

1. Входные и выходные сигналы

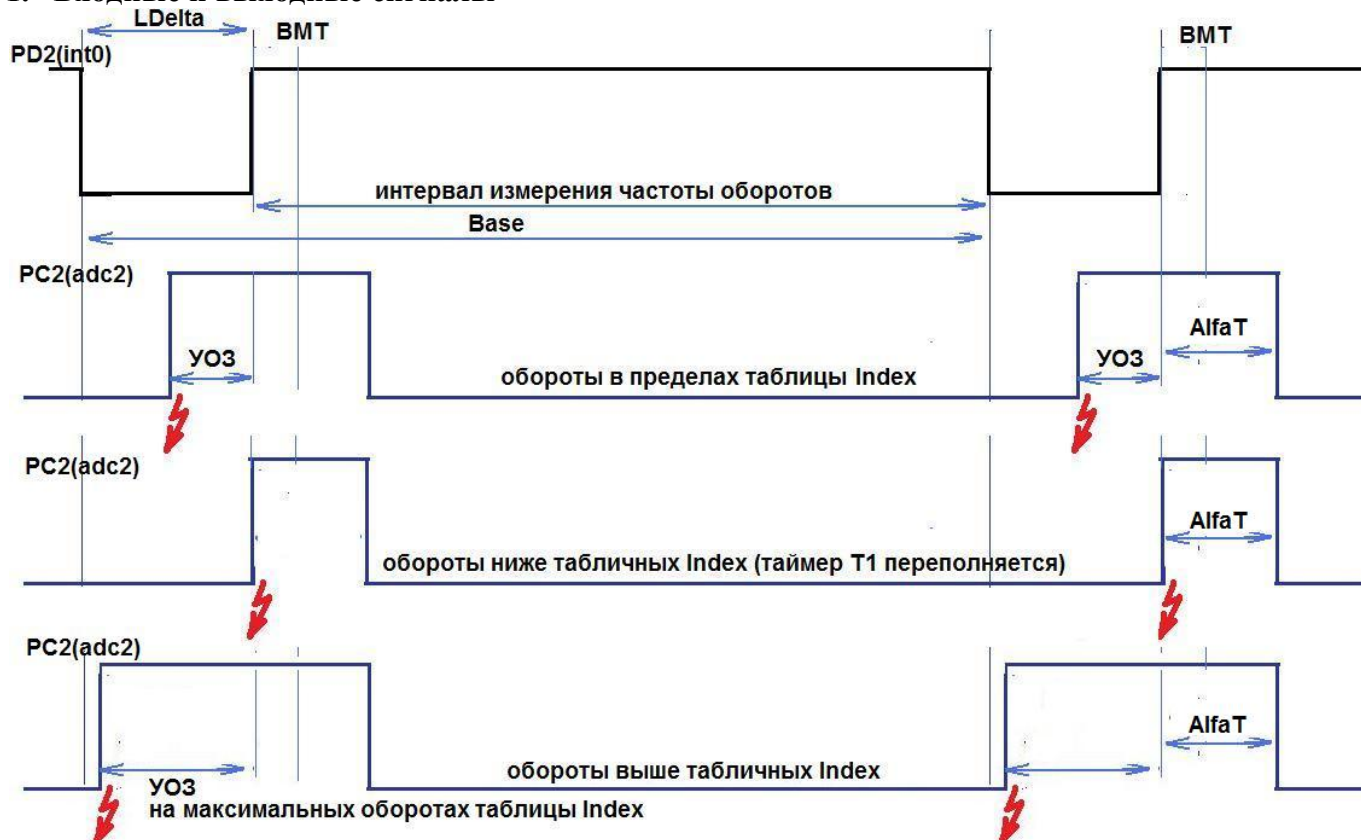


Рис.1

Я думаю комментарии тут не нужны.

2. Процесс заполнения «Главная разметочная таблица.ods» в приложение OpenOffice.

Внимание! Заполняются только зеленые поля, остальные поля служебные.

Поля, выделенные голубым цветом или названия которых выделены этим цветом подлежат переносу в прошивку.

Итак, исходя из графической схемы в пункте 1 необходимо ввести параметры Base и угловой размер метки в градусах LDelta. Теперь он **может быть равен не целому значению градусов!** Главная разметочная таблица автоматически разделит целую и дробную части и заполнит соответствующие ячейки.

У созданной системы есть физическое ограничение на максимальный YO3 в пределах LDelta, связанная с временными затратами на вычисления. В прошивке есть защита на случай превышения времени расчета времени достижения порога срабатывания таймера с целью обеспечения времени задержки искрообразования. Экспериментально определено, что время эквивалентно ~3 градусам! На это можно ориентироваться при проектировании. Не разумно задавать максимальный YO3 выше, чем LDelta-3 градуса. **За это отвечает ячейка C45.**

Далее необходимо задать верхний предел диапазона оборотов вращения ДВС. Пусть будет 10000. Нижний предел определяется разрядностью измерительного таймера T1 и частотой, поступающих на него импульсов. Далее следует заполнить 30 значений оборотов и код частоты Fn. В этом вопросе есть нюансы.

Первое, как расставлять значения? Не надо забывать, что между узловыми точками осуществляется алгоритм линейной аппроксимации, это означает, что если есть участок функции YO3(Обороты) с линейной зависимостью, то его можно обозначить крайними значениями. Напротив, где есть нелинейные участки можно поставить больше точек.

Второе, 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. Это приведет к появлению ступенчатого шага

изменения ОУЗ(оборотов) на высоких их значениях. Для контроля этой ситуации в Главной разметочной таблице **введен столбец Е**. Как правило на высоких оборотах уже нет потребности в резком изменении УОЗ, **и считаю допустимым, если в итоге колонка Е будет показывать «Ступеньки» на паре верхних диапазоне частот вращения двигателя**. Однако следует не допускать появления этого сообщения в на средних и малы оборотах, которое может появиться, если попросить очень крупный шаг по оборотам.

После этого надо заполнить столбец N, где находится базовая зависимость графика УОЗ(обороты). Эта зависимость используется как «по умолчанию», и храниться в программном коде. Рабочие таблицы формируются из неё и хранятся во флэш памяти контроллера. Об это позже.

Главное на помнить, что сдвиг по углу не относительно ВМТ, а по отношению к концу импульса с датчика, а реальный угол отличается в большую сторону на константу во всем диапазоне частот вращения.

Вот так заполнено в примере.

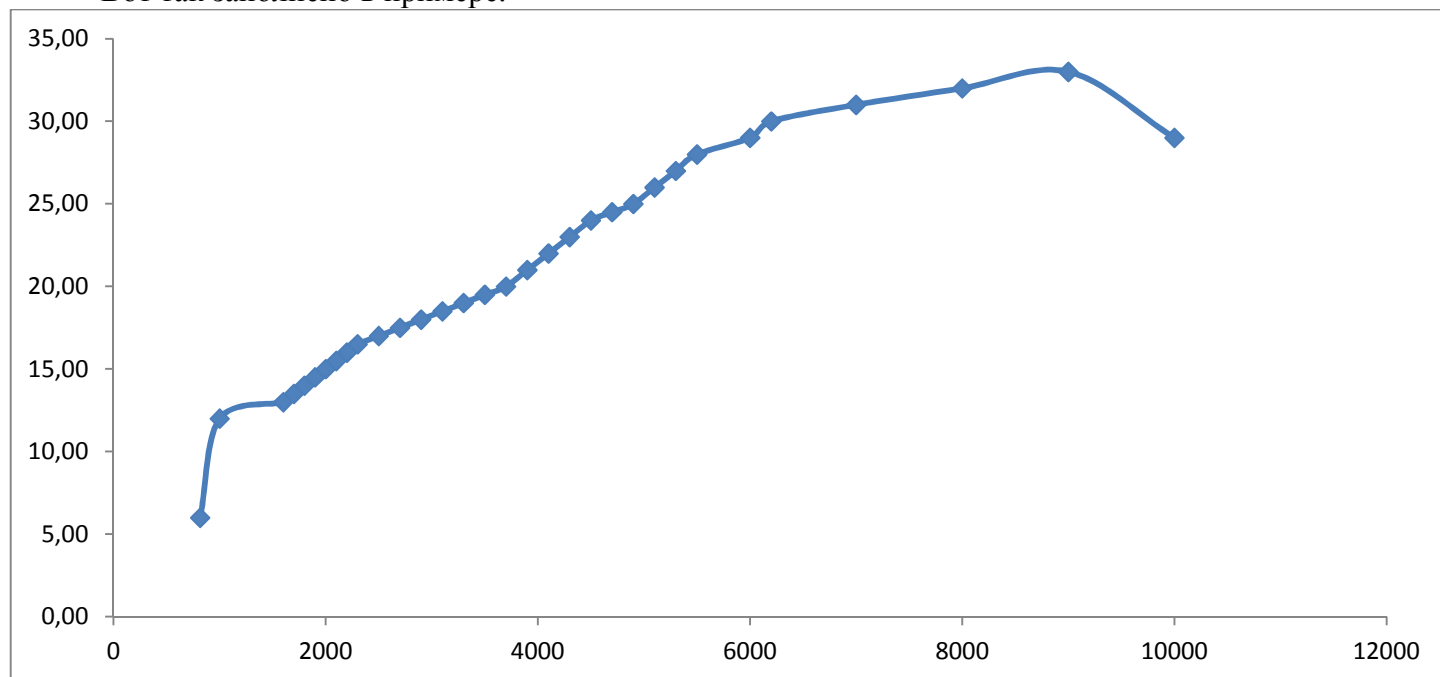


Рис.2

На мой взгляд, в данном примере не разумно заполнена таблица в районе 2000 оборотов. Точки «прилеплены» друг к другу. Зависимость там явно линейная, можно было только в крайних точках задать, а между ними прошивка рассчитает по линейной зависимости. Возможно автор предполагал, что в дальнейшем именно тут будет основанная нелинейность характеристики. Сама форма характеристики мной не обсуждается.

По вопросу задания характеристики УОЗ(обороты) осталось добавить только то, что на оборотах ниже нижней границы (816 оборотов/мин) сигнал зажигания будет вырабатываться по фронту сигнала LDelta. При оборотах выше предельного значения (10000 оборотов/мин) угол опережения зажигания будет оставаться константой, равной последнему табличному значению, в нашем случае - 29 градусов.

Следующий параметр – полная остановка двигателя (Режим СТОП). Алгоритм прошивки устанавливает режим СТОП в случае если нет изменения на входной линии системы в течении некоторого времени. Это время следует **установить в ячейку С49**.

Следующий параметр для настройки AlfaT - угловой размер временного интервала отключения выходного сигнала за меткой LDELTA в градусах. Дело в том, что длина импульса управления изменяется в зависимости от скорости вращения двигателя и заданного угла опережения зажигания, однако прекращается сигнал через AlfaT градусов после завершения метки LDELTA. Этот параметр следует выбирать для обеспечения беспрепятственного заряда емкости CDI зажигания. В TCI зажигании этот параметр может выбираться из условия баланса заряда катушки зажигания и её разогрева.

Последними параметрами для заполнения являются параметры протокола защиты входной линии от импульсных помех. Алгоритм работы защитного протокола подразумевает процесс повторного чтения входной линии при её изменении. Ниже приведена иллюстрация этого процесса.

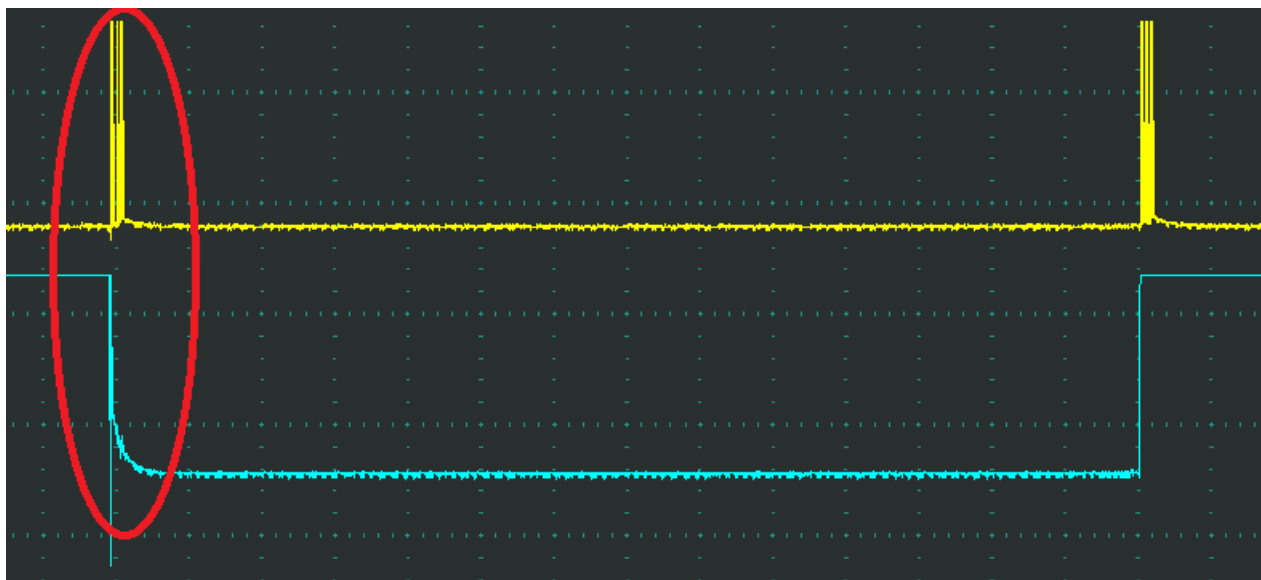


Рис.3

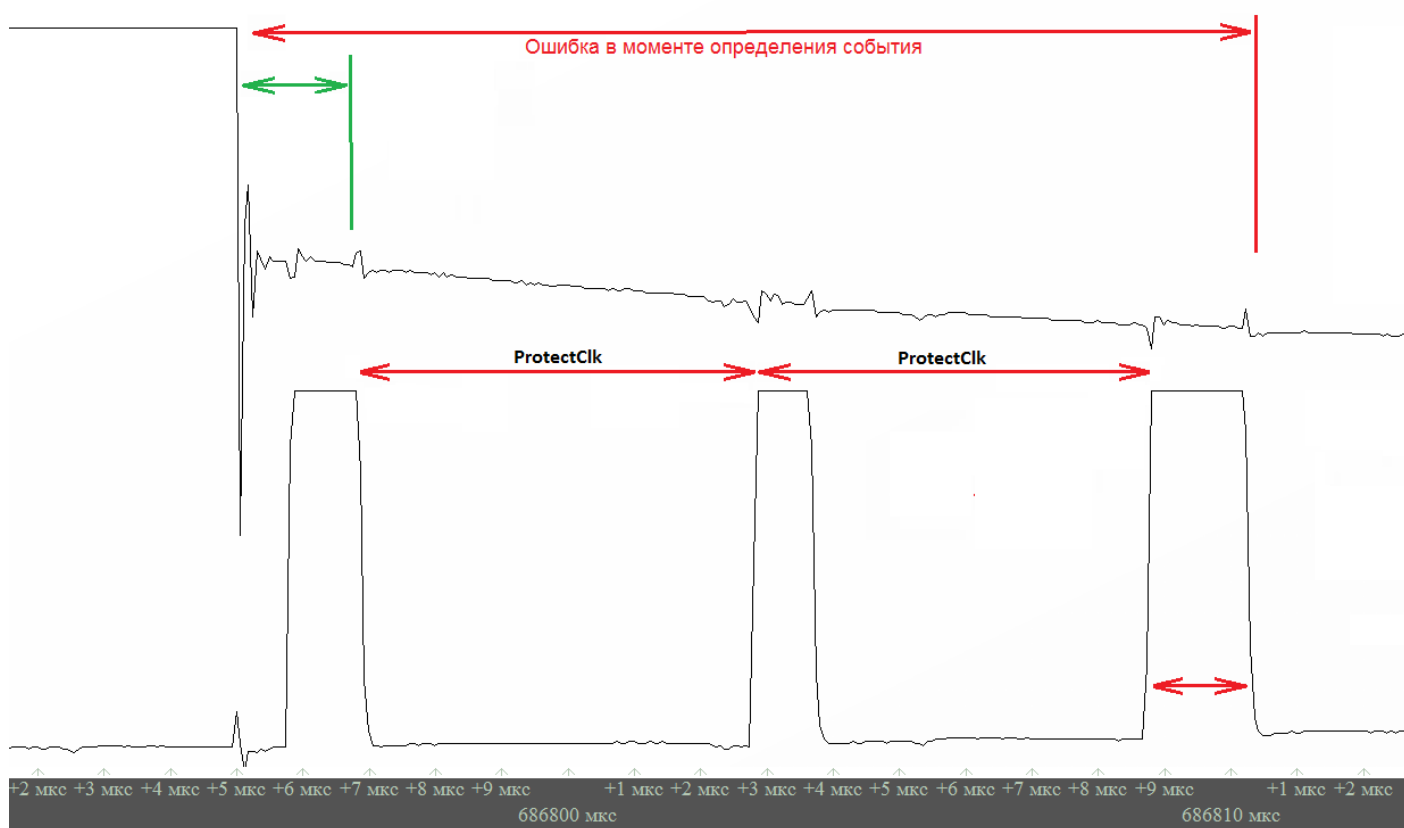


Рис.4

Параметр ProtectClk определяет период повторного чтения состояния входной линии. **Ячейка C53 определяет количество повторных чтений (после первого изменения) протоколом защиты.** Это чтение будет продолжаться до тех пор, пока алгоритм не получит устойчивые значения уровня входной линии на выборке заданной длины. В лучшем случае это будет 3 значения ($C53=2$). Одно значение при изменении и два повторных чтения. В случае неизменения уровня входной линии (импульсная помеха), алгоритм не произведет ни каких действий.

Эти два параметра (ProtectClk и C53) тоже есть компромисс между «хочу лучше защищать входную линию» и увеличением ошибки в моменте определения события. Пара значений 6 и 2 уже протестирована на реальном одноцилиндровом двигателе скутера. **Пусть это будет пара «по умолчанию».**

Отличительной особенностью построенного протокола защиты является возможность компенсации потерянного времени между изменением входного сигнала и моментом принятия решения алгоритмом о происходящем процессе. То есть «Ошибку в моменте определения события» удалось компенсировать путем к коррекции порога OCR1 таймера. Опустим вычисления и обработку экспериментальных данных. Итог приведен в параметре ProtectCorr.

Понятно, что в случае присутствия сильных помех протокол защиты будет вынужден делать лишние шаги и корректирующее слагаемое ProtectCorr не будет полностью компенсировать временные потери. В этом случае следует экспериментально подобрать пару ProtectClk и C53, подавляющие характерные помехи и вновь получить другую оценку коррекции.

На этом процесс заполнения Главной разметочной таблицы завершен.

Немного о случае, когда такой коррекции нет. Рассмотрим классическую схему построения таких систем. Нет повторных чтений. Есть только обработка прерывания при изменении входной линии. **Это время не нулевое (~7 тактов кварцевого генератора контроллера).** Есть еще временные потери, связанные с чтением значения измерительного таймера и его перезапуска. В самом благоприятном случае (корректное построение кода программы) и при кварце 16 МГц суммарные временные потери составляют **до 3 мкс**, а при кварце 4 МГц – **12 мкс**. На рис.4 этот случай представлен зелеными стрелками. Алгоритм с протоколом защиты дает время **~ 18 мкс**. Ниже представлены результаты некомпенсации этих временных потерь.

Обороты 1/мин	Базовая таблица, град	Эффективная таблица, град		
		Обработка прерываний		Протокол защиты
		16 МГц	4 МГц	16 МГц
816	6	5,99	5,93	5,91
1000	12	11,98	11,92	11,89
1600	13	12,97	12,87	12,83
1700	13,5	13,47	13,36	13,32
1800	14	13,97	13,85	13,81
1900	14,5	14,47	14,35	14,29
2000	15	14,96	14,84	14,78
2100	15,5	15,46	15,33	15,27
2200	16	15,96	15,82	15,76
2300	16,5	16,46	16,31	16,25
2500	17	16,96	16,80	16,73
2700	17,5	17,45	17,28	17,21
2900	18	17,95	17,77	17,69
3100	18,5	18,44	18,25	18,17
3300	19	18,94	18,73	18,64
3500	19,5	19,44	19,22	19,12
3700	20	19,93	19,70	19,60
3900	21	20,93	20,69	20,58
4100	22	21,93	21,67	21,56
4300	23	22,92	22,65	22,54
4500	24	23,92	23,64	23,51
4700	24,5	24,42	24,12	23,99
4900	25	24,91	24,60	24,47
5100	26	25,91	25,59	25,45
5300	27	26,90	26,57	26,43
5500	28	27,90	27,56	27,41
6000	29	28,89	28,52	28,35
6200	30	29,89	29,50	29,33
7000	31	30,87	30,43	30,24
8000	32	31,86	31,35	31,14
9000	33	32,84	32,27	32,03
10000	29	28,82	28,19	27,92

Отклонение эффективного угла опережения зажигания бесспорно можно просто учесть при задании исходной таблицы.

3. Перенос данных в исходник прошивки. Компиляция и прошивка.

Исходник прошивки состоит из трех файлов: «m328pdef.inc» - файл, определяющий константы микроконтроллера, «main.asm» - основная часть прошивки, «eeprom&uart.inc» - набор функций для работы с флэш памятью и интерфейсом UART.

Для это цели требуется текстовый редактор самый простой Блокнот. Офисные приложения и WordPad не годятся. Открываем на редактирование файл main.asm.

Смотрим в текст программы и вносим в нужной поля константы из Главной разметочной таблицы:

```
.equ LDelta      =      39
.equ LDeltaFP16  =      0
.equ Base        =     360
.equ Fn          =       7
```

Помните, что выходы Ардуино D6 (OCA=PD6=AIN0) и D5 (T1=PD5=OCOB) должны быть соединены!

```
.equ Ka          =     16
```

Обращаю Ваше внимание на то, что при некоторых значениях этого коэффициента можно сократить время расчета в программе. В тексте прошивки есть описание этого процесса в процедуре расчета задержки искрообразования и программирование "сравнителя" таймера StartIgnition.

```
.equ OUZMaxFP8   =     249
.equ OUZMinFP8   =     19
```

Далее устанавливаем количество рабочих страниц:

```
.equ UOZCount    =      4          (1, 2, 4, 8, 16, 32)
```

Таймаут режима СТОП -

```
.equ StopCntMax  =     61
```

Константы процедуры защиты от пульсаций -

Интервал дискретизации протокола защиты от пульсаций в мкс:

```
.equ ProtectClk  =      6          ;
```

Маска-длина защитного протокола формата:

```
.equ Maskchain   = 0b00000111
```

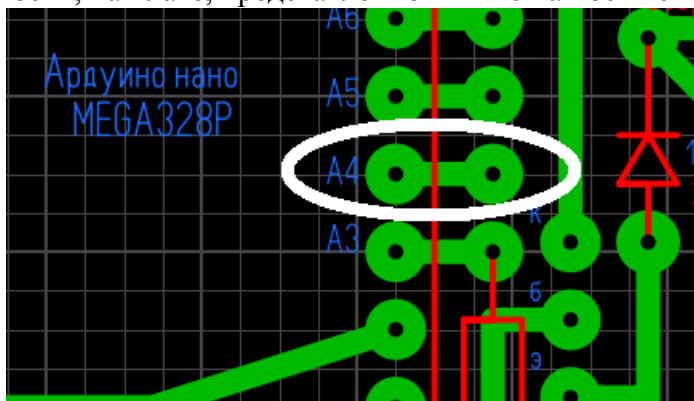
Корректирующая константа к сравнителю для компенсации времени работы алгоритма защиты.

```
.equ ProtectCorr =     19          ;
```

Следующий шаг, настройка входа АЦП для датчика ДПЗД.

.equ OUZCanBit = 4 - Номер канала мультиплексора АЦП, к которому подключен переменный резистор ДПЗД на землю. Этот же вход должен быть подключен через резистор 500 Ом к +5В Arduino. В этом случае при обрыве в цепи ДПЗД или его физического отсутствия алгоритм будет выбирать таблицу с максимальным номером из имеющихся.

В частности, на плате, представленной выше на пост используется этот вход.



В целях борьбы с помехами отсчеты АЦП помещаются в кольцевой буфер длиной
.equ OUZAverCount= 8

Основная программа производит усреднение по кольцевому буферу и в зависимости от числа страниц выбирает её номер как доля напряжения на входе АЦП по отношению 5В линейно. ДПЗД, по идеи, тоже должен осуществлять линейное преобразования угла поворота в его сопротивление. Чем длиннее этот буфер, тем лучше защита от помехи, но больше постоянная времени контура управления – инерция.

Последний шаг – это импорт столбцов M, R, V в конец фала прошивки. Там находятся процедуры FillIndex, FillTable и FillTimeMin, в которые необходимо их вставить.

Процесс подгонки прошивки завершён.

Компиляция и прошивка, а также запуск управляющей программы может быть осуществлен только под системой Windows. Удобно делать в консоли или файловом менеджере Far.

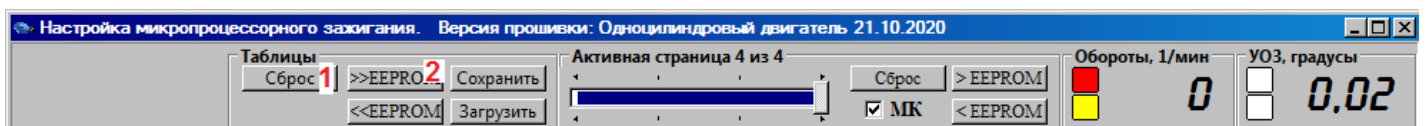
Для компиляции создан командный файл C.bat – исполняем его и если руки не кривые, образуется файл main.hex – собственно прошивка.

Подключаем Ардуинку к USB порту и образованный файл main.hex записываем в Ардуинку с помощью командного файла W.bat. **Важно предварительно исправить в командном файле номер COM порта Windows, к которому подключена Ардуинка, кроме того перед прошивкой не забудьте установить драйвер используемого моста USB-UART. В моем примере это был CH340G.**

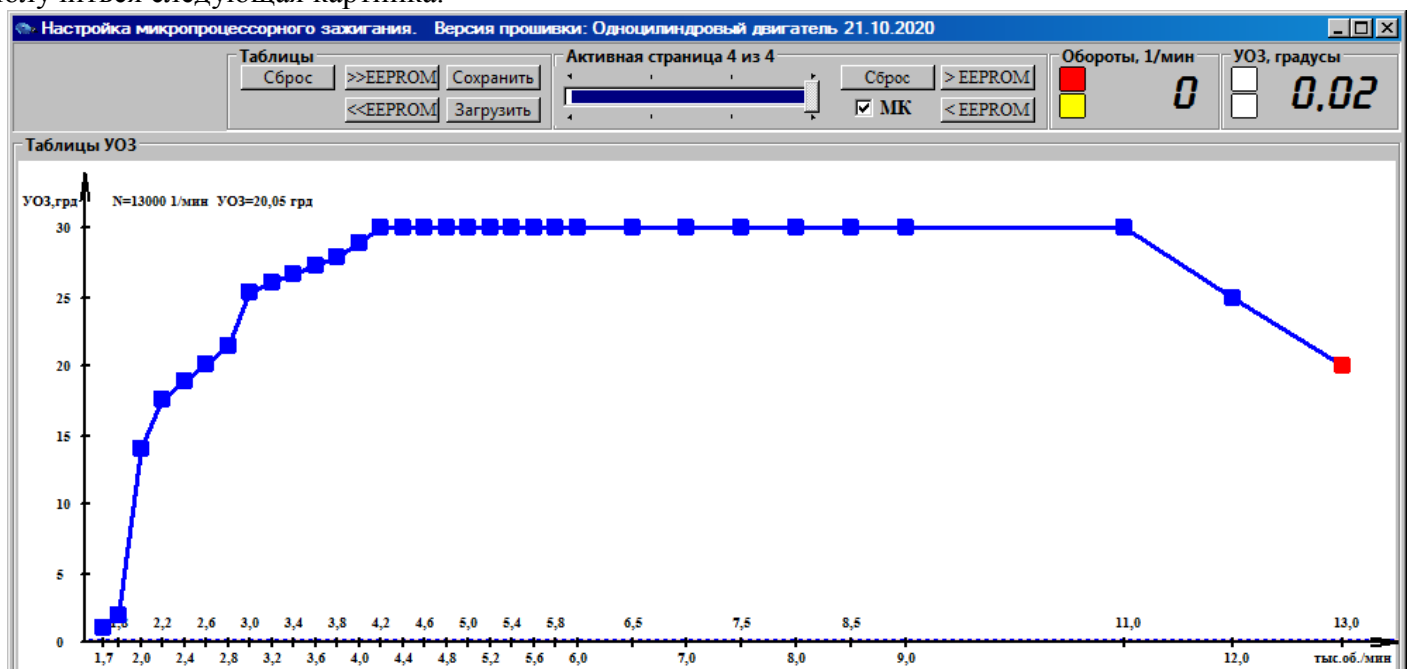
При реализации записи следует помнить, что некоторые мосты USB-UART не имеют прямого прохождения сброса на микроконтроллер на Ардуинке. Так на моем варианте этого нет и приходится сразу после запуска BAT файла на плате нажать кнопку Reset. В этой модели моста нет аппаратной линии сброса, а конденсатор на плате Ардуино, через который проходит этот импульс, заземлен для помехозащитности. На платформах NANO, UNO все это делать не надо!

4. Первый запуск и функционирование.

Настало время первого запуска. На самом деле он уже произошел после прошивки. Тот же интерфейс, который использовался для прошивки будем использовать для настройки системы в целом. Запускаем приложение UOZ.exe нажимаем последовательно две кнопки –



Тем самым мы заполняем все таблицы значениями «по умолчанию» и сохраняем их в флэш МК. При следующем запуске этого делать уже не надо и таблицы автоматически считываются от туда. В итоге получится следующая картинка.



Система готова к работе. Все таблицы одинаковые. Активная таблица 4.

Кнопки из секции «Таблица» относятся ко всем таблицам сразу. Совокупность таблиц можно: установить «по умолчанию», записать и прочитать из флэш, записать и считать с диска. На диске таблицы хранятся в файлах с расширением «u*», где * – число таблиц в наборе. Каждые 2 минуты оболочка автоматически сохраняет таблицы на диск в файле «autosave.u*». При нормальном завершении работы на диск дополнительно сохраняется еще одна копия таблиц в файле «exitsave.u*».

Кнопки из секции «Активная страница» относятся к одной странице. При включенном переключателе «МК» активная страница назначается МК. Если эту деактивировать этот переключатель, то выбор активной таблицы осуществляется ползунковым регулятором, при этом на Ардуинке загорится светодиод. Активную страницу можно сбросить «по умолчанию», записать и считать из флэш. Самое главное, её можно по точечно корректировать визуально, даже при работающем двигателе.

Для этого мышкой выбирается нужный узел и далее или колесиком мышки или просто взявшись за узел следует потащить его в нужную сторону.

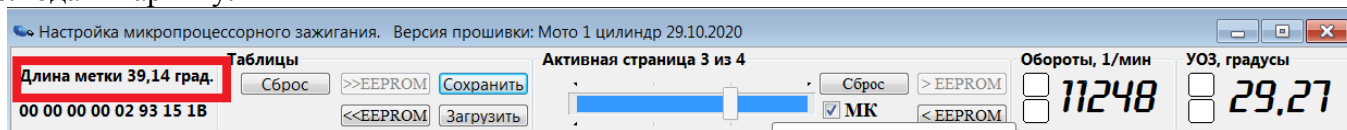
Осталось сказать о том, что все изменения таблиц осуществляются в ОЗУ контроллера. Если в конце сеанса правки не сохранить на флэш, то все ваши изменения будут утрачены.

Программа UOZ теперь имеет и сервисный режим. В этом режиме можно наблюдать за восемью переменными tester[0..7] из прошивки. Отмечу, что старшие четыре байта всегда заняты сервисной информацией о длине интервала LDelta.

В любом месте прошивки можно запомнить любой регистр в ячейки памяти tester в следующем формате:

```
sts    tester,A
sts    tester+1,B
sts    tester+2,r0
sts    tester+3,C
```

Далее программу OUZ следует запустить с параметром tester. На экране тогда можно будет наблюдать картину:



В левой части формы будет видна «оценка» длины метки LDELTA и ниже 8 ячеек памяти. Оценка длины метки не зависит от настроек прошивки (за исключением частоты Fn) и **её можно далее использовать для заполнения Главной разметочной таблицы.**