ATARI HOME COMPUTER SYSTEM

OPERATING SYSTEM USER'S MANUAL

Операционная система
домашнего компьютера АТАРИ

Руководство пользователя

ATARI®

A Warner Communications Company

ATARI HOME COMPUTER SYSTEM
OPERATING SYSTEM USER'S MANUAL

Операционная система домашнего компьютера АТАРИ

Руководство пользователя

Перевод с английского Рачинского Сергея
Общая редакция Селиванова Игоря и Пашкова Дмитрия

Русский текст подготовлен с помощью редактора текстов
SPEEDSCRIPT на компьютере ATARI XE

Молодежный Компьютерный Центр "ВАРИАНТ"
МОСКВА, 1989 г.

Операционная система домашнего компьютера АТАРИ. Руководство пользователя.

ПРИЛОЖЕНИЯ

ПРЕДИСЛОВИЕ

Данное руководство посвящено описанию резидентной операционной системы (ОС) домашнего компьютера ATARI и предназначено для пользователей, знакомых с внутренним поведением системы. В руководстве обсуждаются следующие вопросы:

- Системные функции и методы их использования.
- Взаимодействие и организация подсистем.
- Характеристики периферийных устройств для компьютеров ATARI XL, XE.
- Прогрессивная технология, позволяющая повысить основные возможности ОС.

Читателю было бы полезно ознакомиться с понятиями программирования и его технологией, с языком ассемблера в общем и в частности с версией SYNERTEK 6502, с основами цифровой аппаратуры и связанной с ней терминологией. Для использования всех ресурсов ОС вы будете снабжены всей необходимой информацией, и вам не придется прибегать к методу проб и ошибок и распечатке ОС. Также приведена информация, требующаяся заданиям и включающая в себя ссылки на листинг ОС. Данное руководство не представляет законченного описания аппаратного обеспечения, используемого операционной системой. Кто хочет получить дополнительные сведения, необходимо обратиться к "Руководству по аппаратному обеспечению домашнего компьютера ATARI" ("ATARI HOME COMPUTER HARDWARE MANUAL").

1. Введение.

Общее описание домашней вычислительной системе ATARI.

В целом, операционные системы компьютеров ATARI 800XL, ATARI 65XE и ATARI 130XE идентичны. Однако между ними имеются следующие различия:

- Конструктивное оформление.
- Компьютер ATARI 800XL и ATARI 65XE имеет ОЗУ 64К, ATARI 130XE, в свою очередь, имеет ОЗУ 128К.

Компоновка схем аппаратного обеспечения

- обеспечивает как символьную, так и точечную графику для черно-белых и цветных телевизоров,
- обеспечивает четыре независимых звуковых аудиоканала (управляемых частотой), взаимодействующих с звуковой системой телевизора,
- согласуется с двумя джостиками и четырьмя PADDLE контроллерами,
- согласуется с последовательной шиной ввода/вывода,
- содержит встроенную клавиатуру.

На рисунке 1-1 приведена упрощенная блок-схема аппаратного обеспечения. В руководстве по аппаратному обеспечению можно найти сопутствующую документацию.

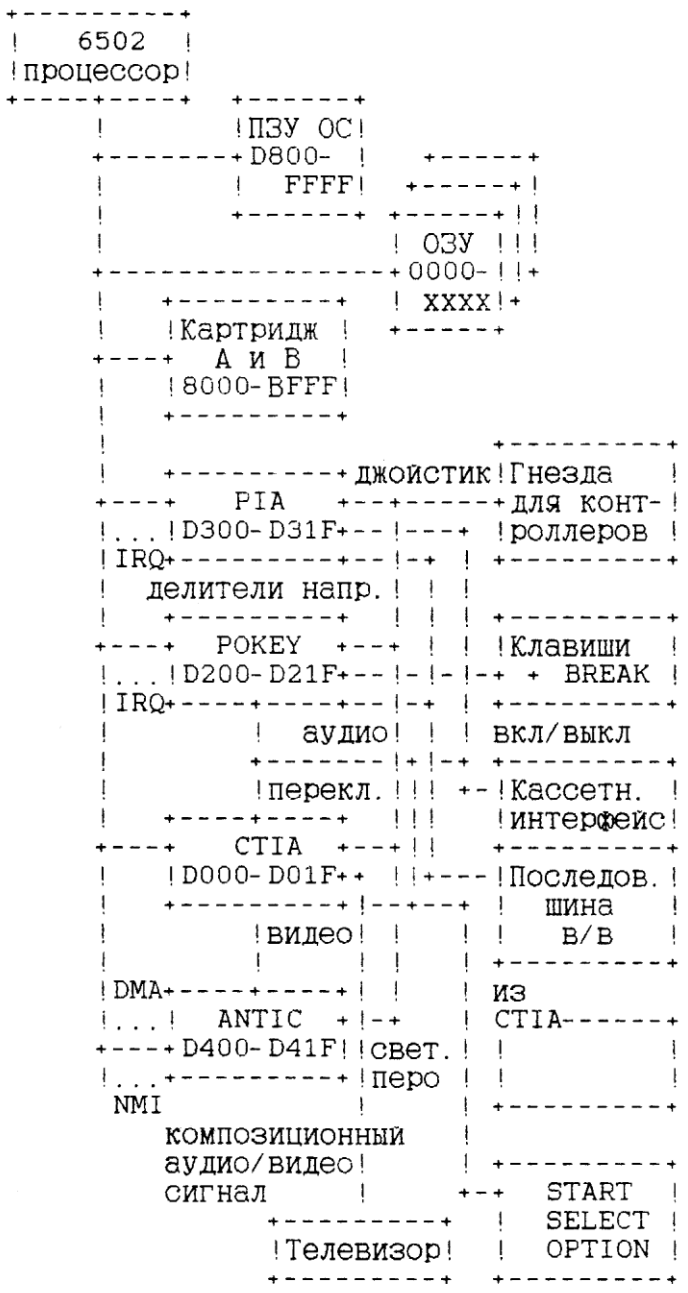


Рис. 1-1 Структурная схема домашнего компьютера ATARI.

ANTIC, CTIA, PIA и POKEY - специальные микросхемы. ANTIC обрабатывает немаскируемые прерывания. POKEY - прерывания по запросам.

IRQ - прерывания по запросам. Используются для связей по портам периферийных устройств, таймера и ввода с клавиатуры.

NMI - немаскируемые прерывания (нажатие RESET, прерывания дисплейной программы и др.).

Обозначения, используемые в руководстве.

В данном руководстве используются следующие обозначения:

Шестнадцатеричные числа.

Все числа, состоящие из двух цифр, которым предшествует знак доллара "$", являются шестнадцатеричными. Все остальные числа (за исключением адресов в памяти) записаны в десятичной системе исчисления, если об этом не сказано в тексте.

Адреса памяти.

Все ссылки на компьютерную память и ячейки ввода/вывода производятся в шестнадцатеричной системе исчисления. Адреса памяти могут быть записаны как в скобках так и без них. (Например: (D20F) и D20F - один и тот же адрес.)

Килобайты памяти.

Объем памяти часто измеряется в килобайтах (32K). Один килобайт составляет 1024 байт памяти.

Паскаль как алгоритмический язык.

В некоторых местах, где необходимо детальное рассмотрение алгоритма, в качестве языка спецификаций используется язык Паскаль (PASCAL), имеющий структуру процедур-блоков. Синтаксис языка Паскаль похож на синтаксис любого другого структурного языка, и у пользователя не должно возникнуть трудностей при встрече с ним.

Схема памяти.

Диаграмма, подобная рисунку 1-2, используется всякий раз при необходимости наглядного представления байт и таблиц.

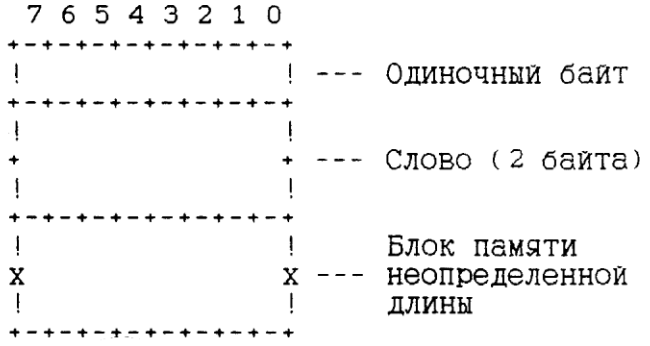


Рис. 1-2 Схема представления памяти.

Бит 7 является старшим битом байта, а бит 0 - младшим битом. В таблицах и рисунках адреса памяти возрастают обычно от верхней части рисунка к нижней.

Форма Бекуса-Наура.

Видоизмененная версия формы Бекуса-Наура используется для представления некоторых синтаксических форм, которые включают в себя следующие металингвистические символы:

::= - оператор присваивания.
() - метасинтаксическая переменная.
: - отделяет возможные подстановки друг от друга.
// - необязательное использование.

Кроме того, используются синтаксическая константа, значение которой устанавливается в каждом отдельном случае.

Например:

(спецификация устройства)::=(имя устройства)/(номер устройства)/:
(имя устройства)::=C:D:E:K:P:R:S
(номер устройства)::=1:2:3:4:5:6:7:8

"Спецификация устройства" требует обязательного наличия "имени устройства", к которому может быть добавлен "номер устройства", и обязательно в конце должно следовать двоеточие. Имя устройства должно соответствовать одному из символов, приведенных в качестве возможных. Номер устройства (в случае его наличия) должен быть записан цифрой от 1 до 8.

Имена переменных ОС.

В списке присваиваний программы ОС устанавливаются имена векторов ПЗУ и ОЗУ, имена переменных базы данных ОЗУ и имена аппаратных регистров. На эти имена и осуществляется в дальнейшем ссылка. Когда используется какое-нибудь из этих имен, то обычно требуется указать адрес в памяти (например BOOTAD 0242).

2. Функциональная организация операционной системы.

Данный раздел, используя общие термины, описывает различные подсистемы резидентной ОС.

Подсистема ввода/вывода.

Подсистема ввода/вывода обеспечивает быстрое взаимодействие между программой и аппаратными средствами. Большинство функций, например считывание или печать символов, являются независимыми от устройств, хотя соответствующее обеспечение имеется и для зависимых от устройств функций. Все периферийные устройства, связанны с символами, являющимися специальными индивидуальными именами (такие как K,D,P и др.), а доступ к ним может быть осуществлен при помощи процедуры центрального ввода/вывода (CIO). Аппаратные регистр обеспечивает доступ к контроллерам (джойстикам и контроллерам PADDLE), такой доступ происходит без поддержки (CIO). Эта информация периодически корректируется для отображения состояния этих устройств.

Обработка прерываний.

Система прерываний управляет всеми прерываниями от аппаратных средств обычным и последовательным способом. По умолчанию все прерывания обрабатываются ОС. По вашему усмотрению отдельные прерывания (или группа прерываний) могут обрабатываться прикладной программой.

Инициализация.

Инициализация системы происходит в двух случаях: при включении питания и перезагрузке системы. Инициализация происходит всякий раз, когда сетевой тумблер переключается в положение ON (вкл.), а инициализация при перезагрузке происходит при нажатии клавиши (RESET).

Включение питания.

При включении питания ОС тестирует память и фиксирует ее конфигурацию. При включении питания система выполняет следующие действия:
- определяет наибольший адрес ОЗУ.
- устанавливает нулевые значения всех ячеек ОЗУ.
- устанавливает вектора прерываний ОЗУ.
- форматирует таблицу устройств.
- производит инициализацию картриджей.
- переводит экран в текстовой режим 24x40.
- при подключении кассетного магнитофона производит загрузку с кассеты.
- проверяет картридж на необходимость загрузки с дискеты.
- производит загрузку с дискеты в случае наличия соответствующего указания и при подключении дисковода.
- передает управление картриджу, программе, загружаемой с дискеты или кассеты, или же программе самотестирования.

Перезагрузка системы.

Нажатие клавиши (RESET) приводит к следующим действиям ОС:
- очистке части ОЗУ.
- перепроверке верхней границы ОЗУ.
- установке векторов прерывания.
- форматированию таблицы устройств.
- иницилизации картриджей.
- переводу экрана в текстовой режим 24x40.
- передаче управления картриджу, программе, загружаемой с дискеты или кассеты, или же программе самотестирования.

Заметим, что нажатие клавиши (RESET) не производит всех действий, что и включение компьютера.

Пакет математики с плавающей запятой.

ПЗУ ОС содержит пакет математики с плавающей запятой (ПЗ), доступный нерезидентным программам таким, как ATARI Бейсик. Сам по себе, пакет не используется другими частями ОС. Числа с плавающей запятой хранятся в памяти в виде двоично-десятичных цифр мантисы и и одного байта экспоненты. Пакет содержит следующие подпрограммы:
- перевод числа из ASCII в число с плавающей запятой и наоборот.
- перевод целого числа в число с плавающей запятой и наоборот.
- сложение, вычитание, умножение и деление чисел с плавающей запятой.
- логарифмическая, экспоненциальная и полиномиальная оценка числел с плавающей запятой.
- очистка, загрузка, хранение и перенос числа с плавающей запятой.

3. Конфигурации (комплекты оборудования).

Компьютеры ATARI XL, XE комплектуются различными компонентами с индивидуальной операционной средой:
- Картриджи могут как использоваться, так и не использоваться.
- По необходимости к компбютеру ATARI может быть присоединена дополнительная память.
- К последовательной шине ввода/вывода возможно подключение различных периферийных устройств.

ОС воспринимает данные свойства без изменения резидентной ОС (см. Раздел 2). Конфигурация машины выясняется при первоначальном включении в сеть и не уточняется до следующей перезагрузки системы. Ниже приведено описание некоторых часто встречающихся конфигураций.

Внешняя среда программ.

ОС позволяет передать управление в любое время одному из четырех типов программ:

- программе самотестирования,
- картриджу,
- программе, загруженной с дискеты,
- программе, загруженной с кассеты.

Вопрос о передаче управления решается на основании информации в картриджах, ввода с клавиатуры, и в зависимости от того подключен дисковод или нет. Детальное описание алгоритмов можно найти в разделе 7.

Режим "самотестирования".

В режиме "самотестирования" производится проверка внутренней памяти компьютера, клавиатуры, аудио и видео режимов. Режим "самотестирования" является внешней средой с самым низким приоритетом. Попасть в него можно из среды высшего приоритета или по умолчанию, т.е. когда никакая другая внешняя среда не задана. Например, в Бейсике оператор BYE переводит ОС в режим "самотестирования". Выход из этого режима осуществляется нажатием клавиши (RESET), в случае если переход в него осуществлялся из среды с высшим приоритетом.

Картридж.

Вставленный картридж обычно обеспечивает основное управление после завершения инициализации. Например: ATARI Бейсик (ATARI BASIC), Баскетбол (BASKETBALL), Компьютерные шахматы (COMPUTER CHESS) и др. Все эти программы на картриджах взаимодействуют непосредственно с пользователем. Хотя картридж и может обеспечить выполнение поддерживающих функций для некоторой другой операционной среды, но пока этого не было сделано. Некоторые картриджи (особенно ориентированные на клавиатуру) могут изменять среду при использовании специальных команд (например BYE) для перехода в режим "самотестирования" или DOS для обращению к дисководу. Другие картриджи не изменяют внешнюю среду. Имейте в виду, аппаратная блокировка не позволяет менять или вставлять картридж при включенном компьютере. Данная особенность вызывает переинициализацию системы при замене картриджа.

Загрузка с дискеты.

Загрузка с дискеты может произойти, а может и не произойти при попытке использования программ на дискетах. Здесь подразумевается, что загрузка все же имела место. Условия загрузки описаны в разделе 7. Управление может быть передано программе на дискете также, как программе обработки файлов. Программа на дискете может обеспечить выполнение некоторых поддерживающих функций, как это делает система обработки файлов. Данная среда является настолько гибкой, что трудно обобщить все ее возможности и ограничения. Единственным требованием является наличие достаточного объема ОЗУ для размещения загружаемой программы.

Загрузка с кассеты.

Внешняя среда для загрузки с кассеты похожа на внешнюю среду закрузки с дискеты, хотя магнитофон является ограниченным устройством ввода/вывода. Он медленно работает и может осуществить лишь последовательный доступ. Заметим, что свойства кассетной загрузки не имеют никакого отношения к хранению на кассетах программ на языках высокого уровня (например ATARI Бейсик) и использованию кассет для хранения данных.

Периферийные устройства.

К системе могут быть подключены периферийные устройства различных типов. Подсоединение осуществляется при помощи обычных соединительных шнуров к последовательной шине или к гнездам передней панели компьютера. Основными функциями устройств является перенос байтов данных (обычно это характерно для устройств последовательной шины) или переноса контактной информации (характерно для игровых контроллеров).

Игровые контроллеры.

ОС периодически (50 или 60 раз в секунду) запрашивает о состоянии стандартных игровых контроллеров (PADDLE или джойстиков) и сохраняет полученные величины в ОЗУ. Присоединение, отсоединение и перекомпоновка данных контроллеров не оказывает никакого влияния на системные операции, поскольку система всегда осуществляет считывание данных со всех этих контроллеров. С управляемых контроллеров происходит считывание, однако информация поступающая в ОС не расшифровывается. Для того, чтобы осуществить считывание с контроллеров необходимо использовать специальные команды (см. Раздел 11).

Магнитофон для записи программ.

Магнитофон ATARI XC12 является специальным периферийным устройством. Для пересылки и получения данных используется последовательная шина. Магнитофон не согласует свои действия с другими периферийными устройствами, подключенными к последовательной шине. Он является последним по счету устройством последовательной шины, поскольку у него нет дополнительного разъема, как у других периферийных устройств. По той самой причине невозможно подсоединение более чем одного магнитофона к одной системе. Система не может обнаружить наличие или отсутствие магнитофона, поэтому его можно подключить или отключить в любой момент времени.

Устройства последовательной шины.

Работа устройств последовательной шины согласуется с протоколом последовательной шины ввода/вывода, за исключением магнитофона. Каждое устройство последовательной шины имеет два одинаковых разъема. Они полностью идентичны и соеденены между собой. Периферийные устройства, соединяются последовательно, в любом порядке, поскольку каждое из них имеет свой идентификатор. Об ограничениях, которые все же существуют будет рассказано в разделе 5.

4. Использование системной памяти.

Память в системе декодируется в полный 64-килобайтный диапозон микрокомпьютера 6502, и дальнейшее расширение памяти не представляется возможным. Память делится на четыре основных области (возможно перекрывание некоторых из них): ОЗУ, область картриджа, область аппаратных регистров и ПЗУ резидентной ОС. Области и их граничные адреса перечислены ниже (все адреса записны в шестнадцатеричной форме):

```
0000-1FFF = ОЗУ (минимально необходимая для операций)
2000-7FFF = Область расширения ОЗУ
8000-9FFF = Картридж А, картридж В (половина от 16К) или ОЗУ
A000-BFFF = Картридж А или ОЗУ
C000-CFFF = Не используется
D000-DFFF = аппаратных регистров
D800-DFFF = Пакет математики с плавающей запятой (ОС)
E000-FFFF = ПЗУ резидентной ОС
```

В данном разделе мы проведем разбиение этих областей на более мелкие функциональные подобласти и подробно остановимся на их использовании.

Область ОЗУ.

Область ОЗУ используется ОС и управляющей программой. Область ОЗУ может быть подразделена на следующие подобласти, с целью побробного знакомства с ней:

Нулевая страница - область адресов нулевой страницы 6502.
Первая страница - область стека 6502.
Страницы со 2 по 4 - переменные ОС и рабочая область пользователя.
Страницы с 7 по xx - свободная область ОЗУ для возможной загрузки мат. обеспечения.(x)
Страницы с xx по верхнюю границу ОЗУ - дисплейная программа и ее данные.(x)
xx - зависит от экранного режима графики и количества свободной памяти ОЗУ.

В следующем параграфе говорится о том, как подобласти ОЗУ используются оперативной системой, а также приводятся некоторые рекомендации к программам пользователя.

Нулевая страница.

Системы команд микропроцессора 6502 и способы адресации выделяют нулевую страницу. Адресация на этой странице (адреса с 0000 по 00FF) происходит гораздо быстрее с использованием меньшего числа командных байт и обеспечивает единственный механизм косвенной адресации. Нулевую страницу следует использовать с особой бережливостью, чтобы любой пользователь мог бы воспользоваться ее частью. Нижную половину нулевой страницы (0000-007F) постоянно занимает операционная система. Данная часть страницы не может быть использована никакой другой программой пока работает ОС и все прерывания, обрабатываемые ОС не будут сняты. Верхняя половина нулевой страницы (0080-00FF) доступна лишь с одним ограничением: при использовании пакета математики с плавающей запятой нельзя использовать ячейки с 00D4 по 00FF.

Первая страница.

Первая страница отведена под аппаратный стек 6502. Команды JSR, PHA и все прерывания влекут за собой запись данных на первую страницу. Использование команд RTS, PLA, RTI влечет за собой, наоборот, считывание данных с первой страницы. Наличие 256-байтового стека вполне достаточно для вызова обычных подпрограмм и обработки прерываний. На использование первой страницы не наложено никаких ограничений. Очевидно, что размера стека явно недостаточно для глубоких рекурсивных процессов или для процессов, в которых каждый раз приходится запоминать громозкие внешние условия. Поэтому для решения серьезных задач следует организовывать собственные стеки. Указатель стека 6502 устанавливается на ячейку 01FF при включении компьютера и при перезагрузке системы. Затем происходит заполнение стека до ячейки 0100. В случае переполнения, стек, с учетом особенностей регистра 6502 - указателя на 8-битовый стек, заполняется с ячейки 0100 и 01FF.

Ячейки с 0200 по 047F отводятся под рабочие переменные ОС, таблицы и буферы данных. Часть этой области может быть задействована лишь при уверенности в том, что конфликтов с ОС не возникнет. Например, буферы принтера и магнитофона могут быть задействованы при отсутствии операций с ними. Объем работ, проводимых с целью выявления отсутствия конфликтов, не идет ни в какое сравнение с приобретенными выгодами. Поэтому страницы со 2 по 4 рекомендуется использовать исключительно для нужд операционной системы.

Рабочая область пользователя.

Ячейки с 0480 по 06FF предназначены для использования программами, за исключением того случая, когда задействован пакет математики с плавающей запятой. Он использует ячейки с 057E по 05FF.

Область загрузки.

Седьмая страница является началом "области загрузки". Когда программа загружается с дискеты или кассеты, область загрузки может начинаться с наименьшего адреса свободного участка памяти (каким является 0700) и заканчиваться выше. Хотя она может начинаться с любого адреса выше 0700 и ниже экранной дисплейной программы. Вершина области загрузки определяет начало "свободной памяти". Когда процесс загрузки завершен, указатель содержит следующей после последней загруженной ячейки адрес. Если ни одна программа не загружена, то указатель содержит величену 0700.

Экранные данные и дисплейная программа.

Когда ОС осуществляет управление экраном дисплея, дисплейная программа, определяющая характеристики изображения и данные на экране, расположена в области ОЗУ до его наивысшего адреса. Нижняя часть данной области определяет конец свободной памяти и зависит от экранного режима, действующего в данный момент. Указатель в базе данных содержит адрес последней допустимой ячейки ниже области экрана.

Область свободной памяти.

К области свободной памяти относится вся область ОЗУ между областью загрузки и началом экранной памяти. Управление областью свободной памяти осуществляется прикладной программой высокого уровня.

Картриджи A и B.

Для вставных картриджей отводятся две восьмикилобайтные области памяти. В компьютерах ATARI XE,XL имеется картридж B, и для его использования отводятся ячейки памяти с 8000 по 9FFF. Картридж A использует ячейки памяти с адресами с A000 по BFFF и иногда ячейки с адресами с 8000 по BFFF для картриджей в 16K. Картридж располагается на тех же адресах, что и ячейки ОЗУ и если он всиавлен объем ОЗУ уменьшается.

Аппаратные регистры.

6502 обращается к аппаратным регистрам как к памяти. Некоторые регистры являются доступными для чтения и записи, в то время как остальные допускают или только чтение или только запись. (Описание всех аппаратных регистров приведено в руководстве к аппаратному обеспечению компьютеров ATARI.) В области аппаратных регистров с D000 по D7FF задействованы лишь следующие ее подобласти.

```
D000-D01F -- CTIA
D200-D21F -- POKEY
D300-D31F -- PIA
D400-D41F -- ANTIC
```

Резидентная ОС и пакет математики с плавающей запятой ПЗУ.

Область, начиная с ячейки D800 по FFFF всегда занята ОС или пакетом математики с плавающей запятой. Следует позаботиться о том, чтобы не использовать случайных точек входа в программы ОС, неподвижность которых не

гарантирована, чтобы в дальнейшем дать возможность сформировать другую ОС функционально совместимую с данной. ОС содержит достаточно векторных входных неподвижных точек в конце ПЗУ и ОЗУ. Пакет математики с плавающей запятой не является векторным, но все документированные входные точки фиксируется. Не следует использовать недокументированные программы, полученные сканированием листинга. Список фиксированных векторов ПЗУ можно найти в приложении J.

Свободная область памяти.

Область свободной памяти расположена между областью загрузки и началом экранной области памяти. Таким образом границы ее являются переменными. MEMLO (02E7) определяет нижнюю часть свободной памяти, а MEMTOP (02E5) определяет ее верхнюю часть. В данном разделе приводятся условия, приводящие к установке и изменению значений данных переменных.

Процесс инициализации ситемы.

ОС определяет размер наименьшего по протяженности блока ОЗУ и сохраняет в памяти границы. Тогда открывается экранный редактор, помещая новую величину в MEMTOP. Возможное помещение в память программы, загруженной с дискеты или кассеты, вызывает обновление величины (более высокой) в MEMLO (см. Раздел 7). MEMLO и MEMTOP будут определять максимальный объем свободной доступной памяти, когда управление окончательно перейдет к управляющей программе. В дальнейшем данный объем свободной памяти может уменьшиться, как это описано в следующем параграфе.

Изменение экранных режимов.

Дисплей использует переменные APPMHI (000E) для хранения адреса, ниже которого расширение MEMTOP невозможно. Это дает возможность защитить часть свободной памяти от перезаписи, которая может возникнуть в связи с изменением экранных режимов. Дисплей установит нулевой режим экрана, скорректирует MEMTOP и укажет на ошибку, если вы потребуете расширения экранной памяти вследствие изменения экранного режима ниже уровня APPMHI. В остальных случаях дисплей установит желаемый экранный режим и скорректирует MEMTOP.

5. Подсистема ввода/вывода.

В данном разделе рассматривается подсистема ввода/вывода операционной системы. Подсистема ввода/вывода представляет собой набор программ, позволяющих осуществить доступ к устройствам на трех различных уровнях. Центральный ввод/вывод (CIO) обеспечивает наивысший уровень доступа к устройствам. Второй уровень позволяет связаться непосредственно с устройством. Самым низким уровнем доступа является последовательный ввод/вывод (SIO). Любой другой более низкий уровень доступа к устройствам подразумевает непосредственное чтение и запись в аппаратные регистры, связанные с устройством. Байт данных - основная единица ввода/вывода. Байт данных может содержать как двоичную (нетекстовую), так и кодированную (текстовую) информацию. Схема кодирования текста, поддерживаемая ОС носит название ATASCII, образованное от слов "ATARI ASCII". Большинство ATASCII кодов совпадают с ASCII с первоначальным отклонением управляющих кодов. В приложении D приведен знакогенератор ATASCII, а в приложениях E,F и G указана реализация этих кодов для дисплея клавиатуры и принтера. Структура подсистемы ввода/вывода приведена ниже.

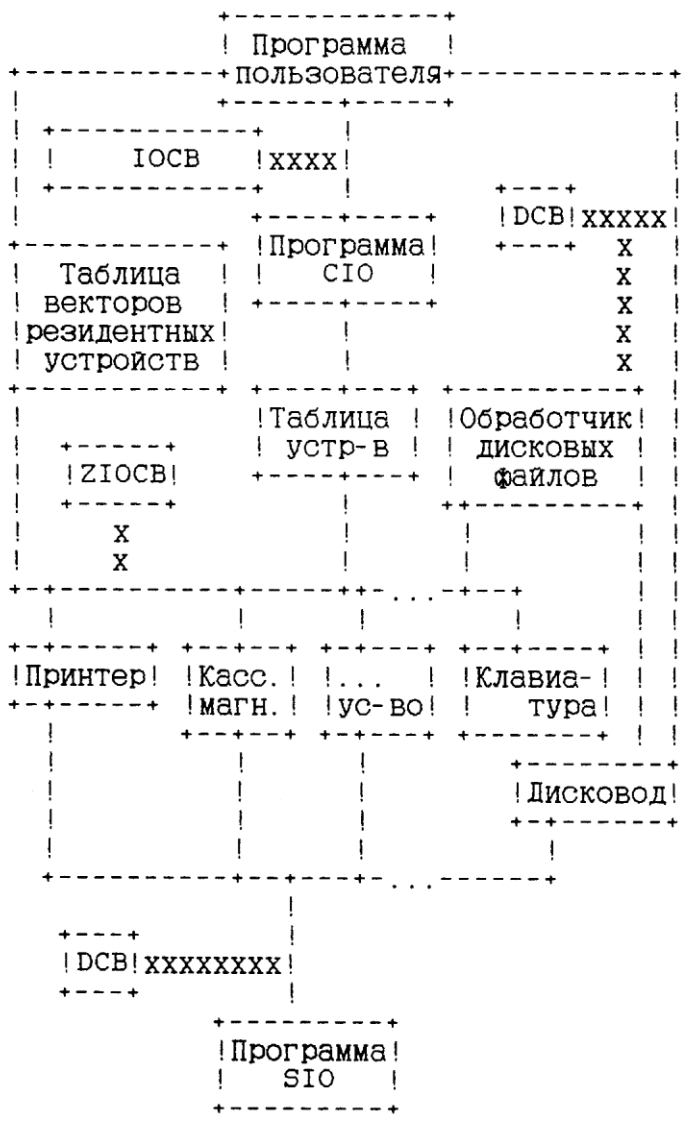


Рис. 5-1 Блок-схема подсистемы ввода/вывода.

----- = управляющие связи.
XXXXX = связи по данным.

Заметим следующее:
- Клавиатура, дисплей и экранный редактор не используют SIO.
- Дисковод доступен прямо из CIO.
- DCB (блок управления устройством) дважды указан в схеме.

Использование CIO.

Удобство CIO обеспечивается наличием единственного интерфейса, в который поступает вся информация о периферийных устройствах в виде, не зависящем от самих этих устройств. Наименьшей переносимой единицей является байт данных. CIO поддерживает обмен группой байт. В основу всех операций ввода/вывода положен принцип "возврати пользователю, когда закончишь", и не

существует способа выполнять вложенные операции ввода/вывода. Ввод/вывод организован по файлам, где файл представляет собой последовательный набор байтов данных. Файл может содержать, а может не содержать текстовые данные, а также может быть организован по записям, где запись представляет собой непрерывную группу байт, оканчивающуюся символом EOL (конец строки). Некоторые файлы отождествляются с устройствами (например с принтером или экранным редактором), в то время как остальные устройства могут содержать несколько файлов, каждый под своим именем (как в случае с дисководом). CIO позволяет осуществить доступ к восьми независимым устройствам/файлам одновременно, поэтому в системе имеется восемь управляющих блоков ввода/вывода (IOCB). Каждый из IOCB может управлять любым устройством/файлом, так как никаких предварительных присваиваний не производиться, за исключением того, что IOCB 0 закрепляется за экранным редактором при включении в сеть или при перезагрузке системы. Чтобы получить доступ к периферийному устройству сначала необходимо установить IOCB командой OPEN, указав системное имя устройства, к которому осуществляется доступ (например K: для клавиатуры, P: для принтера, D:STARS для файла на дискете с именем "STARS" и т.п.). Затем вызывается процедура CIO, выполняющая, на основе данных в IOCB команду OPEN. CIO попытается обнаружить указанное устройство/файл и возвратит байт состояния, содержащий информацию о результатах поиска. Если указанное устройство/файл обнаружено CIO, то CIO сохранит управляющую информацию в IOCB. В этом случае IOCB задействован на протяжении всего времени, пока открыт файл. После открытия файла доступ к нему может быть осуществлен при помощи команд считывания из файла и записи данных в файл. В общем случае чтение из файла может производиться до тех пор, пока в файле есть данные (до конца файла). Запись может осуществляться пока есть свободное место памяти, для данных (до конца свободного места), хотя совсем необязательно производить запись или чтение до этих моментов. Чтение и запись данных обычно производиться из буферов данных пользователя (хотя в некоторых случаях регистр A 6502 позволяет производить перемещение одного единственного байта). Когда пользователь нуждается в доступе к открытому ранее устройству/файлу необходимо закрыть файл. Данный процесс предусматривает выполнение двух действий:

- Закрытие и воспрепятствование изменению входного файла (является обязательным для файлов, подлежащих хранению на дискетах или кассетах).
- Освобождение IOCB для использования в других операциях ввода/вывода.

Философия проектирования CIO.

CIO спроектирован таким образом, что удовлетворяет следующим критериям:

- Перенос данных не должен зависеть от типа устройств.
- Должно быть возможно осуществление доступа к одному байту, к нескольким байтам и какой-либо записи.
- Несколько устройств/файлов должны быть доступны одновременно.
- Обработка ошибок не должна в большой мере зависеть от типа устройства.
- Подключение новых устройств не должно вызвать изменений в системном ПЗУ.

Независимость от устройства.

CIO обеспечивает независимость от устройства благодаря наличию единого входа для всех устройств (и для всех операций), наличию последовательного вызова, не зависящего от устройства. Перенос данных открытого устройства/файла осуществляется без предъявлений каких-либо требований к данному устройству. Правила обработки данных, переносимых по-байтно и по записям, позволяют легко узнать размеры блока действующего устройства.

Способы выборки данных.

CIO обеспечивает два способа доступа к файлам: по байтам и по записям. Доступ по байтам дает возможность рассматривать устройство/файл в виде последовательности байтов. Любое число байт можно считать из файла или записать в него, причем последующая операция будет выполнена в том месте, где закончилась предыдущая. В данном случае записи не имеют никакого значения, возможно считывание или запись нескольких записей. Доступ по записям дает возможность рассматривать поток данных на более высоком уровне (на уровне записей или строк текста). Каждая операция записи в файл создает единственную запись (по определению). Каждая операция чтения подразумевает начало последующей операции с начала записи. Такой доступ не может быть осуществлен более чем к одной записи одновременно. Доступ по записям удобен лишь при работе с текстовыми данными или с двоичными данными не содержащими символа конца строки (!9B). Заметим, что к любому файлу, независимо от способа его создания, может быть осуществлен побайтный доступ. Однако не все файлы можно успешно считать по записям, поскольку файл должен содержать символы конца строки (EOL) в конце каждой записи и нигде больше.

Одновременное использование нескольких устройств/файлов.

При помощи CIO возможно одновременно осуществить доступ к восьми устройствам/файлам, причем независимо друг от друга.

Единая обработка ошибок.

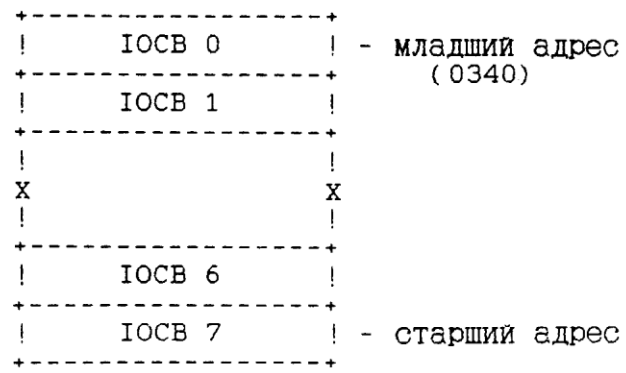
Выявление и устранение всех ошибок осуществляется системой SIO. Доступная информация представляет собой байт состояния каждого устройства/файла. Коды ошибок, насколько это возможно, не зависят от устройств (см. Приложение B).

Подключение дополнительного устройства.

Устройства обозначаются одним символом, например K или P, представляющим собой имя и номер устройства, являющихся частью резидентной системы ПЗУ. Однако возможно подсоединение дополнительных устройств к системе при помощи таблицы устройств ОЗУ. Обычно это производиться при включении питания (как в случае с дисководом), хотя это является возможным в любой момент времени.

Механизм вызова CIO.

Управляющий блок ввода/вывода является первостепенным параметром, связанным с CIO. В системе восемь IOCB расположены последовательно в ОЗУ, как показано на рисунке:



Для каждого открытого устройства/файла требуется один IOCB. За любым устройством/файлом может быть закреплен один из IOCB, хотя принято: нулевой IOCB закреплять за экранным редактором (E:). Обычная операция ввода/вывода требует выполнения следующих действий:

- Введения необходимых параметров в выбранный вами IOCB.
- Помещения номера IOCB умноженного на 16 в регистр X 6502.
- Выполнения команды JSR к входной точке CIO - CIOV (E256).

Возвращение в систему CIO происходит при завершении операции или при обнаружении ошибки. Состояние операции используется IOCB в том виде, в котором оно содержится в регистре Y 6502. Коды возврата 6502 используют содержимое регистра Y. В некоторых случаях байт данных может содержаться в регистре A 6502. Регистр X остается без изменений при любых условиях и во время всех операций. Ниже приведен пример:

```
IOCB2X = $20 ; указатель на IOCB 2
LDX #IOCB2X
JSR CIOV
CPY #0
BMI ERROR
```

В данном параграфе будет рассмотрен каждый байт IOCB со своими собственным именем и адресом. Длина каждого IOCB составляет 16 байт. Некоторые байты могут быть изменены пользователем, а некоторые являются зарезирвированными CIO и/или устройствами.

ID устройства -- ICHID (0340).

ID устройства представляет собой индекс в системной таблице устройств (см. Раздел 9), а изменение его пользователем не возможно. Значение данного байта устанавливается CIO в результате выполнения команды OPEN и остается без изменений, до закрытия устройства/файла, после чего CIO установит значение байта равным $FF.

Номер устройства -- ICDNO (0341).

Номер устройства устанавливается CIO в результате выполнения команды OPEN, и изменение его пользователем невозможно. Данный байт служит для установления различия между двумя и более устройствами одного типа, как например D1 и D2.

Командный байт -- ICCHD (0342).

Командный байт устанавливается пользователем. Его значение определяет команду, которую должен выполнить CIO. CIO не изменяет значение этого байта.

Состояние -- ICSTA (0343).

CIO сообщает пользователю состояние операции при помощи командного байта состояния в результате каждого вызова CIO. Каждый вызов CIO корректирует значение командного байта состояния. Значение старшего (знакового) бита равно единице, если произошла ошибка, и нулю, если ошибки не произошло, а в остальных битах записан код ошибки. В приложении B можно найти список кодов состояния.

Адрес буфера -- ICBAL (0344) и ICBAH (0345).

Данный двухбайтовый указатель устанавливается пользователем, и неизменяется CIO. Указатель содержит адрес начала (младший адрес) буфера, который:
- указывает на данные для операции чтения или записи,
- указывает на спецификацию устройств/файлов для команды OPEN. Можно изменить значение указателя в любое время.

Адрес PUT -- ICPTL (0346) и ICPTH (0347).

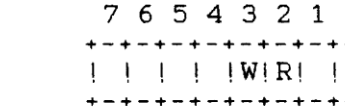
CIO устанавливает значение этого двухбайтого указателя в момент OPEN для входной точки (-1) программы PUT CHARACTER. Указатель предназначен для ATARI Бейсик. Его использование в системе не имеет большого значения. Эта адрес указывает на программу выдачи сообщени "IOCB NOT OPEN" после CLOSE, включения питания и нажатия клавиши (RESET).

Длина буфера -- ICBLL (0348) и ICBLH (0349).

Пользователь устанавливает значение этого двухбайтого счетчика для указания размера буфера данных при считывании и записи, указателем на который служат ICBAL и ICBAH. При выполнении операции OPEN данное значение не используется. После выполнения операции записи или считывания CIO устанавливает значение этого параметра равное числу реально переданных байт данных. Возможно, что условие конца файла или наличие ошибки приведут к тому, что значение счетчика будет меньше длины буфера.

Вспомогательная информация -- ICAX1 (034A) и ICAX2 (034B).

Значения данных байтов устанавливаются пользователем. В них содержиться зависимая от устройств информация, используемая командой OPEN. Два бита ICAX1 всегда используются для определения направления операции OPEN, как показано ниже, где W устанавливается равным единице при выводе (записи), а R устанавливается равным единице при вводе (чтения).



W - запись, R - чтение.

CIO не изменяет значения ICAX1. Пользователю также не следут изменять это значение после того, как было открыто устройство/файл. Оставшиеся биты ICAX1 и все биты ICAX2 содержат информацию, зависящую от устройства. Их назначение будет объяснено далее в этом же разделе.

Остальные байты -- ICAX3-ICAX6.

Управляющее устройство использует оставшиеся четыре байта для обработки команд ввода/вывода для CIO. Никакого определенного назначения эти байты не имеют. Они не могут быть изменены пользователем за исключением случаев, специально определенных в описании устройства. Ссылка на эти байты осуществляется по именам ICAX3, ICAX4, ICAX5 и ICAX6, хотя эти имена не зарезервированы в ОС.

Функции CIO.

CIO использует записи и блоки, а устройства - единичные байты. Все системные устройства способны выполнять от одной до восьми основных функций, на использование которых наложены ограничения зависящие от направления потока данных (например невозможно считать данные с принтера). Оновные функции: OPEN, CLOSE, GET CHARACTERS, PUT CHARACTERS, GET RECORD, PUT RECORD, GET STATUS и SPECIAL.

OPEN - присвоить имя устройству/файлу и приготовиться к выборке.

Устройство/файл необходимо открыть до того, как осуществить к нему доступ. В результате этого процесса осуществляется связь определенного устройства с определенным IOCB, производиться инициализация устройства/файла и всех управляющих переменных CIO, а также производиться пересылка опций устройства к самому устройству. Прежде, чем осуществить операцию OPEN CIO, необходимо установить следующие значения параметров IOCB.

COMMAND BYTE (коммандный байт) - $03.

BUFFER ADDRESS (адрес буфера) - указатель на спецификацию имени устройства/файла.

AUX1 - направляющие биты операции OPEN + информация зависящая от устройства.

AUX2 - зависящая от устройства информация.

После выполнения операции OPEN CIO, будут изменены следующие параметры IOCB:

HANDLER ID (устройство ID) - указатель на системную таблицу устройств; используется только CIO, и не должен изменяться.

DEVICE NUMBER (номер устройства) - номер устройства из спецификации имен устройств/файлов, не должен изменяться.

STATUS (состояние) - результат операции OPEN; коды различных состояний можно найти в приложении B. Обычно отрицательное значение состояния свидетельствует о том, что выполнение операции OPEN не произошло.

PUT ADDRESS (адрес операции PUT) - указатель на процедуру PUT CHARACTER (записать символ) для вновь открытого устройства. Данный указатель использовать не рекомендуется.

CLOSE -- Закрыть доступ к устройству/файлу и освободить IOCB.

Команда CLOSE используется, если доступ к данному устройству/файлу был осуществлен. Процесс, вызванный операцией CLOSE прекращает запись данных, запрашивает устройство о необходимости выполнения каких-либо действий, а затем освобождает IOCB. Перед вызовом CIO необходимо установить значение следующего параметра:

COMMAND BYTE (командный байт) - $0C.

В результате выполнении операции CLOSE CIO изменяет следующие параметры IOCB:

HANDLER ID (устройство ID) - $FF.

STATUS (состояние) - результат операции CLOSE.

PUT ADDRESS (адрес операции PUT) - указатель на процедуру "IOCB NOT OPEN".

GET CHARECTERS -- Считать N символов (побайтный доступ).

В результате применения команды определенное число символов считывается из устройства/файла в буфер пользователя. Символы конца строки не оказывают влияния на данный процесс. После завершения операции в буфере может располагаться несколько символов конца строки, а может не находится ни одного. Имеет место специальный случай, который предусматривает перемещение единичного байта в регистр A 6502, если не установлена длина буфера. Прежде,

чем вызвать CIO необходимо установить значения следующих параметров IOCB:
COMMAND BYTE (командный байт) - !07.
BUFFER ADDRESS (адрес буфера) - указатель на буфер данных.
BUFFER LENGTH - количество байт, которые нужно считать; в случае равенства нулю данные будут возвращены в регистр A 6502.

В результате выполнения команды GET CHARACTERS CIO изменяет следующие значения параметров:
STATUS (состояние) - результат операции GET CHARACTERS.
BYTE COUNT/BUFFER LEGHTH (счетчик длины буфера в байтах) - число считанных в буфер байт. Значение BYTE COUNT обычно равно значению BUFFER LENGTH, за исключением тех случаев, когда имеет место конец файла или ошибка.

PUT CHARACTERS -- Записать N символов (побайтный доступ).

В результате данной операции производится запись определенного числа символов из буфера пользователя в устройство/файл. Символы конца строки не влияют на остановку процесса, хотя при записи в устройство/файл они принимают свое стандартное значение. Символы конца строки CIO не устанавливаются. Имеет место специальный случай, предусматривающий помещение единичного символа в регистр A 6502, если длина буфера равна нулю. Прежде, чем выполнить операцию PUT CHARACTERS необходимо установить следующие параметры IOCB:
COMMAND BYTE (командный байт) - !0B.
BUFFER ADDRESS (адрес буфера) - указатель на буфер данных.
BUFFER LENGTH (длина буфера) - количество байт данных в буфере.

CIO в результате операции PUT CHARACTERS изменяет следующий параметр IOCB:
STATUS (состояние) - результат операции PUT CHARACTERS.

GET RECORD -- Считать до N символов (доступ по записям).

Символы считываются из устройства/файла в буфер пользователя до тех пор, пока не считается и не будет помещен в буфер символ конца строки. Если заполнение буфера произошло еще до считывания символа конца строки, то CIO продолжает считывание до появления символа конца строки, а затем устанавливает состояние указывающее на обрыв записи. В конец буфера символ конца строки не помещается. Прежде, чем вызвать CIO необходимо установить следующие значения параметров IOCB:
COMMAND BYTE (командный байт) - !05.
BUFFER ADDRESS (адрес буфера) - указатель на буфер данных.
BUFFER LENGTH (длина буфера) - максимальное число байт, которое нужно считать (включая символ конца строки).

CIO изменяет следующие параметры в результате операции GET RECORD:
STATUS (состояние) - результат операции GET RECORD.
BYTE COUNT/BUFFER LENGTH (счетчик длины буфера в байтах) - количество байт, считанных в буфер.

PUT RECORD -- Произвести запись до N символов (доступ по записям).

Символы записываются из буфера пользователя в устройство/файл до тех пор, пока не встретится символ конца строки, или буфер не окажется пустым. Если буфер пуст, а символ конца строки не был записан в устройство/файл, то он записывается после последнего символа буфера пользователя. Перед вызовом CIO необходимо установить следующие значения параметров IOCB:
COMMAND BYTE (командный байт) - !09.
BUFFER ADDRESS (адрес буфера) - указатель на буфер данных.
BUFFER LENGTH (длина буфера) - максимальное число байт в буфере.

CIO изменяет следующие параметры в результате операции PUT RECORD:
STATUS (состояние) - результат операции PUT RECORD.

GET STATUS -- Возвращает зависящие от устройства байты состояния.

На контроллер устройства посылается команда STATUS, в ответ на которую контроллер возвращает четыре байта, расположенные в DVSTAT (02EA) и содержащие информацию о состоянии. Перед вызовом CIO необходимо задать значения следующих параметров IOCB:
COMMAND BYTE (командный байт) - ;0D.
BUFFER ADDRESS (адрес буфера) - указатель на спецификацию имен устройств/файлов в том случае, если операция OPEN уже произведена в IOCB. Описание данной опции приведено ниже. После выполнения операции GET STATUS CIO изменяет значение следующих параметров:
STATUS - результат операции GET STATUS. Список кодов возможных состояний можно найти в приложении B.
DVSTAT - четырех байтовый отклик от контроллера устройства.

SPECIAL -- специальная функция.

Любое значение командного байта, превышающее ;0D, расценивается CIO, как специальный случай. Поскольку CIO не может, в этом случае, распознать функцию, он передает управление устройству для полной обработки операции. Перед вызовом CIO пользователем устанавливаются следующие значения параметров IOCB:
COMMAND BYTE (командный байт) - больше ;0D.
BUFFER ADDRESS (адрес буфера) - указатель на спецификацию имен устройств/файлов, в том случае если в отношении IOCB операция OPEN не была произведена. Оисание данной операции OPEN приведено ниже.
Значения остальных байтов IOCB могут быть установлены в зависимости от разновидности команды SPECIAL. После выполнения команды SPECIAL CIO изменяет значения следующих параметров:
STATUS (состояние) - результат выполнения операции SPECIAL. Список кодов, возможных состояний находиться в приложении B.

Возможные опции операции OPEN.

Команда GET STATUS и специальные команды трактуются CIO специальным образом. Для выполнения процесса они могут использовать как уже открытый, так еще неоткрытый IOCB. В случае, если IOCB не открыт, адрес буфера должен содержать указатель на спецификацию имен устройств/файлов, как для команды OPEN. В этом случае CIO откроет IOCB, выполнит заданную команду и снова закроет IOCB.

Спецификация имен устройств/файлов.

Адрес буфера IOCB указывает на спецификацию устройств/файлов, представляющую собой строку символов ATASCII следующего формата:
(спецификация)::=(устройство)/(номер устройства)/:/(имя файла)/(EOL)
(устройство)::=C:D:E:K:P:R:S
(номер устройства)::=1:2:3:4:5:6:7:8
(имя файла) содержит зависящие от устройства характеристики.
(EOL)::=;9B.
Следующие устройства принято записывать следующим образом:
C - кассетный магнитофон.
c D1 по D8 - дисководы для гибких дисков.(X)
E - экранный редактор.
K - клавиатура.
P - 40-столбцовый принтер.
P2 - 80-столбцовый принтер.(X)
c R1 по R4 - интерфейсы RS-232-C.(X)
S - экран дисплея.

Устройства, помеченные (x), имеют нерезидентное управление. По умолчанию (номер устройства) принимается равным единице. Следующие примеры показывают верную запись спецификаций имен устройств/файлов:

```
  C:       Кассетный магнитофон.
  D2:BDAT Файл "BDAT" на дисководе 2.
  D:HOLD  Файл "HOLD" на дисководе 1.
  K:       Клавиатура.
```

Пример ввода/вывода.

Пример, приведенный в данном разделе демонстрирует простой ввод/вывод с помощью процедуры CIO.

```
; Данный фрагмент программы выполняет
; считавание строк текста (записей) из
; файла TESTER, находящегося на
; дисководе 1. Все используемые символы
; определены внутри программы, хотя
; многие из них определены в файле
; присваиваний ОС.
; Программа выполняет следующие
; действия:
; -Открывает файл "D1:TESTER" с помощью
;  IOCB 3
; -Считывает записи до возникновения
;  ошибки или ситуации конца файла (EOF)
; -Закрывает файл
; Присваивание ввода/вывода.
EOL=     $9B      ;символ конца строки
IOCB3=   $30      ;начальный адрес IOCB
                  ;3 (от IOCB 0)
ICHID=   $340     ;ID устройство
                  ;устанавливается CIO
ICDNO=   ICHID+1  ;номер устройства
                  ;устанавливается CIO
ICCOM=   ICDNO+1  ;командный байт
ICSTA=   ICCOM+1  ;байт состояния
                  ;устанавливается CIO
ICBAL=   ICSTA+1  ;адрес буфера (младший
                  ;байт)
ICBAH=   ICBAL+1  ;адрес буфера (старший
                  ;байт)
ICPTL=   ICBAH+1  ;
ICPTH=   ICPTL+1  ;
ICBLL=   ICPTH+1  ;длина буфера (младший
                  ;байт)
ICBLH=   ICBLL+1  ;длина буфера (старший
                  ;байт)
ICAX1=   ICBLH+1  ;AUX 1 (вспомогат. 1)
ICAX2=   ICAX1+1  ;AUX 2 (вспомогат. 2)
OPEN=    $03      ;команда OPEN
GETREC=  $05      ;команда GET RECORD
CLOSE=   $0C      ;команда CLOSE
OREAD=   $04      ;направление OPEN -
                  ;считывание
OWRIT=   $08      ;направление OPEN -
                  ;запись
EOF=     $88      ;величина состояния
                  ;конец файла
CIOV=    $E456    ;вектор входного
                  ;адреса CIO
```

```
; Прежде проводиться инициализация IOCB
; для файла "OPEN".
     LDX #IOCB3   ;открыть доступ к
                  ;IOCB 3
     LDA #OPEN    ;установка команды
     STA ICCOM,X  ;OPEN
                  :
     LDA #NAME    ;установка указателя
     STA ICBAL,X  ;буфера на имя файла
     LDA #NAME/256
     STA ICBAH,X  :
                  :
     LDA #OREAD   ;установить чтение для
     STA ICAX1,X  ;OPEN
                  :
     LDA #0       ;очистить AUX2
     STA ICAX2,X  :
                  :
; Открыть файл.
                  :
     JSR CIOV     ;выполнить операцию
                  ;OPEN
     BPL TP10     ;если значение STATUS
                  ;положительно - хорошо
     JMP ERROR    ;нет - сообщение об
                  ;ошибке
                  :
; Установка параметров для считывания
; записи.
                  :
TR10 LDA #GETREC  ;установить команду
     STA ICCOM,X  ;GET RECORD
                  :
     LDA #BUFF    ;установить указатель
     STA ICBAL,X  ;на буфер данных
     LDA #BUFF/256
     STA ICBAH,X  :
                  :
; Считывание записей.
                  :
LOOP LDA #BUFFSZ  ;перед каждым
     STA ICBLL,X  ;считыванием
     LDA #BUFFSZ/256 ;установить макс.
     STA ICBLH,X  ;размер записи
                  :
     JSR CIOV     ;считать запись
     BMI TP20     ;возможен конец файла
                  :
; Сейчас запись находиться в буфере
; данных BUFF. Она заканчивается
; символом конца строки, а длина ее
; находиться в ICBLL и ICBLH. В данном
; примере с записью никаких действий,
; кроме чтения, не происходит.
                  :
     JMP LOOP     ;следующая запись
                  :
```

```
; Если значение STATUS после считывания
; отрицательные - проверить достижение
; конца файла.
                    ;
TP20 CPY {EOF       ;состояние EOF
     BNE ERROR      ;нет - сообщение об
                    ;ошибке
     LDA {CLOSE     ;да - закрыть файл
     STA ICCOM,X    ;
                    ;
     JSR CIOV       ;закрыть файл
     JMP ^          ;конец программы
                    ;
; Область данных программы примера.
                    ;
NAME .BYTE "D1:TESTER",EOL
BUFFSZ=  80         ;максимальная длина
                    ;записи 80 символов
                    ;(включая EOL)
BUFF=    ^          ;буфер чтения
^=       ^+BUFFSZ
     .END
```

Клавиатура (K:).

Клавиатура представляет собой считывающее устройство, поддерживающее выполнение следующих функций CIO:

- OPEN
- CLOSE
- GET CHARACTERS
- GET RECORD
- GET STATUS (не несущая информации функция).

Клавиатура вызывает следующие ошибочные состояния.

- !80 - прерывание клавишей BREAK,
- S88 - конец файла (вызывается нажатием CTRL 3).

Клавиатура является одним из резидентных устройств. Она имеет набор векторов устройства, начинающихся в ячейке E420. Клавиатура генерирует любой из 256 кодов знакогенератора ATASCII. Заметим, что некоторые клавиши не приводят к генерации знаков. Эти клавиши описаны ниже.

Клавиша инверсного изображения устанавливает значение флага дающего или не дающего возможность инверсии седьмого бита каждого байта считываемых данных. В отношении экранного редактора, инверсия не происходит.

CAPS - клавиша (CAPS) обеспечивает выполнение следующих трех функций:

(SHIFT)(CAPS) - блокировка алфавитных символов, получаемых при предварительном нажатии CAPS,

(CTRL)(CAPS) - блокировка алфавитных символов, получаемых при нажатии вместе с CTRL,

(CAPS) - разблокирование алфавитных символов.

Использование некоторых комбинаций клавиш таких, как (CTRL)+4, (CTRL)+9, (CTRL)+0, (CTRL)+1, а также всех комбинаций, в которых происходит одновременное нажатие клавиши (SHIFT) и (CTRL) является недопустимым. Нажатие клавиши (CTRL)+3 генерирует символ конца строки (EOL) и возвращает состояние конца файла (EOF). Нажатие клавиши (BREAK) генерирует символ конца строки и возвращает состояние (BREAK).

Описание функций CIO.

Ниже приведены специфические для каждого устройства характеристики функций CIO (описанных ранее в данном разделе).

OPEN

Имя устройства - K: При задании номера устройства и спецификации имени, эти параметры игнорируются обрабатывающим устройством. В AUX1 и AUX2 отсутствуют зависящие от устройства биты.

CLOSE

Специальной информации нет.

GET CHARACTERS и GET RECORD

Устройство возвращает в CIO коды ATASCII без редактировании по мере их вывода.

GET STATUS

Устройство не производит никаких действий за исключением установки состояния $01.

Теория операции.

Нажатие клавиши на клавиатуре вызывает прерывание по запросу и передает управление сервисной программе обработки прерываний клавиатуры (см. Раздел 6). Код клавиши считывается и располагается в переменной CH (02FC). Это происходит независимо от запроса на чтение с клавиатуры и воздействует на однобайтовый FIFO ввода с клавиатуры. В приложении L можно найти описание автоматического повтора. Клавиатура управляет значением переменной CH таким образом, чтобы та не содержала значения $FF (пустое состояние) при получении запроса на чтение с устройства. Когда значение CH отлично от $FF устройство получает код клавиши из CH и снова устанавливает значение CH, равное $FF. Байт кода клавиши, полученный из CH имеет следующую форму, отличную от кодов ATASCII:

```
  7             0
 +-+-+-+-+-+-+-+-+
 !C!S!код клавиши!
 +-+-+-+-+-+-+-+-+
```

Где C=1 в случае нажатия клавиши CTRL.
S=1 в случае нажатия клавиши SHIFT.

Оставшиеся шесть бит представляет собой аппаратный код клавиши. Полученный код вспоследствии преобразуется в ATASCII, учитывая следующие правила:

1. Код игнорируется в случае, если биты C и S равны 1.
2. Если бит C установлен (равен 1),то клавиша обрабатывается как (CTRL) код.
3. Если бит S установлен, то клавиша обрабатывается как (SHIFT) код.
4. Если действует блокировка (CTRL), то буквенные символы обрабатываются как коды (CTRL), а все остальные как символы нижнего регистра.
5. Если действует блокировка (SHIFT), то буквенные символы обрабатывается как коды (SHIFT), а все остальные как символы нижнего регистра.

Если результирующий код не совпадает с кодом управления экранным редактором, и если установлен флаг инверсии изображения, устанавливается бит 7 кода ATASCII (вызывающий инверсию изображения).

Таблица преобразования кодов клавиш в коды ATASCII.

Код клавиши	Изобр. клавиши	Нижний регистр	SHIFT	CTRL
00	L	6C	4C	0C
01	J	6A	4A	0A
02	;	3B	3A	7B
03	--	--	--	--
04	--	--	--	--
05	K	6B	4B	0B
06	+	2B	5C	1E
07	*	2A	5E	1F
08	O	6F	4F	0F
09	--	--	--	--
0A	P	70	50	10
0B	U	75	55	15
0C	RETURN	9B	9B	9B
0D	I	69	49	09
0E	-	2D	5F	1C
0F	=	3D	7C	1D
10	V	76	56	16
11	--	--	--	--
12	C	63	43	03
13	--	--	--	--
14	--	--	--	--
15	B	62	42	02
16	X	78	58	18
17	Z	7A	5A	1A
18	4	34	24	--
19	--	--	--	--
1A	3	33	23	9B (X)
1B	6	36	26	--
1C	ESC	1B	1B	1B
1D	5	35	25	--
1E	2	32	22	FD
1F	1	31	21	--
20	,	2C	5B	00
21	ПРОБЕЛ	20	20	20
22	.	2E	5D	60
23	N	6E	4E	0E
24	--	--	--	--
25	M	6D	4D	0D
26	/	2F	3F	--
27	ИНВ.	--	--	--
28	R	72	52	12
29	--	--	--	--
2A	E	65	45	05
2B	Y	79	59	19
2C	TAB	7F	9F	9E
2D	T	74	54	14
2E	W	77	57	17
2F	Q	71	51	11
30	9	39	28	--
31	--	--	--	--
32	0	30	29	--
33	7	37	27	--
34	BACKS	7E	9C	FE

35	8	38	40	--
36	(	3C	7D	7D
37	)	3E	9D	FF
38	F	66	46	06
39	H	68	48	08
3A	D	64	44	04
3B	--	--	--	--
3C	CAPS	--	--	--
3D	G	67	47	07
3E	S	73	53	13
3F	A	61	41	01

(X) (CTRL) 3 вызывает ситуацию EOF (конец файла).
Дополнение к этой таблице (преобразование кода ATASCII в код клавиши) приведено в приложении F.

Дисплей (S:).

Дисплей передставляет собой считывающе-записывающее устройство с устройством управления, поддерживающим выполнение следующих функций CIO.
OPEN
CLOSE
GET CHARACTERS
GET RECORD
PUT CHARACTERS
PUT RECORD
GET STATUS (не несущая информации функция)
DRAW (рисовать)
FILL (заполнять)
Дисплей может вызывать следующие ошибочные состояния:
!84 -- неправильное использование специальной команды.
!8D -- Выход курсора из допустимого диапозона.
!91 -- Экранный режим больше 11.
!93 -- Для выбранного экранного режима недостаточно памяти.
Дисплей является одним из резидентных устройств и поэтому имеет набор векторов устройства, начинающихся в ячейке E410.

Экранные режимы.

Экран дисплея может работать в любом из двадцати состояний (в режимах с 1 по 8 с разбиением экрана на подобласти или без него и в режимах: нулевом и с 9 по 11 без разбиения экрана на подобласти). Нулевой режим предназначен для вывода текстовой информации. Режимы с 1 по 11 предназначены для вывода графической информации (хотя в режимах 2 и 3 возможно изображение некоторого подмножества символов знагогенератора ATASCII).

Нулевой текстовой режим.

В данном режиме экран разбивается на 24 строки по 40 символов в каждой. Левая и правая границы изображения могут быть программно изменены. По умолчанию они принимаются равными от 2 до 39 (при возможных от 0 до 39). Программно управляемый курсор указывает местоположение на экране следующего выводимого символа. На экране курсор представляется в виде инверсного изображения на данной позиции выводимого в данный момент символа. Внутренняя организация экранных текстовых данных представляет собой логические строки переменной длины. Для пустого экрана их число составляет 24. Каждый символ конца строки при отсылке данных на экран служит ограничением логической

строки. Если отсылается более трех физических строк представляющих одну логическую строку (от 1 до 3), всегда является наименьшим для хранения данных этой строки. Текст "поднимается" вверх всякий раз, когда нижняя строка переходит за правую на экране границу или в случае, когда в последней на экране строке появляется символ конца строки. Такое перемещение экрана вызывает удаление самой верхней внутренней логической строки, и последующее перемещение оставшихся логических строк вверх для заполнения образовавшийся пустоты. В случае, если уничтоженная логическая строка превышает физическую, то курсор перемещается наверх. Все данные посылаемые или получаемые с экрана в текстовом режиме представимы в восьмибитовом коде ATASCII, как показано в приложении E.

Текстовые режимы 1 и 2.

В текстовых режимах 1 и 2 экран состоит из 24 строк по 20 символов в каждой (первый режим) или на 12 строк по 20 символов в каждой (второй режим). В данных режимах левая и правая границы не играют никакой роли, а курсор не изображается. Не осуществляется связи экранных данных с логическими строками, и во всех остальных отношениях эти режимы расцениваться управляющим устройством, как графические. Данные отсылаемые на экран или получаемые с экрана имеют следующую форму:

```
  7             0
 +-+-+-+-+-+-+-+-+
 !  C  !    D    !
 +-+-+-+-+-+-+-+-+
```

Где C - поле выбора цвета символа.

Величина C	Цвет по умолч.	Цвет. регис. (прил. H)	Знако-генер. CHBAS= SE0	Знако-генер. CHBAS= SE2
0	зеленый	(PF1)	!-?	(H) (A)
1	золотой	(PF0)	!-?	(H) (A)
2	золотой	(PF0)	~-л	(D) (T)
3	зеленый	(PF1)	~-л	(D) (T)
4	красный	(PF3)	!-?	(H) (A)
5	голубой	(PF2)	!-?	(H) (A)
6	голубой	(PF2)	~-л	(D) (T)
7	красный	(PF3)	~-л	(D) (T)

Где л - подчеркивающая линия.
(H)- (HEART) - черви.
(A)- (ARROW) - стрелка.
(D)- (DIAMOND) - бубны.
(T)- (TRIANGLE) - треугольник.

D представляет собой пятибайтовый обрезанный код ATASCII, выбирающий непосредственно символ в знакогенераторе, выбранном C. Графическое представление символов можно найти в приложении E. Переменная базы данных CHBAS (02F4) дает возможность выбора одного из двух возможных наборов данных. Значение !E0, принимаемое по умолчанию, дает возможность изображать заглавные буквы, числа и обеспечивает изображение букв нижнего регистра и символов специального графического знакогенератора.

Графические режимы (с 3 по 11).

Как показано в приложении Н физические характеристики экрана изменяются в зависимости от выбранного режима графики. В зависимости от режима возможен выбор одного из 16 цветов для каждого пикселя. Также при изменении режима графики изменяется размер экрана в пикселях с 20x12 для наименьшей разрешающей способности. Для вывода графических данных курсор не изображается. Данные, поступающие на графический экран или с него представляются в виде от однобитового до восьмибитного кодов, как показано в приложении Н и на ниже приведенных диаграммах GET/PUT.

Конфигурация разбиения экрана на подобласти.

При разбиении экрана на подобласти нижняя его часть отводиться под четыре строки текста в нулевом режиме. Область текста управляется экранным редактором, а графическая область - управляющим устройством дисплея. Данная конфигурация подразумевает наличие двух курсоров, поэтому управление областями осуществляется независимо. Для работы в режиме разбиения экрана на подобласти необходимо сначала открыть экранный редактор, а затем с помощью другого IOCB (с установленным в AUX1 битом, указывающим на разбиение экрана) открыть дисплей.

Описание функций CIO.

Ниже приведены характеристики стандартных функций CIO (описанных ранее), зависящие от устройства.

OPEN

Имя устройства S. Параметры имени файла и номера устройства игнорируются. Устройство допускает установку следующих опций:

```
       7             0
      +-+-+-+-+-+-+-+-+
AUX1  !   !C!S!W!R!   !
      +-+-+-+-+-+-+-+-+
```

При C=1 не происходит стирание данных с экрана при использовании OPEN.
S=1 разбивает экран на подобласти только для режимов с 1 по 8.
R и W - биты направления (читать, писать).

```
       7             0
      +-+-+-+-+-+-+-+-+
AUX2  !       ! режим !
      +-+-+-+-+-+-+-+-+
```

Где, режим - экранный режим (с 0 по 11).

Примечание: В случае, когда выбран нулевой режим, опции C и S приняты равными 0. Вы используете память совместно с информацией для дисплея. Это является необходимым, так как дисплей отводит высокоадресную память под изображение на экране, а различные экранные режимы используют разный обьем памяти. Прежде, чем вызвать команду OPEN, необходимо присвоить переменной APPMHI (000E) значение наивысшего необходимого адреса ОЗУ. Доступ к экрану будет осуществлен лишь в том случае, если по этому адресу и ниже не зарезвирована память ОЗУ. В зависимости от ответа экрана на команду OPEN переменная MEMTOP (02E5) будет содержать адрес последнего свободного байта ОЗУ, предшествующего области экранной памяти. В результате каждого использования команды OPEN будут изменены значения следующих переменных.

Появляется курсор (CRSINH=0). Устанавливаются принимаемые по умолчанию значения границ табуляции (2 и 39). Значениям цветовых регистров присваиваются значения, принимаемые по умолчанию (см. Приложение H). Табуляция устанавливается в позициях:
7,15,23,31,39,47,55,63,71,79,87,95,103,111,119.

CLOSE

Не предусматривает выполнения устройством каких-либо специальных действий.

GET CHARACTERS и GET RECORD

Данные возвращаются пользователю в нижеприведенных формах, различных для каждого экранного режима. Каждый байт содержит данные для одной позиции курсора (пикселя), а получить от устройства графические упакованные данные не представляется возможным.

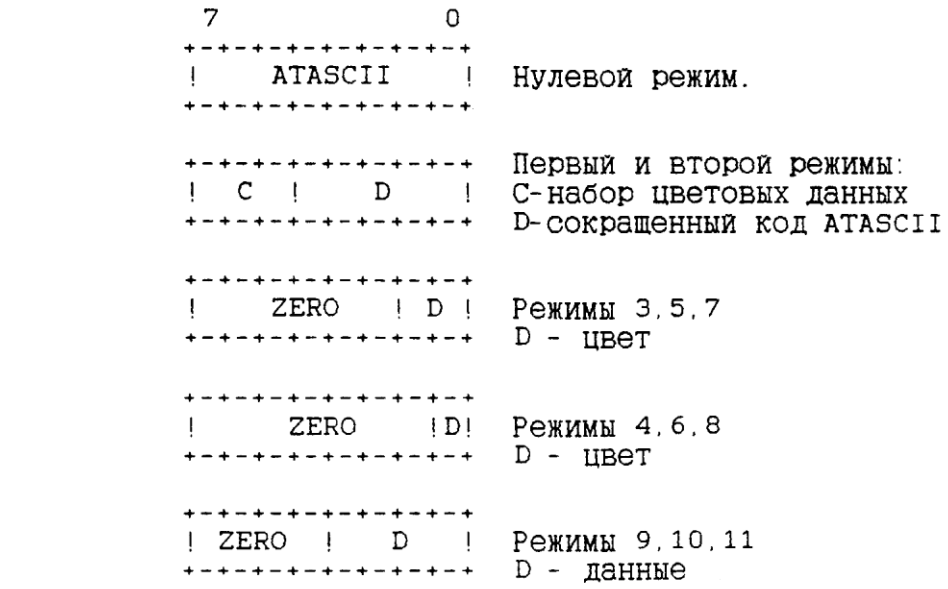


Рис.5-6 Форма данных для команды GET в графических режимах с 3 по 11.

Перевод курсора на следующую позицию осуществляется всякий раз при получении байта данных. В нулевом режиме курсор располагается внутри определенного диапозона. Во всех остальных режимах границы игнорируются курсором.

PUT CHARACTERS и PUT RECORD

Устройство принимает изображенные данные в нижеприведенных формах, зависящих от выбранного экранного режима. Устройство не может получать упакованные данные.

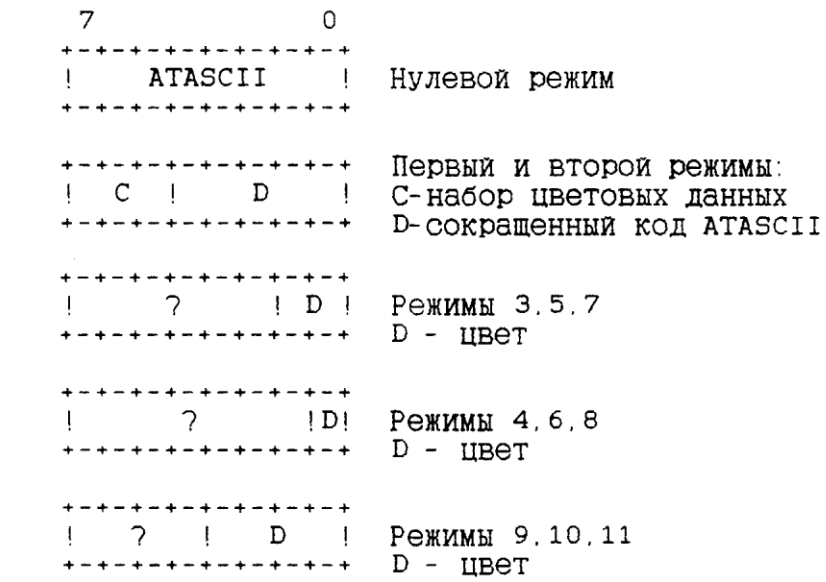


Рис.5-7 Форма данных для команды PUT в графических режимах с 3 по 11.

Замечание: Во всех режимах в случае равенства значения выходного байта величине #9B (конец строки) данный байт будет индентифицирован, как символ конца строки, а если величина байта совпадает с #7D (очистить), то байт будет распознан, как символ, очищающий экран. После каждой записи символа на экране осуществляется перевод курсора на следующую позицию. Для нулевого режима курсор должен помещаться внутри заданного диапозона, а для остальных режимов границы диапозонов игнорируется курсором. В процессе вывода дисплей поддерживает связь с клавиатурой на предмет обнаружения комбинации клавиш (CTRL)(1). При нажатии этой комбинации происходит внутреннее зацикливание устройства, продолжающееся до следующего использования этой комбинации: следствием этого является функция стоп/старт, делающая изображение неподвижным. Заметим, что комбинации (CTR)(1) и функции стоп/старт не соответствует ни один из кодов ATASCII. Управление функцией стоп/старт может осуществляться только с клавиатуры (или изменением значения переменной CH (см. Приложение L, E4).

GET STATUS

Не предусматривает выполнения устройством каких-либо действий за исключением установки состояния #01.

DRAW

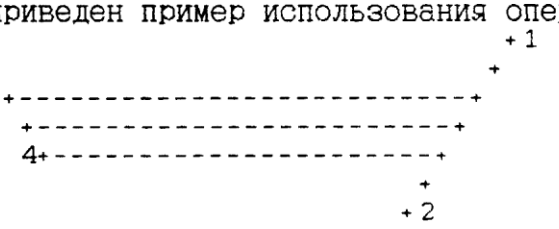
При помощи данной команды можно изобразить на экране "прямую линию", начинающуюся с текущей позиции курсора, а заканчивающуюся в точке, положение которой определяется значением ROWCRS (0054) и COLCRS (0055). Цвет линии выбирается таким, как у последнего символа, обработанного дисплеем и экранным редактором. После выполнения команды местоположение курсора будет определяться значениями ROWCRS и COLCRS. Величина командного байта для DRAW равна #11.

FILL

При помощи данной команды можно закрасить определенным цветом область, задаваемую двумя прямыми. Команда запускается также, как DRAW, но по мере изображения каждой точки прямой программа переходит вправо, выполняя приведенную ниже процедуру (записанную на ПАСКАЛЕ).

```
WHILE PIXEL(ROW,COL)=0 DO
  BEGIN
    PIXEL(ROW,COL):=FILDAT;
    COL:=COL+1;
    IF COL) SCREEN RIGHT EDGE THEN COL:=0
  END;
```

Ниже приведен пример использования операции FILL:



Где: "-" - представление операции FILL.
"+" - конечные точки прямых.
"1" - устанавливает курсор и точку рисунка.
"2" - устанавливает курсор и линию DRAW.
"3" - устанавливает курсор и точку.
"4" - устанавливает размер заполняющих данных, устанавливает курсор и FILL.
В FILDAT (02FD) хранятся заполняемые данные, а в ROWCRS и COLCRS хранятся координаты курсора для конечных точек линии. Величина, хранящаяся в ATACHR (02FB) используется для изображения линии. ATACHR всегда содержит последний считанный или записанный элемент данных, поэтому, если предыдущие шаги были выполнены правильно, то значение в ATACHR не должно изменяться. Величина командного байта FILL составляет ;12.

Переменные изменяемые пользователем.

Некоторые функции дисплея требуют от пользователя определения значений переменных ОС и/или изменение этих значений. Ниже описываются наиболее часто используемые устройством переменные. (Дополнительную информацию можно найти, обратившись к приложению L, B1-55.)

Местонахождение курсора.

В нулевом текстовом режиме, как и в графическом, местоположение курсора опреляют значение двух переменных. ROWCRS (0054) определяет номер строки, а COLCRS (0055) - номер стобца, в котором расположен курсор. Данные номера лежат в диапозоне от нуля до максимального количества строк/стобцов,-1. Курсор может быть установлен за границами текста. Можно осуществлять чтение и запись из этой области. Курсор всегда возвращается в позицию (0,0) (левый крайний угол), как в графических, так и в текстовых режимах. ROWCRS представляет собой единичный байт. Под COLCRS отведено два байта и младший из них имеет низший адрес. После изменения значений этих переменных изображение курсора не сдвигается до момента выполнения следующей по отношению к дисплею операции ввода/вывода.

Возможность и невозможность изображения курсора.

Можно убрать курсор с экрана, установив любое ненулевое значение переменной CRSINH (02F0). Последующий ввод/вывод будет осуществляться без курсора.

Границы текста.

Пользователь может изменить левую и правую границы текста на экране. Операционная система присваивает им значения 2 и 39. Левая граница определяется значением переменной LMARGN (0052), а правая - RMARGN (0053). Наименьшее значение левой границы 0, а наибольшее значение правой - 39. Установленными границами определяется, включительно та часть экрана, на которой выполнение всех операций не изменяет значения переменных положения курсора, описанных ранее в этом параграфе.

Управление цветом.

ОС производит корректировку значения аппаратных цветовых регистров, используя данные из базы данных ОС, как часть нормальной обработки второй стадии VBLANK (см. Раздел 6). Ниже приведены имена переменных, имена аппаратных регистров и функции, выполняемые каждым регистром. Назначение регистров в зависимости от режима можно найти в приложении H.

Перемен.	Аппарат. регистры	Функция
COLOR0	COLPF0	PF0-игровое поле 0
COLOR1	COLPF1	PF1-игровое поле 1
COLOR2	COLPF2	PF2-игровое поле 2
COLOR3	COLPF3	PF3-игровое поле 3
COLOR4	COLBK	BAK-фоновое игр. поле
PCOLR0	COLPM0	PM0-игрок/снаряд 0
PCOLR1	COLPM1	PM1-игрок/снаряд 1
PCOLR2	COLPM2	PM2-игрок/снаряд 2
PCOLR3	COLPM3	PM3-игрок/снаряд 3

Теория операции.

Дисплей в момент выполнения операции OPEN автоматически устанавливает все ресурсы памяти, необходимые для создания и сохранения изображения на экране. Аппаратное обеспечение, генерирующее изображение на экране в графическом режиме требует наличия двух переменных областей памяти:
1) дисплейной программы,
2) области экранной памяти.

Третья область данных необходима для текстовых режимов. В данной области записано экранное представление каждого символа текста. Для полного понимания следующего материала необходимо ознакомиться с руководством к аппаратному обеспечению домашнего компьютера ATARI (ATARI HOME COMPUTER HARDWARE MANUAL). Данная упрощенная блок-схема иллюстрирует взаимодействие памяти и аппаратных регистров, используемых для генерации изображения на экране (без учета графики игрок/снаряд). Заметим, что аппаратные регистры используются для многих других целей.

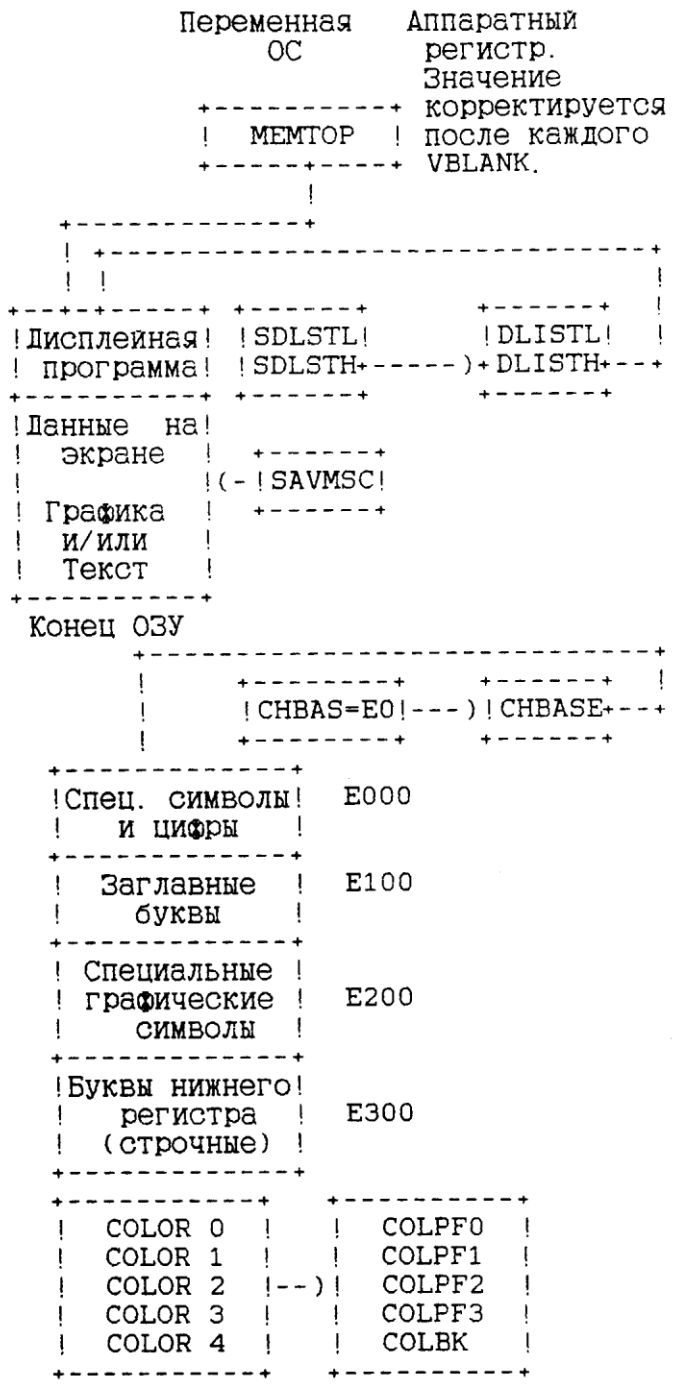


Рис. 5-8 Блок-схема экранного дисплея.

На приведенной диаграмме иллюстрированы следующие взаимоотношения:

1. Переменные SDLSTL/SDLSTH содержат адрес текущей дисплейной программы. Данный адрес переносится в аппаратные регистры адреса дисплейной программы DLISTL и DLISTH как часть процесса VBLANK.
2. Дисплейная программа определяет характеристики экранного изображения и указывает на область памяти, содержащую данные, которые будут изображены.
3. Переменная CHBAS содержит старший байт базового адреса представления символа для символьных данных (только в текстовых режимах).

По умолчанию значение этой величины принимается равным !E0. Этим объясняется то, что представление символов в памяти начинается с адреса E000 (знакогенератор ОС в ПЗУ). Каждый символ представляется в виде матрицы, размером 8x8 бит и занимает 8 байт памяти. Самый большой набор требует 1024 байт, поскольку код символа содержит до 7 старших битов (набор из 128 символов). Знакогенератор, принимаемый по умолчанию, находится в памяти от E000 до E3FF. Коды всех символов преобразуются из представления ATASCII во внутренние коды, как показано ниже.

ATASCII коды	Внутренние коды
00 - 1F	40 - 5F
20 - 3F	00 - 1F
40 - 5F	20 - 3F
60 - 7F	60 - 7F
80 - 9F	C0 - DF
A0 - BF	80 - 9F
C0 - DF	A0 - BF
E0 - FF	E0 - FF

Знакогенератор ПЗУ упорядочен в соответствии с внутренним кодом. Три тезиса побудили различать внутренние и внешние (ATASCII) коды: ATASCII коды во всех отношениях, кроме специальных графических символов близки к ASCII. Коды буквенных, цифровых и пунктационных символов совпадают с ASCII. В первом и во втором текстовых режимах желательно иметь одно подмножество символов, включающее в себя заглавнае буквы, цифры и знаки пунктуации, а также подмножество, включающее в себя строчные буквы и специальные графические символы. В первом и втором текстовых режимах заглавные и строчные буквы должны иметь одинаковые коды. Переменные с COLOR0 до COLOR4 содержат текущие значения цветовых регистров. Аппаратные цветовые регистры получают данные значения в процессе первой стадии VBLANK, обеспечивая тем самым синхронизированное изменение цвета (см. Приложение H). Переменная SAVMSC указывает на низший адрес области экранной памяти. Он соответствует данным, изображенным в левом верхнем углу дисплея. Когда на дисплей поступает команда OPEN, первым делом определяется экранный режим из открытого IOCB. Затем, начиная с конца ОЗУ и далее вниз (как определяет переменная RAMTOP), отводится память сначала для экранных данных, а потом для дисплейной программы. Если имеется достаточно памяти, то область данных очищается, а дисплейная программа создается.

Экранный редактор (E:).

Экранный редактор представляет собой считывающе-записывающее устройство, использующе дисплей и клавиатуру для построчного ввода с использованием интерактивных редактирующих функций и форматированного вывода. Экранный редактор поддерживает выполнения следующих функций CIO:

- OPEN
- CLOSE
- GET CHARACTERS
- GET RECORD
- PUT CHARACTERS
- PUT RECORD
- GET STATUS (не несущая информации функция)

Ошибочные состояния экранного редактора приведены в разделах, посвященных описанию клавиатуры и дисплея. Экранный редактор является одним из резидентных устройств и имеет набор векторов устройства, начинающихся в ячейке E400. Экранный редактор - это программа, считывающая данные с клавиатуры и отображающая каждый символ на дисплее. Экранный редактор может принимать от пользователя данные и посылать их на дисплей, а также считывать

данные с дисплея. Экранный редактор, дисплей и клавиатуру можно рассматривать как одно целое. Большая часть указанных особенностей дисплея и клавиатуры присуща экранному редактору. В данном разделе ограничимся только лишь рассмотрением отклонений от этих особенностей. Также будут рассмотрены свойства, присущие исключительно экранному редактору. Экранный редактор взаимодействует только с текстовыми данными (нулевой экранный режим). В данном разделе разъясняются некоторые особенности разбиения экрана на подобласти. Экранный редактор использует дисплей для считывания графических и текстовых экранных данных. Экранный редактор используется для того, чтобы определить, когда программ будет считывать экранные данные и с какой части экрана будет произведено считывание. Сначала ограничивается перемещение курсора по экрану, чтобы определить размер области экрана, с которой будет производиться считывание. Затем нажимается клавиша (RETURN), чтобы начать считывание указанных данных. После нажатия клавиши (RETURN) внутренняя логическая строка, внутри которой расположен курсор, становится доступной вызывающей программе. Однако конечные пробелы никогда не возвращаются в виде данных. После того, как была осуществлена пересылка всех данных строки в место вызова (при желании можно воспользоваться несколькими функциями READ CHARACTERS), возвращается символ конца строки, а курсор помещается в начало строки, следующей за только что считанной.

Описание функций CIO.

Ниже приведены зависящие от устройства характеристики стандартных функций CIO.

OPEN

Имя устройства - E. При задании номера устройства и имени файла они игнорируются. Экранный редактор допускает следующий выбор:

```
      7             0
     +-+-+-+-+-+-+-+-+
AUX1 !       !W!R! !F!
     +-+-+-+-+-+-+-+-+
```

Где R и W биты направления (считывать, записывать). F=1 указывает на желание произвести "форсированное чтение" (подробнее см. GET CHARACTERS и GET RECORD).

CLOSE

Не предусматривает выполнения устройством каких-либо специальных действий.

GET CHARACTERS и GET RECORD

Обычно экранный редактор возвращает данные только при нажатии на клавиатуре клавиши (RETURN). Однако опция OPEN "форсированное чтение" позволяет осуществлять считывание текстовых данных без вмешательства в этот процесс. Когда посылается команда READ экранный редактор возвращает данные с начала той логической строки, на которой расположен курсор, переводя затем его на начало следующей логической строки. После считывания последней строки, текст на экране "приподнимается". Имеет место особый случай, когда происходит вывод символов, не ограниченных символом конца строки, вследствии чего к данной логической строке присоединяются с клавиатуры дополнительные символы. После нажатия клавиши (RETURN) на место пересылаются только введенные с клавиатуры символы. Это происходит в том случае, если не осуществляется перевод курсора из логической строки с последующим возвращением в эту же логическую строку. В этом случае осуществляется пересылка всей логической строки.

PUT CHARACTERS и PUT RECORD

При получении данных на каждый байт приходится один символ ATASCII. 16 из 256 символов ATASCII представляют собой управляющие коды. Код символа конца строки (EOL) имеет универсальное значение. Большинство остальных управляющих кодов имеют специальное значение лишь для дисплея или принтера. Ниже объясняется механизм обработки экранным редактором управляющих кодов:

CLEAR (↑7D) - Экранный редактор стирает изображение всех данных на экране и отправляет курсор в левый верхний угол экрана.

CURSOR UP (↑1C) - Курсор перемещается вверх на одну физическую строку. При переходе через верхнюю строку курсор оказывается на нижней строке экрана.

CURSOR DOWN (↑1D) - Курсор перемещается вниз на одну физическую строку. При переходе через нижнюю строку курсор оказывается на верхней строке экрана.

CURSOR LEFT (↑1E) - Курсор перемещается на одну позицию влево. При переходе через левую границу экрана курсор устанавливается на позицию той же строки у правой границы экрана.

CURSOR RIGHT (↑1F) - Курсор перемещается на одну позицию вправо. При переходе через правую границу экрана курсор устанавливается на позицию той же строки у левой границы экрана.

BACKSPACE (↑7E) - Курсор перемещается на одну позицию влево (но не через начало логической строки), изменяя символ в этой позиции на пробел (↑20).

SET TAB (↑9F) - Экранный редактор устанавливает точку табуляции для курсора логической строки. Местоположение табуляции в логической и физической строках различны, поскольку длина логической строки может в три раза превышать длину физической строки. Например, табуляция может быть установлена в 15,30,45,6 и 75 позициях логической строки, как показано ниже.

```
0 2      9        19        29        39 Номер экранного столбца.
--L------+---------+---------+---------R Левая и правая границы экрана.
XX--------------T--------------T-------- Логическая строка.
XX------T---------------T-------------T- X - недоступные столбцы.
XX-------------------------------------
```

Необходимо иметь ввиду воздействие левой границы на определение границы логической строки. Ниже показаны значения табуляции, устанавливаемые устройством по умолчанию.

```
0 2      9        19        29        39 Номер экранного столбца.
--L------+---------+---------+---------R Левая и правая границы экрана.
XXT----T-------T-------T-------T-------T Логическая строка.
XX-----T-------T-------T-------T-------T X - недоступные столбцы.
XX-----T-------T-------T-------T-------T
```

CLEAR TAB (↑9B) - Экранный редактор снимает с позиции курсора метку табуляции.

TAB (↑7F) - Курсор устанавливается на следующую позицию табуляции текущей логической строки или в начало следующей строки, если в данной строке не обнаружено границ табуляции. Данная функция не увеличивает длины строки для включения в нее точек табуляции, вышедших из диапозона ее длины (например длина логической строки 38 символов и имеется точка табуляции в 50 позиции).

INSERT LINE (↑9D) - Все физические строки, находящиеся ниже строки, содержащей курсор смещаются вниз на одну позицию. В результате может получиться, что последняя на экране строка "уйдет" с экрана. Образованная пустая физическая строка становится началом новой логической строки. Логическая строка может быть в результате этого разделена на две логические строки. Вторая из них скрепляется с пустой физической строкой.

DELETE LINE (↑9C) - логическая строка, содержащая курсор, стирается, и все данные, следующие за ней сдвигаются вверх для заполнения промежутка. В нижней части дисплея при этом формируются пустые строки.
INSERT CHARACTER (↑FF) - Все физические символы, находящиеся в позиции курсора и далее сдвигаются на одну позицию вправо. Символу, находящемуся в позиции курсора, присваивается значение пробела. При вставке символа в полную строку последний символ этой строки будет утерян. В результате этой функции может возрасти число физических строк по сравнению с логическими.
DELETE CHARACTER (↑FE) - Символ, находящийся в позиции стирается, а оставшаяся часть логической строки (находящаяся справа от курсора) сдвигается на одну позицию влево. В результате этой функции число физических строк по сравнению с логическими может уменьшиться.
ESCAPE (↑13) - Следующий за данным кодом символ будет изображен, если даже он является управляющим кодом. Последовательность (ESC)(ESC) приводит к изображению на экране символа (ESC).
BELL (↑FD) - Генерация звука. Изображение не изменяется.
END OF LINE (↑9B) - В дополнение к функции ограничения символ конца строки вызывает установку курсора на начало следующей логической строки. Когда курсор достигает нижней строки на экране, появление символа конца строки вызывает передвижение экранных данных вверх.

GET STATUS

Устройство не выполняет никаких действий, за исключением установки состояния ↑01.

Изменяемые пользователем переменные.

Местоположение курсора.

В многооконном режиме переменные ROWCRS и COLCRS связаны с графической частью экрана, а две другие переменные TXTROW (0290) и TXTCOL (0291) связаны с текстовым окном. TXTROW задается единственным байтом, а TXTCOL двумя байтами, младший из которых имеет низший адрес. Заметим, что значение старшего байта всегда должно быть равно нулю. "Домом" текстового экрана служит верхний левый угол экрана.

Включение/исключение управляющих кодов в текст.

Обычно все управляющие коды текстового режима обрабатываются по мере получения, но иногда бывает нужно включить эти коды в текст как обычные символы. Это можно сделать, присвоив переменной DSPFLG (02FE) ненулевое значение перед выводом данных, содержащих управляющие символы. Установка нулевого значения DSPFLG возвращает обычный режим обработки управляющих кодов.

Кассетный магнитофон (C:).

Кассетный магнитофон может быть как считывающим устройством, так и записывающим, поддерживающим выполнение следующих функций CIO.
OPEN
CLOSE
GET CHARACTERS
GET RECORD
PUT CHARACTERS
PUT RECORD
GET STATUS (не несущая информации функция)

Кассетный магнитофон может вызвать следующие ошибочные состояния:
 !80 - Прерывание от клавиши (BREAK).
 !84 - Неправильно определен байт AUX1 в операции OPEN.
 !88 - Конец файла.
 !8A-90 - Ошибки SIO (последовательного ввода/вывода)(см. Приложение C).
Кассетный магнитофон является одним из резидентных устройств, имеющих набор векторов устройства, начинающихся в ячейке E440.

Описание функций CIO.

Ниже приведены зависящие от устройства характеристики стандартных функций CIO.

OPEN

Имя устройства C. При задании номера устройства и имени файла они игнорируются. Устройство допускает следующий выбор:

```
        7               0
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
AUX2  !C!               !
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

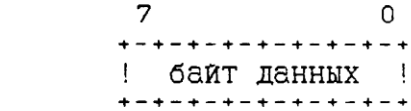
Где C=1 указывает, что считывание или запись производиться без остановки между записями (непрерывный режим). Когда магнитофон открыт для ввода, он воспроизводит характерный однотоновый звук, служащий сообщеним пользователю, что магнитофон установлен для чтения (включен в сеть; подключен к последовательной шине, лента отмотона на начало файла и нажата клавиша (PLAY)). Когда магнитофон готов, можно начать считывание с ленты, нажав любую (кроме (BREAK)), клавишу на клавиатуре. Если магнитофон готов для вывода, он воспроизводит характерный звук, состоящий из двух близких тонов, служащий сообщением, что магнитофон установлен для записи (то же что и для чтения, плюс нажата клавиша (RECORD) на магнитофоне). Когда магнитофон готов, можно начать запись на ленту, нажав любую, кроме (BREAK) клавишу, на клавиатуре. Компьютер не может сообщить о нажатии клавиши (RECORD) или (PLAY). Поэтому может получиться так, что файл не будет записан и об этом не будет сразу сообщено. Потенциальная проблема, связанная с кассетным магнитофоном, заключается в том, что после открытия магнитофона на запись мотор будет продолжать работать, пока не будет записана первая запись (128 байт). Проблем не возникает, если 128 байт были считаны, или магнитофон был закрыт в течении тридцати секунд после операции OPEN, и никаких других операций ввода/вывода последовательной шины не совершалось. Однако, нарушение этих условий вызовет запись шумов перед первой записью, и последующая попытка чтения из этого файла приведет к ошибке. Если задержка между моментами времени открытия файла и записью первой записи слишком продолжительна, то в файл записывается фиктивная строка, состоящая из каких-нибудь безвредных символов, таких как нули, !FF и пробелы (!20). Иногда, после осуществления ввода/вывода с кассеты система издает свистящие звуки. Этот шум можно снять, поместив в ячейку SKCTL (D20F) значение !03, выводя тем самым POKEY из двухтонового (FSK) режима.

CLOSE

Операция CLOSE, примененная в отношении считывания, останавливает мотор магнитофона. Операция CLOSE, примененная в отношении записи, выполняет следующие действия:
- записывает оставшиеся в буфере данные пользователя на ленту,
- записывает символ конца файла,
- останавливает мотор магнитофона.

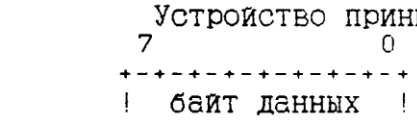
GET CHARACTERS и GET RECORD

Устройство возвращает данные в следующем формате:



PUT CHARACTERS и PUT RECORD

Устройство принимает данные в следующем формате:



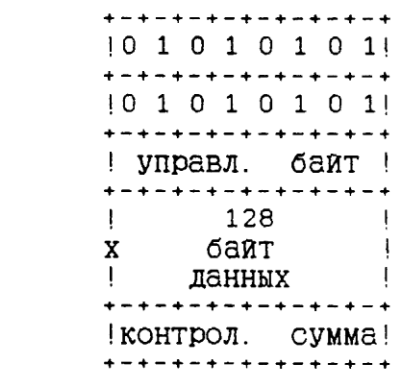
Устройсва не устанавливает значимость отдельных записываемых байт данных. Величина !9B (EOL) - символ конца строки - не вызывает никаких специальных действий.

GET STATUS

Устройство не совершает никаких действий, кроме установки состояния !01.

Теория операции.

Кассетный магнитофон производит считывание и запись всех данных в виде записей фиксированной длины приведенного ниже формата:



Управляющий байт содержит одну из трех следующих величин:
- !FC указывает, что запись полностью заполнена данными.
- !FA указывает, что запись частично заполнена данными; в нее помещено менее 128 байт. Такая ситуация может возникнуть в строке, предшедствующей концу файла. Число занятых пользователем байт данных в этой записи содержится в байте, предшедствующем байту контрольной суммы.
- !FE указывает, что запись является записью конца файла. Для такой записи область данных полностью состоит из нулей.

Программа CIO генерирует и проверяет контрольную сумму. Она является частью записи на ленте, но не содержится в буфере записей устройства CASBUF (03FD). Обработка байт измерения скорости во время чтения описана в приложении L, D1-D7.

Структура файлов.

Кассетный магнитофон осуществляет запись на кассету файлов, имеющих полностью определяемую устройством файловую структуру (гибкий формат). Файл имеет в своем составе три следующих элемента:

- 20 секундовый начальный участок с контрольным тоном.
- Набор фреймов записей данных.
- Фрейм конца файла.

Фреймы данных на кассете имеют следующий формат:

FRAME - тон, предшедствующий записи.
+ запись данных.
+ интервал.

Неинформационные поля фрейма имеют характеристики, зависящие от режима открытия устройства на запись (непрерывного или с паузами). Тон, предшедствующий записи для режима с паузами = 3 секунды контрольного тона, а для непрерывного режима = 25 секунд контрольного тона. Интервал после записи для режима с паузами = до 1 секунды неизестного тона, а для непрерывного режима = от 0 до N секунд неизвестного тона, где N зависит от времени выполнения вашей программы. Интервал между записями состоит из интервала первой записи с добавлением тона, предшедствующего второй записи.

Принтер (P:).

Принтер является устройством, предназначенным только для записи и поддерживающим выполнение следующих функций CIO:

OPEN
CLOSE
PUT CHARACTERS
PUT RECORD
GET STATUS

Принтер может генерировать следующие ошибочные состояния:

!8A-90 - ошибки SIO (см. Приложение C).

Принтер является одним из резидентных устройств и поэтому имеет набор векторов устройства, начинающихся в ячейке E430.

Описание функций CIO.

Ниже приведены зависящие от устройства характеристики стандартных функций CIO.

OPEN

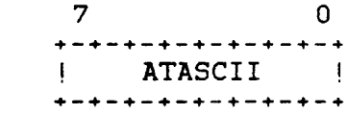
Имя устройства P:. При задании номера устройства и имени файла они игнорируются.

CLOSE

Устройство печатает все оставшиеся в его буфере данные на принтере, заполняя остаток строки пробелами.

PUT CHARACTERS и PUT RECORD

Устройство получает данные печати в следующем формате:



Единственным управляющем кодом для устройства является символ конца строки (EOL). Принтер игнорирует седьмой бит всех байтов данных и печатает оставшееся подмножество, состоящее из 128 кодов (знакогенератор принтера приведен в приложении G). Устройство обеспечивает следующий выбор:

```
       7             0
      +-+-+-+-+-+-+-+-+
AUX2  ! режим  печати !
      +-+-+-+-+-+-+-+-+
```

Где !4E (N) - нормальная печать (40 символов в строке).
!53 (S) - раздвинутая печать (29 символов в строке).
!57 (W) - широкая печать (не поддерживается принтером).
Любое другое значение (включая 00) будет интерпретированно, как нормальный (N) режим и не вызовет ошибки.

GET STATUS

Устройство получает четырехбайтовое состояние с контроллера принтера и помещает его в системную ячейку DVSTAT (02EA). Формат байта состояния приведен ниже.

```
+-+-+-+-+-+-+-+-+
!коман.состояние!  DVSTAT + 0
+-+-+-+-+-+-+-+-+
!предыдущий AUX2!         + 1
+-+-+-+-+-+-+-+-+
!прост. интервал!         + 2
+-+-+-+-+-+-+-+-+
!не используется!         + 3
+-+-+-+-+-+-+-+-+
```

Командное состояние содержит следующие биты состояния и условий:
Бит 0 - получен неправильный командный фрейм.
Бит 1 - получен неправильный фрейм данных.
Бит 7 - информация с контроллера (обычно 0).
Следующий байт содержит величину AUX2, полученную из предшедствующей операции. Байт простоя содержит величину максимального простоя (в секундах), вызываемого контроллером.

Теория операции.

ATARI 820 является построчным принтером, а не посимвольным, поэтому данные должны быть буферизованы и отосланы на устройство в виде записей, соответствующих печатной строке (40 символов для нормальной печати и 29 для раздвинутой). Принтер не имеет специального обозначения для символа конца строки (EOL), поэтому всякий раз при встрече с ним он печатает пробел.

Устройство обработки файлов на дискете (D:).

К моменту написания книги имелось четыре подсистемы управления файлами, поддерживаемых ОС. Начальная версия - 1А. Она и описана в настоящем руководстве. Большую часть из описанного можно применить для версии II, позволяющую работать не только с дискетами одиночной плотности (сектора по 128 байт), но и с дискетами двойной плотности (сектора по 256 байт). В версии III установлены новые структуры расположения файлов (директории), а также в нее внесены изменения для интерфейса. Подсистема обработки файлов включает в себя загружаемый с дискеты (резидентный) обработчик файлов, который хранит на дискете набор проименованных файлов. Возможно подсоединение до четырех дисководов, а на дискете можно расположить до 64 файлов. Системные дискеты ATARI допускают наличие единственного дисковода (D1) и до трех открытых файлов, но как будет описано ниже можно изменять эти параметры.

Устройство, обработки файлов на дискете поддерживает выполнение следующих функций CIO:

OPEN FILE	OPEN DIRECTORY
CLOSE	GET CHARACTERS
GET RECORD	PUT CHARACTERS
PUT RECORD	GET STATUS
NOTE	POINT
LOCK	UNLOCK
DELETE	RENAME
FORMAT	

Устройство обработки файлов на дискете может генерировать следующие ошибочные состояния:

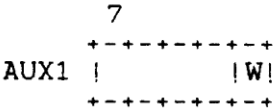
!03 - Последние данные файла.(После следующего чтения наступит конец файла (EOF)).
!88 - Конец файла.
!8A-90 - Ошибки SIO (см. Приложение С).
!A0 - Ошибка в задании номера устройства.
!A1 - Нет буфера для сектора (слишком много открытых файлов).
!A2 - Диск полон.
!A3 - Грубая ошибка ввода/вывода в директории или схеме битов.
!A4 - Внутреннее рассогласование контрольной нумерации файлов.
!A5 - Ошибочно задан номер файла.
!A6 - Ошибка в указательной информации.
!A7 - Данная операция блокирует файл.
!A8 - Неправильное использование специальной команды.
!A9 - Оглавление заполнено (64 файла).
!AA - Файл не найден.
!AB - Ошибка указания (файл не был открыт для корректирования).

Описаний функций CIO.

Ниже приведено описание зависящих от вида устройства стандартных функций CIO.

OPEN FILE

Имя устройства D. Возможно подсоединение до четырех дисководов. Имя диского файла должно состоять не более чем из восьми символов и не может иметь расширение, состоящее более чем из трех символов. Для команды OPEN FILE возможен следующий выбор:



Где W и R - биты направления.
WR=00 - ошибка.
01 - операция OPEN произведена только для чтения.
10 - операция OPEN произведена только для записи.
11 - для чтения и записи (корректирования).
A=1 - указывает на добавления в конец файла при W=1.

Можно использовать следующие возможные опции AUX1:

OPEN INPUT (открыть для ввода) (AUX1=!04).
Указанный файл открыт для ввода. Символы заменители используются для поиска первого соответствия. Если файл не найден, то возвращается код ошибки, и открытия файла не происходит.

OPEN OUTPUT (открыт для вывода) (AUX1=!08).
Указанный файл открывается для вывода, начиная с первого байта этого файла, если он не блокирован. Уже существующий файл будет уничтожен, а затем открыт в том случае, если будет произведена попытка открыть его как новый файл. Если файл еще не существовал , он будет создан. Файл, открытый для вывода, не будет включен в оглавление до момента его закрытия. Если закрытие файла произошло неправильно, то часть или все задействованные им сектора будут утрачены до тех пор, пока дискета не будет переформатирована. Файл, открытый для ввода не может быть одновременно открыт для какого-либо другого вида доступа.

OPEN APPEND (открыт для дополнения) (AUX1=!09).
Указанный файл открывается для вывода, начиная с байта, следующего за последним байтом существующего файла, который уже должен к этому времени существовать в случае, если он не блокирован. Если правильно не закрыть открытый для дополнения файл, то внесенные данные будут утеряны. Существующий файл останется без изменений, а часть или все добавленные сектора будут утрачены до тех пор, пока дискета не будет переформатирована.

OPEN UPDATE (открыть для корректирования) (AUX1=!0C).
Указанный существующий файл будет открыт для корректирования в случае, если тот не блокирован. Разрешается применение операций GET, PUT, NOTE и POINT в любой необходимой комбинации. Если правильно не закрыть файл, открытый для корректирования, может произойти потеря ценной информации. Файл, открытый для корректировки, не может быть расширен.

Спецификация имен устройств/файлов.

Для устройства необходимо задать спецификацию имени устройства/файла в следующем виде:

D(номер):(имя файла)(EOL)

Где (номер)::=1|2|3|4|6

(имя файла)::=(собственное имя).(расширение)(ограничитель)

(собственное имя)::= буква верхнего регистра, за которой могут следовать от 0 до 7 буквенно-цифровых символов. Если длина имени составляет менее восьми символов, то оно будет дополнено пробелами, а если имя превосходит восемь символов, то лишние символы игнорируются.

(расширение)::= от 0 до 3 буквенно цифровых символов. Если расширение опущено, или длина его составляет менее трех символов, оно будет дополнено пробелами, а если длина расширения превосходит 3 символа, то лишние символы игнорируются.

(ограничитель)::= (EOL)|(пробел).

Ниже приведены примеры правильного задания имен файлов на дискете.

```
D1:GAME.SRC
D:MANUAL6
D:.WHY
D3:FILE.
D4:BRIDGE.002
```

Использование символов-заменителей в имени файла

Спецификация имени файла может быть указана с использованием специальных символов-заменителей "`" и "?". Эти символы позволяют сократить имя и/или расширение файла. Символ ? в спецификации допускает наличие в этой позиции любого символа. Например, WH? включает в себя файлы с именами WHO, WHY, WH4 и т.п., но не файл с именем WHAT. Символ `, используемый в имени или расширении файла, заменяет любое количество произвольных символов. Например, WH` включает WHO, WHEN, WHATEVER и т.п. Ниже приведены примеры правильного использования символов заменителей.

`.SCR - файлы, имеющие расширение SCR.

BASIC.` - файлы, имеющие имя BASIC с любым расширением.

`.` - все файлы.

H`.? - файлы, начинающиеся с H и имеющие расширение не более одного символа.

Если заменители использовались в команде OPEN, то будет открыт первый по счету файл (и только один), удовлетворяющий спецификации.

OPEN DIRECTORY

Команда OPEN DIRECTORY позволяет считать информацию из оглавление, относящуюся к выбранному файлу, при помощи стандартных команд GET CHARACTERS и GET RECORD. Считанная информация будет отформатированна как записи ATASCII, пригодные для печати, как показано ниже. Команда OPEN DIRECTORY использует те же параметры CIO, что и стандартная команда OPEN FILE:

```
COMMAND BYTE = $03
BUFFER ADDRESS = указатель на спецификацию имен устройств файлов.
AUX1 = $06.
```

После того, как оглавление открыто, пользователю будет возвращена запись для каждого файла, согласованная со спецификацией OPEN. Запись, содержащая лишь символы ATASCII, форматируется, как показано ниже.

```
                      1
 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
!S!B!   имя файла    !расш.!B!счет.!E!
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Где S = "`" или " ", ` указывает, что файл блокирован.
B = пробел.
имя файла = имя, дополненное пробелами.
расш. = расширение, дополненное пробелами.
счет. = число секторов, входящих в файл.
E = символ конца строки (EOL)(¦9B).

После возвращения последней записи имени файла возвращается еще одна запись. Эта запись указывает на число свободных секторов на дискете. Формат этой записи приведен ниже:

```
                      1
 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
!счет.!    свободные сектора    !E!
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Где счетчик = число не использованных секторов на дискете.
E = (EOL)(¦9B).

Состояния конца файла (¦03 и ¦88) возвращаются, как и в обычном файле данных после считывания последней записи. Открытие другого файла на дискете во время считывания оглавления вызовет последующий сбой, поэтому следует обратить на это внимание, чтобы избежать такой ситуации.

CLOSE

- записывает все оставшиеся данные из буфера в файл на дискете.
- корректирует оглавление и схему распределения памяти данной дискеты.
- освобождает все затраченные внутренние ресурсы.

GET CHARACTERS и GET RECORD

Символы считываются с дискеты и поступают на CIO в виде необработанного потока данных. Ни один управляющий символ ATASCII не несет специальной информации. Состояние ¦88 возвращается в случае, если произошла попытка произвести чтение после последнего байта файла.

PUT CHARACTERS и PUT RECORD

Символы поступают с CIO и записываются на дискету в виде необработанного потока данных. Ни один управляющий символ ATASCII не несет специальной информации.

GET STATUS

Производится проверка файла, и в ICSTA и регистр Y возвращается один из следующих байтов состояния:

¦01 - файл найден и не блокирован.
¦A7 - файл блокирован.
¦AA - файл не найден.

Специальные функции CIO.

Устройство управляющее файлами на дискетах содержит целый набор специальных команд данного устройства. О них рассказывается в следующем параграфе:

NOTE (COMMAND BYTE = !25)

Данная команда возвращает пользователю местоположение на дискете следущего байта, который будет прочитан или записан, и помещает его в ниже указанные переменные.
ICAX3 - младший байт номера сектора на дискете.
ICAX4 - старший байт номера сектора на дискете.
ICAX5 - относительное смещение в секторе в байтах (0-124).

POINT (COMMAND BYTE = !26)

Данная команда позволяет определить точное местоположение на дискете следующего байта, который будет считан или записан. Для использования этой команды необходимо открыть файл с опцией UPDATE (корректирование).
ICAX3 - младший байт номера сектора.
ICAX4 - старший байт номера сектора.
ICAX5 - относительное смещение в секторе в байтах (0-124).

LOCK

Данная команда позволяет предотвратить запись в любой файл. Блокированные файлы не могут быть уничтожены, переименованы, открыты для вывода до тех пор, пока они не будут разблокированы. Перед вызовом CIO задаются следующие параметры IOCB:
COMMAND BYTE = !23
BUFFER ADDRESS = указатель на спецификацию устройства/файла.
После проведения операции LOCK будет изменен следующий параметр IOCB:
STATUS = результат операции LOCK; список кодов возможных состояний можно найти в приложении B.

UNLOCK

Данная команда позволяет снимать блокировку. Перед вызовом CIO задаются следующие параметры IOCB:
COMMAND BYTE = !24
BUFFER ADDRESS = указатель на спецификацию имени устройства/файла.
После проведения операции UNLOCK будет изменен следующий параметр IOCB:
STATUS = результат операции UNLOCK; список кодов возможных состояний можно найти в приложении B.

DELETE

Данная команда позволяет уничтожить любое число незаблокированных файлов из оглавления выбраной дискеты и восстановить, тем самым, область дискеты, которая отводилась под указанные файлы. Перед вызовом CIO необходимо установить следующие параметры IOCB:
COMMAND BYTE = !21
BUFFER ADDRESS = указатель на спецификацию имен устройств/файлов.
После проведения операции DELETE изменяется следующий параметр IOCB:
STATUS = результат операции DELETE; список возможных состояний можно найти в приложении B.

RENAME

Данная команда позволяет изменять имена любого числа неблокированных файлов, расположенных на одной дискете. Требуется задать спецификацию имен файлов в следующей форме:
(спецификация устройства):(спецификация имени файла),(спецификация имени файла)(EOL)
Первое по порядку имя файла будет заменено вторым. Защиты от формирования одинаковых имен не существует. Будучи однажды сформированными, одинаковые имена не могут быть отдельно друг от друга изменены или уничтожены. Однако, команда OPEN FILE всегда выбирает первый файл, удовлетворяющий спецификации, поэтому этот файл всегда будет доступен. Команда RENAME не изменяет содержимого файла, а только меняет имя в оглавлении. Ниже приведены примеры использования команды RENAME:

```
D1:*.SRC,*.TXT
D:TEMP,FDATA
D2:F*,F*.OLD
```

Перед вызовом CIO необходимо установить следующие значения параметров IOCB:
COMMAND BYTE = !20
BUFFER ADDRESS = указатель на спецификацию имени файла.
После проведения операции RENAME будет изменен следующий параметр IOCB:
STATUS = результат операции RENAME; список возможных состояний можно найти в приложении B.

FORMAT

Дискеты с мягким разбиением на сектора должны быть отформатированны до использования их для хранения данных. Команда FORMAT позволяет физически отформатировать дискету. Процесс физического форматирования включает в себя запись новой копии каждого сектора на дискету с мягким разбиением на сектора и заполнить каждый из них нулями. Процесс форматирования создает "пустую" несистемную дискету. После завершения процесса подсистема обработки файлов (FMS) создает первоначальную таблицу данных тома (VTOC) и первоначальное оглавление. Загруженный сектор (1) постоянно зарезирвирован под этот процесс. Перед вызовом CIO необходимо установить следующие значения параметров IOCB:
COMMAND BYTE = !FE
BUFFER ADDRESS = указатель на спецификацию устройства.
После проведения операции FORMAT будет изменен следующий параметр IOCB:
STATUS = результат проведения операции FORMAT; список возможных состояний можно найти в приложении B.
Для создания системной дискеты необходимо записать копии загружаемых файлов в сектора с 2 по N. Это достигается записью файла с именем DOS.SYS. Данное имя всегда распознается подсистемой обработки файлов (FMS), несмотря на то, что первоначально его нет в оглавлении.

Теория операции.

Процесс загрузки с дискеты инициализируется резидентной ОС (см. Раздел 10). ОС считывает с дискеты в память первый сектор и затем передает управление по "адресу продолжения загрузки" (адрес загрузки +6). Программа продолжения загрузки, находящаяся в первом секторе, продолжает загрузку в память оставшейся части подсистемы обработки файлов (FMS), используя дополнительную информацию, хранящуюся в первом секторе. После загрузки FMS будет содержать обработчики дисковых файлов и, возможно, пакет с подпрограммами для дискеты (DOS). После завершения процесса загрузки обработчик дисковых файлов выделит дополнительный участок ОЗУ для создания

буферов секторов. Память для буферов секторов выделяется на основании информации, содержащейся в записи загрузки, как показано ниже.
Байт 9 - максимальное число открытых файлов; один буфер на (максимальное число - 8).
Байт 10 - биты выбора дисковода; один буфер на (1-4 только).
Затем, обработчик дисковых файлов поместит имя D и таблицу векторов устройства в таблицу устройств.
Замечание: Имеется несоответствие между нумерацией секторов дискеты (0-719) обработчиком дисковых файлов и нумерацией секторов дискеты (1-720) дисковым контроллером, в результате которого обработчиком дисковых файлов используются лишь сектора 1-719. Обработчик дисковых файлов для выполнения чтения и записи использует дисковод. Функцией обработки дисковых файлов является поддержка и сохранение структуры оглавления карты битов таким образом, как это описано ниже.

Использование FMS дискеты.

На приведенной ниже схеме указано назначение секторов стандартной дискеты с 720 секторами.

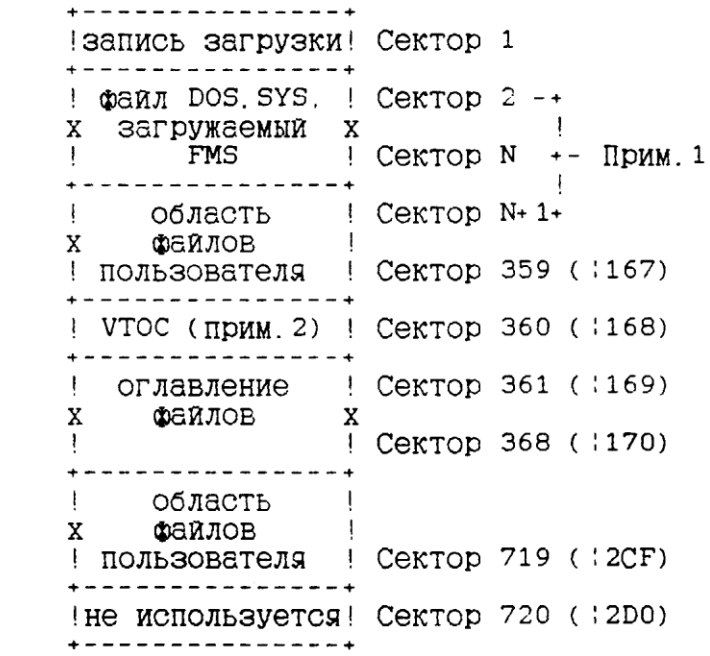


Рис.5-11 Схема использования дискеты подсистемой обработки файлов.

Примечание 1.
В случае, если дискета не системная, область файлов пользователя начинается со второго сектора, а под файл, загружаемый FMS место не отводится. Однако, DOS (DOS.SYS и DUP.SYS) может быть записан на дискету, в которой уже задействованы сектора со 2 по N.
Примечание 2.
VTOC - таблица данных тома.

Формат записи FMS.

Запись загрузки FMS (сектор 1) - специальный вид программы, загружаемой с дискеты. Формат записи загрузки FMS показан ниже.

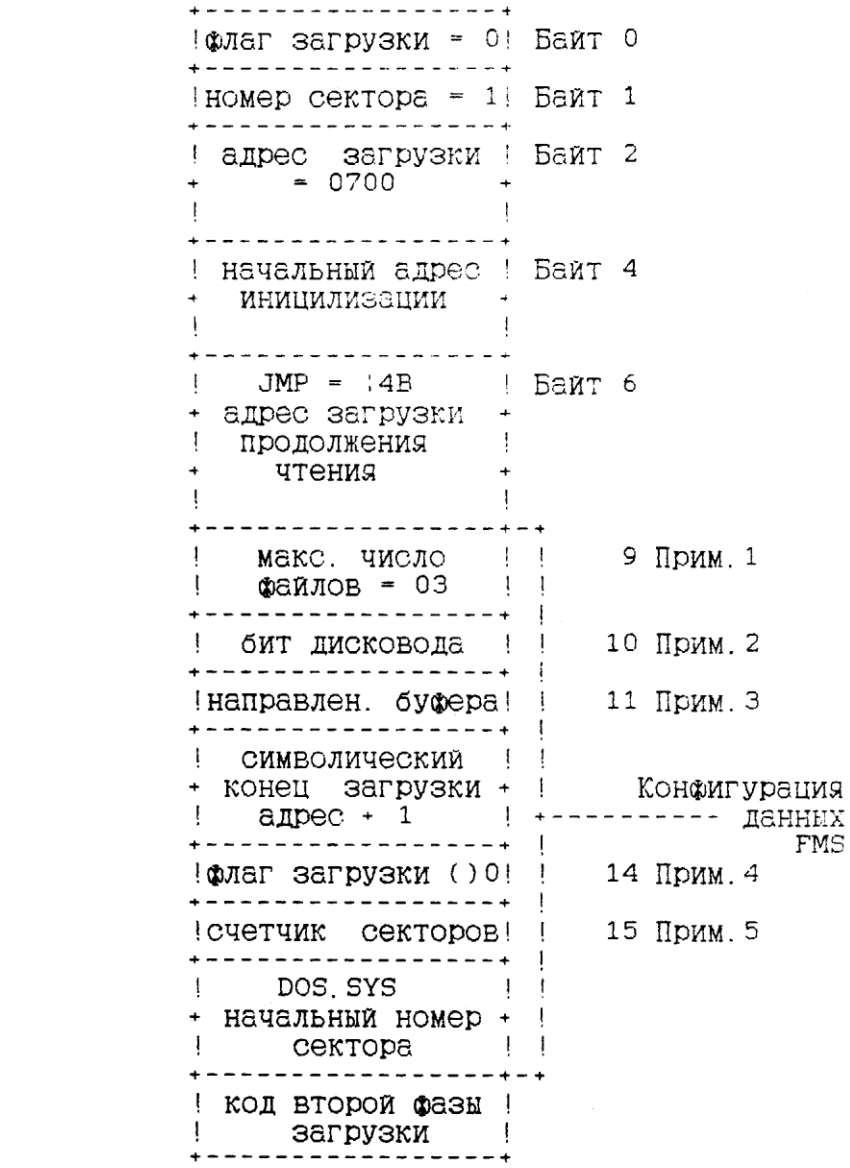
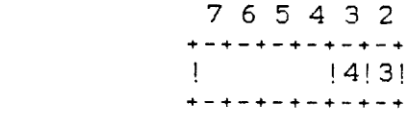


Рис. 5-12 Формат записи загрузки подсистемы обработки файлов.

Байт 9 определяет максимальное допустимое число одновременно открытых файлов. Значение этой величины может изменяться от 1 до 8.

Байт 10 определяет специальные номера дисководов, с которыми предстоит работа, используя нижеприведенную схему раскодирования бит.



Байт 11 определяет направление формирования буфера. Значение байта должно быть равным нулю.

Для того, чтобы процесс загрузки вступил во 2 стадию, значение байта 14 должно быть ненулевым. Данный флаг указывает, что файл DOS.SYS уже записан на дискету.

Данный байт устанавливается как счетчик секторов для файла DOS.SYS. На самом деле этот байт используется редко.

Схема памяти процесса загрузки.

Приведенная ниже диаграмма демонстрирует, как в процессе загрузки сектор (часть файла DOS.SYS) и следующие за ним сектора загружаются в память.

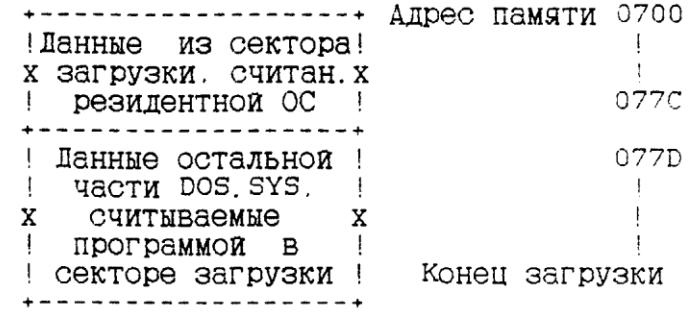
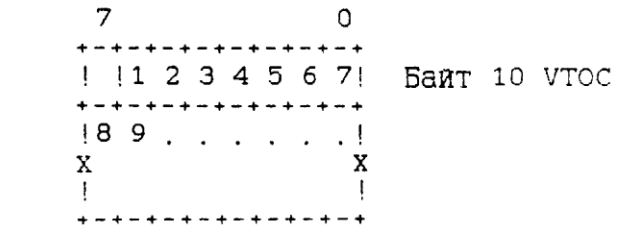


Таблица данных тома.

Формат таблицы данных тома FMS (VTOC, сектор 360) показан на ниже приведенной диаграмме.

Тип оглавления	Байт 0	Прим. 1
Максимальный номер сектора = 02C5 (LO) (HI)	1	Прим. 2
Допустимое число секторов (LO) (HI)	3	Прим. 3
Битовая карта тома	10	

Побитовая схема тома имеет следующую организацию:



В каждой позиции карты ноль означает то, что соответсвующий сектор на дискете задействован, а 1 - что сектор свободен.

Байт типа оглавления должен содержать 0. Номер максимального сектора не используется, поскольку он ошибочно установлен равным 709 (десятичное). На самом деле максимальный сектор равен 719 для обработчика дисковых файлов.

Число доступных секторов устанавливается равным 709 после окончания процесса форматирования. Позже, по мере создания и удаления файлов это число постоянно корректируется, показывая количество свободных секторов. Первоначально зарезирвированными являются сектора 1 и с 360 по 368.

Формат оглавления файлов.

Система обработки файлов отводит восемь секторов (361-368) под оглавление файлов. Каждый сектор содержит информацию о 8 файлах. Таким образом максимальное число файлов в оглавлении - 64.

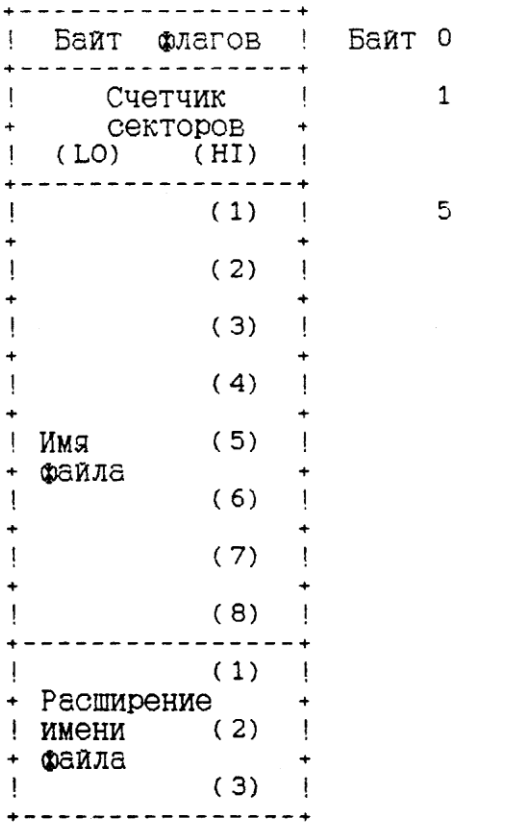


Рис. 5-16 Формат оглавления файлов.

Двоичные разряды байта флагов несут следующую информацию:
Бит 7 = 1 в случае, если файл уничтожен.
Бит 6 = 1 в случае, если файл используется.
Бит 5 = 1 в случае, если файл блокирован.
Бит 0 = 1 в случае, если файл открыт для вывода.
Байт флагов может принимать следующие значения:
 !00 - запись еще не использовалась (нет файла).
 !40 - запись задействована (стандартный закрытый файл).
 !41 - запись задействована (файл открыт для вывода).
 !60 - запись задействована (файл блокирован).
 !80 - запись доступна перед уничтожением файла).
Счетчик секторов показывает число секторов, занимаемых файлом.

Формат сектора файла в подсистеме обработки файлов (FMS).

Ниже приведен формат сектора, содержащего файл данных пользователя.

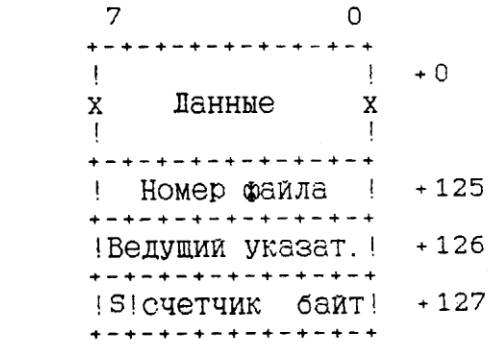


Рис. 5-17 Формат сектора, содержащего файл в подсистеме обработки файлов.

Для определения целостности файла FMS использует номер файла. Номер файла является избыточной информацией. В поле номера файла хранится величина, указывающая положение файла в директории. Если положение файла в директории не соответствует номеру файла, хранящегося в каждом секторе , то обработчик дисковых файлов генерирует ошибку !A4. Поле ведущего указателя содержит десятибитовую величину, указывающую номер следующего сектора файла. Бит S указывает является ли сектор "коротким" (менее 125 байт данных). Когда сектор "короткий", S устанавливается равным 1. Поле счетчика байт содержит количество данных в секторе.

Ввод/вывод без использования CIO.

Вектора резидентных устройств.

Доступ ко всем резидентным устройствам ПЗУ ОС может быть осуществлен при помощи наборов векторов, являющихся частью ПЗУ ОС. Эти вектора замедляют операции ввода/вывода, использующие фиксированные присваивания устройствам. Примером такой операции может служить вывод на дисплей. Для каждого резидентного устройства существует набор векторов, упорядоченных в следующем порядке:

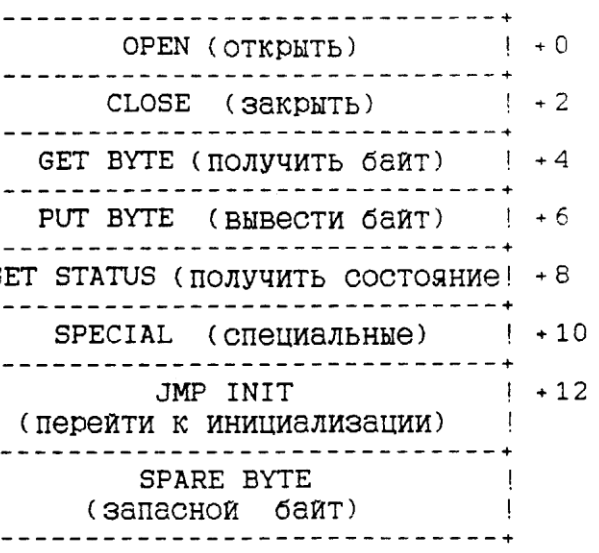


Рис. 5-18 Вектора резидентных устройств.

Детальное описание взаимодействия данных для каждой входной точки устройства можно найти в разделе 9. Каждый из векторов содержит адрес ((LO) - низший, (HI) - высший) входной точки устройства минус 1. Доступ к необходимой программе осуществляется при помощи метода, аналогичного приведенному ниже.

```
VTBASE=:E400          ;База таблицы
                      ;векторов.
      LDA {XX         ;Переход к необх.
      LDX DATA        ;программе.
      JSR GOVEC       ;Переслать данные
                      ;программе.
      LDX {YY         ;Переход к другой
                      ;программе.
      JSR GOVEC       ;Получить данные
      STA DATA        ;от программы.

GOVEC                 ;
      TAY             ;Сохранить регис.A
      LDA VTBASE+1,X  ;Адрес старшего
      PHA             ;байта в стек.
      LDA VTBASE,X    ;Адрес младшего
      PHA             ;байтав стек.
      TYA             ;Восст. регистр A.
      RTS             ;Вернуться в
                      ;программу.
```

Компонента JMP INIT каждого набора векторов совершает переход на программу инициализации. Базовые адреса набора векторов для каждого из устройств приведены ниже.

Экранный редактор (E:)	E400.
Дисплей (S:)	E410.
Клавиатура (K:)	E420.
Принтер (P:)	E430.
Кассетный магнитофон (C:)	E440.

Резидентный дисковод не совместим с CIO, поэтому его интерфейс не использует набор векторов.

Резидентный дисковод.

Резидентный дисковод (который не следует путать с обработчиком дисковых файлов) осуществляет все способы доступа к дискете. Единицей данных, переносимых этим устройством, является сектор, содержащий 128 байт данных. Связь дисковода с пользователем осуществляется при помощи системного блока управления устройством (DCB), используемый также для связи устройства с SIO (см. Раздел 9). Длина DCB составляет 12 байт. Некоторые из них могут быть изменены пользователем, а некоторые используются дисководом и/или процедурой последовательного ввода/вывода (SIO). Пользователь устанавливает требуемые параметры DCB, а затем осуществляет переход JSR DSKINV (E453). Далее дается подробное описание каждого DCB и имен системных файлов для каждого из них.

SERIAL BUS ID (ID последовательной шины) -- DDEVIC (0300).

Значение этого байта устанавливается дисководом и содержит ID последовательной шины для дисковода, к которому должен быть осуществлен доступ. Это значение не может быть изменено пользователем.

DEVICE NUMBER (номер устройства) -- DUNIT (0301).

Значение байта устанавливается пользователем. Оно равно номеру дисковода (1-4), к которому должен быть осуществлен доступ.

COMMAND BYTE (командный байт) -- DCOMND (0302).

Значение байта устанавливается пользователем. Оно кодирует команду, которую должен выполнить дисковод.

STATUS BYTE (байт состояния) -- DSTATS (0303).

Данный байт содержит состояние команды в зависимости от возврата пользователю. Список кодов возможных состояний можно найти в приложении C.

BUFFER ADDRESS (адрес буфера) -- DBUFLO (0304) и DBUFHI (0305).

Данный двухбайтовый указатель содержит адрес источника или места назначения сектора данных дискеты. Для команды состояния дисковода данный адрес задавать не следует. Устройство само помещает адрес буфера в данное поле после его получения.

DISK TIMEOUT VALUE (время, не позднее которого должен быть получен ответ от дисковода) -- DTIMLO (0306).

Значение этой величины в секундах передается устройством для его последующего использования SIO.

BYTE COUNT (счетчик байт) -- DBYTLO (0308) и DBYTHI (0309).

Значение этого двухбайтового счетчика показывает число байт, переносимых с дискеты или на дискету в результате выполнения соответствующей операции. Значение устанавливается устройством.

SECTOR NUMBER (номер сектора) -- DAUX1 (030A) и DAUX2 (030B).

Данный двухбайтовый номер показывает номер сектора на дискете (1-720), который необходимо считать или записать. DAUX1 содержит младший байт, а DAUX2 - старший.

Команды дисковода.

Дисковод воспринимает пять команд:
GET SECTOR прочитать сектор
PUT SECTOR записать сектор
PUT SECTOR WITH VERIFY записать сектор и проверить
STATUS REQUEST запрос о состоянии
FORMAT DISK отформатировать дискету

GET SECTOR (коммандный байт = !52).

Устройство считывает данный сектор в буфер пользователя и возвращает состояние операции. Перед вызовом данной операции необходимо задать следующие параметры блока управления устройством (DCB):
COMMAND BYTE = !52.
DEVICE NUMBER = номер дисковода (1-4).
BUFFER ADDRESS = указатель на 128 байтовый буфер пользователя.
SECTOR NUMBER = номер считываемого сектора.

В зависимости от данных, возвращенных сектором, возможно изменение и некоторых других параметров DCB. Однако, единственный из них, представляющий какой-либо интерес - STATUS BYTE.

PUT SECTOR (командный байт = !50).

Устройство записывает в данный сектор данные из буфера и возвращает состояние операции. Перед вызовом данной операции необходимо задать значения следующих параметров DCB:
COMMAND BYTE = !50.
DEVICE NUMBER = номер дисковода (1-4).
BUFFER ADDRESS = указатель на 128 байтовый буфер пользователя.
SECTOR NUMBER = номер записываемого сектора.

В зависимости от результата операции возможно изменение и некоторых других параметров DCB. Однако, единственный из них, представляющий какой-либо интерес - STATUS BYTE.

PUT SECTOR WITH VERIFY (командный байт = !57).

Устройство производит запись данных из буфера данных в указанный сектор. Данная команда отличается от PUT SECTOR тем, что контроллер считывает сектор данных после записи для ее проверки. За исключением величины COMMAND BYTE последовательность вызова данной процедуры идентична PUT SECTOR.

STATUS REQUEST (командный байт = !53).

Устройство получает четыре байта о состоянии контроллера дисковода и помещает их в системные ячейки, начиная с DVSTAT (02EA). Формат состояния приведен ниже.

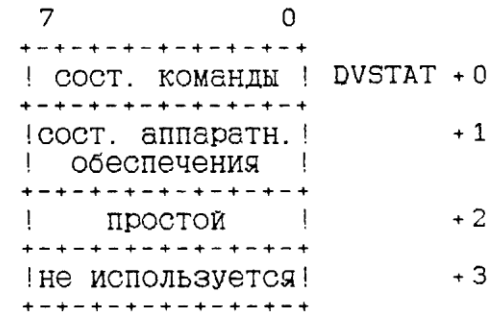


Рис.5-19 Формат DVSTAT четырехбайтового состояния операции.

Биты состояния команды несут следующую информацию:
Бит 0 = 1 - получен неверный командный фрейм.
Бит 1 = 1 - получен неверный фрейм данных.
Бит 2 = 1 - не прошла операция PUT.
Бит 3 = 1 - дискета защищена от записи.
Бит 4 = 1 - активный / резервный.
Байт состояния аппаратного обеспечения содержит регистр состояния чипа контроллера дискеты. Назначение каждого бита данного байта можно найти в документации к этому чипу. Байт простоя содержит величину максимального простоя в секундах для последущего использования ее устройством. Перед вызовом данной операции пользователю необходимо установить следующие значения параметров DCB:
COMMAND BYTE = !53.
DEVICE NUMBER = номер дисковода (1-4).

В зависимости от результата операции возможно изменение некоторых других параметров DCB. Однако, единственный байт, представляющий интерес - STATUS BYTE.

FORMAT DISK (командный байт = !21).

Устройство посылает на контроллер дисковода команду отформатировать дискету и проверить, как прошла операция. Номера всех секторов со сбоями (максимум 63) заносятся в специально отведенный для этого буфер, за которым следуют два байта полностью заполненных единицами (!FFFF). Перед вызовом операции необходимо задать следующие параметры DCB:
COMMAND BYTE = !21.
DEVICE NUMBER = номер дисковода (1-4).
BUFFER ADDRESS = указатель на 128 байтовый буфер.

В зависимости от результата пользователю могут быть полезны значения следующих параметров DCB:
STATUS BYTE = состояние операции.
BYTE COUNT = количество байт в буфере о плохих секторах без учета ограничителя !FFFF. Если плохие сектора отсутствуют, значение счетчика равно нулю.

Ввод/вывод через последовательную шину.

Ввод/вывод на устройства, отличные от клавиатуры, экрана и устройств, подключаемых к контроллерам компьютера ATARI, должен осуществляться через последовательную шину ввода/вывода. Данная шина содержит данные управления и синхронизирующих импульсов, которые используются компьютером для связи с внешними устройствами. Каждое устройство, подключаемое к шине имеет собственный идентификатор и может быть доступно только при прямой адресации. Резидентная операционная система содержит процедуру последовательного ввода/вывода (SIO), обеспечивающую взаимодействие с шиной на уровне стандартизированных языков программирования высокого уровня. SIO используется дисководом, принтером и кассетным магнитофоном. Также SIO предназначен для работы с нерезидентными устройствами (см. Раздел 9) и прикладными программами. Более подробное описание взаимодействия программ с SIO и детальное описание шины приведены в разделе 9.

6. Обработка прерываний.

В разделе 6 описано поведение системы в ответ на различные, приводящие к прерываниям действия, определяются многие вектора ОЗУ и приводятся прцедуры, рекомендуемые к использованию при обработке прерываний. Процессор 6502 обрабатывает прерывания трех типов: перезагрузки чипа, немаскируемые прерываниями (NMI) и маскируемые прерывания (IRQ). Обработка прерываний типа IRQ устанавливается возможной или невозможной командами 6502 CLI и SEI. Невозможность обработки прерываний типа NMI не представляется возможной на уровне процессора. Однако, для прерываний, отличных от прерывания при нажатии клавиши (RESET), это осуществимо в специальной микросхеме ANTIC. Прерывания могут быть вызваны следующим состоянием системы:
Нажатие RESET или включение питания.

NMI - Прерывания дисплейной программы (не используются ОС), конец TV кадра (50/60 Гц), клавиша (RESET).
IRQ - Готовность к выводу через последовательную шину, завершение вывода через последовательную шину, готовность ввода через последовательную шину, прерывание по последовательной шины (не используется системой), прерывающая строка последовательной шины (не используется системой), таймеры 1,2 и 4 специальной микросхемы POKEY, клавиши на клавиатуре, клавиша (BREAK), команда 6502 BRK (не используется системой).

Вектора прерываний по нажатию RESET адресуются ячейкой FFFC в E477, в которой расположен вектор перехода к подпрограмме включения питания. Немаскируемые прерывания (NMI) адресуются ячейкой FFFA в E7B4, где расположена сервисная программа обработки NMI. В свою очередь вектора IRQ адресуются в ячейкой FFFE в E6F3, где располагается сервисная программа обработки маскируемых прерываний. В этих точках можно определить причину прерываний при помощи специальных тестов. Некоторые состояния, приводящие к прерываниям, влекут за собой определенные действия монитора, в то время как для остальных состояний соответствующее прерывание не осуществляется. Система строит вектора ОЗУ таким образом, что в любой момент пользователь может отменить прерывания.

Немаскируемые прерывания.

При немаскируемых прерываниях происходит непосредственная передача управления через вектор ПЗУ сервисной программе обработки NMI. Причину прерывания можно установить по аппаратному регистру NMIST (D40F).

В случае, если произошло прерывание по дисплейной программы NMI осуществляет переход через вектор VDSLST (0200). Поскольку ОС не использует прерывания дисплейной программы. VDSLST указывает на команду RTI.

Если источником прерывания явилось нажатие клавиши (RESET), то производится переход к программе инициализации при перезагрузке системы. Детальное описание можно найти в разделе 7.

Если произошло прерывание по концу TV кадра (VBLANK)- выполняются следующие действия:

Регистры A, X и Y помещаются в стек. Сбрасывается запрос на прерывание в (NMIRES (D40F). Происходит переход через вектор VVBLKI (0222), обычно указывающий на обработчик первой стадии VBLANK. При условии того, что значение VVBLKI (0222) не было изменено, выполняются следующие действия:

Выполняется первая стадия обработки VBLANK.

ОС выясняет установлен ли код программы критичной по времени. Если да, то происходит восстановление всех регистров и возвращение командой RTI из прерывания. Критический код определяется флагом CRITIC (0042) и битом процессора I. Если какое-нибудь из этих значений равно единице, то раздел с прерываниями считается критическим.

Если прерывание осуществлялось не от критического раздела, то выполняется вторая стадия обработки VBLANK.

Данные действия будут выполнены лишь при условии сохранения неизменным VVBKLD.

- Восстанавливаются значения регистров 6502 A, X и Y.
- Выполняется команда RTI.

Замечание:

Вектора ОЗУ VBLANK могут быть изменены пользователем, не оказывая влияния на действия системы. Возможно восстановление первоначальных векторов без необходиности запоминать их. Команда хранящаяся в E45F представляет собой команду перехода (JMP) к обработке первой стадии VBLANK. Адрес, хранящийся в (E460,2) обычно совпадает с величиной хранящейся в VVBLKI. Команда, хранящаяся в E462 представляет собой команду перехода (JMP) к программе выхода из VBLANK. Адрес, хранящийся в (E463,2) обычно совпадает с величиной, хранящейся в VVBLKD. Данные вектора ПЗУ, указывающие на первую стадию обработки VBLANK и на программу выхода из VBLANK помогут в достижении цели.

Замечание:

Любое прерывание VBLANK осуществляет переход через вектор VVBLKI. Через вектор VVBLKI осуществляют переход лишь прерывания VBLANK от разделов с некритическим кодом.

Первая стадия обработки VBLANK.

VBLANK проходит первую стадию обработки всякий раз при возникновении прерывания VBLANK. На первой стадии обработки VBLANK происходит увеличение на единицу счетчика RTCLOK (0012-0014), где RTCLOK+0 является старшим байтом, а RTCLOK+2 - младшим байтом. На первой стадии обработки VBLANK значение системного таймера 1 CDTMV1 (0218,2), если оно не равно нулю уменьшается на единицу. Если же значение таймера стало равным нулю, то происходит косвенный переход JSR через CDTMA1 (0226,2).

Вторая стадия обработки VBLANK.

Вторая стадия обработки VBLANK выполняется в тех случаях, когда не установлен код критический по времени. На второй стадии обработки VBLANK производится обнуление бита I процессора 6502. Это делает возможными IRQ

(прерывания по запросам). Также производится корректировка значений различных аппаратных регистров и переменных ОС в соответствии со следующей схемой:

Переменная ОС	Аппаратный регистр	Причина корректировки
SDLSTH(0231)	DLISTH(D403)	Начало
SDLSTL(0230)	DLISTL(D402)	дисплейной
SDMCTL(02FF)	DMACTL(D400)	программы
CHBAS (02F4)	CHBASE(D409)	
CHACT (02F3)	CHACTL(D401)	
GPRIOR(026F)	PRIOR (D10B)	
COLOR0(02C4)	COLPF0(D016)	Неребочий
COLOR1(02C5)	COLPF1(D017)	режим
COLOR2(02C6)	COLPF2(D018)	
COLOR3(02C7)	COLPF3(D019)	
COLOR4(02C8)	COLBK (D01A)	
PCOLR0(02C0)	COLPM0(D012)	
PCOLR1(02C1)	COLPM1(D013)	
PCOLR2(02C2)	COLPM2(D014)	
PCOLR3(02C3)	COLPM3(D015)	
Константа=8	CONSOL(D01F)	Отключение громкоговорителя консоли.

На второй стадии обработки VBLANK происходит уменьшение значения системного таймера 2 CDTMV2, если оно не равно нулю. Если значение таймера стало равным нулю, выполняется переход JSR через CDTHA2 (0228,2). Также происходит уменьшение значений системных таймеров 3, 4 и 5, если они отличны от нуля. Если же значение одного из них стало равным нулю, то значение соответствующего флага становится равным нулю.

Таймер	Значение таймера	Флаг таймера
3	CDTMV3 (021C,2)	CDTMF3 (022A,1)
4	CDTMV4 (021E,2)	CDTMF4 (022C,1)
5	CDTMV5 (0220,2)	CDTMF5 (022E,1)

Если действует функция автоматического повторения, то символ считывается из клавиатурного регистра POKEY и сохраняется в CH (02FC). На второй стадии обработки VBLANK происходит уменьшение счетчика клавиатуры на единицу, если его значение отлично от нуля, и ни одна клавиша не нажата. На второй стадии обработки VBLANK происходит обработка автоматического повторения клавиатуры. На второй стадии обработки VBLANK происходит считывание данных с игровых контроллеров в переменные ОС в соответствии со следующей таблицей:

Аппаратный регистр	Переменные ОС	Функция
PORTA (D300)	STICK0 (0278)	Джойстики и
	STICK1 (0279)	контроллеры
	PTRIG0 (027C)	PADDLE
	PTRIG1 (027D)	
	PTRIG2 (027E)	
	PTRIG3 (027F)	
POT 0 (D200)	PADDL0 (0270)	Контроллеры
POT 1 (D201)	PADDL1 (0271)	PADDLE
POT 2 (D202)	PADDL2 (0272)	
POT 3 (D203)	PADDL3 (0273)	
TRIG0 (D001)	STRIG0 (0284)	Триггеры
TRIG1 (D002)	STRIG1 (0285)	джойстика

Маскируемые прерывания.

Прерывания по запросу (IRQ) вызывают передачу управления через вектор VIMIRQ (0216). Обычно данный вектор указывает на устройство управления системными прерываниями IRQ. Данное устройство выполняет следующие действия: определяет причину прерываний, исследуя значение регистра IRQST (D20E) и регистры состояния PIA PACTL (D302) и PBCTL (D303). Бит состояния прерывания очищается (устанавливается равным нулю). По выяснению причины одного прерывания происходит его обработка для каждой входной точки обслуживания прерываний. Если необходимо обработать несколько прерываний, то для каждого из них будет сгенегировано отдельное прерывание, которые будут обработаны по очереди. Сервисная программа обработки IRQ взаимодействует с событиями, вызывающими прерывания следующим образом:

- Регистр A 6502 помещается в стек.
- Если прерывание было вызвано готовностью к выводу через последовательную шину ввода/вывода, прерывание очищается и производится переход через вектор VSEROR (020C).
- Если прерывание было вызвано готовностью к вводу через последовательную шину, прерывание очищается, и производится переход через вектор VSERIN (020A).
- Если прерывание было вызвано завершением вывода через последовательную шину, то прерывание очищается, и производится переход через вектор VSEROC (020E).
- Если прерывание было вызвано таймером 1 POKEY, то прерывание очищается, и производится переход через вектор VTIMR1 (0210).
- Если прерывание было вызвано таймером 2 POKEY, то прерывание очищается, производится переход через вектор VTIMR2 (0212).
- Если прерывание было вызвано таймером 4 POKEY, то прерывание очищается. Сервистная программа содержит ошибку, поэтому она переходит к следующему тесту.
- Если прерывание было вызвано нажатием клавиши (отличной от (BREAK), (START), (OPTION), (SELECT)), то прерывание очищается, и происходит переход через вектор VKEYBD (0208).
- Если прерывание было вызвано клавишей (BREAK), прерывание очищается. Установка нулевого значения флага BREAK BRKKEY (0011) повлечет за собой очищение (обнуление) следующих величин:
- флага старт/стоп SSFLAG ((02FF).
- флага задержки курсора CRSINH (02F0).
- флага нерабочего режима ATRACT (004D).

Возвращение из прерывания происходит после восстановления регистра A 6502 из стека.

- Если прерывание было вызвано действующей строкой последовательной шины, прерывание очищается, и происходит переход через вектор VPRCED (0202).
- Если прерывание вызвано прерывающей строкой последовательной шины, прерывание очищается, и происходит переход через вектор VINTER (0204).
- Если прерывание было вызвано командой BRK 6502, происходит переход через вектор VBREAK (0206).
- Если ни одно из указанных событий не имело место, регистр A 6502 восстанавливается и происходит возвращение из прерывания.

Инициализация прерываний.

Полная переинициализация подсистемы прерываний происходит всякий раз при включении питания и нажатии клавиши (RESET). ОС очищает аппаратные регистры и устанавливает значения векторов в соответствии со следующей конфигурацией:

Вектор	Тип	Функция
VDSLST(0200)	NMI	RTI - игнорировать прерывание
VVBLKI(0222)	-"-	Первая стадия VBLANK
CDTMA1(0226)	-"-	Таймер простоя SIO
CDTMA2(0228)	-"-	Нет систем. функций
VVBLKD(0224)	-"-	Возвращение системы из прерывания
VIMIRQ(0216)	IRQ	Системн. обработка IRQ
VSEROR(020C)	-"-	SIO
VSERIN(020A)	-"-	SIO
VSEROC(020E)	-"-	SIO
VTIMR1(0210)	-"-	PLA,RTI - игнорировать прерывание
VTIMR2(0212)	-"-	PLA,RTI - игнорировать прерывание
VTIMR4(0214)	-"-	/ не имеет значение /
VKEYBD(0208)	-"-	Прерывание от клавиат.
VPRCED(0202)	-"-	PLA,RTI - игнорировать прерывание
VINTER(0204)	-"-	PLA,RTI - игнорировать прерывание
VBREAK(0206)	BRK	PLA,RTI - игнорировать прерывание

После инициализации устанавливаются прерывания:
NMI: прерывание по VBLANK разрешено, прерывание по дисплейной программы запрещено.
IRQ: возможны прерывания от клавиатуры, все остальные прерывания запрещены.

Системные таймеры.

ОС содержит пять таймеров общего пользования и часы реального времени. Таймеры - двухбайтовые величины (старший, младший), часы реального времени занимают три байта (старший, средний, младший). Таймеры производят обратный отсчет с некоторой величины до нуля. В зависимости от того, был достигнут нуль или нет, таймеры очищают соответствующий флаг или осуществляют переход JSR через вектор ОЗУ. Часы реального времени производят прямой отсчет, сбрасываясь на нуль при переполнении. Ниже приведена таблица характеристик таймеров и счетчика кадров.

	Название таймера	Флаг/вектор	Назначение
X	CDTMV1(0218)	CDTMA1(0226)	2-байтовый вектор используется SIO
	CDTMV2(021A)	CDTMA2(0228)	2-байтовый вектор
	CDTMV3(021C)	CDTMF3(022A)	1-байтовый флаг
	CDTMV4(021E)	CDTMF4(022C)	1-байтовый флаг
	CDTMV5(0220)	CDTMF5(022E)	1-байтовый флаг
X	RTCLOK(0012)		3-Часы реального времени

X - Эти два таймера обнавляются каждое прерывание VBLANK (на первой стадии обработки). Остальные таймеры могут быть не изменены при установке критического кода.

Советы пользователю.

В данном параграфе описываются методы, применение которых необходимо при использовании прерываний совместно с операционной системой.

Маска прерываний POKEY.

Прерывание ANTIC (прерывание дисплейной программы и конец TV кадра и PIA (прерывания по сигналу на последовательной шине) могут быть маскированы непосредственно (см. Руководство к аппаратному обеспечению). Восемь бит IRQEN (D20E) маскирует прерывания POKEY (прерывания от клавиши (BREAK), знаковых клавиш, прерывания при готовности к последовательному вводу/выводу, при завершении последовательного вывода, прерывания от таймеров 1, 2 и 4). IRQEN является регистром, предназначенным только для записи. Поэтому, для выборочного изменения некоторых бит маски значение этого регистра должно

быть помещено в ОЗУ. Для этой цели используется переменная POKMSK (0010).
Заметим, что сервисная программа обработки IRQ использует и изменяет POKMSK, поэтому изменения в переменную должны вноситься не во время прерываний. Если изменения вносятся на уровне прерываний, сложностей не возникает, поскольку бит I уже установлен (равен 1), а если изменения вносятся на второстепенном уровне, необходимо использовать команды SEI и CLI.

Установка векторов прерываний и таймеров.

Различные сообщения.

Ограничение на очищение бита "I".

Программы обработки VBLANK, дисплейной программы и системного таймера 1 не должны производить очищение бита "I" 6502. Если произошло NMI, вследствие которого был осуществлен переход к какой-либо из этих программ, причем в это время происходила обработка прерывания IRQ, то очищение бита I вызовет повторное прерывание IRQ с непредсказуемыми последствиями.

Ограничения, наложенные на время обработки прерываний.

При использовании последовательного ввода/вывода общее время выполнения первой стадии VBLANK и программы обработки прерываний, составленной пользователем не должно превышать 400 мсек. SIO устанавливает флаг CRITIC при переходе к вводу/выводу через последовательную шину.

Задержка прерываний по причине "WAIT FOR SYNC" (ожидать синхронизации).

Всякий раз при нажатии клавиши на клавиатуре устанавливается WSYNC (D40A) во время подачи звукового сигнала на громкоговоритель консоли. Проблема возникает в случае, если прерывание генерируется в период WAIT FOR SYNC (ожидать синхронизации). Обработка такого прерывания будет задержена на время сканирования одной горизонтальной строки. Восприпятствовать этому невозможно. Можно работать вне данного условия, исследуя VCOUNT (D40B) и задерживая обработку прерываний на одну строку при отсутствии задержки WSYNC.

Блок-схемы.

На следующих страницах иллюстрируется основные процессы обработки прерываний NMI и IRQ.

Обработка IRQ.

1. VIMIRQ.
2. Поместить регистр A в стек.
3. Проверить готовность к последовательному выводу, если нет, то перейти к пункту 7.
4. Очистить состояние.
5. VSEROR.
6. SIO.`
7. Проверить готовность к последовательному вводу, если нет, то перейти к пункту 11.
8. Очистить состояние.
9. VSERIN.
10. SIO.`

11. Проверить завершен ли вывод, если нет перейти к пункту 15.
12. Очистить состояние.
13. SVEROC.
14. SIO.`
15. Проверить не произошло ли прерывание от таймера 1 POKEY, если нет, то перейти к пункту 19.
16. Очистить состояние.
17. VTIMR1.
18. Выход.`
19. Проверить не произошло ли прерывание от таймера 2 POKEY, если нет, то перейти к пункту 23.
20. Очистить состояние.
21. VTIMR2.
22. Выход.`
23. Проверить не произошло ли прерывание от таймера 4 POKEY, если нет, то перейти к пункту 25.
24. Очистить состояние.
25. Проверить не произошло ли прерывание от клавиши на клавиатуры, если нет, то перейти к пункту 29.
26. Очистить состояние.
27. VKEYBD.
28. Клавиатура.`
29. Проверить не произошло ли прерывание от клавиши (BREAK), если нет, то перейти к пункту 33.
30. Очистить состояние, установить флаг BREAK, очистить флаг старт/стоп.
31. Извлечь A из стека.
32. Возвратиться из прерывания.`
33. Проверить не произошло ли прерывание от действующей строки, если нет, то перейти к пункту 37.
34. Очистить состояние.
35. VPRCED.
36. Выход.`
37. Проверить не произошло ли прерывание от прерывающей строки, если нет, то перейти к пункту 41.
38. Очистить состояние.
39. VINTER.
40. Выход.`
41. Проверить не произошло ли прерывание от команды BRK, если нет, то перейти к пункту 44.
42. VBREAK.
43. Выход.`
44. Извлечь A из стека.
45. Возвратиться из прерывания.`

` - конец обработки прерывания.

Обработка NMI.

1. Начало.
2. Проверить не произошло ли прерывание от дисплейной программы, если нет, то перейти к пункту 5.
3. VDSLST.
4. Возвратиться из прерывания.`
5. Поместить регистр A в стек.
6. Проверить не произошло ли прерывание от вертикального пропуска, если да, то перейти к пункту 8.
7. Перезагрузка системы.`

8. Поместить X и Y в стек и очистить состояние.
9. VVBLKI.
10. Первая стадия.
11. Проверить не произошло ли прерывание от критического раздела, если да, то перейти к пункту 15.
12. Очистить бит I.
13. Вторая стадия.
14. VVBLKD и перейти к пунту 16.
15. XITVBL.
16. Восстановить значения регистров.
17. Возвратиться из прерывания.`

` - конец обработки прерывания.

7. Инициализация системы.

В разделе 7 проводится детальное рассмотрение системных процессов при включение питания и при перезагрузке системы. Сначала рассматривается процесс включения питания, а уже затем на основе различий между двумя процессами - процесс перезагрузки системы. Включение питания (холодный старт) и нажатие клавиши (RESET) (горячий старт) приводят к инициализации системы. В дополнение к этому для данных процессов имеются вектора E474 (для перезагрузки системы) и E477 (для включения питания). Таким образом эти процессы могут быть вызваны пользователем. В процессе инициализации системы при перезагрузке производятся часть действий, что и при включении питания. При включении питания производится инициализация области ОС и области пользователя в ОЗУ, в то время как при перезагрузке происходит инициализация только области ОС в ОЗУ. В обоих случаях ОС вызывает инициализацию входных точек программы более высокого уровня, позволяющую прикладной программе провести инициализацию переменных. Нажатие клавиши (RESET) вызывает немаскируемое прерывание (NMI). Перезагрузка чипа 6502 не происходит. Если процессор блокирован, клавиша (RESET) не может его разблокировать. Для того, чтобы решить проблему необходимо выключить и включить компьютер.

Процедура инициализации при включении питания (холодный старт).

При включения питания ОС выполняет действия в следующем порядке:

1. Устанавливает следующие состояния процессора 6502.
- Командой SEI устанавливаются невозможными прерывания IRQ.
- Командой CLD очищается десятичный флаг.
- Указатель стека устанавливается на $FF.
2. ОС устанавливает нулевое значение флага горячего старта WARMST (0008).
3. ОС проверяет наличие диагностического картриджа.
- Адрес картриджа BFFC = 0?
- Память в BFFC не является ОЗУ?
- Бит 7 байта в BFFD = 1?

Если все указанные вопросы имеют положительный ответ, то через вектор в BFFD управление передается диагностическому картриджу.

4. ОС определяет младший адрес памяти, не содержащийся в ОЗУ, исследуя первый байт каждого четырехкилобайтного "блока" на предмет того можно ли построить к нему дополнение или нет. Если построить дополнение возможно, то восстановливается первоначальная величина, и тестирование продолжается. Если же дополнение построить невозможно, то содержимое байта принимается системой первым адресом памяти, отличной от ОЗУ. Старший байт временно располагается в TRAMSZ (0006).
5. Во все нижеприведенные адреса аппаратных регистров (большинство из которых не декодируется аппаратными средствами) помещается нуль.

6. ОС очищает ОЗУ с ячейки 0008 до ячейки, адрес которой определяется в пункте 4.
7. По умолчанию значение вектора управления "без картриджа" устанавливается так, чтобы указывать на программу самотестирования. В случае, если передача управления картриджу не произошла, управление передается через этот вектор в конце инициализации.
8. Флаг холодного старта COLDST (0244) устанавливается равным -1 (локальное использование).
9. Устанавливаются следующие значения границы экрана для физической строки длиной 38 символов: Левая граница - 2, правая - 39. Максимальная длина строки (40 символов) может быть получена установкой границ в 0 и 39. ОС смещает левую границу по причине того, что на многих телевизорах два крайних левых стобца не всегда полностью видны на экране.
10. Происходит инициализация векторов прерываний ОЗУ с VDSLST (0200) по VVBLKD (0224). Значения, устанавливаемые после инициализации, приведены в разделе 6.
11. Устанавливаются требуемые ненулевые значения ячеек ОЗУ ОС, как показано ниже.
- Флаг клавиши (BREAK) BRKKEY (0011) - 1.
- Указатель на верхнюю часть памяти MEMTOP (02E5) = наименьшему адресу памяти, отличной от ОЗУ (вычисляется в пункте 4).
- Указатель на нижнюю часть памяти MEMLO (02E7) = 0700. В дальнейшем, в случае загрузки с дискеты или кассеты это значение может быть изменено.
- Вызывается инициализация следующих резидентных программ:
экранного редактора,
дисплея,
клавиатуры,
принтера,
магнитофона,
монитора CIO,
монитора SIO,
обработчика прерываний.
- Проверяется нажатие клавиши (START). В случае, если она нажата, устанавливается флаг запроса на загрузку с кассеты.
12. Командой CLI устанавливаются возможными прерывания IRQ 6502.
13. Таблица устройств HATABS (031A) инициализируется таким образом, чтобы указывать на резидентные устройства. Информацию, относящуюся к таблице устройств, можно найти в разделе 9.
14. Если ОЗУ не расширено до области картриджа, то для того чтобы определить вставлены картриджи или нет исследуются адреса картриджей A и B. Если значение ячейки 9FFC равно нулю, то осуществляется переход JSR через вектор 9FFC, вызывая инициализацию картриджа B. Подразумевается возврат от картриджа. Если значение ячейки BFFC равно нулю, то выполняется переход JSR через вектор в BFFE, вызывая инициализацию картриджа A. Подразумевается возврат от картриджа.
15. IOCB (УББВ) 0 устанавливается для проведения операции OPEN по отношению к экранному редактору, после чего выполняется операция OPEN. Экранный редактор задействует высшую часть ОЗУ и изменит соответствующим образом значение MEMTOP. Если данная операция не пройдет, процесс полной инициализации будет проведен снова.
16. Производится задержка с целью гарантировать прерывание VBLANK, причем таким образом, чтобы получить устоявшееся изображение перед тем, как продолжить процесс.
17. В случае, если флаг запроса загрузки с кассеты установлен (см. пункт 11) попытаться произвести загрузку с кассеты. Детальное описание загрузки с кассеты можно найти в разделе 10.
18. В случае наличия одного из ниже приведенных условий попытаться произвести загрузку с дискеты:

- Отсутствие картриджей.
- Вставлен картридж B и значение бита 0 в 9FFD равно 1.
- Вставлен картридж A и значение бита 0 в BFFD равно 1.

Подробности операции загрузки с дискеты можно найти в разделе 10.

19. Значение флага COLDST изменяется, чтобы показать, что процесс холодного старта завершен.

20. В данный момент процесс инициализации завершен, и на оставшихся шагах выясняется, куда передать управление. Если вставлен картридж A, а значение бита 2 BFFD равно 1, то осуществляется переход JMP через вектор в BFFA. Если вставлен картридж B, а значение бита 2 9FFD равно 1, то осуществляется переход через вектор в 9FFA. Или переход осуществляется через вектор DOSVEC, который может указывать на программу самотестирования (принимается по умолчанию), программу, загружаемую с кассеты или программу, загружаемую с дискеты. Как показано в разделе 10, значение DOSVEC может быть изменено загружаемой программой.

Процедура инициализации при перезагрузке системы (горячем старте).

При перезагрузке системы выполняются следующие функции в указанном порядке:

A. Совпадает с пунктом 1 холодного старта.
B. Флаг горячего старта WARMST устанавливается равным 1 (истина).
C. Совпадает с пунктами 3-5 холодного старта.
D. Обнуляются ячейки 0200-03FF и 0010-007F ОЗУ ОС.
E. Совпадает с пунктами 9-16 холодного старта.
F. В случае, если в процессе холодного старта загрузка с кассеты произошла успешно, осуществляется переход JSR через вектор CASINI (0002). Подробнее описание загрузки с кассеты можно найти в разделе 10.
G. Совпадает с пунктом 18, за исключением того, что вместо загрузки программы на дискете осуществляется переход JSR через вектор DOSINI (000C) в случае, если в процессе холодного старта загрузка с дискеты завершилась успешно.
H. Совпадает с пунктами 19 и 20.

Заметим, что процедуры инициализации и основные входы в различные программы осуществляются всякий раз при перезагрузке системы и при включении питания (см. шаги 14, 17, 18, 20, F и G). Если используемый пользователем код запуска инициализации должен иметь различные значения для холодного и горячего старта, необходимо рассмотреть флаг (0008). Значение WARMST = 0 означает холодный старт, в противном случае - горячий старт.

8. Пакет математики с плавающей запятой.

В данном разделе описан пакет для работы с двоично кодированными десятичными числами, являющийся резидентным в ПЗУ ОС. Числа представляются в виде 6-байтовых величин: пятибайтовой (10 двоичнокодированных десятичных цифр) мантиссы и однобайтовой экспоненты. Двоично-десятичное представление чисел было выбрано, для того, чтобы избежать ошибок округления при десятичном делении, которые часто встречаются при двоичном представлении. Пакет позволяет производить следующие операции:

- преобразование ASCII в числа с плавающей запятой,
- преобразование чисел с плавающей запятой в ASCII,
- преобразование целых чисел в числа с плавающей запятой,
- деление, умножение, сложение и вычитание чисел с плавающей запятой,
- логарифмические, экспотенциальные и полиномиальные оценки чисел с плавающей запятой,
- обнуление, загрузка, хранение и перемещение чисел с плавающей запятой.

Операция над числами с плавающей запятой выполняются после вызова одной из

поддерживающих программ (каждая из которых имеет фиксированный адрес в ПЗУ) в результате установки значений одного или нескольких псевдорегистров ПЗУ. Результат желаемой операции окажет влияние на значения псевдорегистров. Ниже описаны первичные псевдорегистры. Их адреса приведены в скобках.
FR0 (00D4) - 6-байтовая внутренняя форма числа с плавающей запятой.
FR1 (00E0) - 6-байтовая внутренняя форма числа с плавающей запятой.
FLPTR (00FC) - 2-байтовый указатель (младший, старший) на число с плавающей запятой.
INBUFF (00F3) - 2-байтовый указатель (младший, старший) на текстовой буфер ASCII.
CIX (00F2) - однобайтовый индекс, используемый как сдвиг буфера, на который указывает INBUFF.
LBUFF (0580) - конечный буфер для программы FASC.

Последовательность вызова функций.

При описании программ для работы с числами с плавающй запятой предполагается, что данная программа не изменяет значение псевдорегистра. Числа, приведенные в скобках (xxxx), представляют собой адреса программ в ПЗУ.

Преобразование ASCII в числа с плавающей запятой (AFP).

Функция: Данная программа, получая на входе строку ASCII формирует на выходе внутреннее представление числа с плавающей запятой.
Последовательность вызова:
INBUFF = указатель на буфер, содержащий представление числа в ASCII.
CIX = смещение в буфере на первый байт числа в ASCII.
JSR AFP (D800).
BCS - первый байт числа в ASCII задан неверно.
FR0 = число с плавающей запятой.
CIX = сдвиг буфера на следующий за числом в ASCII байт.
Алгоритм: Программа считывает байты из буфера до тех пор, пока не встретится байт, не входящий в это число. Просмотренные байты преобразуются в число с плавающей запятой. Если первый считанный байт неверен, то в качестве флага устанавливается бит переноса.

Преобразование числа с плавающей запятой в ASCII (FASC).

Функция: Данная программа преобразует внутреннюю форму числа с плавающей запятой в ASCII.
Последовательность вызова:
FR0 = число с плавающей запятой.
JSR FASC (D4E6).
INBUFF = указатель на первый байт числа в ASCII. В последнем байте представления числа в ASCII установлен знаковый бит. Символ конца строки далее не следует.
Алгоритм: Программа преобразует число с плавающей запятой в пригодную для печати форму (ASCII). Указатель INBUFF будет показывать на часть LBUFF, в которой хранится результат.

Преобразование целых чисел в числа с плавающей запятой (IFP).

Функция: Данная программа преобразует двухбайтовое положительное число во внутреннее представление числа с плавающей запятой.
Последовательность вызова:
FR0 = целое число (FR0+0 - младший байт, FR0+1 - старший байт).
JSR IFP (D9AA).
FR0 = представление целого числа в виде числа с плавающей запятой.

Преобразование числа с плавающей запятой в целое (FPI).

Функция: Данная программа преобразует положительное число с плавающей запятой в ближайшее двухбайтовое целое.
Последовательность вызова:
FR0 = число с плавающей запятой.
JSR FPI (D9D2).
BCS - число с плавающей запятой отрицательно или)= 65535.5.
FR0 = двухбайтовое целое (FR0+0 - младший байт, FR0+1 - старший байт).
Алгоритм: Программа в процессе выполнения производит округление до ближайшего целого, а не просто отбрасывает дробную часть.

Сложение чисел с плавающей запятой (FADD).

Функция: Данная программа выполняет сложение двух чисел с плавающей запятой.
Последовательность вызова:
FR0 = число с плавающей запятой (первое слагаемое).
FR1 = число с плавающей запятой (второе слагаемое).
JSR FADD (DA66).
BCS - результат лежит вне диапазона.
FR0 = результат FR0+FR1.
FR1 - изменяется.

Вычитание чисел с плавающей запятой (FSUB).

Функция: Данная программа производит вычитание двух чисел с плавающей запятой.
Последовательность вызова:
FR0 = уменьшаемое число с плавающей запятой.
FR1 = вычитаемое число с плавающей запятой.
JSR FSUB (DA60).
BCS - результат операции FR0-FR1.
FR1 - изменяется.

Умножение чисел с плавающей запятой (FMUL).

Функция: Данная программа выполняет умножение двух чисел с плавающей запятой.
Последовательность вызова:
FR0 = число с плавающей запятой (первый множитель).
FR1 = число с плавающей запятой (второй множитель).
JSR FMUL (DADB).
BCS - результат вышел из диапазона.
FR0 = результат FR0*FR1.
FR1 - изменяется.

Деление чисел с плавающей запятой (FDIV).

Функция: Данная программа производит деление двух чисел с плавающей запятой и проверяет делитель на равенство нулю.
Последовательность вызова:
FR0 = делимое число с плавающей запятой.
FR1 = делитель с плавающей запятой.
JSR FDIV (D828).
BCS - результат вышел из диапазона или значение делителя равно нулю.
FR0 = результат операции FR0/FR1.
FR1 - изменяется.

Логарифмы от чисел с плавающей запятой (LOG и LOG10).

Функция: Данные программы вычисляют натуральный или десятичный логарифм числа с плавающей запятой.

Последовательность вызова:

FR0 = число с плавающей запятой.
JSR LOG (DECD) - для натурального логарифма или JSR LOG10 (DED1) - для десятичного логарифма.
BCS - отрицательное число или переполнение.
FR0 = логарифм с плавающей запятой.
FR1 - изменяется.

Алгоритм: Оба логарифма вычисляются сначала как десятичные при помощи полиномиальной аппроксимации. Натуральный же логарифм вычисляется делением логарифма по основанию 10 на константу LOG10(e). Логарифм числа Z вычисляется следующим образом:

F^(10^^Y)=Z, где 1(=F(10 (нормализация).
L=LOG10 (вычисляется полиномиальной аппроксимацией).
LOG10(Z)=Y+L LOG(Z)=LOG10(Z)/LOG10(e).

Замечание: Данная программа не выдает сообщения об ошибке в случае, если введенное число равно нулю; результат LOG10 в данном случае приблизительно равен -129.5.

Возведение в степень, равную числу с плавающей запятой (EXP и EXP10).

Функция: Возведение в степень.

Последовательность вызова:

FR0 = показатель степени с плавающей запятой (Z).
JSR EXP (DDC0) для e^^Z или JSR EXP10 (DDCC) для 10^^Z.
BCS - переполнение.
FR0 = результат с плавающей запятой.
FR1 - изменяется.

Алгоритм: Сначала, в обоих случаях, вычисления производятся для основания 10. Значение для основания e находится из равенства: e^^X=10^^(X^LOG10(e)). Вычисление степени с основанием 10 производится за два приема при помощи равенства: 10^^X=10^^(I+F)=(10^^I)^(10^^F) - где I - целая часть X, а F - дробная часть. Выражение 10^^F оценивается с помощью полиномиальной аппроксимации, а 10^^I является непосредственной модификацией возведения в степень с плавающей запятой.

Полиномиальная оценка числа с плавающей запятой (PLYEVL).

Функция: Данная программа осуществляет полиномиальную оценку степени N.

Последовательность вызова:

X, Y = указатель (X - младший байт, Y - старший) на список коэффициентов с плавающей запятой (A(I)), упорядоченных в порядке убывания порядков (под каждый коэффициент отводится 6 байт).
A = набор коэффициентов.
FR0 = независимая переменная с плавающей запятой (Z).
JSR PLYEVL (DD40).
BCS - переполнение или другая ошибка.
FR0 = результат операции A(N)^Z^^N+A(N-1)^Z^^(N-1)+...+A(1)^Z+A(0).
FR1 - изменяется.

Алгоритм: Значение полинома P(Z)=SUM(I=0 TO N)(A(I)^Z^^I) вычисляется при помощи приведенного ниже стандартного метода:

P(Z)=(...(A(N)^Z+A(N-1))^+...+A(1))^Z+A(0)

Очистить FR0 (ZFR0).

Функция: Данная программа устанавливает нули во всех разрядах псевдорегистра FR0.
Последовательность вызова:
JSR ZFR0 (DA44).
FR0 = ноль.

Очистить FR1 (ZFR1).

Функция: Данная программа устанавливает нули во всех разрядах псевдорегистра FR1.
Последовательность вызова:
JSR ZFR1 (DA46).
FR1 = ноль.

Загрузка числа с плавающей запятой в FR0 (FLD0R и FLD0P).

Функция: Эти программы загружают в псевдорегистр FR0 число с плавающей запятой, определяемое последовательностью вызова.
Последовательность вызова:
X, Y = указатель (X - младший байт) на число с плавающей запятой.
JSR FLD0R (DD89)
или
FLPTR = указатель на число с плавающей запятой.
JSR FLD0P (DD8D).
FR0 = число с плавающей запятой (в любом случае).
FLPTR = указатель на число с плавающей запятой (в любом случае).

Загрузка числа с плавающей запятой в FR1 (FR1R и FLD1P).

Функция: Данные программы производят загрузку в псевдорегистр FR1 числа с плавающей запятой, определяемого следующей последовательностью вызова.
Последовательность вызова:
Совпадает с предыдущем пунктом, за исключением того, что результат помещается в FR1, а не в FR0. FLD1R (DD98) и FLD1P (DD9C).

Размещение в памяти числа с плавающей запятой из FR0 (FSTOR и FSTOP).

Функция: Данные программы размещают содержимое псевдорегистра FR0 по адресу, определяемому следующей последовательностью вызова:
Последовательность вызова:
Совпадает с описанием, данным в предыдущем пункте, за исключением того, что число с плавающей запятой размещается из FR0, а не помещается в FR0. FSTOR (DDA7) и FSTOP (DDAB).

Перемещение числа с плавающей запятой из FR0 в FR1 (FMOVE).

Функция: Данная программа осуществляет перенос числа с плавающей запятой из FR0 в псевдорегистр FR1.
Последовательность вызова:
JSR FMOVE (DDB6).
FR1=FR0 (в FR0 значение не изменяется).

Использование ресурсов.

Пакет математики с плавающей запятой для выполнения описанных ранее функций использует следующие ячейки ОЗУ:
с 00D4 по 00FF
с 057E по 05FF.
Данные ячейки можно использовать при написании собственной программы лишь в том случае, если в ней не задействован пакет математики с плавающей запятой.

Особенности использования.

Числа с плавающей запятой имеют внутреннее представление в виде шестибайтовых величин, 5 байт (10 двоично кодированных десятичных цифр), которые отводятся под мантису, один байт - под экспоненту (спепень). Мантисса обычно нормируется таким образом, чтобы старший байт (заметим байт, а не двоично кодированная десятичная цифра) был ненулевым. Старший бит байта экспоненты указывает знак мантиссы: мантисса положительна в случае равенства его нулю и отрицательна в противном случае. Оставшиеся семь бит экспоненты указывают степень в системе исчисления по основанию 64. В результате число представляет степень 100 (в десятичной системе). Данный формат хранения дает возможность мантиссе включать до 10 двоично кодированных десятичных цифр даже в том случае, если экспонента представляет собой степень десяти. Мантисса включает 9 двоично кодированных десятичных цифр в случае, когда экспонента является четной степенью десяти. Десятичная запятая обычно располагается на самой правой позиции первого байта.

Экспонента, значение которой превосходит или равно 64 соответсвует числу, большему или равному единице. Нуль соответствует нулевая мантисса и нулевая экспонента. Для проверки результата какой-либо стандартной программы необходимо проверить экспоненту или первый байт мантиссы на равенство нулю. Число с плавающей запятой по абсолютному значению должно лежать в пределах от 10``-98 до 10``+98 или должно равнятся нулю. Между положительными и отрицательными числами наблюдается полная симметрия, за исключением того, что генерация отрицательного нуля не представляется возможной. Все вычисления производятся с точностью до 9 или 10 десятичных цифр, но точность заметно снижается для функций, использующих полиномиальную аппроксимацию (логарифм и возведение в степень). Вместе с тем здесь имеют место все те сложности, которые присущи системам с плавающей запятой. Например, вычитание двух величин, значения которых почти равны, сложение несоизмеримых величин дадут результат с потерей значащих цифр. Анализ диапозона данных и порядок оценки выражения может потребоватся различным прикладным программам. В ниже приведенных примерах для лучшего представления о формах хранения проводится сравнение числа с плавающей запятой с их внутренним представлением. До этого момента все числа выражались в десятичной системе исчисления, но в данных примерах используется шестнадцатеричная система исчисления. Заметим, что десятичное число 64 записывается в шестнадцатеричной системе как 40:
Число: 0.02=2`10``-2=2`100``-1
Хранимая величина: 3F 02 00 00 00 00 (степень числа с плавающей запятой = 40-1)
Число: -0.02=-2`10``-2=-2`100``-1
Хранимая величина: BF 02 00 00 00 00 (степень числа с плавающей запятой = 80+40-1)
Число: +37.0=3.7`10``1=37`100``0
Хранимая величина: 40 37 00 00 00 00 (степень числа сплавающей запятой = 40+0)

```
Число: -4.60312486`10``11=-46.03..`100``5
Хранимая величина: C5 46 03 01 24 86 (степень числа с плавающей запятой =
80+40+5)
Число: 0.0
Хранимая величина: 00 00 00 00 00 00 (особый случай)
```

9. Подключение дополнительных периферийных устройств.

В данном разделе приведены требования к взаимодействию с нерезидентными устройствами, к которым осуществляется доступ через программу CIO. Согласование с SIO определяется для устройств, использующих последовательную шину ввода/вывода. Взаимодействие пользователя с аппаратными средствами формируется подсистемой ввода/вывода на трех различных програмных уровнях: CIO, отдельные устройства и SIO. CIO выполняет следующие функции:

- Введение соответствия между логическими именами устройств и самими устройствами (в ходе OPEN).
- Обслуживание управляющего блока ввода/вывода (IOCB).
- Формирование логических записей.
- Формирование буфера пользователя.

Следующим уровнем являются драйверы устройства. Они выполняют следующие функции:

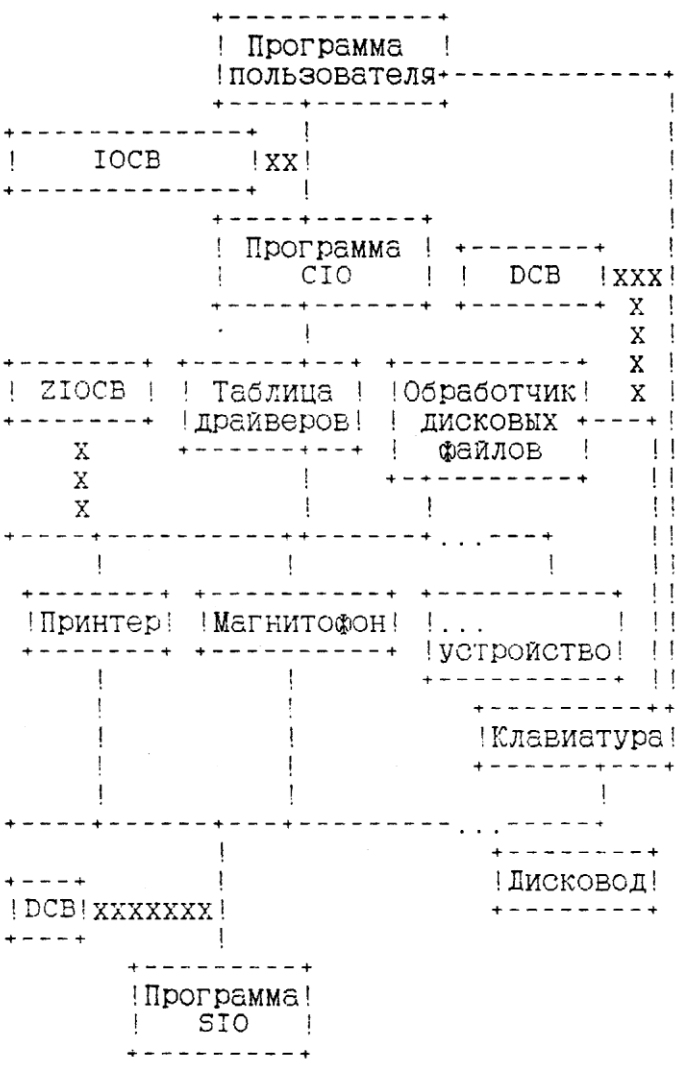
- Инициализация устройства при включении питания и перезагрузке системы.
- Зависящая от устройств поддержка команд CLOSE и OPEN.
- Побайтовый ввод и вывод.
- Специальные операции, зависящие от устройств.
- Поддержка зависящих от устройств команд.
- Управление буфером данных устройства.

SIO находится на самом низком уровне (для периферийных устройств последовательной шины). Он выполняет следующие функции:

- Управление всем вводом/выводом через последовательную шину, соответствующим протоколу шины.
- Возобновление операции при ошибке.
- Возврат ошибочных состояний в ответ на ошибочные явления.

В каждом взаимодействии используется своя управляющая структура:

- Пользователь/CIO Управляющий блок ввода/вывода (IOCB).
- CIO/Устройство IOCB нулевой страницы (ZIOCB).
- Устройство/SIO Блок управления устройством (DCB).



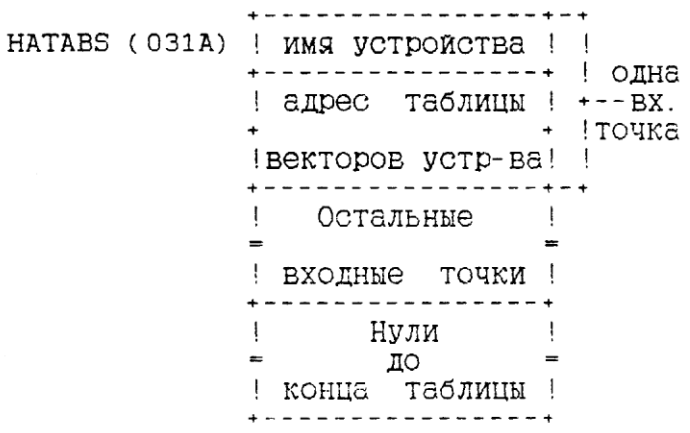
Где DCB - блок управления устройством.
"---" - управление, "XXX" - данные.
Заметим следующее:

1. Клавиатура, дисплей и экранный редактор не используют SIO.
2. Дисковод доступен непосредственно из CIO.
3. DCB (блок управления устройством) дважды указан на схеме.

Рис. 9-1 Блок-схема подсистемы ввода/вывода.

Таблица драйверов устройств.

Таблица драйверов устройств является резидентной таблицей ОЗУ. В ней хранятся односимвольные имена устройств (например K, D, C и т.п.) и адреса драйверов, используемые программой CIO. После инициализации системы при включении питания или при перезагрузке таблица содержит входные точки для следующих устройств: клавиатуры (K), дисплея (S), экранного редактора (E) кассетного магнитофона (C). Для подключения нового устройства, необходимо добавить элемент в таблицу драйверов устройств после инициализации. Формат таблицы приведен ниже.



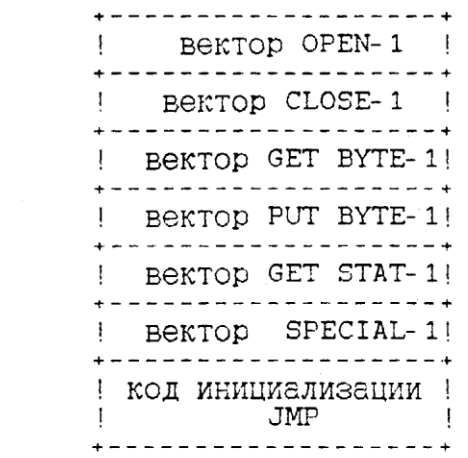
Данная 38-байтовая таблица может содержать 12 входных точек, с учетом того, что последние два байта заполнены нулями. Имя устройства для каждой входной точки представляет собой символ ATASCII, а адрес устройства указывает на таблицу векторов устройства. (Об этом будет рассказано в следующем разделе.)

Взаимодействие CIO с драйверами устройствами.

В данном разделе описано взаимодействие между программой центрального ввода/вывода и отдельными драйверами.

Механизм вызова.

Каждый драйвер имеет следующую таблицу векторов:



Адрес в таблице драйверов устройств указывает на первый байт таблицы векторов. Первые шесть элементов таблицы представляют собой вектора, содержащие адрес без единицы программы, управляющей указанной функцией. Седьмой элемент таблицы представляет собой команду JMP перехода к программе инициализации устройства. При обращении к драйверу CIO использует только адреса этой таблицы. Каждое обращение пользователя к CIO вызывает один или несколько запросов к драйверу по входным точкам, указанными в таблице векторов. Передача параметров осуществляется при помощи регистров 6502 A, X и Y и IOCB нулевой страницы с именем ZIOCB (0020). Обычно регистр A

используется для данных, регистр X содержит индекс IOCB, а регистр Y используется для кода возврата. IOCB нулевой страницы ZIOCB представляет собой копию начального IOCB. Однако, в процессе обработки некоторых команд CIO может изменять адрес буфера и длину буфера ZIOCB, но не начального IOCB (информацию, относящуюся к начальному IOCB можно получить в разделе 5). В приложении B приведены стандартные коды, возвращаемые в CIO в регистр Y. В следующем разделе описано взаимодействие CIO с драйвером устройства для каждого вектора из таблицы векторов устройства.

Инициализация устройства.

Замечание: Данная входная точка не влечет выполнения какой-либо функции по отношению к нерезидентным устройствам ввиду ошибки настоящей ОС - таблица векторов очищается всякий раз при перезагрузке системы и при включениии питания. Тем самым теряется возможность вызова этой входной точки. Структура допускает совместимость с возможной версией ОС, лишенной в дальнейшем этого недостатка. Вызов входной точки должен был осуществляться всякий раз при включении питания и перезагрузке системы, чтобы дать возможность устройству провести инициализацию его аппаратных средств и данных в ОЗУ, используя программу, гарантирующую правильную обработку всех последующих команд CIO.

Поддерживающие функции.

В данном разделе приведено описание функций, взаимодействующих с первыми шестью векторами таблицы векторов устройства. Также в данном разделе приведено краткое, не зависящее от устройства, описание взаимодействия устройства с CIO и рекомендуемые действия для каждого вектора.

OPEN

Драйверу устройства необходимо установить параметры OPEN и выполнить любую необходимую инициализацию, связанную с OPEN.
X = индекс начального IOCB.
Y = 192 (состояние = функция не выполнена устройством).
ICDNOZ (0021) = номер устройства (1-4 для нескольких устройств).
ICBALZ/ICBAHZ (0024/0025) = адрес спецификации имен устройств/файлов.
ICAX1Z/ICAX2Z (002A/002B) = специальная информация об устройстве.
Устройство пытается выполнить указанную операцию OPEN и помещает код возврата в регистр Y. Ответственность за проверку правильности многократного использования OPEN по отношению к одному и тому же устройству или файлу лежит на драйвере устройства.

CLOSE

Драйверу устройства необходимо освободить все занимаемые ресурсы, относящиеся непосредственно к данному имени устройства/файла:
1. Послать все оставшиеся данные.
2. Отметить конец файла.
3. Скорректировать соответствующие директории, схемы размещения и т.п.

Представляют интерес следующие параметры входной точки устройства:
X = индекс начального IOCB.
Y = 192 (состояние = функция не выполнена устройством).
ICDNOZ (0021) = номер устройства (1-4 для нескольких устройств).
ICAX1Z/ICAX2Z (002A/002B) = Специальная информация об устройствах.
Устройство пытается выполнить указанную операцию CLOSE и помещает код возврата в регистр Y. CIO освобождает соответствующий IOCB независимо от результата операции.

GET BYTE

Драйверу устройства необходимо возвратить один байт в регистр A или установить код возврата в регистр Y.
X = индекс начального IOCB.
Y = !92 (состояние = функция не выполнена устройством).
ICNOZ (0021) = номер устройства (1-4 для нескольких устройств).
ICAX1Z/ICAX2Z (002A/002B) = специальная информация об устройстве).
Устройство управления получает байт данных непосредственно от устройства или поддерживаемого устройством буфера, после чего производит возврат в CIO с байтом в регистре A и помещает код возврата в регистр Y. Считывающие устройства, для которых продолжительность операции велика (клавиатура или кассетный магнитофон) должны проверять флаг клавиши (BREAK) BRKKEY (0011) и устанавливать состояние !80 при возникновении условия нажатия клавиши (BREAK). Управление клавишой (BREAK) обсуждается в приложении L, E5 и разделе 12. CIO следит за тем, чтобы не произошло вызова устройства при попытке осуществить чтение из неоткрытого или открытого только для вывода устройства/файла.

PUT BYTE

Драйвер принимает в регистр один байт или возвращает ошибочное состояние в регистре Y.
X = индекс начального IOCB.
Y = !92 (состояние = функция не выполнена устройством).
A = байт данных.
ICDNOZ (0021) = номер устройства (1-4 для нескольких устройств).
ICAX1Z/ICAX2Z (002A/002B) = специальная информация об устройстве).
Драйвер пересылает байт данных к устройству или в поддерживаемый устройством буфер и возвращается в CIO с кодом возврата в регистр Y. Если буфер, поддерживаемый устройством полон, то буферизованные данные перед возвратом в CIO отсылаются к устройству. CIO следит за тем, чтобы вызов устройства не осуществлялся при попытке записи в неоткрытый или открытый только для ввода устройство/файл. Когда обычная операция PUT BYTE определена, следует рассмотреть специальный случай: любое устройство взаимодействует с интерпретатором языка. ATARI Бейсик подчиняется различным набором правил. Поскольку в Бейсике можно осуществить непосредственный вызов входной точки устройства PUT BYTE без использования CIO, вызов PUT BYTE может быть осуществлен и без помощи ZIOCB. Поэтому устройству необходимо использовать IOCB более высокого уровня с тем, чтобы получить любую информацию, которую обычно можно получить из ZIOCB. Заметим, что защита OPEN, обычно обеспечиваемая CIO, в данном случае игнорируется (т.е. допускается операция PUT BYTE по отношению к неоткрытому или открытому только для ввода устройству/файлу).

GET STATUS

Подразумевается, что устройство возвращает четыре байта состояния в память или возвращает ошибочное состояние в регистр Y.
X = индекс начального IOCB.
Y = !92 (состояние = функция не выполнена устройством).
ICDNOZ (0021) = номер устройства (1-4 для нескольких устройств).
ICBALZ/ICBAHZ (0024/0025) = адрес спецификации имен устройств/файлов.
ICAX1Z/ICAX2Z (002A/002B) = специальная информация об устройстве.
Драйвер получает от устройства информацию о состоянии и размещает байты состояния в ячейках с DVSTAT (02EA) по DVSTAT+3, а затем возвращается в CIO

с кодом в регистре Y. IOCB может быть как открыт, так и закрыт в порядке запроса CIO на выполнение операции GET STATUS, причем устройство управления должно следить за соблюдением ограничений. Действия CIO, включающие операцию GET STATUS с открытым и закрытым IOCB и влияние этой операции на параметр адреса буфера описаны в разделе 5.

SPECIAL

Данная входная точка служит для поддержки всех функций, невыполняемых остальными входными точками, такими как переименование файла на дискете (RENAME), рисунка на дисплее (DRAW) и т.п. Обычно CIO производит вызов входной точки SPECIAL, в случае, когда величина командного байта IOCB превосходит $0D. Драйвер должен анализировать командный байт, чтобы определить возможно ли выполнение запрошенной функции.
X = индекс начального IOCB.
Y = $92 (состояние = операция не выполнена устройством).
ICDNOZ (0021) = номер устройства (1-4 для нескольких устройств).
ICCOMZ (0022) = командный байт.
ICBALZ/ICBAHZ (0024/0025) = адрес буфера.
ICBLLZ/ICBLHZ (0028/0029) = длина буфера.
ICAX1Z/ICAX2Z (002A/002B) = специальная информация об устройстве.
Устройство выполняет указанную операцию и возвращает управление CIO с кодом возврата в регистре Y. Запрос CIO на выполнение операции SPECIAL не требует открытия или закрытия IOCB, но устройство обязано проследить за тем, чтобы никакие ограничения не были нарушены. Действия CIO, включающие операции SPECIAL с открытым и с закрытым IOCB и влияние этих операций на параметр адреса буфера, описаны в разделе 5.

Обработка ошибок.

Обработка ошибок существенно облегчается тем обстоятельством, что CIO обрабатывает ошибки внешнего уровня, а SIO обрабатывает ошибки последовательной шины, оставляя обработку оставшихся ошибок устройству. К этим ошибкам относятся.
- выход параметров из диапозона,
- прерывание от клавиши (BREAK),
- неправильная команда,
- чтение после конца файла.

В основу реакции устройств на ошибки положены следующие принципы:
- Сохраняется простота возврата в исходное состояние (а тем самым предсказуемость и неоднородность). Команды к возврату не подаются непосредственно пользователем. Теряется минимальный возможный обьем данных. Производятся попытки сохранить целостность памяти, связанной с устройством - это включает в себя первоначальное размещение структурных элементов после возврата из ошибочного состояния.

Распределение ресурсов.

Технология, приведенная ниже должна быть использована нерезидентными устройствами, которым необходим код и/или область данных в ОЗУ с тем, чтобы гарантировать отсутствие конфликтов с другими частями ОС, включая остальные нерезидентные устройства.

ОЗУ нулевой страницы.

Нулевая страница ОЗУ не содержит запасных байт, и даже если бы такие байты имелись, то возникли бы трудности с их использованием, поскольку не

существует схемы распределения свободных байт между присваиванием различных программ. Поэтому те байты нулевой страницы, которые будут задействованы нерезидентным устройством должны сохранятся и восстанавливатся. Байты, которые следует использовать необходимо тщательно отбирать при помощи следующего критерия:
- Байты недоступны программе прерываний.
- Байты недоступны любой непрерывающей программе в момент времени между изменением устройством байт и последующим восстановлением первоначальных величин.

Простейшая технология сохранения/восстановления использует стек примерно следующим образом:

```
 LDA COLCRS      ;(например)
 PHA             ;запомнить в стеке
 LDA COLCRS+1
 PHA
 LDA HPOINT      ;указатель на устрой-во
 STA COLCRS
 LDA HPOINT+1
 STA COLCRS+1
 XXX (COLCRS),Y
 PLA             ;восстановление старых
 STA COLCRS+1    ;данных
 PLA
 STA COLCRS
```

Заметим, что не нужно вызывать ни дисплей, ни экранный редактор перед восстановлением начальной величины COLCRS, поскольку COLCRS является переменной, используемой данными программами.

ОЗУ ненулевой страницы.

Не существует какой-либо схемы размещения присваиваний в фиксированных областях ОЗУ ненулевой страницы для какого-либо определенного процесса. У устройства имеется три возможности:
1. Провести динамическое распределение в момент инициализации, изменяя MEMLO (02E7).
2. Включить переменные устройств для резидентных устройств в ОЗУ. В данном случае подразумевается изменение MEMLO в процессе загрузки устройства.
3. Если устройство подключено вместо одного из резидентных устройств (за исключением входной точки резидентного устройства из таблицы устройств), то новое устройство может использовать любую, доступную ранее, резидентному устройству ячейку ОЗУ.

Область стека.

В большинстве случаев на использование устройством стека не наложено никаких ограничений.

Взаимодействие устройства с SIO.

В данном разделе описывается взаимодействие между устройствами последовательной шины и программой ввода/вывода через последовательную шину. SIO управляет всеми преобразованиями шины в полном соответствии с независимым от устройств протоколом шины. SIO отвечает за выполнение следующих функций:
- форматирование данных шины и синхронизация с окончанием работы компьютера.
- обнаружение ошибок, повторные попытки и состояния.
- простой шины.
- перемещение данных между шиной и буфером пользователя.

Механизм вызова.

SIO для всех операций имеет единственную входную точку SIOV (E459). Все параметры, необходимые SIO, хранятся в блоке управления устройством (DCB). В DCB входят следующие байты:

ID устройства шины -- DDEVIC (0300)

ID устройства шины устанавливается устройством перед вызовом SIO.

Номер устройства -- DUNIT (0301)

Значение данного байта устанавливается перед вызовом SIO, и указывает к какому из устройств данного типа открыт доступ. Данная величина, как правило, приходит из ICDNOZ. SIO осуществляет доступ к тем устройствам шины, адреса которых равны величине DDEVIC + значение величины DUNIT минус 1. (Наименьший номер устройства равен единице.)

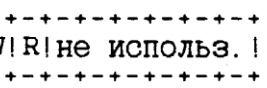
Команда устройству -- DCOMND (0302)

Устройство устанавливает значение одного байта перед вызовом SIO. Он отсылается на устройство шины как часть командного фрейма. Величины командных байтов устройства можно найти в приложении I.

Состояние устройства -- DSTATS (0303)

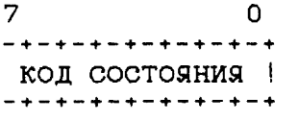
Данный байт является двунаправленным, и используется устройством для указания SIO, какие действия необходимо выполнить после получения и поддерживания командного фрейма. Этот байт будет в дальнейшем использован SIO для определения состояния запрашиваемой операции.

До вызова SIO:



Где W, R = 0, 0 означает, что операция не связанна с переносом данных.
0, 1 означает, что ожидается получение данных от устройства.
1, 0 означает, что ожидается передача данных к устройству.
1, 1 - ошибочное состояние.

После вызова SIO:



Коды возможных состояний операций SIO можно найти в приложении C.

Адрес буфера устройства -- DBUFLO/DBUFHI (0304/0305)

Двухбайтовый указатель устанавливается устройством. Он указывает на исходный буфер или буфер назначения для данных устройства или информации о состоянии.

Простой устройства -- DTIMLO (0306)

Значение этого байта устанавливается устройством. Данный байт определяет время простоя устройства в 64/60 долях секунды. Например, значение 6 означает простой длительностью 6.4 секунды.

Счетчик длины буфера в байтах -- DBYTLO/DBYTHI (0308/0309)

Значение данного двухбайтого счетчика устанавливается устройством для текущей операции и указывает на число байтов данных, переносимых в буфер или из буфера. Данный параметр не является необходимым в случае, если в байте состояния (STATUS) биты W и R равны нулю. Такая ситуация означает, что перенос данных осуществлятся не будет.

Предупреждение: В программе SIO имеется ошибка, приводящая к непредсказуемым результатам, если последний байт буфера имеет адрес в памяти, заканчивающийся !FF (как например, 13FF, 42FF и т.п.).

Вспомогательная информация -- DAUX1/DAUX2 (030A/030B)

Значения этих двух байтов устанавливаются устройством. SIO включает их в состав командного фрейма, помещая в них специальную информацию об устройстве.

Поддерживающие функции.

SIO не производит исследование командного байта, посылаемого к устройству, поскольку все преобразования в шине сводятся в единый протокол. Протокол имеет одну из трех приведенных ниже форм (как это видно из компьютера):
- послать командный фрейм,
- послать командный фрейм и фрейм данных,
- послать командный фрейм и получить фрейм данных.

Форма команды задается значениями битов W и R в байте состояния.

Обработка ошибок.

Большинство ошибок в последовательной шине обрабатываются SIO следующим образом:

Простой шины -- SIO обеспечивает единый простой командного фрейма и байта ACK фрейма данных длительностью 1/60 секунды. Величина максимального простоя байта COMPLETE определена устройством в DTIMLO.

Ошибки шины -- SIO отслеживает и сообщает VART ошибки во фреймах и ошибки выхода за границы. Опознование этих ошибок в любом из полученных байтов приведет к тому, что соответствующий внутренний фрейм будет считатся плохим.

Ошибка контрольной суммы фрейма -- SIO вычисляет контрольную сумму для всех перенесенных фреймов.

Неверный отклик от устройства -- В дополнение к указанным выше ошибочным условиям SIO проверяет правильность значений откликов ACK и COMPLETE (ACK=!41 и COMPLETE=!43). ACK означает подтверждение получения.

Повторная попытка операции с шиной -- SIO произведет повторный запуск полной команды в случае, если после первичного запуска были обнаружены ошибки. Запуск полной команды состоит из четырнадцати попыток послать (и распознать) командный фрейм с последующей попыткой получить код COMPLETE и по возможности фрейм данных.

Замечание: В логике повторного запуска записи данных имеется ошибка, заключающаяся в том, что если командный фрейм зафиксирован контроллером, а фрейм данных не зафиксирован, повторный запуск не определен.

Объединенные коды ошибочных состояний -- SIO генерирует не зависящий от устройств коды ошибок (см. Приложение С).

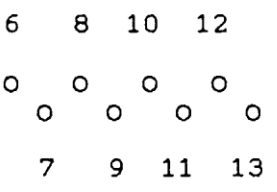
Характеристики и протокол последовательной шины ввода/вывода.

В данном разделе будут описаны следующие аспекты:
- Электрические характеристики последовательной шины ATARI XL, XE,
- Использование шины для передачи байтов данных.
- Организация байтов во фреймы (записи).
- Общие последовательности команд, использующие фреймы и байты отклика для обеспечения связи компьютера с периферийными устройствами.

Электрические характеристики аппаратных средств.

ATARI XL, XE связываются с периферийными устройствами через асинхронные последовательные порты, со скоростью передачи до 19200 бод. Последовательный порт состоит из последовательной шины DATA OUT (передача), последовательной шины DATA IN (получение), а также различных других управляющих линий. Данные передаются и принимаются в виде последовательности из восьми битов (младший бит посылается первым), которым предшествует логический ноль, фиксирующий начало. Логическая единица фиксирует конец последовательности бит.

Последовательная шина DATA OUT передается в виде положительного логического сигнала (+4В = единица/истина/высокий уровень сигнала, 0В = ноль/ложь/низкий уровень сигнала). Последовательная шина DATA OUT обычно принимает новое состояние, когда последовательная шина CLOCK OUT имеет низкий уровень сигнала. Ниже приведен вид сверху разъема последовательной шины на компьютере или периферийном устройстве:



Где: 1 = CLOCK IN компьютера.
2 = CLOCK OUT компьютера.
3 = DATA IN компьютера.
4 = GND (земля).
5 = DATA OUT компьютера.
6 = GND (земля).
7 = COMMAND (командная шина).
8 = управление мотором.
9 = PROCEED (действующая шина).
10 = +5В/READY (готовность).
11 = AUDIO IN компьютера.
12 = +12В.
13 = INTERRUPT (прерывание).

CLOCK IN не используется настоящей ОС и периферийными устройствами. Данная линия может быть использована в будущем для построения синхронных схем связи.
CLOCK OUT - синхронизация последовательной шины. Уровень сигнала CLOCK OUT повышается в начале каждого бита DATA OUT и опускается в середине каждого бита.
DATA IN - последовательная шина данных, идущая к компьютеру.
Штырек 4 GND - является экранирующим проводом заземления.
DATA OUT - последовательная шина данных, идущая от компьютера. Штырек 6 GND является экранирующим проводом заземления.
COMMAND - обычно высокий уровень сигнала, который снижается при посылке с компьютера командного фрейма.
Управление мотором (MOTOR CONTROL) - шина, управляющая мотором кассетного магнитофона (высокий уровень = мотор включен, низкий уровень = мотор выключен).
PROCEED - не используется настоящей ОС и периферийными устройствами. На нее подается высокий уровень сигнала.
+5В/READY указывает, что включен и готов к работе. Данная линия может также быть использована как источник питания в +5В для периферийных устройств с номинальным током 50 мА.
AUDIO IN принимает аудио сигнал с кассеты.
+12В представляет собой источник питания для периферийных устройств ATARI с ненормированным током.
INTERRUPT не используется настоящей ОС и периферийными устройствами. На эту шину подается высокий сигнал.
В кабеле последовательной шины никаких отклонений от вышеуказанной схемы не существует. Тем самым, штырек 3, соответствующий шине компьютера DATA IN соответствует выходной шине периферийного устройства. Штырек 5 действует наоборот.

Электрические характеристики последовательного порта.

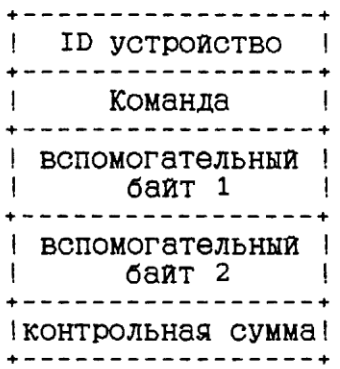
Ввод с периферийного устройства
V1H=2.0В MIN
V1L=0.4В MAX
I1H=20мА MAX V1H=2.0В
I1L=5мА MIN V1L=0.4В
Вывод на периферийное устройство (открыт коллекторный переход биполярного транзистора).
V0L=0.4В MAX 1.6мА
V0H=4.5В MIN с внешним сопротивлением 100кОм
Ввод с Vcc/READY.
V1H=2.0В MIN I1H=1мА MAX
V1L=0.4В MAX
При открытии ввод опускается до логического нуля.

Команды шины.

Протокол шины требует, чтобы все команды исходили из компьютера, и чтобы все периферийные устройства предоставляли данные только после соответствующей команды. Действия шины представляют собой следующую последовательность операций:
- получение командного фрейма от компьютера.
- подтверждение (ACK) периферийного устройства.
- пересылка выбранного фрейма данных к компьютеру или из компьютера.
- завершение операции (COMPLETE) периферийным устройством.

Командный фрейм.

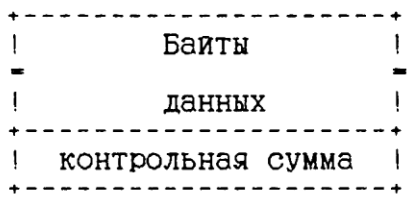
Протокол последовательной шины может работать с тремя типами команд: 1) передача данных, 2) получение данных, 3) мгновенными (без пересылки данных - только команда). Всем трем типам присущ общий элемент - командный фрейм, состоящий из пяти байт информации, полученной с компьютера в то время, когда на линии COMAND был низкий уровень сигнала. Командный фрейм имеет следующий формат:



Устройство ID определяет адресация какого из устройств последовательной шины имеет место (см. Приложение I, где можно найти список ID устройств). Командный байт содержит команду, зависящую от устройства (список команд устройств можно найти в приложении I). Вспомогательные байты содержат информацию, зависящую от устройств. Байт контрольной суммы содержит арифметическую сумму первых четырех байтов (с добавлением переноса после каждого сложения).

Подтверждение командного фрейма.

Адресуемое периферийное устройство обычно отвечает на командный фрейм, отсылая байт ACK (:41) к компьютеру. В случае, если командный фрейм задан неправильно периферийное устройство не отвечает. За командным фреймом (и ACK) может следовать фрейм данных следующего формата:



В зависимости от команды данный фрейм данных может проистекать из компьютера или контроллера устройства. Текущие контроллеры устройств ожидают наличия фреймов данных фиксированной длины, как и компьютер, где длина фреймов данных представляет собой фиксированную функцию от типа устройства и команды. Величина контрольной суммы во фрейме данных представляет собой арифметическую сумму всех предшествующих контрольной сумме байтов данных с добавлением переноса после каждого сложения (также как и для командного фрейма). В случае, если компьютер отсылает фрейм данных к периферийному устройству, оно должно послать ACK, если оно в состоянии принять фрейм данных. Если же периферийное устройство не может принять фрейм данных, оно отсылает NAK (:4E) или не производит никаких действий. (см. первую блок схему раздела 9).

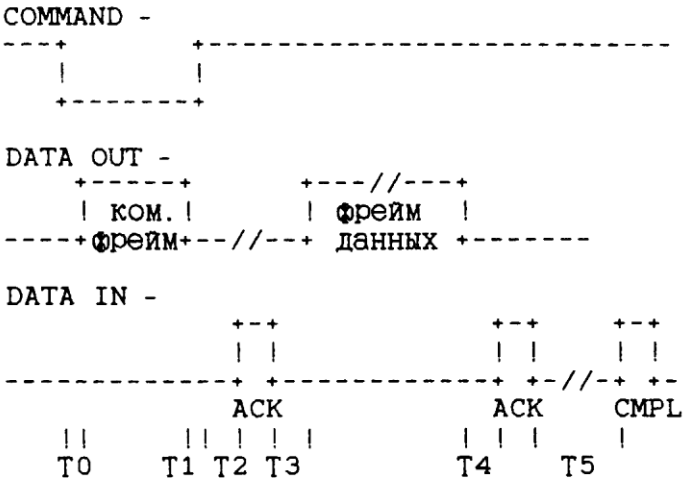
Завершение операции.

По завершении запрашиваемой операции периферийное устройство должно переслать байт завершения операции (COMPLETE (:43)). Местоположение этого байта в последовательности команд указано на временных диаграмах раздела 9 для каждого типа команды. Если нормальное, безошибочное завершение операции невозможно, периферийное устройство должно выдать байт ошибки (ERROR (:45)) вместо COMPLETE.

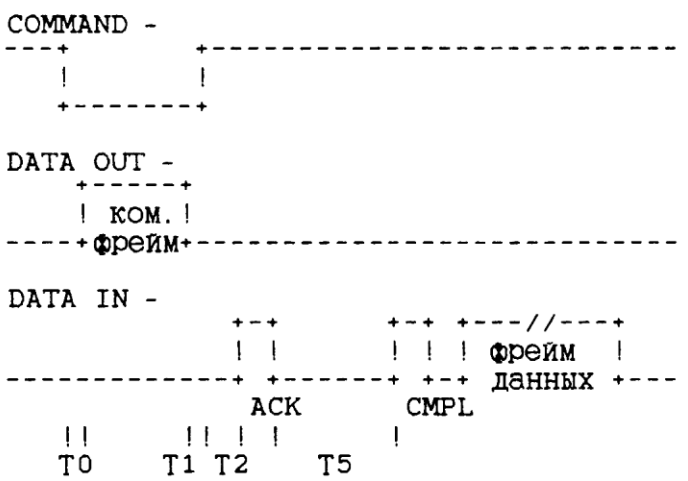
Временная диаграмма шины.

В данном разделе приведены временные диаграммы для трех типов последовательностей команд: пересылки данных, получения данных и управление.

Последовательность пересылки данных:



Последовательность получения данных:



Управляющая последовательность:

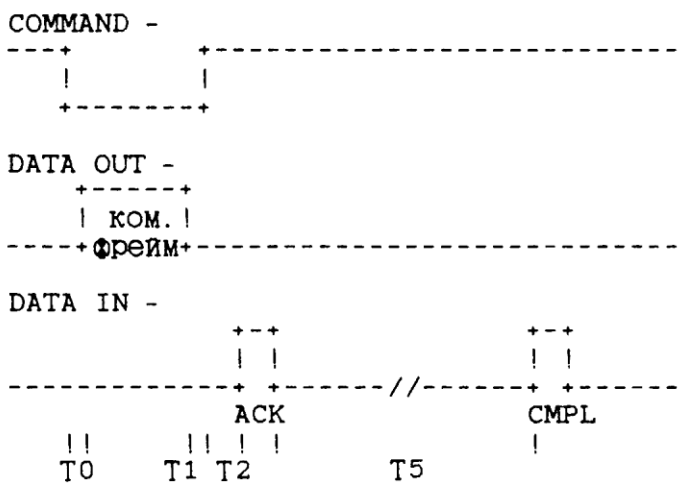


Рис. 9-6 Временные диаграммы последовательной шины.

Компьютер производит задержку (T0) между моментами снижения сигнала на COMMAND и передачи первого байта данных командного фрейма.
T0 компьютера (MIN) = 650 мкс
T0 компьютера (MAX) = 950 мкс
T0 периферийного устройства (MIN) = ??
T0 периферийного устройства (MAX) = ??
Компьютер производит задержку (T1) между моментами передачи последнего бита командного фрейма и повышения сигнала на шине COMMAND.
T1 компьютера (MIN) = 650 мкс
T1 компьютера (MAX) = 950 мкс
T1 периферийного устройства (MIN) = ??
T1 периферийного устройства (MAX) = ??
Периферийное устройство производит задержку между моментами повышения сигнала на COMMAND и передачи байта ACK периферийным устройством.
T2 компьютера (MIN) = 0 мкс
T2 компьютера (MAX) = 16 мкс
T2 периферийного устройства (MIN) = ??
T2 периферийного устройства (MAX) = ??
Компьютер производит задержку (T3) меджу моментами получения последнего бита байта ACK и передачи компьютером первого бита фрейма данных.
T3 компьютера (MIN) = 1000 мкс
T3 компьютера (MAX) = 1800 мкс
T3 периферийного устройства (MIN) = ??
T3 периферийного устройства = ??
Периферийным устройством генерируется задержка (T4) между моментами передачи последнего бита фрейма данных и получением компьютером первого бита байта ACK.
T4 компьютера (MIN) = 850 мкс
T4 компьютера (MAX) = 16 мс
T4 периферийного устройства (MIN) = ??
T4 периферийного устройства (MAX) = ??
Периферийным устройством генерируется задержка (T5) между моментами получения компьютером последнего бита ACK и первого бита байта COMPLETE.
T5 компьютера (MIN) = 250 мкс
T5 компьютера (MIN) = 255 мкс (зависит от устройства)
T5 периферийного устройства (MIN) = ??
T5 периферийного устройства (MAX) = N/A

Внешняя среда устройств.

Подсоединение нерезидентных устройств может осуществлятся по крайней мере тремя способами:
1. Программой, загружаемой с дискеты или кассеты.
2. Резиденцией в картридже (A или B).
3. Загрузкой с устройства последовательной шины.

В данном разделе будут обсуждаться основные механизмы подключения устройств для каждой из этих внешних сред. Чтобы полностью осознать информацию, содержащуюся в этом разделе необходимо перечитать и понять следующие разделы:

Загружаемые программы.

Указатель на таблицу векторов и имя будут вставлены в таблицу устройств программой, загружаемой с дискеты или кассеты сразу после того, как произошла инициализация входной точки загружаемой программы (при включении питания и перезагрузке). Необходимо помнить, что при включении питания и перезагрузке системы таблица устройств очищается от всего, кроме входных точек резидентных устройств.

Резидентные устройства в картридже.

Указатель на таблицу векторов устройства и имя вставляются программой картриджа в таблицу устройств сразу после инициализации входной точки картриджа (при включении питания и перезагрузке системы). Необходимо помнить, что при включении питания и при перезагрузке системы таблица устройств очищается от всего, кроме входных точек резидентных устройств. По этой причине таблица устройств должна каждый раз быть заново установлена процедурой инициализации устройства после каждого ввода.

Блок-схемы.

На следующих страницах приведены блок-схемы, иллюстрирующие последовательный ввод/вывод и действия периферийных устройств для различных форм команд последовательной шины.

Обработка командного фрейма периферийным устройством.

1. Начало.
2. Ожидать понижения уровня сигнала на COMMAND.
3. Получить с шины следующие 5 байт, если ошибка, то вернуться к пункту 2.
4. Ожидать повышения уровня COMMAND.
5. Проверить правильность контрольной суммы, если она неправильна, то перейти к пункту 2.
6. Проверить командный фрейм для этого устройства. Если нет, то перейти к пункту 2.
7. Проверить правильность команды, если команда правильна, то перейти к пункту 9.
8. Послать NAK и перейти к пункту 2.
9. Проверить правильность вспомогательных данных, если они неправильны, то перейти к пункту 8.
10. Послать ACK.
11. Перейти к пунктам 12 или 21, или 27.

Пересылка фрейма данных к периферийному устройству.

12. Перейти к считыванию фрейма данных.
13. Получить N байт с шины, если простой, то перейти к пункту 1.
14. Проверить правильность контрольной суммы, если она правильна, то перейти к пункту 16.
15. Послать ACK и перейти к пункту 1.
16. Послать NAK.
17. Попытаться выполнить указанную операцию.
18. Проверить прошла ли операция, если да ,то перейти к пункту 20.
19. Послать сообщение об ошибке и перейти к пункту 1.
20. Послать завершение COMPLETE.

Пересылка фрейма данных к компьютеру.

21. Попытка выполнить указанную операцию.
22. Проверить прошла ли операция, если да, то перейти к пункту 24.
23. Выдать сообщение об ошибке и перейти к пункту 1.
24. Послать завершение COMPLETE.
25. Послать фрейм данных.
26. Перейти к пункту 1.

Управляющая команда.

27. Попытаться выполнить указанную операцию.
28. Проверить верна ли операция, если да перейти к пункту 31.
29. Выдать сообщение об ошибке и перейти к пункту 1.
30. Послать завершение и перейти к пункту 1.

10. Внешняя среда программ и инициализация.

В данном разделе обсуждаются различные альтернативные операционные среды на основе конфигурации ОС. Нерассмотренные здесь операционные среды имеют право на существование. Для того, чтобы правильно оценить все альтернативные операционные среды, необходимо ясно понимать сущность процессов, происходящих при включении питания и перезагрузке системы.

Картридж.

Большинство игр (а также некоторые трансляторы) поддерживаются операционной средой картриджа. Резидентные программы картриджа находятся под управлением системы с использованием ОС или без нее. Картридж может определить необходима ли загрузка с дискеты при включении питания, имеется ли картридж с управляющей программой, или является ли картридж специальным диагностическим картриджем. Данным опциям поставлены в соответствие биты в головной части картриджа, как показано ниже.

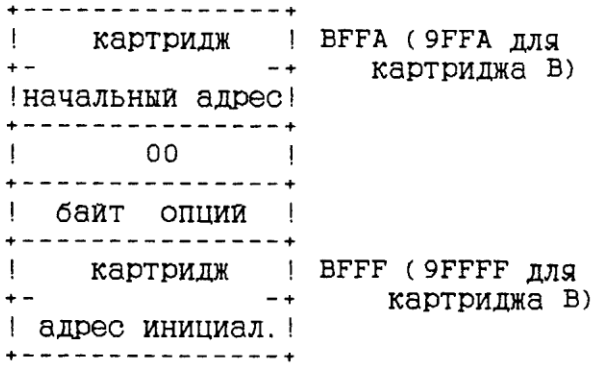


Рис. 10-1 Формат головной части картриджа.

Байт 00 используется, чтобы дать возможность ОС определить вставлен картридж или нет. В ячейках BFFC и 9FFC не будет считан ноль, если эти ячейки заняты ОЗУ, или картридж в гнезде отсутствует. Область ОЗУ отличается от области картриджа своей способностью изменять размеры. Биты байта опций несут следующую информацию.

Бит 0 = 0 - не производить загрузку с дискеты.
1 - произвести загрузку с дискеты.
Бит 2 = 0 - инициализировать, но не запускать картридж.
1 - инициализировать запуск и запустить картридж.
Бит 7 = 0 - если картридж не диагностический.
1 - картридж диагностический и управление будет передано картриджу до инициализации любой из ОС (JMP (BFFE)). Адрес инициализации картриджа определяет ячейку, в которую ОС произведет переход JSR во время включения питания и перезагрузки системы. Как минимум этот вектор должен указывать на команду RTS. Начальный адрес определяет ячейку, в которую ОС произведет переход JMP при включении питания или при перезагрузке системы в случае, если бит 1 байта опций равен 1. Прикладной программе необходимо исследовать переменную WARMST (0008), в случае, если действия системы при включении питания отличаются от действий при перезагрузке системы. (Значение WARMST равно нулю при включении питания и отлично от нуля в любом другом случае).

Картридж без дополнительной загрузки.

Картриджи, в которых отсутствует опция загрузки с дискеты или с кассеты занимают меньший обьем памяти (с 0480 до адреса в MEMTOP (02E5)).

Картридж с загрузки.

Под картридж, в котором присутствует опция загрузки с дискеты или, поддерживающая загрузку с кассеты, небольшой обьем памяти должен быть распределен достаточно аккуратно. Определены следующие области:
0480-06FF - всегда доступны картриджу.
MEMLO/MEMTOP - область, всегда доступная картриджу.

Программы, загружаемые с дискеты.

Программа может быть загружена с дисковода во время включения питания в ответ на одно из следующих условий:
- Ни картридж А, ни В не вставлены.
- Картридж А вставлен и нулевой бит байта опций (BFFD) равен единице.
- Картридж В вставлен и нулевой бит байта опций (9FFD) равен единице.

Столкнувшись с одним из этих условий ОС попытается считать запись загрузки из сектора 1 дисковода 1, а затем передать управление считанной программе. Точная последовательность операций будет описана позже в данном разделе.

Формат файла, загружаемого с дискеты.

Первые шесть байт, формат которых показан ниже, являются ключевой областью файла, загружаемого с дискеты.

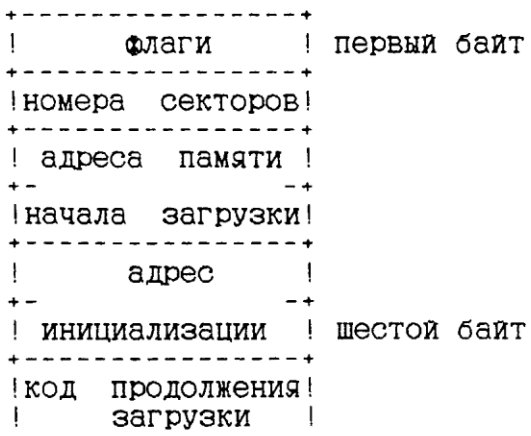


Рис. 10-2 Формат файла, загружаемого с дискеты.

Первый байт располагается в DFLAGS (0240), но в других отношениях не используется. Он должен быть равным нулю. Второй байт содержит число 128-байтовых секторов на дискете, которые необходимо считать в процессе загрузки (включая запись, содержащую эту информацию). Это число может изменятся от 1 до 255, где 0 означает 256. Третий и четвертый байты содержат адрес (младший, старший), по которому загрузчик передает управление после завершения загрузки и при нажатии клавиши (RESET). Дискета с системой управления файлами имеет дополнительные байты в записи загрузки, но этот случай является исключением из общего случая и описан в разделе 5.

Процесс загрузки с дискеты.

Если ни одного картриджа не установлено, загрузка осуществляется выполнением следующих шагов:
1. Считывание первой записи с дискеты в кассетный буфер (0400).
2. Извлечение информации из первых шести байт:
Запомнить байт флагов в DFLAGS (0240,1). Запомнить номера загружаемых секторов в DBSECT (0241,1). Запомнить адрес загрузки в BOOTAD (0242,2). Поместить адрес инициализации в DOSINI (000C,2).
3. Поместить считанную запись по определенному адресу загрузки.
4. Считать оставшиеся записи непосредственно в область загрузки.
5. Переход JSR по адресу загрузки +6, где возможно продолжение многоступенчатого процесса загрузки. Бит переноса характеризует успех этой операции (установленный перенос = ошибка, нулевой перенос = успешное завершение).

Замечание: Во время шага 5 после завершения первоначального процесса загрузки загрузчик передает управление седьмому байту первой записи. Программа должна далее сама продолжить процесс загрузки, если процесс многоспупенчатый. Величина MEMLO (02E7) должна указывать на первую после загруженной программы свободную ячейку ОЗУ. Ее значение должно быть установлено загруженной программой следующим образом:

```
LDA #END+1      ;установить младший
                ;байт
STA MEMLO
STA APPMHI
LDA #END+1/256  ;установить старший
STA MEMLO+1     ;байт
STA APPMHI+1
```

Если передача управления загружаемой программе осуществляется в конце процесса загрузки, прикладной программой к этому моменту должно быть установлено значение вектора DOSVEC (000A). DOSVEC указывает на вход в прикладную программу при повторном старте. Если передача управления загружаемой программе не происходит, DOSVEC остается без изменений.

```
     LDA <RESTRT  ;младший байт RESTART
     STA DOSVEC
     LDA >RESTRT/256
     STA DOSVEC+1
```

6. Косвенный переход JSR через DOSINI для инициализации прикладной программы. Происходит инициализация прикладной программы и возврат из нее.
Замечание: ОС осуществляет переход в точку инициализации при перезакрузке системы и при включении питания. При включении питания и перезагрузке системы может иметь место внутренняя инициализация. Для управляющих прикладных программ инициализация может быть отложена до шага 7.
7. Косвенный переход JMP через DOSVEC для передачи управления прикладной программе.
Замечание: Нажатие клавиши (RESET) после полной загрузки прикладной программы вызовет повторное выполнение шагов 6 и 7.

Образец листинга загружаемой с дискеты программы.

Данная программа может быть загружена с дискеты. При входе в нее ей передается управление.

```
; Начало файла программы
PST  =$0700 ;или любая другая ячейка
`=   PST    ;(начало)
; Информация для управления загрузкой с дискеты
     .BYTE 0
     .BYTE PND-PST+127/128;число записей
     .WORD PST    ;адрес начала загрузки
     .WORD PINIT  ;инициализация прогр.
; Начало продолжения загрузки
     LDA <PND     ;установить нижние
     STA MEMLO    ;границы памяти
     STA APPMHI
     LDA >PND/256
     STA MEMLO+1
     STA APPMHI+1
     LDA <RESTRT  ;установить вектор
     STA DOSVEC   ;повторного старта
     LDA >RESTRT/256
     STA DOSVEC+1
     CLC          ;установить значение
     RTS          ;флага, свидетельств.
                  ;об успешной загрузке
; Входная точка инициализации прикладной программы
PINIT RTS         ;не совершать никаких
                  ;действий для
                  ;управляющей
                  ;прикладной программы
; Далее следует основное тело програмы
RESTRT=`
; Конец основного тела программы
PND=   `          ;"PND" - следующая
     .END         ;свободная ячейка
```

Рис. 10-3 Пример листинга программы, загружаемой с дискеты.

Программа, создающая файлы, загружаемые с дискеты.

Ниже приведен листинг программы, создающей файлы, загружаемые с дискеты.

```
; Данная программа записывает
; единственный файл на дискету и
; используется совместно с процедурой
; получения загружаемых с дискеты
; файлов. Следующим двум символам
; должны быть присвоены значения при
; помощи границ памяти программы,
; которые надлежит скопировать.
; "PST" = начальный адрес программы.
; "PND" = конечный адрес программы.
;
SECSIZ=128      ;размер сектора дискеты
PST=   !0700
PND=   !1324
FLEN=  PND-PST+SECSIZ-1/SECSIZ ;число
                ;секторов в файле
`=     !B000    ;адрес начала программы
BOOTB  BRK      ;загрузка прикладной
                ;прграммы
; Для вызова дисковода установить блок
; управления устройством.
       LDA {FLEN   ;число секторов.
       STA COUNT   ;которые необходимо
                   ;записать
       LDA {1      ;дисковод 1
       STA DUNIT
       LDA "W      ;установить запись с
       STA DCOMND  ;с проверкой
       LDA {PST    ;указатель на начало
       STA DBUFLO  ;прикладной программы
       LDA {PST/256
       STA DBUFHI
       LDA {01     ;установить начальный
       STA DAUX1   ;номер сектора равным
       LDA {00     ; !001
       STA DAUX2
 ; Теперь запишем файл по секторам.
BOT010 JSR DSKINV ;записать один сектор
       BMI DERR    ;ошибка
       LDA DBUFLO  ;увеличить адрес
       CLC         ;памяти
       ADC {SECSIZ
       STA DBUFLO
       LDA DBUFHI
       ADC {0
       STA DBUFHI
       INC DAUX1   ;увеличить на 1 номер
       BNE BOT020  ;сектора
       INC DAUX2
BOT020 DEC COUNT ;есть еще незаписанные
                   ;сектора?
       BNE BOT010  ;да!
       BRK         ;остановить, если
                   ;выполнено
```

```
DERR    BRK             ;остановится при
                        ;ошибке
COUNT   `=`+1           ;счетчик секторов
; Головная часть картриджа.
`=      !BFF9           ;картридж A
INIT    RTS
        .WORD BOOTB
        .BYTE 0,4
        .WORD INIT
        .END
```

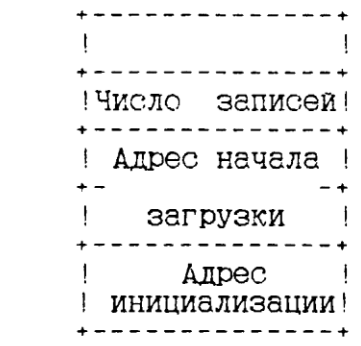
Программы, загружаемые с кассеты.

При включении питания можно произвести загрузку с кассеты также, как и с дискеты. Для загрузки с кассеты необходимо:
- нажать клавишу (START) при включении компьютера.
- В магнитофоне должна находится кассета с файлом отформатированным для загрузки и должна быть нажата клавиша (PLAY).
- После звукового сигнала нажать (RETURN).

Если все эти условия выполнены, ОС произведет чтение файла с кассеты и передаст управление считанной программе. Точная последовательность действий будет приведена далее в этом разделе.

Формат файла, загружаемого с кассеты.

Первые шесть байт файла на кассете имеют следующий формат:



В процессе загрузки с кассеты первый байт не используется. Второй байт содержит число 128-байтовых записей, которые необходимо считать в процессе загрузки (включая запись, содержащую данную информацию). Это число может изменяться от 1 до 255, причем 256 означает 0. Третий и четвертый байты содержат адрес (младший, старший), по которому загрузчик будет передавать управление после завершения процесса загрузки и всякий раз при нажатии (RESET).

Процесс загрузки с кассеты.

Ниже приводится пошаговое описание загрузки с кассеты с учетом того, что ни картридж, ни дисковод не подключены. Описание общего случая можно найти в разделе 7.

1. Считать первую запись на кассете в кассетный буфер.
2. Извлечь информацию из первых шести байт:
 Запомнить число загружаемых записей, запомнить адрес загрузки, запомнить адрес инициализации в CASINI (0002).
3. Поместить считанную запись по адресу загрузки.
4. Считать остальные записи непосредственно в область загрузки.
5. Совершить переход JSR по адресу равному адресу загрузки +6, т.е. туда, где возможно продолжение многоступенчатого процесса загрузки. Бит переноса указывает на успех данной операции (1 - ошибка, 0 - операция прошла успешно).
6. Совершить косвенный переход JSR через CASINI для инициализации прикладной программы. Произойдет инициализация прикладной программы и возврат из нее.
7. Совершить косвенный переход JMP через DOSVEC для передачи управления прикладной программе.

Нажатие клавиши (RESET) после полного завершения загрузки прикладной программы вызовет повторение пунктов 6 и 7.

Замечание: После завершения процесса первоначальной загрузки загрузчик передает управление седьмому байту первой записи. В этой точке программа должна продолжить процесс загрузки (в случае, если процесс многоступенчатый), а затем остановить работающий магнитофон, который работает до этого момента при помощи следующей последовательности команд:

```
LDA (13C
STA PACTL(D302)
```

Прикладная программа должна установить значение MEMLO (0237), указывающее на первую свободную ячейку ОЗУ после загруженной программы следующим образом:

```
LDA (END+1
STA MEMLO
STA APPMHI
LDA (END+1/256
STA MEMLO+1
STA APPMHI+1
```

В случае, если управление системой должно быть передано загруженной программе в конце загрузки, прикладная программа к этому моменту должна установить значение DOSVEC. DOSVEC указывает на входную точку повторного старта прикладной программы. Если управление не должно быть передано загруженной программе, значение DOSVEC остается без изменений.

```
LDA (RESTRT ;младший байт
STA DOSVEC  ;RESTART
LDA (RESTRT
STA DOSVEC+1
```

Замечание: В точку инициализации осуществляется переход при включении питания или перезагрузке системы. Также может иметь место внутренняя инициализация. Для управляющих прикладных программ инициализация может быть отложена до шага 7.

Образец листинга загружаемой с кассеты программы.

Ниже приведена программа, которая может быть загружена с кассеты, и которой передается при входе в нее управление.

```
; Начало файла программы.
PST=    !0700           ;либо любая другая
                        ;ячейка
`=      PST             ;(начало)
; Информация о загрзке с кассеты.
        .BYTE 0         ;не имеет значения
        .BYTE PND-PST+127/128 ;число
                        ;записей
        .BYTE PST       ;адрес начала
                        ;загрузки
        .BYTE PINIT     ;инициализация
                        ;программы
; Начало продолжения загрузки.
        LDA  {!3C       ;остановить магнит.
        STA PACTL
        LDA {PND        ;установить границы
        STA MEMLO       ;памяти
        STA APPMNI
        LDA {PND/256
        STA MEMLO+1
        STA APPMHI+1
        LDA {RESTRT     ;установить вектор
        STA DOSVEC      ;повторного старта
        LDA {RESTRT/256
        STA DOSVEC+1
        CLC             ;установить флаг в
        RTS             ;случае успешной
                        ;загрузки
; Входная точка инициализации прикладной программы.
PINIT   RTS             ;для управляющей
                        ;прикладной
                        ;программы не
                        ;совершать никаких
                        ;действий
; Далее следует основное тело программы.
RESTRT=`
;Здесь основное тело программы заканчивается.
PND=    `               ;"PND"-следующая
        .END            ;свободная ячейка
```

Рис. 10-4 Образец загружаемой с кассеты программы.

Программа для создания файлов, загружаемых с кассеты.

В данном разделе приводится листинг программы, генерирующий файлы, которые могут быть загружены с кассеты. Данная программа не является единственно возможной, и ее элегантность оставляет желать лучшего.

```
; Данная программа записывает
; единственный файл на кассету и
; используется вместе с процедурой,
; дающей возможность файлам быть
; загруженными с кассеты. Следующим
; двум символам необходимо присвоить
; значения при помощи границ памяти
; копируемой программы.
; "PST"- начальный адрес программы
; "PND"- конечный адрес программы
;
PST=   :0700
PND=   :1324
FLEN=  PND-PST+127/128*128 ;округление
                           ;до числа кратного 128
*=     :B000               ;начальный адрес
                           ;программы
BOOTB  LDX {:10            ;используется IOCB 1
;
; Сначала откроем для записи файл на кассете.
       LDA {OPEN           ;открыть уст-во (OPEN)
       STA ICCOM,X
       LDA {OPNOT          ;направление - вывод
       STA ICAX1,X
       LDA {:80            ;выбрать короткий IRG
       STA ICAX2,X
       LDA {CFILE          ;установить указатель
       STA ICBAL,X         ;на имя устройства
       LDA {CFILE/256
       STA ICBAH,X
       JSR CIOV            ;попытка открыть файл
       BMI CERR            ;ошибка
; Перепишем внутренний файл за одну операцию.
       LDA {PUTCHR         ;установить вывод
       STA ICCOM,X         ;символа
                           ;(PUT CHARACTERS)
       LDA {PST            ;установить начало
       STA ICBAL,X         ;прикладной программы
       LDA {PST/256
       STA ICBAH,X
       LDA {FLEN           ;установить число
       STA ICBLL,X         ;байт, которые
       LDA {FLEN/256       ;необходимо
       STA ICBLH,X         ;записать
       JSR CIOV            ;записать внутренний
                           ;файл
       BMI CERR            ;ошибка
```

```
; Теперь, в случае удачной записи, закроем файл.
      LDA #CLOSE  ;установить закрытие
      STA ICCOM,X ;(CLOSE)
      JSR CIOV    ;закрыть файл
      BMI CERR    ;ошибка
      BRK         ;остановиться после
                  ;выполнения
CERR  BRK         ;остановиться при
                  ;ошибке
CFILE .BYTE "C:",CR ;имя файла
; Головная часть картриджа.
*=    $BFF9       ;картридж A
INIT  RTS
      .WORD BOOTB
      .BYTE 0,4
      .WORD INIT
      .END
```

11. Прогрессивные технологии и заметки о применении.

В данном разделе приведена информация по использованию возможностей ОС и некоторых возможностей аппаратных средств, недоступных непосредственно из ОС и находящихся в противоречии с другими частями операционной системы.

Генерация звука.

Возможности генерации звука специальной микросхемой POKEY используется ОС лишь в подсистеме ввода/вывода для генерации тона кассетного FSK и для установки звуковой шины в SIO.

Возможности.

Аппаратными средствами генерируются четыре независимых аудио канала, которые смешиваются и посылаются к телевизору в составе композиционного видео сигнала. Нижеуказанные регистры POKEY связаны с управлением звука. (Это описано в руководстве к аппаратному обеспечению.)

```
AUDCTL (D208)                управление звуком.
AUDC1(D201) и AUDF1(D200) управление каналом 1.
AUDC2(D203) и AUDF2(D202) управление каналом 2.
AUDC3(D205) и AUDF3(D204) управление каналом 3.
AUDC4(D207) и AUDF4(D206) управление каналом 4.
```

Конфликты с ОС.

При генерации звука могут возникнуть два вида конфликтов с ОС:

- ОС может генерировать собственный звуковой сигнал в процессе операций ввода/вывода на кассету или периферийное устройство последовательной шины и при этом отключать любой другой звуковой сигнал.
- ОС не отключает звук при нажатии (BREAK) или (RESET). Программа должна обеспечивать отключение, при необходимости, звука при данных условиях.

Экранная графика.

Возможности аппаратных средств.

Возможности аппаратных средств по представлению экранных данных крайне широки. ОС использует лишь малую часть этих возможностей. Информацию, касающуюся представления данных на экране, можно найти в руководстве к аппаратному обеспечению.

Возможности ОС.

Резидентный дисплей поддерживает работу любых 8 из 11 возможных экранных режимов (В случае, если специальная микросхема CTIA заменена на GTIA, то 11 из 14 возможных.) Резидентный дисплей позволяет получить на экране текстовое окно заданного размера. Как известно из руководства к аппаратному обеспечению аппаратные средства допускают гораздо более широкий выбор, нежели дисплей.

Управление курсором.

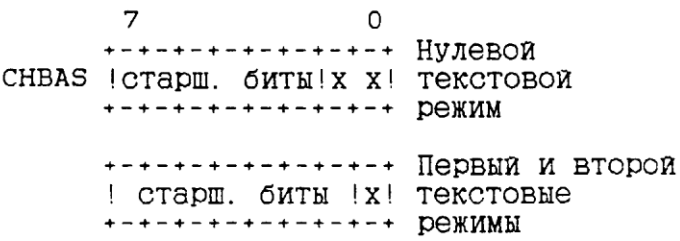
Можно осуществлять непосредственное управление курсором на дисплее в текстовом и графическом режимах (см. Раздел 5, Приложение L, B1-4).

Управление цветом.

Можно изменять значения цветовых регистров, задаваемые при выполнении операции OPEN (см. Приложение L, B7-8). Заметим, что значения величин будут установлены такие, которые принимаются по умолчанию после каждой перезагрузки системы или операции OPEN для дисплея.

Переменный знакогенератор.

В текстовых режимах 1 и 2 доспупны два знакогенератора. Значение переменной базы данных CHBAS (02F4) устанавливает необходимый пользователю знакогенератор. По умолчанию устанавливается значение !E0, определяющее заглавные буквы (верхний регистр), цифры и знаки пунктуации (соответствующи дисплейному коду от !20 до !5F (см. Приложение E)). Альтернативное значение !E2 определяет буквы нижнего регистра, специальные графические символы (соответствующие дисплейному коду от !60 до !7F и от !00 до !1F (см. Приложение E)). Знакогенераторы, определяемые пользователем могут быть доступны из текстовых режимов 0, 1 и 2. Они задают матричное определение символов в ОЗУ. Указатель CHBAS устанавливается на эти определения. CHBAS содержит старшие биты адреса памяти, по которому расположены определения символов, как это показано на рисунке:



(x - неиспользуемые биты адреса, равные нулю).

Рис. 11-1 Адрес в памяти устанавливаемого пользователем знакогенератора.

Каждый символ определяется матрицей 8x8 бит. Символ "T" определяется следующим образом:

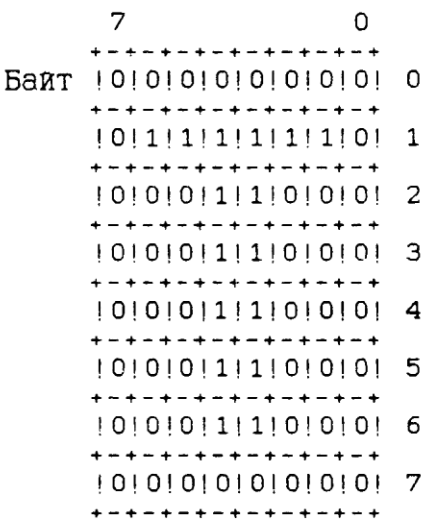


Рис. 11-2 Определяемая пользователем таблица битовой 8x8 матрицы символа.

В памяти каждому символу знакогенератора отводится восемь последовательных байт, причем символы упорядочены по значениям их внутренних кодов (см. Приложение L, B55).

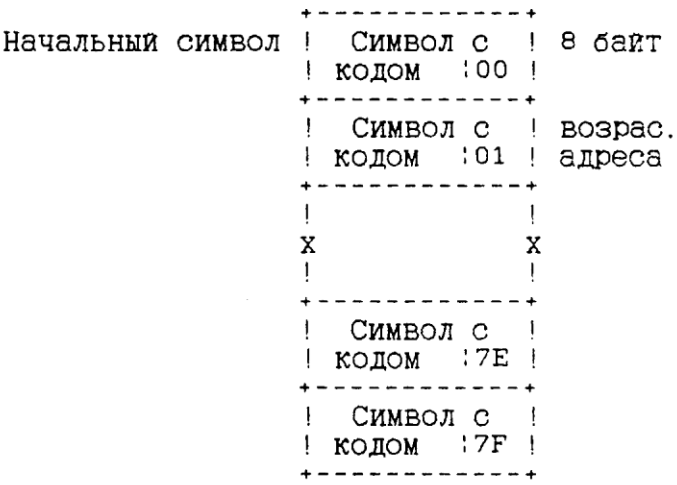


Рис. 11-3 Схема хранения символов.

Графика игрок/снаряд.

ОС не использует возможности аппаратных средств по генерации графики игрок/снаряд. Она может быть использована независимо от ОС без всяких конфликтов.

Возможности аппаратных средств.

Аппаратные средства позволяют размещать и перемещать по экрану объекты ограниченных размеров без воздействия на "игровое поле" (побитовая графика или символьные данные). Возможность управления приоритетами позволяет установить объект с преимущественным изображением в случае возникновения конфликта (перекрытие изображений).

Конфликты с ОС.

Пользователь должен убедиться, что данные игрок/снаряд выровнены по адресам, как это требуется PMASE (D407). Также пользователем должна быть найдена подходящая область памяти, которая была бы гарантирована ОС от использования любой внешней программой.

Считывание с игровых контроллеров.

На второй стадии процесса VBLANK ОС осуществляет считывание с игровых контроллеров (см. Приложене L J1-9):

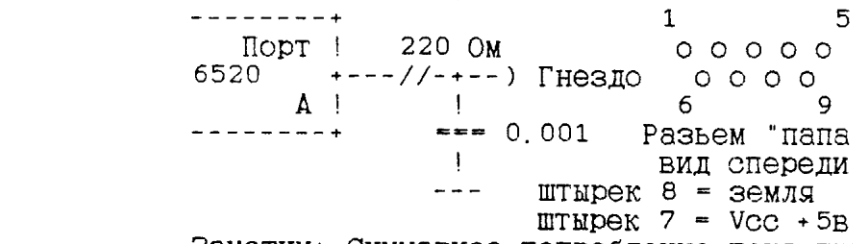
- Джойстики/тригеры 1-2.
- Контроллеры PADDLE/тригеры 1-4.
- Управляемые контроллеры/тригеры 1-2.
- Световое перо/триггер.

В дополнении к этим контроллерам информация может быть получена или передана к устройствам, подключенным к коннекторам консоли при помощи специальной микросхемы PIA.

Информация об аппаратных средствах.

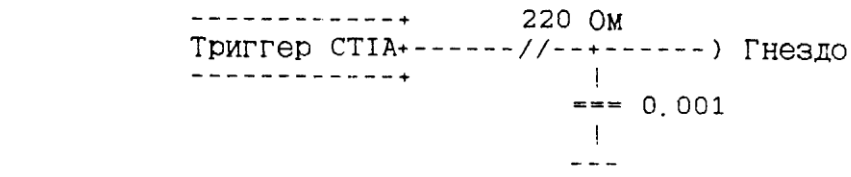
PIA (6520/6820)
Выход: уровни TTL, 1 загрузка.
Вход: уровни TTL, 1 загрузка.
Дополнительную информацию можно найти в руководстве к чипу 6520.

Схема порта A:



Заметим: Суммарное потребление тока при данном напряжении не должно превышать 50 мА.

Схема порта триггера:



Управление из программ.

6520 PIA:

Управление портом А (адрес D302)

```
 7 6 5 4 3 2 1 0
+-+-+-+-+-+-+-+-+
!0!0!1!1!1!X!0!0!записать это в данный регистр
+-+-+-+-+-+-+-+-+
          !   Управление адресацией
          +-- направления данных для порта А
              1 = направление данных в D300,
              0 = данные в D300.
```

Направление данных в порт А (адрес D300)

```
 7 6 5 4 3 2 1 0
+-+-+-+-+-+-+-+-+
!X!X!X!X!X!X!X!X!записать это в данный регистр
+-+-+-+-+-+-+-+-+
 ! ! ! ! ! ! ! !   Управление направлением
 +-+-+-+-+-+-+-+-- данных для порта А
                   1 = вывод, 0 = ввод.
```

Данные порта А (адрес D300)

```
 7 6 5 4 3 2 1 0
+-+-+-+-+-+-+-+-+
! ! ! ! ! ! ! ! !считать или записать этот регистр
+-+-+-+-+-+-+-+-+
 4 3 2 1 4 3 2 1
+-------+ +-------+
Гнездо 2 Гнездо 1-- Номера контактов
```

Четыре порта триггеров: D010, D011, D012, D013

```
 7 6 5 4 3 2 1 0
+-+-+-+-+-+-+-+-+
!0!0!0!0!0!0!0!X!считать данный порт
+-+-+-+-+-+-+-+-+
              !   Значение триггера
              +-- D010 = Порт 1, Штырек 6;
                  D013 = Порт 4, Штырек 6.
```

Дополнительная информация о программировании.

1. При инициализации все порты PIA устанавливаются ОС на ввод.
2. ОС обычно считывает вышеуказанную информацию в ОЗУ один раз за телевизионный кадр (во время вертикального пропуска, следующим образом:

Имя базы данных	Адрес	Данные	Штырек S
STICK0	0278	7 6 5 4 3 2 1 0	
		+-+-+-+-+-+-+-+-+	
		!0!0!0!0!x!x!x!x!	Разъем 1, Шт. 4,3,2,1
		+-+-+-+-+-+-+-+-+	
STICK1	0279		Разъем 2, Шт. 4,3,2,1
STRIG0	0284	7 6 5 4 3 2 1 0	
		+-+-+-+-+-+-+-+-+	
		!0!0!0!0!0!0!0! !	Разъем 1, Штырек 6
		+-+-+-+-+-+-+-+-+	
STRIG1	0285		Разъем 2, Штырек 6
PADDL1	0270	7 6 5 4 3 2 1 0	
		+-+-+-+-+-+-+-+-+	
		!x!x!x!x!x!x!x!x!	Разъем 1, Штырек 5
		+-+-+-+-+-+-+-+-+	
PADDL3	0272		Разъем 2, Штырек 5
PADDL0	0271		Разъем 1, Штырек 9
PADDL2	0273		Разъем 2, Штырек 9
PADDL2	0273		Разъем 2, Штырек 9

Рис. 11-6 Использование Разъемов джойстиков в качестве портов ввода/вывода: Таблица функций штырька.

Считывание со штырьков 5 и 9 производится через схемы контроллера PADDLE. Значение 7 свидетельствует о том, что на штыреке высокий потенциал (или плавающий), а значение 228 означает, что штырек заземлен.

Приложение А.

Значения командного байта CIO.

Ниже приведены шестнадцатеричные величины командного байта CIO.

Для большинства устройств:
03 -- OPEN
05 -- GET RECORD
07 -- GET CHARACTERS
09 -- PUT RECORD
0B -- PUT CHARACTERS
0C -- CLOSE
0D -- GET STATUS

Только для дисплея:
11 -- FILL
12 -- DRAW

Только для обработчика дисковых файлов:
20 -- RENAME
21 -- DELETE
22 -- FORMAT
23 -- LOCK
24 -- UNLOCK
25 -- POINT
26 -- NOTE

Приложение В.

Значения состояний CIO.

Ниже приведены различные возможные значения байта состояния CIO с пояснением.

01 (001) -- Операция завершена (ошибок нет).
80 (128) -- Прерывание клавишей BREAK.
81 (129) -- Попытка открыть уже открытое IOCB (при OPEN).
82 (130) -- Устройство не существует.
83 (131) -- Открыто только для записи.
84 (132) -- Неправильная команда.
85 (133) -- Устройство или файл не открыты.
86 (134) -- Неверный номер IOCB (только для регистра Y).
87 (135) -- Открыто только для чтения.
88 (136) -- Конец файла.
89 (137) -- Запись оборвана.
8A (138) -- Устройство не ответило.
8B (139) -- Нет подтверждения от устройства (NAK).
8C (140) -- Ошибка образования входного фрейма последовательной шины.
8D (141) -- Выход курсора из диапозона.
8E (142) -- Переполнение буфера данных последовательной шины.
8F (143) -- Ошибка в контрольной сумме данных последовательной шины.
90 (144) -- Ошибочные действия устройства.
91 (145) -- Неверный экранный режим.
92 (146) -- Функция не выполнима устройством.
93 (147) -- Недостаточно памяти для выбранного экранного режима.
A0 (160) -- Ошибочно указан номер дисковода.
A1 (161) -- Открыто слишком много дисковых файлов.
A2 (162) -- Дискета полностью заполнена.
A3 (163) -- Грубая ошибка ввода/вывода на дискету.
A4 (164) -- Рассогласование номера внутреннего файла.
A5 (165) -- Ошибка в имени файла.
A6 (166) -- Неверная длина указателя на данные.
A7 (167) -- Файл блокирован.
A8 (168) -- Для дисковода команда неприменима.
A9 (169) -- Директория заполнена (64 файла).
AA (170) -- Файл не найден.
AB (171) -- Неправильный указатель.

Приложение Н.

Характеристики экранных режимов.

N	Гор. поз.	Верт. без разб.	Верт. с разб.	Цвет	Велич. данных	Цвет. регис.	Необх. память с разб.	Необх. память без
0	40	24	--	2	фон	BAK	992	992
					00-FF	PF 2		
					"	PF 1		
1	20	24	20	5	фон	BAK	674	672
					00-3F	PF 0		
					40-7F	PF 1		
					80-BF	PF 2		
					C0-FF	PF 3		

2	20	12	10	5	Фон	BAK	424	420
					00-3F	PF 0		
					40-7F	PF 1		
					80-BF	PF 2		
					C0-FF	PF 3		
3	40	24	20	4	0	BAK	434	432
					1	PF 0		
					2	PF 1		
					3	PF 2		
4	80	48	40	2	0	BAK	694	696
					1	PF 0		
5	80	48	40	4	0	BAK	1174	1176
					1	PF 0		
					2	PF 1		
					3	PF 2		
6	160	96	80	2	0	BAK	2174	2184
					1	PF 0		
7	160	96	80	4	0	BAK	4190	4200
					1	PF 0		
					2	PF 1		
					3	PF 2		
8	320	192	160	2	0	PF 2	8112	8138
					1	PF 1`		
9	80	192	--	1	Примечание 2			
10	80	192	--	9	0	PM 0		
					1	PM 1		
					2	PM 2		
					3	PM 3		
					4	PF 0		
					5	PF 1		
					6	PF 2		
					7	PF 3		
					8	BAK		
					9	BAK		
					A	BAK		
					B	BAK		
					C	PF 0		
					D	PF 1		
					E	PF 2		
					F	PF 3		
11	80	192	--	16	Примечание 3			

Примечания:
`. Цвет в PF 2, яркость в PF 1.
2. Цвет в BAK, яркость определяется заданной величиной (!0-F).
3. Цвет определяется заданной величиной, яркость в BAK.
3. Цвет определяется заданной величиной, яркость в BAK.
PF X ::= цветовой регистр x игрового поля.
PM X ::= цветовой регистр графики игрок/снаряд.
BAK ::= цветовой регистр фона, также может иметь название PF 4.

Ниже приведены величины цветовых регистров, принимаемые по умолчанию.
BAK = !00
PF 0 = !28
PF 1 = !CA
PF 2 = !94
PF 3 = !46.

Приложение I.

ID последовательной шины и содержание команд.

ID устройств последовательной шины:

Дисковод для гибких дисков	D1-D4	!31-34
Принтер	P1	!40
RS-232-C	R1-R4	!50-53

Управляющие коды последовательной шины:

ACK	-	!41 ("A")
NAK	-	!4E ("N")
COMPLETE	-	!43 ("C")
ERR	-	!45 ("E")

Коды команд последовательной шины:

READ	-	!45 ("R")	Диск
WRITE	-	!57 ("W")	Принтер/Диск
STATUS	-	!53 ("S")	Принтер/Диск
PUT (без проверки)	-	!50	Диск
FORMAT SINGL	-	!21 ("!")	Диск
FORMAT MEDIUM	-	!22 (""")	Диск
READ ADDRESS	-	!54 ("T")	
READ SPIN	-	!51 ("G")	Диск
MOTOR ON	-	!55 ("U")	Диск
VERIFY SECTOR	-	!56 ("V")	Диск

Ниже приведена форма цветового регистра:

```
  7 6 5 4 3 2 1 0
 +-+-+-+-+-+-+-+-+
 ! цвет  !ярк. !0!
 +-+-+-+-+-+-+-+-+
```

Где

Цвет	Яркость
0 - серый	0 - минимальная яркость
1 - светло-оранжевый	1
2 - оранжевый	2
3 - красно-оранжевый	3 Нарастание
4 - розовый	4 яркости
5 - пурпурный	5
6 - пурпурно-голубой	6
7 - голубой	7 - максимальная яркость
8 - голубой	
9 - светло-голубой	
A - бирюзовый	
B - зеленовато-голубой	
C - зеленый	
D - желто-зеленый	
E - оранжево-зеленый	
F - светло-оранжевый	

Приложние J.

Вектора ПЗУ.

Ниже указаны вектора JMP ПЗУ ОС с фиксированным адресом. По каждому адресу находится команда JMP на указанную программу.

Имя	Адрес	Ссылка	Функция
DISKIV	E450	`	Инициализация дисковода.
DSKINV	E453	5.4.2	Вход дисковода.
CIOV	E456	5.2	Вход в программу CIO.
SIOV	E459	9.3	Вход в программу SIO.
SETVBV	E45C	6.7.2	Установить программу сис. таймеров.
SYSVBV	E45F	6.3	Вход в 1 стадию VBLANK.
XITVBV	E462	6.3	Выход из VBLANK.
SIOINV	E465	`	Инициализация программы SIO.
SENDEV	E468	`	Переслать доступную программу.
INTINV	E46B	`	Инициализация управления прерываниями.
CIOINV	E46E	`	Инициализация программы CIO.
BLKBDV	E471	3.1.1	Вход в режим "самотестирования".
WARMSV	E474	7	Горячий старт (от (RESET)).
COLDSV	E477	7	Холодный старт (включение питания).
RBLOKV	E47A	`	Вход в блок считанный с кассеты.
CSOPIV	E47D	`	Начало открытого ввода с кассеты.

` Вектора предназначены для внутреннего использования операционной системой.

Ниже приведены адреса входных точек программ ПЗУ, имеющих фиксированные адреса и относящихся к пакету математики с плавающей запятой (п.з.). Подробное описание этих программ можно найти в разделе 8.

AFP	D800	Преобразование ASCII в число с п.з.
FASC	D8E6	Преобразование числа с п.з. в ASCII
IFP	D9AA	Преобразование целого числа в число с п.з.
FPI	D9D2	Преобразование числа с п.з. в целое
FADD	DA66	Сложение чисел с п.з.
FSUB	DA60	Вычитание чисел с п.з.
FMUL	DADB	Умножение чисел с п.з.
FDIV	DB28	Деление чисел с п.з.
LOG	DECD	Натуральный логарифм числа с п.з.
LOG10	DED1	Десятичный логарифм от числа с п.з.
EXP	DDC0	Возведение числа в степень с п.з.
EXP10	DDCC	Возведение 10 в степень с п.з.
PLYEVL	DD40	Полиномиальная оцнка чисел с п.з.
ZFR0	DA44	Очистить FR0
ZF1	DA46	Обнулить число с п.з.
FLD0R	DD89	Загрузить число с п.з.
FLD0P	DD8D	Загрузить число с п.з.
FLD1R	DD98	Загрузить число с п.з.
FLD1P	DD9C	Загрузить число с п.з.
FSTOR	DDA7	Запомнить число с п.з.
FSTOP	DDAB	Запомнить число с п.з.
FMOVE	DD86	Переместить число с п.з.

Ниже указаны базовые адреса векторов для резидентных устройств:

Экранный редактор (E:)	E400
Дисплей (S:)	E410
Клавиатура (K:)	E420
Принтер (P:)	E430
Кассетный магнитофон (C:)	E440

Формат входных точек устройства приведен в разделе 5.

Ниже приведены значенеия векторов прерываний компьютера 6502.

Функция	Адрес	Значение
NMI (немаскируемые)	FFFA	E7B4
RESET (перезагрузка)	FFAC	E477
IRQ (по запросам)	FFFE	E6FE

Приложение К.

Характеристики устройств.

В данном приложении указаны физические характеристики устройств, взаимодействующих с компьютерами ATARI XL, XE. Детально рассмотрены следующие аспекты: допустимый объем данных, скорость переноса данных, формат памяти, взаимодействие с SIO, подсоединение к компьютеру.

Клавиатура.

Скорость ввода с клавиатуры ограничена процедурой считывания с клавиатуры, позволяющей принимать до 60 символов в секунду. Коды клавиш приведены в таблице 5-4. Аппаратные средства клавиатуры не дают возможность буферизации. Использование непрерывного алгоритма вызывает ограничение быстродействия.

Дисплей.

Генератор изображения на телеэкране имеет множество возможностей, не используемых дисплеем (как это было описано в разделе 5 и показано в приложении Н). Существуют дополнительные экранные режимы, генераторы объектов, аппаратный "скроллинг" изображения, а также множество других различных свойств, познакомиться с которыми можно, изучив руководство к аппаратному обеспечению домашнего компьютера ATARI. Поскольку все данные об изобржении хранятся в ОЗУ, скорость корректировки данных ограничена прежде всего прикладными программами, генерирующими и форматирующими данные и осуществляющими доступ к ОЗУ. Генерация изображения из информации, хранящейся в ОЗУ, осуществляется специальными микросхемами ANTIC и CTIA или GTIA при помощи прямого доспупа к памяти, позволяющего осуществить доступ к ОЗУ. Формат внутренней памяти под данные об изображении детально рассмотрен в ATARI HARDWARE MANUAL.

Магнитофон ATARI XC12.

Магнитофон ATARI XC12 имеет следующие характеристики:

Объем данных:
100 символов на пленке С-60 (неформатированной).
Скорость переноса данных:
600 бод (60 символов в секунду).
Примечание: ОС имеет возможность работать с разнои скоростью движения ленты (447-895 бод).

Формат хранения:
Запись проводится в формате 1/4 стереодорожки со скоростью 1 7/8 дюймов в секунду. Лента может быть записана в обоих направлениях, где дорожки 1 и 2 являются соответственно правой и левой для стороны А, а дорожки 3 и 4 являются соответственно правой и левой для стороны В (промышленный стандарт). С каждой стороны левый канал (первый или четвертый) используется под аудио сигнал, а правый (второй или третий) под цифровую информацию. Аудио канал записывается традиционным способом. Цифровой канал записываетсяпри помощи данных FSK, получаемых в двухтоновом режиме POKEY со скоростью до 600 бод. Частота маркера - 5327 Гц, а частота пробела - 3995 Гц. Передача данных является асинхронной и осуществляется последовательно по байтам. POKEY считывает или записывает для каждого байта последовательность бит FSK в следующем порядке:

```
1 начальный бит (пробел)
бит данных 0 -+
бит данных 1  !
.............. +- 0-пробел, 1-маркер.
бит данных 6  !
бит данных 7 -+
1 конечный бит (маркер)
```

Компьютер в состоянии управлять лишь одной функцией движения ленты в магнитофоне: включением и выключением мотора, и только в том случае, если пользователем нажата клавиша PLAY. Для того, чтобы произвести запись необходимо нажать клавиши REC и PLAY. Пользователю следует аккуратно обращаться с этим устройством, поскольку компьютер не имеет возможности контролировать нажатие этих клавиш даже, если магнитофон ATARI XC12 подключен к компьютеру.

Взаимодействие с SIO.

Магнитофон частично использует аппаратные средства последовательной шины, но, как было сказано в разделе 9, не придерживается никакого протокола.

40-стобцовый принтер ATARI 820.

Принтер ATARI 820 имеет следующие характеристики:

```
Объем данных:
40 символов в строке (нормальная печать)
29 символов в строке (печать с наклоном)
Скорость печати: xx символов в секунду.
Формат хранения:
Лист шириной 3 7/8 дюйма.
Матрица 5x7 точек, импульсная печать.
Обычный формат:
40 символов в строке.
6 строк на дюйм (по вертикали).
12 символов на дюйм (по горизонтали).
Формат с наклоном:
29 символов в строке.
6 строк на дюйм (по вертикали).
9 символов на дюйм (по горизонтали).
```

Взаимодействие с SIO.

ID контроллера последовательной шины равно $40. Контроллер поддерживает следующие команды SIO (информацию, относящуюся к устройству, см. раздел 5, а общее описание команд шины в разделе 9):

GET STATUS (узнать состояние).

Компьютер посылает командный фрейм следующего формата:
ID устройства = !40.
Командный байт = !53.
Вспомогательный байт 1 = не имеет значения.
Вспомогательный байт 2 = не имеет значения.
Контрольная сумма = контрольная сумма выше указанных байт.
Контроллер принтера, посылает в ответ фрейм данных, ранее указанного в данном приложении в разделе GET STATUS формата.

PRINT LINE (печать строки).

Компьютер посылает командный фрейм нижеуказанного формата:
ID устройства = !40.
Командный байт = !57.
Вспомогательный байт 1 = не имеет значения.
Вспомогательный байт 2 = не имеет значения.
Контрольная сумма = контрольная сумма вышеуказанных байт.
Компьютер посылает фрейм данных следующего формата:
Крайний левый символ в строке (столбец 1).
Следующий символ строки.
Самый крайний символ в строке (столбец 40 или 29).
Байт контрольной суммы.
Заметим, что размер фрейма данных может, в зависимости от режима печати, указанного в командном фрейме, иметь длину 41 или 30 байт.

Дисковод ATARI 1050.

Дисковод ATARI 1050 имеет следующие основные характеристики:
Объем данных:
720 секторов по 128 байт (формат дискового устройства).
1040 секторов по 128 байт (средняя плотность).
709 секторов по 125 байт (формат обработчика дисковых файлов).
Скорость передачи данных:
Скорость шины: 1920 символов в секунду.
Время поиска: 5.25 мсек на дорожку + 10 - 210 мсек.
Формат хранения:
5 1/4-дюймовая дискета с мягким разбиением на сектора, осуществляемым контроллером.
40 дорожек на дискете.
18 секторов на дорожке.
128 байт в секторе.
Управление осуществляется чипом - форматтером/контроллером NATIONAL INS1771-1. Последовательность секторов на дорожке:
18,1,3,5,7,9,11,13,15,17,2,4,6,8,10,12,14,16.

Взаимодействие с SIO.

ID контроллер последовательной шины лежит в диапазоне (от !31 (для "D1") до !34 (для "D4")). Контроллер поддерживает выполнение следующих команд SIO (информацию по дисководу см. в данном приложении, а общее описание команд шины можно найти в разделе 9):

GET STATUS

Компьютер посылает командный фрейм следующего формата:
ID устройства = !31-34.
Командный байт = !53.
Вспомогательный байт 1 = не имеет значения.
Вспомогательный байт 2 = не имеет значения.
Контрольная сумма = контрольная сумма вышеуказанных байт.
Контроллер дискеты посылает в ответ фрейм данных формата, описанного ранее в этом приложении в разделе STATUS REQUEST.

PUT SECTOR (WITH VERIFY) (вывести сектор с проверкой).

Компьютер посылает командный фрейм следующего формата:
ID устройства = !31-34.
Командный байт = !57.
Вспомогательный байт 1 = низший байт номера сектора.
Вспомогательный байт 2 = высший байт номера сектора.
Контрольная сумма = контрольная сумма вышеуказанных байт.
Компьютер посылает фрейм данных следующего формата:
128 байт данных.
Байт контрольной суммы.
Контроллер дискеты записывает фрейм данных в определенный сектор, затем считывает сектор и сравнивает содержимое с фреймом данных. Байт COMPLETE (завершение) указывает состояние операции.

PUT SECTOR (NO VERIFY) (вывести сектор без проверки).

Компьютер посылает командный фрейм следующего формата:
ID устройства = !31-34.
Командный байт = !50.
Вспомогательный байт 1 = низший байт номера сектора.
Вспомогательный байт 2 = высший байт номера сектора.
Контрольная сумма = контрольная сумма вышеуказанных байт.
Компьютер посылает фрейм данных следующего формата:
128 байт данных.
Байт контрольной суммы.
Контроллер дискеты записывает фрейм данных в определенный сектор, затем посылает значение байта COMPLETE, указывающее состояние операции.

GET SECTOR (получить сектор).

Компьютер посылает командный фрейм следующего формата:
ID устройства = !31-34.
Командный байт = !52.
Вспомогательный байт 1 = низший байт номера сектора.
Вспомогательный байт 2 = высший байт номера сектора.
Контрольная сумма = контрольная сумма вышеуказанных байт.
Контроллер дискеты посылает фрейм данных следующего формата:
128 байт данных.
Байт контрольной суммы.

FORMAT DISKETTE (форматирование дискеты).

Компьютер посылает командный фрейм следующего формата:
ID устройства = !31-34.
Командный байт = !21 или !22.
Вспомогательный байт 1 = не имеет значения.
Вспомогательный байт 2 = не имеет значения.
Контрольная сумма = контрольная сумма вышеуказанных байт.
Контроллер дискеты полностью форматирует дискету (генерирует 40 дорожек с гибким разбиением, заполняя всю область данных нулями), затем считывает каждый сектор для проверки целостности. Во фрейм данных из 128 байт и контрольной суммы возвращаются номера всех плохих секторов (максимальное их число - 63), за которым следуют два байта !FF.

Приложение L.

Функциональное описание переменных базы данных ОС.

В данном приложении содержится описание многих переменных базы данных, которые даны для всех доступных пользователю переменных и некоторых внутренних переменных. Имена переменных, не представляющие какого-либо интереса помечены звездочкой (*). Знакомство с остальными переменными будет полезно следующим классам пользователей.
- Опытным пользователям.
- Разработчикам игр.
- Проектировщикам прикадных программ.
- Авторам системных программ.
- Разработчикам языковых процессоров.

Каждая переменная определяется именем системного файла, за которым следует его шестнадцатеричный адрес и число (десятичное) зарезервированных в базе данных байт.
Общий вид:
(имя)(/адрес/./размер/)
Например:
MEMLO (02E7,2)
Заметим, что у большинства переменных, длина которых - слово (2 байта), младший байт располагается по младшему адресу.

A. Конфигурация памяти.

За общим описанием динамики памяти советуем обратиться к разделу 4, а в разделе 7 можно ознакомиться с подробностями процеса инициализации системы.

A1 MEMLO (02E7,2) -- Наименьший адрес свободной памяти.

MEMLO содержит адрес первой ячейки в области сводобной памяти. Значение устанавливается ОС в процессе инициализации при перезагрузке системы или включении питания и далее не изменяется.

A2 MEMTOP (02E5,2) -- Старший адрес свободной памяти.

MEMTOP содержит адрес первой недоступной ячейки памяти выше области свободной памяти. Значение устанавливается ОС в процессе инициализации при перезагрузке системы или при включении питания, изменяясь впоследствии всякий раз при открытии дисплея в зависимости от выбранного графического режима.

A3 APPMHI (000E,2) -- Нижняя граница экранной памяти.

APPMHI представляет собой управляемую пользователем переменную, содержащую адрес внутри области свободной памяти, ниже которого память, занимаемая под изображение на дисплее, располагаться не может. При включении питания эта переменная инициализируется в ноль.

A4 RAMTOP (006A,1) -- Верхний адрес ОЗУ, отводимого под дисплей (старший байт).

В RAMTOPe постоянно хранится верхний адрес ОЗУ, содержащийся в TRAMSZ для использования дисплеем. Значение устанавливается в процессе инициализации устройства.

A5 RAMSIZ (02E4,1) -- Верхний адрес ОЗУ (только старший байт).

В RAMSIZe постоянно сохраняется верхний адрес ОЗУ, содержащийся в TRAMSZ. Как описано в N1.

B3 OLDROW (005A,1) и OLDCOL (005B,2) -- Предыдущее положение курсора.

OLDROW и OLDCOL принимают значение ROWCRS и COLCRS соответственно перед каждой операцией. Переменные используются только для операции DRAW и FILL.

B4 TXTROW (0290,1) и TXTCOL (0291,2) -- Положение курсора в текстовом окне

TXTROW и TXTCOL определяют положение курсора (строку и столбец соответственно) для элемента данных, который будет записан или считан следующий из текстового окна экрана. Нумерация строк и столбцов начинается с нулевого значения и возрастает соответственно до 3 и 39, причем левый верхний угол текстового окна имеет координаты (0,0).

Границы экрана.

Текстовой экран и текстовое окно графического экрана имеют изменяемые пользователем левую и правую границу, определяющую местоположение текстового курсора.

B5 LMARGN (0052,1) -- Левая граница текста.

LMARGN содержит номер крайнего левого столбца (0-39) текстового экрана. Текстовой курсор будет находится в этом столбце или правее после всех операций до тех пор, пока значение переменной номера столбца курсора не будет непосредственно изменено пользователем. По умолчанию значение LMARGN устанавливается равным 2 при включении питания или при перезагрузке системы.

B6 RMARGN (0053,1) -- Правая граница текста.

RMARGN содержит номер крайнего правого столбца текстового окна экрана. Курсор будет располагаться в этом столбце или левее до тех пор, пока значение переменной номера столбца курсора не будет непосредственно изменено пользователем. По умолчанию значение RMARGN устанавливается равным 39 при включении питания или при перезагрузке системы.

Управление цветом.

На второй стадии процесса VBLANK (см. раздел 6) значение девяти переменных базы данных сохраняются в соответсвующих цветовых аппаратных регистрах. Цветовые регистры разделены на две группы: цвета объектов игрок/снаряд и цвета игрового поля. Цветовые регистры игрового поля используются различными экранными режимами, так как это было показано в приложении Н. Цветовые регистры игрок/снаряд настоящей ОС не используются.

B7 PCOLOR0-PCOLOR3 (02C0,4) -- Цвета графики игрок/снаряд.

Каждая цветовая переменная расположена в соответствующем аппаратном регистре, как это показано ниже:

PCOLR0 (02C0)	COLPM0 (D012)
PCOLR1 (02C1)	COLPM1 (D013)
PCOLR2 (02C2)	COLPM2 (D014)
PCOLR3 (02C3)	COLPM3 (D015)

Каждая цветовая переменная имеет следующий формат:

```
  7 6 5 4 3 2 1 0
 +-+-+-+-+-+-+-+-+
 ! Цвет  ! ярк.!x!
 +-+-+-+-+-+-+-+-+
```

Значение полей цвета и яркости можно узнать из приложения Н.

B8 COLOR0-COLOR4 (02C5,5) -- Цвет игрового поля.

Каждая цветовая переменная хранится в соответствующем апаратном регистре, как показано ниже:

COLOR0 (02C4)	COLPF0 (D016)
COLOR1 (02C5)	COLPF1 (D017)
COLOR2 (02C6)	COLPF2 (D018)
COLOR3 (02C7)	COLPF3 (D019)
COLOR4 (02C8)	COLPF4 (D01A)

Каждая цветовая переменная имеет следующий формат:

```
  7 6 5 4 3 2 1 0
 +-+-+-+-+-+-+-+-+
 ! Цвет  ! ярк.!x!
 +-+-+-+-+-+-+-+-+
```

Информацию о значениях полей и яркости можно найти в приложении Н.

"Скроллинг" текста.

Текстовой экран или текстовое окно графического экрана "проскальзывает" вверх при возникновении двух следующих условий:

- Нижняя строка текста на экране перешла через правую границу.
- Нижняя строка экрана закончена символом конца строки (EOL).

Скроллинг(прогон) вызывает смещение внутренней логической строки, начинающееся с верхней части экрана и последующее перемещение наверх остальных логических строк, чтобы закрыть образовавшуюся брешь. В случае, если уничтоженная логическая строка превосходит одну физическую строку курсор будет также перемещен вверх.

B9 SCRFLG` (02BB,1) -- Флаг скроллинга.

SCRFLG предсвавляет собой рабочую переменную, указывающую число физических строк, удаленных с верхней части экрана минус единица. Когда размер логической строки изменяется от 1 до 3, значение флага SCRFLG от 0 до до 2.

Неработающий режим.

Неработающий режим представляет собой механизм, защищающий телевизионный экран от усиленной бомбардировки люминофора, которая имеет места при сохранении неподвижного изображения на экране в течении длительного времени. Когда компьютер оставлен без присмотра более, чем на 9 минут, интенсивность свечения уменьшается вдвое, а оттенки изменяются каждые 8.3 секунды. Нажатие любой клавиши на клавиатуре выводит компьютер из этого режима. В процессе второй стадии процесса VBLANK цветовые регистры базы данных посылаются в соответствующие аппаратные цветовые регистры, причем перед посылкой они претерпевают следующие изменения:
аппаратный регистр = переменная базы данных COLRSH и DRKMSK.
Обычно COLRSH=!00, а DRKMSK=!FF. При этом вышеописанная операция становится пустой. Когда активизируется нерабочий режим COLRSH становится равным содержимому RTCLOK+1, а DRKMSK=!F6, что приводит к изменению цветов и поддерживанию яркости ниже 50-процентов от максимального уровня. Поскольку увеличение RTCLOK+1 происходит каждые 256/60 доли секунды, а младший бит COLRSH не рассматривается, изменение яркостей цвета будет производится каждые 8.3 секунды.

B10 ATRACT (0040,1) -- Таймер флага неработающего режима.

ATRACT представляет собой таймер (и флаг), управляющий запуском и окончанием неработающего режима. Всякий раз при нажатии клавиши на клавиатуре программа обработки прерываний по запросу (IRQ) устанавливает нулевое значение ATRACT, тем самым сбрасывая неработающий режим. Соответствующим образом используется клавиша (BREAK). В время первой стадии процесса VBLANK ATRACT увеличивается на единицу каждые 4 секунды. Если значение превосходит 127 (после 9 минут без подачи сигнала с клавиатуры), ATRACT устанвливается равным !FE, и это значение сохраняется до выхода из этого режима. Поскольку выход из нерабочего режима осуществляется только на основании поступления сигнала с клавиатуры, некоторым пользователям будет желательно, чтобы ATRACT сбрасывался при включении в работу контроллера Атари, управляемым пользователем последовательной шины ввода/вывода или при подачи каких-либо других признаков жизни.

B11 COLRSH` (004F,1) -- Маска цветового сдвига.

Когда неработающий режим не активизирован, значение COLRSH равно нулю и данная переменная не оказывает никакого влияния на цвета на экране. Когда же неработающий режим активизирован, в COLRSH хранится текущее значение среднего разряда переменной таймера (RTCLOK+1).

B12 DRKMSK` (004E,1) -- Маска затемнения (яркости).

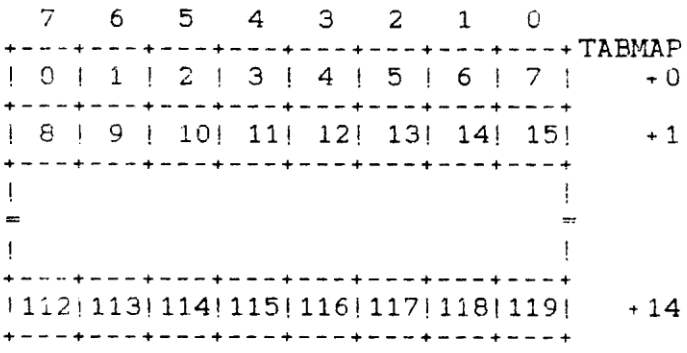
Когда неработающий режим не активизирован значение DRKMSK имеет значение !FE. Иначе значение этой величины равно !F6, что вызывает сброс старшего бита на ноль, гарантируя тем самым, что яркость не превзойдет 50 процентов.

Табуляция.

Использование табуляции совместно с экранным редактором обсуждалось в разделе 5.

B13 TABMAP (02A3,15) -- Карта установки границ табуляции.

Установка границ табуляции осуществляется на 15-байтовой карте (120 бит), где 1 означает, что табуляция установлена. На нижеприведенной диаграмме указано размещение отдельных бит на позициях табуляции.



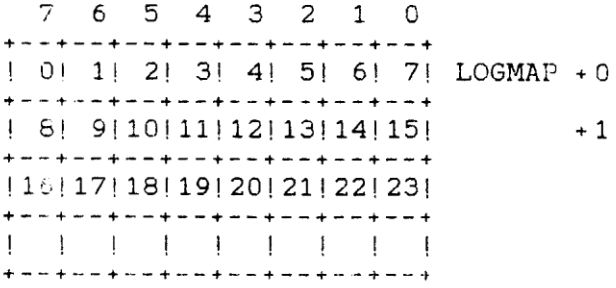
Всякий раз при открытии дисплея или экранного редактора данная карта инициализируется. После инициализации в каждом байте сохраняется значение 101, соответствующее границам табуляции, принимаемым по умолчанию в позициях 7,15,23 и т.д.

Логические текстовые строки.

Текстовой экран подразделен на логические строки, каждая из которых включает от одной до трех физических строк. Первоначально экран при инициализации разбивается на 24 логические строки, каждой из которых соответствует одна физическая. Но ввод данных и/или вставка данных в текст может привести к увеличению логической строки до размера двух или трех физических строк.

B14 LOGMAP` (02B2,1) -- Карта начала логических строк.

В данной четырехбайтовой схеме хранятся номера позиций каждой логической строки, в которых начинаются новые физические строки. Бит, установленный в единицу, указывает начало логической строки. Ниже приведена диаграмма, на которой показано размещение отдельных бит по номерам физических строк.



Все биты устанавливаются равными единице всякий раз, при открытии и очищении текстового экрана. После этого карта редактируется по мере открытия логических строк, редактируемых и уничтожаемых на экране.

B15 LOGCOL` (0063,1) -- Номер столбца в логической строке, в которой находится курсор.

LOGCOL содержит номер столбца в логической строке, в котором находится курсор. Заметим, что логическая строка может включать до трех физических строк. Данная переменная предназначена для внутреннего использования ее дисплеем.

Разбиение экрана на подобласти.

Дисплей совместно с экранным редактором выполняют операции по переводу экрана в оконный режим (см. раздел 5), в котором часть экрана управляется дисплеем, а другая часть представляет собой 4 физические строки в текстовом окошке в нижней части экрана, управляемые экранным редактором.

B16 BOTSCR` (02BF,1) -- Счетчик строк текста на экране.

В BOTSCRe хранится число строк текста на экране: 24 для нулевого режима и 4 для режима с разбиением экрана. Тесты проводятся для величин 4 и 24.

Функции DRAW/FILL.

Алгоритм функции DRAW приведен ниже на языке Паскаль (PASCAL), транслированным с ассемблера.

```
MEWROW:=ROWCRS;  NEWCOL:=COLCRS;
DELTAR:=ABS(NEWROW-OLDROW);
ROWINC:=SING(NEWROW-OLDROW);
DELTAC:=ABS(NEWCOL-OLDCOL);
COLINC:=SING(NEWCOL-OLDCOL);
ROWAC:=0;  COLAC:=0;
ROWCRS:=OLDROW;  COLCRS:=OLDCOL;
COUNTR:=MAX(DELTAC,DELTAR);
ENDPT:=COUNTER;
IF COUNTER=DELTAC
 THEN
   ROWAC:=ENDPT DIV 2
 ELSE
   COLAC:=ENDPT DIV 2
WHILE COUNTR)0 DO
 BEGIN
   ROWAC:=ROWAC+DELTAR;
   IF ROWAC)=ENDPT
    THEN
      BEGIN
        ROWAC:=ROWAC-ENDPT;
        ROWCRS:=ROWCRS+ROWINC
      END;
   COLAC:=COLAC+DELTAC;
   IF COLAC)=ENDPT
    THEN
      BEGIN
        COLAC:=COLAC-ENDPT;
        COLCRS:=COLCRS+COLINC
      END;
   PLOT POINT;
   IF FILFLG()0 THEN FILL LINE;
   COUNTR:=COUNTR-1;
 END;
```

Алгоритм функции FILL (FILL LINE) вкратце описан в разделе 5.

B17 FILDAT (02FD,1) -- Данные FILL.

FILDAT содержит значение области данных в процессе вызова команды FILL, описанной в разделе 5.

B18 FILFLG` (02B7,1) -- Флаг FILL.

FILFLG содержит код для дисплея, указывающий какая из двух операций активна:
FILL (FILFLG()0) или DRAW (FILFLG=0).

B19 NEWROW` (0060,1) или NEWCOL` (0061,2) -- Начальная точка.

NEWROW и NEWCOL при инициализации принимают значения ROWCRS и COLCRS, указывающие конечную точку для команд DRAW и FILL. Это производится таким образом, что значения ROWCRS и COLCRS могут быть изменены в процессе выполнения команды.

B20 HOLD4` (02BC,1) -- Временная память.

HOLD4 используется для сохранения и восстановления величин в ATACHR в процессе выполнения FILL. ATACHR временно устанавливается равным значению в FILDAT для завершения выполнения команды.

B21 ROWINC` (0079,1) и COLINC` (007A,1) -- Инкремент/ декремент строки/столбца.

ROWINC и COLINC представляют собой значения изменения номера строки и столбца. Они устанавливаются в +1 или -1 для указания направления рисунка. ROWING и COLING представляют собой знаки NEWROW-COWCRS и NEWCOL-COLCRS соответственно.

B22 DELTAR` (0076,1) и DELTAC` (0077,2) -- Дельта строки и дельта столбца.

DELTAR и DELTAC содержат абсолютные значения NEWROW-COWCRS и NEWCOL-COLCRS соответственно. Совместно с ROWINC и COLINC эти величины определяют изгиб линии, которая будет нарисована.

B23 COUNTR` (007E,2) -- Счетчик итераций изображения.

COUNTR первоначально содержит наибольшее значение из DELTAR и DELTAC, представляющее собой число итераций, необходимых для изображения требуемой линии. После изображения каждой точки линии значение COUNTR уменьшается на единицу, до тех пор пока не станет равным нулю.

B24 ROWAC` (0070,2) и COLAC` (0072,2) -- Аккумуляторы.

ROWAC и COLAC являются рабочими аккумуляторами, управляющими положением (позициями строки и столбца) изображаемой точкой и функцией инкремента (или декремента).

B25 ENDPT` (0074,2) -- Длина строки.

ENDPT содержит наибольшее из значений DELTAR и DELTAC и используется совместно с ROWAC/COLAC и DELTAR/DELTAC для управления изображением точек изображаемой линии.

Изображение управляющих символов.

Иногда бывает необходимо получить изображение управляющих кодов ATASCII в их графической форме, без выполнения ими управляющих функций. При выводе на экранный редактор это изображение можно получить в двух формах:
1) Форма данных, в которой каждому управляющему символу предшествует специальный символ (ESC).
2) Форма управления режимом.

DMASK является специальной маской, содержащая нули во всех разрядах, которые не связаны с определенным пикселем, и 1 во всех разрядах, которые связаны с рабочим пикселем. DMASK может содержать следующие двоичные величины:

```
11111111 -- 1 и 2 экранные режимы;
            один пиксель на байт.
11110000 -- Экранные режимы с 9-11;
00001111    два пикселя на байт.
11000000 -- Экранные режимы 3,5,7;
00110000    четыре пикселя на байт.
00001100
00000011
10000000 -- Экранные режимы 4,6,8;
01000000    восемь пикселей на байт.
........
00000010
00000001
```

B29 SHFAMT` (006F,1) -- Выравнивание пикселей.

SHFAMT указывает число сдвигов крайнего правого пикселя, необходимое для вывода данных, или число сдвигов вводимых данных для выравнивания их по правому краю при вводе. Значение обычно совпадает со значением DMASK до процесса выравнивания.

Внутренние рабочие переменные.

B30 HOLD1` (0051,1) -- Временное хранение.

B31 HOLD2` (029F,1) -- Временное хранение.

B32 HOLD3` (029D,1) -- Временное хранение.

B33 TMPCHR` (0050,1) -- Временное хранение.

B34 DSTAT` (004C,1) -- Состояние дисплея.

B35 DINDEX (0057,1) -- Режим дисплея.

DINDEX содержит текущий экранный режим, получаемый из четырех младших самого последнего байта AUX1 для операции OPEN.

B36 SAVMSC (0058,2) -- Адрес экранной памяти.

SAVMSC содержит меньший адрес области экранной памяти, данные по этому адресу изображаются в левом верхнем углу экрана.

B37 OLDCHR` (005D,1) -- Сохранение/восст. символа в позиции курсора

В OLDCHRe сохраняется значение символа, находящегося в позиции курсора в тексте. Данная переменная используется для восстановления первоначального символа при перемещении курсора.

B38 OLDADR` (005E,2) -- Адрес памяти курсора.

В OLDADRe сохраняется адрес текущей позиции курсора в тексте. Данная переменая используется совместно с OLDCHR (B37) для восстановления первоначального символа при перемещении курсора.

B39 ADRESS` (0064,2) -- Временное хранение.

B40 MLTTMP/OPNTMP/TOADR` (0066,2) -- Временное хранение.

B41 SAVADR/FRMADR` (0068,2) -- Временное хранение.

B42 BUFCNT` (006B,1) -- Размер текущей логической строки экранного редактора.

B43 BUFSTR` (006C,2) -- Временное хранение.

B44 SWPFLG` (007B,1) -- Управление курсором в оконном режиме.

В оконном режиме данные графического курсора и данные курсора текстового окошка часто обмениваются, чтобы получить значения переменных ROWCRS-OLDADR.

```
ROWCRS B2 -------  TXTROW B4
COLCRS B2 -------  TXTCOL B4
DINDEX B35 ------  TINDEX B49
SAVMSC B36 ------  TXTMSC B52
OLDROW B3 -------    "      "
OLDCOL B3 -------    "      "
OLDCHR B37 ------    "      "
OLDADR B38 ------    "      "
```

SWPFLG используется для отслеживания набора данных, находящегося в области ROWCRS-OLDADR. SWPFLG устанавливается равным !FF в том случае, если данные курсора текстового окна находятся в основной области. В противном случае значение SWPFLG равно нулю.

B45 INSDAT` (007D,1) -- Временное хранение.

B46 TMPROW` (02B8,1) и TMPCOL` (02B9,2) -- Временное хранение.

B47 TMPLBT` (02A1,1) -- Временное хранение.

B48 SUBTMP` (029E,1) -- Временное хранение.

B49 TINDEX` (0293,1) -- Экранный режим текстового окна.

TINDEX является эквивалентом PINDEX для текстового окна и обычно равен нулю, когда SWPFLG имеет нулевое значение (см. B44).

B50 BITMSK` (006E,1) -- Временное хранение.

B51 LINBUF` (0247,40) -- Буфер физической строки.

LINBUF используется в качестве временного буфера одной физической строки текста во время перемещения данных экранным редактором.

B52 TXTMSC (0294,2) -- Адрес памяти разбиения экрана.

TXTMSC аналогичен SAVMSC (B36), только для текстового окна в режиме разделения экрана. Дополнительную информацию можно найти в B44.

B53 TXTOLD` (0296,6) -- Данные курсора для оконного режима.

Дополнительную информацию можно найти в B44.

Преобразование кодов символов во внутренние коды.

Для хранения обрабатываемого в настоящий момент символа используются две переменные (как для считывания, так и для записи). ATACHR содержит величину, поступающую из или в CIO, а CHAR содержит внутренний код, соответствующий значению ATACHR. Поскольку аппаратные средства не в состояние непосредственно интерпретировать символы ATASCII, то нижеприведенные преобразования необходимо выполнять как для записи так и для считывания всех текстовых данных:

Код ATASCII	Внутренний код
00-1F	40-5F
20-3F	00-1F
40-5F	20-3F
60-7F	60-7F
80-9F	C0-DF
A0-BF	80-9F
C0-DF	A0-BF
E0-FF	E0-FF

Дополнительную информацию можно найти в B26.

B54 ATACHR (02FB,1) -- Последний символ ATASCII или изображенная точка.

ATACHR содержит значение кода ATASCII для символа, который был считан или записан последним, или значение графической точки. Данная переменная может также рассматриваться как параметр команд FILL и DRAW.

B55 CHAR` (02FA,1) -- Внутренний код символа.

CHAR содержит значение внутреннего кода символа, который был считан или записан последним.

C. Дисковод.

Описание резидентного дисковода можно найти в разделе 5.

C1 BUFADR` (0015,2) -- Указатель на буфер данных.

Работа BUFADR аналогична временному указателю нулевой странице на текущий буфер дискеты.

C2 DSKTIM` (0246,1) -- Время простоя при форматировании дискеты.

DSKTIM содержит величину простоя для переменной DTIMLO последовательности вызова SIO (см. раздел 9). DSKTIM устанавливается при инициализации равным 160 (что соответствует простою в 171 секунду) и корректируется после каждого запроса о состоянии дискеты. В ней хранится величина, возвращенная в третьем байте фрейма состояния (см. раздел 5). Заметим, что все операции с дискетой, кроме форматирования, имеют фиксированное 7-секундное время простоя, устанавливаемое дисководом.

D. Кассетный магнитофон.

Общее описание кассетного магнитофона можно найти в разделе 5. Кассетное устройство использует аппаратные средства последовательной шины ввода/вывода, но не связывается с протоколом последовательной шины, как было сказано в разделе 9. Следовательно, программа последовательного ввода/вывода (SIO) имеет внутри специальный код для кассеты. Некоторые переменные в данном подразделе используются SIO, а некоторые кассетным устройством.

Определение скорости в бодах.

Номинальная входная скорость должна быть равной 600 бод, но при необходимости может изменяться программой SIO для различных видов моторов, вытянутой ленты и т.п. Начало каждой записи содержит изменяемый набор нулей и единиц и используется только для коррекции скорости путем изменения времени считывания фиксированного набора бит. Определяется истинная скорость в бодах, после чего происходит соответствующая регулировка аппаратных средств. Скорость ввода в бодах изменяется в пределах от 318 до 1407 бод и может регулироваться при помощи данной технологии. Скорость ввода в бодах регулируется путем установки счетчика POKEY, управляющего временем считывания контрольных бит.

D1 CBAVDL` (02EE,1) и CBAUDH` (02EF,1) -- Скорость ленты в бодах.

При инициализации устанавливаются равными 05CC, что соответствует 600 бод. После вычисления скорости в бодах в данных переменных будет содержаться значение счетчика POKEY для скорректированной скорости в бодах.

D2 TIMFLG` (0317,1) -- Флаг простоя при определении скорости в бодах.

TIMFLG используется SIO для игнорирования неправильного определения скорости в бодах. Первоначально флаг устанавливается равным 1, и если он становится равным нулю (по истечении 2 секунд) до того, как считан первый байт записи на кассете, операция будет прервана (см. H24).

D3 TIMER1` (030C,2) и TIMER2` (0340,2) -- Таймеры скорости в бодах.

Данные таймеры указывают время ссылки на начало и конец периода считывания фиксированного набора бит. В первый байт каждого таймера считывается из чипа ANTIC текущее значение счетчика вертикальных строк. Во второй байт каждого таймера считывается текущее значение младшего байта часов реального времени 0C (RTCLOK+2). Разность значений таймеров преобразуется в растровую пару чисел и используется в дальнейшем для создания наглядной таблицы с интерполяцией для определения значений CBAUDL и CBAUDH.

D4 ADDCOR` (030E,1) -- Переменная регулировки интерполяции.

ADDCOR является временной переменной, используемой для интерполяционного вычисления вышеуказанного алгоритма.

D5 TEMP1` (0312,2) -- Временное хранение.

D6 TEMP3` (0315,1) -- Временное хранение.

D7 SAVIO` (0316,1) -- Обнаружение данных для последовательного ввода.

SAVIO используется для хранения состояния 4 бита переменной SKSTAT (D20F) и обнаружения (и последующей корректировки) поступления каждого бита.

Кассетный режим.

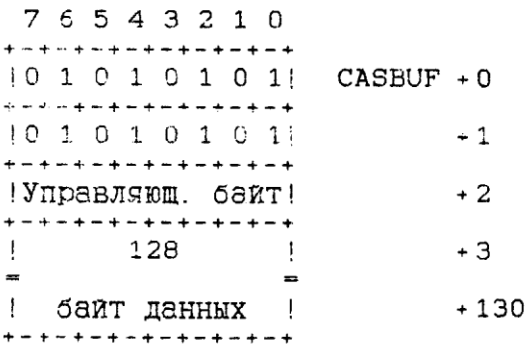
D8 CASFLG` (030F,1) -- Флаг ввода/вывода с кассеты.

CASFLG служит для внутреннего использования SIO для управления прохождением программы через разделенный код. Нулевое значение указывает, что текущая операция является стандартной операцией последовательного ввода/вывода, а ненулевое значение указывает, что текущая операция - операция с кассетой.

Кассетный буфер.

D9 CASBUF` (03FD,131) -- Буфер записи на кассету.

CASBUF представляет собой буфер, используемый кассетным магнитофоном для установки и распаковки записей данных на кассету, и логику загрузки с кассеты при инициализации. Ниже приведен формат стандартной кассетной записи в буфере.



Формат стандартной записи на кассете приведен в разделе 5.

D10 BLIM` (028A,1) -- Размер записи данных на кассете.

BLIM содержит счетчик байт данных, считываемых в настоящий момент с кассеты. Значение BLIM изменяется в диапазоне от 1 до 126, в зависимости от управляющего байта записи, так как это было описано в разделе 5.

D11 BPTR` (003D,1) -- Указатель записи данных на кассете.

BPTR содержит указатель на считываемый участок данных в записи на кассете. Значение изменяется от 0 до текущего значения BLIM. Когда BPTR становится равным BLIM, это свидетельствует о том, что буфер (CASBUF) заполнен, если производилась запись, и пуст, если производилось чтение.

Внутренние рабочие переменые.

D12 FEOF` (003F,1) -- Флаг конца файла на кассете.

FEOF используется кассетным устройством для обнаружения условия конца файла (Управляющий байт = !FE). Нулевое значение указывает, что ситуация EOF еще не обнаружена, а ненулевое значение указывает на наличие данной ситуации. Флаг перезагружается после каждой операции OPEN.

D13 FTYPE` (003E,1) -- Тип интервала между записями.

FTYPE представляет собой копию ICAX2Z из команды OPEN и указывает выбранный тип интервала между записями. Положительное число соответствует нормальному интервалу между записями, а отрицательное значение соответствует расширенным интервалам.

D14 WMODE` (0289,1) -- Флаг режима чтение/запись с кассеты.

WMODE используется кассетным устройством для указания того, является ли текущая операция чтением или записью. Нулевое значение соответствует чтению, а значение $80 соответствует записи.

D15 FREQ` (0040,1) -- Счетчик гудков.

FREQ используется для хранения и подсчета числа гудков, запрашиваемых программой BEEP кассетного устройства, необходимых для операции OPEN.

E. Клавиатура.

Общее описание можно найти в разделе 5.

Считывание и отжатие клавиши.

Считывание регистра кода консоли осуществляется в ответ на прерывание по запросу (IRQ), генерируемое всякий раз при обнаружении нажатия клавиши. Код клавиш сравнивается с кодом клавиши, принятым до этого (CH1). Если коды не совпадают, то принимается новый код, который помещается в FIFO (CH) и в переменную кода предыдущей клавиши (CH1). Если коды совпадают, то новый код принимается только в том случае, если имела место задержка отжатия соответствующей клавиши со временем принятия предыдущей величины. Клавиатура получает все данные клавиши из CH. Всякий раз при выделении кода из однобайтового FIFO устройство помещает в него величину $FF, для индикации того, что код уже считан. Дальнейшая обработка кодов клавиши обсуждается в разделе 5.

E1 CH1` (02F2,1) -- Код предыдущего символа клавиатуры.

CH1 содержит значение кода клавиши, считанного последним.

E2 KEYDEL` (02F1,1) -- Таймер задержки отжатия.

Значение KEYDEL устанавливается равным 3 всякий раз при принятии кода клавиши и уменьшается на единицу каждую 60 долю секунды в процессе второй стадии VBLANK (пока не достигнет нулевого значения).

E3 CH (02FC,1) -- Код символа клавиатуры FIFO.

CH представляет собой однобайтовый FIFO, содержащий либо значение $FF (указывающее, что FIFO пусто). FIFO обычно считывается клавиатурой, но также возможно осуществить считывание из программы пользователя. Данные клавиши могут быть помещены в CH логикой автоматического повтора, как это было описано в разделе, относящемуся к E8.

Специальные функции.

Старт/стоп.

Вывод данных в текстовом или графическом режимах на дисплей или через экранный редактор может быть приостановлен (без потери данных) путем использования комбинации клавиш (CTRL)(1). Нажатие каждой клавиши изменяет значение флага, управляемого вышеуказанными устройствами. Когда значение флага отлично от нуля, устройства ожидают, пока оно снова не станет равным нулю, чтобы продолжить вывод.

E4 SSFLAG (02FF,1) -- Флаг старт/стоп.

Флаг обычно равен нулю, что соответствует выводу без остановки. Флаг преобразуется в дополнительный код при онаружении программой обработки IRQ комбинации (CTRL)(1). Нулевое значение флага устанавливается при включении питания, перезагрузке или обработке клавиши (BREAK).

Клавиша (BREAK).

E5 BRKKEY (0011,1) -- Флаг клавиши (BREAK).

BRKKEY используется для индикации нажатия клавиши (BREAK). Обычно значение этой переменной отлично от нуля и устанавливается равным нулю при нажатии клавиши (BREAK). Программе, обнаруживающей и обрабатывающей условие клавиши (BREAK) (флаг=0), надлежит снова установить ненулевое значение флага. BRKKEY управляется следующими программами ОС: клавиатурой, дисплеем, экранным редактором, кассетным устройством и т.п. Обнаружение нажатия (BREAK) в процессе операции ввода/вывода вызовет остановку операции и возврат пользователю значения !80. Значение флага устанавливается равным отличному от нуля числу при включении питания, перезагрузке системы или при прерывании ожидаемой операции ввода/вывода.

Управляющие клавиши SHIFT и CONTROL.

Управление клавиатурой дает возможность генерации кодов алфавитных символов с A по Z в трех различных режимах:
1) нормальный,
2) символы нижнего регистра,
3) управляющие символы.
В нормальном режиме клавиши постоянных символов алфавита генерируют кода ATASCII символов нижнего регистра (!61-7A). В режиме символов нижнего регистра все клавиши постоянных символов алфавита генерируют коды ATASCII символов верхнего регистра (!41-5A). В управляющем режиме все клавиши постоянных символов алфавита генерируют коды управляющих символов ATASCII (!01-1A). Во всех режимах внесения изменений (добавлением нажатия клавиш (SHIFT) и (CONTROL)) позволит сгенерировать любой необходимый код.

E6 SHFLOK (02BE,1) -- Флаг управления (SHIFT)/(CONTROL).

Обычно в SHFLOK хранится одно из трех значений.
!00 - нормальный режим.
!40 - символы нижнего регистра.
!80 - управляющие символы.
SHFLOK устанавливается в !40 при перезагрузке системы и включении питания и изменяется ОС только в случае нажатии клавиши (CAPS) (или ее одной, или совместно с клавишами (SHIFT) или (CONTROL)).

E7 HOLDCH` (007C,1) -- Переменная хранения символа.

HOLDCH используется для хранения значения текущего символа до обработки логики (SHIFT)/(CONTROL).

Автоповтор.

Функция автоматического повторения вызывается при продолжительном нажатии клавиши на клавиатуре повторением кода клавиши на клавиатуре 10 раз в секунду, после 0.5-секундной задержки. Переменная таймер SRTIMR используется для управления первоначальной задержкой и скоростью повторения. Когда значение SRTIMR становится равным нулю при нажатии клавиши, значение кода клавиши располагается в FIFO (CH). Данные действия производятся на второй стадии обработки VBLANK.

E8 STIMR` (022B,1) -- Таймер автоматического повторения.

STIMR устанавливается двумя независимыми процессами:
1) программой обработки IRQ от клавиатуры, устанавливающей значение первоначальной задержки.
2) программой второй стадии VBLANK, устанавливающей скорость повторения, уменьшающей значение таймера и выполняющей логические действия автоматического повторения.

Управление инверсным изображением.

Клавиатура позволяет непосредственно генерировать более, чем половину всех 256 кодов ATASCII. Однако, коды $80-9A и коды $A0-FC могут быть сгенерированы только с помощью "режима инверсного изображения". В данном режиме все клавиши ATARI действуют как триггеры ON/OFF (вкл/выкл), и все символы (за исключением управляющих символов экранного редактора) будут выданы в инверсном режиме.

E9 INVFLG (02B6,1) -- Флаг инверсного изображения.

INVFLG обычно равен нулю, что соответствует генерации обычных кодов ATASCII (бит 7 = 0). Всякий раз, когда значение INVFLG становится отличным от нуля, генерируются инверсные коды ATASCII (бит 7 = 1). К специальным управляющим кодам это не относится. INVFLG устанавливается равным нулю при включении питания и перезагрузке системы. Клавиатура инвертирует 7 бит INVFLG при нажатии клавиши инверсного изображения. Биты младших разрядов не изменяются и принимаются равными нулю.

Клавиши консоли: (SELECT),(START),(OPTION).

Клавиши консоли считываются непосредственно из аппаратного регистра CONSOL (D10F). См. "Руководство по аппаратному обеспечению компьютера ATARI".

F. Принтер.

Общее описание принтера можно найти в разделе 5.

Буфер принтера.

F1 PRNBUF` (03C0,40) -- Буфер записей принтера.

PRNBUF представляет собой буфер, используемый принтером для упаковки данных, поступающих на принтер с целью пересылки их на контроллер устройства. Длина буфера составляет 40 байт, в которых не содержится ничего за исключением данных принтера.

F2 PBUFSZ` (001E,1) -- Размер записи принтера.

PBUFSZ содержит размер записи принтера в выбранном режиме. Режимы и соответствующие им размеры (в десятичных числах) приведены ниже.

Нормальный	40.
Двойная ширина	20 (не поддерживается устройством).
С наклоном	29.
Запрос ссостояния	4.

F3 PTEMP` (001F,1) -- Временное хранение данных принтера.

PTEMP используется принтером для временного хранения значения символа, выводимого на принтер.

F5 PTIMOT` (001C,1) -- Значение простоя принтера.

PTIMOT содержит значение простоя для переменной последовательности вызова SIO DTIMLO (см. Раздел 9). PTIMOT устанавливается равным 30 (что соответствует 32 секундам простоя) во время инициализации, а затем корректируется после каждой операции запроса о состоянии принтера таким образом, чтобы содержать значение, возвращаемое в третьем байте фрейма состояния (см. раздел 5).

G. Программа центрального ввода/вывода (CIO).

Описание программы CIO приведено в разделе 5.

Параметры, вызываемые пользователем.

Параметры вызова CIO проходят сначала через управляющий блок ввода/вывода, хотя дополнительная информация о состоянии устройства может быть возвращена в DVSTAT, а информация устройства добывается из таблицы устройств (HATABS).

Управляющий блок ввода/вывода.

IOCB - имя, присвоенное 16 байтам, связанных друг с другом восьмью управляющими структурами. См. Раздел 5.

G1 IOCB (0340,16) -- Управляющий блок ввода/вывода.

Метка IOCB представляет собой местоположение первого байта первого IOCB в базе данных. Для переменных с индексами от G2 до G10 приведены адреса только для нулевого IOCB. Адреса остальных IOCB приведены ниже.

```
0340-034F  IOCB (0
0350-035F  IOCB (1
0360-036F  IOCB (2
0370-037F  IOCB (3
0380-038F  IOCB (4
0390-039F  IOCB (5
03A0-03AF  IOCB (6
03B0-03BF  IOCB (7
```

G2 ICHID (0340,1) -- ID устройства.

См. Раздел 5. Инициализируется в $FF при включении питания и перезагрузке системы.

G3 ICDNO (0341,1) -- Номер устройства.

См. Раздел 5.

G4 ICCOM (0342,2) -- Командный байт.

См. Раздел 5.

G5 ICSTA (0343,1) -- Состояние.

См. Раздел 5.

G6 ICBAL,ICBAH (0344,2) -- Адрес буфера.

См. Раздел 5.

G7 ICPTL,ICPTH (0346,2) -- Вектор PUT BYTE.

См. Раздел 5. Инициализируется для указания на программу "IOCB не открыт" при включении питания и перезагрузке системы.

G8 ICBLL,ICBLH (0348,2) -- Длина буфера/счетчика байт.

См. Раздел 5.

G9 ICAX1,ICAX2 (034A,2) -- Вспомогательная информация.

См. Раздел 5.

G10 ICSPR (034C,4) -- Свободные байты для нужд устройства.

Для этих четырех байт не производится присваивание. Устройство, связанное с IOCB, может и не использовать эти байты.

G11 DVSTAT (02E4,4) -- Состояние устройства.

Описание команды GET STATUS см. в разделе 5.

Таблица устройств.

G12 HATABS (031A,38) -- Таблица устройств.

Описание таблицы устройств приводится в разделе 9.

Параметры интерфейса CIO с устройствами.

Связь между CIO и устройством осуществляется при помощи машинных регистров 6502 и структуры данных, называемой IOCB нулевой страницы ZIOCB. ZIOCB представляет собой копию определенного IOCB, используемого текущей операцией.

IOCB нулевой страницы.

G13 ZIOCB (0020,16) -- IOCB нулевой страницы.

IOCB нулевой страницы представляет собой точную копию (за исключением некоторых указанных ниже моментов) IOCB, определенного регистром X 6502 после входа в CIO. CIO копирует IOCB более высокого уровня в IOCB нулевой страницы, выполняет указанную функцию, перемещает IOCB нулевой страницы обратно в IOCB более высокого уровня, а затем возвращается в точку вызова. Хотя IOCB нулевой страницы и IOCB более высокого уровня по определению состоят из 16 байтов, только первые 12 из них перемещаются CIO.

G14 ICHIDZ (0020,1) -- Номер устройства.

См. Раздел 5. При CLOSE устанавливается равным $FF.

G15 ICDNOZ (0021,1) -- Номер привода устройства.

См. Раздел 5.

G16 ICCOMZ (0022,1) -- Командный байт.

См. Раздел 5.

G17 ICSTAZ (0023,1) -- Байт состояния.

См. Раздел 5.

G18 ICBALZ,ICBAHZ (0024,2) -- Адрес буфера.

См. Раздел 5. Данная переменная-указатель изменяется CIO в процессе обработки некоторых команд. Однако перед возвращением пользователю восстанавливается первоначальное значение.

G19 ICPTLZ,IPTHZ

См. Раздел 5. При CLOSE устанавливается указывающим на программу CIO "IOCB не открыт".

G20 ICBLLZ,ICBLHZ (0028,2) -- Длина буфера/счетчик байт.

См. Раздел 5. Данная двухбайтовая переменная, содержащая длину буфера, изменяет CIO в процессе обработки некоторых команд, после чего перед возвращением пользователю в ней размещается преобразованный счетчик байт.

G21 ICAX1Z, ICAX2Z (002A,2) -- Вспомогательная информация.

См. Раздел 5.

G22 ICSPRZ (ICIDNO,CIOCHR) (002C,4) -- Рабочие переменные CIO.

ICSPRZ и ICSPRZ+1 используются CIO при получении соответствующего устройству входной точки из таблицы векторов устройств (см. Раздел 9). ICSPRZ+2 также имеет метку ICIDNO и сохраняет значение регистра X 6502 из входной точки CIO. Регистр X загружается из ICIDNO по возвращению CIO пользователю. ICSPRZ+3 имеет метку CIOCHR и сохраняет значение регистра A 6502 из входной точки CIO, за исключением команд считывания данных. В этом случае байт, считанный последним располагается в CIOCHR. Регистр A 6502 загружается из CIOCHR по возвращении CIO в точку вызова.

Внутренние рабочие переменные.

G23 ICCOMT` (0017 1) -- Указатель таблицы команд.

ICCOMT используется как указатель на таблицу внутренних команд CIO, которая преобразует значение командных байт в сдвиги входных точек устройств (дополнительные сведения см. в разделе 9). ICCOMT содержит значение ICCOMZ, за исключением тех случаев, когда ICCOMZ превосходит !0E. В этом случае значение ICCOMT устанавливается равным !0E.

G24 ICIDNO` (002E,1) -- Сохранение/восстановление вызова CIO регистра X.

См. G22.

G25 CIOCHR` (002F,1) -- Сохранение/восстановление вызова CIO регистра A.

См. G22.

H. Программа последовательного ввода/вывода (SIO).

Информацию, относящуюся к SIO см. в разделе 9.

Параметры, вызываемые пользователем.

Параметры вызова SIO проходят через блок управления устройством, хотя существует дополнительная опция "шумной шины", доступная через независимую переменную.

Блок управления устройством.

H1 DCB (0300,12) -- Блок управления устройством.

Имя DCB относится к 12 байтам в ячейках 0300-030B. Данные байты моделируют механизм перемещения параметров и SIO, их описание можно найти ниже.

H2 DDEVIC (0300,1) -- ID устройства шины.

См. Раздел 9.

H3 DUNIT (0301,1) -- Номер блока устройства.

См. Раздел 9.

H4 DCOMND (0302,1) -- Команда устройству.

См. Раздел 9.

H5 DSTATA (0303.1) -- Состояние устройства.

См. Раздел 9.

H6 DBUFLO,DBUFHI (0304.2) -- Адрес буфера устройства.

См. Раздел 9.

H7 DT1MLO (0306.1) -- Время простоя устройства.

См. Раздел 9.

H8 DBYTOL,DBYTHI (0308.2) -- Длина буфера/счетчик байт.

См. Раздел 9.

H9 DAUX1,DAUX2 (030A.2) -- Вспомогательная информация.

См. Раздел 9.

Управление звуковой шиной.

H10 SOUNDR (0041.1) -- Флаг тихого/шумного ввода/вывода.

SOUNDR представляет собой флаг, используемый для индексации SIO генерировать ли аидио сигнал в схеме телевизора при вводе/выводе через последовательную шину. Нулевое значение SOUNDR указывает, что звук не воспроизводится, а ненулевое значение указывает на возможность наличия звука при включениии питания и перезагрузке системы. SOUNDR устанавливается SIO равным 3.

Управление последовательной шиной.

Логика повторной попытки.

В случае, если первая попытка содержала ошибки, SIO предпримет полную повторную попытку выполнить команду. Полная повторная попытка состоит из 14 попыток послать (и подтвердить) командный фрейм и единственной попытке получить байт COMPLETE и фрейм данных.

H11 CRETRY` (0036.1) -- Счетчик повторных отправлений командного фрейма.

CRETRY управляет внутренней меткой логики повторной попытки, связанной с передачей, получением и полтверждением командного фрейма. CRETRY устанавливается SIO равным 13 в начале поступления каждой команды, давая тем самым возможность после первоначальной попытки еще 13 дополнительных.

H12 DRETRY` (0037.1) -- Счетчик повторных попыток устройства.

DRETRY управляет внешней меткой логики повторного запуска, связанной с проведением повторного запуска после провала получения подтверждения командкого фрейма. DRETRY устанавливается SIO равным единице при входе, давая возможность проведения первого повторного запуска помимо основной попытки.

Контрольная сумма.

В протоколе последовательной шины определено, что все командные фреймы и фреймы данных обязаны иметь в своем составе байт контрольной суммы. Данный байт представляет собой арифметическую сумму (с соответствующим переносом) всех остальных байт, входящих во фрейм.

H13 CHKSUM` (0031,1) -- Значение контрольной суммы.

CHRSUM содержит контрольную форму фрейма, вычисляемую SIO при каждом перемещении фрейма.

H14 CHKSNT` (003B,1) -- Флаг пересылки контрольной суммы.

CHKSNT указывает сервисной программе обработки прерываний при передаче через последовательную шину был ли послан байт контрольной суммы фрейма. Нулевое значение CHKSNT указывает, что байт контрольной суммы еще не был послан. После посылки контрольной суммы CHKSNT становится отличным от нуля.

H15 NOCKSM` (003C,1) -- Флаг, указывающий, что за данными не следует контрольная сумма.

NOCKSM представляет собой флаг, используемый для связи между программой SIO высшего уровня с сервисной программой обработки прерываний передачи на последовательную шину, сообщающей о том, что после следующего ввода контрольная сумма не последует.

Буферизация данных.

Управление общим буфером.

H16 BUFRLO` (0032,1) и BUFRHI` (0033,1) -- Адрес следующего буфера.

В BUFRLO и BUFRHI хранится указатель на местоположение следующего буфера, из которого будет производиться чтение или запись в него. При перемещении командного фрейма указатель устанавливается на величину, содержащуюся в параметрах вызова SIO DBUFLO и DBUFHI, а затем в процессе перемещения данных через последовательную шину уменьшается на единицу сервисной программой обработки прерываний.

H17 BFENLO` (0034,1) и BFENHI (0035,1) -- Адрес конца буфера.

BFENLO и BFENHI формируют указатель на байт, следующий за последним байтом отсылаемого или получаемого фрейма данных (исключая контрольную сумму). BFENLO и BFENHI представляет собой арифметическую сумму BUFRLO и BUFRHI с размером фрейма минус 1.

Выходной буфер командного фрейма.

Формат и описание команд можно найти в разделе 9.

H18 CDEVIC` (023A,1) -- ID устройства в командном фрейме.

Значение CDEVIC устанавливается равным значению, получаемому сложением параметра вызова SIO DDEVIC с DUNIT минус 1.

H19 CCOMND` (023B,1) -- Команда в командном фрейме.

Значение CCOMND устанавливается равным значению параметра вызова SIO DCOMND.

H20 CAUX1` (023C,1) и CAUX2` (023D,1) -- Вспомогательная информация.

Значение CAUX1 и CAUX2 устанавливается равным параметрам вызова SIO DAUX1 и DAUX2 соответственно.

Получение/передача буферизированных данных.

H21 BUFRFL` (0038,1) -- Флаг полного буфера.

BUFRFL представляет собой флаг, используемый сервистной программой прерываний при получении последовательной шиной данных для фиксирования момента, когда была получена основная порция фрейма (без байта контрольной суммы). Нулевое значение BUFRFL указывает, что основная порция еще не получена целиком, а ненулевое значение указывает, что основная порция получена.

H22 RECVDN` (0039,1) -- Флаг получения фрейма.

RECVDN представляет собой флаг, используемый SIO для связи между сервисной программой обработки прерываний при получении последовательной шиной данных с основной программой SIO. Первоначально SIO устанавливает нулевое значение флага, а затем сервисная программа обработки прерываний после получения шиной последнего байта фрейма устанавливает ненулевое значение.

H23 TEMP` (023E,1) -- Однобайтовые данные ввода/вывода SIO.

TEMP используется для получения однобайтовых откликов с контроллеров последовательной шины (ACK,NAK,COMPLETE или ERROR).

H24 XMTDON` (003A,1) -- Флаг передачи фрейма.

XMTDON представляет собой флаг, используемый SIO для связи сервисной программы обработки прерываний от передачи через последовательную шину с основной программой SIO. Первоначально SIO устанавливает нулевое значение флага, а затем после передачи последнего байта фрейма через шину сервисная программа обработки прерываний устанавливает ненулевое значение.

Простой SIO.

Для обеспечения возможности простоя различных операций с внутренним инициализированием SIO использует системный таймер 1. Возможности системных таймеров рассматривались в разделе 6. TIMFLG представляет собой флаг, используемый для связи между SIO и программой запуска таймера, на которую указывает CDTMA1.

H25 TIMFLG` (0317,1) -- Флаг простоя операции SIO.

TIMFLG используется для индикации простоя операции шины. Первоначальное значение флага устанавливается равным единице, а при достижении нуля (после простоя) до завершенеия текущей операции, операция прерывается. См. D2.

H26 CDTMV1` (0218,2) -- Значение системного таймера 1.

Данный двухбайтовый счетчик принимает различные значения в зависимости от операции. См. P4.

H27 CDTMA1 (0026,2) -- Адрес системного таймера 1.

Данный вектор всегда указывает на программу JTIMER, единственной функцией которой является определение установлено на TIMFLG нулевое значение или нет. Данный вектор инициализирует SIO перед каждым использованием. По этой причине системный таймер 1 может быть использован любым процессом, не воспроизводящим SIO во время функции простоя. См. P5.

Внутренние рабочие переменные.

H28 STACKP` (0318,1) -- Сохранение/восстановление указателя стека.

STACKP содержит значение регистра SP 6502 при входе в SIO. Данная величина сохраняется для облегчения выхода из подпрограммы SIO при ошибке.

H29 TSTST` (0319,1) -- Временная состояние.

TSAT используется для возвращения состояния из программы WAIT и содержит одно из значений байтов состояния SIO, указанных в приложении B.

H30 ERRFLG` (023F,1) -- Флаг ошибки ввода/вывода.

ERRFLG используется для связи программы WAIT и программой SIO более высокого уровня. Обычно значение ERRFLG равно нулю, но при получении неправильного байта отклика от устройства устанавливает значение $FF.

H31 STATUS` (0031,1) -- Состояние операции SIO.

STATUS представляет собой переменную нулевой страницы, используемую внутри SIO для хранения состояния операции, помещаемого затем в переменную-параметр последовательности вызова DSTATS при возвращении SIO в точку вызова.

H32 SSKCTL` (0232,1) -- Копия SKCTL.

SSKCTL используется SIO для хранения следа содержимого регистра SKCTL (D20F), который является регистром, предназначенным только для записи.

J. Контроллеры ATARI.

Считывание с контроллеров ATARI происходит на второй стадии VBLANK. Закодированные данные частично декодируются и обрабатываются, как показано в следующих подразделах.

Джойстики.

К консоли компьютера можно подключить до двух джойстиков, каждый из которых имеет 9 позиций джойстика и кнопку-триггер.

J1 STICK0-STICK1 (0278,2) -- Считывание положения джойстика.

2 переменные считывания позиции джойстика содержат закодированный по битам результат чтения.

```
 7 6 5 4 3 2 1 0
+-+-+-+-+-+-+-+-+
!0!0!0!0!R!L!D!U!
+-+-+-+-+-+-+-+-+
```

Где, R=0 указывает на истинность положения "право" (RIGHT).
L=0 указывает на истинность положения "лево" (LEFT).
D=0 указывает на истинность положения "низ" (DOWN).
U=0 указывает на истинность положения "верх" (UP).

Возможны 9 различных комбинаций соответствующих различным положениям джойстика.

центр	0F
верх	0E
верх/право	06
право	07
низ/право	05
низ	0D
низ/лево	09
лево	0B
верх/лево	0A

J2 STRIG0-STRIG1 (0284,2) -- Считывание триггера джойстика.

Каждая из двух переменных считывания с триггера джойстика содержит единственный бит, указывающий положение триггера джойстика, как показано ниже.

```
 7 6 5 4 3 2 1 0
+-+-+-+-+-+-+-+-+
!0!0!0!0!0!0!0!T!
+-+-+-+-+-+-+-+-+
```

Где T=0 указывает, что триггер нажат.

Контроллеры PADDLE.

К компьютеру может быть подключено до 4 контроллеров PADDLE, каждый из которых снабжен потенциометром и триггером.

J3 PADDL0-PADDL3 (0270,4) -- Считывание положения PADDLE.

Положения PADDLE связана одна однобайтовая переменная. Значение изменяются от 228 для полного оборота против часовой стрелки до 1 для полного оборота по часовой стрелке. Значение PADDLE часто преобразуется пользователем таким образом, как это показано ниже для получения нуля для полного оборота против часовой стрелки и 227 для полного оборота по часовой стрелке.

```
VALUE:=228-PADDLX;
```

J4 PTRIG0-PTRIG3 (027C,4) -- Считывание с триггера PADDLE.

Каждая из 4 переменных считывания с триггера PADDLE содержит одиночный байт, указывающий положение триггера PADDLE, как это показано ниже.

```
 7 6 5 4 3 2 1 0
+-+-+-+-+-+-+-+-+
!0!0!0!0!0!0!0!T!
+-+-+-+-+-+-+-+-+
```

Где, T=0 соответствует нажатому триггеру.

Световое перо.

ОС производит считывание положений светового пера и сохраняет коды горизонтальной и вертикальной координат в двух переменных. Данные коды не совпадают полностью с координатами на экране. Ниже приведены коды для различных частей экрана:

Левый край - 67.
Коды последовательно возрастают на единицу, затем доходят до нуля и продолжают монотонно возрастать (на единицу для каждого цветового импульса).
Правый край - 7.
Верхняя часть - 16.
Коды последовательно возрастают на единицу (после двух растровых строк).
Нижняя часть - 111.
Аппаратные средства светового пера производят считывание и блокировку позиций пера 60 раз в секунду, независимо от положения кнопки пера, считывание с которой производится независимо. Для работы со световым пером оно должно располалаться над той частью экрана, которая имеет соответствующую яркость, чтобы активизировать фоточувствительный материал в пере. Пустой (темный) экран не имеет достаточной яркости для работы со световым пером.

J5 LPENH (0234,1) -- Код горизонтальной позиции светового пера.

LPENH содержит код горизонтальной позиции светового пера. Ниже приведен алгоритм (выполненный на Паскале) для перевода кода в экранные координаты (седьмого экранного режима):

```
IF LPEN(33 ` проверить выход за границы `
   THEN ` скорректировать значение по правой границе `
      XPOS:=LPENH+227
   ELSE ` не производить корректировку слева от точки `
      XPOS:=LPENH;
XPOS:=XPOS-67; ` отрегулировать по сдвигу левого края `
IF XPOS(0 THEN XPOS=0;
IF XPOS)159 THEN XPOS:=159;
```

J6 LPRNV (0235,1) -- Код вертикальной позиции светового пера.

LPENV содержит код вертикальной позиции светового пера. Ниже приведен алгоритм (выполненный на Паскале) перевода кода позиции в координаты экрана (седьмого экранного режима).

```
YPOS:=LPENV-16; ` корректировка под сдвиг левого края `
IF YPOS(0 THEN YPOS=0;
IF YPOS)95 THEN YPOS:=95;
```

J7 STICK0-STICK1 (0278,2) -- Считывание с кнопки светового пера.

Положение кнопки светового пера кодируется в одной из переменных STICK0-STICK1 (в зависимости от того какой из портов задействован) следующим образом:

```
 7 6 5 4 3 2 1 0
+-+-+-+-+-+-+-+-+
!       !0!0!0!T!
+-+-+-+-+-+-+-+-+
```

Где, T соответствует нажатой кнопки.

Управляющие контроллеры.

Управляющие контроллеры не имеют конечных положений и поэтому допускают неограниченной вращение в любом направлении. Выходом контроллера служит двухбитовый код Грея, который может быть использован для определения направления вращения. Считывание с контроллера осуществляется с помощью тех же самых аппаратных средств, что и с джойстика. Поэтому для них используются одни и те же переменные базы данных.

J8 STICK0-STICK1 (0278,2) -- Считывание с управляющих контроллеров.

В 2 переменных считывания с управляющих контроллеров закодировано положение контроллера следующим образом:

```
 7 6 5 4 3 2 1 0
+-+-+-+-+-+-+-+-+
!0!0!0!0!1!1!зн.!
+-+-+-+-+-+-+-+-+
```

Где, вращение контроллера по часовой стрелке вызывают следующую последовательность значений (в шестнадцатеричной форме):
0F,0D,0C,0E,0F,0D,...
А вращение контроллера против часовой стрелки вызывает следующую последовательность значений:
0F,0E,0C,0D,0F,0E,...

J9 STRIG0-STRIG1 (0284,2) -- Считывание с управляющих триггеров.

Каждая из двух переменных считывания с триггеров содержит единичный бит, указывающий положение управляющего триггера, как это показано ниже.

```
 7 6 5 4 3 2 1 0
+-+-+-+-+-+-+-+-+
!0!0!0!0!0!0!0!T!
+-+-+-+-+-+-+-+-+
```

Где, T=0 соответствует нажатому триггеру.

К. Обработчик дисковых файлов.

Информацию, относящуюся к обработчику дисковых файлов можно найти в разделе 5.

K1 FMSZPG` (0043,7) -- Область, зарезервированная обработчиком дисковых файлов.

FMSZPG представляет собой область, зарезервированную в базе данных для ниже приведенных переменных. Имена переменных отсутствуют в файле присваиваний системы.

K2 ZBUFP` (0043,2) -- Указатель буфера.

K3 ZDRVA` (0045,2) -- Управляемый указатель.

K4 ZSBA` (0047,2) -- Указатель на буфер сектора.

K5 ERRNO` (0049,1) -- Номер ошибки.

L. Указатель программы диска.

L1 DSKUTL` (001A,2) -- Переменная указателя на нулевую страницу.

M. Пакет математики с плавающей запятой.

M1 FR0 (00D4,6) -- Регистр 0 пакета математики с плавающей запятой.

M2 FRE` (00DA,6) -- Внутренний регистр пакета математики с плавающей запятой.

M3 FR1` (00E0,6) -- Регистр 1 пакета математики с плавающей запятой.

M4 FR2` (00E6,6) -- Внутренний регистр 2 пакета математики с плавающей запятой.

M5 FRX` (00EC,1) -- Не используется.

M6 EEXP` (00ED,1) -- Знак степени (внутренний).

M7 NSIGN` (00EE,1) -- Знак мантисы (внутренний).

M8 ESIGN` (00EF,1) -- Знак экспоненты (внутренний).

M9 FCHRFLG` (00F0,1) -- Флаг первого символа (внутренний).

M10 DIRT` (00F1,1) -- Цифры справа от десятичной точки.

M11 CIX (00F1,1) -- Индекс символа.

M12 INBUFF (00F3,2) -- Указатель на буфер ввода текста.

M13 ZTEMP1` (00F5,2) -- Временное хранение.

M14 ZTEMP4` (00F7,2) -- Временное хранение.

M15 ZTEMP3` (00F9,2) -- Временное хранение.

M16 FLPTR (00FC,2) -- Указатель на число с плавающей запятой.

M17 FPTR2` (00FE,2) -- Использование пакета математики с плавающей запятой.

M18 LBPR1` (057E,1) -- Преамбула LBUFF.

M19 LBPR2` (057F,1) -- Преамбула LBUFF.

M20 LBUFF (0580,96) -- Текстовой буфер.

M21 PLYARG` (05E0,6) -- Регистр пакета математики с плавающей запятой (внутренний).

M22 FPSCR/FSCR` (05E6,6) -- Регистр пакета математики с плавающей запятой (внутренний).

M23 FPSCR1/SCR1` (05EC,6) -- Регистр пакета математики с плавающей запятой (внутренний).

M24 DEGFLG/RADFLG (00FB,1) -- Флаг градусы/радианы.

DEGFLG=0 соответствует радианам, 6 соответствует градусам.

N. Включение питания и перезагрузка системы.

Детальное рассмотрение процессов при включении питания и перезагрузке системы можно найти в разделе 7.

Размер ОЗУ.

В процессе включения питания и перезагрузки системы первый адрес памяти, отличный от ОЗУ выше !1000, размещается и сохраняется при помощи безвредного теста. Первый байт каждого блока из 4К тестируется на предмет того, возможно ли его изменение. Если возможно, восстанавливается первоначальное значение и тестируется следующий блок. Если нет, этот адрес считается концом ОЗУ.

N1 RAMLO`/TRAMSZ` (0004,3) -- Тестовый указатель на данные ОЗУ (временный).

RAMLO+1 содержит младший байт проверяемого адреса (всегда = 0), а TRAMSZ (то же, что RAMLO+2) содержит старший байт проверяемого адреса. RAMLO+0 содержит дополнительное значение данных, первоначально хранившееся в проверяемой ячейке памяти. Позже в процессе инициализации данные переменные используются для полностью несвязанных функций, но вначале значение из TRAMSZ перемещается в переменные RAMSIZ и MEMTOP+1.

N2 TSTDAT` (0007,1) -- Сохранение тестируемого байта данных.

TSTDAT содержит первоначальное значение тестируемой ячейки памяти.

Загрузка с дискеты и кассеты.

Как указывалось в разделе 10, при включении питания возможна загрузка с подключенного дисковода или кассетного магнитофона.

N3 DSINI (000C,2) -- Вектор инициализации загрузки с дискеты.

DOSINI содержит адрес инициализации загруженной с дискеты программы, находящейся в начале файла загрузки в том случае, когда загрузка с дискеты успешно завершилась.

N4 CKEY` (004A,1) -- Флаг запроса на загрузку с кассеты.

CKEY представляет собой внутренний флаг, используемый для индикации нажатия клавиши консоли (START) во время включения питания, указывающий на желательность загрузки с кассеты. При отсутствии запроса на загрузку с кассеты CKEY равно нулю, и CKEY отлично от нуля при наличии запроса на загрузку с кассеты. После загрузки с кассеты флаг снова принимает нулевое значение.

N5 CASSBT` (004B) -- Флаг загрузки с кассеты.

CASSBT используется в процессе загрузки с кассеты и служит для указания разделяющей программе, что имеет место загрузка скассеты, а не с дискеты. Нулевое значение CASSBT соответствует загрузке с дискеты, а ненулевое - загрузке с кассеты.

N6 CASINI (0002,2) -- Вектор инициализации загрузки с кассеты.

CASINI содержит адрес инициализации программы, загруженной с кассеты из начала файла загрузки (см. Раздел 10) при успешном завершении загрузки с кассеты.

N7 BOOT?` (0009,1) -- Флаг успешной загрузки с дискеты/кассеты.

BOOT? указывает обработчику инициализации какая из операций загрузки, в случае ее успешного завершения имела место:

```
 7 6 5 4 3 2 1 0
+-+-+-+-+-+-+-+-+
! ! ! ! ! ! !C!D!
+-+-+-+-+-+-+-+-+
```

Где, C=1 указывает, что завершилась загрузка с кассеты.
D=1 указывает, что завершилась загрузка с дискеты.

N8 DFLAGS` (0240,1) -- Флаги дискеты.

DFLAGS содержит значение первого байта файла загрузки, после загрузки с дискеты. См. Раздел 10.

N9 DBSECT` (0241,1) -- Счетчик секторов при загрузке с дискеты.

DBSECT первоначально во время проведения загрузки устанавливается равным значению второго байта файла загрузки, а затем используется для фиксирования номеров секторов дискеты, считываемых дополнительно.

N10 BOOTAD` (0242,2) -- Адрес памяти загрузки с дискеты.

Первоначально в процессе загрузки с дискеты значение BOOTAD устанавливается равным третьему и четвертому байтам файла загрузки, и впоследствии не изменяется.

Управление внешней средой.

Если к концу включения питания или при перезагрузке системы управление не было передано одному из картриджей (как объяснялось в разделах 7 и 10), то управление передается по адресу, хранящемуся в переменной базы данных DOSVEC.

N11 COLDST` (0244,1) -- Флаг завершения холодного старта.

COLDST используется программой инициализации для обнаружения случая, когда перезагрузка системы прошла до завершения процессов, происходящих при включении питания. В начале холодного старта значение COLDST устанавливается равным $FF, и 0 - при завершении процессов. В случае перезагрузки системы в момент, когда значение отлично от нуля, последовательность действий холодного старта, а не процедура перезагрузки, будет запущена заново.

N12 DOSVEC (000A,2) -- Вектор управления без картриджа.

В начале горячего старта ОС устанавливает DOSVEC для указания на программу самотестирования. Затем, вследствие загрузки с кассеты или дискеты (как объяснялось в разделе 10) для передачи управления новой программе DOSVEC может быть изменен. При всех условиях включения питания и перезагрузке системы, при которых картридж не получает первым управление, оно будет передано через DOSVEC.

Перезагрузка системы.

N13 WARMST (0008,1) -- Флаг горячего старта.

При перезагрузке системы (горячий старт) WARMST устанавливается равным $FF, а при включении питания (холодный старт) равным нулю.

P. Прерывания.

Обработка прерываний описана в разделе 6.

P1 CRITIC (0042,1) -- Флаг раздела с критическим кодом.

CRITIC используется для сообщения обработчику прерывания VBLANK, что раздел с критическим кодом выполняется, не препятствуя прерываниям IRQ. При установке CRITIC обработчик VBLANK приостановит обработку прерываний после стадии 1 до стадии 2 как, если бы был установлен бит I процессора 6502. Нулевое значение CRITIC указывает, что раздел с кодом не является критическим, а ненулевое значение флага говорит о том, что обрабатываемый раздел с кодом - критический.

P2 POKMSK (0010,1) -- Маска прерываний POKEY.

POKMSK представляет собой програмно-поддерживаемую маску прерываний, используемую совместно с разрешением и запрещением различных прерываний POKEY. Наличие этой маски обязательно, поскольку регистр IRGEN (020E), разрешающий прерывания, является регистром, предназначенным только для записи. В любой момент времени у системы может быть несколько пользователей, разрешающих или запрещающих прерывания POKEY независимо друг от друга. POKMSK постоянно корректируется пользователем таким образом, чтобы содержать текущее значение IRGEN.

Системные таймеры.

Подробное описание таймеров можно найти в разделе 6.

Часы реального времени (или счетчик кадров, как его иногда называют) увеличивает свое значение на единицу во время первой стадии процесса VBLANK, как это показано в разделе 6.

P3 RTCLOK (0012,3) -- Счетчик кадров реального времени.

RTCLOK+0 - старший байт, RTCLOK+1 - следующий по старшинству байт, а RTCLOK+2 - младший байт. (См. D3 и предыдущее B10 описание RTCLOK.

Системный таймер 1.

Системный таймер 1 поддерживается на 1 стадии процесса VBLANK и поэтому имеет наивысший приоритет среди всех таймеров.

P4 CDTMV1 (0218,2) -- Значение системного таймера 1.

CDTMV1 содержит нулевое значение, когда таймер не активен. В противном случае его значение указывает число, оставшееся до простоя VBLANK. См. H26.

P5 CDTMA1 (0226,2) -- Адрес перехода системного таймера 1.

CDTMA1 содержит адрес, по которому должен быть осуществлен переход JSR при простое таймера. См. H27 и Раздел 6.

Системный таймер 2.

Системный таймер 2 поддерживается на второй стадии процесса VBLANK и имеет второй по счету приоритет среди таймеров. ОС никак непосредственно не использует системный таймер 2.

P6 CDTMV2 (021A,2) -- Значение системного таймера 2.

CDTMV2 содержит нулевое значение, когда таймер не активен. В противном случае в CDTMV2 хранится число процессов VBLANK, оставшихся до простоя.

P7 CDTMA2 (0028,2) -- Адрес перехода системного таймера 2.

CDTMA2 содержит адрес, по которому производится переход JSR в случае простоя таймера. См. Раздел 6.

Системные таймеры 3, 4 и 5.

Системные таймеры 3, 4 и 5 поддерживаются на второй стадии процесса VBLANK и имеют самый низший приоритет среди таймеров пользователя. ОС не имеет специального назначения для использования этих таймеров.

P8 CDTMV3 (021C,2), CDTMV4 (021E,2) и CDTMV5 (0220,2).

Когда соответствующие таймеры неактивны, данные переменные содержат нулевые значения. В противном случае они содержат число VBLANK, оставшееся до простоя.

P9 CDTMF3 (0022A,1), CDTMF4 (002C,1) и CDTMF5 (002E,1).

Каждая из этих обнобайтовых переменных устанавливается равной нулю при простое соответствующего таймера. ОС никогда не изменяет значения этих байт за исключением случаев простоя (и инициализации).

Векторы прерываний ОЗУ.

Для различных условий прерываний внутри системы существуют различные векторы ОЗУ. Расположение значений этих векторов можно найти в разделе 6.

Вектора прерываний NMI (немаскируемых прерываний).

P10 VDSLST (0200,2) -- Вектор прерываний от дисплейной программы.

Данный вектор не используется ОС. См. Раздел 6.

P11 VVBLKI (0222,2) -- Вектор мгновенного VBLANK.

Данный вектор инициализируется таким образом, чтобы указывать на стадию 1 VBLANK ОС.

P12 VVBLKD (0224,2) -- Вектор отложенного VBLANK.

Данный вектор инициализируется для указания на программу выхода из VBLANK ОС. См. Раздел 6.

Вектора прерываний по запросам (IRQ).

P13 VIMIRQ (0216,2) -- Вектор общих IRQ.

Данный вектор инициализируется для указания на обработку IRQ ОС. См. Раздел 6.

P14 VPRCED (0202,2) -- Действующий сигнал последовательной шины ввода/вывода.

Линия последовательной шины ввода/вывода, вызывающая соответствующее прерывание в данной системе, отсутствует. См. Раздел 6.

P15 VINTER (0204,2) -- Прерывающий сигнал последовательной шины ввода/вывода.

Линия последовательной шины, вызывающая это прерывание, не используется в текущей системе. См. Раздел 6.

P16 V(BREAK) (0206,2) -- Вектор команды BRK.

Данный вектор инициализируется для указания на последовательность PLA, RTI в случае, если ОС не выполнит команду BRK. См. Раздел 6.

P17 VKEYBD (0208,2) -- Вектор прерываний клавиатуры.

Данный вектор инициализируется для указания на сервисную программу обработки прерываний клавиатуры. См. Раздел 6 и коментарии к E1.

P18 VSERIN (020A,2) -- Готовность последовательной шины к получению данных.

Данный вектор инициализируется для указания на сервисную программу обработки прерываний программы SIO. См. Раздел 6.

P19 VSEROR (020C,2) -- Готовность последовательной шины к передаче данных.

Данный вектор инициализируется для указания на сервисную программу обработки прерываний программы SIO. См. Раздел 6.

P20 VSEROC (020E,2) -- Завершение передачи данных через последовательную шину ввода/вывода.

Данный вектор инициализируется для указания на сервисную программу обработки прерываний программы SIO. См. Раздел 6.

P21 VTIMR1 (0210,2), VTIMR2 (0212,2) и VTIMR4 (0224,2) -- Вектора таймеров POKEY.

ОС не использует прерывания таймеров POKEY. См. Раздел 6.

Корректировка аппаратных регистров.

На второй стадии VBLANK, как это описано в разделе 6, значения определенных аппаратных регистров корректируются из переменных базы данных ОС.

P22 SDMCTL` (022F,1) -- Управление прямым доступом к памяти.

Значение SDMCTL устанавливается равным !02 в начале операции OPEN дисплея. Затем оно устанавливается равным !22. Значение SDMCTL на второй стадии процесса VBLANK сохраняется в DMACTL (D400).

P23 SDLSTL` (0230,1) и SDLSTH` (0231,1) -- Адрес дисплейной программы.

После каждой операции OPEN дисплей форматирует новую дисплейную программу и помещает ее адрес в SDLSTL и SDLSTH. Значение этих байт сохраняются в DLISTL (D4002) и DLISTH (D403) на второй стадии процесса VBLANK.

0360-036F	IOCB {2
0370-037F	IOCB {3
0380-038F	IOCB {4
0390-039F	IOCB {5
03A0-03AF	IOCB {6
03B0-03BF	IOCB {7

Примечание:
При корректировке аппаратных регистров из переменных базы данных могут возникнуть проблемы с синхронизацией во времени. Поскольку вторая стадия VBLANK выполняется при разрешенных прерываниях, возможно возникновение прерываний IRQ до корректировки DLISTH и DLISTL. Если время обработки этих прерываний (и остальных гнездовых прерываний) превосходит задержку вертикального пропуска (1 мсек), то регистр указателя на дисплейную программу не будет скорректирован к моменту, когда начнется обработка дисплейной программы.

P24 GPRIOR` (026F,1) -- Управление приоритетом.

В процессе установки режима дисплей изменяет биты GTIA 6 и 7 переменной GPRIOR. Значение GPRIOR сохраняется в PRIOR (D10B) в процессе второй стадии VBLANK.

P25 CHACT` (02F3,1) -- Управление символами.

После каждой команды OPEN дисплей устанавливает CHACT равным !02. Значение CHART помещается в CHACTL на второй стадии процесса VBLANK.

P26 CHBAS (02F4,1) -- Базовый адрес символа.

После каждой команды OPEN дисплей устанавливает значение CHBAS равным !E0. На второй стадии обработки процесса VBLANK значение CHBAS сохраняется в CHBASE (D409). Данная переменная управляет подмножеством символов для 1 и 2 экранных режимов. Значение !E0 соответствует множеству заглавных символов и цифр. тогда как значение !E2 соответствует множеству символов нижнего регистра и специальным графическим символам. Дополнительную информацию см. в B55.

P27 PCOLR` (02C0,4) и COLOR` (02C4,5) -- Цветовые регистры.

См. B7 и B8.

Внутренние рабочие переменные.

P28 INTEMP` (022D,1) -- Временное хранение.

INTEMP используется программой SETVBL (SETVBV).

R. Области пользователя.

Указанные ниже области доступны внешней невложенной операционной среде пользователя. Дополнительную информацию можно получить в разделе 4.

R1 (0080,128).

R2 (0480,640).