Занятие 2

Счетчики. Регистры. Прерывания.

Понятие счетчика.

Что такое счетчик? Ну очевидно из названия – это некоторая сущность, которая что-то считает. С точки зрения теории автоматов (есть такая, лежит в основе проектирования вычислительных систем), счетчик – это конечный автомат. Мы не будем заострять внимания на формальных определениях, а попробуем интуитивно определить понятие счетчика.

Итак, счетчик – это автомат, который что-то считает. Что он может считать? Да все, что подчиняется счету. Что может считать счетчик в вычислительной машине или микропроцессорной системе? Например, он может считать байты, биты – единицы информации. Например, вы передаете из одного узла системы в другой узел некоторую информацию, закодированную в N байт данных. Очевидно, что за передачу необходимого количества байт будет отвечать некоторый счетчик, который должен передать между узлами строго N байт, отсчитывая и пересылая их один за одним.

Также счетчик может считать, например, события. Простой пример – двойной щелчок по левой кнопке мыши. Для реализации этого компонента интерфейса используется счетчик событий. Произошло одно нажатие – произошло одно событие, два нажатия – два события. Дальнейшая логика этого счетчика посылает управляющий сигнал при наступлении двух событий (произошел двойной клик) и затем сбрасывает счетчик (устанавливает его на 0).

Ну и наконец, счетчик может считать время. Такты процессора, работающего на известной частоте, временные интервалы между событиями, период наступления которых известен – основа для такого счета. Пример. Наш микроконтроллер Atmega328 работает на частоте 16МГц, или 16 000 000 Гц. Это означает, что период между тактами частоты равен 1/16000000 секунд. Очевидно, если мы решаем задачу выполнять какое-то действие 1 раз в секунду, мы должны иметь счетчик, который способен посчитать до 15999999, после достижения 15999999 сформировать некоторый сигнал и сброситься в 0. Этот сигнал будет формироваться строго один раз в секунду.

На этом примере, мы произвели действие, которое повсеместно применяется в цифровой технике – мы *поделили частоту*. А сам счетчик в этом случае оказался *делителем частоты*. И действительно: на вход счетчика поступает сигнал с частотой f_0 = 16Мгц, а на выходе мы получаем сигнал с частотой f_1 = 1Гц. Т. е. мы поделили нашу частоту в 16000000 раз. Раз есть делитель частоты, то должен быть и *умножитель частоты*. Но как часто бывает, ломать – не строить, и делить – не умножать, и умножители частоты строятся гораздо сложнее.

Программные и аппаратные счетчики

В микропроцессорных системах иногда условно различают *программный* и *аппаратный* счетчики. Когда говорят про *программный счетчик*, то имеют ввиду просто кусочек программы, в которой реализован счетчик. Могли догадаться, что в основе такого кусочка лежит обычный цикл! Цикл *for* или цикл *while* с какой-то логикой внутри тела цикла.

Когда говорят про *аппаратный счетчик*, то имеют ввиду автомат, реализованный в виде специализированных микросхем или кусочков микросхем, который не подлежит программированию, а может быть только сконфигурирован. Внутри микроконтроллера аппаратный счетчик является элементом периферии по отношению к ядру, это один из периферийных узлов. И это очень важный узел.

Как правило, в периферии микроконтроллеров реализовано несколько счетчиков, потому что они имеют широчайшее применение.

Может возникнуть вопрос: а зачем микроконтроллеру аппаратные счетчики на периферии, если тот же функционал можно реализовать с помощью программы, реализуемой на ядре микроконтроллера? Вероятные ответы на этот вопрос и размышления по этому вопросу –одно из заданий этого урока.

Далее, говоря о счетчиках на этом занятии, мы будем подразумевать аппаратные счетчики в периферии микроконтроллера.

В микроконтроллере ATmega328 установлено 3 счетчика общего назначения. Они отличаются по функционалу. Самый продвинутый – это Timer/Counter 1. Вторым по функционалу можно считать TimerCouter 2, и еще чуть проще – Timer/Counter 0. На прошлом занятии мы давали ссылку на Datasheet микроконтроллера Atmega328. Найдите в техническом паспорте соответствующие разделы, посвященные этим счетчикам и посмотрите на их схемы. Кажется сложными, но это только кажется).

Управление счетчиками. Регистры.

Всеми периферийными узлами контроллера принято управлять через регистры. *Регистр* – это специальный тип памяти (*регистровая память*), который используется в основном для конфигурирования различного оборудования. Каждое периферийное устройство в микроконтроллере имеет свой набор регистров, которые отображаются в общее адресное пространство микроконтроллера, чтобы программист мог записать туда необходимые значения или считать оттуда значения. Как правило, каждый регистр имеет свое назначение и каждый бит регистра тоже. Например. Пусть у нас есть счетчик разрядностью 16 бит. Очевидно, что этот счетчик может считать от 0 до (2^{16}-1), т. е. до 65535. Но что если, нам нужно, чтобы счетчик считал до 3599 ((3600 – 1) - столько секунд в часе, или какого-то иного числа, меньше чем 65535), а после снова сбрасывался в 0? Для этого, у счетчика есть регистр, который определяет, на каком значении будет происходить сброс счетчика. Ясно, что это 16-разрядный регистр, в который разработчик запишет число 3599_{10} = $0xE0F_{16}$ = 111000001111_2. И тогда, как только счетчик достигнет значения 3599, компаратор во внутренней логике счетчика сформирует сигнал, который сбросит счетчик (установит его в 0).

Следующий пример. Разработчику системы может понадобиться, чтобы на внешней линии микроконтроллера при сбросе счетчика формировался управляющий сигнал. И для этого у счетчика также есть специальный бит в специальном регистре – если программист запишет при конфигурировании в этот бит «1», то на выходе контроллера будет формироваться управляющий сигнал, если запишет в этот бит «0», то управляющий сигнал формироваться не будет.

Т.о. регистры – это интерфейс взаимодействия с периферией микроконтроллера. В том числе и с счетчиками. Самое прекрасное во всем этом, что в этой огромной работе не принимает никакого участия ядро микроконтроллера, которое может быть занято гораздо более сложными высокоуровневыми задачами.

Немного отвлечемся и узнаем, что у ядра микроконтроллера тоже есть регистры. Есть специальные регистры (у разных архитектур свои), а есть *регистры общего назначения* – РОН-ы (Регистр Общего Назначения). Регистры – это самый быстрый и самый дрогой тип памяти. Они находятся непосредственно в ядре, скорость обмена с ними максимальна для любой микропроцессорной системы. В процессорах ваших компьютеров и сотовых телефонов большое множество регистров.

Подумайте и скажите, если регистры самый быстрый и дорогой тип памяти, то какой тип памяти следует за ними (чуть медленнее, но чуть более дешевый)? Подсказка – объем этой памяти также является одной из ключевых характеристик микропроцессора. Однако, этот тип памяти пожалуй не применяется в классических микроконтроллерах.

Прерывания.

Чтобы дальше вести повествование, нам нужно ввести еще одно важнейшее понятие из мира вычислительной техники – *прерывания*.

Давайте рассмотрим такую задачу. Пусть мы настроили счетчик, как в примере выше и этот счетчик считает до 3599. И пусть нам, как разработчикам необходимо, чтобы каждый раз, когда счетчик достигает этого значения, наш микрокотроллер выполнял какое-то важное действие. Например – включал и выключал сигнальную лампу на каком-нибудь маяке (пусть это будет ходовой огонь речного судна). Как это сделать? Для этого, в общем случае, нужно, чтобы счетчик каким-то образом проинформировал ядро микроконтроллера о том, что он досчитал до 3599 и собирается сбросится. Самый простой вариант. У нас есть регистр счета – регистр, в котором счетчик хранит текущее значение счета. Ядро микроконтроллера может с каким-то интервалом считывать значение из этого регистра и сравнивать его с имеющимся в программе опорным значением 3599. Но у этого подхода есть два больших минуса. Во-первых, точность времени включения/выключения маяка будет определяться частотой, с которой ядро будет опрашивать регистр счета и производить сравнение. А во-вторых, чем чаще оно будет это делать, тем точнее будет интервал времени включения/выключения маяка, но тем больше ресурсов на простой глупый опрос регистра будет тратить «могуче» ядро.

Было бы здорово, чтобы ядро «не бегало постоянно на кухню проверить готовность борща», а сам счетчик «окликнул» бы ядро, когда достиг установленного значения. И такой механизм есть – прерывания. Внутри микроконтроллера между счетчиком (и другой периферией) и ядром есть специальная линия связи. Когда счетчик или какое-то иное периферийное устройство «подергает» за свою линию, установит на ней сигнал, ядро будет знать о самом факте появления этого сигнала и от какого периферийного устройства пришел этот сигнал (от какого именно счетчика или от иного периферийного узла). И более того. Если в одном из специальных регистров уже самого ядра, установлен бит разрешения прерывания от этого побеспокоившего его периферийного устройства, ядро *прервется* от выполнения своей текущей задачи (что вы там написали ему в коде программы) и начнет на этот сигнал как-то реагировать. Вот почему прерывания называются прерываниями – они прерывают монотонную линейную работу ядра, и заставляют его временно переключиться на выполнение более срочной задачи. Данном примере – включить/выключить маяк.

Как именно ядро будет реагировать на прерывание определяется программистом – в момент прерывания вызывается специальная функция (отдельный кусочек программы, написанной программистом), которая называется обработчиком прерывания. И что вы там в нее напишете – это ваше дело. Что напишете, то и будет делать ядро в этот момент.

На первом этапе нашего проекта, именно благодаря прерываниям осуществлялось считывание сигнала с модуля радиоуправления. Мы тогда использовали два прерывания (две линии, два канала управления). Только не от счетчика и не от какого-то периферийного устройства, а непосредственно, от вывода микроконтроллера. Каждый раз, когда с радиомодуля приходил сигнал, ядро нашего контроллера прерывалось и начинало выполнять обработчик прерывания. ***Откройте код той программы еще раз и постарайтесь осознать его уже с точки зрения только что полученных вами сведений.***

У пытливого ума может возникнуть множество вопросов по этому механизму. Вот вам два из них, ответ на которые вообщем-то один:

1. ***Что будет, если несколько периферийных устройств одновременно начнут «требовать внимания» ядра, одновременно сформируют сигнал на своих линиях прерываний?***
2. ***Что будет, если в то время, пока ядро занято выполнение каких-то срочных действий в обработчике прерывания от одного устройства, его начнет «дергать» по своей линии прерывания другое устройство?***

Поищите ответы на эти вопросы самостоятельно. Чтобы как-то сузить возможные варианты – применительно к архитектуре микроконтроллеров AVR (Вариантов поведения на самом деле может быть несколько – в разных архитектурах реализовано может быть по-разному).