Руководство пользователя

ТК16L Средства разработки скриптов

ЗАО НПФ ПРОРЫВ

Содержание

Цель документа

Целью документа является предоставление всей необходимой информации о средствах разработки скриптов устройств ТК16L. Скрипты устройств ТК16L предназначены для локального автоматизированного управления, взаимодействия контроллера с внешними устройствами, взаимодействия контроллера с системой Телескоп+.
В документ входят разделы:

- Описание функций работы с устройствами
- Описание редактора скриптов

Документ предназначен для разработчиков скриптов, загружаемых в устройства серий ТК16L/E-422.
Документ входит в состав пакета, который предоставляется сторонним разработчикам.

Общие сведения

В данном документе описан порядок разработки скриптов терминальных контроллеров серий ТК16L/E-422. Скрипты используются для организации работы терминальных контроллеров в системе Телескоп+ или для локального автоматизированного управления и взаимодействия с внешними устройствами.
Разработка скриптов ведется в специализированной среде программирования на языках C# или Basic, что позволяет разрабатывать скрипты для поддержки алгоритмов управления любой сложности. В качестве среды программирования применяется приложение **Редактор скриптов**.
Приведено описание типичных классов и методов классов, которые используются в скриптах.

Редактор скриптов

Назначение

Приложение Редактор скриптов фактически представляет собой текстовый редактор с особенностями, специфичными для удобной разработки скриптов для устройств типа ТК16L.

Приложение предназначено для выполнения следующих операций:

- Ввод и редактирование кода;
- Проверка синтаксиса;
- Пошаговая отладка;
- Развертывание скрипта на контроллере;
- Развертывание программы-обработчика скриптов на контроллере.

Приложение **Редактор скриптов** имеет простой интуитивно понятный интерфейс. В редакторе реализуются такие особенности редактирования текста, такие как специфичная подсветка строк скрипта, свертка и развертка текстов программного кода и пр.

Разработка скриптов ведется в отдельных модулях в рамках проекта. Например, можно создать проект для работы с Modbus устройствами, удаленными портами, системой Телескоп+ и пр.

Загрузка редактора скриптов (ScriptStudio.rar) доступна со страницы http://www.proryv.com/support/download/ .

Главное окно

Главное окно приложения **Редактор скриптов** предназначено для добавления и удаления проектов и модулей проектов, а также для редактирования, проверки синтаксиса отладки и пошаговой отладки скриптов. Из главного окна выполняются операции развертывания скрипта на контроллере и развертывания программы-обработчика скриптов на контроллере.

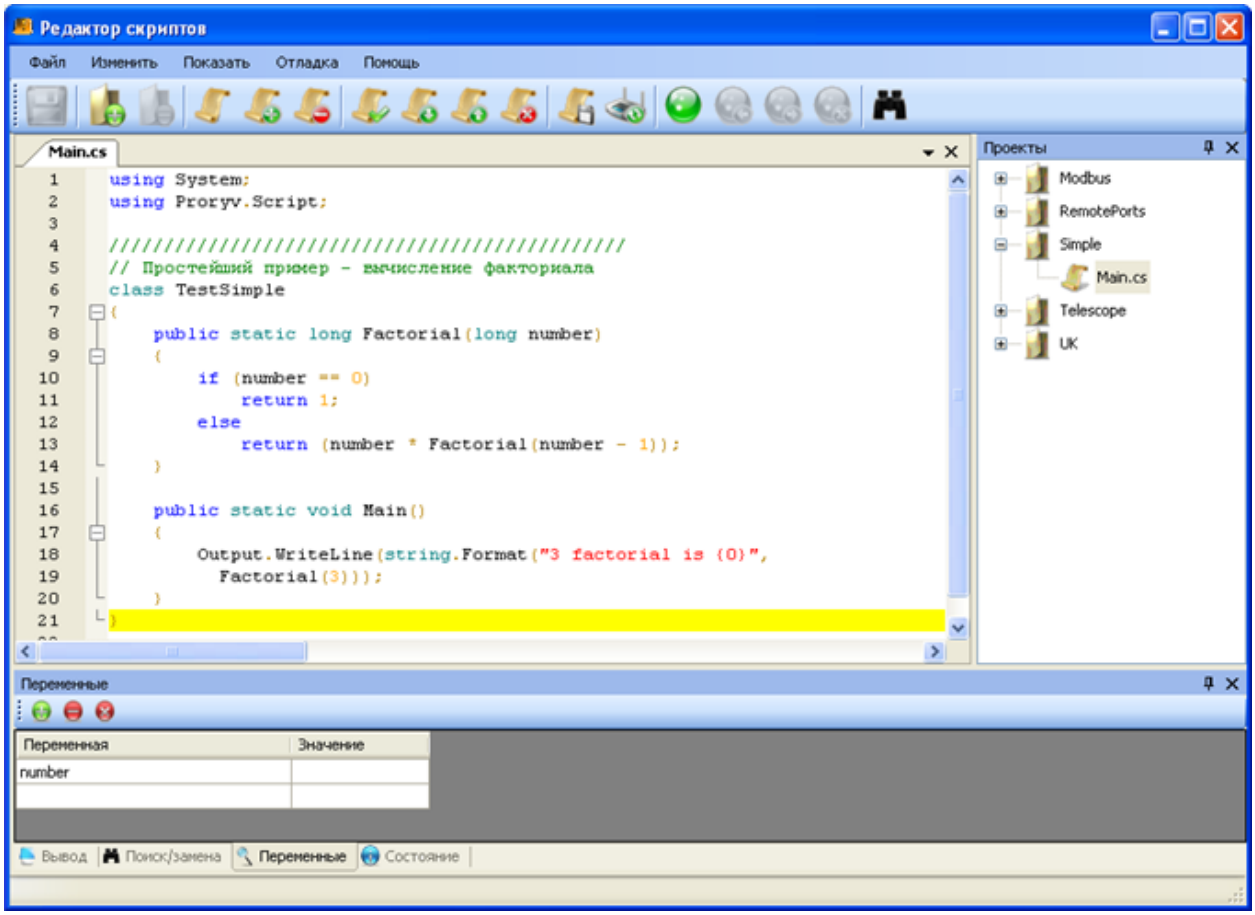


Рисунок 1 Редактор скриптов, главное окно

Панели:

- <панель редактора> – панель предназначена для отображения и редактирования программных текстов модулей. Допускается открывать несколько вкладок для просмотра и редактирования различных модулей.
- <сервисная панель > – нижняя панель предназначена для вывода сообщений, просмотра значений переменных в режиме отладки, информации о поиске и замене фрагментов кода, отображения состояния после запуска скрипта.
- панель **Проекты** – панель предназначена для отображения перечня проектов и скриптов, входящих в проекты, а также выбора наименования скрипта для отображения его программного кода на панели редактора. Для работы с модулем или проектом выберите соответствующий объект на панели **Проекты**.

Опции меню и кнопки панели инструментов

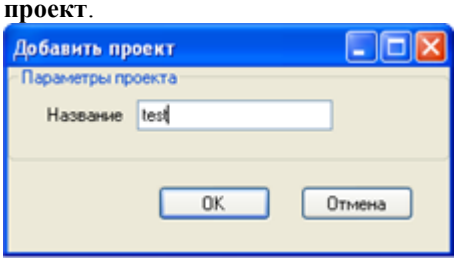
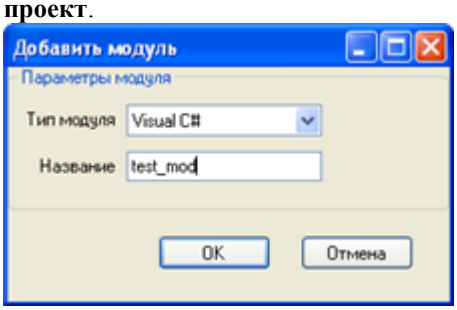
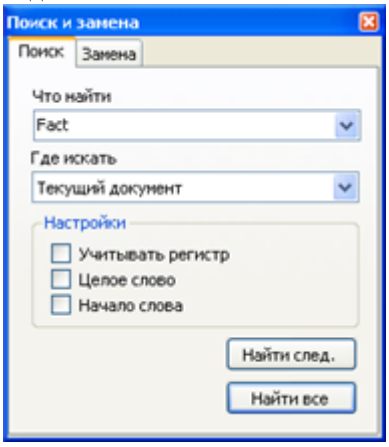
В разделе приведены опции меню и соответствующие кнопки панели инструментов главного окна приложения **Редактор скриптов**. Ряд опций главного меню доступен также из контекстного меню.

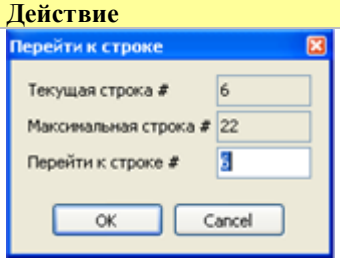
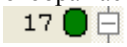
Меню Файл

Опция	Кнопка	Действие
Сохранить		Сохранение выполненных изменений
Выход		Выход из приложения

Меню Изменить

Опция	Кнопка	Действие

Опция	Кнопка	Действие
Добавить проект		Добавление проекта в перечень проектов. Переход в окно **Добавить проект**.  Введите наименование проекта в поле **Название**, нажмите кнопку **OK**.
Удалить проект		Удаление проекта, выбранного на панели **Проекты**.
Показать модуль		Отображение программного кода модуля, выбранного на панели Проекты. На панели редактора откроется новая закладка для отображения модуля.
Добавить модуль		Добавление модуля в состав выбранного проекта. Переход в окно **Добавить проект**.  Выберите тип модуля (язык разработки). Введите наименование модуля в поле **Название.** Нажмите кнопку **OK**.
Удалить модуль		Удаление выбранного модуля.
Поиск и замена		Переход к стандартной форме поиска и замены фрагмента программного кода.  Поиск выполняется в части программного кода модуля, начиная от позиции курсора. При нажатии кнопки **Найти след.** будет выделен первый найденный фрагмент кода. При нажатии кнопки **Найти все** на нижнюю панель выводится информация о номерах строк кода, где был найден заданный фрагмент. Найти все "Fact", Текущий документ, [] Main.cs (7): public static long Factorial(long number) Main.cs (12): return (number * Factorial(number - 1)); Main.cs (17): Output.WriteLine(string.Format("3 factorial is {0}", Main.cs (18): Factorial(3))); Найдено строк: 4
Перейти		Переход на строку указанного номера в скрипте. Переход в окно **Перейти к строке**.

Опция	Кнопка	Действие
		 Введите номер строки в поле **Перейти к строке**, нажмите **OK**.
Закладки: Добавить закладку Предыдущая закладка Следующая закладка Удалить закладки	   	Закладками в редакторе скриптов можно пометить определенные строки программного кода. Закладки являются удобным инструментом для перемещения по тексту программы при редактировании кода. Закладка отображается следующим образом:  Для добавления закладки установите курсор в строке кода и нажмите кнопку **Добавить закладку**. Для удаления всех закладок нажмите кнопку **Удалить закладки**.

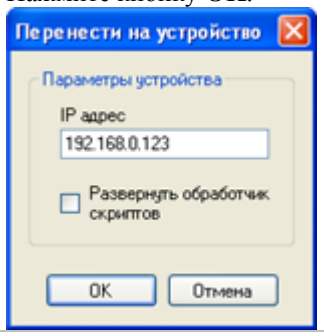
Меню Показать

Меню управляет отображением окна редактора и кода скрипта. Для управления отображением установите/снимите флаг для соответствующей опции.

Опция	Действие
Проекты	Окно редактора: отображение панели **Проекты**.
Вывод	Окно редактора: отображение закладки **Вывод** на нижней панели.
Поиск/замена	Окно редактора: отображение закладки **Поиск/Замена** на нижней панели.
Переменные	Окно редактора: отображение закладки **Переменные** на нижней панели.
Состояние	Окно редактора: отображение закладки **Состояние** на нижней панели.
Номера строк	Скрипт: отображение номеров строк.
Пробелы	Скрипт: отображение знаков пробела.
Перенос строки	Скрипт: отображение знаков переноса строки.
Конец строки	Скрипт: отображение знаков конца строки.
Каталог данных	Переход в окно обозревателя ОС Windows (папка ScriptStudio) для показа содержания каталога данных.

Меню Отладка

Опция	Кнопка	Горячая клавиша	Действие
Компилировать		F6	**Внимание!** Выполняется компиляция выбранного модуля или всех модулей выбранного проекта. Это может быть не тот модуль, вкладка которого открыта в редакторе. Выводится сообщение об ошибке компиляции или об успешной компиляции. Строка, в которой найдена ошибка, помечается следующим образом:  На вкладке **Состояние** выводится сообщение об ошибке/состоянии компиляции. Error: CS1001, Identifier expected. ModuleName: Main.cs LineNumber: 15 Line: p ublic static void Main()
Внутренняя отладка		F11	Выполняется отладка скрипта, включая входы в методы при вызове методов класса. Нажмите повторно кнопку **Внутренняя отладка** для просмотра состояния следующего шага скрипта. На вкладке **Состояние** выводится сообщение о текущем шаге отладки:

Опция	Кнопка	Горячая клавиша	Действие
			State: j Module: Main.cs Line 17 : Output.WriteLine(string.Format("3 factorial is {0}", Call stack: Main() at line 21
Внешняя отладка		F10	Выполняется отладка скрипта, без входов в методы при вызове методов класса. Нажмите повторно кнопку **Внешняя отладка** для просмотра состояния следующего шага скрипта. На вкладке **Состояние** выводится сообщение о текущем шаге отладки: State: j Module: Main.cs Line 20 : } Call stack: WriteLine(3 factorial is 6) at line 18
Сбросить отладку			Выход из режима отладки.
Сохранить компилированные модули			Сохранение компилированных модулей.
Перенести на устройство			Загрузка проекта в устройство. Нажмите кнопку **OK** в форме запроса на подтверждение загрузки. Введите IP адрес устройства в поле **IP адрес** окна **Перенести на устройство**. Установите флаг в **поле Развернуть обработчик скриптов**, если в контроллер не загружен обработчик скриптов. Нажмите кнопку **OK**. 

Последовательность выполнения операций

При разработке скриптов выполняются следующие типичные операции:

1. Добавление нового проекта.
2. Добавление нового скрипта в проект.
3. Ввод кода скрипта.
4. Компиляция скрипта.
5. Отладка скрипта.
6. Сохранение компилированных модулей.
7. Загрузка проекта в устройство.

Таблица классов и методов

В таблице приведено название и назначение классов и методов классов, используемых для разработки скриптов. В столбце Тип литера С используется для обозначения класса, далее следует описание конструктора (К) и/или методов класса. Литера П используется для обозначения перечисления.

Тип	Наименование	Назначение
С	Proryv.Script.Output – класс	Статический класс для вывода сообщений о выполнении скрипта в файл *.log. Файл размещен в одном каталоге со скриптом. Каталог по умолчанию: **0:/NANDFlash/TK16L/DATA/Init/Script/Projects/ProjectName/** где ProjectName – название проекта, в который включен скрипт.
	Output.WriteLine – метод	Выводит сообщение в лог-файл
	Output.WriteLine – метод	Выводит сообщение со списком параметров в лог-файл
	Output.WriteWarning – метод	Выводит предупреждение в лог-файл
	Output.WriteWarning – метод	Выводит предупреждение со списком параметров в лог-файл
	Output.WriteError – метод	Выводит сообщение об ошибке в лог-файл
	Output.WriteError – метод	Выводит сообщение об ошибке со списком параметров в лог-файл
	Output.WriteException – метод	Выводит исключение в лог-файл
	Output.Format – метод	Выполняет подстановку параметров, заданных в списке, в строку
С	Proryv.Script.Array – класс	Статический класс для обработки данных в массивах.
	Array.Sort - метод	Сортирует массив
С	Proryv.Script.DataUtils – класс	Статический класс для обработки данных. Используется, в частности, при обмене данными с системой Телескоп+.
	DataUtils.StrHexToArray – метод	Преобразует hex строку в массив байтов.
	DataUtils.DataToStrHex – метод	Преобразует массив байтов в hex строку.
	DataUtils.CRC16 – метод	Подсчитывает контрольную сумму для фрагмента массива.
	DataUtils.IsSetBit – метод	Проверяет установку бита.
	DataUtils.IsEqual – метод	Сравнивает массивы.
	DataUtils.IpAddressToData – метод	Преобразует IP адрес к массиву байтов.
	DataUtils.DataToIpAddress – метод	Преобразует массив байтов в IP адрес.
	DataUtils.GetData – метод	Возвращает указанную часть массива.
	DataUtils.ByteToData – метод	Преобразует число типа **Byte** в массив байтов.
	DataUtils.DataToByte – метод	Преобразует массив байтов в число типа Byte.
	DataUtils.SByteToData – метод	Преобразует число типа **SByte** в массив байтов. SByte представляет собой 8-битовое целое число со знаком.
	DataUtils.DataToSByte – метод	Преобразует массива байтов в число типа SByte.

Тип	Наименование	Назначение
	DataUtils.UInt16ToData – метод	Преобразует число типа **UInt16** в массив байтов.
	DataUtils.DataToUInt16 – метод	Преобразует массив байтов в число типа UInt16.
	DataUtils.Int16ToData – метод	Преобразует число типа **Int16** в массив байтов.
	DataUtils.DataToInt16 – метод	Преобразует массив байтов в число типа Int16.
	DataUtils.UInt32ToData – метод	Преобразует число типа **UInt32** в массив байтов.
	DataUtils.DataToUInt32 – метод	Преобразует массив байтов в число типа UInt32.
	DataUtils.Int32ToData – метод	Преобразует число типа **Int32** в массив байтов.
	DataUtils.DataToInt32 – метод	Преобразует массив байтов в число типа Int32.
	DataUtils.UInt64ToData – метод	Преобразует число типа **UInt64** в массив байтов.
	DataUtils.DataToUInt64 – метод	Преобразует массив байтов в число типа UInt64.
	DataUtils.Int64ToData – метод	Преобразует число типа **Int64** в массив байтов.
	DataUtils.DataToInt64 – метод	Преобразует массив байтов в число типа Int64.
	DataUtils.BoolToData – метод	Преобразует число типа **Bool** в массив байтов.
	DataUtils.DataToBool – метод	Преобразует массив байтов в число типа Bool.
	DataUtils.ArrayToData – метод	Копирует массив байтов.
	DataUtils.DataToArray – метод	Копирует массив байтов.
	DataUtils.WideStringToData – метод	Преобразует строку (**Unicode**) в массив байтов.
	DataUtils.DataToWideString – метод	Преобразует массив байтов в строку (Unicode).
	DataUtils.DataToSingle – метод	Преобразует массив байтов в число типа **Single**.
	DataUtils.SingleToData – метод	Преобразует число типа Single в массив байтов.
	DataUtils.DataToDouble – метод	Преобразует массив байтов в число типа **Double**.
	DataUtils.DoubleToData – метод	Преобразует число типа Double в массив байтов.
	DataUtils.DateTimeToData – метод	Преобразует дату и время в массив байтов (8 байт, год от 2000).
	DataUtils.DataToDateTime – метод	Преобразует массив байтов в переменную типа DateTime (8 байт, год от 2000).
П	Proryv.Script.Bool – перечисление	Перечисление: False = 0 True = 1
С	Proryv.Script.Stopwatch – класс	Класс для измерения затраченного времени.
К	Stopwatch – конструктор	Инициализирует новый экземпляр класса Stopwatch.
	Reset – метод	Останавливает измерение интервала времени и обнуляет затраченное время.
	Start – метод	Запускает измерение затраченного времени.
	Restart – метод	Останавливает измерение затраченного времени, обнуляет затраченное время и начинает новое измерение времени.
	IsRunning – свойство	Возвращает признак запуска измерения времени.

Тип	Наименование	Назначение
	Stop – метод	Останавливает измерение затраченного времени.
	ElapsedMilliseconds – свойство	Возвращает затраченное время в миллисекундах.
С	Proryv.Script.Watchdog – класс	Класс для реализация процесса типа watchdog: контроль зависания контроллера, перезагрузка контроллера при зависании.
К	Watchdog – конструктор	Инициализирует новый экземпляр класса Watchdog.
	Start – метод	Запускает таймер Watchdog.
	Stop – метод	Останавливает таймер Watchdog.
	Refresh – метод	Обновляет таймер Watchdog.
С	Proryv.Script.RemotePort – класс	Класс для работы по линиям RS485/RS232.
К	RemotePort – конструктор	Инициализирует новый экземпляр класса RemotePort.
	Dispose – метод	Освобождает ресурсы.
	Open – метод	Открывает порт.
	Close – метод	Закрывает порт.
	IsOpen – свойство	Возвращает признак открытия порта.
	IPAddress – свойство	Возвращает IP адрес порта.
	TCPPort – свойство	Возвращает номер TCP порта.
	BaudRate – свойство	Возвращает и устанавливает скорость обмена.
	DataBits – свойство	Возвращает и устанавливает количество бит данных.
	System.IO.Ports.StopBits Stopbits – свойство	Возвращает и устанавливает длину стоп-бита.
	System.IO.Ports.Parity Parity – свойство	Возвращает и устанавливает способ контроля четности.
	ReadTimeoutInterval – свойство	Возвращает и устанавливает таймаут чтения, мс (межсимвольный)
	ReadTimeoutMultiplier – свойство	Возвращает и устанавливает таймаут чтения, мс (множитель)
	ReadTimeoutConstant – свойство	Возвращает и устанавливает таймаут чтения, мс (константа)
	WriteTimeoutMultiplier – свойство	Возвращает и устанавливает таймаут записи, мс (множитель)
	WriteTimeoutConstant – свойство	Возвращает и устанавливает таймаут записи, мс (константа)
	Read – метод	Считывает массив байтов из Modbus устройства, помещает в параметр data.
	Write – метод	Записывает массив байтов в Modbus устройство.
	ClearReadBuffer – метод	Очищает буфер чтения.
	ClearWriteBuffer – метод	Очищает буфер записи.
С	Proryv.Script.Modbus – класс	Класс для работы с Modbus устройствами.
К	Modbus – конструктор	Инициализирует новый экземпляр класса Script.Modbus.
	RemotePort – свойство	Возвращает класс который работает с некоторым удаленным портом.
	MaxTrySend – свойство	Возвращает и устанавливает максимальное количество попыток связи с устройством.
	MaxReadGapRegisters – свойство	Возвращает и устанавливает максимальное количество промежуточных регистров.
	Read – метод	Считывает данные из регистра Modbus устройства с отображением запросов/ответов.
	Read – метод	Считывает данные из регистра Modbus устройства с возможностью управления отображением запросов/ответов.
	Read – метод	Считывает данные из нескольких регистров Modbus устройства с отображением запросов/ответов.
	Read – метод	Считывает данные из нескольких регистров Modbus устройства с возможностью управления отображением запросов/ответов.
	Send – метод	Посылает команду Modbus устройству.

Тип	Наименование	Назначение
	Write – метод	Записывает данные в регистр Modbus устройства с отображением запросов/ответов.
	Write – метод	Записывает данные в регистр Modbus устройства с возможностью управления отображением запросов/ответов.
	Write – метод	Записывает данные в несколько регистров Modbus устройства с отображением запросов/ответов.
	Write – метод	Записывает данные в несколько регистров Modbus устройства с возможностью управления отображением запросов/ответов.
П	Proryv.Script.StateTS – перечисление	Перечисление состояние ТС: Off = 0 (ТС сброшен) On = 1 (ТС установлен)
П	Proryv.Script.StateTU – перечисление	Перечисление состояние ТУ: Off = 0 (сбросить ТУ) On = 1 (установить ТУ)
С	Proryv.Script.UK – класс	Статический класс для работы со встроенным коммутационным устройством (УК).
	Open – метод	Открывает сеанс работы с УК.
	Close – метод	Закрывает сеанс работы с УК.
	GetTS – метод	Возвращает значение для указанного номера дискретного входа (с единицы) в параметре out StateTS value.
	GetTSMask – метод	Возвращает значение маски ТС в параметре.
	GetTIR – метод	Возвращает значение ТИР для указанного номера дискретного входа (с единицы) в параметре out UInt16 value.
	GetTIT – метод	Возвращает значение ТИТ для указанного номера аналогового входа (с единицы) в параметре out UInt16 value.
	SetTU – метод	Устанавливает значение выходного сигнала ТУ для указанного номера выхода телеуправления.
	SetTU – метод	Устанавливает значение выходного сигнала ТУ для указанного номера выхода телеуправления с удержанием сигнала.
	SetTUCyclic – метод	Включает циклическое переключение ТУ для указанного номера выхода телеуправления.
	SetVoltageScale – метод	Устанавливает параметры масштабирования напряжения для указанного номера аналогового входа (с единицы).
	SetCurrentScale – метод	Устанавливает параметры масштабирования тока для указанного номера аналогового входа (с единицы).
	GetVoltage – метод	Возвращает значение напряжения в параметре out double value.
	GetCurrent – метод	Возвращает значение тока в параметре out double value.
	GetTU – метод	Возвращает состояние канала ТУ дял указанного номера входа (с единицы) в параметре value.
С	Proryv.Script.ScriptDeviceParams – класс	Класс содержит параметры устройства: идентификатор, строка инициализации. Используется для работы с системой Телескоп+.
С	Proryv.Script.ScriptEvents – класс	Класс для проверки данных, поступающих от системы Телескоп+.
К	ScriptEvents – конструктор	Инициализирует новый экземпляр класса Script.ScriptEvents.
	IsInit – метод	Проверяет наличие инициализации.
	IsCommand – метод	Проверяеет наличие команды для скрипта.
С	Proryv.Script.ScriptCommands – класс	Класс для отправки команды в систему Телескоп+.
К	ScriptCommands – конструктор	Инициализирует новый экземпляр класса Script.ScriptCommands.
	SendCommand – метод	Отправляет команду в систему Телескоп+.

Классы и методы

Proryv.Script.Output – класс

Статический класс для вывода сообщений о выполнении скрипта в файл *.log. Файл размещен в одном каталоге со скриптом. Каталог по умолчанию:
0:/NANDFlash/TK16L/DATA/Init/Script/Projects/ProjectName/
где ProjectName – название проекта, в который включен скрипт.

Output.WriteLine – метод

Выводит сообщение в лог-файл.

Синтаксис

public void Output.WriteLine(string message)

Параметры

message (тип: string) текст сообщения.

Пример
Output.WriteLine(“Start control”);

Output.WriteLine – метод

Выводит сообщение со списком параметров в лог-файл.

Синтаксис

public void Output.WriteLine(string message, params object[] list)

Параметры

message (тип: string) текст сообщения.

list (тип: params object[]) список параметров.

Пример
Output.WriteLine(“Message, Param1={0}, Param2={1}”, 123, 6.78);

Output.WriteWarning – метод

Выводит предупреждение в лог-файл.

Синтаксис

public void Output.WriteWarning(string message)

Параметры

message (тип: string) текст сообщения.

Пример
Output. WriteWarning (“Warning message”);

Output.WriteWarning – метод

Выводит предупреждение со списком параметров в лог-файл.

Синтаксис
public void Output.WriteWarning(string message, params object[] list)

Параметры

message (тип: string) текст сообщения.

list (тип: params object[]) список параметров.

Пример
Output. WriteWarning (“Warning message, Param1={0}, Param2={1}”, 123, 6.78);

Output.WriteError – метод

Выводит предупреждение в лог-файл.

Синтаксис
public void Output.WriteError(string message)

Параметры

message (тип: string) текст сообщения.

Пример
Output. WriteError (“Error message”);

Output.WriteError – метод

Выводит предупреждение в лог-файл.

Синтаксис
public void Output.WriteError(string message, params object[] list)

Параметры

message (тип: string) текст сообщения.

list (тип: params object[]) список параметров.

Пример
Output. WriteError (“Error message, Param1={0}, Param2={1}”, 123, 6.78);

Output.WriteException – метод

Выводит предупреждение в лог-файл.

Синтаксис
public void Output.WriteException(string message, Exception ex)

Параметры

message (тип: string) текст сообщения.

ex (тип: Exception) исключение.

Пример
Output.WriteException (“Exception message”, ex);

Output.Format – метод

Выполняет подстановку параметров, заданных в списке, в строку.

Синтаксис
public string Output.Format(string message, params object[] list)

Параметры

message (тип: string) строка.

list (тип: params object[]) список аргументов.

Пример
string message = Output.Format ("Value1={0}, Value2={1}", 12, 45.6);
В результате выполнения возвращается строка: Value1=12, Value2=45.6

Proryv.Script.Array – класс

Статический класс для обработки данных в массивах.

Array.Sort - метод

Сортирует массив

Синтаксис
public void Sort(System.Array array, int index, int length)

Параметры

array (тип: System.Array) массив.

index (тип: int) начальный элемент массива.

length (тип: int) количество элементов массива.

Пример
byte[] data = { 0x05, 0x03, 0x07};
Array.Sort(data, 0, data,Length);

Proryv.Script.DataUtils – класс

Статический класс для обработки данных. Используется, в частности, при обмене данными с системой Телескоп+.

DataUtils.StrHexToArray – метод

Преобразует hex строку в массив байтов.

Синтаксис
public byte[] DataUtils.StrHexToArray(string str)

Параметры
str (тип: string) hex строка

Возвращаемое значение
Массив байтов, преобразованный из строки (null если str=null)

DataUtils.DataToStrHex – метод

Преобразует массив байтов в hex строку.

Синтаксис
public string DataUtils.DataToStrHex(byte[] data)

Параметры
data (тип: byte[]) входной массив байтов

Возвращаемое значение
hex строка, преобразованная из массива

DataUtils.CRC16 – метод

Подсчитывает контрольную сумму для фрагмента массива.

Синтаксис
public UInt16 DataUtils.CRC16(byte[] buffer, int offset, int length)

Параметры
buffer (тип: byte[]) массив байтов
offset (тип: int) сдвиг в массиве байтов до целевого фрагмента массива
length (тип: int) количество байтов во фрагменте массива

Возвращаемое значение
Контрольная сумма CRC16 для фрагмента массива

DataUtils.IsSetBit – метод

Проверяет установку бита в битовой последовательности.

Синтаксис
public bool DataUtils.IsSetBit(int mask, int index)

Параметры
mask (тип: int) маска (битовая последовательность)
index (тип: int) номер бита от младшего бита

Возвращаемое значение
Признак установки бита:
true - бит установлен
false - не установлен

DataUtils.IsEqual – метод

Сравнивает массивы.

Синтаксис
public bool DataUtils.IsEqual(byte[] array1, byte[] array2)

Параметры
array1 (тип: byte[]) первый массив байтов
array2 (тип: byte) второй массив байтов

Возвращаемое значение
Признак равенства массивов:
true – массивы равны
false – массивы различны

DataUtils.IpAddressToData – метод

Преобразует IP адрес к массиву байтов и записывает его в заданную позицию исходного массива.

Синтаксис
public void DataUtils.IpAddressToData(string ipAddress, ref byte[] data, ref int offset)

Параметры
ipAddress (тип: string) строка IP адреса
data (тип: ref byte[]) массив байтов
offset (тип: ref int) сдвиг в массиве байтов (позиция, с которой начинается запись массива IP адреса)

DataUtils.DataToIpAddress – метод

Преобразует фрагмент массива байтов в IP адрес.

Синтаксис
public bool DataUtils.DataToIpAddress(byte[] data, ref int offset, out string ipAddress)

Параметры
data (тип: byte[]) массив байтов
offset (тип: ref int) сдвиг в массиве байтов (позиция, с которой начинается считывание массива IP адреса)
ipAddress (тип: out string) строка IP адреса

DataUtils.GetData – метод

Возвращает фрагмент массива.

Синтаксис
public byte[] DataUtils.GetData(byte[] data, int index, int count)

Параметры
data (тип: byte[]) массив байтов
index (тип: int) сдвиг в массиве байтов (позиция начала фрагмента)
count (тип: int) количество байтов во фрагменте

Возвращаемое значение
Фрагмент массива

DataUtils.ByteToData – метод

Преобразует число типа Byte в массив байтов.

Синтаксис
public void DataUtils.ByteToData(byte value, ref byte[] data, ref int offset)

Параметры
value (тип: byte) число
data (тип: ref byte[]) массив байтов

offset (тип: ref int) сдвиг в массиве байтов (позиция, с которой начинается запись массива, преобразованного из числа)

DataUtils.DataToByte – метод

Преобразует массив байтов в число типа Byte.

Синтаксис
public bool DataUtils.DataToByte(byte[] data, ref int offset, out byte value)

Параметры
data (тип: byte[]) массив байтов
offset (тип: ref int) сдвиг в массиве байтов (позиция, с которой начинается считывание фрагмента массива числа)
value (тип: out byte) число

Возвращаемое значение
Признак выполнения преобразования:
true – выполнено
false – ошибка

DataUtils.SByteToData – метод

Преобразует число типа SByte в массив байтов. SByte представляет собой 8-битовое целое число со знаком.

Синтаксис
public void DataUtils.SByteToData(sbyte value, ref byte[] data, ref int offset)

Параметры
value (тип: sbyte) число
data (тип: ref byte[]) массив байтов
offset (тип: ref int) сдвиг в массиве байтов (позиция, с которой начинается запись массива, преобразованного из числа)

DataUtils.DataToSByte – метод

Преобразует массив байтов в число типа SByte.

Синтаксис
public bool DataUtils.DataToSByte(byte[] data, ref int offset, out sbyte value)

Параметры
data (тип: byte[]) массив байтов
offset (тип: ref int) сдвиг в массиве байтов (позиция, с которой начинается считывание фрагмента массива числа)
value (тип: out sbyte) число

Возвращаемое значение
Признак выполнения преобразования:
true – выполнено
false – ошибка

DataUtils.UInt16ToData – метод

Преобразует число типа UInt16 в массив байтов.

Синтаксис
public void DataUtils.UInt16ToData(UInt16 value, ref byte[] data, ref int offset, bool firstHigh)

Параметры
value (тип: UInt16) число
data (тип: ref byte[]) массив байтов
offset (тип: ref int) сдвиг в массиве байтов (позиция, с которой начинается запись преобразованного массива числа в массив)
firstHigh (тип: bool) порядок записи преобразованного массива:
true – начинать со старшего байта
false – начинать с младшего байта

DataUtils.DataToUInt16 – метод

Преобразует массив байтов в число типа UInt16.

Синтаксис
public bool DataUtils.DataToUInt16(byte[] data, ref int offset, bool firstHigh, out UInt16 value)

Параметры
data (тип: byte[]) массив байтов
offset (тип: ref int) сдвиг в массиве байтов (позиция, с которой начинается считывание фрагмента массива)
firstHigh (тип: bool) порядок считывания массива:
true – начинать со старшего байта
false – начинать с младшего байта
value (тип: out UInt16) число

Возвращаемое значение
Признак выполнения преобразования:
true – выполнено
false – ошибка

DataUtils.Int16ToData – метод

Преобразует число типа Int16 в массив байтов.

Синтаксис
public void DataUtils.Int16ToData(Int16 value, ref byte[] data, ref int offset, bool firstHigh)

Параметры
value (тип: Int16) число
data (тип: ref byte[]) массив байтов
offset (тип: ref int) сдвиг в массиве байтов (позиция, с которой начинается запись преобразованного массива числа в массив)
firstHigh (тип: bool) порядок записи преобразованного массива:
true – начинать со старшего байта
false – начинать с младшего байта

DataUtils.DataToInt16 – метод

Преобразует массив байтов в число типа Int16.

Синтаксис
public bool DataUtils.DataToInt16(byte[] data, ref int offset, bool firstHigh, out Int16 value)

Параметры
data (тип: byte[]) массив байтов
offset (тип: ref int) сдвиг в массиве байтов (позиция, с которой начинается считывание фрагмента массива)
firstHigh (тип: bool) порядок считывания массива:
true – начинать со старшего байта
false – начинать с младшего байта

value (тип: out Int16) число

Возвращаемое значение
Признак выполнения преобразования:
true – выполнено
false – ошибка

DataUtils.UInt32ToData – метод

Преобразует число типа UInt32 в массив байтов.

Синтаксис
public void DataUtils.UInt32ToData(UInt32 value, ref byte[] data, ref int offset, bool firstHigh)

Параметры
value (тип: UInt32) число
data (тип: ref byte[]) массив байтов, начиная с младшего байта
offset (тип: ref int) сдвиг в массиве байтов (позиция, с которой начинается запись преобразованного массива числа в массив)
firstHigh (тип: bool) порядок записи преобразованного массива:
true – начинать со старшего байта
false – начинать с младшего байта

DataUtils.DataToUInt32 – метод

Преобразует массив байтов в число типа UInt32.

Синтаксис
public bool DataUtils.DataToUInt32(byte[] data, ref int offset, bool firstHigh, out UInt32 value)

Параметры
data (тип: byte[]) массив байтов, начиная с младшего байта
offset (тип: ref int) сдвиг в массиве байтов (позиция, с которой начинается считывание фрагмента массива)
firstHigh (тип: bool) порядок считывания массива:
true – начинать со старшего байта
false – начинать с младшего байта
value (тип: out UInt32) число

Возвращаемое значение
Признак выполнения преобразования:
true – выполнено
false – ошибка

DataUtils.Int32ToData – метод

Преобразует число типа Int32 в массив байтов.

Синтаксис
public void DataUtils.Int32ToData(Int32 value, ref byte[] data, ref int offset, bool firstHigh)

Параметры
value (тип: Int32) число
data (тип: ref byte[]) массив байтов
offset (тип: ref int) сдвиг в массиве байтов (позиция, с которой начинается запись преобразованного массива числа в массив)
firstHigh (тип: bool) порядок записи преобразованного массива:
true – начинать со старшего байта
false – начинать с младшего байта

DataUtils.DataToInt32 – метод

Преобразует массив байтов в число типа Int32.

Синтаксис
public bool DataUtils.DataToInt32(byte[] data, ref int offset, bool firstHigh, out Int32 value)

Параметры
data (тип: byte[]) массив байтов, начиная с младшего байта
offset (тип: ref int) сдвиг в массиве байтов (позиция, с которой начинается считывание фрагмента массива)
firstHigh (тип: bool) порядок считывания массива:
true – начинать со старшего байта
false – начинать с младшего байта
value (тип: out Int32) число

Возвращаемое значение
Признак выполнения преобразования:
true – выполнено
false – ошибка

DataUtils.UInt64ToData – метод

Преобразует число типа UInt64 в массив байтов.

Синтаксис
public void DataUtils.UInt64ToData(UInt64 value, ref byte[] data, ref int offset, bool firstHigh)

Параметры
value (тип: UInt64) число
data (тип: ref byte[]) массив байтов
offset (тип: ref int) сдвиг в массиве байтов (позиция, с которой начинается запись преобразованного массива числа в массив)
firstHigh (тип: bool) порядок записи преобразованного массива:
true – начинать со старшего байта
false – начинать с младшего байта

DataUtils.DataToUInt64 – метод

Преобразует массив байтов в число типа UInt64.

Синтаксис
public bool DataUtils.DataToUInt64(byte[] data, ref int offset, bool firstHigh, out UInt64 value)

Параметры
data (тип: byte[]) массив байтов, начиная с младшего байта
offset (тип: ref int) сдвиг в массиве байтов (позиция, с которой начинается считывание фрагмента массива)
firstHigh (тип: bool) порядок считывания массива:
true – начинать со старшего байта
false – начинать с младшего байта
value (тип: out UInt64) число

Возвращаемое значение
Признак выполнения преобразования:
true – выполнено
false – ошибка

DataUtils.Int64ToData – метод

Преобразует число типа Int64 в массив байтов.

Синтаксис
public void DataUtils.Int64ToData(Int64 value, ref byte[] data, ref int offset, bool firstHigh)

Параметры
value (тип: Int64) число
data (тип: ref byte[]) массив байтов
offset (тип: ref int) сдвиг в массиве байтов (позиция, с которой начинается запись преобразованного массива числа в массив)
firstHigh (тип: bool) порядок записи преобразованного массива:
true – начинать со старшего байта
false – начинать с младшего байта

DataUtils.DataToInt64 – метод

Преобразует массив байтов в число типа Int64.

Синтаксис
public bool DataUtils.DataToInt64(byte[] data, ref int offset, bool firstHigh, out Int64 value)

Параметры
data (тип: byte[]) массив байтов
offset (тип: ref int) сдвиг в массиве байтов (позиция, с которой начинается считывание фрагмента массива)
firstHigh (тип: bool) порядок считывания массива:
true – начинать со старшего байта
false – начинать с младшего байта
value (тип: out Int64) число

Возвращаемое значение
Признак выполнения преобразования:
true – выполнено
false – ошибка

DataUtils.BoolToData – метод

Преобразует число типа Bool в массив байтов.

Синтаксис
public void DataUtils.BoolToData(bool value, ref byte[] data, ref int offset)

Параметры
value (тип: bool) число
data (тип: ref byte[]) массив байтов
offset (тип: ref int) сдвиг в массиве байтов (позиция, с которой начинается запись преобразованного массива числа в массив)

DataUtils.DataToBool – метод

Преобразует массив байтов в число типа Bool.

Синтаксис
public bool DataUtils.DataToBool(byte[] data, ref int offset, out bool value)

Параметры
data (тип: byte[]) массив байтов
offset (тип: ref int) сдвиг в массиве байтов (позиция, с которой начинается считывание фрагмента массива)
value (тип: out bool) число

Возвращаемое значение

Признак выполнения преобразования:
true – выполнено
false – ошибка

DataUtils.ArrayToData – метод

Копирует массив байтов.

Синтаксис
public void DataUtils.ArrayToData(byte[] value, int valueOffset,
ref byte[] data, ref int dataOffset, int count)

Параметры
value (тип: byte[]) массив байтов - источник
valueOffset (тип: int) сдвиг в массиве байтов - источнике (позиция, с которой начинается считывание фрагмента массива)
data (тип: ref byte[]) массив байтов - приемник
dataOffset (тип: ref int) сдвиг в массиве байтов (позиция, с которой начинается запись фрагмента массива)
count (тип: int) количество копируемых байтов

DataUtils.DataToArray – метод

Копирует массива байтов. По функциональности метод полностью совпадает с DataUtils.ArrayToData.

Синтаксис
public bool DataUtils.DataToArray(byte[] data, ref int dataOffset,
ref byte[] value, int valueOffset, int count)

Параметры
data (тип: byte[]) массив байтов - источник
dataOffset (тип: ref int) сдвиг в массиве байтов (позиция, с которой начинается считывание фрагмента массива)
value (тип: ref byte[]) массив байтов - приемник
valueOffset (тип: int) сдвиг в массиве байтов - приемнике (позиция, с которой начинается запись фрагмента массива)
count (тип: int) количество копируемых байтов

DataUtils.WideStringToData – метод

Преобразует строку (Unicode) в массив байтов. Первые два байта - длина строки.

Синтаксис
public void DataUtils.WideStringToData(string value, ref byte[] data, ref int offset, bool firstHigh)

Параметры
value (тип: string) строка (Unicode)
data (тип: ref byte[]) массив байтов
offset (тип: ref int) сдвиг в массиве байтов (позиция, с которой начинается запись преобразованной строки в массив)
firstHigh (тип: bool) порядок записи байтов для длины строки:
true – начинать со старшего байта
false – начинать с младшего байта

DataUtils.DataToWideString – метод

Преобразует массив байтов в строку (Unicode). Первые два байта - длина строки.

Синтаксис

public bool DataUtils.DataToWideString(byte[] data, ref int offset, bool firstHigh, out string value)

Параметры
data (тип: byte[]) массив байтов
offset (тип: ref int) сдвиг в массиве байтов (позиция, с которой начинается считывание фрагмента массива)
firstHigh (тип: bool) порядок считывания байтов для длины строки:
true – начинать со старшего байта
false – начинать с младшего байта
value (тип: out string) строка (Unicode)

Возвращаемое значение
Признак выполнения преобразования:
true – выполнено
false – ошибка

DataUtils.DataToSingle – метод

Преобразует массив байтов в число типа Single.

Синтаксис
public bool DataUtils.DataToSingle(byte[] data, ref int offset, bool firstHigh, out Single value)

Параметры
data (тип: byte[]) массив байтов
offset (тип: ref int) сдвиг в массиве байтов (позиция, с которой начинается считывание фрагмента массива)
firstHigh (тип: bool) порядок считывания байтов:
true – начинать со старшего байта
false – начинать с младшего байта
value (тип: out Single) число

Возвращаемое значение
Признак выполнения преобразования:
true – выполнено
false – ошибка

DataUtils.SingleToData – метод

Преобразует число типа Single в массив байтов.

Синтаксис
public void DataUtils.SingleToData(Single value, ref byte[] data, ref int offset, bool firstHigh)

Параметры
value (тип: Single) число
data (тип: ref byte[]) массив байтов
offset (тип: ref int) сдвиг в массиве байтов (позиция, с которой начинается запись преобразованного массива числа в массив)
firstHigh (тип: bool) порядок записи байтов:
true – начинать со старшего байта
false – начинать с младшего байта

DataUtils.DataToDouble – метод

Преобразует массив байтов в число типа Double.

Синтаксис
public bool DataUtils.DataToDouble(byte[] data, ref int offset, bool firstHigh, out Double value)

Параметры

data (тип: byte[]) массив байтов
offset (тип: ref int) сдвиг в массиве байтов (позиция, с которой начинается считывание фрагмента массива)
firstHigh (тип: bool) порядок считывания байтов:
true – начинать со старшего байта
false – начинать с младшего байта
value (тип: out Double) число

Возвращаемое значение
Признак выполнения преобразования:
true – выполнено
false – ошибка

DataUtils.DoubleToData – метод

Преобразует число типа Double в массив байтов.

Синтаксис
public void DataUtils.DoubleToData(Double value, ref byte[] data, ref int offset, bool firstHigh)

Параметры
value (тип: Double) число
data (тип: ref byte[]) массив байтов
offset (тип: ref int) сдвиг в массиве байтов (позиция, с которой начинается запись преобразованного массива числа в массив)
firstHigh (тип: bool) порядок записи байтов:
true – начинать со старшего байта
false – начинать с младшего байта

DataUtils.DateTimeToData – метод

Преобразует дату и время в массив байтов (8 байт, год от 2000).

Синтаксис
public void DataUtils.DateTimeToData(DateTime? value, ref byte[] data, ref int offset, bool firstHigh)

Параметры
value (тип: DateTime?) дата и время
data (тип: ref byte[]) массив байтов
offset (тип: ref int) сдвиг в массиве байтов (позиция, с которой начинается запись фрагмента массива)
firstHigh (тип: bool) порядок считывания байтов:
true – начинать со старшего байта
false – начинать с младшего байта

DataUtils.DataToDateTime – метод

Преобразует массив байтов в переменную типа DateTime (8 байт, год от 2000).

Синтаксис
public bool DataUtils.DataToDateTime(byte[] data, ref int offset, bool firstHigh, out DateTime? value)

Параметры
data (тип: byte[]) массив байтов
offset (тип: ref int) сдвиг в массиве байтов (позиция, с которой начинается считывание фрагмента массива)
firstHigh (тип: bool) порядок считывания байтов:
true – начинать со старшего байта
false – начинать с младшего байта
value (тип: out DateTime?) дата и время

Возвращаемое значение
Признак выполнения преобразования:
true – выполнено
false – ошибка

Пример

```
byte[] data = { 0x05, 0x03, 0x07, 0x06};
int offset = 0;
byte valueByte = 0;
DataUtils.DataToByte(data, ref offset, out byte value)
Uint16 value = Array.Sort(data, 0, data,Length);
```

Proryv.Script.Bool – перечисление

False = 0
True = 1

Proryv.Script.Stopwatch – класс

Класс для измерения затраченного времени.

Stopwatch – конструктор

Инициализирует новый экземпляр класса Stopwatch.

Reset – метод

Останавливает измерение интервала времени и обнуляет затраченное время.

Синтаксис

```
public void Reset()
```

Start – метод

Запускает измерение интервала времени.

Синтаксис

```
public void Start()
```

Restart – метод

Останавливает измерение затраченного времени, обнуляет затраченное время и начинает новое измерение интервала времени.

Синтаксис

```
public void Restart()
```

IsRunning – свойство

Возвращает признак запуска измерения времени.

Синтаксис

```
public bool IsRunning { get; }
```

Stop – метод

Останавливает измерение затраченного времени.

Синтаксис

```
public void Stop()
```

ElapsedMilliseconds – свойство

Возвращает значение затраченного времени в миллисекундах.

Синтаксис

```
public uint ElapsedMilliseconds { get; }
```

Пример

```
Stopwatch measure = new Stopwatch();
measure.Restart();
…..
Output.WriteLine(string.Format(“Elapsed time = {0}”,
        measure.ElapsedMilliseconds));
```

Proryv.Script.Watchdog – класс

Класс для реализация процесса типа watchdog: контроль зависания контроллера, перезагрузка контроллера при зависании.

Watchdog – конструктор

Инициализирует новый экземпляр класса Watchdog.

Синтаксис

```
public Watchdog(string name, int period)
```

Параметры

name (тип: string) уникальное имя watchdog
period (тип: int) период обновления, мс

Start – метод

Запускает таймер Watchdog.

Синтаксис

```
public bool Start()
```

Stop – метод

Останавливает таймер Watchdog.

Синтаксис

```
public bool Stop()
```

Refresh – метод

Обновляет таймер Watchdog.

Синтаксис

public bool Refresh()

Пример

```
Watchdog  watchdog = new  Watchdog(“MyWatchdog”, 60000);
watchdog.Start();
while (true)
{
        watchdog.Refresh();
        …
}
        watchdog.Stop();
```

Proryv.Script.RemotePort – класс

Класс для работы по линиям RS485/RS232.

RemotePort – конструктор

Инициализирует новый экземпляр класса RemotePort.

Синтаксис

```
public RemotePort(string ipAddress, int tcpPort)
```

Параметры

ipAddress (тип: string) IP адрес удаленного порта. Можно указывать IP адрес любого контроллера TK16L. Чаще всего используется контроллер с IP=127.0.0.1 (текущий).

tcpPort (тип: int) TCP порт удаленного порта: определяет номер линии

- 1000 - 1 линия RS-485
- 2000 - 2 линия RS-485
- 3000 - 3 линия RS-485
- 4000 - 4 линия RS-485
- 9000 - 1 линия RS-232
- 8000 - 2 линия RS-232

Dispose – метод

Освобождает ресурсы.

Синтаксис

```
public void Dispose()
```

Open – метод

Открывает порт.

Синтаксис

```
public bool Open()
```

Возвращаемое значение

Признак выполнения:

- true – выполнено
- false – ошибка

Close – метод

Закрывает порт.

Синтаксис
public void Close()

IsOpen – свойство

Возвращает признак открытия порта.

Синтаксис
public bool IsOpen { get; }

IPAddress – свойство

Возвращает IP адрес порта.

Синтаксис
public string IPAddress { get; }

TCPPort – свойство

Возвращает номер TCP порта.

Синтаксис
public int TCPPort { get; }

BaudRate – свойство

Возвращает и устанавливает скорость обмена, например 9600.

Синтаксис
public int BaudRate { get; set; }

DataBits – свойство

Возвращает и устанавливает количество бит данных (8 или 7 бит).

Синтаксис
public byte DataBits { get; set; }

System.IO.Ports.StopBits Stopbits – свойство

Возвращает и устанавливает длину стоп-бита (StopBits.None, StopBits.One, StopBits.Two, StopBits.OnePointFive).

Синтаксис
public System.IO.Ports.StopBits Stopbits { get; set; }

System.IO.Ports.Parity Parity – свойство

Возвращает и устанавливает способ контроля четности (Parity.None, Parity.Odd, Parity.Even, Parity.Mark, Parity.Space).

Синтаксис
public System.IO.Ports.Parity Parity { get; set; }

ReadTimeoutInterval – свойство

Возвращает и устанавливает таймаут чтения, мс (межсимвольный).

Синтаксис

```
public int ReadTimeoutInterval { get; set; }
```

ReadTimeoutMultiplier – свойство

Возвращает и устанавливает таймаут чтения, мс (множитель).

Синтаксис

```
public int ReadTimeoutConstant { get; set; }
```

ReadTimeoutConstant – свойство

Возвращает и устанавливает таймаут чтения, мс (константа).

Синтаксис

```
public int ReadTimeoutConstant { get; set; }
```

WriteTimeoutMultiplier – свойство

Возвращает и устанавливает таймаут записи, мс (множитель).

Синтаксис

```
public int WriteTimeoutMultiplier { get; set; }
```

WriteTimeoutConstant – свойство

Возвращает и устанавливает таймаут записи, мс (константа).

Синтаксис

```
public int WriteTimeoutConstant { get; set; }
```

Read – метод

Считывает массив байтов из Modbus устройства, помещает в параметр data.

Синтаксис

```
public bool Read(byte[] data, int index, int count, out int readed)
```

Параметры

data (тип: byte[]) массив данных.
index (тип: int) начальный элемент массива
count (тип: int) количество байт для считывания
readed (тип: out int)

Возвращаемое значение

Признак выполнения считывания:

true – выполнено
false – ошибка

Write – метод

Записывает массив байтов в Modbus устройство.

Синтаксис

public bool Write(byte[] data, int index, int count)

Параметры
data (тип: byte[]) массив данных.
index (тип: int) начальный элемент массива
count (тип: int) количество байт для записи

Возвращаемое значение
Признак выполнения записи:
true – выполнено
false – ошибка

ClearReadBuffer – метод

Очищает буфер чтения.

Синтаксис
public void ClearReadBuffer()

ClearWriteBuffer – метод

Очищает буфер записи.

Синтаксис
public void ClearWriteBuffer()

Пример

```
using System;
using System.IO.Ports;
using System.Threading;
using Proryv.Script;

////////////////////////////////////////////////
// Пример взаимодействия с по линии RS-485/RS-232
class TestRemotePort
{
   ////////////////////////////////////////////////
   // Пример взаимодействия с Modbus устройством по линии RS-485/RS-232
   public static void Main()
   {
      Output.WriteLine("Test RemotePort");

      // Удаленный порт
      // IP адрес - текущий (можно указывать IP адрес любого другого контроллера TK16L),
      // TCP порт - определяет номер линии
      //  (1000 - 1 линия RS-485, 2000 - 2 линия RS-485, 3000 - 3 линия RS-485, 4000 - 4 линия RS-485,
      //   9000 - 1 линия RS-232, 8000 - 2 линия RS-232)
      RemotePort remotePort = new RemotePort("127.0.0.1", 3000);
      // Параметры порта:
      //   BaudRate - скорость COM порта, бит/сек
      //   DataBits - кол-во битов данных
      //   Stopbits - кол-во стоп-битов (StopBits.None, StopBits.One, StopBits.Two,
StopBits.OnePointFive)
      //   Parity - четность (Parity.None, Parity.Odd, Parity.Even, Parity.Mark, Parity.Space)
      //   ReadTimeoutInterval - таймаут чтения (межсимвольный), мс
      //   ReadTimeoutMultiplier - таймаут чтения (множитель), мс
      //   ReadTimeoutConstant - таймаут чтения (константа), мс
      remotePort.BaudRate = 9600;
```

```
        remotePort.DataBits = 8;
        remotePort.Stopbits = StopBits.One;
        remotePort.Parity = Parity.None;
        remotePort.ReadTimeoutInterval = 4;
        remotePort.ReadTimeoutMultiplier = 0;
        remotePort.ReadTimeoutConstant = 500;

        // Открыть порт
        if (remotePort.Open() == false)
        {
           Output.WriteLine("Open - Fail");
           remotePort.Dispose();
           return;
        }
        Output.WriteLine("Open - OK");

        // Послать запрос
        byte[] send = { 0xEE, 0x01, 0x00, 0x00, 0x00, 0x01, 0x20, 0x38, 0x14 };
        // send - массив байтов данных для посылки, 0 - индекс начала данных, send.Length - кол-во
байтов данных
        if (remotePort.Write(send, 0, send.Length) == false)
        {
           Output.WriteLine("Write - Fail");
           remotePort.Dispose();
           return;
        }
        Output.WriteLine(string.Format("Write - OK, Send=({0})",
          ArrayToStr(send, 0, send.Length)));

        // Получить ответ
        byte[] answer = new byte[10];
        int readed = 0;
        // answer - массив байтов данных для ответа, 0 - индекс начала данных, answer.Length - кол-во
байтов данных
        //   readed - кол-во реально считанных байтов
        if (remotePort.Read(answer, 0, answer.Length, out readed) == false)
        {
           Output.WriteLine("Read - Fail");
           remotePort.Dispose();
           return;
        }
        Output.WriteLine(string.Format("Read - OK, Readed={0}, Answer=({1})",
          readed, ArrayToStr(answer, 0, readed)));

        remotePort.Dispose();
        Output.WriteLine("Test RemotePort Complete");
    }

    ////////////////////////////////////////////////
    // Строковое представление массива байт
    private static string ArrayToStr(byte[] array, int index, int count)
    {
        string result = string.Empty;
        if (array == null) return result;
        for (int i = index; i < index + count; i++)
        {
           if (i >= array.Length) break;
           if (i > 0) result = result + ", ";
           result = result + array[i].ToString("X2");
        }
```

```
            return result;
        }
    }
```

Proryv.Script.Modbus – класс

Класс для работы с Modbus устройствами.

Modbus – конструктор

Инициализирует новый экземпляр класса Script.Modbus.

Синтаксис

```
public Modbus(RemotePort remotePort, Output output)
```

Параметры

remotePort удаленный порт.
output вывод отладочной информации.

RemotePort – свойство

Возвращает класс, который работает с некоторым удаленным портом.

Синтаксис

```
public RemotePort RemotePort { get; }
```

MaxTrySend – свойство

Возвращает и устанавливает максимальное количество попыток связи с устройством.

Синтаксис

```
public int MaxTrySend { get; set; }
```

MaxReadGapRegisters – свойство

Возвращает и устанавливает максимальное количество промежуточных регистров. Используется для оптимизации количества запросов. Например, необходимо считать данные по двум адресам 0x0000 и 0x0005. Если MaxReadGapRegisters=0, то будет отправлено два запроса. Если MaxReadGapRegisters=4, то будет отправлен один запрос. Для ускорения опроса рекомендуется минимизировать количество запросов.

Синтаксис

```
public int MaxReadGapRegisters { get; set; }
```

Read – метод

Считывает данные из регистра Modbus устройства с отображением запросов/ответов.

Синтаксис

```
public bool Read(UInt16 deviceAddress, UInt16 itemAddress, out UInt16
itemValue)
```

Параметры

deviceAddress (тип: UInt16) Modbus адрес устройства.
itemAddress (тип: UInt16) Modbus адрес (стандартный) регистра.
itemValue (тип: out UInt16) значение регистра. Для битовых значений:
0-false

1-true

Возвращаемое значение
Признак выполнения считывания:
true – выполнено
false – ошибка

Read – метод

Считывает данные из регистра Modbus устройства с возможностью управления отображением запросов/ответов.

Синтаксис
public bool Read(UInt16 deviceAddress, UInt16 itemAddress, out UInt16 itemValue, bool showTrace)

Параметры
deviceAddress (тип: UInt16) Modbus адрес устройства.
itemAddress (тип: UInt16) Modbus адрес (стандартный) регистра.
itemValue (тип: out UInt16) значение регистра. Для битовых значений:
0-false
1-true
showTrace (тип: bool) управление отображением запросов/ответов
true – отображать запросы/ответы
false – отображать запросы для которых ответ изменился

Возвращаемое значение
Признак выполнения считывания:
true – выполнено
false – ошибка

Read – метод

Считывает данные из нескольких регистров Modbus устройства с отображением запросов/ответов.

Синтаксис
public bool Read(UInt16 deviceAddress, UInt16[] itemAddress, UInt16[] itemValue, int index, int count)

Параметры
deviceAddress (тип: UInt16) Modbus адрес устройства.
itemAddress (тип: UInt16[]) список Modbus адресов (стандартных) регистра.
itemValue (тип: out UInt16[]) список значений регистров. Для битовых значений:
0-false
1-true
index (тип: int) начальный элемент списка
count (тип: int) длина списка

Возвращаемое значение
Признак выполнения считывания:
true – выполнено
false – ошибка

Read – метод

Считывает данные из нескольких регистров Modbus устройства с возможностью управления отображением запросов/ответов.

Синтаксис
public bool Read(UInt16 deviceAddress, UInt16[] itemAddress, UInt16[] itemValue, int index, int count,
bool showTrace)

Параметры
deviceAddress (тип: UInt16) Modbus адрес устройства.
itemAddress (тип: UInt16[]) список Modbus адресов (стандартных) регистра.
itemValue (тип: out UInt16[]) список значений регистров. Для битовых значений:
0-false
1-true
index (тип: int) начальный элемент списка
count (тип: int) длина списка
showTrace (тип: bool) управление отображением запросов/ответов
true – отображать запросы/ответы
false – отображать запросы для которых ответ изменился

Возвращаемое значение
Признак выполнения считывания:
true – выполнено
false – ошибка

Send – метод

Посылает команду Modbus устройству.

Синтаксис

```
public bool Send(byte[] command, out byte[] answer, bool showTrace)
```

Параметры
command (тип: byte[]) команда (без CRC).
answer (тип: out byte[]) ответ (без CRC).
showTrace (тип: bool) отображать запросы/ответы.
true – отображать запросы/ответы
false – отображать запросы для которых ответ изменился

Возвращаемое значение
Признак выполнения отправки команды:
true – выполнено
false – ошибка отправки

Пример

```
byte[] command = {0x03, 0x01};
byte[] answer = null;
Modbus.Send(command, out answer, true);
```

Write – метод

Записывает данные в регистр Modbus устройства с отображением запросов/ответов.

Синтаксис

```
public bool Write(UInt16 deviceAddress, UInt16 itemAddress, UInt16 itemValue)
```

Параметры
deviceAddress (тип: UInt16) Modbus адрес устройства.
itemAddress (тип: UInt16) Modbus адрес (стандартный) регистра.
itemValue (тип: out UInt16) значение регистра. Для битовых значений:
0-false
1-true

Возвращаемое значение
Признак выполнения записи:
true – выполнено

false – ошибка

Write – метод

Записывает данные в регистр Modbus устройства с возможностью управления отображением запросов/ответов.

Синтаксис

```
public bool Write(UInt16 deviceAddress, UInt16 itemAddress,
UInt16 itemValue, bool showTrace)
```

Параметры

deviceAddress (тип: UInt16) Modbus адрес устройства.
itemAddress (тип: UInt16) Modbus адрес (стандартный) регистра.
itemValue (тип: out UInt16) значение регистра. Для битовых значений:
0-false
1-true
showTrace (тип: bool) управление отображением запросов/ответов
true – отображать запросы/ответы
false – отображать запросы для которых ответ изменился

Возвращаемое значение

Признак выполнения записи:
true – выполнено
false – ошибка

Write – метод

Записывает данные в несколько регистров Modbus устройства с отображением запросов/ответов. Если записывается один регистр, то используется команда записи одного регистра.

Синтаксис

```
public bool Write(UInt16 deviceAddress, UInt16[] itemAddress, UInt16[] itemValue,
                  int index, int count)
```

Параметры

deviceAddress (тип: UInt16) Modbus адрес устройства.
itemAddress (тип: UInt16[]) список Modbus адресов (стандартных) регистра.
itemValue (тип: out UInt16[]) список значений регистров. Для битовых значений:
0-false
1-true
index (тип: int) начальный элемент списка
count (тип: int) длина списка

Возвращаемое значение

Признак выполнения записи:
true – выполнено
false – ошибка

Write – метод

Записывает данные в несколько регистров Modbus устройства с возможностью управления отображением запросов/ответов.

Синтаксис

```
public bool Write(UInt16 deviceAddress, UInt16[] itemAddress, UInt16[] itemValue,
                  int index, int count, bool showTrace)
```

Параметры
deviceAddress (тип: UInt16) Modbus адрес устройства.
itemAddress (тип: UInt16[]) список Modbus адресов (стандартных) регистра.
itemValue (тип: out UInt16[]) список значений регистров. Для битовых значений:
0-false
1-true
index (тип: int) начальный элемент списка
count (тип: int) длина списка
showTrace (тип: bool) управление отображением запросов/ответов
true – отображать запросы/ответы
false – отображать запросы для которых ответ изменился

Возвращаемое значение
Признак выполнения записи:
true – выполнено
false – ошибка

Пример

```
using System;
using System.IO.Ports;
using System.Threading;
using Proryv.Script;

////////////////////////////////////////////////
// Пример взаимодействия с Modbus устройством по линии RS-485/RS-232
class TestModbus
{
  public static void Main()
  {
    Output.WriteLine("Test Modbus");

    // Удаленный порт:
    //   IP адрес - текущий (можно указывать IP адрес любого другого контроллера TK16L),
    //   TCP порт - определяет номер линии
    //    (1000 - 1 линия RS-485, 2000 - 2 линия RS-485, 3000 - 3 линия RS-485, 4000 - 4 линия RS-485,
    //     9000 - 1 линия RS-232, 8000 - 2 линия RS-232)
    RemotePort remotePort = new RemotePort("127.0.0.1", 3000);
    // Протокол Modbus
    //   remotePort - удаленный порт
    //   Output - системный экземпляр класса для вывода сообщений
    Modbus modbus = new Modbus(remotePort, Output);
    // Параметры порта:
    //   BaudRate - скорость COM порта, бит/сек
    //   DataBits - кол-во битов данных
    //   Stopbits - кол-во стоп-битов (StopBits.None, StopBits.One, StopBits.Two, StopBits.OnePointFive)
    //   Parity - четность (Parity.None, Parity.Odd, Parity.Even, Parity.Mark, Parity.Space)
    //   ReadTimeoutInterval - таймаут чтения (межсимвольный), мс
    //   ReadTimeoutMultiplier - таймаут чтения (множитель), мс
    //   ReadTimeoutConstant - таймаут чтения (константа), мс
    remotePort.BaudRate = 9600;
    remotePort.DataBits = 8;
    remotePort.Stopbits = StopBits.One;
    remotePort.Parity = Parity.None;
    remotePort.ReadTimeoutInterval = 4;
    remotePort.ReadTimeoutMultiplier = 0;
    remotePort.ReadTimeoutConstant = 500;

    // Чтение Modbus регистров:
    //   deviceAddress - Modbus адрес устройства
```

```
        //   itemAddress - массив читаемых Modbus регистров
        //   itemValue - массив значений читаемых Modbus регистров
        //   0 - начальный индекс читаемых регистров в списках
        //   itemAddress.Length - кол-во читаемых регистров
        UInt16 deviceAddress = 3;
        UInt16[] itemAddress = { 40001, 40002 };
        UInt16[] itemValue = new UInt16[itemAddress.Length];
        if (modbus.Read(deviceAddress, itemAddress, itemValue, 0, itemAddress.Length) == false)
        {
           Output.WriteLine("Read - Fail");
           remotePort.Dispose();
           return;
        }
        Output.WriteLine(string.Format("Read - OK, ItemValues=({0})",
          ArrayToStr(itemValue, 0, itemValue.Length)));

        // Запись Modbus регистров
        //   deviceAddress - Modbus адрес устройства
        //   itemAddress - массив записываемых Modbus регистров
        //   itemValue - массив значений записываемых Modbus регистров
        //   0 - начальный индекс записываемых регистров в списках
        //   itemAddress.Length - кол-во записываемых регистров
        if (modbus.Write(deviceAddress, itemAddress, itemValue, 0, itemAddress.Length) == false)
        {
           Output.WriteLine("Write - Fail");
           remotePort.Dispose();
           return;
        }
        Output.WriteLine(string.Format("Write - OK, ItemValues=({0})",
          ArrayToStr(itemValue, 0, itemValue.Length)));

        remotePort.Dispose();
        Output.WriteLine("Test RemotePort Complete");
    }

    ////////////////////////////////////////////////
    // Строковое представление массива байт
    private static string ArrayToStr(UInt16[] array, int index, int count)
    {
        string result = string.Empty;
        if (array == null) return result;
        for (int i = index; i < index + count; i++)
        {
           if (i >= array.Length) break;
           if (i > 0) result = result + ", ";
           result = result + array[i].ToString("X4");
        }
        return result;
    }
}
```

Proryv.Script.StateTS – перечисление

Перечисление состояние ТС:

Off = 0 (ТС сброшен)

On = 1 (ТС установлен)

Proryv.Script.StateTU – перечисление

Перечисление состояние ТУ:
Off = 0 (сбросить ТУ)
On = 1 (установить ТУ)

Proryv.Script.UK – класс

Статический класс для работы со встроенным коммутационным устройством (УК).

Open – метод

Открывает сеанс работы с УК.

Синтаксис
public bool Open()

Close – метод

Закрывает сеанс работы с УК.

Синтаксис
public void Close()

GetTS – метод

Возвращает значение для указанного номера дискретного входа (с единицы) в параметре out StateTS value.

Синтаксис
public bool GetTS(byte number, out StateTS value)

Параметры
number (тип: byte) номер входа ТС с единицы
value (тип: out StateTS) состояние ТС (0 или1)

Возвращаемое значение
Признак выполнения:
true – выполнено
false – ошибка

GetTSMask – метод

Возвращает значение маски ТС.

Синтаксис
public bool GetTSMask(out UInt32 value)

Параметры
value (тип: out UInt32) битовая маска ТС

Возвращаемое значение
Признак выполнения:
true – выполнено
false – ошибка

GetTIR – метод

Возвращает значение ТИР для указанного номера дискретного входа (с единицы) в параметре out UInt16 value.

Синтаксис
public bool GetTIR(byte number, out UInt16 value)

Параметры
number (тип: byte) номер входа ТИР с единицы
value (тип: out UInt16) значение ТИР

Возвращаемое значение
Признак выполнения:
true – выполнено
false – ошибка

GetTIT – метод

Возвращает значение ТИТ для указанного номера аналогового входа (с единицы) в параметре out UInt16 value.

Синтаксис
public bool GetTIT(byte number, out UInt16 value)

Параметры
number (тип: byte) номер входа ТИТ с единицы
value (тип: out UInt16) значение ТИТ

Возвращаемое значение
Признак выполнения:
true – выполнено
false – ошибка

SetTU – метод

Устанавливает значение выходного сигнала ТУ для указанного номера выхода телеуправления.

Синтаксис
public bool SetTU(byte number, StateTU value)

Параметры
number (тип: byte) номер выхода ТУ с единицы
value (тип: out StateTU) значение выходного сигнала ТУ:
true – включить
false – выключить

Возвращаемое значение
Признак выполнения:
true – выполнено
false – ошибка

SetTU – метод

Устанавливает значение выходного сигнала ТУ для указанного номера выхода телеуправления с удержанием сигнала.

Синтаксис

public bool SetTU(byte number, StateTU value, int hold)

Параметры
number (тип: byte) номер выхода ТУ с единицы
value (тип: StateTU) значение выходного сигнала ТУ:
true – включить
false – выключить
hold (тип: int) задержка сигнала, мс

Возвращаемое значение
Признак выполнения:
true – выполнено
false – ошибка

SetTUCyclic – метод

Включает циклическое переключение ТУ для указанного номера выхода телеуправления.

Синтаксис
public bool SetTUCyclic(byte number, int holdOn, int holdOff)

Параметры
number (тип: byte) номер выхода ТУ с единицы
holdOn (тип: int) задержка во включенном состоянии, мс
holdOff (тип: int) задержка в выключенном состоянии, мс

Возвращаемое значение
Признак выполнения:
true – выполнено
false – ошибка

GetTU – метод

Возвращает значение состояния канала телеуправления.

Синтаксис
public bool GetTU(byte number, out StateTU value)

Параметры
number (тип: byte) номер канала телеуправления (с единицы)
value (тип: StateTU) состояние канала телеуправления

Возвращаемое значение
Признак выполнения:
true – выполнено
false – ошибка

SetVoltageScale – метод

Устанавливает параметры масштабирования напряжения для указанного номера аналогового входа (с единицы).

Синтаксис
public void SetVoltageScale(byte number, double minPhis, double maxPhis, byte minADC, byte maxADC, double coeff)

Параметры
number (тип: byte) номер аналогового входа ТИТ (с единицы)
minPhis (тип: double) минимальное значение физической шкалы (по умолчанию 0 В)

maxPhis (тип: double) максимальное значение физической шкалы (по умолчанию 5 В)
minADC (тип: byte) минимальное значение шкалы АЦП (по умолчанию 51)
maxADC (тип: byte) максимальное значение шкалы АЦП (по умолчанию 255)
coeff (тип: double) поправочный коэффициент (по умолчанию 1)

SetCurrentScale – метод

Устанавливает параметры масштабирования тока для указанного номера аналогового входа (с единицы).

Синтаксис

```
public void SetCurrentScale(byte number, double minPhis, double maxPhis, byte minADC, byte maxADC,
                            double coeff)
```

Параметры

number (тип: byte) номер аналогового входа ТИТ (с единицы)
minPhis (тип: double) минимальное значение физической шкалы (по умолчанию 0 мА)
maxPhis (тип: double) максимальное значение физической шкалы (по умолчанию 20 мА)
minADC (тип: byte) минимальное значение шкалы АЦП (по умолчанию 51)
maxADC (тип: byte) максимальное значение шкалы АЦП (по умолчанию 255)
coeff (тип: double) поправочный коэффициент (по умолчанию 1)

GetVoltage – метод

Возвращает значение напряжения.

Синтаксис

```
public bool GetVoltage(byte number, out double value)
```

Параметры

number (тип: byte) номер аналогового входа ТИТ (с единицы)
value (тип: double) значение напряжения, В

Возвращаемое значение

Признак выполнения:
true – выполнено
false – ошибка

GetCurrent – метод

Возвращает значение тока.

Синтаксис

```
public bool GetCurrent(byte number, out double value)
```

Параметры

number (тип: byte) номер аналогового входа ТИТ (с единицы)
value (тип: double) значение тока, мА

Возвращаемое значение

Признак выполнения:
true – выполнено
false – ошибка

Пример для UK

```
using System;
using System.Threading;
using Proryv.Script;
```

```
////////////////////////////////////////////////
// Пример взаимодействия с платой УК (устройство коммуникационное)
class TestUK
{
   public static void Main()
   {
      Output.WriteLine("Start test");

      // Подключение к плате УК
      if (UK.Open() == true)
         Output.WriteLine("Open UK - OK");
      else Output.WriteError("Open UK - Fail");

      StateTU stateTU = StateTU.Off;
      while (true)
      {
         Thread.Sleep(3000);

         // Получение маски ТС
         UInt32 tsMask = (UInt32)0;
         if (UK.GetTSMask(out tsMask) == true)
            Output.WriteLine(string.Format("Mask TS, Value={0}",
              tsMask));
         else Output.WriteError("Get Mask TS - Fail");

         // Получение состояния ТС1
         byte number = 1;
         StateTS stateTS = StateTS.Off;
         if (UK.GetTS(number, out stateTS) == true)
            Output.WriteLine(string.Format("TS, Number={0}, Value={1}",
              number, stateTS));
         else Output.WriteError(string.Format("Get TS - Fail, Number={0}",
            number));

         // Установка ТУ1 (попеременно)
         if (stateTU == StateTU.Off)
            stateTU = StateTU.On;
         else stateTU = StateTU.Off;
         if (UK.SetTU(number, stateTU) == true)
            Output.WriteLine(string.Format("TU, Number={0}, Value={1}",
              number, stateTU));
         else Output.WriteError(string.Format("Set TU - Fail, Number={0}",
            number));

         // Получение ТИР1
         UInt16 tirValue = (UInt16)0;
         if (UK.GetTIR(number, out tirValue) == true)
            Output.WriteLine(string.Format("TIR, Number={0}, Value={1}",
              number, tirValue));
         else Output.WriteError(string.Format("Get TIR - Fail, Number={0}",
            number));

         // Получение ТИТ2
         number = 2;
         UInt16 titValue = (UInt16)0;
         if (UK.GetTIT(number, out titValue) == true)
            Output.WriteLine(string.Format("TIT, Number={0}, Value={1}",
              number, titValue));
         else Output.WriteError(string.Format("Get TIT - Fail, Number={0}",
```

```
                number));
            }
        }
    }
```

Proryv.Script.ScriptDeviceParams – класс

Класс содержит параметры устройства: идентификатор, строка инициализации. Используется для работы с системой Телескоп+.

DeviceID – поле

Идентификатор устройства.

Синтаксис
public string DeviceID

InitData – поле

Строка инициализации.

Синтаксис
public byte[] InitData

Proryv.Script.ScriptEvents – класс

Класс для проверки данных, поступающих от системы Телескоп+.

ScriptEvents – конструктор

Инициализирует новый экземпляр класса Proryv.Script.ScriptEvents.

IsInit – метод

Проверяет наличие инициализации.

Синтаксис
public bool IsInit(out ScriptDeviceParams[] devicesParams) параметры устройства

Параметры
devicesParams (тип: out ScriptDeviceParams[])

Возвращаемое значение
Признак наличия инициализации:
true – есть
false – нет

IsCommand – метод

Проверяеет наличие команды для скрипта.

Синтаксис
public bool IsCommand(out string deviceID, out byte[] data)

Параметры
deviceID (тип: out string) идентификатор устройства
data (тип: out byte[]) данные

Возвращаемое значение
Признак наличия команды для скрипта:
true – есть
false – нет

Пример

```
using System;
using System.Threading;
using Proryv.Script;

////////////////////////////////////////////////
// Пример взаимодействия со SCADA системой Телескоп+
class TestTelescope
{
   public static void Main()
   {
      Output.WriteLine("Project1, Start");

      try
      {
         while (true)
         {
            Thread.Sleep(3000);

            // Проверка получения инициализации от Телескоп+
            //   devicesParams - параметры устройств (DeviceID - идентификатор устройства (строка),
            //     InitData - параметры устройства (массив байт))
            ScriptDeviceParams[] devicesParams = null;
            if (ScriptEvents.IsInit(out devicesParams) == true)
            {
               Output.WriteLine(string.Format("Init, DevicesParams=({0})",
                  ScriptDeviceParamsToStr(devicesParams)));
            }

            // Проверка получения команды от Телескоп+
            //   deviceID - идентификатор устройства
            //   data - параметры команды
            string deviceID = string.Empty;
            byte[] data = null;
            if (ScriptEvents.IsCommand(out deviceID, out data) == true)
            {
               Output.WriteLine(string.Format("Command, DeviceID={0}, Data=({1})",
                  deviceID, DataUtils.DataToStrHex(data)));
            }

            // Посылка команды для Телескоп+
            //   deviceID - идентификатор устройства
            //   data - параметры команды
            deviceID = "123";
            data = new byte[] { 0x03, 0x04, 0x05, 0x06 }; ;
            if (ScriptCommands.SendCommand(deviceID, data) == true)
            {
               Output.WriteLine("SendCommand - OK");
            }
         }
      }
      catch (Exception ex)
      {
```

```
            Output.WriteException("Main.Main", ex);
        }
    }

//////////////////////////////////////////////////////
// Строковое представление списка параметров устройств
public static string ScriptDeviceParamsToStr(ScriptDeviceParams[] devicesParams)
{
        if (devicesParams == null) return;
    string result = string.Format("[{0}] ", devicesParams.Length);
        for (int i = 0; i < devicesParams.Length; i++)
        {
                if (i > 0) result = result + "; ";
                result = result + string.Format("DeviceID={0}, InitData=({1})",
                        devicesParams[i].DeviceID,
DataUtils.DataToStrHex(devicesParams[i].InitData));
        }
        return result;
}
}
```

Proryv.Script.ScriptCommands – класс

Класс для отправки команды в систему Телескоп+.

ScriptCommands – конструктор

Инициализирует новый экземпляр класса Script.ScriptCommands.

SendCommand – метод

Отправляет команду в систему Телескоп+.

Синтаксис

```
public bool SendCommand(string deviceID, byte[] data)
```

Параметры
deviceID (тип: string) идентификатор устройства
data (тип: byte[]) данные

Возвращаемое значение
Признак выполнения:
true – выполнено
false – ошибка

Пример

См. раздел Proryv.Script.ScriptEvents Пример.

Контакты

140184, Московская обл.,

г. Жуковский-4, а/я 304

тел. (495) 556-6603, 979-94-34, 972-32-71

факс (495) 972-3580

E-mail online@proryv.com

www.proryv.com