Sprinter

Руководство по программированию Sp2000

Ivan Mak

15 августа 2003 г.

Содержание

Часть I
Введение

Данное описание предполагает наличие определенных знаний читателя, а именно знание архитектуры компьютера ZX-Spectrum и их разновидностей, в частности Pentagon-128 и Scorpion-256, а так же знание языка BASIC и некоторое знакомство с языком ассемблера Z80. В описании почти не приводятся сведения, относящиеся к программированию на стандартном ZX-Spectrum. Подразумевается, что эти описания уже имеются или читатель достаточно хорошо знаком с компьютером ZX-Spectrum, что бы знать основные данные, необходимые для понимания излагаемой информации.

Большая часть описания относится к стандартной конфигурации компьютера Sprinter. Там же, где разговор пойдет об иных конфигурациях, я буду это указывать.

1 Архитектура компьютера

Я буду называть конфигурацией машины – конкретную реализацию конкретной схемы в перепрограммируемой логической микросхеме (ППЛМ). Это означает, что машина имеет множество конфигураций, каждая из которых имеет свою схему.

Я так же использую понятие КЭШ-ОЗУ. Это не КЭШ в формальном смысле, а быстрое ОЗУ, в котором процессор может работать на высокой частоте без тактов ожидания. КЭШ-ем это ОЗУ называется только по традиции, подобно КЭШ-у на КР537РУ10 в компьютерах Pentagon-128.

1.1 Краткие данные платы Sp2000

Процессор Z84C15	Контроллер винчестера . . . . . IDE/AT
Тактовая частота 21MHz/3.5MHz	Контроллер клавиатуры . . . 101key/AT
ОЗУ . . . . . от 4Mb до 64Mb	Контроллер мыши . . . . . MS-Mouse
КЭШ ОЗУ 64Kb	Два слота . . . . . стандарт ISA-8
ПЗУ 256Kb	Аналог AY-3-8910 в ПЛМ . . stereo-OUT
Видео-ОЗУ . . . . . 256Kb(512Kb)	COVOX stereo-16bit
Контроллер дисков Кр1818ВГ93	Выход видео на TV или CGA монитор, RGB
Поддержка 1.44Mb формата . . 3.5"диска	Видео-режимы: . . . Spectrum standart
CMOS-часы . . . . . DALLAS	GRAF 320x256x256,640x256x16, TXT 80x32

1.2 Техническая реализация

Ядром машины являются процессор Z84C15 и ППЛМ фирмы ALTERA - EP1K30QC208-3. Кроме них на плате присутствуют микросхема ПЗУ, слот под 72х-пиновый SIMM на 4-64Mb, 256Kb видео-ОЗУ, 64Kb КЭШ-ОЗУ, схема контроллера дисковода на БИС КР1818ВГ93, буферы для подключения джойстика, магнитофона, принтера, клавиатуры, дисководов, винчестера, мыши, буферные микросхемы шины ISA-8 и еще одна ППЛМ фирмы ALTERA – EPM7064SLC100. Эта ППЛМ не меняет своей конфигурации и предназначена для обеспечения синхронизации и начального запуска компьютера. На плате так же предусмотрена возможность подключения CMOS часов – микросхемы DALLAS. Кроме периферии и буферов имеются микросхемы дешифрации, входы которых подключаются к процессору через ППЛМ. Это позволяет легко менять адресацию устройств без какого либо изменения разводки печатной платы.

1.3 Возможности машины

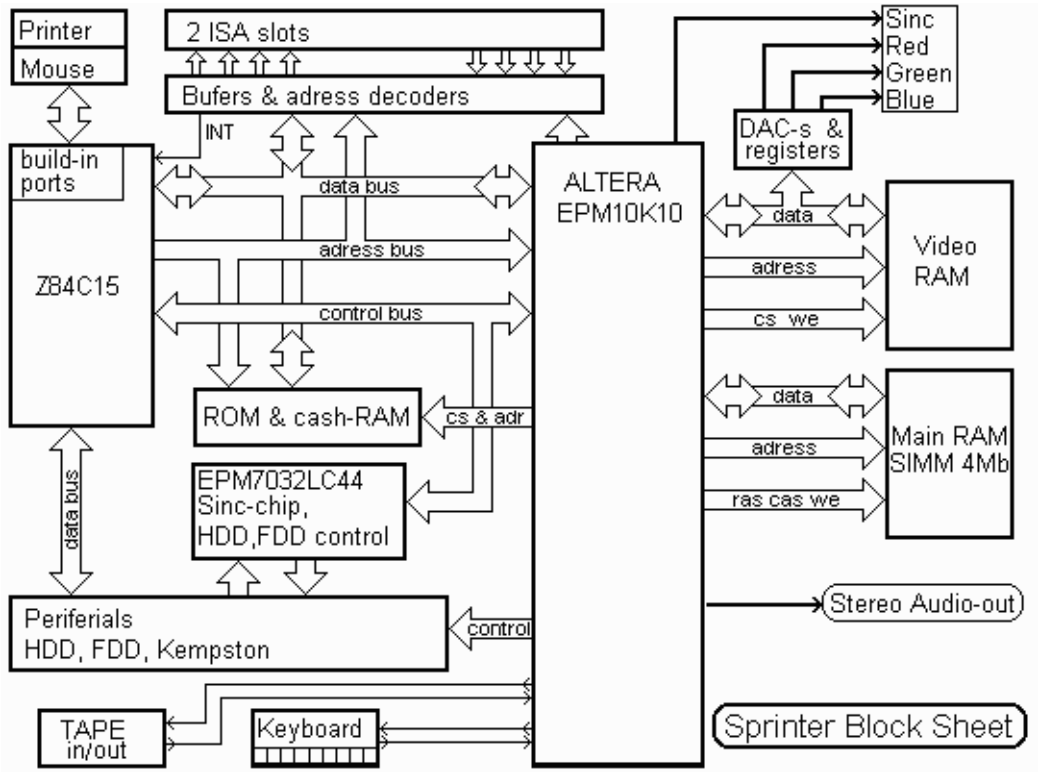
Схема компьютера основана на большой перепрограммируемой логической микросхеме. Подключение периферийных устройств через ППЛМ позволяет получить высокую гибкость машины по конфигурациям.

Программирование ППЛМ осуществляется непосредственно в момент включения, а так же при перезагрузке, что позволяет кардинально менять схему в ППЛМ непосредственно во время работы. Это сильно

выделяет архитектуру Sprinter-а из ряда существующих компьютеров, и многие понятия, присущие обычным машинам, меняют свой смысл. Фактически компьютер имеет гибкую архитектуру, в которой возможны изменения во многих частях схемы. Например, нельзя говорить о конкретных адресах портов подключения периферии, так как они могут быть изменены в одну секунду путем перепрограммирования ППЛМ и данных в ОЗУ, отвечающих за конфигурацию портов. Конкретные адреса появляются только в конкретных конфигурациях, например, в конфигурации ZX-Spectrum.

Перепрограммируемость схемы дает большую свободу фантазии программиста по конфигурации машины. Задумывая конкретную работу программист может определить в какой конфигурации ее можно сделать лучше, а, возможно, и придумать свою конфигурацию, которую затем можно реализовать в ППЛМ и включить перед запуском этой программы.

Блок-схема компьютера Sprinter:



Для простоты некоторые буферы и дешифраторы на схеме не указаны. Количество проводов в шинах так же условны.

Дальнейшее описание архитектуры является описанием конкретных конфигураций и их частей. Но перед этим следует сказать несколько слов о переключении конфигураций.

1.4 Загрузка конфигураций

В момент включения компьютера, а так же после нажатия на RESET вся информация, находившаяся в ППЛМ отвечающая за конкретную конфигурацию, стирается. ППЛМ переходит в режим ожидания загрузки блока данных схемы.

В этот момент процессор полностью отключен от какой либо периферии. В его адресное пространство памяти оказывается включено только ПЗУ и возможно подключение КЭШ ОЗУ. Любая запись в адресное пространство памяти процессора в этот момент приводит к записи данных в ППЛМ и программа в подключенной странице ПЗУ имеет только одну единственную цель – загрузить в ППЛМ данные конфигурации. В этой же странице ПЗУ находятся данные начальной конфигурации. Программа загрузки конфигурации проверяет флаг в КЭШ-памяти и, если он установлен, загружает в ППЛМ данные из КЭШ, если сброшен, то данные из ПЗУ. На этом основано переконфигурирование схемы компьютера.

Для изменения схемы используется функция BIOS, которая загружает в КЭШ-память блок данных конфигурации выставляет флаг, которым является текстовая строка «ACEX_30K_LOADING», после чего производится полный сброс, который осуществляется программно записью в специальную страницу памяти RESET_PAGE. Программа в ПЗУ, запускаемая по сбросу находит флаг «ACEX_30K_LOADING» и начинает загрузку данных в ППЛМ. Одновременно она затирает флаг, что предотвращает повторную загрузку новой конфигурации при нажатии на кнопку RESET и позволяет вернуться после «ручного» сброса в начальную конфигурацию. Затирание флага так же избавляет от мучений в случае подключения неправильной конфигурации во время экспериментов с программами. Нажатие на RESET всегда вернет схему в начальную конфигурацию.

Примечание: Внутренняя информация блока данных ППЛМ является закрытой информацией фирмы ALTERA. Кроме самих микросхем ППЛМ ALTERA поставляет и программное обеспечение для разводки схем внутри ППЛМ. К сожалению, эта программа не может работать на компьютере типа ZX-Spectrum и в ближайшем обозримом будущем не предвидится ее версия для Sprinter-а. Поэтому разработка новых конфигураций может производится только при наличии достаточно мощной машины (все делалось на Pentium-166) и программы разводки схем в ППЛМ – MAX-Plus II.

В связи с этим, в данный момент Sprinter имеет несколько конкретных конфигураций, три из которых (Sprinter-2, Sprinter-1, ZX-Spectrum+AY) записаны в ПЗУ, а остальные могут быть подгружены с дискеты или винчестера. Постоянно ведется совершенствование конкретных конфигураций и разработка новых. В плате Sp2000 эти конфигурации объединены в одной прошивке ППЛМ.

2 Краткое описание конфигураций

В этом параграфе дано краткое описание конфигураций компьютера Sprinter-97. Более полное описание с примерами программирования будет дано в части III. Для платы Sp2000 так же имеются эти конфигурации, поэтому описание приводится без изменений.

2.1 Конфигурация Sprinter-1

Включает в себя конфигурацию Spectrum-128/256, распределение памяти до 4Mb, расширенный экран с режимами Spectrum, Text-80x32, Graf-320x256x256, контроллер дисковода, контроллер IDE винчестера, контроллер клавиатуры AT, подключенной как ZX-Keyboard, 8-bit COVOX.

Эта конфигурация приближена к конфигурации ZX-Spectrum и позволяет работать на обычных спектрумовских программах и постепенно менять их под расширенные режимы экрана и памяти, а так же для работы с новыми устройствами.

2.2 Конфигурация Sprinter-2

Включает в себя конфигурацию Spectrum-128/256, распределение памяти до 4Mb, расширенный экран с режимами Spectrum, Text-80x32, Graf-320x256x256, контроллер дисковода, контроллер IDE винчестера, контроллер клавиатуры AT (построенный на внутреннем последовательном порте процессора), Accelerator, COVOX.

Конфигурация, как и Sprinter-1 приближена к спектрумовской, но имеет более жесткие требования к программам по совместимости. Позволяет использовать акселератор операций с основным и видео-ОЗУ. Акселератор ускоряет операции пересылки блоков данных и заполнения ОЗУ одним байтом до физического предела скорости основного ОЗУ. В последней версии он так же позволяет ускорить логическую обработку блоков данных по функциям AND, OR, XOR.

2.3 Конфигурация ZX-Spectrum+AY

Эта конфигурация максимально приближена к ZX-Spectrum-128/256 и включает в себя схему музыкального сопроцессора AY-3-8910. В этой конфигурации отсутствуют расширенные режимы экрана.

Третья версия схемы AY в соответствии с реальным сопроцессором включает в себя три генератора голосов, генератор шума и регуляторы амплитуды, генератор огибающей.

На данный момент в схеме формирователя огибающей обнаружена небольшая ошибка. В следующей версии AY предполагается данный недостаток исключить.

2.4 Конфигурация Game-1

Похожа на конфигурацию Sprinter-2. Акселератор не имеет логических функций. Для вывода звука имеет COVOX-Blaster (15KHz), позволяющим выводить звук поблочно и освобождать процессорное время для другой работы. Конфигурация ориентирована на использование в играх для Sprinter-a.

2.5 Конфигурация DooM

Является дальнейшим развитием конфигурации Game-1. Акселератор имеет дополнительную функцию по аппаратному растяжению/сжатию вертикальных и горизонтальных линий.

2.6 Конфигурация Video

Конфигурация Video похожа на Game-1, но имеет дополнительную возможность по передачи данных с HDD прямо в видео-память в момент чтения из HDD. В последней версии добавлен режим $GR-256-4x4$, позволяющий выводить видео-ролики с разрешением 160x128 на весь экран (аппаратное удвоение размера пикселя).

2.7 Особенности платы Sp2000

В плате Sp2000 описанные выше конфигурации сведены в две прошивки. Первая содержит Sprinter-1, Sprinter-2 и ZX-Spectrum+AY. Переключение между Sprinter-1 и Sprinter-2 осуществляется через системный порт, а ZX-Spectrum+AY фактически совпадает с конфигурацией Sprinter-1. Схема музыкального сопроцессора присутствует и доступна, как в Sprinter-1, так и в Sprinter-2.

Конфигурации Game-1, DooM, Video, так же будут сведены в одну прошивку и будут переключаться через системный порт.

Часть II
Блоки конфигураций компьютера

В этой части описываются блоки конфигураций. В каждой конфигурации возможно использование тех или иных блоков. О том, какие блоки в каких конфигурациях используются, будет сказано ниже.

3 Основная Память

3.1 Страницы, распределение памяти

Здесь так же приведено описание схемы распределения памяти Sprinter-97. (Для Sp2000 в данный момент она такая же.)

Структура схемы распределения основной памяти Спринтера является двухуровневой.

Основная память компьютера – 4Mb – разделена на блоки по 16kb, задаваемые однобайтовым номером. Эти блоки подключаются к логическим блокам Спектрумовской схемы распределения памяти. Что бы не путать понятия я буду называть блоки основной памяти физическими и писать их номера шестнадцатеричными цифрами – #00..#FF. Блоки спектрумовской схемы распределения памяти я буду называть логическими блоками и писать их номера в десятичной системе – 0..15.

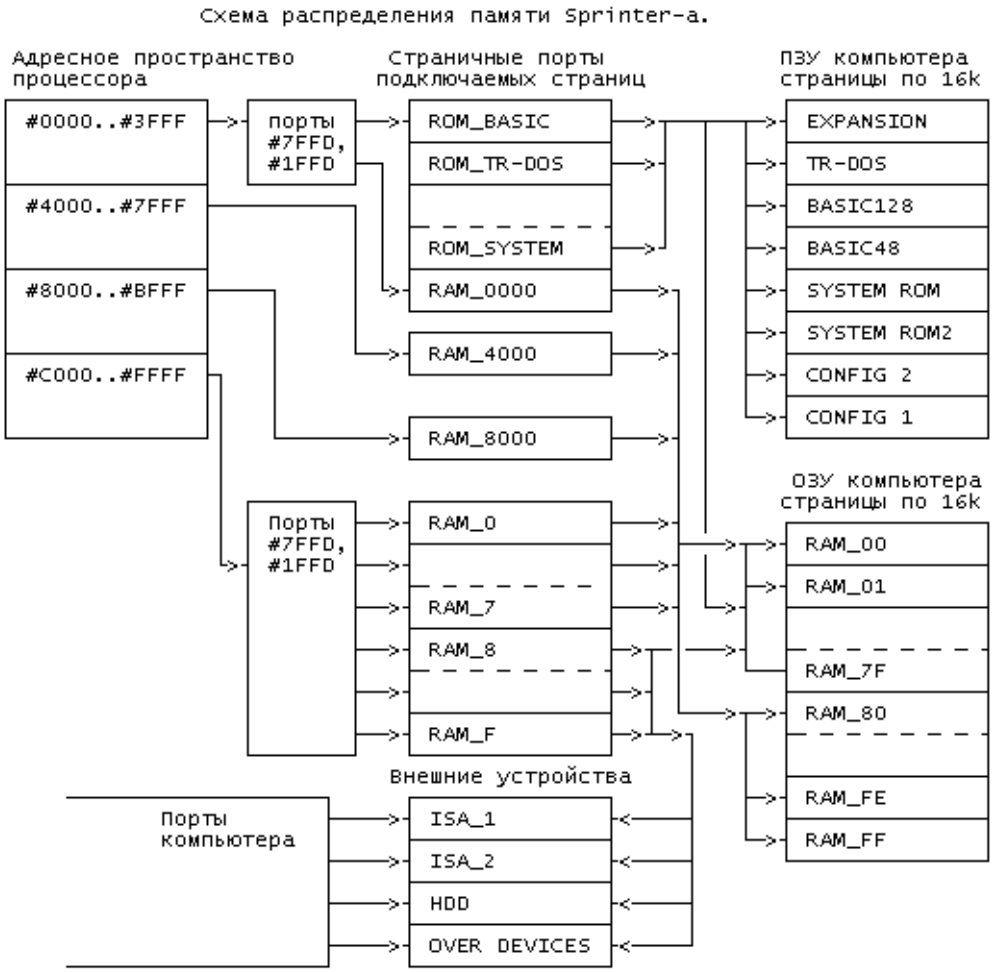
Адресное пространство процессора Z80 разделено на 4 окна с адресами #0000..#3FFF, #4000..#7FFF, #8000..#BFFF и #C000..#FFFF, которые для краткости я буду называть окнами с номерами 0, 1, 2 и 3. В каждом окне процессора располагается логический блок памяти согласно Спектрумовскому распределению. В окно 0 подключено ПЗУ, в окно 1 – страница 5, в окно 2 – страница 2, в окно 3 любая из страниц 0..7

для режима Spectrum-128, 0..15 для режима Scorpion-256 и 0..31 для режима Pentagon-512. Кроме этого, в режиме Scorpion-256 вместо ПЗУ может быть подключена логическая страница с номером 0.

Каждый логический блок имеет свой собственный физический номер подключаемой в него страницы ОЗУ. Логические страницы с номерами 0, 2 и 5 имеют дублированные номера, один для жесткого подключения в окна 0, 1, 2 процессора, второй для подключения в окно 3. В режиме Спектрума номера физических страниц, выставленные в этих блоках, совпадают.

Аналогичным образом на 16kb блоки разделено ПЗУ и КЭШ-ОЗУ. Физический номер страницы ПЗУ имеет значения #E0..#EF, а КЭШ – #F0..#FF. Реально на данный момент для адресации КЭШ ОЗУ используются только 2..1 биты из этого номера.

Каждая страница ПЗУ или ОЗУ имеет свой порт, в котором указывается физический номер страницы. Всего таких портов страниц памяти 32 (в режиме Pentagon-512 добавляются еще 16 портов). 16 портов отвечают за номера страниц ОЗУ, подключаемые в третье окно процессора. Еще три порта отвечают за подключение страниц ОЗУ в окна 0..2 Восемь портов используются для подключения различных страниц ПЗУ. Один порт – для подключения страницы КЭШ-а вместо ПЗУ. И один порт – это порт системной страницы, подключаемой на место ПЗУ сразу после сброса машины по клавишам Ctrl+Alt+Del.



Оставшиеся 3 порта страниц памяти остаются на данный момент в резерве. Схема распределения памяти позволяет подключить в адресное пространство процессора не только ОЗУ или ПЗУ, но и порты и память ISA карт, вставляемых в слот.

При подключении в адреса #C000..#FFFF скорпионовских расширенных страниц ОЗУ, на их место можно переадресовать слоты. Для этого надо просто записать в порт одной из этих страниц значение, соответствующее ISA-слоту, к которому необходимо произвести обращение. Это значение так же указывает

к чему ведется обращение, к портам или памяти слота (конкретные номера будут расписаны ниже).

В некоторых конфигурациях схема распределения памяти упрощается для освобождения ресурсов ППЛМ. Могут отсутствовать порты #1FFD и #7FFD, а так же упрощается схема работы с устройствами, отображаемыми на память.

3.2 Видео-область, спектрумовский и графический режим адресации

Запись в видео-ОЗУ может осуществляться двумя способами – через два режима адресации видеопамяти. Это графический и спектрумовский режимы адресации. Графический режим адресации кроме вывода графики используется и для вывода на текстовый экран. Спектрумовский режим используется для вывода в режиме Спектрума, а так же для задания знакогенераторов.

Видео-область основного ОЗУ и видео-ОЗУ – суть две различных области памяти. О видео-ОЗУ будет сказано ниже, а сейчас речь о видео-области основной памяти.

Графическая видео-область в основной памяти составляет 256Kb ОЗУ и соответствует страницам #50..#5F. При подключении этих страниц в любое окно процессора и при обращении к адресам этого окна мультиплексор адреса переключается в режим графической адресации. В этом режиме биты 3..0 номера страницы не влияют на адрес памяти, а определяют особый режим вывода, о котором будет сказано ниже. Адрес памяти в графическом режиме задается номером, засылаемым в RGADR (он же PORT_Y), о котором так же будет сказано ниже, и десятью младшими битами адреса процессора. В графической видео-области содержится копия видеоданных, посылаемых на экран в графическом режиме (как будет ясно далее, копия не точная).

В спектрумовском режиме адресации видео-область основного ОЗУ располагается в адресах #4000..#5FFF и #C000..#DFFF для спектрумовских страниц 5 и 7. Любая физическая страница основного ОЗУ (кроме #50..#5F) подключенная к этим областям памяти процессора может стать видео-областью. В этом режиме нет привязки страницы основного ОЗУ к странице видео-ОЗУ. Последняя определяется через RGADR.

4 Видео-память

Видео-память Спринтера является теневой памятью. Весь вывод в видео-память производится параллельно с выводом в основную (за исключением некоторых особых случаев). При считывании из области видео-данных считывается информация из основной памяти, а видео остается недоступной.

Недоступность видео-памяти не является принципиальным ограничением и есть признак конфигурации. Перепрограммирование ПЛМ в позволяет получить доступ к видео-ОЗУ, но на данный момент эта возможность не используется ни в одной прошивке, так как практически не особо требуется.

В работе видеоконтроллера используются два основных режима адресации - спектрумовский и графический. В спектрумовском режиме видео-ОЗУ может оказаться тенью любой страницы основного ОЗУ, кроме страниц #50..#5F. Это позволяет при необходимости иметь множество копий спектрумовского экрана, например, для случая разработки многозадачного Спектрума.

В графическом режиме видео-ОЗУ является тенью одного 256-килобайтного блока – страниц #50..#5F. Жесткое закрепление графического образа видео-памяти в основном ОЗУ обусловлено необходимостью иметь точные копии данных видеорежимов для работы компьютера.

4.1 Спектрумовский режим адресации

В Спектрумовском режиме вся видеопамять разбивается на 32 блока по 8 килобайт. Адрес блока определяется номером, записываемым в RGADR. Биты 4..0 являются номером блока, бит 7 разрешает использование 16-килобайтовых страниц, бит 6 запрещает вывод на экран в спектрумовском режиме. Бит 0 вместе с битом 1 порта #7FFD определяет младший бит номера спектрумовского блока видео-ОЗУ. Таким образом достигается подключение двух блоков спектрумовского экрана для страниц 5 и 7.

Для примера, запись в RGADR номера #0C приведет к подключению к адресам #4000..#5FFF блока номер 12, а к адресам #C000..#DFFF (при условии, что установлена спектрумовская страница 7) блока номер 13. Запись в RGADR нечетного значения #0D приведет к подключению в #4000..#5FFF блока номер 13, а в #C000..#DFFF (к 7-й странице) блока номер 12.

Запись в RGADR значения #80 приведет возможности иметь доступ в спектрумовском режиме сразу к 16kb – двум блокам видео-ОЗУ с номерами 0 и 1 в адресах #4000..#7FFF. Реально этот режим не использовался и не проверялся. Он был сделан для возможной совместимости со спектрумовским режимом 512 точек в строке, но остался невостребован.

Если при работе какой либо программы требуется больше памяти, но не требуется вывод в спектрумовском режиме, RGADR рекомендуется устанавливать в значение #C0..#FF. В этом случае вывод в видео-ОЗУ в Спектрумовском режиме не производится.

4.2 Графический режим адресации

В графическом режиме адресации видео-ОЗУ используется тот же RGADR. Подобная реализация признана неудобной и в следующем варианте Sprinter-а предполагается разделить порты для графической и спектрумовской адресации. Неудобство заключается в том, что закрытие спектрумовского режима вывода осуществляется только при значениях этого порта больше #C0, а в графическом выводе используются все значения порта. Из-за этого приходится постоянно следить за положением RGADR и перед записью в адреса #4000..#7FFF устанавливать его в #C0, либо использовать для графического вывода первое окно, что так же не всегда удобно.

Видео-ОЗУ в графическом режиме адресации разбивается на 256 строк по 1024 байта. Байты в строках адресуются линейно младшими битами адреса процессора, а номер строки записывается в RGADR.

Расположение строк видео-ОЗУ таково, что спектрумовская адресация оказыватся как бы "поперек"строк. Это сделано для возможности вывода как горизонтальных, так и вертикальных линий с помощью команды типа LDIR. В графической адресации выводятся горизонтальные линии, в спектрумовской - вертикальные. Реально подобный вывод не использовался, так как оказалось проще выводить вертикальные линии с помощью акселератора (об этом ниже).

4.3 Подрежимы вывода графической адресации

Как было указано, страницы #50..#5F являются графическими. Адрес данных видео-ОЗУ, а так же основного ОЗУ в графической адресации зависит от порта RGADR (PORT_Y), и при установке любого значения из диапазона #50..#5F не меняется. Это используется для введения дополнительных подрежимов вывода, которые используются для ускорения работы со спрайтовой графикой, а так же для удобства программирования в некоторых иных случаях.

Два подрежима задается 2-м и 3-м битами номера страницы из #50..#5F.

Бит 3 отвечает за разрешение записи байта #FF. Если при выводе на экран использовать страницу #58..#5F, то в процессе записи проверятся, не равен ли записываемый байт значению #FF. Если равен, то запись не производится.

Этот режим можно использовать для вывода спрайтов с «прозрачным» цветом (прозрачный цвет #FF). Фигурный спрайт выводится как прямоугольник, а все точки, которые не принадлежат спрайту в этом прямоугольнике записаны байтами #FF.

Бит 2 отвечает за разрешение записи в основное ОЗУ параллельно с видео. Если при выводе использовать страницы #54..#57 или #5C..#5F, то вывод будет осуществляться только в видео-ОЗУ. В основном останутся те данные, которые были записаны туда ранее.

Этот режим используется для временного вывода на экран, например для вывода курсора мыши без необходимости запоминания данных под курсором. Они остаются в основном ОЗУ.

Как видно, биты 2 и 3 действуют независимо. Это позволяет комбинировать оба подрежима так как необходимо программисту.

ВНИМАНИЕ! При программировании вывода на графический экран стоит придерживаться правила, что в биты 0 и 1 номера страницы записываются нули. Это позволит в будущем беспрепятственно использовать эти программы в конфигурациях, где эти биты будут выполнять новые дополнительные функции.

4.4 Распределение Video-RAM

Распределение видео-ОЗУ является частично жестким, частично программным. Жеская часть распределения отвечает где находятся данные видео-режима и данные палитры (более конкретно о них будет сказано

ниже). Программное распределени отвечает, где находятся данные для вывода в графических, текстовых и спектрумовском режимах, открываемых функциями биоса.

Пока для дальнейшего объяснения пока достаточно примерного распределения видео-ОЗУ. Данные палитры находятся в блоках 30-31 спектрумовской адресации или, что то же самое, на концах линий графической адресации – в адресах #03E0..#03FF. Данные режимов экрана находятся в блоках 24..29, или в терминах линий – в адресах #0300..#039F. Область линий #03A0..#03DF является зарезарвированной и ее использование в программах не рекомендуется.

Блоки 0..23 или линии по адресам #0000..#02FF обявляются свободно используемыми в программах. При использовании функций биоса именно в этой области располагаются данные графических и спектрумовских экранов. Спектрумовский экран использует блоки 0 и 1 (#0000..#003F). Первый графический экран использует блоки 2..11 (#0040..#017F). Второй графический – блоки 12..21 (#0180..#02BF). Блоки 22 и 23 (#02C0..#02FF) используются для хранения знакогенераторов текстового режима. Сами же данные текстового режима находятся в области режима экрана и об этом будет сказано ниже.

4.5 Структура экрана, режимы экрана

Выше была описана адресация видео-ОЗУ так, как оно выглядит для процессора. Вывод же изображения на телевизор осуществляется с помощью записаной в ПЛМ видео-карты, которая позволяет менять режимы вывода простой перезаписью информации в видео-ОЗУ. Единственный порт, который меняет выводимое изображение – порт RGMOD, бит 0 которого переключает страницу режима и таким образом полностью меняет содержимое видео-экрана.

Режим экрана задается данными в видео-ОЗУ. Весь экран представляет собой набор квадратов размером 8x8 точек режима разрешения 320x256. При увеличении разрешения количество точек в квадрате увеличивается до 16x8.

Режим экрана для каждого квадратика прописывается в области 24..29 блоков видео-ОЗУ. Весь экран описывается страницей режима, которая может быть переключена битом 0 порта RGMOD. Каждому квадратику на одной странице режима соответствует 8 байт, часть из которых не используется (зарезервировано). Реально используется от 2-х до 6-ти байт.

Весь экран представляет собой набор из 56x39 (56x40) квадратиков, что соответствует телевизионным режимам 312(320) полных линий на кадр. Видимая область составляет 40x32 квадратика. Остальные соответствуют квадратикам, попадающим в область бордера, обратного хода луча и синхроимпульсов. Принципиально возможно использование бордерных квадратиков для вывода изображения, но это не рекомендуется, так как не все телевизоры способны показывать эту область.

Для подробного объяснения адресов режима я воспользуюсь следующими обозначениями:

PM – страница режима (значения 0 или 1).
SQ_{ab} – квадратик с номером по горизонтали a и номером по вертикали b.
$Line_1$ – номер графической линии для режима.
$Line_2$ – номер графической линии для подрежима.
LA – адрес точки в графической линии.
$Mode_0$ – первый байт режима.
$Mode_1$ – второй байт режима.
$Mode_2$ – третий байт режима.
$Mode_4$ – четвертый байт режима.

Для каждого квадратика SQ_{ab}:
$Line_1 = 1 + 2 \cdot a + \#80 \cdot PM$; $Line_2 = Line_1 + 1$; $LA = \#0300 + 4 \cdot b$.

Параметры a и b принимают значения от 0 до 55 и от 0 до 39 соответственно. $SQ_{0.0}$ – квадратик в верхнем левом углу видимой области графического экрана экрана. $SQ_{4.4}$ – самый верхний левый квадратик спектрумовского экрана. $SQ_{39.31}$ – нижний правый квадратик графического экрана. Как уже и было сказано, квадратики с номерами a больше 39 и b больше 31 соответствуют бордеру и области синхроимпульсов. Левый и верхний бордер соответствует квадратикам с $a = 55$ и $b = 38(39)$ соответственно.

Положение байтов $Mode_i$ в видео-ОЗУ соотстветсуют одинаковому $Line$ и адресу в строке $LA + i$. Основной режим определяется в $Line_1$, а подрежим используется при определении основного режима с

разрешением 640 точек по горизонтали. Если в основном режиме появилось указание на 640 точек, то после вывода первых 8-ми точек, производится считывание дополнительного режима, с $Line_2$, которое переопределяет режим второй половинки квадратика. Это позволяет в текстовом 80-символьном режиме иметь независимое программирование символа и знакогенератора.

Режим вывода в каждый момент времени полностью определяется считанными байтами $Mode_{0..3}$. Байт $Mode_3$ реально не используется и зарезервирован, поэтому режим определяется тремя байтами.

Собственно, сам режим определяется байтом $Mode_0$, байт $Mode_1$ является просто байтом адреса данных, а $Mode_2$ используется в спектрумовском режиме как адрес атрибутов, а в текстовом собственно как атрибут символа.

Распишу все режимы:

$ZX-40$ – текстовый режим 40 символов в строке или 1 символ на знакоместо. Используется в спектрумовском режиме. На экране оказываются три различных знакогенератора, которые расположены в памяти ровно так, как полагается для Спектрума.

$ZX-80$ – текстовый режим 80 символов в строке или 2 символа на знакоместо. Отличительной особенностью является возможность на одном экране иметь до 36 знакогенераторов, указываемых для каждого символа.

$GR-256-8$ – графический режим 320x256 точек, 256 цветов. Он же 8x8 точек 256 цветов на одно знакоместо. В этом режиме (а так же в GR-16-16) экран можно представить как «текстовый»40-символьный, в котором каждый символ имеет графический образ в 64 байта. Всего таких символов 3072. При определенном задании режима графические символы сливаются в одно графическое поле, на котором достаточно просто рисуются линии и другие графические примитивы.

$GR-16-16$ – графический режим 640x256 точек, 16 цветов. Он же 16x8 точек 16 цветов на одно знакоместо.

Следует отметить, что любой из вышеуказанных режимов устанавливается в каждый квадратик индивидуально. Таким образом на разных частях экрана могут одновременно присутствовать различные режимы вывода.

4.6 Спектрумовский режим, он же текстовый

$Mode_0$:

bits 3..0 – старшие биты (4..1) адреса блока, откуда выводятся данные. Младший бит определяется 3-м битом порта #7FFD.

bit 4 – "1"признак режима. Значение "0"соответствует графическому режиму.

bit 5 – признак разрешения "0"для 640 точек, "1"для 320 точек.

bit 6..7 – старшие биты адреса в блоке (bit 12..11). Они одинаковы как для байта пикселей так и для байта атрибутов.

Так как в спектрумовском экране используется только три комбинации битов 6..7, четвертая комбинация определяет особый режим – вывод бордера и синхросигналов. Таким образом комбинация "1111"в старших битах режима $Mode_0$ особая. При такой комбинации назначения младших битов изменяется, так как для вывода бордера не требуется никакого адреса данных:

bit 3..2 – значение "11"генерирует в этом квадратике сигнал Blank - гашение.

bit 0 – значение "1"в присутствии сигнала Blank генерирует прерывание для процессора на восьмой линии квадратика. Именно программированием этого бита в нужном квадратике осуществляется "подгонка"сигнала INT под стандарт Pentagon или Scorpion.

$Mode_1$:

Как уже было сказано, этот байт определяет адрес данных в видео-ОЗУ, выводимых в текущий квадратик. В спектрумовском режиме он соответствует восьми младшим битам адреса в блоке и совместно с битами 6..7 байта $Mode_0$ определяет адрес спектрумовского символа в блоке. Недостающие 3 бита (bit 10..8) берутся из счетчика линий в квадратике.

$Mode_2$:

Определяет младшие восемь бит адреса атрибута. В спектрумовском режиме устанавливается равным $Mode_1$, а в текстовом используется как байт атрибута символа. При выборке атрибута производится специальное замешивание битов 6..7 байта $Mode_0$ таким образом, что бы атрибуты оказались в старших адресах блока по известному спектрумовскому закону.

Как использовать этот режим для текстового режима?

Для этого в один из спектрумовских экранов записывается знакогенератор. Прописывается так, что на спектрумовском экране он выглядит как набор ASCII из 256-ти символов, выведенных последовательно в 256 знакомест. У этих символов прописывается атрибут просто равный номеру символа. После этого байты $Mode_1$ и $Mode_2$ оказываются просто равными ASCII коду и атрибуту выводимого символа соответственно. А байт $Mode_0$ указывает на знакогенератор и в в принципе может определить 36 различных знакогенераторов. 3 за счет битов 7..6, еще 3 за счет битов 3..2 (комбинация "11"в этом месте попадает в область режима экрана и палитры, куда не стоит вписывать знакогенератор), и еще 4 за счет битов 1..0. Простым премножением получается $3 \cdot 3 \cdot 4 = 36$ знакогенераторов.

Страница режима устроена таким образом, что бы при выводе линии байты атрибута и цвета одного символа оказались рядом, и символы, расположенные в одной строке выводятся подобно выводу вертикальной линии символов и рядом расположенной вертикальной линии атрибутов. При использовании акселератора вывод строки оказывается очень быстрым (около миллиона символов в секунду).

4.7 Графический режим

Прежде чем расписывать байты режима объясню, как адресуются графические квадратики. Все видео-ОЗУ, разбитое на блоки или линии, можно представить в виде квадратиков 8x8 точек. На линии в 1024 байта умещается 128 квадратиков, каждый из которых по вертикали занимает 8 линий и, таким образом, по вертикали все видео-ОЗУ разбирается на 32 квадратика. Всего 4096 квадратиков, для адресации которых необходимо 12 бит. Они берутся из $Mode_0$ (четыре младших бита) и $Mode_1$ после того, как битом 4 байта $Mode_0$ определен графический режим.

$Mode_0$:

bits 3..0 – старшие биты (4..1) адреса блока, так же как и в спектрумовском, но младший бит номера блока берется из 2-го бита $Mode_1$. Они же формально представляют собой старшие биты номера квадратика по горизонтали.

bit 4 – "0"признак режима (графический режим).

bit 5 – признак разрешения "0"для 640 точек, "1"для 320 точек.

bit 6..7 – номер палитры. В каждом квадратике можно задать одну из четырех графических палитр.

$Mode_1$:

bit 1..0 – Дополнительный адрес данных в блоке. Адрес квадратика по горизонтали.

bit 2 – младший бит блока.

bit 7..3 – Адрес квадратика по вертикали.

Подобное "неправильное"разбиение адреса квадратиков возникло в результате оптимизации схемы "по объему ПЛМ". Ввиду нехватки места было решено, что лучше пожертвовать линейностью адресации в режиме экрана в пользу возможности иметь дополнительные возможности. В новой версии компьютера возможно устранение этой нелинейности.

Графический режим 640 точек использует ту же схему адресации квадратиков. при этом каждый байт рабивается на две точки и таким образом из режима 8x8 256-ти цветных точек получается режим 16x8 16-ти цветных. Первыми выводятся младшие 4 бита, вторыми – старшие.

4.8 Палитра

Устройство палитры достаточно просто. Каждый момент при выводе точки, с одной из линий считывается 3 байта и выводится в 3 регистра с 3-мя ЦАП-ами на выходе. Таким образом, образуется 3 цвета Red, Greeb & Blue, которые определяют один из 16 млн. возможных цветов.

Какие именно три байта выводить определяется схемой видео-карты. Все три байта выводятся всегда из одной области – с концов линий – адресов #03E0..#03FF. Номер линии соответствуют номеру цвета для 256-ти графического цветного режима или номеру атрибута для текстового и спектрумовского режима. В указанных адресах умещается 8 палитр по 256 цветов, считая по 1kb на каждую палитру. Реально из этого килобайта используется только 768 байт. Каждый четвертый байт никуда не выводится и принципиально схемотехнически не может попасть на регистр ЦАП-а. Четыре из этих восьми палитр используются для графических режимов и оставшиеся четыре для текстовой палитры.

Про палитру в тексте следует сказать отдельно. Атрибут формально задает один цвет из палитры и, соответственно, четыре цвета из четырех палитр. из этих четырех цветов на каждую точку попадает только один. Выбор производится по двум битам – FLASH и биту из регистра сдвига в котором находятся восемь точек, считанные из знакогенератора. Таким образом, для каждого атрибута задается цвет бумаги, цвет символа, цвет мерцания бумаги и цвет мерцания символа. Если, например, цвет мерцания бумаги совпадает цветом бумаги, то мерцания на экране не наблюдается. Аналогично для цвета символа. Подобное решение позволяет любую кодировку цвета для каждого атрибута и разрешить проблему с различиями атрибутов в спектрумовском и IBM-ском вариантах. Просто в момент перехода с одного стандарта на другой в палитру загружаются необходимые данные.

С графической палитрой все просто. Байт точки попадает на адрес линии для конкретной палитры и соответствующий ей цвет попадает на ЦАП-ы. В 16-ти цветном режиме используются 16 младших цветов палитры.

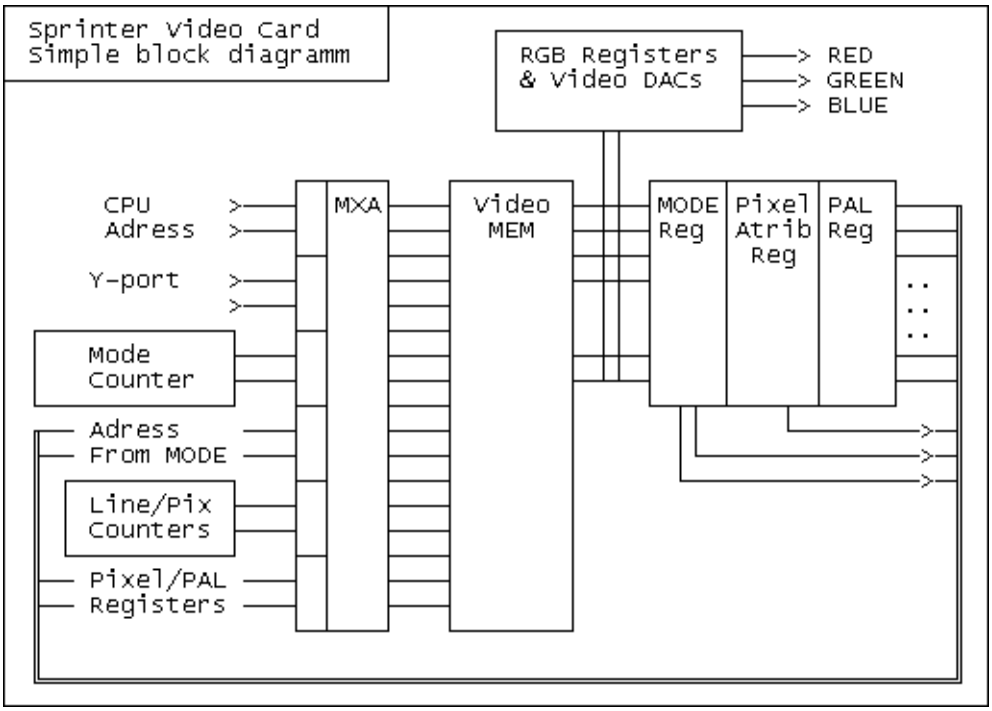
#03E0..#03E2 – графическая палитра 0
#03E4..#03E6 – графическая палитра 1
#03E8..#03EA – графическая палитра 2
#03EC..#03EE – графическая палитра 3
#03F0..#03F2 – текстовая палитра 0 – цвет бумаги
#03F4..#03F6 – текстовая палитра 1 – цвет символа
#03F8..#03FA – текстовая палитра 2 – цвет мерцания бумаги
#03FC..#03FE – текстовая палитра 3 – цвет мерцания символа

Байты расположены в порядке Blue Green Red.

4.9 Процесс вывода на экран

Что бы связать воедино все, что было сказано выше, распишу процесс вывода на экран, происходящий внутри ПЛМ.

Упрощенная блок-схема видео-контроллера:



Тактовый генератор задает частоту 42MHz. Первым делом эта частота делится на 3, а затем на 2, получая 14 и 7 MHz для вывода пикселей. 14MHz в режиме 640 точек и 7MHz в режиме 320 точек. 7MHz делится еще раз на 8 получая номер точки в квадратике. Затем получившаяся частота делится на 56. Этот делитель является и счетчиком знакомест, номер которого подается как часть адреса режима экрана. Полученная после деления на 56 частота является частотой синхронизации строк. Она делится на 8 вертикальным счетчиком строк знакоместа. Далее деление частоты продолжается и для получения 312 или 320 линий на кадр используется управляемый счетчик – он же счетчик знакомест по вертикали и часть адреса режима экрана. И последний счетчик делит полученную частоту кадров на 32 что бы получить сигнал FLASH.

Данные счетчиков знакомест по горизонтали и вертикали подаются как часть адреса на видео-ОЗУ. Остальная часть адреса соответствует тому положению, где находятся данные видео-режима. И, таким образом, из видео-ОЗУ считываются данные режима, а именно 4 байта $Mode_i$. Эти байты определяют в каком режиме выводить данные на экран и из какого места их брать. В соответствии со считанными байтами $Mode_i$ из видео-ОЗУ производится следующая выборка, а именно выборка байта знакогенератора и атрибута для текстового режима или выборка байта цвета для графического. Из этих байтов (а в случае необходимости и из сигнала FLASH) формируется необходимый адрес цвета в палитре и производится следующий цикл считывания из памяти, при котором данные сразу же записываются в цветовые регистры с ЦАП-ами, которые и формируют сигнал цвета для монитора.

Еще раз распишу немного подробнее, по тактам.
Тактовая частота – 42MHz. Период одного пикселя – 6 однотактовых циклов.
Циклы распределяются следующим образом:

$TAKT$ 1 – выборка байта пикселей текстового режима в графическом режиме – холостой ход;
$TAKT$ 2 – выборка текстового атрибута, он же выборка байта пикселя для GRAF-Modes;
$TAKT$ 3 – выборка 3-х байтов палитры для режимов 640 точек, в 320 - холостой ход;
$TAKT$ 4 – выборка слова режима экрана выборка производится один раз в четыре периода;
$TAKT$ 5 – цикл записи от процессора, данные пишутся только когда нужно;
$TAKT$ 6 – выборка 3-х байт палитры с записью в RGB регистры.

Все начинается с такта 4. Адрес для режима определяется счетчиком вертикали и горизонтали. Счита-

ются по 8 точек 320-го режима по вертикали и по 8 строк по горизонтали. Считанные данные помещаются в регистры режима $Mode_i$, описанные выше.

С пятым тактом все ясно. За исключением того, что для графического и текстового режимов изменяется адресация ОЗУ Изменение задается номером страницы Страницы 50..5F считаются графическим ОЗУ. Следует отметить, что понятие графического и текстового режима адресации не совсем верно отражает суть. Вывод данных в экранное ОЗУ может осуществляться в обоих режимах независимо от действительного режима экрана.

6-й такт. Чтение палитры. В текстовом режиме адрес палитры составляет 10 бит. 8 бит – байт атрибута. 1 бит – пиксель. 1 бит – FLASH. Этот бит фактически переключает две палитры с частотой FLASH. Если палитры для данного атрибута совпадают то FLASH не наблюдается.

В графическом режиме адрес палитры составляет 8 бит байта пикселя и 2 бита номера палитры, считанного в 4-м такте, но задержанного на 1 период для синхронного изменения режима и палитры в знакоместе.

Еще один бит адреса используется для разделения палитр текстового и графического режимов. Оставшиеся адреса ОЗУ устанавливаются в "1"и вся палитра оказывается на концах 1024-х байтовых линий графического режима.

Такт 1. Здесь начинает действовать новый режим, считанный в такте 4. Адрес ОЗУ является адресом байта пикселей текстового режима. Байт пикселей записывается в сдвиговый регистр и первый выдвинутый бит начинает действовать в 6-м такте.

Такт 2. Выбирается байт атрибута. Адрес отличается от адреса байта пикселей по известному Спектрумовскому закону. Если перед этим в 4-м такте был считан графический режим, то Адрес атрибута становится другим и соответствует байту пикселя графического режима. Регистр атрибута в графическом режиме является регистром байта пикселя.

Такт 3. Чтение палитры для режима 640 точек. Следует заметить что в графическом режим 640 точек, 4 старших бита регистра пикселя маскируются нулями и в этом такте в младших битах адреса оказываются те четыре бита, которые были замаскированы в 6-м такте. Так организуется 16 цветов на точку.

5 Звуковой выход

Звуковой выход платы Sp2000 представляет собой двухканальный 16-тибитный ЦАП - TDA1543, подключенный к ППЛМ. В данный момент реально используется 10 бит, для одновременного вывода 8-bit COVOX-а и сигнала AY-3-8910.

5.1 AY-3-8910

В ППЛМ платы Sp2000 вписывается схема, содержащая в себе схему аналога музыкального сопроцессора AY-3-8910/8912. Его программирование осуществляется по стандартным описаниям.

5.2 Бипер/Covox

Стандартный спектрумовский бипер (бит 5 порта #FE) выведен через ту же схему, что и Covox, и AY. (TDA1543) Программируется стандартно.

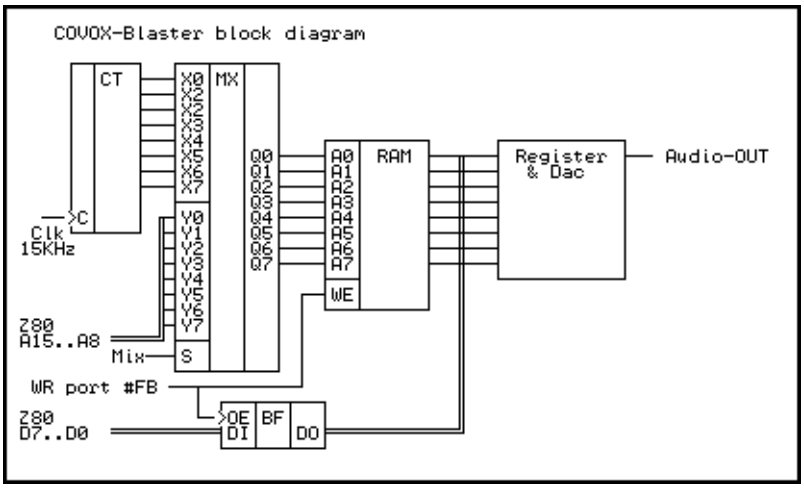
Covox – 8-бит – позволяет выводить WAV сигналы. В стандартной конфигурации Covox подключен на порт #FB и #4F. Вывод последовательности байтов в любой из этих портов приведет появлению сигнала на звуковом выходе.

В стандартной конфигурации на выходы левого и правого канала выдается один и тот же сигнал (монофоническое звучание).

5.3 COVOX-Blaster (CBL)

Covox-Blaster (CBL) – Covox с буферным ОЗУ.

Подключение буферного ОЗУ осуществляется по следующей схеме:



Счетчик работает на частоте 15 или 22 килогерца, в зависимости от состояния порта конфигурации Covox-а. Адрес мультиплексируется на момент записи в порт из процессора, все остальное время данные из ОЗУ записываются в регистр ЦАП-а.

Ввод байтов в буферное ОЗУ осуществляется командой OTIR (OUTI), что позволяет ускорить вывод звука и создать достаточно большие паузы для работы других частей программы. Так как при использовании команды OTIR регистр B (который попадает на A15..A8 процессора) уменьшается, для нормальной работы CBL счетчик считает «назад».

Для контроля за работой Covox-Blaster-а используется бит 7 порта #FE, в который выводится старший бит счетчика. Блок ОЗУ 256 байт условно разбит на две банки по 128 байт, и бит 7 порта #FE указывает какая из банок ОЗУ выводится в ЦАП в конкретный момент времени. Это используется программой вывода для определения, нужно ли подгружать следующие 128 байт в буфер.

В плате Sp2000 COVOX-Blaster включен в основную прошивку и включается через порт управления CBL - #4E. Запись в этот порт значения #80 приводит к включению режима CBL, #00 - включение обычного COVOX-а. Другие биты порта #4E имеют значение и их следует выставлять в 0 для получения описанного выше режима CBL. В дальнейшем этот порт будет устанавливать режимы Stereo, 8/16-bit и частоту.

```
;************************************************************************
;
;       Пример программы для Covox-Blaster-a.
;
;************************************************************************
CLEAR_COVOX:            ; программа для очистки буфера ОЗУ и
                        ; отключения звука
        LD A,80H        ; значение, эквивалентное нулю на выходе Covox
        LD BC,0FBH      ; порт Covox-Blaster-a
CLEAR_CBL:
        OUT (C),A
        DJNZ CLEAR_CBL
        XOR A
        LD (SND_P),A    ; установить в страницу звука 0 (нет звука)
        RET

;************************************************************************
SOUND_START:    ; программа инициализации Covox-Blaster-a
;========================================================================
;       здесь должна располагаться программа, которая
;       произведет рассчет первой страницы данных для COVOX-бластера и
```

```
;       адреса данных. Страница и адрес соответственно в регистры A и HL
;=============================================================================
        LD (SND_A),HL   ; запомнить состояние адреса звука
        LD (SND_P),A    ; запомнить новую страницу WAV-данных
        RET

SND_A   DB 0
SND_P   DW 0
;*****************************************************************************
SOUND:
        PUSH AF
        LD A,(SND_P)    ; проверка, что страница WAV-данных не равна 0
        AND A
        JR Z,RET_ALL    ; иначе выход -- нет звука
        PUSH HL
SND_MORE:
        IN A,(0FEH)     ; бит 7 порта #FE указывает состояние 7-го бита
                        ; счетчика выводимого байта (банк 0/1)
        XOR 0           ; запомненное состояние Covox адреса
COV_ADR EQU $-1
        AND 80H         ; проверить, переключение банки 128 байт
        JP NZ,NO_LD_SND ; если изменения не было, вернуться.

        LD A,(SND_P)
        AND A
        JR Z,RET_ALL

        PUSH DE
        PUSH BC

        LD A,(COV_ADR)  ; взять адрес Covox-a.
        CPL             ; инвертировать
        LD B,A          ; запомнить в B

        LD HL,(SND_A)   ; взять адрес WAV-данных
        LD C,0FBH       ; порт Covox-Blaster-a

        IN A,(PAGE3)    ; запомнить состояние PAGE3
        LD E,A
        LD A,(SND_P)    ; взять номер страницы WAV-данных
        OUT (PAGE3),A   ; переключить PAGE3

        LD D,16         ; повторять 16 раз
L_DDX:
        OUTI            ; выводить в Covox-Blaster
        OUTI            ; (OUTI работает несколько быстрее, чем OTIR)
        OUTI
        OUTI
        OUTI
        OUTI
        OUTI
        OUTI
        DEC D
        JR NZ,L_DDX
```

```
        LD (SND_A),HL   ; запомнить состояние адреса звука

        LD A,H          ; проверить, что адрес не дошел до конца страницы
        AND A
        JP NZ,NO_SNDP   ; если не дошел, идти на выход

        LD A,E          ; вспомнить страницу PAGE3
        OUT (PAGE3),A
;=======================================================================
;       здесь должна располагаться программа, которая
;       произведет рассчет новой страницы данных для COVOX-бластера и
;       адреса данных. Страница и адрес соответственно в регистры A и HL
;=======================================================================
        LD (SND_A),HL   ; запомнить состояние адреса звука
        LD (SND_P),A    ; запомнить новую страницу WAV-данных
        JR NO_SNDP1
NO_SNDP:
        LD A,E          ; вспомнить страницу PAGE3
        OUT (PAGE3),A
NO_SNDP1:
        POP BC
        POP DE
NO_LD_SND:
        POP HL
RET_ALL:
        POP AF
        RET
```

5.4 Sprinter-Sound-Card (SSC)

Разработанная на данный момент Sound-Card для Спринтера не поддерживается ни в одной прошивке ввиду довольно большого занимаемого объема ПЛМ. В плате Sp2000 объем уже достаточен и конкретная прошивка с SSC находится в планах.

Основные параметры SSC:

2 выходных канала (Stereo-OUT)
Шестнадцать 8-битных голосов
256-byte Wave-Tables своя для каждого голоса
8-bit регулятор амплитуды для каждого голоса и каждого канала

Программирование в данном описании не приводится ввиду возможной полной переработки схемы SSC.

6 Акселератор операций с ОЗУ

Акселератор операций с ОЗУ предназначен для ускорения операций по пересылке данных или по заполнению ОЗУ одним байтом. Акселератор присутствует в чисто Sprinter-овских конфигурациях и поэтому никак не мешает работе обычных Spectrum-овских программ.

Основой акселератора является быстрое внутреннее ОЗУ в ППЛМ. Операции по пересылке данных производятся путем записи блока данных в это внутреннее ОЗУ, а затем копировании его в нужное место памяти из этого ОЗУ. После одной записи копирование может производиться несколько раз и таким образом можно производить заполнение экрана текстурами.

Для заполнения экрана одним цветом используется другой режим акселератора. В нем вместо копируемого блока данных из внутреннего ОЗУ производится запись данных с шины процессора, которые в этот момент не изменяются.

Блок данных, записываемый в ОЗУ акселератора может иметь различную длинну из диапазона 1..256 байт.

Управление акселератором производится непосредственно из программы. Для этого изпользуются команды процессора, которые фактически являются операциями типа NOP. Это команды LD A,A; LD B,B; LD C,C; LD D,D; LD E,E; LD H,H, LD L,L

```
Назначение команд следующее:
LD B,B - выключить акселетарор.
LD D,D - включить акселератор в режим приема байта размера блока
         далее следует команда типа LD A,dat, где dat и будет новым
         размером блока. Если размер блока был установлен ранее,
         его можно не устанавливать.
LD C,C - Операция Fill - заполнение одним байтом. Последующая
         команда типа LD (HL),A приведет к заполнению указанного
         ранее количества байт значением A
LD E,E - Операция Fill для графического экрана - заполнение
         вертикальных линий.
LD H,H - rezerved
LD L,L - копирование блока. Последующая команда типа LD A,(HL)
         приведет к заполнению ОЗУ акселератора данными из адреса (HL),
         а команда типа LD (DE),A приведет к перезаписи данных из ОЗУ
         акселератора в основное или видео-ОЗУ.
LD A,A - копирование блока для графического экрана подобна команде
         LD L,L, но работает с вертикальными линиями экрана.

Пример использования акселератора:

; Считаем, что экранная страница уже открыта по адресу #C000
LD HL,#C040     ; адрес начала линии первого экрана
LD DE,#C180     ; адрес начала линии второго экрана
LD BC,#140      ; длина экрана по горизонтали
DI              ; запретить прерывания для работы с акселератором
LD D,D          ; включить акселератор на установку размера блока
LD A,0          ; установить размер блока - 256 байт
LD A,A          ; установить акселератор на копирование
                ; вертикальных линий.
LDIR            ; копировать !
LD B,B          ; выключить акселератор
EI              ; включить прерывания
```

Этот отрезок программы произведет копирование всего экрана с одного экрана на другой. Время его исполения составляет примерно 1.2 инта.

Дополнительные функции акселератора (AND, OR, XOR) работают подобным же образом. Для выполнения логических функций используются команды XOR (HL); OR (HL); AND (HL).

```
Пример кодирования блока в 256 байт.

LD HL,ADRES_1
LD DE,XOR_DAT
DI
LD D,D
```

```
LD A,0     ; число байт, которые надо закодировать
LD L,L
LD A,(DE)  ; взять блок данных в ОЗУ акселератора
XOR (HL)   ; произвести операцию XOR с данными акселератора
LD (HL),A  ; запомнить в ОЗУ результат операции
LD B,B
EI
```

Скорость работы акселератора ограничивается только физической скоростью работы основного ОЗУ. Определить время работы команды с акселератором можно по такой примерной формуле:

Время работы = время работы команды без акселератора + время работы акселератора

Время работы акселератора = число пересылаемых байт /7000000 (секунд)

Отключение прерываний во время работы акселератора необходимо, так как в этот момент сильно меняется система команд процессора и программа на прерывании не сможет работать нормально. В данный момент применяется прошивка, в которой акселератор может работать в режиме со включенными прерываниями. В момент прихода прерывания он отключается и включается обратно по команде RETI. Использовать этот режим следует с осторожностью.

7 КЭШ-ОЗУ

Кэш-ОЗУ подключается двумя способами – через порт #FB стандартно по Pentagon-овски и как подобно ПЗУ через спец-порт...

7.1 Загрузка новых прошивок в ПЛМ

Описание загрузки прошивок с использованием функции BIOS...

8 ISA, порт A20

Доступ к устройствам, подключенным к ISA шине осуществляется через страницы памяти. Для получения этого доступа достаточно установить расширенную Scorpion-овскую страницу и установить в нее номер страницы ОЗУ – #D0..#DF. Бит 1 номера означает выбор доступа к порту или памяти ISA, а бит 2 определяет к какому из двух слотов осуществляется доступ.

9 Внутренние порты Z84C15

Ссылка на описание PIO, SIO и т.д. от Zilog.

Один момент. Конфликт #1F с джойстиком и ВГ93 С джойстиком все просто. Ему вывод в #1F не мешает, а ввод из #1F для работы с портом не требуется. При отключенном TR-DOS, с ВГ93 нет конфликтов, когда же он включен, то SP-DOS вместо #1F использует #0F, это избавляет от конфликта с параллельным портом. Для нормальной работы программ использующих #1F в прошивке сделано аппаратное перенаправление порта #1F в порт #0F. Следовательно команды OUT (#1F),A просто не сработают.

Что бы записать во внутренний порт процессора, надо использовать вывод через BC, т.е. LD BC,#1F : OUT (C),A. Эта команда не перенаправляется. Это причина того, почему некоторые программы могут не увидеть джойстика. Они используют IN A,(C)...

9.1 Мышь

```
; (c) Denis Parinov

CMOUSE EQU #1B
DMOUSE EQU #1A
VSIZEX EQU 256 ;X SIZE SCREEN
VSIZEY EQU 192 ;Y SIZE SCREEN

;INTERUPT
 CALL READ_M
 CALL C,MCORECT
 RET

;READING MOUSE
; HL - X COORD
; DE - Y COORD
;  A - BUTTONS
;      D0 - LEFT
;      D1 - RIGHT

MS_READ LD HL,(PIX_X)
 LD DE,(PIX_Y)
 LD A,(MB)
 RET

;INITIALIZING COM PORT
MS_INIT DI
 LD A,85
 OUT (#10),A
 LD A,45
 OUT (#10),A
 LD A,0
 OUT (CMOUSE),A
 LD A,1
 OUT (CMOUSE),A
 LD A,0
 OUT (CMOUSE),A
 LD A,3
 OUT (CMOUSE),A
 LD A,#41
 OUT (CMOUSE),A
 LD A,4
 OUT (CMOUSE),A
 LD A,#47
 OUT (CMOUSE),A
 LD A,5
 OUT (CMOUSE),A
 LD A,#E0
 OUT (CMOUSE),A
 EI
 RET

READ_M IN A,(CMOUSE)
 RRCA
```

```
 RET NC
 IN A,(DMOUSE)
 LD L,A
 BIT 6,A
 CCF
 RET Z
TST_01 IN A,(CMOUSE)
 RRCA
 JP NC,TST_01
 IN A,(DMOUSE)
 LD E,A
 BIT 6,A
 CCF
 RET NZ
TST_02 IN A,(CMOUSE)
 RRCA
 JP NC,TST_02
 IN A,(DMOUSE)
 LD D,A
 BIT 6,A
 CCF
 RET NZ
 LD A,E
 AND #3F
 LD E,A
 LD A,L
 AND #03
 RRCA
 RRCA
 OR E
 LD E,A
 LD A,D
 AND #3F
 LD D,A
 LD A,L
 AND #0C
 RRCA
 RRCA
 RRCA
 RRCA
 OR D
 LD D,A
 LD A,L
 RLCA
 RLCA
 RLCA
 RES 6,A
 JR NC,STBU
 SET 6,A
STBU RLCA
 RLCA
 AND #03
 LD (MB),A
 LD A,E
```

```
 LD (MX),A
 LD A,D
 LD (MY),A
 SCF
 RET

MCORECT LD HL,(PIX_X)
 LD DE,(MX)
 LD D,0
 BIT 7,E
 JP NZ,DECX
 ADD HL,DE
 LD (PIX_X),HL
 EX DE,HL
 LD HL,VSIZEX-1
 AND A
 SBC HL,DE
 JP NC,YCOO
 LD HL,VSIZEX-1
 LD (PIX_X),HL
 JP YCOO
DECX LD A,E
 NEG
 LD E,A
 AND A
 SBC HL,DE
 LD (PIX_X),HL
 JP NC,YCOO
 LD HL,0
 LD (PIX_X),HL
YCOO LD HL,(PIX_Y)
 LD DE,(MY)
 LD D,0
 BIT 7,E
 JP NZ,DECY
 ADD HL,DE
 LD (PIX_Y),HL
 EX DE,HL
 LD HL,VSIZEY-1
 AND A
 SBC HL,DE
 RET NC
 LD HL,VSIZEY-1
 LD (PIX_Y),HL
 RET

DECY LD A,E
 NEG
 LD E,A
 AND A
 SBC HL,DE
 LD (PIX_Y),HL
 RET NC
 LD HL,0
```

```
 LD (PIX_Y),HL
 RET

PIX_X DEFW 128
PIX_Y DEFW 96

MX DEFB #00
MY DEFB #00
MB DEFB #00
```

9.2 Принтер

Простой драйвер принтера...

9.3 Прерывания от ISA

Режим прерываний от ISA устанавливается через программирование PIO Z84C15 (управление портом B – 1Fh), описанное выше. Назначение битов самого порта следующее (внутренний порт Z84C15 1Eh, port B PIO):

```
        Bit0 - IRQ1 вход прерываний с первого слова
        Bit1 - IRQ2 вход прерываний со сторого слова
        Bit2 - DRQ2 вход
        Bit3 - DACK2 выход
        Bit4 - DRQ1 вход
        Bit5 - DACK1 выход
        Bit6,7 - используются для Printer-а
```

9.4 AT-Клавиатура

При программировании AT-клавиатуры следует помнить, что последовательный порт имеет FIFO буфер на 3 байта. После того, как принят и обработан один байт, стоит проверить нет ли еще принятых байтов и возвращаться из программы обработки клавиатуры только если этих байтов нет.

Программа инициализации клавиатуры и считывания байта.

```
;*********************************************************
;       Процедура инициализации клавиатуры
;*********************************************************
COM_A   EQU     19h
DAT_A   EQU     18h
KBD_INIT:
        LD A,0          ; установка портов режима
        OUT (COM_A),A   ; в соответствии с описанием
        LD A,1          ; последовательного порта Z84C15
        OUT (COM_A),A   ;
        LD A,0
        OUT (COM_A),A
        LD A,3
        OUT (COM_A),A
        LD A,0C1h
OUT (COM_A),A
        LD A,4
        OUT (COM_A),A
        LD A,5h
```

```
        OUT (COM_A),A
        LD A,5
        OUT (COM_A),A
        LD A,062H
        OUT (COM_A),A
        RET

;*********************************************************
;       Процедура считывания байта с клавиатуры
;*********************************************************
READ_KBD:               ; считывание с клавиатуры
        IN A,(COM_A)
        BIT 0,A         ; проверить наличие байта
        SCF             ; установить флаг C
        RET Z           ; и вернуться, если не было байта
        IN A,(DAT_A)    ; считать байт
        AND A           ; сбросить C
        RET             ; и вернуться
```

Особенности спектрумовской реализации (доп бит для F1..F10)

9.5 Таймеры

Как пример, программа вывода Wav (от Алексея Гавриленко?)

10 Контроллер FDD

Все программирование ВГ93 стандартно. Следует обратить внимание только на замену порта #1F на #0F в программе ПЗУ. В случае использования своих программ, этот порт так же должен быть #0F.

10.1 720/1.44

Программа переключения 720/1.44 для платы Sp2000. Обе должны выполняться в режиме TR-DOS и или при специально открытых портах 00BDh и 20BDh.

```
Set_1440:
        LD A,21h
        OUT (0BDh),A
        RET

Set_720:
        LD A,01h
        OUT (0BDh),A
        RET
```

Использовать напрямую не рекомендуется, лучше пользоваться функциями BIOS для работы с дискетами.

11 Контроллер HDD

Краткое описание контроллера, чтение/запись секторов через BIOS. Ссылка на документ по IDE...

12 CMOS

Порты CMOS:
#FFBD – Data Read
#BFBD – Data Write
#DFBD – Adress Write

Записывать какие либо данные в CMOS без ведома BIOS-а не рекомендуется. Исключение – установка регистров времени/будильника.

12.1 Описание регистров CMOS

PDF от Dallas 12887A...

13 Дешифрация ПЗУ/КЭШ/Контроллеры/и т.п.

Особенности обращения к ПЗУ/КЭШ/ISA...

13.1 Схема распределения портов

Sprinter имеет две обособленные группы портов. Первая группа, это внутренние порты процессора Z84C15, вторая – внешние порты. Адресация портов первой группы не может быть изменена, так как эти порты на одном кристалле с процессором. Вторая группа погключается через ППЛМ и их адреса могут изменяться как угодно, с единственным условием, непересечения с адресами первой группы.

О том какие порты имеются на кристалле Z84C15 можно прочитать в документации по этому процессору и здесь я упомяну некоторые из них. Один из последовательных портов используется для ввода данных с активной мыши. Один из параллельных используется для вывода данных, на второй параллельный порт заведены сигналы прерываний и запросов прямого доступа со слотов ISA. Параллельный порт процессора Z84C15 устроен таким образом, что на нем возможна организация прерываний по сигналам приходящим через параллельный порт. Фактически второй параллельный порт используется как контроллер прерываний.

Схема распределения портов второй группы имеет свою особенность. Главной идеей было получение возможности быcро изменять конфигурацию портов без перегрузки ППЛМ. Это достигнуто путем применения карты распределения портов, располагающейся на специальной странице ОЗУ.

При появлении цикла обращения к порту сначала происходит обращение к ОЗУ карты портов. В карте портов записано какой именно порт подключен к данному адресу. Далее происходит внутренняя дешифрация по байту из карты портов и обращение к выбранному порту. В режиме нетурбо это происходит без каких либо задержек, а в режиме турбо процессору выставляется сигнал WAIT в зависимости от необходимой длины цикла обращения к порту.

Для подключения к какому либо адресу или отключения от него какого либо порта достаточно открыть карту портов и вписать в нужное место один байт.

В странице карты портов содержится четыре карты, которые могут переключаться через системный порт. Таким образом можно осуществить быстрое переключение конфигурации портов, что может быть полезно при работе Spectrum-овских программ совместно со Sprinter-овским биосом.

Карты портов расположена в странице 40h. Переключение производится через системный порт (адрес 7Ch/3Ch). При записи в системный порт значений 04h,0Ch,14h,1Ch происходит переключение на одну из четырех карт. Начальное значение 04h.

Номер карты соответсвует адресам A12,A13 в странице (блоки по 4kb). На остальные адреса подаются следующие сигналы:

```
A0    -   A0
A1    -   A1
A2    -   A2
A3    -   A7
A4    -   A13
```

```
A5    -   A5
A6    -   A6
A7    -   A14
A8    -   A15
A9    -   /WR - сигнал записи
A10   -   /DOS - 0 - дос включен, 1 - выключен
A11   -   PN5 - сигнал блокировки порта Пентагона (может отсутствовать)
```

Таким образом порт выбирается именно по указанным сигналам, т.е. например, нельзя назначить на адреса 0050h и 0058h различные порты. Участие в дешифрации сигнала /WR позволяет назначать различные порты на один и тот же адрес на чтение и запись. Так, например, в схеме контроллера дисковода на порт 0FFh на чтение назначено чтение порта джойстика и сигналов DRQ, INTRQ контроллера дисковода, а на запись - запись в микросхему TM9 - системный порт TR-DOS.

Назначение порта производится записью соответствующего байта в карту памяти. Так, для того что бы назначить на некий адрес, скажем, на 7785h некий порт, первым делом следует удостовериться, что этот порт не пересекается ни с какими другими портами. Проще всего, это узнать, прочитав карту памяти из адреса, соответсывющего этому порту. Если в этом месте оказался нуль, значит, порт не занят и его возможно использовать (надо помнить, что не все незанятые порты стоит использовать, так как есть еще и внутренние порты Z84C15).

Как определить адрес, откуда читать байт?

Для этого надо из 7785h выделить биты 0, 1, 2, 5, 6, 7, 13, 14, 15, и установить соответсвенно им биты адреса карты памяти:

```
A0=p_A0=1,  A1=p_A1=0,  A2=p_A2=1,  A3=p_A7=1,  A4=p_A13=1,  A5=p_A5=0,
A6=p_A6=0,  A7=p_A14=1, A8=p_A15=0, A9=xxx,     A10=xxx,     A11=xxx.
```

Значение xxx имеется в виду «не определено» . Это означает, что в карте памяти по всем адресам, у которых из A[11..0] биты A[8..0] равны 010011101b=09Dh следует прописать один и тот же байт, соответсвующий номеру порта. Биты A[15..A12] адреса устанавливаются соответсвенно номеру окна и номеру карты (обычно 1100b для окна C000-FFFF и нулевой карты), таким образом, рассчитанный адрес равен C09Dh, если xxx заменить на нули, а весь набор - C09Dh, C29Dh, C49Dh, C69Dh, C89Dh, CA9Dh, CC9Dh, CE9Dh.

Если необходимо, что бы порт был «виден» только на запись, то A9 устанавливается в 0, если только на чтение - A9=1. Аналогично с битами DOS и PN5. Если нужно, что бы порт был виден только в DOS или только при открытом порте «Пентагона», необходимо устанавливать соответсувующие биты в адресе карты в конкретные значения.

С другой стороны, при необходимости неполной дешифрации (например, порт 8-мибитный), надо установить в «не определено» значения для A13,A14,A15, то есть биты 4,7 и 8 адреса карты портов и рассчитать соответствующий набор адресов.

Ввиду отсутствия дешифрации по части адресам, естественно, открытый для 7785h будет виден и по всем другим адресам, различающимся в битах A[12..8] и A[4..3].

Для наглядности напишу програму, которая откроет на чтение и запись порт 7785h некий виртуальный порт с номером 0D0h.

```
OPEN_PORT:
        LD B,0D0        ; номер виртуального порта
OPEN_P1:
        DI              ; запретить прерывания! Не дай бог нагадить в 40h
                        ; все полетит к чертям.
        IN A,(PAGE3)
        EX AF,AF'       ; сохранить адрес окна C000h
                        ; стеком не пользуюсь намеренно, бог его знает, где
                        ; он, а я страницами щелкаю
        LD A,40h
        OUT (PAGE3),A   ; установить новый адрес окна на страницу 40h
```

```
; Здесь я не буду проверять, что было записано в том месте,
; считая, что там были нули (но в реальном Спринтере стоит проверить,
; не занят ли он). Для проверки этой программы, лучше всего воспользоваться
; портом 0000h, для которого в следующей команде константа
; равна 0C000h, и он точно не зянят. Но здесь так для наглядности.

        LD HL,0C09Dh    ; адрес в карте памяти для порта 7785h на запись
; это адрес, биты которого рассчитаны немного выше. Адрес для карты 0,
; которая и устанавливается при обычной работе.

        LD (HL),B       ; установить порт на чтение в режиме DOS
        SET 1,H         ; включить A9 в 1 (сигнал /WR)
        LD (HL),B       ; установить порт на запись в режиме DOS
        SET 2,H         ; включить A10 в 1 (сигнал DOS)
        LD (HL),B       ; установить порт на запись в режиме не-DOS
        RES 1,H         ;
        LD (HL),B       ; установить порт на чтение в режиме не-DOS

; порт установлен открывать его в режиме отключенного порта пентагона я не
; стал, собственно, и не обязательно, если программа работает в 128-м
; режиме.

        EX AF,AF
        OUT (PAGE3),A   ; восстановить страницу
        EI
        RET

CLOSE_PORT:
        LD B,0          ; значение 0 - порт закрыт
        JR OPEN_P1      ; все остальное так же как при открытии

; теперь, что бы пользоваться портом 0,  достаточно сделать так

        CALL OPEN_PORT
        .....
        LD BC,0
        OUT (C),A
        .....
        LD BC,0
        IN A,(C)
        .....
        CALL CLOSE_PORT

; можно вызывать и OPEN_P1, например

        LD B,0D1h       ; открыть виртуальный порт 0D1h
        CALL OPEN_P1

; Естественно, можно вызывать открытие нового виртуального
; порта и не закрывая старый. Надо только помнить, что порт
; стоит закрыть при выходе из программы.
```

О виртуальных портах.

В Спринтере есть набор нескольких дополнительных портов, которые никак нигде не используются, но просто являются некими ячейками памяти. Это порты с номерами D0h..DFh. Они используются при работе в режиме Pentagon-512 как адреса дополнительных страниц, но в режиме Pentagon-128/Scorpion-256 могут использоваться программистом для каких нибудь особых целей, например, эмуляции какого либо порта для уже написанных программ. А в некоторых случаях могут стать передатчиком дополнительных параметров (хотя и несколько извращенным).

Байты номеров портов для карты памяти (для sp-97):

```
00              - Нет порта
01h..0Fh        - reserved
10h             - порт ВГ93 (1F)
11h             - порт ВГ93 (3F)
12h             - порт ВГ93 (5F)
13h             - порт ВГ93 (7F)
14h             - порт на запись - состояние контроллера дисковода (FF)
15h             - порт на чтение - джойстик и IRQ/INTRQ контроллера
16h..1Fh        - reserved
20h             - HDD - регистр данных
21h             - HDD - регистр состояния/ошибок
22h             - HDD - регистр количества секторов для операций R/W
23h             - HDD - регистр сектора
24h             - HDD - регистр дорожки-low
25h             - HDD - регистр дорожки-high
26h             - HDD - регистр головок/выбора мастер-слэйв
27h             - HDD - регистр команд
28h             - HDD - дополнительный регистр управления 3F6
29h             - HDD - дополнительный регистр состояния 3F7
2Ah..2Fh        - reserved
30h             - ISA-SLOT 1 - memory R/W
31h             - ISA-SLOT 2 - memory R/W
32h             - ISA-SLOT 1 - ports R/W
33h             - ISA-SLOT 2 - ports R/W
34h..3Fh        - reserved
40h             - ZX-Keyboard (порт FE)
41h..7Fh        - reserved
80h..87h        - reserved
88h             - COVOX/COVOX-Blaster
89h..8Fh        - reserved
90h             - AY-8910-port (BFFD)
91h             - AY-8910-port (FFFD)
92..BFh         - reserved
C0h             - Scorpion-256 port (1FFD)
C1h             - Pentagon-128 port (7FFD)
C2h             - Border, write only (FE)
C3h             - reserved
C4h             - port RGADR, PORT_Y (89)
C5h             - port RGMOD  (C9)
C6h..C7h        - reserved
C8h..CFh        - копии C0..C7h (not used!)
D0h..DFh        - Virtual Ports (USER ports)
E0h             - ROM page EXRANSION
E1h             - ROM page TR-DOS
E2h             - ROM page BASIC-128
E3h             - ROM page BASIC-48
E4h             - ROM page EXRANSION'
```

```
E5h             - ROM page TR-DOS'
E6h             - ROM page BASIC-128'
E7h             - ROM page BASIC-48'
E8h             - RAM page (окно 0000-3FFF)
E9h             - RAM page (окно 4000-7FFF)
EAh             - RAM page (окно 8000-BFFF)
EBh             - ROM page SYSTEM
ECh             - RAM page CASHE
EDh..EEh        - reserved
EFh             - ROM page SYSTEM'
F0h..FFh        - RAM pages (окно C000-FFFF)
```

О последних 16-ти номерах поподробнее. Установка в какой либо порт значения от F0h до FFh приведет к одному и тому же результату, что и просто установка в этот порт значения F0h. В схеме сделана переадресация номера порта так, что при значении FXh номер порта берется как F0h+SpectrumPAGE, где SpectrumPAGE - номер спектрумовской страницы, адресуемой по портам 7FFD и 1FFD. Таким образом достигается совместимость с Пентагоном и Скорпионом по распределению памяти и делается возможным установка любого номера страницы спринтеровской памяти для любой страницы Спектрума.

13.2 Конкретные адреса портов, используемые в Sprinter-е

В предыдущей секции было рассказано, как устанавливать порты. Здесь же приводятся данные по уже установленным портам в в конкретных конфигурациях.

Здесь я приведу адресацию портов для конфигураций Sprinter-1 и Sprinter-2. Сразу отмечу, что эти адреса легко могут быть изменены простой программой, в случае появления такой необходимости.

Стандартные порты.
#FE – RD_KBD – порт клавиатуры
#FE – WR_BRD – порт бордюра
#7FFD – порт расширения ZX-Spectrum 128k
#1FFD – порт расширения Scorpion ZS-256
#1F,#0F – RD_KEMPS – порт джойстика. В конфигурации Sprinter-1 порт #1F аппаратно переадресуется на порт #0F #BFFD, #FFFD – AY-PORTS – порты AY-сопроцессора (ZX-Spectrum-256/AY)

Не совсем стандартные порты.
#FB, #4F – порт COVOX-а.

Дополнительные 8-битные порты Sprinter-а.
#82 – PAGE0 – страница ОЗУ, подключаемая вместо ПЗУ через порт #1FFD
#A2 – PAGE1 – страница ОЗУ, подключенная по адресу #4000
#C2 – PAGE2 – страница ОЗУ, подключенная по адресу #8000
#E2 – PAGE2 – страница ОЗУ, подключенная по адресу #C000 Здесь надо отметить особо, через порт #E2 можно изменить любую из 16-ти страниц скорпионовского распределения памяти.
#89 – RGADR и PORT_Y – вертикальная координата точки на графическом экране или страница VIDEO-RAM для спектрумовского режима
#C9 – RGMOD – порт режима экрана. Переключает страницы режима экрана.
#3C, #7C – SYS_PORT – системный порт трогать не рекомендуется
#10..#1F,#EE,#EF,#F0,#F1,#F4 – внутренние порты Z84C15. В отличие от остальных, эти адреса изменить невозможно, так как они находятся вне ПЛМ.

Порты страниц ОЗУ открыты как на запись, так и на чтение. Это позволяет легко выполнять программы, использующие переключение страниц, а затем возвращать эти страницы назад. При работе BIOS-а все страницы сохраняются.

Дополнительные 16-тибитные порты Sprinter-а.

#xx50..#xx55 – порты HDD – использовать внешними программами не рекомендуется. Функции работы с HDD записаны в ПЗУ.

Скрытые порты Sprinter-a.

Скрытыми являются порты которые недоступны в конкретный момент времени, но могут стать доступными после проведения изменений в карте портов. Их адреса не указываются, так как они могут быть выставлены в любое место.

Порт ПЗУ BASIC48
Порт ПЗУ BASIC128
Порт ПЗУ TR-DOS
Порт ПЗУ EXPANSION
Порт ПЗУ SYSTEM

Через эти порты можно установить новые прошивки ПЗУ. Для этого их достаточно записать в ОЗУ с номерами страниц меньше #80 и записать в соответствующий порт номер этой страницы. При таком подключении страницы Эти страницы будут защищены от записи.

Частично скрытыми, так же являются и порты #7FFD, #1FFD в обычном состоянии они доступны только на запись, но значения, записываемые в эти порты можно прочитать, открыв соответствующие порты на чтение.

В других конфигурациях может отсутствовать часть портов или присутствовать новые порты.

14 Сброс.

Простой сброса может быть осуществлен записью в страницу #A0, установленную в расширенную страницу Scorpion-овского распределения портов, т.е. следующей программой:

```
SOFT_RESET:
        DI
        LD A,16
        LD BC,1FFDh
        OUT (C),A
        LD A,0A0h
        OUT (PAGE3),A
        LD (0C000h),A   ; в этот момент подается RESET
        DI              ; глюкоуловитель
        HALT
```

Более сложные варианты сброса, с перегрузкой прошивок, осуществляются через BIOS.

14.1 Старт машины

Описание старта на программном уровне (часть от Дениса)...

14.2 Доступ к HDD через память.

Доступ через память, как пример, программа Video...
В данный момент на плате sp2000 отсутствует.

Часть III
Прошивки ПЛМ

Перегузка через BIOS

15 Sprinter-1

Перечисление всего что есть. Описание возможностей.

15.1 Режимы Spectrum-128/Scorpion-256/Pentagon-512

Соответственно. Пример программного переключения.

15.2 Доступ к функциям биоса и портам.

Список портов доступных/недоступных в конктретных конфигурациях.

16 Sprinter-2

Добавления/убавления относительно Sprinter-1

16.1 Акселератор, блочные операции AND, OR, XOR

Описание акселератора с примерами.

17 Game-1

Описание прошивки, добавление/убавление относительно Sprinter-2

17.1 Акселератор + Covox-Blaster

Использование COVOX-Blaster-а совместно с акселератором.

18 Doom

Добавление относительно Game-1

18.1 Акселератор с растяжением линий

Описание акселератора с растяжениями. Пример программые растяжениями, плюс ссылка на исходник Doom-Demo.

19 Video

Особенности прошивки Video. Использование считывания с HDD прямо в экран.

19.1 Режим экрана 160x128

Дополнения к описанию режимов экрана...

Часть IV
Программирование с использованием функций БИОС-а.

20 Функции биоса

Вызов функций производится через вход в TR-DOS 3D13h. Номер команды задается в регистре C. Установленный на выходе флаг C означает завершение работы функции с ошибкой.

При работе части функций биоса необходимо что бы стек находился в области 8000h..0BFFFh, так как они используют для своей работы переключение страниц PAGE1 и PAGE3. Для устранения каких либо неприятностей связанных со стеком его следует всегда устанавливать в этот диапазон при вызове функций биоса Спринтера.

Вызов функций биос так же может быть осуществлен через вход по RST 18h при подключенном системном ПЗУ, а так же через RST 8 при подключенном ОЗУ в нулевой банке путем установки на адрес RST 8 небольшой программы, переключающейся в ПЗУ биоса

Для подключения системного ПЗУ можно воспользоваться такой последовательностью команд:

```
        DI
        LD A,0
        OUT (07Ch),A    ; после этого в 0-м адресе будет включена ПЗУ биоса
                        ; и программа может вызывать функции через RST 18h,
                        ; просто заменяя этим вызовом вызов CALL 3D13h
; * Обычные функции TR-DOS в этот момент не доступны
```

Что бы вернуться к обычному ПЗУ следует выполнить программу:

```
        LD A,0
        OUT (03Ch),A
```

Вызов из ОЗУ осуществляется через RST 8. При этом на адресе 8 должна располагаться такая программа:

```
        PUSH AF
        LD A,0
        OUT (07Ch),A    ; в этом месте вместо ОЗУ подключится ПЗУ биоса и
                        ; программа уйдет в него.
        POP AF          ; На эту команду происходит возврат при таком
                        ; вызове биоса.
        RET
; Оптимизация кода в этом месте недопустима. Вместо LD A,0 можно установить
; две команды XOR A и DI
```

Далее вызов функций осуществляется аналогично RST 18h, но следует помнить, что адресное пстранство 0000..3FFF во время работы биоса занято ПЗУ и в нем не могут располагаться данные для работы функций.

Вызов новых функций через 3D13h автоматически отключает прерывания. После исполнения функции программа должна включить их при необходимости. Если необходимо что бы прерывания были включены все время, следует использовать режим IM 2, с таблицей, стеком и обработчиком расположенным в области 8000h..BFFFh и пользоваться вызовом через RST 18h или RST 8 В этом случае прерывания в биосе не отключаются.

20.1 Работа с памятью

```
EMM_FN0:                ; определение объемов ОЗУ
        LD C,0C0h       ; функция номер 0C0h
        CALL 3D13h      ; HL - общий объем памяти в страницах по 16kb
                        ; BC - объем свободной памяти в страницах по 16kb

EMM_FN1:                ; инициализация распределения памяти
                        ; стирается вся информация о выделенных ранее блоках
                        ; ОЗУ. Устанавливаются как занятые блоки с
                        ; системной информацией, а так же первые 256kb ОЗУ
        LD C,0C1h       ; функция номер 0C1h
        CALL 3D13h      ; выходных параметров нет

EMM_FN2:                ; выделение блока ОЗУ
        LD B,num_pages  ; запрашиваемое число страниц ОЗУ
        LD C,0C2h       ; номер функции
        CALL 3D13h      ; NC -> A - идентификатор блока
                        ; CF -> A=1 - нет памяти

EMM_FN3:                ; освободить блок ОЗУ
        LD A,id_blk     ; идентификатор блока
        LD C,0C3h       ; номер функции
        CALL 3D13h      ; NC - нормальное завершение
                        ; CF - неверный идентификатор блока
                        ; правильность идентификатора отслеживается не всегда

EMM_FN4:                ; получить физический номер страницы из блока
        LD A,id_blk     ; идентификатор блока
        LD B,page       ; логическая страница в блоке
        LD C,0C4h       ; номер функции
        CALL 3D13h      ; NC -> A - физический номер страницы
                        ; CF -> A=0 - нет такого блока, A=FF - конец блока

EMM_FN5:                ; получить список физических страниц блока
        LD A,id_blk     ; идентификатор блока
        LD HL,bufer     ; буфер длиной 256 байт для размещения списка
                        ; буфер должен быть длиной на единицу больше числа
                        ; страниц в блоке
        LD C,0C5h       ; номер функции
        CALL 3D13h      ; NC -> HL - тот же буфер, B - число страниц в блоке
                        ; данные по адресу HL - список физических страниц по
                        ; порядку. Список заканчивается байтом FF
                        ; CF -> неверный идентификатор блока. Старая
                        ; информация в буфере может быть затерта

EMM_FN6:                ; Получение адресов портов окон
        LD A,win_num    ; номер окна проецирования 0,1,2 или 3
        LD C,0C6h       ; номер функции
        CALL 3D13h      ; NC -> C - 8-мибитный адрес порта, B - номер
                        ; подключенной в данный момент страницы ОЗУ
                        ; CF -> ошибка номера окна
; Функция фактически не используется в данный момент. Адреса портов окон
; не изменялись с самого начала разработки компьютера и, надеюсь, не будут
; меняться. Для соблюдения приличий программисту следует хотя бы один раз
```

```
; вызвать эти функции и сравнить адреса портов с теми, что используются в
; программе и, если они не совпадают, выдать соответствующее предупреждение.
; В данный момент эти порты таковы: PAGE0=82h, PAGE1=0A2h, PAGE2=0C2h,
; PAGE3=0E2h

EMM_FN7:                ; получить следующую страницу блока по предыдущему
        LD A,page       ; физическая страница блока
        LD C,0C7h       ; номер функции
        CALL 3D13h      ; NC -> A - следующая физическая страница блока
                        ;       A=FF - индицирует конец блока
                        ; CF -> ошибка номера страницы
; Информация о распределении памяти хранится в виде RAM Allocation Table,
; похожей на дисковый FAT. Поэтому нахождение физического номера следующей
; страницы по предыдущему физическому номеру происходит значительно быстрее,
; чем поиск по увеличенному на единицу логическому номеру.

EMM_FN8:                ; слияние блоков
        LD A,id_blk1    ; блок номер 1
        LD B,id_blk2    ; блок номер 2
        LD C,9Eh        ; номер функции
        CALL 3D13h      ; NC -> A - блок результата
                        ; CF -> ошибка, неверный номер блока

EMM_FN9:                ; разделение блока
        LD A,id_blk1    ; блок
        LD B,len_blk    ; новая длина блока
        LD C,9Dh        ; номер функции
        CALL 3D13h      ; NC -> A - блок результата, B - блок остатка
                        ; CF -> ошибка, неверный номер блока
```

20.2 Работа с блоками как с RAM-Disk-ами

```
BLK_RD_WR:              ; чтение/запись из/в блок(а) памяти секторами
                        ; по 256 байт
        LD HL,bufer     ; адрес буфера данных
        LD DE,sector    ; абсолютный номер сектора
        LD B,sec_num    ; число секторов
        EX AF,AF'
        LD A,command    ; команда 0 - чтение, FF - запись
        EX AF,AF'
        LD A,id_blk     ; идентификатор блока
        LD C,0C8h       ; номер функции
        CALL 3D13h      ; NC -> нормальное завершение
                        ; CF -> ошибка идентификатора
; * функция на бета-тестировании...

BLK_TO_RAMD:            ; назначить блок памяти RAM-Disk-у
                        ; любой блок памяти может содержать данные
                        ; RAM-Disk-а в формате TR-DOS для подключения этих
                        ; данных в качестве диска и служит эта функция
        LD A,ram_disk   ; номер RAM-Disk-а 0..15 - соответствует
                        ; RAM-Disk-ам от e: до t:
        LD B,id_blk     ; идентификатор блока
```

```
        LD C,0C9h       ; номер функции
        CALL 3D13h      ; NC -> нормальное завершение
                        ; CF -> ошибка: неверный номер RAM-Disk-а или
                        ; RAM-Disk занят

RAMD_CLEAR:             ; освободить RAM-Disk
                        ; освобождение RAM-Disk-а не есть освобождение
                        ; блока ОЗУ. Это просто отключение блока ОЗУ от
                        ; RAM-Disk-а
        LD A,ram_disk   ; номер RAM-Disk-а - 0..15
        LD C,0CAh       ; номер функции
        CALL 3D13h      ; NC -> нормальное завершение, B - идентификатор
                        ; блока отключенного от RAM-Disk-а
                        ; CF -> ошибка: неверный номер RAM-Disk-а или
                        ; RAM-Disk был свободен

GET_RAMD_ST:            ; получение идентификатора блока, назначенного на
                        ; RAM-Disk
        LD A,ram_disk   ; номер RAM-Disk-а 0..15
        LD C,0CEh       ; номер функции
        CALL 3D13h      ; NC -> A - идентификатор блока.
                        ;       A=0 - блок не назначен.
                        ; CF -> ошибка номера RAM-Disk-а
```

20.3 Управление назначением на дисководы

Каждый из 4-х дисководов TR-DOS может быть переназначен для работы с RAM-Disk-ами, винчестером и реальными дисководами.

```
RAMD_TO_DRV:            ; назначение RAM-Disk на дисковод.
        LD A,ram_disk   ; номер RAM-Disk-а
        LD B,drive      ; номер дисковода 0..3 - соответствует дисководам
                        ; A:, B:, C:, D:
        LD C,0CBh       ; номер функции
        CALL 3D13h      ; NC -> нормальние завершение
                        ; CF -> ошибка: неверный номер драйва или рамдиска

FDD_TO_DRV:             ; назначение реального дисковода
        LD A,disk_drive ; номер физического дисковода 0..3
        LD B,drive      ; номер драйва 0..3
                        ; Номер физического дисковода и номер драйва должны
                        ; совпадать, так как компьютер не имеет
                        ; электрической схемы переключения дисководов на
                        ; разные буквы. В будущих версиях железа, возможно,
                        ; это появится.
        LD C,0CCh       ; номер функции
        CALL 3D13h      ; NC -> нормальние завершение
                        ; CF -> ошибка: неверный номер драйва или дисковода

HDD_TO_DRV:             ; назначение винчестера на дисковод
        LD A,hdd_drive  ; Номер винчестера. Должен быть 0. В дальнейшем
                        ; будет иметь значения от 0 до 15 для подключения
                        ; различных разделов и master/slave
        LD B,drive      ; номер драйва 0..3
        LD C,0CDh       ; номер функции
```

```
        CALL 3D13h      ; NC -> нормальние завершение
                        ; CF -> ошибка: неверный номер драйва или винчестера

GET_DRV_ST:             ; получить тип назначения на драйв
        LD A,drive      ; номер драйва 0..3
        LD C,0CFh       ; номер функции
        CALL 3D13h      ; NC -> нормальние завершение. A - тип назначения
                        ; A=0..3 - назначен реальный дисковод A:, B:, C:, D:
                        ; A=4..19 - назначен RAM-Disk - A = ram_disk + 4
                        ; A=40h..4Fh - назначен винчестер (40h+hdd_drive)
                        ; CF -> ошибка номера драйва
```

20.4 Функции управления железом и определение версии

```
FN_VERSION:             ; выдача информации о версии биоса и железа
        LD HL,bufer     ; буфер, куда будет помещена ASCIIZ строка с
                        ; названием и номером версии, конец строки отмечен
                        ; нулем.
        LD C,0EFh       ; номер функции
        CALL 3D13h      ; NC -> HL - тот же буфер с записанной строкой.
                        ;       DE - версия биоса
                        ;       BC - версия железа
                        ;         BC=FFFF - not identifyed
                        ;         BC=FFFE - Sprinter-1
                        ;         BC=FFFD - Sprinter-2
                        ;         BC=FFFC - Spectrum + AY8910
                        ;         BC=FFFB - Game-1
                        ;         BC=FFFA - Video-1
                        ;         BC=FFF9 - DooM
                        ;         Иные значения BC - новые прошивки
                        ;       версия железа выдается только
                        ;       в биосах версий 1.16 и выше
                        ; CF -> ошибка. Очень старая версия, не имеющая
                        ; данной функции

SPRINTER_1:             ; переключение в конфигурацию Sprinter-1
        LD C,0F0h       ; номер функции
        CALL 3D13h      ; NC -> нормальное завершение
                        ; CF -> функция не исполнена, фатальная ошибка
                        ; машину следует перезапустить по RESET

SPRINTER_2:             ; переключение в конфигурацию Sprinter-2
        LD C,0F1h       ; номер функции
        CALL 3D13h      ; NC -> нормальное завершение
                        ; CF -> функция не исполнена, фатальная ошибка
                        ; машину следует перезапустить по RESET

SPRINTER_ALL:           ; переключение конфигурации пользователя
        LD A,page_cnf   ; страница с файлом прошивки для ПЛМ EPF10K10
                        ; страница не может иметь номер больше 127
                        ; Файл прошивки, естественно должен быть уже
                        ; загружен в эту страницу
        LD C,0F3h       ; номер функции
        CALL 3D13h      ; NC -> нормальние завершение
```

```
                        ; CF -> функция не исполнена, фатальная ошибка
                        ; машину следует перезапустить по RESET

; Перед исполнением функций SPRINTER_1, SPRINTER_2 и SPRINTER_ALL биос
; проверяет загруженную прошивку по идентификатору. Если требуемая
; прошивка совпадает с текущей, то операция перезагрузки ПЛМ не
; производится.
; Во время исполнения этих функций наблюдается сбой синхронизации монитора,
; так как в этот момент происходит изменение всей схемы компьютера и
; сигналы синхронизации монитора просто отсутствуют.

FN_SINC:                ; установка синхронизации, очистка режима экрана
                        ; функция может быть отнесена и к группе функций
                        ; вывода на экран, так как полностью очищает
                        ; страницы режима экрана. На всем экране остается
                        ; только бордер
        LD A,sinc_mode  ; режим синхронизации
                        ; A=0 режим по умолчанию - используется для очистки
                        ; страниц режима (отключения вывода всех окон)
                        ; A=1 режим Scorpion - 312 строк в экране,
                        ; положение INT-a, как в Scorpion-256
                        ; A=2 режим Pentagon - 320 строк в экране,
                        ; положение INT-a как в Pentagon-128
        LD C,0F2h       ; номер функции
        CALL 3D13h      ; NC -> нормальное завершение
                        ; CF -> неверный номер режима синхронизации
; изменение режима синхронизации может привести к временному сбою
; синхронизации монитора.

; DCP_FN0:              ; функция управления дешифратором портов.
;        LD C,0F4h      ; В данный момент не доделана
; Функция позволит открывать/закрывать дополнительные порты компьютера

SET_PORTS:              ; глобальная установка портов
        LD A,port_num   ; внутренний номер порта
                        ; F0..FF - страницы Scorpion 0..15, подключаемые в
                        ; адрес 0C000h, страница именно та, которая
                        ; подключена в данный момент через 7FFD,1FFD
                        ; E0=EXPANSION, E1=TR-DOS, E2=BASIC-128, E3=BASIC-48
                        ; E4=EXPANSION',E5=TR-DOS',E6=BASIC-128',E7=BASIC-48'
                        ; E8=RAM0, E9=RAM1, EA=RAM2, EB=SYS0, EC=CASH
                        ; ED,EE - reserv, EF=SYS1
                        ; C0=COPY_1FFD, C1=COPY_7FFD, C2=COPY_BRD, C3-reserv
                        ; C4-reserv, C5=COPY_V_MODE, C6=COPY_SYS, C7-reserv
                        ; C8..CF - альтернативный набор для C0..C7
                        ; D0..DF-reserv - доп. страницы для Pentagon-512
                        ; 80..BF-user_ports!
                        ; 00..7F-внешние порты, использовать не рекомендуется
        LD B,port_data  ; данные, записываемые в страницу
        LD C,0F8h       ; номер функции
        CALL 3D13h      ; B - предыдущее содержание порта

READ_PORTS:             ; глобальное чтение портов
        LD A,port_num   ; внутренний номер порта
```

```
        LD C,0F9h       ; номер функции
        CALL 3D13h      ; B - содержание порта

WRITE_PORTS:            ; глобальная запись портов
        LD A,port_num   ; внутренний номер порта
        LD B,data_port  ; записываемые данные
        LD C,0FAh       ; номер функции
        CALL 3D13h      ;

; Функции SET_PORTS, READ_PORTS и WRITE_PORTS позволяют иметь доступ
; к любым портам компьютера независимо от того, открыты они или нет.
; В данный момент функции недоступны и находятся в разработке
; С помощью этих функций будет возможно прочитать содержимое портов 1FFD и
; 7FFD, например, а так же установить нужные значения в закрытые системные
; порты. Порты User-а позволят эмулировать некоторые устройства,
; отсутствующие в Спринтере, а так же могут дать особый способ
; передачи данных между программами, минуя ОЗУ.

CMOS_RD:                ; читать из регистра CMOS
        LD C,0F6h       ; номер функции
        LD D,cmos_reg   ; номер регистра CMOS
        CALL 3D13h      ; NC - часы есть
                        ; CF - часов нет

CMOS_WR:                ; писать в регистр CMOS
        LD C,0F7h       ; номер функции
        LD D,cmos_reg   ; номер регистра CMOS
        CALL 3D13h      ; NC - часы есть
                        ; CF - часов нет
Функции CMOS_RD,CMOS_WR работают всегда. Если в машине нет микросхемы CMOS,
она эмулируется. Наличие микросхемы определяется функцие CMOS_TEST.

CMOS_TEST:              ; проверить наличие CMOS
        LD C,0F5h       ; номер функции
        CALL 3D13h      ; NC - часы есть
                        ; CF - часов нет


FN_TURBO:               ; функция управления турбо режимом.
        LD A,turbo_mode ; режим турбо: 2 - off, 3 - on
        LD C,08Fh       ; номер функции
        CALL 3D13h      ; NC -> исполнение
                        ; CF -> неверный режим турбо
; * переключение режима турбо может не произойти, если прошивка не
; поддерживает это переключение. При этом ошибки не происходит.
```

20.5 Функции печати и управления режимом экрана

```
WIN_OPEN:               ; функция открытия окна.
        LD IX,win_descriptor    ; описатель окна
;               IX - 32-хбайтовый описатель окна
;               (IX+0) - горизонтальный размер окна в знакоместах
;               (IX+1) - вертикальный размер в знакоместах
;               (IX+2) - положение окна по горизонтали на экране
```

```
;               (IX+3) - положение окна по вертикали на экране
;               (IX+4) - режим знакоместа
;                        bit4=1 - text_mode  bit4=0 - graf_mode
;                        bit5=0 - 16, bit5=1 - 8 точек в знакоместе
;                        graf_mode bit3..0 - не существенны
;                                  bit7..6 - номер палитры
;                        text_mode bit7..6,3..0 - номер знакогенератора
;                                  исключение: bit7..6=B"11" -> бордер
;               (IX+5) - дополнительный режим знакоместа
;                        bit0=1 - указывает на включение спектрумовской
;                                 адресации экрана
;               (IX+6) - положение по X в поле графики (по знакоместам)
;               (IX+7) - положение по Y в поле графики (по знакоместам)
;                        разъяснения о положении в поле графики - ниже
;               (IX+8..31) - зарезервировано (переменные окна)
        LD E,win_flag   ; флаги окна
                        ; бит 0 указывает какую страницу режима включать
                        ; после исполнения функции
                        ; бит 4 указывает на какой странице режима
                        ; открывать окно
        LD HL,win_place ; HL - место на экране по знакоместам
                        ; (копия в IX+2,3), в новых версиях биоса значение
                        ; HL не существенно
        LD C,0B0h       ; номер функции
        CALL 3D13h      ; NC -> A - номер окна
                        ; CF -> ошибка слишком много окон
        LD (id_win),A   ; сохранить идентификатор окна

; * При открытии окна описатель копируется в системную страницу ОЗУ и
; программа может не сохранять его.
; ** В данный момент идентификатор окна всегда равен 0
```

Видео-ОЗУ Спринтера можно представить как одно сплошное поле графики размером 1024 точки по горизонтали на 256 точек по вертикали Положение в поле графики показывает где будет находиться в этом поле верхний левый угол окна. Положение исчисляется в знакоместах. Т.е. Если указано положение по X – 2, по Y – 6, это означает, что верхний угол окна будет расположен по координатам X=16, Y=48 в поле графики видео-ОЗУ Таким образом, если, например, открыть два окна в разных местах, но с одинаковыми координатами в поле графики, на экране окажутся два идентичных окна, данные в которые будут попадать одновременно.

Знакогенераторы текстовых режимов так же располагаются в видео-ОЗУ и имеют конкретные адреса в поле графики. При необходимости иметь на экране как графическое, так и текстовое изображение надо следить, что бы данные графических окон не попадали в поле графики, где расположены знакогенераторы

При использовании какого либо знакогенератора, он занимает часть поля графики по координатам (координаты в знакоместах, т.е. в значениях байта IX+6 описателя окна)

$$X = (8 * (bit3..0 \text{ of mode}))..(8 * (bit3..0 \text{ of mode}) + 7)$$

По Y занимаются все положения. Таким образом, при использовании нескольких знакогенераторов сначала следует использовать знакогенераторы с номерами меняющимися в Bit7..6, так как они попадают в одни и те же координаты поля графики

При открытии графических окон следует помнить, что в этот момент информация текстового экрана находящаяся в этом месте будет утеряна. При открытии текстового окна изменяется информация только в поле графики знакогенератора соответствующему этому текствовому экрану. Если эта информация и информация графического окна не пересекались, то при повторном открытии графического экрана, на нем автоматически восстановится графическая картинка

```
WIN_CLOSE:              ; закрытие окна
        LD A,(id_win)   ; идентификатор окна (пока должен быть 0)
        LD C,0B1h       ; номер функции
        CALL 3D13h      ; NC -> успешное завершение
                        ; CF -> ошибка - неверный идентификатор
                        ; Окно с номером 0 никогда не закрывается и попытка
                        ; закрытия приводит к ошибке


LP_OPEN_S:              ; Открытие стандартных окон.
        LD E,win_flag   ; флаги окна
                        ; bit 0 определяет страницу режима, которая будет
                        ; открыта после исполнения функции
        LD C,080h       ; номер функции
        LD B,win_type   ; тип открываемого окна
;               0 - спектрумовское окно 32x24
;               1 - текстовое окно 64x24
;               3 - текстовое окно 80x32
;               4 - спектрумовское окно, HL - положение окна
;               5 - текстовое окно 64x24, HL - положение окна
;               7 - текстовое окно 80x32, HL - положение окна
;               8 - графическое окно 0, HL - положение окна
;               9 - графическое окно 1, HL - положение окна
        LD HL,win_place ; положение окна для 4..9 типов
        CALL 3D13h      ; выполнить функцию
; ** Функция старая, использовать не рекомендуется.

; Далее, в функциях запоминания, восстановления, перемещения и стирания
; подразумеваются локальные окна в смысле "окно в окне". Идентификатор окна
; относится к глобальному окну, отнисительно которого адресуются локальные


WIN_COPY_WIN:           ; копирование данных текстового окна в память
                        ; запоминание окна
        LD A,(id_win)   ; идентификатор глобального окна (пока должен быть 0)
        LD H,ver_size   ; HL - размер локального окна вертикаль/горизонталь
        LD L,hor_size   ; размер в символах
        LD D,ver_place  ; DE - положение локального окна в глобальном окне
        LD E,hor_place  ; положение по горизонтали в символах
        LD IX,bufer     ; адрес буфера для запоминания данных локального окна
        EX AF,AF'
        LD A,bufer_page ; страница буфера для данных окна
        EX AF,AF'       ; адрес буфера указывается для окна 0C000h
                        ; если адрес указан с 8000h, номер страницы буфера
                        ; не действителен
        LD C,0B2h       ; номер функции
        CALL 3D13h      ; NC -> нормальное завершение
                        ; CF -> ошибка - неверный идентификатор окна
; при работе этой функции через RST 18h или RST 8, обязательна установка
; DI, так как функция пользуется стеком для ускорения своей работы.

WIN_RESTORE_WIN:        ; копирование данных из памяти в текстовое окно
                        ; восстановление окна
        LD A,(id_win)   ; идентификатор глобального окна (пока должен быть 0)
```

```
        LD H,ver_size   ; HL - размер локального окна вертикаль/горизонталь
        LD L,hor_size   ; размер в символах
        LD D,ver_place  ; DE - положение локального окна
        LD E,hor_place  ; положение по горизонтали в символах
        LD IX,bufer     ; адрес буфера данных для локального окна
        EX AF,AF'
        LD A,bufer_page ; страница буфера данных окна
        EX AF,AF'       ; адрес буфера указывается для окна 0C000h
                        ; если адрес указан с 8000h, номер страницы буфера
                        ; не действителен
        LD C,0B3h       ; номер функции
        CALL 3D13h      ; NC -> нормальное завершение
                        ; CF -> ошибка - неверный идентификатор окна
; при работе этой функции через RST 18h или RST 8, обязательна установка
; DI, так как функция пользуется стеком для ускорения работы.

; Данные для функций WIN_COPY_WIN и WIN_RESTORE_WIN имеют одинаковую
; структуру В данный момент эта структура похожа на структуру текстового
; экрана IBM, т.е. данные идут в формате  sym1,atr1,sym2,atr2,.. сплошным
; массивом. Сначала данные для первой строки, затем сразу для второй и т.д.

WIN_GET_SYM:            ; взять символ с экрана
        LD A,(id_win)   ; идентификатор окна (пока должен быть 0)
        LD DE,place     ; положение символа: D - вертикаль, E - горизонталь
        LD C,0B4h       ; номер функции
        CALL 3D13h      ; NC -> нормальное завершение
                        ; L - символ, H - атрибут, B - знакогенератор
                        ; CF -> ошибка неверный идентификатор окна


WIN_PUT_SYM:            ; положить символ на экран
        LD A,(id_win)   ; идентификатор окна (пока должен быть 0)
        LD DE,place     ; положение символа: D - вертикаль, E - горизонталь
        LD B,sym_zg     ; знакогенератор
        LD L,symbol     ; символ
        LD H,atribute   ; атрибут символа
        LD C,0B5h       ; номер функции
        CALL 3D13h      ; NC -> нормальное завершение
                        ; CF -> ошибка неверный идентификатор окна

WIN_SET_ZG:             ; установка знакогенератора
        LD A,sym_zg     ; системный номер знакогенератора
        LD DE,zg_form   ; указатель на 2Kb данных знакогенератора
; Данные знакогенератора должны располагаться в таком виде, в каком они
; выглядели бы как набор символов на спектрумовском экране при переносе 2Kb
; LDIR-ом в адрес 4000h
; * В будущем возможно изменение этого расположения на обычное
        LD C,0B6h       ; номер функции
        CALL 3D13h      ; NC -> завершение
                        ; CF -> ошибка (старая версия, нет функции)

WIN_MOVE_WIN:           ; перемещение окна
        LD A,(id_win)   ; идентификатор глобального окна (пока должен быть 0)
        LD H,ver_size   ; HL - размер локального окна вертикаль/горизонталь
```

```
        LD L,hor_size   ; размер в символах
        LD D,ver_place  ; DE - положение локального окна
        LD E,hor_place  ; положение по горизонтали в символах
        LD IX,new_place ; новое положение локального окна
        LD C,0B2h       ; номер функции
        CALL 3D13h      ; NC -> нормальное завершение
                        ; CF -> ошибка - неверный идентификатор окна
; при работе этой функции через RST 18h или RST 8, обязательна установка
; DI, так как функция пользуется стеком для ускорения работы.

; Даллее следуют функции печати для работы с _текущим_ глобальным окном.
; В данный момент текущим всегда является последнее открытое окно
; На графическом экране функция не работает

LP_PRINT_ALL:           ; печать символов с атрибутом
        LD A,symbol     ; символ
        LD E,atribute   ; атрибут
        LD B,num_sym    ; число выводимых символов
        LD C,081h       ; номер функции
        CALL 3D13h      ; на экран выводится строка из B одинаковых
                        ; символов
                        ; регистры HL,IX - сохраняются

LP_PRINT_SYM:           ; Вывод символов на экран с текущего
                        ; знакоместа без атрибута
        LD A,symbol     ; символ
        LD B,num_sym    ; число выводимых символов
        LD C,082h       ; номер функции
        CALL 3D13h      ; на экран выводится строка из B одинаковых символов
                        ; атрибут остается тот, который был на экране
                        ; регистры HL,IX - сохраняются

LP_PRINT_ATR:           ; печать атрибутов
        LD E,atribute   ; атрибут
        LD B,num_sym    ; число выводимых символов
        LD C,083h       ; номер функции
        CALL 3D13h      ; на экран выводится строка из B одинаковых
                        ; атрибутов. Символы не меняются.
                        ; регистры HL,IX - сохраняются

LP_SET_PLACE:           ; Установка текущего знакоместа в окне
        LD E,hor_place  ; номер символа по горизонтали
        LD D,ver_place  ; номер символа по вертикали
                        ; ** Превышение границ приводит не к ошибке, а к
                        ; переустановке сначала, за вычетом полного
                        ; размера окна
        LD C,084h       ; номер функции
        CALL 3D13h      ; позиция печати устанавливается в соответствии с
                        ; регистром DE

LP_PRINT_LN:            ; Вывод строки символов на экран с текущего
                        ; знакоместа
        LD HL,line_adr  ; адрес строки. Должен быть между 04000h и 0BFFFh
        LD E,atribute   ; атрибут, с которым будет выведена строка
```

```
        LD B,num_sym    ; длина выводимой строки
        LD C,085h       ; номер функции
        CALL 3D13h      ; "исполнение желаний"


LP_PRINT_LN2:           ; Вывод строки символов на экран с текущего
                        ; знакоместа без атрибутов
        LD HL,line_adr  ; адрес строки. Должен быть между 04000h и 0BFFFh
        LD B,num_sym    ; длина выводимой строки
        LD C,086h       ; номер функции
        CALL 3D13h      ; строка будет выведена без изменения атрибутов в
                        ; месте печати


LP_PRINT_LN3:           ; Вывод строки символов на экран с текущего
                        ; знакоместа до разделителя. После разделителя
                        ; выводятся пробелы что бы вывести B символов
        LD HL,line_adr  ; адрес строки. Должен быть между 04000h и 0BFFFh
        LD E,atribute   ; атрибут, с которым будет выведена строка
        LD D,delimiter  ; разделитель
        LD B,num_sym    ; длина выводимой строки
        LD C,087h       ; номер функции
        CALL 3D13h      ; символы из (HL) выводятся на экран, пока не
                        ; встретится символ равный D, далее печатаются
                        ; пробелы, как дополнение строки до B символов


LP_PRINT_LN4:           ; Вывод строки символов длиной B на экран с текущего
                        ; знакоместа до разделителя D, без атрибутов.
        LD HL,line_adr  ; адрес строки. Должен быть между 04000h и 0BFFFh
        LD D,delimiter  ; разделитель
        LD B,num_sym    ; длина выводимой строки
        LD C,088h       ; номер функции
        CALL 3D13h      ; символы из (HL) выводятся на экран, пока не
                        ; встретится символ равный D, далее печатаются
                        ; пробелы, как дополнение строки до B символов
                        ; атрибуты не изменяются

LP_CLS_WIN:             ; очистка экрана
        LD DE,place     ; положение локального окна (глобальное = текущее)
        LD H,ver_size   ; HL - размер локального окна вертикаль/горизонталь
        LD L,hor_size   ; размер в символах
        LD B,atribute   ; атрибут очистки
        LC C,089h       ; номер фунции
        CALL 3D13h      ; выполнение. Произворится выводом пробелов с
                        ; заданным атрибутом

LP_SCROLL_UD:           ; Скроллинг части глобального окна вверх/вниз
        LD B,scroll_type; тип скроллинга 1 - вверх/ 2 - вниз
        LD D,beg_line   ; начальная строка скроллинга
        LD E,num_lines  ; число скроллируемых строк
        LD C,08Ah       ; номер функции
        CALL 3D13h      ; выполнение. Скроллируются полные строки
                        ; глобального окна
```

```
LP_PRINT_LN5:           ; Вывод строки символов на экран с текущего
                        ; знакоместа до разделителя после разделителя
                        ; вывод останавливается
        LD HL,line_adr  ; адрес строки. Должен быть между 04000h и 0BFFFh
        LD E,atribute   ; атрибут, с которым будет выведена строка
        LD B,num_sym    ; максимальная длина выводимой строки
        LD C,08Bh       ; номер функции
        CALL 3D13h      ; символы из (HL) выводятся на экран, пока не
                        ; встретится символ равный D или количество
                        ; символов не превысило B. Далее происходит
                        ; возврат


LP_PRINT_LN6:           ; Вывод строки символов на экран с текущего
                        ; знакоместа до разделителя после разделителя
                        ; вывод останавливается, без атрибутов
        LD HL,line_adr  ; адрес строки. Должен быть между 04000h и 0BFFFh
        LD B,num_sym    ; максимальная длина выводимой строки
        LD C,08Ch       ; номер функции
        CALL 3D13h      ; символы из (HL) выводятся на экран, пока не
                        ; встретится символ равный D или количество
                        ; символов не превысило B. Далее происходит
                        ; возврат. Атрибуты не выводятся

LP_CLS_WIN2:            ; очистка экрана, указанием символа заполнения
        LD DE,place     ; положение локального окна (глобальное = текущее)
        LD H,ver_size   ; HL - размер окна вертикаль/горизонталь
        LD L,hor_size   ; размер в символах
        LD A,symbol     ; символ очистки
        LD B,atribute   ; атрибут очистки
        LC C,08Dh       ; номер фунции
        CALL 3D13h      ; Выполнение. Произворится выводом пробелов с
                        ; заданным атрибутом и символом

LP_GET_PLACE:           ; получить текущее положение вывода на экран
                        ; в глобальном окне
        LD C,08Eh       ; номер функции
        CALL 3D13h      ; в регистр DE будут положены координаты,
                        ; в которых будет напечатан следующий символ
                        ; D - вертикаль, E - горизонталь
```

20.6 Графические функции

```
;       Координаты считаются от верхнего левого угла экрана

PIC_POINT:              ; установить точку
        LD DE,Y_coord   ; координата по вертикали
        LD HL,X_coord   ; координата по горизонтали
        LD A,(id_win)   ; идентификатор граф. окна (пока должен быть 0)
        LD B,color      ; цвет точки
        LD C,0A1h       ; номер функции
        CALL 3D13h      ; поставить точку
```

```
; В действительности ставить точки на экране с помощью функции биоса,
; слишком медленно. Для этого лучше пользоваться прямым выводом данных
; на графический экран. Устройство экрана и способы прямого вывода
; графических данных описаны в файле архитектуры Спринтера.


PIC_SET_PAL:            ; установка палитры
        LD HL,pal_data  ; данные палитры:
                        ; список цветов по четыре байта B,G,R,Y
        LD E,beg_color  ; начальный цвет
        LD D,num_colors ; количество устанавливаемых цветов
        LD B,pal_mask   ; маска при установке палитры. Для нормального
                        ; режима должна быть FF
        LD A,page_pal   ; номер палитры 0..15 значения от 8 до 15 резервные
        LD C,0A4h       ; номер функции
        CALL 3D13h      ; установка палитры

; данные палитры должны представлять собой список приблизительно такого вида:
        DB blue1,green1,red1,0
        DB blue2,green2,red2,0
        ......................
        DB blueN,greenN,redN,0
; N = num_colors. Значение num_colors равное 0 соответствует 256-ти цветам
; при записи в видео-ОЗУ все данные предварительно проходят функцию AND со
; значением pal_mask
```

Страницы палитры 0..3 соответствуют графическим режимам. Для вывода в соответствующей палитре нужно задать соответствующее значение bit7..6 в байте режима знакоместа

Страницы 4..7 соответствуют текстовому режиму и режиму «Spectrum» В странице 4 задается цвет paper для каждого атрибута. В странице 5 задается цвет ink для каждого атрибута. В странице 6 задается цвет paper, которым он будет моргать в режиме flash В странице 7 задается цвет ink, которым он будет моргать в режиме flash Таким образом, для каждого из 256-ти атрибутов задается четыре цвета если цвета 4,5 совпадают с цветами 6,7 то режим flash оказывается отключенным. Для его включения в спектрумовском режиме надо поменять местами цвета 6 и 7. Если надо включить flash в режим IBM-CGA, следует установить цвета 6 и 7 одинаковыми и равными цвету 4 по сути режим flash всегда включен и на экране постоянно меняются цвета paper с 4-го на 6-й, а цвета ink с 5 на 7-й. Если эти пары цветов для атрибута знакоместа устанавливаются одинаковыми, то flash в этом месте не виден. Используя подобное задание цветов текстового режима можно легко добиться совместимости по цветам как со Спектрумом, так и с IBM

```
SET_PAL_INIT: ; установка внутренней палитры.
LD A,PAL_PAGE ; страница палитры
LD B,PAL_N ; номер палитры
; 2 - установка спектрумовской палитры
; 1 - установка графической плаитры
LD C,0A6h ; номер функции
        CALL 3D13h      ; установка палитры
```

20.7 Работа с винчестером и дисками MS-DOS

```
HDD_INIT:               ; инициализация винчестера
        LD C,040h       ; номер функции
        CALL 3D13h      ; NC -> нормальное завершение
                        ; CF -> винчестер не найден
```

```
HDD_RECAL:              ; рекалибровка винчестера
        LD C,041h       ; номер функции
; * Функция зарезервирована для дальнейшего использования

HDD_TEST_IDE:           ; Тест наличия интерфейса IDE
        LD C,042h       ; номер функции
        CALL 3D13h      ; NC -> нормальние завершение
                        ;   в регистре B информация о наличии устройств
                        ;   bit0=1 - есть устройство "master"
                        ;   bit1=1 - есть устройство "slave"
                        ; CF -> ошибка, аппаратная неисправность

HDD_PREPARE:            ; подготовка винчестера к операции чтения/записи
        LD C,043h       ; номер функции
        LD HL,bufer_adr ; адрес буфера данных
        LD A,bufer_page ; страница буфера, если адрес в окне 0C000h
        LD B,sec_num    ; число секторов
        LD DE,sec_low   ; абсолютный номер сектора младшая часть
        LD IX,sec_high  ; абсолютный номер сектора старшая часть
        CALL 3D13h      ; При исполнении производится вся подготовка к
                        ; операциям чтения/записи вычисление
; цилиндров/головок/секторов и занесение их в регистры винчестера
; далее программа может сама только подать команду читать/писать и
; самостоятельно производить считывание/запись данных в винчестер.
; Команда удобна для работы программ в реальном времени, когда необходимо
; кроме чтения/записи данных производить какие либо иные действия.

HDD_READ_BPB:           ; читать BPB
        LD C,044h       ; номер функции
        LD HL,bufer_adr ; адрес буфера для BPB
        LD A,bufer_page ; страница буфера, если адрес в окне 0C000h
        CALL 3D13h      ; NC -> нормальное завершение
                        ; CF -> ошибка

HDD_READ:               ; читать сектора с винчестера
        LD C,045h       ; номер команды
        LD HL,bufer_adr ; адрес буфера данных
        LD A,bufer_page ; страница буфера, если адрес в окне 0C000h
        LD B,sec_num    ; число читаемых секторов
        LD DE,sec_low   ; абсолютный номер сектора младшая часть
        LD IX,sec_high  ; абсолютный номер сектора старшая часть
        CALL 3D13h      ; NC -> нормальное завершение
                        ; CF -> ошибка

; ** При попадании межсекторного промежутка на адрес 0000h
; производится автоматическое переключение страницы ОЗУ по
; RAM Allocation Table.

HDD_WRITE:              ; писать сектора на винчестер
        LD C,046h       ; номер команды
        LD HL,bufer_adr ; адрес буфера данных
        LD A,bufer_page ; страница буфера, если адрес в окне 0C000h
        LD B,sec_num    ; число записываемых секторов
        LD DE,sec_low   ; абсолютный номер сектора младшая часть
```

```
        LD IX,sec_high  ; абсолютный номер сектора старшая часть
        CALL 3D13h      ; NC -> нормальное завершение
                        ; CF -> ошибка

; ** При попадании межсекторного промежутка на адрес 0000h
; производится автоматическое переключение страницы ОЗУ по RAM
; Allocation Table.

HDD_PART:               ; настройка партиций и master/slave
        LD C,047h
; * функция зарезервирована

; В ближайшее время в описание биоса будут добавлены функции работы с FDD и
; CD-ROM
; * В данный момент они имеются, но предполагается их серьезная переделка.
; Некоторые функции зарезервированы для дальнейшего развития. Так же не
; описана часть графических функций, так как они в данный момент подвергаются
; серьезным переделкам.

        LD C,50H        ; зарезервирована

DRV_RESET:              ; Сброс контроллера и настройка на диск
        LD A,drv_type   ; бит 0..3 - номер устройства
                        ; бит 4..7 - тип устройства
                        ;     0 - дисковод
                        ;     6 - ram-disk
                        ;     8 - HDD
                        ;     C - CD-ROM
                        ; остальные номера резервные
        LD C,51h        ;
        CALL 3D13H      ; NC - нормальное завершение
                        ; CF - нет диска или нет устройства

        LD C,52h        ; зарезервировано
        LD C,53h        ; зарезервировано

DRV_VERIFY:             ; проверка секторов
        LD A,drv_type   ; бит 0..3 - номер устройства
                        ; бит 4..7 - тип устройства
                        ;     0 - дисковод
                        ;     6 - ram-disk
                        ;     8 - HDD
                        ;     C - CD-ROM
        LD HL,sec_h     ; страшая часть номера сектора
        LD IX,sec_l     ; младшая часть номера сектора
        LD B,n_sec      ; количество секторов
        LD C,54h        ;
        CALL 3D13h      ; NC - нормальное завершение
                        ; CF - проверка с ошибкой или нет устройства

DRV_READ:               ; чтение с устройства
        LD A,drv_type   ; бит 0..3 - номер устройства
                        ; бит 4..7 - тип устройства
                        ;     0 - дисковод
```

```
                        ;     6 - ram-disk
                        ;     8 - HDD
                        ;     C - CD-ROM
        LD HL,sec_h     ; страшая часть номера сектора
        LD IX,sec_l     ; младшая часть номера сектора
        LD B,n_sec      ; количество секторов
        LD DE,bufer_adr ; адрес буфер для чтения
        LD C,55h        ;
        CALL 3D13h      ; NC - нормальное завершение
                        ; CF - ошибка чтения или нет устройства

DRV_READ:               ; запись на устройства
        LD A,drv_type   ; бит 0..3 - номер устройства
                        ; бит 4..7 - тип устройства
                        ;     0 - дисковод
                        ;     6 - ram-disk
                        ;     8 - HDD
                        ;     C - CD-ROM
        LD HL,sec_h     ; страшая часть номера сектора
        LD IX,sec_l     ; младшая часть номера сектора
        LD B,n_sec      ; количество секторов
        LD DE,bufer_adr ; адрес буфер для чтения
        LD C,56h        ;
        CALL 3D13h      ; NC - нормальное завершение
                        ; CF - ошибка чтения или нет устройства

DRV_DETECT:             ; определение параметров устройства
        LD A,drv_type   ; бит 0..3 - номер устройства
                        ; бит 4..7 - тип устройства
                        ;     0 - дисковод
                        ;     6 - ram-disk
                        ;     8 - HDD
                        ;     C - CD-ROM
        LD C,57h        ;
        CALL 3D13h      ; NC - нормальное завершение
                        ;      A - bit7 - 0 диск 720Kb
                        ;                 1 диск 1.44Mb
                        ; CF - нет устройства или нет носителя

DRV_GET_PAR:            ; получить параметры носителя
        LD A,drv_type   ; бит 0..3 - номер устройства
                        ; бит 4..7 - тип устройства
                        ;     0 - дисковод
                        ;     6 - ram-disk
                        ;     8 - HDD
                        ;     C - CD-ROM
        LD C,58h        ;
        CALL 3D13h      ; NC - нормальное завершение
                        ;    L - число секторов
                        ;    H - число головок
                        ;    DE - количество цилиндров
                        ;    IX - размер сектора в байтах
                        ;    B - доп. параметры
                        ;       для дискет бит7 - тип 1.44/720
```

```
                        ; если в HL,DE все FF - устройства нет
                        ; CF - нет устройства

DRV_SET_PAR:            ; установить параметры носителя
        LD A,drv_type   ; бит 0..3 - номер устройства
                        ; бит 4..7 - тип устройства
                        ;     0 - дисковод
                        ;     6 - ram-disk
                        ;     8 - HDD
                        ;     C - CD-ROM
        LD L,n_secs     ;    L - число секторов
        LD H,n_heads    ;    H - число головок
        LD DE,n_cyls    ;    DE - количество цилиндров
        LD IX,sec_size  ;    IX - размер сектора в байтах
        LD B,ext_par    ;    B - доп. параметры
                        ;       для дискет бит7 - тип 1.44/720
        LD C,59h        ;
        CALL 3D13h      ; NC - нормальное завершение

; Функции не отмеченные как зарезервированные, старые или тестируемые,
; меняться скорее всего не будут.
```

21 Дополнительные сведения по программированию

21.1 Вывод на графический экран

В биосе имеются функции открытия графического экрана на весь экран 320x256 точек. После открытия этого режима экран представляет собой. набор из 256-ти линий, длиной по 320 байт. Соседние точки в линии - это соседние байты. Переключение линий производится через PORT_Y, в котором устанавливается номер линии, выводимой на экран. Номера линий считаются сверху экрана, начиная с нулевой.

Для вывода в графический экран так же требуется открыть соответствующую страницу основного ОЗУ. В этой странице будет содержаться копия видеоизображения.

Видео-ОЗУ является теневым ОЗУ, поэтому информация, находящаяся в основном ОЗУ, под которым находится видео-ОЗУ не обязательно будет совпадать с информацией, находящейся в этом видео-ОЗУ. Запись видео-данных может производиться и без перезаписи данных в оснвном ОЗУ, что оказывается полезным при работе, например, со спрайтами. Для работы со спрайтами так же предусмотрен режим записи в видео-ОЗУ с прозрачным цветом. В этом режиме информация, передаваемая в видео-ОЗУ проверяется на наличие байта #FF. Если этот байт обнаруживается, то цикл записи пропускается и на экране в этом месте остается те данные, какие были ранее. Таким образом на экране можно быстро прорисовывать спрайты, представляющие из себя прямоугольные картинки с «прозрачными» цветами.

Пример программы вывода прямоугольной картинки на экран:

```
PAGE3   EQU #E2
RGADR   EQU #89

        LD A,#50        ; страница графического видеоэкрана
        OUT (PAGE3),A   ; установить в PAGE3
        LD HL,Pucture   ; адрес картинки (Прямые Данные)
        LD DE,#C040+HorPlace  ; положение картинки на экране по горизонтали
        LD A,VerPlace   ; положение картинки на экране по вертикали
        OUT (RGADR),A
        LD B,VerSize    ; высота картинки
LOOP:   PUSH DE         ; запомнить положение на линии
        PUSH BC         ; запомнить счетчик высоты
```

```
LD BC,HorSize   ; длина картинки
LDIR            ; копировать линию
POP BC
POP DE
INC A           ; следующая координата по Y
OUT (RGADR),A
DJNZ LOOP       ; повторять нужное количество раз
```

Управление режимом вывода на экран (включение вывода с прозрачными цветами, отключение копирования в основное ОЗУ) осуществляется через младшие биты порта страницы графического экрана.

21.2 Особые режимы

Режим Bound, программирование режимом экрана (быстрые скроллинги).

21.3 Вывод палитр

Вывод палитры – функция БИОС.

Часть V
Программирование в TR-DOS (дополнительные команды)

22 Работа с HDD и RAM-Disk через TR-DOS

Команды вводятся как обычно в TR-DOS строке побуквенно. Все команды доступные из командной строки работают и из BASIC-а через:

RANDOMIZE USR 15619: REM:comand

При наборе команд через знак "/"следует иметь в виду, что лишние пробелы в начале мешают ее исполнению (система распознает новую команду по знаку "/"в начале команды).

При наборе имен файлов MS-DOS допускаются звездочки и вопросики, по правилам: * – любой набор символов, ? – один любой символ.

22.1 /HDT

/HDT – тест винчестера. Определяет тип винчестера и выводит на экран его марку. Если этого не происходит, это значит, что винчестер либо неверно подключен, либо незовместим (бывает с очень старыми моделями).

22.2 /HDD

/HDD – подключение винчестера к текущему диску A,B,C или D – к тому, на который указывает TR-DOS в начале в командной строки.

22.3 /CAT, /DIR

/CAT /DIR – выдача каталога. Команды **/CAT** и **/DIR** идентичны по исполнению. Работают так же команды типа:

/DIR *.trd – выдача каталога файлов с расширением .trd

/DIR a*.dm* – выдача файлов, имена которых начинаются на a, а расширения на dm
/DIR ??xm.* – выдача файлов, имена которых имеют четыре буквы и оканчиваются на xm

22.4 /LOAD, /SAVE

/LOAD x file.ext – загрузка в RAM-Disk x файла file.ext RAM-Disk будет автоматически создан (или пересоздан, если такой был)

Сокращенная команда: **/file.ext** – загрузка файла в RAM-Disk E

/SAVE x file.ext – сохранение RAM-Disk-а в файла file.ext. При сохранении происходит замещение содержания существующего файла. Новый не создается, длина файла не меняется.

При загрузке и сохранении работают и команды со звездочками и вопросиками, но выполняется только с первым найденым файлом. Например, команда **/*.*** загрузит в RAM-Disk E самый первый файл с винчестера.

ВНИМАНИЕ: Команда **/SAVE** на данный момент является единственной командой TR-DOS, которая меняет содержимое винчестера. При своей работе она ЗАМЕЩАЕТ содержимое файла образом RAM-Disk-а. Если длина файла меньше длины RAM-Disk, то запишется лишь часть данных RAM-Disk-а, по количеству секторов файла.

22.5 /FDD

/FDD – подключение к текущему диску обычного дисковода. Если к этому диску был подключен винчестер или RAM-Disk, то они будут отключены и доступ с этого диска будет осуществляться к обычной дискете.

После команды **/FDD** работают все команды **/SAVE**, **/LOAD**, **/DIR** с дискетами в формате MS-DOS. 1.44 или 720 диск распознается автоматически.

После нее же работают и все стандартные команды TR-DOS, а так же расширенные команды из TR-DOS-5.04Em.

22.6 /RMD

/RMD X – подключение к текущему диску рамдиска X (буквы от E до S). После этой команды текущий дисковод работает с RAM-Disk-ом в только формате TR-DOS. Выполняются все TR-DOS команды, в том числе и FORMAT. При форматировании старое содержимое диска уничтожается, диск приобретает размер 640Kb и в него прописывается пустой каталог.

ВНИМАНИЕ! Если RAM-Disk не был создан или подгружен с винчестера или дискеты, все попытки чтения каталога, файлов и т.п. приведут к появлению ошибки.

Создать RAM-Disk можно командой: **FORMAT "name"**. Обязательно после того, как командой **/RMD X** был подсоединен нужный RAM-Disk.

Подгрузка RAM-Disk-а осуществляется командой **/LOAD**. При этом RAM-Disk создается автоматически. Работа с TR-DOS на этом RAM-Disk-е будет возможна только если подгруженный файл был образом RAM-Disk-а. Если же это не так, то доступ к RAM-Disk-у все равно останется через команды чтения/записи секторов, вызываемые через точку #3D13. Таким образом можно подгружать и использовать в программах любые файлы.

22.7 /CLEAR

После того как RAM-Disk X сохранен (и не обязательно после этого) он может быть удален из памяти командой.

/CLEAR x

Объем памяти компьютера ограничен, поэтому не обязательно все 16 RAM-Disk-ов могут в нем уместиться. Кроме того, размер RAM-Disk определяется с точностью до 16Kb и может занимать все пространство памяти т.е. до 3.4Mb (Остальная память – 256K видео, 256K спектрумовская память и 80K системной памяти всегда заняты.)

22.8 Команды переключения конфигураций

TR-DOS предоставляет простое средство для преключения трех конфигураций, зашитых в ПЗУ непосредственно из командной строки. Для этого служат команды: **/Sprinter 1**, **/Sprinter 2** и **/AY**. Следует иметь в виду, что после переключения в Sprinter-2 спектрумовская клавиатура перестанет работать, и в случае вызова этой команды с консоли произойдет «подвисание». Использование этой команды может быть целесообразно в BASIC программах, которые переключат режим, выполнят некоторые действия и вернут режим в Sprinter 1 или AY, когда вновь возможно использование спектрумовской клавиатуры.

22.9 Дополнительные сервисные команды

Кроме описанных есть еще несколько сервисных команд:

/A: /B: /C: /D: – переключение текущих дисководов без проверки их наличия. Удобно использовать из бейсика при работе с рамдисками, когда надо убрать возникновение ошибок из-за переключения на дисковод, к которому еще ничего не подключено.

/RD – тестовая команда. Выводит внутреннюю таблицу Ram Allocation Table (подобна FAT), по которой можно увидеть какие страницы ОЗУ заняты, какие свободны. Таблица выводится сплошными цифрами в HEX формате. Каждая пара это номер, соответстующий некой странице. Если этот номер равен 00, это означает, что страница свободна. Таблица 16x16 двухбуквенных номеров соответствует 256-ти страницам основного ОЗУ.

Создание новых файлов на HDD или дискетах MS-DOS на данный момент возможно только через командеры или функции DOS, вызываемые из ассемблера. Аналогично, копирование, удаление, перемещение, создание/удаление каталогов и т.д.

Из TR-DOS можно работать только с файлами корневого каталога. Единственная команда TR-DOS меняющая содержимое HDD – команда **/SAVE**, замещающая содержимое файла, в который производится сохранение RAM-Disk-а.

Большие и маленькие буквы в названиях MS-DOS файлов считаются одинаковыми.

23 Дополнения TR-DOS 5.04Em

TR-DOS 5.04Em, на основе которого написан TR-DOS Спринтера, использовался в компьютерах Peters-256.

TR-DOS 5.04Em является дальнейшим развитием версий 5.04E, 5.04E+. По сравнению с версией 5.03 в ней имеются следующие дополнения:

Форматирование дискет в 3-х вариантах:

ORIGINAL – обычный формат TR-DOS;

TURBO – форматирование диска с другим расположением секторов, что приводит к ускорению операций чтения/записи диска примерно в 2 раза. однако, применение этого формата замедляет выполнение команды **VERIFY**, но эта команда реально почти не используется.

TURBO-FAST – форматирование аналогично TURBO, но без операции проверки. Этот режим стоит применять только если есть уверенность в качестве дискет. Применение команды немного ускоряет саму операция форматирования.

QUICK – быстрый формат. Команду можно применять, в случае, если диск уже отформатирован и надо просто удалить с него все файлы. TR-DOS не проводит физического форматирования, а просто прописывает в первых секторах пустой каталог. Следует отметить, что этот формат не годится для односторонних и 40-дорожечных дисководов, так как диск не проверяется и в него прописываются данные, что он двухсторонний, имеет 80 дорожек и 2544 свободных сектора. QUICK-формат автоматически используется для форматирования RAM-Disk-ов.

Выбор формата производится по запросу DOS после ввода обычной команды **FORMAT "name"** клавишами <1>, <2>, <3> или <4>. Клавиша <SPACE> отменяет команду. Если необходимо форматировать

диск из программы, так что бы TR-DOS не запрашивал тип формата, вслед за именем диска, сразу после кавычки надо поставить цифру 1, 2, 3 или 4, в соответствии с необходимым типом формата.

При форматировании на экран выдается сообщение о номере форматируемой в данный момент дорожки и стороны диска.

При записи файла на диск в случае если файл с таким именем на диске есть DOS выдает запрос о стирании старого файла. Если ответить <Enter> или <Y>, то старый файл будет стерт и на его место будет записан новый файл, если он не длиннее старого. Если же новый файл длиннее, то он будет записан как обычно с первого свободного сектора диска, а в каталоге появится еще один стертый файл. В версии 5.03 такой случай просто приводит к остановке программы с ошибкой.

При включении DOS сразу устанавливает, что дисководы A: и B: имеют тип 2x80.. Это приводит к отсутствию начальной проверки дисковода (нет звука "др-др-др"при первом выборе нового дискла). В процессе работы диск проверяется на тип, и DOS работает с ним как надо.

Ускоренная команда **MOVE**. В обычной версии 5.03 команда **MOVE** наиболее длинная по времени работы, кроме того создается впечатление, что дисковод больше стоит чем работает. Это объясняется тем что каждый раз при перемещении файла производится чтение каталога и запись в него. В данной версии чтение и запись каталога производятся один раз. Информация каталога сохраняется в ОЗУ и все операции с каталогом (только в команде **MOVE**) производятся в ОЗУ, что значительно ускоряет работу команды, в среднем в 3 – 4 раза, а в некоторых случаях и более.

Введена новая команда – переименование диска: **MOVE "name"**. При этом команда **MOVE** не выполняется, а просто имя диска заменяется на **"name"**.

В TR-DOS 5.04Em изменен редактор командной строки. Во-первых, устранена неприятность которая возникает при входе в DOS в 128 режиме. Во-вторых, устранено влияние клавиш редактирования, которые перемещают курсор строки в 48-м режиме. И, наиболее важная часть, - возможность вызова предыдущей команды с помощью клавиши <EDIT>. Если Вы хотите повторить операцию или повторить ее с немного измененными параметрами, то можно вызвать ее сразу после выполнения нажав клавишу <EDIT>. Если же Вы что-то набрали в командной строке, то клавиша <EDIT> не сработает (в 5.03 версии это приводит к вызову строки из BASIC программы).

TR-DOS 5.04Em поддерживает работу с RAM-диском, основанном на дополнительном ОЗУ – 128k. Эта функция заменена на более модернизированные функции работы с RAM-Disk-ами Спринтера, описанные выше.

24 Копирование файлов с TR-DOS дискет в RAM-Disk

Последовательность команд для достижения эффекта копирования в RAM-Disk E содержимого дискеты A:

***"A:"** – переключение на диск A
/FDD – подключение к диску A обычного дисковода
***"B:"** – переключение на диск B
/RMD E – подключение к диску B рамдиска E
FORMAT "name" – форматирование RAM-Disk-а (можно не форматировать, а подгрузить файл с HDD командой **/LOAD E file.trd**)
COPY "b:*","a:*" – стандартная команда TR-DOS копирования всех файлов с A: на B:

Действия собственно по копированию с дискет на RAM-Disk могут быть совершены стандартными спектрумовскими командерами, работающими с TR-DOS через точку входа #3D13.

После этих команд на диске B: будет виден обычный каталог (команда TR-DOS: **CAT**) и с этими файлами можно работать как с обычными TR-DOS файлами.

После того, как нужные файлы скопированы на RAM-Disk, его следует сохранить, как описано в следующей секции.

25 Сохранение содержимого RAM-Disk на винчестере

Для примера – RAM-Disk E.

К какому дисководу был подключен RAM-Disk (и был ли подключен вообще) не имеет значения. Имеет значение, что этот RAM-Disk существует в памяти, т.е. был либо создан командой **FORMAT**, либо подгружен ранее с HDD.

Последовательность команд:

/HDD – подключение винчестера (какой диск стоял по умолчанию не имеет значения сработает на любом)

/SAVE E file.trd – сохранение RAM-Disk-а E в файл file.trd

Сохранение подгруженного с HDD RAM-Disk обратно, естветственно, стоит делать только в том случае, если в образе диска произошли изменения. Для избежания возможных глюков рабочие программы стоит держать в отдельном RAM-Disk-е, а изменяемые файлы на другом и иметь резервные копии файлов .trd с важными данными (это общая рекомендация для любых компьютерных систем).

Часть VI
Программирование в Sprinter-DOS

(от Дениса)

А этот текст для того что бы LaTeXне ругался на пустоту в параграфе. А то, блин, развели тут, панимашь, пустых парахрафов, как мух нерезаных!