Возбуждение трехфазного двигателя с постоянными магнитами синусоидальным напряжением, используя ATmega48/88/168

Особенности

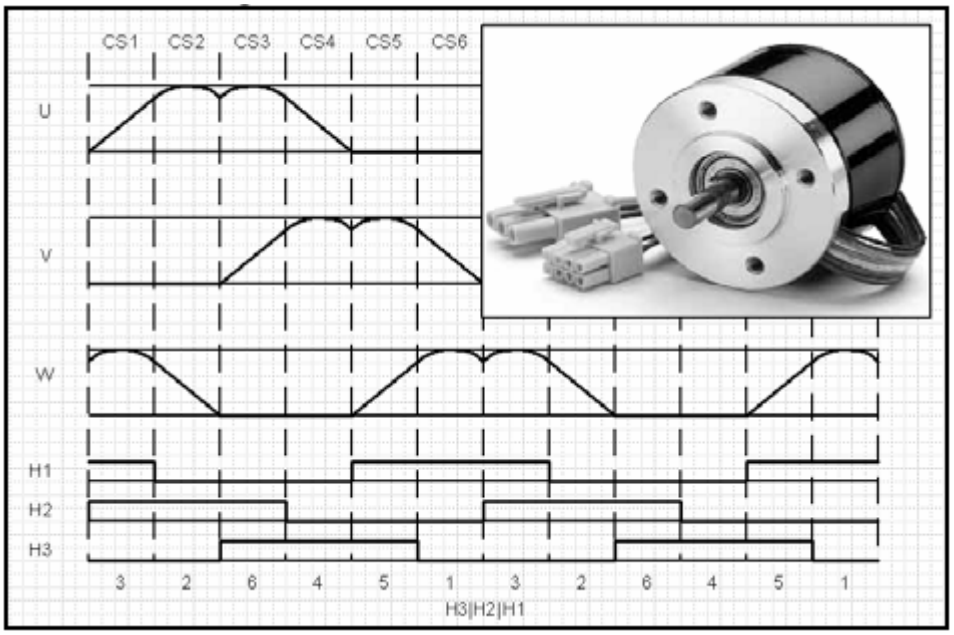
- Трехфазное синусоидальное напряжение
  - 192 шага на электрический оборот
  - 8-битная регулировка амплитуды.
- Программная генерация мертвой зоны.
- Управление от датчиков холла
- Управление скорости путем масштабирование амплитуды синусоиды во время выполнения
  – регулировка скорости по аналоговому сигналу
- Автоматическая синхронизация при старте с вращающимся двигателем
- Безопасный запуск, используя блочную коммутацию в начале коммутации
- Управление направлением вращения по цифровому сигналу
- Безопасный останов и изменения направления
  - активное торможение или движение накатом до остановки
- Опережающий угол коммутации, корректируемый во время выполнения
- Вывод сигнала противовращения
- Вывод сигнала тахогенератора

1 Введение

Эта Рекомендация по Применению (РпП) описывает реализацию запуска трехфазных безщеточных двигателей постоянного тока с датчиками холла синусоидальным напряжением.

Реализация может быть легко изменена, чтобы использовать волны другой формы для возбуждения двигателя, типа синусоиды с третьей гармоникой.

Рисунок 1-2. Возбуждение безщеточного двигателя постоянного тока с датчиками холла.



2 Теоретические предпосылки

В большинстве литературных источников, двигатели с постоянными магнитами разделены на две категории, на основании формы обратной ЭДС (напряжение, наведенное в катушках, когда двигатель вращается). Обратная ЭДС может быть или трапециевидной или синусоидальной формы. Хотя терминология в литературе иногда противоречива, большинство, кажется, соглашается, что безщеточный двигатель постоянного тока (BLDC) имеет трапециевидную обратную ЭДС, а синхронный двигатель с постоянными магнитами (PMSM) имеет синусоидальную обратную ЭДС. И BLDC и PMSM двигатели могут возбуждаться синусоидальными токами, так что здесь не будет сделано различий между ними. Вместо этого, они оба будут упоминаться как двигатель с постоянными магнитами, или PMM.

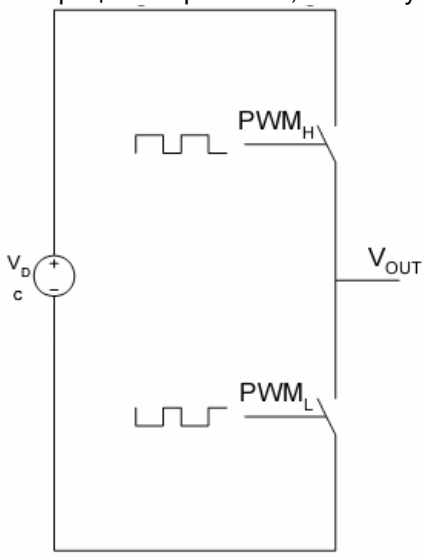
3 Исполнение на AVR ®

Эта РпП описывает, как управлять трехфазным PMM с помощью синусоидальных напряжений. Пример программы может также использоваться как общая информация о том, как генерировать волны различной формы, используя PWM (ШИМ).

3.1 Генерация напряжения

Чтобы управлять трехфазным двигателем с помощью синусоидальных токов, для каждой фазы должны быть сгенерированы независимые напряжения. Драйвер для трехфазного двигателя обычно состоит из трех полумостов, по одному на каждый вывода двигателя. Каждый полумост состоит из двух ключей, обычно это два мощных MOSFET транзистора. Чтобы понимать, как генерируются напряжения для каждой фазы, достаточно рассмотреть один полумост. Рисунок 3-1 показывает один такой полумост, подключенный к источнику постоянного тока.

Рисунок 3-1. Генерация напряжения, используя полумост.



3.1.1 PWM

Среднее напряжение вывода, V_{0UT}, может регулироваться между 0V и V_{DC}, с помощью широтно-импульсной модуляции (по-русски ШИМ, по-английски PWM) в инверсии поданной на два ключа, PWM_H и PWM_L. Среднее напряжение будет пропорционально коэффициенту заполнения (соответствует отношению длительности импульса к длительности периода (0-100%), величина обратная скважности) верхнего ключа. Напряжение на V_{0UT}, в этом случае не будет гладкой кривой, а станет такой же прямоугольной формы как сигнал PWM, приложенный к верхнему ключу.

Если этот сигнал подать на фильтр низких частот (ФНЧ), то напряжение на выходе фильтра будет пропорционально коэффициенту заполнения верхнего ключа.

По нескольким причинам, в устройства управления двигателем обычно не добавляют специальный фильтр низких частот. Во-первых, потому, что двигатель сам действует как ФНЧ. Индуктивность и сопротивление его обмоток создают RL ФНЧ. Дополнительно, инерция ротора и нагрузки создает механический ФНЧ. Выбирая частоту PWM достаточно высокой, можно добиться отсутствия сколь-либо значимой флуктуации в скорости ротора. Во-вторых, токи, поданные на обмотки даже маленького двигателя, могут достигать нескольких ампер. Прохождение такого тока через фильтр, например RC, привело бы к существенной потере мощности в самом фильтре и к нежелательной потере энергии.

3.1.2 Мертвая зона

Переключение транзисторов MOSFET не может быть мгновенным. Рассмотрите снова полумост на Рисунке 3-1. На ключи PWM_H и PWM_L сигналы поданы в инверсии, то есть когда один ключ включается, другой ключ выключается. В течение этого перехода, будет краткий момент, когда один ключ закрылся ещё не полностью, а другой уже начал открываться. Возникает замыкание между напряжением питания и землей через транзисторы с очень малым сопротивлением, соответственно, ток может быть очень большим. Эта ситуация известна как сквозной ток, и её нужно избегать, так как велика вероятность выхода драйвера из строя, если не предусмотрена аппаратная защита.

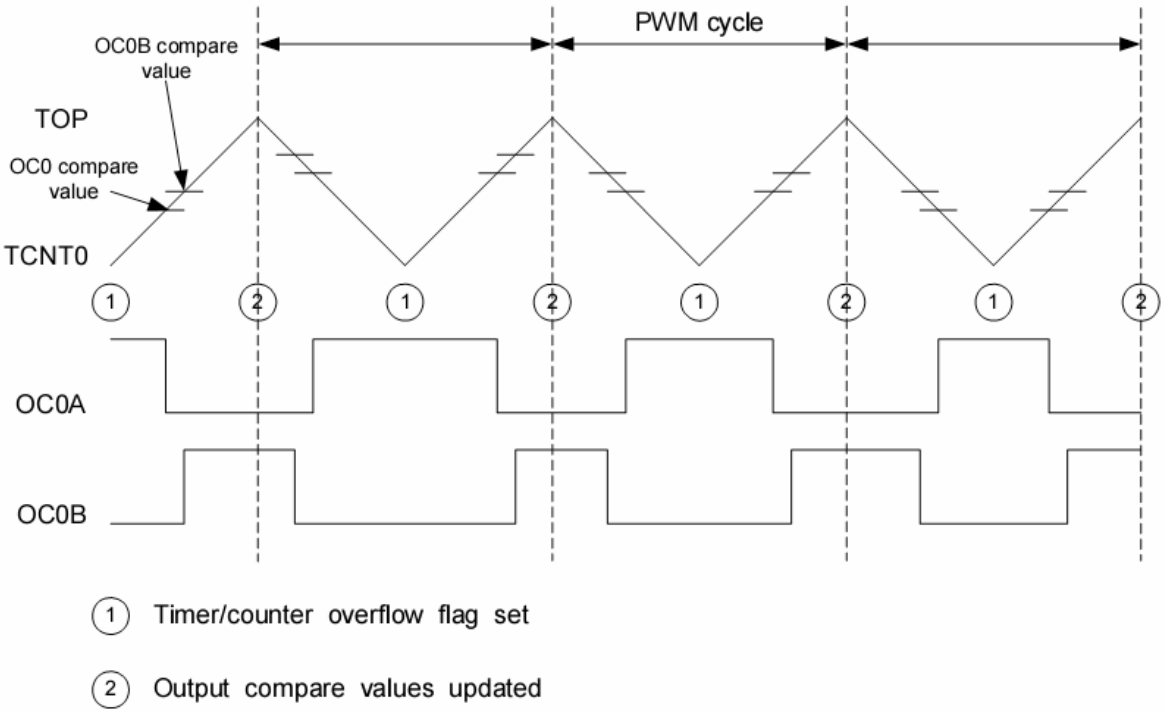
Решение этой проблемы состоит в том, чтобы для каждого переключения PWM добавить маленький период времени, паузу, мертвую зону, где оба ключа выключены.

3.1.3 Генерация PWM с мертвой зоной на AVR

Чтобы понять, как осуществить генерацию PWM с мертвой зоной на AVR, будем рассматривать только одну фазу. Один таймер/счетчик (на примере ATmega48) может управлять одним полумостом, что в Рисунке 3-1. Каждый таймер/счетчик в ATmega48 могут выдавать PWM сразу на два вывода (ножки) микроконтроллера. Таймеры/счетчики имеют несколько режимов PWM, и несколько вариантов конфигурации для выводов.

Для управления полумостом с мертвой зоной, хорошо подходит режим PWM «phase correct». Счетчик работает в двойном наклонном режиме, который позволяет производить выровненные по центрам PWM сигналы с мертвой зоной. Рассмотрите рис.3-2, который показывает, как Timer/Counter0 может быть настроен, чтобы произвести нужные сигналы. Треугольная строка показывает значение таймера/счетчика работающего в режиме «phase correct». В этом примере, вывод OC0A был конфигурирован так, чтобы сбрасываться в 0 по совпадению, когда счетчик идет вверх и устанавливаться в 1, по совпадению, когда счетчик идет вниз. Этот вывод будет связан с верхним ключом полумоста. Аналогично вывод OC0B был сконфигурирован так, чтобы устанавливаться в 1 по совпадению, когда счетчик идет вверх и сбрасываться в 0 по совпадению, когда счетчик идет вниз. Этот вывод будет связан с нижним ключом полумоста. Обратите внимание, что, если значения сравнения для обоих выводов установлены на одно и тоже значение, два вывода будут комплиментарны. Для того чтобы вставить мертвую зону между переключениями верхнего и нижнего ключа, значения сравнения должны быть немного изменены, каждое в своем направлении. Половина мертвой зоны вычитается из значения сравнения для OC0A, а к значению сравнения OC0B она добавляется. Это иллюстрировано короткими горизонтальными штрихами на кривой таймера/счетчика в Рисунке 3-2. В итоге, как видно из рисунка, на выводах OC0A и OC0B, будет присутствовать импульсы переменной ширины с мертвыми зонами в каждом переключении.

Рисунок 3-2. Генерация комплементарных PWM с мертвой зоной.



Чтобы исключить случайный сквозной ток в момент изменения значений, очень важно, чтобы значения сравнения для OC0A и для OC0B изменялись одновременно. Это достигается двойной буферизацией регистров сравнения и использованием прерываний от таймеров/счетчиков AVR. На рис.3-2, каждый цикл PWM отмечен пунктирными линиями. Когда используется режим PWM «phase correct» новые значения сравнения для обоих выводов будут применены в момент времени, отмеченный на рис.3-2 цифрой 2. В момент, отмеченный цифрой 1, таймер/счетчик устанавливает флаг переполнения. Это может быть использовано для периодического прерывания, где значения сравнения могут быть изменены. Если используются 16-битные таймеры/счетчики, их надо сконфигурировать на использование регистра захвата для хранения наибольшего значения счетчика. В этом случае прерывание по захвату будет вызвано в тоже время, когда будут обновлены сравнивающие значения. Использование этого прерывания вместо прерывания переполнения, удваивает число моментов для расчета новых значений сравнения. Так как в этой РпП используются все таймеры/счетчики, то это прерывание используется.

3.1.4 Частота PWM

Когда для генерации PWM в режиме «phase correct» используется 8-битный таймер/счетчик его верхнее значение фиксировано и равно 255. 16-битные таймеры/счетчика также должны работать подобно 8-битным. В режиме «phase correct» один цикл PWM, с верхним значением счетчиков 255, будет состоять из 510 тактов счетчиков, или 510 тактов CPU, если полагать, что тактовый предделитель установлен на значение 1. Тогда частоту PWM можно рассчитать по формуле 3-1.

Формула 3-1. Частота PWM как функция частоты центрального процессора.

$$f_{PWM} = \frac{f_{CPU}}{510}$$

3.2 Генерация формы волны

Используя информацию из раздела 3.1 о том, как настроить AVR, чтобы генерировать выходные напряжения для одного полумоста, можно сделать систему, чтобы управлять двигателем через три полумоста.

3.2.1 Настройка AVR

Каждый таймер/счетчик на ATmega48 способен управлять одним полумостом, чтобы управлять тремя полумостами, используются все три таймера/счетчика. Каждый таймер настроен и связан с одним полумостом согласно разделу 3.1. Важно, что бы эти три таймера работали синхронно. Так как они были запущены в разные моменты, каждый таймер предварительно загружается значением. Важно удостовериться, что таймеры синхронизированы, например, проверить значения регистров трех таймеров, выполнив симуляцию в AVRStudio®.

3.2.2 Генерация волны произвольной формы

Чтобы вычислить значения сравнения для трех таймеров/счетчиков, нужны три шага:

- получение нужного значения
- масштабирование значения к нужной амплитуде
- вставка мертвой зоны

Значение выхода при запуске синхронного двигателя – зависит от позиции ротора. Значения выхода могут быть или рассчитаны или найдены в таблице. В этом документе, для увеличения производительности программы, используется таблица значений.

Значения, сохраненные в таблице, соответствуют максимальной амплитуде выхода. Значение из таблицы должно быть уменьшено до нужной амплитуды выхода. Формула 3-2 показывает, как может масштабироваться значение. Коэффициент заполнения d_0, получен, умножением значения из таблицы d_{table}, на коэффициент масштабирования k_s.

Формула 3-2. Масштабирование амплитуды.

$$d_o = \frac{k_s d_{table}}{2^n}$$

Разрешающая способность выхода может быть рассчитана как $n_0 = n_k+n_t-n$, где n_0, n_k и n_t - разрешающие способности выхода, масштабирующего коэффициента и значения таблицы, соответственно и n – показатель степени делителя в формуле 3-2. Например, разрешающая способность табличных значений - 8 бит, коэффициент масштабирования - 8 бит и экспоненты делителя – 8, будут генерировать значение коэффициента заполнения вывода с разрешающей способностью 8 бит.

Мертвые зоны вставлены, как описано в разделе 3.1.3. и значения сравнения могут быть введены в регистры сравнения таймеров/счетчиков.

Эти шаги выполняются при каждом переполнении таймера/счетчика, чтобы менять значения выходов правильно относительно позиции ротора.

3.3 Генерация синусоиды

Раздел 3.2 объясняет, как сделать волну произвольной формы, сохраненной в таблице значений. В этом разделе, объясняется, как эффективным способом генерировать синусоиду.

3.3.1 Образец вывода

Наиболее прямой подход состоял бы в том, чтобы сохранить в таблице всю синусоиду и использовать эту таблицу для генерации трех синусоид. Однако есть более эффективные способы сделать синусоидальные сигналы для привода двигателя. Ключ к пониманию этого - то, что мы не будем генерировать синусоидальные сигналы для каждого вывода двигателя относительно земли. Мы будем генерировать три синусоидальных напряжения вывод-вывод (дифференциальное напряжение между двумя выводами) со сдвигом фазы на 120° между ними. Таблица 3-1 и Таблица 3-2 показывают, как это может быть достигнуто, не делая полные синусоиды для каждого вывода двигателя. Графическое представление этого показано на рис.3-3, наряду с типичным образцом блочной коммутации.

Таблица 3-1. Выводы и напряжения между ними, вращение вперед

Step	U	V	W	U-V	V-W	W-U
S1-S2	sin(θ)	0	-sin(θ -120)	sin(θ)	sin(θ -120)	sin(θ -240)
S3-S4	-sin(θ -240)	sin(θ -120)	0	sin(θ)	sin(θ -120)	sin(θ -240)
S5-S6	0	-sin(θ)	sin(θ -240)	sin(θ)	sin(θ -120)	sin(θ -240)

Таблица 3-2. Выводы и напряжения между ними, вращение назад

Step	U	V	W	U-V	V-W	W-U
S1-S2	sin(θ)	-sin(θ -120)	0	-sin(θ -240)	-sin(θ -120)	-sin(θ)
S3-S4	-sin(θ -240)	0	sin(θ -120)	-sin(θ -240)	-sin(θ -120)	-sin(θ)
S5-S6	0	sin(θ -240)	-sin(θ)	-sin(θ -240)	-sin(θ -120)	-sin(θ)

У такого подхода есть два преимущества. Во-первых, максимальная амплитуда сгенерированного напряжения вывод-вывод выше, чем при генерации чистой синусоиды на каждом выводе, следовательно, больше вращающий момент и скорость. Во-вторых, каждый оконечный вывод 1/3 времени подключен к «земле», тем самым сокращая потери в мощном каскаде на переключения.

3.3.2 Организация таблицы значений

При организации таблицы возникает необходимо разрешить компромисс между размером таблицы и временем доступа. Рассматривая форму волны на рис.3-3, можно заметить, что информация может быть "сжата" несколькими способами.

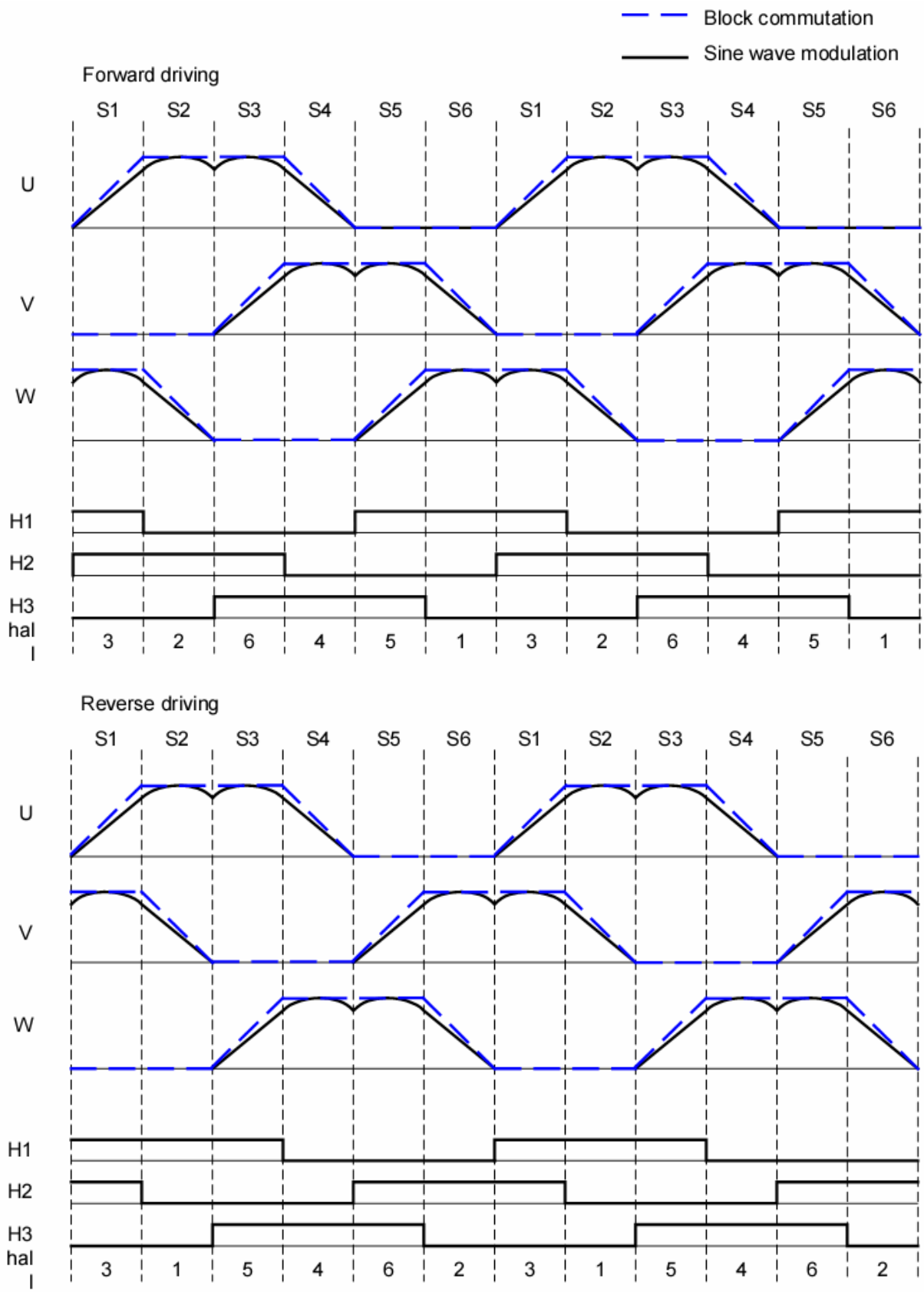
Три волны отличаются только фазой, сдвинутой на 120 градусов, так что возможно сохранить только одну волну и использование её для всех трех выходных напряжений. Кроме того, глядя на форму волны для фазы U, видно, что шаги S3-S4 – зеркальное отражение S1-S2. В S5-S6, U просто является нулевым. Это позволяет сохранить только одну треть таблицы и использовать программу, чтобы найти нужное значение. Первый метод имеет некоторые издержки в корректировке указателей таблицы для указания на правильное место. Последний метод имеет ещё большие издержки.

Так как поиск в таблице должен быть выполнен для всех трех каналов в каждом цикла PWM, даже маленькие издержки могут способствовать значительному росту средней загрузки центрального процессора. Здесь принято решение сохранять в таблице значения для всех трех синусоид, таким образом, гарантируя самый быстрый доступ. Дополнительно, организовав, таблицу как показано в таблице 3-3, можно будет получить все три значения с помощью LPM Z+ , весьма быстрой команды. Чтобы получить значения выхода для вращения двигателя вперед, таблица читается в последовательности U, V, W, в то время как для вращения в обратном направлении, используется последовательность U, W, V.

Таблица 3-3. Организация таблицы синуса.

U_0	U_0 -120°	U_0 -240°	U_1	U_1 -120°	U_1 -240°	….

Рисунок 3-3. Генерация синусоиды



3.4 Синхронизация

Все доступные таймеры/счетчики в ATmega48 используются для генерации PWM. По этому, нет выделенного таймера, доступного для измерения скорости. Решение состоит в том, чтобы использовать прерывание переполнения таймеров/счетчиков как базовое время. Период между прерываниями переполнения используется как 'тик' времени. Продолжительность одного тика равна периоду PWM, который вычисляется по формуле 3-3.

Уравнение 3-3. Время периода PWM.

$$T_{PWM} = \frac{1}{f_{PWM}} = \frac{510}{f_{CPU}}$$

Если, например тактовая частота процессора равна 8МГц, тогда один тик будет длительностью 63.75мкс.

3.5 Датчики позиции и их использование

Так как основной целью является управление синхронным двигателем, необходима некоторая информация о положении ротора, чтобы правильно генерировать синусоиду для двигателя.

Эта информация может быть получена, например, от вращающегося энкодера. Однако многие двигатели с постоянными магнитами оборудованы тремя установленными датчиками холла, которые дают информацию о позициях ротора в приращениях по 60 градусов. Здесь, датчики холла используются как единственные сенсоры положения ротора.

На рис.3-3, три сигнала датчиков холла показаны как H1, H2 и H3. Три выхода датчиков холла преобразуются в одно число, от 1 до 6, где каждый из сигналов датчиков холла обрабатывается как двоичный разряд в порядке $холл = H3\ H2\ H1$. Значения *холла*, соответствующие различным комбинациям *H3*, *H2* и *H1* показаны на Рисунке 3-3.

Состояние датчиков холла и время между изменениями датчиков используется для нескольких целей:

- Фазовая автоподстройка частоты
- Вычисление скорости
- Блочная коммутация
- Обнаружение вращения.
- Синхронизация и изменение направления
- Управление опережением угла коммутации
- Выходной сигнал тахогенератора

3.5.1 Фазовая автоподстройка частоты

Цель волновой генерации - выдавать синусоиду синхронную с вращением ротора. Проблема при использовании датчиков холла как датчика позиции, в том, что актуальная информация о позиции ротора, доступна только каждые 60 градусов. Чтобы решить эту проблему, нужно сделать фазовую подстройку частоты, что сохраняет синусоиду в синхронизации с ротором.

Индекс (или указатель) таблицы значений всегда доступен. Информация, полученная от датчиков холла, используется, чтобы модифицировать этот индекс.

Самая точная информация о позиции ротора доступна только в момент, когда датчик холла изменяет значение. В этой точке, известен точный угол ротора.

В промежуток между изменениями датчика холла, информация о позиции ротора недоступна. Однако мы знаем позицию ротора в момент последнего изменения датчика холла, и можем измерить период времени между двумя последними изменениями датчика холла. Это позволяет

нам вычислять скорость ротора и, принимая скорость постоянной, промежуточная позиция ротора может быть рассчитана формулы 3-4.

Формула3-4. Интерполяция позиции.

$$\dot{\theta} = \omega$$
$$\theta(t) = \theta_0 + \omega t$$

где θ - угловая позиция, θ_0 - угловая позиция в последнем изменении датчика холла, ω - угловая скорость, и t – время, прошедшее с последнего изменения датчика холла.

Не входя в подробность, формула 3-4 может быть аппроксимировано дифференциальным уравнением, показанным в формуле 3-5.

Формула 3-5. Дифференциальное уравнение углового смещения.

$$\theta[k] = \theta[k-1] + \omega T$$

где θ - угловая позиция, k - текущий шаг времени, ω - угловая скорость, и T - размер шага времени.

В этой РпП, угловая позиция представлена индексом таблицы значений, измеряемым в шагах таблицы. Время измерено в тиках. Угловая скорость определена Формуле 3-6.

Формула 3-6. Определение угловой скорости.

$$\omega = \frac{\Delta\theta}{\Delta t}$$

где $\Delta\theta$ - угловое смещение за времена Δt. Так как угловое смещение измерено в шагах таблицы, и время измерено в тиках, модуль угловой скорости - шаги таблицы за один тик.

Индекс таблицы значений изменяется один раз на каждый тик, так что шаг времени, T, в формуле 3-5 равен 1 тик. Таким образом, приращение, используемое для итерации таблицы, между двумя последовательными изменениями датчиков холла, может быть рассчитано по формуле 3-7.

Формула 3-7. Вычисление инкремента индекса таблицы значений.

$$i = \omega T = \frac{\Delta\theta}{\Delta t} T = \frac{n_e}{n_T}$$

где n_e - число элементов таблицы на шаг коммутации (60 градусов вращения), и n_T - число тиков между двумя последними коммутациями.

На каждом тике, инкремент добавляется к индексу таблицы значений, чтобы изменить информацию о позиции.

Алгоритм, осуществляющий фазовую автоподстройку, получается в итоге таким:

1. При каждом изменения датчиков холла:
   - Индекс таблицы значений ставиться в соответствие с позицией ротора
   - Вычисляется инкремент по формуле 3-7.
2. Для каждого тика до следующего изменения датчика холла:
   - Индекс таблицы значений изменяется согласно формуле 3-5.
   - Коэффициенты заполнения PWM модифицируют по таблице значений согласно индексу.
3. При изменении датчиков холла, возврат к пункту 1.

3.5.2 Вычисление скорости

Если требуется контролировать скорость, то нужно рассчитать скорость вращения двигателя. Как показано в разделе 3.5.1, информация о скорости уже рассчитана в виде значения инкремента таблицы значений. Нет необходимости выполнять еще одно «тяжелое» вычисление, чтобы получить информацию о скорости. И при этом не нужно иметь еще одну переменную с информацией о скорости. Вычислительные ресурсы могут быть сохранены для других применений значений скорости, например, встроить управление скоростью в тот же самый модуль, где рассчитывается инкремент, так как это непосредственно соответствует уже доступной информации о скорости.

Чтобы привести скорость к некоторому значению RPM (оборотов в минуту), это значение должно быть сначала преобразовано в соответствующее значение инкремента. Поэтому необходимо знать отношения между скоростью вращения и инкрементом. Как скорость RPM, связана с числом тиков между изменениями датчика холла, показано в формуле 3-8.

Формула 3-8. Вычисление RPM.

$$\omega_{RPM} = 60\frac{\frac{1}{6}\cdot f_{CPU}}{510\cdot n_T}$$

Отсюда выводиться зависимость числа тиков между изменениями датчиков холла от RPM:

Формула 3-9. Число тиков между изменениями холла как функция скорости.

$$n_t = \frac{60\cdot f_{CPU}}{6\cdot 510\cdot \omega_{RPM}}$$

Объединение формулы 3-7 и формулы 3-9 дает инкремент индекса как функцию скорости (RPM):

Формула 3-10. Инкремент как функция скорости в RPM.

$$i = \frac{n_e\cdot 6\cdot 510}{60\cdot f_{CPU}}\omega_{RPM}$$

3.5.3 Блочная коммутация

В момент старта, скорость ротора не известна, до тех пор, пока не будут обнаружены два последовательных изменения датчика холла. Чтобы гарантировать устойчивый запуск, пока скорость ротора не известна, используется блочная коммутация. При работе в режиме блочной коммутации, все 6 выводов PWM используются с одним коэффициентом заполнения, и процесс коммутация заключается в том, что PWM подается только на нужные ключи. Карта вывода меняется при каждом изменении холла. Для каждого направления вращения используется одна таблица, где хранятся значения карты вывода соответствующие состояниям датчика холла. Для подробной информации относительно запуска PMM в режиме блочной коммутации, обратитесь к рекомендации по применению AVR443.
(оригинал - http://www.atmel.com/dyn/resources/prod_documents/doc2596.pdf;
перевод - http://www.gaw.ru/html.cgi/txt/app/micros/avr/AVR443.htm).
Карта вывода, используемая в режиме блочном коммутации, относительно датчика холла также показана на рис.3-3 (штриховая линия) наряду с синусоидальным образцом.

3.5.4 Обнаружение вращения

Последовательность изменений датчика холла может использоваться, чтобы определить фактическое направление вращения. Для каждого направления используется одна таблица, чтобы по текущему значению датчика холла найти следующее, ожидаемое, значение датчика холла. Это используется, чтобы определить фактическое направление вращения.

Сравнение фактического направления вращения с требуемым направлением вращения, позволяет определить, вращается ли двигатель в нужном направлении.

3.5.5 Синхронизация и изменение направления

Когда микроконтроллер только начинает работу, был включен или сброшен, двигатель может уже вращаться. Поэтому важно, что бы программа умела определять состояние двигателя и подстраиваться к этому состоянию.

При изменении направления вращения, ротор должен быть сначала остановлен, а уж потом запущен в противоположном направлении. Если команда изменения направления поступит еще раз, прежде чем двигатель остановился, программа должна повторно синхронизироваться перед подачей напряжений на выводы.

Синхронизация и вращение управляются датчиками холла. Информация о направлении вращения получена, как описано в разделе 3.5.4.

Двигатель полагают остановленным, когда прошло достаточное число тиков с момента последнего изменения датчика холла.

Синхронизация происходит, когда двигатель вращается в нужном направлении и обнаружены не менее двух изменений датчика холла.

3.5.6 Коммутация с опережением

Программируемый угол опережения коммутации может использоваться, чтобы настраивать фазу синусоиды, чтобы заставить её опережать угол ротора. Это необходимо что бы заставить двигатель работать на максимальной скорости и/или эффективности. Угол опережения коммутации, во время выполнения доступен и может быть установлен, чтобы измениться согласно, например скорости ротора или амплитуде. Угол опережения коммутации получается, если добавлять смещение к индексу таблицы значений, таким образом, сдвигая фазу волны. Угол опережения коммутации может быть настроен в шагах таблицы значений (1.8 °).

3.5.7 Вывод сигнала тахогенератора

Выходной сигнал тахогенератора сгенерирован непосредственно от датчика холла, чтобы выдать сигнал, который меняет полярность при каждом изменении датчика холла.

3.6 Обнаружение сверхтока

Блокировка ротора, внезапное изменение нагрузки или быстрое ускорение могут вызвать чрезмерный ток через драйвер и двигатель. Чтобы избежать повреждений из-за сверхтока, очень важно всегда контролировать ток. Обычно, устанавливается шунт между транзисторами драйвера и землей, и, чтобы вычислить ток, измеряется падение напряжения на этом шунте. Это может быть сделано аналого-цифровым преобразователем (ADC) или аналоговым компаратором.

В этом документе использовался ADC для измерения тока. Использование ADC вводит небольшую задержку, из-за времени преобразования, но есть другие преимущества. Аналоговый компаратор работает все время, и поэтому очень чувствителен к шумам от переключений PWM, если не делать сложную внешнюю фильтрацию. Эту проблему проще преодолеть с помощью ADC. Напряжение измеряется в определенный момент, который может быть вызван, например, прерыванием переполнения в таймере PWM (точка 1 на рис.3-2), тогда измерение будет сделано в середине цикла PWM. Если коэффициент заполнения не слишком мал, этот момент будет далеко от любого переключения PWM. В результате потребуется более простая внешняя фильтрация, чтобы получить надежные значения тока.

3.7 Управление скорости

Эта РпП включает примеры и управления скорости разомкнутого цикла и контроллера PID. Также возможно добавить другие виды регуляторов скорости. Как вычислять фактическую скорость ротора, было описано в разделе 3.5.2.

3.7.1 Задатчик скорости

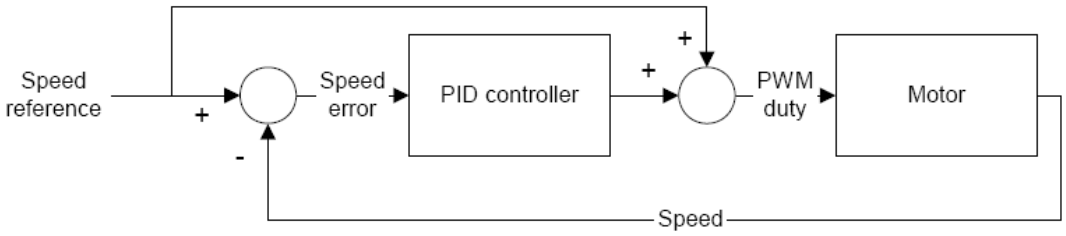
Контроллер, открытого или закрытого цикла, нуждается в некотором задатчике скорости. В этом документе используется аналоговое напряжение, хотя это могло бы быть легко заменено, например командой UART. Аналоговый задатчик измеряется с помощью ADC.

3.7.2 Контроллер скорости

Можно построить программу управления скоростью с закрытым или открытым контуром (с обратной связью (ОС) или без). Управление скоростью без ОС очень просто. Аналоговое значение задатчика скорости, оцифрованное на 8 бит, непосредственно используется, как значение амплитуды на 8 бит для сгенерированных синусоид.

Управление скоростью с ОС также использует значение аналогового задатчика скорости на 8 бит как первичное значение к установке амплитуды. Кроме того, PID (пропорциональная, интегральная производная) контроллер используется, чтобы удостовериться, что скорость точно соответствует желаемой. Измерение ADC используется как точка назначения для контроллера скорости. Так как внутреннее представление скорости – инкремент индекса, взвешенный сигнал должен быть преобразован в то же самое представление. Раздел 3.5.2 описывает отношения между скоростью в RPM и внутренним представлением инкрементом индекса. Блок-схема системы с ОС с подачей вперед показана на рисунке 3-4.

Рисунок 3-4. Контроллер PID с подачей вперед.



Цикл управления скорости - единственная часть программы управления двигателем, которое не управляется прерыванием. Это, потому что вычисления PID производятся слишком долго, чтобы исполнятся в подпрограмме прерывания, без ухудшения управления двигателем. Кроме того, нет необходимости выполнять цикл управления так часто, как коммутации.

4 Выполнение программы

Исходный текст программы, предоставляемой с этой РпП, полностью прокоментирован Doxygen, который объясняет все части кода. Открытие файла 'readme.html' поможет обратиться к полной документации Doxygen в формате html.

Эта глава включает дополнительную информацию, нужную для более полного понятия программы.

4.1 Структура кода

Обратите внимание, что код был написан для высокой эффективности. Из-за этого, почти весь исходный текст содержится в одном файле, чтобы позволить компилятору максимально оптимизировать код. Большинство функций объявлено с директивой " *pragma inline=forced", так как их вызывают из подпрограмм прерывания.

4.2 Использование периферии

Периферийные модули ATmegax8, используемые в этой программе перечислены в Таблице 4-1.

Таблица 4-1. Использование аппаратных модулей.

Модуль	Использование
Timer/counter0	PWM фазы U
Timer/counter1	PWM фазы V
Timer/counter2	PWM фазы W
Каналы ADC	Задатчик скорости/измерение тока
Pin change interrupt 0	Прерывание аварийного завершения
Pin change interrupt 1	Прерывание по изменению датчиков холла
Pin change interrupt 2	Прерывание изменения направления

4.3 Действия, выполняемые в прерываниях

Программа управления двигателем полностью построена на прерываниях, исключая управление скоростью. В Таблице 4-2 показано, за что отвечает каждое прерывание, используемое в программе. Понимание работы каждой подпрограммы обработки прерывания - ключ к пониманию, как работает вся программа.

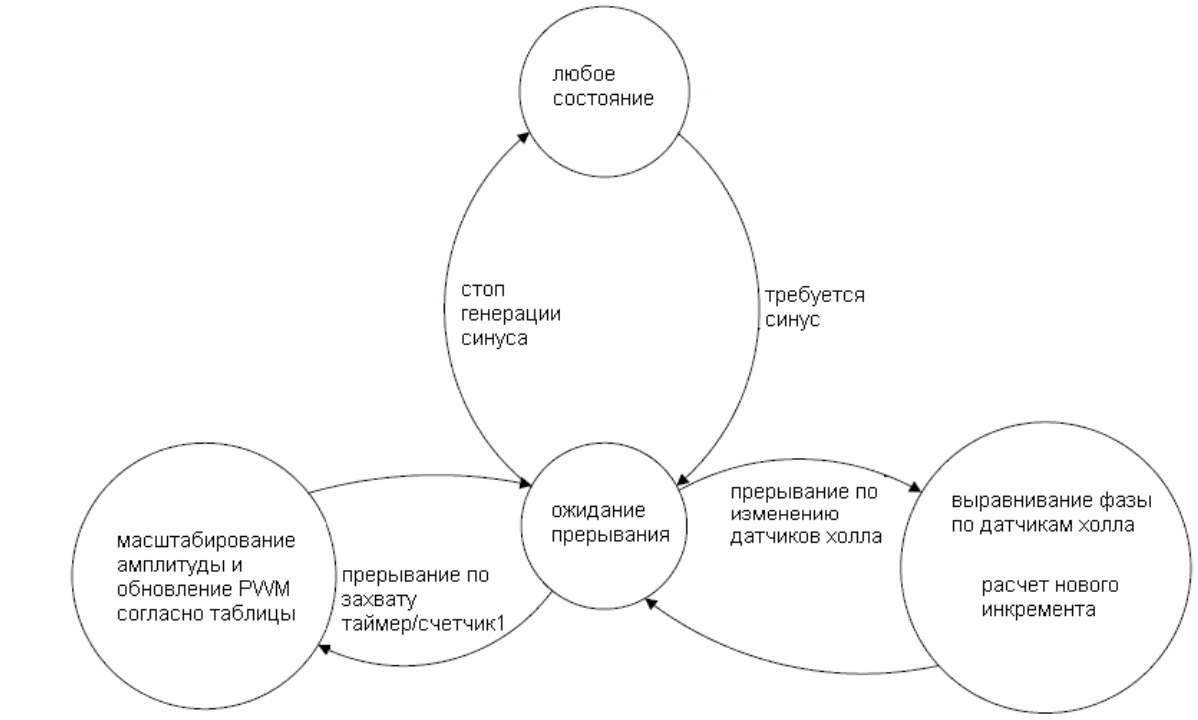
Таблица 4-2. Действия прерываний.

Прерывание	Функция	Ответственность/действие
Timer/counte1 захват	`Timer1CaptureISR`	Обновление значения PWM. Управление коэффициентом заполнения блочной коммутации. Измерение скорости. Обнаружение останова.
Pin change 0	`EmergencyInterruptISR`	Обработка внешнего сигнала выключения
Pin change 1	`HallChangeISR`	Синхронизирует синусоиду с датчиками холла. Блочная коммутация. Обновление сигнала тахогенератора. Обнаружение фактического направления. Проверка синхронизации двигателя и программы. Обновление сигнала противовращения. Вычисление инкремента индекса таблица синуса. Обнаружение останова
Pin change 2	`DirectionInputChangeISR`	Ввод сигнала направления вращения.
ADC complete	`ADCCompleteISR`	Управление амплитудой для синусной/блочной коммутации. Измерение тока. Выбор канала ADC.

4.4 Генерация синусоиды

Две подпрограммы обработки прерывания используются, чтобы получить на выходе синусоиду. `Timer1CaptureISR` изменяет регистр сравнения PWM, один раз для каждого цикла PWM. `HallChangeISR` синхронизирует синусоиду по текущему положению ротора, и вычисляет инкремент индекса таблицы значений. Рисунок 5-2 показывает взаимодействие между этими двумя прерываниями. Состояние, обозначенное как 'Любое состояние', показывает, что не важно в каком состоянии был или будет двигатель.

Рисунок 4-1. Конечный автомат генерации синусоиды.



4.5 Контроль направления и синхронизации

4.5.1 Связные флаги

Необходимы два определения, чтобы объяснить контроль синхронизации и направления:

Синхронизатор указывает ожидаемое значение датчиков холла для направления вращения, которым командует входной сигнал направления. Это нужно, чтобы гарантировать, что синусоида генериться с правильной частотой, фазой и в нужном направлении.

Детектор останова указывает, что датчики холла не менялись в течение определенного числа тиков.

Два флага, которые являются частью глобальной переменной fastFlags, указывают, синхронизирован ли двигатель в настоящее время и/или остановлен. Эти флажки автоматически проверяются подпрограммами обработки прерывания в некоторых событиях. Следующие функции/прерывания изменяют эти флаги:

`CommutationTicksUpdate:`

- `motorStopped=ИСТИНА`, если предопределенный число ‘тиков’ прошло с последнего изменения датчика холла.
- `motorSynchronized=ЛОЖЬ`, если флаг motorStopped был только что установлен на `ИСТИНУ`.

Эту функцию вызывает `Timer1CaptureISR`.

`MotorSynchronizedUpdate:`

- `motorSynchronized=ИСТИНА`, если выполнены критерии синхронизации.
- `motorSynchronized=ЛОЖЬ`, если критерии синхронизации не выполнены.

Эту функцию вызывает `HallChangeISR`.

`HallChangeISR:`

- `motorStopped=ЛОЖЬ` (двигатель не может быть остановленным, если датчики холла изменяются)

`DirectionInputChangeISR:`

- `motorSynchronized=ЛОЖЬ`
- `motorStopped=ЛОЖЬ`

4.5.2 Логика синхронизации и направления

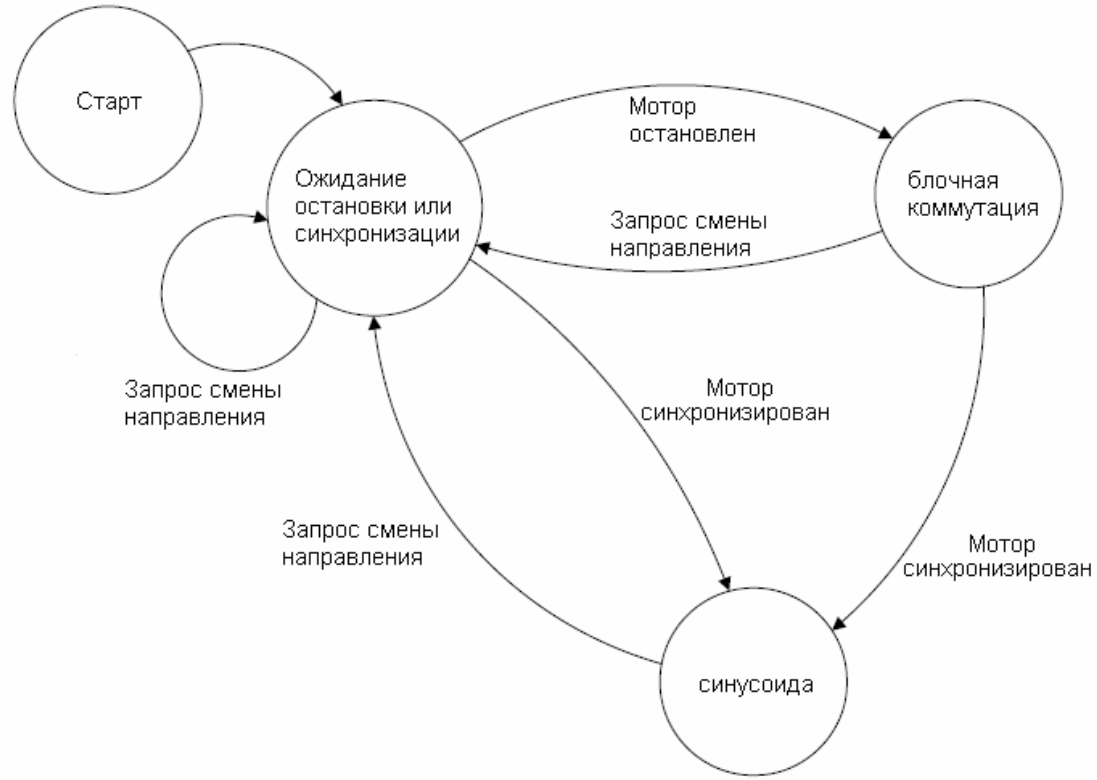
Есть несколько ситуаций, где программа должна синхронизироваться с двигателем, прежде чем начнет генерировать синусоиду:

- Когда двигатель стартует после останова, первые переключения используют блочную коммутацию. Блочная коммутация используется, пока синхронизация не получена. Это гарантирует, что синусоида будет сгенерирована с правильной частотой и фазой, когда двигатель переключится на неё.
- Когда микроконтроллер стартует, а двигатель уже вращается, программа не будет ничего генерировать, пока она не засинхронизируется с двигателем, или пока ротор не прекратит вращение.
- Когда требуется изменить направление вращения, запуск будет заблокирован или инициализировано торможение, пока двигатель не остановиться. Если снова потребуется изменение направления, двигатель может быть синхронизироваться заново. В этом

случае, возобновляется синусоидальный запуск с правильной частотой, не дожидаясь остановки двигателя.

Полностью управление синхронизацией и направлением показаны на рис.4-2. Двигатель остановлен, значит, флажок `motorStopped` ИСТИНЕН. Двигатель синхронизирован, значит, что флажок `motorSynchronized` ИСТИНЕН. Если требуется сменить направление, тогда должно быть выполнено `DirectionInputChangeISR`.

Рисунок 4-2. Конечный автомат управления направлением и синхронизацией.



4.6 Аналого-цифровые преобразования

В этом примере программы, ADC используется, чтобы принимать две величины: задатчик скорости и ток двигателя. ADC способен производить преобразование только одной выборки за раз, так что входные каналы конвертируются по очереди.

Каждое переполнение Timer/counter0 автоматически вызывает осуществление выборки входного значения. Это гарантирует, что выборки происходят в середине цикла PWM, делая каждое текущее измерение, сопоставимым с последним.

Когда период преобразования закончен, вызывается соответствующее прерывание ADC, и полученное значение помещается в глобальную переменную, соответствующую каналу ADC выбранному в настоящее время. Флаг прерывания переполнения Timer/Counter0 должен быть очищен вручную, если прерывание переполнения Timer/Counter0 не выполнено. Новый преобразование не будет вызвано, пока этот флаг не очищен.

Если необходимо обрабатывать больше входных сигналов, подпрограмма прерывания ADC может быть расширена, чтобы добавить больше чтений к циклу.

Обратите внимание, в исходном тексте программы, значение тока двигателя только сохраняется в глобальной переменной. Это значение нигде в программе не используется, для ограничения тока или завершения по сверхтоку, поскольку эти ограничения тока зависят от конкретных устройств.

5 Аппаратные средства

Эта глава описывает, как подключить аппаратные средства для использования согласно с этим документом.

5.1 Назначение выводов

Назначение выводов, используемое в этом документе, показано на рисунке 5-1. Функция каждого вывода описана в Таблице 5-1.

Рисунок 5-1. Назначение выводов

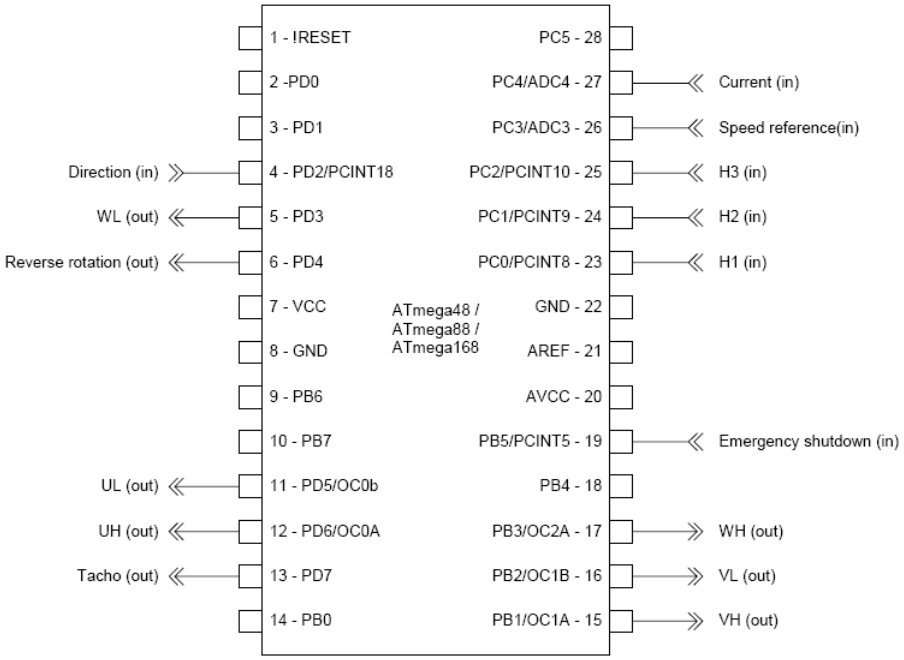


Таблица 5-1. Использование выводов AVR

Pin	Имя	назначение	направление
PD5	UL	Фаза U сигнал управления нижнего ключа	Out
PD6	UH	Фаза U сигнал управления верхнего ключа	Out
PB2	VL	Фаза V сигнал управления нижнего ключа	Out
PB1	VH	Фаза V сигнал управления верхнего ключа	Out
PD3	WL	Фаза W сигнал управления нижнего ключа	Out
PB3	WH	Фаза W сигнал управления верхнего ключа	Out
PCO	H1	Датчик Холла 1	In
PC1	H2	Датчик Холла 2	In
PC2	H3	Датчик Холла 3	In
PC3	Задатчик скорости	Аналоговый вход управления амплитудой. Внешний сигнал в пределах 0-1.1V.	In
PC4	Ток	Измерение тока двигателя. Внешний сигнал в пределах 0-1.1V.	In
PB5	Экстренный останов	Когда измениться логический уровень на этом выводе будет вызвано прерывание `EmergencyInterrupt`.	In
PD2	Направление вращения	0-вперед, 1-назад.	In
PD4	Реверсивное вращение	0 - двигатель вращается согласно сигналу «Направление вращения». 1 – двигатель вращается в обратном направлении или остановлен.	Out
PD7	Тахо	Выходной сигнал (меандр) с частотой в 3 раза выше электрической частоты вращения двигателя.	Out

5.2 Соединение Atmega48/88/168 с драйвером и двигателем

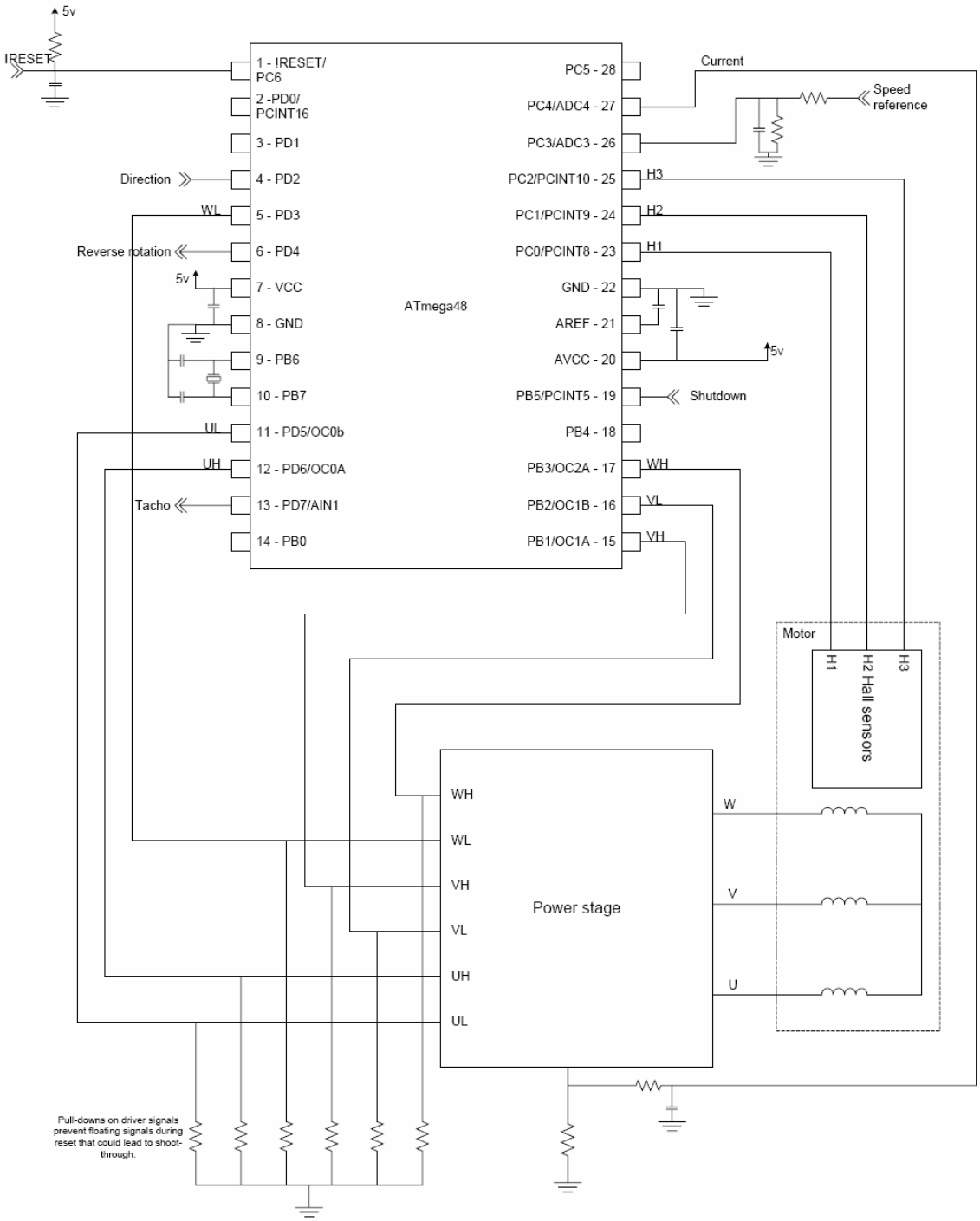
Для работы этой программы требуются:

- 3-фазный двигатель с постоянными магнитами и датчиками холла.
- Драйвер, способный управлять двигателем.
- Atmel ® ATmega48/88/168 микроконтроллер.
- Аналоговый входной сигнал в диапазоне 0-1.1V (для управления скорости).

Рисунок 5-2 показывает концептуальную схему построения полной системы.

Обратите внимание, чтобы программа могла работать, сигналы *завершения* и *направления*, даже если они не нужны, должны быть подключены к соответствующим логическим уровням.

Рисунок 5-2. Концептуальное схемное решение полной системы.



5.2.1 Использование драйвера ATAVRMC100

Эта программа была проверена на наборе для разработки приводов мощных двигателей ATAVRMC100. ATAVRMC100 имеет встроенный микроконтроллер AT90PWM3, но возможно использовать его и как мощный драйвер с другим микроконтроллером через интерфейсы EXT_DRV и SENSOR на плате. Разъем EXT_DRV задокументирован в Руководстве Пользователя ATAVRMC100, доступном на web узле Atmel. Важно стереть AT90PWM3 перед использованием ATAVRMC100 с другим микроконтроллером, чтобы гарантировать, что только один микроконтроллер управляет сигналами.

Таблица 5-2 содержит полный список сигналов, которые должны быть связаны с платой ATAVRMC100. Обратите внимание, что сигнал GND связан с отрицательным выводом шунта, который фактически напрямую связан с общим проводом. Рисунок 5-3 показывает графическое представление разъема EXT_DRV.

Таблица 5-3 показывает необходимые подключения для интерфейса с датчиками холла. Графическое представление этого интерфейса показано на Рисунке 5-4.

Таблица 5-2. Подключение EXT_DRV ATAVRMC100.

Имя сигнала	ATmegax8 pin	сигнал ATAVRMC100	номер вывода EXT_DRV
UH	PD6	H_A	1
UL	PD5	L_A	2
VH	PB1	H_B	3
VL	PB2	L_B	4
WH	PB3	H_C	5
WL	PD3	L_C	6
Ток	PC4	V шунт+	7
GND	GND	V шунтt-	8

Рисунок 5-3. Разъем EXT_DRV.

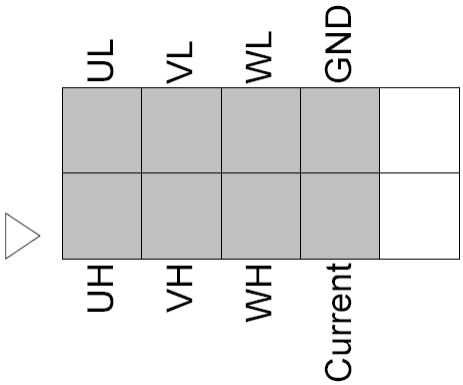
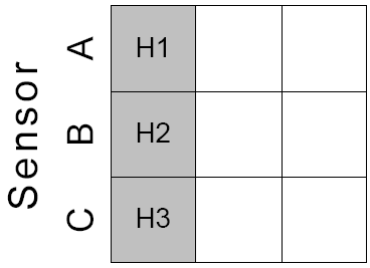


Таблица 5-3. Интерфейс датчиков холла ATAVRMC100.

Имя сигнала	Вывод ATmegax8	Сигнал ATAVRMC100
H1	PCO	Sensor A
H2	PC1	Sensor B
H3	PC2	SensorC

Рисунок 5-4 Подключение датчиков холла



6 Выходные графики

Рисунки 6-1 и 6-2 показывают сгенерированные волны, при вращении вперед и назад с различными скоростями. На осциллографе использовалась функция усреднения, чтобы сгладить формы волны. Оба графика показывают, с верху вниз: напряжения фазы для U, V, W, межфазное напряжение U-V, датчики холла H1, H2 и H3.

Рисунок 6-1 Вращение вперед, 750 RPM

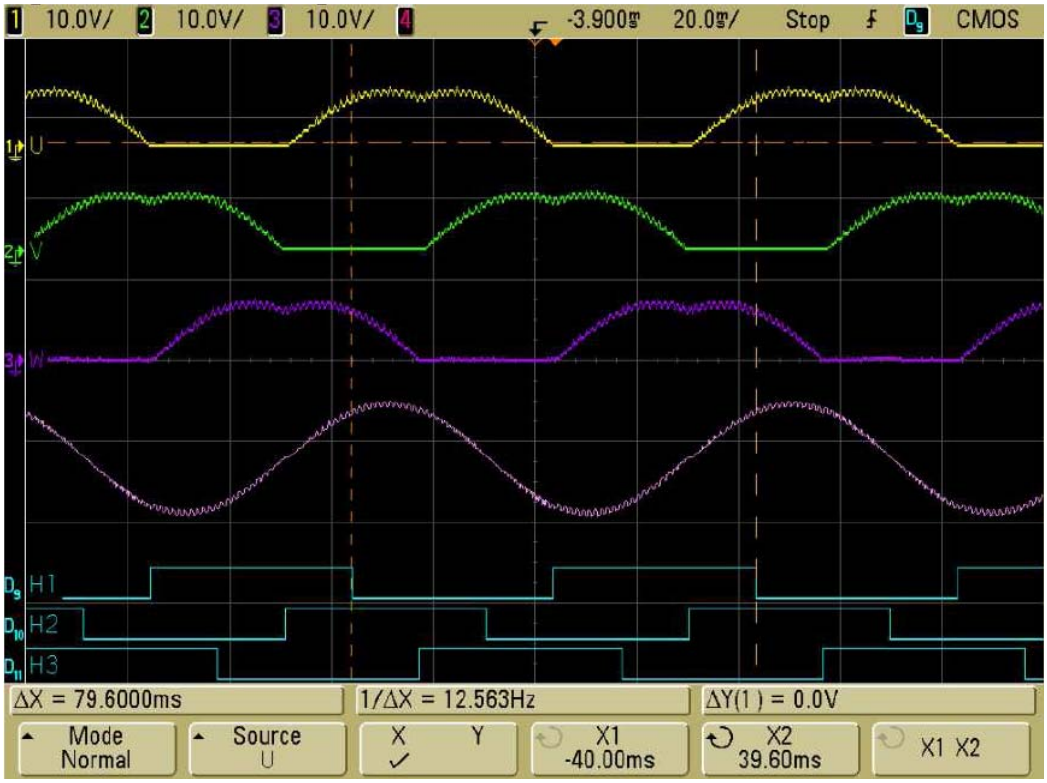
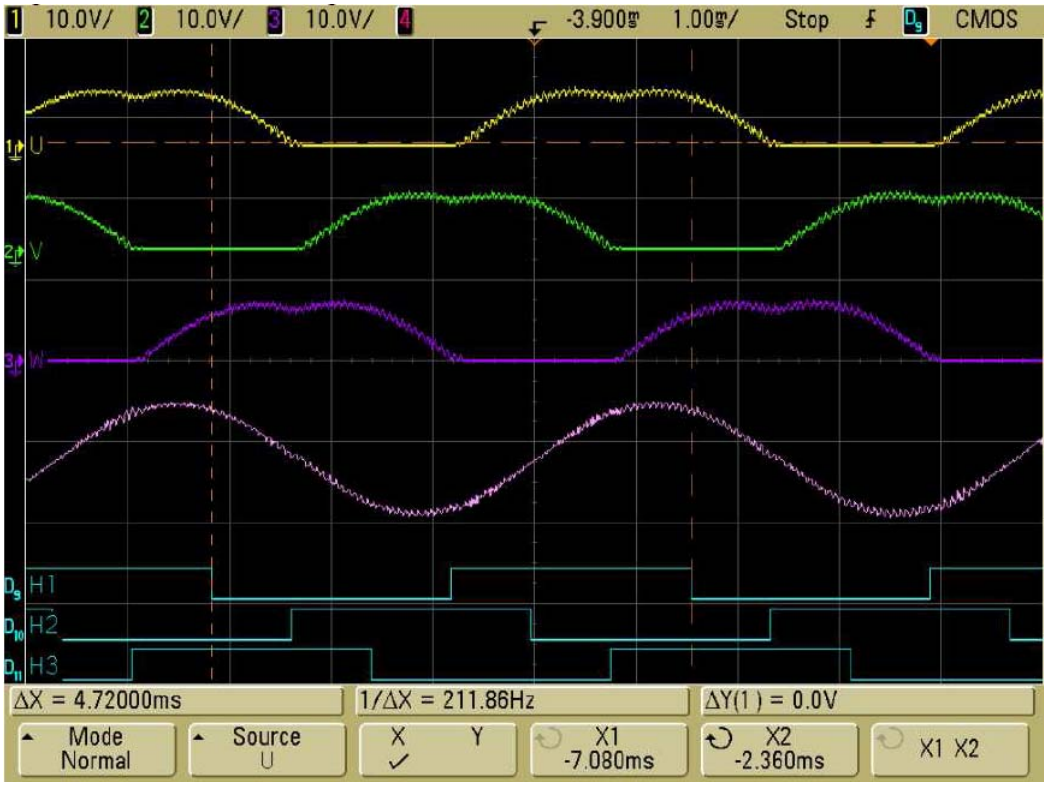


Рисунок 6-2 Вращение назад, 12660 RPM



7 Размер программы и характеристики

Таблица 7-1 показывает использование памяти программ и ОЗУ. Обратите внимание, что размер программы включает почти 900 байт, занятых таблицами значений.

Таблица 7-1. Использование памяти

	Программа	ОЗУ
Открытая система управлением скоростью	~2.6kB	~70B
Управление скоростью PID контроллером	~3.5kB	~90B

8 Ссылки

1. Valentine, R., Motor control electronics handbook, 1998, McGraw-Hill.
2. Atmel corporation, ATAVRMC100 hardware user guide rev. B, February 2006 http://www.atmel.com/dvn/resources/prod documents/doc7551.pdf

9 Источники

1. Оригинал этого документа на английском http://www.atmel.com/dyn/resources/prod_documents/doc8010.pdf
2. Программа описанная здесь http://www.atmel.com/dyn/resources/prod_documents/AVR447.zip