Федеральное государственное автономное образовательное учреждение высшего образования «Санкт-Петербургский государственный электротехнический университет «ЛЭТИ» им. В.И. Ульянова (Ленина)»

Описание функциональных характеристик программного обеспечения и информация, необходимая для установки и эксплуатации ПО «Программа для векторной широтно-импульсной модуляции для системы управления трехфазной электрической машиной с использованием ковариантных и контравариантных координат изображающего вектора»

Санкт-Петербург

2022

Оглавление

Описание функциональных характеристик программного обеспечения

Программа предназначена для векторной широтно-импульсной модуляции для открытой системы управления трехфазной электрической машиной с использованием ковариантных и контравариантных координат изображающего вектора. Существенное отличие от существующих программ векторной широтно-импульсной модуляции в том что при вычислении по заданному вектору напряжения не делается переход к декартовой системе координат d-,q- а расчет идет непосредственно в косоугольной системе координат с использованием ковариантных и контравариантных координат изображающего вектора. Это позволяет сократить объем вычислений.

Программа осуществляет управление трехфазным асинхронным двигателем, например, УАД-12. Управление двигателем осуществляется автономным инвертором напряжения выполненным по мостовой схеме на IGBT-транзисторах. Управление IGBT-транзисторами производится векторной широтно-импульсной модуляцией. Не предполагается наличие обратной связи с двигателем. Алгоритм управления использует ковариантные и контравариантые координаты, которые упрощают управление электрической машиной так как отсутствуют лишние переходы в декартову систему d, q и обратно.

Информация, необходимая для установки и эксплуатации ПО

Используем микроконтроллер STM32F103C8 и фреймворк stm32duino, который позволяет заливать прошивку через USB, без использования внешних компонентов типа ST-Link или USB-UART переходника.

В микроконтроллерах AVR ATMega под бутлоадер можно зарезервировать некоторое количество памяти ближе к концу флеша. С помощью fuse битов можно регулировать с какого адреса будет стартовать программа. Если бутлоадера нет — программа стартует с адреса 0x0000. Если бутлоадер есть — он запускается с некоторого другого адреса (скажем, в ATMega32 с 0x3C00, если размер бутлоадера выбран 2к).

Бутлоадер передает управление основной программе с адреса 0x0000. Т.е. программа всегда стартует с адреса 0x0000. Компилятор и линковщик работают с учетом того, что код будет находится в начале адресного пространства.

В микроконтроллерах STM32 все программы стартуют с адреса 0x0800000. Бутлоадер не является чем-то таким особенным. Это такая же программа, которая стартует с того же самого начального адреса. В процессе работы бутлоадер может принять прошивку (через USB или UART, считать с флешки, принять со спутника, достать из подпространства, whatever...) и записать ее по адресам выше чем находится сам загрузчик. Ну и, конечно же, в конце своей работы передать управление основной программе.

При компиляции прошивки нужно знать куда бутлоадер запишет прошивку и соответствующим образом скорректировать адреса.

Инструкция

Скачать github.com/rogerclarkmelbourne/STM32duino-bootloader. В директории STM32F1\binaries уже есть пакет скомпилированых бутлоадеров под разные платы. Индекс в конце названия файла указывает куда подключен светодиод. В случае платы где светодиод подключен к пину C13, используется файл generic_boot20_pc13.bin.

Прошиваем согласно инструкции. Тут понадобится USB-UART переходник, но можно и с помощью отладчика.

Теперь микроконтроллер готов прошиваться через USB загрузчик. Для того, чтобы подправить прошивку нужно сделать 2 вещи:

Указать линкеру стартовый адрес. В CooCox это делается в настройках проекта, вкладка Link, раздел Memory Areas, Адрес IROM1 Start Address. Бутлоадер занимает первые 8 килобайт, значит стартовый адрес прошивки будет 0x0800000 + 0x2000 = 0x08002000. Поле Size стоит уменьшить на 8к.

Вначале программы перед инициализацией периферии сделать вызов NVIC_SetVectorTable(NVIC_VectTab_FLASH, 0x2000);

UPDATE 17.05.2018: В современной версии STM32Cube функции NVIC_SetVectorTable() нет. Вместо этого можно в файле system_stm32f1xx.c (или аналогичного для другого микроконтроллера) исправить дефайн VECT_TAB_OFFSET

Заливатор прошивки можно взять из проекта stm32duino. В директории tools ищите скрипт под называнием maple_upload. Используется виндовая версия — maple_upload.bat.

Запуск

"maple_upload.bat" COM20 2 1EAF:0003 "Path\To\Firmware.bin"

Вместо COM20 нужно подставить свой порт куда прицепился микроконтроллер.

Путь к прошивке нужно указывать полностью.

1. EAF:0003 — это VID и PID

2 — это параметр AltID, который указывает что прошивку нужно заливать по адресу 0x08002000

Перед тем как заливать прошивку нужно запустить бутлоадер. Самый простой способ — нажать кнопку ресет. После этого запустится загрузчик и несколько секунд будет ждать прошивку. Если в этот момент никто не запустил maple_upload, загрузчик передаст управление основной прошивке.

Чтобы не нажимать каждый раз ресет, платы основанные на libmaple/stm32duino используют трюк. Они слушают usb serial порт. Если там возникает сигнал DTR и передается ключевая последовательность байт, то микроконтроллер перегружается в бутлоадер. Смотреть в функцию rxHook().

Из-за этого может возникнуть неудобство. Если микроконтроллер зависает, то он уже не слушает порт. Следовательно он не может услышать ключевую последовательность и перегрузиться в бутлоадер. Тогда нужно нажимать ресет