ШКОЛА MATLAB. Виртуальные лаборатории устройств силовой электроники в среде MATLAB+Simulink

Урок 22. Автоматическая генерация исполняемого кода средствами MATLAB+Simulink

Данная статья является продолжением материала по многовариантному синтезу асинхронного электропривода, рассмотренному в уроке 21. Затрагиваемые вопросы генерации исполняемого кода средствами MATLAB+Simulink относятся к контроллерам серии STM32F407/F417.

Сергей Герман–Галкин, д. т. н., профессор

germangalkin@gmail.com

Роман Гаврилов, к. т. н.

roman_gavrilov85@mail.ru

Никита Батлин

nikita.bat@mail.ru

Введение

Модельно-ориентированное проектирование (МОП) — современный, эффективный и экономически выгодный метод разработки систем управления, мехатронных и робототехнических систем, обработки сигналов и изображений, построения систем связи, создания встраиваемых систем и пр.

При таком подходе к разработке новых систем вместо физических прототипов и текстовых спецификаций применяется исполняемая модель. Эта модель используется на всех этапах разработки системы или объекта, по ней разработчик проводит имитационное моделирование как всей системы целиком, так и ее отдельных компонентов. Конечным результатом МОП является автоматическая генерация исполняемого кода для управляющего контроллера.

В МОП систем управления разработка происходит в пять этапов:

1. Построение модели объекта управления.
2. Разработка модели системы управления и синтез регуляторов.
3. Объединение модели объекта управления и системы управления, модельное исследование системы в целом, разработка управляющих алгоритмов.
4. Дискретизация системы управления.
5. Генерация исполняемого кода для выбранного типа контроллера.

Наиболее удобным инструментом для реализации МОП систем управления является среда MATLAB+Simulink. Имеющиеся в составе пакета Simulink приложения и библиотеки, как встроенные, так и подключаемые, дают разработчикам уникальные возможности по исследованию, проектированию и отладке систем управления реального времени.

Среда MATLAB+Simulink, помимо широких возможностей динамического моделирования сложных систем, состоящих из подсистем различной физической природы, предоставляет возможности по автоматической генерации С-кода. Генерация С-кода основана на применении пакетов расширений: Real-TimeWorkshop и Real-Time Workshop Embedded Code [10]. Сгенерированный исходный код может применяться для создания приложений, работающих в реальном времени вне среды MATLAB+Simulink, в том числе и для микропроцессоров встроенных систем.

В развитие технологий, изложенных в [3, 8, 9], предлагается методика генерации С-кода из модели Simulink, включающая следующие этапы:

1. Создание Simulink-модели системы управления в соответствии с заданным алгоритмом работы и ее отладка.
2. Доработка Simulink-модели с учетом дискретных преобразований сигналов.
3. Преобразование Simulink-модели в модель для расчетов с фиксированной точкой.
4. Генерация С-кода из модели Simulink.
5. Тестирование сгенерированного С-кода.
6. Интеграция сгенерированного С-кода в среду разработки процессора.

В данной статье рассмотрим процесс создания исполняемого кода для контроллеров серии STM32F407/417 с помощью набора библиотечных модулей Waijung Blockset [11]. Указанная библиотека находится в открытом доступе в Интернете и интегрируется в MATLAB, для этого необходимо выполнить следующие действия:

- Распаковать архив *waijung15_04a.rar* в выбранную директорию.
- Запустить MATLAB от имени администратора.
- В MATLAB в поле **Current Folder** выбрать папку с распакованным архивом *waijung15_04a*.
- Открыть файл *install_waijung.m* в **Editor MATLAB**.
- Запустить процесс установки (кнопка **Run**).

По завершении установки библиотека и компилятор автоматически будут интегрированы в Simulink и MATLAB.

Также необходимо с сайта st.com загрузить, а затем установить утилиту **STM32 ST-LINK Utility**, посредством которой осуществляется связь между компьютером и контроллером.

Разработка алгоритма модельно-ориентированного проектирования в среде MATLAB+Simulink

Интегрированный в библиотеку компилятор генерирует исполняемый код для контроллеров серии STM32F0, STM32F407/417 и nRF51. На рис. 1 представлен набор библиотечных модулей для работы с периферийными устройствами контроллера.

Для контроллера необходима библиотека **STM32F4 Target**, представленная на рис. 2.

На предыдущем уроке (урок 21) была создана имитационная модель асинхронного электропривода, структурная схема которого представлена на рис. 3.

Асинхронный электропривод состоит из асинхронного электродвигателя (М), инкрементного датчика контроля скорости (ДС), установленного на оси двигателя, силового полупроводникового преобразователя (VT), датчиков тока в фазах двигателя (ДТ), источника силового питания, состоящего из выпрямителя (VD) с фильтром (С), контроллера и ЭВМ. По последовательному каналу связи USART от ЭВМ на вход контроллера поступает задание по скорости, в ответ контроллер передает текущую скорость вала двигателя. Частота обмена информацией между контроллером и ЭВМ принята равной 100 Гц при скорости 115200 бит/с.

Имитационная модель электропривода в пакете Simulink с дискретными блоками в системе управления, которая реализуется в контроллере, представлена на рис. 4. Система управления представляет собой двухконтурную систему подчиненного регулирования, включающую контур регулирования тока и контур регулирования скорости, расчет которых был приведен на предыдущем уроке.

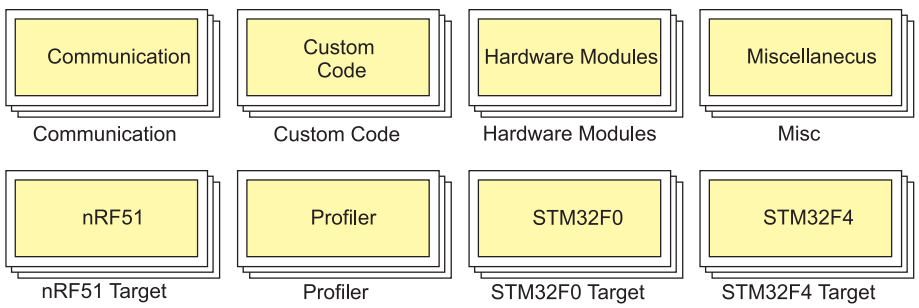

Рис. 1. Набор библиотечных модулей Waijung Blockset

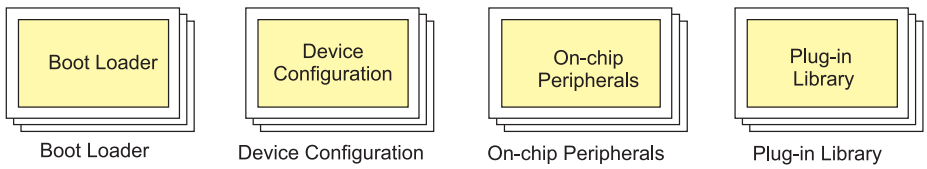

Рис. 2. Набор библиотечных модулей STM32F4 Target

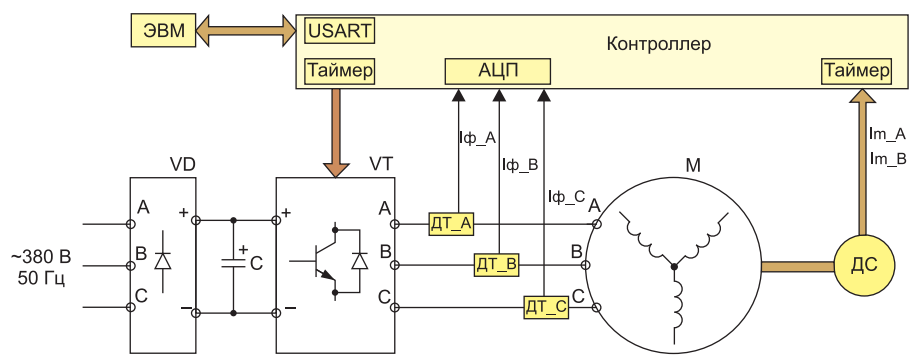

Рис. 3. Структурная схема асинхронного электропривода

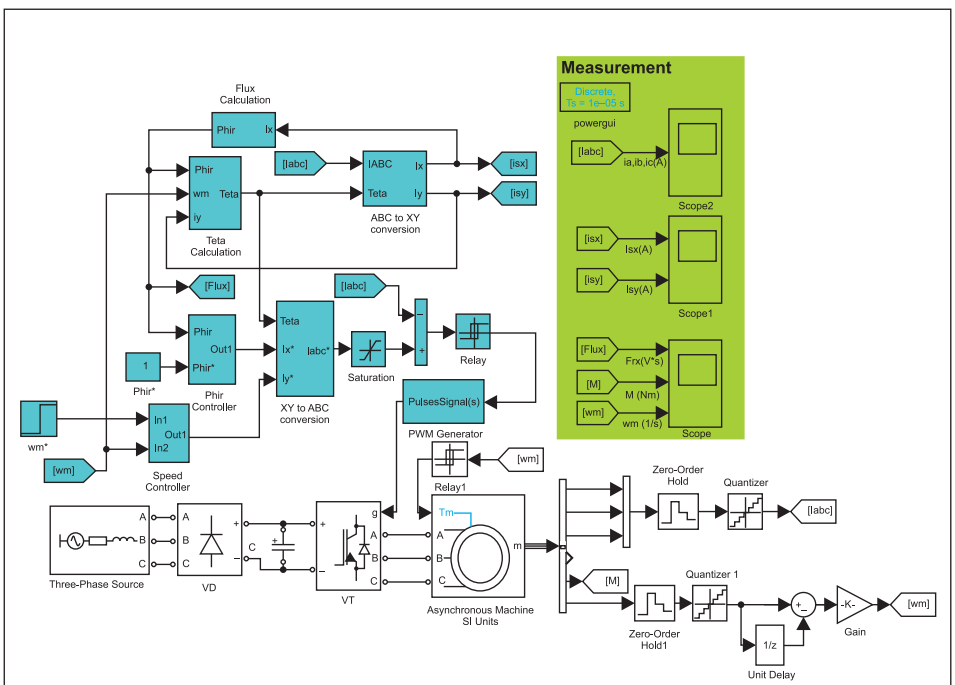

Рис. 4. Дискретная модель асинхронного электропривода

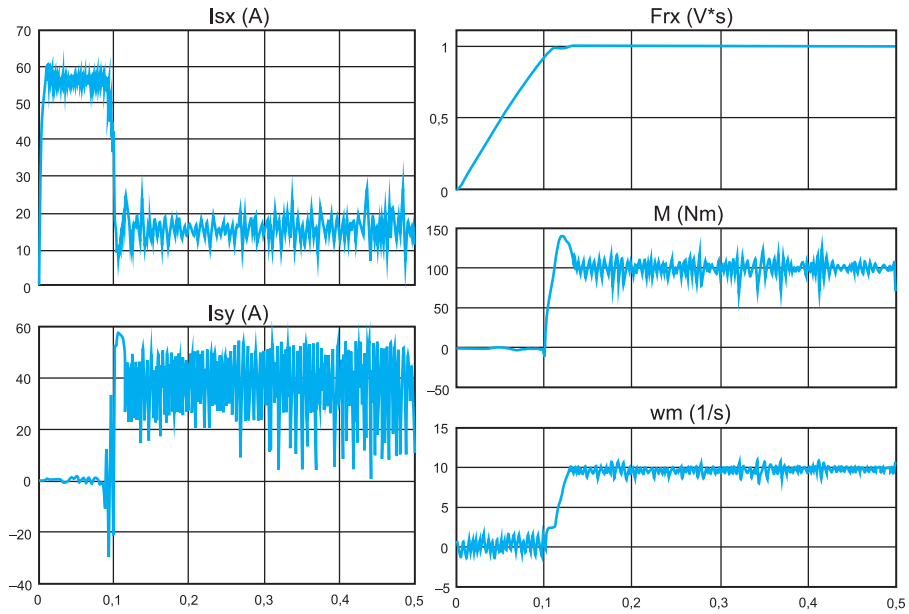

Рис. 5. Переходные процессы в асинхронном электроприводе

Дискретизация регуляторов потока и скорости [1, 2] проводилась исходя из периода программного цикла контроллера, равного 0,001 с. Графики переходных процессов представлены на рис. 5.

Создание исполняемого кода в среде Waijung Blockset

Ниже рассмотрен процесс по созданию исполняемого кода для электропривода с помощью элементов библиотеки **Waijung Blockset**. Разрабатываемая программа должна формировать импульсы управления силовыми ключами инвертора в функции заданной скорости вращения двигателя. При разработке исполняемого кода на базе рассмотренной модели необходимо перейти в библиотеку Waijung Blockset, далее на вкладку **STM32F4 Target→Device Configuration** и перенести блок **Target Setup** в окно модели. В окне настроек микроконтроллера, представленном на рис. 6, необходимо выполнить следующие действия.

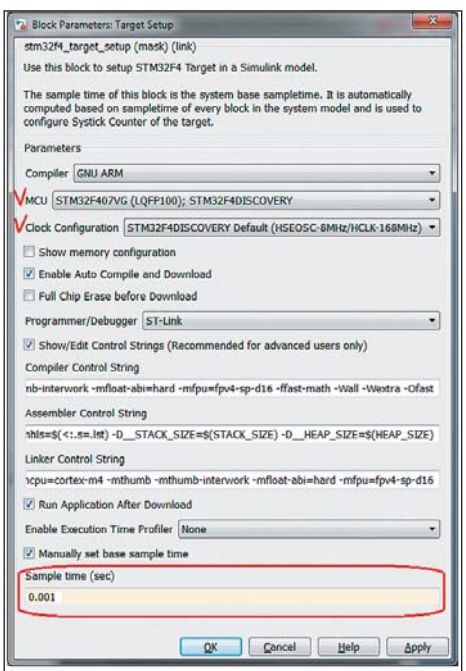

Рис. 6. Окно настройки параметров микроконтроллера

- В выпадающем списке MCU выбрать микроконтроллер STM32F4VG (LQFP100); STM32F4DISCOVERY. Данный контроллер будет использоваться в системе управления.
- В выпадающем списке **Clock Configuration** выбрать STM32F4DISCOVERY Default (HSEOSC-8MHz/HCLK-168MHz), задать тактовую частоту процессора 168 МГц.
- Установить галочку ***Manually set base simple time*** и в поле ***Simple time (sec)*** ввести требуемую величину периода программного цикла микроконтроллера (в данном случае 0,001 с, что соответствует 1000 Гц).

Далее нужно сохранить модель.

Примечание: В названии модели и в пути, где будет сохранена модель, категорически запрещается использовать русские символы и буквы.

Шаг 2. Настройка каналов аналого-цифрового преобразователя (АЦП), которые будут осуществлять чтение текущих токов в обмотках двигателя.

Для этого необходимо перейти в раздел библиотеки **STM32F4 Target→On-chip Peripherals→ADC** и перенести блок **Regular ADC** в окно модели. Двойной щелчок левой кнопки мышки по блоку открывает окно настроек АЦП микроконтроллера, представленное на рис. 7.

В микроконтроллере STM32F4DISCOVERY имеется два 16-канальных АЦП (ADC Module 1 и 2) и один восьмиканальный АЦП (ADC Module 3). Все АЦП могут опрашивать входы, перечисленные в окне настроек, т. о. контроллер тремя АЦП может опрашивать до 16 каналов.

В открывшем окне необходимо выполнить следующие операции:

1. Из выпадающего списка ADC Module выбрать АЦП-модуль, который будет участвовать в опросе датчиков тока плат управления (в данном случае использован первый модуль).
2. Указать входы, задействованные в управлении (в данном случае AN11 (PC1), AN12 (PC2) и AN13 (PC3)).
3. Задать период опроса каналов АЦП. Т. к. данный модуль АЦП участвует в работе контура тока, то частота опроса датчиков тока должна совпадать с частотой программного цикла и быть равной 1 кГц (***Simple time 0,001 sec***).
4. Осуществить масштабирование.

Выходом АЦП является цифровой код, пропорциональной входному напряжению. Для перехода от цифрового кода к напряжению нужно выполнить преобразование:

$$U = \frac{AN_i}{4096} \times 3{,}3,$$

где AN_i — выходной сигнал с АЦП.

Таким образом, после каждого выхода АЦП нужно поставить элемент ***Gain*** из библиотеки **Simulink→MathOperations.**

Ток в обмотках двигателя протекает в двух направлениях, следовательно, в схеме измерения тока введено смещение нуля датчика тока на уровне 2,5 В. Таким образом, максимальное значение для положительного направления тока равно 5 В, максимальное значение для отрицательного тока — 0 В. После компенсации смещения нуля и масштабирования напряжения с АЦП

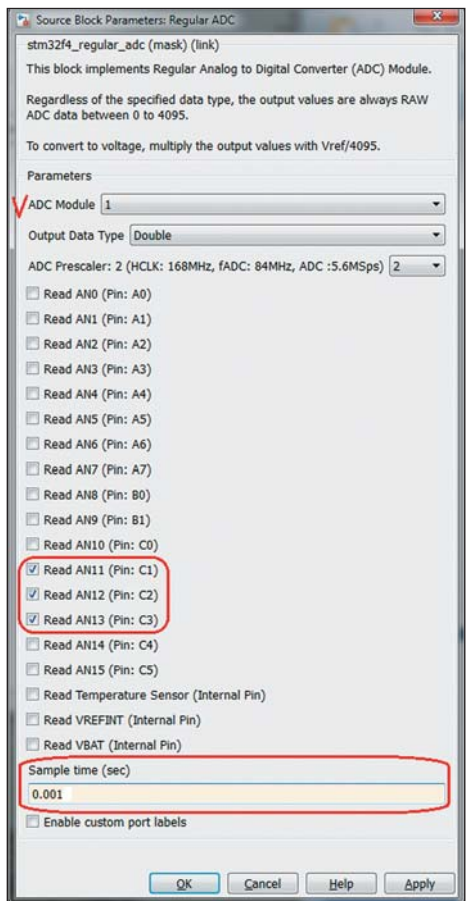

Рис. 7. Окно настроек АЦП

(коэффициент Km) итоговая модель измерения тока примет вид, представленный на рис. 8.

Шаг 3. Настройка таймера, отвечающего за обработку инкрементного датчика положения.

Измерение текущего угла и скорости двигателей часто осуществляется при помощи инкрементных энкодеров фотоэлектрического типа. В связи с чем в библиотеке **Waijung Blockset** присутствует блок обработки сигналов энкодера. Инкрементный энкодер формирует импульсы при вращении вала датчика. Этот тип энкодеров, в отличие от абсолютных энкодеров, работающих в коде Грея, не формирует выходные импульсы, когда его вал находится в покое.

Инкрементный оптический энкодер состоит из следующих компонентов: источника света, диска с метками, фототранзисторной сборки и схемы обработки сигнала. Диск инкрементного энкодера разделен на точно позиционированные прозрачные и непрозрачные области (метки). Количество отметок определяет количество импульсов за один оборот. К примеру, если диск поделен на 1000 меток, тогда за 250 импульсов вал должен повернуться на 90°.

Используемый в данном случае инкрементный 5-В датчик CFS50-AFV11X08 имеет на выходе разностные сигналы A, $\overline{A}$, B, $\overline{B}$, N и $\overline{N}$ и 2048 отчетов на оборот, где сигналы $\overline{A}$, $\overline{B}$ и $\overline{N}$ являются инверсными сигналами по отношению к A, B и N. Сигналы A и B сдвинуты по фазе на 90° относительно друг друга. У инкрементного 5-В датчика дорожки A и B используются для счета. Дорожка N используется при соответствующей параметризации для установки счетчика на загружаемое значение. Датчики с этими шестью сигналами называются симметричными. На рис. 9 показана временная последовательность этих сигналов.

Шаг 4. Анализ импульсов.

DSP-контроллеры могут подсчитывать фронты сигналов посредством таймеров. Обычно анализируется фронт на A ($\overline{A}$) (однократный анализ). Для получения более высокого разрешения, путем параметризации, может быть использован однократный, двойной или четырехкратный анализ.

Множественный анализ возможен только у инкрементных 5-В датчиков с сигналами A и B, сдвинутыми на 90°, или у инкрементных 24-В датчиков с сигналами A* и B*, сдвинутыми на 90°.

Однократный анализ означает, что анализируется только один фронт A; импульсы прямого счета регистрируются при нарастающем фронте на A и низком уровне сигнала на B, а импульсы обратного счета регистрируются при падающем фронте на A и низком уровне сигнала на B. На рис. 10 показан однократный анализ сигналов.

Двойной анализ означает, что анализируются нарастающий и падающий фронт сигнала A; генерируются ли импульсы прямого или обратного счета, зависит от уровня сигнала B. На рис. 11 показан двойной анализ сигналов.

Четырехкратный анализ означает, что анализируются нарастающий и падающий фронты A и B; генерируются ли импульсы прямого или обратного счета, зависит от уровней сигналов A и B. На рис. 12 показан четырехкратный анализ сигналов.

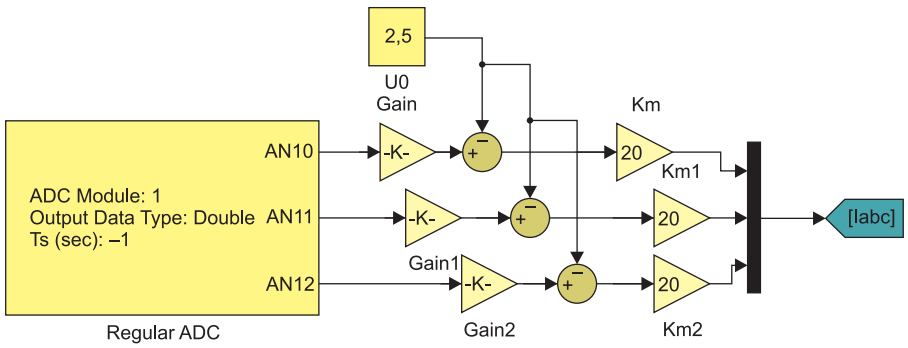

Рис. 8. Модель измерения тока

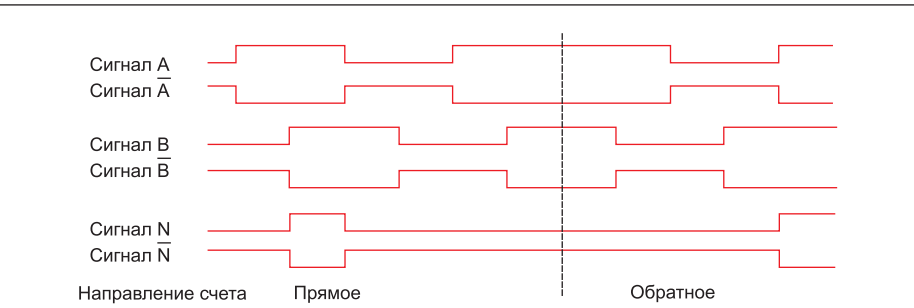

Рис. 9. Временная последовательность импульсов инкрементного датчика

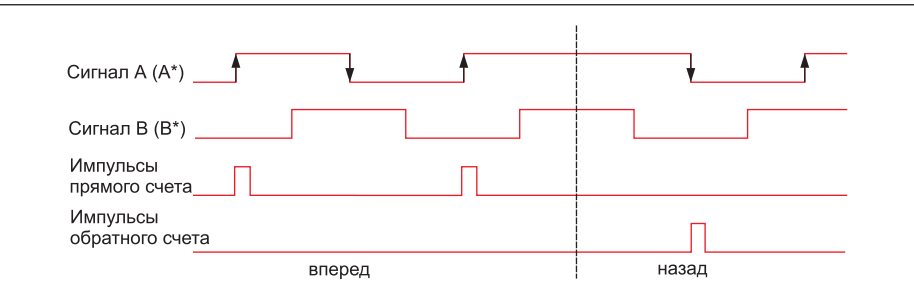

Рис. 10. Однократный анализ сигналов инкрементного датчика

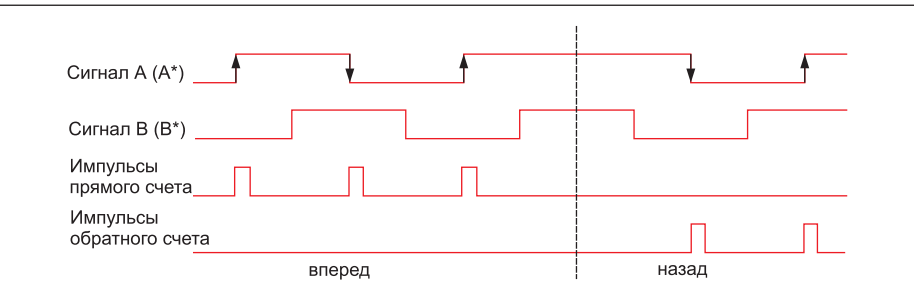

Рис. 11. Двукратный анализ сигналов инкрементного датчика

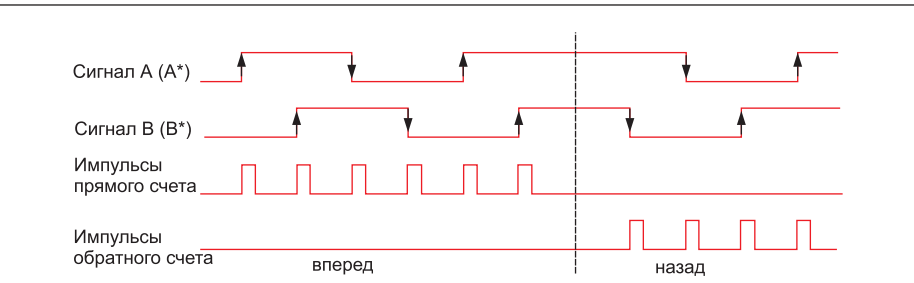

Рис. 12. Четырехкратный анализ сигналов инкрементного датчика

В DSP-контроллерах STM32 используется четырехкратный анализ сигналов, что позволяет программно увеличить точность определения положения в четыре раза.

Датчик обратной связи по скорости (инкрементный оптический энкодер) имеет 2048 отчетов (меток) на оборот. Как правило, блоки, реализующие обработку сигналов с инкрементных датчиков, строятся на таймерах. Для настройки таймера, реализующего алгоритм обработки датчика оси, необходимо перейти в раздел библиотеки **STM32F4 Target→On-chip Peripherals→TIM** и перенести блок **Encoder Read** в окно модели. Двойной щелчок левой кнопки мыши по блоку открывает окно настроек АЦП микроконтроллера, представленное на рис. 13.

В этом элементе необходимо задать два параметра:

- количество отчетов энкодера на оборот (***Number of pulses generated per revolution***), в нашем случае 2048 отчетов;
- период опроса таймера (***Sample time (sec)***), т. е. интервал времени, через который программа будет получать информацию от датчика.

Выходными сигналами с блока **Encoder Read** будут: направление вращения (***Direction***), принимающее два значения («+1» и «–1»), а также число пройденных отчетов (***Position (count)***). Следует отметить, что при первом включении инкрементного датчика его текущее положение будет принято нулевым, и уже от него будет производиться отсчет пройденных меток. Таким образом, без дополнительных программных мер (выставка нулевого положения датчика относительно ротора двигателя) при каждом включении датчик будет принимать новое нулевое положение за исходное. На рис. 14 представлена модель расчета скорости.

Подсистема **Speed Estimate** представлена на рис. 15.

Выходом ***Position (count)*** таймера является число отчетов, которое прошел датчик за интервал времени измерения, программно умноженное на 4 (программное увеличение точности измерений). В блоках **MATLAB Function** [4] из библиотеки **Simulink→User-Defined Functions** прописываются функции. При открытии данного блока запускается **Matlab Editor**, в котором необходимо набрать текст программы. Функция ***Encoder2Angel*** переводит количество отчетов, выданных таймером, в угол в радианах. Текст программы имеет вид:

```
function Angle = Count2Angle(Count)
Temp = cast(Count,'double');
Angle = Temp/(4*4096);
Angle = mod(Angle,2*pi);
end
```

Функция ***Angel2Speed*** производит расчет скорости прямым методом Эйлера с про-

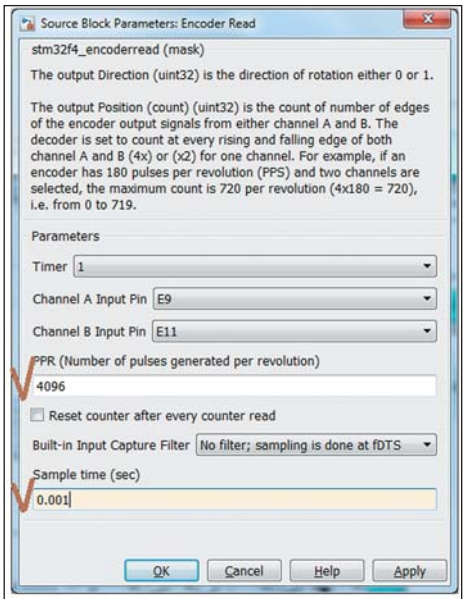

Рис. 13. Окно настроек таймера

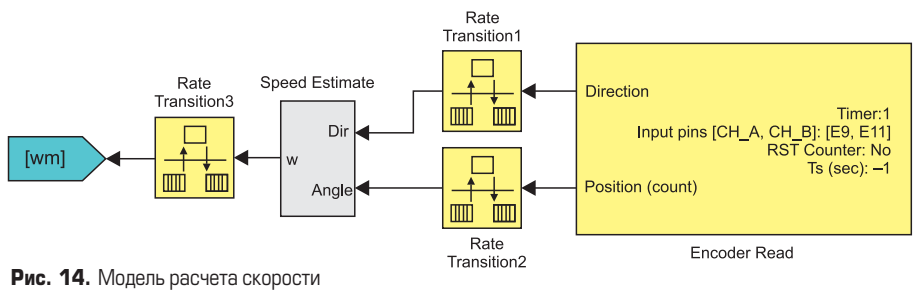

Рис. 14. Модель расчета скорости

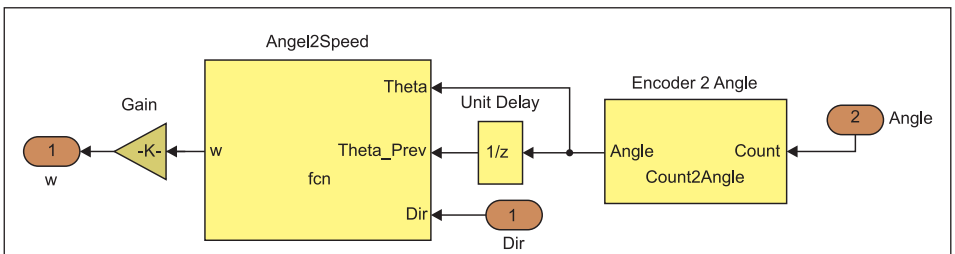

Рис. 15. Подсистема Speed Estimate

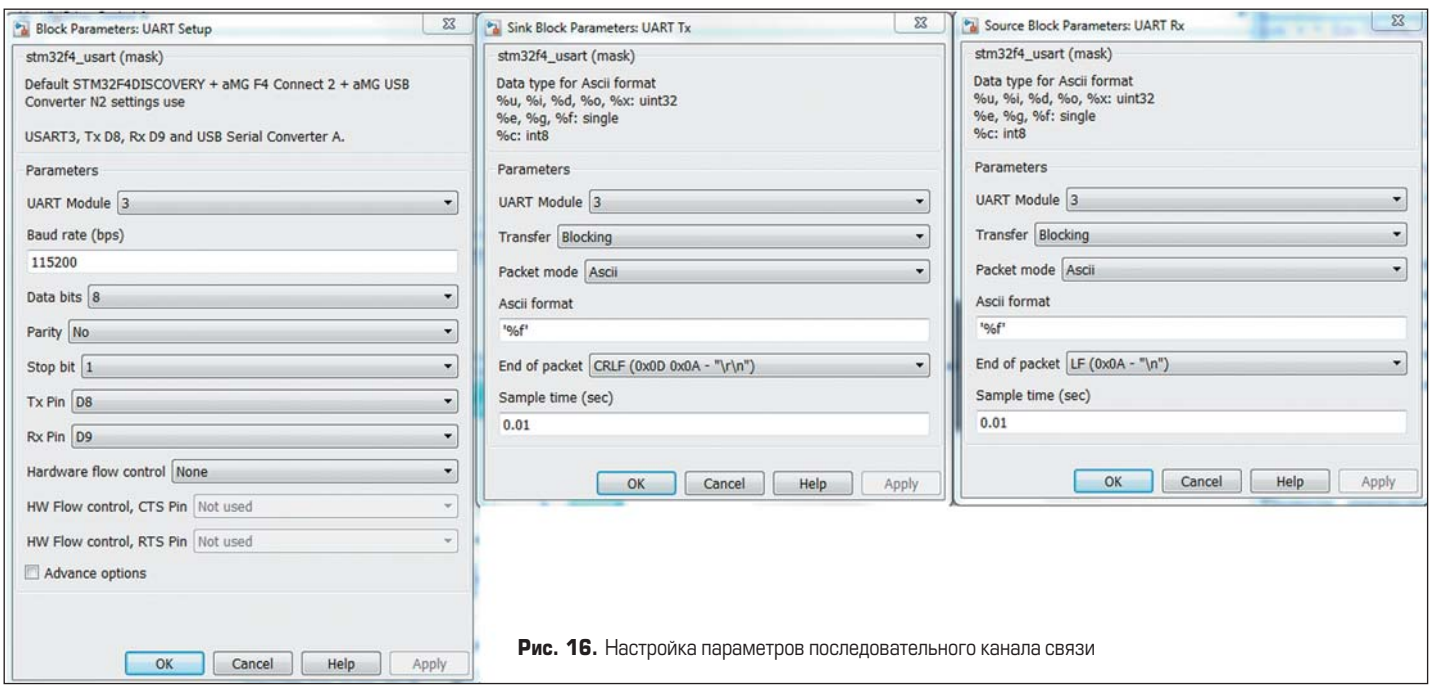

Рис. 16. Настройка параметров последовательного канала связи

пуском моментов времени, когда показания датчика обнуляются. Текст программы имеет вид:

```
function w = fcn(Theta,Theta_Prev,Dir)
% if Dir = 1 -> downcounting, Dir = 0 -> upcounting
if (Theta< Theta_Prev)&& (Dir == 0)
% Positive Roll Over
w = Theta+2*pi-Theta_Prev;
elseif (Theta > Theta_Prev)&& (Dir == 1)
% Underflow
w = Theta-2*pi-Theta_Prev;
else
w = Theta-Theta_Prev;
end
```

Коэффициент ***Gain*** пропорционален периоду программного цикла контроллера и определяется как 1/Ts, где Ts — период программного цикла. В данном случае Gain = 1000.

Блок **Unit Delay** из библиотеки **Simulink→Discrete** задает задержку на один шаг расчета, для чего в открывшемся окне свойств блока в поле ***Sample time*** необходимо записать период программного цикла контроллера (0,001).

Шаг 5. Настройка канала последовательной передачи данных RS232.

Для настройки канала передачи данных необходимо перейти в раздел библиотеки **STM32F4 Target→On-chip Peripherals→UART** и перенести все находящиеся там блоки в окно модели. На рис. 16 представлены окна настройки параметров канала связи (**UART Setup**), передатчика (**UART Tx**) и приемника (**UART Rx**).

В окне **UART Setup** в выпадающем меню **UART Module** необходимо указать номер канала последовательного порта контроллера, по которому будет производиться обмен информацией (в данном случае будем использовать порт 3). В поле **Baud rate** требуется указать скорость передачи данных, а остальные поля оставим по умолчанию. Передача данных из контроллера будет производиться через порт D8, прием данных контроллером — через порт D9.

В окне **UART Tx** в выпадающем меню **UART Module** укажем номер канала контроллера, по которому будет производиться передача данных (механическая скорость ротора). В выпадающем меню выбора режима работы передатчика **Transfer** выберем режим **Blocking** (передача данных после приема). В выпадающем меню **Packet mode** (формат передачи данных) выберем ASCII. В поле **ASCII format** укажем формат передаваемых данных, в нашем случае это символьный ('%f'). В выпадающем меню **End of packet** нужно указать форму окончания пакета, в нашем случае — перевод курсора, перевод строки («\r\n»). В поле **Simple time** зададим период передачи данных.

В окне **UART Rx** в выпадающем меню **UART Module** указывается номер канала контроллера, по которому будет производиться прием данных (заданная скорость). В выпадающем меню выбора режима работы передатчика **Transfer** выберем режим **Blocking** (передача данных после приема). В выпадающем меню **Packet mode** (формат передачи данных) выберем ASCII. В поле **ASCII format** укажем формат передаваемых данных, в нашем случае это символьный ('%f'). В выпадающем меню **End of packet** нужно указать форму окончания пакета, в нашем случае перевод строки («\n»). В поле **Simple time** зададим период передачи данных.

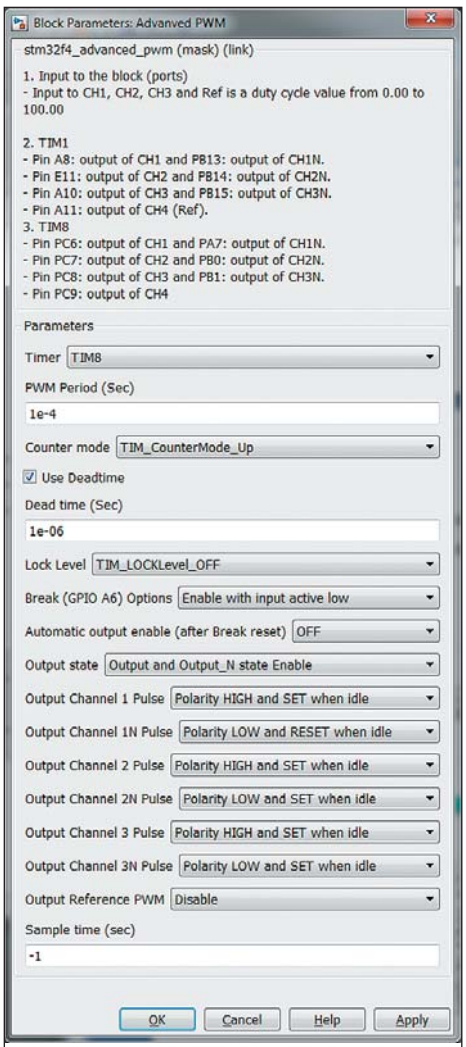

Рис. 17. Окно настройки параметров ШИМ

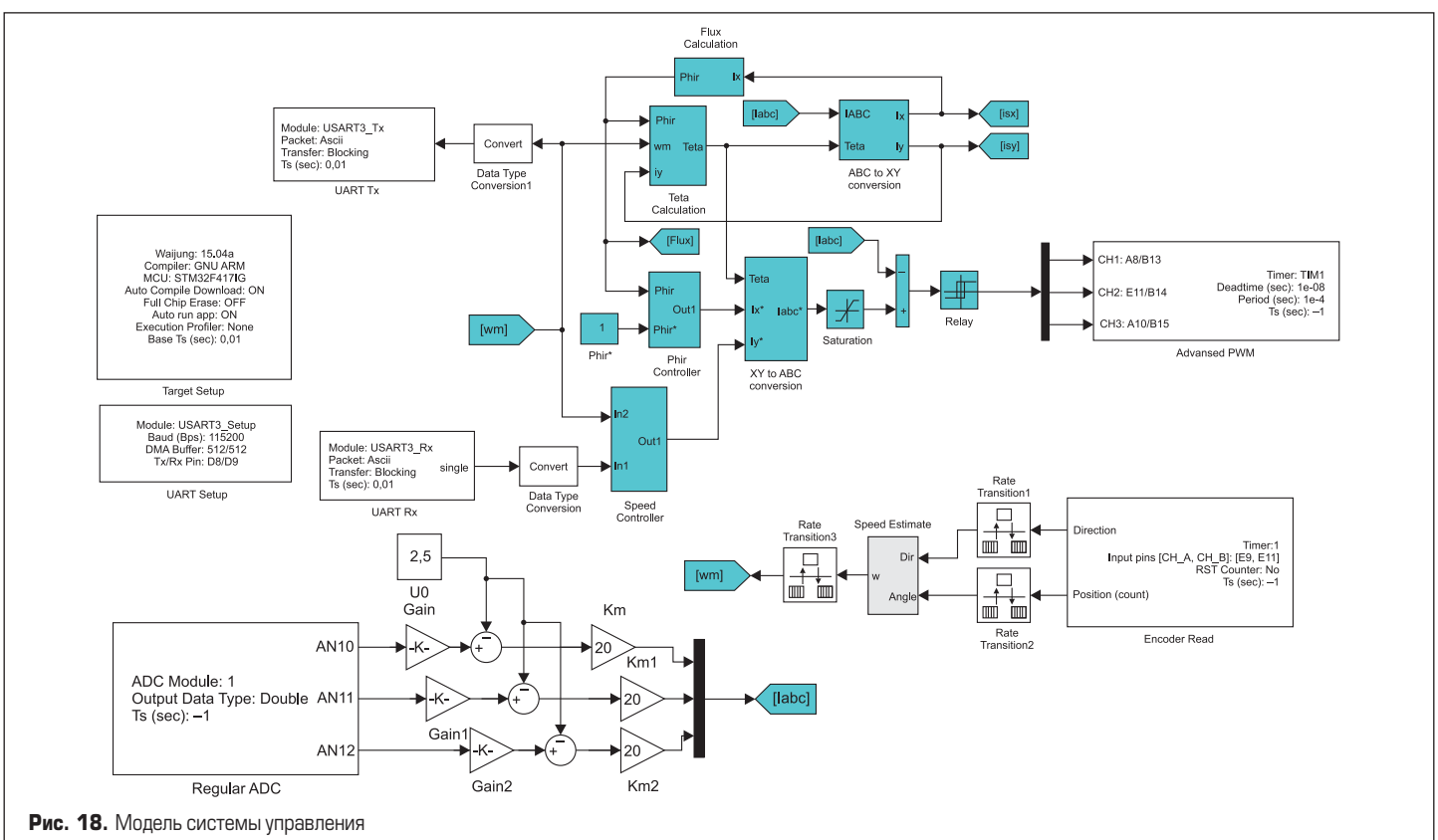

Рис. 18. Модель системы управления

Так как в приемнике и передатчике канала связи формат данных указан как символьный, следовательно, необходима конвертация полученных/принятых данных в требуемый формат посредством блока **Data Type Conversion** из библиотеки **Commonly Used Blocks Simulink**. В окне свойств блока **Data Type Conversion** в выпадающем меню **Output data type** следует выбрать ***Inherit: Inherit via back propagation***.

Шаг 6. Настройка широтно-импульсной модуляции.

Управление фазными токами в обмотках двигателя осуществляется посредством широтно-импульсной модуляции (ШИМ) режима работы силовых ключей инвертора. Для настройки ШИМ необходимо перейти в раздел библиотеки **STM32F4 Target→On-chip Peripherals→TIM** и перенести блок **Advanved PWM** в окно модели. На рис. 17 представлено окно настройки параметров ШИМ.

В выпадающем меню **Timer** необходимо выбрать **TIM8**, т. к. первый таймер используется для обработки сигналов инкрементного датчика. В окне **PWM Period** зададим требуемую частоту ШИМ. В нашем случае частота ШИМ 10 кГц, что соответствует периоду 1e–4c. В выпадающем меню **Output Channel 2 Pulse** выберем значение ***Polarity HIGH and SET when idle;*** в выпадающем меню **Output Channel 2N Pulse** — значение ***Polarity LOW and SET when idle***; в выпадающем меню **Output Channel 3 Pulse** — значение ***Polarity HIGH and SET when idle***; в выпадающем меню **Output Channel 3N Pulse** — значение **Polarity LOW and SET when idle**.

Итоговая модель системы управления представлена на рис. 18.

Теперь, при нажатии кнопки **Build Model**, будет запущен процесс автоматической генерации исполняемого кода для микроконтроллера, его компиляции и записи программы в контроллер (если он подключен к компьютеру), представленный на рис. 19.

По окончании процесса компиляции сгенерированный код будет записан в управляющий контроллер и сохранен в папке проекта.

Заключение

На примере асинхронного электропривода рассмотрены этапы МОП, включающие в себя переоборудование непрерывной системы управления в цифровой аналог в пакете Simulink. С использованием внешней библиотеки **Waijung** и средств среды MATLAB+Simulink цифровая модель системы управления привязана к конкретному контроллеру и периферийным устройствам (АЦП, датчики и пр.). Осуществлена компиляция кода и загрузка его в контроллер, работающий в режиме реального времени.

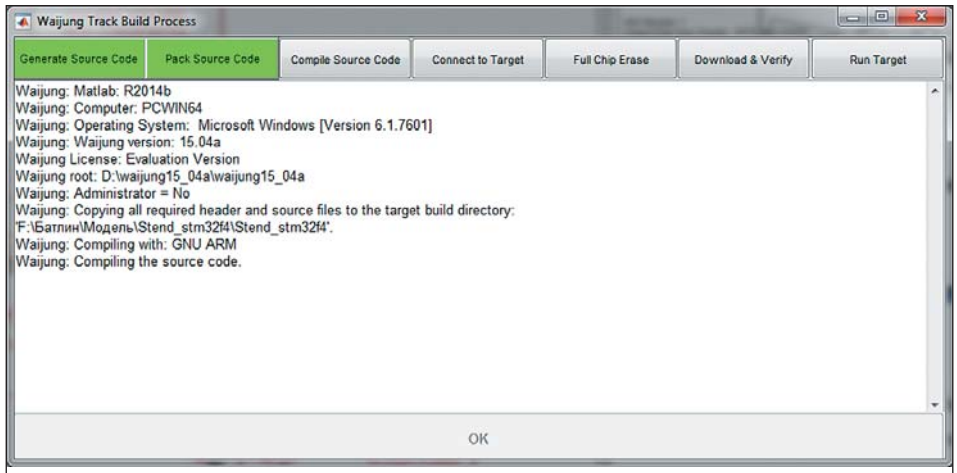

Рис. 19. Генерация исполняемого кода

МОП является простым и удобным инструментарием, позволяющим разработчику в сжатые сроки разработать структуру системы управления, провести ее модельное тестирование и генерировать по ней исполняемый код для контроллера.

Литература

1. Герман-Галкин С. Г. MATLAB & SIMULINK. Проектирование мехатронных систем на ПК. Уч. пособие для вузов. СПб.: «Корона-Век», 2008.
2. Герман-Галкин С. Г. ШКОЛА MATLAB. Виртуальные лаборатории устройств силовой электроники в среде MATLAB+Simulink. Урок 21. Модельный многовариантный синтез асинхронного электропривода // Силовая электроника. 2016. № 4.
3. С. Г. Герман-Галкин, Р. С. Гаврилов, Ю. Н. Мустафаев. Структурные и имитационные модели в модельно-ориентированном проектировании вентильного электропривода для ОПУ // Мехатроника, автоматизация и управление. 2017. № 1.
4. Дьяконов В., Круглов В. Математические пакеты расширения MATLAB. Специальный справочник. СПб.: Питер, 2001.
5. Иванов И. И., Петров С. П. Управление сложными системами. М.: Наука, 1998.
6. Петров С. П. Параметрическая идентификация пространственно распределенных динамических систем // Вопросы управления. 2000. № 2.
7. Ефремов А. А., Зенков С. М. Модельно-ориентированное проектирование для решения задач автоматизации. Тезисы докладов международной научно-практической конференции «Передовые информационные технологии, средства и системы автоматизации и их внедрение на российских предприятиях». Москва, 4–8 апреля 2011 г.
8. Понятский В. М., Гусев А. В., Фимушкин В. С. и др. Компьютерные технологии проектирования приводов летательного аппарата с использованием САПР SolidWorks и MATLAB. Всероссийская научно-техническая конференция «Фундаментальные основы баллистического проектирования». Санкт-Петербург, 28 июня–2 июля 2010 г. Сборник материалов. Т. 2 / Под ред. д. т. н., проф. Кэрта Б. Э. СПб.: Балт. гос. техн. ун-т, 2010.
9. Понятский В. М. Кушников Д. В., Федорищева В. Г. Автоматизированная технология генерации программы в среде MATLAB для реализации алгоритмов управления рулевого привода / Изв. ТулГУ. Сер. «Проблемы проектирования и производства систем и комплексов». Тула: ТулГУ, 2010. Вып. 11.
10. Sidoroff J. K. Control of technical systems // Archives of Computer Science. 1996. Vol. 28. № 5.
11. www.aimagin.com/download/