**MP/M II™**

**Операционная система**

**СИСТЕМНОЕ РУКОВОДСТВО КОНСТРУКТОРА**

Авторское право © 1981

Digital Research

P.O. Box 579

801 Lighthouse Avenue

Pacific Grove, CA 93950

(408) 649-3896

TWX 910 360 5001

Все права защищены

**Авторское право**

Авторское право 1981 Digital Research. Все права защищены. Любая часть этой публикации не может быть воспроизведена, передана, записана, сохранена в поисковой системе или переведена на любой язык или компьютерный язык, в любой форме или каким-либо образом, электронным, механическим, магнитным, оптическим, химическим, ручным или иначе, без предварительного письменного разрешения Digital Research, Post Office Box 579, Pacific Grove, California, 93950.

**Исключение ответственности**

Digital Research не дает никаких заверений или гарантий в отношении содержания данного документа, включая любые гарантии товарности или пригодности для какой-либо конкретной цели. Кроме того, Digital Research сохраняет за собой право пересмотреть эту публикацию и время от времени вносить изменения в содержание настоящего документа без обязательства Digital Research уведомить любое лицо, о таком пересмотре или изменениях.

**Торговые марки**

CP/M - зарегистрированная торговая марка Digital Research. CP/NET, MP/M II, LINK-80, RMAC и PL/I-80 являются торговыми марками Digital Research. Z80 - зарегистрированная торговая марка Zilog, Inc.

"Руководство системы MP/M II" было подготовлено, используя средство форматирования текста TEX-80 Digital Research и распечатано в Соединенных Штатах Америки Commercial Press / Monterey.[[1]](#footnote-1)

Первое издание: сентябрь 1981

Второе издание: август 1982

**Предисловие**

MP/M II™ является многопользовательской операционной системой для любого микрокомпьютера на основе 8-разрядного микропроцессора Zilog Z80® или Intel 8080 или 8085. Как правило, система MP/M II занимает приблизительно 27k. 16k операционной системы должно находиться в общей памяти.

Версия MP/M II, поставляемая Digital Research не может быть непосредственно загружена на какую-либо конкретную аппаратную конфигурацию. Однако весь аппаратно-зависимый код выделен в отдельные подпрограммы, которые могут быть изменены пользователем.

Этот документ описывает процедуры, необходимые для реализации MP/M II для оборудования пользователя. Как минимум, обычное оборудование должно включать в себя процессор 8080, 8085, или Z-80, 32K байт оперативной памяти (ОЗУ), системную консоль и часы реального времени. Это руководство предполагает, что читатель знаком со следующими публикациями Digital Research:

* Руководство пользователя MP/M II;
* Руководство программиста MP/M II.

Также предполагается, что читатель уже реализовал базовую систему ввода-вывода (BIOS) CP/M 2, предпочтительно для компьютера на котором предполагается установка MP/M II.

**Содержание**

Раздел 1 Процедура изменения MP/M II 6

1.1 Подготовка MP/M II к внесению изменений 6

1.2 Настройка MPMLDR 6

1.3 Настройка XIOS 7

1.4 Отладка XIOS 8

1.5 Непосредственно загрузка MP/M II 10

1.5.1 Подготовка загрузки MP/M II используя SYSGEN 11

1.5.2 Настройка генерации загрузчика MP/M II 12

1.5.3 Примеры программ GETSYS и PUTSYS 13

1.6 Загрузка MPM.SYS без MPMLDR 14

1.7 Авторское право и торговая марка Digital Research 15

1.8 Организация диска 15

Раздел 2 MP/М II BIOS 17

2.1 Обзор BIOS MP/M II 17

2.2 Характеристики устройств BIOS и точки входа 18

2.3 Таблицы определения дисков BIOS 24

2.3.1 Формат таблицы параметров дисков 24

2.3.2 Библиотека макросов DISKDEF 28

2.4 Процедуры внешнего доступа 31

2.5 Алгоритм блокирование и разблокирования 32

2.6 Часть общей памяти BNKXIOS 33

Раздел 3 XIOS MP/M II 34

3.1 Обзор XIOS MP/М II 34

3.2 Точки входа MP/M XIOS 34

3.3 Процедуры обработки прерывания 37

3.4 Управление отсчётом времени 38

Раздел 4 Компоненты системного файла MP/M II 39

4.1 Системные данные 39

4.2 Настроенный XIOS 40

4.3 BDOS 40

4.3.1 RESBDOS 40

4.3.2 BNKBDOS 40

4.4 XDOS 40

4.5 Резидентные системные процессы 41

4.6 Переключаемые резидентные системные процессы 41

Раздел 5 Генерация системы 42

5.1 Работа GENSYS 42

5.2 Параметры генерации системы 43

5.2.1 По умолчанию 43

5.2.2 Вершина страницы операционной системы 44

5.2.3 Количество консолей системы 44

5.2.4 Количество принтеров 44

5.2.5 Номер рестарта точки останова 44

5.2.6 Системный вызов пользовательского стека 44

5.2.7 Процессор Z80 44

5.2.8 Число тактов в секунду 45

5.2.9 Системный диск 45

5.2.10 Диск для временных файлов 45

5.2.11 Максимальное число заблокированных записей/процессов 45

5.2.12 Общее число заблокированных записей системы 45

5.2.13 Максимальное количество открытых файлов/процессов 45

5.2.14 Общее количество открытых файлов в системе 45

5.2.15 Память с коммутацией банков 46

5.2.16 Число сегментов пользовательской памяти 46

5.2.17 Начальная страница общей памяти 46

5.2.18 Вывод в консоль информации о выполнении файла 46

5.2.19 Принять записи системных данных 46

5.2.20 Выбор резидентных системных процессов 46

5.2.21 Таблица сегментов памяти 46

5.2.22 Принять таблицу сегментов памяти 47

5.3 Выполнение GENSYS 47

Раздел 6 Загрузчик MP/M 48

6.1 Работа и вывод загрузчика MP/M 48

6.2 Выполнение MPMLDR 49

Приложение 1 Макросы определения дисков 50

Приложение 2 Алгоритмы разблокирования сектора для MP/M II 55

Приложение 3 Пример загрузчика MP/M II BIOS 63

Приложение 4 Листинг источника простого XIOS 65

Приложение 5 Листинг источника XIOS с коммутацией банков 85

**Список таблиц**

Таблица 1–1 Формат файла MPM.SYS 15

Таблица 1–2 Пример организации диска MP/M II 16

Таблица 2–1 Сводка подпрограмм BIOS 18

Таблица 2–2 Элементы заголовка параметров диска 24

Таблица 2–3 Поля блока параметров диска (DPB) 26

Таблица 2–4 Значения BSH и BLM для выбранного BLS 26

Таблица 2–5 Максимальные значения EXM 26

Таблица 2–6 BLS и число записей каталога 27

Таблица 2–7 Краткая информация о внешних процедурах 32

Таблица 3–1 Сводка подпрограмм XIOS 34

Таблица 4–1 Назначения байтов системной страницы данных 39

# Процедура изменения MP/M II

Операционная система MP/M II разработана так, что пользователь может изменять определенный набор подпрограмм, определяющих условия эксплуатации оборудования. Изменяя эти подпрограммы, пользователь может производить дискету, которая работает с любой подсистемой гибких дисков в формате совместимом с дискетой IBM 3740 и другими периферийными устройствами.

Хотя стандартная MP/M II поставляется на гибких дисках одинарной плотности, функции изменения полей позволяют пользователю адаптировать MP/M II к большому разнообразию дисковых подсистем, включая драйвер мини диска (5.25") и систем большой емкости с "жесткими дисками".

Для достижения независимости от устройств, MP/M II изолирует весь аппаратно-зависимый код в модуле XIOS. Пользователь может переписать дистрибутивную версию XIOS, чтобы настроить интерфейс с остальными модулями MP/M II и оборудованием пользователя. Также, пользователь может переписать дистрибутивную версию программы LDRBIOS, которая загружает систему MP/M II с диска.

На самом деле существует две версии XIOS: RESXIOS для систем, без коммутации банков и BNKXIOS для систем с переключением банков памяти. Чтобы избежать повторения этих двух названий в каждой ссылке, термин XIOS относится к обеим версиям.

## Подготовка MP/M II к внесению изменений

Для упрощения процесса изменений, этот документ предполагает, что BIOS CP/M 2 уже реализован на целевой машине MP/M II. Необходимо реализовать как BIOS, так и XIOS потому, что загрузчик MP/M II использует BIOS CP/M 2 для загрузки системы MP/M II. После загрузки MP/M II использует XIOS, а не BIOS. Загрузчик MP/M II называемый LDRBIOS использует BIOS CP/M 2.

Еще одна причина для реализации CP/M 2 на целевой машине MP/M состоит в том, что отладка вашего XIOS проще, если вы можете выполнить SID или DDT в системе CP/M 2.

## Настройка MPMLDR

Чтобы настроить MPMLDR, вы должны интегрировать LDRBIOS для своей аппаратной конфигурации в файл MPMLDR.COM, предоставленный на дистрибутивном диске. Требуемый LDRBIOS может быть просто версией вашего CP/M 2 BIOS, измененный, как описано ниже и переименованный в LDRBIOS.

Настроенный LDRBIOS должен иметь ORG 1700H, выполнять консольные функции вывода и быть в состоянии считать данные из единственного дисковода. Первый вызов, который MPMLDR выполняет к LDRBIOS, является SELDSK: выбор диска. Если ваша система имеет устройства, которые требуют инициализации, поместите код инициализации или, возможно, вызов к LDRBIOS холодный старт в начале обработчика SELDSK.

LDRBIOS должен выполнять только операции, описанные выше. Другие функции могут быть удалены для экономии места. Существует только одно ограничение на объем памяти для LDRBIOS: он не может простираться выше начала MPM.SYS, который он загружает. (GENSYS перечисляет начальный адрес MP/M II в своей карте загрузки). Однако, если вы планируете загружать MP/M II с дискеты, вы столкнетесь с ограничением верхнего адреса LDRBIOS 1A00H для того, чтобы разместить файл MPMLDR.COM на двух системных дорожках.

Протестируйте LDRBIOS полностью, чтобы убедиться, что он правильно осуществляет консольный вывод символов и чтение с диска. Будьте особенно внимательны, что бы случайно не происходило никаких операций записи на диск во время операций чтения и проверьте, что правильно адресуются дорожки и сектора при всех операциях чтения.

Используйте следующие шаги, чтобы интегрировать настроенный LDRBIOS в MPMLDR.COM:

1. Получите доступ к системе CP/M и подготовьте файл LDRBIOS.HEX.
2. Прочитайте файл MPMLDR.COM в память, используя DDT или SID.

A>**DDT MPMLDR.COM**

DDT VERS 2.0

NEXT PC 1780

0100

1. С помощью ввода команды (I), укажите, что файл LDRBIOS.HEX должен быть считан и затем выполните чтение (R) файла. Эта операция выполнит наложение части LDRBIOS загрузчика MP/M.

-**ILDRBIOS.HEX**

-**R**

NEXT PC

1A00 0000

1. Выйдите из отладчика, возвратившись к CCP, выполнив переход к нулевому адресу.

-**G0**

1. Запишите обновленное отображение памяти в файл на диске. Используйте команду SAVE CP/M для записи обновленного образа памяти в файл на диске. В примере ниже, X перед именем файла просто определяет экспериментальную версию и сохраняет оригинал.

A>**SAVE 26 XMPMLDR.COM**

1. Протестируйте XMPMLDR.COM и затем переименуйте его в MPMLDR.COM.

## Настройка XIOS

Поскольку вы адаптируете MP/M II для вашей компьютерной системы, ваш новый XIOS потребует разработки и тестирования программного обеспечения. Два примера XIOS приведены в приложениях и могут использоваться в качестве моделей для настраиваемого пакета.

Точки входа XIOS, включая основные и расширенные, описаны в [Разделах 2](#_MP/М_II_BIOS) и [3](#_MP/M_II_XIOS). Эти разделы, вместе с приложениями, предоставляют вам необходимую информацию для написания своего XIOS. Ваша первоначальная реализация XIOS должна использовать ввод-вывод по опросу без использования любых прерываний. Эта начальная система может работать без прерываний от часов. Реализуйте прерывания только после того, как ваш XIOS будет полностью разработан и протестирован.

Выполните приведенную ниже процедуру для подготовки файла BNKXIOS.SPR или RESXIOS.SPR из настроенного вами XIOS:

1. Скомпилируйте свой BNKXIOS.ASM или RESXIOS.ASM с помощью RMAC или любого другого ассемблера, который может генерировать файл с расширением REL в формате перемещаемых объектных модулей Microsoft.

A>**RMAC BNKXIOS**

1. Соедините файл BNKXIOS.REL или RESXIOS.REL, используя редактор связей LINK-80 Digital Research, чтобы произвести файл BNKXIOS.SPR или RESXIOS.SPR.

A>**LINK BNKXIOS [OS]**

## Отладка XIOS

Вы можете отладить XIOS или резидентный системный процесс с помощью DDT или SID, работающего в CP/M. В общих чертах метод отладки описан в следующих шагах:

1. Определите объем памяти, доступный MP/M II, когда отладчик и CP/M будут резидентным объектом. Сделайте это, загрузив отладчик и затем выведите команду перехода по адресу 0005H. Этот переход к начальному адресу отладчика.

A>**DDT**

DDT VERS 2.0

-**L5**

0005 JMP C800

1. Используя GENSYS под управлением CP/M сгенерируйте файл MPM.SYS, который указывает на вершину памяти определенную на предыдущем шаге, предусмотрев по крайней мере 256 байт для области патча.

...

Top page of operating system (xx) ? C6

...

Также при выполнении GENSYS, укажите номер рестарта точки останова отличающийся от используемого отладчиком CP/M, который вы планируете использовать. Предлагаемый рестарт MP/M II является #6. Однако, обычно может быть использован любой рестарт от #1 до #6. Отладчики CP/M, как правило, используют рестарт #7.

...

Breakpoint RST (xx) ? 6

...

*Примечание*: Если вы также отлаживаете резидентный системный процесс, убедитесь, что выбрали его для включения в MPM.SYS во время выполнения GENSYS.

1. Используя CP/M, загрузите файл MPMLDR.COM в память.

A>**DDT MPMLDR.COM**

DDT VERS 2.0

NEXT PC

1A00 0100

1. Поместите символы “$B” в местах 005DH и 005EH в FCB по умолчанию расположенный начиная с адреса 005CH. Эта операция может быть выполнена с помощью команды I:

-**I$B**

“$B” заставляет MPMLDR прерваться после загрузки файла MPM.SYS. Вы можете задать рестарт точки останова, который будет выполнен MPMLDR, добавив один дополнительный символ к строке, в четвертую позицию FCB по умолчанию.

-**I$B6**

В примере выше, рестарт #6 должен быть выполнен MPMLDR, после завершения загрузки файла MPM.SYS. Если номер рестарта не задан, рестарт по умолчанию - #7. Помните, номер рестарта по адресу 5FH является номером рестарта отладчика CP/M, а не рестартом отладчика MP/M.

1. Выполните программу MPMLDR.COM, введя команду G:

-**G**

1. После команды G, загрузчик MP/M II выполнит загрузку операционной системы MP/M II в память и отобразит карту памяти. Вы можете получить распечатанную копию своей карты распределения памяти при загрузке во время операции GENSYS, введя ^P до выполнения GENSYS.
2. При отладке XIOS, отметьте адрес сегмента памяти BNKXIOS.SPR или RESXIOS.SPR. Вы должны также отметить адрес SYSTEM.DAT. Если вы отлаживаете резидентный системный процесс, отметьте его адрес также. Отладчик отображает фактические адреса в консоли. Если в вашем распечатанном списке XIOS или RSP начинаются с нуля, вы должны добавить начальный адрес, перечисленный в карте распределения памяти при загрузке GENSYS к каждому адресу в листинге, чтобы листинг отражал фактические адреса. Или вы можете собрать код снова с дополнительным оператором ORG, определяющим начальный адрес, перечисленный в карте распределения памяти при загрузке, хотя объектный код, созданный при ассемблировании будет непригодным для использования.
3. Используя команду X, определите начальный адрес выполнения MP/M II. Адрес является первым местом после текущего программного счетчика.

-**X**

....................... P = 09F2 .....

В примере, показанном выше, MP/M II запускается на выполнение с адреса 09F3H, который является первой инструкцией после перезапуска в 09F2H.

1. Начните выполнение MP/M II с помощью команды G, указав начальный адрес и любые необходимые вам в коде точки останова. Фактический адрес памяти может быть определен путем ввода команды H, чтобы добавить начальный адрес сегмента кода, указанный в карте памяти к относительному смещению в вашем XIOS или списка резидентных системных процессов.

В следующем примере показано, как установить точку останова в XIOS в точках входа списка подпрограмм, приведенных в карте памяти:

...

XIOSJMP TBL C300H 0100H

-**G9F3,C30F**

09F3H - начальный адрес выполнения MP/M II и C30FH - переход XIOS к таблице списка адресов подпрограмм.

1. В данный момент у вас есть MP/M II работающий в CP/M и также отладчик CP/M находящийся в памяти. Поскольку во время работы отладчика CP/M прерывания остаются включенными, убедитесь, что код управляемый прерыванием не выполняется через контрольную точку.

Поскольку отладчик CP/M работает с не отключенными прерываниями, отладки обработчика консоли на основе прерываний является довольно сложной задачей. Подходим к этой проблеме, оставив консоль #0 в режиме опроса при отладке других консолей в режиме прерываний. После этого, может потребуется незначительная отладка для адаптации кода на основе прерываний другой консоли в консоль #0. Кроме того, рекомендуется сохранить отладочную версию вашего XIOS с вводом-выводом в консоль #0 по опросу. В противном случае будет не возможно запустить отладчик CP/M в системе MP/M II, потому что отладчик CP/M не сможет получить любой консольный ввод, так как все будет отправляться в обработчик прерываний MP/M консоли #0 .

## Непосредственно загрузка MP/M II

В системах, где MP/M II должна быть загружена непосредственно при “холодном” запуске, а не загружена и выполнена как транзитная программа в CP/M, настроенный файл MPMLDR.COM и загрузчик “холодного” запуска могут быть помещены на первые две дорожки восьмидюймового гибкого диска. Если доступна CP/M программа SYSGEN.COM, используйте ее, чтобы записать файл MPMLDR.COM на первые две дорожки. Если программа SYSGEN.COM не доступна, или если SYSGEN.COM не работает, потому что должны использоваться различные носители, такие как пятидюймовый гибкий диск или жесткий диск, пользователь должен написать две программы: простой загрузчик в память, названный GETSYS, который загружает в память загрузчик MP/M и программу под названием PUTSYS, которая помещает MPMLDR на первые две дорожки диска. Если вы реализовали BIOS CP/M 2, вы, возможно, уже подготовили GETSYS и PUTSYS.

Вместо написания программы GETSYS, вы можете использовать отладчик SID или DDT. Этот способ показан в следующем примере, который также использует SYSGEN вместо PUTSYS. Примеры скелетных программ GETSYS и PUTSYS приведены в [Разделе 1.5.3](#_Примеры_программ_GETSYS).

Чтобы автоматически загрузить и выполнить систему MP/M, вы должны также предоставить загрузчик "холодного" запуска, который загружает загрузчик MP/M в память с первых двух дорожек дискеты. Измените загрузчик CP/M 2 "холодного" запуска следующим образом: измените адрес загрузки на 0100H, и адрес выполнения на 0100H.

Следующие методы загрузки являются специфичными для Intel MDS-800 с ПЗУ начальной загрузки, который загружает первую дорожку в память начиная с адреса 3000H. Однако показанные шаги могут быть применены в общем смысле к любой пользовательской аппаратной среде.

### Подготовка загрузки MP/M II используя SYSGEN

Если доступна программа SYSGEN, используйте следующие шаги для подготовки дискеты холодного запуска MP/M II:

1. Подготовьте файл MPMLDR.COM, интегрировав настроенный вами LDRBIOS, как описано в [Разделе 1.2](#_Настройка_MPMLDR). Протестируйте MPMLDR.COM и проверьте, что он работает правильно.
2. Запустите DDT или SID.

A>**DDT**

DDT VERS 2.0

1. Используя ввод команды (I), определите, что файл MPMLDR.HEX должен быть считан и затем загрузите его в память из файла командой (R) со смещением 880H байт.

-**IMPMLDR.HEX**

-**R880**

NEXT PC

2480 0100

1. Используя команду I, определите, что файл BOOT.HEX должен быть считан и затем загрузите его в память со смещением 900H, которое необходимо для начального загрузчика. Вы можете использовать команду H, чтобы вычислить смещение.

-H900 3000

3900 D900

-**IBOOT.HEX**

-**RD900**

NEXT PC

2480 0000

1. Выйдите из отладчика, возвратившись к CCP, выполнив переход к нулевому адресу.

-**G0**

1. Используйте программу SYSGEN, чтобы записать новый загрузчик "холодного" запуска на первые две дорожки дискеты.

A>**SYSGEN**

SYSGEN VER 2.0

SOURCE DRIVE NAME (OR RETURN TO SKIP)**<cr>**

DESTINATION DRIVE NAME (OR RETURN TO REBOOT)**B**

DESTINATION ON B, THEN TYPE RETURN**<cr>**

FUNCTION COMPLETE

### Настройка генерации загрузчика MP/M II

Если программа SYSGEN не доступна, то используйте следующие шаги, чтобы подготовить дискету холодного запуска MP/M II:

1. Запишите программу GETSYS, которая читает пользовательский файл MPMLDR.COM в память начиная с адреса 3380H и загрузчик "холодного" запуска (или программу начальной загрузки) начиная с адреса 3300H. Закодируйте GETSYS так, чтобы он запускался по адресу 100H (начало TPA).

Или, как в предыдущем примере, вы можете использовать SID или DDT, чтобы выполнить эту функцию вместо того, чтобы писать программу GETSYS.

1. Выполните программу GETSYS, используя инициализированную MP/M II дискету, чтобы увидеть, загружает ли GETSYS загрузчик MP/M, запускающийся с 3380H (операционная система фактически запускается на 128 байтов позже с адреса 3400H).
2. Напишите программу, которая записывает PUTSYS из памяти, начиная с адреса 3380H обратно на первых две дорожки дискеты. Программа PUTSYS должна быть расположена по адресу 200H.
3. Протестируйте программу PUTSYS, используя пустую, неинициализированную дискету для записи части памяти на первые две дорожки. Очистите память и считайте ее назад. Протестируйте PUTSYS полностью, потому что вы будете использовать эту программу для изменения системных дискет MP/M II.
4. Используйте PUTSYS, чтобы поместить загрузчик MP/M II и загрузчик "холодного" запуска на первые две дорожки пустой дискеты.

### Примеры программ GETSYS и PUTSYS

Следующие программы служат основой для программы GETSYS и PUTSYS. Вы должны вставить подпрограммы WRITESEC, чтобы записать определенные сектора.

; Программа GETSYS – чтение дорожек 0 и 1 в память по адресу 3380H

; Регистр Использование

; A (Рабочий регистр)

; B Счетчик дорожек (0, 1)

; C Счетчик секторов (1,2,..., 26)

; DE (Рабочая регистровая пара)

; HL Адрес загрузки

; SP Определение адреса стека

;

START: LXI SP,3380H ; Установить указатель стека на рабочую область

LXI H, 3380H ; Установить начальный адрес загрузки

MVI B, 0 ; Начать с дорожки 0

RDTRK: ; Прочитать следующую дорожку (первоначально 0)

MVI C,1 ; Читать начиная с сектора 1

RDSEC: ; Прочитать следующий сектор

CALL READSEC ; Подпрограмма определенная пользователем

LXI D,128 ; Переместить адр. загрузки в след. 1/2 страницы

DAD D ; HL = HL + 128

INR C ; Sector = Sector + 1

MOV A,C ; Проверить конец дорожки

CPI 27

JC RDSEC ; Возникает Carry если сектор < 27

;

; Прийти сюда в конце дорожки, переместить следующую дорожку

INR B

MOV A,B ; Проверка последней дорожки

CPI 2

JC RDTRK ; Возникает Carry если номер дорожки < 2

;

; Прийти сюда в конце загрузки, остановить сейчас

HLT

;

; Определенная пользователем подпрограмма чтения диска

; Введите номер дорожки в регистре B,

; номер сектора в регистре C, и

; адрес для заполнения в HL

READSEC:

PUSH B ; Сохранить регистры B и C

PUSH H ; Сохранить регистры HL

.................................................

В этой точке выполнить чтение с диска, сделайте

ответвление к метке START, если происходит ошибка

.................................................

POP H ; Восстановить HL

POP B ; Восстановить регистры B и C

RET ; Назад в основную программу

END START

; Программа PUTSYS - записи дорожек 0 и 1 из памяти по адресу 3380H

; Регистр Использование

; A (Рабочий регистр)

; B Счетчик дорожек (0, 1)

; C Счетчик секторов (1,2,..., 26)

; DE (Рабочая регистровая пара)

; HL Адрес загрузки

; SP Определение адреса стека

;

START: LXI SP,3380H ; Установить указатель стека на рабочую область

LXI H, 3380H ; Установить начальный адрес загрузки

MVI B, 0 ; Начать с дорожки 0

WRTRK: ; Записать следующую дорожку (первоначально 0)

MVI C,1 ; Писать начиная с сектора 1

WRSEC: ; Записать следующий сектор

CALL WRITESEC ; Подпрограмма определенная пользователем

LXI D,128 ; Переместить адр. загрузки в след. 1/2 страницы

DAD D ; HL = HL +128

INR C ; Sector = Sector + 1

MOV A,C ; Проверить конец дорожки

CPI 27

JC WRSEC ; Возникает Carry если сектор < 27

;

; Прийти сюда в конце дорожки, переместить следующую дорожку

INR B

MOV A,B ; Проверка последней дорожки

CPI 2

JC WRTRK ; Возникает Carry если номер дорожки < 2

;

; Прийти сюда в конце загрузки, остановить сейчас

HLT

;

; Определенная пользователем подпрограмма записи диска

; Введите номер дорожки в регистре B,

; номер сектора в регистре C, и

; адрес для заполнения в HL

;

WRITESEC:

PUSH B ; Сохранить регистры B и C

PUSH H ; Сохранить регистры HL

.................................................

В этой точке выполнить запись на диск, сделайте

ответвление к метке START, если происходит ошибка

.................................................

POP H ; Восстановить HL

POP B ; Восстановить регистры B и C

RET ; Назад в основную программу

END START

## Загрузка MPM.SYS без MPMLDR

Файл MPM.SYS - полностью перемещаемый абсолютный файл, который может быть перемещен непосредственно в память и затем выполнен без использования MPMLDR. Формат файла MPM.SYS находится в Таблице 1-1, ниже.

Таблица – Формат файла MPM.SYS

|  |  |
| --- | --- |
| **Запись** | **Содержимое** |
| 1 | Первые 128 байтов страницы SYSDAT |
| 2 | Вторые 128 байтов страницы SYSDAT |
| 3-n | Операционная система MP/M в обратном порядке, вершина внизу. |

Фактическое начало страницы SYSDAT в памяти определяется в байте 000 из страницы SYSDAT. Остальная часть операционной системы MP/M II должна быть расположена непосредственно ниже страницы SYSDAT. В Таблице 1-1, n представляет число записей. Байты 120-121 из страницы SYSDAT содержат значение n. Адрес выполнения MP/M определяется адресом страницы, заданным в байте 011 страницы SYSDAT.

MPMLDR может загрузить файл MPM.SYS в память и затем переместить его к месту назначения, определенному на странице SYSDAT (байт 000). Или пользователь может написать отдельную собственную программу, чтобы сразу произвести загрузочный образ памяти из файла MPM.SYS.

## Авторское право и торговая марка Digital Research

Прочитайте свое лицензионное соглашение MP/M II. Оно определяет вашу юридическую ответственность при копировании системы MP/M II. Поместите уведомление об авторском праве:

Copyright © 1981 Digital Research

на этикетке каждой копии ваших индивидуальных дискет с MP/M II которые вы делаете. Также Digital Research просит вас разместить свой серийный номер MP/M II на этикетках любых копий, которые вы делаете. Помните также, что MP/M II является торговой маркой Digital Research, и в первый раз, когда она появляется на этикетке дискеты или в документе, должна сопровождаться товарным знаком, как показано ниже:

MP/M II™

## Организация диска

В этом разделе описывается распределение секторов системы MP/M II, в котором MPMLDR находится на первых двух дорожках дискеты одинарной плотности. Первый сектор (см. **[Таблицу 1-2](#Таблица12)**) содержит раздел с дополнительной программой начальной загрузки. Дисковые контроллеры часто создаются, чтобы перенести дорожку 0, сектор 1 в память в определенное место, часто начиная с адреса 0000H. Программа в этом секторе, называемая BOOT, несет ответственность за доставку оставшихся секторов в память, начиная с адреса 0100H. Если ваш контроллер не имеет встроенного загрузчика сектора, вы можете проигнорировать программу на дорожке 0, секторе 1, и начать загрузку с дорожки 0 сектора 2 в расположение 0100H.

В качестве примера, аппаратный загрузчик "холодного" запуска Intel MDS 800 переносит дорожку 0, сектор 1 в абсолютный адрес 3000H. После загрузки этого сектора, управление передается адресу 3000H, в котором начинается операция начальной загрузки, загружая остаток с дорожки 0 и всю дорожку 1 в память, начиная с адреса 0100H. Помните, что эта программа начальной загрузки мало полезна в среде без MDS, но полезно исследовать ее, потому что вы должны будете дублировать некоторые из ее действий в ваш собственный загрузчик холодного запуска.

Таблица – Пример организации диска MP/M II

| Номер | | | Адрес памяти | Имя модуля MP/M |
| --- | --- | --- | --- | --- |
| Дорожки | Сектора | Страницы |
| 00 | 01 |  | (адрес загрузки) | Загрузчик "холодного" запуска |
| 00 | 02 | 00 | 0100H | MPMLDR |
| " | 03 | " | 0180H | " |
| " | 04 | 01 | 0200H | " |
| " | 05 | " | 0280H | " |
| " | 06 | 02 | 0300H | " |
| " | 07 | " | 0380H | " |
| " | 08 | 03 | 0400H | " |
| " | 09 | " | 0480H | " |
| " | 10 | 04 | 050OH | " |
| " | 11 | " | 0580H | " |
| " | 12 | 05 | 0600H | " |
| " | 13 | " | 0680H | " |
| " | 14 | 06 | 0700H | " |
| " | 15 | " | 0780H | " |
| " | 16 | 07 | 0800H | " |
| " | 17 | " | 0880H | " |
| " | 18 | 08 | 0900H | " |
| " | 19 | " | 0980H | " |
| " | 20 | 09 | 0A00H | " |
| " | 21 | " | 0A80H | " |
| " | 22 | 10 | 0B00H | " |
| " | 23 | " | 0B80H | " |
| " | 24 | 11 | 0C00H | " |
| 00 | 25 | " | 0C80H | MPMLDR |
| 00 | 26 | 12 | 0D00H | LDRBDOS |
| 01 | 01 | " | 0D80H | " |
| " | 02 | 13 | 0E00H | " |
| " | 03 | " | 0E80H | " |
| " | 04 | 14 | 0F00H | " |
| " | 05 | " | 0F80H | " |
| " | 06 | 15 | 1000H | " |
| " | 07 | " | 1080H | " |
| " | 08 | 16 | 1100H | " |
| " | 09 | " | 1180H | " |
| " | 10 | 17 | 1200H | " |
| " | 11 | " | 1280H | " |
| " | 12 | 18 | 1300H | " |
| " | 13 | " | 1380H | " |
| " | 14 | 19 | 1400H | " |
| " | 15 | " | 1480H | " |
| " | 16 | 20 | 1500H | " |
| " | 17 | " | 1580H | " |
| " | 18 | 21 | 1600H | " |
| 01 | 19 | " | 1680H | LDRBDOS |
| 01 | 20 | 22 | 1700H | LDRBIOS |
| " | 21 | " | 1780H | " |
| " | 22 | 23 | 1800H | " |
| " | 23 | " | 1880H | " |
| " | 24 | 24 | 1900H | " |
| " | 25 | " | 1980H | " |
| 01 | 26 | 25 | 1A00H | LDRBIOS |

# MP/М II BIOS

## Обзор BIOS MP/M II

MP/M II BDOS и XDOS периферийные устройства доступны как "логические" устройства через BIOS и XIOS. Чтобы настроить MP/M II для определенной аппаратной среды, системный конструктор должен подготовить подпрограммы BIOS и XIOS, от которых зависят BDOS и XDOS. В этом разделе описывается логические части MP/M II взаимодействующие с BIOS. Раздел 3 описывает то же для XIOS.

BDOS и XDOS вызывают подпрограммы BIOS через "таблицу переходов", расположенную в начале BIOS как показано ниже и в Приложениях D и E. Таблица переходов - последовательность из 26 команд перехода, которые отправляют программное управление в отдельные подпрограммы BIOS. Все подпрограммы должны быть представлены в таблице переходов во время регенерации системы MP/M II. Однако, определенные подпрограммы могут быть "пустыми", т.е. они могут содержать только одну инструкцию RET.

Таблица переходов BIOS должна принять форму, показанную ниже. Отдельные адреса перехода для каждой точки входа перечислены слева. Обратите внимание на то, что точки входа XIOS сразу следуют за последней точкой входа BIOS.

BIOS+00H JMP COMMONBASE ; COMMONBASE, завершить процесс

BIOS+03H JMP WBOOT ; "Теплый" старт, завершить процесс

BIOS+06H JMP CONST ; Проверка готовности символа для консоли

BIOS+09H JMP CONIN ; Чтение входного символа с консоли

BIOS+0CH JMP CONOUT ; Вывод символа на консоль

BIOS+0FH JMP LIST ; Вывод символа на принтер

BIOS+12H JMP PUNCH ; не используется MP/M II

BIOS+15H JMP READER ; не используется MP/M II

BIOS+18H JMP HOME ; Перемещение на дорожку 0

BIOS+1BH JMP SELDSK ; Выбрать драйвер диска

BIOS+1EH JMP SETTRK ; Установить номер дорожки

BIOS+21H JMP SETSEC ; Установить номер сектора

BIOS+24H JMP SETDMA ; Установить адрес DMA

BIOS+27H JMP READ ; Чтение выбранного сектора

BIOS+2AH JMP WRITE ; Запись выбранного сектора

BIOS+2DH JMP LISTST ; не используется MP/M II

BIOS+30H JMP SECTRAN ; Подпрограмма преобразование номера сектора

Каждый адрес перехода соответствует определенной подпрограмме, которая выполняет заданную функцию, как в общих чертах описано в [Разделе 2.3](#_Таблицы_определения_дисков). Три основные функции реализованы вызовами в таблице переходов: завершение процесса через COMMONBASE и WBOOT; простой символьный ввод-вывод через CONST, CONIN, CONOUT и LIST; и дисковый ввод-вывод через HOME, SELDSK, SETTRK, SETSEC, SETDMA, READ, WRITE и SECTRAN.

Предполагается, что все простые операции ввода-вывода символов выполняться в коде ASCII, верхнем и нижнем регистре, с обнуленным старшим битом (четность). BDOS зависит только от подпрограмм CONST, CONIN и CONOUT для простого символьного ввода-вывода. Для устройств ввода данных символ ASCII ^Z (1AH) интерпретируется как условие конца файла.

## Характеристики устройств BIOS и точки входа

BIOS обычно поддерживает три типа устройств: консоли, устройства печати и диски. Характеристики каждого устройства описаны ниже.

**Консоли** - основные интерактивные устройства, которые обеспечивают связь с операторами и доступны через вызовы CONST, CONIN и CONOUT. Как правило, консоли - устройства, такие как экранные терминалы (CRT) или телетайпы. MP/M II поддерживает до 16 консолей или символьных устройств ввода-вывода.

**Устройства печати**, если они существуют в вашей системе, обычно являются устройствами бумажной копии, такими как принтеры или телетайпы. MP/M II поддерживает до 16 устройств печати.

**Диски** доступны через последовательность вызовов различных подпрограмм дискового ввода-вывода. Это функции определения номера диска для доступа, дорожки и сектора на определенном диске и адреса прямого доступа к памяти (DMA) участвующего в операциях ввода-вывода. После того как все эти параметры установлены, производится вызов функции READ или WRITE для выполнения фактической операции ввода-вывода. Обратите внимание, что часто существует один вызов SELDSK для выбора диска, а затем ряд операции чтения или записи на выбранный диск, прежде чем выбрать другой диск для последующих операций. Аналогичным образом может быть один вызов для задания адреса DMA, а затем несколько вызовов чтения или записи с выбранным адресом DMA прежде, чем адрес DMA изменяется. Подпрограммы установки номера дорожки и сектора всегда вызывается перед выполнением операций чтения или записи.

Обратите внимание, что подпрограммы READ и WRITE должны выполнить несколько попыток (стандартно 10), прежде чем сообщить состояние ошибки в BDOS. Если состояние ошибки возвращается в BDOS, он сообщает об ошибке пользователю. Подпрограмма HOME может или не может фактически выполнить поиск дорожки 00, в зависимости от характеристик вашего контроллера. Важным моментом является то, что дорожка 00 была выбрана для следующей операции и часто обрабатывается точно так же, как вызов SETTRK с параметром 00.

**Таблица 2-1** в общих чертах описывает точные обязанности каждой подпрограммы, введенной через таблицу переходов BIOS.

Таблица – Список подпрограмм BIOS

| **Подпрограмма** | **Описание** |
| --- | --- |
| COMMONBASE | Точка входа COMMONBASE устанавливает начальный адрес части XIOS, которая должна находиться в общей памяти. Точка входа COMMONBASE также содержит таблицу переходов, которая позволяет XIOS получить доступ к подпрограммам коммутации блоков пользовательской и системной памяти, диспетчеров MP/M II, XDOS и BDOS, странице SYSDAT и COLDSTART. Эффект вызова к COMMONBASE состоит в том, чтобы завершить вызов программы. Другие внешние процедуры, к которым получает доступ COMMONBASE, описаны в Разделе 2.4. |
| WBOOT | Подпрограмма WBOOT выполняет вызов XDOS завершения процесса для завершения процесса вызова. Подпрограмма должна быть реентерабельной (повторно входимой), и эта точка входа должна быть выше метки COMMONBASE. |
| CONST | Подпрограмма CONST получает состояние консольного устройства, определенного регистром D и возвращает 0FFH в регистре A, если символ готов к чтению, или 00H в регистре A, если символы консоли не готовы. Эта подпрограмма должна быть реентерабельной, и эта точка входа должна быть выше метки COMMONBASE. |
| CONIN | Подпрограмма CONIN читает следующий символ из консольного устройства, определенного регистром D в регистр A, и устанавливает бит четности (старший бит) равным нулю. Если консольный символ не готов, CONIN ожидает, ввода символа перед возвратом. Эта подпрограмма должна быть реентерабельной, и эта точка входа должна быть выше метки COMMONBASE. |
| CONOUT | Подпрограмма CONOUT отправляет символ из регистра C в консольное устройство вывода, определенное регистром D. Символ находится в коде ASCII со старшим обнуленным битом четности. Вы можете включить задержку при отправке перевода строки или возврата каретки, если ваше консольное устройство требует некоторого временного интервала в конце строки (например, терминал TI Silent 700). Вы, при необходимости можете отфильтровать управляющие символы, которые заставляют ваше консольное устройство реагировать странным способом. Например, символ ^Z заставляет терминал Lear-Seigler очищать экран и может быть отфильтрован CONOUT. Эта подпрограмма должна быть реентерабельной, и эта точка входа должна быть выше метки COMMONBASE. |
| LIST | Подпрограмма LIST отправляет символ из регистра C в устройство печати (принтер), определенный регистром D. Символ находится в коде ASCII с нулевой четностью. Эта подпрограмма должна быть реентерабельной, и эта точка входа должна быть выше метки COMMONBASE. |
| PUNCH | Устройство перфоратора не реализовано в MP/M II. Позиция в таблице переходов сохранена для поддержки совместимости с CP/M. Обратите внимание, что MP/M II поддерживает до 16 символьных устройств ввода-вывода, любое из которых может быть считывателем/перфоратором на перфоленту. |
| READER | Считывающее устройство не реализовано в MP/M II. Смотрите примечание выше для PUNCH. |
| HOME | Подпрограмма HOME возвращает головку выбранного диска в положение дорожки 00. Если ваш контроллер позволяет получить доступ к дорожке 0 с помощью флага от диска, двигайте головку до обнаружения флага дорожки 0 . Если ваш контроллер не поддерживает эту функцию, вы можете транслировать вызов HOME в вызов SETTRK с параметром 0. |
| SELDSK | Подпрограмма SELDSK выбирает дисковод, заданный регистром C для дальнейших операций, где регистр C содержит 0 для диска A, 1 для диска B, и таким образом, до 15 для диска P. При каждом выборе диска SELDSK должен возвратить в HL начальный адрес 16-байтовой области, названной заголовком параметров диска (DPH), описанном в [Разделе 2.3](#_Таблицы_определения_дисков_1). Для стандартных дисководов для гибких дисков содержание заголовка и связанных таблиц не изменяется, и таким образом сегмент программы, включенный в демонстрационный XIOS, выполняет эту работу автоматически. При попытке выбрать несуществующий диск, SELDSK возвращает HL=0000H в качестве индикатора ошибки.  На входе в SELDSK можно определить, что указанный диск был выбран в первый раз. Регистре E, бит 0 (младший бит) равен нулю, если диск не был выбран ранее. Эта информация представляет интерес в системах, которые читают конфигурационную информацию с диска, чтобы установить динамическую таблицу определений диска.  Хотя SELDSK должен возвратить адрес заголовка при каждом вызове, желательно отложить фактическую операцию выбора физического диска, пока функции ввода-вывода (чтение или запись) не происходят фактически. Это вызвано тем, что часто производят выбор диска, в конечном счете без выполнения операций ввода-вывода на диск, и многие контроллеры разгружают головки текущего диска до выбора нового привода. Эта разгрузка может вызвать чрезмерный уровень шума и износ диска.  Первый вызов подпрограммы SELDSK, MP/M II делает, только для получения адреса DIRBUF и не должен выполнять никого реального ввода-вывода. |
| SETTRK | Для подпрограммы SETTRK регистр BC содержит номер дорожки для последующего доступа к диску для выбранного в текущий момент диска. Вы можете искать выбранную дорожку сразу, или отложить поиск до следующего фактического чтения или записи. Регистр BC может принимать значения в диапазоне 0-76, соответствующие номерам дорожек для стандартных гибких дисков, и 0-65535 для нестандартных дисковых подсистем. |
| SETSEC | Для подпрограммы SETSEC, регистр BC содержит транслированный номер сектора для последующих обращений к диску на выбранном диске (см. SECTRAN, ниже). Вы можете отправить эту информацию в контроллер сразу, или вместо этого отложить выбор сектора до фактического выполнения операций чтения или записи. Регистр BC может принимать значения в диапазоне 1-26, соответствующие допустимым номерам секторов для стандартных дисководов для гибких дисков, и 0-65535 для нестандартных дисковых подсистем. |
| SETDMA | Для подпрограммы SETDMA регистр BC содержит адрес DMA (доступ памяти на дисках) для последующих операций чтения или записи. Например, если при вызове SETDMA B = 00H и C = 80H, то все последующие операции чтения, считывают свои данные в буфер начиная с 80H до 0FFH, и все последующие операции записи получают свои данные из буфера начиная с 80H до 0FFH, пока не происходит следующий вызов SETDMA. Начальный адрес DMA принимается равным 80H (относительно начала сегмента памяти, из которого был выполнен вызов). Обратите внимание, что контроллер не должен фактически поддерживать прямой доступ к памяти. Например, если все данные будут получены и отправлены через порты ввода-вывода, XIOS, который вы создаете, может использовать 128-байтовую область, начинающуюся в выбранном адресе DMA для буфера памяти во время последующих операций чтения или записи.  Особый случай происходит в подпрограмме SETDMA, когда передаваемый параметр в регистре BC содержит OFFFFH. Этот параметр указывает, что буфер заблокирован, если это происходит, должен быть очищен.  Таким образом вызов к подпрограмме SETDMA интерпретируется как вызов сброса буфера, если передан параметр 0FFFFH. Функция BDOS для сброса буферов транслируется в эту форму вызова подпрограммы SETDMA. Если операция сброса буфера, выполненная в результате передачи параметра 0FFFFH успешна, должен быть выполнен простой возврат. Однако, если происходит дисковая ошибка, текущий обратный адрес должен быть вытолкан от стека, и один из следующих кодов ошибки должен быть возвращен в регистре A:  1 - Состояние неустранимой ошибки  2 - Диск только для чтения |
| READ | Предполагая, что диск был выбран, дорожка была установлена, сектор был установлен, и адрес DMA был определен, подпрограмма READ пытается считать один сектор, основанный на этих параметрах, и возвращает следующие коды ошибки в регистре A:  0 - Нет ошибок  1 - Состояние неустранимой ошибки  Если значение в регистре A равно 0, то MP/M II предполагает, что дисковая операция была завершена успешно. Если происходит ошибка, тем не менее, XIOS должен сделать по крайней мере 10 повторных попыток, чтобы увидеть, является ли ошибка восстанавливаемой. При сообщении об ошибке, BDOS выводит сообщение, "BDOS ERR ON x: BAD SECTOR". Затем в зависимости от режима ошибки вызывающего процесса - процесс вызова завершается или возвращает код ошибки. |
|  | Дополнительный параметр, содержащий абсолютный номер записи для чтения с диска, теперь передается в MP/M II в точке входа в подпрограмму READ. Параметр составляет три байта в длину со старшим байтом в регистре B и младшими двумя байтами в регистре DE. Этот параметр может быть полезным в алгоритмах блокирования/разблокирования.  BNKXIOS MP/M II позволяет частям XIOS находиться в (не общей) памяти с коммутацией банков. Это уменьшает требования общей памяти. Код XIOS для всех дисковых операций, включая READ и WRITE, может находиться не в общей памяти за одним исключением: код, который фактически выполняет передачу данных по адресу DMA, должен находиться в общей памяти. Две дополнительных точки входа в XIOS, называемые SWTUSER и SWTSYS, позволяют переключиться между сегментом памяти пользователя и системным банком, содержащимися в BNKXIOS. Подпрограммы SWTUSER и SWTSYS описаны в [Разделе 2.4](#_Процедуры_внешнего_доступа).  Если вы выполняете разблокирование в своем коде READ и WRITE, вы должны выбрать, поместить ли ваш буфер разблокирования в общую память и затем выполнить единственное перемещение в DMA пользователя, или поместить ваш буфер разблокирования вне общей памяти. Если вы выбираете последнее, тогда вы сначала должны выполнить дополнительное перемещение сектора в общую память и затем другое перемещение в DMA пользователя. Блокирование и разблокирование описаны в Разделе 2.5. |
|  | Дополнительный параметр, содержащий абсолютный номер записи для записи на диск, теперь передается в MP/M II в точке входа в подпрограмму WRITE. Параметр составляет три байта в длину со старшим байтом в регистре B и младшими двумя байтами в регистре DE. Этот параметр может быть полезным в алгоритмах блокирования/разблокирования.  Смотрите предыдущий раздел о подпрограмме READ чтения с диска для обсуждения размещения кода подпрограммы WRITE записи на диск в памяти с коммутацией банков и разблокирования WRITE в вашем коде. |
| LISTST | Подпрограмма LISTST возвращает состояние готовности устройства печати, определенного регистром D. В регистре А возвращается значение 00, если устройство печати не готово принять символ и 0FFH, если символ может быть отправлен на принтер. Обратите внимание на то, что всегда возвращается значение 00. Подпрограмма LISTST должна быть реентерабельной. Эта точка входа сохраняется исключительно для совместимости с CP/M и может обычно быть исключена из XIOS MP/M II, поскольку ни одна из стандартных утилит не использует эту точку входа. |
| SECTRAN | Подпрограмма SECTRAN выполняет преобразование логического номера сектора к физическому и может улучшить общую реакцию MP/M II. Стандартная система MP/M II поставляется с "коэффициентом сдвига" (skew factor) равным 6, с которым пропускаются шесть физических секторов между каждой операцией логического чтения. Этот коэффициентом сдвига обеспечивает достаточный интервал времени между операциями с секторами для заполнения большинством программ свих буферов, не пропуская следующий сектор.  Для компьютерных систем, которые используют быстрые процессоры, память и дисковые подсистемы, вы можете изменить коэффициент сдвига, чтобы улучшить общую реакцию. Однако обратите внимание, что вы должны поддержать версию MP/M II совместимую с дискетами одинарной плотности IBM, использующими коэффициент сдвига 6, для передачи информации в и из вашей компьютерной системы. В целом, SECTRAN получает логический номер сектора в BC и адрес таблицы преобразования в DE. SECTRAN использует номер сектора в качестве индекса в таблице преобразования и возвращает полученный физический номер сектора в HL. Для стандартных систем таблицы и код индексации представлены в XIOS и не должны быть изменены. |

## Таблицы определения дисков BIOS

В этом разделе описаны структура и построение таблиц в BIOS, которые определяют характеристики конкретной дисковой системы, используемой в MP/M II. Эти таблицы могут быть закодированы вручную или сгенерированы автоматически с помощью утилиты DISKDEF, предоставленной с MP/M II. Элементы этих таблиц представлены ниже.

### Формат таблицы параметров дисков

В общем, каждый диск имеет соответствующий (16-байтовый) заголовок параметров диска (DPH), который содержит информацию о дисководе и обеспечивает временную область памяти для определенных операций BDOS. Формат заголовка параметров диска для каждого диска показан ниже.

**Заголовок параметров диска**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | XLT | 0000 | 0000 | 0000 | DIRBUF | DPB | CSV | ALV |  |
|  | 16b | 16b | 16b | 16b | 16b | 16b | 16b | 16b |  |

Каждый элемент - слово (16-разрядное) значение. Значения каждого элемента заголовка параметров диска (DPH) приведены в **Таблице 2-2**.

Таблица – Элементы заголовка параметров диска

|  |  |
| --- | --- |
| **Элемент** | **Описание** |
| XLT | Таблица преобразования логических секторов в физические, если она используется для данного диска или значение равное 0000H, если преобразование секторов не используется (т.е., физические и логические номера секторов являются одинаковыми). Дисководы с идентичными коэффициентами сдвига сектора совместно используют те же самые таблицы преобразования. |
| 0000 | Значения временной памяти используемые BDOS (начальное значение неважно). |
| DIRBUF | Смещение 128-байтовой области временной памяти для операций с каталогами BDOS. Все DPH адресуют ту же самую область временной памяти. Один и тот же DIRBUF используется всеми дисками. |
| DPB | Смещение блока параметров диска для этого диска. Диски с идентичными характеристиками диска адресуют тот же блок параметров диска. |
| CSV | Смещение области временной памяти, используемой программным обеспечением для проверки смены дисков. Это смещение отличается для каждого DPH. |
| ALV | Смещение области временной памяти, используемой BDOS для хранения информации о выделении памяти на диске. Это смещение отличается для каждого DPH. |

С учетом n дисководов, DPH расположены в таблице, в которой первая строка из 16 байтов соответствует диску 0, с последней строкой, соответствующей, диску n-1. Таким образом, таблица отображается в виде:

DPBASE

0 XLT00  0000 0000 0000 DIRBUF DBP00  CSV00  ALV00

1 XLT01  0000 0000 0000 DIRBUF DBP01  CSV01  ALV01

. .

. .

. .

n-1 XLTn-1 0000 0000 0000 DIRBUF DBPn-1 CSVn-1 ALVn-1

где метка DPBASE определяет смещение таблицы DPH относительно начала операционной системы.

Ответственность подпрограммы SELDSK, определенной в предыдущем разделе, состоит в том, чтобы вернуть смещение DPH от начала операционной системы для выбранного диска. Следующая последовательность операций возвращает смещение в таблице, с возвращением 0000H, если выбранный диск не существует.

NDISKS EQU 4 ; Число дисководов

......

SELDSK:

; Выбор диска N заданного в C

;

LXI H,0000H ; Готовность к ошибке

MOV A,C

CPI NDISKS ; N превышает макс. число дисков?

RNC ; Возврат если так

; 0 <= N < NDISKS

MOV L,C

DAD H ; Подготовка для \* 16

DAD H

DAD H

DAD H

LXI D,DPBASE

DAD D ; DPBASE + N \* 16

RET

Таблицы преобразования секторов (с XLT00 до XLTn-1) расположены в другом месте в BIOS, и просто соответствуют один к одному логическим номерам секторов от нуля до номера\_сектора - 1. Блок параметров дисков (DPB) для каждого диска более сложен. Конкретный DPB, который адресуется одним или более DPH, принимает общую форму:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **SPT** | **BSH** | **BLM** | **EXM** | DSM | **DRM** | **AL0** | **AL1** | **CKS** | **OFF** |
| l6b | 8b | 8b | 8b | l6b | l6b | 8b | 8b | 16b | l6b |

где каждый - байт или значение слова, как показывает индикатор "8b" или "16b" в нижнем поле. Поля определены в **Таблице 2-3**.

Таблица – Поля блока параметров диска (DPB)

|  |  |
| --- | --- |
| **Поле** | **Определение** |
| SPT | общее **количество секторов на дорожке**. |
| BSH | **коэффициент сдвига** блока распределения данных, определяется размером блока распределения данных. |
| BLM | **маска блока** распределения данных которая тоже определяется размером блока распределения данных |
| EXM | **маска экстента**, определяется размером блока распределения данных и количеством дисковых блоков. |
| DSM | определяет общую **емкость дисковода**. |
| DRM | определяет общее **количество записей каталога**, которые могут быть сохранены на этом диске. |
| AL0,AL1 | определяет **зарезервированные блоки каталога**. |
| CKS | **размер таблицы проверки каталога**, CKS 8000H помечает диск как постоянный без проверки записей каталога. |
| OFF | **число зарезервированных дорожек** в начале (логического) диска. |

Хотя эти значения таблиц автоматически производятся DISKDEF, стоит изучить вывод каждого поля, таким образом, значения могут быть проверены при необходимости. Значения BSH и BLM (неявно) определяют размер выделения данных BLS, который не является записью в блоке параметров диска. Учитывая, что вы выбрали значение для BLS, значения BSH и BLM приведены в **Таблице 2-4** ниже, где все значения в десятичной системе.

Таблица – Значения BSH и BLM для выбранного BLS

|  |  |  |
| --- | --- | --- |
| **BLS** | **BSH** | **BLM** |
| 1024 | 3 | 7 |
| 2048 | 4 | 15 |
| 4096 | 5 | 31 |
| 8192 | 6 | 63 |
| 16384 | 7 | 127 |

Значение EXM зависит от BLS и от того, является ли значение DSM меньше чем 256 или больше, чем 255, как показано в следующей таблице.

Таблица – Максимальные значения EXM

|  |  |  |
| --- | --- | --- |
| **BLS** | **DSM < 256** | **DSM > 255** |
| 1024 | 0 | Н/Д |
| 2048 | 1 | 0 |
| 4096 | 3 | 1 |
| 8192 | 7 | 3 |
| 16384 | 15 | 7 |

Значение DSM - максимальное число блоков данных, поддерживаемое этим конкретным диском, измеряемое в модулях BLS. Произведение BLS на (DSM+1) являются общим количеством байтов, сохраненных диском и, конечно, должно быть в пределах емкости физического диска, не считая зарезервированных дорожек для операционной системы.

Запись DRM на единицу меньше чем общее количество записей каталога, которые могут принимать 16-битовое значение. Значения AL0 и AL1, однако, определяются DRM. Два значения AL0 и AL1 вместе могут рассматриваться строка из 16 бит, как показано ниже.

|  |  |
| --- | --- |
| AL0 | AL1 |
| 00 01 02 03 04 05 06 07 | 08 09 10 11 12 13 14 15 |

в которой позиция 00 соответствует старшему биту байта, помеченного AL0, и 15 соответствует младшему биту байта помеченного AL1, Каждая позиция двоичного разряда резервирует блок данных для ряда записей каталога, что позволяет в общей сложности 16 блокам данных быть назначенным для записей каталога (биты присваиваются, начиная с 00 и заполняются вправо до позиции 15). Каждая запись каталога занимает 32 байта, как показано в **Таблице 2-6**.

Таблица – BLS и число записей каталога

|  |  |
| --- | --- |
| **BLS** | **Записи каталога** |
| 1024 | 32 x число\_битов |
| 2048 | 64 x число\_битов |
| 4096 | 128 x число\_битов |
| 8192 | 256 x число\_битов |
| 16384 | 512 x число\_битов |

Таким образом, если DRM = 127 (128 записей каталогов) и BLS = 1024, т.е. 32 записей каталогов на блок, требующих 4 зарезервированных блока. В этом случае, установлены 4 старшие биты AL0, в результате чего значения AL0 = 0F0H и AL1 = 00H.

Значение CKS определяется следующим образом: если диск является съемным, то CKS = (DRM+1)/4, где DRM - номер последней записи каталога. Если носитель фиксированный, то устанавливается CKS = 8000H (в этом случае записи каталога не проверяются и диск помечается как постоянный).

Наконец, поле OFF определяет число дорожек, которые пропускаются в начале физического диска. Это значение автоматически добавляется всякий раз, когда вызовется SETTRK и может использоваться как механизм для пропуска дорожек зарезервированных для операционной системы, или для разделения большого диска на более мелкие сегментированные разделы.

Чтобы завершить обсуждение DPB, напомним, что несколько DPH могут адресовать тот же DPB, если характеристики их дисков идентичны. Кроме того, DPB может динамически изменяться при адресации нового диска, просто изменяя указатель в DPH, так как BDOS копирует значения DPB в локальную область при каждом вызове функции SELDSK.

При возврате назад к DPH для определенного диска, обратите внимание, что два значения адресов CSV и ALV остаются. Оба адреса ссылаются на область неинициализированной памяти расположенной после BIOS. Области должны быть уникальными для каждого диска, и размер каждой области определяется значениями в DPB.

Размер области, адресуемой CSV, является CKS байтов, которых достаточно, чтобы содержать информацию о проверке каталога для этого конкретного диска. Если CKS = (DRM+1)/4, то вы должны зарезервировать (DRM+1)/4 байтов, используемых для проверки каталога. Если CKS = 0, что указывает на отсутствие проверки записей каталога или CKS = 8000H, помечает диск как постоянный без проверки записей каталога, то место в памяти не резервируется.

Размер области, адресуемой ALV, определяется максимальным номером блока данных, допустимым для этого конкретного диска и вычисляется как (DSM/8) +1.

### Библиотека макросов DISKDEF

Библиотека макросов под названием DISKDEF значительно упрощает процесс построения таблиц. У вас должен быть доступ к макроассемблеру MAC или перемещаемому макроассемблеру RMAC, распространяемому с MP/M II, чтобы использовать функции DISKDEF. Библиотека макросов входит в состав всех дистрибутивных дисков MP/M II.

Определение дисков BIOS состоит из следующей последовательности макро-операторов:

MACLIB DISKDEF

.....

DISKS n

DISKDEF 0,....

DISKDEF 1,...

.....

DISKDEF n-1

.....

ENDEF

где оператор MACLIB загружает файл DISKDEF.LIB (c того же диска где расположен ваш BIOS) во внутренние таблицы MAC. Далее следует вызов макроса DISKS, который определяет количество дисков, которые необходимо настроить в вашей системе, где n - целое число в диапазоне 1 - 16. Затем следует ряд макро-вызовов DISKDEF, которые определяют характеристики каждого логического диска, от 0 до n-1 (соответствующие логическим дискам от A до P). Обратите внимание на то, что DISKS и макросы DISKDEF генерируют встроенные фиксированные таблицы данных, описанные в предыдущем разделе и, таким образом, должны быть размещены в неисполнимой часть вашего BIOS, обычно непосредственно после таблицы переходов BIOS.

Оставшаяся часть вашего BIOS определяется после макросов DISKDEF с макро-вызовом ENDEF, непосредственно предшествуя оператору END. Макрос ENDEF (конец Diskdef) генерирует необходимые неинициализированные области ОЗУ, которые расположены в памяти выше вашего BIOS.

Форма макро-вызова DISKDEF

**DISKDEF dn,fsc,lsc,[skf],bls,dks,dir,cks,ofs,[k16],[prm]**

где

**dn** - число логических дисков, от 0 до n-1,

**fsc** - первый физический номер сектора (0 или 1),

**lsc** - последний номер сектора,

**skf** - необязательный коэффициент смещения сектора,

**bls** - размер блока выделения данных,

**dks** - общее количество блоков на диске,

**dir** - число записей каталога,

**cks** - число "проверяемых" записей каталога,

**ofs** - смещение дорожек к логической дорожке 00,

**k16** - необязательный флаг совместимости 1.4, который вызывает создание записи 16K/каталог,

**prm** - необязательной флаг, который указывает, что диск постоянный (не может быть удален).

Значение dn это номер диска, определенный в этом вызове макроса DISKDEF. Параметр fsc применяется для различных систем нумерации сектора и обычно 0 или 1. Параметр lsc является последним номером сектора на дорожке. Если присутствует, параметр skf определяет коэффициент смещения сектора, который используется для создания таблицы преобразования секторов в соответствии со смещением. Если количество секторов меньше чем 256, то создается однобайтная таблица, в противном случае каждый элемент таблицы преобразования занимает два байта. Если параметр skf не указан (или равен 0) таблица преобразования не создается.

Параметр bis определяет количество байтов, выделенных для каждого блока данных и принимает значения 1024, 2048, 4096, 8192 или 16384. Как правило, с увеличением размера блоков данных производительность увеличивается так, как уменьшается количество ссылок каталога, и логически соединенные записи данных физически расположены ближе на диске. Кроме того, каждая запись каталога адресует больше данных и резидентная часть BIOS занимает меньше места в ОЗУ. Параметр dks определяет общий размер диска в единицах измерения bis. Т.е. если bis = 2048 и dks = 1000, то общий объем диска составляет 2,048,000 байтов. Если dks больше, чем 255, то параметр размера блока bis должен быть больше, чем 1024. Значение dir - общее количество записей каталога, которое может превысить 255 при необходимости.

Параметр cks определяет число элементов каталога для проверки при каждом сканировании каталога и используется внутренне, чтобы обнаружить смену дисков во время работы системы. Если обнаруживается эта ситуация, MP/M II автоматически помечает диск только для чтения, чтобы впоследствии данные не были уничтожены. Как указано в предыдущем разделе, значение cks равняется dir, если диск легко сменить, что имеет место с подсистемой гибкого диска. Если диск смонтирован постоянно, то значение cks обычно равно 0 и, таким образом, параметр prm должен быть включен, чтобы указать, что диск постоянный.

Значение ofs определяет количество дорожек, которые пропускаются при адресации этого конкретного диска и могут быть использованы для резервирования дополнительного места под операционную систему или для имитации нескольких логических дисков на одном физическом диске большой емкости.

Параметр kl6 включен, когда требуется файловая совместимость с версиями CP/M 1.4, которые были изменены для поддержки дисков более высоких плотности. Этот параметр гарантирует, выделение только 16K для каждой записи каталога, что имело место в предыдущих версиях. Обычно, этому параметру присваивается ноль. Наконец, параметр prm может использоваться для указания того, что диск постоянный. Этот параметр включается, только если дисковые носители не могут быть удалены из дисковода.

Для удобства и экономии пространства таблицы, специальная форма

DISKDEF i,j

задает диску i те же характеристики которые ранее были определены для диска j.

Стандартная система с четырьмя дисками одинарной плотности, которая является совместимой с CP/M 1.4, определяется, используя следующие макро-вызовы:

DISKS 4

DISKDEF 0,1,26,6,1024,243,64,64,2

DISKDEF 1,0

DISKDEF 2,0

DISKDEF 3,0

...

ENDEF

со всеми дисками, имеющими те же значения параметров 26 секторов на дорожке (пронумерованы от 1 до 26), с 6 секторами, пропускаемыми между каждым доступом, размером блока данных 1024 байт, 243 блока данных в общей сложности обеспечивающих объем диска 243k байт, 64 записи проверяемых каталогов и две дорожки для операционной системы.

Макрос DISKS генерирует n заголовков параметров диска (DPH), начиная с адреса таблицы DPH DPBASE, созданной макросом. Каждый блок заголовка диска содержит 16 байтов, как описано выше и соответствует один к одному каждому из определенных дисков. Например, в стандартной системе из четырех дисков, макрос DISKS генерирует таблицу в виде:

DPBASE EQU $

DPE0: DW XLT0,0000H,0000H,0000H,DIRBUF,DPB0,CSV0,ALV0

DPE1: DW XLT0,0000H,0000H,0000H,DIRBUF,DPB0,CSV1,ALV1

DPE2: DW XLT0,0000H,0000H,0000H,DIRBUF,DPB0,CSV2,ALV2

DPE3: DW XLT0,0000H,0000H,0000H,DIRBUF,DPB0,CSV3,ALV3

где метки DPH включены в справочных целях, чтобы показать начало адресов таблиц для каждого диска, от 0 до 3. Значения, содержащиеся в заголовке параметра диска описаны подробно в предыдущем разделе. Адреса таблиц проверки и распределения генерируются макросом ENDEF в области оперативной памяти после кода BIOS и таблиц.

Обратите внимание, что если параметр skf (коэффициент смещения) опущен (или равен 0), таблица преобразования отсутствует, и значение 0000H вставлено в позицию XLT заголовка параметров диска для диска. В последующем вызове, чтобы выполнить преобразование логического сектора в физический, SECTRAN получает адрес таблицы преобразования DE = 0000H, и просто возвращает исходный номер логического сектора из регистровой пары BC в HL. Таблица преобразования создается, когда присутствует параметр skf, и (ненулевой) адрес таблицы помещен в соответствующий DPH. Например, таблица показанная ниже, создана, когда определен стандартный коэффициент смещения skf = 6 в макро-вызове DISKDEF:

XLT0: DB 1, 7,13,19,25, 5,11,17,23, 3, 9,15,21

DB 2, 8,14,20,26, 6,12,18,24, 4,10,16,22

После вызова макроса ENDEF определены несколько неинициализированных областей данных. Эти области данных не должны быть частью BIOS, которая загружается при холодном старте, но должны быть доступны между BIOS и концом памяти. Размер области неинициализированной памяти определяется операторами EQU, созданными макросом ENDEF. Для стандартной системы из четырех жестких дисков макрос ENDEF может произвести:

4C72 = BEGDAT EQU $

(область данных)

4DB0 = ENDDAT EQU $

013C = DATSIZ EQU $-BEGDAT

который указывает, что неинициализированная ОЗУ начинается в расположении 4C72H, заканчивается в 4DB0H-1 и занимает 013CH байт. Вы должны гарантировать, что эти адреса свободны для использования после загрузки системы.

После модификации вы можете использовать программу STAT, для проверки характеристик своего диска, потому что STAT использует блок параметров диска для декодирования сведений о диске. Форма команды STAT

STAT d:DSK:

декодирует блок параметра диска для диска d (d=A,... P) и выводит на экран значения, показанные ниже.

r: 128 Byte Record Capacity

k: Kilobyte Drive Capacity

d: 32 Byte Directory Entries

c: Checked Directory Entries

e: Records / Directory Entry

b: Records / Block

s: Sectors / Track

t: Reserved Tracks

Ниже показаны три примера вызовов макроса DISKDEF с соответствующими значениями параметра STAT. Последний пример производит систему 8 мегабайт.

DISKDEF 0,1,58, ,20 48,256,128,128,2

r=4096, k=512, d=128, c=128, e=256, b=16, s=58, t=2

DISKDEF 0,1,58, ,2048,10 24,300,0,2

r=16384, k=2048, d=300, c=0, e=128, b=16, s=58, t=2

DISKDEF 0,1,58,,16384,512,128,128,2

r=65536, k=8192, d=128, c=128, e=1024, b=128, s=58, t=2

## Процедуры внешнего доступа

Чтобы помочь XIOS доступ к другим точкам входа MP/M, программ MP/M II GENSYS создает динамическую таблицу переходов и помещает в точку входа подпрограммы COMMONBASE. Динамическая часть таблицы переходов содержит пять точек входа, которые обеспечивают доступ к коммутации блоков пользовательской и системной памяти, диспетчеров MP/M II, XDOS и странице SYSDAT. **Таблица 2-7** описывает процедуры внешних точек входа.

В следующем примере показан код, используемый для доступа к внешним процедурам:

COMMONBASE:

JMP COLDSTART

SWTUSER: JMP $-$

SWTSYS: JMP $-$

PDISP: JMP $-$

XDOS: JMP $-$

SYSDAT: DW $-$

COLDSTART:

WBOOT:

MVI C,0

JMP XDOS ; Завершить процесс

Таблица – Краткая информация о внешних процедурах

|  |  |
| --- | --- |
| **Подпрограмма** | **Описание** |
| SWTUSER | Точка входа SWTUSER восстанавливает банк пользователя в вызывающей программе. Она не имеет передаваемых или возвращаемых параметров. Цель SWTUSER состоит в том, чтобы позволить BIOS чтение и запись кода для передачи данных из контроллера диска или буфера в общей памяти к/от буфера DMA в вызывающей программе пользователя. Эта процедура должна вызываться только из общей памяти, которая располагается выше метки COMMONBASE, и она должна использоваться только из дисковых функций BIOS. Внутренне процедура SWTUSER отключает и затем повторно включает прерывания. Таким образом, если отключить прерывания перед вызовом SWTUSER они будут включены по возвращении из SWTUSER. |
| SWTSYS | Точка входа SWTSYS восстанавливает банк BNKBDOS. Она не имеет передаваемых или возвращаемых параметров. Цель SWTSYS состоит в том, чтобы восстановить банк, содержащий переключаемую часть BDOS после передачи данных из контроллера диска или буфера в общей памяти к/от буфера DMA в вызывающей программе пользователя. Эта процедура должна вызываться только из общей памяти. Внутренне процедура SWTSYS отключает и затем повторно включает прерывания. Таким образом, если отключить прерывания перед вызовом SWTSYS, они будут включены по возвращении из SWTSYS. |
| PDISP | Точка входа PDISP вызывает диспетчер вызова. Она предназначена для использования при завершении обработки прерываний, когда процесс будет обработан. Это фактически нулевой вызов процедуры с точки зрения вызывающей программы. |
| XDOS | Точка входа XDOS обеспечивает доступ к функциям XDOS. Функции XDOS требуются для операций с флагами, операций с очередями и опроса устройств. |
| SYSDAT | Запись SYSDAT не является истинной точка входа, это адрес страницы системных данных. Раздел 4 дает определение страницы системных данных. |

## Алгоритм блокирование и разблокирования

При каждом вызове точки входа WRITE BIOS, BDOS включает информацию, которая позволяет эффективно блокировать разблокировать сектора, когда дисковая подсистемы узла имеет размер сектора, кратный размеру основного модуля 128 байт. Этот раздел представляет алгоритм общего назначения для включения в ваш BIOS, который использует информацию BDOS для автоматического выполнения операции.

После каждого вызова WRITE, BDOS предоставляет следующую информацию в регистре C:

0 = отложенная запись сектора,

1 = не отложенная запись сектора,

2 = отложенная запись в первый сектор нового блока данных,

3 = не отложенная запись в первый сектор нового блока данных.

Условия 0 и 2 возникают только для постоянных дисков и позволяют отложенную запись. Условия 1 и 3 происходят для непостоянных (съемных) дисков и вызывают немедленную (не отложенную) запись. Условие 1 также происходит на постоянных дисках при записи к каталог.

Условия 2 и 3 возникают при выполнении операция записи в первый сектор нового блока данных. Алгоритм блокирования/разблокирования не выполняет предварительное чтение записи, если последовательные записи пишутся в новый блок данных. В большинстве случаев, прикладные программы читают или записывают несколько 128-байтных секторов последовательно, и, таким образом, существуют небольшие накладные расходы при задействовании блокирования и разблокирования записей в операции, потому что операции предварительного чтения можно избежать при записи записей.

Алгоритм блокирования и разблокирования в скелетной форме приведен в Приложении B. Файл присутствует на вашем диске MP/M II. Обычно алгоритмы отображают все MP/M II операции чтения сектора на диск узла через промежуточный буфер, который имеет размер сектора диска узла. Всюду в программе значения и переменные, которые касаются секторов, участвующих в операции поиска, снабжены префиксом "sek", в то время как связанные с дисковой системой узла снабжены префиксом "hst". Операторы приравнивания, начинающиеся в строке 24, определяют отображение между MP/M II и хост-системой, и должны быть изменены, если используется другая, чем в примере хост-система.

Точка входа SELDSK очищает флаг буфера узла каждый раз, при регистрации нового диска. Обратите внимание, что хотя точка входа SELDSK вычисляет и возвращает адрес заголовка параметров диска, она физически не выбирает диск узла в этой точке (он выбирается позже в READHST или WRITEHST). Кроме того, SETTRK, SETSEC и SETDMA просто хранят значения, но не предпринимают никаких других действий на данном этапе. SECTRAN выполняет простую функцию возврата физического номера сектора.

Основные точки входа READ и WRITE. Эти подпрограммы занимают место ваших предыдущих операций READ и WRITE.

Фактически физическое чтение или запись осуществляются в WRITEHST или READHST, для которых были подготовлены все значения: hstdsk - номер диска узла, hsttrk - номер дорожки узла, и hstsec - номер сектора узла (который может потребовать перевода в физический номер сектора). Вы должны ввести код в этом месте, который выполняет полное чтение сектора узла, или запись, из или в буфер hstbuf длиной hstsiz. Все другие функции отображения реализуются алгоритмом.

## Часть общей памяти BNKXIOS

Будьте осторожны при выборе какой код XIOS будет размещаться в общей памяти. Этот раздел должен дать вам некоторые полезные рекомендации.

В целом все входы XIOS и BIOS (за исключением дисковых входов ввода-вывода) должны быть выше точки входа подпрограммы COMMONBASE. Таким образом BNKXIOS позволяет вам поместить свои дисковые драйверы в часть кода, который не находится в общей памяти. Однако, есть, некоторые исключения, которые влияют на код и на области данных накопителей на магнитных дисках.

Заголовки параметров диска и блоки параметров диска должны быть в общей памяти.

Структура данных DIRBUF, на которую ссылаются блоки параметров диска, должна находиться в общей памяти.

Весь код опроса дисковых устройств и обработчики прерываний должны находиться в общей памяти.

Хотя можно поместить буфер разблокирования вне общей памяти, это потребует, размещения буфера сектора в общей памяти и дополнительного перемещения 128 байтов данных сначала в общую память и затем в пользовательский буфер DMA. Кроме того, коммутация блоков не может быть разрешена, пока производится операция физического DMA от дискового контроллера диска в буфер разблокирования в необщей памяти.

# XIOS MP/M II

## Обзор XIOS MP/М II

Расширенная система ввода-вывода (XIOS) должна включать аппаратно-зависимый код, который опрашивает устройства, обрабатывает прерывания и выполняет функции управления памятью.

Разработчик системы MP/M II должен подготовить подпрограммы, которые выполняют функции, описанные в Таблице 3-1, затем поместить таблицу переходов, содержащую точки входа XIOS сразу после таблицы переходов BIOS. Большинство подпрограмм XIOS должно быть реентерабельными (повторно входимыми). Таблица переходов XIOS должна принять следующий вид:

BIOS+33H JMP SELMEMORY ; Выбор памяти

BIOS+36H JMP POLLDEVICE ; Опрос устройства

BIOS+39H JMP STARTCLOCK ; Стартовать часы

BI0S+3CH JMP STOPCLOCK ; Остановить часы

BI0S+3FH JMP EXITREGION ; Выход из критической области

BI0S+42H JMP MAXCONSOLE ; Максимальный номер консоли

BIOS+45H JMP SYSTEMINIT ; Инициализация системы

BIOS+48H JMP IDLE ; Холостая процедура (Необязательно)

## Точки входа MP/M XIOS

Каждый адрес перехода соответствует определенной подпрограмме, которая выполняет определенную функцию. Таблица 3-1 описывают точные обязанности каждой точки входа в подпрограмму XIOS.

Таблица – Сводка подпрограмм XIOS

| **Подпрограмма** | **Функция** |
| --- | --- |
| SELMEMORY | Подпрограмма SELMEMORY идентифицирует сегмент памяти, в которой должен выполниться процесс. Каждый раз, когда процесс передается диспетчером для выполнения, операционная система вызывает эту процедуру выбора памяти XIOS. Если у аппаратной среды есть выбор/защита сегмента памяти, SELMEMORY может использовать передаваемый параметр, для выбора/защиты области памяти. Переданный параметр (в регистрах BC) является указателем на дескриптор памяти, который определяет начало памяти, размер, атрибуты и банк выполнения процесса. Таким образом, все другие области памяти могут быть защищены от записи. |
|  | MP/M II вызывает SELMEMORY с отключенными прерываниями из диспетчера. Подпрограмма SELMEMORY не должна разрешать прерывания. Эта подпрограмма должна находиться выше точки входа COMMONBASE. |
| POLLDEVICE | Среда опроса может быть создана, кодированием обработчика опроса устройства XIOS. Цель реализации среды опроса состоит в том, чтобы избежать типичного кода ожидания в состоянии занятости для завершения работы устройства. Существуют также периферийные устройства, которые не могут эффективно работать при прерываниях. XDOS вызывает обработчик опроса устройства (POLLDEVICE) с единственным параметром в регистре C, определяющем опрашиваемое устройство. Написанная пользователем процедура POLLDEVICE может быть закодирована, для получения доступа к подпрограммам опроса устройства через таблицу, которая содержит адреса процедур опроса устройств. Устанавливается зависимость между номером опрашиваемого устройства и самой процедурой опроса. Процедуры опроса должны возвратить в аккумуляторе значение 0FFH, если устройство готово, или 00H, если устройство не готово. POLLDEVICE вызывают из критической области в диспетчере, поэтому, подпрограмма POLLDEVICE не должна включать прерывания. Эта подпрограмма должна находиться выше точки входа COMMONBASE. |
| STARTCLOCK | Процедуры STARTCLOCK и STOPCLOCK устраняют ненужные накладные расходы для обработчика прерываний от системных часов. Системные часы обеспечивают отсчет времени для флага реального времени и процедуры системных тактов (system tick). Однако процедура системных тактов необходима только когда, процесс находится в списке задержки. MP/M II вызывает STARTCLOCK, когда процесс переходит в список задержки, чтобы инициировать отсчет времени системных тактов (см. Раздел 3.4).  В некоторых аппаратных средах нет возможности отключить модуль часов системного времени при сохранении одно секундного флага, используемого для хранения времени суток. В этой ситуации процедура STARTCLOCK просто устанавливает логическую переменную в истину (true), указывая, что есть отложенный процесс. Обработчик прерываний часов может тогда определить, должен ли флаг модуля системного времени быть установлен, тестируя логическую переменную. Эта подпрограмма должна находиться выше точки входа COMMONBASE. |
| STOPCLOCK | Когда системный список задержки очищается, MP/M II вызывает процедуру STOPCLOCK для остановки отсчета времени системных тактов. Это исключает ненужные накладные расходы обработчика прерываний часов системы.  В некоторых аппаратных средах не возможно отключить модуль часов системного времени при сохранении одно секундного флага, используемого для хранения времени суток, т.е. используется единственный источник часов/прерываний по таймеру. |
|  | В этой ситуации процедура STOPCLOCK просто устанавливает логическую переменную в ложь (false), указывая на отсутствие отложенных процессов. Обработчик прерываний часов может тогда определить, должен ли флаг модуля системного времени быть установлен, тестируя логическую переменную. Эта подпрограмма должна находиться выше точки входа COMMONBASE. |
| EXITREGION | MP/M II вызывает процедуру EXITREGION, чтобы протестировать локальный параметр называемый флаг PREEMPT. Если PREEMPT - истина, EXITREGION оставляет прерывания отключенными. Если PREEMPT - ложь, EXITREGION разрешает прерывания. Процедуры обработки прерываний должны устанавливать флагу "PREEMPT" значение истина в начале обработки прерываний. Эта процедура позволяет процедуре обработки прерывания установить флаг системных вызовов MP/M II, оставляя прерывания отключенными до завершения обработки прерываний. Эта подпрограмма должна находиться выше точки входа COMMONBASE. |
| MAXCONSOLE | Процедура MAXCONSOLE позволяет вызывающей программе определить количество физических консолей, поддерживаемых BIOS. Количество физических консолей возвращается в регистре A. Эта подпрограмма должна находиться выше точки входа COMMONBASE. |
| SYSTEMINIT | Процедура инициализации системы выполняет требуемую инициализацию холодного запуска MP/M. Ниже приведена типичная инициализации для системы с коммутацией банков: во-первых, MP/M II инициализирует банк 0, отключает прерывания и вызывает SYSTEMINIT. Затем, SYSTEMINIT устанавливает таблицу переходов прерываний, маски прерываний и базовую страницу каждого банка до возвращения к MP/M II. Наконец, MP/M II разрешает прерывания. Типичная инициализации для системы без коммутации банков будет выполнять те же шаги, но будет инициализироваться только один банк.  MP/M II отключает прерывания и вызывает точку входа SYSTEMINIT до любого другого вызова XIOS. Как указано выше, MP/M II включает прерывания по возврату из SYSTEMINIT. Эта подпрограмма должна находиться выше точки входа COMMONBASE.  В системах с коммутацией банков памяти необходимо установить базовую страницу (0000H-00FFH) в каждом банке памяти. Как MPMLDR, так и сама MP/M предполагают, что основной банк (банк 0) переключен во время выполнения MPMLDR. Основной банк 0 корректно инициализируется MP/M до ввода SYSTEMINIT. Информация, необходимая для инициализации других банков, предоставляется входу SYSTEMINIT в регистрах, определенных ниже:  C - номер рестарта отладчика MP/M  DE - Адрес точки входа отладчика в MP/M.  Поместите в требуемое место переход перезапуска отладчика с адресом, содержавшимся в DE. |
|  | HL - Прямой адрес переходов BIOS. Поместите инструкцию перехода по месту 0000H в базовой странице каждого банка адрес, указанный в HL. |
| IDLE | Процесс IDLE - привязка списка готовых процессов. Ядро MP/M II вызывает процедуру IDLE, в случае, если нет других процессов готовых к выполнению. Обычная процедура IDLE является вызовом диспетчера. Это наиболее эффективно обеспечивает опрос устройств. Если ваша система полностью управляется прерываниями (т.е. без опрашиваемых устройств), вы можете предоставить свою собственную процедуру IDLE, которая может быть следующей:  IDLE:  HALT  RET  Если вы не предоставляете процедуру IDLE, поместите три байта нулей в расположение BIOS+48H. |

## Процедуры обработки прерывания

Операционная система MP/M II предназначена для работы с практически любой архитектурой прерываний, будь то одноуровневой или векторной. Код работающий по прерыванию: на соответствующем уровне сохраняет необходимые регистры, определяет причину прерывания, снимает состояние прерывания, устанавливает соответствующий флаг и затем передает управление диспетчеру.

Обязательно используйте минимальное число уровней стека при сохранении состояния прерванного процесса. Это вызвано тем, что прерванная прикладная программа, особенно если она была написана для среды CP/M, вряд ли обеспечивает дополнительную область стека предусмотренную для прерываний. Примеры расширенной системы ввода-вывода, показанные в приложениях, иллюстрируют метод, который не требует дополнительных уровней стека кроме перезапуска самого прерывания. Этот метод настоятельно рекомендуется.

Работа флагов описана в *Разделе 3 Руководства программиста MP/M II*, при рассмотрении функций XDOS установка флага и ожидание флага. Кратко, флаги синхронизируют процесс с асинхронным событием. В целом процедура обработки прерывания устанавливает определенный флаг, в то время как другой процесс ожидает флаг, который будет установлен.

На логическом уровне выше физического, флаги прерываний могут рассматриваться как обеспечивающие 256 уровней виртуальных прерываний (в MP/M II поддерживаются 32 флага). Таким образом, логические обработчики прерываний ожидают флаги, установленные физическими обработчиками прерываний. Этот механизм позволяет общему XDOS работать потенциально на всех микрокомпьютерах 8080, 8085 и Z80®, независимо от аппаратной среды.

В качестве примера, рассмотрим аппаратную среду с линейной структурой прерываний. Т.е. обеспечивается один уровень прерываний, и устройства должны быть опрошены для определения причины прерывания. Как только причина прерывания определена, устанавливается определенный флаг, указывающий, что произошло это конкретное прерывание.

В конце обработки прерывания должен быть сделан переход к диспетчеру MP/M II. Это делается путем перехода к точке входа PDISP. Этот переход предоставляет ресурс процессора готовому процессу с самым высоким приоритетом, как правило, процесс готовится установив флаг в обработчике прерывания, а затем разрешает прерывания перед переходом, чтобы возобновить выполнение этого процесса.

Из обработчика прерываний должен быть выполнен только XDOS или BDOS вызов 133: установка флага. Любые другие вызовы XDOS или BDOS отправляются в диспетчер, который затем разрешает прерывания, прежде чем выполнение обработчика прерываний будет завершено.

Рекомендуется использовать прерывания только для асинхронных операций, таких как консольный ввод или завершенная дисковая операция. В целом такие операции, как консольный вывод не должны управляться прерываниями, потому что система обладает большей гибкостью, при выполнении консольного вывода по опросу во время простоя, не увеличивая накладные расходы диспетчера для каждого передаваемого символа. Это - особенно справедливо для более высоких скоростей передачи данных.

Если система требует выполнения возврата из прерывания инструкции (RETI), обработчик прерываний должен выполнить RETI прежде, чем диспетчер перейдет через точку входа PDISP.

## Управление отсчётом времени

XIOS должен обеспечить две системы отсчёта времени: односекундный флаг для реального времени и системные такты для управления списком задержки. Операция односекундного флага логически отделена от операции системных тактов, хотя они физически могут совместно использовать один и тот же источник прерывания часов/таймера. Процедура односекундного флага устанавливает флаг #2 в каждую секунду реального времени. MP/M II использует флаг #2, для поддержки часов времени суток.

Процедура системных тактов, когда включена STARTCLOCK, устанавливает флаг #1 в единичные интервалы системного времени. Рекомендуемая единица измерения времени - период 16.67 миллисекунд, соответствующий частоте тактов 60 Гц. При работе с 50 Гц используйте период 20 миллисекунд. MP/M II использует, системные такты для управления списком задержки пока список задержки не станет пустым, в котором процедура времени системных тактов будет отключена STOPCLOCK.

Частота системных тактов критически важна, потому что она определяет частоту отправки для ограниченных по скорости процессов. Если частота слишком высока, чрезмерными отправками существенно загружают систему. Если частота слишком низкая, соответственно ограниченные по скорости процессы, сохраняют ресурс ЦП в течение более длительных периодов времени.

# Компоненты системного файла MP/M II

Системный файл MP/M II - MPM.SYS, состоит из многих компонентов: страницы системных данных, настроенного XIOS, RESBDOS и BNKBDOS, XDOS и BNKXDOS, TMP и резидентных системных процессов. MPM.SYS находится в каталоге с пользовательским кодом 0 и обычно имеет атрибут "только для чтения". Загрузчик MP/M II читает файл MPM.SYS в память, чтобы перевести систему MP/M II в рабочее состояние.

## Системные данные

Страница системных данных содержит 256 байт, используемых GENSYS для динамической настройки системы MP/M II. Страница системных данных может быть подготовлена, используя программу GENSYS, или она может быть подготовлена вручную, используя DDT или SID. **Таблица 4-1** описывает назначения байтов.

Таблица – Назначения байтов системной страницы данных

| **Байт** | **Содержание** |
| --- | --- |
| 000-000 | Mem$top, вершина страницы памяти |
| 001-001 | Nmb$cns, число системных консолей (процессов терминальных сообщений) |
| 002-002 | Brkpt$RST, номер рестарта точки останова |
| 003-003 | Добавление системного вызова пользовательских стеков, логическая |
| 004-004 | Коммутация банков, логическая |
| 005-005 | Версия Z80, логическая |
| 006-006 | Переключаемый BDOS, логическая |
| 007-007 | Страница таблицы переходов XIOS |
| 008-008 | Начальная страница RESBDOS |
| 009-010 | Адрес главной таблицы конфигурации CP/NET |
| 011-011 | Начальная страница XDOS |
| 012-012 | Начальная страница RSP (BNKXIOS top+1) |
| 013-013 | Начальная страница BNKXIOS |
| 014-014 | Начальная страница BNKBDOS |
| 015-015 | Max$mem$seg, максимальный номер сегмента памяти |
| 016-047 | Первоначальная таблица сегментов памяти |
| 048-063 | Таблица векторов точек останова, заполненная отладчиками |
| 064-079 | Зарезервированы для MP/M II |
| 080-095 | Системный вызов пользовательской таблицы указателя стека |
| 096-119 | Зарезервированы для MP/M II |
| 120-121 | Количество записей в файле MPM.SYS |
| 122-122 | Число тактов/сек |
| 123-123 | Системный диск |
| 124-124 | Начальная страница общей памяти |
| 125-125 | Число резидентных системных процессов (RSP) |
| 126-127 | Массив адресов Listcp |
| 128-143 | Subflg, массив флагов submit |
| 144-186 | Зарезервированы для MP/M II |
| 187-187 | Максимальное количество заблокированных записей/процессов |
| 188-188 | Максимальное количество открытых файлов/ процессов |
| 189-190 | Число устройств печати |
| 191-192 | Указатель на начало таблицы блокировок свободного места |
| 193-193 | Общая число записей заблокированных системой |
| 194-194 | Общее число файлов открытых в системе |
| 195-195 | Журнал регистрации, логическая |
| 196-196 | Диск для временных файлов |
| 197-197 | Число принтеров |
| 197-241 | Зарезервированы для MP/M II |
| 242-242 | Начальная страница переключаемой XDOS |
| 243-243 | Начало дескриптора процессов TMP |
| 244-244 | Начало CONSOLE.DAT |
| 245-246 | Точка входа BDOS/XDOS |
| 247-247 | Начало TMP.SPR |
| 248-248 | Nmbrsps, Число переключаемых RSP |
| 249-249 | Начальный адрес BRSP |
| 250-251 | Brspl, на не резидентные процессы RSP |
| 252-253 | Sysdatadr, Адрес сегмента внутренних данных XDOS |
| 254-255 | Rspl, ссылка на резидентный системный процесс |

## Настроенный XIOS

Настроенный XIOS получается или из файла под названием RESXIOS.SPR или из файла под названием BNKXIOS.SPR. Файл XIOS c расширением SPR содержит странично перемещаемую версию настроенного пользователями XIOS. Стандартный метод генерации XIOS использует программу LINK Digital Research. Альтернативный метод описан в Разделе 1.

## BDOS

Базовая дисковая операционная система (BDOS) находится в двух странично-перемещаемых файлах, названных RESBDOS и BNKBDOS. Эти два файла содержат код консоли, печатающего устройства и управления файлами на диске.

### RESBDOS

Файл с названием RESBDOS.SPR – странично перемещаемый файл, содержащий обработку логической консоли и устройства печати, а также резидентную часть дисковой файловой системы, которая обеспечивает интерфейс с BNKBDOS.

### BNKBDOS

Файл под названием BNKBDOS.SPR - странично перемещаемый файл, содержащий нерезидентную часть переключаемой BDOS.

## XDOS

Файл XDOS с именем XDOS.SPR - странично перемещаемый файл, содержащий приоритетное ядро MP/M II. Ядро содержит следующие фрагменты кода: корневой модуль, диспетчер, управление очередью, управление флагами, управление памятью, обработчик терминала, процесс сообщений терминала, интерпретатор командной строки, синтаксический анализатор имен файлов и управление отсчётом времени.

## Резидентные системные процессы

Расширение файла RSP идентифицирует резидентный системный процесс. Файлы RSP, распространяемые в дистрибутиве MP/M II, включают: отображение состояния системны во время выполнения (MPMSTAT), диспетчер печати (SPOOL), аварийное прекращение работы именованных процессов (ABORT) и планировщик (SCHED). Во время генерации системы GENSYS предлагает вам выбирать какие RSP включать в файл MPM.SYS.

Пользователи могут подготовить пользовательские резидентные системные процессы. Резидентные системные процессы должны соответствовать следующим правилам:

* Файл должен быть странично перемещаемым. Странично перемещаемые файлы могут быть сгенерированы с помощью редактора связей LINK, или командных файлов MACSPR.SUB или ASMSPR.SUB. Выходной файл должен быть переименован с присвоением расширения RSP.
* Первые два байта резидентного системного процесса зарезервированы для адреса BDOS/XDOS. Таким образом резидентный системный процесс может получить доступ к BDOS/XDOS, загрузив два байта из относительного адреса 0000-0001H и затем выполнить PCHL.
* Дескриптор процесса для резидентного системного процесса должен начаться с позиции третьего байта.

## Переключаемые резидентные системные процессы

Переключаемый резидентный системный процесс состоит из двух частей: резидентной части и кода процесса. Резидентная часть содержит дескриптор процесса, и очереди или другие структуры данных, которые должны располагаться в общей памяти. Эта часть следует правилам, приведенным выше для резидентных системных процессов. Присутствие переключаемой части определяется присвоением индексу сегмента памяти дескриптора процесса значения нуля вместо 0FFH. Имя указанное в дескрипторе процесса используется для получения переключаемой части, которая имеет расширение файла BRS.

Вторая часть переключаемого системного процесса фактически является часть кода для процесса. Правила для части BRS следующие:

* Файл должен быть странично перемещаемым. Странично перемещаемые файлы может быть сгенерирована с помощью редактора связей LINK или используя процедуру, описанную в Разделе 1. Выходной файл должен быть переименован с присвоением расширения BRS.
* Байты 0000-0001H в переключаемом RSP, зарезервированы для адреса резидентной части RSP. Таким образом переключаемый RSP должен получить доступ к функциям BDOS/XDOS, косвенно загрузив два байта из относительного адреса 0000-0001H, которые указывают на начало резидентной части RSP, которые в свою очередь содержат адрес точки входа BDOS/XDOS.
* Байты 0002-0003H в переключаемом RSP, должны содержать начальное значение указателя стека для процесса. Таким образом стек для переключаемого RSP находится в переключаемой части RSP и должен быть инициализирован таким образом, что бы обратный адрес вершины стека являлся адресом точки входа переключаемого RSP.
* Байты, 0004-000BH в переключаемом RSP, должны содержать ASCII имя процесса. Оно используется для отображения во время выполнения GENSYS и MPMLDR.

# Генерация системы

## Работа GENSYS

Генерация системы MP/M II состоит из подготовки системного файла данных и объединения обязательных и дополнительных файлов кода для создания файла с названием MPM.SYS. Программа GENSYS выполняет эти задачи и может быть выполнена в MP/M II или в CP/M. GENSYS автоматизирует процесс генерации системы, запрашивая у пользователя дополнительные параметры и затем готовит файл MPM.SYS. Следующий пример иллюстрирует выполнение операции GENSYS.

0A>gensys

MP/M II V2.0 System Generation

Copyright (C) 1981, Digital Research

Default entries are shown in (parens).

Default base is Hex, precede entry with # for decimal

Use SYSTEM.DAT for defaults (Y) ?

Top page of operating system (FF) ?

Number of TMPs (system consoles) (#2) ?

Number of Printers (#1) ?

Breakpoint RST (06) ?

Add system call user stacks (Y) ?

Z80 CPU (Y) ?

Number of ticks/second (#60) ?

System Disk (E:) ?

Temporary file drive (E:) ?

Maximum locked records/process (#16) ?

Total locked records/system (#32) ?

Maximum open files/process (#16) ?

Total open files/system (#32) ?

Bank switched memory (Y) ?

Number of user memory segments (#3) ?

Common memory base page (C0) ?

Dayfile logging at console (Y) ? <- Вывод в консоль информации о

выполнении файла

SYSTEM DAT FF00H 0100H

TMPD DAT FE00H 0100H

USERSYS STK FD00H 0100H

XIOSJMP TBL FC00H 0100H

Accept new system data page entries (Y) ?

RESBDOS SPR F000H 0C00H

XDOS SPR CE00H 2200H

Select Resident System Processes:

SCHED RSP (N) ?

ABORT RSP (N) ? y

SPOOL RSP (N) ? y

MPMSTAT RSP (N) ? y

ABORT RSP CD00H 0100H

SPOOL RSP CC00H 0100H

MPMSTAT RSP CB00H 0100H

BNKXIOS SPR B800H 1300H

BNKBDOS SPR 9500H 2300H

BNKXDOS SPR 9200H 0300H

TMP SPR 8F00H 0300H

SPOOL BRS 8700H 0800H

MPMSTAT BRS 7900H 0E00H

LCKLSTS DAT 7700H 0200H

CONSOLE DAT 7500H 0200H

Enter memory segment table:

Base,size,attrib,bank (75,8B,80,00) ?

Base,size,attrib,bank (00,C0,00,01) ?

Base,size,attrib,bank (00,C0,00,02) ?

Base,size,attrib,bank (00,C0,00,03) ? 00,ff,0,0

\*\*\* Memory conflict - segment trimmed \*\*\*

Base,size,attrib,bank (00,75,00,00)?

MP/M II Sys 7500H 8B00H Bank 00

Memseg Usr 0000H C000H Bank 01

Memseg Usr 0000H C000H Bank 02

Memseg Usr 0000H 7500H Bank 00

Accept new memory segment table entries (Y) ?

\*\* GENSYS DONE \*\*

## Параметры генерации системы

В этом разделе рассматриваются вопросы, связанные с ответами на каждый из запросов GENSYS, показанного в примере выше.

### По умолчанию

Программа GENSYS выводит на экран входные значения по умолчанию в круглых скобках. Основание системы счисления - шестнадцатеричное если значению не предшествует символ #, определяющий десятичное основание. Начальная подсказка определяет, должны ли использоваться внутренние значения по умолчанию GENSYS, или из сгенерированного в последний раз файла SYSTEM.DAT.

### Вершина страницы операционной системы

Введите две шестнадцатеричные цифры ASCII, чтобы задать вершину страницы операционной системы. Самый верхний адрес, используемый MP/M II, является XXFFH, где XX введенное значение.

### Количество консолей системы

Этот параметр определяет количество консолей системы, для которых создаются процессы терминальных сообщений (TMP), для генерации пользовательских подсказок и отправки командных строк интерпретатору командной строки (CLI). Область общей памяти под названием TMPD.DAT зарезервирована для дескрипторов процессов TMP. Четыре дескриптора процессов TMP могут быть помещены в каждой странице TMPD.DAT. Каждая системная консоль также требует 256 байт памяти для стека и буферных зон в нерезидентной области памяти под названием CONSOLE.DAT. MP/M II поддерживает максимум до 16 консольных символьных устройств ввода-вывода, из которых 8 могут быть системными консолями и связанными с TMP. Во время инициализаций MP/M II вызов XIOS получает фактическое максимальное количество физических консолей, поддерживаемых XIOS. Это количество используется, если оно меньше, чем количество, определенное во время GENSYS.

### Количество принтеров

Этот параметр определяет количество физических принтеров, поддерживаемых XIOS. Это число используется программой MPMSTAT, когда она выводит на экран состояние системных принтеров.

### Номер рестарта точки останова

Введите номер рестарта точки останова, который будет использоваться отладчиками MP/M. Рекомендуемые рестарты от RST #1 до RST #6.

### Системный вызов пользовательского стека

Если вы хотите выполнять \*.COM файлы CP/M, введите yes. Утвердительный ответ заставляет выполнять переключение стека, когда системные вызовы выполняются из программы пользователя. Вызовы BDOS требуют больше места для стека в MP/M II, чем в CP/M. Утвердительный ответ заставляет GENSYS выделять область общей памяти под названием USERSYS.STK. Размер этой области определяется числом сегментов пользовательской памяти, для 0-3 сегментов требуются 100H байт, и 4-7 сегментов требуются 200H байт.

Обратите внимание на то, что это влияет только на вызовы BDOS, не затрагивая вызовы XDOS. XDOS является реентерабельной и не выполняет переключения стека. Поэтому, если ваша программа выполняет любые вызовы XDOS, вы должны удостовериться, что выделили достаточно места под стек.

### Процессор Z80

Должен быть сделан утвердительный ответ только, если ваш процессор действительно Z80. Если определено, диспетчер MP/M II сохраняет и восстанавливает набор альтернативных регистров Z80.

### Число тактов в секунду

Значение этого параметра может использоваться программами приложений, чтобы определить число тактов в секунду. Это значение может меняться в зависимости от системы MP/M II.

### Системный диск

Диск, введенный здесь, используется для второго поиска, если запрашиваемый файл в CLI не найден на диске по умолчанию.

### Диск для временных файлов

Диск, введенный здесь, используется в качестве диска для временных файлов на диске. Этот параметр используется SUBMIT, когда она генерирует временные файлы $n$.SUB. Также к этому параметру прикладные программы могут получить доступ в странице системных данных в качестве диска, на котором можно создать временные файлы.

### Максимальное число заблокированных записей/процессов

Этот параметр определяет максимальное количество записей, которые единственный процесс (обычно одна программа) может заблокировать в любой момент времени. Это число может колебаться от 0 до 255 и должно быть меньше или равно общего числа заблокированных записей системы.

### Общее число заблокированных записей системы

Этот параметр определяет общее количество заблокированных записей для всех процессов, выполняющихся в MP/M II в любой момент времени. Это число может колебаться от 0 до 255 и должно быть больше или равно максимальному числу заблокированных записей для каждого процесса.

Можно разрешить каждому процессу использовать либо общую систему блокировки записей выделенной памяти, или позволить каждому процессу, блокировать лишь часть доступную всей системе. Первый метод предполагает динамическую область памяти, в которой один процесс может заблокировать другие процессы, если он израсходует все доступные ресурсы.

### Максимальное количество открытых файлов/процессов

Этот параметр определяет максимальное количество файлов, которые один процесс (обычно одна программа) может открыть в любой момент времени. Это число может колебаться от 0 до 255 и должно быть меньше или равно общему количеству открытых файлов в системе.

### Общее количество открытых файлов в системе

Этот параметр определяет общее количество открытых файлов для всех процессов, выполняющихся в MP/M II в любой момент времени. Это число может колебаться от 0 до 255 и должно быть больше или равно максимальному количеству открытых файлов для каждого процесса.

Можно разрешить каждому процессу использовать общий ресурс открытых файлов в системе или позволить каждому процессу открывать только часть общего для системы количества. Первый метод предполагает динамическую область памяти, в которой один процесс может заблокировать другие процессы, если он израсходует все доступные ресурсы.

### Память с коммутацией банков

Если ваша система не использует память с коммутацией банков, то вы должны ответить "N". В противном случае отвечайте "Y" и последующие дополнительные вопросы (как показано в Разделе 5.2.2).

### Число сегментов пользовательской памяти

Число сегментов пользовательской памяти должно быть в диапазоне 1 - 7 и должно быть больше или равно числу системных консолей.

### Начальная страница общей памяти

В ответ на этот запрос, введите адрес нижней страницы памяти общей для всех банков. GENSYS проверяет, что все модули, требующие постоянное место в общей памяти расположены выше этого адреса.

### Вывод в консоль информации о выполнении файла

Утвердительный ответ заставляет сгенерированную систему MP/M II отображать текущее время, имя и расширение файла и номер пользователя для каждой выполненной команды.

### Принять записи системных данных

Если значения, введенные в первых 16 запросах, приемлемы, то введите YES. Иначе, любые из заданных параметров могут быть изменены, с помощью повторных запросов GENSYS, вводя возврат каретки, где значения не должны быть изменены.

### Выбор резидентных системных процессов

GENSYS ищет в каталоге все файлы с расширением RSP. Каждый найденный файл перечисляется и включается в сгенерированный системный файл, если вы отвечаете "Y". Выполняется проверка, чтобы удостовериться, что указанные RSP находятся в или выше адреса общей базы.

### Таблица сегментов памяти

Сегментация памяти определяется введенными параметрами. Вам будет предложено определить начало, размер, атрибуты и банк для каждого сегмента памяти. Программа GENSYS позволяет ввести только количество сегментов, указанное в ответе на запрос о количестве сегментов памяти пользователя.

По умолчанию первый введенный сегмент устанавливается для операционной системы. Он становится нулевой записью в таблице сегментов памяти. Он включается в MP/M II с переключением банков при выполнении BNKXIOS, BRS и BNKBDOS. Первая запись не учитывается в вашем числе сегментов памяти пользователя.

Используя битовую карту памяти выполняется большое количество проверок на наличие ошибок, чтобы гарантировать, что никакие сегменты памяти не накладываются друг на друга. Программу GENSYS можно настроить, таким образом, что несуществующая память для конкретной аппаратной конфигурации будет предварительно отмечена на битовой карте.

Порядок записей в таблице сегментов памяти также критически важен. Первая запись зарезервирована для операционной системы. Остальные записи могут быть определены пользователем. При задании сегментов памяти пользователя абсолютные области TPA (сегменты, начинающиеся с 0000H), должны быть определены в порядке убывания, от большего к меньшему. Ввод сегментов в этом порядке заставляет диспетчер памяти MP/M II выделять самую большую доступную область TPA для выполнения программы COM, потому что он осуществляет линейный поиск в таблице сегментов памяти для первого доступного сегмента, начинающегося с нуля. Упорядочивание перемещаемых сегментов (не начинающихся с 0000H) не является критически важным, потому что диспетчер памяти MP/M II подбирает наиболее подходящий вариант для этих сегментов.

Байт атрибута обычно определяется как 00. Однако, если вы хотите предварительно выделить сегмент памяти, определите значение FFH.

Значение байта банка является индексом, который может использоваться XIOS для получения значение, которое будет отправлено в аппаратные средства коммутации банков для выбора указанного банка. Значения 0, 1, 2... используются для идентификации сегментов памяти. Значение байта банка 0 используется для нерезидентной части MP/M II.

### Принять таблицу сегментов памяти

Отрицательный ответ на этот запрос позволяет изменить записи сегментов памяти до принятия.

## Выполнение GENSYS

У программы GENSYS есть автоматический режим, который упрощает повторную генерацию файлов MPM.SYS. Он полезен в режиме отладки при тестировании, редактировании XIOS и последующем выполнения GENSYS для создания нового файла MPM.SYS. Автоматический режим определяется следующим образом:

0A>**GENSYS $A**

Автоматический режим применяется для имитации ввода <cr> для каждого запроса GENSYS.

# Загрузчик MP/M

## Работа и вывод загрузчика MP/M

Программа MPMLDR загружает файл MPM.SYS и ответвления к адресу выполнения операционной системы MP/M II. MPMLDR может быть выполнен в CP/M или загружен из первых двух дорожек диска загрузчиком "холодного" запуска.

MPMLDR выводит на экран системную загрузку и конфигурацию. Это не требует никаких действий от оператора. В следующем примере файл MPM.SYS, подготовленный в первом примере GENSYS, показанный в [Разделе 5](#_Работа_GENSYS), загружен в память и выполнен.

MP/M-II V2.0 Loader

Copyright (C) 1981, Digital Research

Nmb of consoles = 2

Breakpoint RST # = 6

Z80 Alternate register set saved/restored by dispatcher

Memory Segment Table:

SYSTEM DAT FF00H 0100H

TMPD DAT FE00H 0100H

USERSYS STK FD00H 0100H

XIOSJMP TBL FC00H 0100H

RESBDOS SPR F000H 0C00H

XDOS SPR CE00H 2200H

ABORT RSP CD00H 0100H

Spool RSP CC00H 0100H

MPMSTAT RSP CB00H 0100H

BNKXIOS SPR B800H 1300H

BNKBDOS SPR 9500H 2300H

BNKXDOS SPR 9200H 0300H

TMP SPR 8F00H 0300H

Spool BRS 8700H 0800H

Mpmstat BRS 7900H 0E00H

LCKLSTS DAT 7700H 0200H

CONSOLE DAT 7500H 0200H

-------------------------

MP/M II Sys 7500H 8B00H Bank 0

Memseg Usr 0000H C000H Bank 1

Memseg Usr 0000H C000H Bank 2

Memseg Usr 0000H 7500H Bank 0

MP/M II V2.0

Copyright (C) 1981, Digital Research

0A>

## Выполнение MPMLDR

Два параметра могут быть определены в MPMLDR. Первый параметр используется, чтобы вызвать повреждение к отладчику CP/M после того, как загрузка будет завершена. Параметр является символьной строкой $Bn, помещенной в поле имени FCB файла по умолчанию, начинающееся в 005DH. Символ n является номером рестарта отладчика CP/M. Если n не введен, используется значение по умолчанию 7. Пример этого параметра показан в [Разделе 1.4](#_Отладка_XIOS).

Второй параметр может определить альтернативное имя файла для загрузки кроме стандартного файла MPM.SYS. Этот параметр задается, поместив имя файла с расширением файла SYS в FCB по умолчанию, начинающемся в 005CH, или, если параметр $Bn также определяется во втором FCB по умолчанию, начинающемся в 006CH. Хорошее применение этого второго параметра должно было бы включить управляемый с помощью меню выбор файла SYS в LDRBIOS в точке входа SELDSK. Таким образом оператору предложили бы выбрать надлежащий файл SYS для его среды MP/M. Пользовательский код в точке входа SELDSK предложил бы оператору имя файла и затем поместил бы выбранное имя файла SYS в FCB по умолчанию, начинающийся в 005CH.

Приложение 1 Макросы определения дисков

; MP/M II V2.0 disk re-definition library

;

; Copyright (c) 1979, 1980, 1981

; Digital Research

; Box 579

; Pacific Grove, CA

; 93950

;

; MP/M II logical disk drives are defined using the

; macros given below, where the sequence of calls

; is:

;

; disks n

; diskdef parameter-list-0

; diskdef parameter-list-1

; ...

; diskdef parameter-list-n

; endef

;

; where n is the number of logical disk drives attached

; to the MP/M II system, and parameter-list-i defines the

; characteristics of the ith drive (i=0,1,...,n-1)

;

; each parameter-list-i takes the form

; dn,fsc,lsc,[skf],bls,dks,dir,cks,ofs,[k16],[prm]

; where

; dn is the disk number 0,1,...,n-1

; fsc is the first sector number (usually 0 or 1)

; lsc is the last sector number on a track

; skf is optional "skew factor" for sector translate

; bls is the data block size (1024,2048,...,16384)

; dks is the disk size in bls increments (word)

; dir is the number of directory elements (word)

; cks is the number of dir elements to checksum

; ofs is the number of tracks to skip (word)

; k16 is an optional 0 which forces 16K/directory entry

; prm is an optional 0 which marks drive as permanent

;

; for convenience, the form

; dn,dm

; defines disk dn as having the same characteristics as

; a previously defined disk dm.

;

; a standard four drive MP/M II system is defined by

; disks 4

; diskdef 0,1,26,6,1024,243,64,64,2

; dsk set 0

; rept 3

; dsk set dsk+1

; diskdef %dsk,0

; endm

; endef

;

; the value of "begdat" at the end of assembly defines the

; beginning of the uninitialize ram area above the bios,

; while the value of "enddat" defines the next location

; following the end of the data area. the size of this

; area is given by the value of "datsiz" at the end of the

; assembly. note that the allocation vector will be quite

; large if a large disk size is defined with a small block

; size.

;

dskhdr macro dn

;; define a single disk header list

dpe&dn: dw xlt&dn,0000h ;translate table

dw 0000h,0000h ;scratch area

dw dirbuf,dpb&dn ;dir buff,parm block

dw csv&dn,alv&dn ;check, alloc vectors

endm

;

disks macro nd

;; define nd disks

ndisks set nd ;;for later reference

dpbase equ $ ;base of disk parameter blocks

;; generate the nd elements

dsknxt set 0

rept nd

dskhdr %dsknxt

dsknxt set dsknxt+1

endm

endm

;

dpbhdr macro dn

dpb&dn equ $ ;disk parm block

endm

;

ddb macro data,comment

;; define a db statement

db data comment

endm

;

ddw macro data,comment

;; define a dw statement

dw data comment

endm

;

gcd macro m,n

;; greatest common divisor of m,n

;; produces value gcdn as result

;; (used in sector translate table generation)

gcdm set m ;;variable for m

gcdn set n ;;variable for n

gcdr set 0 ;;variable for r

rept 65535

gcdx set gcdm/gcdn

gcdr set gcdm - gcdx\*gcdn

if gcdr = 0

exitm

endif

gcdm set gcdn

gcdn set gcdr

endm

endm

;

diskdef macro dn,fsc,lsc,skf,bls,dks,dir,cks,ofs,k16,prm

;; generate the set statements for later tables

if nul lsc

;; current disk dn same as previous fsc

dpb&dn equ dpb&fsc ;equivalent parameters

als&dn equ als&fsc ;same allocation vector size

css&dn equ css&fsc ;same checksum vector size

xlt&dn equ xlt&fsc ;same translate table

else

cksz set (cks)/4

secmax set lsc-(fsc) ;;sectors 0...secmax

sectors set secmax+1;;number of sectors

als&dn set (dks)/8 ;;size of allocation vector

if ((dks) mod 8) ne 0

als&dn set als&dn+1

endif

css&dn set cksz ;;number of checksum elements

;; generate the block shift value

blkval set bls/128 ;;number of sectors/block

blkshf set 0 ;;counts right 0's in blkval

blkmsk set 0 ;;fills with 1's from right

rept 16 ;;once for each bit position

if blkval=1

exitm

endif

;; otherwise, high order 1 not found yet

blkshf set blkshf+1

blkmsk set (blkmsk shl 1) or 1

blkval set blkval/2

endm

;; generate the extent mask byte

blkval set bls/1024 ;;number of kilobytes/block

extmsk set 0 ;;fill from right with 1's

rept 16

if blkval=1

exitm

endif

;; otherwise more to shift

extmsk set (extmsk shl 1) or 1

blkval set blkval/2

endm

;; may be double byte allocation

if (dks) > 256

extmsk set (extmsk shr 1)

endif

;; may be optional [0] in last position

if not nul k16

extmsk set k16

endif

;; now generate directory reservation bit vector

dirrem set dir ;;# remaining to process

dirbks set bls/32 ;;number of entries per block

dirblk set 0 ;;fill with 1's on each loop

rept 16

if dirrem=0

exitm

endif

;; not complete, iterate once again

;; shift right and add 1 high order bit

dirblk set (dirblk shr 1) or 8000h

if dirrem > dirbks

dirrem set dirrem-dirbks

else

dirrem set 0

endif

endm

dpbhdr dn ;;generate equ $

ddw %sectors,<;sec per track>

ddb %blkshf,<;block shift>

ddb %blkmsk,<;block mask>

ddb %extmsk,<;extnt mask>

ddw %(dks)-1,<;disk size-1>

ddw %(dir)-1,<;directory max>

ddb %dirblk shr 8,<;alloc0>

ddb %dirblk and 0ffh,<;alloc1>

if nul prm

ddw %(cks)/4,<;check size>

else

ddw 8000h+cksz,<;permanent disk with check size>

endif

ddw %ofs,<;offset>

;; generate the translate table, if requested

if nul skf

xlt&dn equ 0 ;no xlate table

else

if skf = 0

xlt&dn equ 0 ;no xlate table

else

;; generate the translate table

nxtsec set 0 ;;next sector to fill

nxtbas set 0 ;;moves by one on overflow

gcd %sectors,skf

;; gcdn = gcd(sectors,skew)

neltst set sectors/gcdn

;; neltst is number of elements to generate

;; before we overlap previous elements

nelts set neltst ;;counter

xlt&dn equ $ ;translate table

rept sectors ;;once for each sector

if sectors < 256

ddb %nxtsec+(fsc)

else

ddw %nxtsec+(fsc)

endif

nxtsec set nxtsec+(skf)

if nxtsec >= sectors

nxtsec set nxtsec-sectors

endif

nelts set nelts-1

if nelts = 0

nxtbas set nxtbas+1

nxtsec set nxtbas

nelts set neltst

endif

endm

endif ;;end of nul fac test

endif ;;end of nul bls test

endm

;

defds macro lab,space

lab: ds space

endm

;

lds macro lb,dn,val

defds lb&dn,%val&dn

endm

;

endef macro

;; generate the necessary ram data areas

begdat equ $

dirbuf: ds 128 ;directory access buffer

dsknxt set 0

rept ndisks ;;once for each disk

lds alv,%dsknxt,als

lds csv,%dsknxt,css

dsknxt set dsknxt+1

endm

enddat equ $

datsiz equ $-begdat

force: db 0 ;force out last byte in hex file

endm

Приложение 2 Алгоритмы разблокирования сектора для MP/M II

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

;\* \*

;\* Sector Deblocking Algorithms for MP/M II V2.0 \*

;\* \*

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

;

; utility macro to compute sector mask

smask macro hblk

;; compute log2(hblk), return @x as result

;; (2 \*\* @x = hblk on return)

@y set hblk

@x set 0

;; count right shifts of @y until = 1

rept 8

if @y = 1

exitm

endif

;; @y is not 1, shift right one position

@y set @y shr 1

@x set @x + 1

endm

endm

;

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

;\* \*

;\* MP/M to host disk constants \*

;\* \*

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

0800 = blksiz equ 2048 ;MP/M allocation size

0200 = hstsiz equ 512 ;host disk sector size

0014 = hstspt equ 20 ;host disk sectors/trk

0004 = hstblk equ hstsiz/128 ;MP/M sects/host buff

0050 = cpmspt equ hstblk \* hstspt ;MP/M sectors/track

0003 = secmsk equ hstblk-1 ;sector mask

smask hstblk ;compute sector mask

0002 = secshf equ @x ;log2(hstblk)

;

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

;\* \*

;\* BDOS constants on entry to write \*

;\* \*

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

0000 = wrall equ 0 ;write to allocated

0001 = wrdir equ 1 ;write to directory

0002 = wrual equ 2 ;write to unallocated

;

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

;\* \*

;\* The BDOS entry points given below show the \*

;\* code which is relevant to deblocking only. \*

;\* \*

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

;

; DISKDEF macro, or hand coded tables go here

0000 = dpbase equ $ ;disk param block base

;

boot:

wboot:

;enter here on system boot to initialize

0000 AF xra a ;0 to accumulator

0001 326901 sta hstact ;host buffer inactive

0004 326B01 sta unacnt ;clear unalloc count

0007 C9 ret

;

home:

;home the selected disk

0008 3A6A01 lda hstwrt ;check for pending write

000B B7 ora a

000C C21200 jnz homed

000F 326901 sta hstact ;clear host active flag

homed:

0012 C9 ret

;

seldsk:

;select disk

0013 79 mov a,c ;selected disk number

0014 326001 sta sekdsk ;seek disk number

0017 6F mov l,a ;disk number to HL

0018 2600 mvi h,0

rept 4 ;multiply by 16

dad h

endm

001A+29 DAD H

001B+29 DAD H

001C+29 DAD H

001D+29 DAD H

001E 110000 lxi d,dpbase ;base of parm block

0021 19 dad d ;hl=.dpb(curdsk)

0022 C9 ret

;

settrk:

;set track given by registers BC

0023 60 mov h,b

0024 69 mov l,c

0025 226101 shld sektrk ;track to seek

0028 C9 ret

;

setsec:

;set sector given by register c

0029 79 mov a,c

002A 326301 sta seksec ;sector to seek

002D C9 ret

;

setdma:

;set dma address given by BC

002E 60 mov h,b

002F 69 mov l,c

0030 227401 shld dmaadr

0033 C9 ret

;

sectran:

;translate sector number BC

0034 60 mov h,b

0035 69 mov l,c

0036 C9 ret

;

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

;\* \*

;\* The READ entry point takes the place of \*

;\* the previous BIOS defintion for READ. \*

;\* \*

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

read:

;read the selected MP/M sector

0037 AF xra a

0038 326B01 sta unacnt ;unacnt = 0

003B 3C inr a

003C 327201 sta readop ;read operation

003F 327101 sta rsflag ;must read data

0042 3E02 mvi a,wrual

0044 327301 sta wrtype ;treat as unalloc

0047 C3B500 jmp rwoper ;to perform the read

;

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

;\* \*

;\* The WRITE entry point takes the place of \*

;\* the previous BIOS defintion for WRITE. \*

;\* \*

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

write:

;write the selected MP/M sector

004A AF xra a ;0 to accumulator

004B 327201 sta readop ;not a read operation

004E 79 mov a,c ;write type in c

004F 327301 sta wrtype

0052 E602 ani wrual ;write unallocated?

0054 CA6E00 jz chkuna ;check for unalloc

;

; write to unallocated, set parameters

0057 3E10 mvi a,blksiz/128 ;next unalloc recs

0059 326B01 sta unacnt

005C 3A6001 lda sekdsk ;disk to seek

005F 326C01 sta unadsk ;unadsk = sekdsk

0062 2A6101 lhld sektrk

0065 226D01 shld unatrk ;unatrk = sectrk

0068 3A6301 lda seksec

006B 326F01 sta unasec ;unasec = seksec

;

chkuna:

;check for write to unallocated sector

006E 3A6B01 lda unacnt ;any unalloc remain?

0071 B7 ora a

0072 CAAD00 jz alloc ;skip if not

;

; more unallocated records remain

0075 3D dcr a ;unacnt = unacnt-1

0076 326B01 sta unacnt

0079 3A6001 lda sekdsk ;same disk?

007C 216C01 lxi h,unadsk

007F BE cmp m ;sekdsk = unadsk?

0080 C2AD00 jnz alloc ;skip if not

;

; disks are the same

0083 216D01 lxi h,unatrk

0086 CD5201 call sektrkcmp ;sektrk = unatrk?

0089 C2AD00 jnz alloc ;skip if not

;

; tracks are the same

008C 3A6301 lda seksec ;same sector?

008F 216F01 lxi h,unasec

0092 BE cmp m ;seksec = unasec?

0093 C2AD00 jnz alloc ;skip if not

;

; match, move to next sector for future ref

0096 34 inr m ;unasec = unasec+1

0097 7E mov a,m ;end of track?

0098 FE50 cpi cpmspt ;count MP/M sectors

009A DAA600 jc noovf ;skip if no overflow

;

; overflow to next track

009D 3600 mvi m,0 ;unasec = 0

009F 2A6D01 lhld unatrk

00A2 23 inx h

00A3 226D01 shld unatrk ;unatrk = unatrk+1

;

noovf:

;match found, mark as unnecessary read

00A6 AF xra a ;0 to accumulator

00A7 327101 sta rsflag ;rsflag = 0

00AA C3B500 jmp rwoper ;to perform the write

;

alloc:

;not an unallocated record, requires pre-read

00AD AF xra a ;0 to accum

00AE 326B01 sta unacnt ;unacnt = 0

00B1 3C inr a ;1 to accum

00B2 327101 sta rsflag ;rsflag = 1

;

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

;\* \*

;\* Common code for READ and WRITE follows \*

;\* \*

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

rwoper:

;enter here to perform the read/write

00B5 AF xra a ;zero to accum

00B6 327001 sta erflag ;no errors (yet)

00B9 3A6301 lda seksec ;compute host sector

rept secshf

ora a ;carry = 0

rar ;shift right

endm

00BC+B7 ORA A ;CARRY = 0

00BD+1F RAR ;SHIFT RIGHT

00BE+B7 ORA A ;CARRY = 0

00BF+1F RAR ;SHIFT RIGHT

00C0 326801 sta sekhst ;host sector to seek

;

; active host sector?

00C3 216901 lxi h,hstact ;host active flag

00C6 7E mov a,m

00C7 3601 mvi m,1 ;always becomes 1

00C9 B7 ora a ;was it already?

00CA CAF100 jz filhst ;fill host if not

;

; host buffer active, same as seek buffer?

00CD 3A6001 lda sekdsk

00D0 216401 lxi h,hstdsk ;same disk?

00D3 BE cmp m ;sekdsk = hstdsk?

00D4 C2EA00 jnz nomatch

;

; same disk, same track?

00D7 216501 lxi h,hsttrk

00DA CD5201 call sektrkcmp ;sektrk = hsttrk?

00DD C2EA00 jnz nomatch

;

; same disk, same track, same buffer?

00E0 3A6801 lda sekhst

00E3 216701 lxi h,hstsec ;sekhst = hstsec?

00E6 BE cmp m

00E7 CA0E01 jz match ;skip if match

;

nomatch:

;proper disk, but not correct sector

00EA 3A6A01 lda hstwrt ;host written?

00ED B7 ora a

00EE C45E01 cnz writehst ;clear host buff

;

filhst:

;may have to fill the host buffer

00F1 3A6001 lda sekdsk

00F4 326401 sta hstdsk

00F7 2A6101 lhld sektrk

00FA 226501 shld hsttrk

00FD 3A6801 lda sekhst

0100 326701 sta hstsec

0103 3A7101 lda rsflag ;need to read?

0106 B7 ora a

0107 C45F01 cnz readhst ;yes, if 1

010A AF xra a ;0 to accum

010B 326A01 sta hstwrt ;no pending write

;

match:

;copy data to or from buffer

010E 3A6301 lda seksec ;mask buffer number

0111 E603 ani secmsk ;least signif bits

0113 6F mov l,a ;ready to shift

0114 2600 mvi h,0 ;double count

rept 7 ;shift left 7

dad h

endm

0116+29 DAD H

0117+29 DAD H

0118+29 DAD H

0119+29 DAD H

011A+29 DAD H

011B+29 DAD H

011C+29 DAD H

; hl has relative host buffer address

011D 117601 lxi d,hstbuf

0120 19 dad d ;hl = host address

0121 EB xchg ;now in DE

0122 2A7401 lhld dmaadr ;get/put MP/M data

0125 0E80 mvi c,128 ;length of move

0127 3A7201 lda readop ;which way?

012A B7 ora a

012B C23401 jnz rwmove ;skip if read

;

; write operation, mark and switch direction

012E 3E01 mvi a,1

0130 326A01 sta hstwrt ;hstwrt = 1

0133 EB xchg ;source/dest swap

;

rwmove:

;C initially 128, DE is source, HL is dest

0134 1A ldax d ;source character

0135 13 inx d

0136 77 mov m,a ;to dest

0137 23 inx h

0138 0D dcr c ;loop 128 times

0139 C23401 jnz rwmove

;

; data has been moved to/from host buffer

013C 3A7301 lda wrtype ;write type

013F E601 ani wrdir ;to directory?

0141 3A7001 lda erflag ;in case of errors

0144 C8 rz ;no further processing

;

; clear host buffer for directory write

0145 B7 ora a ;errors?

0146 C0 rnz ;skip if so

0147 AF xra a ;0 to accum

0148 326A01 sta hstwrt ;buffer written

014B CD5E01 call writehst

014E 3A7001 lda erflag

0151 C9 ret

;

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

;\* \*

;\* Utility subroutine for 16-bit compare \*

;\* \*

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

sektrkcmp:

;HL = .unatrk or .hsttrk, compare with sektrk

0152 EB xchg

0153 216101 lxi h,sektrk

0156 1A ldax d ;low byte compare

0157 BE cmp m ;same?

0158 C0 rnz ;return if not

; low bytes equal, test high 1s

0159 13 inx d

015A 23 inx h

015B 1A ldax d

015C BE cmp m ;sets flags

015D C9 ret

;

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

;\* \*

;\* WRITEHST performs the physical write to \*

;\* the host disk, READHST reads the physical \*

;\* disk. \*

;\* \*

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

writehst:

;hstdsk = host disk #, hsttrk = host track #,

;hstsec = host sect #. write "hstsiz" bytes

;from hstbuf and return error flag in erflag.

;return erflag non-zero if error

015E C9 ret

;

readhst:

;hstdsk = host disk #, hsttrk = host track #,

;hstsec = host sect #. read "hstsiz" bytes

;into hstbuf and return error flag in erflag.

015F C9 ret

;

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

;\* \*

;\* Unitialized RAM data areas \*

;\* \*

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

;

0160 sekdsk: ds 1 ;seek disk number

0161 sektrk: ds 2 ;seek track number

0163 seksec: ds 1 ;seek sector number

;

0164 hstdsk: ds 1 ;host disk number

0165 hsttrk: ds 2 ;host track number

0167 hstsec: ds 1 ;host sector number

;

0168 sekhst: ds 1 ;seek shr secshf

0169 hstact: ds 1 ;host active flag

016A hstwrt: ds 1 ;host written flag

;

016B unacnt: ds 1 ;unalloc rec cnt

016C unadsk: ds 1 ;last unalloc disk

016D unatrk: ds 2 ;last unalloc track

016F unasec: ds 1 ;last unalloc sector

;

0170 erflag: ds 1 ;error reporting

0171 rsflag: ds 1 ;read sector flag

0172 readop: ds 1 ;1 if read operation

0173 wrtype: ds 1 ;write operation type

0174 dmaadr: ds 2 ;last dma address

0176 hstbuf: ds hstsiz ;host buffer

;

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

;\* \*

;\* The ENDEF macro invocation goes here \*

;\* \*

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

0376 end

00AD ALLOC 0800 BLKSIZ 0000 BOOT 006E CHKUNA

0050 CPMSPT 0174 DMAADR 0000 DPBASE 0170 ERFLAG

00F1 FILHST 0008 HOME 0012 HOMED 0169 HSTACT

0004 HSTBLK 0176 HSTBUF 0164 HSTDSK 0167 HSTSEC

0200 HSTSIZ 0014 HSTSPT 0165 HSTTRK 016A HSTWRT

010E MATCH 00EA NOMATCH 00A6 NOOVF 0037 READ

015F READHST 0172 READOP 0171 RSFLAG 0134 RWMOVE

00B5 RWOPER 0003 SECMSK 0002 SECSHF 0034 SECTRAN

0160 SEKDSK 0168 SEKHST 0163 SEKSEC 0161 SEKTRK

0152 SEKTRKCMP 0013 SELDSK 002E SETDMA 0029 SETSEC

0023 SETTRK 016B UNACNT 016C UNADSK 016F UNASEC

016D UNATRK 0000 WBOOT 0000 WRALL 0001 WRDIR

004A WRITE 015E WRITEHST 0173 WRTYPE 0002 WRUAL

Приложение 3 Пример загрузчика MP/M II BIOS

title 'MP/M II V2.0 Skeleton Ldrbios'

; Copyright (C) 1978, 1979, 1980, 1981

; Digital Research

; Box 579, Pacific Grove

; California, 93950

0000 = false equ 0

FFFF = true equ not false

1700 org 1700h

0080 = buff equ 0080h ;default buffer address

; jump vector for individual routines

1700 C33317 jmp boot

1703 C33317 wboote: jmp wboot

1706 C33617 jmp const

1709 C33417 jmp conin

170C C33517 jmp conout

170F C33917 jmp list

1712 C33817 jmp punch

1715 C33717 jmp reader

1718 C33C17 jmp home

171B C33B17 jmp seldsk

171E C33D17 jmp settrk

1721 C33E17 jmp setsec

1724 C33F17 jmp setdma

1727 C34117 jmp read

172A C34217 jmp write

172D C33A17 jmp list$st ; list status poll

1730 C34017 jmp sect$tran ; sector translation

boot:

wboot:

gocpm:

1733 C9 ret

crtin: ; crt: input

1734 C9 ret

crtout: ; crt: output

1735 C9 ret

crtst: ; crt: status

1736 C9 ret

ttyin: ; tty: input

1737 C9 ret

ttyout: ; tty: output

1738 C9 ret

lptout: ; lpt: output

1739 C9 ret

lpt$st:

173A C9 ret

1734 = conin equ crtin

1736 = const equ crtst

1735 = conout equ crtout

1737 = reader equ ttyin

1738 = punch equ ttyout

1739 = list equ lptout

173A = listst equ lptst

seldsk: ;select disk given by register c

173B C9 ret

;

home: ;move to home position

173C C9 ret

;

settrk: ;set track number given by c

173D C9 ret

;

setsec: ;set sector number given by c

173E C9 ret

;

setdma: ;set dma address given by regs b,c

173F C9 ret

;

sect$tran: ; translate the sector # in <c> if needed

1740 C9 ret

;

read: ;read next disk record

; (assuming disk/trk/sec/dma set)

1741 C9 ret

;

write: ;disk write function

1742 C9 ret

;

1743 end

Приложение 4 Листинг источника простого XIOS

title 'MP/M II V2.0 DSC-2 Basic & Extended I/O Systems'

cseg

maclib diskdef

;

; bios for micro-2 computer

;

;

0000 = false equ 0

FFFF = true equ not false

;

FFFF = debug equ true

FFFF = ldcmd equ true

;

FFFF = MHz4 equ true

if MHz4

0086 = dlycnst equ 086h

else

dlycnst equ 054h

endif

;

; org 0000h

;

; jump vector for individual subroutines

; jmp coldstart ;cold start

0000 C34900 jmp commonbase

wboot:

0003 C35A00 jmp warmstart ;warm start

0006 C35F00 jmp const ;console status

0009 C36800 jmp conin ;console character in

000C C37100 jmp conout ;console character out

000F C3DF00 jmp list ;list character out

0012 C38100 jmp rtnempty ;punch not implemented

0015 C38100 jmp rtnempty ;reader not implemented

0018 C3CA02 jmp home ;move head to home

001B C3DB02 jmp seldsk ;select disk

001E C30503 jmp settrk ;set track number

0021 C32203 jmp setsec ;set sector number

0024 C33A03 jmp setdma ;set dma address

0027 C34003 jmp read ;read disk

002A C34503 jmp write ;write disk

002D C30101 jmp pollpt ;list status

0030 C32803 jmp sectran ;sector translate

0033 C30C02 jmp selmemory ; select memory

0036 C3F301 jmp polldevice ; poll device

0039 C30D02 jmp startclock ; start clock

003C C31302 jmp stopclock ; stop clock

003F C31802 jmp exitregion ; exit region

0042 C31F02 jmp maxconsole ; maximum console number

0045 C32202 jmp systeminit ; system initialization

0048 00 db 0 ; force use of internal dispatch @ idle

; jmp idle ; idle procedure

;

commonbase:

0049 C35A00 jmp coldstart

004C C30000 swtuser: jmp $-$

004F C30000 swtsys: jmp $-$

0052 C30000 pdisp: jmp $-$

0055 C30000 xdos: jmp $-$

0058 0000 sysdat: dw $-$

coldstart:

warmstart:

005A 0E00 mvi c,0

005C C35500 jmp xdos ; system reset, terminate process

;

;

;I/O handlers

;

;

; MP/M II V2.0 Console Bios

;

;

0003 = nmbcns equ 3 ; number of consoles

0083 = poll equ 131 ; XDOS poll function

0086 = makeque equ 134 ; XDOS make queue function

0089 = readque equ 137 ; XDOS read queue function

008B = writeque equ 139 ; XDOS write queue function

008D = xdelay equ 141 ; XDOS delay function

0090 = create equ 144 ; XDOS create process function

0000 = pllpt equ 0 ; poll printer

0001 = plco0 equ 1 ; poll console out #0

0002 = plco2 equ 2 ; poll console out #1

0003 = plco3 equ 3 ; poll console out #2 (Port 3)

0004 = plci3 equ 4 ; poll console in #2 (Port 3)

if debug

0005 = plci0 equ 5 ; poll console in #0

endif

;

const: ; Console Status

005F CD7A00 call ptbljmp ; compute and jump to hndlr

0062 8E00 dw pt0st ; console #0 status routine

0064 0901 dw pt2st ; console #1 (Port 2) status rt

0066 C301 dw pt3st ; console #2 (Port 3) status rt

conin: ; Console Input

0068 CD7A00 call ptbljmp ; compute and jump to hndlr

006B 9D00 dw pt0in ; console #0 input

006D 9901 dw pt2in ; console #1 (Port 2) input

006F CB01 dw pt3in ; console #2 (Port 3) input

conout: ; Console Output

0071 CD7A00 call ptbljmp ; compute and jump to hndlr

0074 C200 dw pt0out ; console #0 output

0076 A701 dw pt2out ; console #1 (Port 2) output

0078 D701 dw pt3out ; console #2 (Port 3) output

;

ptbljmp: ; compute and jump to handler

; d = console #

; do not destroy d !

007A 7A mov a,d

007B FE03 cpi nmbcns

007D DA8300 jc tbljmp

0080 F1 pop psw ; throw away table address

rtnempty:

0081 AF xra a

0082 C9 ret

tbljmp: ; compute and jump to handler

; a = table index

0083 87 add a ; double table index for adr offst

0084 E1 pop h ; return adr points to jump tbl

0085 5F mov e,a

0086 1600 mvi d,0

0088 19 dad d ; add table index \* 2 to tbl base

0089 5E mov e,m ; get handler address

008A 23 inx h

008B 56 mov d,m

008C EB xchg

008D E9 pchl ; jump to computed cns handler

;

; ASCII Character Equates

;

005F = uline equ 5fh

007F = rubout equ 7fh

0020 = space equ 20h

0008 = backsp equ 8h

005F = altrub equ uline

;

; Input / Output Port Address Equates

;

0040 = data0 equ 40h

0041 = sts0 equ data0+1

0041 = cd0 equ sts0

0048 = data1 equ 48h

0049 = sts1 equ data1+1

0049 = cd1 equ sts1

0050 = data2 equ 50h

0051 = sts2 equ data2+1

0051 = cd2 equ sts2

0058 = data3 equ 58h

0059 = sts3 equ data3+1

0059 = cd3 equ sts3

;

; Poll Console #0 Input

;

if debug

polci0:

pt0st:

if ldcmd

008E 3AAF00 lda pt0cntr

0091 B7 ora a

0092 3E00 mvi a,0

0094 C0 rnz

endif

0095 DB41 in sts0

0097 E602 ani 2

0099 C8 rz

009A 3EFF mvi a,0ffh

009C C9 ret

;

pt0in:

if ldcmd

009D 21AF00 lxi h,pt0cntr

00A0 7E mov a,m

00A1 B7 ora a

00A2 CAB600 jz ldcmd0empty

00A5 35 dcr m

00A6 2AB000 lhld pt0ptr

00A9 7E mov a,m

00AA 23 inx h

00AB 22B000 shld pt0ptr

00AE C9 ret

pt0cntr:

00AF 04 db ldcmd0empty-pt0ldcmd

pt0ptr:

00B0 B200 dw pt0ldcmd

pt0ldcmd:

00B2 746F6420 db 'tod '

ldcmd0empty:

endif

00B6 0E83 mvi c,poll

00B8 1E05 mvi e,plci0

00BA CD5500 call xdos

00BD DB40 in data0

00BF E67F ani 7fh

00C1 C9 ret

;

else

pt0st:

; return 0ffh if ready,

; 000h if not

lda c0inmsgcnt

ora a

rz

mvi a,0ffh

ret

;

; Console #0 Input

;

c0inpd:

dw c2inpd ; pl

db 0 ; status

db 32 ; priority

dw c0instk+18 ; stkptr

db 'c0in ' ; name

db 0 ; console

db 0ffh ; memseg

ds 36

c0instk:

dw 0c7c7h,0c7c7h,0c7c7h

dw 0c7c7h,0c7c7h,0c7c7h

dw 0c7c7h,0c7c7h,0c7c7h

dw c0inp ; starting address

c0inq:

dw 0 ; ql

db 'c0inque ' ; name

dw 1 ; msglen

dw 4 ; nmbmsgs

ds 8

c0inmsgcnt:

ds 2 ; msgcnt

ds 4 ; buffer

c0inqcb:

dw c0inq ; pointer

dw ch0in ; msgadr

ch0in:

db 0

c0inuqcb:

dw c0inq ; pointer

dw char0in ; msgadr

char0in:

db 0

c0inp:

mvi c,makeque

lxi d,c0inq

call xdos ; make the c0inq

c0inloop:

mvi c,flagwait

mvi e,6

call xdos ; wait for c0 in intr flag

mvi c,writeque

lxi d,c0inqcb

call xdos ; write c0in queue

jmp c0inloop

pt0in:

; return character in reg A

mvi c,readque

lxi d,c0inuqcb

call xdos ; read from c0 in queue

lda char0in ; get character

ani 7fh ; strip parity bit

ret

;

endif

;

; Console #0 Output

;

pt0out:

; Reg C = character to output

00C2 DB41 in sts0

00C4 E601 ani 01h

00C6 C2D200 jnz tx0rdy

00C9 C5 push b

00CA 0E83 mvi c,poll

00CC 1E01 mvi e,plco0

00CE CD5500 call xdos ; poll console #0 output

00D1 C1 pop b

tx0rdy:

00D2 79 mov a,c

00D3 D340 out data0

00D5 C9 ret

;

; poll console #0 output

;

polco0:

00D6 DB41 in sts0

00D8 E601 ani 01h

00DA C8 rz

00DB 3EFF mvi a,0ffh

00DD C9 ret

;

;

; Line Printer Driver: TI 810 Serial Printer

; TTY Model 40

;

initflag:

00DE 00 db 0 ; printer initialization flag

list: ; List Output

pt1out:

; Reg c = Character to print

00DF 3ADE00 lda initflag

00E2 B7 ora a

00E3 C2ED00 jnz pt1xx

00E6 3E27 mvi a,27h

00E8 D349 out 49h ; TTY Model 40 init

00EA 32DE00 sta initflag

pt1xx:

00ED DB49 in sts1

00EF E601 ani 01h

00F1 C2FD00 jnz tx1rdy

00F4 C5 push b

00F5 0E83 mvi c,poll

00F7 1E00 mvi e,pllpt

00F9 CD5500 call xdos ; poll printer output

00FC C1 pop b

tx1rdy:

00FD 79 mov a,c ; char to register a

00FE D348 out data1

0100 C9 ret

;

; Poll Printer Output

;

pollpt:

; return 0ffh if ready,

; 000h if not

0101 DB49 in sts1

0103 E601 ani 01h

0105 C8 rz

0106 3EFF mvi a,0ffh

0108 C9 ret

;

; Poll Console #1 (Port 2) Input

;

pt2st:

; return 0ffh if ready,

; 000h if not

0109 3A6F01 lda c2inmsgcnt

010C B7 ora a

010D C8 rz

010E 3EFF mvi a,0ffh

0110 C9 ret

;

; Console #1 (Port 2) Input

;

c2inpd:

0111 0000 dw 0 ; pl

0113 00 db 0 ; status

0114 22 db 34 ; priority

0115 5701 dw c2instk+18 ; stkptr

0117 6332696E20 db 'c2in ' ; name

011F 02 db 2 ; console

0120 FF db 0ffh ; memseg

0121 ds 36

c2instk:

0145 C7C7C7C7C7 dw 0c7c7h,0c7c7h,0c7c7h

014B C7C7C7C7C7 dw 0c7c7h,0c7c7h,0c7c7h

0151 C7C7C7C7C7 dw 0c7c7h,0c7c7h,0c7c7h

0157 7F01 dw c2inp ; starting address

c2inq:

0159 0000 dw 0 ; ql

015B 6332696E71 db 'c2inque ' ; name

0163 0100 dw 1 ; msglen

0165 0400 dw 4 ; nmbmsgs

0167 ds 8

c2inmsgcnt:

016F ds 2 ; msgcnt

0171 ds 4 ; buffer

c2inqcb:

0175 5901 dw c2inq ; pointer

0177 7901 dw ch2in ; msgadr

ch2in:

0179 00 db 0

c2inuqcb:

017A 5901 dw c2inq ; pointer

017C 7E01 dw char2in ; msgadr

char2in:

017E 00 db 0

c2inp:

017F 0E86 mvi c,makeque

0181 115901 lxi d,c2inq

0184 CD5500 call xdos ; make the c2inq

c2inloop:

0187 0E84 mvi c,flagwait

0189 1E08 mvi e,8

018B CD5500 call xdos ; wait for c2 in intr flag

018E 0E8B mvi c,writeque

0190 117501 lxi d,c2inqcb

0193 CD5500 call xdos ; write c2in queue

0196 C38701 jmp c2inloop

pt2in:

; return character in reg A

0199 0E89 mvi c,readque

019B 117A01 lxi d,c2inuqcb

019E CD5500 call xdos ; read from c2 in queue

01A1 3A7E01 lda char2in ; get character

01A4 E67F ani 7fh ; strip parity bit

01A6 C9 ret

;

; Console #1 (Port 2) Output

;

pt2out:

; Reg C = character to output

01A7 DB51 in sts2

01A9 E601 ani 01h

01AB C2B701 jnz tx2rdy

01AE C5 push b

01AF 0E83 mvi c,poll

01B1 1E02 mvi e,plco2

01B3 CD5500 call xdos ; poll console #1 output

01B6 C1 pop b

tx2rdy:

01B7 79 mov a,c

01B8 D350 out data2

01BA C9 ret

;

; poll console #1 output

;

polco2:

01BB DB51 in sts2

01BD E601 ani 01h

01BF C8 rz

01C0 3EFF mvi a,0ffh

01C2 C9 ret

;

; Poll Console #2 (Port 3) Input

;

polci3:

pt3st: ; return 0ffh if ready,

; 000h if not

01C3 DB59 in sts3

01C5 E602 ani 2

01C7 C8 rz

01C8 3EFF mvi a,0ffh

01CA C9 ret

;

; Console #2 (Port 3) Input

;

pt3in: ; return character in reg A

01CB 0E83 mvi c,poll

01CD 1E04 mvi e,plci3

01CF CD5500 call xdos ; poll console #0 input

01D2 DB58 in data3 ; read character

01D4 E67F ani 7fh ; strip parity bit

01D6 C9 ret

;

; Console #2 (Port 3) Output

;

pt3out: ; Reg C = character to output

01D7 DB59 in sts3

01D9 E601 ani 01h

01DB C2E701 jnz tx3rdy

01DE C5 push b

01DF 0E83 mvi c,poll

01E1 1E03 mvi e,plco3

01E3 CD5500 call xdos ; poll console #2 (Port 3) output

01E6 C1 pop b

tx3rdy:

01E7 79 mov a,c

01E8 D358 out data3 ; transmit character

01EA C9 ret

;

; Poll Console #2 (Port 3) Output

;

polco3:

; return 0ffh if ready,

; 000h if not

01EB DB59 in sts3

01ED E601 ani 01h

01EF C8 rz

01F0 3EFF mvi a,0ffh

01F2 C9 ret

;

;

; MP/M II V2.0 Xios

;

polldevice:

; Reg C = device # to be polled

; return 0ffh if ready,

; 000h if not

01F3 79 mov a,c

01F4 FE06 cpi nmbdev

01F6 DAFB01 jc devok

01F9 3E06 mvi a,nmbdev; if dev # >= nmbdev,

; set to nmbdev

devok:

01FB CD8300 call tbljmp ; jump to dev poll code

devtbl:

01FE 0101 dw pollpt ; poll printer output

0200 D600 dw polco0 ; poll console #0 output

0202 BB01 dw polco2 ; poll console #1 output

0204 EB01 dw polco3 ; poll console #2 output

0206 C301 dw polci3 ; poll console #2 input

if debug

0208 8E00 dw polci0 ; poll console #0 input

endif

0006 = nmbdev equ ($-devtbl)/2 ; number of devices to poll

020A 8100 dw rtnempty; bad device handler

;

; Select / Protect Memory

;

selmemory:

; Reg BC = adr of mem descriptor

; BC -> base 1 byte,

; size 1 byte,

; attrib 1 byte,

; bank 1 byte.

; this hardware does not have memory protection or

; bank switching

020C C9 ret

;

; Start Clock

;

startclock:

; will cause flag #1 to be set

; at each system time unit tick

020D 3EFF mvi a,0ffh

020F 322F04 sta tickn

0212 C9 ret

;

; Stop Clock

;

stopclock:

; will stop flag #1 setting at

; system time unit tick

0213 AF xra a

0214 322F04 sta tickn

0217 C9 ret

;

; Exit Region

;

exitregion:

; EI if not preempted or in dispatcher

0218 3A3104 lda preemp

021B B7 ora a

021C C0 rnz

021D FB ei

021E C9 ret

;

; Maximum Console Number

;

maxconsole:

021F 3E03 mvi a,nmbcns

0221 C9 ret

;

; System Initialization

;

systeminit:

;

; This is the place to insert code to initialize

; the time of day clock, if it is desired on each

; booting of the system.

;

0222 3EC3 mvi a,0c3h

0224 323800 sta 0038h

0227 214702 lxi h,inthnd

022A 223900 shld 0039h ; JMP INTHND at 0038H

022D 0E90 mvi c,create

if debug

022F 111101 lxi d,c2inpd

else

lxi d,c0inpd

endif

0232 CD5500 call xdos

0235 3A3004 lda intmsk

0238 D360 out 60h ; init interrupt mask

023A ED56 db 0edh,056h ; Interrupt Mode 1

; \*\* Z80 Instruction \*\*

023C FB ei

023D CDCA02 call home

0240 0E84 mvi c,flagwait

0242 1E05 mvi e,5

0244 C35500 jmp xdos ; clear first disk interrupt

; ret ; & return

;

; Idle procedure

;

;idle:

; ret

; -or-

; ei

; hlt

; ret ; for full interrupt system

;

; MP/M II V2.0 Interrupt Handlers

;

0084 = flagwait equ 132

0085 = flagset equ 133

008E = dsptch equ 142

inthnd:

; Interrupt handler entry point

; All interrupts gen a RST 7

; Location 0038H contains a jmp

; to INTHND.

0247 222904 shld svdhl

024A E1 pop h

024B 222D04 shld svdret

024E F5 push psw

024F 210000 lxi h,0

0252 39 dad sp

0253 222B04 shld svdsp ; save users stk ptr

0256 312904 lxi sp,lstintstk ; lcl stk for intr hndl

0259 D5 push d

025A C5 push b

025B 3EFF mvi a,0ffh

025D 323104 sta preemp ; set preempted flag

0260 DB60 in 60h ; read interrupt mask

0262 E640 ani 01000000b ; test & jump if clk int

0264 C28F02 jnz clk60hz

;

0267 DB80 in stat ; read disk status port

0269 E608 ani 08h

026B C27802 jnz diskintr

if not debug

in sts0

ani 2

jnz con0in

endif

026E DB51 in sts2

0270 E602 ani 2

0272 C28002 jnz con2in

; ... ; test/handle other ints

;

0275 C3B502 jmp intdone

diskintr:

0278 AF xra a

0279 D380 out cmd1 ; reset disk interrupt

027B 1E05 mvi e,5

027D C38702 jmp concmn ; set flag #5

if not debug

con0in:

in data0

sta ch0in

mvi e,6

jmp concmn ; set flag #6

endif

con2in:

0280 DB50 in data2

0282 327901 sta ch2in

0285 1E08 mvi e,8

; jmp concmn ; set flag #8

concmn:

0287 0E85 mvi c,flagset

0289 CD5500 call xdos

028C C3B502 jmp intdone

clk60hz:

; 60 Hz clock interrupt

028F 3A2F04 lda tickn

0292 B7 ora a ; test tickn, indicates

; delayed process(es)

0293 CA9D02 jz notickn

0296 0E85 mvi c,flagset

0298 1E01 mvi e,1

029A CD5500 call xdos ; set flag #1 each tick

notickn:

029D 210004 lxi h,cnt60

02A0 35 dcr m ; dec 60 tick cntr

02A1 C2AD02 jnz not1sec

02A4 363C mvi m,60

02A6 0E85 mvi c,flagset

02A8 1E02 mvi e,2

02AA CD5500 call xdos ; set flag #2 @ 1 sec

not1sec:

02AD AF xra a

02AE D360 out 60h

02B0 3A3004 lda intmsk

02B3 D360 out 60h ; ack clock interrupt

; jmp intdone

;

; ...

; Other interrupt handlers

; ...

;

intdone:

02B5 AF xra a

02B6 323104 sta preemp ; clear preempted flag

02B9 C1 pop b

02BA D1 pop d

02BB 2A2B04 lhld svdsp

02BE F9 sphl ; restore stk ptr

02BF F1 pop psw

02C0 2A2D04 lhld svdret

02C3 E5 push h

02C4 2A2904 lhld svdhl

; The following dispatch call will force round robin

; scheduling of processes executing at the same priority

; each 1/60th of a second.

; Note: Interrupts are not enabled until the dispatcher

; resumes the next process. This prevents interrupt

; over-run of the stacks when stuck or high frequency

; interrupts are encountered.

02C7 C35200 jmp pdisp ; MP/M dispatch

;

;

; Disk I/O Drivers

;

; Disk Port Equates

;

0080 = cmd1 equ 80h

0080 = stat equ 80h

0081 = haddr equ 81h

0082 = laddr equ 82h

0083 = cmd2 equ 83h

;

;

home: ;move to the track o0 position of current drive

02CA CDDA03 call headload

; h,l point to word with track for selected disk

homel:

02CD 3600 mvi m,00 ;set current track ptr back to 0

02CF DB80 in stat ;read fdc status

02D1 E604 ani 4 ;test track 0 bit

02D3 C8 rz ;return if at 0

02D4 37 stc ;direction=out

02D5 CDC203 call step ;step one track

02D8 C3CD02 jmp homel ;loop

;

seldsk:

;drive number in c

02DB 210000 lxi h,0 ;0000 in hl produces select error

02DE 79 mov a,c ;a is disk number 0 ... ndisks-1

02DF FE02 cpi ndisks ;less than ndisks?

02E1 D0 rnc ;return with HL = 0000 if not

;make sure dummy is 0 (for use in double add to h,l)

02E2 AF xra a

02E3 323A04 sta dummy

02E6 79 mov a,c

02E7 E607 ani 07h ;get only disk select bits

02E9 323904 sta diskno

02EC 4F mov c,a

;set up the second command port

02ED 3A3C04 lda port

02F0 E6F0 ani 0f0h ;clear out old disk select bits

02F2 B1 ora c ;put in new disk select bits

02F3 F608 ori 08h ; force double density

02F5 323C04 sta port

; proper disk number, return dpb element address

02F8 69 mov l,c

02F9 29 dad h ;\*2

02FA 29 dad h ;\*4

02FB 29 dad h ;\*8

02FC 29 dad h ;\*16

02FD 113F04 lxi d,dpbase

0300 19 dad d ;HL=.dpb

0301 226E04 shld tran ;translate table base

0304 C9 ret

;

;

;

settrk: ;set track given by register c

0305 CDDA03 call headload

;h,l reference correct track indicator according to

;selected disk

0308 79 mov a,c ;desired track

0309 BE cmp m

030A C8 rz ;we are already on the track

settkx:

030B CDC203 call step ;step track-carry has direction

;step will update trk indicator

030E 79 mov a,c

030F BE cmp m ;are we where we want to be

0310 C20B03 jnz settkx ;not yet

;have stepped enough

seekrt:

;need 10 msec delay for final step time and head settle time

0313 3E14 mvi a,20d

; call delay

; ret ;end of settrk routine

;

delay: ;delay for c[A] X .5 milliseconds

0315 C5 push b

delay1:

0316 0E86 mvi c,dlycnst ;constant adjusted to .5 ms loop

delay2:

0318 0D dcr c

0319 C21803 jnz delay2

031C 3D dcr a

031D C21603 jnz delay1

0320 C1 pop b

0321 C9 ret ;end of delay routine

;

setsec: ;set sector given by register c

0322 0C inr c

0323 79 mov a,c

0324 323604 sta sector

0327 C9 ret

;

sectran:

;sector number in c

;translate logical to physical sector

0328 2A6E04 lhld tran ;hl=..translate

032B 5E mov e,m ;E=low(.translate)

032C 23 inx h

032D 56 mov d,m ;DE=.translate

032E 7B mov a,e ;zero?

032F B2 ora d ;00 or 00 = 00

0330 2600 mvi h,0

0332 69 mov l,c ;HL = untranslated sector

0333 C8 rz ;skip if so

0334 EB xchg

0335 42 mov b,d ;BC=00ss

0336 09 dad b ;HL=.translate(sector)

0337 6E mov l,m

0338 62 mov h,d ;HL=translate(sector)

0339 C9 ret

;

setdma: ;set dma address given by registers b and c

033A 69 mov l,c ;low order address

033B 60 mov h,b ;high order address

033C 223704 shld dmaad ;save the address

033F C9 ret

;

;

read: ;perform read operation.

;this is similar to write, so set up read

; command and use common code in write

0340 0640 mvi b,040h ;set read flag

0342 C34703 jmp waitio ;to perform the actual I/O

;

write: ;perform a write operation

0345 0680 mvi b,080h ;set write command

;

waitio:

;enter here from read and write to perform the actual

; I/O operation. return a 00h in register a if the

; operation completes properly, and 01h if an error

; occurs during the read or write

;

;in this case, the disk number saved in 'diskno'

; the track number in 'track'

; the sector number in 'sector'

; the dma address in 'dmaad'

;b still has r/w flag

0347 3E0A mvi a,10d ;set error count

0349 323B04 sta errors ;retry some failures 10 times

;before giving up

tryagn:

034C C5 push b

034D CDDA03 call headload

;h,l point to track byte for selected disk

0350 C1 pop b

0351 4E mov c,m

; decide whether to allow disk write precompenstation

0352 3E27 mvi a,39d ;inhibit precomp on trks 0-39

0354 B9 cmp c

0355 DA5C03 jc allowit

;inhibit precomp

0358 3E10 mvi a,10h

035A B0 ora b

035B 47 mov b,a ;goes out on the same port

; as read/write

allowit:

035C 2A3704 lhld dmaad ;get buffer address

035F C5 push b ;b has r/w code c has track

0360 2B dcx h ;save and replace 3 bytes below

;buf with trk,sctr,adr mark

0361 5E mov e,m

;figure correct address mark

0362 3A3C04 lda port

0365 E608 ani 08h

0367 3EFB mvi a,0fbh

0369 CA6E03 jz sin

036C E60F ani 0fh ;was double

;0bh is double density

;0fbh is single density

sin:

036E 77 mov m,a

;fill in sector

036F 2B dcx h

0370 56 mov d,m

0371 3A3604 lda sector ;note that invalid sector number

;will result in head unloaded

;error, so dont check

0374 77 mov m,a

;fill in track

0375 2B dcx h

0376 C1 pop b

0377 79 mov a,c

0378 4E mov c,m

0379 77 mov m,a

037A 7C mov a,h ;set up fdc dma address

037B D381 out haddr ;high byte

037D 7D mov a,l

037E D382 out laddr ;low byte

0380 78 mov a,b ;get r/w flag

0381 D380 out cmd1 ;start disk read/write

rwwait:

0383 C5 push b

0384 D5 push d

0385 E5 push h

0386 0E84 mvi c,flagwait

0388 1E05 mvi e,5

038A CD5500 call xdos ; wait for disk intrpt flag

038D E1 pop h

038E D1 pop d

038F C1 pop b

0390 71 mov m,c ;restore 3 bytes below buf

0391 23 inx h

0392 72 mov m,d

0393 23 inx h

0394 73 mov m,e

0395 DB80 in stat ;test for errors

0397 E6F0 ani 0f0h

0399 C8 rz ;a will be 0 if no errors

; error from disk

039A F5 push psw ;save error condition

;check for 10 errors

039B 213B04 lxi h,errors

039E 35 dcr m

039F C2A603 jnz redo ;not ten yet. do a retry

;we have too many errors. print out hex number for last

;received error type. cpm will print perm error message.

03A2 F1 pop psw ;get code

;set error return for operating system

03A3 3E01 mvi a,1

03A5 C9 ret

redo:

;b still has read/write flag

03A6 F1 pop psw ;get error code

03A7 E6E0 ani 0e0h ;retry if not track error

03A9 C24C03 jnz tryagn ;

;was a track error so need to reseek

03AC C5 push b ;save read/write indicator

;figure out the desired track

03AD 113204 lxi d,track

03B0 2A3904 lhld diskno ;selected disk

03B3 19 dad d ;point to correct trk indicator

03B4 7E mov a,m ;desired track

03B5 F5 push psw ;save it

03B6 CDCA02 call home

03B9 F1 pop psw

03BA 4F mov c,a

03BB CD0503 call settrk

03BE C1 pop b ;get read/write indicator

03BF C34C03 jmp tryagn

;

;

;

step: ;step head out towards zero

;if carry is set; else

;step in

; h,l point to correct track indicator word

03C2 DAD503 jc outx

03C5 34 inr m ;increment current track byte

03C6 3E04 mvi a,04h ;set direction = in

dostep:

03C8 F602 ori 2

03CA D380 out cmd1 ;pulse step bit

03CC E6FD ani 0fdh

03CE D380 out cmd1 ;turn off pulse

;the fdc-2 had a stepp ready line. the fdc-3 relies on

;software time out

03D0 3E10 mvi a,16d ;delay 8 ms

03D2 C31503 jmp delay

; ret

;

outx:

03D5 35 dcr m ;update track byte

03D6 AF xra a

03D7 C3C803 jmp dostep

;

headload:

;select and load the head on the correct drive

03DA 213D04 lxi h,prtout ;old slect info

03DD 46 mov b,m

03DE 2B dcx h ;new select info

03DF 7E mov a,m

03E0 23 inx h

03E1 77 mov m,a

03E2 F610 ori 10h ; enable interrupt

03E4 D383 out cmd2 ;select the drive

03E6 E6EF ani 0efh

;set up h.l to point to track byte for selected disk

03E8 113204 lxi d,track

03EB 2A3904 lhld diskno

03EE 19 dad d

;now check for needing a 35 ms delay

;if we have changed drives or if the head is unloaded

;we need to wait 35 ms for head settle

03EF B8 cmp b ;are we on the same drive

03F0 C2F803 jnz needdly

;we are on the same drive

;is the head loaded?

03F3 DB80 in stat

03F5 E680 ani 80h

03F7 C8 rz ;already loaded

needdly:

03F8 AF xra a

03F9 D380 out cmd1 ;load the head

03FB 3E46 mvi a,70d

03FD C31503 jmp delay

; ret

;

; BIOS Data Segment

;

0400 3C cnt60: db 60 ; 60 tick cntr = 1 sec

intstk: ; local intrpt stk

0401 C7C7C7C7C7 dw 0c7c7h,0c7c7h,0c7c7h,0c7c7h,0c7c7h

040B C7C7C7C7C7 dw 0c7c7h,0c7c7h,0c7c7h,0c7c7h,0c7c7h

0415 C7C7C7C7C7 dw 0c7c7h,0c7c7h,0c7c7h,0c7c7h,0c7c7h

041F C7C7C7C7C7 dw 0c7c7h,0c7c7h,0c7c7h,0c7c7h,0c7c7h

lstintstk:

0429 0000 svdhl: dw 0 ; saved Regs HL during int hndl

042B 0000 svdsp: dw 0 ; saved SP during int hndl

042D 0000 svdret: dw 0 ; saved return during int hndl

042F 00 tickn: db 0 ; ticking boolean,true = delayed

if debug

0430 44 intmsk: db 44h ; intrpt msk, enables clk intrpt, & con2

else

intmsk: db 54h ; intrpt msk, enables clk intrpt, & con0/2

endif

0431 00 preemp: db 0 ; preempted boolean

;

scrat: ; start of scratch area

0432 00 track: db 0 ; current trk on drive 0

0433 00 trak1: db 0 ; current trk on drive 1

0434 00 trak2: db 0

0435 00 trak3: db 0

0436 00 sector: db 0 ; currently selected sctr

0437 0000 dmaad: dw 0 ; current dma address

0439 00 diskno: db 0 ; current disk number

043A 00 dummy: db 0 ; must be 0 for dbl add

043B 00 errors: db 0

043C 00 port: db 0

043D 00 prtout: db 0

043E 00 dnsty: db 0

;

disks 2

043F+= DPBASE EQU $ ;BASE OF DISK PARAMETER BLOCKS

043F+00000000 DPE0: DW XLT0,0000H ;TRANSLATE TABLE

0443+00000000 DW 0000H,0000H ;SCRATCH AREA

0447+70045F04 DW DIRBUF,DPB0 ;DIR BUFF,PARM BLOCK

044B+1005F004 DW CSV0,ALV0 ;CHECK, ALLOC VECTORS

044F+00000000 DPE1: DW XLT1,0000H ;TRANSLATE TABLE

0453+00000000 DW 0000H,0000H ;SCRATCH AREA

0457+70045F04 DW DIRBUF,DPB1 ;DIR BUFF,PARM BLOCK

045B+50053005 DW CSV1,ALV1 ;CHECK, ALLOC VECTORS

0800 = bpb equ 2\*1024 ;bytes per block

0010 = rpb equ bpb/128 ;records per block

00FF = maxb equ 255 ;max block number

diskdef 0,1,58,,bpb,maxb+1,128,128,2,0

045F+= DPB0 EQU $ ;DISK PARM BLOCK

045F+3A00 DW 58 ;SEC PER TRACK

0461+04 DB 4 ;BLOCK SHIFT

0462+0F DB 15 ;BLOCK MASK

0463+00 DB 0 ;EXTNT MASK

0464+FF00 DW 255 ;DISK SIZE-1

0466+7F00 DW 127 ;DIRECTORY MAX

0468+C0 DB 192 ;ALLOC0

0469+00 DB 0 ;ALLOC1

046A+2000 DW 32 ;CHECK SIZE

046C+0200 DW 2 ;OFFSET

0000+= XLT0 EQU 0 ;NO XLATE TABLE

diskdef 1,0

045F+= DPB1 EQU DPB0 ;EQUIVALENT PARAMETERS

0020+= ALS1 EQU ALS0 ;SAME ALLOCATION VECTOR SIZE

0020+= CSS1 EQU CSS0 ;SAME CHECKSUM VECTOR SIZE

0000+= XLT1 EQU XLT0 ;SAME TRANSLATE TABLE

;

046E tran: ds 2

;

endef

0470+= BEGDAT EQU $

0470+ DIRBUF: DS 128 ;DIRECTORY ACCESS BUFFER

04F0+ ALV0: DS 32

0510+ CSV0: DS 32

0530+ ALV1: DS 32

0550+ CSV1: DS 32

0570+= ENDDAT EQU $

0100+= DATSIZ EQU $-BEGDAT

0570+00 FORCE: DB 0 ;FORCE OUT LAST BYTE IN HEX FILE

0571 00 db 0 ;force out last byte in hex file

0572 end

Приложение 5 Листинг источника XIOS с коммутацией банков

page 0

TITLE 'XIOS200, Copyright 1980, ALTOS COMPUTER SYSTEMS'

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

; ALTOS COMPUTER SYSTEMS

; 2360 BERING DRIVE

; SAN JOSE, CALIFORNIA 95131

;

; Copyright 1980, ALTOS COMPUTER SYSTEMS

;

; This program is a copyright program product of

; ALTOS COMPUTER SYSTEMS and is distributed to the

; owners of ALTOS SUN SERIES 8000 computers for

; use on those systems only. Any other use of this

; software constitutes a breach of the copyright

; license to the purchaser. However, permission is

; granted to use this listing as a sample for the

; construction of the reader's own XIOS.

;

; VERSION NUMBER: 1.12\*

; VERSION DATE: June 28, 1980

; Add support for CP/M version 2.0

; Add support for Hard disk drives

; Add support for disk MODE selection

; Provide compatability MODE for 1.4 operatio

; Remove CTC/1791 counter reset

; CORRECT HARD DISK SEEK PROBLEM

; Add code to recover from WD1791 going to sl

; Initialize parallel port for Centronics pri

; VERSION DATE: March 17, 1981

; Virtual disk in banks 1,2,3: M DISK

; VERSION DATE: April 11, 1981

; Conditional assembly for virtual disks

; Conditional assembly for MP/M 2.0

; VERSION DATE: April 14, 1981

; Equates added for LDRBIOS hooks

; VERSION DATE: April 16, 1981

; Testing for bank setup added

;

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

;

; Mode 0 IBM single density

; 1 ALTOS double density Version 2.0

; 2 ALTOS double density Version 1.4

; 3 ALTOS hard disk Version 2.0 (8 MEG

; 4 ALTOS HARD DISK VERSION 2.0 (8 MEG

; 5 ALTOS HARD DISK VERSION 2.0 (8 MEG

; 6 ALTOS HARD DISK VERSION 2.0 (4 MEG

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

; ASSEMBLER CONTROL STATEMENTS

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

MACLIB DISKDEF

MACLIB Z80S

FFFF = TRUE EQU OFFFFH ;VALUE FOR TRUE

0000 = FALSE EQU NOT TRUE ;VALUE FOR FALSE

0000 = mdisk equ false ;Virtual Disk cond asm bool

FFFF = mpm20 equ true ;MP/M 2.0 cond asm Boolean

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

1700 = ldrbiosbase equ 1700h ; for M

0037 = density$mask$offset equ 37h ;density mask offset from LDRBI

OOBB = misc$params$offset equ 0bbh ;misc. parameters offset from L

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

; THE FOLLOWING EQUATES ARE USER MODIFIABLE BASED ON

; PARTICULAR USER SYSTEM AND OPTIONS SELECTED.

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

FFFF = DMA EQU TRUE ;DMA HARDWARE SUPPORT ??

FFFF = HARDSK EQU TRUE ;HARD DISK SUPPORT

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

;

; THE FOLLOWING CONSTANTS APPLY TO THE DEBLOCKING OF

; SECTORS LARGER THAN 128 FOR THE ALTOS DOUBLE DENSIT

; AND THE ALTOS HARD DISK.

;

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

4000 = BLKSIZ EQU 16384 ;CP/M ALLOCATION SIZE

0400 = HSTSIZ EQU 1024 ;HOST DISK SECTOR SIZE

0010 = HSTSPT EQU 16 ;HOST DISK SECTORS PER TRAC

0008 = HSTBLK EQU HSTSIZ/128 ;CP/M SECTORS PER HOST BUFF

0080 = CPMSPT EQU HSTBLK \* HSTSPT ;CP/M SECTORS PER TRACK

0007 = SECMSK EQU HSTBLK - 1 ;SECTOR MASK

0003 = SECSHF EQU 3 ;LOG2(HHSTBLK)

PAGE

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

;

; THE FOLLOWING EQUATES APPLY TO THE RELOCATABILITY

; OF THE CBIOS AND SHOULD NOT BE USER ALTERED.

;

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

FFFF RELOC EQU TRUE ;RELOCATABLE VERSION ??

if mdisk

maxdsk equ 13

else

IF HARDSK

OOOC MAXDSK EQU 12 ;MAXIMUM NUMBER OF LOGICAL

ELSE

MAXDSK EQU 4 ;MAXIMUM NUMBER OF LOGICAL

ENDIF

endif

IF RELOC

0000 ORG 0000H

ELSE

ORG 0C000H

ENDIF

0000 BASE EQU $

0000 = WRALL EQU 0 ;WRITE TO ALLOCATED

0001 = WRDIR EQU 1 ;WRITE TO DIRECTORY

0002 = WRUAL EQU 2 ;WRITE TO UNALLOCATED

0004 = NMBCNS EQU 4 ; NUMBER OF CONSOLES

0083 = POLL EQU 131 ; XDOS POLL FUNCTION

0084 = FLAGWT EQU 132 ; XDOS FLAG WAIT FUNCTION

0085 = FLAGST EQU 133 ; XDOS FLAG SET FUNCTION

0005 = HDFLAG EQU 5 ;HARD DISK FLAG FOR WAIT & SET

0006 = FPYFLAG EQU 6 ;FLOPPY DISK FLAG FOR WAIT & SET

0000 = PLLPT EQU 0 ; POLL PRINTER

0001 = PLCOO EQU PLLPT+1 ; POLL CONSOLE OUT #0 (CRT:)

0002 = PLCOl EQU PLCO0+1 ; POLL CONSOLE OUT #1 (CRT:)

0003 = PLCO2 EQU PLCO1+1 ; POLL CONSOLE OUT #2 (CRT:)

0004 = PLCO3 EQU PLCO2+1 ; POLL CONSOLE OUT #3 (CRT:)

0005 = PLCI0 EQU PLCO3+1 ; POLL CONSOLE IN #0 (CRT:)

0006 = PLCIl EQU PLCI0+1 ; POLL CONSOLE IN #1 (CRT:)

0007 = PLC12 EQU PLCI1+1 ; POLL CONSOLE IN #2 (CRT:)

0008 = PLC13 EQU PLC12+1 ; POLL CONSOLE IN #3 (CRT:)

0009 = MEMPORT EQU 009H ; MEMORY SELECT PORT

0002 = MEMSK EQU 002H ; MEMORY SELECT MASK

PAGE

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

;

; JUMP VECTORS FOR ENTRIES TO CBIOS ROUTINES

;

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

; EXTERNAL JUMP TABLE (BELOW XIOS BASE)

;PDISP EQU $-3

;XDOS EQU PDISP-3

if mpm20

0000 C3040B jmp commonbase

else

JMP COLDSTART ;COLD START

endif

WBOTE:

0003 C3150B JMP WARMSTART ;WARM START

0006 C3790B JMP CONST ;CONSOLE STATUS

0009 C3840B JMP CONIN ;CONSOLE CHARACTER IN

OOOC C38FOB JMP CONOUT ;CONSOLE CHARACTER OUT

OOOF C3A90C JMP LIST ;LIST CHARACTER OUT - THIS

; ; "CLIST" IF SETUP PROGRAM

; ; PARALLEL PRINTER PORT

0012 C31A0B JMP RTNEMPTY ;PUNCH NOT IMPLEMENTED

0015 C31AOB JMP RTNEMPTY ;READER NOT IMPLEMENTED

0018 C3F902 JMP HOMEIT ;MOVE HEAD TO HOME

001B C30302 JMP SELDSK ;SELECT DISK

001E C36D02 JMP SETTRK ;SET TRACK NUMBER

0021 C37302 JMP SETSEC ;SET SECTOR NUMBER

0024 C35502 JMP SETDMA ;SET DMA ADDRESS

0027 C38BO2 JMP READ ;READ DISK

002A C39602 JMP WRITE ;WRITE DISK

002D C3BC0C JMP POLLPT ;LIST STATUS

0030 C3D605 JMP SECTRAN ;SECTOR TRANSLATE

; EXTENDED 1/0 SYSTEM JUMP VECTOR

0033 C3E90C JMP SELMEMORY ; SELECT MEMORY

0036 C3CB0C JMP POLLDEVICE ; POLL DEVICE

0039 C3050D JMP STARTCLOCK ; START CLOCK

003C C30B0D JMP STOPCLOCK ; STOP CLOCK

003F C3100D JMP EXITREGION ; EXIT REGION

0042 C3170D JMP MAXCONSOLE ; MAXIMUM CONSOLE NUMBER

0045 C39D12 JMP SYSTEMINIT ; SYSTEM INITIALIZATION

0048 00 NOP ; NO JMP HERE

0049 00 NOP ; FOR MP/M DELAY

004A 00 NOP

004B C3A102 JMP SETMOD ;ROUTINE TO SET DISK MODE

004E C3EE02 JMP RETMOD ;ROUTINE TO RETURN CURRENT

if not mpm20

COLDSTART:

WARMSTART:

MVI C,0 ; SEE SYSTEM INIT

; COLD & WARM START INCLUDE

; FOR COMPATIBILITY WITH CP/M

JMP XDOS ; SYSTEM RESET, TERMINATE PROCESS

RTNEMPTY:

XRA A ; NOT USED

RET

endif

LAST:

005E ORG (((LAST-BASE)+0A2H) AND 0FF00H) +05EH

INTERUPT:

005E 470B DW FLOPPY$INT ;FLOPPY DISK INTERRUPT

0060 1C0B DW NULL$INT ;

0062 1C0B DW NULL$INT ;

0064 1C0B DW NULL$INT ;

0066 1A0D DW INTlHND ;CTC INTERRUPT

0068 1C0B DW NULL$INT ;

006A 5E0B DW HARD$INT ;HARD DISK INTERRUPT

006C 1C0B DW NULL$INT

006E 1C0B DW NULL$INT

if not mpm20

NULL$INT:

EI

RETI

endif

PAGE

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

;

; WORK AND CONTROL AREAS FOR CBIOS SERVICES

;

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

0070 FFFFFFFFFFTRK0: DB 0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH

007C 0408102010SEL0: DB 004H,008H,010H,020H,010H,010H,010H,020H,020H

0088 0000000003MODE: DB 000H,000H,000H,000H,003H,004H,005H,003H,004H

0094 0000000000TCNT: DB 000H,000H,000H,000H,000H,000H,000H,000H,000H

00A0 0000000000PCNT: DB 000H,000H,000H,000H,000H,000H,000H,000H,000H

00AC 00 DISKNO: DB 000H ;CURRENT DRDRIVE

00AD 00 TRAKNO: DB 000H ;CURRENT TRACK

00AE 00 HEADNO: DB 000H ;CURRENT HEHEAD

00AF 0000 DMAADR: DW 000H ;CURRENT DMA ADRESS

00B1 00 SECTNO: DB 000H ;CURRENT SECTOR

00B2 0000 DPEPTR: DW 000H ;CURRENT DP

00B4 0000 DBLKAD: DW 000H ;CURRENT EX

00B6 0000 MPARMS: DW 000H ;MISC. PARA

00B8 10 HTK1: DB 10H ;HARD DISK

00B9 20 HTK2: DB 20H ;HARD DISK

;

; PARAMETER FLAGS

;

; 0100H = DOUBLE HEADED DRIVES

; 0200H = CENTRONICS PRINTER FOR LIST DEVICE

; 0400H = FOUR DRIVE SYSTEM [ A B C D ]

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

;

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

;

; NOTE:

; NO CHANGES ARE TO BE MADE TO THE ABSOLUTE LOCATIONS

; ANY FIELDS PRIOR TO THIS POINT. EXTERNAL PROGRAMS A

; DEPENDENT UPON THE LOCATION OF THE PRECEEDING DATA.

;

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

IF NOT DMA

NMIRTN: DB 0EDH,0A2H,0EDH,045H ;FAKE INI A

ENDIF

00BA C37D DMAS1: DB 0C3H,07DH ;FIRST PART

00BC 0000 DMASA: DW 000H ;ADDRESS FO

00BE 0004 DMALEN: DW 1025-1 ;LENGTH FOR

00C0 54CE68CEA5DMAS2H: DB 054H,0CEH,068H,0CEH,0A5H,020H ;HARD DISK

0OC6 14288507 DMAS2F: DB 014H,028H,085H,007H ;FLOPPY DISK

00CA 8ACF01CF DMAS3: DB 08AH,0CFH,001H,0CFH ;LAST PART

00CE 01 DMAS3F: DB 001H ;001=READ,

00CF CF87 DB 0CFH,087H ;SETUP DMA,

PAGE

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

;

; CONTROL BLOCKS FOR DISK DRIVER

;

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

00D1 = DPBASE EQU $ ;START OF DISK PARAMETER BL

00D1 B5010000 DPEO: DW XLT0,0000H ;TRANSLATE TABLE AND WORK A

00D5 00000000 DW 0000H,0000H ;SCRATCH AREA

00D9 9D12D40D DW DIRBUF,DPB0 ;DIR BUFF, PARM BLOCK

00DD 3EO81EO8 DW CSV0,ALV0 ;CHECK VECTOR, ALLOC VECTOR

00E1 B5010000 DPE1: DW XLT0,0000H ;TRANSLATE TABLE AND WORK A

00E5 00000000 DW 0000H,0000H ;SCRATCH AREA

00E9 9D12D40D DW DIRBUF,DPB0 ;DIR BUFF, PARM BLOCK

00ED 7EO85EO8 DW CSV1,ALV1 ;CHECK VECTOR, ALLOC VECTOR

00F1 B5010000 DPE2: DW XLT0,0000H ;TRANSLATE TABLE AND WORK A

00F5 00000000 DW 0000H,0000H ;SCRATCH AREA

00F9 9D12D40D DW DIRBUF,DPB0 ;DIR BUFF, PARM BLOCK

00FD BE089EO8 DW CSV2,ALV2 ;CHECK VECTOR, ALLOC VECTOR

0101 B5010000 DPE3: DW XLT0,0000H ;TRANSLATE TABLE AND WORK A

0105 00000000 DW 0000H,0000H ;SCRATCH AREA

0109 9D12D40D DW DIRBUF,DPB0 ;DIR BUFF, PARM BLOCK

010D FE08DE08 DW CSV3,ALV3 ;CHECK VECTOR, ALLOC VECTOR

IF HARDSK

0111 00000000 DPE4: DW 0000H,0000H ;TRANSLATE TABLE AND WORK A

0115 00000000 DW 0000H,0000H ;SCRATCH AREA

0119 9D12010E DW DIRBUF,DPB3 ;DIR BUFF, PARM BLOCK

011D 5E091E09 DW CSV4,ALV4 ;CHECK VECTOR, ALLOC VECTOR

0121 00000000 DPE5: DW 0000H,0000H ;TRANSLATE TABLE AND WORK A

0125 00000000 DW 0000H,0000H ;SCRATCH AREA

0129 9D12100E DW DIRBUF,DPB4 ;DIR BUFF, PARM BLOCK

012D 9E095E09 DW CSV5,ALV5 ;CHECK VECTOR, ALLOC VECTOR

0131 00000000 DPE6: DW 0000H,0000H ;TRANSLATE TABLE AND WORK A

0135 00000000 DW 0000H,0000H ;SCRATCH AREA

0139 9D121F0E DW DIRBUF,DPB5 ;DIR BUFF, PARM BLOCK

013D DE099E09 DW CSV6,ALV6 ;CHECK VECTOR, ALLOC VECTOR

0141 00000000 DPE7: DW 0000H,0000H ;TRANSLATE TABLE AND WORK A

0145 00000000 DW 0000H,0000H ;SCRATCH AREA

0149 9D12010E DW DIRBUF,DPB3 ;DIR BUFF, PARM BLOCK

014D 1E0ADE09 DW CSV7,ALV7 ;CHECK VECTOR, ALLOC VECTOR

0151 00000000 DPE8: DW 0000H,0000H ;TRANSLATE TABLE AND WORK A

0155 00000000 DW 0000H,0000H ;SCRATCH AREA

0159 9D12100E DW DIRBUF,DPB4 ;DIR BUFF, PARM BLOCK

015D 5E0AlE0A DW CSV8,ALV8 ;CHECK VECTOR, ALLOC VECTOR

0161 00000000 DPE9: DW 0000H,0000H ;TRANSLATE TABLE AND WORK A

0165 00000000 DW 0000H,0000H ;SCRATCH AREA

0169 9D121F0E DW DIRBUF,DPB5 ;DIR BUFF, PARM BLOCK

016D 9E0A5E0A DW CSV9,ALV9 ;CHECK VECTOR, ALLOC VECTOR

0171 00000000 DPEA: DW 0000H,0000H ;TRANSLATE TABLE AND WORK A

0175 00000000 DW 0000H,0000H ;SCRATCH AREA

0179 9D122E0E DW DIRBUF,DPB6 ;DIR BUFF, PARM BLOCK

017D C20A9E0A DW CSVA,ALVA ;CHECK VECTOR, ALLOC VECTOR

0181 00000000 DPEB: DW 0000H,0000H ;TRANSLATE TABLE AND WORK A

0185 00000000 DW 0000H,0000H ;SCRATCH AREA

0189 9D122E0E DW DIRBUF,DPB6 ;DIR BUFF, PARM BLOCK

018D E60AC20A DW CSVB,ALVB ;CHECK VECTOR, ALLOC VECTOR

ENDIF

if mdisk

; Virtual disk parameter header

DPEC: DW 0000H,0000H ;TRANSLATE TABLE AND WORK A

DW 0000H,0000H ;SCRATCH AREA

DW DIRBUF,DPB7 ;DIR BUFF, PARM BLOCK

DW CSVC,ALVC ;CHECK VECTOR, ALLOC VECTOR

endif

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

0191 B5010000 MODL0: DW XLT0,000H ;MODEL DPE FOR MODE 0

0195 00000000 DW 000H,000H

0199 9D12D40D DW DIRBUF,DPB0

019D CF010000 MODL1: DW XLT1,0000H ;MODEL DPE FOR MODE 1

01A1 00000000 DW 0000H,0000H

01A5 9D12E30D DW DIRBUF,DPB1

01A9 CF010000 MODL2: DW XLT2,0000H ;MODEL DPE FOR MODE 2

01AD 00000000 DW 0000H,0000H

OlBl 9D12F20D DW DIRBUF,DPB2

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

01B5 01070D1319XLT0: DB 1,7,13,19,25,5,11,17,23,3, 9,15,21

01C2 0208OE141A DB 2,8,14,20,26,6,12,18,24,4,10,16,22

XLT1:

XLT2:

01CF 0102030405 DB 01,02,03,04,05,06,07,08,09,10,11,12,13

01DC 0E0F101112 DB 14,15,16,17,18,19,20,21,22,23,24,25,26

01E9 lBlC1D1E1F DB 27,28,29,30,31,32,33,34,35,36,37,38,39

01F6 28292A2B2C DB 40,41,42,43,44,45,46,47,48,49,50,51,52

PAGE

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

;

; DISK ACCESS ROUTINES

;

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

SELDSK:

0203 79 MOV A,C ;LIMIT SELECT TO REAL OPTIO

0204 FE0C CPI MAXDSK ;

JRNC SELERR ; INVALID DRIVE

0206+303A DB 030H,SELERR-$-1 ;---- FAKE JRNC INSTRUCTION

; MOV A,E ; TEST FOR INITIAL SELECT

; ANI 1 ; E = 0 IS FIRST TIME

; PUSH PSW ;

0208 1600 MVI D,0 ;

020A 59 MOV E,C ; TRANSLATE TABLE

020B 214602 LXI H,DTBLT ; FOR LOGICAL TO PHYSICAL

020E 19 DAD D ;

020F 4E MOV C,M ; C = PHYSICAL DRIVE

0210 79 MOV A,C ; M translates to the 12 di

if mdisk

CPI 12

JZ VIRTUAL

endif

; POP PSW ; RESTORE TEST

; JRNZ SELSDP ; BYPASS SELECT

SETDSK:

0211 0600 MVI B,0 ;

0213 217C00 LXI H,SEL0 ; BASE OF SELECT MASKS

0216 09 DAD B ;

0217 7E MOV A,M ; GET SELECT BYTE

0218 A7 ANA A ; CHECK FOR VALID DRIVE

JRZ SELERR ; DRIVE NOT CONFIGURED

0219+2827 DB 028H,SELERR-$-1 ;---- FAKE JRZ INSTRUCTION

021B 79 MOV A,C ;

021C FE04 CPI 4 ; CHECK FOR FLOPPY

JRC SELSDP ;

021E+380F DB 038H,SELSDP-$-1 ;---- FAKE JRC INSTRUCTION

0220 7E CHKHRD: MOV A,M ; RESTORE SELECT BYTE

0221 D320 OUT 20H ;

0223 C5 PUSH B ;

0224 0E01 MVI C,1 ; DELAY FOR 1 MS

0226 CD8207 CALL DELAY ;

0229 Cl POP B ;

022A DB24 IN 24H ; CHECKFOR HARD DISK READY

022C 17 RAL ; 80H = READY

JRNC SELERR ;

022D+3013 DB 030H,SELERR-$-1 ;----FAKE JRNC INSTRUCTION

SELSDP:

022F 79 MOV A,C ;

if mdisk

VIRTUAL:

endif

0230 32E60A STA NEWDSK ;SAVE FOR I/O LATER

0233 2600 MVI H,0 ;

0235 69 MOV L,C ;COMPUTE DP HEADER ADDRESS

0236 29 DAD H ;\* 2

0237 29 DAD H ;\* 4

0238 29 DAD H ;\* 8

0239 29 DAD H ;\* 16 (DP HEADER SIZE)

023A 11D100 LXI D,DPBASE ;START OF DP HEADERS

023D 19 DAD D ;POINT TO CORRECT ONE

023E 22B200 SHLD DPEPTR ;SAVE ADDRESS OF CURRENT DP

0241 C9 RET ;

0242 210000 SELERR: LXI H,0 ; INDICATE ERROR

0245 C9 RET ; AND RETURN

; A,B,C,D,E,F,G,H,I,J, K, L, M N,O

0246 0001020304DTBLT: DB 0,1,2,3,4,5,6,7,8,9,10,11,12,0,0

SETDMA:

0255 60 MOV H,B ;TO ALLOW SAVING

0256 69 MOV L,C ;

0257 22AF00 SHLD DMAADR ;

if mpm20

025A 23 inx h ;test for flush buffers

025B 7D mov a,l

025C B4 ora h

025D C0 rnz ;HL = FFFFh is flush buffer

025E 21F00A lxi h,hstwrt

0261 7E mov a,m

0262 3600 mvi m,0

0264 B7 ora a

0265 C8 rz

0266 CD6D04 call writehst ;flush host write if pendin

0269 B7 ora a

026A C8 rz ;return if no error

026B El pop h

endif

026C C9 ret

SETTRK:

026D 60 MOV H,B ;TO ALLOW SAVE

026E 69 MOV L,C ;

026F 22E70A SHLD NEWTRK ;SAVE NEXT TRACK NUMBER

0272 C9 RET ;RETURN TO CALLER

SETSEC:

0273 79 MOV A,C ;FOR SAVE

0274 32E90A STA NEWSEC ;

0277 C9 RET ;RETURN TO CALLER

SETDEN:

0278 117C00 LXI D,SEL0 ;START OF SELECT/DENSITY MA

027B 2AE60A LHLD NEWDSK ;NEXT DRIVE ADDRESS

027E 2600 MVI H,000H ;ENSURE ZERO FOR SINGLE BYT

0280 19 DAD D ;POINT TO CORRECT MASK

0281 79 MOV A,C ;ISOLATE DENSITY BIT

0282 E601 ANI 00000001B ;

0284 4F MOV C,A ;SAVE FOR NOW

0285 7E MOV A,M ;LOAD SELECT DENSITY MASK

0286 ME ANI 11111110B ;RESET CURRENT DENSITY SETT

0288 B1 ORA C ;SET NEW VALUE

0289 77 MOV M,A ;RESTORE MASK IN TABLE

028A C9 RET ;RETURN TO CALLER

if mdisk

MREADSECTOR:

call compbank ;compute bank

di

call chgbank

lxi b,128

lxi d,localbuf

lhld addroff

ldir ;block move into the dma ar

mvi a,02h ; select bank 0

out 09h

ei

lxi b,128

lhld dmaadr

xchg

lxi h,localbuf

ldir

xra a

ret

mbankno db 0

addroff dw 0

localbuf ds 128

compbank:

lda newtrk

mov h,a

ani 0fh ;save track rem 16

mov l,a

mov a,h ;restore track

mvi h,0

ani 0f0h ; bank is high order nibble

rar ! rar ! rar ! rar

inr a

sta mbankno ; which bank we want

dad h ;trk 0-15

dad h ; \* 2

dad h ; \* 4

mov e,l

mov d,h

dad d

dad d ; \* 24:

lda newsec ; figure offset within the

mov e,a

mvi d,0

dad d ; add sector offset within

dad h ! dad h ! dad h ! dad h ! dad h ! dad h !

shld addroff ; (track \* 24 + sector) \* 1

ret

endif

READ:

if mdisk

LDA NEWDSK

CPI 12 ;VIRTUAL DISK ?

iz MREADSECTOR

endif

028B CDEE02 CALL RETMOD ;WHAT TYPE OF I/O ??

028E FE03 CPI 003H

0290 DAE405 ic READSOFT ;FLOPPY DISK DRIVE ....

0293 C36B03 imp READHARD ;HARD DISK I/O

if mdisk

mwritesector:

call compbank

lhld dmaadr

lxi d,localbuf

lxi b,128

ldir

di

call chgbank

lxi d,localbuf

lxi b,128

lhld addroff

xchg

ldir

mvi a,02h ; select bank 0

out 09h

ei

xra a

ret

chgbank:

lda mbankno

ral

ral

ral

ani 018h

ori memsk

out 009h

ret

endif

WRITE:

if mdisk

lda newdsk

cpi 12

jz mwritesector

endif

0296 CDEE02 CALL RETMOD ;WHAT TYPE OF 1/0 ??

0299 FE03 CPI 003H ;

029B DAF205 JC WRITESOFT ;FLOPPY DISK

029E C37E03 JMP WRITEHARD ;HARD DISK 1/0

PAGE

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

;

; ROUTINES TO SET AND RETURN THE CURRENT DRIVE MODE

;

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

SETMOD:

02A1 21E60A LXI H,NEWDSK ; SAVE NEWDSK IN STACK

02A4 7E MOV A,M ;

02A5 F5 PUSH PSW ;

02A6 70 MOV M,B ;

02A7 C5 PUSH B ;

; MVI E,0 ; INDICATE INITIAL SELECT

02A8 48 MOV C,B ; CALL DISK SELECT

02A9 CD0302 CALL SELDSK ;

02AC Cl POP B ;

02AD 7C MOV A,H ; CHECK FOR BAD SELECT

02AE B5 ORA L ;

JRZ SMERR ; YES - ABORT CHANGING

02AF+2832 DB 028H,SMERR-$-1 ;---- FAKE JRZ INSTRUCTION

02B1 68 MOV L,B ; B AND L DRIVE #

02B2 2600 MVI H,000H ;

02B4 78 MOV A,B ;CHECK MODE SET VALIDITY

02B5 FE04 CPI 004H ;ONLY VALID FOR FLOPPY DISK

JRNC SMERR ;INVALID DRIVE FOR MODE SET

02B7+302A DB 030H,SMERR-$-1 ;---- FAKE JRNC INSTRUCTION

02B9 118800 LXI D,MODE ;START OF MODE BYTES

02BC 19 DAD D ;

02BD 71 MOV M,C ;SAVE NEW MODE BYTE

02BE E5 PUSH H ;SAVE MODE BYTE ADDRESS

02BF 79 MOV A,C ;SETUP FOR DENSITY CHANGE

02CO B7 ORA A ;

02C1 0E00 MVI C,000H ;ASSUME SINGLE DENSITY MODE

JRZ SETSEL ;VERIFY ASSUMPTION

02C3+2802 DB 028H,SETSEL-$-1 ;---- FAKE JRZ INSTRUCTION

02C5 0E01 MVI C,001H ;SET FOR DOUBLE DENSITY MOD

02C7 CD7802 SETSEL: CALL SETDEN ;SET DENSITY BASED ON LOW B

02CA El POP H ;RESTORE

02CB 6E MOV L,M ;PICKUP MODE AGAIN

02CC 2600 MVI H,000H ;FOR SINGLE BYTE PRECISION

02CE 7D MOV A,L ;SAVE MODE IN ACCUMULATOR F

02CF 29 DAD H ;\* 2

02DO 29 DAD H ;\* 4

02D1 E5 PUSH H ;SAVE \* 4

02D2 29 DAD H ;\* 8

02D3 Dl POP D ;REGAIN \* 4

02D4 19 DAD D ;\* 12

02D5 119101 LXI D,MODL0 ;FIRST MODEL DPE

02D8 19 DAD D ;POINT TO THIS ONE

02D9 EB XCHG ;SETUP TEMPORARILY AS DESTI

02DA 2AB200 LHLD DPEPTR ;ADDRESS OF CURRENTLY SELEC

02DD EB XCHG ;SETUP TO ALTER

02DE 010COO LXI B,12 ;LENGTH FOR MOVE

LDIR ;DO MOVE

02El+EDBO DB 0EDH,0B0H ;---- FAKE LDIR INSTRUCTION

02E3 F1 SMERR: POP PSW ;

02E4 E5 PUSH H ;

02E5 32E60A STA NEWDSK ; RESTORE ORIGINAL NEWDSK

02E8 4F MOV C,A ;

02E9 CD0302 CALL SELDSK ;

02EC E1 POP H ;

02ED C9 RET ;RETURN TO CALLER

RETMOD:

02EE 118800 LXI D,MODE ;START OF MODE BYTES

02F1 2AE60A LHLD NEWDSK ;NEXT DRIVE FOR I/O

02F4 2600 MVI H,000H ;RESET FOR SINGLE BYTE QUAN

02F6 19 DAD D ;POINT TO IT ....

02F7 7E MOV A,M ;LOAD IT FOR CALLER

02F8 C9 RET ;RETURN, WITH CURRENT MODE

PAGE

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

;

; THIS IS THE HOME DEVICE ROUTINE

;

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

02F9 3AE60A HOMEIT: LDA NEWDSK ; CHECK FOR FIRST HOME

02FC FE0C CPI 12 ; CHECK FOR VIRTUAL DISK

02FE C20803 JNZ REALDISK

0301 AF XRA A ; VIRTUAL DISK

0302 67 MOV H,A ; SET TRACK TO ZERO

0303 6F MOV L,A

0304 22E70A SHLD NEWTRK

0307 C9 RET

REALDISK:

0308 FE04 CPI 4 ; CHECK FOR FLOPPY

JRC HOME ; DO NOT BYPASS FLOPPY HOME

030A+380E DB 038H,HOME-$-1 ;---- FAKE JRC INSTRUCTION

030C 4F MOV C,A ;

030D 0600 MVI B,0 ; POINT TO PRESENT TRACK ST

030F 217000 LXI H,TRK0 ;

0312 09 DAD B ;

0313 7E MOV A,M ; CHECK IF INITIALIZED

0314 FEFF CPI 0FFH ;

0316 3E00 MVI A,0 ;

0318 C0 RNZ ; YES - RETURN WITH NO ER

0319 77 MOV M,A ;

HOME:

031A 3AE60A LDA NEWDSK ;GET VALUE OF DRIVE FOR HOME

031D FE04 CPI 004H ;IS IT A HARD DISK ??

JRNC HOMEHARD ;YES, PROCESS

031F+3022 DB 30H,HOMEHARD-$-1 ;---- FAKE JRNC INSTRUCTION

HOMESOFT:

0321 CD5205 CALL DSKSEL ;SELECT CORRECT DRIVE (IN A

0324 3AF60A LDA ERFLAG ;

0327 B7 ORA A ;CHECK FOR ERRORS DURING SE

JRNZ HOME1A ;EXIT IF ERRORS

0328+2016 DB 020H,HOME1A-$-1 ;---- FAKE JRNZ INSTRUCTION

032A CDB305 CALL POINT ;POINT TO TRACK REGISTER SA

032D 3600 MVI M,000H ;RESET TO TRACK ZERO

032F CD1905 CALL DBL$UPDATE ;

0332 3E0A MVI A,00AH ;HOME COMMAND

0334 CD6307 CALL FINTFIX ;CLEAR ANY PENDING INTERRUP

;AND ISSUE COMMAND

;

0337 CD3A07 HOMEl: CALL FPYWAIT ;WAIT UNTIL I/O COMPLETE

033A 3AFC0A LDA STATUS ;PICKUP STATUS BYTE

033D E698 ANI 10011000B ;CHECK STATUS

033F C8 RZ ;RETURN WITH GOOD ESULT

0340 3E01 HOMElA: MVI A,001H ;SET ERROR ON HOME

0342 C9 RET ;AND RETURN

HOMEHARD:

IF HARDSK

0343 CD5205 CALL DSKSEL ;SELECT CORRECT DRIVE (IN A

0346 CDB305 CALL POINT ;POINT TO SAVE AREA

0349 3600 MVI M,000H ;SET TO TRACK ZERO

034B EB XCHG ;POINT TO SELECT WORD

034C 7E MOV A,M ;LOAD SELECT MASK

034D E6F0 ANI 11110000B ;RESET HEAD MASK

034F 77 MOV M,A ;SAVE

0350 D320 OUT 020H ;WRITE HEAD/SELECT MASK

0352 3E20 MVI A,020H ;HOME COMMAND

0354 CD2107 CALL INTFIX ;CLEAR ANY PENDING INTERRUP

; ;AND ISSUE COMMAND

0357 CD1707 HOME2: CALL WAIT0 ;WAIT UNTIL I/O COMPLETE

035A 0E14 MVI C,20 ;DELAY FOR 20 MILLISECONDS

035C CD8207 CALL DELAY ;

035F AF XRA A ;SET NEW TRACK REGISTER TO

0360 D322 OUT 022H ;FOR CONTROLLER

; LXI H,MHM ;\*\*\*DEBUG\*\*\*

; CALL MSPRT ;\*\*\*DEBUG\*\*\*

0362 3AFC0A LDA STATUS ;PICKUP STATUS BYTE

0365 E65D ANI 01011101B ;CHECK STATUS

0367 C8 RZ ;

0368 3E01 MVI A,001H ;SET ERROR ON HOME

ENDIF

036A C9 RET ;AND RETURN

PAGE

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

;

; THESE ARE THE HARD DISK UNBLOCK/REBLOCK AND READ AN

; ROUTINES CALLED BY THE BDOS SOFTWARE.

;

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

READHARD:

IF HARDSK

036B AF XRA A ;RESET UNALLOCATED COUNT

036C 32F10A STA UNACNT ;

036F 3EO1 MVI A,001H ;READ THE SELECTED CP/M SEC

0371 32F80A STA READOP ;

0374 32F70A STA RSFLAG ;MUST READ DATA

0377 3EO2 MVI A,WRUAL ;

0379 32F90A STA WRTYPE ;TREAT AS UNALLOCATED

JR RWOPER ;TO PERFORM THE READ

037C+1864 DB 018H,RWOPER-$-1 ;---- FAKE JR INSTRUCTION -

ENDIF

WRITEHARD:

IF HARDSK

037E AF XRA A ;WRITE THE SELECTED CP/M SE

037F 32F80A STA READOP ;NOT A READ OPERATION

0382 79 MOV A,C ;WRITE TYPE IS PASSED IN RE

0383 32F90A STA WRTYPE ;

if mpm20

0386 E602 ani WRUAL ;IS IT WRITE UNALLOCATED ??

JRZ CHKUNA ;CHECK FOR UNALLOCATED

0388+2817 DB 028H,CHKUNA-$-1 ;---- FAKE JRZ INSTRUCTION

else

CPI WRUAL ;IS IT WRITE UNALLOCATED ??

JRNZ CHKUNA ;CHECK FOR UNALLOCATED

endif

;

; WRITE TO UNALLOCATED, SET PARAMETERS

;

038A 3E80 MVI A,BLKSIZ/128 ;NEXT UNALLOC RECS

038C 32F10A STA UNACNT ;

038F 3AE60A LDA NEWDSK ;DISK FOR I/O

0392 32F20A STA UNADSK ;UNADSK = NEWDSK

0395 2AE70A LHLD NEWTRK ;

0398 22F30A SHLD UNATRK ;UNATRK = NEWTRK

039B 3AE90A LDA NEWSEC ;

039E 32F50A STA UNASEC ;UNASEC = NEWSEC

;

; CHECK FOR WRITE TO UNALLOCATED SECTOR

;

CHKUNA:

03A1 3AF10A LDA UNACNT ;ANY UNALLOCATED REMAIN ??

03A4 B7 ORA A ;

JRZ ALLOC ;SKIP IS NOT

03A5+2833 DB 028H,ALLOC-$-1 ;---- FAKE JRZ INSTRUCTION

;

; MORE UNALLOCATED RECORDS REMAIN

;

03A7 3D DCR A ;UNACNT = UNACNT - 1

03A8 32F10A STA UNACNT ;

03AB 3AE60A LDA NEWDSK ;SAME DISK ??

03AE 21F20A LXI H,UNADSK ;

03B1 BE CMP M ;NEWDSK = UNADSK ??

JRNZ ALLOC ;SKIP IF NOT

03B2+2026 DB 020H,ALLOC-$-1 ;---- FAKE JRNZ INSTRUCTION

;

; DISKS ARE THE SAME

;

03B4 21F30A LXI H,UNATRK ;

03B7 CD6104 CALL NEWTRKCMP ;NEWTRK = UNATRK ??

JRNZ ALLOC ;SKIP IF NOT

03BA+201E DB 020H,ALLOC-$-1 ;---- FAKE JRNZ INSTRUCTION

;

; TRACKS ARE THE SAME

;

03BC 3AE90A LDA NEWSEC ;SAME SECTOR ??

03BF 21F50A LXI H,UNASEC ;

03C2 BE CMP M ;NEWSEC = UNASEC ??

JRNZ ALLOC ;SKIP IF NOT

03C3+2015 DB 020H,ALLOC-$-1 ;---- FAKE JRNZ INSTRUCTION

;

; MATCH, MOVE TO NEXT SECTOR FOR FUTURE REFERENCE

;

03C5 34 INR M ;UNASEC = UNASEC + 1

03C6 7E MOV A,M ;END OF TRACK ??

03C7 FE80 CPI CPMSPT ;COUNT CP/M SECTORS

JRC NOOVF ;SKIP IF NO OVERFLOW

03C9+3809 DB 038H,NOOVF-$-1 ;---- FAKE JRC INSTRUCTION

;

; OVERFLOW TO NEXT TRACK

;

03CB 3600 MVI M,000H ;UNASEC = 0

03CD 2AF30A LHLD UNATRK ;

03DO 23 INX H ;

03D1 22F30A SHLD UNATRK ;UNATRK = UNATRK + 1

;

; MATCH FOUND, MARK AS UNNECESSARY READ

;

NOOVF:

03D4 AF XRA A ;ZERO TO ACCUMULATOR

03D5 32F70A STA RSFLAG ;RSFLAG = 0

JR RWOPER ;TO PERFORM THE WRITE

03D8+1808 DB 018H,RWOPER-$-1 ;---- FAKE JR INSTRUCTION

;

; NOT AN UNALLOCATED RECORD, REQUIRES PRE-READ

;

ALLOC:

03DA AF XRA A ;ZERO TO ACCUMULATOR

03DB 32F10A STA UNACNT ;UNACNT = 0

03DE 3C INR A ;ONE TO ACCUMULATOR

03DF 32F70A STA RSFLAG ;RSFLAG = 1

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

;

; THE FOLLOWING CODE IS COMMON TO BOTH READ AND WRITE

;

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

RWOPER:

03E2 AF XRA A ;ZERO TO ACCUMULATOR

03E3 32F60A STA ERFLAG ;NO ERRORS YET ....

03E6 3AE90A LDA NEWSEC ;COMPUTE HOST SECTOR

REPT SECSHF ;COMPUTE HOST SECTOR

ORA A ;CARRY = 0

RAR ;SHIFT RIGHT

ENDM

03E9+B7 ORA A ;CARRY = 0

03EA+1F RAR ;SHIFT RIGHT

03EB+B7 ORA A ;CARRY = 0

03EC+1F RAR ;SHIFT RIGHT

03ED+B7 ORA A ;CARRY = 0

03EE+1F RAR ;SHIFT RIGHT

03EF 32EE0A STA NEWHST ;HOST SECTOR TO SEEK

;

; ACTIVE HOST SECTOR ??

;

03F2 21EFOA LXI H,HSTACT ;HOST ACTIVE FLAG

03F5 7E MOV A,M ;

03F6 3601 MVI M,001H ;ALWAYS BECOMES 1

03F8 B7 ORA A ;WAS IT ALREADY ??

JRZ FILLHST ;FILL HOST IF NOT

03F9+2821 DB 28H,FILLHST-$-1 ;---- FAKE JRZ INSTRUCTION

;

; HOST BUFFER ACTIVE, SAME AS SEEK BUFFER

;

03FB 3AE60A LDA NEWDSK ;

03FE 21EA0A LXI H,HSTDSK ;SAME DISK ??

0401 BE CMP M ;NEWDSK = HSTDSK ??

JRNZ NOMATCH ;

0402+2011 DB 20H,NOMATCH-$-1 ;---- FAKE JRNZ INSTRUCTION

;

; SAME DISK, SAME TRACK ??

;

0404 21EB0A LXI H,HSTTRK ;

0407 CD6104 CALL NEWTRKCMP ;NEWTRK = HSTTRK ??

JRNZ NOMATCH ;

040A+2009 DB 20H,NOMATCH-$- 1 ;---- FAKE JRNZ INSTRUCTION

;

; SAME DISK, SAME TRACK, SAME BUFFER ??

;

040C 3AEE0A LDA NEWHST ;

040F 21ED0A LXI H,HSTSEC ;NEWHST = HSTSEC ??

0412 BE CMP M ;

JRZ MATCH ;SKIP IF MATCH

0413+2824 DB 028H,MATCH-$-1 ;---- FAKE JRZ INSTRUCTION

;

; PROPER DISK, BUT NOT CORRECT SECTOR

;

NOMATCH:

0415 3AF0A LDA HSTWRT ;HOST WRITTEN ??

0418 B7 ORA A ;

0419 C46D04 CNZ WRITEHST ;CLEAR HOST BUFFER

;

; MAY HAVE TO FILL HOST BUFFER

;

FILLHST:

041C 3AE60A LDA NEWDSK ;

041F 32EAOA STA HSTDSK ;

0422 2AE70A LHLD NEWTRK ;

0425 22EBOA SHLD HSTTRK ;

0428 3AEEOA LDA NEWHST ;

042B 32EDOA STA HSTSEC ;

042E 3AF70A LDA RSFLAG ;NEED TO READ ??

0431 B7 ORA A ;

0432 C47FO4 CNZ READHST ;YES, IF 1

0435 AF XRA A ;ZERO TO ACCUMULATOR

0436 32FOOA STA HSTWRT ;NO PENDING WRITE

MATCH:

0439 3AE90A LDA NEWSEC ;MASK BUFFER NUMBER

043C E607 ANI SECMSK ;LEAST SIGNIF BITS

043E 6F MOV L,A ;READY TO SHIFT

043F 2600 MVI H,OOOH ;DOUBLE COUNT

REPT 7

DAD H ;

ENDM

0441+29 DAD H ;

0442+29 DAD H ;

0443+29 DAD H ;

0444+29 DAD H ;

0445+29 DAD H ;

0446+29 DAD H ;

0447+29 DAD H ;

;

; HL NOW HAS RELATIVE HOST BUFFER ADDRESS

;

0448 119C0E LXI D,HSTBUF ;

044B 19 DAD D ;HL = HOST ADDRESS

044C EB XCHG ;NOW IN DE

044D 2AAF00 LHLD DMAADR ;GET/PUT CP/M DATA

0450 EB XCHG ;SET FOR Z80 LDIR INSTRUCTION

LXI B,128 ;LENGTH OF MOVE

0451 3AF80A LDA READOP ;WHICH WAY ??

0454 B7 ORA A ;

0455 C23D0E JNZ RWMOVE ;SKIP IF READ

;

; WRITE OPERATION, MARK AND SWITCH DIRECTION

;

0458 3E01 MVI A,001H ;

045A 32F00A STA HSTWRT ;HSTWRT = 1

045D EB XCHG ;SWAP DIRECTION

045E C33D0E jmp rwmove

endif

PAGE

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

;

; UTILITY SUBROUTINE FOR 16 BIT COMPARE

;

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

IF HARDSK

NEWTRKCMP:

0461 EB XCHG ;HL = UNATRK OR HSTTR

0462 21E70A LXI H,NEWTRK ;

0465 1A LDAX D ;LOW BYTE COMPARE

0466 BE CMP M ;SAME ??

0467 C0 RNZ ;RETURN IF NOT

0468 13 INX D ;TO CHECK HIGH BYTE

0469 23 INX H ;

046A 1A LDAX D ;

046B BE CMP M ;SETS FLAGS

046C C9 RET ;

PAGE

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

;

; WRITEHST PERFORMS THE PHYSICAL WRITE TO THE HOST DI

; READHST PERFORMS THE PHYSICAL READ FROM THE HOST DI

;

; HSTDSK = HOST DISK NUMBER

; HSTTRK = HOST TRACK NUMBER

; HSTSEC = HOST SECTOR NUMBER

; RETURN ERROR FLAG IN ERFLAG

;

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

WRITEHST:

046D 3E05 MVI A,005H ;SETUP DMA FOR WRITE

046F 32CE00 STA DMAS3F ;

0472 3EO2 MVI A,002H ;WRITE COMMAND

0474 32FA0A STA CMD ;SAVE FOR LATER

0477 219B0E LXI H,HSTBUF-1 ;WRITE MUST WRITE CONTROL B

047A 22BC00 SHLD DMASA ;

JR HRWO ;

047D+1810 DB 018H,HRWO-$-l ;---- FAKE JR INSTRUCTION -

READHST:

047F 3E01 MVI A,001H ;SETUP DMA FOR READ

0481 32CE00 STA DMAS3F ;

0484 3E04 MVI A,004H ;READ COMMAND

0486 32FA0A STA CMD ;SAVE FOR LATER

0489 219C0E LXI H,HSTBUF ;READ ONLY DATA BYTES

048C 22BC00 SHLD DMASA ;

HRWO:

048F 3E05 MVI A,05 ;FIVE RETRIES

0491 32020B STA T$RETRIES ;SETUP TEMPORARY RETRIES CO

0494 3EFF MVI A,0FFH ;INIT TOGGLE SO THAT NO HOM

0496 32030B STA HOME$TOGGLE ;ALTERNATE RETRIES WILL BE

;OTHER RETRIES WILL BE DONE

HRW1:

0499 3AED0A LDA HSTSEC ;HOST SECTOR NUMBER

049C 32B100 STA SECTNO ;SAVE SECTOR NUMBER

049F 3AEA0A LDA HSTDSK ;PICKUP DRIVE ID FOR SELECT

04A2 CD5205 CALL DSKSEL ;SELECT CORRECT DRIVE FOR I

04A5 CDB305 CALL POINT ;POINT TO TRACK REGISTER SA

04A8 EB XCHG ;POINT TO SELECT MASK

04A9 3EF0 MVI A,11110000B ;TO REMOVE CURRENT HEAD SEL

04AB A6 ANA M ;

04AC 77 MOV M,A ;

04AD E5 PUSH H ;SAVE MASK ADDRESS

04AE CD3205 CALL SETHED ;COMPUTE CORRECT HEAD NUMBE

04B1 7D MOV A,L ;TRACK NUMBER AFTER HEAD CA

04B2 32AD00 STA TRAKNO ;

04B5 E1 POP H ;RESTORE MASK ADDRESS

04B6 3AAE00 LDA HEADNO ;TO OR IN NEW HEAD NUMBER

04B9 B6 ORA M

04BA 77 MOV M,A ;SAVE NEW DRIVE/HEAD SELECT

04BB E67F ANI 07FH ; MASK OFF LARGE DRIVE FLAG

04BD D320 OUT 020H ;WRITE IT TO SELECT NEW HEA

04BF 0E01 MVI C,1 ;DELAY FOR 1 MILLISECOND

04C1 CD8207 CALL DELAY ;

HRW2:

04C4 CDB305 CALL POINT ;IS A SEEK NECESSARY ??

04C7 3AAD00 LDA TRAKNO ;CHECK

04CA BE CMP M ;WELL ??

JRZ HRW5 ;NO SEEK NECESSARY

04CB+2814 DB 028H,HRW5-$-l ;---- FAKE JRZ INSTRUCTION

HRW3:

04CD D322 OUT 022H ;WRITE NEW TRACK NUMBER

04CF 46 MOV B,M ;SAVE TEMPORARILY

04D0 77 MOV M,A ;UPDATE TRACK REGISTER SAVE

04D1 78 MOV A,B ;OLD TRACK NUMBER

04D2 D321 OUT 021H ;TO OLD TRACK REGISTER

04D4 3E10 MVI A,010H ;SEEK COMMAND

04D6 CD2107 CALL INTFIX ;CLEAR ANY PENDING INTERRUP

;AND ISSUE COMMAND

04D9 CD1707 HRW4: CALL WAIT0 ;WAIT FOR I/O

04DC 0E14 MVI C,20 ;DELAY AFTER SEEK FOR 20 MILLISECONDS

04DE CD8207 CALL DELAY

HRW5:

04E1 3AB100 LDA SECTNO ;SET SECTOR

04E4 D321 OUT 021H ;

HRW6:

04E6 21BA00 LXI H,DMAS1 ;SETUP DMA FOR HARD DISK I/O

04E9 010006 LXI B,0600H ;

OUTIR ;

04EC+EDB3 DB 0EDH,0B3H ;---- FAKE OTIR INSTRUCTION

04EE 21C000 LXI H,DMAS2H ;

04F1 010006 LXI B,0600H ;

OUTIR ;

04F4+EDB3 DB OEDH,OB3H ;---- FAKE OTIR INSTRUCTION

04F6 21CA00 LXI H,DMAS3 ;

04F9 010007 LXI B,0700H ;

OUTIR ;

04FC+EDB3 DB OEDH,0B3H ;---- FAKE OTIR INSTRUCTION

04FE 3AFA0A LDA CMD ;PICKUP I/O COMMAND

0501 CD2107 CALL INTFIX ;CLEAR ANY PENDING INTERRUP

;AND ISSUE COMMAND

0504 CD1707 HRW7: CALL WAIT0 ;WAIT FOR COMPLETION

0507 3E5D MVI A,01011101B ;SETUP STATUS AND MASK

0509 32FB0A STA MASK ;SAVE FOR STATUS CHECK

050C CDAE06 CALL CHECK$STAT ;CHECK STATUS FROM 1/0

050F C8 RZ ;OK ??

0510 3AO30B LDA HOME$TOGGLE ;

0513 2F CMA ;CHANGE TOGGLE SO THAT HOME

0514 32030B STA HOME$TOGGLE ;

JR HRW1 ;RETRY I/O

0517+1880 DB 018H,HRW1-$-l ;---- FAKE JR INSTRUCTION -

ENDIF

PAGE

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

;

; DOUBLE SIDED TRACK REGISTER UPDATE ROUTINE

;

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

DBL$UPDATE:

0519 3AB600 LDA MPARMS ;CHECK FOR DOUBLE SIDED DRIVE

051C E601 ANI 1 ; IS FLAG SET

051E C8 RZ ; NO -- SO RETURN

051F 3AAC00 LDA DISKNO ;CURRENT DISK DRIVE

0522 FE04 CPI 004H ;IS IT A FLOPPY

0524 D0 RNC ;NO, RETURN WITHOUT UPDATE

0525 E602 ANI 00000010B ;IS THIS DRIVE 2 OR 3 ??

0527 7E MOV A,M ;WE WERE CALLED WITH (HL) P

JRZ DBL$LOW ;IT MUST BE DRIVE ZERO OR 0

0528+2804 DB 28H,DBL$LOW-$-1 ;---- FAKE JRZ INSTRUCTION

052A 2B DCX H ;BACKUP TO OTHER SIDE POINT

052B 2B DCX H ;

JR DBL$SAVE ;

052C+1802 DB 18H,DBL$SAVE-$-1 ;---- FAKE JR INSTRUCTION

DBL$LOW:

052E 23 INX H ;BUMP UP TO DRIVE TWO OR TH

052F 23 INX H

DBL$SAVE:

0530 77 MOV M,A ;UPDATE OTHER SIDE REGISTER

0531 C9 RET

PAGE

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

;

; ROUTINE TO COMPUTE HEAD NUMBER FROM TRACK NUMBER

; TRACK NUMBER IS IN HL ON ENTRY

;

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

IF HARDSK

SETHED:

0532 2AEB0A LHLD HSTTRK ;CP/M TRACK NUMBER (0-800)

0535 E680 ANI 80H ; CHECK FOR LARGE DRIVE

0537 7D MOV A,L ;LOW ORDER

JRZ SETH14 ; SMALL DRIVE

0538+2806 DB 028H,SETH14-$-1 ;---- FAKE JRZ INSTRUCTION

053A E607 ANI 00000111B ;GET TRACK MOD 8 (HEAD NUMB

053C 0E03 MVI C,3 ;LIMIT LOOP FOR DIVIDE BY E

JR SETDVD ;

053E+1804 DB 018H,SETDVD-$-1 ;---- FAKE JR INSTRUCTION -

0540 E603 SETH14: ANI 00000011B ;GET TRACK MOD 4 (HEAD NUMB

0542 0E02 MVI C,2 ;LIMIT LOOP FOR DIVIDE BY F

0544 32AE00 SETDVD: STA HEADNO ;SAVE AS HEAD NUMBER

0547 B7 SHD1: ORA A ;ENSURE CARRY IS ZERO

0548 7C MOV A,H ;FOR SHIFT

0549 1F RAR ;ONE BIT

054A 67 MOV H,A ;

054B 7D MOV A,L ;LOW ORDER

054C 1F RAR ;CARRY PARTICIPATES FROM HI

054D 6F MOV L,A

054E 0D DCR C ;END OF DIVIDE YET ??

JRNZ SHD1 ;NO, CONTINUE

054F+20F6 DB 020H,SHD1-$-1 ;---- FAKE JRNZ INSTRUCTION

0551 C9 RET ;RETURN TO CALLER, TRACK IN

ENDIF

PAGE

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

;

; DISK DRIVE SELECT ROUTINE

; ON ENTRY, THE ACCUMULATOR CONTAINS THE DRIVE

; RETURNS CARRY SET FOR HARD DISK SELECTED

; RETURNS CARRY RESET FOR FLOPPY DISK SELECTE

;

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

DSKSEL:

0552 FE04 CPI 004H ;IS IT HARD DISK ??

JRNC SELHARD ;YES, GO PROCESS ....

0554+3045 DB 30H,SELHARD-$-1 ;---- FAKE JRNC INSTRUCTION

SELSOFT:

0556 21AC00 LXI H,DISKNO ;CURRENT DRIVE NUMBER

0559 BE CMP M ;SAME DRIVE AS LAST TIME ??

JRZ SLS3 ;YES, DONT BOTHER WITH UNLO

055A+2819 DB 028H,SLS3-$-1 ; ---- FAKE JRZ INSTRUCTION

055C 77 MOV M,A ;UPDATE WITH CURRENT DRIVE

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

;

; WE WILL NOW FORCE THE HEAD TO UNLOAD PRIOR TO THE S

; TO ENSURE THAT WHEN WE RETURN TO THIS DISK WE WILL

; LOAD AND WAIT FOR THE HEAD TO SETTLE.

;

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

SLS1:

055D DB04 IN 004H ;ENSURE FLOPPY PORT NOT BUS

055F 1F RAR ;

JRC SLS1 ;

0560+38FB DB 038H,SLS1-$-1 ;---- FAKE JRC INSTRUCTION

0562 DB05 IN 005H ;READ THE TRACK REGISTER

0564 D307 OUT 007H ;ENSURE WE DONT MOVE THE HE

0566 3E12 MVI A,012H ;SEEK AND UNLOAD HEAD

0568 CD6307 CALL FINTFIX ;CLEAR ANY PENDING INTERRUPT

; AND ISSUE COMMAND

056B CD3A07 SLS2: CALL FPYWAIT ;WAIT HERE FOR INTERRUPT

056E 3AFC0A LDA STATUS ;HOW DID THE I/O GO?

0571 E698 ANI 10011000B ; CHECK

JRNZ SLSERR ;EXIT IF ERROR

0573+2020 DB 020H,SLSERR-$-l ;---- FAKE JRNZ INSTRUCTION

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

;

; WE WILL NOW LOAD THE SELECT MASK AND SELECT THE DRIVE

; EVEN IF ITS THE SAME DRIVE BECAUSE THE DENSITY MAY

; HAVE CHANGED.

;

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

SLS3:

0575 CDB305 CALL POINT ;POINT TO TRACK SAVE AREA

0578 EB XCHG ;POINT TO SELECT MASK

0579 3AAD00 LDA TRAKNO ;NEXT TRACK FOR 1/0

057C FE02 CPI 002H ;IS IT TRACK ZERO OR ONE

057E 3EFF MVI A,11111111B ;ASSUME NO ....

JRNC SLS4 ;VERIFY ASSUMPTION

0580+3002 DB 030H,SLS4-$-1 ; ---- FAKE JRNC INSTRUCTION

0582 3EFE MVI A,11111110B ;FORCE SINGLE DENSITY FOR 0

SLS4:

0584 A6 ANA M ;LOAD MASK AND CORRECT IF N

0585 D308 OUT 008H ;SELECT IT

0587 DB04 IN 004H ;IS DRIVE READY?

0589 17 RAL ;

JRC SLSERR ;IF NOT .... BRANCH

058A+3809 DB 038H,SLSERR-$-1 ;---- FAKE JRC INSTRUCTION

058C EB XCHG ;RESTORE TRACK REGISTER ADD

058D 7E MOV A,M ;PICK UP TRACK NUMBER

058E D305 OUT 005H ;GIVE IT TO CONTROLLER

0590 AF XRA A ;ENSURE CARRY IS RESET

0591 32F60A STA ERFLAG ;ALSO ZERO ERROR INDICATOR

0594 C9 RET

0595 AF SLSERR: XRA A ;ENSURE CARRY IS RESET

0596 3C INR A ;SET TO 1 FOR ERROR FLAG

0597 32F60A STA ERFLAG ;SHOW ERROR

059A C9 RET

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

;

; THIS ROUTINE SETS UP THE HARD DISK BY SELECTING THE

; DRIVE AND RELOADING THE HEAD AND TRACK REGISTERS IN

; HARD DISK CONTROLLER READY FOR 1/0 LATER.

;

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

SELHARD:

IF HARDSK

059B 21AC00 LXI H,DISKNO ;CURRENT DRIVE SELECTED

059E BE CMP M ;SAME ??

059F C8 RZ ;YES, NO NEW SELECT NECESSA

05AO 77 MOV M,A ;UPDATE DISKNO

SLH1:

05AI CDB305 CALL POINT ;TRACK SAVE REGISTER

05A4 EB XCHG ;POINT TO SELECT MASK

05A5 7E MOV A,M ;LOAD DRIVE/HEAD VALUE

05A6 D320 OUT 020H ;WRITE IT TO SELECT PORT

05A8 EB XCHG ;REGAIN ADDRESS OF TRACK RE

05A9 7E MOV A,M ;LOAD OLD TRACK NUMBER

05AA D322 OUT 022H ;WRITE IT TO OLD TRACK REGI

05AC OE14 MVI C,20 ;DELAY FOR 20 MILLISECONDS

05AE CD8207 CALL DELAY ;

05B1 37 STC ;SET CARRY TO SHOW HARD DIS

ENDIF

05B2 C9 RET ;RETURN TO CALLER

PAGE

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

;

; SUBROUTINE TO POINT TO CURRENT TRACK REGISTER SAVE

;

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

POINT:

05B3 2AAC00 LHLD DISKNO ;PICKUP CURRENT DISK

05B6 7D MOV A,L ;

05B7 2600 MVI H,O ;RESET HIGH ORDER HALF

05B9 117000 LXI D,TRKO ;LOAD TRACK POINTER

05BC 19 DAD D ;POINT TO CURRENT TRACK PTR

05BD 54 MOV D,H ; DE = TRACK

05BE 5D MOV E,L ;

05BF 010C00 LXI B,12 ;

05C2 09 DAD B ; HL = SELECT

IF HARDSK

05C3 FE04 CPI 4 ;

JRC PNTFN ; FLOPPY DISK

05C5+380D DB 038H,PNTFN-$-1 ;---- FAKE JRC INSTRUCTION

05C7 3E10 MVI A,10H ;

05C9 A6 ANA M ; CHECK DRIVE SELECT

JRZ PNTH2 ; MUST BE DRIVE # 2

05CA+2805 DB 028H,PNTH2-$-1 ;---- FAKE JRZ INSTRUCTION

05CC 11B800 LXI D,HTK1 ; POINT TO DRIVE 1

JR PNTFN ;

05CF+1803 DB 018H,PNTFN-$-1 ;---- FAKE JR INSTRUCTION -

05D1 11B900 PNTH2: LXI D,HTK2 ; POINT TO DRIVE 2

ENDIF

05D4 EB PNTFN: XCHG ; SWITCH

05D5 C9 RET ; HL = TRACK DE = SELECT

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

;

; ROUTINE TO TRANSLATE SECTOR NUMBER

;

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

SECTRAN:

05D6 EB XCHG ;TABLE ADDRESS IS IN DE (NO

05D7 7C MOV A,H ;IS THERE A TABLE ADDRESS ?

05DB B5 ORA L ;

JRZ STRN2 ;NO, JUST RETURN ENTERED QU

05D9+2807 DB 028H,STRN2-$-1 ;---- FAKE JRZ INSTRUCTION

STRN1:

05DB 0600 MVI B,000H ;ENSURE OK FOR SINGLE BYTE

05DD 09 DAD B ;ADD SECTOR NUMBER

05DE 6E MOV L,M ;LOAD TRANSLATED VALUE

05DF 2600 MVI H,000H ;

05E1 C9 RET ;NEW VALUE RETURNED IN HL

STRN2:

05E2 09 DAD B ;RETURN SAME VALUE AS ENTER

05E3 C9 RET

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

;

; ROUTINES TO DO FLOPPY I/O

;

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

READSOFT:

05E4 3E9F MVI A,09FH ;MASK FOR READ STATUS

05E6 32FB0A STA MASK ;

05E9 3E01 MVI A,001H ;SETUP DMA FOR READ

05EB 32CE00 STA DMAS3F ;

05EE 3E8C MVI A,08CH ;READ COMMAND

JR SRW1

05FO+180F DB 018H,SRW1-$-1 ;---- FAKE JR INSTRUCTION

WRITESOFT:

05F2 3EFF MVI A,0FFH ;MASK FOR WRITE STATUS

05F4 32FB0A STA MASK ;

05F7 CD6B0E CALL MVDTB ;

05FA 3E05 MVI A,005H ;SETUP DMA FOR WRITE

05FC 32CE00 STA DMAS3F ;

05FF 3EAC MVI A,0ACH ;WRITE COMMAND

SRW1:

0601 32FAOA STA CMD ;

0604 211D13 LXI H,FPYBUF ;

0607 22BC00 SHLD DMASA ;

060A 3AE60A LDA NEWDSK ;

060D CD5205 CALL DSKSEL ;SELECT DRIVE FOR I/O

0610 3AF60A LDA ERFLAG ;CHECK FOR SELECT ERROR

0613 B7 ORA A ;

0614 C0 RNZ ;RETURN IF ERROR

SRW2:

0615 3EOA MVI A,10 ;SET NUMBER OF TRIALS

0617 32020B STA T$RETRIES ;SAVE FOR RETRY ROUTINE

061A AF XRA A ;

061B 32030B STA HOME$TOGGLE ;FORCE HOME PRIOR TO EACH R

LOAD$HEAD:

061E DB08 IN 008H ;IS HEAD LOADED ??

0620 E602 ANI 00000010B ;CHECK IT ....

JRNZ REMOVE$LD ;YES, ITS LOADED, DONT RELO

0622+201F DB 20H,REMOVE$LD-$-1 ;---- FAKE JRNZ INS

0624 DB05 IN 005H ;DUMMY SEEK TO START HEAD L

0626 D307 OUT 007H ;KEEP IT SHORT ....

0628 3ElA MVI A,01AH ;START HEAD LOADING

062A CD6307 CALL FINTFIX ;CLEAR ANY PENDING INTERRUP

;AND ISSUE COMMAND

062D CD3AO7 LDH1: CALL FPYWAIT ;WAIT FOR I/O TO COMPLETE

0630 3AFCOA LDA STATUS ;HOW DID IT GO?

0633 E698 ANI 10011000B ;CHECK

JRNZ CHECKIT ;DO NOT GO ON IF ERROR

0635+2044 DB 20H,CHECKIT-$-1 ;---- FAKE JRNZ INS

0637 OE10 MVI C,16 ;WAIT HERE FOR 16 MS

0639 CD8207 CALL DELAY ;CALL WAIT ROUTINE

063C CDB305 CALL POINT ;REESTABLISH TRACK REGISTER

063F 36FE MVI M,254 ;ENSURE FURTHER SEEK AND DE

JR TRKTST ;

0641+1807 DB 018H,TRKTST-$-1 ;---- FAKE JR INSTRUCTION -

REMOVE$LD:

0643 21FA0A LXI H,CMD ;POINT TO I/O COMMAND

0646 3EFB MVI A,11111011B ;REMOVE HEAD LOAD BIT

0648 A6 ANA M ;DO IT ....

0649 77 MOV M,A ;SAVE IT BACK INTO CMD

TRKTST:

064A CDB305 CALL POINT ;RESTORE TRACK REGISTER POI

064D 3AE70A LDA NEWTRK ;GET NEW TRACK NUMBER

0650 32AD00 STA TRAKNO ;SAVE IN COMMON PLACE

0653 BE CMP M ;SAME AS LAST TIME ??

JRZ FSECSET ;YES, DONT BOTHER WITH SEEK

0654+281A DB 28H,FSECSET-$-1 ;---- FAKE JRZ INSTRUCTION

0656 77 MOV M,A ;SAVE IT

0657 D307 OUT 007H ;ALSO SEND IT TO CONTROLLER

0659 CD1905 CALL DBL$UPDATE ;DOUBLE SIDED SUPPORT

FLOPPY$SEEK:

065C 3E1A MVI A,01AH ;SEEK COMMAND WITH HEAD LOA

065E CD6307 CALL FINTFIX ;CLEAR ANY PENDING INTERRUP

; ;AND ISSUE COMMAND

0661 CD3A07 FPS1: CALL FPYWAIT ;WAIT FOR I/O TO COMPLETE

0664 3AFC0A LDA STATUS ;HOW DID IT GO?

0667 E698 ANI 10011000B ;CHECK

JRNZ CHECKIT ;DO NOT GO ON IF ERROR

0669+2010 DB 20H,CHECKIT-$-1 ;---- FAKE JRNZ INSTRUCTION

066B 0E10 MVI C,16 ;SET FOR 16 MS DELAY

066D CD8207 CALL DELAY

FSECSET:

0670 3AE90A LDA NEWSEC ;SET SECTOR

0673 32B100 STA SECTNO ;SAVE IN COMMONN PLACE

0676 D306 OUT 006H ;

0678 CD8706 CALL FLOPPYIO ;DO I/O

CHECKIT:

067B CDAE06 CALL CHECK$STAT ;CHECK STATUS OF I/O

067E 3AF60A LDA ERFLAG ;SETUP TO RETURN TO BDOS

0681 CC7E0E CZ MVDFB ;

0684 C8 RZ ;EITHER OK OR PERMANENT ERROR

JR LOAD$HEAD ;ERROR, JUST RETRY THIS SAM

0685+1897 DB 18H,LOAD$HEAD-$-1 ;---- FAKE JR INSTRUCTION

PAGE

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

;

; THIS IS THE ROUTINE THAT DOES THE FLOPPY DISK I/O

;

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

FLOPPYIO:

IF NOT DMA

LXI H,066H ;MOVE DATA FROM 066H TO SAV

LXI D,SAVE1 ;

LXI B,004H ;

LDIR ;MOVE IT

LXI H,NMIRTN ;SET NMI ROUTINE TO NMI ADD

LXI D,066H ;

LXI B,004H ;

LDIR ;MOVE IT

LDA CMD ;IS IT A WRITE ??

ANI 20H ;

JZ FRD ;NO, LEAVE INI CMD IN LOW M

LXI H,067H ;POINT TO COMMAND AREA

MVI M,0A3H ;MAKE IT AN OTI CMD

FRD EQU $ ;LABEL

ENDIF

IF DMA

0687 21BA00 LXI H,DMAS1 ;INITIALIZE DMA

068A 010006 LXI B,0600H ;

OUTIR ;WRITE TO DMA

068D+EDB3 DB 0EDH,0B3H ;---- FAKE OTIR INSTRUCTION

068F 21C600 LXI H,DMAS2F ;

0692 010004 LXI B,0400H ;

OUTIR ;WRITE TO DMA

0695+EDB3 DB 0EDH,0B3H ;---- FAKE OTIR INSTRUCTION

0697 21CAOO LXI H,DMAS3 ;

069A 010007 LXI B,0700H ;

OUTIR ;WRITE TO DMA

069D+EDB3 DB 0EDH,0B3H ;---- FAKE OTIR INSTRUCTION

ENDIF

069F OE07 MVI C,007H ;PORT ADDRESS FOR I/O

06A1 211D13 LXI H,FPYBUF ;DMA ADDRESS

06A4 3AFAOA LDA CMD ;I/O COMMAND

06A7 CD6307 CALL FINTFIX ;CLEAR ANY PENDING INTERRUPT

; ;AND ISSUE COMMAND

06AA CD3AO7 FWT1: CALL FPYWAIT ;WAIT HERE FOR I/O TO COMPLETE

IF NOT DMA

LXI H,SAVE1 ;SETUP TO REPLACE DATA

LXI D,066H ;COPIED FROM NMI LOCATION

LXI B,004H

LDIR ;MOVE IT ....

ENDIF

06AD C9 RET ;RETURN, I/O COMPLETED

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

;

; WE WILL NOW CHECK THE STATUS OF THE I/O OPERATION

; RETURN WITH CONDITION CODE ZERO = NO RETRY

; RETURN WITH CONDITION CODE NON ZERO = RETRY

;

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

CHECK$STAT:

06AE 21F60A LXI H,ERFLAG ;POINT TO ERROR INDICATOR

06B1 3600 MVI M,000H ;ASSUME OK

06B3 21FC0A LXI H,STATUS ;CHECK STATUS

06B6 3AFB0A LDA MASK ;MASK FOR UNWANTED BIT REMO

06B9 A6 ANA M ;

06BA 77 MOV M,A ;SAVE CLEANED STATUS

06BB C8 RZ ;OK, SO RETURN

CHKS0:

06BC CDEE02 CALL RETMOD ;

06BF FE03 CPI 003H ;HARD DISK ??

06CI 21FC0A LXI H,STATUS ;

06C4 7E MOV A,M ;RELOAD STATUS BYTE

JRNC CHKS2 ;YES, CHECK FOR DRIVE READY

06C5+3006 DB 030H,CHKS2-$-1 ;---- FAKE JRNC INSTRUCTION

CHKS1:

06C7 FE80 CPI 080H ;IS FLOPPY DISK NOT READY ?

JRZ BADIO ;YES, DONT BOTHER WITH RETRY

06C9+283D DB 028H,BADIO-$-1 ;---- FAKE JRZ INSTRUCTION

JR CHKS3 ;GO TO BAD MESSAGE ROUTINE

06CB+1819 DB 018H,CHKS3-$-1 ;---- FAKE JR INSTRUCTION

CHKS2:

06CD FE00 CPI 000H ;IS HARD DISK NOT READY ??

JRZ BADIO ;YES, BYPASS ERROR MESSAGE

06CF+2837 DB 28H,BADIO-$-1 ;---- FAKE JRZ INSTRUCTION

06D1 E640 ANI 01000000B ;IS IT WRITE FAULT ??

JRZ CHKS3 ;NO, CONTINUE ON

06D3+2811 DB 28H,CHKS3-$-1 ;---- FAKE JRZ INSTRUCTION

06D5 CDB305 CALL POINT ;POINT TO TRACK REGISTER

06D8 EB XCHG ;POINT TO SELECT MASK

06D9 7E MOV A,M ;

06DA F640 ORI 01000000B ;TURN ON WRITE FAULT CLEAR

06DC D320 OUT 020H ;

06DE 7E MOV A,M ;RESET CLEAR

06DF D320 OUT 020H ;

06E1 0E14 MVI C,20 ;DELAY JUST TO BE SAFE

06E3 CD8207 CALL DELAY ;

CHKS3:

06E6 3A030B LDA HOME$TOGGLE

06E9 B7 ORA A ;IS A HOME NEEDED ON THIS R

JRNZ CHKS4 ;

06EA+200B DB 020H,CHKS4-$-1 ;---- FAKE JRNZ INSTRUCTION

06EC 3AFCOA LDA STATUS ;SAVE STATUS OVER HOME

06EF F5 PUSH PSW ;

06FO CDlA03 CALL HOME ;RESET DEVICE TO HOME

06F3 Fl POP PSW ;

06F4 32FC0A STA STATUS ;SAVE FOR ERROR MESSAGE

CHKS4:

06F7 119400 LXI D,TCNT ;BUMP TEMP ERROR COUNTUER

06FA CD0F07 CALL ADDERRORS ;

06FD 21020B LXI H,T$RETRIES ;PICKUP RETRY COUNT

0700 35 DCR M ;DECREMENT COUNT OF RETRIES

0701 C0 RNZ ;

0702 11A000 LXI D,PCNT ;BUMP PERMANENT ERROR COUNTUER

0705 CD0F07 CALL ADDERRORS ;

BADIO:

0708 21F60A LXI H,ERFLAG ;SET PERMANENT ERROR

070B 3601 MVI M,001H ;DO IT ....

070D AF XRA A ;RESET TO PRECLUDE RETRIES

070E C9 RET ;RETURN TO CALLER

ADDERRORS:

070F 2AAC00 LHLD DISKNO ;BUMP COUNT OF DISK ERRORS

0712 2600 MVI H,000H ;

0714 19 DAD D ;POINT TO ERROR REGISTER

0715 34 INR M ;

0716 C9 RET ;

PAGE

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;

; THIS IS HARD DISK WAIT ENTRY

;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

WAIT0:

0717 C5 PUSH B ; SAVE RETRY COUNTER

0718 0E84 MVI C,FLAGWT ; FUNCTION FLAG WAIT

071A 1E05 MVI E,HDFLAG ; DEVICE IS HARD DISK

071C CD100B CALL XDOS

071F C1 POP B ; RESTORE RETRY COUNTER IN

; READ OR WRITE IS OK, ACCUMULATOR CONTAINS ZERO

0720 C9 RET

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;

; THE FOLLOWING CODE GUARANTEES THAT HARD DISK FLAG I

; IT APPEARS THAT WE OCCASIONALLY GET FLAG SET AS A R

; OF AN INTERRUPT FROM THE HARD DISK, WHEN WE DO

; NOT EXPECT IT.

;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

INTFIX:

0721 F5 PUSH PSW

0722 C5 PUSH B

0723 D5 PUSH D

0724 E5 PUSH H

0725 0E85 MVI C,FLAGST

0727 1E05 MVI E,HDFLAG

0729 CD100B CALL XDOS ;EITHER FLAG 5 WILL BE SET

; ;IT IS ALREADY SET - IN WHI

; ;THIS REQUEST WILL BE IGNOR

072C 0E84 MVI C,FLAGWT

072E 1E05 MVI E,HDFLAG

0730 CD100B CALL XDOS ;NOW CLEAR THE FLAG

0733 E1 POP H

0734 D1 POP D

0735 C1 POP B

0736 F1 POP PSW ;RESTORE REGISTERS

0737 D323 OUT 023H ;ISSUE COMMAND TO HARD DISK

0739 C9 RET

PAGE

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;

; THIS IS FLOPPY DISK WAIT ENTRY

;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

FPYWAIT:

073A C5 PUSH B ;SAVE RETRY COUNT

073B E5 PUSH H

073C 0E84 MVI C,FLAGWT ; FUNCTION IS FLAG WAIT

073E 1E06 MVI E,FPYFLAG ; WAIT FOR FLOPPY

0740 CD100B CALL XDOS

0743 F5 PUSH PSW

0744 3AD00D LDA FPYTIME ;DID WD1791 GO TO SLEEP?

0747 B7 ORA A

JRNZ NOPPYRST ;IF STILL AWAKE, SKIP RESET

0748+2015 DB 20H,NOFPYRST-$-1 ;---- FAKE JRNZ INSTRUCTION

074A DB09 IN 009H ;GET CURRENT BANK NUMBER

074C E618 ANI 00011000B ;REMOVE OTHER INFO

074E D309 OUT 009H ;RESET WD1791

0750 0E01 MVI C,1 ;DELAY 1 MILLISEC

0752 CD8207 CALL DELAY

0755 F602 ORI 00000010B ;END RESET

0757 D309 OUT 009H

0759 3AE60A LDA NEWDSK ;MAKE SURE CURRENT DISK AND

075C 32ACOO STA DISKNO ; THE SAME

NOFPYRST:

075F F1 POP PSW

0760 E1 POP H

0761 C1 POP B ;RESTORE RETRY COUNT IN <C>

0762 C9 RET

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;

; THE FOLLOWING CODE GUARANTEES THAT FLOPPY DISK FLAG

;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

FINTFIX:

0763 F5 PUSH PSW

0764 C5 PUSH B

0765 D5 PUSH D

0766 E5 PUSH H

0767 0E85 MVI C,FLAGST

0769 1E06 MVI E,FPYFLAG

076B CD100B CALL XDOS

076E 0E84 Mvi C,FLAGWT

0770 1E06 MVI E,FPYFLAG

0772 CD100B CALL XDOS

0775 210301 LXI H,00103H ;SET TIME OUT INDICATOR

0778 22D00D SHLD FPYTIME ; TIME TO BE BETWEEN 2

077B E1 POP H

077C D1 POP D

077D C1 POP B

077E Fl POP PSW

077F D304 OUT 004H ;ISSUE COMMAND TO FLOPPY

0781 C9 RET

if not mpm20

FPYTIME:

DW 0

FPYTCNT:

DW 0

endif

PAGE

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

;

; THIS IS THE DELAY ROUTINE. IT WILL LOOP HERE FOR TH

; NUMBER OF MILLISECONDS SPECIFIED IN REGISTER C.

;

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

DELAY:

0782 0664 DEL1: MVI B,100 ;FORCE DELAY FOR 1 MILLISEC

0784 00 DEL2: NOP ;INSTRUCTIONS TO FILL IN TI

0785 29 DAD H ;

0786 29 DAD H ;

0787 05 DCR B ;AT ONE MILLISECOND YET

0788 C28407 JNZ DEL2 ;NO, KEEP ON LOOPING

078B 0D DCR C ;END OF REQUESTED INTERI

078C C28207 JNZ DEL1 ;NO, KEEP ON

078F C9 RET ;RETURN TO CALLER

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

;\* NOTE: THE INITIALIZATION CODE WILL BE

;\* OVERWRITTEN BY DIRBUF & FPYBUF

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

if not mpm20

DIRBUF EQU $

endif

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

;

; DISK CONFIGURATION TABLE

;

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

IF HARDSK

; ; PIN C

0790 0000000000DSCNO: DB 00H,00H,00H,00H,00H,00H,00H,00H

0798 1000000000 DB 10H,00H,00H,00H,00H,00H,10H,00H

07AO 9090900000 DB 90H,90H,90H,00H,00H,00H,00H,00H

07A8 0000000000 DB 00H,00H,00H,00H,00H,00H,00H,00H

07BO 1000002000 DB 10H,00H,00H,20H,00H,00H,10H,20H

07B8 0000000000 DB 00H,00H,00H,00H,00H,00H,00H,00H

07CO 9090902000 DB 90H,90H,90H,20H,00H,00H,00H,20H

07C8 909090AOAO DB 90H,90H,90H,0A0H,0A0H,0A0H,0H,0H

ENDIF

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

;

; SET UP DISK CONFIGURATON

;

; [ THIS CODE EXECUTED ONLY ONCE ]

;

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

07DO 217EOO SDCONF: LXI H,SEL0+2 ;POINT TO DRIVE C:

07D3 3AB600 LDA MPARMS ;

07D6 E605 ANI 05H ; TEST FOR FOUR FLOPPIES

07D8 C3DE07 JMP SDDBL ; YES SKIP THE ZAP

07DB 77 MOV M,A ;

07DC 23 INX H ; ZAP C: AND D:

07DD 77 MOV M,A ;

SDDBL:

07DE 118000 LXI D,SEL0+4 ;POINT TO DRIVE E:

IF HARDSK

07E1 DB25 IN 025H ;READ CONFIGURATION PORT

07E3 E607 ANI 07H ;STRIP OFF HIGH PART

07E5 17 RAL ;

07E6 17 RAL ;

07E7 17 RAL ;

07E8 0600 MVI B,0 ;

07EA 4F MOV C,A ;POINT TO CONFIGURATION TAB

07EB 219007 LXI H,DSCN0 ;

07EE 09 DAD B ; INDEX TO RIGHT ENTRY

07EF 0608 MVI B,8 ;

07F1 7E SDL1: MOV A,M ; CHANGE ALL SELECT MASKS

07F2 12 STAX D ;

07F3 13 SDOK: INX D ; NEXT

07F4 23 INX H ; DRIVE

DJNZ SDL1 ;

07F5+10FA DB 010H,SDL1-$-1 ;---- FAKE DJNZ INSTRUCTION

ENDIF

IF NOT HARDSK

XCHG ;

MVI B,8 ;

XRA A ;

SDL2: MOV M,A ; ZAP ALL HARD DRIVES

INX H ;

DJNZ SDL2 ;

ENDIF

07F7 C9 RET

07F8 = INITEND EQU $

07F8 E5 XETMOD: PUSH H ;SAVE MODE BYTE ADDRESS

07F9 79 MOV A,C ;SETUP FOR DENSITY CHANGE

07FA B7 ORA A ;

07FB 0EO0 MVI C,000H ;ASSUME SINGLE DENSITY MODE

JRZ XETSEL ;VERIFY ASSUMPTION

07FD+2802 DB 028H,XETSEL-$-1 ;---- FAKE JRZ INSTRUCTION

07FF 0E01 MVI C,001H ;SET FOR DOUBLE DENSITY MOD

0801 CD7802 XETSEL: CALL SETDEN ;SET DENSITY BASED ON LOW B

0804 El POP H ;RESTORE

0805 6E MOV L,M ;PICKUP MODE AGAIN

0806 2600 MVI H,000H ;FOR SINGLE BYTE PRECISION

0808 7D MOV A,L ;SAVE MODE IN ACCUMULATOR F

0809 29 DAD H ;\* 2

080A 29 DAD H ;\* 4

080B E5 PUSH H ;SAVE \* 4

080C 29 DAD H ;\* 8

080D D1 POP D ;REGAIN \* 4

080E 19 DAD D ;\* 12

080F 119101 LXI D,MODL0 ;FIRST MODEL DPE

0812 19 DAD D ;POINT TO THIS ONE

0813 EB XCHG ;SETUP TEMPORARILY AS DESTI

0814 2AB200 LHLD DPEPTR ;ADDRESS OF CURRENTLY SELEC

0817 EB XCHG ;SETUP TO ALTER

0818 010C00 LXI B,12 ;LENGTH FOR MOVE

LDIR ;DO MOVE

081B+EDB0 DB 0EDH,0B0H ;---- FAKE LDIR INSTRUCTION

081D C9 RET ;RETURN TO CALLER

PAGE

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

;

; THE FOLLOWING AREA CONTAINS THE DISK/WORK SAVE AREA

; USED BY THE CBIOS IN THE NORMAL COURSE OF ACTIVITY.

;

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

if mpm20

;tempbuf equ (dirbuf-base)+128

else

TEMPBUF EQU (DIRBUF-BASE)+256

ORG TEMPBUF+((INITEND-BASE)/TEMPBUF)\*((INITEND-BASE

endif

081E BEGDAT EQU $ ;START OF BDOS AREA

;DIRBUF: DS 128 ;OVERLAYS SYSTEMINIT CODE

081E ALV0: DS 32

083E CSV0: DS 32

085E ALV1: DS 32

087E CSVl: DS 32

089E ALV2: DS 32

08BE CSV2: DS 32

08DE ALV3: DS 32

08FE CSV3: DS 32

IF HARDSK

091E ALV4: DS 64

095E CSV4: DS 0

095E ALV5: DS 64

099E CSV5: DS 0

099E ALV6: DS 64

09DE CSV6: DS 0

09DE ALV7: DS 64

0A1E CSV7: DS 0

0A1E ALV8: DS 64

0A5E CSV8: DS 0

0A5E ALV9: DS 64

0A9E CSV9: DS 0

0A9E ALVA: DS 36

0AC2 CSVA: DS 0

0AC2 ALVB: DS 36

0AE6 CSVB: DS 0

endif

if mdisk

ALVC: DS 32 ;VIRTUAL DISK

CSVC: DS 0

endif

if not mpm20

if hardsk

DS 1 ;MUST PRECEDE HSTBU

HSTBUF: DS 1024 ;HOST BUFFER AREA

DS 1 ;MUST FOLLOW HSTBUF

ENDIF

FPYBUF EQU DIRBUF+128 ; FLOPPY I/O BUFFER

endif

0AE6 NEWDSK: DS 1 ;SEEK DISK NUMBER

0AE7 NEWTRK: DS 2 ;SEEK TRACK NUMBER

0AE9 NEWSEC: DS 1 ;SEEK SECTOR NUMBER

0AEA HSTDSK: DS 1 ;HOST DISK NUMBER

0AEB HSTTRK: DS 2 ;HOST TRACK NUMBER

0AED HSTSEC: DS 1 ;HOST SECTOR NUMBER

0AEE NEWHST: DS 1 ;SEEK SHR SECSHF

0AEF HSTACT: DS 1 ;HOST ACTIVE FLAG

0AF0 HSTWRT: DS 1 ;HOST WRITTEN FLAG

0AF1 UNACNT: DS I ;UNALLOCATED RECORD

0AF2 UNADSK: DS 1 ;LAST UNALLOCATED D

0AF3 UNATRK: DS 2 ;LAST UNALLOCATED T

0AF5 UNASEC: DS 1 ;LAST UNALLOCATED S

0AF6 ERFLAG: DS 1 ;ERROR REPORTING

0AF7 RSFLAG: DS 1 ;READ SECTOR FLAG

0AF8 READOP: DS 1 ;1 IF READ OPERATIO

0AF9 WRTYPE: DS 1 ;WRITE OPERATION TY

0AFA 00 CMD: DB 0 ;COMMANDS FOR NEXT

0AFB 00 MASK: DB 0 ;STATUS MASKS BUFFE

0AFC 00 STATUS: DB 0 ;STATUS SAVE LOCATI

0AFD 00000000 SAVE1: DB 00H,00H,00H,00H ;SAVE AREA FOR NMI

0B01 00 P$RETRIES: DB 000H ;COUNTER FOR PERMAN

0B02 00 T$RETRIES: DB 000H ;COUNTER FOR TEMPOR

HOME$TOGGLE:

0B03 00 DB 000H ;INDICATOR TO TELL

; ;.. IF HOME SHOULD

page

if mpm20

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

;\*

;\* M P / M 2 . 0 C 0 M M 0 N B A S E

;\*

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

commonbase:

0B04 C3150B jmp coldstart

0B07 C30000 swtuser: jmp $-$

0B0A C30000 swtsys: jmp $-$

0B0D C30000 pdisp: jmp $-$

0B10 C30000 xdos: jmp $-$

0B13 0000 sysdat: dw $-$

COLDSTART:

WARMSTART:

0B15 0E00 MVI C,0 ; SEE SYSTEM INIT

; COLD & WARM START INCLUDE

; FOR COMPATIBILITY WITH CP

0B17 C3100B JMP XDOS ; SYSTEM RESET, TERMINATE P

rtnempty:

0B1A AF xra a

0B1B C9 ret

NULL$INT:

0B1C FB EI

RETI

0B1D+ED4D DB 0EDH,04DH ;---- FAKE RETI INSTRUCTION

endif

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

;

; CENTRONICS PRINTER ROUTINE (WITH SEPARATE BUSY TEST

;

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

CNSTAT:

0B1F 3E01 MVI A,001H ;TO SET STROBE HIGH

0B21 D310 OUT 010H

0B23 DB10 IN 010H ;READ PRINTER STATUS

0B25 E620 ANI 020H ;REMOVE ALL BUT BUSY BIT

0B27 3EFF MVI A,0FFH ;ASSUME NOT BUSY

0B29 C8 RZ ;CHECK ASSUMPTION

0B2A AF XRA A ;SET TO SHOW STILL BUSY

0B2B C9 RET ;

CLIST:

0B2C CD1F0B CALL CNSTAT ;IS PRINTER READY NOW?

0B2F B7 ORA A

JRNZ CLIST1 ;IF READY, SKIP POLL

0B30+2009 DB 020H,CLIST1-$-1 ;---- FAKE JRNZ INSTRUCTION

0B32 C5 PUSH B

0B33 0E83 MVI C,POLL ; POLL DEVICE

0B35 1E00 MVI E,PLLPT ; PRINTER

0B37 CD100B CALL XDOS ;WAIT FOR PRINTER TO FREE U

0B3A C1 POP B ;

CLIST1:

0B3B 79 MOV A,C ;CHARACTER TO PRINT

0B3C D311 OUT 011H ;WRITE IT TO DATA PORT

0B3E 3E00 MVI A,000H ;TO FORCE STROBE LOW

0B40 D310 OUT 010H ;

0B42 3E01 MVI A,001H ;TO FORCE STROBE HIGH

0B44 D310 OUT 010H ;

0B46 C9 RET ;

PAGE

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

;

; DISK INTERRUPT ROUTINE

;

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

FLOPPY$INT:

0B47 22C80D SHLD SVDHL

0B4A 21500B LXI H,FDINTH

0B4D C37F0D JMP INTINIT

FDINTH:

0B50 DB04 IN 004H ;GET STATUS

0B52 32FC0A STA STATUS ;SAVE FOR I/O ROUTINE

0B55 3E00 MVI A,0 ;STOP TIMING OF RESPONSE TO

0B57 32D10D STA FPYTIME+1 ;

0B5A 1E06 MVI E,FPYFLAG ;SHOW I/O COMPLETED

JR HDSTFLG

0B5C+1813 DB 18H,HDSTFLG-$-1 ;---- FAKE JR INSTRUCTION

HARD$INT:

0B5E 22C80D SHLD SVDHL

0B61 21670B LXI H,HDINTH

0B64 C37F0D JMP INTINIT

HDINTH:

0B67 DB24 IN 024H ;GET STATUS

0B69 32FCOA STA STATUS ;SAVE FOR CHECK LATER

0B6C AF XRA A

0B6D D323 OUT 023H ;RESET INTERRUPT BY RELOADING

0B6F 1E05 MVI E,HDFLAG ;SHOW I/O COMPLETED

HDSTFLG:

0B71 0E85 MVI C,FLAGST

0B73 CD100B CALL XDOS

0B76 C3670D JMP INTDONE

PAGE

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

;

; CONSOLE DISPLAY ROUTINES

;

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

;

CONST: ; CONSOLE STATUS

0B79 CD9A0B CALL PTBLJMP ; COMPUTE AND JUMP TO HANDLER

0B7C AD0B DW PT0ST ; CONSOLE #0 STATUS ROUTINE

0B7E EC0B DW PT1ST ; CONSOLE #1 STATUS ROUTINE

0B80 2B0C DW PT2ST ; CONSOLE #2 STATUS ROUTINE

0B82 6A0C DW PT3ST ; CONSOLE #3 STATUS ROUTINE

CONIN: ; CONSOLE INPUT

0B84 CD9A0B CALL PTBLJMP ; COMPUTE AND JUMP TO HANDLER

0B87 B80B DW PT0IN ; CONSOLE #0 INPUT

0B89 F70B DW PTlIN ; CONSOLE #1 INPUT

0B8B 360C DW PT2IN ; CONSOLE #2 INPUT

0B8D 750C DW PT3IN ; CONSOLE #3 INPUT

CONOUT: ; CONSOLE OUTPUT

0B8F CD9AOB CALL PTBLJMP ; COMPUTE AND JUMP TO HANDLER

0B92 CA0B DW PT0OUT ; CONSOLE #0 OUTPUT

0B94 090C DW PT1OUT ; CONSOLE #1 OUTPUT

0B96 480C DW PT2OUT ; CONSOLE #2 OUTPUT

0B98 870C DW PT3OUT ; CONSOLE #3 OUTPUT

;

PTBLJMP: ; COMPUTE AND JUMP TO HANDLER

; D = CONSOLE #

; DO NOT DESTROY <D>

0B9A 7A MOV A,D

0B9B FE04 CPI NMBCNS

JRC TBLJMP

0B9D+3803 DB 038H,TBLJMP-$-1 ;---- FAKE JRC INSTRUCTION

0B9F F1 POP PSW ; THROW AWAY TABLE ADDRESS

0BA0 AF XRA A

0BA1 C9 RET

TBLJMP: ; COMPUTE AND JUMP TO HANDLER

; A = TABLE INDEX

0BA2 87 ADD A ; DOUBLE TABLE INDEX FOR ADR OFFST

0BA3 E1 POP H ; RETURN ADR POINTS TO JUMP TBL

0BA4 5F MOV E,A

0BA5 1600 MVI D,0

0BA7 19 DAD D ; ADD TABLE INDEX \* 2 TO TBL BASE

0BA8 5E MOV E,M ; GET HANDLER ADDRESS

0BA9 23 INX H

0BAA 56 MOV D,M

0BAB EB XCHG

0BAC E9 PCHL ; JUMP TO COMPUTED CNS HANDLER

PAGE

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

;

; SERIAL PORT ADDRESS EQUATES

;

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

001C = DATA0 EQU 01CH ;CONSOLE #0 DATA

001D = STS0 EQU DATA0+1 ;CONSOLE #0 STATUS

002C = DATA1 EQU 02CH ;CONSOLE #1 DATA

002D = STS1 EQU DATA1+1 ;CONSOLE #1 STATUS

002E = DATA2 EQU 02EH ;CONSOLE #2 DATA

002F = STS2 EQU DATA2+1 ;CONSOLE #2 STATUS

002A = DATA3 EQU 02AH ;CONSOLE #3 DATA

002B = STS3 EQU DATA3+1 ;CONSOLE #3 STATUS

001E = LPTPRT0 EQU 01EH ;PRINTER #0 DATA

001F = LPTSTS0 EQU LPTPRTO+1 ;PRINTER #0 STATUS

0028 = LPTPRT1 EQU 028H ;PRINTER #1 DATA

0029 = LPTSTS1 EQU LPTPRT1+1 ;PRINTER #1 STATUS

PAGE

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

;

; POLL CONSOLE # 0 INPUT

;

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

POLCI0:

PT0ST: ; TEST CONSOLE STATUS

0BAD AF XRA A ; RETURN 0FFH IF READY

0BAE D31D OUT STS0 ; 000H IF NOT

0BB0 DBlD IN STS0 ;

0BB2 E601 ANI 1 ; RX CHAR ?

0BB4 C8 RZ ; NO

0BB5 3EFF MVI A,0FFH ; YES - SET FLAG

0BB7 C9 RET

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

;

; CONSOLE # 0 INPUT

;

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

;

PT0IN: ; RETURN CHAR IN REG A

0BB8 CDAD0B CALL POLCI0 ;IS IT READY NOW?

0BBB B7 ORA A ;

JRNZ PT0IN1 ;IF READY, SKIP POLL

0BBC+2007 DB 20H,PT0IN1-$-1 ;---- FAKE JRNZ INSTRUCTION

0BBE 0E83 MVI C,POLL ;

0BC0 1E05 MVI E,PLCI0 ; POLL CONSOLE #0 INPUT

0BC2 CD100B CALL XDOS ;

0BC5 DB1C PT0IN1: IN DATA0 ; READ CHARACTER

0BC7 E67F ANI 7FH ; STRIP PARITY

0BC9 C9 RET ;

;

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

;

; CONSOLE # 0 OUTPUT

;

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

;

PT0OUT: ; REG C = CHAR TO OUTPUT

0BCA CDDDOB CALL POLCO0 ;IS IT READY NOW?

0BCD B7 ORA A ;

JRNZ PT0OUT1 ;IF READY, SKIP POLL

0BCE+2009 DB 20H,PT0OUT1-$-l ;---- FAKE JRNZ INSTRUCTION

0BD0 C5 PUSH B ;

0BD1 OE83 MVI C,POLL ;

0BD3 lE01 MVI E,PLCO0 ;

0BD5 CD100B CALL XDOS ; POLL CONSOLE #0 OUTPUT

0BD8 Cl POP B

PT0OUT1:

0BD9 79 MOV A,C ;

0BDA D31C OUT DATA0 ; TRANSMIT CHARACTER

0BDC C9 RET ;

;

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

;

; POLL CONSOLE # 0 OUTPUT

;

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

;

POLCO0: ; RETURN 0FFH IF READY

0BDD 3E10 MVI A,10H ; 000H IF NOT

0BDF D31D OUT STS0 ; RESET INT BIT

0BE1 DBlD IN STS0 ; READ STATUS

0BE3 E60C ANI 0CH ; MASK FOR DTR AND TXE

0BE5 FE0C CPI 0CH ; MUST HAVE BOTH

0BE7 3E00 MVI A,0 ;

0BE9 C0 RNZ ; RETURN NOT READY

0BEA 3D DCR A ;CHANGE "A" TO 0FFH

0BEB C9 RET ; RETURN READY

PAGE

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

;

; POLL CONSOLE # 1 INPUT

;

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

POLCI1:

PT1ST: ; TEST CONSOLE STATUS

0BEC AF XRA A ; RETURN 0FFH IF READY

0BED D32D OUT STS1 ; 000H IF NOT

0BEF DB2D IN STS1 ;

0BF1 E601 ANI 1 ; RX CHAR ?

0BF3 C8 RZ ; NO

0BF4 3EFF MVI A,0FFH ; YES - SET FLAG

0BF6 C9 RET ;

;

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

;

; CONSOLE # 1 INPUT

;

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

;

PT1IN: ; RETURN CHAR IN REG A

0BF7 CDECOB CALL POLCI1 ;READY NOW?

0BFA B7 ORA A ;

JRNZ PT1IN1 ;IF READY, SKIP POLL

0BFB+2007 DB 020H,PT1IN1-$-1 ;---- FAKE JRNZ INSTRUCTION

0BFD 0E83 MVI C,POLL ;

0BFF 1E06 MVI E,PLCI1 ; POLL CONSOLE #1 INPUT

0C01 CD100B CALL XDOS ;

0C04 DB2C PT1IN1: IN DATA1 ; READ CHARACTER

0C06 E67F ANI 7FH ; STRIP PARITY

0C08 C9 RET ;

;

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

;

; CONSOLE # 1 OUTPUT

;

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

;

PT1OUT: ; REG C = CHAR TO OUTPUT

0C09 CD1C0C CALL POLCO1 ;ARE WE READY NOW?

0C0C B7 ORA A ;

JRNZ PT1OUT1 ;IF READY, SKIP POLL

0C0D+2009 DB 20H,PT1OUT1-$-1 ;---- FAKE JRNZ INSTRUCTION

0C0F C5 PUSH B ;

0C10 0E83 MVI C,POLL ;

0C12 lE02 MVI E,PLCO1 ;

0C14 CD100B CALL XDOS ; POLL CONSOLE #1 OUTPUT

0C17 C1 POP B ;

PT1OUT1:

0C18 79 MOV A,C ;

0C19 D32C OUT DATA1 ; TRANSMIT CHARACTER

0C1B C9 RET ;

;

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

;

; POLL CONSOLE # 1 OUTPUT

;

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

;

POLCO1: ; RETURN 0FFH IF READY

0C1C 3E10 MVI A,10H ; 000H IF NOT

0C1E D32D OUT STS1 ; RESET INT BIT

0C20 DB2D IN STS1 ; READ STATUS

0C22 E60C ANI 0CH ; MASK FOR DTR AND TXE

0C24 FE0C CPI 0CH ; MUST HAVE BOTH

0C26 3E00 MVI A,0 ;

0C28 C0 RNZ ; RETURN NOT READY

0C29 3D DCR A ;CHANGE "A" TO OFFH

0C2A C9 RET ; RETURN READY

PAGE

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

;

; POLL CONSOLE # 2 INPUT

;

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

POLCI2:

PT2ST: ; TEST CONSOLE STATUS

0C2B AF XRA A ; RETURN 0FFH IF READY

0C2C D32F OUT STS2 ; 000H IF NOT

0C2E DB2F IN STS2 ;

0C30 E601 ANI 1 ; RX CHAR ?

0C32 C8 RZ ; NO

0C33 3EFF MVI A,0FFH ; YES - SET FLAG

0C35 C9 RET ;

;

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

;

; CONSOLE # 2 INPUT

;

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

;

PT2IN: ; RETURN CHAR IN REG A

0C36 CD2B0C CALL POLCI2 ;READY NOW?

0C39 B7 ORA A ;

JRNZ PT2IN1 ;IF READY, SKIP POLL

0C3A+2007 DB 020H,PT2IN1-$-1 ;---- FAKE JRNZ INSTRUCTION

0C3C 0E83 MVI C,POLL ;

0C3E 1E07 MVI E,PLCI2 ;POLL CONSOLE #2 INPUT

0C40 CD100B CALL XDOS ;

0C43 DB2E PT2IN1: IN DATA2 ; READ CHARACTER

0C45 E67F ANI 7FH ; STRIP PARITY

0C47 C9 RET ;

;

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

;

; CONSOLE # 2 OUTPUT

;

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

;

PT2OUT: ; REG C = CHAR TO OUTPUT

0C48 CD5BOC CALL POLCO2 ;READY NOW?

0C4B B7 ORA A ;

JRNZ PT2OUT1 ;IF READY, SKIP POLL

0C4C+2009 DB 20H,PT2OUT1-$-1 ;---- FAKE JRNZ INSTRUCTION

0C4E C5 PUSH B ;

0C4F 0E83 MVI C,POLL ;

0C51 1E03 MVI E,PLCO2 ;

0C53 CD100B CALL XDOS ; POLL CONSOLE #2 OUTPUT

0C56 C1 POP B ;

PT2OUT1:

0C57 79 MOV A,C ;

0C58 D32E OUT DATA2 ; TRANSMIT CHARACTER

0C5A C9 RET ;

;

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

;

; POLL CONSOLE # 2 OUTPUT

;

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

;

POLCO2: ; RETURN 0FFH IF READY

0C5B 3E10 MVI A,10H ; 000H IF NOT

0C5D D32F OUT STS2 ; RESET INT BIT

0C5F DB2F IN STS2 ; READ STATUS

0C61 E60C ANI 0CH ; MASK FOR DTR AND TXE

0C63 FE0C CPI 0CH ; MUST HAVE BOTH

0C65 3E00 MVI A,0 ;

0C67 C0 RNZ ; RETURN NOT READY

0C68 3D DCR A ;CHANGE "A" TO 0FFH

0C69 C9 RET ; RETURN READY

PAGE

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

;

; POLL CONSOLE # 3 INPUT

;

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

POLCI3:

PT3ST: ; TEST CONSOLE STATUS

0C6A AF XRA A ; RETURN 0FFH IF READY

0C6B D32B OUT STS3 ; 000H IF NOT

0C6D DB2B IN STS3 ;

0C6F E601 ANI 1 ; RX CHAR ?

0C71 C8 RZ ; NO

0C72 3EFF MVI A,0FFH ; YES - SET FLAG

0C74 C9 RET

;

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

;

; CONSOLE # 3 INPUT

;

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

;

PT3IN: ; RETURN CHAR IN REG A

0C75 CD6A0C CALL POLCI3 ;READY NOW?

0C78 B7 ORA A ;

JRNZ PT3IN1 ;IF READY, SKIP POLL

0C79+2007 DB 20H,PT3IN1-$-1 ;---- FAKE JRNZ INSTRUCTION

0C7B 0E83 MVI C,POLL ;

0C7D 1E08 MVI E,PLCI3 ; POLL CONSOLE #3 INPUT

0C7F CD100B CALL XDOS ;

0C82 DB2A PT3IN1: IN DATA3 ; READ CHARACTER

0C84 E67F ANI 7FH ; STRIP PARITY

0C86 C9 RET ;

;

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

;

; CONSOLE # 3 OUTPUT

;

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

;

PT3OUT: ; REG C = CHAR TO OUTPUT

0C87 CD9A0C CALL POLCO3 ;READY NOW?

0C8A B7 ORA A ;

JRNZ PT3OUT1 ;IF READY, SKIP POLL

0C8B+2009 DB 20H,PT3OUT1-$-1 ;---- FAKE JRNZ INS

0C8D C5 PUSH B ;

0C8E OE83 MVI C,POLL ;

0C90 1E04 MVI E,PLCO3 ;

0C92 CD100B CALL XDOS ; POLL CONSOLE #3 OUTPUT

0C95 C1 POP B ;

PT3OUT1:

0C96 79 MOV A,C ;

0C97 D32A OUT DATA3 ; TRANSMIT CHARACTER

0C99 C9 RET ;

;

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

;

; POLL CONSOLE # 3 OUTPUT

;

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

;

POLCO3: ; RETURN 0FFH IF READY

0C9A 3E10 MVI A,10H ; 000H IF NOT

0C9C D32B OUT STS3 ; RESET INT BIT

0C9E DB2B IN STS3 ; READ STATUS

0CA0 E60C ANI 0CH ; MASK FOR DTR AND TXE

0CA2 FE0C CPI 0CH ; MUST HAVE BOTH

0CA4 3E00 MVI A,0 ;

0CA6 C0 RNZ ; RETURN NOT READY

0CA7 3D DCR A ;CHANGE "A" TO OFFH

0CA8 C9 RET ; RETURN READY

PAGE

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

;

; LINE PRINTER #0 DRIVER

;

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

LIST: ;LIST OUTPUT #0

0CA9 CDBC0C CALL POLLPT

0CAC B7 ORA A ;IS PRINTER READY NOW?

JRNZ LIST1 ;IF READY, SKIP POLL

0CAD+2009 DB 020H,LIST1-$-1 ;---- FAKE JRNZ INSTRUCTION

0CAF C5 PUSH B

0CB0 0E83 MVI C,POLL ; POLL PRINTER STATUS

0CB2 1E00 MVI E,PLLPT ;

0CB4 CD100B CALL XDOS ;

OCB7 C1 POP B ;

LIST1:

0CB8 79 MOV A,C ; CHARACTER TO PRINT

0CB9 D31E OUT LPTPRT0 ;

0CBB C9 RET ;

;

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

;

; POLL PRINTER OUTPUT

;

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

;

POLLPT: ; RETURN 0FFH IF READY

0CBC 3E10 MVI A,10H ; 000H IF NOT

0CBE D31F OUT LPTSTS0 ; RESET INT BIT

0CC0 DB1F IN LPTSTS0 ; READ STATUS

0CC2 E60C ANI 0CH ; MASK FOR DTR AND TXE

0CC4 FE0C CPI 0CH ; MUST HAVE BOTH

0CC6 3E00 MVI A,0 ;

0CC8 C0 RNZ ; RETURN NOT READY

0CC9 3D DCR A ;CHANGE "A" TO OFFH

0CCA C9 RET ; RETURN READY

;

PAGE

;

; MP/M 1.0 EXTENDED I/O SYSTEM

;

POLLDEVICE:

; REG C = DEVICE # TO BE POLLED

; RETURN 0FFH IF READY,

; 000H IF NOT

0CCB 79 MOV A,C

0CCC FE09 CPI NMBDEV

JRC DEVOK

0CCE+3802 DB 038H,DEVOK-$-1 ;---- FAKE JRC INSTRUCTION

0CD0 3E09 MVI A,NMBDEV ; IF DEV # >= NMBDEV,

; SET TO NMBDEV

DEVOK:

0CD2 CDA20B CALL TBLJMP ; JUMP TO DEV POLL CODE

DEVTBL:

0CD5 BC0C DW POLLPT ; POLL PRINTER OUTPUT - THIS WIL

; ; SPECIFIED PARALLEL PORT FOR P

0CD7 DD0B DW POLCO0 ; POLL CONSOLE #0 OUTPUT

0CD9 1C0C DW POLCO1 ; POLL CONSOLE #1 OUTPUT

0CDB 5B0C DW POLCO2 ; POLL CONSOLE #2 OUTPUT

0CDD 9A0C DW POLCO3 ; POLL CONSOLE #3 OUTPUT

0CDF AD0B DW POLCI0 ; POLL CONSOLE #0 INPUT

0CE1 EC0B DW POLCI1 ; POLL CONSOLE #1 INPUT

0CE3 2B0C DW POLCI2 ; POLL CONSOLE #2 INPUT

0CE5 6A0C DW POLCI3 ; POLL CONSOLE #3 INPUT

0009 NMBDEV EQU ($-DEVTBL)/2

0CE7 1A0B DW RTNEMPTY ; BAD DEVICE HANDLER

PAGE

; SELECT / PROTECT MEMORY

SELMEMORY:

; REG BC = ADR OF MEM. DESCRIPTOR

; BC -> BASE 1 BYTE,

; SIZE 1 BYTE,

; ATTRIB 1 BYTE,

; BANK 1 BYTE.

;

; BIOS TABLE MODIFIED

0CE9 FE20 CPI 20H ;

0CEB CAEB0C JZ $

0CEE 210300 LXI H,3 ; POINT TO BANK

0CF1 09 DAD B

0CF2 7E MOV A,M ; GET IT

0CF3 32030D STA BANKNO ; SAVE BANK NUMBER

0CF6 17 RAL ;

0CF7 17 RAL ;

0CF8 17 RAL ;

0CF9 E618 ANI 018H ; MASK FOR PIO

0CFB F602 ORI MEMSK ;

0CFD 32040D STA CURMEM ; STORE CURRENT BANK MASK

0D00 D309 OUT 009H ; SET PIO

0D02 C9 RET

0D03 00 BANKNO: DB 0 ; LAST SELECTED MEMORY BANK NUMBER

0D04 00 CURMEM: DB 0 ; LAST SELECTED MEMORY BANK MASK

; START CLOCK

STARTCLOCK:

; WILL CAUSE FLAG #1 TO BE SET

; AT EACH SYSTEM TIME UNIT TICK

0D05 3EFF MVI A,0FFH

0D07 32CE0D STA TICKN

0D0A C9 RET

; STOP CLOCK

STOPCLOCK:

; WILL STOP FLAG #1 SETTING AT

; SYSTEM TIME UNIT TICK

0D0B AF XRA A

0D0C 32CE0D STA TICKN

0D0F C9 RET

; EXIT REGION

EXITREGION:

; EI IF NOT PREEMPTED

0D10 3ACF0D LDA PREEMP

0D13 B7 ORA A

0D14 C0 RNZ

0D15 FB EI

0D16 C9 RET

; MAXIMUM CONSOLE NUMBER

MAXCONSOLE:

0D17 3E04 MVI A,NMBCNS

0D19 C9 RET

; MP/M 1.0 INTERRUPT HANDLERS

008E = DSPTCH EQU 142

INT1HND:

; INTERRUPT 1 HANDLER ENTRY POINT

T20MS:

0D1A 22C80D SHLD SVDHL

0D1D 21220D LXI H,TIMERINT

JR INTINIT

0D20+185D DB 18H,INTINIT-$-1 ;---- FAKE JR INSTRUCTION

TIMERINT:

0D22 3ACE0D LDA TICKN

0D25 B7 ORA A ; TEST TICKN, INDICATES

; DELAYED PROCESS (ES)

JRZ NOTICKN

0D26+2807 DB 28H,NOTICKN-$-1 ;---- FAKE JRZ INSTRUCTION

0D28 0E85 MVI C,FLAGST

0D2A 1E01 MVI E,1

0D2C CD100B CALL XDOS ; SET FLAG #1 EACH TICK

NOTICKN:

0D2F 219D0D LXI H,CNTX

0D32 35 DCR M ; DEC TICK CNTR

JRNZ NOT1SEC

0D33+2032 DB 20H,NOT1SEC-$-1 ;---- FAKE JRNZ INSTRUCTION

0D35 3E7D MVI A,125

0D37 2B DCX H

0D38 96 SUB M

0D39 77 MOV M,A ; \*\*\* TOGGLE COUNT 62 <-> 6

0D3A 23 INX H

0D3B 77 MOV M,A ; \*\*\* ACTUAL #/SEC = 62.5

0D3C 0E85 MVI C,FLAGST

0D3E 1E02 MVI E,2

0D40 CD100B CALL XDOS ; SET FLAG #2 @ 1 SEC

0D43 2AD00D LHLD FPYTIME ;IS FLOPPY TIME CHECK IN EF

0D46 7C MOV A,H

0D47 B7 ORA A

JRZ NOT1SEC ;IF NOT IN EFFECT, FINISH

0D48+281D DB 28H,NOT1SEC-$-1 ;---- FAKE JRZ INSTRUCTION

0D4A 2D DCR L ;SUBTRACT A SECOND

0D4B 22D00D SHLD FPYTIME ;SAVE FOR NEXT TIME

JRNZ NOTlSEC ;IF NOT TOO LONG, FINISH

0D4E+2017 DB 20H,NOT1SEC-$-1 ;---- FAKE JRNZ INSTRUCTION

0D50 65 MOV H,L ;ZERO OUT INDICATOR

0D51 22D00D SHLD FPYTIME ;PREVENT RE-ENTRY OF THIS R

0D54 0E85 MVI C,FLAGST ;

0D56 1E06 MVI E,FPYFLAG ;

0D58 CD100B CALL XDOS ;CAUSE I/O FOR FLOPPY TO CO

0D5B 3E90 MVI A,10010000B

0D5D 32FCOA STA STATUS ;SHOW ERROR IN FLOPPY I/O

0D60 2AD20D LHLD FPYTCNT

0D63 23 INX H ;COUNT TIMES WD1791 GOES TO

0D64 22D20D SHLD FPYTCNT

NOT1SEC:

INTDONE:

0D67 AF XRA A

0D68 32CF0D STA PREEMP ; CLEAR PREEMPTED FLAG

0D6B C1 POP B

0D6C D1 POP D

0D6D 2ACA0D LHLD SVDSP

0D70 F9 SPHL ; RESTORE STK PTR

0D71 F1 POP PSW

0D72 2ACC0D LHLD SVDRET

0D75 E5 PUSH H

0D76 210D0B LXI H,PDISP ; MP/M DISPATCH

0D79 E5 PUSH H ; PUT ON STACK FOR RETURN

0D7A 2AC80D LHLD SVDHL

; THE FOLLOWING DISPATCH CALL WILL FORCE ROUND ROBIN

; SCHEDULING OF PROCESSES EXECUTING AT THE SAME PRIORITY

; EACH 1/32ND OF A SECOND.

; NOTE: INTERRUPTS ARE NOT ENABLED UNTIL THE DISPATCHER

; RESUMES THE NEXT PROCESS. THIS PREVENTS INTERRUPT

; OVER-RUN OF THE STACKS WHEN STUCK OR HIGH FREQUENCY

; INTERRUPTS ARE ENCOUNTERED.

RETI ; DISPATCH

0D7D+ED4D DB 0EDH,04DH ;---- FAKE RETI INSTRUCTION

INTINIT: ;SAVE MACHINE STATE FOR INTRPT HNDL

0D7F 22C60D SHLD ADRINTHD

0D82 E1 POP 141

0D83 22CC0D SHLD SVDRET

0D86 F5 PUSH PSW

0D87 210000 LXI H,0

0D8A 39 DAD SP

0D8B 22CA0D SHLD SVDSP ; SAVE USERS STK PTR

0D8E 31C60D LXI SP,LSTINTSTK ; LCL STK FOR INTR HNDL

0D91 D5 PUSH D

0D92 C5 PUSH B

0D93 3EFF MVI A,OFFH

0D95 32CF0D STA PREEMP ; SET PREEMPTED FLAG

0D98 2AC60D LHLD ADRINTHD

0D9B E9 PCHL ;JUMP TO INTERRUPT HANDLER

;

; BIOS DATA SEGMENT

;

0D9C 3E TOGCNT: DB 62 ; TOGGLE COUNTER 62 <-> 63

0D9D 3E CNTX: DB 62 ; TICK CNTR TO 1 SEC

INTSTK: ; LOCAL INTRPT STK

OD9E C7C7C7C7C7 DW 0C7C7H,0C7C7H,0C7C7H,0C7C7H,0C7C7H

ODA8 C7C7C7C7C7 DW 0C7C7H,0C7C7H,0C7C7H,0C7C7H,0C7C7H

ODB2 C7C7C7C7C7 DW 0C7C7H,0C7C7H,0C7C7H,0C7C7H,0C7C7H

0DBC C7C7C7C7C7 DW 0C7C7H,0C7C7H,0C7C7H,0C7C7H,0C7C7H

LSTINTSTK:

0DC6 0000 ADRINTHD: DW 0 ; INTERRUPT HANDLER ADDRESS

0DC8 0000 SVDHL: DW 0 ; SAVED REGS HL DURING INT HNDL

0DCA 0000 SVDSP: DW 0 ; SAVED SP DURING INT HNDL

0DCC 0000 SVDRET: DW 0 ; SAVED RETURN DURING INT HNDL

0DCE 00 TICKN: DB 0 ; TICKING BOOLEAN,TRUE = DELAYEI

0DCF 00 PREEMP: DB 0 ; PREEMPTED BOOLEAN

if mpm20

FPYTIME:

0DD0 0000 DW 0

FPYTCNT:

0DD2 0000 DW 0

endif

PAGE

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

;

; THESE ARE THE DISK TYPE DEFINITION BLOCKS

; EACH OF WHICH CORRESPONDS TO A PARTICULAR MODE.

;

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

0DD4 = DPB0: EQU $ ;VERSION 2.0, IBM SINGLE

0DD4 1A00 DW 26 ;SECTORS PER TRACK

0DD6 03 DB 3 ;BLOCK SHIFT

0DD7 07 DB 7 ;BLOCK SHIFT MASK

0DD8 00 DB 0 ;EXTENT MASK

0DD9 F200 DW 242 ;DISK SIZE MINUS 1

0DDB 3F00 DW 63 ;DIRECTORY MAX

0DDD C0 DB 192 ;ALLOC0

0DDE 00 DB 0 ;ALLOC1

0DDF 1000 DW 16 ;CHECK AREA SIZE

0DE1 0200 DW 2 ;OFFSET TO START TRACK

0DE3 = DPB1: EQU $ ;VERSION 2.0, IBM DOUBLE

0DE3 3400 DW 52 ;SECTORS PER TRACK

0DE5 04 DB 4 ;BLOCK SHIFT

0DE6 0F DB 15 ;BLOCK SHIFT MASK

0DE7 01 DB 1 ;EXTENT MASK

0DE8 F200 DW 242 ;DISK SIZE MINUS 1

0DEA 7F00 DW 127 ;DIRECTORY MAX

0DEC C0 DB 192 ;ALLOC0

0DED 00 DB 0 ;ALLOC1

0DEE 2000 DW 32 ;CHECK AREA SIZE

0DF0 0200 DW 2 ;OFFSET TO START TRACK

0DF2 = DPB2: EQU $ ;VERSION 1.4 ALTOS DOUBLE D

0DF2 3000 DW 48 ;SECTORS PER TRACK

0DF4 04 DB 4 ;BLOCK SHIFT

0DF5 0F DB 15 ;BLOCK SHIFT MASK

0DF6 00 DB 0 ;EXTENT MASK (1.4 COMPATABI

0DF7 E000 DW 224 ;DISK SIZE MINUS 1

0DF9 5F00 DW 95 ;DIRECTORY MAX

0DFB C0 DB 192 ;ALLOC0

0DFC 00 DB 0 ;ALLOC1

0DFD 1800 DW 24 ;CHECK AREA SIZE

0DFF 0200 DW 2 ;OFFSET TO START TRACK

IF HARDSK

if mpm20

DPB3: DISKDEF 3,0,127,,16384,512,512,0,1,,0

0E01+= DPB3 EQU $ ;DISK PARM BLOCK

0E01+8000 DW 128 ;SEC PER TRACK

0E03+07 DB 7 ;BLOCK SHIFT

0E04+7F DB 127 ;BLOCK MASK

0E05+07 DB 7 ;EXTNT MASK

0E06+FF01 DW 511 ;DISK SIZE-1

0E08+FF01 DW 511 ;DIRECTORY MAX

0E0A+80 DB 128 ;ALLOC0

0E0B+00 DB 0 ;ALLOC1

0E0C+0080 DW 8000H+CKSZ ;PERMANENT DISK WIT

0E0E+0100 DW 1 ;OFFSET

0000+= XLT3 EQU 0 ;NO XLATE TABLE

DPB4: DISKDEF 4,0,127,,16384,512,512,0,513,,0

0E10+= DPB4 EQU $ ;DISK PARM BLOCK

0E10+8000 DW 128 ;SEC PER TRACK

0E12+07 DB 7 ;BLOCK SHIFT

0E13+7F DB 127 ;BLOCK MASK

0E14+07 DB 7 ;EXTNT MASK

0E15+FF01 DW 511 ;DISK SIZE-1

0E17+FF01 DW 511 ;DIRECTORY MAX

0E19+80 DB 128 ;ALLOC0

0E1A+00 DB 0 ;ALLOC1

0E1B+0080 DW 8000H+CKSZ ;PERMANENT DISK WIT

0E1D+0102 DW 513 ;OFFSET

0000+= XLT4 EQU 0 ;NO XLATE TABLE

DPB5: DISKDEF 5,0,127,,16384,512,512,0,1025,,0

0E1F+= DPB5 EQU $ ;DISK PARM BLOCK

0E1F+8000 DW 128 ;SEC PER TRACK

0E21+07 DB 7 ;BLOCK SHIFT

0E22+7F DB 127 ;BLOCK MASK

0E23+07 DB 7 ;EXTNT MASK

0E24+FF01 DW 511 ;DISK SIZE-1

0E26+FF01 DW 511 ;DIRECTORY MAX

0E28+80 DB 128 ;ALLOCO

0E29+00 DB 0 ;ALLOC1

0E2A+0080 DW 8000H+CKSZ ;PERMANENT DISK

0E2C+0104 DW 1025 ;OFFSET

0000+= XLT5 EQU 0 ;NO XLATE TABLE

DPB6: DISKDEF 6,0,127,,16384,288,512,0,513,,0

0E2E+= DPB6 EQU $ ;DISK PARM BLOCK

0E2E+8000 DW 128 ;SEC PER TRACK

0E30+07 DB 7 ;BLOCK SHIFT

0E31+7F DB 127 ;BLOCK MASK

0E32+07 DB 7 ;EXTNT MASK

0E33+1F01 DW 287 ;DISK SIZE-1

0E35+FF01 DW 511 ;DIRECTORY MAX

0E37+80 DB 128 ;ALLOCO

0E38+00 DB 0 ;ALLOC1

0E39+0080 DW 8000H+CKSZ ;PERMANENT DISK

0E3B+0102 DW 513 ;OFFSET

0000+= XLT6 EQU 0 ;NO XLATE TABLE

else

DPB3: DISKDEF 3,0,127,,16384,512,512,0,1

DPB4: DISKDEF 4,0,127,,16384,512,512,0,513

DPB5: DISKDEF 5,0,127,,16384,512,512,0,1025

DPB6: DISKDEF 6,0,127,,16384,288,512,0,513

endif

ENDIF

if mdisk

DPB7: EQU $ ;VIRTUAL DISK

DW 24 ;SECTORS PER TRACK

DB 3 ;BLOCK SHIFT

DB 7 ;BLOCK SHIFT MASK

DB 0 ;EXTENT MASK

DW 142 ;DISK SIZE MINUS 1

DW 63 ;DIRECTORY MAX

DB OCOH ;ALLOC0

DB 0 ;ALLOC1

DW 0 ;CHECK AREA SIZE

DW 0 ;OFFSET TO START TRACK

endif

page

;

; MOVE SUBROUTINE

;

if hardsk

RWMOVE:

0E3D D5 push d

0E3E E5 push h

0E3F CD070B call swtuser ;switch in user bank

0E42 E1 pop h

0E43 D1 pop d

0E44 018000 lxi b,128

LDIR ;MOVE DATA TO/FROM BUFFER

0E47+EDB0 DB 0EDH,0B0H ;---- FAKE LDIR INSTRUCTION

0E49 CD0A0B call swtsys ;switch system back in

;

; DATA HAS BEEN MOVED TO/FROM HOST BUFFER

;

0E4C 3AF90A LDA WRTYPE ;WRITE TYPE ??

if mpm20

0E4F E601 ani WRDIR ;TO DIRECTORY ??

JRZ RWEND ;NO, JUST END UP HERE

0E51+280D DB 028H,RWEND-$-1 ;---- FAKE JRZ INSTRUCTION

else

CPI WRDIR ;TO DIRECTORY ??

JRNZ RWEND ;NO, JUST END UP HERE

endif

;

; CLEAR HOST BUFFER FOR DIRECTORY WRITE

;

0E53 3AF60A LDA ERFLAG ;CHECK PRIOR TO DIR ACTIVIT

0E56 B7 ORA A ;ERRORS ??

JRNZ RWEND ;SKIP IF SO

0E57+2007 DB 020H,RWEND-$-1 ;---- FAKE JRNZ INSTRUCTION

0E59 AF XRA A ;ZERO TO ACCUMULATOR

0E5A 32F00A STA HSTWRT ;BUFFER WRITTEN

0E5D CD6D04 CALL WRITEHST ;

RWEND:

0E60 3AF60A LDA ERFLAG ;

0E63 B7 ORA A ;IF ERRORS, RESET SO NO MATCH

0E64 C8 RZ ;NONE, JUST RETURN

0E65 21EA0A LXI H,HSTDSK ;

0E68 36FF MVI M,OFFH ;CANT POSSIBLY MATCH, MUST

ENDIF

0E6A C9 RET ;

MVDTB:

0E6B 2AAF00 LHLD DMAADR ; MOVE DATA TO FLOPPY BUFFER

0E6E E5 push h ;

0E6F CD070B call swtuser ;switch in user bank,

0E72 E1 POP h ; cannot access non-common BNKXIO

0E73 111D13 LXI D,FPYBUF ;

0E76 018000 LXI B,128 ; 128 BYTES

LDIR ;

0E79+EDB0 DB 0EDH,0B0H ;---- FAKE LDIR INSTRUCTION

0E7B C30AOB jmp swtsys ;switch system back in

RET

0E7E F5 MVDFB: PUSH PSW ; MOVE DATA FROM FLOPPY BUF

0E7F 3AFA0A LDA CMD ;

0E82 E620 ANI 20H ;CHECK FOR READ

JRNZ MVDFX ;NO - BYPAS MOVE

0E84+2013 DB 20H,MVDFX-$-1 ;---- FAKE JRNZ INSTRUCTION

0E86 2AAF00 LHLD DMAADR ;

0E89 E5 push h

0E8A CD070B call swtuser ;switch in user bank,

0E8D D1 POP d ; cannot access non-common BNKXIO

0E8E 211D13 LXI H,FPYBUF ;

0E91 018000 LXI B,128 ; 128 BYTES

LDIR ;

0E94+EDBO DB 0EDH,0B0H ;---- FAKE LDIR INSTRUCTION

0E96 CDOAOB call swtsys ;switch system back in

0E99 F1 MVDFX: POP PSW ;

0E9A C9 RET ;

IF HARDSK

0E9B DS 1 ;MUST PRECEDE HSTBU

0E9C HSTBUF: DS 1024 ;HOST BUFFER AREA

129C DS 1 ;MUST FOLLOW HSTBUF

ENDIF

PAGE

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

;

; INITIALIZE MP/M: REAL TIME CLOCK & DISKS

;

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

if mpm20

129D = dirbuf equ $

131D = fpybuf equ dirbuf+128

endif

SYSTEMINIT:

; C = BREAKPOINT RESTART NUMBER

; DE = BREAKPOINT RESTART HANDLER ADDRESS

; HL = DIRECT XIOS INTERCEPT JUMP TABLE ADDRESS

129D 225E13 SHLD SVDJT

12A0 69 MOV L,C

12A1 2600 MVI H,0

12A3 29 DAD H

12A4 29 DAD H

12A5 29 DAD H ;HL = RESTART JUMP ADDRESS

12A6 226013 SHLD SVDBPA

if not mdisk

12A9 2A130B lhld sysdat

12AC 2E0F mvi l,15 ;hl = .nmbmemsegs

12AE 46 mov b,m ;b = nmbmemsegs

test$bank$setup$loop:

12AF 23 inx h

12B0 23 inx h

12B1 23 inx h

12B2 23 inx h ;hl = .memseg(i).bank

12B3 7E mov a,m

12B4 B7 ora a

12B5 C2BF12 jnz bank$setup

12B8 05 dcr b

12B9 C2AF12 jnz test$bank$setup$loop

12BC C3CE12 jmp after$bank$setup

bank$setup:

12BF 3E1A MVI A,01AH ; SELECT BANK 3

12C1 CD4813 CALL STMVTR ; SET UP VECTORS

12C4 3E12 MVI A,012H ; SELECT BANK 2

12C6 CD4813 CALL STMVTR ; SET UP VECTORS

12C9 3E0A MVI A,00AH ; SELECT BANK 1

12CB CD4813 CALL STMVTR ; SET UP VECTORS

after$bank$setup:

else

mvi a,1ah ; bank 3 select for directo

out 09h

lxi h,0bffeh

mvi a,0e5h

cmp m

inx h

jrnz fill

cmp m

jrz dontfill

fill:

mov m,a ;set directory initialized

dcx h

mov m,a

lxi b,07ffh ;first 2 k of bank one gets

lxi h,0

lxi d,1

mvi a,0ah ; select bank 1

out 09h

mvi m,Oe5h

ldir

dontfill:

endif

12CE 3E02 MVI A,002H ; SELECT BANK 0

12DO CD4813 CALL STMVTR ; SET UP VECTORS

1203 213717 lxi h,ldrbiosbase+density$mask$offset

;;;;; LXI H,1737H ; MOVE PARAMETERS CHANGED B

12D6 117C00 LXI D,SEL0 ; THE SETUP PROGRAM

12D9 010400 LXI B,4 ; 4 SELECT MASKS

LDIR ;

12DC+EDB0 DB 0EDH,0B0H ;---- FAKE LDIR INSTRUCTION

12DE 118800 LXI D,MODE ;

12E1 010400 LXI B,4 ; 4 MODE BYTES

LDIR ;

12E4+EDB0 DB 0EDH,0B0H ;---- FAKE LDIR INSTRUCTION

12E6 2ABB17 lhld ldrbiosbase+misc$params$offset

;;;;; LHLD 17BBH ; GET MISC. PARAMETERS

12E9 22B600 SHLD MPARMS ;

12EC 3AB600 LDA MPARMS ; NOW TEST FOR CENTRONICS P

12EF E602 ANI 2 ;

JRZ PRTOK ; NO - LEAVE SERIAL

12FJ+2814 DB 028H,PRTOK-$-1 ;---- FAKE JRZ INSTRUCTION

12F3 212C0B LXI H,CLIST ;

12F6 221000 SHLD WBOTE+13 ; CHANGE PRINTER ROUTINE

12F9 211F0B LXI H,CNSTAT ; AND STATUS CHECK

12FC 22D50C SHLD DEVTBL ;

12FF 3E03 MVI A,003H ;INITIALIZE PARALLEL PORT

1301 D313 OUT 013H

1303 3E0F MVI A,00FH ;

1305 D313 OUT 013H

PRTOK:

1307 010300 LXI B,003H ;SET THE MODE FOR DRIVES IN

MODESET:

130A CD2F02 CALL SELSDP ;SELECT DRIVE FOR MODESET

130D 218800 LXI H,MODE ;

1310 09 DAD B ;POINT TO CORRECT MODE BY

1311 C5 PUSH B ;SAVE COUNT OF DRIVES

1312 41 MOV B,C ; B = DRIVE #

1313 4E MOV C,M ;

1314 CDF807 CALL XETMOD ;SET MODE

1317 C1 POP B ;

1318 0D DCR C ;END OF LIST YET ??

1319 F20A13 JP MODESET ;SET MODE FOR ALL DRIVES

131C CDD007 CALL SDCONF ;SET DISK CONFIGURATION

131F 018000 LXI B,80H

1322 CD5502 CALL SETDMA ;SET DMA ADDRESS

1325 E5 push h

if mpm20

1326 2A130B lhld sysdat

1329 2E07 mvi 1,7

132B 7E mov a,m

else

lxi h,INTERUPT

mov a,h

endif

132C E1 pop h

132D ED47 DB 0EDH,047H ;---- FAKE STAI INSTRUCTION

132F 3E60 MVI A,60H ; SET VECTOR FOR CTC

1331 D330 OUT 30H ; CTC CHANNEL 0

1333 3EA7 MVI A,0A7H ; RESET / LOAD TIME CONST

1335 D333 OUT 33H ; CHANNEL 3

1337 3EFA MVI A,250 ; TIME CONSTANT

1339 D333 OUT 033H ;

IF HARDSK

133B AF XRA A ;ZERO ACCUMULATOR

133C 32EF0A STA HSTACT ;SET HOST BUFFER INACTIVE

133F 32F10A STA UNACNT ;SET UNALLOCATED COUNT TO Z

1342 219B0E LXI H,HSTBUF-1 ;SETUP WRITE CONTROL BYTE F

1345 360D MVI M,00DH ;

ENDIF

1347 C9 RET ;

STMVTR:

1348 D309 OUT MEMPORT

134A 3EC3 MVI A,0C3H ; SET VECTORS FOR BDOS

134C 320000 STA 0 ; JMP INSTRUCTION

134F 2A5E13 LHLD SVDJT ;

1352 220100 SHLD 1

1355 2A6013 LHLD SVDBPA

1358 77 MOV M,A

1359 23 INX H

135A 73 MOV M,E

135B 23 INX H

135C 72 MOV M,D

135D C9 RET ;

135E SVDJT: DS 2 ; SAVED DIRECT JUMP TABLE ADDRESS

1360 SVDBPA: DS 2 ; SAVED BREAK POINT ADDRESS

if mpm20

1362 = xiosend equ $

139D = fdbuf equ (dirbuf-base)+256

139D org fdbuf+((xiosend-base)/fdbuf)\*((xiosend-base)-fd

139D 00 db 0

endif

139E END

070F ADDERRORS 0DC6 ADRINTHD 12CE AFTERBANKS 03DA ALLOC

081E ALV0 085E ALV1 089E ALV2 08DE ALV3

091E ALV4 095E ALV5 099E ALV6 09DE ALV7

OA1E ALV8 0A5E ALV9 0A9E ALVA 0AC2 ALVB

0708 BADIO 0D03 BANKNO 12BF BANKSETUP 0000 BASE

081E BEGDAT 4000 BLKSIZ 067B CHECKIT 06AE CHECKSTAT

0220 CHKHRD 06BC CHKS0 06C7 CHKS1 06CD CHKS2

06E6 CHKS3 06F7 CHKS4 03A1 CHKUNA 0B2C CLIST

0B3B CLIST1 0AFA CMD 0B1F CNSTAT 0D9D CNTX

0B15 COLDSTART 0B04 COMMONBASE 0B84 CONIN 0B8F CONOUT

0B79 CONST 0080 CPMSPT 083E CSV0 087E CSV1

08BE CSV2 08FE CSV3 095E CSV4 099E CSV5

09DE CSV6 0A1E CSV7 0A5E CSV8 0A9E CSV9

OAC2 CSVA 0AE6 CSVB 0D04 CURMEM 001C DATA0

002C DATA1 002E DATA2 002A DATA3 0OB4 DBLKAD

052E DBLLOW 0530 DBLSAVE 0519 DBLUPDATE 0782 DEL1

0784 DEL2 0782 DELAY 0037 DENSITYMAS 0CD2 DEVOK

0CD5 DEVTBL 129D DIRBUF OOAC DISKNO FFFF DMA

00AF DMAADR 00BE DMALEN 00BA DMAS1 00C6 DMAS2F

00C0 DMAS2H 00CA DMAS3 00CE DMAS3F 00BC DMASA

0DD4 DPB0 0DE3 DPB1 0DF2 DPB2 0E01 DPB3

0E10 DPB4 0E1F DPB5 0E2E DPB6 00D1 DPBASE

00D1 DPE0 00E1 DPE1 00F1 DPE2 0101 DPE3

0111 DPE4 0121 DPE5 0131 DPE6 0141 DPE7

0151 DPE8 0161 DPE9 0171 DPEA 0181 DPEB

0OB2 DPEPTR 0790 DSCNO 0552 DSKSEL 008E DSPTCH

0246 DTBLT 0AF6 ERFLAG 0D10 EXITREGION 0000 FALSE

139D FDBUF 0B50 FDINTH 041C FILLHST 0763 FINTFIX

0085 FLAGST 0084 FLAGWT 0B47 FLOPPYINT 0687 FLOPPYIO

065C FLOPPYSEEK 0661 FPS1 131D FPYBUF 0006 FPYFLAG

ODD2 FPYTCNT 0DD0 FPYTIME 073A FPYWAIT 0670 FSECSET

06AA FWT1 0B5E HARDINT FFFF HARDSK 0005 HDFLAG

OB67 HDINTH 0B71 HDSTFLG 00AE HEADNO 031A HOME

0337 HOME1 0340 HOME1A 0357 HOME2 0343 HOMEHARD

02F9 HOMEIT 0321 HOMESOFT 0B03 HOMETOGGLE 048F HRW0

0499 HRW1 04C4 HRW2 04CD HRW3 04D9 HRW4

04E1 HRW5 04E6 HRW6 0504 HRW7 0AEF HSTACT

0008 HSTBLK 0E9C HSTBUF OAEA HSTDSK 0AED HSTSEC

0400 HSTSIZ 0010 HSTSPT 0AEB HSTTRK 0AF0 HSTWRT

0OB8 HTK1 00B9 HTK2 07F8 INITEND 0D1A INTJHND

OD67 INTDONE 005E INTERUPT 0721 INTFIX 0D7F INTINIT

OD9E INTSTK 0051 LAST 062D LDH1 1700 LDRBIOSBAS

OCA9 LIST 0CB8 LISTI 061E LOADHEAD 001E LPTPRTO

0028 LPTPRT1 001F LPTSTS0 0029 LPTSTS1 0DC6 LSTINTSTK

OAFB MASK 0439 MATCH 0D17 MAXCONSOLE 000C MAXDSK

0000 MDISK 0009 MEMPORT 0002 MEMSK 00BB MISCPARAMS

0088 MODE 130A MODESET 0191 MODL0 019D MODL1

01A9 MODL2 00B6 MPARMS FFFF MPM20 0E7E MVDFB

OE99 MVDFX 0E6B MVDTB 0AE6 NEWDSK 0AEE NEWHST

OAE9 NEWSEC 0AE7 NEWTRK 0461 NEWTRKCMP 0004 NMBCNS

0009 NMBDEV 075F NOFPYRST 0415 NOMATCH 03D4 NOOVF

OD67 NOTlSEC 0D2F NOTICKN 0B1C NULLINT 00A0 PCNT

OBOD PDISP 0005 PLCI0 0006 PLCI1 0007 PLC12

0008 PLC13 0001 PLCOO 0002 PLCO1 0003 PLC02

0004 PLC03 0000 PLLPT 05D4 PNTFN 05D1 PNTH2

05B3 POINT 0BAD POLCIO 0BEC POLCI1 0C2B POLCI2

0C6A POLCI3 0BDD POLCOO 0C1C POLCO1 0C5B POLCO2

0C9A POLCO3 0083 POLL 0CCB POLLDEVICE 0CBC POLLPT

0DCF PREEMP 0B01 PRETRIES 1307 PRTOK 0BB8 PTOIN

0BC5 PTOIN1 0BCA PTOOUT 0BD9 PTOOUT1 0BAD PTOST

0BF7 PT1IN 0C04 PT1IN1 0C09 PT1OUT 0C18 PT1OUT1

0BEC PT1ST 0C36 MIN 0C43 PT21N1 0C48 PT2OUT

0C57 PT20UT1 0C2B PT2ST 0C75 PT31N 0C82 PT31N1

0C87 PT30UT 0C96 PT30UT1 0C6A PT3ST 0B9A PTBLJMP

028B READ 036B READHARD 047F READHST 0AF8 READOP

05E4 READSOFT 0308 REALDISK FFFF RELOC 0643 REMOVELD

02EE RETMOD 0AF7 RSFLAG 0B1A RTNEMPTY 0E60 RWEND

0E3D RWMOVE 03E2 RWOPER 0AFD SAVE1 07D0 SDCONF

07DE SDDBL 07F1 SDL1 07F3 SDOK 0007 SECMSK

0003 SECSHF 00B1 SECTNO 05D6 SECTRAN 007C SELO

0203 SELDSK 0242 SELERR 059B SELHARD 0CE9 SELMEMORY

022F SELSDP 0556 SELSOFT 0278 SETDEN 0255 SETDMA

0211 SETDSK 0544 SETDVD 0540 SETH14 0532 SETHED

02A1 SETMOD 0273 SETSEC 02C7 SETSEL 026D SETTRK

0547 SHD1 05A1 SLH1 055D SLS1 056B SLS2

0575 SLS3 0584 SLS4 0595 SLSERR 02E3 SMERR

0601 SRW1 0615 SRW2 0D05 STARTCLOCK 0AFC STATUS

1348 STMVTR 0D0B STOPCLOCK 05DB STRN1 05E2 STRN2

001D STS0 002D STS1 002F STS2 002B STS3

1360 SVDBPA 0DC8 SVDHL 135E SVDJT 0DCC SVDRET

0DCA SVDSP 0B0A SWTSYS 0B07 SWTUSER OB13 SYSDAT

129D SYSTEMINIT 0D1A T20MS 0BA2 TBLJMP 0094 TCNT

12AF TESTBANKSE 0DCE TICKN 0D22 TIMERINT 0D9C TOGCNT

00AD TRAKNO 0B02 TRETRIES 0070 TRKO 064A TRKTST

FFFF TRUE 0AF1 UNACNT 0AF2 UNADSK 0AF5 UNASEC

0AF3 UNATRK 0717 WAIT0 0B15 WARMSTART 0003 WBOTE

0000 WRALL 0001 WRDIR 0296 WRITE 037E WRITEHARD

046D WRITEHST 05F2 WRITESOFT 0AF9 WRTYPE 0002 WRUAL

0B10 XDOS 07F8 XETMOD 0801 XETSEL 1362 XIOSEND

01B5 XLT0 01CF XLT1 01CF XLT2 0000 XLT3

0000 XLT4 0000 XLT5 0000 XLT6

1. Оригинал на английском языке [↑](#footnote-ref-1)