В заключение авторы благодарят А. А. Князева, Ю. А. Антонова за творческое преодоление технологических трудностей, возникших при реализации структур, удовлетворяющих рекомендациям CAMAC.

ЛИТЕРАТУРА

1. В. И. Виноградов. Современное развитие программно-управляемых модульных структур для автоматизации измерений и управления экспериментами.— Автометрия, 1972, № 4.
2. EUR 4100e. CAMAC. A Modular Instrumentation System for Data Handling. ESONE Committee, 1972.
3. EUR 4600e. CAMAC. Organisation of Multi-Grate Systems. ESONE Committee, 1972.
4. В. И. Виноградов, В. И. Петрова, В. Г. Муратов, В. И. Кадашевич. Специализированный процессор системных взаимодействий и обмена данными в структурах, удовлетворяющих рекомендациям CAMAC. Препринт ЛИЯФ, № 18. Л., 1973.

Поступила в редакцию 19 апреля 1973 г.

УДК 681.3.06

С. В. БРЕДИХИН, П. М. ПЕСЛЯК
(Новосибирск)

СРЕДСТВА ПРОГРАММИРОВАНИЯ ДЛЯ CAMAC

0. **Введение.** Цель настоящей статьи — описать наиболее важные направления, характерные для современного состояния развития средств программирования для CAMAC. Вообще говоря, различные направления рассматриваются здесь с сугубо утилитарной точки зрения, независимо от того, насколько они интересны сами по себе. Конструкция CAMAC предполагается известной. Краткое введение к различным разделам имеет целью лишь напомнить определения и условиться относительно терминологии.

В настоящее время существует несколько различных подходов к созданию средств общения высокого уровня с CAMAC. Уже сейчас имеется ряд серьезных работ, которые следует обсудить. Именно это обстоятельство послужило стимулом для написания статьи.

Разумеется, осуществление подобной цели не свободно от недостатков, связанных с новизной проблемы, не говоря уже о тех несовершенствах, в которых прямо виноваты авторы. Так, нам пришлось излагать лишь самые существенные вопросы, повсюду добиваться сжатости, ограничиваясь краткими примерами в одних случаях и вовсе опуская примеры в других. Нам остается лишь надеяться, что изложение не достигло той степени краткости, которая сделала бы статью непонятной. В этой статье всюду словом CAMAC мы пользуемся как именем собственным. Слова «CAMAC система» обозначают некоторую совокупность аппаратных и программных средств, организованную в соответствии с рекомендациями [1, 2]. Слова «CAMAC аппаратура» обозначают аппаратуру CAMAC системы, слова «CAMAC язык» — программные средства доступа к CAMAC аппаратуре.

0.0. Язык CAMAC. CAMAC предлагает пользователю аппаратные средства, в некотором смысле аналогичные программным средствам языка высокого уровня. Иначе, CAMAC аппаратурой (например, модулем) можно пользоваться подобно программам и процедурам: она имеет однозначно определенные связи с другими частями системы и до-

пускает возможность использования без знания деталей ее организации. Однако для практического использования системы «ЭВМ — аппаратура САМАС», как правило, необходимо знать ассемблер управляющей машины, структуру управления САМАС аппаратурой и организацию ее сопряжения с управляющей ЭВМ. Этого, разумеется, нельзя требовать от широкого круга пользователей, поскольку эти вещи считаются достаточно сложными.

Отсюда вытекают основные требования к языку САМАС как к базе общения между человеком, САМАС системой и управляющей ЭВМ. Язык САМАС должен:

использовать все возможности аппаратуры САМАС;

быть достаточно общим, чтобы отражать алгоритмы любой сложности;

обладать логической стройностью и простотой основных понятий, что в значительной мере снижает вероятность появления ошибок;

быть удобным, наглядным и стимулировать пользователей к программированию в четком и недвусмысленном стиле;

опираясь на общность основных понятий, быть машинно независимым.

По нашему мнению, языки с алголоподобной структурой удовлетворяют большинству из этих требований.

0.1. Базовый язык. Современные ЭВМ оснащены широким набором базового обеспечения, ассемблерами, интерпретаторами, компиляторами, которые допускают возможность расширения. Поэтому большинство создателей программного обеспечения для САМАС идут по пути модификаций имеющегося базового обеспечения. Как правило, эти модификации не содержат средств обработки данных, арифметических и логических операций и т. п. Таким образом, язык САМАС выглядит как некоторое расширение базового языка.

1. **САМАС ассемблеры.** Ассемблер — это транслятор, который вырабатывает машинный код из входного языка, подобного по структуре машинному. При этом символические единицы языка ассемблера (входного языка) соответствуют естественным единицам машины. Одним из приемов повышения эффективности программирования на языке ассемблера является создание макрогенераторов (предпроцессоров), которые считывают текст программы, находят в нем макроопределения и подставляют в месте их вызова заранее определенные последовательности инструкций языка ассемблера. Затем полученный текст подвергается обычному процессу ассемблирования. Под словами «САМАС ассемблер» следует понимать ассемблер с макрогенератором (макроассемблер), входной язык которого расширен набором макроопределений, описывающих особенности управления аппаратурой САМАС.

В отличие от стандартных внешних устройств специфика обмена данными между ЭВМ с малой разрядной сеткой (12/16 разрядов) и аппаратурой САМАС состоит в том, что для организации обмена с САМАС необходимо передавать соответствующий управляющий код, содержащий информацию о *C*, *N*, *A*, *F*. Здесь основная проблема — сложность адресной структуры САМАС. Когда в обмене занято несколько модулей САМАС, становится сложной проблема модификации их адресов. Формально преодолеть эти трудности помогает макроассемблер, с соответствующей библиотекой макроопределений.

1.0. Пример подобного макроассемблера приведен в [3], где описана модификация ассемблера DAP16 для «Honeywell-315 (316)», которая получила название *CAMACRO*. Модификация состоит в добавлении к ассемблеру макрогенератора.

Параметрами макроопределений служат *C*, *N*, *A*, *F*, которые должны быть упакованы в одно слово. В *CAMACRO* это достигается путем определения составных параметров. Поясним на примере.

Пример: пусть CAMAC функция F (9) имеет макроопределение §F09 и три параметра C, N и A. Программа, которая загружает в регистр ЭВМ код C, N, A, F(9), выполненная на *CAMACRO*, выглядит так:

(а) § F09 вызов макроопределения

(б) #1=[1, 2, §1] [3, 7, §2] [8, 11, §3] [12, 16, 9]§
определение параметра *CNAF*

(в) *LDA*=#1 загрузка *CNAF*

Параметр #1 конструируется во время вызова макроопределения в виде 16-рязрядной константы следующим образом:

разряды 1—2 из параметра 1(C)

разряды 3—7 из параметра 2(N)

разряды 8—11 из параметра 3(A)

в разрядах 12—16 всегда записана константа 9.

1.1. *CAMPACK* [4] представляет собой набор подпрограмм ввода — вывода, разработанный для PDP11 и интерфейса ICP11. *CAMPACK* и ассемблер PAL11 образуют CAMAC ассемблер. Все 52 подпрограммы выполнены в виде макроопределений с соответствующими параметрами, которые могут быть переменными. Это позволяет модифицировать адреса аппаратуры CAMAC и вычислять их в процессе исполнения программы. Существенно, что эти макроопределения можно использовать как подпрограммы языка ФОРТРАН.

2. **CAMAC интерпретаторы.** Интерпретатор — это транслятор, который во время трансляции каждого оператора входного языка выполняет действия, диктуемые этим оператором. Словами «CAMAC интерпретатор» мы будем обозначать такие интерпретаторы, возможности которых расширены с целью управления CAMAC аппаратурой. Поскольку ниже речь будет идти только о CAMAC интерпретаторах, слово CAMAC мы будем опускать.

2.0. Интерпретатор *CAMCONV* [5] предназначен для целей тестирования и наладки CAMAC аппаратуры. Он реализован на PDP11; управление аппаратурой осуществляется через интерфейс ICP11.

Операторы управления задаются в виде $C_xN_yA_zF_t$, где x, y, z, t могут быть только целыми числами. Остальные операторы имеют, как правило, однобуквенную мнемонику. В языке имеются инструкции для перевода исполняемой программы в цикл ожидания как безусловный, так и зависящий от Q. Арифметические операции отсутствуют. Связь оператора с системой осуществляется путем печати соответствующих сообщений на телетайпе, например, при появлении запроса *LAM* или при обнаружении ошибки. Используя телетайп, оператор может прервать исполнение программы для внесения изменений.

Приведем пример программы, выполненной на *CAMCONV*. Пусть требуется прочитать и напечатать в двоичном виде содержимое регистра, расположенного в крейте № 1, модуле № 3, по субадресу 0. Соответствующая программа такова:

— *C1N3A0F0*	чтение регистра
— *PB*	печать содержимого
— *E*	запуск программы

Вопросам организации простых CAMAC интерпретаторов посвящена также [6].

2.1. В статьях [7] и [8] содержатся описание расширения интерпретатора *FOCAL* для PDP8 и эскизы его реализации. Управление аппаратурой CAMAC производится через специальный крейт-контроллер, который помимо *CNAF* нуждается в дополнительной информации о типе команды, названной крейт-функцией, и, в случае передачи данных, о типе данных.

В [7] сообщается об интерпретаторе *FOCADAT*, который может генерировать CAMAC команды и осуществлять передачу данных без прерывания от аппаратуры CAMAC. Интерпретатор *FOCALIT*, о кото-

ром говорится в [8], в дополнение к возможностям *FOCADAT* обслуживает до 72-х различных прерываний от CAMAC аппаратуры. Строка программы, относящаяся к CAMAC операциям, снабжается номером, звездочкой и имеет вид:

CF, C, N, A, F, FB, HW, (LW), (Q)

Параметр *CF* определяет тип команды.

Общие возможности *FOCADAT* и *FOCALIT* таковы:

(а) генерация CAMAC команд без манипуляций с данными;

* 0, *C, N, A, F, (Q)*

(здесь и далее *C, N, A, F, Q* несут обычный смысл)

(б) специальные CAMAC команды:

* 1, *C* сброс
* 2, *C* пуск
* 3; *C* установка запрета
* 4, *C* снятие запрета
* 5, *C, L, R* чтение *LAM*

(в) CAMAC команды, оперирующие с данными:

* 6, *C, N, A, F, FB, HW, (LW), (Q)*

чтение данных

* 7, *C, N, A, F, FB, HW, (LW), (Q)*

запись данных

(здесь *FB* — формат данных, *HW* — 12 старших разрядов, *LW* — 12 младших разрядов CAMAC данных).

В дополнение к этому *FOCALIT* имеем две возможности:

* 9, *C* выключить прерывание
* 10, *C* включить прерывание

Оба интерпретатора применяются для целей тестирования и управления CAMAC аппаратурой, например для управления приводом шаговых двигателей.

2.2. В статье [9] развиваются идеи [7] применительно к двум версиям интерпретатора *BASIC* на PDP11. Так же, как и в [7], обе модификации работают со специальным крейт-контроллером, о котором мы упоминали в п. 2.1.

Первая модификация используется только для управления одним крейтом. Программирование CAMAC команд производится посредством функции расширения *EXF*, которая имеется во входном языке. (Отметим, что CAMAC интерпретатор, описанный в [10], также использует *EXF*-функцию.) В качестве параметров ей задаются *N, A, F* и данные *D*. Строка, содержащая CAMAC команду, выглядит так:

⟨*номер строки*⟩ *LETU=EXF (M, N, A, F, D)*.

Определено пять типов функции *EXF*, которые различаются по значению первого параметра *M:*

M=0 — команда модулю
M=1 — сброс
M=2 — пуск
M=3 — установка запрета
M=4 — снятие запрета

Параметры могут задаваться в виде формул, что позволяет модифицировать адреса в процессе исполнения.

Пример: ниже приведена программа для записи числа 1000 по субадресу 0 в станции *N*=3, 4, ..., 10 с помощью команды *F* (16).

```
100 FOR N=3T010
110 LET U=EXT (0, N, 0, 16, 1000)
120 NEXI N
```

Вторая модификация может обслуживать до восьми пользователей одновременно. Суть второй модификации заключается во введении в язык *BASIC* следующих CAMAC команд (табл. 1).

Таблица 1

Имя CAMAC команды	Параметры	Назначение
CZ	*C*	Пуск
CC	*C*	Сброс
CIZ	*C*	Установка запрета
CIR	*C*	Снятие запрета
CF	*F, C, N, A*	Команды, не манипулирующие с данными
CR	*F, C, N, A*	Чтение данных
CW	*F, C, N, A, D*	Запись данных

Пример: строка программы для записи ранее вычисленного значения *U* по субадресу 0, модуля № 1, крейта № 2, с помощью команды *F* (17) выглядит так:

300 *CW* 17, 2, 1, 0, *V*

2.3. С помощью CAMAC интерпретаторов просто разрабатывать и отлаживать программы. Однако их нельзя рекомендовать как универсальное средство, поскольку интерпретаторы непригодны для программирования установок, обслуживающих критичные к времени эксперименты, и, как правило, их базовые языки недостаточно гибки для манипуляций с кодами в памяти машины.

3. **Библиотеки CAMAC процедур.** Термин «библиотека процедур (или подпрограмм)» хорошо известен и интуитивно понятен. Основным направлением работ в этой области являются расширение состава библиотечных процедур и усовершенствование методов доступа к ним. В настоящее время большинство программ, входящих в библиотеки, составляется на языках высокого уровня. Это позволяет использовать пакеты библиотечных программ на машинах разного типа.

3.1. Вопросам установления связи с CAMAC аппаратурой при помощи библиотеки специальных процедур, выполненных на языке высокого уровня, посвящена работа [11]. Все процедуры набора (кроме одной — базовой) написаны на языке ФОРТРАН и вызываются с помощью оператора *CALL*. Базовая процедура — *CMCBSC* управляет системой ввода — вывода конкретной ЭВМ и устройством сопряжения машина — CAMAC. Она выполняется на ассемблере этой машины или на другом языке, пригодном для этих целей. Для выполнения конкретных операций ввода — вывода, связанных с управлением аппаратурой CAMAC, остальные процедуры библиотеки обращаются к *CMCBSC*. Параметры процедур задаются в виде простых переменных и могут вычисляться в процессе исполнения.

4. **CAMAC компиляторы.** Компилятор — это транслятор с проблемно-ориентированного языка высокого уровня. В соответствии с этим словами «CAMAC компилятор» будем обозначать компилятор с языка высокого уровня, ориентированного на управление CAMAC аппаратурой (всюду далее CAMAC языка).

CAMAC языкам посвящена значительная часть настоящего обзора. Это сделано умышленно с целью привлечь читателей к размышлениям, а возможно и к работе над созданием языков CAMAC и реализации компиляторов с этих языков.

4.0. Краткое описание одного из CAMAC языков приведено в [12]. Любая программа, выполненная на этом языке, состоит из двух секций: секции описания аппаратуры, секции операторов управления.

Каждая строка программы из секций описания или управления должна начинаться знаком (/) — начало строки и заканчиваться знаком (;) — конец строки. В операторах управления допускается обращение к аппаратуре только по имени, присвоенному этой аппаратуре.

4.0.0. Для явного определения имен имеются следующие формы описания:

(а) описание ветви

/⟨*имя ветви*⟩=⟨*целое число*⟩;

(например: /*BR*1=3; означает, что ветвь № 3 имеет имя *BR*1)

(б) описание крейта

/⟨*имя крейта*⟩=⟨*имя ветви*⟩/⟨*номер крейта*⟩;

(например: присвоение имени *CRAT*1 крейту № 5, ветви № 3 выглядит так: /*CRAT*1=*BR*1/5;)

(в) в языке допускается описание как отдельного регистра в модуле, так и массива регистров, расположенных в одном или нескольких модулях. Общая форма описания одного регистра такова:

/⟨*имя регистра*⟩ (⟨*имя крейта*⟩, *N*, *SA*);

где *N* и *SA* — номер модуля и субадрес соответственно

(например: /*SCAL*(*CRAT*1, 5, 3); означает, что имя *SCAL* присвоено регистру, расположенному по субадресу № 3, в модуле № 5, крейте № 5, ветви № 3).

Описание массива регистров в общем случае выглядит так:

/⟨*имя массива*⟩ (⟨*имя крейта*⟩, *Nb*:*Ne*:*I*, *SAb*:*SAe*:*J*); где *Nb* и *Ne* — наименьший и наибольший номера модулей в крейте, регистры которых определены в массиве; *SAb* и *SAe* — наименьший и наибольший субадреса регистров; *I* и *J* — приращения, используемые для определения наборов *N* и *SA*. Например, если использовать приведенное выше определение ветви и крейта, то

/*SCALARRAY*(*CRAT*1, 5 : 9 : 2,0 : 3 : 1);

означает, что массиву регистров, расположенному в модуле №5 (субадреса 0, 1, 2, 3), модуле №7 (субадреса 0, 1, 2, 3), модуле №9 (субадреса 0, 1, 2, 3), крейта № 5, ветви № 3 присвоено имя *SCALARAY* и он может рассматриваться при операциях CAMAC как единое целое.

4.0.1. Каждой CAMAC функции *F*(*n*) в данном языке присвоено имя. Например, *F*(0) имеет имя *LECGB*1, *F*(17) — *TESAPB*, *F*(1) — *RAZ*1, *F*(16) — *ECRSNR*. Эти функции являются основными операторами языка. В языке допускается 4 типа конструкций операторов управления:

(а) операторы чтения и записи данных

/⟨CAMAC *функция*⟩⟨CAMAC *адрес*⟩⟨*адрес памяти ЭВМ*⟩;

(например, строка /*LECGB SCAL BETA*; означает, что с помощью команды *F*(0) считывается содержимое регистра *SCAL* и заносится в память ЭВМ по адресу *BETA*)

(б) операторы условных переходов

/*если* ⟨CAMAC *функция*⟩⟨CAMAC *адрес*⟩=0 *переход* ⟨*метка*⟩;

(например: строка /*IF TESAPB BR*1=0 *GO TO L*1; означает, что если статус ветви № 3 равен нулю, то перейти к выполнению оператора с меткой *L*1)

(в) операторы, изменяющие состояние аппаратуры

/⟨CAMAC *функция*⟩⟨CAMAC *адрес*⟩;

(например: если принять во внимание данное выше определение массива *SCALARRAY*, то строка /*RAZ*1 *SCALARRAY*(5); предписывает сбросить содержимое регистра *B*(3)*C*(5)*N*(7)*A*(0)

(г) операторы, выполняющие параллельные действия

/⟨CAMAC *функция*⟩⟨CAMAC *адрес*⟩... ⟨CAMAC *адрес*⟩;

(например: установка начальных значений в регистры $B(3)C(5)N(5)A(0)$ и $B(3)C(5)N(7)A(0)$ массива *SCALARRAY* с помощью команды $F(16)$ выглядит так:

/ECRSNR SCALARRAY(0), *SCALARRAY*(4);).

4.1. Фундаментальные работы по созданию САМАС языка высокого уровня проводятся рабочей группой по математическому обеспечению комитета ESONE. К этой группе относится автор работы [13], который агитирует за стандартный язык САМАС. В качестве основных требований к языку он выдвигает требование инвариантности языка к аппаратуре САМАС и ЭВМ, а также требование простоты понимания программы, записанной на этом языке. В данном пункте кратко излагается проект САМАС языка, разработанный этой рабочей группой.

Приведенное ниже описание указывает, какие возможности имеются в языке, и содержит формат основных предложений. Формальное описание языка и подробные примеры приведены в [14].

Программа, управляющая САМАС аппаратурой и занимающаяся обработкой данных, разбита на сегменты, выполненные на САМАС языке, и на сегменты, выполненные на базовом языке.

Сегмент программы, выполненный на САМАС языке, состоит из двух различных классов предложений: описаний и операторов. Существенная идея совершенствования описания состоит в иерархическом развитии имен. С этой целью для описания мелких элементов аппаратуры типа регистра или конкретного бита в регистре используются вспомогательные имена, присвоенные крейту или ветви. Операторы вызывают исполнение команд, относящихся к определенным в описании ресурсам, и ссылаются на них только по именам. Это делает программу в значительной степени инвариантной по отношению к оборудованию системы.

4.1.0. Директивы транслятора. Чтобы выделить сегменты программы, написанные на языке САМАС, от сегментов, выполненных на базовом языке, и определить некоторые элементы самого САМАС языка, в него введены управляющие директивы.

Приведем перечень основных директив:

CAMACSEGMENT	означает начало сегмента САМАС в общей программе;
ENDSTATEMENT	используется для завершения каждого предложения САМАС языка;
BEGINCAMAC	указывает на начало последовательности САМАС предложений;
ENDCAMAC	указывает на конец последовательности САМАС предложений;
CAMACSEGMENTEND	указывает на конец САМАС сегмента;
NOTE	указывает на начало поля комментариев в САМАС сегменте;
SEQV, CNAME, CDMD, CDCL, CREF	— эти директивы указывают на начало последовательности секций описания в САМАС сегменте;
CAST	указывает на конец последовательности секций описания. После этой директивы следуют исполняемые операторы САМАС языка.

4.1.1. Структура программы. Предложения САМАС языка в программе организованы в сегменты. Сегмент ограничен директивами *CAMACSEGMENT* и *CAMACSEGMENTEND*. САМАС сегмент может быть написан исключительно на языке САМАС или с использованием директив *ENDCAMAC*, *BEGINCAMAC* может содержать предложения

базового языка. Лексическая структура сегмента, выполненная на чистом CAMAC языке, проиллюстрирована на следующем примере:

⟨*предложения на базовом языке*⟩
CAMACSEGMENT ⟨*имя сегмента*⟩

SEQV
CNAME
CDMD
CDCL
CREF
CACT
} *секции описания*

⟨*операторы CAMAC языка*⟩
CAMACSEGMENTEND
⟨*предложения на базовом языке*⟩

4.1.2. Описания. Имеется 5 типов описательных предложений, каждое из которых появляется в определенной секции.

4.1.2.0. Предложения эквивалентности. Секция эквивалентности начинается директивой *SEQV* и содержит предложения, в которых определяются или переопределяются различные идентификаторы. Имеется два вида предложений эквивалентности.

Первый вид — это предложение *EQV*, позволяющее программисту переименовывать некоторые основные единицы языка. Например, утомительно ставить в конце каждого предложения директиву *ENDSTATEMENT*.

Обычно директиве *ENDSTATEMENT* при помощи *EQV* устанавливают в соответствии эквивалентный ей символ, например точку.

Второй вид — это предложение типа равенства (=). Такие предложения присваивают численные величины символьным идентификаторам.

Пример:

CEQV ENDSTATEMENT
.EQV ENDSTATEMENT ENDSTATEMENT
EQV NOTE.
$M=10$ \ *число модулей.*
$W=2$ \ *шаг.*
$F=W$ \ *начальный адрес.*
$L=(M-1).W+F$ \ *конечный адрес.*

4.1.2.1. Предложения, именующие CAMAC аппаратуру. Секция присвоения имен CAMAC аппаратуре начинается с директивы *CNAME* и содержит описательные предложения, присваивающие имена ветвям, крейтам, модулям и регистрам. Основными единицами аппаратуры CAMAC, которые нужно именовать, являются регистры. Правила образования именующих предложений позволяют ясно именовать единицы аппаратуры или именовать их согласно иерархии: сначала давать имена большим единицам, т. е. ветвям, крейтам, а затем использовать их для определения имен более мелких единиц, таких как модули и регистры.

Например:

CNAME. APPARATUS $=B(1)$.
CONTROLCRATE $=APPARATUS\ C(1)$.
Z $=CONTROLCRATE$.
COUNTER $=Z\ N(4)A(0)$.

Единицы CAMAC аппаратуры можно также формировать в массивы. Поясним это на примере:

CNAME.
DATACRATEX $=APPARATUS\ C(2)$.
DX $=DATACRATEX$.
MEANTEMPS$(1:M)$ $=DXN(F:L:W)A(0)$.

Массив *MEANTEMPS* содержит 10 регистров крейта 2, ветви № 1, размещенных в модулях, начиная с $N=2$ до $N=20$ с шагом 2. Все регистры массива имеют одинаковые субадреса $A(0)$.

4.1.2.2. Предложения, именующие запросы аппаратуры CAMAC. Запросы или сигналы *LAM* соответствуют не только ветви, крейту или модулю. В пределах модуля запрос может соответствовать субадресу или даже отдельному биту в регистре. Все эти единицы могут получить CAMAC имена. Таким образом, на всех уровнях запросы могут быть ассоциированы с CAMAC именами, которые должны быть определены в *CNAME*-секции.

4.1.2.3. Предложения, именующие участки памяти вычислительной машины. Эта секция имеет имя *CDCL*. В ней определяются два вида имен: одни из них указывают на буфера для обмена данными, с помощью других определяются области для хранения адресов аппаратуры CAMAC.

4.1.2.4. Предложения, именующие глобальные переменные. В данном языке предполагается, что имена переменных локальны по отношению к сегменту в том случае, если они специально не выделены для общего пользования. Описание имен глобальных переменных осуществляется в секции *CREF*. Благодаря этим описаниям на них можно ссылаться и в других сегментах.

4.1.3. Операторы. Операторы — это представленные в мнемонической форме функции, которые вызывают активность CAMAC системы. В каждом сегменте программы операторы размещены в секции с именем *CACT*. Всем операторам можно присваивать метки для дальнейших ссылок.

4.1.3.0. Оператор повторения. За операторами передачи данных (п. 4.1.3.1.) и управления (п. 4.1.3.2.) может следовать оператор *REPEAT*, указывающий на повторение предшествующего оператора В качестве его параметра может быть переменная или константа, указывающая число повторений.

4.1.3.1. Операторы передачи данных. Операторы этой группы оперируют с двумя адресами данных, один из которых является адресом аппаратуры *CAMAC*, а другой может быть либо *CAMAC* адресом, либо адресом памяти ЭВМ.

Операторы передачи данных могут иметь вид $F(n)$. Они могут быть также представлены с помощью стандартной мнемоники (табл. 2). В этой таблице и далее A означает субадрес регистра; $G1$, $G2$ — номер группы регистра.

Пример: строка

GETDATA READ MEANTEMPS BUFFER.

означает чтение данных массива *MEANTEMPS* и занесение содержимого в буфер ЭВМ. Здесь *GETDATA* — метка оператора.

4.1.3.2. Операторы управления. Параметрами операторов управления могут быть только адреса аппаратуры CAMAC. Перечень этих операторов приведен в табл. 2. Здесь B — адрес ветви, C — адрес крейта, I — позиция бита в регистре.

Пример: строка

CLEAR SCALERARRAY.

означает сброс регистров массива.

4.1.3.3. Операторы перехода. Операторы перехода служат для выполнения безусловных переходов и переходов, зависящих от ответа по Q-шине.

Безусловный переход определяется указателем *GOTO* и меткой оператора, которому следует передать управление.

В конструкциях оператора условного перехода, кроме указателя *GOTO* и метки, используются условия перехода *IF* или *IFNOT*. Ответ по Q-шине опрашивается с помощью операторов *LAM* в случае испол-

Таблица 2

ОПЕРАТОРЫ ПЕРЕДАЧИ ДАННЫХ

Мнемоника	CAMAC адрес	Группа	CAMAC функция
READ или *MOVE*	*A*	*G*1	*F*(0)
	A	*G*2	*F*(1)
READCLR	*A*	*G*1	*F*(2)
READCOMP	*A*	*G*1	*F*(3)
READLAM	*A*	—	*F*(8)
READSTAT	*A*	—	*F*(27)
WRITE или *MOVE*	*A*	*G*1	*F*(16)
	A	*G*2	*F*(17)
CLEARSEL	*A*	*G*1	*F*(21)
	A	*G*2	*F*(23)
SETSEL	*A*	*G*1	*F*(18)
	A	*G*2	*F*(19)
TRANSFER или *MOVE*	*A*	*G*1	*F*(0)
	A	*G*2	*F*(1)
	A	*G*1	*F*(16)
	A	*G*2	*F*(17)

нения $F(8)$, оператора *STATUS* при исполнении $F(27)$ и просто Q в случае исполнения CAMAC функции, отличной от $F(8)$ и $F(27)$. Спецификация операторов *LAM* и *STATUS* приведена в табл. 4.

Пример: строка

IF LAM CONTROL GOTO GETDATA.

предписывает выполнить проверку запроса *CONTROL* и, если запрос установлен, передать управление оператору с меткой *GETDATA*.

5. **Заключение.** В заключении мы хотим обратить внимание читателей на два вопроса и сделать замечание по поводу литературы.

5.0. Язык CAMAC, разработанный в качестве дополнения к базовому языку, весьма зависим от этого базового языка. Большинство авторов молчаливо предполагают гармоничное слияние специфического CAMAC языка с базовым. Так ли это на самом деле? В настоящее время нет достаточного опыта по работе с системой CAMAC, а тем более по ее программному обеспечению, чтобы утвердительно или отрицательно ответить на этот вопрос. Так или иначе сегодня живет тезис: «...расширение существующей системы программирования помогает избежать работы по созданию полностью новой системы».

5.1. Следует отметить, что разработка средств программирования высокого уровня для CAMAC — это не только вопрос создания языка и реализации транслятора, но также и вопрос создания операционной системы, поскольку CAMAC программа в основном имеет дело с каналами ввода — вывода управляющей машины. В настоящее время этот вопрос наименее разработан.

Таблица 3

ОПЕРАТОРЫ УПРАВЛЕНИЯ			
Мнемоника	CAMAC адрес	Группа	CAMAC функция
CLEAR	*A*	*G*1	*F*(9)
	A	*G*2	*F*(11)
	I	*G*1	*F*(21)
	I	*G*2	*F*(23)
GLEARLAM	*A*	*G*1	*F*(10)
	I	*G*2	*F*(23) по *A*(12)
ENABLE или *SET*	*A*	—	*F*(26)
	I	*G*1	*F*(18)
	I	*G*2	*F*(19)
DISABLE	*A*	—	*F*(24)
	I	*G*1	*F*(21)
	I	*G*2	*F*(23)
EXECUTE	*A*	—	*F*(25)
INITIALIZE	*B*		Генерирует *BZ*
	C		*F*(26) в *N*(28)*A*(8)
SETINHIBIT	*C*		*F*(26) в *N*(30)*A*(8)
CLEARINHIBIT	*C*		*F*(24) в *N*(30)*A*(9)

Таблица 4

ОПЕРАТОРЫ УСЛОВНОГО ПЕРЕХОДА			
Мнемоника	CAMAC адрес	Группа	CAMAC функция
LAM	*A*	—	*F*(8)
	I	*G*2	*F*(1) по *A*(14)
STATUS	*A*	—	*F*(27)
	I	*G*2	*F*(1) по *A*(14)
	I	*G*1	*F*(0)

5.2. Данный обзор не претендует на полноту и детальность изложения. В нем мы стремились охарактеризовать только основные направления работ по программированию для CAMAC систем. Скромная библиография вызвана теми же причинами. Кроме литературы, на которую мы ссылались в обзоре, на формирование наших взглядов существенно повлияла [15]. В ней представлена картина развития серии языков возрастающей сложности, изложены идеи по технологии их реализации. Авторы отечественного обзора по входным языкам про-

граммирования для CAMAC [16] придерживаются взглядов [15], эта работа содержит ряд конкретных примеров.

Нам приятно поблагодарить А. Н. Гинзбурга, который стимулировал эту работу, Ю. К. Постоенко за ценные обсуждения и замечания, а также Н. Г. Щербакову, которая выполнила значительную часть переводов иностранной литературы,

ЛИТЕРАТУРА

1. EUR 4100. Revised Description and Specification. ESOME Committee, 1972.
2. EUR 4600. Specification of the Branch Highway and CAMAC Grate Controller Type A. ESONE Committee, 1972.
3. F. R. Golding, A. C. Peatfield, K. Spurling. CAMACRO-an Aid to CAMAC Interface Programming.— CAMAC Bulletin, 1972, N 5.
4. CAMAC package: CAMPACK. STAP-Schumberger, October, 1972.
5. CAMCONV. Conversational Language for Writing and Executing CAMAC Functions. STAP-Schumberger, October, 1972.
6. R. M. Keyser. COMP11, a CAMAC-Oriented Monitor for the PDP11.— CAMAC Bulletin, 1973, N 7.
7. F. May, H. Halling, K. Petreczek. FOCAL Overlay for CAMAC Data and Command Handling.— CAMAC Bulletin, 1971, N 1.
8. F. May, W. Marschik, H. Halling. A. FOCAL Interrupt Handler for CAMAC.— CAMAC Bulletin, 1973, N 6.
9. H. Halling, K. Zwoll, W. John. CAMAC Averlay for Single-User BASIC and Modification of 8-user BASIC for the PDP11. CAMAC Bulletin, 1973, N 6.
10. I. Bals, E. Agostino. An Extendes BASIC Language for CAMAC Programming.— CAMAC Bulletin, 1973, N 7.
11. R. F. Thomas, Specifications for Standard CAMAC Subroutines.— CAMAC Bulletin, 1973, N 6.
12. M. Sarquiz, P. Valois. An Approach to a CAMAC Language.— CAMAC Bulletin, 1971, N 2.
13. P. Wilde. A CAMAC Language.— CAMAC Bulletin, 1972, N 3.
14. Proposal for a CAMAC Language.— CAMAC Bulletin (Supplement), 1972, N 5.
15. P. Wilde. A Survey Paper.— ESONE-Software Working Group, RHEL, Chilton, 14 November, 1970.
16. Ю. Ф. Рябов. В. П. Хомутников. О входных языках для систем, использующих CAMAC. Препринт ЛИЯФ, № 24, Л., 1973.

Поступила в редакцию 19 февраля 1974 г.

УДК 681.3.06 : 51

П. М. ПЕСЛЯК, Э. А. ТАЛНЫКИН

(Новосибирск)

ЯЗЫК СИСТЕМНОГО ПРОГРАММИРОВАНИЯ ДЛЯ МИНИ-ЭВМ

Введение. Авторы этого сообщения на протяжении трех лет занимались вопросами построения математического обеспечения ЭВМ различных классов. В круг задач входило: обеспечение нестандартных внешних устройств (графопостроителей, дисплеев, аналого-цифровых и цифроаналоговых преобразователей, устройств считывания полутоновых изображений и т. п.), а также систем для сбора и обработки данных в реальном масштабе времени.

Основная часть всех этих работ проводилась с использованием языков ассемблера, хотя неадекватность этого инструмента целям разарботки достаточно очевидна. Использование макрогенераторов дает лишь некоторое механическое облегчение, но не освобождает от необходимости помнить о машинных кодах.