ULTRALOGIC – СИСТЕМА ПОДГОТОВКИ ПРОГРАММ ДЛЯ ПРОМЫШЛЕННЫХ КОНТРОЛЛЕРОВ

Сергей Шакиров, Ренат Биюсов, Борис Якубович, Валерий Журавлев

Рассматривается система программирования промышленных контроллеров, базирующаяся на рекомендациях стандарта МЭК IEC-1131.

ВВЕДЕНИЕ

Технологический бум, вызванный массовым нашествием на просторы СНГ так называемых свободно программируемых промышленных контроллеров, потребовал услуг высококвалифицированных профессионалов, способных решить вопросы программирования и внедрения этих контроллеров. Кроме хорошего программиста, нужны были также электронщик и технолог, досконально знающий автоматизируемый процесс. Затраты времени и средств на эту работу, как правило, находились в прямой зависимости от способностей и амбиций программиста, входящего в состав группы автоматизации. Зачастую при таком подходе программист оставался единственным человеком, способным разобраться в своем творении, со всеми вытекающими отсюда последствиями. Абсурдность данной ситуации породила стремление создать некие технологические языки программирования, доступные киповцам и снимающие завесу таинства с процесса программирования. В результате технологических языков было создано столько, сколько коллективов занималось этой проблемой. Наши умельцы от программирования и по сей день с энтузиазмом наступают на одни и те же грабли, разрабатывая собственные языковые средства, типа «есл пр то дл», в то время как уже несколько лет существует стандарт Международной Электротехнической Комиссии IEC-1131. Стандарт IEC-1131 явился квинтэссенцией опыта «братских» капиталистических стран в области языков программирования для систем автоматизации технологических процессов.

Стандарт специфицирует 5 языков программирования.

Sequential Function Chart (SFC) – язык последовательных функциональных блоков.

Function Block Diagram (FBD) – язык функциональных блоковых диаграмм.

Ladder Diagrams (LD) – язык релейных диаграмм.

Structured Text (ST) – язык структурированного текста.

Instruction List (IL) – язык инструкций.

Языки **ST** и **IL** являются неким программистским «эсперанто», поскольку они вобрали в себя наиболее общие операторы языков типа **Pascal** и ассемблер и обеспечивают совместимость стандарта с ранними версиями программного обеспечения производителей контроллеров. Язык **LD** отдает дань поклонникам стиля ALLEN-BRADLEY, когда программы похожи на электросхемы релейной логики. Язык **SFC** позволяет осуществлять программирование на алгоритмическом уровне, но предполагает конечную реализацию программы на других языках.

Язык функциональных блоковых диаграмм **FBD** вышел из рамок ограничений языка релейных схем и по существу решил все проблемы, связанные с использованием лингвистических языков в управлении техпроцессами. Этот язык служит для построения и детального описания алгоритмов управления технологическим процессом. Он предоставляет пользователю возможность естественным для инженера-киповца образом построить любую сложную процедуру, состоящую из библиотечных

блоков (те, кто знаком с пакетом **P-CAD**, сразу поймут, в чем дело). В качестве библиотечных блоков используются как элементарные функции, так и алгоритмы **П**, **ПИ**, **ПИД**-регулирования, фильтрация сигналов, стабилизация заданных параметров. «Джентльменский набор» из математических и статистических функциональных блоков позволяет просто организовать необходимые вычисления и обработку сигналов.

В рамках данной статьи мы хотим представить читателям систему **ULTRALOGIC**, которая предназначена для разработки программ промышленных контроллеров с помощью простых инструментальных средств, используя в качестве языка программирования язык функциональных блоковых диаграмм.

«Простота» в данном случае достигается применением методов объектного визуального программирования, когда пользователь собирает программу, как домик из кубиков детского конструктора. При этом исключаются «ошибки пальца», широко распространенные в лингвистических языках программирования, опасные операторы циклов, проблемы с захватом памяти и т. п. Система максимально ориентирована на то, чтобы инженер-специалист в области автоматизации работал с понятным ему технологическим контроллером, а не компьютером с его мудреной системой команд, памятью и, страшно подумать – прерываниями. **ULTRALOGIC** рассматривает контроллер как «черный ящик», связанный с объектом управления посредством формальных устройств аналогового и дискретного ввода/вывода. Подобный подход вовсе не предполагает тривиальности системы. Для людей любознательных и склонных к самовыражению **ULTRALOGIC** имеет механизм вызова внешних процедур, написанных на других языках, таких как ассемблер, C, Pascal. Ориентированная на IBM PC совместимые контроллеры, система **ULTRALOGIC** фактически является независимой по отношению к аппаратной платформе целевого контроллера. Для этого в системе есть специальный инвариантный компилятор, который использует подготовленную ранее информацию об аппаратной платформе контроллера и его конфигурации. Информация готовится пользователем в диалоговом режиме. Система предоставляет пользователю возможность создания и накопления собственных функциональных блоков, что в сочетании с поддержкой иерархического проектирования служит источником неиссякаемого вдохновения аппаратчика для создания и накопления новых «кубиков», охватывающих широкий спектр возможных приложений. Этот механизм является мощным инструментом для облегчения разработки и улучшения читабельности программы, когда один или несколько функциональных блоков полностью описывают управление тем или иным технологическим процессом.

ULTRALOGIC представляет собой интегрированный комплекс программ в операционной среде DOS или Windows и включает в себя графические средства, компиляторы, средства интерактивного диалога, настройки и отладки проектов.

ULTRALOGIC функционирует на IBM PC совместимом компьютере, с помощью которого может производиться и отладка программы на объекте. Использование Notebook вместо пульта качественно изменило процесс отладки. Оно позволяет осуществлять оперативный мониторинг процесса, осциллографирование любых переменных в реальном времени, простой подбор параметров регулирования, быстрое исправление и мгновенную перекомпиляцию проекта, удаленную отладку, доступ к любой справочной информации об объекте. Notebook на коленях – целый мир по сравнению с технологическим пультом, с его кнопочками и светодиодными индикаторами.

АРХИТЕКТУРА СИСТЕМЫ

ULTRALOGIC состоит из двух частей: системы программирования и системы исполнения (рис. 1).

Система программирования содержит собственно средства подготовки проектов, менеджер проектов и средства их отладки.

Менеджер проектов объединяет в себе
- редактор переменных;
- конфигуратор контроллера;
- менеджер программ;
- компиляторы.

Система отладки содержит загрузчик программ, сетевой драйвер, средства осциллографирования и удаленной отладки.

Система исполнения функционирует на целевом контроллере, который может базироваться как на платформе INTEL, так и на другой аппаратной платформе. Любой экзотический контроллер может быть подключен к системе **ULTRALOGIC**, если он имеет систему команд и средства загрузки программы (естественно, после того как соответствующий компилятор будет интегрирован в систему программирования). В одних случаях программа в контроллер может загружаться, например, по каналу последовательной связи, в других с

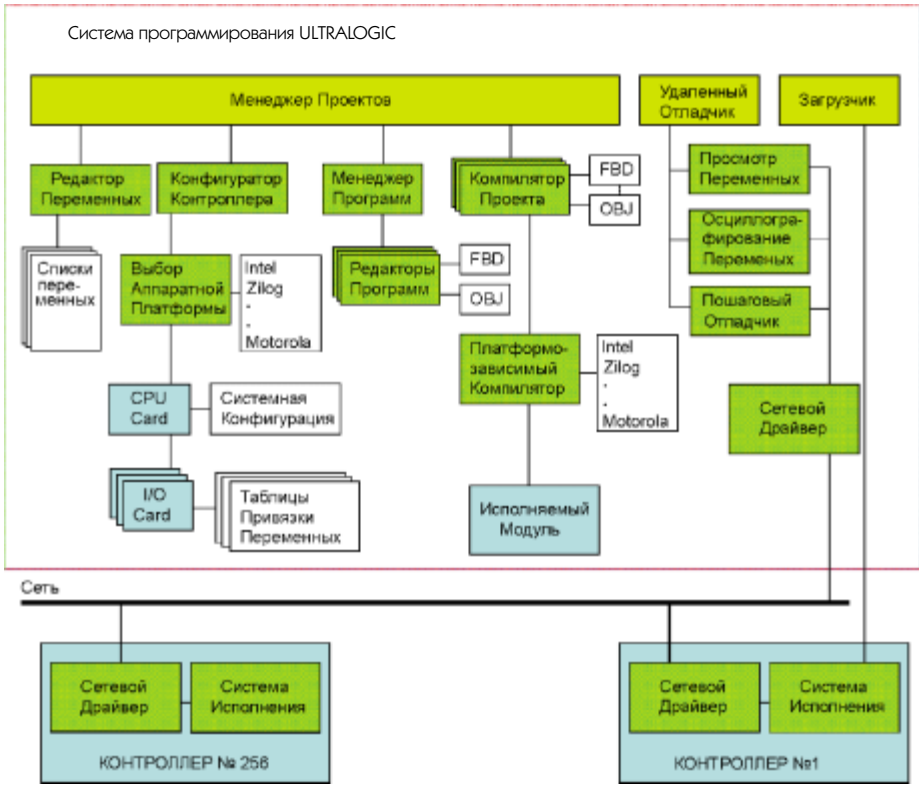


Рис. 1 Архитектура системы ULTRALOGIC

помощью внешне программируемой микросхемы памяти.

ULTRALOGIC использует метод сетевого взаимодействия между контроллерами (системами исполнения), управляющими технологическим процессом, и системой визуализации данных верхнего уровня (SCADA/MMI). Сетевые ресурсы автоматически включаются в порождаемый код системы исполнения. Количество участников сети может достигать 256.

Временная диаграмма цикла работы системы исполнения (контроллера) представлена на рис. 2.

Рис. 2. Циклограмма функционирования системы исполнения на целевом контроллере

Сеанс сетевого обмена с верхним уровнем показан в цикле контроллера условно. На самом деле сетевой обмен носит случайный характер и всегда инициируется системой верхнего уровня. В систему встроен драйвер, поддерживающий обмен по RS-485. В качестве опции поставляются драйверы для сетевого обмена, использующие протокол IPX/SPX.

БАЗОВЫЕ КОНЦЕПЦИИ ULTRALOGIC

Программа управления в **ULTRALOGIC** представляется как некоторое логическое программное устройство, описывающее технологический процесс и операции над переменными величинами (параметрами) этого процесса. Программа может содержать следующие базовые типы объектов:

- переменные;
- константы;
- комментарии;
- функциональные блоки.

Поддерживаемые типы переменных приведены в табл. 1.

Типы констант соответствуют основным типам переменных.

Любой переменной могут быть присвоены следующие атрибуты:

Таблица 1

Поддерживаемые типы переменных в системе ULTRALOGIC

Тип переменной	Принимаемые значения
Двоичные переменные (**Boolean**)	**TRUE** – истина, **FALSE** – ложь
Переменные целого типа (**Integer**)	**-32768**... **+32767**
Переменные с плавающей точкой (**Float**)	**±1.18E-38**... **±3.4E+38** (**IEEE 754**)
Таймерные переменные (**Timer**)	не более **8760h59m59s99**
Здесь **h** – часы, **m** – минуты, **s** – секунды, **99** – сотые доли секунды.	

Public – глобальная переменная, может использоваться всеми программами проекта;

Network – переменная доступна другим участникам сетевого обмена.

По отношению ко входам и выходам контроллера переменные могут иметь признак:

Input – входная переменная, логически соединенная со входом контроллера, или

Output – выходная переменная, логически соединенная с выходом (выходами) контроллера.

Таблица 2

Примеры базовых функций языка FBD

Функции двоичного типа	**NOT**, **AND**, **OR**, **XOR**, **SET**, **RESET**
Функции управления программой	**RETURN**, **GOTO**, **CALL**, **TSTART**, **TSTOP**, **GSTART**, **GSTOP**
Арифметические функции	**ADD**, **SUB**, **DIV**, **MUL**
Функции сравнения	**=**, **<>**, **>**, **<**, **=>**, **<=**
Математические функции	**ABS**, **EXPT**, **LOG**, **SQRT**
Тригонометрические функции	**ACOS**, **ASIN**, **ATAN**, **COS**, **SIN**, **TAN**

Язык функциональных блоковых диаграмм (FBD) описывает функции между входными переменными и выходными переменными (рис. 3). Эти функции описываются в виде сочетания элементарных функциональных блоков. Выход функционального блока может быть соединен с другими блоками. Каждый функциональный блок представляет из себя прямоугольник, внутри которого имеется обозначение функции, выполняемой блоком.

Один или несколько функциональных блоков, соединенных между собой, и образуют программу на языке FBD.

Имеются следующие формальные правила языка FBD:

- функциональные блоки могут располагаться произвольно в поле программы;

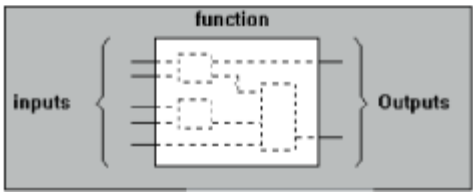


Рис. 3. Язык FBD описывает функции между входными и выходными переменными

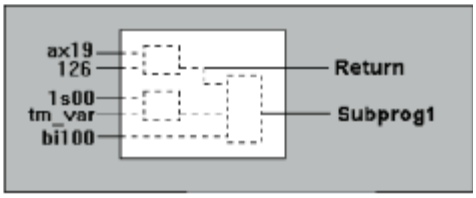


Рис. 4. Входные и выходные переменные языка FBD

- не может быть свободных (несоединенных) входов и выходов функционального блока;
- любая связь (NET) может иметь имя переменной;
- входы и выходы функциональных блоков, присоединенные к связям, имеющим одинаковые имена, считаются соединенными;
- очередность выполнения блоков в программе: слева направо, сверху вниз.

Переменные FBD-программ присоединяются к входным/выходным точкам функциональных блоков (рис. 4.).

На входе FBD-блока может быть

- константное выражение;
- любая внутренняя или входная переменная;
- выходная переменная.

На выходе FBD-блока может быть любая внутренняя или выходная переменная.

В табл. 2 приведены примеры базовых функций языка FBD.

Программы на языке FBD напоминают электрические принципиальные схемы логических устройств и формально соблюдают алгоритмы их работы.

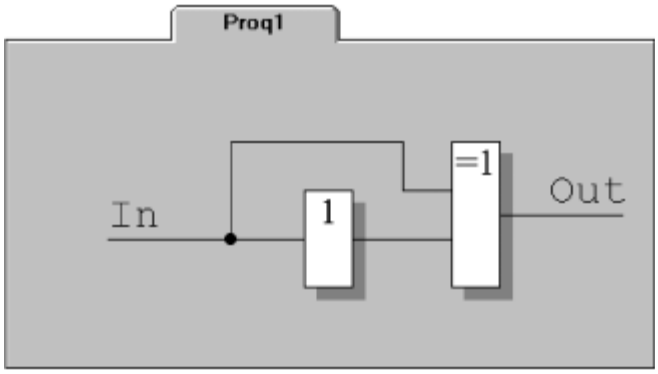
Однако, несмотря на всю схожесть с электрическими схемами, язык **FBD** содержит метки, операторы условного и безусловного переходов, которые свойственны традиционным процедурным языкам программирования. Примеры 1-4 иллюстрируют реализацию некоторых простых функций с помощью языка FBD.

МЕНЕДЖЕР ПРОЕКТОВ

Окно, где создаются и редактируются проекты, называется окном менеджера проектов. Оно представляет собой набор секций с закладками в нижней и верхней части. Каждая секция предна-

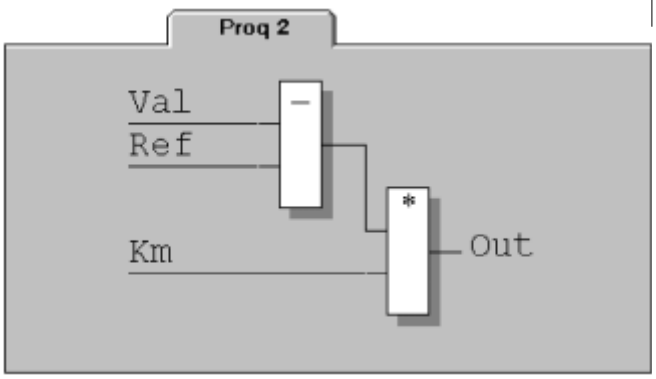
Пример 1. Обнаружение перепада сигнала IN

Двоичная переменная **OUT** примет значение **TRUE** только тогда, когда переменная **IN** изменит свое состояние.



Пример 2. Пропорциональный регулятор

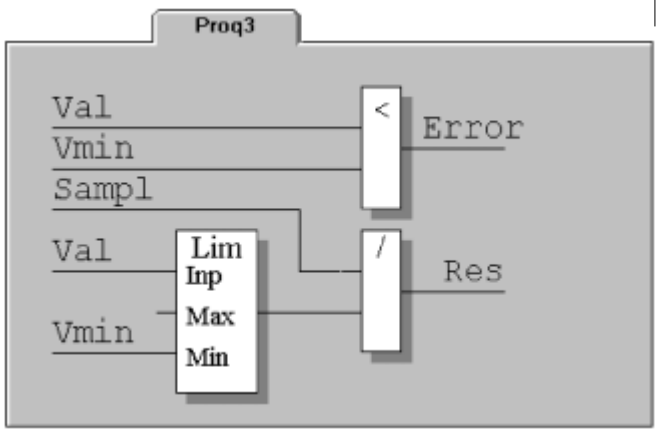
Val – регулируемый параметр;
Ref – уставка, заданное значение параметра;
Km – коэффициент пропорциональности;
Out – сигнал регулирования.



Разность (величина рассогласования) между измеренным значением регулируемого параметра **Val** и его заданным значением **Ref** умножается на Km и используется в качестве регулирующего воздействия.

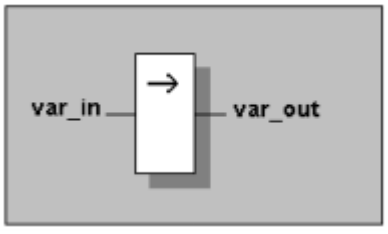
Пример 3. Ограничение нижнего предела делителя

Программа обеспечивает деление значения переменной **Sampl** на значение переменной **Val**. Переменная делителя **Val** ограничена снизу значением **Vmin**. Переменная **Error** принимает значение **TRUE**, если **Val** < **Vmin**.



Пример 4. Оператор преобразования типов

Приводит тип переменной **var_in** к типу переменной **var_out**.
Функция преобразования выбирается автоматически, исходя из назначенных на входе и выходе блока типов переменных.



Пример преобразования var_in в двоичный тип

Тип **var_in**	Тип и значение **var_out**
Двоичный (**B**)	**var_in**
Целый (**I**)	**FALSE** для **var_in** = **0**, **TRUE** в других случаях
Плавающий (**F**)	**FALSE** для **var_in** = **0.0**, **TRUE** в других случаях
Таймерный (**T**)	**FALSE** для **var_in** = **0S**, **TRUE** в других случаях

значена для определенных типов данных, составляющих проект. Названия секций приведены на нижних закладках, названия разделов в секции – на верхних закладках. В менеджер проектов входят следующие секции:

Program – программы проекта как на языке FBD, так и на других языках;

Variables – списки и атрибуты всех переменных и констант проекта;

Config – описание конфигурации системной части контроллера, модулей контроллера, таблицы привязки переменных проекта к выходам и входам модулей, калибровочные таблицы измерительных каналов;

Options – текстовая неформальная информация о проекте.

Секция программирования

Секция программирования (Program) содержит титульный лист и листы программ (рис. 5). На титульном листе приведен список программ, входящих в проект, с указанием языка программирования и комментария. Каждая программа имеет свой лист, над которым располагается закладка с именем программы.

Автоматизируемый технологический процесс разбивается на отдельные формальные задачи, выполняемые последовательно. Порядок выполнения задач может быть изменен произвольным образом. Программа может состоять из множества программ, находящихся друг с другом в определенных отношениях, образующих иерархическое дерево. Программы, состояние которых объявлено как **Start** (Auto Run), активизируются при запуске системы и выполняются в каждом цикле работы контроллера. Программы, состояние которых объявлено как **Stop**, активизируются при выполнении тех или иных условий.

Собственно программирование осуществляется с помощью специального графического редактора (рис. 6).

При этом пользователь с помощью мыши устанавливает функциональные блоки в поле программы, соединяет их связями, присваивает связям имена переменных. Имена могут непосредственно назначаться в поле программы или вызываться из списков в секции **Variables**. Все шаги по составлению программы записываются в файл-сценарий, благодаря чему можно производить откат к предыдущему состоянию (Undo) и возврат к правкам (Redo). Число шагов Undo/Redo задается пользователем в пределах от 0 до 16000. Редактор позволяет выполнять групповые опера-

ции, операции в окне. Любой вход и выход двоичных блоков может быть изменен на инверсный простым нажатием кнопки мыши. Количество входов функциональных блоков может задаваться при вызове блока и варьируется от 2 до 32. Данная операция применима только к тем блокам, для которых она допустима, например для двоичной функции AND или операции сложения ADD. Редактор может включать разметку поля, масштабировать изображение, автоматически изменяет изображение указателя мыши в зависимости от типа операции. Все манипуляции с объектами осуществляются с помощью мыши. Назначение кнопок и порядок работы с объектами совпадают с общепринятыми в Windows соглашениями.

Разработка собственных функциональных блоков также происходит с помощью графического редактора. Алгоритм разработки пользовательского функционального блока (User Functional Block, UFB) следующий.

1. Используя базовые функциональные блоки, составляют программу, реализующую функции создаваемого UFB.

2. Внешним связям UFB присваиваются имена, которые будут использоваться в графическом изображении блока как функция того или иного вывода.

3. Программа сворачивается в прямоугольную картинку с помощью команды **Picture**.

4. Специальными командами формируется изображение UFB:

Wide – задать ширину UFB;

InpPins – задать число входов UFB;

OutPins – задать число выходов UFB;

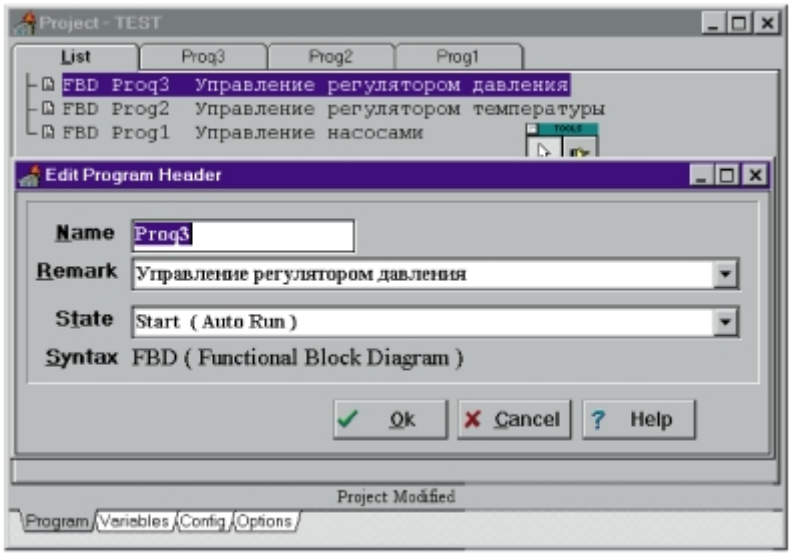


Рис. 5. Вид экрана при работе в секции программирования

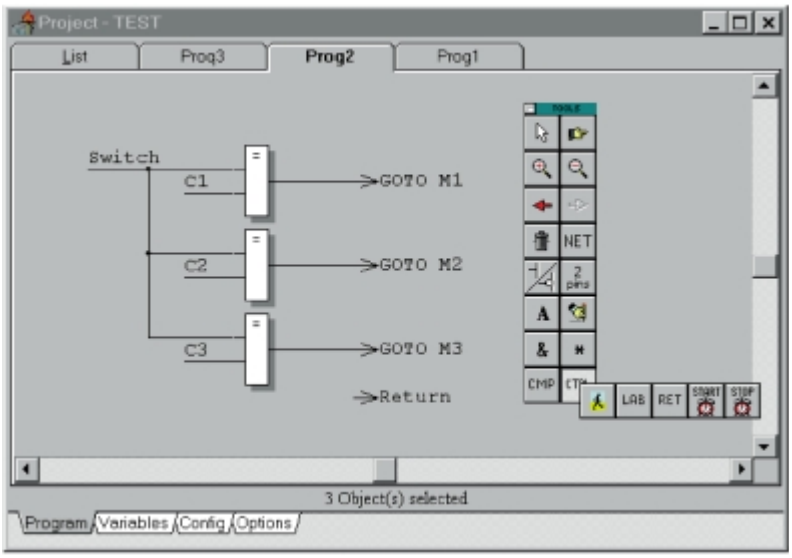


Рис. 6. Для программирования на языке FBD используется специальный графический редактор

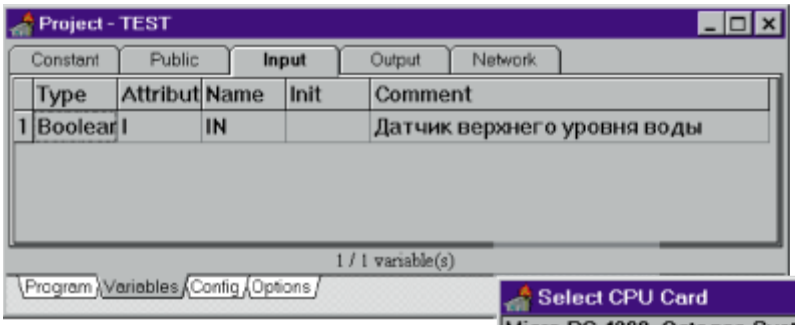


Рис. 7. Переменные проекта для удобства работы разбиты на 5 групп

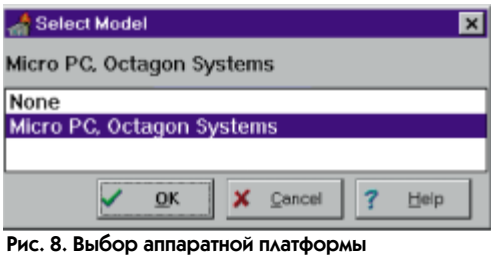


Рис. 8. Выбор аппаратной платформы контроллера в процессе конфигурирования

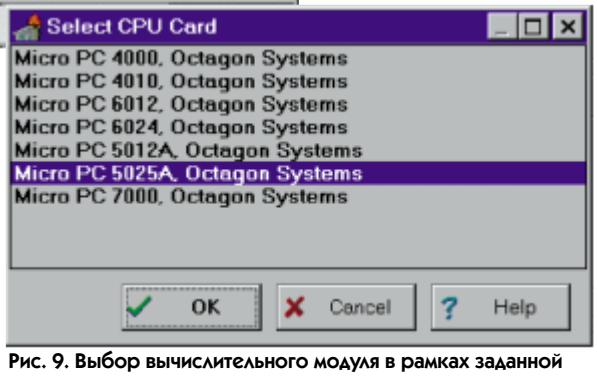


Рис. 9. Выбор вычислительного модуля в рамках заданной аппаратной платформы

NameElem – ввод заголовка UFB;

PinName – ввод названия входов и выходов UFB;

PinType – выбор типов входов и выходов UFB;

VarType – выбор типов переменных, подключаемых ко входам и выходам UFB.

Полученные элементы можно использовать для создания других библиотечных элементов и т. д., причем число вложений не ограничено.

Секция переменных

В секции **Variables** осуществляется ввод глобальных переменных проекта, сформированных в списки по тем или иным признакам (рис. 7). Разбиение переменных на списки носит формальный характер и служит для удобства просмотра, поиска и редактирования. Переменные сформированы в 5 списков: Constant, Public, Input, Output, Network. Переменные и константы в этих списках доступны всем программам проекта, а также конфигуратору контроллера. Переменные, введенные на этапе конфигурирования, назначенные как входы и выходы контроллера, заносятся в эти списки автоматически. Переменные, имена которых введены как имена связей функциональных блоков в тело програм-

мы, являются локальными и в списки не заносятся.

Секция конфигурирования

В секции конфигурирования (**Config**) последовательно осуществляются следующие действия:

- выбирается аппаратная платформа контроллера (рис. 8);
- выбирается тип вычислительного модуля внутри платформы (рис. 9);
- указываются системные установки, такие как наличие сети, сторожевого таймера, настройки компилятора (рис. 10);
- выбираются типы используемых модулей ввода/вывода контроллера (рис. 11);
- осуществляется привязка переменных ко входам и выходам соответствующих модулей.

В результате этих действий в секции **Config** на листе **Model** сформируется описание вычислителя контроллера, на листе **Modules** – типы модулей, входящих в контроллер, а на каждом последующем листе – описание привязки конкретного модуля к входным и выходным переменным (рис. 12). Хотя по умолчанию **ULTRALOGIC** предлагает переменным системные имена, которые указывают на конкретный тип переменной и ее адрес подключения в контроллере, пользователь может называть переменные произвольным образом.

Все действия происходят в режиме интерактивного диалога путем выбора соответствующих опций из последовательно возникающих окон.

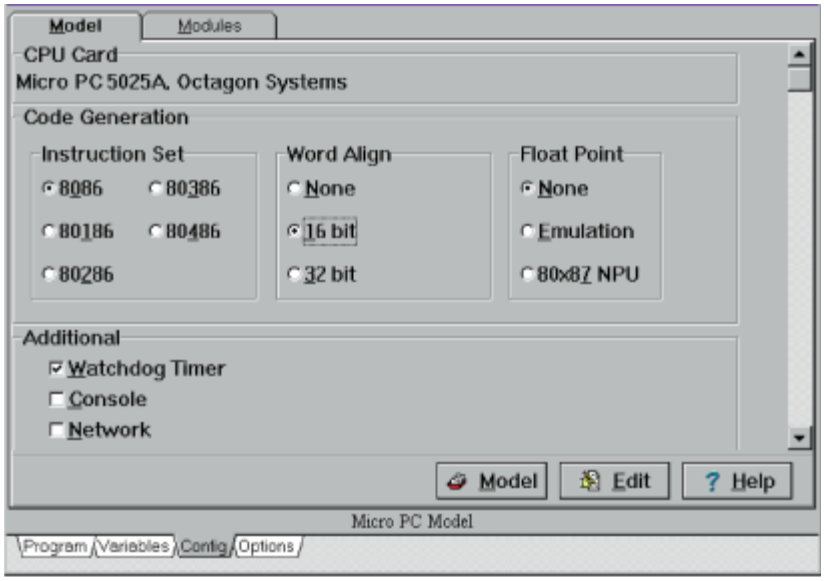


Рис. 10. Во время конфигурирования можно задать различные системные установки

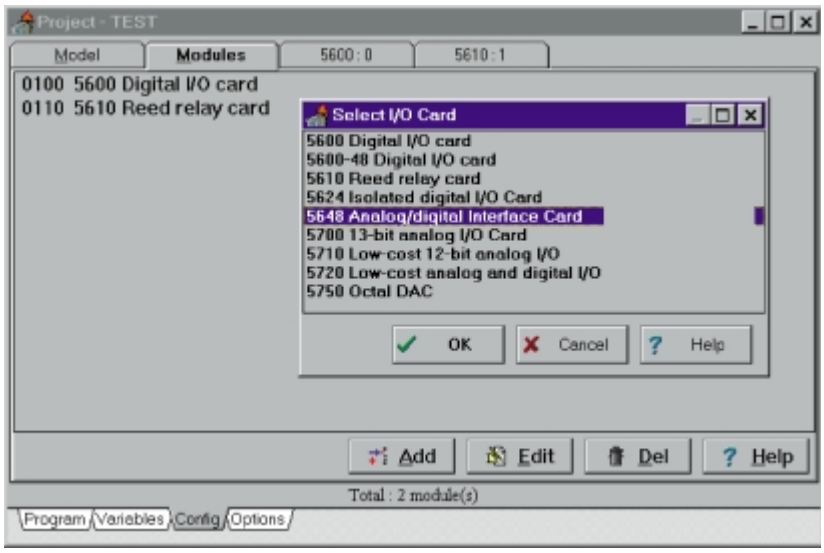


Рис. 11. Выбор используемых модулей ввода/вывода на этапе конфигурирования

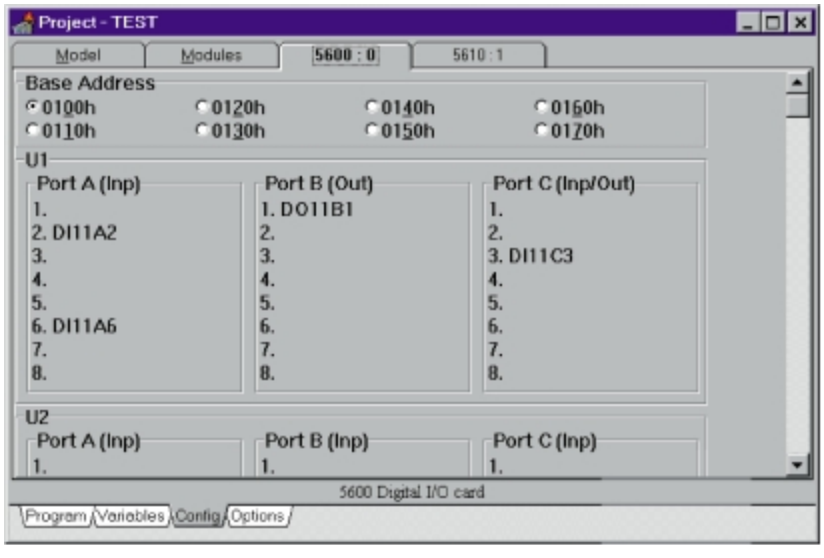


Рис. 12. Назначение сигналов конкретным каналам ввода/вывода во время конфигурации контроллера

ЗАГРУЗКА И ОТЛАДКА ПРОГРАММ

Готовый проект компилируется, после чего полученный код системы исполнения загружается в контроллер.

В этот код автоматически встраивается драйвер сетевого обмена, который обеспечивает мониторинг и отладку. Каждая переменная, участвующая в сетевом обмене, имеет двойное имя, состоящее из имени переменной и префикса, являющегося сетевым адресом контроллера. Например, переменная с именем 03.Var1 принадлежит контроллеру с сетевым адресом 03. Общее количество участников сети может быть 256, и для системы визуализации они представляются как переменные единого технологического процесса, безотносительно к территориальному расположению контроллеров. Инициатором обмена всегда является отладчик или диспетчерская система верхнего уровня. Она запрашивает у контроллеров переменные для визуализации и передает им список новых значений переменных, являющихся уставками и режимами работы. Удаленный отладчик имеет режимы пошагового исполнения программы, позволяет задавать и удалять точки останова, обеспечивать визуализацию переменных в точках останова.

Принимаемые отладчиком переменные могут быть направлены на осциллографирование. Число одновременно осциллографируемых переменных не ограничено. Графики масштабируются по амплитуде и временной шкале, мо-

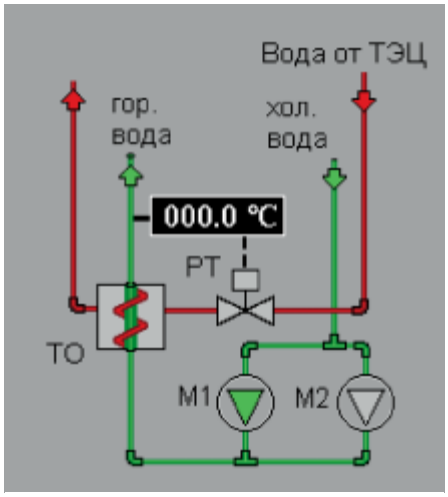


Рис. 13. Пример автоматизированной системы подогрева воды

гут быть сохранены на диске для последующего анализа. Минимальный квант времени между отсчетами одной переменой – 10 мс, максимальное количество точек осциллограммы –16000.

ПРАКТИЧЕСКИЙ ПРИМЕР

Рассмотрим пример проекта автоматизированной системы подогрева воды (рис. 13).

Холодная вода нагревается в теплообменнике ТО. Давление холодной воды обеспечивает двигатель M1; двигатель M2 является резервным. Количество горячей воды от ТЭЦ в теплообменнике регулируется аналоговым регулятором PT с приводом постоянного тока. Температура нагреваемой воды зависит от положения регулятора PT.

Проект содержит две программы: **Motors_Ctrl** – управление насосами (рис. 14) и **Reg_Ctrl** – управление регулятором температуры (рис. 15). Проект имеет следующие переменные и константы.

Входные переменные:

Alarm_M1 – сигнал неисправности двигателя M1;
Alarm_M2 – сигнал неисправности двигателя M2;
T_wat – температура воды в контуре горячего водоснабжения.

Выходные переменные:

Start_M1 – пуск двигателя M1;
Start_M2 – пуск двигателя M2;
Ctrl – управление приводом PT.

Глобальные переменные:

T_stab – температура стабилизации (задатчик);
Start – пуск одного из двигателей.

Константы, определяющие параметры ПИД-регулятора:

Кп – коэффициент пропорционального управления;
Кдиф – коэффициент дифференциального управления;
Кинт – коэффициент интегрального управления.

Переменные Start и T_stab являются сетевыми, и их значение устанавливает система визуализации верхнего уровня. При получении сигнала Start и отсутствии сигнала Alarm_M1 включается двигатель M1. В том случае, если двигатель M1 неисправен (Alarm_M1= True), включается двигатель M2. Функциональный блок Aver вычисляет среднее значение измеренной температуры воды. Число замеров задается параметрически (в приведенном примере 1000). Вычисленное значение температуры подается на вход ПИД-регулятора. На другой вход регулятора подается значение уставки (в данном примере T_stab). Величина рассогласования между заданным и истинным значением параметра является аргументом функции регулирования. Коэффициенты регулирования задаются параметрически. Выходной сигнал регулятора ограничивается разрешенными пределами цифро-аналогового преобразователя и непосредственно управляет приводом постоянного тока. Все приведенные функции данного проекта реализуются с помощью модуля **5710** Analog Input Card фирмы Octagon Systems.

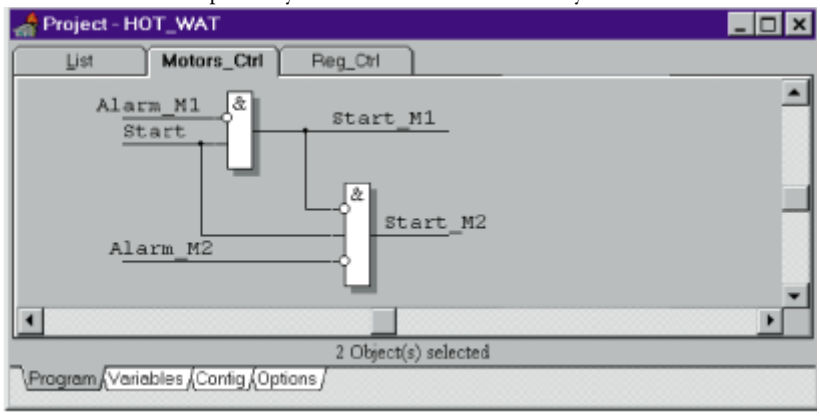


Рис. 14. Программа управления насосами в системе подогрева воды

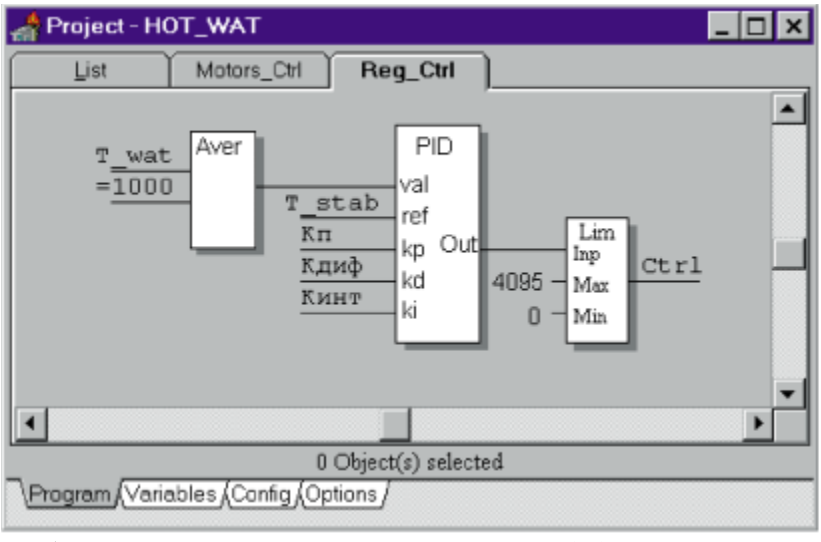


Рис. 15. Программа управления регулятором температуры (PT) в системе подогрева воды

ЗАКЛЮЧЕНИЕ

ULTRALOGIC является одной из немногих отечественных систем программирования, удовлетворяющих требованиям стандарта **IEC-1131**. Это обстоятельство дает основания надеяться на хорошие перспективы ее применения и развития. С помощью **ULTRALOGIC** были реализованы серьезные проекты по автоматизации инженерных сооружений больших зданий, процессов технологических испытаний ряда объектов, научного эксперимента.

Авторы выражают благодарность фирме ProSoft, которая оказала широкую техническую поддержку и консультации при разработке системы.

По вопросу приобретения системы обращаться в фирму ProSoft.
Телефон: (095) 234-0636
Демонстрационная версия программы доступна на web-сервере http://www.prosoft.ru