KTechLab

George John
george@space-kerala.org
www.space-kerala.org

11 June,20

Оглавление

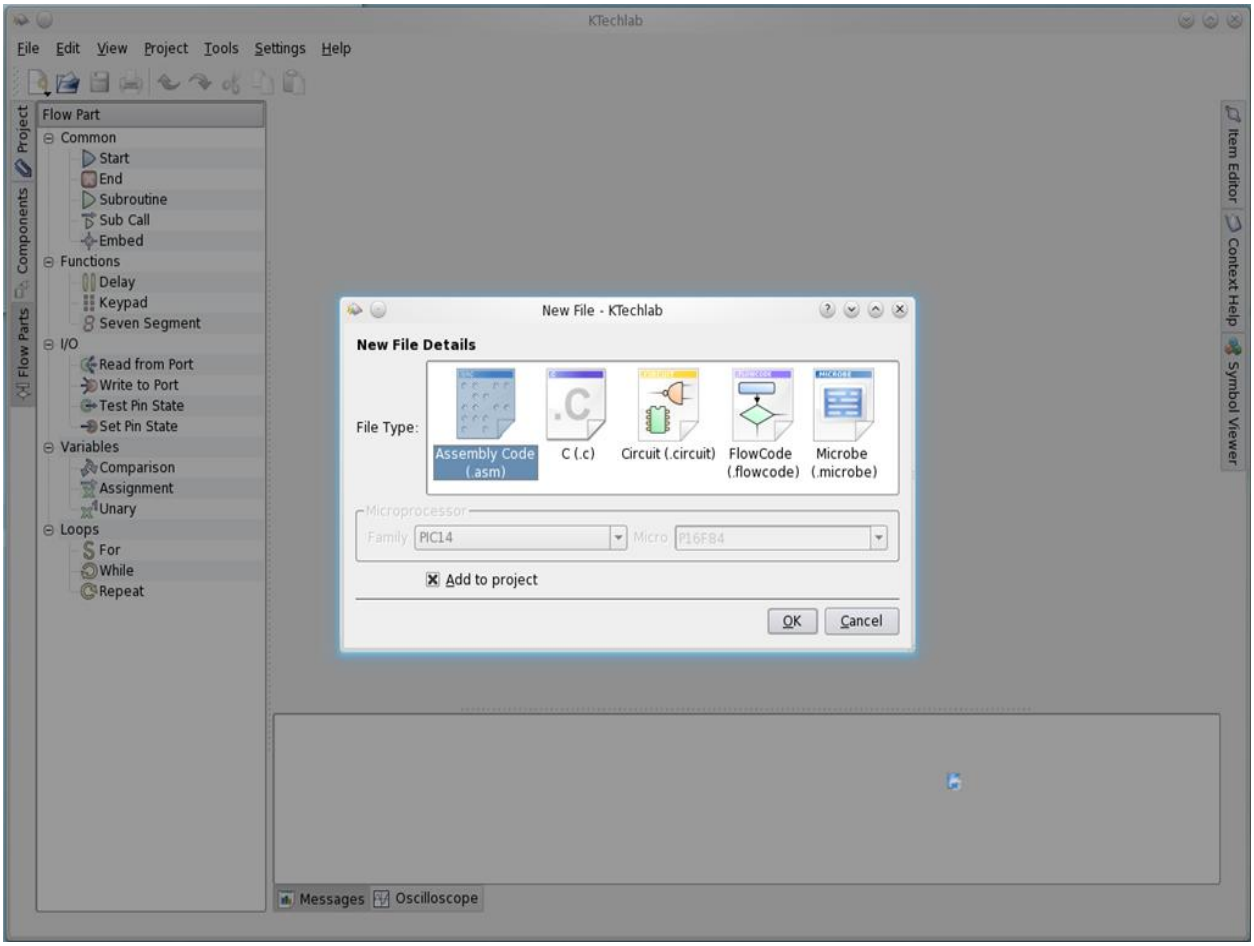
Часть 1. Введение 4
Часть 2. Симуляция электрических цепей 6
2.1 Аналоговые цепи 7
2.1.1 Однополупериодный выпрямитель 7
2.1.2 Свойство конденсатора 8
2.1.3 Осциллятор 9
2.1.4 Сопротивление 10
2.2 Цифровые схемы 11
2.2.1 Логические вентили 11
2.2.2 Триггер 11
2.2.3 Intergerated Circuit 14
2.3 Микроконтроллер 17
Часть 3. Микроконтроллер PIC16F84 18
3.1 PIC16F84 диаграмма выводов 18
3.1.1 От RA0 до RA4 18
3.1.2 От RB0 до RB7 19
3.1.3 VSS и VDD 19
3.1.4 OSC1/CLK IN и OSC2/CLKOUT 19
3.1.5 MCLR 19
3.1.6 INT 19
3.1.7 T0CK1 19
3.2 Регистры 20
3.2.1 STATUS 21
3.2.2 TRISA и TRISB 21
3.2.3 PORTA и PORTB 21
3.2.4 W 21
3.3 Загрузка программы в микросхему с использованием Pikdev 22
3.3.1 Конфигурирование PIKdev 22
3.3.2 Загрузка программы 22
3.4 Базовая схема включения PIC-контроллера 24
Часть 4. Flow Code 25
4.1 Символы Flowcode 25
4.2 Запись в порты ввода-вывода (I/O) 27
4.3 Чтение из портов I/O 28

4.4 Больше программ 29
Часть 5 . Microbe 34
5.1 Loops (циклы) 34
5.2 Условное ветвление 35
5.3 Примеры кода 36
5.3.1 Запись в порты I/O 36
5.3.2 Чтение из портов I/O 36
5.3.3 Больше программ 38
Часть 6 . Small Device C Compiler 40

Часть 1.

Введение

KTechLab – это Open Source (с открытым кодом) интегрированная среда разработки (IDE) и симуляции схем электроники, и среда программирования PIC микроконтроллеров. Характеризующаяся большими возможностями в работе со схемами – автосоединение и симуляция, общие электронные и логические элементы – KtechLab служит инструментом проверки идей при обучении или в любительской практике. Новый файл создается либо щелчком по кнопке инструментальной панели, либо пунктом New раздела File основного меню, либо нажатием *CTRL+n*. После этого вы получите окно, как показано на рисунке ниже.



Есть пять опций:

- **Assemble Code**

Эта опция позволяет нам писать программу на языке ассемблера. Ассемблер — язык низкого уровня. Утилита программы, называемая ассемблер, используется для трансляции команд ассемблера в машинный код (hex код).

- **C Programming**

Мы можем программировать микроконтроллер, используя язык Си, после выбора этой опции. Код Си компилируется компилятором Small Device C Compiler (SDCC).

- **Circuit**

В этом разделе можно симулировать электрические цепи.

- **Flow Code**

Flowchart (блок-схема), пожалуй, то место, где следует начать программирование микроконтроллеров. Мы можем программировать контроллер (PIC), используя графические языковые блоки.

- **Microbe**

Microbe — это язык программирования похожий на BASIC.

KTechLab поддерживает целый ряд Open Source PIC программаторов, позволяя завершить PIC программу быстрой и легкой ее трансляцией в реальный PIC контроллер.

Часть 2.

Симуляция электрических цепей

Давайте вначале обсудим симуляцию какой-нибудь простой электрической цепи. Чтобы нарисовать схему, выберем новый файл с опцией Circuit. Мы можем выбрать компоненты на панели Components. Если компоненты не доступны, щелкните по закладке Components в левой части боковой панели. Компоненты разделены на семь категорий (0.3.6):

- **Sources (Источники)**

Включает источники напряжения и тока.

- **Discrete (Дискретные)**

Включает резисторы, конденсаторы, диоды, транзисторы и т.д.

- **Switches (Переключатели)**

Включает разные типы переключателей.

- **Outputs (Выходы)**

Включает осциллографический пробник, вольтметр, амперметр, индикаторы, семисегментный дисплей, матричный дисплей и т.д.

- **Logic**

Включает логические вентили, такие как И, ИЛИ, НЕ, исключающее ИЛИ, тактовый генератор и т.д.

- **Connections**

Включает шину, последовательный и параллельный порты и т.д.

- **Integrated Circuits**

Включает разные типы микросхем, включая PIC.

Щелкните по компоненту, который вам нужен, и, удерживая левую клавишу мышки, перетащите его в рабочую область. Для соединения разных компонентов вначале щелкните по свободному концу компонента. Теперь указатель мышки изменился, превратившись в значок «плюс». Передвигайте указатель мышки, удерживая левую клавишу, к компоненту, с которым требуется соединение. Линия приобретает красный цвет. Теперь отпустите левую клавишу мышки,

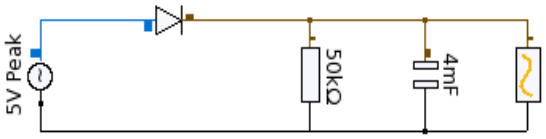
чтобы завершить соединение. Похожим образом меняются значения компонента. Достаточно дважды щелкнуть по компоненту, чтобы появилось текстовое поле на инструментальной панели. Измените значения, используя стрелки вверх и вниз. А чтобы увидеть выходную осциллограмму, присоедините осциллографический пробник Voltage Probe к выходному выводу и щелкните по закладке *Oscilloscope* Oscilloscope на нижней панели.

2.1 Аналоговые цепи

Мы можем симулировать аналоговые цепи, используя KTechLab.

2.1.1 Однополупериодный выпрямитель

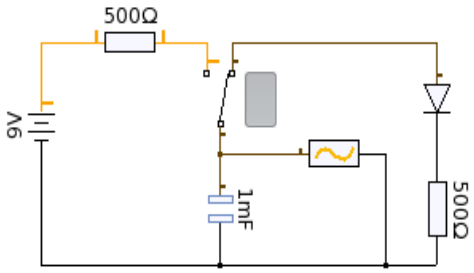
В качестве первого эксперимента получим сигнал однополупериодного выпрямителя. Вначале выберем источник переменного напряжения на панели Components (секция Sources, Voltage Signal). Выберем диод, сопротивление и конденсатор в секции Discrete, а в секции Outputs осциллографический пробник (Voltage Probe). Сделаем соединения, как показано ниже.

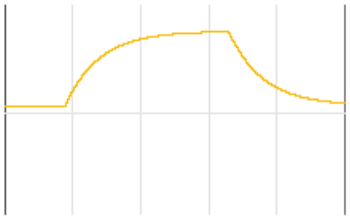


Вид сигнала получается с помощью осциллографического пробника и отображается в нижнем окне вывода.

2.1.2 Свойство конденсатора

Теперь изучим свойство конденсатора, используя KTechLab. Вначале соберем цепь, как показано на рисунке ниже. Активизируем осциллографический пробник, затем будем заряжать и разряжать конденсатор, используя переключатель.

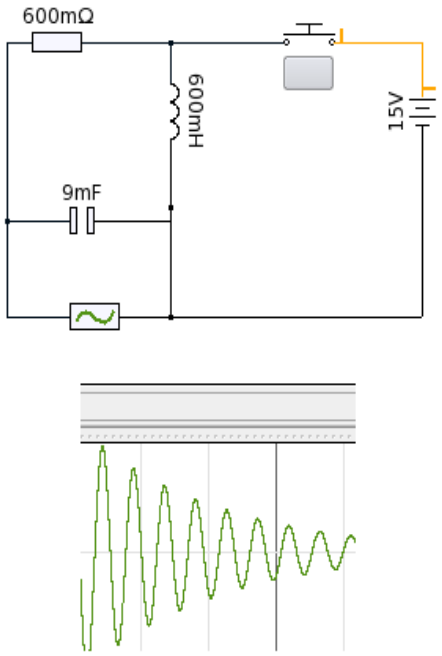




Рассмотрите осциллограмму. Измените емкость и изучите различия в полученных кривых.

2.1.3 Осциллятор

Теперь посмотрим, как происходит осцилляция в колебательном контуре, диаграмма соединений показана ниже.



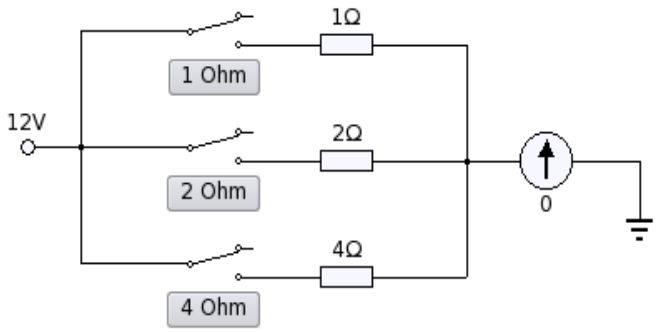
Зарядите конденсатор, нажав и отпустив выключатель. Когда конденсатор заряжен, на одной обкладке больше электронов, чем на другой. Когда же он разряжается через цепь, электроны возвращаются к положительной обкладке, что делает обкладки конденсатора нейтральными или разряженными. Однако этот процесс протекает иначе, если вы разряжаете конденсатор через катушку. Когда ток протекает через катушку, вокруг нее создается магнитное поле. Это магнитное поле генерирует напряжение, пересекая катушку, что меняет направление потока электронов. Благодаря этому конденсатор не разряжается полностью. Чем меньше катушка, тем быстрее разряжается конденсатор. Теперь самое интересное. Когда конденсатор полностью разряжается через катушку, магнитное поле вокруг катушки начинает сжиматься.

Напряжение, индуцируемое этим магнитным полем, перезаряжает конденсатором в обратной полярности. А затем конденсатор начинает вновь разряжаться через катушку, создавая магнитное поле. Процесс продолжается до тех пор, пока конденсатор не разрядится полностью, благодаря сопротивлению.

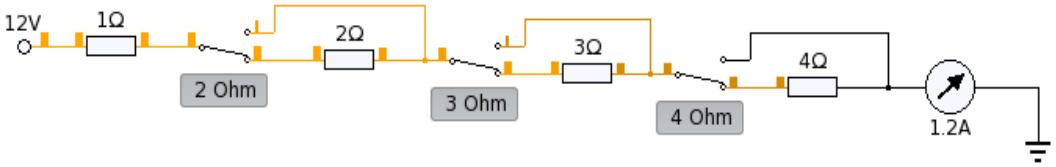
2.1.4 Сопротивление

Найдите ток через сопротивления, когда они соединены параллельно и последовательно. Диаграммы соединения представлены ниже.

Параллельное соединение



Последовательное соединение



2.2 Цифровые схемы

Мы можем симулировать логические схемы, используя KTechLab. Давайте начнем с логических вентилей.

2.2.1 Логические вентили

Логические вентили обрабатывают сигналы, представляющие «истинно» или «ложно». Обычно положительное напряжение +Vs представляет «истинно», а 0V — «ложно». Некоторые базовые вентили обсуждаются ниже.

- **NOT Gate** (вентиль НЕ)

Выход Q становится «истинным», когда вход A в состоянии НЕ «истинно».

- **AND Gate** (вентиль И)

Если входы A и B оба «истинны», тогда выход в состоянии «истинно».

- **OR Gate** (вентиль ИЛИ)

Выход Q в состоянии «истинно», вход A ИЛИ вход B в состоянии «истинно» (или оба входа).

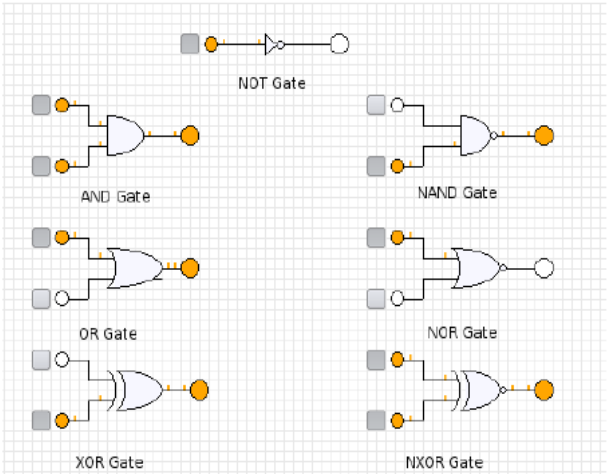
- **NAND Gate** (вентиль НЕ-И)

Если вентиль НЕ подключен к выходу вентиля И, мы получим вентиль НЕ-И.

- **NOR Gate** (вентиль НЕ-ИЛИ)

Если вентиль НЕ подключен к выходу вентиля ИЛИ, мы получим вентиль НЕ-ИЛИ.

Рисунок ниже показывает симуляцию логических вентилей при использовании KTechLab.



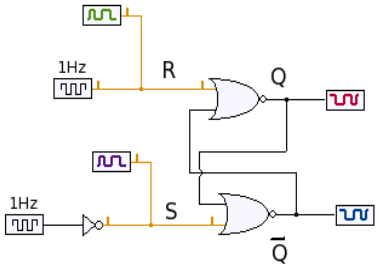
2.2.2 Триггер

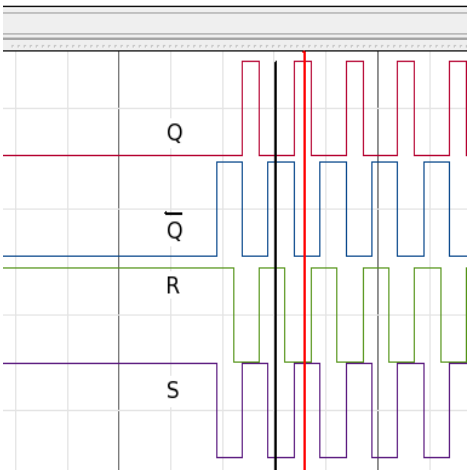
Триггеры — это синхронизируемые, с двумя устойчивыми состояниями, устройства хранения, способные хранить один бит. В этом случае синхронизация означает, что выход меняется с приходом тактового импульса, то есть, выход меняется, когда меняется тактовый сигнал. Есть несколько типов триггеров, наиболее употребительные из которых описаны в следующих параграфах.

Set-Reset (S-R) триггер

SR-триггер (установка-сброс) имеет два входа, называемых вход Set (установка) и вход Reset (сброс, или R).

Он также имеет два выхода, называемых основной Q и его дополнение НЕ-Q. Простое представление SR-триггера можно получить из пары вентилей НЕ-ИЛИ с перекрестными обратными связями, то есть, выход одного вентиля соединен с одним из двух входов другого вентиля и наоборот. Свободный вход одного из вентилей НЕ-ИЛИ используется как R, тогда как вход другого, как S. Выход вентиля с «R» входом используется в качестве Q выхода, а выход вентиля с «S» входом, как НЕ-Q.



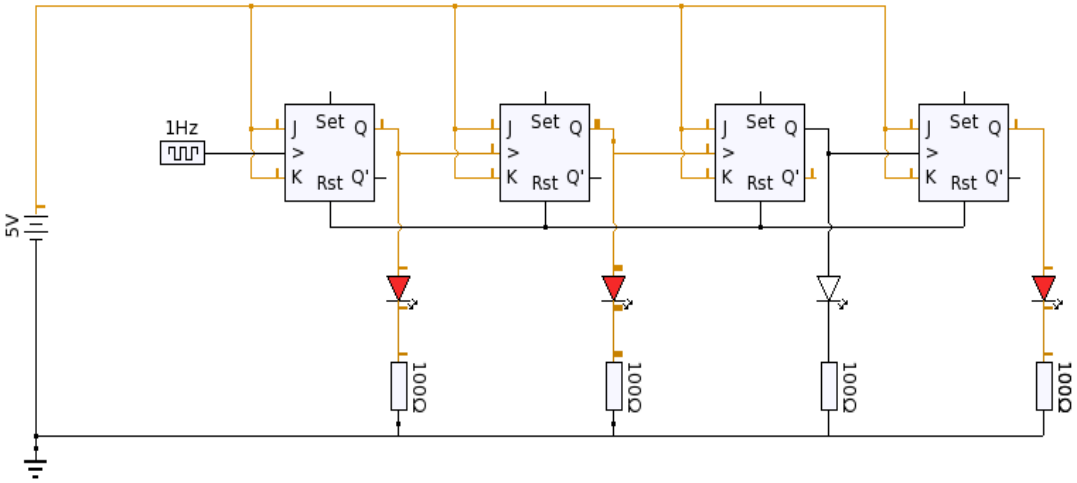


Таким образом, установка выхода Q SR-триггера в «0» требует, чтобы R = 1, а S = 0; в свою очередь, установка Q в «1», требует, чтобы S = 1, а R = 0. В реальных приложениях триггеры тактируются так, чтобы можно было контролировать момент, когда выход меняет свое состояние, откликаясь на изменение состояния входа. Цифровой тактовый вход синхронизируемых триггеров обычно обозначается как C.

JK-триггер

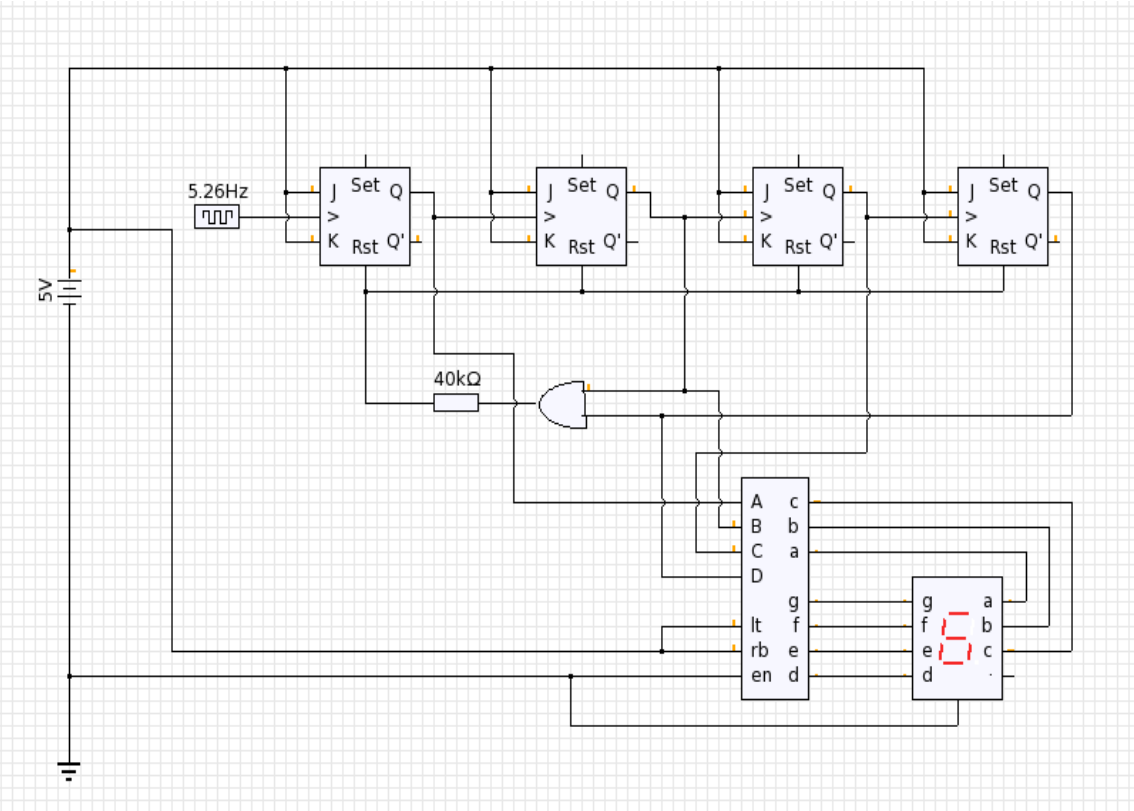
JK-триггер отличается от SR-триггера тем, что его следующее состояние выхода определяется настоящим состоянием и состоянием его входов. Заметьте, что у JK-триггера вход S теперь называется J, а вход R — называется K. Таким образом, у JK-триггера выход не изменится, если оба входа J и K в состоянии «0», но изменится на противоположный, если оба входа примут значение «1».

Соберем простой четырехбитный счетчик, используя JK-триггеры. Схема показана ниже. Счетчик считает от 0000 до 1111, то есть, от 0 до 15.



Отметьте резисторы, включенные в цепь светодиодов. Их назначение — избежать перегрузки по току выходов Q и светодиодов.

Давайте теперь соберем десятичный счетчик, модифицировав предыдущую схему. Предыдущий счетчик считает от нуля до пятнадцати, а мы должны ограничить счетчик так, чтобы он считал от нуля до девяти. Для этого включим вентиль И между выходом второго и четвертого триггера, а выход вентиля И подключим к выводам сброса (Rst) всех триггеров. Когда счетчик досчитает до десяти, выходы второго и четвертого триггеров принимают значение «1» и счетчик будет сброшен в 0000.



На схеме присутствует микросхема, которую можно найти в секции Intergerated Circuit. Некоторые из этих микросхем:

2.2.3 Intergerated Circuit

Интегральные микросхемы появились благодаря экспериментальным открытиям, которые показали, что полупроводниковые устройства могут выполнять функции электронных лам, и благодаря подвижкам в технологии производства полупроводников в середине 20 века. Интеграция большого числа транзисторов в маленькой микросхеме было большим прорывом в сравнении с ручной сборкой схем с использованием дискретных электронных компонентов. Возможности массового производства интегральных схем, их надежность и подвижки в блочном встраивании позволили разработчикам электроники быстро адаптироваться к стандартным микросхемам вместо разработок на дискретных транзисторах.

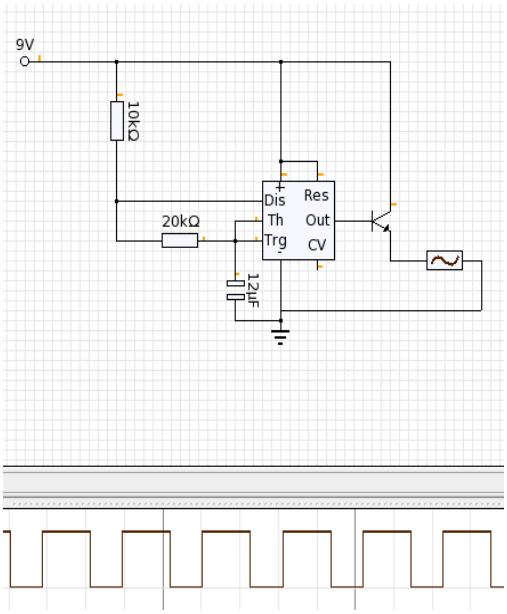
Есть два главных преимущества микросхем над дискретными схемами: цена и характеристики. Цена ниже, поскольку микросхемы со всеми их компонентами «печатаются» как целое методами фотолитографии, а не собираются транзистор за транзистором. Характеристики выше, поскольку компоненты переключаются быстро и потребляют меньшую мощность по причине того, что компоненты меньше и расположены ближе друг к другу. Среди наиболее «продвинутых» микросхем есть микропроцессоры или «ядра», которые управляют всем — от компьютеров до сотовых телефонов и микроволновых печей. Цифровые микросхемы памяти и специализированные интегральные схемы — это примеры другого семейства интегральных схем, которые важны для современного информационного общества. В следующем разделе мы обсудим симуляцию разных микросхем в KTechLab.

PIC

PIC — это семейство микроконтроллеров Гарвардской архитектуры, созданное Microchip Technology на основе PIC1650, первоначально разработанного General Instrument's Microelectronics Division. PIC популярны и у разработчиков, и у любителей, благодаря их низкой цене, большим возможностям, большой пользовательской базе, обширной коллекции описаний применения, доступности дешевых или свободных средств разработки и возможности последовательного программирования (и перепрограммирования с флеш-памятью).

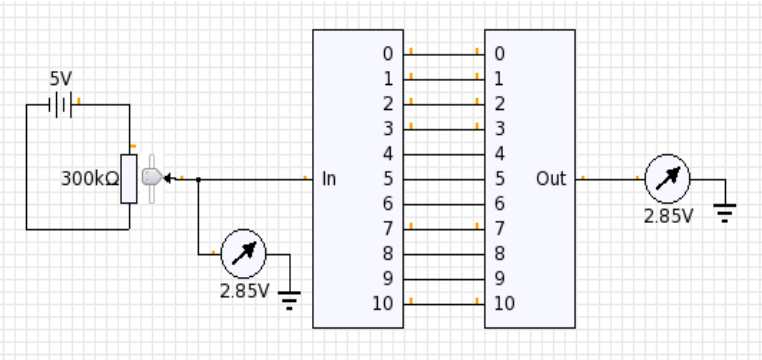
555 таймер

Семейство микросхем 555 таймеров может использоваться либо в моностабильном режиме, либо в автоколебательном. В моностабильном режиме или режиме «одиночного импульса» каждый раз при переключении 555 таймера, его выход переходит в высокое состояние на заданное время, а затем возвращается в низкое состояние до прихода следующего переключающего сигнала. В автоколебательном режиме таймер периодически переключает себя самостоятельно, превращаясь в осциллятор, производящий последовательные импульсы. Рисунок ниже показывает генератор прямоугольных импульсов на таймере 555.



ADC/DAC (АЦП/ЦАП)

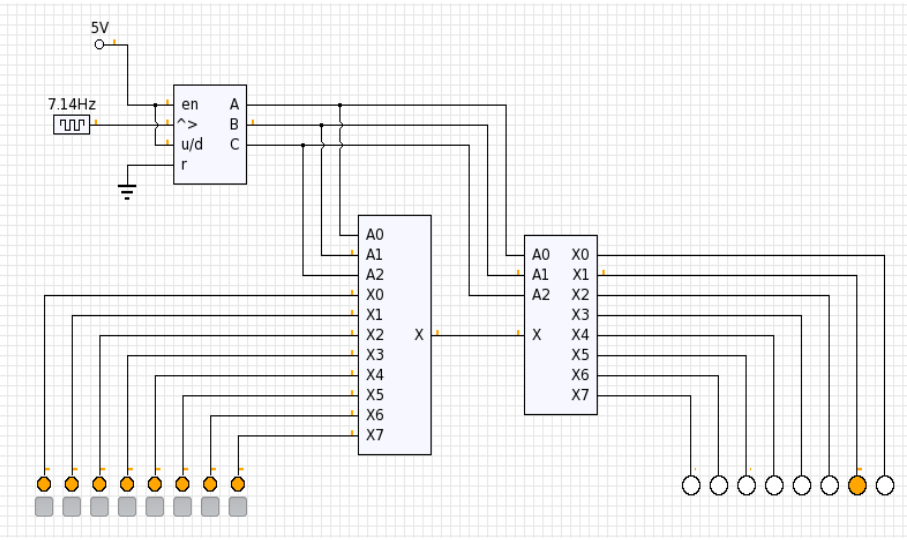
Аналого-цифровой преобразователь (аббревиатура ADC, A/D или A в D) — это электронная внутренняя цепь (internal circuit, i/c), конвертирующая постоянный сигнал в двоичное дискретное число. Обратная операция выполняется цифро-аналоговым преобразователем (DAC). Обычно ADC — это электронное устройство, конвертирующее напряжение входного аналогового сигнала (или ток) в двоичное число. Цифровой выход может быть использован разными схемами кодирования, такими как двоичные и двух-комплементарные двоичные. Рисунок ниже показывает АЦП и ЦАП вместе, АЦП преобразует входное напряжение в цифру, а ЦАП возвращает его в аналоговую форму. Ниже показано, что точность увеличивается с количеством битов преобразования (с увеличением разрешения). Количество битов может быть увеличено двойным щелчком по микросхеме и изменением значения в окне свойств.



Multiplexer и Demultiplexer

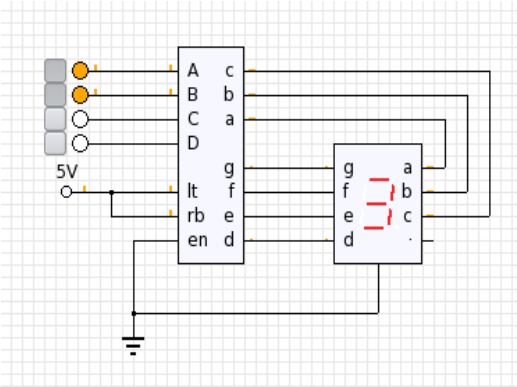
Мультиплексор (или mux, можно найти и термин muldex для комбинации мультиплексор-демультиплексор) — это устройство, которое выбирает один из многих источников данных и выводит этот источник в единственном канале.

Демультиплексор (или demux) — это устройство, имеющее единственных вход, которые выбирает одну из множества выходных линий данных и соединяет единственный вход с выбранной выходной линией. Мультиплексор часто используется с комплиментарным демультиплексором на приемном конце.



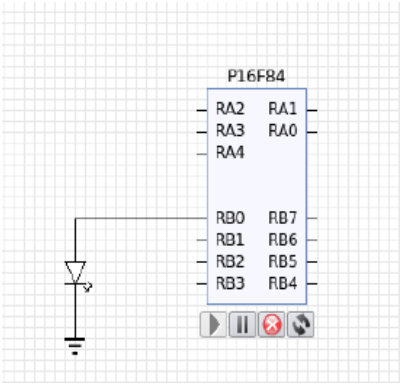
BCD to Sevensegment Decoder (дешифратор семисегментного индикатора)

BCD to Seven Segment Decoder принимает данные в BCD форме и преобразует их в выход для семи сегментов. Микросхема осуществляет декодирование, связанное с активацией нужных сегментов (a-g), требуемых для представления двоичного числа на входе. Есть 4 двоичных входа в декодер и семь выходов для сегментов (a-g).



2.3 Микроконтроллер

Создайте новый файл с опцией Circuit. Затем из меню компонентов выберите PIC, сделайте требуемые соединения и дважды щелкните по иконке PIC, чтобы загрузить требуемую программу в PIC. Щелкните кнопку проигрывания на панельке под микросхемой, чтобы запустить симуляцию. Рисунок ниже показывает пример схемы. В нее мы загрузим программу для мигания светодиодом на выходе Port B,0 микроконтроллера с паузой 100 мс. Мы обсудим программирование микроконтроллера позже.



Часть 3.

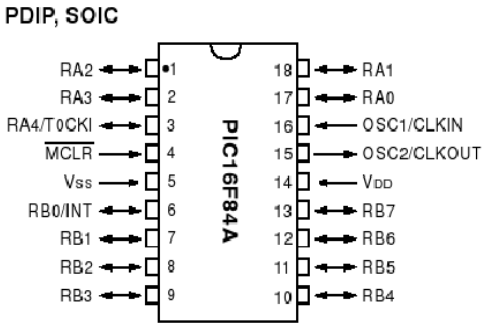
Микроконтроллер PIC16F84

Давайте немного поговорим о микропроцессорах. Микропроцессор — это программируемый электронный компонент, который объединяет функции центрального процессора (CPU) в единственной полупроводниковой интегрированной микросхеме (IC). Микропроцессор состоит из:

- Ядра
- Памяти (и ROM, и RAM)
- Нескольких параллельных цифровых входов и выходов (I/O)

Микропроцессор может выполнять хранимый набор инструкции, чтобы осуществить определенную пользователем задачу. Микропроцессор, когда он интегрирован с таймером, последовательным I/O, АЦП, дает нам микроконтроллер. Microchip производит серию микроконтроллеров, называнную PIC. Сейчас на рынке доступны разные серии микроконтроллеров. Мы сконцентрируем внимание на PIC16F84. Когда вы изучите, как программировать один тип PIC, изучение остальных станет много легче.

3.1 PIC16F84 диаграмма выводов



Мы переберем все выводы, поясняя, для чего используется каждый из них.

3.1.1 От RA0 до RA4

RA — это двунаправленный порт. То есть, он может быть сконфигурирован и как входной, и как выходной. Номер, следующий за RA, это номер бита (от 0 до 4). Так что, мы имеем один 5-

битовый направляемый порт, где каждый бит может конфигурироваться как Input (Вход) или Output (Выход).

3.1.2 От RB0 до RB7

RB — это второй двунаправленный порт. Он ведет себя также, как RA, исключая то, что он 8-битовый.

3.1.3 VSS и VDD

Это выводы питания. VDD — положительный, а VSS — отрицательный или 0V. Максимальное напряжение питания может быть 6V, минимальное — 2V.

3.1.4 OSC1/CLK IN и OSC2/CLKOUT

Это выводы, где мы присоединяем внешний тактовый генератор для тактирования микроконтроллера.

3.1.5 MCLR

Этот вывод используется для очистки локализации памяти внутри PIC (то есть, когда мы хотим перепрограммировать его). При обычном использовании вывод подключается к положительному полюсу питания.

3.1.6 INT

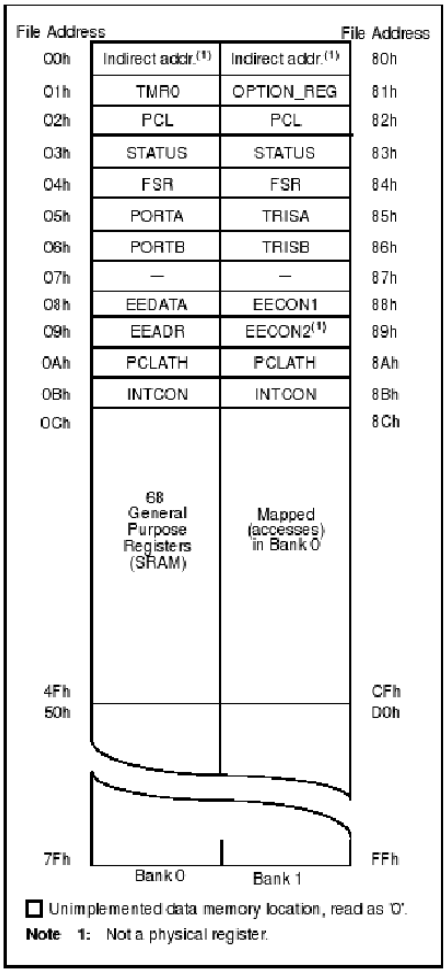
Это входной вывод, который может отслеживаться. Если вывод переходит в высокое состояние, мы можем осуществить перезапуск программы, остановить ее или реализовать другую единичную функцию по нашему выбору. Мы не будем этим увлекаться. Ex:Interrupt используется в случае сканирования клавиатуры. Когда одна из клавиш нажимается, вывод прерывания становится высоким, и тогда микроконтроллер останавливает текущую работу программы и начинает выполнять программу, определяющую нажатую клавишу.

3.1.7 T0CK1

Это еще один вход тактирования, оперирующий со внутренним таймером. Он оперирует изолированно от основного тактового генератора. И опять, мы не будем часто пользоваться этим.

3.2 Регистры

Регистр — это место внутри PIC, куда мы можем записывать, откуда можем читать или и то, и другое. Думайте о регистре, как о клочке бумаги, куда вы можете заглянуть или записать информацию. Рисунок ниже показывает карту регистров внутри PIC16F84.



Первое, что бросается в глаза — это разделение на два Bank 0 и Bank 1. Bank 1 используется для управления текущей операцией PIC, например, сказать PIC, какой бит порта A используется на ввод, а какой на вывод. Bank 0 используется для операций с данными. Вот пример: положим, мы хотим сделать один бит порта A «высоким». Вначале нам нужно перейти в Bank 1 для установки правильного бита или вывода порта A, как выходного. Затем мы возвратимся в Bank 0 и пошлем логическую 1 на этот вывод.

Самые употребительные регистры в Bank 1, которые мы собираемся использовать, это STATUS, TRISA и TRISB. Первый позволяет нам вернуться в Bank 0, TRISA позволяет выбрать, какой вывод порта A выходной, а какой входной, TRISB позволяет сделать это же для порта B.

Регистр SELECT в Bank 0 позволяет нам переключаться на Bank 1. Давайте взглянем на эти три регистра:

3.2.1 STATUS

Для перехода от Bank 0 к Bank 1 мы обратимся к регистру STATUS. Мы сделаем это, установив бит 5 регистра STATUS в 1. Для переключения обратно к Bank 0 мы установим бит 5 регистра STATUS в 0.

3.2.2 TRISA и TRISB

Регистры находятся по адресу 85h и 86h соответственно. Для программирования вывода на выход или на вход мы просто отправляем 0 или 1 для нужного бита в регистр. Это можно сделать в двоичной или шестнадцатеричной форме, но двоичная нагляднее. Если вы не слишком хорошо знакомы с преобразованием двоичных чисел в шестнадцатеричные и наоборот, используйте научный калькулятор.

Итак, порт A имеет 5 выводов, соответственно, 5 битов. Если вы хотите задать один из выводов на вход, отправьте 1 в соответствующий бит. Если хотите задать вывод на выход, отправьте 0 для этого бита. Биты организованы так же, как и выводы, другими словами бит 0 — это RA0, бит 1 — RA1, бит 2 — RA2 и т. д. Рассмотрим пример. Положим мы хотим установить RA0, RA3 и RA4 на выход, а RA1 и RA2 на вход, отправляем 00110 (06h). Заметьте, что нулевой бит справа, как показано:

Выводы порта A	RA4	RA3	RA2	RA1	RA0
Номер бита	4	3	2	1	0
Двоичное число	0	0	1	1	0

Аналогично для порта B:

Выводы порта B	RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0
Номер бита	7	6	5	4	3	2	1	0

3.2.3 PORTA и PORTB

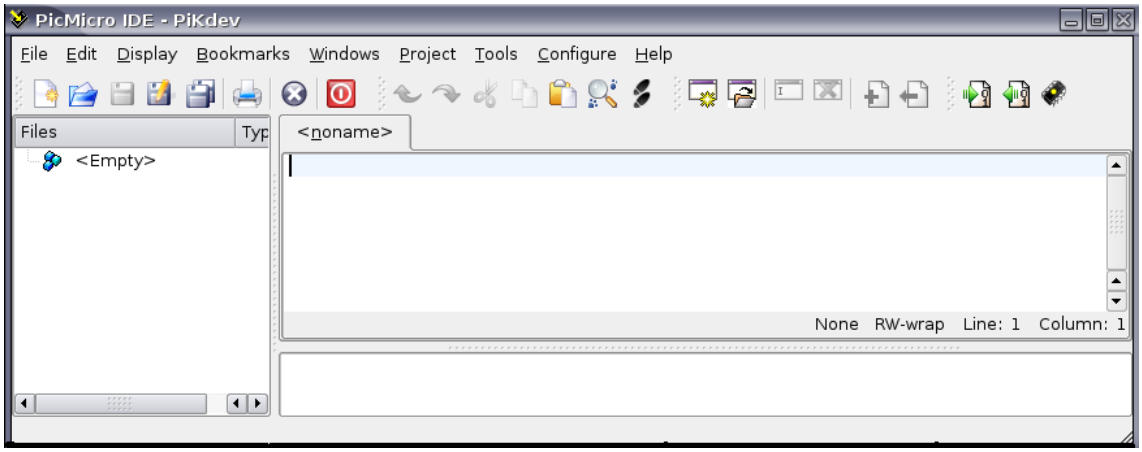
Чтобы установить один из наших выходных выводов в высокое состояние, мы просто посылаем 1 в соответствующий бит регистра PORTA или PORTB. Формат имеет тот же вид, что и для регистров TRISA и TRISB. Для чтения, в каком состоянии находится вывод порта, высоком (1) или низком (0), мы можем выполнить проверку, установлен ли соответствующий бит в 1 или 0. Но перед тем, как показать пример кода, нужно объяснить назначение еще двух регистров, w и f.

3.2.4 W

Регистр W — это регистр общего назначения, в который вы можете отправить любое значение, которое вам нужно. Как только вы присвоили значение регистру W, вы можете прибавить к нему другое значение или вывести его. Если вы присвоили другое значение регистру W, его содержимое переписывается.

3.3 Загрузка программы в микросхему с использованием Pikdev

Конвертируйте программу в hex, используя для этого кнопку  с иконкой ракеты, и сохраните ее. Загрузите программу в микросхему, используя Pikdev (для запуска Pikdev нажмите Alt+F2 и введите pikdev). Снимки экрана с программой Pikdev показаны ниже.



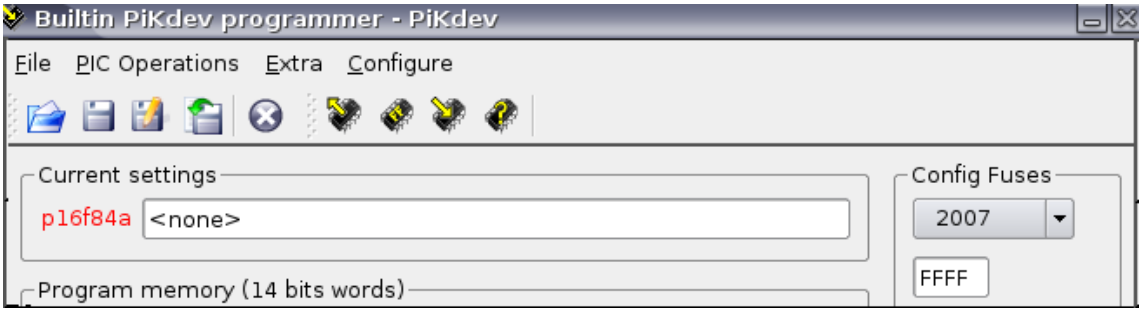
3.3.1 Конфигурирование PIKdev

Выберите в основном меню Configure —> Pikdev (или нажмите Ctrl+Shift+P).

Выберите опцию  Assembler . В окне, которое появится, выберите процессор и для ассемблера, и для дизассемблера, в нашем случае выбираемый процессор p16f84a. Щелкните по клавише Apply и нажмите ОК.

3.3.2 Загрузка программы

Чтобы загрузить программу в микросхему, вначале щелкните по иконке микросхемы в меню  . Вы получите новое окно с четырьмя иконками, как показано ниже.



 => Прочитать данные из микросхемы.

 => Стереть программу в микросхеме.

 => Записать программу в микросхему.

 => Проверить программу в микросхеме.

Config Fuses (слово конфигурации)

2 ☐ WDT **Watchdog timer** (сторожевой таймер) — это механизм, который микроконтроллер использует, чтобы оградить себя от «застревания» программы. Как и в других электрических цепях, в микроконтроллере может проявиться сбой в работе. К сожалению, микроконтроллер также имеет программу, где могут обнаруживаться проблемы. Когда случается подобное, микроконтроллер остановит работу и будет оставаться в таком состоянии, пока кто-нибудь не «сбросит» его. Для таких случаев создан механизм сторожевого таймера. После некоторого периода времени watchdog «сбрасывает» микроконтроллер (микроконтроллер, фактически, производит самосброс). Сторожевой таймер работает на простом принципе: если обнаруживается переполнение таймера, микроконтроллер сбрасывается и начинает выполнять программу с самого начала. Таким образом, сброс произойдет в случае и правильного, и неправильного функционирования. Для предотвращения сброса контроллера в случае правильного функционирования сбросьте сторожевой таймер.

1 ☐ OSC-1
0 ☒ OSC-0 => Выбор осциллятора

Есть несколько основных методов использования тактового генератора для микроконтроллера. Микроконтроллер имеет внутренний осциллятор и два вывода (16F84, выводы 15 и 16) для подключения задающего элемента тактового генератора. Версия PIC16F84A совместима с операциями от DC (постоянный ток) до 20 МГц. Достаточно точный метод обеспечивается с помощью RC (резистор, конденсатор) цепи, подключенной к выводу 15. Режим RC может использоваться, когда частота не критична к таким проблемам, как переход двух входов в высокое состояние, переводящий нужный выход в высокое состояние. Постоянная времени RC цепи определяет частоту, где

t = R*C

и

f = 1/T

Рекомендуется, чтобы при планировании использования RC метода, значение резистора оставалось между 5 кОм и 100 кОм, что требует значения конденсатора, приблизительно 20 пФ.

Сопротивление от 2.2 кОм и меньше может сказаться на стабильности осциллятора вплоть до его остановки. А слишком большое сопротивление, как 1 МОм, может привести к излишней чувствительности осциллятора к шумам, влажности и утечкам.

Кварцевый резонатор дает наибольшую точность. С кристаллом требуется использовать два конденсатора, приблизительно по 22 пФ (см. список ниже). Microchip рекомендует следующие значения конденсаторов для кварцевого осциллятора. Большая емкость увеличит стабильность осциллятора, но и увеличит время его входа в рабочий режим.

Режим	Частота	Конденсатор	
LP	23 кГц	68	100 пФ
LP	23 кГц	68	100 пФ
LP	200 кГц	15	33 пФ
XT	100 кГц	100	150 пФ
XT	2 МГц	15	33 пФ
XT	4 МГц	15	33 пФ
HS	4 МГц	15	33 пФ
HS	10 МГц	15	33 пФ

RC, LP, XT или HS опции выбираются установкой или сбросом OSC-0 и OSC-1 в слове конфигурации (config-Fuses)

Режим	OSC-1	OSC-0
LP	0	0
XT	0	1
HS	1	0
RC	1	1

Вначале очистите данные в микросхеме, щелкнув по кнопке Erase  . Выберите hex-файл, предназначенный для загрузки в микросхему, и щелкните по кнопке Programming  .

3.4 Базовая схема включения PIC-контроллера

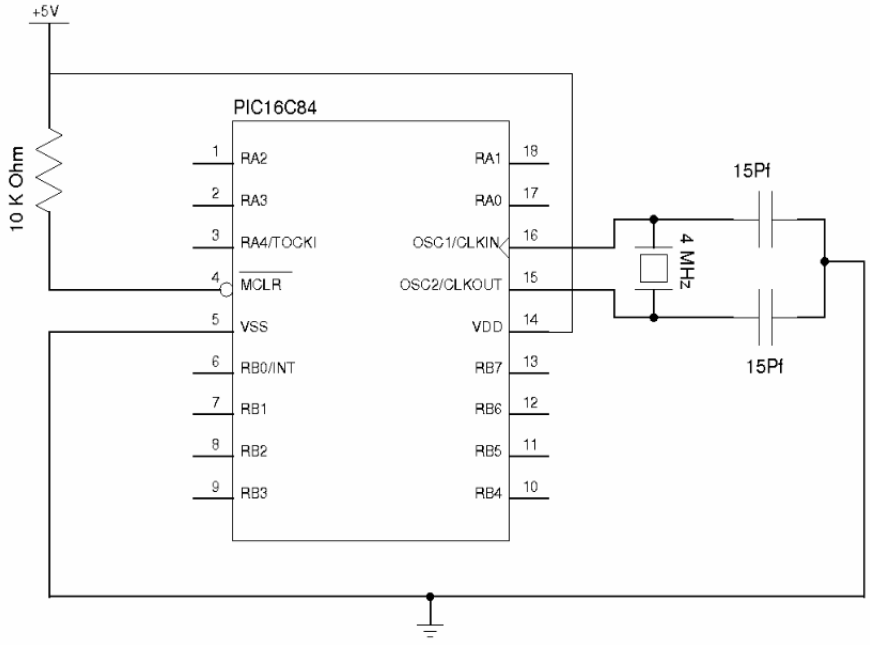


Рисунок выше показывает основную схему включения PIC микроконтроллера (для любой PIC серии). Кварц 4 МГц подключен к выводам OSC 1 и OSC 2. MCLR подключен к положительному выводу питания +5В через резистор 10 кОм.

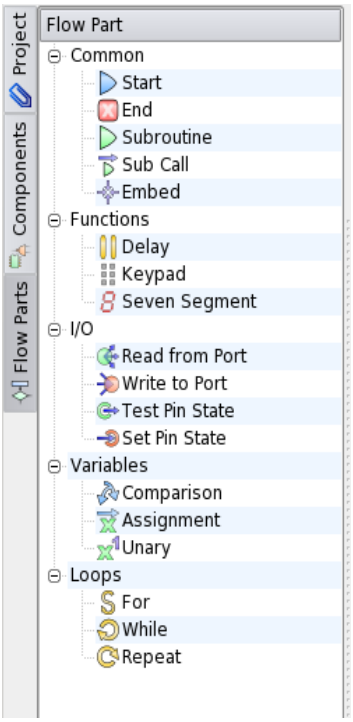
Часть 4.

Flow Code

Давайте обсудим программирование с помощью flowcode (графическое программирование). Щелкните по иконке newfile и выберите опцию FlowCode из окна подменю. Вы увидите картинку с микросхемой PIC-контроллера в верхнем левом углу рабочего окна, она показывает начальные установки (это не часть блок-схемы). Вы можете изменить эти начальные установки, щелкнув по каждому из выводов PIC, чтобы включить или выключить их. Щелкнув и перетащив выводы влево или вправо, вы зададите, будет ли вывод входным или выходным. Программа нуждается в начале и конце, что вы можете сделать, перетащив элементы Start и End с панели графических элементов программы в рабочую область. Затем вы можете перетащить любые другие блоки программы в рабочую область и соединить их.

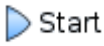
4.1 Символы Flowcode

Символы flowcode похожи на символы блок-схемы. Окно элементов Flow part состоит из пяти секций:



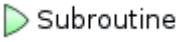
- **Common**

Содержит элементы Start, Stop, Subroutine (подпрограмма), Sub Call (вызов подпрограммы) and Embed (вставка).

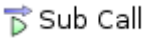


=> Начало программы.

=> Конец программы.



=> Позволяет программисту разделить программу на небольшие части, с которыми легче работать. Подпрограмма может быть вызвана много раз в одной и той же программе, что избавляет от необходимости повторять код.



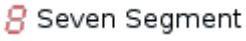
=> Вызов подпрограммы.

- **Functions** (функции)

Содержит элементы Delay (задержка), Keypad (клавиатура) и Sevensegment (семисегментный индикатор).

=> Задерживает процесс на заданное время (пауза).

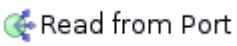
=> Чтение данных с клавиатуры. Перед использованием щелкните по кнопке Advanced... на рисунке PIC и создайте определение карты выводов (pin map).



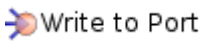
=> Отображает цифру с помощью семи сегментов. Эта функция конвертирует цифру в BCD формат.

- **I/O** (входы и выходы)

Содержит Read from Port (читать из порта), Write to Port (записать в порт), Test pin state (проверить состояние выводов) и Set pin state (установить состояние выводов).



=> Читать данные из порта в переменную.



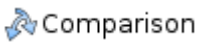
=> Записать данные в порт.

=> Используется для проверки состояния отдельных выводов.

=> Для установки состояния отдельных выводов.

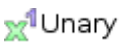
- **Variables** (переменные)

Содержит Comparison (сравнение), Assignment (присвоение) и Unary (унарные).



=> Сравнение переменных.

=> Присвоение значения переменной.



=> Выполняет унарные операции, такие как сдвиг влево, сдвиг вправо, увеличение и уменьшение на единицу.

- **Loops** (циклы)

Содержит циклы For, While и Repeat

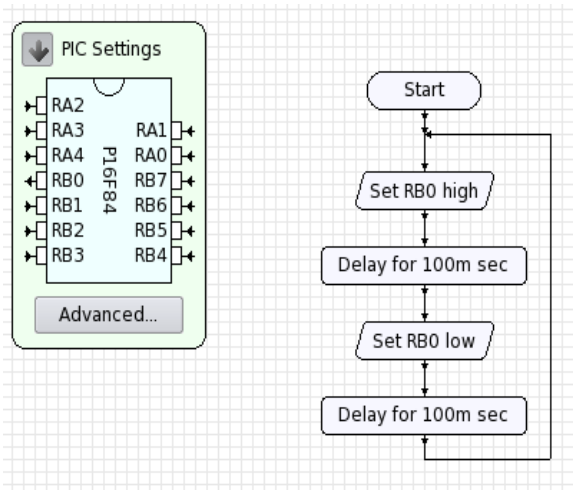
Циклы могут создаваться для выполнения блока кода заданное число раз (7 или 17, или 70). Альтернативно, циклы могут создаваться для повторения блока кода пока заданное условие не изменится к желаемому состоянию. Например, цикл может выполняться до тех пор, пока условие не изменится с «истинно» на «ложно». В этом случае блок кода при выполнении должен обновлять проверяемое условие, чтобы цикл был завершен в некоторой точке. Если условие проверки не меняется где-то внутри цикла, цикл никогда не будет прекращен (бесконечный цикл). Это может приводить к логическим ошибкам.

Перетаскивание подпрограммы добавляет диаграмму с большим окном, в котором вы можете расположить другие программные элементы. Ограничим рассмотрение микроконтроллером PIC16F84A в следующих разделах.

4.2 Запись в порты ввода-вывода (I/O)

Порты микроконтроллера могут конфигурироваться как Input (входные) или Output (выходные). Мы можем записать данные в выходной порт. Нарисуем блок-схему для установки и сброса вывода PORTB,0 с интервалом в 100 миллисекунд. Вначале мы должны установить вывод PORTB,0 как выходной. Это можно сделать двумя способами:

- Щелкнуть по стрелке RB0 (на рисунке микросхемы) и потянуть ее влево.
- Щелкнуть по кнопке Advanced... (на рисунке микросхемы) и установить вывод PORTB,0 как выходной, переустановив последний бит регистра TRIS порта B. Нарисуем блок-схему и сохраним ее.



Как вы можете видеть, элемента End в программе нет. Это делает выполнение кода нескончаемым. Если вы вставите End в блок-схему, код выполнится единожды и не даст увидеть что-либо на выходе. Для симуляции блок-схемы следуйте процедуре, описанной в разделе «Симуляция электрических цепей. Микроконтроллер».

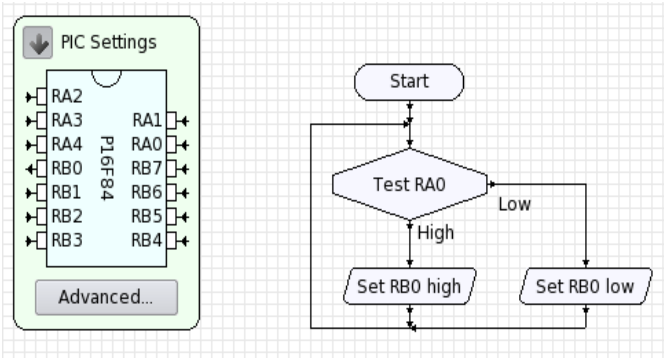
Установка и сброс PORTB,0 эквивалентны записи 1 и 0 в порт B. Поскольку порт B восьмибитовый, мы можем записать от 0 до 255 в PORTB. Если мы запишем в порт число восемь, будет установлен вывод RB3 порта B, как показано в таблице ниже:

Вывод порта B	RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0
Номер бита	7	6	5	4	3	2	1	0
Двоичное	0	0	0	0	1	0	0	0

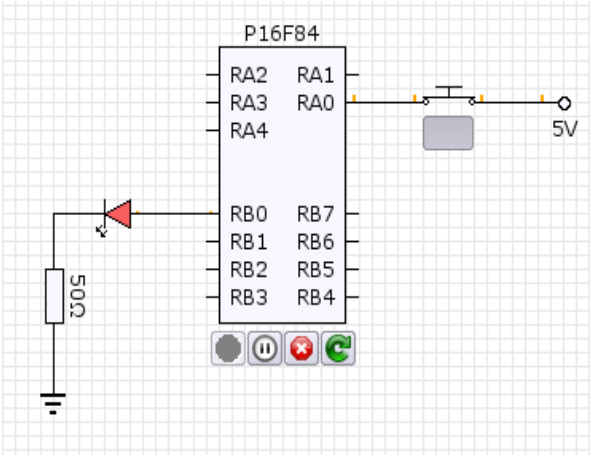
4.3 Чтение из портов I/O

Мы можем читать данные из порта, когда он сконфигурирован для ввода. Нарисуем блок-схему для проверки состояния входа PORTA,0 и установки PORTB,0, если вход в высоком состоянии и сброса в противном случае.

Поскольку по умолчанию порт конфигурируется на ввод, нет необходимости конфигурировать вывод PORTA,0 как входной. Сконфигурируем нулевой вывод PORTB как выходной, щелкнув по стрелке на PORTB,0 и потянем ее влево. Блок-схема показана ниже:



Сохраните блок-схему и симулируйте ее. Схема для симуляции кода показана ниже.



4.4 Больше программ

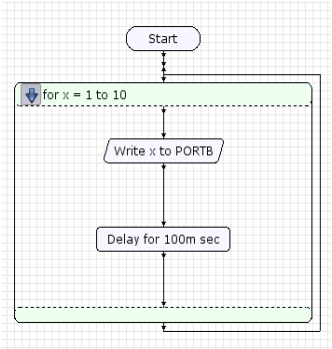
Нарисуем блок-схему для отображения двоичного эквивалента чисел от 0 до 9.

В отличие от десятичной системы достаточно только двух цифр 0 и 1 для представления чисел в двоичной системе. Двоичная система играет главную роль в компьютерной науке и технологии. Первые 20 чисел в двоичной нотации: 1, 10, 11, 100, 101, 110, 111, 1000, 1001, 1010, 1011, 1100, 1101, 1110, 1111, 10000, 10001, 10010, 10011 и 10100, — оригинал которых лучше воспринимается, когда они переписаны в следующем виде:

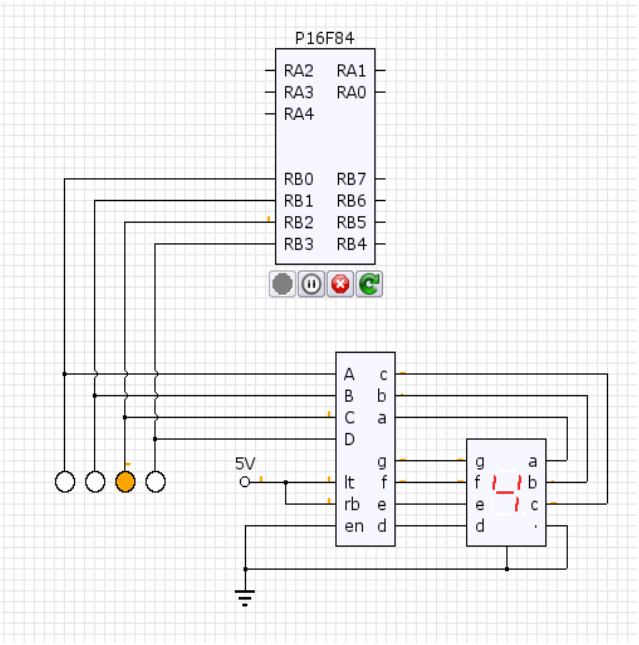
1: 00001	11: 01011
2: 00010	12: 01100
3: 00011	13: 01101
4: 00100	14: 01110
5: 00101	15: 01111
6: 00110	16: 10000
7: 00111	17: 10001
8: 01000	18: 10010
9: 01001	19: 10011
10: 01010	20: 10100

В случае цифровой схемы уровень высокого напряжения, например, 5 В представлен «1», а уровень низкого напряжения представлен «0» вольт.

Вначале мы должны выбрать порты: в данном случае выбран PORTB. Поскольку это восьмибитовый порт, команда PORTB = 1 эквивалентна записи 00000001 в PORTB, то есть, десятичное число конвертировано в двоичное и затем записано в порт. Максимально возможное число, которое можно записать в порт — это 11111111 = 225. Аналогично, в случае порта A (5 бит) максимальное число, которое может быть записано в порт, это 11111 = 31. Установим выводы порта B как выходные (OUTPUT) и нарисуем блок-схему, как показано ниже:

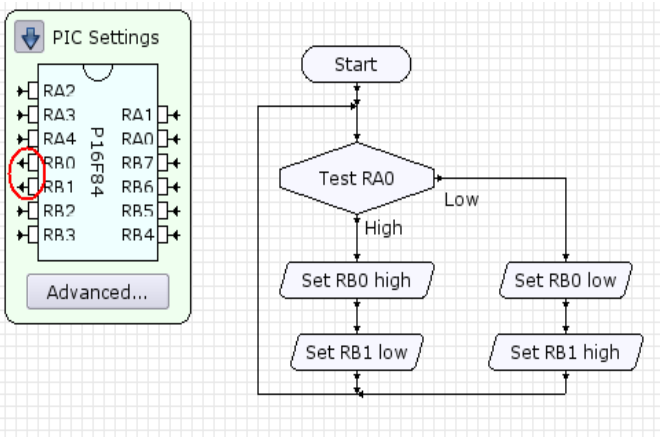


В данном случае мы используем цикл for. Цикл будет выполняться до тех пор, пока значение x не станет равно девяти, а когда x станет больше девяти, тогда произойдет выход из цикла. Схема представлена ниже:

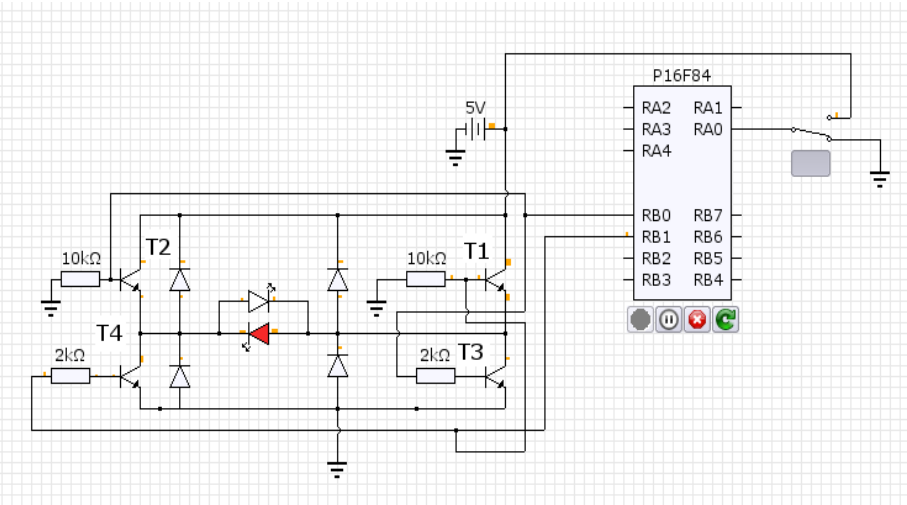


Программа для управления мотором постоянного тока с изменением направления вращения в зависимости от состояния, высокое или низкое, вывода PORTA,0.

Мотор постоянного тока может вращаться влево или вправо за счет изменения полярности источника питания. Выберем два вывода PORTB как выходные, а вывод PORTA,0 как входной. Когда вход в высоком состоянии, установим вывод PORTB,0 в высокое состояние, вывод PORTB,1 в низкое, и наоборот.



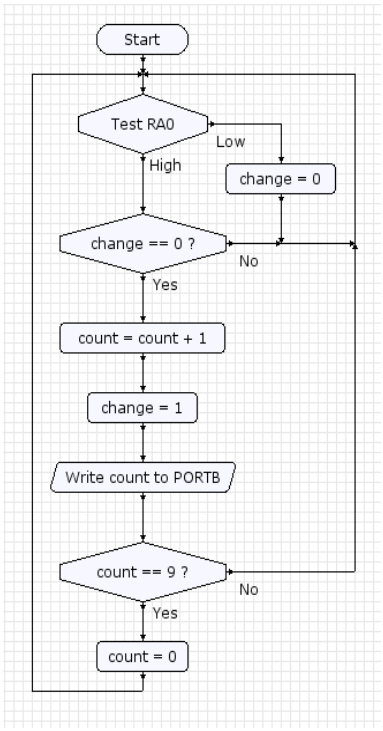
Поскольку мотор постоянного тока потребляет большой ток, мы должны использовать дополнительные транзисторные схемы для подключения мотора. Схема показана ниже:



Когда RB0 устанавливается в высокий уровень, транзисторы T2 и T3 открываются; ток протекает: *батарейка->T2->мотор->T3 и земля*. Когда в состояние с высоким уровнем переходит RB1, открываются транзисторы T1 и T4; ток протекает: *батарейка->T1->мотор->T4 и земля*. Диоды служат для защиты транзисторов от противоэдс.

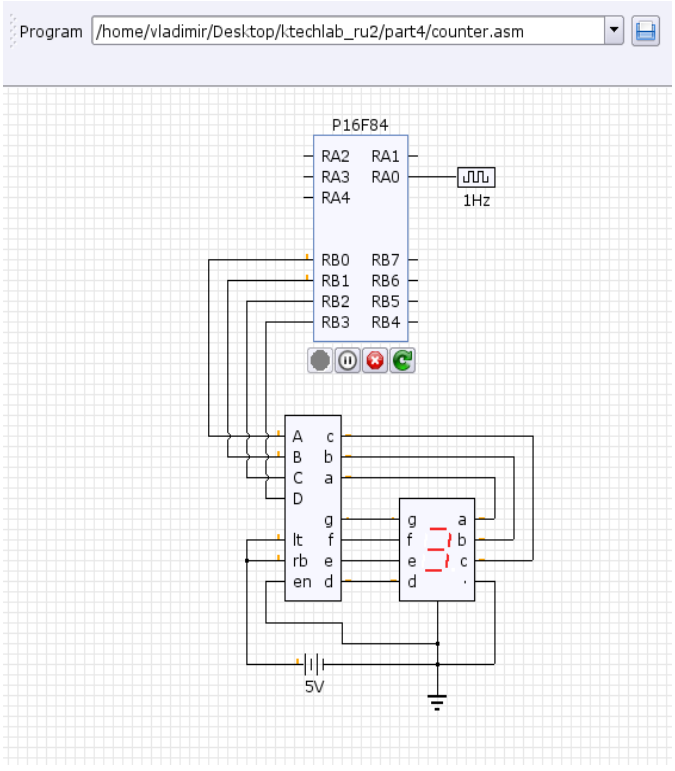
Разработка счетчика с использованием микроконтроллера (счет от 0 до 9).

Вывод PORTA,0 выбран в качестве входа, а PORTB предназначен на вывод.

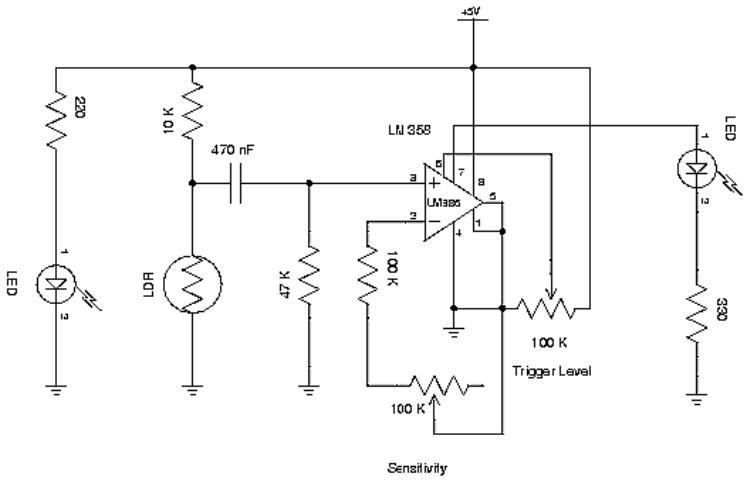


Логика программы: Объявляем две переменные, count и change. Когда RA0 в высоком состоянии, count увеличивается на единицу (count = count + 1), иначе нет. Рассматривайте источник входного сигнала, как генератор прямоугольных импульсов с частотой 1 Гц. Вход будет в высоком состоянии в течение 500 миллисекунд, но контроллер работает на частоте 4 МГц, и count может увеличиваться более одного раза за один импульс. Вначале мы проверим, обнулено ли change, и если да, то count увеличится на единицу, но и change примет значение единицы. Переменная change обнулится только тогда, когда генератор на входе имеет низкий уровень.

Схема симуляции:



На практике вместо генератора прямоугольных импульсов применяют подходящую цепь, как, например, показано ниже. При прерывании света от светодиода выход переходит в высокое состояние. Такую схему можно использовать для подсчета количество людей, входящих в комнату и т. д.



Необходимые компоненты: LM 358, фоторезистор, светодиод (красный), сопротивления (100 кОм, 47 кОм, 10 кОм, 330 Ом и 220 Ом), подстроечные резисторы (100 кОм - 2) и конденсатор (470 нФ).

Часть 5 .

Microbe

Microbe — это простой язык, который помогает легко программировать микроконтроллеры. В настоящее время microbe поддерживает PIC 16F84, 16F628, 16F627 и 16F877. Конструкции языка даны ниже.

5.1 Loops (циклы)

- **Goto**

label:
{
statement 1
statement 2
statement 3
.....
statement n
}
goto label

- **For**

Цикл for может выполнять блок кода фиксированное или заданное число раз. Его синтаксис следующий:

for *инициализация* to *проверяемое условие*
{
блок кода
}

Цикл, представленный выше, будет выполняться до нарушения проверяемого условия.

- **While**

Цикл while используется для выполнения блока кода до тех пор, пока некое условие «истинно». Если условие «ложно» изначально, блок кода совсем не будет выполняться. Цикл while проверяет условие перед выполнением блока кода, так что иногда цикл может никогда не выполняться, если не встречается начальное условие. Его синтаксис представлен ниже.

while *условие*
{
выполняемый блок, если условие «истинно»
}

- **Repeat**

Этот цикл также выполняет блок кода до тех пор, пока удовлетворены условия. Разница между циклом « repeat ...until» и «while» в том, что цикл while проверяет условие до начала выполнения содержания цикла; а цикл «repeat» проверяет условие после выполнения кода хотя бы раз. Как отмечено выше, если проверяемое условие «ложно» для цикла «while» вхождения в код не будет. Поскольку условие проверяется «внизу» цикла «repeat», его блок кода всегда выполняется хотя бы однажды. Его синтаксис показан ниже.

repeat
{
блок кода, выполняемый пока условие «истинно»
}
until *условие*

5.2 Условное ветвление

if *условие* then
{
выполняемый блок, если условие «истинно»
}
else
{
выполняемый блок в противном случае
}

5.3 Примеры кода

Прежде всего, следует задать тип микроконтроллера, что выполнено в заявлении P16F84. Сконфигурируем порты на вход или на выход установкой или сбросом битов в регистрах TRIS.

5.3.1 Запись в порты I/O

Мы уже обсуждали запись в порты ввода-вывода. Теперь давайте посмотрим, как это можно сделать, используя microbe. Напишем программу на языке microbe, устанавливающую и сбрасывающую вывод PORTB,0 с интервалом в 100 миллисекунд.

```
P16F84
TRISB = 0
PORTA = 0
PORTB = 0
setpin:
    PORTB.0 = high
    delay 100
    PORTB.0 = low
    delay 100
goto setpin
end
```

В программе выше мы использовали переход GOTO к метке *setpin*, delay 100 означает Delay (задержать) выполнение программы на 100 миллисекунд,

PORTB.0 = high означает установить вывод PORTB,0 в высокое состояние.

PORTB.0 = low означает установить вывод PORTB,0 в низкое состояние.

5.3.2 Чтение из портов I/O

Программа, читающая состояние вывода PORTA,0 и устанавливающая вывод PORTB,0, если вывод порта A в высоком состоянии и наоборот.

Зададим вывод PORTA,0 как входной, PORTB как выходной с помощью TRISA = 1 и TRISB = 0, соответственно. Очистим порты, используя команды PORTA = 0 и PORTB = 0. Проверим вывод PORTA,0 с помощью условия if: если состояние высокое, установим PORTB,0, иначе сбросим PORTB,0. Программа показана ниже.

```
P16F84
TRISA = 1
TRISB = 0
PORTA = 0
PORTB = 0
testpin:
if PORTA.0 is high then
{
    PORTB.0 = high
}
else
```

```
{
    PORTB.0 = low
}
goto testpin
end
```

5.3.3 Больше программ

Программа, отображающая двоичный эквивалент от нуля до девяти.

Выберем PORTB как выходной. Программа представлена ниже.

```
P16F84
TRISB = 0
PORTB = 0
forloop:
for x = 1 to 9
{
    PORTB = x
    delay 100
}
goto forloop
end
```

Примечание: PORTB = x означает запись x в порт B.

Программа для управления мотором постоянного тока с изменением направления вращения в зависимости от состояния, высокое или низкое, вывода PORTA,0.

```
P16F84
TRISB = 4
PORTB = 0
testpin:
if PORTA.0 is high then
{
    PORTB.0 = high
    PORTB.1 = low
    goto testpin
}
else
{
    PORTB.0 = low
    PORTB.1 = high
    goto testpin
}
end
```

Разработка счетчика с использованием микроконтроллера (счет от 0 до 9).

Как и в случае блок-схемы, мы так же выбираем RA0 как входной вывод, а PORTB как выходной. Код показан ниже.

```
P16F84
count = 0
x=0
TRISB = 0
PORTA = 0
PORTB = 0
testpin:
if PORTA.0 is high then
{
    if x == 0 then
    {
        count = count+1
        x=1
        PORTB = count
        if count == 8 then
        {
            count = 0
        }
        goto testpin
    }
    else
    {
        goto testpin
    }
}
else
{
    x=0
    goto testpin
}
end
```

Часть 6 .

Small Device C Compiler

SDCC (Си компилятор небольших устройств) — это свободный, с открытым кодом, многоцелевой (retargetable), оптимизирующийся ANSI-C компилятор, разработанный Sandeep Dutta для 8-битовых микропроцессоров. Текущая версия подходит к базирующимся на Intel MCS51 микропроцессорам (8031, 8032, 8051, 8052 и т. д.), вариантам Dallas DS80C390, базируемым на HC08 (когда-то Motorola) и Zilog Z80 микропроцессорам. Он может предназначаться и для других микропроцессоров — поддержка Microchip PIC и Atmel AVR развивается. Весь исходный код компилятора распространяется на основании GPL. SDCC использует ASXXXX и ASLINK, многоцелевые с открытым кодом ассемблер и компоновщик. SDCC имеет развитой язык, приспособленный для использования с разными микроконтроллерами и эффективно поддерживает оборудование.

Если у вас есть начальные знания о программировании на языке Си, можно легко программировать микроконтроллеры, используя SDCC. Давайте рассмотрим небольшую программу, читающую данные из порта A и выводящие их в порт B.

```
/* Define processor and include header file. */
# define 16f84
# include”pic/pic16f84.h”
int x;
void main(void)
{
    TRISA =0x1F;
    TRISB =0x00;
    while(1)
    {
        x=PORTA;
        PORTB = x;
    }
}
```

Для компиляции программы используйте команду: *sdcc -mpic14 -p16f84 filename.c*. Это сгенерирует asm, hex и много сопутствующих файлов кода. Для симуляции этой программы используйте KTechLab, двойным щелчком по элементу PIC открыв окно и загрузив ваш файл с расширением cod (filename.cod).