**Описание программного обеспечения**

**MCB4\_Demo**

**Руководство программиста/пользователя**

**Москва 14.09.2023**

**Оглавление**

[1. Введение 4](#_Toc145593294)

[2. Обзор файловой структуры 5](#_Toc145593295)

[2.1. Организация каталогов 5](#_Toc145593296)

[2.2. Список файлов проекта 5](#_Toc145593297)

[3. Архитектура программного обеспечения 9](#_Toc145593298)

[3.1. Наименование переменных, констант и типов данных 9](#_Toc145593299)

[3.2. Модульность программы 9](#_Toc145593300)

[3.3. Дискретные автоматы 11](#_Toc145593301)

[3.4. Прерывания и их обслуживание 12](#_Toc145593302)

[3.5. Управление распределением памяти 13](#_Toc145593303)

[3.6. Организация объявления глобальных переменных 14](#_Toc145593304)

[3.7. Ход выполнения программы 14](#_Toc145593305)

[3.8. Введение в CANopen 15](#_Toc145593306)

[3.9. Форматы вычислений и единицы измерения 17](#_Toc145593307)

[4. Описание модулей ПО 19](#_Toc145593308)

[4.1. Структуры системы управления 19](#_Toc145593309)

[4.1.1. *Структура скалярного управления* (режим работы = 3) 19](#_Toc145593310)

[4.1.2. Структура частотно-токового управления (режим работы = 4) 20](#_Toc145593311)

[4.1.3. *Режим удержания* (режим работы = 2) 21](#_Toc145593312)

[4.1.4. Структура векторного датчикового управления синхронным двигателем (режим работы = 5) 22](#_Toc145593313)

[4.1.5. Структура векторного датчикового управления для синхронного двигателя с дополнительным внешним контуром положения (режим работы = 10) 23](#_Toc145593314)

[4.1.6. Режим автонастройки смещений ДПР (режим работы = 29) 24](#_Toc145593315)

[4.2. ПИД регулятор 24](#_Toc145593316)

[4.3. Кривая U/f 25](#_Toc145593317)

[4.4. Задатчик интенсивности (ЗИ) 26](#_Toc145593318)

[4.5. Модуль ШИМ 26](#_Toc145593319)

[4.6. Модуль АЦП 31](#_Toc145593320)

[4.7. Модуль ДПР на дискретных элементах Холла 31](#_Toc145593321)

[4.8. Модуль инкрементального энкодера 32](#_Toc145593322)

[4.9. Модуль цифрового осциллографа (даталоггер) 34](#_Toc145593323)

[4.10. Модуль защит 35](#_Toc145593324)

[4.11. Отображение текущих параметров привода (модуль CurPar) 36](#_Toc145593325)

[4.12. Различные важные параметры, не относящиеся к конкретному модулю 36](#_Toc145593326)

[4.12.1. Настройки системы 36](#_Toc145593327)

[4.12.2. Переменные для отладки 37](#_Toc145593328)

[4.12.3. Счетчики 37](#_Toc145593329)

[5. Быстрый старт 38](#_Toc145593330)

[5.1. Подключение к ПК через RTCON 38](#_Toc145593331)

[5.2. Настройка базовых значений 40](#_Toc145593332)

[5.3. Настройка модуля защит 41](#_Toc145593333)

[5.4. Настройка модуля АЦП 41](#_Toc145593334)

[5.5. Запуск привода в скалярной системе управления 42](#_Toc145593335)

[5.5.1. Настройка регуляторов тока 45](#_Toc145593336)

[5.6. Системы управления синхронным двигателем 48](#_Toc145593337)

[5.6.1. Частотно-токовое управление 48](#_Toc145593338)

[5.6.2. Проверка и настройка ДПР 50](#_Toc145593339)

[5.6.3. Векторное двухконтурное управление 51](#_Toc145593340)

[5.6.4. Позиционное управление 52](#_Toc145593341)

[5.7. Системы управления асинхронным двигателем 53](#_Toc145593342)

[5.7.1. Скалярное управление 53](#_Toc145593343)

[5.7.2. Настройка регуляторов тока 53](#_Toc145593344)

[5.7.3. Частотно-токовое управление 53](#_Toc145593345)

[5.7.1. Проверка и настройка ДПР 53](#_Toc145593346)

[6. Заключение 55](#_Toc145593347)

# Введение

Данное руководство предназначено для программистов, которые начинают знакомство с программным обеспечением отладочный платы MCB4. Документ описывает основные функции и организацию демонстрационного программного обеспечения (далее ПО), а также содержит руководство по пусконаладке электропривода с применением данного ПО.

ПО реализует различные структуры управления электродвигателями различных типов.

ПО микроконтроллера предназначено для среды разработки Code Composer Studio v5 или новее.

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 RTCON для Windows, которое при помощи переходника CAN-USB или USB-UART конвертора взаимодействует через CANopen с ПО микроконтроллера и обеспечивает интерфейс пользователя. Для работы со словарем объектов CANopen используется программа COODEdit. Документация на драйвер CANopen и программы персонального компьютера (RTCON, COODEdit) представлена отдельно в виде руководств пользователя.

# Обзор файловой структуры

## Организация каталогов

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 (машина состояний или дискретный автомат): эти модули в своей основе не являются законченными блоками для реализации какой-либо одной функции, а представляют собой логическую функцию, построенную на принципах дискретных автоматов, и служат для отслеживания состояний работы, смены режимов и управляют остальными модулями. Также префикс SM\_ имеют некоторые модули-обёртки, которые не содержат своего дискретного автомата, но группируют вызовы других модулей ПО (SM\_Net, SM\_Sys).

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

Все «стандартные» поставляемые производителем микроконтроллера \*.h и \*.c файлы находятся в директориях include и src соответственно. Все самостоятельно написанные пользовательские файлы аналогично размещены в Vinclude и Vsrc.

## Список файлов проекта

|  |  |
| --- | --- |
| *Каталог* ***cmd*** | |
| ***f2810\_nonBIOS\_flash.cmd*** | файл компоновки памяти проекта. |
| ***lib*** | |
| **IQmath.lib** | Библиотека для работы с функционалом дробных вычислений с фиксированной точкой |
| * *Каталог* ***obj*** * Объектные файлы, реализующие работу драйвера CANopen. | |
| * *Каталог* ***include*** * Заголовочные файлы стандартных библиотечных модулей для работы с микроконтроллером и его периферией. | |
| * *Каталог* ***src***   Стандартные библиотечные модули для работы с микроконтроллером и его периферией. | |
| *Каталог* ***Vinclude*** | |
| **build.h** | конфигурационный заголовочный файл для настройки проекта |
| **CANBlockTransferInterface.h** | заголовочный файл пользовательских функций интерфейса блочной передачи данных драйвера CANopen. |
| **CANopen\_drv.h** | главный заголовочный файл драйвера CANopen. |
| **clarke.h** | заголовочный файл модуля фазных преобразований. |
| **co\_ODvars.h** | объявление служебных структур драйвера CANopen. |
| **cood1.h** | объявление служебных переменных для словаря объектов 1 драйвера CANopen. |
| **DRV\_INTERFACE.h** | заголовочный файл для работы модулем банка аварий |
| **filter.h** | заголовочный файл инерционного звена первого порядка (фильтр). |
| **Global\_time.h** | заголовочный файл модуля работы с реальным временем |
| **ipark.h** | заголовочный файл модуля обратных координатных преобразований. |
| **main.h** | главный заголовочный файл проекта. |
| **mbod.h** | заголовочный файл словаря драйвера MODBUS. |
| **park.h** | заголовочный файл модуля координатных преобразований. |
| **SM\_CmdLogic.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\_ADCCalibration.h** | заголовочный файл модуля калибровки АЦП |
| **V\_AutoOffset.h** | заголовочный файл модуля автоподстройки смещения датчиков тока модуля АЦП. |
| **V\_bits\_to\_enum\_numbers.h** | заголовочный файл модуля "листания" кодов аварий при наличии нескольких аварий одновременно. |
| **V\_CANtoRS.h** | драйвер для работы CANopen драйвера через RS (UART) и USB. |
| **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\_DIO.h** | заголовочный файл модуля обработки дискретных входов/выходов |
| **V\_DPR\_eCAP.h** | заголовочный файл модуля расчёта скорости и положения по трём дискретным датчикам (например, на элементах Холла). |
| **V\_event\_log.h** | заголовочный файл модуля сохранения событий и времени возникновения в энергонезависимую память |
| **V\_Excitation.h** | заголовочный файл модуля обмотки возбуждения. |
| **V\_Fan.h** | заголовочный файл модуля управления вентилятором охлаждения |
| **V\_fifo.c** | заголовочный файл модуля работы с кольцевым буфером типа FIFO |
| **V\_hzprof.h** | заголовочный файл кривой U(f). |
| **V\_led.h** | заголовочный файл управления светодиодной индикацией. |
| **V\_MBVarsConv.h** | заголовочный файл вспомогательного модуля драйвера MODBUS для конвертации данных. |
| **V\_ModBus.h** | заголовочный файл драйвера MODBUS. |
| **V\_pid\_reg3.h** | заголовочный файл ПИД-регулятора. |
| **V\_PWM\_Module.h** | заголовочный файл модуля реализации ШИМ. |
| **V\_QEP.h** | заголовочный файл модуля оценки скорости и положения при помощи eQEP. |
| **V\_relay\_reg.h** | заголовочный файл релейного регулятора. |
| **V\_rmp\_ctrl.h** | заголовочный файл задатчика интенсивности. |
| **V\_rotor\_observer.h** | заголовочный файл модуля наблюдателя потокосцепления ротора для асинхронного двигателя. |
| **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. |
| **V\_UserMemory.h** | заголовочный файл работы с энергонезависимой пользовательской памятью. |
| **V\_UdControl.h** | заголовочный файл модуля плавного заряда звена постоянного тока силового преобразователя |
| *Каталог* ***Vsrc*** | |
| **CANBlockTransferInterface.c** | файл с пользовательскими функциями интерфейса блочной передачи данных драйвера CANopen. |
| **CANopenUDfuncs.c** | пользовательские функции драйвера CANopen (обращение к памяти, callback и т.п.). |
| **clarke.c** | модуль фазных преобразований. |
| **сood1.c** | файл словаря объектов драйвера CANopen второго модуля CAN. |
| **DRV\_INTERFACE.с** | Модуль банка аварий |
| **filter.с** | модуль инерционного звена первого порядка (фильтр). |
| **ipark.c** | модуль инверсных координатных преобразований. |
| **main.c** | главный файл проекта. Содержит main(), а также обработчики прерываний. |
| **mbod.c** | файл словаря объектов драйвера MODBUS. |
| **park.c** | модуль координатных преобразований. |
| **SM\_CmdLogic.c** | обработка команд и заданий. |
| **SM\_Ctrl.c** | основной дискретный автомат 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\_ADCCalibration.с** | модуля калибровки АЦП |
| **V\_AutoOffset.c** | модуль автоподстройки смещения датчиков тока модуля АЦП. |
| **V\_bits\_to\_enum\_numbers.c** | модуль "листания" кодов аварий при наличии нескольких аварий одновременно. |
| **v\_call\_back.c** | Файл функций, вызываемых драйвером CANOpen |
| **V\_CANtoRS.c** | драйвер для работы CANopen драйвера через RS (UART) и USB |
| **V\_CurPar.c** | расчет наблюдаемых текущих параметров. |
| **v\_data\_log.c** | 4-х канальный логгер для осциллографирования в реальном времени. |
| **V\_DPR\_eCAP.c** | драйвер датчика положения ротора на трех дискретных элементах (например, Холла). |
| **V\_event\_log.с** | модуля сохранения событий и времени возникновения в энергонезависимую память |
| **V\_excitation.c** | модуль управления обмоткой возбуждения. |
| **V\_Fan.h** | модуль управления вентилятором охлаждения |
| **V\_hzprof.c** | кривая U(f). |
| **V\_led.c** | управление светодиодной индикацией. |
| **V\_MBVarsConvert.c** | файл вспомогательного модуля драйвера MODBUS для конвертации данных. |
| **V\_ModBus.c** | драйвер MODBUS. |
| **V\_pid\_reg3\_pos.c** | ПИД-регулятор для контура положения. |
| **V\_pid\_reg3.c** | ПИД-регулятор общего назначения. |
| **V\_PWM\_Module.c** | модуль реализации ШИМ. |
| **V\_QEP.c** | модуль оценки скорости и положения при помощи eQEP. |
| **V\_relay\_reg.c** | модуль релейного регулятора. |
| **v\_rmp\_ctrl.c** | задатчик интенсивности. |
| **V\_rotor\_observer.c** | модуль наблюдателя потокосцепления ротора для асинхронного двигателя. |
| **V\_RTC\_Clock.c** | модуль работы с часами реального времени. |
| **V\_SSI\_Encoder.c** | модуль работы с датчиком положения ротора с SSI интерфейсом. |
| **V\_UdControl.с** | модуль плавного заряда звена постоянного тока силового преобразователя |
| **V\_UserMemory.c** | модуль работы с энергонезависимой пользовательской памятью. |

# Архитектура программного обеспечения

## Наименование переменных, констант и типов данных

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*;

TSM\_Ctrl *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\*); // Функция относится к модулю DPReCAP

void AdcDrv\_slow\_calc(TAdcDrv\*); // Функция относится к модулю AdcDrv

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

**typedef** **struct**

{

**long** speed\_ref;//!< Задание скорости

**long** Iq\_ref;//!< Задание токоограничения для векторной системы управления

} TRefs;

## Модульность программы

В проекте не используются классы Си++ из-за различных причин, таких как:

1. Меньшая прозрачность генерируемого компилятором ассемблерного кода.
2. Проблемы переносимости ПО между разными компиляторами, проблемы с операцией взятия адреса метода класса (реализуется в разных компиляторах по-разному, с разным синтаксисом, где-то не поддерживается вообще).
3. Нежелательность использования операций динамического выделения памяти во встраиваемых системах управления: одна из сильных сторон Си++ оказывается не задействована.

Вследствие этого модульность программы достигается средствами языка Си. "Модуль" в терминах языка Си представляет собой структуру, внутри которой размещаются необходимые переменные модуля, а также указатели на функции модуля, предназначенные для "внешнего" использования – имитация класса Си++ средствами Си. Сама реализация функций находится в отдельном файле, а имена функций имеют префиксы, содержащими имя модуля, чтобы гарантированно не пересекаться в названии с другими функциями проекта. В саму же структуру модуля указатели на эти функции добавляются с короткими и простыми именами.

Как уже было сказано, обычно модуль состоит из двух файлов – заголовочного \*.h файла, в котором определяется структура, её инициализатор, а также \*.c файла, в котором реализуются «методы» модуля, т.е. содержатся функции работы с этим модулем. В каждую такую функцию помимо остальных аргументов, обязательно передается указатель на экземпляр структуры модуля, чтобы обращаться к членам конкретного экземпляра такого «самодельного» класса. Пример:

Заголовочный файл модуля цифрового фильтра:

**struct** SFilter

{

**long** input;//! Вход

**long** output;//! Выход

**long** T; //! T=Ts/Tfiltra где - Tfiltra постояннаЯ времени фильтра

**void** (\*calc)(**volatile** **struct** SFilter\*); //указатель на функцию расчета

};

**typedef** **volatile** **struct** SFilter TFilter;

//Инициализатор структуры

**#define** **FILTER\_DEFAULTS** {0,0,\_IQ(0.01),TFilter\_Calc} //Функция расчета

**void** **TFilter\_Calc**(TFilter\*);

Файл с реализацией функций модуля:

**#include** "V\_filter.h"

//! Инерционное звено 1/(Tp+1)

//! Используется как фильтр первого порядка

**void** **TFilter\_Calc**(TFilter \*p) {

p->output = **\_IQmpy**(p->T,(p->input-p->output)) + p->output;

}

Для использования модуля фильтра пользователь должен объявить в основной программе переменную (или несколько, если необходимо) с типом TFilter и присвоить ей инициализатор **FILTER\_DEFAULTS**:

TFilter *myFilter1* = **FILTER\_DEFAULTS**;

TFilter *myFilter2* = **FILTER\_DEFAULTS**;

Затем для работы фильтра нужно с определенной частотой вызова фильтра (например, 10кГц) передавать фильтруемые данные в переменную input, вызывать функцию расчета через указатель calc, а затем забирать отфильтрованный сигнал с output:

*myFilter1*.input=*ADCInput1*;

*myFilter1*.calc(&*myFilter1*);

*filteredADCInput1*=*myFilter1*.output;

Каждый модуль может содержать различные функции, но имеется следующее соглашение на именование: указатель на функцию (далее – «метод» в терминологии объектно-ориентированного программирования) инициализации должен называться init, а методы, выполняющие однотипные действия и вызывающиеся с заданной дискретностью по времени должны иметь название calc. Например, так может называться метод обработки датчика положения ротора, обработки модуля защит, обработки АЦП и так далее. Данные методы ничего не принимают (кроме указателя на экземпляр своего модуля) и не возвращают в явном виде, но результаты их работы как-то отражаются в переменных структуры их модуля или в работе периферии. Если в модуле имеется несколько методов, которые надо вызывать с разной дискретизацией по времени, то имя метода дополняется пояснением, с какой частотой его необходимо вызывать. Например, метод calc\_10k, calk\_1k и т.п. Если разработчик модуля не имеет строгих требований к частоте вызова, но хочет подчеркнуть, что одна функция расчета выполняется значительно дольше или реже другой, то допустимы наименования fast\_calc и slow\_calc. Если метод модуля выполняет одну конкретную операцию, например, отправку данных или расчет частоты вращения, то допускается более конкретное наименование метода, например, send или rotorSpeedCalc.

## Дискретные автоматы

Дискретные автоматы (ДА) – или, как их чаще называют, машины состояния, конечные автоматы, state machines – широко применяются в данном проекте. Простыми словами, дискретный автомат – это такой подход к программированию, где программу разбивают на фиксированное число состояний, в каждом из которых выполняются одни и те же однотипные действия, а переход между состояниями выполняется по каким-то событиям. Программа дискретного автомата ни в коем случае не зацикливается внутри каждого состояния, а, выполнив нужные действия, возвращает управление, чтобы через некоторое время быть вызванной снова. В зависимости от сложности ПО в системе управления может работать сразу несколько ДА.

Легко представить дискретный автомат защит, который в состоянии «норма» каждый раз проверяет наступление аварийных событий, а при их наступлении переводит автомат в состояние «авария». В состоянии «авария» дискретный автомат выключает и обесточивает электрооборудование и ожидает команду сброса аварии от пользователя. Получив такую команду ДА переходит в состояние «сброс аварии», где производит действия по обнулению флагов аварии, после чего автоматически переходит в состояние «норма». В зависимости от требуемого быстродействия реакции на аварию подобный дискретный автомат должен вызываться с определенной частотой, скажем 10кГц, выполнен в виде отдельного программного модуля и выполнятся постоянно независимо от работы других модулей программы.

Помимо действий, которые выполняются в стационарном состоянии дискретного автомата, часто требуется выполнить какие-то действия однократно при смене состояния дискретного автомата. Например, при попадании в состояние «норма» дискретного автомата защит можно произвести сброс команд на запуск привода, если они были поданы за время нахождения привода в состоянии аварии, чтобы предотвратить неожиданный пуск. Для этой цели используется переменная entry (для краткости E), которая служит флагом первого вхождения в состояние. Типичный для данного проекта синтаксис дискретного автомата на два состояния приведен ниже, где p это указатель на текущий модуль:

//считаем дискретный автомат

**if** (p->state\_prev != p->state) //смена состояния?

p->E = 1;//Возводим флаг первого вхождения "entry"

**else**

p->E = 0;//иначе сбрасываем

p->state\_prev = p->state;

**switch** (p->state) {//в зависимости от текущего состояния

**case** **PROT\_ON\_OK**: {//Норма

**if** (p->E == 1) { //Первое вхождение

*cmd*.all = 0;// обнулим командное слово

}

//есть аварии?

**if** ((p->masked\_bit\_fault1 | p->masked\_bit\_fault2) != 0) {

p->state = **PROT\_FAIL**; //переходим в состояние аварии

}

**break**;

}

**case** **PROT\_FAIL**: { //состояние аварии (сработала защита)

**if** (p->E == 1) {//Первое вхождение

*pwm12*.Off(&*pwm12*); //Выключение ШИМ.

}

**if** (*cmd*.bit.trip\_reset == 1)//команда на сброс аварии

p->state = **PROT\_ON\_OK**;//идем в "норма"

**break**;

}

}

Подробнее о дискретных автоматах можно прочитать в общеобразовательных статьях, которые можно в большом количестве найти в сети Интернет по запросу «конечные автоматы на практике». Например, статья Козаченко В.Ф. «Эффективный метод программной реализации дискретных управляющих автоматов во встроенных системах управления» (<http://motorcontrol.ru/wp-content/uploads/2015/11/state_mashine.pdf>).

## Прерывания и их обслуживание

В проекте ПО используется ряд прерываний микроконтроллера. Все прерывания системы (за исключением CAN), конфигурируются в модуле SM\_Sys. Существует два прерывания для выполнения задач реального времени. В них выполняется большая часть программного кода. Эти два прерывания следующие:

1. **Прерывание «10кГц».** Таймер общего назначения CPUTimer0 используется для создания прерывания с частотой 10кГц. В обработчике этого прерывания рассчитывается основная часть системы управления: модули, реализующие саму структуру системы управления, модуль ШИМ, модуль защит, модуль датчика положения ротора и т.п. При этом частота ШИМ инвертора может быть любой и не привязана к данному прерыванию.
2. **Прерывание «1кГц».** Используется Таймер 4 менеджера событий. Прерывание необходимо для задач, не требующих большой скорости выполнения, однако требующих строгую дискретизацию по времени. В нем обрабатываются различные сетевые интерфейсы и различные дискретные автоматы.

Второй таймер общего назначения CPUTimer1 используется для расчета длительности обработки прерываний. Счетчик таймера тактируется с частотой работы ядра микроконтроллера. 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.

В системе прерываний ПО реализована явная иерархия по приоритету прерываний. Например, прерывание 10кГц должно быть строго приоритетнее прерывания 1кГц. Если бы было наоборот, то, 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.

На Рис. 3.1 показана диаграмма прерываний, их приоритетов и решаемых в них задач.



Рис. . Распределение задач по прерываниям

Существует немного другой, также широко распространенный подход для организации прерываний для систем управления электродвигателями: создается прерывание на таймере, ответственном за генерирование ШИМ, в которое (вместо прерывания 10кГц данного ПО) помещается весь программный код структуры управления электродвигателем. В этом подходе есть как плюсы, так и минусы.

Плюсы:

1. Управляющее воздействие на задание напряжения формируется с той же частотой, с которым его может воспроизвести инвертор: это гарантирует максимально возможное быстродействие системы управления.
2. Отсутствие биений на квазичастоте по сравнению с решением асинхронного прерывания. В случае близких, но не равных частот ШИМ и расчета системы управления формирование задания напряжения будет все время «плыть» относительно частоты ШИМ инвертора, внося переменную фазовую ошибку в реализацию вектора напряжения. При синхронной частоте расчета СУ и ШИМ такой проблемы нет.

Минусы:

1. Строгие ограничения на допустимую частоту ШИМ, чтобы гарантированно «успеть» произвести расчет системы управления на этой частоте. Такой подход часто невозможен для высоких частот ШИМ порядка нескольких десятков кГц.
2. Проблемы с изменением частоты ШИМ при работе: необходимо пересчитывать все коэффициенты интегрирования, коэффициенты регуляторов и производных дифференциальных уравнений, так как частота расчета модулей меняется.
3. Сложность в отладке при возможности изменения частоты ШИМ: необходимо проводить специальные тесты на максимальной частоте ШИМ, чтобы убедиться в достаточной производительности ПО при расчете системы управления на такой частоте.

Так как данный проект ориентирован на общую задачу электропривода без привязки к конкретному инвертору, частоте ШИМ и требований, то был выбран асинхронный подход в реализации прерываний расчета системы управления и ШИМ.

## Управление распределением памяти

Для разметки памяти микроконтроллера в среде CCS используется файл компоновки памяти f2810\_nonBIOS\_flash.cmd. Файл является стандартным для данного типа микроконтроллеров. При необходимости файл можно редактировать и перераспределять память согласно требованием разрабатываемого программного обеспечения.

## Организация объявления глобальных переменных

В начале файла main.c объявляются все программные модули и их инициализаторы. В комментариях написано назначение каждого модуля. Эти модули объявлены как со спецификатором extern в заголовочном файле main.h. Подключив такой заголовочный файл во все файлы программы можно обращаться к любому модулю программы из любого места – модулям АЦП, ШИМ, модулю защит и т.п.

Такой подход имеет свой плюс в простоте, однообразности, но имеет минус во времени компиляции и зависимости каждого файла от общего заголовочного файла main.h (теряется локальность модулей).

## Ход выполнения программы

Выполнение программы для проекта в среде CCS начинается с функции main.

В начале функции main вызываются функции инициализации тактирования и подготовки микроконтроллера к работе, а затем вызывается функция инициализации главного модуля-обертки всего проекта *sm\_sys*.init, в котором вызываются функции инициализации всех остальных модулей проекта.

Внутри данной функции вызываются необходимые функции инициализации всех других модулей программы. Далее разрешаются прерывания макросом EINT, после чего в основном файле запускается бесконечный фоновый цикл **while(1)**. После этого этапа ход основной программы идет главным образом тремя путями: в прерывании 1кГц, 10кГц, и вне их в фоновом цикле. Вызовы всех модулей этих трех «потоков» также осуществляются через модуль-обертку *sm\_sys*. По событиям могут также возникать другие прерывания (CAN, датчик положения и т.п.).

В файле main.c после функции **main** также определяются функции-обработчики, вызываемые практически по всем прерываниям в системе (исключение – драйвер CANopen, который содержит обработчик прерывания в своем файле).

Все структуры управления электродвигателями реализованы в файле SM\_Ctrl.c в виде дискретного автомата, каждому состоянию которого соответствует та или иная структура управления. Например, структура скалярного управления синхронным и асинхронным двигателем реализуется в состоянии, определяемом константой **CTRL\_RUN\_U2F**, принимает на вход задание частоты вращения, модулем задатчика интенсивности формирует плавную выходную частоту, модулем кривой U(f) находит соответствующее этой частое напряжение, формирует интегратор углового положения для вектора напряжения, при помощи модуля координатных преобразований переводит амплитуду и угол вектора в задание напряжения в декартовых осях и передает задания в модуль ШИМ. Все эти состояния и структуры управления более подробно описаны. Сейчас необходимо отметить, что структура управления двигателем набирается из отдельных готовых модулей, как из конструктора, и не содержит в своем теле других сервисных функций (защиты, обработки датчиков, дискретных входов и т.п.) – эти функции реализованы отдельно и независимо. После этапа инициализации состояние дискретного автомата равно **CTRL\_STOP**, что соответствует отключенному приводу.

Для обработки команд управления пользователя используется дискретный автомат команд управления, реализованный в файле SM\_CmdLogic.c. Его обработка происходит в прерывании 1кГц, он принимает команды запуска и останова, а также следит за состоянием ДА защит (SM\_Protect.c), управляя при этом автоматом структур управления SM\_Ctrl, запуская его и останавливая по необходимости.

Другие модули программы по большей части работают автономно и рассчитываются постоянно, независимо от того, работает привод или нет. К таким модулям можно отнести:

* Коммуникационные драйверы протоколов CANopen и MODBUS.
* Модуль АЦП (файл V\_adc.c).
* Модуль защит (SM\_Protect.c).
* Модуль расчета текущих параметров (V\_CurPar.c).
* Модуль цифрового осциллографа (V\_data\_log.c).
* Модули обработки датчиков положения (V\_DPR\_eCAP.c, V\_SSI\_Encoder.c, V\_QEP.c).
* Модуль управления светодиодами (V\_led.c).
* Модуль часов реального времени (V\_RTC\_Clock.c).

## Введение в CANopen

В ПО данного контроллера используется протокол CANopen для связи с ПК (программа RTCON) или другими устройствами. Связь может осуществляться штатно через CAN интерфейс, однако также в данном ПО существует драйвер V\_CANtoRS.c, осуществляющий взаимодействие по CANopen через последовательные интерфейсы UART и USB. Подробное описание драйвера CANopen приведено в документе **«Описание драйвера CANopen»**. Здесь будут описаны основные положения, которое необходимо знать для начала работы с ПО. 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 expedited SDO – подтверждаемые посылки с 8-ю байтами данных. С помощью них можно как читать, так и записывать данные. Посылки SDO можно послать любому узлу в сети. Посылки содержат номер адресата, данные, контрольную сумму, адрес объекта (индекс, подындекс) и ряд служебных данных. Используются для конфигурирования и пусконаладки устройства.
* Реализует сервис PDO – неподтверждаемые посылки с 8-ю байтами данных. Данный сервис используется для обмена оперативной информацией между устройствами в рабочем режиме (реализации распределенной системы управления).
* Реализует сервис Heartbeat – не подтверждаемые посылки, сигнализирующие о наличии устройства в сети и передающие его статус.
* 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.
* Каждый объект имеет формат данных, в котором 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 (RTCON) и т.п.). Это сделано для экономии свободной памяти контроллера. Сам текст – это «название» объекта, понятное пользователю и отображаемое как имя параметра.
* Для дополнительной группировки индексов словаря служащих одной задаче были введены группы параметров. 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 внешнее устройство могло 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.
* После загрузки словаря объектов он сохраняется на управляющем устройстве (программа RTCON сохраняет словарь в файл), 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, чего достаточно для реализации любого проекта.
* 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 флеш памяти. После какой-либо перенастройки привода пользователь с помощью ПУ должен дать команду сохранения, после которой все значения словаря объектов сохраняются на флеш. 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 (RTCON) 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 пользоваельской энергонезависимой памяти драйвер CANopen проверяет общую контрольную сумму загруженных данных. Если она не сходится, то драйвер загружает параметры по умолчанию. Не сходиться она может по двум причинам: произошла какая-то ошибка (данные перетерты, отказ работы модуля памяти), а также когда 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 CAN определяются в файле cood1.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 разработано специальное ПО COODEdit редактор словарей объектов CANopen. Описание на данное ПО можно найти в файле «**Описание программы COODEdit»** или в директории с самой программной.

Если требуется найти в ПО контроллера переменную, соответствующую какому-либо параметру на пульте управления (или программе RTCON), это легко сделать с использованием файла cood1.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 или по номеру индекса.

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

## Форматы вычислений и единицы измерения

Практически весь проект построен на использовании целочисленных вычислений с фиксированной точкой в формате IQ8.24 со знаком, где 8 разрядов отводится под целую часть, а 24 под дробную. Для обеспечения таких вычислений реализована библиотека IQmath, содержащая функции умножения, деления, синуса и т.п. в указанном формате.

Весь проект реализован с использованием относительных единиц (о.е.) измерения. 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 по умолчанию в данном проекте равен 5А. Таким образом, если переменная, в которой хранится значение тока в программе равна единице (в о.е.), это обозначает ток в 5А. Такой подход является классическим для встраиваемых систем управления и выбран для исключения проблем переполнения переменных или нехватки точности. При изменении мощности преобразователя достаточно переопределить базовые параметры (скажем, увеличить базовый ток в 100 раз), при этом значения переменных внутри системы управления останутся в тех же рабочих диапазонах.

Обычно базовая величина выбирается близкой к номинальному значению параметра, чтобы обеспечить как запас "вверх" по переполнению, так и вниз по точности. Т.е. стремятся, чтобы относительные единицы были близки к единице при работе системы управления с номинальными значениями. Однако в силу формата 8.24 вполне допустимо в 10-20 раз отойти от базы без последствий в виде переполнения или значительной потери точности.

Программа RTCON, 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 RTCON, наоборот, поделит введенное значение на базовую величину и отправит данные в микроконтроллер. Посмотреть истинное значение в IQ формате можно выбрав пункт «Инфо» через контекстное меню интересующего параметра в RTCON.

Список базовых значений можно посмотреть в группе параметров RTCON **Базов. парам**. После изменения базовых параметров необходимо дать команду на сохранение значений параметров в энергонезависимую память (через программу RTCON), а затем перезагрузить ПО микроконтроллера, чтобы при инициализации были применены новые базовые коэффициенты. Также необходимо дать команду RTCON на обновление словаря объектов CANopen, чтобы RTCON перечитал новые значения базовых параметров и корректно отображал физические единицы.

Нужно помнить, что при изменении базового коэффициента все уставки и защиты в физических единицах пропорционально изменятся. Например, если защита по максимальному току была равна 2.0 в о.е. (удвоенное значение базы), то при повышении базового тока в десять раз она все равно останется равной 2.0 в о.е., но будет соответствовать в 10 раз большему току в физической величине. Тоже самое касается частот вращения и остальных величин.

Благодаря применению вычислений в относительных единицах система управления практически нечувствительна к изменению базовых единиц и не использует их в своих расчетах. Исключением является ввод аналоговых данных (оцифрованных данных АЦП: измерение токов, напряжений). Для преобразования аналоговых сигналов в относительные единицы системы управления в модуле АЦП (V\_adc.c) используются как значение базовых величин параметра, так и коэффициенты настройки аналоговых датчиков, показывающих, сколько единиц физической величины соответствует максимальному измеряемому коду АЦП. Кроме того, базовые единицы частоты вращения используются при инкрементировании углового положения для формирования вращения вектора тока/напряжения.

# Описание модулей ПО

## 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 несколько видов структур управления для различных типов двигателей, каждая из которых реализована в виде отдельного состояния дискретного автомата. Желаемое состояние дискретного автомата, в которое он переходит после запуска привода, задается в переменной (**Настройки СУ.режим работы**). Текущее состояние дискретного автомата доступно в параметре **ДА sm\_ctrl.sm\_state [510Bh.01h].**

Реализованы следующие структуры управления приводами:

|  |  |  |
| --- | --- | --- |
| Константа в файле build.h | N | Описание |
| CTRL\_RUN\_U2F | 3 | Скалярная структура обеспечивает вращение вектора напряжения с заданной частотой и амплитудой, определяемой в зависимости от частоты по закону U(f). Подходит для синхронного и асинхронного двигателей. |
| CTRL\_RUN\_I2F | 4 | Частотно-токовая структура обеспечивает вращение вектора тока с заданной частотой и поддержание его на заданном уровне амплитуды посредством регуляторов тока. Подходит для синхронного и асинхронного двигателей. |
| CTRL\_FLUXING | 2 | Структура поддержания постоянного тока удобна для запуска синхронного двигателя, а также для настройки регуляторов тока синхронного и асинхронного двигателей. |
| CTRL\_RUN\_VECTOR\_SM | 5 | Структура векторного датчикового управления для синхронного двигателя, имеющая контуры тока и скорости. |
| CTRL\_RUN\_VECTOR\_SM\_ENCODER\_POS | 10 | Структура векторного датчикового управления для синхронного двигателя имеющая контуры тока, скорости, положения: обеспечивает отработку приводом заданного положения. |
| CTRL\_AUTO\_SENSOR\_TUNING | 29 | Структура для автоматической настройки смещения датчиков положения на элементах Холла и энкодера синхронного двигателя, имеющая в основе частотно-токовую структуру управления. |

Более подробно все структуры управления будут описаны ниже.

### *Структура скалярного управления* (режим работы = 3)

Режим работы U/f обеспечивает традиционное скалярное управление АД/СД с регулированием напряжения по заранее заданной кривой *U/f*. Настройка кривой (подробнее модуль рассмотрен в п. 4.3) ограничена двумя опорными точками через следующие параметры группы **Кривая U-f**: **Нулевая опорная частота**, **Нулевое опорное напряжение**, **Первая опорная частота**, **Первое опорное напряжение**. Разгон производится от задатчика интенсивности ЗИ (подробнее рассмотрен в п. 4.4). Структура скалярной СУ представлена на Рис. 4.1.



Рис. . Структура системы скалярного управления

Заданная частота вращения поступает на задатчик интенсивности (ЗИ), который формирует плавное изменение выходной частоты. Выход ЗИ поступает как на блок кривой U/f, который табличным методом по текущей частоте вращения вычисляет необходимую амплитуду вектора напряжения, так и на интегратор угла положений, который формирует заданный угол вектора напряжения. Далее вектор напряжения в полярной системе координат (амплитуда и угол) преобразуются в прямоугольную систему координат в виде двух заданий напряжения Uα\_зад и Uβ\_зад, которые поступают на модуль ШИМ (см. п. 4.5). Модуль ШИМ управляет шестиключевым инвертором напряжения таким образом, чтобы реализовать заданный вектор напряжения. Какая-либо обратная связь в структуре отсутствует: датчики токов фаз используются только для аварийного отключения при превышении уставки (см. модуль защит п. 4.10).

### Структура частотно-токового управления (режим работы = 4)

Служебный режим, в котором происходит вращение вектора тока статора (поля двигателя) с фиксированной амплитудой и заданной 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. Структура управления по сути формирует из инвертора напряжения инвертор тока программным способом. Исторически частотно-токовой системой управления обычно называют другую, более сложную структуру управления на основе инвертора тока, предназначенную исключительно для асинхронного двигателя и стремящуюся приблизиться по своим показателям к векторной структуре управления. Однако в данном документе под частотно-токовым управлением будет иметься ввиду прямое управление частотой и амплитудой вектора тока двигателя по заданию пользователя без каких-либо других законов регулирования.

Данная структура «инвертора тока» подходит как для синхронной, так и асинхронной машины. Ротор машины при этом должен увлекаться полем статора и вращаться за ним: у синхронной машины должен вращаться синхронно с полем статора, а у асинхронной «проскальзывать», т.е. вращаться чуть медленнее в двигательном режиме. Для синхронной машины 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.

В эксплуатации такой режим используется редко, так как двигатель с ним работает нестабильно, режим уступает по возможностям режиму векторного управления. В некоторых случаях похожий тип управления применяется для синхронных серводвигателей и называется микрошаговым управлением. Также пуск двигателя в таком режиме (как синхронной, так и асинхронной машины) часто применяется как начальный этап запуска бездатчиковой векторной системы управления, когда положение магнитного потока ротора двигателя по ЭДС не может быть вычислено на низкой скорости.

На Рис. 4.2 приведена структура системы управления в данном режиме.



Рис. . Структура системы частотно-токового управления

Задание частоты вращения вектора тока двигателя **Задания.Скорость** поступает на задатчик интенсивности ЗИ (п.4.4), 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 двухфазную декартову систему координат *α, β*. Ток фазы *c* вычисляется по первому закону Кирхгофа (обмотки трехфазного 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 (п. 4.5).

Задание на амплитуду тока (**Задания.Ток удержания**) подается на один из регуляторов, в данном случае на регулятор тока (п. 4.2) 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 никак не зависят от фактического положения ротора, как это обеспечивается при векторном управлении, а лишь поворачиваются с заданной скоростью независимо от состояния машины и положения её ротора.

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* (режим работы = 2)

Этот режим соответствует режиму фиксации двигателя под током и желателен перед переходом в режим частотно-токового управления (режим работы 4, вращение вектора тока с фиксированной амплитудой и нарастающей частотой). Дли синхронной машины он необходим для того, чтобы ротор двигателя предварительно повернулся и зафиксировался в определенном механическом состоянии: дождаться затухания механических колебаний. При этом последующее начало вращения вектора тока в режиме частотно-токового управления выполняется с заданной фазы и сопровождается меньшими колебаниями ротора. Для асинхронной машины в этом состоянии можно дождаться нарастания потока возбуждения перед началом вращения.

Также данная структура удобна для настройки регуляторов тока путем наблюдения осциллограммы реакции на скачок задания.

Структурная схема системы управления в данном режиме аналогична схеме частотно-токового управления (рассмотренной в 4.1.2) за одним исключением: угол положения вектора тока Θ неизменен и задается параметром **Задания.theta\_elec [5112h.04h].**

### Структура векторного датчикового управления синхронным двигателем (режим работы = 5)

Основной недостаток скалярного управления заключается в малом быстродействии, как при регулировании скорости, так и при регулировании момента. Это объясняется тем, что для перехода в некоторый установившийся режим требуется завершение процессов распределения моментообразующего тока и потокообразующего тока, которое в системах скалярного управления происходит само по себе через некоторое время. Именно это время переходных процессов и определяет ту плохую динамику, которая присуща электроприводу со скалярным управлением. Однако если не пускать переходные процессы в двигателе на «самотек», можно избежать перераспределения токов и связанных с этим переходных процессов и контролировать состояние электрической машины в каждый момент времени, увеличив быстродействие системы регулирования. Так как требуется регулирование двух ортогональных токов, один из которых является потокообразующим, а другой моментообразующим, то система управления была названа векторной.

Под системами векторного управления принято подразумевать системы, работающие при постоянстве потокосцепления ротора, хотя система управления может быть построена и для постоянства потокосцепления статора или общего потока машины.

Таким образом, векторное управление позволяет управлять непосредственно моментом двигателя, наиболее рационально использовать возможности привода, контролировать все переходные процессы и не «пускать их на самотек». Для синхронных приводов с цифровой системой управления такой режим работы является наиболее применяемым на практике.

Если записать уравнение момента синхронного двигателя с постоянными магнитами (СДПМ) в осях d, q, получим следующее выражение:



Приравняв , получим:



где Ψ*PM* — составляющая потокосцепления статора от постоянного магнита на роторе,

M – момент двигателя,

*isq* – ток статора по оси q, перпендикулярный направлению потока постоянных магнитов,

*isd* – ток статора по оси d, сонаправленный потоку постоянных магнитов,

*Ls* – индуктивность статора.

Получается, наилучшие энергетические показатели будут достигаться при таком управлении, когда ток , тогда в двигателе будет лишь моментобразующий ток и омические потери будут минимальными. 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).

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.

На Рис. 4.3 показана структура системы датчикового векторного управления синхронным двигателем с постоянными магнитами (СДПМ). Задание частоты **Задания.Скорость** (*ω\*зад* на Рис. 4.3) поступает на задатчик интенсивности ЗИ (п. 4.4), который обеспечивает линейный разгон. С его выхода задание частоты поступает на сумматор ПИД регулятора скорости (п. 4.2), в который также заводится обратная связь по скорости с датчика положения ротора (п.4.7 или п. 4.8). 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.



Рис. . 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 (модуль ШИМ, п. 4.5). 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 получаются 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*.

Данная структура может использовать разные типы датчиков положения. Выбор типа датчика положения осуществляется в переменной **Базов. парам. Тип датчика положения [5113h.09h]**.

Также следует отметить, что выход регулятора скорости (задание на ток/момент) имеет ограничения: **Рег. скорости.Максимум выхода [5101h.04h]** и **Рег. скорости.Минимум выхода [5101h.05h]**. Установка минимума и максимума происходит автоматически в зависимости от задания тока для векторного управления в параметре **Задания.Макс. ток ВУ**. При этом по умолчанию минимум установлен нулем, чтобы исключить отрицательный момент двигателя, а максимум устанавливается равным значению этого параметра.

### Структура векторного датчикового управления для синхронного двигателя с дополнительным внешним контуром положения (режим работы = 10)

Структура позиционирования (Рис. 4.4) при помощи дополнительного внешнего контура положения на основе структуры векторного управления, описанной выше в п.п. 4.1.4, реализует поддержание приводом заданного пользователем положения ротора *θ\*зад*. Регулятор положения (РП) имеет тип ПИД (п. 4.2), хотя дифференциальная часть может быть опущена. Из практических соображений в качестве регулятора скорости (РС) рекомендуется П-регулятор (Ки и Кд необходимо обнулить). Допустима и другая конфигурация: ПИ-регулятор скорости и П регулятор положения (в этом случае обнуляются Ки и Кд регулятора положения).



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

Структура работает при использовании ДПР типа «инкрементальный энкодер» (п. 4.8), так как точности ДПР на элементах Холла недостаточно для данного режима. Пользователь задает желаемое положение вала ротора в электрических градусах в параметре **Задания.theta\_elec**. Допустимо задавать положение не превышающее 50 оборотов: в обратной связи регулятора положения используется переменная углового положения, отслеживающая число полных оборотов. Параметр **Задания.Скорость** в данном режиме ограничивает максимальную частоту вращения (выход регулятора положения). Если регулятор скорости пропорциональный, эту переменную необходимо задавать из соображений, чтобы мог быть достигнут максимально разрешенный ток на статоре при максимальном задании скорости (на практике обычно выше, чем желаемое ограничение скорости для сервопривода). Параметр **Задания.Макс. ток ВУ** в данном режиме ограничивает максимальный моментообразующий ток статора (выход регулятора скорости). Рекуперация по умолчанию разрешена, так как без нее невозможна корректная работа сервопривода. Однако при типичных частотах работы серводвигателей реальное повышение напряжения на инверторе обычно маловероятно. В качестве обратной связи по контуру положения используется электрическое угловое положение, учитывающее совершенное валом двигателя число оборотов.

### Режим автонастройки смещений ДПР (режим работы = 29)

Режим работы по структуре полностью аналогичен структуре частотно-токового управление, 4.1.2, однако в данный режим добавлена логика автоматической подстройки смещений угловых положений датчиков положения ротора (параметры **ДПР-Холл.СмещУгол** и **ДПР-Энкодер. СмещУгол**). Смещения подстраиваются таким образом, чтобы максимуму тока фазы A соответствовало нулевое показание углового положения обоих датчиков положения. Этот режим корректно настраивает датчики положения ротора при условии отсутствия нагрузки на валу, относительно невысокой частоте вращения и правильному чередованию фаз (при задании положительной частоты вращения датчики положения показывают также положительную частоту вращения). Этот режим необходим перед первым пуском векторной структуры управления и снимает необходимость ручной настройки датчиков. Для корректной настройки двигатель должен сделать в таком режиме несколько полных оборотов.

## ПИД регулятор

Упрощенная схема ПИД-регулятора показанна на Рис. 4.5.



Рис. . Структура ПИД-регулятора

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, Kd 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 (например, в два раза). Аналогично повышать Ki до достижения приемлемого быстродействия регулятора или до появления колебательности. Попеременной настройкой коэффициентов добиться требуемого быстродействия и формы переходного процесса (технический оптимум, симметричный оптимум и т.п.). Удобнее всего настраивать регулятор наблюдая в осциллограмме реакцию на скачок задания, хотя иногда можно настроить регулятор и «на слух», определяя, когда начались колебания и снижая коэффициент.

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» от Texas Instruments, который можно легко найти в сети Интернет.

## Кривая U/f

Кривая U/f – зависимость напряжения, прикладываемого к двигателю, от частоты. 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. . 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.

## Задатчик интенсивности (ЗИ)

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



Рис. . Пуск и торможение по задатчику интенсивности

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

Темп ускорения и замедления — это время, за которое частота с нуля достигает номинальной. Т.е., если номинальная частота соответствует 100Гц, 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**.

## Модуль ШИМ

Модуль ШИМ (широтно-импульсной модуляции) предназначен для управления ключами инвертора и отработки заданного вектора напряжения. Здесь будет описана организация самого распространённого модуля ШИМ для шестиключевого инвертора напряжения и предназначенного для трехфазных синхронных и асинхронных двигателей. Модули ШИМ для ДПТ и ВИД в данном пункте рассмотрены не будут.

Задание на вход модуля поступает в виде заданий мгновенных напряжений *Uα.зад* и *Uβ.зад* и 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 напряжений задания *Uα.зад* и *Uβ.зад*.

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.

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

На Рис. 4.8 показана общая структура модуля ШИМ.



Рис. . Схема алгоритма 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 компенсации (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’ = Ualpha \* 1/[ 1+(Uизмер - Uном)\* Квых/ Uном],

Ubeta’ = Ubeta\* 1/[ 1+(Uизмер - Uном)\* Квых/ Uном],

где

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

Ualpha, Ubeta – напряжения по осям альфа, бета до блока компенсации пульсаций;

Ualpha’, Ubeta’ – напряжения по осям альфа, бета после блока компенсации пульсаций;

Uном – номинальное напряжение на звене постоянного тока;

Uизмер – измеренное напряжение на звене постоянного тока.

Следует заметить, что работа блока компенсации 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. Для более детального понимания работы блока коллинеарного ограничения вектора напряжения обратимся к рисунку ниже (справедливо только для векторной ШИМ).



Рис. . 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 B и C. 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. Например, при напряжении на звене постоянного тока 540 В , 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. 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 в нашем примере 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).

Также следует отметить, что в некоторых типах инверторов напряжения применяется так называемое «бутстрепное» питание драйверов ключей. Это значит, что для питания драйвера верхних ключей инвертора используется конденсатор, который заряжается в то время, когда открыт нижний ключ инвертора. Поэтому в схеме ШИМ для таких инверторов 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 драйвера верхних ключей останутся без питания, что в большинстве случаев приводит к срабатыванию аппаратной аварии драйверов ключей инвертора. Поэтому перед работой с инвертором следует выяснить, как у него организовано питание драйверов ключей и соответствующим образом настроить параметр **ШИМ.U\_lim**.

При этом параметр **ШИМ. U\_lim** по-разному работает для векторной и скалярной ШИМ. Для векторной ШИМ его можно задать равным 90%. Для синусоидальной ШИМ предельным работоспособным значением является ограничение 70% в силу более низкого выходного напряжения в данном типе модуляции.

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 (синусоидальный или векторный). Блок управляется параметрами:

* **ШИМ.Тип ШИМ**: выбирает синусоидальный или 6-ти секторный векторный алгоритм ШИМ;
* **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.

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

Скважности синусоидальной ШИМ в относительных единицах рассчитываются по формулам:





В формулах принимается, что напряжения *Uα.зад* и *Uβ.зад* заданы в относительных единицах. Первая система уравнений пересчитывает задание из двух фаз в три фазы, получая заданные напряжения для фаз A,B,C. Вторая система уравнений рассчитывает значения скважностей на инверторе. Формулы интуитивно понятны: скважность отклоняется от значения 50% пропорционально напряжению, требуемого для конкретной фазы.

Для шестисекторной векторной ШИМ скважности рассчитываются по более сложному методу, предложенному к.т.н. Изосимовым Д.Б. (ИПУ РАН). Метод подробно рассмотрен в диссертационной работе Чуева Павла Вячеславовича «Разработка систем векторного управления асинхронными приводами на базе специализированных сигнальных микроконтроллеров», которая доступна, например, по электронному адресу <http://motorcontrol.ru/wp-content/uploads/2015/11/Chuev_vector_control.pdf> , глава 3.

Синусоидальная ШИМ, в отличие от векторной, управляет всеми ключами инвертора на периоде ШИМ (больше потери), имеет на 15% более низкое доступное выходное напряжение по сравнению с векторной ШИМ, но имеет меньшую степень подверженности влиянию мертвого времени инвертора. Известен метод ШИМ, сочетающий преимущества обоих методов и рассмотренный в статье Анучин А.С. «ШИРОТНО-ИМПУЛЬСНАЯ МОДУЛЯЦИЯ МЕТОДОМ РЕАЛИЗАЦИИ МГНОВЕННЫХ ФАЗНЫХ ПОТЕНЦИАЛОВ ДЛЯ ТРЕХФАЗНЫХ ИНВЕРТОРОВ НАПРЯЖЕНИЯ», которая доступна по электронному адресу <http://motorcontrol.ru/wp-content/uploads/2015/11/PW1.pdf> . Однако в данном ПО такой метод ШИМ не реализован.

После расчета скважностей на три стойки ключей инвертора в относительных единицах, они умножаются на период несущего таймера ШИМ и загружаются в регистры сравнения модулей ШИМ.

Далее рассчитанные скважности стоек инвертора поступают на блок ограничения минимальной скважности, реализованный в микроконтроллере аппаратно. Прикладывать к ключам управляющий сигнал, сравнимый с временем их коммутации в ряде случаев нежелательно (для мощных инверторов), нужно искусственно ограничить минимальные 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 ограничения минимальной скважности не используется. После всех описанных преобразований скважности GammaA, GammaB, GammaC попадают непосредственно на выход контроллера в виде ШИМ-сигнала и управляют 6ю ключами инвертора.

## Модуль АЦП

Модуль реализован в файле V\_adc.c в виде модуля с названием TAdcDrv. Модуль измеряет токи фаз двигателя (в общем случае токи фазы A, B, C, D, но в зависимости от типа аппаратной части могут отсутствовать те или иные датчики) и напряжение звена постоянного тока инвертора. Параметры отображаются в группе АЦП.

Для совместимости программного кода с разными типами инверторов применяется лишь одно измерение тока на одном периоде ШИМ. В некоторых типах инверторов может применяться шунтовые датчики токов фаз, в которых измерить корректно ток в фазе можно только в тот момент, когда включен нижний ключ инвертора. Измерение тока может производиться 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 нуля при счете вверх-вниз. В этот момент по событию запускается измерение токов АЦП – см. Рис. 4.10.



Рис. . Синхронизация ШИМ и измерения токов АЦП

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, а затем из них вычисляется среднее значение, которое уже используется в системе управления. При большей частоте ШИМ большее количество точек будет применяться для усреднения.

При необходимости в зависимости от особенностей своей аппаратной конфигурации пользователь может реорганизовать работу модуля АЦП. Текущая конфигурация является наиболее совместимой со всеми возможными типами инверторов и датчиков тока.

Отдельно стоит упомянуть о преобразовании изменяемого АЦП сигнала в целочисленный формат переменных системы управления IQ8.24. Такой пересчет осуществляется с учетом текущих базовых единиц, выбранных для измеряемой величины, а также с учетом настроек конкретного канала: коэффициента усиления и смещения. Чтобы упростить вычисления, выполняющиеся на частоте 10кГц вызова основной функции обработки АЦП, в фоновой функции расчёта модуля подготавливаются промежуточные коэффициенты, которые включают в себя все остальные с учетом необходимого преобразования форматов. В основной функции расчета модуля используются эти промежуточные коэффициенты и минимальный набор математических операций для обеспечения высокого быстродействия.

## Модуль ДПР на дискретных элементах Холла

ДПР состоит из трех дискретных элементов (чаще всего это элементы Холла, реагирующие на магнитное поле, но бывают датчики и на другом физическом принципе), которые меняют свое выходное электрическое состояние в зависимости от положения ротора двигателя. дискретных сигналов, поступающих на вход контроллера, программным драйвером формируется дискретное значение углового положения с шагом в 60 электрических 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. . 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 инкрементального энкодера

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. В энкодере может присутствовать, а может отсутствовать индексная (реперная) метка – одна метка на один оборот вала двигателя, которая обычно служит для определения абсолютного положения вала ротора, когда вал провернется до этой метки в процессе работы. Количество импульсов на оборот в инкрементальном энкодере может быть разным и в зависимости от задачи варьируется от 64 до 100000 меток на оборот или даже более широких пределах. Следует отметить, что существует некоторая путаница между количеством периодов датчика и его количеством импульсов. При количестве периодов датчика, например, 1000, суммарное количество импульсов по обоим каналам на обороте составит 4000. В параметр **ДПР-Энкодер.Кол-во периодов [5152,01]** вводится именно число до учетверения.

В модуле обработки механический угол поворота вала ротора преобразуется в электрический (электрический оборот равен механическому, разделенному на число пар полюсов)

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**.

Для работы с синхронным двигателем в векторной системе управления важно знать абсолютное значение углового положения ротора, чтобы обеспечить правильную ориентацию вектора тока статора относительно него. Однако после включения системы управления это положение в общем случае по инкрементальному датчику положения неизвестно, так как инкрементальный энкодер показывает только относительное смещение ротора. Абсолютное положение может быть известно только при достижении индексной метки, но чтобы её достичь, надо начать вращение, а векторную систему нельзя включить, пока нет известного положения ротора. Чтобы выйти из замкнутого круга применяют либо вспомогательный датчик (на элементах Холла), либо запускают систему управления в разомкнутом по положению виде, дожидаются срабатывания индексной метки и потом переходят в векторную систему управления. Существуют и другие методы. Чтобы не усложнять данное ПО реализацией частных случаев работы с энкодером, пользователю предлагается после каждого перезапуска самостоятельно проводить настройку инкрементального энкодера: задавать в параметре начальное смещение угла, чтобы при условно нулевом угле поворота ротора датчик также показывал нулевой электрический угол. Это смещение задается параметром **ДПР-Энкодер.СмещУгол**. Это смещение необходимо настраивать каждый раз заново при включении 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 может быть ручной или автоматической. 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. в зависимости от задачи. Следует напомнить, что такая настройка нужна только для синхронной машины: асинхронный двигатель или ДПТ знания абсолютного положения ротора для работы двигателя не требуют.

В то время как определение углового положения по аппаратному подсчету меток не вызывает вопросов, следует пояснить способ определения частоты вращения (скорости). В модуле реализовано два принципиально разных способа расчета скорости:

1. Аппаратный метод, основанный на возможностях модуля QEP засекать время между заданным количеством импульсов (или между каждым импульсом). При этом количество меток, между которыми засекается время, с ростом скорости вращения автоматически переключается: на низкой скорости вращения засекается время между каждой следующей меткой (для повышения быстродействия измерений), а с ростом скорости делитель количества меток увеличивается (для повышения точности измерений). Этот способ показывает хорошую точность измерения скорости, но имеет проблемы с помехоустойчивостью: если аппаратная часть формирует «ложные» импульсы энкодера (например, от помех работы инвертора), то данный способ захватывает ложное «малое» время между метками и приводит к ложному определению скорости вращения. Кроме того, модуль плохо ведет себя на сверхнизкой скорости вращения и при реверсе, так как засекает время только между двумя соседними импульсами, произошедшими при вращении в одном направлении, и выдает частоту вращения равной нулю при дрожании вала ротора на две метки, что может быть нежелательно для сервоприводов.
2. Программный метод, основанный на вычислении скорости вращения по производной изменения углового положения (которое меньше подвержено влиянию помех). Метод в какой-то мере лишен двух недостатков аппаратного способа измерения скорости, но имеет существенно более низкую точность определения скорости (которой, впрочем, достаточно для большинства практических задач).

По умолчанию активирован программный способ измерения скорости, как более робастный. Но имеется возможность сменить алгоритм обнулив параметр **ДПР-Энкодер.Рассчитывать скорость программно [5152h.03h]**. В активированном программном способе измерения скорости существенной является настройка фильтрации производной углового положения, которая задается в параметре **ДПР-Энкодер.К быстродействия фильтра скорости [5152h.0Bh]**. Чем больше этот параметр, тем слабее значение производной углового положения «зафильтровано», выше быстродействие по времени, но ниже точность измерения скорости.

## Модуль цифрового осциллографа (даталоггер)

Модуль осуществляет цифровое осциллографирование (запись в четыре массива) значений из четырех выбранных пользователем переменных. Модуль работает совместно с режимом «выборка» осциллографа ПО RTCON.

Пользователь может выбрать в словаре объектов до четырех любых переменных, которые он хочет осциллографировать, и добавляет их в осциллограф RTCON. Затем нужно перейти на вкладку осциллографа «выборка». В этот момент RTCON 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 микронтроллера 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 RTCON управляет осциллографом (команда на запуск, опрос готовности данных и т.п.).

Чтобы иметь возможность видеть предысторию, а именно осциллограмму до и во время наступления определенного события, применяется следующая схема. После окончания загрузки осциллограммы из микроконтроллера RTCON 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 дальше ещё записывается еще некоторое количество точек, а затем осциллограф останавливается. Параметром **dlog.Количество точек предыстории [5150h.0Ah]** в процентах задается, сколько точек осциллограммы кольцевого буфера в процентах должно остаться. Так, если задано значение 70%, то пользователь увидит осциллограмму, где 179 точек будут показывать график до наступления события (вызова функции dlog.set\_mode), а 77 точек график после события. Такая схема позволяет увидеть данные в системе управления, например, как до наступления аварийной ситуации, так и после.

Загрузка записанной осциллограммы в RTCON 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 разрядности (в отличие от «непрерывного» режима, где переменные запрашиваются в реальном времени, без усечения формата, но медленно). Если добавить в осциллограф какую-либо 32х разрядную переменную в формате Uint32, то в режиме «выборка» не будет видно изменения его младших 16ти разрядов, что обычно неинформативно. Необходимо для отображения такой переменной копировать её значение внутри программы МК в 16ти разрядную переменную, например, в отладочные переменные Debug3,4 и смотреть отдельно её младшую и, при необходимости, старшую часть отдельно.

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

## 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;

Список идентифицируемых аварий представлен в таблице.

| **№** | **Тип аварии** | **Описание** | **Параметр порога срабатывания в группе «Защиты»** | **Бит маскирования аварии** |
| --- | --- | --- | --- | --- |
|  | **Понижение Udc** | Понижение напряжения на ЗПТ ниже уставки | **Минимальное напряжение** | mask1L.b1  1111 1110 |
|  | **Повышение Udc** | Повышение напряжения на ЗПТ выше уставки | **Максимальное напряжение** | mask1L.b2  1111 1101 |
|  | **Резерв** |  |  | mask1L.b3  1111 1011 |
|  | **Резерв** |  |  | mask1L.b4  1111 0111 |
|  | **Резерв** |  |  | mask1L.b5  1110 1111 |
|  | **Резерв** |  |  | mask1L.b6  1101 1111 |
|  | **Резерв** |  |  | mask1L.b7  1011 1111 |
|  | **Резерв** |  |  | mask1L.b8  0111 1111 |
|  | **Сбой прогр-ы 1кГц** | Программная ошибка 1кГц (превышение времени выполнения программного кода в прерывании 1кГц) | - | mask1H.b1  1111 1110 |
|  | **Сбой прогр-ы 10кГц** | Программная ошибка 10кГц (превышение времени выполнения программного кода в прерывании 10кГц) | - | mask1H.b2  1111 1101 |
|  | **Резерв** |  |  | mask1H.b3  1111 1011 |
|  | **Резерв** |  |  | mask1H.b4  1111 0111 |
|  | **АППАРАТНАЯ** | Аппаратная авария инвертора | - | mask1H.b5  1110 1111 |
|  | **Резерв** |  |  | Mask1H.b6  1101 1111 |
|  | **Резерв** |  |  | mask1H.b7  1011 1111 |
|  | **Резерв** |  |  | mask1H.b7  0111 1111 |
|  | **Резерв** |  |  | mask2L.b2  1111 1101 |
|  | **Резерв** |  |  | mask2L.b3  1111 1011 |
|  | **Резерв** |  |  | mask2L.b4  1111 0111 |
|  | **Резерв** |  |  | mask2L.b5  1110 1111 |
|  | **Макс. ток фазы А** | Превышение тока фазы А; | **Максимальный ток** | mask2L.b6  1101 1111 |
|  | **Макс. ток фазы В** | Превышение тока фазы B | **Максимальный ток** | mask2L.b7  1011 1111 |
|  | **Макс. ток фазы С** | Превышение тока фазы C | **Максимальный ток** | mask2L.b8  0111 1111 |
|  | **Резерв** |  |  | mask2H.b1  1111 1110 |
|  | **Резерв** |  |  | mask2H.b2  1111 1101 |
|  | **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 инвертор отключается (время реакции не более 100мкс). Каждая авария может быть замаскирована. При её маскировании ПО контроллера не обрабатывает событие возникновения аварии и преобразователь продолжит работу (если это возможно…). Для маскирования 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), также в таблицы приведены примеры масок.

## Отображение текущих параметров привода (модуль CurPar)

Модуль CurPar, реализованный в файлах V\_CurPar.c и V\_CurPar.h осуществляет сбор и расчет текущих параметров привода. Значения доступны в группе **Наблюдаемые**. В данном пункте 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 (fвых на Рис. 4.1).
* **Акт. мощность** **–** текущая потребляемая двигателем активная мощность. Вычисляется исходя из вектора напряжения, заданного на модуль ШИМ и показаний датчиков токов фаз.
* **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.

## Различные важные параметры, не относящиеся к конкретному модулю

### Настройки системы

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

**Настройки СУ.РежРаб [5114h.02h] –** выбор структуры управления двигателем, подробнее описано в п.4.1.

**Настройки СУ.АЦП авто смещ. [5114h.04h] –** разрешение работы модуля автоматической подстройки смещения каналов датчиков тока АЦП.

**Настройки СУ.Перезагрузка [5114h.05h] –** установка единицы в этот параметр выполняет перезагрузку ПО микроконтроллера.

### Переменные для отладки

В словарь объектов выведены несколько служебных отладочных переменных, в которые можно внутри ПО микроконтроллера присваивать любые данные и наблюдать их затем в ПО RTCON. 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 RTCON «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.

### Счетчики

В группе параметров **Счетчики** содержатся параметры со счетчиков прерываний. По их инкрементированию 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 тактах прерывания 10кГц и 1кГц соответственно. В ПО микроконтроллера это время высчитывается таймером общего назначения. По показаниям данных счетчиков удобно оценивать процент загрузки ядра микроконтроллера.

# Быстрый старт

## Подключение к ПК через RTCON

Программа ПК RTCON, работающая под ОС Windows, используется как пульт управления приводом, средство настройки параметров системы управления и просмотра осциллограмм. Есть два способа подключения RTCON:

1. Штатно через CAN интерфейс. Для этого используемая плата должна содержать CAN выход, а у пользователя иметься один из CAN адаптеров, поддерживаемых программой RTCON (см. руководство пользователя в директории с программой). При этом драйвер CANopen будет работать полнофункционально, загрузка осциллограмм проходить быстро, в сеть CAN можно подключить несколько устройств одновременно.
2. Через UART интерфейс (RS232, USB-UART переходник и т.п.), представляемый в компьютере как COM порт (настоящий или виртуальный). В этом случае драйвер CANopen и RTCON работают с ограничениями: используется только SDO сервис CANopen обмена сообщениями, что негативно влияет на время загрузки осциллограмм. Кроме того, в сети может быть только одно устройство. Также в программе RTCON нужно указывать номер виртуального COM порта в параметрах модуля связи, который может произвольно меняться после перезагрузки компьютера или смене USB порта.
3. Через USB интерфейс, представляемый в компьютере как виртуальный COM порт. В этом случае драйвер CANopen и RTCON имеют те же возможности и ограничения, что и при подключении через UART-интерфейс, однако скорость работы через USB выше (примерно на 40%).

Для подключения через интерфейс CAN необходимо:

1. Подключить используемый USB-CAN переходник к плате согласно документации на переходник и плату через трехпроводной кабель (CAN-H, CAN-L, GND). При небольшой длине кабеля в раде случаев допустимо использовать двухпроводную связь. В общем случае на линии также должен быть установлен минимум один терминальный резистор (резистор параллельно CAN-H, CAN-L номиналом порядка 120 Ом).
2. Установить драйвера для ПК на используемый USB-CAN переходник (с официального сайта производителя, для МАРАФОН это <http://can.marathon.ru/page/prog/chai> ).
3. Запустить программу RTCON, открыть в меню Сервис – Параметры модуля связи, выбрать из списка сверху используемый модуль связи, нажать Ок.

Для подключения через интерфейс UART необходимо:

1. Подключить ПК к интерфейсу UART используемой платы любым доступным способом: через интегрированный на плату USB-UART переходник, через внешний переходник, через COM порт компьютера и преобразователь уровней и т.п. Достаточно трехпроводного интерфейса: RX, TX, GND.
2. Установить драйвера для ПК на используемый USB-UART переходник. Убедиться, что в диспетчере устройств Windows появился соответствующий виртуальный или настоящий COM порт. Запомнить номер порта.
3. Запустить программу RTCON, открыть в меню Сервис – Параметры модуля связи, выбрать из списка сверху «USB-RS». Задать слева внизу параметр «Номер COM порта для альтернативных модулей» равным номеру используемого COM порта, нажать Ок.

Для подключения через интерфейс USB необходимо:

1. Подключить ПК к интерфейсу USB соответствующим кабелем. Устройство автоматически определяется как COM-порт.
2. Запустить программу RTCON, открыть в меню Сервис – Параметры модуля связи, выбрать из списка сверху «USB-RS». Задать слева внизу параметр «Номер COM порта для альтернативных модулей» равным номеру используемого COM порта, нажать Ок.

Далее, если программа в микроконтроллере работает, подключение выполнено правильно, модуль связи настроен согласно инструкции, то RTCON должен отобразить текущие устройства в сети, т.е. показать в верхней левой части окна вкладку «Узел N» с названием «MCB4\_2810», станет доступна кнопка «Подключиться». После нажатия на неё можно приступать к работе. На Рис. 5.1 показано окно программы RTCON после успешного подключения.

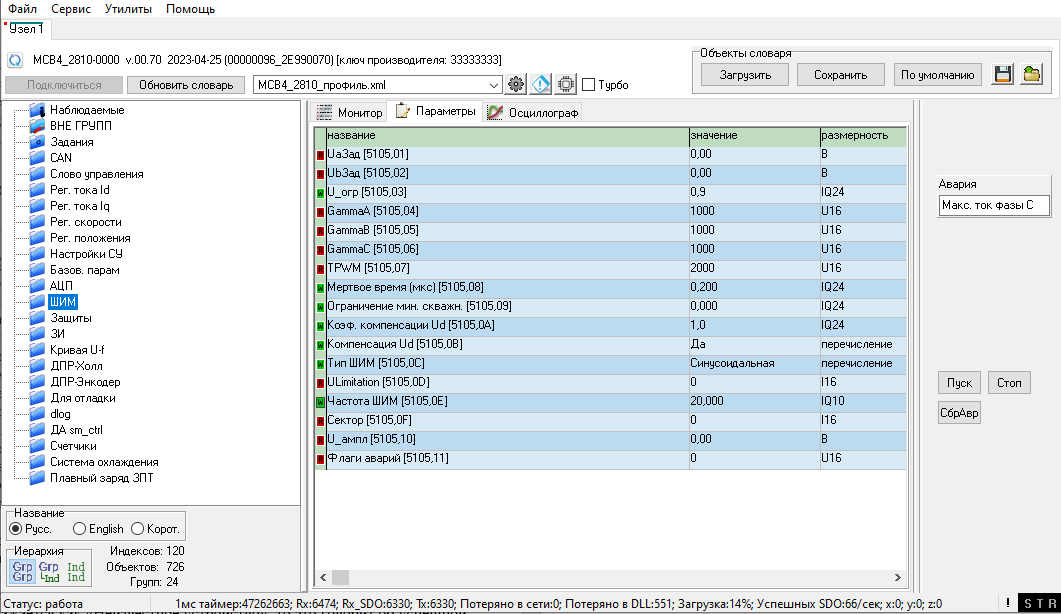


Рис. . Окно программы RTCON после подключения к устройству

Если подключение не удается, то следующие рекомендации могут помочь в диагностике проблемы:

1. Если программа RTCON не запускается, то можно попробовать временно отключить антивирусные программы (по опыту разработчиков один раз был замечен конфликт с антивирусной программой Avast). Также при первом запуске брандмауэр может спросить о разрешении доступа к сети TCP IP, доступ можно запретить, так как он нужен для предоставления внешнего доступа к сети CAN на чтение другим программам (опцию TCP сервера можно в будущем отключить в настройках).
2. Следует обратить внимание на статус программы RTCON слева снизу. Статус «Работа» информирует об успешном открывании модуля связи и его работе. Если же идет последовательный перебор статусов «ожидание»/«запуск», то необходимо проверить корректность установки драйверов на модуль связи, проверить, что он правильно выбран в параметрах модуля связи в меню RTCON, что модуль связи не задействован в данный момент другим приложением (или другой запущенной копией RTCON).
3. Если устройство не отображается на вкладке устройств, то необходимо проверить работу программы в микроконтроллере, корректность настройки выводов используемого интерфейса связи в микроконтроллере (если используется не поддерживаемая по умолчанию аппаратная конфигурация), корректность подключения кабелей. Микроконтроллер должен каждую секунду отправлять статусные сообщения сервиса Heartbeat CANopen, можно воспользоваться внешним осциллографом и проверить наличие посылок в линии связи.
4. Если устройство отображается как «Неизвестное устройство», то это говорит об успешной передаче посылок от микроконтроллера в RTCON, но об отсутствии связи в обратном направлении.
5. Если вместо одного узла в сети с названием «ARM\_MCP» появилось два с названием «Эмулятор», это значит в настройках RTCON вместо переходника связи CAN выбран эмулятор. Откройте в программе RTCON Сервис-Параметры модуля связи. Выберите нужный вам модуль связи.
6. Если ничего из перечисленного не помогает, обратитесь в поддержку.

После успешного подключения к устройству необходимо ознакомиться с интерфейсом программы RTCON. Подробное описание доступно в справке, которую можно открыть как из меню Справка, так и найти файл «**RTCON руководство пользователя.pdf**» в директории с программой. В данном руководстве приведено краткое описание приёмов работы с программой по мере необходимости.

В левой части окна программы Рис. 5.1 расположен список объектов CANopen (параметры устройства), в центре – значения параметров или осциллограф, а в правой – панель управления устройства. Для простоты последующего указания параметра будет применяться такое именование: **Группа.Параметр**. Например, **АЦП.Ток фазы A**. В данном случае параметр – это некоторый объект словаря CANopen. Открывая группы с параметрами можно видеть список параметров в них в средней части окна. Параметры с зелеными пиктограммами можно менять (параметры для записи), с красными – только для чтения. После изменения значения параметра нужно нажимать клавишу Enter.

## Настройка базовых значений

Cистема управления в ПО МК построена в относительных единицах, а значит перед началом работы важно правильно выбрать базовые значения, относительно которых будут нормироваться физические значения для получения относительных единиц. Настройка таких параметров производится в группе Базов. парам. Желательно (но не обязательно), чтобы базовые величины были близки к номинальным. Тем не менее, допускается отклонение даже в несколько раз. Таким образом, если номинальный ток, с которым ожидается работать, не на порядок отличается от текущего значения базового параметра, базовый параметр можно не изменять. Если же отличие существенно и есть риск переполнения значения формата IQ24 или потеря точности, рекомендуется изменить базовые параметры. Рекомендуется давать базовым коэффициентам круглые, легко запоминаемые значения, чтобы в случае необходимости пересчет из относительных единиц и обратно можно было производить в уме. Таким образом, в случае необходимости нужно скорректировать следующие параметры:

1. **Базов. парам.Номинальный ток [5113,02]**, который является базовым значением для фазных токов двигателя.
2. **Базов. парам.Номинальное напряжение ЗПТ [5113,05]**, который является базовым значением для напряжения ЗПТ (Udc).
3. **Базов. парам.Номинальная скорость [5113,04]**, который аналогично определяет масштаб для частоты вращения в об/мин.
4. **Базов. парам.Число пар полюсов [5113,01]**, который должен быт равен паспортному значению числа пар полюсов двигателя и определяет автоматически рассчитываемый базовый параметр частоты тока **Базов. парам.Номинальная частота [5113,08]**.

После изменения базовых параметров требуется сохранение параметров контроллера кнопкой «Сохранить параметры» в RTCON, перезагрузка контроллера (перезапуск программы микроконтроллера), а затем операция обновления словаря RTCON (кнопкой обновить словарь слева сверху) для считывания из микроконтроллера значений новых масштабирующих коэффициентов (базовых параметров).

Для реального оборудования базовые параметры могут быть заполнены исходя из паспортных параметров оборудования.

## 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 амплитудному току в фазе, минимальному напряжению на ЗПТ, максимальному напряжению на ЗПТ и максимальной скорости вращения. Уставки защит при пусконаладочных работах должны быть как можно ближе к реальным ожидаемым значениям – лучше лишний раз убедиться в срабатывании программной защиты и отодвинуть её уровень, чем допустить выхода оборудования из строя. Рекомендуется настроить уровень защиты по напряжению звена постоянного тока в диапазоне 10% от ожидаемого значения (параметры **Защиты.Минимальное напряжение [510E,09]** и **Защиты.Максимальное напряжение [510E,08]**). Защита по максимальному току фазы (**Защиты.Максимальный ток [510E,07]**) должна быть настроена также на уровне 10% выше ожидаемого тока фазы.

Ожидаемые значения – те, которые будут на двигателе при его первом пробном запуске. Когда речь идет о реальном оборудовании, то необходимо руководствоваться его номинальными данными. При первом запуске рекомендуется начать со значений токов привода не выше 10% от номинала и, по возможности, с пониженного напряжения питания.

При выходе параметра за диапазон уставок защит инвертор отключится, а на панели RTCON справа будет индицироваться текст сработавшей защиты на панели «Авария».

## Настройка модуля АЦП

Если модуль обработки данных, который принимает и обрабатывает данные с АЦП микроконтроллера (далее модуль АЦП), настроен неверно, то большинство защит инвертора, не говоря уже об отображении величин, работать не будут. Для трехфазного шестиключевого инвертора обычно имеется три главных измеряемых аналоговых величины: два тока фаз и напряжение звена постоянного тока (ЗПТ, оно же Udc).

В данном модуле АЦП каждая измеряемая величина представлена, как минимум, тремя объектами (смотрите группу параметров АЦП в RTCON):

1. 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 – 3В, на него подается сигнал с датчика напряжения на звене постоянного тока и «0» на входе контроллера соответствует 0 В на ЗПТ, 3 вольта соответствует 100В на ЗПТ, то коэффициент равен 100.
2. Смещение. Если измеряемая величина равна нулю, а реально измеряется не нулевое значение (аддитивная ошибка), можно установить смещение нулевого уровня сигнала. Это смещение задается в тех единицах измерения, в которых возвращает значение сам модуль АЦП, т.е. смещение может быть в диапазоне -4095…+4095.
3. 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.

Правильно настройку коэффициентов канала модуля АЦП производить по двум точкам, чтобы корректно настроить коэффициент смещения и усиления. Первой точкой удобнее выбрать ноль измеряемого сигнала и подобрать такое смещение, чтобы измеряемое значение также было нулевым. Затем подать некоторый постоянный уровень и подобрать уже коэффициент усиления.

Настройку коэффициентов измеряемых каналов АЦП удобно начать с напряжения на звене постоянного тока (ЗПТ). Считая, что смещение уже настроено верно и равно нулю, рассчитайте или подберите такой коэффициент усиления, чтобы отображаемое измеряемое значение соответствовало реальному напряжению на ЗПТ. Для этого необходимо в программе RTCON изменять значение параметра **АЦП.Коэффициент напряжения ЗПТ [5116,0D]** и наблюдать значение **АЦП.Напряжение ЗПТ [5116,0F]**.

Калибровку приёма сигналов датчиков тока можно произвести разными способами: имитационным сигналом на измерительный вход АЦП или при работе реального двигателя. Проще всего ввести преобразователь в работу и по амперметру в фазах двигателя настроить коэффициенты усиления **АЦП.Коэффициент тока фазы А [5116,01]** и **АЦП.Коэффициент тока фазы В [5116,04]**. Ток фазы C для трехфазного двигателя настраивать не требуется, так как ток рассчитывается как сумма токов в двух других фазах со знаком минус (по закону Кирхгофа). 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 пункт 1).

Настройка смещения может выполняться автоматически состоянии останова. Для этого нужно включить модуль автосмещения токов 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. Если автоматический расчет смещения отключен, то на выключенном оборудовании необходимо настроить коэффициенты смещения датчиков тока, чтобы токи фаз по показаниям АЦП были около нуля (параметры **АЦП.Смещение тока фазы А [5116,02]** и **АЦП.Смещение тока фазы В [5116,05]**).

Если запуск реального оборудования с ненастроенными датчиками может быть опасен (привод на мощность выше 1кВт и напряжение ЗПТ выше 50В), можно подать имитацию сигнала датчиков тока, рассчитать, какому току это соответствует и подобрать коэффициенты усиления, чтобы измеренное АЦП значение этому соответствовало.

Если же идет работа с низковольтным отладочным комплектом, то можно запустить привод и настроить датчики токов по реальным фазным токам, измеряемым внешним прибором: амперметром, токовыми клещами. Для этого нужно произвести пуск двигателя (см. последовательность пуска в п.5.5) и наблюдать в программе RTCON ток в фазах двигателя. Если значения в программе RTCON и на амперметре отличаются, подстроить коэффициенты. Если используется амперметр на постоянный ток, то можно задать нулевую частоту вращения и настроить датчики тока на постоянном токе. 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».

## Запуск привода в скалярной системе управления

У системы управления существует несколько различных структур управления в зависимости от типа электродвигателя и желаемых характеристик. Например, классическая структура с пропорциональным изменением напряжения и частоты (U/f) для асинхронного и синхронного двигателя, частотно-токовая система управления, векторная система управления и т.п. Желаемая структура управления задается её номером 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.

Режим работы U/f обеспечивает традиционное скалярное управление АД/СД с регулированием напряжения по заранее заданной кривой *U/f*. Структура скалярной СУ представлена на Рис. 5.2.



Рис. . Структура системы скалярного управления

Кривая *U/f* задана двумя опорными точками через следующие параметры группы **КриваяU-f**: **Нулевая опорная частота**, **Нулевое опорное напряжение**, **Первая опорная частота**, **Первое опорное напряжение**. Нулевую точку можно для первого пуска задать нулями, а первую задать значениями, равными номинальным для текущего двигателя. Напряжение задается как фазное, амплитудное. Разгон производится через модуль задатчика интенсивности (темп разгона задается в **ЗИ.T разгона до номинала [5000,01]**).

Перед первым пуском еще раз проверьте защиту максимального тока фаз в модуле защит (**Защиты.Максимальный ток [510E,07]**) установите её 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.

Для первого запуска можно задать низкое значение частоты вращения: 5-10 об/мин в параметре **Задания.Скорость [5112,01]** или, что более удобно, на панели управления справа.

Также при работе с реальным приводом необходимо сконфигурировать модуль ШИМ: выбрать частоту ШИМ (**ШИМ.Частота ШИМ [5105,0E]**), установить величину мертвого времени (**ШИМ.Мертвое время (мкс) [5105,08]**) согласно документации на аппаратное обеспечение.

Убедившись, что АЦП работоспособно (значение напряжения ЗПТ в модуле АЦП адекватно, значения токов фаз колеблются вблизи нуля), задана верная структура управления и нет проблем в аппаратной части, можно дать команду пуска: для этого нажмите на кнопку Пуск на панели управления в правой части окна RTCON. Привод должен перейти в работу и на панели оперативного управления справа сверху статус должен смениться с «ГОТОВ» на «РАБОТА».

Для наблюдения токов фаз удобно смотреть их изменение во времени, т.е. осциллограмму или «график». Чтобы добавить интересующий параметр в цифровой осциллограф программы RTCON, нужно щелкнуть по параметру правой кнопкой, выбрать пункт «Добавить в осциллограф» и любой из четырех каналов.

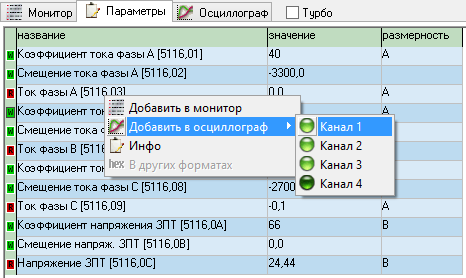


Рис. . Добавление параметра в осциллограф

Примерные осциллограммы токов фаз (**АЦП.Ток фазы A, B, C**) и заданного фазного напряжения (**ШИМ.UaЗад**) показаны на Рис. 5.4. Для более частого обновления параметров можно воспользоваться кнопкой «турбо» рядом с вкладной осциллографа, а можно перейти на вкладку «выборка» внутри окна осциллографа (подробнее о работе с осциллографом читайте в руководстве к программе RTCON, которое открывается через меню Помощь).

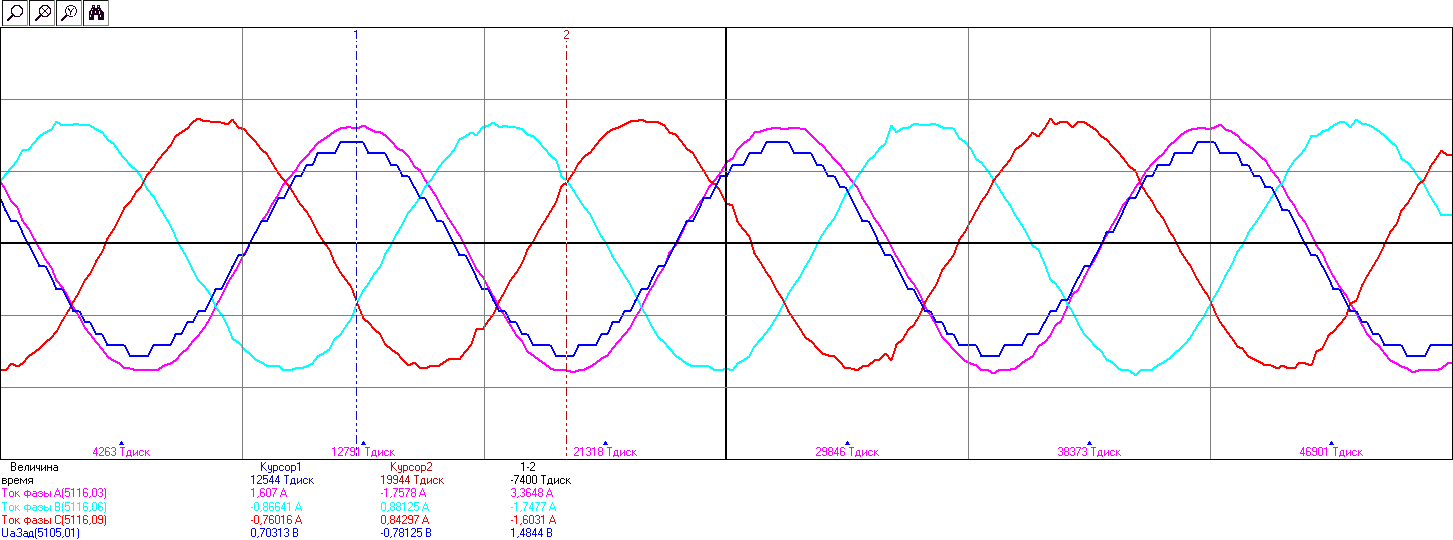


Рис. . Осциллограмма тестового запуска

Также можно визуально убедиться, что двигатель крутится.

Если при запуске привода сработала защита, что можно видеть по статусу системы управления «АВАРИЯ», то необходимо определить какая: это написано ниже на панели «Авария». Если сработала защита по превышению фазного тока, то можно попробовать уменьшить заданное напряжение в параметре **Кривая U-f.Первое опорное напряжение [511A,06]** и повторить запуск. Если сработала защита по понижению напряжения ЗПТ, то это говорит о возможно недостаточном по мощности источнике питания (хотя и возможно КЗ). Против этого также может помочь снижение заданного напряжения в кривой U/f, и, как следствие тока и мощности. Если сработала защита повышения напряжения ЗПТ, это странно для запуска на данной частоте вращения и может говорить о помехах в измеряемом сигнале либо недостаточной ёмкости звена постоянного тока инвертора. Если сработала аппаратная авария, то это говорит либо о коротком замыкании на выходе инвертора, либо о неправильной работе драйверов ключей инвертора. Срабатывание других защит на данном этапе маловероятно.

Если система управления в состоянии «РАБОТА», но токи фаз по-прежнему равны нулю, а двигатель не крутится, это может говорить о недостаточном приложенном напряжении. Попробуйте постепенно поднять напряжение нулевой точки, **Кривая U-f.Нулевое опорное напряжение [511A,04]**. При значении не выше 10% от максимального напряжения токи точно должны начать протекать, а двигатель должен начать крутиться. Если этого не произошло, нужно разбираться в аппаратной части или попробовать пройти шаги сначала, сбросив настройки привода на заводские кнопкой «Загрузить по умолчанию» справа сверху.

Если двигатель вращается, а осциллограммы токов похожи на приведенные, то этап первичной пусконаладки завершен. Далее можно подать команду останова (кнопка Стоп на панели управления в правой части окна RTCON), а можно поднять задание частоты вращения и попытаться раскрутить двигатель быстрее.

### Настройка регуляторов тока

Регуляторы тока необходимы для замкнутых по току структур управления, включая векторное управление. Для их настройки необходимо наблюдать на осциллограмме реакцию на скачок задания тока при остановленном двигателе. Для продолжения работы по этому пункту необходимо уже иметь корректно настроенные датчики токов и остальное, т.е. успешно запустить скалярное управление согласно главе 4.

Для этого необходимо включить замкнутую по току структуру управления: режим удержания – в **Настройки СУ.Режим работы [5114,02]** ввести значение «2». Подробнее структура описана в п.п. 4.1.3. В осциллограф RTCON добавим мгновенные токи фаз из группы **АЦП** (через контекстное меню параметра). Также может быть интересно наблюдать заданное напряжение (**ШИМ.UaЗад [5105,01]**).

Для правильного отображения процесса пуска в параметр **dlog.StartBitL** вводим значение «00000010», в **dlog.StartBitH** – «00000000», в параметре **dlog.Количество точек предыстории [5150,0A]** ставим 0%, чтобы начало осциллограммы соответствовало времени скачка задания. 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 RTCON, а также в главе 4.9.

Настраивать регуляторы токов для реальной эксплуатации привода желательно с уровнем тока, близким к номинальному. Однако для начала можно задать и меньшей уровень тока. Задание тока настраивается в параметре **Задания.Ток удержания [5112,03].**

Нажимаем кнопку «Пуск» на панели управления в правой части окна RTCON, заходим во вкладку «Осциллограф» и нажимаем кнопку «Обновить» (Рис. 5.5).

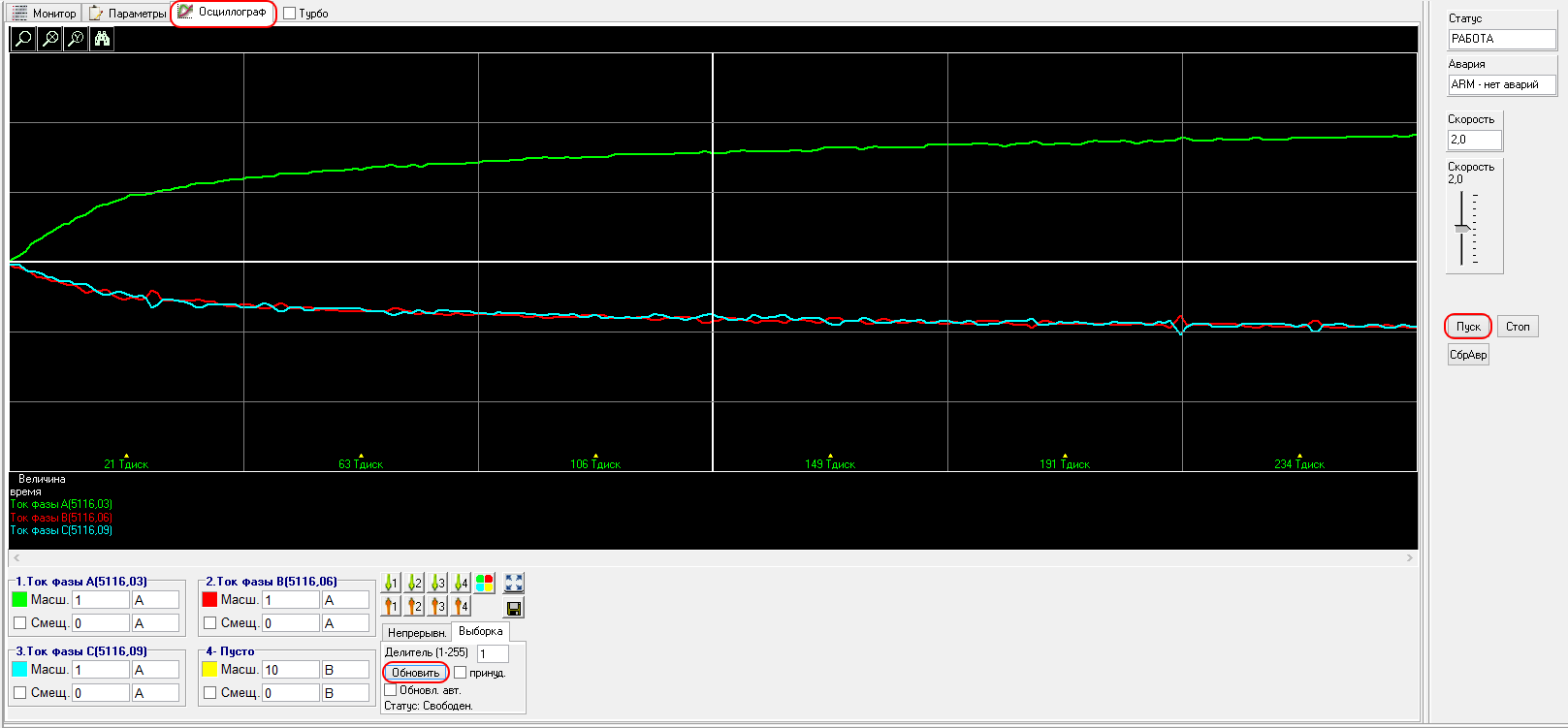


Рис. . Окно осциллографа

Если при нажатии кнопки «Обновить» появляется окно Рис. 5.6, значит по одному из активных триггеров произошел захват осциллограммы и её можно загрузить. Однако если это первый запуск осциллографа после изменения параметров осциллографирования (набора каналов, делителя), то необходимо отказаться от загрузки осциллограммы, так как там «мусор».

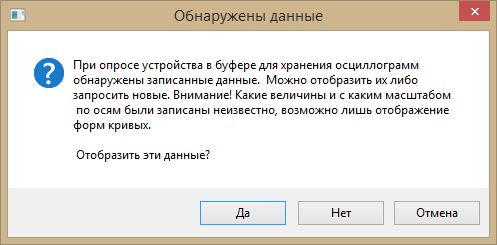


Рис. . Диалоговое окно загрузки захваченных осциллограмм

Для отображения осциллограммы запуска в появившемся окне нужно щелкнуть по кнопке «Да». Переходной процесс при ненастроенных регуляторах может выглядеть подобно Рис. 5.7.

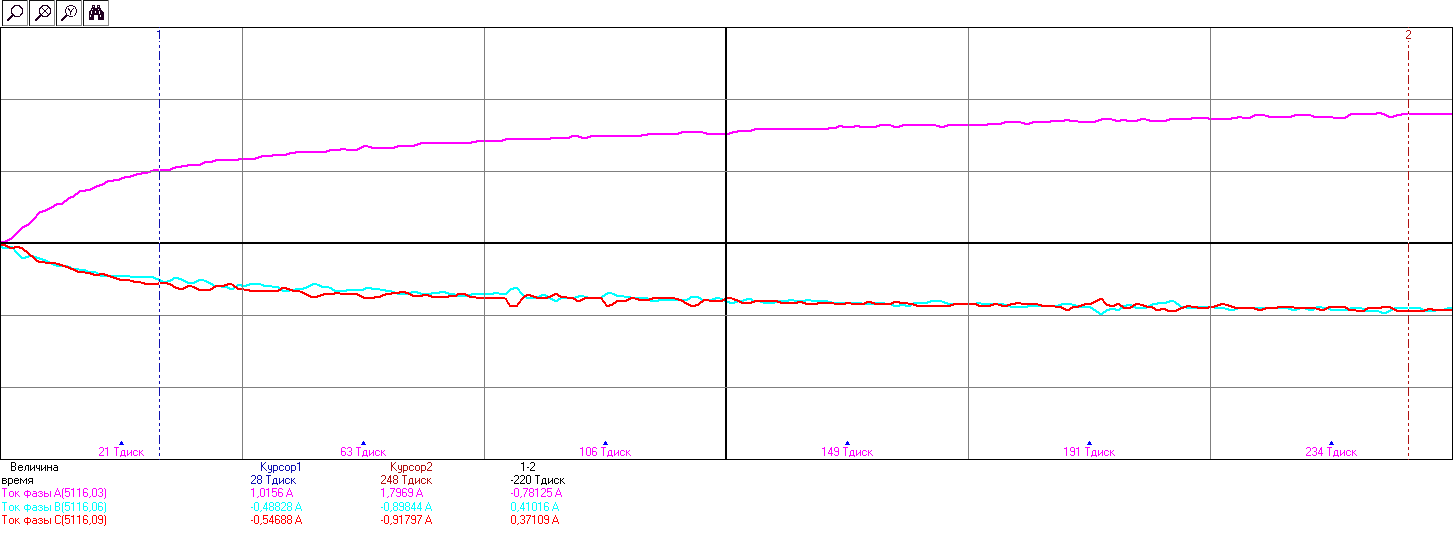


Рис. . Переходной процесс пуска при ненастроенных регуляторах

Чтобы видеть процесс с максимальным разрешением по времени, рекомендуется установить делитель на вкладке «выборка» равным 1. Для его применения нужно нажать кнопку «Обновить».

Если сработала какая-либо защита, то необходимо исследовать проблему. В этом должна также помочь осциллограмма переходного процесса. Наиболее вероятно срабатывание защиты по превышению фазного тока. Если это так, необходимо убедиться, что задание тока меньше уставки защиты, а также попробовать дополнительно уменьшить значение пропорционального коэффициента регуляторов. При запуске системы управления на новом аппаратном обеспечении зачастую могут быть перепутаны знаки или порядок датчиков токов фаз, а также выходы управления ШИМ инвертора. В этом случае переходной процесс на осциллограмме будет выглядеть экспоненциально расходящимся: необходимо вернуться к скалярному режиму работы с сверить показания измерения токов фаз с реальными данными внешнего прибора, а также с заданным напряжением на модуле ШИМ (на низкой частоте вращения напряжению **ШИМ.UaЗад [5105,01]** должен соответствовать **АЦП.Ток фазы А [5116,03]**).

Процесс подбора коэффициентов регуляторов тока рассмотрен на рисунках ниже. Сначала интегральный коэффициент обнуляется (параметры **Рег. тока Id.Интегральный коэфф. Ki [5100,06]** и **Рег. тока Iq.Интегральный коэфф. Ki [5102,06]**), а пропорциональный (**параметры Рег. тока Id.Пропорц. коэфф. Kp [5100,03]** и **Рег. тока Iq.Пропорц. коэфф. Kp [5102,03]**) постепенно повышается до появления колебательности переходного процесса. С каждым изменением коэффициенты нужно подавать команду останов и пускать привод снова, чтобы загрузить осциллограмму и оценить вид переходного процесса. Коэффициенты для обоих регуляторов тока (Iq, Id) должны быть равны и изменяются одновременно. На Рис. 5.8 показан переходной процесс регулятора тока на скачок задания с перерегулированием. Как видно, в контуре тока есть статическая ошибка (задание равно 2А, а фактическое значение меньше – около 1А). Устранить её поможет интегральная часть регулятора. После получения похожего переходного процесса с несколькими колебаниями пропорциональный коэффициент нужно уменьшить и перейти к повышению интегрального коэффициента.

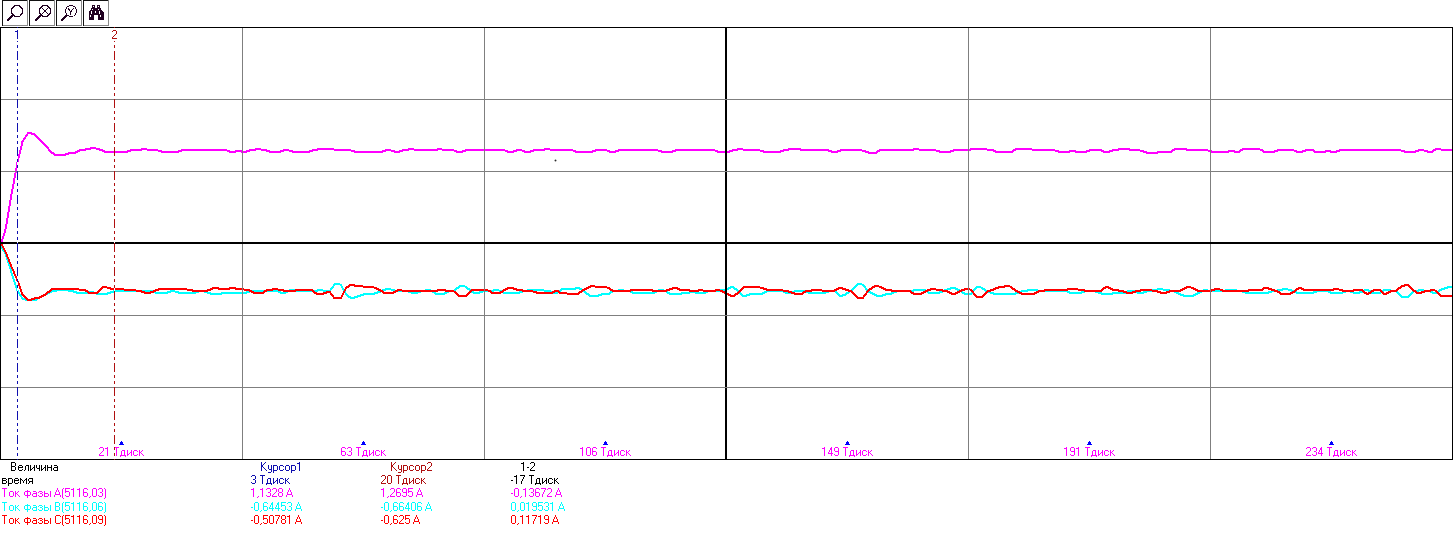


Рис. . Переходной процесс регулятора тока на скачок задания с перерегулированием

При настройке желательно следить, чтобы заданное на инвертор напряжение не достигало максимального ограничения, что можно видеть по выставлению флага **ШИМ.ULimitation [5105,0D]**, который удобно также добавить на осциллограмму. Если во время переходного процесса флаг выставляется, необходимо снизить задание тока.

По окончанию грубой настройки необходимо произвести финальный подбор обоих коэффициентов регуляторов, чтобы получить переходной процесс, похожий на настройку на технический оптимум. Желаемый переходной процесс показан на Рис. 5.9.

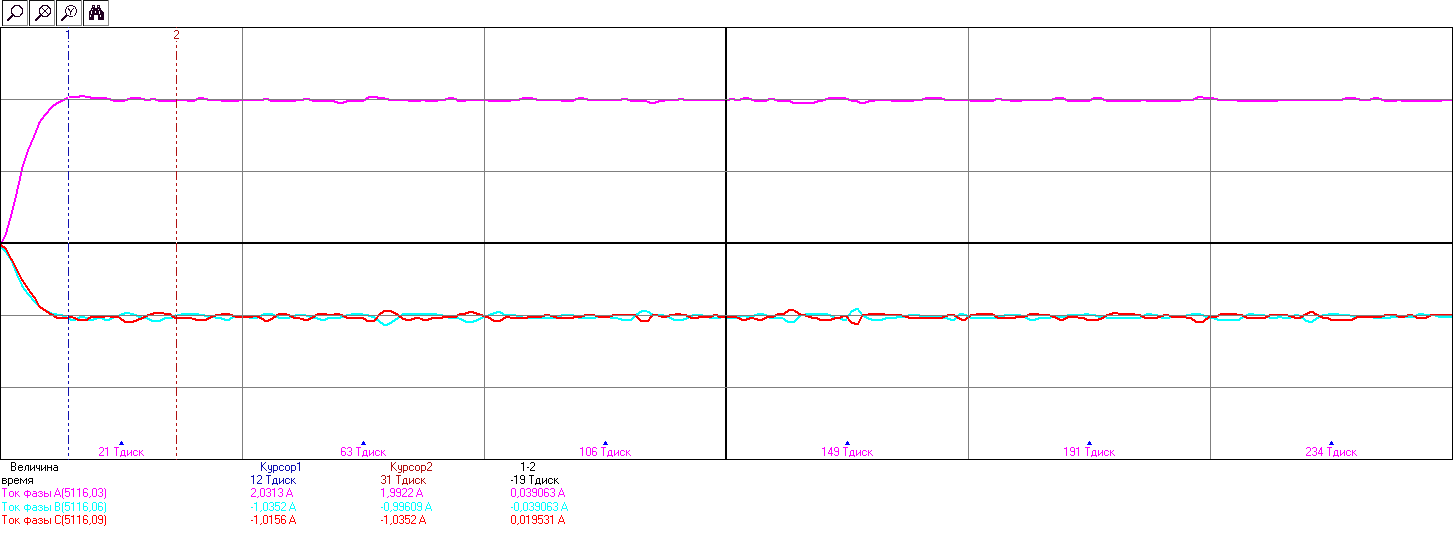


Рис. .. Настройка регуляторов тока на технический оптимум

При получении аналогичной осциллограммы регуляторы тока можно считать настроенными. Можно сохранить текущие настройки в энергонезависимую память, нажав кнопку «Сохранить параметры» в RTCON. Также в данном режиме можно дополнительно откалибровать показания датчиков тока, так как постоянный ток легче измерить и сравнить с заданием, чем переменный.

## Системы управления синхронным двигателем

### Частотно-токовое управление

Частотно-токовое управление является наиболее простым способом привести во вращение синхронную машину. Подробнее структура данного способа управления рассмотрена в п. 4.1.2. Для запуска структуры управления необходимо провести успешный запуск скалярной системы управления и настроить регуляторы токов (п.п. 5.5.1). Только после этого можно следовать далее.

Для выбора частотно-токовой структуры управления необходимо установить **Настройки СУ.Режим работы [5114,02]** = «4». Установить желаемое задание амплитуды тока **Задания.Макс. ток ВУ [5112,02]**, желаемую частоту вращения **Задания.Скорость [5112,01]**. Начать рекомендуется с небольшого тока, порядка 10% номинала и небольшой частоты вращения, порядка 1-10 об/мин. В осциллограф рекомендуется добавить три тока фаз и напряжение ЗПТ из группы АЦП.

Подать команду Пуск на панели в RTCON. Двигатель должен вращаться со скоростью (частотой вращения), заданной в **Задания.Скорость [5112,01]**, форма токов – синусоидальная (Рис. 5.10).

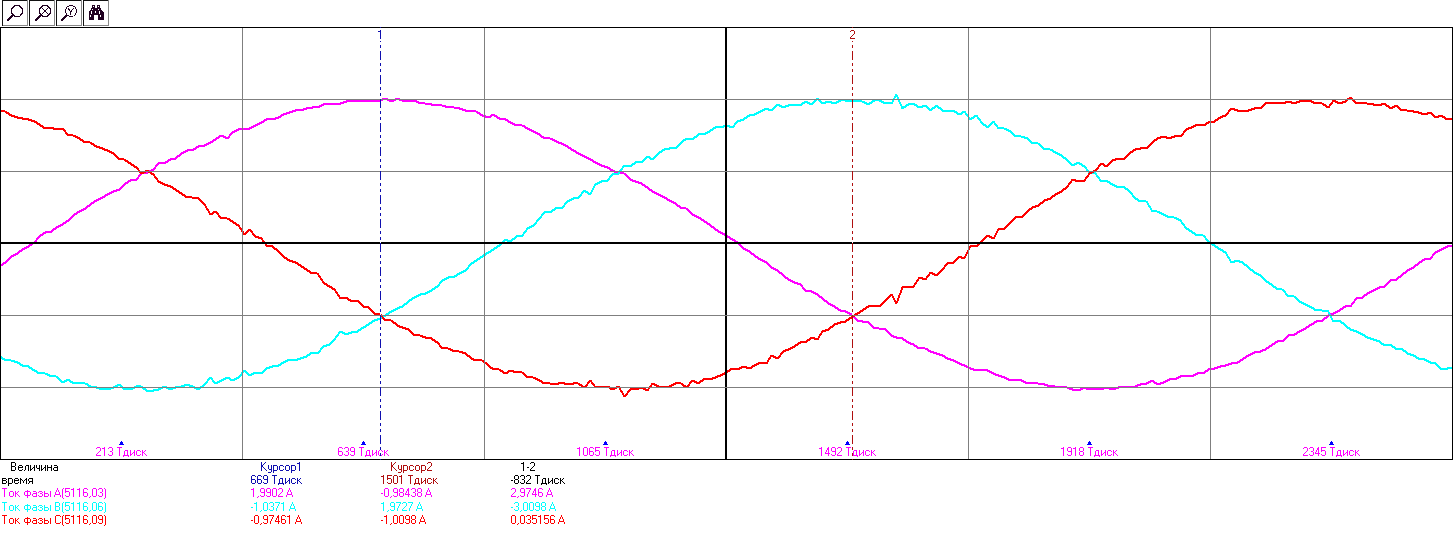


Рис. .. Форма токов фаз при работе частотно-токовой структуры управления

При низкой частоте вращения осциллограмму токов можно наблюдать как в непрерывном режиме осциллографа (рекомендуется активировать опцию «Турбо» рядом с вкладкой осциллографа для более быстрого обновления графиков), так и в режиме «выборка» с большим делителем. С повышением частоты вращения для наблюдения формы токов фаз необходимо пользоваться только режимом «выборка», так как непрерывный режим реального времени не будет успевать обновлять данные с нужной дискретностью по времени.

Вал двигатель должен вращаться. Если двигатель не крутится или крутится нестабильно, хотя токи синусоидальны и равны заданию, можно попробовать увеличить задание амплитуды тока, а также снизить задание частоты вращения.

Если двигатель крутится стабильно, то для повышения частоты вращения рекомендуется предварительно настроить темп разгона (задатчик интенсивности) в параметре ЗИ.T разгона до номинала [5000,01]. Подробнее о ЗИ можно прочитать в п.п. 4.4.. Чем медленнее разгон, тем меньше вероятность выпадения двигателя из синхронизма во время разгона. Далее можно повысить задание частоты вращения в переменной **Задания.Скорость [5112,01]**. При этом есть риск «вываливания двигателя из синхронизма» – ротор может не успеть следовать за полем статора, магнитная пружина «разорвется», после чего ротор двигателя остановится и будет качаться в одном положении. Если это произошло, нужно увеличить задание на ток статора, снизить задание частоты вращения или замедлить темп разгона.

С ростом частоты вращения необходимо следить, что напряжение на инверторе имеет запас: флаг ограничения выходного напряжения ШИМ.ULimitation [5105,0D] не выставляется в единицу.

Если при разгоне привода срабатывает какая-либо защита, необходимо изучить её причины. Наиболее вероятно, что могут сработать защиты по превышению тока статора или повышения напряжения ЗПТ. Проще всего исследовать проблему путем наблюдения осциллограммы события аварии. Для этого необходимо добавить интересующие переменные в осциллограф (например, токи фаз и напряжение ЗПТ), поставить параметр **dlog.Количество точек предыстории [5150,0A]** = 70%, чтобы видеть предысторию развития аварии, сконфигурировать флаг срабатывания триггера на событие останова привода (**dlog.StartBitL [5150,07]= 00000000**, **dlog.StartBitH [5150,08]= 00000001**), открыть режим «выборка», настроить желаемый делитель и нажать кнопку «обновить» с отказом от загрузки уже записанной осциллограммы (если такой запрос будет). Теперь осциллограф готов к захлопыванию осциллограммы по аварии: необходимо воспроизвести аварийное отключение, после чего нажать кнопку «обновить» и согласиться с загрузкой записанной по событию осциллограммы. Например, на Рис. 5.11 показана форма токов фаз при срабатывании максимально-токовой защиты при разгоне привода в частотно-токовом управлении. Причина такой аварии – раскачка машины с данным типом управления при работе на высоких частотах вращения (малая демпфируемость колебаний).

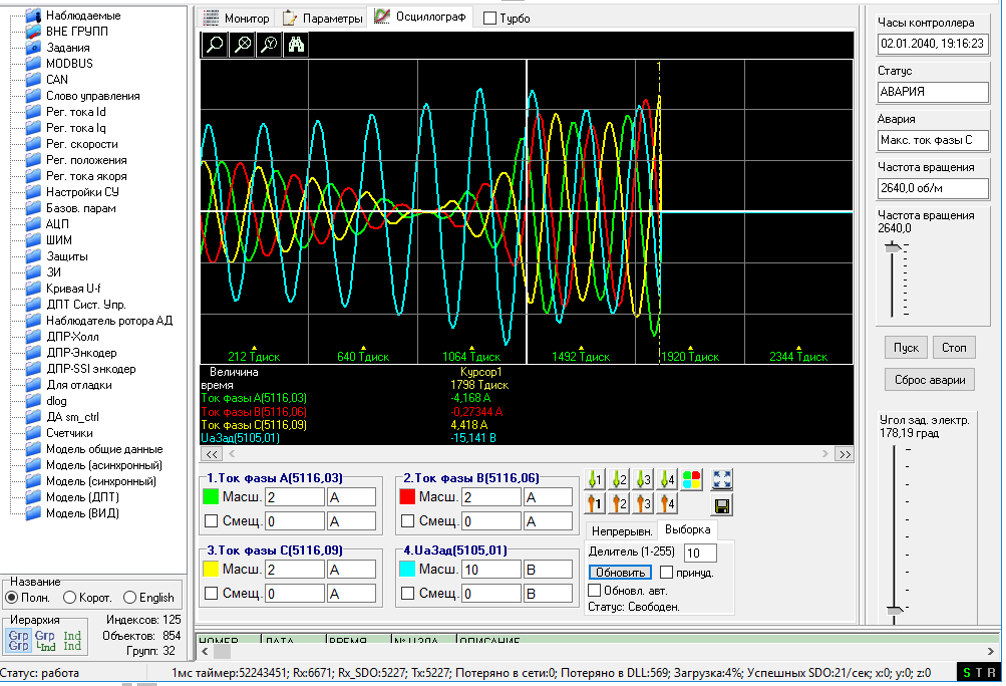


Рис. .. Форма токов фаз при срабатывании максимально-токовой защиты при разгоне привода в частотно-токовом управлении

На данном этапе лабораторная работа с частотно-токовым управлением закончена, читатель может дальше самостоятельно проводить исследования работы привода в этом режиме управления.

### Проверка и настройка ДПР

Перед тем как запустить векторную СУ, нужно проверить работу датчиков положения ротора. Необходимо выбрать используемый тип датчика положения в параметре **Базов. парам.Тип датчика положения [5113,09]** (инкрементальный энкодер или на датчиках Холла). В случае инкрементального энкодера необходимо настроить число меток в параметре **ДПР-Энкодер.Кол-во периодов [5152,01]**, а также задать наличие индексной метки в параметре **ДПР-Энкодер.Наличие репера [5152,02]**. Параметры реального датчика положения можно в его документации. Следует отметить, что существует некоторая путаница между количеством периодов датчика и его количеством импульсов. При количестве периодов датчика, например, 1000, суммарное количество импульсов по обоим каналам на обороте составит 4000. В параметр **ДПР-Энкодер.Кол-во периодов [5152,01]** вводится именно число до учетверения. После выбора и настройки датчика необходимо подать команду сохранения параметров и перезагрузить микроконтроллер, сбросив питание или вписав «1» в параметр **Настройки СУ.Перезагрузка [5114,05]**.

Далее необходимо следует запустить двигатель в частотно-токовой системе управления (режим работы 4, см. п.п. 5.5.1), задать небольшую скорость, к примеру, 2 об/мин и наблюдать осциллограммы сигналов токов и сигналов датчиков положения: **ДПР-Холл.ЭлУгол [5155,01]** (для датчика на элементах Холла) или **ДПР-Энкодер.Угол. электр. [5152,08]** (угловое положение с квадратурного энкодера в электрических градусах).

Корректный сигнал с ДПР для датчика на элементах Холла представлен на Рис. 5.12, а для инкрементального энкодера – на Рис. 5.13.

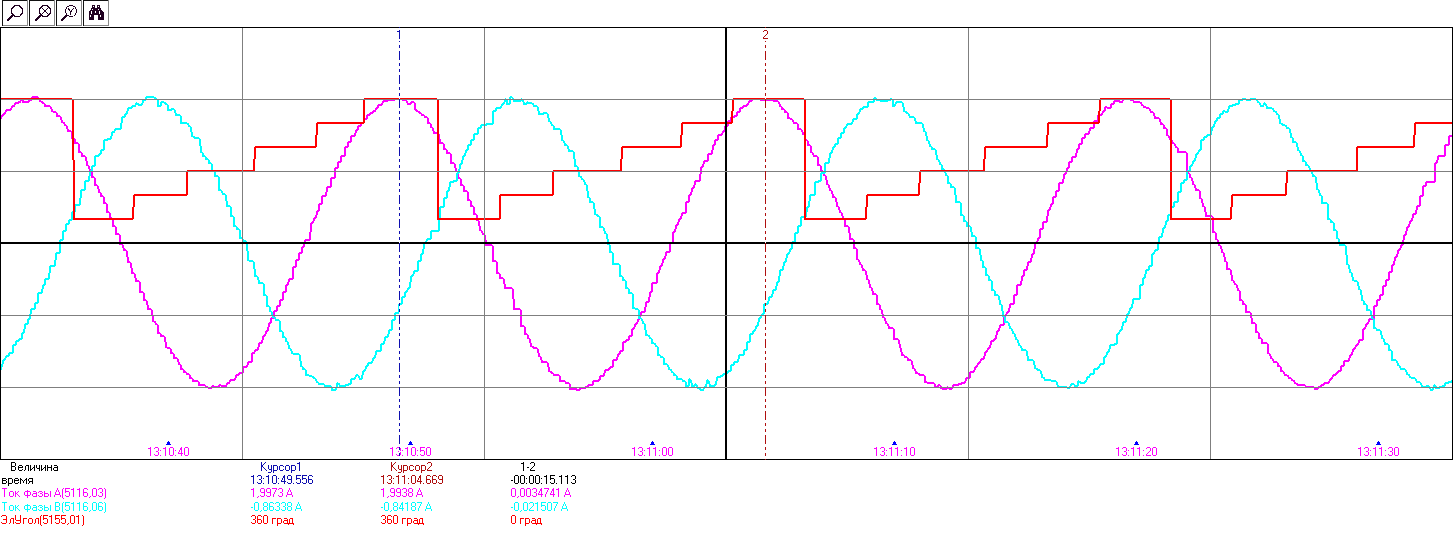


Рис. . Осциллограмма сигнала ДПР на элементах Холла

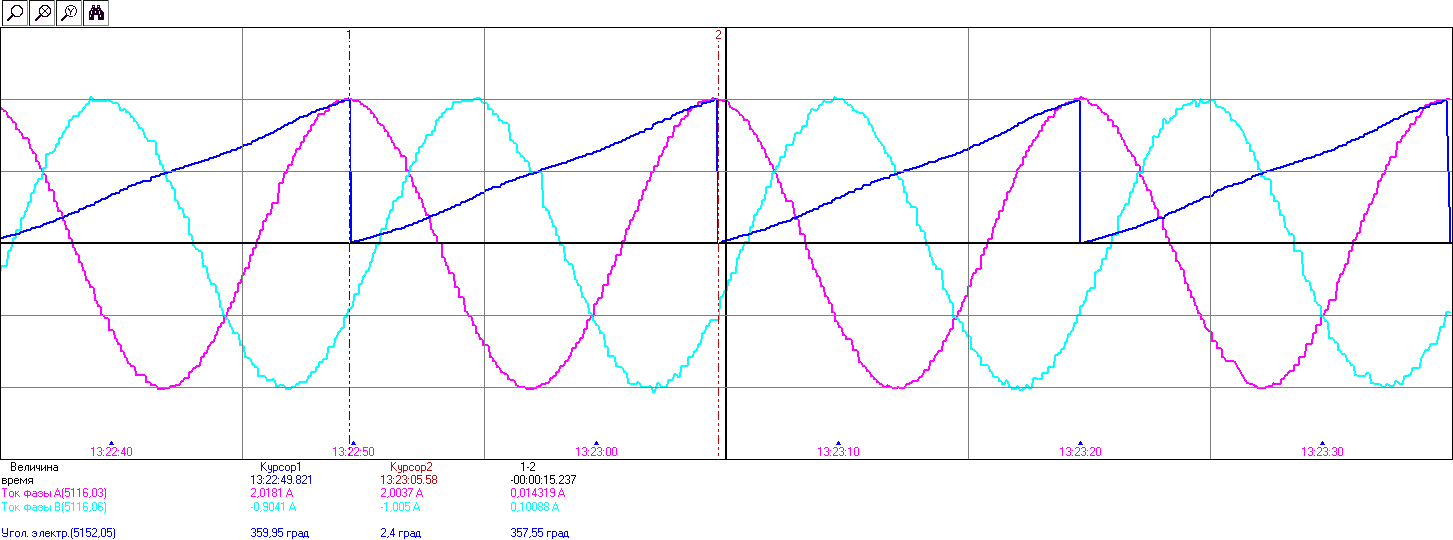


Рис. . Осциллограмма сигнала инкрементального энкодера

Сигнал с ДПР любого типа должен иметь равномерно возрастающий характер, его период должен совпадать с периодом тока в фазах, а обнуление углового сигнала ДПР должно происходить в районе максимума тока в фазе A. В этом случае датчик считается настроенным.

Если сигнал ДПР отсутствует или «совсем не похож» на представленные или отсутствует, проверьте правильность подключения проводов ДПР к разъемам платы контроллера. Полезным может быть наблюдение параметра, показывающего двоичные значения входов контроллера, на которые заведены каналы датчика. Они отображены в параметрах **ДПР-Энкодер.GPIOsValue [5152h.0Dh]** и **ДПР-Холл.Дискретные входы [5155h.09h]**.

Если характер сигнала ДПР спадающий, а не нарастающий, поменяйте местами любые две фазы питания двигателя или датчика положения. Если период сигнала ДПР отличается от периода тока, проверьте **параметр Базов. парам.Число пар полюсов**: он настроен неправильно (не соответствует двигателю). 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 микроконтроллер. В случае инкрементального энкодера необходимо также проверить корректность настроенного числа меток. Если обнуление сигнала ДПР не совпадает по времени с максимумом тока в фазе A (так и должно быть в общем случае), настройте смещение датчика положения параметрами **ДПР-Холл.СмещУгол** или **ДПР-Энкодер. СмещУгол** так, чтобы эти два события происходили одновременно. Для инкрементального энкодера такую настройку смещения необходимо делать каждый раз при включении в работу, так как процедура «причаливания» и сигнал индексной метки в данном ПО не используются соответствующим образом. 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 [5114,02]** = 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 относительно текущего углового положения вектора тока. 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**).

На данном этапе настройка датчиков положения закончена.

### Векторное двухконтурное управление

Для запуска двигателя в векторной системе управления необходимо предварительно настроить регуляторы токов (п.п. 5.5.1) и датчик положения ротора (п.п. 5.6.2). Далее для выбора векторной двухконтурной структуры управления (п.п. 4.1.4) необходимо установить **Настройки СУ.режим работы** = 5.

Векторная СУ с плохо настроенным регулятором скорости или грубым датчиком положения (на элементах Холла или инкрементальным энкодером с малым числом меток) обычно неустойчиво работает на низких скоростях (менее 100-500 об/мин в зависимости от двигателя), поэтому задавать следует бо́льшие значения задания скорости в параметре **Задания.Скорость [5112,01]**. Также необходимо установить желаемое токоограничение амплитуды тока в параметре **Задания.Макс. ток ВУ [5112,02]**.

Настройки регулятора скорости в группе **Рег. Скорости** рекомендуется для первого пуска задать мягкими: Кп порядка 0.05, Ки 0.01. В осциллограф удобно добавить параметры из группы **Наблюдаемые**: скорость, амплитудный ток, мощность и оставить работать в непрерывном режиме. Далее можно произвести пуск векторной системы управления.

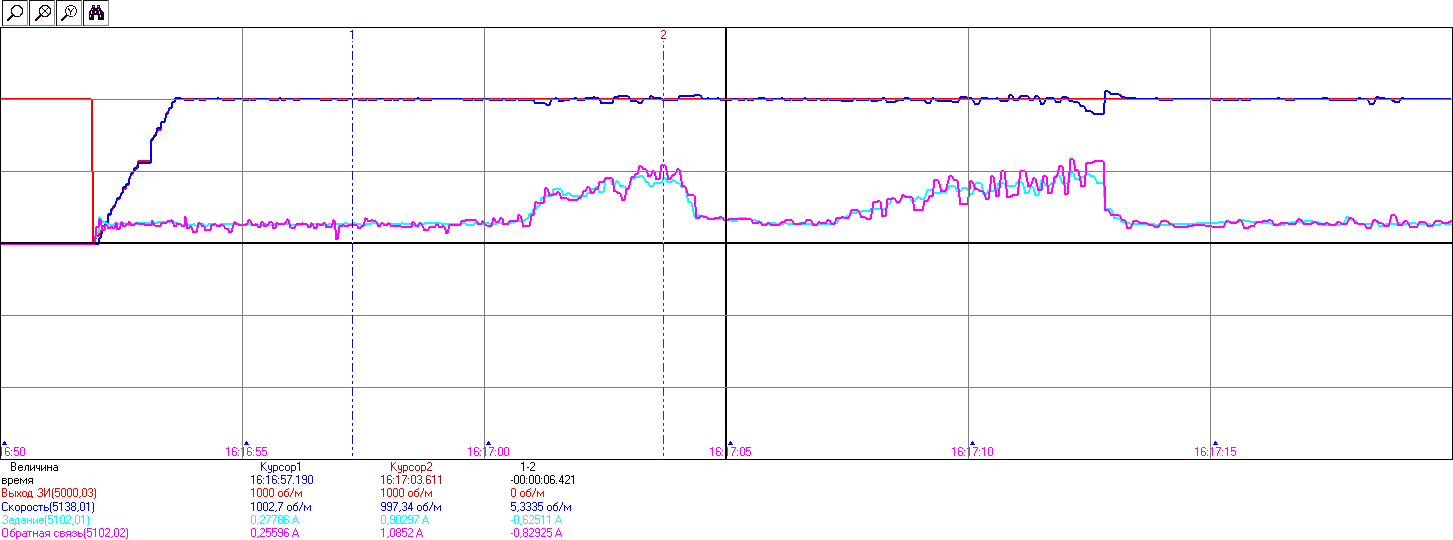


Рис. . Осциллограмма работы векторной системы управления

На Рис. 5.14 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).

Регулятор скорости векторной СУ, как и регуляторы тока, рекомендуется использовать пропорционально-интегральный. Его настройки так же можно изменять для получения требуемой динамики и точности регулирования скорости. Настройку можно проводить по методике, аналогичной настройке регуляторов тока, только наблюдать в осциллограмме уже частоту вращения и задание на ток. Для такой методики необходимо предварительно отключить задатчик интенсивности, обнулив **ЗИ.T разгона до номинала [5000h.01h]**. При настройке следует следить, чтобы задание на ток не достигало токоограничения. Если оно достигает, следует либо увеличить токоограничение, либо снизить задание скорости.

Также при использовании инкрементального датчика положения и программного способа расчета скорости на поведение привода сильно влияет параметр фильтрации сигнала по скорости **ДПР-Энкодер.К быстродействия фильтра скорости [5152h.0Bh]**.

### Позиционное управление

Для запуска структуры, замкнутой по положению необходимо использовать инкрементальный датчик положения ротора. Необходимо пройти все шаги пусконаладочных работ до п.п. 5.6.3 включительно и качественно настроить регулятор скорости в этом пункте. Далее необходимо задать **Настройки СУ.режим работы** = 10, чтобы выбрать замкнутую по положению структуру управления, а затем обнулить параметр **ДПР-Энкодер.Число оборотов**: если он окажется не нулевой, то привод после запуска начнет «отматывать» обороты назад, стремясь к нулевому заданию положения (и нулевому счетчику оборотов). Параметры регулятора положения по умолчанию настроены на достаточно жесткое поддержание положения. Для первого запуска рекомендуется сделать регулятор более мягким, а интегральную составляющую отключить полностью. Первый раз рекомендуется запускать привод с небольшим разрешенным током (**Задания.Макс. ток ВУ**), а также ограничить частоту вращения **Задания.Скорость [5112,01]** на низком уровне порядка 50 об/мин. В параметре **Задания.theta\_elec** пользователем задается желаемое положение в электрических градусах, к которому привод должен следовать значением параметра **ДПР-Энкодер.Угол электрический не огранич. [5152h.17h]**. Можно запустить привод и попробовать задать разные положения. На осциллографе в непрерывном режиме интересно смотреть задание углового положения, текущее угловое положение, текущий ток статора и скорость из группы **Наблюдаемые**.

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. Рекомендуется, как минимум, испытать две схемы с регуляторами: ПИ-регулятор скорости и П регулятор положения, а также П-регулятор скорости и ПИ-регулятор положения (в этом случае значение **Задания.Скорость [5112,01]** должно быть большим, чтобыпропорциональный регулятор скорости мог реализовать полное задание на ток). Регуляторы тока всегда должны быть ПИ.

После завершения настройки можно сохранить все параметры привода в энергонезависимую память. Это делается в программе RTCON путем нажатия на кнопку «Сохранить параметры», расположенной справа сверху. При этом значения всех параметров CANopen, доступных на запись, сохранятся в пользовательской флеш памяти микроконтроллера и будут в дальнейшем восстанавливаться при перезагрузке микроконтроллера в процедуре инициализации драйвера CANopen. Также можно сохранить все настройки привода в файл на компьютер, воспользовавшись меню Сервис-Перенос параметров (подробнее см. руководство пользователя на RTCON).

## Системы управления асинхронным двигателем

### Скалярное управление

Запуск скалярного управления, как самого простого, вместе с настройкой защит и модуля АЦП подробно рассмотрен в п.п. 5.5.

### Настройка регуляторов тока

Настройка регуляторов тока асинхронной машины ничем не отличается от настройки регуляторов синхронной, поэтому необходимо обратиться к методике п.п. 5.5.1.

### Частотно-токовое управление

Настройки частотно-токовой системы управления асинхронной машины похожа на аналогичную для синхронной машины, поэтому необходимо обратиться к методике п.п. 5.6.1. Принципиальное отличие заключается в том, что асинхронный двигатель не может «вывалиться из синхронизма» по принципу работы, а поэтому работает более устойчиво и имеет меньше проблем с запуском. Тем не менее, некоторая колебательность асинхронного двигателя в данной структуре управления также может присутствовать.

### Проверка и настройка ДПР

Перед тем как запустить векторную СУ, нужно проверить работу датчиков положения ротора. Для векторной системы АД в данном ПО предусмотрен только один датчик положения – инкрементальный. Необходимо выбрать данный тип датчика положения в параметре **Базов. парам.Тип датчика положения [5113,09]** и настроить число меток в параметре **ДПР-Энкодер.Кол-во периодов [5152,01]**, а также задать наличие индексной метки в параметре **ДПР-Энкодер.Наличие репера [5152,02]**. Параметры реального датчика положения можно в его документации. Следует отметить, что существует некоторая путаница между количеством периодов датчика и его количеством импульсов. При количестве периодов датчика, например, 1000, суммарное количество импульсов по обоим каналам на обороте составит 4000. В параметр **ДПР-Энкодер.Кол-во периодов [5152,01]** вводится именно число до учетверения. После выбора и настройки датчика необходимо подать команду сохранения параметров и перезагрузить микроконтроллер, сбросив питание или вписав «1» в параметр **Настройки СУ.Перезагрузка [5114,05]**.

Далее необходимо следует запустить двигатель на холостом ходе в частотно-токовой системе управления (режим работы 4, см. п.п. 5.7.3), задать небольшую скорость, к примеру, 2-10 об/мин и наблюдать осциллограммы сигналов токов и, убедившись что двигатель крутится, нужно наблюдать сигнал датчика положения **ДПР-Энкодер.Угол. электр. [5152,08]** (угловое положение с квадратурного энкодера в электрических градусах).

Нужно убедиться, что при вращении двигателя угловое положение с датчика увеличивается вверх, а параметр **ДПР-Энкодер.Скорость [5152,07]** показывает положительное значение скорости. Если это не так и датчик работает «в обратную сторону», то нужно либо перевернуть две фазы двигателя, либо два канала квадратурного сигнала датчика.

Далее необходимо разогнать двигатель на боле высокую скорость (не меньше четверти номинала) и задать амплитуду тока также не меньше четверти номинала. Убедиться, что значение скорости в параметре **ДПР-Энкодер.Скорость [5152,07]** чуть ниже (на 1-5%) чем задание скорости. Если разница в задании скорости и текущей скорости по датчику больше, то необходимо найти проблему: наиболее вероятно, что неверно задано количество пар полюсов двигателя, количество меток датчика, или СУ не была перезагружена после сделанных изменений настроек.

# Заключение

Описанное в документе программное обеспечение является демонстрационным, реализует основные базовые возможности микроконтроллера и базовые системы управления для систем электроприводов.

Пользователь в праве видоизменять представленное ПО под свои задачи, переписывая его как целиком, так и отдельные модули. При этом пользователь должен осознавать сопутствующие риски.

ООО «НПФ ВЕКТОР» и разработчики демонстрационного ПО MCB4\_Demo не несут ответственности за возможный ущерб при работе указанного ПО на оборудовании пользователя.

По любым проблемам, вопросам и предложениям обращайтесь на специальные раздел интернет-форума ООО «НПФ ВЕКТОР»:

* <https://motorcontrol.ru/forum/viewtopic.php?f=8&t=326>



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

**Приложение 1**

**Cигналы отладочной платы MCB4**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **PIN** | **GPIO** | **I/O** | **Сигнал на проце** | **Сигнал на плате** | **Разъем** | **Применение** | **Примечание** |  |
| 64 | GPIOF6 | O | CANTXA | CAN\_TX\_OI/CAN\_TX\_MCB | X1(49) | CAN на внешний преобразователь | Выбирается джампером | |
| 65 | GPIOF7 | I | CANRXA | CAN\_RX\_OI/CAN\_RX\_MCB | X1(50) |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  | CAN\_H | X5(1) | CAN на плате MCB4 | Выбирается джампером | |
|  |  |  |  | CAN\_L | X5(2) |
|  | | | | | | | | |
| 111 | GPIOF4 | O | SCITXDB | SCITXDB | X1(61) | UART1 |  | |
| 112 | GPIOF5 | I | SCIRXDB | SCIRXDB | X1(62) |  | |
| 66 | GPIOG4 | O | SCITXDA | SCITXDA | - | UART2 | С USB на процессор | |
| 67 | GPIOG5 | I | SCIRXDA | SCIRXDA | - |
|  | | | | | | | | |
| 19 | GPIOF12 | O | MDXA | MDXA | X1(53) | SPI |  | |
| 18 | GPIOF13 | I | MDRA | MDRA | X1(54) |  | |
| 21 | GPIOF9 | I/O | MCLKRA | MCLKRA | X1(55) |  | |
| 22 | GPIOF10 | I/O | MFSXA | MFSXA | X1(56) |  | |
|  | | | | | | | | |
| 31 | GPIOF0 | O | SPISIMOA | EEPROM\_SIMO | - | Модуль энергонезависимой памяти |  | |
| 32 | GPIOF1 | I | SPISOMIA | EEPROM\_SOMI | - |  | |
| 27 | GPIOF2 | I/O | SPICLKA | EEPROM\_CLK | - |  | |
| 28 | GPIOF3 | I/O | SPISTEA | EEPROM\_CS | - |  | |
|  | | | | | | | | |
| 68 | GPIOA0 | O | PWM1 | PWM1 | X1(11) | ШИМ | Управление стойкой 1 (PWM1 - верхний, PWM2 - нижний) | |
| 69 | GPIOA1 | O | PWM2 | PWM2 | X1(12) |
| 70 | GPIOA2 | O | PWM3 | PWM3 | X1(13) | Управление стойкой 2 (PWM3 - верхний, PWM4 - нижний) | |
| 71 | GPIOA3 | O | PWM4 | PWM4 | X1(14) |
| 72 | GPIOA4 | O | PWM5 | PWM5 | X1(15) | Управление стойкой 3 (PWM5 - верхний, PWM6 - нижний) | |
| 75 | GPIOA5 | O | PWM6 | PWM6 | X1(16) |
| 33 | GPIOB0 | O | PWM7 | PWM7 | X1(17) | - | |
| 34 | GPIOB1 | O | PWM8 | PWM8 | X1(18) | Вентилятор | |
| 35 | GPIOB2 | O | PWM9 | PWM9 | X1(19) | Управление зарядом ЗПТ | |
| 36 | GPIOB3 | O | PWM10 | PWM10 | X1(20) | - | |
| 37 | GPIOB4 | O | PWM11 | PWM11 | X1(21) | - | |
| 38 | GPIOB5 | O | PWM12 | PWM12 | X1(22) | - | |
| 76 | GPIOA6 | I | T1PWM | T1PWM | X1(23) | - | |
| 77 | GPIOA7 | I | T2PWM | T2PWM | X1(24) | - | |
| 40 | GPIOB6 | I | T3PWM | T3PWM | X1(25) | - | |
| 41 | GPIOB7 | I | T4PWM | T4PWM | X1(26) | - | |
| 54 | GPIOB11 | I | GPIO | TDIRB | X1(27) | Сигнал общей аварии | |
| 55 | GPIOB12 | I | GPIO | TCLKINB | X1(28) | Сброс аварии | |
|  |  |  |  |  |  |  |  | |
| 81 | GPIOD0 | I | PDPINTA | PDPINTA | X1(31) | Аварии | Общая авария | |
| 84 | GPIOD1 | I | EVASOC | EVASOC | X1(32) | Авария на стойке 1 | |
| 60 | GPIOD5 | I | PDPINB | PDPINB | X1(33) | Авария на стойке 2 | |
| 61 | GPIOD6 | I | EVBSOC | EVBSOC | X1(34) | Авария на стойке 3 | |
| 47 | GPIOB14 | I | C5TRIP | C5TRIP | X1(35) | Авария на нижних ключах | |
| 48 | GPIOB15 | I | C6TRIP | C6TRIP | X1(36) | - | |
|  |  |  |  |  |  |  |  | |
| 78 | GPIOA8 | I | QEP1 | QEP1 | X1(39) | Квадратурный энкодер |  | |
| 79 | GPIOA9 | I | QEP2 | QEP2 | X1(40) |  | |
| 80 | GPIOA10 | I | QEPI1 | QEPI1 | X1(41) |  | |
|  |  |  |  |  |  |  |  | |
| 43 | GPIOB8 | I | CAP4 | CAP4 | X1(44) | Модуль захвата |  | |
| 44 | GPIOB9 | I | CAP5 | CAP5 | X1(45) |  | |
| 45 | BPIOB10 | I | CAP6 | CAP6 | X1(46) |  | |
|  |  |  |  |  |  |  |  | |
| 85 | GPIOA11 | I | GPIO | TDIRA | X1(5) | GPIO |  | |
| 86 | GPIOA12 | I | GPIO | TCLKINA | X1(6) |  | |
| 89 | GPIOA13 | I | GPIO | C1TRIP | X1(3) |  | |
| 90 | GPIOA14 | I | GPIO | C2TRIP | X1(1) |  | |
| 91 | GPIOA15 | I | GPIO | C3TRIP | X1(2) |  | |
| 46 | GPIOB13 | I | GPIO | C4TRIP | X1(4) |  | |
| 108 | GPIOE1 | i | GPIO | XINT2\_ADCSOC | X1(7) |  | |
| 101 | GPIOF14 | O | GPIO | XF\_PLL\_DIS | VD6 |  | |
| 106 | GPIOE0 | I | GPIO | XINT1\_XBIO | VD7 |  | |
| 107 | GPIOE2 | I | GPIO | XNMI\_XINT13 | X1(65) |  | |
|  |  |  |  |  |  |  |  | |
| 119 | ADCINA7 | AI | ADCINA7 | ADCINA7 | X2(15) |  | Внешний АВ2 (Uсм - JP4 на ОИ ) |  |
| 120 | ADCINA6 | AI | ADCINA6 | ADCINA6 | X2(13) |  | Внешний АВ1 (Uсм - JP3 на ОИ ) |  |
| 121 | ADCINA5 | AI | ADCINA5 | ADCINA5 | X2(11) |  | Внешний потен. |  |
| 122 | ADCINA4 | AI | ADCINA4 | ADCINA4 | X2(9) |  | Смещение 0,03 В |  |
| 123 | ADCINA3 | AI | ADCINA3 | ADCINA3 | X2(7) |  | Смещение 0,75 В |  |
| 124 | ADCINA2 | AI | ADCINA2 | ADCINA2 | X2(5) |  | Напряжение ЗПТ |  |
| 125 | ADCINA1 | AI | ADCINA1 | ADCINA1 | X2(3) |  | Ток фазы W |  |
| 126 | ADCINA0 | AI | ADCINA0 | ADCINA0 | X2(1) |  | Ток фазы U |  |
| 9 | ADCINB7 | AI | ADCINB7 | ADCINB7 | X3(4) | - | - | |
| 8 | ADCINB6 | AI | ADCINB6 | ADCINB6 | X3(3) | - | - | |
| 7 | ADCINB5 | AI | ADCINB5 | ADCINB5 | X3(2) | - | - | |
| 6 | ADCINB4 | AI | ADCINB4 | ADCINB4 | X3(1) | - | - | |
| 5 | ADCINB3 | AI | ADCINB3 | ADCINB3 | X2(23) | - | - | |
| 4 | ADCINB2 | AI | ADCINB2 | ADCINB2 | X2(21) | - | - | |
| 3 | ADCINB1 | AI | ADCINB1 | ADCINB1 | X2(19) | - | - | |
| 2 | ADCINB0 | AI | ADCINB0 | ADCINB0 | X2(17) |  | Датчик темп. |  |