НПИ «НАУЧНЫЙ ЦЕНТР»

ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ДВК

КНИГА 5

РУКОВОДСТВО СИСТЕМНОГО ПРОГРАММИСТА

МОСКВА 1990 год

АННОТАЦИЯ

Данная книга содержит сведения о генерации системы, программировании внешних устройств и о мониторе расширенной памяти, контрольных задачах и программе пакетной обработки.

В главе «УСТАНОВКА И ГЕНЕРАЦИЯ СИСТЕМЫ» содержится описание процедуры установки и модификации ОС ФОДОС-2, поставляемой на различных дистрибутивных носителях. Описывается процесс генерации ОС ФОДОС-2, в результате которого создается вариант ОС, ориентированный на определенную конфигурацию аппаратных средств и требуемую область применения.

Глава «ПРОГРАММИРОВАНИЕ ПЕРИФЕРИЙНЫХ УСТРОЙСТВ» представляет собой руководство системного программиста по написанию и работе с драйверами в ОС ФОДОС-2. В главе описаны особенности функционирования драйверов специальных устройств.

В главе «МОНИТОР РАСШИРЕННОЙ ПАМЯТИ» описывается монитор, позволяющий использовать память емкостью более 32К слов на ЭВМ с диспетчером памяти. Монитор расширенной памяти имеет набор программных запросов для расширения области логической адресации программы.

Глава «КОНТРОЛЬНЫЕ ЗАДАЧИ» описывает порядок работы с контрольными задачами, проверяющими правильность функционирования ОС.

Глава «ПРОГРАММА ПАКЕТНОЙ ОБРАБОТКИ» содержит описание языка управления потоком заданий и действий оператора при работе с программой.

УСТАНОВКА И ГЕНЕРАЦИЯ СИСТЕМЫ
РУКОВОДСТВО СИСТЕМНОГО ПРОГРАММИСТА

1. ОБЩИЕ СВЕДЕНИЯ О СИСТЕМЕ ФОДОС-2

Операционная система реального времени ФОДОС-2 представляет собой высокоэффективную операционную систему для одного пользователя и предназначена для решения задач в реальном масштабе времени, а также разработки программ в интерактивном режиме.

Система предоставляет пользователю следующие возможности:

— однопользовательский режим работы;
— мультитерминальную поддержку;
— организовать доступ к памяти объемом до 4096К байт;
— обеспечение мультипрограммного режима работы (управление выполнением до 2-х задач пользователя и одновременное выполнение программы регистрации ошибок, как системной задачи);
— использование расширенной памяти в качестве системного устройства;
— прием и обработку командных строк операционной системы;
— обработку программных запросов, использующих расширенную память средствами языка АССЕМБЛЕР и ФОРТРАН;
— генерацию системы на определенную аппаратную конфигурацию и требуемую область применения.

Для обеспечения нормального функционирования операционной системы необходим следующий минимальный состав технических средств:

— центральный процессор;
— ОЗУ (от 64 до 4096К байт);
— терминал;
— таймер;
— устройство внешней памяти на гибких или жестком дисках.

2. СТРУКТУРА СИСТЕМЫ

2.1. Мониторы ФОДОС-2

Управление работой ФОДОС-2 осуществляется с помощью одного из трех мониторов: монитора одного задания (SJ), монитора основного — фонового задания (FB) и монитора расширенной памяти (XM).

2.1.1. Монитор одного задания

Монитор SJ представляет собой наименьший из поставляемых мониторов ФОДОС-2; для резидентной части монитора требуется 2К слов памяти. Он имеет наивысшую скорость реакции на прерывание, поэтому его целесообразно использовать для решения задач, требующих большой скорости передачи данных. Монитор SJ обслуживает все периферийные устройства (кроме диспетчера памяти) и работает в любой конфигурации с объемом памяти не менее 16К слов, позволяя использовать до 28К слов памяти. Поставляемый монитор SJ поддерживает мультитерминальный режим, т. е. режим при котором одна программа может работать с несколькими терминалами. При генерации системы можно получить монитор SJ, не поддерживающий мультитерминальный режим.

2.1.2. Монитор основного-фонового задания

Монитор FB представляет собой наименьший (5.3К слов памяти) из мониторов ФОДОС-2, поддерживающих режим мультипрограммирования. Он обеспечивает выполнение двух полностью независимых заданий: основного задания и фонового задания. Поставляемый монитор FB поддерживает к тому же системные задания, т. е. вместе с основным и фоновым заданиями может выполняться до шести системных заданий. При генерации системы можно получить монитор FB, не поддерживающий системные задания.

Фоновый режим работы монитора FB аналогичен режиму работы монитора одного задания; все системные средства, имеющиеся в распоряжении пользователя при работе с монитором SJ, также доступны ему при работе в фоновом режиме под управлением монитора FB.

Для своей работы монитор FB требует не менее 16К и позволяет использовать до 28К слов памяти.

Областью применения монитора FB являются процессы, когда пользователю одновременно с работой в реальном масштабе времени необходимо выполнение обработки данных и других системных операций. Если в этом нет необходимости, для экономии ресурсов системы лучше использовать монитор SJ.

2.1.3. Монитор расширенной памяти

Монитор XM имеет все характерные особенности монитора FB, дополнительно к этому он позволяет обслуживать до 2M слов памяти. Монитор XM требует, чтобы USR была резидентной в памяти. Резидентная часть монитора XM (включая USR) занимает 9.2K слов памяти.

Если пользователю не нужно более 32K слов памяти, то рекомендуется использовать монитор FB или SJ.

Таблица 1

КРАТКАЯ ХАРАКТЕРИСТИКА ПОСТАВЛЯЕМЫХ МОНИТОРОВ

Характеристика	SJ	FB	XM
Размер памяти для резидентной части монитора (слов)	2.0K	5.3K	6.6K + 2.6K USR
Обслуживание памяти (слов)	16—28K	16—28K	32—2048K
Размер на диске (в блоках)	78	91	102
Поддержка основного задания	Нет	Да *	Да *
Поддержка системного задания	Нет	Да	Да
Поддержка таймера (.MRKT, .CMKT) смена даты и сброс времени в полночь)	Да *	Да	Да
Печать сообщений об ошибках при системных ошибках ввода-вывода	Да *	Да	Да
Мультитерминальная поддержка	Да *	Да *	Да *
Поддержка стартового файла DATIME	Да *	Да *	Да *
Поддержка четности памяти	Нет *	Нет *	Нет *
Поддержка «календаря»	Да *	Да *	Да *
Использование программного запроса FETCH	Да	Да	Да *
Поддержка всех команд клавиатурного монитора	Да *	Да *	Да *
Выполнение команд плавающей запятой	Да *	Да *	Да *
Поддержка BATCH	Да *	Да *	Да *
Регистрация ошибок	Да *	Да *	Да *
Поддержка периферийных устройств	Все	Все	Все

ПРИМЕЧАНИЕ. * означает, что данная характеристика может быть либо выбрана, либо нет в процессе генерации.

2.2. Компоненты дистрибутивного набора

Введем понятие дистрибутивного носителя. Дистрибутивным носителем называется носитель, на котором находятся все компоненты операционной системы. Используя дистрибутивный носитель пользователь может создать конкретную операционную систему, соответствующую составу оборудования и области применения. Набор файлов на дистрибутивном носителе называется дистрибутивным набором. Компоненты дистрибутивного набора описаны в табл. 2.

Операционная система ФОДОС-2 поставляется либо на гибких дисках, либо на магнитной ленте. Независимо от носителя операционная система содержит одни и те же компоненты.

Получив дистрибутивный носитель пользователю следует, во-первых, установить систему. Подробно установка системы описана в разд. 3. Затем, если необходимо, пользователь может сгенерировать новую операционную систему ФОДОС-2, отличную от поставляемой. Подробно генерация системы описана в разд. 4.

Таблица 2

КОМПОНЕНТЫ ДИСТРИБУТИВНОГО НАБОРА

Имя файла	Размер в блоках	Описание
1	2	3
Мониторы		
FMONSJ.SYS	78	Монитор одного задания
FMONFB.SYS	91	Монитор основного — фонового задания
FMONXM.SYS	102	Монитор расширенной памяти
Драйверы		
BA.SYS	7	Драйвер программы пакетной обработки для мониторов SJ, FB
BAX.SYS	7	Драйвер программы пакетной обработки для монитора XM
DD.SYS	5	Драйвер ленты кассетного типа для мониторов SJ, FB
DDX.SYS	5	Драйвер ленты кассетного типа для монитора XM
DP.SYS	3	Драйвер пакета дисков (29 Мб) для мониторов SJ, FB
DU.SYS	4	Драйвер гибкого и винчестерского дисков для мониторов SJ, FB
DUX.SYS	4	Драйвер гибкого и винчестерского дисков для монитора XM
DW.SYS	4	Драйвер винчестерского диска для мониторов SJ и FB

1	2	3
DWX.SYS	4	Драйвер винчестерского диска для монитора XM
DX SYS	4	Драйвер гибких дисков для мониторов SJ, FB
DXX.SYS	4	Драйвер гибких дисков для монитора XM
DY SYS	4	Драйвер гибких дисков с двойной плотностью для мониторов SJ, FB
DYX SYS	4	Драйвер гибких дисков с двойной плотностью для монитора XM
EL SYS	5	Драйвер регистратора ошибок для монитора SJ
LD SYS	8	Драйвер логических дисков для мониторов SJ, FB
LDX.SYS	8	Драйвер логических дисков для монитора XM
LP.SYS	2	Драйвер построчно-печатающего устройства для мониторов SJ, FB
LPX SYS	2	Драйвер построчно-печатающего устройства для монитора XM
LS SYS	3	Драйвер построчно-печатающего устройства последовательного типа для мониторов SJ, FB
LSX.SYS	3	Драйвер построчно-печатающего устройства последовательного типа для монитора XM
MT.SYS	9	Файлово-ориентированный драйвер магнитной ленты для мониторов SJ, FB
MTX.SYS	9	Файлово-ориентированный драйвер магнитной ленты для монитора XM
MX.SYS	10	Драйвер гибкого диска одинарной плотности для мониторов SJ и FB
MXX.SYS	10	Драйвер гибкого диска одинарной плотности записи для монитора XM
NL.SYS	2	Драйвер фиктивного устройства для мониторов SJ, FB
NLX.SYS	2	Драйвер фиктивного устройства для монитора XM
PC.SYS	2	Драйвер перфоленточного устройства ввода-вывода для мониторов SJ, FB
PCX.SYS	2	Драйвер перфоленточного устройства ввода-вывода для монитора XM
RK.SYS	3	Драйвер кассетного магнитного диска типа СМ-5400 для мониторов SJ, FB
RKX.SYS	3	Драйвер кассетного магнитного диска типа СМ-5400 для монитора XM
VM.SYS	3	Драйвер расширенной памяти для мониторов SJ, FB
VMX.SYS	2	Драйвер расширенной памяти для монитора XM

1	2	3
Другие системные файлы		
SL.SYS	13	Редактор командной строки для мониторов SJ, FB
SLX.SYS	16	Редактор командной строки для монитора XM
SWAP.SYS	26	Файл свопинга
Вспомогательные программы		
BATCH.SAV	26	Программа пакетной обработки
BINCOM.SAV	24	Программа сравнения двоичных файлов
BUP.SAV	37	Программа получения копий
CREF.SAV	6	Программа перекрестных ссылок
DIR.SAV	19	Программа получения справочника
DUMP.SAV	9	Программа печати
DUP.SAV	45	Программа обслуживания устройства
EDIT.SAV	19	Редактор текста
ELINIT.SAV	7	Программа регистрации ошибок
ERRLOG.REL	9	Программа регистрации ошибок
ERROUT.SAV	18	Программа регистрации ошибок
FORMAT.SAV	21	Программа форматирования дисков
IND.SAV	51	Процессор косвенных управляющих файлов
K13.SAV	55	Экранный редактор текста
LIBR.SAV	24	Библиотекарь
LINK.SAV	49	Редактор связей
MACRO.SAV	60	Ассемблер
MDUP.MT	56	Программа загрузки магнитной ленты
MDUP.SAV	19	Программа обслуживания магнитной ленты
PAT.SAV	10	Программа модификации объектных модулей
PATCH.SAV	10	Программа модификации файлов
PIT.SAV	29	Программа обмена
RESORC.SAV	22	Программа получения информации о состоянии системы
SIPP.SAV	21	Программа модификации загрузочных модулей
SLP.SAV	13	Программа модификации текстовых файлов
SRCCOM.SAV	26	Программа сравнения исходных файлов
SYSMAC.SML	45	Системная макробиблиотека
TFP.SAV	48	Программа форматирования текста
Дополнительные файлы		
DATIME.SAV	3	Стартовый файл задания даты и времени
FMSJ.MAP	20	Карта загрузки дистрибутивного монитора SJ
FMFB.MAP	28	Карта загрузки дистрибутивного монитора FB
FMXM.MAP	31	Карта загрузки дистрибутивного монитора XM
STARTS.COM	1	Стартовый командный файл

1	2	3
Файлы генерации системы		
SYSGEN.COM	80	Косвенный управляющий файл генерации системы
SJFB.ANS	8	Файл ответов для создания мониторов SJ, FB
XM.ANS	4	Файл ответов для создания монитора XM
Отладчики		
ODT.OBJ	8	Отладчик
VDT.OBJ	8	Виртуальный отладчик
Библиотеки		
SYSLIB.OBJ	47	Системная библиотека объектных модулей
SYSMAC.MAC	41	Исходный файл системной макробиблиотеки
Загрузчики		
MBOOT.BOT	1	Первый начальный загрузчик магнитной ленты
MSBOOT.BOT	3	Второй начальный загрузчик магнитной ленты
Контрольные задачи		
CONTBG.MAC	2	Контрольная задача для монитора SJ
CONTFG.MAC	2	Контрольная задача для монитора FB
CONTXM.MAC	17	Контрольная задача для монитора XM
CONT.MAC	2	Контрольная задача
Исходные файлы		
BA.MAC	19	Исходный файл драйвера программы пакетной обработки
BSTRAP.MAC	53	Исходный файл начального загрузчика
DD.MAC	24	Исходный файл драйвера ленты кассетного типа
DP.MAC	8	Исходный файл драйвера пакета дисков (29 Мбайт)
DU.MAC	27	Исходный файл драйвера гибкого и винчестерского дисков
DW.MAC	30	Исходный файл драйвера винчестерского диска
DX.MAC	18	Исходный файл драйвера гибких дисков
DY.MAC	21	Исходный файл драйвера гибких дисков с двойной плотностью
EDTGBL.MAC	28	Исходный файл глобальных имен
EL.MAC	15	Исходный файл драйвера регистратора ошибок
ELCOPY.MAC	14	Исходный файл для регистратора ошибок
ELINIT.MAC	14	Исходный файл для регистратора ошибок
ELTASK.MAC	8	Исходный файл для регистратора ошибок
ERRTXT.MAC	5	Исходный файл для регистратора ошибок
FB.MAC	1	Условный файл для генерации монитора FB
FSM.MAC	30	Исходный файл файлово-ориентированного драйвера магнитной ленты
KMON.MAC	118	Исходный файл клавиатурного монитора

1	2	3
KMOVLY.MAC	203	Исходный оверлейный файл клавиатурного монитора
LD.MAC	44	Исходный файл драйвера логических дисков
LP.MAC	7	Исходный файл драйвера построчно-печатающего устройства
LS.MAC	11	Исходный файл драйвера построчно-печатающего устройства последовательного типа
MTTEMT.MAC	16	Исходный файл мультитерминальных программных запросов
MTTINT.MAC	44	Исходный файл подпрограмм обслуживания прерываний для мультитерминального режима
MX.MAC	24	Исходный файл драйвера одинарной плотности
MY.MAC		Исходный файл драйвера удвоенной плотности
NL.MAC	1	Исходный файл драйвера фиктивного устройства
PC.MAC	3	Исходный файл драйвера перфоленточного устройства ввода-вывода
RMONFB.MAC	142	Исходный файл резидентного монитора FB (XM)
RMONSJ.MAC	68	Исходный файл резидентного монитора SJ
SJ.MAC	1	Условный файл для генерации монитора SJ
TM.MAC	24	Исходный файл драйвера магнитной ленты
TRMTBL.MAC	10	Исходный файл таблицы для мультитерминального режима
TT.MAC	5	Исходный файл драйвера терминала
USR.MAC	64	Исходный файл программы USR
VM.MAC	16	Исходный файл драйвера расширенной памяти
XM.MAC	1	Условный файл для генерации монитора XM
XMSUBS.MAC	32	Исходный файл подпрограмм для монитора XM

3. НАСТРОЙКА И ПРОВЕРКА СИСТЕМЫ

В данном разделе описаны действия, которые необходимо выполнить пользователю по установке дистрибутивной операционной системы для работы на гибких дисках или дисках RK: и DP:. Под установкой операционной системы понимается следующее:

— загрузка дистрибутивного носителя;
— снятие копии с дистрибутивного носителя;
— создание рабочей системы из выбранных компонент;
— модификация (если требуется) рабочей системы;
— снятие копии с рабочей системы;

— проверка работоспособности рабочей системы.

ПРИМЕЧАНИЕ. Команды, используемые в данном документе, приводятся в полной форме. При их выполнении пользователь может использовать краткую форму записи как команд, так и переключателей.

3.1. Установка системы, поставляемой на гибких дисках, для работы на гибких дисках

3.1.1. Загрузка дистрибутивного носителя

Для загрузки системы с диска необходимо выполнить следующие операции:

1) Включить ЭВМ согласно Техническому описанию на ЭВМ.

2) Установить первый дистрибутивный том на привод 0.

3) После появления подсказки @ на экран дисплея ввести команду:

@B
$XX0

где XX — имя устройства (MX или MY)
Ответ: ФОДОС ф B03.00
Дата [дд—мм—гг]?
Ввести текущую дату: день, порядковый номер месяца, год и подать команду <ВК>.
Ответ: время [чч : мм : сс]?
Ввести текущее время: часы, минуты, секунды и подать команду <ВК>.
Ответ: стартовый файл?
Печать: STARTS<ВК>
Ответ: SET TT SCOPE

3.1.2. Копирование дистрибутивного носителя

Пользователю надо выполнить следующие операции:

1) инициализировать справочники томов и осуществить сканирование на плохие блоки всех дисков, предназначенных для копии дистрибутивного носителя. Для этого нужно последовательно устанавливать чистые дискеты на первый привод и подавать команду:

INITIALIZE/BADBLOCKS XX1: <ВК>

где: XX — MX или MY.
Ответ: XX1:/INITIALIZE;ARE YOU SURE?
Печать: Y <ВК>
Ответ: ?DUP—I—NO BAD BLOCKS DETECTED XX1:

2) копировать все файлы с первого дистрибутивного тома на том копии по команде:

SQUEEZE/OUTPUT:XX1: XX0: <ВК>

По окончанию копирования система напечатает точку;

3) записать начальный загрузчик на том копии:

COPY/BOOT XX1: FMONSJ.SYS XX1: <ВК>

Ответ: .

4) удалить том копии с привода 1;

5) для копирования остальных дистрибутивных томов подать команду:

SQUEEZE/WAIT/OUTPUT:XX1: XX0: <ВК>

Ответ: MOUNT OUTPUT VOLUME IN XX1:; CONTINUE?

Установить чистую инициализированную дискету на привод 1 и напечатать: Y <ВК>

Ответ: MOUNT INPUT VOLUME IN XX0:; CONTINUE?

Заменить на нулевом приводе первый дистрибутивный том следующим и напечатать: Y <ВК>

Ответ: MOUNT SYSTEM VOLUME IN XX0:; CONTINUE?

Заменить том на приводе ноль первым дистрибутивным и напечатать: Y <ВК>

Ответ: .

6) удалить том копии с привода 1.

Если процесс создания копии дистрибутивного носителя незавершен, то необходимо повторить операции, начиная с лункта 5).

В дальнейшей работе следует использовать копию дистрибутивной операционной системы. Для этого нужно установ*ить первый том копии либо на привод 1 и загрузить систе*му по команде BOOT XX1:<ВК>, либо на нулевой привод и аппаратно перезагрузить систему.

Перед созданием рабочей системы необходимо отменить защиту файлов на всех томах копии дистрибутивной системы подавая команды:

UNPROTECT/SYSTEM *.* <ВК>

UNPROTECT/SYSTEM XX1:*.* <ВК>

3.1.3. Создание рабочей системы

В дальнейшем под системным томом будем понимать загружаемый том рабочей системы.

Для создания рабочей системы пользователю необходимо выполнить следующие операции (предполагаем, что первый том копии установлен на нулевом приводе):

1) инициализировать тома, предназначенные для рабочей системы по команде:

INITIALIZE/BADBLOCKS XX1: <ВК>

где XX — DX или DY, MX или MY

2) копировать выбранные файлы с первого тома копии на первый том рабочей системы по команде:
COPY/SYSTEM/QUERY XX0: XX1: <ВК>
Ответ: FILES COPIED:
XX0:<имфайл. тип> TO XX1: <имфайл. тип>?
Печать: Y <ВК> для включения файла в систему.
N <ВК> для исключения файла из системы.

После того, как все файлы будут перечислены, система напечатает точку.

3) для копирования файлов с незагружаемых томов на тома рабочей системы следует подать команды:
SET USR NOSWAP <ВК>
COPY/WAIT XX1:<имфайл: тип> XX0:<имфайл. тип> <ВК>
Ответ: MOUNT INPUT VOLUME IN XX1:;CONTINUE?

Поместить том, содержащий файл, который нужно скопировать, на привод 1 и напечатать: Y <ВК>
Ответ: MOUNT OUTPUT VOLUME IN XX0:;CONTINUE?

Поместить на привод ноль том, на который нужно скопировать файл и напечатать: Y <ВК>
Ответ: MOUNT SYSTEM VOLUME IN XX0:;CONTINUE?

Поместить на привод ноль первый том копии и напечатать: Y <ВК>

Повторить эту операцию для копирования всех необходимых файлов на тома рабочей системы.

4) записать на первый том рабочей системы начальный загрузчик по команде:
COPY/BOOT XX1:FMONYY.SYS XX1:<ВК>
где: YY — SJ, FB или XM
Ответ: .

5) поместить первый том рабочей системы (системный том) на привод ноль и перезагрузить систему.

При необходимости пользователь может выполнить модификацию рабочей системы (п. 3.4).

3.1.4. Копирование рабочей системы

Перед копированием рабочей системы в целях ее сохранения следует, во-первых, защитить все файлы от удаления. Для этого используется команда:
PROTECT/SYSTEM *.* <ВК>
для защиты файлов системного тома и команда:
PROTECT/SYSTEM XX1:*.* <ВК>
для защиты файлов остальных томов рабочей системы. Затем нужно инициализировать тома, предназначенные для ко-

пии рабочей системы по команде:
INITIALIZE/BADBLOCKS XX1: <ВК>
Для копирования системного тома нужно использовать *команду: SQUEEZE/OUTPUT:XX1: XX0: <ВК> и затем записать* на копию начальный загрузчик по команде:
COPY/BOOT XX1:FMONYY.SYS XX1: <ВК>
где: YY — SJ, FB или XM.

Для копирования остальных томов нужно использовать команду: SQUEEZE/WAIT/OUTPUT:XX1: XX0: <ВК>

Для детальной информации о последней команде см. п. 3.1.2.

3.1.5. Проверка рабочей системы

Проверка правильности функционирования рабочей системы осуществляется с помощью контрольных задач (см. [1]).

3.2. Установка системы, поставляемой на гибких дисках, для работы на дисках RK: или DP:

3.2.1. Загрузка дистрибутивного носителя

Для загрузки системы с диска см. п. 3.1.1.

3.2.2. Копирование дистрибутивного носителя

Для копирования системы, поставляемой на гибких дисках, на диск RK: или DP: пользователю необходимо выполнить следующие операции:

1) отформатировать чистый диск (если это диск RK:) по команде:
FORMAT RKN: <ВК>
где: N — номер привода.
Ответ: RKN:/FORMAT ARE YOU SURE?
Печать: Y <ВК>
Ответ: ?FORMAT—I—FORMATTING COMPLETE

2) инициализировать том, предназначенный для копии, одновременно проверяя его на плохие блоки, по команде:
INITIALIZE/BADBLOCKS XXN: <ВК>
где: XX — RK или DP
N — номер привода.
Ответ: XXN:/INITIALIZE;ARE YOU SURE?
Печать: Y <ВК>
Ответ: ?DUP—I—NO BAD BLOCKS DETECTED XXN:

3) для копирования первого дистрибутивного тома на том копию подать команду:
SQUEEZE/OUTPUT:XXN: YY0: <ВК>
где: XX — RK или DP
N — номер привода
YY — DX или DY.

По окончании копирования система напечатает точку.

4) для копирования остальных дистрибутивных томов нужно последовательно устанавливать их на привод 1 и подавать команду: COPY/SYSTEM YY1: XXN:<ВК>

По окончанию копирования система напечатает точку.

5) записать начальный загрузчик на том копии:

COPY/BOOT XXN:FMONZZ.SYS XXN: <ВК>

где: ZZ — SJ, FB или XM.

Ответ: .

6) загрузить систему с тома копии по команде:

BOOT XXN: <ВК>

Перед созданием рабочей системы следует отменить защиту файлов на томе копии дистрибутивной системы по команде:

UNPROTECT/SYSTEM *.* <ВК>

3.2.3. Создание рабочей системы

Для создания рабочей системы необходимо выполнить следующие операции:

1) отформатировать чистый диск (если диск RK:) по команде: FORMAT RKM:<ВК>

где: M — номер привода;

2) инициализировать том, предназначенный для рабочей системы, одновременно проверяя его на плохие блоки, по команде: INITIALIZE/BADBLOCKS XXM:<ВК>

где: XX — RK или DP

M — номер привода;

3) копировать выбранные файлы на том рабочей системы по команде: COPY/SYSTEM/QUERY XXN: XXM:<ВК>

Ответ: FILES COPIED

XXN:<имфайл. тип> TO XXM:<имфайл. тип>?

Печать: Y <ВК> для включения файла в систему.

N <ВК> для исключения файла из системы.

После того, как все файлы будут перечислены, система напечатает точку;

4) записать на том рабочей системы начальный загрузчик:

COPY/BOOT XXM:FMONZZ.SYS XXM:<ВК>

где: ZZ — SJ, FB или XM;

5) загрузить рабочую систему по команде:

BOOT XXM: <ВК>

При необходимости пользователь может выполнить модификацию рабочей системы (п. 3.4).

3.2.4. Копирование рабочей системы

Перед копированием рабочей системы в целях ее сохранения следует, во-первых, защитить все файлы от удаления.

Для этого используется команда:
PROTECT/SYSTEM *.*<ВК>

3.2.4.1. Копирование рабочей системы на гибкие диски

Пользователю необходимо выполнить следующие операции:

1) инициализировать справочники томов и осуществить сканирование на плохие блоки всех дисков предназначенных для копии рабочей системы. Для этого нужно последовательно устанавливать чистые дискеты на привод 0 и подавать команду: INITIALIZE/BADBLOCKS YY0:<ВК>
где: YY — DX или DY, MX или MY
Ответ: YY0:/INITIALIZE;ARE YOU SURE?
Печать: Y <ВК>
Ответ: ?DUP — I — NO BAD BLOCKS DETECTED YY0:
.

2) копировать файлы с системного тома на том копии по команде: COPY/SYSTEM/QUERY XXM: YY0:<ВК>
Ответ: FILES COPIED
XXM:<имфайл. тип> TO YY0:<имфайл. тип>?
Печать: Y <ВК> для включения файла в операцию.
N <ВК> для исключения файла из операции.

Если недостаточно места на одном диске для размещения всех файлов, заменить заполненный диск чистым (инициализированным) и подать данную команду еще раз, причем отвечать N для всех файлов, которые уже были скопированы;

3) записать начальный загрузчик на том копии, который должен быть загружаемым, по команде:
COPY/BOOT YY0:FMONZZ.SYS YY0:<ВК>
где: ZZ — SJ, FB или XM
Ответ: .

Заметим, что на этом томе должны быть файлы: SWAP.SYS, монитор, драйвер системного устройства, TT.SYS (если монитор SJ и он без мультитерминальной поддержки).

3.2.4.2. Копирование рабочей системы на другой диск

Пользователю надо выполнить следующие операции:

1) отформатировать чистый диск (если диск RK:) по команде: FORMAT RKN:<ВК>
где: N — номер привода.
Ответ: RKM:/FORMAT ARE YOU SURE?
Печать: Y<ВК>
Ответ: ?FORMAT—I—FORMATTING COMPLETE
.

2) инициализировать том, предназначенный для копии, одновременно проверяя его на плохие блоки, по команде:

INITIALIZE/BADBLOCKS XXN:<BK>
где: XX — RK или DP
N — номер привода.
Ответ: XXN:/INITIALIZE;ARE YOU SURE?
Печать: Y<BK>
Ответ: ?DUP—I—NO BAD BLOCKS DETECTED XXN:

3) копировать все файлы по команде:
SQUEEZE/OUTPUT:XXN: XXM:<BK>
По окончанию копирования система напечатает точку;

4) записать начальный загрузчик на том копии по команде: COPY/BOOT XXN:FMONZZ.SYS XXN:<BK>
где: ZZ — SJ, FB или XM;

5) загрузить систему с тома копии по команде:
BOOT XXN:<BK>

3.2.5. Проверка рабочей системы

Проверка правильности функционирования рабочей системы осуществляется с помощью контрольных задач (см. [1]).

3.3. Установка системы, поставляемой на магнитной ленте, для работы на дисках RK: или DP:

3.3.1. Загрузка дистрибутивного носителя

При загрузке дистрибутивной магнитной ленты на ЭВМ создается минимальная система на диске и под ее управлением выполняются последующие шаги установки системы: копирование дистрибутивного носителя и создание рабочей системы.

Для загрузки дистрибутивной магнитной ленты необходимо выполнить следующие операции:

1) установить магнитную ленту на привод ноль;

2) воспользоваться начальным загрузчиком, описанным в приложении.

На терминале появится:
MSBOOT B03.00
*

3) напечатать имя программы, которая создает минимальную систему на диске: MDUP. MT<BK>
Ответ: MDUP B03.00
*

4) инициализировать диск и проверить его на плохие блоки, предварительно убедившись, что диск установлен, готов и на него разрешена запись: XXN:/Z/B<BK>
где: XXN — имя устройства (RK: или DP:) и номер привода.
Ответ: *

5) напечатать: XXN:A=MT0:<ВК>

По этой команде MDUP.MT копирует на системный том файл свопинга, монитор SJ, драйверы системного устройства, магнитной ленты и построчно-печатающего устройства, программы PIP, DUP, DIR. Когда файлы будут скопированы, MDUP загружает в память минимальную систему с диска и передает ей управление. Система печатает:

ФОДОС ф В03.00
?KMON—F—INVALID COMMAND

Сообщение об ошибке появляется потому, что мониторы ФОДОС-2 после загрузки обращаются к программе задания даты и времени DATIME.SAV, которая не входит в минимальную систему. После копирования DATIME.SAV с ленты на диск и перезагрузки системы сообщение об ошибке появляться не будет.

Для копирования оставшихся файлов дистрибутивного набора на диск следует подать команду:

COPY/SYSTEM/NOREPLACE MT0: XXN:<ВК>

По окончанию копирования система напечатает точку.

3.3.2. Копирование дистрибутивного носителя

Пользователю необходимо выполнить следующие операции.

1) заменить дистрибутивную ленту на нулевом приводе чистой магнитной лентой;

2) подать следующие команды:

ASSIGN MT0: TAP:<ВК>
ASSIGN XXN: DIS:<ВК>

где: XXN — имя устройства (RK или DP) и номер привода системного тома.

@DISMT <ВК>

Команды из косвенного командного файла будут печататься на терминале.

3.3.3. Создание рабочей системы

Так как на диске уже имеется полный дистрибутивный набор, то создание рабочей системы сводится просто к удалению не нужных пользователю файлов по команде:

DELETE/SYSTEM/QUERY *.*<ВК>
Ответ: FILES DELETED
DK:<имфайл.тип>?
Печать: Y<ВК> для удаления файла из системы
N<ВК> для того, чтобы оставить файл в системе.

После того, как все файлы будут перечислены, система

напечатает точку. После этого рекомендуется сжать том с рабочей системой по команде SQUEEZE DK:<ВК>

При необходимости пользователь может выполнить модификацию рабочей системы (п. 3.4).

3.3.4. Копирование рабочей системы

Перед копированием рабочей системы в целях ее сохранения следует, во-первых, защитить все файлы от удаления. Для этого используется команда: PROTECT/SYSTEM *.*<ВК>

Затем следует выполнить следующие операции:

1) установить чистую магнитную ленту на нулевом приводе;

2) инициализировать магнитную ленту по команде:
INITIALIZE/FILE:MBOOT.BOT MT0:<ВК>
Ответ: MT0:/INITIALIZE;ARE YOU SURE?
Печать: Y <ВК>
Ответ: .

3) копировать файлы на ленту в определенном порядке.
COPY MSBOOT.BOT MT0:MSBOOT.BOT <ВК>
COPY MDUP.MT MT0:MDUP.MT/POSITION:—1 <ВК>

Аналогично копируются файлы:
SWAP.SYS
FMONSJ.SYS
TT.SYS (если монитор без мультитерминальной поддержки)
Драйвер системного устройства
MT.SYS
LP.SYS
PIP.SAV
DUP.SAV
DIR.SAV

Остальные файлы рабочей системы копируются в произвольном порядке по команде:
COPY/SYSTEM/QUERY DK: MT0:/POSITION:—1 <ВК>
Ответ: FILES COPIED:
DK:<имфайл.тип> TO MT0:<имфайл.тип>
Печать: Y <ВК> для включения файла в операцию.
N <ВК> для исключения файла из операции.

После того, как все файлы будут перечислены, система напечатает точку.

3.3.5. Проверка рабочей системы

Проверка правильности функционирования рабочей системы осуществляется с помощью контрольных задач (см. [1]).

3.4. Модификация рабочей системы

Пользователь может, если требуется, изменить некоторые особенности операционной системы. Перечень этих особенностей приведен в табл. 3.

Таблица 3

ИЗМЕНЕНИЕ ОСОБЕННОСТЕЙ ОПЕРАЦИОННОЙ СИСТЕМЫ

Особенность	Пункт описания изменения	Описание
1	2	3
Использование терминала в качестве устройства печати по умолчанию	3.4.1	Если аппаратная конфигурация не включает построчно-печатающее устройство, то можно изменить устройство вывода по умолчанию (которое используют некоторые команды КЯС) на терминал. Для этого следует изменить стартовый командный файл STARTS.COM
Изменение числа абсолютных базовых адресов программных секций разрешенных в редакторе связей	3.4.2	Обычно по переключателю /Q редактора связей можно указать до 8 абсолютных базовых адресов программных секций. можно модифицировать LINK для изменения этого числа
Выделение оверлейных драйверов из библиотеки SYSLIB.OBJ	3.4.3	Если используются только программы, написанные на языке АССЕМБЛЕР, и программы могут быть оверлейными, то можно выделить из SYSLIB.OBJ оверлейные драйверы и создать библиотеку меньшего размера
Включение устройств в систему	3.4.4	При загрузке системы начальный загрузчик включает в систему периферийные устройства аппаратной конфигурации, если соответствующие драйверы имеются и достаточно свободных мест в таблицах монитора. Можно включить устройство в систему используя команды REMOVE и INSTALL. Можно также: — изменить адреса регистров состояний и векторов следующих устройств: LP:, LS:,DD:,RK:,DX:,MX:,DY:,MY:,DU:,DW: — включить аппаратный драйвер магнитной ленты
Модификация BATCH для увеличения свободной области на системном томе	3.4.5	Можно модифицировать BATCH так, чтобы при выполнении пакета заданий BATCH загружала системные программы не с системного устройства SY:, а с DK:. В результате этого можно хранить системные программы (PIP, DIR и т. д.) не на системном томе

1	2	3
Работа системы в неполной памяти	3.4.6	Можно модифицировать систему так, чтобы она выполнялась в памяти меньшей, чем имеется в аппаратной конфигурации
Введение 22-х битной адресации	3.4.7	Можно модифицировать монитор XM для введения режима 22-х битной адресации
Установка верхнего предела на размер файла	3.4.8	Можно модифицировать монитор так, чтобы создаваемые файлы не могли превысить определенного размера
Изменение устройства по умолчанию для косвенных командных файлов	3.4.9	По умолчанию монитор отыскивает косвенные командные файлы на DK:. Можно модифицировать монитор для изменения этой особенности
Изменение типа файла по умолчанию для косвенных командных файлов	3.4.10	По умолчанию тип файла косвенных командных файлов — COM. Можно модифицировать монитор для изменения этой особенности
Изменение устройства по умолчанию для команды FRUN	3.4.11	Если подается команда FRUN, то по умолчанию монитор отыскивает основное задание на DK:. Можно модифицировать монитор для изменения этой особенности
Изменение типа файла по умолчанию для команды FRUN	3.4.12	По умолчанию тип файла для команды FRUN —.REL. Можно модифицировать монитор для изменения этой особенности
Использование командами КЯС «E» и «D» памяти выше фоновой области	3.4.13	Обычно монитор позволяет проверять (команда E)и модифицировать (команда D) лишь ячейки из фоновой области памяти. Можно модифицировать монитор для отмены этого ограничения
Изменение числа строк в листинге редактора связей	3.4.14	Число строк в листинге редактора связей 60 (десятичное). Можно модифицировать файл LINK.SAV для изменения этого числа
Изменение устройства загрузки по умолчанию процессора косвенных управляющих файлов (IND)	3.4.15	Процессор косвенных управляющих файлов загружается по умолчанию с системного устройства SY:. Можно модифицировать монитор для изменения этой особенности

3.4.1. Использование терминала в качестве устройства печати по умолчанию

Если аппаратная конфигурация не включает построчно-печатающее устройство, то для изменения устройства вывода по умолчанию с LP: на TT: для команд КЯС (например, PRINT, DUMP) следует добавить в стартовый командный файл команду: ASSIGN TT LP <ВК>

3.4.2. Изменение числа абсолютных базовых адресов программных секций, разрешенных в редакторе связей.

По переключателю /Q редактора связей LINK можно указать до 8-ми абсолютных базовых адресов программных секций. Для изменения этого числа следует выполнить следующую модификацию файла LINK.SAV.

```
RUN SIPP <ВК>
*LINK.SAV <ВК>
SEGMENT?0 <ВК>
BASE?0 <ВК>
OFFSET?1440 <ВК>
SEGMENT BASE OFFSET OLD NEW?
000000    000000 1440    10   NNN  <ВК>
000000    000000 1442    ...  СУ/Y <ВК>
*СУ/С
```

NNN — должно быть от 1 до 177 (восьмеричное).

3.4.3. Выделение оверлейных драйверов из SYSLIB.OBJ

Для построения библиотеки, которая включает только оверлейные драйверы, необходимые при создании оверлейных программ, нужно выполнить следующее:

1) выделить из SYSLIB оверлейные драйверы для памяти, меньшей 28К слов:

```
LIBRARY/EXTRACT SYSLIB OHANDL <ВК>
GLOBAL?$OVRH <ВК>
GLOBAL?  <ВК>
```

2) выделить из SYSLIB оверлейные драйверы для памяти свыше 28К слов:

```
LIBRARY/EXTRACT SYSLIB VHANDL <ВК>
GLOBAL?$OVRH <ВК>
GLOBAL?  <ВК>
```

3) объединить полученные два файла в новую библиотеку, которая может иметь любое имя, включая SYSLIB:

```
LIBRARY/REMOVE/CREATE NEWLIB VHANDL,OHANDL <ВК>
GLOBAL?$OVRH <ВК>
GLOBAL?  <ВК>
```

где: NEWLIB — имя новой библиотеки.

Если нужно создать библиотеку, включающую только один вид оверлейных драйверов, то следует подать команду:

LIBRARY/CREATE NEWLIB XHANDL <BK>
где X — O для OHANDL и V для VHANDL.

3.4.4. Включение устройств в систему

При загрузке системы начальный загрузчик включает в систему периферийные устройства аппаратной конфигурации, если соответствующие драйверы имеются и достаточно свободных мест в таблицах монитора. Если драйверов больше, чем свободных мест, то начальный загрузчик использует определенную, основанную на приоритете, схему для включения устройства в систему.

Чтобы определить, какие устройства включены в систему и сколько имеется свободных мест, используется команда SHOW.

Если при загрузке системы устройство не включается в систему, это значит, что либо нет свободного места в таблицах монитора, либо этого устройства нет в аппаратной конфигурации, либо нет драйвера этого устройства. Для включения устройства в систему следует убедиться, что соответствующий драйвер находится на системном томе, и что это устройство есть в аппаратной конфигурации. Если нет свободного места в таблицах монитора, то следует удалить неиспользуемое устройство по команде REMOVE и включить необходимое устройство по команде INSTALL.

Пример:

REMOVE LS:<BK>
INSTALL LP:<BK>

Если какое-либо из периферийных устройств (LP:,LS:,DD:, RK:,DX:,MX:,DY:,MY:,DU:) имеет нестандартные адреса регистров состояний или векторов, то перед использованием этого устройства следует подать команду:

SET XX: CSR=NNNNNN <BK> или
SET XX: VECTOR=MMM <BK>
где: XX — имя устройства
NNNNNN — действительный адрес регистра состояния
MMM — действительный адрес вектора.

Если устройство обслуживается вторым контроллером, то команды следующие:

SET XX: CSR2=NNNNNN <BK> или
SET XX: VEC2=MMM <BK>

Данные команды изменяют файл драйвера указанного устройства. Новые значения адресов остаются в действии даже после перезагрузки системы.

Для использования аппаратного драйвера магнитной ленты следует выполнить генерацию системы, т. к. дистрибутив-

ный набор содержит лишь файлово-ориентированный драйвер магнитной ленты. Для включения сгенерированного аппаратного драйвера в систему следует переименовать его по команде:

RENAME MTHD.SYS MT.SYS <ВК>

и затем перезагрузить систему.

3.4.5. Модификация BATCH для увеличения свободной области на системном томе

Для увеличения свободной области на системном томе при использовании программы пакетной обработки можно сделать так, чтобы BATCH обращался к системным программам, расположенным не на SY:, а на DK:.

Для этого необходимо, во-первых, выполнить следующую модификацию файла BATCH.SAV:

```
RUN SIPP <ВК>
*BATCH.SAV <ВК>
BASE?26722 <ВК>
OFFSET?NNNNNN <ВК>
  BASE OFFSET                 OLD      NEW
  26722 NNNNNN                40       125 <ВК>
  26722 NNNNNN+2(или +1)      40       СУ/Y <ВК>
*СУ/С
```

где NNNNNN принимает следующие значения для программ.

DIR — 1367
MACRO — 2007
FORTRA — 2020
LINK — 2037
PIP — 2052
BASIC — 2071

Затем следует скопировать программы, для которых сделаны изменения, на другой том и удалить их с системного тома и, в заключение, подать команду ASSIGN для назначения новому тому логического имени DK:.

3.4.6. Работа системы в неполной памяти

При загрузке мониторов ФОДОС-2 размер памяти, имеющейся в аппаратной конфигурации определяется системным загрузчиком автоматически. Если требуется, чтобы система выполнялась в неполной памяти (например, мониторы SJ и FB могут использовать нижнюю часть в 16 или 20К слов блока памяти в 24К слов), то необходимо выполнить следующую модификацию мониторов SJ и FB. Заметим, что нельзя модифицировать монитор XM, т. к. он требует 32К слов памяти.

```
RUN SIPP <ВК>
*FMONXX.SYS <ВК>
где XX — FB или SJ
BASE?1000 <ВК>
OFFSET?30 <ВК>
BASE    OFFSET      OLD       NEW?
1000      30        407       240  <ВК>
1000      32        13704     12704 <ВК>
1000      34        177570    NNNNNN <ВК>
1000      36        42704     СУ/Y <ВК>
*СУ/С
```

где NNNNNN принимает следующие значения:

40000	Для памяти	8К слов	110000	для памяти	18К слов
44000	»	9К слов	120000	»	20К слов
50000	»	10К слов	124000	»	21К слов
54000	»	11К слов	130000	»	22К слов
60000	»	12К слов	134000	»	23К слов
64000	»	13К слов	140000	»	24К слов
70000	»	14К слов	144000	»	25К слов
74000	»	15К слов	150000	»	26К слов
100000	»	16К слов	154000	»	27К слов
104000	»	17К слов	160000	»	28К слов

Затем следует записать начальный загрузчик модифицированного монитора по команде COPY/BOOT и перезагрузить систему. Система будет выполняться в неполной памяти до тех пор пока не будет сделана новая модификация монитора.

3.4.7. Введение 22-битной адресации

Система ФОДОС-2 позволяет на ЭВМ «ЭЛЕКТРОНИКА МС 1201» ввести 22-битную адресацию. Для этого необходимо выполнить следующую модификацию монитора XM.

```
RUN SIPP <ВК>
*FMONXM.SYS <ВК>
BASE? 0 <ВК>
OFFSET?..FQ22 <ВК>
BASE        OFFSET      OLD       NEW
000000      ..FQ22      10YY      4YY <ВК>
000000      ..FQ22+2    XXXXXX    СУ/Y <ВК>
*СУ/С
```

Здесь ..FQ22 представляет собой адрес, который определяется из карты загрузки монитора.

3.4.8. Установка верхнего предела на размер файла

Мониторы ФОДОС-2 при выполнении макрокоманды .ENTER с неопределенной длиной файла резервируют либо половину свободной области на томе, либо всю вторую по величине свободную область в зависимости от того, что больше.

Если необходимо ограничить размер файла, создаваемого таким образом, то следует выполнять модификацию файла монитора.

```
RUN SIPP <ВК>
*FMONXX.SYS <ВК>
где XX — SJ, FB или XM.
BASE? $RMON <ВК>
OFFSET? 314 <ВК>
  BASE    OFFSET  OLD      NEW?
  $RMON   314     177777   NNNNNN <ВК>
  $RMON   316     XXXXXX   СУ/Y <ВК>
*СУ/С
```

Здесь $RMON представляет собой адрес, который определяется из карты загрузки монитора, NNNNNN — восьмеричное число, определяющее максимальный размер файлов в блоках.

3.4.9. Изменение устройства по умолчанию для косвенных командных файлов

Обычно при вызове косвенного командного файла (по команде @ ИМФАЙЛ) устройством по умолчанию, на котором монитор отыскивает командный файл, является DK:. Для изменения этой особенности необходимо выполнить следующую модификацию:

```
RUN SIPP <ВК>
*FMONXX.SYS <ВК>
где XX — SJ, FB или XM.
BASE? 0 <ВК>
OFFSET?..ATDK <ВК>
  BASE    OFFSET     OLD        NEW
  000000  ..ATDK     15270      ;R <ВК>
  000000  ..ATDK     <DK>       ;RNNN <ВК>
  000000  ..ATDK+2   <AW1>      СУ/Y <ВК>
*СУ/С
```

Здесь ..ATDK представляет собой адрес, который определяется из карты загрузки монитора. NNN — новое имя устройства по умолчанию.

3.4.10. Изменение типа файла по умолчанию для косвенных командных файлов

Обычно косвенные командные файлы имеют по умолчанию тип файла .COM. Для изменения этой особенности необходимо выполнить следующую модификацию.

```
RUN SIPP <ВК>
*FMONXX.SYS <ВК>
где XX — SJ, FB или XM.
BASE? 0 <ВК>
OFFSET? ..ATFX <ВК>
  BASE    OFFSET    OLD       NEW?
  000000  ..ATFX    12445     ;R <ВК>
  000000  ..ATFX    <COM>     ;RNNN <ВК>
  000000  ..ATFX+2  <XXX>     СУ/Y <ВК>
*СУ/С <ВК>
```

Здесь ..ATFX представляет собой адрес, который определяется из карты загрузки монитора. NNN — новый тип файла по умолчанию.

3.4.11. Изменение устройства по умолчанию для команды FRUN

Обычно при загрузке основного задания (по команде FRUN) устройством по умолчанию, на котором монитор отыскивает файл, является DK:. Для изменения этой особенности необходимо выполнить следующую модификацию:

```
RUN SIPP <ВК>
*FMONXX.SYS <ВК>
где XX — FB или HM.
BASE? 0 <ВК>
OFFSET?..FRDK <ВК>
  BASE    OFFSET    OLD       NEW?
  000000  ..FRDK    015270    ;R <ВК>
  000000  ..FRDK    <DK>      ;RNNN <ВК>
  000000  ..FRDK+2  <XXX>     СУ/Y
*СУ/С
```

Здесь ..FRDK представляет собой адрес, который определяется из карты загрузки монитора. NNN — новое имя устройства по умолчанию.

3.4.12. Изменение типа файла по умолчанию для команды FRUN

Обычно при загрузке основного задания (по команде FRUN) типом файла по умолчанию является .REL. Для из-

менения этой особенности необходимо выполнить следующую модификацию:

```
RUN SIPP <ВК>
*FMONXX.SYS <ВК>
```

где XX — FB или XM.

```
BASE? 0 <ВК>
OFFSET? ..FRUX <ВК>
  BASE    OFFSET     OLD        NEW?
  000000  ..FRUX     070524     ;R <ВК>
  000000  ..FRUX     <REL>      ;RNNN <ВК>
  000000  ..FRUX+2   <ANG>      СУ/Y <ВК>
*СУ/С
```

Здесь ..FRUX представляет собой адрес, который определяется из карты загрузки монитора. NNN — новый тип файла по умолчанию.

3.4.13. Использование командами КЯС «E» и «D» памяти выше фоновой области

Обычно монитор позволяет проверять (команда E) и модифицировать (команда D) лишь ячейки из фоновой области памяти. Для отмены этого ограничения необходимо выполнить следующую модификацию.

```
RUN SIPP <ВК>
*FMONXX.SYS <ВК>
```

где XX — SJ, FB или XM.

```
BASE? 0 <ВК>
OFFSET?..EMON <ВК>
  BASE    OFFSET     OLD        NEW?
  000000  ..EMON     103041     240 <ВК>
  000000  ..EMON+2   103007     СУ/Y <ВК>
*СУ/С
```

Здесь ..EMON представляет собой адрес, который определяется из карты загрузки монитора.

3.4.14. Изменение числа строк в листинге редактора связей

Число строк в листинге редактора связей 60 (десятичное). Для изменения этого числа необходимо выполнить следующую модификацию:

```
RUN SIPP <ВК>
*LINK.SAV <ВК>
SEGMENT? 0 <ВК>
BASE? 0 <ВК>
OFFSET? 1436 <ВК>
```

```
SEGMENT  BASE    OFFSET  OLD   NEW?
000000   000000  1436    74    NNN <ВК>
000000   000000  1440    ...   СУ/Y <ВК>
*СУ/С
```

Здесь NNN — требуемое число (восьмеричное) строк в листинге редактора связей.

3.4.15. Изменение устройства загрузки по умолчанию процессора косвенных управляющих файлов (IND)

После команды SET KMON IND процессор косвенных управляющих файлов (IND) загружается по умолчанию с системного устройства. Для изменения этой особенности необходимо выполнить следующую модификацию:

```
RUN SIPP <ВК>
*FMONXX.SYS <ВК>
```

где XX — SJ, FB или XM.

```
BASE? 0 <ВК>
OFFSET? ..INDN <ВК>
BASE?    OFFSET    OLD       NEW?
000000   ..INDN     75250     ;R <ВК>
000000   ..INDN     <SY>      ;РУСТ <ВК>
000000   ..INDN+2   <IND>     СУ/Y <ВК>
*СУ/С
```

Здесь ..INDN — представляет собой адрес, который определяется из карты загрузки монитора. УСТ — имя устройства, с которого будет загружаться IND.

4. ДОПОЛНИТЕЛЬНЫЕ ВОЗМОЖНОСТИ

Если пользователю требуется создать новую операционную систему, отличную от поставляемой, то следует выполнить генерацию системы. Под генерацией системы ФОДОС-2 понимается создание управляющих программ (мониторов и драйверов) операционной системы с помощью косвенного управляющего файла SYSGEN.COM.

4.1. Структура процесса генерации

Процесс генерации состоит из следующих этапов:

— подготовка к генерации;
— запуск косвенного управляющего файла SYSGEN.COM и ответы на его вопросы;
— трансляция и редактирование.

4.1.1. Подготовка к генерации

Перед генерацией операционной системы пользователю необходимо определить:

— все периферийные устройства, которые должна обслуживать операционная система;

— адреса регистров состояний и векторов этих периферийных устройств;

— функции, которые должен осуществлять монитор.

4.1.2. Выполнение SYSGEN.COM

Для запуска процесса генерации следует подать команду: RUN IND.SAV SYSGEN.COM <ВК> или подать две команды:
SET KMON IND <ВК> и @SYSGEN <ВК>

Ниже приведены все вопросы и ответы, возможные при работе с SYSGEN.COM. При однократном выполнении SYSGEN некоторых из этих вопросов может не быть, т. к. их появление зависит от ответов на предыдущие вопросы. В результате выполнения SYSGEN создает выходные файлы:

SYSGEN.CND — файл параметров, определяющий выбранные в диалоге SYSGEN особенности;
SYSGEN.TBL — файл, определяющий таблицы монитора для периферийных устройств;
SYSGEN.BLD — командный файл создания мониторов и драйверов;
SYSGEN.MON — командный файл создания мониторов;
SYSGEN.DEV — командный файл создания драйверов;
SYSGEN.ANS — необязательный файл ответов на вопросы SYSGEN.

4.1.2.1. Диалог SYSGEN

После запуска управляющего файла SYSGEN.COM на терминале печатаются вопросы, описанные ниже.

ХОТИТЕ ИСПОЛЬЗОВАТЬ СОЗДАННЫЙ РАНЕЕ ФАЙЛ ОТВЕТОВ (N)?

Если напечатать Y<ВК>, то ответы на вопросы диалога будут браться из файла ответов. Причем, вопросы печататься не будут. Файл ответов может быть либо создан ранее в процессе диалога SYSGEN, либо взят из дистрибутивного набора. Дистрибутивный набор содержит два файла ответов SJFB.ANS и XM.ANS для создания дистрибутивных мониторов и драйверов.

КАКОЙ ФАЙЛ ОТВЕТОВ ХОТИТЕ ИСПОЛЬЗОВАТЬ (SYSGEN.ANS)?

Следует напечатать спецификацию файла ответов и <ВК>. Если указанный файл не будет найден, имеет недопустимое имя или произошла ошибка чтения, то появится со-

ответствующее сообщение об ошибке.

ХОТИТЕ ВЕРНУТЬСЯ В НАЧАЛО (N)?

Этот вопрос появится, если файл ответов не будет найден. Если ответить Y<ВК>, то на терминале появится первый вопрос. Если N<ВК> или <ВК>, то появится следующий вопрос.

ХОТИТЕ СОЗДАТЬ ФАЙЛ ОТВЕТОВ (N)?

Если ответить Y<ВК>, то будет создаваться файл, содержащий ответы на вопросы SYSGEN. Этот файл ответов может быть использован позднее при последующей генерации системы.

УКАЖИТЕ СПЕЦИФИКАЦИЮ ФАЙЛА ОТВЕТОВ (SYSGEN.ANS)?

Следует указать имя и номер привода устройства, на котором должен размещаться файл ответов, имя файла и его тип.

Если указанный файл уже существует, то появится следующий вопрос:

ХОТИТЕ СОЗДАТЬ НОВЫЙ ФАЙЛ SYSGEN.ANS (N)?

Если ответить Y<ВК>, то существующий файл с таким именем будет удален.

После этого SYSGEN, прежде чем открыть выходные файлы, проверяет, существуют ли защищенные файлы с таким же именем. Если какой-либо защищенный файл существует, то SYSGEN печатает сообщение:

?SYSGEN—F—ЗАЩИЩЕННЫЙ ФАЙЛ <имфайл.тип> УЖЕ СУЩЕСТВУЕТ

и завершает свою работу

На любой из последующих трех вопросов следует ответить Y<ВК> или N<ВК>, в зависимости от того, какой монитор необходимо сгенерировать:

ХОТИТЕ МОНИТОР SJ (Y)?

ХОТИТЕ МОНИТОР FB (Y)?

ХОТИТЕ МОНИТОР XM (Y)?

Если на все три вопроса ответили N<ВК>, то SYSGEN печатает сообщение: ?SYSGEN—E—МОНИТОР НЕ ВЫБРАН и появится следующий вопрос:

ХОТИТЕ ВЕРНУТЬСЯ К ПЕРВОМУ ВОПРОСУ О ТИПЕ МОНИТОРА (Y)?

Если ответить N<ВК>, то SYSGEN завершит работу.

Следующие вопросы определяют особенности генерируемых мониторов:

1. ХОТИТЕ ПОДДЕРЖКУ ТАЙМЕРА МОНИТОРОМ SJ (N)?

Если ответить Y<ВК>, то монитор SJ сможет обрабатывать макрокоманды .MRKT и .CMKT, а также увеличивать дату в полночь. Эта поддержка, однако, добавляет сколо 300 (десятичное) слов к резидентному монитору.

2. ХОТИТЕ ПОДДЕРЖИВАТЬ ТАЙМ-АУТ ПЕРИФЕРИЙНЫХ УСТРОЙСТВ (N)?

Если система генерируется на обслуживание сети ЭВМ, то следует ответить Y<ВК>. Эта поддержка позволяет драйверам использовать программный запрос .MRKT.

3. ХОТИТЕ СООБЩЕНИЯ ОБ ОШИБКАХ ПРИ СИСТЕМНЫХ ОШИБКАХ ВВОДА-ВЫВОДА (Y)?

Следует ответить Y<ВК> для того, чтобы вместо останова системы при обнаружении монитором SJ ошибки в операции ввода-вывода печаталось сообщение об ошибке.

4. ХОТИТЕ ПОДДЕРЖКУ СИСТЕМНЫХ ЗАДАНИЙ (N)?

Поддержка системных заданий позволяет выполнять одновременно основное задание, фоновое задание и до 6-ти системных заданий. Следует ответить Y<ВК>, если регистрация ошибок или/и программа формирования очереди на печать будут выполняться как системные задания.

5. ХОТИТЕ ИСПОЛЬЗОВАТЬ ПРОГРАММНЫЙ ЗАПРОС .SPCPS (N)?

Если от монитора FB (или XM) требуется обслуживание программного запроса .SPCPS, то следует ответить Y<ВК>.

6. ХОТИТЕ МУЛЬТИТЕРМИНАЛЬНУЮ ПОДДЕРЖКУ (N)?

Если требуется сгенерировать монитор SJ, FB или XM, который обслуживал бы более одного терминала (мультитерминальный режим работы), то следует ответить Y<ВК>. В мультитерминальном режиме фоновое задание использует системный терминал, а основное задание может использовать как системный, так и любой другой дополнительный терминал. Общее число терминалов должно быть не более 8.

7. ХОТИТЕ АСИНХРОННОЕ ИЗМЕНЕНИЕ СОСТОЯНИЯ ТЕРМИНАЛА (Y)?

При ответе Y<ВК> программа пользователя может опрашивать состояние терминала при помощи слова асинхронного состояния терминала. Адрес этого слова определяется во время присоединения терминала к заданию по запросу .MTATCH. Монитор изменяет содержимое этого слова при изменении состояния терминала. Разряды слова асинхронного состояния имеют следующие значения:

— 15 разряд устанавливается при двойном нажатии СУ/С;
— 14 разряд устанавливается при возможности ввода с терминала;

— 13 разряд устанавливается, если буфер вывода пуст.

11. ХОТИТЕ ПОДДЕРЖИВАТЬ ТАЙМ-АУТ ТЕРМИНАЛОВ (Y)?

Если ответить Y<ВК>, то монитор через определенные интервалы времени будет устанавливать готовность терминалов, которые находятся в состоянии «не готово».

12. УКАЖИТЕ РАЗМЕР (В ЛИТЕРАХ) БУФЕРА ВЫВОДА ДЛЯ ТЕРМИНАЛА (40):

Размер буфера вывода для терминалов может быть от 10 до 134 знаков. Каждая литера буфера добавляет N байтов к резидентной части монитора, где — N общее число терминалов в системе.

13. УКАЖИТЕ РАЗМЕР (В ЛИТЕРАХ) БУФЕРА ВВОДА ДЛЯ ТЕРМИНАЛОВ (134):

Размер буфера ввода для терминалов может быть от 74 до 254 знаков. Каждая литера буфера добавляет N байтов к резидентной части монитора, где — N общее число терминалов в системе.

14. ХОТИТЕ ИСПОЛЬЗОВАТЬ ПРОГРАММНЫЙ ЗАПРОС .FETCH МОНИТОРОМ XM (Y)?

Если ответить N<ВК>, то перед использованием фоновым заданием любого периферийного устройства, драйвер устройства должен быть загружен в память. Если ответить Y<ВК>, тогда драйвер периферийного устройства может быть выбран фоновым заданием, по мере необходимости.

15. ХОТИТЕ ПОДДЕРЖКУ «КАЛЕНДАРЯ» (N)?

Если ответить N<ВК>, то при работе системы непрерывно длительный промежуток времени нужно вновь устанавливать дату и время в начале каждого месяца. Если ответить Y<ВК>, то система будет делать это автоматически.

20. ХОТИТЕ ПОДДЕРЖКУ ВСЕХ КОМАНД КЛАВИАТУРНОГО МОНИТОРА (Y)?

Набор команд клавиатурного монитора состоит из трех подмножеств: команд обслуживающих программ, команд языков и минимального набора. Если ответить Y<ВК>, то монитор будет поддерживать выполнение всех команд. Если ответить N<ВК>, то SYSGEN будет спрашивать о поддержке каждого подмножества команд. Если не выбрать ни одного подмножества, то монитор будет поддерживать только команду RUN.

21. ХОТИТЕ ПОДДЕРЖКУ КОМАНД ОБСЛУЖИВАЮЩИХ ПРОГРАММ (Y)?

Набор команд обслуживающих программ состоит из следующих:

BACKUP	BOOT	COPY	CREATE
DELETE	DIFFERENCE	DIRECTORY	DUMP
EDIT	FORMAT	INITIALIZE	PRINT
PROTECT	RENAME	SHOW	SQUEEZE
TYPE	UNPROTECT		

Следует ответить Y<ВК> или N<ВК> в зависимости от того, включать или нет в монитор поддержку данного набора команд.

22. ХОТИТЕ ПОДДЕРЖКУ КОМАНД ЯЗЫКОВ (Y)?

Набор команд языков состоит из следующих:

BASIC	COMPILE	DIBOL	EXECUTE
FORTRAN	LIBRARY	LINK	MACRO

Следует ответить Y<ВК> или N<ВК> в зависимости от того, включать или нет в монитор поддержку данного набора команд.

23. ХОТИТЕ ПОДДЕРЖКУ КОМАНД МИНИМАЛЬНОГО НАБОРА (Y)?

Командами минимального набора являются следующие:

ABORT	ASSIGN	B	CLOSE
D	DATE	DEASSIGN	DISMOUNT
E	FRUN	GET	INSTALL
LOAD	MOUNT	R	REENTER
REMOVE	RESET	SRUN	START
SUSPEND	TIME	UNLOAD	

Следует ответить Y<ВК> или N<ВК> в зависимости от того, включать или нет в монитор поддержку данного набора команд.

26. ХОТИТЕ ПОДДЕРЖКУ СТАРТОВОГО ФАЙЛА DATIME (Y)?

Если ответить Y<ВК>, то при загрузке системы монитор будет обращаться к программе задания даты и времени DATIME.SAV.

27. ХОТИТЕ ПОДДЕРЖКУ КОМАНД ПЛАВАЮЩЕЙ ЗАПЯТОЙ (N)?

ЭВМ имеет блок плавающей запятой, позволяющий выполнять 46 указанных команд. Следует ответить Y<ВК>, если в системе их выполнение предполагается.

28. ХОТИТЕ ПРОВЕРКУ ЧЕТНОСТИ ПАМЯТИ (N)?

ЭВМ имеет устройство проверки четности памяти. Если ответить Y<ВК>, то при ошибках четности памяти будут печататься сообщения об этом и, если выбрана регистрация ошибок, то указанные ошибки также будут зарегистрированы. Если ответить N<ВК>, то при появлении указанной ошибки будет происходить останов системы.

появлении указанной ошибки будет происходить останов системы.

29. ХОТИТЕ СООБЩЕНИЯ О НАРУШЕНИИ ПИТАНИЯ (N)?

При нарушении питания происходит останов системы. Если ответить Y<ВК>, то будет печататься сообщение, объясняющее причину этого останова.

30. ХОТИТЕ ИСПОЛЬЗОВАТЬ ПАКЕТНУЮ ОБРАБОТКУ (N)?

При ответе Y<ВК> в системе будет предусмотрено обслуживание программы BATCH.

31. ХОТИТЕ ПОДДЕРЖКУ РЕГИСТРАТОРА ОШИБОК (N)?

При процессе регистрации ошибок происходит регистрация и запись в виде файла ошибок, встречающихся в операциях ввода-вывода и ошибок четности памяти (если имеется соответствующая поддержка).

32. СКОЛЬКО ПРИВОДОВ ДИСКОВ ДОЛЖЕН ОБСЛУЖИВАТЬ РЕГИСТРАТОР ОШИБОК (10)?

Следует указать общее количество приводов дисков (включая диски DX: и DY:, MX: и MY:). Допустимый ответ от 1 до 34 (десятичное).

Следующие вопросы определяют периферийные устройства, поддержку которых следует включить в генерируемую систему ФОДОС-2.

ВВЕДИТЕ ИМЯ УСТРОЙСТВА [DD]:

Для получения списка всех устройств следует подать ?<ВК>. Для включения устройства в систему следует напечатать двухбуквенное имя устройства и <ВК>. После чего имя устройства будет занесено в таблицы монитора и драйвер этого устройства будет создан в процессе генерации.

ВВЕДИТЕ СЛЕДУЮЩЕЕ ИМЯ УСТРОЙСТВА [DD]:

Если выбор устройств завершен, то следует напечатать <ВК>.

33. ХОТИТЕ ПОДДЕРЖКУ ВТОРОГО КОНТРОЛЛЕРА УСТРОЙСТВА <ИМУСТ> (N)?

SYSGEN задает этот вопрос, если выбрана поддержка устройств: DX:, MX:, DY:, MY:, DD:. Каждый контроллер этих устройств поддерживает только два привода. Следует ответить Y<ВК>, если имеется контроллер, обслуживающий два дополнительных привода. Однако систему ФОДОС-2 можно загружать лишь с приводов 0 и 1.

34. ХОТИТЕ ПОДДЕРЖКУ ДИСКОВ DY: или MY: ТОЛЬКО С ДВОЙНОЙ ПЛОТНОСТЬЮ (N)?

SYSGEN задает этот вопрос, если выбрана поддержка устройств DY:, MY:. Устройства DY:, MY: могут обслуживать диски как с обычной, так и с двойной плотностью. Следует ответить Y<ВК>, если не требуется обслуживания дисков обычной плотности.

35. УКАЖИТЕ АДРЕС РЕГИСТРА СОСТОЯНИЯ <N-го> КОНТРОЛЛЕРА <ИМУСТ> (NNNNNN)?

SYSGEN задает этот вопрос, если выбрана поддержка устройств: DD:, DX:, DY:. Когда этот вопрос появляется на терминале, то вместо <N-го> печатается слово «первого», а если выбрана поддержка второго контроллера, то вопрос появляется еще и вместо <N-го> печатается слово «второго». Допустимый диапазон для NNNNNN от 160000 до 177570.

36. УКАЖИТЕ АДРЕС ВЕКТОРА <N-го> КОНТРОЛЛЕРА <ИМУСТ> (NNN)?

Допустимый диапазон для NNN от 100 до 474.

41. НУЖЕН ФАЙЛОВО-ОРИЕНТИРОВАННЫЙ ДРАЙВЕР МАГНИТНОЙ ЛЕНТЫ (Y)?

SYSGEN задает этот вопрос, если выбрана поддержка магнитной ленты. Следует ответить Y<ВК> при необходимости иметь файлово-ориентированный драйвер магнитной ленты.

42. СКОЛЬКО ПРИВОДОВ МАГНИТНОЙ ЛЕНТЫ НУЖНО ОБСЛУЖИВАТЬ (2)?

Число приводов может быть от 1 до 4.

45. ИМЕЕТ ПОСТРОЧНО-ПЕЧАТАЮЩЕЕ УСТРОЙСТВО НЕСТАНДАРТНЫЕ АДРЕСА ВЕКТОРА ИЛИ РЕГИСТРА СОСТОЯНИЯ (N)?

В случае ответа Y<ВК> появляются два следующих вопроса.

46. УКАЖИТЕ АДРЕС РЕГИСТРА СОСТОЯНИЯ ПОСТРОЧНО-ПЕЧАТАЮЩЕГО УСТРОЙСТВА (177514)?

Адрес должен быть от 160000 до 177570.

47. УКАЖИТЕ АДРЕС ВЕКТОРА ПОСТРОЧНО-ПЕЧАТАЮЩЕГО УСТРОЙСТВА (200)?

Адрес должен быть от 100 до 474.

48. УКАЖИТЕ АДРЕС РЕГИСТРА СОСТОЯНИЯ ПОСТРОЧНО-ПЕЧАТАЮЩЕГО УСТРОЙСТВА ПОСЛЕДОВАТЕЛЬНОГО ТИПА (176500)?

SYSGEN задает этот и следующий вопросы, если выбрана поддержка устройства LS:. Адрес должен быть от 160000 до 177500.

49. УКАЖИТЕ АДРЕС ВЕКТОРА ПОСТРОЧНО-ПЕЧАТАЮЩЕГО УСТРОЙСТВА ПОСЛЕДОВАТЕЛЬНОГО ТИПА (300)?

Адрес должен быть от 100 до 474.

ХОТИТЕ ПОДДЕРЖИВАТЬ ДОПОЛНИТЕЛЬНЫЕ УСТРОЙСТВА (N)?

Если требуется обслуживать периферийные устройства, не поддерживаемые ФОДОС-2, и имеются драйверы этих устройств, то следует ответить Y<ВК>.

ВВЕДИТЕ ДВУХБУКВЕННОЕ ИМЯ ДОПОЛНИТЕЛЬНОГО УСТРОЙСТВА [DD:]

SYSGEN генерирует команды трансляции и редактирования драйвера дополнительного устройства и вносит имя устройства в таблицы монитора.

ВВЕДИТЕ СЛЕДУЮЩЕЕ ИМЯ ДОПОЛНИТЕЛЬНОГО УСТРОЙСТВА [DD:]

Если перечислены все дополнительные устройства, то следует напечатать <ВК>.

50. СКОЛЬКО СВОБОДНЫХ ПОЗИЦИЙ ОСТАВИТЬ В ТАБЛИЦАХ МОНИТОРА ДЛЯ ПЕРИФЕРИЙНЫХ УСТРОЙСТВ (0)?

В мониторе будет зарезервировано место только для тех драйверов, которые определены в процессе диалога. Если на последний вопрос дать ненулевой ответ, то в таблицах монитора будет зарезервировано место для новых драйверов, использование которых предполагается в дальнейшем.

55. СКОЛЬКО ЛОКАЛЬНЫХ ТЕРМИНАЛОВ, ВКЛЮЧАЯ СИСТЕМНЫЙ, НУЖНО ОБСЛУЖИВАТЬ (1)?

Так как системный терминал (консоль) всегда является локальным, то ответ не может быть нулевым. Число терминалов должно быть не более 8.

57. СКОЛЬКО УДАЛЕННЫХ ТЕРМИНАЛОВ НУЖНО ОБСЛУЖИВАТЬ (0)?

Число удаленных терминалов должно быть не более 7. Заметим, что общее число локальных и удаленных терминалов должно быть не более 8.

58. УКАЖИТЕ АДРЕС РЕГИСТРА СОСТОЯНИЯ ПЕРВОГО (СИСТЕМНОГО) ТЕРМИНАЛА (177560)?

Адрес должен быть от 160000 до 177500.

59. УКАЖИТЕ АДРЕС ВЕКТОРА ПЕРВОГО (СИСТЕМНОГО) ТЕРМИНАЛА (60)?

Адрес должен быть от 60 до 474.

Следующие вопросы SYSGEN зависят от ответов на вопросы 36 и 57. SYSGEN повторяет вопросы 58 и 59, спраши-

вая об адресах регистров состояний и векторов всех дополнительных локальных и удаленных терминалов.

Адресами регистров состояний и векторов всех дополнительных локальных терминалов по умолчанию являются следующие:

176500	300
176510	310
176520	320
176530	330
176540	340
176550	350
176560	360

Адресами регистров состояний и векторов всех удаленных терминалов по умолчанию являются следующие:

175610	310
175620	320
175630	330
175640	340
175650	350
175660	360
175670	370

УКАЖИТЕ ИМЯ И НОМЕР ПРИВОДА УСТРОЙСТВА ДЛЯ ДИСКА С ИСХОДНЫМИ ФАЙЛАМИ ДЛЯ ГЕНЕРАЦИИ (RK1)?

В дальнейшем этот диск будем называть исходным диском.

УКАЖИТЕ ИМЯ И НОМЕР ПРИВОДА УСТРОЙСТВА ДЛЯ ВЫХОДНЫХ (СГЕНЕРИРОВАННЫХ) ФАЙЛОВ (RK0)?

В дальнейшем этот диск будем называть вторым диском.

УКАЖИТЕ ИМЯ И НОМЕР ПРИВОДА УСТРОЙСТВА ДЛЯ ВЫВОДА КАРТЫ ЗАГРУЗКИ (RK0)?

Карту загрузки можно выводить на диск, на терминал. на построчно-печатающее устройство.

ХОТИТЕ СОХРАНИТЬ ОБЪЕКТНЫЕ ФАЙЛЫ (Y)?

Если на втором диске достаточно места для хранения объектных файлов и они нужны в целях дальнейшего изменения системы, то следует ответить Y<ВК>.

ХОТИТЕ СОХРАНИТЬ РАБОЧИЕ ФАЙЛЫ (Y)?

Если ответить Y<ВК>, то файлы SYSGEN.* (где * — BLD, MON, DEV, CND, TBL) будут удалены после завершения процесса генерации.

4.1.3. Трансляция и редактирование

После ответа на все вопросы SYSGEN следует скопировать файлы SYSGEN.CND и SYSGEN. TBL с системного тома

на диск, содержащий исходные файлы для генерации (исходный диск). Затем следует обратиться к одному из трех командных файлов (SYSGEN.MON, SYSGEN.DEV, SYSGEN.BLD), подав команду:

@SYSGEN. XXX <ВК>

где XXX — MON, DEV или BLD.

SYSGEN.MON — вызывает трансляцию и редактирование файлов, необходимых для генерации мониторов.

SYSGEN.DEV — вызывает трансляцию и редактирование файлов, необходимых для генерации драйверов.

SYSGEN.BLD — вызывает командные файлы SYSGEN.MON и SYSGEN.DEV.

Заметим, что все сгенерированные файлы будут иметь тип .SYG.

Для генерации операционной системы ФОДОС-2 требуется около 800 свободных блоков на исходном диске для размещения файлов типа .MAG и около 450 блоков на втором диске для размещения объектных файлов и файлов типа .SYG

Необходимо, чтобы нижеперечисленные файлы были на системном томе:

SWAP.SYS
Монитор
Необходимые драйверы
MACRO.SAV
LINK.SAV
SYSMAC.SML
SYSLIB.OBJ
SYSGEN.COM
PIP.SAV
DIR.SAV
DUP.SAV

Если генерация системы ФОДОС-2 проводится на носителях большой емкости (диски RK:, DP:), то системный диск может быть и исходным и вторым диском одновременно. Не следует проводить генерацию системы на устройстве DX: из-за малой емкости гибких дисков.

После того, как процесс трансляции и редактирования файлов мониторов и драйверов завершится, следует переименовать полученные файлы типа .SYG в тип .SYS, записать на полученный диск начальный загрузчик с вновь сгенерированного монитора, скопировать файл SWAP.SYS и загрузить новую систему.

5. СООБЩЕНИЯ СИСТЕМНОМУ ПРОГРАММИСТУ

В процессе диалога SYSGEN может печатать сообщения об ошибках. Эти сообщения печатаются по-русски и не требуют специального пояснения.

ПРИЛОЖЕНИЕ

Таблица 4

ПРОГРАММА НАЧАЛЬНОЙ ЗАГРУЗКИ МАГНИТНОЙ ЛЕНТЫ

Ячейка	Содержимое
1	2
1000	012700
1002	172524
1004	005310
1006	012740
1010	060011
1012	105710
1014	100376
1016	005710
1020	100767
1022	012710
1024	060003
1026	105710
1030	100376
1032	005710
1034	100777
1036	005007

ПЕРЕЧЕНЬ ССЫЛОЧНЫХ ДОКУМЕНТОВ

1. Операционная система ФОДОС-2
Контрольные задачи.
Руководство оператора

ПРОГРАММИРОВАНИЕ ПЕРИФЕРИЙНЫХ УСТРОЙСТВ РУКОВОДСТВО СИСТЕМНОГО ПРОГРАММИСТА

1. ОБЩИЕ СВЕДЕНИЯ О ПРОГРАММЕ

В данном документе будет рассмотрена структура стандартного драйвера и общее описание схемы обычного драйвера, подробно описаны дополнительные средства (внутренняя очередность, параметры команды SET, поддержка таймаута устройства ввода-вывода, специальные функции, регистрация ошибок и специальные услуги, имеющиеся в системе ХМ) для драйверов и их выполнение. Будет показано отличие системного драйвера от стандартного (это необходимо для записи начального загрузчика на системное устройство).

Перед написанием драйвера устройства необходимо:
— подробно изучить устройство;
— изучить структуру стандартного драйвера устройства;
— изучить схему драйвера устройства;
— подумать об использовании специальных средств;
— изучить примеры драйверов;
— подготовить блок-схему драйвера устройства;
— написать программу драйвера;
— установить, проверить и отладить драйвер.

2. СТРУКТУРА ПРОГРАММЫ

Драйвер устройства системы ФОДОС-2 состоит из шести следующих секций:
— определения;
— заголовок;
— инициирование ввода-вывода;
— обработка прерываний;

— завершение ввода-вывода;
— окончание драйвера.

Каждая секция — это отдельный логический блок, содержащий код для определенной цели. Так как макробиблиотека системы ФОДОС-2 предусматривает специальные запросы для генерации требуемого кода для каждой части, то написание их несложно, необходимо лишь внимательнее изучить примеры драйверов устройств.

2.1. Секция определений

Исходный файл драйвера устройства начинается с секции определений, которая включает директиву .MCALL для запроса .DRDEF и других запросов. Большую часть работ в секции определений осуществляет запрос .DRDEF.

2.1.1. Запрос .DRDEF. Запрос .DRDEF используется в начале драйвера устройства и выполняет следующие функции:
— выдает директиву .MCALL для всех относящихся к драйверу запросов;
— обеспечивает значения по умолчанию для условных обозначений системы;
— вызывает запрос .QELDF для определения смещения элементов очереди;
— определяет комбинации разрядов состояния устройства;
— определяет размер устройства в блоках (DDDSIZ);
— определяет идентификацию устройства (DD$COD);
— устанавливает слово состояния устройства, используя DDDSIZ и DD$COD;
— обеспечивает значения по умолчанию для регистра состояния устройства (DD$CSR) и вектора прерывания (DD$VEC);
— делает символы DD$CSR и DD$VEC глобальными.
Здесь и далее DD — двухсимвольное имя устройства.

Формат запроса .DRDEF следующий:

.DRDEF NAME, CODE, STAT, SIZE, CSR, VEC

где NAME — двухсимвольное имя устройства (например, MT для магнитной ленты);
CODE — восьмеричное число, определяющее устройство (см. табл. 1);
STAT — комбинация разрядов состояния устройства (см. табл. 2);
SIZE — размер устройства в блоках (по 256 слов); если устройство нефайловой структуры, то SIZE=0;
CSR — адрес регистра состояния устройства по умолчанию;
VEC — адрес вектора прерывания устройства по умолчанию.

Запрос .DRDEF выдает директиву .MCALL для следующих запросов:

.DRAST	.DRBEG	.DRFIN
.DRBOT	.DREND	.DRSET
.DRVTB	.FORK	.QELDF

Кроме того, если в драйвере при генерации системы выбрана поддержка таймера (TIM$IT=1), то .DRDEF выдает директиву .MCALL для запросов .TIMIO и .CTIMIO.

2.1.1.1. Условия генерации системы

В системе ФОДОС-2 широко используются условные директивы при транслировании. Секции программы на исходном языке участвуют в транслировании или нет, в зависимости от значения условных символов. Например, ФОДОС-2 использует условный символ ERL$G для показа — должна ли транслироваться подпрограмма регистрации ошибок или нет. Если в драйвере устройства используются условные символы, то они должны согласовываться со стандартными, использующимися в ФОДОС-2. При равенстве нулю условного символа данная особенность не включается при транслировании программы, в случае, когда условный символ равен 1, эта особенность присутствует; ФОДОС-2 использует только значения 0 и 1; если условные символы имеют значения иные, чем 0, то .DRDEF приравнивает их 1.

Запрос .DRDEF приравнивает нулю, при генерации системы, TIM$IT (для тайм-аута устройства), MMG$T (для поддержки расширенной памяти) и ERL$G (для регистрации ошибок), если не определить их в начале файла при транслировании.

2.1.1.2. Смещение элементов очереди

Запрос .DRDEF вызывает запрос .QELDF для символического определения смещений элементов очереди. Ниже приведены сгенерированные смещения элементов очереди:

Q.LINK=0	(связь со следующим элементом очереди)
Q.CSW=2.	(указатель для слова состояния канала)
Q.BLKN=4.	(номер физического блока)
Q.FUNK=6.	(код специальной функции)
Q.JNUM=7.	(номер задания)
Q.UNIT=7.	(номер привода устройства)
Q.BUFF=^O10	(адрес буфера пользователя)
Q.WCNT=^O12	(счетчик слов)
Q.COMP=^O14	(код подпрограммы завершения)
Q.ELGH=^O16	(размер элемента очереди)

Так как драйвер обычно рассматривает только смещения элементов очереди относительно Q.BLKN, запрос .QELDF определяет также следующие символические смещения:

```
Q¤LINK=-4
Q¤CSW=-2
Q¤BLKN=0
Q¤FUNC=2
Q¤JNUM=3
Q¤UNIT=3
Q¤BUFF=4
Q¤WCNT=6
Q¤COMP=10
```

2.1.1.3. Определение символов

Для определения символов используются операторы прямого присваивания. Обычно таким образом определяются регистры устройств и другие необходимые внутренние символы. Ниже приведены примеры из драйверов устройств системы ФОДОС-2.

Например, для определения внутреннего символа для перевода строки (код 12 в КОИ-7):

```
LF=12                          ; перевод строки
```

Регистры устройства определяются следующим образом:

```
RKDS=RK¤CSR                    ; регистр состояния привода
RKER=RKDS+2                    ; регистр ошибки
RKCS=RKDS+4                    ; регистр состояния
RKWC=RKDS+6                    ; регистр счетчика слов
```

Запрос .DRDEF определяет для пользователя следующие символы:

```
HDERR¤=1        ; разряд неисправимой ошибки в CSW
EOF=20000       ; разряд конца файла в CSW
```

2.1.2. Байт идентификации устройства

Младший байт слова состояния устройства, байт идентификации устройства, определяет каждое устройство в системе. Чтобы указать конкретное устройство, необходимо указать определенный код в аргументе CODE запроса .DRDEF. Восьмеричные значения кода приведены в табл. 1.

Для создания кода идентификации устройств, которые не поддерживаются системой ФОДОС-2, можно пользоваться значением кода 374 для первого непредусмотренного устройства, 373 — для второго и т. д. Главное, чтобы эти коды не противоречили кодам, которые будут использоваться в будущем.

Таблица 1

Имя	Код	Устройство
RK	0	Кассетный магнитный диск
	1	зарезервирован
EL	2	Логическое устройство регистрации ошибок
LP	3	Построчно-печатающее устройство
TT, BA	4	Системный терминал типа 15ИЭ-00-013 или драйвер управления пакетной обработкой
	5	зарезервирован
DY	6	Гибкий диск с двойной плотностью записи (диаметр 203 мм)
PC	7	Перфоленточное устройство ввода (FS 1501) и вывода (ПЛ—150)
	10	зарезервирован
MT	11	Магнитная лента типа СМ 5300.01
	12—20	зарезервированы
DP	21	Пакет магнитных дисков (29 Мбайт)
DX	22	Гибкий диск с одинарной плотностью записи (диаметр 203 мм)
	23—24	зарезервированы
NL	25	Фиктивное устройство
	26—33	зарезервированы
DD	34	Магнитная лента кассетного типа
	35—40	зарезервированы
LS	41	Построчно-печатающее устройство последовательного типа
MQ	42	Драйвер обмена между заданиями
	43—45	зарезервированы
LD	46	Драйвер логического диска
VM	47	Драйвер расширенной памяти
DU	50	Диски винчестерского типа и гибкие мини-диски
DW	53	Винчестерский диск
SL	51	Редактор командной строки
MX	376	Гибкий диск с одинарной плотностью записи, двухсторонний (диаметр 133 мм)
MY	375	Гибкий диск с двойной плотностью записи, двухсторонний (диаметр 133 мм)

2.1.3. Слово состояния устройства

Слово состояния устройства определяет каждое физическое устройство в системе ФОДОС-2 и содержит информацию о том, какое это устройство: произвольного или последовательного доступа к данным. Слово состояния устройства записывается в нулевой блок файла драйвера устройства и в таблицу $STAT, когда устройство установлено; программный запрос .DSTATUS возвращает значение слова состояния устройства в выполняющуюся программу. Запрос .DRDEF устанавливает слово состояния устройства на основании аргументов CODE и STAT.

В табл. 2 показаны значения разрядов в слове состояния устройства. Запрос .DRDEF использует символ DDSTS для определения слова состояния устройства.

Таблица 2

Разряд	Символ	Значение
1	2	3
0—7	—	Байт идентификации устройства
8	VARSZ¤	0 = запрос .SPFUN 373 недопустим для этого драйвера 1 = драйвер допускает использование запроса .SPFUN 373 (возвращает размер тома)
9	ABTIO¤	0 = вход в драйвер осуществляется не в точке преждевременного прерывания по нормальному выходу из программы 1 = вход в драйвер осуществляется в точке преждевременного прерывания всякий раз по окончании программы
10	SPFUN¤	0 = запрос .SPFUN недопустим 1 = драйвер допускает использование .SPFUN
11	HNDLR¤	0 = вводит драйвер в точку входа преждевременного прерывания, если в прерванном задании есть активный элемент очереди 1 = вводит драйвер в точку входа преждевременного прерывания во всех случаях преждевременных прерываний. Этот разряд игнорируется в системе SJ
12	SPECL¤	1 = устройство специальной справочной структуры (например, MT)
13	WONLY¤	1 = устройство только для записи
14	RONLY¤	1 = устройство только для чтения
15	FILST¤	0 = устройство последовательного доступа к данным (например, MT) 1 = устройство произвольного доступа к данным (например, RK, DX, MX)

Разряд 11 в слове состояния устройства устанавливается для драйверов устройств, которые перемещают элемент очереди при вводе и организуют внутреннюю очередь, и для устройств, таких, как магнитная лента, которые имеют внутренние данные, которые могут измениться при преждевременном прерывании. Предполагается, что все драйверы устройств, у которых установлен разряд 15, являются устройствами файловой структуры в системе ФОДОС-2 для большинства программ, обслуживающих систему. Самым простым способом определения слова состояния устройства является ис-

пользование мнемоники для комбинации двоичных разрядов, которые определяют запрос .DRDEF. Таким образом, можно создать аргумент STAT посредством объединения (с помощью операции «ИЛИ») соответствующих символов из вышеприведенного перечня:

```
FILST¤  ==100000 ;файловая структура произвольного
                 ;доступа к данным
RONLY¤  ==40000  ;только чтение
WONLY¤  ==20000  ;только запись
SPECL¤  ==10000  ;несправочная структура
HNDLR¤  ==4000   ;ввод драйвера при преждевременном
                 ;прерывании
SPFUN¤  ==2000   ;использование специальных функций
ABTIO¤  ==1000   ;всегда выполняет преждевременное
                 ;прекращение входа
VARSZ¤  ==400    ;поддержка драйвером томов
                 ;переменного размера
```

Пример:

```
Для RK:   FILST¤
Для MT:   SPECL¤! SPFUN¤
Для LP:   WONLY¤
```

В этом примере приведен аргумент STAT для драйверов устройств RK, MT, LP.

2.1.4. Слово размера устройства

Аргумент SIZE в запросе .DRDEF определяет размер устройства в блоках. Запрос .DRDEF помещает значение размера устройства в DDDSIZ. Если устройство не является устройством произвольного доступа к данным, необходимо поместить значение 0 в SIZE. Например, размер устройства DX: равен 486 блокам (746 восьмеричных); размер устройства PC: равен 0, т. к. оно не является устройством произвольного доступа к данным.

Программный запрос .DSTATUS возвращает значение слова размера устройства в выполняющуюся программу.

2.2. Секция заголовка

В заголовке драйвера вызывается запрос .DRBEG для установки первых пяти слов драйвера. Кроме того, этот запрос записывает 5 слов информации в блок 0 файла драйвера, в ячейки с 52 по 60, и создает несколько глобальных символов. Данные, которые устанавливаются в секции заголовка, используются при загрузке драйвера в память посредством программного запроса .FETCH или команды монитора LOAD. Содержимое ячейки 176 используется начальным загрузчиком

для проверки наличия аппаратуры устройства во время установки драйвера.

2.2.1. Информация в блоке 0

В табл. 3 приведены 5 слов в блоке 0, которые запрос .DRBEG устанавливает посредством директивы .ASECT, и три слова, которые .DRBOT устанавливает для загружаемых устройств. В таблице соответствующая мнемоника показана в квадратных скобках, двухсимвольное имя устройства представлено посредством DD.

Код сравнения установки, являющийся параметром, описан в п. 6.2.3.

Таблица 3

Ячейка	Содержимое (и мнемоника)
1	2
52	Размер драйвера в байтах [DDEND—DDSTRT]
54	Размер устройства в блоках [DDDSIZ]
56	Слово состояния устройства [DDSTS]
60	Слово состояния для отражения текущих дополнительных средств генерации системы [ERL$G+<MMG $T*2>+<TIM$IT*4>]
62	Указатель начала первичного драйвера (из .DRBOT)
64	Размер первичного драйвера в байтах (из .DRBOT)
66	Смещение от начала первичного драйвера к началу подпрограммы считывания начального загрузчика (из .DRBOT)
176	Адрес регистра команд и состояния CSR [DD$CSR]
200	Начало кода сравнения установки

2.2.2. Первые пять слов драйвера

В табл. 4 приведены 5 слов, которые запрос .DRBEG генерирует в начале секции определений драйвера.

Таблица 4

Слово	Символ	Устройство
1	2	3
1	DDSTRT::	Вектор устройства (для одновекторных устройств); смещение к таблице векторов (для многовекторных устройств)
2	——	Смещение к точке входа обработки прерывания
3	——	Приоритет (340)
4	DDLQE::	Указатель последнего элемента очереди
5	DDCQE::	Указатель текущего элемента очереди

2.2.3. Запрос .DRBEG используется для установки информации в блоке 0 и первых пяти слов драйвера. Этот запрос генерирует также соответствующие глобальные символы данного драйвера. Перед использованием .DRBEG необходимо с помощью запроса .DRDEF определить DD$CSR, DD$VEC, DDDSIZ и DDSTS. Формат для .DRBEG следующий: .DRBEG NAME, где NAME — двухсимвольное имя устройства.

2.2.4. Многовекторные драйверы: запрос .DRVTB

Драйверы устройств системы ФОДОС-2 могут обслуживать устройства, имеющие более одного вектора. Например, драйвер PC обрабатывает прерывание посредством вектора 70 для считывателя с перфоленты и посредством вектора 74 для перфоратора. Если данное устройство имеет один вектор прерывания, то его драйвер должен иметь таблицу трехсловных элементов для каждого вектора. Элемент для каждого вектора состоит из ячейки вектора, точки входа по прерыванию, значения слова состояния процессора (или PS). Для установки заголовка драйвера необходимо вызвать запрос .DRVTB два или большее число раз. Запрос .DRVTB устанавливает таблицу трехсловных элементов для каждого вектора многовекторного устройства. Ее необходимо поместить в драйвере между запросами .DRBEG и .DREND (.DRBOT) до подпрограммы обработки прерываний. Запрос .DRVTB необходимо вызывать один раз для каждого вектора, программные запросы должны появляться в драйвере один за другим. Формат запроса следующий: .DRVTB NAME, VEC,INT[,PS]

где NAME — двухсимвольное имя устройства; его необходимо указать в первом вызове .DRVTB; во всех последующих вызовах его можно опустить;

VEC — ячейка вектора; она должна находиться между 0 и 474; первым вектором обычно является DD$VEC, значение которого должно быть кратно четырем;

INT — символическое имя подпрограммы обработки прерываний; оно должно быть определено в драйвере и обычно принимает форму DDINT;

PS — произвольное значение, которое можно использовать для обозначения четырех младших разрядов нового слова состояния процессора в векторе прерывания; значение по умолчанию равно нулю.

Пример:

```
; Таблица  векторов перфоленточного устройства ввода-вывода
.IF PR114$                          ; если перфоленточное
                                    ; устройство ввода-вывода
    .DRVTB PC,PC$VEC,PCINT          ; таблица ввода с перфо-
                                    ; ленты
    .DRVTB ,PP$VEC,PPINT            ; таблица вывода на пер-
                                    ; фоленту
.ENDC
```

В этом примере показаны строки и коды, которые генерирует запрос .DRVTB.

Пример:

```
.WORD <PC$VEC>&<^C3>,PCINT-.,340!0 ; таблица
                                   ; ввода
                                   ; с перфо-
                                   ; ленты
.WORD <PP$VEC>&<^C3>,PPINT-.,340!0 ; таблица
                                   ; вывода на
                                   ; перфоленту
.WORD 0                            ; для окон-
                                   ; чания таб-
                                   ; лицы
```

В этом примере приведена таблица вектора, сгенерированная запросом .DRVTB. Из примера видно, что разряды приоритета PS всегда установлены семеркой, даже если аргумент PS опущен.

2.2.5. Коды условий PS

В запросе .DRVTB существенны только разряды кодов условий аргумента PS. Они могут быть полезны, если имеется общая точка входа в подпрограмму обработки прерывания для двух или большего числа векторов и необходимо определить, через какой вектор возникло прерывание. Например, драйвер PC имеет раздельные точки входа прерывания для двух его векторов, поэтому он может легко определить источник прерывания. Прерывания посредством вектора 70 идут к подпрограмме PCINT:; прерывания посредством вектора 74 идут к PPINT:.

Если бы драйвер PC имел только одну точку входа прерывания, названную PCINT:, то в этом случае, драйвер мог определить, какой вектор использовал прерывание посредством установки кодов условий в PS для векторов. Для вектора, считывающего устройства 70 он мог оставить C-разряд чистым, а для вектора 74 он мог установить C-разряд. Затем

PCINT: управление может переходить к различным подпрограммам, основанным на значении С-разряда в новом PS.

Пример:

```
; Таблица векторов перфоленточного устройства ввода-вывода
.IF EQ PR11$X                          ; если перфоленточное
                                       ; устройство ввода-
                                       ; вывода
        .DRVTB PC,PR$VEC,PCINT         ; С-разряд чист
        .DRVTB ,PP$VEC,PCINT,1         ; С-разряд установлен
.ENDC
```

В этом примере показано как вызывается запрос .DRVTB, и определяется значение кодов условий в PS.

2.3. Секция инициирования ввода-вывода содержит пять выполняемых команд драйвера. Цель этой секции — начать передачу данных. Необходимо написать позиционно-независимый код (PIC) для драйвера.

Когда выдается программный запрос .READ или .WRITE, требующий устройства ввода-вывода, управление сначала передается клавиатурному монитору. Затем клавиатурный монитор вызывает драйвер первого слова после пятисловного заголовка самого драйвера. Он делает вызов каждый раз, когда новый элемент очереди становится первым элементом очереди драйвера. Такая ситуация возникает, когда элемент добавляется в пустую очередь драйвера или когда элемент становится первым в очереди, потому что предыдущий элемент был освобожден. Если любой из параметров в запросе ввода-вывода недопустим для устройства (например, слишком большой номер блока, слишком высокий номер привода и т. д.), драйвер сразу же должен завершить секцию ввода-вывода и сигнализировать о невосстановимой (фатальной) ошибке.

Код инициирования ввода-вывода выполняется при нулевом приоритете процессора в системном режиме. Это означает, что не может произойти переключение контекста, невозможно вызвать подпрограмму завершения, и прерывание по вектору 4 и 10 вызывает фатальный останов системы. В этой секции можно использовать все регистры. Пятое слово заголовка драйвера, DDCQE, содержит указатель текущего элемента очереди в его третьем слове, Q.BLKN.

Организованная в очередь система ввода-вывода гарантирует, что запросы передачи данных преобразуются в последовательную форму таким образом, что драйверам устройств системы ФОДОС-2 нет необходимости вводиться повторно.

Поэтому, можно уменьшить размер драйвера посредством объединения, а не разделения чистого кода и сегментов данных.

2.3.1. Указания для начала передачи данных

Так как целью секции инициирования ввода-вывода является начало передачи данных, то для этого необходимо указать соответствующие команды. Следующие этапы представляют руководство для обобщенной секции инициирования ввода-вывода.

1. Необходимо решить, сколько раз драйвер будет повторно выполнять передачу данных в случае, если произойдет ошибка. Необходимо инициализировать счетчик повторных передач посредством перемещения в него максимального числа повторных передач. Следующие две строки иллюстрируют этот этап:

```
MOV    #RKCNT,(PC)+   ; RKCNT=максимум #
                      ; повторных передач
RETRY: .WORD    0     ; счетчик повторных передач
```

2. Указатель текущего элемента очереди необходимо поместить в регистр и получить число приводов для устройства и число блоков для передачи из элемента очереди. Это иллюстрируют следующие строки кода:

```
MOV   RKCQE,R5          ; засылает указатель текущего
                        ; элемента очереди
MOV   @R5,R2            ; R2 = номер блока
MOV   Q$UNIT-1(R5),R4   ; R4 = номер запрашиваемого
                        ; привода
ASR   R4                ; сдвигает номер привода на
ASR   R4                ; три разряда вправо
ASR   R4                ; в младший байт
SWAB  R4                ; размещает номер привода в
                        ; трех разрядах старшего
                        ; байта
BIS   #^C<DAUNIT>,R4    ; изолирует привод в разрядах
                        ; выборки привода
```

3. Затем вычисляется адрес для начала передачи данных в устройство. Используемые при этом команды зависят от структуры устройства. После того, как адрес вычислен, его необходимо записать в ячейку памяти. Если будет необходимо повторно выполнить передачу, то адрес не придется вычислять повторно.

```
        .
        .
        .
        MOV     R3,(PC)+; сохраняется адрес в DISKAD
DISKAD: .WORD   0       ; ячейка памяти для сохранения
                        ; вычисленного адреса
```

4. Описанные выше этапы 1—3 выполняются только один раз для каждого запроса ввода-вывода данных из выполняющейся программы. Однако в случае восстановимой ошибки можно начать повторно передачу как часть операции повторения. Поэтому, если поместить метку, чтобы использовать ее как точку входа повторной передачи, то можно избежать повторения этапов 1—3. Следующие этапы могут быть выполнены более одного раза: они выполняются один раз для первого пуска ввода-вывода, и они могут быть выполнены вновь, если ошибка ввода-вывода вызывает повторную передачу. В этой точке драйвер должен определить, является ли запрос ввода-вывода считыванием, записью или установкой. Затем он должен генерировать соответствующий код операции и засылать его в регистр состояния устройства. Этот этап фактически инициирует передачу ввода-вывода.

```
        CSIE    =100                    ; разрешение прерывания
        FNWRITE =12                     ; запись
        CSGO    =1                      ; разряд GO
        .
        .
        .
AGAIN:  MOV     RKCQE,R5                ; указатель элемента
                                        ; очереди
        MOV     #CSIE!FNWRITE!CSGO,R3   ; устанавливает запись
        MOV     #RKDA,R4                ; указатель регистра
                                        ; адреса диска
        .
        .
        .
```

5. И последнее, возврат к прерванной программе необходимо осуществлять через монитор. Затем, когда передача ввода-вывода заканчивается, устройство будет прервано, и управление перейдет к драйверу в точке входа прерывания в секции обработки прерывания драйвера.

```
        RTS     PC      ; ожидание прерывания
```

2.4. Секция обработки прерываний

Управление передается секции обработки прерываний драйвера, когда устройство прерывается или когда программа, запрашивающая передачу ввода-вывода, преждевременно прерывается. Код в этой секции должен сначала определить,

имела ли передача данных ошибку, была ли передача данных завершена или нет, а затем предпринять соответствующее действие.

Первым шагом в кодировании секции обработки прерываний является установка точки входа преждевременного прерывания посредством использования запроса .DRAST (эти точки входа иногда рассматриваются как точки входа асинхронного прерывания). Именем точки входа по умолчанию является DDINT. Обычно драйвер устройства вызывается у точки входа прерывания, когда прерывание возникло. Однако, при некоторых обстоятельствах драйвер вызывается у точки входа преждевременного прерывания.

2.4.1. Точка входа преждевременного прерывания

Существует ряд ситуаций, которые вызывают преждевременное прерывание в организованной в очередь системе ввода-вывода:

— двойное нажатие СУ/С может преждевременно прервать выполняющуюся программу;
— программный запрос .HRESET вызывает преждевременное прерывание;
— прерывание по 4 или 10 вектору или любое другое условие, которое приводит к следующему типу невосстановимой (фатальной) ошибки ?MON—F—.

Преждевременное прерывание, независимо от того, введен драйвер или нет, зависит от двух факторов. Драйвер всегда вводится в точке входа преждевременного прерывания (слово сразу же перед обычной точкой входа прерывания), если выполнялся активный элемент очереди, который принадлежит преждевременно прерванному заданию. В мониторах FB и XM драйвер вводится всегда независимо от наличия элемента очереди, если HNDLR$ (разряд 11) установлен в слове состояния устройства. Если HNDLR$ установлен, прекращение выполнения программы рассматривается в двух случаях: (1) — преждевременное прерывание во время операции ввода-вывода; (2) — преждевременное прерывание во время выполнения другой операции.

Монитор SJ игнорирует этот разряд. К тому же, драйвер не может выйти, когда прекращается работа в системе SJ, монитор SJ автоматически выполняет запрос .RESET.

Во всех условиях, при входе в драйвер, регистр R4 всегда содержит номер преждевременно прерванного задания. Регистры R0—R3 должны быть сохранены и восстановлены. Когда возникает преждевременное прерывание, в некоторых устройствах важно остановить ввод-вывод. Символьно-ориен-

тированные устройства, как, например, PC:, попадают под эту категорию. При преждевременном прерывании драйвер должен остановить устройство, чтобы, например, предотвратить выход из-под контроля (отклонение) ленты. Он также должен убедиться, что устройство не может прерваться вновь. Поэтому символьно-ориентированные устройства обычно содержат подпрограмму преждевременных прерываний; точка входа преждевременного прерывания — это просто команда перехода к этой подпрограмме. Например, драйвер PC имеет подпрограмму преждевременного прерывания, которая запрещает прерывание в перфоленточном устройстве ввода-вывода. Затем драйвер вводится в монитор в секцию завершения ввода-вывода. Следующие строки взяты из драйвера PC:

```
PCDONE:  CLR    @#PC$CSR   ; отменяет прерывание
                           ; перфоленточного
                           ; устройства ввода
         CLR    @#PP$CSR   ; отменяет прерывание
                           ; перфоленточного
                           ; устройства вывода
```

Другим устройствам, как, например, диски, разрешается попытка завершения передачи ввода-вывода, даже если возникает преждевременное прерывание. Фактически, попытка преждевременного прерывания в середине операции может вызвать разрушение данных или форматирование информации на диске. Поэтому, вместо того, чтобы иметь отдельную подпрограмму преждевременных прерываний, большинство драйверов дисков игнорируют преждевременные прерывания. Таким образом, команда RTS PC размещается у точки входа преждевременного прерывания, которая просто возвращает управление монитору.

Если в драйвере используется запрос .FORK, то существует специальная процедура, которая имеет место, если возникает преждевременное прерывание. Необходимо поместить 0 в F.BADR (адрес подпрограммы .FORK, смещенный на 2) в FORK-блоке. Это предотвращает монитор от попытки выполнения бессмысленной после преждевременного прерывания подпрограммы .FORK.

2.4.2. Понижение приоритета до приоритета устройства

Когда возникает прерывание, драйвер вводится с приоритетом 7. Так же, как и в подпрограмме обработки прерываний, первой задачей драйвера является понижение приоритета процессора до приоритета устройства, позволяя таким образом устройствам с большим приоритетом прервать эту подпрограмму обработки прерываний. Вместо использования

вызова .INTEN, как и в подпрограмме обработки прерываний, для понижения приоритета используется запрос .DRAST.

2.4.3. Запрос .DRAST используется для установки точки входа прерывания, точки входа преждевременного прерывания и для снижения приоритета процессора. Этот запрос устанавливает также глобальный символ ¤INPTR, который содержит указатель подпрограммы ¤INTEN в резидентном мониторе. Этот указатель заполняется начальным загрузчиком (для системного устройства) или во время запроса .FETCH (для устройства данных).

Формат запроса .DRAST следующий: .DRAST NAME, PRI[,ABO]

где NAME — двухсимвольное имя устройства;

PRI — приоритет устройства, т. е. приоритет, при котором должна выполняться подпрограмма обработки прерываний;

ABO — произвольный аргумент, который представляет собой метку точки входа преждевременного прерывания; если этот аргумент отсутствует, запрос генерирует команду RTS PC у точки входа преждевременного прерывания, которая является словом, предшествующим точке входа прерывания.

Пример:

```
        .DRAST  PP,4,PCDONE
        .GLOBL  ¤INPTR                  ; делает этот символ
                                        ; глобальным
        BR      PCDONE                  ; точка входа
                                        ; преждевременного
                                        ; прерывания
PPINT:: JSR     R5,@¤INPTR              ; переход в монитор
                                        ; по коду .INTEN
        .WORD   ^C<4*^O40>&^O340        ;новый приоритет
```

В этом примере показаны запрос .DRAST из драйвера PC и код, который он генерирует.

Пример:

```
        .DRAST  RK,5
        .GLOBL  ¤INPTR          ; делает этот символ
                                ; глобальным
        RTS     PC              ; осуществляет возврат
                                ; из преждевременного
                                ; прерывания
RKINT:: JSR     R5,@¤INPTR      ; переход в монитор
                                ; по коду .INTEN
        .WORD   ^C<5*^O40>&^O340 ;новый приоритет
```

В этом примере приведена часть драйвера RK, который не имеет подпрограммы преждевременного прерывания.

2.4.4. Основные правила для программирования секции обработки прерываний

Так как целью этой секции является вычисление результатов последнего действия устройства, то нет необходимости составлять для этого команды. В основном, код должен определять, была ли передача ошибочной, была ли она закончена или нет.

1. Если возникла ошибка

Если во время передачи возникла ошибка, драйвер должен определить, была ли эта ошибка восстановимой или невосстановимой, после того, как операция повторится.

Если ошибка невосстановимая, драйвер должен немедленно выводиться через секцию завершения ввода-вывода.

Если ошибка восстановимая, драйвер должен подготовиться к повторной передаче. Он должен уменьшить счетчик допустимых повторных операций. Затем, на FORK-уровне, он должен снова перейти к секции инициирования ввода-вывода для возобновления передачи. Если число повторных передач достигло допустимого (счетчик повторений равен нулю), то необходимо рассматривать эту неисправность как невосстановимую ошибку. В этом случае драйвер должен переходить в секцию завершения ввода-вывода.

Снижение до FORK-уровня необязательно для обработки ошибки. Использовать запрос .FORK или нет, зависит от промежутка времени, необходимого для установки повторной передачи. Вызов .FORK полезен тем, что позволяет использовать регистры R0—R3, давая возможность использовать общие подпрограммы для повторных передач. Если запрос .FORK не используется, то для использования допустимы только регистры R4 и R5.

2. Выполнение повторных передач на FORK-уровне

Запрос .FORK вызывает возврат к резидентному монитору, который отвергает текущее прерывание. Код, следующий за .FORK, выполняется при нулевом приоритете, а не при приоритете устройства, после того, как все другие прерывания были обработаны, но до того, как любое задание или его подпрограммы завершения смогут выполниться. Код, следующий за .FORK, выполняется так же, как и основная часть секции обработки прерываний драйвера, в системном режиме (это тот же режим, в котором выполняется секция инициирования ввода-вывода). Таким образом, переключение контекста защищено во время выполнения кода на FORK-уровне, и лю-

бое прерывание по 4 или 10 вектору является причиной фатального останова системы.

Пример:

```
        .FORK   RKFBLK      ; вызов .FORK
        JSR     R5,@ $FKPTR ; переход в монитор по
                            ; коду .FORK
        .WORD   RKFBLK—.    ; смещение к элементу
                            ; FORK-очереди
RKRETR: CLRB    RETRY+1     ; переустановка признака
        BR      AGAIN       ; переход в секцию
                            ; инициирования
                            ; ввода-вывода
```

В этом примере приведена часть драйвера RK, где драйвер понижает приоритет до FORK-уровня для выполнения повторных передач данных после возникновения восстановимой ошибки. FORK-уровень идеален для выполнения повторных передач, так как это может быть длительный процесс. Здесь же показан вызов .FORK и его расширение.

3. Если передача не была завершена

Передача считается незавершенной, если много символов или много блоков данных остались непереданными. Драйвер должен перезапустить устройство и выйти по команде RTS PC в ожидание следующего прерывания.

4. Если передача была завершена

Когда передача завершена, драйвер может просто выйти через секцию завершения ввода-вывода.

2.5. Секция завершения ввода-вывода обеспечивает общий путь выхода, чтобы информировать монитор о том, что драйвер выполнил текущий запрос, и, поэтому, монитор может освободить текущий элемент очереди. Хотя другие секции драйвера являются различными отдельными частями, секция завершения ввода-вывода — это фактически расширение секции обработки прерываний, и граница между этими двумя секциями — искусственная. Управление не переходит к секции завершения ввода-вывода в результате вызова монитора, вызова подпрограммы или перехода, а переходит лишь в результате обычного хода выполнения через секцию обработки прерывания. Выполнение передается в секцию завершения ввода-вывода, когда обнаружена невосстановимая ошибка, когда число повторных передач в случае восстановимой ошибки достигло допустимого или когда передача данных закончена. (Если сразу же будет обнаружена невосстановимая ошибка, то можно непосредственно переходить из этой секции в секцию инициирования ввода-вывода.)

1. Если произошла ошибка

Существует два вида ошибок, вызывающих передачу управления в секцию завершения ввода-вывода: невосстановимые ошибки, которые вызывают немедленный переход в эту секцию, и восстановимые ошибки, которые исчерпали допустимое число повторных передач и которые переходят в эту секцию после последней неудачной попытки повторной передачи. При передаче управления монитору можно считать оба эти случая аналогичными.

Сначала необходимо установить разряд невосстановимой ошибки для канала, разряд 0 в слове состояния канала (CSW). Второе слово элемента очереди ввода-вывода, Q.CSW, указывает слово состояния канала. Затем необходимо переходить к подпрограмме завершения ввода-вывода в резидентном мониторе. Для генерации кода этого перехода используется запрос .DRFIN.

Пример:

```
BIS     #HDERR¤,@—(R5)      ;устанавливается разряд
                            ;невосстановимой
                            ;ошибки
                            ;(R5 указывается третье
                            ;слово элемента очереди;
                            ;указателем CSW
                            ;является второе слово)
.DRFIN  RK                  ;переход в монитор
```

В этом примере приведены строки кода из драйвера RK. Они показывают, как драйвер устанавливает разряд невосстановимой ошибки и осуществляет возврат в монитор.

2. Если передача была завершена

Для блочно-ориентированных устройств, как, например, диск, драйвер просто запрещает прерывание и выполняет переход в монитор. Запрос .DRFIN генерирует код для выполнения перехода.

Для символьно-ориентированных (или словно-ориентированных) устройств, как, например, перфоленточное устройство ввода-вывода, эта процедура сложнее, потому что драйвер должен сообщить об окончании файла заданию, которое запросило передачу ввода-вывода. Примерами условий, которые вызывают конец файла, являются: отсутствие перфоленты в перфоленточном устройстве ввода и обнаружение CУ/Z, напечатанного на системном терминале. Когда драйвер фактически обнаруживает условие EOF в операции READ, он должен установить внутренний признак EOF, поместить последний символ в буфер пользователя, а затем заполнить нулями

остальной буфер. Затем драйвер должен переходить обратно в монитор, как если бы EOF не был обнаружен, и буфер просто бы заполнился. Драйвер ожидает, пока не будет вызван вновь признак EOF пользователю.

Драйвер PC использует разряд готовности в слове состояния перфоленточного устройства ввода-вывода как внутренний признак EOF.

Пример:

```
1¤:     CLRB    @—(R4)          ;очищает байт (R4 ука-
                                ;зывает адрес буфера)
        INC     (R4)+           ;изменяет адрес буфера
        DEC     @R4             ;уменьшает счетчик ос-
                                ;тавшихся байтов
        BNE     1¤              ;цикл до готовности
PCDONE: CLR     @#PC¤CSR        ;запрещает прерывание
                                ;перфоленточного уст-
                                ;ройства ввода
        CLR     @#PP¤CSR        ;запрещает прерывание
                                ;перфоленточного уст-
                                ;ройства вывода
        CLR     PCFBLK+2        ;очищает FORK-блок для
                                ;недопущения диспетче-
                                ;ризации
PCFIN:  .DRFIN  PC              ;переход к окончанию
                                ;ввода-вывода
```

В этом примере показано, как драйвер PC заполняет нулями буфер пользователя, когда он обнаруживает конец файла, устанавливает внутренний признак EOF и переходит обратно в монитор.

Когда драйвер вызывается вновь с новым элементом очереди для другой операции READ, он сначала проверяет внутренний признак EOF. Найдя его установленным, драйвер устанавливает разряд EOF слова состояния канала, разряд 13, и переходит обратно в монитор. Резидентный монитор, в конце концов, очищает этот разряд, когда осуществляется следующий запрос ввода-вывода для этого канала.

Пример:

```
MOV     #PC$CSR,R5      ;запрашивает перфолен-
                        ;точное устройство вво-
                        ;да-вывода, засылает CSR
TST     (R5)+           ;устройство готово?
BPL     PCGORD          ;да, начало передачи
```

```
BIS     #EOF$,@—(R4)        ; не готов вход,
                            ; устанавливает EOF
BR      PCFIN               ; и операция завершения
```

В этом примере показано, как драйвер PC проверяет разряд готовности устройства, который он использует как признак своего EOF, устанавливает разряд EOF для программы пользователя и возвращается обратно в монитор. Благодаря этому соглашению для обозначения конца файла (EOF) программа воспринимает символьно-ориентированные устройства как устройства произвольного доступа к данным, что не противоречит идее системы ФОДОС-2 о независимости устройств.

2.5.1. Запрос .DRFIN используется для генерации команд для перехода обратно в монитор в конце секции завершения ввода-вывода. Он делает указатель текущего элемента очереди глобальным символом и генерирует позиционно-независимый код для возврата в монитор. Когда после возврата управление переходит к монитору, он освобождает текущий элемент очереди.

Формат запроса .DRFIN следующий: .DRFIN NAME
где NAME — двухсимвольное имя устройства.

2.6. Секция окончания драйвера

Целью секции окончания драйвера является объявление некоторых глобальных символов и установка таблицы указателей для смещений в резидентном мониторе. Указатели заполняются во время загрузки, если это драйвер системного устройства. В противном случае, они заполняются, когда драйвер делается резидентным посредством запроса .FETCH или команды монитора LOAD. Секция окончания драйвера предусматривает также символ для определения размера драйвера. Для генерации окончания драйвера используется запрос .DREND.

2.6.1. Запрос .DREND

Формат запроса .DREND следующий: .DREND NAME
где NAME — двухсимвольное имя устройства.

2.6.2. Фиктивные устройства

В системе ФОДОС-2 предусмотрена возможность написания драйвера для фиктивного устройства (устройство, которое не прерывается и не является большим запоминающим устройством), чтобы использовать преимущества организованной в очередь системы ввода-вывода и того факта, что драйверы могут оставаться резидентными в памяти. Примерами драйверов для фиктивных устройств являются драйверы NL (драйвер фиктивного устройства) и MQ (драйвер обмена между заданиями).

Все выполняемые коды таких драйверов должны содержаться в секции инициирования ввода-вывода. Затем драйвер должен вызвать запрос .DRFIN для окончания операции и возврата элемента очереди. Так как фиктивные устройства не прерываются, то драйверам не нужна секция обработки прерываний и запрос .DRAST.

3. НАСТРОЙКА И ПРОВЕРКА ПРОГРАММЫ

3.1. Общее описание драйвера устройства

Ниже приведен пример, в котором показана структура для простого драйвера устройства SK.

Пример:

```
.TITLE SK          V05.01
;Драйвер устройства SK
.IDENT /V05.01/
.SBTTL секция определений
.MCALL .DRDEF
.DRDEF SK,377,WONLY$,0,177514,200
SKBR=SK$CSR+2                                  ; регистр буфера SK
SKIE=100                                       ; разряд разрешения
                                               ; прерывания
.SBTTL      секция заголовка
            .DRBEG SK
.SBTTL      секция инициирования
            MOV       SKCQE,R4                 ; R4 указывает текущий
                                               ; элемент очереди
            ASL       Q$WCNT(R4)               ; делает из счетчика слов
                                               ; счетчик байтов
            BEQ       SKDONE                   ; переход к непосредствен-
                                               ; ному окончанию
            BCC       SKERR                    ; устройство только запи-
                                               ; си — запрос считывания
                                               ; запрещен
RET:        BIS       #SKIE,@#SK$CSR           ; разрешение прерывания
            RTS       PC                       ; ожидание прерывания
  .SBTTL    секция обработки прерывания
            .DRAST    SK,4,SKDONE
            MOV       SKCQE,R4                 ; R4 указывает текущий
                                               ; элемент очереди
            BIT       #100200,@#SK$CSR         ; ошибка или готовность?
            BMI       RET                      ; ошибка, необходимо ис-
                                               ; правление
            BEQ       RET                      ; нет готовности — выход
                                               ; и ожидание
            BIC       #SKIE,@#SK$CSR           ; запрещение прерываний
            .FORK     SKFBLK                   ; код, переводящий про-
                                               ; цесс на FORK — уровень
            ADD       #Q$WCNT,R4               ; переместить указатель
                                               ; элемент очереди
```

```
SKNEXT:  TSTB    @#SK$CSR        ; готовность для следую-
                                 ; щего символа?
         BPL     .RET            ; нет — переход назад
         TST     @R4             ; печатать немного левее?
         BEQ     SKDONE          ; нет — передача законче-
                                 ; на
         MOVB    @-(R4),R5       ; принять символ
         INC     (R4)+           ; увеличить указатель бу-
                                 ; фера
         INC     @R4             ; увеличить счетчик сим-
                                 ; волов
         BIC     #^C<177>,R5     ; 7-разрядный ASCII
         MOVB    R5,@#SKBR       ; переслать символ в уст-
                                 ; ройство
         BR      SKNEXT          ; то же самое для других
                                 ; символов

.SBTTL   секция завершения ввода-вывода
SKERR:   BIS     #HDERR$,@-(P4)  ; установить разряд ошиб-
                                 ; ки в CSW
SKDONE:  BIC     #SKIE,@#SK$CSR  ; запретить прерывания
         .DRFIN  SK              ; возврат в монитор
SKFBLK:  .WORD   0,0,0,0         ; элементы FORK-очереди
.SBTTL   секция окончания драйвера
         .DREND  SK
.END
```

3.2. Драйверы, которые формируют внутреннюю очередь

Драйвер устройства может использовать одну или больше своих собственных очередей, необходимых для запросов ввода-вывода, вместо использования обычной очереди ввода-вывода монитор-драйвер. Внутренняя очередь предназначена для одновременного выполнения нескольких операций, т. е. драйвер может обслужить несколько запросов с одним обращением к устройству. Внутренняя очередность возможна в этом случае, но многое зависит от конкретной ситуации. Для драйвера и устройств, для которых внутренняя очередность невозможна или нецелесообразна, ее не рекомендуется использовать.

Другим примером является очередь сообщений системы ФОДОС-2, выполненная посредством драйвера MQ для связи системных заданий. Если одно задание посылает сообщение второму заданию, и второе задание не принимает сообщение, драйвер MQ ожидает. Если процесс приема сообщения осуществляется через очередь, то драйвер MQ обрабатывает его.

Для этого он использует первоначально посланный запрос из очереди монитор — драйвер, ставит его во внутреннюю очередь и, затем, обслуживает запрос приема.

Вообще, драйвер следует простой процедуре для осуществления формирования в очередь. Когда запрос ввода-вывода делается для драйвера, он является первым и единственным запросом в очереди монитор — драйвер. Как только запрос будет принят, драйвер ставит его во внутреннюю очередь и очищает DDCQE и DDLQE для «удаления» запроса из очереди монитор — драйвер. Важно, что элемент очереди все еще занят — он еще используется драйвером.

3.2.1. Формирование внутренней очереди

Когда драйвер сначала вводится для запроса, в шестом слове он должен проверить наличие элемента очереди. Недопустимый запрос немедленно приводит к фатальной ошибке.

Если запрос сделан для быстрозавершающейся процедуры, как, например, установка перфоленты, драйвер выполнит операцию. Затем он выдаст запрос .DRFIN для освобождения элемента очереди и сообщение о завершении операции для запрашивающей программы. В общем, драйвер выполняет операцию, если эта операция может выполниться быстро и синхронно.

Если запрос сделан для процедуры, требующей вычислений и времени, драйвер ставит запрос во внутреннюю очередь посредством использования слова связи элементов очереди. Слово связи равно 0, так как этот элемент является первым и единственным в очереди.

В общем, драйвер устанавливает запрос во внутреннюю очередь, если этот запрос требует работы и времени и должен выполняться асинхронно. Если во внутренней очереди этот запрос является первым, драйвер начинает операцию, ожидает ее завершения и выводится командой RTS PC. Если этот запрос не является первым во внутренней очереди, драйвер не начинает операцию и выводится командой RTS PC.

3.2.2. Обработка прерываний для драйверов, формирующих внутреннюю очередь

Когда операция завершается, драйвер переходит в точке входа прерывания DDINT:. Если внутренних очередей больше, чем одна, драйвер определяет, какой запрос включает это прерывание. Если операция незавершена, драйвер начинает ее вновь и возвращается в монитор. Если передача за-

вершена, драйвер должен поставить запрос, находящийся во внутренней очереди, снова в очередь ввода-вывода монитор — драйвер посредством установки DDCQE и DDLQE. В этой ситуации драйверу нужно вернуть запрос в основную очередь ввода-вывода, но ему нужно также продолжать выполнение (а не сразу возвращаться в монитор), чтобы проверить внутреннюю очередь на случай другого невыполненного запроса.

Чтобы вернуть запрос в монитор без вывода, драйвер должен выполнить подпрограмму .DRFIN.

Пример:

```
MOV     DDCQE,—(SP)     ;в случае, когда в очереди мо-
                        ;нитор — драйвер есть эле-
                        ;мент, тогда возможно преры-
                        ;вание
MOV     R4,DDCQE        ;элемент внутренней очереди
MOV     R4,DDLQE        ;помещается в очередь мони-
                        ;тор — драйвер
CLR     Q$LINK(R4)
MOV     PC,R4           ;выход на запрос
ADD     #DDCQE—.,R4     ;.DRFIN
MOV     @#54,R5         ;через
JSR     PC,@270(R5)     ;подпрограмму JSR
MOV     @SP,DDCQE       ;восстановление возможных
MOV     (SP)+,DDLQE     ;других элементов очереди
.
.
.
(Проверка новой внутренней очереди и начало другой операции, если необходимо)
.
.
RTS     PC              ;возврат
```

В этом примере показано, как драйвер использует внутреннюю очередь. (R4 указывает элемент внутренней очереди, его третье слово.)

3.2.3. Процедуры преждевременного прерывания для драйверов, формирующих внутреннюю очередь

Несмотря на то, устанавливает ли драйвер запросы во внутреннюю очередь или нет, ФОДОС-2 использует счетчик невыполненных запросов ввода-вывода для каждого канала. Общее число невыполненных запросов ввода-вывода находится в резидентном мониторе. Когда задание преждевременно прерывается, некоторые невыполненные запросы ввода-выво-

да должны устраняться из счетчиков. Это происходит автоматически, если драйвер полагается на очередь ввода-вывода монитор — драйвер.

Если же драйвер осуществляет внутреннюю очередь ввода-вывода, он должен следовать процедуре понижения счетчика невыполненных запросов ввода-вывода. Эта процедура должна гарантировать, что драйвер будет вводиться, когда задание преждевременно прерывается, несмотря на то, имеет драйвер активные элементы очереди или нет, он устанавливает разряд 11, HNDLR$, в слове состояния устройства DDSTS, когда драйвер вызывает запрос .DRDEF. В системах FB и XM это приводит к тому, что драйвер будет вводиться при всех преждевременных прерываниях, даже если в очереди монитор — драйвер ничего нет. (Монитор SJ игнорирует этот разряд, так как в системе одного задания такой проблемы нет.)

Если драйвер вводится в точке входа преждевременного прерывания, то он должен проверять внутреннюю очередь на элементы, принадлежащие преждевременно прерванному заданию. (R4 всегда содержит номер преждевременно прерванного задания.) Драйвер должен очищать внутреннюю очередь от этих элементов, и должна следовать остановка процедуры для снижения счетчика монитора невыполненных запросов ввода-вывода. Регистры R0—R3 должны быть сохранены и если DDCQE имеет ненулевое значение, то драйвер:

— перемещает следующий внутренний элемент для прекращения задания;
— связывает элементы вместе посредством слова связи элементов (ELW), последнее слово связи элементов должно быть нулевым; устанавливает DDLQE в точку последнего элемента в преждевременно прерванном задании;
— если DDCQE указывает на элемент, принадлежащий преждевременно прерванному заданию, ввод-вывод прекращается, и используется запрос .DRFIN; если ввод-вывод не прекращается, используется директива RTS PC, ожидается прерывание, потом используется .DRFIN; если DDCQE не указывает на элемент, принадлежащий прерванному заданию, просто используется директива RTS PC.

Если DDCQE имеет нулевое значение, то драйвер:

— перемещает элементы внутренней очереди, которые принадлежат прерванному заданию, если их нет, то просто используется директива RTS PC;
— связывает элементы вместе, как описано выше, устанавливает DDCQE в точку первого элемента очереди, а DDLQE—

в точку последнего (последнее слово связи элементов должно быть нулевым);
— использует запрос .DRFIN.

3.3. Параметры SET

Команда клавиатурного монитора SET позволяет изменить определенные характеристики драйвера устройства. Драйвер должен находиться на системном устройстве файлом с именем DD&.SYS (DDX.SYS для XM монитора), где DD — двухсимвольное имя устройства.

Пример:

```
SET LP WIDTH=80
```

В этом примере устанавливается восьмидесятиколоночная ширина печати для устройства LP: (по умолчанию — 132 колонки).

Другой тип команды SET может разрешать или запрещать функцию.

Пример:

```
SET LP CR       (устанавливает возврат каретки;
                то же — по умолчанию)
SET LP NOCR     (не делает возврат каретки)
```

В этом примере показано, как с помощью префикса NO отрицается параметр CR (префикс NO всегда стоит перед параметром).

Драйвер устройства может содержать код для осуществления различных параметров. Ниже описано, как добавлять параметр SET к драйверу. Это добавление влияет только на файл драйвера, нет необходимости делать изменения в мониторе. (Параметры SET действительны и для устройств данных, и для системных устройств.)

3.3.1. Как выполняется команда SET

Команда SET целиком и полностью приводится в действие таблицей в блоке 0 файла драйвера и посредством установки подпрограмм, также в блоке 0, которые изменяют команды и данные в блоках 0 и 1 драйвера. (Блок 0 относится к адресам с 0 до 776, заголовок драйвера начинается в блоке 1, в ячейке 1000 файла драйвера.)

После печати команды SET на пульте терминала, монитор производит анализ командной строки и ищет файл драйвера системного устройства DD.SYS (DDX.SYS в XM мониторе). Нет необходимости в том, чтобы этот драйвер был установлен в выполняющейся системе. Затем монитор считывает блоки 0 и 1 драйвера в область буфера USR в памяти. Он просматривает таблицу в блоке 0, пока не найдет элементы таблицы для указанного параметра SET. По элементу таблицы он мо-

жет найти определенную подпрограмму, предназначенную для выполнения этого параметра и изменений, допускаемых этой подпрограммой, таких как NO или числовые значения. Затем монитор выполняет подпрограмму, которая содержит команды, изменяющие код в блоках 0 и 1 драйвера. Код в блоке 1 является частью основания драйвера и содержит команды для установок по умолчанию для всех параметров SET. После изменения кода, монитор записывает блоки 0 и 1 обратно на системное устройство. Поэтому, в результате команды SET, некоторые команды или данные в драйвере изменяются. Однако другие копии драйвера, находящиеся в памяти, недействительны.

3.3.2. Формат таблицы SET

Таблица для параметров SET состоит из серии четырехсловных элементов для параметра. Таблица начинается с ячейки 400 блока 0 драйвера и оканчивается нулевым словом. Для генерации таблицы используется запрос .DRSET (см. п. 3.3.3).

Первое слово таблицы является значением, которое будет передано в R3 для подпрограммы SET, связанной с параметром, когда монитор обрабатывает этот параметр. Это слово может быть числовым значением — таким, как число колонок по умолчанию для построчно-печатающего устройства — или командой для замены на другую команду в блоке 1 драйвера. Это слово не должно быть нулевым.

Второе и третье слова таблицы являются наименованием параметра в кодах RADIX-50 (например, CR или WIDTH). В таблице символы выравнены влево и заполнены пробелами.

Младший байт четвертого слова является указателем подпрограммы, которая изменяет код. Старший байт показывает тип допустимого параметра SET. Установка кода 100 показывает, что требуется восьмеричный аргумент. Установка кода 200 показывает, что префикс NO действителен для этого параметра. На рис. 1 показана таблица параметров SET.

Значение, находящееся в R3, для подпрограммы SET	
Имя параметра в кодах RADIX—50 (два слова)	
Коды для допустимого типа команд SET	Указатель подпрограммы SET

Рис. 1

3.3.3. Запрос .DRSET

Запрос .DRSET используется для установки таблицы параметров посредством вызова запроса один раз для каждого параметра таким образом, чтобы вызовы запроса следовали один за другим. Необходимо использовать запрос .DRSET после .DRDEF и перед запросом .DRBEG.

Формат вызова запроса .DRSET следующий:

.DRSET OPTION,VAL,RTN [,MODE]

где OPTION — имя параметра SET (например, CR или WIDTH); оно может содержать не более шести буквенно-цифровых знаков и не должно содержать пробелы или табуляцию;

VAL — параметр, который будет передан в R3 для подпрограммы; он может быть числовой константой, такой, как минимальная ширина колонки, или целой командой, заключенной в угловые скобки, для замены содержимого блоков 0 или 1 драйвера; он не должен быть нулевым;

RTN — имя подпрограммы, которая изменяет код в блоках 0 или 1 драйвера; подпрограмма должна следовать за таблицей параметров в блоке 0, но не выше адреса 776;

MODE — произвольный аргумент для показа типа параметра SET; необходимо ввести префикс NO для показа, что он действителен для параметра; необходимо ввести NUM, если требуется десятичное значение, OCT — если восьмеричное; пропуск аргумента MODE показывает, что OPTION не принимает ни префикс NO, ни числовые аргументы; конструкция <NO,NUM> показывает, что требуется и префикс NO, и десятичное значение; конструкция <NO,OCT> показывает, что требуется и префикс NO, и восьмеричное значение; пропуск аргумента MODE вызывает появление нуля в старшем байте последнего слова элементов таблицы.

Сначала запрос .DRSET выдает директиву .ASECT и устанавливает счетчик адреса 400, в начале таблицы. Он генерирует также нулевое слово для окончания таблицы. Так как запрос .DRSET оставляет счетчик адреса в конце таблицы, то необходимо поместить подпрограмму для изменения кода сразу же после того, как запрос .DRSET вызовет драйвер. Это гарантирует, что они расположатся в блоке 0 драйвера.

3.3.4. Подпрограмма изменения драйвера

Драйверу нужна одна подпрограмма для каждого допустимого параметра SET. Для версии NO для каждого параметра используется та же подпрограмма. Целью подпрограммы является изменение кода в основной части драйвера, ос-

нованное на команде SET, напечатанной на пульте терминала.

Подпрограммы должны следовать сразу после таблицы параметров, и они должны находиться в блоке 0 после таблицы и ниже адреса 1000. Код в основной части драйвера, которую изменяет подпрограмма, должен находиться в блоке 1 драйвера, в пределах первых 256-ти слов.

Имя подпрограммы является точкой входа по умолчанию. Эта точка входа для параметров, которые не принимают ни числовые значения, ни префикс NO, и для параметров, которые принимают префикс NO, но в данный момент его не имеют. Точкой входа для параметров, которые позволяют использовать префикс NO и имеют его, является точка входа по умолчанию +4.

При входе в подпрограмму для всех параметров разряд переноса чист, и регистры R0, R1 и R3 содержат информацию по использованию подпрограммы. Если для этого параметра допустимы числовые значения, то регистр R0 содержит числовое значение, содержащееся в командной строке с SET; R1 содержит указанный номер канала как часть имени устройства (если номер канала не указан, то разряд признака установлен); R3 содержит слово VAL таблицы параметров SET.

Подпрограмма может показать, что команда недопустима, посредством возврата с установленным разрядом переноса. Например, для построчно-печатающего устройства SET WIDTH не допускает ширину менее 30. Если параметры подпрограммы показывают неисправность, монитор печатает сообщение об ошибке и не записывает блоки 0 и 1. Таким образом можно сделать проверку после изменения кодов в блоке 1.

Добавив подпрограммы для каждого параметра в драйвере, можно использовать следующую строку кода, чтобы убедиться, что границы размера не нарушены:

```
.IIF GT,<.1000>, .ERROR . — 1000 ;код SET слишком велик!
```

Эта секция завершается директивой .ASECT, после которой необходимо установить счетчик адресов 1000. Затем можно продолжать обработку остального кода драйвера, начинающегося с запроса .DRBEG, который устанавливает заголовок драйвера.

3.3.5. Примеры параметров SET

Следующие далее примеры взяты из драйвера построчно-печатающего устройства и демонстрируют параметры SET, описанные ранее:

```
SET LP WIDTH=80
SET LP CR
SET LP NOCR
```

Сначала драйвер вызывает запрос .DRSET для установки таблицы параметров для двух параметров: WIDTH и CR.

Первый вызов показывает, что установлен параметр WIDTH построчно-печатающего устройства, что десятичное 30 — это значение по умолчанию, что O.WIDTH — это имя подпрограммы, которая изменяет код этого параметра, и что WIDTH — цифровой аргумент: .DRSET WIDTH,30.,O.WIDTH, NUM.

Следующий вызов показывает, что установлен параметр CR построчно-печатающего устройства, что «NOP» должен переходить к подпрограмме, что O.CR — это имя подпрограммы, которая изменяет код этого параметра, и что этот параметр CR может использоваться с приставкой NO: .DRSET CR,NOP,O.CR,NO.

Эти два вызова генерируют следующую таблицу:

```
.ASECT
.=400
.WORD   30.                     ; минимальное значение
                                ; WIDTH
.RAD50  \WIDTH\                 ; имя параметра
.BYTE   <O.WIDTH-400>/2
.BYTE   100
NOP                             ; невыполняемая команда
.RAD50  \CR  \                  ; имя параметра
.BYTE   <O.CR-400>/2
.BYTE   200
.WORD   0                       ; конец таблицы
```

Подпрограммы для обработки этих параметров следуют сразу же за таблицей.

Пример:

```
O.WIDTH:MOV     R0,COLCNT       ; перемещает значение от
                                ; пользователя
        MOV     R0,RSTC+2       ; в две постоянные ячейки
        CMP     R0,R3           ; сравнивает новое значе-
                                ; ние с минимальным зна-
                                ; чением WIDTH, 30
        RTS     PC              ; возврат, С-разряд
                                ; установлен на ошибку
```

В этом примере показана подпрограмма изменения параметра WIDTH. Команды в подпрограмме O.WIDTH изменяют данные в двух ячейках блока 1 драйвера.

Пример:

```
O.CR:  MOV    (PC)+,R3          ;точка входа для «CR»;
                                ;перемещает адрес сле-
                                ;дующей строки в R3
       BEQ    RSTC—CROPT+.      ;новая команда
       MOV    R3,CROPT          ;точка входа для «NOCR»
                                ;(O.CR+4);
                                ;перемещает или «NOP»,
                                ;или предыдущую строку
                                ;в CROPT
       RTS    PC                ;возврат
```

В этом примере приведена подпрограмма O.CR, которая имеет две точки входа: для параметра «CR» подпрограмма вводится у O.CR, для параметра «NOCR» — у O.CR+4. Важно, что:

1. Подпрограмме удается заменить одну из двух команд, расположенных в блоке 1;
2. Команда NOP перемещается в CROPT, если выбран параметр «NOCR»;
3. Команда BEQ RSTC—CROPT+. Перемещается в CROPT, если выбран «CR»;
4. Во время выполнения подпрограмм при обработке параметров SET регистры R4 и R5 недоступны для использования.

Конструкция команды BEQ необходима, потому что переход будет транслироваться в ячейку, отличную от той, из которой он будет выполнен. Во всех подпрограммах команда перехода должна использовать следующую конструкцию для генерации правильного адреса: BR A—B+.

где A — место назначения команды перехода;
B — адрес команды перехода;
. — текущее значение счетчика ячеек.

Обычно только подпрограммы для параметров, допускающих префикс NO, используют эти команды перехода.

И, наконец, приведен код секции обработки прерываний драйвера, который изменяется вышеприведенными подпрограммами. Код для изменения должен быть расположен в блоке 1 драйвера, в первых 256-ти словах.

Пример:

```
COLCNT: .WORD   COLSIZ          ;печать оставшихся
          .                     ;символов строки
          .                     ;
CHRTST: CMPB    R5,#HT          ;это символ табуляции?
        BEQ     TABSET          ;да, сбросить табуляцию
```

```
        CMPB  R5,#LF          ; это перевод строки?
        BEQ   RSTC            ; да, восстановить счетчик
                              ; колонок
        CMPB  R5,#CR          ; это возврат каретки?
CROPT:  NOP                   ; «NOP», если нет;
                              ; иначе «BEQ RSTC—
                              ; CROPT+.»
                              ; посредством подпро-
                              ; граммы SET в блоке 0
                              ; (если параметр «CR»)
        CMPB  R5,#FF          ; это перевод формата?
        BNE   IGNORE          ; нет, это не печать
RSTC:   MOV   #COLSIZ,COLCNT  ;переустановка счетчика
                              ;колонок
```

Из этого примера видно, как подпрограммы из блока 0 могут изменять данные и команды в блоке 1 драйвера.

3.4. Как проверить и отладить драйвер устройства

Как только новый драйвер будет оттранслирован, отредактирован и включен в систему (установлен), можно начинать его проверку. Во время отладки необходимо не забывать о том, что каждый раз необходимо устранять старый драйвер и устанавливать новый после создания его новой версии DD(X).SYS.

Проверка драйвера заключается в следующих трех стадиях.

1. Использование ODT для наблюдения за драйверами в процессе передачи ими данных (см. п.п. 3.4.1 и 3.4.2).

2. Проверка драйвера командами клавиатурного монитора, программами работы с файлами и программами ФОРТРАН или БЕЙСИК. Например, команду COPY (см. [2]) можно использовать для копирования данных на или с устройства или использовать для копирования PIP (см. [3]). Хорошо использовать драйвер с операторами BASIC INPUT или PRINT, или с операторами FORTRAN READ, или WRITE. Если драйвер устанавливает разряд в слове состояния устройства, который показывает, что драйвер предназначен для устройства системы ФОДОС-2 справочной структуры, DUP будет работать правильно в устройстве, без дальнейших изменений, т. е., можно использовать DUP для инициализации устройства (по переключателю /Z) и объединения неиспользуемой области (по переключателю /S). Программе RESORC не требуется изменения для признания нового устройства и включения его в сообщение SHOW DEVICES.

3. Дать драйверу расширенную разработку с прикладной программой, которая использует режим ожидания ввода-вывода, асинхронный ввод-вывод и подпрограмму завершения.

Когда драйвер пройдет всю проверку успешно, можно использовать его как часть системы ФОДОС-2.

3.4.1. Использование ODT для проверки драйвера

Лучший способ использования ODT для проверки драйвера — это выполнение ODT как основного задания. Если используется монитор SJ, то имеет смысл переключить его на FB на время отладки. Во время отладки рекомендуется быть основным пользователем.

Необходимо загрузить систему с помощью аппаратного загрузчика, не начиная выполнений системных заданий и загрузки драйверов.

Ниже приведена команда, по которой ODT связывается с основным заданием: LINK/MAP/FOREGROUND ODT.

Затем необходимо загрузить драйвер устройства, который необходимо отладить: LOAD DD[X].

Теперь необходимо выполнить команду SHOW D. Запомним адрес, определенный для драйвера устройства, например, 131634. Вычитаем 6 (восьмеричное) и получаем базу адреса драйвера:

```
131634
—   6
------
131626
```

Запускается ODT как основное задание:

```
FRUN ODT
ODT V01.04
*
```

Засылается в регистр смещения 0 значение, вычисленное из адреса по команде SHOW D: 131626;0R

Можно перемещаться по драйверу в памяти по мере следования инструкций листинга трансляции. Пять первых слов являются заголовком; первая исполняемая команда является шестым словом. Поэтому лучше всего установить первую точку разрыва у шестого слова: 0,12; 0B

Другие точки разрыва можно поставить в тех частях драйвера, которые необходимо проверить во время отладки. Другим критическим местом является точка входа прерывания. Соответствующую ячейку можно найти проверкой листинга драйвера (точка входа прерывания называется DDINT:).

После установки точек разрыва можно выходить из ODT: 0;G

Теперь можно пытаться использовать драйвер. Например, использовать DUP для инициализации устройства или PIP для копирования на устройство данных, или выполнить тест-программу, предназначенную специально для этой цели. Когда выполнение достигает первой точки разрыва, ODT принимает управление. ODT используется как обычно для проверки ячеек и их значений или изменения команд. Приоритет ODT по умолчанию равен 7; это предотвращает от вмешательства других прерываний во время сеанса отладки.

Если работа драйвера устраивает, то необходимо устранить из него точки разрыва и приступить к дальнейшему выполнению посредством драйвера:

;B
;P

Недопустима разгрузка основного задания (ODT), если точки разрыва еще установлены в драйвере.

3.4.2. Использование ODT в XM

Необходимо тщательно выбрать место для ODT в памяти. Можно связать его с прикладной программой или так, чтобы он размещался в памяти там, где он не будет разрушен. Если точка разрыва должна быть выбрана во внутреннем режиме, ODT не должен размещаться в области PAC1 (ячейки с 20000 по 37776). Самое безопасное место для ODT — это секция основного задания (см. п. 3.4.1).

При отладке с использованием ODT страница ввода-вывода должна быть отображена.

Установка точек разрыва требует осторожности. После введения ODT необходимо проверить вектор прерывания по точке разрыва (BPT) в ячейках 14 и 16 нижней памяти. После установки точек разрыва необходимо вручную установить разряды текущего режима, разряды 14 и 15, PS в ячейке 16. Затем ожидается точка разрыва. Значение 11 предназначено для режима пользователя, 00 для системного режима. Программы работы с файлами системы ФОДОС-2, такие как PIP и DUP, выполняются в режиме пользователя и предполагают, что разряды режима будут установлены 11.

После установки точек разрыва необходимо напечатать 0;G для выхода из ODT. Это заставляет ODT выполнить запрос .EXIT, который разрушает вектор BPT. Поэтому после выхода из ODT необходимо вручную реконструировать содержимое вектора посредством использования команды DEPOSIT следующим образом:

D 14=(истинное содержимое ячейки 14),(истинное содержимое ячейки 16)

Необходимо, чтобы другие задания в это время не выполнялись, так как переключение контекста вызовет выход из строя этого метода.

4. ПОДПРОГРАММА ОБРАБОТКИ ПРЕРЫВАНИЙ

Этот раздел описывает способы, с помощью которых программа может передавать данные между памятью и периферийными устройствами. Сначала излагается программируемый ввод-вывод без прерывания, затем вводятся концепции для использования прерываний для обслуживания устройств ввода-вывода посредством сравнения преимуществ и недостатков подпрограмм внутренней обработки прерываний и драйверов устройств. После этих общих указаний описывается структура подпрограмм обработки прерываний и показывается в деталях, как организовать и писать их. Пример схемы основной программы, которая содержит подпрограмму обработки прерываний, заканчивает обсуждение. В конце раздела — применение подпрограмм обработки прерываний в XM-системе.

4.1. Программируемый ввод-вывод

Первым способом передачи данных между памятью и периферийным устройством является программируемый ввод-вывод. В соответствии с этим способом программа оперирует с устройствами, запрещая прерывания и используя признаки для координации передачи данных. Программа проверяет разряд готовности в регистре состояния соответствующего устройства, посылает данные в соответствующий момент и, затем, ожидает другой сигнал готовности или делает другую обработку и, время от времени, подсчет устройств. Программируемый ввод-вывод зависит от конкретного устройства и не должен использовать возможности операционной системы, предназначенные для процессов ввода-вывода. Кроме того, он занимает ресурсы системы до окончания ввода-вывода.

Однако программируемый ввод-вывод в то же время является лучшим практически методом. Например, резидентный монитор использует программируемый ввод-вывод для печати своего сообщения об ошибке: ?MON—F—SYSTEM—HALT. Сначала он выполняет операцию RESET для остановки всех активных процессов ввода-вывода. Потом он ждет в сжатом цикле для системного терминала, чтобы напечатать сообщение об ошибке по одному символу за раз. Очевидно, в такой ситуации, где монитор может быть испорчен, ни одно другое задание или передача данных не могут выполняться, и системный терминал разрешен только как устройство вывода.

Подпрограмма монитора .PRINT также может быть испорчена и не может быть использована. Учитывая эти требования, программируемый ввод-вывод является лучшим способом для печати сообщения об ошибке.

В прикладных программах можно использовать программируемый ввод-вывод для критических по времени устройств, когда программа должна реагировать так быстро, как только символ будет находиться в регистре.

Пример:

```
;
; R1 указывает текст сообщения.
; TTPS — слово в памяти, содержащее адрес регистра состоя-
; ния печатающего терминала; его признак готовности уста-
; навливается в последнем по счету разряде младшего байта
; (разр. 7).
; TTPB — слово в памяти, содержащее адрес буфера термина-
; ла.
; Пересылка символов в буфер восстанавливает признак заня-
; тости в регистре состояния
;
5$:     TSTB    @TTPS           ; проверить, TT — использу-
                                ; ется?
        BPL     5$              ; если да, проверить снова
        MOVB    (R1)+,@TTPB     ; если нет, печатать сим-
                                ; вол
        BNE     5$              ; переход назад, если пе-
                                ; чатать дальше
```

В этом примере из RMON демонстрируется программируемый ввод-вывод.

Драйвер устройства гибких дисков с одинарной плотностью записи (в дальнейшем просто гибкие диски, если не оговорена особо плотность записи), DX, предусматривает другой пример программируемого ввода-вывода. При считывании данных с гибкого диска по одному сектору за раз драйвер сначала требует считывания одного сектора. Устройство гибких дисков завершает операцию считывания, размещает данные во внутреннем буфере и выходит к прерыванию. Драйвер запрещает прерывания и использует программируемый ввод-вывод для передачи данных из внутреннего буфера в память. Когда внутренний буфер снова готов для считывания следующего сектора, драйвер разрешает прерывание снова.

Пример:

```
;
```

```
; R4 указывает регистр состояния гибкого диска;
; R5 указывает внутренний буфер;
; R2 указывает буфер данных в памяти.
;
TRBYT:  TSTB    @TTPS       ;ожидает окончания пе-
                            ;редачи
        BPL     TRBYT       ;ветвление, если переда-
                            ;ча не закончена
EFBUF:  MOVB    @R5,(R2)+   ;передача символа
        DEC     @SP         ;проверка на окончание
                            ;счетчика
        BGT     TRBYT       ;дальнейшая передача
```

4.2. Ввод-вывод, обрабатываемый по прерываниям

Хотя программируемый ввод-вывод в некоторых случаях лучше, вообще, лучшим способом обработки устройств ввода-вывода является обработка прерываний. В соответствии с этим способом программа начинает передачу ввода-вывода, не прекращая обработку. Когда передача завершается, устройство выходит на прерывание. Подпрограмма обработки прерываний определяет, завершена передача или нет или произошла ошибка, и она выполняет соответствующие функции (продолжает передачу, возвращается в программу или, возможно, повторяет передачу в случае ошибки). Преимуществом использования прерываний является возможность выполнения двух и более конкурирующих процессов, и ресурсы системы не монополизируются.

4.2.1. Как работают прерывания

Прерывания — вынужденные передачи выполнения программы — возникают в случае таких внешних причин, как завершение ввода-вывода. Состояние процессора перед прерыванием сохраняется в стеке, так что обработка может продолжаться спокойно после возвращения из прерывания. Процессор сохраняет слово состояния процессора (PS), которое содержит текущее состояние машины, и счетчик команд (PC), который указывает адрес возврата.

Затем процессор загружает новое содержимое PS и PC из двух предварительно назначенных ячеек нижней памяти, называемых вектором прерывания. Эти слова содержат адрес подпрограммы обработки прерывания и новое слово состояния процессора (PS), которое указывает новый приоритет процессора. Когда подпрограмма обработки прерывания завершается, она выполняет команду RTI, которая восстанавливает старые PC и PS из стека, и возобновляется выполнение в прерванной программе.

4.2.2. Приоритеты устройств и процессора

Обработка прерывания тесно связана с приоритетом процессора и устройств. Рис. 2 иллюстрирует структуру **приоритетов ФОДОС-2.**

рис.2

Каждое устройство системы имеет назначенный ему приоритет, и в первую очередь будет обслужено устройство с более высоким приоритетом после прерывания. Кассетная лента, например, имеет приоритет 6, диск, обычно, 5, терминал и другие символьно-ориентированные устройства — 4. Эти приоритеты точно назначены системой и, вообще, регулируются посредством переключателя сменных приоритетов в каждом интерфейсе устройств ввода-вывода. Можно управлять порядком устройств с одинаковым приоритетом. Для этих устройств первое занявшее ЦП (центральный процессор) обслуживается вперед других устройств, когда одновременно возникают прерывания.

Центральный процессор (ЦП) оперирует с любым одним из восьми уровней приоритета, от 0 до 7. Когда ЦП работает с приоритетом 7, ни одно устройство не может его прервать. Когда ЦП оперирует с более низким приоритетом, причиной прерывания может стать лишь устройство с более высоким приоритетом. Можно устанавливать приоритеты процессора внутри подпрограммы обработки прерывания, изменяя слово состояния процессора. В системе ФОДОС-2 программное обеспечение предусматривает это, и можно напрямую изменять PS самостоятельно. Оно включает программные запросы .MTPS и .MFPS и запросы .INTEN и .FORK.

Система прерываний позволяет процессору последовательно сравнивать его собственный приоритет с приоритетом прерывающего устройства и выявлять устройство с более высоким приоритетом, чем приоритет процессора. Эта система может быть вложенной, т. е. обработка одного прерывания может быть прервана прерыванием с более высоким приоритетом. Обслуживание устройств с низким приоритетом продолжается, когда закончено обслуживание устройств с более высоким приоритетом.

4.2.3. Слово состояния процессора (PS) занимает верхний адрес страницы ввода-вывода.

Оно содержит информацию о текущем состоянии машины. Эта информация содержит текущий приоритет процессора, текущий и предшествующий операционные режимы, коды условий, описывающих результаты последней директивы и указатель, вызывающий прерывание после выполнения директивы (используется для отладки программ).

Рис. 3 показывает разряды PS. Разряды с 5 по 7 определяют текущий приоритет.

При изменении разрядов, необходимо изменять приоритет ЦП. Можно изменить приоритет до 7, например, для предотвращения возникновения любого другого прерывания. Когда необходима обработка особого прерывания, можно изменить приоритет процессора до приоритета устройства так, что только устройства с высшим приоритетом будут прерывать эту подпрограмму обработки. (Обслуживаемое устройство прервать нельзя.) Вообще, нет необходимости в самостоятельном доступе к PS, можно использовать запросы системы ФОДОС-2, такие, как .INTEN и .FORK, для изменения приоритета процессора.

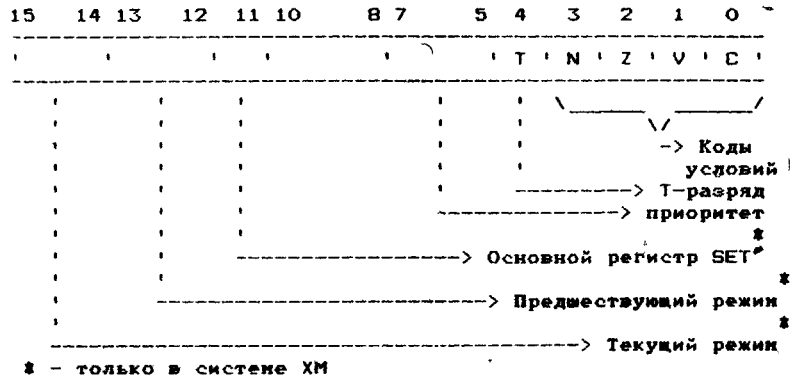


рис.3.

4.3. Внутренние подпрограммы обработки прерываний в сравнении с драйверами устройств

Т. к. в системе ФОДОС-2 используются как программируемый (непрерывный), так и драйверный (прерывный) ввод-вывод, когда необходимо включить новое устройство в систему, которое уже не поддерживается системой, сначала необходимо решить, использовать внутреннюю подпрограмму обработки прерываний или написать драйвер. Каково бы ни было решение, и подпрограмма обработки прерываний, и драйвер могут включать как программируемую секцию ввода-вывода, так и драйверный код. Обычный интерфейс системы ФОДОС-2 между монитором и периферийным устройством есть драйвер устройства, который выполнен как файл образа памяти на устройстве массового хранения и находится в памяти, если он необходим для выполнения ввода-вывода устройства (см. разд. 2). Драйвер устройства обычно включает в себя подпрограмму обработки прерываний.

Если для использования выбрана подпрограмма обработки прерываний, необходимо разместить программу в памяти так, чтобы программа напрямую меняла регистр состояния и регистр буфера для указанного устройства, и подпрограмма обработки прерываний могла обслуживать прерывания внутри собственных кодов, т. е. коды программы обработки прерываний всегда должны быть в памяти.

Если решено использовать драйвер, коды подпрограммы обработки прерываний должны находиться в драйвере, а не в программе. Выходить из основной программы необходимо посредством запросов .READ и .WRITE, и монитор, и драйвер вместе инициируют передачу данных, обрабатывают прерывания и сообщают программе, когда передача завершается. В системе SJ или для фонового задания в FB драйвер должен быть резидентом, только когда программа непосредственно нуждается в нем для выполнения ввода-вывода (т. е. драйвер должен быть всякий раз резидентным, когда файл или канал открыт). Для основного и системного задания для FB и XM необходимо загружать драйвер, используя команду монитора LOAD) перед выполнением программы, так что драйвер всегда резидентен.

Следует использовать внутреннюю подпрограмму обработки прерываний для сенсоров и управляющих устройств, таких, как аналого-цифровые преобразователи. С помощью драйверов следует обслуживать устройства файловой структуры, такие, как диски, за исключением жестких дисков вин-

честерского типа. Этим методом можно обслуживать много подсоединяемой аппаратуры.

Двумя большими преимуществами внутренних подпрограмм обработки прерываний являются скорость и количество управляющей информации, т. к. монитор не участвует в передаче данных, внутренняя подпрограмма обработки прерываний может часто обрабатывать прерывания надежнее, чем драйверы устройств. Если скорость обработки прерываний не является определяющим фактором программы, можно выбрать написание внутренней подпрограммы обработки прерываний, даже если устройство — диск.

Внутренняя подпрограмма обработки прерываний имеет доступ ко всем регистрам состояния и регистрам буфера. (Драйвер тоже имеет доступ ко всем этим регистрам, но не имеет программа, использующая драйвер). Она может передать часть информации программе. Это обеспечивает большую гибкость в способе программного вызова подпрограммы обработки прерываний и в количестве информации, возвращаемой из подпрограммы обработки прерываний.

Тремя большими преимуществами использования драйверов является обеспечение независимости устройства от программы пользователя, они могут разделять время процессора с другими процессами и просты в использовании. Драйверы имеют стандартный протокол для присоединения к монитору системы ФОДОС-2. Есть также стандартный протокол для интерфейса между монитором и программой, так что любая программа, которая согласована со стандартами монитора, может использовать драйверы. Это программы пользователя, вспомогательные программы системы и языковые процессоры ФОДОС-2: АССЕМБЛЕР, ФОРТРАН и БЕЙСИК. Таким образом, драйвер делает новое устройство доступным большому количеству программ без специальной модификации, кроме того, драйвер для устройств произвольного доступа к данным делает файловую систему ФОДОС-2 применимой на устройстве без дополнительных затрат. Напротив, внутренняя подпрограмма обработки прерываний делает новое устройство доступным одной прикладной программе.

Драйверы устройств удобны для использования. Они стандартны для обслуживания устройств ввода-вывода, процедура для их написания и использования ясна и доступна, поскольку ФОДОС-2 предусматривает использование запросов при написании драйверов; имеются также команды клавиатурного монитора для установки драйвера в таблицу устройств монитора и загрузки в память. Кроме того, драйверы

позволяют извлекать выгоду из программных запросов монитора для выполнения передачи данных. Наконец, драйвер это лишь способ соединения устройства с виртуальным заданием в системе ХМ.

Рис. 4 иллюстрирует различия между внутренней подпрограммой обработки прерываний и драйверами.

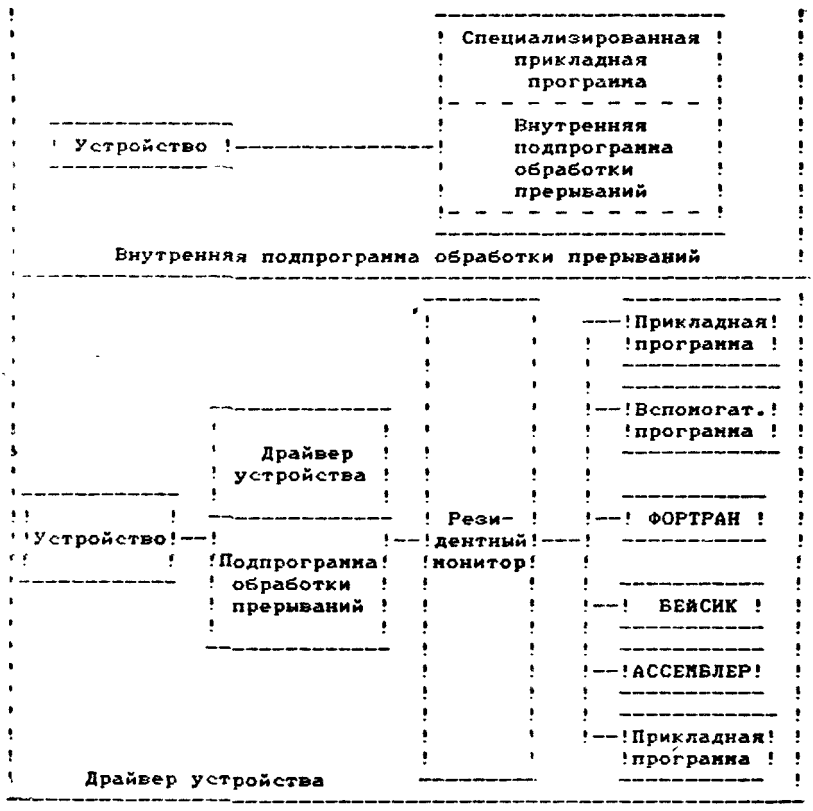


Рис.4.

4.4. Как планировать подпрограммы обработки прерываний

Наиболее важной частью написания подпрограммы обработки прерываний является момент ее планирования. Ниже приведено руководство.

1. Изучить устройство.

2. Изучить структуру подпрограммы обработки прерываний.

3. Изучить схему подпрограммы обработки прерываний.
4. Продумать требования к программе.
5. Подготовить блок-схему программы.
6. Написать коды.
7. Проверить и отладить программу.

4.4.1. Изучить устройство

Этот пункт является решающим для написания правильно работающей подпрограммы обработки прерываний. Необходимо внимательно изучить документацию к нему. Невзирая на формат документации (будь то руководство или брошюра), она должна содержать существенную информацию, которая необходима для поддержки его системой ФОДОС-2. Необходимо получить эту информацию.

Необходимо понять, как работает устройство, в чем оно нуждается и как оно обслуживает передачи данных. Можно использовать следующий проверочный список для уверенности в том, что имеется достаточная информация специфических устройств для написания подпрограммы обработки прерываний. Необходимо изучить каждый вопрос, прежде чем писать подпрограмму обработки прерываний. Некоторые из следующих вопросов неприменимы ко всем типам устройств: некоторые относятся для большего числа носителей, некоторые больше относятся к сенсорам и коммуникациям. Необходимо хорошо разобраться в каждом вопросе, чтобы увидеть, подходит ли он к данному устройству.

1. Что такое вектор прерывания устройства?

Необходимо решить, что сделать вектором прерываний, приняв во внимание как конфликт с существующими устройствами, поддерживаемыми системой ФОДОС-2, так и конфликт с устройствами, поддерживаемыми другими операционными системами, если они используются. Если вектор выбран, необходимо убедиться, что устройство установлено правильно, и аппаратура скачет по этому адресу. В системе ФОДОС-2 используются все вектора по адресам ниже 500, и их нельзя использовать как вектора прерываний. В разд. 2 приводится список векторов периферийных устройств.

2. Что такое регистры состояния?

Необходимо изучить, где эти регистры расположены и что означает в них каждый разряд.

3. Какой у устройства приоритет?

4. Устройство произвольного доступа к памяти или программируемой передачи (словно- или символьно-ориентированное).

5. Где находятся регистры буферов данных?

Необходимо изучить, где расположены регистры и что означают разряды в каждом случае.

6. Каковы коды для типичных операций?

Необходимо изучить, как инициировать любую операцию, манипулируя разрядами регистров устройства. Драйверы устройств должны выполнять операции чтения, записи, поиска и сброса.

7. Когда происходит прерывание устройства?

Одни устройства прерываются для каждого символа, другие ориентируются на слова, блоки или пакеты. Некоторые устройства прерываются дважды для определенных операций, таких, как поиск или сброс привода. Необходимо искать выход, если устройство делает это, и планировать заранее, как будет приниматься в расчет эта информация позже.

8. Какой привод является основным для передачи данных?

Это, конечно, относится к предыдущему вопросу, но необходимо определить, как посылаются запросы ввода-вывода в устройство: байтами, словами или блоками. Если, например, программа оперирует словами, а устройство ориентировано на символы, необходимо преобразовать слова в байты в подпрограмме обработки.

9. Устройство нуждается в положительном или отрицательном счетчике байтов?

Некоторым устройствам требуется отрицательный счетчик байтов или слов. Если данное устройство — одно из них, необходимо сделать счетчик байтов (слов) в подпрограмме обработки прерываний отрицательным.

10. Какова структура устройства, его геометрия?

Если устройство — диск, необходимо разобраться, какова структура цилиндров, дорожек и секторов, определить их размер, выяснить, требует ли устройство чередования записи и если да, то изучить, как оптимизировать скорость записи. (Чередование описывает процесс записи данных на вращающееся устройство, которое требует программного вмешательства между секторами. Диск вращается постоянно, данные записываются в один сектор, программа вмешивается, как только минует смежный сектор, тогда следующие данные записываются в следующий имеющийся сектор).

11. Каков механизм буферизации?

Некоторые устройства передают данные программе по одному символу за раз. Другие собирают данные во внутренний буфер или передают их пакетами. Необходимо решить, как буферизовать данные в программе, убедиться, что размер области, отведенной под буфер, достаточно велик.

12. Как выполнять вычисление адреса данных на устройстве?
Это относится к структуре устройства. Необходимо изучить устройство и определить, как искать нужные данные. Номера блоков должны быть преобразованы в специальные адреса устройства. (Некоторые процессоры не имеют инструкций умножения и деления.)

13. Какие собственные операции требуются устройству?
Некоторым устройствам требуется выполнить сброс в начальное состояние перед повторением, другим — чтобы устройство было доступным или чтобы заполнение диска было гарантировано перед выполнением любой операции на нем. Необходимо, например, делать сброс в начальное состояние после незавершенного поиска или ошибки привода.

14. Как обслуживать ошибки и условия исключения?
Сначала необходимо решить, какие ошибки будут невосстановимыми, и передача должна быть преждевременно прервана, и какие ошибки восстановимы, и должно быть повторение передачи. Обычные восстановимые ошибки: ошибки контрольной суммы, ошибки запаздывания данных, временные ошибки. Необходимо решить, сколько времени повторять передачу для восстановимых ошибок и как обслуживать условия невосстановимых ошибок.

15. На что обратить внимание при преждевременном прерывании?
Необходимо выяснить, относительно быстрое или медленное устройство. Недопустимо использование очистки контроллера, если только один привод двухприводного контроллера затрагивается в результате преждевременного прерывания программы, т. к. это может помешать работе со вторым приводом. Такое же внимание стоит уделить двухпортовым устройствам.

4.4.2. Подготовка блок-схемы программы

Многие программисты готовят блок-схему после написания программы или совершенно ею пренебрегают. Однако блок-схема может помочь найти логические неточности и ошибки. К сожалению, блок-схема не окажет большой помощи в условиях конкуренции заданий. (Условия такого рода — ситуация, в которой два или более процессов пытаются модифицировать структуру общих данных одновременно; в результате, структура данных разрушается, и нужно идти на компромисс. Он может быть найден посредством прерывания устройства, пока выполняется подпрограмма обработки прерываний, благодаря измененному приоритету процессора.)

При прогнозировании программы необходимо внимательно исследовать каждый шаг: необходимо иметь в виду, что может случиться, если прерывания будут происходить в каждой директиве.

Необходимо потратить достаточное количество времени для очистки и прямого обслуживания условия ошибки; если программа хорошо обслуживает условия ошибки, то в нормальном состоянии программа работает так же хорошо.

4.4.3. Написание кодов

Можно заимствовать коды из других подпрограмм обработки прерываний. Необходимо начать с общих частей, затем добавить детали для подгонки к специфическому устройству. Когда коды написаны, необходимо проверить логику, протранслировать, проверить и отладить.

4.4.4. Проверка и отладка

Проверка подпрограммы с внутренней обработкой прерываний — попытка выполнить ее. Если подпрограмма составлена правильно, она будет способна считывать и записывать данные точно, без потери некоторых данных и будет правильно обслуживать условия ошибки. Если обнаружились ошибки, необходимо связать подпрограмму с ODT (не VDT) и попробовать выполнить ее шаг за шагом. Сделать корректировку подпрограммы, снова оттранслировать и снова выполнить ее, если необходимо.

4.5. Структура подпрограммы обработки прерываний

Этот подраздел дает набросок общей структуры внутренней подпрограммы обработки прерываний. Пользователь должен выбрать из нее лишь то, что применимо к его случаю.

4.5.1. Защита векторов: запрос .PROTECT

В системах FB и XM, где выполняется более одного задания, следует использовать программный запрос .PROTECT для защиты векторов прерывания перед засылкой в них значений. Это гарантирует, что вектор уже не будет относиться к мониторному или другому заданию, обеспечивает пользовательскому заданию собственность вектора и защищает его от вмешательства другого задания или монитора путем установки разрядов в карте защиты памяти (разд. 3 детально описывает карту защиты нижней памяти). Пользовательскому заданию необходимо немедленно выходить в случае ошибки запроса .PROTECT. Задание не должно иметь доступа к векторам, которые всегда используются. Пример использования запроса .PROTECT приведен в подразделе 4.6. см. в [1] формат программного запроса .PROTECT.

Несмотря на то, что запрос .PROTECT не работает в сис-

теме SJ, его целесообразно использовать в программе. Запрос не оказывает действия, немедленно возвращающего в программу, тем не менее использование его упрощает позже переход, если программа будет выполняться в среде FB.

4.5.2. Установка вектора прерывания

Программа пользователя должна позаботиться о пересылке адреса подпрограммы обработки прерываний в первое слово вектора прерывания. Система ФОДОС-2 требует, чтобы все прерывания поднимали приоритет процессора до 7, поэтому программа пользователя должна заполнить второе слово вектора прерывания новым приоритетом, равным 7.

Пример:

```
XXVEC=220                    ;определяется вектор
 PR7=340                     ;приоритет 7 равен 340
;
;точкой входа для подпрограммы
;обработки прерываний является
;ISREP:
    .PROTECT #AREA,#XXVEC    ;защита вектора
    BCS      NOVEC           ;вектор используется
    MOV      #ISREP,@#XXVEC  ;установка первого слова
    MOV      #PR7,@#XXVEC+2  ;установка второго слова
```

В этом примере показан типичный случай установки двухсловного вектора прерывания. Программе не следует устанавливать вектор, пока он защищен. Здесь XX — имя устройства и 220 и 222 — адреса векторов прерывания.

4.5.3. Чистая остановка: запрос .DEVICE

Программный запрос .DEVICE является многозначительным лишь в системах FB и XM. Его предназначение — замена устройства (посредством очистки его разряда разрешения прерывания), если выполняющаяся программа преждевременно прервана по СУ/С или если программа выполнена. (Формат программного запроса .DEVICE см. в [1], а в подразделе 4.6 — пример использования запроса .DEVICE.)

Этот запрос необязателен в области SJ. Однако, целесообразно его использовать в пользовательской программе. Запрос не предпринимает действий возвращения непосредственно в программу пользователя, однако использование его упрощает позже переход, если программа будет выполняться в области FB.

Если программа выполняется в среде SJ, монитор ожидает окончания любого ввода-вывода, если имеется активная очередь невыполненных элементов. В среде FB, когда программа выполняется, монитор не только ожидает, если имеется ак-

тивная очередь невыполненных элементов, но и входит в точку входа драйвера устройства в том месте, где он был преждевременно прерван. Если задание было преждевременно прервано по СУ/С или если оно вышло на запрос .HRESET, монитор SJ выполняет аппаратную очистку для прекращения ввода-вывода на любом устройстве. Если для данного устройства назначена аппаратура, необходимо убедиться, что произошел чистый останов, когда получен сигнал инициализации канала.

4.5.4. Понижение приоритета процессора: запрос .INTEN

Когда возникает прерывание, управление передается подпрограмме обработки прерываний, адрес которой находится в первом слове вектора прерывания. В этой точке приоритет процессора равен 7, и все остальные прерывания запрещаются. Если требуется запретить некоторые прерывания, это относится к этому коду. Он должен быть краток и эффективен и не должен разрушать содержимое регистров. Если подпрограмма нуждается в использовании регистров, она должна сохранять их содержимое и восстанавливать их перед выходом из запроса .INTEN. Если код выполняется на приоритете 7 слишком долго, произойдет прерывание системы в скрытом виде (мера, которой система быстро отвечает на прерывание). Хорошей будет обработка не более 50 микросекунд на приоритете 7.

Следует понижать приоритет процессора для устройства как только возможно. Это значит, что только устройства с высоким приоритетом будут способны прервать подпрограмму обработки прерываний данного устройства. Для понижения приоритета необходимо использовать запрос .INTEN. Указатель стека и общие регистры R0—R5 должны содержать те же значения, которые уже содержались там до вызова запроса .INTEN, когда подпрограмма обработки прерываний покидает запрос .INTEN. Если подпрограмма обработки прерываний записана не в позиционно-независимом коде (PIC), необходимо использовать следующий формат: .INTEN PRIO

Запрос .INTEN генерируется в следующий код:

```
JSR     R5,@54
.WORD   ^C<PRIO*40>&340
```

Если подпрограмма обработки прерываний использует позиционно-независимый код (PIC), необходимо использовать .INTEN со вторым аргументом PIC. (Аргумент может быть всем чем угодно, только не пробелом.)

```
.INTEN PRIO,PIC
```

Этот запрос генерируется в следующий позиционно-независимый код:

```
MOV     @#54,-(SP)
JSR     R5,@(SP)+
.WORD   ^C<PRIO*40>&340
```

Оба формата вызывают переход к подпрограмме монитора INTEN, которая понижает приоритет процессора и в системах FB и XM переключает режим в системный. Затем монитор вызывает подпрограмму обработки прерываний как соподпрограмму. R4 и R5 можно использовать после выхода из запроса. Недопустимо разрушение содержимого любого другого регистра. Если требуются регистры R0—R3, необходимо сохранить их в стеке или памяти и восстановить перед окончанием. Если требуется сохранить значения в процессе работы запроса .INTEN, необходимо сохранить их в памяти перед вызовом и восстановить после выхода из него. Таким образом, если содержимое PS важное, такое как значения разрядов условий, следует сохранить их перед использованием вызова .INTEN, т. к. .INTEN вызывает переключение системного стека в FB и XM, следует избегать чрезмерного использования стека в подпрограмме обработки прерываний.

Можно хранить и восстанавливать регистры и PS, используя ячейки памяти вместо стека.

Хранение в ячейках памяти должно предохранить подпрограмму прерываний от повторного входа. Если подпрограмма будет использоваться для многих устройств, необходимо позаботиться при планировании о повторном входе.

4.5.5. Использование программного запроса .SYNCH

Запрос .SYNCH полезен в системах FB и XM. Его цель — убедиться, что задание выполнено правильно, когда подпрограмма обработки прерываний выполняет программный запрос. (В системе SJ не работает.)

Если необходимо выдать один или несколько запросов из подпрограммы обработки прерываний, первым должен следовать запрос .SYNCH. Запрос .INTEN передает управление в системный режим, а программный запрос .SYNCH может выполняться только в состоянии пользователя. Запрос .SYNCH сам обслуживает переход назад пользователя. Никогда не следует выдавать из подпрограммы обработки прерываний запросы, требующие USR, даже после использования .SYNCH. Можно также использовать .SYNCH после .FORK. При использовании запроса .SYNCH, R0—R3, SP должны содержать те же значения, какие были после использования запроса .INTEN.

В табл. 5 показан формат SYNCH-блока, который работает подобно элементу очереди завершения. Информация в 7-словном SYNCH-блоке размещена во главе соответствующей очереди завершения заданий. Следовательно, коды, следующие за .SYNCH выполняются как подпрограмма завершения в режиме пользователя с приоритетом 0. Поэтому программа пользователя должна запрещать прерывания перед запросом .SYNCH или она должна быть подготовлена для устройства, чтобы прерывание опять выполнялось перед выполнением .SYNCH. SYNCH-блок доступен для повторного использования, если Q.COMP равно 0. Можно проверить SYNCH-блок путем подачи следующего запроса .SYNCH. Если управление передается к возврату по ошибке (слово, следующее за вызовом .SYNCH), значит, блок по-прежнему используется.

Таблица 5

Смещение	Имя	Владелец	Содержимое
1	2	3	4
0	Q.LINK	——	Зарезервировано
2	Q.CSW	Пользователь	Номер задания
4	Q.BLKN	——	Зарезервировано
6	Q.FUNK	——	Зарезервировано
10	Q.BUFF	Пользователь	R0 — аргумент для перехода
12	Q.WCNT	Монитор	—1
14	Q.COMP	Пользователь	0 при транслировании; монитор затем сохраняет содержимое этого слова

Вообще, может пройти много времени между вызовом .SYNCH и возвратом. Сначала монитор переключается в режим пользователя, и запрашивается план перехода для определения наличия переключателей контекста. Затем подпрограмма завершения фонового задания ожидает вычисления границы блокировки основного задания. Это может занять много времени, прежде чем коды, следующие за .SYNCH, действительно выполнятся.

В коде следующего вызова .SYNCH R0 и R1 свободны для использования, т. к. они имеются в любой подпрограмме завершения. Однако необходимо сохранить R2—R5, если подпрограмма .SYNCH использует их. R4 и R5 не сохраняются при вызове. Поэтому, если их содержимое важно, необходимо

сохранить их в памяти перед вызовом .SYNCH. Можно использовать Q.BUFF в SYNCH-блоке для пересылки значения в R0 для подпрограммы.

Запрос .SYNCH имеет необычный возврат по ошибке. Первое слово после .SYNCH — адрес возврата по ошибке, второе слово после .SYNCH — возврат при успешном завершении.

В среде SJ подпрограммы, следующие за вызовом .SYNCH (фактически подпрограммы завершения), будут вмонтированы (т. е. они могут прерывать друг друга). Они выдаются сериями в FB и XM. В SJ механизм запроса похож на схему FB и XM, но не делает его дважды.

4.5.6. Выполнение на FORK-уровне: запрос .FORK

Программный запрос .FORK выдает другой, более низкий приоритет процессора. Когда используется вызов .FORK, FORK-блок добавляется к FORK-очереди, которая организована FIFO (первым вошел, первым вышел). FORK-подпрограммы (все коды, следующие за запросом .FORK) выполняются в системном режиме с приоритетом 0 после обработки всех прерываний, но перед переходом в режим пользователя. Переключение контекстов запрещено во время выполнения FORK-подпрограмм.

R4 и R5 сохраняются во время вызова .FORK. Кроме того, R0—R3 можно использовать после запроса .FORK. Подобно .SYNCH, запрос .FORK предполагает, что R0—R3 и SP не изменяются после возвращения из запроса .INTEN. Недопустимо использование запроса .FORK без предварительного запроса .INTEN.

Необходимо предоставить 4-словный блок памяти для элемента FORK-очереди, нижние три слова содержат R4,R5 и адрес возврата PC. Первое слово — слово связи, которое должно быть 0, когда выдается запрос .FORK. т. к. подпрограмму .FORK не следует перезапускать, необходимо убедиться, что устройство не может быть прервано между моментом выдачи .FORK и моментом начала выполнения подпрограммы (коды, следующие за вызовом).

Недопустимо повторное использование FORK-блока, пока работает подпрограмма .FORK. Предполагается, что FORK-блок является свободным, когда вызов, который использовал его, возвращается. Табл. 6 иллюстрирует содержимое FORK-блока.

Обычно запрос .FORK используется в драйверах устройств. Чтобы использовать его в подпрограмме обработки прерываний, необходимо сначала установить указатель, называе-

Таблица 6

Смещение	Имя	Владелец	Содержимое
1	2	3	3
0	F.BLNK	Монитор	Слово связи
2	F.BADR	Монитор	Адрес подпрограммы .FORK
4	F.BR5	Монитор	Область хранения R5
6	F.BR4	Монитор	Область хранения R4

мый ¤FKPTR. Рекомендуется сделать это в основной программе следующим образом:

```
        MOV     @#54,R4
        ADD     402(R4),R4
        MOV     R4,¤FKPTR
        .
        .
        .
¤FKPTR: .WORD   0
XXFBLK: .WORD   0,0,0,0
```

Затем в подпрограмме обработки прерываний можно использовать обычную форму запроса .FORK:

```
.FORK XXFBLK
```

Макрорасширение запроса .FORK выглядит следующим образом:

```
JSR     R5,@¤FKPTR
.WORD   XXFBLK—.
```

В системе SJ запрос .FORK не поддерживается, если не выбрана поддержка таймера во время генерации системы. Вместо этого монитор моделирует процесс путем сохранения R0—R3 перед вызовом подпрограммы обработки прерываний. Затем он не пытается выдавать серии FORK-подпрограмм. В подпрограмме обработки прерываний нет свободных для использования регистров перед вызовом .INTEN. После вызова .INTEN можно использовать регистры R4 и R5.

Запрос .FORK имеет различные применения в системах реального времени, т. к. он допускает откладывать обработку растянутых, но не критических по времени прерываний, пока не будут обработаны все другие прерывания.

Понижение приоритета приводит к проблемам в драйверах других устройств, которые будут перезапускаться. Использование .SYNCH не всегда решает проблему, монитор SJ только моделирует .SYNCH и сбрасывает приоритет к 0, который

приводит к тем же проблемам перезапуска драйверов. Монитор FB должен выполнить переключение контекста после возврата из .SYNCH в режим пользователя, выполняемый в стеке *пользователя. Переключение контекста — длительный* процесс и не проходит до конца, если идет завершение основного задания.

Запрос .FORK решает проблему. Он возвращается на приоритет 0, только когда все другие прерывания уже обработаны, но перед передачей управления в прерванную программу пользователя.

4.5.7. Итог по .INTEN, .FORK и .SYNCH

Табл. 7 обобщает действия вызовов .INTEN, .FORK и .SYNCH. Рис. 5 показывает состояние регистров для каждого вызова.

Таблица 7

Макровызов	Новый приоритет	Новый стек	Регистры, доступные для использования после вызова	Данные, сохраняющиеся при вызове
1	2	3	4	5
.INTEN	Устройства	Системный	R4, R5	Нет
.FORK	0	Системный	R0 — R5	R4, R5
.SYNCH	0	Пользователя	R0, R1	R0

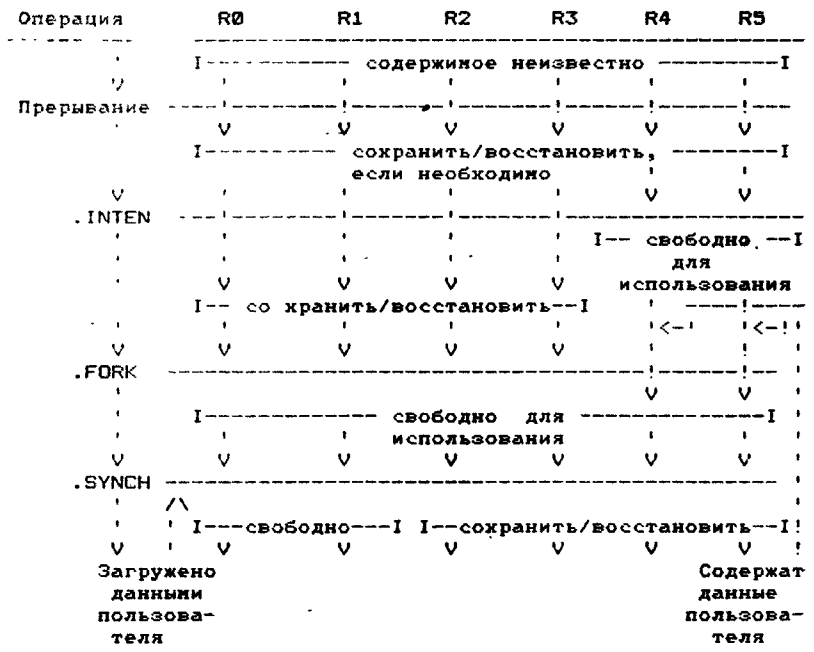
4.5.8. Выход из обработки прерывания: RTS PC

Запрос .INTEN заставляет монитор вызывать подпрограмму обработки прерываний как соподпрограмму. В конце этой подпрограммы, для выхода из нее необходимо использовать инструкцию RTS PC, по которой управление возвращается в монитор, который восстанавливает R4, R5 и выполняет команду RTI.

Аналогично осуществляется выход из подпрограмм .FORK и .SYNCH с помощью команды RTS PC. Стек должен быть таким же, что и перед входом, и регистры должны иметь начальные значения.

4.6. Схематическая конструкция подпрограммы обработки прерываний

В примере, приведенном ниже, показана основная программа, которая содержит внутреннюю подпрограмму обработки прерываний. Эта программа представляет некоторые



задачи инициализации и приостанавливает себя. Когда данные поступают с периферийного устройства, подпрограмма обработки прерываний собирает их. Когда все данные собраны, подпрограмма обработки прерываний продолжает основную программу, которая может потом обрабатывать новую информацию до тех пор, пока снова не приостановит себя. Выполнение основной программы может включать некоторую обработку новых данных или запись данных в файл, используемый фоновым заданием анализа данных. Т. к. этот пример доводит номер задания до 2, он не может нормально выполняться в системе, имеющей признак системного задания.

В примере XX — двухсимвольное имя устройства.

Пример:

** Основная программа **

```
XXVEC=VVV       ;вектор устройства
PR7=340         ;приоритет 7
DEVPRI=5        ;приоритет устройства
                ;равен 5
                ;(0—7, но не 000—340)
```

```
        XXCSR=NNNNNN            ;регистр состояния уст-
                                ;ройства
        IENABLE=100             ;разряд разрешения пре-
                                ;рывания
START:  .PROTECT#LIST,#XXVEC    ;защита вектора
        BCS   ERROR             ;обработка ошибки
                                ;.PROTECT
        MOV   #ISREP,@#XXVEC    ;установка первого сло-
                                ;ва вектора
        MOV   #PR7,@#XXVEC+2    ;установка второго слова
                                ;вектора
        .DEVICE #LIST,#DEVLST   ;блокировать устройство
                                ;от выхода или прежде-
                                ;временного прерывания
;
;строки кода инициализируют входные буферы в подпро-
;грамме обработки;
;инициализируют другие указатели и признаки
;
SPND: BIS #IENABL,@#XXCSR ;разрешение прерывания
      .SPND               ;ожидание появления не-
                          ;которых данных
;
;строки кода хранят данные;
;переустанавливаются некоторые признаки
;
      BR      SPND        ;ожидаются другие дан-
                          ;ные
DEVLST: .WORD XXCSR       ;список для .DEVICE
        .WORD 0
        .WORD 0
LIST:   .BLKW 3           ;блок аргументов EMT
ERROR:  .                 ;подпрограммы для об-
                          ;работки ошибок
        .
        .
          ** Подпрограмма обработки прерываний **
ISPEP:  .                 ;точка входа прерыва-
                          ;ния;
        .                 ;приоритет равен 7
        .
        .INTEN  DEVPRI    ;но не #DEVPRI.
                          ;понижается до приори-
                          ;тета устройства; в дан-
```

```
                                    ; ном случае, в систем-
                                    ; ном режиме, доступны
                                    ; R4 и R5
;
; если есть другие данные для выборки:
;
        BR          RETURN
;
; если нет больше данных для выборки:
;
        .SYNCH  #SYNBLK     ; возврат в основную про-
                            ; грамму для обработки
                            ; данных
        BR          SYNERR  ; возврат по ошибке
                            ; .SYNCH
        .RSUM               ; возобновление основной
                            ; программы
RETURN: RTS         PC      ; ожидание следующего
                            ; прерывания
        .
        .
        .
SYNBLK: .WORD   0,2,0,0,0,—1,0  ; здесь: 2 — номер основ-
                                ; ного задания
SYNERR:                         ; обработка ошибок
                                ; .SYNCH
```

5. ДОПОЛНИТЕЛЬНЫЕ ВОЗМОЖНОСТИ

5.1. Тайм-аут устройства ввода-вывода

Используя дополнительное средство, тайм-аут устройства, драйвер может назначить подпрограмму завершения, которая будет выполняться, если прерывание не возникло в указанный интервал времени. Таким образом, драйвер может выполнять подпрограмму завершения по истечении этого интервала времени без вызова .SYNCH и соответствующей ему потенциальной задержки.

Выбор средства тайм-аута производится во время генерации системы. Тайм-аут используется частично в мультитерминальном мониторе системы ФОДОС-2. Параметр автоматически включен в систему, если выбрана мультитерминальная поддержка. В противном случае, если это средство используется в драйвере, необходимо включить его во время генерации системы.

Для обеспечения тайм-аута устройства в системе ФОДОС-2 предусмотрены два запроса: .TIMIO и .CTIMIO. Они доступны только драйверам устройств. Если при трансляции файла драйвера в условии TIM$IT равен 1, то запрос .DRDEF выдает директиву .MCALL для запросов .TIMIO и .CTIMIO.

5.1.1. Запрос .TIMIO используется в секции инициирования ввода-вывода для формирования вызова тайм-аута. Этот запрос может располагаться в любом месте драйвера, кроме уровня прерывания. Если необходимо выдать запрос на уровне прерывания, то сначала должен следовать запрос .FORK.

Запрос .TIMIO планирует выполнение подпрограммы завершения после истечения указанного интервала времени. Подпрограмма завершения выполняется в контексте задания, указанного в блоке таймера. В XM-системе подпрограмма завершения выполняется с внутренним отображением, так как она является частью подпрограммы обработки прерываний. Как обычно, R0 и R1 используются в подпрограммах завершения. Когда вводится подпрограмма завершения, R0 содержит последовательное число запросов тайм-аута.

Так как для выдачи запроса .TIMIO или .CTIMIO необходимо переходить к FORK-уровню (нулевому приоритету процессора), драйвер должен запретить прерывание устройства перед использованием .FORK, или должен быть тщательно закодирован во избежание проблемы повторного ввода. Запрещено повторно использовать блок таймера, пока не исчезнет элемент таймера и не будет введена подпрограмма завершения, или пока элемент таймера не будет успешно устранен.

Формат запроса .TIMIO следующий: .TIMIO TBK,HI,LO

где TBK — адрес блока таймера, семисловного псевдоэлемента очереди таймера; нельзя использовать номерной знак (#) перед TBK;

HI — константа, обозначающая старшее слово двухсловного интервала времени;

LO — константа, обозначающая младшее слово двухсловного интервала времени.

Формат блока таймера показан в табл. 8.

Хотя запрос .TIMIO перемещает старшее и младшее слова времени в блок таймера, необходимо обозначать их правильно при макровызове. Интервал времени выражается в тиках. Имеется 50 десятичных тиков.

Младшее слово времени вмещает значение до 65535 тиков. Это эквивалентно приблизительно 1310 с или приблизительно 21,8 мин. Если необходимо указать интервал времени в 21,8

Таблица 8

Смещение	Имя	Владелец	Значение
1	2	3	4
0	C.HOT	.TIMIO	Старшее слово времени
2	C.LOT	.TIMIO	Младшее слово времени
4	C.LINK	Монитор	Связь со следующим элементом очереди; 0 показывает отсутствие элемента
6	C.JNUM	Пользов.	Номер задания владельца; передается из элемента очереди
10	C.SEQ	Пользов.	Номер запроса таймера. Допустимый диапазон использования номеров от 177000 до 177377
12	C.SYS	Монитор	—1
14	C.COMP	Пользов.	Адрес подпрограммы завершения, которая будет выполняться, если возникнет тайм-аут. Монитор обнуляет это слово, когда вызывает подпрограмму завершения, показывая, что блок таймера имеется в наличии для повторного использования

минутах и менее, то аргумент HI должен содержать 0, и аргумент LO в запросе .TIMIO должен содержать число тиков.

Если необходимо указать интервал времени продолжительнее, чем 21,8 мин, то старшее слово должно быть представлено в виде слова переноса. Каждый интервал, продолжительностью 21,8 мин, вызывает перенос 1 в старшее слово. Поэтому, для обозначения интервала, чуть большего, чем 21,8 мин, необходимо определять 1 для аргумента HI и 0 для аргумента LO. Для обозначения 43,6 мин в аргумент HI перемещается 2, 0 — в аргумент LO и т. д. так как двухсловное время позволяет показать до 65565 единиц, каждая длиной в 21,8 мин, то самый большой интервал, который можно указать, равен 2,7 года.

Единственными информационными словами, которые необходимо установить в блоке таймера, являются номер задания, последовательный номер и адрес подпрограммы завершения. Номер задания получается из текущего элемента очереди и, затем, помещается в блок таймера. Необходимо назначить последовательный номер, начиная с 177000, и можно работать до самого высшего номера последовательности, 177377. Номер задания и последовательный номер передаются подпрограммой завершения, когда она вводится. Адрес подпрограммы завершения необходимо переместить к седьмому слову блока таймера.

Ниже приведено макрорасширение запроса .TIMIO:

```
.TIMIO      TBK,HI,LO
JSR         R5,@ $TIMIT   ; указатель конца драйвера
.WORD       TBK—.
.WORD        0            ; код для .TIMIO
.WORD       HI            ; старшее слово интервала вре-
                          ; мени
.WORD       LO            ; младшее слово интервала
                          ; времени
```

5.1.2. Запрос .CTIMIO

После возникновения условия, которое ожидал драйвер, необходимо выдать вызов устранения тайм-аута, который запрещает подпрограмму завершения. Вызов .CTIMIO используется в драйвере для устранения запроса тайм-аута. Выполнение должно происходить в системном режиме, когда выдается вызов. Если .CTIMIO используется на уровне прерывания, то сначала выдается вызов .FORK.

Например, драйвер построчно-печатающего устройства может проверять на условие автономности. Когда программа запрашивает передачу ввода-вывода, секция инициирования ввода-вывода драйвера вызывает немедленное прерывание. Затем секция обработки прерываний драйвера проверяет разряд ошибки устройства. Если разряд установлен, построчно-печатающее устройство неавтономно, то драйвер печатает сообщение, устанавливает посредством .TIMIO двухминутный таймер и возвращается в монитор по команде RTS PC для ожидания другого прерывания. Устройство не должно прерываться снова, пока условие ошибки не будет зафиксировано оператором. Если за две минуты не возникло прерывание, подпрограмма завершения таймера печатает другое сообщение об ошибке, устанавливает другой двухминутный таймер и возвращается опять в монитор по RTS PC для ожидания прерывания (пример драйвера построчно-печатающего устройства см. п. 5.1.3.3).

В этом примере, когда возникает прерывание и разряд ошибки чист, драйвер выдает вызов .CTIMIO для устранения временного ожидания.

В виде другого примера, драйвер диска может установить таймер до того, как начнет операцию поиска. Т. к. прерывания ищутся дважды, драйвер не должен устранять таймер после первого прерывания. Когда возникает второе прерывание, хотя поиск уже завершен, драйвер должен устранить таймер.

Если интервал времени в каком-то применении уже истек, и устройство поэтому находится в тайм-ауте, выполнение запроса .CTIMIO не удается. Так как подпрограмма завершения была помещена в очередь, вызов .CTIMIO возвращается с установленным разрядом переноса. Обычно можно игнорировать это условие.

Формат вызова .CTIMIO следующий: .CTIMIO TBK где TBK — адрес семисловного блока таймера; блок, указанный в вызове .CTIMIO, должен быть таким же, как уже использованный соответствующим запросом .TIMIO.

Запрос .CTIMIO расширяется следующим образом:

```
.CTIMIO
JSR         R5,@ $TIMIT   ; указатель конца драйвера
.WORD       TBK—.
.WORD       1             ; код для .CTIMIO
```

Если задание пользователя преждевременно прервется и драйвер введется в точке входа преждевременного прерывания, то необходимо сразу же устранить любые важные запросы таймера. Однако если подпрограмма завершения таймера уже была введена, необходимо дождаться ее выполнения.

5.1.3. Применение тайм-аута устройства

Подержка тайм-аута устройства используется в системе ФОДОС-2 лишь в нескольких случаях. Однако в ряде условий применяются запросы таймера.

5.1.3.1. Мультитерминальная обработка в системе ФОДОС-2 использует тайм-аут устройства для проверки состояния удаленных терминалов. Эта проверка осуществляется с помощью специальной подпрограммы, которая проверяет готовность каждого удаленного терминала и использует запрос .TIMIO.

5.1.3.2. Обычная процедура таймера для драйвера диска

Драйвер диска может выполнить любую процедуру таймера для любой операции диска. Целью подпрограммы таймера является аннулирование или восстановление любой операции, которая длится слишком долго. Если операция не заканчивается за отведенный промежуток времени, то, скорее всего, на диске ошибочно разрушены некоторые операции.

Секция инициирования ввода-вывода устанавливает таймер, используя запрос .TIMIO. Затем драйвер начинает операции, запрошенные заданием: операции считывания, записи или установки. Драйвер возвращается в монитор по команде RTS PC и ожидает прерывания устройства. Если прерывание возникает до истечения интервала времени, драйвер аннулирует таймер и выполняет обычную последовательность про-

верки ошибок в результате передачи. Вообще, драйвер или снижается до FORK-уровня для восстановления неправильной операции, или выводится в монитор посредством .DRFIN для удаления текущего элемента очереди.

Если прерывание не возникло за указанный промежуток времени, начинает выполняться подпрограмма завершения таймера. Ее первым действием должно быть моделирование прерывания. Это действие дублирует среду драйвера после этого прерывания и гарантирует, что стек имеет необходимую информацию. Затем подпрограмма завершения таймера действует так, как если бы устройство было прервано, но передача была ошибочной. Подпрограмма завершения таймера просто переходит в правильную секцию кодов в секции обработки прерываний драйвера устройства для окончания процесса.

Подпрограмма завершения таймера должна использовать следующие команды для моделирования прерывания и ввода системного режима:

```
MOV     @SP,—(SP)   ; отводит место для стека
CLR     2(SP)       ; имитировать прерывание
                    ; PS=0
.MTPS   #340        ; переход на приоритет 7
.INTEN  0,PIC       ; ввод системного режима
```

После входа драйвера в системный режим, он предпринимает соответствующее действие как результат тайм-аута. Драйвер снова может пытаться выполнить операцию. Для этого он уменьшает счетчик повторений, снижается до FORK-уровня и переходит в секцию инициирования ввода-вывода. Код в секции инициирования ввода-вывода устанавливает другой таймер, повторяет передачу и возвращается в монитор по команде RTS PC для ожидания следующего прерывания.

Если драйвер решает, что тайм-аут показывает серьезную ошибку, одну из тех, которые не должны повторяться вновь, то для передачи может последовать та же самая процедура, которая происходит, когда счетчик повторений исчерпан. В этом случае драйвер устанавливает разряд невосстановимой ошибки в слове состояния канала, а затем вводится в монитор вызовом .DRFIN для удаления текущего элемента очереди. Перед тем, как драйвер перейдет к удалению текущего элемента очереди с помощью .DRFIN, он должен аннулировать любой запрос таймера, который еще не закончен.

5.1.3.3. Пример драйвера построчно-печатающего устройства

Ниже приведен пример, состоящий из выдержек, взятых из драйвера построчно-печатающего устройства системы ФОДОС-2, модифицированного для использования поддержки таймера для проверки условия автономности устройства.

Когда секция инициирования ввода-вывода драйвера начинает передачу, это приводит к немедленному прерыванию и заставляет секцию обработки прерываний выполнять проверку разряда ошибки в CSR. Если есть ошибка, управление переходит к подпрограмме OFFLIN, которая выдает вызов .SYNCH для входа в режим пользователя, печатает сообщение об ошибке, затем устанавливает двухминутный таймер, после чего драйвер возвращается в монитор по команде RTS PC и ожидает прерывание устройства.

Если устройство было прервано, это означает, что условие ошибки было исправлено оператором. Драйвер аннулирует таймер и проверяет снова разряд ошибки, чтобы убедиться, что проблемы отсутствуют. Если ошибок нет, то драйвер обрабатывается как обычно. Если ошибка есть, драйвер циклится в подпрограмме OFFLIN. Если прерывание не возникнет за 2 минуты, то начинает выполняться подпрограмма завершения таймера. Она печатает сообщение об ошибке, устанавливает другой двухминутный таймер и возвращается в монитор по команде RTS PC для ожидания прерывания.

Пример:

```
; секция инициирования ввода-вывода
        .DRBEG  LP
        MOV     LPCQE,R4    ; R4 указывает текущий эле-
                            ; мент очереди
        ASL     6(R4)       ; переход от счетчика слов к
                            ; счетчику байтов
        BCC     LPERR       ; запрос считывания недопус-
                            ; тим
        BEQ     LPDONE      ; переход на немедленное окон-
                            ; чание
RET:    BIS     #100,@LPS   ; разрешение прерывания, на-
                            ; чало передачи
        RTS     PC
; секция обработки прерываний
.ENABL  LSB
        .DRAST  LP,4,LPDONE
        CLR     @LPS        ; запрещение прерываний
        .FORK   FRKBLK
        TST     TICMPL      ; активный элемент таймера?
        BEQ     1$          ; нет
```

```
          .CTIMIO TIMBLK  ;да, удалить его
          BCS     1$      ;ошибка
          CLR     TICMPL  ;и не устанавливать его снова
1$:       MOV     LPCQE,R4; R4 указывает текущий эле-
                          ;мент очереди
          TST     @(PC)+  ;ошибочное условие?
LPS:      .WORD   LP$CSR  ;регистр состояния построч-
                          ;но-печатающего устройства
ERROPT:   BMI     OFFLIN  ;да, идти на исправление
          .
          .
          .
;секция завершения ввода-вывода
LPDONE:   CLR     @LPS    ;возврат из прерывания
          .DRFIN  LP
          .
          .
          .
;построчно-печатающее устройство автономно, предупрежде-
;ние печатать каждые две минуты
OFFLIN:   MOV     LPCQE,R5        ;указывает элемент оче-
                                  ;реди
          MOVB    Q$JNUM(R5),R5   ;устанавливает номер те-
                                  ;кущего задания
          ASR     R5              ;сдвинуть его
          ASR     R5              ;вправо
          ASR     R5              ;на три разряда
          BIC     #^C<16>,R5      ;выделяет номер задания
          MOV     R5,SYJNUM       ;сохраняет его для
                                  ;.SYNCH
          MOV     R5,TIJNUM       ;сохраняет его для
                                  ;.TIMIO
          .SYNCH  SYNBLK,PIC      ;переход в режим поль-
                                  ;зователя
          RTS     PC              ;синхронизация законче-
                                  ;на, продвинуться
1$:       CLR     TICMPL          ;показывает, что мы по-
                                  ;пали сюда
          TST     @LPS            ;ошибка все еще есть?
          BPL     2$              ;нет, закончить
          MOV     PC,R0           ;в случае подпрограммы
                                  ;завершения печатает со-
                                  ;общение
```

```
        ADD     #MESSAG—.,R0 ; указатель сообщения
                             ; как PIC
        .PRINT               ; печатать его
        MOV     PC,R0        ; в PIC-кодах, указывая
                             ; для
        ADD     #1$—.,R0     ; .TIMIO подпрограмму
                             ; завершения ,
        MOV     R0,TICMPL    ; запоминает его
        .TIMIO  TIMBLK,0,2*60.*60. ;устанавливает двухминутный
                             ;таймер
2$:     RTS     PC           ; возврат назад
FRKBLK: .WORD   0,0,0,0      ; FORK-блок
TIMBLK: .WORD   0            ; блок таймера: старшее
                             ; слово времени
        .WORD   0            ; младшее слово времени
        .WORD   0            ; связь со следующим эле-
                             ; ментом очереди
TIJNUM: .WORD   0            ; номер задания
        .WORD   177000+3     ; последовательный номер
        .WORD   0            ; монитор устанавливает
                             ; здесь —1
TICMPL: .WORD   0            ; адрес подпрограммы за-
                             ; вершения
SYNBLK: .WORD   0            ; SYNCH-блок
SYJNUM: .WORD   0            ; номер задания
        .WORD   0,0,0,—1,0   ; остальные
MESSAG: .ASCIZ «?LP—W—LP OFF LINE—PLEASE CORRECT»
        .EVEN
        .DREND  LP
```

В этом примере приведен драйвер построчно-печатающего устройства.

5.2. Регистрация ошибок — это дополнительное средство системы ФОДОС-2, предназначенное для обеспечения надежности системы. Драйверы устройств, которые поддерживают регистрацию ошибок, вызывают регистратор ошибок после каждой передачи ввода-вывода. Регистратор ошибок собирает статистику о деятельности устройств ввода-вывода, которую можно использовать для проверки их надежности.

Поддержка регистрации ошибок выбирается в процессе генерации системы. Регистрация ошибок может выполняться под управлением мониторов XM или FB. Если система способна выполнять системное задание, регистратор ошибок выполняется как системное задание; в противном случае, он выполняется как обычное основное задание. Содержимое

файла параметров генерации системы для регистрации ошибок следующее:

ERL$G — *если это значение равно 1, то регистрация ошибок* разрешена для этой системы;

ERL$S — это условие определяет число 256-словных блоков, используемых под внутренний буфер регистрации в мониторе SJ;

ERL$U — максимальное количество индивидуальных приводов устройств, для которых регистратор ошибок собирает статистику; значение по умолчанию равно 10, максимальное число приводов равно 30; каждый привод добавляет семь слов в регистратор ошибок; для каждого блока требуется одна позиция (например, для системы с кассетными дисками требуется две позиции); значение этой переменной устанавливается при ответе на вопрос во время генерации системы.

Необходимо пересмотреть требования, предъявляемые к памяти и времени, до использования регистратора ошибок, так как регистратор ошибок создает определенное минимальное количество дополнительных глав для каждой передачи ввода-вывода, и сам регистратор ошибок использует почти 2К слов памяти. Однако регистратор ошибок может выполняться не каждый раз, чтобы требующаяся ему память была доступна программам пользователя и вызовы драйвером регистратора ошибок возвращались бы немедленно. Самый удобный способ использования регистрации ошибок системы — это использование ее в виде проверки, когда есть сомнения в надежности системы.

Все коды драйвера, которые предназначены только для регистрации ошибок, должны быть размещены среди команд условной трансляции. Эти команды включают код регистрации ошибок, если символ ERL$G=1, и опускают его в противном случае. Таким образом, коды регистрации ошибок или включаются в драйвер при его транслировании, или нет.

5.2.1. Когда и как вызывать регистратор ошибок

Драйвер устройства вызывает регистратор ошибок после каждой передачи ввода-вывода, независимо от того, была передача успешной или нет. Если передача была ошибочной, драйвер вызывает регистратор ошибок один раз для каждой повторной попытки передачи и еще раз, когда разрешенное количество повторных попыток уже исчерпано. Так как вызовы регистратора ошибок должны быть упорядочены, драй-

вер может вызывать его только во время инициирования ввода-вывода или следом за вызовом запроса .FORK.

Драйвер должен установить регистры вызова регистратора ошибок.

5.2.1.1. Регистрация успешной передачи

Регистры R4 и R5 устанавливаются перед вызовом регистратора ошибок после каждой успешной передачи следующим образом:

R5 — должен указывать на третье слово текущего элемента очереди;

R4 — содержит два байта информации: старший байт является байтом идентификации устройства, DD$COD, младший байт равен —1.

5.2.1.2. Регистрация невосстановимой ошибки

Перед вызовом регистратора ошибок необходимо установить регистры с R2 по R5 после возникновения невосстановимой ошибки. Примерами невосстановимых ошибок являются: неавтономность устройства, блокировка устройства записи, отсутствие ленты в перфоленточном устройстве ввода и т. д. Восстановимая ошибка, которая исчерпала число повторных передач, рассматривается как невосстановимая ошибка.

R5 — должен указывать на третье слово текущего элемента очереди;

R4 — содержит два байта информации: старший байт является байтом идентификации устройства, DD$COD; младший байт является 0;

R3 — содержит два байта информации: старший байт содержит общее число повторных попыток, разрешенных для этой передачи; младший байт содержит число регистров устройства, содержимое которых должно появляться в протоколе ошибок;

R2 — указывает на буфер в драйвере, который содержит регистры устройства для регистрации.

5.2.1.3. Регистрация восстановимой ошибки

Перед вызовом регистратора ошибок необходимо установить регистры R2—R5 после возникновения восстановимой ошибки. Обычно восстановимые ошибки могут быть исправлены посредством повторной попытки передачи. Примерами восстановимых ошибок являются: ошибки времени выполнения, аппаратные ошибки считывания или записи.

Необходимо установить в драйвере счетчик общего числа повторных попыток, разрешенных для каждой передачи. Счетчик должен уменьшаться всякий раз, когда выполняется повторная передача для восстановимой ошибки. Когда счетчик

достигает 0, регистратор ошибок рассматривает эту ошибку как невосстановимую. При восстановимой ошибке сообщение об ошибке печатается отдельной записью для каждого повтора данной передачи.

Сообщения обо всех повторных передачах печатаются даже в том случае, когда регистры были идентичны. Сообщение не делает различий между восстановимой и невосстановимой ошибками. Оно печатает только содержимое регистров во время ошибки и значение счетчика повторных передач. Невосстановимую ошибку можно узнать лишь на выходе, так как она будет появляться со счетчиком повторных передач, равным 0.

R5 — должен указывать на третье слово текущего элемента очереди;

R4 — содержит два байта информации: старший байт является байтом идентификации устройства, DD$COD; младший байт является текущим значением счетчика повторных передач;

R3 — содержит два байта информации: старший байт содержит общее число повторных попыток, разрешенных для этой передачи; младший байт содержит число регистров устройства, содержимое которых должно появляться в сообщении об ошибке;

R2 — указывает буфер в драйвере, который содержит регистры устройства для регистрации.

5.2.1.4. Различия между восстановимой (SOFT) и невосстановимой (HARD) ошибками

Сам регистратор ошибок не различает восстановимые и невосстановимые ошибки и записывает одну и ту же информацию в обоих случаях. Однако посредством проверки сообщения можно определить невосстановимую ошибку, потому что в этом случае передача исчерпает все свои повторные попытки и будет иметь записи для каждой из этих повторных попыток, в том числе и ту, для которой счетчик повторных передач равен 0.

В случае, когда ошибки могут быть исправлены пользователем, такие, как автономное устройство или ограниченная запись, регистратор ошибок может не вызываться. Обычно сообщается лишь об аппаратных ошибках диска или ленты, так как эти ошибки рассматриваются с точки зрения надежности устройств.

5.2.1.5. Вызов регистратора ошибок

После того, как необходимые регистры будут установле-

ны, можно вызвать регистратор ошибок: JSR PC,@ $ELPTR где $ELPTR — его указатель в резидентном мониторе.

Запрос .DREND отводит место в драйвере для этого указателя. Указатель заполняется во время загрузки (для системного устройства) или по запросу .FETCH или по команде монитора LOAD (для устройства данных). Если регистратор ошибок не работает, то монитор возвращается непосредственно к драйверу. Если регистратор ошибок работает, то слово связи в RMON содержит его точку входа.

Пример:

```
$ERLOG: MOV (PC)+,-(SP) ; входит здесь из драйвера, по-
                        ; мещая следующее слово в
                        ; стек
$ELHND::  .WORD  0      ; 0, если регистратор ошибок
                        ; не работает; иначе, содержит
                        ; точку входа регистратора
                        ; ошибок
          BNE    1$     ; переход, если загружен
          TST    (SP)+  ; очистка стека
1$:       RTS    PC     ; вызов регистратора ошибок
                        ; или возврат к драйверу
```

Приведенные выше строки кода из RMON показывают, как выполняется вызов регистратора ошибок.

Команды монитора SRUN или FRUN устанавливают точку входа регистратора ошибок; команда UNLOAD EL обнуляет $ELHND.

При возвращении из вызова регистратора ошибок, регистры R0—R3 восстанавл-ся в драйвере, R4 и R5 — разрушены.

5.3. Специальные функции

Иногда драйверам необходимо выполнять действия указанных устройств, для которых нет соответствующих программных запросов ФОДОС-2. Примеры таких действий: перемотка магнитных лент и считывание или запись абсолютных секторов на гибких дисках. Программный запрос .SPFUN обеспечивает средства программам для инициации таких специальных функций. Когда программа выдает запрос .SPFUN, должен присутствовать и код специальной функции в виде одного из аргументов. Этот код сообщает драйверу, какие специальные функции он должен выполнить. Например, кодом, который приказывает драйверу MT выполнять автономную перемотку магнитной ленты, является 372.

5.3.1. Программный запрос .SPFUN

Формат программного запроса .SPFUN следующий:

.SPFUN AREA,CHAN,FUNC,BUF,WCNT,BLK[,CRTN]
Полное описание аргументов для .SPFUN см. в[1].

Для использования вызова специальных функций в драйвере устройства необходимо определить интерфейс между программным запросом и драйвером устройства. Таким образом, значения аргументов BUF, WCNT и BLK зависят от конкретных специальных функций, которые вызывают запрос. Если запрос вызывает передачу данных, аргументы имеют свои обычные значения. Однако, хотя монитор осуществляет проверку, чтобы BUF являлся действительным адресом в области задания, он не гарантирует, что BUF+WCNT еще находится в области задания. Поэтому необходимо указывать допустимые значения, если используется запрос .SPFUN для передачи данных.

Если вызов специальной функции заключается в возврате одного значения, то BUF должен быть однословной буферной областью. Можно изменять WCNT и BLK по своему усмотрению. Они могут быть спецификациями одного слова, указателями больших буферов и т. д., так как драйвер интерпретирует их в соответствии с кодом специальной функции. Монитор не изменяет их, когда передает в драйвер. Например, он не изменяет счетчик слов из положительного в отрицательный.

5.3.2. Поддержка специальных функций в драйвере устройства

Для осуществления поддержки вызова специальных функций в драйвере устройства необходимо задать SPFUN$, один из разрядов значения STAT, которое предусмотрено для запроса .DRDEF. Это значит, что драйвер допускает использование специальных функций.

Затем определяется символика в драйвере для представления типов специальных функций, которые может выполнить драйвер. Например, драйвер устройства гибких дисков с двойной плотностью записи DY определяет следующие коды специальных функций:

```
SIZ$FN  =373    ;устанавливает размер устройства
WDD$FN =375     ;записывает метку удаленных данных
WRT$FN  =376    ;записывает указанный сектор
RED$FN  =377    ;считывает указанный сектор
```

Все коды специальных функций должны быть отрицательными значениями байта (т. е. в диапазоне восьмеричных чисел от 200 до 377); рекомендуется, чтобы определенный код специальной функции представлял одну и ту же операцию на

всех устройствах. Поэтому необходимо ознакомиться со всеми уже существующими кодами специальных функций (см. [1]), чтобы убедиться, есть ли соответствующий данной специальной функции код или он отсутствует. Если код уже существует, то его смело можно использовать, если же таковой отсутствует, то необходимо присвоить этой функции любой свободный код, начиная с 200 в сторону кода 377. Это поможет избежать в будущем конфликтов ввиду появления новых кодов следующих версий системы.

Когда драйвер вводится для передачи ввода-вывода, он должен проверить четвертое слово элемента очереди, чтобы убедиться, является ли оно запросом специальной функции. Q.FUNC, являющийся младшим байтом четвертого слова элемента очереди ввода-вывода, содержит код специальной функции. В стандартных запросах ввода-вывода для операции считывания, записи или установки этот байт равен 0. Для вызовов специальной функции этот байт равен отрицательному значению кода специальной функции. Необходимо проверять, чтобы этот код был допустимым для данного устройства, в противном случае, необходимо уходить на невосстановимую ошибку.

Если это запрос специальной функции, драйвер должен инициировать эту функцию и возвратиться по команде RTS PC. В секции обработки прерываний драйвер должен проверить, как обычно, наличие ошибок и определить завершение операции. Драйвер возвращает или данные, или слово, информирующие о состоянии, в буфер пользователя вызванной программы.

Так осуществляется выполнение специальных функций для определенного устройства, можно установить правило вызова для этой функции в программном запросе .SPFUN, а также правило возврата из драйвера. Драйвер должен рассматривать аргументы соответственно для каждого различного вызова специальной функции.

5.3.3. Тома переменного размера

Драйвер может управлять устройством, которое допускает использование томов двух (или больше) различных размеров. Примером такого драйвера является драйвер DY, который может обслуживать гибкие диски с одинарной или с двойной плотностью записи на одном приводе устройства.

Драйвер устройства, которое поддерживает тома различных размеров, должен передавать размер в блоках самого маленького тома в параметр SIZE запроса .DRDEF. Это зна-

чение возвращается в выполняющуюся программу, когда она выдает программный запрос .DSTATUS.

Если выполняющейся программе важно знать размер установленного в данный момент тома, программа может сделать вызов .SPFUN. Драйвер должен быть в состоянии дать ответ на запрос посредством возвращения фактического размера тома в однословную буферную область. Драйвер также должен осуществлять поддержку специальных функций, как описано выше. Стандартный код специальной функции для определения фактического размера тома равен 373.

5.3.4. Устройства со специальными справочниками

Монитор ФОДОС-2 может сопрягаться с устройствами файловой структуры, имеющими нестандартные (т. е. не системы ФОДОС-2) справочники. Примером такого устройства является магнитная лента. Ее драйвер устанавливает разряд 12 (SPECL$) слова состояния устройства. USR обрабатывает операции справочника для устройств справочной структуры ФОДОС-2; для специальных устройств драйвер должен обрабатывать операции справочника, как .LOOKUP, .ENTER, .CLOSE и .DELETE, так же, как и передачу данных.

Монитор запрашивает специальную операцию справочника посредством засылки положительного ненулевого значения в байт кода функции элемента очереди. Положительные коды функций являются стандартными для всех устройств. Они приведены ниже:

Код:	1	2	3	4
функция:	закрытие CLOSE	удаление DELETE	просмотр LOOKUP	ввод ENTER

Эти функции соответствуют программным запросам .CLOSE, .DELETE, .LOOKUP и .ENTER (см. [1]). Запрос .RENAME не поддерживается для специальных устройств.

Пятое слово в элементе очереди для специальной функции справочника содержит указатель блока определителя файла, содержащего имя устройства, имя и тип файла в RADIX-50.

Программные ошибки (такие, как «файл не найден» или «справочник заполнен»), возникающие в драйвере специального устройства во время операций справочника, возвращаются в монитор. Код каждой ошибки выбирается для определенного типа ошибки. Код ошибки возвращается посредством засылки его в SPUSR (ошибка USR специального устройства), расположенного у фиксированного смещения 272 от начала резидентного монитора. Аппаратные ошибки возвраща-

ются обычным способом посредством установки нулевого разряда в слове состояния канала (второе слово элемента очереди).

Программные запросы для операций справочника специальных устройств обрабатываются стандартными программными запросами. Когда, например, выдается запрос .LOOKUP, монитор проверяет разряд специального устройства в слове состояния устройства. Если устройство имеет специальную справочную структуру, то соответствующий код функции вводится в элемент очереди, и элемент непосредственно устанавливается в очередь драйвера, обходя обработку посредством USR. Независимость устройства сохраняется, так как операции .LOOKUP, .ENTER, .CLOSE и .DELETE очевидны для пользователя.

Для специальных устройств запрос .LOOKUP возвращает размер файла в шестое слово элемента очереди (Q.WCNT). Для запроса .ENTER шестое слово возвращает размер нового файла.

5.4. Драйверы устройств в XM-системе

Драйверы устройств в системах SJ и FB требуют нескольких изменений для правильной работы в системе XM. Предварительно необходимо ознакомиться с условными обозначениями в системе XM.

5.4.1. Условные обозначения наименований и условные обозначения системы

При написании драйвера устройства пишется общий исходный файл с именем DD.MAC, где DD — это двухсимвольное имя устройства, включающее код, который обеспечивает поддержку расширенной памяти в директивах условной трансляции. Обозначением условия, которое обеспечивает поддержку расширенной памяти при генерации системы, является MMG$T, которое имеет значение 0, если поддержка расширенной памяти не выбрана, или 1, если выбрана поддержка расширенной памяти.

Это означает, что код расширенной памяти транслируется только тогда, когда значение, условно обозначенное MMG$T, равно 1. Далее необходимо протранслировать исходный файл DD.MAC с условным файлом системы, SYCND, и с XM.MAC, создавая DDX.OBJ для XM-системы или DD.OBJ для систем SJ или FB. Эта процедура гарантирует, что дополнительные средства системы, которые поддерживают драйвер, согласуются с дополнительными средствами данного монитора.

5.4.2. XM-среда

В системе XM драйверы должны размещаться в пределах

младших 28К слов физической памяти. Может быть еще одно ограничение, в зависимости от наличия или отсутствия поддержки .FETCH в XM-мониторе.

Если XM-монитор осуществляет поддержку .FETCH, драйвер не может размещаться в пределах области физической памяти, отображенной посредством PAC1 (регистр адреса страницы 1), областью, которая включает ячейки памяти от 20000 до 37776. Перед вызовом программ, использующих драйверы необходимо сделать их резидентными по команде монитора LOAD, которая приводит к ограничению адреса.

Если XM-монитор поддерживает .FETCH (определяется во время генерации системы), ограничение распространяется только на размещение драйвера в памяти, он не должен превышать 28К слов. При поддержке .FETCH нет надобности в загрузке драйверов XM перед вызовом программ. Однако, даже при поддержке .FETCH, тот же самый драйвер может быть загружен по команде монитора LOAD.

Когда драйверы введены, они выполняются с внутренним отображением, которое разрешает доступ к младшим 28К словам памяти плюс страница устройства ввода-вывода (см. раздел 6). Однако программе, запрашивающей передачу ввода-вывода, нет необходимости иметь такое же отображение, что и внутреннее. Фактически программа может попадать в одну из трех возможных категорий:

— привилегированное задание, отображение которого аналогично внутреннему;

— привилегированное задание, которое отображается в адреса физической памяти свыше 28К слов;

— виртуальное задание с любым видом отображения.

Основную трудность для драйверов в системе XM представляет связь с буфером данных пользователя. Эта трудность возникает оттого, что программа, запрашивающая передачу ввода-вывода, выдает 16-разрядный виртуальный адрес буфера в программном запросе, несмотря на то, что часть области виртуальной адресации пользователя может быть отображена где-либо еще в физической памяти. Поэтому драйвер должен найти фактический 18- или 22-разрядный физический адрес буфера данных пользователя перед перемещением информации в него и из него. Монитор следит за тем, чтобы буферная область памяти занимала смежные ячейки в физической памяти.

Дело в том, что в системе XM ячейки физической памяти выражены в виде 18- или 22-разрядных адресов, это важно, когда необходимо обозначить адрес в пределах самого драй-

вера в виде адреса буфера. Если, например, драйвер содержит строку нулей, которую он записывает в устройство как часть инициализации, драйвер устанавливает операцию записи устройства, обозначая адрес строки в драйвере как адрес буфера. Так как драйвер размещается в пределах младших 28К слов физической памяти, его физический адрес может быть выражен как его виртуальный 16-разрядный адрес плюс дополнительные разряды для ХМ (разряды 16 и 17 для 18-разрядного адреса или разряды с 16 по 21 для 22-разрядного адреса), которые должны быть нулевыми.

На рис. 6 показана ХМ-система. Программа, которая запрашивает передачу ввода-вывода, отобразила область буфера данных в физическую память выше границы 28К слов.

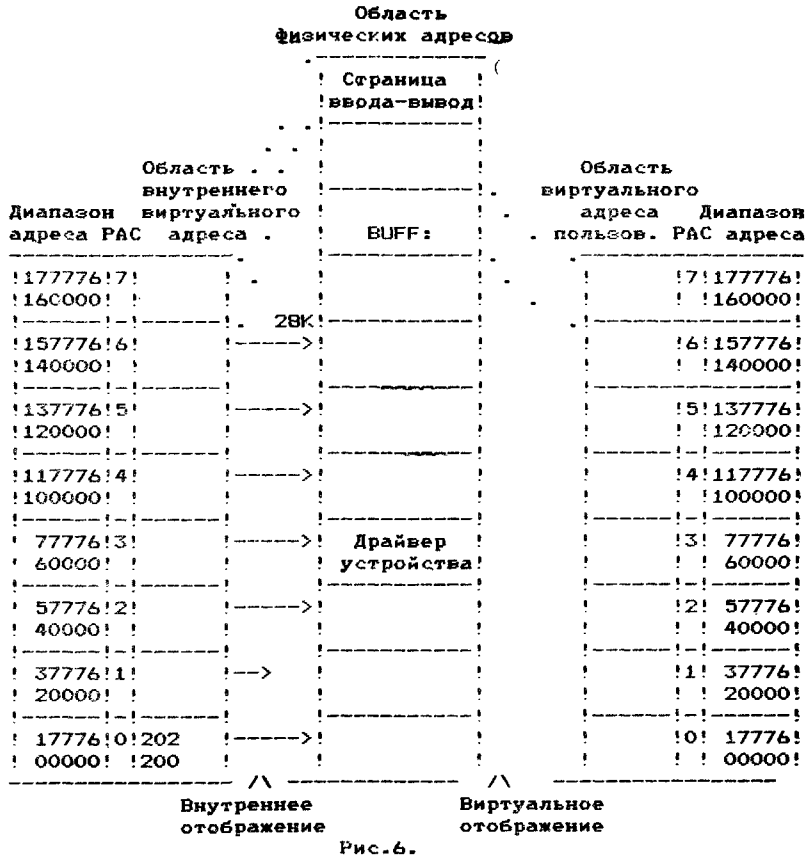


Рис.6.

Монитор ФОДОС-2 предоставляет подпрограммы для драйверов для доступа к буферу данных пользователя в физической памяти. Далее описаны эти подпрограммы и ситуации, в которых они используются.

5.4.3. Элемент очереди в XM

Для того, чтобы найти фактический буфер пользователя в физической памяти, драйверу требуется дополнительное информационное слово в элементе очереди. Это слово является значением для PAC1, которое, будучи объединено с адресом виртуального буфера пользователя, обеспечивает физический адрес буфера. Хотя для драйверов в XM-системе используется только одно дополнительное слово, элемент очереди предоставляет место для двух слов. Эти два слова, со смещениями 20 и 22, сохраняются для будущего использования в системе и не должны использоваться пользователем. Когда условно обозначенный системой MMG$T установлен 1, запрос .QELDF, вызванный посредством .DRDEF в начале драйвера, расширяется при генерации допустимых смещений для элемента очереди XM. Ниже приведено это макрорасширение:

```
Q.LINK=0          (связь со следующим элементом очереди)
Q.CSW=2.          (указатель слова состояния канала)
Q.BLKN=4.         (номер физического блока)
Q.FUNC=6.         (код специальной функции)
Q.JNUM=7.         (номер задания)
Q.UNIT=7.         (номер привода устройства)
Q.BUFF=^O10       (адрес виртуального буфера пользователя)
Q.WCNT=^O12       (счетчик слов)
Q.COMP=^O14       (код подпрограммы завершения)
Q$LINK=-4         (символы для облегченного обращения)
Q$CSW=-2
Q$BLKN=0
Q$FUNC=2
Q$JNUM=3
Q$UNIT=3
Q$BUFF=4
Q$WCNT=6
Q$COMP=^O10
Q.PAR=^O16        (значение PAC1)
Q$PAR=^O12
Q.ELGN=^O24       (размер элемента очереди)
```

5.4.4. Устройства произвольного доступа к данным: подпрограмма $MPPHY

Устройства произвольного доступа к данным, например, большинство дисков, работают, обычно, с 18-разрядными или 22-разрядными адресами, поэтому их драйверам нет необходимости отображаться в буфер пользователя. Эти драйверы используют подпрограмму монитора $MPPHY для поиска буфера пользователя в физической памяти. $MPPHY использует значение Q.PAR элемента очереди и адрес виртуального буфера Q.BUFF для создания истинного 18- или 22-разрядного адреса буфера пользователя.

Формат вызова подпрограммы $MPPHY следующий:

JSR PC,@ $MPPTR

где $MPPTR — содержит указатель подпрограммы $MPPHY в резидентном мониторе; запрос .DREND размещает этот указатель в конце драйвера; указатель заполняется во время начальной загрузки (для системного устройства) или во время LOAD (для устройства данных).

Перед вызовом: R5 должен указывать на Q.BUFF, пятое слово элемента очереди.

После вызова: (SP), первое слово в стеке, содержит шестнадцать младших разрядов физического адреса буфера;
2(SP), второе слово в стеке, содержит 2 старших разряда физического адреса буфера в разрядах 4 и 5 (для 18-разрядного адреса) или в разрядах с 4 по 9 (для 22-разрядного адреса);
R5 указывает на Q.WCNT, шестое слово элемента очереди; значение не изменяется.

Пример:

```
CMP     (R5)+,(R5)      ;переходит к адресу бу-
                        ;фера в элементе очереди
JSR     PC,@ $MPPTR     ;переводит виртуальный
                        ;адрес пользователя в
                        ;физический
MOV     (SP)+,—(R4)     ;засылает 16 младших
                        ;разрядов в RKBA, стар-
                        ;шие разряды — в стек
MOV     (R5)+,—(R4)     ;засылает счетчик слов в
                        ;RKWC
BEQ     7$              ;счетчик равен 0=поиск
BMI     5$              ;счетчик отрицательный
                        ;=запись всех до по-
                        ;следнего
```

```
        NEG     @R4                     ;счетчик положительный
                                        ;=считывать, фиксируя
                                        ;счетчик для контролле-
                                        ;ра
        MOV     #CSIE!FNREAD!CSGO,R3    ;функция считывания
5$:     BIS     (SP)+,R3                ;присоединение старших
                                        ;разрядов адреса в функ-
                                        ;цию
        MOV     R3, -(R4)               ;начало операции
6$:     RTS     PC                      ;выход из прерывания
```

Этот пример взят из драйвера RK.

5.4.5. Символьно-ориентированные устройства: подпрограммы $GETBYT и $PUTBYT

Драйверы символьно-ориентированных устройств, таких как перфоленточное устройство ввода-вывода построчно-печатающее устройство, должны сами передавать данные из устройства в область буфера пользователя. Устройство само использует регистры в странице ввода-вывода для записи одного символа за раз. Драйвер может использовать две подпрограммы монитора — $GETBYT и $ PUTBYT — для передвижения данных между страницей ввода-вывода и областью буфера пользователя.

5.4.5.1. Подпрограмма $GETBYT

Драйвер может использовать подпрограмму монитора $GETBYT для перемещения байта из области буфера физической памяти в стек. Затем драйвер может перемещать символы в регистр буфера данных устройства страницы ввода-вывода и инициировать передачу ввода-вывода.

Формат вызова подпрограммы $GETBYT:

```
JSR PC,@ $GTBYT
```

где $GTBYT — содержит указатель подпрограммы $GETBYT в резидентном мониторе; запрос .DREND размещает этот указатель в конце драйвера; указатель заполняется во время начальной загрузки (для системного устройства) или во время LOAD (для устройства данных).

Перед вызовом:

R4 должен указывать на Q.BLKN, третье слово элемента очереди.

После вызова:

(SP), первое слово в стеке, содержит следующий байт из буфера пользователя в младшем байте; содержимое старшего байта не определено;

R4 не изменяется.

Адрес буфера (Q.BUFF) в элементе очереди устанавливается 1. Если возникает переполнение отображения, подпрограмма вычитает 20000 из значения в Q.BUFF и добавляет 200 к значению в Q.PAR. Более подробно переполнение отображения описано в п. 5.4.7.

Пример:

```
MOV     PCCQE,R4          ;указатель текущего элемента
                          ;очереди
MOV     #PP$CSR,R5        ;указатель регистра состоя-
                          ;ния перфоленточного устрой-
                          ;ства ввода-вывода
TST     (RT)+             ;ошибка?
BMI     PPERR             ;да, в устройстве нет ленты
TST     Q$WCNT(R4)        ;есть еще символы для выво-
                          ;да?
BEQ     PCDONE            ;нет, готовность к передаче
INC     Q$WCNT(R4)        ;уменьшается счетчик байтов
                          ;(он отрицательный)
JSR     PC,@$GTBYT        ;изымается байт из буфера
                          ;пользователя
MOVB    (SP)+,@R5         ;выводит его на перфоленту
```

Этот пример из драйвера PC показывает как драйвер получает байт из буфера пользователя и выводит его на перфоленту.

5.4.5.2. Подпрограмма $PUTBYT

*После успешной передачи данных драйвер может получ*ить символ из регистра буфера данных устройства страницы ввода-вывода и занести его в стек. Затем он может использовать подпрограмму $PUTBYT для перемещения байта из стека в буфер пользователя физической памяти.

Формат вызова подпрограммы $PUTBYT следующий:

```
JSR PC,@$PTBYT
```

где $PTBYT — содержит указатель подпрограммы $PUTBYT в резидентном мониторе; запрос .DREND размещает этот указатель в конце драйвера; указатель заполняется во время начальной загрузки (для системного устройства) или во время LOAD (для устройства данных).

Перед вызовом:

R4 должен указывать на Q.BLKN, третье слово элемента очереди; байт для передачи в буфер пользователя должен находиться вверху стека; символ должен находиться в младшем байте первого слова стека; старший байт неопределен.

После вызова:

слово, содержащее символ для передачи, удаляется из стека. R4 не изменяется.

Адрес буфера (Q.BUFF) в элементе очереди устанавливается 1. Если возникает переполнение отображения, подпрограмма вычитает 20000 из значения в Q.BUFF и добавляет 200 к значению в Q.PAR. Более подробно переполнение отображения описано в п. 5.4.7.

Пример:

```
MOV     PRCQE,R4          ; R4 указывает на Q.BLKN
MOVB    @#PRB,—(SP)       ; засылает символ
JSR     PC,@$PTBYT        ; перемещает его в буфер поль-
                          ; зователя
DEC     Q$WCNT(R4)        ; уменьшает счетчик байтов
```

Этот пример из драйвера PC показывает, как драйвер получает символ из перфоленточного устройства ввода-вывода и перемещает его в буфер пользователя.

5.4.6. Другие устройства: подпрограмма $PUTWRD

Подпрограмма монитора $PUTWRD подобна $PUTBYT, с тем исключением, что $PUTWRD перемещает в буфер пользователя физической памяти слово, а не байт. Она полезна, когда драйверу необходимо передать информацию о состоянии словом в буфер пользователя, а не символ данных из устройства. $PUTWRD могут использовать драйверы любого типа устройств.

Формат вызова подпрограммы $PUTWRD следующий:

```
JSR PC,@$PTWRD
```

где $PTWRD — содержит указатель подпрограммы PUTWRD в резидентном мониторе; запрос .DREND размещает этот указатель в конце драйвера; указатель заполняется во время начальной загрузки (для системного устройства) или во время LOAD (для устройства данных).

Перед вызовом:

R4 должен указывать на Q.BLKN, третье слово элемента очереди; слово для передачи в буфер пользователя должно находиться вверху стека.

После вызова:

слово для передачи удаляется из стека.
R4 не изменяется.

Адрес буфера (Q.BUFF) в элементе очереди устанавливается 1. Если возникает переполнение отображения, подпрограмма вычитает 20000 из значения в Q.BUFF и добавляет 200 к значению в Q.PAR. Более подробно переполнение отображения описано в п. 5.4.7.

Пример:

```
MOV     #DDNBLK,—(SP)  ; помещает размер в блоках в
                       ; стек
MOV     DYCQE,R4       ; R4 указывает на Q.BLKN
JSR     PC,@ $PTWRD    ; вызов подпрограммы
```

Этот пример из драйвера DY показывает как драйвер отвечает на вызов специальной функции, которая запрашивает размер установленного в данный момент тома. В данном случае имеется гибкий диск с двойной плотностью записи. Драйвер использует $PUTWRD для перемещения размера тома в область буфера пользователя.

5.4.7. Драйверы, которые имеют непосредственный доступ к буферу пользователя

В некоторых ситуациях требуются комбинации процедур, описанных выше. В других, требуется только часть драйвера для передачи. Некоторые драйверы не могут использовать подпрограммы монитора и должны иметь непосредственный доступ к буферу пользователя.

Процедура драйвера для отображения в буфер пользователя следующая.

Устройства, такие как гибкий диск, передают данные по сектору за раз между самим диском и внутренним буфером данных диска. Однако гибкий диск не является устройством произвольного доступа к данным, поэтому драйвер DX не может использовать подпрограмму монитора $MPPHY. Кроме того, другие подпрограммы монитора для символьно-ориентированных устройств, имеющих в наличии внутренний буфер данных устройства, слишком медленны для практического использования. Поэтому, драйвер DX отображается в буфер пользователя физической памяти, а затем выполняет операцию ввода-вывода так, как если бы она была простой передачей между памятью и устройством. Драйвер осуществляет это отображение, заимствуя область PAC1.

$P1EXT копирует из драйвера в стек монитора необходимые инструкции по передаче данных, таким образом, перемещение инструкций происходит из возможной области PAC1. Затем $P1EXT устанавливает необходимое значение PAC1 и выполняет команды, скопированные в стек. После окончания $P1EXT восстанавливает PAC1, очищает стек монитора и засылает в драйвер по словам следующий список команд.

Вызов подпрограммы $P1EXT осуществляется по команде JSR R0, следующей после числа байтов для копирования в стек монитора плюс 2, списка команд для осуществления пе-

редачи данных и значения PAC1 (Q.PAR) из элемента очереди.

Пример:

```
        MOV     @#SYSPTR,R4     ;R4 содержит указатель
                                ;базы монитора
        MOV     P1EXT(R4),(PC)+ ;засылает адрес подпро-
                                ;граммы $P1EXT
$P1EXT: .WORD   P1EXT           ;указатель подпрограм-
                                ;мы $P1EXT
        .
        .
        .
;——— удаляет две следующие ниже строки, если диспетчер
;——— памяти не используется
        JSR             R0,@$P1EXT  ;разрешает монитору вы-
                                    ;полнять следующие ко-
                                    ;ды
        .WORD           PARVAL—.    ;число байтов для копи-
                                    ;рования+2
;———
2$:     TSTB            @R4         ;проверяет признак го-
                                    ;товности передачи
        BPL             2$          ;ожидает готовность
3$:     MOVB            (R2)+,@R5   ;перемещает символ из
                                    ;буфера пользователя на
                                    ;гибкий диск
        TSTB            @R4         ;устанавливает CSR для
                                    ;следующего раза
        DECB            R1          ;уменьшает счетчик пере-
                                    ;дач
        BNE             2$          ;если не 0, передавать
                                    ;остальные символы
;——— если используется диспетчер памяти, ограничиться
;——— списком значений PAC1
PARVAL:         .WORD       0       ;удаляется, если испол-
                                    ;зуется диспетчер памяти
;———
;далее продолжается обычный процесс
```

В этом примере приведена команда из драйвера DX, которая иллюстрирует описанную выше технику. R1 — счетчик переданных байтов, R2 — указатель буфера пользователя, R4 — указатель регистра состояния устройства гибких дисков и R5 — указатель регистра данных устройства гибких дисков. P1EXT — указатель значения фиксированного смещения подпрограммы $P1EXT относительно монитора.

К списку команд, переходящему в $P1EXT, накладываются следующие ограничения:
— в списке не должно быть команд, ссылающихся на другие ячейки драйвера, предназначенные для относительной адресации внутри самого списка;
— список команд может использовать стек для временного хранения, если невозможно перемещение других ранних значений из стека или сброса других значений в стек после его готовности;
— если используется список команд, R0 должен быть сохранен и восстановлен;
— не рекомендуется использовать список команд более 32-х слов, так как область стека ограничена.

Чтобы драйвер имел доступ непосредственно к буферу пользователя, необходимо понять, как область PAC1 отображается в область пользователя. Рис. 7 поможет разобраться в этом вопросе. Он показывает виртуальное задание в обычной XM-системе с буфером пользователя, размещенным в физической памяти свыше 28K слов. Программа пользователя отображается в буфер через PAC6. Драйвер использует PAC1, помещая туда значение Q.PAR из элемента очереди,

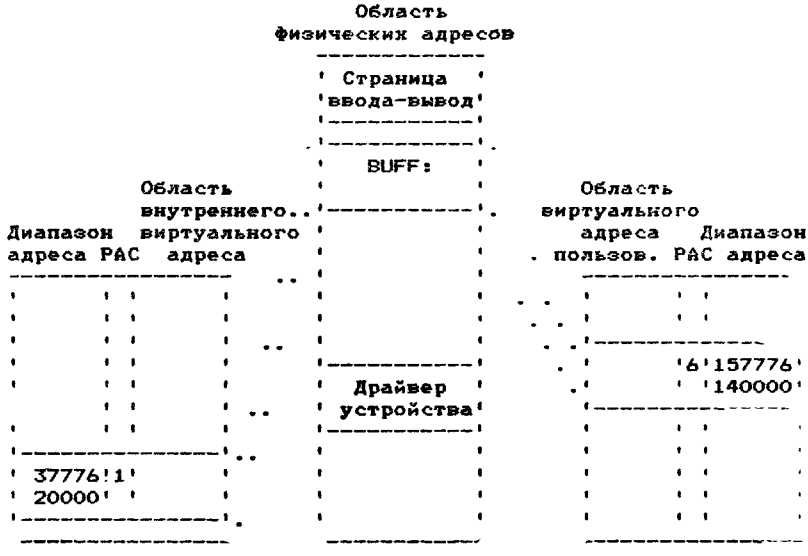


Рис.7.

Ниже приведен рис.8, на котором показана работа отображения.

а затем использует значение Q.BUFF из элемента очереди для доступа к буферу пользователя. PAC1 отображается в физическую память секциями 32-словных блоков (десятичное) и, в лучшем случае может отображать область размером 4K слов. (Размер страницы РПС1 (регистр признака страницы) установлен всегда так, что отображается вся страница.) Если буфер пользователя начинается с ячейки физической памяти, которая не кратна 32-м словам, PAC1 отображается в первую же область выше начала буфера, кратную 32-м словам. Область отображения PAC1 может начинаться в физической памяти с любого адреса, две младшие восьмеричные цифры которого являются нулями. Таким образом, при отображении PAC1 в буфер пользователя будет отображено 4K или (4K—31) слов (десятичное).

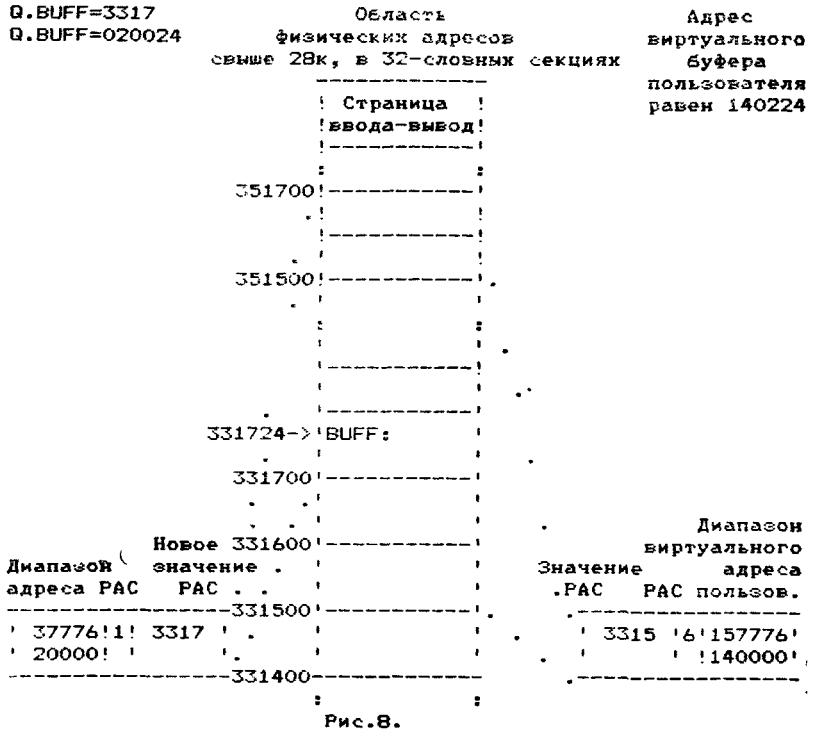


Рис.8.

На рис. 8 показана область буфера, размещенная у ячейки 331724 физической памяти с прикладной программой, отображенной в буфер посредством PAC6. Буфер на 24 (восьме-

ричное) байта выше 331700, которое является 32-словной границей. $P1EXT помещает значение Q.PAR, 3317, в PAC1, заменяя значение PAC1 по умолчанию (0200). Это вызывает отображение PAC1 в 4K-словную область физической памяти, начиная с адреса 331700. В результате, когда драйвер обращается к внутренним виртуальным адресам в диапазоне от 20000 до 37776, он выбирает ячейки физической памяти с 331700 по 351676. Так как значением в Q.BUFF является 20024, то используя это значение, драйвер может иметь доступ к области буфера пользователя в ячейке 331724.

Если количество данных для передачи слишком большое, то может потребоваться перемещение указателя буфера и регулирование отображения для его учета. Существует два способа перемещения указателя буфера. Самый легкий способ — это модификация PAC1. Например, каждым 32-м словам, перемещенным через буфер, добавляется 1 к значению PAC1. Драйвер DX, например, передает 64 слова за раз, добавляя 2 к значению PAC1 после каждой передачи, чтобы избежать переполнения отображения.

Другим способом перемещения указателя буфера является модификация значения Q.BUFF посредством изменения значения в самом элементе очереди. Сначала необходимо после модификации значения Q.BUFF сравнить новое значение с 40000. Если значение больше или равно 40000, из него надо вычесть 20000 и прибавить затем 200 к Q.PAR. Это позволит не только следить за регулированием отображения, но и поможет избежать переполнение отображения.

И, наконец, необходимо выбрать любую ячейку в буфере пользователя, если дано смещение байта от начала буфера. По существу, необходимо определить число 32-словных секций в 16-разрядном смещении байта делением на 100 (восьмеричное) и прибавлением частного к PAC1, а остатка к Q.BUFF. Тогда можно правильно выбрать ячейку в буфере пользователя.

Например, предположим, что нужен байт со смещением 12345 от начала буфера, показанного на рис. 8. Частное от деления 12345 на 100 равно 123, остаток — 45. Прибавление 123 к текущему значению Q.PAR, которое равно 3317, дает 3442 для нового значения PAC1. Прибавление 45 к значению Q.BUFF, которое равно 020024, дает 020071 — новый адрес буфера (в данном случае получен адрес байта).

5.5. Подпрограмма обработки прерываний в системе XM

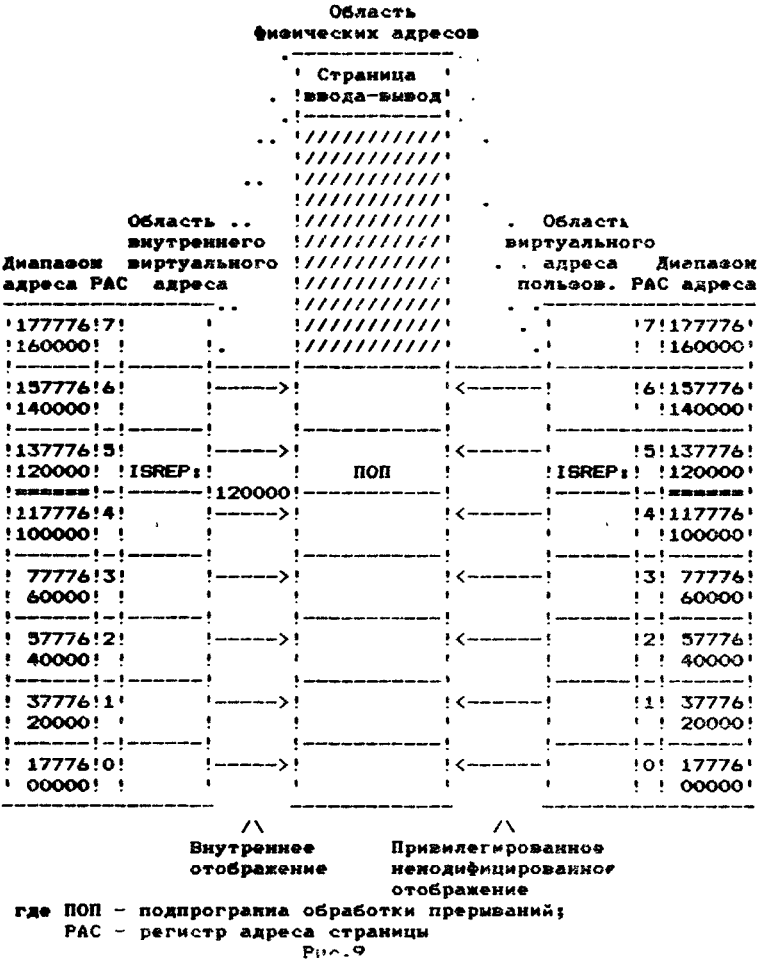
Существуют два вида заданий в системе XM, виртуальное задание и привилегированное задание, виртуальные задания

не могут содержать внутреннюю подпрограмму обработки прерываний (см. разд. 4). По самому определению виртуального отображения, виртуальные задания не имеют доступа к странице ввода-вывода устройства. Следовательно, они не могут установить разряд разрешения прерывания устройства или передавать данные в (или из) регистр буфера данных устройства.

Если задание, содержащее внутреннюю подпрограмму обработки прерываний, должно выполняться в системе ХМ, то оно должно быть выполнено как привилегированное задание. Привилегированное отображение предоставляет программе нижние 28К слов памяти и доступ к странице ввода-вывода и позволяет программе отображать частями область виртуального адреса пользователя в расширенную физическую память, если этого требует программа.

Когда прерывания происходят в ХМ, подпрограмма их обработки выполняет отображение внутренне, не пользовательски. Это означает, что как бы программа ни отображала некоторые свои области виртуальных адресов в расширенную память, подпрограмма обработки прерываний выполняет по умолчанию внутреннее отображение нижних 28К слов памяти плюс страницы ввода-вывода. Происходит следующее: сначала ограничения ХМ требуют, чтобы отображение подпрограммы обработки прерываний и других используемых ею данных было то же, что и внутреннее отображение в любое время, когда происходит прерывание.

Рис. 9 содержит схему внутреннего отображения по умолчанию, которое предусматривает доступ к нижним 28К словам памяти и к странице ввода-вывода. Это также схема отображения для привилегированного задания, если оно начало выполняться. Эта схема отображения эффективна всегда, когда прерывание обрабатывается. (Заштрихованная область на рисунке представляет собой память, которая недоступна пользовательскому заданию.) На рис. 9 векторы прерывания 200 и 202 содержат точку входа подпрограммы обработки прерываний, называемую ISREP:, и значение 340, которое является новым приоритетом PS. Если произошло прерывание, система использует внутреннее отображение для поиска подпрограммы обработки прерываний. В этом примере она начинается с адреса 120000. После этого привилегированное отображение и внутреннее отображение одинаковы на этой диаграмме, подпрограмма обработки прерываний находится в физической памяти точно в месте точки внутреннего отображения, тогда она может быть выполнена правильно.

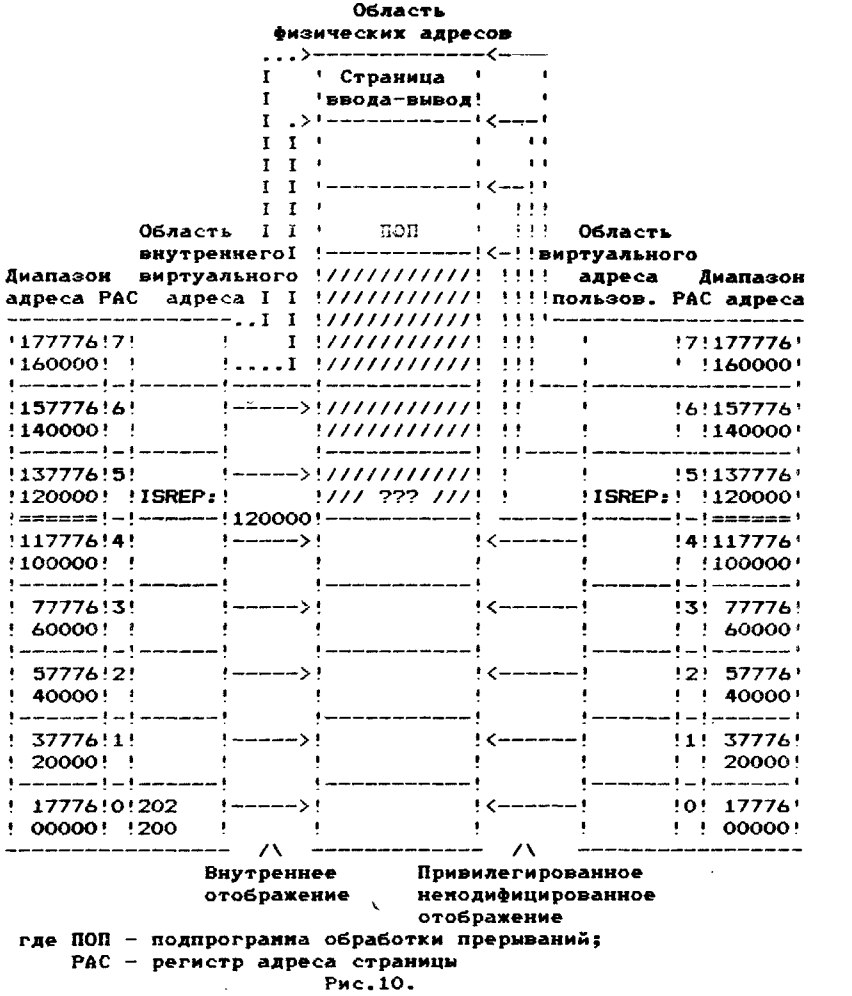


где ПОП - подпрограмма обработки прерываний;
РАС - регистр адреса страницы

Рис.9

Рис. 10 содержит привилегированное задание, которое изменяет отображение виртуального адреса пользователя. (Заштрихованная на рис. 10 область представляет память, которая не доступна пользователю.) Например, подпрограмма обработки прерываний не может быть выполнена правильно, если прерывание произошло потому, что подпрограмма обработки прерываний не была расположена в физической памяти в должном месте. Указанная область памяти до внутреннего отображения содержит произвольные данные или команды.

Второе ограничение для подпрограммы обработки прерываний в ХМ относится к области монитора, использующей

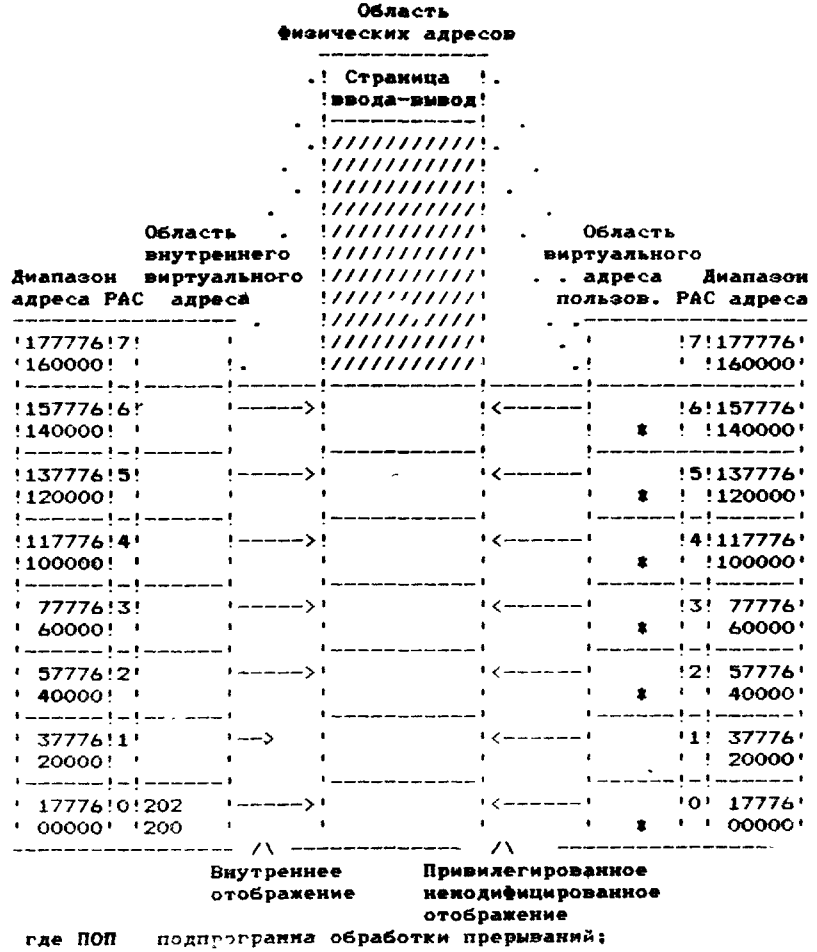


где ПОП - подпрограмма обработки прерываний;
PAC - регистр адреса страницы

Рис.10.

регистр адреса страницы 1 (PAC1) при внутреннем отображении. PAC1 контролирует отображения виртуальных адресов с 20000 до 37776. Если XM-монитор загружается при внутреннем отображении, то виртуальный адрес отображается в тот же самый физический адрес. Однако монитор сам использует PAC1 для отображения блоков области EMT и буфера данных пользователя. Когда же система выполняется, внутренние виртуальные адреса, расположенные в PAC1, могут быть отображены в любое место физической памяти, и у

пользователя нет возможности проконтролировать это. Недопустимо, чтобы подпрограмма обработки прерываний и все данные, которые ей необходимы, располагались в виртуальном адресе области, отображенной в PAC1. Рис. 11 иллюстрирует это ограничение. Допустимое нахождение подпрограммы обработки прерываний предполагается тогда, когда привилегированное отображение совпадает с внутренним отображением во время прерывания (на диаграмме отмечено «*»).



где ПОП - подпрограмма обработки прерываний;
PAC - регистр адреса страницы

Рис.11.

Если подпрограмма обработки прерываний нуждается в окне в памяти, она может заимствовать PAC1 так же, как это делает монитор. Она должна сохранить содержимое, значения которого ей необходимы, и восстановить истинное содержимое сразу же после выхода. Это можно сделать с помощью запросов .INTEN или .FORK, но не .SYNCH.

Если система использует драйвер MQ для связи с системным заданием, и параметр условной трансляции MQH$P2=1 определен во время генерации системы, все ограничения для PAC1 также относятся к PAC2 — области адресов с 40000 до 57777 включительно.

Если в подпрограмме обработки прерываний используется запрос .SYNCH, то строки кодов, следующих за .SYNCH, выполняются почти как подпрограмма завершения. Подпрограмма завершений выполняется в XM с регистром пользователя, стеком пользователя и пользовательским отображением. Однако после кодов, следующих за .SYNCH, ослабевает роль подпрограммы обработки прерываний, она выполняется с регистром пользователя, исключая внутреннее отображение. Таким образом, коды, следующие за запросом .SYNCH, вызванным в XM-мониторе, должны соблюдать те же ограничения, что и для главного тела подпрограммы обработки прерываний: ее отображение должно совпадать с внутренним отображением в то время, когда произошло прерывание или когда выполняется подпрограмма завершения. И последнее, она должна полностью соблюдать ограничения, налагаемые на PAC1 и PAC2.

6. УСТАНОВКА ДРАЙВЕРА УСТРОЙСТВА

6.1. Драйверы системного устройства и начальный загрузчик

В следующих пунктах описанию файлов монитора предшествует объяснение, как создать драйвер системного устройства или как изменить существующий драйвер для использования его в качестве драйвера системного устройства. Дано довольно подробное описание деталей первичного драйвера и различных программ начального загрузчика. В конце подраздела дана информация об установке посредством DUP нового системного устройства.

6.1.1. Файл монитора должен находиться на системном устройстве и может иметь любое имя, но требуемым типом файла всегда является .SYS. В ФОДОС-2 они имеют опреде-

ленное название: FMONBL.SYS, FMONSJ.SYS и FMONFB.SYS. Если монитор создается в процессе генерации системы, именем монитора всегда является FMONXX.SYG, где XX — одно из четырех: BL, SJ, FB или XM. Перед использованием монитор необходимо переименовать, чтобы его тип был .SYS.

Блоки с 0 по 4 каждого файла монитора содержат вторичный загрузчик. Вторичный загрузчик загружает драйвер системного устройства и монитор в память. Он также модифицирует таблицы монитора для соединения монитора с драйвером устройства и присваивает имена устройствам по умолчанию DK и SY.

Сам файл монитора не содержит код указателя устройства, к тому же не имеет связей с любым конкретным драйвером устройства до начальной загрузки. Вместо этого каждый драйвер устройства, который может быть использован в качестве драйвера системного устройства, имеет специальный блок кода указателя устройства, названный первичным драйвером, который используется вторичным загрузчиком для считывания файла драйвера системного устройства и файла монитора с системного устройства Вторичный загрузчик имеет в своем собственном блоке 0 место для записи первичного драйвера.

6.1.2. Для создания драйвера системного устройства необходимо к стандартному драйверу устройства данных прибавить первичный драйвер. Драйвер системного устройства может содержать параметры SET. Поэтому, если параметры SET являются частью драйвера, нет необходимости удалять их при создании драйвера системного устройства.

6.1.2.1. Первичный драйвер, который прибавляется к стандартному драйверу устройства данных, состоит из четырех частей:

— подпрограммы входа;
— начального загрузчика программного обеспечения;
— подпрограммы считывания начального загрузчика;
— подпрограммы ошибок начального загрузчика.

Первичный драйвер работает вместе с начальным загрузчиком системы ФОДОС-2 для загрузки нового системного устройства. Первичный драйвер целиком содержится в программной секции (P—SECT) DDBOOT, где DD — это двухсимвольное имя устройства. Код выполняется в ячейке 0 физической памяти.

6.1.2.2. Подпрограмма входа

Точкой входа для первичного драйвера является DDBOOT::. Эта ячейка должна содержать только две коман-

ды и они должны следовать за последовательностью начальных загрузчиков. Этими командами являются NOP и переход к началу начального загрузчика программного обеспечения. Если начало начального загрузчика программного обеспечения дальше пределов действия команды BR, то можно использовать команду JMP, которая начинает загрузку программного обеспечения.

Пример:

```
RKBOOT::NOP
        BR      BOOT1
```

В этом примере приведена подпрограмма входа для драйвера RK (BOOT1 определяется в первичном драйвере).

Любой аппаратный загрузчик вызывает коды в П-секции DDBOOT для загрузки с ячейки 0 памяти. Он начинает также выполнение с DDBOOT::.

6.1.2.3. Начальный загрузчик программного обеспечения выполняется в результате перехода из подпрограммы входа. До входа все регистры могут быть использованы начальным загрузчиком программного обеспечения, который должен выполнить следующие функции в таком порядке.

1. Установить стек с ячейки 10000.
2. Сохранить номер привода устройства, с которого была загружена система (значение в диапазоне от 0 до 7). Метод, который используется для нахождения номера привода меняется в зависимости от устройства; некоторые номера приводов передаются в R0, другие должны извлекаться из CSR. Номер привода можно сохранить в стеке или, если это необходимо, где-либо в памяти.
3. Вызвать подпрограмму считывания начального загрузчика для считывания всего начального загрузчика.
4. Поместить указатель подпрограммы считывания начального загрузчика в B$READ.
5. Поместить значение «B$DNAM» в RADIX-50 в B$DEVN.
6. Записать номер привода устройства в B$DEVU.
7. Перейти к B$BOOT в начальном загрузчике системы ФОДОС-2 для продолжения.

Начальный загрузчик программного обеспечения должен находиться в первичном драйвере сразу же ниже ячейки DDBOOT+664. (Ячейки с 664 по 776 содержат подпрограмму ошибок, созданную посредством запроса .DREND).

6.1.2.4. Подпрограмма считывания начального загрузчика

Целью этой подпрограммы является считывание тома на приводе устройства, с которого была загружена система. Она

вызывается начальным загрузчиком системы и начальным загрузчиком программного обеспечения (см. п. 6.1.2.3).

Интерфейс, посредством которого другие подпрограммы передают информацию в подпрограмму считывания начального загрузчика, следующий:

R0 содержит номер считываемого блока;
R1 содержит счетчик считываемых слов;
R2 содержит адрес буфера памяти, в котором будут записываться данные.

Подпрограмма считывания начального загрузчика может использовать все имеющиеся регистры и стек.

Подпрограмма считывания начального загрузчика должна быть непрерываемой подпрограммой при считывании тома в соответствии с параметрами, передаваемыми в R0—R2. При ошибке подпрограмма должна переходить к BIOERR. Если ошибок нет, она должна возвращаться по команде RTS PC с очищенным разрядом переноса.

Подпрограмма считывания начального загрузчика должна находиться в первичном драйвере с ячейки DDBOOT+210. (Ячейка 210 — самый младший адрес, с которого может начинаться подпрограмма считывания.)

6.1.2.5. Подпрограмма ошибок начального загрузчика

Эта подпрограмма начинается с ячейки BIOERR::. Код в этой подпрограмме полностью поставляется посредством запроса .DREND, который находится в конце первичного драйвера.

6.1.2.6. Запрос .DRBOT используется для установки первичного драйвера. Он выдает также запрос .DREND для пометки конца драйвера, поэтому первичный драйвер не будет загружаться в память во время обычных операций. Вообще, код в первичном драйвере не должен быть позиционно независимым (PIC). Однако любое обращение не в PIC-кодах должно быть выражено относительно DDBOOT::. (Ячейками с 60 по 206 запрещено пользоваться.)

Формат запроса .DRBOT следующий: .DRBOT NAME, ENTRY,READ

где NAME — двухсимвольное имя устройства;
ENTRY — точка входа в подпрограмму начального загрузчика программного обеспечения;
READ — точка входа в подпрограмму считывания начального загрузчика.

Запрос .DRBOT помещает указатель в начало первичного драйвера, в ячейку 62 файла драйвера. Он помещает размер первичного драйвера в байтах в ячейку 64. Размер первично-

го драйвера, включая подпрограмму ошибок, полученную посредством .DREND, не должен превышать 1000 восьмеричных байтов. Ячейка 66 содержит смещение от начала первичного драйвера к началу подпрограммы считывания начального загрузчика.

Перед вызовом запроса .DRBOT необходимо вызвать запрос .DREND. Затем поместить код первичного драйвера между .DRBOT и .DREND, помня, что размер первичного драйвера не должен превышать одного блока, т. е. 1000 восьмеричных байт, включая подпрограмму ошибок и ячейки с 60 по 206. Из вышесказанного следует, что запрос .DREND вызывается дважды в драйвере системного устройства: один раз посредством .DRBOT и один раз, когда он используется в конце первичного драйвера. Первое появление .DREND закрывает несистемную часть драйвера устройства и устанавливает таблицу указателей в мониторе среди других элементов. Второй вызов запроса .DREND создает подпрограмму ошибок начального загрузчика, BIOERR, вместо повторения таблицы указателей.

Если для загрузки нового устройства используется команда монитора BOOT, DUP передает номер системного привода в первичный драйвер, в ячейку 4722, и в R0. Если устройство загружено аппаратным загрузчиком или вспомогательной программой не системы ФОДОС-2, первичный драйвер должен определить номер привода загруженного устройства и сохранить его в ячейке 4722 и в регистре R0.

6.1.3. DUP и процесс начальной загрузки

В этом пункте показано, как DUP выполняет три команды, относящиеся к начальной загрузке. Это следующие команды:

```
BOOT DDN:FILNAM
COPY/BOOT XXN:FILNAM DDM:
BOOT DDN:
```

6.1.3.1. Команда BOOT DDN:FILNAM. Эта команда используется для загрузки программного обеспечения указанного файла монитора с указанного устройства. В командной строке DD — двухсимвольное имя устройства; N — номер привода этого устройства. И файл нового монитора, и файл нового драйвера устройства должны находиться на системном устройстве DD.

После подачи этой команды DUP проверяет, чтобы устройство было устройством произвольного доступа к данным. Затем она размещает файл монитора FILNAM.SYS на устройстве (тип файла .SYS является и обязательным, и типом

файла по умолчанию). DUP считывает пять первых блоков, блоки с 0 по 4, в буфер памяти. Эти блоки содержат вторичный загрузчик монитора.

Предпоследнее слово в блоке 4 содержит префикс драйверов, связанных с этим монитором. DUP использует его для построения имени файла драйвера устройства, обычно DD.SYS или DDX.SYS. DUP считывает блок 0 файла драйвера устройства в буфер памяти, используя содержимое ячеек 62 и 64 для поиска первичного драйвера и считывания его в буфер памяти.

Затем она копирует первичный драйвер в буфер у начала вторичного загрузчика, который также находится в буфере памяти. Она загружает информацию, показанную в табл. 9, для первичного драйвера и вторичного загрузчика.

Таблица 9

Смещение от начала буфера памяти	Содержимое
1	2
4722	Номер загружаемого привода
4724—4726	Имя загружаемого файла в RADIX-50
5000	Дата загрузки
5002—5004	Время загрузки

После этого DUP копирует первичный драйвер и вторичный загрузчик из буфера памяти в ячейки памяти с 0 по 5004. Затем она переходит к ячейке 1000, в начало вторичного загрузчика, чтобы он мог загрузить монитор и драйвер системного устройства в память.

Рис. 12 иллюстрирует эту процедуру.

6.1.3.2. Команда COPY/BOOT XXN:FILNAM DDM:

Эта команда используется для копирования вторичного загрузчика из файла монитора на устройстве XX в блоки 2, 3, 4 и 5 устройства DD. В командной строке XX — двухсимвольное имя устройства, на котором находится файл монитора; N — номер его привода; DD — двухсимвольное имя устройства, которое должно получить начальный загрузчик; M — номер его привода.

После подачи этой команды DUP проверяет, чтобы устройство было устройством произвольного доступа к данным. Затем она осуществляет поиск файла монитора FILNAM.SYS на устройстве XXN:. Она считывает пять первых блоков

файла монитора, блоки с 0 по 4, в буфер памяти. Эти блоки содержат вторичный загрузчик монитора.

DUP далее размещает соответствующий файл драйвера на устройстве DD:. Затем она считывает блок 0 файла драйвера устройства в буфер памяти, используя содержимое ячеек 62 и 64 для поиска первичного драйвера и считывания его в буфер памяти.

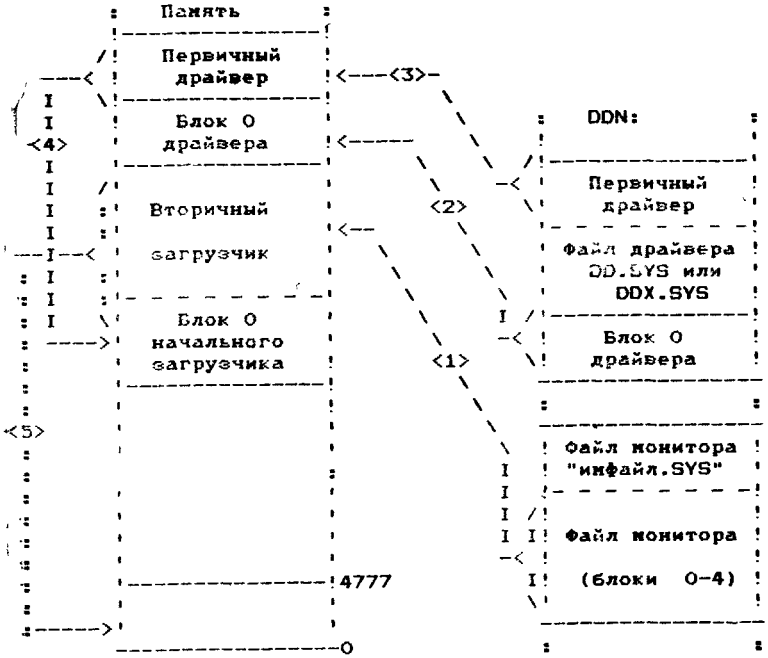


Рис.12.

Драйвер системного устройства DD должен находиться на устройстве DD: перед тем, как будет производиться копирование начального загрузчика на устройство. DUP загружает два слова в RADIX-50 для имени файла в ячейки 4724 и 4726 буфера памяти. Затем DUP копирует первичный драйвер в блок 0 устройства DD:. И, наконец, она записывает вторичный загрузчик в блоки со 2-го по 5-ый устройства DD:.

Рис. 13 иллюстрирует эту процедуру.

6.1.3.3. Команда BOOT DDN:

Эта команда используется для загрузки программного обеспечения указанного устройства, которое уже содержит

вторичный загрузчик указанного монитора в блоках со 2-го по 5-ый (помещенный туда по команде монитора COPY/BOOT). В командной строке DD — двухсимвольное имя загружаемого устройства; N — номер его привода. Файл нового монитора и драйвер нового устройства должны находиться на устройстве DDN:.

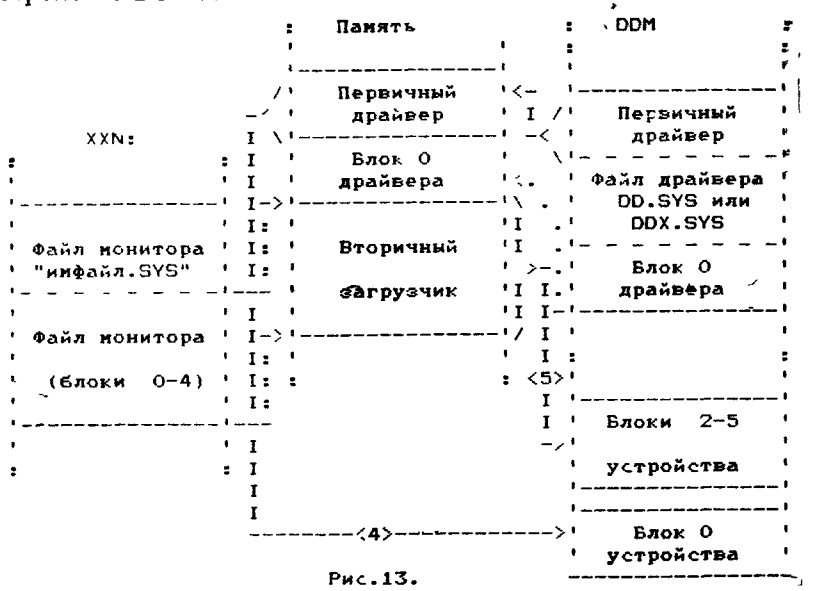


Рис.13.

После подачи команды DUP проверяет, чтобы устройство DD: было устройством произвольного доступа к данным. Затем она считывает блоки 2, 3, 4, 5 в буфер памяти. Эти блоки содержат вторичный загрузчик монитора. Первичный драйвер уже находится в ячейках с 0 по 776.

DUP производит поиск файла соответствующего драйвера на устройстве DDN:. Эта процедура является проверкой, чтобы том содержал драйвер системного устройства, так как только тогда он может быть загружен.

Затем DUP извлекает имя файла монитора из ячеек 724 и 726 блока 4 и производит поиск файла монитора на устройстве, чтобы убедиться в том, что он действительно там существует.

Далее DUP загружает информацию, приведенную в табл. 10 для первичного драйвера и вторичного загрузчика.

DUP копирует первичный драйвер и вторичный загрузчик из устройства в ячейки памяти с 0 по 4777. Затем она пере-

Таблица 10

Смещение от начала буфера памяти	Содержимое
1	2
4722	Номер загружаемого привода
5000	Дата загрузки
5002—5004	Время загрузки

ходит к ячейке 1000, в начало вторичного загрузчика, точку входа которого указывает DUP, чтобы вторичный загрузчик мог загрузить монитор и драйвер системного устройства в память.

Рис. 14 иллюстрирует эту процедуру.

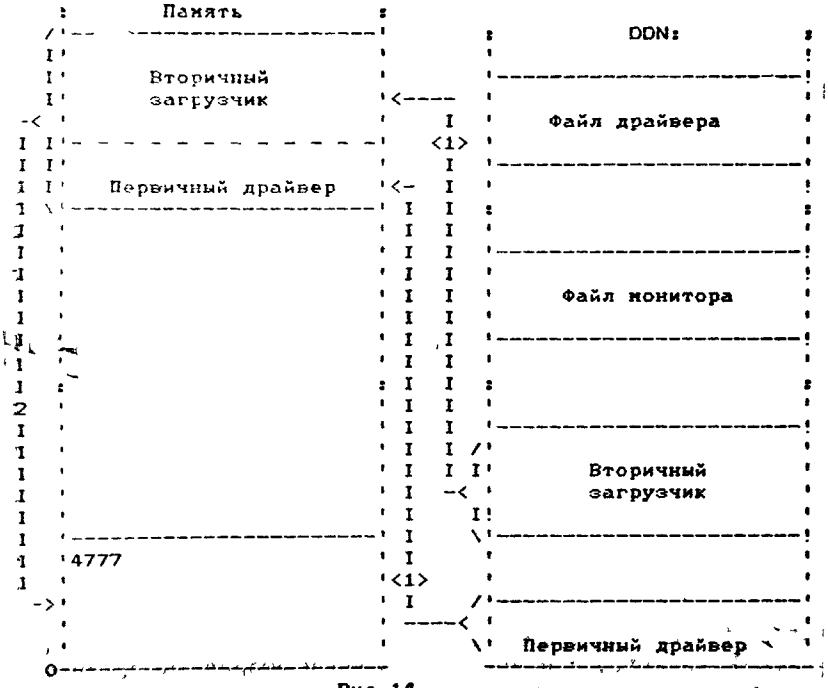


Рис. 14.

6.2. Как транслировать, связывать и устанавливать драйвер устройства

В последующих пунктах подробно описаны несложные процедуры: транслирование, связывание и установка драйвера нового устройства.

6.2.1. Транслирование драйвера устройства

Исходный файл драйвера может быть назван DD.MAC, где DD — двухсимвольное имя устройства. Для наглядности можно использовать переключатель АССЕМБЛЕРа /SHOW:MEB, который позволяет напечатать макрорасширение запросов .DRBEG и DRAST.

Для транслирования драйвера в системах SJ и FB используется следующая команда:

MACRO/CROSSREFERENCE/SHOW:MEB/LIST SYCND+DD/OBJECT

Для транслирования драйвера в XM-системе используется следующая команда: MACRO/CROSSREFERENCE/SHOW:MEB/LIST XM+SYCND+DD/OBJECT:DDX

где XM — исходный файл в системе ФОДОС-2, который показывает наличие в системе средства расширенной памяти, поддерживающего среду FB;

SYCND — условный файл системы; если система создана в процессе генерации, необходимо использовать этот файл, чтобы при транслировании драйвера условия драйвера и условия монитора соответствовали друг другу, и драйвер мог работать в этой среде; можно опустить этот файл, если транслируется драйвер устройства, которое будет выполняться с дистрибутивным монитором ФОДОС-2, или использовать файл SYCND.DIS, который является частью дистрибутивного пакета.

6.2.2. Связывание драйвера устройства

Если трансляция исходного файла прошла без ошибок, значит можно его связать. Для этого ВР системах SJ и FB используется следующая команда: LINK/MAP/EXECUTE:DD SYS DD

Для связывания драйвера в XM-системе используется следующая команда: LINK/MAP/EXECUTE:DDX.SYS DDX

6.2.3. Установка драйвера устройства

Перед использованием нового драйвера необходимо добавить информацию о нем в таблицу устройств монитора. Процедура добавления нового устройства называется установкой. В системе ФОДОС-2 имеются две отдельные подпрограммы, которые могут установить драйвер устройства: начальный загрузчик и команда монитора INSTALL. Обе подпрограммы требуют, чтобы в системе имелась аппаратура устройства до начала установки драйвера этого устройства.

(В п. 6.2.3.6 описывается способ обхода этого ограничения, если необходимо установить драйвер для несуществующего устройства.)

6.2.3.1. Использование начального загрузчика для автоматической установки драйверов

Подпрограмма начального загрузчика сначала осуществляет поиск драйвера системного устройства на устройстве, с которого загружается система, и устанавливает его. Затем она просматривает отдельные файлы драйверов на системном устройстве и пытается установить соответствующий драйвер для каждого устройства, которое она находит в системе. Если аппаратуры нет в наличии, начальный загрузчик не устанавливает устройство.

Затруднения в этой процедуре возникают тогда, когда имеется больше файлов драйверов, чем позиций устройства. Дистрибутивный монитор сохраняет одну позицию для каждого устройства, поддерживаемого в системе ФОДОС-2. Монитор созданный в процессе генерации системы, сохраняет одну позицию для каждого запрашиваемого пользователем устройства. Кроме того, он гарантирует ряд запрошенных пользователем пустых позиций. Считается, что позиция сохраняется для определенного устройства, если таблица монитора $PNAME имеет вход для этого устройства. Позиция пуста, если таблица $PNAME имеет нулевое слово.

Подпрограмма автоматической установки устройства при загрузке обладает набором приоритетов для определения, какие драйверы необходимо установить, если драйверов больше, чем позиций. Если все позиции пусты, начальный загрузчик устанавливает драйвер системного устройства и первые встретившиеся на системном устройстве драйверы, аппаратура устройств которых имеется в наличии. Например, если система имеет только пустые позиции, начальный загрузчик устанавливает драйвер системного устройства и первые семь законных драйверов, которые встретятся на системном устройстве.

Если некоторое количество позиций сохраняется для указанных устройств (т. е. устройства имеют входы в таблице $PNAME), начальный загрузчик резервирует эти позиции для соответствующих драйверов, пока он проверяет наличие соответствующей аппаратуры. Если аппаратура имеется в наличии, начальный загрузчик устанавливает драйвер устройства. Если аппаратуры нет в наличии, начальный загрузчик очищает его вход в $PNAME, создавая таким образом пустую позицию.

Рис. 15 показывает алгоритм, который использует начальный загрузчик для установки драйвера устройства.

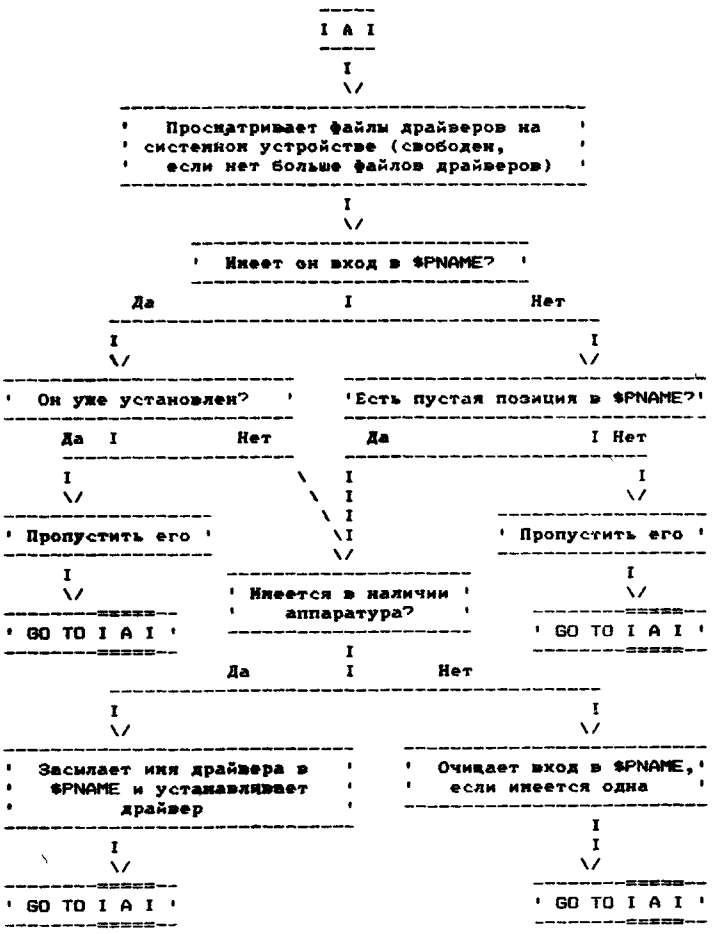


Рис.15.

Как видно из рис. 15, драйверы с входами в таблице $PNAME имеют высший приоритет во время загрузки. Если файл драйвера находится на системном устройстве, и аппаратура имеется в наличии, начальный загрузчик всегда устанавливает драйвер. При самостоятельном написании драйвера у пользователя не должно возникать затруднений при его установке в системе ФОДОС-2, так как можно рассчитывать

при установке драйвера на начальный загрузчик, в случае, если драйвер находится на системном устройстве, и имеется пустая позиция в таблицах монитора. Если в системе нет свободной позиции, то можно просто создать их (одну или более) путем записи файла драйвера устройства на системное устройство и перезагрузки системы. Можно также использовать команду монитора INSTALL (см. п. 6.2.3.2) для установки нового драйвера без перезагрузки системы. (Этот драйвер может быть одним из таких, которые начальный загрузчик не может установить из-за отсутствия свободных позиций, или это вновь созданный и скопированный на системное устройство.) Или, если система создана в процессе генерации, можно использовать запрос DEV (см. п. 6.2.3.3) для резервирования позиции для драйвера нового устройства и указать ему приоритет для установки во время загрузки.

Рис. 16 кратко излагает способы, которыми можно установить драйвер нового устройства

6.2.3.2. Использование команды INSTALL для установки драйверов вручную

Перед использованием команды INSTALL необходимо использовать команду SHOW для определения наличия пустых позиций устройств в системе. Если их нет, необходимо использовать команду REMOVE для удаления ненужного устройства и освобождения места для нового устройства, которое добавляется по команде INSTALL. (Форматы этих команд см. в [2].)

Если позиция устройства уже имеется, устройство будет установлено автоматически во время следующей загрузки системы. Если использовать команды REMOVE и INSTALL для добавления нового устройства к системе, то необходимо подавать эти команды каждый раз после каждой загрузки. Для автоматической установки нового устройства во время каждой загрузки, необходимо поместить команды REMOVE и INSTALL в косвенный файл запуска системы. Это избавляет пользователя от печати этих команд и дает видимость, что устройство постоянно установлено

6.2.3.3. Использование запроса DEV для помощи автоматической установки .

Если система создана в процессе генерации, можно связать исходный файл для добавления нового устройства к таблице $PNAME, отдавая этим самым предпочтение автоматической установке драйвера. Редактируемый файл — это файл SYSTBL.MAC — один из файлов, которые транслируются при создании файла монитора.

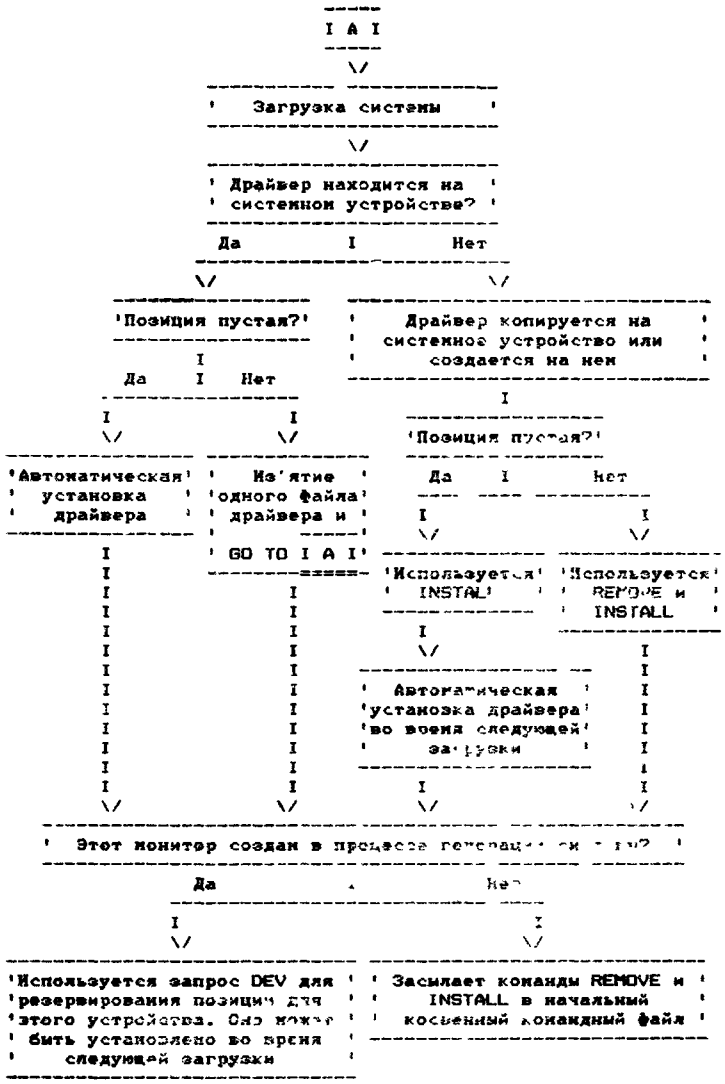


Рис.16.

Необходимо использовать запрос DEV в файле SYSTBL. MAC, чтобы устройство было постоянно добавлено к системе.

Формат запроса DEV следующий: DEV NAME,S

где NAME — двухсимвольное имя устройства;

S — слово состояния устройства (можно не указывать этот аргумент).

Пример:

```
DEV     RK      ;устанавливает диск
DEV     LP      ;устанавливает построчно-печатающее
                ;устройство
DEV     MT      ;устанавливает магнитную ленту
```

В этом примере показаны случаи использования запроса DEV в файле SYSTBL.MAC.

После редактирования файла SYSTBL.MAC и добавления запроса DEV для нового устройства, необходимо оттранслировать его, используя следующую команду:

MACRO/OBJECT:TBXX XX+SYCND+SYSTBL

где XX — это SJ, FB или XM; SJ.MAC, FB.MAC и XM.MAC — файлы ФОДОС-2, которые определяют параметры условной трансляции; эти параметры показывают, разрешена или нет, основная/фоновая обработка; XM.MAC показывает также, что имеется поддержка расширенной памяти.

Как только трансляция закончится, необходимо отредактировать объектные файлы для создания нового монитора.

6.2.3.4. Установка устройств, аппаратура которых имеется в наличии

Обе подпрограммы ФОДОС-2, и начальный загрузчик, и команда монитора INSTALL, могут устанавливать драйверы только тех устройств, аппаратура которых имеется в наличии в данной конфигурации системы. Подпрограммы находят ячейку 176 в блоке 0 драйвера устройства и проверяют ее содержимое (обычно это регистр состояния устройства, CSR). Если аппаратура устройства отсутствует, происходит таймаут канала, вызывая прерывание по вектору 4, из поля подпрограммы установки. В результате, ни команда INSTALL, ни подпрограмма начального загрузчика не будут устанавливать драйвер устройства. К тому же, команда INSTALL печатает сообщение:

?KMON—F—ILLEGAL DEVICE INSTALLATION

Подпрограммы установки предполагают, что аппаратура устройства имеется в наличии, если CSR отвечает на запрос канала. Однако этой простой проверки недостаточно в отдельных случаях для определения, аппаратура какого устройства имеется в наличии. Например, некоторым устройствам назначаются одинаковые адреса в странице ввода-вывода для одного или большего числа их регистров состояния. Если система ФОДОС-2 проверила адрес совместно используемой страницы ввода-вывода, то еще неизвестно, какое из двух

устройств имеется в наличии и какой драйвер необходимо устанавливать. Устройства гибких дисков обеих плотностей записи данных, например, имеют одинаковый адрес канала и одинаковый номер регистров состояния в странице ввода-вывода. Когда система пытается установить драйвер DX, она должна определить наличие аппаратуры и выяснить, является ли это устройство устройством гибких дисков или нет. Ясно, что система не должна устанавливать драйвер DX, если имеется аппаратура устройства гибких дисков с двойной плотностью записи.

Существует несколько отличий между двумя или большим числом устройств, которые следуют из регистров страницы ввода-вывода. Каждый драйвер одного из трудноопределимых устройств может проверить это различие и информировать подпрограмму установки ФОДОС-2, устанавливать рассматриваемый в данный момент драйвер устройства или нет.

6.2.3.5. Подпрограмма сравнения установки

Все драйверы ФОДОС-2 для устройств с совместно используемыми адресами страницы ввода-вывода содержат подпрограмму сравнения установки, чтобы различать, аппаратура какого устройства имеется в наличии, и чтобы разрешить или запретить установку данного драйвера. При написании драйвера устройства можно включить в него свою подпрограмму сравнения установки.

Вообще, подпрограммы сравнения установки, различающие аппаратура какого устройства имеется в наличии, основываются на трех следующих условиях.

1. Из двух устройств, совместно использующих несколько регистров, одно устройство имеет больше регистров, чем другое.

2. Если два устройства совместно используют адреса для всех регистров и если они имеют одинаковое число регистров, иногда одно устройство имеет разряд считывания/записи, где другое имеет только разряд считывания.

3. Иногда устройство имеет индивидуальный разряд или байт идентификации.

Затем подпрограммы сравнения установки определяют какое устройство имеется в наличии, основываясь на результатах проверки одного из вышеуказанных условий. Как только определение будет сделано, подпрограмма сравнения установки сигнализирует подпрограмме установки ФОДОС-2, нужно устанавливать данный драйвер или нет, а потом возвращается в монитор с установленным разрядом переноса для предот-

вращения установки и с очищенным разрядом переноса для разрешения установки.

Подпрограмма сравнения установки может использовать регистры R0 и R1; остальные регистры должны быть сохранены и восстановлены.

6.2.3.5.1. Точка входа подпрограммы сравнения установки

Подпрограмма сравнения установки должна начинаться с ячейки 200 блока 0 драйвера и не должна размещаться выше ячейки 356. Ячейка 200 — это точка входа, которую начальный загрузчик и команда монитора INSTALL используют для установки устройства данных. Ячейка 202 — это точка входа, которую начальный загрузчик использует для установки системного устройства. Команда монитора INSTALL никогда не использует эту ячейку. Если все равно, как устанавливается драйвер, как устройство данных или как системное устройство, необходимо поместить команду NOP в ячейку 200. Если драйвер должен быть установлен как драйвер системного устройства, необходимо использовать следующие команды для предотвращения его установки любыми другими способами:

```
        .=200                   ; несистемная точка входа
        BR      ERROR           ; переход к подпрограмме ошиб-
        .                       ; ок
        .
        .
ERROR:  SEC                     ; установка С-разряда для за-
                                ; прещения установки
        RTS     PC              ; возврат
```

6.2.3.5.2. Если аппаратура данного драйвера имеет дополнительный регистр

Если данный драйвер предназначен для устройства, которое совместно использует адрес страницы ввода-вывода с другим устройством, можно определить, какое устройство имеется в наличии, если оба устройства имеют различное количество регистров. Когда устройство для данного драйвера имеет на один регистр больше, чем другое, необходимо использовать следующие команды для проверки дополнительного регистра:

```
MOV     176,R0          ; устанавливает совместно исполь-
                        ; зуемый CSR
TST     N(R0)           ; проверяет дополнительный регистр
                        ; со смещением N от совместно ис-
                        ; пользуемого CSR
```

```
RTS     PC      ; возврат (с установленным с-разря-
                ; дом в случае недопустимого уст-
                ; ройства)
```

Эта подпрограмма проверяет дополнительный регистр. Если это не описанное устройство, канал прерывается, вызывается прерывание по вектору 4, и устанавливается разряд переноса. Подпрограмма сравнения установки возвращается в монитор с установленным разрядом переноса, показывая, что аппаратуры соответствующего устройства нет в наличии и драйвер не будет установлен.

С другой стороны, если дополнительный регистр реагирует на проверку, команда TST возвращается с очищенным разрядом переноса, означая, что аппаратура соответствующего устройства имеется в наличии и что ФОДОС-2 должен установить данный драйвер.

6.2.3.5.3. Если аппаратура для этого драйвера имеет несколько регистров

Если аппаратура для другого устройства, которое использует адрес страницы ввода-вывода совместно с устройством для этого же драйвера, имеющим больше регистров, то драйвер может проводить проверку на отсутствие дополнительных регистров. Если дополнительный регистр не найден, система должна установить данный драйвер.

Пример:

```
        MOV     176,R0  ; устанавливает совместно ис-
                        , пользуемый CSR
        TST     N(R0)   , проверяет дополнительные
                        ; регистры со смещением N от
                        ; 176. Есть устройство?
        BCC     1$      ; да, другое устройство
        CLC             ; нет, очищает С-разряд
        RTS     PC      ; устанавливает данный драй-
                        ; вер
1$:     SEC             , устанавливает С-разряд
        RTS     PC      ; не устанавливает данный
                        , драйвер
```

В этом примере подпрограмма сравнения установки проверяет наличие дополнительного регистра для другого устройства. Если его нет, система ФОДОС-2 устанавливает данный драйвер.

6.2.3.5.4. Если имеется разряд или байт идентификации

Если устройства, совместно использующие адрес страницы ввода-вывода, также совместно используют разряд или байт идентификации, то подпрограмма сравнения установки

может проверить разряд или байт и определить, аппаратура какого устройства имеется в наличии. Затем она может разрешить или запретить установку данного драйвера, основываясь на полученной информации.

В ФОДОС-2, например, устройства DX: и DY: совместно используют CSR. Разряд 11, названный CSRX02, очищается, если имеется аппаратура устройства DX:, и устанавливается — если DY:.

Пример:

```
        .ASECT
        .=200                   ;начало подпрограммы срав-
                                ;нения
        NOP                     ;производит проверку для сис-
                                ;темного или несистемного
                                ;устройства
        BIT   #CSRX02,@#176     ;это устройство DY:?
        BEQ   1$                ;нет, это DX:. Не устанавли-
                                ;вать драйвер DY
        TST   (PC)+             ;очищает С-разряд, переходит
                                ;к команде SEC. Имеется DY:,
                                ;устанавливать драйвер DY
1$:     SEC                     ;устанавливает С-разряд, не
                                ;устанавливает драйвер DY
        RTS   PC                ;возврат в монитор
```

В приведенном примере драйвер DY должен устанавливаться только в том случае, если имеется в наличии аппаратура устройства DY:.

6.2.3.5.5. Если одно устройство имеет разряд считывания/записи

Если одно из устройств, совместно использующих адрес страницы ввода-вывода, имеет разряд считывания/записи в CSR, где другое устройство имеет только разряд считывания, подпрограмма сравнения может определить аппаратуру, имеющуюся в наличии, посредством использования процедуры проверки разряда и разрешения или запрета установки данного драйвера на основании результатов проверки. Подпрограмма должна считывать разряд, изменять его и записывать обратно в CSR. Затем подпрограмма должна снова считывать разряд. Если значение разряда изменилось, имеется в наличии устройство с регистром считывания/записи. Если значение осталось неизменным, имеется в наличии устройство с регистром только считывания. Подпрограмма может установить соответственно разряд переноса и вернуться в мони-

тор. Если разряд переноса установлен, ФОДОС-2 не устанавливает этот драйвер. Если разряд переноса очищен, ФОДОС-2 устанавливает данный драйвер.

6.2.3.5.6. Игнорирование аппаратных ограничений

Если по какой-то причине необходимо установить драйвер устройства, аппаратуры которого нет в наличии в конфигурации данной системы, можно обойти начальный загрузчик и подпрограмму INSTALL посредством выполнения SIPP. Необходимо очистить ячейки 176 и 200 в блоке 0 файла драйвера, затем можно использовать команду INSTALL или перезагрузить систему для установки драйвера устройства.

6.3. Содержимое системного драйвера

В табл. 11 показано расположение системного драйвера после трансляции и редактирования. Ячейки, не указанные в табл. 11, зарезервированы для будущего использования.

Таблица 11

Ячейка	Содержимое
1	2
0	
52:	Размер драйвера в байтах (DDEND—DDSTRT)
54:	Размер устройства в блоках (DDSIZE)
56:	Слово состояния устройства (DDSTS)
60:	Слово параметров SYSGEN
62:	Начальный адрес первичного драйвера (из .DRBOT)
64:	Размер первичного драйвера в байтах (из .DRBOT)
66:	Смещение от начала первичного драйвера до начала подпрограммы считывания начального загрузчика (из .DRBOT)
70:	Смещение к области данных драйвера
110:	Новое имя в RADIX-50
112:	Номер версии; —1 для завершения списка
176:	Адрес CSR (DD$CSR)
200:	Начало кода установки устройства данных, если другое (или 0)
202:	Начало кода установки системного устройства, если другое
356:	Верхняя граница кода установки
360:	Резервируется для памяти, использующей разрядовые комбинации (360—377)
400:	Начало кода параметров SET (из .DRSET)
776:	Верхняя граница кода параметров SET
1000:	Адрес вектора (DD$VEC) (из .DRBEG)
1002:	DDINT—. (Из .DRBEG)
1004:	Новое слово состояния процессора (PSW) (из .DRBEG)
1006:	DDLQE (из .DRBEG)
1010:	DDCQE (из .DRBEG)
1012:	Точка входа драйвера

1	2
N	Точка входа по преждевременному прерыванию (из .DRAST; может быть более 1777)
N+2	Точка входа по прерыванию (из .DRAST; может быть более 1777)
1776:	Верхняя граница области, изменяемой кодом SET DD$END=. (Из .DREND; конец драйвера устройства)
DD$END:	$RLPTR: (из .DREND) $MPPTR: (из .DREND) $GTBYT: (из .DREND) $PTBYT: (из .DREND) $PTWRD: (из .DREND) $ELPTR: (из .DREND) $TIMIT: (из .DREND) $INPTR: (из .DREND) $FKPTR: (из .DREND) DDEND=. (Из .DREND)
DDEND: DDBOOT:	NOP начало первичного загрузчика (из .DRBOT) BR вход метки ENTRY из .DRBOT
	1)
ENTRY-14	020 (из .DRBOT)
ENTRY-12	Тип контроллера (из .DRBOT)
	2)
ENTRY-10	020 (из .DRBOT)
	3)
ENTRY-6	Контрольная сумма (из .DRBOT)
ENTRY-4	0 (из .DRBOT)
	4)
ENTRY-2	Тип гибкого диска (из .DRBOT)
	5)
ENTRY:	BR .+2 или BMI .+2 (Из .DRBOT) Начало подпрограммы считывания первичного загрузчика
662:	Верхняя граница первичного загрузчика
664:	Начало кода ошибок начального загрузчика
776:	Конец кода ошибок начального загрузчика

ПРИМЕЧАНИЯ:

1) Этот байт показывает тип ЦП. Значение 20 определяет процессор мини-ЭВМ.

2) Этот байт показывает тип файловой структуры диска. Значение 20 определяет файловую структуру ФОДОС-2.

3) Байт контрольной суммы — контрольная сумма предшествующих трех байтов. Она вычисляется как дополнение к сумме байтов.

4) Этот байт содержит номер идентификации загрузчика в разрядах 0—6 и признак одинарной или двойной плотности записи на гибких дисках в разряде 7. Значение может быть:
разряд 7=0 гибкий диск с одинарной плотностью записи
разряд 7=1 гибкий диск с двойной плотностью записи

5) Рекомендуется, чтобы адрес подпрограммы считывания первичного загрузчика ENTRY был расположен выше ячейки 120 в блоке начального загрузчика. Это поможет избежать конфликта с вектором и системной областью монитора при загрузке монитора.

7. ПРОГРАММИРОВАНИЕ СПЕЦИАЛЬНЫХ УСТРОЙСТВ

В этом разделе описываются структура и правила написания программ драйверов для конкретных устройств.

1. Драйвер магнитной ленты: MT.
2. Драйверы гибких дисков: DX и DY.
3. Драйвер перфоленточного устройства ввода-вывода: PC.
4. Драйвер системного терминала: TT.
5. Драйвер фиктивного устройства: NL.
6. Драйвер расширенной памяти: VM.
7. Драйвер логического диска: LD.

7.1. Драйвер магнитной ленты (MT)

Магнитная лента — это устройство файловой структуры с последовательным доступом к данным. Драйвер магнитной ленты системы ФОДОС-2 поддерживает файловую структуру, совместимую с метками магнитной ленты и форматом ленты. Формат ленты предоставляет пользователю возможности доступа к контроллеру ленты, не принимая во внимание специфику устройства.

В системе ФОДОС-2 имеется два типа драйверов магнитной ленты: аппаратный драйвер (MTHD.SYS) и драйвер файловой структуры (MT.SYS). Команда SET изменяет характеристики драйверов: количество дорожек, плотность записи и соответствие привода ленты. Эти команды применимы ко всем приводам определенного контроллера.

Драйвер MT поддерживает до 8 приводов ленты на одном контроллере. Рекомендуется использовать драйвер файловой структуры (если специальные обстоятельства не заставят использовать аппаратный драйвер), таким образом, только драйверы файловой структуры могут взаимодействовать с вспомогательными программами системы ФОДОС-2. В последующих пунктах описываются эти драйверы.

В этом разделе используется несколько присущих магнитной ленте сокращений: BOT — для начала ленты; EOT — для физического конца ленты; LEOT — для логического конца ленты и EOF — для конца файла. LEOT состоит из метки EOF1, которая включает один маркер ленты, за которым следуют два маркера ленты.

7.1.1. Драйвер магнитной ленты файловой структуры содержит аппаратный драйвер, описанный в п. 7.1.2, и модули файловой структуры. Модуль файловой структуры, который предназначен для работы с любым драйвером магнитной ленты, позволяет драйверу принимать запросы с файловой структурой. В драйвере магнитной ленты файловой структуры можно использовать аппаратные команды. Драйвер магнитной ленты файловой структуры называется MT.SYS. Поставляемые версии этого драйвера поддерживают приводы ленты 0 и 1. Во время генерации системы можно добавить поддержку для большего числа приводов (2—7).

Лента, содержащая два файла, имеет следующий формат:
VOL1 HDR1∗данные∗EOF1∗HDR1∗данные∗EOF1∗∗∗
VOL1, HDR1, EOF являются метками ленты. Конструкция «∗» является маркером ленты.

7.1.1.1. Поиск номера файла

Драйвер файловой структуры осуществляет поиск файлов на ленте, взяв за основу последовательный номер файла на ленте.

1. Когда последовательный номер указанного файла больше номера файла, на котором позиционирована лента, драйвер перематывает ленту вперед. Например, если лента в данный момент позиционирована у файла номер 1, а желаемый номер — 2, лента перематывается вперед от маркера ленты после файла номер 1 к маркеру ленты в начале файла номер 2.

2. Когда последовательный номер указанного файла меньше, чем номер файла, на котором позиционирована лента, драйвер оптимизирует время поиска посредством продвижения ленты вперед или назад, в зависимости от местонахождения файла. Практически, драйвер почти всегда перематывает ленту в начало, а затем осуществляет поиск. Например, пусть номер файла, на котором позиционирована лента, — 2, а нужный файл имеет последовательный номер — 1. Лента перематывается к началу тома. Затем она перематывается вперед к маркеру ленты у начала файла номер 1. Для другого примера допустим, что номер файла, на котором позиционирована лента, равен 9 и последовательный номер указанного файла равен 6. Лента перематывается к началу тома и производит поиск в прямом направлении.

Если удалить драйвер посредством команды UNLOAD или программного запроса .RELEAZE, позиция файла теряется. В этой ситуации лента перематывается назад до тех пор, пока аппаратный драйвер не найдет BOT или метку, от которой он может установить позицию ленты.

7.1.1.2. Поиск по имени файла

Драйвер файловой структуры может отыскивать файлы на ленте, взяв за основу имя файла. Подпрограмма для поиска имен файлов использует алгоритм, который позволяет драйверу узнавать имена и типы файлов из других операционных систем. Драйвер использует поле идентификации файла, транслируя содержимое в узнаваемое имя файла. Это имя файла согласуется с именем файла, записанным в формате RADIX-50. Формат следующий: FILNAM.TYP

где FILNAM — допустимое в системе ФОДОС-2 имя файла, состоящее из шести символов (недостающее число символов в имени файла справа заполняется пробелами, если необходимо);

TYP — тип файла, состоящий из трех символов (недостающее число символов в типе файла справа заполняется пробелами, если необходимо).

Алгоритм, который использует драйвер, позволяет системе ФОДОС-2 считывать и использовать ленты, записанные под управлением ранних версий системы.

7.1.1.3. Программные запросы

В следующих подпунктах описывается, как функционируют запросы для магнитной ленты.

7.1.1.3.1. Программный запрос .ENTER записывает метку заголовка файла HDR1 и маркер ленты на ленту, после чего лента позиционируется после маркера ленты. Запрос инициализирует несколько внутренних таблиц, включая элементы для последнего записанного блока и текущего номера блока. (Последний блок или файл на ленте всегда считается только что записанным). Информация для внутренних таблиц и элементов для последнего записанного блока верна, если запрос .SPFUN выполняется по этому каналу. Файлы, открытые запросом .ENTER, не имеют выполняемых в них специальных функций, кроме того случая, когда должен записываться блок нестандартного размера (который не равен 256 словам). Для записи нестандартного блока необходимо открыть файл запросом .ENTER; затем выдать запрос записи .SPFUN. После завершения операции необходимо закрыть файл запросом .CLOSE. Если должен выполняться просмотр файла, необходимо открыть ленту запросом .LOOKUP нефайловой структуры. Табл. 12 показывает возможные значения аргумента SEQNUM для запросов .ENTER.

Программный запрос .ENTER имеет следующий формат:
.ENTER AREA,CHAN,DBLK,SEQNUM

Таблица 12

Аргумент SEQNUM	Имя файла	Предпринятое действие	Положение ленты
1	2	3	4
Больше 0	Задано	Устанавливается последовательный номер файла и выполняется запрос .ENTER	Файл найден: готова к записи. Файл не найден: у логического конца ленты LEOT (LEOT отличается от физического конца ленты)
0	Задано	Лента перематывается, на ней ищется имя указанного файла. Если файл найден, выдается сообщение об ошибке, если нет, указанный файл записывается	Файл найден: перед файлом. Файл не найден: готова к записи
—1	Задано	Лента позиционируется у LEOT, и файл записывается	Готова к записи
—2	Задано	Лента перематывается, на ней ищется указанное имя файла. Указанный файл записывается на место найденного файла или на место LEOT (что встретится раньше)	Готова к записи
0	Не задано	Выполняется запрос .LOOKUP нефайловой структуры	Лента перемотана

Запрос .ENTER возвращает ошибки, показанные в табл. 13.

Таблица 13

Код байта 52	Значение
1	2
0	Канал используется
1	Устройство заполнено. Во время записи HDR1 был обнаружен EOT. Лента позиционируется у первого маркера ленты, который следует за последней меткой EOF1 на ленте
2	Устройство уже используется. Магнитная лента имеет открытый файл на указанном приводе
3	Файл существует, его удаление невозможно

1	2
4	Файл с указанным номером не найден. Лента позиционируется так же, как и в случае, когда устройство заполнено
5	Ошибка недопустимого аргумента. Аргумент SEQNUM не должен принимать значения в диапазоне от —3 до —32767 или не указано имя файла для .ENTER

Запрос .ENTER выдает невосстановимую ошибку справочника во время записи файла.

7.1.1.3.2. Программный запрос .LOOKUP вызывает поиск и считывание указанной метки HDR1. После этого запроса лента позиционируется перед первым блоком данных файла. Табл. 14 показывает значение последовательного номера файла для запроса .LOOKUP, который имеет следующий формат:

.LOOKUP AREA,CHAN,DBLK,SEQNUM

Описание аргументов см. в [1].

Таблица 14

Аргумент SEQNUM	Имя файла	Предпринятое действие	Положение ленты
1	2	3	4
0	Не задано	Выполняется .LOOKUP нефайловой структуры	Перематывается
—1	Не задано	Выполняется .LOOKUP нефайловой структуры	Не перематывается
Больше 0	Не задано	Выполняется .LOOKUP файловой структуры для файла с указанным номером	Файл найден: готова считывать первый блок данных. Файл не найден: на месте LEOT
0	Задано	Лента перематывается в начало, затем имя файла используется для выполнения .LOOKUP файловой структуры	Файл найден: готова считывать первый блок данных. Файл не найден: на месте LEOT
—1	Задано	Выполняется .LOOKUP файловой структуры для файла с указанным именем. Лента не перематывается	Файл найден: готова считывать первый блок данных. Файл не найден: на месте LEOT

1	2	3	4
Больше 0	Задано	Лента позиционируется у файла с указанным номером, выполняется запрос .LOOKUP файловой структуры. Если файл не найден, то выдается сообщение об ошибке	Файл найден: готова считывать первый блок данных. Файл не найден: на месте LEOT

Если канал открыт посредством запросов .LOOKUP нефайловой структуры (имя файла не задано, последовательный номер файла 0 или —1), .READ, .READC, .READW, необходимо использовать счетчик слов, равный размеру физического блока на ленте; запросы .WRITE, .WRITC и .WRITW используют счетчик слов для определения размера блока на ленте. Это условное обозначение используется вместо использования в виде размера блока по умолчанию (512 байт) и выполнения блокировки и деблокировки. Этот запрос почти идентичен считыванию или записи по запросу .SPFUN, который не сообщает об ошибках (BLK=0). Блок ошибки и блок состояния не должны попадать в область свопинга USR.

Запрос .LOOKUP возвращает ошибки, показанные в табл. 15.

Таблица 15

Код байта 52	Значение
1	2
0	Канал используется
1	Файл не найден. Лента позиционируется после первого маркера ленты, следующего за последним EOF1 на ленте
2	Устройство используется. Магнитная лента уже имеет открытый файл
5	Ошибка недопустимого аргумента. Аргумент SEQNUM принимает значения в диапазоне от —2 до —32767. Запрос .LOOKUP для аппаратного драйвера должен иметь положительное значение аргумента SEQNUM

Запрос .LOOKUP выдает невосстановимую ошибку справочника таким же образом, как и запрос .ENTER.

7.1.1.3.3. Программные запросы .READX

Термин .READX/.WRITX относится к следующей группе программных запросов: .READ, .READC, .READW, .WRITE, .WRITC, .WRITW.

Запросы .READX считывают данные с магнитной ленты блоками по 512 байтов каждый. Эта группа запросов описывается здесь для файлов, открытых запросами .ENTER и .LOOKUP файловой структуры. В дополнение к этому описанию имеются описания .READX и .WRITX, соответствующие запросам .LOOKUP нефайловой структуры (см. п.п. 7.1.2.11 и 7.1.2.12).

Если запрос выдается менее, чем для 512 байтов, драйвер считывает указанное число байтов. Если запрос выдан более, чем для 512 байтов, драйвер выполняет запрос с многочисленными 512-байтовыми запросами (последний запрос может быть менее, чем для 512 байтов). Запросы .READX действительны в файле, открытом посредством запроса .LOOKUP. Они также действительны в файле, открытом посредством запроса .ENTER, предусмотренного для того, чтобы запрошенный номер блока не превышал номер последнего записанного блока (возвращается код 0). Если считывается маркер ленты, подпрограмма повторно позиционирует ленту так, что следующий запрос опять вызывает считывание маркера ленты. Когда запрос .CLOSE выдается для файла, открытого запросом .ENTER, лента не позиционируется после последнего записанного блока. Это приводит к потере информации, когда программа выдает запрос считывания для блока, записанного последним, и не может повторно сосчитать последний блок, тем самым позиционируя ленту в конце данных.

Основные требования для номеров блока:

1. .READX: когда .LOOKUP используется (для поиска файла) с этим запросом, драйвер пытается позиционировать ленту у блока с указанным номером. Когда это невозможно, выдается код ошибки 0 (код EOF), и лента позиционируется после последнего блока в файле.

2. .WRITEX и .READX: во введенном файле выполняется проверка для определения, является ли запрошенный блок последним блоком в файле. Если является, лента не перематывается, и выдается код ошибки 0.

Формат запроса .READX следующий:

.READX AREA,CHAN,BUF,WCNT,BLK[,CRTN]

В табл. 16 приведены ошибки, которые возвращают запросы .READX.

Таблица 16

Код байта 52	Значение
1	2
0	Попытка считывания после маркера ленты или образован блок, который слишком велик
1	В канале возникла невосстановимая ошибка
2	Канал не открыт

7.1.1.3.4. Программные запросы .WRITX записывают данные на магнитную ленту в блоки по 512 байтов. Если запрос выдан для менее, чем 512 байтов, драйвер осуществляет запись 512-ти байтов, начиная с адреса буфера. Если запрос выдан для более, чем 512 байтов, драйвер выполняет многочисленные 512-байтовые передачи.

Запросы .WRITX действительны в файле, открытом посредством .ENTER или .LOOKUP нефайловой структуры.

Запросы .WRITX имеют следующий формат:

.WRITX AREA,CHAN,BUF,WCNT, BLK[,CRTN]

В табл. 17 приведены ошибки, которые возвращают запросы .WRITX.

Таблица 17

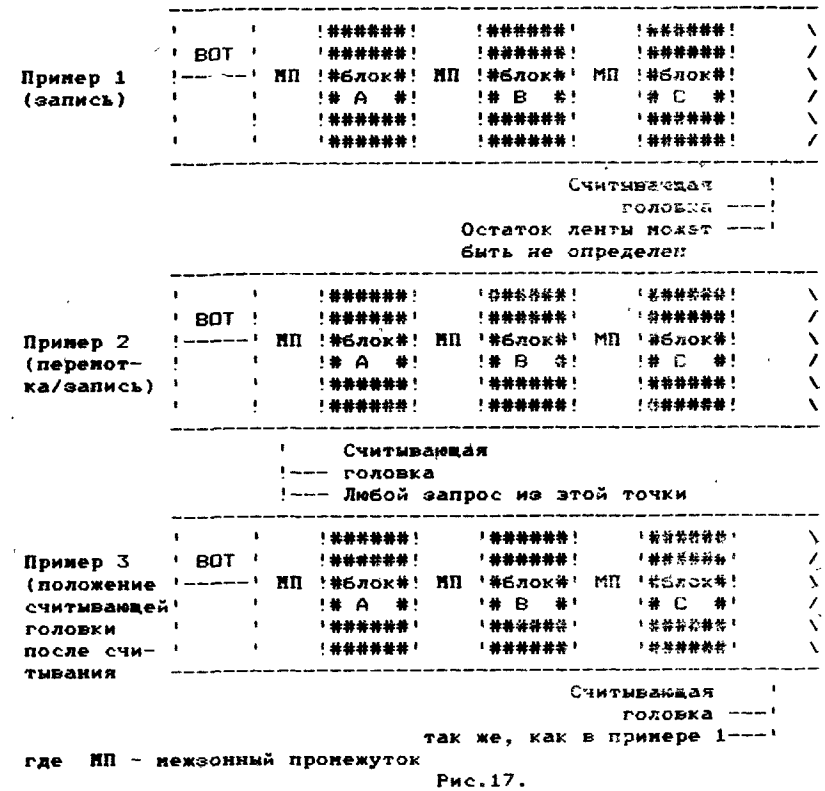
Код байта 52	Значение
1	2
0	Конец ленты. Это значит, что данные не были записаны, но предыдущий блок действителен. Или же, если номер блока слишком велик
1	В канале возникла аппаратная ошибка
2	Канал не открыт

После операции записи остальная лента может оставаться неопределенной (см. рис. 17).

В примере 1 на рис. 17 блоки A, B и C записаны на ленту считывающей головкой, позиционированной в межзонном промежутке (МП) сразу же после блока C. Любая операция в прямом направлении на приводе ленты, исключая команды записи (т. е. запись, удаление или запись межзонного проме-

жутка или запись маркера ленты), приводит к неопределенным результатам вследствие аппаратных ограничений.

В примере 2 на рис. 17 считывающая головка позиционирована у BOT после операции перемотки так, что операции считывания могут считывать блоки A, B и C. Считывающая головка позиционирована как показано в примере 3. Выполняется условие, и действуют все ограничения, рассмотренные в примере 1.



где МП - межзонный промежуток

Рис.17.

7.1.1.3.5. Программные запросы .DELETE и .RENAME недопустимы в операциях с магнитной лентой, и любая попытка их выполнения приводит к коду недопустимой операции (код 2), который возвращается в байт 52.

7.1.1.3.6. Программный запрос .CLOSE действует тремя различными способами в зависимости от того, как был открыт файл.

1. Когда файл открыт запросом .ENTER, файл закрывается записью маркера ленты, метки EOF1 и еще более, чем трех маркеров ленты. В этой операции лента позиционируется в левом положении, как раз перед вторым маркером ленты, у LEOT. Остаток ленты уже не считывается.

2. Когда файл открыт запросом .LOOKUP файловой структуры, лента позиционируется после маркера ленты, за которым следует метка EOF1 для этого файла.

3. Когда файл открыт запросом .LOOKUP нефайловой структуры, никакое действие не предпринимается, и канал освобождается.

Запрос .CLOSE имеет следующий формат: .CLOSE CHAN

Этот запрос выдает невосстановимую ошибку справочника, если обнаружена неисправность. Ошибка может быть исправлена запросом .SERR.

7.1.1.3.7. Программный запрос .SPFUN .SPFUN может выполнять асинхронные операции справочника без USR, что делает его полезным при длительных поисках на ленте. Особенно он полезен для программистов в системе мультизаданий, не имеющих возможности ожидать длительных поисков на ленте, которые могут возникнуть во время запросов .ENTER и .LOOKUP. Он также очень полезен для пользователей в мониторах FB и XM, которые не хотят блокировать USR. Этот запрос позволяет запрашивать .ENTER и .LOOKUP после того, как .LOOKUP нефайловой структуры присваивает канал драйверу магнитной ленты. Если этот запрос выдается для канала, который не был открыт запросом .LOOKUP нефайловой структуры, возникают непредсказуемые результаты. Запрос .SPFUN имеет следующий формат:

.SPFUN AREA,CHAN,#—20.,BUF,,BLK

где —20. — код для асинхронного запроса справочника;

Таблица 18

Слово	Значение
1	2
0—2	Имя файла в RADIX-50
3	Один из следующих кодов: 3 — для .LOOKUP 4 — для .ENTER
4	Значение последовательного номера файла. См. соответствующие разделы для .LOOKUP или .ENTER для завершения информации по интерпретации этого значения
5—6	Зарезервированы

BUF — адрес 7-словного блока с форматом, приведенным в табл. 18.

BLK — адрес 4-словного блока состояния и ошибки, используемого для возвращения ошибок .LOOKUP и .ENTER, которые обычно передаются в байт 52; этим запросом используется только первое слово BLK; другие три слова сохраняются для дальнейшего использования и должны быть 0; когда первое слово BLK 0, информация об ошибке не возвращается; этот блок всегда должен отображаться, когда программа выполняется в среде расширенной памяти.

Ниже приведен пример программирования.

```
.TITLE пример работы асинхронного справочника
                .ENABLE LC          ; печатать строчными буквами
                .NLIST  BEX         ; не запоминать текст строки
.MCALL .LOOKUP, .SPFUN, .CLOSE, .PRINT, .EXIT
; определения
        ASYREQ=     —20.            ; асинхронный запрос
        LOOKUP=     3               ; код LOOKUP для асинхронного запроса
        ENTER=      4               ; код ENTER для асинхронного запроса
        CHAN=       0               ; используется канал 0
        FNF=        1               ; 1=ошибка «Файл не найден»
        FSN=        0               ; использовать 0 как последовательный номер файла
; пример допускает, что драйвер магнитной ленты загружен
START:  .LOOKUP#AREA,#CHAN,#NFSBLK,#0
                                    ; открыть канал для следующего запроса
        BCS     LOOKER              ; ветвление, если обнаружена ошибка
.SPFUN #AREA,#CHAN,#ASYREQ,#COMBLK,#ERRBLK
                                    ; выполнять LOOKUP
        BCC     FILFND              ; ветвление, если файл найден
        CMP     #FNF,ERRBLK         ; ошибка «Файл не найден»?
        BEQ     NOTFND              ; ветвление, если да
        MOV     #ASYERR,R0          ; нет, другая ошибка
        BR      CLOSE
```

```
LOOKER: MOV     #LOOERR,R0 ;ошибка «NFS LOOKUP
                           ;FAILED»
        BR      CLOSE
FILFND: MOV     #OK,R0     ;сообщение об успешном
                           ;исходе
        BR      CLOSE
NOTFND: MOV     #FNFERR,R0 ;сообщение о том, что
                           ;файл не найден
CLOSE:  .PRINT             ;печать ошибки, указы-
                           ;ваемой R0
        .CLOSE  #CHAN      ;очистка ...
        .EXIT              ;возврат в монитор
  ;область данных
AREA:   .BLKW   5          ;блок аргументов EMT
NFSBLK: .RAD50  /MT/       ;использ-ся для открытия
        .WORD   0          ;магнитной ленты в слу-
        .WORD   0          ;чае нефайловой струк-
                           ;туры
        .WORD   0
COMBLK:.RAD50   /FILNAMTYP/ ;имя файла, который не-
                           ;обходимо найти
        .WORD   LOOKUP     ;код асинхронной опера-
                           ;ции для LOOKUP
        .WORD   FSN        ;последовательный но-
                           ;мер файла для LOOKUP
        .WORD   0,0        ;зарезервировано (долж-
                           ;но быть 0)
ERRBLK: .WORD   1          ;установка первого слова
        .WORD   0,0,0      ;не 0, так здесь возвра-
                           ;щаются ошибки
  ;сообщения
LOOERR: .ASCIZ/NON—FILE—STRUCTURED LOOKUP FAILED/
OK:     .ASCIZ /FILE FOUND, LOOKUP SUCCESFUL/
FNFERR: .ASCIZ /FILE NOT FOUND/
ASYERR: .ASCIZ /ERROR IN ASINCHRONOUS REQUEST/
        .EVEN
        .END   START
SYMBOL   TABLE
AREA     000130R ERRBLK   000170R LOOKUP =000003
ASYERR   000317R FILFND   000104R NFSBLK   000142R
ASYREQ   177754  FNF      000001  NOTFND   000112R
CHAN    =000000  FNFERR   000300R OK       000242R
CLOSE    00011GR FSN     =000000  START    000000R
COMBLK   000152R LOOERR   000200R ...V1    =000003
```

```
ENTER   =000004  LOOKER =00007GR...V2    =000027
  .ABS.    000000  000
           000354  001
ERRORS DETECTED:  0
VIRTUAL MEMORY USED: 9472 WORDS (37 PAGES)
DINAMIC MEMORY AVAILABLE FOR 72 PAGES
,SSM012/L:TTM=SSM102
```

7.1.1.4. Выдача вызовов аппаратного драйвера с модулем файловой структуры

Драйвер магнитной ленты предназначен для выполнения двух различных типов доступа: файлово-ориентированный доступ (в этом случае магнитная лента подобна диску; такой доступ делает магнитную ленту малозависимым устройством) и доступ к аппаратным командам, таким, как считывание, запись, пропуск и т. д..

Когда драйвер произвольной магнитной ленты использует файлово-ориентированные команды, он следит за последовательным номером файла, на котором позиционирована лента, т. е. он может оптимизировать движение ленты во время поиска файла. Когда драйвер выбирает данные из файла на магнитной ленте, используя запросы .READX/.WRITX, он фиксирует как номер текущего блока, так и номер последнего считанного блока. Аргумент номера блока может быть использован для моделирования устройства произвольного доступа к данным даже в файлах, открытых посредством .ENTER.

Два метода, описанные выше, могут быть объединены: т. е. можно использовать команды аппаратного драйвера в файле магнитной ленты. Из этого вытекают следующие выводы.

1. Когда принимается первая команда аппаратного драйвера, информация о последовательном номере хранящегося файла и о его номере блока, описанная выше, удаляется и не восстанавливается до тех пор, пока не будет выполнена команда .CLOSE и другая команда открытия файла. .CLOSE перематывает ленту и, в случае, если файл был открыт посредством запроса .ENTER, записывает файл на ленту независимо от команд, выданных с момента открытия файла. Данная лента больше не удовлетворяет стандарту. Когда файл закрыт, драйвер магнитной ленты не может записать размер файла, потому что размер файла потерян для драйвера. На месте размера он записывает 0. Поле последовательного номера файла будет правильным.

2. Исключение из правила, приведенного выше, возникает, когда необходимо открыть ленту с файловой структурой и записывать блоки данных, размер которых отличается от стандартного размера блока (512 байт), который используют запросы .WRITX. Драйвер магнитной ленты фиксирует число записанных блоков, метка EOF1 действительна до тех пор, пока не будут использованы другие команды, отличные от команды записи .SPFUN. Если используются другие команды, размер файла потерян.

Рекомендуется выдавать команды .SPFUN для файла магнитной ленты только в случае, описанном выше в пункте 2.

7.1.2. Аппаратные драйверы магнитной ленты принимают только аппаратные запросы. Они применимы в операциях ввода-вывода, где не существует файловой структуры. Любой запрос файловой структуры, который выдается для аппаратного драйвера, приводит к возникновению ошибки ввода-вывода в справочнике монитора. Аппаратный драйвер — это поднабор драйвера магнитной ленты файловой структуры. Аппаратные драйверы используются, если не нужна дополнительная поддержка файловой структуры. Хотя аппаратные драйверы являются частью дистрибутивного пакета, перед использованием их нужно переименовать. Можно использовать ряд команд монитора (см. табл. 19), которые заменяют драйвер файловой структуры MT на аппаратный драйвер MT.

Таблица 19

Команда	Действие
1	2
REMOVE MT	Удаляет драйвер файловой структуры
RENAME/SYS MT.SYS MTFS.SYS	Сохраняет драйвер файловой структуры
RENAME/SYS MTHD.SYS MT.SYS	Создает новый аппаратный драйвер
INSTALL MT	Устанавливает новый драйвер

Аппаратный драйвер выбирается посредством программных запросов .LOOKUP нефайловой структуры, запросов специальной функции .SPFUN и запросов .READ, .READC, .READW, .WRITE, .WRTIC, .WRITW и .CLOSE. Аппаратный драйвер может выполнять операции ввода-вывода в физических блоках, позиционировать ленту и освобождаться от ошибок.

7.1.2.1. Сообщения об исключениях

Те запросы .SPFUN, которые принимаются аппаратным драйвером, сообщают об окончании файла и условиях невосстановимой ошибки посредством байта 52 в области коммуникации системы. Кроме того, они используют аргумент, обычно используемый для номера блока в виде указателя 4-словного блока состояния и ошибки в порядке поступления указательной информации об исключительных условиях. Когда аргумент номера блока равен 0, эта указательная информация не возвращается. Содержимое этих слов является неопределенным, если нет исключительных условий и если разряд переноса не установлен. Блок определяется следующим образом: слова 1 и 2 содержат указательную информацию; слова 3 и 4 сохраняются для дальнейшего использования и должны быть 0. Поэтому в программе необходимо инициализировать слова 3 и 4 только один раз. Система модифицирует слова 1 и 2 только тогда, когда появляются сообщения об исключениях.

Указательная информация, возвращенная в первом слове для условия «Конец файла» (EOF), показана в табл. 20. (Разряд переноса установлен, байт 52 равен 0.)

Таблица 20

Первое слово восьмеричный код	Значение
1	2
1	Лента только перед EOF (маркер ленты обнаружен)
2	Лента только перед EOT (маркер ленты не обнаружен)
3	Лента перед EOT и EOF (маркер ленты обнаружен)
4	Лента перед BOT (маркер ленты не обнаружен)

Когда маркер ленты встречается во время операции пропуска, количество непропущенных блоков возвращается в слове 2. EOT, маркер ленты и BOT возвращаются аппаратным драйвером как EOF.

Указательная информация, возвращенная в первом слове в случае невосстановимой ошибки, показана в табл. 21. (Разряд переноса установлен, а байт 52 равен 1.)

Аппаратный драйвер выдает невосстановимую ошибку, если он получает запрос иной, чем .LOOKUP нефайловой

структуры, .CLOSE или любой запрос .SPFUN, неопределенный для аппаратного драйвера.

Когда программа выполняется в XM-среде, блок состояния для сообщения об ошибках должен каждый раз отображаться.

Таблица 21

Первое слово восьмеричный код	Значение
1	2
0	Нет дополнительной информации (включает ошибку четности и все другие ошибки, которые не перечислены ниже)
1	Нет привода ленты
2	Контроллер потерял позицию ленты. Когда происходит эта ошибка, необходимо перемотать или вернуть ленту к известной позиции
3	Была выбрана несуществующая память
4	Лента блокирована для записи
5	Последний считанный блок имел больше информации
6	Был считан короткий блок. Второе слово состояния содержит разность между числом запрошенных слов и числом считанных слов

7.1.2.2. Считывание и запись физических блоков

Аппаратный драйвер считывает и записывает блоки любого размера. Запросы для считывания и записи переменного количества слов осуществляются посредством двух кодов .SPFUN.

Запрос .SPFUN имеет следующий формат:

.SPFUN AREA,CHAN,#370,BUF,WCNT,BLK[,CRTN]

где 370 — код функции для операции считывания;

BLK — адрес 4-словного блока состояния и ошибки, используемого для возврата исключительных условий;

CRTN — произвольный аргумент, который указывает подпрограмму завершения, которая будет введена после выполнения запроса.

Этот запрос возвращает ошибки, показанные в табл. 22. Дополнительная указательная информация для этих ошибок возвращается в первых двух словах блока состояния аргумента BLK.

Запрос .SPFUN для записи переменного количества слов в блок имеет формат:

.SPFUN AREA,CHAN,#371,BUF,WCNT,BLK[,CRTN]

где 371 — функциональный код для операции записи.

Таблица 22

Код байта 52	Код первого слова	Указательная информация
1	2	3
EOF значение=0	1	Лента только перед EOF (маркер ленты обнаружен)
	2	Лента только перед EOT (маркер ленты не обнаружен)
	3	Лента перед EOF и EOT (маркер ленты обнаружен)
Невосстановимая ошибка значение=2	0	Нет дополнительной информации (справочная документация для конкретного привода ленты для всевозможных ошибочных состояний)
	1	Нет привода ленты
	2	Контроллер потерял позицию ленты
	3	Выбрана несуществующая память
	4	Лента блокирована для записи
	5	Последний считанный блок имел больше информации
	6	Был считан короткий блок. Второе слово состояния содержит разность между числом запрошенных и числом считанных слов

Этот запрос возвращает ошибки, показанные в табл. 23. Дополнительная указательная информация для этих ошибок возвращается в первых двух словах блока состояния.

Таблица 23

Код байта 52	Код первого слова	Указательная информация
1	2	3
EOF значение=0	1	Лента только перед EOF (маркер ленты обнаружен)
	2	Лента только перед EOT (маркер ленты не обнаружен)
	3	Лента перед EOF и EOT (маркер ленты обнаружен)
Невосстановимая ошибка значение=1	0	Нет дополнительной информации (справочная документация для конкретного привода ленты для всевозможных ошибочных состояний)
	1	Нет привода ленты
	2	Контроллер потерял позицию ленты
	3	Выбрана несуществующая память
	4	Лента блокирована для записи

7.1.2.3. Перемотка ленты вперед и назад

Аппаратный драйвер принимает команды, которые перематывают ленту вперед или назад блок за блоком или до тех пор, пока не будет обнаружен маркер ленты. Когда обнаружен маркер ленты, драйвер сообщает о нем наряду с числом непропущенных блоков. Эти команды могут быть использованы для отработки команды перемотки ленты до ее маркера посредством передачи числа, большего, чем максимальное число блоков на ленте. Лента позиционируется после маркера ленты или последнего пропущенного блока. Два запроса перемотки имеют следующие форматы.

Команда для перемотки вперед на блок имеет следующий формат: .SPFUN AREA,CHAN,#376,,WCNT,BLK[,CRTN]
где 376 — функциональный код для операции перемотки вперед;
WCNT — число блоков для последующей перемотки (не должно превышать 65534);
CRTN — произвольный аргумент, указывающий подпрограмму завершения, которая будет вводиться после выполнения запроса.

Этот запрос возвращает ошибки, показанные в табл. 24. Дополнительная указательная информация для этих ошибок возвращается в двух первых словах блока состояния.

Таблица 24

Код байта 52	Код первого слова	Указательная информация
1	2	3
EOF значение=0	1	Лента только перед EOF (маркер ленты обнаружен)
	2	Лента только перед EOT (маркер ленты не обнаружен)
	3	Лента перед EOF и EOT (маркер ленты обнаружен). Второе слово в блоке состояния содержит число запрошенных для перемотки блоков (WCNT) минус число пропущенных блоков, если обнаружен маркер ленты или BOT. Иначе его значение не определено
Невосстановимая ошибка значение=1	0	Нет дополнительной информации (справочная документация для конкретного привода ленты для всевозможных ошибочных состояний)
	1	Нет привода ленты
	2	Контроллер потерял позицию ленты

Вследствие аппаратных ограничений рекомендуется, чтобы команды перемотки вперед не подавались, если бобина ленты установлена после маркера EOT.

Формат команды перемотки назад на блок следующий:

SPFUN AREA,CHAN,#375,,WCNT,BLK[,CRTN]

где 375 — функциональный код для операции перемотки назад.

Этот запрос возвращает ошибки, показанные в табл. 25. Дополнительная указательная информация для этих ошибок возвращается в первых двух словах блока состояния.

Таблица 25

Код байта 52	Код первого слова	Указательная информация
1	2	3
EOF значение=0	1	Лента только перед EOF (маркер ленты обнаружен)
	2	Лента только перед EOT (маркер ленты не обнаружен)
	3	Лента перед EOF и EOT (маркер ленты обнаружен)
	4	Лента перед BOT (маркер ленты не обнаружен). Второе слово в блоке состояния содержит число запрошенных для перемещения блоков (WCNT) минус число пропущенных блоков, если обнаружен маркер ленты или BOT. В противном случае, его значение не определено
Невосстановимая ошибка значение=1	0	Нет дополнительной информации (справочная документация для конкретного привода ленты для всевозможных ошибочных состояний).
	1	Нет привода ленты
	2	Контроллер потерял позицию ленты

7.1.2.4. Перемотка

Драйвер принимает команду перемотки и перематывает ленту к BOT. Драйвер MT не может принимать другие запросы, пока не будет завершена операция перемотки.

Запрос перемотки имеет следующий формат:

.SPFUN AREA,CHAN,#373,,,BLK[,CRTN]

где 373 — функциональный код для операции перемотки;

CRTN — произвольный аргумент, который указывает подпрограмму завершения, которая будет введена после выполнения запроса.

Этот запрос возвращает ошибки, показанные в табл. 26. Дополнительная указательная информация возвращается в блок состояния.

Таблица 26

Код байта 52	Код первого слова	Указательная информация
1	2	3
Невосстановимая ошибка значение=1	0	Нет дополнительной информации (справочная документация для конкретного привода ленты для всевозможных ошибочных состояний)
	1	Нет привода ленты

7.1.2.5. Перемотка и переход к автономности

Этот запрос такой же, как и перемотка, с тем исключением, что привод ленты берется автономным, а затем перематывается к BOT. Драйвер может принимать команды после начала перемотки.

Запрос перемотки и перехода к автономности имеет следующий формат: .SPFUN AREA,CHAN,#372,,,BLK[,CRTN]
где 372 — функциональный код для операции перемотки и перехода к автономности;
CRTN — произвольный аргумент, указывающий подпрограмму завершения, которая будет введена после выполнения запроса.

Этот запрос возвращает тот же самый код ошибки, и указательную информацию, что и запрос перемотки (см. табл. 26).

7.1.2.6. Запись с расширенным межзонным промежутком

Этот запрос позволяет производить запись на лентах, имеющих дефектные участки. Он идентичен запросу записи, функциональный код которого равен 374. Ошибки аналогичны ошибкам для запроса записи (см. п. 7.1.2.2).

7.1.2.7. Запись маркера ленты

Аппаратный драйвер принимает запрос для записи маркера ленты. Этот запрос имеет следующий формат:
.SPFUN AREA,CHAN,#377,,,BLK[,CRTN]
где 377 — функциональный код для записи маркера ленты.

Этот запрос возвращает ошибки, показанные в табл. 27. Дополнительная указательная информация для этих ошибок возвращается в первых двух словах блока состояния (аргумента BLK).

Таблица 27

Код байта 52	Код первого слова	Указательная информация
1	2	3
EOF значение=0	1	Лента только перед EOF (маркер ленты обнаружен)
Невосстановимая ошибка значение=1	0	Нет дополнительной информации (справочная документация для конкретного привода ленты для всевозможных ошибочных состояний)
	1	Нет привода ленты
	2	Контроллер потерял позицию ленты
	4	Лента блокирована для записи

7.1.2.8. Восстановление ошибок

Любые ошибки, обнаруженные во время операций перемотки, вызывают преждевременное прерывание, и выдается сообщение о невосстановимой (позиционной) ошибке. Если обнаружена ошибка паритета считывания, то драйвер файловой структуры и аппаратный драйвер выполняют следующие операции.

1. Перемотка назад на блок и повторное считывание. Если безуспешно, процедура повторяется до пяти неудачных попыток.

2. Перемотка назад на 5 блоков, перемотка вперед на четыре блока, затем считывание записи.

3. Повторяет вышеуказанные пункты 1 и 2 восемь раз или пока блок не будет успешно считан.

Драйвер выполняет следующие операции при обнаружении ошибок паритета при считывании после записи.

1. Перематывает ленту назад на один блок.

2. Удаляет данные на 76.2 мм ленты и повторно записывает блок. Никогда не делает попыток повторно записать блок на дефектные участки, так как этот блок может быть ненадежен и позднее может вызвать проблемы.

3. Повторяет вышеуказанные пункты 1, 2, если считывание после записи по-прежнему отсутствует. Если таким образом пропускается 7620 мм ленты, выдается сообщение о невосстановимой ошибке.

7.1.2.9. Программный запрос .LOOKUP нефайловой структуры

Аппаратный драйвер принимает запрос .LOOKUP нефайловой структуры, который необходим для открытия канала в

устройстве перед тем, как любые операции ввода-вывода будут выполнены. Он заставляет аппаратный драйвер пометить привод как занятый, и никакой другой канал не может быть открыт для этого привода, пока не будет выдан запрос .CLOSE. Этот запрос имеет следующий формат:

.LOOKUP AREA,CHAN,DBLK,SEQNUM

где SEQNUM — аргумент, который указывает, должна ли перематываться лента; когда этот аргумент равен 0, лента перематывается, когда он равен —1, лента не перематывается.

В табл. 28 показаны ошибки, возвращаемые этим запросом.

Таблица 28

Код байта 52	Значение
1	2
0—1	Запрос не имеет смысла
2	Устройство используется. Выбранный привод уже присоединен к другому каналу
3	Нет привода ленты
4	Обнаружен недопустимый аргумент: обнаружено имя файла или аргумент SEQNUM отличен от 0 или —1

7.1.2.10. Программный запрос .CLOSE

Аппаратный драйвер принимает запрос .CLOSE и заставляет драйвер отмечать привод как свободный для использования. Этот запрос имеет следующий формат: .CLOSE CHAN

7.1.2.11. Программные запросы .WRITX нефайловой структуры

Аппаратный драйвер принимает запросы .WRITX, которые записывают различное количество слов в блок на ленте. Поле номера блока игнорируется. Эти запросы имеют следующий формат: .WRITX AREA,CHAN,BUF,WCNT[,,CRTN]

Эти запросы возвращают ошибки, показанные в табл. 29. Дополнительная указательная информация отсутствует.

Таблица 29

Код байта 52	Значение
1	2
0	Маркер ленты EOT не был обнаружен
1	В канале возникла невосстановимая ошибка
2	Канал не открыт

7.1.2.12. Программные запросы .READX нефайловой структуры

Эти запросы считывают различное количество слов из блока на ленте. Они игнорируют маркер EOT и сообщают о конце файла, когда считан маркер ленты. Поле номера блока игнорируется. Запросы имеют следующий формат:

.READX AREA,CHAN,BUF,WCNT[,,CRTN]

Эти запросы возвращают ошибки, показанные в табл. 30. Дополнительная указательная информация отсутствует.

Таблица 30

Код байта 52	Значение
1	2
0	Попытка считывания после маркера ленты или при создании блока, который слишком велик
1	В канале возникла невосстановимая ошибка
2	Канал не открыт

7.2. Драйверы гибких дисков (DX и DY)

Запросы .SPFUN позволяют считывать и записывать абсолютные секторы на гибких дисках. (В дальнейшем гибкие диски с одинарной плотностью записи будем называть просто гибкими дисками, в отличие от гибких дисков с двойной плотностью записи данных.) Драйвер DY принимает дополнительный запрос .SPFUN для определения размера (в 256-словных блоках) тома, установленного на указанном приводе. В гибких дисках с двойной плотностью записи секторы имеют размер 128 слов. Система ФОДОС-2 обычно считывает и записывает их группами из двух секторов. В гибких дисках секторы имеют размер 64 слова. Система ФОДОС-2 считывает и записывает их группами по четыре сектора. Секторы могут выбираться индивидуально посредством запроса. .SPFUN. Формат запроса следующий:

.SPFUN AREA,CHAN,FUNC,BUF,WCNT,BLK,CRTN

где FUNC — код функции, которая будет выполняться; коды и функции приведены в табл. 31.

BUF — для функциональных кодов 377, 376, 375 является ячейкой 129-словного буфера (для гибких дисков с двойной плотностью записи) или 65-словного буфера (для гибких дисков); первое слово буфера, слово признака, обычно, установлено 0;

Таблица 31

Код	Функция
1	2
377	Считывание физического сектора
376	Запись физического сектора
375	Запись физического сектора с меткой удаленных данных
374	Зарезервировано
373	Определяет размер тома устройства в 256-словных блоках (только DY)

если первое слово установлено 1, осуществляется считывание физического сектора, содержащего метку удаленных данных; фактическая область данных буфера расширяется от второго слова к концу буфера;
BUF — для функционального кода 373 является ячейкой однословного буфера, в котором возвращается размер тома, установленного на указанном приводе; (для гибких дисков возвращается 494; для гибких дисков с двойной плотностью записи возвращается 988)
WCNT — для функций 377, 376, 375 является абсолютным числом дорожек для считывания или записи от 0 до 76;
WCNT — для функции 373 резервируется и должен быть установлен 1;
BLK — для функций 377, 376, 375 является абсолютным числом секторов для считывания или записи с 1 по 26;
BLK — для функции 373 резервируется и должен быть установлен 0.

Гибкие диски должны открываться запросом .LOOKUP нефайловой структуры. Аргументы BUF, WCNT, BLK имеют разные значения, когда они используются гибкими дисками. Ниже приведен запрос, выполняющий синхронное считывание сектора с дорожки 0, сектора 7 в 65-словную область, названную BUFF:

```
.SPFUN #RDLIST,#377,#BUFF,#0,#7,#0
```

Каждый из драйверов, и DX, и DY, может поддерживать два контроллера, и каждый контроллер обслуживает два привода. Например, если драйвер DX создается в процессе генерации системы с поддержкой двух контроллеров, он будет поддерживать 4 устройства: DX0, DX1, DX2, DX3. DX0 и DX1 — это приводы 0 и 1 стандартного гибкого диска с вектором 264 и регистром состояния канала (CSR) 177170. DX2 и DX3 — это приводы 0 и 1 другого контроллера. Процесс вво-

да-вывода может действовать всего один раз, даже если имеется два контроллера. Совмещение ввода-вывода в драйвере недопустимо.

7.3. Драйвер перфоленточного устройства ввода-вывода (PC)

Когда подается команда, лента должна находиться в устройстве ввода, иначе возникает входная ошибка. В этом случае любые операции второго прохода недопустимы при использовании перфоленточного устройства ввода. Например, редактирование и транслирование на устройстве PC: не работает. Когда инициируется второй проход, возникает входная ошибка. Правильная процедура заключается в передаче файла с перфоленты на диск, а затем, выполнение операции в новом файле.

Драйвер PC во время операции ввода заполняет нулями буфер, если кончилась лента во время считывания. При следующем запросе считывания драйверу PC устанавливается разряд EOF и разряд переноса при возврате по окончании ввода-вывода.

7.4. Драйвер системного терминала (TT)

При использовании драйвера TT системный терминал может рассматриваться как периферийное устройство. Необходимо соблюдать следующие обозначения и ограничения:

1. Управляющий символ «^» печатается, когда драйвер готов для ввода.

2. CУ/Z используется для обозначения конца ввода в TT:. После CУ/Z не требуется нажатия <ВК>. Если CУ/Z не печатается, драйвер TT принимает символы, пока счетчик слов запроса ввода не будет исчерпан.

3. CУ/O, напечатанное, когда осуществляется вывод на TT:, вызывает игнорирование всего буфера вывода. (Все символы образуют текущую очередь.)

4. Однократная печать CУ/C во время ввода на TT: вызывает возврат в монитор. Если осуществляется вывод на TT:, для возврата в монитор требуется двухкратная печать CУ/C, если выполняется FB-монитор. Если выполняется SJ-монитор, требуется только однократная печать CУ/C для завершения вывода на TT:.

5. Драйвер TT может использоваться в определенный момент только одним заданием (основным или фоновым) и только для одной функции (ввода или вывода). Коммуникация терминала для задания, не использующего TT:, вообще, не затрагивается.

6. Можно печатать сначала на TT:; символы изымаются из кольцевого буфера ввода перед обращением к клавиатуре. Завершающее CУ/Z может быть также напечатано заранее.

7. Если основной код задания использует TT: для ввода, а подпрограмма завершения выдает запрос .TTYIN, то напечатанные символы будут пропущены по .TTYIN и на TT: непредсказуемо.

8. Если задание посылает данные в TT: для вывода и затем выдает запрос .TTYOUT или запрос .PRINT, вывод из последнего задерживается до завершения драйвером своей передачи. Если операция вывода на TT: начата, когда выходной кольцевой буфер мониторного терминала не пуст (перед окончанием прямой печати), драйвер завершает операцию передачи до того, как содержимое буфера напечатается.

7.5. Драйвер фиктивного устройства NL принимает все запросы считывания и записи. В операциях вывода этот драйвер действует как приемник данных. Когда программа вызывает NL, драйвер сразу же возвращается в монитор, показывая, что вывод завершен. Драйвер не возвращает ошибок и не вызывает прерываний. При операциях ввода NL возвращает немедленную индикацию EOF для всех запросов; данные не передаются. Следовательно, содержимое буфера ввода остается неизменным.

7.6. Драйвер расширенной памяти VM имеет произвольный доступ к расширенной памяти, подобно диску. Команда INITIALIZE используется для инициализации расширенной памяти, зарезервированной для области VM: перед копированием файлов в область, и можно получить справочник файлов в области VM: используя команду DIR. Можно даже копировать мониторы SJ или FB и начальный загрузчик в VM: и работать в системе SJ или FB. Особенности драйвера содержат:

— определение размера памяти во время установки драйвера, но не во время загрузки драйвера;
— адрес базы может быть установлен для любого адреса расширенной памяти; по умолчанию адрес базы драйвера VM в SJ и FB находится на границе 28K слов нижнего предела расширенной памяти; базовый адрес по умолчанию для версии XM драйвера VM находится на границе 128K слов;
— драйвер VM поддерживается как монитором XM, так и мониторами SJ и FB; XM-монитор поддерживает VM как том данных, но не как системный том.

Код установки драйвера VM определяет размер памяти, во время установки драйвера. После определения размера памяти код установки драйвера резервирует всю расширенную память, расположенную над адресом базы драйвера. Драйвер не нуждается в выполнении этой операции каждый раз во время загрузки, вследствие чего ускоряется процесс загрузки драйвера.

Если необходимо изменить размеры памяти, имеющейся в распоряжении драйвера VM, посредством изменения адреса базы, необходимо переместить и переустановить драйвер. Большинство других драйверов необходимо только перезагрузить после команды SET без переустановки. Необходимо также повторно инициализировать VM: после изменения адреса базы, используя команду INIT VM:. Система ФОДОС-2 печатает предупреждающее сообщение: VM—W—REMOVE AND REINSTALL всякий раз, когда требуется изменить адрес базы, по команде: SET VM BASE=N.

Если нежелательно использование VM и резервирование им памяти для собственного использования, имеются несколько вариантов. Можно удалить драйвер с системного диска так, что он не будет установлен во время загрузки системы. Можно установить адрес базы выше верхней границы имеющейся в распоряжении памяти, что будет препятствовать установке драйвера. Или можно поместить команду в файл запуска команды для удаления драйвера VM из данной системы после того, как начальный загрузчик установил его. Иначе код установки драйвера VM будет всегда резервировать расширенную память для своего собственного использования, вследствие чего она становится недоступной для пользовательской программы.

Адрес базы (N), использованный в команде SET VM BASE=N, — это затребованный адрес базы в восьмеричном представлении, деленный на 100 (восьмеричное). Например, используем значение 1600 для установки адреса базы на границе адреса 28K слов или 10000 для установки адреса базы на границе адреса 128K слов; любое другое значение между 1600 и верхней границей физической памяти также допустимо.

В табл. 32 приведены соответствия между размерами памяти K слов и соответствующими значениями N.

Рис. 18 показывает 22-разрядную систему с адресом базы, равным 10000 (128K слов).

Если используется монитор XM, и аппаратура не имеет 22-разрядную адресацию, драйвер по умолчанию устанавли-

Таблица 32

K—слова	N
1	2
28	1600
32	2000
64	4000
96	6000
128	10000
256	20000
512	40000
1024	100000

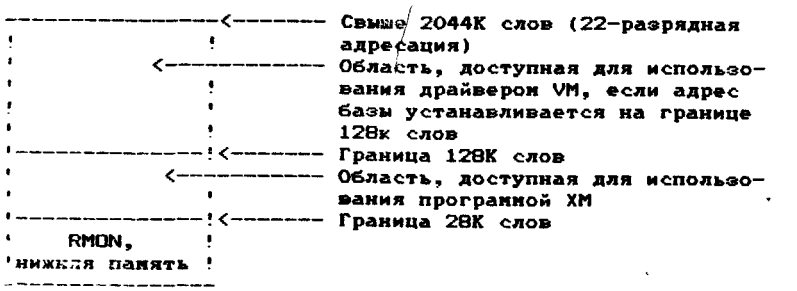


Рис.18

ваться не будет; необходимо заменить адрес базы на более низкое значение перед использованием VM: с системой XM. Можно еще использовать расширенную память и для программы XM, и для тома VM:, но область, указанная для одной будет урезаться областью, занятой другой. Рис. 19 показывает 18-разрядную систему с базовым адресом, установленным VM на 3600 (60K слов).

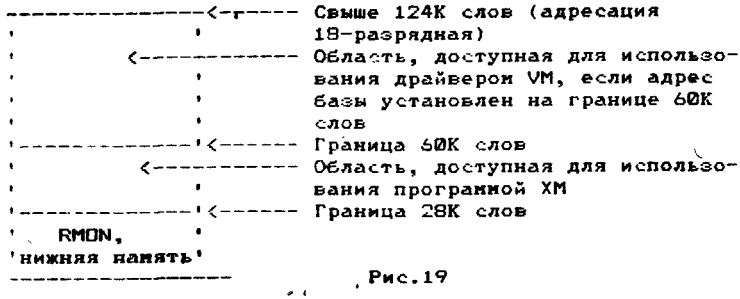


Рис.19

Большинство устройств произвольного доступа к данным, например, диски, не могут обрабатывать более, чем 18-разрядный адрес (128К слов). Если имеется одно из таких устройств и более 128К слов памяти в системе, то установка базы VM: на границе 128К слов гарантирует, что ни одна из программ пользователя не будет создавать буфер данных, который данный диск не сможет считывать или записывать, позволяя в то же время эффективное использование памяти свыше 128К слов драйвером VM.

7.7. Программа установки логического диска (LD) предназначена для определения логических дисков, как части физических дисков. Каждый логический диск определяется файлом. Можно использовать команды клавиатурного монитора и программы работы с файлами для инициализации, копирования и обслуживания логических дисков также, как и физических.

Так как каждый логический диск имеет собственный справочник, то разделяя физический диск на логические диски, можно получить область под справочник большего размера. Установка логических дисков позволяет логически группировать файлы и выполнять некоторые операции с файлами и дисками более быстро.

7.7.1. Выполнение программы установки логического диска

Для вызова LD с системного устройства следует убедиться, что LD установлен и подать с терминала команду после того, как монитор напечатает точку: R LD.SYS<ВК>.

В XM-системе команда будет выглядеть следующим образом:

R LDX.SYS<ВК>

После вызова LD печатает звездочку (*) и ожидает ввода командной строки. Если в это время нажать клавишу <ВК>, то LD печатает номер своей версии. Для выхода из программы LD и передачи управления монитору следует подать команду СУ/С, если LD ожидает ввода командной строки, или дважды СУ/С, если LD выполняет операцию.

Режим работы программы LD задается введением с терминала командной строки.

Формат командной строки следующий: Входспф/прк

где входспф — спецификация файла, определяемого как логический диск;

прк — переключатель (см. табл. 33).

В командной строке можно указать до шести спецификаций входных файлов. Тип файла по умолчанию — .DSK.

7.7.2. Переключатели LD

В табл. 33 приведены переключатели программы LD.

Таблица 33

Переключатель	Функция
1	2
/A:DDD	Назначает имя логическому диску. Используется с переключателем /L
/C	Сравнивает все логические диски, назначенные файлам на установленном в данный момент томе
/L:N	Устанавливает логический диск и связывает его с файлом на диске или удаляет логический диск и отменяет связь его с файлом на диске
/R:N	Запрещает запись на логический диск. Когда используется переключатель /R:N, имеется только доступ к чтению
/W:N	Разрешает запись на логический диск. Когда используется переключатель /W:N, имеется доступ к чтению/записи

7.7.2.1. Переключатель /A:DDD следует использовать с переключателем /L для назначения логического имени логическому диску. Аргумент DDD — логическое имя (от одного до трех символов). Первый символ должен быть буквой, двоеточие после логического имени включается по умолчанию. После назначения логического имени логическому диску, можно обращаться к логическому диску, используя форму LDN: или логическое имя устройства.

Следующая команда назначает логическое имя VOL логическому диску с номером 2 (LD2:), когда он определяется файлом DK:LOGFIL.DSK: *LOGFIL.DSK/L:2/A:VOL

7.7.2.2. Переключатель /C проверяет все назначения логических дисков. При использовании переключателя /C, LD проверяет текущее назначение логическим дискам файлов на томах, которые установлены. Этот переключатель наиболее полезен после перемещения или удаления файлов на томе или удаления тома с устройства. Если файл логического диска перемещен, LD определяет новое расположение так, что можно продолжать использование этого логического диска. Если удален файл логического диска или том, содержащий файл логического диска, не установлен, LD отменяет назначения логического диска. В случае, если том удаляется, отмена назначения — временная.

После операции сжатия по команде монитора SQUEEZE (или с использованием переключателя /S программы DUP, см. [2], [3]) или начальной загрузки система автоматически

выполняет операцию по переключателю /C для модификации назначений логических дисков.

Переключатель /C не должен повторяться в командной строке (должен встречаться не более одного раза).

7.7.2.3. Переключатель /L:N устанавливает логический диск, связывая его с файлом на устройстве или освобождая логический диск с номером N от связи с файлом. Для установки логического диска с номером N используется следующая командная строка: спф/L:N

где спф — спецификация файла, связанного с логическим диском с номером N; файл может размещаться на физическом диске или на другом логическом диске;

N — номер логического диска, связанного с файлом; после его установки, к логическому диску можно обращаться как к LDN:; аргумент N должен принимать целые значения в диапазоне от 0 до 7.

Для освобождения логического диска от связи с файлом следует подать следующую команду: /L:N

Можно установить и удалить в одной командной строке несколько логических дисков. Например, следующая команда связывает логический диск с номером 0 с файлом MYFILE.DSK на DK0:, логический диск с номером 4 с файлом DATFIL.DSK на DY0: и удаляет логический диск с номером 2:

*DK0:MYFILE/L:0,DY0:DATFIL.DSK/L:4,/L:2

Можно также переназначить логический диск с номером N, указав /L:N и другое имя файла.

7.7.2.4. Переключатель /R:N используется для запрещения записи на логический диск. После команды с этим переключателем доступ к логическому диску будет осуществляться только по чтению. Аргумент N — номер логического диска (N — целое, от 0 до 7). Следующая команда устанавливает LD3: для файла JMS.TXT на DY1: и устанавливает запрещение записи:

*JMS.TXT/L:3/R:3

Следующая далее команда устанавливает запрещение записи на LD4:: */R:4

7.7.2.5. Переключатель /W:N используется для разрешения записи на логический диск. После этой команды к логическому диску имеется доступ по чтению/записи. Аргумент N — номер логического диска (N — целое, от 0 до 7). Этот режим выполняется по умолчанию.

Следующая команда устанавливает логический диск

LD5:, связывает его с файлом JMS.DSK на DK0: и разрешает запись на этом логическом диске: *DK0:JMS/L:5/W:5

7.8. Драйвер логического диска (LD) осуществляет поддержку логического диска в системе ФОДОС-2. Драйвер LD принимает запросы ввода-вывода так же, как это делает любой другой драйвер диска. Посредством вставленных таблиц трансляции драйвер LD определяет, какой физический диск и какое смещение начального блока должно использоваться для каждого запроса ввода-вывода LD. Когда соответствующий физический диск и номер блока определены, драйвер LD заносит номер блока и номер привода в элемент очереди ввода-вывода так, что они соответствуют значениям для назначенного физического диска. Затем драйвер LD помещает элемент очереди в очередь ввода-вывода для физического диска так, чтобы мог произойти фактический ввод-вывод.

Кроме действий, описанных выше, драйвер может выполняться как программа. При выполнении драйвер LD допускает использование командных строк интерпретатора командной строки (CSI) и переключателей для инициализации, назначения, проверки, разрешения записи или блокировки записи привода логического диска.

7.8.1. Таблицы трансляции LD

В драйвере LD находятся четыре таблицы трансляции. Первая таблица трансляции начинается с метки HANDLR и содержит одно слово для каждого номера привода LD:, максимум 8. Эта таблица имеет порядковый номер, равный номеру привода LD:, умноженному на 2.

Разряды с 0 по 5 каждого слова в таблице HANDLR содержат порядковый номер в таблице драйвера в RMON для физического устройства, соответствующего номеру привода LD:.

Разряд 6 является признаком, который сигнализирует, что элемент таблицы OFFSET для привода LD: может быть неправильным. Этот разряд устанавливается каждый раз при сжатии тома. Драйвер LD, когда он использует привод LD:, проверяет этот разряд; если разряд установлен, драйвер проверяет элемент таблицы блока OFFSET перед обработкой.

Разряд 7 является признаком, который сигнализирует, что содержимое поля порядкового номера, разряды с 0 по 5, может быть неправильным и будет проверено и исправлено. Драйвер LD устанавливает разряд 7 для всех приводов, если у элемента разряд LDREL$ установлен в CONFG2.

Разряды с 8 по 10 содержат номер привода физического диска, назначенный для привода логического диска.

Разряды 11, 12, и 14 не используются.

Разряд 13 является разрядом блокировки записи. Если он установлен, то привод LD: предназначен только для считывания.

Разряд 15 является разрядом назначения. Если он установлен, то привод LD: является назначенным, если он очищен, то привод не назначен.

Вторая таблица трансляции начинается с метки OFFSET и содержит одно слово для каждого номера привода LD:, максимум 8. Эта таблица имеет порядковый номер, равный номеру привода LD:, умноженному на 2. Каждое слово содержит смещение в блоках от начала назначенного физического диска до начала области на физическом диске, назначенный для номера привода LD:.

Третья таблица трансляции начинается с метки SIZE и содержит одно слово для каждого номера привода LD:, максимум 8. Эта таблица имеет порядковый номер, равный номеру привода LD:, умноженному на 2. Каждое слово содержит размер области в блоках на физическом диске, назначенной для привода логического диска.

Четвертая таблица трансляции начинается с метки NAME и содержит 4 слова для каждого номера привода LD:. Эта таблица имеет порядковый номер, равный номеру привода LD:, умноженному на 8. Первое слово каждого 4-словного элемента содержит двухсимвольное имя физического диска в RADIX-50, назначенное приводу логического диска. Это слово должно быть фактически именем устройства, а не логически присвоенным именем, и не должно иметь номер привода как часть имени. Второе, третье и четвертое слова каждого 4-словного элемента содержат имя и тип файла в RADIX-50, назначенные как логический диск.

7.8.2. Другие разряды, используемые драйвером LD

Драйвер LD использует разряд 4 (LDREL$) в CONFG2, фиксированное смещение монитора 370. Этот разряд устанавливается всякий раз, когда драйвер разгружается или освобождается. Драйвер LD проверяет этот разряд, чтобы убедиться, назначен ли драйвер приводу LD, удаленный из памяти после того, как он был последний раз использован. Если разряд установлен, драйвер LD устанавливает разряд 7 во всех элементах таблицы HANDLR, затем очищает разряд LDREL$. Когда драйвер LD начинает обработку запроса ввода-вывода, он проверяет разряд 7 для запрошенного привода LD:. Если разряд 7 установлен, драйвер LD проверяет, чтобы драйвер для диска, назначенного этому номеру привода

LD: находился в памяти, затем очищает разряд. Драйвер LD проверяет и очищает разряд 7 для привода, только когда запрос ввода-вывода посылается для этого привода. Проверка только при абсолютной необходимости гарантирует, что драйвер LD не будет тратить время на сравнение приводов, которые никогда не могут быть использованы в индивидуальной программе пользователя.

7.8.3. Специальный переключатель LD (/$)

Когда подается команда клавиатурного монитора MOUNT для назначения привода логического диска, MOUNT строит командную строку для назначения привода логического диска, размещает команду в цепной области и сцепляется с LD.SYS.MOUNT включает в командную строку переключатель /L, который определяет назначение логического привода. MOUNT включает также дополнительно переключатель /$.

LD.SYS принимает командную строку и использует код переключателя /L для назначения привода логического диска. Затем, если переключатель /$ был включен в командную строку, LD проверяет, находится ли в памяти драйвер устройства, затребованный этим логическим приводом. Если драйвер устройства не загружен, LD вызывает команду LOAD для этого драйвера, помещает команду в цепную область и передает команду обратно в KMON для выполнения. Таким образом, любые запрошенные драйверы для логических приводов, назначенных по команде MOUNT, являются автоматически загруженными. Хотя переключатель /$ первоначально предназначен для использования в MOUNT, можно использовать /$ с /L, если необходимо самостоятельно выполнить LD.SYS для назначения логических приводов.

8. СООБЩЕНИЯ СИСТЕМНОМУ ПРОГРАММИСТУ

Все сообщения системы ФОДОС-2 приведены в книге 6 глава «Сообщения системы».

ПЕРЕЧЕНЬ ССЫЛОЧНЫХ ДОКУМЕНТОВ

1. Операционная система ФОДОС-2
   Системная макробиблиотека.
   Руководство программиста.
2. Операционная система ФОДОС-2
   Командный язык системы.
3. Операционная система ФОДОС-2
   Программы работы с файлами.
   Руководство оператора.

МОНИТОР РАСШИРЕННОЙ ПАМЯТИ
РУКОВОДСТВО ПРОГРАММИСТА

1. ОПРЕДЕЛЕНИЯ

Ниже дано описание терминов, встречающихся в настоящем документе.

Адресное пространство — набор адресов ячеек, имеющихся в наличии при работе в определенном режиме процессора (внутреннем или режиме пользователя).

Блок — наименьшая единица памяти (32 слова), с которой может работать диспетчер памяти (ДП).

Виртуальный адрес — 16-разрядный адрес, сформированный процессором. Термин «виртуальный» применяется потому, что адрес подлежит преобразованию (расширению) с помощью содержимого регистра адреса страницы (РАС) ДП.

Область — непрерывный сегмент физической памяти.

Динамическая область — область физической памяти, используемая в процессе работы программы.

Статическая область — область, расположенная в нижней области (28К) физической памяти; имеет идентификатор ноль и содержит базовый сегмент программы; создается при загрузке программы и не может изменяться программными запросами.

Нижняя область памяти — физическая область памяти, имеющая адрес в диапазоне 0—157776 (восьмеричное).

Отображение — процесс, при котором виртуальному адресу ставится в соответствие ячейка физической памяти. Процесс выполняется диспетчером памяти.

Пространство виртуальных адресов программы (PVAS) — область адресации, выбираемая программой и имеющая размер 32К слов.

Окно — часть PVAS, начальный адрес которой кратен 4К слов. Размер окна может изменяться от 32 слов до 28К слов.

Расширенная память — физическая память, имеющая адреса выше 177776 (восьмеричное).

Режим — диспетчер памяти может работать в одном из двух режимов (внутреннем или режиме пользователя). Для каждого режима ДП имеет отдельный набор регистров РАС/РПС — из восьми пар РАС/РПС каждый. Режим определяется разрядами (15,14) слова состояния процессора (ССП).

РАС (регистр адреса страницы) — регистр диспетчера памяти, содержащий базовый адрес или константу распределения. ДП имеет 16 РАС: 8 для внутреннего режима и 8 — для режима пользователя.

РПС (регистр признака страницы) — регистр ДП, содержащий информацию о странице (длину страницы, направление расширения и ключ доступа). ДП имеет 16 РПС (8 — для внутреннего режима и 8 — для режима пользователя).

Страница — одна из 8 секций виртуального адресного пространства в 32К слов, начальный адрес каждой страницы кратен 4К слов. Размер страницы изменяется от 0 до 4096 (десятичное) слов. Номера страниц — от 0 до 7.

Физический адрес — адрес, формируемый ДП для выборки определенной ячейки памяти.

Пространство логических адресов программы (PLAS) — область памяти, необходимая для работы программы. Обычно она ограничена до 32К слов областью виртуальной адресации, однако может быть расширена посредством создания оверлейной структуры или использования возможностей расширенной памяти, предоставляемых монитором.

БООБ (блок определения области) — блок размером 3 слова в программе пользователя для определения области и хранения информации о ней. Используется как область связи между программой и монитором (см. также БООК).

БООК (блок определения окна) — блок размером 7 слов в программе пользователя. Используется для определения окна и хранения информации о нем. Как и БООБ, служит для связи программы с монитором. Значения, записываемые программой в БООБ или БООК, определяют или модифицируют запрашиваемую операцию.

УБОБ (управляющий блок области) — блок размером 3 слова в смешанной области задания. Используется монитором при отображении.

ОСА (очередной свободный адрес) задания — виртуальный старший адрес задания, округленный в сторону увеличения до 4К.

ВСА (виртуальный старший адрес) — наибольший виртуальный адрес, используемый оверлеями расширенной памяти, округленный до 32 (десятичное), минус 2 (две младшие цифры восьмеричного адреса всегда 76).

XM .SETTOP — XM — особенность программного запроса SETTOP; предоставляет заданиям пользователя возможность получения дополнительной памяти.

2. НАЗНАЧЕНИЕ И УСЛОВИЯ ПРИМЕНЕНИЯ ПРОГРАММЫ

Монитор расширенной памяти предоставляет пользователю возможность использовать память размером свыше 32К слов (см. раздел 4). Он представляет собой расширение монитора FB. Программы, разработанные в системе мониторов SJ и FB, будут выполняться и в системе под управлением монитора XM, однако, программы, которые должны использовать монитор XM, не будут выполняться в системе под управлением мониторов SJ и FB.

Для использования монитора расширенной памяти в операционной системе ФОДОС-2 необходимо иметь следующие аппаратные и программные компоненты:

1) монитор XM и драйверы для XM;
2) память емкостью не менее 32К слов;
3) диспетчер памяти;
4) блок расширенной арифметики.

3. ХАРАКТЕРИСТИКИ ПРОГРАММЫ

При обработке программ пользователя монитор осуществляет один из двух типов отображений: виртуальный или привилегированный. В соответствии с типом отображения программы пользователя делятся на виртуальные или привилегированные. Тип отображения определяется разрядом 10 в слове состояния задания (ССЗ). Он должен задаваться во время загрузки программы.

Виртуальный тип предоставляет пользователю большую (по сравнению с привилегированным) область адресного пространства (32К) для отображения в расширенную память.

Этот тип отображения выбирается установкой разряда 10 в ССЗ.

В системе расширенной памяти отображение по умолчанию является привилегированное отображение. Разряд 10

ССЗ очищен. Привилегированные задания могут иметь доступ к области векторов и связи с системой, а также к странице ввода-вывода.

Диспетчер памяти имеет два рабочих режима: внутренний и режим пользователя. Режим определяется разрядами 15, 14 в слове состояния процессора (ССП) и устанавливается монитором: 00 — внутренний, 11 — пользователя.

RMON и привилегированные программы выполняются во внутреннем режиме. KMON и виртуальные программы выполняются в режиме пользователя.

3.1. Особенности использования расширенной памяти

В табл. 1 приводится перечень особенностей использования программных запросов при работе с монитором расширенной памяти.

Таблица 1

Запрос	Особенности использования
1	2
.CDFN	Указанная область памяти для каналов должна находиться в нижних 28К физической памяти
.QSET	Указанная область памяти для элементов очереди должна располагаться в нижних 28К физической памяти. Для каждого элемента очереди должно отводиться 10 (десятичное) слов
.CNTXSW	Для виртуальных программ не применим, т. к. подпрограммы обработки прерываний недопустимы в виртуальных заданиях
.SETTOP	Применим только для области виртуальных адресов

4. АДРЕСАЦИЯ В СИСТЕМЕ РАСШИРЕННОЙ ПАМЯТИ

16-разрядное слово, используемое в операционной системе ФОДОС-2, не позволяет адресовать более 32К слов памяти. Однако, преобразование адресов с помощью диспетчера памяти позволяет адресовать до 124К слов памяти (плюс 4К слов на страницу ввода-вывода) на 18-разрядных процессорах с общей шиной и до 2044К слов памяти (плюс 4К слов на страницу ввода-вывода) на 22-разрядных процессорах с каналом Q—BUS. На рис. 1 показано адресное пространство для 18-разрядного и 22-разрядного слова.

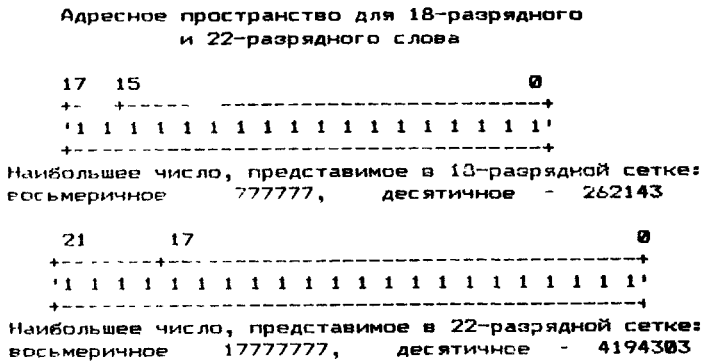


Рис. 1

4.1. Регистры адреса страницы и регистры признака страницы

Монитор системы ФОДОС-2 взаимодействует с диспетчером памяти через посредство регистров активной страницы, которые расположены в странице ввода-вывода. Каждый из них состоит из двух 16-разрядных слов, как показано на рис. 2: регистра адреса страницы (РАС) и регистра признака страницы (РПС).

Регистр активной страницы

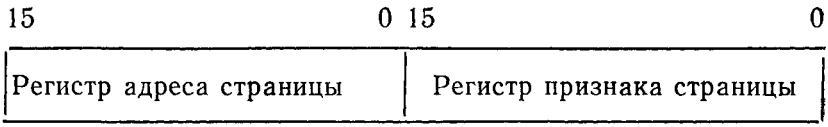


Рис. 2

Набор из восьми РАС и восьми РПС содержит информацию, необходимую для описания и перемещения восьми виртуальных адресных страниц. РПС описывает, какую часть страницы следует отобразить в память. РАС описывает, в какое место в памяти следует поместить страницу.

РАС и РПС имеют номера от 0 до 7 — по одному РАС/РПС на каждую виртуальную адресную страницу в виртуальном адресном пространстве 32К слов.

4.1.1. Регистр адреса страницы

Восемь РАС соответствуют восьми виртуальным адресным страницам. РАС содержит адрес физической памяти в блоках размером 32 (десятичное) слова — поле адреса страницы —

для конкретной виртуальной адресной страницы. Содержимое РАС показано на рис. 3. Разряды 0—11 используются для 18-разрядной адресации; разряды 0—15 — для 22-разрядной адресации.

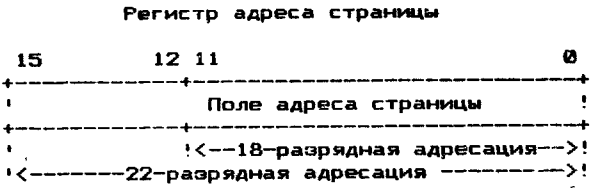


Рис. 3

4.1.2. Регистр признака страницы

РПС содержит информацию о расширении страницы, длине страницы и контроле доступа к конкретной странице. Как и РАС, РПС соответствуют виртуальным адресным страницам. Содержимое РПС показано на рис. 4.

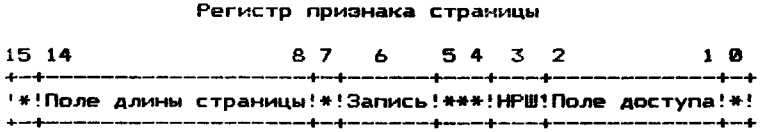


Рис. 4

Поле доступа описывает, какой доступ можно осуществить к странице, или может ли конкретный доступ прервать текущую операцию. Ниже приведены значения этого поля:

Величина	Значение
00	Страница не является резидентной. Прерывается любая попытка доступа к странице.
01	Резидентна. Только чтение. Прерывается любая попытка записи. (ФОДОС-2 это значение не использует).
10	Страница недоступна — все попытки доступа к этой странице прерываются. (ФОДОС-2 это значение не использует).
11	Резидентна. Чтение/запись. Разрешен любой доступ.

Поле НРШ — поле направления расширения страницы — указывает, в каком направлении — к старшим адресам (код 0) или к младшим адресам (код 1) может расширяться страница. ФОДОС-2 всегда использует только код 0, код 1 не используется.

Поле ЗАПИСЬ указывает, производилась ли запись в страницу с тех пор, как она загружена в память. ФОДОС-2 это поле не использует.

Поле длины страницы указывает длину страницы в блоках размером 32 (десятичное) слова.

4.2. Преобразование 16-разрядного адреса в 18- или 22-разрядный адрес

Информация, необходимая диспетчеру памяти для преобразования 16-разрядного виртуального адреса в 18- или 22-разрядный физический адрес, содержится в виртуальном адресе и соответствующем ему наборе РАС/РПС. Назначение полей виртуального адреса показано на рис. 5.

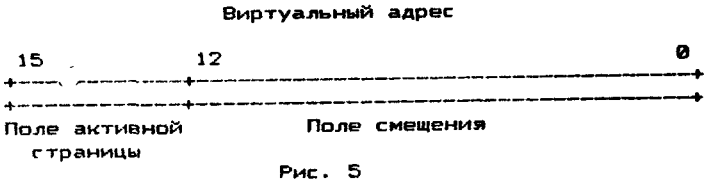


Рис. 5

Разряды 13—15 виртуального адреса составляют поле активной страницы. Это поле определяет, каким РАС/РПС воспользуется диспетчер памяти для создания физического адреса.

Разряды 0—12 составляют поле смещения — оно содержит адрес относительно начала страницы.

Остальная информация, необходимая для создания физического адреса, содержится в поле адреса страницы соответствующего РАС. Преобразование 16-разрядного виртуального адреса в 18- или 22-разрядный физический адрес показано на рис. 6. В этом примере РАС6 содержит число 5460 (восьмеричное), и виртуальный адрес 157746 преобразуется в физический адрес 565746. Разряды 12—15 РАС участвуют в 22-разрядной адресации.

Как видно из рис. 6, разряды 13—15 виртуального адреса указывают, который из РАС — в данном случае РАС6 — следует использовать. Диспетчер памяти складывает величину в разрядах 6—12 виртуального адреса с содержимым РАС6. Результат этого сложения диспетчер памяти помещает в разряды 6—17 или 6—21 физического адреса. Преобразование адреса завершается тем, что диспетчер памяти копирует разряды 0—5 виртуального адреса в разряды 0—5 физического адреса.

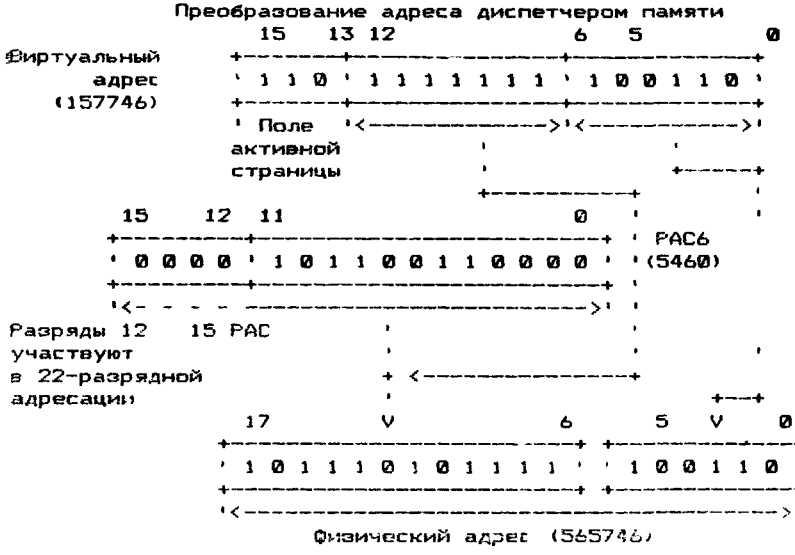


Рис. 6

ПРИМЕЧАНИЕ. Преобразование адресов происходит при установленном нулевом разряде (разряд включения ДП) регистра состояния SR0 диспетчера памяти (адрес регистра 777572). Другой регистр состояния ДП, SR2 (адрес регистра 777576, доступ — только чтение) содержит 16-разрядный виртуальный адрес, который в данный момент ДП преобразует в физический. ФОДОС-2 этот регистр не использует, однако, пользователь может сам проверить его — например, в случае сбоя ДП. Оба регистра расположены в странице ввода-вывода.

5. ОБЩЕЕ ОПИСАНИЕ РАБОТЫ РАСШИРЕННОЙ ПАМЯТИ ФОДОС-2

Доступ к расширенной памяти в ФОДОС-2 обеспечивается программными запросами, которые обрабатывает монитор ХМ.

С помощью запросов расширенной памяти монитор ХМ выполняет следующие операции:

— делит виртуальную память на адресные окна;

— создает области в расширенной памяти;

— отображает окна виртуальных адресов в созданные области.

Перечисленные выше операции должны выполняться перед выборкой любой ячейки в расширенной памяти (свыше 28К).

ПРИМЕЧАНИЕ. Две первые операции могут выполняться в любом порядке, но обязательно перед третьей операцией.

Ниже следует краткое описание каждой операции.

5.1. Создание виртуальных адресных окон

Диспетчер памяти (ДП) делит виртуальную память (PVAS) на 8 страниц по 4К слов. Страницы нумеруются от 0 до 7 (см. рис. 7) соответственно 8 регистрам PAC/РПС.

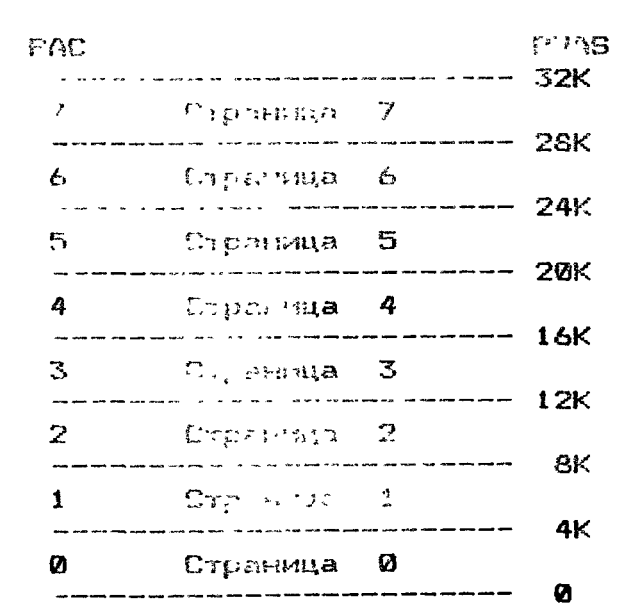


Рис. 7

Монитор XM делит виртуальное адресное пространство на окна. Окно представляет собой сегмент адресного пространства, начальный адрес которого должен быть кратен 4К. Размер окна может быть от 32 слов до 28К слов.

Окна подобны оверлейным сегментам. Всего может быть определено до 8 окон. Одновременное отображение всех окон не является обязательным.

Размер и основание (начало) окна определяется блоком определения окна (БООК), который задает пользователь и который имеется у каждого фактически определенного окна.

Запросы отображения должны обращаться к БООК, который содержит спецификации окна, параметры отображения и информацию о состоянии окна.

Идентификацию окна осуществляет монитор, который заносит индекс (число от 1 до 7) в соответствующий БООК во время выполнения запроса .CRAW.

Адресное окно с нулевым индексом является статическим окном и не может быть изменено программными запросами. Для виртуальных программ это окно создается монитором автоматически и отображается в статическую область.

Каждая виртуальная программа имеет одно статическое окно, в котором находится корневой сегмент программы; он отображается в соответствующую статическую область физической памяти, когда выполняется команда RUN или FRUN.

Когда программа использует программные запросы расширенной памяти, между виртуальным адресным пространством и физической памятью устанавливается соответствие в виде окон и областей. До тех пор, пока виртуальный адрес не входит в имеющееся адресное окно, он в отображении не участвует. Аналогично, окно может быть отображено только в то пространство, которое представляет собой ранее определенную область или ее часть.

Пример:

Программе пользователя требуются две рабочие области (см. рис. 8) — одна на 6К слов, другая на 8К слов. В этом случае пространство виртуальных адресов делится на 3 окна:
— статическое окно на 8К слов для базового сегмента;
— динамическое окно на 6К слов для 1-й рабочей области;
— динамическое окно на 8К слов для 2-й рабочей области.

Следует отметить, что определенные таким образом окна совмещают регистры PAC: окно 1 использует PAC2 и PAC3, тогда как окно 2 использует PAC4 и PAC5. При этом размер окна 1 равен 6К слов, и в области виртуального адреса имеется разрыв между 14К и 16К.

Эта область неопределенного виртуального адреса получена в результате аппаратного ограничения ДП, согласно которому окна должны начинаться с адреса, кратного 4К слов. В данном случае разрывности можно избежать, если поменять местами окна 1 и 2.

Как только программа определила окна и области, она

может использовать программные запросы для выполнения следующих операций:
— отобразить окно на всю область или ее часть;
— аннулировать отображение окна в одной области, чтобы отобразить его в другую область;
— аннулировать отображение окна из одной части области для отображения его в другую часть той же области.

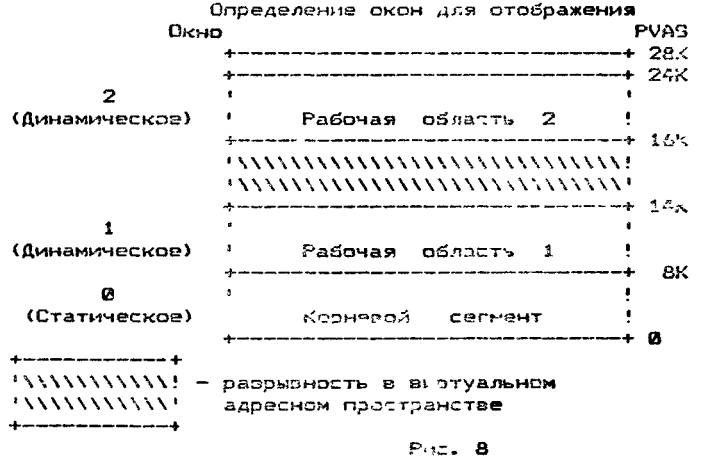


Рис. 8

5.2. Распределение и перераспределение областей

Другая операция, которую следует выполнить перед тем, как пользователь получит доступ к расширенной памяти, заключается в создании динамических областей. Монитор XM позволяет создавать до трех динамических областей (с помощью запроса .CRRG).

Эти области расположены в расширенной памяти и не включают основную (статическую) область программы, расположенную в нижних 28K памяти. Размер динамической области может быть от 32-х слов до 96K слов, но он всегда кратен 32: 32, 64, 96, ..., 96K. Это ограничение позволяет указать размер области (в блоках по 32 слова) в 16-разрядном слове.

Когда область создана, монитор присваивает ей идентификатор, который заносит в первое слово БООБ. Любой последующий запрос программы, обращенный к этой области, должен использовать идентификатор области.

Аннулирование динамической области осуществляется также через программный запрос (.ELRG). Когда динамическая область аннулируется, занимаемая ею область памяти возвра-

щается в список свободной памяти и может быть вновь использована. Кроме того, для всех отображенных на область окон в момент ее аннулирования действие отображения автоматически прекращается (отображение отменяется).

5.3. Отображение окон на области

Отображением называется процесс установления соответствия между виртуальными адресами и ячейками физической памяти. Фактически операция отображения представляет собой функцию, осуществляемую аппаратно, т. е. диспетчером памяти. Поэтому после того, как в БООБ и в БООК будут установлены необходимые параметры, ДП с помощью набора регистров РАС/РПС перемещает обращения к адресам из виртуальной памяти в физическую.

Необходимо помнить, что пользователь не может иметь непосредственного доступа к расширенной памяти, пока не отобразит часть виртуального адресного пространства в некоторую область физической памяти.

Понятие расширенной памяти может быть кратно изложено следующим образом:

1) программа пользователя оперирует с адресами из виртуального адресного пространства, ограниченного 16-разрядным словом адресации;

2) диспетчер памяти расширяет виртуальный адрес до 18-разрядного физического адреса, что позволяет адресовать до 128К слов физической памяти;

3) с помощью программных запросов виртуальное адресное окно может быть отображено на последовательные сегменты физической памяти посредством изменения величин смещения.

Отобразить окно на область можно двумя способами:

1) используя запрос .CRAW;

2) используя запрос .MAP.

В первом случае отображение окна происходит сразу же после создания этого окна и реализуется тем же запросом .CRAW, который создает окно. Для этого необходимо, чтобы в БООК был предварительно занесен идентификатор созданной области и в слове состояния БООК был установлен разряд WS.MAP, вызывающий автоматическое выполнение запроса .MAP после создания окна по запросу .CRAW.

Во втором случае для отображения ранее созданных окон используется запрос .MAP. Таким образом, в первом случае осуществляется неявное отображение окна, во втором — явное.

В БООК может быть указано смещение относительно начала области, с которым пойдет отображение. Если окно выходит за пределы области, то монитор отображает только часть окна.

Отменить отображение окна можно по запросу .UNMAP.

5.4. Ввод-вывод в отображении расширенной памяти

Монитор XM обслуживает ввод-вывод в пределах виртуального адресного пространства программы независимо от физического расположения буферов данных. Однако, ко времени выполнения запросов .READ или .WRITE буферы должны уже быть отображены в физическую память. Кроме того, буферы должны быть физически смежными и полностью находиться в пределах адресного окна. Такое ограничение необходимо потому, что адреса буферов вывода определены как виртуальные адреса программы.

Монитор преобразует виртуальный адрес в физический, когда выполняется программный запрос. Этот процесс позволяет программе пользователя освобождать буферы, используемые в .READ/.WRITE и .READC/WRITC после выполнения этих запросов и повторно отображать их при входе в соответствующую подпрограмму завершения.

ПРИМЕЧАНИЕ. Подпрограммы завершения должны оставаться отображенными до тех пор, пока не завершится передача данных.

6. РАСПРЕДЕЛЕНИЕ ПАМЯТИ

6.1. Распределение памяти при загрузке монитора XM

Расположение компонент монитора XM показано на рис. 9 (следует отметить, что оно сходно с расположением компонент монитора FB). Когда монитор XM только вызван, драйвер системного устройства и резидентный монитор используют нижние 28K слов памяти, так что расширенная память — выше 28K и ниже 124K — не используется. Другие загруженные драйверы устройств располагаются ниже RMON, еще ниже — основное и системные задания, если они есть, и USR.

Резидентный монитор работает во внутреннем режиме процессора и может иметь доступ к нижним 28K слов памяти и к странице ввода-вывода. USR также работает во внутреннем режиме и всегда резидентна в системе расширенной памяти. KMON работает в режиме пользователя, однако, поскольку это привилегированное фоновое задание, оно использует то же отображение, что и резидентный монитор. Физические ячейки 0—500 содержат векторы.

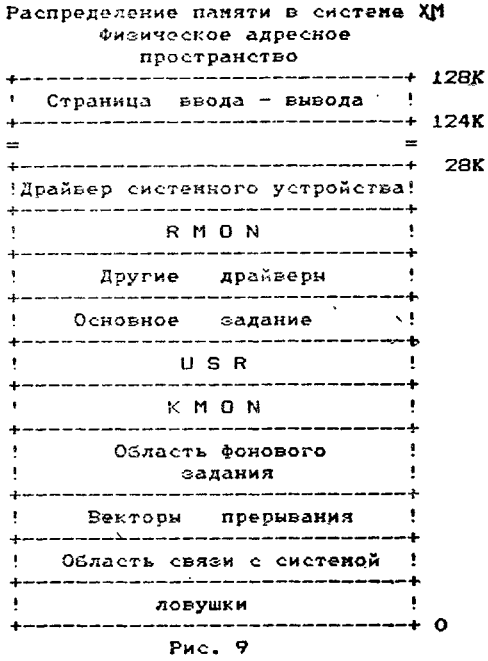


Рис. 9

6.2. Отображение виртуальных заданий

Задания, использующие виртуальное отображение, выполняются в режиме пользователя и не используют внутреннее отображение. Виртуальные фоновые задания загружаются в память со смещением 500, при этом они не могут загружаться на области, которые занимают USR, резидентный монитор или страница ввода-вывода. Виртуальное отображение более всего подходит для задания, не требующего привилегированного доступа к векторной области, монитору или странице ввода-вывода, поскольку виртуальное отображение защищает эти системные области от виртуальных заданий.

Первые 500 байт каждого файла виртуального задания составляют его виртуальную векторную область и область связи с системой. Статическое окно включает в себя виртуальные адреса между нулевым виртуальным адресом программы и ее старшим адресом. Размер статической области изменяется в зависимости от размера задания и от того, фоновое оно или основное.

Когда виртуальное задание вызывается впервые, оно может иметь доступ только к виртуальным адресам в пределах

его собственной программы, а также к тем, которые отображены в физическую память. Виртуальное задание, однако, может использовать остальное виртуальное адресное пространство от своего старшего адреса до границы 32К слов. Оно может создавать одну или несколько областей в расширенной памяти, а также одно или несколько виртуальных адресных окон, после чего отобразить окно в область и получить тем самым доступ к расширенной памяти. Если виртуальное задание отменяет отображение окна, то оно не может использовать входящие в это окно виртуальные адреса, пока повторно не отобразит его. Виртуальное задание может использовать XM .SETTOP, а также оверлейную структуру в расширенной памяти.

6.2.1. Выбор типа отображения

Выбор виртуального отображения осуществляется установкой 10 разряда слова состояния задания до того, как программа начнет работать. Если конкретное задание предполагается использовать только как виртуальное, то разряд 10 можно установить во время трансляции, указав в исходной программе:

```
.ASECT
. = 44
.WORD 2000
.PSECT
```

При желании выбор виртуального отображения можно осуществить с помощью программы SIPP, изменив ячейку 44 в загрузочном модуле задания (тип файла .SAV, .REL или .SYS) до того, как начнет работать программа.

ПРИМЕЧАНИЕ. Не следует изменять значение 10-го разряда ССЗ во время работы программы. Это может нарушить правильную обработку запросов ввода-вывода; последствия этого шага предсказать трудно.

6.2.2. Виртуальные фоновые задания

Для вызова виртуального фонового задания используется команда R, причем тип файла должен быть .SAV. Виртуальное фоновое задание загружается в память с адреса 500. Старший адрес этого задания равен его размеру в восьмеричной системе плюс 500.

Статическая область виртуального фонового задания начинается с физического адреса 500 и простирается до самого низкого адреса, используемого USR. Тем самым запрещается доступ виртуального фонового задания к физической области

от 0 до 500, следовательно, векторы защищены от виртуальных заданий. Отображение виртуального фонового задания показано на рис. 10.

Рис. 10

На рис. 11 показано, как виртуальное фоновое задание может отобразить окно в статическую область и использовать память, находящуюся ниже USR.

6.3. Виртуальные основные или системные задания

Для выполнения виртуального основного задания используется команда монитора FRUN, а виртуального системного задания — команда SRUN. В процессе редактирования связей эти задания следует связать в файл типа .SAV, как фоновые задания, а не типа .REL или .SYS, как основные или системные задания. Использовать команды FRUN или SRUN для выполнения файла .SAV типа — виртуального задания — можно потому, что виртуальным основным заданиям не требуется информация о перемещении. Таким образом, файлы

типа .SAV занимают на диске меньше места, чем файлы типа .REL или .SYS, и в память они загружаются быстрее.

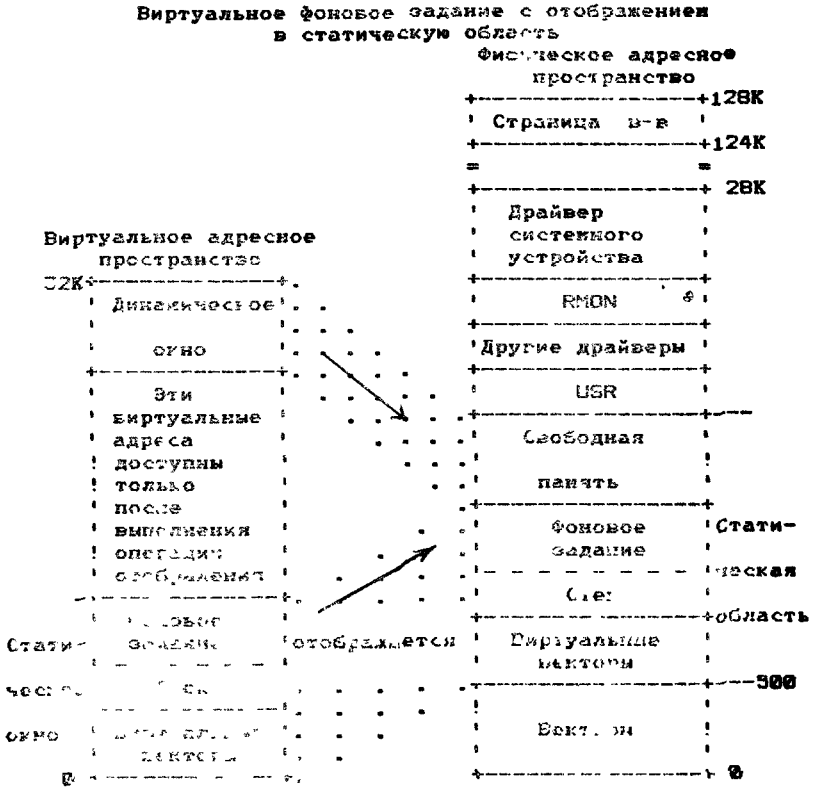


Рис. 11

При загрузке основное задание располагается в памяти ниже последнего драйвера или ранее загруженного системного задания. При необходимости USR сдвигается вниз, освобождая место для основного задания. При редактировании связей основное задание по умолчанию получает базовый адрес 1000 (если только это не файл типа .SAV: его виртуальные адреса 0—500 представляют собой его виртуальную векторную область и область связи с системой). Как и в случае виртуального фонового задания, статическое окно начинается с виртуального адреса ноль и простирается до старшего адреса основной программы, округленного до кратного 32 слов.

Статическая область начинается с ячейки ноль и простирается до старшего адреса программы. Смешанная область основного задания располагается в физической памяти непосредственно перед программой. В смешанную область, однако, виртуальные адреса не отображаются, так что виртуальное основное задание не может иметь доступа к смешанной области; следовательно, смешанная область защищена от виртуального основного задания. Отображение основного или системного задания показано на рис. 12.

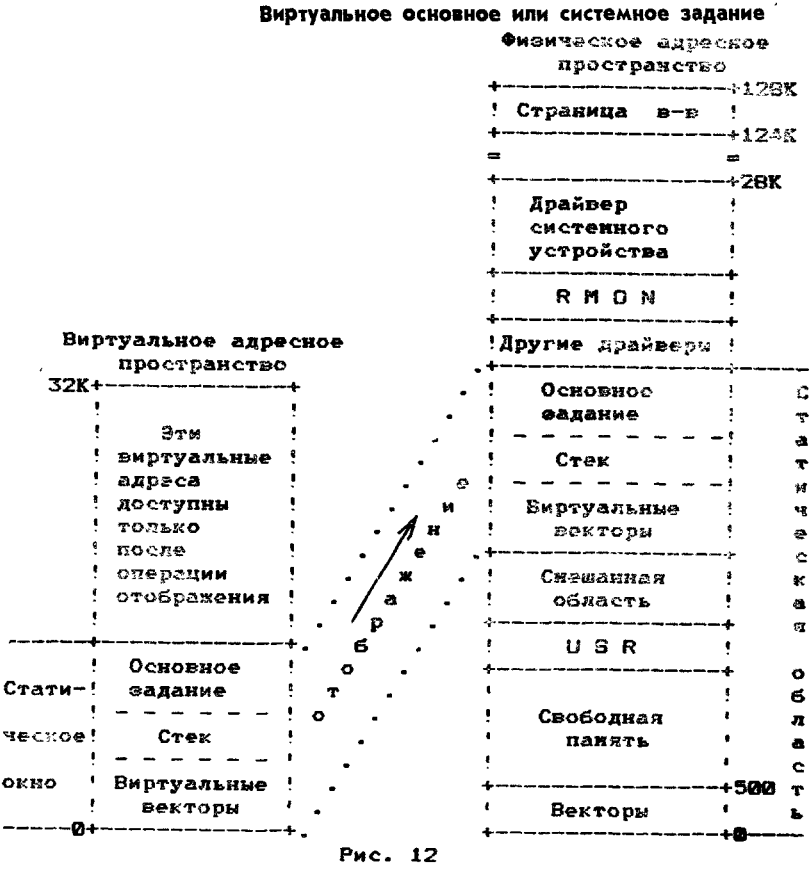


Рис. 12

6.4. Привилегированные задания

Отображением по умолчанию в мониторе расширенной памяти является привилегированное отображение. Признаком привилегированного отображения является очищенный раз-

ряд 10 ССЗ. Для привилегированного задания среда монитора XM весьма сходна с мониторами SJ или FB. Оно может иметь доступ к нижним 28К слов памяти и к странице ввода-вывода. Все обслуживающие программы ФОДОС-2 в среде монитора расширенной памяти работают как привилегированные задания.

Как и виртуальные, привилегированные задания выполняются в режиме пользователя процессора. Однако, монитор копирует содержимое регистров PAC/РПС внутреннего режима в PAC/РПС режима пользователя. Таким образом, отображение по умолчанию для привилегированных заданий то же самое, что и отображение по умолчанию внутреннего режима.

Привилегированные задания располагают виртуальным адресным пространством 32К слов. Однако, большая часть этого виртуального пространства уже отображена: это программные средства операционной системы, страница ввода-вывода и — в случае привилегированного основного или системного задания—фоновое задание или клавиатурный монитор. Привилегированное задание может изменить свое отображение по умолчанию с помощью оверлеев расширенной памяти или программных запросов. Например, программа, которой лишь на короткое время нужна страница ввода-вывода может отменить ее отображение, когда надобность в ней отпадет.

Следует отметить, что концепции статического окна и статической области не относятся к привилегированным заданиям. Монитор, однако, резервирует одно окно и одну область. Таким образом, привилегированные задания могут располагать семью динамическими окнами и тремя динамическими областями, как и виртуальные задания.

Когда привилегированное задание создает окно и выполняет программные запросы отображения, привилегированное отображение по умолчанию для этого виртуального адресного пространства временно отменяется. Монитор отображает окно в новую область памяти, используя содержимое внутренних управляющих блоков окна (УБОК). Когда привилегированное задание отменяет отображение окна, монитор повторно отображает это виртуальное адресное пространство в соответствии с содержимым PAC/РПС внутреннего режима. Этим привилегированное задание отличается от виртуального, которое в случае отмены отображения не может использовать входящие в окно виртуальные адреса до тех пор, пока окно не будет повторно отображено.

Поскольку подпрограммы обслуживания прерываний выполняются в отображении внутреннего режима, привилегированные задания, содержащие пользовательские подпрограммы обслуживания прерываний, не должны изменять отображений программ обслуживания прерываний, страницы ввода-вывода или частей монитора в течение любого периода времени, когда может произойти прерывание. Для монитора важно, что отображения режима пользователя и внутреннего режима идентичны, когда обслуживаются прерывания пользователя.

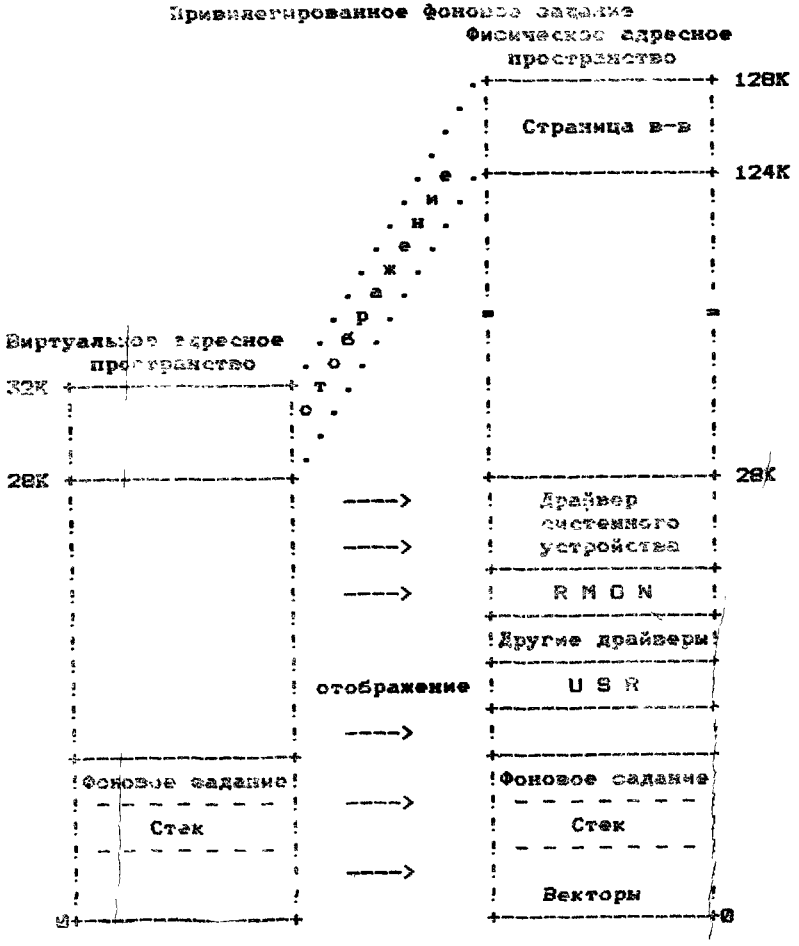


Рис. 13

6.4.1. Привилегированные фоновые задания

Для выполнения привилегированного фонового задания следует подать команду монитора R или RUN. Отображение для привилегированного фонового задания показано на рис. 13.

6.4.2. Привилегированные основные или системные задания

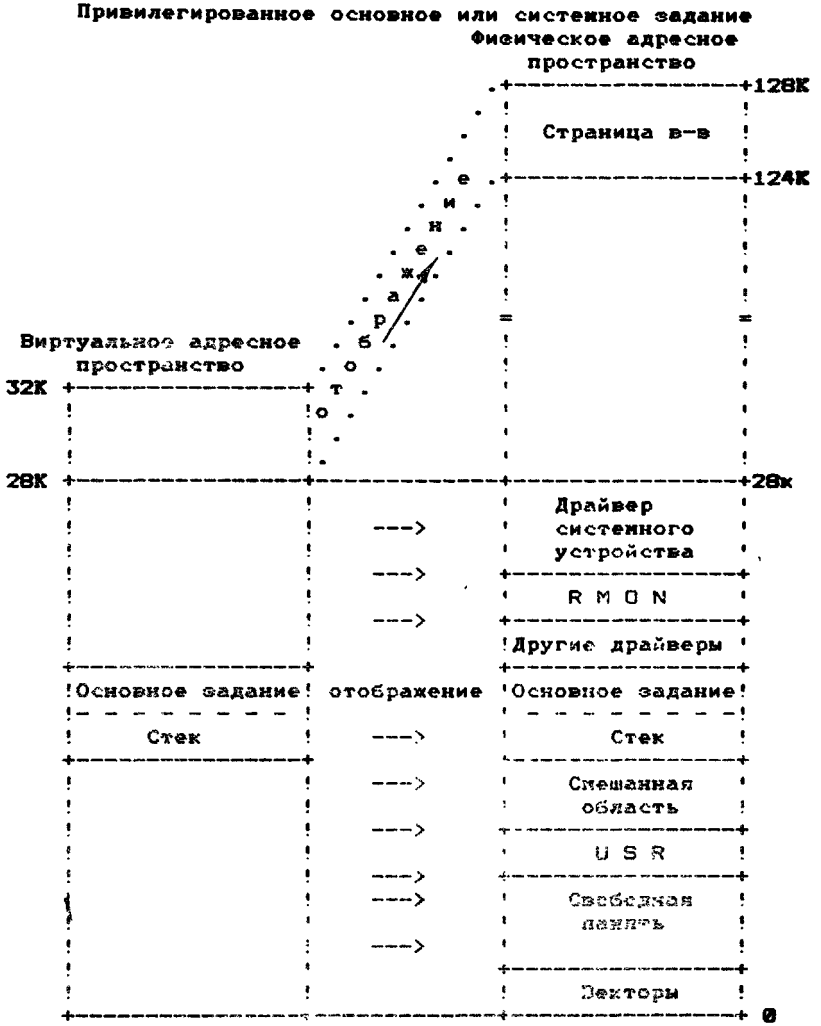


Рис. 14

Для выполнения привилегированного основного задания следует подать команду монитора FRUN, а привилегированного системного задания — команду SRUN. Отображение для привилегированного основного или системного задания показано на рис. 14.

6.5. Различия между виртуальными и привилегированными заданиями

В табл. 2 приведены различия между виртуальными и привилегированными заданиями.

Таблица 2

Характеристика	Виртуальное задание	Привилегированное задание
1	2	3
Значение 10 разряда ССЗ	1	0
Первоначально доступное адресное пространство	Только виртуальные адреса в пределах программы	32К слов (нижние 28К слов плюс страница ввода-вывода)
Потенциально доступное адресное пространство	32К слов. Создает окна, описывающие адреса между старшим адресом программы и границей 32К слов	32К слов. Если некоторые части виртуального адресного пространства уже используются (например, фоновым заданием), то это задание может отменить отображение этих частей и повторно отобразить эти адреса в память выше 28К. Некоторые области надо оставлять отображенными всякий раз, когда работает пользовательская программа обработки прерываний
Преимущества	Обеспечивает защиту операционной системы и других программ. Забирает у других заданий минимум физической памяти	Совместимо с мониторами SJ и FB

1	2	3
Процедура вызова	Фоновое: команда R (.SAV) Основное: FRUN или SRUN (.REL, .SYS, .SAV — рекомендуется .SAV)	Фоновое: команда R или RUN (.SAV) Основное: FRUN или SRUN (.REL, .SYS)
Статическое окно	От нулевого виртуального адреса программы до ее старшего адреса	Нет. Все динамические
Статическая область	Фоновое: от физической ячейки 500 до адреса USR Основное: от физической ячейки 0 до физического старшего адреса задания	Нет Все динамические
Возможное количество окон	7 плюс статическое окно	7 (1 окно зарезервировано)
Возможное количество областей	3 плюс статическая область	3 (1 область зарезервирована)

6.6. Переключение монитора между заданиями

В системе двух заданий монитор сохраняет специфическую для задания информацию, когда текущее задание заменяется новым. Монитор восстанавливает эту информацию, когда первое задание возобновляет работу, — эта процедура называется переключением монитора между заданиями.

В мониторе расширенной памяти каждое задание может быть или виртуальным, или привилегированным. Следовательно, в мониторе расширенной памяти работы по переключению заданий у монитора больше.

Когда монитор переключает текущее задание, то он сохраняет в стеке задания и смешанной области задания следующую информацию:

— PS;
— PC;
— SP (в смешанной области);
— регистры R0—R5;
— регистр PAC1 внутреннего режима (только для XM);
— вектор прерывания по отказу ДП (только для XM);

— вектор BPT (только для XM);
— вектор IOT (только для XM);
— вектор TRAP;
— область связи с системой (ячейки 40—52);
— ячейка 56 (только для мультитерминальных систем);
— слово состояния процессора плавающей запятой (ППЗ) и регистры плавающей запятой (если имеется аппаратура плавающей запятой);
— все данные, указанные программой в запросе .CNTXSW;
— стек и смешанная область (как часть задания).

Монитор, однако, не сохраняет содержимого регистров PAC/PПC текущего задания. Поэтому программам пользователя не следует непосредственно манипулировать с регистрами диспетчера памяти, поскольку во время переключения монитора их содержимое теряется. Монитор также игнорирует программный запрос .CNTXSW в виртуальном задании. Все задание целиком сохраняется при переключении, а доступ к векторной области виртуальному заданию запрещен в любом случае.

Когда происходит переключение монитора на новое задание, сначала предполагается, что оно привилегированное, и содержимое регистров внутреннего отображения копируется в регистры пользователя. Задание может иметь доступ к нижним 28К слов памяти плюс к странице в—в. После этого монитор проверяет, не клавиатурный ли это монитор. Если это так, то выполнение продолжается без дальнейших изменений.

Если новое задание — привилегированное, то монитор затем проверяет УБОК и УБОБ в смешанной области задания. Если задание определило и отобразило одно или несколько окон, то монитор восстанавливает отображение, в основе которого — содержимое внутренних управляющих блоков, и изменяет, таким образом, привилегированное отображение по умолчанию для этих окон.

Если новое задание виртуальное, то монитор очищает регистры отображения пользователя. Затем он просматривает УБОК и УБОБ в смешанной области задания. Монитор отображает только ту часть виртуального адресного пространства задания, которая была определена в окне и отображена в область в то время, когда произошло переключение задания. Конечно, любая попытка обращения к неотображенному адресу вызывает сбой диспетчера памяти. Неиспользованные части виртуального адресного пространства остаются неотображенными, если только виртуальное задание не отобразит их явно.

7. ОБРАЩЕНИЕ К ПРОГРАММЕ. ВХОДНЫЕ И ВЫХОДНЫЕ ДАННЫЕ

7.1. Получение доступа к расширенной памяти

Чтобы получить доступ к расширенной памяти, пользователь должен использовать следующие программные запросы монитора:

.CRAW — создает адресное окно;
.ELAW — аннулирует адресное окно;
.CRRG — создает область;
.ELRG — аннулирует область;
.MAP — отображает адресное окно в область;
.UNMAP — отменяет отображение адресного окна;
.GMCX — устанавливает состояние отображения.

Сначала необходимо выполнить следующие операции:

1) создать блоки определения области, используя запрос .RDBBK, или определить параметры и установить БООБ с помощью запроса .RDBDF;

2) создать необходимые области для размещения программы, используя запрос .CRRG для каждой области. Область аннулируется по запросу .ELRG;

3) создать блоки определения окна, используя запрос .WDBBK, для каждого окна или определить параметры и установить БООК с помощью запроса .WDBDF;

4) создать необходимые окна для размещения программы, используя запрос .CRAW для каждого создаваемого окна. Окно аннулируется по запросу .ELAW;

5) отобразить окна в области, используя запросы .MAP или .CRAW. Отображение окна отменяется запросом .UNMAP или повторным применением запроса .MAP.

После выполнения этих операций пользователь получает доступ к расширенной памяти.

Сложность запросов требует специальных средств связи между программой пользователя и монитором. Эта связь реализуется через структуры данных, с помощью которых:

1) программа указывает, какие действия должен выполнить монитор;

2) монитор сообщает программе о результатах запрашиваемых действий.

Имеются следующие типы структур данных:

1) блок определения окна (БООК);
2) блок определения области (БООБ);
3) управляющий блок окна (УБОК);

4) управляющий блок области (УБОБ);
5) список свободной памяти.

Программные запросы расширенной памяти используют эти структуры как области связи между программой и монитором. Значения, записываемые программой в блок, определяют или модифицируют запрашиваемую операцию. После того, как монитор выполнил операцию, он записывает информацию о выполненных действиях в ячейки этого блока.

Структура данных (БООК и БООБ) должна быть определена пользователем в соответствии с описанными ниже правилами.

7.2. Блок определения окна

Блок определения окна (см. рис. 15) используется для определения окна и хранения информации о нем. БООК может быть создан по запросу .WDBBK во время трансляции. Размер блока равен 7-ми словам. Три первых слова используются для определения параметров окна.

Первое слово содержит следующую информацию:

W.NID (младший байт) — содержит код идентификации окна (записывается монитором); используется запросами для отыскания требуемого окна;

N.NAPR (старший байт) — содержит номер регистра активной страницы, определяющего базовый виртуальный адрес окна (задается пользователем). W.NAPR может принимать значения от 0 до 7; окна должны начинаться с адреса, кратного 4К (см. табл. 3).

ПРИМЕЧАНИЕ. Значение 0 в W.NAPR задавать не следует, т. к. 0-е окно является статическим и не может быть переназначено.

Таблица 3

Начальный виртуальный адрес окна (восьмеричный)	Значение параметра W.NAPR
1	2
0	0
20000	1
40000	2
60000	3
100000	4
120000	5
140000	6
160000	7

W.NBAS — начальный виртуальный адрес окна; заполняется монитором.

W.NSIZ — размер окна в блоках по 32 (десятичное) слова (задается пользователем); если размер не кратен 4K, то в виртуальном адресном пространстве возникает разрыв, т. к. следующее окно должно начинаться с адреса, кратного 4K.

Оставшиеся 4 слова БООК используются при отображении окна в область расширенной памяти.

Блок определения окна

Символьное обозначение		Смещение (в байтах)
	+----------+----------+	0
W.NID W.NAPR	' Номер регистра ' Идентификатор ' активной страницы ' окна '	
	+----------+----------+	2
W.NBAS	' Виртуальный адрес начала окна (в байтах) '	
	+----------+	4
W.NSIZ	' Размер окна ' (в блоках по 32 слова) '	
	+----------+	6
W.NRID	' Идентификатор области, связанной ' с окном '	
	+----------+	10
W.NOFF	' Смещение в области ' (в блоках по 32 слова) '	
	+----------+	12
W.NLEN	' Длина отображаемого окна ' (в блоках по 32 слова) '	
	+----------+	14
W.NSTS	' Слово состояния окна '	
	+----------+	

Рис. 15

W.NRID — идентификатор области, в которую будет отображаться окно.

W.NOFF — смещение от начала области (в блоках по 32 слова), с которым пойдет отображение окна.

W.NLEN — размер отображаемого окна (в блоках по 32 слова), которое будет отображаться в область; если W.NLEN=0, то отображается все окно или та его часть, которая может разместиться в данной области; при этом после отображения в ячейке W.NLEN будет содержаться фактическая длина отображенного окна (или его части).

W.NSTS — слово состояния окна. Разряды слова состояния окна определены следующим образом:

W.NSTS	WS.CRW		WS.UNM	WS.ELW		WS.MAP		
	15	14	13	12 ...	8	7	...	0

WS.MAP — если пользователь этот разряд установил, то во время выполнения запроса .CRAW после создания окна автоматически происходит отображение этого окна в указанную область.

WS.CRW — устанавливается монитором, если адресное окно создано.

WS.UNM — устанавливается монитором, если одно или несколько окон было аннулировано для создания и отображения данного окна.

WS.ELW — устанавливается монитором при аннулировании одного или нескольких окон.

7.3. Управляющий блок окна

Управляющий блок окна (УБОК) — это область из семи слов, расположенная в смешанной области задания. Содержимое УБОК контролируется монитором. Виртуальное задание отводит один УБОК статическому окну. Для привилегированных заданий один УБОК резервируется монитором, и программа его использовать не может. Таким образом, все задания могут иметь до семи динамических окон, состояние которых монитор обслуживает через посредство УБОК, структура которых показана на рис. 16 и в табл. 4.

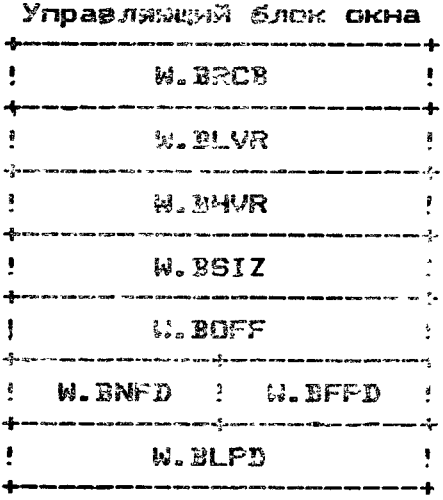


Рис. 16

Таблица 4

УПРАВЛЯЮЩИЙ БЛОК ОКНА

Смещение в байтах	Символ	Кто вносит изменения	Содержание
1	2	3	4
0	W.BRCB	Подпрограмма .MAP монитора; очищается по запросу .UNMAP	Указатель УБОБ области, в которую это окно отображается. Если значение нулевое, то окно не отображается
2	W.BLVR	Подпрограмма .CRAW монитора	Нижний предел виртуального адреса окна
4	W.BHVR	Подпрограмма .MAP монитора	Верхний предел виртуального адреса окна
6	W.BSIZ	Подпрограмма .CRAW монитора; очищается по запросу .ELAW	Размер окна в блоках по 32 слова. Если значение нулевое, то этот УБОК свободен
10	W.BOFF	Подпрограмма .MAP монитора	Смещение (в блоках по 32 слова) в области, с которым пойдет отображение
12	W.BFPD	Подпрограмма .CRAW монитора	Младший байт адреса первого РПС, действующего на это окно
13	W.BNPD	Подпрограмма .MAP монитора	Номер РПС, действующего на это окно
14	W.BLPD	Подпрограмма .MAP монитора	Содержимое последнего РПС, действующего на это окно

7.4. Блок определения области

Блок определения области (см. рис. 17) используется для определения области и хранения информации о ней. Размер блока — 3 слова.

Блок определения области может быть создан по запросу .RDBBK во время трансляции.

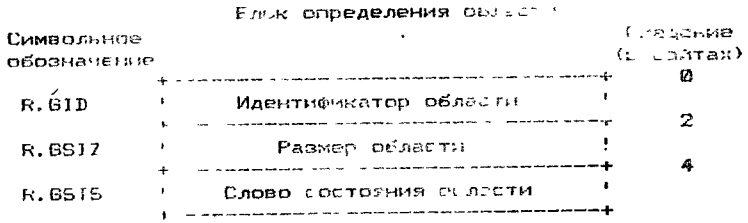
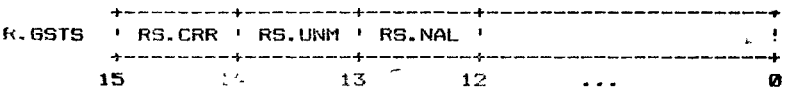


Рис. 17

R.GID — идентификатор области (устанавливается монитором при выполнении запроса .CRRG); идентификатор должен быть использован при обращении к области с помощью программных запросов.

R.GSIZ — размер динамической области в блоках по 32 слова; задается пользователем.

R.GSTS — слово состояния области; заполняется монитором. Разряды слова состояния определены следующим образом:



RS.CRR — устанавливается, если область создана успешно.

RS.UNM — устанавливается, если при аннулировании данной области было отменено отображение одного или нескольких окон.

RS.NAL — устанавливается, если невозможно создать область требуемого размера.

7.5. Управляющий блок области

Управляющий блок области (УБОБ) — это область из трех слов в смешанной области задания. Содержимое УБОБ контролируется монитором. Виртуальное задание отводит один УБОБ статической области. Для привилегированных заданий один УБОБ резервируется монитором, и программа его использовать не может. Таким образом, все задания могут иметь до трех динамических областей, состояние которых монитор обслуживает через посредство УБОБ, их структура показана на рис. 18 и в табл. 5.

7.6. Создание БООК

Для создания БООК используются два запроса: .WDBDF и .WDBBK.

Управляющий блок области

R.RADD	
R.BSIZ	
R.BNWD	R.BSTA

Рис. 18

Таблица 5

УПРАВЛЯЮЩИЙ БЛОК ОБЛАСТИ

Смещение в байтах	Символ	Кто вносит изменения	Содержание
1	2	3	4
0	R.BADD	Подпрограмма .CRRG монитора	Начальный адрес области в блоках по 32 слова
2	R.BSIZ	Подпрограмма .CRRG монитора	Размер области в блоках по 32 слова. Если это слово нулевое, то этот УБОБ свободен
4	R.BSTA	Монитор во время выполнения; очищается подпрограммой .CRRG	Этот байт всегда нулевой, если только область не была создана по ХМ .SETTOP. Тогда монитор устанавливает разряд 1, называемый R.STOP
5	R.BNWD	Подпрограмма .CRRG монитора очищает этот байт; .MAP увеличивает его, .UNMAP — уменьшает	Количество окон, отображенных в данный момент в эту область

По запросу .WDBBK создается БООК. При этом смещения для БООК и разряды слова состояния определяются автоматически, т. к. по запросу .WDBBK автоматически вызывается запрос .WDBDF. Поэтому пользователю не нужно определять .WDBDF, когда используется запрос .WDBBK.

Формат макрокоманды:

.WDBBK W.NAPR,W.NSIZ[,W.NRID,W.NOFF,W.NLEN, W.NSTS]

где аргументы определяют значения для БООК.

ПРИМЕЧАНИЕ. В квадратные скобки заключены аргументы, не являющиеся обязательными.

Когда требуется определить только смещения для БООК, используется запрос .WDBDF, имеющий формат: .WDBDF

По запросу .WDBDF определяются следующие параметры:

1) смещения в БООК —

```
W.NID  = 0
W.NAPR = 1
W.NBAS = 2
W.NSIZ = 4
W.NRID = 6
W.NOFF =10
W.NLEN =12
W.NSTS =14
```

2) размер БООК в байтах —

```
W.NLGH =16
```

3) разряды слова состояния (W.NSTS) БООК —

```
WS.CRW =100000
WS.UNM = 40000
WS.ELW = 20000
WS.MAP =   400
```

Пример:

Необходимо создать БООК, который определяет окно, имеющее длину 76 (десятичное) блоков и начальный виртуальный адрес 20К. Окно должно отображаться в область, начиная с 50-го блока от начала области, причем полностью (если позволяет размер оставшейся части области), либо столько, сколько уместится.

Этим требованиям удовлетворяет БООК, созданный по запросу: .WDBBK 5,76.,0,50.

Макрорасширение имеет вид:

```
.BYTE 5
.WORD
.WORD 76.
.WORD 0
.WORD 50.
.WORD
.WORD
```

7.7. Создание БООБ

Как для блоков определения окна, так и для блоков определения области имеется два запроса, которые выполняют аналогичные функции.

По запросу .RDBBF определяются разряды слова состояния и смещения для БООБ.

По запросу .RDBBK создается БООБ. При этом смещения для БООБ и разряды слова состояния определяются автоматически, т. к. .RDBBK автоматически вызывает запрос .RDBDF, аналогично тому, как это делает .WDBBK. Поэтому пользователю не нужен запрос .RDBDF, когда используется .RDBBK.

Формат макрокоманды: .RDBBK RGSIZ
где RGSIZ — размер динамической области (соответствует R.GSIZ для БООБ) в блоках по 32 слова.

Когда требуется определить только смещения и разряды слова состояния для БООБ, используется запрос .RDBBF, формат которого: .RDBDF

Запрос .RDBDF определяет следующие параметры:

1) смещения БООБ

R.GID =0
R.GSIZ =2
R.GSTS =4

2) размер БООБ в байтах:

R.GLGH =6

3) разряды слова состояния (R.GSTS) БООБ:

RS.CRR=100000
RS.UNM= 40000
RS.NAL= 20000

Пример:

С помощью запроса .RDBBK создать БООБ, определяющий динамическую область размером 102 (десятичное) блока.

Указанным требованиям удовлетворяет запрос .RDBBK #102.

Макрорасширение имеет вид:

```
.WORD 0
.WORD 102.
.WORD 0
```

7.8. Элемент очереди ввода-вывода

В отличие от мониторов SJ и FB, где элемент очереди ввода-вывода состоит из семи слов, в мониторе XM элемент очереди ввода-вывода состоит из десяти слов. Подробное описание этой структуры см. в [2].

7.9. Список свободной памяти

Монитор использует список свободной памяти для распределения областей в расширенной памяти. Список представляет собой таблицу из 10 (десятичное) пар слов, расположенную в P-секции XMSUBS; адрес вершины таблицы

$XMSIZ. Старшее слово в каждой паре указывает размер свободной области в расширенной памяти в блоках по 32 слова. Младшее слово пары содержит адрес этой области, деленный на 100 (восьмеричное). Таблица заканчивается величиной —1.

Во время загрузки монитора таблица содержит только один элемент: старшее слово пары содержит общее количество расширенной памяти, младшее — число 1600. Когда задание запрашивает область в расширенной памяти, монитор просматривает таблицу в поисках области подходящего размера. Из всех областей требуемого размера задание получит область с наименьшим начальным адресом. Монитор сокращает общее количество свободной памяти в старшем слове первой пары и указывает соответствующее значение для ее начального адреса.

8. ЗАПРОСЫ РАСШИРЕННОЙ ПАМЯТИ

8.1. Создание области (.CRRG)

Этот запрос создает динамическую область в расширенной памяти для выдавшей запрос программы.

Формат макрокоманды: .CRRG AREA[,RGADR]
где AREA — адрес блока аргументов EMT, имеющего вид:
AREA: .BYTE 0,36
.WORD RGADR

RGADR — адрес блока определения области (этот аргумент необязателен, если пользователь сам заполнил 2-е слово блока аргументов).

Подпрограмма .CRRG монитора проверяет сначала R.GSIZ в БООБ. Если размер области задан неверно (нулевой или размер более 96К слов), то в байт 52 заносится код ошибки 10.

Далее подпрограмма ищет свободный УБОК. Если его нет, то в байт 52 заносится код ошибки 6.

Максимально возможную память для области пользователь может запросить, указав 96К в качестве размера области. Подпрограмма .CRRG монитора просматривает список свободной памяти монитора в поисках нужных областей. Если память требуемого размера монитор предоставить не может, то в байт 52 заносится код ошибки 7, а в R0 размер наибольшей доступной области. Пользователю следует снова подать запрос .CRRG, указав новое содержимое R0 в качестве размера области. Если и в этом случае выдается ошибка, значит, какое-то задание в системе уже перехватило часть

памяти. Запрос .CRRG, в котором в качестве размера области указаны новые значения, взятые из R0, следует подавать до тех пор, пока не будет получена область, — это и есть нормальное завершение запроса. Подпрограмма .CRRG тогда заносит в БООБ (R.GID) идентификатор области и устанавливает RS.CRR в слове состояния области. Кроме этого, она очищает R.BSTA и R.BNWD и заносит соответствующие значения в R.BADD и R.BSIZ, находящиеся в УБОБ. Полученная таким образом память удаляется из списка свободной памяти монитора и резервируется для задания пользователя.

8.2. Аннулировать область (.ELRG)

Этот запрос аннулирует динамическую область в физической памяти; освободившаяся область памяти может быть вновь использована.

Формат макрокоманды: .ELRG AREA[,RGADR]
где AREA — адрес блока аргументов EMT, имеющего вид:

```
AREA:   .BYTE   1,36
        .WORD   RGADR
```

RGADR — адрес блока определения области (этот аргумент необязателен, если пользователь сам заполнил 2-е слово блока аргументов).

Подпрограмма .ELRG монитора сначала проверяет на допустимость указанный пользователем идентификатор области. Для виртуальных заданий нулевой идентификатор обозначает статическую область, аннулировать которую пользователь не может. Для привилегированных заданий областей с нулевым идентификатором не существует. В этих случаях, а также в случае недопустимого идентификатора области в УБОБ, в байт 52 заносится код ошибки 2.

Далее подпрограмма очищает RS.CRR и RS.UNM в слове состояния области. Если в эту область были отображены какие-либо окна, подпрограмма отменяет отображение и устанавливает RS.UNM. Память, занимаемая областью, вносится в список свободной памяти монитора. Завершая обработку запроса, подпрограмма .ELRG очищает УБОБ.

8.3. Создание адресного окна (.CRAW)

Этот запрос создает виртуальное адресное окно и, если установлен разряд WS.MAP в слове W.NSTS БООК, отображает его в область физической памяти (идентификатор области должен быть предварительно занесен в W.NRID БООК).

Формат макрокоманды: .CRAW AREA[,WADR]
где AREA — адрес блока аргументов EMT, имеющего вид:

```
AREA:   .BYTE   2,36
        .WORD   WADR
```

WARD — адрес блока определения окна (этот аргумент необязателен, если пользователь сам заполнил 2-е слово блока аргументов указателем адреса).

Подпрограмма .CRAW монитора сначала проверяет на допустимость W.NAPR в БООК, и если номер регистра активной страницы указан неверно, то в байт 52 заносится код ошибки 0. Затем подпрограмма устанавливает базовый адрес окна в БООК (W.NBAS) и проверяет на допустимость указанный пользователем в БООК (W.NSIZ) размер окна. Окно не должно превосходить 32К слов. Если размер окна указан неверно, в байт 52 заносится код ошибки 0. Подпрограмма очищает WS.ELW, WS.UNM и WS.CRW в слове состояния окна.

Далее подпрограмма проверяет, не происходит ли наложения нового окна на старые. Если задание виртуальное и новое окно перекрывает статическое окно, то в байт 52 заносится код ошибки 0. Во всех остальных случаях, когда новое окно перекрывает старое, подпрограмма аннулирует старое окно, а его отображение, если оно было, отменяет. Если при создании нового окна пришлось аннулировать старое, подпрограмма устанавливает WS.ELW в слове состояния окна, а если при этом пришлось отменить отображение, устанавливает и WS.UNM.

В байт 52 заносится код ошибки 1, если подпрограмма не находит свободного УБОК.

Нормальная обработка запроса завершается модификацией структур данных. Соответствующие значения заносятся в УБОК (W.BSIZ, W.BLVR и W.BFPD), идентификатор окна заносится в БООК (W.NID), и в слове состояния окна устанавливается WS.CRW.

8.4. Аннулировать окно (.ELAW)

Этот запрос аннулирует ранее созданное виртуальное адресное окно.

Формат макрокоманды: .ELAW AREA[,WADR]
где AREA — адрес блока аргументов EMT, имеющего вид:

```
AREA:   .BYTE  3,36
        .WORD  WADR
```

WADR — адрес идентификатора окна, которое надо аннулировать.

Аналогично .ELRG, подпрограмма .ELAW сначала находит соответствующий УБОК и проверяет на допустимость W.NID в БООК. В случае нулевого или недопустимого идентификатора окна в байт 52 заносится код ошибки 3. После

этого подпрограмма очищает WS.CRW и WS.UNM в слове состояния окна.

Если окно было отображено, подпрограмма отменяет отображение и устанавливает WS.UNM. Подпрограмма аннулирует окно, очищая W.BSIZ в УБОК. Завершая обработку запроса, подпрограмма устанавливает WS.ELW в слове состояния окна.

8.5. Отображение окна (.MAP)

Этот запрос отображает ранее созданное виртуальное адрестное окно в динамическую область расширенной памяти.

Формат макрокоманды: .MAP AREA[,WADR]

где AREA — адрес блока аргументов EMT, имеющего вид:

```
AREA:   .BYTE  4,36
        .WORD  WADR
```

WADR — адрес блока определения окна, которое надо отобразить; в него должен быть заранее занесен идентификатор области, в которую будет выполняться отображение (этот аргумент необязателен, если пользователь сам заполнил 2-е слово блока аргументов).

Подпрограмма монитора .MAP сначала находит УБОК, соответствующий указанному в запросе окну. Для этого проверяется W.NID, и если значение его нулевое или недопустимое, в байт 52 заносится код ошибки 3. После этого подпрограмма находит УБОБ области, в которую отображается окно, и если идентификатор области имеет недопустимое значение, в байт 52 заносится код ошибки 2.

Подпрограмма далее проверяет находящееся в БООК (W.NOFF) смещение в области, начиная с которого будет отображаться окно. Если смещение выходит за пределы области, в байт 52 заносится код ошибки 4.

Подпрограмма проверяет находящуюся в БООК (W.NLEN) длину отображаемого окна. Если значение W.NLEN нулевое, то для отображения окна подпрограмма отводит всю остальную часть области — от смещения и до конца области. Если эта память больше, чем окно, подпрограмма определяет необходимый для отображения размер и заносит его в W.NLEN. Таким образом, если до запроса .MAP пользователь указал нулевое значение W.NLEN, то после выполнения запроса W.NLEN содержит фактическую длину отображенного окна.

Если же пользователь указал ненулевое значение W.NLEN, т. е. задал длину явно, подпрограмма проверяет ее на допустимость. Если длина отображаемой части окна больше раз-

мера самого окна, или окно выходит за пределы области, в байт 52 заносится код ошибки 4.

Подпрограмма затем увеличивает находящийся в УБОБ счетчик отображенных в эту область окон (R.BNWD).

Если окно уже было куда-либо отображено, подпрограмма отменяет старое отображение и устанавливает WS.UNM в слове состояния окна; в противном случае WS.UNM подпрограмма очищает.

После этого подпрограмма загружает пользовательский набор PAC/РПС соответствующими значениями для отображения данного окна в данную область.

Завершая обработку запроса .MAP, подпрограмма обновляет следующие величины в УБОК: W.BRCB, W.BHVR, W.BOFF, W.BNPD и W.BLPD.

8.6. Отменить отображение (.UNMAP)

Запрос .UNMAP отменяет действие запроса .MAP для окна (т. е. отменяет отображение) и восстанавливает в этом окне обычную адресацию.

Формат макрокоманды: .UNMAP AREA[,WADR]
где AREA — адрес блока аргументов EMT, имеющего вид:

```
AREA:   .BYTE  5,36
        .WORD  WADR
```

WADR — адрес блока определения.

Подпрограмма .UNMAP монитора сначала находит соответствующий УБОБ и проверяет W.NID в БООК. Если его значение нулевое или недопустимое, в байт 52 заносится код ошибки 3. Если окно в данный момент не отображено, в байт 52 заносится код ошибки 5.

Для того, чтобы отменить отображение окна, подпрограмма модифицирует соответствующие структуры данных: очищает W.BRCB в УБОК и уменьшает R.BNWD в УБОБ.

Если задание виртуальное, подпрограмма очищает регистры РПС, соответствующие этому окну, так что программа пользователя больше не может обращаться к виртуальным адресам в этом окне.

Если задание привилегированное, монитор копирует содержимое регистров РПС внутреннего режима в регистры РПС режима пользователя. Таким образом, отображением по умолчанию становится отображение внутреннего режима.

Завершая обработку запроса, подпрограмма устанавливает WS.UNM в слове состояния окна.

8.7. Запись состояния окна отображения (.GMCX)

Запрос .GMCX записывает информацию о состоянии созданного окна в указанную пользователем область памяти.

Состояние окна записывается в формате БООК и может быть использовано в последующих операциях отображения. .GMCX позволяет модифицировать окна посредством изменения отдельных полей БООК. Это форма .SAVESTATUS для адресных окон.

Для созданного, но не отображенного окна запрос .GMCX модифицирует следующие поля БООК:

1) W.NARP — номер регистра PAC/РПС окна (определяет начальный виртуальный адрес окна);

2) W.NBAS — виртуальный базовый адрес окна;

3) W.NSIZ — размер окна в блоках по 32 слова.

Если окно, состояние которого запрашивается, отображено в область, то запрос .GMCX наряду с вышеперечисленными полями модифицирует следующие дополнительные поля:

4) W.NRID — идентификатор области;

5) W.NOFF — значение смещения в область;

6) W.NLEN — фактическая длина отображенного окна;

7) W.NSTS — состояние разряда WS.MAP; устанавливается в слове состояния окна.

Формат макрокоманды: .GMCX AREA[,WADR]
где AREA — адрес блока аргументов EMT, имеющего вид:

```
AREA:   .BYTE  6,36
        .WORD  WADR
```

WADR — адрес области памяти, куда будет записано состояние отображения окна.

Подпрограмма .GMCX монитора сначала находит соответствующий УБОК. Если пользователь указал окно с нулевым идентификатором, то будет получено состояние статического окна для виртуального задания. Для привилегированного задания окон с нулевым идентификатором не существует, и тогда в байт 52 заносится код ошибки 3.

Подпрограмма устанавливает W.NAPR в БООК равным трем старшим разрядам W.BLVR из УБОК.

Далее подпрограмма заносит соответствующие значения в находящиеся в БООК W.NBAS, W.NSIZ и W.NRID.

Если окно в данный момент не отображено, подпрограмма очищает находящиеся в БООК W.NOFF, W.NLEN и W.NSTS. Если окно отображено, подпрограмма помещает в W.NOFF смещение в области, в W.NLEN длину отображенного окна и устанавливает WS.MAP в слове состояния окна.

9. СООБЩЕНИЯ

Во время выполнения запросов расширенной памяти монитор XM выполняет проверку ошибок и управление состоянием. Все программные запросы расширенной памяти генерируют коды ошибок. Если ошибки обнаружены, то устанавливается разряд C, и код ошибки записывается в байт ошибки (байт 52).

В дополнение к кодам ошибок предусмотрено два слова состояния (одно в блоке определения окна, другое в блоке определения области) для регистрации состояния запрашиваемых операций.

После завершения запрашиваемых операций монитор XM устанавливает соответствующие разряды в слове состояния области или в слове состояния окна (в зависимости от типа запроса), чтобы указать, какие действия были произведены.

В табл. 6 и 7 приведены коды, выдаваемые запросами, и разряды слова состояния окна и области.

Таблица 6

КОДЫ ОШИБОК ЗАПРОСОВ РАСШИРЕННОЙ ПАМЯТИ

Код ошибки	Запросы							Причина
	.CRAW	.CRRG	.ELAW	.ELRG	.GMCX	.MAP	.UNMAP	
1	2							3
0	+							Слишком большое окно, или W.NAPR>7, или новое окно перекрывает статическое окно (для виртуальных заданий)
1	+							Нет свободных УБОК — попытка создать более 7 окон для задания. Отменить отображение одного из старых окон и создать новое, или перераспределить окна в виртуальном адресном пространстве
2				+		+		Указан недопустимый идентификатор области

1	2							3
3			+		+	+	+	Указан недопустимый идентификатор окна
4						+		Недопустимая комбинация «Смещение в области / размер отображаемого окна»
5							+	Указанное окно не было отображено
6		+						Нет свободных УБОБ — попытка создать более 3 областей для задания. Аннулировать старую область и затем создать новую, или перераспределить области в физической памяти. Следует также учесть, что память, получаемая по XM .SETTOP, и оверлеи расширенной памяти требуют отдельной области
7		+						Область требуемого размера не может быть создана. Размер наибольшей доступной области заносится в R0 (область при этом не создается!)
10		+						Недопустимый размер области — нулевой или размер более 96К слов

Таблица 7

СЛОВО СОСТОЯНИЯ РАСШИРЕННОЙ ПАМЯТИ

Слово состояния	Имя разряда	Номер разряда	Ввод-вывод	Маска	Комментарий
1	2	3	4	5	6
Слово состояния области (R.GSTS)	RS.CRR	15	Вывод	100000	Устанавливается, если область создана успешно
	RS.UNM	14	Вывод	40000	Устанавливается, если отображение одного или нескольких окон от-

1	2	3	4	5	6
					менено в результате аннулирования области
	RS.NAL	13	Вывод	20000	Устанавливается, если создать область нужного размера в данный момент невозможно
Слово состояния окна (W.NSTS)	WS.CRW	15	Вывод	100000	Устанавливается, если адресное окно создано успешно
	WS.UNM	14	Вывод	40000	Устанавливается, если отображение одного или нескольких окон отменено по запросу .CRAW или .MAP
	WS.ELW	13	Вывод	20000	Устанавливается, если одно или несколько окон по запросу .CRAW были аннулированы
	WS.MAP	8	Ввод	400	Установить, если окно отображается по запросу .CRAW

10. НЕКОТОРЫЕ ПРИЛОЖЕНИЯ РАСШИРЕННОЙ ПАМЯТИ

10.1. Оверлеи в расширенной памяти

Нижние 28К слов памяти заполняются очень быстро: здесь располагаются RMON, USR, драйверы устройств, а также основное, системное (их может быть несколько) и фоновое задания. Чтобы избежать перегруженности памяти и исполь-

зовать ее оптимально, следует сделать корневые сегменты основных, фоновых и системных заданий по возможности наименьшими. Вместо сегментирования программ и использования дисков оверлеи можно помещать в расширенную память. Если программы пользователя не требуют доступа к монитору или странице в—в, их следует сделать виртуальными заданиями.

Корневой сегмент может быть минимальным по размеру. В него необходимо включить элементы очереди, каналы, подпрограммы обслуживания прерываний (если они есть — в виртуальных заданиях их нет) и команда JMP к первому оверлею. Для ускорения выполнения оверлеи могут постоянно быть резидентными в расширенной памяти. Чтобы поместить оверлеи в расширенную память, надо использовать переключатель /V редактора связей. Клавиатурный монитор во время работы создает область, используя информацию из оверлейного драйвера и таблиц. Оверлейный драйвер создает и отображает окна. Более подробно этот материал изложен в [1].

10.2. Буферы или массивы в расширенной памяти

Чтобы поместить большой буфер или массив в расширенную память, следует сначала создать достаточно большую область, вмещающую этот массив, после чего определить, какое виртуальное адресное пространство программа пользователя может отвести для доступа к массиву, и создать окно такого размера. Затем надо написать подпрограмму, переводящую обращения к массиву в команды повторного отображения окна в соответствующую часть области. Этот случай показан на рис. 19 (XM .SETTOP может создать буфер в расширенной памяти автоматически).

10.3. Многопользовательские программы

Система расширенной памяти представляет собой удобное средство для реализации многопользовательских прикладных задач. Например, пользователь может разработать языковый интерпретатор, которым одновременно могут пользоваться несколько программистов. Для реализации этой задачи программу следует разделить на две секции: секцию команд и данных, содержащую интерпретатор, и отдельную рабочую область чтения/записи для каждого пользователя. Соответственно, часть виртуального адресного пространства следует отвести под рабочую область пользователя и создать окно требуемого размера. Кроме того, следует определить количество пользователей, которые будут работать с программой, и создать область, размер которой больше размера окна во столько же раз. Переключение на других пользователей ин-

терпретатор осуществляет с помощью повторного отображения.

Многопользовательская программа может использовать оверлеи в расширенной памяти. В этом случае одну из областей надо использовать для оверлеев и одну — в качестве рабочей области.

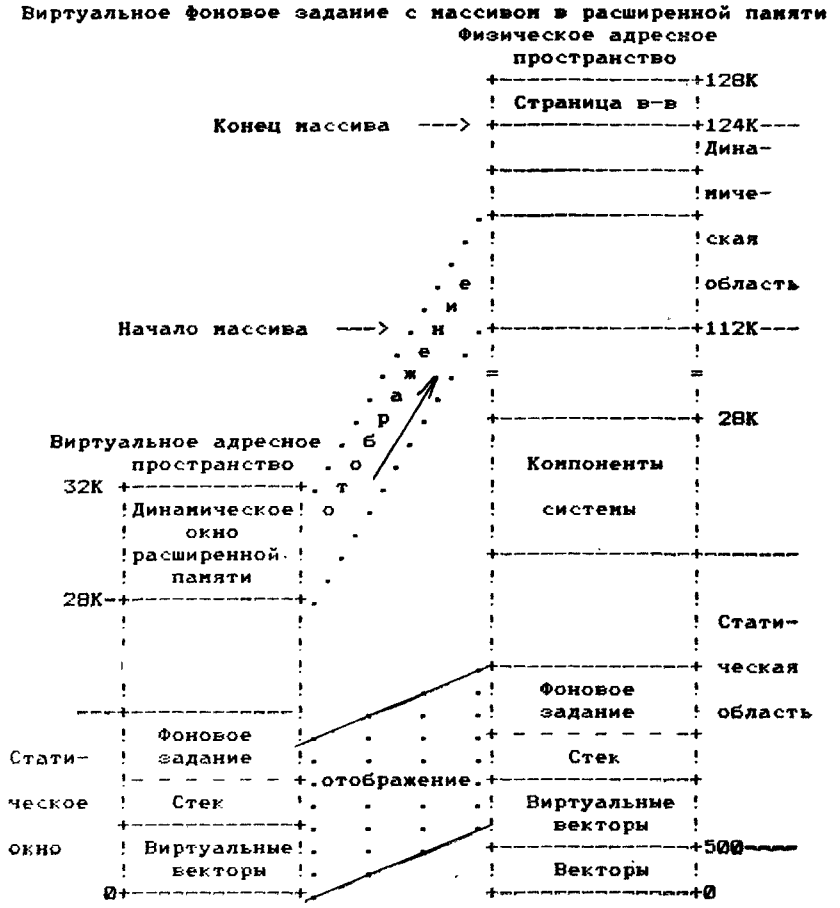


Рис. 19

10.4. Рабочая область в расширенной памяти

Рабочую область бывает удобно располагать в расширенной памяти, а не записывать ее на диск.

В мониторе FB фоновое задание получает дополнительную память с помощью запроса .SETTOP, она располагается выше задания, но ниже USR. Для основного задания дополнительную память можно отвести по команде FRUN/BUFFER:N. Когда память зарезервирована по команде FRUN, программа может определить ее размер и заказать ее по запросу .SETTOP. В обоих случаях дополнительная память располагается в нижних 28K слов памяти.

В мониторе XM дополнительная память может быть получена в физической памяти выше или ниже границы 28K слов. Эта особенность позволяет работать таким заданиям, которые требуют слишком много памяти, чтобы обойтись без отображений. Возможность получения дополнительной памяти весьма выгодна для виртуальных заданий, поскольку они могут получать память вплоть до виртуального адреса 177776 (32K слов) с помощью XM .SETTOP. Вся память, полученная по запросу .SETTOP, расположена в расширенной памяти; виртуальные основные задания не требуют команды FRUN/BUFFER:N для получения дополнительной памяти.

10.5. Использование XM .SETTOP

Существует два способа использовать XM .SETTOP. Если в программе есть оверлеи в расширенной памяти, то переключатель /V редактора связей использует XM .SETTOP автоматически. Он также использует XM — особенность директивы .LIMIT (далее в тексте — XM .LIMIT), включает оверлейный драйвер расширенной памяти (VHANDL) в файл задания и устанавливает оверлейную структуру расширенной памяти. Для этого надо подать команду монитора LINK/PROMPT и на следующей строке указать /V.

Если в программе нет оверлеев, или они — только в нижней памяти (переключатель /O редактора связей), то использовать XM .SETTOP следует по команде монитора LINK с переключателем /XM, при этом используется как XM .SETTOP, так и XM .LIMIT. Однако оверлейный драйвер расширенной памяти в файл задания не включается, и оверлейная структура расширенной памяти для программы не устанавливается.

Для всех программ директива .LIMIT выдает в качестве наибольшего значения очередную ячейку памяти, доступную для задания. Дополнительная память, получаемая программой по запросу .SETTOP в расширенной памяти, всегда начинается с восьмеричного адреса, который выдает директива .LIMIT. Это справедливо для всех программ независимо от того, использует ли программа XM .SETTOP.

Работа запроса .SETTOP без использования его XM-особенности в расширенной памяти показана в п. 10.5.2. Работа XM .SETTOP и XM .LIMIT показана в п. 10.5.3.

10.5.1. Виртуальный старший адрес и очередной свободный адрес

Для понимания XM-особенности запроса .SETTOP важно видеть различие между старшим адресом программы, виртуальным старшим адресом и очередным свободным адресом. На рис. 20 показано виртуальное адресное пространство программы, имеющей оверлеи как в нижней (переключатель /O), так и в расширенной памяти (переключатель /V редактора связей).

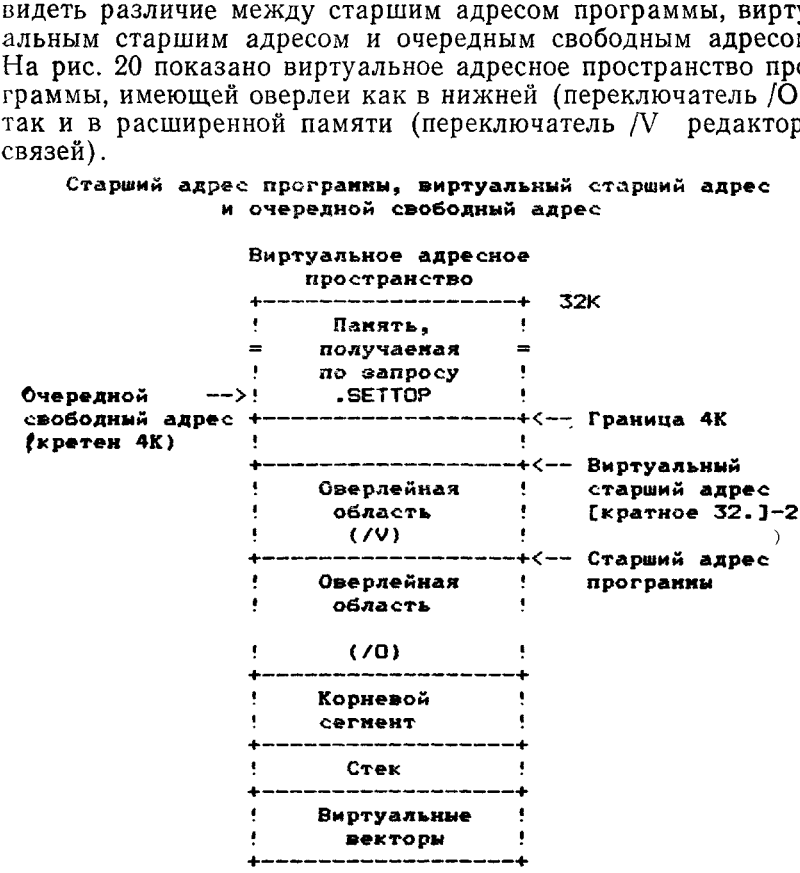


Рис.20

Старший адрес программы — это наибольший виртуальный адрес, используемый корневым сегментом программы и ее оверлеями в нижней памяти, если они есть.

Виртуальный старший адрес (ВСА) — это наибольший виртуальный адрес, используемый оверлеями расширенной памяти, округленный до 32 (десятичное), минус 2 (в восьмеричном представлении две младшие цифры адреса всегда 76).

Именно это значение печатается в карте загрузки в виде NNNNNN, как показано на следующем примере:

VIRTUAL HIGH ADDRESS =NNNNNN=DDDDD.WORDS
NEXT FREE ADDRESS =MMMMMM

Редактор связей должен вычислить значение очередного свободного адреса. Для задания, которое использовало XM .SETTOP, он округляет ВСА до очередной границы 4K слов. Таким образом, очередной свободный адрес (ОСА) — это адрес последнего слова виртуального адресного пространства, определяемого старшим РАС/РПС этого задания, плюс 2. Он всегда расположен на границе 4K слов (восьмеричный — всегда кратен 20000).

В качестве примера предлагается задание с оверлеями в расширенной памяти, ВСА которого равен 55076. Его очередной свободный адрес, вычисленный редактором связей, равен 60000 — это начало следующих 4K слов виртуального адресного пространства. Именно это значение выдается в карте загрузки в качестве очередного свободного адреса (NEXT FREE ADDRESS). Для приводимого примера эта строка в карте загрузки выглядит так:

VIRTUAL HIGH LIMIT =055076=DDDDD. WORDS
NEXT FREE ADDRESS =060000

Конечно, если в программе нет оверлеев расширенной памяти, то у нее нет и ВСА, и старший адрес программы не округляется. В картах загрузки программ, не имеющих оверлеев или имеющих только оверлеи, полученные по /O, — старший адрес программы печатается в виде MMMMMM, как показано на следующем примере (эта строка выдается во всех картах загрузки независимо от того, имеется ли расширенная память или нет).

TRANSFER ADDRESS=NNNNNN,
HIGH LIMIT=MMMMMM= DDDDD. WORDS

10.5.2. .SETTOP без XM-особенности

Если с помощью редактора связей XM-особенность .SETTOP не используется, то .SETTOP имеет в расширенной памяти весьма ограниченное применение.

Для привилегированного задания, не меняющего отображение по умолчанию, .SETTOP работает так же, как и в мониторах SJ или FB. Если привилегированное задание создает виртуальное адресное окно и отображает его в область расширенной памяти, то на старший адрес программы отображение не влияет. Величина, полученная по запросу .SETTOP и обозначающая наибольший доступный программе адрес, по-прежнему находится в нижних 28K слов памяти.

Когда монитор выполняет адресную проверку для программных запросов, он сначала следит за тем, попадает ли адрес (блока аргументов, буфера данных и т. п.) в отображенное динамическое окно. Если нет, то монитор следит за тем, попадает ли адрес в область задания в нижней памяти. Если адрес не выдерживает обеих проверок, то монитор выдает сообщение об ошибке, и задание прерывается.

Если задание виртуальное, то во время загрузки старший адрес программы устанавливается равным старшему виртуальному адресу, который использует корневой сегмент и любые оверлеи в нижней памяти (/O). Если задание выполняет собственные операции отображения, то они не влияют на старший адрес программы в том плане, в каком рассматривается .SETTOP. Так что запрос .SETTOP без XM-особенности не имеет значения для этих виртуальных заданий, поскольку работает исключительно в нижних 28К слов памяти. Виртуальные задания используют режим пользователя и, следовательно, отображаются в соответствии с содержимым пользовательского набора PAC/PПC. Виртуальное задание не имеет доступа к не предназначенным для него областям памяти, поскольку отображается только в отведенное для него пространство. Поэтому обращение к запросу .SETTOP, если не подавалась команда монитора LINK/XM или /V в редакторе связей, не добавит дополнительной памяти. Получаемую величину виртуальное задание может использовать для собственного отображения имеющейся области и последующего ее использования.

Когда монитор выполняет адресную проверку для виртуального задания, он игнорирует границы программы и проверяет только, входит ли виртуальный адрес в окно, которое в настоящий момент отображается. Если не входит, то происходит сбой диспетчера памяти.

10.5.3. XM .SETTOP может быть весьма полезен для привилегированных и виртуальных заданий (для привилегированных, однако, в меньшей степени).

Для виртуальных заданий .SETTOP не только получает виртуальное адресное пространство выше виртуального старшего адреса, начиная с OCA, — .SETTOP также автоматически отображает дополнительную память в физическую область памяти. В итоге задание в среде расширенной памяти может выдавать запрос .SETTOP и получать необходимое ему виртуальное адресное пространство, не заботясь о тонкостях управления расширенной памятью.

Для привилегированных заданий XM .SETTOP работает

так же, как и .SETTOP без XM-особенности, если не считать того, что для привилегированных заданий XM .SETTOP в качестве очередного свободного адреса использует новое значение, которое выдает XM .LIMIT. Начало буфера, таким образом, всегда оказывается на границе 4K слов. Указывать для .SETTOP любой адрес, лежащий ниже этой границы 4K, не разрешается.

Для привилегированных и виртуальных программ редактор связей помещает в ячейки 0 и 2 файла задания два информационных слова. Ячейка 0 содержит VIR в кодах RADIX-50. Ячейка 2 содержит значение ОСА-2, который может существенно отличаться от виртуального старшего адреса.

10.5.3.1. Директива .LIMIT

Для заданий, работающих под управлением мониторов SJ и FB, а также под управлением монитора XM, если не используется XM .SETTOP, директива .LIMIT выдает программе два значения, именно:

— самый младший виртуальный адрес, используемый программой (обычно 0);

— старший адрес программы плюс 2 (например, 1644+2==1646).

В программах расширенной памяти, использующих XM .SETTOP, директива .LIMIT выдает существенно отличное значение:

— самый младший виртуальный адрес, используемый программой (обычно 0);

— очередной свободный адрес (всегда на границе 4K), который обычно не равен старшему адресу программы +2.

10.5.3.2. «Пустоты» в виртуальном адресном пространстве

Редактор связей всегда располагает оверлейную область расширенной памяти (/V) на границе 4K в виртуальном адресном пространстве программы пользователя. Это ограничение обусловлено аппаратными требованиями. Вследствие этого могут быть «пустоты» между старшим адресом программы и началом виртуальной оверлейной области. Попытка обращения программы пользователя к виртуальным адресам внутри такой «пустоты» вызовет сбой системы. Аналогично, любое дополнительное виртуальное адресное пространство, полученное программой с помощью XM .SETTOP, также начинается на границе 4K слов. Это означает, что существует «пустота» между виртуальным старшим адресом программы и началом дополнительной памяти. Программа не может обращаться к **адресам внутри этой «пустоты».**

10.5.4. XM .SETTOP и привилегированные задания

Когда привилегированное задание выдает запрос .SETTOP и очередной свободный адрес выше основания USR, то программа уже использует виртуальное адресное пространство выше начала монитора. Поскольку свободной памяти, которую можно было бы отобразить, начиная с OCA программы, нет, монитор не может получить дополнительную память для этой программы. Таким образом, привилегированное задание никогда не может получить память выше SYSLOW — основания USR. Запрос .SETTOP выдает значение OCA-2 в ячейку 50 программы и в R0. Это наибольший используемый адрес.

Если память в наличии имеется, монитор пытается получить ее, определяя размер области по указанному пользователем в запросе .SETTOP аргументу. Память эта всегда располагается в пределах нижних 28K слов. Привилегированное задание никогда не может получить область виртуального адресного пространства ниже, чем OCA-2. Кроме того, OCA, полученный с помощью XM .SETTOP, всегда располагается на границе 4K слов, и задание не может указывать в запросе .SETTOP адрес, лежащий ниже. Следовательно, задание теряет пространство между своим последним адресом и очередной границей 4K слов.

Когда в памяти нет основного задания, привилегированное фоновое задание может получить некоторую область памяти с помощью .SETTOP. Доступная память часто имеется и тогда, когда в памяти присутствует и основная программа.

Поскольку основные задания загружаются в память ниже последнего драйвера устройства и выше USR, то получить дополнительную память по запросу .SETTOP они не могут. Вследствие этого, привилегированным основным заданиям запрещается использовать оверлеи в расширенной памяти. Это также означает, что они не могут указать переключатель /V редактора связей (или команду LINK/FOREGROUND /PROMPT или LINK/FOREGROUND/XM), чтобы использовать XM .SETTOP и XM .LIMIT.

10.5.5. XM .SETTOP и виртуальные задания

Монитор проверяет, имеется ли в наличии какая-либо доступная расширенная память. Если OCA равен 200000, то программа уже использует виртуальное адресное пространство, которое контролирует PAC7. Запрос выдает значение 177776 в ячейку 50 и в R0.

Если .SETTOP может получить дополнительную память, начиная с очередного свободного адреса (на границе 4K), то

монитор создает область необходимого размера в расширенной памяти. Если этой области недостаточно, монитор создает наибольшую возможную область (значение, которое выдает .SETTOP, надо проверить). Затем монитор создает окно и отображает его в новую область. Новое значение наибольшего доступного адреса он возвращает в ячейку 50 и в R0. Если нет в наличии доступной памяти, или УБОБ или УБОК, то в ячейку 50 и в R0 выдается первоначальное значение наибольшего доступного адреса.

Например, если программа выдает запрос .SETTOP с адресным аргументом, то монитор отображает виртуальное адресное пространство, начиная с очередной границы 4K слов выше ВСА программы до указанного адреса включительно. При этом отображается указанный адрес, но может быть отображено также до 31 (десятичное) слова дополнительно.

Если указанный в запросе .SETTOP адрес расположен ниже, чем наибольший используемый адрес, то .SETTOP выдает значение ОСА-2 в ячейку 50 и R0. Статическое окно и виртуальные оверлейные области, созданные по переключателю /V редактора связей, нельзя аннулировать с помощью аргумента в запросе .SETTOP.

Если первый запрос .SETTOP из программы завершился успешно и область расширенной памяти для программы имеется, то программа может и далее подавать запросы .SETTOP для контроля области, однако, дополнительной памяти эти запросы уже не дадут.

Если указанный в очередном .SETTOP аргумент меньше, чем указанный в карте связей первоначальный ОСА-2, то монитор возвращает в ячейку 50 и R0 старый ОСА-2 и аннулирует область и окно, если они есть (вместе со всей хранящейся там информацией). Конечно, позднее, можно опять подать запрос .SETTOP и создать новую область. Можно также изменить размер буфера, выполнив повторное отображение в ту же самую область.

Чтобы получить область большего размера, следует сначала подать запрос .SETTOP с аргументом, значение которого меньше, чем старший адрес в настоящее время. В результате область и все данные в ней будут аннулированы. Затем следует подать другой запрос .SETTOP и указать в нем большее значение; в результате будет создана новая область (любые данные из первого буфера будут потеряны). Следует отметить, что для обеспечения целостности данных пользователя существует только одно окно для области .SETTOP в системе расширенной памяти.

Чтобы получить область памяти меньшего размера, чем по предыдущему запросу .SETTOP, следует подать другой .SETTOP, аргумент которого меньше предыдущего, но больше или равен OCA. В результате этого размер окна по-прежнему равен размеру области, но отображается меньшая часть окна.

Виртуальные фоновые и основные задания наиболее часто используют XM .SETTOP. Это позволяет им легко и быстро создавать обширные буферы в расширенной памяти, что способствует разгрузке нижней памяти.

Для основных заданий запрос .SETTOP работает в целом так же, как и для фоновых заданий. Для виртуальных основных заданий, не использующих XM .SETTOP, единственно доступная дополнительная область может быть получена по команде FRUN/BUFFER:N. Для заданий, использующих XM .SETTOP, переключатель /BUFFER игнорируется (задание не может иметь буферов и в нижней, и в расширенной памяти).

10.5.6. Сводка результатов действия запроса .SETTOP

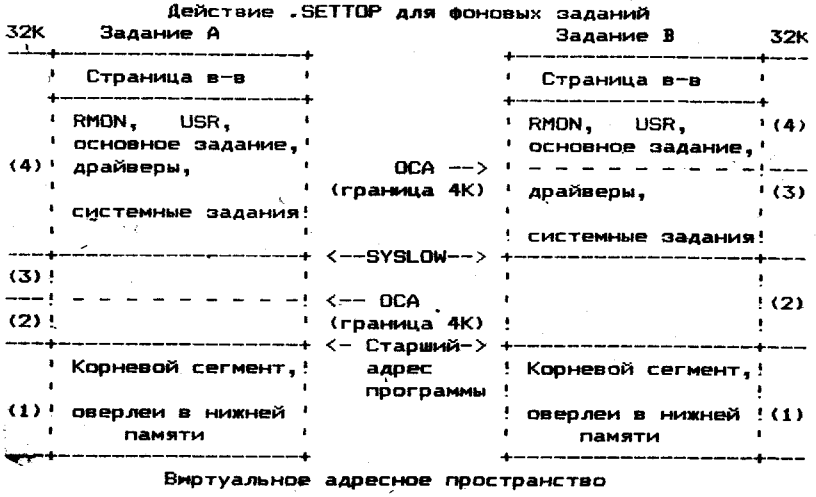


Рис. 21

На рис. 21 и 22 и в табл. 8 и 9 приводится сводка результатов действия запроса .SETTOP. Задание А на рис. 21 — это фоновое задание, OCA которого располагается ниже SYSLOW (основание USR). В скобках приведены области значений аргументов запроса .SETTOP.

Таблица 8

ДЕЙСТВИЕ SETTOP ДЛЯ ФОНОВЫХ ЗАДАНИЙ

	Виртуальное задание		Привилегированное задание	
1	2		3	
Аргумент .SETTOP	.SETTOP без XM-особенности	XM .SETTOP	.SETTOP без XM-особенности	XM .SETTOP
Старший адрес задания A после запроса .SETTOP				
(1) (2) (3)	(1) (2) (3)	OCA - 2 Отображ. 1) в (3)	(1) (2) (3)	OCA - 2 OCA - 2 (3)
(4)	SYSLOW - 2	Отображ. 1) в (4)	SYSLOW - 2	SYSLOW - 2
#0 # - 2	0 SYSLOW - 2	OCA - 2 Отображ. 1) в 32K	0 SYSLOW - 2	OCA - 2 SYSLOW - 2
Старший адрес задания B после запроса .SETTOP				
(1) (2) (3) (4)	(1) (2) SYSLOW - 2 SYSLOW - 2	OCA - 2 OCA - 2 OCA - 2 Отображ. 1) в (4)	(1) (2) SYSLOW - 2 SYSLOW - 2	OCA - 2 OCA - 2 OCA - 2 OCA - 2
#0 # - 2	0 SYSLOW - 2	OCA - 2 Отображ. 1) в 32K	0 SYSLOW - 2	OCA - 2 OCA - 2

1)при наличии памяти; в противном случае для области .SETTOP выделяется наибольшая возможная память.

Таблица 9

СТАРШИЙ АДРЕС ОСНОВНОГО ЗАДАНИЯ ПОСЛЕ .SETTOP

Аргумент .SETTOP	Виртуальное задание	
1	2	
	.SETTOP без XM-особенности	XM .SETTOP
(1)	(1)	OCA - 2
(2)	Выше OHIGH или BUFF	OCA - 2
#0	0	OCA - 2
# - 2	Выше OHIGH или BUFF	Отображение в 32K

Рис. 22

ПЕРЕЧЕНЬ ССЫЛОЧНЫХ ДОКУМЕНТОВ

1. Операционная система ФОДОС-2
   Редактор связей
   Руководство оператора
2. Операционная система ФОДОС-2
   Системная макробиблиотека

КОНТРОЛЬНЫЕ ЗАДАЧИ. РУКОВОДСТВО ОПЕРАТОРА.

1. КОНТРОЛЬНАЯ ЗАДАЧА N1

1.1. Назначение программы и условия выполнения программы

Контрольная задача N1 (программа CONTBG) предназначена для проверки работы операционной системы ФОДОС-2 под управлением монитора одного задания.

Методика проверки операционной системы ФОДОС-2 включает:

1) Внесение изменений в исходный текст программы CONTBG;

2) Транслирование программы CONTBG и получение листинга;

3) Получение загрузочного модуля программы CONTBG;

4) Запуск программы CONTBG и управление ее выполнением.

Для выполнения программы CONTBG операционная система должна включать следующие системные программы:

FMONSJ.SYS
DW.SYS
LP.SYS
EDIT.SAV
MACRO.SAV
LINK.SAV
PIP.SAV
DUP.SAV
DIR.SAV
MX.SYS
MY.SYS
SYSMAC.SML
CONTBG.MAC
SWAP.SYS

1.2. Выполнение программы

Для обеспечения загрузки и выполнения программы CONTBG пользователю необходимо загрузить операционную систему ФОДОС-2 (см. книгу 1).

Работа с программой CONTBG предусматривает следующую последовательность операций:

1) Установить чистый гибкий диск на 1-ый привод и подать команду:
ASSIGN ZZ DK <BK>
Ответ: .
Печать: INITIALIZE ZZ: <BK>
Ответ: ZZ:/INIT; ARE YOU SURE?
Печать: Y <BK>
Ответ: .
где ZZ — MX1 или MY1;

2) Использовать редактор текста для изменения исходной программы CONTBG, подав команду
EDIT SY:CONTBG.MAC <BK>
Ответ: *
Печать: F;CУ/I.ASCII$$
Ответ: *
Печать: OAD$$
Ответ: *
Печать: EX$$
Ответ: .

3) Если конфигурация ЭВМ включает построчно-печатающее устройство — подать команду
ASSIGN LP LST <BK>
Ответ: .
Печать: SET LP LC <BK>
Ответ: .

Если построчно-печатающего устройства нет — подать команду
ASSIGN TT LST <BK>
Ответ: .

4) Протранслировать CONTBG.MAC и получить листинг. Подать команду
MACRO/LIST:LST:/ENABLE:LC SY:CONTBG <BK>
Ответ: ERRORS DETECTED:0

.
Печатается листинг CONTBG на устройстве LST:;

5) Получить загрузочный модуль, используя редактор связей, вызвать программу CONTBG;

Печать: LINK/EXECUTE:SY: CONTBG <ВК>
Ответ: .
Печать: R CONTBG <ВК>
Ответ: фоновая программа CONTBG.
Если изменения в исходный
текст внесены неверно — то
это строка последняя.
Изменения внесены правильно.
Печать: СУ/С
СУ/С
Ответ: ^C
Ответ: ^C

На этом проверка операционной системы ФОДОС-2 под управлением монитора одного задания закончена.

ПРИМЕЧАНИЕ. Если изменения в исходную программу внесены неправильно (в этом случае строка «Изменения внесены правильно» в 5) не печатается), то подать команду: RENAME SY:CONTBG.BAK SY:CONTBG.MAC <ВК>
и перейти к 2).

2. КОНТРОЛЬНАЯ ЗАДАЧА N2

2.1. Назначение программы и условия выполнения программы

Контрольная задача N2 (программа CONTFG) предназначена для проверки работы операционной системы ФОДОС-2 в реальном масштабе времени под управлением монитора двух заданий.

Методика проверки операционной системы ФОДОС-2 включает:

1) Транслирование программы CONTFG;

2) Получение загрузочного модуля программы CONTFG;

3) Загрузку программы CONTFG в основную область памяти и ее запуск;

4) Загрузку программы CONTBG в фоновую область памяти и ее запуск;

5) Управление выполнением программ.

Для выполнения программы CONTFG необходимо не менее 16К слов оперативной памяти. Программа CONTFG выполняется в основной области памяти, посылая каждые 5 се-

кунд сообщение программе CONTBG, работающей в фоновой области памяти. CONTBG принимает сообщение и печатает его на терминале, сопровождая печать сообщения посылкой кода 007 (для дисплея — подачей звукового сигнала). Если CONTBG не загружена, то CONTFG ставит сообщения в очередь. После загрузки программы CONTBG печатаются все сообщения, которые поставлены в очередь и, как только очередь сообщений станет пустой, возобновится заданный цикл передачи-приема сообщений, т. е. печать сообщения на терминале, сопровождаемая посылкой кода 007, будет происходить каждые 5 секунд.

Для выполнения программы CONTFG операционная система должна включать, по крайней мере, следующие системные программы:
FMONFB.SYS
FMONSJ.SYS
DW.SYS
MX.SYS
MY.SYS
LP.SYS
MACRO.SAV
LINK.SAV
PIP.SAV
DUP.SAV
DIR.SAV
EDIT.SAV
SYSMAC.SML
CONTBG.SAV
CONTFG.MAC
SWAP.SYS

2.2. Выполнение программы

Для обеспечения загрузки и выполнения программы CONTFG пользователю необходимо загрузить операционную систему ФОДОС-2 (см. п. 1.2.).

Работа с программой CONTFG предусматривает следующую последовательность операций:

1) Для загрузки в память монитора двух заданий подать команду
BOOT FMONFB.SYS <ВК>
Ответ: ФОДОС Ф/0 В03.00
Стартовый файл [имфайл.тип]?
Подать команду STARTS<ВК>
Включить таймер.

2) Установить чистый гибкий диск на 1-ый привод и подать команду ASSIGN ZZ DK <BK>

Ответ:
Печать: INITIALIZE ZZ <BK>
Ответ: ZZ:/INIT; ARE YOU SURE?
Печать: Y <BK>
Ответ: .
где ZZ — MX1 или MY1;

3) Протранслировать программу CONTFG.MAC подав команду

MACRO/ENABLE:LC SY:CONTFG <BK>
Ответ: ERRORS DETECTED:0
.

4) Получить загрузочный модуль программы CONTFG для работы в основной области памяти, подав команду LINK/FOREGROUND CONTFG <BK>
Ответ: .

5) Загрузить CONTFG в основную область памяти, подав команду FRUN CONTFG <BK>
Ответ: .
F>
Основная программа CONTFG посылает
каждые 5 секунд сообщение
фоновой программе CONTBG.
Программа CONTBG печатает
принятое сообщение на
терминале.
Печать: СУ/В
Ответ: В>

6) Загрузить CONTBG в фоновую область памяти, подав команду R CONTBG <BK>
Ответ: фоновая программа CONTBG.
Если изменения в исходный
текст внесены неверно — то
эта строка последняя.
Изменения внесены правильно.
— Работает основная программа —
— работает основная программа —

(Печатаются все сообщения, которые поставлены в очередь программой CONTFG, и как только очередь сообщений станет пустой, печать сообщений будет происходить через каждые 5 секунд);

7) Освободить фоновую область памяти;

Печать: СУ/С
СУ/С
Ответ: ^C
^C

8) Освобожденную (от программы CONTBG) фоновую область памяти можно использовать для работы с любой системной программой, например, с помощью программы получения справочника, напечатать на терминале справочник устройства DK:, подав команду DIRECTORY <ВК>
Ответ: дд—ммм—гг
(печатается справочник устройства DK:)

9) Повторно загрузить программу CONTBG для печати на терминале сообщений, которые были поставлены в очередь программой CONTFG в то время, когда печатался справочник устройства DK:, подав команду R CONTBG
Ответ: фоновая программа CONTBG.
Если изменения в исходный
текст внесены неверно — то
эта строка последняя.
Изменения внесены правильно.
— Работает основная программа —
— Работает основная программа —

10) Освободить фоновую область памяти, дважды подав команду СУ/С
Печать: СУ/С
СУ/С
Ответ: ^C
^C

Прием сообщений прекращен;

11) Для того, чтобы освободить основную область памяти — подать команду
СУ/F
Ответ: F>
Печать: СУ/С
СУ/С
Ответ: ^C
^C
B>

Печать: UNLOAD F <ВК>

На этом проверка работы операционной системы ФОДОС-2 под управлением монитора двух заданий закончена.

ПРИМЕЧАНИЕ. Для повторного выполнения контрольных задач подать команду RENAME SY:CONTBG.BAK SY:CONTBG.MAC <ВК> и перейти к п. 1.2.

3. КОНТРОЛЬНАЯ ЗАДАЧА N3

3.1. Назначение программы и условия выполнения программы

Контрольная задача N3 (программа CONTXM) предназначена для проверки работы операционной системы ФОДОС-2 под управлением монитора расширенной памяти.

Методика проверки операционной системы ФОДОС-2 включает:

1) Транслирование программы CONTXM;
2) Получение загрузочного модуля CONTXM;
3) Загрузка программы CONTXM в память и ее запуск;
4) Управление выполнением программы.

Операционная система должна включать, по крайней мере, следующие системные программы:

FMONXM.SYS
FMONSJ.SYS
MYX.SYS
MXX.SYS
DWX.SYS
MX.SYS
MY.SYS
RK.SYS
PIP.SAV
DUP.SAV
DIR.SAV
CONTXM.MAC
SWAP.SYS
MACRO.SAV
LINK.SAV
SYSMAC.SML

3.2. Выполнение программы

Для обеспечения загрузки и выполнения программы CONTXM пользователю необходимо загрузить операционную систему ФОДОС-2.

Работа с программой CONTXM предусматривает следующую последовательность операций:

1) Для загрузки в память монитора расширенной памяти подать команду *BOOT FMONXM.SYS* <ВК>
Ответ: ФОДОС РФ/0 В03.00
Стартовый файл [имфайл.тип]?
Подать команду STARTS<ВК>
Ответ: .

2) Установить чистый гибкий диск на 1-ый привод и подать команду: ASSIGN ZZ DK <ВК>
Ответ: .
Печать: INITIALIZE ZZ: <ВК>
Ответ: ZZ:/INIT; ARE YOU SURE?
Печать: Y <ВК>
Ответ: .
где ZZ — MX1 или MY1;

3) Протранслировать CONTXM.MAC
MACRO/ENABLE:LC SY:CONTXM
Ответ: ERRORS DETECTED:0

.

4) Получить загрузочный модуль, вызвав редактор связей
Печать: LINK/EXECUTE:SY:CONTXM
Ответ: .

5) Выполнить программу CONTXM, подав команду
R CONTXM <ВК>
(На терминале печатается справочная информация о работе программы CONTXM и указание формата командной строки для работы с программой CONTXM)
Ответ: напечатать спецификации выходного и входного файлов и нажать клавишу <ВК>

Программа выполняет копирование и сравнение файлов. По завершении работы программа выдает сообщение:
КОПИРОВАНИЕ ФАЙЛОВ ЗАВЕРШЕНО
СРАВНЕНИЕ ДАННЫХ ЗАВЕРШЕНО
НЕСОВПАДЕНИЙ НЕ ОБНАРУЖЕНО
ПРОГРАММА CONTXM ОТРАБОТАЛА

На этом проверка работы операционной системы ФОДОС-2 под управлением монитора расширенной памяти закончена.

4. КОНТРОЛЬНАЯ ЗАДАЧА N4

4.1. Назначение программы и выполнение программы

Контрольная задача N4 (CONT) предназначена для проверки программной совместимости программ, разработанных

в рамках операционной системы ФОДОС-2, под управлением мониторов расширенной памяти XM и однозадачного SJ. Программа CONT вычисляет иррациональное число «E».

Работа с программой CONT предусматривает следующую последовательность операций:

1) выполнить трансляцию, получить загрузочный модуль и запустить программу CONT под управлением монитора XM:
Печать: EXECUTE CONT <BK>
Ответ: E=2,718281828...
где E — результат вычисления;

2) выполнить трансляцию, получить загрузочный модуль и запустить программу CONT под управлением монитора SJ:
Печать: EXECUTE CONT <BK>
Ответ: E=2,718281828...
где E — результат вычисления;

3) результаты, полученные при выполнении программы CONT под управлением мониторов XM и SJ должны совпадать.

ПРОГРАММА ПАКЕТНОЙ ОБРАБОТКИ РУКОВОДСТВО ОПЕРАТОРА

1. НАЗНАЧЕНИЕ ПРОГРАММЫ И УСЛОВИЯ ВЫПОЛНЕНИЯ ПРОГРАММЫ

Программа пакетной обработки (BATCH) предназначена для выполнения пакета заданий.

Программа BATCH позволяет:

1) выполнять пакетную обработку под управлением монитора одного задания (SJ) или в фоновом режиме под управлением мониторов основного-фонового задания (FB, XM);

2) вводить поток заданий с любого устройства, поддерживаемого системой ФОДОС-2;

3) использовать команды клавиатурного монитора (КЯС) в потоке заданий;

4) выводить регистрационный файл на любое устройство, поддерживаемое системой ФОДОС-2 (кроме MT:).

Для работы BATCH под управлением монитора SJ необходимо, как минимум, 12К слов оперативной памяти. Для работы BATCH в фоновом режиме под управлением монитора FB необходимо, как минимум, 16К слов оперативной памяти.

Для выполнения своих операций BATCH использует системные программы ФОДОС-2;

BASIC.SAV (если используются программы на языке БЕЙСИК)

BA.SYS

BATCH.SAV

CREF.SAV (если используются программы на языке АССЕМБЛЕР)

SYSLIB.OBJ (если используются программы на языке ФОРТРАН и АССЕМБЛЕР)

FORTRA.SAV (если используются программы на языке ФОРТРАН)

LINK.SAV
MACRO.SAV (если используются программы на языке АССЕМБЛЕР)
SYSMAC.SML (если используются программы на языке АССЕМБЛЕР)
PIP.SAV
DIR.SAV

Перед работой с BATCH необходимо убедиться, что на системном устройстве находятся перечисленные выше файлы.

2. ОСНОВНЫЕ ПОНЯТИЯ

Задание — основная независимая единица работы ЭВМ, задаваемая пользователем. Каждое задание описывается с помощью управляющих операторов системно-независимого языка BATCH. Задания могут быть одиночными или сгруппированы в один пакет, который образует входной поток заданий. Задание может содержать команды КЯС и системных программ. Режим выполнения программой BATCH таких команд называется системным режимом работы BATCH. Пакет заданий создается пользователем с помощью редактора текста.

Процесс пакетной обработки заключается в следующем: BATCH воспринимает входной поток заданий, транслирует его и организует в виде файла на магнитном носителе. Результат трансляции представляет собой управляющий файл BATCH. Затем драйвер BATCH (BA.SYS) обрабатывает этот файл и, таким образом, выполняет пакет заданий пользователя.

Регистрационный файл — файл, который формируется программой BATCH в результате выполнения пакета заданий и включает в себя команды из пакета заданий и сообщения, предназначенные для вывода на терминал.

3. СИСТЕМНО-НЕЗАВИСИМЫЙ ЯЗЫК BATCH

Системно-независимый язык BATCH предназначен для составления заданий пользователя.

Задание на языке BATCH состоит из данных и управляющих операторов, записанных с помощью символов кода КОИ-7 (коды 40—137). Назначение основных символов приведено в табл. 1.

Таблица 1

Символ	Назначение
1	2
Пробел	Разделитель полей управляющего оператора
!	Ограничитель комментария
»	Преобразователь символа. Используется также в строке текста, содержащей символы разделителей
¤	Признак управляющего оператора BATCH
.	Разделитель для типа файла
—	Символ продолжения строки, если после него следует один из следующих символов: комбинация <ВК> и <ПС>; любое число пробелов и табуляций, за которыми следует <ВК> и <ПС>; ограничитель комментария; пробелы за которыми следует ограничитель комментария. Если за дефисом (—) следует другой символ, то дефис воспринимается как минус, указывающий отрицательную величину
/	Признак переключателя
0 — 9	Числовые компоненты строки
:	Ограничитель имени устройства; может быть использован для разделения имени переключателя и его аргумента
A — Z	Буквенные компоненты строки
=	Разделитель имени переключателя и его аргумента
\	Признак директивы оператора
+	Разделитель файлов, используемых в одной и той же операции. Используется также для обозначения положительной величины
,	Разделитель аргументов в команде
*	Конструкция «звездочка» для спецификации файлов
<ВК>, <ПС>	Символы возврата каретки и перевода строки. Указывают конец строки или конец записи

В системном режиме BATCH интерпретирует символы из табл. 1, иначе (см. п. 3.4).

3.1. Формат управляющих операторов

Управляющий оператор может занимать от одного до трех полей — поле команды, поле спецификации и поле комментария, которые распознаются по порядку их следования и ограничительным символам. Поле спецификации может в свою очередь состоять из нескольких полей, что определяется числом спецификаций файлов в управляющем операторе.

Синтаксис управляющего оператора:

¤команда/прк спф/прк [!ком]

где /прк — переключатель;

/спф — спецификация файла;

ком — комментарий.

Количество символов в управляющем операторе должно быть не более 80, не считая пробелов, табуляции и комментария. Допускается продолжение оператора на следующую строку с помощью дефиса (-).

3.1.1. Поле команды

Поле команды в управляющем операторе BATCH содержит операцию, которая должна быть выполнена. Оно состоит из имени команды и переключателей поля команды. Имя команды в первой позиции имеет символ ¤. Пробелы в имени команды не допускаются. Существует две формы имени команды: полная и сокращенная, состоящая из ¤ и первых трех символов имени команды. Функции переключателей распространяются на весь управляющий оператор. Переключатель должен непосредственно следовать за именем команды и начинаться с наклонной черты (/).

ПРИМЕЧАНИЕ. Для всех переключателей поля команды существуют противоположные по своим функциональным действиям переключатели, которые записываются с префиксом /NO (/DELETE — /NODELETE) и являются переключателями по умолчанию, кроме /NOWAIT для команд ¤MOUNT и ¤DISMOUNT, и /NOOBJECT для команды ¤LINK.

Поле команды ограничивается пробелом, табуляцией или возвратом каретки.

Команды BATCH приведены в табл. 2.

Команда	Функция
1	2
¤SEQ[UENCE]	Присваивает заданию идентифицирующий номер
¤JOB	Указывает начало задания
¤EOJ	Указывает конец задания
¤MOU[NT]	Сообщает оператору о необходимости установки определенного тома на устройство
¤DIS[MOUNT]	Сообщает оператору о необходимости снятия тома с устройства
¤FOR[TRAN]	Транслирует исходную программу на языке ФОРТРАН
¤BAS[IC]	Транслирует исходную программу на языке БЕЙСИК
¤MAC[RO]	Транслирует исходную программу на языке АССЕМБЛЕР
¤LIB[RARY]	Указывает библиотеки, которые BATCH должна использовать в операции редактирования
¤LIN[K]	Редактирует объектные модули
¤RUN	Вызывает выполнение программы

1	2
¤CAL[L]	Передает управление другому управляющему файлу BATCH, выполняет его и возвращает управление вызывающему заданию
¤CHA[IN]	Передает управление другому управляющему файлу BATCH
¤DAT[A]	Указывает начало данных
¤EOD	Указывает конец данных
¤MES[SAGE]	Выдает сообщение оператору
¤COPY	Копирует файлы
¤CRE[ATE]	Создает файлы из данных, включенных в задание
¤DEL[ETE]	Удаляет файлы
¤DIR[ECTORY]	Обеспечивает справочником указанного тома
¤PRI[NT]	Печатает файлы
¤FODS	Указывает, что следующие строки являются командами системного режима

В табл. 3 приведены переключатели поля команды.

Таблица 3

Переключатель	Функция
1	2
/BAN[NER]	Включает в регистрационный файл заголовок задания. Используется только с командой ¤JOB
/CRE[F]	Создает во время трансляции вспомогательный файл таблицы перекрестных ссылок. Используется только с командой ¤MACRO
/DEL[ETE]	Удаляет входные файлы после выполнения операции. Используется с командами ¤COPY и ¤PRINT
/DOL[LARS]	Указывает, что данные, непосредственно следующие за командой, могут иметь символ ¤ в первой позиции строки. Используется с командами ¤CREATE, ¤DATA, ¤FORTRAN и ¤MACRO. BATCH прекращает чтение данных из входного потока, если встретит физический конец файла или одну из следующих команд: ¤JOB, ¤SEQUENCE, ¤EOD, ¤EOJ
/LIB[RARY]	Включает в операцию редактирования библиотеку по умолчанию (SYSLIB.OBJ) и все библиотечные файлы, определенные в задании по команде ¤LIBRARY. Используется с командами ¤LINK и ¤MACRO
/LIS[T]	Создает вспомогательный файл листинга на устройстве LST: или записывает данные из задания на регистрационное устройство LOG:. Использу-

1	2
	ется с командами ¤BASIC, ¤CREATE, ¤ DATA, ¤JOB, ¤FORTRAN и ¤ MACRO
/MAP	Создает вспомогательный файл карты загрузки на устройстве LST:. Используется с командами ¤FORTRAN, ¤LINK и ¤ MACRO
/OBJ[ECT]	Создает вспомогательный объектный файл. Используется с командами ¤FORTRAN, ¤LINK и ¤MACRO. При использовании с командой ¤LINK включает вспомогательные объектные файлы в операцию редактирования. Является в этом случае переключателем по умолчанию
/FODS	Устанавливает системный режим работы BATCH. Используется только с командой ¤JOB
/RUN	Редактирует и/или вызывает программу. Используется с командами ¤BASIC, ¤FORTRAN, ¤LINK и ¤MACRO
/TIM[E]	Записывает величину времени суток в регистрационный файл. Используется только с командой ¤JOB
/UNI[QUE]	Осуществляет орфографическую проверку на единственность переключателей и команд. Использование этого переключателя допускает сокращение имен команд и переключателей до символов, сохраняющих единственность трактовки их имен. Используется только с командой ¤JOB
/WAI[T]	Задерживает выполнение задания и ждет действия оператора. Используется командами ¤DISMOUNT, ¤MOUNT и ¤MESSAGE. Является переключателем по умолчанию для команд ¤MOUNT и ¤DISMOUNT. Печатает вопросительный знак и ждет ответа. Ответ может содержать директивы оператора
/WRI[TE]	Сообщает оператору о необходимости разрешения записи на том. Используется только с командой ¤MOUNT

3.1.2. Поле спецификации

Поле спецификации следует непосредственно за полем команды. Оно используется для включения в команду имен устройств и файлов. Поля спецификации отделяются от поля команды и друг от друга пробелами. Если поле спецификации содержит более одного файла, используемого в одной и той же операции, соответствующие этим файлам поля отделяются символом «плюс» (+).

Пример:

¤MACRO F1+F2/SOURCE F3/OBJECT

При необходимости повторения команды для различных файлов, поля спецификации отделяются запятой.

Пример:

¤MACRO F1/SOURCE F2/OBJECT F3/MAP,F4 + F5/SOURCE F6/OBJECT

В этом примере АССЕМБЛЕР транслирует файл F1 и создает для него объектный файл F2 и файл карты загрузки F3, затем транслирует файлы F4 и F5 и создает для них объектный файл F6.

Функции переключателей поля спецификации распространяются только на то поле, за которым они непосредственно следуют.

В зависимости от используемой команды поле спецификации может содержать спецификации файлов, переключатели поля спецификации (табл. 4) или произвольную строку символов кода КОИ-7.

Переключатели поля спецификации должны следовать непосредственно за спецификацией файлов; они указывают как используется файл.

Таблица 4

Переключатель	Функция
1	2
/BAS[IC]	Указывает исходный файл на языке БЕЙСИК
/EXE[CUTABLE]	Указывает на создание файла формата отображения памяти, как результата операции редактирования
/FOR[TRAN]	Указывает исходный файл на языке ФОРТРАН
/INP[UT]	Указывает входной файл; является переключателем по умолчанию
/LIB[RARY]	Указывает библиотечный файл для включения его в операцию редактирования
/LIS[T]	Указывает файл листинга
/LOG[ICAL]	Указывает логическое имя устройства
/MAC[RO]	Указывает исходный файл на языке АССЕМБЛЕР
/MAP	Указывает файл карты загрузки
/OBJ[ECT]	Указывает объектный файл
/OUT[PUT]	Указывает выходной файл
/PHY[SICAL]	Указывает физическое имя устройства
/SOU[RCE]	Указывает исходный файл
/VID	Указывает идентификатор тома

3.1.2.1. Имена устройств

Каждое устройство в поле спецификации обозначается стандартным именем устройства (два или три символа). Если не указан номер привода для устройства, BATCH предполагает привод 0.

Допускается присвоение устройствам логических имен.

Перед работой с BATCH пользователю необходимо присвоить устройствам логические имена LOG: и LST:, которые используются BATCH в качестве устройств по умолчанию для вывода, соответственно, регистрационных файлов и файлов листингов (регистрационным устройством LOG: может быть любое устройство, кроме MT:).

3.1.2.2. Спецификация файлов

Формат спецификации файлов в управляющих операторах BATCH соответствует общему формату спецификаций файлов операционной системы ФОДОС-2. Если тип файла не указан, BATCH присваивает ему стандартное значение типа файла (табл. 5).

Таблица 5

Тип файла	Характеристика файла
1	2
.BAS	Исходный файл на языке БЕЙСИК
.BAT	Файл команд BATCH
.CTL	Управляющий файл BATCH
.CTT	Вспомогательный файл BATCH
.DAT	Файл данных на языке БЕЙСИК или ФОРТРАН
.DIR	Файл листинга справочника
.FOR	Исходный файл на языке ФОРТРАН
.LST	Файл листинга
.LOG	Регистрационный файл BATCH
.MAC	Исходный файл на языке АССЕМБЛЕР
.MAP	Файл карты загрузки
.OBJ	Объектный файл
.SOU	Вспомогательный исходный файл
.SAV	Файл формата отображения памяти

В некоторых управляющих операторах BATCH (¤COPY, $CREATE, $DELETE, $DIRECTORY и $PRINT) допускается использование конструкции звездочка (*) для обозначения имени или типа файла.

ПРИМЕЧАНИЕ. В управляющих операторах BATCH нельзя использовать встроенные конструкции (* или %). Однако, они допустимы в спецификациях файлов для системного режима работы BATCH.

3.1.3. Поле комментария

Поле комментария распознается по ограничительному символу (!), который может занимать любую позицию в управляющем операторе, кроме первой. BATCH воспринимает все

символы, следующие за восклицательным знаком и предшествующие возврату каретки, как комментарий.

3.1.4. Вспомогательные файлы

Если управляющий оператор не содержит поле спецификации, BATCH создает, так называемые, вспомогательные файлы, необходимые для выполнения требуемой команды.

Пример:

```
$MACRO/RUN/OBJECT/LIST
Исходная программа $EOD
```

В этом примере BATCH создает следующие вспомогательные файлы: исходный, объектный, файл листинга программы и файл формата отображения памяти.

BATCH выводит вспомогательные файлы на устройство по умолчанию DK: или на устройство листинга LST: (устройство, предназначенное для вывода листингов программ). Если устройство вывода имеет файловую структуру, BATCH присваивает вспомогательным файлам следующие имена:

HHHMMM.LST — файл листинга (выводится на LST:)
HHHMMM.MAP — файл карты загрузки (выводится на LST:)
HHHППП.OBJ — объектный файл (выводится на DK:)
HHHППП.SOU — исходный файл (выводится на DK:)
Вспомогательному файлу формата отображения памяти (выводится на DK:) BATCH присваивает имя 000000.SAV;

где HHH — последние три цифры номера задания, присвоенного по команде $SEQUENCE. Если команда $SEQUENCE не используется, BATCH приписывает HHH значение 000;

MMM — номер файла листинга (или карты загрузки), созданного BATCH с тех пор, как был загружен драйвер BA.SYS. Первый такой файл имеет номер 000;

ППП — номер объектного, формата отображения памяти или исходного файла, созданного BATCH во время выполнения текущей команды. Первый такой файл имеет номер 000.

BATCH устанавливает ППП в 000 каждый раз после выполнения команд $LINK, $MACRO, которые используют вспомогательные файлы.

3.2. Общие правила составления задания

Для работы с BATCH пользователь должен придерживаться следующих правил составления задания:

1) символ $ занимает первую позицию в строке управляющего оператора;

2) каждое задание начинается командой $JOB и ограничивается командой ¤EOJ;

3) имя команды и переключателя определяется полностью

или по первым трем символам для команды и соответствующим символам для переключателя;

4) конструкция звездочка (*) определяется только в командах ¤COPY, ¤CREATE, ¤DELETE, ¤DIRECTORY и ¤PRINT или в тех командах системного режима работы BATCH, которые обычно допускают ее использование;

5) максимальная длина управляющего оператора — 80 символов, исключая пробелы, табуляцию и комментарий;

6) для вызова BATCH нельзя использовать косвенный файл.

3.3. Команды BATCH

Все команды BATCH должны начинаться символом ¤, который должен занимать первую позицию в строке.

3.3.1. Команда ¤SEQUENCE. Команда ¤SEQUENCE является необязательной. Если пользователь определяет ее, она должна непосредственно предшествовать команде ¤JOB. Команда ¤SEQUENCE присваивает заданию идентифицирующий номер. Если номер, указанный пользователем, представляет собой число, состоящее из менее чем трех символов, BATCH дополняет его нулями слева.

Синтаксис команды:

¤SEQUENCE ID [!ком]

где ID — номер задания (десятичное число без знака).

Пример:

```
¤SEQUENCE 3 !номер задания 003
¤JOB
```

3.3.2. Команда ¤JOB. Команда ¤JOB указывает начало задания. Каждое задание должно иметь свою команду ¤JOB и заканчиваться командой ¤EOJ.

Синтаксис команды:

¤JOB [/прк...] [!ком]

где /прк — один из следующих переключателей: /BANNER, /NOBANNER, /LIST, /NOLIST, /FODS, /NOFODS, /TIME, /NOTIME, /UNIQUE и /NOUNIQUE.

Пример:

```
¤JOB/TIME/UNIQUE
```

По этой команде BATCH допускает сокращение имен команд и переключателей в задании и записывает величину времени суток в регистрационный файл.

3.3.3. Команда ¤EOJ. Команда ¤EOJ указывает конец задания. Для каждого задания она должна быть последней командой.

Синтаксис команды:

¤EOJ [!ком]

3.3.4. Команда ¤MOUNT. Команда ¤MOUNT присваивает устройству логическое имя и сообщает оператору о необходимости установки на устройство определенного тома (когда BATCH встретит ¤MOUNT при выполнении задания, она напечатает эту команду на системном терминале).

Синтаксис команды:

¤MOUNT [/прк...] фим:[/PHYSICAL] [/VID=X] — [лим:/LOGICAL] [!ком]

где /прк — один из следующих переключателей: /WAIT, /NOWAIT, /WRITE и /NOWRITE;

фим: — физическое имя и номер привода устройства. Если номер привода не указан, оператор может ввести его в ответ на воспросительный знак, который печатает BATCH;

/PHYSICAL — переключатель, указывающий физическое имя устройства. Если для спецификации устройства не определен ни /PHYSICAL, ни /LOGICAL, BATCH предполагает переключатель /PHYSICAL;

/VID=X — указывает идентификатор тома (X). Идентификатор тома представляет собой имя, присвоенное тому для облегчения его поиска оператором и наносится на этикетку тома. Если идентификатор содержит пробелы, его необходимо заключить в кавычки («X»);

лим: — логическое имя устройства;

/LOGICAL — переключатель, указывающий логическое имя.

Пример:

```
¤MOUNT/WAIT/WRITE RK:/VID=TIMM 3:/LOGICAL
?
```

Эта команда сообщает оператору о необходимости выбора привода для диска RK: и установки на него тома TIMM. Оператор выбирает привод, устанавливает том с именем TIMM, разрешает на него запись и в ответ на вопрос печатает номер выбранного привода. BATCH присваивает диску логическое имя 3.

3.3.5. Команда ¤DISMOUNT. Команда ¤DISMOUNT отменяет логическое имя устройства, присвоенное командой ¤MOUNT и сообщает оператору о необходимости разгрузки определенного устройства (когда BATCH встретит ¤DISMOUNT при выполнении задания, она напечатает эту команду на системном терминале).

Синтаксис команды:

¤DISMOUNT [/прк...] лим:[/LOGICAL] [!ком]

где /прк — один из следующих переключателей: /WAIT и /NOWAIT;

лим: — логическое имя устройства, которое необходимо отменить;
/LOGICAL — переключатель, указывающий логическое имя устройства.

Пример:

¤DISMOUNT/WAIT OUT:/LOGICAL
RK2:?

Эта команда сообщает оператору о необходимости разгрузки устройства OUT: и отменяет логическое имя OUT: (OUT: было присвоено RK2:). Оператор разгружает RK2: и печатает возврат каретки.

3.3.6. Команда ¤BASIC. Команда ¤BASIC вызывает трансляцию исходной программы на языке БЕЙСИК.

Синтаксис команды:

¤BASIC [/прк...] [испф/прк1] [!ком]

где /прк — один из следующих переключателей: /RUN, /NORUN, /LIST и /NOLIST;

испф — спецификация исходного файла. Если она не указана, непосредственно за командой должен следовать текст исходной программы. Переключатели (/прк1) для спецификации исходного файла могут быть следующие: /BASIC, /SOURCE и /INPUT (табл. 4). Исходная программа после команды ¤BASIC ограничивается командой ¤EOD или любой командой BATCH.

3.3.7. Команда ¤FORTRAN. Команда ¤FORTRAN вызывает трансляцию исходной программы на языке ФОРТРАН.

Синтаксис команды:

¤FORTRAN [/прк...] [испф[/прк1]] [спф/OBJECT]—
[спф/LIST] [спф/EXECUTE]—
[спф/MAP] [спф/LIBRARY] [!ком]

где /прк — один из следующих переключателей: /RUN, /NORUN, /OBJECT, /NOOBJECT, /LIST, /NOLIST, /MAP, /NOMAP, /DOLLARS и /NODOLLARS;

испф — спецификация исходного файла. Если она не указана, непосредственно за командой должен следовать текст исходной программы. Переключатели (прк1) для спецификации исходного файла могут быть следующие: /FORTRAN, /SOURCE и /INPUT (табл. 4). Исходная программа после команды ¤FORTRAN ограничивается командой ¤EOD или любой командой BATCH. Если символ ¤ находится в первой позиции исходной программы, то команда ¤FORTRAN/DOLLARS не используется, текст исходной программы должен следовать за командой ¤CREATE/DOLLARS;

спф/OBJECT — спецификация для объектного файла, созда-

ваемого транслятором. Если она не указана, но с командой ¤FORTRAN определен переключатель /OBJECT, то BATCH создает вспомогательный объектный файл, который удаляет после операции редактирования;

спф/LIST — спецификация для файла листинга, создаваемого транслятором. /LIST указывает файл листинга;

спф/EXECUTE — спецификация для файла формата отображения памяти. Если она не указана, BATCH создает вспомогательный файл формата отображения памяти, который потом стирает. Если эта спецификация включена в команду, за ней должен следовать переключатель /EXECUTE;

спф/MAP — спецификация для файла карты загрузки. Если она включена в команду, за ней должен следовать переключатель /MAP;

спф/LIBRARY — спецификация файла библиотеки. Переключатель /LIBRARY указывает BATCH на включение этого файла в операцию редактирования.

Пример:

¤FORTRAN/LIST/OBJECT PROG.FOR

По этой команде BATCH транслирует программу PROG.FOR и создает вспомогательный файл листинга и вспомогательный объектный файл.

3.3.8. **Команда ¤MACRO.** Команда ¤MACRO вызывает трансляцию исходной программы на языке АССЕМБЛЕР.

Синтаксис команды:

¤MACRO [/прк...] [испф[/прк1]] [спф/OBJ] [спф/LIST]— [спф/MAP] [спф/LIBRARY] [спф/EXECUTE] [!ком]

где /прк — один из следующих переключателей: /RUN, /NORUN, /OBJECT, /NOOBJECT, /LIST, /NOLIST, /CREF, /NOCREF, /MAP, /NOMAP, /DOLLARS, /NODOLLARS, /LIBRARY, /NOLIBRARY;

испф — спецификация исходного файла. Если она не указана, непосредственно за командой должен следовать текст исходной программы. Переключатели (/прк1) для спецификации исходного файла могут быть следующие: /MACRO, /SOURCE и /INPUT (табл. 4). Исходная программа после команды ¤MACRO ограничивается командой ¤EOD или любой командой BATCH. Если символ ¤ находится в первой позиции исходной программы, то команда ¤MACRO/DOLLARS не используется, текст исходной программы должен следовать за командой ¤CREATE/DOLLARS;

спф/OBJECT — спецификация для объектного файла, создаваемого транслятором. Если она не указана, но с командой ¤MACRO определен переключатель /OBJECT, BATCH соз-

дает вспомогательный объектный файл, который удаляет после операции редактирования. Если он включена в команду, за ней должен следовать переключатель /OBJECT;
спф/LIST — спецификация для файла листинга, создаваемого транслятором. /LIST указывает файл листинга;
спф/MAP — спецификация для файла карты загрузки. Если она включена в команду, за ней должен следовать переключатель /MAP;
спф/LIBRARY — спецификация файла библиотеки. Переключатель /LIBRARY указывает BATCH на включение этого файла в операцию редактирования;
спф/EXECUTE — спецификация для файла формата отображения памяти. Если она не указана, но с командой ¤MACRO определен переключатель /RUN, BATCH создает вспомогательный файл формата отображения памяти. Если эта спецификация включена в команду, за ней должен следовать переключатель /EXECUTE.

Пример:

¤MACRO/LIST/OBJECT PROG.MAC

По этой команде BATCH транслирует программу PROG.MAC и создает вспомогательный файл листинга и вспомогательный объектный файл.

3.3.9. Команда ¤LIBRARY

Команда ¤LIBRARY позволяет указать библиотечные файлы для включения в операцию редактирования по команде с переключателем /LIBRARY. Первоначально библиотекой по умолчанию является SISLIB.OBJ.

Синтаксис команды:

¤LIBRARY спф [!ком]

или

¤LIBRARY спф+SYSLIB [!ком]

где спф — библиотечный файл, тип файла по умолчанию .OBJ; SISLIB — системная библиотека ФОДОС-2.

Пример:

¤LIBRARY LIB1.OBJ+LIB2.OBJ+SISLIB.OBJ

По этой команде BATCH включает в операцию редактирования (кроме SISLIB.OBJ) две библиотеки LIB1.OBJ и LIB2.OBJ.

3.3.10. Команда ¤LINK. Команда ¤LINK создает из объектных файлов файлы формата отображения памяти. ¤LINK редактирует файлы, указанные в команде, со всеми вспомогательными объектными файлами, созданными BATCH в результате выполнения задания или после выполнения предыдущей (если таковая была) операции редактирования. Такие

вспомогательные объектные файлы являются результатом выполнения команды ¤MACRO/OBJ, в которой не была указана спецификация для объектных файлов.
Синтаксис команды:
¤LINK[/прк] [спф/OBJECT] [спф/LIBRARY] [спф/MAP]—[спф/EXECUTE] [ком!]
где /прк — один из следующих переключателей: /LIBRARY, /NOLIBRARY, /MAP, /NOMAP, /OBJECT, /NOOBJECT, /RUN, /NORUN;
спф/OBJECT — спецификация объектного файла. Переключатель /OBJECT является переключателем по умолчанию;
спф/LIBRARY — спецификация библиотечного файла. Переключатель /LIBRARY указывает BATCH на включение этого файла в операцию редактирования;
спф/MAP — спецификация для файла карты загрузки, создаваемого по команде ¤LINK. Если она включена в команду, за ней должен следовать переключатель /MAP;
спф/EXECUTE — спецификация для файла формата отображения памяти. Если эта спецификация включена в команду, за ней должен следовать переключатель /EXECUTE.

Пример:
¤LINK RUN
По этой команде BATCH редактирует все вспомогательные объектные файлы, созданные в результате выполнения задания или после выполнения предыдущей команды ¤LINK и запускает готовую программу.

3.3.11. Команда ¤RUN. Команда ¤RUN вызывает выполнение программы (файла формата отображения памяти).
Синтаксис команды:
¤RUN спф [!ком]
где спф — спецификация файла, который необходимо выполнить.

Пример:
¤RUN DIR
¤DATA
LP:=DK:/L
¤EOD
В этом примере BATCH вызывает программу DIR и передает ей командную строку.

3.3.12. Команда ¤CALL. Команда ¤CALL передает управление другому управляющему файлу BATCH, временно задерживая выполнение текущего управляющего файла. BATCH выполняет новое задание до тех пор, пока не встретит ¤EOJ или пока не произойдет удаление задания (новое

задание также может содержать команду ¤CALL; глубина таких вложений может равняться 31). Затем BATCH возвращает управление вызывающему заданию, команде, непосредственно следующей за командой ¤CALL. Регистрационный файл вызываемого задания BATCH включает в регистрационный файл вызывающего задания.

Синтаксис команды:

¤CALL спф [!ком]

где спф — спецификация файла, которому необходимо передать управление.

BATCH не допускает использование переключателей с командой ¤CALL. Если в спецификации файла указан тип .CTL, BATCH предполагает управляющий файл. Если тип файла не указан, BATCH предполагает файл команд (.BAT) и транслирует его перед выполнением вызываемого задания.

3.3.13. Команда ¤CHAIN. Команда ¤CHAIN передает управление другому управляющему файлу, но не возвращает его вызывающему заданию.

Синтаксис команды:

¤CHAIN спф [!ком]

BATCH не допускает использование переключателей с командой ¤CHAIN. Если в спецификации файла указан формат .CTL, BATCH предполагает управляющий файл. Если тип файла не указан, BATCH предполагает файл команд (.BAT) и транслирует его перед выполнением вызываемого задания.

В задании BATCH за командой ¤CHAIN всегда должна следовать команда ¤EOJ.

3.3.14. Команда ¤DATA. Команда ¤DATA используется для включения данных в задание BATCH. Данные, включенные в задание таким образом, не требуют обозначения (имени файла). BATCH передает эти данные соответствующей программе, имитируя ввод системного терминала.

Синтаксис команды:

¤DATA [/прк...] [!ком]

где /прк — один из следующих переключателей: /DOLLARS, /NODOLLARS, /LIST, /NOLIST.

Данные после команды ¤DATA ограничиваются любой командой BATCH. Однако, если используется ¤DATA/DOLLARS, они должны быть ограничены командой ¤EOD.

Пример:

```
¤RUN PIP
¤DATA
LP:=COPY.MAC/A
¤EOD
```

Если команда ¤DATA используется с программами на языке ФОРТРАН, необходимо вставить CTRL/Z после последней строки данных перед ¤EOD (или перед следующей командой BATCH, если ¤EOD не используется).

Пример:

```
¤FORTRAN/RUN A.FOR
¤DATA
1
2
3
^Z <RET> <LF>
¤EOD
¤RUN PIP
```

3.3.15. Команда ¤EOD. Команда ¤EOD указывает в задании BATCH конец данных или конец текста исходной программы.

Синтаксис команды:

¤EOD [!ком]

Команда ¤EOD означает конец данных, ассоциированных с любой из следующих команд:

¤BASIC
¤CREATE
¤DATA
¤FORTRAN
¤MACRO

3.3.16. Команда ¤MESSAGE. Команда ¤MESSAGE выдает на системный терминал сообщение оператору. Она обеспечивает взаимосвязь задания и оператора.

Синтаксис команды:

¤MESSAGE [/прк] соб [!ком]

где /прк — один из следующих переключателей: /WAIT, /NOWAIT;

соб — сообщение (строка символов кода КОИ-7, которая должна соответствовать одной строке терминала). BATCH печатает сообщение на системном терминале.

Пример:

```
¤MESSAGE/WAIT установите запасной том на MT0:
```

По этой команде BATCH печатает на системном терминале сообщение:

```
установите запасной том на MT0:
?
```

Оператор устанавливает том магнитной ленты и печатает возврат каретки.

3.3.17. Команда ¤COPY. Команда ¤COPY копирует файлы с одного тома (устройства) на другой в режиме отображения памяти («слово в слово»). В спецификациях входных и выходных файлов могут быть использованы конструкции * и %. Допускается объединение нескольких входных файлов в один выходной; в этом случае в спецификации выходного файла нельзя использовать эти конструкции.

Синтаксис команды:

¤COPY [/прк] выходспф [..., выходспф]/OUTPUT — входспф [..., входспф] [/INPUT] [!ком]

где /прк — один из следующих переключателей: /DELETE, /NODELETE;

выходспф — спецификация для выходного файла; должен быть указан тип файла;

/OUTPUT — переключатель, указывающий спецификацию для выходного файла;

входспф — спецификация входного файла;

/INPUT — переключатель, указывающий входной файл. Является переключателем по умолчанию.

Пример:

¤COPY *.*/OUT MX1:*.OBJ,PC:/OUT MX1:*.MAC

По этой команде BATCH копирует все файлы с типом .OBJ с MX1: на DK: и все файлы с типом .MAC с MX1: на PC:.

3.3.18. Команда ¤CREATE. Команда ¤CREATE создает файл из данных, включенных в задание непосредственно за командой ¤CREATE. Данные после ¤CREATE ограничиваются любой командой BATCH. Однако, если используется ¤CREATE/DOLLARS, они должны быть ограничены командой ¤EOD.

Синтаксис команды:

¤CREATE [/прк] спф [!ком]

где /прк — один из следующих переключателей: /DOLLARS, /NODOLLARS, /LIST, /NOLIST;

спф — спецификация для создаваемого файла.

Пример:

```
¤CREATE/LIST BAB.MAC
    Исходная программа на языке АССЕМБЛЕР
¤EOD
```

В этом примере BATCH создает из данных, непосредственно следующих за командой ¤CREATE, новый файл (BAB.MAC) на устройстве DK: и записывает все данные в регистрационный файл.

3.3.19. Команда ¤DELETE. Команда ¤DELETE удаляет файлы с указанного тома.

Синтаксис команды:
¤DELETE спф [..., спф] [!ком]
где спф — спецификация файла, который необходимо стереть.

Пример:

¤DELETE TEST.*

По этой команде BATCH удаляет на DK: все файлы с именем TEST (независимо от типов файлов).

3.3.20. Команда ¤DIRECTORY. Команда ¤DIRECTORY позволяет получить справочник указанного тома. Если в команде спецификация для файла листинга справочника не указана, BATCH записывает справочник в регистрационный файл.

Синтаксис команды:
¤DIRECTORY [спф/LIST] [спф[..., спф]] [/INPUT] [!ком]
где спф/LIST — спецификация и переключатель, указывающие файл листинга справочника тома;
спф/INPUT — спецификация и переключатель, указывающие файлы, о которых требуется получить справочную информацию.

Пример:

¤DIRECTORY

По этой команде BATCH записывает справочник DK: в регистрационный файл.

3.3.21. Команда ¤PRINT. Команда ¤PRINT распечатывает содержимое указанных файлов на устройстве LST:.

Синтаксис команды:
¤PRINT [/прк] спф[..., спф] [/INPUT] [!ком]
где /прк — один из следующих переключателей: /DELETE, /NODELETE;
спф — спецификация файла, который требуется распечатать;
/INPUT — переключатель, указывающий входной файл.

3.3.22. Команда $FODS. Команда $FODS устанавливает системный режим работы BATCH. BATCH интерпретирует информацию из задания до появления символа $ в начале строки как команды монитору ФОДОС-2 и системным программам (см. п. 3.4).

Синтаксис команды
$FODS [!ком]

3.4. Системный режим работы BATCH

Системный режим работы BATCH может быть установлен по команде $FODS или $JOB/FODS. BATCH будет находиться в этом режиме до тех пор, пока:

1) не обнаружит символ $ в первой позиции команды (если режим установлен по команде $FODS);

2) не обнаружит команду $EOJ (если режим установлен по команде $JOB/FODS).

Символы (.), (*), ($), табуляция или пробел являются управляющими, если они располагаются в первой позиции строки. Точка (.) указывает команду монитора.

Пример:

```
.R PIP
```

Звездочка (*) указывает строку данных (строку, не адресованную монитору или драйверу BATCH). В строке данных не должно быть комментария.

Пример:

```
*FILE.DAT/D
```

Эта строка является командной строкой программы обмена (PIP).

Символ $ указывает команду BATCH. Табуляция <ТАБ> или пробел указывают строку, адресованную драйверу BATCH.

При работе с BATCH в системном режиме необходимо учитывать особенности системных программ. Например, для инициализации диска по программе обслуживания устройств (DUP) нужно предусмотреть в задании ответ на вопрос (если он будет поставлен) ARE YOU SURE?

Пример:

```
$FODS
.R DUP
*MX1:/Z
*Y
```

Системный режим работы BATCH позволяет создавать, так называемые, BATCH-программы. Такие программы состоят из стандартных команд системного режима (команд монитора, командных строк системных программ и т. п.) и специальных команд. Специальные команды системного режима дают возможность динамически управлять выполнением задания.

3.4.1. Средства создания BATCH-программ

К средствам создания BATCH-программ относятся:

1) метки;
2) переменные;
3) команды регистрации ввода-вывода с терминала;
4) специальные управляющие команды;
5) комментарии.

3.4.1.1. Метки

Операторы и команды, входящие в задание, можно помечать с помощью меток. Метка должна начинаться с первой

позиции в строке и ограничиваться двоеточием (:) и возвратом каретки.

3.4.1.2. Переменные

Переменная представляет собой величину, значение которой заменяется во время выполнения программы. В BATCH-программе допускается определение 26 переменных (символы от A до Z).

Значение переменных присваивается по оператору LET.
Синтаксис оператора:
<ТАБ>LET X="C
или
<ТАБ>LET X=N
где X — переменная;
"C — символ КОИ-7, код которого присваивается переменной;
N — восьмеричное число (0—377); от 0 до 177 — положительное; от 200 до 377 — отрицательное.

Пример:
<ТАБ> LET A="0
В этом примере переменной A присваивается значение 60.

Увеличение значения переменной на 1 осуществляется путем определения перед ней символа %.

Синтаксис команды:
<ТАБ>%A

Условная передача управления в BATCH-программе осуществляется по оператору IF.
Синтаксис оператора:
<ТАБ>IF(X—"C) метка 1, метка 2, метка 3
или
<ТАБ>IF (X—N) метка 1, метка 2, метка 3
где X — переменная;
"C — символ КОИ-7, код которого сравнивается со значением переменной;
N — восьмеричное число (0—377);
метка 1
метка 2 — метки в BATCH-программе, одной из которых
метка 3 передается управление.

В зависимости от значения выражения (X—"C) или (X—N), драйвер BATCH передает управление:
на метку 1, если значение $<$ 0;
на метку 2, если значение $=$ 0;
на метку 3, если значение $>$ 0.

В операторе IF перед метками допускается определение символов плюс «+» и минус «—». Если перед меткой опре-

делен символ «—», BATCH осуществляет поиск этой метки от начала задания. Если перед меткой не определен символ или определен символ «+», BATCH осуществляет поиск этой метки после оператора IF.

Безусловная передача управления в BATCH-программе осуществляется по оператору GOTO.

Синтаксис оператора:

<ТАБ>GOTO метка

где метка — метка в BATCH-программе, на которую передается управление.

В операторе GOTO перед метками также допускается определение символов «+» и «—» (аналогично оператору IF).

ПРИМЕЧАНИЕ. Если BATCH не обнаружит метку, указанную в операторе, она прекратит выполнение программы.

3.4.1.3. Команды регистрации ввода-вывода с терминала.

Существует ряд команд (табл. 6), которые используются для управления вводом-выводом с/на терминал. Если в программе не указана ни одна из них, BATCH предполагает TTYOUT.

Таблица 6

Команда	Функция
1	2
<ТАБ>NOTTY	Не записывает вводимую и выводимую информацию с терминала в регистрационный файл (комментарии записываются)
<ТАБ>TTYIN	Записывает вводимую с терминала информацию в регистрационный файл
<ТАБ>TTYIO	Записывает вводимую и выводимую информацию с терминала в регистрационный файл
<ТАБ>TTYOUT	Записывает выводимую на терминал информацию в регистрационный файл

3.4.1.4. Специальные управляющие команды

В командах системного режима, начинающихся с точки (.) и звездочки (*), используются специальные управляющие команды драйверу BATCH.

Синтаксис команды:

/текст/

где текст — одно из значений, перечисленных в табл. 7.

Пример:

¤JOB/FODS
.R MACRO

Таблица 7

Значение	Функции
1	2
CTY	Осуществляет ввод с системного терминала; печатает вопрос (?) и ждет ответа
FF	Выводит содержимое регистрационного буфера
NL	Вставляет новую строку
X	Вставляет в командную строку символ КОИ-7, код которого является значением переменной X
«СООБЩЕНИЕ»	Печатает сообщение на системном терминале

```
* '«введите командную строку MACRO» ''CTY'
```

Оператор получит на терминале сообщение, напечатает ответ и <ВК>.

```
Введите командную строку MACRO
?FILE,FILE=FILE
```

Пример:

```
.ASSIGN ' «напечатайте имя устройства LST» ' 'CTY' LST
```

Оператор получит на терминале сообщение, напечатает ответ и <ВК>.

```
Напечатайте имя устройства LST
?MX1:
```

3.4.1.5. Комментарии

Комментарии в BATCH-программе могут быть записаны в виде отдельного оператора.

Пример:

```
<ТАБ>  !задание требует ввода с терминала
```

3.4.2. Пример BATCH-программы

В данном примере BATCH-программа транслирует и редактирует 10 программ пользователя.

```
¤JOB/FODS
        TTYIO
        LET N=«0
LOOP:
.R MACRO
*PROG'N'=PROG'N'
.R LINK
*PROG'N'=PROG'N'
        %N
        IF(N—«9)—LOOP,—LOOP,END
END:
¤EOJ
```

4. ВЫПОЛНЕНИЕ ПРОГРАММЫ

Перед вызовом программы BATCH необходимо:

1) загрузить драйвер BA.SYS и драйвер регистрационного устройства по команде LOAD;

2) присвоить регистрационному устройству логическое имя LOG по команде ASSIGN;

3) присвоить устройству, предназначенному для вывода листингов, логическое имя LST по команде ASSIGN.

Для вызова BATCH с системного устройства следует подать с терминала команду R BATCH <ВК>
или BATCH <ВК>
после того, как монитор напечатает на терминале точку. После вызова BATCH печатает звездочку и ожидает ввода командной строки.

Если в это время нажать клавишу <ВК>, то BATCH печатает номер своей версии.

5. КОМАНДЫ ОПЕРАТОРА

5.1. Формат командной строки

Режим работы программы BATCH задается введением с терминала командной строки.

Формат командной строки:

[[выходспф] [,регспф] [/прк...]=] входспф[..., входспф]—[/прк...]

где выходспф — спецификация выходного файла (управляющего файла BATCH). Устройство для этого файла должно быть устройством произвольного доступа к данным. Если спецификация выходного файла не указана, BATCH создает управляющий файл на DK: с тем же именем, что и первый входной файл, и типом .CTL.

регспф — спецификация регистрационного файла. Устройство по умолчанию — LOG:, длина файла — 64 (десятичное) блока, тип файла — .LOG.

входспф — спецификация входного файла. Тип файла по умолчанию — .BAT. Если указан тип .CTL, BATCH предполагает управляющий файл (в этом случае он должен быть единственным входным файлом).

/прк — один из переключателей табл. 8.

5.2. Директивы оператора

Директивы оператора используются для обслуживания выполняющегося пакета заданий.

Таблица 8

Переключатель	Функция
1	2
/N	Транслирует файл команд, создает управляющий файл, генерирует сообщение ABORT JOB в начале регистрационного файла и передает управление монитору
/T:N	Определяет переключатель /NOTIME как переключатель по умолчанию для команды JOB (N=0); (для N=1 переключатель по умолчанию /TIME)
/U	Разгружает драйвер BA.SYS
/X	Указывает, что входной файл есть управляющий файл (если не указан тип .CTL)

Формат директивы:
\ДИР
где ДИР — одна из директив табл. 9.
Для того, чтобы ввести директиву, необходимо нажать клавишу <ВК>. Как только BATCH выполнит очередную команду из пакета заданий, она напечатает на терминале <ВК>, <ПС> и будет ожидать директивы оператора.

Пример:

```
.R BATCH
*KLEX
записать файлы на диск
?\A\E
\ECOPY DX1:FILE.MAC DX:
FILES COPIED:
DX1:FILE.MAC TO DX:FILE.MAC
\E\F\B
END BATCH
```

В этом примере входным файлом для BATCH является файл команд KLEX.BAT. BATCH печатает на терминале сообщение и ждет ответа. Оператор использует директивы BATCH для ввода команды монитору ФОДОС-2.

5.3. Окончание работы программы

После выполнения пакета заданий BATCH печатает на терминале сообщение END BATCH и передает управление монитору.

Для выхода из программы BATCH и передачи управления монитору необходимо: нажать клавишу <ВК> (после того,

Таблица 9

Директива	Функция
1	2
\@	Направляет последующие символы на системный терминал
\A	Обозначает ввод с системного терминала
\B	Обозначает ввод из пакета заданий
\C	Направляет последующие символы в регистрационный файл
\D	Рассматривает последующие символы как данные пользователя
\E	Направляет последующие символы монитору ФОДОС
\F	Форсирует вывод регистрационного буфера; если за этой директивой не следует другая, BATCH печатает сообщение об ошибке, прекращает выполнение задания и передает управление монитору ФОДОС-2
\G	Получает символы с системного терминала, пока не встретится возврат каретки
\HN	Управляет выводом в регистрационный файл; для N=0 регистрирует только .TTYOUT и .PRINT; для N=1 — .TTYOUT, .PRINT и .TTYIN; для N=2 не регистрирует .TTYOUT, .PRINT и .TTYIN; для N=3 регистрирует только .TTYIN
\IVXLABEL1? LABEL2? LABEL3?	Оператор IF вызывает условный переход, V — имя переменной от A до Z; X — значение, с которым сравнивается переменная V; LABEL1, LABEL2, LABEL3 — 6-символьные метки. Если длина метки менее 6 символов, дополнить пробелами. Если V—X<0, управление передается LABEL1; V—X=0 — LABEL2; V—X>0 — LABEL3. Направление для поиска метки указано ?; если ? есть 0, поиск начинается с начала задания; если ? есть 1, поиск метки начинается после оператора IF
\JLABEL?	Безусловный переход, LABEL — 6-символьная метка, ? — 0 или 1. если ? — 1, метка — обращение к элементу программы, находящемуся впереди по ходу ее выполнения; если ? — 0, метка — ссылка назад
\KV0	Увеличивает на 1 значение переменной V, V — имя переменной от A до Z
\KVLN	Хранит 8-битное число N в переменной V
\KV2	Возвращает значение переменной V в программу
\LLABEL	Вводит метку как 6-символьную буквенно-цифровую строку в пакет заданий. Метки не должны содержать символ (\). Все символы свыше 6 игнорируются

как BATCH выполнит очередную команду из пакета заданий, она напечатает <ВК>, <ПС>; напечатать \F и <ВК>. BATCH напечатает сообщение об ошибке и передает управление монитору.

Для немедленного выхода из программы следует дважды подать команду СУ/С.

6. СООБЩЕНИЯ ОПЕРАТОРУ

?BA—U—BC

Причина. В управляющем файле обнаружена ошибка (наиболее вероятно — после редактирования файла).

Действие. Проверить управляющий файл. Получить его повторно.

?BA—U—FE

Причина. Оператор подал директиву \F.

Действие. Не требуется.

?BA—U—IO

Причина. Ошибка чтения управляющего файла или записи в регистрационный файл (наиболее вероятно — переполнение регистрационного файла).

Действие. Увеличить размер регистрационного файла с помощью конструкции [:N].

?BA—U—LU

Причина. Программа BATCH не смогла найти свободный канал для использования. (Это возможно, если все 16 (десятичное) каналов открыты).

Действие. Проверить программу и убедиться, что канал оставлен открытым и не используется.

?BATCH—F—«¤»MISSING

Причина. В первой позиции командной строки отсутствует символ ¤.

Действие. Проверить задание.

?BATCH—F—ABORT JOB

Причина. Ошибка во время транслирования программой BATCH файла команд.

Действие. Распечатать файл и устранить ошибки.

?BATCH—F—AMBIGUOUS COMMAND

Причина. Неоднозначное определение команды.

Действие. Определить команду однозначно.

?BATCH—F—AMBIGUOUS OPTION

Причина. Неоднозначное определение переключателя.

Действие. Определить переключатель однозначно.

?BATCH—F—BATCH FATAL ERROR

Причина. Аппаратная ошибка во время работы BATCH.

Действие. Проверить исправность и правильность включения аппаратуры. Проверить диск на плохие блоки. Снять защиту записи.

?BATCH—F—BATCH HANDLER NOT RESIDENT

Причина. Драйвер BA.SYS отсутствует в памяти.

Действие. Загрузить драйвер по команде LOAD.

?BATCH—F—BATCH STACK OVERFLOW

Причина. Недопустимая глубина вложения команд ¤CALL в пакете заданий.

Действие. Проверить пакет заданий.

?BATCH—F—CHANNEL IN USE

Причина. Сбой в работе операционной системы.

Действие. Перезагрузить BATCH.

?BATCH—F—DISMOUNT ERROR

Причина. Указанное логическое имя устройства не существует.

Действие. Присвоить устройству логическое имя.

?BATCH—F—DUPLICATE OPTION

Причина. В управляющем операторе несколько раз указан один и тот же переключатель.

Действие. Исправить пакет заданий.

?BATCH—F—EOF WITH NO ¤EOJ

Причина. Файл не ограничен командой ¤EOJ.

Действие. Проверить задание.

?BATCH—F—FILE NOT FOUND

Причина. Входной файл не найден или для входного файла, который не является управляющим, определен переключатель /X.

Действие. Проверить введенную командную строку.

?BATCH—F—INPUT ERROR

Причина. Аппаратная ошибка чтения файла команд (.BAT).

Действие. Проверить исправность и правильность включения аппаратуры. Проверить диск на плохие блоки.

?BATCH—F—INPUT FILE

Причина. В командной строке не указан входной файл.

Действие. Ввести правильную командную строку.

?BATCH—F—INSUFFICIENT MEMORY

Причина. Программе BATCH недостаточно памяти для загрузки драйвера устройства, необходимого для операций ввода/вывода или для выполнения команд ¤MOUNT или ¤DISMOUNT в пакете заданий.

Действие. Проверить командную строку или команды ¤MOUNT или ¤DISMOUNT в пакете заданий.

?BATCH—F—INVALID '+'

Причина. Неправильное использование символа «+», например, в команде ¤RUN, или в выходной спецификации.

Действие. Проверить задание (командную строку).

?BATCH—F—INVALID CHARACTER

Причина. Указан недопустимый символ кода КОИ-7.

Действие. Распечатать регистрационный файл для обнаружения ошибки.

?BATCH—F—INVALID COMMAND LINE

Причина. В командной строке обнаружена ошибка.

Действие. Ввести правильную командную строку.

?BATCH—F—INVALID CONSTRUCTION

Причина. Неправильный формат оператора IF; недопустимая команда «текст».

Действие. Проверить задание.

?BATCH—F—INVALID COPY OF HANDLER

Причина. Копия драйвера BA.SYS в оперативной памяти имеет ошибку.

Действие. Разгрузить драйвер и загрузить его повторно.

?BATCH—F—INVALID DEVICE

Причина. В командной строке указано несуществующее или недопустимое устройство.

Действие. Проверить введенную командную строку. Использовать другое устройство.

?BATCH—F—INVALID LOG DEVICE

Причина. Указано недопустимое регистрационное устройство (MT: или PC:).

Действие. Проверить введенную командную строку. Присвоить имя LOG другому устройству.

?BATCH—F—INVALID OPTION

Причина. В командной строке обнаружен недопустимый переключатель.

Действие. Ввести правильную командную строку.

?BATCH—F—INVALID OPTION COMBINATION

Причина. В управляющем операторе обнаружена недопустимая комбинация переключателей.

Действие. Ввести правильную командную строку.

?BATCH—F—INVALID SEQUENCE ARGUMENT

Причина. Идентифицирующий номер в команде ¤SEQUENCE не является числовым.

Действие. Исправить команду.

?BATCH—F—INVALID VARIABLE

Причина. Указанная переменная не является буквой (A—Z).

Действие. Проверить задание.

?BATCH—F—INVALID VID

Причина. Идентификатор тома в команде ¤MOUNT определен неправильно.

Действие. Исправить команду.

?BATCH—F—LINE TOO LONG

Причина. Длина командной строки более 80 символов.

Действие. Ввести правильную командную строку.

?BATCH—F—LOG DEVICE ERROR

Причина. Аппаратная ошибка при выводе на регистрационное устройство.

Действие. Проверить исправность и правильность включения аппаратуры. Проверить диск на плохие блоки. Снять защиту записи.

?BATCH—F—NO CONTROL FILE

Причина. Попытка записи управляющего файла (.CTL) на устройство нефайловой структуры.

Действие. Исправить командную строку.

?BATCH—F—NO ¤EOJ

Причина. В пакете заданий командам ¤JOB или ¤SEQUENCE не предшествует команда ¤EOJ конца предыдущего задания.

Действие. Исправить пакет заданий.

?BATCH—F—NO FILE

Причина. Спецификация файла отсутствует, где она необходима.

Действие. Исправить пакет заданий.

?BATCH—F—NO FILE NAME BEFORE «.»

Причина. В спецификации файла указан тип файла, но отсутствует его имя.

Действие. Указать правильную спецификацию файла.

?BATCH—F—NO ',' IN ¤LIB

Причина. В команде ¤LIB используется символ «,».

Действие. Исправить команду.

?BATCH—F—NO LOGICAL DEVICE

Причина. В команде ¤MOUNT не указано логическое имя устройства.

Действие. Исправить команду.

?BATCH—F—NO PHYSICAL DEVICE

Причина. В команде ¤MOUNT не указано физическое имя устройства.

Действие. Исправить команду.

?BATCH—F—OUTPUT DEVICE FULL

Причина. На указанном устройстве недостаточно места для управляющего файла (.CTL).

Действие. Сжать том по команде SQUEEZE. Стереть или записать на другой том ненужные файлы. Использовать другой том.

?BATCH—F—OUTPUT ERROR

Причина. Устройство вывода для управляющего файла — магнитная лента, либо ошибка при выводе управляющего файла.

Действие. Проверить введенную командную строку. Проверить исправность и правильность включения аппаратуры. Проверить диск на плохие блоки. Снять защиту записи.

?BATCH—F—OUTPUT FILE NOT OPEN

Причина. Сбой в работе операционной системы. Возможно, что ошибка в BATCH.

Действие. Использовать другую копию BATCH.

?BATCH—F—PLEASE ASSIGN LOG,LST

Причина. Устройство листинга или регистрационное устройство не определено.

Действие. Присвоить устройствам логические имена LOG: и LST: по команде ASSIGN.

?BATCH—F—PLEASE LOAD LOG HANDLER

Причина. Драйвер регистрационного устройства отсутствует в памяти.

Действие. Загрузить драйвер по команде LOAD.

?BATCH—F—PROTECTED FILE ALREADY EXISTS

Причина. Попытка создать файл с именем уже существующего защищенного файла.

Действие. Возможны следующие действия:

— использовать команду монитора UNPROTECT или переключатель PIP/Z для изменения защиты существующего файла;

— использовать другое имя для создания нового файла.

?BATCH—F—RETURN FROM CALL ERROR

Причина. Ошибка при возврате из подпрограммы (¤CALL).

Действие. Проверить управляющий файл; если необходимо, получить его повторно. Повторить операцию.

?BATCH—F—SEPARATOR MISSING

Причина. Спецификация файла не ограничена пробелом, символом «+», точкой или возвратом каретки.

Действие. Исправить команду.

?BATCH—F—TOO MANY FILE DESCRIPTORS

Причина. В команде определено более 6-ти спецификаций файлов.

Действие. Исправить команду.

?BATCH—F—TOO MANY OUTPUT FILES

Причина. В командной строке определено более двух выходных файлов.
Действие. Ввести правильную командную строку.

?BATCH—F—UNKNOWN COMMAND
Причина. В задании обнаружена недопустимая команда.
Действие. Проверить и исправить задание.

ПРИЛОЖЕНИЕ

ПРОВЕРКА РАБОТЫ BATCH

Ниже приведен пример задания, с помощью которого осуществляется проверка работы BATCH. В этом задании BATCH создает программу на языке АССЕМБЛЕР, затем транслирует ее, редактирует и запускает. (Задание создается пользователем по редактору текста с именем EXAMPL.BAT).

```
¤JOB
¤MESSAGE      пример задания BATCH
¤MESSAGE      создание программы
¤CREATE/LIST  EXAMPL.MAC
              .ENABL LC
.SBTTL пример для BATCH
              .MCALL .PRINT,.EXIT
      START:  .PRINT #MESSAG
              .EXIT
      MESSAG: .ASCIZ (пример программы для BATCH)
              .EVEN
              .END START
¤EOD
¤MACROEXAMPL EXAMPL/OBJ EXAMPL/LIST
¤LINK   EXAMPL EXAMPL/EXECUTE
¤PRINT/DELETE EXAMPL.LST
¤MESSAGE      пуск программы
¤RUN    EXAMPL
¤DELETE EXAMPL.OBJ+EXAMPL.SAV+EXAMPL.MAC
¤MESSAGE      печать справочника
¤DIRECTORY    DK:EXAMPL.*
¤MESSAGE      конец примера
¤EOJ
```

Для выполнения этого задания необходимо подать с терминала следующую последовательность команд (BATCH печатает сообщения из задания на терминале):

```
.LOAD BA
.ASSIGN DX1 LOG
```

```
.ASSIGN DX1 LST
.R BATCH
*EXAMPL
пример задания BATCH
создание программы
пуск программы
печать справочника
конец примера
END BATCH
```

В результате выполнения задания BATCH создает следующий регистрационный файл:

```
¤JOB
¤MESSAGE        пример задания BATCH
¤MESSAGE        создание программы
¤CREATE/LIST  EXAMPL.MAC
                .ENABL LC
.SBTTL пример для BATCH
                .MCALL .PRINT,.EXIT
     START:     .PRINT #MESSAG
                .EXIT
     MESSAG:    .ASCIZ (пример программы для BATCH)
                .EVEN
                .END START
     ¤EOD
¤MACRO EXAMPL EXAMPL/OBJ EXAMPL/LIST
¤LINK EXAMPL EXAMPL/EXECUTE
¤PRINT/DELETE EXAMPL.LST
¤MESSAGE        пуск программы
¤RUN EXAMPL
пример программы для BATCH
¤DELETE EXAMPL.OBJ+EXAMPL.SAV+EXAMPL.MAC
¤MESSAGE        печать справочника
¤DIRECTORY      DK:EXAMPL.*
EXAMPL.LOG      64      EXAMPL.BAT      2
EXAMPL.CTL      2       EXAMPL.CTT      1
  4 FILES, 69 BLOCKS
  3521 FREE BLOCKS
¤MESSAGE        конец примера
¤EOJ
```

СОДЕРЖАНИЕ

УСТАНОВКА И ГЕНЕРАЦИЯ СИСТЕМЫ РУКОВОДСТВО СИСТЕМНОГО ПРОГРАММИСТА

ПРОГРАММИРОВАНИЕ ПЕРИФЕРИЙНЫХ УСТРОЙСТВ РУКОВОДСТВО СИСТЕМНОГО ПРОГРАММИСТА

МОНИТОР РАСШИРЕННОЙ ПАМЯТИ РУКОВОДСТВО ПРОГРАММИСТА

КОНТРОЛЬНЫЕ ЗАДАЧИ. РУКОВОДСТВО ОПЕРАТОРА

ПРОГРАММА ПАКЕТНОЙ ОБРАБОТКИ. РУКОВОДСТВО ОПЕРАТОРА

Ответственный за выпуск М. Г. Бойкова
Редактор Т. А. Савельева
Корректор В. Н. Лыткина

Сдано в набор 2.03.90. Подписано в печать 23.05.90.
Формат 60×84¹/₁₆. Бумага писчая № 1. Гарнитура литературная.
Печать высокая. Усл. печ. л. 16,8. Тираж 20 000 экз. Заказ 682.

Ленинградское отделение РППО «Союзбланкоиздат»

Великолукская городская типография управления издательств, полиграфии и книжной торговли Псковского облисполкома, 182100, г. Великие Луки, ул. Полиграфистов, 78/12.