ООО Конструкторское Бюро "АГАВА"
620026 г. Екатеринбург, ул. Бажова, 174
тел/факс: (343)-262-92-76, 78, 87
agava@kb-agava.ru; www.kb-agava.ru

Программируемый логический контроллер

АГАВА 6432.20 ПЛК1

ИНСТРУКЦИЯ ЭКСПЛУАТАЦИОННАЯ СПЕЦИАЛЬНАЯ

АГСФ.421445.001ИС
/Редакция 1.2/

Екатеринбург
2012

Содержание

Введение

Инструкция эксплуатационная специальная содержит сведения, необходимые для обеспечения правильной эксплуатации и полного использования технических возможностей *программируемого логического контроллера АГАВА6432.20 ПЛК1*, далее по тексту *ПРИБОР, ПЛК* или *КОНТРОЛЛЕР* в среде ISaGRAF.

Детальное описание работы в среде проектирования ISaGRAF Workbench приводится в сопроводительной документации в комплекте поставки ISaGRAF Workbench.

В данной инструкции описывается работа со специальными для ПЛК АГАВА6432.20 ПЛК1 С-функциями, функциональными блоками и устройствами ввода-вывода.

Набор С-функций, функциональных блоков и устройств ввода-вывода описывается в файле конфигурации *.tdb. Реализация функций происходит в целевой системе ISaGRAF. Поэтому важно соблюдать при работе соответствие между версией целевой системы ISaGRAF и файлом конфигурации *.tdb, который поставляется совместно с соответствующей целевой системой.

Используемые термины и сокращения:
ПК – персональный компьютер;
ПЛК – программируемый логический контроллер;
СП – среда проектирования ISaGRAF Workbench;
ОС – операционная система;
ПО – программное обеспечение;
ОЗУ – оперативное запоминающее устройство;
ФС – файловая система;
ФБ – функциональный блок;
УВВ – устройство ввода-вывода;

1 Специальные функции и функциональные блоки

Beep

Описание: Подача звукового сигнала

Аргументы:

continues	DINT	Время продолжительности звукового сигнала в 0,1сек.

Возвращаемые значения:

	DINT	1 – если выполнение прошло успешно; 0 – если произошла ошибка.

Пример:

```
Ret := Beep(10); (* Подача звукового сигнала длительностью 1 сек *)
```

BLScr

Описание: Управление подсветкой индикатора

Аргументы:

R	BOOL	Управление красной подсветкой: TRUE – включена, FALSE – выключена.
G	BOOL	Управление зеленой подсветкой.
B	BOOL	Управление голубой подсветкой.

Возвращаемые значения:

	DINT	1 – если выполнение прошло успешно; 0 – если произошла ошибка.

Пример:

```
Ret := BLScr(TRUE, TRUE, FALSE);
(* Включение красной и зеленой подсветки индикатора *)
```

D_ClrScr

Описание: Очистка экрана и установка курсора в верхний левый угол экрана

Аргументы:

desc	DINT	Дескриптор индикатора.

Возвращаемые значения:

	DINT	1 – если выполнение прошло успешно; 0 – если произошла ошибка.

Пример:

```
Ret := D_ClrScr(d_ind);
(* Очищает экран и устанавливает курсор в верхний левый угол*)
```

D_CurVisible

Описание: Включение/выключения курсора на дисплее

Аргументы:

desc	DINT	Дескриптор индикатора.
visible	BOOL	TRUE – курсор включен; FALSE – курсор выключен.

Возвращаемые значения:

DINT	1 – если выполнение прошло успешно; 0 – если произошла ошибка.

Пример:

```
Ret := D_CurVisible(d_ind, FALSE); (* Выключает курсор на дисплее *)
```

D_DisplayInit

Описание: Инициализация дисплея.
Функция должна вызываться только один раз до первого обращения к другим функциям работы с дисплеем.

Аргументы: отсутствуют.

Возвращаемые значения:

DINT	Значение дескриптора дисплея, либо 0 – если произошла ошибка.

Пример:

```
d_ind := D_DisplayInit(); (* Инициализирует дисплей *)
```

D_PrintDINTScr

Описание: Выводит на экран значение типа DINT в текущую позицию курсора

Аргументы:

desc	DINT	Дескриптор индикатора.
par	DINT	Значение для вывода на экран.

Возвращаемые значения:

DINT	1 – если выполнение прошло успешно; 0 – если произошла ошибка.

Пример:

```
Ret := D_PrintDINTScr(d_ind, 1000); (* Выводит на экран число 1000 *)
```

D_PrintInvScr

Описание: Установка инверсного/неинверсного режима вывода на экран

Аргументы:

desc	DINT	Дескриптор индикатора.
invert	BOOL	TRUE – включение инверсного режима вывода; FALSE – выключение инверсного режима вывода.

Возвращаемые значения:

DINT	1 – если выполнение прошло успешно; 0 – если произошла ошибка.

Пример:

```
Ret := D_PrintInvScr(d_ind, TRUE);
(* Включение инверсного режима вывода на экран *)
```

D_PrintREALScr

Описание: Выводит на экран значение типа REAL в текущую позицию курсора

Аргументы:

desc	DINT	Дескриптор индикатора;
par	REAL	Значение для вывода на экран;
prec	DINT	Точность вывода (число знаков после запятой).

Возвращаемые значения:

DINT	1 – если выполнение прошло успешно; 0 – если произошла ошибка.

Пример:

```
Ret := D_PrintRealScr(d_ind, 1.2345, 2); (* Выводит на экран 1.23 *)
```

D_PrintScr

Описание: Выводит на экран строку в текущую позицию курсора
Строка может содержать кириллические символы.

Аргументы:

desc	DINT	Дескриптор индикатора;
in_string	STRING	Строка для вывода.

Возвращаемые значения:

DINT	1 – если выполнение прошло успешно; 0 – если произошла ошибка.

Пример:

```
Ret := D_PrintScr(d_ind, ‘Привет’); (* Выводит Привет на экран *)
```

D_SetCurPos

Описание: Устанавливает курсор в указанное местоположение
Координаты левого верхнего угла: X=1, Y=1.

Аргументы:

desc	DINT	Дескриптор индикатора;
X	DINT	Позиция X курсора;
Y	DINT	Позиция Y курсора;

Возвращаемые значения:

DINT	1 – если выполнение прошло успешно; 0 – если произошла ошибка.

Пример:

```
Ret := D_SetCurPos(d_ind, 1, 2);
(* Устанавливает курсор на первый символ второй строки сверху *)
```

DayTime

Описание: Возвращает строку, содержащую время, или дату или день недели

Аргументы:

SEC	DINT	Число секунд сначала 01/01/1970
SEL	DINT	0 – для строки даты (в формате DD/MM/YYYY); 1 – для строки времени; 2 – для строки дня недели.

Возвращаемые значения:

STRING	Строка, содержащая дату или время, или день недели.

Пример:

```
String := DayTime(Now(), 0);
(* Записывает в String строку, содержащую текущую дату *)
```

F_Close

Описание: Закрывает файл

Аргументы:

desc	DINT	Дескриптор закрываемого файла

Возвращаемые значения:

DINT	1 – если выполнение прошло успешно; 0 – если произошла ошибка.

Пример:

```
ret := F_Close(f_desc); (* Закрывает файл с дескриптором f_desc *)
```

F_Copy

Описание: Копирует файл(ы).
Копирование происходит отдельным фоновым процессом.
Если файл назначения уже существует, он перезаписывается.

Аргументы:

name_source	STRING	Полный путь и имя копируемого файла (файлов). В именах файлов допускается использовать шаблоны типа *.
name_destination	STRING	Полный путь места назначения файла (файлов). Путь должен существовать.

Возвращаемые значения:

DINT	1 – запуск процесса копирования произошел успешно; 0 – если произошла ошибка.

Пример:

```
ret := F_Copy('/mnt/mmc/*.log', '/mnt/usbmem');
(* Копирует все файлы с расширением log в каталог /mnt/usbmem *)
```

F_Delete

Описание: Удаление файла. Если файл не открыт, производится его удаление, в противном случае удаление произойдет после закрытия файла. (версия прошивки от 28.03.2012 и новее).

Аргументы:

filename	STRING	Полный путь и имя файла.

Возвращаемые значения:

DINT	1 – если выполнение прошло успешно; 0 – если произошла ошибка.

Пример:

```
ret := F_Delete('/mnt/mmc/file.bin');
(* Удаляет файл /mnt/mmc/file.bin *)
```

F_Eof

Описание: Проверка на конец файла

Аргументы:

desc	DINT	Дескриптор проверяемого файла

Возвращаемые значения:

DINT	1 – если конец файла достигнут; 0 – если конец файла не достигнут.

Пример:

```
ret := F_Eof(f_desc);
```

F_Flush

Описание: Вызывает немедленную запись всех данных файла из буфера на физический носитель

Аргументы:

desc	DINT	Дескриптор файла

Возвращаемые значения:

DINT	1 – если выполнение прошло успешно; 0 – если произошла ошибка.

Пример:

```
ret := F_Flush(f_desc);
```

F_OpenReadBinary

Описание: Открытие на чтение бинарного файла.

Аргументы:

filename	STRING	Полный путь и имя файла.

Возвращаемые значения:

DINT	Значение дескриптора файла, либо 0 – если произошла ошибка.

Пример:

```
f_desc := F_OpenReadBinary('/mnt/mmc/file.bin');
(* Открывает на чтение в бинарном режиме файл /mnt/mmc/file.bin *)
```

F_OpenReadString

Описание: Открытие на чтение текстового файла.

Аргументы:

filename	STRING	Полный путь и имя файла.

Возвращаемые значения:

DINT	Значение дескриптора файла, либо 0 – если произошла ошибка.

Пример:

```
f_desc := F_OpenReadString('/mnt/mmc/file.txt');
(* Открывает на чтение в текстовом режиме файл /mnt/mmc/file.txt *)
```

F_OpenWriteBinary

Описание: Открытие на запись бинарного файла.
Если файл не существует, создается новый. Если существует, файл дописывается.

Аргументы:

filename	STRING	Полный путь и имя файла.

Возвращаемые значения:

DINT	Значение дескриптора файла, либо 0 – если произошла ошибка.

Пример:

```
f_desc := F_OpenWriteBinary('/mnt/mmc/file.bin');
(* Открывает на запись в бинарном режиме файл /mnt/mmc/file.bin *)
```

F_OpenWriteString

Описание: Открытие на запись текстового файла.
Если файл не существует, создается новый. Если существует, файл дописывается.

Аргументы:

filename	STRING	Полный путь и имя файла.

Возвращаемые значения:

DINT	Значение дескриптора файла, либо 0 – если произошла ошибка.

Пример:

```
f_desc := F_OpenWriteString('/mnt/mmc/file.txt');
(* Открывает на запись в текстовом режиме файл /mnt/mmc/file.txt *)
```

F_ReadDINT

Описание: Читает из файла значение типа DINT

Аргументы:

desc	DINT	Дескриптор файла

Возвращаемые значения:

par	DINT	Прочитанное значение.
ok	DINT	1 – если выполнение прошло успешно; 0 – если произошла ошибка; 2 – достигнут конец файла.

Пример:

```
ReadDINT(f_desc); (* ReadDINT – экземпляр функционального блока F_ReadDINT*)
IF ReadDINT.ok = 1 THEN
Val := ReadDINT.par;
END_IF;
(* Читает из файла с дескриптором f_desc значение типа DINT и в случае успешного чтения присваивает это значение переменной Val *)
```

F_ReadString

Описание: Читает из файла строку

Аргументы:

desc	DINT	Дескриптор файла

Возвращаемые значения:

str	STRING	Прочитанная строка.
ok	DINT	1 – если выполнение прошло успешно; 0 – если произошла ошибка; 2 – достигнут конец файла.

Пример:

```
(* ReadString – экземпляр функционального блока F_ReadString *)
ReadString(f_desc);
IF ReadString.ok = 1 THEN
str := ReadString.str;
END_IF;
(* Читает строку из файла с дескриптором f_desc и в случае успешного чтения присваивает ее значение переменной str *)
```

F_Seek

Описание: Устанавливает указатель текущей позиции файла

Аргументы:

desc	DINT	Дескриптор файла
offset	DINT	Значение смещения для установки указателя
origin	DINT	Начало отсчета смещения: 0 – с начала файла; 1 – с текущей позиции; 2 – с конца файла.

Возвращаемые значения:

DINT	1 – если выполнение прошло успешно; 0 – если произошла ошибка.

Пример:

```
ret := F_Seek(f_desc, 4, 0);
(* Устанавливает указатель в файле с дескриптором f_desc на 4 позицию относительно начала файла *)
```

F_WriteDINT

Описание: Записывает в файл значение типа DINT
Примечание: запись буферизируется. Чтобы немедленно выполнить запись на физический носитель следует воспользоваться функцией F_Flush, либо закрытием файла функцией F_Close.

Аргументы:

desc	DINT	Дескриптор файла.
par	DINT	Значение для записи.

Возвращаемые значения:

DINT	1 – если выполнение прошло успешно; 0 – если произошла ошибка.

Пример:

```
ret := F_WriteDINT(f_desc, 1000);
(* Записывает число 1000 в файл с дескриптором f_desc *)
```

F_WriteString

Описание: Запись строки в файл.
Примечание: запись буферизируется. Чтобы немедленно выполнить запись на физический носитель следует воспользоваться функцией F_Flush, либо закрытием файла функцией F_Close.

Аргументы:

desc	DINT	Дескриптор файла.
str	DINT	Строка для записи.

Возвращаемые значения:

	DINT	1 – если выполнение прошло успешно; 0 – если произошла ошибка.

Пример:

```
ret := F_WriteString(f_desc, 'Hello');
(* Записывает строку Hello в файл с дескриптором f_desc *)
```

GetTimeString

Описание: Получение строки времени в формате YYYY/MM/DD HH:MM:SS

Аргументы:

SEC	DINT	Число секунд, прошедшее с 01/01/1970

Возвращаемые значения:

	STRING	Строка, содержащая время

Пример:

```
time := GetTimeString(Now()); (* Записывает в строку time текущее время *)
```

GetTimeStruct

Описание: Возвращает структуру времени

Аргументы:

SEC	DINT	Число секунд, прошедшее с 01/01/1970

Возвращаемые значения:

YEAR	DINT	Год
MONTH	DINT	Месяц
DAY	DINT	День
HOUR	DINT	Часы
MINUTE	DINT	Минуты
SECOND	DINT	Секунды

Пример:

```
GetTime(Now()); (* GetTime – экземпляр функционального блока GetTimeStruct *)
Year := GetTime.year;
```

K_InputDINT

Описание: Ввод с клавиатуры числа типа DINT
ФБ выполняется циклически, пока ввод числа не завершится клавишей «Ввод». Редактирование осуществляется клавишей «Стоп».

Аргументы:

keydesc	DINT	Дескриптор клавиатуры.
keydisp	DINT	Дескриптор индикатора. Если 0, ввод отображаться на экране не будет.

Возвращаемые значения:

value	DINT	Введенное значение.
ok	DINT	Состояние ввода: 1 – ввод завершен, число введено; 2 – ввод завершен, число не вводилось; 3 – ввод не завершен.

Пример:

```
inpDINT(keyb, disp);  (*inpDINT – экземпляр функционального блока K_InputDINT *)
IF inpDINT.ok = 1 THEN
        rt:= D_PrintScr(disp, '$nВвели: ');
        rt:= D_PrintDINTScr(disp, inpDINT.value);
ELSIF inpDINT.ok = 2 THEN
        rt:= D_PrintScr(disp, '$nНичего не ввели');
END_IF;
```

K_InputREAL

Описание: Ввод с клавиатуры числа типа REAL
ФБ выполняется циклически, пока ввод числа не завершится клавишей «Ввод». Редактирование осуществляется клавишей «Стоп».

Аргументы:

keydesc	DINT	Дескриптор клавиатуры.
keydisp	DINT	Дескриптор индикатора. Если 0, ввод отображаться на экране не будет.

Возвращаемые значения:

value	REAL	Введенное значение.
ok	DINT	Состояние ввода: 1 – ввод завершен, число введено; 2 – ввод завершен, число не вводилось; 3 – ввод не завершен.

Пример:

```
inpREAL(keyb, disp); (*inpREAL – экземпляр функционального блока K_InputREAL *)
IF inpREAL.ok = 1 THEN
        rt:= D_PrintScr(disp, '$nВвели: ');
        rt:= D_PrintREALScr(disp, inpREAL.value, 3);
ELSIF inpREAL.ok = 2 THEN
        rt:= D_PrintScr(disp, '$nНичего не ввели');
END_IF;
```

K_KeybInit

Описание: Инициализация клавиатуры.
Функция должна вызываться только один раз до обращения к другим функциям работы с клавиатурой.

Аргументы: Отсутствуют

Возвращаемые значения:

DINT — Дескриптор клавиатуры. Если значение дескриптора больше нуля, инициализация прошла успешно.

Пример:

```
keyb := K_KeybInit();
```

K_ReadKey

Описание: Читает код нажатой клавиши

Аргументы:

desc DINT — Дескриптор клавиатуры

Возвращаемые значения:

DINT — Код нажатой клавиши, либо 0 – если клавиша не нажата.
Коды клавиш (шестнадцатеричные значения):

Клав.	Код	ASCII	Клав.	Код	ASCII
0	30h	‘0’	,	2eh	‘.’
1	31h	‘1’	-	2dh	‘-‘
2	32h	‘2’	меню	6dh	‘m’
3	33h	‘3’	ввод	0dh	CR
4	34h	‘4’	вверх	75h	‘u’
5	35h	‘5’	вниз	64h	‘d’
6	36h	‘6’	пуск	70h	‘p’
7	37h	‘7’	стоп	73h	‘s’
8	38h	‘8’	свет	6ch	‘l’
9	39h	‘9’	звук	7ah	‘z’

Пример:

```
key := K_ReadKey(keyb);
```

LEDFailure

Описание: Управление светодиодом «Авария»

Аргументы:

on	BOOL	TRUE – включить FALSE - выключить

Возвращаемые значения:

DINT	1 – если выполнение прошло успешно; 0 – если произошла ошибка.

Пример:

```
ret := LEDFailure(TRUE); (* Включение светодиода «Авария» *)
```

LEDProgramm

Описание: Управление светодиодом «Программа»

Аргументы:

on	BOOL	TRUE – включить FALSE - выключить

Возвращаемые значения:

DINT	1 – если выполнение прошло успешно; 0 – если произошла ошибка.

Пример:

```
ret := LEDProgramm(TRUE); (* Включение светодиода «Программа» *)
```

LEDWork

Описание: Управление светодиодом «Работа»

Аргументы:

on	BOOL	TRUE – включить FALSE - выключить

Возвращаемые значения:

DINT	1 – если выполнение прошло успешно; 0 – если произошла ошибка.

Пример:

```
ret := LEDWork(TRUE); (* Включение светодиода «Работа» *)
```

N_FtpGet

Описание: Загружает файл с удаленного устройства по протоколу FTP
Загрузка происходит отдельным фоновым процессом.

Аргументы:

server_name	STRING	IP-адрес или имя FTP-сервера.
fname_ftp	STRING	Полный путь и имя файла на FTP-сервере.
fname_local	STRING	Полный путь и имя файла для сохранения на локальный носитель.
login	STRING	Имя пользователя для FTP-сервера, либо пустая строка.
pass	STRING	Пароль пользователя, либо пустая строка.

Возвращаемые значения:

DINT	1 – запуск процесса загрузки произошел успешно; 0 – если произошла ошибка.

Пример:

```
ret:=N_FtpGet('ftp.kb-agava.ru', '/outgoing/settings.txt', '/mnt/mmc/settings.txt', 'plcagava', '12345');
(* выполняет вход на FTP-сервер ftp.kb-agava.ru под пользователем plcagava с паролем 12345, загружает файл /outgoing/settings.txt с FTP-сервера и сохраняет его в /mnt/mmc/settings.txt *)
```

N_FtpPut

Описание: Загружает файл на удаленное устройства по протоколу FTP
Загрузка происходит отдельным фоновым процессом.

Аргументы:

server_name	STRING	IP-адрес или имя FTP-сервера.
fname_ftp	STRING	Полный путь и имя файла на FTP-сервере.
fname_local	STRING	Полный путь и имя файла на локальном носителе.
login	STRING	Имя пользователя для FTP-сервера, либо пустая строка.
pass	STRING	Пароль пользователя, либо пустая строка.

Возвращаемые значения:

DINT	1 – запуск процесса загрузки произошел успешно; 0 – если произошла ошибка.

Пример:

```
ret:=N_FtpPut('ftp.kb-agava.ru', '/incoming/plc.log', '/mnt/mmc/plc.log', 'plcagava', '12345');
(* выполняет вход на FTP-сервер ftp.kb-agava.ru под пользователем plcagava с паролем 12345, загружает файл /mnt/mmc/plc.log на FTP-сервер в  /incoming/plc.log *)
```

N_MailSend

Описание: Посылает e-mail с содержанием из строки
Выполнение происходит отдельным фоновым процессом.

Аргументы:

to_addr	STRING	Адрес получателя.
from_addr	STRING	Адрес отправителя.
subj	STRING	Содержание темы письма.
host	STRING	Имя или IP-адрес почтового сервера через который происходит отправка.
content	STRING	Содержание письма.

Возвращаемые значения:

DINT	1 – запуск процесса mail-клиента произошел успешно; 0 – если произошла ошибка.

Пример:

```
ret:=N_MailSend('collector@kb-agava.ru', 'fromplc@kb-agava.ru', 'logger', 'mail.kb-agava.ru', 'Plc started');
(* отправляет  e-mail с содержанием Plc started по адресу collector@kb-agava.ru от fromplc@kb-agava.ru с темой сообщения - logger через почтовый сервер mail.kb-agava.ru. *)
```

N_MailSendFile

Описание: Посылает e-mail с содержанием из файла
Выполнение происходит отдельным фоновым процессом.

Аргументы:

to_addr	STRING	Адрес получателя.
from_addr	STRING	Адрес отправителя.
subj	STRING	Содержание темы письма.
host	STRING	Имя или IP-адрес почтового сервера через который происходит отправка.
filename	STRING	Полный путь с именем файла, содержание которого будет в теле письма.

Возвращаемые значения:

DINT	1 – запуск процесса mail-клиента произошел успешно; 0 – если произошла ошибка.

Пример:

```
ret:=N_MailSendFile('collector@kb-agava.ru', 'fromplc@kb-agava.ru', 'logger', 'mail.kb-agava.ru', '/mnt/mmc/system.log');
(* отправляет  e-mail с содержанием из файла /mnt/mmc/system.log по адресу collector@kb-agava.ru от fromplc@kb-agava.ru с темой сообщения - logger через почтовый сервер mail.kb-agava.ru. *)
```

Now

Описание: Возвращает текущее время в секундах, начиная с 01/01/1970

Аргументы: Отсутствуют

Возвращаемые значения:

	DINT	Текущее время в секундах, начиная с 01/01/1970

Пример:

```
time := Now();
```

S_Reboot

Описание: Выполняет перезагрузку контроллера.
Все процессы завершаются, файлы закрываются.

Аргументы:

delay	DINT	Задержка перезагрузки в секундах.

Возвращаемые значения:

	DINT	0 – если произошла ошибка.

Пример:

```
ret := S_Reboot(3); (* Перезагрузка контроллера через 3 сек. *)
```

S_Shutdown

Описание: Выполняет останов работы контроллера.
Запуск контроллера произойдет при следующей подаче питания.
Все процессы завершаются, файлы закрываются.

Аргументы:

delay	DINT	Задержка останова в секундах.

Возвращаемые значения:

	DINT	0 – если произошла ошибка.

Пример:

```
ret := S_Shutdown(3); (* Останов работы контроллера через 3 сек. *)
```

SerialClose

Описание: Закрывает коммуникационный порт

Аргументы:

desc	DINT	Дескриптор линии связи.

Возвращаемые значения:

DINT	1 – если выполнение прошло успешно; 0 – если произошла ошибка.

Пример:

```
ret := SerialClose(s_desc);
```

SerialOpen

Описание: Открывает коммуникационный порт

Аргументы:

port STRING Строка с именем порта:

Физ. порт	Имя порта
RS485-1	/dev/ttyS2
RS485-2	/dev/ttyS3
RS485-3	/dev/ttyS0
RS232	/dev/ttyS1

Возвращаемые значения:

DINT	Дескриптор порта. Если значение дескриптора больше нуля, открытие порта произошло успешно.

Пример:

```
desc := SerialOpen('/dev/ttyS1');  (* Открывает порт RS-232 *)
```

SerialReceive

Описание: Принимает байты по линии связи

Аргументы:

desc	DINT	Дескриптор линии связи.
data	STRING	Получаемая информация.
len	DINT	Число байтов, которые необходимо получить.

Возвращаемые значения:

DINT	Число полученных байтов или -1 если произошла ошибка.

Пример:

```
n_bytes := SerialReceive(s_desc, buf, 10);  (*Считывает до 10 байтов с линии связи*)
```

SerialSend

Описание: Отправляет данные по линии связи

Аргументы:

desc	DINT	Дескриптор линии связи.
data	STRING	Передаваемая информация.
len	DINT	Число байтов, которые необходимо передать.

Возвращаемые значения:

DINT	1 – если выполнение прошло успешно; 0 – если произошла ошибка.

Пример:

```
n_bytes := SerialSend(s_desc, buf, 10);  (* Отправляет 10 байтов по линии связи *)
```

SerialSet

Описание: Устанавливает параметры линии связи.

Аргументы:

desc	DINT	Дескриптор линии связи.
baud	DINT	Скорость в бодах. Поддерживаемые скорости: 50 75 110 134 150 200 300 600 1200 1800 2400 4800 9600 19200 38400 57600 115200 230400 460800 921600
sbits	DINT	Число стоповых битов 1 или 2.
parity	STRING	Четность ‘none’, ‘even’, ‘odd’.
flow	STRING	Управление потоком ‘off’, ‘xonxoff’, ‘hardw’.
modem	BOOL	Управление линиями модема: FALSE – нет, TRUE – есть.

Возвращаемые значения:

DINT	1 – если выполнение прошло успешно; 0 – если произошла ошибка.

Пример:

```
ret := SerialSet(s_desc, 9600, 1, 'none', 'off', FALSE);
(* Устанавливает параметры линии связи со скоростью 9600бод, 1 стоп-бит, без контроля четности, управление потоком и линиями модема выключено *)
```

SetTime

Описание: Установка системного времени и часов реального времени.

Аргументы:

YEAR	DINT	Год.
MONTH	DINT	Месяц.
DAY	DINT	День.
HOUR	DINT	Часы.
MINUTE	DINT	Минуты.
SECOND	DINT	Секунды.

Возвращаемые значения:

DINT	1 – если выполнение прошло успешно; 0 – если произошла ошибка.

Пример:

```
ret := SetTime(2010, 7, 13, 16, 23, 10); (* Устанавливает время 16:23:10 13/07/2010 *)
```

WDT_Reset

Описание: Сброс счетчика watchdog таймера.

Аргументы:

desc	DINT	Дескриптор watchdog таймера.

Возвращаемые значения:

DINT	1 – если выполнение прошло успешно; 0 – если произошла ошибка.

Пример:

```
ret := WDT_Reset(wdt_desc); (* Сбрасывает счетчик watchdog таймера *)
```

WDT_Set

Описание: Инициализирует watchdog таймер и задает время таймаута. Если до истечения этого времени не будет выполнена функция *WDT_Reset*, произойдет аппаратная перезагрузка контроллера.

Аргументы:

timeout	DINT	Время таймаута в секундах.

Возвращаемые значения:

DINT	Дескриптор watchdog таймера. Если значение больше нуля, инициализация прошла успешно.

Пример:

```
wdt_desc := WDT_Set(5); (* инициализирует watchdog таймер и устанавливает время таймаута 5сек. *)
```

2 УСТРОЙСТВА ВВОДА-ВЫВОДА

MODBUS-RTU

Для организации обмена данными прикладной программы ISaGRAF с внешним оборудованием по шине RS-485 применяются драйверы, реализующие протокол MODBUS-RTU. Название набора драйверов – ***mbtru_:master_:*** Этот набор состоит из следующих устройств:

- Устройство линии MODBUS ***modbus_line(* *)*** служит для открытия и инициализации линии для устройств MODBUS. Это устройство должно присутствовать для каждой используемой линии RS-485 для соединения по протоколу MODBUS. Параметры устройства ***modbus_line(* *)*** следующие:

baudrate – скорость соединения, бит/с. Допустимые значения: 110, 300, 600, 1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200, 230400, 460800, 921600 бит/с.

parity – контроль четности. Допустимые значения: none, even, odd.

stop_bits – число стоп-битов. Допустимые значения: 1, 2.

nc – номер физической линии. Допустимые значения: 0, 1, 2, 3. Значению 0 соответствует линия RS485-1, 1 – RS485-2, 2 – RS485-3, 3 – RS232.

- Набор комплексных устройств MODBUS Master для реализации связи с подчиненными устройствами, подключенными к линии RS-485:

cpx_bool_in(* *) – чтение дискретных данных типа BOOL от подчиненного устройства (функция modbus 02 (0x02));

cpx_bool_out(* *) – запись дискретных данных типа BOOL в подчиненное устройство (функция modbus 15 (0x0F));

cpx_int_in(* *) – прием данных типа UINT от подчиненного устройства (функция modbus 04 (0x04) или 03 (0x03) задается параметром при конфигурации устройства);

cpx_int_out(* *) – передача данных типа UINT подчиненному устройству (функция modbus 16 (0x10)).

Параметры каждого из комплексных устройств MODBUS Master следующие:

nc – номер физической линии от 0 до 3 на которой установлено подчиненное устройство;

node – адрес подчиненного устройства от 1 до 247;

base_address – начальный адрес данных от 0 до 65535;

empty_cycles – число пустых циклов между запросами от 0 до 65535;

timeout – таймаут ожидания ответа от подчиненного устройства в миллисекундах от 1 до 65535.

func – выбор функции MODBUS для устройства cpx_int_in 04 или 03;

В составе каждого комплексного устройства MODBUS Master присутствуют простые устройства:

bool_in(* *) в составе ***cpx_bool_in(* *)*** для монтажа дискретных данных на ввод.

bool_out(* *) в составе ***cpx_bool_out(* *)*** для монтажа дискретных данных на вывод.

int_in(* *) в составе ***cpx_int_in(* *)*** для монтажа данных типа UINT на ввод.

int_out(* *) в составе ***cpx_int_out(* *)*** для монтажа данных типа UINT на вывод.

mbstatus(* *) для определения состояния обмена. Монтируется на ввод в структуру пользовательского типа ***mbstat***. Описание структуры см. ниже.

mbenable(* *) для разрешения приема/передачи соответствующего типа данных от подчиненного устройства. Монтируется как канал на вывод типа BOOL. При установке состояния канала в значение TRUE происходит обмен с подчиненным устройством.

Вызов устройств на ввод происходит в начале каждого цикла прикладной программы, а на вывод – в конце каждого цикла. Таким образом, периодичность опроса подчиненных устройств задается временем цикла прикладной программы. Если в комплексном устройстве

задать параметр empty_cycles, то обращение к подчиненному устройству будет производиться через казанное число циклов.
Обращение к выходным устройствам происходит только в том случае, если выходное значение изменилось.

- Комплексное устройство ***cpx_slave(* *)*** служит для реализации связи по протоколу MODBUS ведущего устройства с ПЛК. Данное комплексное устройство имеет следующие параметры:

nc – номер физической линии от 0 до 3 к которой подключено ведущее устройство;

node – сетевой адрес ПЛК от 1 до 247;

timeout – минимальная задержка ответа в мс. после получения запроса от ведущего устройства. Допустимые значения: от 0 до 65535 мс.

В составе комплексного устройства ***cpx_slave(* *)*** присутствуют следующие простые устройства:

slave_receiver(* *) – для прослушивания линии;

slave_bool_in(* *) – для приема переменных типа BOOL с направлением на ввод (функции modbus 05 (0x05) и 15 (0x0F));

slave_bool_out(* *) – для передачи переменных типа BOOL с направлением на вывод (функции modbus 02 (0x02));

slave_int_in(* *) – для приема переменных типа UINT с направлением на ввод (функции modbus 06 (0x06) и 16 (0x10));

slave_int_out(* *) – для передачи переменных типа UINT с направлением на вывод (функции modbus 03 (0x03) и 04 (0x04));

mbstatus(* *) для определения состояния обмена. Монтируется на ввод в структуру пользовательского типа ***mbstat***. Описание структуры см. ниже.

mbenable(* *) для разрешения приема/передачи данных от ПЛК. Монтируется как канал на вывод типа BOOL. При установке состояния канала в значение TRUE происходит обмен с ведущим устройством. В противном случае ПЛК на запросы отвечает исключением SLAVE DEVICE BUSY (код 06).

Для простых устройств обмена данными ***slave_bool_in(* *)***, ***slave_bool_out(* *)***, ***slave_int_in(* *)*** и ***slave_int_out(* *)*** задается параметр base_address со значением базового адреса данных для запросов ведущего устройства.

Прием запроса от ведущего устройства происходит в начале каждого цикла прикладной программы. Ответ устройств драйвера с направлением на ввод производится после приема запроса, а с направлением на вывод – после окончания каждого цикла прикладной программы. Таким образом, время реакции ПЛК на запрос ведущего устройства может составлять до значения времени цикла прикладной программы, плюс время, заданное в параметре timeout.

Для получения информации о состоянии обмена по протоколу MODBUS служит простое устройство драйвера ***mbstatus(* *)***. Это устройство имеет канал с направлением на ввод с типом структуры пользователя ***mbstat.*** Данная структура имеет следующие значения полей:

err – код состояния типа SINT. Значения кода состояния:

err	Состояние
> 0	Обмен завершился успешно.
0	Общая ошибка.
-1	Принятый код функции не может быть обработан на подчиненном устройстве.
-2	Адрес данных указанный в запросе недоступен данному подчиненному.
-3	Величина, содержащаяся в поле данных запроса, является недопустимой величиной для подчиненного.
-4	Невосстанавливаемая ошибка имела место, пока подчиненный пытался выполнить затребованное действие.
-5	Подчиненный принял запрос и обрабатывает его, но это требует много време-

	ни.
-6	Подчиненный занят обработкой команды.
-7	Подчиненный не может выполнить программную функцию, принятую в запросе.
-8	Ошибка паритета чтения памяти.
-11	Ошибка порта.
-12	Ошибка контрольной суммы пакета.
-13	Таймаут.
-14	Подчиненный не найден в таблице устройств.
-15	Адрес запрашиваемого подчиненного не совпадает с адресом, заданным для ПЛК.

cnt_req – общее число запросов. Тип данных – UDINT;

cnt_crc_err – число ошибок CRC. Тип данных – UDINT;

cnt_timo_err – число ошибок таймаута подчиненного устройства. Тип данных – UDINT;

resp_time – время ответа подчиненного устройства в мс. Тип данных – UDINT;

В каждом канальном устройстве ***cpx_bool_in(* *)***, ***cpx_bool_out(* *)***, ***cpx_int_in(* *)***, ***cpx_int_out(* *)***, ***slave_bool_in(* *)***, ***slave_bool_out(* *)***, ***slave_int_in(* *)*** и ***slave_bool_out(* *)*** задано значение максимального числа обслуживаемых каналов. При использовании ISaGRAF Workbench с различными лицензиями на число каналов, значения максимального числа обслуживаемых каналов устройств можно изменять утилитой Построителя целевого определения TDBuild, входящий в состав ISaGRAF Workbench.