



**Программный комплекс  
автоматизации экспериментальных  
и технологических установок "ACTest<sup>©</sup>"**

**Модуль  
"ACTest<sup>©</sup> - Control"**

Руководство пользователя  
Версия 1.14





## Содержание

НАЗНАЧЕНИЕ.....	4
РАБОТА С РЕДАКТОРОМ СКРИПТОВ.....	4
ПРАВИЛА НАПИСАНИЕ СКРИПТА .....	5
ФУНКЦИИ ДЛЯ УПРАВЛЕНИЯ ЭКСПЕРИМЕНТОМ.....	6
ПРИМЕР НАПИСАНИЯ СКРИПТА .....	8

## Назначение

**ACTest<sup>©</sup>-control** - это расширение комплекса ACTest, которое позволяет проводить управляемый эксперимент по заранее заданной циклограмме. Циклограмма описывается в виде скрипта.

Скрипт – это небольшая программа, состоящая из последовательности операторов, которые выполняются интерпретатором скриптов и изменяют алгоритм работы подсистемы сбора данных.

С помощью данного расширения можно производить следующие действия:

1. управление проведением эксперимента (запуск/останов сбора данных, закрытие эксперимента);
2. управление визуализацией (показать определённый экран визуализации);
3. управление выдачей сигналов на цифровые выходы и ЦАП;
4. управление протоколированием данных;
5. интерпретатору скрипта доступны данные входных и расчётных каналов. Благодаря этому в скрипте есть возможность организации ветвления сценария по условию (доступны операторы цикла, проверки условия). Также есть возможность корректировать данные входных и расчётных каналов.

Скрипт не рассчитан на проведения ресурсоёмких вычислений, поэтому все математические операции предпочтительнее выполнять средствами математики ACTest<sup>©</sup>.

Время реакции системы на то или иное событие – один макрокадр. На время макрокадра также накладываются дополнительные условия - оно не должно быть меньше 0,3с.

## Работа с редактором скриптов

Скрипт можно писать на двух языках программирования: **PascalScript**, **C++Script**. Сохраняется он в папке Data как текстовый файл script.txt. Для удобства написания скрипта в модуль подготовки и проведения эксперимента встроен редактор скриптов (закладка "Скрипт").

Окно редактора скриптов поделено на три части (см. Рисунок 1). Слева находится дерево функций, предназначенных для управления экспериментом. Все функции объединены в группы. Для раскрытия списка функций достаточно кликнуть мышкой по знаку плюс, расположенного слева от названия группы. При клике мышкой по имени функции в нижнем окне появляется её описание.

Справа расположено окно редактирования. В нём пользователь может написать скрипт, который и будет управлять ходом эксперимента. Редактор оснащён подсветкой синтаксиса.

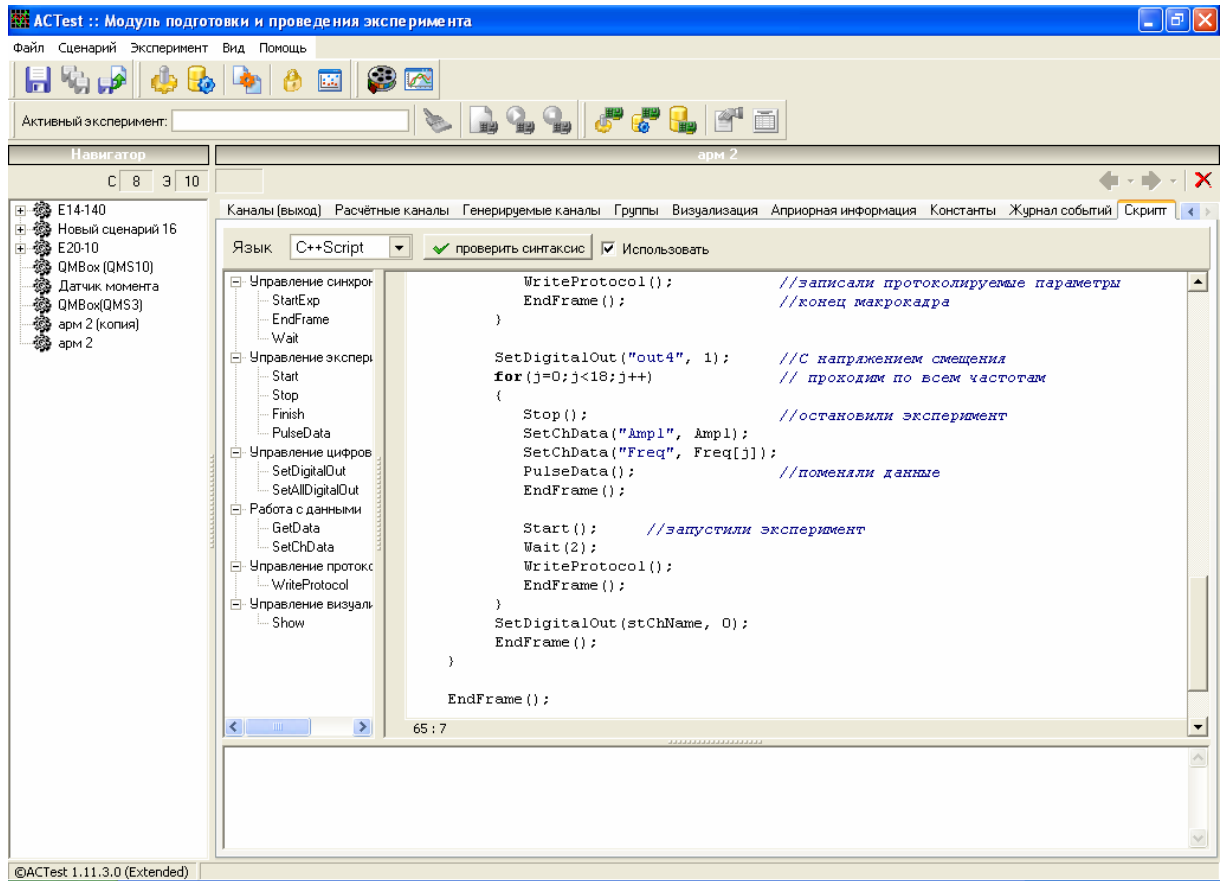


Рисунок 1. Закладка редактора скриптов

Внизу расположено информационное окно. Помимо того, что в него выводится описание функций, в него заносится ещё и результаты компиляции скрипта. Это либо сообщение о том, что скрипт успешно скомпилирован, либо описание ошибки. При возникновении ошибки в окне редактирования курсор появится именно в том месте, где она и возникла.

В верхней части окна находится панель инструментов. На ней есть выпадающий список с поддерживаемыми скриптовыми языками. Перед началом написания скрипта надо выбрать язык. Также на панели инструментов есть кнопка запуска компиляции для проверки синтаксиса скрипта. Для того чтобы использовать скрипт во время проведения эксперимента, на той же панели инструментов надо поставить флажок на элементе управления "Использовать".

Написанный в редакторе скрипт сохраняется в базе данных и привязывается к сценарию эксперимента.

## Правила написание скрипта

Скрипт представляет собой набор операторов, которые будут последовательно выполнены. Помимо специальных функций доступны и операторы обеспечивающие ветвление при выполнении скрипта: **if**, **for**, **while**.

Первой строкой в скрипте описывается используемый язык:

**#language C++Script**

или

**#language PascalScript.**

Далее могут следовать подключение других файлов, объявления переменных, констант и описание прототипов функций. После этого идёт описание главной исполняемой функции. В **C++Script** исполняемая функция отделяется фигурными скобками - {тело функции}. В **PascalScript** - операторами **begin** и **end**: begin тело функции end.

Первым оператором в теле главной функции должен быть оператор **StartExp()**. Он предназначен для синхронизации работы модуля регистрации и интерпретатора скриптов.

Единицей синхронизации работы модуля регистрации реального времени и интерпретатора скриптов является макрокадр. В скрипте последовательность операторов должна быть разбита на макрокадры. Т.е. блок операторов, которые должны будут выполняться за один макрокадр, должен оканчиваться оператором **End-Frame()** или **Wait()**. При этом надо учитывать, что их выполнение начнётся только с началом следующего макрокадра, с небольшой временной задержкой.

## Функции для управления экспериментом

- **void StartExp()**

Используется для синхронизации работы модуля регистрации реального времени с выполнением скрипта. С этой функции должна начинаться главная функция скрипта. В дальнейшем она не используется. Параметров нет.

- **void EndFrame()**

Функция предназначена для синхронизации работы модуля регистрации реального времени с выполнением скрипта. Применяется для разделения группы операторов, выполняемых за один макрокадр. Параметров у функции нет.

- **void Wait(int FrameCount)**

Функция предназначена для синхронизации работы модуля регистрации реального времени с выполнением скрипта. Применяется для реализации паузы в управлении. Время передаётся в качестве параметра функции и выражается в числе макрокадров.

Параметры:

**KadrCount** – число макрокадров.

- **void SetDigitalOut (LPTSTR ChannelName, int TtlOut)**

Функция предназначена для управления выдачей сигнала на заданный цифровой выход. После вызова функции значения цифровых выходов сохраняется. То есть, если на какой-то канал было выдано значение (1), а в следующий макрокадр на этом канале сигнала не должно быть, то значение на этом канале необходимо обнулить.

Параметры:

**ChannelName** – символьный массив с именем цифрового канала, на который надо выдать сигнал;

**TtlOut** – переменная задаёт состояние цифрового канала (0 – нет сигнала, 1- есть сигнал).

- **void SetAllDigitalOut(int Flag)**

Функция предназначена для управления выдачей сигнала на все цифровые выходы.

Параметры:

**Flag** - переменная задаёт состояние цифровых каналов (0 – нет сигнала, 1- есть сигнал).

- **void SetChData(LPTSTR ChannelName, float Value)**  
Функция предназначена для записи значения в канал с именем **ChannelName**. Если этот канал имеет несколько точек, то запись ведётся в первое значение.  
Параметры:  
**ChannelName** – символьный массив с именем канала, в который записывается значение;  
**Value** – записываемое значение.
  
- **void GetData(LPTSTR ChannelName, float &Buffer)**  
Функция предназначена для получения значения с канала с именем **ChannelName**. Если канал имеет несколько точек, то читается первое значение.  
Параметры:  
**ChannelName** - символьный массив с именем канала, в который записывается значение;  
**Buffer** – адрес переменной, в которую будет записано значение канала.
  
- **void WriteProtocolAll()**  
Функция предназначена для записи параметров всех групп протоколирования в файл протокола. Параметров нет.
  
- **void WriteProtocol (intNumberGroup)**  
Функция предназначена для записи параметров определенной группы протоколирования в файл протокола.  
Параметры:  
**NumberGroup** – номер нужной группы протоколирования.
  
- **void Stop()**  
Функция предназначена для остановки сбора данных. При вызове функции останавливается сама плата сбора данных. Параметров нет.
  
- **void Start()**  
Функция предназначена для запуска сбора данных. Параметров нет.
  
- **void PulseDate()**  
Функция предназначена для обновления данных входных и выходных каналов (например, АЦП или ЦАП). Параметров нет.
  
- **void Finish()**  
Функция предназначена для закрытия эксперимента. Параметров нет.
  
- **void Show(int Action, int NumberWindow)**  
Функция предназначена для показа нужного окна визуализации. При этом заданный элемент визуализации отображается по верх всех окон.  
Параметры:  
**Action** – должно принимать значение 1;  
**NumberWindow** – номер окна визуализации которое надо показать.

## Пример написания скрипта

В этом примере строится АЧХ нескольких каналов. На вход некоторого блока с ЦАП подаётся синус заданной частоты и амплитуды и затем измеряется выходной сигнал с блока. Получаемые данные протоколируются и затем в Excel строится график. К ЦАП коммутируется последовательно несколько каналов посредством переключения реле. Реле переключаются подачей на них сигнала с цифровых выходов.

```
#language C++Script //язык скрипта

int i; //здесь можно объявить глобальные переменные
int j;
float Freq[4]; //массив с частотами
float Ampl = 5; //амплитуда сигнала, выдаваемого с ЦАП

String stChName; //имя канала
{ //начало главной функции скрипта
    //заполнили массив частот
    Freq[0] = 2;
    Freq[1] = 3;
    Freq[2] = 4;
    Freq[3] = 5;
    StartExp(); //ждём начала эксперимента
    for(i=0;i<4;i++) //последовательновыставляем
        единички на цифровых выходах с первого по четвёртый
    {
        SetAllDigitalOut(0); //обнулили все цифровые выходы

        stChName = "out" + IntToStr(i); //формируем имя канала
        SetDigitalOut(stChName, 1); //выставляем единичку на
        нужный цифровой выход
        for(j=0;j<4;j++) //проходим по всем частотам
        {
            Stop(); //остановили сбор данных
            SetChData("Ampl", Ampl); //задали амплитуду выдаваемого сигнала
            SetChData("Freq", Freq[j]); //задали частоту выдаваемого
            сигнала
            PulseData(); //передали новые данные
            (сигнал) в буфер ЦАП
            EndFrame(); //конец макрокадра

            Start(); //запустили сбор данных
            Wait(2); //подождали 2 макрокадра
            WriteProtocolAll(); //записали протоколируемые
            параметры
            EndFrame(); //конец макрокадра
        }
        EndFrame();
        Finish(); //закрыли эксперимент
    }
}
```

В начале главной функции скрипта заполняется массив частот. Затем выполнение скрипта приостанавливается до запуска эксперимента. В цикле мы последовательно коммутируем к выходам ЦАП 4 канала. Во вложенном цикле последовательно выдаём с ЦАП сигнал разной частоты. Для этого сначала останавливается сбор данных, затем происходит передача сигнала в буфер ЦАП и запуск сбора данных. После запуска сбора данных выдерживается пауза в 2 макрокадра и происходит запись протоколируемых параметров в файл протокола. Как только проходят все циклы эксперимент завершается.