АКАДЕМИЯ НАУК СССР
СИБИРСКОЕ ОТДЕЛЕНИЕ
А В Т О М Е Т Р И Я

№ 5 1984

ПРОГРАММНЫЕ СРЕДСТВА АВТОМАТИЗАЦИИ

УДК 681.3.06

А. В. ИОФФЕ, Э. А. ТАЛНЫКИН
(Новосибирск)

BUSIC — ДИАЛОГОВАЯ СИСТЕМА С ЯЗЫКОВЫМ ПРОЦЕССОРОМ ДЛЯ НАЛАДКИ И ТЕСТИРОВАНИЯ ОБОРУДОВАНИЯ

Введение. В процессе настройки оборудования, работающего на линии с ЭВМ и не имеющего автономного управления, инженеру необходимы детальный оперативный доступ к элементам аппаратуры, «видимым» через сопряжение, и возможность строить на основе таких действий более сложные управляющие программы. Другими словами, реализация настроечных «сценариев» требует программирования. При работе в рамках операционной системы для этой цели необходима разработка адекватного драйвера, обеспечивающего полную функциональную «видимость» оборудования, после чего любую управляющую программу можно реализовать, используя подходящий для этого язык программирования. Такую реализацию нельзя назвать оперативной, поскольку здесь присутствуют, как минимум, процессы трансляции и редактирования связей. От инженера подобный подход требует достаточно серьезного уровня программистской подготовки.

В настоящей статье представляется диалоговая система программирования (BUSIC), обеспечивающая возможность работы с оборудованием в пошаговом режиме и по сколь угодно сложной программе. Внешне система напоминает классические интерактивные языковые системы для инженерных расчетов, такие как BASIC и FOCAL; ее функциональное наполнение — это унифицированные средства работы с аппаратурой и простейшие программные конструкции общего назначения. Подобные идеи использовались ранее в [1, 2] для обеспечения настройки и тестирования модулей КАМАК.

Рассматриваемая система ориентирована на архитектуру типа общей шины, обеспечивающую естественную стандартизацию доступа к оборудованию через механизм адресации, при котором регистры устройства адресуются как ячейки памяти. Специализированной остается интерпретация значений, записываемых или читаемых по адресам, выделенным для внешних устройств. Это позволяет использовать для работы с внешними регистрами обычные языковые переменные, не вводя специальных понятий.

Одно из основных требований к системе — возможность реализации режимов «осциллографирования» в реальном времени. Другими словами, система должна обеспечивать частоту управляющих воздействий на аппаратуру, определяемую техническими характеристиками ЭВМ, без внесения дополнительных задержек. Для выполнения данного требования в системе BUSIC принят режим исполнения программ с предварительной компиляцией непосредственно в машинные команды. Это делает реализацию несколько более трудоемкой по сравнению с традицион-

ным в таких системах исполнением путем интерпретации. Для пользователя это различие остается невидимым.

Наличие компилятора и развитые средства программирования позволяют разрабатывать в системе BUSIC довольно серьезные программы, работающие на этапе эксплуатации устройства. Например, тестовое программное обеспечение может реализовываться путем систематизации микротестов, полученных в результате настройки.

Система BUSIC создавалась как основное средство для наладки и тестирования комплекса оборудования синтезирующей системы визуализации [3]. За сравнительно короткий срок (немногим более года) была настроена аппаратура объемом 17000 корпусов интегральных микросхем.

Общие принципы. Основные понятия. Система работает под управлением команд, вводимых с терминала. Команда есть строка текста, завершаемая возвратом каретки. Входные строки могут начинаться номером (десятичное число в интервале 1..65535). Строки без номеров сразу после ввода анализируются и исполняются. Такой режим работы называется пошаговым или калькуляторным. Строки с номерами запоминаются без какой-либо обработки и составляют собственно программу. Независимо от последовательности ввода строки программы выстраиваются в порядке возрастания номеров, и если введена строка с уже имеющимся в программе номером, то новая строка замещает старую.

Существует набор команд манипуляции текстом программы, включающий команды редактирования, распечатки, сохранения в файле и т. д.

Если в процессе работы ввести строку вида «# имя файла», то последующие строки входного потока будут вводиться из указанного файла до его исчерпания. Интерпретация входных строк не зависит от источника, так что в файле может быть записана программа, любая последовательность команд или данных. Такие файлы называются косвенными командными файлами. В тексте подобного файла может быть вызван другой косвенный файл и т. д. После исчерпания вызванного файла происходит чтение вызвавшего. По желанию пользователя строки, вводимые из файла, могут печататься на терминале.

Запуск программы производится командой RUN. Перед запуском программа компилируется в машинные команды, если это необходимо (т. е. после последней компиляции были изменения в тексте), а затем выполняется передача управления на первую команду скомпилированной программы. Операторы, исполняемые в калькуляторном режиме, также предварительно компилируются. Им условно приписывается номер 0 и позиция за последним оператором программы. Если в памяти имеется программа, то для работы в калькуляторном режиме необходимо, чтобы программа не содержала синтаксических ошибок.

Прежде чем приступить к изложению языка системы BUSIC, следует остановиться на организации аппаратных средств комплекса, на который она ориентирована. При работе системы старшие 4 К виртуального адресного пространства отображены на страницу ввода-вывода общей шины, что позволяет иметь доступ к любым регистрам подключенных к ней устройств. Для этого необходимо описать переменную и дать ей принудительное размещение по соответствующему адресу (160000..177776). При дальнейших манипуляциях такая переменная ничем не отличается от обычных.

Для управления специализированным оборудованием используется отдельная тестовая шина, которая служит для тестирования, автономного управления и «холодной загрузки» комплекса. Тестовая шина имеет 24 разряда адреса и 16 разрядов данных. Всякий функционально замкнутый блок комплекса обладает выходом на тестовую шину, при этом число регистров и их назначение определяются разработчиком исходя из соображений полноты и удобства реализации всех необходимых

режимов управления. Например, если некоторый блок системы имеет внутреннюю память, то она адресуется целым сегментом адресов тестовой шины.

Тестовая шина подключается к общей шине через специальный адаптер, который имеет возможность отображения свободных сегментов общей шины на тестовую шину (режим окна), что позволяет адресоваться к последней непосредственно с помощью машинных команд. Компилятор системы BUSIC использует это для оптимизации доступа к регистрам аппаратуры.

Языковые возможности. Программа в системе BUSIC представляется последовательностью текстовых строк. Строка состоит из номера (метки) и предложения. Формат языка системы BUSIC не свободен, и некоторые конструкции должны целиком размещаться в строке. Лексическими единицами языка являются идентификаторы, ключевые слова, литералы, знаки операций и разделители. Идентификатор есть последовательность букв либо цифр, начинающаяся с буквы. Ключевые слова представляются списком выделенных идентификаторов, играющих роль синтаксических разделителей в структуре предложений программы.

Ключевые слова можно сокращать, пока сокращение остается однозначным. Контекст ключевых слов строго определен синтаксисом, поэтому они не являются зарезервированными идентификаторами и могут использоваться в качестве имен для вводимых пользователем объектов. Примеры:

A1, RFOG, COUNT — идентификаторы,

FOR, DEFINE, END — ключевые слова.

Литералы служат для изображения значений и разделяются на числовые и текстовые. Числовые литералы записываются в десятичной либо восьмеричной системе исчисления. В десятичной форме за последовательностью цифр следует символ «D», а в восьмеричной — «B». По умолчанию принимается восьмеричная форма. Текстовые литералы записываются последовательностью символов, заключенных в апострофы. Примеры:

17, 20000000, 100B, 77777777B

199D, 1000D

'ТЕКСТ'

Символ «двойная кавычка» является началом комментариев, к которым относится весь остаток строки.

Строка текста программы без номера-метки представляет собой синтаксическое понятие предложения. Предложение — это последовательность операторов (возможно, пустая), разделенных точкой с запятой.

Операторы программы задают некоторые манипуляции над внутрипрограммными объектами (константами и переменными).

Всякий объект должен быть описан до его использования в программе. Описание константы ставит в соответствие некоторому идентификатору значение выражения, и это значение остается неизменным на протяжении исполнения программы. Примеры описаний констант:

10 CONST ОДИН = 1, СТО = 100D, CHAN = 11000000

20 CONST A = 1750, B = (A + ОДИН − СТО) * 3 + CHAN

Переменная в отличие от константы есть именованный объект, способный хранить значение, которое может изменяться в ходе выполнения программы. В языке различаются два вида переменных: простые переменные и массивы. Простая переменная представляется 16-битовым набором, который может интерпретироваться как целое число. Массив — вектор из простых переменных. Доступ к отдельным элементам массива осуществляется с помощью индекса. Границы индексов массива определяются при его описании двумя возможными способами:

нижний индекс ... верхний индекс

нижний индекс : размер массива

Приведем пример описания переменных:

10 DEFINE X, Y(5), Z(2..100D), W(3 : 4)

Здесь X — простая переменная, Y — массив с индексами в интервале 0..4, Z — массив с индексами 2..100, W — массив с индексами 3..6.

Все приведенные примеры представляют переменные, которые размещаются в памяти компилятором (этот процесс пользователем не контролируется). При описании переменной можно задать принудительное размещение в памяти, на общей или тестовой шине. Адрес на общей шине записывается вслед за описателем переменной в виде адресного выражения, заключенного между символами «обратная косая черта». В зависимости от значения это может быть математический адрес памяти либо адрес периферийного регистра. Адрес размещения на тестовой шине записывается вслед за описателем переменной в виде адресного выражения в квадратных скобках. Адресное выражение может содержать лишь константы и литералы, т. е. должно вычисляться до исполнения программы. Размещение может быть задано относительно определенной ранее переменной. Приведем примеры:

```
10 CONST BASE = 20000000, N = 20D
20 DEFINE A[BASE], IO(4096D)\160000\, C(1 : 200B)
30 DEFINE D (5 : 2) = C(N + 5), KBCSR = IO(17560)
```

Здесь BASE и N — константы; A — массив на тестовой шине; IO — страница внешних регистров общей шины; C — массив с автоматическим размещением; D — массив из двух элементов, начало которого совпадает с 25-м элементом массива C; KBCSR — регистр управления клавиатуры консольного терминала.

Синтаксис таков, что всякий оператор языка BUSIC начинается с ключевого слова. Это позволяет при работе в калькуляторном режиме рассматривать операторы на одном концептуальном уровне с командами системы.

Оператор присваивания служит для изменения значений переменных и записывается следующим образом:

LET *обозначение переменной = выражение*

Обозначение переменной может определять простую переменную, элемент массива либо вырезку, т. е. вектор, полученный сужением диапазона индексов некоторого массива. Вырезка задается указанием имени массива и ее диапазона (в случае если массив входит в вырезку целиком, достаточно одного имени). Диапазон вырезки записывается парой индексов, разделенных многоточием либо начальным индексом и размером вырезки. Эти параметры могут быть выражениями самого общего вида. В правой части оператора присваивания простой переменной (вырезке) должно стоять выражение, определяющее новое значение переменной (вырезку). Таким образом, в языке разрешено присваивание векторных значений. Оператор присваивания — единственный оператор языка, в котором можно опускать ключевое слово (LET).

Выражение в языке BUSIC строится из первичных поставщиков значений, знаков операций и скобок, управляющих порядком выполнения операций. В качестве поставщиков значений могут служить изображение литерала, имя константы или простой переменной, имя массива с индексом. Кроме того, в языке есть конструкторы битовых наборов, которые записываются в виде последовательности слогов, разделенных запятыми и заключенных в квадратные скобки. Слог есть сегмент единичных битов набора. Его диапазон задается номерами граничных битов, разделенных многоточием, либо номером граничного бита и размером слога. Биты в слове нумеруются с нуля справа налево. Приведем примеры битовых наборов, записывая рядом их восьмеричное представление:

```
[17, 2]            = 100004
[16 : 4, 3]        = 074010
[17..14, 5, 2..3]  = 170054
[17 : 10, 0]       = 177401
[17, 14, 11, 6, 3, 0] = 111111
```

Конструктор битового набора не является изображением значения, так

как определяющие диапазон параметры могут быть сколь угодно сложными выражениями.

Операции разделяются по приоритетам на пять уровней. При отсутствии скобок операции одного уровня исполняются в порядке написания, а разных — в порядке приоритетов. Имеются следующие классы операций (перечисленные в порядке возрастания приоритетов): операции сравнения, аддитивные, мультипликативные, логические, одноместные.

В языке есть шесть операций числового сравнения: больше (>), меньше (<), равно (=), не равно (≠), больше или равно (>=), меньше или равно (<=). Если отношение верно, результат представляется битовым набором из единиц, а иначе равен нулю.

Имеется две аддитивные операции: сложение (+) и вычитание (—), а также две мультипликативные: умножение (*) и деление (/).

Набор логических операций следующий: логическое И (∧), логическое ИЛИ (∨), исключающее ИЛИ (#), чистка под маской (!), сдвиг вправо (—>), сдвиг влево (<—) и извлечение поля (%).

Результатом операции чистки под маской является значение левого операнда с нулевыми битами в тех позициях, где есть единицы в правом операнде. В операциях сдвига левый операнд — сдвигаемое значение, а правый — счетчик сдвигов. Операция извлечения поля состоит в следующем: значения левого и правого операндов сдвигаются вправо (по правилам операции сдвига) до появления единицы в младшем бите правого операнда, после чего сдвинутые значения логически перемножаются. Если правый операнд равен нулю, то результат полагается равным нулю.

Одноместных операций две: поразрядная инверсия (') и двоичное дополнение (—).

Приведем примеры:

```
10 X = 17D; Y(N/4) = [15 : 2] + 4
20 KBCSR = 100
30 C(5) = -(X + 'W(2) ∧ Z(2)) + N * -2
40 Z(5...10) = C(2...5); C = Z
```

В языке имеются средства управления ходом выполнения программы: операторы перехода, операторы вызова подпрограммы и возврата, условные предложения и итеративные циклы.

Оператор перехода позволяет организовать безусловную передачу управления на некоторое предложение программы по его метке (GO TO *номер предложения*).

Оператор вызова подпрограммы (CALL *номер предложения*) также позволяет передать управление, но если далее выполнится оператор возврата (RETURN), то управление вернется оператору, следующему за вызовом. Вызовы могут быть вложенными.

Если по ходу программы необходимо исполнить предложение, то составляющие его операторы будут исполняться последовательно. В языке есть понятие условия выполнимости, которое, будучи истинным, не влияет на ход программы, а будучи ложным, запрещает выполнение следующих за ним операторов (но лишь в пределах одного предложения). Это условие является не оператором, а некоторой приставкой к оператору, служащей его динамическим «охранником». Условия выполнимости бывают двух видов: (IF *выражение*:) либо (IFNOT *выражение*:). Условие первого вида считается выполненным, если выражение отлично от нуля, а второго — если равно нулю. Приведем примеры:

```
10 MAX = B;  IF A > B:  MAX = A
11 IF X ∧ [7]:          CALL 1000
12 IFNOT X ∧ [17, 5, 0]:  CALL 2000
```

В предложении 10 переменная MAX получит наибольшее из значений переменных A и B. Вызов подпрограммы в строке с номером 11 произойдет при равенстве седьмого бита переменной X единице, а под-

го (TO *выражение*) с указанным шагом. В циклическом процессе участвуют все операторы, следующие за оператором цикла до оператора завершения цикла (NEXT *переменная*). Такого сорта пары могут быть правильно вложенными друг в друга. Приведем пример фрагмента программы:

```
10 CONST L = 10, H = 100
20 DEF   A(L..H), I
30 FOR I = L TO H
40       A(I) = 0
50 NEXT I
```

Предложения 30, 40 и 50 представляют фрагмент, обращающий в нуль все элементы массива A.

В языке имеются операторы ввода-вывода, позволяющие программировать операции обмена с терминалом и файлами. Язык обеспечивает гибкие средства форматирования выходной информации.

Программа завершает свою работу в результате исполнения оператора END или после исполнения последнего предложения программы.

Средства отладки. Как и большинство диалоговых систем с языковым процессором, BUSIC имеет встроенные отладочные средства. Поскольку главное назначение системы — настройка оборудования, то отладочный режим для нее часто бывает рабочим, т. е. используется для поиска ошибок не только в программе, но и в аппаратуре при правильной работе программы. Традиционная схема реализации позволяет легко встраивать средства отладки в механизм интерпретатора с возможностью полного контроля над исполнением. BUSIC является компилирующей системой, и при реализации описываемых далее возможностей есть определенные технические трудности. Следует также отметить, что для работы отладочных средств требуются дополнительные ресурсы процессора и, возможно, значительные. В случае поиска ошибок в программе с этим легко смириться, а при наладке аппаратуры могут недопустимо измениться временные диаграммы, т. е. возникает дополнительное требование минимизации процессорных затрат на отладочные механизмы.

В системе BUSIC имеется возможность прерывания (приостановки) программы, т. е. перехода в режим диалога с целью контроля или изменения текущего состояния программных объектов. Независимо от причины приостановки программы продолжить ее исполнение, если не был изменен программный текст, можно командой CONTINUE. Приостановка может быть предусмотрена в самой программе, задана извне либо произведена с терминала в любой момент работы программы.

Оператор приостановки (STOP) переводит систему в калькуляторный режим, и программа может быть продолжена только командой CONTINUE. Можно также задать приостановку перед исполнением некоторого предложения программы, не меняя ее текста. Это достигается командой BREAK с номером предложения в качестве аргумента, выданной перед запуском программы либо продолжением. Отсутствие аргумента в команде BREAK отменяет действие команды.

В любой момент программа может быть прервана нажатием кнопки «Стоп» на клавиатуре терминала. Прерывание происходит в ближайшей (по времени) возможной точке обычно перед исполнением следующего предложения. Некоторые операторы, например операторы вывода и циклы, допускают прерывание до завершения, поскольку могут исполняться в течение длительных отрезков времени.

Существует режим пошагового исполнения программы. Команда STEP позволяет исполнить одно предложение программы и вернуться в состояние приостановки.

При переходе в режим приостановки на терминале печатается номер строки, перед (в) которой произошел останов. Далее система находится в калькуляторном режиме, причем доступны все программные объекты. Часто для контроля за ходом работы программы требуется производить просмотр одних и тех же объектов или выполнение однотипных действий. Существует команда

DISPLAY *строка*

которая позволяет запомнить строку, являющуюся аргументом команды, и исполнять ее как самостоятельную команду при каждом переходе в режим приостановки. Например, после команды

DISPLAY TYPE 'A, B, C =', A, B, C

при каждой приостановке будут распечатываться значения переменных A, B и C. Команда

DISPLAY CALL 1000

позволяет автоматически выполнять на каждой приостановке сколь угодно сложную подпрограмму.

В системе BUSIC имеются средства трассировки программы, управления режимами компиляции и исполнения, а также другие возможности технического характера. Все команды и операторы, относящиеся к режиму отладки, допускают сокращение до одной буквы, что удобно для оперативной диалоговой работы.

Заключение. Представленная работа демонстрирует пример внедрения языковой программной системы в сферу деятельности инженеров-разработчиков аппаратуры. Здесь не описаны существующие в системе специализированные средства, имеющие отношение к конкретным особенностям задачи [3]. Мы попытались дать неформальное представление о наиболее общих решениях, принятых при разработке.

Авторы считают необходимым отметить вклад в работу С. Л. Ивашина, реализовавшего ряд внутренних механизмов системы и все компоненты драйверного уровня. Авторы признательны также большому коллективу инженеров, которые фактически проводили тестирование системы, принимая ее непосредственно в рабочую эксплуатацию.

ЛИТЕРАТУРА

1. **Бредихин С. В., Песляк П. М.** Простой интерактивный язык для КАМАК.— Новосибирск, 1975. (Препринт/АН СССР, Сиб. отд-ние, ИАиЭ; № 28).
2. **Бредихин С. В., Песляк П. М.** CATY-M: система для программирования аппаратуры КАМАК. (Модифицированный вариант).— Новосибирск, 1981. (Препринт/АН СССР, Сиб. отд-ние, ИАиЭ; № 166).
3. **Ковалев А. М., Талныкин Э. А.** Машинный синтез визуальной обстановки.— Автометрия, 1984, № 4.

Поступила в редакцию 20 апреля 1984 г.

УДК 681.3.06

И. М. КАГАНСКИЙ

(Новосибирск)

СТРУКТУРИРОВАНИЕ ПРОГРАММ СРЕДСТВАМИ МАКРОАССЕМБЛЕРА В ОС ЕС

Введение. Язык программирования Ассемблер предоставляет программисту полную свободу в отношении стиля. Нередко ассемблер-программы написаны так, что к ним неприменимо понятие стиля. Была