г. Ростов-на-Дону,
пер.Нахичеванский, 64
тел.: +7 (863) 227-18-52
info@smart-elec.ru
smart-elec.ru

Руководство по программированию контроллера AlphaSE Controller из ПО Alpha SE Configurator 4.7

г.Ростов-на-Дону
2020г.

Глава ***стр.***

О программе

AlphaSE Configurator является Windows приложением и служит для программирования контроллера AlphaSE Controller, загрузки обновлений и настройки её основных параметров. AlphaSE Controller позволяет управлять 127 модулями по протоколу ADNet+ и 247 модулями по протоколу ModBus RTU 9600 8N1. Рекомендуем ознакомиться с обучающими видео по установке, настройке и программированию AlphaSE Controller на нашем сайте.

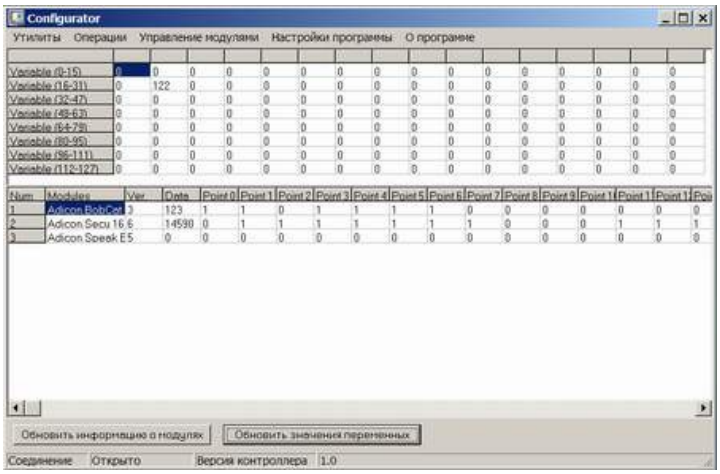


Рис.1

Установка и настройка программы

Программа AlphaSE Configurator не требует специальной установки. Запустите ее на компьютере, где уже запущена программа контроллера AlphaSE Controller.

Если программа запустилась без сообщений об ошибках, соединение с контроллером установлено успешно. Если при запуске программы появилось сообщение об ошибке, обратитесь к разделу «Возможные неисправности».

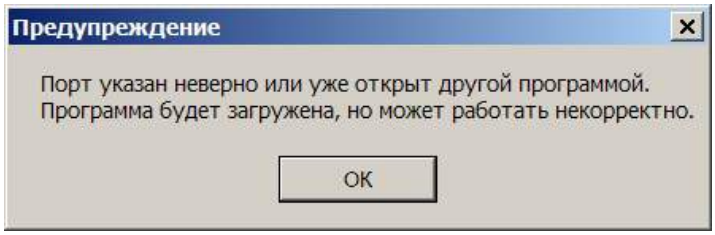


Рис. 2

Адресация модулей, работающих по протоколу ADNet+

Эта глава не относится к модулям, работающим по протоколу ModBus.

При построении новой системы автоматизации контроллер должен задать адреса всем модулям расширения, имеющим программную адресацию, и узнать адреса модулей, имеющих аппаратную адресацию.

Для выполнения данной операции необходимо проделать следующую последовательность действий:

1. Зайти в форму адресации *Утилиты ->Мастер адресации модулей... (см.*

рис. 3)

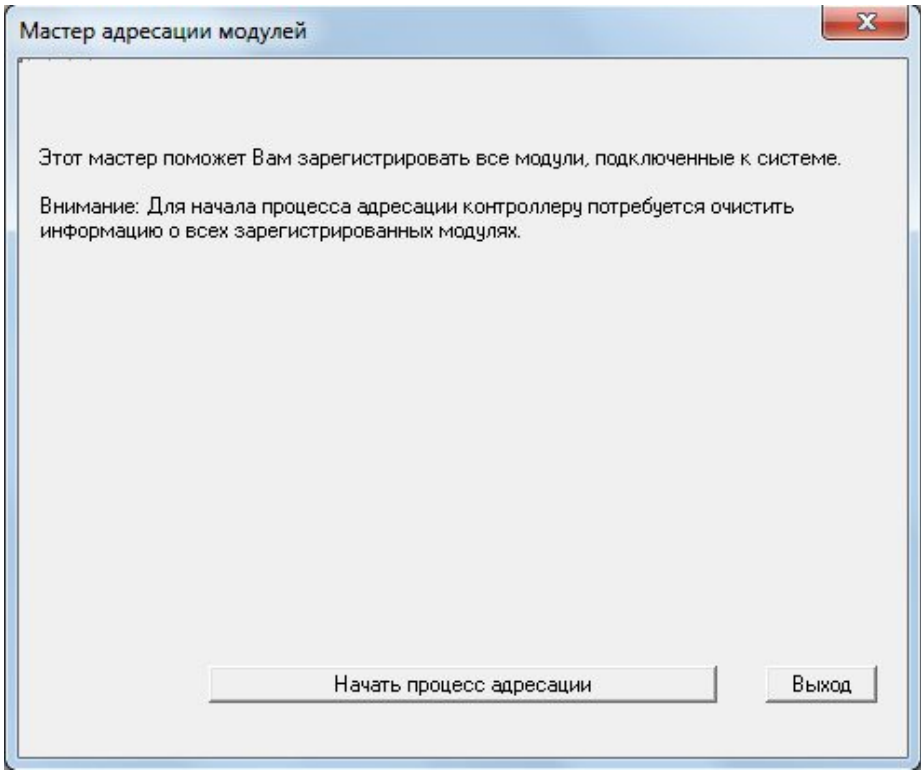


Рис.3

2. Нажать кнопку «Начать процесс адресации». Контроллер очистит у себя имеющуюся информацию об адресованных модулях.

3. Следовать инструкции приведенной на форме адресации.*(см. Рис. 4)*

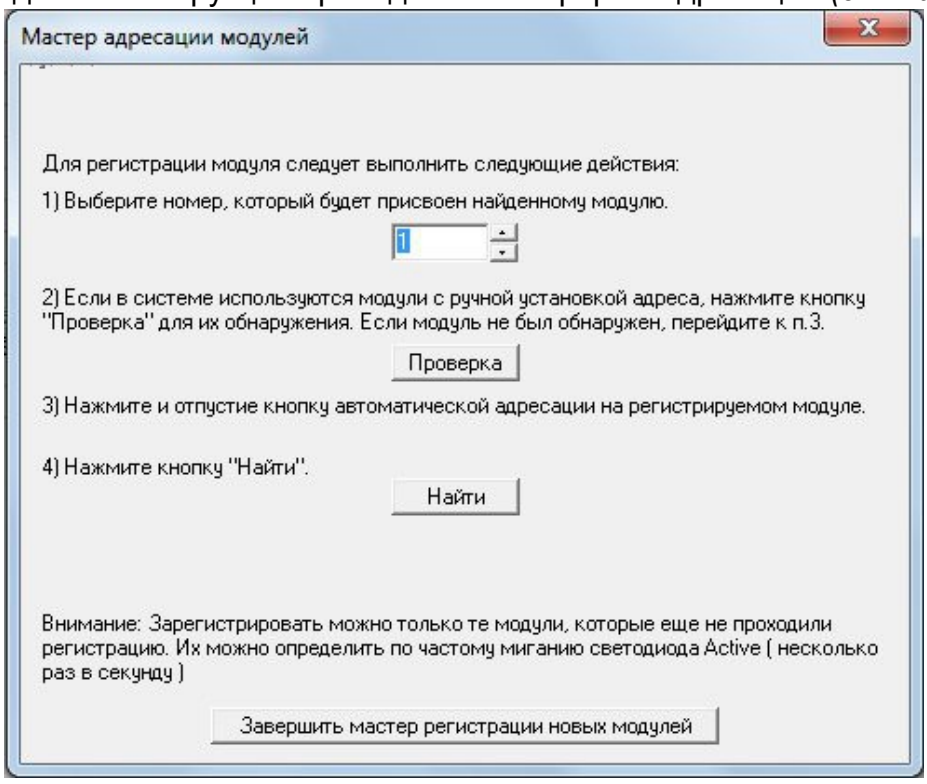


Рис.4

4. После адресации всех модулей нажать кнопку «Завершить мастер адресации модулей». Программа выдаст информацию по всем адресованным модулям системы с которыми сможет работать контроллер AlphaSE. *(см. Рис. 5)*

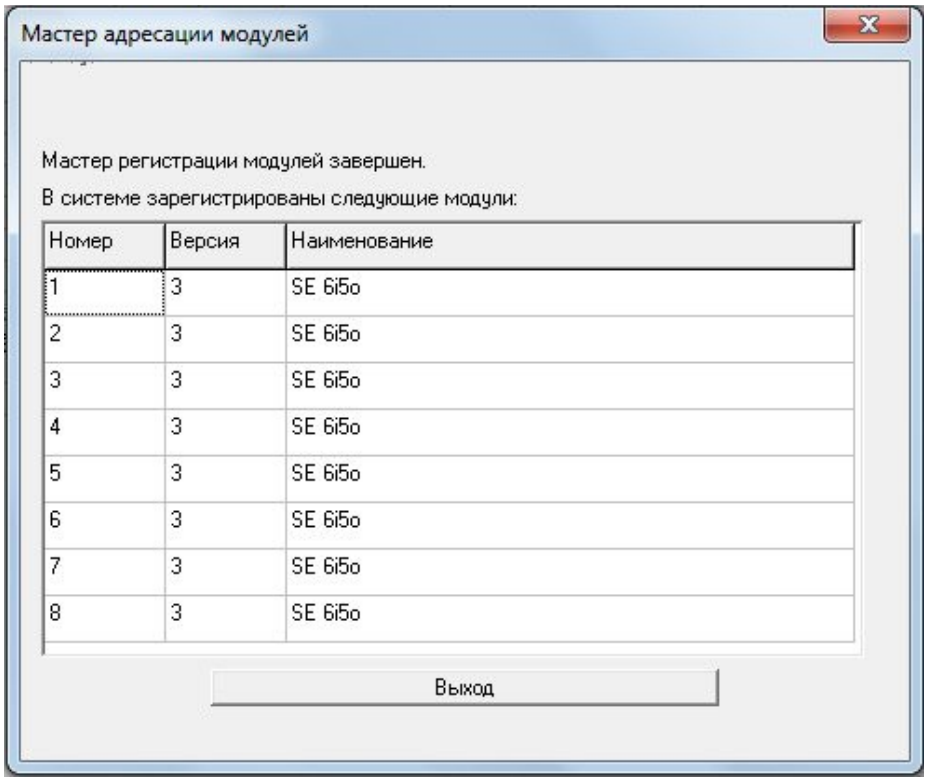


Рис.5

Частота опроса модулей, работающих по протоколу ADNet+

Эта глава не относится к модулям ModBus.

Скорость работы контроллера зависит от количества адресованных модулей и размера пользовательской программы. Если количество модулей расширения приближается к 10 шт, Вы можете начать ощущать задержку в работе контроллера. Например, между нажатием сценарной кнопки и выполнением какого-либо действия, указанного в пользовательской программе, задержка может составлять до 1 секунды. Этот эффект связан с ограниченной скоростью передачи данных между контроллером и модулями расширения.

Одним из способов уменьшения задержки является задание частоты опроса модулей в форме Операции *->Частота опроса модулей...(см. Рис. 6)*

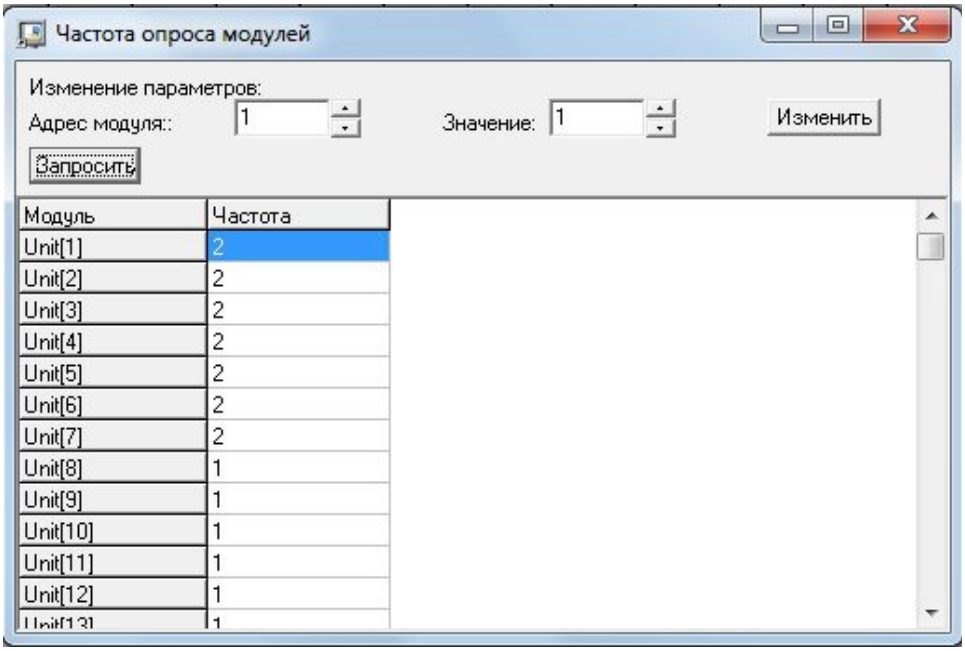


Рис.6

Если частота опроса модуля равна 1, модуль опрашивается каждый раз после выполнения пользовательской программы. Если частота опроса модуля равна 2, модуль опрашивается через раз. Чем больше значение, тем реже опрашивается модуль.

Задав модулям, не требующим быстрого отклика, большее значение можно в несколько раз увеличить скорость работы контроллера.

Программирование контроллера

Для программирования контроллера необходимо зайти в предназначенный для этого редактор *Утилиты ->Редактор программ (см. рис. 7),* открыть уже имеющийся программный файл или начать написание программы с чистого листа.

Рис.7

После написания программы ее следует сохранить и откомпилировать (преобразовать из вида понятного человеку в файл понятный контроллеру). Для

запуска процесса компиляции следует выбрать пункт *Программа -> Компилировать* или нажать *Ctrl+F9*. Если программный файл имеет ошибки, то в процессе компиляции они будут выявлены, пользователю будет выдано сообщение о соответствующей ошибке с указанием места ее расположения. Если программный файл написан без ошибок и процесс компиляции прошел успешно, пользователь получит сообщение о завершении компиляции. Скомпилированный файл будет создан в той же папке, что и исходный программный файл, но будет иметь расширение cpl.

После компиляции программы ее можно загрузить в контроллер. Только после этого контроллер будет действовать по новой программе, следуя всем ее инструкциям.

Для загрузки программы следует выполнить следующую последовательность действий:

1. Открыть главное окно программы
2. Выбрать пункт меню *Операции->Загрузка программы в контроллер.*
3. Выбрать скомпилированный программный файл и нажать кнопку открыть.

После этого начнется загрузка скомпилированного файла. Данная операция может занять несколько минут. В процессе загрузки программы контроллер сохраняет свою работоспособность и действует по старой программе. Смена программы в контроллере происходит после полной загрузки новой программы и продолжается около 0,01 секунды.

Программирование контроллера ведется на языке, напоминающем язык программирования С, далее будет приведено его описание.

Знакомство с языком

Переменные, типы данных и возможные операции над ними

Переменные – ячейки памяти контроллера, предназначенные для хранения числовых данных. Контроллер предоставляет для работы программы 1024 переменных. В каждую из переменных программа контроллера может записать данные и потом прочесть их в любое время.

Каждая переменная имеет тип unsigned short int (беззнаковое короткое целое). Диапазон возможных значений - от 0 до 65535. В некоторых случаях переменные используются, как логические, в этом случае 0 – ложь, все значения больше 0 считаются истиной.

Если требуется присвоить переменной с номером 15 значение 1, программный код будет выглядеть следующим образом:

var[15]=1;

Контроллер поддерживает арифметические и логические операции при работе с переменными и числами. Результат выполнения такой операции должен быть сохранен в переменную.

При составлении программы можно использовать следующие операции:

Операция	**Расшифровка**	**Тип**	**Синтаксис**
+	Сложение 2 переменных	арифметическая	var[1]=var[2]+var[3];
-	Вычитание	арифметическая	var[1]=var[2]-var[3];
*	Умножение	арифметическая	var[1]=var[2]*var[3];

/	Деление	арифметическая	var[1]=var[2]/var[3];
%	Остаток от деления	арифметическая	var[1]=var[2]%var[3];
!=	Не равно. Если переменные справа и слева не равны – возвращается истина, иначе - ложь	логическая	var[1]=var[2]!=var[3];
==	Равно. Если переменные справа и слева не равны – возвращается ложь, иначе -истина.	логическая	var[1]=var[2]==var[3];
>	Больше. Если значение переменной слева больше значения переменной справа, возвращается истина, иначе ложь	логическая	var[1]=var[2]>var[3];
<	Меньше. Если значение переменной слева меньше значения переменной справа, возвращается истина, иначе ложь	логическая	var[1]=var[2]<var[3];
>=	Больше-равно. Если значение переменной слева больше или равно значению переменной справа, возвращается истина, иначе ложь	логическая	var[1]=var[2]>=var[3];
<=	Меньше-равно. Если значение переменной слева меньше или равно значению переменной справа, возвращается истина, иначе ложь	логическая	var[1]=var[2]<=var[3];
$!=	Не равно. Если переменные слева только что стала не равна переменной справа – возвращается истина, иначе — ложь. Операция может выполняться только в случае размещения слева переменной.	логическая	var[1]=var[2]$!=var[3];
$==	Равно. Если переменные слева только что стала равна переменной справа – возвращается истина, иначе — ложь. Операция может выполняться	логическая	var[1]=var[2]$==var[3];

	только в случае размещения слева переменной.		
$>	Больше. Если переменные слева только что стала больше переменной справа – возвращается истина, иначе — ложь. Операция может выполняться только в случае размещения слева переменной.	логическая	var[1]=var[2]$>var[3];
$<	Меньше. Если переменные слева только что стала меньше переменной справа – возвращается истина, иначе — ложь. Операция может выполняться только в случае размещения слева переменной.	логическая	var[1]=var[2]$<var[3];
$>=	Больше-равно. Если переменные слева только что стала равна или больше переменной справа – возвращается истина, иначе — ложь. Операция может выполняться только в случае размещения слева переменной.	логическая	var[1]=var[2]$>=var[3];
$<=	Меньше-равно. Если переменные слева только что стала равна или меньше переменной справа – возвращается истина, иначе — ложь. Операция может выполняться только в случае размещения слева переменной.	логическая	var[1]=var[2]$<=var[3];
&&	Логическое И. Если обе переменных истины, возвращается истина, иначе – ложь.	логическая	var[1]=var[2]&&var[3];
||	Логическое ИЛИ. Если хотя бы одна из переменных истина, возвращается истина, иначе – ложь.	логическая	var[1]=var[2]||var[3];
!	Логическое НЕ Если оператор справа >0, оператор слева станет	логическая	var[1]=!var[2];

	равным 0. Если оператор справа =0, оператор слева станет равным 1. Пример: Результат !32 = 0; Результат !65000 = 0; Результат !0 = 1;		

Если требуется произвести вычисления более чем с 2 переменными, то операцию следует разбить на несколько более простых.
Пример
Исходное выражение:

var[3]=var[4]+var[5]*var[7];

Представление в виде понятном компилятору:

var[1023]=var[5]*var[7];
var[3]=var[4]+var[1023];

Для вычисления используется дополнительная переменная с номером 255.

Таймеры

Таймер – переменная, значение которой автоматически увеличивается каждую секунду на единицу.

Таймеры очень часто применяются при программировании контроллера, если требуется выполнить определенную последовательность действий после наступления какого-либо события с задержкой.

Любую из 1023 переменных контроллера можно сделать таймером. Для запуска таймера в переменной требуется выполнить следующий код -

StartTimer(<номер переменной>, <начальное значение>);

для остановки -

StopTimer(<номер переменной>);

Функции

Функции – команды контроллера предназначенные для изменения или получения состояний исполнительных модулей умного дома (реле, датчики освещенности, влажности, температуры и др), а также для выполнения операций со временем или таймерами. В том числе выше описанные команды запуска и остановки таймеров являются функциями.

Каждая функция возвращает значение – результат своего выполнения. Для примера рассмотрим функцию получения текущего времени:

var[2]=GetTime();

Результат – текущее время будет сохранено в переменную. Если функция была выполнена в 12:22:38, то в переменной var[2] окажется значение 1222 (ЧЧММ) – часы и минуты без разделительного знака ':'.

Результатом выполнения операции запуска таймера, является 1, если функция была выполнена успешно и 0 - если произошла ошибка. Т.к. вероятность появления ошибки при выполнении функции запуска таймера =0, результат можно не проверять.

Некоторые функции для своего выполнения требуют задания параметров – если требуется включить реле блока Secu16, то функции необходимо указать с каким именно блоком требуется произвести эту операцию, с каким реле и что именно требуется сделать.

Для включения 1 реле (8 точка) блока SE 6i5o с номером 4 требуется выполнить следующую функцию:

var[255]=Set_Relay_State(4,8,1);

В качестве параметров могут выступать не только константы, но и переменные:

var[255]=Set_Relay_State(var[1],var[2],var[3]);

Полный список функций приведен ниже:

Функция	***Описание***
GetTime()	Возвращает системное время контроллера в виде ЧЧММ (часы минуты)
GetHour()	Возвращает текущий час
GetMin()	Возвращает текущую минуту
GetDay()	Возвращает текущий день с начала месяца
GetMonth()	Возвращает номер месяца с начала года
GetYear()	Возвращает номер года начиная с 2000
GetDayofWeek()	День недели по счету(0 – 6), начиная с понедельника.
StartTimer(par1,par2)	Запускает таймер в переменной с номером par1 с начального значения par2
StopTimer(par1)	Останавливает таймер в переменной с номером par1
Random(par1)	Возвращает произвольное число от 0 до par1.
GetVarByNum(par1)	Возвращает значение переменной с номером par1. Если par1 больше максимального значение, возвращается 0.
SetVarByNum(par1,par2)	Присваивает переменной с номером par1 значение par2. Если par1 больше максимального значения, присвоения не происходит.
OnChangeVarValue(par1)	Возвращает 1, если переменная var[par1] изменилась с прошлого витка программы. Если переменная не изменилась, возвращается 0.
SetVarDefValue(par1,par2)	Присваивает переменной var[par1] начальное значение par2 — при загрузке контроллера в переменной var[par1] будет

	храниться значение par2. Кроме этого переменной var[par1] будет присвоено значение par2. Если функция выполнена успешно, возвращается значение 1.
Shutdown(par1)	Выключает / перезагружает контроллер. Данная функция предназначена для выключения контроллера программным способом при получении сигнала от источника бесперебойного питания о критическом заряде батареи или при зависании программных модулей. Если par1 равен 0, контроллер перезагрузится. Если пар1 равен 1, выключится.
GetPrgParam(par1)	Чтение параметра программы. par1 — номер параметра. Возвращает значение параметра или 0, если параметра с данным номером не существует. См. главу «Параметры программы»
SetPrgParam(par1, par2)	Задание параметра программы. par1 — номер параметра, par2 – значение. Возвращает 0, в случае изменения параметра, 1 — в случае ошибки.

Функции для передачи данных стороннему ПО

Функция	***Описание***
SendRemoteCmd(par1,par2,par3,par4)	Функция передачи команд программным модулям системы автоматизации Alpha. Par1 — номер удаленного сервера; Par2-4 см. в описании программных модулей. Подробнее об использовании данной функции можно узнать в документации к программным модулям Alpha: DataSender, Alpha: SMS Sender и пр.
StatVar(par1)	Сохраняет значение переменной при её изменении в таблицу stat базы данных MySQL. Таблица может использоваться для ведения логов температуры в помещениях, работы какого-либо оборудования и т.п. Par1 - номер переменной для отслеживания. В таблицу записывается, время события, тип события (0 значение изменилось, 1

	сохранение в связи с запуском контроллера), номер переменной, значение). Из базы данных записи удаляются автоматически, если им более 1 года.

Функции для работы с модулями ModBus

Функция	***Описание***
ReadInt16Register(par1,par2,par3,par4,par5)	Функция чтения значения регистра типа Int16 устройства ModBus. Par1 – команда чтения регистра (3 - *Read Holding Registers* , 4 - *Read Input Registers*) par2 – номер устройства par3 – номер регистра par4 – количество регистров для чтения par5 – номер переменной для сохранения результата. Результатом выполнения функции является код ошибки. Если результат не равен 0, при выполнении операции произошла ошибка. Пример: Если количество запрашиваемых регистров равно 3, результат будет сохранен в переменные с номерами par5, par5+1, par5+2. Если в функции планируется задавать значение par4 отличное от 1, обязательно убедитесь, что модуль ModBus поддерживает запросы более 1 регистра.
PresetInt16Reg(par1,par2,par3,par4)	Функция записи значения регистра типа Int16 устройства ModBus. Par1 – команда записи регистра (по умолчанию 16) par2 – номер устройства par3 – номер регистра par4 – записываемое значение. Если функция вернула значение отличное от 0, при передаче данных модулю возникла ошибка. Необходимо помнить, что после передачи команды модулю ModBus изменение значения регистра может происходить с некоторой задержкой (до 1 секунды).
GetMBUnitStatus(par1)	Функция возвращает количество ошибок, которые произошли при выполнении команд чтения и записи параметров модуля. Если модуль не отвечает или отвечает с

	ошибкой 5 раз подряд, все команды для этого модуля будут игнорироваться следующие 250 циклов программы. Функция возвращает значения от 0 до 5. par1 – номер устройства

Функции для работы с модулями про протоколу ADNet+.

Функция	***Описание***
Get_Point_State(par1,par2)	Возвращает состояние точки par2 блока Secu16 (номер блока в системе par1) Используется для модулей: SE8i8o, SE 16i, SE 6i5o
Set_Relay_State(par1, par2, par3)	Устанавливает состояние выходного реле модуля расширения. Par1 – номер модуля, par2 — номер параметра, par3 – значение (1 замкнуть, 2 разомкнуть). Функция используется при работе с блоком SE 6i5o
Set_Relays_State(par1,par2,par3)	Устанавливает значение всех выходов блока расширения (номер блока в системе par1). Par2 – операция над выходами модуля, Par3 – маска. Варианты par2: 0 – Включение по маске 1 – Выключение по маске 2 – установка значений выходов по маске. Пример1: Состояние выходов: 10000000 Операция: 0 Маска: 00011111 Результат: 10011111 Пример2: Состояние выходов: 11111110 Операция: 1 Маска: 00111111 Результат: 11000000 Пример3: Состояние выходов: 00000010 Операция: 2 Маска: 00110000 Результат: 00110000 Используется для модулей: SE 16i, SE 6i5o

	Внимание модуль SE8i8o работает только с операцией 2
Set_Analog_Output(par1,par2,par3)	Команда предназначена для управления модулем SE 2o 0-10v. Модуль имеет 2 выходных канала 0 -10 Вольт. Значение каждого канала может вартьироваться от 0 до 31. Par1 — номер модуля, par2- операция, par3 — уровень Par2: <0> - установка значения 0 канала в уровень <par3> <1> - увеличение значения 0 канала на уровень <par3> <2> - уменьшение значения 0 канала на уровень <par3> <3> - установка значения 1 канала в уровень <par3> <4> - увеличение значения 1 канала на уровень <par3> <5> - уменьшение значения 1 канала на уровень <par3> Если происходит попытка установить уровень больше 31, уровень устанавливается = 31.
GetUnitData(par1)	Возвращает значение поля Data модуля. Для датчиков SE Light, SE Humb, BobCat X в поле Data хранится значение измеряемого ими параметра. (номер модуля в системе par1)
GetUnitData0(par1)	Возвращает младший байт поля Data модуля. (номер модуля в системе par1)
GetUnitData1(par1)	Возвращает старший байт поля Data модуля. (номер модуля в системе par1)
GetUnitActiveStatus(par1)	Возвращает показатель активности модуля с номером par1. Если модуль работает исправно, возвращается 1, если модуль не опрашивается возвращается 0. Центральный контроллер постоянно опрашивает состояние всех подключенных модулей. Если модуль не ответил на запрос контроллера несколько раз подряд,

	контроллер перестает его опрашивать. Функция GetUnitActiveStatus позволяет узнать из программы опрашивается ли модуль с указанным номером.
Get_Unit_Param(par1, par2)	Чтение параметра из модуля расширения. Par1 – номер модуля, par2 — номер параметра.
Set_Unit_Param(par1, par2, par3)	Задание параметра в модуле расширения. Par1 – номер модуля, par2 — номер параметра, par3 – значение.
SEEZ_Play(par1,par2)	Возвращает результат выполнения операции воспроизведения сообщения par2 блоком SpeakEasy (номер блока в системе par1)
SEEZ_Rec(par1,par2)	Возвращает результат выполнения операции записи сообщения par2 блоком SpeakEasy (номер блока в системе par1)

Условный оператор

Условный оператор позволяет проверить некоторое действие и в зависимости от результатов проверки выполнить то или иное действие. Таким образом условный оператор – средство ветвления программных вычислений и действий.
Структуру условного оператора можно представить следующим образом

if (условие) {Блок действий 1};

или

if (условие) {Блок действий 1} else {Блок действий 2};

Условный оператор работает по следующей схеме - вычисляется значение условия, если условие истинно (значение больше 0) – выполняется *Блок действий 1*, иначе - *Блок действий 2*

Пример

```
if (var[3]==1)
{
   var[4]=0;
}
else
{
   var[4]=1;
};
```

Программа может иметь вложенные условные операторы, что облегчает процесс программирования и сокращает количество программного кода.

Оператор безусловного перехода goto

Иногда при выполнении программы требуется из одного места программы

«перепрыгнуть» в другое, для того чтобы пропустить выполнение блока команд.

Для этих целей в язык программирования был введен оператор безусловного перехода. Для указания места в которое требуется перепрыгнуть, используются метки label[]. Блок программы с использованием оператора goto и меток может выглядеть следующим образом:

```
Блок операций 1
goto label[4];
Блок операций 2
label[4]:
Блок операций 3
```

В одной программе могут встречаться до 256 меток, номер метки указан в квадратных скобках.

Комментарии

Для облегчения восприятия программы, программист может в код программы вставлять небольшие текстовые комментарии с описанием того, что именно выполняет блок команд и для каких целей он используется. Любой комментарий начинается символом «#», далее следует текст комментария. Комментарием будет считаться любой текст до конца строки начиная с символа «#».

Параметры пользовательской программы

Пользовательская программа имеет свои параметры, которые могут быть прочитаны или изменены функциями GetPrgParam и SetPrgParam.

№	Описание	Операции
10	Время ожидания ответа от ADNet+ устройства в миллисекундах. Параметр начинает работать сразу после изменения. Значение не сохраняется в энергонезависимую память и будет сброшено на значение по умолчанию (100 млс.) после перезагрузки контроллера. Данным параметром рекомендуется пользоваться при опросе/управлении модулей, подключенных через Ethernet шлюзы, когда время ответ может быть превышено из за скорости передачи данных сети.	Чтение/ Запись
20	Время ожидания ответа от ModBus устройства в миллисекундах. Параметр начинает работать сразу после изменения. Значение не сохраняется в энергонезависимую память и будет сброшено на значение по умолчанию (100 млс.) после перезагрузки контроллера. Данным параметром рекомендуется пользоваться при опросе/управлении модулей, подключенных через Ethernet шлюзы, когда время ответ может быть превышено из за скорости передачи данных сети.	Чтение/ Запись

Горячие клавиши программы AlphaSE Controller

Горячие клавиши позволяют управлять работой основных служб программы AlphaSE Controller: опрос устройств ADNet+, выполнение пользовательской программы, обмен данными с MySQL сервером.

Данный функционал полезен в случае, когда работа данных служб вызывает подвисание программы AlphaSE Controller и, как следствие, невозможность подключения к ней программы AlphaSE Configurator.

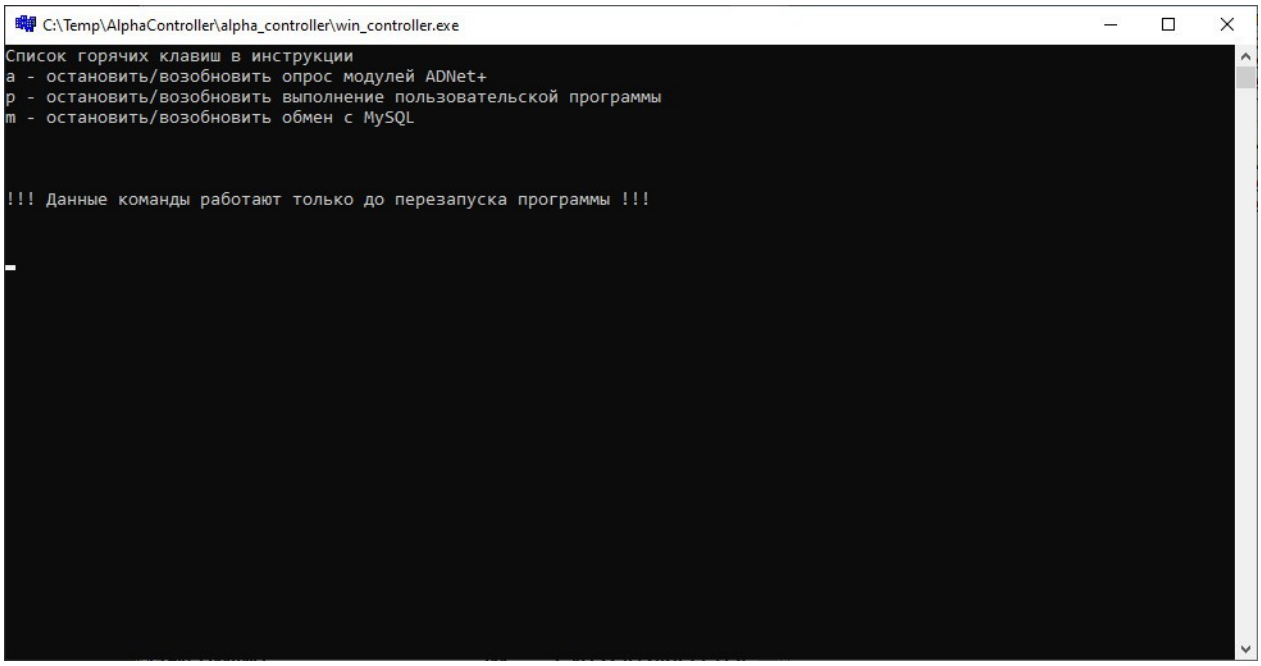
Рассмотрим типовую проблему. К контроллеру подключены несколько модулей по протоколу ModBus. В каждом цикле пользовательской программы идет опрос регистров этих модулей. В случае отключения питания модулей AlphaSE Controller будет посылать им запросы, но не будет получать ответы. Ожидание ответа от одного модуля длится 100млс. Если не ответят 20 модулей, подключенных к шине, суммарное время потраченное контроллером на их опрос займет 2 секунды. В течение этого времени программа контроллер будет находиться в подвисшем состоянии и может вызвать ошибки в работе AlphaSE Configurator.

Список горячиших клавиш:

a – остановить/возобновить опрос модулей по протоколу ADNet+;

p - остановить/возобновить выполнение пользовательской программы;

m - остановить/возобновить выгрузку данных в MySQL и получение из СУБД команд на исполнение;



Примеры программ для модулей SE и Adicon

Автоматическое управление освещением двора частного дома

Оборудование:

- Контроллер Alpha SE v.2.0,
- Secu16,
- BobCat L,

- Датчик движения.

Задание:

Освещение двора должно включаться автоматически при плохом уровне освещенности, обнаружении движения на территории двора или открытии входных дверей в дом или во двор. Выключение через 5 минут после пропадания движения и закрытия всех дверей.

Описание блока Secu16
Номер в системе - 1
Вход 0 – датчик открытия входной двери дома,
Вход 1 – датчик открытия входной двери двора,
Вход 2 – датчик движения, установленный во дворе дома,
Выход 8 – реле управления освещением

Описание блока BobCat L
Номер в системе – 2

Программный код

```
var[0]=Get_Point_State(1,0); #   Состояние входной двери в дом
var[1]=Get_Point_State(1,1); #   Состояние входной двери во двор
var[2]=Get_Point_State(1,2); #   Состояние уличного датчика движения
var[3]=GetUnitData(2); #   Уровень освещенности двора
var[4]=Get_Point_State(1,9); #   Состояние реле управления освещением

#   Сравнение порогового уровня освещенности с имеющимся
if (var[3]<200)
{
  var[255]=var[0]||var[1]; #   Если открыта дверь во двор или дом
  var[254]=var[2]||var[255]; #   Или сработал датчик движения
  if (var[254]==1) #   Если наступило хотя бы одно из событий
  {
    if (var[4]==0) #   Если свет еще не горит
    {
      var[255]=Set_Relay_State(1,9,1); #   Включаем свет
    };
    var[255]=StartTimer(5,1); #   Запускаем таймер автоматического
освещения в переменной 5
  };
};

if (var[5]==300) #   Если прошло 5 минут (300 секунд)
{
  var[255]=Set_Relay_State(1,9,0); #   Выключаем свет
  var[255]=StopTimer(5); #   Останавливаем таймер
};
```

Автоматическое управление поливом в оранжерее

Оборудование:

- Контроллер Alpha SE v.2.0,
- Secu16,
- Эл. клапан 3 шт.

Задание:
Обеспечить полив растений каждое утро по графику:
8:00 - 8:15 - зона1,
8:15 - 8:30 - зона2,
8:30 - 8:45 - зона3.

Описание блока Secu16
Выход 8 – реле управления водяным клапаном зоны 1,
Выход 9 – реле управления водяным клапаном зоны 2,
Выход 10 – реле управления водяным клапаном зоны 3,

Программный код

```
var[0]=GetTime(); #   Время
var[1]=Get_Point_State(1,8); #   Состояние клапана 1 зоны
var[2]=Get_Point_State(1,9); #   Состояние клапана 2 зоны
var[3]=Get_Point_State(1,10); #   Состояние клапана 3 зоны


#   Если пришло время полива первой зоны
if (var[0]==800) # 08:00
{
   if (var[1]==0) # Если  клапан еще не открыт
   {var[255]=Set_Relay_State(1,8,1);}; #   Открываем клапан первой зоны
};

if (var[0]==815 ) # 08:15
{
   if (var[1]!=0) # Если  клапан зоны 1 открыт
   {var[255]=Set_Relay_State(1,8,0);}; #   Закрываем клапан зоны 1

   if (var[ 2]==0) # Если  клапан зоны 2 еще не открыт
   {var[255]=Set_Relay_State(1,9,1); }; #   Открываем клапан зоны 2
};

if (var[0]==830 ) # 08:30
{
   if (var[ 2]!=0) #   Если клапан зоны 2 открыт
   {var[255]=Set_Relay_State(1,9,0);}; #   Закрываем  клапан зоны 2

   if (var[ 3]==0) #   Если клапан зоны 3 еще не открыт
   {var[255]=Set_Relay_State(1,10,1);}; #   Открываем клапан зоны 3
 };

if (var[0]==845 ) # 08:45
{
```

```
   if (var[3]!=0) #   Если клапан зоны 3 открыт
   {var[255]=Set_Relay_State(1,10,0);};#   Закрываем клапан зоны 3
};
```

Включение уличного освещения по графику с привязкой к месяцу

Управление уличным освещением, как в прочем и другими устройствами, с привязкой к месяцу является достаточно распространенной задачей.

Для каждого из месяцев года характерны свои особенности: средняя температура, время рассвета, время заката и прочее.

В этом примере мы постараемся описать схему управления уличным освещением и получить достаточно компактный код.

Оборудование:

- Контроллер Alpha SE v.1.0,
- Secu16.

Задание:

Реализовать систему уличного освещения. Включение уличного освещения должно происходить автоматически в
январе – 17:00,
феврале – 17:00,
марте – 18:00,
апреле – 18:00,
майе – 19:00,
июне - 19:00,
июле – 19:30,
августе - 19:30,
сентябре – 19:00,
октябре – 18:30,
ноябре - 18:00,
декабре – 17:00.
Выключение освещения в данной задаче не рассматривается.

Время включения освещения для каждого месяца будет храниться в переменных 0-11. Для избежания проблем с утратой значений переменных после перезагрузки контроллера, время включения освещения следует задать не только в самих переменных, но и в их начальные значения (см. гл. «Использование начальных значений переменных»).

Описание блока Secu16
Выход 8 – реле управления уличным освещением.

Программный код

```
#var[0] - время включения уличного освещения в январе
#var[1] - время включения уличного освещения в феврале
#var[2] - время включения уличного освещения в марте
#var[3] - время включения уличного освещения в апреле
```

```
#var[4] - время включения уличного освещения в майе
#var[5] - время включения уличного освещения в июне
#var[6] - время включения уличного освещения в июле
#var[7] - время включения уличного освещения в августе
#var[8] - время включения уличного освещения в сентябре
#var[9] - время включения уличного освещения в октябре
#var[10] - время включения уличного освещения в ноябре
#var[11] - время включения уличного освещения в декабре
#Secu16 unit 1 point 8 - реле управляющее уличным освещением

var[20]=GetMonth(); #Получаем номер месяца
var[21]=Get_Point_State(1,8);  # Получаем текущее состояние уличного
освещения
var[22]=GetTime(); # Получаем текущее время
var[23]=GetVarByNum(var[20]); #Получаем время включения освещения в
текущем месяце

if (var[22]==var[23]) #Если пришло время включения уличного освещения
{
  if (var[21]==0) #И освещение еще не включено
  {
    var[255]=Set_Relay_State(1, 8, 1); # Включаем освещение
  };
};
```

Рекомендации по программированию

Выделение переменных общего назначения

Зачастую, при написании программы контроллера одни и те же функции используются по нескольку раз. К таким функциям можно отнести: GetTime(), GetHour(), GetMin(), GetDay(), GetMonth(), GetYear(), GetDayofWeek(). Для избежания увеличения размера программы рекомендуется пользоваться следующим методом оптимизации кода:

- Выделите блок переменных под данные общего назначения, которые возможно будут использоваться несколько раз в ходе выполнения программы.
- Вначале программы произведите присваивание этим переменным значений функций.

```
var[0]=GetTime(); # Время
var[1]=GetDay(); #   День месяца
var[3]=Get_Point_State(1,8); #   Состояние клапана 1 зоны
var[4]=Get_Point_State(1,9); #   Состояние клапана 2 зоны
var[5]=Get_Point_State(1,10); #   Состояние клапана 3 зоны
```

- Обязательно оставьте текстовые комментарии, поясняющие значения этих переменных. Это позволит в ходе написания программы тратить меньше времени на поиск необходимой переменной.
- В ходе написания программы пользуйтесь уже полученными переменными и не вызывайте функции повторно . Такой ход позволит увеличить скорость

выполнения программы, т. к. операция присваивания переменной гораздо быстрее вызова функции.

Использование комментариев

В ходе написания программы оставляйте подробные комментарии выполняемых действий. Это позволит быстрее понять логику программы при ее последующей модернизации (Заказчик может захотеть изменить логику работы умного дома через, год-два после установки системы). Использование комментариев не сказывается на скорости выполнения программы.

Использование начальных значений переменных

Предположим, что заказчик решил установить систему автоматизации в своем магазине. Рассмотрим 2 задачи которые должен будет взять на себя контроллер:

-управление вывеской (должна включаться автоматически каждый вечер в 19:00 и выключаться в 23:30),

-выключение забытого освещения во всех помещения в 19:00.

Все статические данные (время включения вывески и отключения освещения), которые в будущем все же могут измениться, лучше хранить в переменных. При использовании такого подхода изменение времени включения вывески и выключения освещения не потребует модернизации программного кода и займут всего пару минут.

Но в связи с тем, что значения всех переменных теряются при отключении питания контроллера рекомендуется использовать возможность задания начальных значений переменных. Начальные значения хранятся в энергонезависимой памяти контроллера и при загрузке контроллера автоматически присваиваются переменным.

Таким образом при использовании переменных: var[0] – для хранения времени включения вывески, var[2] – выключения вывески, var[3] – выключения забытого освещения, требуется задавать не только значение самих переменных, но и их начальные значения.

Возможные неисправности

При запуске программы появляется сообщение об ошибке «Порт закрыт или уже открыт другой программой»

- Убедитесь, что программа Alpha Controller запущена.
- Убедитесь, что не запущена еще одна программа Alpha SE Configurator.
- Убедитесь, что в программе Alpha Controller не возникают ошибки при работе с модулями расширения. Если сообщения об ошибках при опросе модулей расширения SE или Adicon появляются, подождите пока контроллер не прекратит их опрашивать и запустите Alpha SE Configurator повторно. Если модуль SE или Adicon не отвечает на 10 запросов от контроллера подряд, его опрос прекращается на 1000 циклов.