Привет друзья, сегодня мы с вами изучим один из последних уроков по стандартным функциям ардуино, и речь пойдет о прерываниях, очень много кто не понимает эту важную тему и не умеет работать с прерываниями, хотя на деле там все очень просто, и скоро вы в этом убедитесь.

Что же такое прерывание. Возьмём к примеру ардуино нано, у которой есть два прерывания, они выведены на пины 2 и 3. У других моделей ардуино прерываний может быть больше, и они могут находится на других пинах, подробнее смотрите на официальном сайте ардуино. Итак, эти пины могут обрабатываться обработчиком прерываний, который грубо говоря измеряет на них напряжение, и при наличии логического нуля или единицы, то есть 0 или 5 вольт, подает сигнал процессору для выполнения каких то действий. Стоп стоп стоп скажете вы, а чем же этот обработчик прерываний отличается от функции дижтл Рид, если выполняет те же самые на первый взгляд функции? Вот в этом то и вся соль, чтобы считать напряжение функцией дижтл Рид, например отследить нажатие кнопки, вам нужно вписать эту команду в код, и периодически опрашивать нужный пин на наличие напряжения, правильно? И если вы опрашиваете пин в основном в цикле программы луп, то весь код должен выполняться постоянно и нигде не тормозить, то есть даже дилеев быть не должно.

Обработчик прерываний опрашивает пин постоянно, независимо от того какая часть кода сейчас выполняется, есть ли там задержки или какие то длинные и сложные расчеты, вообще без разницы. То есть вы можете например отследить нажатие кнопки в любом месте кода, не обращаясь к пину вообще. Например есть какой то невероятно трудный хреново написанный многоуровневый быдлокод с кучей замкнутых циклов, задержек и пачкой невероятных расчетов, на которые уходят целые секунды выполнения, а вам нужно постоянно следить за датчиком, сигнал с которого идёт очень короткий, взять тот же тахометр на датчике холла. Пихать дижтл рид во все щели кода не очень хорошая идея, надеюсь это очевидно. И тут на помощь приходит прерывание. Это была первая фишка прерываний.

Вторая фишка: обработка сигнала с прерывания происходит практически мгновенно, скорость чтения составляет буквально несколько микросекунд, целых 4, если мне не изменяет память. Именно поэтому в таких своих проектах как хронограф, прибор для измерения скорости пули, и тахометр, прибор для определения частоты вращения, я подключал датчик именно на прерывание, чтобы обеспечить максимальную точность на высокой скорости работы.

Также при отработке например крутильных энкодеров и датчиков холла при помощи дижтл Рид вы никогда не добьётесь стабильной работы, иногда будут проскальзывания и несрабатывания, потому что между опросами пина будут другие куски кода, во время исполнения которых вы не сможете считать сигнал даже если он будет, а прерывание работает всегда.

Третья фишка прерываний: они способны пробудить ардуино от глубокого сна. Я тоже постоянно использую это в своих проектах: ардуино находится в режиме глубокого энергосбережения и просыпается по нажатию на кнопку. Про спящие режимы мы поговорим в следующих уроках.

Итак, три применения для прерываний: опрос кнопок и датчиков в любом месте кода с высокой скоростью, а также пробуждение системы ото сна.

Для использования прерывание нужно подключить и настроить, для этого есть команда attachInterrupt, где в скобках указываются номер прерывания, имя вызываемой функции и режим работы. Номера прерываний начинаются с 0 и не совпадают с номерами цифровых пинов, я это

подчеркну. Для наглядности вот картинка с номерами прерываний для разных моделей ардуино. Я работаю с ардуино нано, потому что она дешёвая и ее не так жалко спалить. Ну и потому что у меня их много. Допустим я подключаю кнопку к прерыванию номер 1, это пин номер 3. Кнопку подключаю как в уроке про кнопки, вторую ногу в землю, и использую внутренний подтягивающий резистор, режим инпут пуллап. Если сейчас вы ничего не поняли, то смотрите урок про кнопки, там это подробно разбиралось.

Настраиваем прерывание при помощи функции аттач интеррапт. Сначала указываем номер прерывания, у нас это номер 1, а не 3, как хотелось бы написать. Далее идёт имя функции, которая будет вызвана обработчиком. Эту функцию нужно обязательно создать, пусть это будет пустая функция с названием myinterrupt. Название этой функции вписывается как второй параметр подключения прерывания без скобочек. Третий параметр это режим работы, их тут несколько, а именно 4 штуки. Лоу - прерывание срабатывает, когда на пине логический ноль. Райсинг - срабатывает, когда на пин приходит скачок напряжения от логического нуля до логической единицы, например от 0 до 5 вольт. Фоллинг - то же самое, только наоборот, падение напряжения с высокого уровня на низкий. Эти два режима как раз подходят для кнопки. Четвертый режим - ченж, срабатывает при изменении уровня напряжения в любую сторону, эдакое объединение райсинг и фоллинг. Рассмотрим наш пример: пока кнопка не нажата, на пине высокий сигнал, потому что включена подтяжка. Как только я нажму кнопку, на пине будет установлен низкий сигнал, так как кнопка замкнет на пин землю. Значит при нажатии получаем падение логического сигнала, и в настройке прерывания пишем фоллинг, если хотим, чтобы прерывание сработало при нажатии.

Как это работает: при срабатывании прерывания выполняется набор команд, входящих в созданную нами функцию, причем абсолютно неважно, какая часть кода выполнялась в данный момент. Выполнение кода останавливается, выполняетсч набор функций обработчика прерываний, в нашем случае майинтеррапт, и затем продолжается выполнение основного кода с того момента, где было прерывание.

Сам блок функций, который выполняется в момент срабатывания прерывания, имеет ряд ограничений: во первых здесь не работает задержка дилей, вообще. Во вторых, здесь не меняет своего значения результат функции миллис. Он всегда будет такой, какой был при вызове нашей функции. То есть если нам нужно узнать время, то это можно сделать только один раз за вызов. Для простоты можно представить, что блок функций выполняется мгновенно, даже время не успевает пройти. В третьих - чтобы изменить значение какой то переменной, она должна иметь атрибут, иначе работа будет некорректной. Например каждый вызов прерывания я хочу увеличивать значение счётчика, и использовать его значение в основном цикле программы. Для этого переменная счётчика должна иметь атрибут волатайл, ничего сложного. Сделаем: переменная хранит число нажатий. При вызове прерывания увеличивается на единицу. И выводится в основном цикле каждые полсекунды. Я специально ставлю дилей чтобы показать, что прерывание работает всегда.

И ещё: если выводить что то в порт внутри функции отработки прерывания, часть символов может потеряться, так как программа буквально спешит выполнить все как можно быстрее, прям бежит и тапочки с ног слетают.

Для отключения обработчика прерываний есть функция детач интеррапт, думаю тут комментариев не требуется.

Давайте разберем пример с обработкой прерывания и флажком. Создадим флажок с атрибутом волатайл, по умолчанию он равен нулю. В обработке прерывания поднимаем флажок. В основном цикле программы если флажок поднят, шлем в порт строчку, и флажок опускаем. Таким образом можно передать в основной код факт срабатывания прерывания, я обычно именно так и работаю. Также можно запомнить время срабатывания и тоже его выводить, таким образом у меня работает и тахометр и хронограф. Видите, ничего сложного!

Также иногда бывает нужно отключить на некоторое время отработку прерываний, есть функция nointerrupts, которая приостановливает обработку прерываний. И функция interrupts, которая включает их обратно.

На этом сегодня все, спасибо за внимание, всем пока.