```
//Тактирование ФУОЗ
#define ChipFrequency 16   //Частота микроконтроллера в МГц
#define Fn             9   //Частота алгоритма ФУОЗ в МГц  = ChipFrequency/(1+Fn)/2 . Определяет величину rpm[M].
#define ASys        -10.0  //град. угол поворота маховика, на котором выполняются системные функции ФУОЗ (АЦП)

//Входные линии D2(int0) и D3(int1).
#define InlinesMode 0      //Режим работы входных линий
                           //0 - Используется только вход D2(int0). События (Events) маскируются 0b00000011
                           //1 - Используются два входа D2(int0) и D3(int1). События (Events) маскируются 0b00110011
                           //2 - Используются два входа D2(int0) и D3(int1). События (Events) не маскируются
#define RepeatCnt   2      //Количество повторных чтений, 1..7
#define ProtectTime 6      //Интервал времени между повторными чтениями входных линий, 5..12 мкс
                           //Константа TimerCorrect дает коррекцию расчетного времени появления искры.
#define Beta       40      //Начало отсчета (угловое расстояние установки датчика от ВМТ) Beta > УОЗmax!!!
#define Q          1       //Кратность применения ФУОЗ на обороте маховика (количество цилиндров)

/*
Виртуализация входных линий. Моментами изменения состояния ФУОЗ являются моменты изменения некоего виртуального
импульса Delta, который "формируется" в известных угловых координатах от ВМТ цилиндра или цилиндров - начало отсчета.
Этот импульс может возникать и два раза за оборот (параметр Q), тем самым обеспечивая расчет и формирования искры в
случае двухцилиндрового варианта с раздельным процессом искрообразования. Виртуальный импульс Delta "формируется"
программой на базе анализа состояния входных линий. В случае если входная линия одна, виртуальный импульс совпадает с
физическим */
#define Delta      30.0    //Угловой размер метки (длина сигнала датчика)

/*
События (Events) имеют формат  - 0bBBBBAAAA   BBBB - 4 момента времени линии Int1  AAAA - 4 момента времени линии Int0
в моменты их совместного изменения. В силу того, что в этом скетче используется режим InlinesMode 0, далее Events
автоматически укорочен до 2 бит и автоматически предопределены константы D2Down и D2Up. Виртуальный импульс Delta
совпадает с физическим на входе D2 */
#define Event0  D2Down     //Задать код события 0
#define Action0 StartDelta //Задать название функции, соответствующее коду события 0
```

```
#define Event1  D2Up         //Задать код события 1
#define Action1 StopDelta    //Задать название функции, соотвествующее коду события 1

//Выходы
#define OutLine      14     //Импульс зажигания (0-1-0). Выходная линия  7-19
#define OutPulseMode 1      //Режим расчета длины импульса зажигания на линии OutLine
/*
0 - Расчет длины импульса в микросекундах от момента зажигания. PulseLength указывается в мкс.
1 - Расчет длины импульса в градусах от момента зажигания. PulseLength указывается в градусах.
2 - Расчет длины импульса на указанный в PulseLength угол поворота маховика. PulseLength указывается в градусах. */
#define PulseLength  120    //Длина (угол) импульса зажигания на линии OutLine
#define OutLineStopInv      //Уровень выходной линии в режиме Останова (Selectr = M+1)

//Сетка оборотов
const double C = 60000000/Q*ChipFrequency/(1+Fn)/2; //Константа временного масштаба ФУОЗ.
#define  M 8                                        //Количество узловых точек на зависимости УОЗ(обороты)
/*
Сетка оборотов rpm[M-1]..rpm[1] задаются пользователем. Два нижних предела определяется разрядностью вычислений -
используется 16 разрядный таймер с расширением разрядности до 24. При параметрах Fn=9 и ChipFrequency=16 МГц ->
rpm[M+1] = round(C/0x80000) = 91 мин-1, rpm[M] = round(C/0xFFFF) = 732 мин-1 */
    rpm[M+1] <= rpm[M] <= ....<= rpm[7] <= rpm[6] <= rpm[5] <= rpm[4] <= rpm[3] <= rpm2] <= rpm[1] <
Selectr M+1  |  M      |      |  7      |  6      |  5      |  4      |  3      |  2     |  1      |  0
                732             1000      2000      4000      6000      8000      10000    16000            */
volatile byte Selectr = M+1;                        //Номер диапазона оборотов
/*
Selectr= M+1    - Двигатель остановлен
Selectr= M      - Двигатель вращается медленно. УОЗ = ALowMode. Режим "Низкие обороты", он же "Ниже табличных")
Selectr= M-1..1 - УОЗ задан таблицей и рассчитывается прошивкой. Режим "Нормальные обороты", он же "Табличные обороты")
Selectr= 0      - Двигатель вращается быстро. УОЗ = AHighMode. Режим "Высокие обороты", он же "Выше табличных")

Программа работает вместо "оборотов" работает с "индексами" index. Внимание! Большему номеру соответствуют меньшие
обороты  */
```

```
volatile const word index[M] =
{round(C/16000),round(C/10000),round(C/8000),round(C/6000),round(C/4000),round(C/2000),round(C/1000),0xFFFF};
//index[0]= round(C/rpm[1]) index[1]= round(C/rpm[2]) index[2]= round(C/rpm[3])...

//Таблицы УОЗ
#define TableCount 4                      //Количество страниц УОЗ
/*
Если есть необходимость переключать таблицы, то организуем буферизированный режим работа АЦП для управления номером
активной таблицы УОЗ   */
#if (TableCount > 1)
  #define  ADCMux 7                       //Канал АЦП для управления таблицами
  #define  ADCDefaultValue 0xFF           //Начальное значение ячеек буфера АЦП
  #define  ADCBuffCount 16                //Длина буфера АЦП
  volatile byte ADCBuff[ADCBuffCount];    //Буфер АЦП
  volatile byte ADCBuffptr = 0;           //Указатель на номер ячейки для записи
  volatile bool AllowAverag = false;      //Разрешить однократное усреднение буфера АЦП
#endif

/*
Константы для расчета задержки ФУОЗ
         ........  732    1000   2000   4000   6000   8000   10000  16000  ........
Обороты  ........  rpm[8] rpm[7] rpm[6] rpm[5] rpm[4] rpm[3] rpm[2] rpm[1] ........
УОЗ      ALowMode  A[8]___A[7]___A[6]___A[5]___A[4]___A[3]___A[2]___A[1]   AHighMode     */
#define ALowMode  -5.0   //град. на всех оборотах rpm <= rpm8
#define AHighMode 15.0   //град. на всех оборотах rpm1 < rpm
/*
Az[][0] =round(1.0*32768*(Beta-AHighMode)*Q/360)
Az[][k] =round(1.0*32768*(Beta-(A[k+1]*index[k]-A[k]*index[k-1])/(index[k]-index[k-1]))*Q/360)  k=1..M-1
Bz[][0] =-TimerCorrect
Bz[][k] =round(1.0*index[k]*index[k-1]/(index[k]-index[k-1])*(A[k+1]-A[k])*Q/360)-TimerCorrect  k=1..M-1

Для M=8 таблица Обороты - УОЗ и расчеты по ним сведены в файле "таблица M=8.ods"      */
```

```
volatile const int Az[][M] ={{2276,-2579,455,455,4005,3823,3641,3186},{2276,-2579,455,455,4005,3823,3641,3186},{2276,-2579,455,455,4005,3823,3641,3186},{2276,-2579,455,455,4005,3823,3641,3186}};
volatile const int Bz[][M] ={{-17,427,-17,-17,-884,-817,-684,-17},{-17,427,-17,-17,-884,-817,-684,-17},{-17,427,-17,-17,-884,-817,-684,-17},{-17,427,-17,-17,-884,-817,-684,-17}};

#include "fuoz.h"          //Файл с вспомогательными #define
```