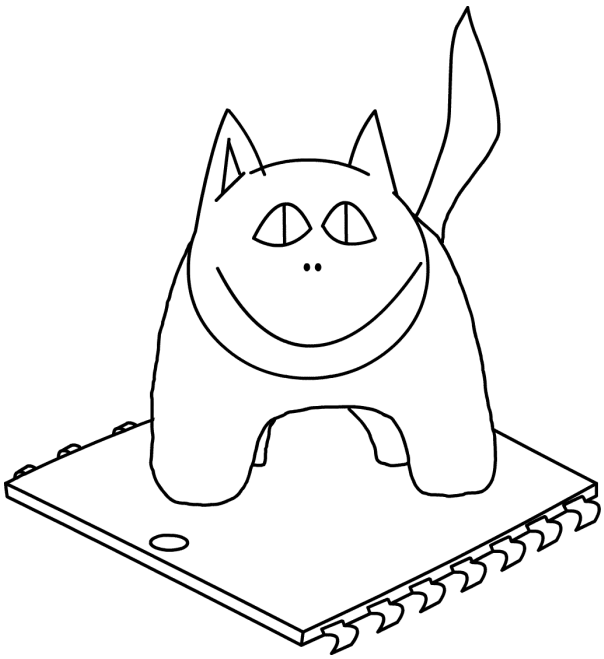

Turbo 2+

Персональный компьютер Turbo 2+

Внутренняя архитектура
и внешние устройства
(издание второе, исправленное)

Москва 2005

Оглавление

I. Внутренняя архитектура

1. Общие принципы.

Спектрум-совместимый ПК ATM-turbo 2+ (дальше – просто ATM-2+), обладая всеми возможностями данного класса машин, одновременно сильно выделяется из их числа за счет на порядок более гибкой архитектуры и обширного списка внешних устройств, интегрированных в материнскую плату.

Гибкость архитектуры определяется тем, что, если в большинстве современных моделей ZX-SPECTRUM из трех основных компонентов: управление портами, графическими режимами и структурой памяти – реализовано в зачаточном состоянии только последнее, то в ATM-2+ возможно изменять режимы функционирования всех компонентов в достаточно широком диапазоне.

2. Условные обозначения.

При характеристике портов используются следующие обозначения:

OUT – порт используется для вывода данных в него.
IN – порт используется для ввода данных из него.
Dn – шина данных. **n** = 0 – 7.
An – шина адреса. **n** = 0 – 15.
#nnnn – шестнадцатиричное значение порта. **n** = 0 – F
%nnnnnnnnnnnnnnnn – двоичное значение порта. **n** = 0 – 1
Здесь же: **x** = адресные биты, значение которых может меняться.
n = (как отдельная буква) значение данного бита адреса может быть любым.

Дополнительно в скобках указываются принудительно установленные или сброшенные биты.

3. Конфигурация карты портов.

В ATM-2+ существует два основных вида портов: **открытые** и **теневые**.

Открытые порты: характерны тем, что всегда присутствуют в адресном пространстве и до них в любом режиме работы компьютера можно добраться прямой командой OUT/IN.

К портам такого типа прежде всего относятся все стандартные порты ZX-SPECTRUM 48/128, а также и некоторое количество специфических портов ATM-2+.

Теневые порты: характерны тем, что доступны только при определенных условиях, а в обычном режиме работы в адресном пространстве отсутствуют. К ним относится большинство специфических (нехарактерных для ZX-128) портов ATM-2+, а также порты Beta-Disk интерфейса (дальше – порты TR-DOS). Последние имеются также и в обычных моделях ZX-128 с TR-DOS. Но, в отличие от данных моделей Спектрума, ATM-2+ имеет дополнительные возможности управления всеми теневыми портами.
Существует два способа работы с теневыми портами:

Первый, доступный и в остальных Спектрумах, осуществляется путем активизации сигнала **DOSEN**, которая открывает теневые порты для программного доступа. Эта активизация происходит при одновременном выполнении следующих условий:

a) Сигнал **ROM2**=1 (в порту #7FFD бит D4=1. В обычном ZX это означает активность ПЗУ BASIC-48).
b) Передача управление на адреса из промежутка #3Dxx, #7Dxx, #BDxx, #FDxx с обязательным наличием в этом промежутке ПЗУ (сигнал **ROM** должен быть активен). Так как в обычных Спектрумах ПЗУ может находиться только по адресам #0000-#3FFF, то, по факту, возможно пользоваться только промежутком #3Dxx. При этом происходит включение ПЗУ TR-DOS.
c) После активизации DOSEN работа с теневыми портами возможна вплоть до выключения сигнала ROM, то есть, вплоть до исполнения любой команды из ОЗУ, после чего сигнал DOSEN отключается, закрывая для доступа теневые порты.

Пример работы с теневыми портами в обычном ZX-режиме ATM-turbo 2+:

```
        CALL SHADOW     ; адрес возврата на стек
        …..
SHADOW  LD   BC,#2A53   ; адрес в ПЗУ TR-DOS, где лежит команда OUT (C),A
        PUSH BC         ; адрес на стек
        LD   BC,#nnnn   ; конкретный номер теневого порта
        LD   A,#nn      ; значение, выдаваемое в порт
        JP   #3D2F      ; переход в промежуток #3Dxx на команду RET
```

(в дальнейшем, при приведении примеров работы с теневыми портами, будет подразумеваться, что они уже включены вышеприведенным образом)

Второй способ существует только в ATM-2+: по нему открыть теневые порты можно путем сброса определенного адресного бита в специальном **системном порту** (о нем ниже отдельно), после чего все теневые порты открываются для прямого доступа (а в ZX-режиме автоматически включается и ПЗУ TR-DOS). Установка этого бита обратно скрывает теневые порты из адресного пространства.

Особенность применения данного метода заключается в том, что системный порт сам относится к подмножеству теневых портов. Поэтому для первоначального доступа к нему (в случае, если теневые порты изначально спрятаны) надо также применить первый способ, а уж потом только работать с ним (и другими портами) напрямую.

При определенных вариантах конфигурации памяти, может возникнуть ситуация, когда ПЗУ будет полностью выключено из адресного пространства. В этом случае, первый способ доступа к теневым портам через точки #3Dxx лишается смысла, так как не произойдет активизации DOSEN. Если прямой доступ к теневым портам в таком режиме будет отключен, то снова получить к ним доступ можно будет только по сигналу RESET.

3.1. Системный порт ATM-turbo 2+ (теневой)

out #FF77 = %1x1111xx01110111 (A0=A2=A4=1; A3=A7=0)

Значения шины данных:

D0 - RG0 \
D1 - RG1 > переключение экранных режимов.
D2 - RG2 /
RG0=1 RG1=1 RG2=0 - обычный sinclair режим 256x192 пикселей
RG0=0 RG1=1 RG2=0 - аппаратный мультиколор 640x200 пикселей
RG0=0 RG1=0 RG2=0 - EGA 320x200 (16 цветов) пикселей
RG0=0 RG1=1 RG2=1 - текстовый режим 80x25 символов

Остальные комбинации этих трех сигналов включают т.н. «технологические» экраны, являющиеся недокументированным побочным результатом схемы материнской платы и имеющие нелинейную структуру, что затрудняет их использование в программных целях.

D3=1 - программное включение **TURBO**-режима (7.0 МГц). **D3=0** – 3.5МГц.
D4 - для 7.10 - сигнал **Z_1** - зарезервирован, необходимо устанавливать в 0.
D5 - для 7.10 - **Z_I** прерывания от HSYNC (50 Гц) 1-разрешены 0- запрещены.
D6 - сигнал **VE1** (**=1** - запрет функционирования 8031 (все запросы идут к zx-клавиатуре))
D7 – сигнал **VE0** (не используется, по умолчанию необходимо устанавливать в 1)

Значения шины адреса:

A8=0 - сигнал **PEN**, выключение диспетчера памяти (о нем позже) - в каждую четверть адресного пространства памяти устанавливается страница ПЗУ с BIOS CP/M... При включении в сеть компьютера диспетчер как раз выключен.
Здесь такой порядок расположения страниц ПЗУ (число в скобках - значение при ПЗУ 271000):

0(4) - BASIC 48
1(5) - TR-DOS
2(6) - BASIC 128
3(7) - CP/M

Если подключено ПЗУ 271000, то 0-3 - это дополнительные страницы, а 4-7 - вышеприведенная прошивка. При подключении еще более объемной ПЗУ, данные страницы всегда будут иметь четыре последних (максимальных) номера. Это – стандарт!

A9=0 - сигнал **CPM** - включается постоянное действие сигнала **DOSEN**, в результате становится возможным прямой доступ ко всем скрытым портам и ПЗУ TR-DOS. Если **A9=1**, то доступ ко всем расширенным портам, в том числе и к этому, прекращается. Таким образом, сбросить этот бит обычными методами нельзя - можно это сделать только из области TR-DOS, где эти порты открываются автоматически (аппаратно). Оттуда

можно сделать OUT со сброшенным **A9**, и тогда при выходе из TR-DOS расширенные порты просто не выключатся, да и ПЗУ TR-DOS останется в области #0000-#4000 (если, конечно, не установить туда ОЗУ).
A14=0 - сигнал **PEN2**, разрешает запись палитры (об этом позже) через дисководный порт **#FF** (о нем ниже).

Рекомендуется пользоваться при работе с системным портом следующими значениями:

a) ***#4177 – для включения теневых портов.***
b) #0177 – для включения теневых портов и разрешения доступа к палитре.
c) ***#FF77 – для выключения теневых портов.***

4. Принципы управления памятью в ATM-turbo 2+

Память в ATM-2+ бывает двух типов:

ПЗУ (Постоянное Запоминающее Устройство) – информация в ней (самые необходимые для первоначального старта и работы процедуры) записывается при помощи специальных внешних устройств - программаторов, не может быть изменена программными средствами и не пропадает после выключения питания.

ОЗУ (Оперативное Запоминающее Устройство) – представляет собой память, содержимое которой можно изменять программными средствами, но которое теряется после выключения питания.

ATM-2+ позволяет адресовать до 1024Кб ОЗУ и 1024 Кб ПЗУ, хотя их иногда бывает подключено и меньше (особенно ПЗУ).

Процессор Z80, однако, позволяет адресовать одновременно лишь 64Кб или 65536 байт, что является лишь небольшой частью от общего количества ОЗУ и ПЗУ. Чтобы решить проблему доступа ко всему объему памяти, был реализован страничный принцип адресации памяти.

Суть его заключается в том, что вся память разделена на несколько частей, из которых для работы одновременно доступна только одна. Но эти части можно менять местами (отключить от доступа прежнюю и включить какую-то другую) при помощи задания определенного значения в специальных портах управления памятью (о них - ниже).

Конкретно в ATM-2+ этот принцип реализован следующим образом:
ОЗУ и ПЗУ разбито на страницы по 16Кб каждая (соответственно из 1024Кб получаем 64 страницы ОЗУ (пронумерованных от #00 до #3F) и 64 страницы ПЗУ). Адресное пространство Z80 (далее – карта памяти) также разбито на четыре участка-окна (далее – просто «окна», а на рисунках: CPU 0 – CPU 3) по 16Кб каждое (см. рис.1).

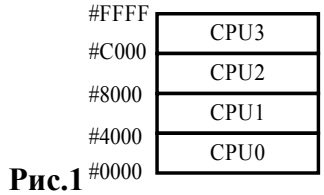

Рис.1

Архитектура АТМ-2+ устроена таким образом, что в любое из этих окон может быть включена любая из 64 страниц ОЗУ или 64-х (возможных) ПЗУ. Эта архитектура расширенная по сравнению с обычными ZX-SPECTRUM 128, где в окне 0 можно переключать только страницы ПЗУ (и только в некоторых клонах включать страницу 0 ОЗУ), в окне 1 постоянно находится страницы 5 ОЗУ, а в окне 2 – постоянно страницы 2 ОЗУ. И только в окне 3 по адресу #C000 можно менять любые страницы ОЗУ (и только ОЗУ).

Кроме того, в АТМ-2+ не одна, а две карты памяти (то есть два альтернативных адресных пространства), переключаемых по сигналу ROM2 (бит D4 порта #7FFD). То есть имеется возможность запрограммировать два независимых друг от друга расположения страниц в окнах 0-3 и быстро переключаться между ними через порт #7FFD. Именно так эмулируется в АТМ-2+ архитектура стандартного ZX-128: две карты памяти программируются идентично, за исключением окон 0 в обеих картах. В одной карте там ПЗУ BASIC 48, а в другой – ПЗУ BASIC 128, в результате переход из одной карты в другую означает лишь переключение страниц ПЗУ в соответствии со стандартом ZX-128. Архитектура памяти АТМ-2+ представлена на рисунке 2:

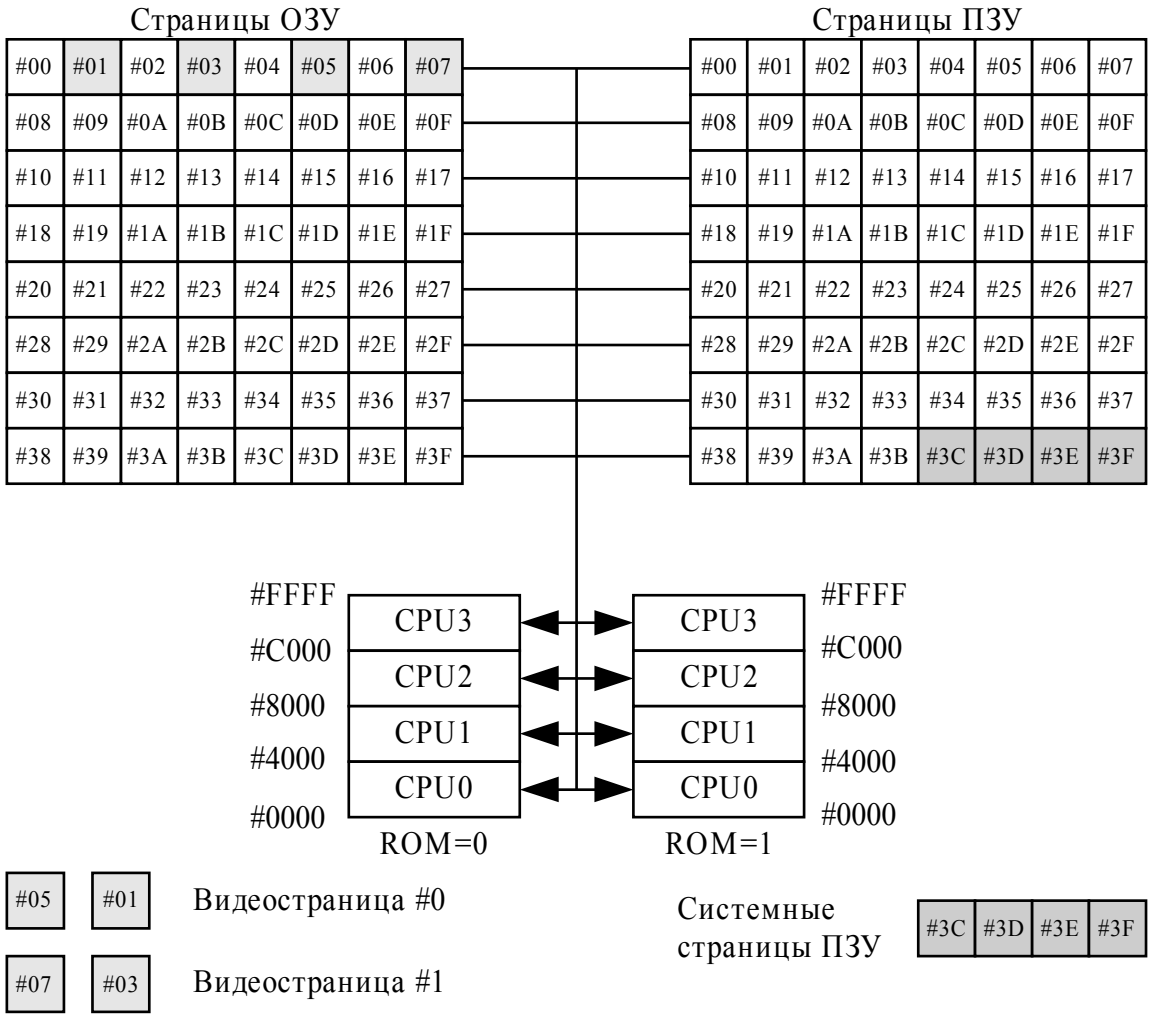

Рис.2

Карта памяти (текущая) АТМ-2+ программируется через использование целой серии портов.

4.1. Диспетчер памяти ATM-turbo 2+ (теневой)

out #FFF7 = %xx11111111110111 (A0=A2=A4=A7=1; A3=0)

Значения шины данных:

D0–D5 – инверсный номер страницы ОЗУ или ПЗУ (#00-#3F). То есть, если надо выбрать, к примеру, страницу #05 (двоичное значение %000101), то в эти биты надо выдать его инверсное значение - %111010.
D6 – определитель того, что выбирается битами D0-D5 – ОЗУ или ПЗУ.

D6=0 – ПЗУ
D6=1 – ОЗУ

D7 – коммутатор диспетчера памяти и страничного порта **#7FFD** (о нем ниже).

D7=0 – номер страницы памяти полностью формируется из инверсных битов **D0–D5** диспетчера памяти (о чем сказано выше).

D7=1 – при формировании страницы памяти принимает участие порт **#7FFD**:

а) **Выбор ОЗУ:** при формировании номера страницы инверсные биты **D0-D2** диспетчера памяти подменяются неинверсными битами **D0-D2** порта **#7FFD**. Остальные разряды берутся, как и раньше, из инверсных битов **D3-D5** диспетчера памяти.

Пример (предполагается, что теневые порты открыты):

```
LD    A, #2B       ;номер страницы 43 dec
PUSH  AF
SCF                ;инвертируем для диспетчера памяти
AND   %11111000    ;D7=1 – коммутация #7FFD включена.
LD    BC, #FFF7    ;включение страницы в окно 3 (#C000)
OUT   (C), A       ;выводим в порт биты 3-5 страницы
POP   AF           ;восстанавливаем номер страницы.
AND   %00000111    ;отсекаем лишние биты.
LD    BC, #7FFD
OUT   (C), A       ;выводим в порт биты 0-2 страницы
```

б) **Выбор ПЗУ:** разрешено автоматическое подключение ПЗУ TR-DOS (**D0** диспетчера памяти подменяется состоянием сигнала **DOSEN**, индицирующим включение скрытых портов) при переходе на смещение **#3Dxx** того окна, где данное ПЗУ с включенным коммутатором установлено при одновременном установленном сигнале **ROM2** (об условиях включения TR-DOS и включения скрытых портов смотрите выше). В противном случае переход на эти адреса ничего не меняет.

Значения шины адреса:

A14-A15 - выбор четверти адресного пространства, где надо изменять страницы (то есть где будут действовать все установленные или сброшенные биты **Dn** и **An** рассмотренные у этого порта выше).

A14	A15	Окно включения	Порт
0	*0*	*#0000-#3FFF*	*(порт #3FF7)*
1	*0*	*#4000-#7FFF*	*(порт #7FF7)*
0	*1*	*#8000-#BFFF*	*(порт #BFF7)*
1	*1*	*#C000-#FFFF*	*(порт #FFF7)*

4.2. Порт страниц ZX-128 (открытый)

out #7FFD = %0111111111111101 (A1=A15=0, A9=1*)

*(ошибка платы – в дешифрации не нужен и даже вреден! В версии от NedoPC удаляется с 20.06.2005)

Почти соответствует стандартному страничному порту в обычных моделях ZX-SPECTRUM 128.

Значения шины данных:

D0-D2 - выбор одной из восьми страниц по 16Кб из 128Кб по адресу **#C000** (а при использовании вместе с диспетчером памяти – и в других окнах).
D3 - номер видеостраницы:

D3=0 – изображение берется из страницы **5(1)**.
D3=1 – изображение берется из страницы **7(3)**.
(Подробнее о видеорежимах и их расположении смотрите ниже)

D4 – в обычном ZX-128 - выбор одной из двух страниц ПЗУ: при **0 - BASIC-128**, при **1 - BASIC-48**. В АТМ-2+ также используется для быстрой смены карт памяти, так как на каждое значение этого бита можно установить в диспетчере памяти свою.
D5 - при равенстве 1 - блокирует порт #7FFD для совместимости со Спектрумом-48.
D6-D7 - не используются.

В качестве примера на рисунке 3 приведены значения для диспетчера памяти для двух стандартных режимов АТМ-2+: CP/M и ZX-128 (при стандартном ПЗУ 64Кб):

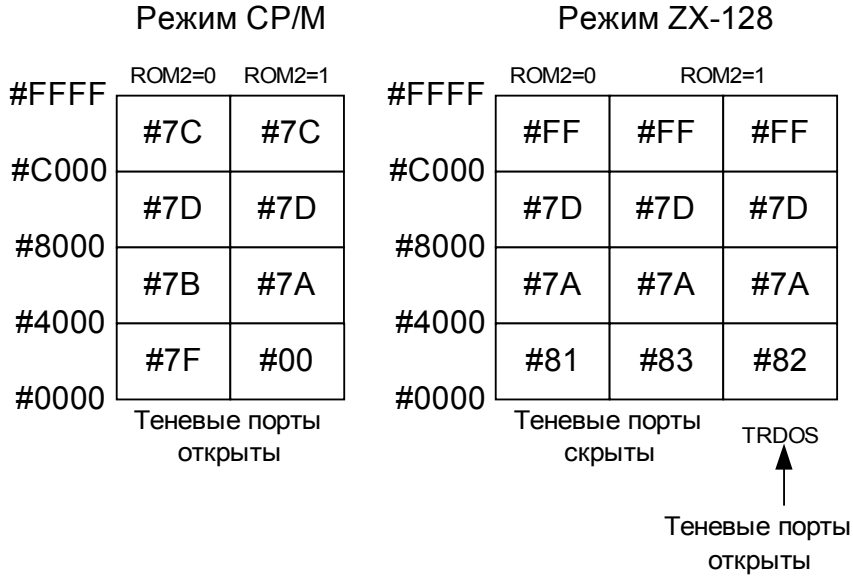

Рис.3

5. Устройство видеоконтроллера ATM-turbo 2+

Экран ATM-turbo 2+ представляет собой поле, поделенное на две неравные части:
а) Основное поле – расположено в центре общего экрана и занимает почти всю площадь. Именно сюда выводится основное изображение.
б) Бордюр – окаймляет основное поле и способен окрашиваться в заданный контроллером цвет.

Видеоконтроллер, посредством ряда портов, способен управлять пропорциями бордюра и основного экрана, пиксельным разрешением, цветовым разрешением и палитрой. В результате такой гибкости доступны несколько различных видеорежимов, управляющиеся через системный порт **#FF77** (описывался выше) сигналами RG0, RG1 и RG2.

Сигнал RG0* – регулирует пропорции экрана и пикселей.

RG0=1 соответствует стандартным пропорциям ZX-экрана (4:3). Пиксели в данном случае имеют форму квадрата с соотношением высоты к ширине как 1:1.

RG0=0 увеличивает основной экран вширь (и совсем немного по вертикали). Пиксели принимают пропорцию с соотношением высоты к ширине как 1:0.5 (то есть ужимаются в два раза). Кроме этого, перестраивается система атрибутов с атрибута на знакоместо (8x8 пикселей) на атрибут на байт (8x1 пикселей). Действие RG0 изображено на рисунке 4.

- *- изначально изменение значения сигнала RG0 на противоположное приводило к перепутыванию адресного пространства по принципу подмены местами адресов A5,A6,A7 и соответственно, A8,A9,A10. Это сильно затрудняло программный переход от ZX-экрана (единственного, при котором RG0=1) к расширенным и обратно. В версии ATM-2+ от NedoPC group это перепутывание устранено. Владельцы других версий могут сделать это самостоятельно путем перепайки одного проводка на плате.*

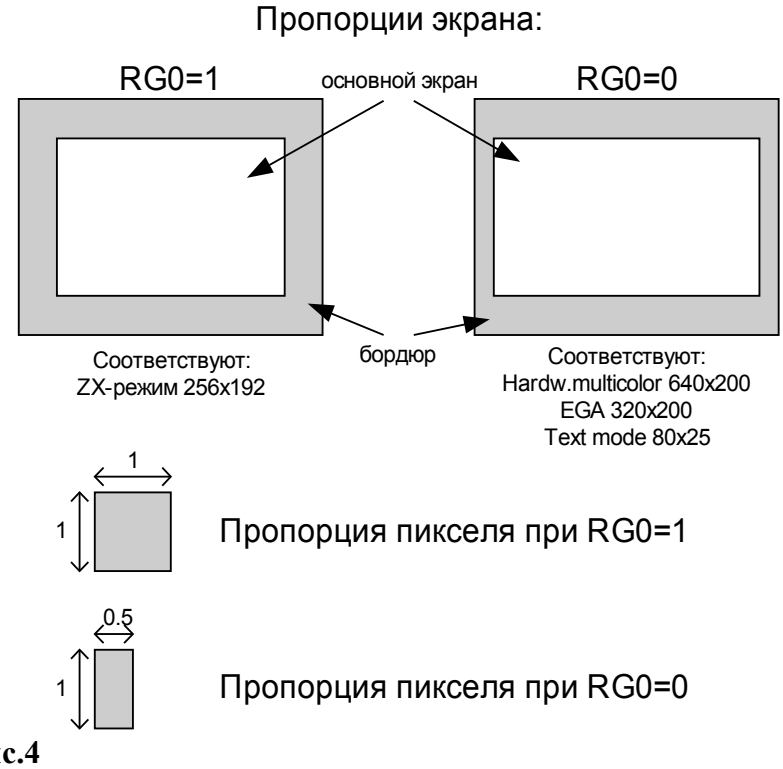

Рис.4

Сигнал RG1 – регулирует отношение битовой плоскости и атрибутов.

RG1=1 – существует отдельное поле изображения (один бит – один пиксель), и отдельное поле атрибутов (атрибут на знакоместо или атрибут на байт – в зависимости от положения сигналов RG0 и RG2).

RG1=0 – деление на изображение и атрибуты отсутствует. Все байты видеоданных преобразуются в пары пикселей (по 4 бита на каждый) с независимой цветовой окраской каждый (16 цветов). Кроме того, если был сброшен сигнал RG0, то пиксели восстанавливают свою пропорцию ширины к высоте как 1:1.

Сигнал RG2 – переключает контроллер из режима независимого отображения пикселей в режим посимвольного вывода и обратно.

RG2=1 – в ответ на вывод в видео-ОЗУ байта на экране аппаратными средствами рисуется символ 8x8 пикселей, изображение которого берется из знакогенератора в специальном ПЗУ. В этом режиме атрибут приходится на одно знакоместо. Доступ к отдельным пикселям невозможен.

RG2=0 – доступ к изображению осуществляется на уровне отдельных битов. Наличие и форма атрибутов определяется сигналами RG0 и RG1.

На основе комбинирования этих трех сигналов получаются различные видеорежимы, которые будут описаны ниже.

5.1. ZX-экран 256x192 (RG0=1, RG1=1, RG2=0)

Расположение:

Страница **#05** ОЗУ при D3=0 порта #7FFD или страница **#07** ОЗУ при D3=1 порта #7FFD. Пропорции пикселей 1:1. Размер экрана 6912 байт (непрерывных), 256 пикселей (или 32 байта (по 8 пикселей в каждом) в каждой из 192 строк)

Структура:

Смещение в странице:
#0000 - #17FF - поле монохромного изображения (1 бит на пиксель) = 6144 байт.
#1800 - #1AFF - поле атрибутов = 768 байт.

Поле монохромного изображения имеет линейную структуру по горизонтали и нелинейную – по вертикали. Структурно оно делится на три последовательные равные части, размером по 2048 байт каждая, как показано на рисунке 5:

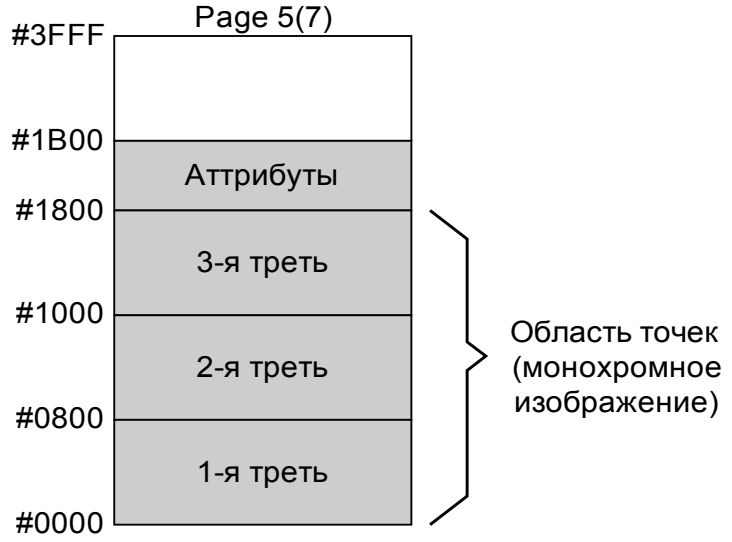

Рис.5

Суть такого деления в том, что нелинейность проявляется именно внутри этих третей – каждая из них разделена на 8 последовательно идущих групп строк, по 8 строк в каждой. И, если последовательно заполнять экранную область памяти каким-либо значением, то сначала окрасятся последовательно друг за другом первые строки всех восьми групп, потом вторые и.т.д. вплоть до вывода изображения на все восьмые строчки всех восьми групп строк. После чего все начнется сначала, но уже во второй трети, а после – в третьей. И только затем все монохромное изображение окрасится различными цветами.

Такая «чересполосица» получается из-за того, что видеоадресация относительно адресации ОЗУ перепутана – в ней адресные линии 5,6,7 и 8,9,10 поменяны местами. Наглядно это можно представить так (**An** – адреса ОЗУ, **Sn** - видеоадреса):

```
                              vvvvvvvvvvvvvvvv   vvvvvvvvvvvvvvvv
A0   A1   A2   A3   A4   A5   A6   A7   A8   A9   A10  A11  A12
S0   S1   S2   S3   S4   S8   S9   S10  S5   S6   S7   S11  S12
                              ^^^^^^^^^^^^^^^^   ^^^^^^^^^^^^^^^^
```

В результате первые 32 байта (так как **A0-A4 = S0-S4**) располагаются последовательно (1 строка = 32 байта = 256 пикселей по 8 пикселей на байт). Но в результате того, что S5-S10, адресующие 64 строки, перепутаны относительно адресов ОЗУ, то строки внутри групп из каждых 64 строк (всего 2048 байт) идут не последовательно. Но сами эти группы (а всего их три) идут друг за другом, так как
S11-S12 = A11-A12.

Расположение байта в экранной области можно высчитать по следующей формуле:

X*2048+Y*32+Z*256+N

Здесь:
X – номер трети экрана (0-2)
Y – номер группы строк внутри трети (0-7)
Z – номер строки внутри группы (0-7)
N – номер байта в строке (0-31)

Структура самого монохромного изображения такова:
На один пиксель изображения приходится один бит цвета. Следовательно, каждая точка изображения может принимать только два цветовых оттенка. Один оттенок при BIT=0, другой – при BIT=1. На каждый байт видеопамяти приходится по 8 пикселей. При этом старшие биты будут соответствовать пикселям слева (чем старше бит, тем левее пиксель), а младшие – правым, как показано на примере байта на рисунке 6:

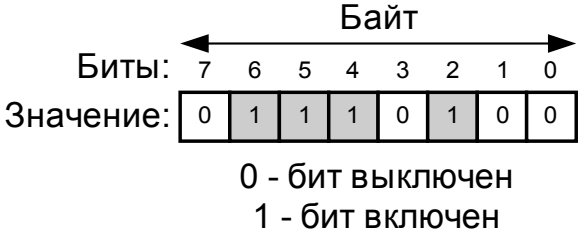

Рис.6

Какие конкретно цвета будут соответствовать включенным и выключенным битам, определяется в области атрибутов.

Область атрибутов, в отличие от области изображения имеет линейную структуру. Она располагается с адреса #1800 по #1AFF и занимает всего 768 (#0300) байт. Каждый байт атрибутов определяет окраску как включенных, так и выключенных пикселей изображения. Но влияет он не на каждый пиксель в отдельности, а на целое знакоместо, являющееся квадратом размером 8x8 пикселей (8 байт). Таким образом, весь экран представляет собой прямоугольник в 32x24 знакомест (всего их - 768). В каждом из знакомест может одновременно быть только два цвета. Один – для выключенных пикселей (т.н. «цвет фона» или PAPER), другой – для включенных (т.н. «цвет тона» или INK).

Структура байта атрибутов:
D0-D2 - цвет INK (тон)
D3-D5 - цвет PAPER (фон)
D6 - задает повышенную яркость (BRIGHT) всему знакоместу (D6=1 – BRIGHT ON).
D7 - включает в пределах знакоместа мерцание (FLASH) – попеременно (с частотой примерно 1 Гц) путем смены местами INK и PAPER(**D7=1 – FLASH ON**).

Таким образом, INK или PAPER может быть раскрашен в один из восьми цветов, которые, к тому же, могут иметь (INK и PAPER одновременно) две градации яркости. Итого – 16 цветовых позиций. В ATM-2+ эти позиции могут быть любой окраски, выбираемой из специальной палитры. Но в ZX-режиме эти цвета программируются в соответствии с окраской на стандартных Спектрумах:

	BRIGHT 0	**BRIGHT 1**
Цвет 0	черный	черный (в некоторых ZX – темно-серый)
Цвет 1	синий	ярко-синий
Цвет 2	красный	ярко-красный
Цвет 3	сиреневый	ярко-сиреневый
Цвет 4	зеленый	ярко-зеленый
Цвет 5	голубой	ярко-голубой
Цвет 6	желтый	ярко-желтый
Цвет 7	белый	ярко-белый

Цвета при **BRIGHT 1** можно рассматривать как позиции 8-15 данной таблицы.

5.2. Текстовая консоль 80x25 (RG0=0, RG1=1, RG2=1)

Расположение:

Страницы **#05** и **#01** ОЗУ при D3=0 порта #7FFD или страница **#07** и **#03** ОЗУ при D3=1 порта #7FFD.
Пропорции пикселей 1:0.5. Размер экрана 4000 байт (разделен на 4 независимые группы).

Структура:

Структура похожа на ZX экран делением на поле монохромного изображения и поле атрибутов. Основное различие заключается в том, что в этом видеорежиме нельзя работать с отдельными пикселями. Работа идет с отдельными знакоместами (8x8 пикселей). При этом один байт из видеоОЗУ определяет содержание конкретного знакоместа: в него аппаратными средствами из знакогенератора, прошитого в специальное ПЗУ, выводится соответствующее этому байту (по стандарту КОИ-8) изображение символа. Таких знакомест – 2000 (80 столбцов и 25 строк, или 640x200 пикселей).

Столбцы символов делятся на группы четных и нечетных, которые располагаются в различных местах видеоОЗУ. Строки четных и нечетных знакомест имеют длину в 64 байта, однако из них отображаются только первые 40. Аналогично расположены и байты атрибутов. Расположение строк – линейное.

Расположение знакомест и атрибутов экрана в ОЗУ представлено на рисунке 7:

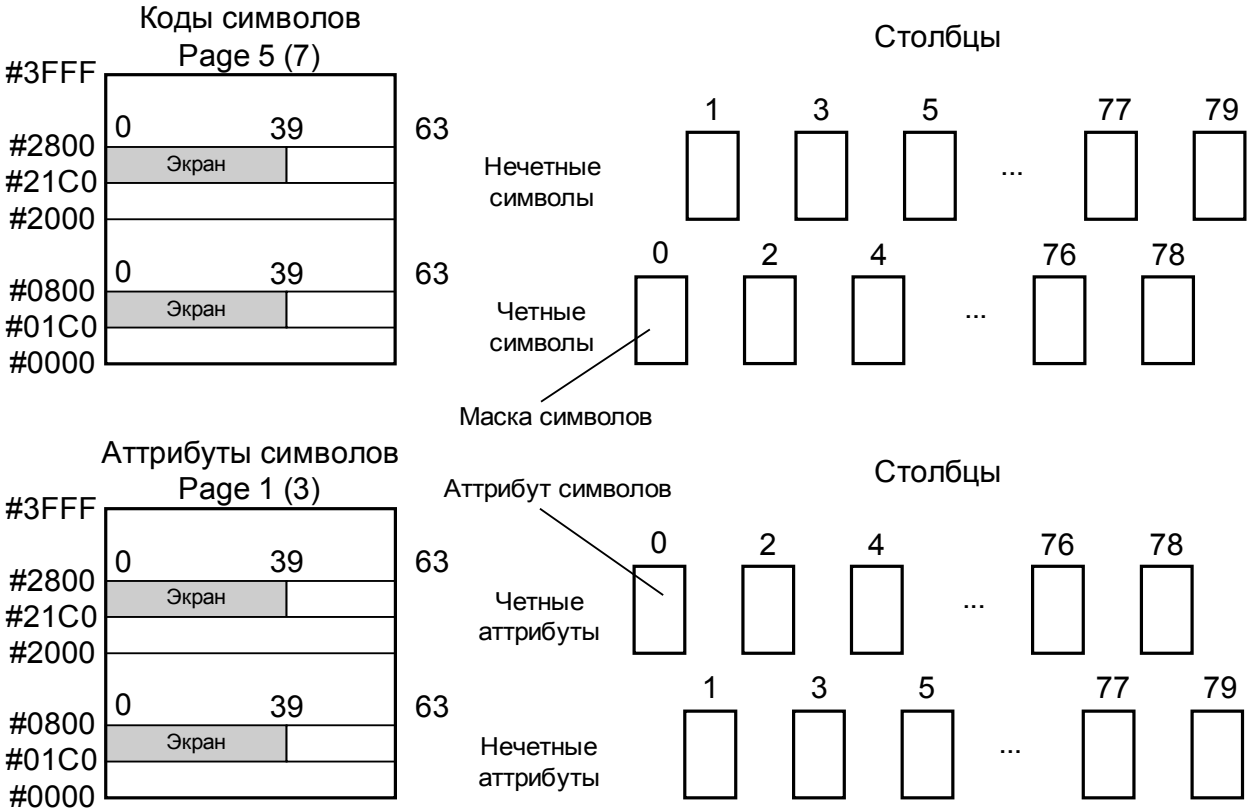

Рис.7

Структура страницы 5(7):
#01C0 - #07FF - Знакоместа четных символов (1000 байт)
#21C0 - #27FF - Знакоместа нечетных символов (1000 байт)

Структура страницы 1(3):
#01C0 - #07FF - Атрибуты **нечетных(!)** символов (1000 байт)
#21C0 - #27FF - Атрибуты **четных(!)** символов (1000 байт)

(!) – области четных/нечетных атрибутов противоположны аналогичным областям символов!

Структура байта атрибутов:
D0,D1,D2,D6 - цвет INK (окраска символа)
D3,D4,D5,D7 - цвет PAPER (фон за символом)

Видно, что байт атрибутов в режиме 80x25 схож с аналогичным в ZX-режиме, за исключением того, что биты **D6** и **D7** используются для задания яркости (точнее – выбор цветов 8-15) отдельно для INK и PAPER, а режима FLASH не существует.

5.3. Режим аппаратного мультиколора 640x200 (RG0=0, RG1=1, RG2=0)

Расположение:

Страницы **#05** и **#01** ОЗУ при **D3=0** порта **#7FFD** или страница **#07** и **#03** ОЗУ при **D3=1** порта **#7FFD**. Пропорции пикселей 1:0.5. Размер экрана 32000 байт (разделен на 4 независимые группы).

Структура:

Структура этого экрана (640x200 пикселей или 80 столбцов (по 8 пикселей каждый) на 200 строк) представляет собой нечто среднее между консолью и ZX-режимом. С одной стороны, он имеет такие же пропорции и размер пикселей, как и в консоли, но только здесь доступны сами пиксели в отдельности, а не целые символы. С другой стороны, существует аналогичное ZX-режиму деление на монохромное изображение и поле атрибутов, но возможности применения атрибутов здесь расширены (см.ниже).

Столбцы монохромных байтов делятся на группы четных и нечетных, которые располагаются в различных местах видеоОЗУ. Строки четных и нечетных знакомест имеют длину в 40 байтов каждая. Аналогично расположены и байты атрибутов. Расположение строк – линейное.

На рисунке 8 показано расположение монохромного изображения и атрибутов экрана в страницах ОЗУ:

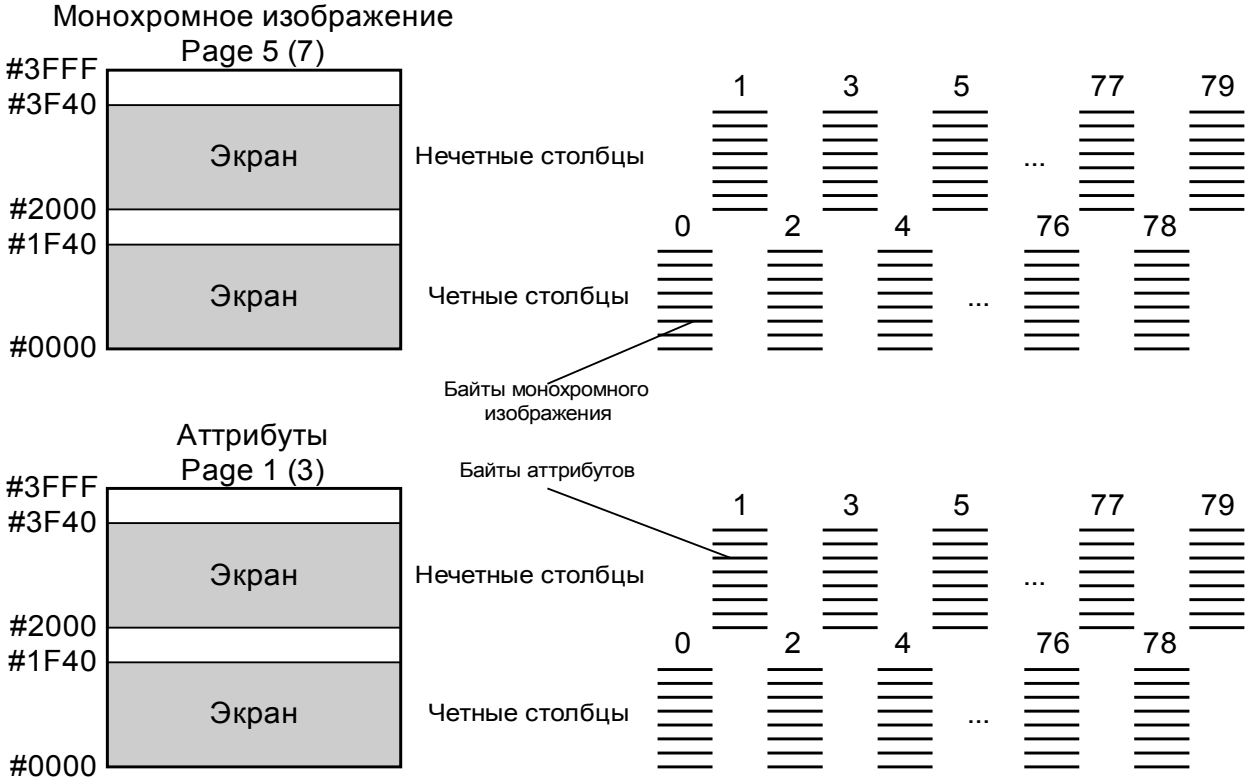

Рис.8

Структура байта монохромного изображения полностью аналогична таковому в ZX-режиме, где, чем старше бит, тем левее пиксель, которому он соответствует (см. рис.6).

Структура страницы 5(7):
#0000 - #1F3F - Знакоместа четных символов (8000 байт)
#2000 - #3F3F - Знакоместа нечетных символов (8000 байт)

Структура страницы 1(3):
#0000 - #1F3F - Атрибуты четных символов (8000 байт)
#2000 - #3F3F - Атрибуты нечетных символов (8000 байт)

Структура байта атрибутов:
Байт атрибутов по своей структуре полностью совпадает с аналогичным в режиме консоли 80x25:

D0,D1,D2,D6 - цвет INK (тон)
D3,D4,D5,D7 - цвет PAPER (фон)

Однако есть очень важное и существенное отличие в отображении цвета. В данном видеорежиме **байт атрибута приходится** не на одно знакоместо (квадрат 8x8 пикселей), как в ZX-режиме и консоли, а **на один «знакоряд»** - горизонтальную полоску из 8 пикселей. Другими словами, один байт атрибута приходится на один байт изображения, что аппаратно повторяет эффект **мультиколора** в ZX-режиме, достигаемого программным путем с затратой значительного количества процессорного времени и позволяет выводить в 8 раз более насыщенные цветом изображения, чем в ZX-режиме 256x192.

5.4. EGA-режим 320x200(x16) (RG0=0, RG1=0, RG2=0)

Расположение:

Страницы **#05** и **#01** ОЗУ при **D3=0** порта **#7FFD** или страница **#07** и **#03** ОЗУ при **D3=1** порта **#7FFD**. Пропорции пикселей 1:1. Размер экрана 32000 байт (разделен на 4 независимые группы).

Структура:

Этот экран (320 пикселей (или 160 байт) по горизонтали и 200 пикселей по вертикали) отличается от всех остальных тем, что здесь нет понятия атрибутов и монохромного изображения. Здесь каждый пиксель изображения может быть раскрашен любым из 16 независимых цветов, что является аналогом одного из типов экрана EGA-режима на IBM PC 286. При этом на каждый байт экранной области ОЗУ приходится по два пикселя (далее – «пиксельные пары» или просто «пары»).

Данные пары, также как и байты в других расширенных экранах, располагаются в разных частях видеопамяти столбцами. Каждая из четырех таких частей содержит в себе по 40 пар в строке.

Расположение экрана в видеопамяти и его структура представлены на рисунке 9:

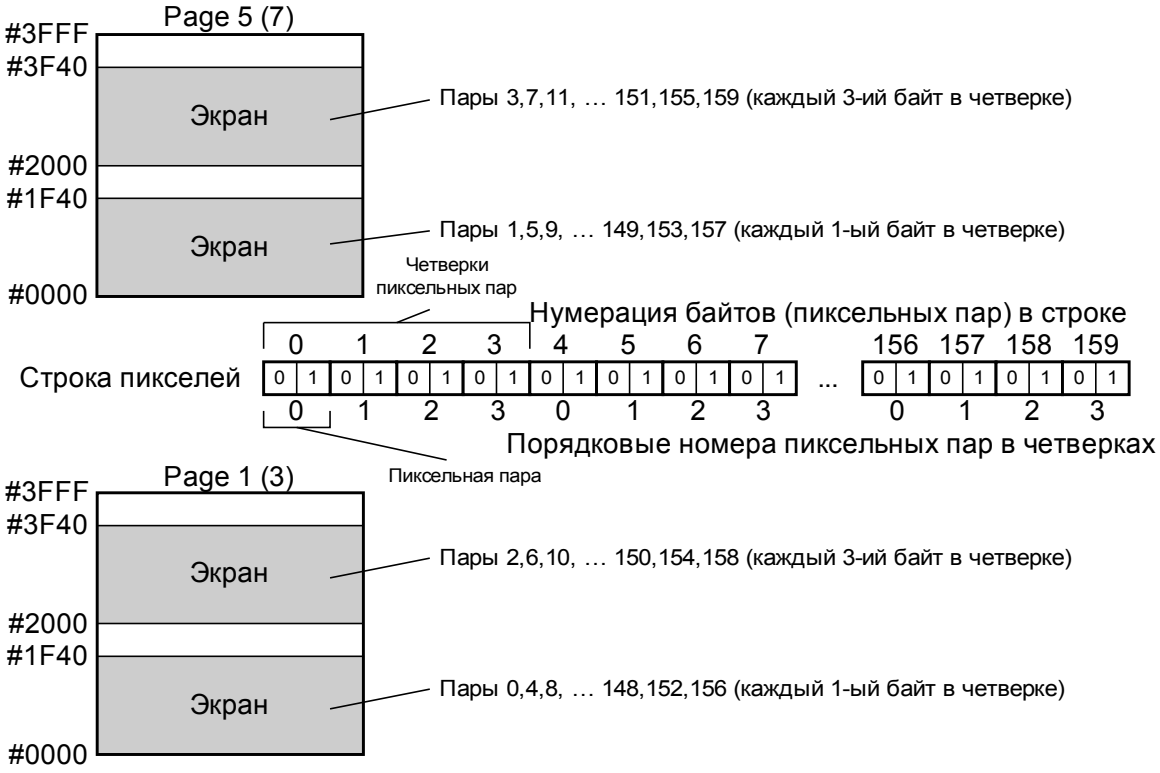

Рис.9

Структура страницы 5(7):
#0000 - #1F3F - Пары 1,5,9, …149,153,157 – каждый 1-й байт в четверке (8000 байт)
#2000 - #3F3F - Пары 3,7,11, …151,155,159 – каждый 3-й байт в четверке (8000 байт)

Структура страницы 1(3):
#0000 - #1F3F - Пары 0,4,8, …148,152,156 – каждый 0-й байт в четверке (8000 байт)
#2000 - #3F3F - Пары 2,6,10, …150,154,158 – каждый 2-й байт в четверке (8000 байт)

Структура пиксельной пары представлена на рисунке 10:

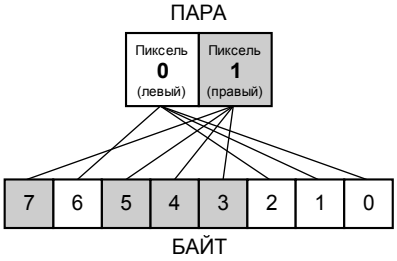

Рис.10

Другими словами,
D0,D1,D2,D6 - соответствуют левому пикселю каждой пары, а
D3,D4,D5,D7 - соответствуют правому пикселю каждой пары.
Хорошо просматривается, что по структуре пиксельная пара – это преобразованный байт атрибутов.

5.5. Порт бордюра/звука (открытый)

out #nnFE = %nnnnnnnn1111x110 (A0=0, A1=A2=1)

Практически полностью стандартный порт ZX-48/128, за исключением небольшой детали:

Значения шины данных:

D0-D2 – сигналы **BRD0 – BRD2**, определяют 8 цветов бордюра (цвета 0 - 7)
D3 - запись сигнала на магнитофон
D4 - управление звуковым каналом (Beeper)
D5-D7 - не используются.

Значения шины адреса:

A3 – инверсный сигнал (относительно **D0-D2**) **BRD3**, который, совместно с сигналами **BRD-BRD2**, позволяет окрашивать бордюр не восемью цветами, как в обычных ZX-48/128, а всеми 16.
Здесь: **A3=1 - BRD3=0**: Стандартные 8 цветов бордюра (цвета 0 - 7)
A3=0 - BRD3=1: Дополнительные цвета бордюра (цвета 8 - 15)

Получаемые значения портов:

a) ***#FE – работа с основным набором цветов бордюра***
b) ***#F6 – работа с дополнительным набором цветов бордюра***

5.6. Порт палитры/выборки FDD (теневой)

out #nnFF = %nnnnnnnn11111111 (A0=A1=A2=1)

Данный порт несет в себе две функции.

Во-первых, является специальным системным портом контроллера FDD. Данная его функция будет рассмотрена в соответствующем разделе. А здесь просто кратко в скобках будут указаны соответствующие сигналы.

Во-вторых, установка палитры. Данная функция будет подробно описана здесь. Эта функция становится доступна только при активизации через порт **#xx77** сигнала **PEN2**. В противном случае порт выполняет исключительно функцию управления дисководами.

Видеосистема ATM-turbo 2+ позволяет выводить на экран одновременно (программные способы смешивания цветов не в счет) только 16 цветов. Но сами эти цвета могут быть совершенно различными. В соответствии каждой из этих 16-цветовых позиций может быть поставлен любой цвет из специальной 64-цветовой палитры.

Значения шины данных:

Палитра:	**Beta-disk Interface:**
D0 – B (Blue)	(DSEL 0/1)
D1 – R (Red)	не используется
D2 – **1** - не используется	(FDC reset)
D3 – **1** - не используется	(FDC halt)
D4 – G (Green)	(Side 0/1)
D5 – b (Low Blue)	не используется
D6 – r (Low Red)	не используется
D7 – g (Low Green)	не используется

Здесь:
b, r, g - цвета с пониженной яркостью, раза в два тусклее цвета в режиме **BRIGHT 0**. **B, R, G** - цвета обычной яркости – соответствуют **BRIGHT 0** в ZX-бейсике. При включении одновременно битов **B, R, G** и **b, r, g** получается цвет повышенной яркости, соответствующий **BRIGHT 1** в ZX-бейсике.

Таким образом, каждый цвет из RGB имеет 4 градации яркости - от **черного** до **BRIGHT 1**. Путем комбинации этих шести битов и получаем 64 самых разнообразных оттенка.

Все данные **инверсны**, т.е. например **Dn=0** - цвет включен, **Dn=1** - выключен. Номер цветовой позиции (0 - 15), в которой заменяется палитра, определяется следующим образом:

Какой цвет из 16 в данный момент выводится на монитор (то есть установлен на шине данных контроллера графики), тот и подлежит замене. То есть, это или текущий цвет бордюра (если луч монитора сейчас находится в его области), или текущий выводимый атрибут (или INK, или PAPER - в зависимости от того, был ли установлен бит пикселя или нет).

Удобнее всего менять цвет именно в соответствии с цветом бордюра, так как в противном случае надо точно знать, какой номер цвета установлен на шине, что при сложных и часто меняющихся изображениях бывает затруднительно. Также необходимо точно рассчитать, когда луч монитора выйдет из области бордюра и войдет в область основного экрана. А узнать момент, когда луч монитора однозначно находится в области бордюра, можно по приходу сигнала прерывания. После него некоторое время (пока выводится бордюр) он находится именно там.

Пример (предполагается, что уже **PEN2=1** и теневые порты открыты):

```
EI                      ; разрешаем прерывания
HALT                    ; ожидаем сигнал INT
LD    A, #03
OUT   (#FE), A          ; устанавливаем цветовую позицию 3
LD    A, %11111000
OUT   (#FF), A          ; «раскрашиваем» ее в белый цвет (BRIGHT 0)
LD    A,#05
OUT   (#F6), A          ; устанавливаем цветовую позицию 13 (8+5)
LD    A, %00110000
OUT   (#FF), A          ; «раскрашиваем» ее в ярко-белый (BRIGHT 1)
```

5.7. Порт атрибутов (открытый)

in #nnFF = %nnnnnnnn11111111 (A0=A1=A2=1)

Данный порт полностью соответствует аналогичному порту в фирменных Спектрумах. В прочих же отечественных клонах он далеко не везде был реализован. Его функция – чтение текущего атрибута, выводимого на экран.

Значения шины данных:

D0-D7 - значение текущего атрибута, полностью аналогичное по структуре байту атрибутов в ZX-экране.

II. Внешние устройства

ПК ATM-turbo 2+ обладает периферией, являющейся одной из самых развитых среди прочих клонов ZX-SPECTRUM. Главной ее особенностью является то, что, она интегрирована в основную материнскую плату компьютера. Ниже приводится полное описание ее узлов:

1. Звуковая подсистема ATM-turbo 2+

Обработка звуковых сигналов в ATM-2+ возможна посредством нескольких устройств:
А) Порт магнитофона
Б) Музыкальный сопроцессор
В) ЦАП/АЦП

1.1. Порт бордюра/звука (открытый)

out #nnFE = %nnnnnnnn1111x110
(A0=0, A1=A2=1)

Этот порт уже описывался ранее. Он является практически полностью стандартным портом ZX-48/128, за исключением использования дополнительной адресной линии **A3** для увеличения возможностей управления цветом бордюра (см. выше).

Значения шины данных:

D0-D2 – сигналы **BRD0 – BRD2**, определяют 8 цветов бордюра (цвета 0 - 7)
D3 - запись сигнала на магнитофон
D4 - управление звуковым каналом (Beeper)
D5-D7 - не используются.

Здесь путем установки или сброса бита **D4** посылается сигнал на динамик. Если установку/сброс сигнала делать с большой частотой, то можно получить на динамике звук соответствующей высоты. Звук генерируется переключением двух уровней напряжения, так как управляет им всего один бит, поэтому громкостью управлять затруднительно, но можно (используя широтно-импульсную модуляцию).

Бит **D3** функционирует аналогично, с той лишь разницей, что он более мощный и используется не для вывода звука на динамик, а для сохранения звукового сигнала на магнитофонной ленте, при использовании ее в качестве внешнего носителя данных.

Значения шины адреса:

A3 – инверсный сигнал (относительно **D0-D2**) **BRD3**, который, совместно с сигналами **BRD-BRD2**, позволяет окрашивать бордюр не восемью цветами, как в обычных ZX-48/128, а всеми 16.

Получаемые значения портов:

c) ***#FE – работа с основным набором цветов бордюра***
d) ***#F6 – работа с дополнительным набором цветов бордюра***

1.2. Порт механической клавиатуры/звука (открытый)
in #xxFE = %xxxxxxxx11111110 (A0=0, A1=A2=1)

Этот порт используется, прежде всего, для опроса стандартной механической 40-клавишной ZX-клавиатуры, а также для считывание сигнала с магнитофона. Он является полностью стандартным портом для любого ZX, за исключением небольшой детали.

Значения шины данных:

D0 – D4 – чтение одной из пяти клавиш одного из восьми полурядов клавиатуры (см. ниже). Данные биты инверсны. Т.е. сброшенный бит означает нажатую клавишу. В противном случае этот бит должен быть установлен.

D5 – специальный, специфический для ATM1,2,2+ системный **сигнал Z**. Через строго определенное время после прихода сигнала INT на небольшой промежуток времени становится **равен 0**, после чего вновь до следующего INT принимает **значение 1**. Ранее использовался для построения защиты ПЗУ от копирования (количество тактов от INT до обнуления являлось ключом для декодирования). Однако в последних версиях ПЗУ от МикроАРТа уже не использовался. Так что фактически на сегодня этот сигнал является не нужным.

D6 – чтение сигнала с магнитофона.

D7 – не используется (при чтении всегда **равен 1**, но нельзя использовать в программах факт его равенства единице).

Значения шины адреса:

A8-A15 - выбор восьми полурядов клавиатуры, путем сброса соответствующего бита в ноль. Соответствие сброшенных битов полурядам наглядно видно на примере данной таблицы:

		D0	D1	D2	D3	D4	D4	D3	D2	D1	D0		
#F7FE(A11=0)	%**11110111**11111110	1	2	3	4	5	6	7	8	9	0	%**11101111**11111110	**#EFFE(A12=0)**
#FBFE(A10=0)	%**11111011**11111110	Q	W	E	R	T	Y	U	I	O	P	%**11011111**11111110	**#DFFE(A13=0)**
#FDFE(A9=0)	%**11111101**11111110	A	S	D	F	G	H	J	K	L	Ent	%**10111111**11111110	**#BFFE(A14=0)**
#FEFE(A8=0)	%**11111110**11111110	CS	Z	X	C	V	B	N	M	SS	Sp	%**01111111**11111110	**#7FFE(A15=0)**

Сброс более одного из битов **A8–A15** позволяет одновременно опрашивать сразу несколько соответствующих полурядов. Однако в таком случае невозможно точно определить, какая именно из клавиш была нажата. Можно лишь определить, что в данном подмножестве клавиш нажата одна (или несколько) из них. Тем не менее, такой метод опроса клавиатуры удобен в случаях, когда важен сам факт нажатия клавиши, без необходимости уточнения.

В случае, если на плате используется также и контроллер AT/XT-клавиатуры на основе м/с i8031/51(1816ВЕ31), то он накладывает ограничения на использование некоторых комбинаций сброшенных битов A8–A15. Подробнее это описано в соответствующем разделе.

1.3. Порты 3-канального музыкального сопроцессора AY-3-8910(AY-3-8912)/YM-2149 (все открытые)

Во всех основных моделях ZX-SPECTRUM 128K вот уже много лет используется трехканальный (каналы A,B,C) музыкальный сопроцессор AY-3-8910 или его аналоги AY-3-8912 и YM-2149. Не стал исключением и ATM-2+. Полную информацию по программированию этого устройства читайте в соответствующей литературе по ZX-SPECTRUM. А ниже дана краткая характеристика портов музыкального сопроцессора и его регистров.

1.3.1. Порт выбора регистров
out #FFFD = %1111111111111101
(A1=0, A14=A15=1, A9=1*)

***(** ошибка платы – в дешифрации не нужен и даже вреден! В версии от NedoPC удаляется с 20.06.2005**)**

Значения шины данных:

D0-D3 – выбор одного 16 регистров (номера R0-R15) для вывода в них данных.
D4-D7 – не используются.

Данный порт выбирает для работы один из 16 регистров сопроцессора. С этого момента данный регистр становится активным, и все операции ввода-вывода в сопроцессоре идут именно с ним до того, как через этот порт не будет выбран какой-то другой регистр.

1.3.2. Порт чтения данных
in #FFFD = %1111111111111101
(A1=0, A9=A14=A15=1)

Значения шины данных:

D0-D7 – данные из текущего регистра сопроцессора. Читается текущее число, содержащееся в активном на данный момент регистре.

1.3.3. Порт записи данных
out #BFFD = %1011111111111101
(A1=A14=0, A15=1, A9=1*)

***(** ошибка платы – в дешифрации не нужен и даже вреден! В версии от NedoPC удаляется с 20.06.2005**)**

Значения шины данных:

D0-D7 – данные, выводимые в активный на данный момент регистр музыкального сопроцессора.

1.3.4. Краткое описание регистров музыкального сопроцессора

Спаренные регистры R0/R1, R2/R3 и R4/R5: используются для выработки частоты тона соответственно каналов **A**, **B** и **C**. Необходимые 12-разрядные значения образуются из 8 бит младшего по номеру регистра и четырех младших бит старшего по номеру регистра.

Регистр R6: младшие пять разрядов этого регистра задают частоту шума.

Регистр R7: через этот регистр осуществляется управление звуковыми каналами и регистрами ввода/вывода:
D0-D2: используются для управления выводом частоты тона. Установка **D0**, **D1** или **D2** приведет к запрещению вывода частоты тона в каналы **A**, **B** и **C** соответственно.
D3-D5: используются для управления выводом частоты шума. Установка **D3**, **D4** или **D5** приведет к запрещению вывода частоты шума в каналы **A**, **B** и **C** соответственно.
D6-D7: устанавливают режим работы каналов **IRB** и **IRA** соответственно. При сброшенном бите канал работает на ввод, а при установленном – на вывод информации.

Регистры R8, R9 и R10: их **D0-D3** управляют амплитудой, а **D4** - включает огибающую соответственно в каналах **A**, **B**, **C**. Если в канале выключены генераторы тона и шума (они накладываются перемножением, а при выключении дают логическую единицу), на выход канала идёт амплитудный уровень напряжения (или огибающая).

Спаренные регистры R11/R12: образуют шестнадцатиразрядное значение огибающей выходного сигнала. Регистр **R11** несет младший байт, а **R12** – старший.

Регистр R13: его четыре младшие разряда управляют формой и режимом огибающей выходного сигнала:
D0 – затухание;
D1 – чередование;
D2 – нарастание;
D3 – продолжение;

Регистры R14 и R15: используются соответственно для связи с каналами ввода/вывода **IRA** и **IRB**. Содержимое этих регистров можно в любой момент считывать и записывать, на формировании звука это никак не отражается.

1.4. Цифро-Аналоговый Преобразователь (Covox) (открытый)

out #nnFB = %nnnnnnnn11111011 (A2=0, A0=A1=1)

Данное устройство позволяет выводить на внешний разъем ток переменного напряжения, а точнее 256 различных его уровней. В принципе, оно применяется в самых различных целях, но чаще всего - как генератора звука, так как, в отличие от однобитного динамика, ЦАП позволяет выводить не 2, а 256 уровней напряжения, что дает возможность формировать звук, приближенный по качеству к реальным звукам. Также данный порт используется для работы с принтером, о чем написано в соответствующем разделе.

Значения шины данных:
D0-D7 – данные ЦАП.

1.5. Восьмиканальный Аналого-Цифровой Преобразователь

Данное устройство является обратным по отношению к COVOX и предназначено для считывания аналогового сигнала извне и перевода его, в зависимости от уровня напряжения, в цифровую форму в диапазоне чисел от 0 до 255. Это можно использовать как для оцифровки аналоговых сигналов (например, звука), используя ATM-2+ в качестве подобия осциллографа, или для обслуживания аналоговых датчиков.
АЦП управляется несколькими портами:

1.5.1. Порт флагов (открытый)

in #7FFD = %0111111111111101
(A1=A15=0, A9=1)

Данный порт используется не только для АЦП, но в контроллере IDE. Кроме того, в ранних версиях ATM-2 некоторые сигналы использовались контроллером XT-клавиатуры, но сейчас надобность в этом отпала. На данный момент порт выглядит так:

Значения шины данных:

D0-D5 – не используются (установлены в 1). Но при опросе порта желательно их маскировать.
D6 – сигнал **WIRQ** (прерывание от IDE-устройства). **D6=1** – приход прерывания.
D7 – **ADCS**: сигнал готовности АЦП. **D7=0** – АЦП готово к опросу, иначе – занят.

1.5.2. Порт вывода данных на АЦП: (открытый)

in #7DFD = %0111110111111101
(A1=A9=A15=0)

Порт непосредственного чтения данных с текущего канала (из восьми возможных).

Значения шины данных:

D0-D7 – данные с АЦП.

Чтобы считанные данные были достоверными, чтение должно осуществляться только после прихода сигнала готовности.

Текущий номер канала (**0 - 7**) устанавливается сигналами **BRD0 – BRD2**, путем вывода данных в биты **D0-D2** порта #**FE.** Другими словами, один из восьми стандартных цветов бордюра ZX-SPECTRUM определяет активным один из восьми каналов АЦП.

2. Контроллер XT/AT-клавиатуры и интерфейса RS-232

Начиная с версии 7.00, в компьютере появилась м/с i8031/51 (1816ВЕ31), что кардинально изменило работу порта **#FE** (при его чтении, естественно) и дало возможность подключить IBM XT клавиатуру (или современную AT-клавиатуру - надо только поставить другую прошивку ПЗУ).

Для своей работы контроллер использует тот же порт #FE, что и механическая клавиатура, но его функции в зависимости от режима работы будут совершенно разными.

1) В случае, если **контроллер не подключен** (не вставлена i8031, специальное ПЗУ или контроллер выключен программно через порт **#FF77**), то порт **#FE** исполняет свои стандартные функции, описанные в предыдущих разделах.

2) **При подключенном контроллере** у порта **#FE** появляются новые функции, выполнение которых зависит от режимов работы контроллера, которые также можно программировать через порт **#FE**.

Контроллер клавиатуры функционирует следующим образом:

При попытке прочитать содержимое клавиатуры, когда процессор дает команду **IN A, (#FE)**, взводится специальный триггер и процессор останавливается сигналом WAIT. При этом в микроконтроллере 8031 генерируется прерывание. После необходимых процедур трансляции адресных линий, микроконтроллер выдает на шину данных процессора состояние клавиатуры, сигнал WAIT сбрасывается и процессор продолжает свою работу.

Наличие 8031 и расширенной клавиатуры (101 клавиша) позволило упростить опрос клавиатуры в CP/M до простого чтения кода клавиши. Однако, это потребовало **введения дополнительных команд** для управления 8031. И, поскольку единственно возможный способ передать данные в 8031 - это управление состоянием старшей половины шины адреса Z80 в момент чтения порта **#FE**, то этот способ и используется.

Управляющие последовательности клавиатуры:

_#55_ - указывает, что следующий байт является кодом команды для клавиатуры. В ответ клавиатура возвращает код **#AA** - это можно использовать для проверки наличия 8031.

```
LD    A, #55
IN    A, (#FE)
CP    #AA
JR    NZ, NO_XT
LD    A, COMM ;команда (см. ниже)
IN    A, (#FE)
LD    A, ARG1 ;дополнительные параметры (могут отсутствовать)
IN    A, (#FE)
LD    A, ARG2 ;дополнительные параметры (могут отсутствовать)
IN    A, (#FE)
```

_#00_ - Читает код нажатой клавиши.

```
        XOR   A
        IN    A, (#FE)
```
;в аккумуляторе код клавиши (список кодов приводится ниже)

_#80_ - читает 1-й байт флагов клавиатуры

```
        LD    A, #80
        IN    A, (#FE)
```

_#40_ - читает 2-й байт флагов клавиатуры

```
        LD    A, #40
        IN    A, (#FE)
```

Описание команд управления контроллером (вызываемых по коду **_#55_**) смотрите в **приложении 1**.

Сам контроллер имеет четыре режима работы (0-3):

Режим 0: Spectrum KBD – эмуляция стандартной механической клавиатуры. Здесь порт **in #FE** выглядит **ПОЧТИ** так же, как и при работе с механической клавиатурой:

Значения шины данных:

D0 – D4 – чтение одной из пяти клавиш одного из восьми полурядов клавиатуры. Данные биты инверсны.
D5 – специальный, специфический для ATM1,2,2+ системный **сигнал Z**.
D6 – чтение сигнала с магнитофона.
D7 – не используется (при чтении всегда **равен 1**).

Значения шины адреса:

A8-A15 - выбор восьми полурядов клавиатуры, путем сброса соответствующего бита в ноль, **или отправка управляющих кодов #00**, **#55**, **#40**, **#80** (и после них – дополнительных команд) в контроллер XT/AT-клавиатуры.

Во **ВСЕХ** остальных трех режимах порт **in #FE** выглядит следующим образом:

Значения шины данных:

D0 – D7 – данные, получаемые от контроллера (коды клавиш, флаги, сигналы с RS-232 и т.д.)

Значения шины адреса:

A8-A15 – передача данных в контроллер (управляющие коды **#00**, **#55**, **#40**, **#80**, конкретные команды, данные для RS-232 и т.д.)

Опишем эти режимы работы:

Режим 1: RD code KBD – чтение из буфера ASCII-кода последней нажатой клавиши без учета флагов и регистров (Ctrl, Shift, Alt и т.д.).

Режим 2: CP/M KBD – похож на режим 1, но, в зависимости от значения шины адреса, читает либо код клавиши из буфера, либо данные из флаговых регистров о нажатии управляющих клавиш (Ctrl, Shift, Alt и т.д.).

Режим 3: Direct RD – прямое чтение SCAN-кода последней нажатой клавиши (**по стандарту XT-клавиатуры!**)

Среди встроенных функций контроллера XT/AT-клавиатуры являются также –

a) Последовательный интерфейс RS-232
b) Часы и календарь.

Они опрашиваются и программируются также через подмножество команд контроллера, полный список которых смотрите в **приложении 1**.

3. Параллельный интерфейс Centronix (порт принтера) (открытый)

Данный интерфейс является упрощенным (за счет исключения некоторых управляющих сигналов) стандартного LPT-порта на PC. Поэтому число устройств, подключаемых к этому порту, ограничено, фактически, только принтерами. Интерфейс управляется путем записи и чтения порта **#FB:**

out #nnxB = %nnnnnnnnx1111011
(A2=0, A0=A1=1)

Порт вывода данных на внешнее устройство.

Значения шины данных:

D0-D7 – данные, выводимые на печатающее устройство.

Значения шины адреса:

A7=0 – стробирование данных.

Получаем подмножество портов:
a) ***#FB – вывод данных без строба***
b) ***#7B – вывод данных со стробированием***

Порт **#FB/#7B** одновременно используется также и для вывода данных на ЦАП/COVOX. Данная его функция описана выше в соответствующем разделе.

in #nnFB = %nnnnnnnn11111011
(A2=0, A0=A1=1)

Чтение состояния внешнего устройства.

Значения шины данных:

D0-D6 – не используются. Всегда установлены в 1.
D7 – сигнал занятости/готовности внешнего устройства:
D7=1 – принтер не готов к приему данных (переполнение буфера, нет бумаги, или просто отключен).
D7=0 – принтер готов к приему данных.

Таким образом, работа с печатающим устройством выглядит примерно так:

```
        LD    A, DATA     ;данные для печати
        PUSH  AF
NORDY   IN    A, (#FB)    ;читаем состояние печатающего устройства
        AND   %10000000   ;маска на незадействованные биты
        CP    #00         ;проверка на готовность
        JR    NZ, NORDY   ;уход в цикл ожидания готовности
        POP   AF
        OUT   (#FB), A    ;вывод данных в порт принтера
        OUT   (#7B), A    ;стробирование данных
        OUT   (#FB), A    ;подтверждение вывода
        RET
```

4. Порт внешних устройств
(открытый)

in/out #nnFA = %nnnnnnnn11111010
(A0=A2=0, A1=1)

Значения шины данных:

D0-D7 – вывод/ввод данных через разъем внешних устройств.

Данный порт и упомянутый разъем представляют собой нечто похожее на урезанную системную шину. На этот разъем выведены восемь бит с порта **#FA**, а также восемь бит данных с Centronix (**#FB**) и управляющие сигналы чтения/записи и сброса. При этом через порт **#FA** передаются/получаются данные. А значение порта **#FB** представляет собой восьмибитную шину адреса.

Таким образом, используя совместно порты **#FA** и **#FB,** можно через данный разъем адресовать (читать и выводить данные для управления) до 256 различных внешних устройств. Реально данный порт использовался для подключения и управления программатором **UNIPROG** и специальным HAYES-модемом **Z-Contact 1200**. Также этот порт можно задействовать в качестве полного LPT-порта, компенсировав урезанную функциональность собственно порта **#FB**.

5. BETA-Disk 128 интерфейс
(все порты - теневые)

BETA-Disk интерфейс представляет собой программно аппаратный узел управления накопителями на гибких магнитных дисках. Он состоит из:

a) ОС TR-DOS, прошитой в ПЗУ
b) Системы автоматического подключения ПЗУ TR-DOS по переходу к адресам #3Dxx и по NMI (кнопка «MAGIC»).
c) Порты взаимодействия с микросхемой контроллера дисководов 1818ВГ93.

Рассмотрение первого пункта (ОС TR-DOS) выходит за рамки данной книги.
Функционирование автоматического переключения ПЗУ (пункт второй) подробно рассмотрен в начале книги в описании архитектуры.
Ниже представлено краткая характеристика портов взаимодействия с дисководами и микросхемой 1818ВГ93.

Данное описание не предполагает освещения вопросов программирования микросхемы и работы с дисководами. За данной информацией следует обратиться к специальной литературе.

5.1. Системный регистр (запись)
out #nnFF = %nnnnnnnn11111111
(A0=A1=A2=A3=A4=A7=1)

Значения шины данных:

D0 – выбор одного из двух активных дисководов (**0 – A**, **1 – B**).
D1 – не используется
D2 – аппаратный сброс микроконтроллера 1818ВГ93
D3 – если записать туда единицу, то вертушка дисковода, заведённая какой-либо командой, через некоторое время плавно остановится**.**
D4 – выбор магнитной головки (стороны диска). **0 –** нижняя сторона, **1 –** верхняя.
D5–D7 – не используются.

Данный порт при определенных условиях также одновременно используется для формирования цветовой палитры компьютера. Об этой функции читайте выше в соответствующем разделе.

5.2. Системный регистр (чтение)
in #nnFF = %nnnnnnnn11111111
(A0=A1=A2=A3=A4=A7=1)

Значения шины данных:

D0-D5 – не используются (равны **1**)
D6 – сигнал **DRQ/** - запрос данных микроконтроллером (**D6=0** – есть запрос).
D7 – сигнал **INTRQ/** - сигнал окончания выполнения команды (**D7=0** – есть окончание).

5.3. Регистр команд

out #nn1F = %nnnnnnnn00011111
(A5=A6=A7=0, A0=A1=A2=A3=A4=1)

Значения шины данных:

D0-D7 – команды микроконтроллера (описание команд смотрите в соответствующей литературе).

5.4. Регистр состояний

in #nn1F = %nnnnnnnn00011111
(A5=A6=A7=0, A0=A1=A2=A3=A4=1)

Значения шины данных:

D0-D7 – данные микроконтроллера о выполнении команд и состоянии дисководов.

5.5. Регистр дорожки

in/out #nn3F = %nnnnnnnn00111111
(A6=A7=0, A0=A1=A2=A3=A4=A5=1)

Значения шины данных:

D0-D7 – чтение/запись номера физической дорожки, на которой находится головка дисковода. Контроллер сравнивает с ним номер дорожки, считываемый из заголовков секторов.

5.6. Регистр сектора

in/out #nn5F = %nnnnnnnn01011111
(A5=A7=0, A0=A1=A2=A3=A4=A6=1)

Значения шины данных:

D0-D7 – чтение/запись логического номера сектора, с которым будет производиться операция.

5.7. Регистр данных

in/out #nn7F = %nnnnnnnn01111111
(A7=0, A0=A1=A2=A3=A4=A5=A6=1)

Значения шины данных:

D0-D7 – чтение/запись информации на гибких носителях, а также передача вспомогательных параметров некоторых команд.

6. IDE-интерфейс

Встроенный в материнскую плату IDE-интерфейс позволяет подключать любые два IDE-совместимых устройства самой разнообразной емкости. Однако, на данный момент, существует программная поддержка только жестких дисков (HDD). Управление интерфейсом осуществляется следующими портами:

6.1 Порт флагов (открытый)

in #7FFD = %0111111111111101 (A1=A15=0, A9=1)

Данный порт уже упоминался в разделе, посвященном АЦП, так как позволяет считать два сигнала – сигнал готовности АЦП (**ADCS**), который описывался в соответствующем разделе, и прерывание от IDE-устройства (**WIRQ**). Разберем этот сигнал, а также сам порт поподробнее еще раз:

Значения шины данных:

D0-D5 – не используются (установлены в 1). Но при опросе порта желательно их маскировать.
D6 – сигнал **WIRQ** (прерывание от IDE-устройства) с IDE-разъема. **D6=1** – приход прерывания. Означает, что IDE-устройство выполнило текущую операцию и можно прерваться и выйти из цикла ожидания. **D6=0** – означает, что IDE-устройство занято выполнением операции и не готово к продолжению работы, а значит следует подождать.
Этот сигнал использовался в основном в старых винчестерах и в старых версиях IDE-интерфейса. А сейчас во многих других IDE-интерфейсах для ZX он даже не задействован, так что использовать его, особенно при работе с современными винчестерами, не обязательно.
D7 – **ADCS**: сигнал готовности АЦП. **D7=0** – АЦП готово к опросу, иначе – занят.

6.2. Порты обмена данными (теневые)

in/out #nxxF = %nnnnnnnxxxx01111 (A4=0, A0=A1=A2=A3=1)

Этот адрес является общим для нескольких портов, занятых управлением IDE-устройством и обменом данными с ним. Опишем основные сигналы:

Значения шины данных:

D0–D7 – данные передаваемые/получаемые для IDE.

Значения шины адреса:

A8 – выборка для операций младших или старших восьми бит 16-битной шины данных IDE-устройства. Как известно, шина данных ZX-SPECTRUM имеет только 8 бит, а

поэтому компьютер имеет возможность работать с шиной IDE только по частям. **A8=0** – работа идет с младшими восемью битами, **A8=1** – работа идет со старшими восемью битами.

A5,A6,A7 – внутренняя трехбитная шина адреса контроллера IDE (соответственно сигналы **HD0,HD1,HD2** на разъеме IDE-интерфейса или сигналы **BA5,BA6,BA7** на схеме ATM-2+). Позволяет выбирать для работы один из восьми регистров (**0 – 7**) IDE-интерфейса. Состояние остальных адресных битов (**A9 – A15**) не имеет значения. На основе комбинаций данных адресов получаем для работы с IDE следующее подмножество портов:

a) ***in/out #nx0F (A5=0, A6=0, A7=0) – порт данных. Это <u>единственный</u> регистр IDE, которому для работы требуются все 16 бит шины данных IDE (а значит и адрес A8 шины адреса компьютера). <u>Все остальные</u> регистры IDE работают исключительно с обычными младшими битами шины данных D0 – D7.***

Отсюда получаем порты:

in/out #000F – чтение/запись младших 8 бит (или битов D0 – D7)
in/out #010F – чтение/запись старших 8 бит (или битов D8 – D15)

Подробнее об особенностях и порядке работы с половинками шины адреса IDE-интерфейса читайте ниже в специальном разделе.

b) ***in/out #n02F (A5=1, A6=0, A7=0) – порт ошибок/предкомпенсации***
in – получение данных об ошибках в выполнении текущих команд.
out – ранее использовался для указания номера цилиндра на HDD, с которого необходима предкомпенсация. В современных жестких дисках не используется.

c) ***in/out #n04F (A5=0, A6=1, A7=0) – порт счетчика секторов. Определяет, сколько секторов подряд надо записать или считать (а также хранит информацию о том, сколько их еще не считано/записано).***

d) ***in/out #n06F (A5=1, A6=1, A7=0) – порт сектора. Содержит начальный номер сектора для любой операции с данными. Номер сектора может быть любой от 1 до числа секторов на дорожке (не бывает более 63).***

e) ***in/out #n08F (A5=0, A6=0, A7=1) – младший регистр цилиндров. Содержит младшие 8 бит начального номера цилиндра для любой дисковой операции. После выполнения команды этот регистр модифицируется и всегда отражает текущий номер цилиндра.***

f) ***in/out #n0AF (A5=1, A6=0, A7=1) – старший регистр цилиндров. Аналогичен предыдущему, но содержит старшие начального номера цилиндра.***

g) ***in/out #n0CF (A5=0, A6=1, A7=1) – регистр накопителя/головки. Содержимое этого регистра задает и хранит номер накопителя (0-1 или Master/Slave) и головки (0-15) при выполнении некоторых команд.***

h) ***in/out #n0EF (A5=1, A6=1, A7=1) – регистр команд/состояния.***
in – содержит набор различных флагов, информирующих о состоянии IDE-устройства.
out – только для записи: содержит код команды, посылаемой IDE-устройству.

ВНИМАНИЕ(!): конкретное содержание вышеперечисленных регистров (особенно флаговых) зависит от того, какое именно IDE-устройство (HDD, CD-ROM или что-то другое) присоединено к разъему. Поэтому подробную информацию о работе с этими регистрами ищите в специальной литературе по программированию каждого из вышеперечисленных устройств в отдельности! Пример драйвера для IDE-HDD смотрите в Приложении 2.

6.3. Работа с 16-битной шиной данных IDE-интерфейса

В этом разделе рассматривается принцип сопряжения восьмибитной шины данных ATM-2+ с шестнадцатибитной шиной данных IDE интерфейса. Так как разрядности компьютера и IDE различны, то ATM-2+ вынужден работать с данными от IDE по частям, через два различных порта.

1) Порт **in/out #FE0F** (точнее подмножество портов, определяемое адресами **A5-A7**) – **A8=0**. Он выполняет две функции. Во-первых, передает на шину данных IDE, или получает от нее младшие 8 бит (**D0–D7**), а во-вторых, активизирует винчестер, то есть осуществляет непосредственные чтение/запись очередных 16 бит данных. При чтении старшие 8 бит данных (так как младшие читаются/пишутся напрямую через порт) защелкиваются в специальный регистр, а при записи для старших 8 бит данных берется текущее значение, хранящееся в этом регистре. Для управления этим регистром существует свой порт.

2) Порт **in/out #FF0F** (точнее подмножество портов, определяемое адресами **A5-A7**) – **A8=1**. У него одна функция – чтение из специального регистра или запись в него данных для старших 8 бит (**D8–D15**) шины данных IDE. При этом непосредственного обращения к IDE-устройству не происходит! Обмен данными идет только внутри самого компьютера. Другими словами, данный порт только подготавливает к записи в специальном регистре свою половинку данных, или считывает ее оттуда после того, как данные попали в регистр после обращения к IDE-устройству. Непосредственным же обменом данных (разумеется, предварительно подготовленных) руководит порт **#FE0F**.

С учетом вышесказанного, **примерная схема чтения данных** выглядит следующим образом (здесь и далее по умолчанию считается, что порты уже открыты, нужные команды на чтение и запись отданы. Также не рассматриваются процедуры ожидания готовности и учета ошибок):

```
                LD      HL, BUFFER ;куда будем считывать данные
                . . .
                . . .
RD_DAT          LD      BC, #FE0F  ;выбираем порт для младших 8 бит (адрес A8=0).
                IN      A, (C)     ;считываем в регистр A младшие 8 бит и одновременно
                                   ;защелкиваем в спец.регистр старшие 8 бит.
                LD      (HL), A    ;записываем считанный байт в текущую ячейку памяти
                INC     HL         ;увеличиваем адрес считывания
                INC     B          ;устанавливаем A8=1, адрес порта теперь  #FF0F.
                IN      A, (C)     ;считываем старшие 8 бит в регистр A из спец.буфера
                LD      (HL),A     ;записываем их в память
                INC     HL         ;увеличиваем адрес считывания
```

Таким образом, при помощи процедуры RD_DAT мы считаем 16 бит данных или два байта из текущего сектора. А так как у значительного числа IDE-устройств (и прежде всего у HDD) сектор равен 512 байт, то для полного его считывания необходимо повторить эту процедуру в цикле 256 раз. Эту задачу можно сильно упростить и ускорить благодаря тому, что бит управления половинками данных IDE приходится именно на адрес A8. Это дает возможность воспользоваться для считывания целого сектора командой ввода массива значений из порта – INIR. С ее использованием процедура считывания сектора будет выглядеть примерно так:

```
            LD    HL, BUFFER ;куда будем считывать данные
            . . .
            . . .
RD_SEC      LD    BC, #000F  ;задаем порт (A8=0!) для считывания.
            INIR             ;считываем первые 256 байт
            INIR             ;считываем вторые 256 байт
                             ;(если сектор больше 512 байт, то ряд команд INIR
                             ;можно продолжить)
```

Так как команда INIR в процессе ввода данных из порта с каждым циклом уменьшает на единицу регистр B, то мы получаем попеременно то установку, то сброс A8 в старшей половинке адреса порта **#nn0F**, в результате выполняется описанная выше последовательность – считывание младших 8 бит + защелкивание старших в спец.регистре / считывание старших 8 бит из регистра.

В отличие от чтения, **схема записи данных** на IDE-устройство более сложная. Это происходит потому, что запись также происходит сразу по 16 бит, а значит необходимо сначала поместить старшие 8 бит данных в спец.регистр, и уже только потом осуществлять запись вместе с младшими битами. Такая «обратная» запись делает невозможным использование команды OTIR. С учетом этого, примерная процедура записи на винт (при тех же предварительных условиях, что и в процедуре чтения) выглядит так:

```
            LD    HL, BUFFER ;откуда будем брать данные для записи
            . . .
            . . .
RW_DAT      INC   HL         ;берем из памяти старший байт из выводимой пары
            LD    BC, #010F  ;задаем порт для операции со спец.регистром (A8=1!)
            LD    A, (HL)    ;загружаем старший байт в регистр A
            OUT   (C), A     ;защелкиваем его в спец.регистре
            DEC   HL         ;берем из памяти младший байт из выводимой пары
            DEC   B          ;устанавливаем адрес A8=0. Порт теперь #000F
            LD    A, (HL)    ;загружаем младший байт в регистр A
            OUT   (C), A     ;записываем все 16 бит (из порта и из регистра) на IDE
            INC   HL         ;переходим к следующей паре адресов памяти,
            INC   HL         ;откуда будет происходить считывание
```

Этим методом мы запишем на IDE-устройство два байта текущего сектора. Чтобы записать весь сектор, необходимо повторить подпрограмму RW_DAT 256 раз (или больше, если сектор устройства больше 512 байт). Ее можно, конечно, еще сократить и оптимизировать путем применения команд OUTI/OUTD, однако сделать ее такой же короткой и быстрой, как процедура чтения, к сожалению, не удастся. В результате процесс чтения с винчестера и других IDE-устройств всегда будет быстрее процесса записи.

7. Незадействованные порты:
(теневые)

in/out #FFE7 = %nnnnnnn111100111
(A3=A4=0, A0=A1=A2=A8=1)

in/out #FEE7 = %nnnnnnn011100111
(A3=A4=A8=0, A0=A1=A2=1)

Данные порты в ATM-turbo 2+ (v7.xx) не задействованы ни для одного внешнего устройства, однако их адреса в дешифраторе присутствуют. Они являются рудиментами, оставшимися от ATM-turbo 2 (v6.xx), где они использовались в старом варианте контроллера XT-клавиатуры. Сейчас же при попытке что-то записать в них данные уйдут в никуда, а при чтении из них будет считываться число **#FF** (**D0–D7=1**). Однако вполне возможно, что эти порты будут задействованы в новых HARDWARE-разработках для ATM.

III. Приложения

Приложение 1. Программирование контроллера i8031

Режим работы контроллера определяется двумя младшими битами кода передаваемого в контроллер по команде: **0x55 0x08 <mode>.**

0 - Spectrum KBD
1 - RD Code KBD
2 - Mode CP/M
3 - Direct RD

Остальные биты не влияют на работу контроллера, хотя судя по предварительному описанию старший бит должен был переключать режим RUS/LAT.

mode=0 режим Spectrum KBD

В этом режиме контроллер эмулирует подключение к порту клавиатуры обычной матрицы 5*8 клавиш.

Каждое нажатие клавиши преобразуется в одно или два замыкания в узлах этой матрицы. При сканировании порта клавиатуры на выход выдается код соответствующий запрошенной адресной линии матрицы. Дополнительно к этому коду добавляется код, считанный с "родного" порта клавиатуры, что позволяет считать линию входа магнитофонного интерфейса и сигналы с джойстиков или механической клавиатуры. Добавка идет командой OR.

Соответствие клавиш IBM-клавиатуры и матрицы Spectrum'а задается таблицей:

```
;-------------------------------------
; Scan-code IBM(1) -> code Spectrum
; D7 - Symbol Shift
; D6..D4 - Number bit Adress (A8=000..A15=111)
; D3 - Caps Shift
; D2..D0 - Number bit Data (D0=001..D4=101)

L_4B6:
        db  39h         ;01 ESC             CapSh + 1
;
        db  31h         ;02 1/!
        db  32h         ;03 2/@
        db  33h         ;04 3/#
        db  34h         ;05 4/$
        db  35h         ;06 5/%
;
        db  45h         ;07 6/^
        db  44h         ;08 7/&
        db  43h         ;09 8/*
        db  42h         ;0A 9/(
        db  41h         ;0B 0/)
;
```

```
        db 0E4h     ;0C -/_         SymSh+Kl_J
        db 0E2h     ;0D =/+         SymSh+Kl_L
        db  49h     ;0E BS          CapSh+Kl_0
        db  3Bh     ;0F TAB         CapSh+Kl_3
;
        db  21h     ;10 Q
        db  22h     ;11 W
        db  23h     ;12 E
        db  24h     ;13 R
        db  25h     ;14 T
;
        db  55h     ;15 Y
        db  54h     ;16 U
        db  53h     ;17 I
        db  52h     ;18 O
        db  51h     ;19 P
;
        db 0D5h     ;1A [/{         SymSh+Kl_Y
        db 0D4h     ;1B ]/}         SymSh+Kl_U
        db  61h     ;1C ENTER
        db  88h     ;1D Ctrl        CapSh+SymSh
;
        db  11h     ;1E A
        db  12h     ;1F S
        db  13h     ;20 D
        db  14h     ;21 F
        db  15h     ;22 G
;
        db  65h     ;23 H
        db  64h     ;24 J
        db  63h     ;25 K
        db  62h     ;26 L
;
        db 0D2h     ;27 ;/:         SymSh+Kl_O
        db 0D1h     ;28 '/"         SymSh+Kl_P
        db  91h     ;29 `/~         CapSh+Kl_A
;
        db  08h     ;2A Left Shift  CapSh
        db  92h     ;2B \ / |       CapSh+Kl_S
;
        db  02h     ;2C Z
        db  03h     ;2D X
        db  04h     ;2E C
        db  05h     ;2F V
;
        db  75h     ;30 B
        db  74h     ;31 N
        db  73h     ;32 M
;
        db 0F4h     ;33     ,/<     SymSh+Kl_N
        db 0F3h     ;34     ./>     SymSh+Kl_M
        db  85h     ;35     //?     SymSh+Kl_V
```

```
        db  80h         ;36     Right Shift     SymSh
        db 0F5h         ;37     [*]             SymSh+Kl_B
        db  3Ch         ;38     Alt             CapSh+Kl_4
;
        db  71h         ;39 SPACE
;
        db  3Ah         ;3A CapsLock            CapSh+Kl_2
;
        db 0B1h         ;3B F1                  SymSh+Kl_1
        db 0B2h         ;3C F2                  SymSh+Kl_2
        db 0B3h         ;3D F3                  SymSh+Kl_3
        db 0B4h         ;3E F4                  SymSh+Kl_4
        db 0B5h         ;3F F5                  SymSh+Kl_5
        db 0C5h         ;40 F6                  SymSh+Kl_6
        db 0C4h         ;41 F7                  SymSh+Kl_7
        db 0C3h         ;42 F8                  SymSh+Kl_8
        db 0C2h         ;43 F9                  SymSh+Kl_9
        db 0C1h         ;44 F10                 SymSh+Kl_0
;
        db      0       ;45 NumLock
        db      0       ;46 ScrollLock
;
        db  3Ch         ;47     [7]             CapSh+Kl_4
        db  4Ch         ;48     [8] [Up]        CapSh+Kl_7
        db  3Dh         ;49     [9]             CapSh+Kl_5
        db 0E4h         ;4A     [-]             SymSh+Kl_J
        db  3Dh         ;4B     [4]             CapSh+Kl_5
        db  35h         ;4C     [5]             5
        db  4Bh         ;4D     [6]             CapSh+Kl_8
        db 0E3h         ;4E     [+]             SymSh+Kl_K
        db  4Ah         ;4F     [1]             CapSh+Kl_9
        db  4Dh         ;50     [2]             CapSh+Kl_6
        db  4Bh         ;51     [3]             CapSh+Kl_8
        db  84h         ;52     [Insert]        SymSh+Kl_C
        db  49h         ;53     [Delete]        CapSh+Kl_0
        db 0E5h         ;57     F11             SymSh+Kl_H
        db  94h         ;58     F12             SymSh+Kl_F
```

mode=1 - RD code KBD

В этом режиме, независимо от состояния адресной линии, контроллер возвращает CP/M код последней нажатой клавиши.
При этом регистр контроллера, хранящий этот код, сбрасывается в 0.

mode=2 - CP/M KBD

В этом режиме, в зависимости от состояния двух старших битов адреса сканирования, контроллер возвращает:

A15=0,A14=0 - CP/M код последней нажатой клавиши.
Регистр после чтения сбрасывается.

A15=0,A14=1 - состояние регистра управляющих клавиш;
- **d0** - Shift (1-нажата)
- **d1** - Ctrl (1-нажата)
- **d2** - ALT (1-нажата)
- **d3** - всегда 0
- **d4** - Caps Lock trigger
- **d5** - Num Lock trigger
- **d6** - Scroll Lock trigger
- **d7** - RUS(1)/LAT(0)

A15=1,A14=0 - состояние дополнительного регистра
- **d0** - Right Shift
- **d1..d7** - всегда 0

A15=1,A14=1 - не несет полезной информации

mode=3 - Direct RD

В этом режиме, независимо от состояния адресных линий, контроллер возвращает скан-код последней нажатой клавиши клавиатуры IBM, причем код XT-клавиатуры ***(справедливо для версий прошивки, начиная с v2.1. В версиях v1.x – v2.0 тип SCAN-кода всегда соответствовал типу подключенной клавиатуры – XT или AT, что негативно отражалось на совместимости с некоторым ПО)***.

Управление контроллером:

Для управления контроллером при чтении порта клавиатуры выставляется адрес сканирования, равный **0x55**. При этом контроллер возвращает в качестве ответа код **0xAA**. Это является признаком готовности контроллера к приему кода команды в следующем цикле чтения порта.

Код сканирования клавиатуры в цикле передачи команды в 6 младших битах содержит собственно код команды, а в двух старших битах адресный код команды.
Необходимо отметить, что при приеме незадействованного кода команды контроллер в ответ выдает код = **0xFF**.
В частности, в игре MINER наличие контроллера определяется выполнением команды с кодом = **0x00**. Поскольку этот код не является командой, то контроллер отвечает на **0x55 -> 0xAA**, а на **0x00 -> 0xFF**. Этот факт и является признаком наличия контроллера.

Коды команд:

1) чтение версии прошивки контроллера

0x55 **0x01** - первый байт версии (обычно он и читается)
0x55 **0x41** - второй
0x55 **0x81** - третий
0x55 **0xC1** - четвертый

Эти команды читают содержимое ПЗУ контроллера начиная с адреса **0x2C**. Судя по всему, автор прошивки зарезервировал эти адреса для размещения кодов версии.
В XT-прошивке в этих адресах записаны коды **6,0,1,0**. Текст в конце прошивки при этом указывает **V1.06**.
В AT-прошивке (оригинальной от МикроАРТа) в этих адресах записаны коды 1,0,0,0 Текст в конце прошивки при этом указывает V1.00.
В BIOS ATM проверяется только первый байт. При этом наличие контроллера опознается сравнением этого байта либо с 10, либо с 16 (проверка производится дважды).

2) очистка буфера SPECTRUM клавиатуры

0x55 **0x07|0x47|0x87|0xC7**

Очищается буфер, в котором хранятся скан-коды нажатых клавиш в режиме 0. Нажатие клавиши заносит ее скан-код в буфер, отжатие должно удалить этот скан-код из буфера. Похоже, эта команда введена на всякий случай, поскольку при корректной работе контроллера буфер должен автоматически очищаться при нажатии и отжатии клавиш. Команда возвращает коды **0xAA** и **0xFF**.

3) Установка режима работы контроллера

0x55 **0x08|0x48|0x88|0xC8 <mode>**

В результате выполнения этой команды в переменной контроллера сохраняется код <mode> (**0–3.** Режимы описаны выше). Реально в контроллере используются только 2 младших бита этого кода, определяющие текущий режим работы контроллера.
Команда возвращает коды **0xAA** и **0xFF**.

4) Чтение регистров CP/M кода клавиатуры

0x55 **0x09** - байт предыдущего скан-кода клавиатуры
0x55 **0x49** - байт текущего скан-кода клавиатуры
0x55 **0x89** - байт регистра управляющих клавиш:

d0	-	Shift (1-нажата)
d1	-	Ctrl (1-нажата)
d2	-	ALT (1-нажата)
d3	-	всегда 0
d4	-	Caps Lock trigger
d5	-	Num Lock trigger
d6	-	Scroll Lock trigger
d7	-	RUS(1)/LAT(0)

0x55 **0xC9** - байт дополнительного регистра:

d0 - Right Shift

d1..d7 - всегда 0

По сути, эти команды дублируют чтение регистров контроллера, выполняемое при его работе в режиме 2.

5) Установка режима RUS

0x55 **0x0A|0x4A|0x8A|0xCA**

Эта команда устанавливает в 1 бит **d7** регистра управляющих клавиш контроллера.
Команда возвращает коды **0AAh 0xFF**.

6) Установка режима LAT

0x55 **0x0B|0x4B|0x8B|0xCB**

Эта команда сбрасывает в 0 бит **d7** регистра управляющих клавиш контроллера.
Команда возвращает коды **0AAh 0xFF**.

7) Установка режима ожидания.

0x55 **0x0C|0x4C|0x8C|0xCC**

Команда устанавливает контроллер в режим выдачи сигнала **/WAIT** на процессор Z80. То есть, она эквивалентна нажатию клавиши **PAUSE/BREAK**. Для выхода из этого режима надо нажать эту клавишу.
Команда возвращает коды **0AAh 0xFF**.

8) Выдача сигнала /RESET

0x55 **0x0D|0x4D|0x8D|0xCD**

На Z80 выдается сигнал сброса, что перезагружает всю систему. Эквивалентно нажатию **Ctrl/Alt/Del**

9) Чтение регистров текущего времени

0x55 **0x10** - секунды
0x55 **0x50** - минуты
0x55 **0x90** - часы
0x55 **0xD0** - дни

Читается содержимое регистров часов реального времени, реализованных на основе встроенного таймера микроконтроллера. Таймер в обоих оригинальных прошивках настроен на работу при частоте тактирования контроллера 7 МГц (однако сейчас уже сделаны версии для других частот тактирования). При отключении питания таймер, естественно, сбрасывается.

Примечание: Похоже, что здесь была ошибка с установкой базового адреса регистра текущего времени. Если сдвинуть его на один байт, то первой командой читались бы тики по 20 мсек (от 0 до 49), а дальше секунды, минуты и часы.

10) Установка регистров текущего времени

0x55 **0x11** - <секунды>
0x55 **0x51** - <минуты>
0x55 **0x91** - <часы>
0x55 **0xD1** - <дни>

Эти команды записывают передаваемые данные в регистры часов реального времени.

11) Чтение регистров текущей даты

0x55 **0x12** - день
0x55 **0x52** - месяц
0x55 **0x92** - год
0x55 **0xD2** - столетие

12) Запись регистров текущей даты

0x55 **0x13** - <день>
0x55 **0x53** - <месяц>
0x55 **0x93** - <год>
0x55 **0xD3** - <столетие>

13) Установка в 1 линий порта P1

0x55 **0x14|0x54|0x94|0xD4 <cod>**

Выполняется команда:
orl P1, <cod>

14) Сброс в 0 линий порта P1

0x55 0x15|0x55|0x95|0xD5 <cod>
Выполняется команда:
cpl <cod> ;инверсия кода
anl P1,<cod>

Предполагалось, видимо, что эти команды позволят программно управлять последовательным портом контроллера. Но я считаю их использование нецелесообразным, тем более, что реально можно задействовать только биты **3(DTR)** и **4(RTS).** С остальными битами при таком способе управления можно получить непредсказуемый результат работы контроллера. Поэтому в прошивках, начиная с v3.0 введены специальные команды управления RS-232.

; порт P1
; P10 - CD input
; P11 - CTS input
; P12 - RI input
; P13 - DTR out
; P14 - RTS out
; P15 - INT_T out
; P16 - /RES out
; P17 - W_ON out

15) Чтение линий порта P3

0x55 0x16|0x56|0x96|0xD6

Команда должна выполнять чтение состояния линий порта P3 контроллера, но в результате ошибки (в оригинальной прошивке от МикроАРТ – v1.00/v1.06. Позднее - исправлено) всегда выдает код **0xFF**.

; порт P3

; P30 - RX	**input**	входной сигнал RS232
; P31 - TX	**output**	выходной сигнал RS232
; P32 - CLK_K	**input**	тактовый вход от клавиатуры IBM
; P33 - /KEYRD	**input**	вход чтения порта клавиатуры
; P34 - VE1	**input**	вход сигнала VE1 порта SYS ATM
; P35 - DATA_K	**input**	вход данных от клавиатуры IBM
; P36 - /VWR	**output**	выход ЗАПИСЬ от контроллера
; P37 - /VRD	**output**	выход ЧТЕНИЕ от контроллера

Судя по линиям порта, команда годится только для анализа линии VE1.

16) Чтение линий порта P1

0x55 0x17|0x57|0x97|0xD7

Та же самая ошибка приводит к чтению 0xFF (также позднее исправлено).

; <u>порт P1</u>		
; P10 - CD	**input**	вход RS232
; P11 - CTS	**input**	вход RS232
; P12 - RI	**input**	вход RS232
; P13 - DTR	**out**	выход RS232
; P14 - RTS	**out**	выход RS232
; P15 - INT_T	**out**	выход RS232
; P16 - /RES	**out**	выход /RESET компьютера
; P17 - W_ON	**out**	выход разрешения сигнала /WAIT

Команда годится для чтения линий **CD, CTS, RI**.

Таблица CP/M-кодов клавиатуры

	<u>код CP/M</u>	**<u>C/K</u>**	**<u>Клавиша IBM</u>**
L_400:	db 1Bh	;01	ESC
	db 31h	;02	1
	db 32h	;03	2
	db 33h	;04	3
	db 34h	;05	4
	db 35h	;06	5
	db 36h	;07	6
	db 37h	;08	7
	db 38h	;09	8
	db 39h	;0A	9
	db 30h	;0B	0
	db 2Dh	;0C	-/_
	db 3Dh	;0D	=/+
	db 8	;0E	BS
;			
	db 9	;0F	TAB
	db 51h	;10	Q
	db 57h	;11	W
	db 45h	;12	E
	db 52h	;13	R
	db 54h	;14	T
	db 59h	;15	Y
	db 55h	;16	U
	db 49h	;17	I
	db 4Fh	;18	O
	db 50h	;19	P
	db 5Bh	;1A	[
	db 5Dh	;1B	]
	db 0Dh	;1C	Enter
;			

```
                db  41h         ;1E         A
                db  53h         ;1F         S
                db  44h         ;20         D
                db  46h         ;21         F
                db  47h         ;22         G
                db  48h         ;23         H
                db  4Ah         ;24         J
                db  4Bh         ;25         K
                db  4Ch         ;26         L
                db  3Bh         ;27         ;/:
                db  27h         ;28         '/"
;
                db  60h         ;29         `/~
;
                db  5Ch         ;2B         \ / |
;
                db  5Ah         ;2C         Z
                db  58h         ;2D         X
                db  43h         ;2E         C
                db  56h         ;2F         V
                db  42h         ;30         B
                db  4Eh         ;31         N
                db  4Dh         ;32         M
                db  2Ch         ;33         ,
                db  2Eh         ;34         .
                db  2Fh         ;35         /
;
                db 0AAh         ;37         [*]
;
                db  20h         ;39         SPACE
;
                db  61h         ;3B         F1
                db  62h         ;3C         F2
                db  63h         ;3D         F3
                db  64h         ;3E         F4
                db  65h         ;3F         F5
                db  66h         ;40         F6
                db  67h         ;41         F7
                db  68h         ;42         F8
                db  69h         ;43         F9
                db 6Ah          ;44         F10
```

Цифровое поле клавиатуры

```
                db  80h+37h     ;47         [7]
                db  80h+38h     ;48         [8]
                db  80h+39h     ;49         [9]
                db  80h+2Dh     ;4A         [-]
                db  80h+34h     ;4B         [4]
                db  80h+35h     ;4C         [5]
                db  80h+36h     ;4D         [6]
                db  80h+2Bh     ;4E         [+]
```

```
                db  80h+31h   ;4F          [1]
                db  80h+32h   ;50          [2]
                db  80h+33h   ;51          [3]
                db  80h+30h   ;52          [0]
                db  80h+2Eh   ;53          [.]
;-----------------------------------
                db  6Bh       ;57          F11
                db  6Ch       ;58          F12
;-----------------------------------
```

Таблица для курсорных клавиш и для некоторых клавиш на цифровом поле клавиатуры.

```
L_511:      db 76h      ;E0   47h   Home
            db 70h      ;E0   48h   Cur Up
            db  74h     ;E0   49h   Page Up
            db 0ADh     ;     4Ah   [-]
            db  72h     ;E0   4Bh   Cur Left
            db 0B5h     ;     4Ch   [5]
            db  73h     ;E0   4Dh   Cur Right
            db 0ABh     ;     4Eh   [+]
            db  77h     ;E0   4Fh   End
            db  71h     ;E0   50h   Cur Down
            db  75h     ;E0   51h   Page Down
            db  78h     ;E0   52h   Insert
            db  79h     ;     53h   [./Del]
```

Команды управления портом RS232 и модемом для контроллера клавиатуры ATM-turbo 2+ (введены в прошивках, начиная с v3.0).

1) Команды чтения возвращают один байт:

55h 02h - чтение регистра данных RS232
(при отсутствии данных всегда 00h);

55h 42h - чтение регистра статуса RS232
Возвращаемый байт содержит:

```
; d0 - готовность приемника   RD (1-приемник готов)
; d1..d4 - всегда 0
; d5 - готовность передатчика TD (1-передатчик готов)
; d6 - буфер передатчика пуст TE (1-буфер передатчика пуст)
; d7 - признак работы с прерываниями (1-INT разрешен) -
этот бит показывает текущее состояние бита
разрешения прерывания, записываемого командой
```

55h 43h ;<STAT_RS>.

55h 82h - чтение регистра состояния модема

```
; d7 - DCD (Data Carrier Detect)
; d6 - RI  (Ring Indicator)
; d5 - DSR (Data Set Ready)
; d4 - CTS (Clear To Send)
; d3..d0  - всегда 0
```

55h **0C2h** - чтение счетчика буфера приема
(показывает, сколько байт принято)
максимальный размер буфера 57 байт.

2) Команды записи:

55h **03h <data>** - запись данных в регистр передатчика RS232

55h **43h <data>** - запись в регистр управления модема
; d0 - DTR (Data Terminal Ready)
; d1 - RTS (Request To Send)
; d2..d7 - не имеют значения (пока может быть)

55h **83h <data>** - запись в регистр управления RS232
; d0..d6 - пока не имеют значения
; d7 - бит разрешения прерывания (1-разрешено прерывание)

Разрешает выдачу сигнала INT_T с выхода порта P1 при превышении числа принятых в буфер приемника 50 байт (длина буфера 57 байт).

55h **0C3h <data>** - установка скорости обмена по RS232.
Для совместимости сделано, как в IBM PC:

<data>	<speed>
1	115200
2	57600
3	38400
6	19200
12	9600
24	4800
48	2400
98	1200

Любое другое значение игнорируется, но все указатели и счетчики приема/передачи сбрасываются в исходное состояние.

Приложение 2. Пример драйвера для IDE-HDD в среде CP/M

Так как это пример драйвера CP/M, то предполагается, что все теневые порты по умолчанию включены.

Файл: HDDRV.ASM

```
#include "allvars.ash"          ; ссылка на файл переменных CP/M, через
                                ; которые передаются команды и параметры для
                                ; выполнения задания (здесь не приводятся)
#include "ddef.asm"             ; файл с описателем параметров конкретного
                                ; внешнего устройства хранения данных

HD_DRV::
; описание портов контроллера
tlrd        equ   7ffdh         ; флаговый порт (чтение WIRQ)
hd_dat      equ   0fh           ; порт данных
hd_err      equ   2fh           ; порт ошибок
hd_seccnt   equ   4fh           ; порт счетчика секторов
hd_sect     equ   6fh           ; номер сектора
hd_cyllo    equ   8fh           ; мл. байт номера цилиндра
hd_cylhi    equ   0afh          ; ст. байт номера цилиндра
hd_head     equ   0cfh          ; номер головки
hd_comm     equ   0efh          ; порт команды
; Поскольку HDD имеет 16 разрядную шину а процессор 8-разрядную при операциях
; записи/чтения старший байт защелкивается в спец. буфер (адрес #010F)

; таблица команд драйвера  (соответствует номерам команд 0–6, получаемых от системы)
            DW    HD_RES        ; сброс HD
            DW    HD_SEEK       ; позиционирование HD
            DW    HD_FRM        ; форматирование HD
            DW    HD_REC        ; рекалибровка HD
            DW    HD_RD         ; чтение сектора
            DW    HD_NOP        ; запрещенная команда
            DW    HD_WR         ; запись сектора

HD_tst:     bit   6,(IY+DDTYP)  ; HDD уже опознан?
            ret   nz            ; да
            bit   7,(IY+DDTYP)  ; нет диска
            jr    nz,_1err
            ld    bc,hd_comm
            in    a,(c)         ; проверяем, подключен ли шлейф?
            inc   a
            jr    z,_1err       ; нет!
            ld    d,0
            call  HD_wt1
            or    a
            jr    nz,_1err
            ld    bc,hd_cylhi   ; проверяем регистры номера цилиндра
```

```
                ld      a,55h
                out     (c),a
                cpl
                ld      c,hd_cyllo
                out     (c),a
                ld      c,hd_cylhi
                in      a,(c)
                cp      55h
                jr      nz,_1err
                ld      c,hd_cyllo
                in      a,(c)
                cp      0AAh
                jr      nz,_1err


                set     6,(IY+DDTYP)        ; устанавливаем признак опознания

; програмка записи номера цилиндра в регистры HDD

hd_scl:         ld      hl,(RQCYL)          ; номер цилиндра
                ld      bc,hd_cyllo
                out     (c),l
                ld      c,hd_cylhi
                out     (c),h
                ret

_1err:          set     7,(IY+DDTYP)        ; сброс признака действительности
                pop     af                  ; SP=SP+2
nr_err:         ld      a,_NRDY             ; HDD не готов !
                or      a
                ret

HD_RES:
HD_REC:         call    HD_tst
                ld      bc,hd_comm
                ld      a,10h
                out     (c),a               ; записываем команду
HD_wait::       ld      d,64
hd_wt1:         ld      bc,hd_comm
                ld      hl,0
hd_w_lp:        in      e,(c)
                bit     7,e
                jr      z,hd_w_dn           ; появился сигнал готовности
                dec     hl
                ld      a,l
                or      h
                jr      nz,hd_w_lp
                dec     d
                jr      nz,hd_w_lp
                jr      nr_err              ; нет готовности устройства

hd_w_dn:        bit     0,e
```

```
                jr      nz,is_hd_err        ; установлен бит ошибки
;               and     50h
;               sub     50h
;               jr      nz,hd_w_lp

HD_NOP:         xor     a
                ret

; обработка ошибок
is_hd_err:
                ld      bc,hd_err
                in      b,(c)
                bit     0,b
                ld      a,_NO$ADDR$MARK
                ret     nz
                bit     2,b
                jr      nz,hd_err_ab
                bit     4,b
                ld      a,_NO$DATA
                ret     nz
                bit     6,b
                jr      z,hd_sh1
                bit     2,e
                ld      a,_CRC$ERR
                ret     z
                xor     a
                ret
hd_sh1:          bit    7,b
                ld      a,_FATAL$ERROR
                ret     nz
                ld      a,_HRDERR
                ret
hd_err_ab:
                bit     6,e
                ld      a,_NRDY
                ret     nz
                bit     5,e
                ld      a,_WR$PROT
                ret     nz
                bit     4,e
                ld      a,_IOERR
                ret     nz
                ld      a,_COMERR
                ret

; позиционирование на выбранный цилиндр
HD_SEEK:        call    HD_tst
                ld      hl,(RQCYL)
                ld      a,h
                or      l
                jr      z,HD_REC        ; если нулевой - выполняем рекалибровку
                call    hd_scl          ; устанавливаем номер цилиндра
```

```
                ld      c,hd_comm
                ld      a,70h
                out     (c),a           ; команда позиционирования
                jr      HD_wait

; подготовка к записи/чтению
hd_set_rw:
                ld      a,(RQHEAD)      ; номер головки
                or      0a0h
                ld      bc,hd_head
                out     (c),a
                ld      a,(RQSECT)      ; номер сектора
                inc     a               ; (начиная с 1)
                ld      c,hd_sect
                out     (c),a
                ld      a,d
                ld      c,hd_seccnt
                out     (c),a           ; счетчик секторов (при чтении/записи всегда 1;
                                        ; при форматировании - количество секторов на
                                        ; цилиндре)
                call    hd_scl          ; устанавливаем номер цилиндра
                ld      c,hd_comm
                ret

; чтение сектора
HD_RD:          call    HD_tst          ; проверка наличия HDD (если надо)
                ld      d,1             ; счетчик секторов
                call    hd_set_rw       ; подготовка к чтению
                ld      a,20h
                out     (c),a           ; команда чтения
                call    HD_wait         ; ожидание готовности HDD
                or      a
                ret     nz
                ld      hl,(DCBUF)      ; буфер для чтения
                ld      bc,hd_dat       ; порт данных
                inir                    ; чтение 1-х 256 байт
                inir                    ; чтение 2-х 256 байт
                ld      bc,hd_comm
                in      a,(c)
                and     8
                ret     z
                ld      a,_OVERRUN
                ret

;подготовка к записи (форматированию)
hd_setwr:
                call    hd_set_rw
                ld      a,30h           ; команда записи
                out     (c),a
; ожидание сигнала готовности записи или ошибки
hd_w_w:         in      e,(c)
                bit     0,e
```

```
                jp      nz,is_hd_err        ; ошибка
                bit     3,e
                jr      z,hd_w_w            ; к записи не готов
                ld      bc,hd_dat+256       ; порт 010fh
                ret

; запись сектора
HD_WR:          call    HD_tst
                ld      d,1
                call    hd_setwr            ; подготовка к записи
                ld      hl,(DCBUF)          ; адрес буфера
                dec     d                   ; D=0
                inc     hl

hd_wlp:
                ld      a,(hl)              ; ст. байт
                out     (c),a               ; вывод старшего байта
                dec     b                   ; порт мл. байта
                dec     hl
                ld      a,(hl)
                out     (c),a               ; вывод младшего байта
                inc     b
                inc     hl
                inc     hl
                inc     hl

                push    bc
                ld      bc,hd_comm
                in      a,(c)
                pop     bc

                dec     d
                jr      nz,hd_wlp           ; продолжаем, пока не будут записаны 256 слов
; ожидание появления сигнала WIRQ
hd_wwait:
                ld      bc,tlrd
hd_wlpd:        in      a,(c)
                and     40h
                jr      z,hd_wlpd
; сигнал WIRQ появился - данные записаны
                jp      HD_wait

; форматирование цилиндра
; P.S. Форматирование - процесс заполнения дорожки одним байтом указанным
; в RQBLN (обычно в CP/M имеет значение 0e5h), эта операция нужна для очистки
; директории.


HD_FRM:         call    HD_SEEK             ;
                or      a
                ret     nz
                ld      d,(IY+DSECTT)       ; количество секторов для форматирования
```

```
                call    hd_setwr
hd_f2:          ld      a,(RQBLN)       ; байт заполнитель
                out     (c),a           ; записываем в защелку старший байт
                dec     b
                ld      e,b
hd_f1:          push    bc
                ld      bc,hd_comm
                in      b,(c)
                pop     bc
                out     (c),a           ; записываем младший байт (старший защелкнут)
                dec     e
                jr      nz,hd_f1

                ld      bc,tlrd
hd_f3:          in      a,(c)           ; ждем сигнала WIRQ
                and     40h
                jr      z,hd_f3

                ld      bc,hd_comm
                in      e,(c)
                bit     0,e
                jp      nz,is_hd_err    ; обнаружена ошибка
                bit     3,e
                jp      z,HD_wait       ; счетчик секторов обнулился
                ld      bc,hd_dat+256   ; адрес ст. байта (порт 010fh)
                jr      hd_f2           ; продолжаем (следующий сектор)

                end
```

Файл: DDEF.ASM

```
IOBYTE          EQU     3
                aseg
                ORG     0
DVALID:         DS      1       ;+ ПРИЗНАК ДЕЙСТВИТЕЛЬНОСТИ
DTYP:           DS      1       ;+ КОД УСТРОЙСТВА
DUS:            DS      1       ;+ НОМЕР ПРИВОДА
DDTYP:          DS      1       ;- КОД ПРИВОДА  (Hi bit - unbuferized)
DHEADF:         DS      1       ;+ ЧИСЛО ФИКСИРОВАННЫХ ГОЛОВОК
DHEADR:         DS      1       ;+ ЧИСЛО СМЕННЫХ ГОЛОВОК
DCYLN:          DS      2       ;+ ЧИСЛО ЦИЛИНДРОВ НА ДИСКЕ
DSECTT:         DS      1       ;+ ЧИСЛО СЕКТОРОВ НА ДОРОЖКЕ
DBYTES:         DS      2       ;+ ЧИСЛО БАЙТОВ В СЕКТОРЕ
DALTCYL:        DS      1       ;+ ЧИСЛО СИСТЕМНЫХ ДОРОЖЕК
DBEGCYL:        DS      2       ;+ НОМЕР НАЧАЛЬНОГО ЦИЛИНДРА
DBLDR:          DS      2       ;+ ЧИСЛО БЛОКОВ НА ДИСКЕ
DBLTR:          DS      2       ;+ ЧИСЛО БЛОКОВ НА ДОРОЖКЕ
DTRACK:         DS      2       ;+ ЧИСЛО ДОРОЖЕК НА ДИСКЕ
DSEKTL:         DS      1       ;+ ДЛИНА НОМЕРА БЛОКА В СЕКТОРЕ
DDIRENT:        DS      2       ;+ ЧИСЛО ЗАПИСЕЙ В ДИРЕКТОРИИ
DIF0:           DS      1       ;+ РАЗМЕТКА ПЕРВОЙ ДОРОЖКИ
DIF1:           DS      1       ;+ РАЗМЕТКА ВТОРОЙ ДОРОЖКИ
```

```
DIF2:       DS   1              ;+ РАЗМЕТКА ВСЕХ ОСТАВШИХСЯ
                                ;  ДОРОЖЕК

DTIF:       DS   1              ;? СМЕЩЕНИЕ ПЕРВОГО СЕКТОРА

DF8:                            ;+ ПРИЗНАК 8-ДЮЙМОВОГО ДИСКА
            DS   1              ;  ИЛИ НОМЕР НАЧАЛЬНОЙ ГОЛОВКИ

DFMFM:      DS   1              ;+ ПЛОТНОСТЬ ЗАПИСИ
DFN:        DS   1              ;+ РАЗМЕР СЕКТОРА
DFGPL:      DS   1              ;+ GAP3 ДЛЯ ЧТЕНИЯ/ЗАПИСИ
DFGPF:      DS   1              ;+ GAP3 ДЛЯ ФОРМАТА

DFSRHUT:                        ;+ ВРЕМЯ ШАГА
            DS   1              ;  ИЛИ ВРЕМЯ ШАГА ПРИ SEEK

DFHLT:                          ;+ ВРЕМЯ ОПУСКАНИЯ ГОЛОВКИ
            DS   1              ;  ИЛИ ВРЕМЯ ШАГА ПРИ RECALIBRATE

DFMOTOR:    DS   1              ;+ ПРИЗНАК ВКЛЮЧЕНИЯ МОТОРА

; КОДЫ ОШИБОК, ВОЗВРАЩАЕМЫЕ ДИСКОВОЙ СИСТЕМОЙ

_ADRERR     EQU  08H            ;ОШИБКА АДРЕСАЦИИ
_CHNFND     EQU  09H            ;КАНАЛ НЕ ПРИСОЕДИНЕН
_HRDERR     EQU  40H            ;ОШИБКА АППАРАТУРЫ
_INVALID    EQU  41H            ;НЕСООТВЕТСТВИЕ ДРАЙВЕРА
                                ;АППАРАТУРЕ
_DTYPER     EQU  50H            ;НЕДОПУСТИМЫЙ НОМЕР ДРАЙВЕРА В
                                ;КАНАЛЕ
_DRNFND     EQU  51H            ;ДРАЙВЕР ОТСУТСТВУЕТ
_COMERR     EQU  52H            ;ЗАПРЕЩЕННАЯ КОМАНДА
_IOERR      EQU  53H            ;ОШИБКА ВВОДА/ВЫВОДА
_WR$PROT    EQU  54H            ;ЗАЩИТА ОТ ЗАПИСИ

_FATAL$ERROR
            EQU  56H            ;НЕ ОБРАБОТАННАЯ ФАТАЛЬНАЯ ОШИБКА

_NRDY       EQU  59H            ;НЕТ ГОТОВНОСТИ АППАРАТУРЫ
                                ;(TIMEOUT)

_NO$DATA    EQU  81H            ;СЕКТОР НЕ НАЙДЕН

_NO$ADDR$MARK
            EQU  82H            ;АДРЕСНЫЙ МАРКЕР НЕ НАЙДЕН

_OVERRUN    EQU  83H            ;OVERRUN
_CRC$ERR    EQU  84H            ;ОШИБКА В CRC

MAXDRVN     EQU  7              ;МАКСИМАЛЬНЫЙ НОМЕР ДРАЙВЕРА
;---------------------------------
```

```
; КОМАНДЫ ДИСКОВОЙ СИСТЕМЫ
; ПЕРЕДАЮТСЯ ЧЕРЕЗ RQCOM
;
_RESET      EQU   0
_SEEK       EQU   1
_FORMAT     EQU   2
_RECAL      EQU   3
_READ       EQU   4
_WSECT      EQU   5
_WRITE      EQU   6
_SETCH      EQU   7
_GETCH      EQU   8

            cseg
```

Приложение 3. Краткая таблица портов ATM-turbo 2+

Порты	IN	OUT	описание	Стр.
#FF77 (теневой)		D0 – RG0 D1 – RG1 D2 – RG2 D3 – turbo ON/OFF D4 – Z_1 (надо 0) D5 – Z_I D6 – VE1 D7 – VE0 (надо 1) A8 – PEN ON/OFF A9 – CPM ON/PFF A14 – PEN2 ON/OFF	Системный порт, управляющий графикой, тактовой частотой, а также некоторыми вспомогательными сигналами для диспетчера памяти и контроллера XT/ AT-клавиатуры	Стр.5
#FFF7 (теневой)		D0 – PG0(inv) D1 – PG1(inv) D2 – PG2(inv) D3 – PG3(inv) D4 – PG4(inv) D5 – PG5(inv) D6 – ROM/RAM D7 – Pages-trigger A14-A15 – Pages-window	Порт диспетчера памяти, конфигурирующего адресное пространство.	Стр.8
#7FFD (открытый)	D0-D5 – Not used(=1) D6 – WIRQ (from HDD) D7 - ADCS	D0 – PG0 D1 – PG1 D2 – PG2 D3 – ROM2 D4 – SCR2 D5 – Lock 48 D6-D7 – Not used	OUT: Страничный порт ZX-128 IN: Порт флагов готовности АЦП и IDE.	Стр.9 Стр.26 Стр.33
#FE (открытый)	D0-D4 – ZX-keyboard DATA(inv)/ (i8031 data) D5 – Z signal/ (i8031 data) D6 – Tape IN/ (i8031 data) D7 – Not used (=1)/ (i8031 data) A8-A15 – полуряды ZX-keyboard/ (i8031 data)	D0 – BRD0 D1 – BRD1 D2 – BRD2 D3 – Tape OUT D4 – Sound D5-D7 – Not used A3 – BRD3(inv)	OUT: Порт управления цветом бордюра и выводом сигналов на магнитофон IN: a) Чтение механической клавиатуры и магнитофона b) Работа с контроллером XT/ AT-клавиатуры	Стр.19 Стр.23 Стр.27

#FF (теневой)	D0-D5 – Not used(=1) D6 – DRQ/ D7 – INTRQ/	D0 – Blue(inv)/ (DSEL 0/1) D1 – Red(inv)/ (Not used) D2 – Not used/ (FDC reset) D3 – Not used/ (FDC halt) D4 – Green(inv)/ (Side 0/1) D5 – Low Blue(inv)/ (Not used) D6 – Low Red(inv)/ (Not used) D7 – Low Green(inv)/ (Not used)	OUT: a) Порт палитры b) Системный порт BETA-диска IN: Системный порт BETA-диска	Стр.19 Стр.31
#FF (открытый)	D0-D7 – attributes		Порт атрибутов. Считывает с видеоконтроллера байт текущего отображаемого атрибута	Стр.21
#FFFD (открытый)	D0-D7 – AY-data	D0-D3 – AY-registers D4-D7 – Not used (должны быть в нуле)	OUT: Выбор регистра AY IN: Ввод данных из регистров муз.процессора AY	Стр.24
#BFFD (открытый)		D0-D7 – AY-data	Вывод данных в регистры муз.процессора AY	Стр.24
#FB (открытый)	D0-D6 – Not used(=1) D7 – LPT busy/ready	D0-D7 – LPT/Covox Data A7 – Strobe(inv)	Одновременно LPT-порт и порт COVOX (ЦАП)	Стр.25 Стр.29
#7DFD (открытый)	D0-D7 – ADC-data		Порт чтения АЦП	Стр.26
#FA (открытый)	D0-D7 – data	D0-D7 – data	Порт внешних устройств	Стр.30
#1F (теневой)	D0-D7 – data	D0-D7 – data	BETA-диск: OUT: Регистр команд IN: Регистр состояний	Стр.32
#3F (теневой)	D0-D7 – data	D0-D7 – data	BETA-диск: Регистр дорожки	Стр.32
#5F (теневой)	D0-D7 – data	D0-D7 – data	BETA-диск: Регистр сектора	Стр.32
#7F (теневой)	D0-D7 – data	D0-D7 – data	BETA-диск: Регистр данных	Стр.32

#FFEF (теневой)	D0-D7 – IDE-data A5,A6,A7 – IDE-Registers A8 – D0-D7/D8-D15 change	D0-D7 – IDE-data A5,A6,A7 – IDE-Registers A8 – D0-D7/D8-D15 change	Порт работы с IDE-устройствами	Стр.33
#FFE7 (теневой)	D0-D7 – always =1 (Not used)	D0-D7 – Not used	Существующий, но незадействованный порт	Стр.37
#FEE7 (теневой)	D0-D7 – always =1 (Not used)	D0-D7 – Not used	Существующий, но незадействованный порт	Стр.37

Наши адреса

Интернет сайт: http://www.nedopc.com
Сайт поддержки ATM: http://atmturbo.nedopc.com
Электронные адреса:
chunin@mail.ru (Чунин Роман).
max_timonin@mail.ru (Тимонин Максим)

В книге использованы материалы:

1) Камиль Каримов «Описание контроллера клавиатуры»
2) А.Ларченко, Н.Родионов «ZX-Spectrum&TR-DOS для пользователей и программистов»
3) Газета «АБЗАЦ» N21 (статья «Spectrum и винчестер»)

NedoPC group, Тимонин Максим, 2005 год.