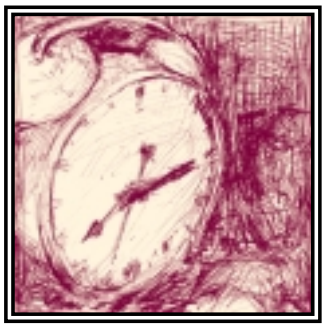

Технологический язык программирования IBM PC совместимых контроллеров

Илья Аблин

Описана система технологического программирования MicPlus.

При выборе технических средств автоматизации одним из основных критериев является наличие программных средств, позволяющих эффективно реализовать прикладные задачи пользователя. Причем под эффективностью понимается прежде всего скорость разработки этих задач и наличие готовых решений для стандартных функций контроля и управления. Естественно, наилучший результат получится, когда специалист в области автоматизации технологии сможет свое понимание задачи воплотить в жизнь без посредников-программистов. Именно этой цели и служат технологические языки программирования. Отсутствие или малое распространение таких языков существенно сдерживает применение свободно программируемых IBM PC совместимых контроллеров. Конечный пользователь зачастую предпочитает выбрать специализированное устройство, обладающее ограниченными возможностями применения, только из-за того, что для его применения не нужно обращаться к программистам. В результате такие устройства, как, например, MicroPC, более популярны среди обладающих квалифицированными кадрами системных интеграторов или производителей комплексного оборудования, чем среди представителей автоматизируемых предприятий. Конечно, нельзя сказать, что до сих пор проблема не была решена совсем. Так, существует ряд западных пакетов (ISaGRAF и др.), практически реализующих международный стандарт на языки для программируемых контроллеров (IEC 1131-3). Однако нам неизвестно об их адаптации к каким-либо из широко распространенных в России IBM PC совместимых контроллеров, в том числе и к MicroPC. Кроме того, применение таких пакетов здесь сдерживается их высокой стоимостью.

В этой статье нам хотелось бы описать свой собственный подход к решению вопроса. Подход, рожденный многолетним опытом работы с технологическими контроллерами, включая и участие когда-то некоторых наших сотрудников, помогающих ныне в развитии дела автоматизации братской Америке, в создании системного программного обеспечения контроллеров Ломиконт – хита 80-х, – устаревшей морально, но до сих пор трудящейся на просторах нашей Родины надежной рабочей лошадки.

Прежде чем рассказать о собственных достижениях, необходимо, видимо, сделать небольшой экскурс в историю вопроса.

Длительное время, фактически с самого своего появления программируемые контроллеры в основном представляли собой устройства одного из трех типов: логический контроллер для управления дискретными процессами, регулирующий контроллер для управления непрерывными процессами и свободно программируемый контроллер для универсального применения. Несводимость первых двух типов контроллеров к третьему была вызвана ограниченными возможностями программного обеспечения свободно программируемых контроллеров. Как для логических, так и для регулирующих контроллеров языки программирования к середине 80-х годов окончательно сложились и были полностью адекватны тем задачам, для которых они применялись. Однако для сложных случаев многофункционального использования удовлетворительного решения не

было найдено. Попытка введения международного стандарта для программирования контроллеров отразила сложившуюся картину – именно многофункциональный язык описан наименее четко. Чтобы понять причину этой ситуации, стоит вкратце рассмотреть особенности основных языков программирования контроллеров.

Первые технологические контроллеры были по преимуществу логическими, и соответствующую специфику имел один из первых технологических языков. Решение было самое простое и в то же время безошибочное – имитировать проектирование той техники, на смену которой пришли контроллеры. Так появился язык релейно-контактных схем. Благодаря сведению новой сущности к старой привычной форме был обеспечен психологически безболезненный переход на новую технику. Язык релейно-контактных схем (РКС) обладал и еще одним преимуществом: оказалось, что процедуры ввода и отладки программ на языке РКС легко могут быть реализованы на примитивных пультах, использовавшихся тогда для программирования контроллеров. По тем временам портативность пульта могла быть обеспечена только существенным ограничением его возможностей, а для программирования на РКС хватало однострочного дисплея и полутора десятков кнопок. Ограниченные возможности техники, имевшей небольшой объем памяти и простейший пульт, необходимость оперативной коррекции программ предъявляли и еще одно требование – язык должен был быть реализован как интерпретируемый. Язык РКС легко вписывался во все имевшиеся тогда ограничения.

Успех программируемых логических контроллеров вскоре разделили одноканальные и многоканальные программируемые цифровые регуляторы. И здесь был использован тот же хорошо себя зарекомендовавший подход: для программирования был создан язык функциональных блоков, повторявший методику создания систем регулирования на использовавшейся ранее технической базе, когда отдельные электронные или пневматические блоки, реализующие ту или иную функцию, соединяли между собой для получения более сложных функциональных возможностей. У нас в стране язык функциональных блоков был применен во всех поколениях широко распространенных контроллеров Ремиконт.

Полученный опыт выявил основные недостатки созданных реализаций. Одни недочеты были заметны даже в той области применения, для которой эти языки предназначались, другие проявляли себя при попытке решить задачи, где потребность в логическом управлении сочеталась с функциями регулирования и вычислений, что характерно, например, для управления периодическими процессами в химии или энергетике. Язык РКС оказался неудобен для описания управления такими шаговыми процессами, где линейный порядок шагов нарушался разветвлениями, параллелизмом и рециклами. Многоканальные схемы регулирования, представленные на языке функциональных блоков, были слишком громоздки и малопредставимы на примитивных пультах. Взаимопроникновение языков, характерное для уже второго их поколения, – вставка функциональных блоков в РКС, использование логических функций в языке функциональных блоков – решало задачу при небольшом отклонении от функционального предназначения языка, но было бессильно в случае их действительно многофункционального применения. Сложные схемы регулирования и вычисления плохо вписывались в логику построения РКС, логические и вычислительные задачи громоздко и ненаглядно решались на языке функциональных блоков. Здесь и начались поиски, которые дали несколько вариантов решения проблемы.

Первый успех принесли попытки организации общей структуры программы. Такой подход позволил произвести декомпозицию задачи за счет введения модульности программ и их иерархического представления. Удачный пример этого подхода – создание французской фирмой Telemecanique языка Графсет, который явился образцом для разработки подобных языков другими фирмами и стал стандартом de facto еще до того, как официальное включение языка шаговых последовательностей в международный стандарт сделало его стандартом de jure. Основные свойства Графсета, явившегося практическим применением разработанных в теории автоматов лет за 15 до его создания сетей Петри, – это представление программы в виде шагов и переходов, а также наличие нескольких одновременно работающих программ и организующей программы, которая отвечает за их включение и отключение. Интересно, что логика самих шагов и переходов во многих реализациях может быть описана на языке РКС. Введение в язык шаговых последовательностей параллелизма шагов, наличие ветвлений, а в некоторых версиях и циклов, обеспечило в сочетании с применением РКС и структуризацией задачи достаточную гибкость для потребностей управления подавляющим большинством дискретных и периодических процессов.

Необходимость сочетания непрерывного и дискретного управления на большинстве объектов химии, нефтехимии, металлургии, энергетики и других отраслей вела к поиску универсального технологического языка. Началом пути к нему стала попытка адаптации универсальных языков программирования. В первых свободно программируемых контроллерах применялся ассемблер, затем появились языки высокого уровня (Си, Бейсик), но в любом случае для того, чтобы писать на этих языках, нужны были профессиональные программисты, а сам процесс разработки программ был достаточно трудоемким и требовал кроссовых средств, реализованных на компьютере. Использование пультов для программирования исключалось из-за сложности самих языков, а также значительных ресурсов, необходимых для компиляции и рекомпиляции в случае внесения изменений. Предложенное в ряде реализаций стандарта внешнее представление программы на любом из «гостированных» языков с возможностью переключения между ними проблему не сняло, поскольку в практическом применении оказалось ненаглядным.

С другой стороны, стремительное развитие персональных компьютеров привело к тому, что лучшим пультом программирования для любого контроллера стал обычный офисный сначала laptop, а теперь и notebook. В результате ограничения на сложность языка были сняты и оказалось возможным перейти от программ-интерпретаторов, снижавших требования к памяти, занимаемой прикладной программой, и облегчавших процесс программирования с помощью специализированных пультов, к компиляторам, работающим на инструментальной машине и создающим загрузочный код для исполнения в контроллере.

Принцип адаптации универсальных языков программирования был прост: в язык вводились новые типы переменных – «входы» и «выходы» контроллера, а также создавалась некоторая библиотека подпрограмм, реализующих наиболее часто встречающиеся алгоритмы. Программирование упростилось, но все еще оставалось сложным для рядового технолога или «автоматчика-киповца». И тогда был сделан следующий

шаг – создание на базе распространенных языков программирования процедурного типа специализированных технологических языков, ориентированных именно на конечного пользователя на объекте. (В международном стандарте такой технологический язык получил название «структурированный текст».) Основным отличием текстовых технологических языков от универсальных стало, во-первых, резкое упрощение синтаксиса и семантики (сокращение числа типов операторов, сложности выражений и т. п.), а во-вторых, введение в язык специальных технологических понятий, реализующих типовые функции контроля и управления. Здесь открылся огромный простор для творчества, и число языков такого типа на первом этапе, видимо, совпало с числом фирм-разработчиков. Такая разноголосица естественна для периода становления любой области техники, что нашло отражение и в достаточно скупом описании языка в созданном по горячим следам стандарте.

Наши собственные поиски шли в том же направлении. Убедившись, с одной стороны, в функциональной ограниченности графических языков (РКС, функциональных блоков и шаговых последовательностей) и как никто другой зная недостатки примитивного текстового языка Микрол контроллеров Ломиконт, мы предприняли попытку создания такого языка, который бы удовлетворял нашему пониманию потребностей пользователей. Название Микрол+ возникло раньше реализации самого языка с целью подчеркнуть некоторую преемственность по отношению к языку Микрол контроллеров Ломиконт. В то же время одинокий плюс после названия «Микрол» не отражает всей разницы в функциональной мощности этих языков.

Естественно, увеличение возможностей неизбежно в чем-то усложняет язык, хотя и позволяет создавать более компактные и надежные программы, легко тиражировать найденные решения типовых задач. Однако все дополнительные и более сложные элементы языка не обязательны для использования и их описание вынесено за рамки основного руководства по языку. Еще один шаг в направлении облегчения жизни разработчиков был сделан за счет создания тесной связи между интегрированной средой разработки, проектом, в рамках которого должна вестись работа, и самим языком.

Первоначально система технологического программирования MicPlus создавалась для контроллеров серии «Техноконт» – ТСМ51 и МФК [1]. Однако, учитывая то обстоятельство, что контроллер МФК, в сущности, является расширением системы ввода-вывода MicroPC, мы выяснили, что адаптация MicPlus к любому IBM PC совместимому контроллеру не составляет особого труда и не обязательно должна быть произведена авторами системы. Для облегчения этой процедуры в состав интегрированной среды включены соответствующие диалоговые средства. Каждый новый контроллер подключается к системе путем описания используемых плат ввода-вывода, их информационной емкости, типа и имен обслуживающих эти платы драйверов.

Система MicPlus состоит из двух частей: инструментальной, предназначенной для разработки пользовательских программ, и целевой, работающей непосредственно в контроллере (рис. 1). Целевая система содержит монитор реального времени и средства ввода-вывода информации через платы и каналы связи. Монитор обеспечивает вытесняющую приоритетную многозадачность и управляет работой подсистемы ввода-вывода, к которой, в свою оче-

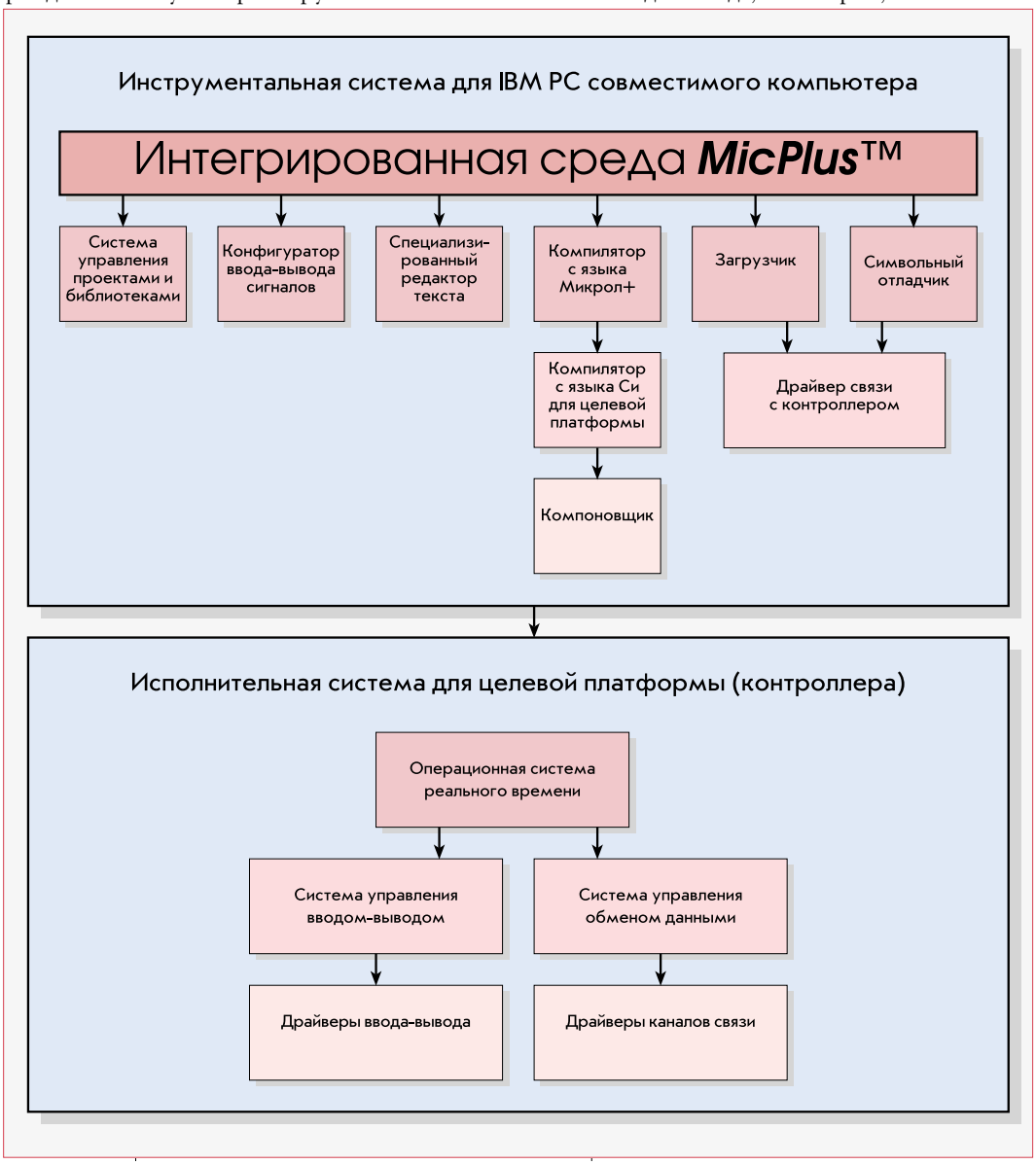

Рис. 1. Архитектура системы программирования на технологическом языке Микрол Плюс

редь, подключаются драйверы для конкретных плат. Монитор также обеспечивает загрузку и запуск прикладных программ, оформленных в виде отдельных EXE-модулей.

Основное понятие, с которым работает пользователь в интегрированной системе MicPlus, – проект. Проект содержит список контроллеров, в каждом из которых может быть несколько задач – программ пользователя (ПрП). Задачам указывается приоритет и период запуска. Отсутствие периода означает выполнение задачи в непрерывном цикле. Пользователь описывает конкретный контроллер, указывая последовательность расположения плат из списка имеющихся для данного типа контроллера. Впоследствии при описании переменных каждому имени переменной ставится в соответствие ее тип (аналоговый или дискретный, вход или выход) и адрес (номер платы и номер сигнала данного типа на плате). Наряду с таким «географическим» адресом переменной можно использовать и ее порядковый номер в контексте – последовательном массиве значений сигналов данного типа, который формируется драйверами в памяти контроллера.

Структура пользовательской программы включает заголовок и программные блоки. В заголовке, состоящем из нескольких разделов, описываются переменные, константы, шаблоны текстовых сообщений, список вызываемых внешних библиотек, а также текст используемых в программе процедур и функций. Программные блоки, в свою очередь, имеют заголовок, в котором приведен перечень локальных переменных, и программные секции, содержащие непосредственно текст программы, описывающей логику контроля и управления технологическим объектом. В интегрированной среде структура программы представлена в виде дерева (рис. 2), где каждая часть – заголовок или программный блок – может быть видна в сжатом или раскрытом виде. Блоки и секции имеют имена. Блокам свойственны состояние (включенное или отключенное) при первом запуске программы и кратность выполнения по отношению к ее периоду запуска. Входящие в состав блока программные секции также имеют первоначальное состояние. Блоки и секции во время своего выполнения могут быть включены и отключены непосредственно программой пользователя или средствами верхнего уровня. Количество блоков и секций, а также

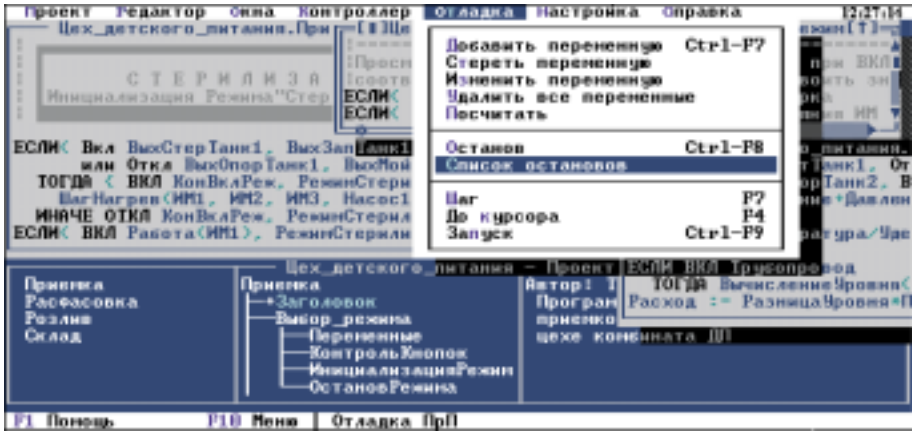

Рис. 2. Экран интегрированной системы программирования MicPlus

их размер ограничены только общим объемом оперативной памяти контроллера. Текст секции вводится в окне специализированного редактора, поддерживающего ввод любых конструкций языка по нажатию соответствующих им «горячих» клавиш, а также обеспечивающего синтаксическую раскраску текста, которая облегчает его восприятие и визуальный контроль. Количество открытых одновременно для редактирования окон программных секций или разделов заголовка практически не ограничено. В произвольном месте программы может быть расположен комментарий.

Язык Микрол+ поддерживает ряд типов данных: ВА, АВ – аналоговые входы и выходы (целое – 2 байта), ВД, ДВ – дискретные входы и выходы, РА и РД – рабочие (промежуточные) переменные, Ве – вещественные (3 байта мантисса и 1 байт порядок), Дл – длинные (двойное целое – 4 байта), КБ и КС – ключи блоков и секций (дискретные переменные, управляющие запуском и остановом соответствующих разделов программы). В дальнейшем предполагается также осторожное (не в ущерб простоте языка) использование некоторых сложных типов данных (массивов и структур). Над всеми аналоговыми переменными можно выполнять четыре действия арифметики. Результат может быть присвоен переменной любого типа, кроме входной. Дискретные переменные, кроме входных, можно включать и отключать, а также проверять их состояние. Дискретным константам ВКЛ и ОТКЛ, с помощью которых производятся эти действия, пользователь может присваивать различные имена-синонимы, используемые в программе одновременно, например, «Открыть/Закрыть», «Вперед/Назад», «Есть/Нет».

Кроме аналоговых и дискретных переменных, в языке Микрол+ существуют также специальные переменные для счета времени – таймеры. Имеется пять типов таймеров. Три из них – это таймеры счета времени с дискретностью счета 10 миллисекунд (ТМД), одна секунда (ТМС) и одна минута (ТММ) соответственно. Кроме них, в языке имеется два специальных тайме-

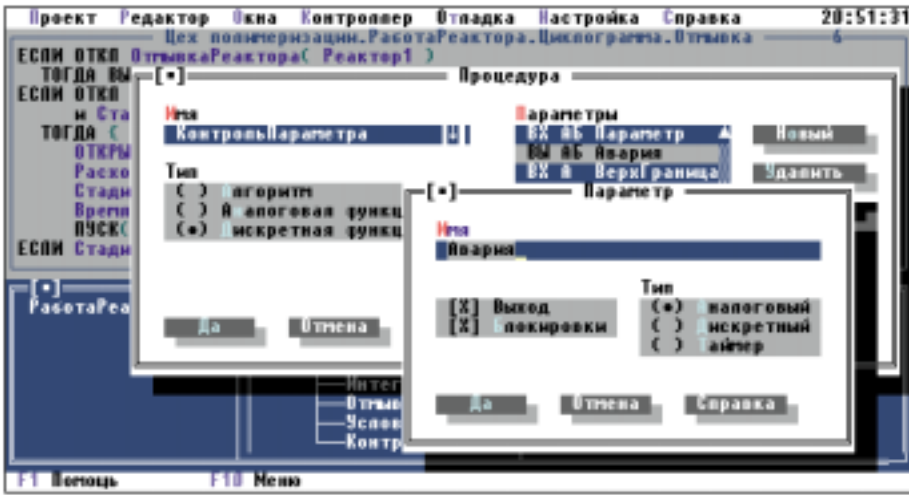

Рис. 3

ра: ВРЕМЯ (отсчет астрономического времени) и ДАТА (отсчет календарной даты).

Наряду со значением таймер счета времени характеризуется состоянием: СТАРТ (таймер считает время) или СТОП (значение таймера со временем не изменяется). Если исходное значение таймера счета времени положительно, то оно растет до максимального целого значения и таймер отключается (счет времени). Если исходное значение таймера отрицательно, то оно растет до нуля и таймер также отключается (выдержка времени).

Обновление значения входных переменных в соответствии с текущим значением физических входов контроллера производится перед очередным циклом ПрП; передача значений выходных переменных на выходы контроллера производится после его завершения. Приращение физического времени, накопившееся за очередной цикл выполнения ПрП, прибавляется к значению таймера перед началом нового цикла.

Каждая переменная наряду со значением характеризуется состоянием блокировки. Значение разблокированной переменной формируется в самом контроллере, при этом значения входных переменных ВД и ВА зависят от входных сигналов на модулях ввода дискретных и аналоговых сигналов, в то время как значения остальных переменных могут быть изменены в процессе выполнения программы. Если же переменная блокирована, изменение сигналов на входе контроллера не приводит к изменению соответствующих входных переменных, а при выполнении программы не происходит программного изменения заблокированных рабочих или выходных переменных, включения или отключения блокированных блоков и секций. Блокировка таймера означает, что управляющая программа не может ни запустить, ни остановить таймер, ни присвоить ему произвольное значение времени. Если таймер запущен, счет времени в нем продолжается независимо от состояния блокировки. Блокирование блока или секции не останавливает их работы, но не позволяет программе управления включать или отключать их.

Блокируя переменные, оператор может с помощью средств верхнего уровня или с пульта контроллера, если таковой имеется, производить ручное управление исполнительными механизмами, задавать значения входов (например, при неисправности датчиков входной информации), изменять задание регулятора и т. д. Операция блокирования незаменима также при отладке программы.

```
ЕСЛИ ( Давление < НОРМА И готов Продукт(Реактор1) )
  ТОГДА { Закрыть Клапан1, Клапан2;
          Включить Насос;
  }
  ИНАЧЕ { Сообщение Авария( Давление );
          АварийнаяСигнализация ( Давление );
          Открыть Клапан2;
          Пуск( ВремяАварии );
  }
```

Пример программы на языке Микрол+

В языке Микрол+ существует два основных типа операторов: безусловный и условный. Выполнение условного оператора зависит от результата проверки логического условия, представленного в виде простого или сложного логического выражения, в то время как безусловный оператор выполняется всегда, когда в процессе выполнения программы до него дошла очередь.

Условный оператор имеет вид ЕСЛИ... ТОГДА... ИНАЧЕ... Безусловные операторы – это арифметические выражения, операторы включения или отключения дискретных переменных, оператор перехода на метку (только вперед по тексту), вызов процедуры, вывод технологического сообщения (эти сообщения, предназначенные для информирования оператора-технолога или для отладки ПрП, могут быть выведены на принтер, дисплей и т. п.). Исполнительная часть условного оператора аналогична простому или составному безусловному оператору. Арифметические и логические выражения могут содержать функции.

В языке осознанно отсутствуют операторы цикла, поскольку они могут быть элементарно реализованы через цикл самой программы, а их наличие в программе не только приводит к ее потенциальной ненадежности, но и делает невозможным любой прогноз относительно гарантированного времени выполнения.

Функции и процедуры могут быть созданы пользователем для конкретной ПрП – в этом случае они помещаются в ее заголовке, а могут быть рассчитаны и на применение в различных программах – тогда создаются библиотеки. Пользователь может формировать библиотеки из программ, написанных как на Микрол+, так и на Си. Уже существующие библиотеки обеспечивают решение задач регулирования, динамических преобразований и других.

Созданная программа пользователя транслируется входящим в состав системы компилятором, который создает не загрузочный код, а промежуточное представление - программу на языке Си. Именно благодаря этому подходу и обеспечивается легкий перенос системы программирования на различные целевые платформы, а также ее открытость и расширяемость.

Найденные в ходе трансляции ошибки отображаются в специальном окне, из которого возможен переход на соответствующую строку программы.

Созданная программа может быть загружена в контроллер и запущена на выполнение в рабочем или отладочном режиме (рис. 3). В системе реализованы все стандартные функции символьного отладчика (остановы, трассировка, просмотр, изменение и вычисление переменных).

Имеющийся опыт применения системы программирования MicPlus для контроллеров серии «Техноконт» показал возможность создания наглядных, компактных и самодокументируемых программ, которые могут быть перенесены с одного типа контроллера на другой без каких-либо изменений в исходном тексте ПрП. Весьма важной оказалась возможность тиражирования всех однажды найденных решений в последующих проектах. Наметившаяся тенденция ко все более широкому использованию IBM PC совместимых контроллеров обещает этой мощной системе разработки программ контроля и управления большое будущее.●

Литература

1. Сережин Л.П. Многофункциональный комплекс программно-аппаратных средств для построения распределенных систем управления – МФК «Техноконт»// Приборы и системы управления.– 1995.– № 10.