СПЕЦИАЛИЗИРОВАННОЕ КОНСТРУКТОРСКОЕ БЮРО
ПРОМАВТОМАТИКА
www.skbpa.ru
124460, г. Москва, Зеленоград, а/я 18; e-mail: isagraf@skbpa.ru; тел./факс: (495) 221-91-65

Контроллеры с поддержкой функций программируемой логики ПЛК-166.02 и ПЛК-84.М1

(версия 1.31)

Руководство по программированию.
Особенности реализации системы.

Служба поддержки: isagraf@skbpa.ru

г. Зеленоград

2003г.

Оглавление

Демонстрационный режим ПЛК для контроллеров ТК-84.М1[/2] и ТК-166.02

Приложения ISaGRAF будут полноценно выполняться только на контроллерах, лицензированных фирмой СКБ ”Промавтоматика” (ПЛК-84М1[/2] и ПЛК-166.02 или модернизированных ТК-84М1 или ТК-166.02). При загрузке программы для ПЛК в нелицензированный контроллер ТК-84М1[/2] или ТК-166.02, ядро ISaGRAF Target будет работать в демонстрационном режиме со следующими ограничениями:

- ***пользовательская программа выполняется только 10 минут после загрузки в контроллер или после аварии питания;***
- ***не поддерживается резервное сохранение пользовательской программы в энергонезависимой памяти;***

Установка и настройка ISaGRAF Workbench.

Программируемые контроллеры совместимы с версиями ISaGRAF 3.3X - 3.5X.

СКБ «Промавтоматика» поставляет адаптированную версию ISaGRAF 3.4 и при её установке все библиотеки, компоненты и примеры программ устанавливаются автоматически.

Если вы пользуетесь стандартной версией ISaGRAF или версией, поставляемой под какой либо другой контроллер, то для получения совместимой с ПЛК-166.02 и ПЛК-84.М1 версии необходимо проделать следующие действия:

Установить ISaGRAF Workbench без библиотек поддержки аппаратуры третьих фирм.

Библиотеки устройств, поддерживаемых контроллерами ПЛК-166.02 и ПЛК-84.М1, устанавливать вручную из директории LIBRARY следующим образом:

- В окне ISaGRAF-Управление проектами выберите меню ***Инструменты >> Библиотеки***.
- В открывшемся окне выберите тип библиотеки: «Комплексное оборудование».
- Выбираете меню ***Инструменты >> Архив***
- В появившемся окне, нажав кнопку ***смотреть***, выберите путь к архиву библиотек.
- Если Вы правильно выбрали путь, то в списке ***Архив*** появится список поддерживаемых устройств. Выделите все устройства и нажмите кнопку ***Восстановить***.
- В списке ***Система разработки*** должны появиться соответствующие библиотечные элементы.

Файлы из директории ISA_EXE следует скопировать в папку ISAWIN\EXE.

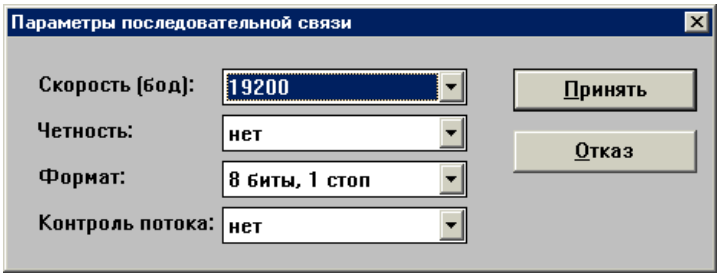
Примеры приложений находятся в папке EXAMPLE. Импортировать их в систему можно через меню ***Инструменты >> Архив >> Проекты***.

Прошивки ядра для контроллеров находятся в папке TARGET.

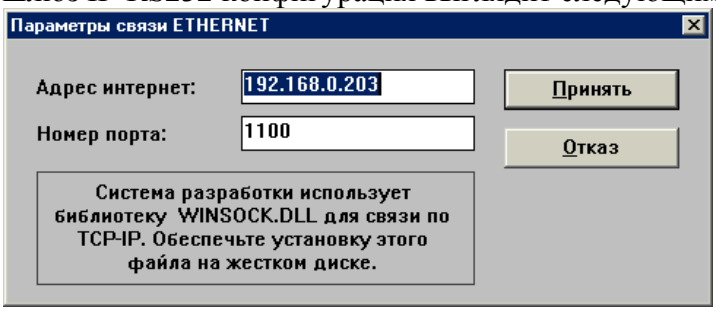
Интерфейс ПЛК – ISaGRAF Workbench.

Загрузка прикладной задачи в контроллер и отладка выполняется через интерфейс RS-232 по протоколу MODBUS или через шлюз IP-RS232.

Параметры связи при работе через RS232: скорость 19200, чётности нет, один стоп-бит.

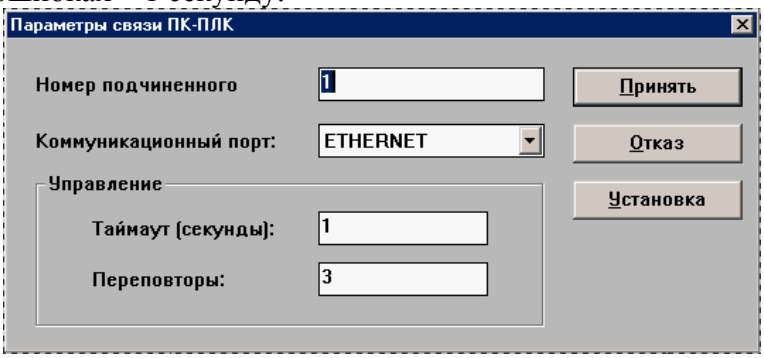


При работе через шлюз IP-RS232 конфигурация выглядит следующим образом:



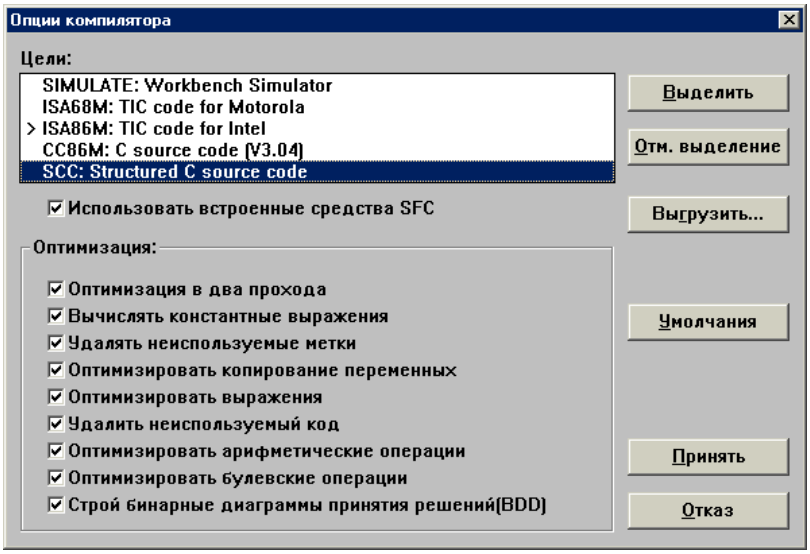
Адрес интернет должен соответствовать IP адресу шлюза, через который будет производиться отладка.

Номер подчиненного должен соответствовать номеру, заданному в контроллере для отладчика ISaGRAF. Число переповторов рекомендуется устанавливать равным 3, а таймаут связи при ошибках – 1 секунду.



Если через этот же канал и с теми же параметрами подключается SCADA по протоколу MODBUS, возможно использование адаптеров RS-232/RS-485 для перехода на интерфейс RS-485. На компьютер, в этом случае, ставят адаптер ***A232/485-PC***, а на ПЛК – адаптер ***A232/485***.

Для корректной работы приложения, в опциях компилятора необходимо выбрать **ISA86M: TIC code for Intel** и поддержку программ SFC. Опции оптимизации на работу и размер приложения оказывают влияние только в лучшую сторону, поэтому их рекомендуется включить.



OPC сервер для ПЛК.

OPC сервер позволяет очень просто получить интерфейс ко всем переменным приложения ISaGRAF, что значительно упрощает интеграцию ПЛК в современные SCADA системы. Связь OPC сервер – ПЛК может быть обеспечена как напрямую через порт RS-232 так и через шлюз IP-RS232. OPC сервер является бесплатным и доступен на сайте www.altersys.com.

Ограничения реализации.

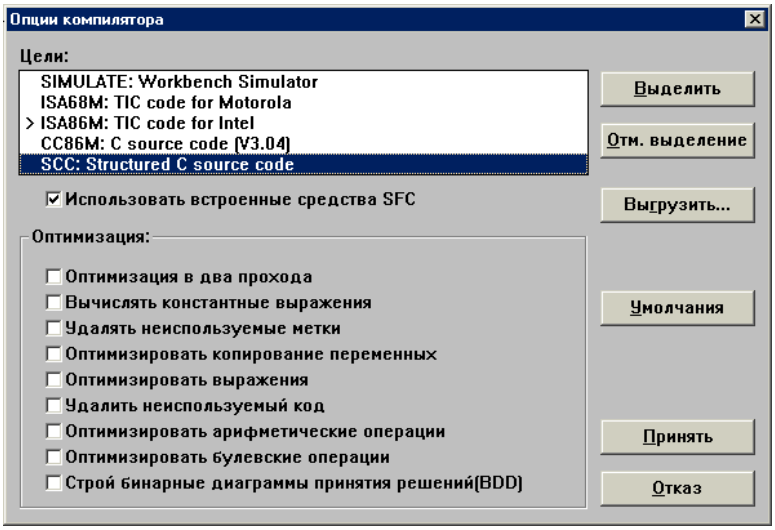
Ограничение на размер кода приложения – 64 Кбайта. Размер приложения можно увидеть, посмотрев на размер файла **appli.x8m** в директории проекта. Этот параметр, также, можно увидеть при загрузке приложения в окне сообщений отладчика.

Максимальное количество переменных приложения можно вычислить следующим способом:

На переменные целого, вещественного и булевского типа выделен сегмент 16 Кбайт. Переменные целого и вещественного типа занимают по 4 байта, а переменные булевского–по одному байту.

Если учесть, что ISaGRAF Workbench ограничивает суммарное количество переменных целого и вещественного типа числом 4096, то данная реализация почти не урезает возможностей системы.

Если приложение использует большое количество строковых преобразований, возможен эффект резервирования слишком большого количества строковых переменных, что иногда приводит к нерациональному использованию памяти контроллера и её переполнению. Эту ситуацию можно определить по сообщению отладчика после загрузки приложения. Проблема разрешается снятием всех опций оптимизации кода приложения.



В версии ISaGRAF 3.5 представители компании AlterSys обещали устранить эту неприятную особенность компилятора.

Периферийные устройства ввода/вывода ПЛК.

Иногда, встроенных каналов ввода вывода ПЛК недостаточно для решения поставленной задачи. В таких случаях применяют периферийные устройства расширения ввода/вывода. Все периферийные модули расширения датчиков и устройства ввода-вывода подключаются к ПЛК через интерфейс RS-485 по протоколу MODBUS. Параметры связи с каждым устройством (скорость, чётность и номер клиента) описываются в прикладной задаче. Диапазон скоростей: от 1200 до 57600 бод. Всего на один контроллер можно подключить до 255 устройств.

ПЛК поддерживают следующие периферийные устройства ввода-вывода:

КР-Д16А8

КР-16Р

КР-8А

ПИК-УВП

Программируемая панель ввода-вывода с графическим дисплеем.

УВП-280А

УВП-280Б

Возможно создание модулей ввода-вывода для любых устройств, поддерживающих протокол MODBUS RTU на интерфейсе RS-485 или RS-232.

Для получения доступа к встроенным ресурсам ввода-вывода программируемого контроллера необходимо при описании устройств ввода-вывода установить устройство, соответствующее типу программируемого контроллера по адресу **0**.

Тип контроллера	Название устройства в библиотеке
ПЛК-166.02	PLC16602
ПЛК-84М1	PLC84M1
ПЛК-84М1/2	PLC84M12

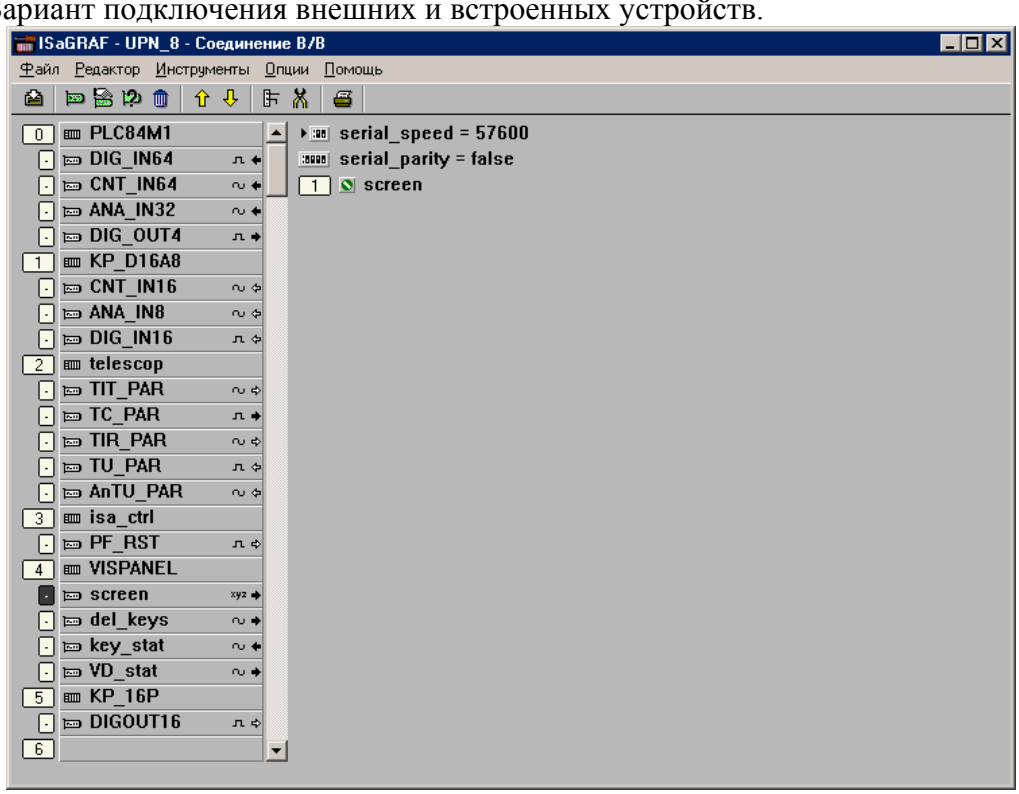
Для управления способом запуска пользовательской задачи после сбоя питания контроллера необходимо добавить устройство **ISA_CTRL** на любой свободный адрес. После этого запись и обновление переменной, подключенной к параметру **PF_RST,** влияет на работу следующим образом:

Значение *TRUE* – контроллер перезапустит программу пользователя и проинициализирует все переменные.

Значение *FALSE* – контроллер продолжит выполнять программу с того места, где она была прервана сбоем питания. Значения переменных сохраняются.

Внимание! Ознакомьтесь с разделом *Особенности работы выходов телеуправления (ТУ) на ПЛК-84М1, ПЛК-84М1/2, ПЛК-166.02 и КР-16Р.*

Вариант подключения внешних и встроенных устройств.



Здесь показана конфигурация соединений ввода-вывода для контроллера ПЛК-166.02. Как можно видеть в левой части окна, встроенные ресурсы контроллера подключены к адресу 0. Этот адрес не может использоваться для работы с внешней шиной, так как протокол MODBUS не поддерживает индивидуальное обращение к сетевым контроллерам через адрес 0. Поэтому на этот адрес нельзя конфигурировать контроллеры, работающие через внешнюю шину. На примере, контроллер КР-Д16А8 подключен по адресу 1 и работает как внешнее расширение на 16 цифровых и 8 аналоговых входов, а КР-16Р подключен по адресу 5 и обеспечивает расширение на 16 релейных выходов. В правой части окна отображаются присоединённые переменные выбранного устройства или группы, а также параметры обмена, если это устройство подключено к внешней шине расширения. В нашем случае, устройство VisPanel (Универсальная графическая панель отображения) подключено к шине под 4 адресом клиента и имеет следующие параметры обмена: скорость 57600, четность выключена. Пользователь может произвольно размещать устройства ввода-вывода в адресном пространстве и настраивать для каждого параметры связи.

Работа с переменными ввода-вывода.

Все операции обновления переменных ввода-вывода происходят только при помощи вызова стандартной функции **OPERATE**. Формат её вызова выглядит так:

Result:=OPERATE(ActionVariable, param1,param2), где

ActionVariable—переменная над которой будет выполняться операция ввода-вывода, т.е. если эта переменная объявлена как вход, то будет опрошено устройство, к которому эта переменная привязана и результат будет присвоен этой переменной. Для выходной переменной происходит обратное (значение переменной присваивается полю устройства ввода-вывода).

Param1—индекс операции которая будет произведена над устройством ввода-вывода.Он может принимать следующие значения:

1–стандартная операция ввода-вывода.

10–снятие статуса аварии питания для контроллера расширения ввода/вывода.

Param2—опции операции ввода-вывода для внешних устройств, подключенных по протоколу MODBUS. Для встроенных устройств ввода-вывода контроллера этот параметр игнорируется. Старшие 16 бит указывают количество попыток обмена с ведомым устройством в случае отсутствия ответа или при ошибке в принятом пакете. В случае нулевого значения производится одна попытка. Младшие 16 бит указывают, сколько миллисекунд контроллер ждет до начала ответной посылки от ведомого контроллера. В случае нулевого значения, если ответ не последовал через 4T, то обмен считается несостоявшимся. T-длительность одного символа на данной скорости.

Пример:

Param2 = 16#00030064, озачает, что будет предпринято 3 попытки обмена и начало ответа будет ожидаться 4T+100 миллисекунд.

Result—статус обмена с внешним устройством ввода-вывода.

AND_MASK (Result,128)=128 – устройство ещё не выполнило предыдущую команду или не готово.

AND_MASK (Result,64)=64 – ошибка обмена. Контроллеру не удалось выполнить обмен по причине отсутствия связи (исчерпано заданное количество попыток).

AND_MASK (Result,32)=32 – ошибка питания внешнего устройства. Это означает, что внешнее устройство настроено на режим обработки аварий питания и произошёл сбой питания. Для перевода контроллера в штатный режим обмена необходимо дать команду снятия аварии питания (см. **param1=10**).

AND_MASK (Result,31) – оставшееся число попыток обмена. С помощью этого параметра можно контролировать качество связи с конкретным контроллером.

Для любой целой или вещественной переменной ввода-вывода может быть применена таблица преобразования шкалы. Она назначается непосредственно в окне описания переменной. Преобразование шкалы при помощи функций не поддерживается!

Если устройство ввода-вывода содержит серию однотипных входов или выходов, то в процессе обмена будут обработаны все элементы этой группы. Например, если функция **OPERATE** получила команду обработать один из аналоговых токовых входов контроллера расширения КР-Д16А8, то в результате будут опрошены все аналоговые входы этого контроллера и результат появится во всех переменных, привязанных к аналоговым входам этого контроллера.

Опрос устройств, не использующих внешнюю шину расширения и протокол MODBUS, происходит непосредственно, и не нарушает стандартного цикла выполнения задач. При работе нескольких задач с внешними устройствами через общую шину, арбитраж происходит следующим образом: задача, первой начавшая обмен через внешнюю шину ждет завершения обмена для обработки полученных данных. Если в это время еще одна задача претендует на обмен, то она замораживается и ждёт освобождения шины первой задачей. После окончания обмена первая задача освобождает шину, при этом, если в её коде сразу следует еще один запрос на работу с внешней шиной, то приоритет к захвату шины отдаётся конкурирующей задаче, если она претендует на захват шины.

Подключение MODBUS SCADA.

Параметры связи соответствуют параметрам обмена с ISaGRAF Workbench: скорость 19200, чётности нет, один стоп-бит.

При описании карты адресов MODBUS для SCADA в пакете ISaGRAF Workbench переменные типа Timer, Integer и Real необходимо размещать только по четным адресам, так как реально они являются 32-х разрядными и занимают по 2 регистра. Переменные типа Real необходимо размещать на адресах, больших 1000hex, а переменные типа Integer—на адресах до 1000hex.
Переменные типа Real при описании в SCADA имеют тип Float, а переменные типа Integer и Timer – тип Longint. Для переменных типа Real, Integer, Timer поддерживается запись со стороны SCADA. Переменные типа Message допускают только чтение.
Все переменные небулевского типа читаются как командой 03, так и командой 04. Переменные булевского типа читаются командами 01 и 02.
Пример:

Тип переменной	Адрес в карте адресов MODBUS для SCADA	Описание в WonderWare Intouch
Boolean	1	адрес 2 D или 1 DI
Integer	2h	адрес 40003 L или 30003 L или 2 HRL или 2 IRL
Message	4	адрес 30005-30010 M или 40005-40010 M
Real	1000h	адрес 44097 F или 34097 F или 4096 HRF или 4096 IRF

Работа со SCADA “Телескоп+”.

Работа по радиоканалу или по физической выделенной линии со SCADA “Телескоп+” поддерживается следующим образом.

Для обмена данными с вышеуказанной системой, в окне описания соединений ввода/вывода в системе ISaGRAF Workbench подключается виртуальное комплексное устройство ввода-вывода TELESCOP, к которому могут быть подключены переменные ввода-вывода с целью мониторинга со стороны SCADA. Для SCADA “Телескоп+” это виртуальное устройство представляется как максимально возможный набор модулей расширения входов/выходов и подлежит штатной обработке со стороны SCADA. Поля этого устройства обновляются также при помощи стандартной функции **OPERATE**.

Изменение времени фильтрации входов телесостояния (ТС) на ПЛК-84М1[/2] и ПЛК-166.02.

По умолчанию, время фильтрации встроенных цифровых входов ПЛК-84М1 и ПЛК-166.02 задаётся из программы локального пульта управления или приходит в команде инициализации от SCADA “Телескоп+”. При задании времени фильтрации с помощью локального пульта, можно установить опцию запрета инициализации каждого конкретного входа из сети телемеханики. Установка времени фильтрации цифровых входов также поддерживается и из прикладной программы ISaGRAF. Для этого предусмотрена функция ***TC_FLTR***. Она имеет следующий формат:

Result:=TC_FLTR(Num_Inp, Filter_time, disable_TM);

Num_Inp – номер цифрового входа, если этот параметр равен 0, то процедура установки времени фильтрации будет применена ко всем цифровым входам контроллера.
Filter_time – время фильтрации в миллисекундах.
disable_TM – значение TRUE означает, что попытки изменения времени фильтрации со стороны SCADA “Телескоп+” для этого входа будут игнорироваться.
Result – код ошибки:

0-операция успешно выполнена.
1-некорректное время фильтрации.
2-неверный номер входа.

Выдача состояния ТИТ в SCADA “Телескоп+” по инициативе прикладной программы ПЛК.

По умолчанию, состояние входов телеизмерения выдаётся на сервер либо с заданным периодом, либо по запросу оператора, либо по нарушению уставок. Для выдачи значений ТИТ по инициативе прикладной программы применяются функции:

dummy:INTEGER:=TIT_PUTI(VarToSend:INTEGER);

dummy:INTEGER:=TIT_PUTR(VarToSend:REAL);

Переменная VarToSend должна быть присоединена в пространстве внутренних аналоговых входов контроллера(PLC84M1=>ANA_IN32), либо в пространстве виртуальных аналоговых входов интерфейса SCADA “Телескоп+” (TELESCOP=>TIT_PAR). В противном случае, в окно отладчика будет выдана соответствующая ошибка, и операция будет проигнорирована.

Особенности работы выходов телеуправления (ТУ) на ПЛК-84М1[/2], ПЛК-166.02 и КР-16Р.

Встроенные каналы телеуправления ПЛК имеют задержку на включение порядка 20мс. Она обусловлена, в основном, временем реакции реле.

Для внешних устройств управления (КР-16Р) время реакции на включение рассчитывается следующим образом:

Тту[мс]=100000/Vmodbus+40мс, где

Vmodbus – скорость интерфейса с этим устройством в битах/секунду.

В случае, если программа допускает одновременный опрос других устройств по внешней шине, то время реакции может увеличиться на время опроса этих устройств. Поэтому, для критичных ко времени реакции ТУ задач лучше использовать внутренние выходы телеуправления ПЛК или оптимизировать обращение к внешним устройствам ввода вывода в плане наименьшей нагрузки на шину. Например, увеличить период опроса и/или поднять скорость интерфейса.

При пропадании питания, КР-16Р приводит все выходы в исходное состояние и при появлении не восстанавливает их состояние, имевшее место на момент аварии питания. Авария питания любого периферийного контроллера может быть обнаружена по значению, возвращаемому OPERATE (см.выше). О настройке периферийных устройств на режим отработки аварии питания можно узнать в руководстве по пусконаладке этих устройств.

При пропадании питания, ПЛК-84М1 или ПЛК-166.02 приводит все выходы в исходное состояние и при появлении восстанавливает их состояние, имевшее место на момент аварии питания. Если это, по каким-либо причинам Вас не устраивает, то не используйте опцию продолжения приложения после аварии питания.

Работа с массивами.

Реализация стандартных функций работы с массивами ARCREATE, ARREAD и ARWRITE – стандартна со следующими дополнениями:

- Введены еще три дополнительные функции ARCREATE_R, ARREAD_R и ARWRITE_R, обеспечивающие аналогичный доступ к массивам, но уже для вещественного типа.
- Размер массивов с индексами с 0 по 15 ограничен размером остатка сегмента приложения (64 Кбайта).
- При рестарте контроллера после сбоя питания, информация в массивах, расположенных в ОЗУ, не очищается, но рассчитывать на её достоверность не стоит, т.к. механизмы, проверяющие сохранность этой области памяти отсутствуют.

- Индексы массивов от 16 до 23 соответствуют массивам, располагаемым в энергонезависимой памяти EEPROM. Их суммарный максимальный размер равен 1600 ячеек. Ресурс микросхемы соответствует 100000 записей, поэтому при записи редко меняющихся значений лучше предварительно считать нужную ячейку и записывать в неё только в случае изменения значения.
- При чтении и записи какой-либо задачей массивов, расположенных в энергонезависимой памяти, все остальные задачи, претендующие на обращение к EEPROM, останавливаются до завершения операции чтения/записи.

Извлечение и упаковка битовых переменных.

Нередко, в прикладной программе появляется необходимость расшифровки переменной, содержащей битовую маску или установки битов в переменной.
Безусловно, все эти функции можно реализовать с помощью функций AND_MASK и OR_MASK, но в этом случае программа хуже читается, имеет больший размер кода и большее время выполнения. Для решения этой задачи разработан ряд нестандартных функций для упаковки и распаковки битовых переменных, как поодиночке, так и группами.

Функция **dummy:INTEGER:=SET_BIT(bitmask:INTEGER, pos:INTEGER, bit:BOOLEAN)**

Устанавливает бит с номером **pos** переменной **bitmask** в значение переменной **bit**. Младший бит имеет номер 0. Переменная **pos** может принимать значения от 0 до 31.

Функция **dummy:BOOLEAN:=GET_BIT(bitmask:INTEGER, pos:INTEGER)**

Возвращает значение бита с номером **pos** переменной **bitmask**. Младший бит имеет номер 0. Переменная **pos** может принимать значения от 0 до 31.

Функция **result:INTEGER:=PACK_BIT(bitmask:INTEGER, pos:INTEGER, bit0..bit7:BOOLEAN)**

Устанавливает 8 бит начиная с позиции под номером **pos** переменной **bitmask** в значения группы переменных **bit0, bit1,... bit7**. Значение переменной **bit7** кладётся в позицию **pos-8,** значение переменной **bit0** кладётся в позицию **pos-1**. Остальные разряды переменной **bitmask** остаются неизменными. Младший бит имеет номер 0. Переменная **pos** может принимать значения от 0 до 31. Результат доступен в переменной **result**.

Например, вызов этой функции в следующем контексте:
bitmask:=PACK_BIT(bitmask, 8, true, true, true, false, false, false, false, true);
при начальном bitmask равном 0, приведёт к состоянию переменной **bitmask = 16#00E1= 11100001b.**

Функция **dummy:INTEGER:=UNPK_BIT(bitmask:INTEGER, pos:INTEGER, bit0..bit7:BOOLEAN)**

Извлекает 8 бит переменной **bitmask,** начиная с позиции под номером **pos** в группу переменных **bit0, bit1,... bit7**. Значение бита в позицию **pos-8** назначается переменной **bit7,** значение бита в позицию **pos-1** помещается в переменную **bit0** . Младший бит имеет номер 0. Переменная **pos** может принимать значения от 0 до 31.

Например, вызов этой функции в следующем контексте:
dummy:=PACK_BIT(bitmask, 8, bit0, bit1, bit2, bit3, bit4, bit5, bit6, bit7);

при начальном bitmask равном **16#00C3=11000011b**, приведёт к следующему состоянию битовых переменных: **bit0=true, bit1=true, bit2=false, bit3=false, bit4=false, bit5=false, bit6=true, bit7=true**.

Сохраняемые переменные.

Нередко может понадобиться, чтобы после рестарта контроллера и приложения ISaGRAF в частности, некоторые переменные сохраняли свои значения. Для этого, при описании такой переменной необходимо установить флажок «хранить». Переменные с атрибутом сохранения не могут иметь начальное значение, и поэтому необходимо в приложении писать код инициализации по какому-нибудь событию.

В ПЛК-84М1 и ПЛК-166.02 реализована двойная буферизация сохраняемых переменных (в ОЗУ и в EEPROM). Все необходимые для сохранения и восстановления переменных функции должны вызываться из пользовательской программы. Автоматически никакие действия над сохраняемыми переменными не выполняются!

Функции сохранения и восстановления вызываются при помощи системной функции SYSTEM.

Формат вызова:

Rez:=SYSTEM(Команда, параметр);

Команды:

1. SYS_INITANA, SYS_INITBOO, SYS_INITMSG, SYS_INITTMR.
   Инициализирует переменные, имеющие значение по умолчанию.

2. SYS_RESTANA, SYS_RESTBOO, SYS_RESTMSG, SYS_RESTTMR.
   Если параметр равен «0» -- производится проверка контрольной суммы для переменных соответствующего типа. Если контрольная сумма соответствует, то возвращается ненулевое значение.

   Если параметр равен «1» -- производится восстановление сохраняемых переменных из энергонезависимого ОЗУ и проверка контрольной суммы. Если контрольная сумма соответствует, то возвращается ненулевое значение.

   Если параметр равен «2» -- производится восстановление сохраняемых переменных из EEPROM проверка контрольной суммы. Если контрольная сумма соответствует, то возвращается ненулевое значение.

3. SYS_SAVANA, SYS_SAVBOO, SYS_SAVMSG, SYS_SAVTMR
   Если параметр равен «0» -- обновление контрольной суммы для переменных соответствующего типа.

   Если параметр равен «1» -- производится сохранение переменных с аттрибутом сохранения в энергонезависимом ОЗУ.

   Если параметр равен «2» -- производится сохранение переменных с аттрибутом сохранения в энергонезависимом ОЗУ и EEPROM.

Фрагменты программы, обеспечивающие работу с сохраняемыми переменными вещественного и целого типа.

Обновление информации в ОЗУ и EEPROM.

```
IO_REZ:=SYSTEM(SYS_SAVANA,2);   (* сохранить контр. сумму в ОЗУ и EEPROM *)
```

Проверка корректности и восстановление в случае необходимости.

```
IF SYSTEM(SYS_RESTANA,0)=0  THEN;  (* проверить CRC области переменных*)
    IF SYSTEM(SYS_RESTANA,1)=0 THEN;  (* восстановить из RAM и проверить CRC *)
        IF SYSTEM(SYS_RESTANA,2)=0 THEN; (* восстановить из EEPROM и проверить CRC*)
                    (* загрузить значения по умолчанию *)
        END_IF;
    END_IF;
END_IF;
```

Поддержка протокола MODBUS Master
из пользовательской программы.

Эти функции не являются стандартными для **ISaGRAF Workbench** и введены с целью упрощения поддержки интеллектуальных устройств третьих фирм, работающих с протоколом Modbus в качестве ведомого устройства.
С помощью этих функций могут быть описаны любые, даже нестандартные команды. Ведомое устройство может работать с любой скоростью, с чётность и без неё. **Не** поддерживается формат данных с нечётной чётностью.
Для организации протокола MODBUS Master, со стороны ПЛК поддерживается ряд специальных функций. Работают они с входным и выходным буферами. Эти буферы организуются в переменных строкового типа. Максимальная длина этих переменных должна быть больше максимального принимаемого или передаваемого пакета. Лучше её поставить равной 255 символам.
Последовательность работы с буферами должна быть следующей:

1) формирование в выходном буфере команды запроса ведомому устройству при помощи функций упаковки переменных различного типа.
2) Выполнение обмена при помощи вызова функции **MBSTR_EX**. И первичный анализ ответа по длине ответа.
3) Разбор полученного ответа при помощи функций распаковки данных из входного буфера в переменные.

Для упаковки данных в выходной буфер предназначены следующие функции:

dummy:INTEGER:=SPUTBYTE(outbuf:MESSAGE,pos:INTEGER,byte:INTEGER)
Упаковывает младшие 8 бит параметра **byte** целого типа в строковую переменную **outbuf**, выполняющую роль выходного буфера. Смещение в буфере задаёт параметр **pos.**

dummy:INTEGER:=SPUTWORD(outbuf:MESSAGE,pos:INTEGER,word:INTEGER,seq:INTEGER)
Упаковывает младшие 16 бит параметра **word** целого типа в строковую переменную **outbuf**, выполняющую роль выходного буфера. Смещение в буфере задаёт параметр **pos.** Порядок упаковки байт определяется параметром **seq**. При **seq=16#0012** старший байт кладётся в буфер первым, при **seq=16#0021** в буфер кладётся первым младший байт.

dummy:INTEGER:=SPUTINT(outbuf:MESSAGE,pos:INTEGER,dword:INTEGER,seq:INTEGER)
Упаковывает 32 бита параметра **dword** целого типа в строковую переменную **outbuf**, выполняющую роль выходного буфера. Смещение в буфере задаёт параметр **pos.** Порядок упаковки байт определяется параметром **seq**. При **seq=16#1234** старший байт кладётся в буфер первым, при **seq=16#4321** в буфер кладётся первым младший байт. Также допустимы любые другие перестановки байт. Например: если dword имеет значение **16#012345678** и **seq=16#2143**, то в буфер попадёт последовательность **23 01 78 56 hex**.

dummy:INTEGER:=SPUTREAL(outbuf:MESSAGE,pos:INTEGER,real:REAL,seq:INTEGER)
Фунуция аналогична **SPUTINT**, только входная переменная для упаковки имеет тип **REAL**. Все остальные параметры соответствуют функции **SPUTINT**.

Для распаковки данных из входного буфера предназначены следующие функции:

byte:INTEGER:=SGETBYTE(inbuf:MESSAGE,pos:INTEGER)
Распаковывает байт из входного буфера **inbuf** по адресу **pos** и возвращает его как результат вызова функции.

word:INTEGER:=SGETWORD(inbuf:MESSAGE,pos:INTEGER,seq:INTEGER)
Распаковывает двухбайтное слово из входного буфера **inbuf** со смещением **pos** и возвращает его как результат вызова функции. Порядок извлечения байт определяется параметром **seq**. При **seq=16#0012** старший байт извлекается из буфера первым, при **seq=16#0021** из буфера извлекается первым младший байт.

dword:INTEGER:=SGETINT(inbuf:MESSAGE,pos:INTEGER,seq:INTEGER)

Распаковывает четырёхбайтное слово из входного буфера **inbuf** со смещением **pos** и возвращает его как результат вызова функции. Порядок распаковки байт определяется параметром **seq**. При **seq=16#1234** старший байт извлекается из буфера первым, при **seq=16#4321** из буфера извлекается первым младший байт. Также допустимы любые другие перестановки байт. Например: если в буфере лежит последовательность **12 34 56 78 hex** и **seq=16#4321**, то в функция вернёт значение **16#78563412**

real:REAL:=SGETREAL(inbuf:MESSAGE,pos:INTEGER,seq:INTEGER)

Функция аналогична **SGETINT**, только входное значение – имеет вещественный тип. Все остальные параметры соответствуют функции **SGETTINT**.

Для выполнения обмена с ведомым устройством вызывается функция **MBSTR_EX**. Её формат описан ниже:

```
counter:INTEGER:=MBSTR_EX(  outbuf:MESSAGE,
                            inbuf :MESSAGE,
                            len :INTEGER,
                            speed :INTEGER,
                            wait_answer :INTEGER;
                            attempt :INTEGER,
                            parity :BOOLEAN,
                            stopbits :BOOLEAN )
```

расшифровка параметров функции MBSTR_EX:

outbuf – пакет запроса ведомому устройству.

inbuf – в эту переменную помещается пакет с ответом.

len – длина запроса в байтах.

speed – скорость, на которой будет производиться обмен.

wait_answer – время ожидания ответа в миллисекундах. Если wait_answer=100, то ответ будет ожидаться в течение 4T+100 миллисекунд, где T-время передачи одного символа.

attempt – количество попыток обмена.

parity – чётность данных. false=чётности нет, true=чётность(even).

stopbits – количество стоп-битов. false= 1 стоп-бит, true= 2 стоп-бита.

Функция возвращает количество байт в ответе, принятом от ведомого устройства.

Так как входной и выходной буфера находятся в переменных строкового типа , что не позволяет просматривать их содержимое штатными средствами отладки ISaGRAF. Для упрощения анализа принятых или отправляемых пакетов была разработана функция, преобразующая данные из переменной-буфера в строковую переменную шестнадцатеричного формата, которая легко просматривается в отладчике.

result:MESSAGE:= SHEXCONV (buf:MESSAGE, pos:INTEGER, len:INTEGER);

расшифровка параметров функции SHEXCONV:

buf – переменная буфера обмена, подлежащая расшифровке,

pos – начальная позиция для расшифровки,

len – количество байт для расшифровки.

Пример использования описанных выше функций можно найти в архиве примеров. Он называется **MB_FREE**.

Уведомление управляющей программы об изменении переменных через протокол MODBUS верхнего уровня.

В некоторых приложениях необходимо иметь уведомление о изменении переменных, отображённых на адресное пространство протокола MODBUS верхнего

уровня. Для решения этой задачи применяется функция GETMBEX. Она имеет следующий формат:

MBmaster_info:INTEGER:=GETMBEX();

Переменная MBmaster_info содержит информацию о модификации регистров. Если функция вернула значение 0, то модификации с момента предыдущего вызова не произошло. Если значение не равно 0, значение переменной MBmaster_info расшифровывается следующим образом:

Признак источника,8 бит	Количество регистров, 8 бит	Начальный адрес регистров , 16 бит

Признак источника может принимать следующие значения:

1- модификация произошла через интерфейс RS-485.

2- модификация произошла через шлюз локального пульта управления.

3- модификация произошла через шлюз протокола «ПРОРЫВ».

Второе и третье поле представляют собой, соответственно, количество записанных регистров и их начальный адрес.

Фрагмент программы, использующий функцию GETMBEX с целью проверки на модификацию локальным пультом.

```
getAddr:=GETMBEX();
IF getAddr<>0 THEN
        ModAddr:=AND_MASK(getAddr,16#FFFF); (* маскируем адрес *)
        ModSize:=SHR(getAddr,16);           (* выдвигаем размер *)
        ModChannel:=SHR(ModSize,8);         (* получаем признак источника *)
        ModSize:=AND_MASK(ModSize,16#00FF); (* маскируем размер *)

        IF (ModAddr>=EE_begin) AND (ModAddr<EEt_end) THEN  (* проверяем на попадание в заданный диапазон *)
                IF ModChannel=2 THEN  (* модификация с локального пульта *)
                   …
                   …
                END_IF;
        END_IF;
END_IF;
```

В ядре, также, реализована возможность вычисления контрольной суммы переменных, закреплённых в адресном пространстве MODBUS.

Для подсчёта контрольной суммы булевских переменных от адреса **begin** до адреса **end** необходимо вызвать функцию:

CRC**:INTEGER:=**MBOO_CRC**(**begin:INTEGER, end:INTEGER**);**

Для аналоговых – функцию:

CRC**:INTEGER:=**MANA_CRC**(**begin:INTEGER, end:INTEGER**)**.

Параметры **begin** и **end** указывают на начальный и конечный адреса (включительно) в адресном пространстве MODBUS SCADA.

Получение серийного номера контроллера прикладной программой.

Для определения серийного номера контроллера, на котором исполняется приложение, применяется функция SERIAL_N. Она имеет следующий формат.

Serial_number:INTEGER:=SERIAL_N();

Обращение к переменным через индексную переменную.

В системе ISaGRAF отсутствует поддержка массивов переменных, что накладывает определённые ограничения на написание программ, работающих с большими объёмами однородных данных.

В частности, большие неудобства вызывает работа с большим количеством однородных входов или выходов. Порой, приходится копировать большие куски программы много раз, а отладка такой программы очень кропотливое занятие.

Чтобы облегчить написание таких программ, рекомендуется применять описанные ниже функции. Принцип индексации построен на том, что переменные ввода-вывода в памяти контроллера располагаются в том же порядке, как они прикреплены к устройству ввода-вывода. Если группа переменных, из которых нужно составить массив является внутренней, то её необходимо фиктивно назначить выходной и привязать к плате BOOL_ARR, если это булевская переменная или к плате ANA_ARR, если это переменная целого или вещественного типа. Каждый массив может содержать не более 128 элементов. Количество плат BOOL_ARR и ANA_ARR ограничено только максимальным количеством переменных ввода-вывода.
Любое обращение к массиву выполняется через ссылку на базовую (нулевую) переменную. Базовая переменная для группы должна быть выше всех переменных для этой группы. Базовая переменная доступна в массиве под нулевым индексом.

Чтение целочисленной переменной через индекс группы.
result:INTEGER := I_INDX_R (base:INTEGER, index:INTEGER);
расшифровка параметров функции I_INDX_R:
base – базовая переменная для группы,
index – индекс для чтения,
result – значение переменной, находящейся в группе под номером **index**.

Запись целочисленной переменной через индекс группы.
result:INTEGER := I_INDX_W (base:INTEGER, index:INTEGER, value: INTEGER);
расшифровка параметров функции I_INDX_W:
base – базовая переменная для группы,
index – индекс для записи,
value – значение, которое будет записано, в переменную группы под номером **index**.
result – копия записанного значения.

Чтение вещественной переменной через индекс группы.
result:REAL := R_INDX_R (base:INTEGER, index:INTEGER);
расшифровка параметров функции R_INDX_R:
base – базовая переменная для группы,
index – индекс для чтения,
result – значение переменной, находящейся в группе под номером **index**.

Запись вещественной переменной через индекс группы.
result: REAL := R_INDX_W (base:INTEGER, index:INTEGER, value:REAL);
расшифровка параметров функции R_INDX_W:
base – базовая переменная для группы,
index – индекс для записи,
value – значение, которое будет записано, в переменную группы под номером **index**.
result – копия записанного значения.

Чтение булевской переменной через индекс группы.
result:BOOLEAN := B_INDX_R (base:INTEGER, index:INTEGER);
расшифровка параметров функции B_INDX_R:
base – базовая переменная для группы,
index – индекс для чтения,

result – значение переменной, находящейся в группе под номером **index**.

Запись булевской переменной через индекс группы.
result:BOOLEAN := B_INDX_W (base:INTEGER, index:INTEGER, value: BOOLEAN);
расшифровка параметров функции B_INDX_W:
base – базовая переменная для группы,
index – индекс для записи,
value – значение, которое будет записано, в переменную группы под номером **index**.
result – копия записанного значения.

Блокировка входных переменных из кода прикладной программы.

Следующий набор функций позволяет значительно облегчить отладку сложных систем управления при помощи подмены входных воздействий на вычисляемые программой значения.

Также эти функции позволяют блокировать значения несправных датчиков с целью подмены на расчетные аварийные данные или на значения с резервных датчиков. Конечно, это можно сделать и другим способом, но использование этих функций значительно уменьшает размер кода программы, не ухудшая её читаемость.

dummy:INTEGER := BOOL_BLK(var: BOOLEAN, cmd: BOOLEAN);
Блокирует присвоение входной булевской переменной реальных входных значений, получаемых при выполнении функции OPERATE. Входной параметр **var –** переменная, подлежащая блокировке.

dummy:INTEGER := INT_BLK(var: INTEGER, cmd: BOOLEAN);
Блокирует присвоение входной целочисленной переменной реальных входных значений, получаемых при выполнении функции OPERATE. Входной параметр **var –** переменная, подлежащая блокировке.

dummy:INTEGER := REAL_BLK(var: REAL, cmd: BOOLEAN);
Блокирует присвоение входной переменной с плавающей точкой реальных входных значений, получаемых при выполнении функции OPERATE. Входной параметр **var –** переменная, подлежащая блокировке.

Переменная **cmd** указывает, какое действие необходимо совершить над данным входом:
cmd = TRUE – заблокировать обычное функционирование ввода/вывода;
cmd = FALSE – разблокировать переменную;

Так как входные переменные не допускают прямое присвоение в коде программы, в библиотеку добавлены функции, позволяющие обходить этот запрет.
dummy:INTEGER := BOOL_EQ(dest_blocked: BOOLEAN, source: BOOLEAN);
dest_blocked – заблокированная переменнная, которой будет присвоено значение **source. source** может представлять собой как константу, так и другую переменную.

dummy:INTEGER := INT_EQ(dest_blocked: INTEGER, source: INTEGER/REAL);
dest_blocked – заблокированная переменнная, которой будет присвоено значение **source. source** может представлять собой как константу, так и другую переменную.

dummy:INTEGER := REAL_EQ(dest_blocked: REAL, source: INTEGER/REAL);
dest_blocked – заблокированная переменнная, которой будет присвоено значение **source. source** может представлять собой как константу, так и другую переменную.

Пока не поддерживаются:

1. Стандартные функциональные блоки: SIG_GEN, RS, SR, R_TRIG, F_TRIG, CTU, CTD, CTUD, TON, TOF, TP, SEMA, CMP, STACKIN, DERIVAT, HYSTER, AVERAGE, LIM_ALR, BLINK, INTEGRA.

2. Работа с файлом ресурсов F_ROPEN, F_WOPEN, F_CLOSE, F_EOF, FA_READ, FA_WRITE, FM_READ, FM_WRITE – пока нет таких задач, если найдете применение – сделаем.