РОССИЙСКАЯ АКАДЕМИЯ НАУК
СИБИРСКОЕ ОТДЕЛЕНИЕ
АВТОМЕТРИЯ
№ 2 1996

УДК 681.3.06

**В. Е. Зюбин**
*(Новосибирск)*

# ЯЗЫК СПАРМ — СРЕДСТВО ПРОГРАММИРОВАНИЯ МИКРОКОНТРОЛЛЕРОВ

Предлагается новый специализированный язык высокого уровня, предназначенный для программирования устройств логического управления. Язык имеет блочное строение, в котором алгоритм работы устройства представляется как совокупность описаний слабозависимых параллельно исполняемых процессов в текстовом виде. Излагается синтаксис языка.

**Введение.** Проблема создания языка для программируемых контроллеров привлекает внимание специалистов, однако многочисленные «классические» языки, разработанные в теории автоматов (язык регулярных функций, таблицы включения, матрицы перехода, язык временных функций, язык релейно-контактных схем и т. д., см., например, [1—3]), оказались малопригодными для описания работы сложных дискретных устройств. Общим недостатком разработанных языков является то, что на них практически невозможно описать работу объектов большой размерности, в частности, объектов, предполагающих параллелизм исполнения алгоритма. Процесс формирования записи на этих языках мало соответствует методике составления технических заданий, во многих языках отсутствуют арифметические операции и возможность задания полноценных идентификаторов.

Определенный интерес представляет язык ЯРУС [4]. Предложенный в этом языке способ представления алгоритма работы в виде совокупности параллельно исполняемых автоматов позволил существенно увеличить размерность описываемых объектов. К сожалению, избыточно-сложный синтаксис, отсутствие идентификаторов и арифметических операций, отдельные особенности реализации негативным образом сказались на его распространении.

Одна из последних попыток решить проблему программирования микроконтроллеров — язык ISaGRAF [5], предложенный фирмой "CJ International". Несмотря на широкий набор возможностей и сервисных услуг среды ISaGRAF, ряд принципиальных недостатков ограничивает область его применения:

1. ISaGRAF представлен совокупностью пяти разнородных специализированных языков, каждый из которых, не являясь самодостаточным, предназначен для решения строго определенных, иногда частично перекрывающихся подклассов задач, т. е. язык ISaGRAF не предоставляет единой методологии при создании программ.

2. Параллельно исполняемые процессы в среде ISaGRAF имеют жесткую иерархическую структуру. Такой подход является слабоэффективным по двум причинам. Во-первых, жесткая иерархическая организация несовместима с возможностью динамического изменения хода выполнения программы, например, при описании действий в случае отказа исполнительных органов, во-вторых, жесткая связь «порождающий процесс — дочерний процесс» обус-

ловливает дублирование, т. е. неоправданное увеличение исполняемого кода, при необходимости инициировать исполнение некоторого процесса на разных уровнях иерархической структуры.

3. В языке ISaGRAF отсутствуют средства обмена статусной информацией, т. е. в среде ISaGRAF для получения информации о текущем состоянии неко-

При этом ни один из вышеприведенных языков не предоставляет возможности программирования мультипроцессорных сред, которые в настоящее время являются неотъемлемой частью современных автоматизированных систем управления многомерными объектами.

Хотя следует заметить, что в определенных случаях, при относительно простых алгоритмах управления, применение таких языков достаточно эффективно.

На основе анализа требований, предъявляемых задачами описания многомерных объектов, и недостатков, присущих существующим подходам, можно перечислить необходимые характеристики эффективного языка программирования сложных устройств автоматики:

1. Ориентация на описание устройств большой размерности, в частности, устройств, алгоритмы которых предполагают параллелизм исполнения.

2. Отсутствие ограничений на структуру описываемых алгоритмов, например, в виде жестких иерархических связей.

3. Содействие четкому стилю программирования за счет единой методологической основы создания программ, легкого для запоминания русскоязычного синтаксиса, возможности создания полноценных идентификаторов и отсутствия дополнительных требований к квалификации программиста.

4. Полный доступ к портам и ячейкам памяти, предполагающий возможность создания объектов произвольной разрядности (в том числе и логических) вне зависимости от физического уровня реализации аппаратуры.

При реализации языка необходимо учитывать следующие факторы: облегчение адаптации языка на процессорах различной архитектуры, сокращение накладных временных расходов, связанных с организацией параллельного исполнения алгоритма, наличие интерфейса с другими языками программирования.

Ниже излагается синтаксис языка СПАРМ [6], предназначенного для решения этого класса задач.

В качестве основы для создания методики описания алгоритма работы дискретного устройства была взята теория конечных автоматов — теория хорошо разработанная и в настоящее время наиболее близкая к методике составления технического задания.

Для описания дискретного устройства принята математическая модель, используемая в [2], с некоторыми несущественными изменениями в обозначениях:

$$A = \{S, X, Y, \delta, \lambda, s_1\} \text{ — автомат,}$$

где $S$ — множество состояний (алфавит состояний); $X$ — множество входных сигналов (входной алфавит); $Y$ — множество выходных сигналов (выходной алфавит); $\delta$ — функция переходов; $\lambda$ — функция выходов; $s_1$ — начальное состояние автомата.

**Описание языка.** СПАРМ имеет блочное строение. Алгоритм работы контроллера представляется в виде совокупности процессов. Процесс — это, по сути, $C$-автомат, наделенный дополнительными свойствами, описанными ниже, которые относятся к взаимодействию между процессами.

Разбиение алгоритма работы сложного устройства на процессы проводится исходя из функциональных, конструктивных, технологических и других соображений, что позволяет представить такой алгоритм как совокупность слабозависимых, хорошо обозримых процессов.

Описание работы внутри процесса состоит из описания последовательности состояний, возникающих при функционировании устройства.

Описание процессов составляет первый уровень языка.

Второй уровень, образованный совокупностью состояний, задает функцию выходов, функцию переходов и взаимодействие между процессами.

Описание алгоритма работы устройства на языке СПАРМ состоит из двух частей: часть 1 содержит описание констант и физических портов ввода/вывода, часть 2 состоит из процессов, внутри которых располагаются описание входного/выходного алфавита (переменных) и описания состояний этих процессов.

Таким образом задаются все шесть элементов множества $A$: $S$, $X$, $Y$, $\delta$, $\lambda$ и $s_1$, которым является состояние, описанное внутри процесса первым.

**Константы.** Определение понятия «константа» не отличается от принятого в «классических» языках программирования.

Описание константы, расположенное в первой части программы, устанавливает отношение эквивалентности между некоторым произвольно выбранным программистом идентификатором и числом.

**Порты ввода/вывода.** Порты ввода/вывода рассматриваются как отдельные компоненты языка, которые служат для задания привязки переменных к интерфейсной аппаратуре. Порты различаются по принадлежности ко входным или выходным портам, по размеру и адресу. Синтаксис языка предоставляет программисту возможность определять физическую ячейку памяти как порт ввода/вывода.

Описание портов устанавливает отношение эквивалентности между некоторым произвольно выбранным идентификатором и портом (или ячейкой памяти).

**Переменные.** В языке СПАРМ переменные различаются по следующим признакам: размеру, наличию зависимости от состояния портов ввода/вывода, степени доступа.

Описания переменных располагаются до описания первого состояния процесса. Каждая используемая в процессе переменная должна быть так или иначе описана. Описание переменной устанавливает отношение эквивалентности между некоторым произвольно выбранным идентификатором и битом (последовательностью битов) конкретного физического порта, тем самым позволяя в дальнейшем при описании алгоритма работы устройства отвлечься от способов реализации интерфейсной аппаратуры и оперировать только терминами состояний объектов. Кроме того, описание переменных совместно с описанием портов несет информацию, достаточную для автоматической, а значит, безошибочной генерации всех действий по считыванию, преобразованию в переменные и модификации физических портов.

По размеру различаются логические, короткие целые, целые и длинные целые переменные.

По зависимости от состояния портов ввода/вывода различаются переменные, зависимые от состояния портов и не зависимые от них. Переменные, зависимые от состояния портов, — это те переменные, у которых значение битов поставлено в зависимость от состояния битов конкретного физического порта.

Степень доступа к переменной может быть трех типов: «локальная» (видимая только из процесса, в котором она описана), «полностью открытая» (видимая из любого процесса), «частично открытая» (видимая из нескольких, но не всех процессов). В случае «полностью» или «частично открытой» переменной достаточно, единожды описав ее в некотором процессе, в дальнейшем вместо повторного описания использовать ссылку.

**Процессы и состояния.** Как уже было отмечено, процесс — это $C$-автомат, наделенный дополнительными свойствами. Дополнительные свойства процесса заключаются в возможности влиять на ход выполнения алгоритмов работы других процессов и изменять ход выполнения своего алгоритма работы в зависимости от условий, возникающих в других процессах.

1. Процесс имеет три выделенных состояния: «нормального останова», «останова по ошибке» и паузы. Два состояния останова необходимы для контроля причин, по которым процесс прекратил свое выполнение. Состояние паузы необходимо для обеспечения останова процесса на заданное время.

2. Процесс имеет доступ к информации о состоянии любого другого процесса-автомата (ситуации, возникшей в другом процессе).

3. Процесс имеет возможность влиять на состояния любого другого (включая, разумеется, и себя) процесса, т. е. переводить процессы в другие состояния, как выделенные (состояния останова и паузы), так и не выделенные.

4. Возможны процессы, использующие пересекающиеся множества входных и выходных переменных, т. е. процесс имеет доступ к переменным, принадлежащим другому процессу.

Интересно отметить, что свойства п. 3 связаны с изменением некоторых значений и в языке СПАРМ трактуются как операции над множеством выходных сигналов, а свойства п. 2, по сути, являются операциями с множеством входных сигналов (связаны с анализом некоторых значений).

Строго говоря, необходимо ввести понятие $P$-автомата, который представляет собой множество процессов: $P = \{p_1, ..., p_n, ..., p_N\}$. Отдельно взятый процесс $p_n$, в свою очередь, задается множеством из шести элементов:

$$p_n = \{S^n, X^n, Y^n, \delta^n, \lambda^n, s_1^n\}.$$

Несмотря на то что в языке СПАРМ эти элементы трактуются более широко, смысловое значение элементов множества $p_n$ совпадает со смысловым значением элементов «классического» автомата $A$.

Описание процесса устанавливает отношение эквивалентности между некоторым произвольно выбранным программистом идентификатором и процессом для того, чтобы при описании алгоритма работы устройства оперировать именем процесса, кратко обозначающим выполняемую процессом функцию, не отвлекаясь на то, как конкретно эта функция реализована. Описание процесса состоит из имени процесса и тела процесса. Тело процесса содержит описание переменных процесса и описание состояний процесса. Описание переменных процесса располагается до описания состояний процесса.

В свою очередь, описание состояния устанавливает отношение эквивалентности между некоторым произвольно выбранным оператором идентификатором и состоянием процесса. Описание состояния содержит имя состояния и тело состояния. Тело состояния представляет собой определение функции выходов и функции переходов с помощью обычных принятых в алгоритмических языках условных и безусловных выражений.

Существуют два ограничения, продиктованные потребностью исключить возможность создания состояний, из которых нет переходов в какое-либо другое состояние:

а) в состоянии обязательно присутствует выражение, изменяющее состояние текущего процесса;

б) внутри тела состояния нет возможности организации циклов и переходов, хотя эти возможности и предоставляются другими алгоритмическими языками.

**Разбиение на процессоры.** Принятый в языке способ представления алгоритма работы устройства в виде совокупности слабозависимых параллельно исполняемых процессов позволяет разносить запрограммированный алгоритм по активным устройствам мультипроцессорных сред, широко используемых в настоящее время при автоматизации промышленных объектов. Распределение алгоритма по процессорам производится с помощью прагмы «//НОВЫЙ ПРО-

ЦЕССОР», которая может располагаться в тексте программы между описаниями процессов.

**Такт.** Введение в синтаксис языка понятия такта преследует двоякую цель. Во-первых, описание такта задает время реакции системы на внешние события, т. е., другими словами, тактирует переходы процессов из состояния в состояние; во-вторых, обеспечивает привязку алгоритма работы устройства к таймеру, организуя полезную и в большинстве случаев необходимую службу времени.

Можно сделать еще одно интересное замечание: строго говоря, введение такта является, по сути, введением в множество входных сигналов $X$ $P$-автомата неявных временных входных сигналов. Очевидно, что такая трактовка времени полностью снимает вопрос о синхронности $P$-автомата, поскольку условие устойчивости состояния вырождается в утверждение об устойчивости состояния при отсутствии течения времени.

**Методика создания программ.** Суммируя вышеизложенное, методику создания программ сводим к следующим простым правилам.

1. Функционирование устройства разбивается на несколько процессов, по возможности независимых или слабозависимых. Это разбиение происходит по конструктивным, функциональным и другим соображениям.

2. Создаются описания процессов, которые содержат описания последовательности состояний, возникающих во время функционирования устройства.

3. Создаются описания состояний, которые содержат некоторую последовательность операций над регистрами ввода/вывода и условия перехода из данного состояния в следующее.

**Синтаксический граф языка СПАРМ.** Одним из способов описания синтаксиса языка, или, другими словами, правил построения корректных языковых конструкций, является способ описания порождающих правил с помощью синтаксического графа, предложенного в [7]. Эта достаточно удобная и наглядная форма во многих случаях предпочтительнее формы Бэкуса — Наура.

Правила чтения записанного в такой форме синтаксиса следующие:

1) символом при такой форме записи называется элемент, обведенный прямоугольной либо овальной линией;

2) последовательностью при такой форме записи называется последовательность разделенных стрелками символов;

3) стрелки указывают направление чтения синтаксического графа;

4) символ, обведенный прямоугольной линией, — это терминальный символ, т. е. набор конкретных символов, который фиксирован и не может быть заменен в тексте на другой;

5) символ, обведенный овальной линией, — это нетерминальный символ, т. е. символ, имеющий нетривиальный смысл, который раскрывается отдельной последовательностью;

6) каждая последовательность раскрывает смысл некоторого указанного перед последовательностью нетерминального символа;

7) раздвоение стрелки указывает на возможность выбора пути чтения последовательности.

Символы пробела, табуляции, перевода строки и комментарии — это разделительные символы, которые должны использоваться программистом для формирования удобочитаемого листинга программы. Несколько имен, чисел и (или) резервированных слов, расположенных последовательно, должны быть отделены, по крайней мере, одним разделительным символом, чтобы транслятор мог распознавать их как различные объекты. Комментарием в языке СПАРМ является текст, начинающийся последовательностью символов '/' и '*'. Комментарий заканчивается последовательностью символов '*' и '/'.

В любом месте, где разрешено использование разделительного символа, допускается включать конструкции языка Си. Такие конструкции начинаются последовательностью из двух символов '//', которая действует до конца строки.

Синтаксические диаграммы языка СПАРМ приведены в приложении.

**Заключение.** Разработанный язык принадлежит к классу специализированных языков высокого уровня и предназначен для программирования алгоритма работы автоматизированных систем управления.

Среди основных результатов, достигнутых в данной работе, можно особо отметить предусмотренные в языке средства для задания привязки переменных к интерфейсной аппаратуре, предоставление русскоязычному пользователю возможности создания полноценных идентификаторов для переменных различной разрядности, широкий набор логических и арифметических операций с переменными длиной до 32 разрядов.

Конструкция языка способствует четкому стилю программирования, в котором программы легко читаются и модифицируются, а листинги программ могут служить в качестве удобной формы документирования запрограммированных алгоритмов.

Текстовая форма записи, хотя и считается менее наглядной по сравнению с графической формой описания алгоритмов, однако является более предпочтительной по сравнению с графической при описании сложных объектов большой размерности, поскольку предоставляет широкий набор выразительных средств, не накладывает ограничений на длину идентификаторов, размер и сложность записей логических и арифметических функций, что в конечном счете облегчает работу пользователя.

Язык дает полный доступ ко всем портам и ячейкам памяти.

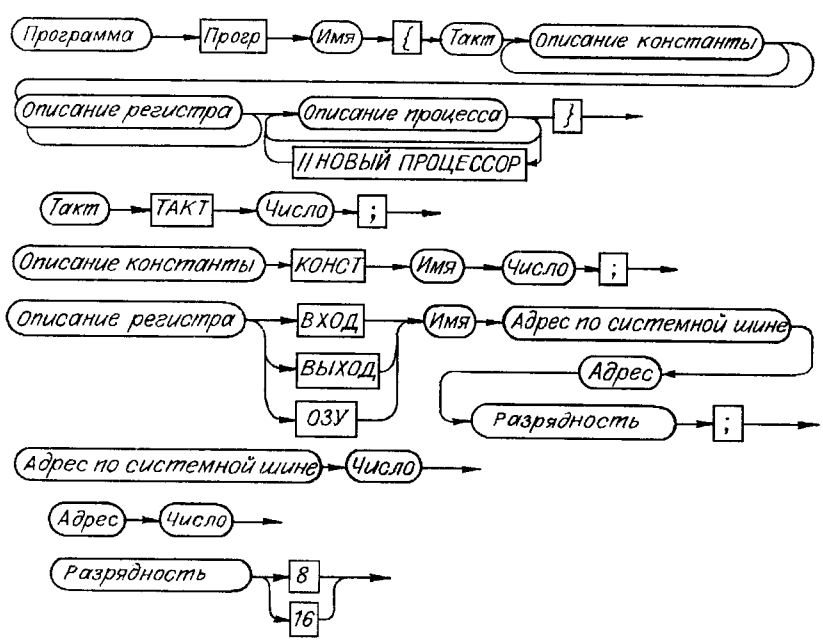
Блочная структура записи позволяет описывать объекты управления практически любой размерности.

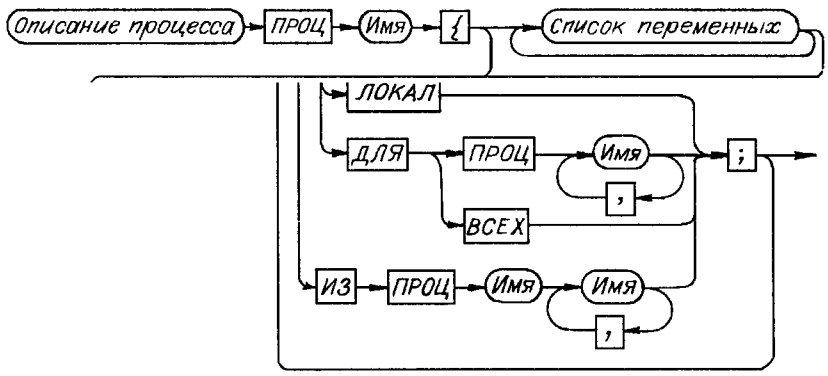
Язык предоставляет возможность распределять алгоритм работы по активным устройствам мультипроцессорных сред.

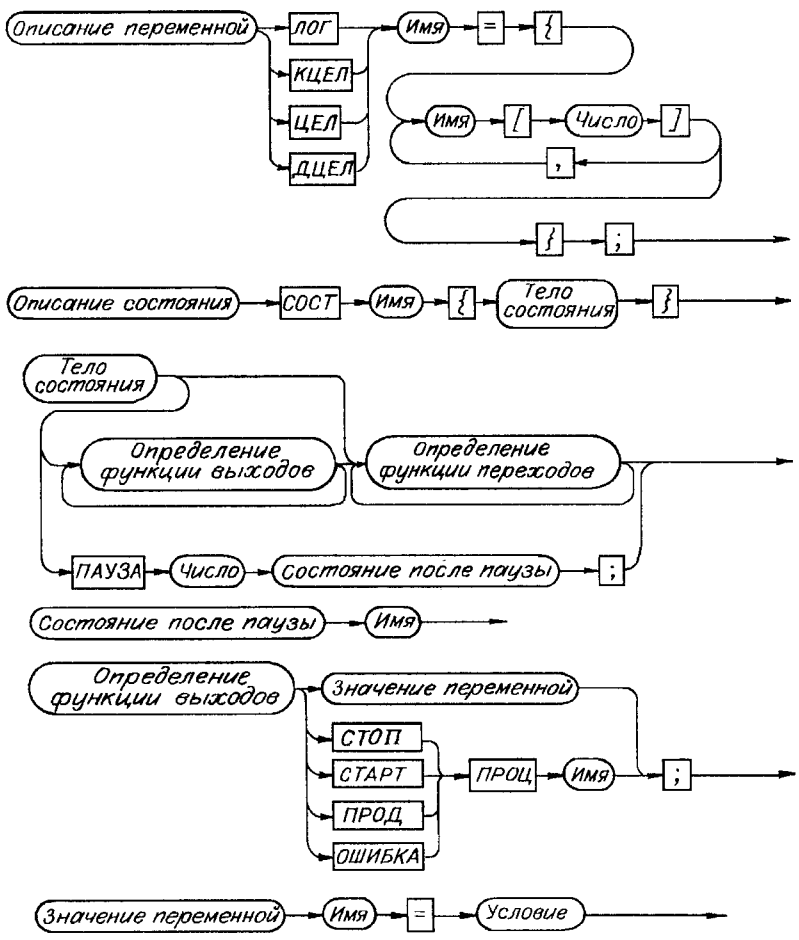
Встроенный интерфейс с языком Си позволяет пользователю создавать программные блоки на других языках программирования.

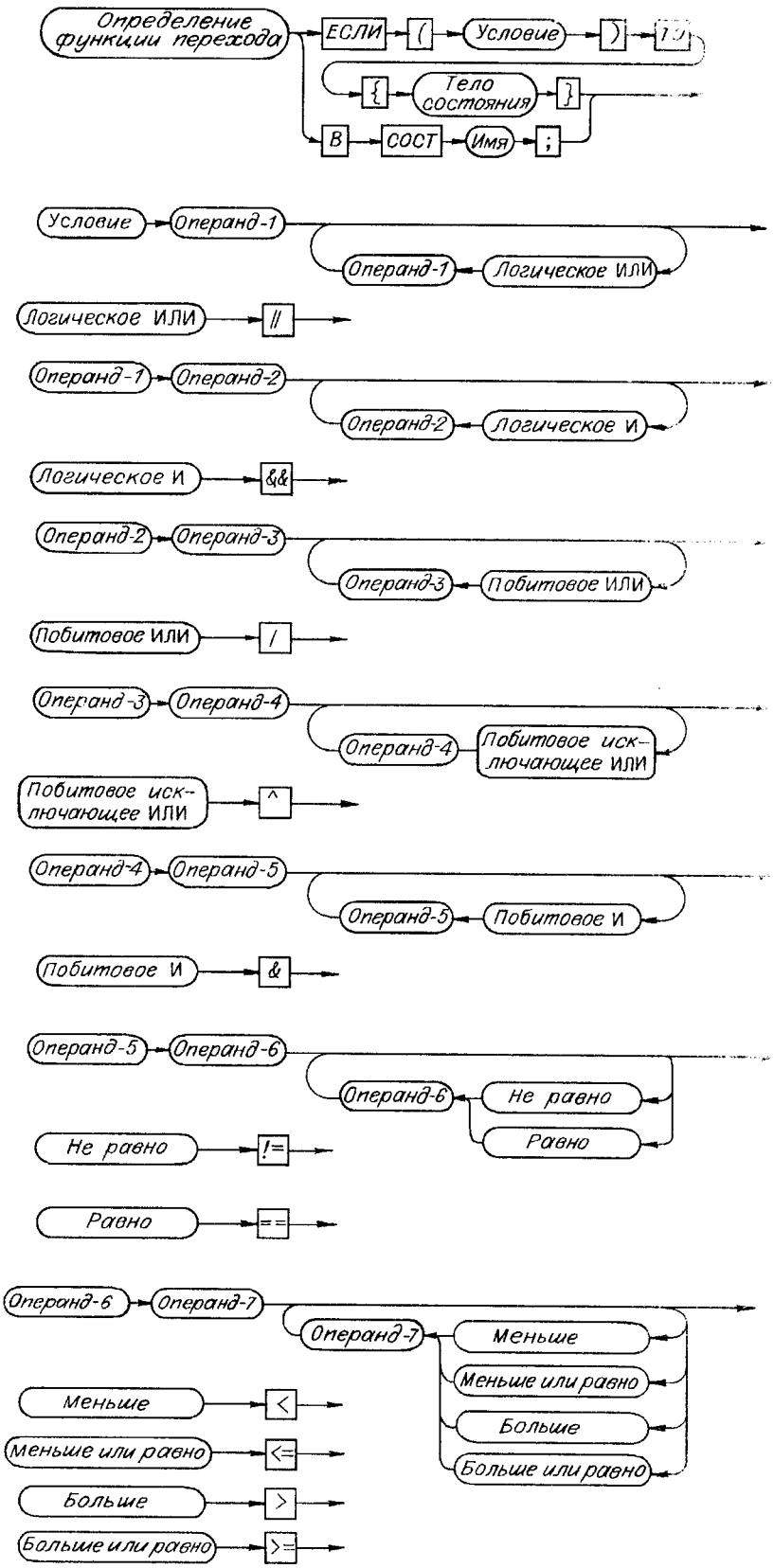
В настоящее время автором предполагается ввести в синтаксис языка возможность создания макроописаний. Прорабатывается вариант включения в синтаксис языка конструкций для поддержки сетевой организации мультипроцессорных систем и обеспечения автоматической реконфигурации мультипроцессорной системы в случае отказа составных элементов.

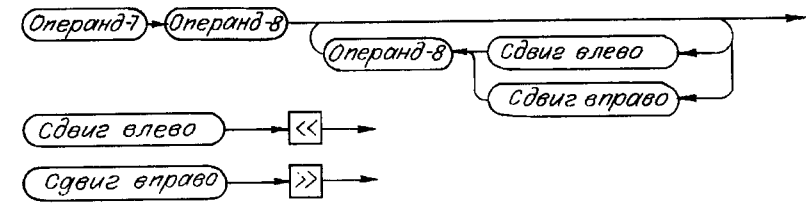
**ПРИЛОЖЕНИЕ**

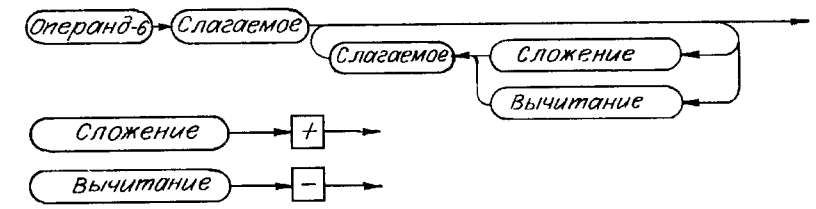


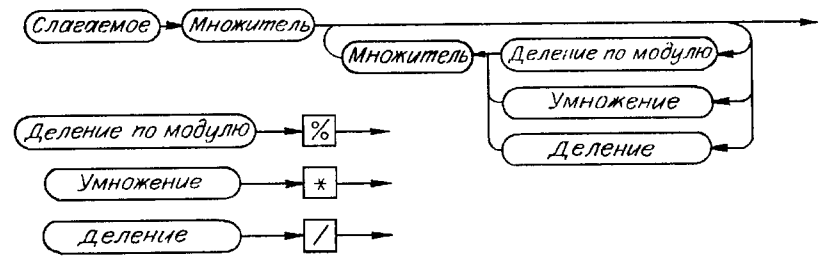

Описание процесса
ПРОЦ
Имя
{
Список переменных
ЛОКАЛ
ДЛЯ
ПРОЦ
Имя
,
ВСЕХ
;
ИЗ
ПРОЦ
Имя
Имя
,

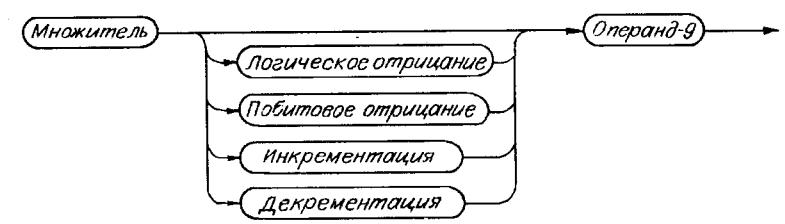

Описание переменной
ЛОГ
КЦЕЛ
ЦЕЛ
ДЦЕЛ
Имя
=
{
Имя
[
Число
]
,
}
;
Описание состояния
СОСТ
Имя
{
Тело состояния
}
Тело состояния
Определение функции выходов
Определение функции переходов
ПАУЗА
Число
Состояние после паузы
;
Состояние после паузы
Имя
Определение функции выходов
Значение переменной
СТОП
СТАРТ
ПРОД
ОШИБКА
ПРОЦ
Имя
;
Значение переменной
Имя
=
Условие

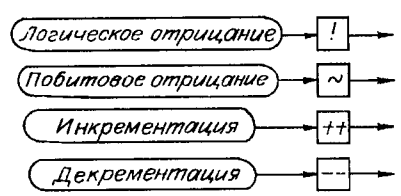

Определение функции перехода
ЕСЛИ
(
Условие
)
{
Тело состояния
}
В
СОСТ
Имя
;
Условие
Операнд-1
Операнд-1
Логическое ИЛИ
Логическое ИЛИ
||
Операнд-1
Операнд-2
Операнд-2
Логическое И
Логическое И
&&
Операнд-2
Операнд-3
Операнд-3
Побитовое ИЛИ
Побитовое ИЛИ
|
Операнд-3
Операнд-4
Операнд-4
Побитовое исключающее ИЛИ
Побитовое исключающее ИЛИ
^
Операнд-4
Операнд-5
Операнд-5
Побитовое И
Побитовое И
&
Операнд-5
Операнд-6
Операнд-6
Не равно
Равно
Не равно
!=
Равно
==
Операнд-6
Операнд-7
Операнд-7
Меньше
Меньше или равно
Больше
Больше или равно
Меньше
<
Меньше или равно
<=
Больше
>
Больше или равно
>=

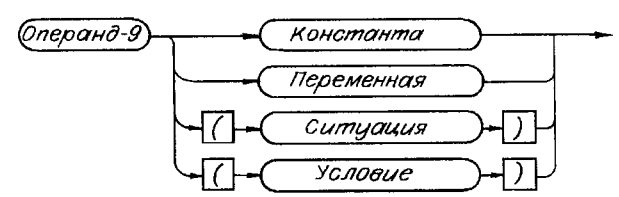

Операнд-7
Операнд-8
Операнд-8
Сдвиг влево
Сдвиг вправо
Сдвиг влево
<<
Сдвиг вправо
>>

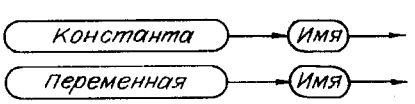

Операнд-6
Слагаемое
Слагаемое
Сложение
Вычитание
Сложение
+
Вычитание
-

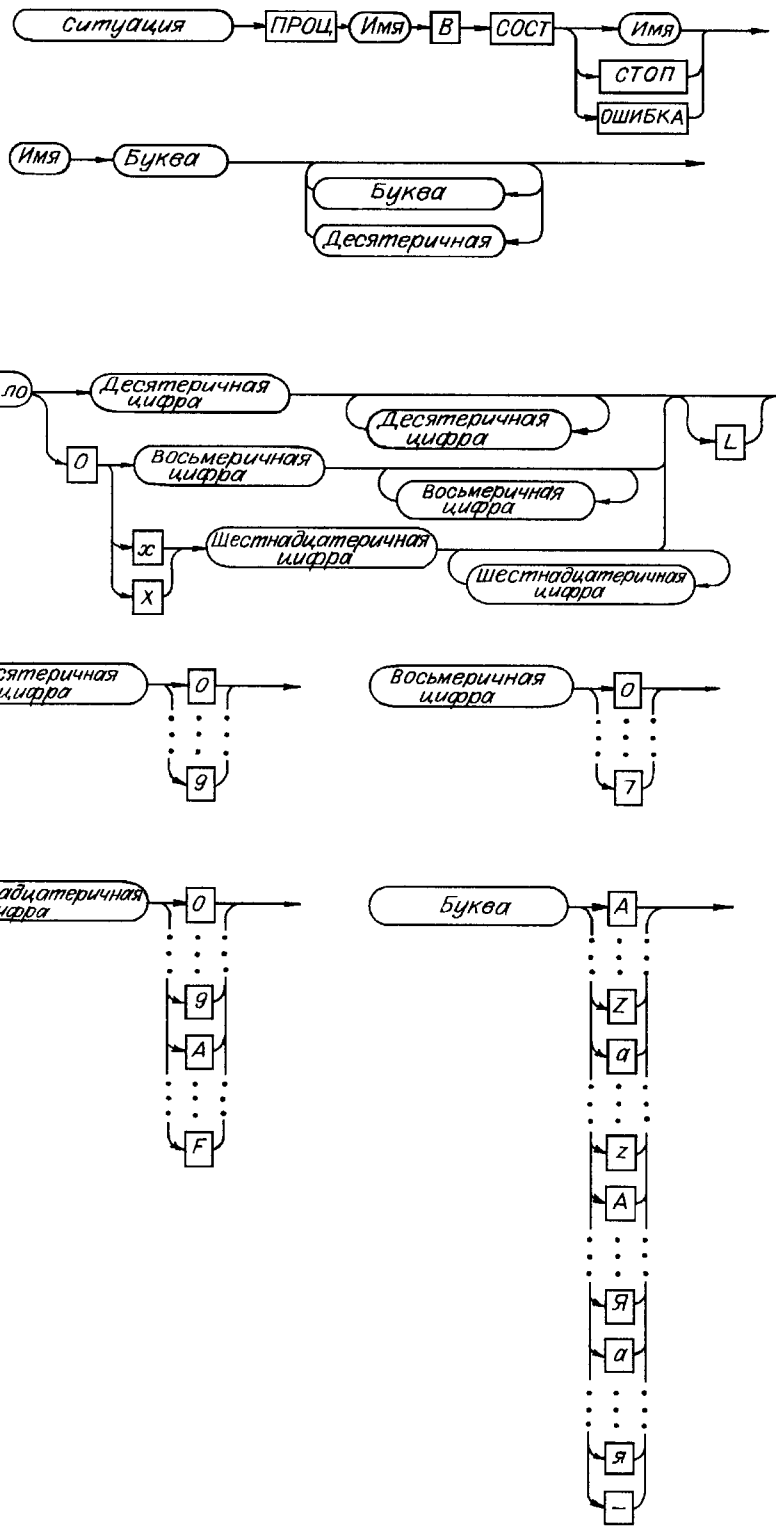

Слагаемое
Множитель
Множитель
Деление по модулю
Умножение
Деление
Деление по модулю
%
Умножение
*
Деление
/


Множитель
Операнд-9
Логическое отрицание
Побитовое отрицание
Инкрементация
Декрементация


Логическое отрицание
!
Побитовое отрицание
~
Инкрементация
++
Декрементация
--


Операнд-9
Константа
Переменная
(
Ситуация
)
(
Условие
)


Константа
Имя
Переменная
Имя


Ситуация
ПРОЦ
Имя
В
СОСТ
Имя
СТОП
ОШИБКА
Имя
Буква
Буква
Десятеричная
Число
Десятеричная цифра
Десятеричная цифра
L
0
Восьмеричная цифра
Восьмеричная цифра
x
X
Шестнадцатеричная цифра
Шестнадцатеричная цифра
Десятеричная цифра
0
9
Восьмеричная цифра
0
7
Шестнадцатеричная цифра
0
9
A
F
Буква
A
Z
a
z
A
Я
a
я
_

## СПИСОК ЛИТЕРАТУРЫ


1. **Шенброт И. М.** Системы программно-логического управления и их применение: Перспективный аналитический обзор. М.: ВИНИТИ, 1986.
2. **Баранов С. И.** Синтез микропрограммных автоматов. М.: Энергия, 1974.
3. **Трахтенброт Б. А., Барздинь Я. М.** Конечные автоматы. Поведение и синтез. М.: Наука, 1970.
4. **Кузнецов О. П., Макаревский А. Я., Марковский А. В., Окуджава В. Ш., Шипилина Л. Б.** ЯРУС — язык описания работы сложных автоматов // Автоматика и телемеханика. 1972.


*Поступила в редакцию 8 июня 1995 г.*