МИКРОПРОЦЕССОРНЫЕ СРЕДСТВА И СИСТЕМЫ

5 | 1987

ISSN 0233-4844

Машинная графика: интерактивное конструирование реалистических видеообъектов на базе АРМ с векторным процессором

Интерфейс VME — средство для построения мощных многопроцессорных систем

Программные трюки на МАКРО-11: необычные приемы программного решения «острых» ситуаций для микроЭВМ «Электроника 60»

БИС контроллера прерываний К588ВН1 выполнена в соответствии с требованиями стандартного интерфейса, обслуживает запросы на прерывание от двух внешних устройств

Четырехразрядные однокристальные ЭВМ КБ1013ВК1-2, КБ1013ВК4-2 содержат на кристалле: ЦП, ОЗУ данных и ПЗУ программ, таймер-счетчик, контроллер жидкокристаллических дисплеев и др. функциональные узлы, характерные для комплекта БИС одноплатной микроЭВМ

ГРАФИЧЕСКИЙ АССИСТЕНТ ДЛЯ СИНТЕЗА, КАТАЛОГИЗАЦИИ И ЭКСПОНИРОВАНИЯ ЦВЕТНЫХ ИЗОБРАЖЕНИЙ

(К ст. Кулачева А. П.)

Фрагмент сложного восточного коврового узора, построенный из элементов изображения средствами графического ассистента ГРАСС для ЭВМ серии «Электроника»

Многокомпонентное изображение, иллюстрирующее элементарные возможности графического ассистента ГРАСС в плане шрифтовых наборов и построения геометрических фигур. Автором и исполнителем иллюстрации является ученик 3 класса Миша Рамендик

Пролетарии всех стран, соединяйтесь!

ОРГАН ГОСУДАРСТВЕННОГО КОМИТЕТА СССР ПО ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКЕ И ИНФОРМАТИКЕ

Издается с 1984 года

МП МИКРОПРОЦЕССОРНЫЕ СРЕДСТВА И СИСТЕМЫ

ВЫХОДИТ ШЕСТЬ РАЗ В ГОД

НАУЧНО-ТЕХНИЧЕСКИЙ И ПРОИЗВОДСТВЕННЫЙ ЖУРНАЛ 5 / 1987 МОСКВА

СОДЕРЖАНИЕ

Сдано в набор 20.08.87. Т-20968
Подписано к печати 10.11.87
Формат 84×108¹/₁₆. Бумага № 1.
Высокая печать. Усл. печ. л. 10.08.
Уч.-изд. л. 14.1. Тираж 90 186 экз.
Заказ 228. Цена 1 р. 10 к.
Орган Государственного комитета СССР по вычислительной технике и информатике
Московская типография № 13
ПО «Периодика» ВО «Союзполиграфпром Госкомиздата СССР
107005, Москва, Денисовский пер., дом 30

На первой странице обложки — Машинная графика (см. статью Алексеенко М. М. и др.).

КАК ПЕРЕСТРОИТЬСЯ ПРОГРАММИСТАМ

У нас где-то на 50 тыс. ЭВМ работает около 200 тыс. программистов. Через несколько лет количество программистов возрастет от силы процентов на тридцать, а количество ЭВМ (естественно, с учетом персональных) перевалит за миллион. Обстановка изменится радикально. Возникает вопрос: как перестроиться программистам.

Горизонты науки в программировании чисты и безграничны. Однако ее дальнозоркий взгляд не всегда различает ту ближайшую точку, на которую надо прыгнуть в данный момент, когда очередная точка опоры уходит из под ног.

Наша редакция получает немало тревожных писем от профессиональных программистов, которые предвидят опаснейшие последствия механического переноса стиля, методов и организации разработки и использования программного продукта, (сложившиеся в есэвско-эсэмовский период программирования 70-х годов) на работу в условиях персональных ЭВМ, локальных сетей и растровых дисплеев. Подборка некоторых из этих писем публикуется в данном выпуске журнала (см. стр. 18, 74).

Что же было такого характерного в прошлом, чего никак нельзя пропускать в будущее. Конечно, нет простого ответа на этот вопрос. Но первая и, пожалуй, главная доля ответа в том, что вчера мы должны были работать на таких машинах и таких программных средствах, какие получались, а не какие были нужны.

В результате каждая ЭВМ обрастала десятком инженеров и программистов, которые старались каким-то образом обжить установку и сделать ее приемлемой для пользователя.

Заметим, что с точки зрения общего расхода человеческих ресурсов — это те же несколько десятков тысяч человек, которые составляют территориальную службу ИБМ. Однако наши «парасистемные» программисты работают в полной изоляции друг от друга, каждый на своем месте, решая по существу одну и ту же задачу, но с нюансами различий, которые лишают их труд какой бы то ни было всеобщности. Отсюда то, что мы читаем в письмах и наблюдаем в своих вычислительных центрах.

Это одно из наиболее печальных проявлений внеэкономического подхода к разработке программного обеспечения.

Мы озабоченно сверяем даты освоения тех или иных технических параметров ЭВМ, поверяя нашу работу с мировой тенденцией. Гораздо большим предметом беспокойства представляется то, что промышленность программных средств, составляющая сейчас в мировой экономике отрасль с оборотом в десятки миллиардов долларов, выступает в нашей инфраструктуре бесплотным призраком, не имеющим ни цены, ни качеств потребительского продукта, ни рынка, ни иных каналов распределения, ни рекламы, ни службы рекламаций, ни авторов, ни вознаграждения.

Несколько лет назад ГКНТ СССР начал разработку нормативных материалов, позволяющих трактовать программный продукт как средства производства. Эта работа однако затянулась. Другим недостатком проекта было практическое игнорирование потребительских свойств программного продукта. Общее направление реформы хозяйственного механизма и хотя и медленная, но все же ускоряющая обороты деятельность ГКВТИ СССР позволяют надеяться на формирование к концу пятилетки основных контуров экономики программирования.

Остается, однако, не менее трудная задача концентрации сверхрассеянной армии программистов в точках производства коммерческого программного продукта. Пожалуй, наиболее фундаментальная проблема — пробудить творческие силы программистов, направив их на создание общественно полезного продукта. Надо тактично, но непреклонно преодолевать пораженческую горечь «50-летних программистов», с которой они говорят об их неспособности включиться в работу по 4-му и 5-му поколениям ЭВМ. Не менее важно бороться с конформизмом и меркантилизмом некоторых молодых специалистов, эксплуатирующих дефицит профессии, но не рвущихся к повышению общезначимости своей работы.

И все-таки есть серьезные основания для оптимизма.

Старшее поколение программистов сыграет свою стабилизирующую роль в перестройке, привнеся в общественное сознание огромный и далеко не бесполезный опыт, а также романтическое восприятие профессии, сложившееся в 50-е и 60-е годы.

Что до молодежи, то мы очень скоро начнем вкушать плоды фронтальной компьютеризации и информатизации школы, которая уже сейчас имеет у себя больше компьютеров, чем имела вся страна пять лет назад.

Достаточно зайти на вечер в любой из компьютерных клубов для того, чтобы оценить безграничность энтузиазма и пытливой энергии армии школьников, которая завтра вторгнется в студенческие аудитории и на рабочие места общественного производства.

А. П. Ершов

УДК 621.3.049.771.14.001.2 : 681.3

В. С. Свиридович, Т. Г. Черноусова, Б. Н. Чернуха, В. А. Бобков

КОНТРОЛЛЕР ПРЕРЫВАНИЙ К588ВН1

(Продолжение цикла статей по расширенному комплекту серии К588. См. № 1, 1987 г.)

Микросхема контроллера прерываний (КПРВ) К588ВН1 обеспечивает прерывания центрального процессора (ЦП) в соответствии с требованиями интерфейса микроЭВМ «Электроника 60». Предназначена для обслуживания запросов на прерывание от двух внешних устройств (ВУ). Выполнена на основе КМДП-технологии с самосовмещенным поликремниевым затвором в 28-выводном металлокерамическом корпусе типа 4119.28-3-02 (рис. 1). Назначение выводов микро-

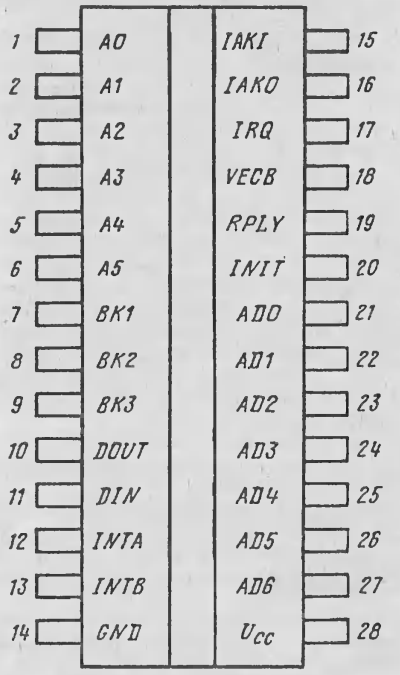


Рис. 1. Условное графическое обозначение микросхемы К588ВН1

схемы К588ВН1 приведено в таблице, ниже даны основные характеристики контроллера.

Напряжение питания U_{CC}, В 5±10 %
Ток потребления I_{CC}, мкА . . 200

Выходное напряжение, В:
низкого уровня U_{OL} . . . 0,4
высокого уровня U_{OH} . . . —0,4

Выходной ток, мА:
низкого уровня I_{OL} . . . 0,8
высокого уровня I_{OH} . . . —0,4

Входное напряжение, В:
низкого уровня U_{IL} 0,8
высокого уровня U_{IH} . . . —0,8

Время записи:
$t_{WR\ (RPLY-DOUT)}$, нс . . . 250

Время чтения:
$t_{RD\ (RPLY-DIN)}$, нс . . . 250

Время реакции на прерывание:
$t_{RC\ (IRQ-INT)}$, нс . . . 300

В составе микропроцессорного комплекта серии К588 микросхема К588ВН1 может использоваться автономно (рис. 2) или совместно с микросхемой селектора адреса (СА) К588ВТ1 (рис. 3).

В вычислительных системах, насчитывающих несколько контроллеров

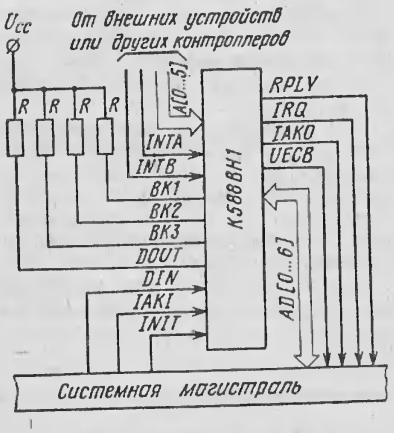


Рис. 2. Схема включения контроллера К588ВН1 в автономном режиме

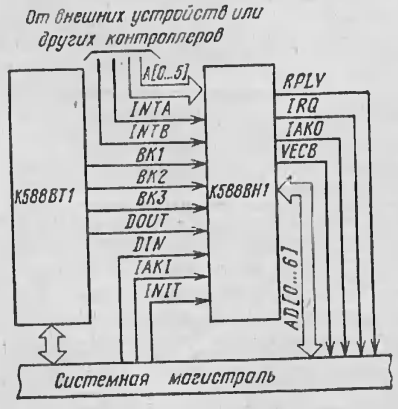


Рис. 3. Схема включения контроллера К588ВН1 и селектора адреса 588ВТ1

прерываний К588ВН1, наиболее высоким приоритетом обладает та микросхема, которая расположена ближе к ЦП на линии предоставления прерывания.

Назначение выводов микросхемы К588ВН1

Вывод	Обозначение	Назначение
1...5	A0 ... A5	Входы адреса вектора прерывания
7...9	BK1 ... BK3	Входы выбора регистра состояния и внешних устройств
10	DOUT	Запись данных
11	DIN	Чтение данных
12	INTA	Запрос на обслуживание прерывания от внешнего устройства А
13	INTB	Запрос на обслуживание прерывания от внешнего устройства В
14	GND	Общий
15	IAKI	Вход предоставления прерывания
16	IAKO	Выход предоставления обслуживания следующему устройству
17	IRQ	Выход требования прерывания
18	VECB	Выход обслуживания прерывания от внешнего устройства В
19	RPLY	Выход ответа данных
20	INIT	Вход начальной установки
21...27	AD0 ... AD6	Входы-выходы данных
28	U_{cc}	Напряжение питания

Контроллер К588ВН1 состоит (рис. 4) из 7-разрядного регистра состоя-

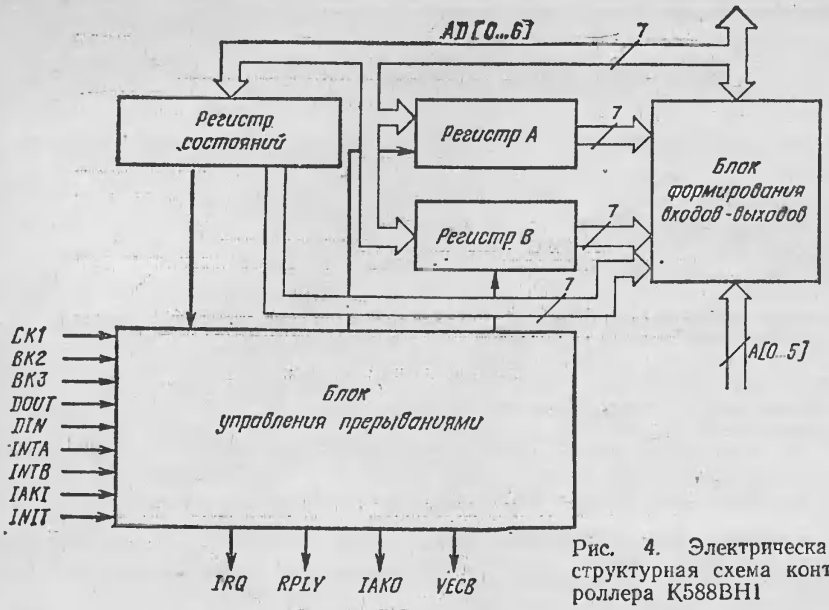


Рис. 4. Электрическая структурная схема контроллера К588ВН1

устройства А, 7-разрядного регистра внешнего устройства В, блока управления прерываниями, блока формирования входов-выходов.

Разряды регистра состояния имеют следующее назначение:

РС0 определяет режим выдачи адреса вектора прерывания ВУ А. При РС0=1 выдача адреса вектора прерывания ВУ А в системную магистраль производится из регистра А. При РС0=0 адрес вектора прерывания ВУ А выдается в системную магистраль с шин А [0...5].

РС1 задает режим выдачи адреса вектора прерывания ВУ В. При РС1=1 выдача адреса вектора прерывания ВУ В в системную магистраль производится из регистра В. При РС1=0 адрес вектора прерывания ВУ В выдается в системную магистраль с шин А [0...5].

РС2 обеспечивает приоритетность обслуживания ВУ. При РС2=1 высший приоритет имеет ВУ В. При РС2=0 высший приоритет имеет ВУ А.

РС3 предназначен для маскирования запроса на обслуживание прерывания от ВУ А. При РС3=1 маскируется запрос на обслуживание прерывания от ВУ А. При РС3=0 разрешается обслуживание запроса на прерывание от ВУ А.

РС4 используется для маскирования запроса на обслуживание прерывания от ВУ В. При РС4=1 маскируется запрос на обслуживание прерывания от ВУ В. При РС4=0 разрешается обслуживание запроса на прерывание ВУ В.

По сигналу начальной установки (INIT) разряды РС [0...4] устанавливаются в нуль. Если запрос на обслуживание прерывания от ВУ А или В пришел в то время, когда РС3=1 или РС4=1, то после очистки этих разрядов будет обслужен соответствующий запрос на прерывание.

РС5 осуществляет начальную установку микросхемы. При РС5=1 значения разрядов регистра состояния не изменяются. После установки микросхемы в начальное состояние производится автоматический сброс РС5 в нуль.

РС6 устанавливает признак, показывающий, от какого ВУ было обслужено прерывание. При РС6=1 последним обслуживался запрос на прерывание от ВУ В. При РС6=0 последним обслуживался запрос на прерывание от ВУ А. По сигналу INIT значение РС6 не изменяется. По сигналу «Общий сброс», поступающему из ЦП, производится начальная установка КПРВ.

При автономном использовании микросхемы выдача адреса вектора прерывания происходит только с шин А [0...5], а наиболее приоритетным при обслуживании запросов на прерывание ЦП от ВУ является ВУ А.

Если КПРВ К588ВН1 применяется совместно с СА К588ВТ1, то после загрузки регистров микросхема выполняет обслуживание запросов на прерывание ЦП от двух ВУ в соответствии с режимом, установленным регистром состояния. При этом микросхема СА осуществляет дешифрацию адресов системной магистрали и при обращении ЦП к регистрам КПРВ управляет чтением и записью данных регистров.

В автономном режиме К588ВН1 выполняет цикл обслуживания запросов на прерывание от ВУ (рис. 5). Запрос на обслуживание прерывания от ВУ осуществляется по фронту входного сигнала INTA (INTB) при переходе из высокого уровня в низкий.

Получив сигнал INTA (INTB), микросхема выдает в ЦП сигнал требования прерывания ТПР. При получении IRQ ЦП выдает КПРВ сигнал предоставления прерывания IAKI, затем происходит сброс сигнала IRQ и запрещение прохождения выходного сигнала IAKO.

Если сигнал IRQ был выдан другим КПРВ К588ВН1, то по сигналу IAKI на данной микросхеме на выводе IAKO появится низкий уровень сигнала, который поступает на вход IAKI следующего КПРВ, т. е. производится поочередный опрос всех КПРВ. Контроллер КПРВ, запросивший прерывание, запретит распространение этого сигнала.

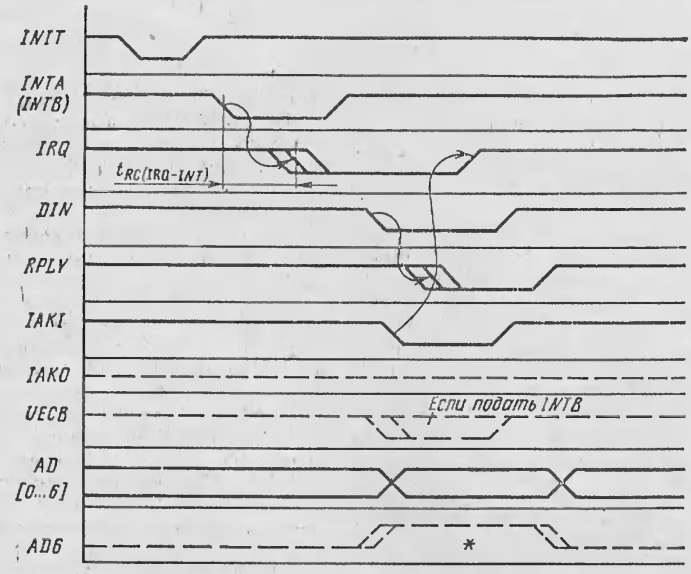


Рис. 5. Временная диаграмма работы микросхемы К588ВН1 при обслуживании прерывания от внешнего устройства

По сигналу DIN, поступающему из ЦП, микросхема К588ВН1 выдает на шину данных AD [0...6] адрес вектора прерывания, а в ЦП сигнал RPLY, информирующий о том, что адрес вектора прерывания выдан на шину данных.

Если во время обслуживания запроса на прерывание от первого ВУ пришел запрос на обслуживание прерывания от второго, то КПРВ перейдет к обслуживанию запроса от второго ВУ после обслуживания первого.

Если до сигнала IAKI поступают одновременно два сигнала запроса на обслуживание прерывания от ВУ А и В, то они будут обслужены согласно приоритету.

Если контроллер используется в сочетании с СА, то обращение к нему осуществляется в следующих циклах интерфейса микроЭВМ «Электроника 60»: запись (рис. 6), чтение (рис. 7), запись — модификация — чтение, обслуживание запроса на прерывание от ВУ.

В режиме записи при выдаче одного из сигналов ВК1, ВК2, ВК3 и сигнала записи DOUT микросхема осуществляет прием данных с шин А [0...6] в соответствующий регистр, а также выдает в ЦП сигнал RPLY, информирующий о том, что данные приняты в регистр с шин AD [0...6].

В режиме чтения при наличии одного из сигналов ВК1, ВК2, ВК3 и сигнала чтения DIN, контроллер выдает в системную магистраль данные регистра состояния или адрес вектора прерывания соответствующего ВУ и сигнал RPLY.

В соответствии с требованиями интерфейса микроЭВМ «Электроника 60» временная диаграмма работы КПРВ К588ВН1 при выполнении

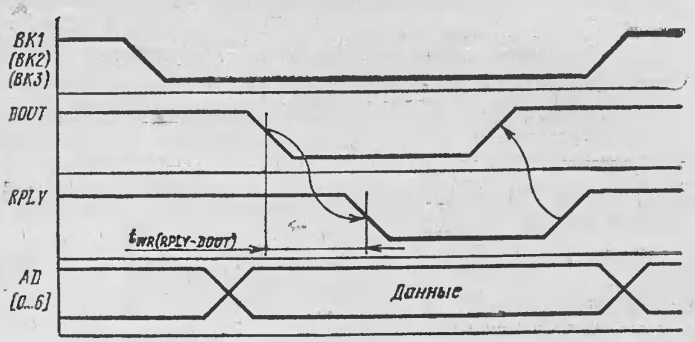


Рис. 6. Режим записи

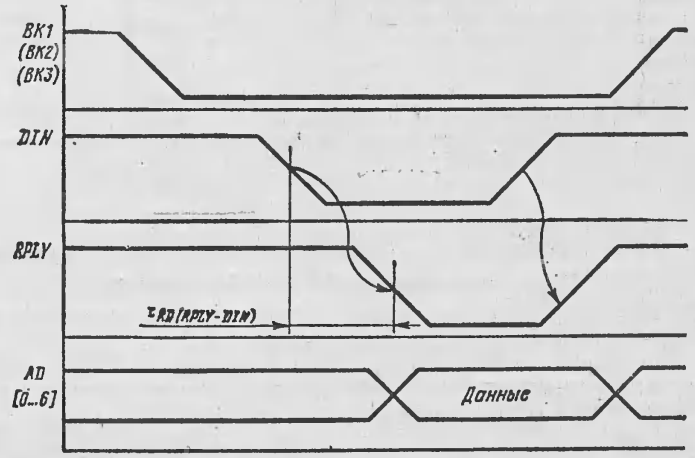


Рис. 7. Режим чтения

цикла запись — модификация — чтение состоит из циклов записи и чтения.

При совместном использовании контроллера и селектора адреса обслуживание запроса на прерывание от ВУ происходит аналогично работе в автономном режиме.

Статья поступила 23 апреля 1987 г.

УДК 621.3.049.77 : 681.3.06

С. А. Морозов, Д. М. Барановский, Л. К. Минкин, А. П. Семикин, А. Д. Черкай

ОДНОКРИСТАЛЬНЫЕ ЭВМ СЕРИИ КБ1013

Однокристальные 4-разрядные ЭВМ КБ1013ВК1-2, КБ1013ВК4-2 представляют собой функционально законченные устройства, изготовленные по КМОП-технологии с алюминиевыми затворами на основе метода стандартных элементов. ЭВМ имеют архитектуру гарвардского типа, вертикальное последовательное микропрограммное управление с естественной адресацией и содержат на кристалле: центральный процессор, ОЗУ данных и ПЗУ программ, таймер-счетчик, контроллер жидкокристаллических дисплеев (ЖКД), входной буфер и выходной регистр данных, формирователь внешних прерываний, логику сброса, блок управления резервированием мощности, тактовый генератор, устройство синхронизации. Микросхемы серии КБ1013 имеют идентичную структуру и отличаются объемами внутренней памяти программ, внутреннего ОЗУ, памяти «изображения», организацией контроллера ЖКД и выходного регистра данных, числом источников питающих напряжений (табл. 1).

Условное графическое обозначение микросхем приведено на рис. 1 и 2. Назначение выводов показано в табл. 2 и 3.

Структурная организация

Структурные схемы ОЭВМ приведены на рис. 3 и 4. Четырехразрядное арифметическое логическое устройство (АЛУ) обеспечивает выполнение арифметических и логических функций и занесение их результатов в аккумулятор (рис. 5).

А-мультиплексор выбирает данные из ОЗУ или из регистра слов ОЗУ DPL, которые запоминаются в Acc или поступают на вход сумматора-компаратора.

Б-мультиплексор выбирает данные из А-мультиплексора, сумматора-компаратора или входного буфера данных; выбранные данные запоминаются в Acc.

Сумматор-компаратор — это устройство комбинационного типа со сквозным последовательным переносом, осуществляющее логические и арифметические операции над данными из двух источников: А-мультиплексора и Acc.

Триггер переноса является многофункциональным устройством. Его можно программно устанавливать или сбрасывать, а также запоминать

Таблица 1

Основные параметры микросхем серии КБ1013

Параметр	Микросхема	
	КБ1013ВК1-2	КБ1013ВК4-2
Объем памяти программ (ПЗУ), байт	1827	2772
Объем памяти данных (ОЗУ), полубайт	65	96
Объем памяти «изображений», полубайт	18	32
Организация контроллера ЖКД:		
число выводов сегментов	36	32
число выводов общих электродов	2	4
временна́я мультиплексия	1:2	1:4
разрядность регистра вывода, бит	4	8
последовательный регистр вывода	Нет	Есть
дешифратор сегментного кода	Есть	Нет
число источников питающих напряжений	2	1
Число команд ассемблера	58	53

Таблица 2

Назначение выводов ОЭВМ КБ1013ВК1-2

Вывод	Обозначение	Назначение
60, 1 ... 3	O_{48}/K3 ... O_{18}/K0	Вывод сегментной группы / двунаправленный порт старшей тетрады команд
56 ... 59	O_{47}/K7 ... O_{17}/K3	Вывод сегментной группы / двунаправленный порт младшей тетрады команд
4	TEST0	Тестовый вход 0
5	TEST1	Тестовый вход 1
6	OSC_{out}	Выход тактового генератора
7	OSC_{in}	Вход тактового генератора
9 ... 11, 17	R_{3-0}	Выводной порт данных общего назначения
12	U_{CC}	Напряжение источника питания —3 В ±10 %
13	U_{CC1}	Напряжение источника питания —1,5 В ±10 %
14, 15	COM0, COM1	Выводы общих электродов ЖКД
18	CLR	Сброс
19	INT0	Флаг внешнего прерывания
20	INT1	Флаг внешнего прерывания
21	GND	Общий
22, 23, 25, 26	D0 ... D3	Вводной буфер данных общего назначения
27 ... 30	O_{40} ... O_{10}	Выводы сегментных групп
31 ... 34	O_{41} ... O_{11}	
35 ... 38	O_{42} ... O_{12}	
39 ... 42	O_{43} ... O_{13}	
43 ... 46	O_{44} ... O_{14}	
47 ... 50	O_{45} ... O_{15}	
51, 53 ... 55	O_{46} ... O_{16}	

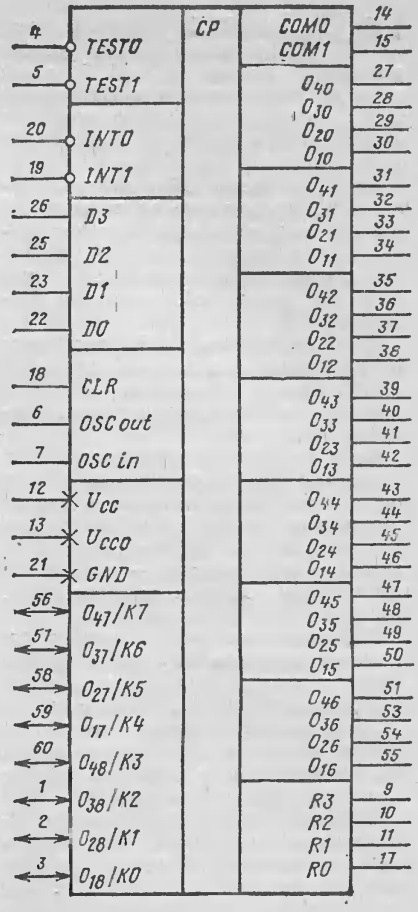


Рис. 1. Условное графическое обозначение ОЭВМ КБ1013ВК1-2

в нем значение переноса из старшего разряда и проверять на наличие нуля.

Устройство выборки микрокоманд (УВМ) управляет адресацией микропрограммного ПЗУ. В состав УВМ входят буферы адреса страниц и слов, счетчик команд, регистры возврата из подпрограмм, блок управления адресацией, дешифраторы слов и страниц.

Выполнение текущей и формирование адреса следующей команды происходит в течение одного машинного цикла. Поля распределения памяти программ приведены на рис. 6, 7. ПЗУ ОЭВМ КБ1013ВК4-2 состоит из 44 страниц по 63 слова каждая, а ПЗУ КБ1013ВК1-2 из 29 страниц также по 63 слова в каждой. Для адресации слов используются 6 младших разрядов счетчика команд. Если выполняемая команда не является командой передачи управления, то содержимое шести младших разрядов счетчика команд увеличивается на единицу. Полную замену содержимого программного счетчика выполняют команды JMP, CZP, CAL. Разница между ними состоит в том, что выполнение команды JMP не со-

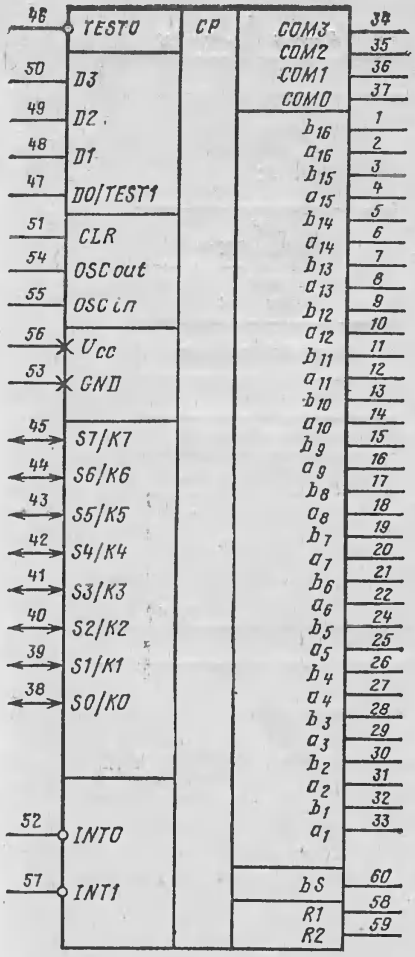


Рис. 2. Условное графическое обозначение ОЭВМ КБ1013ВК4-2

Таблица 3

Назначение выводов ОЭВМ КБ1013ВК4-2

Вывод	Обозначение	Назначение
1 ... 22, 24 ... 33	b_{16}, a_{10} ... b_1, a_1	Сегментные выводы ЖКД
34 ... 37	COM3/Acc3...COM0/Acc0	Выводы общих электродов ЖКД / выводы аккумуляторного регистра
38 ... 45	S0/K0 ...S7/K7	Выводной программируемый порт общего назначения / двунаправленный порт команд
46	TEST0	Тестовый вход 0
47 ... 50	D0/TEST1, D1 ... D3	Входной буфер данных общего назначения
51	CLR	Сброс
52	INT0	Флаг внешнего прерывания 0
53	GND	Общий
54	OSC_{out}	Выход тактового генератора
55	OSC_{in}	Вход тактового генератора
56	Ucc	Напряжение источника питания —3 В ±10 %
57	INT1	Флаг внешнего прерывания 1
58, 59	R1, R2	Выводы управления пьезоэлементом
60	bS	8-разрядный последовательный порт вывода

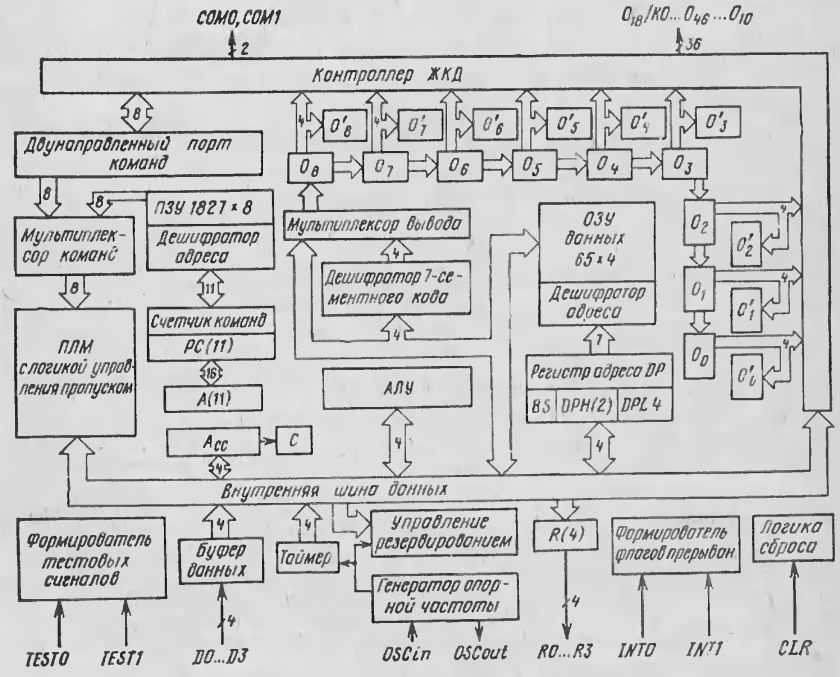


Рис. 3. Структурная схема ОЭВМ КБ1013ВК1-2

провождается записью в регистры возврата из подпрограммы.

Команды RT, RTS, определяющие возврат из подпрограммы, обеспечивают запись из регистра возврата в счетчик команд. Если возврат осуществляется по команде RTS, то следующая за ней команда пропускается.

Устройство считывания-записи данных (УС/ЗД) управляет адресацией запоминающего устройства ОЭВМ в режимах считывания и записи 4-разрядных данных.

В состав УС/ЗД входят мультиплексор записи, регистры слов (DPL) и страниц (BS, DPH), дешифраторы слов и страниц.

Запись данных в ОЗУ осуществляется через мультиплексор записи и дешифратор страниц из Acc и с шины команд. Данные из Acc заносятся непосредственно во все разряды адресуемого 4-разрядного регистра. Данные с шины команд позволяют модифицировать информацию лишь в одном из четырех битов адресуемого регистра. Считываемые данные из ОЗУ поступают через дешифратор страниц в АЛУ. Регистр слов используется не только для адресации ОЗУ, но и в качестве регистра общего назначения, содержи-

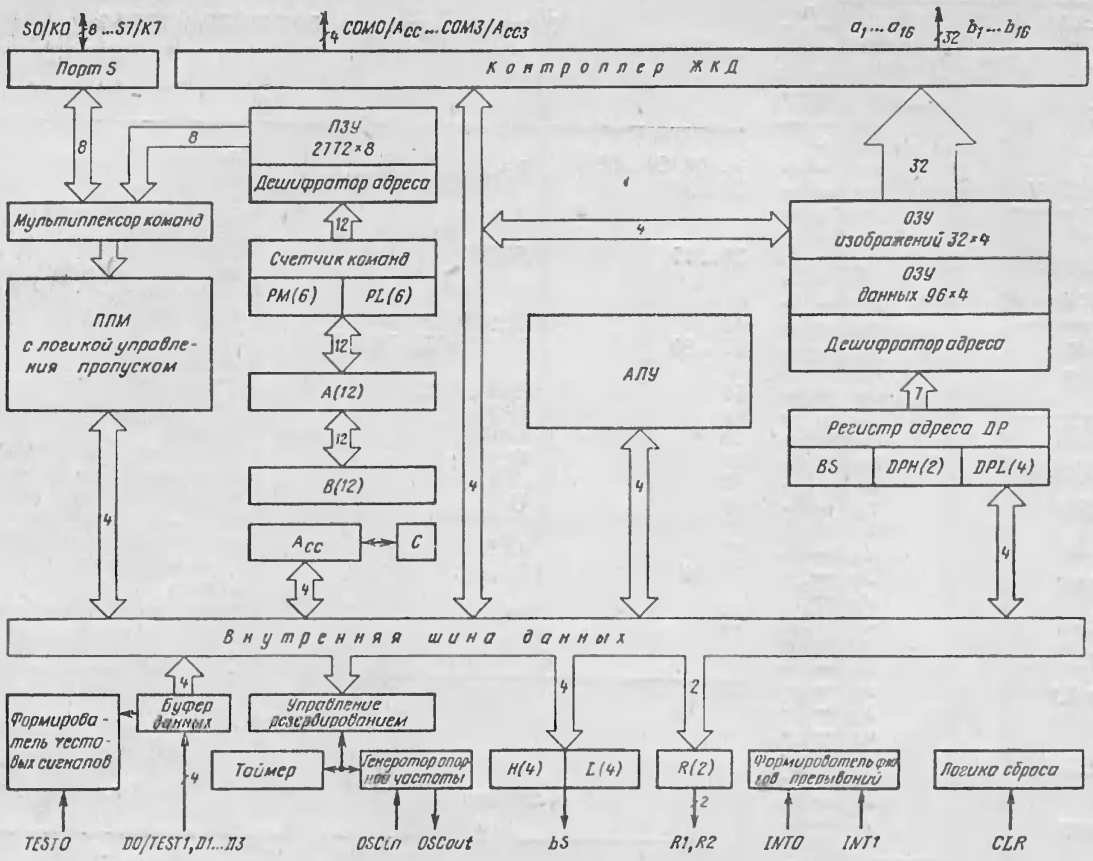


Рис. 4. Структурная схема ОЭВМ КБ1013ВК4-2

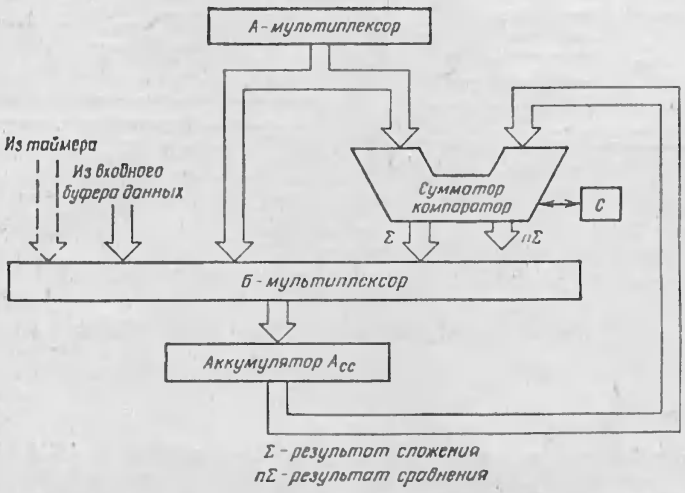


Рис. 5. Организация АЛУ ОЭВМ

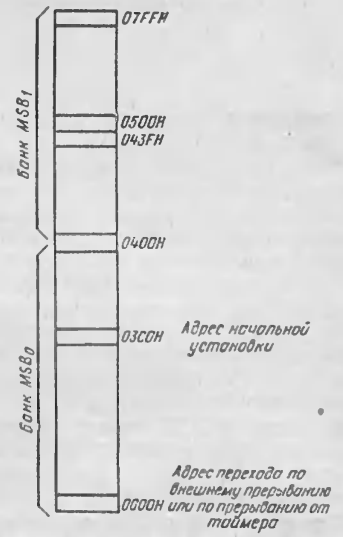


Рис. 6. Поле распределения памяти программ ОЭВМ КБ1013ВК1-2; адреса 0440H ... 047FH, 048OH ... ... 048FH, 04COH ... 04FFH не используются

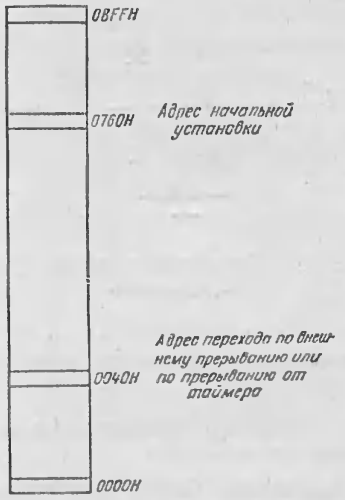


Рис. 7. Поле распределения памяти программ ОЭВМ КБ1013ВК4-2; адреса 000ОН ... 003FH — таблица переходов к подпрограммам

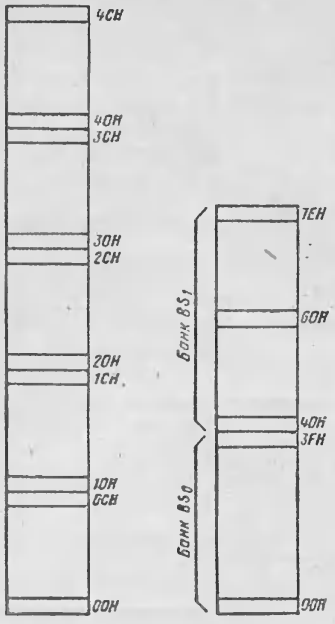


Рис. 8. Поле распределения памяти данных ОЭВМ КБ1013ВК1-2; адреса 0DH ... 0FH, 1DH ... 1FH, 2DH ... ...2FH, 3DH ... 3FH, 40H ... 7FH не используются

Рис. 9. Поле распределения памяти данных ОЭВМ КБ1013ВК4-2, адреса 6ОН ... 7FH — адресное пространство для «памяти изображений»

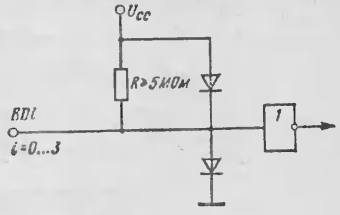


Рис. 10. Организация входного буфера данных

мое которого можно обменять на содержимое Acc, а также декрементировать и инкрементировать. Поля распределения памяти данных приведены на рис. 8, 9. Адресное пространство имеет страничную организацию. ОЗУ ОЭВМ КБ1013ВК4-2 состоит из восьми страниц по 16 слов в каждой, а ОЗУ КБ1013ВК1-2 — из пяти страниц по 13 слов в каждой. Адрес выбранного слова запоминается в 4-разрядном DPL-регистре, адрес страницы — в триггере банка BS и 2-разрядном регистре страниц DPH. Эти регистры образуют регистр адреса ОЗУ DP.

Таймер состоит из 14 последовательно включенных каскадов делителя частоты и селектора частоты. На вход первого каскада поступает тактовая частота 16384 Гц (при подключении внешнего генератора с частотой $F_т = 32\,768$ Гц). С выхода 14-го каскада снимается частота 1 Гц. Переключение разряда этого каскада фиксируется в триггере-защелке, который можно программно обрабатывать. Для ОЭВМ КБ1013ВК1-2 четыре старших разряда таймера (T11—T14) могут загружаться по команде LDF в Acc для дальнейшей обработки. В ОЭВМ КБ1013ВК4-2 разряды таймера T14, T11 по командам S11, S14 могут программно опрашиваться.

Устройство управления режимом малой потребляемой мощности осуществляет функцию резервирования, которая организована очень экономично с точки зрения потребления электроэнергии. Под управлением прикладной программы включается режим резервирования (HALT), в котором логические сигналы, тактирующие основные узлы микроЭВМ, «останавливаются».

В ОЗУ сохраняется записанная информация. В этом режиме потребление от источника питания Icc составляет 1/5 от обычной величины. Выход из состояния резервирования осуществляется по прерыванию от таймера или входного буфера данных. Поскольку в режиме HALT функционирование может быть возобновлено по сигналам прерывания, этот режим используется для временной приостановки работы программы. Следовательно, путем чередования обычного функционирования с режимом HALT можно достичь снижения потребления энергии всей системой. Так, при чередовании этих режимов во временно́м соотношении 1 : 4 будет получено общее снижение тока потребления в 3 раза по сравнению с непрерывным функционированием в обычном режиме.

На рис. 10 показан один из четырех разрядов входного логического буфера данных; каждый вход имеет резистор 5 МОм, подключенный к Ucc. Информацию из буфера данных по команде ICD можно передать в Acc для дальнейшей обработки.

Контроллер жидкокристаллических дисплеев реализует временную диаграмму управления ЖКД, работающего с мультиплексией 1 : 2 для ОЭВМ КБ1013ВК1-2, 1 : 4 — для ОЭВМ КБ1013ВК4-2. В состав контроллера ЖКД входят формирователи сигналов общих электродов, формирователи сегментного кода, схема управления.

Схема управления осуществляет автоматическую синхронизацию сигналов общих электродов и сегментного кода (рис. 11, 12). Сегмент считается «выбранным» при разности напряжений между сегментом и общим электродом, равной 2,6...3,3 В и частоте управляющего напряжения в диапазоне 30...100 Гц. Управление переменным напряжением повышает срок службы дисплея, так как уменьшается миграция примесей в жидком кристалле на общие электроды. Задающий кварцевый тактовый генератор с внешней частотозадающей цепью работает на частоте параллельного резонанса кварцевого генератора, номинальное значение которого 32 768 Гц. Тактовый генератор служит для выработки опорных колебаний $F = 16\,384$ Гц, которые необходимы для формирования фазовых тактирующих импульсов, используемых во всех устройствах ОЭВМ. На выводы OSC_{in}, OSC_{out} можно подавать также сигналы от внешнего источника тактовых сигналов. Период опорной частоты соответствует машинному циклу ОЭВМ, равному 61 мкс.

В ОЭВМ предусмотрены два флага внешних прерываний: INT0, INT1, которые проверяются командами S10, S11. Организация входов флагов прерывания приведена на рис. 13.

Дешифратор сегментного кода содержит (рис. 14): банк 0, банк 1, формирователь младшего разряда банка 0, триггер выбора банка, мультиплексор. Дешифратор преобразует шестнадцатеричный код из Acc в сегментный для управления ЖКД. Прошивка дешифратора приведена в табл. 4. Формирователь младшего разряда банка 0 управляет «точкой» 8-сегментного индикатора. Мультиплексор дешифратора подключает ко входу мультиплексора вывода один из двух банков. В ОЭВМ КБ1013ВК4-2

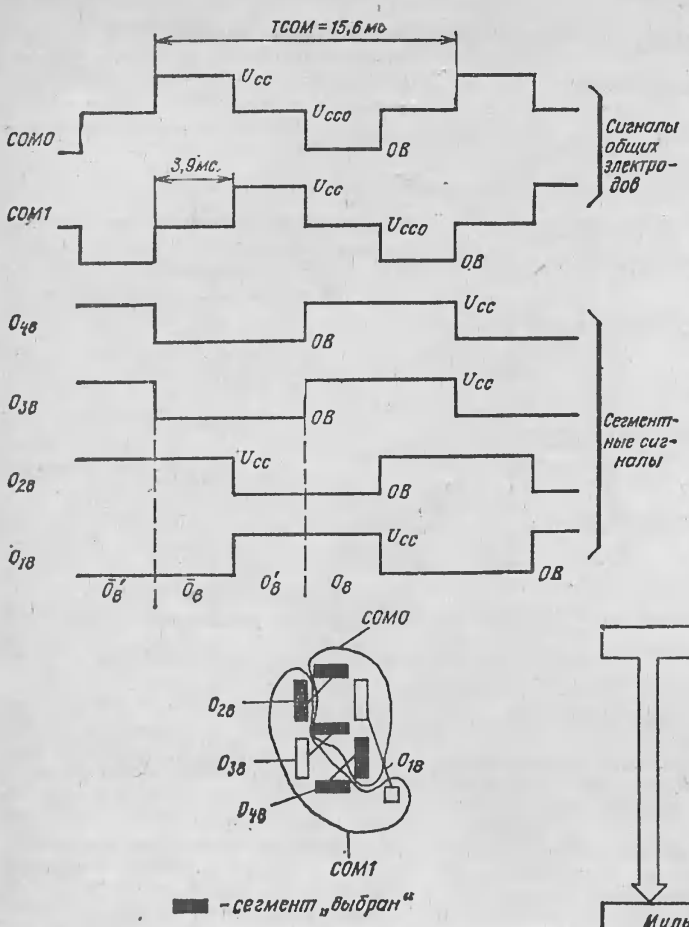


Рис. 11. Временные диаграммы формирования цифры «5» на 8-сегментном индикаторе для ОЭВМ КБ1013ВК1-2

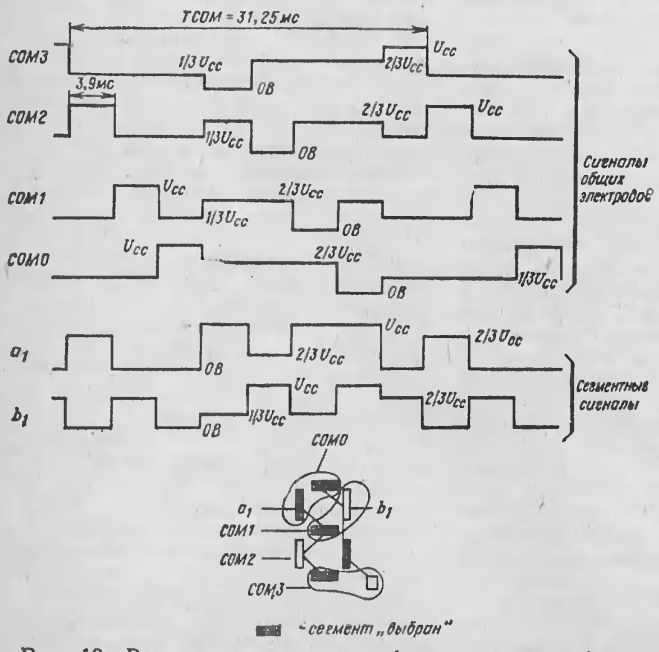


Рис. 12. Временные диаграммы формирования цифры «5» на 8-сегментном индикаторе ОЭВМ КБ1013ВК4-2

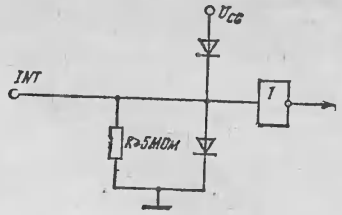


Рис. 13. Организация входа флага прерывания

дешифрация сегментного кода осуществляется программно.

Конструктивные особенности и электрические параметры

Микросхемы ОЭВМ выполнены в 60-выводных планарных пластмассо-

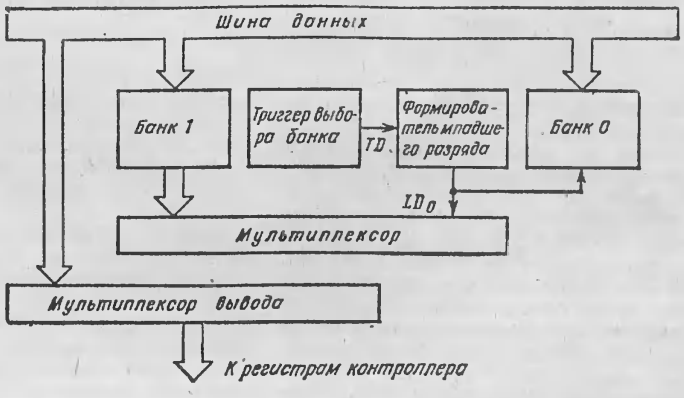


Рис. 14. Организация дешифратора 7-сегментного кода

вых корпусах. Типовое включение их приведено на рис. 15 и 16. Включение конденсатора между общим «+» и выводом CLR обеспечивает задержку нарастания отрицательного напряжения на этом выводе при подаче питания, что необходимо для начальной установки счетчика команд на пусковой адрес программы. Начальная установка ОЭВМ производится также нажатием кнопки CLR. Ячейки ОЗУ очищаются только программным путем.

Электрические характеристики ОЭВМ и условия эксплуатации микроЭВМ приведены ниже.

	мин.	макс.
Входное напряжение, В:		
низкого уровня	U_{CC}	$0{,}9U_{CC}$
высокого уровня	$0{,}1U_{CC}$	0
Выходное напряжение, В:		
низкого уровня	—	$U_{CC}+0{,}2$
высокого уровня	−0,2	—
Время машинного цикла, мкс	61	61
Ток потребления, мкА	—	60
Диапазон рабочих температур, °С	−10	+55
Напряжение питания, В	−3,3	−2,7

Таблица 4

Прошивка дешифратора 7-сегментного кода

Acc	Банк 1	Банк 0	
		$LD_0 = 0$	$LD_0 = 1$
0	B	F	E
1	9	1	0
2	7	Д	C
3	F	9	8
4	Д	3	2
5	E	B	A
6	E	F	E
7	B	3	2
8	F	F	E
9	F	B	A
A	4	1	0
Б	0	1	0
C	Д	3	2
Д	E	B	A
E	4	3	2
F	0	3	2

Вывод в порт O′	Вывод в порт O

Система команд и средства разработки прикладных программ

Система команд содержит 58 базовых команд для ОЭВМ КБ1013ВК1-2 и 53 — для КБ1013ВК4-2 (табл. 5). Машинные команды разделены на 8 форматов (табл. 6). Декодирование кодов команд и формирование условий пропуска осуществляется с помощью программируемого логического массива. При выработке условий учитываются признаки, характерные для различных форматов команд (двух- или однобайтовых), и признаки, полученные в результате выполнения предыдущей команды.

Для ОЭВМ серии КБ1013 разработаны следующие программы отладки прикладного программного обеспечения.

Кроссассемблер, осуществляющий кроме непосредственного перевода программ, написанных на языке ассемблера ОЭВМ, в язык объектных кодов, проверку данных на допустимость (с соответствующей диагностикой) обнаруженных ошибок. Это облегчает исправление ошибок в программе и предоставляет возможность отслеживать логику ее выполнения на программно-логической модели.

Дизассемблер для преобразования объектных кодов в текстовой файл прикладной программы на языке ассемблера ОЭВМ.

Программно-логическая модель — интерактивный отладчик для программ, написанных на языке ассемблера — позволяет программисту уп-

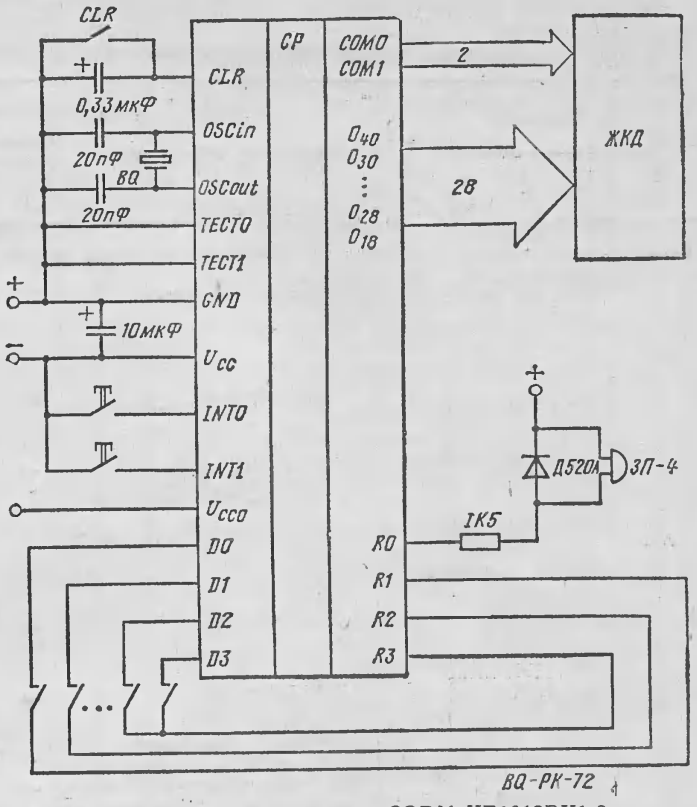


Рис. 15. Типовое включение ОЭВМ КБ1013ВК1-2

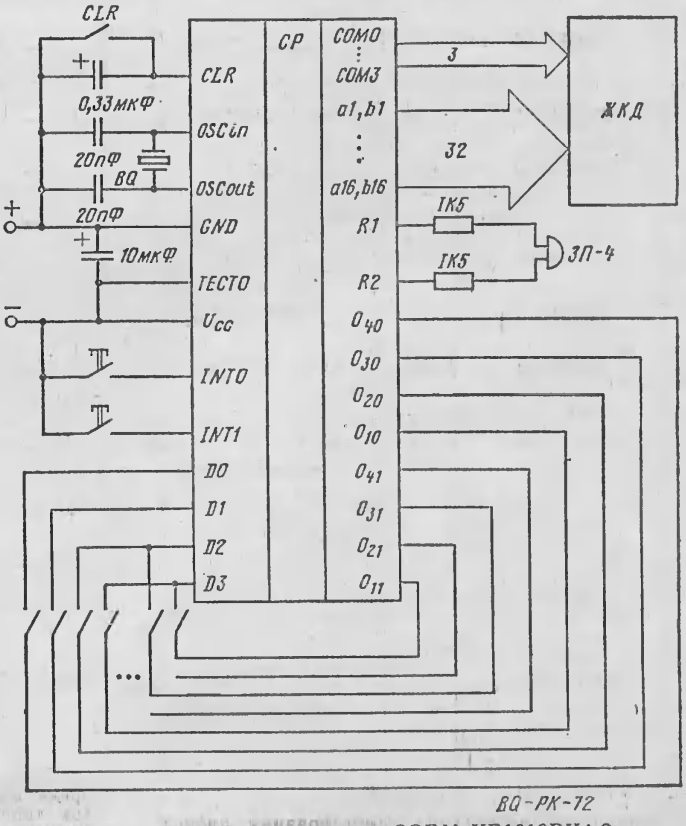


Рис. 16. Типовое включение ОЭВМ КБ1013ВК4-2

Таблица 5

Система команд однокристальных ЭВМ КБ1013ВК1-2 и КБ1013ВК4-2

Мнемоника	Код операции К7...К0	Формат	Перемещение информации	Условие пропуска	Комментарий	КБ1013ВК1-2	КБ1013ВК4-2
			Команды загрузки и обмена				
LC	$0010C_3C_2C_1C_0$	II	$C_{3-0} \rightarrow Acc_{3-0}$	—	Загрузка аккумулятора четырьмя битами данных; выполнение следующей LC-инструкции как NOP-инструкции	+	+
LM	00011000	IV	ОЗУ(DP) → Acc	—	Загрузка аккумулятора содержимым ОЗУ, адресуемым DP-регистром	+	+
LE	$000110B_1B_0$	III	ОЗУ(DP) → Acc DPH⊕B→DPH	—	Загрузка аккумулятора содержимым ОЗУ, адресуемым DP-регистром; Выполнение «исключающего ИЛИ» между DPH и двумя битами непосредственных данных	+	+
LAF	01011111 $0Y_6Y_5Y_4Y_3Y_2Y_1Y_0$	V	Y → DP	—	Загрузка DP-регистра семью битами непосредственных данных	+	+
LAS	$0100C_3C_2C_1C_0$	II	O → BS $C_{1,0} \rightarrow$ DPH $C_3 \div C_2$, 0, C_3, $C_2 \rightarrow$ $\rightarrow DPL_{3-0}$	—	Загрузка DP-регистра четырьмя битами непосредственных данных, сброс триггера BS выбора банка ОЗУ	+	—
			BS → BS, $C_{1,0} \rightarrow$ DPH $C_3 \div C_2$, $C_3 \div C_2$, C_3, $C_2 \rightarrow$ $\rightarrow DPL_{3-0}$	—	Загрузка DP-регистра четырьмя битами непосредственных данных; сохранение триггера BS выбора банка ОЗУ	—	+
LDF	01011110	IV	$T_{14-11} \rightarrow$ Acc	—	Загрузка в аккумулятор четырех старших разрядов таймера	+	—
BS0	01101011	IV	0→BS	—	Сброс триггера выбора банка ОЗУ	+	—
BS1	00000010	IV	1→BS	—	Установка триггера выбора банка ОЗУ	+	—
				—	Установка триггера выбора банка ОЗУ на время одного следующего машинного цикла	—	+
XL	00001011	IV	DPL↔Acc	—	Обмен аккумулятора с DPL-регистром	+	+
XM	00010000	IV	ОЗУ (DP)↔Acc	—	Обмен аккумулятора с ОЗУ, адресуемым DP-регистром	+	+
XI	00010100	IV	ОЗУ (DP)↔Acc DPL⊕1→DPL	DPL=7	Обмен аккумулятора с ОЗУ, адресуемым DP-регистром; пропуск следующей команды, если DPL=FH; инкрементирование DPL	+	—
				DPL=FH	Обмен аккумулятора с ОЗУ, адресуемым DP-регистром; пропуск следующей команды, если DPL=FH; инкрементирование DPL	—	+
XEI	$000101B_1B_0$	III	ОЗУ (DP)↔Acc DPL+1→DPL DPH⊕B→DPH	DPL=7	Обмен аккумулятора с ОЗУ, адресуемым DP-регистром; пропуск следующей команды если DPL=7; инкрементирование DPL; выполнение «исключающего ИЛИ» между DPH и двумя битами непосредственных данных	+	—

Мнемоника	Код операции К7...К0	Формат	Перемещение информации	Условие пропуска	Комментарий	КБ1013ВК1-2	КБ1013ВК4-2
				DPL=FH	Обмен аккумулятора с ОЗУ, адресуемым DP-регистром; пропуск следующей команды, если DPL=FH; инкрементирование DPL; выполнение «исключающего ИЛИ» между DPH и двумя битами непосредственных данных	—	+
XD	00011100	IV	ОЗУ (DP)↔Acc DPL—1→DPL	DPL=0	Обмен аккумулятора с ОЗУ, адресуемым DP-регистром; пропуск следующей команды, если DPL=0, декрементирование DPL	+	+
XED	000111B_1B_0	III	ОЗУ (DP)↔Acc DPL—1→DPL DPH+B→DPH	DPL=0	Обмен аккумулятора с ОЗУ, адресуемым DP-регистром, пропуск следующей команды, если DPL=0; выполнение «исключающего ИЛИ» между DPH и двумя битами непосредственных данных	+	+
XE	000100B_1B_0	III	ОЗУ (DP)↔Acc DPH⊕B→DPH	—	Обмен аккумулятора с ОЗУ, адресуемым DP-регистром, выполнение «исключающего ИЛИ» между DPH и двумя битами непосредственных данных	+	+
LH	01011001	IV	Acc→H	—	Загрузка аккумулятора в регистр H	—	+
LL	01100000	IV	Acc→L	—	Загрузка аккумулятора в регистр L	—	+
BM0	000001B_1B_0	III	0→ОЗУ(DP)$_{бит}$	—	Сброс одинарного бита регистра ОЗУ, адресуемого DP-регистром, обозначенного двумя битами непосредственных данных	+	+
BM1	000011B_1B_0	III	1→ОЗУ(DP)$_{бит}$	—	Установка одинарного бита регистра ОЗУ, адресуемого DP-регистром, обозначенного двумя битами непосредственных данных	+	+
SM1	010101B_1B_0	III	—	(DP)$_{бит}$=1 —1	Пропуск следующей команды, если одинарный бит регистра ОЗУ, адресуемого DP-регистром, обозначенный двумя битами непосредственных данных, установлен	+	+
			Арифметические и логические команды				
AM	00001000	IV	Acc+ОЗУ(DP)→Acc	—1	Сложение аккумулятора с содержимым ОЗУ, адресуемым DP-регистром	+	+
AC	00001001	IV	Acc+ОЗУ(DP)+C→ →Acc; $П_3$→C	$П_3$=1	Сложение аккумулятора с содержимым ОЗУ, адресуемым DP-регистром и с флагом переноса; установка переноса из старшего разряда в флаге переноса; пропуск следующей команды, если установился перенос из старшего разряда	+	+
A10	00111010	IV	Acc+10_{10}→Acc	—	Сложение аккумулятора с 10_{10} для коррекции двоично-десятичного вычитания	+	+

Мнемоника	Код операции К7 ... К0	Формат	Перемещение информации	Условие пропуска	Комментарий	КБ1013ВК1-2	КБ1013ВК4-2
AS	$0011C_3C_2C_1C_0$	II	$Acc_{3-0}+C_{3-0}\rightarrow Acc_{3-0}$, $C_{3-0}\neq 10_{10}$	$П_3=1$	Сложение аккумулятора с четырьмя битами непосредственных данных; пропуск следующей команды, если установился перенос из старшего разряда	+	+
RAC	01101011	IV	$\rightarrow Acc \rightarrow C \rightarrow$	—	Циклический сдвиг содержимого аккумулятора вправо через флаг переноса	—	+
CLL	01101000	IV	$0\rightarrow Acc$, $0\rightarrow LD_0$	—	Очистка аккумулятора и сброс младшего разряда 0 банка дешифратора сегментного кода	+	—
COM	00001010	IV	$\overline{Acc}\rightarrow Acc$	—	Формирование дополнения аккумулятора	+	+
CLC	01100110	IV	$0\rightarrow C$	—	Очистка флага переноса	+	+
STC	01100111	IV	$1\rightarrow C$	—	Установка флага переноса	+	+
SCO	01010010	IV	—	$C=0$	Пропуск следующей команды, если флаг переноса сброшен	+	+
SAO	01011010	IV	—	$Acc=0$	Пропуск следующей команды, если аккумулятор очищен	+	+
INC	01100100	IV	$DPL+1\rightarrow DPL$	$DPL=7$	Инкрементирование DPL; пропуск следующей команды, если DPL=7	+	—
				$DPL=FH$	Инкрементирование DPL; пропуск следующей команды, если DPL=FH	—	+
DEC	01101100	IV	$DPL-1\rightarrow DPL$	$DPL=0$	Декрементирование DPL; пропуск следующей команды, если DPL=0	+	+
SAM	01010011	IV	—	$Acc=(DP)$	Пропуск следующей команды, если Acc равен содержимому ОЗУ, адресуемому DP-регистром	+	+
SAL	01011011	IV	—	$Acc=DPL$	Пропуск следующей команды, если аккумулятор равен содержимому DPL	+	+
NOP	00000000	IV	—	—	Нет операции	+	+
			Команды ввода-вывода				
ICD	01101010	IV	$D\rightarrow Acc$	—	Ввод в аккумулятор инвертированной информации из входного буфера данных	+	+
OAR	00000001	IV	$Acc\rightarrow R$	—	Передача содержимого аккумулятора в выводной порт R	+	—
OS0	01100011	IV	$0\rightarrow S_0$, $S_{n-1}\rightarrow S_n$, где $n=1...7$	—	Сброс младшего разряда регистра порта вывода S, сдвиг содержимого регистра S влево на 1 разряд	—	+
OS1	01100010	IV	$1\rightarrow S_0$, $S_{n-1}\rightarrow S_n$, где $n=1...7$	—	Установка младшего разряда регистра порта вывода; сдвиг содержимого регистра влево на 1 разряд	—	+
OA0	01100011	VI	$Acc\rightarrow O_8$, $0\rightarrow O_{48}$, $O_4\rightarrow O_7\rightarrow ...\rightarrow O_0$	—	Вывод содержимого аккумулятора в регистр O_8 порта вывода O; сброс старшего разряда регистра O_8; параллельный сдвиг порта O вправо на четыре разряда	+	—

Мнемоника	Код операции К7...К0	Формат	Перемещение информации	Условие пропуска	Комментарий	КБ1013ВК1-2	КБ1013ВК4-2
OA1	01100010	IV	$Acc \rightarrow O_8$ $1 \rightarrow O_{48}$ $O_4 \rightarrow O_7 \rightarrow \ldots \rightarrow O_6$	—	Вывод содержимого аккумулятора в регистр O_8 порта вывода O; установка старшего разряда регистра O_8; параллельный сдвиг порта O вправо на четыре разряда	+	—
DAF	01011101	IV	$(Acc)_{DS} \rightarrow O_8$ $O_4 \rightarrow O_7 \rightarrow \ldots \rightarrow O_0$	—	Вывод в регистр O_8 порта вывода O дешифрованного значения содержимого аккумулятора; параллельный сдвиг порта O вправо на четыре разряда	+	—
DAS	01100001	IV	$(Acc)_{DS} \rightarrow O_8$ $O_8 \rightarrow O_7$	—	Вывод в регистр O_8 порта вывода O дешифрованного значения содержимого аккумулятора; передача содержимого регистра O_8 в регистр O_7	+	—
ABS	01011001	IV	$O_8 \rightarrow O_8'$ $O_7 \rightarrow O_7'$	—	Передача содержимого регистров O_8, O_7 порта вывода O соответственно в регистры O'_8, O'_7 порта вывода O′	+	—
ABF	01011100	IV	$O_n \rightarrow O'_n$, где n=0...8	—	Передача содержимого регистров порта O соответственно в регистры порта O′	+	—
CTB	01100000	IV	$TD \rightarrow TD$	—	Формирование дополнения триггера выбора банка дешифратора сегментного кода	+	—
LD0	01101001	IV	$\overline{TD} \rightarrow LD_0$	—	Передача значения триггера выбора банка дешифратора сегментного кода в младший разряд банка 0 дешифратора сегментного кода	+	—
EN	00000011	IV	$Acc_3 \rightarrow EN_3$ $Acc \rightarrow EN_0$	—	Передача старшего и младшего разрядов аккумулятора в контроллер ЖКД, разрешение вывода портов O, O′ на ЖКД, если $Acc_3=0$, $Acc_0=1$	+	—
BEL	01100001	IV	$Acc_0 \rightarrow B_1$ $Acc_0 \rightarrow B_0$	—	Передача двух младших разрядов аккумулятора в регистр управления звуковой сигнализацией, разрешение звуковой сигнализации при $Acc=1$, $Acc_0=0$	—	+
LCD	000000011	IV	$Acc_0 \rightarrow B_0$	—	Передача младшего разряда аккумулятора в контроллер ЖКД, включение контроллера ЖКД при $Acc_0=1$	+	+
LOP	011011011	IV	$C \rightarrow E$	—	Передача флага переноса в триггер управления резистивным делителем напряжения контроллера ЖКД; отключение делителя напряжения при C=0	+	—

Мнемоника	Код операции К7 ... К0	Формат	Перемещение информации	Условие пропуска	Комментарий	КБ1013ВК1-2	КБ1013ВК4-2
			Команды передачи управления				
BR	$10W_5W_4W_3W_2W_1W_0$	I	$W_5 - 0 \rightarrow PC_5 - 0$	—	Переход на адрес, определяемый шестью битами непосредственных данных, в пределах установленной страницы ПЗУ	+	+
JMP	$0111C_3C_2C_1C_0$ $10W_5W_4W_3W_2W_1W_0$	II, I	$C_3 - 0$, $W_5 - 0 \rightarrow PC_9 - 0$ $MSB \rightarrow PC_{10}$	—	Переход на адрес, определяемый десятью битами непосредственных данных в одном из двух банков ПЗУ, выбираемым старшим разрядом счетчика команд PC_{10}	+	—
	$0111a_{11}a_{10}a_9a_8$ $a_7a_6a_5a_4a_3a_2a_1a_0$	II	$a_{11} - 0 \rightarrow PC_{10} - 0$	—	Переход на адрес, определяемый двенадцатью битами непосредственных данных	—	+
LP	$0111C_3C_2C_1C_0$	II	$C_3 ... 0 \rightarrow PC_9 - 6$ $PC_5 - 0 + 1 \rightarrow PC_5 - 0$ $MSB \rightarrow PC_{10}$	—	Установка адреса страницы в одном из 2-х банков ПЗУ, выбираемых старшим разрядом счетчика команд	+	—
CZP	$11W_5W_4W_3W_2W_1W_0$	I	$010H \rightarrow PC_{10} - 0$ $W_5 - 0 \rightarrow PC_5 - 0$ $PC_{10} - 0 \rightarrow A_{10} - 0$	—	Вызов подпрограммы на странице 010H по адресу слова, определяемому шестью битами непосредственных данных; адрес возврата в регистре возврата A (11) сохраняется	+	—
	$11a_{13}a_{12}a_{11}a_{10}a_9a_8$ $a_7a_6a_5a_4a_3a_2a_1a_0$	VIII	$04H \rightarrow PC_{11} - 8$ $a_7 - 0 \rightarrow PC_7 - 0$ $A_{11} - 0 \rightarrow B_{11} - 0$ $PC_{11} - 0 + 1 \rightarrow A_{11} - 0$	—	Обращение к одной из ячеек страницы 000H по адресу слова, определяемому шестью битами непосредственных данных первого байта команды; переход к подпрограмме по адресу, определяемому восемью битами непосредственных данных второго байта команды; сохранение адреса возврата из подпрограммы в верхнем регистре стека возврата; передача старого значения верхнего регистра стека в нижний	—	+
CAL	$0111C_3C_2C_1C_0$ $11W_5W_4W_3W_2W_1W_0$	II, I	$C_3 - 0$, $W_5 - 0 \rightarrow PC_9 - 0$ $MSB \rightarrow PC_{10}$ $PC_{10} - 0 + 1 \rightarrow A_{10} - 0$	—	Переход к подпрограмме по адресу, определяемому десятью битами непосредственных данных в одном из 2 банков ПЗУ, выбираемым старшим разрядом счетчика команд PC_{10}; сохранение адреса возврата в регистре возврата A (11)	+	—
	$0111111a_9a_8$ $a_7a_6a_5a_4a_3a_2a_1a_0$	VII	$0 \rightarrow PC_{11}$ $0 \rightarrow PC_{11}$ $a_9 - 0 \rightarrow PC_9 - 0$	—	Переход к подпрограмме по адресу, определяемому десятью битами непосредственных данных; сохранение адреса возврата в верхнем регистре стека возврата; передача старого значения верхнего регистра стека в нижний	—	+

Мнемоника	Код операции (К7...К0)	Формат	Перемещение информации	Условие пропуска	Комментарий	КБ1013ВК1-2	КБ1013ВК4-2
CBR	$11W_5W_4W_3W_2W_1W_0$	I	PC_{10-8}, $PC_{5,4} \rightarrow PC_{10-8}$, $PC_{5,4}$, $W_{5,4} \rightarrow PC_{7,6}$; $W_{3-0} \rightarrow PC_{3-0}$	—	Переход к подпрограмме (в пределах 4 страниц, 16 слов), определяемый шестью битами непосредственных данных	+	—
CMS	01101101	IV	$\overline{MSB} \rightarrow MSB$	—	Формирование дополнения триггера выбора банка ПЗУ	+	—
JPA	00000011	IV	$Acc_{3-0} \rightarrow PC_{3-0}$	—	Передача содержимого аккумулятора в четыре младших разряда счетчика команд $PC_3 ... PC_0$	—	+
RT	01101110	IV	—	—	Возврат из подпрограммы	+	+
RTS	01101111	IV	—	—	Возврат из подпрограммы; безусловный пропуск следующей команды	+	+
SI1	01010000	IV	—	INT1=0	Пропуск следующей команды, если флаг внешнего прерывания INT1 не установлен	+	+
SI0	01010001 01011110	IV	—	INT0=0	Пропуск следующей команды, если флаг внешнего прерывания INT0 не установлен	+	—
SYN	01100101	IV	$0 \rightarrow T_{14-1}$	—	Сброс таймера; запуск таймера	+	+
TIM	01011000	IV	—	TIME=0	Пропуск следующей команды, если триггер-защелка старшего разряда таймера сброшен, сброс триггера-защелки старшего разряда таймера	+	+
SI4	01101000	IV	—	$T_{14}=1$	Пропуск следующей команды, если старший разряд T_{14} таймера установлен	—	+
SI1	01101001	IV	—	$T_{11}=1$	Пропуск следующей команды, если разряд T_{11} таймера установлен	—	+
HLT	01011101 00000000	IV	—	Ожидание прерывания	Установка режима резервирования; переход на адрес 0000Н ПЗУ из режима резервирования по нулевому значению триггера-защелки TIME старшего разряда таймера или по нулевому значению на одном из входов буфера D	+	—
	01011101	IV	—	Ожидание прерывания	Установка режима резервирования; переход на адрес 0040Н ПЗУ из режима резервирования по нулевому значению триггера-защелки TIME старшего разряда таймера или по нулевому значению на одном из входов буфера D	—	+

Таблица 6

Формат команд ОЭВМ серии КБ1013ВК

Формат команды	Разряды кода команды							
	К7	К6	К5	К4	К3	К2	К1	К0
I	КОП		W_5	W_4	W_3	W_2	W_1	W_0
II	КОП				C_3	C_2	C_0	C_1
III	КОП						B_1	B_0
IV	КОП							
V	КОП							
	0	$У_6$	$У_5$	$У_4$	$У_3$	$У_2$	$У_1$	$У_0$
VI	КОП							
	КОП							
VII	КОП				a_{11}	a_{10}	a_9	a_8
	a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0
VIII	КОП		a_{13}	a_{12}	a_{11}	a_{10}	a_9	a_8
	a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0

равлять выполнением прикладной программы и проверять содержимое любой переменной или элемента массива в процессе выполнения.

Программа автоматической прошивки ПЗУ.

Программы отладки работают под управлением операционных систем РАФОС на мини-ЭВМ «Электроника 100/25», «Электроника 79».

ОЭВМ серии КБ1013 предназначены для применения в калькуляторах, часах, таймерах, электронных играх, контроллерах радиоэлектронной аппаратуры, электроприборов и вычислительных устройств, автомобильных контроллерах и т. д.

Широкий спектр применений обеспечивается малым энергопотреблением (менее 200 мкВт) и наличием встроенного контроллера ЖКД.

В настоящее время на базе ОМЭВМ серии КБ 1013 выпускаются электронные игры «Ну, погоди!», «Электроника 24-01», «Веселый повар», «Тайны океана» и др.

По вопросам получения дополнительной информации обращаться в Головной консультативно-технический центр. Тел. 468-13-70, Москва.

Статья поступила 15 мая 1987 г.

ПИСЬМО ЧИТАТЕЛЯ

Уважаемый Главный редактор!

Пишу Вам во второй раз (если помните, «записки парасистемного программиста»). На этот раз — в связи с появлением в печати сообщений о ВНТК «СТАРТ» и ЭВМ пятого поколения в «нашем» варианте.

Кто я такой, Вы уже примерно представляете из «записок». Поэтому, а также, будучи полностью согласным с В. Е. Котовым (ЭКО, 1987, № 3), претендовать на роль разработчика ЭВМ пятого поколения никак не могу. Действительно, уже немолодой специалист (35 лет), за плечами тяжкий груз ЕС ЭВМ, практически никаких научных заделов (проще сказать — невежествен), так как все мое время ушло на подковывание блох. Нет, ЭВМ пятого поколения такие, как я, не сделают. Однако и мы тоже кое-чем можем в этом помочь.

Мы, эксплуатационники, лучше, чем кто-либо другой, знаем, как самые хорошие идеи могут в конце концов упереться в тысячу досадных мелочей вроде нестабильного питания, плохого заземления, неконтактов в разъемах, неудобных клавиатур, плохих тестов и т. п. Возможно, ЭВМ 5-го поколения не будут нуждаться в клавиатурах и будут питаться от батарейки. Все равно, найдется, если не тысяча, так пятьсот мелочей, которые можно увидеть лишь в процессе опытной эксплуатации. И тем раньше увидеть, чем раньше эта эксплуатация начнется, чем более квалифицированные эксплуатационники будут привлечены к этой эксплуатации.

Считаю полезным для будущих ЭВМ, если их создатели поищут по Союзу и сформируют несколько групп специалистов эксплуатационников (электронщиков и системных программистов) с целью использовать их примерно по следующей схеме.

Первый этап. Обучение с минимальным отрывом от производства. Этот этап продолжается до тех пор, пока не появятся опытные образцы устройств и модели. В этот период сформированные группы собираются периодически на краткосрочные курсы-семинары. В процессе работы авторы разработок в награду за свое усердие по ликвидации безграмотности смогут получить массу острых критических замечаний, причем, некоторая часть из них может оказаться полезной для улучшения эксплуатационных характеристик будущей техники.

Обязательными, на мой взгляд, должны быть следующие условия:
основное время обучаемые специалисты должны проводить на своей работе, в своих гудящих, холодных машинных залах, рядом со своими подопечными монстрами третьего поколения. Иначе они увязнут в будущем. Все плохое очень быстро забывается. Мощность отрицательной обратной связи резко упадет;
в процессе обучения специалистов не нужно разделять, обучая одних — только «железу», других — только программированию. Более того, ряд специалистов нужно в совершенстве обучить будущему «железу» и программированию. Эта группа должна ориентироваться на участие в разработке тестового ПО надежности.

Второй этап. Теперь уже имеются средства для опытной эксплуатации ПО, есть опытные образцы устройств. Имеется эксплуатационная документация. Пришла пора оторвать обученных специалистов от насиженных мест всерьез и надолго. Теперь они должны моделировать эксплуатацию. Сначала им должны быть доступны только документация пользователя и сами образцы. Вопросы разработчику задавать можно, но ответы на них — только с указанием на соответствующее место в документации. При необходимости документация дорабатывается.

Затем эксплуатационники должны получить доступ к документации разработчика. Это, а также самый тесный контакт с самим разработчиком могут оказаться полезными для выявления ряда скрытых дефектов, отражающихся на надежности. Это же необходимо и для разработки тестового ПО, разработки сервисной и контрольно-измерительной аппаратуры, для чего привлечение опытных эксплуатационников весьма полезно.

Что касается тестового ПО, то я убежден, что тесты должны составлять кто угодно, только не разработчики тестируемого устройства. Особенно это касается периферии. Никто кроме «парасистемных» программистов не сделает лучше тесты для периферийных устройств; тесты надежности, «гоняющие» технику в режимах, в которых ее использует программное обеспечение. Никто кроме них не поможет разработчику ПО найти точку в алгоритме, скажем, обмена с дисководом, где разработчик забыл, что его любимец может выдать сбой. Для меня лично нет большей радости, чем написать программу, которая сломает ЭВМ. За одну битую ЭВМ я десять небитых отдам наверняка.

Но это — радость в локальном смысле. В глобальном же, я успокоюсь только тогда, когда увижу ЭВМ, которые доставляют людям больше радости (хотя бы на полпорядка), чем огорчений.

Мой адрес: 197022, Ленинград, Кировский пр., д. 55, кв. 83, тел. 234-58-74.
Лишак Евгений Вениаминович

УДК 681.3.06

Ю. М. Погудин

МИАСС — СИСТЕМА МИКРОПРОГРАММИРОВАНИЯ НА ЯЗЫКЕ АМДАСМ

Кросс-система МИАСС, работающая на СМ ЭВМ под управлением ОС РВ, РАФОС или ИНМОС, дает возможность описывать язык уровня ассемблера, подходящий для спецвычислителя, и составлять микропрограммы на этом языке. Объединенный набор средств разработки микропрограмм ориентирован на модульную технологию. Название системы — сокращение слова «микроассемблер» — характеризует способ составления микропрограмм.

Назначение и условия применения

Система предназначена прежде всего для разработчиков спецпроцессоров, использующих микропроцессорные комплекты с распределенными вычислительными возможностями (типа К1804). Система облегчит проектирование любого цифрового устройства с микропрограммной реализацией функций. Она может оказаться полезной и как универсальное кросс-средство программирования на языке уровня ассемблера.

Система настраивается на произвольную архитектуру вычислителя и пригодна для роли кросс-системы микропрограммирования в любой САПР на базе ЭВМ с набором команд PDP-11 (СМ-4, «Электроника 60», ДВК). Текст системы кодирован на языке Си. Си-реализация совмещает ассемблерную эффективность с высокой мобильностью кода. Между ОС РВ и РАФОС перенос обеспечивается единой системой программирования C/DECUS.

В среде ОС РВ системе требуется не более 20К слов оперативной памяти инструментальной машины.

Возможности системы микропрограммирования

Система эксплуатируется в диалоговом режиме как задача базовой операционной системы. Собственный монитор системы интерпретирует командные строки, вводимые с клавиатуры или из косвенного командного файла.

Входом системы являются текстовые файлы на языках настройки и ассемблера АМДАСМ [1]. Базовые операционные системы предоставляют удобные средства для подготовки и ведения рабочего архива. Препроцессор системы обеспечивает сборку исходных текстов при трансляции. Применяются многострочные макроподстановки с параметрами из внешних файлов и механизм условной трансляции.

Система выполняет следующие основные функции:

определение языка символического кодирования средствами настройки языка АМДАСМ для микропрограммирования вычислителя конкретной архитектуры;

трансляцию микропрограмм, составленных из универсальных мета-ассемблерных компонент АМДАСМа и уникальных мнемоник, введенных на фазе настройки;

редактирование связей перемещаемых микропрограммных модулей, расположенных в общем адресном пространстве;

пост-обработку кода, т. е. преобразование результата трансляции в форму, учитывающую разбиение адресного пространства на реальные запоминающие устройства;

рабочее документирование программ, в том числе в виде карт прошивки ПЗУ.

В системе нет загрузчиков кода в микропрограммное ЗУ макета или в буфер программатора. Предполагается, что такие компоненты кросс-САПР зависят от условий эксплуатации и будут программироваться разработчиком САПР. Выходом системы МИАСС являются файлы фиксированного загрузочного формата. Загрузчик пользователя, легко встраиваемый в систему путем ее перегенерации, должен читать код из таких файлов и выводить его из кросс-ЭВМ подходящим способом.

АМДАСМ — универсальный микроассемблер

Настройка на архитектуру вычислителя и последующее программирование для него ведется средствами языка АМДАСМ (AMDASM), разработанного Дж. Миком и Дж. Бриком [1]. Есть основания говорить о формирующейся традиции использования именно этого языка для микропрограммирования спецпроцессоров. Доступность литературы о языке [2, 3], его реализации [2, 3] и его приложениях [4] — лишь внешнее проявление этого факта, но оно может оказаться решающим при сравнении АМДАСМа с его аналогами (см., например, [5]). АМДАСМ удобнее кросс-ассемблера, построенного из базового макроассемблера по известной схеме [6], так как он учитывает специфику микропрограммного уровня кодирования.

Реально АМДАСМ распадается на два языка: настройки и ассемблера. Язык настройки придает универсальную гибкость языку ассемблера, позволяя работать со словами произвольной длины и любым числом независимых микроконструкций. Пользователь задает идентификаторы управляющих битовых полей и характерных комбинаций таких полей, которые называются форматами микроинструкций. Форматы могут содержать переменные поля, на место которых возможна подстановка в процессе ассемблирования, поля с жестко заданными битовыми значениями и поля, значение которых остается неопределенным даже после трансляции. В целом совокупность мнемонических обозначений, введенных пользователем вместе с описанием форматов микроинструкций задает язык символического кодирования, составляющий специализированную часть языка ассемблера, на котором пользователь будет программировать для своей аппаратуры.

При составлении микропрограмм можно использовать такие традиционные средства языка ассемблера, как метки, прямое управление счетчиком адреса, подстановки параметров в инструкцию, указываемую по имени, арифметические выражения из констант, их имен, меток, текущего значения счетчика адреса. Они образуют универсальную часть языка ассемблера АМДАСМ, не зависимую от настройки на конкретную архитектуру.

Язык АМДАСМ учитывает специфику кодирования микропрограммного уровня в таких средствах, как свободное форматирование микрослов при трансляции, наложение микроинструкций по полям с неопределенным значением, модификация битовых полей при подстановках в качестве операндов микроинструкций, кон-

текстно-обусловленный выбор основания системы счисления для констант при подстановках.

Особенности реализации АМДАСМа в системе

Существующие реализации АМДАСМа [2, 3] не имеют средств автоматической компоновки независимо транслированных перемещаемых модулей. Размещение микропрограмм в объектном адресном пространстве выполняется вручную, а в исходном тексте используются абсолютные адресные выражения. Принцип модульности в традиционном понимании для разработки микропрограммного продукта здесь не применим. Предполагается, что крайне низкий объектный уровень программирования требует от разработчика отчетливого представления о размещении и взаимосвязях всех фрагментов микропрограмм в любой момент отладки, поэтому надежность ручной компоновки считается удовлетворительной. Выигрышем при абсолютном кодировании является простота локальных исправлений: любой фрагмент микропрограммы вплоть до отдельного микрослова может транслироваться и загружаться в память макета независимо от оставшейся части программы, что позволяет избегать ощутимых затрат на перекомпоновку и редактирование связей.

Проблема локальных исправлений существует в любой системе разработки программ, но в САПР программно-аппаратных комплексов она особенно остра. В полный отладочный цикл входят здесь такие громоздкие операции, как перегрузка кода из инструментальной машины в макет и аппаратное тестирование. Для локальных исправлений необходимы минимальные временные затраты. С другой стороны, быстрота локальных исправлений абсолютного кода противоречит надежности и простоте помодульной отладки с автокомпоновкой перемещаемых модулей.

Реализация АМДАСМа в системе основана на компромиссном решении, а именно: переместимость исходного текста модулей, раздельная трансляция модулей и автоматическое редактирование связей совмещены с генерацией неперемещаемого объектного кода и ручным заданием абсолютной точки размещения модуля. Такой подход поддерживается внешними средствами языка управления системой. Язык АМДАСМ в системе не имеет существенных отличий от варианта Дж. Мика и Дж. Брика. Чтобы пользоваться преимуществами модульной разработки, достаточно соблюдать некоторые необременительные ограничения. В частности, чтобы исходный модуль был переместимым, необходимо ограничиться ссылками на метки (или выражениями, содержащими метки) и не использовать абсолютные адреса.

В момент трансляции модуль размещается в объектном адресном пространстве и настраиваются все его внешние ссылки за счет импорта адресов внешних меток из рабочей области системы. В свою очередь в процессе трансляции модулей на языке настройки и языке ассемблера АМДАСМ в рабочую область экспортируются все глобальные объекты из этих модулей. Общий контекст трансляции модуля состоит из определений фазы настройки и адресных констант точек входа, содержащихся в рабочей области в момент трансляции.

Любой фрагмент микропрограммы может транслироваться в общем контексте, а затем загружаться в макет без изменений в основном массиве микрокода. При выполнении локального исправления программист должен быть уверен, что исправление действительно локально и не разрушает сложившуюся в текущий момент отладки глобальную структуру взаимосвязей фрагментов разрабатываемой микропрограммы. Гибкость данного подхода состоит в том, что в любой момент отладки существует выбор между быстрым локальным исправлением и надежной перекомпоновкой-перетрансляцией всей микропрограммы.

В системе нет ограничений на длину объектного микрослова или объем таблиц имен и форматов. Реализация обеспечивает динамическое определение этих размеров и выделение соответствующих объемов памяти инструментальной машины.

Язык управления системой

Язык управления системой — технологическая оболочка языка АМДАСМ. Принцип модульности поддерживается путем разделения операции настройки, трансляции и пост-обработки. Внутренняя операционная среда системы однозначно характеризуется наборами типов файлов и директив — операций над этими типами. Типы файлов системы МИАСС:

КОД	ВНУТРИСИСТЕМНАЯ ФУНКЦИЯ
FC	КОСВЕННЫЙ КОМАНДНЫЙ ФАЙЛ.
DS	ФАЙЛ ИСХОДНОГО ТЕКСТА МОДУЛЯ ОПРЕДЕЛЕНИЙ.
DL	ФАЙЛ ЛИСТИНГА МОДУЛЯ ОПРЕДЕЛЕНИЙ.
OW	ФАЙЛ ОБРАЗА РАБОЧЕЙ ОБЛАСТИ НА ДИСКЕ.
EL	ФАЙЛ ЛИСТИНГА ГЛОБАЛЬНЫХ ОБ'ЕКТОВ.
IL	ФАЙЛ ЛИСТИНГА ЛОКАЛЬНЫХ ОБ'ЕКТОВ.
AS	ФАЙЛ ИСХОДНОГО ТЕКСТА МИКРОПРОГРАММНОГО МОДУЛЯ.
AL	ФАЙЛ ЛИСТИНГА МИКРОПРОГРАММНОГО МОДУЛЯ.
AO	ФАЙЛ ОБ'ЕКТНОГО КОДА МИКРОПРОГРАММНОГО МОДУЛЯ.
ML	ФАЙЛ ЛИСТИНГА РАЗМЕЩЕНИЯ МИКРОПРОГРАММНОГО МОДУЛЯ.
CO	ФАЙЛ МИКРОПРОГРАММНОГО МОДУЛЯ В ЗАГРУЗОЧНОМ ФОРМАТЕ.
CL	ФАЙЛ ЛИСТИНГА ЗАГРУЗОЧНОГО КОДА.
PL	ФАЙЛ ЛИСТИНГА КООРДИНАТ ЗУ.

Директивы языка управления задаются в виде независимых командных строк. Командные строки имеют формат LF/CD LP, где LF — список файлов директивы, CD — код директивы, LP — список параметров, модифицирующих ее выполнение. Для краткого описания формата списка файлов и действия директивы используем следующую нотацию: роль файла (входной/выходной/модифицируемый) обозначаем символами I, O и M, форму (текст-код) — символами TX и CD в записи псевдонима файла, а тип — двухлитерным стандартным расширением имени файла. Псевдонимена DN, FN и SN означают имя файла по умолчанию, первое и второе установленные имена файлов соответственно. Директивы системы в такой нотации:

КОД	ДИРЕКТИВА	СПИСОК ФАЙЛОВ-АРГУМЕНТОВ
FC	ПЕРЕХОД К КОСВЕННОМУ УПРАВЛЕНИЮ	ITX.FC FN;SN
FN	ЗАДАНИЕ ИМЕНИ ФАЙЛА ПО УМОЛЧАНИЮ	DN;FN;SN
DS	ТРАНСЛЯЦИЯ МОДУЛЯ ОПРЕДЕЛЕНИЙ	ITX.DS
DL	ВЫВОД ЛИСТИНГА МОДУЛЯ ОПРЕДЕЛЕНИЙ	OTX.DL=ITX.DS
EL	ВЫВОД ЛИСТИНГА ГЛОБАЛЬНЫХ ОБ'ЕКТОВ	OTX.EL
IL	ВЫВОД ЛИСТИНГА ЛОКАЛЬНЫХ ОБ'ЕКТОВ	OTX.IL
OW	ВЫГРУЗКА РАБОЧЕЙ ОБЛАСТИ	OCD.OW
IW	ЗАГРУЗКА РАБОЧЕЙ ОБЛАСТИ	ICD.OW
SW	УСТАНОВКА СОСТОЯНИЯ РАБОЧЕЙ ОБЛАСТИ	
ES	РАЗМЕЩЕНИЕ ВЫПОЛНЯЕМОГО МОДУЛЯ	ITX.AS
AS	ТРАНСЛЯЦИЯ ВЫПОЛНЯЕМОГО МОДУЛЯ	OCD.AO=ITX.AS
AO	ЗАДАНИЕ АДРЕСА СВЯЗЫВАНИЯ	ICD.AO
AL	ВЫВОД ЛИСТИНГА ВЫПОЛНЯЕМОГО МОДУЛЯ	OTX.AL=ICD.AO;ITX.AS
ML	ВЫВОД ЛИСТИНГА РАЗМЕЩЕНИЯ МОДУЛЯ	OTX.ML=ICD.AO
RO	СОВМЕЩЕНИЕ ОБ'ЕКТНЫХ МОДУЛЕЙ	MCD.AO=ICD.AO
PX	ДООПРЕДЕЛЕНИЕ НЕОПРЕДЕЛЕННЫХ ПОЛЕЙ	
PH	УСТАНОВКА РАЗМЕРОВ ЗУ ПО ГОРИЗОНТАЛИ	
PV	УСТАНОВКА РАЗМЕРОВ ЗУ ПО ВЕРТИКАЛИ	
PL	ВЫВОД ЛИСТИНГА КОНФИГУРАЦИИ ЗУ	OTX.PL
CO	ГЕНЕРАЦИЯ ЗАГРУЗОЧНОГО МОДУЛЯ	OCD.CO=ICD.AO
CL	ВЫВОД ЛИСТИНГА ЗАГРУЗОЧНОГО МОДУЛЯ	OTX.CL=ICD.CO
LD	ВСТРОЕННАЯ ЗАГРУЗКА МОДУЛЯ	ICD.CO

Ниже будет описано действие каждой директивы. Совпадение кодов директив с кодами типов файлов подчеркивает их тесную функциональную связь.

При задании командных строк можно полностью опускать список файлов. В этом случае будут использоваться стандартные расширения и имя DN, установленное последней директивой FN. Файл с псевдонименем OTX — всегда экран терминала. Литеры ¤, % и & в

позициях имени файла указывают на имена DN, FN и SN, установленные в системе директивой FN или FC. Литера ☐ в позиции числового параметра позволяет ссылаться на значение, установленное директивой FN или FC.

Удобным приемом, облегчающим работу оператора, является косвенное управление системой из командных файлов, содержащих типичные цепочки директив. Выборка командных строк из косвенного файла инициируется директивой FC, допускающей сокращенную форму командной строки LF@LP. Имеется возможность параметризации при обращении к командному файлу путем установки имен и числового параметра при переходе к косвенному управлению.

В ответ на запрос '?' система выдает экспресс-справку о своих директивах на экран терминала. Командная строка, начинающаяся литерой #, передается монитору базовой операционной системы, что позволяет запускать утилиты, не выходя из системы и сохраняя текущее состояние рабочей области.

Разработка микропрограммы в системе

Настройка системы на работу с конкретной архитектурой вычислителя состоит в заполнении рабочей области системы определениями форматов и констант путем трансляции модулей настройки директивой DS. Для получения листинга модуля определений издается отдельная директива DL. Пример такого листинга приведен на рис. 1.

Под управлением параметра директивы DS рабочая область либо очищается перед трансляцией, либо пополняется новыми определениями. Во втором случае можно комбинировать определения из разных модулей. Все константы и форматы считаются глобальными объектами и их имена нельзя использовать дважды без предварительной очистки рабочей области. Директива EL позволяет просмотреть список всех глобальных имен, определенных в текущий момент работы с системой.

Содержимое рабочей области удобно выгружать на диск директивой OW, чтобы не транслировать модули настройки всякий раз при входе в систему. При запуске системы для восстановления настроечной информации достаточно издать директиву IW. С помощью директивы SW можно управлять размером области ОЗУ инструментальной машины, отводимой под рабочую область.

Размещение и трансляция. Трансляция выполняемых модулей на языке ассемблера АМДАСМ в системе МИАСС совмещена с выполнением компоновочных операций: микропрограммный модуль размещается в объектном адресном пространстве; внешние ссылки согласуются с использованием информации о точках входа, хранимой в рабочей области; объектный модуль неперемещаем, но не требует в дальнейшем редактирования связей.

```
EXAMPL.DL          11-OKT-86 CB 14:32      ВРИМЕР МОДУЛЯ ОПРЕДЕЛЕНИЯ

  1                TITLE ПРИМЕР МОДУЛЯ ОПРЕДЕЛЕНИЙ
  2     ;
  3     ; "        КОМЕНТАРИЙ В ЛЮБОЙ СТРОКЕ СЛЕДУЕТ ЗА СИМВОЛОМ ";"
  4                WORD 32                 ;ОПРЕДЕЛЯЕТСЯ ДЛИНА МИКРОСЛОВА.
  5     ,PC:       EQU B#1010              ;ОПРЕДЕЛЕНИЕ ИМЕНИ КОНСТАНТЫ.
                   10   ДЛИНА= 4 1010
  6                R1=1                    ;ЕЩЕ ОДНА КОНСТАНТА, ИСПОЛЬЗОВАНА
                   1    ДЛИНА= 1 1
  7                                        ;СОКРАЩЕННАЯ ФОРМА ОПЕРАТОРА EQU.
  8     ADR:       SUB 8XNH8               ;ПОДФОРМАТ ИЗ ПЕРЕМЕННОГО ПОЛЯ.
                   NNNNNNNN
  9     GOTO:      DEF 24X,ADR             ;ФОРМАТ ИСПОЛЬЗУЕТ ПОДФОРМАТ ADR.
                   XXXXXXXXXXXXXXXXXXXXXXNNNNNNNN
 10     CALL:      DEF H#5,PC,ADR,ADR,BX   ;ЕЩЕ ОДИН ФОРМАТ, ИСПОЛЬЗУЕТ ИМЯ
                   0101101ONNNNNNNNNNNNNNNXXXXXXX
 11                                        ;ПОДФОРМАТА И КОНСТАНТЫ
 12     TEST:      DEF H#3,12V,XN8ABC,     ;ПОЛЯ СО ЗНАЧЕНИЕМ ПО УМОЛЧАНИЮ.
                   0011NNNNNNNNNNNNNOOOOOOOOOOOXXXX
                         10101011110OXXXXXXXXXX
 13                /   129,X08X,4X         ;СТРОКА ПРОДОЛЖЕНИЯ ОПЕРАТОРА.
 14     MISDEF:    DEF 24X                 ;КОРОТКИЙ ФОРМАТ
                   XXXXXXXXXXXXXXXXXXXXXXXX

***>               ОШИБКА 14 НЕВЕРНАЯ ДЛИНА ФОРМАТА
 15                END
```

Рис. 1. Листинг модуля настройки

Директива ES размещает модули и заносит в рабочую область адреса всех глобальных меток и значения глобальных констант этого модуля. Директива ES является основным средством пополнения контекста трансляции на этапе ассемблирования. Выполнив директиву ES последовательно для всех модулей разрабатываемого набора микропрограмм, можно сформировать общий контекст трансляции всех модулей этого набора. Кроме настроечной информации, созданной директивой ES, контекст содержит адреса всех точек входа всех модулей набора и значения всех глобальных констант. Последующая трансляция любого модуля набора в общем контексте сопровождается одновременным связыванием всех его внешних обращений, поэтому не требуется никакого дополнительного редактирования связей. Характерно, что трансляция модуля в общем контексте будет трансляцией в собственном контексте, так как рабочая область уже содержит глобальные объекты данного модуля в момент трансляции. Для сохранения общего контекста можно использовать директивы OW и IW.

Директива AS транслирует исходный выполняемый модуль в объектный в контексте определений, существующих в рабочей области. Директива AL позволяет получить листинг выполняемого модуля (рис. 2), а директива ML — листинг карты размещения модуля в объектном адресном пространстве (рис. 3).

Характерным примером независимой трансляции модулей служит последовательность командных строк:

```
EXAMPL/FN;  ИМЯ МОДУЛЯ EXAMPL БЕРЕТСЯ ПО УМОЛЧАНИЮ.
/DS;        ТРАНСЛИРУЕТСЯ МОДУЛЬ НАСТРОЙКИ EXAMPL.DS.
/DL;        ФОРМИРУЕТСЯ ЛИСТИНГ НАСТРОЙКИ РИС. 1.
/EL;        НА ЭКРАН ВЫВОДИТСЯ СПИСОК ИМЕН РИС. 3.
/ES H#100;  НЕПЕРЕМЕЩАЕМЫЙ МОДУЛЬ EXAMPL.AS (СМ. РИС. 2)
            РАЗМЕЩАЕТСЯ С АДРЕСА 100 (ШЕСТНАДЦАТЕРИЧНОЕ).
OTHER/ES;   ПЕРЕМЕЩАЕМЫЙ МОДУЛЬ OTHER.AS (СМ. РИС. 2)
            РАЗМЕЩАЕТСЯ СРАЗУ ЖЕ ЗА ПРЕДЫДУЩИМ МОДУЛЕМ.
/EL 1;      НА ЭКРАН ВЫВОДИТСЯ ЛИСТИНГ ТОЧЕК ВХОДА (РИС 3).
/EL 2;      ВЫВОДИТСЯ ЛИСТИНГ ГЛОБАЛЬНЫХ КОНСТАНТ (РИС.3).
/AS H#100;  EXAMPL.AS ТРАНСЛИРУЕТСЯ С АДРЕСА H#100.
/AL;        ФОРМИРУЕТСЯ ЛИСТИНГ ЭТОГО МОДУЛЯ (СМ. РИС.2).
/IL;        ВЫВОДИТСЯ СПИСОК ЕГО ЛОКАЛЬНЫХ ИМЕН (РИС. 3).
OTHER/FN;   ИМЯ МОДУЛЯ OTHER БЕРЕТСЯ ПО УМОЛЧАНИЮ.
/AS;        OTHER.AS ТРАНСЛИРУЕТСЯ, РАЗМЕЩАЯСЬ ЗА EXAMPL.AS.
/AL;        ЛИСТИНГ OTHER.AS - НА ЭКРАН (РИС. 2).
/ML;        КАРТА РАЗМЕЩЕНИЯ OTHER.AS - НА ЭКРАН (РИС. 3).
```

Директива AO позволяет размещать объектный модуль вслед за другим, директива RO — объединять независимо транслированные объектные модули в одном файле и исправлять объектный модуль без полной перетрансляции.

Пост-обработка объектного микропрограммного модуля состоит в доопределении неопределенных полей оттранслированного объектного кода и преобразовании его в форму, учитывающую разбиение объектной микропамяти на реальные БИС ЗУ. Файл загрузочного формата после пост-обработки имеет структуру, максимально облегчающую перегрузку кода из инструментальной машины в память макета или буфер программатора.

Директива PX устанавливает значение бит полей, оставшихся неопределенными после трансляции, и может задавать инверсию каждого слова кода при загрузочном отображении. Директивы PH и PV указывают горизонтальные (вдоль микрослова) и вертикальные (вдоль оси адресов) размеры БИС ЗУ в макете или

```
EXAMPL.AL       11-ОКТ-86 СБ 14:32      ПРИМЕР ВЫПОЛНЯЕМОГО МОДУЛЯ

  1    0        TITLE ПРИМЕР ВЫПОЛНЯЕМОГО МОДУЛЯ
  2    0        ORG H#100               ;УСТАНОВКА СЧЕТЧИКА АДРЕСА
  3  100 A:     TEST PC+10,567          ;ССЫЛКА НА ФОРМАТ С ПОДСТАНОВКОЙ
                 00110000000101000001011101111XXXX
  4  100                                ;ФАКТИЧЕСКИХ ЗНАЧЕНИЙ АРГУМЕНТОВ
  5  101 B:;    FF 8(H8F)X,8X           ;СВОБОДНЫЙ ФОРМАТ
                 00001111XXXXXXXXXXXXXXXXXXXXXXXX
  6  102        TEST & GOTO A+PC        ;СОВМЕЩЕНИЕ ФОРМАТОВ
                 00111010101111100XXXXXXXX00001010
  7  102        RES 20                  ;РЕЗЕРВИРОВАНИЕ ПАМЯТИ
  8  117 C:;    CALL OTHER,X            ;ОБРАЩЕНИЕ К ВНЕШНЕЙ МЕТКЕ И
                 01011010100000010001011 1XXXXXXX
  9  117                                ;СЧЕТЧИК КОМАНД КАК АРГУМЕНТ
 10  117        ALIGN 64;               ;ВЫРАВНИВАНИЕ НА ГРАНИЦУ СТРАНИЦЫ
 11  140        GOTO B
                 XXXXXXXXXXXXXXXXXXXXXXX01000010
 12  140        END

OTHER.AL        11-ОКТ-86 СБ 14:32      ДРУГОЙ ВЫПОЛНЯЕМЫЙ МОДУЛЬ

  1    0                  TITLE ДРУГОЙ ВЫПОЛНЯЕМЫЙ МОДУЛЬ
  2  141 B08C0020 OTHER:: TEST B;NEW
  3  141                  NEW=.64
  4  141                  RES 20
  5  156 00680000 ENTRY:: FF 8(PC+B#2)X:,8X
  6  157 00000000         FF 16X
  7  158 0000E800         GOTO C
  8  158                  END
```

Рис. 2. Листинги выполняемых модулей (в разных форматах)

```
EXAMPL.EL       11-ОКТ-86 СБ 14:32      СПИСОК ИМЕН ГЛОБАЛЬНЫХ ОБ'ЕКТОВ

РЕЗЕР SUB       РЕЗЕР ALIGN     РЕЗЕР WORD      ФОРМ  GOTO
РЕЗЕР FF        КОНСТ PC        РЕЗЕР LIST      РЕЗЕР TITLE
РЕЗЕР ORG       ФОРМ  MISDEF    РЕЗЕР EJECT     ФОРМ  CALL
РЕЗЕР SPACE     КОНСТ R1        РЕЗЕР RES       РЕЗЕР END
РЕЗЕР DEF       ФОРМ  TEST      РЕЗЕР NOLIST    РЕЗЕР EQU

ВСЕГО ИМЕНОВАННЫХ ГЛОБАЛЬНЫХ ОБ'ЕКТОВ 20

ENTRIES.EL    11-ОКТ-86 СБ 14:32    СПИСОК ТОЧЕК ВХОДА

  101 B           117 C           141 OTHER       156 ENTRY

ВСЕГО ТОЧЕК ВХОДА 4

CONST.EL    11-ОКТ-86 СБ 14:32    СПИСОК ГЛОБАЛЬНЫХ КОНСТАНТ

   40 NEW           A PC            1 R1
ВСЕГО ГЛОБАЛЬНЫХ КОНСТАНТ 3

EXAMPL.IL    11-ОКТ-86 СБ 14:32    СПИСОК ИМЕН ЛОКАЛЬНЫХ ОБ'ЕКТОВ

МЕТКА A
ВСЕГО ИМЕНОВАННЫХ ЛОКАЛЬНЫХ ОБ'ЕКТОВ 1

OTHER.ML        11-ОКТ-86 СБ 14:32      КАРТА  РАЗМЕЩЕНИЯ  МОДУЛЯ

 141  141    156  158
ГРАНИЦЫ МОДУЛЯ  141 -  158
ДЛИНА МИКРОСЛОВА 32 БИТ

EXAMPL.PL     - 11-ОКТ-86 СБ 14:32      КАРТА РАСПРЕДЕЛЕНИЯ КОДА ВО ЗУ

          0-7   8-17 18-1F
  0-F      0     3     6
 10-2F     1     4     7
 30-AF     2     5     8
```

Рис. 3. Информационные листинги системы

изделии. Директива PL формирует листинг с изображением разбиения адресного пространства на сеть ЗУ. Директива CO отображает объектный модуль в загрузочный с использованием информации о структуре физического адресного пространства. Для вызова загрузчика пользователя без выхода из системы в языке управления зарезервирована директива LD.

Помодульная отладка микропрограмм. Разбиение микропрограмм на части и независимая трансляция заметно повышают эффективность отладки. При таком подходе к разработке системы микропрограмм можно выделить пять технологических операций: сборку системы из модулей, расширение системы новым модулем, локальное исправление в одном из модулей, локальное исправление фрагмента модуля, большую перестройку всей системы. Цепочки директив, реализующие эти операции:

Сборка микропрограммной системы из модулей

```
DEFROM/FC;          ОПИСАНИЕ КООРДИНАТ БИС ЗУ.
DEFINS/IW;          ЗАГРУЗКА НАСТРОЕЧНОГО КОНТЕКСТА.
FIRST/ES 0;         ПЕРВЫЙ МОДУЛЬ РАЗМЕЩАЕТСЯ С АДРЕСА 0.
SECOND/ES;          ВПЛОТНУЮ ЗА НИМ РАЗМЕЩАЕТСЯ ВТОРОЙ.
  ...
LAST/ES;            РАЗМЕЩАЕТСЯ ПОСЛЕДНИЙ МОДУЛЬ СИСТЕМЫ.
COMMON/OW;          СОХРАНЕНИЕ ОБЩЕГО КОНТЕКСТА НА ДИСКЕ.
COMMON=FIRST/AS 0;  ТРАНСЛЯЦИЯ ПЕРВОГО МОДУЛЯ С АДРЕСА 0
                    В "ОБЩИЙ" ОБ'ЕКТНЫЙ ФАЙЛ.
SECOND=SECOND/AS;   ТРАНСЛЯЦИЯ ВТОРОГО МОДУЛЯ.
COMMON=SECOND/RO;   ВТОРОЙ ОБ'ЕКТНЫЙ МОДУЛЬ ОБ'ЕДИНЯЕТСЯ
                    С ПЕРВЫМ В ОБЩЕМ ОБ'ЕКТНОМ ФАЙЛЕ.
  ...
LAST=LAST/AS;       ТРАНСЛИРУЕТСЯ ПОСЛЕДНИЙ МОДУЛЬ.
COMMON=LAST/RO;     ОН ВКЛЮЧАЕТСЯ В ОБЩИЙ ФАЙЛ.
COMMON/CO;          ОБЩИЙ ОБ'ЕКТНЫЙ ФАЙЛ ПРЕОБРАЗУЕТСЯ В
                    ФАЙЛ ЗАГРУЗОЧНОГО ФОРМАТА.
COMMON/LD;          ВЫГРУЗКА КОДА ИЗ КРОСС-МАШИНЫ.
```

Включение нового модуля в систему

```
COMMON/IW;          ЗАГРУЗКА ОБЩЕГО КОНТЕКСТА ТРАНСЛЯЦИИ
                    ОТЛАЖЕННОЙ ЧАСТИ СИСТЕМЫ.
LAST/AO;            АДРЕС РАЗМЕЩЕНИЯ НОВОГО МОДУЛЯ - СРАЗУ
                    ЗА ПОСЛЕДНИМ МОДУЛЕМ.
NEW/ES;             РАЗМЕЩЕНИЕ НОВОГО МОДУЛЯ.
COMMON/OW;          СОХРАНЕНИЕ РАСШИРЕННОГО КОНТЕКСТА.
LAST/AO;            ИЗВЛЕЧЕНИЕ АДРЕСА РАЗМЕЩЕНИЯ.
NEW=NEW/AS;         ТРАНСЛЯЦИЯ НОВОГО МОДУЛЯ.
COMMON=NEW/RO;      ВКЛЮЧЕНИЕ ЕГО В ОБЩИЙ ОБ'ЕКТНЫЙ ФАЙЛ.
```

Локальное исправление модуля

```
COMMON/IW;          ЗАГРУЗКА ОБЩЕГО КОНТЕКСТА.
BAD/FN;             ИМЯ МОДУЛЯ С ОШИБКОЙ - ПО УМОЛЧАНИЮ.
PREVIOUS/AO;        ИЗВЛЕЧЕНИЕ АДРЕСА РАЗМЕЩЕНИЯ.
/AS;                ПЕРЕТРАНСЛЯЦИЯ ОШИБОЧНОГО МОДУЛЯ.
/CO;                ПРЕОБРАЗОВАНИЕ ЕГО В ЗАГРУЗОЧНУЮ ФОРМУ.
/LD;                ВЫГРУЗКА МОДУЛЯ ИЗ КРОСС-СИСТЕМЫ.
```

Командный файл для исправления фрагмента модуля

```
H=TI:/AS X;         С КЛАВИАТУРЫ ТЕРМИНАЛА ВВОДИТСЯ (ВПЛОТЬ
                    ДО ОПЕРАТОРА END ИЛИ CNTRL/Z) ТЕКСТ
                    ИСПРАВЛЯЕМОГО ФРАГМЕНТА. АДРЕС РАЗМЕЩЕ-
                    НИЯ РАВЕН УСТАНОВЛЕННОМУ ПАРАМЕТРУ.
H/CO;               ПРЕОБРАЗОВАНИЕ ЕГО В ЗАГРУЗОЧНУЮ ФОРМУ.
H/LD;               ВЫГРУЗКА ФРАГМЕНТА ИЗ КРОСС-ЭВМ. ДАЛЕЕ
                    РАЗРАБОТЧИК МОЖЕТ ПРИСТУПАТЬ К ТЕСТИРО-
                    ВАНИЮ НА МАКЕТЕ, А СИСТЕМА ВЫПОЛНЯЕТ
                    ЕЩЕ ОДНУ ДИРЕКТИВУ:
X=H/RO;             РЕЗУЛЬТАТ ИСПРАВЛЕНИЯ СОХРАНЯЕТСЯ В
                    ОБ'ЕКТНОМ ФАЙЛЕ С УСТАНОВЛЕННЫМ ИМЕНЕМ.
```

Локальные исправления возможны, если они не смещают границ размещения модуля в адресном пространстве и не меняют адресов точек входа. В каждом модуле можно использовать единственную точку входа, совпадающую с нижней границей модуля. Верхнюю границу модуля, используя оператор ALIGN языка АМДАСМ, можно выравнивать на адрес, кратный указанному числу, резервируя пространство под возможное расширение модуля. Такая техника программирования позволит выполнять локальные исправления практически для каждого модуля набора без накладных расходов на перекомпоновку.

Система разработана в отделе информатики ВЦ СО АН СССР (Новосибирск). Телефон для справок:35-11-53.

ЛИТЕРАТУРА

1. Brick J., Mick J. Microprogramming Ups Your Option in MP-system Design.— EDN.— 1978.— Vol. 23, № 2.— P. 105—110.
2. Семейство АМ2900 со вспомогательным оборудованием / Перевод материала фирмы Advanced Micro Devices.— ВЦП: № Г-21484 (№ 82/13440 ГПНТБ).— 146 С.
3. Мозговой Г. П., Семенова С. С., Семин Е. И., Трещалин О. В. Мета-ассемблер МЕАСС для

микропроцессорных систем с наращиваемой разрядностью // Микропроцессорные средства и системы.— 1986.— № 3.— С. 20—22.
4. Мик Дж., Брик Дж. Проектирование микропроцессорных устройств с разрядномодульной организацией. В 2-х кн. — М.: Мир, 1984.—733 с.
5. Алексенко А. Г., Гапоненко А. В., Иванников А. Д., Курилов И. Д. Разработка и отладка микропрограммного обеспечения цифровых систем на основе секционированных микропроцессоров // Микропроцессорные средства и системы.— 1986.— № 1.— С. 37—43.
6. Лукьянов Д. А. Как написать кросстранслятор с языка ассемблера // Микропроцессорные средства и системы.— 1985.— № 4.— С. 35—41.

Статья поступила 11 декабря 1986 г.

УДК 681.3.06

А. К. Рауд, Р. Л. Смелянский

ОТЕЧЕСТВЕННЫЕ КРОСС-СИСТЕМЫ

Во всех направлениях применения микропроцессорных систем (МПС) неотъемлемую часть технологии их разработки и применения составляет программа. Тем самым, с точки зрения применения, микропроцессорная техника (МПТ) — это неразделимый комплекс программных и аппаратных средств. Причем аппаратная часть МПС в ходе эксплуатации изменяется редко, в то время как программная часть довольно часто (например, при переналадке технологической линии меняются программы встроенных микропроцессорных устройств (МПУ) управления).

Доля затрат на разработку ПО как по стоимости, так и по времени составляет бóльшую часть стоимости и общих временны́х затрат на разработку и на внедрение микропроцессорных систем. Поэтому «классический» путь разработки ПО — по окончании разработки технических средств — стал не пригодным. Теперь, чтобы сократить стоимостные и временны́е затраты, разработку ПО надо начинать вместе с разработкой технических средств или даже до начала их проектирования. Только в этом случае можно сократить общие сроки изготовления и внедрения готового продукта.

Одним из возможных инструментов для поддержки процесса программирования устройств микропроцессорной техники являются кросс-системы. Кросс-системы отличаются от так называемых резидентных систем тем, что архитектура инструментальной ЭВМ не совпадает с архитектурой разрабатываемого микропроцессорного (целевого) устройства. Различаются кросс-системы, работающие на больших универсальных ЭВМ или мини-ЭВМ под управлением мощных, хорошо развитых операционных систем, кросс-системы, работающие на мощных автономных микроЭВМ общего назначения и, наконец, кросс-системы, работающие на автономных мощных микроЭВМ, соединенных в локальную сеть с возможностью разделения общих ресурсов [1].

В течение последних лет в стране разработано и введено в эксплуатацию больше тридцати кросс-систем для программирования устройств микропроцессорной техники. С 1982 г.

Таблица 1

Технологические возможности кросс-систем

Тип	Введение библиотек и редактирование текстов	Трансляция программ	Редактирование связей модулей	Отладочное выполнение модулей	Отладочные средства	Документирование программы	Выпуск эксплуатационной документации по ГОСТ	Подготовка тестов	Тестирование функционирования программы	Тестирование временны́х характеристик программы	Документирование процесса реализации	Тестирование совместной работы аппаратуры и программы	Настройка на конфигурацию	Настройка на тип микропроцессора	Перевод программы в микроЭВМ
КАС580/МОСК580	+	а		м	+	+									л
СППМ-580	+	а		п	+	+									
САПО «Э-60»	+	ма	+	м	+	+			м						л
CSDS/580	+	я	+	м	+	+			м						
КС580/ЛГУ	+	я	+	м	+	+			м						
КОМПРМИС	+	я		м	+	+									л
ИСПО ПО АСНИ	+	я	+	м	+	+			м	м		м	+		
П 4.4	+	а		м	+	+									
КОСА	+	ма		м	+	+							+	+	
КОМ-2	+	я	+	м	+	+			м	м			+		
ТЕМП	+	ма	+	м	+	+	+		м		+		+	+	
СЕРП	+	ма	+	м	+	+		+	м	м	+	м	+	+	л
АРМ2-05	+	а, (я)	+	п/м	+	+	+		п/м	п		п	+	+	н
ПРА	+	ма	—	м	+	+	+	+	—	м	—	—	—	—	л
РУЗА	+	ма	+	м	+	+	+	+	м	м	+	—	—	+	л

Примечание:

\+ — работа поддерживается системой или операционным окружением (например, системы, как правило, не содержат специальных средств для ведения библиотек или редактирования текстов, а предлагают для этого средства, имеющиеся в операционных системах);
а — ассемблер;
ма — макроассемблер;
я — язык программирования среднего или высокого уровня;
м — программные модели аппаратуры;
п — непосредственно на аппаратуре;
л — перфолента;
н — непосредственное управление программатором ПЗУ.

целевой подгруппой технологии программирования микропроцессорной техники (ЦПГ МПТ) при рабочей группе по технологии программирования ГУВТ ГКНТ СССР составлен каталог средств автоматизации разработки микропроцессорных систем. В статье дан анализ свойств систем [2—24] с точки зрения поддержки ими разработки программ.

Цикл жизни программного обеспечения можно подразделить на следующие этапы [25]: анализ и спецификация проблемы, проектирование (декомпозиция на модули и алгоритмизация), реализация (кодирование, отладка, тестирование, документирование) и эксплуатация. Затраты на разработку составляют около 40 % [26], затраты на эксплуатацию (обучение пользователей, локализация, исправление ошибок и модификация программного обеспечения [27]) около 60 % от общих затрат.

Все рассматриваемые кросс-системы обеспечивают только этап реализации программ (табл. 1). Основное внимание во всех кросс-системах уделяется трансляции и автономной отладке модулей. Проверка временных характеристик, необходимая для тестирования программ реального времени, возможна не во всех системах. Средства создания тестов есть только в одной системе. Основной документацией, получение которой автоматизировано, являются распечатки программных модулей. Выпуск эксплуатационной документации и документирование процесса разработки поддерживается только в четырех системах. Однако следует учитывать, что существует множество самостоятельных систем подготовки документаций, работающих в ОС ЕС, которыми можно пополнять функции кросс-систем.

Из табл. 1 видно, что пользователь вынужден комплектовать инструментальное обеспечение из нескольких систем, так как ни одна из рассматриваемых систем не поддерживает всех этапов жизненного цикла программ. Поэтому определенное преимущество имеют системы, работающие в ОС ЕС. Серьезными трудностями использования ЕС ЭВМ являются малая надежность этих машин и сложность перевода программ в целевую ЭВМ. Традиционно для этого используется перфолента, однако на ЕС ЭВМ для других целей она применяется мало, а из конфигураций современных микроЭВМ постепенно исчезает. Определенные возможности для перевода программ предоставляет терминальная станция ЕС 7970 [28].

Большинство из рассматриваемых кросс-систем работают на ЕС ЭВМ в ОС ЕС в пакетном режиме (табл. 2). Они слабо документированы и распространяются в основном только разработчиками. В ФАП переданы только П 4.4 (в НПО «Алгоритм») и СЕРП (в НПО «Центрпрограммсистем»), а АРМ2-05 производится серийно заводами страны.

Рассматриваемые системы написаны на ассемблере или на ПЛ/1, что делает невозможным их перенос на другую ЭВМ или операционную систему.

Сводные данные о характеристиках устройств на МПТ и классов ПО для них, поддерживаемых рассматриваемыми кросс-системами, приведены в табл. 3. По видам применения микропроцессорные системы разделены на:

автономные микроЭВМ: конструктивно законченные вычислительные устройства с собственной традиционной периферией;

встроенные МПУ — вычислительные устройства, являющиеся неотъемлемой частью технологического оборудования, куда они встроены, выполняющие сложные функции управления и, как правило, не имеющие стандартных внешних устройств;

Таблица 2

Технические характеристики кросс-систем

Тип	Операционная система	Инструментальная ЭВМ	Минимальный объем основной памяти, К байт	Режим работы	Необходимые дополнительные устройства	Объем программ кросс-системы, К байт	Язык программирования	Наличие документации	Распространение
КАС580/МОСК580	ОС ЕС	ЕС ЭВМ		пакетный	—				
СППМ-580	ОС ЕС	ЕС ЭВМ	128	пакетный	—	130	ПЛ/1	+	—
САПО-«Э60»	ОС ЕС	ЕС ЭВМ	120	пакетный	—		ассемблер, ПЛ/1	+	—
CSDS/580	ОС ЕС	ЕС ЭВМ	170	пакетный	ЕС7066	170	ассемблер, Фортран	+	—
КС К580/ЛГУ	ОС ЕС	ЕС ЭВМ	256	пакетный	—	400	ассемблер, ПЛ/1	ЕСПД	—
КОМПРМИС	ОС ЕС	ЕС ЭВМ	120	пакетный	—	200	ассемблер, ПЛ/1	—	—
ИСПО ПО АСНИ	ДОС ЕС ОС ЕС	ЕС ЭВМ	256	пакетный диалоговый	ЕС7066	1000	ассемблер, ПЛ/1	—	—
П 4.4	ДОС СМ	СМ-3 СМ-4	48	диалоговый	—	94	ассемблер	ЕСПД	А
КОСА	ОС ЕС	ЕС ЭВМ	256	пакетный	—	200	ассемблер	+	—
КОМ-2	ДОС АСВТ	М4030.1		пакетный			ассемблер	+	—
ТЕМП	ДИСПАК	БЭСМ-6	192	пакетный	—	270	автокод	ЕСКД	—
СЕРП	ОС ЕС ОС РВ	ЕС ЭВМ	макс. 300	диалоговый	ЕС7920	500	ассемблер	ЕСПД	Ц
АРМ2-05	СПО СМ1800	СМ-4 СМ1800		диалоговый пакетный	—			ЕСПД ЕСКД	В Ц
ПРА	РАФОС	СМ1800	64	диалоговый	—	250	ассемблер	ЕСКД	Ц
РУЗА	ОС 6.1	ВУМС 025 ЕС ЭВМ	300	пакетный диалоговый пакетный	—	500	ассемблер	ЕСПД	А

Примечание. А — централизованные фонды алгоритмов и программ НПО «Алгоритм», Ц — НПО «Центрпрограммсистем», В — завод, выпускающий комплекс.

Таблица 3

Характеристики разрабатываемых программ

Тип	Микропроцессорная серия, микроЭВМ	Количество процессоров	Вид применения	Характеристика ПО	Ограничения на объем ПО
КАС580/МОСК580	К580	1	Контроллер	Общее назначение	200 оп.
СППМ-580	К580	1	То же	То же	
САПО «Э-60»	«Электроника 60»	1	Автономная микроЭВМ	»	
CSDS/580	К580	1	Контроллер Автономная микроЭВМ	»	
КС580/ЛГУ	К580	1	Автономная микроЭВМ	»	
КОМПРМИС	«Электроника 60»	1	То же	»	
ИСПО ПО АСНИ	«Электроника 60»	1	»	Реального времени	32К слов
П 4.4	К580	1	Встроенная микроЭВМ	То же	64К байт
КОСА	К580 К584 «Электроника С5/02, 12, 21, 21М спец.»	1	То же	Общее назначение	
КОМ-2	СМ 1800	1	Встроенная микроЭВМ	То же Реального времени	64К байт
ТЕМП	К580 спец.	1	То же	Общего назначения	
СЕРП	К580, К581, К1801, К1810, спец.	Несколько	»	То же	Объем основной памяти объектного устройства
АРМ2-05	К580 СМ 1800, спец.	1	Встроенная микроЭВМ	То же Реального времени	
ПРА	К580 К1810 К1801	1	То же	То же	20К байт
РУЗА	К580 К1810 К1801	1	»	»	Объем основной памяти объектного устройства

контроллеры — функционально самостоятельные узлы вычислительной или технологической аппаратуры с узкоспециализированными функциями управления.

Программное обеспечение (ПО) разделено на ПО общего назначения — программы, к которым не определено жестких требований по временным характеристикам, и ПО реального времени — это ПО, для которого такие требования предъявляются.

Большинство рассматриваемых кросс-систем (табл. 4) поддерживаются только языками программирования уровня ассемблера. Одни из них совпадают по синтаксису с резидентными ассемблерами (например, КС К580/ЛГУ, CSDS/580, СЕРП и др.), другие отличаются синтаксисом и мнемоникой команд (например, КОСА, СППМ-580, ТЕМП и АРМ2-05). Единственным языком высокого уровня в рассматриваемых кросс-системах является ПАСКАЛЬ— в системе КОМПРМИС реализован кросс-транслятором. С помощью кросс-трансляторов реализованы языки среднего уровня PL/M, АССОЛ/М и ПЛ-11. Остальные трансляторы — резидентные: в КОМ-2 они сделаны на базе модели аппаратуры, а в АРМ2-05 — на терминальной микроЭВМ СМ-1800. Оптимизирующим является только транслятор с языка ассемблера в системе КОСА.

Для работы с кросс-системой программисту необходим набор технологических языков. В рассматриваемых системах в лучшем случае унифицирован синтаксис либо существует стилевое единство. Предлагаемые в кросс-системах средства отладки (трассировка, диагностическая печать и др.) являются традиционными, но их уровень требует от пользователей достаточно высокой квалификации — им должен быть программист-профессионал.

Ключевая проблема по организации отладки и тестирования в кросс-системах — выполнение создаваемой программы. На универсальной ЭВМ она решается с помощью программной модели целевой аппаратуры (имитатора), а в гибридных системах непосредственно на целевой аппаратуре.

Создание имитатора существенно дешевле, чем создание специальных аппаратных средств отладки. Программные модели строятся быстрее, чем аппаратные средства, и легче поддаются изменениям, поэтому кросс-системы предоставляют возможность отладки программ независимо от степени готовности целевой аппаратуры.

Возможность отладки программ независимо от готовности целевой аппаратуры оказывает непосредственное

влияние на технологию разработки самой аппаратуры. Эта возможность достигается с помощью имитаторов. Объем основной памяти, необходимое быстродействие памяти и процессоров, внешних устройств и т. д. можно оценить непосредственно на реальных программах [29, 30].

При использовании имитаторов (табл. 5) возникает сложная проблема определения соответствия модели и моделируемой аппаратуры (либо ее проекта). С помощью специальных языков моделирования и генераторов [10, 31] достигается сокращение трудозатрат построения имитаторов [32]. С другой стороны, применяемые в генераторах универсальные механизмы моделирования не порождают моделей такого быстродействия, как при программировании вручную. Быстродействие модели является существенной характеристикой системы, так как определяет реактивность системы в целом. Это важно, например, при организации диалоговой отладки. Однако, как было отмечено, чем выше быстродействие модели, тем больше расходы на ее создание. Разумная грань определяется возможностями инструментальной машины, объемом моделируемой аппаратуры и необходимостью моделировать операционное окружение [33, 34, 35].

Выводы. Рассмотренные в данной статье кросс-системы являются инструментальными средствами профессионального программиста.

Основные области применения кросс-систем характеризуются:

промышленным характером проекта (коллектив программистов работает с жестким временным графиком);

средними и большими объемами программ проекта;

разработкой программ специализированных процессоров (с микропрограммным управлением);

совместной разработкой программ и аппаратуры устройства;

разработкой системного программного обеспечения многопроцессорных устройств и сетей;

специализированными инструментальными средствами для программистов непрофессионалов;

инструментальным средством для обучения использования МПТ и изучения возможностей МПТ в вузах.

В настоящее время основной инструментальной ЭВМ является ЕС ЭВМ с ОС ЕС, хотя для поддержки средств программирования микропроцессорных устройств у нее есть недостатки: сложность непосредственно (on-line) подключения целевой аппаратуры и программаторов постоянной памяти при переводе программ в целевую ЭВМ; малая надежность; сложность ОС ЕС (из широкого спектра возможностей для инструментальной поддержки используется незначительная доля).

Принципиальными недостатками кросс-систем на больших инст-

Таблица 4

Языковые средства кросс-систем

Тип	Язык программирования	Используемые языки	Возможности отладки			
			трассировка	диагностическая печать	останов по контрольным точкам	отладка в понятиях исходного языка
КАС580/МОСК580	Абсолютный ассемблер	ЯУЗ ОС ЕС				
СППМ-580	То же	ЯУЗ ОС ЕС, управление отладчиком	+	+		
САПО «Э60»	Ассемблер «Электроника 60», макрогенератор	ЯУЗ ОС ЕС, управление отладчиком, управление редактором связей	+	+		
CSDS/580	Перемещаемый макроассемблер для 8080	ЯУЗ ОС ЕС, управление отладчиком, управление редактором связей	+	+		
КС К580/ЛГУ	Макроассемблер К580/ЛГУ; ассемблер для 8080 АССОЛ/М	ЯУЗ ОС ЕС, ассемблер ЕС ЭВМ, управление редактором связей	+			
КОМПРМИС	ПАСКАЛЬ	ЯУЗ ОС ЕС, управление отладчиком		+		
ИСПО ПО АСНИ	Макроассемблер «Электроника 60»; ПЛ/11	ЯУЗ ОС ЕС, управление редактором связей, управление отладчиком	+	+		
П 4.4	Ассемблер	ЯУЗ ДОС СМ; управление редактором связей, управление отладчиком		+		
КОСА	Настраиваемый макроассемблер (оптимизирующий)	ЯУЗ ОС ЕС, управление редактором связей, управление отладчиком, моделирование аппаратуры	+	+	+	
КОМ-2	Ассемблер, ПЛ/М, БЕЙСИК, ПАСКАЛЬ	ЯУЗ ДОС АСВТ, управление редактором связей, управление отладчиком	+	+		
ТЕМП	Макроавтокод, параметрический, настраиваемый на систему команд	ЯУЗ ДИСПАК, язык отладки, язык описания глобальных переменных, язык описания условий применения системы, управление с информационной базой	+	+		
СЕРП	Макроассемблер, интел 8080, макроассемблер МАС-11 для К1801АМ1 и макроассемблер АСМ86 для К181РВМ86	Язык технологии операции, язык отладки, язык редакторских связей, язык моделирования окружения (тестирования), язык моделирования аппаратуры	+	+	+	+

Окончание табл. 4

Тип	Язык программирования	Используемые языки	Возможности отладки: трассировка	диагностическая печать	останов по контрольным точкам	отладка в понятиях исходного языка
АРМ2-05	Универсальный ассемблер, ПАСКАЛЬ, БЕЙСИК, PL/M	Диалоговый язык управления, язык документирования, метаязыки	+	+	+	+
ПРА	Ассемблер	23 отладки, язык СУБД	+	+	+	+
РУЗА	Ассемблер, PL/1, L	Язык спецификаций, язык отладки, язык диалоговый, язык СУБД	+	+	+	+

Таблица 5

Характеристики моделируемой аппаратуры

Тип	Способ адаптации	Объект моделирования: процессор-память	устройство ввода-вывода	время	Быстродействие
КАС580/МОСК-580	—	+	—	—	
САПО «Э60»	—	+	—	—	
CSDS/580	—	+	—	—	
КС580/ЛГУ	Макросредства ассемблера ЕС ЭВМ	+	—	—	
КОМПРМИС	—	+	—	—	5
ИСПО ПО АСНИ	Специальный язык	+	+	+	
П 4.4	—	+	—	—	
КОСА	Специальный язык	+	+	—	280
КОМ-2	Макросредства ассемблера М4030.1	+	+	+	7
ТЕМП	Параметрическая	+	—	—	1000
СЕРП	Специальный язык	+	+	+	100 К580 280 К581 250 К1810
ПРА	Специальный язык	+	+	+	
РУЗА	Специальный язык	+	+	+	

Примечание. Быстродействие имитатора выражено средним числом команд инструментальной ЭВМ, необходимых для выполнения одной команды микропроцессора.

рументальных ЭВМ являются изолированность процессов разработки основных компонентов микропроцессорной системы (программного обеспечения и аппаратуры) и отсутствие средств поддержки этапа интеграции программ с аппаратурой, а затем отсутствие у программиста ощущения автономности (исход выполнения его работы на ЭВМ зависит от множества неконтролируемых им событий).

Трудности внедрения кросс-систем объясняются бедными возможностями моделирования аппаратуры, сложностью построения моделей и необходимостью знания технологических языков и операционной системы инструментальной ЭВМ; недостатками компонентов инструментальной системы (например, малая надежность инструментальной ЭВМ, чрезмерная сложность операционной системы и т. д.); бедным набором языков программирования, отсутствием обеспечения всех этапов жизненного цикла программного обеспечения (кроме реализации программы).

Кросс-системы еще долго будут использоваться для поддержки проектов, в которых целесообразнее применять резидентные инструментальные средства, так как:

значительное количество программистов микропроцессорных устройств привыкли к хорошему технологическому сервису больших ЭВМ общего назначения;

объем производства автономных микроЭВМ с подходящей для программирования конфигурацией явно недостаточен;

специализированные аппаратные средства отладки (внутрисхемные эмуляторы) разрабатываются медленно и объем их выпуска недостаточен.

Реальной альтернативой кросс-системы на больших инструментальных ЭВМ общего назначения являются инструментальные системы на мощных микроЭВМ (отладочные системы), соединенные в локальную сеть с возможностью разделения дорогих периферийных устройств.

Адрес для справок: 117234, Москва, Ленинские горы, МГУ, факультет ВМиК. Лаборатория вычислительных комплексов. Телефон: 939-46-71.

ЛИТЕРАТУРА

1. **Warren F. E.** Understand the tradeoffs in development system selection // EDN.— 1981.— V. 26.— N 12.— P. 103—109.
2. **Герасимов И. В., Арбузов Т. А., Костичев С. В.** Методика разработки и отладки программ целевого назначения для МП К580 с использованием комплекса КАС-580/МОСК-580.: Сб. статей.— М.: Знание, 1982.— С. 125—129.
3. **Гриншпан Л. А., Малюш Я. Т., Шерлинг Д. Р.** СППМ-580 — система подготовки программ для микропроцессорной серии К580.— Инф. листок БелНИИТИ № 326, 1981, 3 с.
4. **Система** автоматизированной подготовки и отладки программ микроЭВМ «Электроника 60». Инструкция для пользователя.— ЭНИМС, 1981.

5. Тисс П. А. Кросс-система проектирования программного обеспечения для микропроцессорной серии К580 // Микропроцессорные системы. / Л.: ЛДНТИ, 1982.
6. Гнедин В. А. Инструментальная система КОМПРМИС для программирования микропроцессоров и микроЭВМ с использованием языка ПАСКАЛЬ // Автоматизация программирования аналогово-цифровых и микропроцессорных систем / М.: Знание, 1982, С. 115—120.
7. Гнедин В. А. Промежуточные языки как средство обеспечения программной совместимости микропроцессорной техники // Автоматизация разработки и моделирования вычислительных и микропроцессорных систем / М.: Знание, 1983. С. 109—113.
8. Цифровая имитация автоматизированных систем / А. А. Болтянский, В. А. Виттих, М. А. Ко раблин, Г. Н. Куклин. А. А. Сидоров и др.— М.: Наука, 1983,— С. 264.
9. Шамашов М. А. Кросс-система программирования для микроИВК // Р-технология программирования. Тез. докл. 1-й Всес. конф.— Ч. 3.— Киев: ИК АН УССР, 1983, С. 27—29.
10. Шамашов М. А. Функциональное описание и имитационное моделирование КАМАК-систем // УСиМ.— 1984.— № 2.— С. 106—111.
11. САПР П 4.4 (Алгоритмы и программы). Ереван, 1980. Деп. СФАП СНПО «Алгоритм», № И 20.005.016.
12. Котляров В. П, Самочадин А. В. Универсальная система автоматизации программирования для микроЭВМ с оптимизацией проблемных программ // Электронная промышленность.— 1981. Вып. 7.
13. Самочадин А. В. Оптимизатор структурированных ассемблерных программ для микроЭВМ. // Технология программирования микропроцессорной техники / Таллин: Валгус, 1982. — С. 89—99.
14. Гоппа Т. А., Иткин Л. К., Кузнецова Л. А. Состав и функции системы КОМ-2 // Математическое обеспечение и автоматизация проектирования ЭВМ / М.: Тр. ИНЭУМ.—1981.—Вып. 87.
15. Иткин Л. К., Гоппа Т. А., Кузнецова Л. А. Мобильная система моделирования и кросспрограммного обеспечения СМ-1800 // Автоматизация разработки и моделирования вычислительных и многопроцессорных систем / М.: Знание, 1983.— С. 123—126.
16. Липаев В. В., Каганов Ф. А. Система автоматизации технологии разработки комплексов управляющих программ для микропроцессора и микроЭВМ (ТЕМП.) // УСиМ.— 1980.— № 1.— С. 32—
17. Каганов Ф. А. Автоматизация проектирования комплексов управляющих программ и микропрограмм для микроЭВМ.// VIII Всесоюзное совещание по проблемам управления. Тез. докл.— Кн. 3 / Таллин: Валгус, 1980.—
18. Рауд Р. СЕРП-система разработки программ, управляющих микроЭВМ класса СМ1800 // Программное обеспечение АСУ / Калинин.—1980.— С. 130—133.
19. Рауд Р., Бергсон А. Использование функционально-эквивалентных модулей при разработке программ УЦВМ // Программирование.— 1981.— № 4.— С. 50—56.
20. Ольман В., Рауд Р. Моделирование аппаратуры мультипроцессорных устройств в СЕРП // Автоматика и вычислительная техника.—1983.— № 4.— С. 25—29.
21. Забара С. С., Мильнер А. Д. Проблемно - ориентированный комплекс автоматизированных рабочих мест. — Киев: Знание, 1981.—27 с.
22. Мильнер А. Д. Лингвистическое обеспечение инструментального комплекса АРМ2-05: Предпосылки, концепции, обзор // Программирование микропроцессорной техники / Изд. АН ЭССР.— Таллин.— 1984.— С. 30—41.
23. Шутрик А. А. РУЗА — система автоматизации разработки программ для управляющих и микроЭВМ // Микропроцессорные средства и системы. — 1984. — № 2.— С. 46—49.
24. Липаев В. В., Каганов Ф. А., Керданов А. В. и др. Система автоматизации проектирования программ на базе персональных ЭВМ (система ПРА)// Микропроцессорные средства и системы.—1985.— № 4.— С. 42—
25. Lehman M. Programms, Life Cycles and Lows of Soffware Evolution.— Proc. IEEE.— V. 68.— № 9.— P. 1060—1076.
26. Merwin R. E. Software management through product control.— Electronic Design.—1981.—V. 29.— № 1.— P. 190—199.
27. Schindler M. 1981 Technology Forecast: Software.— Proc. of Europcomp.—1974.— P. 59—73.
28. Липаев В. Б., Насекина Т. В., Виноградова Е. М. Система автоматизации программирования терминальной станции ТС 7906 // Автоматизация разработки и моделирования вычислительных и микропроцессорных систем / М.: Знание, 1983.— С. 100—103.
29. Рауд Р. Технология реализации микропроцессорных устройств в СЕРП // Семинар по технологии программирования МПК по ВТ / София, 1982.— С. 61—65.
30. Bruggere T. H. On schedule reliable software depends on sound methodology. — END. — 1981.— V. 26.— № 1.— P. 152—155.
31. Курьяма А., Рауд Р. Технология проектирования микроЭВМ в СЕРП. // Технология программирования микропроцессорной техники / Таллин: Валгус, 1982.— С. 23—33.
32. Piloty R., Barbacci M., Borrione D., Dietmeyer D., Hill F., Scelly P. Conlan Report.— Lecture Notes in Computer Science.— V. 151.— Springer Verlag.—1983.—174 p.
33. Витенберг И. М., Герханова Т. И., Либеров А. Б. Обработка системных вызовов при моделировании программ // Автоматизация программирования аналого-цифровых и микропроцессорных систем / М.: Знание, 1982.— С. 90—95.
34. Рауд Р. Технология тестирования программ микроЭВМ в СЕРП // Синтез, тестирование, верификация и отладка программ / Рига, 1981.—С. 186—187.
35. Cavouras J. C., Davis R. H. Simulation Tools in Computer System Design Methodologies.— Computer Journal. — 1981. — V. 24.— № 1.— P. 25—28.

Статья поступила 6 ноября 1985 г.

РЖ ВИНИТИ

4Б653. Графический сопроцессор на одном кристалле Intel 82786. A graphics coprocessor chip — the 82786. *Randall Martin, Johary Arun.* «Electron. Eng.», 1986, 58, № 717, 55—56, 58, 61, 63, (англ.).

Фирма Intel (США) разработала специализированную графическую БИС типа Intel 82786, ориентированную на совместную работу с 32-разрядным микропроцессором (МП) типа Intel 80386. На кристалле БИС размещены графический процессор (ГП), дисплейный процессор (ДП), блок сопряжения с МП-шиной и контроллер динамического ОЗУ. ГП обрабатывает списки графических команд высокого уровня и создает в видеопамяти растровое представление графического изображения и текстов. Контроллер динамического ЗУ обеспечивает ГП и ДП доступ к видеопамяти со скоростью обмена до 40 Мбайт/с. ДП управляет образованием окон на экране ЭЛТ. Высота окна может быть от одной строки растра до высоты экрана, ширина окна от 1/16 до полной ширины экрана. Для кодирования каждого элемента изображения используется от 1 до 8 разрядов; допускается наложение изображений одно на другое с различной разрядностью элементов. При использовании одной БИС Intel 82786 разрешающая способность генерируемого изображения до 640×840 элементов.

УДК 681.3

В. П. Котляров

ТЕХНОЛОГИЯ РАЗРАБОТКИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ВСТРОЕННЫХ МИКРОЭВМ И ПОДДЕРЖИВАЮЩИЕ ЕЕ ИНСТРУМЕНТАЛЬНЫЕ КОМПЛЕКСЫ

Интенсивное внедрение средств микропроцессорной техники в современное автоматизированное производство, несмотря на наличие достаточной номенклатуры встроенных микроЭВМ и микропроцессорных систем для широкого спектра промышленных применений, сдерживается сложностью разработки программного обеспечения (ПО). Средства автоматизированной разработки ПО, созданные для основных семейств микроЭВМ, требуют от пользователя специальной программистской подготовки [1], что трудно осуществить в широких масштабах. Потребности в специалистах по программированию для микроЭВМ и микропроцессорных систем растут существенно скорее, чем соответствующие возможности их подготовки. Поэтому перспективно привлекать к программированию систем непрофессионалов в программировании — инженеров из промышленности, хорошо знакомых с «физикой» автоматизируемых процессов, путем создания специализированных инструментальных средств и технологий разработки ПО, ориентированных на особенности применения и эксплуатации технических средств [2, 3], а также на квалификацию персонала.

Системные программисты создают для каждого семейства микроЭВМ инструментальные системы (технологические комплексы (ТК) автоматизации разработки ПО на базе универсальных языков программирования, проблемные программисты — программное обеспечение проблемных систем для встроенных микроЭВМ или микроконтроллеров с помощью инструментария, построенного на первом этапе. Недостаток подхода — потребность в огромном числе инженеров с программистским образованием.

Предложено расширить традиционную схему, введя этап создания проблемно-ориентированных ТК, в которых за счет настройки языковых средств программирования и управления вычислительным процессом на проблемную область существенно понижен уровень программистской квалификации пользователей — программистами проблемных систем становятся инженеры (программисты-непрофессионалы). Системные программисты создают универсальные комплексы для каждого из микропроцессорных семейств, проблемные программисты используют универсальные комплексы для разработки проблемных ТК, а непрофессиональные или проблемные программисты с помощью проблемных ТК создают системы автоматизации производственных процессов.

Новая технологическая схема учитывает ведущуюся повсеместно работу по переквалификации инженеров из промышленности в программистов проблемных систем и повышает их роль. Необходимо автоматизировать этап разработки проблемных ТК, так как надо обеспечить создание уже не нескольких ТК по числу выпускаемых комплексов, а сотен — по числу областей применения. Эта задача возложена на системных программистов, которые для каждого микропроцессорного комплекта теперь создают метакомплексы или универсальные ТК, предназначенные для быстрой разработки проблемных ТК по техническому заданию заказчика. В результате достигается эффективное распределение усилий высококвалифицированных специалистов в решении важной народнохозяйственной задачи.

Аппаратные средства профессиональных технологических комплексов

В настоящее время для реализации как универсальных так и проблемных ТК используют большие инструментальные ЭВМ (ЕС ЭВМ) с терминальными станциями (ЕС 7970, 7990) или резидентные АРМ (АРМ2-05.11). Это стационарное оборудование, установка и эксплуатация которого на производственном участке не всегда возможна, а требуемая для его обслуживания квалификация пользователей-программистов достаточно высока, поскольку наряду со знанием физически наглядного проблемно-ориентированного языка программирования здесь требуется умение работать с довольно сложным языком управления.

В реальных условиях современного производства в дополнение к стационарным необходимы переносные варианты ТК, которые непосредственно на объекте обеспечивают средствами автоматизации различные этапы разработки ПО. Для этих целей выгодно использовать персональные микроЭВМ, в которых для каждой проблемной области создаются проблемно-ориентированные системы программирования [4], фиксируемые в постоянной или перепрограммируемой памяти. В этом случае ТК реализуется как встроенное устройство, постоянно включенное в состав контроллера. В ряде применений ТК представляет собой автономное устройство, которое либо подключено по локальной сети к группе контроллеров, либо переносится от одного контроллера к другому.

Аппаратные средства ТК настраиваются на особенности применения за счет специализированной комплектации функциональными модулями. На рис. 1 изображена структура универсального технологического комплекса в максимальной комплектации. Типовая структура проблемного комплекса выделена пунктирной линией. Аппаратура устройства передачи данных (УПД) обеспечивает доступ к ресурсам стационарных ЭВМ через локальную сеть, а цифровые каналы устройства связи с объектом (УСО) — подключение кассетного магнитофона. Структура целевой микроЭВМ или контроллера, для которого ведется разработка программного обеспечения на проблемном ТК, выделена на рисунке штрихпунктирными линиями. Конфигурация контроллера может быть расширена, например, функциональным моду-

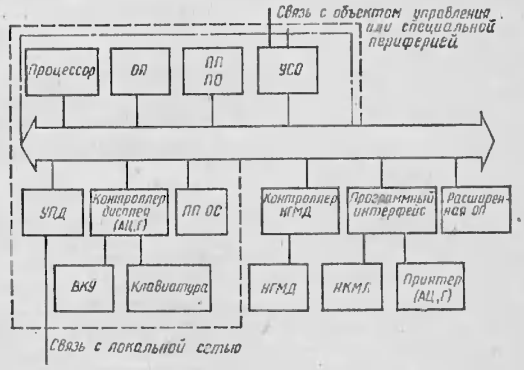


Рис. 1. Структура аппаратных средств универсального и проблемного ТК

лем дисплейного адаптера для применений, использующих человеко-машинную диалоговую процедуру управления объектом.

Программное обеспечение профессиональных технологических комплексов

Программное обеспечение ТК — интегрированная система [5, 6], содержащая: транслятор проблемно-ориентированного языка, обеспечивающий статический контроль типов и программных интерфейсов; гибкую операционную среду, адаптируемую к окружению; символический отладчик; сборщик-конкретизатор программного продукта и проблемно-ориентированную информационную базу.

Построение программного обеспечения осуществлено на основе концепции открытой (расширяемой) системы [7, 8]. В ядро системы входит пакет встроенных понятий: действий, типов данных, констант и механизм расширения. Все остальное программное обеспечение представляет собой расширение ядра. Процесс создания программного обеспечения ТК, ориентированного на проблемную область, заключается в сборке новой интегрированной системы [9] из действий и типов, накопленных в информационной базе универсального технологического комплекса, а также созданных на базе языковых средств ядра или накопленных пакетов. Сборкой нового проблемного ТК занимается проблемный программист, который формирует перечень объектов, непосредственно использованных в проблемно-ориентированном языке. Действия и типы данных, на базе которых построены объекты, подключаются к интегрированной системе в процессе сборки автоматически. Созданный таким образом ТК поддерживает сборочный подход к разработке программного обеспечения проблемных систем [10].

Сборочная технология разработки программного обеспечения в профессиональных технологических комплексах

Существенная особенность итеративного процесса разработки программного продукта (рис. 2) — охват средствами интегрированного инструментария практически всех этапов разработки и сопровождения программного продукта. Это позволяет силами небольшой группы непрофессиональных или проблемных программистов вести на проблемно-ориентированном языке разработку программных проектов среднего размера.

Описываемый технологический процесс базируется на событийной модели, разрабатываемой для каждой прикладной программной системы и развиваемой от этапа разработки спецификаций до этапа сопровождения. Функциональные возможности системы, объявленные в техническом задании, реализуются сетью модулей («черных ящиков»), связи между которыми определяются реакциями модулей на воздействия окружающей среды. Каждый модуль сети (как это делают программные модули в языках программирования) обеспечивает переработку входной информации в выходную. Вместе с тем каждый модуль наряду с обработкой событийной сети классифицирует результат по схеме событий, заданной при описании модуля, и для каждого класса результатов выдает особую реакцию-ситуацию. Поскольку систематизация производится внутри модуля, извне о его реакции на воздействие можно судить по использованию отдельного управляющего выхода из модуля, обеспечивающего «активную» реакцию на событие соответствующего класса. Таким образом, модуль представляет собой «черный ящик», в котором управляющие выходы в соответствии с классификацией результатов представляют полную группу событий на выходе. Такое строение модуля, с одной стороны, скрывает способ реализации — алгоритм выдачи реакции на воздействия, с другой — с помощью свойства полноты гарантирует при пользовании модулем отсутствие каких-либо неучтенных реакций на его выходе.

Внутреннее устройство модуля реализуется в процессе нисходящей детализации его функций с помощью других модулей, устроенных аналогичным образом. Рассмотрим этапы технологической схемы разработки ПО.

Анализ требований — формализуются и документируются основные ресурсные ограничения и критерий качества программной системы, подлежащей разработке (средствами автоматизации не поддерживается); выбранные требования, цели, критерии проектирования программного продукта фиксируются в эксплуатационной документации.

Разработка спецификаций — создается и документируется описание разрабатываемой системы, а также определяются и фиксируются способы контроля соответствия спецификации ее последующей реализации [11, 12].

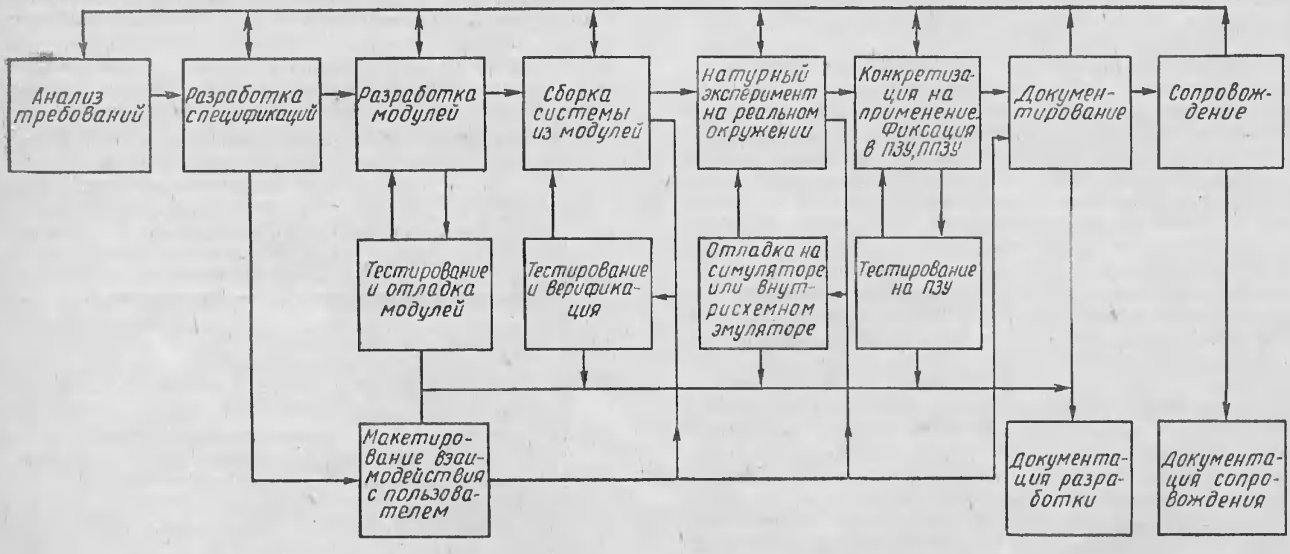


Рис. 2. Технологическая схема разработки ПО для встроенных применений

Модуль технологической схемы описывает действие, тип данного или само данное. На каждый модуль разрабатывается спецификация, ее текст хранится в информационной базе и содержит: аннотацию (неформальное и краткое описание целей и функциональных возможностей модуля с точки зрения пользователя), описание интерфейса (видимая часть модуля охватывает описание действия или типа данного вместе с указанием типов входных и выходных параметров, а также список ситуаций, вырабатываемых модулем [9]), описание эталона и в ряде случаев детальную спецификацию.

Если из спецификации, заданной только интерфейсом, не следует достаточно прозрачного алгоритма для реализации модуля на языке программирования, то применяется пошаговая детализация спецификаций. Детальная спецификация — сеть, состоящая из модулей, на базе которых построен данный (список модулей, используемых при создании данного, фиксируется). Связь между модулями осуществляется в соответствии с правилами конструктивной логики программ.

Поскольку для каждого модуля в процессе детализации интерфейс не изменяется, полученная сеть спецификаций на любом уровне детализации образует «каркас», который, оставляя гибкость в функциональном наполнении модулей (т. е. в выборе способа реализации), гарантирует «жесткость» междумодульных связей во всем процессе разработки системы.

Модификация спецификации модуля допустима лишь без изменения интерфейса. Изменение интерфейса означает замену модуля и приводит к корректировке каркаса. Процесс пошаговой детализации спецификаций продолжается до тех пор, пока в информационной базе не будут найдены уже реализованные модули с такими же спецификациями или полученная спецификация не будет базироваться на модулях, достаточно простых для реализации.

Важнейший компонент любой спецификации — способ контроля специфицируемого модуля. Необходимо определить объект и дать задание на его реализацию, а также задать способ проверки соответствия реализации и спецификации.

Эталон описывает процедурный способ контроля каждого шага детализации. Поддерживаются три способа задания эталона:

задание тестов, позволяющих контролировать реализацию с помощью известных результатов, полученных для некоторых точек (состояний системы);

задание макета [13], т. е. программной модели, позволяющей на уровне детализации, определенной в спецификации, проводить эксперименты, предназначенные для отработки интерфейса взаимодействия с моделью или реальным окружением и пользователя с системой на ранних стадиях проектирования. В результате эксперимента с макетом можно в любой момент сформировать множество тестовых последовательностей, которым с определенной степенью детализации должна будет удовлетворять реализация;

задание тестов, позволяющих контролировать свойства модели или утверждений, распространяющихся на классы результатов. Поскольку реализация действия над любым объектом в системе поддерживает разделение всего множества значений выходных величин на классы эквивалентности, принадлежность к которым идентифицируется соответствующим множеством ситуаций, то решение задачи контроля некоторого свойства сводится к решению задачи достижимости [14]. Для этого формулировка соответствующего свойства осуществляется в процедурном виде, во введенных на соответствующем уровне детализации понятиях, а контроль его правильности — на множестве классов совместных выходных данных всех входящих в модель действий. Контроль реализуется на множестве экспериментов-тестов, которые подобраны так, что на них получают все ситуации, входящие в совместные классы результатов. Достижимость любой ситуации, которая приводит к аномальным с точки зрения спецификации результатам, является критерием противоречивости спецификации и реализации. В результате зафиксированная в эталоне совокупность тестов гарантирует проверку непротиворечивости реализации со спецификацией с точностью до детализации.

Разработка модулей — на основе спецификации создаются отдельные модули, не имеющие реализации в системе проектирования; их спецификации хранятся для последующего многократного использования в процессе сборки. Реализация модулей производится методами синтезирующего программирования [10] на одном из универсальных языков программирования или на языке сборки, который является одновременно средством программирования и средством включения новых модулей в ТК.

Главное в языке сборки — не набор модулей (операторов, типов данных), который зависит от проблемной ориентации, а общие правила построения, расширения и контроля новых понятий. В целях сокращения объема ядра ТК синтаксис языка сборки предельно упрощен и содержит средства описания интерфейсов модулей, создания новых типов данных, новых действий, создания и обработки ситуаций [9].

Например:

описание действия:
ДЕЙСТВИЕ ⟨имя⟩[⟨имя_параметра⟩:⟨тип...⟩]/[⟨ситуация⟩...]
описание команды-действия, подключаемого в виде кодовой процедуры:
КОМАНДА ⟨имя⟩⟨адрес_входа⟩[⟨тип⟩]/[⟨ситуация⟩...]
описание встроенного типа:
ТИП ⟨имя⟩⟨имя_пакета_процедур_генерации_ввода-вывода⟩
описание конструируемого типа:
КОНСТРУКТОР ⟨имя⟩[⟨имя_параметра⟩:⟨тип⟩...]
описание ситуации:
ПРИ ⟨имя_ситуации⟩⟨имя_действия⟩

В ядре всегда присутствует встроенный пакет, содержащий минимальный набор действий и типов данных. Понятия ядра неявно входят в контекст любого модуля.

В интегрированном технологическом комплексе разработанный на языке сборки модуль проходит этап «ввода — редактирования — трансляции», в процессе которого обеспечивается статический контроль типов в теле модуля и контроль внешнего интерфейса. Контроль функционирования обеспечивается на этапе «тестирования — выполнения», где поддерживается выполнение множества прогонов по тестам, заданным в спецификации, и сравнение экспериментально полученных результатов с эталонными. Расхождения анализируются в процессе символьной отладки, где обеспечивается регулировка видимости уровня детализации объектов, проведение экспериментов над отдельными объектами, возможность проводить анализ состояния системы в заданных пользователем точках и условиях. Если в процессе отладки выявляются причины расхождений, то для исправления исходного текста происходит переход к «вводу — редактированию — трансляции», после чего продолжается «тестирование — выполнение». Может случиться так, что аномалия потребует увеличения множества тестов эталона. В этом случае происходит корректировка спецификации с помощью новых данных, которые можно получить на макете. Пополнение эталона новыми тестами не может нарушить каркас систе-

мы, поскольку интерфейс модуля сохраняется неизменным.

Модули, созданные в рамках других технологий, проходят этап автономной отладки в соответствующих системах программирования и используют язык сборки лишь на этапе включения в сборочную систему.

Сборка — основной способ разработки программных систем в ТК. Ее цель — получение программной системы из реализованных ранее модулей на основе спецификации, а также комплексная отладка и верификация созданной системы. На этом этапе применяется сборочная форма программирования, которая «основана на существующих полуфабрикатах и реализует принцип многократности использования модулей программного обеспечения» [10]. В процессе сборки осуществляется: трансляция модуля системы через реализацию входящих в контекст модулей; объединение в теле модуля тел, входящих в контекст, и сборка автономной системы; конкретизация, позволяющая ограничить объем собранной автономной системы до объема используемых в ней компонент. Реализованные модули программных систем подключаются к информационной базе, где в случае необходимости комплексируются в пакеты.

На этапе сборки поддерживается процесс проектирования программного обеспечения методом сверху вниз, так как тела модулей описываются отдельно от интерфейсов. Тела реализуются на языке сборки и в кодах соответствующей микроЭВМ в рамках традиционной технологии, с помощью любых языков, но с соблюдением интерфейсных соглашений на способы вызова и возврата модуля со стороны системы, передачи параметров, задания и обработки ситуации.

Если модуль включен в системы с интерфейсом, отличным от искомого, то его используют в теле модуля, обладающего требуемым интерфейсом. В объемлющем модуле можно скорректировать число и типы входных или выходных параметров, а также список выдаваемых модулем ситуаций. Если вложенный модуль выдает ситуации, не перечисленные в интерфейсе объемлющего, то их обработка обязательно должна быть осуществлена в теле объемлющего модуля. В процессе вычислений появление любой ситуации, не предусмотренной в интерфейсе, квалифицируется как аномальное событие. Поскольку в сборочной системе компоновка осуществляется на базе накопленных, а не специально синтезированных модулей, то в классы аномальных попадают те нормальные для модуля классы результатов, которые не используются в конкретной системе. Для успешной сборки крайне важно обеспечить свойство полноты при классификации результатов и формирования множества выходных ситуаций (учитывающих логику алгоритма, ресурсные ограничения и конечность исполнения) и при обработке ситуаций, перечисленных в интерфейсах модулей, собранных в систему.

Реализованная программная система подлежит контролю на соответствие спецификации. Для этого тесты, входящие в спецификации собранных модулей, объединяются в набор — пирамиду тестов (рис. 3), которая образует эталон модуля. Этот набор используется для реализации тестовых прогонов и определения расхождений на модульном уровне между экспериментально полученными и зафиксированными в эталоне результатами.

Критерием полноты тестирования сети модулей является полный перебор ситуаций, возникающих на выходе каждого модуля, использованного при построении сети. Достижимость каждой ситуации гарантируется реализацией всего множества тестов эталона. Например, для сети (рис. 3) перебор всех ситуаций достигается при выполнении экспериментов над модулями 1, 2, ..., 11. Число экспериментов определяется числом тестов, входящих в тестовые наборы модулей (T_1). Совокупность тестовых наборов T_1, T_2...T_{11} образует пирамиду тестов модуля 11.

Причины найденных в процессе тестирования расхождений со спецификацией анализируются в процессе символьной отладки, допускающей регулирование уровня детализации, что позволяет проводить эксперименты только над теми объектами, видимость которых открыта для наблюдений.

Натурный эксперимент — контролируется взаимодействием системы с аппаратным и программным обеспечением в реальном времени. Цель этапа — подтверждение работоспособности и соответствия спецификациям при работе с окружением в реальном времени и с учетом других ресурсных ограничений. Методика контроля, поддерживаемая в ТК, основана на проверке правильности протоколов асинхронных взаимодействий программных процессов, свойства которых задаются в спецификации с помощью алгебры протокольных выражений [15], формируются спецификации на последовательности входных и выходных событий. Учет в протокольных выражениях ограничений на время реакции, обеспечиваемых за счет таймирования, гарантирует реализацию взаимодействия с окружением без зависаний. Проверка осуществляется на базе серии экспериментов, которые дают протокол входных и выходных событий, сравниваемый с эталонным. С помощью этой методики можно контролировать выполнение и других ресурсных ограничений. Ее действие можно распространить на синхронные протоколы, если в протокольных выражениях учитывать не только последовательности событий, но и ограничения на абсолютные времена их выполнения.

Сложность контроля последовательностей синхронных событий заключается во внесении инструментальными средствами временных искажений в протокол входных и выходных событий. Решение этой задачи в ТК возможно в двух частных случаях: когда точность фиксации интервалов времени между синхронными событиями не превышает временных затрат на работу инструментальных средств в процессе реализации протокола и когда асинхронные события в протоколе перемежаются с синхронными, что позволяет использовать инст-

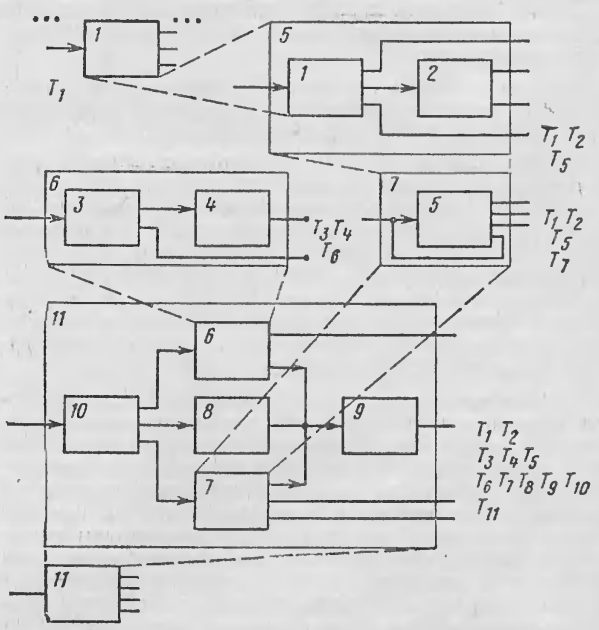


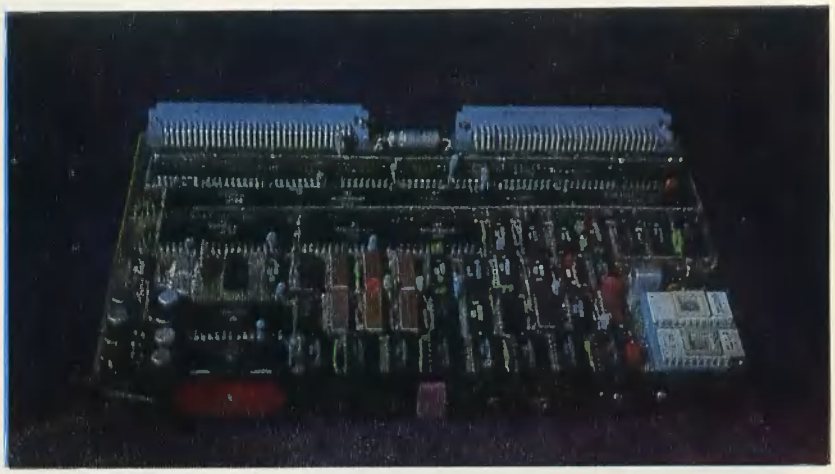
Рис. 3. Структура конкретного модуля, полученного по сборочной технологии

МИКРОПРОЦЕССОРНАЯ СИСТЕМА МС-80

(К ст. Королева В. Н., Жаркова А. С., Зубченко А. П., Штанакова А. Ю.)

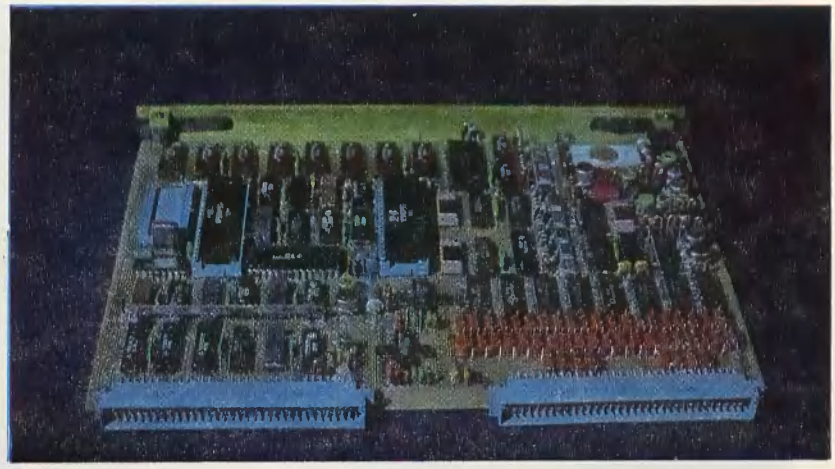
МС-80 — базовая агрегатируемая система, предназначенная для построения программируемых контроллеров (ПК) и распределенных систем управления. Технические средства МС-80 выполнены в виде функционально законченных модулей, компонуемых в общем каркасе. Микросхемы высокой степени интеграции обеспечивают при небольших габаритных размерах широкие функциональные возможности ПК. В зависимости от сложности задач изменением числа модулей можно компоновать ПК с различным числом входных-выходных дискретных и аналоговых сигналов. Буквенно-цифровой дисплей и клавишная панель встроены. К ПК можно подключить видеотерминал и печатающее устройство со стандартными интерфейсами ИРПР, ИРПС или «токовая петля 20 мА». Это позволит контролировать и регистрировать большое количество информации.

Программируемый контроллер на базе МС-80



Модуль управления

Модуль ввода-вывода аналоговых сигналов

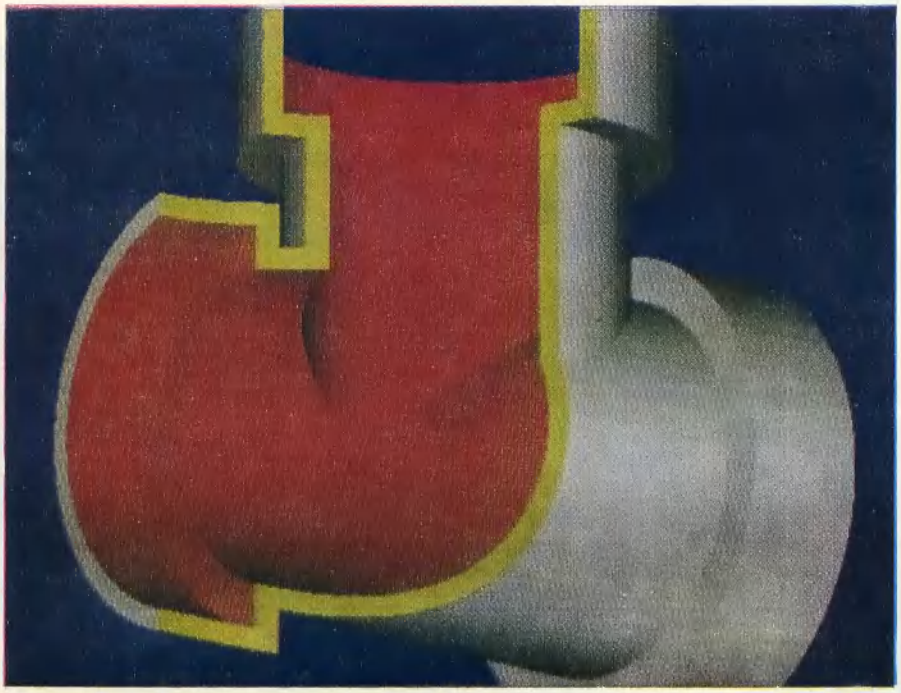


ВЫСОКОСКОРОСТНОЕ АВТОМАТИЗИРОВАННОЕ КОНСТРУИРОВАНИЕ С ИСПОЛЬЗОВАНИЕМ ВЕКТОРНОГО ПРОЦЕССОРА

(К ст. Алексеенко М. М. и др.)

В машиностроительных САПР представление геометрии объектов как совокупности пространственных примитивов наиболее эффективно с точки зрения автоматизации всего цикла конструкторских работ. Однако для получения изображений в этом случае требуется объем вычислений, составляющий десятки миллионов операций с плавающей запятой.

Одним из путей достижения требуемой производительности АРМ для машиностроительных САПР является включение в его архитектуру векторного процессора и создания программного обеспечения визуализации на основе параллельных вычислений. Высокая производительность векторного процессора (до 12 млн. опр/с) и параллельные алгоритмы позволяют снизить время получения изображений до 10 с.



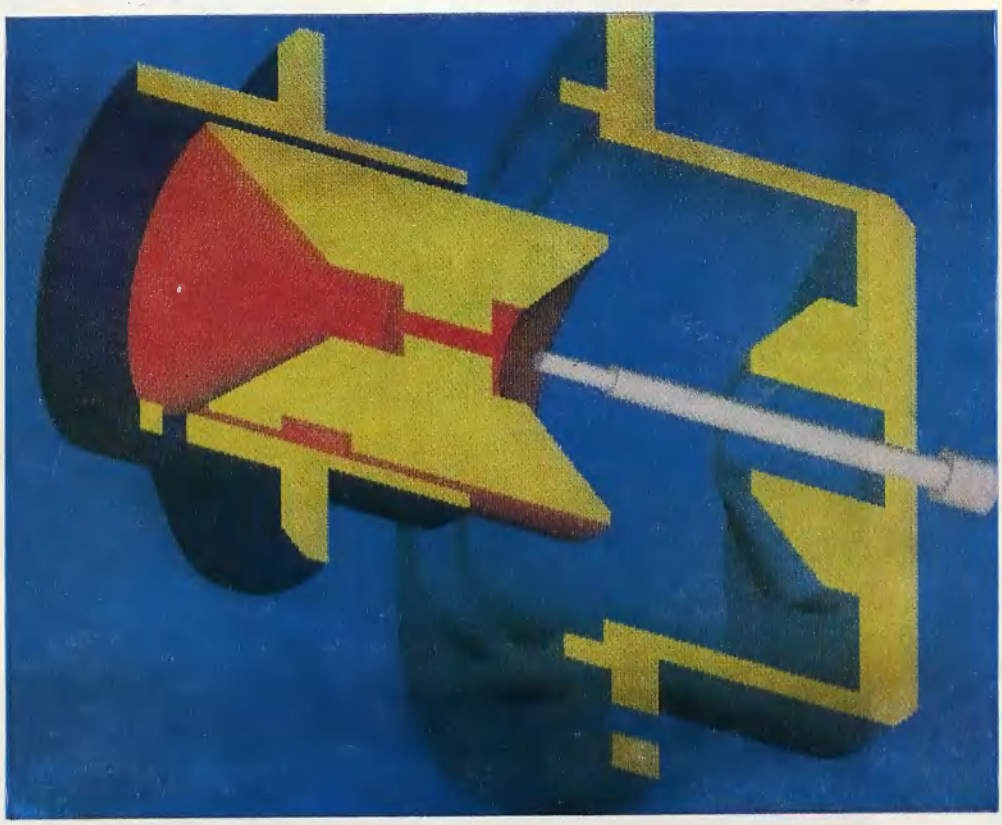
В системе интерактивного геометрического моделирования (СИГМа) объект представляется как результат выполнения операций над пространственными примитивами (параллелепипед, сфера, цилиндр, конус и др.). Программы СИГМы реализованы на векторном процессоре, что обеспечивает получение реалистических изображений объектов за 10 с

Одновременный расчет изображения по 1024 точкам позволяет за десять секунд получать изображения объектов и их сечений при любых изменениях масштаба, освещенности и положения объектов, обеспечивая пользователю наиболее полное представление о конструируемом объекте

На первом этапе работ, проводимых в Институте Автоматизации Проектирования АН СССР, основное внимание уделяется:

— разработке системы интерактивного геометрического моделирования для автоматизированного проектирования (СИГМа);
— разработке проблемно-ориентированного машиностроительного языка конструирования (МАЯК), являющегося интерактивной надстройкой над СИГМой.

Результаты работы показывают, что, в зависимости от сложности объекта время построения его изображения может быть снижено до 5...20 с. При этом в САПР обеспечивается интерактивное взаимодействие в терминах геометрических и конструкторских операций.



Наличие базы геометрических данных позволяет автономно конструировать фрагменты сложных объектов, а затем производить их «сборку». Возможность быстрого получения различных реалистических изображений объекта изменяет представление о конструкторской документации в САПР, в частности, сборочных чертежах

Укажите координаты позиционирования компоненты

1. X=0.0
2. Y=0.0
3. Z=10.0
4. Psi=0.0
5. Theta=0.0
6. Phi=0.0

Машиностроительный язык конструирования (МАЯК) является интерактивной надстройкой над СИГМой и обеспечивает эффективное взаимодействие пользователей САПР в терминах геометрических и конструкторских операций

〈МАЯК〉: Работает программа визуализации «СИГМА»

ТРЕХУРОВНЕВЫЙ КОМПЛЕКС ДЛЯ АВТОМАТИЗАЦИИ НАУЧНЫХ ИССЛЕДОВАНИЙ НА РАДИОТЕЛЕСКОПЕ РАТАН 600

(К ст. Ерухимова Б. Л., Черненкова В. Н.)

Подвижный облучатель со вторичным зеркалом

Общий вид радиотелескопа

Фото Сухарева Ю. В.

рументальные средства на фрагментах программы, обеспечивающих обработку асинхронных событий. Возможности второго случая ограничены тем, что о правильности синхронных взаимодействий можно судить только по состояниям переменных, фиксируемым на входах и выходах синхронных фрагментов программы.

Система поддерживает накопление информации, полученной в процессе тестирования-отладки. Пакет сбора статистики по программным компонентам входит в информационную базу. Наглядное представление об интегральных результатах тестирования системы дает изображение сети модулей, нагруженное статистической информацией.

Конкретизация к применению — обеспечивает управление ресурсной эффективностью программного продукта за счет использования информации, конкретизирующей условия применения системы. Сочетание методик сборки и конкретизации при построении программных систем позволяет эффективно использовать продукт, накопленный в прежних разработках [16].

Основной критерий, удовлетворяемый на этом этапе,— корректное преобразование программного продукта, учитывающее выполнение ресурсных ограничений, в особенности для управляющих программ микроЭВМ, сформулирован в работе [16].

Практически полезность инструментальной системы оценивается сокращением объема работы по созданию ПО проблемных систем на конкретном языке программирования. Этот факт является причиной включения в инструментальную систему большого числа универсальных средств: пакетов прикладных и системных подпрограмм, операторов, директив, типов и других объектов, полезность которых тщательно выверена на широком круге применений. Однако связанный с расширением предоставляемых возможностей рост объема памяти инструментальных систем тяжелым бременем ложится на портативные ТК. Сохранить в них средства сокращения текста разрабатываемых программ в условиях ограничений на ресурсы возможно, если наряду со средствами расширения языков конструкций ввести средства усечения возможностей инструментальной системы, не используемых в конкретном применении. Например, можно сократить список транслируемых операторов, список распознаваемых операционной системой директив, список модулей в библиотеке и т. п.

На этой же фазе в ТК выполняется преобразование программного обеспечения для фиксации в постоянную или полупостоянную память.

Документирование — значительный этап процесса разработки — формирует необходимую документацию на программный продукт в соответствии с требованиями ЕСПД и осуществляет сборку в текстовые файлы на магнитном носителе. При этом программная и эксплуатационная документация, полученная на предыдущих этапах, соединяется с документацией, разработанной вручную.

Сопровождение — этап, поддерживаемый в ТК путем сборки и передачи пользователю специальной версии исполнительной программной системы вместе с фрагментом откладчика и операционной среды, настроенными на выдачу информации о состоянии проблемной системы в ограничениях (описанных в техническом задании) на уровень видимости объектов, а также на точки контроля и формат выдачи состояний. Это позволяет на этапе сопровождения фиксировать аномальные события в работе системы силами пользователя в понятиях соответствующей проблемной области.

Основные этапы описанной технологии были проверены и внедрены в рамках работ по реализации семейства резидентных универсальных ТК на базе серийных микроЭВМ 15 ВУМС-28-025, ДВК-2 (2М), «Электроника С5-21М», «Электроника С5-41» и создания с их помощью проблемных ТК для химико-технологических процессов, систем вентиляции и кондиционирования, управления промышленными работами, испытания электрических машин, контроля систем управления воздушным движением, сетей магистральной связи, электронных АТС, систем обучения.

Примеры применения ТК

На рис. 4 изображен фрагмент программы управления цеховой системой приточной вентиляции, в рамках которого реализуется автоматный закон управления двумя системами 05П007, 05П008, обеспечивающими регулирование с необходимой точностью температуры и давления в рабочей зоне цеха. Отображение текущего состояния эксплуатируемой системы осуществляется на мнемосхеме, изображенной на экране графического видеоконтрольного устройства. В ней явно отображается динамика переключения клапанов К1...К6 и значения регулируемых параметров. В функции проблемного ТК входит создание средств автоматизации проектирования программного обеспечения для конкретных систем приточной вентиляции производственных помещений и средств программирования законов управления цеховой системой в соответствии с требованиями технологии производственного процесса.

В процессе решения подобной задачи проблемный программист создает типовую схему управляющей программы и новую мнемосхему, используя базовые фрагменты изображения и примитивы управления, хранящиеся в информационной базе проблемного ТК. Он также создает программу, позволяющую вводить новый план управления с помощью таблицы, заполняя ее клетки прямо на дисплее управляющей установки. С эксплуатацией управляющей установки и ее табличным перепрограммированием справится уже непрофессиональный программист или конечный пользователь.

Язык программирования и управления проблемного ТК был сформирован по техническому заданию заказчика и ориентирован на создание контроллеров для систем, построенных на базе типовой номенклатуры установок для кондиционирования, вентиляции и подогрева помещений.

Другой пример приведен на рис. 5, где на левом экране изображен фрагмент программы управления простейшим технологическим процессом штамповки, а на правом — текущее состояние макета робота. Макет функционирует на отдельной микроЭВМ и соединяется с контроллером специальной косой. Поскольку коса соединяет управляющие выходы контроллера с управляющими входами макета, а выходы, имитирующие сиг-

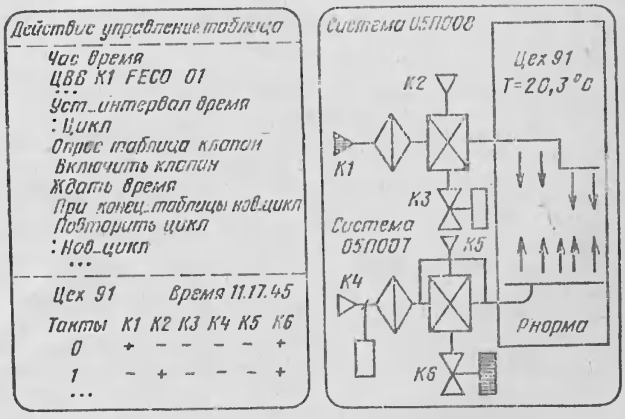


Рис. 4. Проблемный ТК для разработки ПО контроллеров, регулирующих температуру и давление в рабочей зоне цеха

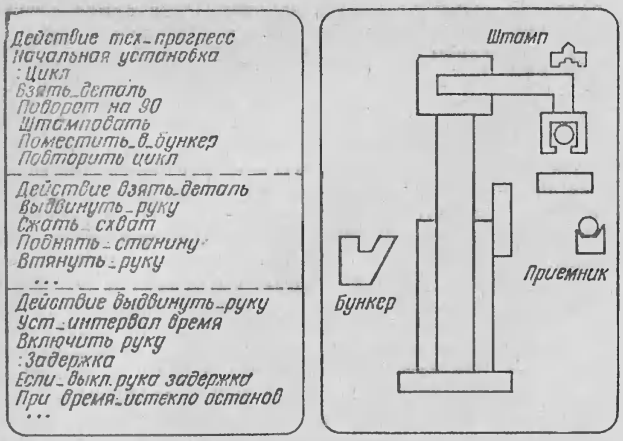


Рис. 5. Проблемный ТК для разработки ПО цикловых роботов и робототехнических комплексов

налы датчиков обратной связи макета,— с соответствующими входами контроллера, то на таком двухмашинном комплексе можно осуществить комплексную, а в ряде применений и натурную отладку для широкого класса цикловых роботов.

В рассмотренном примере задание нового техпроцесса осуществляется путем перепрограммирования. Хотя текст новой управляющей программы ТЕХПРОЦЕСС при этом изменится, ее компоненты, такие как ВЗЯТЬ — ДЕТАЛЬ, ВЫДВИНУТЬ — РУКУ, ВКЛЮЧИТЬ, УСТАНОВИТЬ — ИНТЕРВАЛ, не изменяются. Они являются модулями многократного применения и используются непрофессиональным программистом для составления новой технологической программы. Язык программирования для подобного ТК был создан в процессе разработки контроллеров, предназначенных для систем управления цикловыми роботами и робототехническими комплексами.

Сравнение с другими инструментальными системами

Сравнение характеристик ТК [4] с характеристиками инструментальных систем на универсальных языках программирования: ассемблер, Паскаль, Фортран, FIG FORTH, Бейсик, Quasic операционной системы ОС ДВК проводилось на задачах управления для различных об-

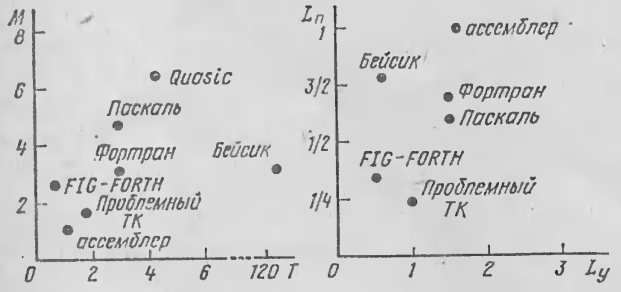


Рис. 6. Характеристики эффективности проблемных ТК по сравнению с традиционными системами автоматизации разработки ПО для одинаковых областей применения

ластей применения и показало существенный выигрыш по ресурсной эффективности продукта и производительности труда в проблемно-ориентированном ТК по сравнению с универсальными системами программирования. В качестве метрик ресурсной эффективности (рис. 6) программного продукта были использованы относительные показатели объема занимаемой памяти (M) и времени исполнения (T) программы на некотором языке программирования по отношению к характеристикам реализации той же программы на ассемблере. Трудоемкость и интегральность оценивались параметрически по относительным показателям холстедовской длины программы (L_n) и длину протокола (L_y), требуемого для эксплуатации программного изделия. На рис. 6 приведены показатели, вычисленные на контроллерных программах общим объемом в 140К байт. Результаты, полученные для частных проблемных областей применения ПТК, подтверждают вывод о перспективности применения проблемных технологий для разработки ПО встроенных микроЭВМ.

Отметим свойства профессиональных ТК и поддерживаемой ими сборочной технологии программирования, важные для пользователей:

дружественность, достигаемая простотой эксплуатации комплексов и человеко-машинным интерфейсом (ориентированным на поддержку проблемной терминологии при программировании и управлении вычислительным процессом); предотвращением ошибок и нейтрализацией ошибочных ситуаций; использованием графических мнемосхем, полиэкранов, меню;

надежность, гарантируемая статическим контролем типов, интерфейсов, контролем полноты обработки ситуаций, использованием абстрактных типов данных; контекстной защитой доступа к программным ресурсам;

интегрируемость, получаемая за счет единого процесса «ввода — редактирования — трансляции» либо «тестирования — выполнения» на каждом шаге проектирования программной системы. Например, в интегрируемой системе КОМФОРТ [9] трансляция и связывание производится автоматически без дополнительного обращения пользователя к соответствующим процедурам операционной системы. Набранная программа или ее фрагмент сразу же могут быть выполнены;

символьная отладка обеспечиваемая на всех уровнях и для всех объектов целевых программ. Пользователь имеет возможность регулировать уровень детализации и расширять возможности системы отладки;

расширяемость, достигаемая введением новых типов данных и операций над ними в рамках соответствующего контекста. Разрешается вводить новые понятия на базе существующих или создавать их как встроенные с помощью языка ассемблера;

ресурсная эффективность, получаемая с помощью сборочно-конкретизирующего программирования. Итог работы комплекса — программный продукт, способный к автономному функционированию в контроллере без операционной системы, ориентированный на проблемное применение, подготовленный для фиксации в память с односторонней выборкой. Резидентный вариант программного обеспечения ТК занимает 18К байт и может быть размещен на плате ПЗУ;

мобильность гарантируемая реализацией большей части программных средств комплекса машинно-независимым способом на базе машинно-зависимого ядра объемом в 2...4К байта. Перенос обеспечивается перепрограммированием ядра на новой микроЭВМ и автоматической раскруткой остального программного обеспечения;

интеграция со штатными операционными системами, например ДОС ДВК или ФОДОС, осуществляется путем введения программного обеспечения технологического комплекса в операционную систему в виде пакета прикладных программ сложной структуры;

способность функционирования в рамках многомашинной ассоциации реализуется средствами расширяемой операционной среды.

В настоящее время проводятся работы по созданию универсальных ТК КОМФОРТ для разработки на их основе проблемных ТК, ориентированных на применение в многомашинных системах со статистическим разделением функций и локальных сетях микроЭВМ.

Адрес для справок: 195251, Ленинград, ул. Политехническая, д. 29, ЛПИ, кафедра «Информационные и управляющие системы». Телефон: 552-97-03.

ЛИТЕРАТУРА

1. Мясников В. А. Подготовка специалистов по применению микропроцессорной техники // Микропроцессорные средства и системы.— 1984.— № 2.— С. 3—5.
2. Липаев В. В., Каганов Ф. А., Керданов Ю. В. и др. Система автоматизации проектирования программ на базе персональных ЭВМ (Система ПРА) // Микропроцессорные средства и системы.— 1985.— № 4.— С. 42—45.
3. Липаев В. В., Каганов Ф. А. Адаптируемые кросс-системы проектирования программ на базе больших универсальных ЭВМ и микроЭВМ // Микропроцессорные средства и системы.— 1984.— № 1.— С. 61—65.
4. Котляров В. П., Морозов Н. Б. Технологические комплексы разработки программного обеспечения встроенных микроЭВМ // Индивидуальные диалоговые системы на базе микроЭВМ / Персональные компьютеры / ДИАЛОГ-84-МИКРО / —Л.: Наука, 1984.
5. Витенберг И. М., Либеров Л. Б. О технологии программирования персональных ЭВМ. // Программное обеспечение и применение микропроцессорных систем и устройств.— М.: МДНТП, 1986.
6. Бетелин В. Б. О проблеме автоматизации программирования.— Докл. АН СССР.— 1986 — Т. 286.— № 1.
7. Баранов С., Кириллин В., Ноздрунов Н. Реализация языка ФОРТ на дисплейном комплексе ЕС 7970 // Программирование микропроцессорной техники.— Таллин, Валгус.
8. Берс А. Л., Поляков В. Г., Руднев С. Б. О системе программирования высокого уровня со смешанными вычислениями для персональных микропроцессорных комплексов // Актуальные проблемы развития архитектуры и программного обеспечения ЭВМ и вычислительных сетей.— Новосибирск, ВЦ СО АН, 1983.
9. Кпреев С. П., Котляров В. П., Морозов Н. Б. Резидентные средства автоматизации разработки программного обеспечения для встроенных микроЭВМ /. Программное обеспечение и применение микропроцессорных систем и устройств.—М.: МДНТП, 1986.
10. Ершов А. П. Комплексное развитие системного прогаммного обеспечения — постановка проблемы. Препринт 469.— Новосибирск, ВЦ СО АН СССР, 1983.
11. Лавров С. С. Расширяемость языков. Подходы и практика // Прикладная информатика.— М.: Финансы и статистика, 1984, вып. 2.
12. Агафонов В. Н. Языки и средства спецификации программ // Требования и спецификации в разработке программ.— М.: Мир, 1984.
13. Громов Г. Р. Национальные информационные ресурсы: проблемы промышленной эксплуатации.— М.: Наука, 1984.— 240 С.
14. Bardzin J. M. The problem of reachability and verification of programs // Lec. Notes in Comp. Sci., 1979 № 74.
15. Дробинцев Д. Ф., Котляров В. П. Использование протокольных выражений в спецификациях программного обеспечения встроенных микроЭВМ для повышения надежности в функционировании программных модулей // Тез. докл. Всес. научно-техн. конф. «Программные средства как продукция производственно-технического назначения». Секция: технология разработки программных средств.— Калинин, 1985.
16. Котляров В. П., Морозов Н. Б. Особенности алгоритмов смешанных вычислений для управляющих программ микроЭВМ // Трансляция и преобразование программ.— Новосибирск, ВЦ СО АН СССР, 1984.

УДК 681.3.06

С. О. Бочков, Р. Л. Смелянский

СИМВОЛЬНЫЙ ОТЛАДЧИК ДЛЯ ЯЗЫКА Си

Отладка — наиболее трудоемкий этап создания программного продукта, занимающий 30...50 % времени создания программы [1—4] и 15...40 % затрат на сопровождение [4]. При этом в течение последних двадцати лет затраты на отладку не сокращаются, а остаются на одном уровне [5, 6].

К настоящему времени существует ряд мощных отладочных систем [7] для вычислительных систем с богатым набором ресурсов и периферийного оборудования [8, 9]. Для «бедных» ресурсами микроЭВМ развитие отладочных средств прослеживается в следующих направлениях:

создание простейших диалоговых систем для отладки в кодах или в терминах ассемблерных команд [10, 11], которые позволяют выполнять программу с любого оператора, в пошаговом режиме, устанавливать контрольные точки и обеспечивают доступ к памяти по адресам (для работы необходимо иметь таблицы с адресами переменных);

включение отладочных средств в интерпретаторы и трансляторы с языков высокого уровня [12, 13] (для отладки в неинтерактивном режиме);

создание диалоговых отладчиков для языков высокого уровня, позволяющих вести отладку в терминах исходного языка [13, 14]. Такие системы являются наиболее удобными для пользователя, но и они имеют недостатки: построение на базе трансляторов, из-за чего при каждом исправлении программы в ходе отладки ее необходимо перетранслировать; отладчик и редактор существуют независимо; отладчики обычно требуют 256К байт и более [13], которой не обладает большинство серийно выпускаемых в нашей стране микроЭВМ.

Для устранения этих недостатков разработан символьный диалоговый интерпретирующий отладчик для языка Си [15], позволяющий работать с программами, прошедшими препроцессорную обработку (стандартный препроцессор языка Си). Отладчик реализован на языке Си в ОС РАФОС на микроЭВМ «Электроника 60», имеющей оперативную память 56К байт (объем отладчика — 26К байт без транслятора программы во внутреннее представление). Завершается реализация многооконного интерфейса.

В настоящее время осуществлен перенос отладчика на ПЭВМ «Корвет», но из-за ограниченности ресурсов ПЭВМ эффективное использование возможно лишь на следующих моди-

фикациях с увеличенным объемом памяти.

Возможности отладчика

Отладка программы ведется в диалоговом режиме. Пользователь набирает на экране терминала команду и после ее выполнения получает приглашение ввести следующую. При этом на экране могут отображаться: текст программы, трасса ее выполнения, значения переменных и т. д.

Отладчик предоставляет следующие возможности:

выполнение программы с любого места;

пошаговое выполнение;

трассировка специально указанных операторов, операторов управления, вызовов функций либо всех операторов;

установка статистических и динамических (по выполнению условия) контрольных точек;

обратный ход (воссоздание состояния программы на несколько операторов назад);

доступ к переменным в описанном виде (включая массивы и структуры);

проверка соответствия формальных и фактических параметров;

задание команд отладчика, которые должны выполняться после каждого оператора программы при истинности соответствующих условий;

редактирование программы в терминах языка.

Команды отладчика:

ИДИ — выполнение программы с точки останова, можно явно указать, с какого оператора (ключ НАЧАТЬ). Можно задать останов по достижению определенных операторов (ключ КОНЧИТЬ), выполнение команд отладчика (кроме ИДИ, ШАГ) после каждого оператора программы при истинности условного выражения (ключ ЕСЛИ), условное выражение здесь — это выражение в терминах языка Си (без обращения к функциям), но со специальными операциями ЧТЕНИЕ() и ЗАПИСЬ() контроля обращения к переменным. Ключей ЕСЛИ может быть несколько. Ключ ДАЛЬШЕ — действие ключей КОНЧИТЬ и ЕСЛИ из последней выполненной команды ИДИ будет продолжаться. УСЛОВИЕ — запоминает условие и присваивает ему символическое имя. ПРИСВОИТЬ — присваивает значения переменным. ВЫДАТЬ — показывает значения переменных. ТРАССА — устанавливает режим трассировки. ГДЕ — адрес точки останова. ЧТО — распечатывает значения доступных в данной точке переменных. ШАГ — выполняет указанное количество операторов программы. Использование команды с отрицательным аргументом (для этого необходимо заранее установить специальный режим) позволяет воссоздать состояние программы на несколько операторов назад (обратный ход). ДВИГАТЬ — передвигает точку останова. ЗАПОМНИТЬ — устанавливает точки запоминания состояния памяти. Возможности, предоставляемые этой командой, сходны с возможностями метода обратного хода, но в этом случае мы работаем со статистическими точками в программе. При выполнении программы (по команде ИДИ или ШАГ) с точки запоминания состояния памяти или при передвижении на нее точки останова (по команде ДВИГАТЬ) восстановится содержимое памяти, соответствующее последнему прохождению через эту точку. РЕДАКТОР — вызывает внутренний редактор программ. Основные операции редактора: вставить, заменить, удалить оператор. МАКРО — запоминает последовательность команд отладчика. ФАЙЛ — можно изменять конфигурацию отлаживаемой программы (можно работать с неполной программой, когда определены не все функции и внешние переменные). ЛИСТИНГ — работа с текстом программы. СТОП (<CTRL/C>) — прервать очередную команду отладчика. ? — справки по отладчику. КОНЕЦ — окончание сеанса.

Можно установить режим проверки соответствия типов формальных и фактических параметров.

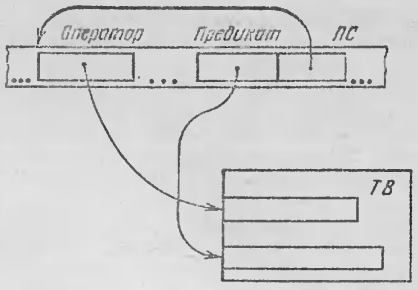
Допускается сокращение имен команд и ключей до одной буквы.

Отличительной чертой является ориентация командного языка на события (выполнение какого-либо оператора программы или истинность некоторого утверждения о значениях переменных, а также их изменении и использовании). Это сокращает объем информации, с которой работает программист во время отладки. Увеличена мощность средств отладки.

Внутренняя организация

Блочная структура и сложная организация управления в Си замедляют интерпретацию исходного текста, поэтому интерпретируется специальное внутреннее представление (ВП) программы, которое является новым для подобных приложений. В основу ВП легла идея разделения управляющей и вычислительной частей программы. За счет свойств этого представления удается автоматически, без перетрансляции корректировать ВП при изменении текста программы.

Внутреннее представление (ВП) выбиралось, исходя из следующих критериев: скорость интерпретации; простота и скорость трансляции исходного текста в ВП; простота автоматического внесения исправлений в ВП при редактировании программы, т. е. чтобы локальные изменения в исходном тексте влекли за собой локальные изменения в ВП; компактность ВП; взаимная однозначность соответствия операторов исходной элементам ВП (для точной диагностики в терминах языка); независимость от входного языка.



Структура внутреннего представления программы.

При трансляции программы в ВП проверяется синтаксис исходного текста и собирается информация, необходимая при отладке.

Внутреннее представление (см. рис.) состоит из логической схемы (ЛС) и таблицы выражений (ТВ). Логическая схема разработана на основе языка логических схем Ляпунова [16], в ней собирается вся логика управления программы. В таблице выражений в польской инверсной записи хранятся выражения в смысле языка Си. Доступ в ТВ осуществляется по ссылкам из ЛС. Оператор ЛС — ссылка в ТВ на вычисляемое выражение, предикат состоит из ссылки в ТВ на условие перехода и ссылки по ЛС, определяющей переход в случае истинности выражения.

Для каждой функции (программного модуля) создаются свои ЛС и ТВ. Достаточно, чтобы во время выполнения программы в памяти находилось ВП только текущей функции, это позволит отлаживать программы любого размера.

Представление выражений в виде польской инверсной записи, состоящей из кодов операций и ссылок в память, обеспечивает высокую скорость интерпретации. Высокая скорость трансляции программы в ВП достигается однопроходностью трансляции.

Изменения, вносимые пользователем в программу, подразделяются по объему исправления ВП на три категории:

локальные изменения выражения в рамках одного оператора, при этом заново транслируется в ВП только измененный оператор, автоматически создается новая запись в ТВ и изменяется ссылка в ЛС;

изменение логики программы (добавление новых операторов, изменение операторов управления), при этом также транслируется только измененный оператор, и автоматически создается новая ЛС на основе старой;

изменения в объявлении переменных, заново транслируется в ВП та часть программы (блок, функция, мо-

дуль), где это объявление доступно.

Внутренний редактор отладчика, работающий в терминах языка Си, автоматически исправляет ВП.

Внутреннее представление отладчика компактно — для функции, состоящей из 100 операторов Си, размеры ВП (ЛС+ТВ) составляют примерно 1,5К байта (объектный код такой функции для Си в ОС РАФОС для микроЭВМ «Электроника 60» — 1,8К байт).

Одновременно с созданием ВП распределяется статическая память и формируются таблицы с дополнительной информацией, которая используется в ходе отладки (таблица переменных, таблица соответствия операторов исходной программы элементам ЛС). Один оператор языка Си может транслироваться в несколько элементов ЛС; взаимная однозначность соответствия достигается путем выделения у каждого элемента ЛС поля признака начала нового оператора в исходной программе.

Адрес для справок: 117234, Москва, Ленинские горы, МГУ, факультет ВМиК, Лаборатория вычислительных комплексов, к. 764. Телефон: 939-46-71

ЛИТЕРАТУРА

1. Зелковиц М., Марвин В. Принципы разработки программного обеспечения.— М.: Мир, 1982.— С. 11—21, 38—43.
2. Шнейдерман И. Б. Психология программирования.— М.: Радио и связь, 1984.
3. Безбородов Ю. М. Индивидуальная отладка программ.— М.: Наука, 1982.— С. 39—71.
4. Гласс Р., Нуазо Р. Сопровождение программного обеспечения.— М.: Мир, 1983.
5. Ван Тассел Д. Стиль, разработка, эффективность, отладка и испытание программ.— М.: Мир, 1985.— С. 181—182.
6. Брукс Ф. П. мл. Как проектируются и создаются программные комплексы? Мифический человеко-месяц.— М.: Наука, 1979.— С. 21—22, 109—115.
7. Proc. of symposium on high level debugging // ACM Sigplan Notices.— 1983.— V. 18.— № 3.
8. Cargil T. Implementation of the Blit Debugger // Software Pract. and Exper.— 1985.— V. 15.— № 2.— P. 153—168.
9. Adams E., Muchnick S. Dbxtool: A window — based symbodic debugger for Sun workstation // Software Pract. and Exper.— 1986.— V. 16.— № 7.— P. 653—670.
10. Pancrapakesan S., Subramain S., Venkateswaran H. An interactive assembly level debugging system // Software Pract. and Exper.—1985.— V. 15.—№ 1.— P. 59—64.
11. Зурахинский В. И., Файнберг Б. А. Трассирующий отладчик для микроЭВМ «Электроника 60» // Программирование.— 1985.— № 4.— С. 93—94.
12. Suydam W. New C programing tods emphasize debugging capabilities//Computer design.—1985.— V. 24.— № 5.— P. 48.
13. Vose G. M. C-language development tools // BYTE.— 1984.— V. 9.— № 13.— P. 119.
14. EDN.—1985.— № 14.
15. Кернеган Б., Ритчи Д., Фьюэр А. Язык программирования Си: Задачи по языку Си.— М.: Финансы и статистика, 1985.
16. Ляпунов А. А. О логических схемах программ // Проблемы кибернетики.— 1958.— № 1.

Статья поступила 6 ноября 1985 г.

УДК 681.3.06

Г. В. Борзов, М. М. Ляпунов

ПРОГРАММНЫЕ ТРЮКИ НА МАКРО-11

Трюками мы называем необычные приемы программирования. Трюк должен быть лаконичным и давать выигрыш в быстродействии или в объеме программы. Каждый трюк несет в себе элемент неожиданности, например использует команду очистки для передачи управления или наоборот.

Отношение к трюкам может быть различным. Некоторые уравновешенные люди признают, что в этом что-то есть, но избегают фокусов, дабы не усложнять жизнь. Другие видят в трюках эстетические свойства и восхищаются ими настолько, что забывают о назначении программы. Третьи находят прямую связь между трюками, трюкачеством и «бит-жонглерством» и считают все это безусловно вредными привычками плохо воспитанных программистов. Четвертые воспринимают трюк как анекдот для роботов (или для тех программистов, которые в своем развитии (деградации) достигли уровня роботов).

Вероятно, в каждом из этих мнений есть что-то от истины. Мы считаем, что к месту употребленный трюк, снабженный, когда надо, комментариями, ничего, кроме пользы, принести не может. Некоторые трюки, входя в обиход, становятся привычными и воспринимаются как нормальные приемы программирования. В большинстве случаев чистый выигрыш от них невелик, но бывают ситуации, когда время работы программы жестко ограничено, и только трюк может спасти положение. Наконец, знакомство с трюками полезно и тем, кто их не использует, так как позволяет глубже осознать особенности ЭВМ и чувствовать себя свободнее.

Из-за своей необычности трюки, как правило, реализуются только на языках ассемблерного типа или непосредственно в машинных кодах. Некоторые трюки для машин с системой команд «Электроники 60» описаны ниже. Часть из них мы обнаружили в недрах операционных систем, часть изобрели сами (вероятно, не впервые).

Вокруг С-разряда

1. Выход из подпрограммы.

Часто бывает удобно покидать подпрограмму с признаком успешного или неуспешного ее завершения в С-разряде. Обычно конец такой подпрограммы имеет вид

```
12$:    CLC
        RETURN          ; УСПЕШНОЕ ЗАВЕРШЕНИЕ
13$:    SEC
        RETURN          ; НЕУСПЕШНОЕ ЗАВЕРШЕНИЕ
```

Одну команду здесь можно сэкономить:

```
12$:    TST     (PC)+   ; УСПЕШНОЕ ЗАВЕРШЕНИЕ
13$:    SEC             ; НЕУСПЕШНОЕ ЗАВЕРШЕНИЕ
        RETURN
```

В данном случае команда SEC используется в качестве операнда команды TST, которая сбрасывает С-разряд и передает управление команде RETURN (RTS PC).

2. BLO=BCS, BHIS=BCC.

Эти пары команд эквивалентны полностью. Точнее говоря, для компьютера это не четыре команды, а только две, каждой из которых в МАКРО-11 соответствует по две мнемоники. Для программиста естественно употреблять BLO и BHIS после команд сравнения беззнаковых двоичных чисел (например, адресов), а BCC и BCS — в том случае, когда явно анализируется С-разряд. Используя эквивалентность мнемоник, иногда

достаточно поставить операнды в командах сравнения в правильной последовательности, чтобы заметно сократить размер программы. Подпрограмма из следующего примера получает на входе в R0 код символа ASCII и выясняет, является ли этот символ буквой, т. е. лежит ли его код в одном из диапазонов 101...132 (A...Z) и 140...176 (Ю...Ч). Результат возвращается в С-разряде (0 — буква, 1 — не буква):

```
LETTER: CMP     R0,#'A
        BLO     99$
        CMP     #'Z,R0
        BHIS    99$
        CMP     R0,#'Ю
        BLO     99$
        CMP     #'Ч,R0
99$:    RETURN
```

3. Сохранение в стеке и восстановление С-разряда может быть выполнено командами

ROL —(SP)

и

ROR (SP)+

4. Команды, не изменяющие С-разряд.

Тот факт, что команды MOV, BIC, INC и некоторые другие не изменяют значения С-разряда, может послужить причиной их нестандартного применения. Так, завершение подпрограммы из примера 1 может быть развито следующим образом:

```
12$:    TST     (PC)+           ; УСПЕШНЫЙ ВЫХОД (C=0)
13$:    SEC                     ; НЕУДАЧА (C=1)
        MOV     (SP)+,R0        ; ВОССТАНОВЛЕНИЕ R0
        MOV     (SP)+,R1        ;  И R1 ИЗ СТЕКА
        BIC     R2,R2           ; ОЧИСТКА R2
        INC     (SP)+           ; ПРОДВИЖЕНИЕ УКАЗАТЕЛЯ СТЕКА
        RETURN
```

Для продвижения указателя стека на два слова можно использовать команду

MOV (SP)+, (SP)+

Она не меняет С-разряда, но устанавливает признаки N и Z, поэтому ее можно применить для анализа знака числа в представлении с плавающей запятой, расположенного в стеке, и при этом удалить его из стека.

5. С-разряд и прерывания.

При выполнении процедуры прерывания С-разряд получает новое значение из младшего разряда второго слова вектора прерывания. Это обстоятельство используется в программе, которая подсчитывает число тактов в 32-разрядном слове:

```
        .ASECT
        .=100
            .WORD   TIMER,341
        .PSECT
TIME1:  .WORD   0       ; МЛАДШИЕ РАЗРЯДЫ
TIME2:  .WORD   0       ; СТАРШИЕ РАЗРЯДЫ

TIMER:  ADC     TIME1
        ADC     TIME2
        RTI
```

Это короче, чем комбинация

```
TIMER:  INC     TIME1
        BNE     99$
        INC     TIME2
99$:    RTI
```

Более сложный пример использования С-разряда при прерываниях — программа в разделе «большой трюк» (см. ниже).

Ненормальный JSR

1. Кратное выполнение подпрограмм.

Команда CALL @PC(JSR PC, @PC) передает управление непосредственно следующей за ней инструкции, записывая ее же адрес в стек, и при этом занимает только одно слово в памяти. Следующий пример показывает, как это применяется для выполнения подпрограммы 2, 4 или вообще 2 в степени N раз:

```
...     ...     ...
REPT4:  CALL    @PC
REPT2:  CALL    @PC
REPT1:  ...
        ...
        RETURN
```

Управление этой подпрограмме передается командой CALL REPT1 (однократное выполнение), CALL REPT2 (двукратное) или CALL REPT4 (четырехкратное).

2. Как стереть всю память одной командой?

Для этого достаточно записать по адресу 0 команду JSR PC, —(PC) (код 004747), занести в указатель стека адрес конца памяти и запустить компьютер с адреса 0. Команда передает управление сама себе за счет адресации с автоуменьшением PC и каждый раз записывает «адрес возврата» (т. е. число 0) в стек. Когда стек заполнит всю память, он обнулит и ячейку с нулевым адресом, где хранится команда; при очередном цикле процессор прочитает этот нуль, воспримет его как команду HALT и остановится.

Этот трюк напоминает курьезные шахматные задачи типа «поставить мат в полхода». Как ни странно, нам удавалось употреблять его с пользой — при откладке аппаратуры и в программах начальной загрузки сателлитных микроЭВМ.

Есть еще один похожий трюк, который удалось применить только для проверки степени владения знаниями у изучающих систему команд. Задается вопрос: какая команда может переписать сама себя в другую ячейку памяти и передать своей копии управление? Правильный ответ: MOV—(PC), —(PC) (014747).

3. JSR для сохранения регистров.

Пусть имеется несколько однотипных устройств, для которых желательно использовать общую программу обработки прерывания. Адрес входа в эту программу будет присутствовать в первом слове каждого вектора прерывания; для идентификации устройств удобно использовать младшие четыре разряда второго слова каждого вектора, записывая в них номер конкретного устройства. После прерывания этот номер нужно извлечь из младших разрядов слова состояния процессора. Единственное неудобство состоит в том, что для обработки номера требуется регистр, а его значение, естественно, надо предварительно сохранить; если это делать, как обычно, командой MOV RX, —(SP), то эта команда нарушит слово состояния и номер будет потерян. Обходные пути преодоления этой «мелочи» состоят в предварительной записи слова состояния в ячейку памяти или в стек. Существенно лучше (как по затратам памяти, так и по быстродействию) использовать команду JSR RX, @PC, которая не нарушает порядка выполнения команд и занимает только одно слово:

```
        .ASECT
        .=VECT1
            .WORD   INT,341
        .=VECT2
            .WORD   INT,342
        .....
INT:    JSR     R0,@PC
        MFPS    R0      ; ИЛИ MOV @#177776,R0
        BIC     #177760,R0
        ...
        MOV     (SP)+,R0
        RTI
```

4. JSR в качестве RTS.

Этот пример можно рассматривать как расширение предыдущего. Подпрограмма SAVERG сохраняет в стеке значения регистров R0..R5 и при этом не нарушает слова состояния процессора. Обращение к подпрограмме производится командой JSR R5, SAVERG:

```
SAVERG: JSR     R4,@PC
        JSR     R3,@PC
        JSR     R2,@PC
        JSR     R1,@PC
        JSR     R0,@R5
```

Последняя команда заодно возвращает управление вызывающей программе. Значения регистров сохраняются в стеке, но не сохраняются с самих регистрах; при обработке прерываний это, как правило, и не нужно.

Варианты параллельного ветвления

Пусть в регистре R0 содержится число в диапазоне 0...N и требуется в соответствии с ним передать управление одной из меток L0...LN. Тривиальное решение состоит в (N+1)-кратном повторении пары команд CMP R0, #1 и BEQ L1. При малых N такая программа выглядит нормально; когда N больше пяти, она становится громоздкой и неэффективной. Немногим лучше вариант с повторением пары DEC R0 и BEQ L1 (или BMI L1), а также вариант циклического просмотра таблицы. Предпочтительнее параллельная организация ветвления, когда число команд, выполняемых для анализа R0 и передачи управления, не зависит от значения N. Например:

1. Параллельное ветвление с неявной таблицей переходов:

```
        ASL     R0
        ADD     R0,PC
        BR      L0
        BR      L1
        ...
        BR      LN
```

2. Параллельное ветвление с явной таблицей переходов:

```
        ASL     R0
        JMP     @TABLE(R0)      ; ИЛИ CALL @TABLE(R0)
TABLE:  .WORD   L0,L1,...LN
```

Первый вариант хорош тем, что содержит только позиционно-независимые коды.

3. Рассеянная таблица.

Бывает так, что число N велико, но фактически в диапазоне 0...N только некоторые значения важны; если число в R0 не совпало ни с одним из них, то требуется передать управление на метку NOACT (ничего не делать). Решение, приведенное в предыдущем примере, здесь вполне пригодно, но может потребовать большой таблицы, безыдейно заполненной главным образом словами .WORD NOACT. Положение несколько улучшается, если использовать не одну таблицу, а две, причем во второй (короткой) хранить адреса переходов, в первой — относительные адреса второй. Экономия памяти достигается благодаря тому, что значения относительных адресов малы, и первая таблица составлена не из слов, а из байтов:

```
        MOVB    TABLE1(R0),R0
        JMP     @TABLE2(R0)     ; ИЛИ CALL @TABLE2(R0)
        ...
TABLE1: .BYTE 0,0,...A1-TABLE2,0,...
        .BYTE A2-TABLE2,0...A3-TABLE2,0...
        ...
TABLE2: .WORD   NOACT
A1:     .WORD   L1
A2:     .WORD   L2
        ...
AK:     .WORD   LK
```

В приведенном варианте программы число меток L1...LK не может превышать 63 (десятичных) из-за расширения знака байта во время выполнения команды MOVB. Располагая метку TABLE2 посередине таблицы, можно это число увеличить до 127. Это может основательно испортить внешний вид программы; предпочтительнее следующая запись:

```
        MOVB    TABLE1(R0),R0
        JMP     @TABLE2+200(R0)
;       ...
TABLE1:
.IRP    LAB,<X,X,...A1,X,...A2...>
        .BYTE   LAB-TABLE2-200
.ENDM

TABLE2:
X:      .WORD   NOACT
A1:     .WORD   L1
A2:     .WORD   L2
        ...
AK:     .WORD   LK
```

Благодаря байтовой организации TABLE1 команда ASL R0 становится ненужной, поэтому потери быстродействия по сравнению с предыдущим вариантом невелики.

Разное

1. Кольцевой буфер

Пусть имеется устройство ввода, поставляющее процессору байты, и пусть требуется обрабатывать прерывание от этого устройства, просто записывая эти байты в кольцевой буфер. Это можно сделать, например, следующим образом:

```
.ASECT
.=VECTOR
        .WORD   INT,340

.PSECT
LENGTH=100
RING:   .BLKB   LENGTH
POINTR: .WORD   RING
...
INT:    MOVB    @#DATARG,@POINTR
        INC     POINTR
        CMP     POINTR,#RING+LENGTH
        BNE     1$
        MOV     #RING,POINTR
1$:     ...
        ...
        RTI
```

Как видно отсюда, «закольцовывание» требует четырех довольно длинных команд на модификацию указателя. При обслуживании нескольких быстродействующих устройств потери времени могут стать слишком большими. Попытка модифицировать указатель, загружая его значение в регистр, не дает эффекта, так как требует дополнительных затрат на сохранение и восстановление. Предлагаемое ниже решение может показаться «нечестным» и неожиданным:

```
.ASECT
.=VECTOR
        .WORD   340,INT

LENGTH=400
RING=1000

.PSECT
POINTR: .WORD   RING

INT:    MOVB    @#DATARG,@POINTR
        INCB    POINTR  ; МОДИФИЦИРУЕТСЯ ТОЛЬКО
                        ; МЛАДШИЙ БАЙТ УКАЗАТЕЛЯ
        ...
        RTI
```

Применимость этого приема ограничивается тем, что кольцевой буфер должен иметь длину в точности 256 (десятичных) байтов и должен быть расположен на границе 256-байтового блока.

2. Двойной вход в подпрограмму

Часто в практике программирования возникают две почти одинаковые подпрограммы, отличающиеся какой-либо мелочью, расположенной, к сожалению, где-нибудь в середине алгоритма. В таких случаях естественнее писать не две подпрограммы, а только одну, но с двумя точками входа. Факт передачи управления на ту или иную точку входа отмечается флагом, который в дальнейшем анализируется для ветвления. Следующий способ реализует это довольно экономным образом:

```
ENTER1: MOV     (PC)+,R5
ENTER2: CLR     R5
        ...
        TST     R5
        BEQ     .+4

...     ...
        RETURN
```

Здесь флаг располагается в регистре R5. При входе на ENTER2 флаг обнуляется; при входе на ENTER1 команда CLR R5 используется как операнд команды MOV, т. е. в R5 попадает код команды CLR R5, который пулю не равен.

«Большой трюк»: реентерабельный сопрограммный коммутатор прерываний ввода-вывода.

Этот пример является комбинацией нескольких предыдущих. Пусть имеется несколько однотипных устройств ввода-вывода (например, терминалов), подключенных к одному процессору. Требуется организовать работу с ними по прерываниям. Алгоритм управления предполагается достаточно сложным и не независимым, т. е. способ обработки прерывания ввода зависит от того, какие действия выполнены к данному моменту с каналом вывода, и наоборот. В таких условиях удобно реализовать алгоритм как сопрограмму — забыть на время о том, что существует фоновый процесс и прерывания как таковые, и писать программу так, как если бы весь процессор был предназначен исключительно для управления вводом-выводом. В тот момент, когда по алгоритму требуется ожидание готовности ввода или вывода, программа должна обращаться к специальной процедуре ожидания, которую можно представить себе следующим образом:

```
WAITIO: TSTB    @#177560    ; ГОТОВНОСТЬ ВВОДА ЕСТЬ?
        BMI     1#          ; ДА - НА ВОЗВРАТ С C=0
        TSTB    @#177564    ; ГОТОВНОСТЬ ВЫВОДА ЕСТЬ?
        BPL     WAITIO      ; НЕТ - ПРОДОЛЖИТЬ ОЖИДАНИЕ
        SEC                 ; C=1 - ПРИЗНАК ГОТОВНОСТИ ВЫВОДА
1#:     RETURN
```

(для определенности здесь использованы адреса регистров системного терминала 177560...177566). Вызов процедуры прерывания осуществляется командой CALL WAITIO; по окончанию ожидания управление возвращается на следующую за CALL WAITIO команду, т. е. процедура ожидания является обычной подпрограммой по отношению к программе управления.

Следующий шаг должен состоять в замене процедуры ожидания другой специальной программой — сопрограммным коммутатором, который вместо малополезного повторения TSTB будет передавать управление фоновому процессу (командой RTI) и вновь получать управление по прерыванию. Сопрограммный коммутатор желательно организовать так, чтобы обслуживание всех однотипных устройств производилось параллельно и независимо по одной и той же программе.

Следующую реализацию сопрограммного коммутатора мы сочли возможным отнести к трюкам, так как она лаконична (по сравнению с перечнем функций) и основана на нестандартном применении команд:

```
;-----------------------------
; ОПИСАНИЕ ВЕКТОРОВ ПРЕРЫВАНИЯ
;-----------------------------

.ASECT
.=VECT1
        .WORD   INT1,340,INT1,341
.=VECT2
        .WORD   INT2,340,INT2,341
.................
.=VECTN
        .WORD   INTN,340,INTN,341
```

```
;-------------------------
; СОПРОГРАММНЫЕ КОММУТАТОРЫ
; И ОБЛАСТИ ПЕРЕМЕННЫХ
; (ПО ЧИСЛУ УСТРОЙСТВ)
;-------------------------

.PSECT
INT1:   JSR     R5,@(PC)+       ; СОПРОГРАММНЫЙ
        .WORD   START           ; КОММУТАТОР
BASE:   MOV     (SP)+,-(R5)     ; ДЛЯ ПЕРВОГО
        MOV     (SP)+,R5        ; УСТРОЙСТВА
        RTI                     ;
; ЗДЕСЬ ДОЛЖНЫ РАСПОЛАГАТЬСЯ ОПИСАНИЯ КОНСТАНТ И
; ПЕРЕМЕННЫХ ДЛЯ ПЕРВОГО УСТРОЙСТВА, НАПРИМЕР:
ADDRIN: .WORD   177562
ADDROU: .WORD   177566
POINTR: .WORD   BUFFER
BUFFER: .BLKB   100
;       И Т.Д.

;-----------------------------------------------
INT2:   JSR     R5,@(PC)+       ; СОПРОГРАММНЫЙ
        .WORD   START           ; КОММУТАТОР
        MOV     (SP)+,-(R5)     ; ДЛЯ ВТОРОГО
        MOV     (SP)+,R5        ; УСТРОЙСТВА
        RTI                     ;
; КОНСТАНТЫ И ПЕРЕМЕННЫЕ ДЛЯ ВТОРОГО УСТРОЙСТВА,
; РАСПОЛОЖЕННЫЕ В ТОМ ЖЕ ПОРЯДКЕ, ЧТО И ДЛЯ ПЕРВОГО:
        .WORD   177272
        .WORD   177276
        .WORD   .+2
        .BLKB   100
;       И Т.Д.

;-----------------------------------------------

...     ...     ...


INTN:   JSR     R5,@(PC)+       ; СОПРОГРАММНЫЙ
        .WORD   START           ; КОММУТАТОР
                                ; ДЛЯ N-ГО
        ...                     ; УСТРОЙСТВА
        ...                     ;
        ...                     ;

;-----------------------------------
; ПРОГРАММА УПРАВЛЕНИЯ ВВОДОМ/ВЫВОДОМ
;-----------------------------------

START:  BCS     1#      ; НА ОБРАБОТКУ ПРЕРЫВАНИЯ ВЫВОДА

; ОБРАБОТКА ПРЕРЫВАНИЯ ВВОДА:
; ОБРАЩЕНИЕ К ОБЛАСТИ ПЕРЕМЕННЫХ ДОЛЖНО
; ПРОИЗВОДИТЬСЯ ИНДЕКСНО ПО R5, НАПРИМЕР:

        MOVB    @ADDRIN-BASE(R5),@POINTR-BASE(R5)
        INCB    POINTR-BASE(R5)

        ...
        ...

        CALL    @R5     ; ОБРАЩЕНИЕ К КОММУТАТОРУ
                        ; ДЛЯ ОЖИДАНИЯ ГОТОВНОСТИ
        BCS     2#      ; НА ОБРАБОТКУ ПРЕРЫВАНИЯ ВЫВОДА

        ...
        ...

1#:     ...
        ...
        CALL    @R5     ; ЕЩЕ ОДНО ОБРАЩЕНИЕ К КОММУТАТОРУ
        BCS     3#      ; НА ОБРАБОТКУ ВЫВОДА

        ...
        ...
        BR      START   ; ЭТА КОМАНДА ЗДЕСЬ ПОСТАВЛЕНА
                        ; ДЛЯ ИЛЛЮСТРАЦИИ ТОГО, ЧТО
                        ; КОНЦА АЛГОРИТМ НЕ ИМЕЕТ

;-----------------------------------------------
```

Описанный коммутатор не полностью эквивалентен процедуре ожидания, основанной на командах TSTB: нельзя использовать регистры и необходимо возвращать стек в исходное состояние перед каждым обращением к коммутатору. При необходимости можно достигнуть полной эквивалентности (ценой некоторой потери быстродействия). Для этого достаточно ввести собственные стековые буферы в области переменных каждого устройства и команды сохранения / восстановления регистров (включая указатель стека). Правда, после таких изменений программа заметно теряет лаконичность, и ее уже трудно назвать трюком.

Телефон для справок: 196-94-05, Москва.

Статья поступила 9 апреля 1987 г.

УДК 681.3.06

В. М. Арпаксыд, И. Б. Володарский, А. Е. Дорфман

МОБИЛЬНАЯ ОПЕРАЦИОННАЯ СИСТЕМА ДЛЯ ПЭВМ «ИСКРА 226»

Операционная система УНИКС-226 (УНИКС — унифицированный инструментальный комплекс), разработанная для ПЭВМ «Искра 226», совместима с операционными системами ИНМОС, ДЕМОС и UNIX версии 7. УНИКС-226 — однопользовательская однопрограммная система с фоновым вводом-выводом, обеспечивающая независимость программ от внешних устройств и способов доступа к информации.

Рекомендуемая область применения — автоматизация административно-управленческой деятельности, инженерные и планово-экономические расчеты, обработка текстов, автоматизация проектирования, информационно-справочные и обучающие системы.

Средства ОС УНИКС-226 могут быть сгруппированы следующим образом:

системные вызовы и стандартные библиотечные функции, обеспечивающие интерфейс прикладных программ с операционной системой, организацию ввода-вывода, управление программами и памятью, вычисление математических функций;

файловая система и драйверы внешних устройств (терминал, гибкие и кассетные магнитные диски, магнитная лента, печатающее устройство, телекоммуникационный и приборный интерфейс);

командный язык, служащий для управления работой системы и позволяющий строить командные файлы и легко расширять существующий набор команд;

базовый набор команд — стандартных программ-утилит;

язык программирования Си и инструментальные средства разработки программ (редактор текстов, компоновщик модулей, библиотекарь);

базовые средства форматирования текстов;

пакеты прикладных программ.

Программный и пользовательский интерфейсы УНИКС-226 соответствуют аналогичным операционным системам. Отличия обусловлены особенностями «Искры 226». Самым существенным является отсутствие средств порождения параллельных процессов из прикладных программ (из-за нехватки аппаратных ресурсов); в то же время имеются встроенные команды, работающие параллельно с основным процессом (фоновые процессы), и программные каналы, реализованные на уровне оболочки.

Для ускорения обмена с внешними устройствами используются программные механизмы буферной кэш-памяти, предварительного чтения и отложенной записи. Имеются специальные средства буферизации управляющих данных для навигации в файловой системе.

Внутренняя организация файловой системы УНИКС-226 значительно отличается от традиционных решений. Были предприняты меры по повышению ее надежности и устойчивости к сбоям аппаратуры. Блоки управляющей информации рассредотачиваются по диску; связи между ними, отражающие логические связи в файловой иерархии, поддерживаются цепочкой указателей; наиболее важная информация дублируется. Сбой сектора диска может привести к потере максимум одного файла. Структура файловой системы восстанавливается даже при утрате корневых каталогов.

На уровне прикладных программ УНИКС-226 обеспечивает высокую степень совместимости «Искры 226» с любыми ЭВМ (и любыми операционными системами), для которых реализован Си-компилятор со стандартной библиотекой поддержки. В силу этого внедрение УНИКС-226 позволит использовать готовые программы из других систем и сохранить разработанное программное обеспечение при переходе к ЭВМ нового поколения.

УНИКС-226 эксплуатируется с января 1987 г.

Адрес для справок: 252011, г. Киев-11, Печерский спуск, 19, Киевское отделение Украинского государственного проектного и проектно-конструкторского института «Тяжпромавтоматика». Тел. 290-82-56.

Сообщение поступило 4 апреля 1987 г.

УДК 681.3.06

А. Б. Борковский

ТЕКСТОВАЯ БАЗА ДАННЫХ

«Интеллектуализация» ЭВМ должна обеспечивать работу машины с неподготовленным пользователем. Можно выделить два аспекта «интеллектуализации»:

автоматическое выполнение сложных действий, не требующее от пользователя написания программ или подробного задания алгоритмов. Для этого используются средства и методы искусственного интеллекта — базы знаний, экспертные системы и пр.;

удобная работа с большими объемами информации без знания языков манипулирования данными. Развитые средства непосредственного взаимодействия, предоставляя множество мелких удобств, позволяют пользователю вручную достигать цели.

Текстовая база NOTES данных относится к средствам второго вида.

Что такое текстовая база данных

Текстовые базы данных оперируют плохо структурированной информацией, элементы которой неудобно представлять в табличном виде, как это принято в обычных базах данных. Последние ориентированы в первую очередь на выполнение массовых операций — сортировку, поиск, подготовку отчетов и обработку прикладными программами; к ним обращаются подготовленные пользователи; за структурой и содержимым базы данных следит администратор базы данных.

Текстовые базы данных предназначены для интерактивной работы с разнородной информацией и ориентированы на пользователя с минимальной подготовкой. Поэтому взаимодействие с ней опирается на удобство работы «вручную» с данными, а не на составление сложных запросов. Все операции текстовой базы выполняются с помощью управляющих клавиш или меню, и пользователю не надо изучать специальный язык.

Текстовая база данных представляет собой совокупность КАРТОЧЕК, содержащих текстовую информацию. Тексты карточек имеют произвольный формат, они не обязаны быть похожими друг на друга. В каждой карточке выделяется одно или несколько КЛЮЧЕВЫХ СЛОВ, обеспечивающих поиск и доступ к карточкам. Одно и то же ключевое слово может входить в несколько карточек. Отсюда другое название для текстовых баз данных — картотека.

В разработанной для ПЭВМ типа IBM PC картотеке NOTES можно выделить следующие характерные операции:

просмотр списка ключевых слов;

поиск ключевых слов по заданному шаблону;

раскрытие ключевых слов — показ первых строчек карточек;

просмотр и редактирование текста выбранной карточки;

занесение отредактированной карточки в картотеку;

«откладывание» просмотренной карточки;

установка шаблона, определяющего подмножество ключевых слов для работы;
создание новой карточки и занесение ее в картотеку;
вывод одной, нескольких или всех карточек в текстовый файл или на печать;
ввод в картотеку новых карточек из текстового файла;
удаление карточек и/или ключевых слов;
выполнение операций операционной системы.

Выбранные карточки просматриваются и редактируются в отдельных окнах (можно иметь до четырех рабочих окон одновременно). Встроенный в картотеку редактор текстов обеспечивает все традиционные операции экранного редактора: просмотр; вставку и удаление символов, строк и выделенных фрагментов; контекстный поиск; работу с двумя (русским и латинским) алфавитами и пр. В тексте карточек ключевые слова выделяются цветом.

Картотека NOTES реализована на основе системы управления окнами *. Каждый упорядоченный набор объектов (список ключевых слов, строки текста карточки, меню) отображается в отдельном окне. Работа с любым окном производится единообразно: клавиши управления курсором обеспечивают прокрутку и выбор текущего объекта, нажатие функциональной клавиши вызывает соответствующее действие над текущим (указываемым курсором) объектом. При этом аналогичные действия (удаление, раскрытие, запись в файл, подсказка) вызываются одинаковыми клавишами во всех окнах. Система управления окнами позволяет также изменять размер и расположение окон на экране; установленные параметры окон запоминаются и используются при последующих вызовах картотеки.

Работа с картотекой NOTES

Работа с картотекой NOTES основана на непосредственном взаимодействии: для выполнения операции пользователь выбирает необходимый объект, указывая на него курсором, и нажимает функциональную клавишу. Строка в нижней части экрана содержит краткую подсказку о возможных в данный момент действиях; более полная инструкция вызывается клавишей «подсказка».

Работа со списком ключевых слов. После вызова картотеки на экране изображено окно с алфавитно-упорядоченным списком ключевых слов. Вы можете просматривать этот список, перемещая по нему курсор, или сразу найти необходимое ключевое слово, нажав клавишу «поиск» и задав образец поиска; поиск производится по первым буквам. Команда «поиск» используется также для выбора группы карточек.

Выбрав в окне ключевых слов интересующее Вас слово, Вы можете: раскрыть его, посмотреть связанную с ним карточку, напечатать или записать в файл карточки с этим ключевым словом.

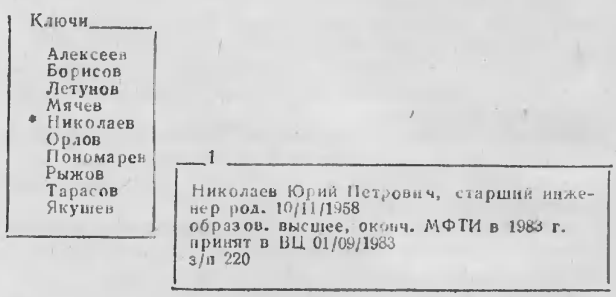
Для раскрытия надо нажать клавишу «раскрыть»; при этом изображение слова заменяется списком заголовков (первых строчек) карточек, в которых оно встречается. Повторное нажатие клавиши «раскрыть» возвращает изображение только ключевого слова.

Указав на ключевое слово или на заголовок карточки в окне ключевых слов, можно вызвать всю карточку в рабочее окно клавишей «взять карточку». Имеется несколько рабочих окон; цифра нажатия перед клавишей «взять карточку» указывает номер окна, в которое помещается карточка. Одновременное нажатие клавиши «Alt» и любой цифры позволяет перемещаться между окнами; при переходе в другое окно содержимое текущего окна не изменяется.

Перейти в рабочее окно можно также нажав клавишу «создать». При этом открывается рабочее окно для ввода новой карточки.

Операции в рабочем окне. В рабочем окне с текстом карточки можно работать, как в обычном экранном редакторе: перемещаться по тексту и прокручивать его, вставлять, удалять, восстанавливать и копировать литеры и строки. Текст, удаленный из карточки в одном рабочем окне, можно вставить в карточку другого окна. При редактировании строки выделение ключевых слов цветом заменяется управляющим символом перед словом, который может быть удален, как обычная литера, и вставлен клавишей «отметить ключ». Отредактированная карточка может быть записана клавишей «записать». При этом список ключевых слов не изменяется. Чтобы занести в картотеку ссылки на измененную или созданную карточку, необходимо вызвать клавишей «изменить» меню модификации.

Клавиша «выход» возвращает Вас в окно ключевых слов; при этом изображение на экране имеет следующий вид (слева — окно ключевых слов, справа — рабочее окно):



Добавление и удаление карточек. Карточки можно удалять двумя способами: клавишей «удалить» в окне ключевых слов; с помощью меню модификации, вызываемого из рабочего окна.

Добавляются карточки в картотеку одним из трех способов: вводом карточки в пустое рабочее окно клавишей «создать»; модификацией имеющейся карточки; вводом карточек из текстового файла.

Клавиша «удалить» уничтожает связь ключевого слова и карточки. Если курсор указывает на заголовок карточки при «раскрытии» ключевого слова, уничтожается ссылка на эту карточку; если же курсор указывает на ключевое слово, уничтожаются все ссылки этого ключевого слова. Карточки, на которые нет ссылок, удаляются из картотеки.

Меню модификации содержит три пункта:

занести новую — карточка в текущем рабочем окне заносится в картотеку;
занести старую — в списке ключевых слов убираются все ссылки на карточку в текущем окне, и карточка заново заносится в картотеку;
удалить — карточка в текущем окне удаляется из картотеки.

Операция «занести старую» фактически эквивалентна последовательному выполнению операций «удалить» и «занести новую».

Главное меню. Нажатие клавиши «выход» в окне ключевых слов вызывает главное меню, содержащее три альтернативы:

ключи — переход в окно ключевых слов;
шаблон — задание шаблона для ключевых слов, показываемых в окне ключевых слов (например, шаблон «пет? фамилия, н?» выделяет карточки, содержащие либо ключевое слово, начинающееся на букву «н»,

* Борковский А. Б. Многооконное текстовое взаимодействие с персональной ЭВМ // Микропроцессорные средства и системы. — 1984. — № 4. — С. 47.

либо ключевое слово «фамилия» и ключевое слово, начинающееся на «пет»);

ввод-вывод — вызывает меню ввода-вывода.

Меню ввода-вывода содержит следующие альтернативы:

сохранить все — содержимое картотеки выводится в текстовый файл. При этом каждая карточка оформляется как абзац — последовательность строк, первая из которых начинается пробелом, а остальные — отличной от пробела литерой;

сохранить часть — карточки, удовлетворяющие заданному шаблону выводятся в текстовый файл;

имя файла — задать имя файла, в который выводятся карточки клавишей «записать»;

загрузить — ввести карточки из текстового файла. Каждый абзац заносится как отдельная карточка; ключевые слова должны быть отмечены кодом ctrl_K;

создать — текущая карточка очищается, и в нее загружается указанный текстовый файл.

Реализация и использование

Картотека NOTES написана на языке TURBO-Pascal. Настоящая версия является прототипом соответствующей компоненты интегрированной системы СПЕКТР, разрабатываемой в Лаборатории программного обеспечения ПК ВЦ АН СССР (исходные тексты могут быть получены у автора).

Картотека NOTES находится в стадии опытной эксплуатации. Ее используют в основном в качестве записной книжки для хранения разнообразной текущей информации, библиографических ссылок и т. д. Автор использовал картотеку NOTES для подготовки к изданию англо-русского словаря по информатике и программированию с толкованиями. Объем словаря около 6000 карточек общим объемом более 800К байт.

Телефон для справок: 135-13-40, г. Москва.

Статья поступила 14 мая 1986 г.

Приложение.

СООТВЕТСТВИЕ КОМАНД NOTES И КЛАВИШ ДЛЯ IBM PC

Команды списка ключевых слов:

⟨подсказка⟩	— F1, shift_F1
⟨выход⟩	— F10
⟨поиск⟩	— F2
⟨удалить⟩	— ctrl_F3
⟨создать⟩	— F7
⟨записать⟩	— shift_F8
⟨раскрыть⟩	— Grey Plus
⟨взять карточку⟩	— Enter
⟨изменить окно⟩	— alt_W
⟨переключение окон⟩	— alt+цифра

Команды рабочего окна (кроме команд редактирования):

⟨записать⟩	— shift_F10
⟨изменить⟩	— ctrl_F10
⟨отложить⟩	— F8
⟨след. карточка⟩	— Ctrl_PgDn
⟨пред. карточка⟩	— Ctrl_PgUp
⟨отметить ключ⟩	— ctrl_K
⟨поиск текущего слова⟩	— shift_F2
⟨изменить окно⟩	— alt_W

МАГИСТРАЛИ МП-СИСТЕМ

УДК 681.327

И. Ф. Колпаков

ШИНА VME И ЕЕ ПРИМЕНЕНИЯ

Введение

Шина VME является единственным в настоящее время международным стандартом для 32-разрядных микропроцессорных систем [1]. В мире производится 1051 изделие (на основе шины VME) 147 фирмами; прогнозируемый пик использования шины VME приходится на 1986—1990 г.г., а время применения — примерно до 2000 г. [2]. В 16-разрядном варианте шина VME представляется особенно привлекательной, так как она неизбыточна (т. е. более экономична и надежна).

Характеристики

Шина VME используется как самостоятельно, так и в сочетании с шинами VMX и VMS. Назначение каждой из них демонстрируется на структурной схеме (рис. 1) [3]. VME — 32-разрядный асинхронный параллельный интерфейс с 7 уровнями прерывания. Он позволяет сосредоточить в одном крейте память до 4 Г байт. Захват шины модулем, который должен в данный момент ею управлять, осуществляется специальным модулем-арбитром шины. Максимальная скорость обмена по шине VME — 24...57 М байт/с. Для шины VMX она доходит до 70 М байт/с, т. е. близка к скорости обмена по шине Fastbus [4]. Конструктивно шина размещена на крейте с вставными модулями-платами (рис. 2). В крейте VME используется евромеханика с разъемом С96 (стандарт DIN). Высота крейта 16-разрядной шины (разъем J_2) может быть 100 мм, в 32-разрядном варианте — 233 мм. Модули в глубину имеют размер 160 мм. Преимущество VME — наличие полного набора интерфейсных кристаллов [5, 6], а также ориентация на системы реального времени.

Интерфейсные кристаллы выполняются либо как монолитные БИС, либо как программируемые диодные матрицы. Последние представляют наиболее быстрый путь преодоления технологического барьера. В набор необходимых интерфейсных кристаллов входят генератор и обработчик прерываний, арбитр приоритетов, ведущий и ведомый контроллер.

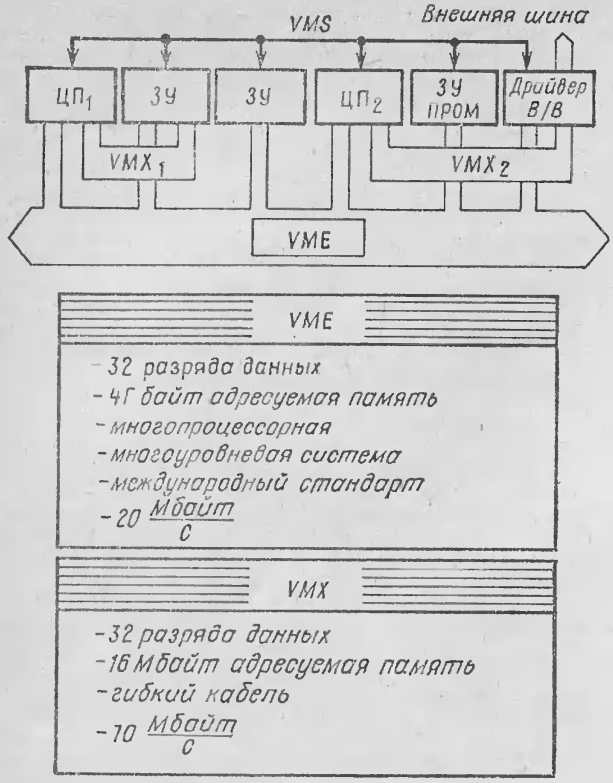


Рис. 1. Структурная схема шин VME, VMX и VMS

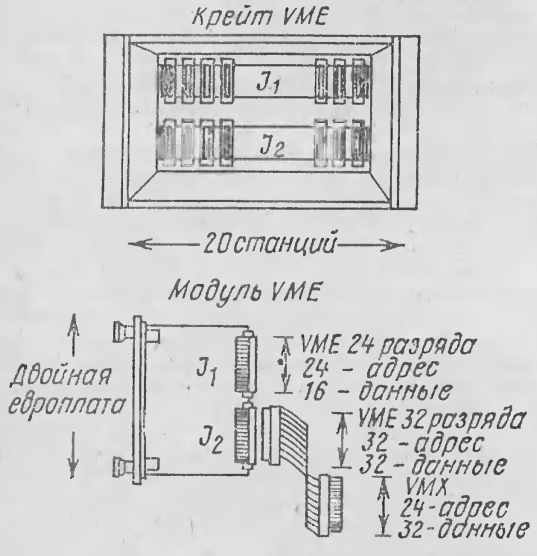


Рис. 2. Внешний вид крейта и модуля VME

Области применения

Шина VME рассчитана на широкую область применения от систем реального времени до параллельных мультипроцессорных систем обработки данных (суперкомпьютеры). Разработаны и выпускаются разнообразные наборы модулей VME и целые системы. В число наиболее распространенных типов модулей входят процессорные платы с различными микропроцессорами (WE 32100, Intel 80386, MC 68020), модули памяти, модули каналов аналоговых и цифровых входов-выходов, модули интерфейсов различных периферийных устройств, интерфейсы локальных сетей и сетей ЭВМ, интерфейсы шин Q-bus, Unibus и канала IBM, контроллеры графических дисплеев, интерфейс приборной шины МЭК.625, интерфейсы персональных компьютеров. Конкретные существующие области применения модулей VME [7—9]: автоматизированные системы управления технологическими процессорами (АСУ ТП), системы автоматизированного проектирования (САПР), станки с числовым программным управлением (ЧПУ), робототехника, гибкие производственные системы (ГПС), машинная графика, локальные сети, периферийные устройства ЭВМ, спецпроцессоры, приборостроение, автоматизация научных исследований (АСНИ).

Автоматизация научных исследований

Шина VME является естественным дополнением к широко применяемой шине КАМАК. Она позволяет ввести мультипроцессорные средства в системы автоматизации научных исследований. В этой области диапазон применения шины VME распространяется на системные контроллеры больших автоматизированных систем, графические станции экспериментальных стендов и установок, спецпроцессоры обработки данных, эмуляторы существующих высокопроизводительных ЭВМ, локальные сети лабораторий, спектрометры и системы управления ускорителями и другими базовыми установками науки.

Системные контроллеры на основе шины VME позволяют заменить на стандартной основе системный крейт для систем со многими ветвями и объединять множество компьютеров в систему. В графических станциях с помощью модулей VME можно создать новые поколения набора дисплейных процессоров и интерфейсов. Широкие возможности применения шины VME открывает для создания мультипроцессорных систем обработки научных данных, что, в конечном итоге, существенно повысит производительность компьютеров.

Спектрометры и другие системы физики элементарных частиц и атомного ядра

Структурная схема аппаратуры сбора и обработки данных современного спектрометра физики элементарных частиц в обобщенном виде показана на рис. 3 [10—12].

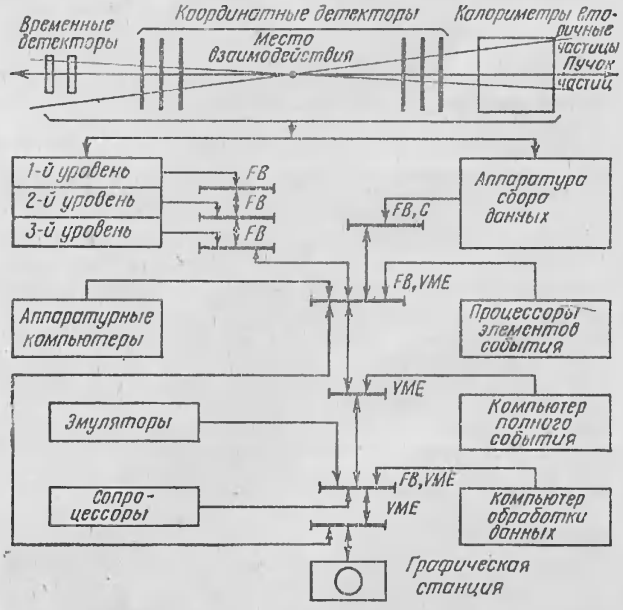


Рис. 3. Шина VME в структурной схеме современных спектрометров физики элементарных частиц

Поток информации, отобранной даже с помощью триггера от детекторов, может достигать единиц и десятков Мбайт/с. Для регистрации и обработки таких потоков информации недостаточно использования самых высокопроизводительных компьютеров. Поэтому поток информации от детекторов или их частей распараллеливается аппаратурой сбора данных по элементам образа события. Аппаратура сбора данных при числе каналов до 10^4 выполняется в виде модулей КАМАК, при бо́льшем числе каналов ($10^5 \ldots 10^6$), либо при большо́й множественности событий (>10) — в стандарте Fastbus. Считывание и предварительная обработка элементов образа события осуществляется микропроцессорными контроллерами, которые обычно выполняются как модули Fastbus или VME. Здесь же производится буферизация событий в модулях памяти. Число процессоров элементов события в установке может составлять десятки единиц. Сложение полного события производится через системный крейт VME обычно в компьютере типа VAX-11/780 или VAX 8600. Окончательно данные обрабатываются, как правило, компьютерами типа IBM 3081.

Для обработки событий в реальном времени производительности этих компьютеров недостаточно, поэтому процесс обработки также распараллеливается. Здесь наблюдается два подхода. Либо выделяется ядро из наиболее часто встречающихся команд и тогда применяются эмуляторы, либо выделяются алгоритмы, занимающие основное время компьютера обработки данных, и они выполняются на сопроцессорах — спецпроцессорах обработки таких алгоритмов. Число эмуляторов может достигать десятков, число спецпроцессоров — сотен единиц. Контроль информации, поступающей со спектрометра, и проверка установки в целом осуществляется так называемым аппаратурным компьютером, который также контролирует обработку части событий.

Современные спектрометры связаны с локальной сетью обработки данных с использованием больших компьютеров. Аппаратура локальных сетей (процессоры связи, интерфейсы) выполняется на модулях VME. В системах управления ускорителями необходимы также системные контроллеры, аппаратура локальных сетей, средства визуального представления данных и модули VME для приема и преобразования информации с датчиков.

Применения шины VME для исследования элементарных частиц и атомного ядра распространяются на системные крейты [12—16], микросуперкомпьютеры (спецпроцессоры и микропроцессорные кластеры) [17—19], графические станции спектрометров [20], собиратели образа события [12, 18, 21, 22], аппаратурные компьютеры [23], спецпроцессоры отбора данных, спецпроцессоры считывания [6, 24], спецпроцессоры обработки данных, процессоры предварительной обработки в ядерной спектроскопии [25], многоканальный многопараметрический амплитудный анализ [26, 27], многопроцессорные распределенные системы управления ускорителями [28, 29], генераторы функций управления питанием ускорителей [30], интерфейс персональных компьютеров [23, 31], интерфейс канала IBM и контроллеры периферийных устройств [32, 33].

Ведущие физические лаборатории мира уделяют большое внимание применению шины VME. Так, например, в ЦЕРН реализуется проект PRIAM [34] — создание систем автоматизации спектрометров и ускорителей элементарных частиц на основе шины и модулей VME. В Национальной ускорительной лаборатории им. Э. Ферми осуществляется проект Advanced Computer Program [17, 35], имеющий целью создание высокоэффективных систем обработки данных, превышающих по производительности существующие суперкомпьютеры в $10^2 \ldots 10^6$ раз. На основе крейта VME создана система, эквивалентная по производительности 15 ЭВМ типа VAX-11/780. Об актуальности применения шины VME в научных исследованиях свидетельствует международная конференция «VME bus in Physics», проведенная в ЦЕРНе 7—8 октября 1985 г.

В институтах социалистических стран также ведутся разработки на основе шины VME. Наибольший опыт в этом отношении приобрели: в Болгарии — Центральная лаборатория автоматизации и научного приборостроения (ЦЛАНП), где разработаны рабочие места научного работника и проектировщика на основе компьютера в крейте VME [36], и в Венгрии — Институт вычислительной техники и автоматизации ВАН (SZTAKI), где создан набор модулей VME для компьютерной графики и локальных сетей [9]. Шина VME применяется также в Институте ядерных проблем в Польше (Сьерк) [37] и СССР [38].

Заключение

Выбор стандартного интерфейса, или шины, во многом определяет успех программы автоматизации. Так, удачный выбор шины КАМАК в начале 70-х годов позволил осуществить успешную программу автоматизации в последующие годы.

Продолжающийся прогресс в области сверхбольших интегральных схем и, в особенности, схем памяти, качественно меняет ситуацию примерно каждые 3 года, поэтому едва ли можно рекомендовать стандарт шины на большой период времени, в особенности учитывая потенциальные перспективы шины P.896 (Future bus) [39]. Однако на ближайшие годы шина и модули VME представляются единственным целесообразным стандартом для широкого применения.

Телефон для справок: 926-22-22, г. Москва.

ЛИТЕРАТУРА

1. VME bus. (IEC 821 BUS), Specification Manual, Rev.C. VME bus Int. Trade Assoc., Scotsdale, Ariz., 1985.
2. Rosenberg R. Battle of the buses: and the winner is VME bus, as Multibus-II gets off to a slow start // Electronics.— 1985.— November 5,— P. 48—51.
3. Schellekens A. VMEbus Standards. In: Proc. of VMEbus in Physics Conf., CERN 86—01, Geneva, 1986.— P. 16—31.
4. Larsen R. S. Status of the FASTBUS Standard Data Bus // IEEE Trans.— 1981.— NS-28, No. 2.— P. 322—329.
5. Nissen N. VMEbus Interface Chip Set. In: Proc. of VMEbus in Physics Conf., CERN 86—01, Geneva, 1986.— P. 42—55.
6. Gustafson L., Gallino P. VME Protocol Chipset using programmable logic devices. In: Proc. of VMEbus in Physics Conf., CERN 86—01, Geneva, 1986.— P. 56—64.
7. Mitch Beedie. Focus on VMEbus peripheral boards // Electronic Design, 1985.— June 27.— P. 157—170.
8. Жаботинский Ю. Д., Сердцев А. А. Системы технического зрения для промышленных роботов // Зарубежная радиоэлектроника.— 1985.— № 12, С. 23—33.
9. LAN VME Modules. Sztaki, Budapest, 1985.
10. Sendall D. M. Real-Time Systems Architectures. CERN-Data Handling Division, DD/85/17, August 1985.
11. Cittolin S. UA1 Data Acquisition System. In: Proc. of the Int. Conf. on Instrumentation for Colliding Beam Physics, SLAC Report 250, Stanford, June 1982.— P. 151—156.

12. Cittolin S., Demeulin M., Giacomelli P. et al. UA1 VME Readout System. In: Proc. of VMEbus in Physics Conf., CERN 86—01, Geneva, 1986.— P. 65—118.
13. Conetti S., Haire M., Kuchela K. Fast, Microprocessor Driven, Data Acquisition for Fermilab Experiment E-705 // IEEE Trans., NS-32, No. 4, 1985, p. 1318—1320.
14. LeVine M. J. The E802 Data Acquisition Complex // IEEE Trans.— 1985.— NS-32, No. 4. P. 1376—1378.
15. Vander Molen A., Au R., Fox R., Glynn T. New Multiprocessor Front End Data Acquisition System at NSCL // IEEE Trans.— 1985.— NS-32, No. 4,— P. 1395—1396.
16. Brisson J. C., Farthouat Ph., Gandois B. et al. The OPAL VMEbus Data Collection System. In: Proc. of VMEbus in Physics Conf., CERN 86—01, Geneva, 1986.— P. 119—134.
17. Gaines I., Areti H., Beil J. et al. Fermilab's Advanced Computer R & D Programm // IEEE Trans.— 1985.— NS-32, No. 4.— P. 1397—1404.
18. Beier G., Kappen L., Lutter R., Schöffel K. et al. MILLE — A Dataflow Controlled Multiprocessor System // IEEE Trans.— 1985.— NS-32, No. 4.— P. 1426—1428.
19. Balamuth D. P., Kutt P. H., Bybell D. P., Van Berg R. A Multiple Processor System for Acquisition and Analysis of Nuclear Physics Data // IEEE Trans.— 1985.— NS-32, No. 4.— P. 1429—1431.
20. Ziem P., Drescher B., Kapper K., Kowallik P. Multiprocessor Aided Data Acquisition at VEDAS // IEEE Trans.— 1985.— NS-32, No. 4, P. 1417—1421.
21. Pietarinen E. UA1 Data Acquisition System // IEEE Trans.— 1985.— NS-32, No. 4, P. 1463—1466.
22. Eichler R. A. HERA Data Acquisition System // IEEE Trans.— 1985.— NS-32, No. 4, P.1490—1493.
23. Parkman C., Perrin Y., Petersen J. et al. VALET-PLUS a VMEbus System for Electronic Equipment Tests Using Your Faviourite Personal Computer. In: Proc. of VMEbus in Physics Conf., CERN 86-01, Geneva, 1986.— P. 259—268.
24. Eckerlin G., Elsen E., Schmitt H.v.d. et al. Front End Processing for a 100 MHz Flash-ADC-System. In: Proc. of VMEbus in Physics Conf., CERN 86-01, Geneva, 1986.— P. 147—150.
25. Jääskeläinen M., Carlen L. Data Acquisition System for Nordball. In: Proc. of VMEbus in Physics Conf., CERN 86-01, Geneva, 1966.— P. 135—146.
26. Beier G., Kappen L., Lutter R., et al MLLE—a Dataflow Controller Multibprocessor System. In: Proc. of VMEbus in PhysicsConf.,CERN 86-01, Geneva, 1986.— P. 151—154.
27. Minor M. M., Shera E. B., Lillberg J. W. Loss-Free Gamma-Ray Counting on the VMebus. In: Proc. of VMEbus in Physics Conf., CERN 86-0-, Geneva, 1986.— P. 169—173.
28. Altaber P., Innocenti P. G., Rausch R. A VME Multiprocessor Architecture for the LEP/SPS Control System. In: Proc. of VMEbus in Physics Conf., CERN 86-01, Geneva, 1986.— P. 216—237.
29. Gournay J. F., Gourey G., Garreau F. et al. The New Control System of the Saciey Linear Accelerator. In: Proc. of VMEbus in Physics Conf., CERN 86-01, Geneva, 1986.— P. 305—312.
30. Martinod P., Mughai G., Savioz J. Semanaz P. A VMEbus Approach for the Control of the Closed Orbit Correction Power Supplies within the SPS Supercycle. In: Proc. of VMEbus in Physics Conf. CERN 86-01. Geneva, 1986— P. 174—189.
31. Taylor B. G. Personal Computer Access to the VME Bus. In: Proc. of VMEbus in Physics Conf., CERN 86-01, Geneva, 1986.— P. 249—258.
32. Alexander J. A VME Interface to an IBM Mainframe Computer. In: Proc. of VMEbus in Physics Conf., CERN 86-01, Geneva, 1986.— P. 291—296.
33. Marry B., Moreton A., Smith A. IBM/VME Channel High Speed Parallel Interface. In: Proc. of VMEbus in Physics Conf., CERN 86-0-, Geneva, 1986.— P. 243—249.
34. Eck C. PRIAM and VMEbus at CERN. In: Proc. of VMEbus in Physics Conf., CERN 86-0-, Geneva, 1986.— P. 3—7.
35. Nash T., Bracker S., Gaines I. Fermilab's Advanced Computer Program, FN-383, FNAL, Chicago, Ill.— 1983.
36. Интерлаб 1600. Система автоматизации научных исследований, проспект ЦЛАНП, София, 1984.
37. Rzymkowski K. VME — New Modular Standart for Microprocessor Applications. XII Международный симпозиум по ядерной электронике, Д13-85-359, Дубна, 1985.— С. 16.
38. Золотухин Ю. Н. VME bus. Концепция и основные характеристики. XII Международный симпозиум по ядерной электронике, Д13-85-359, Дубна, 1985.— С. 15.
39. Futurebus. Draft Proposed Standard P896.1 // IEEE Inc., November 1983.

Статья поступила 11 июня 1986 г.

УДК 681.327

В. А. Канцеров, А. С. Першин

VME — МАГИСТРАЛЬ НОВОГО ПОКОЛЕНИЯ

Современная электронная технология позволяет создавать микропроцессоры, сравнимые по вычислительным возможностям с 32-разрядными супермини-ЭВМ. Параллельно с развитием микропроцессоров идет совершенствование магистралей.

Современные шины ориентированы в основном на многопроцессорную конфигурацию и обеспечивают передачу данных с длиной слова 8, 16, 24, 32 бит, допуская использование микропроцессоров различной разрядности. Адресное пространство подразделяется на области пользователя, супервизора, ввода-вывода и др., каждая из которых достигает 4Г байта. Для обеспечения эффективной работы многопроцессорных комплексов вводятся многоуровневые системы арбитрации и прерываний. Для подключения плат используются накладные разъемы, что повышает надежность соединений. Несмотря на отличие электронных частей спецификаций, практически все новые магистрали используют единый механический стандарт — Евромеханику. Среди современных 32-разрядных шин (Futurebus, NuBus, MULTIBUS-2) особое место занимает VME, наиболее распространенная среди 16-32-разрядных микропроцессорных магистралей. В ноябре 1984 г. образовалась Международная ассоциация по поддержке и распространению архитектуры VME—VITA (VME bus International Trade Association), в которую входит и СССР.

Семейство VME включает три магистрали: собственно VME, последовательную линию передачи сообщений VMS и дополнительную локальную шину, объединяющую до шести модулей.

Протокол магистрали VMS прошел уже несколько редакций, однако его широкое применение ограничивалось отсутствием специализированных БИС контроллера, выпущенных в 1986 г. фирмой Motorola. VMS ориентирована на связь модулей внутри одного крейта, а не на организацию межкрейтного обмена.

Необходимость дополнительной магистрали возникает при организации многопроцессорных систем, процессоры которых имеют локальные ресурсы (дополнительная память, специальные устройства ввода-вывода и др.), общение с которыми перегружает основную магистраль VME. С помощью дополнительной магистрали возможно осуществление межкрейтной связи. Предложено несколько вариантов организации локальной шины: VMX, MVMX32 VMC, VSB.

VMX имеет 12 линий адреса и 32 линии данных. Линии адреса мультиплексированы, что обеспечивает адресное пространство 16М байт. Скорость передачи по VMX несколько выше, чем по VME, за счет простого протокола и меньшей длины самой магистрали. Спецификация VMX прошла уже две редакции и получила довольно широкое распространение.

MVMX32 предложена осенью 1984 г. фирмой Motorola как магистраль памяти, ориентированная на микропроцессор M68020, имеет 32-разрядные мультиплексированные шины адреса и данных, содержит специальную линию CACHE для управления кэш-памятью. Подключение дополнительных процессоров не предусмотрено. Магистраль не нашла поддержки у других фирм.

VMC представляет собой адекватное отображение интерфейса микропроцессора M68000 на контакты разъема P2, имеет 32 линии адреса и 16 линий данных. Системы арбитрации и прерываний такие же, как и у M68000. Разработана и используется в основном в ЦЕРНе.

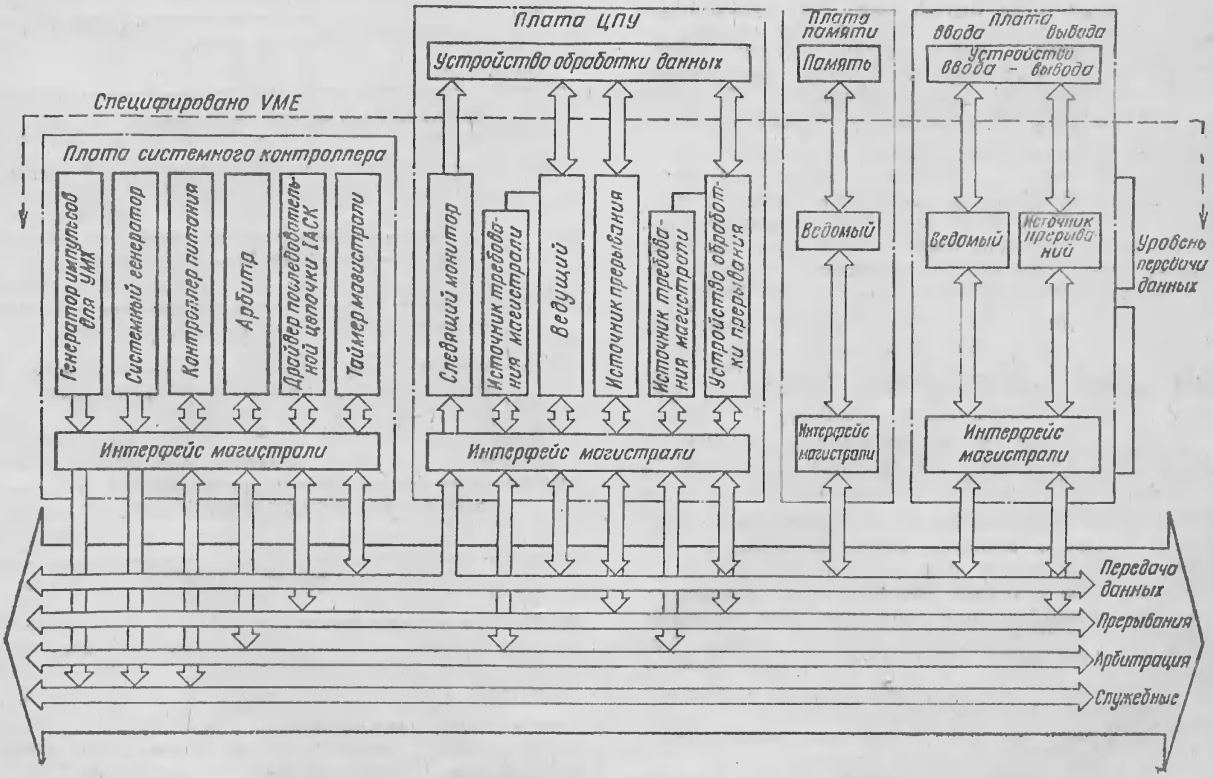


Рис. 1. Структурная схема организации VME магистрали

Наличие стольких разнородных спецификаций исключает унификацию модулей различных изготовителей. Поэтому МЭК создал специальную группу для разработки магистрали, объединяющей функциональные возможности VMX и MVMX32. Результатом работы этой группы стала новая шина VSB (VME Subsystem bus), завоевывающая сейчас все больше сторонников.

Магистраль VME разработана для решения следующих задач:

обеспечение связи между устройствами по шине без нарушения функционирования других подключенных к ней модулей;

спецификация электрических и механических характеристик устройств, необходимых для нормальной совместной работы модулей;

разработка протоколов, определяющих порядок взаимодействия между шиной и подключенными к ней устройствами;

разработка терминологии для описания системных протоколов;

расширение возможностей реконфигурации систем для повышения их эффективности без нарушения совместимости;

создание систем, производительность которых ограничивалась бы возможностями входящих в них устройств, а не системным интерфейсом.

Система VME (рис. 1) состоит из интерфейсных логических схем, набора функциональных модулей, варьируемого в зависимости от конфигурации, и четырех групп сигнальных линий: передачи данных, арбитрации, прерываний и служебной шины.

Магистраль передачи данных

VME включает в себя асинхронную параллельную шину передачи данных, содержащую функциональные модули: ведущего, ведомого, следящего монитора, таймера магистрали.

Магистраль передачи данных позволяет процессорам передавать двоичные данные со скоростью, удовлетворяющей требованиям самых современных 32-разрядных процессоров. Структура магистрали обеспечивает совместное применение в одной системе 8-, 16- и 32-разрядных модулей, работающих на разных частотах.

Ведущий запускает цикл магистрали (последовательность уровней и фронтов на линиях магистрали, в результате которой происходит передача адреса и данных между ведущим и ведомым) для обмена данными с ведомым. *Ведомый* воспринимает цикл, начатый ведущим. При выполнении определенных условий отвечает на него, передавая или принимая данные.

Следящий монитор отслеживает передачу данных по магистрали. При обращении по присвоенным ему адресам сигнализирует размещенной на плате схеме. Например, сигналом может быть запрос прерывания процессору. В такой конфигурации процессор А, записывая данные в ячейку глобальной памяти, находящуюся под наблюдением монитора на плате процессора В, вызовет прерывание процессора В.

Таймер магистрали контролирует время ответа и заканчивает слишком длинные циклы сигналом $\overline{\text{BERR}}$. Так как VME использует асинхронный протокол, ведущий обязан ждать ответа ведомого для завершения цикла. Для предотвращения зависания (при попытках обращения к несуществующим ячейкам) таймер магистрали автоматически выставляет сигнал ошибки $\overline{\text{BERR}}$ по истечении определенного времени, определяемого характеристиками конкретной системы.

Можно выделить три функциональные группы линий, входящих в магистраль передачи данных:

Адресные A01...A31, AM0...AM5, $\overline{\text{DS0}}$, $\overline{\text{DS1}}$, $\overline{\text{LWORD}}$;

передачи данных D00...D31; управляющие $\overline{\text{AS}}$, $\overline{\text{DS0}}$, $\overline{\text{DS1}}$, $\overline{\text{BERR}}$, $\overline{\text{DTACK}}$, $\overline{\text{WRITE}}$.

Адресные линии. Минимальная единица адресации — байт. A02...A31 определяют одну из четырехбайтных групп (совокупность байт, чьи адреса отличаются только в двух младших разрядах), а выбор одного или нескольких байт внутри группы осуществляется линиями $\overline{\text{DS0}}$, $\overline{\text{DS1}}$, A01, $\overline{\text{LWORD}}$ (табл. 1). VME допускает невыровненные передачи данных, при которых происходит обмен данными с нарушением границ четырехбайтных групп. Такая передача позволяет ускорить работу процессоров, использующих другие виды разбиения памяти на слова и длинные слова.

Шесть линий модификации адреса AE0...AM5 несут дополнительную информацию о цикле передачи данных. Комбинации уровней (табл. 2) на этих линиях подразделяются на три группы: специфицированные, определяемые пользователем, резервные.

Таблица 1

Размер передаваемого по магистрали блока

Тип цикла	$\overline{\text{DS1}}$	$\overline{\text{DS0}}$	A01	$\overline{\text{LWORD}}$
«Только адресация»	1	1	×	×
Чтение или запись четного байта				
байт (0)	0	1	0	1
байт (2)	0	1	1	1
Чтение или запись нечетного байта				
байт (1)	1	0	0	1
байт (3)	1	0	1	1
Чтение или запись двух байтов				
байты (0, 1)	0	0	0	1
байты (2, 3)	0	0	1	1
Чтение или запись четырех байтов				
байты (0, 3)	0	0	0	0
Блочное чтение или запись побайтно	*	*	*	1
Блочное чтение или запись по два байта	1	0	*	1
Блочное чтение или запись по четыре байта	0	0	0	0
Чтение — модификация — запись одного байта				
байт (0)	0	1	0	1
байт (1)	1	0	0	1
байт (2)	0	1	1	1
байт (3)	1	0	1	1
Чтение — модификация — запись двух байтов				
байты (0, 1)	0	0	0	1
байты (2, 3)	0	0	1	1
Чтение — модификация — запись четырех байтов				
байты (0, 4)	0	0	0	0
Невыровненные передачи, чтение или запись				
байты (0, 2)	0	1	0	0
байты (1, 3)	1	0	0	0
байты (1, 2)	0	0	1	0

*) Уровни сигналов на этих линиях определяются четностью первого передаваемого байта или слова.
×) Уровни сигналов на этих линиях могут быть любыми.

Таблица 2

Коды линий адресного модификатора

16 код	Линии AM 5 4 3 2 1 0	Адрес (бит)	Режим	Функция
3F	1 1 1 1 1 1	24	С	блочная передача
3E	1 1 1 1 1 0	24	С	область программ
3D	1 1 1 1 0 1	24	С	область данных
3C	1 1 1 1 0 0			резервная
3B	1 1 1 0 1 1	24	П	блочная передача
3A	1 1 1 0 1 0	24	П	область программ
39	1 1 1 0 0 1	24	П	область данных
38	1 1 1 0 0 0			резервная
30...37				резервные
28...2F				резервные
2D	1 0 1 1 0 1	16	С	внешние устройства
2A...2C				резервные
29	1 0 1 0 0 1	16	П	внешние устройства
28	1 0 1 0 0 0			резервная
20...27				резервные
18...1F				определяемые пользователем
10...17				определяемые пользователем
0F	0 0 1 1 1 1	32	С	блочная передача
0E	0 0 1 1 1 0	32	С	область программ
0D	0 0 1 1 0 1	32	С	область данных
0C	0 0 1 1 0 0			резервная
0B	0 0 1 0 1 1	32	П	блочная передача
0A	0 0 1 0 1 0	32	П	область программ
09	0 0 1 0 0 1	32	П	область данных
08	0 0 1 0 0 0			резервная
00...07				резервные

Примечание. С-режим супервизора; П-режим пользователя.

Типы адресов, определяемые пользователем, могут применяться для разделения памяти между отдельными процессорами в мультипроцессорных системах. В этом случае каждый процессор имеет собственное 32-разрядное пространство адресов, защищенное аппаратно.

Объединительная панель может содержать как 16, так и 32 линии передачи данных. В последнем случае используется второй разъем на задней панели модулей.

Управляющие линии

$\overline{AS}$ — синхронизация адреса. Спад сигнала на этой линии сигнализирует ведомому, что информация на адресных шинах верна и может быть использована.

$\overline{DS0}$, $\overline{DS1}$ — стробы данных, $\overline{DTACK}$ — подтверждение данных. В цикле записи спад сигнала на любой из линий $\overline{DS0}$, $\overline{DS1}$ (неважно на какой первой) указывает на истинность установленной ведущим информации, а спадом на $\overline{DTACK}$ ведомый сообщает о ее принятии. В цикле чтения ведомый сопровождает выдачу данных спадом на $\overline{DTACK}$, а ведущий фронтом на $\overline{DS0}$, $\overline{DS1}$ сигнализирует об их приеме.

$\overline{BERR}$ — низкий уровень на этой линии, устанавливаемый ведомым или таймером магистрали сообщает ведущему об ошибке при передаче данных. При попытке записать информацию в память только на чтение, ведомый установкой $\overline{BERR}$ сигнализирует ведущему об ошибке. При использовании сигнала $\overline{BERR}$, низкий уровень на линии подтверждения данных $\overline{DTACK}$ не устанавливается, и наоборот.

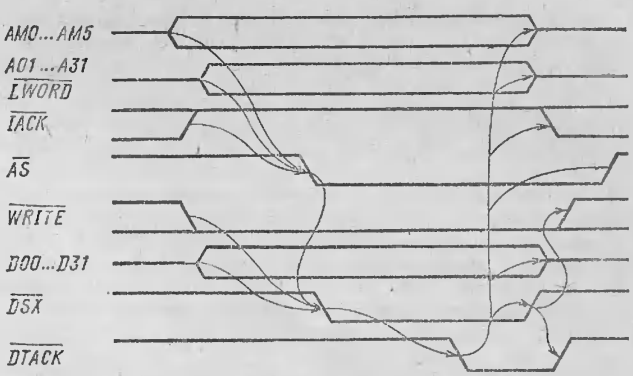


Рис. 2. Временные диаграммы цикла записи

WRITE — запись. Уровень сигнала на этой линии устанавливается до прихода первого из стробов данных и определяет направление передачи. Низкий уровень означает передачу информации от ведущего к ведомому.

В VME определено пять основных циклов магистрали данных: чтение/запись, чтение — модификация — запись, блочное чтение/запись, подтверждение прерывания, «только адресация». Во всех случаях, кроме цикла «только адресация» могут передаваться 8-, 16- и 32-разрядные данные.

Чтение и запись (рис. 2) используются для передачи 1, 2, 3 и 4 байт данных.

Ведомый считает информацию на линиях AM0...AM5, $\overline{LWORD}$, $\overline{IACK}$ A01...A31, истинной в момент прихода строба адреса $\overline{AS}$, а на линиях WRITE, D00...D31 — по получении любого из стробов данных $\overline{DS0}$, $\overline{DS1}$. Так как сигналы на линиях $\overline{DS0}$, $\overline{DS1}$ устанавливаются ведущим после $\overline{AS}$, допустимо дешифрировать адрес в момент их прихода, игнорируя $\overline{AS}$. Ведомый может принять $\overline{DS0}$, $\overline{DS1}$ раньше строба $\overline{AS}$, если линия $\overline{AS}$ тяжело нагружена. Ведущий снимает информацию со всех линий сразу по получении $\overline{DTACK}$ и обязан выключить драйверы до снятия последнего из сигналов $\overline{BBSY}$ и $\overline{AS}$ только при освобождении магистрали.

В цикле *«чтение-модификация-запись»* (рис. 3) происходит чтение информации и ее перезапись по тому же адресу. Этот цикл не может быть прерван другим ведущим, так как строб AS не снимается, что позволяет организовывать семафоры в мультипроцессорных системах.

Пример. Два процессора совместно используют принтер. Имеется ячейка памяти, один из разрядов

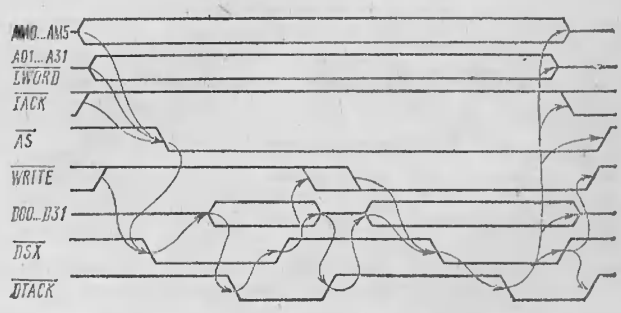


Рис. 3. Временные диаграммы цикла «чтение — модификация — запись»

которой указывает на занятость принтера. Процессор А перед началом печати должен проверять готовность принтера, считывая эту ячейку, и занять его, устанавливая соответствующий бит в цикле записи, указывая, что он занял принтер. В промежутке между этими циклами процессор В, выиграв арбитрацию, может в свою очередь проверить готовность принтера и установить бит его занятости. В результате оба процессора станут одновременно печатать разные тексты. Для предотвращения подобных конфликтов и введен цикл «чтение-модификация-запись».

Блочные передачи (рис. 4) применяются для передачи от 1 до 256 байт данных. Этот цикл используется для обращения к последовательным ячейкам памяти. Ведущий адресует первую ячейку, а затем передает данные, не заботясь об адресе, который автоматически наращивается на нужную величину ведомым в соответствии с типом передаваемых данных. В начале передачи ведомый запоминает адрес в своем адресном счетчике, ведущий последовательно выдает данные, сопровождая соответствующими стробами. Принимая данные, ведомый использует для внутриплатной адресации свой счетчик, наращивая его после каждой передачи.

Адрес в цикле блочной передачи не должен пересекать 256 байтных границ, *изменяться могут только 7 младших разрядов адреса* A01...A07. Поэтому старший адрес вычисляется просто и проверяется лишь однажды, в самом начале передачи, и ведомому нет необходимости проверять на соответствие границам модуля полный 32-разрядный адрес в каждом акте передачи данных, что существенно упрощает схему ведомого. Ведущий сохраняет непрерывный контроль над магистралью данных в течение всего времени передачи, удерживая низкий уровень на линии $\overline{AS}$ (как и в цикле «чтение-модификация-запись»). Если ведущему необходимо передать более 256 байт данных, он должен инициировать несколько циклов блочной передачи.

Разработчики аппаратуры могут предусмотреть аппаратное снятие и установку сигнала $\overline{AS}$ при пересечении адресом 256-байтных границ. В этом случае программист может передавать блоки данных любой длины.

В цикле *«только адресация»* (рис. 5) передачи данных не происходит, ведущий выставляет адрес, стробирует его и спустя некоторое время сам завершает обмен (ведомый не отвечает). Этот цикл применяется для повышения производительности процессорных плат, позволяя процессору начать цикл магистрали прежде, чем выяснится, где находится адресат (на самой плате процессора (локальные адреса) или вне ее). Ведущий завершает цикл магистрали без передачи по ней данных, если расположенная на плате логика после выставления адреса обнаруживает его локальность.

В цикле *«подтверждение прерывания»* (рис. 6) происходит считывание статусной информации источника

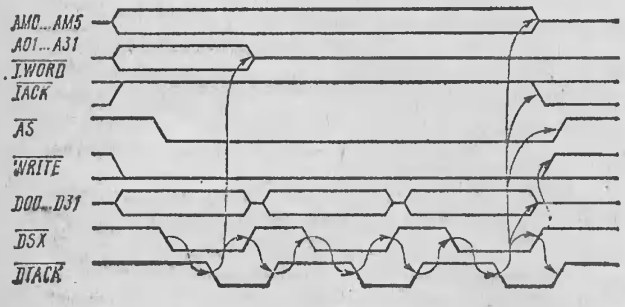


Рис. 4. Временные диаграммы цикла блочной передачи данных

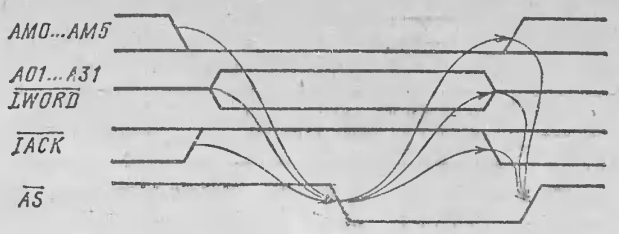


Рис. 5. Временные диаграммы цикла «только адресации».

прерывания. Слово статуса может быть 8-, 16-, 32-разрядным. Устройство обработки прерываний проводит этот цикл при поступлении запроса от источника. Признаком цикла подтверждения прерывания служит низший уровень на линии $\overline{IACK}$.

Магистраль VME имеет раздельные стробы адреса и данных, что позволяет ведущему выставить адрес для следующего цикла в момент передачи данных предыдущего. *Этот способ выставления адреса называется адресным конвейером.* В этом случае платы памяти могут начинать выборку следующих данных, не ожидая окончания передачи предыдущих, что обеспечивает сокращение среднего времени выборки данных.

Шина данных оптимизирована для повышения производительности микропроцессорных систем. Она обеспечивает необходимую пропускную способность любому микропроцессору, поскольку задержки, вносимые магистралью, незначительны по сравнению с собственными временами процессора. Асинхронный протокол позволяет каждому прибору работать с требуемой скоростью. Пропускная способность VME магистрали определяется электрическими характеристиками объединительной панели, быстродействием логики используемых плат и обязательными задержками, вносимыми протоколом.

При определении минимального времени цикла VME (рис. 7) задержки логических схем подключенных модулей не учитываются. Характеристики нагруженной шины обеспечивают время распространения сигнала (15 нс) от первой до последней станции (с учетом отражений). Стробы $\overline{AS}$ и $\overline{DS0}$, $\overline{DS1}$ выставляются ведущим через 35 нс после одновременной выдачи адреса и данных. Протокол разрешает ведомому выдачу сигналов $\overline{DTACK}$, $\overline{BERR}$ не ранее чем через 30 нс после прихода первого из стробов $\overline{DS0}$, $\overline{DS1}$. Получая ответ ведомого, ведущий освобождает магистраль, а новый цикл может быть начат только через 40 нс. Таким

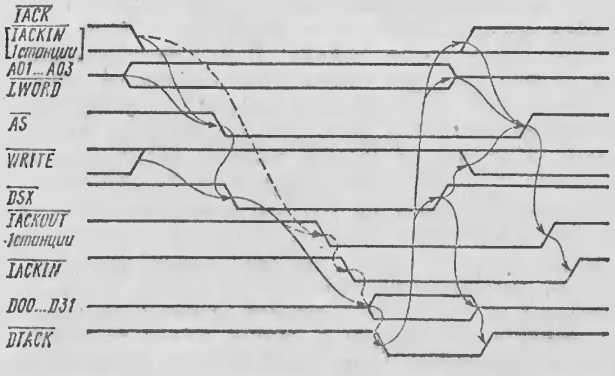


Рис. 6. Временные диаграммы цикла «подтверждение прерывания»

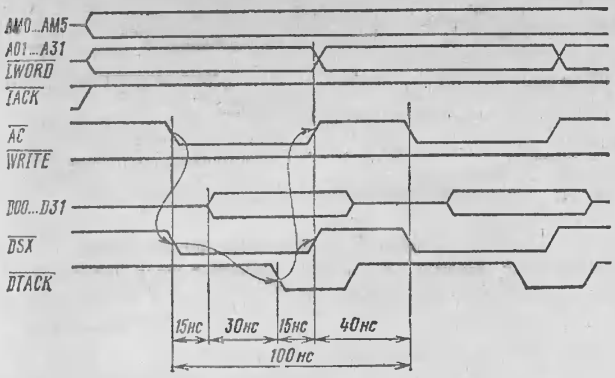


Рис. 7. Максимальная пропускная способность VME магистрали

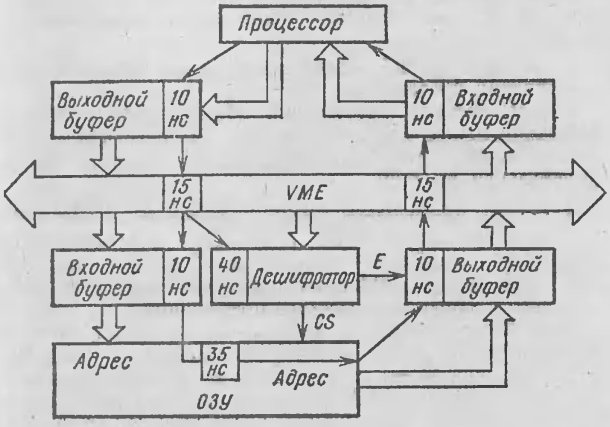


Рис. 8. Реальная скорость передачи данных по магистрали

образом, минимальное время цикла равно 100 нс, что соответствует скорости передачи 40 Мбайт/с.

Рис. 8 иллюстрирует практически достижимую скорость передачи данных по VME магистрали при сегодняшнем уровне технологии. Процессор выполняет операцию чтения быстродействующего модуля памяти, однако данные передаются не в блочном режиме. На каждое слово данных передается свой адрес. Получаемая скорость передачи (28 Мбайт/с) достаточна для большинства применений магистрали.

Арбитрация магистрали

В многопроцессорных системах часто возникает ситуация, когда магистраль требуется одновременно нескольким ведущим или устройствам обработки прерываний. Перед каждым циклом передачи данных по магистрали ведущий или устройство обработки прерываний обязаны получить разрешение на ее использование. Процесс, определяющий, какой из ведущих сможет использовать магистраль при одновременном поступлении требований, называется *арбитрацией*.

Магистраль арбитрации решает две основные задачи: предотвращение одновременного использования шины двумя ведущими и оптимальное распределение магистрали данных по требованиям от различных ведущих.

Она использует: 4 линии требования ($\overline{BR0}$...$\overline{BR3}$); четыре последовательные цепочки предоставления магистрали ($\overline{BG0}$...$\overline{BG3}$); линии «занято» ($\overline{BBSY}$) и «освободить» ($\overline{BCLR}$).

Магистраль арбитрации использует два типа функциональных модулей: арбитр и источник требований магистрали (см. рис. 1).

Арбитр принимает запросы от источников требований магистрали и предоставляет контроль над ней только одному из них при одновременном поступлении нескольких требований. Описаны три типа арбитров магистрали:

приоритетный присваивает жесткие приоритеты линиям требования магистрали: $\overline{BR0}$ (наинизший), $\overline{BR3}$ (наивысший). Вырабатывает сигнал $\overline{BCLR}$ при поступлении требования от ведущего с более высоким приоритетом, чем текущий;

циклический осуществляет циклическую смену приоритетов после каждого акта арбитрации. Если в предыдущем цикле магистраль была предоставлена по требованию линии $\overline{BRn}$, наивысшим приоритетом будет обладать линия $\overline{BRn-1}$. Арбитр с циклической сменой приоритетов может использовать линию $\overline{BCLR}$ для указания ведущему, что поступил запрос более высокого уровня на использование магистрали;

одноуровневый воспринимает только запросы по линии $\overline{BR3}$ и выдает разрешение по последовательной цепочке BG3IN/BG3OUT.

Кроме перечисленных типов арбитров допускаются и любые другие. Например, арбитр может рассматривать линию $\overline{BR3}$ как обладающую наивысшим приоритетом, а на линиях требования $\overline{BR0}$...$\overline{BR2}$ менять приоритет циклически.

Источник требований магистрали размещается на одной плате с ведущим или устройством обработки прерываний, выставляет запрос $\overline{BRn}$ на передачу по магистрали и ожидает разрешение по последовательной цепочке $\overline{BGnIN}$/$\overline{BGnOUT}$. Устанавливает сигнал $\overline{BBSY}$ (удерживая его до окончания обмена) по получению контроля над магистралью. Существуют два способа освобождения магистрали ведущим:

по выполнению — после завершения передачи всех данных;

по требованию — при появлении сигнала на одной из шин $\overline{BR0}$...$\overline{BR3}$ / или $\overline{BCLR}$. В этом случае сокращается среднее время доступа по магистрали, если имеется один ведущий, использующий ее значительно чаще остальных.

Следует отметить, что определены только способы освобождения магистрали, никаких временных ограничений на работу ведущего не накладывается. Например, контроллеру диска, получившему в самом начале передачи блока из 256 слов в режиме прямого доступа к памяти сигнал $\overline{BCLR}$, может потребоваться значительное время для нормального завершения операций на магистрали и предоставления ее другим ведущим.

Особое место занимают операции при появлении сигнала $\overline{ACFAIL}$ (авария сетевого питания), когда необходимо скорейшее освобождение магистрали. Рекомендуется, чтобы ведущий, получив $\overline{ACFAIL}$, освободил магистраль не позже чем через 200 мкс.

Рассмотрим последовательность сигналов при одновременном поступлении двух запросов разных уровней (рис. 9). Два источника требований магистрали A и B одновременно выставляют сигналы на шинах $\overline{BR1}$ и $\overline{BR2}$ соответственно. Приоритетный арбитр отвечает сигналом $\overline{BG2IN}$ в первой станции крейта. Получая сигнал $\overline{BG2IN}$ по последовательной цепочке $\overline{BG2IN}$/

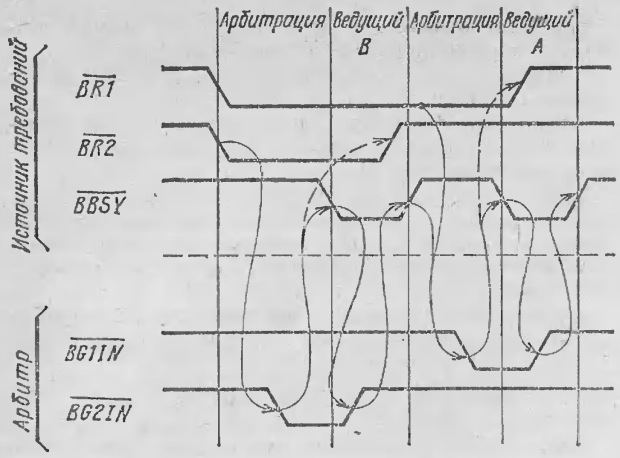


Рис. 9. Последовательность сигналов при одновременном поступлении двух запросов разных уровней

BG2OUT, источник B запрещает его дальнейшее прохождение, выставляет сигнал занятости магистрали BBSY и снимает требование с линии BR2, сообщая одновременно своему ведущему о доступности магистрали данных. Следующий цикл арбитрации начнется в момент снятия сигнала BBSY ведущим B. Сигнал BBSY может быть снят по окончании работы с магистралью или во время последнего цикла передачи данных. В этом случае арбитрация будет происходить одновременно с передачей данных, а новый ведущий, выигравший ее, сможет начать свои операции сразу по окончании цикла передачи данных (по снятию сигнала AS). При блочных передачах сигнал BBSY рекомендуется снимать в конце блока, так как арбитрация проводится по снятию сигнала BBSY, а за время передачи блока возможен приход запросов более высокого уровня, которые смогут принять участие только в следующем цикле арбитрации.

Магистраль прерываний

Протокол VME описывает магистраль прерываний, включающую в себя линии, необходимые для выработки, приема и обслуживания прерываний. Любая система, использующая эту магистраль, содержит некоторый набор программ, называемых подпрограммами обслуживания прерываний. Каждая из этих подпрограмм может рассматриваться как отдельная задача, запускаемая соответствующим прерыванием. Спецификация VME не накладывает никаких ограничений на эти подпрограммы.

Процесс прерывания можно разделить на три фазы (рис. 10):

выработка запроса прерывания — с момента установки низкого уровня источником прерывания на шине запроса IRQn до получения магистрали данных устройством обработки прерываний в цикле арбитрации;

подтверждение прерывания — устройство обработки прерываний считывает по магистрали данных статус источника прерывания и передает управление соответствующей подпрограмме;

обслуживание прерывания — работает подпрограмма обслуживания прерывания (не определяется протоколом VME).

Магистраль прерываний (рис. 11) использует:

IRQ1...IRQ7 — 7 линий запроса прерывания. IRQ7

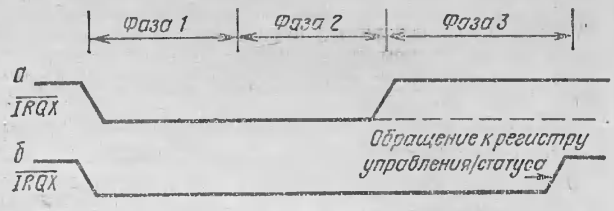


Рис. 10. Три фазы процесса прерывания:
а — снятие запроса в цикле «подтверждение прерывания»; *б* — снятие запроса по обращению к регистру

в приоритетных системах обладает наивысшим приоритетом.

IACK — подтверждение прерывания. Соединяет все станции крейта с контактом IACKIN первой станции. Низкий уровень на этой линии инициирует последовательную цепочку подтверждения прерывания через ее драйвер.

IACKIN/IACKOUT — последовательная цепочка подтверждения прерывания.

Работу системы прерываний обеспечивают три типа функциональных модулей (см. рис. 1): источник прерываний, модуль обработки прерываний, драйвер последовательной цепочки.

Источник прерываний генерирует запрос прерывания по одной из линий IRQ1...IRQ7. При получении подтверждения сообщает устройству обработки прерываний свой статус, позволяющий последнему запустить программу обработки данного прерывания.

Устройство обработки прерываний выделяет запросы прерывания, получает магистраль данных в цикле арбитрации, считывает в специальном цикле магистрали данных (подтверждения прерывания) статусную информацию источника прерывания, после чего запускает подпрограмму обслуживания. Устройство может занимать любую станцию крейта.

Драйвер последовательной цепочки располагается в первой станции крейта. В его функции входит инициация последовательной цепочки подтверждения прерывания, гарантирующей участие только одного источника в цикле подтверждения прерывания.

Процесс прерывания начинается в момент посылки запроса по одной из линий IRQn источником прерывания. Устройство обработки прерываний, обнаруживая

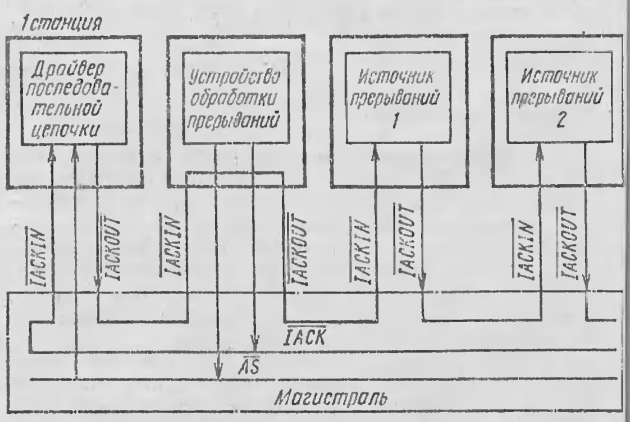


Рис. 11. Линии магистрали прерывания

требование от источника, запрашивает магистраль данных в стандартном цикле арбитрации и ожидает разрешение на использование; получая его, запускает цикл подтверждения прерывания (см. рис. 6), выставляя на шинах адреса A01...A03 уровень обрабатываемого прерывания и сопровождая его сигналом IACK. Низкий уровень сигнала на линии IACK указывает ведомому, что инициирован именно цикл подтверждения прерывания.

Драйвер последовательной цепочки передает сигнал подтверждения с шины IACK на вход IACKIN первой станции в момент прихода любого из стробов данных (DS0 или DS1).

Получая подтверждение, источник проверяет выполнение следующих условий.

запрашивал ли *он* прерывание;

соответствует ли *его* приоритет уровням на линиях A01...A03;

ожидаемая ведущим разрядность данных (шины DS0, DS1, LWORD) больше или равна разрядности его слова статуса.

При невыполнении хотя бы одного условия источник прерывания пропускает сигнал дальше по последовательной цепочке с IACKIN на IACKOUT. Если все условия выполнены, источник выставляет свой статус на шины данных, сообщая об этом сигналом DTACK. Принимая статусную информацию источника по шине данных, устройство обработки прерываний освобождает магистраль, заканчивая цикл подтверждения прерывания.

Многие БИС периферийных контроллеров способны генерировать запросы прерывания. Однако не существует стандартного метода снятия этих запросов.

Протокол VME определяет два способа снятия запроса прерывания с шин IRQn: по обращению к регистру (RORA); по подтверждению прерывания (ROAK).

Процессор в подпрограмме обработки прерывания считывает статусный регистр или записывает регистр управления периферийного устройства, которое интерпретирует это обращение как сигнал на снятие запроса. Этот метод позволяет программисту идентифицировать в программе обработки источник прерывания.

Периферийное устройство снимает запрос автоматически уже в цикле подтверждения прерывания, освобождая программиста от заботы о нем.

Системы, использующие прерывания, могут быть подразделены на две группы: с централизованной обработкой запросов, имеющих только один модуль, обрабатывающий все запросы прерываний, и распределенной обработкой запросов, включающих несколько модулей (до 7), каждый из которых обрабатывает свою группу запросов прерываний.

В системах с централизованной обработкой запросов все прерывания принимаются одним устройством обработки, все программы обслуживания прерываний выполняются одним процессором. Такая структура применяется в машинах или системах управления, где главный процессор (супервизор) координирует работу подчиненных устройств, которые, как правило, представляют собой специализированные процессы. В этих системах каждая из 7 линий запроса имеет собственный приоритет, причем наивысший — IRQ7.

Системы с распределенной обработкой запросов включают от 2 до 7 центров обработки прерываний, каждый из которых обрабатывает свою группу запросов (от 1 до 6). Внутри этих групп запросы могут иметь свои приоритеты. Такая структура удобна для многопроцессорных систем с распределенной обработкой данных, включающих несколько равноправных процессоров. Процессору A, чтобы связаться с B, достаточно запросить прерывание по одной из соответствующих процессору B линий IRQn. При одновременном возникновении запросов прерывания в разных группах очередность обслуживания будет определяться приоритетом соответствующего модуля обработки прерываний на магистрали передачи данных.

Служебная магистраль

Служебная магистраль определяет сигналы, предназначенные для организации работы модулей VME, содержит функциональные модули системного генератора, генератора тактовых импульсов для последовательной магистрали, контроллера питания (см. рис. 1).

Системный генератор формирует импульсы с частотой 16 МГц, задает стандартную частоту для определения временных интервалов, располагается на плате системного контроллера в первой станции (его работа никак не синхронизирована с сигналами на шинах VME).

Генератор тактовых импульсов для последовательной магистрали выдает сигналы специальной формы, используемые интерфейсами шины VMS, расположенными на платах VME. Параметры сигнала заданы в спецификации VMS магистрали.

Контроллер питания обеспечивает необходимую информацию о состоянии источника питания. Как правило, источники питания подключаются к сети переменного тока, поэтому контроллер питания следит за сетью и выдает сигнал о ее сбоях — ACFAIL. Этот сигнал вызывает аварийное завершение работы системы. Иногда контроллер питания имеет специальный тумблер, позволяющий «вручную» перезапускать систему без выключения питания.

Служебные линии VME:

SYSCLC	; шина системного генератора
SYSRESET	; сброс системы, может иметь ручное управление из разных мест
SYSFAIL	; неработоспособность системы — возникновение аварийных ситуаций в модулях системы
ACFAIL	; авария сетевого питания, управляется контроллером питания
+5 В	; основной источник питания
+12 В	; питание RS-232 драйверов МОП-микросхем
—12 В	; питание аналоговых приборов и получение —5,2 В для микросхем ЭСЛ
+5В STDBY	; батарейное питание для памяти, календаря и пр.
RESERVED	; резервные, не могут быть использованы разработчиками.

Включение питания

После подачи питающего напряжения контроллер питания устанавливает высокий уровень на линии ACFAIL и в течение 200 мс поддерживает низкий на SYSRESET (рис. 12). Для управления линией SYSRESET используются драйверы с открытым коллектором, что дает возможность модулям задержать снятие сигнала SYSRESET на время, необходимое для выполнения процедуры начального пуска.

Многие модули выполняют программы самотестирования при включении питания. Во время их выполнения на линии SYSFAIL, также управляемой драйверами с открытым коллектором, должен поддерживаться

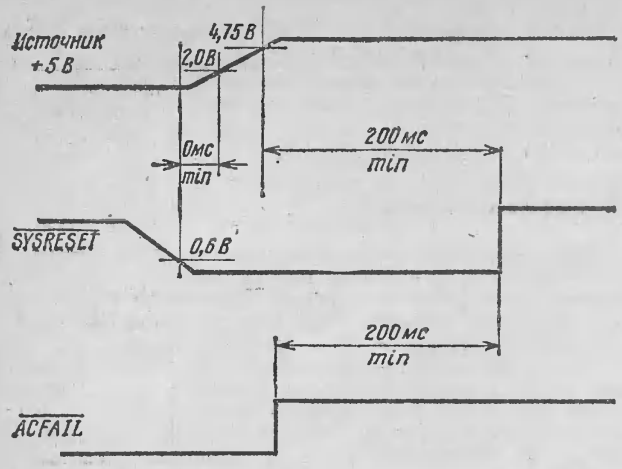


Рис. 12. Временные соотношения сигналов при запуске системы

низкий уровень. Так как сигнал SYSFAIL, в отличие от SYSRESET, не запрещает обмен по шине VME, именно он использован для обеспечения необходимого времени самотестирования всем модулям системы. Для этого вводится специальный регистр на платах, разряд которого определяет уровень на SYSFAIL. При включении питания этот разряд устанавливается и может быть снят записью в регистр после окончания программ самотестирования. Таким способом обеспечивается время, достаточное для выполнения всех тестовых программ.

Требования к электрическим параметрам объединительной панели и приемопередатчикам

К объединительной панели, представляющей собой одну или две многослойные печатные платы длиной не свыше 500 мм, может быть подключено через специальные разъемы до 21 модуля. При использовании разъема P1 магистраль имеет 24 разряда адреса и 16 разрядов данных (табл. 3), а при добавлении разъема P2 — 32 разряда адреса и 32 разряда данных (рис. 13). Разъемы должны удовлетворять следующим требованиям:

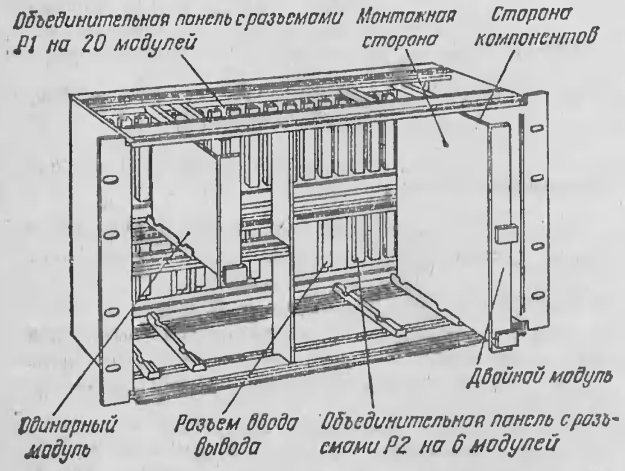


Рис. 13. Общий вид крейта VME с модулями

Максимально допустимое напряжение между соседними контактами, В	100
Сопротивление контакта при максимальном токе, МОм	50
Сопротивление между соседними контактами, МОм	100
Максимальный ток контакта, А	1,5

Низкий уровень установившегося сигнала передатчиков шины должен быть меньше 0,6 В (для приемников — меньше 0,8 В), высокий — больше 2,4 В (для приемников — больше 2,0 В). Магистраль VME требует использования определенных приемопередатчиков (таблицы 4, 5).

Протокол VME магистрали разработан с учетом времен переходных процессов на шине. Времена предустановки и удержания сигналов учитывают, что сущест-

Таблица 3

Назначение контактов разъемов P1 и P2

Контакт	Разъем P1			Разъем P2
	A	B	C	B
1	D00	BBSY	D08	+5В
2	D01	BCLR	D09	ЗЕМЛЯ
3	D02	ACFAIL	D10	РЕЗ.
4	D03	BG0IN	D11	A24
5	D04	BG0OUT	D12	A25
6	D05	BG1IN	D13	A26
7	D06	BG1OUT	D14	A27
8	D07	BG2IN	D15	A28
9	ЗЕМЛЯ	BG2OUT	ЗЕМЛЯ	A29
10	SYSCLK	BG3IN	SYSFAIL	A30
11	ЗЕМЛЯ	BG3OUT	BERR	A31
12	DS1	BR0	SYSRESET	ЗЕМЛЯ
13	DS0	BR1	WORD	+5В
14	WRITE	BR2	AM5	D16
15	ЗЕМЛЯ	BR3	A23	D17
16	DTACK	AM0	A22	D18
17	ЗЕМЛЯ	AM1	A21	D19
18	AS	AM2	A20	D20
19	ЗЕМЛЯ	AM3	A19	D21
20	IACK	ЗЕМЛЯ	A18	D22
21	IACKIN	SERCLC	A17	D23
22	IACKOUT	SERDAT	A16	ЗЕМЛЯ
23	AM4	ЗЕМЛЯ	A15	D24
24	A07	IRQ7	A14	D25
25	A06	IRQ6	A13	D26
26	A05	IRQ5	A12	D27
27	A04	IRQ4	A11	D28
28	A03	IRQ3	A10	D29
29	A02	IRQ2	A09	D30
30	A01	IRQ1	A08	D31
31	—12В	+5В (бат.)	+12В	ЗЕМЛЯ
32	+5В	+5В	+5В	+5В

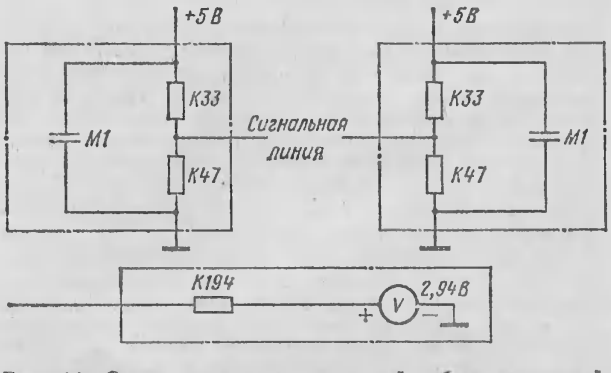


Рис. 14. Схема согласования линий объединительной панели

Таблица 4

Параметры передатчиков

Линии	Iol min мА	Uol max В	Ioh mix мА	Uoh min В	Ios min мА	Ios max мА	Тип микросхемы
$\overline{DR0}$ $\overline{DS1}$ $\overline{AS}$	64	0.6	3	2.4	50	225	К531АП4 К1531АП4 К1531АП5
A01...A31 D00...D31 AM0...AM5 $\overline{IACK}$ $\overline{LWORD}$ $\overline{WRITE}$	48	0.6	3	2.4	50	225	К1531АП5
SERCLC SYSCLC $\overline{BCLR}$	64	0.6	3	2.4	50	225	К531АП4 К1531АП4 К1531АП5
$\overline{BGXOUT}$ $\overline{IACKOUT}$	8	0.6	0.4	2.7			микросхемы ТТЛ
$\overline{BR0}$...$\overline{BR3}$ $\overline{BBSY}$ $\overline{IRQ1}$...$\overline{IRQ7}$ $\overline{DTACK}$ $\overline{BERR}$ $\overline{SYSFAIL}$ $\overline{SYSRESET}$ $\overline{ACFAIL}$ $\overline{SERDAT}$	48	0.6					К531ЛА13 (открытый коллектор)

Примечание: Iol — выходной ток низкого уровня;
Uol — выходное напряжение низкого уровня;
Ioh — выходной ток высокого уровня;
Uoh — выходное напряжение высокого уровня;
Ios — ток короткого замыкания.

Таблица 5

Параметры приемников

Линии	I Iol+Il I max мкА	I Ioh + Ih I max мкА	Ct max пФ	Тип микросхем
$\overline{DS0}$ $\overline{DS1}$ $\overline{AS}$	450	100	20	К555АП3 К555АП4 К555АП5
A01 — A31 D00 — D31 AM0 — AM5 $\overline{IACK}$ $\overline{LWORD}$ $\overline{WRITE}$	700	150	20	К555АП3 К555АП4 К555АП5
SERCLC SYSCLC $\overline{BCLR}$	600	50	12 (с драйвером) 20 (без драйвера)	К555АП3 К555АП4 К555АП5
$\overline{BGXOUT}$ $\overline{IACKOUT}$	600	50	20	К555АП3 К555АП4 К555АП5
$\overline{BR0}$ — $\overline{BR3}$ $\overline{BBSY}$ $\overline{IRQ1}$ — $\overline{IRQ7}$ $\overline{DTACK}$ $\overline{BERR}$ $\overline{SYSFAIL}$ $\overline{SYSRESET}$ $\overline{ACFAIL}$ $\overline{SERDAT}$	400	50	20	К555АП3 К555АП4 К555АП5

Примечание. I Iol — ток утечки на выходе при напряжении низкого уровня,
I Il — входной ток низкого уровня;
I Ioh — ток утечки на выходе при напряжении высокого уровня;
I Ih — входной ток высокого уровня;
Ct — общая емкость, вносимая контактом.

вующие передатчики не обеспечивают требуемые уровни сигналов до момента затухания отражений от концов линий. На обоих концах линий предусмотрены согласующие резисторы (рис. 14), которые обеспечивают высокий уровень на линиях, управляемых передатчиками с тристабильными выходами и схемами с открытым коллектором; уменьшают отражение от концов линий; улучшают фронты сигналов при отключении тристабильных передатчиков.

Проводники объединительной панели обеспечивают характеристическое волновое сопротивление не менее 100 Ом. Однако отверстия на платах и контакты разъемов, внося дополнительную емкость, снижают его. Сопротивление линий без вставленных плат 50-60 Ом считается достаточным для нормальной работы системы.

Платы VME могут быть одно- и двукратной высоты и имеют один или два 96-контактных разъема. В соответствии с конструктивами МЭК 297-1, 297-3, 297-3A, 603-2 *(Евромеханика)* на плате двойной высоты при использовании многослойного монтажа размещается до 100 микросхем в стандартных 16-выводных корпусах DIP.

Адрес для справок: 115409, Москва, М-409, Каширское шоссе, 31, МИФИ, кафедра № 7, телефон 324-81-54.

ЛИТЕРАТУРА

1. Proceedings of VME in Physics Conference / CERN, Geneva; 1985.
2. VME Specification / Rev. C. 1. October — 1985.
3. VME bus TRAINING COURSES Presented by Plessey Microsystems, CERN, Geneva; 1985.

Статья поступила 28 ноября 1986 г.

О семинаре «МП»

Второй семестр работы учебного семинара «МикроЭВМ и микропроцессоры: основы применения» (о его тематике см. в «МП», № 2, 1987, с. 40) будет проходить с января по июнь 1988 г. Абонементы будут продаваться в кассе Политехнического музея (подъезд № 9, тел. 923-69-16).

УДК 681.327.8

В. А. Коломейцев, Ю. А. Степченков, А. В. Филин

ИНТЕРФЕЙС С ПОСЛЕДОВАТЕЛЬНЫМ АРБИТРАЖЕМ: РЕАЛИЗАЦИЯ И ПУТИ УСОВЕРШЕНСТВОВАНИЯ

Магистральные модульные структуры играют важную роль в развитии цифровых вычислительных систем. Такие структуры обладают преимуществами в отношении критерия «производительность — стоимость», простоты организации межмодульного взаимодействия, контроля передач и других операций на уровне системного интерфейса, степени стандартизации технических решений, построения систем с различными уровнями сложности и др. [1]. Одной из подобных структур является 32-разрядная магистраль VME (Versa Module Europe) — МЭК 821, обладающая возможностями для построения многопроцессорных систем повышенного быстродействия, большого объема памяти, с архитектурой открытого типа [2].

Достоинства интерфейса VME: высокие технические характеристики; относительно высокий уровень проработки идеологии системного параллельного интерфейса; достаточно гибкий арбитраж и относительно простой протокол обмена; возможность построения на его основе простых и соответственно дешевых систем с небольшим объемом оборудования; объединение в одной системе модулей, работающих с 8-, 16- и 32-разрядными данными и 16-, 24- и 32-разрядными адресами.

Тем не менее данный интерфейс не свободен от определенных недостатков. Здесь можно отметить отсутствие контроля передач, использование последовательного арбитража по цепочечным линиям (daisy chain), невозможность осуществления «широковещательных» передач типа «один — всем» и «все — одному» (broadcast и broadcall), большое число функциональных линий, которое вызывает необходимость применения второго интерфейсного разъема для обеспечения полного набора свойств, и др. Такие недостатки существенно ограничивают использование VME в системах, где достоверность обработки информации, надежность функционирования, а также живучесть (отказоустойчивость) являются определяющими характеристиками. Другие же системы вполне могут базироваться на интерфейсе VME.

В СССР VME рассматривается как один из возможных интерфейсов для микроЭВМ [3]. Уже сейчас имеется достаточно много импортных комплексов с этим интерфейсом. В то же время в отечественной литературе отсутствуют публикации об особенностях реализации этого интерфейса. Успех вновь разрабатываемых систем зависит от многих составляющих, и в частности от глубины проработки закладываемых технических решений, некорректности в которых могут проявиться не на простых конфигурациях систем, а значительно позднее — на сложных системах. С другой стороны, незнание возможных путей улучшения характеристик систем на начальных этапах делает, как правило, невозможным использование их на последующих этапах разработки и эксплуатации систем.

Все это позволяет считать, что настоящий момент является своевременным как для предложения решений в области схемотехники, так и для рассмотрения путей улучшения характеристик систем на базе интерфейса VME, тем более, что такие предложения начали появляться в зарубежной печати.

Программируемые ИМС для реализации узлов

Устройство выдачи запросов на передачу данных (Requester) с полным набором функций, предусмотренных в спецификации на интерфейс VME, может быть реализовано с использованием одной программируемой интегральной микросхемы (ИМС) типа PAL (Programmable Array Logic) и двух передатчиков шины [4, 5]. Преимущество такого построения устройства заключается прежде всего в экономии места на плате, минимизация которого является особенно актуальной задачей при создании микрокомпьютеров с достаточно сложным для них интерфейсом VME.

В схеме устройства, предложенной в [4], используется элемент PAL16R4, опытные образцы которого уже освоены отечественной промышленностью. Особенностью этого элемента является наличие входа тактовых импульсов, предназначенного для устранения необходимости введения в схему пассивных элементов задержки (ПЭЗ), традиционно применяемых схемотехниками при проектировании устройств ВТ. Подключение ПЭЗ к выходам PAL нецелесообразно, поскольку из-за отсутствия в имеющихся ИМС PAL выходов с открытым коллектором снижается помехозащищенность схем, построенных на их основе.

К сожалению, в предложенной автором [4] схеме, на наш взгляд, имеется ряд нарушений требований логического характера и в отношении электрических характеристик. Например, из-за асинхронности прихода активного уровня сигнала BGXIN по отношению к сигналу SYSCLK и при возможном перекосе сигналов (неодинаковое время прохождения сигналов от входов к выходам) вероятна такая ситуация, когда на выходном контакте 17 появится активный уровень BRX, а на выходном контакте 15 — активный уровень BGXOUT, что в следующем такте приведет к переходу сигнала на контакте 15 в пассивное состояние. Это, естественно, противоречит спецификации [5], поскольку сброс сигнала BGXOUT является блокированным по отношению к сигналу BGXIN.

Далее, в предлагаемой схеме вариант освобождения шины по приходу активного уровня сигнала BCLR, во-первых, рассматривается как альтернативный по отношению к другим вариантам, а во-вторых, не работает в принципе, поскольку активный уровень RBC (контакт 12) появится лишь тогда, когда (при удовлетворении остальным условиям перехода в активное состояние RBC по приходу активного BCLR) сигнал DWB перейдет в пассивное состояние, т. е. когда шина устройству будет уже не нужна в любом случае. Таким образом, перехвата шины устройством с более высоким приоритетом, по существу, не произойдет.

Наконец, в схеме существует еще одна некорректность, связанная с невыполнением требований электрических характеристик приемников сигналов стробирования. Согласно спецификации приемники, подключаемые к линиям передачи сигналов стробирования, должны иметь гистерезис по крайней мере на 200 мВ, а PAL не отвечает этим требованиям. Поэтому на входе с линии шины SYSCLK следует поставить один из элементов 74LS240/241/244 или аналогичную схему отечественного производства К555АП3П/АП4П.

В предлагаемой схеме (рис. 1) внесены некоторые изменения в программирование PAL16R4 (табл. 1) и согласованы выполняемые функции со спецификацией.

Предлагаем также свой вариант исполнения арбитра (рис. 2), построенного на одной ИМС PAL20R8 (табл. 2) и способного работать во всех трех режимах: PRI, RR и ONE, предусмотренных в [5]. Он может состоять из одной ИМС лишь в том случае, если тактовый вход связан с тактовым генератором непосредствен-

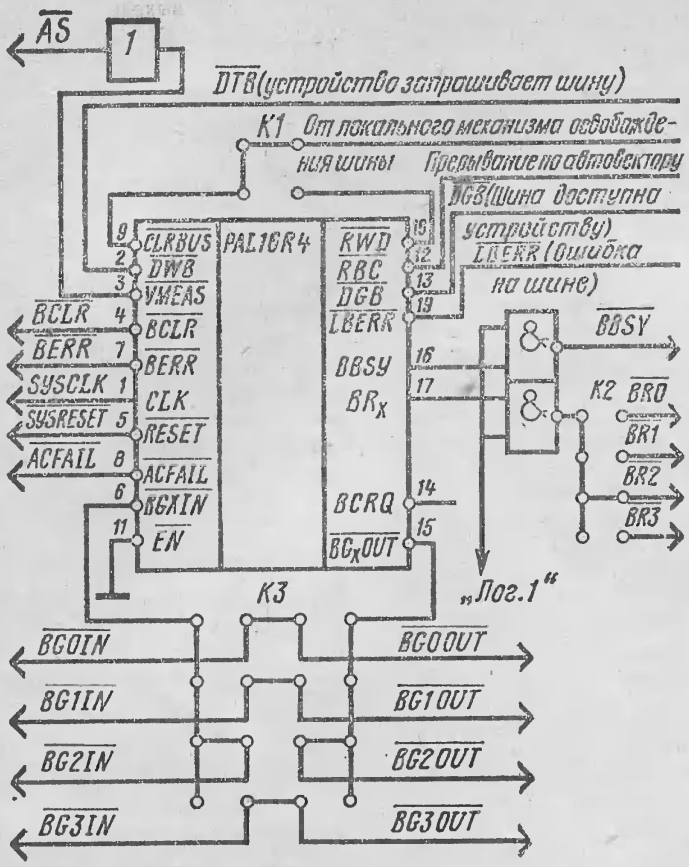


Рис. 1. Электрическая схема устройства Requester

Таблица 1

REQUESTER. СПЕЦИФИКАЦИЯ PAL

```
/BCRQ:=BRX·/CLRBUS·RESET
      +/BCRQ·BRX·RESET
      +/BCRQ·/BRX·BBSY·
      ·/BGXOUT·/BGXIN·RESET
      +/BRX·BBSY·BGXOUT·
      ·/CLRBUS·/BGXIN·RESET

/BRX:=BBSY
     +BCRQ·/BRX·/CLRBUS
     +/BRX·DWB
     +/BRX·/BBSY·DWB
     +/BCRQ·/BRX·/DWB
     +/DWB·/ACFAIL·BGXIN
     +/RESET

/BBSY:=/BRX·/BBSY
      +BRX·/BBSY·BGXIN
      +/BRX·/CLRBUS·BGXIN
      +/BCRQ·/BRX·BGXIN
      +/BGXOUT
      +/RESET

/BGXOUT:=/BRX·/BBSY·/BGXIN·
        ·RESET
        +/BGXOUT·/BGXIN

IF (VCC)/DGB=BBSY·VMEAS·
             +/DGB·BBSY
             +/DGB·BBSY·
             ·/VMEAS

IF (VCC/RWD:=/DGB·VMEAS·
            ·DWB

IF (VCC)/RBC:=/DGB·/BCLR·
              ·VMEAS·/DWB
              +/DGB·/ACFAIL·
              VMEAS·/DWB
              +/RBC·DGB
IF (VCC)/LBERR=/BERR·/DGB
```

но, а не через шину. Это требование удовлетворяется вполне естественным образом, поскольку и тактовый генератор, и арбитр располагаются на одной плате. Выход EN управляет входом разрешения выходных элементов с тремя состояниями, поэтому рекомендуется назначать контакт для сигнала EN как можно ближе к этому входу.

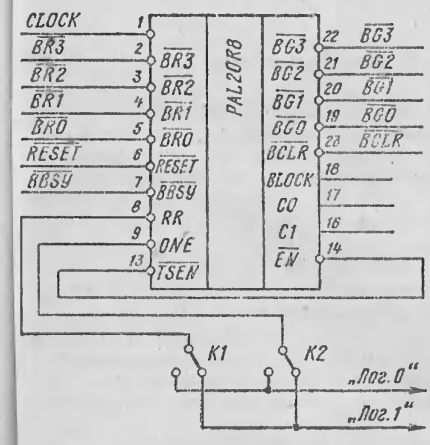


Рис. 2. Электрическая схема арбитра

В схеме арбитра активные уровни всех входных, а также выходных сигналов BG3, BG2, BG1, BG0 — низкие. Счетчик циклического изменения приоритетов для варианта RR построен на C0 и C1. Выход BLOCK является вспомогательным и служит для организации задержки сигнала разрешения с выхода EN.

Способ улучшения характеристик арбитра

Интерфейс VME относится к системам с последовательным арбитражем (рис. 3, *а*), которые отличаются простотой реализации функций арбитража и отсутствием логических ограничений на число устройств в системе. Однако увеличение времени арбитража и снижение реактивности системы прерываний при увеличении числа устройств одного уровня приоритета — это препятствия для построения высокопроизводительных многопроцессорных систем расширенных конфигураций.

Источником сигналов разрешений доступа к магистрали BR3 ... BR0 является арбитр системы, фиксируемый на посадочном месте A01 [5], на котором также производится перекоммутация параллельного сигнала IACK в цепочечный сигнал IACK OUT. По сравнению с линиями BGX, соединяющими последовательно только соответствующие одноуровневые устройства прямого доступа, линия IACK IN/OUT соединяет все устройства Interrupt Requester [5] независимо от уровня приоритета. Именно здесь число последовательно соединенных устройств, а следовательно и время распространения сигнала, может быть велико. Размещение арбитра системы в центре крейта (каркаса) на позиции A10 при общем числе позиций, равном 20 (рис. 3,*б*), уменьшает вдвое максимальное время трансляции указанных сигналов. Такое подключение арбитра дает эффект увеличения вдвое числа уровней прямого доступа и уровней прерывания при сохранении исходного числа линий интерфейса VME. Линии BR3 ... ...BR0 в месте подключения арбитра разрываются и для левой (Left) и правой (Right) группы модулей (M), им присваиваются названия BR3L ... ...BR0L и RR3R ... BR0R соответственно. Приоритет правой и левой половин крейта фиксируется заранее, и

Таблица 2

Арбитр. Спецификация PAL

```
/BG3:=/BR3·BBSY·BG3·BG2·BG1·BG0·/C0·/C1·RESET
     +/BR3·BR2·BR1·BR0·BBSY·BG3·BG2·BG1·BG0·C0·/C1·RESET·ONE
     +/BR3··BR1·BR0·BBSY·BG3·BG2·BG1·BG0·/C0·C1·RESET·ONE
     +/BR3·BR0·BBSY·BG3·BG2·BG1·BG0·C0·C1·RESET·ONE
     +/BG3·BBSY·/C0·/C1·RESET
     +/BG3·BG2·BG1·BG0·BBSY·C0·/C1·RESET·ONE
     +/BG3·BG1·BG0·BBSY·/C0·C1·RESET·ONE
     +/BG3·BG0·BBSY·C0·C1·RESET·ONE

/BG2:=/BR2·BR3·BBSY·BG3·BG2·BG1·BG0·/C0·/C1·RESET·ONE
     +/BR2·BBSY·BG3·BG2·BG1·BG0·C0·/C1·RESET·ONE
     +/BR2·BR1·BR0·BBSY·BG3·BG2·BG1·BG0·/C0·C1·RESET·ONE
     +/BR2·BR0·BR3·BBSY·BG3·BG2·BG1·BG0·C0·C1·RESET·ONE
     +/BG2·BG3·BBSY·/C0·/C1·RESET·ONE
     +/BG2·BBSY·C0·/C1·RESET·ONE
     +/BG2·BG1·BG0·BG3·BBSY·/C0·C1·RESET·ONE
     +/BG2·BG0·BG3·BBSY·C0·C1·RESET·ONE

/BG1:=/BR1·BR3·BR2·BBSY·BG3·BG2·BG1·BG0·/C0·/C1·RESET·ONE
     +/BR1·BR2·BBSY·BG3·BG2·BG1·BG0·C0·/C1·RESET·ONE
     +/BR1·BBSY·BG3·BG2·BG1·BG0·/C0·C1·RESET·ONE
     +/BR1·BR0·BR3·BR2·BBSY·BG3·BG2·BG1·BG0·C0·C1·RESET·ONE
     +/BG1·BG3·BG2·BBSY·/C0·/C1·RESET·ONE
     +/BG1·BG2·BBSY·C0·/C1·RESET·ONE
     +/BG1·BBSY·/C0·C1·RESET·ONE
     +/BG1·BG0·BG3·BG2·BBSY·C0·C1·FESET·ONE

/BG0:=/BR0·BR3·BR2·BR1·BBSY·BG3·BG2·BG1·BG0·/C0·/C1·RESET
     +/BR0·BR2·BR1·BBSY·BG3·BG2·BG1·BG0·C0·/C1·RESET·ONE
     +/BR0·BR1·BBSY·BG3·BG2·BG1·BG0·/C0·C1·RESET·ONE
     +/BR0·BBSY·BG3·BG2·BG1·BG0·C0·C1·RESET·ONE
     +/BG0·BG3·BG2·BG1·BBSY·/C0·/C1·RESET·ONE
     +/BG0·BG2·BG1·BBSY·C0·/C1·RESET·ONE
     +/BG0·BG1·BBSY·/C0·C1·RESET·ONE
     +/BG0·BBSY·C0·C1·RESET·ONE

/BLOCK:=BG3·BG2·BG1·BG0
       +/RESET

/C0:=RR
    +/RR·BG3·BG2·BG1·BG0·BLOCK·C0
    +/RR·/C0·/BG3
    +/RR·/C0·/BG2
    +/RR·/C0·/BG1
    +/RR·/C0·/BG0
    +/RESET

/C1:=RR
    +·RR·BG3·BG2·BG1·BG0·BLOCK·C1·C0
    +·RR·BG3·BG2·BG1·BG0·BLOCK·/C1·/C0
    +/RR·/C1·/BG3
    +/RR·/C1·/BG2
    +/RR·/C1·/BG1
    +/RR·/C1·/BG0
    +/RESET

IF (VCC)/BCLR:=/BBSY·/BG0·/BR1·RR·ONE·RESET
              +/BBSY·/BG0·/BR2·RR·ONE·RESET
              +/BBSY·/BG0·/BR3·RR·ONE·RESET
              +/BBSY·/BG1·/BR2·RR·ONE·RESET
              +/BBSY·/BG1·/BR3·RR·ONE·RESET
              +/BBSY·/BG2·/BR3·RR·ONE·RESET
IF (VCC)/EN:=BBSY·BLOCK
            +/ONE
```

при одновременном приходе в арбитр сигналов BRXL и BRXR одного уровня их приоритет не должен быть одинаковым. В функциональном плане сигналы BRXL и BRXR можно объединить элементом ИЛИ и в качестве сигнала BRX подавать на вход арбитра, выполненного в соответствии

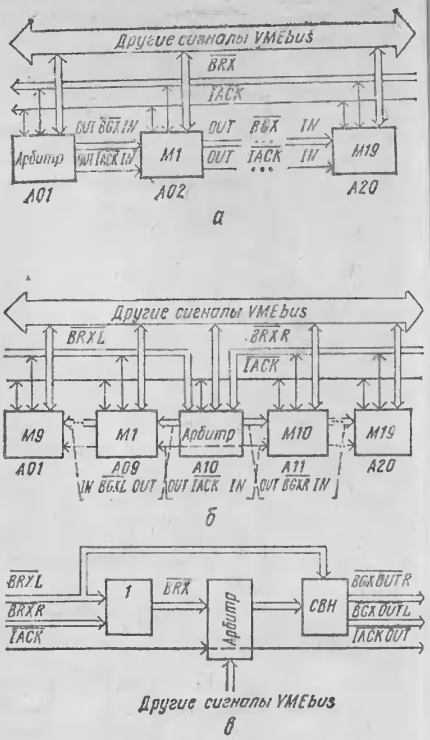


Рис. 3. Способ снижения времени доступа к интерфейсу:

а — система с оконечным расположением арбитра (прототип); *б* — система с центральным расположением арбитра; *в* — функциональная схема арбитра при его подключении в центре системы

с табл. 2, который для этих целей следует дополнить на выходе схемой выбора направления (рис. 3, *в*). Эта схема предназначена для коммутации сигналов разрешения на правую или левую половину каркаса. Ее реализация в функциональном плане соответствует схеме выбора направления, имеющейся в любом устройстве Requester.

Формирование сигнала IACK OUT.

В зависимости от компоновки системы возможны две альтернативы. Если модули разных уровней прерывания заранее установить по разные стороны от арбитра, то контакты IACK и IACK OUT арбитра можно объединить (как и предусмотрено спецификацией) и вывести на контакты IACK IN модулей, находящихся на посадочных местах A09 и A11. Когда такое размещение модулей по каким-либо причинам представляется неудобным (например, при динамическом способе присвоения приоритетов), можно предложить реализацию арбитра, не связанного с этим ограничением. Для этого необходимо разделить сигналы IRQ1 ... IRQ7 на

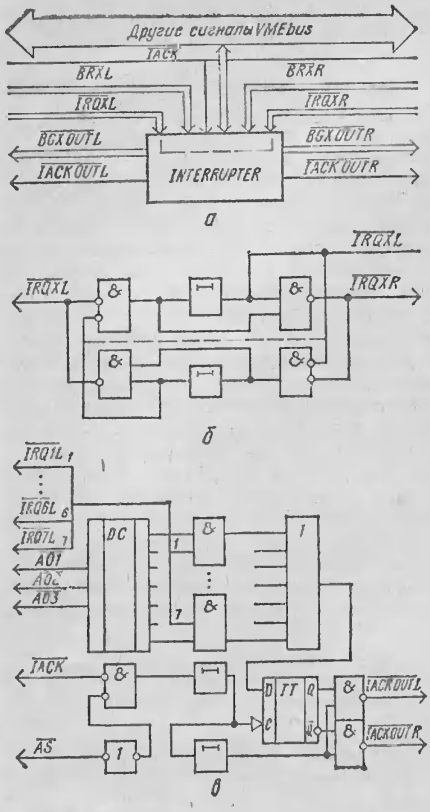


Рис. 4. Способ повышения реактивности системы без изменения протокола:

а — центральное подключение арбитра при произвольном размещении устройства Interrupter; *б* — функциональная схема ретрансляции двунаправленных сигналов BRX в устройстве Interrupter; *в* — функциональная схема устройства Interrupter

IRQ1L/R ... IRQ7L/R (рис. 4, *а*) также, как это было описано выше для сигналов BRX.

Передача сигналов запроса прерывания от правой группы модулей к левой и обратно производится в арбитре с помощью стандартной схемы ретрансляции двунаправленного сигнала, используемой в расширителях интерфейса (рис. 4, *б*). В работе модулей при этом ничего не меняется, и сигнал IRQX любого модуля остается доступным всем другим модулям. Лишь арбитру приходится выполнить дополнительную функцию по анализу направления выдачи сигналов IACK OUT L/R. В остальном арбитр работает аналогично схеме Interrupt Requester, которая принимает и дешифрирует код на линиях A03 ... A01, сравнивает с уровнем приоритета сигнала IRQXL/R, принимает сигнал IACK IN (в данном случае IACK) и по сигналу AS формирует один из сигналов IACK OUT L или IACK OUT R (рис. 4, *в*).

Улучшение протокола обмена.

Рассмотренные выше предложения находятся в полном соответствии со спецификацией на интерфейс VME, следовательно, нет препятствий для их применения. Возможны и другие пути улучшения характеристик арбитража, которые требуют усовершенствования протокола обмена. Одним из таких путей является следующий.

На характеристики системы большое влияние оказывает время не только установки сигналов по цепочечной линии, но и их сброса. Устройство, выигравшее арбитраж и занявшее шину, не сбросит сигнал BBSY, а следовательно, и не разрешит арбитру проведения следующего цикла арбитража до тех пор, пока не получит сброс сигнала BGXIN на своем входе (рис. 5, *а*).

В редакции Revision C спецификации интерфейса VME уже предусмотрены средства для исключения влияния времени распространения сброса сигнала IACK IN/OUT на реактивность системы прерываний. Согласно этой редакции все устройства Interrupt Requester, проигравшие арбитраж, должны одновременно запретить трансляцию (осуществить сброс) сигнала IACK OUT (если они его выставляли) при появлении на шине активного уровня сигнала DTACK от устройства, выигравшего арбитраж.

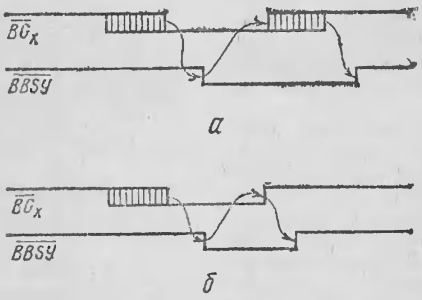


Рис. 5. Временная диаграмма арбитража:

а — без изменения протокола (прототип); *б* — с улучшением протокола

Аналогичный подход исключения влияния времени сброса сигналов на время арбитража можно предложить также для устройств Requester (с жестким требованием к времени доступа к шине), подключенных к уровню BRX (рис. 5, *б*). В этом случае арбитр и все устройства DTB Requester должны при появлении на шине активного уровня сигнала BBSY, минимальная длительность которого в соответствии с протоколом VME может составить 90 нс (спецификация Rev. C), одновременно запретить трансляцию (осуществить сброс) активного уровня сигнала BGXOUT.

Предлагаемое дополнительное требование к арбитру, заключающееся в том, что он должен суметь зарегистрировать импульс в 90 нс, не является ограничением. Это видно, например, из того, что при последовательном доступе (ascending access) потенциальные исполнители должны отслеживать отсутствие активного уровня сигнала на линиях DS0, DS1 и AS, минимальная длительность которых 40 нс.

Особенность комплексирования системы. Если все устройства DTB Requester системы выполнены так, что в них используется сигнал SYSCLK (см. рис. 1), то оказывается возможным произвольное сочетание и расположение в системе как модулей, подчиняющихся спецификации Rev. в [5], так и модулей, разработанных в соответствии с предложенной доработкой. В этом случае будет иметь место полная аппаратная совместимость с модулями зарубежных фирм. Если же не во всех устройствах DTB Requester используется сигнал по линии SYSCLR, то аппаратная совместимость не гарантируется.

Справки по телефону: 238-41-11, Москва.

ЛИТЕРАТУРА

1. Филин А. В. Об одном классе малых ЭВМ с одномагистральной структурой системного интерфейса ввода-вывода: Сб. статей: Структурная организация процессоров и вычислительных комплексов СМ ЭВМ / Под ред. А. Н. Кабалевского.— М.: ИНЭУМ, 1979.— Вып. 77.— С. 3—14.

2. Рыбаков В. Г. Современные магистрально-модульные системы. Характеристики и тенденции развития (обзор) // Приборы и техника эксперимента.—1983.— № 6.— С. 7—28.

3. Интерлаб. Микрокомпьютерная система. — София: БАН-ЦЛАН, 1984.—68 с.

4. Peterson Wade. VME Bus Requester Releases Bus Four Ways.— EDN. — 1985. — Vol. 30. — August 8.— P. 241—242.

5. Шина VME. Руководство по эксплуатации. Пер. № СР-82290.— М.: ВЦП, 1983.—239 с.

Статья поступила 6 июня 1986 г.

УДК 681.3.022

В. М. Брябрин, Б. Т. Сираджов

ПАКЕТ ДЕМОНСТРАЦИОННОЙ ГРАФИКИ АЛЬФА-ГРАФ ДЛЯ ПЭВМ ТИПА ЕС-1841

Пакет демонстрационной графики АЛЬФА-ГРАФ предназначен для применения в задачах обработки информации, требующих подготовки и использования изображений. Его программы обеспечивают создание и редактирование изображений, формирование текстовых фрагментов на фоне графических изображений, демонстрацию изображений на экране дисплея и вывод их на устройства твердой копии.

Изображения создаются выводом на экран и трансформацией элементарных геометрических фигур, формированием из отрезков прямых (например, методом резиновой нити или экскизированием) и динамическим перемещением объектов из одной позиции в другую.

По форматам внутреннего представления изображений можно выделить векторное и растровое представления. В первом случае изображения строятся на основе композиции примитивных элементов, задаваемых координатами опорных и характеристических точек, а во втором — на основе точечных элементов изображения.

Архитектура графического процессора

Графический процессор — это программное средство, получающее на входе числовую, графическую, а также текстовую информацию. После их обработки на выходе создается графическое изображение, которое можно вывести на экран дисплея, печатающее устройство и графопостроитель (рис. 1).

Структура векторного изображения. Векторное изображение во внутреннем представлении разбивается на несколько иерархических уровней. Графический процессор использует три уровня (рис. 2).

На нижнем уровне находятся так называемые примитивы вывода. Набор примитивов включает в себя следующее: прямая, треугольник, прямоугольник, окружность (эллипс, дуга), ломаная, палитра с образцами красок, символы русского и латинского алфавитов.

На следующей ступени иерархии находятся сегменты. Сегмент — это группа примитивов, объединенных в единое целое. К сегментам так же, как и к примитивам, применяются различные операции преобразования.

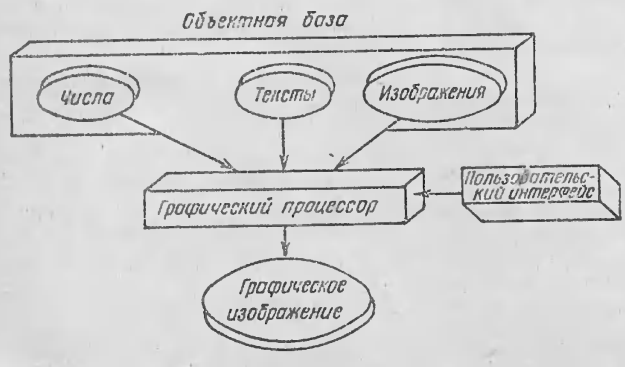


Рис. 1. Графический процессор

Наконец, третий уровень — это собственно изображение, которое также подвергается преобразованиям.

Примитивы. Сразу после вывода на экран образца примитива инициализируется процедура его обработки, с помощью которой непосредственно изменяются параметры примитива. Образцы примитивов выводятся на экран с помощью функции ВЫВЕСТИ ПРИМИТИВ в указываемую курсором точку. В дальнейшем эта точка отражается во внутренней структуре примитива в качестве параметра, идентифицирующего данный конкретный примитив (точка привязки).

Примитивы отличаются друг от друга разными наборами параметров. Каждый тип примитива (рис. 3) имеет свой код. Следует различать код примитива и точку привязки. Коды примитивов фактически являются кодами команд, которые воспроизводят эти примитивы на экране, а точки привязки служат для их выбора.

Преобразование примитивов. Для преобразования примитивов используется функция ПРЕОБРАЗОВАТЕЛЬ ПРИМИТИВА. В зависимости от типа примитива одни и те же клавиши воздействуют на его изменение по-разному (рис. 4). Клавиши «Зафиксировать» и «Отменить» (рис. 5) служат соответственно для фиксации изменения параметров и отмены изменений.

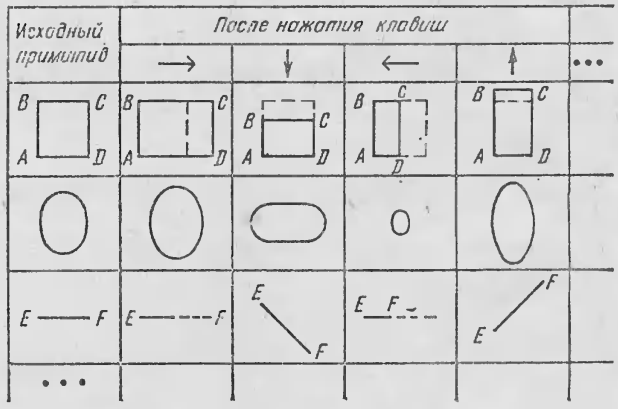


Рис. 4. Преобразование примитивов:

A, B, C, D — вершины прямоугольника; E, F — вершины прямой; стрелки — обозначения клавиш управления курсором

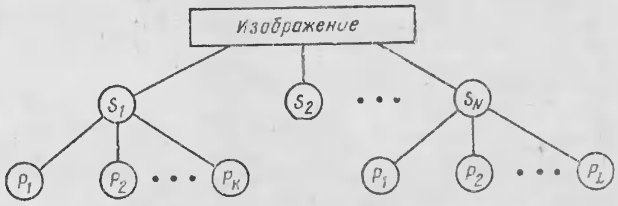


Рис. 2. Иерархическая структура векторного изображения:

S_1, S_2, ..., S_N — сегменты; P_1, P_2 ..., P_K — примитивы; N — число сегментов; K, L — число примитивов

Рис. 3. Структура примитива:

CD — код; X, Y — координаты точки привязки; CLR — цвет; EOP — ограничитель; ... — другие параметры

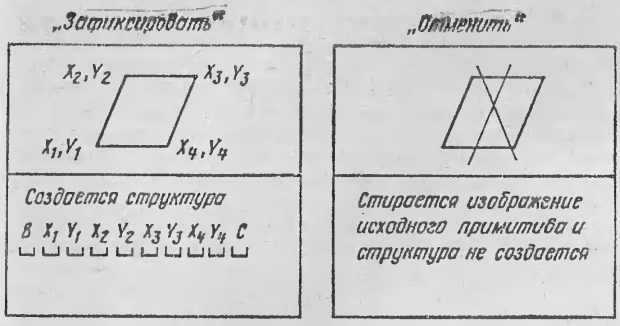


Рис. 5. Действие клавиш «Зафиксировать» и «Отменить»:
X_1, Y_1, X_2, Y_2, X_3, Y_3, X_4, Y_4 — координаты вершин примитива; B — код примитива; C — цвет примитива

Наложение текста на изображение. Графический процессор позволяет накладывать на изображение текстовую информацию. Это достигается благодаря наличию библиотеки, содержащей в себе изображения символов русского и латинского алфавитов. Все эти символы сохраняются в виде совокупности координат вершин ломаной, что дает возможность задавать шрифты любой высоты, ширины, цвета и выводить их в любую точку экрана (рис. 6).

Символы алфавитов выводятся на экран функцией ВЫВЕСТИ ПРИМИТИВ в указываемую курсором точку экрана. Последовательность символов образует сегмент. Таким образом, существуют «текстовые» и «нетекстовые» сегменты.

Сегмент образуется после вывода и преобразования любого числа примитивов с помощью функции СОЗДАТЬ СЕГМЕНТ. Формирование сегмента влечет образование локальной координатной системы, в которой определяются все параметры сегмента. Текущая позиция курсора становится центром локальной координатной системы и в том числе служит для идентификации сегмента (точка привязки). Координаты этой точки, в свою очередь, отсчитываются от начальной точки координатной системы экрана, т. е. центра экрана (рис. 7).

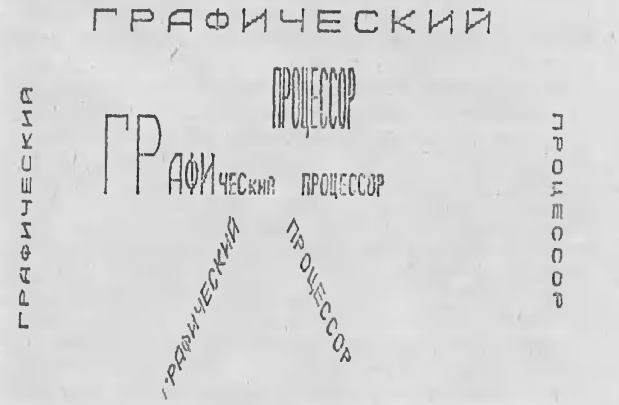


Рис. 6. Использование шрифтов

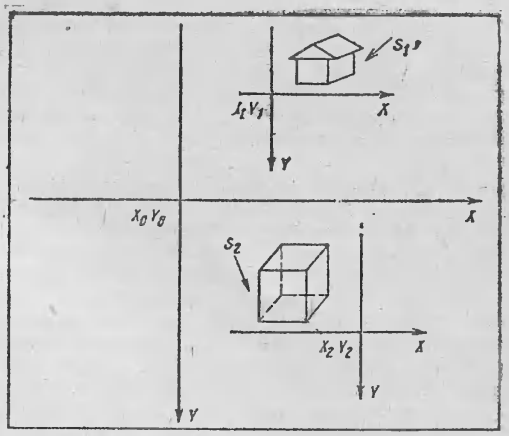


Рис. 7. Координатная система экрана и сегментов:
X_0, Y_0 — центр координатной системы экрана; X_1,Y_1 и X_2, Y_2 — центры локальных координатных систем сегментов S_0 и S_2

Функция СОЗДАТЬ СЕГМЕНТ образует структуру сегмента (рис. 8) и заполняет ее конкретными значениями. Примитивы и сегменты в дальнейшем будем называть графическими объектами, или просто объектами.

X	Y	SX	SY	ANG	FLG	P_1	P_2	...	P_N	EOS

Рис. 8. Структура сегмента:
X и Y — координаты точки привязки сегмента; SX и SY — масштабные коэффициенты; ANG — угол поворота; FLG — параметр, помечающий текстовые сегменты; P_1, P_2, ..., P_N — примитивы, входящие в состав сегмента; EOS — ограничитель сегмента; N — число примитивов

Преобразование сегментов. К графическим объектам применяются такие специфические операции, как «Взять объект» и «Отпустить объект». Первая из них означает, что после указания и выделения необходимого объекта пользователь намерен выбрать его и осуществить преобразования над ним, а вторая, наоборот, означает отказ пользователя от выбора данного объекта или преобразований над уже выбранным объектом. Функция указания, выделения и выбора реализуется с помощью логического устройства выбора. Суть его работы (рис. 9) заключаются в следующем. При обращении к функции происходит вычисление расстояний от текущей позиции курсора до точек привязок всех объектов и находится минимальное из них; соответствующий объект начинает после этого мерцать на экране.

Далее пользователь может произвести следующие операции: выбрать текущий объект («взять объект»), для чего нужно нажать соответствующую клавишу; про-

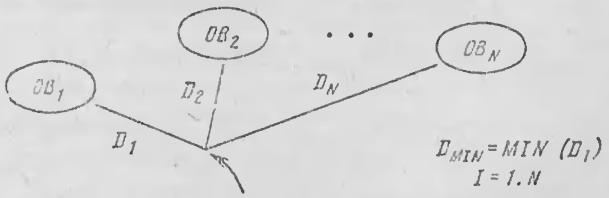


Рис. 9. Работа устройства выбора:
OB_1, OB_2 ..., OB_N — объекты; D_1, D_2 ..., D_N — расстояние от текущей позиции курсора до объектов; D_{min} — минимальное из этих расстояний; N — число объектов

должить перебор объектов, перемещаясь последовательно по структуре изображения; отказаться от выбора («отпустить объект»). Эти операции выполняются с помощью функции ИСКАТЬ ОБЪЕКТ.

Функции преобразования графических объектов:

ПЕРЕМЕСТИТЬ ОБЪЕКТ. Выбранный объект перемещается вслед за движением курсора;

МАСШТАБИРОВАТЬ ОБЪЕКТ. Масштабирование производится по вертикали и/или по горизонтали после ввода пользователем масштабных коэффициентов;

ПОВЕРНУТЬ ОБЪЕКТ. Поворот осуществляется относительно точки привязки после задания пользователем значения угла поворота в градусах;

СКОПИРОВАТЬ СЕГМЕНТ. Создание копии сегмента предполагает образование идентичного сегмента с одинаковыми параметрами;

ЗЕРКАЛЬНО ОТОБРАЗИТЬ СЕГМЕНТ. Зеркальное отображение сегмента производится относительно осей X и Y;

УДАЛИТЬ СЕГМЕНТ. При удалении сегмента в специальный системный массив заносится информация об удалении сегмента;

ВОССТАНОВИТЬ СЕГМЕНТ. Последний удаленный сегмент может быть восстановлен;

ОБЪЕДИНИТЬ ДВА СЕГМЕНТА. Под объединением двух сегментов понимается включение примитивов, входящих в один сегмент, в состав другого сегмента;

РАЗДЕЛИТЬ СЕГМЕНТ. При разделении сегмента на два пользователю необходимо указать границу разделения, другими словами, указать примитив, который будет последним в первом из получаемых сегментов;

ИЗМЕНИТЬ ЦВЕТ СЕГМЕНТА. При изменении цвета сегмента изменяются цвета всех входящих в него примитивов;

ИЗМЕНИТЬ ПОСЛЕДОВАТЕЛЬНОСТЬ ВЫВОДА СЕГМЕНТОВ — позволяет изменить последовательность предъявления сегментов;

СОХРАНИТЬ СЕГМЕНТ — обеспечивает сохранение сегмента в отдельном файле на внешнем носителе;

ОТПУСТИТЬ СЕГМЕНТ — позволяет отказаться от намерения модифицировать сегмент.

Изображение. С точки зрения представления данных структура векторного изображения — одномерный линейный массив (рис. 10).

Преобразование изображения. К изображению могут быть применены функции масштабирования, поворота, перемещения и объединения и др. Перемещение изображения производится в виде прокрутки по вертикали и горизонтали. Операции прокрутки выполняются функциями:

ПРОКРУТИТЬ ПО ВЕРТИКАЛИ и ПРОКРУТИТЬ ПО ГОРИЗОНТАЛИ. В качестве аргументов этим функциям передаются параметры, определяющие направление и шаг прокрутки;

ОБЪЕДИНИТЬ ИЗОБРАЖЕНИЯ. Функция позволяет объединить в пределах одного изображения два;

ВЗЯТЬ ПОСЛЕДНИЙ ОБЪЕКТ. Функция дает возможность пользователю редактировать последний выведенный объект.

Структура растрового изображения — это массив точечных элементов изображения (точек растра). По сравнению с векторным изображением, где преобразованию подвергаются структуры (примитивы, сегменты, изображение), в случае создания растрового изображения можно манипулировать только выделенными прямоугольными частями изображения. К таким частям применяются так называемые «электронные» способы преобразования. Например, для перемещения или копирования части изображения нужно «вырезать» эту часть и «склеить» с другой частью.

Преобразование растрового изображения. Функции, применяемые при создании и редактировании растрового изображения:

ВЫДЕЛИТЬ ЧАСТЬ ИЗОБРАЖЕНИЯ — выделяет прямоугольную часть изображения и запоминает ее в буфере;

ПЕРЕМЕСТИТЬ ЧАСТЬ ИЗОБРАЖЕНИЯ — позволяет перемещать содержимое буфера по экрану вслед за движением курсора;

СКОПИРОВАТЬ ЧАСТЬ ИЗОБРАЖЕНИЯ — копирует содержимое буфера в произвольное место экрана;

СВОБОДНОЕ РИСОВАНИЕ — позволяет выводить на экран линию определенной толщины и цвета и перемещать ее по экрану.

Интерфейс с пользователем

Способы организации диалога. Интерфейс с пользователем строится на основе двух методов: меню и клавишно-командного управления. Меню задается в виде перечня допустимых команд, которые изображаются либо в текстовом виде, либо с помощью пиктограмм. При клавишно-командном способе организации диалога каждая команда инициируется при нажатии определенной клавиши на клавиатуру.

Каждый из этих способов имеет свои достоинства и недостатки. Преимущества меню: все команды находятся «на виду» у пользователя и для выполнения какого-либо из них достаточно подвести курсор к соответствующей позиции и нажать клавишу «Исполнить». Основные недостатки меню: часть рабочей поверхности экрана теряется для создания изображения; многочисленные перемещения по экрану для выбора команд иногда затрудняют пользователей.

Преимущества клавишно-командного способа: исключаются затраты на организацию меню, вся поверхность экрана используется для формирования изображения, а количество перемещений по экрану становится минимальным. Однако пользователю приходится помнить все соответствия между клавишами и командами в различных состояниях программы. Для частичного устранения этого недостатка применяется высвечивание на экране подсказки, где отражаются соответствия клавиш и команд.

Использование поверхности экрана. При любом способе организации диалога предпочтительно разделить поверхность экрана на четыре части: рабочую область для формирования изображения, область меню, область сообщений и область отражения состояния программы. Рабочая область для формирования изображения занимает наибольшую часть экрана. Область меню и подсказок о соответствии клавиш и команд размещается в пределах одной строки, а область сообщений для выдачи информации об ошибках пользователя совмещается со строкой меню.

Отражение состояния программы осуществляется в небольшом окне в левом нижнем углу экрана (рис. 11). Меню пиктограмм, изображенное в нижней части экра-

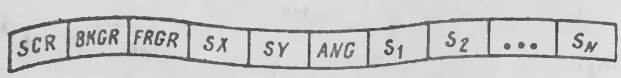


Рис. 10. Структура изображения:

SCR — указатель режима экрана; BKGR — цвет фона; FRGR — цвет палитры; SX и SY — масштабные коэффициенты; ANG — угол поворота; S_1, S_2, ..., S_N — сегменты

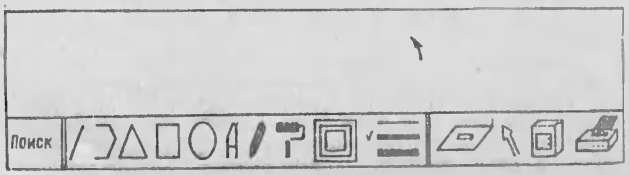


Рис. 11. Использование поверхности экрана

па, включает в себя условные изображения графических примитивов и инструментов: прямой, ломаной, треугольника, прямоугольника, эллипса, алфавита, карандаша, валика для покраски, палитры с красками, обозначений толщины линии. Здесь же даны условные обозначения основных операций. Пиктограмма дискеты соответствует группе операций ввода-вывода изображений. Стрелка обозначает переход к операциям указания и выбора объектов. Изображение дисплея служит для инициации команд установки цветовых параметров экрана и перехода в режим демонстрации слайдов. Пиктограмма печатающего устройства инициирует команды вывода на устройства твердой копии.

Взаимодействие с объектной базой

Пакет АЛЬФА-ГРАФ представляет собой диалоговый процессор, который может использоваться автономно или в составе интегрированной системы АЛЬФА. В последнем случае изображения, создаваемые с помощью АЛЬФА-ГРАФ, заносятся в объектную базу системы АЛЬФА. Объектная база — это среда и программные средства для хранения информационных объектов и доступа к ним. Под информационным объектом понимается структура данных, отображающая реальный физический объект, процесс, явление.

Взаимодействие с объектной базой охватывает функции сохранения, загрузки, удаления, объединения изображений, а также просмотра каталога объектов в базе.

Изображение сохраняется в объектной базе в трех форматах. Эти форматы определяются исходя из целей дальнейшего использования изображения: в растровом формате (для вывода только на экран); в векторном формате (для вывода на экран и графопостроитель), в формате внутреннего представления печатающего устройства.

Телефон для справок: 135-13-40, г. Москва

Статья поступила 12 июня 1986 года

УДК 681.3.022

И. А. Большаков

«ЧЕРЕПАШЬЯ» ГРАФИКА НА ДВК

Введение

В ряде прикладных задач нужно создавать на экране контурные изображения, состоящие из прямолинейных отрезков и, возможно, огрубленных дуг окружностей, а фон сохранять в неприкосновенности. Иными словами, обновлению при генерации изображения подлежит лишь небольшое число из 24×80=1920 ячеек экрана, причем эти ячейки могут быть объединены в последовательности, обладающие свойством «квазинепрерывности». Полное обновление содержимого экрана потребовало бы существенно бо́льших временных затрат, чем обход обновляемого контура, даже если последний имеет самопересечения и разрывы.

В сформулированных условиях полезно привлечь алгоритмические приемы, известные в зарубежной литературе под именем «черепашьей» графики (Turtle graphics). Воспроизводящее изображение «устройство» (курсор, перо графопостроителя) представляется черепашкой, ползущей по полю и оставляющей на нем следы. Для управления ею достаточно всего двух видов элементарных приказов*: ШАГНУТЬ В НАПРАВЛЕНИИ (А), где А — угол, измеренный в градусах или как-то по-иному, и ОТМЕТИТЬ ТОЧКУ, т. е. поставить помету в месте текущего расположения черепашки. Например, изображение прямолинейного отрезка можно получить серией пар (шаг, отметка), где все шаги делаются в одном направлении. Можно и продвигаться без отметок, формируя, где надо, контуры с разрывами.

Известен алгоритмический язык Лого [2], прочно опирающийся на данную идею и особо пригодный для обучения программированию новичков. В тех же западных ПЭВМ, где Лого не реализован, аналогичные «черепашьи» алгоритмы воплощаются в виде библиотек программ на Бейсике, Паскале и др. языках.

В статье описана иерархическая совокупность подпрограмм «черепашьей» графики, которая рассчитана на входящий в ДВК дисплей 15ИЭ-00—013 и воплощена на Паскале в среде ОС ДВК (РАФОС). Рассматривается построение как статических, так и динамически меняющихся («анимизированных») изображений на экране.

Создатели современных электронных игр для ДВК подчас преодолевают существенно бо́льшие алгоритмические трудности, поддерживая в реальном времени смены изображения сразу в нескольких частях экрана. Для этого требуется особая техника, к черепашьей графике в чистом виде не сводимая. В статье лишь фиксируется и тем самым популяризируется то, что известно пока немногим, и притом делается это целиком средствами известного языка высокого уровня.

Базисные элементы графики

Введем на экране восемь направлений:

TYPE DIRECTION=0 .. 7;

где направление 0 смотрит строго вправо (на восток), а прочие делят круг на 8 частей, располагаясь по часовой стрелке: 1 — вправо вниз (на юго-восток), 2 — вниз (на юг), 3 — влево вниз (на юго-запад) и т. д. Элементарное перемещение влево-вправо совершается вдоль строки экрана на одно знакоместо, вниз-вверх — вдоль столбца на одну строку, диагональные же перемещения подразумевают одновременное изменение на ±1 номер строки и столбца. Поскольку расстояние между строками примерно вдвое больше расстояния между столбцами, диагональные направления отнюдь не являются биссектрисами углов прямоугольной сетки.

Перемещения курсора и прочие действия на экране осуществляются с помощью управляющих последовательностей, начинающихся символом ESC=CHR (27) (в русской нотации кодов КОИ — это символ АР2).

Все «рисунки» на экране дисплея воспроизводятся только с помощью оператора WRITE, аргументами которого берутся собственно управляющие символы и литеры латинского алфавита, в рамках управляющих последовательностей действующие тоже как управляющие. Заранее заполним 4-символьный строковый массив SYMSTR:'CBDA', элементы которого связаны с перемещениями курсора в направлениях 0, 2, 4, 6.

В качестве базисных и вспомогательных графических подпрограмм в первую очередь необходимы:

Базисные и вспомогательные подпрограммы

Название	Аргументы оператора WRITE	Производимые операции
MARK	SYMB, ESC, 'D'	Вывод символа SYMB в текущую точку экрана при сохранении местоположения курсора

* Чаще при описании «черепашьей» графики пользуются другими, менее элементарными приказами, но функционально они эквивалентны приведенным ниже.

Окончание

Название	Аргументы оператора	Производимые операции
STEP	ESC, SYMSTR [DIR DIV 2+1], а при нечетном направлении DIR — дополнительно ESC, SYMSTR [(DIR+1) MOD 8 DIV 2+1]	Шаг курсора в направлении DIR
CLEAR	ESC, 'H', ESC, 'J'	Очистка экрана и отвод курсора в левый верхний угол
LOCATE	ESC, 'Y', CHR (31+Y), CHR (31+X)	Абсолютная адресация курсора в точку (X, Y), где X∈ (1 ... 80), Y∈ (1 ... 24) — координаты столбца и строки экрана

Подпрограммы STEP и MARK — это как раз ШАГНУТЬ и ОТМЕТИТЬ, т. е. примитивы черепашки. У MARK есть аргумент — конкретный символ SYMB, проставляемый в качестве пометы. Он может быть и пробельным.

Вывод произвольного символа в текущую позицию экрана автоматически сопровождается перемещением курсора вправо. Чтобы это скомпенсировать, подпрограмма MARK и выводит последовательность ESC, 'D', возвращающую курсор на прежнее место.

Подпрограммы, непосредственно надстроенные над базисными

В первую очередь нужна подпрограмма PUTSYMB (SYMB : CHAR; DIR : DIRECTION), включающая пару операторов MARK (SYMB); STEP (DIR) или в обратном порядке. Тогда прямолинейный в направлении DIR : DIRECTION заполненный символами SYMB : CHAR отрезок длиной LEN : INTEGER проводится подпрограммой SEGMENT с телом

```
FOR I:=1 TO LEN DO PUTSYMB (SYMB, DIR);
```

Из прямолинейных отрезков можно построить многие изображения, начинающиеся направлением DIR0:

1. Треугольник со сторонами SIZE, SIZE, 2*SIZE (допустимы DIR0=3 и 5);

```
PROCEDURE TRIAN (SYMB : CHAR;
DIR0 : DIRECTION; SIZE : INTEGER);
BEGIN
  SEGMENT (SYMB, DIR0, SIZE);
  SEGMENT (SYMB, (DIR0+3) MOD 8, 2*SIZE);
  SEGMENT (SYMB, (DIR0+6) MOD 8, SIZE);
END;
```

2. Прямоугольное «окно» или параллелепипед размером WIDTH на HIGTH:

```
PROCEDURE WINDOW (SYMB : CHAR;
                  DIR0 : DIRECTION;
          WIDTH, HIGHT : INTEGER);
BEGIN
  SEGMENT (SYMB, DIR0, WIDTH);
  SEGMENT (SYMB, (DIR0+2) MOD 8, HIGHT);
  SEGMENT (SYMB, (DIR0+4) MOD 8, WIDTH);
  SEGMENT (SYMB, (DIR0+6) MOD 8, HIGTH);
END;
```

3. Меандр (цикл ломаной «волны») размером SIZE на SIZE:

```
PROCEDURE MEANDR (SYMB : CHAR;
                  DIR0 : DIRECTION;
                  SIZE : INTEGER);
BEGIN
  SEGMENT (SYMB, DIR0, SIZE);
  SEGMENT (SYMB, (DIR0+6) MOD 8, SIZE);
  SEGMENT (SYMB, DIR0, SIZE);
  SEGMENT (SYMB, (DIR0+2) MOD 8, SIZE);
END;
```

4. Прямоугольник (для блок-схем) размером WIDTH+2 на HIGHT+2:

```
PROCEDURE BLOCK (WIDTH, HIGTH : INTEGER);
BEGIN
  PUTSYMB ('+', 0);
  SEGMENT ('—', 0, WIDTH);
  PUTSYMB ('+', 2);
  SEGMENT ('I', 2, HIGHT);
  PUTSYMB ('+', 4);
  SEGMENT ('—', 4, WIDTH);
  PUTSYMB ('+', 6);
  SEGMENT ('I', 6, HIGHT);
END;
```

Может также оказаться полезным изображение «шарика» (минимальной округлой фигуры). Она формируется циклом

```
FOR DIR:=0 TO 7 DO
  IF DIR MOD 4 <> 2 THEN PUTSYMB ('*', DIR);
```

«Окружности» более крупных размеров (с радиусом 5 ... 11 строк) хорошо строятся алгоритмом, близким к алгоритму Брезенхэма [1]. В преломлении к «черепашьей» графике идея построения такова. Выбирается заведомо правильная точка на окружности, например ее верхняя точка (X_0, Y_0—R), где X_0, Y_0 — целочисленные координаты центра, R — радиус. Начальным направлением движения DIR=0 задается построение окружности по часовой стрелке. «Черепашка» пробует три направления — текущее и два отличающихся от него на ±1. Сравнивается, при каком направлении она наименьшим образом отклонится от окружности: $[(X-X_0)/2]^2+[(Y-Y_0)/2]^2=R^2$ (деление на 2 призвано приблизить фигуру на экране к окружности). Перемещение производится по направлению с наименьшим отклонением, после чего пробы и шаги повторяются, и так до возврата в исходную точку.

Динамически меняющиеся изображения

Можно полагать, что приемы «анимизации» изображений окажутся полезными не только в электронных играх, где они являются основой, но и для массы иных приложений. Опишем некоторые из этих приемов.

Организация мерцания участка текста. Дисплей 15ИЭ-00-013 сам по себе, внепрограммно, не имеет возможности организовать мерцание участка изображения на экране, но этого можно добиться программой. Поскольку время процессора ДВК делить в однозадачном режиме не с чем, откажемся от использования таймера, а временны́е задержки, когда они нужны, будем организовывать пустыми циклами. Дополнительно предположим (и это, к сожалению, ограничение Паскаля), что длина мерцающей подстроки нам известна заранее. Тогда, например, мерцание в течение нескольких секунд строки «ЧЕРЕПАШЬЯ ГРАФИКА» длиной 17 можно осуществить циклом

```
FOR J:=1 TO 50 DO
  BEGIN
  WRITE ('ЧЕРЕПАШЬЯ ГРАФИКА');
  FOR I:=1 TO DELAY DO;
  FOR I:=1 TO 17 DO STEP (4);
  FOR I:=1 TO 17 DO WRITE (BL);
  FOR I:=1 TO DELAY DO;
  FOR I:=1 TO 17 DO STEP (4);
  END;
```

Здесь BL — литера пробела, а DELAY≈800 — константа, которую можно подобрать для регулировки частоты (около 2 Гц) и продолжительности мерцаний.

Перемещение малого объекта по экрану. Малый объект (например, «шарик») можно перемещать по экрану следующим приемом:

рисуется объект с помощью непробельного символа;

он же рисуется с помощью пробельного символа (тем самым стирается);

перемещается на шаг в выбранном направлении;

рисование повторяется в указанной последовательности и т. д.

Вот как это выглядит программно в случае шарика, движущегося слева направо по экрану (BALL — подпрограмма рисования шарика):

```
CLEAR; LOCATE (14, 5);
FOR I := 5 TO 75 DO
  BEGIN
    BALL ('*'); BALL (BL); STEP (0);
  END;
```

Заметим, что стирание здесь обязательно, иначе изображение покажется «смазанным». Приведенный алгоритм не только не требует искусственных замедлений, но сам ведет к довольно медленному пересечению шариком экрана. Возможным объяснением тому является большое число независимых вызовов встроенного оператора WRITE. На построение, стирание и одношаговое перемещение шарика требуется 77 вызовов, на полное перемещение по экрану — около 5,5 тыс. вызовов.

«Заполнение сосуда жидкостью». В задачах деловой графики часто строятся линейные (т. е. горизонтальные) и столбчатые (т. е. вертикальные) диаграммы и гистограммы. Их можно строить, как чисто статические объекты, но впечатление значительно сильнее, когда они строятся динамически, с имитацией движения жидкости в сосуде.

Пусть идет речь о столбчатой диаграмме. Высоты отдельных столбиков перед ее построением известны. Восклицательными знаками или символом вертикальной черты быстро построим боковые стенки «сосуда» высотой с очередной столбик, а затем «нальем» в него «темную жидкость» следующим образом.

Среди литер дисплея 15ИЭ-00—013 имеются знаки подчеркивания, дефиса, надчеркивания CHR (126) и полного затемнения литерной матрицы CHR (127). С некоторым замедлением будем подменять очередную снизу пробельную позицию внутри «сосуда» последовательно на знаки подчеркивания, дефиса, надчеркивания и затемнения. После того как появился последний из знаков, сделаем шаг вверх и все повторим и т. д. У человека возникает полная иллюзия заполнения сосуда. Подпрограмма FILLING, выполняющая эту задачу:

```
PROCEDURE FILLING (HIGHT : INTEGER);
  VAR I, J, K : INTEGER;
BEGIN
  SEGMENT ('!', 6, HIGTH);
  STEP(0); STEP(1);
  SEGMENT ('!', 2, HIGTH);
  STEP(5);
  FOR I := 1 TO HIGTH DO
    BEGIN
      FOR J := TO 3 DO
        BEGIN
          MARK (FILCHARS [J]);
      FOR K : =TO DELAY DO;
        END;
      MARK (CHR(127));
      IF I < HIGHT THEN STEP(6);
    END;
  FOR I : HIGHT-1 DOWNTO 1 DO STEP(2);
  STEP(4);
END;
```

Здесь FILCHARS — строковый массив из трех упомянутых литер «—», «—», «—».

Арсенал средств черепашьей графики на ДВК можно существенно расширить: приблизить более сложные кривые или (с помощью литер вертикальной черты и затемнения) имитировать движение жидкости в горизонтальных «трубах»; с помощью символов FILCHARS приближенно отобразить еще четыре угловых направления, между 0 и 1, 3 и 4, 4 и 5, 7 и 0, а это позволит с лучшим качеством имитировать круговые диаграммы, изображать часовую стрелку, в том числе движущуюся, и др. Отражение траектории черепашки при ее косом столкновении с краями экрана позволяет создавать сложные самопересекающиеся орнаменты. Подобные приемы графики можно множить исходя из личного вкуса и потребностей.

Наконец, полезно отметить, что все предложенные средства и приемы без каких-либо изменений оказались пригодными и для дисплея «Электроника МС 6105», являющегося составной частью ДВК-3М2. Это и понятно, поскольку мы опирались не на одиночные управляющие символы, а на управляющие эскейп-последовательности, в обоих дисплеях действующие одинаково.

Телефон для справок: 155-45-15 г. Москва

ЛИТЕРАТУРА

1. Фоли Дж., вэн Дэм А. Основы интерактивной машинной графики. Кн. 2.— М.: Мир, 1985.— 368 с.
2. Хротко Г. Языки программирования высокого уровня для микроЭВМ.— М.: Изд. МЦНТИ, 1985.— 91 с.

Статья поступила 12 августа 1986 года

УДК 681.3.022

С. Н. Попов

ГРАФИЧЕСКИЙ РЕДАКТОР ДЛЯ ПЭВМ «МИКРОША»

Существенное отличие персональных ЭВМ от ЭВМ других типов — возможность гибкой программной перестройки характеристик внешних устройств под разнообразные требования пользователей, так как в ПЭВМ дисплей и клавиатура интегрированы с процессором и блоком оперативной памяти. Не является исключением из этого правила и одноплатная ПЭВМ «Микроша». Дисплейный контроллер ПЭВМ выполнен на базе БИС КР580ВГ75 и может быть программно настроен на различные форматы отображения информации на экране. Это свойство архитектуры ПЭВМ и использовано в описываемом ниже графическом редакторе.

В ПЭВМ «Микроша» дисплей в основном предназначен для отображения алфавитно-цифровой информации. Однако в знакогенераторе предусмотрен ряд специальных символов для формирования изображений в псевдографическом режиме.

Одним из интересных применений ПЭВМ является их использование в качестве устройства отображения информации в условиях аудитории или класса. Преподаватель заранее готовит необходимый иллюстративный материал (графики, чертежи, схемы, формулы и т. д.). Во время занятий подготовленные изображения быстро выводятся на экран телевизора. При этом время занятий используется более рационально.

Разработанный для ПЭВМ «Микроша» графический редактор в основном и предназначен для данной цели. Однако он может применяться для подготовки графи-

ческих данных и обучения детей составлению простейших программ.

Общие сведения о редакторе

Особенности аппаратурной реализации дисплейного контроллера «Микроша» позволяют получать на экране телевизора интересные эффекты, похожие на мультипликацию. Так как экранный буфер может находиться в любом месте ОЗУ ПЭВМ, то имеется возможность организовать в ОЗУ несколько экранных буферов и быстро отображать хранящуюся в них информацию на экране дисплея. Чтобы переключение буфера не приводило к срыву синхронизации изображения, оно производится во время обратного хода кадровой развертки.

При работе программы Монитор на экране дисплея отображается 25 строк алфавитно-цифровой информации по 64 символа в каждой. В псевдографическом режиме, который поддерживается интерпретатором языка Бейсик, возможно отображение 128 точек по горизонтали и 50 точек по вертикали. При работе с графическим редактором экран дисплея содержит 128 точек по горизонтали и 64 точки по вертикали. При этом вертикальные линии, так же как и горизонтальные, отображаются как непрерывные. Такой формат отображения достигается путем соответствующей настройки контроллера КР580ВГ75.

Конечно, по сравнению с растровыми графическими дисплеями указанное разрешение совсем невелико, но, как показал опыт, оно вполне достаточно для тех применений, о которых говорилось выше. Кроме того, «платой» за высокое разрешение, как правило, является необходимость использования ОЗУ большого объема и значительные временные затраты по обработке таких изображений.

Графический редактор позволяет организовать в памяти ПЭВМ 11 экранных буферов для хранения графической информации и один буфер для текстовой информации с форматом отображения 25 строк по 64 символа в каждой.

При работе редактора все буферы (табл. 1) имеют свой номер (от 0 до 11): текстовый — номер 0, а графические от 1 до 11. С учетом жестких ограничений на объем и на время работы программа «графический редактор» написана на языке ассемблера микропроцессора КР580ИК80А и имеет объем 5К байт. В ПЭВМ «Микроша» предусмотрена возможность подключения внешнего модуля ПЗУ объемом до 16К байт. В этом модуле помимо графического редактора могут одновременно находиться интерпретатор языка Бейсик и редактор текстов.

Работа с редактором заключается в следующем. После запуска на экране появляется меню директив (табл. 3). Выдав соответствующую директиву, пользователь может создавать новые изображения в ручном или автоматическом режиме, сохранять их на магнитной ленте, загружать с магнитной ленты ранее подготовленные изображения и вносить в них изменения, в автоматическом режиме выводить изображения на экран в произвольном порядке и сопровождать их различными звуковыми эффектами.

Редактор позволяет, кроме подготовки графических изображений, выводить на экран и алфавитно-цифровую информацию. Символы при этом формируются в режиме псевдографики с использованием матрицы размером 6×8 точек (используется тот же формат отображения и тот же набор символов, что и в ПЗУ знакогенерато-

Таблица 1

Распределение памяти ПЭВМ при работе с графическим редактором

ПЗУ монитора	F800H
Регистры периферийных БИС	C000H
Бейсик, редактор текста (ПЗУ)	9400H
Программа-редактор (ПЗУ)	8000H
Текстовый буфер (0)	76D0H
Рабочие ячейки монитора и редактора	7500H
Графический буфер (11)	
~~	
Графический буфер (1)	0000H

Таблица 2

Команды ручного режима

Клавиша	Команда	Примечание
←	Курсор влево	УС+H
→	Курсор вправо	УС+X
↑	Курсов вверх	УС+Y
↓	Курсор вниз	УС+Z
F2	Курсор вверх-вправо	УС+A
↖	Курсор вверх-влево	УС+L
F5	Курсор вниз-вправо	УС+D
F3	Курсор вниз-влево	УС+B
УС+P	«Прозрачный курсор»	
УС+K	Курсор — «карандаш»	
УС+L	Курсор — «ластик»	
УС+F	Пометить начало отрезка	
УС+S	Провести отрезок	
УС+M	Пометить начало блока	ВК
УС+J	Пометить конец блока	ПС
УС+N	Переслать блок в позицию курсора	
УС+I	Проинвертировать изображение в блоке	ТАБ
УС+I	Стереть блок	
УС+E	Снять определение блока	
УС+U		
УС+]	Сдвинуть экран вверх на 2 точки	
УС+T	Сдвинуть экран вниз на 2 точки	
УС+O	Сдвинуть экран влево на 2 точки	
УС+V	Сдвинуть экран вправо на 2 точки	
АР2	Включить текстовой экран	УС+[

ра дисплея). В этом случае на экран дисплея можно вывести до семи строк алфавитно-цифровой информации по 20 символов в каждой. Размер символов примерно в 3 раза больше, чем в стандартном режиме отображения, и они легко читаемы даже при большом удалении от экрана. Рассмотрим каждый из режимов работы редактора подробно.

Ручной режим создания изображений

После выдачи директивы PN (ручной режим) на экране дисплея отображается текущее содержимое буфера, номер которого определяет параметр N. В первоначальном запуске редактора проводится очистка всех буферов. В последующих запусках редактора содержимое буферов не стирается. При вводе директив и их параметров последний набранный символ может быть удален нажатием клавиши ← (стрелка влево). Ввод директивы заканчивается нажатием на клавишу ВК. Директивы, набранные с какими-либо ошибками, просто игнорируются, при этом выдается звуковой сигнал индикации ошибки (высокий тон, затем низкий).

В качестве графического курсора используется мигающий квадратик, равный по размерам одной отображаемой графической точке. С помощью команд, перечисленных в табл. 2, его можно перемещать в произвольное место экрана. Все команды выдаются путем нажатия и удержания клавиши УС и одной из указанных клавиш. Основные команды перемещения графического курсора выдаются простым нажатием на клавишу функциональной клавиатуры.

При перемещении курсора по экрану возможны три режима: 1) курсор перемещается по экрану, не изменяя его содержимого («прозрачный курсор»); 2) за курсором остается след в виде светлой линии (курсор — «карандаш»); 3) перемещение курсора приводит к стиранию точек на экране (курсор — «ластик»). Выбор одного из этих режимов осуществляется соответствующими командами (см. табл. 2).

Кроме получения изображений на экране с помощью курсора — «карандаша», оператор может формировать отрезки прямых линий следующим образом. Курсор помещается в то место экрана, где будет начало отображаемого отрезка и оператор выдает команду УС+F. Затем курсор помещается в то место, где будет конец отрезка (предварительно следует включить «прозрачный» курсор) и выдается команда УС+S. На экране сразу появляется отрезок прямой линии, соединяющий эти две точки. Установленные с помощью команды УС+F координаты начала отрезка сохранятся до того момента, когда эта команда будет выдана вновь.

При формировании изображений часто возникает необходимость переноса части изображения из одного места экрана в другое или стирания его. Для этого служат команды работы с блоком. Под блоком в данном случае понимается прямоугольная область на экране, заданная координатами левого нижнего и правого верхнего углов. В любой момент работы редактора может быть определен только один блок.

Для того чтобы пометить блок, оператор помещает курсор в его левый нижний угол и выдает команду УС+M, затем — в правый верхний угол блока и выдает команду УС+J. Чтобы получить копию блока в другом месте экрана, оператор должен переместить курсор в левый нижний угол перемещенного блока и выдать команду УС+C. Команда копирования сохраняет определение блока, поэтому его можно копировать многократно в разные места экрана. Если команда УС+J выдается до того, как была выдана команда УС+M или начало блока было помечено на другом экране, то она просто игнорируется.

Команда УС+U отменяет заданное ранее определение блока. Команда УС+E стирает все, что находится внутри помеченного блока. После ее выполнения определение блока снимается. Команда УС+I инвертирует изображение, находящееся внутри блока. Повторная выдача этой команды восстанавливает первоначальный контраст.

Манипуляции с блоком можно проводить не только в пределах одного экрана. Блок, помеченный на одном из экранов, можно скопировать на любом другом. Для этого после задания блока оператор меняет текущий экран и выдает директиву копирования. Нажатие на клавишу АР2 при работе в ручном режиме всегда приводит к выводу на экран меню директив (табл. 3).

Таблица 3

Формат текстового экрана

МИКРОША ГРАФИЧЕСКИЙ РЕДАКТОР ВЕРСИЯ 1.0 1987 г.
»ОСНОВНЫЕ ДИРЕКТИВЫ«

Р — РУЧНОЙ РЕЖИМ	У — СТРОКА СОСТОЯНИЯ (Д)
З — ЗАПИСЬ ЭКРАНА НА ЛЕНТУ	Ф — ФОРМИРОВАНИЕ МАКРОКОМАНДЫ
Ч — ЧТЕНИЕ ЭКРАНА С ЛЕНТЫ	Т — ЗАПИСЬ МАКРОКОМАНДЫ НА ЛЕНТУ
К — КОПИРОВАНИЕ ЭКРАНОВ	Л — ЧТЕНИЕ МАКРОКОМАНДЫ С ЛЕНТЫ
С — СТИРАНИЕ ЭКРАНА	Г — ВЫПОЛНЕНИЕ МАКРОКОМАНДЫ
ВАША КОМАНДА?	
ПОЛЕ ДЛЯ МАКРОКОМАНДЫ	

Используя команды, приведенные в табл. 2, оператор может сдвинуть содержание графического экрана вверх, вниз, влево или вправо на 2 точки. При этом часть сдвинутого изображения будет потеряна.

Для того чтобы оператор мог легко ориентироваться в установленных режимах работы, в самом низу экрана отображается так называемая строка состояния. Назначение отдельных полей строки следующее:

- К=Л — вид курсора (К — «карандаш», Л — «ластик», П — прозрачный»);
- Э=1 — номер текущего графического экрана;
- X=103 — позиция курсора по горизонтали;
- Y=20 — позиция курсора по вертикали;
- XL=99 — координата начала отрезка по горизонтали;
- YL=12 — координата начала отрезка по вертикали;
- B=1 — номер графического экрана, на котором помечен блок. Если номер равен 0, то блок не помечен;
- X1=10 — координата левого нижнего угла блока по горизонтали;
- Y1=11 — координата левого нижнего угла блока по вертикали;
- X2=30 — координата правого верхнего угла блока по горизонтали;
- Y2=20 — координата правого верхнего угла блока по вертикали.

Если координаты начала отрезка или блока не определены, то в соответствующих позициях строки состояния помещаются пробелы. Информация в строке состояния модифицируется при каждом изменении координат курсора на экране.

Пользуясь этой справочной информацией, можно легко определить, какие текущие режимы работы установлены, и осуществить «привязку» местоположения курсора к изображению. Переключив курсор в «прозрачный» режим и перемещая его по изображению, можно провести оцифровку изображения и использовать эти данные в прикладной программе.

Команда У (см. табл. 3) разрешает или запрещает отображение строки состояния. Первоначально отображение строки разрешено, о чем свидетельствует буква Д в круглых скобках. При каждой выдаче команды это условие заменяется на противоположное.

Если при работе в ручном режиме оператор нажмет одну из алфавитно-цифровых клавиш, то на экран дис-

плея будет выведено изображение соответствующего символа, а графический курсор автоматически переместится вправо на шесть позиций. Оператор может подкорректировать положение курсора, которое всегда определяет левый нижний угол выводимого символа. Таким образом, возможно формирование надстрочных и подстрочных символов.

Выход из ручного режима осуществляется нажатием на клавишу АР2. На экране при этом отображается первоначальное меню директив.

Работа с магнитофоном и другие директивы

Подготовленные в ручном режиме изображения могут быть сохранены на магнитной ленте и в дальнейшем вновь введены в память ПЭВМ. Для записи служит директива З. Ее формат ЗN1, N2. Если N2 не указан, то на ленте записывается только содержимое экрана с номером N1. В противном случае на ленту будет записано содержимое экранов с номерами от N1 до N2 включительно. Формат записи такой же, как и при работе с программой Монитор. Поэтому эта информация может быть введена в память ПЭВМ и при помощи директивы Монитора I.

Для чтения с магнитофона предусмотрена директива **Ч**, по которой информация вводится в то место памяти, откуда она была записана.

Для копирования содержимого одного экрана на другой служит директива **К**. Ее формат КN1, N2. Параметр N1 определяет экран-источник, а параметр N2 — экран-получатель информации. При копировании содержимое экрана-источника не изменяется.

Для стирания всего содержимого экрана служит директива **СN**. Параметр N определяет номер экрана.

Автоматический режим формирования изображений

Одна из особенностей описываемого редактора — возможность автоматического формирования изображений и управление последовательностью их отображения на экране дисплея. Это позволяет формировать динамически изменяющиеся изображения, похожие на мультипликацию. Для этого оператор должен подготовить специальную макрокоманду, описывающую требуемые действия.

Макрокоманда может вводиться как с клавиатуры, так и с магнитной ленты. На текстовом экране (поле макрокоманды) для нее отведены нижние 17 строк (см. табл. 3). Таким образом, максимальная длина макрокоманды 1088 символов (17×64=1088).

Ввод макрокоманды с клавиатуры дисплея начинается с выдачи директивы **Ф**. При этом текстовый курсор (мигающая черточка) помещается в левом верхнем углу поля макрокоманды. Оператор может поместить курсор в любое место поля макрокоманды с помощью команд, перечисленных в табл. 4. Клавиша **Зб** служит для стирания символа, стоящего слева от курсора. Вся информация в поле макрокоманды стирается путем двукратного нажатия на клавишу СТР.

При вводе макрокоманды клавишу ВК следует нажать только в самом конце, и на экране появится светлый прямоугольник — признак конца макрокоманды. Выполнение директивы **Ф** заканчивается нажатием на клавишу АР2. При последующем запуске макрокоманды на использование с помощью директивы **Г** следует предварительно убедиться, что признак конца макрокоманды установлен в соответствующем месте поля макрокоманды.

С помощью директивы **Т** макрокоманду можно записать на магнитную ленту, а директивы **Л** — считать с ленты в ОЗУ ПЭВМ. Формат записи также соответствует формату, используемому в программе Монитор. Следует учитывать, что так как текст макрокоманды хранится только в области текстового экрана, то при случайном нажатии на кнопку Сброс он будет утерян. Поэтому перед выходом в Монитор необходимо записать текст макрокоманды на магнитную ленту.

Таблица 4

Ввод и редактирование макрокоманды

Команда	Назначение	Примечание
←	Курсор влево на одну позицию	УС+H
→	Курсор вправо на одну позицию	УС+X
↑	Курсор вверх на одну позицию	УС+Y
↓	Курсор вниз на одну позицию	УС+Z
↖	Курсор в левый верхний угол поля	УС+L
ЗБ	Стереть символ слева от курсора	
СТР	Стереть всю макрокоманду	
АР2	Закончить ввод (редактирование)	

Макрокоманда состоит из последовательности операторов и операндов. Операнды разделяются с помощью символов «,» — запятая. Между отдельными операторами макрокоманды допускается помещать любое число пробелов. Между операторами и операндами не должно быть других символов.

Большинство действий по манипуляции с изображениями в ручном режиме доступны и в автоматическом режиме (табл. 5). Выполнение оператора **D** приводит к выдаче на экран содержимого указанного буфера. При этом он автоматически становится буфером для записи и считывания информации, т. е. становится текущим буфером. Чтобы готовить изображение в «невидимом» в данный момент буфере, необходимо назначить его буфером для записи (оператор **W**). Если после этого требуется проводить вывод изображений в отображаемый в данный момент буфер, необходимо вновь применить оператор **D**. В любом случае общее правило такое: модификация содержимого буфера возможна, только если он назначен как буфер для записи с помощью оператора **W**.

Оператором **P** меняется позиция графического курсора. Третий операнд этого оператора задает контраст для операторов **L** и **T**. Начальная точка для проведения отрезков прямых линий определяется текущей позицией графического курсора.

Наличие двух операторов цикла (**S** и **F**) позволяет организовать вложенные циклы. При этом допустимым являются вложения вида **S — F — O — Q** и **F — S — Q — O**. Вложения вида **S — F — Q — O** и **F — S — O — Q** недопустимы.

После запуска макрокоманды с помощью директивы **Г** из основного меню она начинает выполняться. Если обнаруживается какая-либо ошибка, то выполнение макрокоманды заканчивается и выдается специальный звуковой сигнал индикации ошибки (низкий тон, затем высокий тон). Если при выполнении макрокоманды нажата клавиша F4, то выполнение макрокоманды заканчивается досрочно.

Макрокоманды можно запустить и без выхода в основное меню. Для этого, находясь в режиме формирования макрокоманды, необходимо поместить текстовый курсор под тот оператор, начиная с которого будет выполняться макрокоманда, и нажать клавиши УС+G.

Таблица 5

Операторы макрокоманды

Оператор	Назначение	Операнды
DN	Назначить буфер для отображения	N — номер буфера
WN	Назначить буфер для записи	То же
RN	Назначить буфер для чтения	»
Y	Очистить текущий буфер	Нет
CN1, N2	Копирование содержимого буфера	N1 — источник, N2 — получатель
UN	Сдвинуть экран вверх на N×2 точки	N — число
NN	Сдвинуть экран вниз на N×2 точки	N — число
EN	Сдвинуть экран влево на N×2 точки	N — число
HN	Сдвинуть экран вправо на N×2 точки	N — число
PX, Y, Z	Высветить точку	X, Y — координаты, Z — контраст (0 или 1)
LX, Y	Провести отрезок	X, Y — координаты конца отрезка
BX1, Y1, X2, Y2	Определить блок в текущем буфере	X1, Y1 — левый нижний, X2, Y2 — правый верхний
I	Инвертировать изображение блока	Нет
J	Стереть блок	»
M	Переслать блок в позицию курсора	»
Z	Снять определение блока	»
SN	Начало цикла 1	N — число повторений
Q	Конец цикла 1	Нет
FN	Начало цикла 2	N — число повторений
O	Конец цикла 2	Нет
T «строка»	Выдача текста строки на экран	Строка в кавычках
K	Ожидание ввода любого символа с клавиатуры	Нет
AN1, N2	Выдача звукового сигнала	N1 — высота тона, N2 — длительность
XN	Временная задержка	N — число десятых долей секунд

Таким образом, в поле макрокоманды можно поместить несколько отдельных макрокоманд, которые запускаются при помощи описанных действий. Эту возможность используют и для отладки макрокоманд.

С помощью макрокоманд на экране дисплея можно получать различные интересные эффекты. Рассмотрим примеры:

D1KD2KD3KD4KD5KD6KD7KD8K

— макрокоманда последовательно выведет на экран содержимое первых 8 буферов. Вывод очередного буфера осуществляется простым нажатием на любую клавишу;

D1P20, 20, 1T «ВНИМАНИЕ!»B18, 18, 20, 28S4IG40, 40Q

— макрокоманда включает на отображение первый графический буфер, затем в центре экрана появится сообщение «ВНИМАНИЕ», а затем 4 раза проинвертирует его, сопровождая звуковым сигналом;

S50D1X2D2X2D3X2D4X2D5X2D6X2D7X2D8X2Q

— макрокоманда реализует режим мультипликации. В первых восьми буферах в ручном режиме подготовлены изображения, немного отличающиеся друг от друга («кадры фильма»). С задержкой в 0,2 с они последовательно выводятся на экран в цикле 50 раз.

Не вызывает сомнений, что пользователи могут найти и другие способы манипулирования изображениями с помощью макрокоманд.

Таким образом, описанный графический редактор позволяет создавать различные изображения с использованием графики низкого разрешения, манипулировать с ними в автоматическом режиме и сохранять результаты работы на магнитной ленте для последующего использования. Редактор имеет небольшой объем, размещается в ПЗУ и является компонентом системного программного обеспечения ПЭВМ «Микроша».

Для получения программы и дополнительной информации следует обращаться по телефону 235-83-82 (Москва, МИЭМ), понедельник — 10...14, суббота — 11...13.

Статья поступила 26 февраля 1987 г.

УДК 681.3.022

А. П. Кулаичев

ГРАФИЧЕСКИЙ АССИСТЕНТ ДЛЯ СИНТЕЗА, КАТАЛОГИЗАЦИИ И ЭКСПОНИРОВАНИЯ ЦВЕТНЫХ ИЗОБРАЖЕНИЙ

Синтезирование многоцветных графических изображений необходимо при решении задач в области технического конструирования средств отображения информации, эргономически оптимальных для человека-оператора при управлении сложными и динамичными объектами, где цветовое и графическое кодирование информации позволяет резко повысить надежность и эффективность работы системы; при исследовании и проектировании средств диалога в системах человек-машина, когда ищутся оптимальные формы взаимосвязи действий пользователя с ответами информационной системы; в работах по техническому дизайну и эстетике; при подготовке иллюстративного материала, который может предъявляться непосредственно на экране монитора или использоваться для изготовления слайдов.

Для решения указанных задач необходимо автоматизированное рабочее место конструктора на базе персонального компьютера с цветной графикой и соответствующим программным обеспечением, включающим

в себя удобный графический редактор и средства управления банком изображений. Приведенные соображения и отсутствие подходящих графических пакетов побудили к созданию программной системы ГРафический АССистент (ГРАСС) для микроЭВМ серии «Электроника», дополненной платой контроллера АЭ.410.10, разработанной ИАиЭ СО АН СССР в стандарте магистрали ЭВМ и обеспечивающей формирование 8-цветных изображений в телевизионном растре 256×256 точек. Система ГРАСС состоит из трех блоков: графического редактора, каталога и исполнительного блока. Вход в каждый блок производится по командной строке в формате ОС РАФОС с указанием входных и выходных файлов и соответствующего квалификатора.

Графический редактор (ГРЕД)

При создании графических редакторов в отличие от текстовых редакторов разработчик сталкивается со специфическими проблемами сжатия информации и организации экранного редактирования. Например, использование полного дампинга экрана для сохранения и экспонирования изображения резко ограничивает объем банка изображений (до 10 изображений на стандартной дискете) и динамические возможности их последовательного экспонирования. Вместе с тем большинство технических изображений можно образовать наложением небольшого числа простейших геометрических элементов, а динамические изображения, как правило, представляют собой ряд последовательных изменений отдельных элементов. В таком случае каждый элемент может быть описан шестью величинами: тип, цвет, X-; Y-координаты, X-; Y-масштабы. Это позволяет хранить на одной дискете несколько сотен различных изображений. В отношении же экранного редактирования трудно обеспечить оперативный возврат к изменению ранее сформированных элементов («редактирование назад») из-за сложной для решения в общем виде задачи «вытаскивания» элементов изображения из-под позднее наложенных сверху следов других элементов.

В редакторе ГРЕД элемент изображения состоит из двух одинаковых геометрических символов, масштабы и координаты которых изменяются независимо в процессе редактирования, а атрибуты занимают 16 байт. Наряду со встроенным набором символов различного типа конфигурации (геометрические фигуры, специальные знаки, цифры, буквы русского, латинского и греческого алфавитов) пользователю предоставляются средства формирования собственных дополнительных наборов символов, которые могут храниться в отдельных дисковых файлах. Как правило, изображения стандартных символов формируются из двух цветовых компонентов (задающих основной цвет и цвет фона), хотя допустимо использование и шести. Физические цвета компонентов могут оперативно изменяться операциями редактирования.

Экранное изображение конструируется последовательным оставлением следов (фиксированием изображения) рабочего или редактируемого элемента с записью атрибутов рабочего элемента в выходной файл. Оставление следа на экране может производиться в одном из трех режимов: 1) посредством фиксации изображений двух символов элемента в точках их расположения на экране; 2) в форме перемещения образа символа из точки расположения смежного символа в точку расположения рабочего символа элемента без нарушения фонового изображения, с постепенным изменением масштабов (если масштабы двух символов элемента не одинаковы) и с фиксацией изображения только в конечной точке; 3) аналогично режиму 2, но на экране остаются следы во всех промежуточных положениях символа в процессе его движения — фиксация

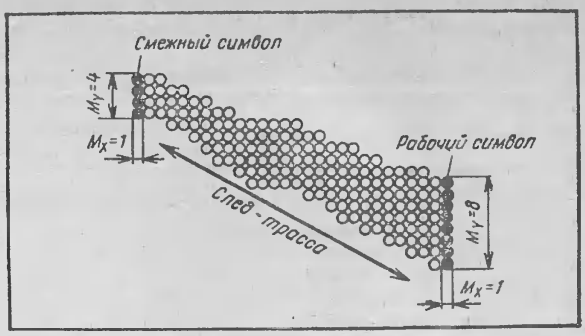


Рис. 1. Построение следа-трассы по двум символам типа «точка» (зачернены)

трассы (рис. 1). Применение режимов 2, 3 — это дополнительное средство конструирования динамических изображений (фильмов). Другим способом является запись в выходной файл признака требуемого числа повторений экспонирования некоторой последовательности элементов. Однако часто оказывается достаточным использование обычной динамики, определенной ненулевым временем вывода на экран очередных элементов.

В процессе редактирования только один из двух символов рабочего элемента (рабочий символ) экспонируется на экране. Другой (смежный) символ не виден на экране до выполнения операции смены рабочего символа. Изображение рабочего символа визуально легко идентифицируется, поскольку только оно в отличие от фонового изображения (ранее оставленных следов) может перемещаться по экрану и менять цвет при исполнении соответствующих операций редактирования. Перемещение рабочего символа по экрану не нарушает фонового изображения (скользит поверх фона). Это системно обеспечивается сохранением и восстановлением фонового изображения в поле рамки рабочего символа при любых его изменениях. Для большинства стандартных символов используется рамка в 7×9 точек (минимальный размер изображения символа). При однократном изменении масштаба размер рабочего символа увеличивается/уменьшается на размер исходной рамки по соответствующему измерению (рис. 2). Для символов пользователя максимальный размер исходной рамки составляет 20×20 точек. Когда пользователь формирует собственный символ, система запрашивает требуемый размер рамки и экспонирует ее на экране. Далее в поле этой рамки обычной техникой оставления следов рисуется требуемое изображение символа, что завершается записью символа в набор пользователя.

Для удобства редактирования имеется возможность устанавливать и менять операционный режим (режим выполнения операций) из четырех вариантов: 1) режим

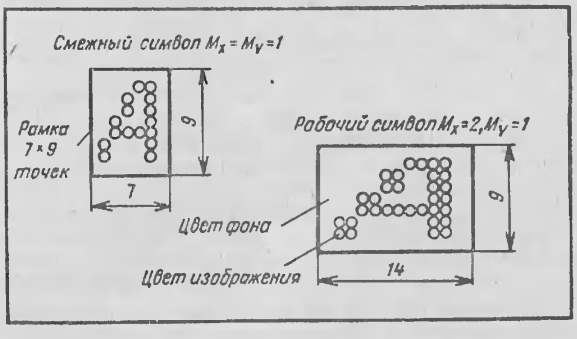


Рис. 2. Символы типа буквы «А» (X — масштаб рабочего символа равен 2)

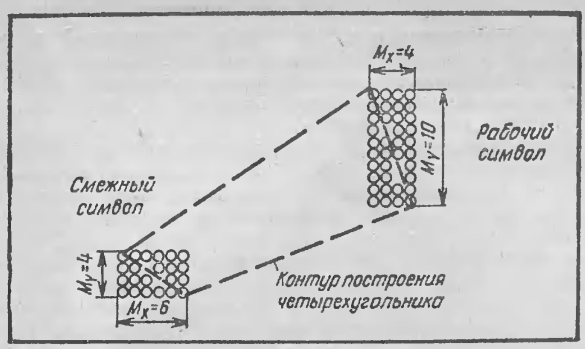


Рис. 3. Использование символов типа «точка» для получения двух прямоугольников или произвольного четырехугольника

раздельного редактирования символов элемента (действие операций перемещения и изменения масштабов распространяется только на рабочий символ), при нажатии на клавиши в нижнем регистре экспонируются цифры и русские буквы; 2) режим синхронного редактирования, когда в отличие от режима 1 действие любой операции распространяется на оба символа элемента, при этом первоначальная разница в координатах и масштабах символов сохраняется; 3) режим совместного редактирования, когда в отличие от режима 2 смежный символ всегда совмещен с рабочим символом по координатам и масштабам; 4) в отличие от режима 3 при нажатии на клавиши в нижнем регистре экспонируются латинские буквы. Таким образом, установка типа символа, являющегося изображением цифры или буквы русского или латинского алфавитов, производится нажатием соответствующей клавиши в нижнем регистре в зависимости от установленного операционного режима. Установка других типов конфигурации стандартных символов и символов пользователя производится специальными операциями с указанием порядкового номера типа символа.

Наиболее употребительным инструментом для конструирования изображений является символ типа «точка». Так, простым изменением масштабов из точки можно получить любые формы окрашенного прямоугольника с горизонтальным основанием (рис. 3). Посредством следа-трассы между двумя символами элемента (режим 3) получаются горизонтальные и наклонные линии и окрашенные четырехугольники (см. рис. 1). Кроме того, в редакторе имеется операция построения произвольного окрашенного четырехугольника (частный случай — треугольник) или его периметра, в котором положение угловых точек определяется крайними точками диагоналей двух символов элемента (см. рис. 3). Положение этих крайних точек в свою очередь можно изменить обычными операциями масштабирования и перемещения символов.

Результат работы редактора — выходной файл, в котором последовательно записаны атрибуты элементов сформированного изображения. Этот файл можно использовать в качестве входного файла для повторного редактирования или экспонирования полученного изображения в исполнительном блоке. В первом случае операцией считывания вызываются требуемые (обычно — последовательные) элементы входного файла, а операциями редактирования очередной считанный элемент изменяется и записывается в новый выходной файл. «Редактирование назад» решено посредством определения операции переназначения выходного файла в качестве входного без выхода из блока.

Полный список операций редактора ГРАСС, выполнение каждой из которых производится нажатием на указанную клавишу, приведен в нижеследующей таблице, где использованы обозначения клавиатуры дис-

Список операций редактора ГРАСС

Клавиша	Назначение операции	Пояснение
?	Установка признака операторного режима	N — номер режима
M	Установка признака режима оставления следа	N — номер режима
T	Операция оставления следа элемента на экране	N=1 — след выполняется цветом фона
C	Очистка экрана	N — номер цвета фона
P	Смена рабочего символа	При каждом исполнении
A	Выдача атрибутов рабочего элемента	N=1 — производится на принтер для всех элементов входного файла
S	Смена типа конфигурации символов элемента	N — номер типа (по умолчанию устанавливается следующий тип)
U	Установка символа из набора пользователя	N — номер типа конфигурации
F	Формирование символа из набора пользователя	N — номер типа конфигурации
⟨ПС⟩	Запись изображения символа	В набор пользователя
O	Установка символа типа «эллипс»	Размеры полуосей задаются X-, Y-масштабами
@	Установка символа типа «четырехугольник»	N=1, 2, 3, 4, 5 — уточняет вид четырехугольника
—	Совмещение смежного символа с рабочим	По координатам и масштабам
⟨ГТ⟩	Горизонтальная табуляция	Перемещение символа на одну алфавитную позицию вправо
.	В центр экрана	Перемещение рабочего символа
/ или \	Симметричное отражение через ось X или Y, проходящую по центру экрана	Рабочий символ перемещается в симметричную позицию
Стрелки простые	Перемещение символа в четырех направлениях	N — число шагов перемещения рабочего символа
Наклонная стрелка	Перемещение символа в нулевую позицию	Рабочий символ перемещается в верхний левый угол экрана
Стрелки с планками	Увеличение/уменьшение X, Y-масштабов	N — множитель изменения масштаба рабочего символа
¬|	Смена масштабов	X-, Y-масштабы рабочего символа взаимно меняются по величине
N, G, B, R, Y, V, L, D	Установка цвета: белый, зеленый, синий, красный, желтый, малиновый, голубой, черный	N — номер условного цветового компонента для обоих символов элемента

Окончание табл.

Клавиша	Назначение операции	Пояснение
J	Задание числа повторений N-предшествующих элементов	Число повторений задается в ответ на запрос системы
I	Ввод элемента из входного файла	N — порядковый номер элемента (по умолчанию — следующий элемент)
W	Запись рабочего элемента в выходной файл	N — число повторений записи
:	Скобки макрокоманды	Заключенная в скобки последовательность операций запоминается как макрокоманда
;	Выполнение макрокоманды	N — число повторных исполнений
Q	Выход из блока с несохранением выходного файла	
E	Выход из блока с сохранением выходного файла	N=1 — переназначение входного файла без выхода из блока

Примечание. Прототипом рассмотренного редактора является телевизионный редактор TVED, разработанный Третьяковым В. П., Платоновым С. В., Дерием Б. Н. в ИАиЭ СО АН СССР. Заимствования принципиального характера касаются понятий элемента изображения и режимов оставления следа.

плея типа 15ИЭ-00—13. Некоторым операциям может предшествовать набор числового аргумента, который в таблице обозначается через N. При отсутствии числового аргумента он в большинстве случаев полагается по умолчанию равным 1.

Подсистема управления банком изображений

Средства управления банком изображений сосредоточены в блоке каталога и исполнительном блоке и предназначены соответственно для формирования и использования банка изображений.

Банк изображений представляет собой множество карт, каждая из которых имеет четыре раздела: номер, используемый для редактирования карты в блоке каталога; собственно изображение, представленное последовательностью элементов и сформированное средствами графического редактора; ключ, представляющий собой алфавитно-цифровой идентификатор карты для вызова ее изображения на экран при работе в исполнительном блоке; список следующих карт, перечисляющий те карты, изображения которых могут быть вызваны сразу после вызова данной карты при работе в исполнительном блоке. Наличие раздела следующих карт позволяет организовывать банки изображений с достаточно сложной графовой структурой взаимосвязи карт (например, неупорядоченное множество, линейная или иерархическая структура, структура с произвольными циклами). В случае пустого списка следующих карт по умолчанию полагается, что карта связана со всеми картами, не имеющими предшествующих.

Директивы блока каталога позволяют создавать и уничтожать карты по их номерам, выдавать содержимое разделов карт и редактировать отдельные разделы. Изображение карты можно сформировать для вывода на телевизионный монитор посредством изображения к блоку графического редактора или для вывода на алфавитный терминал с использованием средств самого блока каталога. Результатом работы в блоке каталога является выходной файл, включающий заголовок, в котором в порядке номеров сформированных карт перечисляются ключи, списки следования и указатели на описания изображений и последующий набор описаний изображений карт.

Исполнительный блок предназначен для использования созданного средствами блоков каталога и редактора банка изображений указанного в виде входного файла. После входа в блок на экран выдается изображение карты с нулевым номером. Такая карта, если она сформирована, может содержать начальную информацию о банке изображений и инструкцию к работе с ним. Далее посредством набора соответствующих ключей могут вызываться требуемые изображения. При этом, как отмечено выше, в каждый момент времени действенны только ключи, которые относятся к картам из текущего списка следующих карт (если все списки следования пустые, то в любой момент действенны ключи всех карт). Если ключ набран с ошибкой, то выдается изображение карты с нулевым ключом (при условии, что таковая карта определена в текущем множестве следующих карт). Данная возможность позволяет при формировании каталога предусматривать реакции на ошибочные или поисковые действия пользователя, содержать подсказки и т. п.

Таким образом, рассмотренные средства позволяют создавать неупорядоченные банки изображений и сложно организованные диалоговые информационные системы, в которых возможные в каждый момент изменения информационного поля экрана зависят от предшествующей цепочки действий — директив пользователя (набранных ключей). При формировании каждого изображения в блоке графического редактора можно предусмотреть, чтобы предшествующее изображение на экране полностью стиралось или пофрагментно дополнялось новым изображением.

При работе исполнительного блока система ведет протокол, в котором фиксируются действия пользователя и временные интервалы от выдачи каждого изображения до выполнения следующего действия. Протокол позволяет проводить эргономический анализ эффективности принятой в конкретном банке изображений формы представления информации и средств диалога.

Опыт эксплуатации системы ГРАСС показал эффективность ее применения не только для решения перечисленных задач, но и для знакомства с особенностями диалога с ЭВМ и обучения алгоритмической проработке задач как преподавателей и сотрудников, так и школьников, включая учащихся младших классов. Рассмотренные графические и диалоговые возможности содержат не менее увлекательные и наглядные игровые компоненты, чем популярные компьютерные игры, но в отличие от последних способствуют совершенствованию не только скорости реакции, но и развитию алгоритмического мышления. Наличие в блоке каталога возможности формирования изображений для вывода на алфавитный терминал позволяет применять систему ГРАСС для обучения и диалога и при отсутствии телевизионного контроллера и монитора. Основные принципы автономной работы подсистемы управления банком изображений ранее опробованы в реализации на БЭСМ-6 *.

Система ГРАСС написана на языке MACRO-11 и в формате загрузки занимает объем около 14К байт. Посредством корректировки небольшого числа программных компонентов систему можно оперативно перенастроить на работу с другим типом телевизионного контроллера, в котором используется протокол магистрали ЭВМ или КАМАКа.

Телефон для справок: 939-50-05, Москва.

Статья поступила 4 марта 1987 г.

* Кулаичев А. П., Рамендик Д. М., Славуцкая М. В. Диалоговая система подготовки и предъявления зрительной информации// Вопросы психологии.— 1985.— № 4.— С. 118—120.

УДК 681.3.022

М. М. Алексеенко, М. А. Березовский, Б. Н. Свирин, А. К. Яблонский, В. В. Яким

ИНТЕРАКТИВНАЯ СИСТЕМА ГЕОМЕТРИЧЕСКОГО МОДЕЛИРОВАНИЯ ДЛЯ СМ ЭВМ И ВЕКТОРНОГО ПРОЦЕССОРА

Возможность создания сквозных систем автоматизированного проектирования (САПР) в области машиностроения в значительной мере зависит от полноты информации в ЭВМ о конструируемом объекте. В связи с этим в САПР используются все более сложные модели, в частности для описания геометрии объектов.

Одна из таких моделей — представление объекта как совокупности реалистических* примитивов или конструктивной модели твердого тела (КМТТ). Такое представление является более близким к реальному объекту по сравнению с традиционно используемыми в интерактивном конструировании 2—2,5D моделями и наиболее эффективно с точки зрения автоматизации всего цикла конструкторско-технологических работ.

Однако с ростом полноты данных о геометрии объекта манипулирование этими данными усложняется. Так, при решении задач машиностроительного проектирования объем вычислений, необходимых для визуализации объектов в КМТТ-представлении,— примерно 100 млн. операций с плавающей запятой. Учитывая, что для поддерживания интерактивности работы в САПР время создания изображения в КМТТ не должно превышать 10 с, нижняя граница производительности машиностроительного АРМ должна быть не менее 10 млн. операций с плавающей запятой в секунду.

Один из способов достижения требуемой производительности АРМ — включение в графическую подсистему специализированного геометрического процессора, ориентированного на решение только задач обработки геометрических данных. Узкая специализация геометрического процессора снижает экономические показатели графической подсистемы АРМ и ничего не дает для решения таких не менее важных задач САПР, как размерная и прочностная обработка, кинематическое и динамическое моделирование, разработка технологии и др.

В связи с этим важное значение приобретает вопрос о разработке такой архитектуры технических средств машиностроительного АРМ, которая при сохранении характерных для СМ ЭВМ малых габаритов и низкой стоимости обеспечивала бы требуемую производительность по наиболее широкому спектру задач САПР.

Оптимальное соотношение «стоимость/производительность» для машиностроительного АРМ достигается созданием комплекса, включающего центральную универсальную ЭВМ и присоединенный векторный процессор. В этом комплексе на векторный процессор возлагается решение вычислительно емких задач визуализации и инженерного анализа проектируемых объектов, а центральная ЭВМ обеспечивает возможность диалогового проектирования, включая взаимодействие с интегрированным банком геометрических данных. Важно подчеркнуть, что стоимость комплекса оказывается существенно ниже, чем при использовании эквивалентной универсальной ЭВМ.

Исходя из вышесказанного в Институте автоматизации проектирования АН СССР на базе комплекса технических средств в составе центральной ЭВМ типа СМ 1700 и векторного процессора типа ИЗОТ 2001С (производства НРБ) ведутся работы по созданию программного обеспечения, ориентированного на задачи машиностроения. Основные направления работы на первом этапе:

разработка системы интерактивного геометрического моделирования для автоматизированного проектирования (СИГМа);

разработка проблемно-ориентированного машиностроительного языка конструирования (МАЯК), являющегося интерактивной надстройкой над СИГМой.

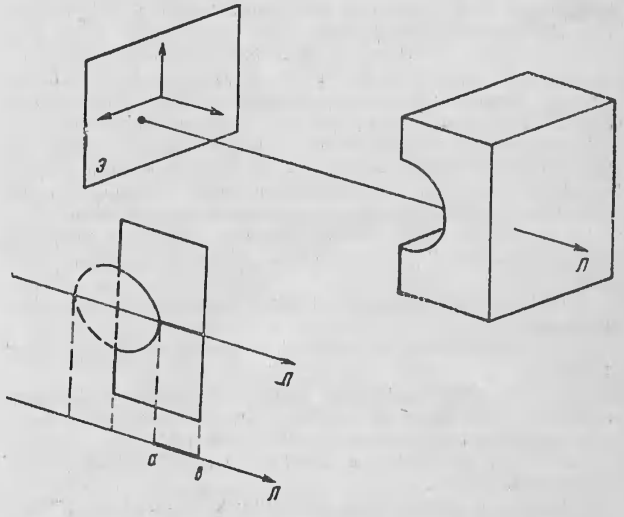
Система интерактивного геометрического моделирования

В СИГМе объект конструирования представляется как результат выполнения ограниченного набора операций над объемными примитивами. Типичный набор примитивов состоит из параллелепипеда, конуса, цилиндра и т. п. В качестве операций обычно используются объединение, пересечение и разность. При достаточном числе примитивов можно сформировать объект произвольной сложности.

Программное обеспечение СИГМы позволяет изображать объекты с любой точки наблюдения и под произвольными углами. Для построения изображений объектов применяется метод, известный под названием «трассировка луча» [1]. Суть его заключается в том, что из каждого элемента матрицы выпускается луч, параллельный направлению наблюдения. Найдя точки пересечения луча с поверхностями примитивов, можно свести трехмерные объемные операции к более простым одномерным (см. рисунок). Видимым наблюдателю участком объекта является ближайшая к экрану точка его поверхности (C). Освещенность этой точки определяет яркость соответствующего элемента изображения. Значение освещенности находится как величина, пропорциональная скалярному произведению вектора нормали к поверхности и направлению пучка света [2].

Для построения полного изображения описанная процедура повторяется для каждого элемента матрицы экрана.

Реализация трассировки луча на векторном процессоре производилась с учетом особенностей его аппаратной архитектуры [2]. С точки зрения программирова-



Сведение трехмерной объемной операции к одномерной. На рисунке изображен объект, сформированный как разность параллелепипеда и цилиндра, а также отрезки, на которые луч $Л$ поделен поверхностями примитивов. Точками поверхности результирующего объекта являются a и b, причем видимой на изображении будет точка a, поскольку она ближе к плоскости экрана $Э$

* Реалистичность изображения подразумевает передачу объемных и цветовых характеристик объекта.

ния эти особенности сводятся к тому, что максимальная производительность процессора достигается на действиях над большими массивами чисел с плавающей запятой, расположенными в памяти регулярным образом. Вычисления в алгоритме [2] выполняются одновременно для 1024 лучей, образующих квадратный фрагмент экрана размером 32×32 элемента. Такое укрупнение дало возможность эффективно запрограммировать все стадии трассировки луча, что обеспечило общую высокую производительность системы.

Время построения тестовых изображений (см. разворот) составило 5...20 с против десятков минут, затрачиваемых современными промышленными САПР, базирующимися на супермини-ЭВМ.

СИГМа реализована в виде двух компонент, одна из которых размещена в центральной ЭВМ, а другая — в векторном процессоре. Первая компонента обеспечивает межпроцессорный интерфейс и запрограммирована на языке Фортран; объем этих программ составил около 300 операторов. Вторая компонента выполняет непосредственно построение изображения. Общий объем этой части системы, написанной на языках Фортран и ассемблера векторного процессора, составляет примерно 1500 операторов без учета объема библиотечных подпрограмм.

Машиностроительный язык конструирования

Система МАЯК выступает в качестве надстройки над СИГМой для реализации интерактивного взаимодействия пользователя-конструктора в терминах геометрических и конструкторских операций, чем выгодно отличается от подходов, например [4, 5]. Под конструкторской операцией понимается определенная последовательность геометрических операций, эквивалентная типовому конструкторскому действию, например, скругление грани, снятие фаски, сопряжения с различными видами посадок и т. п.

В средства интерактивного взаимодействия МАЯК входят графический цветной дисплей с программно-управляемым графическим курсором (перекрестье) и функциональная клавиатура в стандарте СМ ЭВМ. Экран дисплея разделен на четыре функциональных окна (см. разворот, рис. 4): для вывода меню и значений изменяемых параметров, сообщений системы, визуализации конструируемого объекта (основное окно) и визуализации присоединяемой компоненты (вспомогательное окно). Данные и команды вводятся с помощью функциональной клавиатуры, на которой логически выделены четыре поля: алфавитно-цифровых данных, селекторов меню, команд и управления графическим курсором.

Программное обеспечение первой очереди МАЯКа обеспечивает следующие функциональные возможности:

задание условий визуализации геометрических данных в основном и вспомогательном окнах экрана;

выбор вида работ (оригинальная или прототипная разработка, включая коррекцию ранее созданного объекта);

выбор геометрических и позиционных параметров примитивов;

позиционирование компонентов конструируемого объекта;

хранение геометрических данных в специализированной базе данных и получение справочной информации;

синтаксический анализ входных данных;

обработку прерываний диалога, инициируемых пользователем;

создание и выполнение командных файлов конструктора.

Таким образом, программное обеспечение МАЯКа реализует конструкторские операции, включая задание фактических параметров примитивов КМТТ, автономное конструирование объектов и их позиционирование при «сборке» сложных конструкций.

Программное обеспечение МАЯКа написано на языке PL/1. Объем программ составил около 3 тыс. операторов. При этом время работы программ поддержки конструкторских операций оказывается несущественным по сравнению с временами геометрического моделирования и визуализации объекта.

Телефон для справок: 408-61-88, Москва.

ЛИТЕРАТУРА

1. Roth S. D. Ray Casting for Modelling Solids.— Computer Graphics and Image Processing.— 1982.— Vol. 18.— P. 109—144.
2. Березовский М. А., Яблонский А. К. Параллельный алгоритм визуализации для системы интерактивного геометрического моделирования на векторном процессе // Автометрия.— 1987.— № 5.
3. Ньюмен В. Н., Спрулл Р. Ф. Принципы интерактивной машинной графики.— М.: Мир, 1973.
4. Wesly M. A., Lozano-Perez T., Lieberman L. I., Lavin M. A., Grossman D. D. A Geometric Modeling System for Automated Mechanical Assembly.— IBM J. Res. Develop.— January. 1980.— Vol. 24.— № 1.
5. Kunwoo Lee, Guy Andrews. Inference of the positions of components in an assembly (part 2).— Computer-aided design.— January / February.— 1985.— Vol. 17 — № 1.

Статья поступила 14 июня 1987 г.

Уважаемый Главный редактор!

Не знаю, по адресу ли я обращаюсь, и будет ли Вам интересно то, что я хочу сказать.

Сначала несколько слов о себе.

Я математик, закончил университет. После вуза работал программистом (не «великим», но зато «старшим»). Вначале на кафедре в институте, а последние 7 лет на заводе в отделе АСУП. Завод большой. 13 тысяч человек; отдел АСУП — около 100 человек.

У нас наработано много программ, около 300 выходных форм (не очень ценимых руководством предприятия). В последнее время завод не выполняет план, меняются директора, может им бы помогло исправить положение внедрение хороших подсистем АСУП, хотя бы учета, но им не до этого. У нас, как и повсюду, «Принцип главного руководителя АСУ» не применяется. Но это в большей степени проблемы директора, чем мои, и пишу я об этом, чтоб Вы знали, что мой случай типичный, и только поэтому, на мой взгляд, может представлять для Вас интерес.

У нас на ВЦ две ЕС-1022 и одна ЕС-1036. Работаем в ДОСе и четыре года в ОСе. На тридцать шестой даже сгенерировали СВМ, правда, еще не придумали, зачем.

Какие проблемы? Последние года три решили мы превращать количество в качество: продумать единую НСИ (а то много дублирования развелось: у нас три шифратора цехов, а цехов всего-то до сотни), найти программную среду получше.

Рядом с нами региональный центр обслуживания «Запад-ЭВМкомплекс». Правда, из программных средств у них выбор крайне небогатый, проявляем интерес к базам данных. ИНЕС видели в действии. Завод купил подсистему «Основные фонды» на ИНЕСе. Не знаю как на ЕС-1036 (туда ее еще не переместили), но на ЕС-1022 с 512К памяти — жалкое зрелище.

Постепенно выяснили, что самое лучшее — это «Спектр». Но и это лучшее хвалили по-математически: «от противного», по сравнению с другими базами.

Может сказалось отсутствие информации, но мы очередной раз решили ПС сделать сами (надо ли говорить, что это не лучший выход). Только где взять информацию о существующих ПС. Пользователей ЕС ЭВМ никто не собирает, негде поделиться опытом. Каталог ФАП Минприбора в Калинине уже перевалил за 1000 единиц, а все аннотации почти одинаковые, что выбирать? Где обзоры существующих ПС, баз данных с учетом типов ЭВМ, задач, структур предприятий? Не знаю.

У нас в отделе 10 программистов. Программы пишем на языке ассемблера. Привыкли писать быстрые программы, занимающие мало места на магнитных дисках, и поэтому кажется несколько странным, что ИНЕС вводит перфокарты

(Продолжение см. на стр. 85).

УДК 681.325.5—181.4

В. Н. Королев, А. С. Жарков. А. П. Зубченко, А. Ю. Штанаков

МИКРОПРОЦЕССОРНАЯ СИСТЕМА МС-80

МС-80 — базовая агрегатируемая система, предназначенная для построения программируемых контроллеров (ПК) и распределенных систем управления. Технические средства МС-80 выполнены в виде **функционально законченных модулей**, компонуемых в общем каркасе. Микросхемы высокой степени интеграции обеспечивают при небольших габаритных размерах широкие функциональные возможности ПК. В зависимости от сложности задачи, изменяя количество и структуру функциональных модулей, можно компоновать ПК с различным количеством входных-выходных дискретных и аналоговых сигналов. Буквенно-цифровой дисплей и клавишная панель встроены. Можно подключить к ПК видеотерминал и печатающее устройство со стандартными интерфейсами ИРПР, ИРПС или «токовая петля 20 мА». Это позволяет контролировать и регистрировать большое количество технологической информации.

Для обмена информацией между техническими средствами ПК часто используют интерфейсы — упрощенные варианты широко распространенных интерфейсных шин: И-41, S-100, Q-bus, Multibus и т. д. Такое решение позволяет реализовать простые процедуры обмена с помощью сравнительно небольшого количества линий интерфейсной магистрали (ИМ), но затрудняет наращивание функциональных возможностей ПК. Упрощенная структура ИМ ограничивает область применения таких устройств локальным и сублокальным уровнями управления.

Для устройств управления более высокого уровня (несколькими ПК) нужны универсальные ИМ [1].

Системный интерфейс ADA-32 разработан для связи между техническими средствами МС-80. На его основе можно реализовать высоконадежные устройства управления различной степени сложности — программируемые контроллеры; устройства сбора, обработки и регистрации информации; мультипроцессорные информационно-управляющие системы. Структуры распределенных систем управления станут технически однородными, затраты на их обслуживание и ремонт уменьшатся. Интерфейс ADA-32 позволяет резервировать информационную шину, диагностировать и локализировать неисправности, программно генерировать структуру технических средств устройств управления.

Асинхронный механизм обмена информацией по мультиплексированным линиям адреса-данных допускает подключение к ИМ медленных и высокоскоростных контроллеров внешних устройств. При реализации ПК с упрощенными процедурами обмена информацией используется только минимально-необходимый состав линий ИМ. Мультипроцессорный режим работы на верхнем уровне управления обеспечивается распределенным арбитражем доступа к системным ресурсам, предусматривающим наличие на каждом активном модуле (ведущем) схемы-арбитра, непосредственно управляющей захватом управления ИМ. К ИМ подключаются не более 32 модулей, каждый из которых может быть ведущим. Географическая адресация и мультиплексирование информации позволяют адресовать в каждом модуле до 2^{32} 8-, 16-, 24- или 32-разрядных слов памяти и устройств ввода-вывода. **Мультиплексирование** снижает пропускную способность интерфейса, но сокращает число линий связи при обмене информацией, т. е. **увеличивает надежность устройств управления.**

Операция обмена информацией между техническими средствами, подключенными к ИМ, состоят из циклов инициализации-статуса (И—С), логической адресации и обмена данными.

В цикле И—С ведущий модуль инициализирует один из модулей, подключенных к ИМ, идентифицирует его и задает (с помощью служебных регистров) определенный режим работы.

Генерировать структуры систем управления из избыточного набора модулей, подключенных к ИМ, блокировать работу неисправных и подключать вместо них резервные модули можно программно. **За счет программной реконфигурации системной структуры при отказах** можно полностью или частично дублировать технические средства управления, т. е. создавать **отказоустойчивые системы с высокой живучестью**, сохраняющие работоспособность в случае частичных отказов аппаратуры [2].

Циклы логической адресации и обмена данными — обычный асинхронный механизм обмена между ведущим и ведомыми модулями. **Блочный обмен и контроль по четности**, проводимые во всех циклах, значительно **повышают скорость и достоверность операций обмена информацией.** Для повышения надежности работы организована обработка информации при сбоях сети электропитания.

Все сигнальные линии ИМ и линии питающих напряжений выведены на разъем типа СНП 59—96, причем на ИМ расположены розетки, а на модулях — вилки. Второй такой разъем служит для подключения напряжений питания аналоговых схем модулей, и связи их с внешними устройствами. Варианты исполнения ИМ — магистраль каркаса и кабельная магистраль. Вторая связывает отдельные модули, каркасы и конструктивно разнородные устройства, способные обмениваться информацией по протоколу ADA-32. Максимальная длина всех сегментов кабельной магистрали — до 1 м.

В системе МС-80 конструктивно определены два типа модулей.

Тип А — основной функциональный модуль на печатной плате (235×160 мм), снабженный рамкой и экстракторами, —предназначен для реализации элементов управления, устройств ввода-вывода, запоминающих устройств и т. п.

Тип В — сервисный модуль (часть передней панели компоновочного каркаса), служит для компоновки устройств клавишного ввода и отображения информации.

Связь с ИМ осуществляется через модуль — адаптер типа А. Модули второго типа могут и не иметь непосредственной связи с ИМ, если они служат консольным устройством какого-либо модуля типа А. Все функциональные модули и источник электропитания разме-

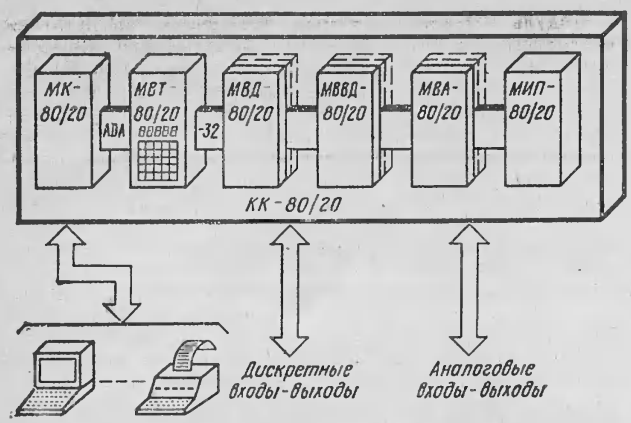


Рис. 1. Структурная схема ПК на базе МС-80

щены в компоновочном каркасе с шагом, кратным 20 мм. Ширина модулей различна, но кратна 20 мм. Это отражено в обозначении типоразмера. Так, ширина передней панели модулей A1 или B1 равна 20 мм.

В настоящее время в рамках МС-80 для создания на ее основе ПК разработан ряд технических средств.

Модуль управления МК-80/20 (рис. 1) управляет системными ресурсами ИМ, в том числе приемом и обработкой запросов прерывания, самоарбитражем ИМ, формирует временные интервалы. МК-80/20 управляет двухсторонней связью с периферийными устройствами по интерфейсу ИРПС, или «токовая петля 20 мА» (посредством двух последовательных каналов), двухсторонней связью с устройствами по интерфейсу ИРПР (посредством четырех стробируемых двунаправленных восьмиразрядных каналов), программированием расположенного на плате ПЗУ.

Основные технические характеристики модуля МК-80/20

Объем ОЗУ, К байт	4
Максимальный объем ПЗУ, К байт	16
Количество каналов формирования временных интервалов	4
Длительность формируемых временных интервалов, с	25×10^{-4}—12×10^{6}
Количество последовательных каналов обмена информацией	2
Расстояние приема-передачи информации по последовательному каналу (витая пара), м, не менее	1000
Количество параллельных каналов обмена информацией	4×8
Расстояние приема-передачи информации по параллельному каналу, м, не менее	7
Типоразмер модуля	A1
Потребляемая мощность, ВА, не более	20

Центральный процессорный элемент (см. рис. 2) — микропроцессор КР580ВМ80. Особенность модуля — наличие ОЗУ со сверхмалым потреблением мощности и резервным питанием от аккумуляторных батарей. В качестве ПЗУ можно установить разнообразные типы микросхем: с электрической перезаписью (КР1611РР1, КМ1609РР1), с ультрафиолетовым стиранием (К573РФ4, ..., К573РФ6), с финишным программированием (КР1610РЕ1, КМ568РЕ4).

Модуль ввода — таймер МВТ-80/20 работает в качестве пульта ввода информации с клавиатуры, буквенно-цифрового индикаторного устройства и устройства сигналов времени.

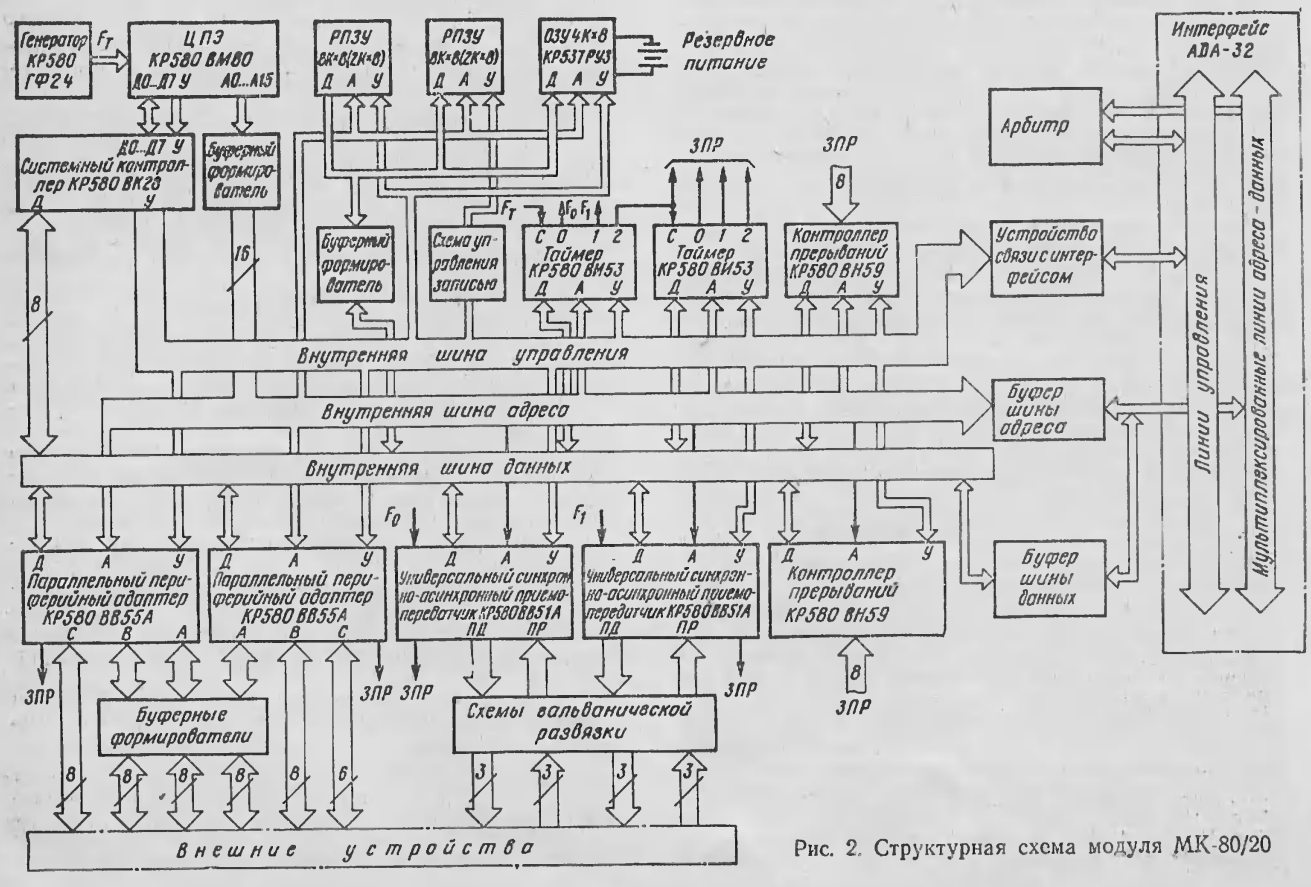


Рис. 2. Структурная схема модуля МК-80/20

Основные технические характеристики МВТ-80/20

Количество разрядов буквенно-цифрового индикатора	6
Отображаемые сигналы	0...9, A, B, C, D, E, F
Количество дискретных индикаторов типа «включено-выключено»	6
Количество кнопок клавиатуры	23
Потребляемая мощность, ВА, не более	8
Типоразмер модуля	B4+A1

Модуль содержит встроенный знакогенератор и буферное ЗУ для хранения кодов отображаемых символов. Информация со встроенных часов реального времени по запросу передается на ИМ. **Благодаря выполнению устройства сигналов времени на КМОП-микросхемах, оно защищено от кратковременных (до 1 мин) отключений источников электропитания.** Модуль связан с ИМ адаптерной платой типа А.

Модуль МВА-80/20, который предназначен для ввода-вывода и обработки аналоговых сигналов, может работать автономно (в качестве контроллера аналоговой информации) и под управлением МК-80/20 (для расширения функциональных возможностей системы). Встроенный микропроцессор КР580ВМ80 программно устраняет температурную и временную составляющие погрешности измерения, корректирует дрейф «0», оптимизирует коэффициент усиления масштабирующего усилителя, контролирует работоспособность измерительного тракта, управляет работой АЦП и ЦАП. АЦП гальванически развязан от остальных схем модуля. Алгоритм работы модуля «зашит» во встроенное РПЗУ К573РФ5. Цифровые эквиваленты измеренных значений входных сигналов хранятся в ОЗУ, откуда могут быть выданы по запросу на ИМ. Граничные значения уставок измеряемых аналоговых сигналов записываются с ИМ в ОЗУ модуля. При несоответствии уставкам формируется интерфейсный сигнал запроса прерывания.

Основные технические характеристики МВА-80/20

Количество разрядов АЦП	12
Количество разряров ЦАП	10
Количество каналов АЦП	48(24×2)
Количество каналов ЦАП	2
Диапазон входных-выходных аналоговых сигналов, В	0...10,24
Погрешность преобразования АЦП, %, не более	0,1
Время преобразования АЦП, мкс, не более	200
Время установки выходного напряжения ЦАП, мкс, не более	50
Мощность потребления, ВА, не более	10
Типоразмер модуля	A1

Модули ввода МВВД-80/20 и вывода МВД-80/20 дискретных сигналов служат соответственно для приема информации от датчиков различных типов и для дискретного управления исполнительными механизмами. **Схемы гальванической развязки выходных и входных каскадов повышают помехозащищенность модулей.** При смене уровня напряжения на любом из входов модуль МВВД-80/20 формирует интерфейсный сигнал запроса прерывания и фиксирует код номера соответствующего входа. Благодаря этому **модуль пригоден для работы в условиях быстропротекающих и аварийных процессов,** исключает циклы периодического опроса входных сигналов.

Основные технические характеристики МВВД-80/20

Количество дискретных входов	32
Входной ток, мА, не более	10
Мощность потребления, ВА, не более	5
Типоразмер модуля	A1

Основные технические характеристики МВД-80/20

Количество дискретных выходов	32
Выходной ток, А, не более	0,5
Мощность потребления, ВА, не более	7
Типоразмер модуля	A1

Модуль источника питания МИП-80/20 обеспечивает постоянным стабилизированным напряжением аналоговые и цифровые схемы технических средств МС-80. Схема МИП-80/20 бестрансформаторная с промежуточным преобразованием напряжения сети в напряжение 150 В 20 кГц. Выходные напряжения стабилизируются широтно-импульсной модуляцией. Схема обнаружения и обработки сбоев в сети питания формирует интерфейсные сигналы «Сброс», «Запрет работы памяти», «Снижение напряжения сети», «Авария электропитания». Это гарантирует бесперебойную работу МС-80 в условиях нестабильного электропитания. **Для повышения помехоустойчивости источник питания снабжен сетевым фильтром.** Все каналы питающих напряжений защищены от превышения напряжения и короткого замыкания в цепи нагрузки. Для ограничения пусковых токов в момент включения в состав МС-80 введена схема «мягкого запуска».

Основные технические характеристики МИП-80/20

Напряжение питания цифровых схем, В	+5, ±12
Ток нагрузки по каналам +5В, А, не более	15
+12В, А, не более	2
—12В, А, не более	1
Напряжения питания аналоговых схем, В	±15×2 (с гальванической развязкой) ±15
Ток нагрузки по каналам питания аналоговых схем, А, не более	2
Питание	Сеть 200^{+22}_{-33} В, 50 Гц
Типоразмер модуля	A3+B2

Компоновочный каркас КК-80/20 (в двух вариантах) служит для установки 6 или 12 функциональных модулей. В каркас встраивается кросс-плата (физическая реализация системного интерфейса ADA-32). На задней стенке расположены разъемы для подключения линий ввода-вывода и резервного электропитания.

Небольшие габаритные размеры каркасов (260×130×260 мм и 260×260×260 мм) позволяют использовать ПК в качестве аппаратуры, встраиваемой в объект управления.

Одно из применений МС-80 — микропроцессорные контроллеры весоизмерительных устройств дозирования многокомпонентных смесей при изготовлении изделий из полимерных материалов. Компоненты дозируются в бункер-смеситель, установленный на платформу, механически связанную с датчиком силы. Точность взвешивания дозируемых компонентов определяет качественные характеристики изготовляемых изделий. Использование ПК при взвешивании исключило ручную математическую обработку информации, повысило точность измерения и достоверность отображаемой информации. Для составления рабочих программ на этой стадии использовалось ПО инструментальной микроЭВМ СМ1800.

По результатам 18-месячных производственных испытаний трех комплектов ПК на различных технологических объектах среднее время безотказной работы $T_б$— 4750 ч. Для технологических процессов с повышенными требованиями к надежности аппаратуры был применен метод резервирования ПК дублированием с восстановлением. Среднее время безотказной работы $T_{б.р} = 3\,T_б/2 + \mu\,T_б/2$; где μ — интенсивность восстановления технических средств (1/час). При средней интенсивности восстановления работоспособности ПК $\mu = 0{,}05$ (предполагается, что любая неисправность устраняется за 20 ч в соответствии с условиями промышленной эксплуатации) время безотказной работы для дублированных контроллеров составляет 564 тыс. ч.

Помехозащищенность ПК на базе МС-80 достаточно высока. Без дополнительных мер контроллеры сохраняют работоспособность при импульсных помехах в цепи питания амплитудой до 420 В [3].

Резидентное ПО МС-80 состоит из монитора, библиотеки арифметических функций и элементарных функций управления и обработки информации (например, ввода значений переменной, их усреднения, отображения и вывода на печатающее устройство). Для более полного использования возможностей ПК на базе МС-80 разрабатывается язык программирования в выражениях алгебры Буля и в символике релейно-контактных схем [4]. Язык выполняет функции обработки дискретной и аналоговой информации, транслятора рабочей программы, монитора и редактора текста. Это увеличит гибкость и эффективность применения ПК, сократит сроки разработки рабочих программ и их коррекции в условиях эксплуатации.

Адрес для справок: 659303, г. Бийск Алт. края, ул. П. Мерлина, д. 25/3, кв. 51, Королеву Владимиру Николаевичу.

ЛИТЕРАТУРА

1. Прангишвили И. В., Стецюра Г. Г. Микропроцессорные системы.— М.: Наука, 1980.
2. Прангишвили И. В. Микропроцессоры и локальные сети микроЭВМ в распределенных системах управления.— М.: Энергоатомиздат, 1985.
3. Шишонок Н. А., Репкин В. Ф., Барвинский Л. Л. Основы теории надежности и эксплуатации радиоэлектронной техники.— М.: Советское радио, 1964.
4. Мишель Ж., Лоржо К., Эспьо Б. Программируемые контроллеры.— М.: Машиностроение, 1986.

Статья поступила 24 сентября 1986 г.

УДК 681.32

Я. И. Томсинский

КАК РАЗРАБОТАТЬ ЭЛЕКТРОННУЮ СХЕМУ?

...электроника — это искусство, которое основано на нескольких основных законах и включает в себя большое количество практических правил и приемов. *П. Хоровиц, У. Хилл.* Искусство схемотехники. — М.: Мир, 1984.

Так, на двух ИС К176ТМ2, двух ИС К176ЛА7 и одной ИС К176ЛА9 может быть реализован 4-разрядный регистр. Каждый разряд такого регистра состоит из D-триггера и распределительного устройства, наличие которого увеличивает функциональные возможности регистра, делая его универсальным.

Аналоговые и цифровые интегральные схемы. Справочное пособие. — М.: Радио и связь, 1985.

1. Введение. Появление огромного количества наименований интегральных микросхем поставило перед разработчиками ранее несуществовавшую проблему выбора из ассортимента в несколько тысяч типов микросхем (выпускаемых электронной промышленностью) наиболее удобных для решения поставленной задачи, и создания на их основе оптимальных устройств. Решение этой проблемы — очень нелегкая задача из-за практически полного отсутствия литературы по разработке электронных схем, специалистов, наконец самой специальности — «Разработка электронных схем». Грамотная разработка стала уделом энтузиастов. Серийный выпуск микропроцессоров и микропроцессорно-ориентированных ИС не упростил, а еще более усложнил жизнь разработчикам РЭА, заставив их решать вопрос, применять или не применять микропроцессор в конкретном разрабатываемом устройстве.

Цель предлагаемой статьи — хоть чуть-чуть облегчить жизнь разработчика в существующих условиях. К сожалению, изменить сами условия автор не имеет возможности. Надеюсь, что Минэлектронпром, Минвуз, ГКНТ и издательство «Радио и связь» совместными усилиями предпримут хоть какие-нибудь меры по организации обучения разработчиков и обеспечению их полноценной информацией.

Для достижения поставленной цели предлагается несколько отличный от общепринятого алгоритм разработки электронных схем, основанный на привлечении коллективных знаний и опыта.

2. Выработка технических требований к разрабатываемому устройству — наиболее ответственный этап разработки, так как завышение этих требований может привести к значительному усложнению и удорожанию устройства, а также к снижению его надежности, в то время как следствие заниженных требований — неработоспособность разрабатываемой аппаратуры. Не следует стремиться к излишней универсальности устройства. Очень вероятно, что, когда заложенные «на перспективу» дополнительные функции потребуются, устройство уже устареет. На этом этапе необходимо привлечь крупнейших специалистов разрабатывающей организации, которые смогли бы «прикинуть» сложность реализации того или иного параметра устройства.

3. Составление функциональной схемы и изучение существующей элементной базы для ее реализации. Ознакомление с аналогичными устройствами. Если это возможно, попытайтесь составить функциональную схему самостоятельно, не знакомясь с устройствами-предшественниками. Нет смысла делать эту схему слишком подробной — все равно она будет уточняться на протяжении всего процесса разработки.

Тщательное изучение существующей и перспективной элементной базы — необходимое условие успешной разработки. К сожалению, основные сведения по этому вопросу можно почерпнуть лишь из устных бесед с коллегами и выуживая информацию из справочников. Если после изучения элементной базы функциональную схему не изменили, возможны три варианта: либо эта схема слишком проста, либо ее разработал крупный специалист, либо... изучение элементной базы надо продолжить.

Знакомство с аналогичными устройствами —занятие, безусловно, очень полезное, особенно если разработчик способен на критику и самокритику. Даже самое безграмотное устройство может содержать вполне рациональные мысли по замыслу или схемотехнике.

Результат этого этапа — скорректированная (если это необходимо) функциональная схема и примерный перечень основных комплектующих изделий, в первую очередь, микросхем, для ее реализации. По этому перечню разработчик заказывает в информационной службе своей организации подробную документацию.

4. Разработка принципиальной схемы. Как невозможно без любви к искусству, без вдохновения, без соответствующего опыта и специальных знаний написать картину, так и невозможно без соблюдения этих условий разработать красивую принципиальную схему устройства. Схемотехника — это искусство! Если Вы не согласны с этим утверждением, то поручите составление принципиальной схемы кому-нибудь другому или, по крайней мере, не пишите об этом процессе статьи и книги.

Обычный вопрос, который мне задают коллеги, **какие микросхемы**

можно рекомендовать для применения в конкретном разрабатываемом устройстве? К сожалению, не существует литературы, которая смогла бы ответить на этот вопрос.

Гораздо легче ответить на вопрос, **какие изделия применять не надо,** — снятые или снимаемые с производства, морально устаревшие. Автору удалось найти книгу, в которой приведен **достаточно подробный список морально устаревших микросхем** [1]. В этом справочнике предлагаются сведения о нескольких сотнях советских микросхем, ни одна из которых не может быть рекомендована для применения по вышеназванным причинам.

Применять или не применять микропроцессоры? Правильный ответ на этот вопрос, если ответ не очевиден, **может дать лишь сравнительная таблица параметров различных вариантов исполнения устройства. Обязательное условие достоверности результата сравнения — грамотная разработка всех вариантов схемы.**

Применение готовых схем узлов устройства, опубликованных в технической литературе, отнюдь не всегда облегчает жизнь разработчика. Наличие в литературе большого количества неудачных, а зачастую и просто безграмотных принципиальных схем, ошибок, опечаток приводит к необходимости тщательного анализа намечаемой к их использованию схемы. Например, в [2] на с. 62 приведена схема АЦП последовательного приближения на К572ПВ1, содержащая 35 компонентов, в том числе 5 микросхем. Низкое входное сопротивление АЦП на большом сигнале (менее 1 кОм) делает весьма ограниченным его применение. В рекламном проспекте на К572ПВ1 приведена схема, содержащая 4 компонента (2 микросхемы и 2 диода), обладающая теми же параметрами, однако не имеющая указанного выше недостатка (схема работает, если включить пятый, забытый компонент — резистор на выходе компаратора). Читателю предлагается самому решить, какую роль играют оставшиеся 30 компонентов в первой схеме.

Даже самые простые схемы, приведенные в технической литературе, не стоит применять без тщательного анализа ее работы (или причин неработоспособности). Если читатель хочет попрактиковаться и оценить свои аналитические способности, он может попытаться найти все ошибки в принципиальных схемах, приведенных в работе [3] на рис. 3,4. Их достаточно много, так что скучно не будет.

Как показывает практика, **наибольшие трудности у разработчиков вызывает сопряжение электронных устройств с реальными объектами.** Внимательное изучение всей номенклатуры микросхем, выпускаемых электронной промышленностью, значительно упрощает решение этой задачи. Но не удивляйтесь, если в справочной литературе Вы не найдете основных параметров той или иной микросхемы, которую кто-то из коллег рекомендовал использовать в разрабатываемом устройстве — это вполне обычная ситуация.

Итак, разработка схемы закончена. **С целью объективной оценки результатов разработки необходимо устроить обсуждение** (заседание техсовета) с участием ведущих специалистов разрабатывающей организации **(лучше с привлечением специалистов «со стороны»). Предлагается примерный список вопросов для обсуждения.**

Все ли примененные комплектующие изделия серийно выпускаются промышленностью, нет ли среди них устаревших?

Насколько обосновано применение дискретных приборов (транзисторов, диодов, конденсаторов, резисторов и т. д.), нельзя ли уменьшить их количество за счет микросхем?

Достаточно ли обосновано применение микросхем малой степени интеграции, снизит ли количество оборудования применение микросхем с большей степенью интеграции?

Все ли сделано в схеме для снижения потребляемой мощности, уменьшения количества источников питания?

Критична ли разработанная схема к замене комплектующих изделий, внешних устройств?

Приняты ли все меры для снижения количества межплатных связей с внешними устройствами, оптимально ли количество типономиналов плат?

Какие меры предприняты для защиты от возможных помех, развязаны ли по питанию аналоговая и цифровая части схемы?

Какие меры предприняты для защиты от превышения предельных режимов эксплуатации отдельных изделий, входящих в состав разработанного устройства, а также внешних устройств из-за выхода из строя блока питания, обрыва или замыкания внешних связей и т. д.?

Что сделано для обеспечения работоспособности устройства в заданном диапазоне внешних условий (напряжения питания, температуры и т. д.)?

Какие меры приняты для облегчения проверки работоспособности; наладки и ремонта устройства; защиты от неправильных действий оператора и т. д.?

По результатам обсуждения принципиальные электрические схемы корректируются.

5. **Заключение.** Тому, кто никогда не занимался разработкой электронных устройств, многие утверждения предлагаемой статьи покажутся очевидными. В самом деле, например, предложение снизить потребляемую мощность, отнюдь не ново, однако, основная элементная база для построения цифровых устройств — по-прежнему микросхемы серии К155 (133)...

Телефон для справок: 254-70-68 (Ленинград).

ЛИТЕРАТУРА

1. Лавриненко В. Ю. Справочник по полупроводниковым приборам.— Киев: Техника, 1984.
2. Воробьев Н. В., Вернер В. Д. Микропроцессоры. Элементная база и схемотехника средств сопряжения.— М.: Высшая школа, 1984.
3. Корольков А. А., Раденко М. Е., Сеньков В. К. Применение БИС КР580ВВ51 для реализации последовательных интерфейсов микропроцессорных систем // Микропроцессорные средства и системы.—1985.— № 1.— С. 82—85.
4. Хоровиц П., Хилл У. Искусство схемотехники.— М.: Мир, 1984.

Статья поступила 16 октября 1986 г.

От редакции:

Рассматриваемое письмо содержит материал, критикующий сложившуюся практику в разработке РЭА, и даже предлагающий определенные алгоритмы решений. Однако нужно прежде всего отметить относительную поверхностность предложений. Список такого типа можно было бы дополнить:

«Лучше при разработке использовать те ИС, которые есть у разработчика, а не самые лучшие и современные, но которые достать невозможно» — принцип журавля в небе.

«Принцип экономии жизни» — если необходимо что-либо разработать, не стремись к совершенству — для создания самого совершенного устройства, превосходящего все аналоги одновременно, не хватит жизни.

«Будь немного радиолюбителем» — именно радиолюбитель от бедности разрабатывает схемы, в которых компоненты выполняют максимум функций одновременно. Читай наши и зарубежные любительские журналы.

«Вовремя остановись» — если осваивать всю существующую элементную базу для разработки аппаратуры не останется времени. Начинай осваивать «экзотические» БИС только тогда, когда поймешь, что их применение позволит решить задачу на порядок более просто.

«Экономь»: линии связи, если их много (применяя последовательный код); карандаши — старайся сделать схему проще и компактнее, чтобы чертить ее было бы проще.

Д. А. Лукьянов

УДК 681.327

О. Б. Малежин. Н. О. Крыликов, Д. Л. Преснухин

ИНТЕРФЕЙС ПАРАЛЛЕЛЬНОГО ВВОДА-ВЫВОДА МИКРОЭВМ «ЭЛЕКТРОНИКА 60» НА ОСНОВЕ БИС СЕРИЙ К1801, К588

Для подключения внешних устройств к магистральному параллельному интерфейсу (МПИ) по ОСТ 11.305.903-80 [1] часто используется устройство параллельного обмена И2 15КС-180-032 [2] или устройства, разработанные пользователем на микросхемах средней степени интеграции [3, 4].

При большом количестве внешних устройств (например, в управляющих системах) применение параллельных интерфейсов на микросхемах большой степени интеграции дает значительный выигрыш в объеме аппаратуры.

Например, с использованием интерфейса параллельного ввода-вывода, реализованного на БИС серии К1801 (рис. 1), возможно построение устройств связи с ВУ различной конфигурации. Интерфейс предназначен для записи-считывания по 16-разрядным раздельным каналам с формированием соответствующих управляющих сигналов.

Адреса регистров состояний, источника и приемника ВУ задаются с помощью переключателей SA1. Многофункциональные устройства параллельного интерфейса D10, D11 настраиваются на режим передачи информации подключением выводов RC0, RC1. Направление передачи информации через магистральные приемопередатчики D1...D4 определяется сигналами, формируемыми микросхемой D9.

Логика работы интерфейса определяется сигналами МПИ, поступающими через приемопередатчики D5..., ...D7 на управляющие входы контроллера интерфейса D8. Сигналы ТРЕБОВАНИЕ А и ТРЕБОВАНИЕ Б (RQB) устанавливаются устройством пользователя и применяются в качестве флагов требования прерывания или флагов, состояния которых могут быть проверены программно. Сигналы PCO0 (CSR0) и PCO1 (CSR1) — разряды регистра состояний — могут использоваться для имитации запросов прерывания ВУ в режиме автономной проверки.

Сигналы ВВОД ДАННЫХ (DTR) и ВЫВОД ДАННЫХ (NDR) вырабатываются контроллером при чтении регистра приемника и записи в регистр источника соответственно и служат для стробирования данных ВУ. С помощью RC-цепи между выводами RD0 и RD1 можно увеличивать длительность указанных сигналов. По сигналам B0R и B1R происходит запись младшего и старшего байтов в регистр младшего байта D10 и регистр старшего байта D11 соответственно. Сигнал ORR вырабатывается контроллером при чтении регистра источника.

Информационные и управляющие сигналы для связи с ВУ буферизуется с помощью микросхем D12...D18. Для построения устройств с несколькими 16-разрядными интерфейсами параллельного ввода-вывода на каждый дополнительный интерфейс требуется одна БИС К1801ВП1-033, две БИС К1801ВП1-034, группа буферных элементов.

Функциональные возможности интерфейсных схем серии К588 позволяют строить на их основе устройства различной конфигурации. Схема одного из вариантов интерфейса параллельного ввода-вывода, позво-

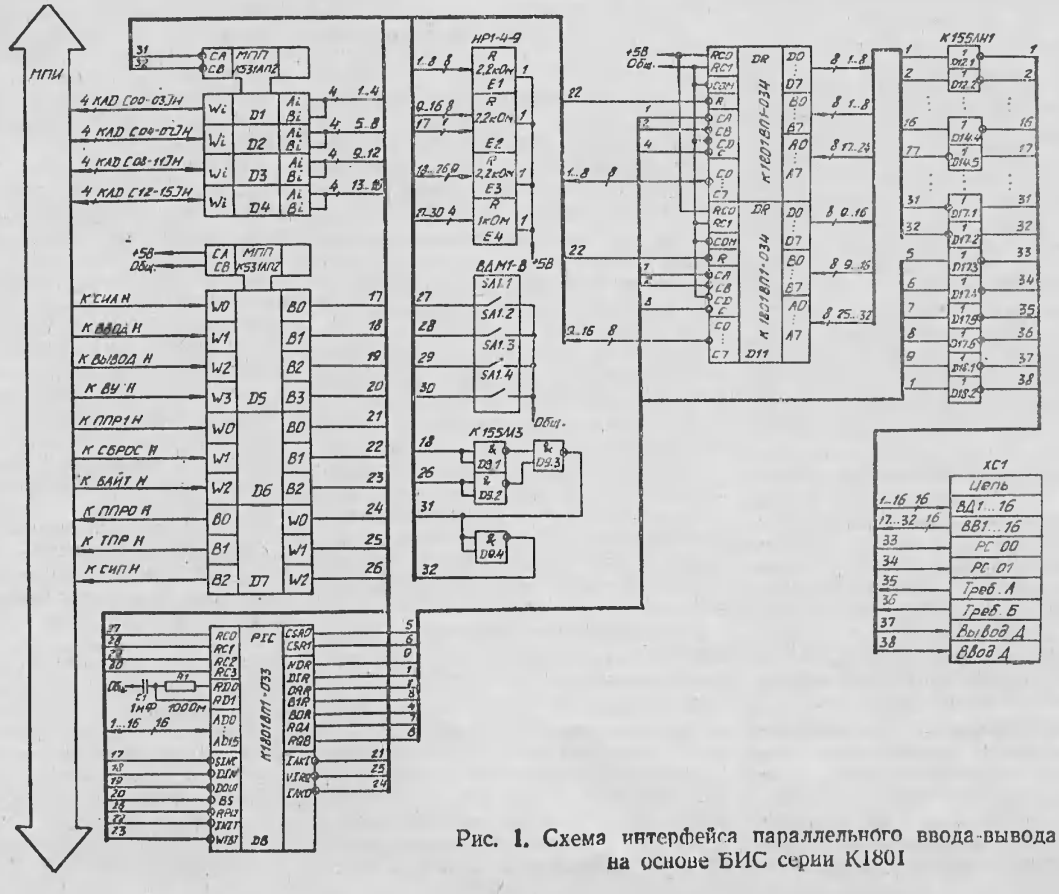


Рис. 1. Схема интерфейса параллельного ввода-вывода на основе БИС серии К1801

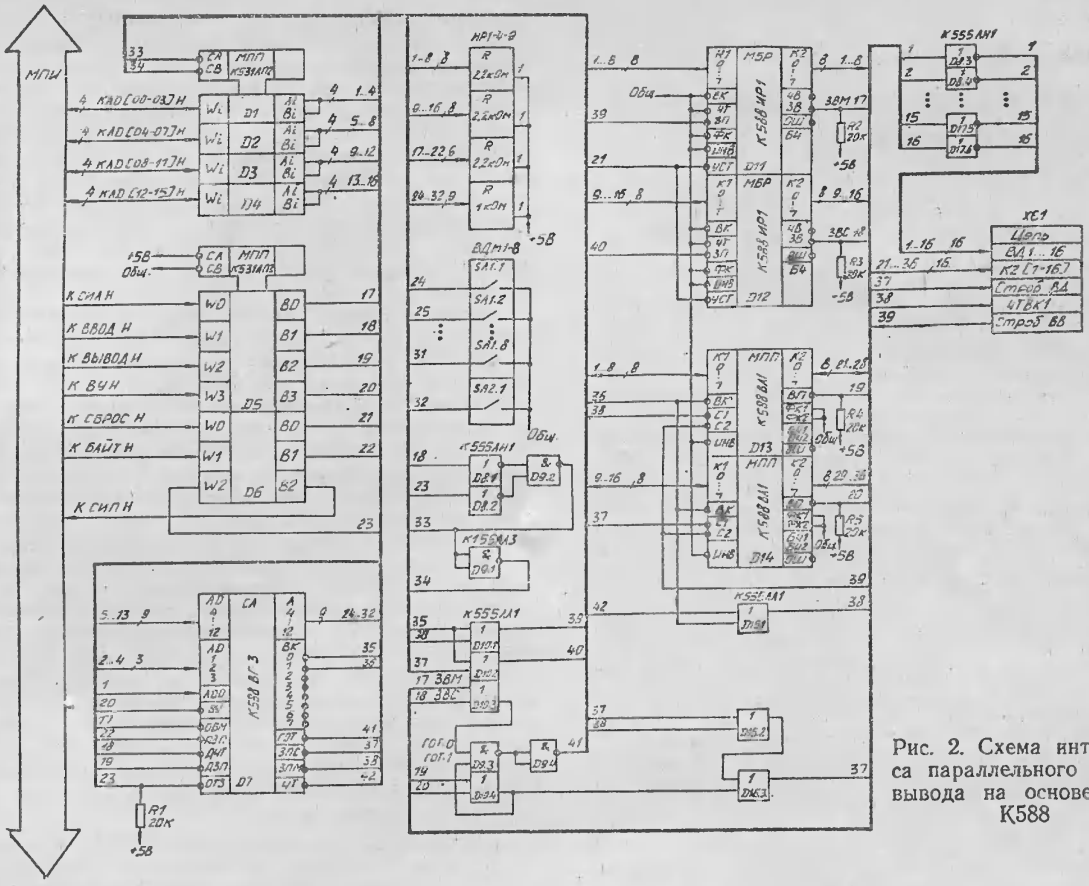


Рис. 2. Схема интерфейса параллельного ввода-вывода на основе БИС К588

ляющая организовать вывод 16-разрядного слова и двунаправленный ввод-вывод 16-разрядных слов, представлена на рис. 2.

Адрес области памяти регистров ВУ устанавливается переключателями SA1, SA2. Направление передачи информации через приемопередатчики D1...D4 определяется сигналами с выходов вентилей D9.1, D9.2, а логика работы интерфейса — сигналами МПИ, проходящими через приемопередатчики D5, D6, аналогично схеме, приведенной на рис. 1.

БИС селектора адреса D7 сравнивает поступающий на входы AD4...12 адрес с адресом, установленным на входах A4...12, дешифрует разряды AD1...3 и, в соответствии с этим, выбирает один из восьми регистров ВУ посредством одного из сигналов ВК0... ...ВК7.

В зависимости от внешних сигналов МПИ селектор адреса формирует сигналы для организации циклов записи (ЗПМ — запись младшего байта, ЗПС — запись старшего байта) или чтения — сигнал ЧТ.

Вход ГОТ является входом сигнала ответа ВУ. Многофункциональные буферные регистры D11, D12 включены таким образом (входы ВК и ЧТ подключены к шине «Лог. 0»), что информация на выходах канала К2 остается неизменной до следующей записи в регистры по сигналам с выходов вентилей D10.1 и D10.2. Наличие уровней «Лог. 0» на выходах ЗВ (сигналы ЗВМ и ЗВС) свидетельствует о том, что информация записана в регистры.

Магистральные приемопередатчики D13, D14 включаются в работу при подаче на входы ВК уровня «Лог. 0» с выхода ВК1 БИС D7. Передача информации из канала К1 в канал К2 происходит при подаче на входы С1 сигналов селектора адреса ЗПМ и ЗПС. По сигналам ВП, означающим что передача по одному из направлений (К1 → К2 или К2 → К1) выполнена, на выходе вентиля D10.4 формируется сигнал ответа ВУ ГОТ. 1. Информация, передаваемая из канала К2 в ВУ, сопровождается стробом (сигнал СТРОБ ВД). Передача информации из канала К2 в канал К1 осуществляется при выдаче селектором адреса сигнала ЧТ. При этом формируется сигнал ЧТ ВК1, означающий, что ВУ по адресу, соответствующему сигналу ВК1, должно выставить данные. Внешнее устройство устанавливает данные, сопровождаемые стробом (сигнал СТРОБ ВВ), по которому происходит передача информации из канала К2 в канал К1.

Предлагаемую схему можно наращивать подключением шести дополнительных регистров, адресуемых сигналами ВК2...ВК7 и реализуемых на рассмотренных или других БИС серии К588.

Справки по телефону: 534-54-71, Москва

ЛИТЕРАТУРА

1. ОСТ 11.305.903—80. МСВТ. Технические средства. Интерфейс межмодульный. Техническое описание.
2. Устройство параллельного обмена И2 15КС-180-032. Техническое описание и инструкция по эксплуатации. 3.858.383. ТО. 1980.
3. Горбачев С. Ф., Демин А. П. Оперативное запоминающее устройство с внешним скоростным каналом ввода-вывода информации в микроЭВМ «Электроника 60». // Микропроцессорные средства и системы.— 1986.— № 3.— С. 64—66.
4. Тарутин О. Б. Интерфейс четырех внешних устройств стандарта ИРПР к микроЭВМ «Электроника 60». // Микропроцессорные средства и системы.— 1986.— № 5.— С. 23—24.

Статья поступила 12 марта 1987 г.

УДК 681.325

А. П. Дианов, Н. Н. Щелкунов

СИСТЕМА ПРОЕКТИРОВАНИЯ МИКРОПРОЦЕССОРНЫХ УСТРОЙСТВ

(Продолжение цикла. Начало см. в №№ 5, 6 за 1986 г., №№ 1, 2, 3, 4 за 1987 г.)

Системы проектирования микропроцессорных устройств — новый вид инструментальных средств автоматизации процесса разработки. Они отличаются универсальностью, дешевизной, малогабаритностью и простотой в использовании, что обеспечивает их широкое применение в повседневной инженерной практике.

Система представляет собой обязательный базовый комплект устройств (рис. 1), расширяемых факультативными средствами: программирующими модулями, внутрисхемными эмуляторами и исполнительными процессорами. Расширение реализуется через три стандартных интерфейса, один из которых разработан специально для нужд проектирования.

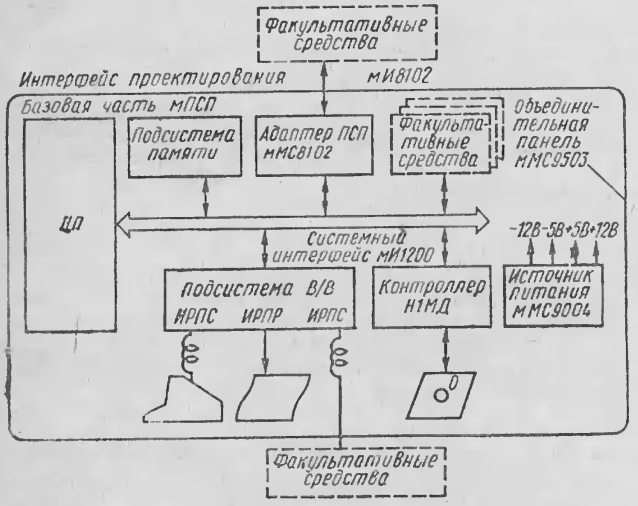


Рис. 1. Структурная схема мПСП

Система проектирования (СП) — недорогая компактная система, интегрированная специально для инструментальной поддержки микропроцессорных средств и изделий на их основе. Пример СП — система iPDS фирмы Intel [1], обеспечивающая высокоэффективную поддержку фазы разработки прикладных МС, промышленное тестирование изделий в процессе их производства и техническое обслуживание во время их эксплуатации. Система достаточно компактна и проста в обращении, ее основное назначение — массовое повышение производительности труда инженеров, занятых в сфере разработки, производства и использования микропроцессорных средств и систем.

Инструментальные пакеты обеспечивают ускорение процесса проектирования ПО, **средства внутрисхемной эмуляции** (СВЭ) решают проблему испытательного прогона прикладного ПО на целевой аппаратуре и их совместную отладку. (Пример пакета разработки ПО — набор инструментальных средств по подготовке программ для микросхем на базе МП К1810ВМ86 [2].) Апробированный с помощью СВЭ в режиме реального времени программный код записывается в ПЗУ с помощью входящих в состав СП средств занесения информации (программаторов) [3, 4].

Программные имитаторы и исполнительные пакеты [2,5] — более дешевые средства для испытательного прогона прикладного ПО, чем СВЭ, обладающие многими чертами, присущими методу внутрисхемной эмуляции. Однако в программных имитаторах отсутствует стыковка с аппаратурой проектируемого изделия, а исполнительным пакетам необходимо отлаженное и работоспособное ядро, способное поддерживать резидентную программу системного монитора-отладчика.

Система построена по магистрально-модульному принципу с использованием внутрисистемного интерфейса мИ1200, формируемого одноплатными 8-разрядными микроЭВМ семейства мМС1200 [6, 7].

Достоинство СП — наличие встроенного специализированного интерфейса проектирования мИ8102, служащего для подключения к системе аппаратных средств эмуляции, персональных модулей программирования, логических анализаторов, пошаговых отладчиков и т. д. При необходимости базовые системные средства СП дополняются модулями факультативного расширения из семейства мМС, ориентированными на системную магистраль мИ1200. Зарезервированные для этой цели места объединительной панели мМС9503 могут быть использованы для установки факультативных средств прикладного проектирования и тестирования, которые уже разработаны. Свободный от системных функций стандартный последовательный канал ИРПС резервируется для дистанционного подключения средств проектирования. Типовым вариантом его использования служит организация доступа целевых микросистем к ресурсам и файлам СП, например, с помощью исполнительных пакетов [2, 5].

Построение СП подобно системе iPDS, в состав которой введен специальный интерфейс проектирования iPDX [1]. Архитектура интерфейса отличается от организации мИ8102; функциональное назначение одно и то же. В отличие от внутрисистемных интерфейсов эти специальные шины предусматривают ряд стробируемых линий, величины напряжений на которых управляются программным способом (что очень важно при построении программаторов). Интерфейс проектирования следует рассматривать как специальное расширение системного интерфейса, упрощающее задачу построения средств проектирования.

Специальный шинный адаптер мМС8001 согласовывает системную магистраль И41 комплекса СМ1800 с шиной мИ1200, обеспечивая возможность расширения ресурсов СМ1800 модулями семейства мМС. Предусматриваются объединительные панели на 24 (мМС9501) или 8 мест (мМС9503), комбинируемые с источниками питания встроенного (мМС9004, мМС9001) или автономного типа (мМС9003) [7]. Для реализации последовательного интерфейса, резервируемого под факультативные средства проектирования, введен 3-канальный ИРПС-адаптер мМС4501. Адаптер мМС8102 может быть подключен непосредственно на системную магистраль СМ1800.

Построение СП на базе типового микровычислительного комплекса (МВК) СМ1800 (рис. 2) позволяет ускорить процесс разработки СП с последующим переходом на более компактные и дешевые структуры (см. рис. 1) при массовом внедрении системы. Модули семейства мМС и системы на их основе были отлажены с помощью СП на базе МВК СМ1800. Семейство получило достаточное развитие и появилась возможность компоновать автономные системы. Представленный на рис. 2 комплекс может рассматриваться как более мощная и дорогая профессиональная система проектирования, имеющая такое же отношение к мПСП (см. рис. 1), как система iMDS (Intellec) к iPDS.

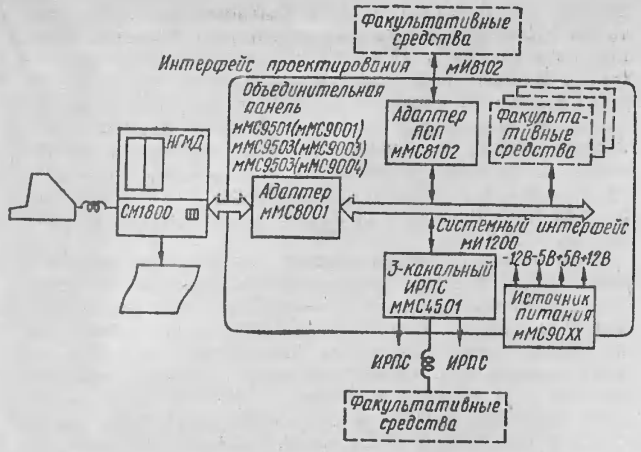


Рис. 2. Система проектирования мМС0603 на базе СМ1800

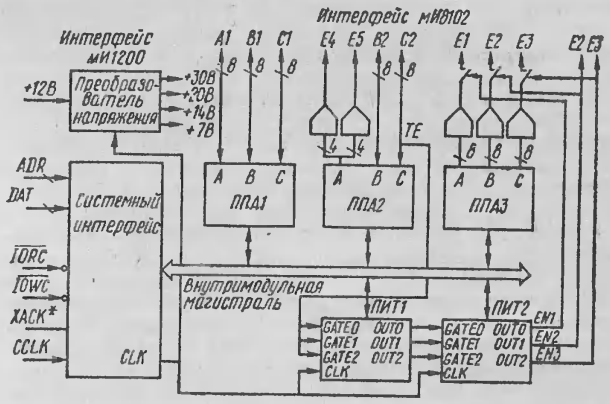


Рис. 3. Структурная схема адаптера ПСП мМС8102

Основа программного обеспечения системы проектирования — специально разработанная операционная среда ДОС1800 или ISIS-II с экранным редактором EDIT80. Написаны инструментальные программные пакеты, работающие в рамках данной ОС. В настоящее время СП состоит из четырех пакетов:

Пакет подготовки ПО для микропроцессора КР580ВМ80/ВМ85, в который входят: FORT80 — транслятор с языка Фортран, BASIC — интерпретатор с языка Бейсик, PLM80 — транслятор с языка PL/M80, ASM80 — макроассемблер, LINK — программа разрешения внешних ссылок, LOCATE — программа настройки на абсолютные адреса, LIB — библиотекарь, OBJHEX — преобразователь абсолютного файла в шестнадцатеричный формат, HEXOBJ — преобразование шестнадцатеричного файла в абсолютный формат.

Пакет разработки программ для микропроцессора К1810ВМ86/ВМ88 [2], состоящий из: PLM86 — транслятора с языка PL/M 86, ASM 86 — макроассемблера, CONV 86 — преобразователя ASM 80 в ASM 86, LINK 86 — программы разрешения внешних ссылок, LOC 86 — программы настройки на абсолютные адреса, LIB 86 — библиотекаря, OH 86 — преобразователя объектного файла в шестнадцатеричный формат.

В составе пакета подготовки программ для однокристального микропроцессора К1816ВЕ48 находится всего лишь одна программа: ASM 48 — макроассемблер.

Пакет для подготовки прикладного ПО микроконтроллеров семейства К1816ВЕ51 включает: PLM 51 — транслятор с языка PL/M 51, ASM 51 — макроассемблер, CONV 51 — преобразователь ASM 48 в ASM 51, RL 51 — программа для разрешения внешних ссылок и настройки на абсолютные адреса, LIB 51 — библиотекарь.

Коренное **отличие** СП от МВК общего назначения — наличие специализированного интерфейса проектирования мИ8102, формируемого с помощью адаптера мМС8102 (рис. 3). В его состав входят: логика системного интерфейса, три программируемых периферийных адаптера (ППА) с пятью программируемыми источниками питания (ПИП) на их выходах, преобразователь напряжения для питания ПИП и два программируемых интервальных таймера (ПИТ), управляющих ключами на выходах ПИП. Все приборы и устройства соединяются внутримодульной магистралью типа Microbus, формируемой логикой системного интерфейса.

Управляемый адаптером интерфейс проектирования содержит пять 8-разрядных шин ввода-вывода A1, B1, C1, B2 и C2 (которые запрограммированы либо на ввод, либо на вывод) и пять выходных линий с независимо программируемым напряжением питания (0 ... 25,5 В для E1 ... E3 и 0 ... 15 В для E4, E5). Выходы трех источников E1 ... E3 стробируются программно-управляемыми сигналами разрешения EN1 ... EN3, два из которых включены в состав интерфейса мИ8102, максимальный ток нагрузки ПИП в импульсном режиме равен 1 А.

Программно-доступные регистры ППА размещаются в пространстве портов ввода-вывода и занимают область 20 байт. Конкретные физические адреса портов, принятые в адаптере мМС8102, и рекомендуемое для пользователя их функциональное назначение указаны на рис. 4. Восьмиразрядная шина A1 резервируется для

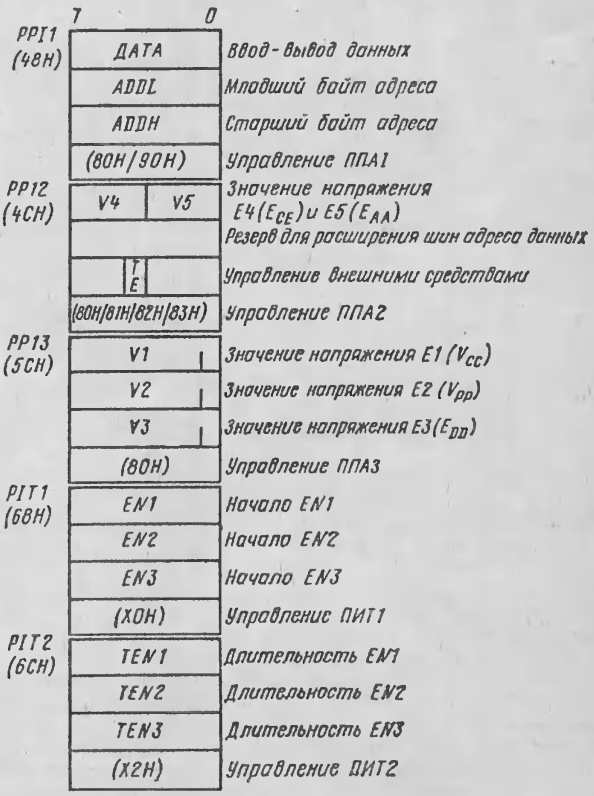


Рис. 4, Программно-доступные регистры адаптера ПСП мМС8102

ввода-вывода данных, шины C1 и B1 предназначены для вывода адресной информации. Канал C2 отводится под управление внешними средствами и может рассматриваться как два независимых 4-разрядных канала C27 ... C24 и C23 ... C20. Одна из линий шины C24 (TE) несет определенную функциональную нагрузку по управлению счетом ПИТ1. При TE-1 счет ПИТ1 разрешен, в противном случае — запрещен. Резервный канал B2 используется при расширении разрядности внешних шин адреса данных, управления или состояния.

Порты A3, B3, C3 и A2 резервируются для программирования величины напряжений E1 ... E5. Под каждый источник E1 ... E3 отводится 8-разрядный порт A3 ... C3, что дает возможность управлять напряжением с точностью до 0,1 В. Для дополнительных источников E4 и E5 отводится по 4 разряда порта A2, обеспечивающих более грубый шаг изменения напряжения, равный 1 В. Рекомендуемое назначение источников следующее: E1 — напряжение V_{cc}, E2 — напряжение V_{pp} при программировании ПЗУ [8], E3 — напряжение на шине данных, E4 — напряжение на входе CE, E5 — напряжение на шине адреса при программировании логических матриц [9].

Шесть портов SEN1 ... SEN3 и TEN1 ... TEN3 управляют началом и длительностью стробов EN1 ... EN3, открывающих выходные ключи ПИП1 ... ПИП3. С этой целью первый интервальный таймер программируется для работы в режиме 0 (прерывание по окончанию счета), а второй — в режиме 1 (одновибратор импульса программируемой длительности). После загрузки необходимых значений в счетчики ПИТ1 и ПИТ2 первый таймер отпускается сигналом TE-1. По окончании счета на соответствующем выходе таймера появится перепад из «Лог. 0» в «Лог. 1», который запустит связанный с данным выходом счетчик второго таймера. На выходе второго счетчика сформируется инверсный импульс, открывающий выход источника питания. Подобная логика генерации позволяет сформировать систему из трех произвольно расположенных друг к другу импульсов, начало и длительность которых определяются программным способом с дискретом в один период CCLK.

Оставшиеся 5 портов служат для управления режимами ППА и ПИТ, рекомендуемое содержание которых приведено на рис. 4 (X — произвольная шестнадцатеричная цифра). Интерфейсная логика, служащая для преобразования системного интерфейса мИ1200 во внутримодульную магистраль типа Microbus, реализована на двух шинных драйверах К589АП16 (DD4, DD5), ПЗУ К556РТ5 (DD6) и ряде логических схем малой степени интеграции (DD1, DD2, DD3, DD10) (рис. 5). Интерфейсная логика возвращает инверсный сигнал подтверждения обмена ХАСК, необходимый в СП. На базе ПЗУ К556РТ5 реализована логика выборки кристаллов. Карта прошивки ПЗУ, соответствующая распределению адресов, приведена на рис. 6. Старшая часть микросхемы не используется.

Основу программируемых источников питания составляет проверенная на практике схема, содержащая цифро-аналоговый преобразователь К572ПА1, быстродействующий операционный усилитель К544УД2 и выходной буфер на трех транзисторах КТ3102, КТ909, КТ203

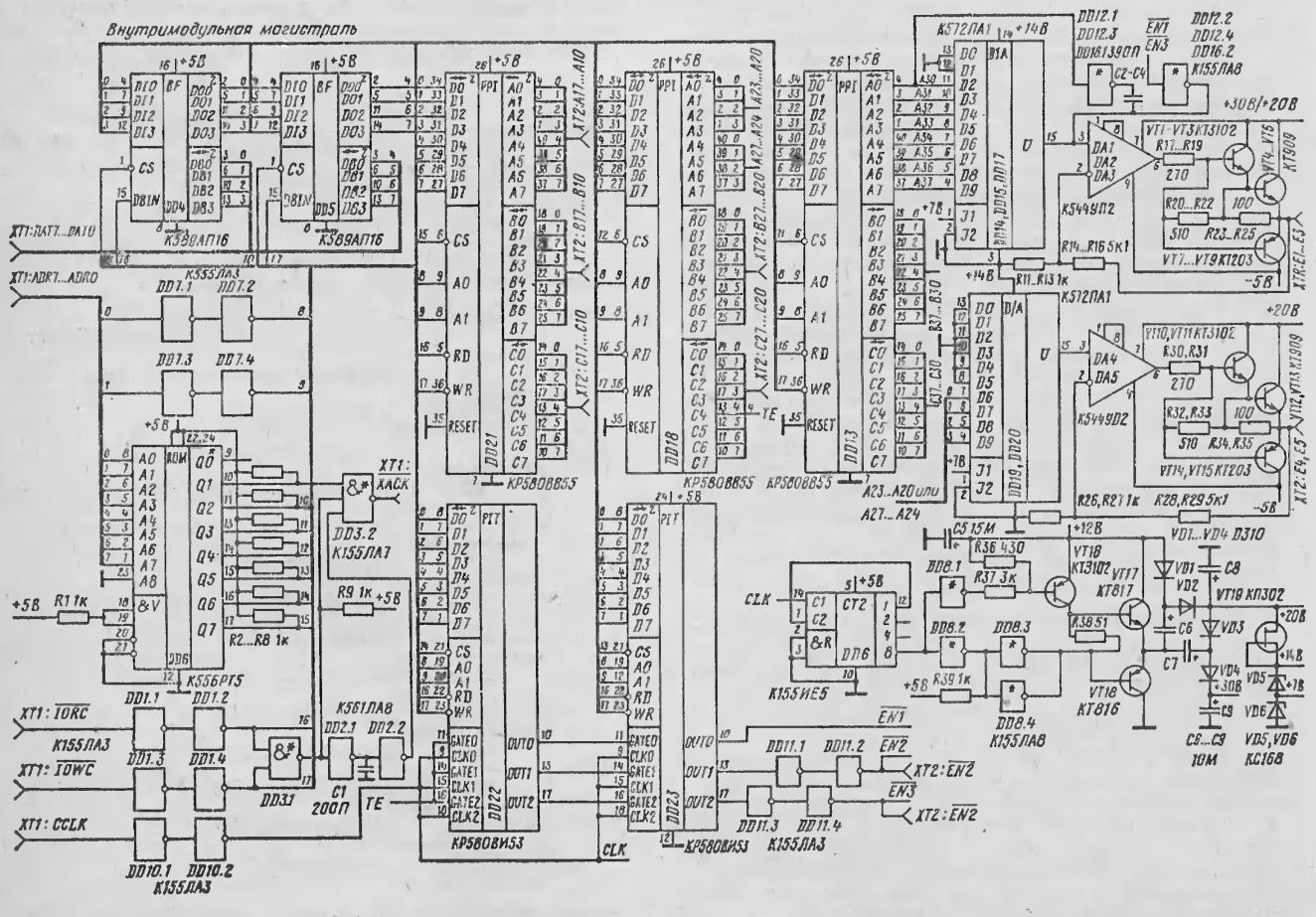


Рис. 5. Принципиальная схема адаптера ПСП мМС8102

.	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
.000.	FD	FD	FD	FD	FD	FD	FD	FD	FD	FD	FD	FD	FD	FD	FD	FD	.
.010.	FD	FD	FD	FD	FD	FD	FD	FD	FD	FD	FD	FD	FD	FD	FD	FD	.
.020.	FD	FD	FD	FD	FD	FD	FD	FD	FD	FD	FD	FD	FD	FD	FD	FD	.
.030.	FD	FD	FD	FD	FD	FD	FD	FD	FD	FD	FD	FD	FD	FD	FD	FD	.
.040.	FD	FD	FD	FD	FD	FD	FD	FD	7B	7B	7B	7B	EB	EB	EB	EB	.
.050.	FD	FD	FD	FD	FD	FD	FD	FD	FD	FD	FD	FD	F3	F3	F3	F3	.
.060.	FD	FD	FD	FD	FD	FD	FD	FD	BB	BB	BB	BB	DB	DB	DB	DB	.
.070.	FD	FD	FD	FD	FD	FD	FD	FD	FD	FD	FD	FD	FD	FD	FD	FD	.
.080.	FD	FD	FD	FD	FD	FD	FD	FD	FD	FD	FD	FD	FD	FD	FD	FD	.
.090.	FD	FD	FD	FD	FD	FD	FD	FD	FD	FD	FD	FD	FD	FD	FD	FD	.
.0A0.	FD	FD	FD	FD	FD	FD	FD	FD	FD	FD	FD	FD	FD	FD	FD	FD	.
.0B0.	FD	FD	FD	FD	FD	FD	FD	FD	FD	FD	FD	FD	FD	FD	FD	FD	.
.0C0.	FD	FD	FD	FD	FD	FD	FD	FD	FD	FD	FD	FD	FD	FD	FD	FD	.
.0D0.	FD	FD	FD	FD	FD	FD	FD	FD	FD	FD	FD	FD	FD	FD	FD	FD	.
.0E0.	FD	FD	FD	FD	FD	FD	FD	FD	FD	FD	FD	FD	FD	FD	FD	FD	.
.0F0.	FD	FD	FD	FD	FD	FD	FD	FD	FD	FD	FD	FD	FD	FD	FD	FD	.

Рис. 6. Карта прожига логики выборки кристаллов

[3]. Для уменьшения габаритов реализации введены небольшие изменения. Управление скоростью нарастания выходного напряжения E1 ... E3 выполняется с помощью младшего разряда информационной шины, выключение выходного напряжения реализуется программным и аппаратным способами. Возможность аппаратного выключения выходного напряжения и управление скоростью нарастания в ПИП4 и ПИП5 отсутствуют. Для получения необходимых опорных напряжений ПИП служит встроенный в модуль преобразователь напряжения. Все линии интерфейса выведены на отдельный разъем XT2 с противоположной по отношению к XT1 стороны платы.

Справки по телефону: 408-62 44, Москва.

ЛИТЕРАТУРА

1. Development System Hardbook, Order Number: 210940—003. Intel Corp. USA, 1985.
2. Щелкунов Н. Н., Дианов А. П. Техника программирования 16-разрядных микроконтроллеров // Микропроцессорные средства и системы.— 1987.— № 2.— С. 11—14.
3. Дианов А. П., Щелкунов Н. Н. Модуль программирования микросхем ПЗУ // Микропроцессорные средства и системы.— 1985.— № 3.— С. 80—83.
4. Дианов А. П., Щелкунов Н. Н. Технические средства программирования логических схем // Микропроцессорные средства и системы.— 1986.— № 2.— С. 77—80.
5. Щелкунов Н. Н., Дианов А. П. Техника программирования 8-разрядных микроконтроллеров // Микропроцессорные средства и системы.— 1986.— № 6.—С. 23—28.
6. Щелкунов Н. Н., Дианов А. П. Универсальный одноплатный микроконтроллер // Микропроцессорные средства и системы.— 1986.— № 5.— С. 65—69.
7. Дианов А. П., Щелкунов Н. Н. Малогабаритные источники питания для микросистем // Микропроцессорные средства и системы.— 1987.— № 3.— С. 73—76.
8. Дианов А. П., Щелкунов Н. Н. Методика программирования микросхем ПЗУ // Микропроцессорные средства и системы.— 1985.— № 3.— С. 75—79.
9. Щелкунов Н. Н., Дианов А. П. Процедуры программирования логических матриц // Микропроцессорные средства и системы.— 1986.— № 2.—

Статья поступила 21 ноября 1986 г.

(Окончание. Начало см. на с. 74).

два часа, сортирует их четыре часа или как ОКА под информацию на 3000 человек занимает весь диск. ПС снимут с нас рутинные программы ввода, слияния—корректировки, печати по ограничениям и произвольного поиска в последовательном наборе данных. Мы сразу решили ввести в дело дисплей и многие функции сделать диалоговыми. Может это и не ново, не знаю.

Год думали, год делали (вдвоем). Ничего особенно трудного в этой работе не было. Хорошо пошла.

Сейчас работу от создания макета ввода в НД до набивки информации с дисплея, ввода ее в НД и печати НД по любым ограничениям реквизитов можно проделать часа за два, причем в большинстве случаев (если нет особо сложных контролей), это под силу и не программисту. Кстати, одну задачу, за которую заплатили «варягам» 20 000 руб. и она не работала, сделали за 2 недели вдвоем.

Получилась, как Вы понимаете, система ведения линейных файлов (это тоже не от любви к базам). Мы решили, что это будут программы расчетов, преобразования и информации, какие-то сложные печати и видеограммы. А все остальное легко формализуется через таблицу описания структуры записи НД. Если добавить, что мы разработали метод произвольного извлечения информации из последовательного НД, который по характеристикам времени доступа и размещения НД на диске лучше индексно-последовательного и прямого, то станет понятно, что линейные файлы нас вполне удовлетворяют.

Дополнительно к тому, что у нас уже есть, решили приобрести какие-нибудь ППП универсального вывода на печать и дисплей, чтобы хоть их самим не делать. В центре обслуживания ничего практически нам не посоветовали. Нет у них четкого деления ППП по назначению Сейчас будем экспериментировать с «Телесправкой» и «Марс—Проба—Комплекс». Они нам их могут продать, но вышли мы на эти ППП сами (опять же по слухам: что-то где-то на них работает). Поговорили мы с людьми, которые внедряют ДИСОД (по их словам — расширение «Спектра»). Раньше я думал (это же с точки зрения здравого смысла!), что базу данных придумали, чтобы в основном, избавиться от программирования, или хотя бы понизить уровень необходимого программирования. А выяснилось, что ввода нет, универсального вывода нет. А что есть? Есть хранение и улучшенная корректировка, есть язык манипулирования данными. Но если я должен писать программы ввода, контроля, печати, то неужели я сам в конкретной программе не смогу связать файлы и достать из конкретного файла конкретную информацию. Вы можете подумать, что я мало знаю о хороших сторонах баз данных. Может быть, но то что я знаю (наложение связей между данными, язык запросов), по-моему не искупает тех недостатков. Видимо, глупо делать базу данных только для хранения информации, без универсального ввода и вывода. И это, как говорят, неплохая база (ДИСОД).

Так зачем эти 1000 единиц ПС в ФАПе. Если я не прав, то значит, я просто чего-то не знаю, и видимо, в этом не только моя вина. Я думаю Вы поняли, что я самый средний, а значит таких как я тысячи и АСУ внедряют именно они, а потом плачем, что АСУ не дает эффекта. Или надо забрать с заводов функции проектирования АСУ, или надо говорить об их проблемах и помогать. На ВЦ, где нет достаточного количества достаточно опытных программистов, можно применять готовые системы типа «Бухгалтерский учет» или «Основные фонды» (которые, кстати, должны адаптировать авторы ППП, причем, так сказать «до победы», до результата, что весьма редко). Если же на ВЦ есть программисты, способные написать конкретные для данного предприятия программы, то им надо дать или инструментальные средства поддержки типа ППП копирования (восстановления магнитных носителей, диалоговых систем, универсальной печати и вывода видеограмм). Во всяком случае, решения типа ТПР или БКТР, которые время от времени насаждаются сверху, очень трудно внедряются, потому что, кто не умеет программировать — не могут приспособить ППП к своему производству, а кто умеет — не хочет возиться с тем, что заведомо хуже своего (хотя бы в силу своей универсальности).

Кроме того, если бы одновременно с ТПР и ППП в АСУП внедрялись и типовые алгоритмы и формы в отделах управления, а то ведь на каждом заводе свое понятие о том, какие виды оплат включать в средний заработок, не говоря уже о более сложных вопросах, типа алгоритма расчета калькуляции или балансов.

В «Колонке редактора» первого номера за этот год Вы пишете об информатике, не упоминая, на какой элементной базе, на каких ЭВМ будут решаться ее задачи. Я думаю — это справедливо. Но тогда почему же здесь номер посвящен ПЭВМ. Да, название обязывает, это интересно, здесь будущее. Я сам выписываю «МП» ради ПЭВМ. Несколько моих знакомых собрали самодельные ПЭВМ, я хотел бы на них поработать, так сказать, в свободное от работы время. Но где же журнал, который будет говорить о ЕС, о ВЦ и отделах АСУП на предприятиях и в институтах, о самих АСУ? А ведь именно мы на тысячах предприятиях пытаемся ввести новые «информационные технологии», или это возможно лишь на малых машинах? Тут, мне кажется, есть провал. Разве «Интегрированная система для решения прикладных задач» нужна только для ПЭВМ? Да ее с руками оторвут ВЦ, только предложи. По этому пути, мне кажется, надо идти, а не громоздить многотысячные (и по объему и по затратам) базы данных, да еще и каждый свою.

Вот, видимо, все, что я хотел Вам сказать. Таковы проблемы типичного программиста типичного завода. Такое впечатление, что о них никто не знает. Во всяком случае у нас в городе несколько подобных ВЦ и все придумывают велосипед. Связей между ВЦ почти нет.

Подскажите, где найти ответы на эти вопросы.

С уважением Ю. В. Слуцкий

ПРИМЕНЕНИЕ МП-ТЕХНИКИ

Краткое сообщение

УДК 681.142:55.2

Б. Л. Ерухимов, В. Н. Черненков

ТРЕХУРОВНЕВЫЙ КОМПЛЕКС ДЛЯ АВТОМАТИЗАЦИИ НАУЧНЫХ ИССЛЕДОВАНИЙ НА РАДИОТЕЛЕСКОПЕ РАТАН 600

Конструкция радиотелескопа РАТАН 600 представляет собой кольцо антенны переменного профиля диаметром 600 м, внутри которого расположены четыре подвижных облучателя (см. вкладку). На облучателях находятся локальные информационно-управляющие структуры (ЛИУС), в задачу которых входит сбор и первичная обработка радиоастрономической информации.

Центральная ЭВМ типа СМ-4 (рис. 1) удалена от облучателей на 1,5 км (расположена в лабораторном корпусе за пределами кольца) и связана с ЛИУС по двум коаксиальным кабелям через последовательный интерфейс (5Р.036.14 и 5Р.036.15 — разработка СКБ НП АН СССР), осуществляющий обмен 16-разрядными словами со скоростью до 20 К слов/с. ЛИУС входят в многотерминальную ОС NTS на правах обычных DL-терминалов. Режим самотестирования в интерфейсах позволил стандартизовать реакцию центральной ЭВМ на состояние ЛИУС.

Для загрузки ЛИУС по линии под управлением центральной ЭВМ использовано штатное ПЗУ пультового режима процессора (команда L). Каждый комплекс ЛИУС (рис. 2) содержит две микроЭВМ «Электроника МС 1201.01», связанные через встроенные параллельные интерфейсы, интерфейс КАМАК. У каждой микроЭВМ — собственный набор внешних устройств, одна из машин — ведущая, другая — ведомая. К последовательному интерфейсу ведущей микроЭВМ подключен терминал пользователя и интерфейс связи с центральной ЭВМ.

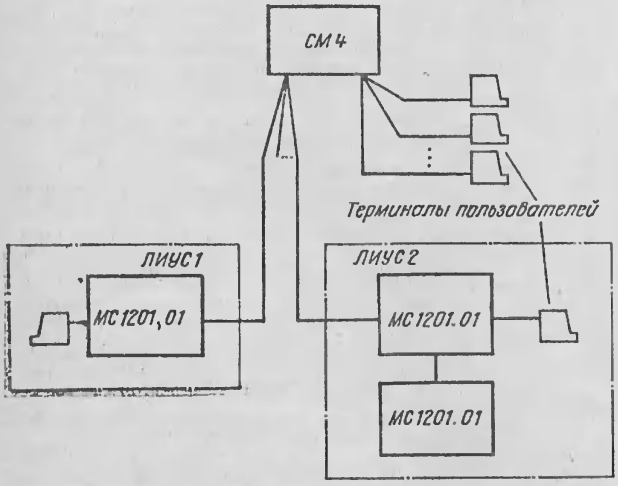


Рис. 1. Архитектура трехуровнего комплекса на РАТАН 600

Порт встроенного интерфейса ведомой микроЭВМ, используемой для межпроцессорной связи, имеет адреса стандартного терминала для реализации начальной загрузки ведомой машины с использованием стандартного пультового режима.

Комплекс работает под управлением ОС NTS и обеспечивает режимы:

эмуляции микроЭВМ терминала центральной ЭВМ на правах абонента многопользовательской системы;

компоновки автономных программных модулей ЛИУС на центральной ЭВМ в режиме эмуляции терминала с последующей загрузкой в микроЭВМ по той же линии связи (кросс-система реального времени);

автономной работы с возможностью обмена данными с центральной ЭВМ.

На центральной ЭВМ (как автономное задание) выполняется программа HMON, обслуживающая сеть ЛИУС. Она ожидает запросы, обеспечивая исполнение с разделением времени, загружает ЛИУС и выводит в режим эмуляции терминала, загружает рабочие программы, передает и принимает данные от ЛИУС с организацией файлов на дисках центральной ЭВМ, запускает на центральной ЭВМ процессы обработки (как автономные задания). Программа занимает 7К байт ОЗУ и основное время находится в режиме ожидания, поэтому сеть ЛИУС практически не влияет на эффективность работы обычных пользователей.

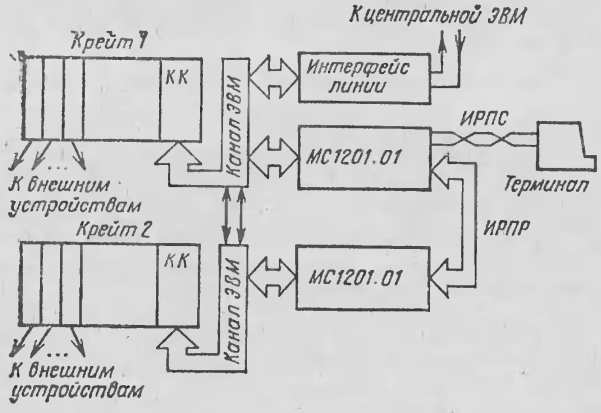


Рис. 2. Архитектура двухмашинного комплекса ЛИУС

Программа пользователя в ЛИУС может общаться с центральной ЭВМ с помощью ряда программных запросов, реализованных как функции ФОРТРАН. Например, при выполнении функции I-IFDRW («DEV: : FILE. EXT», BUFFER, NI, «DEV : CMD. COM») на устройстве DEV центральной ЭВМ открывается новый файл FILE. EXT, в который записывается NI слов из массива BUFFER ЛИУС-программы, файл закрывается. Затем в режиме DETACH-задания запускается командный файл CMD. COM.

Базовое программное обеспечение ЛИУС состоит из загрузочного модуля LE 60. LDA и объектной библиотеки LIBNMS, представляющей собой вариант объектной библиотеки ФОРТРАН, сгенерированной для микроЭВМ и содержащей дополнительные процедуры, поддерживающие программные запросы.

Адрес для справок: 357140, Ставропольский край, ст. Зеленчукская, РАТАН 600, лабораторный корпус.

Статья поступила 12 января 1986 г.

УДК 681.325.5

В. И. Кулешова

МИКРОПРОЦЕССОРНЫЙ КОМПЛЕКТ СЕРИИ КР580

Состав микропроцессорного комплекта (МПК) серии КР580 и основные технические характеристики входящих в него микросхем представлены в статье, опубликованной в предыдущем номере. Первые микросхемы этого МПК (КР580ВМ80А, КР580ВВ51А, КР580ВИ53, КР580ВТ57, КР580ВН59, КР580ВВ55А) выпускаются серийно с 1979 г. и довольно подробно описаны в литературе [1—6].

Данный материал посвящен тем микросхемам комплекта, которые непосредственно взаимодействуют с микропроцессором (МП) КР580ВМ80А, расширяют его функциональные возможности и позволяют создавать экономичные и компактные микропроцессорные системы. Кроме того, рассмотренные ниже микросхемы серии КР580 могут использоваться совместно с МПК и микросхемами других серий при условии соблюдения требований, указанных в НТД.

Микросхема КР580ГФ24 представляет собой генератор тактовых импульсов, формирующий:

высокоуровневые тактовые сигналы Ф1 и Ф2 с несовпадающими фазами;

тактовый сигнал Ф2Т, по уровню совместимый с ТТЛ-схемами и синхронизированный с сигналом Ф2;

сигнал STSTB «строб состояния», фиксирующий состояние шины данных микропроцессора;

сигнал RESET «установка».

Условное графическое обозначение микросхемы приведено на рис. 1, *а, б*, назначение выводов — в табл. 1, электрическая структурная схема микросхемы КР580ГФ24 представлена на рис. 2.

Генератор опорной частоты при подключении к выводам XTAL1 и XTAL2 кварцевого резонатора обеспечивает высокую стабильность частоты, определяемую основной частотой возбуждения кварцевого резонатора, которая не должна превышать 27 МГц. Выход генератора опорной частоты выведен на внешний вывод OSC и соединен внутри микросхемы со счетчиком-делителем, входящим в состав тактового генератора.

Тактовый генератор состоит из счетчика-делителя на 9, логических дешифраторов, формирующих требуемые тактовые импульсы, выходных формирователей, вспомогательных логических схем и триггеров для генерации выходных сигналов: Ф1, Ф2, Ф2Т. Тактовые импульсы Ф1 и Ф2 управляют МОП-входами микропроцессора КР580ВМ80А. Тактовый импульс Ф2Т используется для управления ТТЛ-входами в режиме прямого обращения к памяти.

Отрицательный сигнал STSTB, длительность которого равна одному периоду частоты опорного генератора, формируется микросхемой КР580ГФ24 при поступлении на ее вход с микропроцессора КР580ВМ80А сигнала SYNC «синхронизация», свидетельствующего о начале машинного цикла.

При поступлении входного сигнала RESIN микросхема КР580ГФ24 с помощью триггера Шмитта и триггера Т1 вырабатывает сигнал RESET,

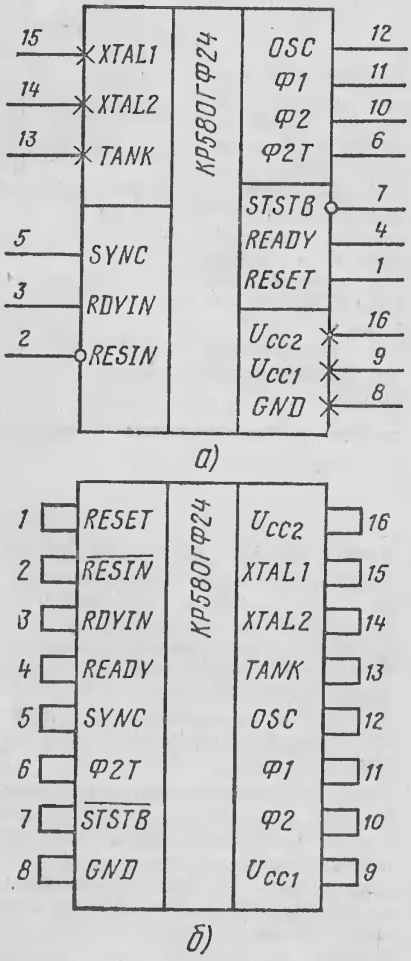


Рис. 1. Условное графическое обозначение микросхемы КР580ГФ24:

а — по функциональному назначению выводов; *б* — по порядку расположения выводов

Таблица 1

Назначение выводов микросхемы КР580ГФ24

Вывод	Обозначение	Назначение
1	RESET	Установка (выход)
2	RESIN	Установка (вход)
3	RDYIN	Готовность (вход)
4	READY	Готовность (выход)
5	SYNC	Синхронизация
6	Ф2Т	Фаза 2 с уровнем ТТЛ
7	STSTB	Строб состояния
8	GND	Общий
9	U_{CC1}	Напряжение питания 12 В
10, 11	Ф2, Ф1	Фаза 2,1
12	OSC	Выход осциллятора
13	TANK	Вход колебательного контура
14, 15	XTAL2, XTAL1	Кварцевый резонатор
16	U_{CC2}	Напряжение питания 5 В

синхронизированный с тактовым сигналом Ф2. По сигналу RESET осуществляется установка в исходное состояние различных устройств микропроцессорной системы. Наличие в микросхеме триггера Шмитта позволяет подавать на вход RESIN сигнал с пологим фронтом.

Сигнал RESET может формироваться автоматически при включении напряжения питания при условии подключения к выводу RESIN RC-цепочки, представленной на рис. 3. С помощью триггера Т2 осуществляется стробирование входного сигнала RDYIN «готовность» тактовым сигналом Ф2. Основные статические параметры микросхемы КР580ГФ24 приведены в табл. 2, динамические — в табл. 3, временны́е диаграммы функционирования — на рис. 4.

Микросхема КР580ВК28 (КР580ВК38) выполняет функции системного контроллера и шинного формирователя для МП КР580ВМ80А.

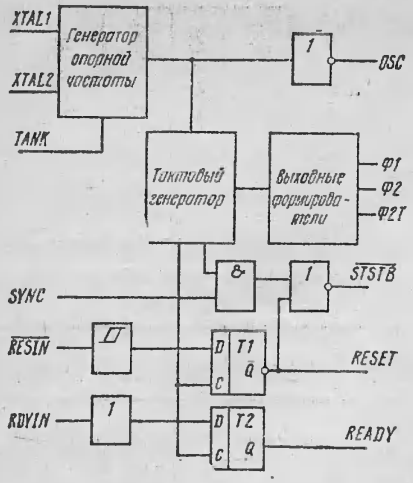


Рис. 2. Электрическая структурная схема микросхемы КР580ГФ24

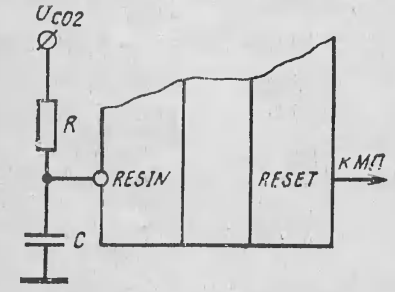


Рис. 3. Схема формирования сигнала RESET

Таблица 2

Статические параметры

Параметр, обозначение, единица измерения	КР580ГФ24	КР580ВК28, КР580ВК38	КР580ИР82, КР580ИР83	КР580ВА86	КР580ВА88
Входное напряжение низкого уровня, U_{IL}, В, не более	0,8	0,8	0,8	0,8 (А) 0,9 (В)	0,8 (А) 0,9 (В)
Входное напряжение высокого уровня, U_{IH}, В, не менее	2,0 2,6 (RESIN)	2,0	2,0	2,0	2,0
Выходное напряжение низкого уровня, U_{OL}, В, не более (при I_{OL}, мА)	0,45 (2,5) (15—Ф2Т, OSC)	0,45 (10) (2—D0—D7)	0,45 (32)	0,45 (10; А) (32; В)	0,45 (10; А) (32; В)
Выходное напряжение высокого уровня, U_{OH}, В, не менее (при I_{OH}, мА)	2,4 (—1) 3,6 (RESET, READY) (—0,1) 9,4 (Ф1, Ф2) (—0,1)	2,4 (—1) 3,6 (D0—D7) (—0,1)	2,4 (—5)	2,4 (—1; А) (—5; В)	2,4 (—1; А) (—5; В)
Входной ток низкого уровня, I_{IL}, мА, не более	0,25	0,75	0,2	0,2	0,2
Входной ток высокого уровня, I_{IH}, мкА, не более	10	100	50	50	50
Выходной ток низкого и высокого уровня в состоянии «выключено», I_{OZL}, I_{OZH}, мкА, не более	—	100	50	50	50
Ток потребления, I_{CC} (I_{CC1}/I_{CC2}), мА, не более	115 (12)	190	160	160	130

Микропроцессор в начале каждого машинного цикла выдает на шину данных байт информации (слово состояния), определяющий последующие его действия. Фиксация слова состояния и выработка необходимых управляющих сигналов обращения к ОЗУ или к устройствам ввода-вывода (УВВ) осуществляется системным контроллером. В зависимости от значений отдельных разрядов слова состояния системный контроллер осуществляет 10 режимов работы (типов машинных циклов) (табл. 4).

Условное графическое обозначение микросхемы КР580ВК28 (КР580ВК38) представлено на рис. 6 *а, б*. Назначение выводов приведено в табл. 5. Электрическая структурная схема микросхемы КР580ВК28 (КР580ВК38) представлена на рис. 7.

Отличие микросхемы КР580ВК28 от микросхемы КР580ВК38 состоит в способе формирования сигналов I/OW, MEMW. Микросхема КР580ВК28 формирует эти сигналы относительно сигнала WR «запись», а микросхема КР580ВК38 — относительно сигнала STSTB «строб состояния», что позволяет при применении в микропроцессорной системе микросхемы КР580ВК38 использовать ЗУ и УВВ с более широким диапазоном быстродействия.

Двунаправленный шинный формирователь осуществляет буферирование 8-разрядной шины данных и автоматический контроль направления передачи данных. Подключение системного контроллера к шине данных микропроцессора осуществляется с помощью двунаправленных выводов D0...D7, к системной шине — с помощью двунаправленных выводов DB0...DB7.

При необходимости с помощью сигнала BUSEN «управление системной шиной» выводы DB0...DB7 системного контроллера могут быть переведены в состояние «выключено».

Регистр состояния выполнен на шести D-триггерах и предназначен для хранения информации о состоянии микропроцессора, поступающей по шине данных D0...D7. Запись в регистр состояния осуществляется по сигналу STSTB, поступающему в начале каждого машинного цикла.

Декодирующая матрица в зависимости от режима работы микропроцессора, зафиксированного в регистре состояния, и входных управляющих сигналов HLDA, WR, DBIN формирует сигнал INTA «подтверждение прерывания» или сигналы чтения (записи) при обращении к ОЗУ или УВВ.

Основные статические параметры микросхемы КР580ВК28 (КР580ВК38) приведены в табл. 2, динамические — в табл. 6, 7, временные диаграммы функционирования — на рис. 8, 9.

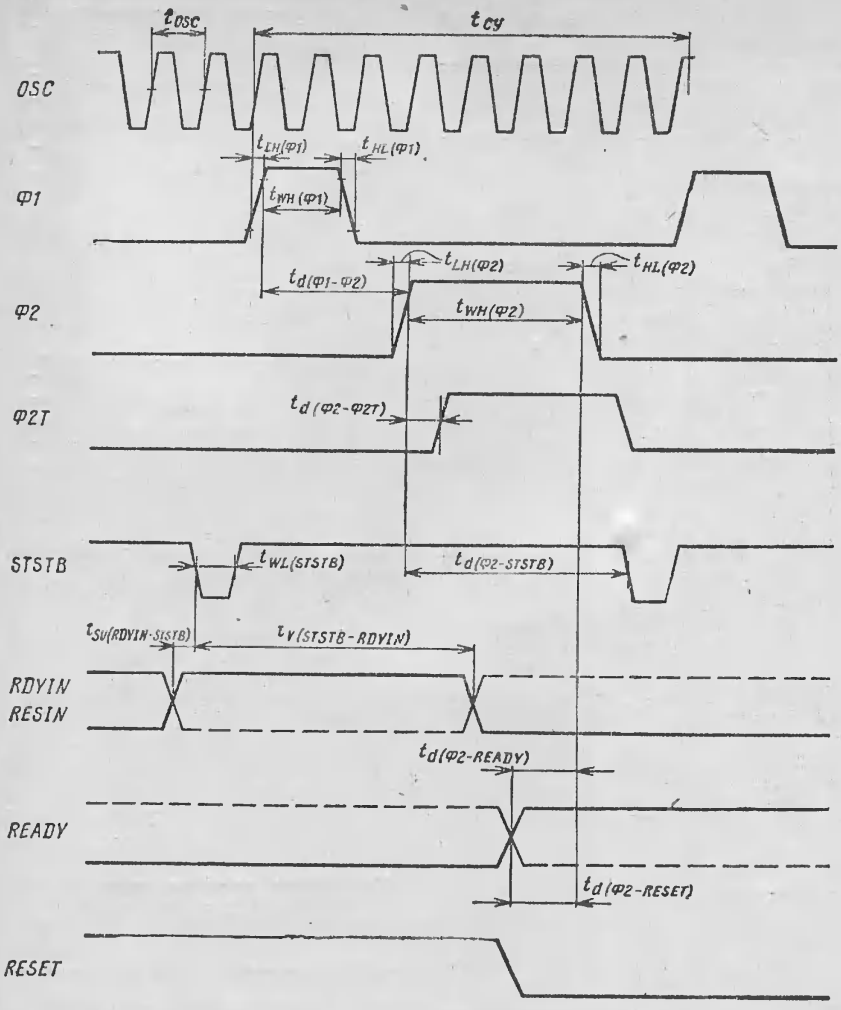
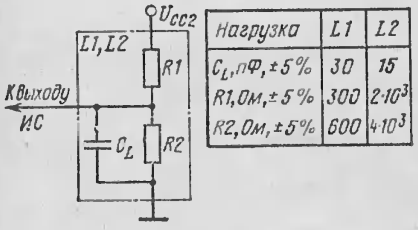


Рис. 4. Временные диаграммы функционирования микросхемы КР580ГФ24



Нагрузка	L1	L2
C_L, пФ, ±5%	30	15
R1, Ом, ±5%	300	$2 \cdot 10^3$
R2, Ом, ±5%	600	$4 \cdot 10^3$

Рис. 5. Схема подключения нагрузки при измерении динамических параметров микросхемы КР580ГФ24

Микросхема КР580ИР82 (КР580ИР83) представляет собой 8-разрядный буферный регистр, предназначенный для ввода-вывода информации со стробированием. Микросхема КР580ИР83 отличается от микросхемы КР580ИР82 тем, что имеет инвертирующие выходы. Разводка выводов обеих микросхем одинакова.

Условное графическое обозначение микросхем представлено на рис. 11 и 12, назначение выводов приведено в в табл. 8. Электрическая структурная схема микросхем КР580ИР82 (КР580ИР83) представлена на рис. 13.

Каждая из микросхем состоит из 8 триггеров D-типа и 8 выходных буферов, имеющих на выходе состояние «выключено». Управление передачей информации осуществляется с помощью сигнала STB «строб».

При поступлении на вход STB сигнала высокого уровня осуществляется нетактируемая передача информации от входа DI до выхода DO. При подаче на вход STB сигнала низкого уровня микросхема хранит информацию предыдущего такта; при подаче на вход STB положительного перепада импульса происходит «защелкивание» входной информации.

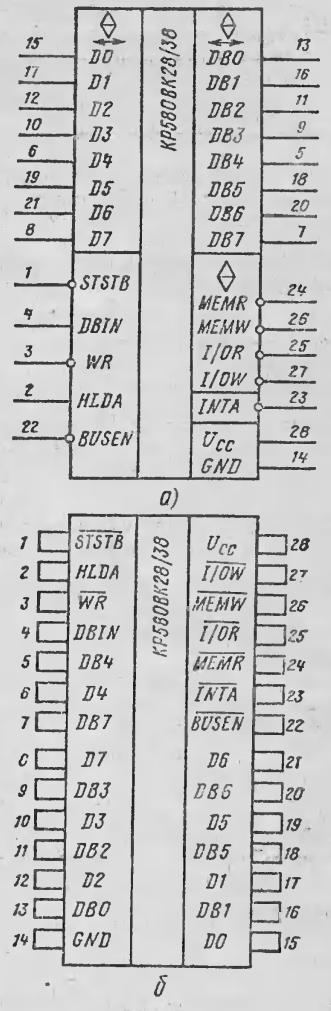


Рис. 6. Условное графическое обозначение микросхемы КР580ВК28 (КР580ВК38):

а — по функциональному назначению выводов; *б* — по порядку расположения выводов

Выходные буферы микросхемы КР580ИР82 (КР580ИР83) управляются сигналом ОЕ «разрешение выхода». При поступлении на вход ОЕ сигнала высокого уровня выходные буферы переводятся в состояние «выключено».

Основные статические параметры микросхемы КР580ИР82 (КР580ИР83) приведены в табл. 2, динамические — в табл. 9, временные диаграммы функционирования представлены на рис. 14.

Микросхема КР580ВА86 (КР580ВА87) представляет собой двунаправленный 8-разрядный шинный формирователь с высокой нагрузочной способностью и позволяет осуществить связь микропроцессора с пери-

Таблица 3

Динамические параметры микросхемы КР580ГФ24

Параметр	Обозначение	Норма, нс: не менее	Норма, нс: не более	Условия измерения
Длительность цикла	t_{CY}	$9t_{OSC}$		
Длительность тактового сигнала Ф1	$t_{WH(Ф1)}$	$\frac{2t_{CY}}{9}-20$	—	$C_L=20$
Длительность тактового сигнала Ф2	$t_{WH(Ф2)}$	$\frac{5t_{CY}}{9}-35$	—	То же
Время задержки тактового сигнала Ф2 относительно Ф1	$t_{d(Ф1-Ф2)}$	$\frac{2t_{CY}}{9}$	$\frac{2t_{CY}}{7}+20$	»
Время фронта и спада тактовых сигналов Ф1 и Ф2	$t_{LH(Ф1)}$ $t_{LH(Ф2)}$	—	20	
	$t_{HL(Ф1)}$ $t_{HL(Ф2)}$	—	20	Нагрузка L1 (рис. 5)
Время задержки тактового сигнала Ф2Т относительно Ф2	$t_{d(Ф2-Ф2T)}$	−5	15	(Выход Ф2Т)
Длительность сигнала STSTB	$t_{WL(STSTB)}$	$\frac{t_{CY}}{9}-15$	—	Нагрузка L2 (рис. 5)
Время задержки сигнала STSTB относительно тактового сигнала Ф2	$t_{d(Ф2-STSTB)}$	$\frac{6t_{CY}}{9}-30$	$\frac{6t_{CY}}{9}$	(Выход STSTB)
Время* сохранения сигнала RDYIN относительно сигнала STSTB	$t_{V(STSTB-RDYIN)}$	$\frac{4t_{CY}}{9}$	—	То же
Время задержки сигналов RESET и READY относительно тактового сигнала Ф2	$t_{d(Ф2-RESET)}$ $t_{d(Ф2-READY)}$	$\frac{4t_{CY}}{9}-25$	—	Нагрузка L2 (рис. 5) (Выходы: RESET и READY)
Время* установления сигнала RDYIN относительно сигнала STSTB	$t_{SU(RDYIN-STSTB)}$	$50-\frac{4t_{CY}}{9}$		

Примечание: * — параметры режима измерений.

Таблица 4

Слово состояния и типы машинных циклов

Разряды шины данных	Информация состояния МП (машинные циклы)	Выборка команды	Чтение из памяти	Запись в память	Чтение из стека	Запись в стек	Чтение из УВВ	Запись в УВВ	Разрешение прерывания	Разрешение останова	Разрешение прерывания во время останова
		Слово состояния									
		1	2	3	4	5	6	7	8	9	10
D0	INTA	0	0	0	0	0	0	0	1	0	1
D1	WO	1	1	0	1	0	1	0	1	1	1
D2	STACK	0	0	0	1	1	0	0	0	0	0
D3	HLTA	0	0	0	0	0	0	0	0	1	1
D4	OUT	0	0	0	0	0	0	1	0	0	0
D5	M1	1	0	0	0	0	0	0	1	0	1
D6	INP	0	0	0	0	0	1	0	0	0	0
D7	MEMR	1	1	0	1	0	0	0	0	1	0
Управляющие сигналы		MEMR	MEMR	MEMW	MEMR	MEMW	I/OR	I/OW	INTA	NONE	INTA

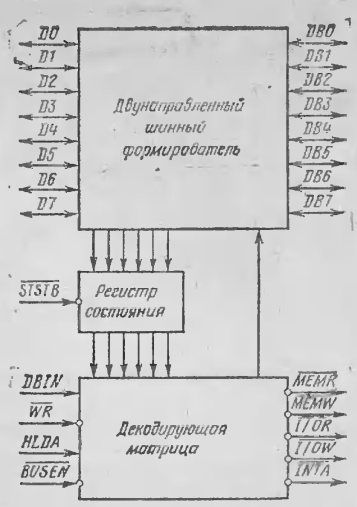


Рис. 7. Электрическая структурная схема микросхемы КР580ВК28 (КР580ВК38)

Таблица 5

Назначение выводов микросхемы КР580ВК28 (КР580ВК38)

Вывод	Обозначение	Назначение
6,8,10, 12,15, 17,19, 21	D0 ... D7	Шина данных
5,7,9, 11,13, 16,18 20	DB0 ... DB7	Системная шина
1	STSTB	Строб состояния
2	HLDA	Подтверждение захвата
3	WR	Запись
4	DBIN	Прием
14	GND	Общий
22	BUSEN	Управление системной шиной
23	INTA	Подтверждение прерывания
24	MEMR	Чтение памяти
25	I/OR	Чтение УВВ
26	MEMW	Запись в память
27	I/OW	Запись в УВВ
28	U_{cc}	Напряжение питания 5 В

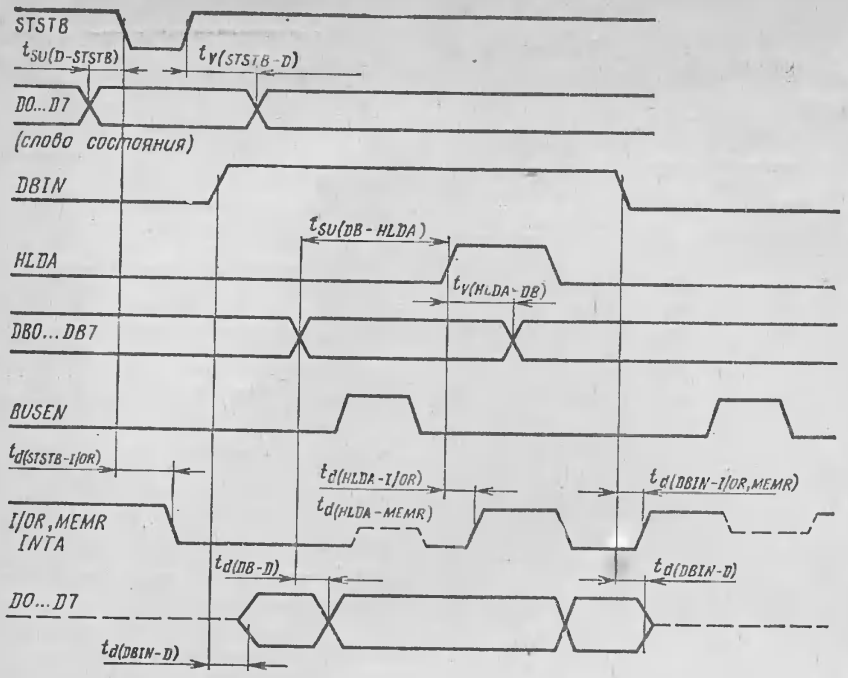


Рис. 8. Временные диаграммы функционирования микросхемы КР580ВК28 (КР580ВК38) в режиме чтения

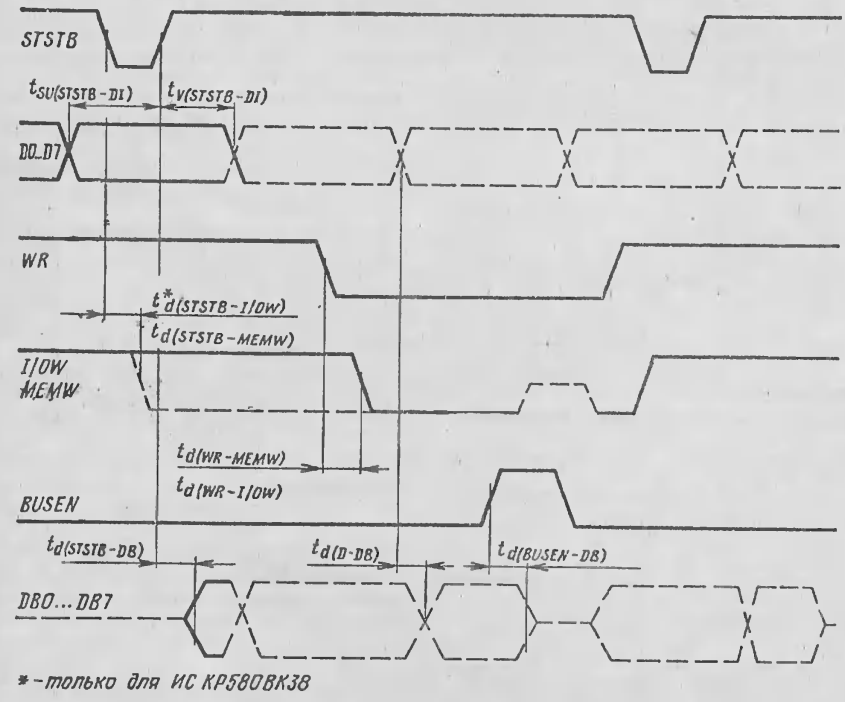
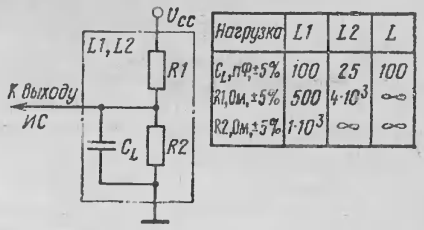


Рис. 9. Временные диаграммы функционирования микросхемы КР580ВК28 (КР580ВК38) в режиме записи



Нагрузка	L1	L2	L
C_L, пФ, ±5%	100	25	100
R1, Ом, ±5%	500	$4 \cdot 10^3$	∞
R2, Ом, ±5%	$1 \cdot 10^3$	∞	∞

Рис. 10. Схема подключения нагрузки при измерении динамических параметров КР580ВК28 (КР580ВК38)

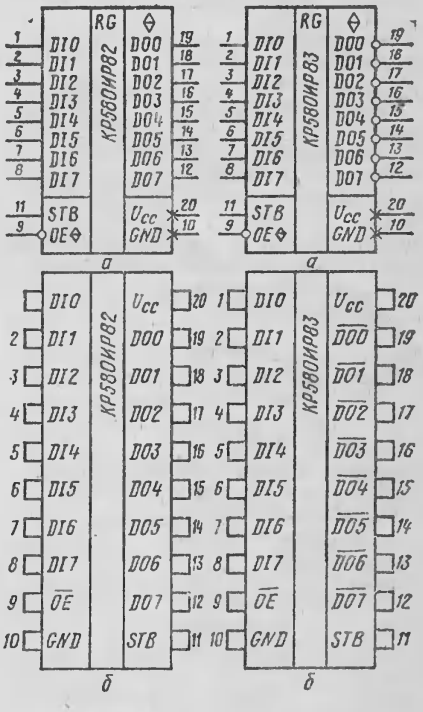


Рис. 11. Условное графическое обозначение микросхемы КР580ИР82:
а — по функциональному назначению выводов; *б* — по порядку расположения выводов

Рис. 12. Условное графическое обозначение микросхемы КР580ИР83:
а — по функциональному назначению выводов; *б* — по порядку расположения выводов

ферийными устройствами ввода-вывода информации. Микросхема КР580ВА87 отличается от микросхемы КР580ВА86 тем, что двунаправленная передача осуществляется с инверсией. Разводка выводов обеих микросхем одинакова.

Условное графическое обозначение представлено на рис. 16, 17, назначение выводов приведено в табл. 10.

Таблица 6

Динамические параметры микросхемы КР580ВК28 (КР580ВК38), режим чтения

Параметр	Обозначение	Норма, нс		Условия измерения
		КР580ВК28	КР580ВК38	
Время задержки управляющих сигналов I/OR, MEMR, INTA относительно сигнала STSTB, не более	$t_{d(STSTB-I/OR)}$ $t_{d(STSTB-MEMR)}$ $t_{d(STSTB-INTA)}$	60 60 60	60 60 60	Нагрузка L1 (рис. 10) (Выходы: I/OR, MEMR, INTA)
Время задержки информации на шине D относительно системной шины, не более	$t_{d(DB-D)}$	30	30	Нагрузка L2 (рис. 10) (Выход D)
Время задержки информации на шине D относительно сигнала DBIN, не более	$t_{d(DBIN-D)}$	45	45	То же
Время задержки управляющих сигналов I/OR, MEMR относительно сигнала DBIN, не более	$t_{d(DBIN-I/OR)}$ $t_{d(DBIN-MEMR)}$	30 30	30 30	Нагрузка L1 (рис. 10)
Время задержки управляющих сигналов I/OR, MEMR, INTA относительно сигнала HLDA, не более	$t_{d(HLDA-I/OR)}$ $t_{d(NLDA-MEMR)}$ $t_{d(NLDA-INTA)}$	25 25 25	25 25 25	Нагрузка L1 (рис. 10)
Время * установления информации на системной шине относительно сигнала HLDA, не менее	$t_{SU(HLDA-DB)}$	10	10	
Время * установления информации на шине D относительно сигнала STSTB, не менее	$t_{SV(STSTB-D)}$	8	8	
Время * сохранения информации на шине D относительно сигнала STSTB, не менее	$t_{V(STSTB-D)}$	5	5	
Время * сохранения информации на системной шине относительно сигнала HLDA, не менее	$t_{V(HLDA-DB)}$	20	20	

Примечание: * — параметры режима измерений.

Нагрузка	L1	L2
U_L, В	2,14	1,5
R, Ом, ±5%	52,7	180
C_L, пФ, ±5%	300	300

Рис. 13. Схема подключения нагрузки при измерении динамических параметров КР580ИР82 (КР580ИР83)

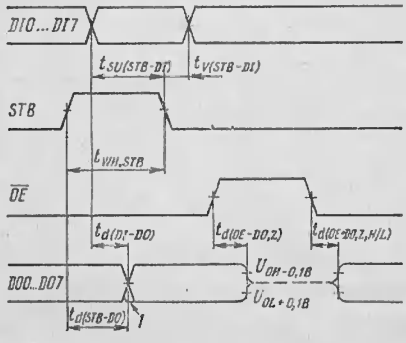


Рис. 14. Временные диаграммы функционирования микросхемы КР580ИР82 (КР580ИР83):

1 — только для КР580ИР83 — на выходе может быть низкий уровень сразу же по приходу высокого уровня STB→

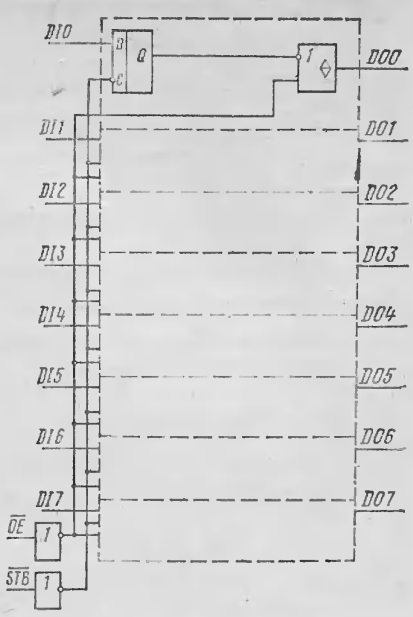


Рис. 15. Электрическая структурная схема микросхемы КР580ИР82 (КР580ИР83)

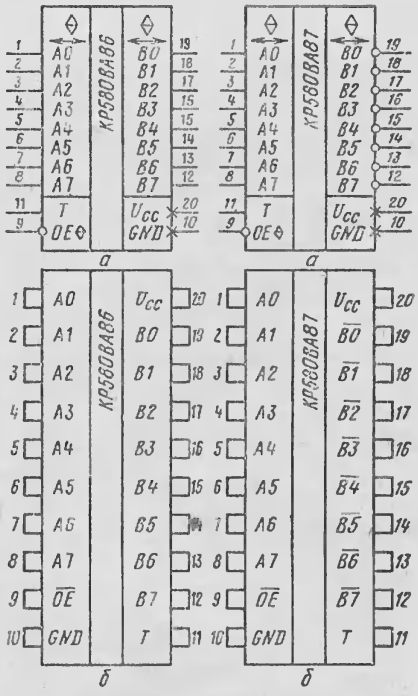


Рис. 16. Условное графическое обозначение микросхемы КР580ВА86:

а — по функциональному назначению выводов; б — по порядку расположения выводов

Рис. 17. Условное графическое обозначение микросхемы КР580ВА87:

а — по функциональному назначению выводов; б — по порядку расположения выводов

Таблица 7

Динамические параметры микросхем КР580ВК28 (КР580ВК38), режим записи

Параметр	Обозначение	Норма, нс		Условия измерения
		КР580ВК28	КР580ВК38	
Время задержки управляющих сигналов I/OW, MEMW относительно сигнала WR, не более	$t_{d(WR-I/OW)}$ $t_{d(WR-MEMW)}$	45	—	Нагрузка L (рис. 10)
Время задержки * управляющих сигналов I/OW, MEMW относительно сигнала STSTB, не более	$t_{d(STSTB-I/OW)}$ $t_{d(STSTB-MEMW)}$	—	60	То же
Время задержки информации на системной шине DB относительно сигнала STSTB, не более	$t_{d(STSTB-DB)}$	30	30	»
Время задержки информации на системной шине DB относительно шины D, не более	$t_{d(D-DB)}$	40	40	»
Время задержки информации на системной шине DB относительно сигнала BUSEN, не более	$t_{d(BUSEN-DB)}$	30	30	»

Примечание: * — только для КР580ВК38.

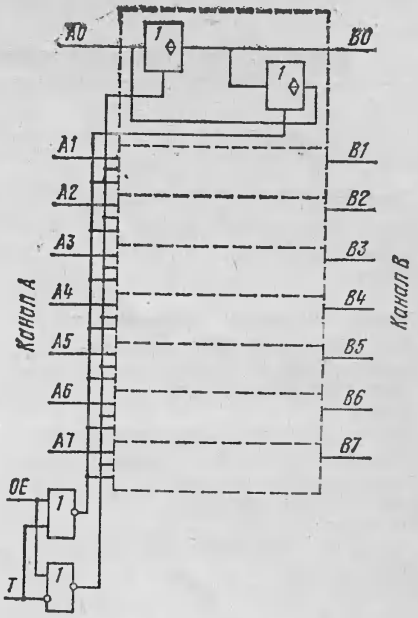


Рис. 18. Электрическая структурная схема микросхемы КР580ВА86 (КР580ВА87)

Таблица 8

Назначение выводов микросхемы КР580ИР82 (КР580ИР83)

Вывод	Обозначение	Назначение
1...8	DI0 ... DI7	Входы регистра
9	OE	Разрешение выхода
10	GND	Общий
11	STB	Строб
19...12	D10 ... D17	Выходы регистра
20	U_{cc}	5 В

Таблица 9

Динамические параметры микросхемы КР580ИР82 (КР580ИР83)

Параметр	Обозначение	Норма, нс		Условия измерения
		КР580ИР82	КР580ИР83	
Время задержки от входа до выхода, не более	$t_{d(DI-DO)}$	30	22	Нагрузка L1 (рис. 15)
Время задержки от строба до выхода, не более	$t_{d(STB-DO)}$	45	40	Нагрузка L2 (рис. 15)
Время задержки от сигнала OE до перехода в состояние «выключено», не более	$t_{d(OE-DO, Z)}$	18	18	То же
Время задержки от сигнала OE до перехода выхода из состояния «выключено», не более	$t_{d(OE-DO, Z, H/L)}$	30	30	»
Время * установления входного сигнала DI относительно строба, не менее	$t_{SU(STB-DI)}$	0	0	
Время * сохранения входного сигнала DI относительно строба, не менее	$t_{V(STB-DI)}$	25	25	
Длительность * сигнала строба, не менее	$t_{WH, STB}$	15	15	

Примечание: * — параметры режима измерений.

Таблица 10

Назначение выводов микросхемы КР580ВА86/87

Вывод	Обозначение	Назначение
1...8	A0 ... A7	Шина А (вход-выход)
9	OE	Разрешение выхода
10	GND	Общий
11	T	Направление передачи
19...12	B0 ... B7	Шина В (вход-выход)
20	U_{cc}	5 В

Таблица 11

Динамические параметры микросхемы КР580ВА86 (КР580ВА87)

Параметр	Обозначение	Норма, нс		Условия измерения
		КР580ВА86	КР580ВА87	
Время задержки от входа до выхода, не более	$t_{d(A-B)}$ $t_{d(B-A)}$	30 30	22 22	Нагрузка L3 (B) L1 (A), (рис. 20)
Время задержки от сигнала ОЕ до перехода выхода в состояние «выключено», не более	$t_{d(OE-A, Z)}$ $t_{d(OE-B, Z)}$	18 18	18 18	Нагрузка L4 (B) L2 (A), (рис. 20)
Время задержки от сигнала ОЕ до перехода выхода из состояния «выключено», не более	$t_{d(OE-A, Z, H/L)}$ $t_{d(OE-B, Z, H/L)}$	30 30	30 30	Нагрузка L2 (A) L4 (B) (рис. 20)
Время * установления сигнала Т относительно сигнала ОЕ, не менее	$t_{SU(OE-T)}$	30	30	
Время * сохранения сигнала Т относительно сигнала ОЕ, не менее	$t_{V(OE-T)}$	18	18	

Примечание: * — параметры режима измерений.

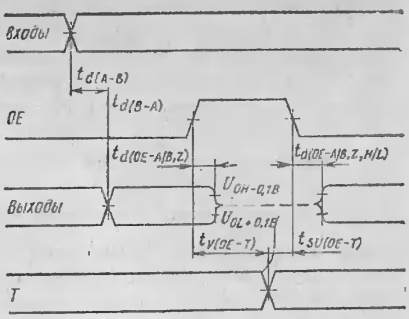
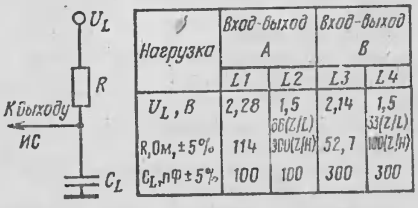


Рис. 19. Временные диаграммы функционирования микросхемы КР580ВА86 (КР580ВА87)



Нагрузка	Вход-выход A		Вход-выход B	
	L1	L2	L3	L4
U_L, В	2,28	1,5	2,14	1,5
R, Ом, ±5%	114	66(Z/L) 300(Z/H)	52,7	53(Z/L) 100(Z/H)
C_L, пФ ±5%	100	100	300	300

Рис. 20. Схема подключения нагрузки при измерении динамических параметров КР580ВА86 (КР580ВА87)

Электрическая структурная схема микросхемы КР580ВА86 (КР580ВА87) представлена на рис. 18.

Режим работы микросхемы КР580ВА86 (КР580ВА87) определяется управляющими сигналами ОЕ «разрешение выхода» и Т «направление передачи». При поступлении на вход ОЕ сигнала высокого уровня информационные выходы А и В переходят в состояние «выключено».

При наличии на входе ОЕ сигнала низкого уровня направление передачи информации определяется сигналом Т. При подаче на вход Т сигнала высокого уровня осуществляется передача информации с канала А в канал В, при подаче на вход Т сигнала низкого уровня — с канала В в канал А.

Основные статические параметры микросхемы КР580ВА86 (КР580ВА87) приведены в табл. 2, динамические — в табл. 11, временные диаграммы функционирования представлены на рис. 19.

Телефон для справок: 536-57-55, Москва

ЛИТЕРАТУРА

1. Архитектура и проектирование микроЭВМ. Организация вычислительных процессов / Под редакцией Л. Н. Преснухина.— М.: Высшая школа, 1986.
2. Прангишвили И. В. Микропроцессоры и микроЭВМ. — М.: Энергия, 1979.
3. Алексенко А. Г., Голицын А. А., Иванников А. Д. Проектирование РЭА на микропроцессорах.— М.: Радио и связь, 1984.
4. Дж. Коффрон. Технические средства микропроцессорных систем. — М.: Мир, 1983.
5. Дж. Хилбурн, П. Джулич. МикроЭВМ и микропроцессоры.— М.: Мир, 1979.
6. Б. Соучек. Микропроцессоры и микроЭВМ.— М.: Сов. радио, 1979.

Статья поступила 6 июля 1987 г.

РЖ ВИНИТИ

4Б437. Новые возможности в графике персональных ЭВМ. New issues in PC graphics *McNierney Ed.* „Dr. Dobb's J.", 1986, 11, № 11, 30—32, 35—36, 38 (англ.)

Рассматриваются вопросы эффективного использования новых возможностей графических контроллеров третьего поколения, включающих в себя МП типа Intel 82786 и Texas Instruments 34010. Эти процессоры содержат аппаратные средства построения графических элементов (окружностей, линий, текстов и др.) и действительно имеют МП-архитектуру, что позволяет разрабатывать и выполнять сложные графические алгоритмы параллельно с работой основного процессора. Указанные контроллеры предоставляют возможности адресации до 512 М байт оперативной памяти, а также аппаратные средства управления окнами и многостраничной экранной памятью. Приводится краткая сравнительная характеристика особенностей архитектуры и возможностей графических МП. Излагаются проблемы эффективного использования новых МП в качестве сопроцессоров.

3Б158. [Графика для персональных ЭВМ фирмы ЭВМ]. IBM PC graphics reach new heights multi-standart cards, monitors lend versatility. *Dalton Richard.* „Comput. Graph. World", 1986, 9. № 5. 67—68, 70—71, 73 (англ.) Место хранения —ГПНТБ СССР.

Существенное улучшение графических возможностей персонального компьютера фирмы IBM обеспечивается новым графическим адаптером EGA (Enhanced Graphics Adapter), который фирма начала производить в начале 1985 г. Новый адаптер позволяет выводить на экран 16 цветов одновременно и с лучшим разрешением, чем старый адаптер CGA (Color Graphics Adapter). К сожалению, не все программы, работающие на CGA, будут работать и на ECA. Однако по прогнозу фирмы Future Computing (США), к 1990 г. около 90 % проданных графических плат будут совместимы со стандартом EGA. Автором статьи испытывались две такие платы — QuadEGA фирмы Quadram (США) и Spectra — EGA фирмы Genoa Systems (США), причем первая из них поставляется вместе с программным обеспечением и стоит дешевле, чем сама EGA фирмы IBM. Для использования EGA совместных плат необходим новый монитор с повышенной частотой горизонтального сканирования строк. Такие мониторы поставляются как фирмой IBM, так и другими фирмами.

УДК 621.3.049.77 : 681.3.06

Морозов С. А., Барановский Д. М., Минкин Л. К., Семикин А. П., Черкай А. Д. **Однокристальные ЭВМ серии КБ 1013** // Микропроцессорные средства и системы.—1987.—№ 5.—С. 5.

Рассмотрены основные функции и характеристики, структура, система команд, конструктивные особенности и электрические параметры однокристальных 4-разрядных ЭВМ КБ1013ВК1-2, КБ1013ВК4-2.

УДК 681.3

Котляров В. П. **Технология разработки программного обеспечения встроенных микроЭВМ и поддерживающие ее инструментальные комплексы** // Микропроцессорные средства и системы.— 1987.— № 5.— С. 29.

Описывается подход к разработке промышленного программного продукта для задач управления, учитывающий реальные пропорции, сложившиеся в квалификации программирующих инженерно-технических кадров на производстве. Предлагается разработка проблемного обеспечения на базе профессиональных технологических комплексов, ориентированных на проблемную терминологию и методы ведения программных разработок инженерами из промышленности.

УДК 681.3.06

Борковский А. Б. **Текстовая база данных** // Микропроцессорные средства и системы.—1987.—№ 5.—С. 41.

Текстовая база данных «Картотека NOTES» для ПЭВМ типа IBM PC обеспечивает хранение, поиск и редактирование текстовых записей произвольной длины. Текстовые записи идентифицируются ключевыми словами с отношением «многие — к многим» между карточками и словами. Непосредственное взаимодействие и встроенный экранный редактор облегчают использование картотеки неподготовленным пользователем.

УДК 681.327

Колпаков И. Ф. **Шина VME и ее применения.**— Микропроцессорные средства и системы.— 1987.— № 5.— С. 43.

Шина VME является в настоящее время единственным международным стандартом широкого применения для 32-разрядных многопроцессорных систем. Рассмотрены основные характеристики шины VME и ее структурная схема, главные области применения, в частности, ее использование в спектрометрах и системах управления ускорителей физики элементарных частиц.

УДК 681.327

Канцеров В. А., Першин А. С. **VME — магистраль нового поколения** // Микропроцессорные средства и системы.— 1987.— № 5.— С. 47.

Представлены основные характеристики VME магистрали, подробно рассмотрены назначение шин и протокол обмена, конструктивная и элементная базы.

УДК 681.327.8

Коломейцев В. А., Степченков Ю. А., Филин А. В. **Интерфейс с последовательным арбитражем: реализация и пути усовершенствования** // Микропроцессорные средства и системы. — 1987. — № 5. — С. 56.

Предложены схемотехнические решения некоторых функциональных узлов интерфейса VME, реализующих протокол обмена в части арбитража. Рассмотрены варианты улучшения характеристик арбитража.

UDC 621.3.049.77 : 681.3.06

Morozov S. A., Baranovsky D. M., Minkin L. K., Semikin A. P., Cherkay A. D. **Single-chip microcomputers of КБ1013 famity.** // Microprocessor devices and systems.— 1987.— № 5.— P. 5.

Main functions and technical specifications, structure and instruction set as well as peculiarities of 4-bit single-chip computer ICs КБ1013ВК1-2 and К1013ВК4-2 are given.

UDC 681.3

Kotlyarov V. P. **Software design technology for built-in microprocessor devices and professional software tools which support it.** // Microprocessor devices and systems.— 1987.— № 5.— P. 29.

New approach to software development process for built-in microprocessor controllers which takes into account real qualification of programming personnel in the industry is described. The author proposes to create target programs using professional technological software packages which enable software description and development by engineers in the industry using problem-oriented terms and methods.

UDC 681.3.06

Borkovsky A. B. **A text database.** // Microprocessor devices and systems.— 1987.— N 5.— P. 41.

A text database NOTES for IBM PC provides storing, retrieval and editing of text records. Each record can have arbitrary format and its length is unlimited. The records are accessed by keywords with «many-to-many» relation between records and keywords. The desktop-like interface permits scrolling through keywords list or the list of records titles and working with sevaral records at the same time. The package includes full-featured screen text editor to read, modify and create text record.

UDC 681.327

Kolpakov I. F. **VME bus and its applications.** // Microprocessor devices and systems.—1987.— № 5.— P. 43.

VME bus is now the only general purpose international bus standard for 32-bit multiprocessor computer systems. Main technical features and structure of VME bus are explained in the article. The dominant areas of VME bus are explained in the article. The dominant areas of VME bus application are shown, inccluding high-energy physics spectrometers and elementary particle accelerator control systems.

Kantserov V. A., Pershin A. S. **VME — microcomputer bus of a new generation.** // "Microprocessor devices and systems".— 1987.— N 5.— P. 47.

The article describes main features of VME bus, paying special attention to data transfer protocol and bus lines assignment, its mecranical consruction and component base.

UDC 681.327.8

Kolomejtsev V. A., Stepanchenkov Yu. A., Filin A. V. **System interface with sequential priority resolvers: realization and the ways of enhancement.** // "Microprocessor devices and systems".— 1987.— N 5.— P. 56.

The description of circuit design of some functional units in VME bus interface, performing bus arbitrage according to standard data transfer protocol. The ways of enhancing arbiter operation are also discussed.

УДК 681.3.06

Брябрин В. М., Сираджов Б. Т. **Пакет демонстрационной графики АЛЬФА-ГРАФ для ПЭВМ типа ЕС-1841** // Микропроцессорные средства и системы.— 1987.—№ 5.—С. 60.

В статье описываются архитектура и функциональные возможности пакета демонстрационной графики АЛЬФА-ГРАФ ПЭВМ типа ЕС-1841 для операционной системы Альфа-ДОС/ДОС-16.

УДК 681.3.022

Большаков И. А. **«Черепашья» графика для ДВК.** // Микропроцессорные средства и системы.— 1987.—№ 5.—С. 63.

Описываются базисные и вспомогательные подпрограммы на языке Паскаль, имитирующие на дисплее 15ИЭ-00-013 приемы так называемой черепашьей графики — построение контурных изображений путем пошагового перемещения курсора по экрану. Такой дисплей обычно работает в составе широко известного советского микрокопьютера — ДВК. Развиваются приемы построения не только статических, но и динамически меняющихся («анимизированных») изображений — мерцаний, перемещений объектов по экрану и др.

УДК 681.3.022

Попов С. Н. **Графический редактор для ПЭВМ «Микроша».** // Микропроцессорные средства и системы.—1987.—№ 5.—С. 65.

Рассматривается один из компонентов системного программного обеспечения персональной ЭМВ «Микроша» — графический редактор, позволяющий формировать изображения в ручном и автоматическом режимах с использованием графики низкого разрешения (128 точек по горизонтали, 64 точки по вертикали). В ОЗУ ПЭВМ могут одновременно располагаться до 11 графических экранов. Редактор предназначен для подготовки иллюстративных материалов к лекциям или урокам, подготовки данных для прикладных программ, для обучения составлению простейших программ.

Кулаичев А. П. **Графический ассистент для синтеза, каталогизации и экспонирования цветных изображений.** // Микропроцессорные средства и системы.— 1987.—№ 5.—С. 69.

Рассмотрены основные возможности программной системы, включающей в себя графический редактор и подсистему управления банком изображений.

УДК 681.3.022

Алексеенко М. М., Березовский М. А., Свирин Б. Н., Яблонский А. К., Яким В. В. **Интерактивная система геометрического моделирования для СМ ЭВМ и векторного процессора.** // Микропроцессорные средства и системы.—1987.—№ 5.—С. 73.

В работе рассматриваются вопросы создания программно-технического обеспечения машиностроительных САПР. Предложена архитектура высокопроизводительного АРМ на базе СМ ЭВМ и векторного процессора. Приводится описание двух программных комплексов: системы интерактивного геометрического моделирования (СИГМа) и проблемно-ориентированного машиностроительного языка конструирования (МАЯК).

УДК 681.325.5—181.4

Королев В. Н., Жарков А. С., Зубченко А. П., Штанаков А. Ю. **Микропроцессорная система МС-80.** // Микропроцессорные средства и системы.—1987.—№ 5.—С. 75.

Даются аппаратные и программные характеристики микропроцессорной системы МС-80 для распределенного управления в АСУ ТП. Приводится структурная схема программируемых контроллеров на базе МС-80 и кратко характеризуются системный интерфейс и структура технических средств системы.

UDC 681.3.06

Bryabrin V. M., Syradzhov B. T. **ALPHA-IBM/PC-compatible EC1841 computer.** // Microprocessor **GRAPH: Graphics demonstration software kit for** devices and systems.— 1987.— № 5.— P. 60.

The article describes ALPHA-GRAPH graphics demo software package structure and functional features for EC1841 microcomputer running under ALPHA-DOS/DOS-16 operating system.

UDC 681.3.022

Bolshakov I. A. **Turtle graphics support for DVK computer.** // Microprocessor devices and systems.— 1987.— № 5.— P. 63.

Basic and auxilary subroutines written in PASCAL language for turtle graphics support of 15ИЭ-00-013 display (the one most commonly used with ДВК microcomputers) are described. These subroutines perform turtle grafhic imitation, i. e. enable contour video patterns creation using stepwise cursor moves across video screen. The author gives means for drawing not only static, but also dynamic images which are animated by flashing or various movements.

UDC 681.3.022

Popov S. N. **Graphics editor for "Microsha" personal computer.** // Microprocessor devices and systems.— 1987.— № 5.— P. 65.

Graphics editor, a component of the system software for "Microsha" PC, is revealed. The program can create video images in both manual and automatic modes and uses low-resolution graphics (128 by 64 pixels). The computer RAM can store up to 11 graphic screens simultaneously. The editor can be used to prepare illustrations for lessons or lectures, as well as for data input to other programs, and as programming tutorial aid.

UDC 681.3.022

Kulaichev A. P. **Graphic assistent for colour image synthesis, exposure and catalogue handling.** // Microprocessor devices and systems.— 1987.— № 5.— P. 69

Main features of software kit including graphics editor and database subsystem are shown.

UDC 681.3.022

Alekseenko M. M., Berezovsky M. A., Swirin B. N., Yablonsky A. K., Yakim V. V. **Interactive geometric modeliing system for SM computer and vector processor.** // Microprocessor devices and systems.— 1987.— № 5.— P. 73.

Some aspects of hardware/software support design for machinery CAD systems are discussed. The architecture of powerful work station based on PDP-11 compatible computer and vector processor is proposed. The description of two software packages are given: the interactive geometric modelling system (СИГМа) and problem-oriented machinery designer's language (МАЯК).

UDK 681.325.5—181.4

Korolev V. N., Zharkov A. S., Zubchenko A. P., Shtanakov A. Yu. **Microprocessor—based system MS-80.** // Microprocessor devices and systems.— 1987.— № 5.— p. 75.

The article describes the hardware and software of MS-80 microprocessor system. MS-80 is designed for CAM distributed control applications. The structural diagram of programmable controllers based on MS-80 is given together with the description of system interface and of the system hardware structure.

МИКРОПРОЦЕССОРНАЯ ТЕМАТИКА В КНИГАХ «ЭНЕРГОАТОМИЗДАТА»

Энергоатомиздат готовит к выпуску в свет в конце 1987 — начале 1988 года серию книг с общим названием «Микропроцессорные БИС и их применение» под редакцией лауреата Государственной премии СССР, доктора технических наук, профессора Домрачева В. Г.

В серию войдут следующие книги:

1. **Басманов А. С., Широков Ю. Ф.** Микропроцессоры и однокристальные микроЭВМ: Номенклатура и функциональные возможности.—8,5 л.: ил.—(В обл.): 45 к.

2. **Одноплатные** микроЭВМ/В. Г. Домрачев, С. Н. Иванов, А. Ф. Романов, Ю. Н. Чернышов.—10 л.: ил.—(В обл.): 50 к.

3. **Цифровая** обработка информации на основе быстродействующих БИС/С. А. Гамкрелидзе, А. Б. Завьялов, П. П. Мальцев, В. Г. Соколов.—8,5 л.: ил.—(В обл.): 45 к.

4. **Широков Ю. Ф., Муренко Л. Л., Чурков В. М.** Программаторы запоминающих и логических интегральных микросхем.—10 л.: ил.—(В обл.): 50 к.

5. **Домнин С. Б., Иванов Е. А., Муренко Л. Л.** Средства комплексной отладки микропроцессорных устройств.—10 л.: ил.—(В обл.): 50 к.

6. **Белов А. М., Иванов Е. А., Муренко Л. Л.** Средства автоматизации программирования микропроцессорных устройств.—10 л.: ил.—(В обл.): 50 к.

7. **Овчинников В. В.** Архитектура распределенных информационно-вычислительных микропроцессорных систем.—7,5 л.: ил. 40 к.

8. **Бойченко Е. В.** Методы схемотехнического проектирования распределенных информационно-вычислительных микропроцессорных систем.—7,5 л.: ил. 40 к.

В книгах обобщены и систематизированы материалы по:

составу, структуре, характеристикам, функциональным возможностям, выбору и применению микропроцессорных комплектов и однокристальных микроЭВМ широкого применения последних разработок (К1810, К1814, К1816, К1820, К1821 и др.);

средствам отладки и программирования микропроцессорных устройств (ориентированным на использование микроЭВМ);

созданию одноплатных микроЭВМ с применением новых высокопроизводительных микропроцессоров, например К1810;

архитектуре и проектированию распределенных микропроцессорных систем.

В книгах отражено современное состояние отечественной микропроцессорной техники. Авторы ставили своей задачей придать книгам практическую направленность, поэтому сделана ставка на отечественные разработки как собственно БИС, так и отладочных средств, которыми могут воспользоваться многочисленные специалисты по созданию микропроцессорных систем в различных отраслях промышленности. Одновременно книги могут быть полезными для аспирантов и студентов, специализирующихся в области вычислительной и информационно-измерительной техники, систем управления и обработки информации.

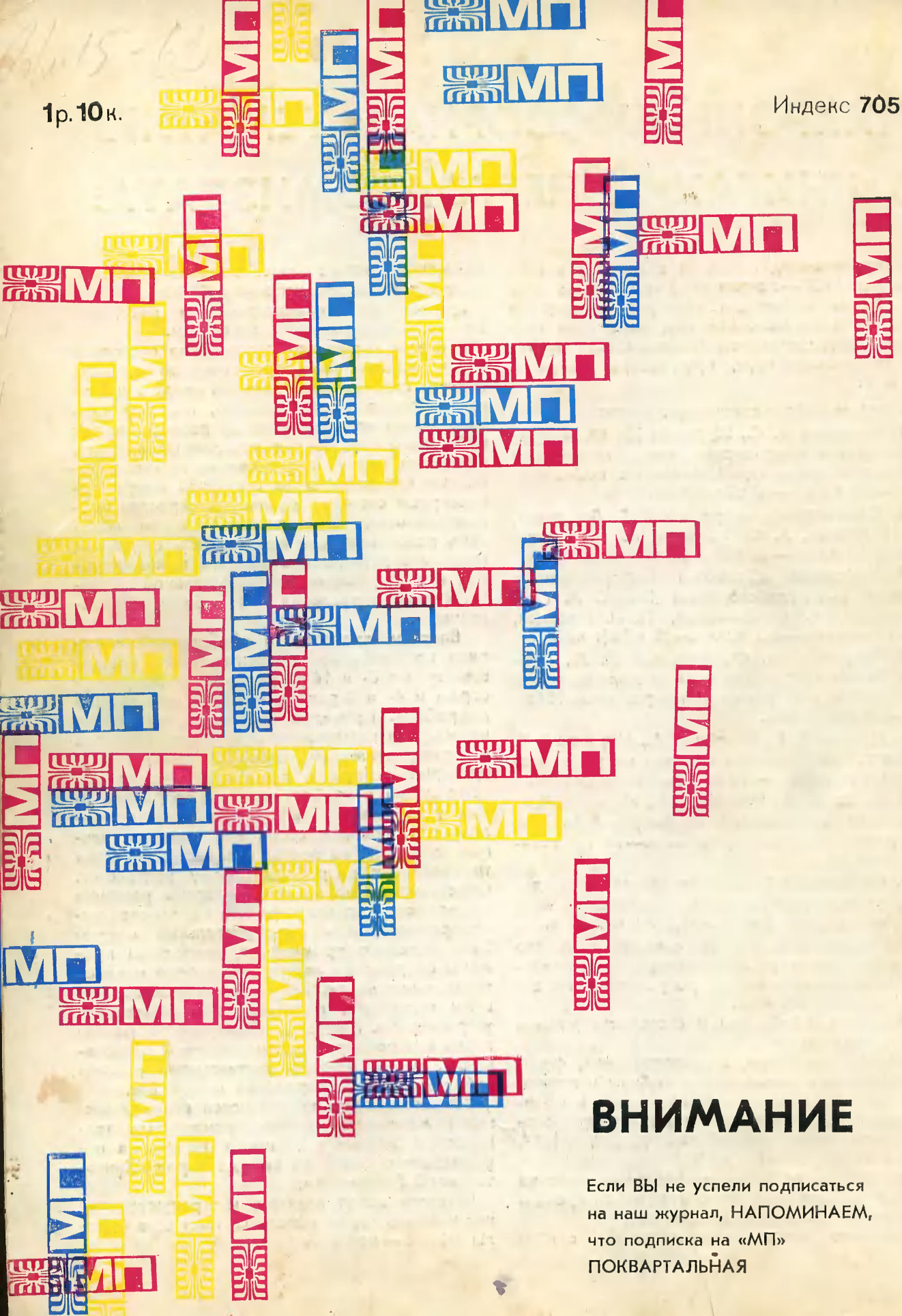
Впервые представлен обобщенный материал по наиболее перспективным в настоящее время 8- и 16-разрядным микропроцессорам и 4- и 8-разрядным однокристальным микроЭВМ. Приведен сравнительный анализ имеющейся номенклатуры микропроцессоров и однокристальных микроЭВМ, рассмотрены критерии выбора и даны практические рекомендации разработчикам микропроцессорных устройств по выбору наиболее приемлемого микропроцессора или однокристальной микроЭВМ на основе анализа технических требований к разрабатываемому устройству. Описаны типовые схемотехнические решения микропроцессорных устройств на основе микропроцессоров и однокристальных микроЭВМ широкого применения, приведены примеры сопряжения микропроцессоров и однокристальных микроЭВМ с клавиатурой, средствами индикации, внешними запоминающими устройствами. Раскрыты возможности реализации контроллеров промышленного назначения. С единых позиций рассмотрены различные методы проектирования микроЭВМ.

Заявки на книги принимаются всеми книжными магазинами, распространяющими техническую литературу, после получения информации о сроке их выхода через «Книготорговый бюллетень».

Читатели могут заказать интересующие их книги также через магазины, имеющие отделы «Книга — почтой».

1р. 10к.

Индекс 70588



ВНИМАНИЕ

Если ВЫ не успели подписаться на наш журнал, НАПОМИНАЕМ, что подписка на «МП» ПОКВАРТАЛЬНАЯ

МИКРОПРОЦЕССОРНЫЕ СРЕДСТВА И СИСТЕМЫ

Москва · 1987 | 5