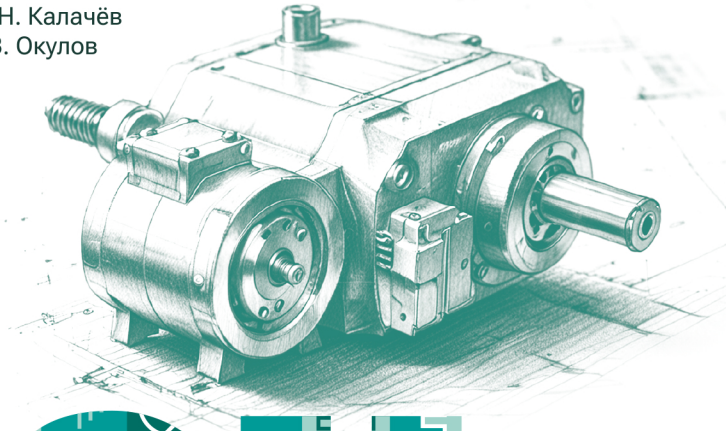
SIM IN TECH

Ю.Н. Калачёв
Е.В. Окулов



Проектирование электроприводов

Руководство практика

Ю.Н. Калачёв Е.В. Окулов

Проектирование электроприводов

(Руководство практика)

В книжке нет формул
и много красивых картинок

ДМК ПРЕСС
ИЗДАТЕЛЬСТВО

Москва, 2025

УДК 681.1
БКК 31.291
К17

Калачёв Ю. Н., Окулов Е. В.
К17 Проектирование электроприводов — М.: ДМК Пресс, 2025.
- 69 с.: ил.
ISBN 978-5-93700-375-1

Данная книга ориентирована на инженеров, начинающих заниматься разработками электроприводов и преобразователей.

В ней кратко изложены основные принципы проектирования этих устройств, приведены структуры их аппаратной и программной частей. Расставлены акценты, на которые необходимо обратить особое внимание.

На примере реального электропривода рассмотрен процесс построения его модели с последующей кодогенерацией программы управляющего микроконтроллера.

ISBN 978-5-93700-375-1

Оглавление

Тот Боян, исполнен дивных сил,
Приступая к вещему напеву,
Серым волком по полю кружил,
Как орёл, под облаком парил,
Растекался мыслию по древу.

Н. Заболоцкий

1 ПРЕДИСЛОВИЕ

Здесь авторы не хотят, подобно вещему Бояну, «растекаться мыслию по древу».

На то есть много разных теоретически и дидактически правильных учебников…

Эту книжку можно рассматривать как продолжение книги «Основы регулируемого электропривода» Ю. Н. Калачёва и Д. В. Самохвалова, так как она посвящена практической реализации описанных там принципов.

Будем считать, что с основами электропривода читатель знаком, и перейдём непосредственно к практике его проектирования, а поможет нам в этом среда *SimInTech*.

Это отечественная программная среда, позволяющая моделировать и проектировать различные устройства, в том числе и электроприводы.

«А почему, например, не *Simulink*?» - спросите вы, и мы вам ответим: ***«Потому что …!»***

2 Почему же всё-таки *SimInTech*

Хватит жить чужим умом - мы предлагаем отечественный софт, который по ряду показателей превосходит зарубежные аналоги.

А тулбокс «Электропривод» *SimInTech* спроектирован разработчиками электроприводов себе же в помощь, именно так, как удобно самим этим разработчикам … Более того, он и проектировался в процессе конкретных разработок.

Очень интересным, на наш взгляд, является и практический опыт кодогенерации программы контроллера электропривода, описанный в данной книге.

А ещё разработчики среды *SimInTech* вполне адекватные и доступные люди, которые всегда готовы ответить на любой вопрос пользователя, и это действительно важно.

Также среди преимуществ *SimInTech* стоит упомянуть скорость счёта моделей и «лёгкость» среды с точки зрения требований к «железу».

3 Этапы проектирования

Любое проектирование начинается с анализа и корректировки технического задания (ТЗ).

Этот очень ответственный этап, так как часто изменить что-то после утверждения ТЗ уже невозможно.

На этом этапе работы необходимо согласовать с заказчиком требования к изделию, которые, с одной стороны, удовлетворят его желания, а с другой, будут реалистичны, как со стороны технической реализации, так и со стороны других немаловажных аспектов:

- финансов
- сроков разработки
- оборудования для испытаний
- сопроводительной документации
- сертификации
- и т.д.

В утверждённом ТЗ предварительно определяются такие параметры привода как напряжение питания и ток потребления, номинальная и максимальная мощность, момент или усилие на рабочем органе привода, КПД системы и т.д.

Следующий этап разработки – создание функциональной схемы привода. На этом этапе определяются тип и структура входящих в привод основных функциональных блоков, таких как:

- электродвигатель (тип, характеристики)
- силовой блок (тип и структура)
- датчиков (типы и варианты установки)
- система управления (структура).

Ну а далее - проработка принципиальных схем, разводка печатных плат, написание программ управления и связи, изготовление, испытания и … исправление ошибок, снова испытания и т.д.

Наш путь тернист, и на нём часто возникают непредвиденные материальные и временные затраты, например, на реанимацию, или (не дай Бог) реинкарнацию взорвавшихся узлов …

А начальство недовольно, оно-то думает: «*Чего там сложного – битики перещёлкивать?!!!*».

Вот тут нам может здорово помочь моделирование. Оно позволяет обнаружить ошибки на ранних стадиях проектирования, не доводя до испытаний (и взрывов).

4 Выбор двигателя

Важными факторами при выборе типа двигателя являются его масса, габариты, КПД и стоимость.

Если говорить о двигателях переменного тока, то самыми дешёвыми являются асинхронные двигатели (далее АД).

Однако самый высокий КПД имеют синхронные двигатели с постоянными магнитами (далее СДПМ). Они же имеют лучшие массогабаритные показатели.

Тип выбираемого двигателя определяется тем фактором, который в конкретном случае важнее.

Основными требованиями, предъявляемыми к любому типу двигателя, являются:

- мощность
- момент
- рабочие скорости вращения

Идеальным является тот случай, когда двигатель специально проектируется под сформулированные в Т.З. на привод требования. Однако чаще бывает по-другому. Проектировщику приходится использовать уже существующие доступные двигатели, которые не всегда идеальны для решения поставленной задачи.

Однако чудес не бывает, как бы этого ни хотелось не очень компетентным начальникам.

Если требуется, например, высокая динамика и точность, глубокое регулирование с минимальными пульсациями момента, то, скорее всего, надо применять дорогой СДПМ с синусоидальной ЭДС.

А если требования к пульсациям момента на так строги, то можно применить и более дешёвый синхронный двигатель с трапецеидальной ЭДС.

Если требования к массе и габаритам позволяют, то можно использовать и асинхронный двигатель.

В некоторых случаях, при небольшой требуемой мощности и мало изменяющемся моменте нагрузки, идеальным выбором является шаговый двигатель.

Что касается двигателей постоянного тока, то их мы здесь не рассматриваем в силу того, что они, вследствие капризности щёточно-коллекторного механизма и дороговизны, применяются достаточно редко в основном – по старой памяти.

5 Механическая передача

Смысл использования механической передачи, как правило, заключается в повышении момента за счёт снижения скорости, ну и/или в преобразовании вращательного движения в возвратно-поступательное.

Наличие механической передачи в приводе предполагает сопутствующие неприятности: люфты, упругости, снижение КПД, необходимость дополнительного обслуживания (смазка).

Эти неприятности увеличивают стоимость привода, снижают его надёжность, точность и быстродействие.

Однако в некоторых случаях без редуктора не обойтись.

Например, при одинаковом моменте маломощный высокоскоростной двигатель с понижающим редуктором может иметь значительно меньшие габариты и массу, чем двигатель с большим количеством пар полюсов.

На больших мощностях картина часто обратная. Высокомоментный многополюсный двигатель может быть в несколько раз меньше, легче и дешевле, чем, например, 50-герцовый двигатель с понижающим редуктором, обеспечивающим нужную скорость и момент.

Выбор механической передачи (или её исключение за счёт выбора соответствующего двигателя) – это вопрос оптимизации структуры конкретного привода, при решении которого необходимо учитывать все требования соответствующего ТЗ.

6 Структура силового преобразователя

Питание силового преобразователя электропривода может осуществляться от источника постоянного напряжения (например, аккумулятора) или от внешней сети переменного тока.

Общий вид структуры преобразователя при питании от внешней сети переменного тока приведён на Рис. 6.1.

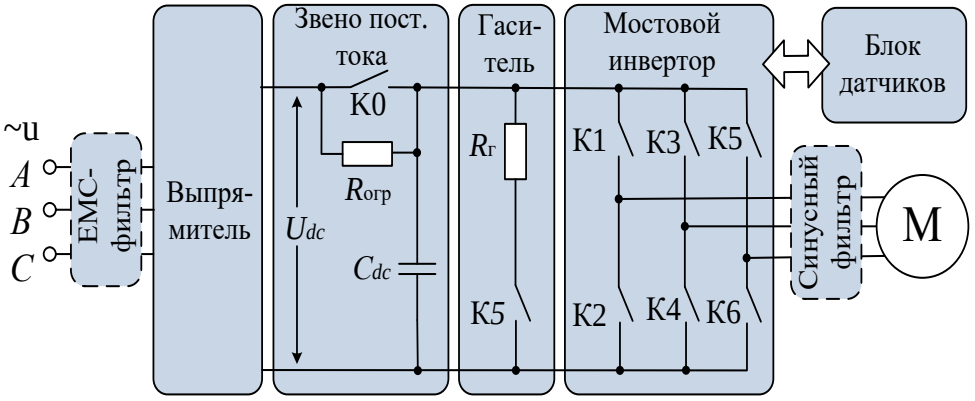


Рис. 6.1

Рассмотрим структуру преобразователя более подробно.

В его состав входят:

- выпрямитель
- звено постоянного тока
- гаситель энергии торможения
- мостовой инвертор
- датчики
- при необходимости фильтры (EMC и синусный).

В зависимости от требований ТЗ структура силового преобразователя может меняться. Например, при питании от сети постоянного тока не понадобится выпрямитель, а если привод работает только в двигательном режиме, то и тормоз-гаситель не нужен.

При работе привода от аккумуляторной батареи также можно отказаться от гасителя, обеспечив при торможении режим рекуперации энергии в батарею.

На малых мощностях или при отсутствии соответствующих требований ТЗ не используют и входные/выходные фильтры.

6.1 Выпрямитель

6.1.1 Диодный выпрямитель

Самым простым является диодный выпрямитель, однофазный или трёхфазный.

Важно помнить, что при работе диодного выпрямителя на конденсатор звена постоянного тока (C_{dc} на Рис. 6.1) токи фаз силовой сети носят ярко выраженный импульсный характер. Эти токи могут по амплитуде во много раз превосходить средний ток, потребляемый инвертором электропривода. Такой характер потребления может плохо сказываться на форме питающего напряжения – создавать в сети дополнительные гармоники.

Дабы несколько сгладить этот неприятный эффект, иногда применяют три дросселя, которые устанавливают в фазы перед выпрямителем, или один дроссель, между выпрямителем и фильтром звена постоянного тока (см. Рис. 6.2).

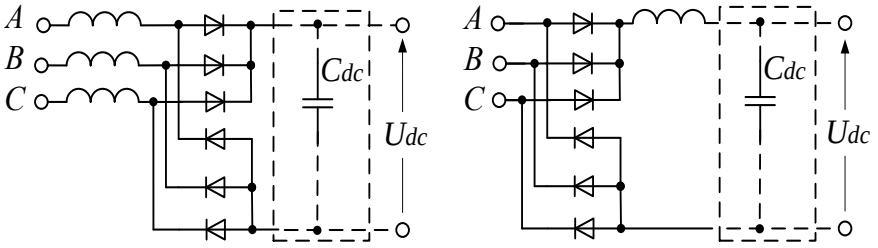


Рис.6.2

Диоды выпрямителя выбираются с учётом работы в импульсном режиме, с хорошим запасом по току. Величину импульсного тока можно оценить аналитически или более точно получить на модели.

Обычно заказчик указывает в ТЗ ссылку на стандарты, которым должно соответствовать разрабатываемое устройство. Например, современный стандарт *IEC*-1000-3-2 налагает жёсткие ограничения на уровень гармоник, создаваемых в сети электропитания любыми устройствами мощностью свыше 75 Вт.

Дроссели на Рис. 6.2 немного уменьшают уровень высших гармоник тока и напряжения сети, но не решают проблему радикально, так как для хорошего сглаживания они должны иметь достаточно большую индуктивность, что часто приводит к недопустимому возрастанию их габаритов, веса и стоимости.

Соблюсти требования стандарта помогают различные схемы корректоров мощности.

БОЛЕЕ ПОДРОБНО о корректорах мощности можно почитать в книге: В. Милешин, Д. Овчинников, Управление транзисторными преобразователями электроэнергии, Техносфера, Москва 2011.

К корректорам мощности можно отнести и активный выпрямитель, который, кроме снижения уровня гармоник, ещё обеспечивает и рекуперацию энергии. О нём речь пойдёт далее.

6.1.2 Выпрямители с рекуперацией энергии

Активный выпрямитель обеспечивает активный характер тока сети по отношению к её напряжению, что идеально с точки зрения минимизации уровня гармоник, возникающих в сети при выпрямлении.

Кроме того, с помощью этого устройства энергию генераторного торможения двигателя можно отдавать обратно в силовую сеть (рекуперировать). Структуру выпрямителя поясняет Рис. 6.3.

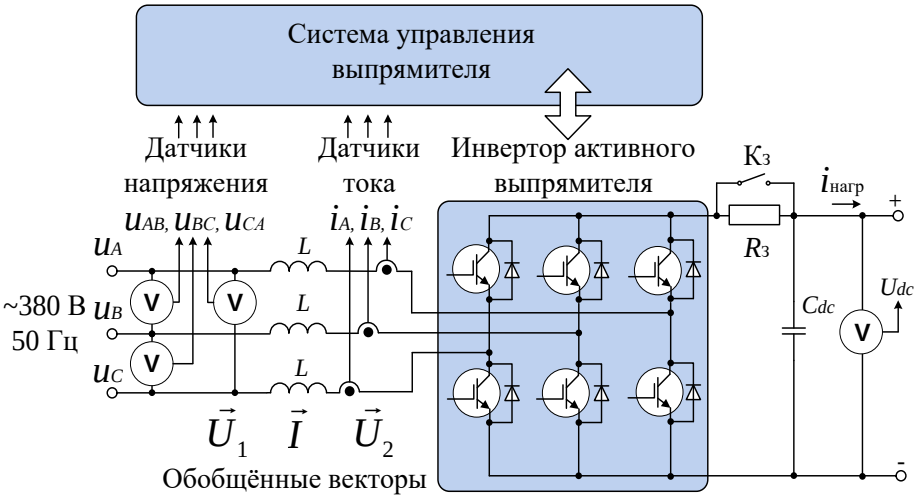


Рис. 6.3

Состоит активный выпрямитель из мостового инвертора, подключаемого к сети через дроссели (см. Рис. 6.3) и системы управления.

При включении выпрямителя инвертор, работающий в режиме повышающего преобразователя, заряжает ёмкость C_{dc} до напряжения U_{dc}, величина которого несколько выше амплитуды питающего напряжения сети.

После этого U_{dc} можно рассматривать как источник напряжения, из которого инвертор с помощью ШИМ способен формировать трёхфазные напряжения различной амплитуды и фазы относительно нулевой (средней) точки силовой сети.

На Рис. 6.3 вектор напряжения сети имеет обозначение $\vec{U}_1$, а $\vec{I}$ и $\vec{U}_2$ – это вектор тока трёхфазного дросселя и вектор напряжения, формируемого инвертором.

ПОДРОБНО об обобщённых пространственных векторах читайте в разделе 4 книги «Основы регулируемого электропривода» Ю. Н. Калачёва и Д. В. Самохвалова.

Измеряя фазу вектора силовой сети ($\vec{U}_1$) и управляя с помощью инвертора с ШИМ фазой вектора $\vec{U}_2$, можно менять фазу вектора тока дросселей $\vec{I}$ относительно $\vec{U}_1$. А управляя амплитудой вектора $\vec{U}_2$, можно менять амплитуду $\vec{I}$, поддерживая постоянным напряжение U_{dc} .

Таким образом, можно обеспечить синфазность векторов $\vec{U}_1$ и $\vec{I}$ при потреблении энергии из сети или их противофазность при передаче энергии в сеть, что и соответствует активному режиму обмена энергией между преобразователем и сетью.

Заметим, что возможность передачи энергии в сеть обеспечивается тем, что, как уже было сказано, поддерживаемое U_{dc} несколько превышает амплитуду напряжения сети.

За счёт повышения тактовой частоты ШИМ в схеме выпрямителя удаётся снизить индуктивность, а следовательно, массу и габариты дросселей при обеспечении синусоидальной формы токов фаз и малых токовых ШИМ- пульсациях.

Кроме активного выпрямителя для рекуперации энергии в преобразователях иногда применяются тиристорные выпрямители. Они, как и активный выпрямитель, могут обеспечить плавный заряд фильтра и рекуперацию, однако не обеспечивают активного характера обмена энергией с сетью.

БОЛЕЕ ПОДРОБНО:

- принципы работы активного выпрямителя описаны в книжке: Ю. Н. Калачёв, Д. В. Самохвалов, «Основы регулируемого электропривода (Антиучебник)», ДМК Пресс, раздел 8.
- о тиристорных выпрямителях можно почитать в старой, но хорошей книжке: В. И. Преображенский, «Полупроводниковые выпрямители», второе издание, переработанное и дополненное, Москва, Энергоатомиздат, 1986, разделы 3, 4, 5.

6.2 Звено постоянного тока

Звено постоянного тока состоит из силового конденсаторного фильтра и ограничителя зарядного тока.

При необходимости в его состав входит также гаситель энергии генераторного торможения.

6.2.1 Силовой фильтр

В качестве конденсаторов фильтра в звене постоянного тока обычно применяются электролитические или плёночные конденсаторы большой ёмкости.

К выбору этих конденсаторов надо подходить осознанно.

Электролиты надо использовать только специализированные (импульсные), с низким эквивалентным последовательным сопротивлением (*ESR*). В них меньше потерь. Например, отечественные серии К50-84 или импортные *EPCOS* серии *B*43509.

Потери в плёночных конденсаторах, по сравнению с электролитами, существенно меньше.

Допустимое напряжение конденсаторов фильтра выбирается с некоторым запасом, исходя из максимально возможного напряжения питающей сети и возможного повышения U_{dc} при генераторном торможении двигателя.

Кроме допустимого напряжения, важнейшим параметром для выбора конденсатора является допустимое среднеквадратичное значение его тока (I_{rms}), которое обычно приводится в справочных данных.

В двигательном режиме работы привода конденсатор заряжается через выпрямитель и разряжается в инвертор. При этом значение I_{rms} сильно зависит от формы тока через конденсатор и рассчитать его аналитически затруднительно.

Хорошим инструментом в выборе ёмкости фильтра является модель привода. При этом среднеквадратичное значение тока фильтрующего конденсатора в модели можно оценивать на периоде питающей силовой сети.

При одинаковой ёмкости значение допустимого I_{rms} плёночных конденсаторов в разы выше, чем у электролитических.

В последнее время, в силу лучших частотных свойств, в приводах, особенно мощных, в фильтре звена постоянного тока чаще применяются плёночные конденсаторы, хотя они несколько дороже электролитов.

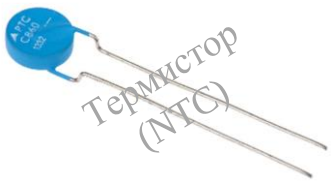
6.2.2 Ограничитель зарядного тока

Ток заряда силового фильтра, возникающий в момент подключения преобразователя к силовой сети, надо ограничивать, как минимум, на уровне допустимых ударных токов диодов выпрямителя.

При мощностях свыше одного киловатта в качестве ограничителя зарядного тока обычно применяется контактор (иногда твердотельное реле) с запараллеленным зарядным резистором (К0 и $R_{огр}$ на Рис. 6.1).

В момент подключения преобразователя к сети ток заряда конденсатора C_{dc} ограничивается величиной резистора $R_{огр}$. После завершения заряда ключ К0 замыкается.

Требуемую мощность рассеяния резистора можно посчитать аналитически, а можно оценить при моделировании. Для обеспечения некоторого запаса по мощности резистор обычно рассчитывают на несколько непрерывных циклов заряда/разряда ёмкости фильтра.



При небольших мощностях привода (до сотен Вт) иногда отказываются от контактора, а для ограничения зарядного тока применяют термисторы *(NTC)* – резисторы, которые резко уменьшают своё сопротивление при нагревании. Термистор включается вместо сопротивления $R_{огр}$ между выпрямителем и фильтром. В момент подключения преобразователя к сети термистор холодный и величина его сопротивления ограничивает зарядный ток. В дальнейшем при работе привода происходит нагрев термистора, его сопротивление уменьшается, и он перестаёт существенно влиять на работу преобразователя.

6.2.3 Гаситель энергии торможения

Ключ К5 на Рис. 6.1 обеспечивает рассеяние энергии в случае перехода двигателя в генераторный режим, например, при торможении.

Генерируемая при этом электрическая энергия передаётся в конденсатор фильтра, диоды выпрямителя закрываются, и начинается рост U_{dc}. При достижении этим напряжением заданного порога включается ключ К5 и происходит разряд конденсатора через балластный резистор $R_{Г}$.

Выключение К5 должно происходить с некоторым гистерезисом относительно включения, иначе может начаться высокочастотный процесс переключения ключа К5, приводящий к выходу его из строя.

6.3 Мостовой трёхфазный инвертор

Структура инвертора приведена на Рис. 6.4.

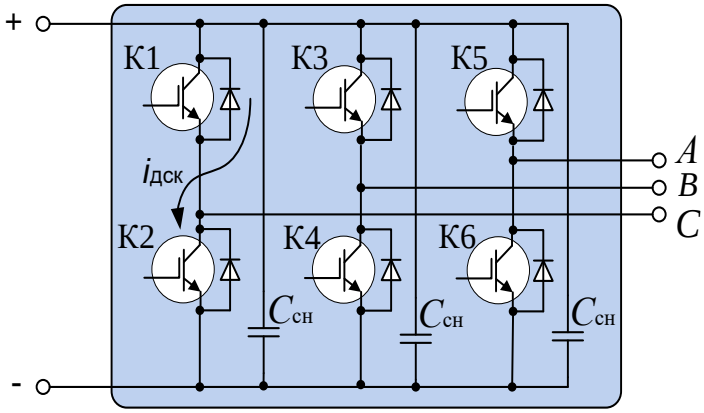


Рис. 6.4

На этом рисунке в качестве ключевых элементов изображены *IGBT*- транзисторы с обратными диодами. Эти диоды необходимы для протекания реактивных токов обмотки двигателя.

Для исключения сквозных токов в плечах инвертора при управлении ключами одного плеча (например, ключи К1, К2) используют гарантируемую временную паузу между выключением одного транзистора плеча и включением другого – «мёртвое время».

ВАЖНО!!!

- Длительность мёртвого времени влияет на величину среднего напряжения, прикладываемого к двигателю. Пусть, например, U_{dc}=300В, частота ШИМ равна 20кГц (период 50мкс), и мёртвое время составляет 1,5 мкс. При этом время, когда оба ключа стойки инвертора выключены, на периоде ШИМ составляет 3 мкс (примерно 6% от длительности периода). Таким образом, при максимальной скважности ШИМ двигателю в среднем «достанется» не более 282 В.
- На низких частотах работы привода, когда среднее напряжение, прикладываемое к обмоткам мало, мёртвое время вносит существенные нелинейные искажения в ШИМ-преобразование.

Несмотря на наличие защитных пауз, «сквозняки» инвертора не убираются окончательно. Они неминуемо возникают при включении транзистора плеча (например, ключа К2) в момент протекания реактивного тока через обратный диод оппозитного ключа (К1). Включение транзистора при этом происходит при открытом диоде. Время этого открытого состояния диода определяется его параметром, называемым «временем обратного восстановления», то есть временем, в течение которого при выключении диода восстанавливается его способность блокировать протекание обратного тока.

Данные сквозные токи ($i_{дск}$ на Рис. 6.4) вызывают помехи и дополнительные потери в инверторе. Для их минимизации обратные диоды инвертора должны иметь малое время восстановления, как минимум сравнимое с временем отпирания транзисторов инвертора.

6.3.1 Силовые ключи

Величина напряжения в звене постоянного тока (U_{dc}) и требуемая частота переключения ключей инвертора (частота ШИМ) зависят от параметров выбранного двигателя.

Оба эти фактора, величина U_{dc} и частота ШИМ, в свою очередь, определяют выбор типа силовых ключей.

В качестве полностью управляемых ключей инвертора в настоящее время применяются:

- полевые транзисторы *MOSFET*
- *IGBT* – транзисторы
- *SiC MOSFET* – полевые транзисторы на карбиде кремния.

При напряжениях в звене постоянного тока ниже 200В в инверторе удобно использовать *MOSFET*-транзисторы. Они достаточно легко управляются и обеспечивают высокие частоты переключения с малыми динамическими потерями.

ВНИМАНИЕ !!!
Топология *MOSFET*-транзисторов предполагает наличие паразитного обратного диода. При этом далеко не все транзисторы нам подходят в силу большого времени восстановления этого обратного диода. Будьте бдительны при выборе.

При более высоких напряжениях в звене постоянного тока и частотах переключения до 20 кГц, особенно при большой требуемой

мощности двигателя, вне конкуренции будут *IGBT*-транзисторы. Обратные диоды обычно встраиваются в модули таких транзисторов и уже имеют соответствующие времена обратного восстановления, но, тем не менее, проверить эти параметры стоит.

Транзисторы на карбиде кремния (*SiC MOSFET*) достаточно дороги, но они могут использоваться и на больших напряжениях, и на высоких частотах (от 20 кГц и выше). Соответствующие обратные диоды в них также уже встроены.

6.3.2 Драйверы

Драйверами в электроприводе принято называть узлы управления силовыми ключами.

Главная функция драйвера – преобразование логического сигнала управления ключом в сигнал с параметрами, необходимыми для открывания и закрывания конкретного полупроводникового прибора. Обычно для управления транзисторным ключом необходим большой, но короткий импульсный ток и двухполярное (иногда однополярное) напряжение.

При увеличении мощности и напряжения звена постоянного тока преобразователя в драйвер обычно вводят гальваническую развязку по сигналам управления и защиту управляемого полупроводникового ключа от нештатных режимов работы.

Промышленностью (в основном зарубежной) выпускается большое количество интегральных драйверов, ориентированных на различные типы полупроводниковых ключей.

Можно спроектировать и собственный драйвер. Для полноценной работы такой драйвер как минимум должен обеспечивать защиту ключа от токовой перегрузки и от снижения питания самого драйвера.

Неплохо ещё иметь защиту и от перенапряжений на ключе.

При нормальной работе правильно спроектированного преобразователя эти перенапряжения возникать не должны. Однако потенциально опасной является ситуация срабатывания защиты от токовой перегрузки транзистора. Например, при КЗ происходит аномальное нарастание тока инвертора с последующим срабатыванием защиты и выключением всех ключей. Следствием этого может быть перенапряжение в звене постоянного тока, вызванное индуктивностью силовой шины. Дабы этого избежать, в драйверах применяется быстродействующая аппаратная обратная связь по напряжению на ключах, ограничивающая перенапряжения за счёт снижения темпа запирания ключей.

Гальваническая развязка по цепям управления чаще всего осуществляется с помощью оптронов. Кроме допустимого напряжения развязки важным параметром оптрона является допустимое *dU/dt*. На него следует обращать особое внимание, иначе неминуемы муки с нечётким управлением ключами и ложным срабатыванием защит.

Рекомендуем к применению, например, оптроны *HCPL*-M456, *HCPL*-0661 и им подобные.

Реже применяется трансформаторная развязка. При правильной конструкции трансформатора она надёжнее и более помехоустойчива, но дороже и сложнее.

Питание драйвера может быть бутстрепным или независимым.

Наиболее часто бутстрепное питание применяется при напряжениях звена постоянного тока менее 200 В и использовании в качестве ключей инвертора полевых транзисторов (*MOSFET*) с однополярным управлением.

Вариант схемы бутстрепного питания драйверов стойки инвертора представлен на Рис. 6.5.

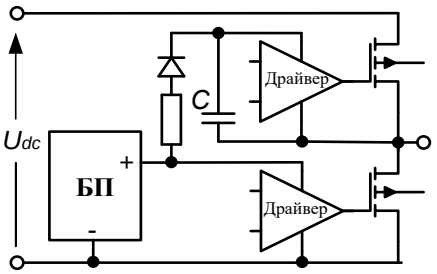


Рис. 6.5

Источником напряжения (и тока) управления верхним транзистором стойки является конденсатор (*C* на Рис. 6.5), заряжаемый от источника питания драйвера нижнего ключа (БП на Рис. 6.5) в течение времени открытого состояния нижнего транзистора.

Плюсами бутстрепного питания являются простота реализации, компактность и дешевизна, а минусом - ограничение мощности драйвера и скважности ШИМ (обычно на уровне 0.95-0.97).

С бутстрепным питанием можно использовать интегральные драйверы без гальванической развязки, например, полумостовой *IR*2113 (отечественный аналог 1308ЕУ3), трёхфазный *IR*2130 или одиночный с оптической развязкой *ACPL*3120 (отечественный аналог 249АП1Р). Этих драйверы имеют защиту только от понижения питания и сравнительно слабый токовый выход (импульсный ток до 2А).

Для управления мощными транзисторами, требующими больших импульсных токов управления, используются драйверы с независимым питанием. Гальваническая развязка питания в них осуществляется с помощью трансформаторного *DC-DC* преобразователя.

При использовании интегрального драйвера нужно руководствоваться рекомендациями изготовителя микросхемы. Обычно в спецификации на микросхему приводят схему включения, методику расчёта дополнительных элементов и даже рекомендации по правильной трассировке печатной платы, на которую драйвер устанавливается.

БОЛЕЕ ПОДРОБНО о принципах построения и защитах драйверов советуем читать в:

- учебнике Анучина А. С., «Системы управления электроприводов», Москва. Издательский дом МЭИ, 2015, раздел 3.1.
- технических описаниях (дейташитах) зарубежных интегральных драйверов.

6.3.3 Снабберные конденсаторы

Кроме ключевых элементов К1 … К6, инвертор обычно содержит ещё снабберные конденсаторы ($C_{сн}$ на Рис. 6.4).

Это высокочастотные конденсаторы небольшой ёмкости (от долей до единиц микрофарад). Их назначение – исключать перенапряжения на ключах, которые могут возникать при переключениях вследствие наличия паразитных индуктивностей силовых шин.

В качестве снабберов применяются специализированные плёночные полипропиленовые конденсаторы, например, отечественные конденсаторы серии К78-54 или импортные, *EPCOS* серии *B*32656*S*. Такие конденсаторы способны длительно работать с большими значениями реактивной мощности. Часто они изготовляются сразу с плоскими выводами для монтажа непосредственно под болты силового модуля.

Основные критерии выбора снабберов – величина ёмкости и допустимое напряжение.

В первом приближении можно считать, что на 1кВт мощности преобразователя нужны снабберы общей ёмкостью 0,1 мкФ. Это условная величина, и она должна корректироваться по результатам испытаний инвертора.

Допустимое напряжение снабберного конденсатора должно быть выше (с запасом) максимально возможного напряжения в шине постоянного тока инвертора.

6.4 Фильтры

EMC–фильтры (фильтры электромагнитной совместимости) используются для подавления электромагнитных помех, источником которых является ШИМ–инвертор преобразователя частоты.

Чаще всего они представляют из себя комбинацию индуктивно-резистивно-ёмкостных цепей. Устанавливаются они обычно между силовой сетью и выпрямителем.

Промышленностью, родной и зарубежной, выпускается довольно большое количество таких фильтров.

При выборе готового EMC-фильтра стоит обращать внимание на его основные характеристики: действующее значение тока нагрузки, частоту переменного напряжения, рабочее напряжение и напряжение пробоя.

Проектирование собственного фильтра помех — это отдельная и сложная задача, однако иногда это единственно возможное решение, например, когда покупной фильтр не подходит по габаритам.

Синусные фильтры устанавливаются на выход инвертора преобразователя. Они режут высокие гармоники ШИМ-напряжения и позволяют получать из него некое подобие синуса.

Особенно они важны там, где применяются длинные линии между преобразователем и двигателем (от сотен метров). В таких линиях высокие гармоники ШИМ-напряжения могут вызывать высокочастотные колебания, приводящие к недопустимым перенапряжениям на обмотке двигателя. Типовая структура синусного фильтра и вид линейных напряжений на его входе и выходе изображены на Рис. 6.6.

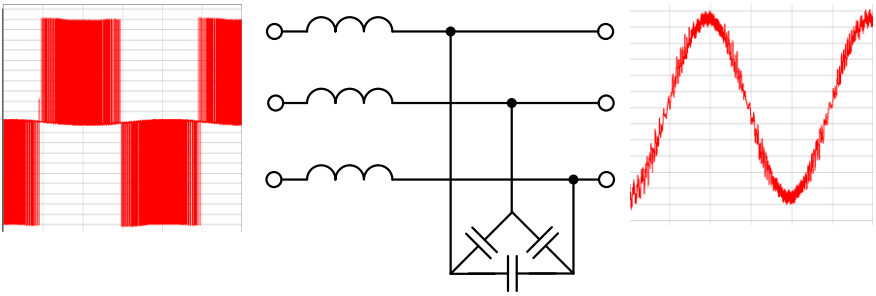


Рис. 6.6

При выборе параметров таких фильтров удобно использовать моделирование.

6.5 Датчики

В электроприводах используются датчики следующих величин:

- тока (фаз двигателя, звена постоянного тока)
- напряжения (силовой сети, звена постоянного тока)
- температуры
- скорости и положения.

БОЛЕЕ ПОДРОБНО о применяемых датчиках написано в книгах:

- Ю. Н. Калачёв, Д. В. Самохвалов, Основы регулируемого электропривода (Антиучебник), ДМК Пресс, раздел 8.
- Анучин А. С., Системы управления электроприводов, Москва, Издательский дом МЭИ, 2015, разделы 3.2 и 3.4.

7 Конструкция преобразователя

Неправильный подход к конструкции преобразователя может привести к полной неработоспособности привода.

7.1 Силовой блок

Конструкция силовых проводников блока должна минимизировать возникающие при работе преобразователя помехи, вызванные высокочастотными переходными процессами в силовой схеме.

Колебания в силовых шинах возникают при переключении ключей инвертора. Их причиной является паразитная индуктивность этих шин ($L_{Ш}$ на Рис. 7.1), через которую большой фильтрующий конденсатор звена постоянного тока (C_{dc} на Рис. 7.1) соединяется со снабберными конденсаторами небольшой ёмкости ($C_{СН}$ на Рис. 7.1).

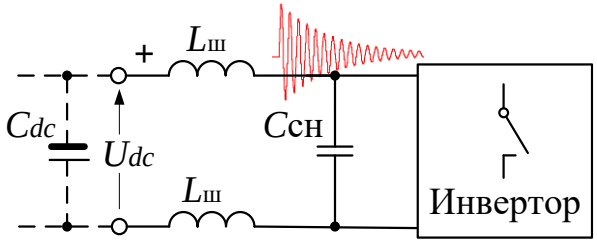


Рис. 7.1

При этом конденсатор фильтра (C_{dc}), в силу его большой ёмкости по отношению к снабберам ($C_{\text{СН}}$), можно рассматривать как источник напряжения (U_{dc} на Рис. 7.1).

При переключении ключей инвертора в силовых шинах возникают колебания с резонансной частотой образующегося паразитного *CL*-контура. Эти колебания имеют высокую частоту, слабо затухают и могут достигать достаточно больших амплитуд. Они вызывают перенапряжения в силовых шинах и тепловые потери в элементах силового блока. Особенно это не нравится снабберам, которые от обиды надуваются и могут выходить из строя. Ну а далее начинаются неприятности уже с транзисторами инвертора…

Избежать колебаний и вызванных ими проблем помогает правильная конструкция силовых шин, так называемый сэндвич (Рис. 7.2).

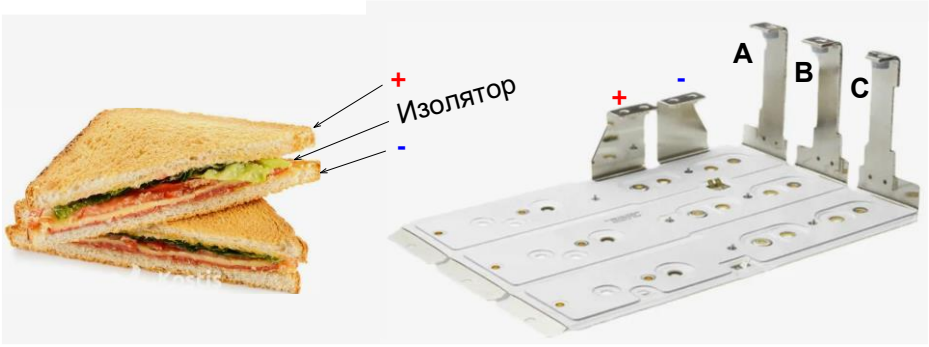


Рис. 7.2

Плюсовая и минусовая шины делаются максимально большой площади, располагаются непосредственно друг над другом и разделяются тонким слоем диэлектрика. В такой конструкции распределённая емкость, возникающая между шинами, эффективно компенсирует их весьма небольшую индуктивность – и колебаний не возникает. В некоторых случаях при этом можно даже обойтись без снабберов.

При этом важно правильно размещать конденсаторы силовой шины. Они должны располагаться как можно ближе к ключам инвертора.

Эффект колебаний выражен тем ярче, чем больше мощность (ток) инвертора. На мощностях свыше 7-10 кВт конструкция типа «сэндвич» строго рекомендуется. Однако и на меньших мощностях, по возможности, шины лучше делать «сэндвичами», при этом можно использовать в качестве силовых шин слои печатной платы.

Ниже на Рис. 7.3 в качестве примера приведён вариант правильной конструкции силового блока преобразователя (по материалам компании «Электротекс»).

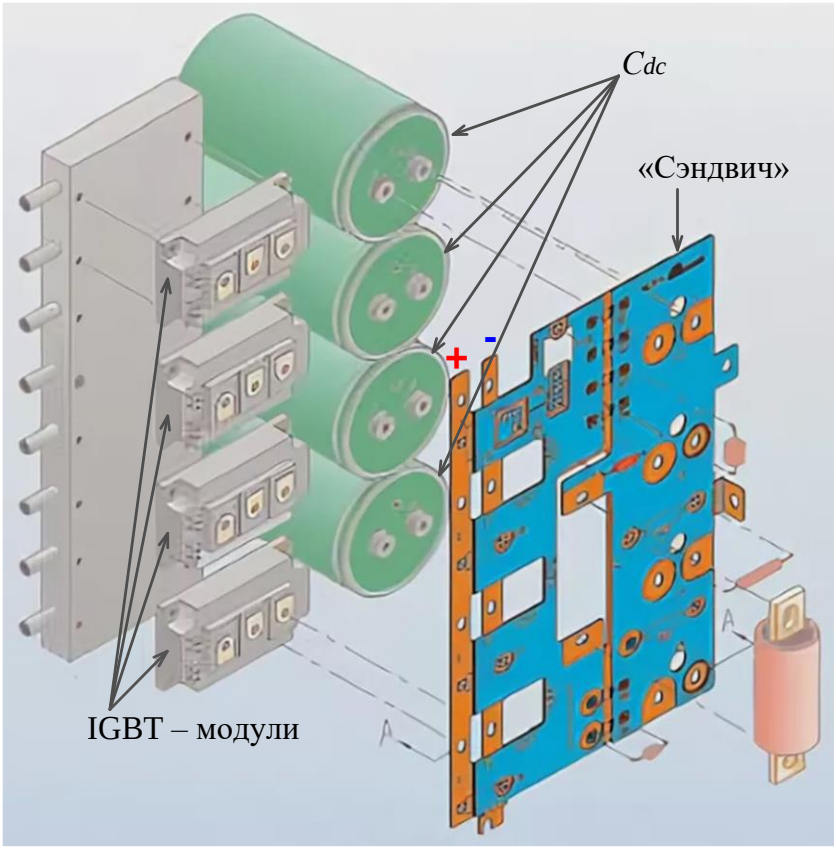


Рис. 7.3

Рис. 7.4

Несколько слов стоит сказать и о размещении драйверов.

Во избежание помех драйверы в идеале надо устанавливать непосредственно на силовые модули ключей, обеспечивая тем самым минимальную длину соединений.

На Рис. 7.4 показан вариант правильной установки двухканального драйвера на модуль силовой стойки *IGBT*-ключей.

7.2 Система управления

Не стоит устанавливать плату/платы системы управления вплотную к силовым шинам и ключам. Хорошо отделить её от силового блока экраном.

Разводка платы/плат СУ – это вопрос, который требует отдельного рассмотрения (не в этой книге).

РЕКОМЕНДУЕМ ЧИТАТЬ в интернете книгу Семёна Тютюкова «Практические рекомендации по разработке печатных плат».

7.3 Раскладка проводов

Провода можно разделить на три группы.

1. **Силовые:** питающая силовая сеть, выходы фаз двигателя.

   Их надо по максимуму удалять от всех остальных проводов, полезно тащить их в заземлённых экранах, например, по правой стенке шкафа (если, конечно, преобразователь находится в шкафу).

2. **Несиловые:** низковольтное питание узлов и плат, различные логические команды и последовательные интерфейсы.

   Эти провода можно проложить, например, по противоположной, левой стенке шкафа в жгуте. Пересечения с силовыми проводами надо исключать.

   Провода последовательных интерфейсов тяните в рекомендуемых для этих интерфейсов кабелях. Не злоупотребляйте плоским (ленточным) кабелем.

3. **Провода аналоговых сигналов**

   Пожалуйста, отнеситесь к ним наиболее внимательно. По возможности их укоротите и удалите от всех остальных проводов. Иногда требуется витая пара в экране.

7.4 Три совета начинающему конструктору (*наболело*)

1. Слушайте опытного дядю.
2. Если дядя не прав, смотрите совет 1.
3. Возьмите за правило собирать первый экземпляр спроектированного вами изделия собственными руками. Ибо вскоре сборщик перестанет материться, и вы возрадуетесь.

Если изделие работает и сборщик не матерится,
то Вы – опытный дядя.

8 Структура системы управления

Типовая структура системы управления преобразователя (далее СУ) показана на Рис. 8.1.

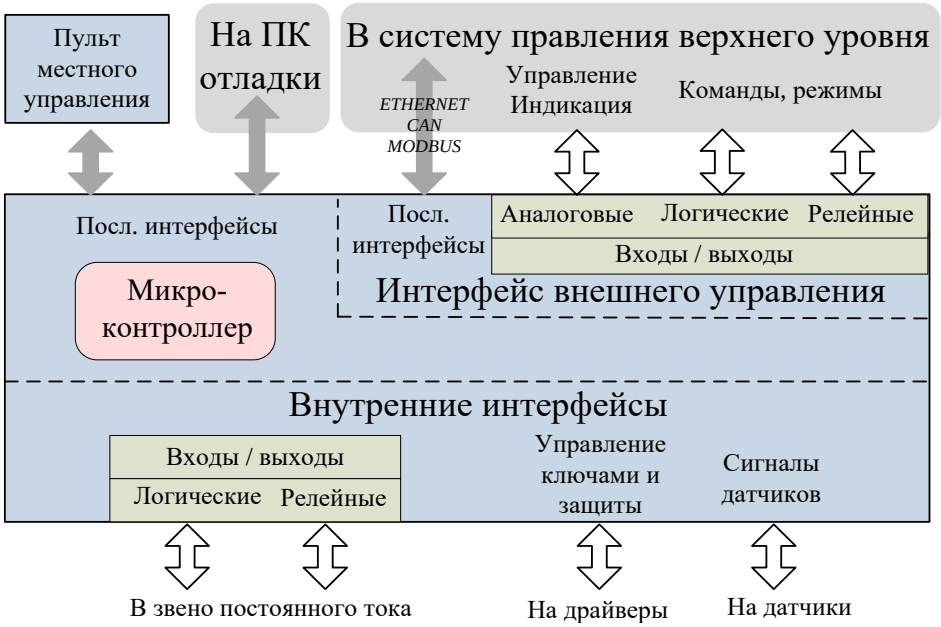


Рис. 8.1

Интерфейсы СУ разделяются на внешние и внутренние.

> **Интерфе́йс** (от англ. *interface*) или **стык** — граница между двумя функциональными объектами.

Внешние интерфейсы обеспечивают обмен с системой управления верхнего уровня. Обычно это последовательный интерфейс, по которому осуществляется управление приводом от какого-то внешнего устройства.

Часто существует ещё некоторый набор каналов типа «ДА/НЕТ» и аналоговых каналов, по которым осуществляется информационный обмен с вышестоящей СУ.

Конкретная структура внешних интерфейсов системы управления определяется требованиями ТЗ. Там указываются необходимые последовательные интерфейсы (если они есть), например, *CAN*, *Ethernet* и т.д. и приводятся их протоколы обмена.

Также ТЗ определяет количество, формат и смысл дискретных (ДА/НЕТ) и аналоговых каналов обмена с устройством верхнего уровня.

К внутренним интерфейсам относятся команды и сигналы управления инвертором и другими элементами звена постоянного тока, а также сигналы связи с датчиками (тока, напряжения, температуры, скорости, положения).

Конкретный вид внутренних интерфейсов СУ зависит от состава силового блока и определяется разработчиком.

Кроме того, часто преобразователь имеет пульт местного управления, который связывается с СУ по последовательному интерфейсному каналу.

Удобно также иметь ещё один технологический последовательный канал для подключения СУ к внешнему компьютеру (ноутбуку). Через него можно осуществлять программирование, параметрирование и индикацию различных рабочих величин в процессе отладки и настройки привода.

ПРИМЕЧАНИЕ!!!

- Для внешних интерфейсов необходима гальваническая развязка.
- Сигналы внутренних интерфейсов также развязываются гальванически. Исключения могут составлять маломощные низковольтные преобразователи, в которых общая точка платы системы управления связана с отрицательным полюсом звена постоянного тока (такое тоже бывает).

БОЛЕЕ ПОДРОБНО об аналоговых интерфейсах с датчиками тока и напряжения читайте в учебнике: Анучин А. С., Системы управления электроприводов, Москва, издательский дом МЭИ, 2015, 373 с., раздел 3.3.

Сердцем СУ является микроконтроллер — однокристальный компьютер, который объединяет в одной интегральной микросхеме микропроцессор, память и устройства ввода-вывода.

9 Выбор микроконтроллера

9.1.1 Требования к архитектуре микроконтроллера

Тип применяемого микроконтроллера во многом определяется степенью сложности задачи, которую предстоит решить, а также количеством и типом необходимых датчиков и интерфейсов.

Количества каналов АЦП должно хватать для связи со всеми аналоговыми датчиками. Если в силовом инверторе два токовых датчика, лучше подбирать контроллер с минимум двумя блоками АЦП, чтобы иметь возможность считывать значения тока в фазах одновременно, без задержки между измерениями.

Важным является наличие на борту требуемых интерфейсов в нужном количестве. Если *UART* и *SPI* есть на всех микроконтроллерах, то специфические интерфейсы, такие как *CAN*, *Ethernet* встречаются уже реже. А иногда нужны и совсем экзотические – типа *ARINC* или МКИО (*MILSTD*1533).

Также нужно не забывать про датчики скорости и/или положения, они тоже подключаются к микроконтроллеру по цифровому или аналоговому интерфейсу.

Для реализации ШИМ микроконтроллер обязательно должен иметь хотя бы один специализированный таймер с блоком генерации защитных пауз инвертора.

Удобно, когда в микроконтроллере есть хотя бы один ЦАП – его можно использовать для вывода на осциллограф данных отладки в режиме реального времени.

ОБ АРХИТЕКТУРЕ СИСТЕМЫ УПРАВЛЕНИЯ читайте также в учебнике: Анучин А. С., Системы управления электроприводов, Москва, издательский дом МЭИ, 2015, 373 с., раздел 3.1.

9.1.2 Программные циклы контуров регулирования

СУ электропривода строится по принципам подчинённого регулирования и состоит из нескольких вложенных контуров регулирования. В максимальной конфигурации это:

- контур тока (управление моментом)
- контур регулирования скорости
- контур регулирования положения.

БОЛЕЕ ПОДРОБНО – в книжке Ю. Н. Калачёва и Д. В. Самохвалова «Основы регулируемого электропривода (Антиучебник)» читайте о:

1. подчинённом регулировании – раздел 3.3
2. структурах СУ различных двигателей - разделы 6.1.9 … 6.1.11, 6.2.3, 6.5.4 … 6.5.7, 7.7.

Самый короткий программный цикл имеет внутренний контур тока, он, как правило, считается с частотой ШИМ, которую, в свою очередь, выбирают, исходя из допустимых пульсаций тока статора.

Для 50-герцовых двигателей частота ШИМ обычно лежит в диапазоне 3-7 кГц. Высокая частота ШИМ (до 50 кГц) применяется для управления высокочастотными электродвигателями с относительно малой электрической постоянной времени статора.

ВАЖНО помнить, что за повышение частоты ШИМ приходится всегда расплачиваться увеличением динамических потерь в ключах инвертора.

Программа контура тока начинается с опроса датчиков. Выбор момента этого опроса достаточно важен.

Если задать размах треугольника опоры ШИМ, превышающим размах модулируемых сигналов, то в области вершин треугольника всегда будет существовать временной промежуток, в котором не будет создающих помех коммутаций ключей инвертора. С целью исключения этих помех в сигналах датчиков, запуск программы контура тока (опрос АЦП) осуществляют в бескоммутационной паузе, в районе максимума (или минимума) опорного треугольника ШИМ (см. Рис. 9.1).

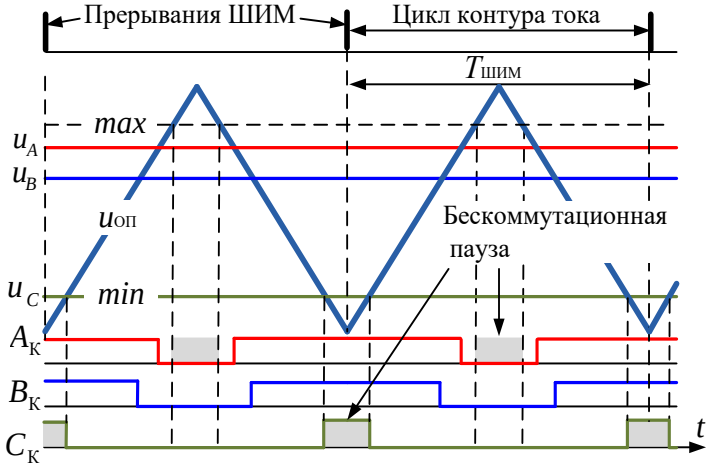


Рис. 9.1

БОЛЕЕ ПОДРОБНО о минимизации помех при опросе датчиков читайте в книге:

Ю. Н. Калачёв, Д. В. Самохвалов, Основы регулируемого электропривода (Антиучебник), ДМК Пресс, раздел 9.1.

Контуры управления скоростью и положением имеет смысл считать на более низкой, чем контур тока, частоте, так как механическая постоянная времени двигателя существенно (минимум в десять раз) больше электрической. Да и микроконтроллер это разгружает.

Кроме того, при вычислении скорости по сигналу дискретного датчика положения (инкрементного или *n*-разрядного абсолютного) возникнет помеха, вызванная дифференцированием дискретного сигнала. При этом слишком малое время цикла программы скоростного контура (времени дифференцирования) может оказаться вредным вследствие высокого уровня этой помехи, а некоторое увеличение времени цикла будет эквивалентно её фильтрации.

Также в сигнале датчика положения неминуемо присутствуют и другие помехи, и всё вышесказанное применимо и к ним.

Как правило, один цикл расчёта внешнего контура соответствует нескольким циклам расчёта внутреннего.

9.1.3 Прерывания

ПРЕРЫВАНИЕ – это вызов определённой подпрограммы, генерируемый аппаратной частью микроконтроллера. В результате прерывания выполнение текущей программы останавливается и происходит переход к выполнению вызываемой подпрограммы.
Если в одной программе несколько прерываний, то порядок их исполнения в случае одновременного вызова определяется настройкой уровня их приоритетов.

Обычно расчёт цикла контура тока запускается по прерыванию, которое генерируется таймером микроконтроллера в районе вершины опорного треугольника ШИМ (см. Рис. 9.1).

Такой вызов программы контура тока решает сразу несколько задач:

- позволяет синхронизировать момент измерения токов с нужным моментом времени в пределах периода ШИМ (по вершине опорного треугольника ШИМ, см. раздела 9.1.2)

- жёстко фиксирует период вызова цикла расчёта контура тока, к которому привязаны коэффициенты интегрирования его ПИ регуляторов
- позволяет использовать эти же прерывания для инкрементирования программного таймера, вызывающего программу внешнего (медленного) контура управления, который считается один раз за несколько циклов счёта контура тока.

С помощью прерываний также удобно организовывать обмен через различные интерфейсы с внешним миром.

Если в программе используется более одного прерывания, наивысший приоритет следует устанавливать прерыванию по таймеру ШИМ.

9.1.4 Структура цикла программы управления

Программа контура тока состоит из следующих этапов:

1. опрос датчиков (тока, положения ротора и т.д.)
2. выполнение алгоритма регулирования тока
3. выдача управляющих воздействий на ШИМ-генератор.

Внешние, по отношению к контуру тока, контуры скорости и положения считаются один раз за несколько циклов контура тока. Их этапы аналогичны:

1. опрос датчика (скорости/положения)
2. выполнение алгоритма регулирования
3. выдача сигнала задания на подчинённый контур.

Помимо выполнения программ контуров регулирования, микроконтроллер выполняет ещё ряд сервисных функций.

Таких как:

- общение через последовательный интерфейс с системой управления верхнего уровня
- опрос и выдача дискретных сигналов внешнего интерфейса
- реализация алгоритмов различных защит и т.д.

Программы контуров скорости/положения и сервисных функций можно выполнять в фоновом режиме, конечно, если в контуре тока имеется некоторый запас времени. То есть время, необходимое для расчётов контура тока, меньше периода ШИМ. Этот резерв времени и используется для фоновых расчётов.

На Рис. 9.2 приведён пример варианта структуры программы управления, иллюстрирующий вышесказанное.

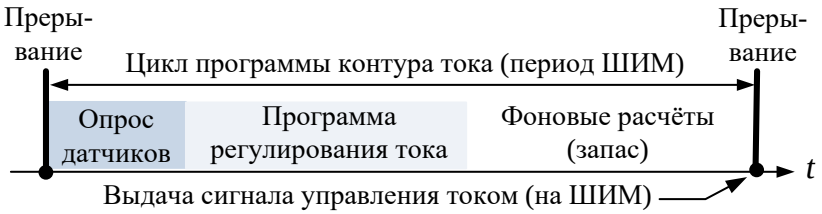


Рис. 9.2

Допустим, контур тока привода работает с периодом 50 мкс (частота ШИМ 20кГц).

Прежде чем начать счёт регулирующей части программы контура, необходимо получить новые значения токов фаз, дождавшись окончания АЦП-преобразований сигналов всех аналоговых датчиков. В среднем одно преобразование на встроенном АЦП занимает, в зависимости от типа микроконтроллера, от 1 до 5 мкс.

Время опроса датчика положения ротора, работающего по последовательному интерфейсу, может занимать до 10 мкс (например, датчик с *SPI* на частоте 2 МГц перешлёт 2 байта данных за ≈ 8 мкс).

Если запустить АЦП-преобразования и опрос датчика положения одновременно, то можно сэкономить немного времени.

Пусть опрос всех датчиков отнимет у нас 8 мкс, а 15 мкс (30% от времени цикла) необходимо зарезервировать на фоновый расчёт алгоритмов регулирования контуров скорости/положения, обмен данными с внешним окружением и другие сервисные функции привода.

Тогда для счёта собственно программы контура тока останется 27 мкс.

Хорошо, если программа контура тока содержит только анализ положения ротора по датчику (ДПР) и программу расчёта одного ПИ-регулятора тока.

А уложится ли микроконтроллер в нужный цикл, если, например, необходимо векторное регулирование скорости с преобразованием координат, компенсацией перекрёстных связей, да ещё с переходом между зонами постоянства момента и постоянства мощности?

Всё зависит от его производительности.

ДОПОЛНИТЕЛЬНО ЧИТАЙТЕ о структуре программы управления в учебнике: Анучин А. С., Системы управления электроприводов, Москва. Издательский дом МЭИ, 2015, раздел 3.5.

9.1.5 Рекомендации по выбору микроконтроллера

Итак, микроконтроллер, по возможности, должен удовлетворять требованиям раздела 9.1.1 и, самое главное, успевать считать программу с нужным циклом.

Для реализации управления шаговым двигателем или синхронным двигателем без использования векторного управления обычно достаточно любого современного 8-, 16- и тем более 32-битного микроконтроллера.

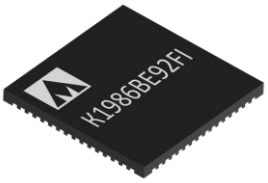
Часто для построения СУ в современных электроприводах (в том числе с векторным управлением) используются 32-битные микроконтроллеры с архитектурой *ARM M3*, *M4F* или *RISC-V*.

Конечно, хорошо применять специализированные микроконтроллеры для электропривода, которые имеют на борту:

- сопроцессор, выполняющий операции с вещественными числами с плавающей точкой (*FPU*)
- аппаратный блок вычисления координатных преобразований.

Но иногда выбора нет (например, при требовании ТЗ использовать отечественную элементную базу), и приходится применять простой универсальный микроконтроллер без *FPU*. В этом случае можно использовать целочисленный способ вычислений, приводя все числа к формату с фиксированной точкой (*fixpoint*).

ВЫЧИСЛЕНИЯ *fixpoint* — быстрее и менее ресурсоёмки, чем операции с плавающей точкой. При этом дробные числа (*float*) приводятся к целым с фиксированной точкой умножением на коэффициент К, например, на 1000, а дробный остаток отбрасывается. Однако программисты при этом должны бдеть, так как в *fixpoint* возникает опасность переполнения разрядной сетки и потери точности.



По опыту авторов данных строк, реализовать векторное управление с использованием *fixpoint* вычислений на базе современного отечественного микроконтроллера (например, 1986BE9x) не так уж и сложно, при этом рабочая частота контура тока может достигать 20 кГц.

СМОТРИТЕ МИКРОКОНТРОЛЛЕРЫ

- западные *STM*32*F*103, *STM*32F407, *TMS320F28335*
- китайские *NS*32*F*103*VBT*6, *GD*32*F*427*ZGT*6
- отечественные К1986ВЕ9х и К1948ВК018 «АМУР».

10 Синтез модели привода

ОБЩИЕ СВЕДЕНИЯ о *SimInTech*, необходимые для понимания дальнейшего материала, можно почерпнуть в книге: Б. А. Карташов, Е. А. Шабаев, О.С. Козлов, А. М. Щекатуров, «Среда динамического моделирования технических систем», разделы 2.2 … 2.6.

10.1 Стандартная библиотека «Электропривод»

10.1.1 Модели двигателей и механической передачи

При моделировании любого объекта надо учитывать только те его свойства, которые будут существенны в данном конкретном случае, иначе можно погрязнуть в мелочах, которые будут только путать и увеличивать время счёта модели.

Этот принцип в полной мере относится к моделированию различных электродвигателей.

Например, фазную катушку СДПМ можно представить как комбинацию индуктивности, активного сопротивления и ЭДС. Однако активные сопротивления катушек зависят от температуры, а с индуктивностями и потоками вообще всё непросто!

Фазные индуктивности, взаимные индуктивности фаз и потокосцепления магнита ротора с катушками зависят от положения ротора, причём все упомянутые выше зависимости, в свою очередь, зависят от степени насыщения машины. Другими словами, для каждого сочетания составляющих тока статора в системе dq (i_{Sd}, i_{Sq}) существуют свои зависимости всех индуктивностей и потокосцеплений от угла поворота ротора. А ведь есть ещё потери в стали и зубцовый момент …

Как ни странно, все эти особенности можно учесть в модели – но нужно ли?

В разделе электропривод среды *SimInTech* существует ряд упрощённых моделей двигателей различных типов (синхронный, асинхронный, шаговый). Упрощение заключается в том, что параметры статорной обмотки и ротора двигателя считаются постоянными, не учитываются потери в стали и зубцовый момент (для СДПМ).

Это модели первого приближения. Ими можно пользоваться на этапе предварительного проектирования, когда привода в «железе» ещё нет.

А когда преобразователь и двигатель появятся, можно перейти ко второму этапу – уточнению модели двигателя.

В качестве примера рассмотрим привод с СДПМ. В учебниках описаны СДПМ либо с синусоидальной, либо с трапецеидальной формой ЭДС. Для упрощённой модели можно считать, что это так и есть.

В действительности же форма ЭДС двигателя может существенно отличаться и от синуса, и от трапеции. Для примера на Рис. 10.1 приведена снятая осциллографом фазная ЭДС реального двигателя.

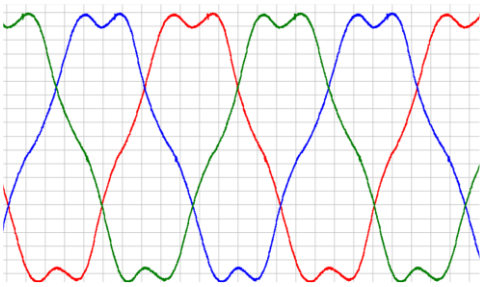


Рис. 10.1

Современные осциллографы позволяют запомнить осциллограмму в файл *.csv. Затем этот файл можно использовать в модели двигателя (в *SimInTech* такая возможность есть).

На готовом двигателе можно измерить и другие параметры: индуктивности фаз и взаимные фазные индуктивности, фазное сопротивление, величину и форму зубцового момента. С учётом этих измерений можно уточнить параметры и структуру модели.

Для проверки такой уточнённой модели можно провести эксперименты, заставив двигатель работать в некоторых тестовых режимах. Затем снимаемые при этом рабочие параметры сравниваются с параметрами моделирования тех же режимов. Далее можно «подрихтовать» модель и подогнать её под эксперимент. Это даёт, как правило, уже хорошее совпадение поведения модели с реальностью.

Как пример на Рис. 10.2, приведены осциллограммы тока фазы двигателя с вышеприведённой ЭДС, работающего в тестовых режимах рядом с графиками работающей в тех же режимах уточнённой модели.

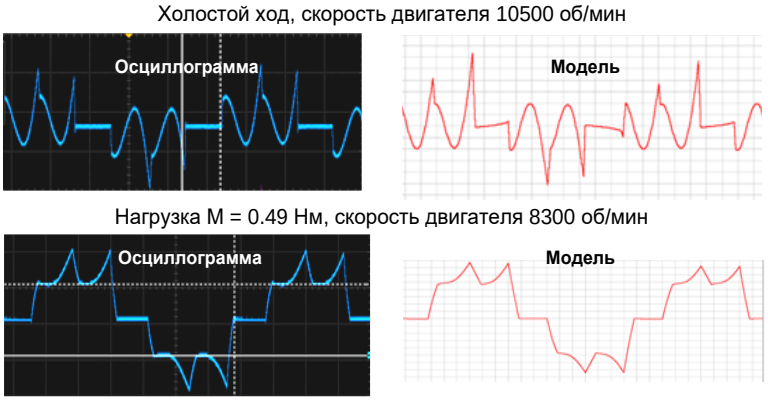


Рис. 10.2

А можно, конечно, если это нужно, смоделировать двигатель и ещё точнее.

Наиболее точно все режимы работы двигателя считаются программами типа *Ansys Maxwel*. Это программы для 3*D*-моделирования магнитных полей методом конечных элементов. Они применяются для проектирования двигателей, но при моделировании привода неудобны в силу своей громоздкости.

Такие программы могут формировать файлы зависимостей параметров двигателя от электрического угла и степени насыщения машины.

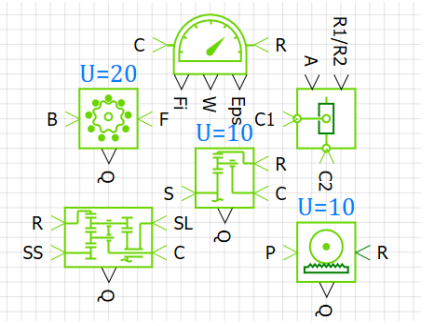
В *SimInTech* существует модель СДПМ, которая может работать с файлами параметров, сформированными программой 3*D*-моделирования полей *EasiMotor FEM*. В результате получается точная и достаточно быстрая одномерная модель, дающая тот же результат, что и «тяжёлый» и долгий 3*D*-расчёт методом конечных элементов.

ПРИМЕЧАНИЕ

- Описание моделей двигателей различных типов можно найти в книжке: Ю. Н. Калачёв, Д. В. Самохвалов, Основы регулируемого электропривода (Антиучебник), ДМК Пресс, разделы 3, 6, 7.
- Не все существующие в *SimInTech* модели двигателей выложены в разделе «Электропривод» среды. Кроме того, раздел постоянно обновляется. Для получения консультаций по моделям рекомендуем связываться с разработчиками среды.

Для моделирования механической передачи используются элементы раздела «Механика» среды *SimInTech*.

Стадии моделирования те же, что и у модели двигателя:



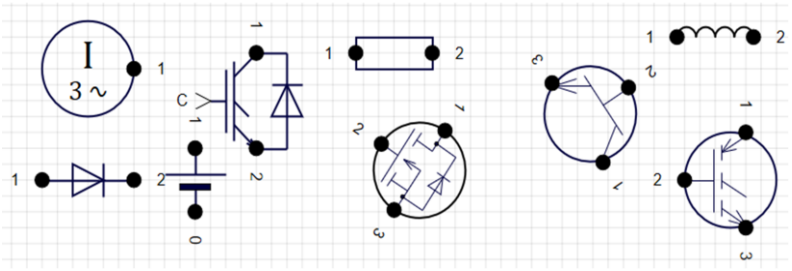
- упрощённая модель, основанная на расчётных данных
- уточнённая модель, основанная на измерениях
- уточнение параметров модели механики при сравнении результатов моделирования и работы в тестовых режимах.

10.1.2 Элементы силовой схемы

Модель силовой схемы привода строится на основе элементов раздела «Электрические цепи - Динамика».

Там можно найти модели:

- различных полупроводниковых приборов: диодов, транзисторов, тиристоров
- пассивных *R-L-C* элементов
- источников тока и напряжения
- и т.д.



А для удобства пользователя в разделе «Электропривод» можно найти уже скомпонованный транзисторный трёхфазный мостовой инвертор с обратными диодами. ВАХ транзисторов и диодов инвертора можно настроить с помощью свойств этих элементов.

Заметим, что в данной модели открывание/закрывание транзисторов и диодов происходит мгновенно. При моделировании привода в целом эта идеализация вполне оправдана.

Переходные процессы включения/выключения полупроводниковых элементов имеет смысл моделировать, только если цель моделирования – исследование самих этих процессов. В *SimInTech* существуют модели диодов и транзисторов, которые для этого можно использовать. Однако при этом вследствие очень коротких фронтов переключения таких ключей время интегрирования модели (квантование по времени) должно быть очень маленьким (наносекунды).

Если модель всего электропривода будет работать с таким шагом, мы недопустимо затянем время её счёта, и исследование, например, параметров движения рабочего органа электропривода, будет связано с долгим и мучительным ожиданием.

ПРИМЕЧАНИЕ

Задачи с существенно разными временными характеристиками при моделировании вообще желательно не смешивать.

10.1.3 Элементы системы управления

При построении структуры системы управления электропривода используются принципы подчинённого регулирования.

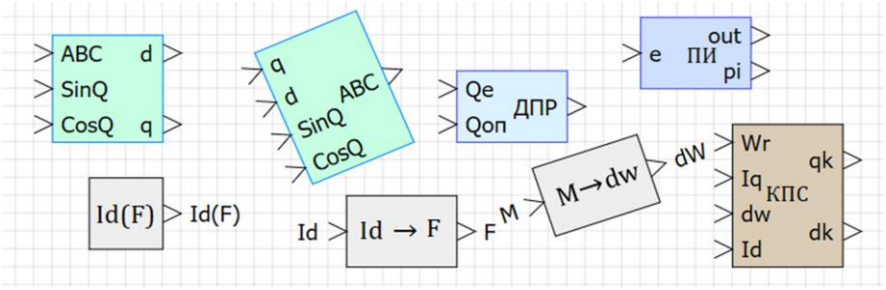
Существует ограниченное число вариантов таких структур.

БОЛЕЕ ПОДРОБНО в книжке Ю. Н. Калачёва и Д. В. Самохвалова «Основы регулируемого электропривода (Антиучебник)» читайте о:

1. структурах СУ синхронных электроприводов в разделах 6.1.8 ...6.1.11 и 6.2.3
2. структурах СУ асинхронных приводов в разделе 7.7.

В свою очередь, в этих структурах можно выделить типовые блоки, из которых они состоят. Модели таких типовых блоков и содержатся в разделе «Электропривод» среды *SimInTech*.

Это ПИ-регуляторы, преобразователи координат, узлы ШИМ, блоки компенсации перекрёстных связей для синхронного и асинхронного двигателя, блоки вычислители момента и потока, модель ротора асинхронного двигателя, ограничители *dq*-напряжений и т.д.



ВНИМАНИЕ

- Для лучшего понимания схем моделей, рассматриваемых далее, рекомендуем просмотреть книжку Ю. Н. Калачёва «*SimInTech* – моделирование в электроприводе», ДМК Пресс, Москва 2021. В ней описаны элементы раздела «Электропривод».
- Также можно читать справку по элементам раздела «Электропривод» в *SimInTech*.

10.2 Структура модели

Микроконтроллер системы управления получает информацию о состоянии привода, считывая значения с датчиков.

Затем в течение некоторого времени (программного цикла) осуществляются необходимые вычисления, и в конце цикла микроконтроллер выдаёт управляющее воздействие на исполнительную часть электропривода, которая в общем случае состоит из силового преобразователя, двигателя, механической передачи и рабочего органа.

Таким образом, выдача управляющего воздействия осуществляется в определённые моменты времени с задержкой относительно считывания состояния электропривода, равной программному циклу.

В цифровых системах этот эффект называется квантованием по времени.

Кроме этого, существует ещё и квантование по уровню, определяющееся разрядность микроконтроллера и его АЦП.

Эффекты квантования можно строго учесть при моделировании в *SimInTech*, разбив модель привода на две части: модель системы управления (управляющего алгоритма) и модель исполнительной части (всё остальное).

Проекты синхронизируются и объединяются в единый пакет модели привода с помощью встроенной в *SimInTech* базы данных сигналов (См. Рис. 10.3)

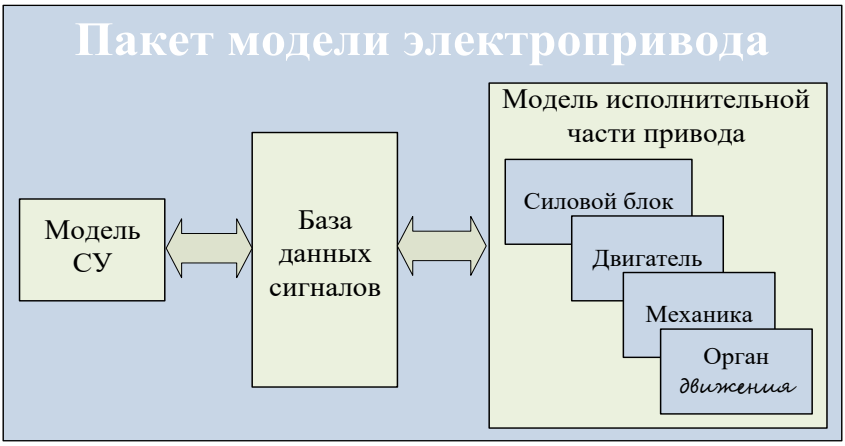


Рис. 10.3

ПОДРОБНО О БАЗЕ ДАННЫХ СИГНАЛОВ читайте в справочной системе *SimInTech*.

10.2.1 Модель исполнительной части привода

Данная модель должна адекватно отражать происходящие в исполнительной части привода процессы, в том числе и самые быстрые. А таковыми в ней являются электрические процессы.

Соответственно время интегрирования модели должно быть существенно меньше, чем время самого быстрого моделируемого переходного электрического процесса. Учитывая то, что динамику процессов переключения силовых ключей мы не моделируем (см. раздел 10.1.2), самым быстрым, как правило, является процесс нарастания тока в обмотке статора двигателя. На него и стоит ориентироваться.

Для обычных 50-герцовых двигателей, как правило, хватает времени интегрирования 1мкс. Для высокочастотных двигателей это время надо уменьшать.

Кроме того, время интегрирования должно быть:

- не больше длительности самого короткого управляющего сигнала, например, защитной паузы инвертора (см. раздел 6.3)
- существенно меньше времени интегрирования проекта модели (моделей) цифровой СУ (о ней смотрите далее).

10.2.2 Модель СУ

Как уже упоминалось, СУ электропривода строится по принципам подчинённого регулирования и состоит из нескольких вложенных контуров регулирования. В максимальной конфигурации это:

- контур тока (управление моментом)
- контур регулирования скорости
- контур регулирования положения.

При моделировании контуров регулирования в параметрах расчёта соответствующего проекта надо выбирать метод интегрирования Эйлера и время интегрирования, равное времени программного цикла моделируемого контура.

При этом, как и в реальном приводе, информационный обмен проекта контура с моделью исполнительной части и все расчёты в нём будут осуществляться один раз за такт интегрирования, и управляющее воздействие будет задерживаться относительно считывания сигналов датчиков на время интегрирования, равное времени программного цикла.

Соответственно, если контуры СУ имеют разное время программных циклов, то они и моделируются в различных проектах пакета модели, имеющих различное время интегрирования.

Как уже говорилось, синхронизация работы этих проектов в пакете модели электропривода осуществляется с помощью встроенной в *SimInTech* базы данных сигналов.

10.3 Пример модели привода

Ниже в качестве примера рассмотрим модель реально разработанного и изготовленного синхронного электропривода руля высоты беспилотного летательного аппарата.

Пакет модели состоит из двух проектов: модели цифровой системы управления и модели исполнительной части привода.

Сначала рассмотрим **<u>модель исполнительной части</u>.** Она изображена на Рис. 10.4.

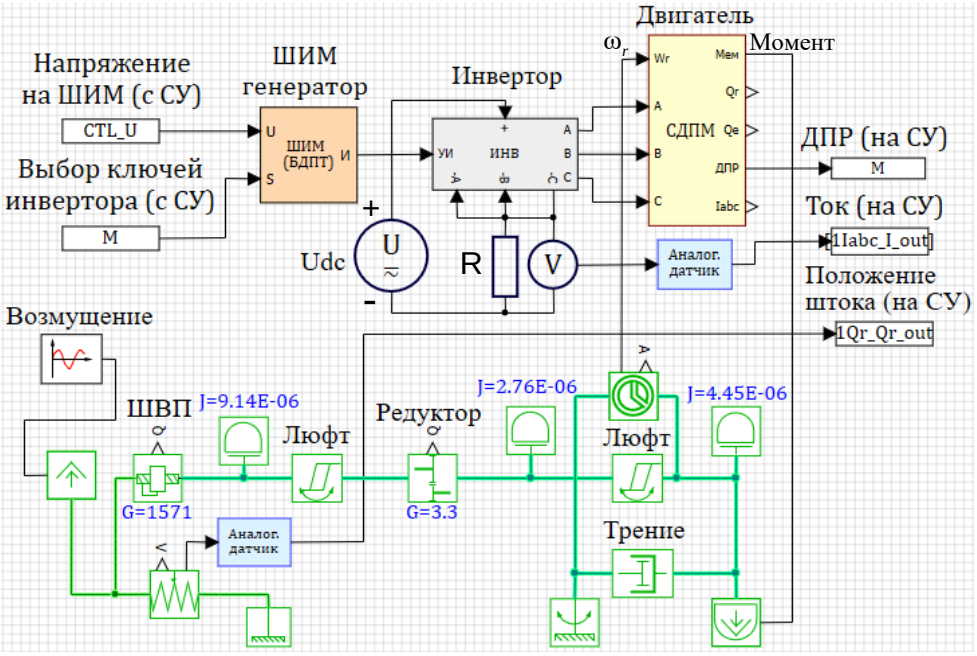


Рис. 10.4

Используемая в проекте модель СДПМ уточнена с помощью введения измеренных формы ЭДС и зубцового момента реального двигателя. Кроме того, параллельно фазным катушкам в модель введены зависящие от частоты активные сопротивления, учитывающие потери на намагничивание.

В модель двигателя также встроен датчик, определяющий положения ротора с точностью до 60° (ДПР).

БОЛЕЕ ПОДРОБНО в книжке Ю. Н. Калачёва и Д. В. Самохвалова «Основы регулируемого электропривода (Антиучебник)» читайте о:

- магнитных потерях СДПМ в разделе 6.4
- ДПР в разделе 6.2.3.

Моделируемый привод является низковольтным ($U_{dc} = 27$ В) и маломощным. В таких приводах в качестве датчика тока можно использовать резистор, включённый в минусовую шину инвертора. При этом общую точку СУ привода соединяют с минусовой шиной напряжения U_{dc}, что позволяет не использовать гальваническую развязку при вводе сигнала тока через АЦП в процессор СУ.

Соответственно, на Рис. 10.4 сигнал с шунта (R) масштабируется, фильтруется и подаётся на выход модели (сигнал «Ток»).

Зелёным цветом на Рис. 10.4 выделены блоки механической части привода, состоящей из понижающего цилиндрического редуктора и ШВП, преобразующей вращение в линейное перемещение штока. Обе передачи промоделированы с учётом люфтов, инерции и трения. Здесь же в модели механической части находится модель датчика положения штока с блоком масштабирования сигнала. Блок «Возмущение» на Рис. 10.4 моделирует заданное в ТЗ возмущение на штоке.

Кроме описанных выше узлов, в модели присутствует ШИМ-преобразователь. В реальном приводе он находится в таймере микроконтроллера системы управления, но моделировать его удобнее в проекте исполнительной части, так как это аппаратный узел и его тактовая частота существенно выше, чем частота счёта программы СУ (время программного цикла). В свойствах ШИМ-преобразователя выбран специальный вид ШИМ, обеспечивающий бутстрепное питание драйверов силового преобразователя, что в данном приводе и реализовано.

Время интегрирования модели исполнительной части привода выбрано равным 0.5 мкс.

Модель СУ в нашем примере реализует алгоритм коммутации ключей инвертора по ДПР (датчику положения ротора), что обычно используется для управления СДПМ с трапецеидальной ЭДС.

Это двухконтурная система управления положением с демпфирующей связью по скорости. Её описание можно найти в разделе 6.2.3 Антиучебника Ю. Н. Калачёва и Д. В. Самохвалова.

Модель системы управления приведена на Рис. 10.5.

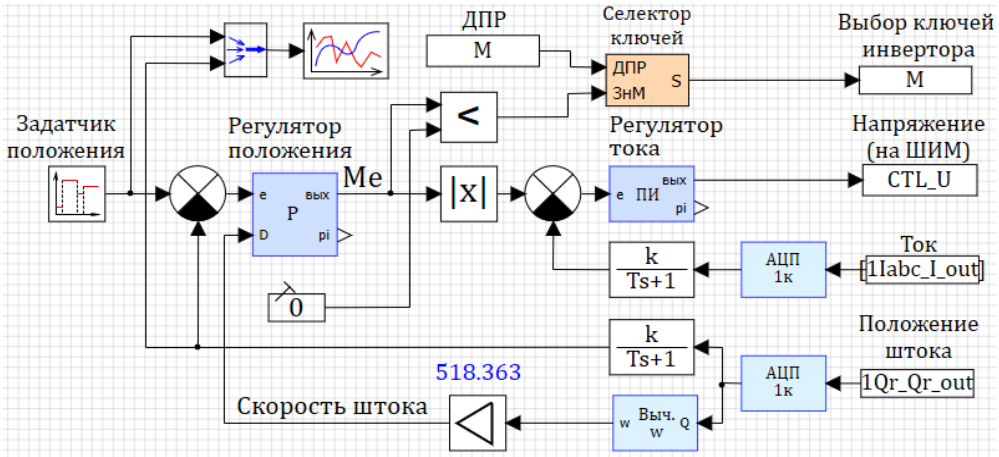


Рис. 10.5

Контур положения является внешним по отношению к контуру тока. Его можно было смоделировать отдельно (см. раздел 9.1.2), но в данном случае для упрощения модели он размещён в едином проекте с контуром тока.

Регулятор положения контура задаёт момент, необходимый для отработки рассогласования по положению (ПИ-регулирование), и вычитает из него сигнал, пропорциональный скорости, осуществляя тем самым необходимое демпфирование системы.

Скорость штока вычисляется как производная от его положения.

Чем больше скорость, тем быстрее изменяется положение и тем сильнее отрицательная обратная связь по скорости демпфирует процесс. Степень демпфирования определяется в свойствах регулятора положения заданием значения соответствующего коэффициента.

Время интегрирования модели СУ равно такту ШИМ (50 мкс).

Проекты СУ и исполнительной части привода объединяются с помощью базы данных сигналов в единый пакет, в котором синхронизируются одинаково именованные входные/выходные сигналы проектов.

Запустив пакет модели и настроив коэффициенты регуляторов СУ, можно исследовать любые режимы работы привода.

ПОДРОБНО о структуре цифрового ПИ-регулятора и способах его настройки читайте в книге Ю. Н. Калачёва и Д. В. Самохвалова «Основы регулируемого электропривода (Антиучебник)», разделы 3.5 и 3.7.

Например, можно оценить пульсации токов и момента двигателя или посмотреть, как будут вести себя токи фаз при различных возмущениях и какова при этом будет точность позиционирования.

В ТЗ к моделируемому приводу было записано, что при наличии внешнего гармонического возмущения с частотой 10 Гц и амплитудой 500 Нм шток должен перемещаться на максимальную длину (40 мм) за время не более 0.5 с.

Ниже, на Рис. 10.6, показан смоделированный процесс отработки приводом скачков задания положения при заданном в ТЗ возмущении.

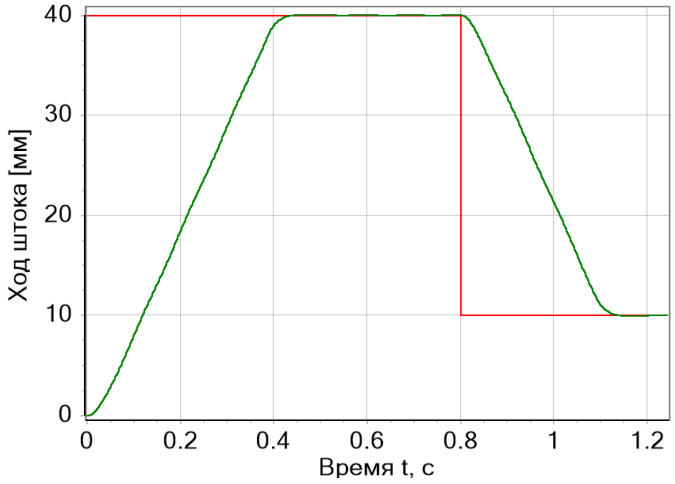


Рис. 10.6

Как видим, максимальное перемещение в 40 мм шток осуществляет примерно за 0.4 с, что удовлетворят требованиям ТЗ.

11 Кодогенерация

Как правило, в написании ПО контроллера СУ участвуют два человека:

1. инженер-приводчик (пишет ТЗ программисту или чаще объясняет ему на пальцах, что должна делать программа)
2. программист (пишет исполняемый код микроконтроллера).

Приводчик обычно не разбирается в тонкостях программирования, однако разбирается в «железе» и понимает, что нужно сделать.

Программист - напротив, в программировании – «зубр», но часто не очень в курсе, чем ему предстоит управлять.

Предполагается, что в единстве и борьбе этих двух противоположностей должна родиться истина. Однако вместо истины на стыке этих двух сущностей часто рождаются проблемы.

В жизни требуются годы взаимной «притирки», в результате которой инженер-приводчик начинает понимать специфику процесса программирования, а программист «въезжает» в область управления двигателем. Где взять таких готовых специалистов – обычно загадка. Штучный товар!

С нуля быстро получить результат не получится, немало времени уйдёт на создание понятного программисту ТЗ, написание и отладка программы займут ещё больше времени, а ошибки в коде, допущенные программистом, могут обернуться дополнительными материальными и временными потерями (если ты спалил ключи – ты хороший программист, а если ещё нет – то «зелень»).

Есть ли способ избежать ошибок и ускорить процесс разработки? Да, для этого можно попробовать использовать инструмент кодогенерации *SimInTech*.

Просто добавьте в модель системы управления приводом блоки инициализации выбранного микроконтроллера, настройте кодогенератор и лёгким движением руки получите из модели файл с *C*-текстом программы. Затем с помощью стандартных сред работы с программным обеспечением (например, *Keil* и т. п.) можно превратить этот текст в программный код (скомпилировать) и загрузить в микроконтроллер (см. Рис 11.1).

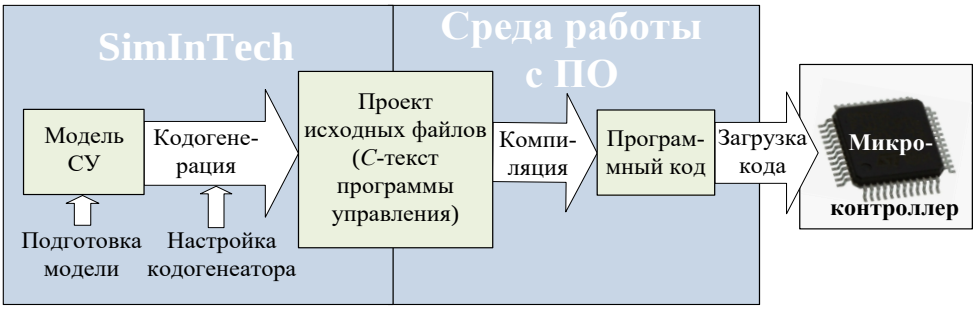


Рис. 11.1

Далее на конкретном реализованном в жизни примере покажем, как из модели двухконтурной системы управления приводом подъёмного устройства (лебёдки) автоматически получить соответствующую программу управления для конкретного микроконтроллера.

11.1 Специализированная библиотека «Электропривод»

При генерации программы из модели, построенной на элементах стандартной библиотеки «Электропривод», могут возникнуть некоторые проблемы.

Во-первых, текст программы будет неоптимален. В нём может оказаться много лишних операторов, которые затянут время вычислений программ контуров управления.

Во-вторых, для выполнения вычислений не всегда будут выбираться оптимальные с точки зрения быстродействия методы.

Во-третьих, в полученном тексте программы будут использоваться числа с плавающей точкой (*float*). Однако далеко не всегда у разработчика есть возможность использовать микроконтроллер с *FPU*.

Например, если ТЗ предписывает использовать только отечественную элементную базу - выбор у нас невелик.

Допустим из всего «многообразия» отечественных микросхем мы остановились на продукции «Миландр», как самой доступной и распространённой. Тогда в настоящее время возможно использование всего двух типов микроконтроллеров: К1986ВЕ1Т или К1986ВЕ9х.

Первый работает на высокой тактовой частоте до 144 МГц, но имеет медленное ядро *ARM M*1, второй работает на частоте 80 МГц, но в его основе лежит более совершенное ядро *ARM M*3. Сочетание тактовой частоты и продвинутости/отсталости архитектуры в результате даёт примерно одинаковое быстродействие этих контроллеров.

FPU в обоих случаях отсутствует, а значит, операции с плавающей точкой будут выполняться микроконтроллерами довольно медленно.

Значит ли это, что эти контроллеры не подходят для использования в электроприводах? Отнюдь. Если сократить время вычислений за счёт применения формата с фиксированной точкой (*fixpoint*) они вполне подойдут.

Для того чтобы решить все три вышеупомянутые проблемы, при построении модели надо применять не стандартную, а специализированную библиотеку элементов. Её элементы аналогичны элементам стандартной библиотеки, но дополнительно обеспечивают:

- оптимизацию получаемого при кодогенерации текста программы (в нём не возникает «излишеств нехороших»)
- использование табличных или быстрых приближённых способов расчёта сложных вычислительных функций, таких как извлечение квадратного корня, синус, косинус и т. д.

- возможность выбора способа расчёта, целочисленного с фиксированной точкой (*fixpoint*) или с плавающей точкой (*float*).

Ниже на рисунках представлена немного упрощённая модель векторной системы управления скоростью СДПМ (привод лебёдки), построенная на элементах специализированной библиотеки.

Модель СУ состоит из проектов двух контуров управления: внутреннего контура тока и внешнего контура скорости.

ОПИСАНИЕ такой СУ можно найти в разделах 6.1.8 и 6.1.9 книги Ю. Н. Калачёва и Д. В. Самохвалова «Основы регулируемого электропривода, Антиучебник»).

Модель контура тока СУ соответствует стандартной структуре управления проекциями вектора тока статора в ориентированной по потоку ротора системе координат *dq*. Её вид представлен на Рис. 11.2.

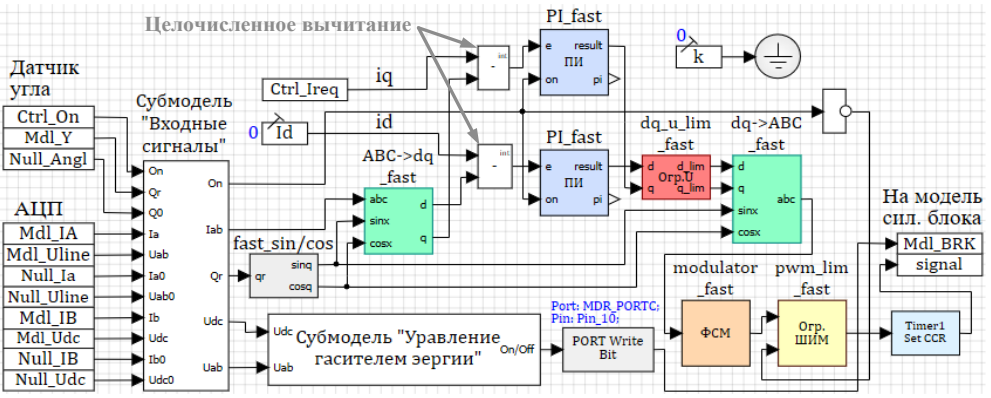


Рис. 11.2

Значение i_d поддерживается равным 0, а определяющая момент величина i_q задаётся внешним контуром скорости.

Основу системы регулирования составляют преобразователи координат (*ABC=>dq*, *dq=>ABC*), ПИ-регуляторы (*PI_fast*), ограничитель напряжения (*dq_u_lim_fast*) и формирователь способа модуляции (*modulator_fast*). Эти элементы специализированной библиотеки выполняют те же функции, что и их аналоги из стандартной библиотеки.

Сигналы полученных фазных напряжений с выхода формирователя закона модуляции подаются на блоки «Огр. ШИМ» и «*Timer1 Set CCR*». Эти блоки при кодогенерации обеспечивают запись значений управляющих фазных напряжений в нужном формате в регистры таймера ШИМ.

Модель самого ШИМ-преобразователя находится в проекте исполнительной части пакета модели электропривода. Ранее уже писалось, что таймер ШИМ - это аппаратный узел, и его моделирование требует более мелкого шага интегрирования, чем время цикла расчёта контура тока.

Кроме собственно системы регулирования в контуре тока находятся две субмодели:

- субмодель «Входные сигналы» обеспечивает работу АЦП и выдачу на систему регулирования сигнала включения (*Ctrl_On*)
- субмодель «Управление гасителем энергии» обеспечивает логику включения ключа - гасителя энергии торможения.

Вид модели контура скорости рассматриваемой СУ представлен на Рис. 11.3.

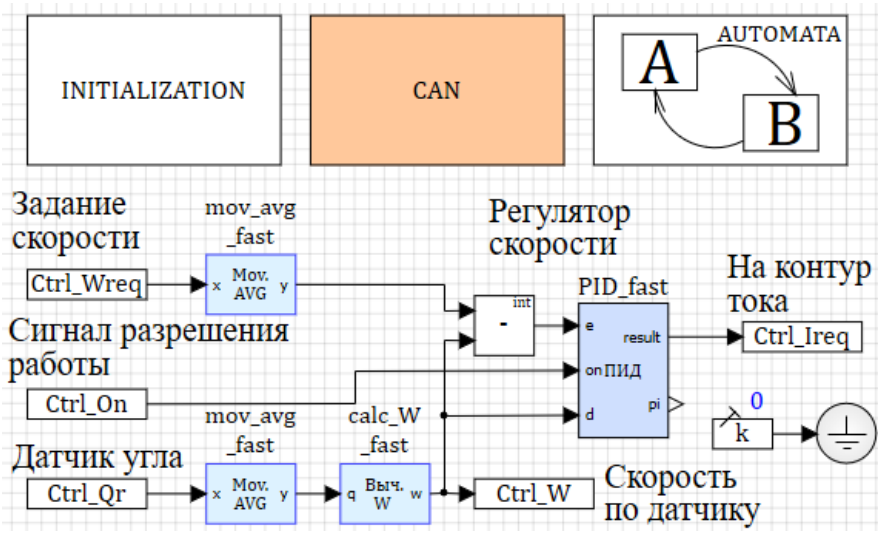


Рис. 11.3

Регулирование скорости осуществляется ПИД-регулятором (*PID_fast*). На его вход подаётся сигнал разности между заданным значением скорости и её величиной, вычисленной блоком *calc_W_fast* по сигналам датчика положения ротора (датчика угла).

Блоки *Mov.AVG* осуществляют фильтрацию помех в сигналах задания скорости и датчика угла.

Контур скорости включает в себя также субмодели инициализации микроконтроллера (*INITIALIZATION*), информационного обмена (*CAN*) и сервисных функций (*AUTOMATA*). О них будет написано в следующих разделах.

На Рис. 11.4 приведены свойства ПИ-регулятора использованного в модели контура тока (*PI_fast* на Рис. 11.2). Обратите внимание на выбранное значение свойства «Способ расчёта».

Название	Имя	Формула	Значение
Внешнее ограничение	ex_lim		☐ Нет
Внешнее разрешение	ex_on		☑ Да
Максимальная интегральная сумма	PI_max	2000	2000
Минимальная интегральная сумма	PI_min	-2000	-2000
Пропорциональный коэффициент	Kp	2.5	2.5
Интегральный коэффициент	Ki	600	600
Способ расчета	type		fixpoint

Рис. 11.4

Блоки рассматриваемых проектов используют способ расчёта *fixpoint*, что снижает требования к производительности микроконтроллера и позволяет, реализовать векторное управление СД на отечественном Миландре, или дешёвом *STM32F103*.

При моделировании привода все вышеописанные «фишки» блоков специализированной библиотеки полностью сохраняются.

Если пользователь выбрал целочисленный вариант расчёта блоков, то вычисления при моделировании будут производиться в целых числах. В результате на этапе моделирования мы сможем, в том числе, оценить влияние на работу привода целочисленного способа расчёта.

11.2 Инициализация микроконтроллера

Секция *INITIALIZATION* (см. Рис. 11.3) необходима для привязки модели к конкретному типу микроконтроллера. Она обеспечивает генерацию части программы, отвечающей за настройки его периферии (портов, интерфейсов) и базовых функций (например, ШИМ). Назовём процесс создания этой секции инициализацией микроконтроллера.

Прежде, чем приступить к инициализации, зайдём в настройки наших проектов модели СУ лебёдки. Откроем в меню проектов вкладки

«Параметры расчёта» и выберем разделы «Генерация кода» (см. Рис. 11.5). В полученных подменю заполним строки «Имя алгоритма». В нашем случае для модели контура тока мы использовали имя *vector_pwm,* а для контура скорости – *speed_ctrl.*

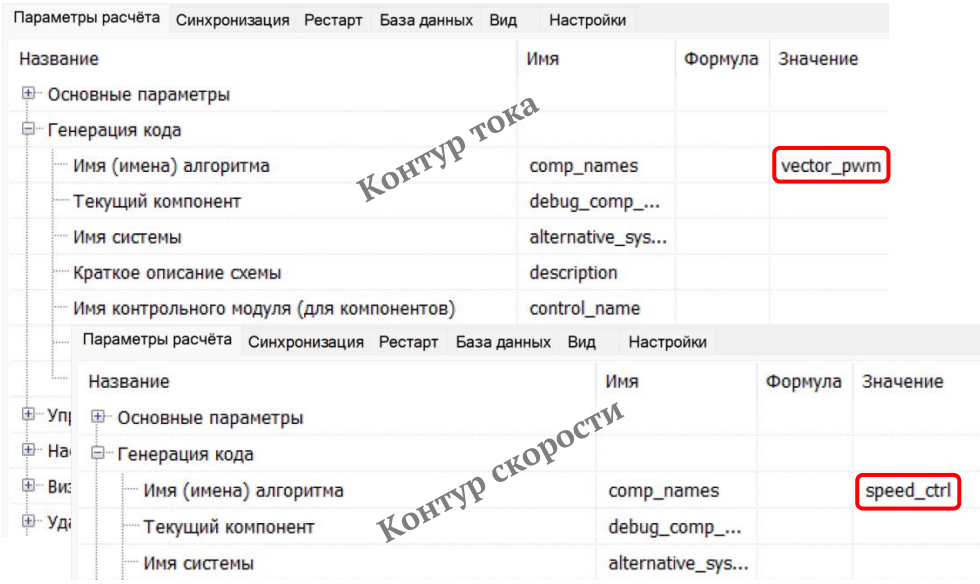


Рис. 11.5

Теперь можно приступить к инициализации (конфигурированию) микроконтроллера. Для работы с различными типами микроконтроллеров в *SimInTech* существуют специальные библиотеки. В нашем случае будем использовать микроконтроллер Миландр К1986ВЕ9х и его библиотеку (Рис. 11.6).

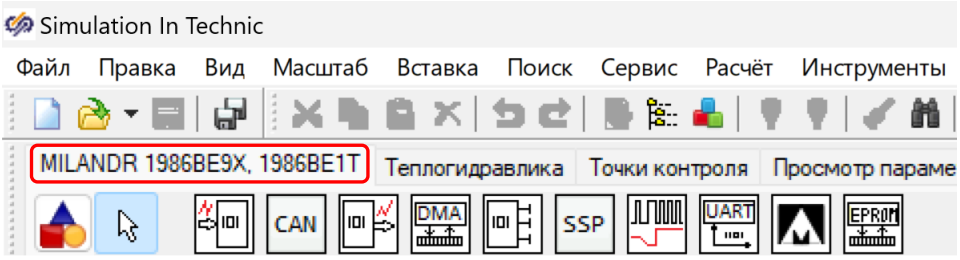


Рис. 11.6

Библиотека содержит блоки инициализации для таймеров, ЦАП, АЦП, портов ввода/вывода, интерфейсов (*CAN*, *UART*, *SPI*) и т.д.

Для осуществления инициализации достаточно поместить соответствующие блоки библиотеки в проект одного из контуров, настроить их параметры и при желании задать порядок кодогенерации, соединяя выходы одних блоков с входами других.

На Рис.11.7 приведён пример секции инициализации микроконтроллера К1986ВЕ9х.

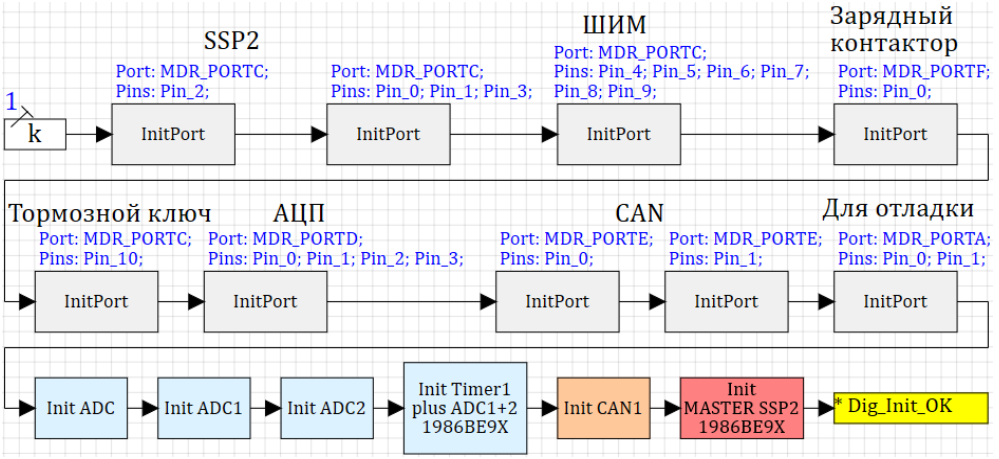


Рис. 11.7

По рисунку можно понять, что с помощью данных блоков настраиваются функции выводов портов микроконтроллера, режимы работы АЦП, таймера ШИМ и интерфейсов обмена *CAN* и *SPI*.

Чтобы не загромождать схему контура скорости, все блоки инициализации удобно упаковать внутрь стандартного блока «Субмодель» (см. Рис. 11.8).

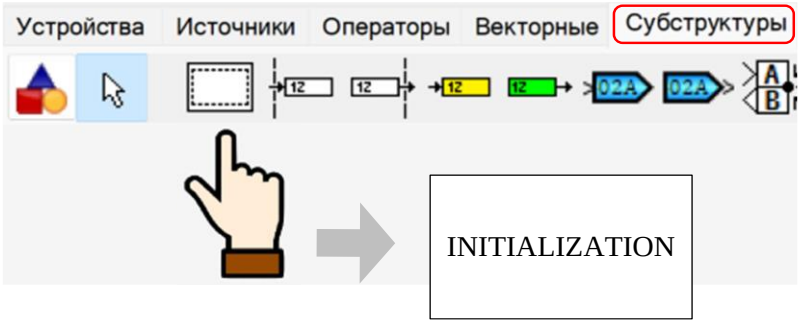


Рис. 11.8

В качестве примера осуществим настройку ШИМ-генератора и назначим соответствующие ножки микроконтроллера выходами таймера ШИМ.

ПРИМЕЧАНИЕ

Перед настройкой необходимо изучить описание режимов работы ШИМ, приведённое в спецификации на микроконтроллер К1986ВЕ9х.

Меню свойств блока инициализации таймера ШИМ (*Init Timer*1 на Рис. 11.7) представлено на Рис. 11.9.

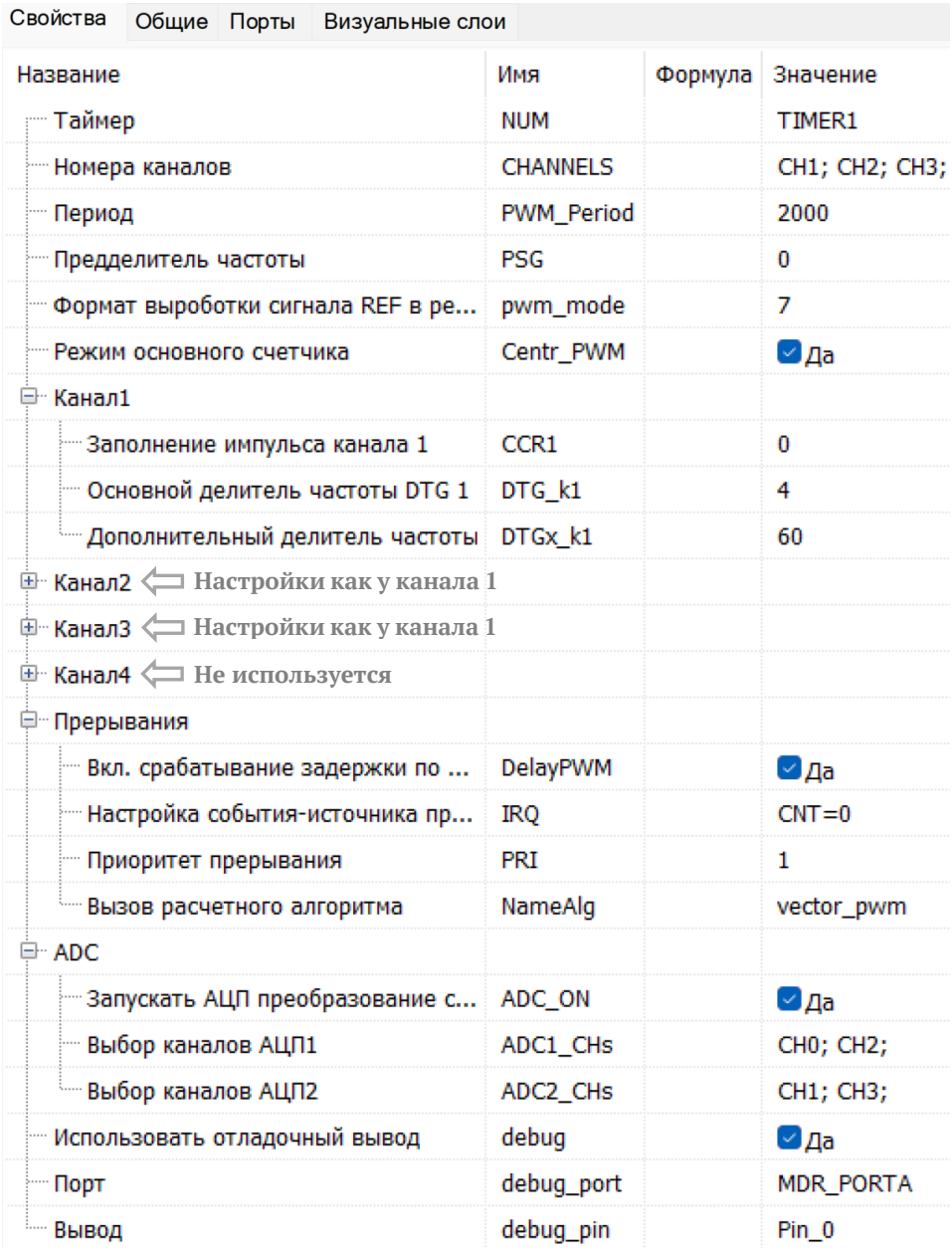


Рис. 11.9

Таймер управляет шестью ключевыми транзисторами (тремя силовыми стойками инвертора). Каждый из каналов таймера может формировать два сигнала управления, прямой и инверсный. Исходя из

этого, установим в строке «Таймер» значение «*TIMER*1» и в строке «Номера каналов» значения «*CH*1, *CH*2, *CH*3».

1. Настроим режимы работы таймера.
   - Установив в строке «Режим основного счётчика» значение «*Centr_PWM* - да», выберем режим симметричной ШИМ. В этом режиме счётчик таймера сначала инкрементируется до максимального значения, а затем декрементируется до нуля.
   - Тактовая частота работы счётчика определяется выбранным режимом предделителя частоты и тактовой частоты микроконтроллера. При записи в строке «Предделитель частоты» значения «0» тактовая частота счётчика таймера будет равна частоте тактирования микроконтроллера (F_{CPU}=80 МГц).
   - Записав в строку «Период» значение «2000», зададим максимальное значение счётчика, при этом частота симметричного треугольника ШИМ будет равна 20 кГц ($(80 \cdot 10^6)/(2*2000)$).
   - Задав коэффициенты деления основного и дополнительного делителей частоты (*DTG_k*1 и *DTGx_k*1), можно настроить длительность мёртвого времени. Зададим его равным 3 мкс. $t_{ЗП} = \left(DTG\_k\ 1 \cdot DTGx\_k\ 1\right)/F_{CPU} = \left(4 \cdot 60\right)/(80 \cdot 10^6) = 3 \cdot 10^{-6}$ с.
   - В строке «Формат выработки сигнала *REF* в режиме ШИМ» установим значение «*pwm_mode*=7» (подробности см. в спецификации на микроконтроллер 1986ВЕ9х, обычно используют 7 или 6).
2. Настроим прерывания (вызов программы контура тока)
   - В строке «Настройка события-источника прерывания» выберем «*CNT*=0», при этом прерывание будет осуществляться в момент достижения счётчиком таймера нулевого значения.
   - Зададим высокий приоритет прерыванию *PRI* = 1.
   - В строке «Вкл. срабатывание задержки по истечению периода ШИМ» запишем «Да», это нам понадобится далее для настройки частоты вызова медленного (внешнего) контура управления.
   - Зададим имя вызываемого прерыванием расчётного алгоритма. Оно должно совпадать с именем алгоритма, заданного в параметрах проекта нашего контура тока («*NameAlg* = *vector_pwm*» см. Рис. 11.5).

3. Настроим опрашиваемый АЦП (раздел «*ADC*» на Рис. 11.9)
   - Включим опрос АЦП в прерывании (*ADC_ON* - да). В этом случае выполнение программы прерывания начнётся с опроса выбранных каналов АЦП.
   - Выберем канал *CH*0 и *CH*2 для АЦП1 (измерение тока фазы *A* и напряжения в сети) и канал *CH*1 и *CH*3 для АЦП2 (измерение тока фазы B и напряжения звена постоянного тока).
4. Можно активировать свойство «Использовать отладочный вывод» («Да»). Тогда при входе/выходе из прерывания программа будет «дёргать» выбранную ножку микроконтроллера (*Pin*_0 порта *A*, см. Рис. 11.9). Подключив к ней осциллограф, можно будет измерить время расчёта контура тока. Это будет очень полезно в дальнейшем, при отладке программы.
5. Остаётся назначить выводы (пины) портов микроконтроллера выходами каналов таймера. Пример настройки свойств блока инициализации порта приведён на Рис 11.10 (какие выводы микроконтроллера могут быть использованы для этого, можно узнать из спецификации на микроконтроллер К1986ВЕ9х).

Свойства | Общие | Порты | Визуальные слои

Название	Имя	Формула	Значение
Порт	PORT		MDR_PORTC
Номера пинов	PINS		Pin_4; Pin_5; Pin_6; Pin_7; Pin_8; Pin_9;
Направление ввода	PORT_OE		OUT
Тип порта	PORT_MODE		DIGITAL
Функция	PORT_FUNC		OVERRID
Скорость	PORT_SPEED		FAST

Рис. 11.10

Остальные блоки настраиваются аналогичным способом (смотрите спецификацию на микроконтроллер). Подробно на этом более останавливаться не будем, примеры настройки всех библиотечных блоков микроконтроллеров Миландра лежат тут:
*SimInTech**Demo**Microprocessors*\Примеры – МК 1986ВЕ9Х, 1986ВЕ1Т.

ПРИМЕЧАНИЕ
Заметим, что работа с любым микроконтроллером предполагает изучение его спецификации, а это серьёзный документ. Например, спецификация на К1986ВЕ9Х — это книжка на 500 страниц.

11.3 Сервисные функции привода

Почти невозможно представить себе электропривод без встроенных сервисных функций, таких как контроль за исправностью, переключение между разными режимами работы и др.

Обычно приводы имеют развитую систему различных защит. Например, защиту от недопустимого понижения/повышения напряжения питания, максимально-токовую и время-токовую защиты, защиту от ошибки отработки заданного параметра и т.д.

В основу многих защит положена простая логика сравнения измеренного или вычисленного значения с заданным порогом. Обычно, если это значение выше/ниже порога в течение заданного времени, защита должна сработать и выполнить некоторые действия.

Например, осуществить отключение силовых ключей, выдать сигнал индикации, сообщить о случившемся системе управления верхнего уровня и т. д.

Иногда после срабатывания некоторых защит требуется не только отключиться, но и через определённое время осуществить автоматический сброс защиты с последующим перезапуском привода.

Бывает, однако, и такое, что никакие защиты вообще не нужны. Например - в спецтехнике. Бессмысленно защищать транзистор привода рулевой машинки, например, от перегрева, если при этом изделие не долетит до цели.

Перечень защит, последовательность действий в случае их срабатывания и способ выдачи сигналов статуса в систему управления верхнего уровня обычно определяются в ТЗ.

Там же определяется и логика переключения между различными режимами работы привода.

В соответствии с этими требованиями СУ должно управлять состоянием привода в зависимости от его текущего состояния и предыстории. При этом электропривод можно рассматривать как конечный автомат, имеющий некоторое количество входных и несколько выходных состояний, например: «Готов (к работе)», «Работа» и «Защита» и т. п.

В *SimInTech* существует раздел конечных автоматов.

Используя элементы этого раздела, можно:

1. задать возможность перехода между состояниями (карту переходов)

2. задать выходные сигналы, соответствующие каждому состоянию
3. реализовать логику перехода из состояния в состояние - например, из состояния «Работа» в состояние «Защита» и обратно
4. определить сигналы вызывающие эти переходы.

Вид проекта построенного на элементах раздела «Конечные автоматы» представлен на Рис. 11.11.

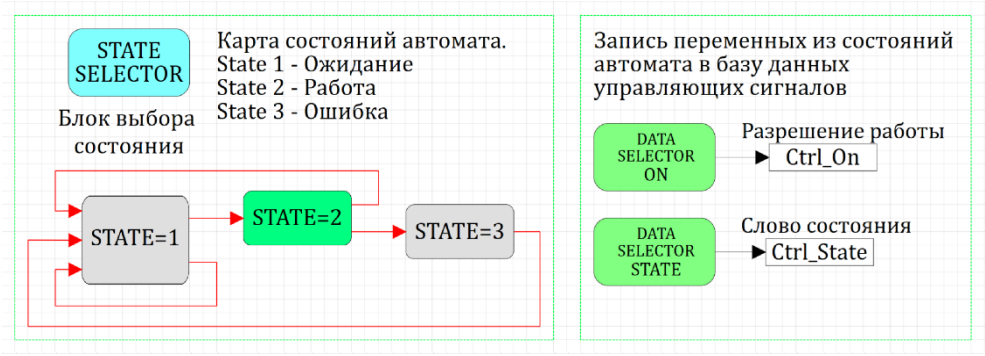


Рис. 11.11

ПРИМЕЧАНИЕ

Рекомендуем дополнительно почитать соответствующие разделы справок по блокам конечных автоматов и посмотреть примеры из *C:\SimInTech\Demo*Конечные автоматы. При возникновении вопросов не стесняйтесь их задавать разработчикам среды.

SimInTech позволяет отладить логику работы конечного автомата отдельно от общей модели привода, что достаточно удобно.

Для отладки модель обычно дополняется нужными источниками входных сигналов, которые можно подавать с блоков-генераторов или вручную, используя элементы интерфейса из панели «Примитивы»: кнопки, крутилки, ползунки и т.д.

Время интегрирования модели при этом можно задавать достаточно большим.

После завершения отладки блока управления состояниями он вставляется в модель, что при дальнейшей кодогенерации автоматически обеспечивает встраивание его программы в программу соответствующего контура.

Обычно этот блок располагают в медленном (внешнем) контуре управления. В нашем примере он находится в субмодели «*AUTOMATA*» в проекте контура скорости (см. Рис. 11.3).

11.4 Последовательный интерфейс

В нашем примере для обмена информацией между СУ привода и внешними устройствами применяется последовательный интерфейс *CAN*. Он может использоваться как для управления, так и для отладки программы микроконтроллера СУ из среды *SimInTech*.

ПРИМЕЧАНИЕ
Использование *CAN* не обязательно, но удобно и современно. Однако никто не запрещает использовать традиционный последовательный интерфейс *UART* в его защищённом исполнении *RS*-485…

Пусть привод управляется от компьютера (ПК), связь с которым осуществляется с помощью конвертора *USB-CAN VS-COM*. Блок *SimInTech*, выполняющий это конвертирование, так и называется – *USB-CAN* и находится в библиотеке «Устройства».

На Рис. 11.12 приведено, для примера, содержание простейшего проекта *SimInTech*, позволяющего управлять приводом от ПК и выводить на экран получаемые от СУ привода данные.

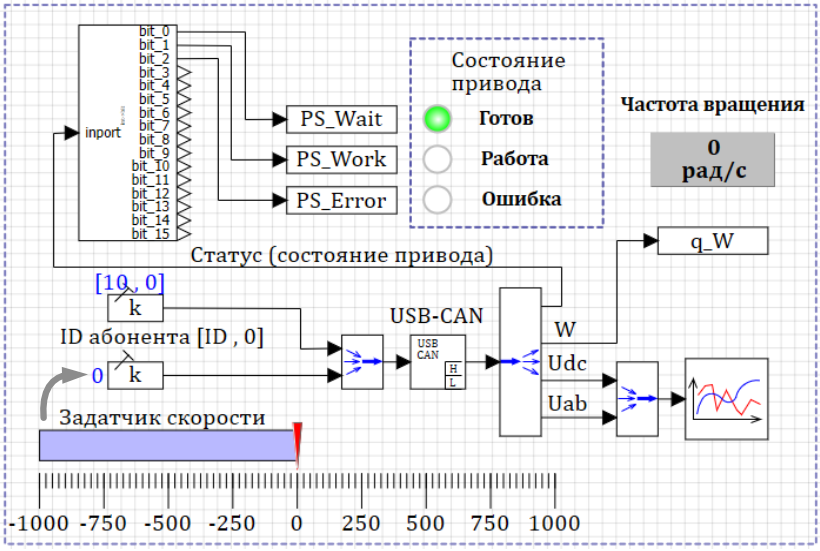


Рис. 11.12

Прямо из этого проекта, запущенного на ПК в *SimInTech*, можно задавать частоту вращения двигателя, а также принимать и индицировать:

- слово состояний привода (Готов, Работа, Ошибка)
- частоту вращения двигателя (*W*)
- напряжения силовой сети и шины постоянного тока (*Uab, Udc*).

На вход блока “*USB-CAN”* в проекте подаются:

- адрес (*ID*) устройства СУ (в нашем случае вектор [10, 0])
- частота вращения двигателя (задаётся движком блока «Линейный прибор» из библиотеки «Сигналы» («Панель примитивов»).

Выделение битов статуса «*PS_Wait*», «*PS_Work*», «*PS_Error*» осуществляется блоком «Распаковка битов» (библиотека «Логические»). Для индикации статуса используются блоки «Индикатор» библиотеки «Электропривод».

Скорость двигателя выводится на цифровой индикатор «Отображение произвольной величины» библиотеки «Электропривод».

Напряжения выводятся на блок «Временной график» библиотеки «Вывод данных».

ПРИМЕЧАНИЕ

В *SimInTech* при желании можно создавать и существенно более сложные и информативные пульты управления и индикации.

Настройка блока “*USB-CAN”* производится с помощью задания его свойств (см. Рис. 11.13).

Свойства | Общие | Порты | Визуальные слои

Название	Имя	Формула	Значение
Подключение устройства			
Скорость обмена	CANSpeed		1M
Номер Com порта	PortNumber	4	4
Режим работы	WorkMode		Нормальный
Передача данных			
Временная метка	TimeStamp		0
Количество абонентов	AbonentsQty	1	1
Шаг передачи (сек)	TimeStep	1/100	0.01
Формат отправляемого кадра	SendFormat		INTEGER (4);
Прием информации			
Итендификатор	FilterIDs	11	[11]
Маска	FMasks	2047	[2047]
Признак расширения	FExtands		[0]
Формат принимаемого кадра	ReciveFormat		WORD (2); SIGNEDWORD(2)...
Порядок байт			
Формат WORD	WordOrder		AB
Формат INTEGER	IntegerOrder		AB CD
Формат LONG	LongOrder		AB CD
Формат FLOAT	FloatOrder		AB CD
Формат DOUBLE	DoubleOrder		AB CD EF GH

Рис. 11.13

Обратите внимание на настройку форматов отправляемого и принимаемого кадров. В нашем примере ПК получает 4 числа: беззнаковое длиной 2 байта (*WORD*(2)) и три знаковых числа длиной 2 байта каждый (*SIGNEDWORD*(2)). А отправляет одно знаковое число длиной 4 байта (*INTEGER*(4)). Суммарная длительность одного принимаемого кадра *CAN* не может превышает 8 байт.

ПОДРОБНО о блоке "*USB-CAN*" читайте в справке *SimInTech*.

Рассмотрим теперь субмодель *CAN*, расположенную в контуре скорости (см. Рис. 11.3). Её содержание представлено на Рис. 11.14.

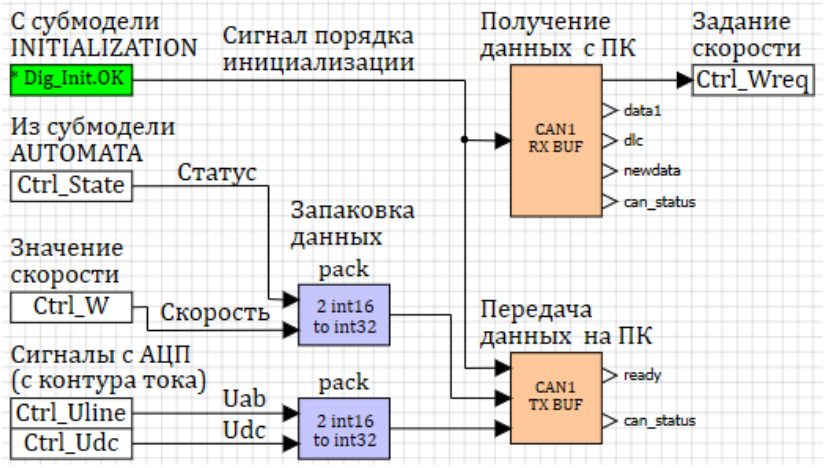


Рис. 11.14

ПРИМЕЧАНИЕ

Частота обмена пакетами по шине *CAN* ограничена особенностями работы операционных систем на ПК. Передавать пакеты данных от СУ на ПК можно достаточно часто (в нашем примере с частотой контура скорости 1 кГц), а отправлять данные с ПК на СУ преобразователя, скорее всего, не получится с частотой более 100 Гц.

В модели на Рис. 11.14 используются блоки (*CAN*1) из раздела *MILANDR* библиотеки блоков *SimInTech*.

Микроконтроллер получает заданную частоту вращения в рад/с в виде целого 32-битного числа, а передаёт на ПК четыре 16-битных слова (Статус, Скорость, *Uab*, *Udc*), запакованные в два стандартных 32-битных буфера *CAN*. Для преобразования 16-битных чисел в 32-битное используются блоки «запаковка» из специализированной библиотеки «Электропривод».

На Рис. 11.15 приведены свойства блока инициализации *CAN*1 для 1986BE9x, расположенного в субмодели *INITALIZATION* контура скорости.

Свойства : mdr_Init_CAN_1

Свойства | Общие | Порты | Визуальные слои

Название	Имя	...	Значение
Выбор CAN контроллера	CAN_NUM		CAN1
Прием собственных пакетов	ROP		ENABLE
Подтверждение собственных пакетов	SAP		DISABLE
Режим самотестирования	STM		DISABLE
Режим «Только прием»	ROM		DISABLE
Значение размера фазы PSEG	PSEG		CAN_PSEG_Mul_3TQ
Значение размера фазы SEG1	SEG1		CAN_SEG1_Mul_8TQ
Значение размера фазы SEG2	SEG2		CAN_SEG2_Mul_8TQ
Значение размера фазы SJW	SJW		CAN_SJW_Mul_4TQ
Семплирование	SB		CAN_SB_1_SAMPLE
Предделитель системной частоты	BRP		3
Максимальное значение сетчиков о...	ERR_MAX	...	255
Базовый предделитель частоты CP...	PSG		0
IRQ			
Включить прерывание по приему...	IRQ_EN		Да
Приоритет прерывания CAN	IRQ_PRI		2
Выбор буферов на прием	BUF_ARR		CAN_BUFFER_0 ;

Рис. 11.15

Приём сообщений реализован через использование прерывания, приоритет которого ниже, чем у прерывания по таймеру ШИМ.

ПРИМЕЧАНИЕ

В инициализации *CAN* предлагаем разобраться самостоятельно, изучая справку *SimInTech*, документацию на микроконтроллер и настройки нашего примера.

11.5 Преобразование модели в текст программы

11.5.1 Настройка кодогенератора

Откроем вкладку «Кодогенератор» и выберем подменю «Настройки» (см. Рис. 11.16).

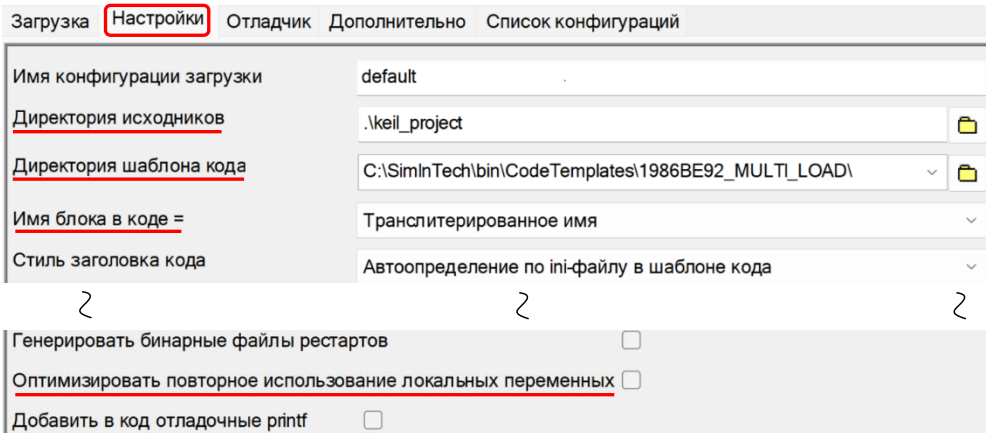


Рис. 11.16

В строке «Директория шаблона кода» указывается путь к папке с шаблоном кодогенерации.

Шаблон — это заранее подготовленный проект в *IDE Keil v*5, в который при кодогенерации встраивается сгенерированный *C*-код программы управления.

Для выбранного нами микроконтроллера шаблон находится по адресу: «*C\SimInTech\bin\CodeTemplates\1986BE92_MULTI_LOAD*».

В строке «Директория исходников» указывается адрес, по которому будут помещаться результаты кодогенерации (файлы проекта *Keil*). В случае, изображённом на Рис. 11.16, файлы будут помещены в папку размещения нашей модели, в директорию *keil_project*. Туда же при кодогенерации копируется шаблон, дополненный результатами кодогенерации (файлами **.inc)*.

И последнее, желательно снять галочку в пункте «Оптимизировать повторное использование локальных переменных». Иногда эта функция может мешать.

Остальные настройки для начала лучше не трогать и оставить без изменений.

БОЛЕЕ ПОДРОБНО о кодогенерации читайте в справке *SimInTech\Demo\Microhuocessors*\Примеры-МК1986ВЕ9Х, «Генерация Си-кода для микропроцессоров с использованием среды *SimInTech*»

11.5.2 Кодогенерация

Войдём во вкладку «Загрузка». Выберем файлы модели *SimInTech*, предназначенные для кодогенерации (кнопка с изображением плюса). Выбранные файлы появятся в списке окна (см. Рис. 11.17).

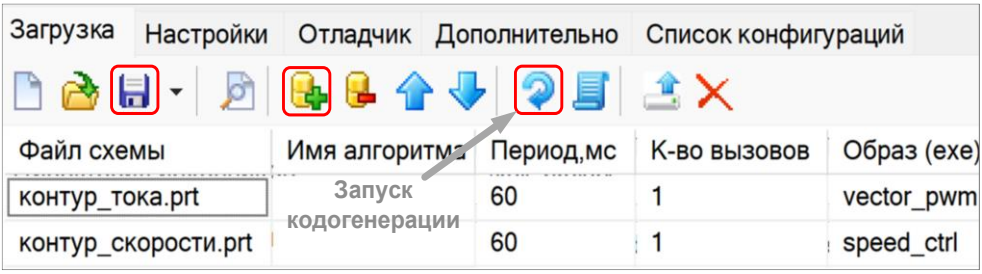


Рис. 11.17

Имя проекта внутреннего контура необходимо разместить в верхней строке окна. Его программа должна вызываться по прерываниям ШИМ (см. настройку таймера ШИМ на Рис. 11.9). В строке ниже помещается имя проекта внешнего контура. Блок инициализации надо размещать именно в этом проекте. Такой порядок обеспечивает генерацию текста основной программы на базе расположенного в нижней строке внешнего контура (скорости), из которой по прерываниям ШИМ будет вызываться программа внутреннего контура (тока).

Порядок строк можно изменять кнопками-стрелочками.

Сохраним настройки кодогенератора в файл (кнопка дискета).

Запустим кодогенерацию. Если в процессе возникнут ошибки, программа об этом сообщит, а если всё пройдёт нормально, то мы увидим в отчёте сообщение «Кодогенерация прошла успешно» и в папке, которую мы назвали *keil_project*, появятся файлы проекта в среде *Keil*.

11.6 Компиляция и зашивка программы

Для дальнейшей работы нам понадобится *Keil*. Это программная среда, предназначенная для разработки и отладки (дебага) программного обеспечения микроконтроллеров на базе *Arm Cortex-M* (наш случай). *Keil* позволяет превратить текст программы в программный код (скомпилировать) и залить его с помощью программатора в микроконтроллер.

Установочные файлы версии *Keil*, ограниченной по размеру кода программы на уровне 32 Кб, можно скачать с официального сайта разработчика программы. Однако в условиях санкций, возможно, придётся поднять флаг с весёлым Роджером и скачать исходники, сами знаете где.

Также нам понадобится установить в *Keil* библиотеку для контроллеров Миландр (версии *C* 1.5.3). Для этого достаточно просто скопировать все файлы в папку, где *Keil* хранит библиотеки по умолчанию: *C:\Keil_v5\ARM\PACK\Keil\MDR1986BExx\1.5.3*.

Немного расскажем, как пользоваться *Keil*-ом и что из себя представляет проект, созданный *SimInTech* в ходе кодогененрации.

Перейдём в папку с проектом (*keil_project*), полученным в ходе кодогенерации, и откроем файл проекта *Keil*. По умолчанию он называется *Test_Driver.uvprojx*. На Рис. 11.18 представлен вид окна программы с нашим проектом.

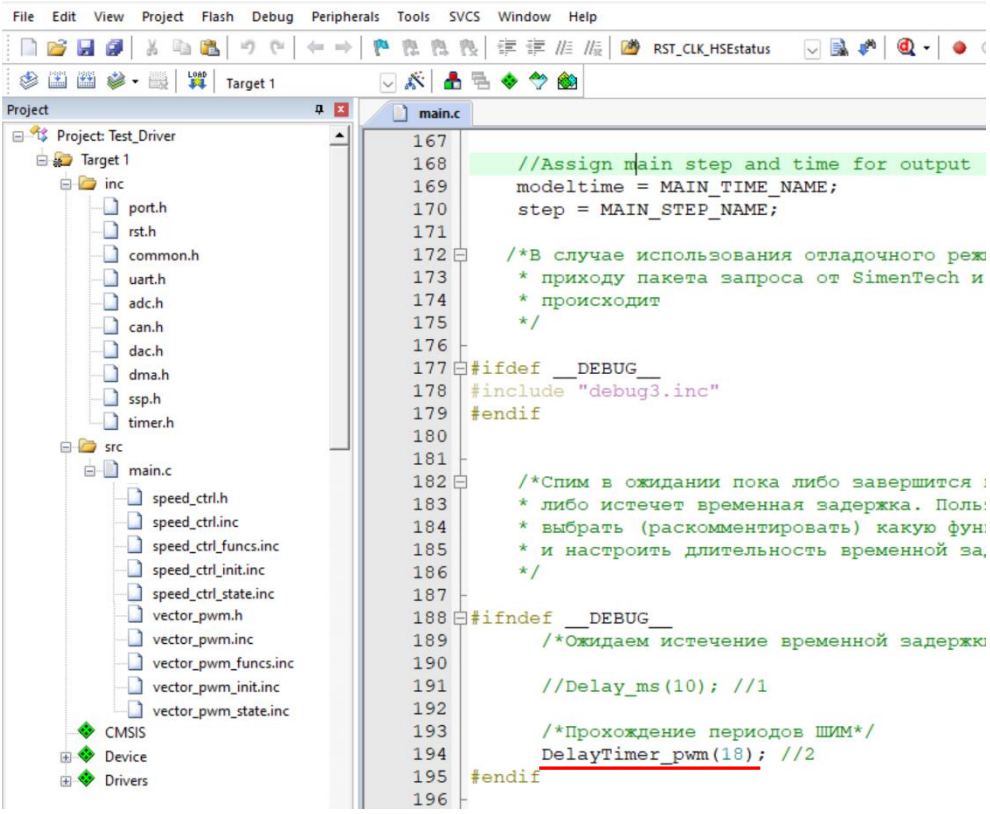


Рис. 11.18

В папке *inc* проекта лежат файлы «заголовки», в которых описаны функции и переменные, необходимые, например, для инициализации периферии нашего микроконтроллера.

В папке *src* находятся «исходники» (*C*-тексты) программ: файлы с расширениями «*.c*» и «*.inc*».

В файле *main.c* находится текст основной программы управления, устанавливающей порядок вызова функций, описанных в файлах **.inc*.

Внимательный читатель, немного знакомый с особенностями применения микроконтроллеров, заметит, что мы нигде не производили настройку тактовой частоты *CPU* микроконтроллера. На самом деле это не так, функция настройки тактовой частоты присутствует в шаблоне проекта *Keil* и по умолчанию настроена на базовую частоту 80 МГц (при использовании внешнего кварца на 8 МГц). При необходимости пользователь может отредактировать настройку тактовой частоты, всё необходимое для этого лежит в файлах *inc/rst.h* и *main.c*.

Пользователю остаётся задать частоту вызова программы контура скорости. Для этого используется программный счётчик микроконтроллера *DelayTimer*, привязанный в нашем случае к прерыванию по таймеру *TIMER*1.

НАПОМИНАЕМ, что ранее в блоке инициализации таймера мы включили опцию «задержка по истечению периода ШИМ», что позволяет использовать в программе функцию *DelayTimer_pwm*.

Для того, чтобы понять, как эта функция работает, вспомним, что программа контура скорости считается в фоновом режиме в цикле контура тока (см. раздел 9.1.3).

Кроме алгоритма непосредственного регулирования скорости, программа скоростного контура исполняет различные служебные, интерфейсные и защитные алгоритмы. Пусть все эти вычисления выполняются в фоновом режиме в течение *n*-циклов контура тока.

Если в нашей программе в качестве аргумента функции *DelayTimer_pwm* записать некоторое число *m*, то она будет запускать цикл контура скорости на каждые *n*+*m* циклов контура тока.

Допустим:

- в рассматриваемой нами СУ для выполнения программы скоростного контура (*main.c*) в фоновом режиме будет достаточно двух циклов контура тока
- программу *main.c* мы хотим запускать с частотой 1 кГц.

Тогда, при частоте ШИМ, равной 20 кГц, функция задержки должна иметь вид: *DelayTimer_pwm* (18). При этом один расчёт контура скорости будет соответствовать 20-ти циклам контура тока.

Итак, чтобы задать частоту вызова программы контура скорости:

1. откроем проект *Keil*
2. найдём в дереве проекта файл с именем *main.c* и откроем его
3. откорректируем текст, так, как показано в подчёркнутой строке на Рис. 11.18.

Теперь можно скомпилировать наш проект *Keil* и загрузить прошивку в микроконтроллер при помощи программатора, как показано на Рис. 11.19.

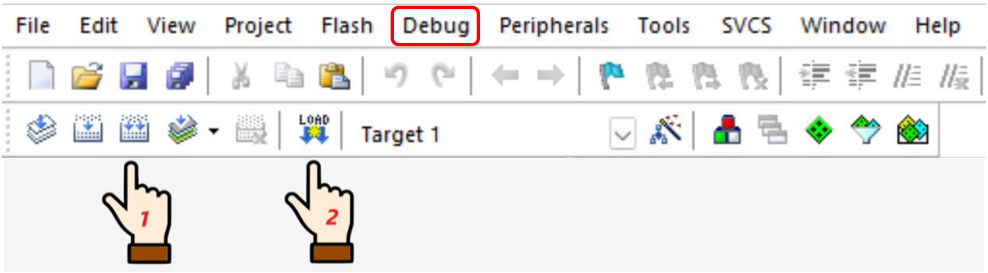


Рис. 11.19

Для измерения частоты выполнения программы контура скорости осциллографом, в *main.c* неплохо предусмотреть возможность «дёргать» заранее настроенной ножкой какого-либо порта микроконтроллера. Для этого нужно раскомментировать пару строчек кода, как показано на Рис. 11.20. Но не забудьте добавить блок инициализации порта на схему проекта и настроить нужную ножку на выход (на Рис. 11.20 это ножка 1 порта A).

```
156     /*Необходимо для замера времени выполнения алгоритма,
157     * необходимо пин 0 порт А инициализировать, выставляем 1
158     */
159     PORT_SetBits ( MDR_PORTA, PORT_Pin_1);
160
161
162     //Call main program step (для RT задержки использовать MA
163      MAIN_CYCLE_PROGRAM_NAME();
164
165     /*Сбрасываем порт*/
166     PORT_ResetBits ( MDR_PORTA, PORT_Pin_1);
```

Рис. 11.20

ПРИМЕЧАНИЕ

- Файлы текста функций основной программы (*speed_ctrl.inc, vector_ctrl.inc*) и секции инициализации (*speed_ctrl_init.inc*) можно открыть в *Keil*, посмотреть, а при необходимости даже изменить вручную.
- Вообще-то, использование исключительно *Keil* не обязательно. Пользователь, взяв за основу существующий шаблон, может создать свой собственный – для любой другой среды разработки программного обеспечения (*IDE*).

11.7 Отладка программы

Во-первых, необходимо убедиться, что контроллер работает, а программы контуров регулирования вызываются с правильной периодичностью (в нашем случае – 50 мкс и 1 мс). Для этого можно использовать отладочные выводы микроконтроллера и осциллограф.

На отладочном выводе, который мы заранее настроили в блоке таймера (см. Рис. 11.9), надо измерить частоту импульсов и их длительность. Частота соответствует частоте вызова прерывания (частоте вызова контура тока), а длительность показывает время расчёта контура. Если мы всё сделали правильно, то увидим на осциллографе импульсы частотой 20 кГц и длительностью несколько меньше 50 мкс, например, 30 мкс.

Далее необходимо убедиться, что и алгоритм контура скорости также работает на желаемой частоте (1 кГц). Для этого подключим наш осциллограф к отладочному выводу, который мы заранее настроили в *main.c* (см. Рис. 11.20). И что же мы тут увидим?

Возможны варианты.

Если мы угадали (см. предыдущий раздел) и программе контура скорости достаточно для расчёта в фоновом режиме двух циклов прерываний таймера ШИМ, то мы увидим импульсы частотой 1 кГц.

Если двух циклов контура тока не хватило, то частота импульсов будет меньше. В этом случае придётся:

- скорректировать значение аргумента функции *DelayTimer* (см. Рис. 11.18) в меньшую сторону
- вновь скомпилировать программу и запрограммировать наш микроконтроллер
- проверить частоту вызова контура скорости
- и т. д. – до получения нужного результата

Кроме отладочных выводов, для настройки программы удобно использовать встроенный в микроконтроллер ЦАП. На него можно в процессе работы выводить любую отладочную информацию, например, из контура тока.

Ещё одним удобным инструментом отладки может стать последовательный интерфейс (в нашем примере *CAN*). Используя его, можно управлять приводом и выводить отладочную информацию на экран ПК так, как это описано в разделе 11.4.

В общем, тот, кто будет работать, разберётся.

Если надо – поможем, обращайтесь!

11.8 Как создать свой собственный блок

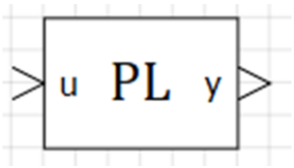


Рис 11.21

Часто разработчику требуется реализовать функцию, которой нет в библиотеках *SimInTech*. Для этого можно использовать универсальный блок «Язык программирования» из библиотеки «Динамические» (Рис. 11.21). По умолчанию этот блок реализует функцию трансляции входа на выход.

В *SimInTech* используется внутренний упрощённый язык программирования, похожий на *Turbo Pascal*. В справочной документации есть достаточно подробный раздел, посвящённый этому языку, разобраться несложно.

Процесс создания нового блока рассмотрим на примере блока, моделирующего датчик угла.

Добавим блок «язык программирования» на схему чистого листа модели общего вида, откроем его и сделаем так, чтобы он транслировал угол поворота ротора двигателя в диапазоне 0...2 π в цифровой 12-битный код. На Рис. 11.22 приведена реализация этой функции.

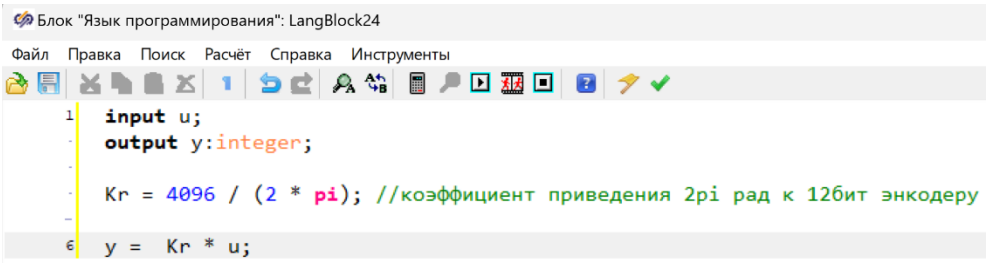


Рис. 11.22

Обратите внимание, что выход блока представляет собой целое число, так как мы принудительно определили выходную переменную ***y***, как ***integer***.

Теперь дополним скрипт блока тремя секциями, которые распознаются кодогенератором так, как описано далее.

- **Секция *header***

В секции описываются глобальные переменные и функции, которые могут использоваться нашей программой.

- **Секция *data***

Кодогенератор преобразует секцию в программу, которая будет исполняться всего один раз при инициализации микроконтроллера.

Секцию можно использовать, например, при создании блоков настройки периферии микроконтроллера.

- **Секция *code***

Программа этой секции будет вызываться каждый раз при обращении к нашему блоку. В неё можно поместить *C*-текст, который при кодогенерации добавится в соответствующую программу.

Допустим, в результате кодогенерации мы хотим получить программу опроса цифрового абсолютного энкодера ИДМ 20А-2-1-1, подключаемого к микроконтроллеру 1986ВЕ9х по шине *SSI*.

Тогда в секцию *code* помещается *C*-текст, который будет соответствовать запросу и чтению данных, полученных от датчика по шине *SSI*.

На Рис. 11.23 показан пример реализации этой задачи для микроконтроллера Миландр К1986В9*х*.

Блок "Язык программирования": LangBlock24

Файл Правка Поиск Расчёт Справка Инструменты

```
1   input u;
    output y:integer;

    Kr = 4096 / (2 * pi); //коэффициент приведения 2pi рад к 12бит энкодеру

    header = "";
    data = "";
    code = "
    /*Запросим у датчика текущее положение*/
10    MDR_SSP1->DR = 0;

    /*Ждем окончания передачи данных*/
    while ((MDR_SSP1->SR & SSP_FLAG_BSY) != 0) {}

    /*Получим значение кода угла поворота дачтика из приемного
    буфера SSP1 и запишем на выход блока*/
17  %out:0% = MDR_SSP1->DR & 0xfff;
    ";

20  y =  Kr * u;
```

Рис. 11.23

Обратите внимание на форму записи переменной %*out*:0%. Такая запись указывает кодогенератору, что нужно обратиться к выходу 0 блока (отсчёт нумерации входов и выходов блока идёт сверху вниз с нуля). Тип данных этой переменной определяется типом данных выхода блока, который мы задали выше. В нашем случае для выхода блока

будет создана переменная типа *integer*.

Для обращения к входу нашего блока нужно использовать запись вида *%input*: 0%.

Немного доработаем блок и добавим:

- вход калибровки (*null* – задание значения угла, принимаемого за нулевое положение ротора)
- свойство *SPI* с выбором значений *SSP*1 или *SSP*2 (выбор порта *SPI* для подключения датчика положения).

С учётом доработок перепишем программу блока (см. Рис. 11.24).

```
input y, null:integer;
output qr:integer;

initialization
const Kr = 4096 / (2 * pi); //коэффициент приведения 2pi рад к 12бит энкодеру
//выбор SPI задан перечислением, SPI1 соответствует число 0, SPI2 - 1
//temp будем использовать для подстановки нужных символов при кодогенерации
if SPI = 0 then temp = "SSP1";
if SPI = 1 then temp = "SSP2";

if not autotranslate then begin
  header = "
  /*Переменные блока датчика положения*/
  int16_t  %unikname%_int_angle = 0;
  ";
  data = "
  /*Тут можно было сделать что-то до начала работы программы.
    Написанный здесь код вызывается в процессе инициализации микроконтроллера
  */";

  code = "
  /*Запросим у датчика текущее положение*/
  MDR_"+temp+"->DR = 0;
  /*Ждем окончания передачи данных*/
  while ((MDR_"+temp+"->SR & SSP_FLAG_BSY) != 0) {}
  %unikname%_int_angle = (int16_t)(MDR_"+temp+"->DR & 0xfff);
  /*Откалибруем датчик*/
  %unikname%_int_angle = %unikname%_int_angle - (int16_t)(%input:1%);
  if (%unikname%_int_angle > 4095) %unikname%_int_angle = %unikname%_int_angle - 4095;
  if (%unikname%_int_angle < 0) %unikname%_int_angle = %unikname%_int_angle + 4095;
  /*Угол поворота*/
  %out:0% = (int32_t) %unikname%_int_angle;
  ";
end
end

qr =  Kr * y - null; //пересчитаем радианы в числ. знач. и откалибруем "ноль"
if qr > 4095 then qr = qr - 4095;
if qr < 0  then qr = qr + 4095;
```

Рис. 11.24

В секцию «*header*» мы добавили переменную, в которую будут сохраняться промежуточные значения вычислений. Странная, на первый взгляд, запись вида «*%unikname%_int_angle*» предписывает кодогенератору создать уникальный префикс для переменной «*int_angle*».

Это может понадобиться, чтобы избежать путаницы в случае использования в схеме проекта нескольких одинаковых блоков.

В секции *code* вместо «*+temp+*» кодогенератор подставит значение *SSP1* или *SSP2* (в зависимости от выбранного значения *SPI* в свойствах блока).

Внешний вид созданного Вами блока можно настроить.

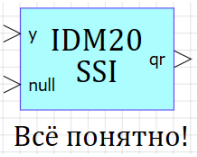
Для этого щёлкните правой кнопка мыши по блоку, в выпадающем меню выберите строку «Свойства объекта», а затем зайдите во вкладку «Общие».

Выбирая нужную строку в этой вкладке и изменяя соответствующий столбец «Значение», можно задать:

- имя блока
- размеры блока, цвет его фона и рамки и т.д.

После изменений не забывайте закрывать и сохранять сделанное.

Для того, чтобы написать ваш текст под блоком:



- щёлкните по блоку левой кнопкой мыши
- ещё раз щёлкните по выделенной под блоком области и в появившееся поле введите желаемый текст, например: «Всё понятно!»...

Или нет?

ПОДРОБНЕЕ о блоке «язык программирования» и функциях встроенного языка программирования можно почитать в справочном разделе документации к *SimInTech* (например, там вы сможете прочитать, зачем мы использовали в блоке секцию *initialization*).

ВАЖНО!!!

Если в секции *code* блока указать несуществующий адрес входного или выходного порта, то при кодогенерации проекта вы получите ошибку без указания точного места, где она допущена. Её поиск может занять много времени и сильно испортить настроение. Внимательно следите за адресацией портов!

ПРИМЕЧАНИЕ

Если в нашем, построенном на основе блока «Язык программирования», элементе нет прямого обращения к периферии микроконтроллера, то не обязательно в секции «*code*» писать текст на *C*. Можно использовать внутренний язык *SimInTech* и включить в свойствах блока опцию «*autotranslate*» - Да . В этом случае, при кодогенерации трансляция в *C* произойдёт автоматически.

12 Краткое послесловие

Предвидим, что при прочтении данной книжки у читателей возникнут вопросы.

Наша цель состояла в том, чтобы помочь вам правильно их сформулировать.

Ищите ответы – «и обрящете»!

Готовы, по возможности, способствовать.

Ю. Н. Калачёв Е. В. Окулов и весь коллектив «3В Сервис»

Список литературы

1. В. Милешин, Д. Овчинников, Управление транзисторными преобразователями электроэнергии, Техносфера, Москва 2011.
2. В. И. Преображенский, Полупроводниковые выпрямители, Второе издание, переработанное и дополненное, Москва, Энергоатомиздат 1986.
3. С. Тютюков, «Практические рекомендации по разработке печатных плат», интернет.
4. Ю. Н. Калачёв, Д. В. Самохвалов, Основы регулируемого электропривода, (Антиучебник, вариант 2, исправленный и дополненный), интернет.
5. А. С. Анучин, Системы управления электроприводов, Москва. Издательский дом МЭИ, 2015.
6. Б. А. Карташов, Е. А. Шабаев, О. С. Козлов, А. М. Щекатуров, Среда динамического моделирования технических систем *SimInTech*, ДМК Пресс, Москва, 2017.
7. Ю. Н. Калачёв, *SimInTech* – моделирование в электроприводе, ДМК Пресс, Москва 2021.

Книги издательства «ДМК Пресс» можно купить оптом и в розницу
на складе издательства по адресу:
Москва, ул. Электродная, д. 2, стр. 12, офис 7,
тел. **+7 (499) 322–19–38**,
а также заказать на сайте **www.dmkpress.com**
с доставкой в любой регион РФ.

Калачёв Ю. Н., Окулов Е. В.

Проектирование электроприводов (Руководство практика)

Главный редактор *Мовчан Д. А.*
Зам. главного редактора *Яценков В. С.*
editor@dmkpress.com

Формат 70×100 1/16.
Гарнитура «PT Serif». Печать цифровая.
Усл. печ. л. 5,77. Тираж 150 экз.

Веб-сайт издательства: *www.dmkpress.com*

Данная книга ориентирована на специалистов, занимающихся разработками электроприводов и преобразователей.

В ней кратко изложены основные принципы проектирования этих устройств, приведены структуры их аппаратной и программной частей.

На примере реального электропривода подробно рассмотрен процесс построения его модели с последующей кодогенерацией программы управляющего микроконтроллера.

ISBN 978-5-93700-375-1

9 785937 003751

Москва, 2025

@SIMINTECH