтобы обозначить принадлежность СИ-функций к определенной структуре, используется следующая схема. Структура, содержащая в себе переменные и ссылки на функции модуля, названа классом. А функции, реализация которых находится в *.c файле, названы методами с принадлежностью к соответствующему классу. Это сделано специальными директивами doxygen, такими как «\class», «\addtogroup» «\memberof».

8. Заключение

Данный документ постоянно дорабатывается и обновляется.

Последнюю на данный момент версию документации Вы можете найти на нашем сайте: http://motorcontrol.ru/production/controlcards/controlcard_nt32m4f1/.

По любым проблемам, вопросам и предложениям обращайтесь на интернет-форум ООО «НПФ Вектор» http://motorcontrol.ru/forum/, а также звоните по телефону +7(495)303-37-54.

Если вы нашли ошибку в программном обеспечении или документации просьба по возможности оповестить об этом разработчиков ООО «НПФ Вектор» любым удобным способом.