Быстрый старт с пакетом ISaGRAF

Виктор ЛИФЕРЕНКО
display1@mail.ru

ISaGRAF — комплекс ПО для автоматизации процессов управления, который применяется в самых различных областях техники и использует стандарт IEC 61131-3. ISaGRAF позволяет ускорить разработку и внедрение проектов, уменьшить время их выхода на рынок.

Основным компонентом технологии является среда разработки приложений (Workbench) — для проектирования, компиляции, симуляции, загрузки приложения в контроллер и отладки. Применение пакета целесообразно сначала изучить на примере демонстрационной версии (Demo), которую можно скачать с сайта компании «Фиорд» (www.fiord.com). Проект состоит из двух конфигураций, которые представляют собой две аппаратные платформы (рис. 1). Под конфигурацией понимается программный объект, который превращается в целевую функцию после его загрузки на исполнительный программируемый логический контроллер (ПЛК или PLC — обе аббревиатуры употребляются с равной частотой). Каждая конфигурация содержит по одному ресурсу, который определяет цикл работы ПЛК (рис. 1). В общем случае на одной платформе может быть расположено несколько ресурсов. Виртуальная машина выполняет программы ресурса в едином цикле. Все программы ресурса снова и снова выполняются в порядке, определенном пользователем, от первой программы до последней. Перед выполнением первой программы читаются входы. После выполнения последней программы обновляются входные данные. Цель проекта Demo состоит в генерации выходного мигающего сигнала в течение 10 с с периодом в 1 с и паузой, равной 5 с. Управление запуском и остановом генерации мигания осуществляется из удаленного ПЛК (Config 1) при помощи управляющей булевой переменной Run. В составе проекта имеются два ресурса: MASTER (ведущий) и SLAVE (ведомый), которые связаны между собой.

Два указанных ресурса расположены по одному на каждой конфигурации. Каждый ресурс содержит параметры, группы переменных, программы, функции и функциональные блоки. Взаимодействие между ПЛК организуется при помощи списка связывания. Список связывания между ресурсами можно вызвать, щелкнув по стрелке связи между ресурсами MASTER и SLAVE (рис. 2). Связь осуществляется при помощи драйвера ETCP. ETCP — это стандартный драйвер,

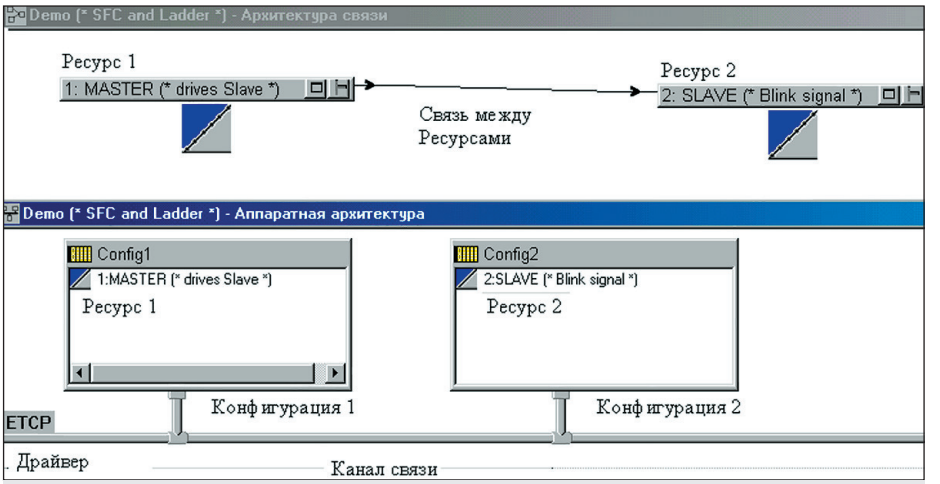

Рис. 1. Структура проекта Demo

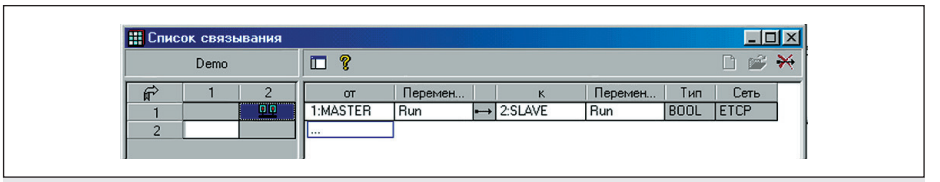

Рис. 2. Список связывания между ресурсами MASTER и SLAVE

разработанный фирмой CJ International для связи по сети Ethernet с использованием протокола TCP/IP. ETCP — это сокращение от Enhanced TCP/IP (расширенный протокол TCP/IP), поскольку в данном случае он эмулирует поведение field-bus. Таким образом, ETCP служит виртуальным field-bus.

Программа DriverSlave

В первом ресурсе для написания программы DriverSlave использован редактор SFC, окно которого представлено на рис. 3. Редактор имеет интуитивно понятный интерфейс. Предыдущий шаг передает свои функции следующим шагам. В этом состоит основное достоинство языка SFC.

В самом общем виде программа представляет собой комбинацию циклически повторяющихся шагов и переходов. Программирование при помощи языка SFC представляет собой графическое составление чередующихся элементов. В редакторе применен графический способ организации составления программ Drag and Drop.

Рассмотрим SFC-программу первого ресурса (рис. 4). В первом ресурсе объявлены две переменные типа TIME (TimeOff, TimeOn), которые определяют периоды включения и выключения исполнительной программы второго ресурса, и одна переменная типа BOOL (Run) которая управляет действиями LD-программы второго ресурса (рис. 5).

Программа содержит две пары шаг/переход: S1-T1 и S2-T2. Шаг всегда чередуется с переходом. На диаграмме активен начальный шаг, который помечен маркером. Переходы T1 и T2 обозначены скрещенными под прямым углом отрезками прямых линий.

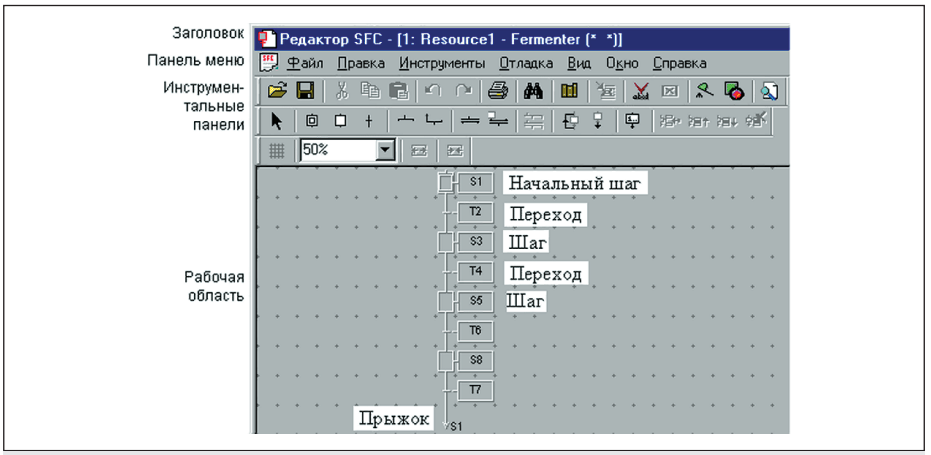

Рис. 3. Общий вид редактора SFC

Рис. 4. Ресурс 1 и программа SFC

Шаг S1 не содержит никакого действия (блок действий без имени или кода игнорируется).

Переход T1 ожидает, пока время активности шага S1 не превысит значение переменной TimeOff (начальное значение переменной TimeOff — 5 с).

Шаг S2 состоит из одного булевского несохраняемого действия. Переменная Run устанавливается в состояние TRUE, когда шаг S2 становится активным, и сбрасывается в состояние FALSE, когда шаг S2 дезактивизируется. Переход T2 ожидает, пока время активности шага S2 не превысит значение переменной TimeOn (значение переменной TimeOn — 10 с). Описания переменных в первом ресурсе вызываются из словаря, для чего нужно воспользоваться кнопкой и выбрать из разворачивающегося меню название программы. В нашем случае это будет иконка программы DriverSlave (* pilote the Blink *) (рис. 5). Временными переменными S1.T и S2.T обозначаются продолжительность шага S1 и S2 соответственно. Время выполнения цикла программы составляет 15 с.

TimeOff		TIME		t#5s
Run		BOOL		
TimeOn		TIME		t#10s

Рис. 5. Группа переменных, объявленная в первом ресурсе

Программа ресурса Slave

Программа BlinkOut составлена на языке LD с помощью редактора DGE. Окно редактора и булево уравнение с комментариями представлено на рис. 6.

Кнопки на панели инструментов реализуют следующие процедуры (рис. 6):

- F2 — добавление контакта слева;
- F3 — добавление контакта справа;
- F4 — добавление контакта параллельно;
- F5 — добавление витка;
- F6 — добавление блока слева;
- F7 — добавление блока справа;
- F8 — добавление блока параллельно;
- F9 — добавление прыжка.

Ресурс 2 и программа LD BlinkOut без деталей представлены на рис. 7. В программе объявлены три переменные:

- Run — тип BOOL;
- Time — тип TIME;
- OutV — тип BOOL (рис. 8).

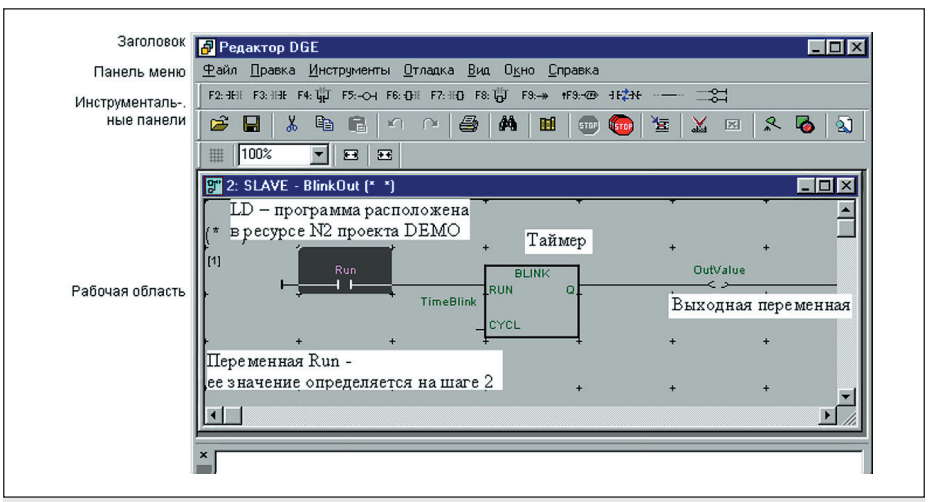

Рис. 6. Вид редактора DGE и программа BlinkOut ресурса Slave

Имя	Алиас	Тип	[]	Нач. зн...
Run		BOOL		
Time...		TIME		T#1S
OutV...		BOOL		

Рис. 8. Группа переменных, объявленная во втором ресурсе

Логическое уравнение представлено в виде последовательной цепи из трех элементов: контакта Run, блока BLINK и витка OutV.

Программа работает следующим образом: из конфигурации 1 по линии связи поступает значение переменной Run.

Блок BLINK находится в библиотеке (рис. 9).

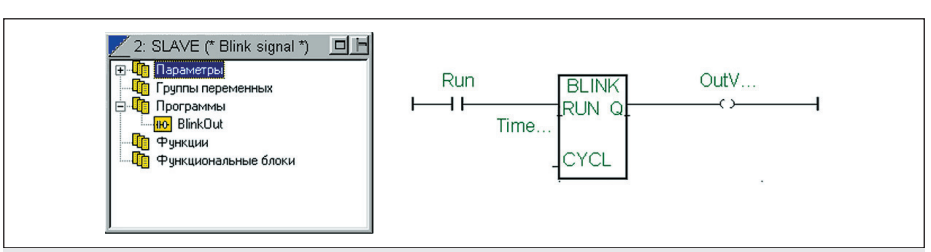

Рис. 7. Ресурс 2 и программа LD

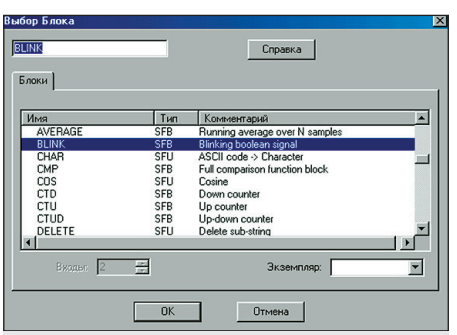

Рис. 9. Окно библиотеки блоков

Переменная Run устанавливается в состояние TRUE, когда шаг S2 становится активным, и сбрасывается в состояние FALSE, когда шаг S2 дезактивизируется. В момент активности шага S2 срабатывает блок BLINK, который генерирует сигнал длительностью 1 с для выходного витка OutV. Эпюра сигнала на входе и выходе блока BLINK представлена на рис. 10. Блок управляет выходной переменной OutValue, которая смонтирована к единственному каналу устройства (рис. 11).

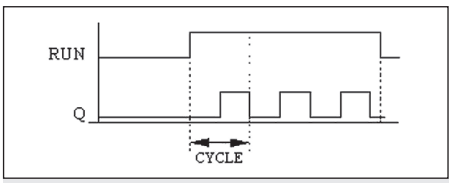

Рис. 10. Эпюра сигнала на входе и выходе блока BLINK

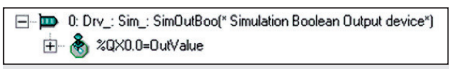

Рис. 11. Монтаж переменной OutValue

Генерация кода проекта осуществляется с помощью кнопки, расположенной на стандартной панели инструментов.

Процесс моделирования запускается щелчком на кнопке.

Работа с проектом Demo

Среда разработки проекта открывается щелчком на значке ISaGRAF PRO.

Проект Demo открывается с помощью кнопки «Открыть». В диалоговом окне «Открытие файла» необходимо выбрать каталог Prj, затем каталог Demo (рис. 12).

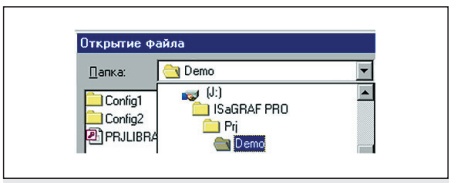

Рис. 12. Открытие файла Demo

После старта с помощью кнопки будут запущены отладчик, менеджер конфигурации, ядро и панель ввода/вывода симулятора. Виртуальные машины начинают выполнение ресурсов. Иконка каждого ресурса окрашивается в зеленый цвет, и в панели заголовка ресурса появляется текст RUN RUN - CODE 1: MASTER (* drives Slave *).

Для прекращения моделирования необходимо войти в среду разработки ISaGRA и нажать кнопку «Остановить».

С целью закрепления изученного материала предлагаем произвести следующие действия:

- для просмотра текущего активного шага открыть SFC-программу DriveSlave;
- для проверки значений переменных Run, TimeBlink и OutValue открыть LD-программу BlinkOut»;
- для проверки значений переменных и ввода новых значений открыть словарь (щелчок в ячейке, где отображается значение);
- для просмотра значения переменной Out Value открыть инструмент монтажа ввода/вывода.
- создать собственный проект.

Создание проекта

После основательного изучения демо-проекта можно приступить к составлению собственного проекта. Для начала воспользуемся одним языком из пяти, чтобы упростить и ограничить задачу. Выберем язык лестничных диаграмм LD.Наш проект будет состоять из двух PLC — ведущего и ведомого. Ведомый PLC генерирует мигающий сигнал по команде ведущего. Отличие от предыдущего проекта заключается в том, что ведущий PLC генерирует постоянный во времени управляющий сигнал. Управляющий сигнал формируется простым уравнением RUN=A AND B.

Создадим новый проект NProject из двух ресурсов, которые расположим на двух конфигурациях, воспользовавшись последовательно двумя кнопками на панели инструментов и (рис. 13).

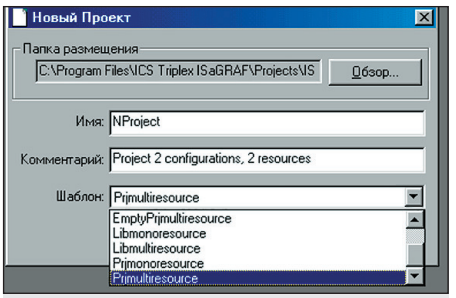

Рис. 13. Окно для создания нового проекта

После нажатия кнопки OK появится окно менеджера проекта «Архитектура связи», состоящее из двух ресурсов (рис. 14).

Создадим описание проекта: выберем ресурс 1, затем из меню «Инструменты» вызовем «Редактор описания» и введем текст описания (рис. 15).

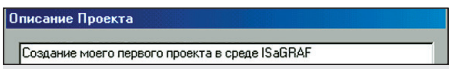

Рис. 15. Окно «Описание проекта»

Воспользуемся каскадом разворачивающихся меню и создадим LD-программу в первом ресурсе (рис. 16).

Введем имя программы — LD1 (рис 17).

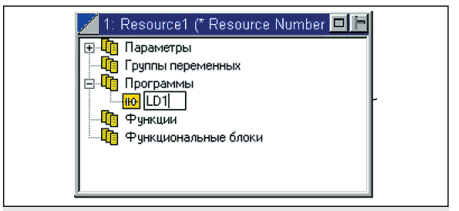

Рис. 17. Введение имени программы

После создания пустой программы появится сообщение об изменении архитектуры проекта — здесь нужно нажать кнопку Да.

В первом ресурсе составим простую программу Run=A FND B. Переменные назначаются из разворачивающегося меню «Выберите переменную». Выбираем переменные A, B, Run.

Окончательно схема И будет в первом ресурсе выглядеть следующим образом (рис. 18). Переменные A и B назначены в списке, изображенном на рис. 19.

Для проверки правильности компиляции нажимаем кнопку на панели инструментов. Если программа прошла проверку, то появится сообщение, показанное на рис. 20.

Аналогично выбираем имя программы второго ресурса — LD2.

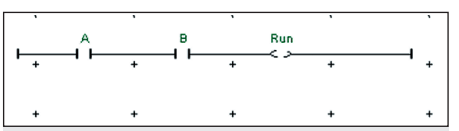

Рис. 18. Схема И в первом ресурсе

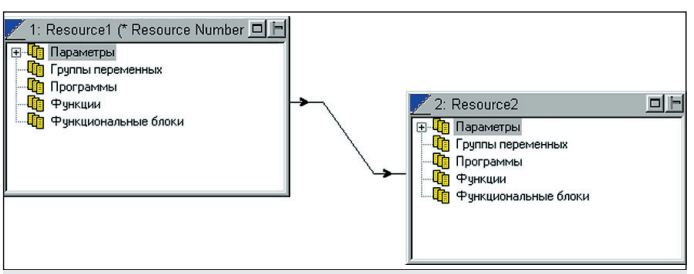

Рис. 14. Окно «Архитектура связи»

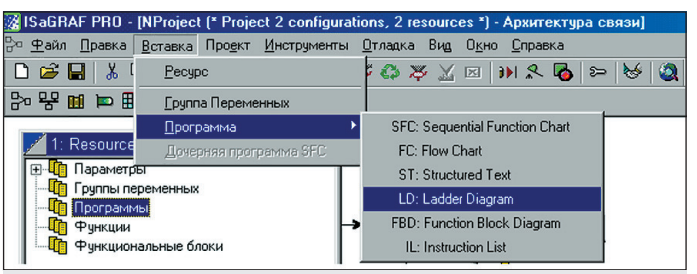

Рис. 16. Создание программы LD: Ladder Diagram

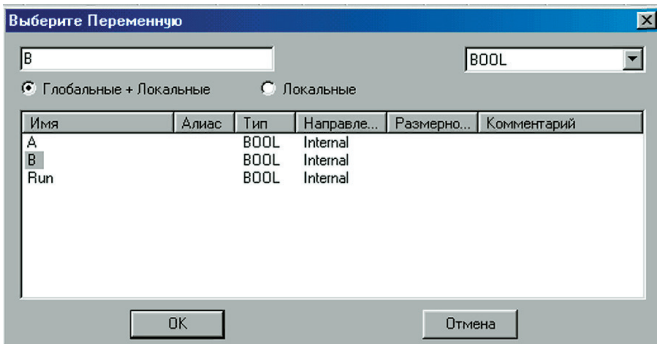

Рис. 19. Переменные первого ресурса

Если все было выполнено правильно, то появится сообщение, приведенное на рис. 25.

Теперь нажмите кнопку старт и кнопку TRUE для переменных A и B в программе LD1. Открыв словарь проекта, можно наблюдать значения переменных во времени, включать и выключать ведомый PLC, а также создавать другие схемы управления.

Итак, мы дали начальное представление об основах технологии открытой автоматизации ISaGRAF. Много полезной информации и примеров реализации пакета, а также консультации по ISaGRAF и ее составляющим можно получить на сайте компании «Фиорд» (www.fiord.com).

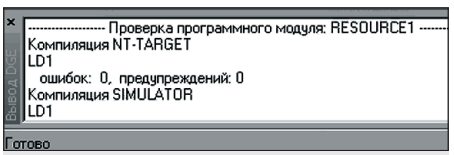

Рис. 20. Проверка программного модуля

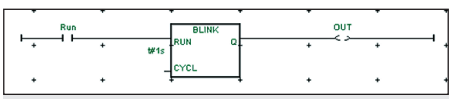

Рис. 21. Программа второго ресурса

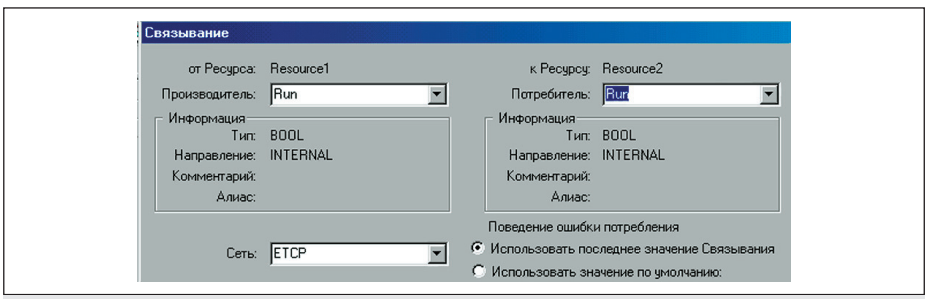

Рис. 22. Связывание ресурсов

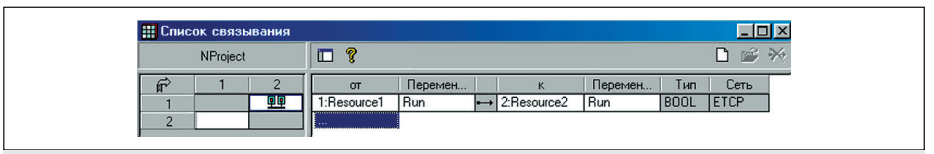

Рис. 23. Список связывания ресурсов

Пользуясь теми же процедурами, что и при составлении программы для ресурса 1, составляем программу для ресурса 2, полностью идентичную программе второго ресурса из проекта Demo. Вызываем переменную Run и затем присоединяем блок справа.

Программа второго ресурса будет выглядеть, как показано на рис. 21.

Организуем связь между ресурсами, щелкнув по стрелке связи между ними. Появится панель связывания, в которой имеются списки переменных производителя (ресурс 1) и потребителя (ресурс 2), из которых выбираем переменную Run (рис. 22).

В списке связывания появится сообщение о связи ресурса 1 и ресурса 2 по сети ETCP с передачей переменной Run (рис. 23).

Назначаем значение TRUE для переменных A и B в первом ресурсе, для чего щелчком на переменной вызываем диалоговое окно «Запись BOOL» (рис. 24).

Если процедура связывания ресурсов была выполнена правильно, то можно приступить к построению проекта, нажав кнопку .

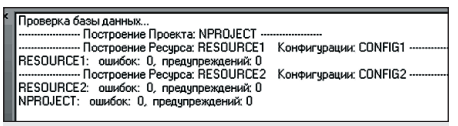

Рис. 25. Сообщение об ошибках

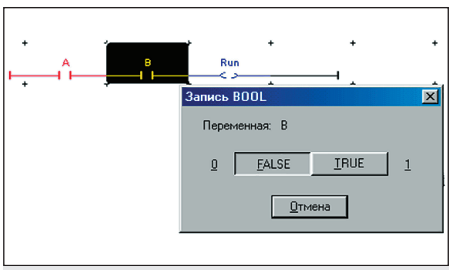

Рис 24. Запись значений переменных в работающем проекте