Блок-схемы реализации режимов обмена в рамках процедур таблицы приведены на рис. 2—4.

Диалоговый режим описания эскиза топологии позволяет легко обеспечить правильность синтаксического описания, так как трансляция идет в шаговом режиме. Транслятор в языке ЭТОП программно замкнут относительно ЭВМ «Электроника-100И» и обеспечивает частичный семантический контроль входной информации.

ВЫВОДЫ

Проведен анализ графической информации о топологии ИС и приведен состав информации, необходимой для описания эскиза топологии.

Разработаны методика графического описания эскиза топологии и диалоговый язык для описания эскиза топологии, учитывающий особенности полуавтоматов ввода графической информации в режиме «on line».

ЛИТЕРАТУРА

1. Р. В. Дмитришин. Об одной стратегии адаптивной оптимизации цепей на ЭВМ.— В кн.: Материалы конференции «Автоматизация научных исследований на основе применения ЭВМ». Новосибирск, 1974.
2. Г. Л. Левин. О входном языке моделирования электронных схем.— Вопросы радиоэлектроники, 1972, вып. 1.
3. В. А. Селютин. Язык для описания топологии МОП — БИС.— В кн.: Микроэлектроника, т. 2, вып. 5. М., АН СССР, 1973.
4. G. Bracchi, Ferrary. A Language for Treating Geometric Patterns in a Two — Dimensional Space.— Comm. of ACM, 1971, v. 14, № 1.
5. В. И. Зубков, В. А. Маслов, Г. С. Остапенко, В. Н. Харин. Устройство ввода графической информации в ЭВМ.— В кн.: Электроника. Воронеж, Изд. политехнического института, 1972.
6. В. Н. Харин, В. С. Квачева. Идентификация топологии и электрической схемы в полупроводниковых интегральных схемах.— В кн.: Материалы семинара «Современные методы разработки РЭЛ», М., МДНТП им. Ф. Э. Дзержинского, 1974.

Поступила в редакцию 17 сентября 1974 г.

УДК 681.306

Ю. З. КЕКЕЕВ, Э. А. ТАЛНЫКИН, Н. С. ЯКОВЕНКО
(Новосибирск)

ТЕКСТОВОЙ РЕДАКТОР НА БАЗЕ АЛФАВИТНО-ЦИФРОВОГО ДИСПЛЕЯ

Основные принципы. В статье описывается реализация текстового редактора на базе алфавитно-цифрового дисплея «Videoton-340». Дисплей обладает автономной памятью на размер экрана (16 строк по 80 символов), что позволяет хранить на экране текст без дополнитель-

ных затрат на генерацию изображения. Клавиатура дисплея включает русские и латинские буквы, цифры, знаки пунктуации, а также функциональные клавиши, позволяющие управлять размещением текста на экране, а точнее, осуществлять операции редактирования в памяти дисплея. Позиция текущего символа обозначается на экране специальным маркером (световым пятном), так что при нажатии некоторой клавиши на клавиатуре дисплея изображение соответствующего символа появляется непосредственно над маркером, который затем переходит на одну позицию вправо. Кроме операций редактирования: вставить строку, удалить строку, вставить символ, удалить символ — функциональная клавиатура позволяет управлять перемещением маркера по экрану.

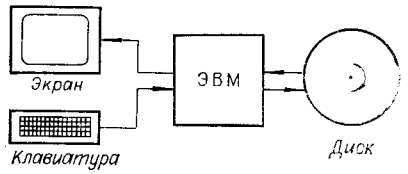

Рис. 1.

Все сказанное относится к автономной работе дисплея (в режиме «off line»). При работе на линии с ЭВМ клавиатуру и экран дисплея можно рассматривать как два совершенно различных устройства (рис. 1). После нажатия клавиши на клавиатуре в ЭВМ поступает код, а на экране ничего не изменяется. Здесь заботу о том, чтобы вводимые символы воспроизводились на экране, должна брать на себя ЭВМ, а именно выводить на экран каждый получаемый с клавиатуры символ. Такой режим работы («эхо») может быть реализован или программно, или аппаратно, как одна из функций интерфейса. В нашем случае интерфейс обеспечивает обе возможности, позволяя программно переключаться между ними.

В программной реализации режима «эхо» заключается принцип работы предлагаемого редактора. Такой режим позволяет проанализировать введенный код, а затем вывести совсем другой или даже некоторую комбинацию. Это значительно расширяет функциональные возможности клавиатуры. Например, можно организовать табуляцию, т. е. выдавать по получении символа «TAB» пробелы до ближайшей справа позиции, из размеченных ранее.

Совершенно естественно возникает идея разместить в памяти ЭВМ образ текста на экране, синхронно отображая на нем все изменения, происходящие на экране. В памяти можно разместить текст и большего объема, а наличие диска позволяет получить ситуацию, изображенную на рис. 2. На диске хранится текст, вообще говоря, любой длины. В памяти находится копия некоторого участка его, а некоторая часть текста из памяти отображается на экране.

Далее код, поступающий с клавиатуры, интерпретируется в соответствии с его функциональным назначением, что сводится к возможным изменениям на экране, в памяти и на диске. Интерпретация символа на экране чаще всего совпадает с его аппаратной интерпретацией, т. е. в режиме «off line», но иногда может от нее отличаться.

При перемещении маркера в пределах экрана в операциях участвует только сам экран и его прообраз в памяти (см. рис. 2). Если, скажем, попытаться переместить маркер вверх, когда он уже находится в первой строке, то прообраз экрана начинает двигаться вверх по тексту в памяти, последняя строка на экране пропадает и сверху появляется новая. Если же в этой ситуации не хватает текста в памяти, то прообраз памяти начинает переме-

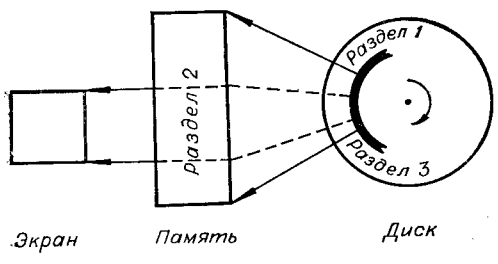

Рис. 2.

щаться по диску, т. е. последняя строка из памяти переписывается на диск, а сверху считывается новая. Этот факт остается скрытым, и весь процесс редактирования выглядит очень естественно. Человек получает возможность видеть текст через «окно», которое свободно перемещается вверх и вниз по тексту.

Обработка печатного символа также, как и операции редактирования, в пределах экрана происходит практически мгновенно. Максимальное время, связанное с перемещение в памяти, на диске и высвечиванием новой строки, составляет 1/2 с. При этом отсутствует ощущение ожидания, поскольку изменения на экране происходят перед исполнением внутренних операций, а не наоборот.

Наряду с функциональными возможностями клавиатуры, реализованными аппаратно, в качестве стандартных заложены различные режимы табуляции, а также возможность получать нажатием одной клавиши часто употребляемые при написании программ слова: PROCEDURE, FUNCTION, BEGIN и т. п. Функциональная интерпретация клавиш имеет возможность расширения, т. е. во время работы пользователь может определить способ интерпретации некоторого кода, скажем, приписать некоторой клавише комбинацию действий других функциональных клавиш или печатных символов. Кроме того, имеется богатый набор управляющих возможностей: высветить текст с заданной строки, двигать экран по тексту вверх или вниз, удалить некоторый участок текста, поменять местами два различных участка и т. д. Все управляющие функции оформлены как директивы редактора.

Редактор является полной системой и помимо собственно процесса редактирования занимается введением архива текстов. Каждый пользователь редактора может иметь доступ к своему тексту через пароль и имя. С текстами в архиве редактора можно производить обычные действия: удаление, создание, смену имени, копирование на любой носитель, копирование в архив операционной системы или в системную область для трансляции. Редактор реализован на ЭВМ HP2116B на языке высокого уровня [1].

Структура данных. В целом текст состоит из (см. [2]) совокупности строк, имеющих сквозную нумерацию. В процессе редактирования (см. рис. 2) текст можно представить разделенным на три плавающих и различных по структуре раздела. Если процесс редактирования рассматривать как однократный просмотр сверху вниз, то первый раздел представляет элемент готовой продукции и находится на диске. Второй раздел располагает в памяти, где собственно и происходит редактирование. Третий раздел представляет участок текста, подлежащий просмотру, и может размещаться частично на диске, частично на другом носителе. Если учесть возможность свободного движения вверх и вниз по тексту, а также возможность иметь исходный редактируемый текст не на диске, а на другом носителе, все сказанное выше, так же как и рис. 2, представляет упрощенную картину. Обмен информацией происходит только между соседними разделами на уровне строк.

Первый раздел организован по принципу стека и обслуживается двумя подпрограммами: *описать выход* и *читать выход*. В начале работы редактора первый раздел пуст и начинает последовательно заполняться при продвижении вниз по тексту. При чтении из первого раздела результатом будет являться последняя, записанная туда строка, которая затем вычеркивается. Запись в первый раздел происходит последовательно, т. е. новая строка будет переписана вслед за более поздней из присутствующих в разделе.

Третий раздел, аналогично первому, обслуживается двумя программами: *читать вход* и *писать вход*, но кроме идентичного стека име-

ет дополнительный ресурс, обеспечиваемый исходным редактируемым текстом. Сущность ресурса в том, что при чтении, если стек пуст, читается строка исходного текста. Таким образом, исходный текст может находиться на устройстве с последовательным доступом. При операции записи или в случае чтения, когда стек не пуст, вступает в силу механизм из первого раздела. Следует отметить также, что все эти подпрограммы производят логическое чтение и запись. Они динамически делят между собой свободную память, буферизуя операции обмена. Физическое же обращение к диску происходит при записи лишь в случае переполнения буфера, а при чтении,— когда буфер пуст.

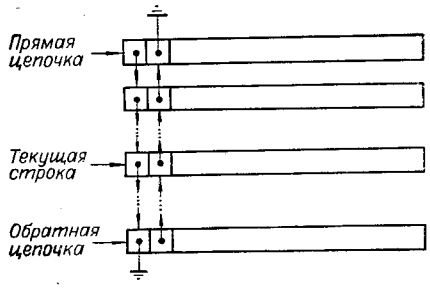

Рис. 3.

В отличие от описанной безадресной структуры хранения информации текст второго раздела представляется в виде двойного списка (рис. 3). Каждый элемент списка является записью, состоящей из двух ссылочных полей, указывающих на следующую и предыдущую строки соответственно. Ссылочные поля образуют две цепочки (прямую и обратную), одна из которых строится в порядке возрастания, а вторая — в порядке убывания номеров строк (но не адресов в памяти). Каждая из цепочек завершается пустой ссылкой.

На рис. 3 видно, что по модулю четырех упомянутых выше подпрограмм внутренняя структура данных симметрична относительно направлений вниз и вверх. Следующие шесть переменных: указатель прямой цепочки, указатель обратной цепочки, указатель текущей строки, номер текущей строки на экране, номер текущей позиции в строке и номер текущей строки в тексте задают полную систему независимых координат, определяющую состояние системы.

Такая, удобная для операций редактирования и симметричная структура данных в памяти, а также отведенная на задний план структура размещения информации на диске позволили сделать логическую (программную) часть редактора до предела простой и наглядной.

Проблема «сборки мусора», возникающая во всякой системе обработки списков, обходится явным присоединением каждой освобождаемой записи к списку свободных строк.

Логическая структура. Центральное место в программной структуре редактора занимает переключатель по коду введенного символа, позволяющий быстро активизировать нужную интерпретирующую программу.

Основные принципы, выдерживаемые относительно горизонтальных перемещений, состоят в запрещении выхода за пределы текущей строки с помощью клавиш клавиатуры, вызывающих горизонтальное перемещение маркера. Хотя в режиме «off line», например, движение маркера вправо вызывает автоматический переход на следующую строку (вследствие цикличности памяти дисплея).

Если при перемещении вниз или вверх обнаруживается конец цепочки, вызывается специальная программа, устраняющая эту ситуацию с помощью комбинации четырех подпрограмм обмена информацией между разделами (см. выше).

Списковая структура обеспечивает простоту в операциях над строками. Например, для удаления строки или группы последовательных строк достаточно изменить всего две ссылки. Для того чтобы вставить в текст новую строку, необходимо взять ее из свободного списка и по-

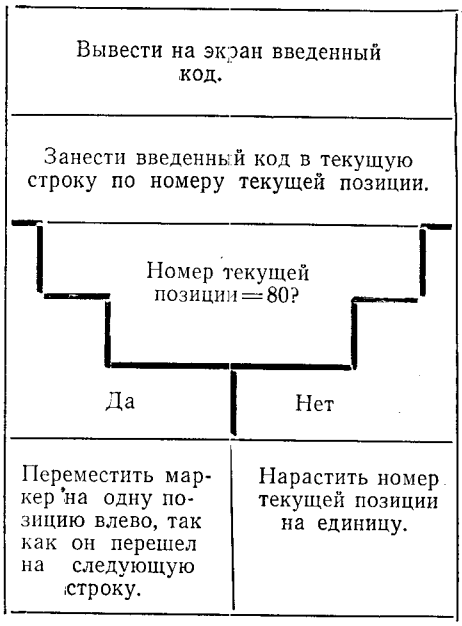

Рис. 4

местить в нужное место в тексте. Здесь для настройки цепочек требуется изменить четыре ссылки. В случае если свободный список пуст, вызывается программа, освобождающая одну строку в верхушке той цепочки, которая не пересекается с прообразом экрана. Освобождаемая строка переписывается в первый или третий разделы текста.

При редактировании внутри строки происходит физическое перемещение элементов текста. Списковая структура здесь не дает видимого выигрыша во времени и требует значительного дополнительного объема памяти.

Для примера на рис. 4 приведена блок-схема интерпретации информационного символа (буквы, цифры и т. п.). На рис. 5 изображена интерпретация клавиши, вызывающей перемещение маркера вверх по экрану.

Организация. В работе системы можно выделить четыре основных состояния (рис. 6). В состоянии C_0 работает монитор, осуществляющий интерпретацию директив. В этом состоянии система обеспечивает доступ к архиву: распечатка, копирование на любые носители, создание, разрушение текстов, смена имен, копирование в архив операционной системы или на трансляцию. Если монитор успешно распознает директиву, с диска вызывается исполняющая программа, которой передаются параметры директивы, а затем управление. После того как вызванная программа завершает работу, она вызывает в память монитор, который снова ждет очередной директивы. В этом отношении система является неограниченно расширяемой.

По директиве EDIT вызывается собственно программа-редактор, которая принимает переданные ей монитором параметры, инициализирует внутренние структуры данных (считывает в память необходимую часть текста, выстраивает списки, выставляет указатели) и высвечивает на экране текст, начиная с первой строки. Из 16 строк, имеющихся на экране, под текст отводится только 13, одна строка содержит разметку позиций, а остальные используются как управляющая область для сообщений в процессе работы и приема управляющих директив. Теперь система находится в состоянии C_1, где можно собственно редактировать текст в соответствии с положением маркера на экране. Изображение текста полностью соответствует его копии в памяти и в таком же виде он будет переписан на диск.

Нажатием клавиши «ETX» (конец текста) система переводится в состояние C_2, где она способна принимать управляющие директивы. При входе в C_2 маркер переводится в управляющую область. Директивы в этом состоянии распознаются по нажатию одной клавиши (в большинстве случаев это первая буква слова, определяющего содержание директивы), и, если код распознан, печатается полное название директивы, после чего необходимо ввести требуемые параметры. Если директива не предполагает параметров, он исполняется сразу. После исполнения директивы система снова возвращается в состояние C_1. Петля в C_2 на рис. 6 отражает случай нераспознанной директивы. В состоянии C_2 пользователь может: разметить табулятор; узнать номер

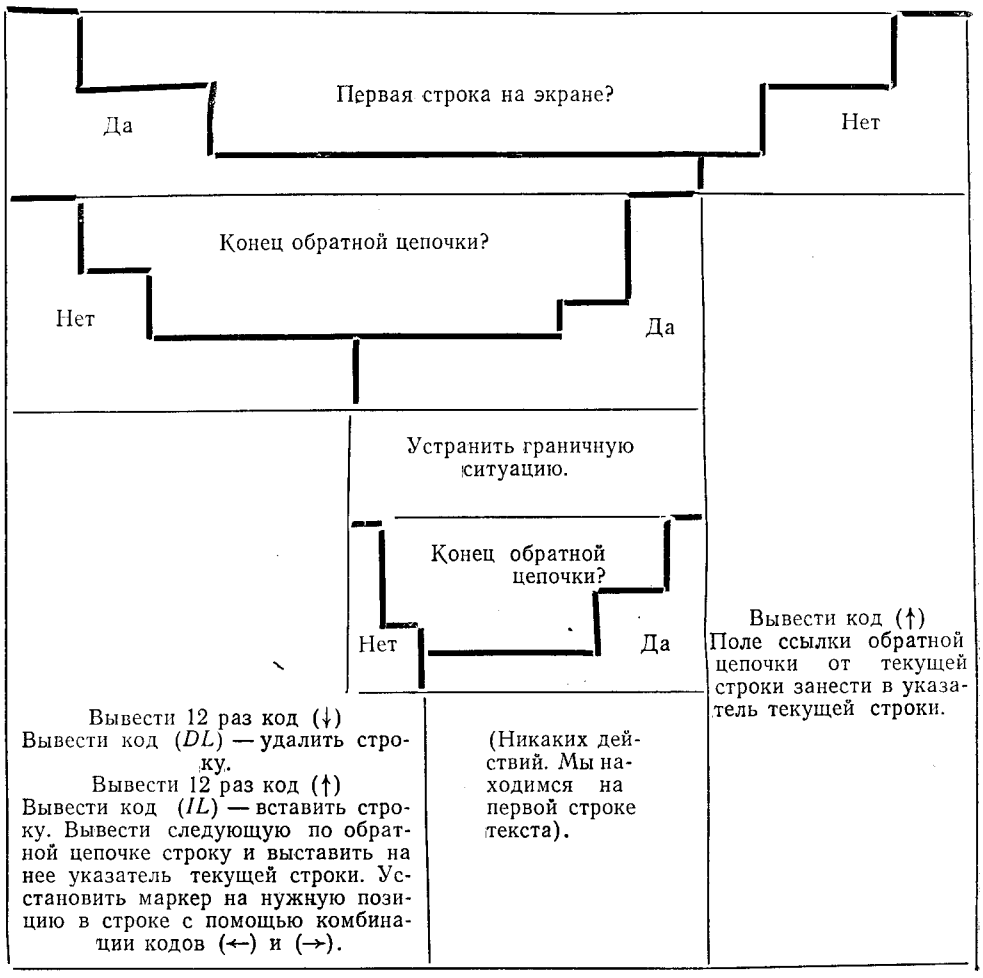

Рис. 5

строки, на которой находится маркер; высветить текст с заданной строки, вызвать автоматическое движение текста вверх или вниз относительно экрана; завершить процесс редакции; определить новую интерпретацию некоторой клавиши, для чего необходимо набрать последовательность клавиш, оканчивающуюся клавишей «ETX», после этого первая из них в состоянии C_3 будет интерпретироваться комбинацией кодов остальных клавиш, не включая «ETX». Переход в состояние C_3 осуществляется нажатием клавиши «T» на регистр «CTRL». Если в состоянии C_3 нажимается клавиша, не предусмотренная в стандартной таблице или не определенная в C_2, она игнорируется (соответствует петле в C_3 на рис. 6). Переход из C_3 в C_1 происходит автоматически после отработки одного кода. По директиве завершения редакции в C_2 происходит копирование всего текста в раздел 1, вызов монитора и переход в состояние C_0.

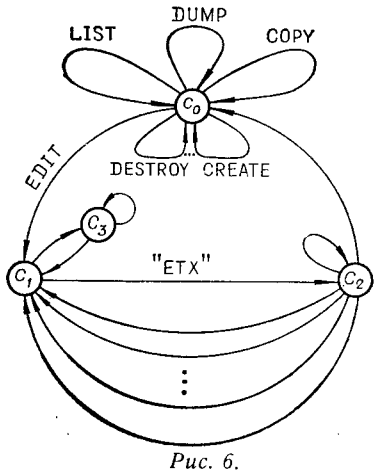

Рис. 6.

Заключение. Эксплуатация редактора в течение года в ИАЭ СО АН СССР показала его высокую эффективность при редактировании программ и любых других текстов. В совокупности с системой документирования [3], модернизированный вариант которой был реализован также на HP2116B, редактор является мощным средством для подготовки различного рода документов.

Так как дисплей «Videoton-340» входит в стандартный комплект ЕС 1010 и, по-видимому, будет включаться в другие системы этой серии, адаптация описанной разработки на ЕС ЭВМ представляется перспективной.

ЛИТЕРАТУРА

1. П. М. Песляк, Э. А. Талныкин. Язык системного программирования для мини-ЭВМ.— Автометрия, 1974, № 4.
2. D. E. Knuth. The Art of Computer Programming. V. 1. Addisson. Massachusetts, Wesley Publishing Company, Inc. Reading, 1968.
3. В. А. Мелешихин, П. М. Песляк, Э. А. Талныкин. Система автоматизации документированная на базе ЭВМ «Минск-32».— Автометрия, 1974, № 4.

Поступила в редакцию 30 сентября 1974 г.

УДК 681.833 : 519.2

А. Н. ДОМАРАЦКИЙ, Л. Н. ИВАНОВ, В. А. ПОПОВ

(Новосибирск)

МНОГОПРОЦЕССОРНЫЙ ЦИФРОВОЙ КОРРЕЛЯТОР

Применение достижений современной вычислительной техники предопределяет создание структурных схем оперативных статистических анализаторов в виде многопроцессорных устройств [1]. Поэтому такая структура была положена в основу построения статистического анализатора САДКО [2], предназначенного для решения комплексных задач обработки экспериментальных данных в реальном масштабе времени. В качестве составной части САДКО, осуществляющей оперативный корреляционный анализ случайных сигналов в широкой полосе частот, разработан коррелятор [3], использующий многопроцессорную структуру анализатора.

Подобное построение коррелятора обладает целым рядом преимуществ: повышением быстродействия устройства пропорционально числу процессоров; возможностью обработки данных от нескольких датчиков; расширением частотной полосы устройства [1, 4].

Алгоритм работы коррелятора имеет вид

$$K_x(\nu\delta\tau + \mu\Delta\tau) = \frac{1}{N}\sum_{i=1}^{N} x(i\Delta t)\,x(i\Delta t + \nu\delta\tau + \mu\Delta\tau),$$
$$\nu = 1, 2, \ldots, m; \qquad \mu = 1, 2, \ldots, l; \qquad n = ml, \tag{1}$$

где Δt — шаг дискретизации квантованных по уровню реализаций случайного сигнала $x(t)$, N — объем выборки, n — количество вычисляемых точек корреляционной функции. Частные произведения $x(t)x(t+\tau)$ определяются в m процессорах, работающих с временным сдвигом $\delta\tau$ (величину $\delta\tau$ можно сделать намного меньше интервала Δt).

В корреляторе анализатора САДКО реализован алгоритм (1) при $m=8$, $l=64$, $n=512$, т. е. он является устройством параллельно-после-