1. ФУОЗ НА МИКРОКОНТРОЛЛЕРЕ

Немного банальностей для общего понимания. Типовая блок схема формирователя угла опережения зажигания (ФУОЗ) на микроконтроллере (МК) представлена на рисунке 1.

Рис.1 Блок схема ФУОЗ

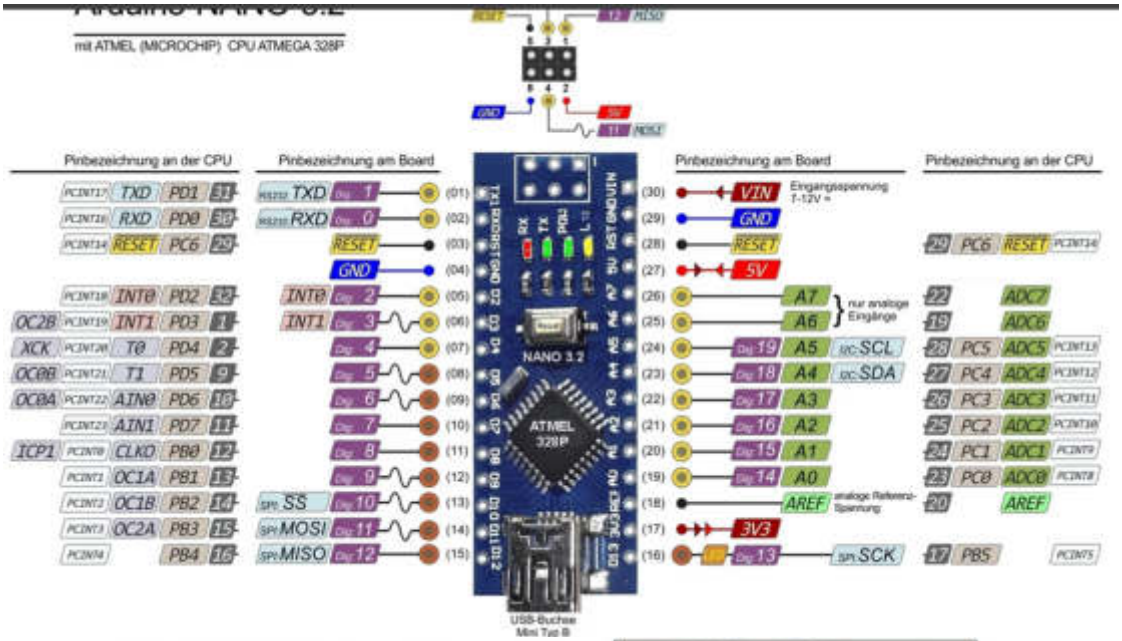
Модель датчиков и схемотехника формирователей сигналов от датчиков для МК не является предметом этой работы. В качестве датчиков могут выступать датчики Холла, оптические или индукционные датчики. У каждых из них есть свои достоинства и недостатки. Сигналы с датчиков поступают на входы МК. Весь алгоритм работы ФУОЗ – программа, которая производит замер характерных временных интервалов и по их значениям производит расчет необходимых временных задержек для работы формирователя импульса зажигания по заданному закону изменения угла опережения зажигания (УОЗ) от оборотов. Удобной частью такой системы является наличие интерфейсного модуля, с помощью которого появляется возможность коррекции параметров устройства или осуществление мониторинга параметров.

Модель МК выбирают из требований обеспечения работоспособности устройства во всем диапазоне числа оборотов двигателя. Быстродействия МК должно хватать для контроля состояния входных линий методом постоянного периодического опроса. Такой способ контроля обусловлен соображениями помехозащищенности. Естественно, что это МК с высокой тактовой частотой. Существуют даже готовые операционные системы для таких устройств с характерными постоянными времени опроса.

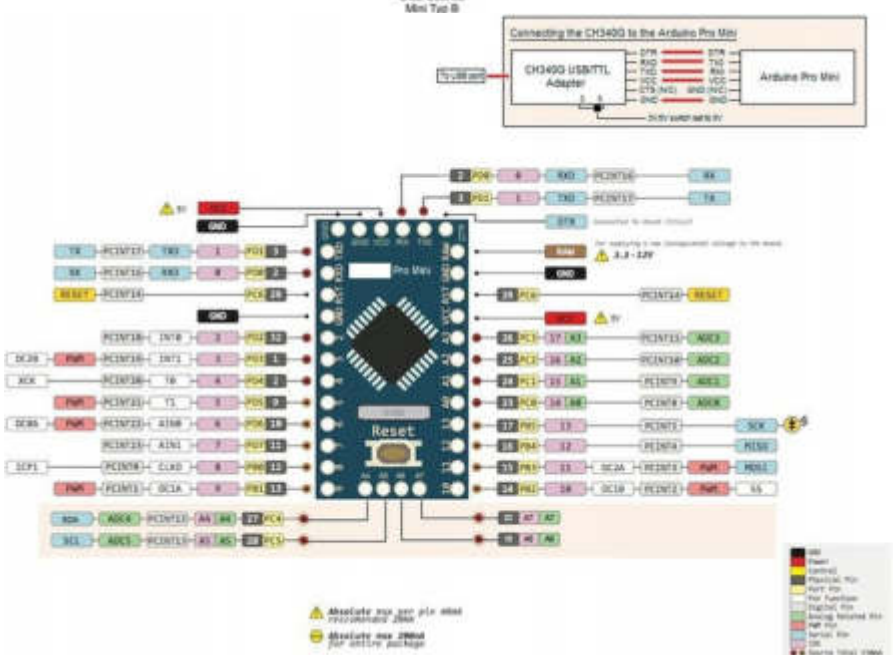
Широкое распространение получили «радиолюбительские» ФУОЗ на основе 8 разрядных микроконтроллеров с тактовой частотой 4-20 МГц. Например, семейства STM32, PIC или AVR. Несмотря ни низкую частоту работы МК, его мощностей хватает на вычисление и управление, однако процедуру постоянного периодического контроля состояний линий датчиков заменяют механизмом аппаратных прерываний. Следует отметить, что эта замена в силу наличия большого количеств импульсных помех является некорректной без дополнительной программной обработки полученных данных. Типичные признаки проявления прямого применения - пропуски в зажигании из-за ошибочно принятых за событие входного датчика импульсных помех. Примером дополнительной программной обработки могут служить «ослепление» линий датчиков (в моменты времени, когда изменения входных сигналов не должны происходить) или «повторное чтение» (после возникновения прерывания, программа производит дополнительный контроль линии на изменение). Второй способ считаю единственно верным или главным, если оба механизма используются в программе.

2. ФУОЗ НА ПЛАТФОРМЕ ARDUINO.

Представленный ниже ФУОЗ является примером описанного выше варианта на «медленном» МК. В отличии от других подобных устройств предлагается использовать готовые платы, на которых уже установлен МК и все необходимые для его работы элементы, включая и UART интерфейс – платформа Arduino совместимых плат с кварцевым резонатором 16 МГц (Рис. 2). Внутренний загрузчик можно при желании отключить.



Arduino Nano



Arduino Pro mini

Рис. 2 Платформа

Отмечу, что вариант Arduino Pro mini мне кажется предпочтительнее из-за отключаемого (внешнего) преобразователя UART-USB, ибо в 99 процентах рабочего времени ФУОЗ он не нужен. Кроме того, сам преобразователь UART-USB тоже является устройством на основе микроконтроллера, и как показала практика, в неблагоприятных условиях подвержен «зависанию» при больших импульсных помехах.

Все программное обеспечение ФУОЗ содержит три уровня. Уровень Ядро (подключаемый при компиляции файл **CORE.INC**) - обеспечивает основные функции: контроль входных линий, измерение временных интервалов, расчетный модуль, работу четырех каналов АЦП, сервисные функции и прочее. Уровень Интерфейс - программа для персонального компьютера (ПК) **UOZ.EXE** (далее UOZ) для операционной системы Windows вместе интерфейсным модулем Ядра обеспечивает взаимодействие с пользователем (настройка параметров, индикация текущего

состояние, протоколирование работы Ядра). **Оба этих уровня носят завершенный характер и при переходе от двигателя к двигателю в коррекции не нуждаются.**

Главный уровень (файл **MAIN.ASM**) или Моторная часть - обеспечивает смысловую связь Ядра с сигналами датчиков, формирователями импульсов зажигания. **Именно эту часть программного обеспечения и надлежит изменять в процессе адаптации к конкретному двигателю и системе датчиков.** В этом блоке задаются все константы Ядра и устанавливаются режимы работы его отдельных частей.

Одним из предлагаемых решений является квазивиртуализация сигналов входных датчиков. Пользователь фактически сам определяет каким внешним физическим событиям соответствуют основные этапы функционирования Ядра, воздействуя на него изменением переменных и регистров Ядра, вызовом в нужные моменты времени макросов StartDelta и StopDelta, которые является отражением состояния виртуального входного датчика. В свою очередь, Ядро вызывает в нужные моменты времени соответствующие подпрограммы Моторной части. Эта квазивиртуализация стала следствием применения алгоритма защиты входных физических линий от импульсных помех, заключающегося в процедуре дополнительного чтения входных линий после их изменения на входах микроконтроллера, способных вызывать прерывание. Все вышеизложенное дало возможность использовать построенную систему и для двух-трех цилиндровых двигателей (последовательный запуск Ядра между цилиндрами), но и с разными типами датчиков и формирователей их сигналов.

3. ОСНОВЫ ФУНКЦИОНИРОВАНИЯ

Пусть есть виртуальный импульс от датчика известного углового размера Δ (метка Delta) перед верхней мертвой точкой цилиндра (ВМТ) - Рис.3. Угловое расстояние между меткой Delta и ВМТ - β определяется конструкцией датчика. Метка Delta возникает один раз в заданный интервал вращения Base (угол-база), значение которого следует выбирать по Таблице 1.

Таблица 1

Цилиндры	1	2(1)	3(1)
Base	360	180	120

Примечание: Ядро применяется последовательно от цилиндру к цилиндру.

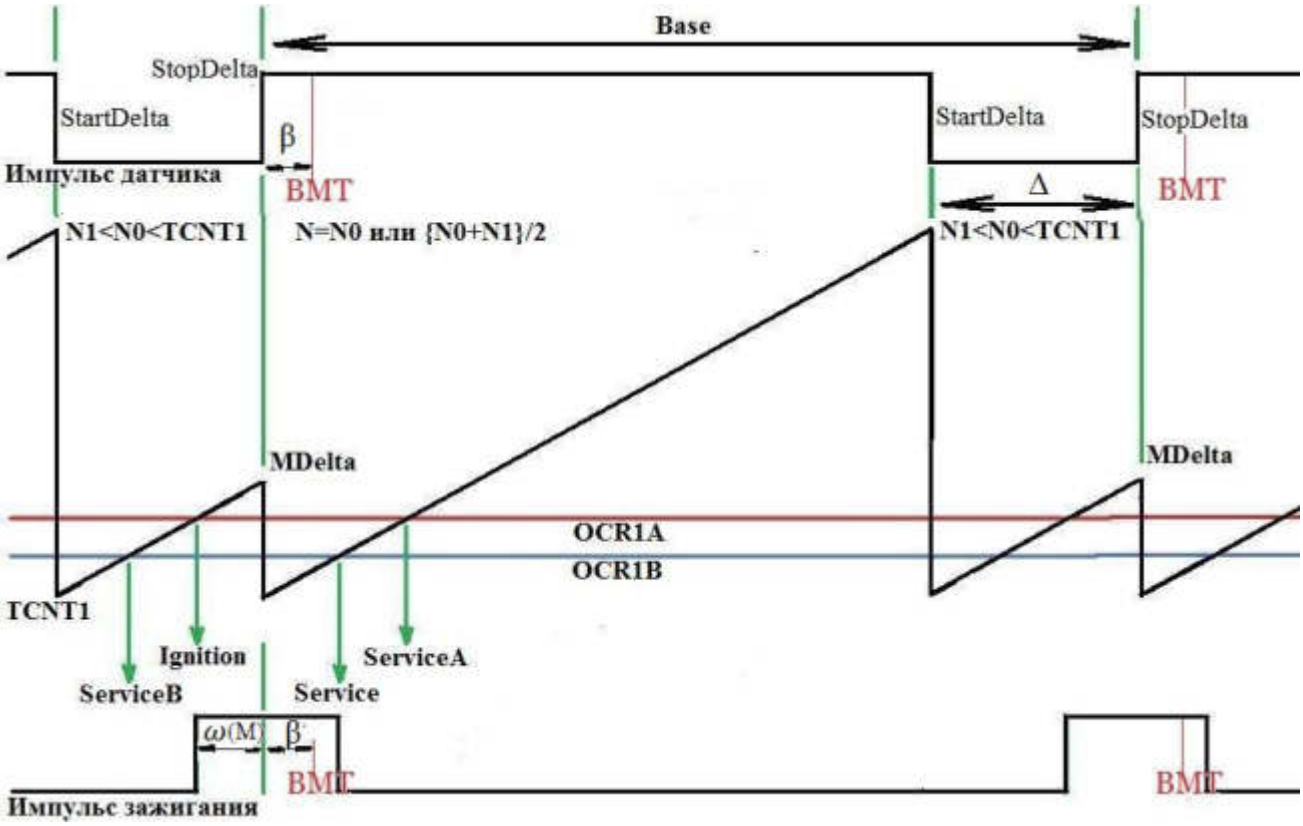


Рис. 3 Принцип работы системы

Центральным узлом Ядра является 16 разрядный счетчик T1 в нормальный режиме, на вход которого подключен управляемый тактовый генератор – 8 разрядный таймер T0 в режиме CTC. Количественное значение тактовой частоты ***F*** определяется соответствующей константой в Моторной части - ***Fn***. Формула для расчета тактовой частоты с кварцевым резонатором 16 МГц имеет вид:

$$F = \frac{16\ \text{МГц}}{2 \cdot (1 + Fn)} = \frac{8000000}{(1 + Fn)}, \text{Гц}$$

Для обеспечения работы этой связки таймеров необходимо соединить линии D5 и D6 (5 и 6) используемой платформы с МК (линии PD5 и PD6 МК) – Рис.4.

Значения счетчика таймера T1 накопленное от события StopDelta до StartDelta и являющееся оценкой периода вращения маховика в отсчетах тактовой частоты, поступает на сдвигающую цепочку →N0→N1. Данное решение обусловлено предоставлением возможности принятия за оценку скорости вращения маховика как значение счетчика таймера в последнем цикле, так и среднее значение двух последних циклов. Полученное значение используется для расчета времени задержки искрообразования. Значения счетчика таймера T1, накопленное от события StartDelta до StopDelta, является оценкой длины метки MDelta в отсчетах тактовой частоты на данной скорости вращения маховика. Значения MDelta используется для расчета времени отключения выходного сигнала.

Таймер T1 имеет два равнозначных канала сравнения с предустановленными значениями OCR1A и OCR1B. Канал OCR1A используется в Ядре для формирования временной задержки искрообразования в интервале между метками StartDelta до StopDelta. Основной математический аппарат, используемый в Ядре, направлен на расчет о программирование значения порога срабатывания OCR1A при заданных оборотах. С этим действием связаны две подпрограммы Моторной части. Во-первых, это подпрограмма Ignition, которая вызывается ядром при превышении значения таймера T1 кода в канале OCR1A. Это есть момент подачи искры в цилиндр. **Обращаю внимание**, что искра подается по зависимости ω(M), см. Рис.3, относительно не ВМТ, а конца метки Delta, которая смещена на угол β от неё, зависящий от конструкции двигателя и датчиков. Следовательно реальный угол опережения зажигания, не может быть меньше, чем β и не больше, чем Δ+β. Во-вторых, это подпрограмма UOZMinError вызывается при попытке установить слишком низкий уровень OCR1A. В силу того, что математический аппарат требует времени для проведения вычислений, то установить порог срабатывания OCR1A ниже, чем значение таймера, которое уже накопилось во время работы математического блока, невозможно. При наступлении этого события активируется флаг Status.UOZMin и искрообразования через обычный механизм не происходит. Пользователь сам решает надо ли формировать сигнал зажигания по этому событию.

Очевидно, что при малых оборотах счетчик T1 может переполняться. Ядро помечает это событие как режим Низких оборотов (LowMode). В этом случае искрообразование через механизм Ignition и UOZMinError не происходит и следует позаботиться пользователю в Моторной части об искрообразовании во время детектирования события StopDelta. В зависимости от величины тактовой частоты, поступающего на таймер T1 сигнала нижний предел оборотов, при которых включится режим Низких оборотов, должен быть менее или равен характерным холостым оборотам двигателя.

Второй канал OCR1B используется Ядром в качестве средства отключения включенного сигнала зажигания в интервале между метками StopDelta до StartDelta. Расчет и программирование значения OCR1B осуществляется после получения оценки MDelta. Для обеспечения этого в Ядро встроены подпрограммы SetService и SetEOCR1B. При условии превышения текущего значения таймера значения OCR1B Ядром вызывается программа **Service**, в котором и следует выключить сигнал управления зажигания цилиндра.

В случае чрезвычайно низких оборотов счетчик таймера T1 начнет переполняться и при оценке величины MDelta. Вплоть до этого момента, механизм отключения включенного сигнала зажигания функционирует за счет искусственного увеличения разрядности OCR1B до 24 бит (EOCR1B). Данное решение позволяет выдерживать корректные временные интервалы при отключении импульса зажигания при искрообразовании начиная с 50-200 оборотов в минуту, что очень важно для режима запуска. Дополнительно это позволяет получать оценку частоты вращения двигателя программой **UOZ** (кнопка «T» стр. 33)**.**

Подпрограммы Ignition и Service являются не отключаемыми. Дополнительно к ним существуют еще две отключаемые подпрограммы ServiceB и ServiceA. Подпрограмма ServiceB может вызываться ядром при равенстве величины счетчика с порогом OCR1B на интервале между метками StartDelta до StopDelta. Для использования этого функционала следует разрешить это действие при настройке Ядра и производить программирование порога после метки StartDelta. Аналогичная ситуация с функционалом ServiceA на интервале между метками StopDelta до StartDelta. Программировать порог OCR1A для этого следует программировать после метки StopDelta.

Еще одна функция, возложенная на таймер T1 это детектирование полной остановки двигателя (StopMode). Как было указано выше, в режиме Низких оборотов (LowMode) происходит переполнение таймера T1. Ядро проводит подсчет числа переполнения при условии отсутствия вызова событий StartDelta и/или StopDelta. Считаем, что двигатель остановлен, если этих событий не было заданной количество секунд.

Рассмотрим механизм обработки входных сигналов – рис. 4. Сигналы от двух входных датчиков поступают на входы микроконтроллера, способные вызывать асинхронные прерывания int0 и int1, вектора обработки прерываний которых указывают на одну и туже программу обработки прерывания. При возникновении любых изменений на входных линиях происходит чтение всех линий и размещение их логического состояния в соответствующих сдвиговых регистрах. Так начинается выполнения протокола защиты от пульсаций, причем на время его выполнения прерывания int0 и int1 игнорируются. Одновременно с этим активизируется таймер T2, настроенный на вызов программного прерывания с периодом от нескольких микросекунд (ProtectClk). С этим периодом (интервалом дискретизации) продолжается чтение состояния входных линий и запись их состояния в сдвиговые регистры DH1 и DH2. Процесс останавливается при повторении на всех каналах одного (своего) уровня в течении ProtectCount тактов таймера T2. На рис.4 и рис.5 представлен случай ProtectCount=2. На этом процесс выполнения протокола защиты от пульсаций завершается. Результирующие логические уровни размещаются в младшие биты нибблов сдвигового регистра Событий (Events). Таким образом, содержание этого регистра изменяется в результате изменения любого входного сигнала и отражает их состояние в одни и те же моменты времени (моменты изменения любого из сигналов), причем текущем и трех предыдущих. Данное решение, вместе подпрограммой Моторной части **Actions,** является частью квазивиртуализации сигналов входных датчиков. Подпрограмма Action должна строить реакцию программной части на регистр Событий и выполнять макросы StartDelta и StopDelta при двух разных значения Events.

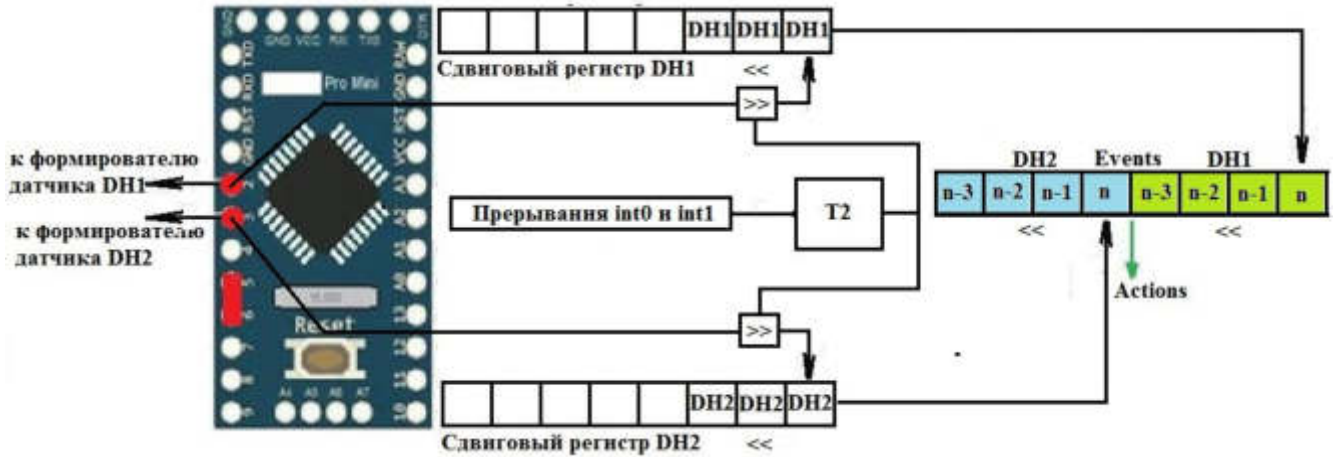


Рис. 4 Механизм обработки входных сигналов

Следует отметить следующие моменты. Во-первых, канал датчика DH2 является отключаемым. В случае его отключения изменения на линии D3(3) используемой платформы с МК не приводят к возникновению прерывания int1 и сдвиговая цепочка DH2 заполняется логическими нулями. Во-вторых, выполнение протокола защиты от пульсаций «отодвигает» момент реакции всего программного комплекса на реальные изменения входных линий. В лучшем случае этот процесс будет длиться ProtectCount * ProtectClk микросекунд плюс характерные временные интервалы вызова и обработки прерываний микроконтроллера - порядка 6 мкс. (Рис.5) Суммарные размеры этих временных потерь от 10 до 100 мкс. **Эти временные потери компенсируются при работе математического аппарата Ядра** аддитивным слагаемым со знаком минус при расчете значения OCR1A.

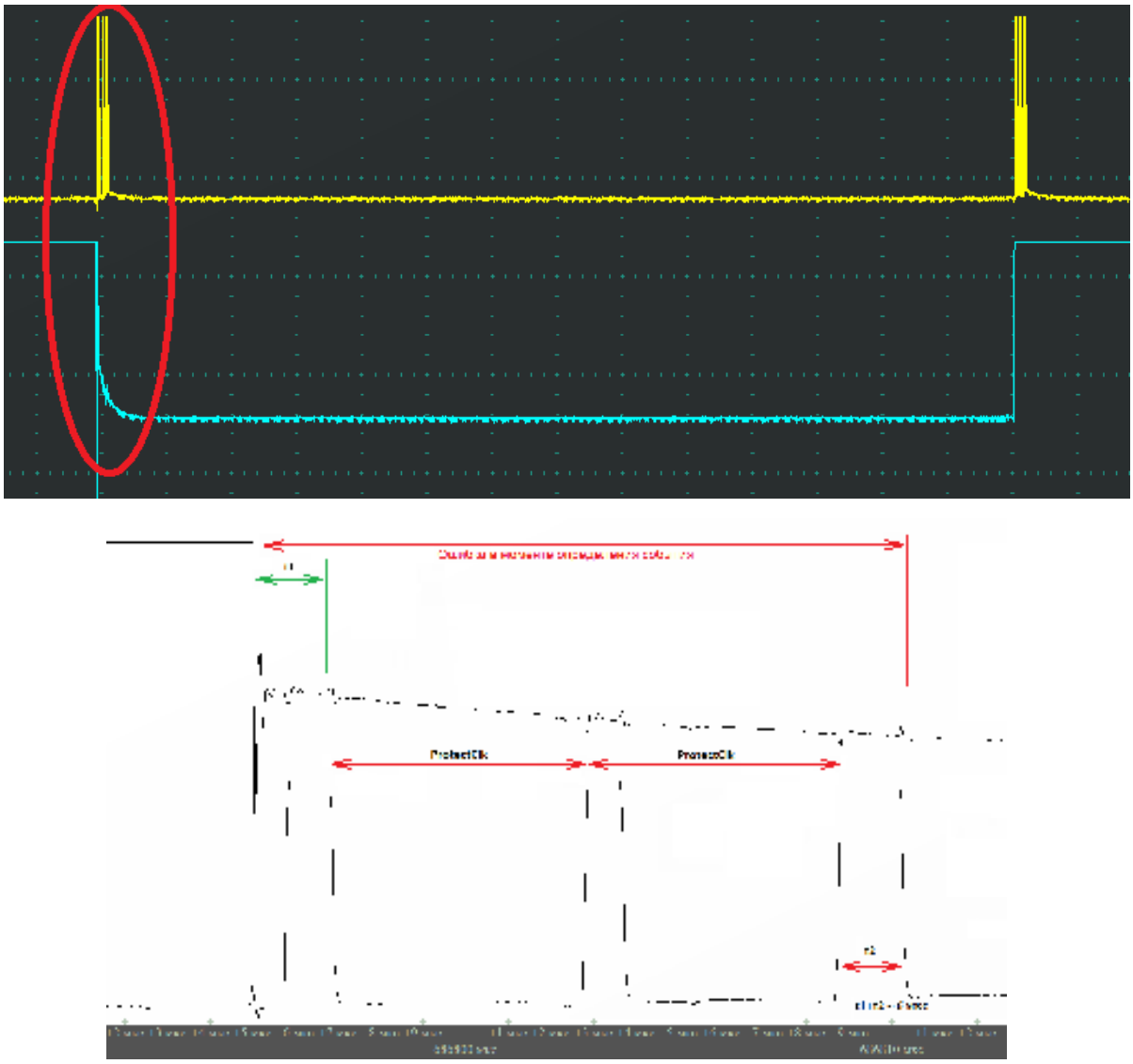


Рис. 5 Иллюстрация временных потерь при повторном чтении входной линии

На рис.6 приведена иллюстрация квазивиртуализации. Приведены три характерных примера. На рис. 6а представлен самый простой и очевидный случай - одноцилиндровый двигатель с одним оптическим датчиком или датчиком Холла (Base=360). Кроме того, ниже приведен пример подпрограммы Actions для этог случая:

Actions:

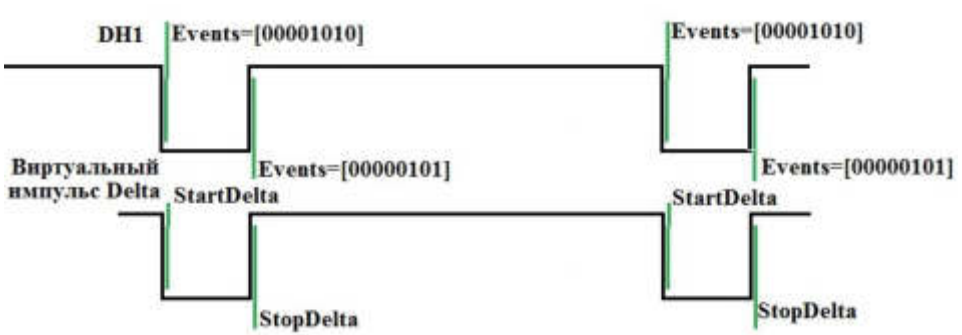
C_Ign:	cpi	Events, 0b00001010	-	Сравнение регистра Событий с признаком начала метки на DH1
	brne	C_VMT	-	Если не равны, то проверка следующего кода
	StartDelta		-	Макрос отметки времени (начало виртуального импульса Delta)
	…..			
exit:	ret		-	Выход из подпрограммы
C_VMT:	cpi	Events, 0b00000101	-	Сравнение регистра Событий с признаком окончания метки на DH1
	brne	exit	-	Если не равны, то выход из подпрограммы
	StopDelta		-	Макрос отметки времени (окончание виртуального импульса Delta)
	…..			
	ret		-	Выход из подпрограммы

На рис. 6 б) представлен другой пример - одноцилиндровый двигатель с индуктивным датчиком (Base=360). Предполагается, что положительная полуволна сигнала индуктивного датчика через формирователь поступает на канал DH1, отрицательная полуволна – на вход DH2. Характерных событий уже не 2, а четыре: начало положительного импульса, завершение положительного импульса, начало отрицательного импульса и момент его завершения. Красной меткой отмечены значения регистра Событий, соотвествующих началу и завершению виртуального импульса Delta. Внешний вид подпрограммы Action будет аналогичниым предыдущему случаю с учетом иных характерных значений кодов сравнения - вместо пары 00001010/00000101 будет пара 10111110/11101011.

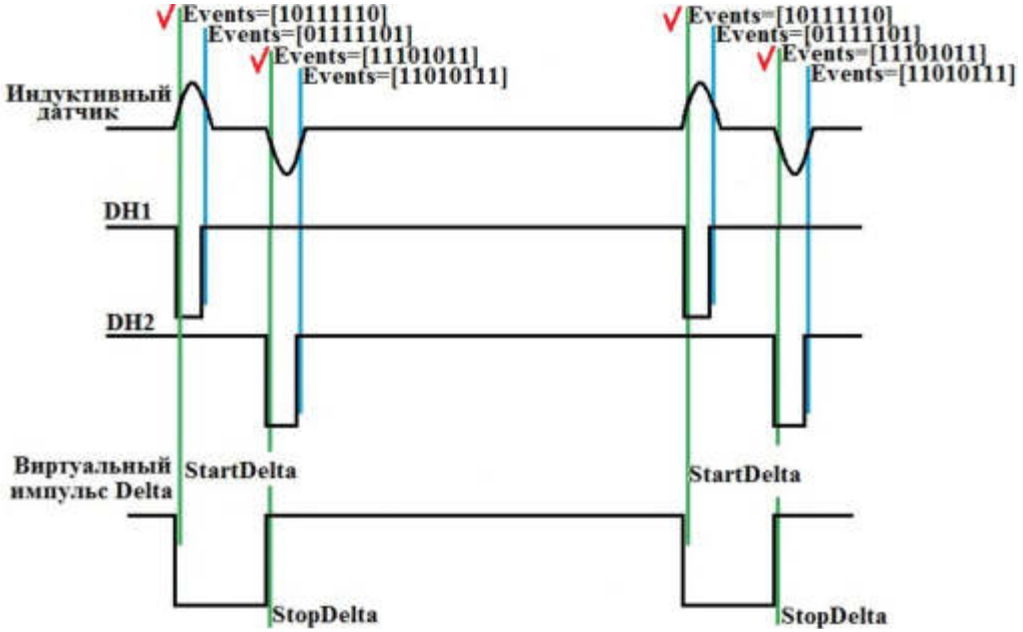
Третий пример более сложный - двухцилиндровый двигатель Ветерок с двумя униполярными датчиками Холла с реакцией от двух магнитных «башмаков» маховика (Base=180). Два других «башмака» маховика имеют другое направления магнитного поля. Виртуальный импульс Delta локализуется при переходе от цилиндра к цилиндру. Искрообразование раздельное благодаря применению дополнительного свободного флага Пользователя, которому придадим статус флага первого цилиндра (Расширенный регистр состояния EStatus, таблица 3 Продолжение).

Actions:

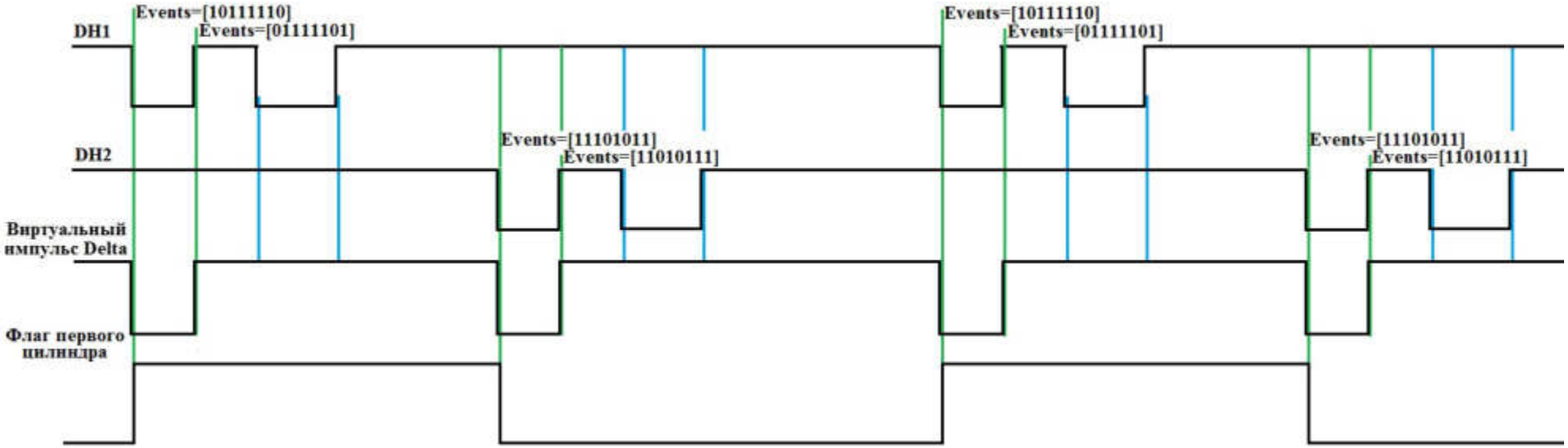
```
C_Ign1: cpi     Events, 0b10111110   -  Код начала метки Delta на первом цилиндре
        brne    C_Ign2               -  Если не равны, то проверка следующего кода
        StartDelta                   -  Макрос отметки времени (начало виртуального
                                        импульса Delta)
        sbr     EStatus,Ign1         -  Включить флаг первого цилиндра
        …..
        ret                          -  Выход из подпрограммы
C_Ign2: cpi     Events, 0b11101011   -  Код начала метки Delta на втором цилиндре
        brne    C_VMT1               -  Если не равны, то проверка следующего кода
        StartDelta                   -  Макрос отметки времени (начало виртуального
                                        импульса Delta)
        cbr     EStatus,Ign1         -  Выключить флаг первого цилиндра
        …..
        ret                          -  Выход из подпрограммы
C_VMT1: cpi     Events, 0b01111101   -  Код завершения метки Delta на первом цилиндре
        brne    C_VMT2               -  Если не равны, то проверка следующего кода
        StopDelta                    -  Макрос отметки времени (окончание виртуального
                                        импульса Delta)
        sbr     EStatus,Ign1         -  Включить флаг первого цилиндра
        …..
        ret                          -  Выход из подпрограммы
C_VMT2: cpi     Events, 0b11010111   -  Код завершения метки Delta на втором цилиндре
        brne    exit                 -  Если не равны, то выход
        StopDelta                    -  Макрос отметки времени (окончание виртуального
                                        импульса Delta)
        cbr     EStatus,Ign1         -  Выключить флаг первого цилиндра
        …..
   exit: ret                         -  Выход из подпрограммы
```



а)



б)



в)

Рис. 6 Изменение регистра Событий (Events) во времени

4. МАТЕМАТИЧЕСКИЙ АППАРАТ

При таком определения интервалов Δ и Base как на рис. 3, расчет временной задержки, для записи в канал OCR1A производится по формуле

$$\text{OCR1A} = N \cdot \frac{\Delta - \omega(M,\dots)}{Base - \Delta} = N \cdot \Omega(M), \quad \text{где } \Omega(M) = \frac{\Delta - \omega(M,\dots)}{Base - \Delta}$$

где M – частота вращения маховика, а N – оценка периода вращения маховика в отсчетах тактовой частоты таймера T1, $\omega(M,\dots)$ – УОЗ относительно конца метки Delta и является функцией и текущих оборотов и прочих параметров.

Зависимость $\Omega(M)$ при разумном выборе параметров Base, Δ и $\omega(M,\dots)$ не превосходит единицы. Данный фактор позволил для вычисления использовать аппаратные возможности целочисленной арифметики микроконтроллера – так называемый формат Q1.7 (**в Ядре он указан как - FP8**). Старший бит - целая часть числа (или 0 или 1), семь младших разрядов – дробная часть числа. Аппаратный умножитель микроконтроллера позволяет получить число в формате Q1.15 (**в Ядре - FP16**). К сожалению, рассматриваемый контроллер не имеет аппаратного делителя и эту функцию приходиться реализовывать программно – «столбиком».

Верхним индексом будем обозначать разрядность формата с фиксированной точкой, все умножения и деления теперь целочисленные, разрядность результата корректируется с учетом перевода чисел в форматы Q1.7/Q1.15. Следует отметить, что $\Omega(M)$ иногда даже много меньше единицы. Это приводит к потере значимости разрядов при применении целочисленной арифметики конечной разрядности. Чтобы избежать этого, применен масштабирующий **четный** (далее будет обоснован выбор четных значений) коэффициент Ka, на который значения $\Omega(M)$ сначала умножаются, а потом общий результат делиться на него.

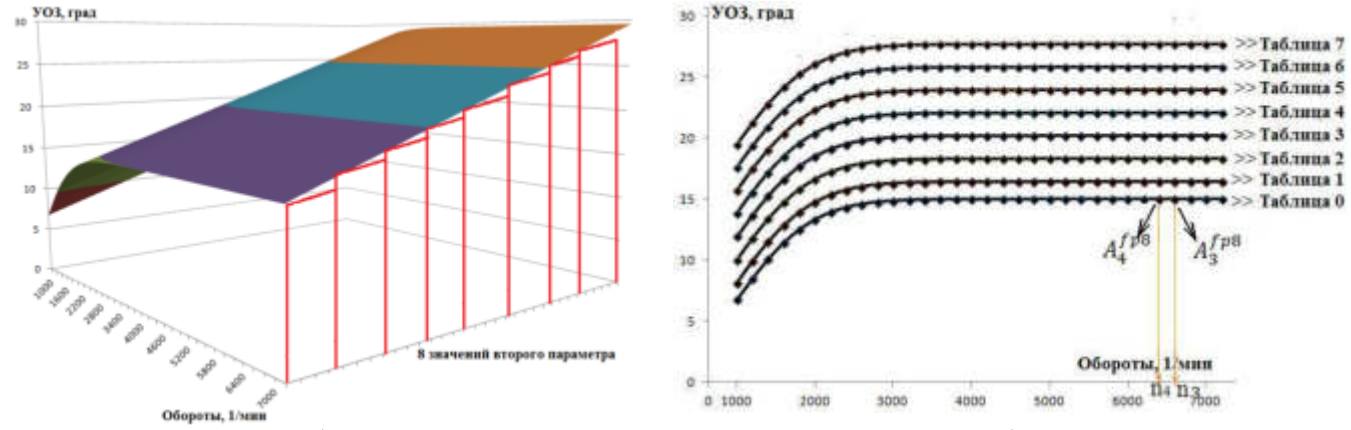
$$\text{OCR1A} = N \cdot \frac{A^{fp8}(M)}{Ka \cdot 128}, \qquad A^{fp8}(M) = \Omega(M) \cdot Ka \cdot 128$$

Обращаю внимание, что **меньшему значению УОЗ соответствует большее значение** $A^{fp8}(M)$. Величину Ka можно оценить по формуле

$$Ka = 2 \cdot int\left[\frac{Base - \Delta}{\Delta}\right] \tag{1}$$

где int - операция выделения из вещественного числа целой его части в формате Q8.0 (обычное восьмиразрядное целое).

В силу того, что аналитической зависимости $\omega(M,\dots)$ в общем случае нет, то используется стандартный подход - интерполяция функции, заданной таблично. В этой связи множитель $A^{fp8}(M)$ рассчитывается заранее и сводится в совокупность таблиц – разные наборы зависимостей УОЗ от оборотов с выбором таблицы по величине еще одного внешнего фактора. На рис. 7 представлена иллюстрация этого перехода от аналитической зависимости $\omega(M,\dots)$ к таблицам A^{fp8}.



а) б)

Рис. 7 Переход от аналитической зависимости α(M…) к таблицам A^{fp8}

Таким образом, в устройстве реализована линейная интерполяция трехмерной поверхности УОЗ по оси оборотов и выбор табличных значений по оси второго параметра, в качестве которого может выступать один из внешних факторов: угол поворота дроссельной заслонки, температура окружающей среды, напряжение датчика абсолютного давления, просто потенциометр на панели или переключатель между 2 таблицами путем коммутации входа на массу или к +5В.

Общее число таких таблиц определяется константой TblCount и может быть до 32 шт. В каждый момент времени Ядро работает с одной таблицей (строка выделена желтым цветом в Таблице №2), номер которой задан в переменной TblSelectr. После получения значения счетчика таймера T1 (N) накопленное от метки StopDelta до StartDelta ядро определяет диапазон оборотов – номер ячейки в текущей таблице (Selectr). Если обороты выше заданных в таблице (случай $n_0 > N$), то дальнейший расчет производится по значению A_0^{fp8}. Если обороты ниже заданных в таблице (переполнение счетчика T_1), расчет не производиться и зажигание должно происходит по событию конца метки Delta. Итак, пусть $n_4 > N \geq n_3$. Тогда Selectr=3 - отмечено зеленый цветом в Таблице №2. Тогда в расчетной форме будут использованы пары значений $[n_4, A_4^{fp8}]$ и $[n_3, A_3^{fp8}]$ – Рис. 6 б).

Таблица 2.

	Selectr	31	30	29	28	…	6	5	4	3	2	1	0
	Границы диапазонов	n_{31}	n_{30}	n_{29}	n_{28}	…	n_6	n_5	n_4	n_3	n_2	n_1	n_0
TblSelectr	Таблица 0	A_{31}^{fp8}	A_{30}^{fp8}	A_{29}^{fp8}	A_{28}^{fp8}	…	A_6^{fp8}	A_5^{fp8}	A_4^{fp8}	A_3^{fp8}	A_2^{fp8}	A_1^{fp8}	A_0^{fp8}
…	…	…	…	…	…	…	…	…	…	…	…	…	…
	Таблица 4	A_{31}^{fp8}	A_{30}^{fp8}	A_{29}^{fp8}	A_{28}^{fp8}	…	A_6^{fp8}	A_5^{fp8}	A_4^{fp8}	A_3^{fp8}	A_2^{fp8}	A_1^{fp8}	A_0^{fp8}

Для границ диапазонов действуют следующие условия: $n_{31} \equiv 65535$, $n_{31} > n_{30} > n_{29} \ldots > n_2 > n_1 > n_0$. Связь с реальными оборотами M определяется формулой:

$$n_q = round\left[\frac{Base - \Delta}{6 \cdot M_q} \cdot F\right], \quad q = 0..31 \tag{2}$$

где $round$ - операция округления вещественного числа до обычного шестнадцатиразрядного целого (в частности для n_q - формат Q16.0), M_q – обороты маховика в 1/мин, F – частота тактовых импульсов таймера ***T1***. В дальнейшем принимаем round(x)=int(x+0,5).

Общая форма расчета временной задержки для записи в канал OCR1A (формат Q16.0) с линейной интерполяцией, адаптированная для применения целочисленной математики микроконтроллера с фиксированной точкой Q1.7/Q1.15 выглядит следующим образом:

$$\mathrm{OCR1A} = \frac{N \cdot \left\{B_k^{fp8} \cdot A_{k+1}^{fp8} + \left(1^{fp8} - B_k^{fp8}\right) \cdot A_k^{fp8}\right\}}{\mathrm{Ka} \cdot 128 \cdot 128}, \qquad B_k^{fp8} = int\left[\frac{N - n_k}{n_{k+1} - n_k} \cdot 128\right]$$

$$1^{fp8} = 128, \qquad \mathrm{k} = Selectr.$$

Инструкция аппаратного умножения микроконтроллера *fmul(B,A)* производит автоматическую коррекцию разрядности результата при умножении двух числе в формате Q1.7 в формат Q1.15 – сдвигает результат влево на один разряд (умножает на 2 результат). В этой связи последнее выражение в фигурных скобках принимает вид:

$$B_k^{fp8} \cdot A_{k+1}^{fp8} + \left(1^{fp8} - B_k^{fp8}\right) \cdot A_k^{fp8} = \frac{fmul\left\{B_k^{fp8} \cdot A_{k+1}^{fp8}\right\} + fmul\left\{1^{fp8} - B_k^{fp8},\ A_k^{fp8}\right\}}{2}$$

Номер активной таблицы $TblSelectr$ (формат Q8.0) выбирается Ядром в зависимости от уровня аналогового напряжения на одном из четырех входов микроконтроллера ADC_i (A4..A7, i=4..7):

$$TblSelectr = int\left[ADC_i \cdot \frac{TblCount}{256}\right]$$

Любой другой из вышеуказанных четырех каналов АЦП может быть использован для работы ручного корректора – в диапазоне [- Апертура… +Апертура]:

$$C^{fp16} = \frac{\text{Апертура} \cdot Ka \cdot 32768}{Base - \Delta} \cdot \left\{2 \cdot \frac{ADC_j}{256} - 1\right\}$$

где j – номер канала АЦП ручного корректора УОЗ.

Последним слагаемым в формуле для расчета временной задержки для записи в канал OCR1A является слагаемое, учитывающее затраты времени на выполнение протокола защиты от пульсаций ΔT в формате Q16.0. Таким образом, итоговая формула будет иметь вид:

$$\text{OCR1A} = \frac{N \cdot \left[fmul\left\{B_k^{fp8} \cdot A_{k+1}^{fp8}\right\} + fmul\left\{128 - B_k^{fp8},\ A_k^{fp8}\right\} + C^{fp16}\right]}{\text{Ka} \cdot 32768} - \ \Delta\text{T}$$

$$A_k^{fp8} = int\left[\frac{\Delta - \alpha(M, \dots)}{Base - \Delta} \cdot Ka \cdot 128\right],\ B_k^{fp8} = int\left[\frac{N - n_k}{n_{k+1} - n_k} \cdot 128\right],\qquad k = Selectr,$$

$$C^{fp16} = -round\left[\frac{\text{Апертура} \cdot Ka \cdot 32768}{Base - \Delta}\right] + 2 \cdot round\left[\frac{\text{Апертура} \cdot Ka \cdot 32768}{Base - \Delta} \cdot \frac{ADC_j}{256}\right] \qquad (3)$$

$$\Delta T = round\left[(ProtectCount \cdot\ ProtectClk + 6)\frac{8}{1 + Fn}\right]$$

$$TblSelectr = \ int\left[ADC_i \cdot \frac{TblCount}{256}\right]$$

i – номер канала АЦП для выбор таблицы, j – номер канала АЦП ручного корректора УОЗ

Обращаю внимание, что обилие множителей и делителей на 128 и 32768 это взаимный переход между форматом целых чисел и форматом Q1.7/Q1.15. Как правило, все умножения на эти числа делаются заранее при настройке параметров, а вот деление напрямую не делается вовсе в силу того, что оно суть перераспределение разрядов в итоговом результате. Например, на этапе общей подготовки введем константу

$$Correct^{fp16} = round\left[\frac{\text{Апертура} \cdot Ka \cdot 32768}{Base - \Delta}\right] \qquad (4)$$

Тогда целочисленное выражение для аддитивного слагаемого ручного корректора будет таковым:

$$C^{fp16} = -Correct^{fp16} + fmul\left\{ADC_j,\ High(Correct^{fp16})\right\} + \\ +High\left[fmul\left\{ADC_j,\ Low(Correct^{fp16})\right\}\right] + 256 \cdot \text{Carry}|_{fmul\{ADC_j,\ Low(Correct^{fp16})\}} \qquad (5)$$

где $High$ и Low - операция выделения старшей и младшей половины шестнадцатиразрядного числа соответственно. Здесь используется нестандартный способ применения команды *fmul* для автоматического умножения результата на 2. Учитывая особенность команды *fmul,* к результату третьего слагаемого следует добавить 256, если при его вычислении возникло переполнение. Во втором слагаемом такое переполнение невозможно в силу способа задания константы.

Для $TblSelectr$ выражение выглядит так:

$$TblSelectr = \ High(ADC_i \cdot TblCount) \qquad (6)$$

Сложнее с B_k^{fp8}. Шестнадцатиразрядную разность $N - n_k$ дополняем справа 1 нулевым байтом (эквивалент умножения на 256) и делим программно это 24 битное слово на шестнадцатиразрядную разность $n_{k+1} - n_k$. Исходя из способа определения исходных разностей, целая часть результата будет меньше единицы и будет находиться во втором байте результата. Дробная часть - в младшем байте результата. Старший байт результата тождественно равен 0. Теперь если эту не нулевую пару регистров сдвинуть вправо на 1 разряд, то получиться искомое выражение для B_k^{fp8}. Отмечу, что можно сделать и так, что разность $n_{k+1} - n_k$ будет являться степенью 2, тогда можно сократить объем операций путем замены деления на сдвиг да еще и с учетом множителя 128. Пусть $n_{k+1} - n_k = 256\ (512, 1024, 2048 \dots .)$ тогда:

$$B_k^{fp8} = int\left[\frac{N - n_k}{n_{k+1} - n_k} \cdot 128\right] = \text{Сдвиг вправо } [N - n_k] \text{ на } 2\ (3, 4, 5 \dots) \text{ разряда.}$$

Последний нюанс - нормализация результата вычисления OCR1A в форме (3). Выражение $N \cdot \left\{ fmul\ [B_k^{fp8},\ A_{k+1}^{fp8}] + fmul[1^{fp8} - B_k^{fp8},\ A_k^{fp8}] + C^{fp16}\right\}$ имеет 32 разрядный результат в формате Q17.15. В этом результате точка, разделяющая целую и дробную часть, расположена следующим образом:

Деление на 32768 в формуле (3) означает, что результат необходимо сдвинуть вправо 15 разрядов. Альтернативно можно весь результат сдвинуть влево на 1 разряд и взять старшие два байта. И в том и в другом случае деление на 32768 было бы выполнено. Однако, в формуле (2) есть еще один делитель - Ka. Изначально ставилось условие, что он четный. Это означает, что можно при дальнейшем делении на него по формуле (2) предварительно его «ополовинить» и «откинуть» младшие 16 бит результата многоразрядного умножения. Таким образом, после получения многоразрядного результата умножения достаточно использовать только старшие 16 бит и программно поделить их содержимое на константу Ka/2.

В случае, если константа Ka является степенью 2, то длинную подпрограмму деления 16 битного результата на 8 разрядную константу можно заменить сдвигом вправо на нужное количество бит. В Ядре предусмотрен этом механизм, что сокращает время выполнения вычислений. Оставшаяся не выполненная операция шестнадцатиразрядного суммирования результата с $-\Delta T$ трудностей не представляет.

Как было указано на стр. 4 функция отключения сигнала зажигания выполняет подпрограмма Service, которая вызывается по факту превышения значения счетчика T1 значения канала OCR1B с учетом механизма искусственного увеличения его разрядности до 24 (EOCR1B). Подпрограмма для задания этого порога срабатывания SetService встроена в макрос StopDelta. Следует лишь задать численное значение угла $\alpha \neq 0$ (рис.8) поворота маховика после завершения метки Delta, при котором сигнал зажигание следует отключить.

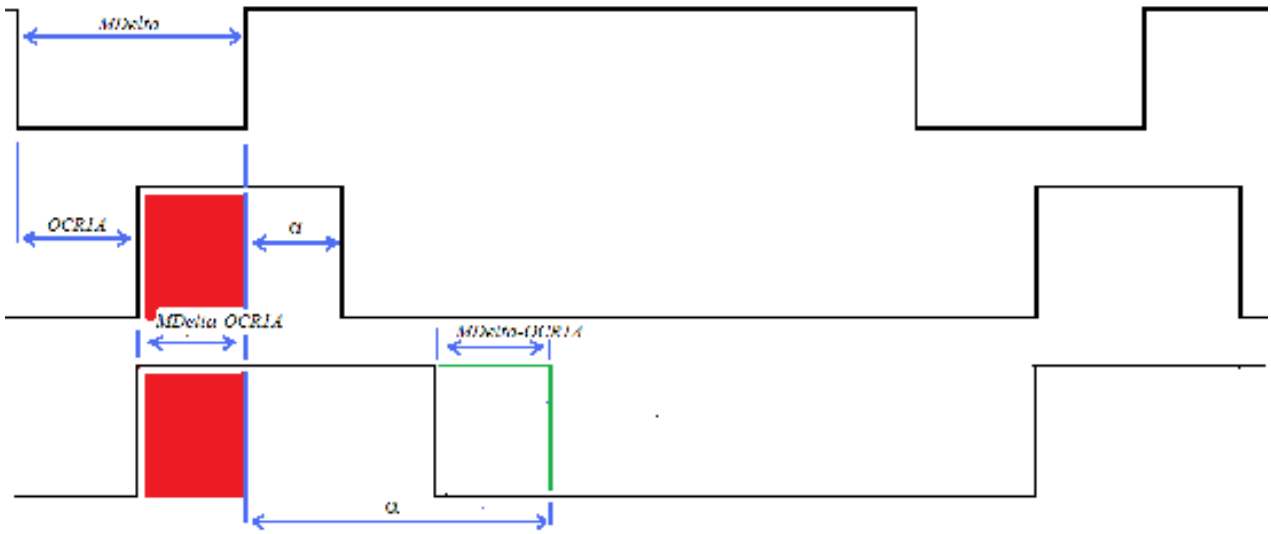


Рис. 8 Сигналы управления зажиганием и выходные каскады

Ядром поддерживается два режима работы: без учета времени от фактического момента появления сигнала зажигания до конца метки Delta (второй сверху график на рис.8) и с его компенсацией (нижний график на рис.8). На рис. 8 красным полем выделена зона потенциального момента искрообразования. В первом режиме формула расчета EOCR1B имеет вид:

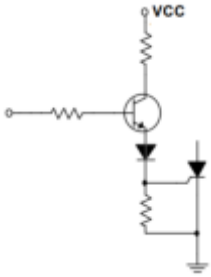
$$\mathrm{EOCR1B} = \frac{1}{256} \cdot \text{Delta} \cdot round\left[\frac{\alpha}{\Delta} \cdot 256\right] = High(\text{Delta}) \cdot A + High\{Low(\text{Delta}) \cdot A\},$$
$$A = round\left[\frac{\alpha}{\Delta} \cdot 256\right], \quad 0 < \alpha \leq Base - \Delta \qquad (7)$$

Этот режим предназначен для работы в системах CDI зажигания с выходным каскадом, представленном на рис. 9а, для обеспечения исчезновения тока на управляющем электроде тиристора до момента, когда на аноде тиристора может появиться первый импульс заряда от магнето.

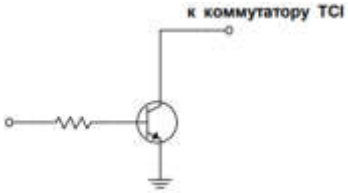
Во втором режиме формула расчета EOCR1B имеет вид:

$$\mathrm{EOCR1B} = High(\text{Delta}) \cdot A + High\{Low(\text{Delta}) \cdot A\} - \text{Delta} + \mathrm{OCR1A} \qquad (8)$$

Не трудно видеть, что в этом режиме длина импульса в угловом выражении, в отличии от предыдущего случая, остается константой. Этот режим необходим для работы в системах зажигания с транзисторным коммутатором типа XX.37.34, при использовании которых скважность сигнала управления равна 3. Выходной каскад ФУОЗ может быть выполнен в виде, как на рис. 9б).



а) CDI



б) XX.37.34

Рис. 9 Выходные каскады

Для работы вышеупомянутой серией коммутаторов необходимо включить соответствующую опцию Ядра в порту управления CoreSetup и задать значение угла α из таблицы:

Двигатель	Угол, градусы	
	Base	α
Одноцилиндровый	360	120
Двухцилиндровый с одним датчиком (искра подается одновременно в два цилиндра)	180	60
Двухцилиндровый с двумя датчиками (искра подается раздельно по цилиндрам)	180	120

5. ИСПОЛЬЗОВАНИЕ ПАМЯТИ МИКРОКОНТРОЛЛЕРА

Границы диапазонов n_k определяются в Моторной части. При запуске Ядра эта таблица каждый раз заноситься **в оперативную память** контроллера (ОЗУ), которая очищается при выключении устройства. Набор таблиц A_k^{fp8} хранятся **в энергонезависимой памяти микроконтроллера**. Данные, записанные в эту память, не будут сбрасываться даже при отключении питания. При запуске Ядра этот набор таблиц также загружается **в оперативную память** для работы Ядра в обычном режиме. В дальнейшем пользователь при подключении программы управления **UOZ** может выполнять следующие действия с таблицами:

1. Сброс в исходной состояние одной или всех таблиц в оперативной памяти из внесенной в программу **базовой таблицы.**
2. Сохранять на диск персонального компьютера и загружать с него в оперативную память контролера весь набор таблиц.
3. Сохранять в энергонезависимой памяти как одну активную, так и все таблицы из оперативной памяти для дальнейшего использования в рабочем цикле при следующем включении устройства.
4. Загружать из энергонезависимой памяти как одну активную, так и все таблицы.
5. Редактировать активную таблицу в визуальном режиме на экране компьютера даже во время работы двигателя.

Следует отметить, что в силу аппаратных ограничений микроконтроллера, внешние прерывания очень не желательны во время записи в энергонезависимую память, поэтому в Ядре она запрещена при работающем двигателе. Для сохранения изменений, внесенных в таблицы во время работы системы, следует после остановки двигателя **принять соответствующие меры по выполнению п.3 вышеуказанного перечня**.

6. АЦП

Вернемся к упомянутому выше АЦП. Как было указано, любые два канала мультиплексора АЦП из четырех старших каналов {выводы контроллера - PC4(adc4/scl), PC5(adc5/sda), adc6, adc7}могут быть использованы как для выбора активной таблицы, так и для функционирования ручного корректора УОЗ. Два других канала могут быть использованы по усмотрению пользователя. Например, измерение уровня напряжений. Каждый из каналов включается Пользователем отдельно в Моторной части. АЦП работает с максимальным временем преобразования (~ 104мкс) в восьмиразрядном режиме, причем используются старшие 8 из 10 разрядов. Такое решение увеличивает помехозащищенность в силу более крупного шага квантования уровня ~ 19,5 мВ, а не 4,9 мВ в случае использования 8 младших разрядов.

Для работы каналов АЦП в Ядре предусмотрен кольцевой буфер для накопления и усреднения его отсчетов. Длина усреднения определяется константой ADCBufferSize. АЦП работает в двух режимах. Во-первых, режим останова двигателя. В этом случае Ядро последовательно пере-

ключает включенные Пользователем каналы мультиплексора АЦП, запускает процесс преобразования и по флагу готовности размещает новый отсчет в очередной ячейки буфера соответствующего канала.

Во-вторых, режим низких и нормальных оборотов. В этом режиме все операции АЦП производятся Ядром при вызове макросов StartDelta и StopDelta. При выполнении макроса StartDelta происходит чтение и размещение отсчета АЦП в буфер выбранного канала и переключение мультиплексора на следующий канал, номер которого определяется в макросе StopDelta. Там же происходит запуск нового преобразования АЦП. На каждом цикле основной программы Ядра происходит усреднение включенных Пользователем каналов и размещение средних значений в оперативной памяти микроконтроллера. В случае, если канал отключен, в то соответствующей ячейки будет максимальное значение. Получить среднее значение нужного канала можно с помощью макроса GetADC (Таблица 4).

7. ГЛАВНАЯ РАЗМЕТОЧНАЯ ТАБЛИЦА

В силу того, что все параметры тесно взаимосвязаны, их совместный «подбор» осуществлен в электронной таблице в формате Open Office - «**Главная разметочная таблица.ods**» (Рис. 10)**.**

Поля, отмеченные **желтым цветом** являются полями для ввода информации. Зеленные поля или желтые поля, название которых выделены зеленым – для копирования в Моторную часть прошивки. Голубые поля и поля белым фоном – носят информационный характер и обычно являются промежуточным результатом для формирования зеленых полей.

Заполнение таблицы начинается с указания углового размера метки Delta (C43) и её смещения относительно ВМТ (C40). Далее следует задать угол-базу Base (C42). В полях «Примечание» (раздел констант) для некоторых параметров предусмотрен контроль параметров. В случае отсутствия контроля или **соответствия значения физической/математической модели появляется надпись «Ok».** Далее следует задать значения таблицы УОЗ (N5-N36) в зависимости от скорости вращения маховика (A5-A36). Это самый важный момент при заполнении Главной разметочной таблицы.

Во-первых, следует варьируя параметром Fn (C43) при заданных ранее Δ и Base, следует добиться необходимого значения нижнего передела скорости вращения маховика - A5. Далее необходимо задать верхний предел контролируемого диапазона оборотов вращения - A36. Пусть это будет 6200. Далее следует заполнить 30 значений оборотов (A6-A35). Основные моменты, о которых надо помнить при заполнении значений. Во-первых, не надо забывать, что между узловыми точками осуществляется линейная интерполяция, это означает, что если есть участок исходной функции УОЗ(Обороты) с линейной зависимостью, то его можно обозначить крайними значениями. Напротив, где есть нелинейные участки можно поставить больше точек.

Во-вторых, подбор кода частоты Fn тоже суть компромисс двух факторов. С одной стороны хочется уменьшать частоту тактирования таймера, что бы получить как можно меньшее значение минимальных оборотов в таблице, с другой стороны с уменьшением частоты ухудшается разрешающая способность алгоритма в целом. Это заключается в несоответствии расстояния между значениями измерительного таймера конкретному значению оборотов двигателя, что приведет к появлению ступенчатого шага изменения ОУЗ от оборотов на высоких их значениях. Контроль этой ситуации также возложен на столбец «Примечание» (раздел оборотов - E). Как правило, на высоких оборотах уже нет потребности в резком изменении УОЗ**, и считаю допустимым, если в итоге колонка E будет показывать «Ступеньки» на паре верхних значений диапазонов частот вращения двигателя**. Однако следует не допускать появление этого сообщения на средних и малых оборотах, которое может появиться, если задать очень крупный шаг по оборотам. Впоследствии, значения ячеек столбца A автоматически пересчитывается в таблицу «Код таймера» n_q по формуле 2 и далее формируется столбец M для дальнейшего переноса в прошивку.

При заполнении таблицы УОЗ (N5-N36) обязательно уделяйте внимание «Примечаниям» в разделе констант. Значения этих ячеек пересчитываются в относительные значения (относительно времени появления метки Delta) и далее в ячейки для переноса в Моторную часть в качестве базовой таблицы, которую потом можно будет использовать для клонирования в набор рабочих таблиц прошивки в программе **OUZ**.

	A	B	C	D	E	F	G	M	N	P	Q	R	S
1	Обороты, 1/мин	Период, сек	Интервал Base-Delta, сек	Selectr	Примечание	Код таймера		В текст программы FillIndex	Базовая таблица, град		Табличные значения		В текст программы FillTable
2						Index			Относительно ВМТ	Относительно Delta	Вещественное	FP8	
3													
4	Менее 617	-	-	1F	Зажигание по завершению метки Delta = 8 град.	DEC	HEX						
5	617	0,0972	0,0902		Ok	65535	FFFF	FTW0xFFFF;617 1/мин	8,00	0	1,87	239	FTB0xEF;8 град.
6	700	0,0857	0,0795	1E	Ok	57835	E1EB	FTW0xE1EB;700 1/мин	8,10	0,1	1,86	238	FTB0xEE;8,1 град.
7	800	0,0750	0,0696	1D	Ok	50606	C5AE	FTW0xC5AE;800 1/мин	8,20	0,2	1,85	237	FTB0xED;8,2 град.
8	900	0,0667	0,0619	1C	Ok	44983	AFB7	FTW0xAFB7;900 1/мин	8,30	0,3	1,85	236	FTB0xEC;8,3 град.
9	1000	0,0600	0,0557	1B	Ok	40485	9E25	FTW0x9E25;1000 1/мин	8,60	0,6	1,83	234	FTB0xEA;8,6 град.
10	1100	0,0545	0,0506	1A	Ok	36804	8FC4	FTW0x8FC4;1100 1/мин	9,30	1,3	1,77	227	FTB0xE3;9,3 град.
11	1180	0,0508	0,0472	19	Ok	34309	8605	FTW0x8605;1180 1/мин	10,00	2	1,72	221	FTB0xDD;10 град.
12	1270	0,0472	0,0438	18	Ok	31878	7C86	FTW0x7C86;1270 1/мин	12,00	4	1,58	202	FTB0xCA;12 град.
13	1400	0,0429	0,0398	17	Ok	28918	70F6	FTW0x70F6;1400 1/мин	14,00	6	1,44	184	FTB0xB8;14 град.
14	1500	0,0400	0,0371	16	Ok	26990	696E	FTW0x696E;1500 1/мин	16,00	8	1,29	166	FTB0xA6;16 град.
15	1690	0,0355	0,0329	15	Ok	23956	5D94	FTW0x5D94;1690 1/мин	18,00	10	1,15	147	FTB0x93;18 град.
16	1920	0,0313	0,0290	14	Ok	21086	525E	FTW0x525E;1920 1/мин	20,00	12	1,01	129	FTB0x81;20 град.
17	2200	0,0273	0,0253	13	Ok	18402	47E2	FTW0x47E2;2200 1/мин	22,00	14	0,86	110	FTB0x6E;22 град.
18	2530	0,0237	0,0220	12	Ok	16002	3E82	FTW0x3E82;2530 1/мин	24,50	16,5	0,68	87	FTB0x57;24,5 град.
19	2700	0,0222	0,0206	11	Ok	14994	3A92	FTW0x3A92;2700 1/мин	26,00	18	0,57	74	FTB0x4A;26 град.
20	2940	0,0204	0,0189	10	Ok	13770	35CA	FTW0x35CA;2940 1/мин	28,00	20	0,43	55	FTB0x37;28 град.
21	3200	0,0188	0,0174	0F	Ok	12652	316C	FTW0x316C;3200 1/мин	29,00	21	0,36	46	FTB0x2E;29 град.
22	3400	0,0176	0,0164	0E	Ok	11907	2E83	FTW0x2E83;3400 1/мин	30,00	22	0,29	37	FTB0x25;30 град.
23	3600	0,0167	0,0155	0D	Ok	11246	2BEE	FTW0x2BEE;3600 1/мин	30,00	22	0,29	37	FTB0x25;30 град.
24	3800	0,0158	0,0146	0C	Ok	10654	299E	FTW0x299E;3800 1/мин	30,00	22	0,29	37	FTB0x25;30 град.
25	4000	0,0150	0,0139	0B	Ok	10071	2757	FTW0x2757;4000 1/мин	30,00	22	0,29	37	FTB0x25;30 град.
26	4200	0,0143	0,0132	0A	Ok	9611	258B	FTW0x258B;4200 1/мин	30,00	22	0,29	37	FTB0x25;30 град.
27	4400	0,0136	0,0127	09	Ok	9089	2381	FTW0x2381;4400 1/мин	30,00	22	0,29	37	FTB0x25;30 град.
28	4600	0,0130	0,0121	08	Ok	8801	2261	FTW0x2261;4600 1/мин	30,00	22	0,29	37	FTB0x25;30 град.
29	4800	0,0125	0,0116	07	Ok	8530	2152	FTW0x2152;4800 1/мин	30,00	22	0,29	37	FTB0x25;30 град.
30	5000	0,0120	0,0111	06	Ok	8022	1F56	FTW0x1F56;5000 1/мин	30,00	22	0,29	37	FTB0x25;30 град.
31	5200	0,0115	0,0107	05	Ok	7766	1E56	FTW0x1E56;5200 1/мин	30,00	22	0,29	37	FTB0x25;30 град.
32	5400	0,0111	0,0103	04	Ok	7522	1D62	FTW0x1D62;5400 1/мин	30,00	22	0,29	37	FTB0x25;30 град.
33	5600	0,0107	0,0099	03	Ok	7291	1C7B	FTW0x1C7B;5600 1/мин	30,00	22	0,29	37	FTB0x25;30 град.
34	5800	0,0103	0,0096	02	Ok	6981	1B45	FTW0x1B45;5800 1/мин	30,00	22	0,29	37	FTB0x25;30 град.
35	6000	0,0100	0,0093	01	Ok	6772	1A74	FTW0x1A74;6000 1/мин	30,00	22	0,29	37	FTB0x25;30 град.
36	6200	0,0097	0,0090	00	Ok	6578	19B2	FTW0x19B2;6200 1/мин	30,00	22	0,29	37	FTB0x25;30 град.
37	Более 6200	-	-	11	УОЗ не изменяется = 30 град.	-	-	-		-	-	-	-
38													

	Константа Name	Значение	Краткое описание	Примечание
39				
40	LDelta	8	Угловое смещение метки Delta относительно ВМТ	Ok
41	LDelta	26	Угловой размер метки в градусах	Ok
42	Base	360	Угол-база в градусах	
43	Fn	10	Код частоты тактовых импульсов на таймере T1 F=0,727272727272727 Mhz	
44	LMaxUOZ	31	Ограничение на УОЗ 'сверху' в градусах относительно ВМТ. Не более 31 град.!	Ok
45	LMinUOZ	8	Ограничение на УОЗ 'снизу' в градусах относительно ВМТ. Не менее 8 град.!	Ok
46	LDTa	17	Угловой размер временного интервала отключения выходного сигнала коммутатора ТСЗ в град. Нижняя граница 4К 1/мин!	Ok
47	LCorrect	3	Угловая апертура ручного корректора УОЗ на 3 градусов в '+' и '-'	Ok
48	LDAD	3	Угловая апертура корректора нагрузки (при максимальной нагрузке)	Ok
49	LStopTime	4	Время [сек] детектирования режима СТОП	Ok

Рис. 10 Главная разметочная таблица

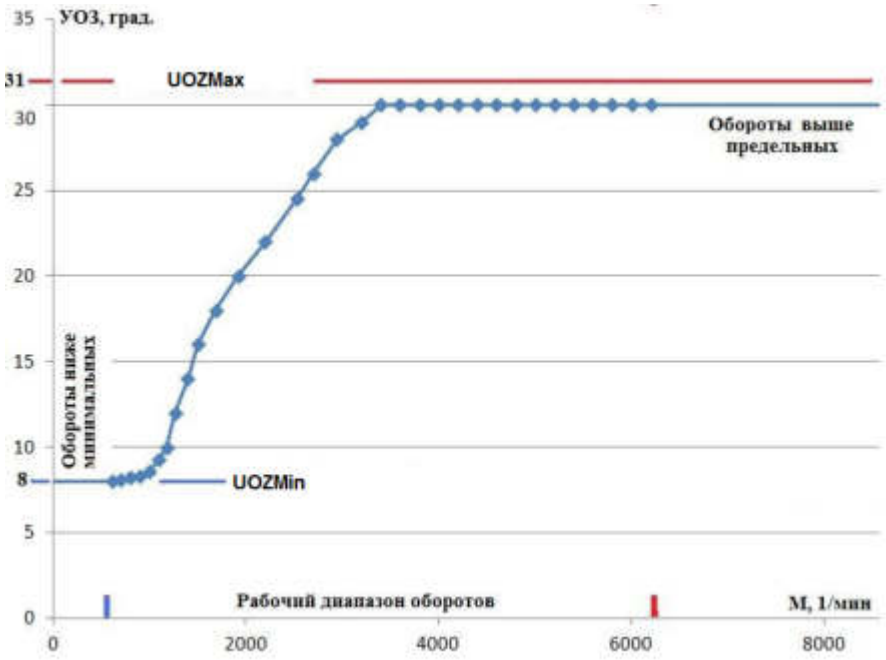


Рис. 11 Модельная зависимость УОЗ от оборотов

На рис. 11 представлена зависимость УОЗ от оборотов, запрограммированная в примере на рис.10. Кроме того, на рисунке изображена графическая интерпретация еще двух параметров Главной разметочной таблицы - UOZMax и UOZMin. Ядро при своей работе при изменении УОЗ внешней программой UOZ или ручным корректором, будет удерживать его значение в этом диапазоне. В случае попытки выхода из него включатся ограничители и будут активизироваться соответствующие флаги Ядра.

Необходимо сделать важное замечание. В силу особенностей примененного математического аппарата, а именно способность сохранять УОЗ постоянным при оборотах выше заданных в таблице на уровне последнего табличного, в случае когда нет необходимости регулировать УОЗ на высоких оборотах можно значительно уменьшить нижний предел контролируемых оборотов за счет высоких. Достигается это путем уменьшения тактовой частоты (увеличения Fn). Например, для представленной на рис. 8 и 9 ситуации можно увеличить в два раза Fn=20 и снизить верхний контролируемый порог оборотов до 4000 1/мин. Математический аппарат будет удерживать УОЗ на оборотах выше 4000 1/мин константой. Однако нижний контролируемый предел снизиться до с 617 до 320 1/мин, а остальные тридцать точек еще точнее опишут необходимую зависимость УОЗ от оборотов – рис.12.

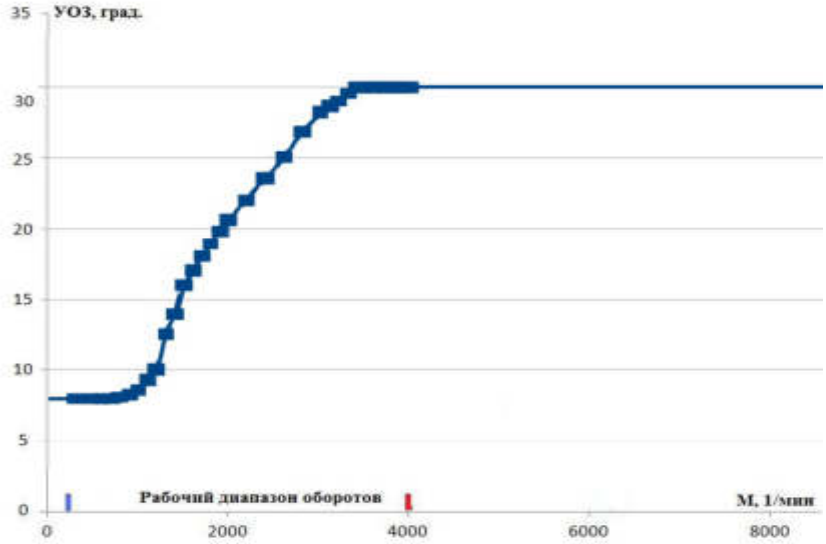


Рис. 12 Снижение нижнего уровня контролируемых оборотов.

Ячейка C46 задает параметр α для форм (7) и (8). Апертура ручного корректора УОЗ, работающего по форме (4) и (5) с плавной регулировкой УОЗ в зависимости от уровня напряжения на одном из входов АЦП задается в ячейки C47. В ячейке D46 указан нижний предел оборотов двигателя, при которых, заданные выше угловые пропорции сигналов управления зажиганием могут быть выдержены.

По мимо все вышеперечисленного, в разработанном Ядре добавлен экспериментальный корректор нагрузки УОЗ только для **одноцилиндрового четырехтактного двигателя.** Ячейка C48 определяет угловую апертуру корректора нагрузки – максимальное запаздывание зажигания в зависимости от динамики вращения маховика между тактами двигателя. Чем больше нагрузка двигателя, тем «позднее» зажигание.

Последний параметр для настройки это время - детектирования режима StopMode. Ячейка C49 задает время в секундах, а ячейка C65 содержит расчет константы StopCntMax в формате Q8.0 по формуле:

$$StopCntMax = round\left[\frac{\text{Время}}{65536}\cdot F\cdot\frac{360}{Base}\right] = round\left[\frac{\text{Время}\cdot 43945.3125}{1+Fn}\cdot\frac{1}{Base}\right] \qquad (9)$$

8. МОТОРНАЯ ЧАСТЬ

Как было указано в Разделе 2, Главный уровень (файл **MAIN.ASM**) или Моторная часть - обеспечивает смысловую связь Ядра с сигналами датчиков, формирователями импульсов зажигания. В Ядре определены «полезные» макросы и подпрограммы, которые можно использовать в Моторной части. Кроме того, из Моторной части есть доступ к регистрам состояния и портам управления Ядра и также общим регистрам для выполнения вычислений и манипуляций Таблица 3. Для удобства программирования были переопределены названия основных регистров процессора ATmega328p, и только часть (выделены зеленым цветом) из них можно использовать в Моторной части для работы:

Таблица 3.

328p	Ядро	Назначение и допустимые операции
r0		(R/W) Общий регистр.
r1		(R/W) Общий регистр.
r2	PSREG	(R) Временное хранение регистра SREG в программах обработки прерывания
r3	Zerro	(R) Регистр с нулевым значением 0x00
r4	Full	(R) Регистр с максимальным значением 0xFF
r5	Selectr	(R) Номер ячейки в текущей таблице УОЗ (стр. 10)
r6	DeltaL	(R) Ширина (младший и старший байт) выбранного регистром Selectr диапазона значений таймера T1. (стр.10)
r7	DeltaH	
r8	RaznostL	(R) Разность (младший и старший байт) между текущим значением отсчетов таймера T1 и табличным значением из выбранного регистром Selectr диапазона (стр. 10)
r9	RaznostH	
r10	N1L	(R) Значение счетчика таймера T1 (младший и старший байт) на предыдущем цикле работы
r11	N1H	
r12	N0L	(R) Значение счетчика таймера T1 (младший и старший байт) на текущем цикле работы
r13	N0H	
r14	NL	(R) Значение счетчика таймера T1 (младший и старший байт), используемый для работы Ядра. В зависимости от состояния бита №6 порта управления CoreSetup (см. ниже) эта регистровая пара либо повторяет значение пары N0, либо являются средним значением пары N0 и N1
r15	NH	
r16	A	(R/W) Общий регистр.
r17	B	(R/W) Общий регистр.
r18	C	(R/W) Общий регистр.
r19	D	(R/W) Общий регистр.
r20	Temp	(R/W) Регистр для использования в программах обработки прерывания
r21	DH1Chain	(R) Цепочки бит протокола защиты от импульсных помех датчиков. См. рис. 4.
r22	DH2Chain	
r23	Events	(R) Селектр событий. См. рис. 4.

Таблица 3.Продолжение

328p	Ядро	Назначение и допустимые операции
r24	Status	(R) Регистр состояния (нумерация от младшего бита к старшему) 7: LowMode - Режим низких оборотов (стр.4) 6: StopMode - Двигатель остановлен (стр.5) 5: UOZMax - Расчетный УОЗ ограничен снизу (рис.9) 4: UOZMin - Расчетный УОЗ ограничен сверху (рис.9) 3: DH2 - Состояние линии датчика 2 канала - PD3(int1) 2: DH1 - Состояние линии датчика 1 канала - PD2(int0) 1: Protect - Флаг выполнения протокола защиты от пульсаций 0: UARTCtrl - Флаг управления TblSelectr. UART/Ядро
r25	EStatus	(R) Расширенный регистр состояния 7: RError - Буфер приема UART заполнен 6: RnewCMD - В буфере приема UART есть команда 5: TError - Буфер передачи UART заполнен 4: TnotEmpty - Есть данные в буфере передачи UART 3: EOCR1BEmpty - Расширенный регистр EOCR1B пуст 2: Ign2 - Флаг пользователя 2 1: Ign1 - Флаг пользователя 1 0: Delta - Флаг виртуальной метки Delta (стр.8)
r26 (XL)		(R/W) Общий регистр (регистровая пара X).
r27 (XH)		
r28 (YL)		(R/W) Общий регистр (регистровая пара Y).
r29 (YH)		
r30 (ZL)		(R/W) Регистр (регистровая пара Z) для использования в программах обработки прерывания
r31 (ZH)		

Перечень «полезных» макросов Ядра приведен в Таблице 4.

Таблица 4

Макрос	Назначение и пример использования
store	Запись регистра в порт ввода/вывода. В макросе автоматически подставляется команда out или sts в зависимости от номера порта ввода/вывода. store 0x3F,A заменяется на out 0x3F,A store 0x40,A заменяется на sts 0x40,A
load	Чтение регистра из порта ввода/вывода. В макросе автоматически подставляется команда in или out в зависимости от номера порта ввода/вывода. load A,0x3F заменяется на in A,0x3F load A,0x40 заменяется на lds A,0x40
lsrw	Сдвиг в право регистровую пару – lsrw Z
subiw	Вычитание из регистровой пары константы - subiw X,0x1234
ldiw	Загрузка регистровой пары – ldiw Z,0x3456
FTW	Запись в ОЗУ двух байт, адресованных регистровой парой X с её автоматическим уменьшением. Используется для заполнения Границ диапазонов, рассчитанных по формуле (2). Таблица заполняется от старшего адреса к младшему.
FTB	Запись в ОЗУ байта, адресованного регистровой парой X с её автоматическим уменьшением. Используется для заполнения базовой таблицы A^{fp8}. Таблица заполняется от старшего адреса к младшему.
pusha	Сохранение (в стеке)/восстановление (из стека) всех общих регистров, выделенных зеленым цветом в таблице 3) . Параметров нет.
popa	
StartDelta	Начало виртуального импульса Delta. Сохранение результата оценки скорости вращения маховика. Расчет УОЗ по формуле (3).
StopDelta	Завершение виртуального импульса Delta. Сохранение значения оценки MDelta. Вычисление и установки величины EOCR1B по формулам (7) и (8).

Таблица 4.Продолжение

Макрос	Назначение и пример использования
SetLEDCtrl	Установить режим работы светодиода на плате SetLEDCtrl Режим Режим Роль светодиода на плате Arduino 0 - Отключен 1 - Состояние флага виртуальной метки Delta (EStatus.Delta) 2 - Состояние входной линии D2(PD2) на Arduino - стр.2 3 - Состояние входной линии D3(PD3) на Arduino - стр.2 4 - Состояние выходной линии A3(PC3) на Arduino - стр.2 5 - Состояние выходной линии A2(PC2) на Arduino - стр.2 6 - Состояние выходной линии A1(PC1) на Arduino - стр.2 7 - Состояние выходной линии A0(PC0) на Arduino - стр.2 8 - Состояние флага Status.StopMode 9 - Состояние флага Status.LowMode 10 - Состояние флага Status.UOZMax 11 - Состояние флага Status.UOZMin 12 - Состояние флага пользователя EStatus.Ign1 13 - Состояние флага пользователя EStatus.Ign2 14 - Состояние флага Status.UARTCtrl 15 - Светодиод включен
GetADC	Загрузить в регистр среднего значения выбранного канала АЦП (стр. 14). Номер канала - 4..7. В случае, если канал отключен, в то соответствующей ячейки будет максимальное значение. GetADC A,7

Перечень подпрограмм Ядра для использования в Моторной части приведен в Таблице 5.

Таблица 5

Подпрограмма	Назначение
StartIgnition	Расчет и установка OCR1A по формуле (3)
SetService	Вычисление величины EOCR1B по формуле (7) с последующей коррекцией результата по формуле (8) при необходимости
SetEOCR1B	Операция с 24 битным «расширенным» регистром EOCR1B

Помимо регистров состояния в Ядре есть еще три порта управления (таблица 6):

Таблица 6.

328p	Ядро	Назначение
GPIOR0	CoreSetup	Порт управление Ядром (1-выкл., 0 - вкл.) 7: DH2Disable - Отключение датчика DH2 6: AverageDisable - Отключение усреднения по двум отсчетам N Мнение автора - для двухтактного двигателя следует отключить функцию, для четырехтактного - включить 5: ServiceADisable - Отключен вызов события ServiceA 4: ServiceBDisable - Отключен вызов события ServiceA 3: C3734Disable - Отключен корректор для коммутатора XX.37.34 2: LoadDisable - Отключен корректор нагрузки 1: ADCDisable - Отключен АЦП. Автоматически устанавливается, если хотя бы один канал АЦП из четырех и сбарсывается в противном случае. При включенном канале (каналах) АЦП можно отключить все сразу установив этот флаг. 0: Резерв - Доступ к порту управления осуществляется с помощью макросов load и store и инструкций sbi, cbi, sbic, sbis.

Таблица 6. Продолжение

328p	Ядро	Назначение
GPIOR1	ECoreSetup	Расширенный порт управление Ядром 7: ADCCan7 - Канал 7 АЦП включен 6: ADCCan6 - Канал 6 АЦП включен 5: ADCCan5 - Канал 5 АЦП включен 4: ADCCan4 - Канал 4 АЦП включен 3: LEDCrtl3 - Действие светодиода на плате Arduino. 2: LEDCrtl2 1: LEDCrtl1 0: LEDCrtl0 Доступ к порту управления осуществляется только с помощью макросов load и store. Флаги ADCCan4 - ADCCan7 устанавливаются и сбрасываются автоматически. Битами LEDCrtl управляется через макрос SetLEDCtrl – (Таблица 4.Продолжение)
GPIOR2	RCoreSetup	Порт управления пользователя. Доступ к порту управления осуществляется только с помощью макросов load и store. Порт управления RCoreSetup ни как не влияет на работу Ядра и может быть использован для любых целей.

Все действия по настройке должны выполняться один раз на стадии запуска Ядра.

Рассмотрим подробно процесс формирования Моторной части для Главной разметочной таблицы из рис.10. для одноцилиндрового четырехтактного двигателя с оптическим датчиком или датчиком Холла, который выдают один импульс шириной 26 градусов за оборот двигателя, смещенный относительно ВМТ на 8 градусов. Ниже приведен типовой листинг с подробными комментариями. В фале **MAIN.ASM** строка или её часть, начинающаяся с символа «;» игнорируется при создании прошивки.

Зеленым цветом отмечены поля, которые заполнил пользователь при настройкой системы под свой двигатель. Собственно, в этом и заключается весь необходимый процесс «программирования».

Все параметры просто переносятся их Главной разметочной таблицы в прошивку. Самое сложное – скопировать ячейки M5-M36 в тело подпрограммы FillIndex (64 - 95 строка), а ячейки S5-S36 в тело подпрограммы FillTable (97 - 128 строка).

Обращаю внимание на строки 8 и 35. Каждая из них имеет две альтернативные редакции для двух систем зажигания (Рис.8, формулы 7 и 8). Строки на голубом фоне – для системы зажигания на основе коммутатора XX.37.34.

№	Метка	Команда	Комментарии
1		.include "m328pdef.inc"	Подключение файла с основными определениями микроконтроллера 328p

```
;*****************************************     Карта входов и выходов
; Порт B
; PB0(icp)          in      !Свободен
; PB1(osc1)         in      !Свободен
; PB2(ss)           in      !Свободен
; PB3(mosi)         in      !Свободен
; PB4(miso)         in      !Свободен
; PB5(sck)          out     !Вспомогательный светодиод на Ардуино
;*****************************************
; Порт D
; PD0(rxd)                  !UART обмен
; PD1(txd)                  !UART обмен
; PD2(int0)         in      !Вход датчика DH1
; PD3(int1)         in      !Вход датчика DH2(отключен)
; PD4(t0)           in      !Свободен
; PD5(t1)           in      !Соединить с PD6
; PD6(ain0)         out     !Соединить с PD5
; PD7(ain1)         in      !Свободен
;*****************************************
; Порт C
; PC0(adc0)         out     !Сигнал зажигания выход
; PC1(adc1)         in      !Свободен
; PC2(adc2)         in      !Свободен
; PC3(adc3)         in      !Свободен
; PC4(adc4/scl)     in      !Канал АЦП выбора таблицы
; PC5(adc5/sda)     in      !
;       adc6        in      !
;       adc7        in      !Канал АЦП корректора
```

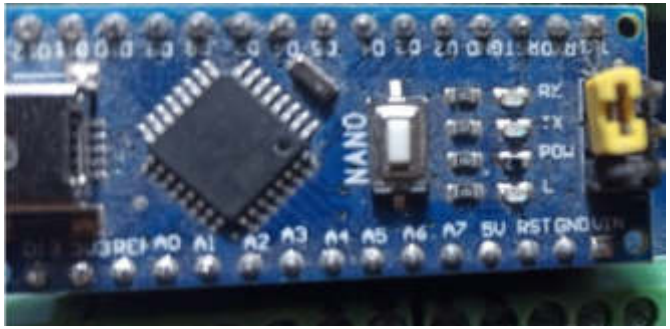
2	#define	VersionName	"1Ц.ХТ "	Определение версии прошивки. До 60 символов строка произвольная.
3	#define	_Betta	8.0	Угловое расстояние от ВМТ до метки Delta, β в градусах
4	#define	_Delta	26.0	Угловой размер метки Delta, Δ в градусах
6	#define	_UOZMax	31.0	Максимальный угол опережения зажигания, в градусах
7	#define	_UOZMin	8.0	Минимальный угол опережения зажигания, в градусах

8	#define	_Alfa 12.0	Угловое расстояние отключения сигнала зажигания после метки Delta, α в градусах
8	#define	_Alfa 120.0	Альтернативная редакция для коммутатора XX.37.34
9	#define	_StopTime 4.0	Время (сек) детектирования режима остановки, сек
10	#define	_Manual 13.0	Угловая апертура ручного корректора УОЗ на +/- (не более половины Delta). Ручной корректор отключается при ADCCanCorSelectr = 0. Середина кода АЦП соответствует сдвигу 0 градусов. Расчет действия корректора по формулам (4) и (5).
11	#define	_Load 3.0	Угловая апертура корректора нагрузки в "-" (не более Delta)
12		.equ Base = 360	Угол-база в градусах
13		.equ Fn = 10	Код частоты тактовых импульсов, поступающих на таймер T1
14		.equ TblCount = 2	Количество таблиц УОЗ 2^N (N=1, 2, 4, 8, 16, 32). Номер активной таблицы переключается напряжением на входе канала АЦП ADCCanTblSelectr. При ADCCanTblSelectr = 0 номер таблицы выбирается максимальным из доступных.
15		.equ ADCBufferSize = 8	Длина кольцевых буферов каналов АЦП (2, 4, 8, 16, 32)
16		.equ LoadBufferSize = 32	Длина кольцевого буфера корректора нагрузки (32, 64, 128)
18		.equ ADCCanTblSelectr = 4	Номер канала мультиплексора датчика ДАД/ДПЗД (выбор активной таблицы)
19		.equ ADCCanCorSelect = 7	Номер канала мультиплексора ручного корректора УОЗ
20		.equ ADCCanA = 0	Номер канала А пользователя. ADCCanA =0 – канал отключен.
21		.equ ADCCanB = 0	Номер канала Б пользователя. ADCCanB =0 – канал отключен.
22		.equ ProtectClk = 6	Интервал дискретизации протокола защиты от пульсаций в мкс
23		.equ ProtectCount = 2	Количество повторных чтений протоколом защиты с интервалом ProtectClk после первого изменения линии датчика. Значение может быть в пределах 1-7.
24		.equ CmdLineBit = 0	Номер бита сигнала управления порта C.
25		.equ StartDeltaCode = 0b00001010	Код начала метки Delta на цилиндре (Рис. 6а)
26		.equ StopDeltaCode = 0b00000101	Код завершения метки Delta на цилиндре (Рис. 6а)
27		.include "core.inc"	Подключение Ядра (сегмент кода, сегмент данных)
28		.dseg	Согласование адресов Ядра и Моторной части. Если необходимы дополнительные переменные, то между строками 29 и 30 их можно определить с помощью директивы .byte После строки 31 начинается код Моторной части.
29		.org EndCoreDseg	
		;Переменная: .byte 16	
30		.CSEG	
31		.org EndCoreCSeg	

После начальных настроек Ядра и настройки всей системы в целом **один раз** вызывается Ядром подпрограмма reset. Здесь следует настроить входные и выходные линии, не занятые Ядром. Предпочтительно в качестве выходных линий использовать линии порта PORTC. Кроме того, здесь необходимо установить режим работы светодиода на плате Arduino. Только после выполнения этой подпрограммы происходит запуск Ядра в целом. В этой подпрограмме можно при необходимости манипуляций или вычислений использовать все общие регистры и регистры для программ обработки прерываний из таблицы 3, выделенные зеленым и желтым цветом.

32	reset:	sbi DDRC,CmdLineBit	Назначение линии PC0(adc0) контролера для работы на выход
33		cbi PORTC,CmdLineBit	Деактивировать выходную линию
34		SetLEDCtrl 1	Светодиод на Arduino отражает состояние флага виртуальной метки Delta
35		ldi A,0b10111100	Настройка параметров Ядра
35		ldi A,0b10110100	Альтернативная редакция для коммутатора XX.37.34
36		store CoreSetup,A	
37		ret	Возврат в вызывающую программу

Ядро устроено как совокупность программ обработки прерываний во время работы «основной» зацикленной программы. У неё много своих функций, связанных с обменом по UART интерфейсу, АЦП, буферами для усреднения и прочее. На каждом своем цикле «основная» программ производит вызов подпрограммы loop Моторной части. В этой подпрограмме можно при необходимости манипуляций или вычислений использовать все общие регистры из таблицы 3, выделенные зеленым цветом. В простейшем случае эта подпрограмма может состоять из одной инструкции ret - возврат. Ниже приведен вариант, обеспечивающий оперативный (без перепрошивки) способ включения и отключения корректора нагрузки перемычкой на разъеме для внутрисхемного программирования Arduino путем изменения порта управления CoreSetup.



38	loop:	sbic PINB,3	В зависимости от уровня напряжения на выводе, к которому подключена перемычка или включить или выключить флаг LoadDisable порта управления и обнулить расчетную добавку к УОЗ
39		rjmp loop1	
40		sts PLoad+1,zerro	
41		sts PLoad,zerro	
42		sbi CoreSetup,LoadDisable	Выключить корректор
43		ret	Возврат в вызывающую программу
44	loop1:	cbi CoreSetup,LoadDisable	Включить корректор
45		ret	Возврат в вызывающую программу

Следующий блок Моторной части уже был предварительно описан на стр. 6 и 7. По сути это подпрограмма обработки прерывания после выполнения протокола защиты Ядром. Подпрограмма может быть организована как Case по значениям регистра Events. Регистр Events несет информацию о произошедшем событии (состояние входных линий) с предысторией во времени, зафиксированных при изменениях на любом из каналов (Рис. 4).
В любом случае должно быть два ключевых события: "Начало метки Delta" и "Конец метки Delta". В них следует выполнить макросы StartDelta и StopDelta при двух разных значения Events. Никакие регистры (кроме temp, ZL, ZH) изменять в подпрограмме нельзя без предварительного сохранения в стеке и последующего их восстановления в стеке. Для этого можно использовать как отдельные команды push/pop, так и макросы pusha/popa.
Обращаю внимание на строку с номером 54. На странице 4 было указано, что в режиме Низких оборотов (LowMode) искрообразование через механизм Ignition и UOZMinError (см. ниже) не происходит. Указанная строка и обеспечивает искрообразование во время детектирования события StopDelta.

46	Actions:	cpi Events, StartDeltaCode	Сравнение регистра Событий с признаком начала метки на DH1
47		brne C_VMT	Если не равны, то проверка следующего кода
48		**StartDelta**	Макрос отметки времени (начало виртуального импульса Delta)
49		cbi PORTC,CmdLineBit	Отключить сигнал зажигания принудительно в начале виртуального импульса Delta
50	C_exit:	ret	Возврат в вызывающую программу
51	C_VMT:	cpi Events, StopDeltaCode	Сравнение регистра Событий с признаком окончания метки на DH1
52		brne C_exit	Если не равны, то выход из подпрограммы
53		**StopDelta**	Макрос отметки времени (окончание виртуального импульса Delta)
54		sbi PORTC,CmdLineBit	Включить сигнал зажигания принудительно на конце метки DELTA
55		ret	Возврат в вызывающую программу

Подпрограмма UOZMinError вызывается при попытке установить слишком низкий уровень OCR1A (стр.4). В рассматриваемом случае ограничимся подачей сигнала зажигания по этому событию. В этой подпрограмме можно при необходимости манипуляций или вычислений использовать все общие регистры и регистры для программ обработки прерываний из таблицы 3, выделенные зеленым и желтым цветом.

56	UOZMinError:	sbi PORTC,CmdLineBit	Включить сигнал зажигания
57		ret	Возврат в вызывающую программу

Основная подпрограмма для формирования искры (Рис.3). Это неотключаемая программа обработки прерывания по превышению значения таймера T1 с OCR1A. В режиме останова и низких оборотов (LowMode) вызова этой подпрограммы не произойдет. Никакие регистры (кроме temp, ZL, ZH) изменять в подпрограмме нельзя без предварительного сохранения в стеке и последующего их восстановления в стеке.

58	Ignition:	sbi PORTC,CmdLineBit	Включить сигнал зажигания
59		ret	Возврат в вызывающую программу

Отключаемая программа обработки прерывания (по превышению значения таймера T1 с OCR1A вне импульса Delta) - Рис.3. В режиме останова вызова этой подпрограммы не произойдет. Никакие регистры (кроме temp, ZL, ZH) изменять в подпрограмме нельзя без предварительного сохранения в стеке и последующего их восстановления в стеке.

60	ServiceA:	ret	Возврат в вызывающую программу

Неотключаемая программа обработки прерывания (по превышению значения таймера T1 с OCR1B вне импульса Delta) - Рис.3. Единственная роль этой подпрограммы – выключение линии управления зажиганием в расчетный момент времени. Никакие регистры (кроме temp, ZL, ZH) изменять в подпрограмме нельзя без предварительного сохранения в стеке и последующего их восстановления в стеке.

61	Service:	cbi PORTC,CmdLineBit	Отключение сигнала зажигания
62		ret	Возврат в вызывающую программу

Отключаемая программа обработки прерывания (сравнение Таймера T1 с OCR1B внутри импульса Delta) - Рис.3. Никакие регистры (кроме temp, ZL, ZH) изменять в подпрограмме нельзя без предварительного сохранения в стеке и последующего их восстановления в стеке.

63	ServiceB:	ret	Возврат в вызывающую программу

Заполнение базовой таблицы A^{fp8} и таблицы границы диапазонов с помощью макросов FTW и FTB

64	FillIndex:	FTW 0xFFFF	617 1/мин
65		FTW 0xE1EB	700 1/мин
66		FTW 0xC5AE	800 1/мин
67		FTW 0xAFB7	900 1/мин
68		FTW 0x9E25	1000 1/мин
69		FTW 0x8FC4	1100 1/мин
70		FTW 0x8605	1180 1/мин
71		FTW 0x7C86	1270 1/мин
72		FTW 0x70F6	1400 1/мин
73		FTW 0x696E	1500 1/мин
74		FTW 0x5D94	1690 1/мин
75		FTW 0x525E	1920 1/мин
76		FTW 0x47E2	2200 1/мин
77		FTW 0x3E82	2530 1/мин
78		FTW 0x3A92	2700 1/мин
79		FTW 0x35CA	2940 1/мин
80		FTW 0x316C	3200 1/мин
81		FTW 0x2E83	3400 1/мин
82		FTW 0x2BEE	3600 1/мин
83		FTW 0x299E	3800 1/мин
84		FTW 0x2789	4000 1/мин
85		FTW 0x25A7	4200 1/мин
86		FTW 0x23F1	4400 1/мин
87		FTW 0x2261	4600 1/мин
88		FTW 0x20F2	4800 1/мин
89		FTW 0x1FA1	5000 1/мин
90		FTW 0x1E6A	5200 1/мин
91		FTW 0x1D49	5400 1/мин
92		FTW 0x1C3D	5600 1/мин
93		FTW 0x1B44	5800 1/мин
94		FTW 0x1A5B	6000 1/мин
95		FTW 0x1982	6200 1/мин
96		ret	Возврат в вызывающую программу

97	FillTable:	FTB 0xEF	8 град.
98		FTB 0xEE	8,1 град.
99		FTB 0xED	8,2 град.
100		FTB 0xEC	8,3 град.
101		FTB 0xEA	8,6 град.
102		FTB 0xE3	9,3 град.
103		FTB 0xDD	10 град.
104		FTB 0xCA	12 град.
105		FTB 0xB8	14 град.
106		FTB 0xA6	16 град.
107		FTB 0x93	18 град.
108		FTB 0x81	20 град.
109		FTB 0x6E	22 град.
110		FTB 0x57	24,5 град.
111		FTB 0x4A	26 град.
112		FTB 0x37	28 град.
113		FTB 0x2E	29 град.
114		FTB 0x25	30 град.
115		FTB 0x25	30 град.
116		FTB 0x25	30 град.
117		FTB 0x25	30 град.
118		FTB 0x25	30 град.
119		FTB 0x25	30 град.
120		FTB 0x25	30 град.
121		FTB 0x25	30 град.
122		FTB 0x25	30 град.
123		FTB 0x25	30 град.
124		FTB 0x25	30 град.
125		FTB 0x25	30 град.
126		FTB 0x25	30 град.
127		FTB 0x25	30 град.
128		FTB 0x25	30 град.
129		ret	Возврат в вызывающую программу

9. АППАРАТНЫЕ СРЕДСТВА ПРОГРАММРОВАНИЯ МК

Как было указано в разделе 2, предполагается использовать готовые Arduino совместимые платы с кварцевым резонатором 16 МГц, на которых уже установлен МК и все необходимые для его работы элементы. Основное отличие между платами – наличие или отсутствие UART интерфейса. На платах типа Arduino Nano или UNO такой интерфейс установлен. Если последнего нет на выбранной плате, например Arduino Pro mini или просто самодельная плата с микроконтроллером Atmega328P, то возникает необходимость в приобретении дополнительного блока USB-UART интерфейса, который может быть выполнен по одному из базовых проектов:

а) б) в)

Рис. 13 Базовые проекты USB-UART интерфейса

Во-первых, самый простой четырех проводной – Рис.13 а). По двум проводам осуществляется обмен данными, два других используются для питания устройства от USB порта. Во-вторых, аналогичный четырех проводной интерфейс в корпусе – рис.13 б). В третьих, пяти проводной преобразователь, который имеет одну дополнительную линию для активации линии Reset микроконтроллера при активации COM порта, закрепленного за разъемом USB на корпусе ПК. Наверное, можно найти и другие варианты преобразователей.

Наличие корпуса варианта рис.13 б) является его солидным преимуществом. Недостатком этого варианта, как и варианта рис.13 а), является отсутствие пятого провода, по которому происходит сброс МК (RESET) и запуск встроенной **программы начальной загрузки, с помощью которого и будет производить передачу прошивки в МК**. Без определенной сноровки при запуске программы прошивки использовать эти интерфейсы не получиться – необходимо в нужный момент нажимать на кнопку Reset на плате Arduino. Несмотря на этот недостаток, автором для работы с готовым изделием на базе Arduino Pro mini используется вариант б). В лаборатории все отладочные работы проводились с платами Arduino Nano или UNO.

Кроме того, так как основой этих преобразователей является специализированная микросхема с USB интерфейсом, то необходимо установить в операционную систему персонального компьютера её драйвер – рис. 14.

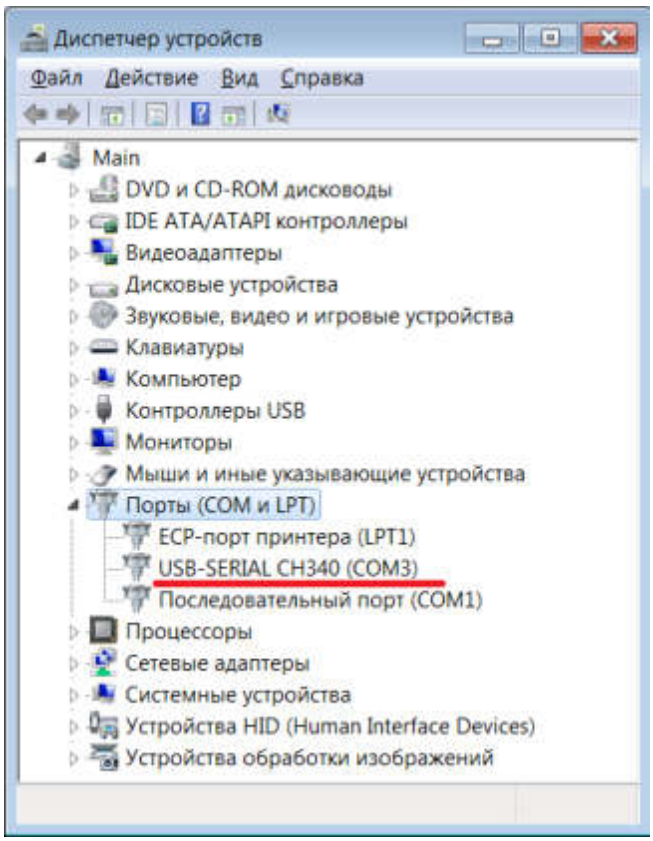
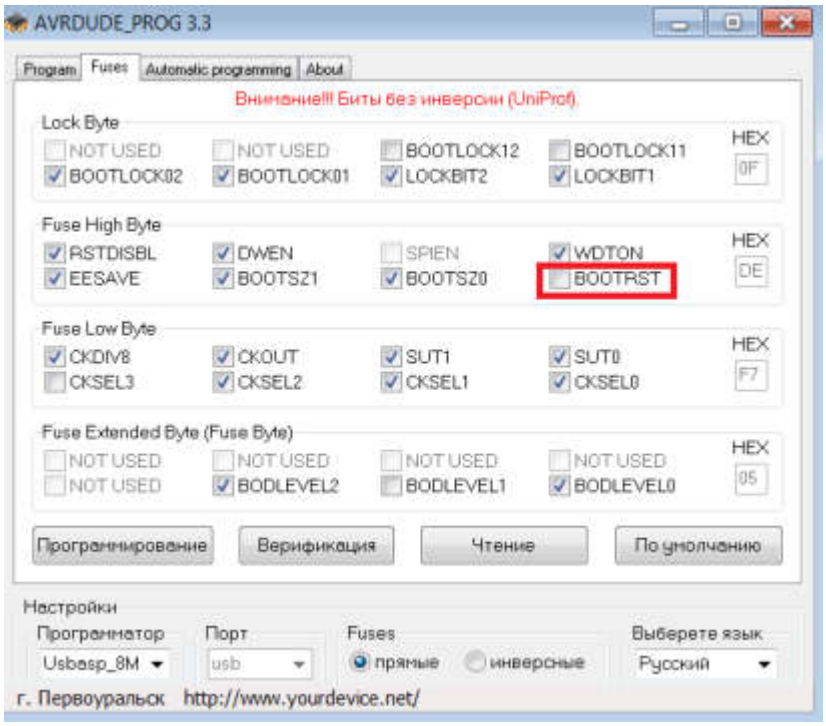


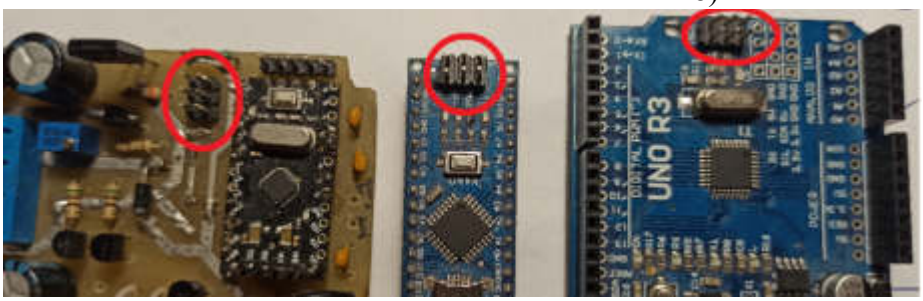
Рис. 14 Подключенное устройство

Если в наличии есть один из вариантов внутрисхемного программатора, например USBASP как на рис. 15 а), то **первичную прошивку готовым HEX файлом можно осуществить через него** с помощью программы AVRDUDE_PROG или ей подобной. На плате Arduino предусмотрен специальный разъем. На «пилотной» плате автора такой разъем установлен тоже – рис. 15 в).

а)



б)



в)

Рис. 15 Применение внутрисхемного программатора

Еще один весомый плюс, о котором упоминалось в разделе 2 и который дает наличие такого программатора это возможность отключения начального загрузчика по средствам прямого программирования FUSE МК BOOTRST. В случае, представленном на рис.15 б), **начальный загрузчик включен**.

10. КОМПИЛЯЦИЯ И ПРОШИВКА С ПОМОЩЬЮ НАЧАЛЬНОГО ЗАГРУЗЧИКА

Ниже приведено описание процесса компиляции и прошивки под управлением операционной системы Windows. Состав исходного набора файлов для формирования файла прошивки:

avrasm2.exe	Компилятор ASM для микроконтроллеров AVR
avrdude.exe	Консольная программа, предназначенная для прошивки микроконтроллеров AVR
avrdude.conf	
libusb0.dll	
m328pdef.inc	Подключаемый файл с определением имен регистров и битов МК ATmega328P
core.inc	Подключаемый файл – Ядро прошивки
main.asm	Моторная часть
main.hex	Собранный файл прошивки
C.bat	Пакетный файл для компиляции
W.bat	Пакетный файл для записи прошивки в МК с помощью механизма начального загрузчика Arduino.
Главная разметочная таблица.ods	Главная разметочная таблица в формате OpenOffice.

Перед началом работы следует определиться с номером USB порта, к которому подключена плата. Это действие необходимо для корректировки пакетного файла W.bat, в котором следует указать этот номер (выделено желтым цветом):
W.bat:

avrdude -v -p atmega328p -c arduino -P COM7 -b 115200 -D -U flash:w:"main.hex":i

Номер порта можно узнать в Диспетчере устройств – рис. 14. Иногда возникает проблема с установкой скорости (зеленый цвет). Как следует из личных наблюдений автора, на клонах плат Arduino Nano и Arduino Pro mini, приобретенных на Алиэкспресс, установлена старая версия начального загрузчика, который работает на скорости 57600. На клонах плат Arduino UNO уже новый загрузчик, который работает на скорости 115200. На современных и оригинальных платах – 115200. В общем, один раз придется экспериментально попробовать обе скорости передачи. Альтернатива - сразу пользоваться внутрисхемным программатором для записи прошивки или записи «свежего» варианта начального загрузчика Arduino. Для компиляции прошивки и дальнейшей работы интерфейсной программы **UOZ номер порта и его скорость не имеют значения**.

Редактировать файл Моторной части (перенос информации из Главной разметочной таблицы), а также выполнять пакетные файлы удобно в **FAR Manager** - консольный файловый менеджер для операционных систем семейства Windows.

Дальше всё просто! Первый шаг - компиляция. Для осуществления этого действия следует запустить пакетный файл **C.bat**. При отсутствии ошибок образуется файл **main.hex** – готовая прошивка. Второй шаг – запись готовой прошивки в МК по средствам пакетного файла **W.bat.** После завершения прошивки МК автоматически перезапустится и приблизительно через 2-4 секунды Ваша прошивка получит управление. При необходимости, в дальнейшем можно с помощью внутрисхемного программатора отключить начальный загрузчик, так как это было описано в предыдущем разделе. Скорость запуска прошивки в этом случае сократиться до **20-40 мс,** что дает возможность использования предлагаемой системы в без батарейных блоках зажигания.

11. ИНТЕРФЕЙСНАЯ ЧАСТЬ UOZ

Интерфейсная часть создана в среде программирования Lazarus (Free Pascal) версии 2.0 с дополнительным установленным компонентом для работы с COM портами - Sdpo. После запуска UOZ происходит автоматический поиск подключенных плат Arduino с соответствующей прошивкой. При подключении нескольких устройств программа выберет устройство с меньшим номером порта.

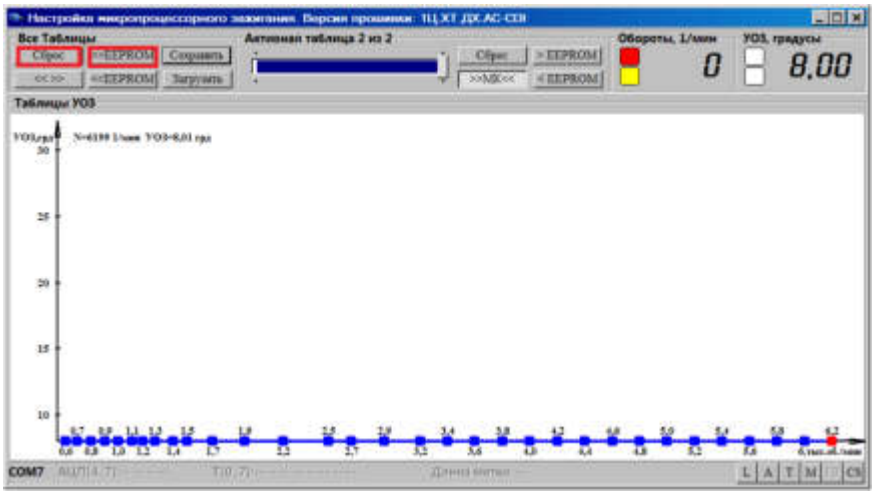
В случае первого подключения платы Arduino на экране появится изображение, представлено на рис. 16 а). Как было указано в разделе 5, МК прочитал **изначально пустую** область энергонезависимой памяти и в соответствии с значением констант заполнил свою оперативную память для работы как с набором таблиц A_k^{fp8} . Первый шаг, который необходимо сделать в этом случае - произвести сброс таблиц с базовое состояние и записать их в энергонезависимую память. Для этого необходимо последовательно нажать на кнопки «**Сброс**» и «**>>EPROM**» (выделено красным), при этом внешний вид программы будет таким, как на рис.16 б). На этом этапе все таблицы устанавливаться одинаковыми и численно равными базовой – как в Главной разметочной таблице. На экране все таблицы наложены друг на друга.

Программа UOZ все параметры для своей работы считывает из МК, поэтому она одна на все возможные версии системы, текстовое описание которых также отображается на верхней части формы приложения OUZ. В нижней части формы указан номер COM порта, через который происходит обмен данными между МК и приложением OUZ.

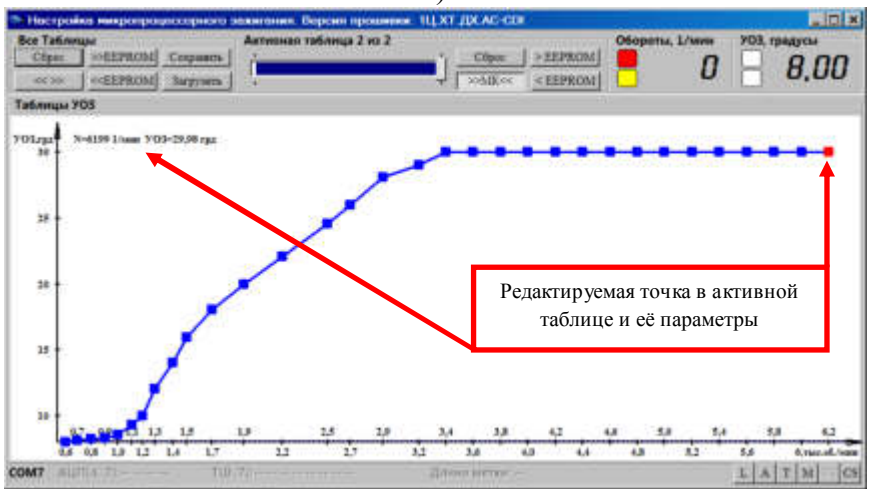
Рассмотрим секцию управления «Все Таблицы». Кроме уже описанных, на ней присутствуют кнопок управления - рис. 16:

<<EEPROM	- скопировать таблицы A_k^{fp8} из энергонезависимой памяти в ОЗУ для работы (выполняется автоматически при запуске системы);
Сохранить	- сохранение на диск компьютера все таблицы из ОЗУ;
Загрузить	- загрузка с диска компьютера все таблицы в ОЗУ;
<< >>	- зарезервировано для работы с табличном режиме.

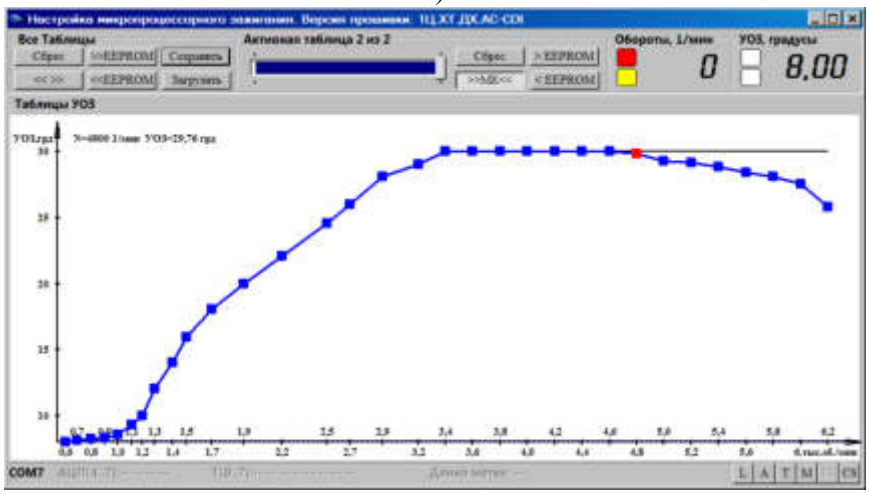
Добавлю, что программа и сама производит каждые две минуты автосохранение всех таблиц на диске в файлах «**autosave.u***», где вместо «*» указано количество таблиц в наборе. При нормальном завершении работы программы происходит еще одно контрольное сохранение в файл «**exitsave.u***». Все они могут быть в последствии загружены в ОЗУ МК. Напомню, что при выключении питания или повторном запуске программы OUZ таблицы в ОЗУ будут заменены таблицами из энергонезависимой памяти, поэтому не забывайте нажимать кнопку «**>>EEPROM**».



а)



б)



в)

Рис. 16 Программа UOZ
(первый запуск, двигатель остановлен)

Следующая секция управления «Активная таблица». В названии секции указано общее количество страниц и номер активной (рабочей) таблицы в данный момент времени. Как было сказано выше, номером активной таблицы управляет МК в зависимости от значения соответствующего канала АЦП. При работающей программе OUZ эту функцию можно и нужно «отбирать» у МК **для редактирования в визуальном режиме значений ячеек выбранной таблицы (Активной таблице) даже на работающем двигателе**. Для выполнения редактирования следует выбрать необходимую точку и просто или передвинуть по вертикали или покрутить «колесико» на мышке. На рис. 16 б) выбрана точка самых высоких оборотов. Численное значение оборотов и угла опережения зажигания **в абсолютных величинах** на этой точке указан справа перед осью ординат.Как только сдвинете одну точку от исходного значения как на рис. 16 в), будет видно и другая таблица, которая была скрыта за активной (в прошивке сейчас только две таблицы).

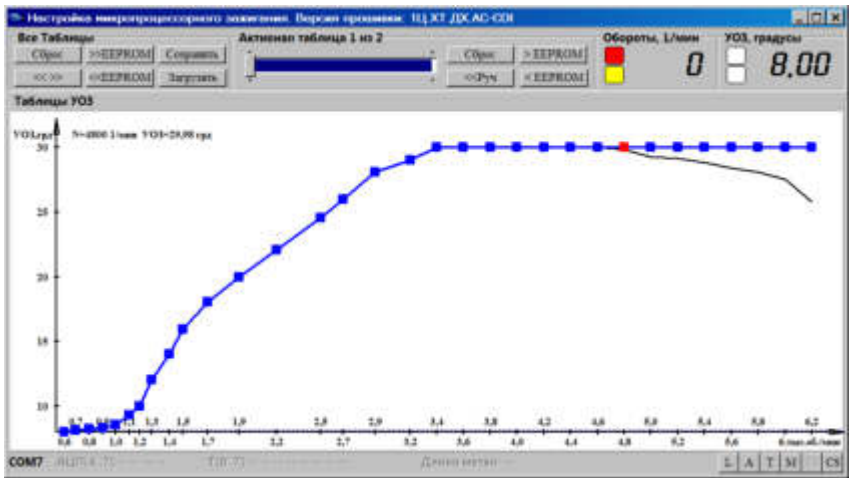
Для переключения программы в режим ручного выбора Активной таблицы следует нажать на кнопку «**>>МК<<**» - рис. 17 а). При этом активизируется переключатель таблиц и кнопка поменяет имя на «**<<Руч.**», как бы указывая на него - рис. 17 б). После этого можно выбрать номер нужной таблицы и соответствующая зависимость УОЗ будет выделена синим цветом - рис. 17 в).

Роль остальных органов управления секции Активная таблица заключается в следующем:

Сброс	- сбросить Активную таблицу в исходное состояние (как при первом запуске);
>EEPROM	- копировать Активную таблицу из ОЗУ в энергонезависимую память;
<EEPROM	- копировать Активную таблицу из энергонезависимой памяти в ОЗУ.

а)

б)



в)

Рис. 17 Переключение режим ручного выбора Активной таблицы

Следующие секции управления программы удобно рассмотреть на работающем «двигателе» – рис. 18. В рабочем поле программе появилось синие перекрестие, которое указывает на рабочую точку двигателя в данный момент. Информация обновляется четыре раза в секунду.

Секция «**Обороты**» отражает численное значение оборотов двигателя и состояние двух флагов состояния: Полная остановка – красный квадрат и низкие обороты – желтый квадрат на Рис. 17.

Секция УОЗ, отражает численное значение угла опережения зажигания. Кроме того может отражать состояние флагов ограничений УОЗ сверху и снизу – белые квадраты (верхний или нижний) будут окрашены желтый цвет при наступлении этих событий. Еще раз обращаю внимание, что кнопку «**>>EEPROM**» следует нажимать при изменении таблиц в ОЗУ МК. Так при переходе с состояния Рис.17 на состояние Рис. 18 она не была нажата и изменения не сохранились. Возможно, позднее в программу будет добавлен контроль изменений с выдачей соответствующего предупреждения при выходе из неё. Кроме того, напоминаю, что на работающем двигателе эту операцию произвести нет возможности и кнопки работы с энергонезависимой памятью недоступны.

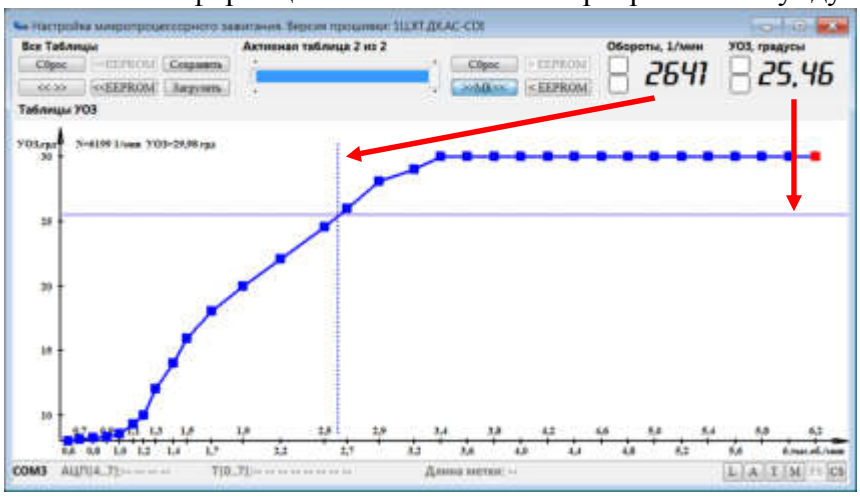


Рис. 18 Программа UOZ
(другой компьютер, «двигатель» запущен)

На нижней части рабочего поля программы (Рис. 19) расположены поля для отражения технической информации: средние значения четырех каналов АЦП, восьми дополнительных параметров и оценка длины метки Delta. Первые два набора отражаются в шестнадцатеричном виде. Длина метки отображается в градусах. Каждое из этих наборов активизируется при нажатых кнопках «А», «Т», «М».

Единственно полезной ролью для Пользователя кнопки «Т» является то, что при её нажатом состоянии секция обороты продолжает работать вплоть до 100 оборотов (зависит от параметров Fn и Delta) за счет оценки частоты вращения по служебным данным, передаваемым при её включении. В противном случае секции обороты перестает функционировать при включении режима Низкие обороты (LowMode).

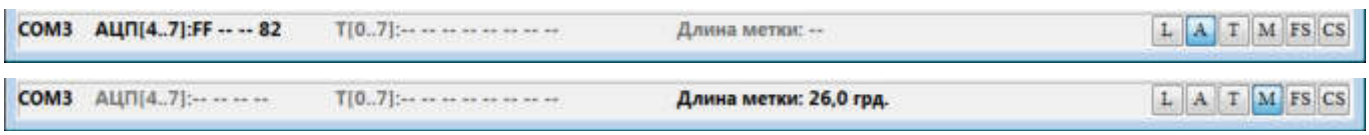


Рис. 19 Кнопки «А» и «М»

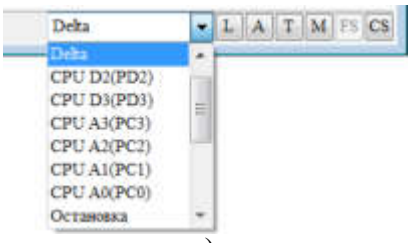
Оценку длины метки Delta можно использовать для уточнения её длины в случаях если при изготовлении датчика по какой-то причине произошла ошибка или вообще её размер не известен. В этом случае при произвольном задании прочих параметров следует точно заполнить строки 12, 13, 25, 28, 35 исходного текста прошивки. Далее следует запрограммировать МК, сопрячь с программой UOZ и придать движение маховику, например шуруповертом. Измеренное значение Delta позволит скорректировать остальные параметры прошивки.

При нажатии на кнопку «L» справа от нее появиться переключатель режима работы светодиода на плате Arduino, в котором он будет работать до очередного перезапуска системы - рис. 20 а).

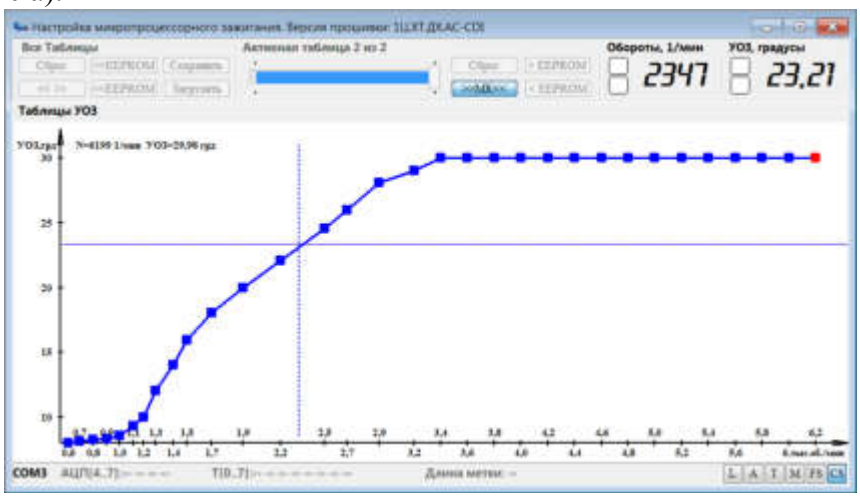
При любой нажатой кнопки («A», «T», «M») активизируется кнопка «FS», нажатии на при которую происходит запись внеочередной строки в файл протокола «core.log».

Очередная запись (2 раза в секунду) в протокол осуществляется при нажатой кнопке «CS», при этом большинство органов управления будут заблокировано – рис. 20 б).

Содержание файла протокола даст возможность проведения дополнительного анализа работы двигателя.



а)



б)

Рис. 20 Кнопки «L» и «CS»

Файл текстовый и его формат следующий:

1	2	3	4	5	6	7	8	9	10	11	12
Дата	Время	N(-1)	N(0)	M	1/мин	УОЗ	LOW	STOP	Status	EStatus	Таблица

1. Дата.
2. Время.
3. Значения счетчика таймера T1, накопленное от метки StopDelta до StartDelta на предыдущем обороте двигателя (цепочка →N0→N1 стр.4).
4. Значения счетчика таймера T1, накопленное от метки StopDelta до StartDelta на текущем обороте двигателя (цепочка →N0→N1 стр.4).
5. Значения счетчика таймера T1, накопленное от метки StartDelta до StopDelta.
6. Расчетное значение скорости вращения двигателя.
7. Угол опережения зажигания.
8. Флаг низких оборотов.
9. Флаг полной остановки двигателя.
10. Регистр состояния Status в битовом представлении.
11. Регистр расширенного состояния EStatus в битовом представлении.
12. Номер активной таблицы.