Сводная инфа по Sprinter`y.

Мирошниченко Александр aka Sayman
14.09.2016.

Содержание.

1. **Историческая справка.**

Спринтер 2000 разрабатывался в конце 90х – начале 2000х Иваном Макарченко (Ivan Mak, позднее Winglion, умер 9 апреля 2013го года).
Продавался Питерской фирмой «Петерс Плюс». С 2005го года Петерс Плюс сняла с производства и продажи все модели Спринтера и покинула платформу zx (чуть позднее и вовсе закрылась). В 2007м году Иван начал постепенно открывать тему Спринтера, а в 2009 году выложил в свободный доступ все имеющиеся материалы (включая исходники ядра, схему и файлы для производства).

На данный момент в массах находятся 3 версии – Sp2000, Sp2002s и Sp2003. Соотношение распространённости мне не известно. Отличаются между собой видеопамятью. На моделях Sprinter 2000 установлена видеопамять W24512AK-15 в DIP корпусах. На моделях Sprinter 2002s и Sprinter 2003 установлена видеопамять AS7C1024-12JC в SOJ корпусах. При этом, на Sp2002s память установлена «квадратом» (4 микросхемы), а на Sp2003 в шахматном порядке (так же 4 микросхемы). Однако, есть рабочий экземпляр (можно сказать, что это Sp2016) с работающей видеопамятью **W24010AS-35**.

Последняя версия биоса в собранном виде – 3.04. Стабильной считаются 3.03 и 3.04 (предположительно, для Sp2002s – нужен биос версии 3.03, а для Sp2003 нужен биос 3.04). Различия в этих биосах только в ядре (конфигурация заливаемая в fpga). Скорей всего, разница там в таймингах. Примечательно то, что на Sp2002s и Sp2003, если прошить биос ниже 3.03, то можно наблюдать периодические (или даже постоянные) артефакты на экране. При этом на Sp2016 (будем этот экземпляр так называть), таковых проблем не обнаружено.

2. **Некоторые особенности машины.**

Спринтер обладает следующими характеристиками:

- процессор Z84C15;
- RAM – 4096кб;
- Fast-RAM – 256кб;
- VRAM – 512кб;
- ROM – 256кб;
- внешние накопители на fdd, hdd;
- клавиатура AT (можно подключить PS/2 клавиатуру);
- Serial mouse;
- графические режимы: zx std, 320x256x8, 640x256x4, TXT 80x32;
- звук: Железная эмуляция AY-3-8910 стерео-OUT, covox/covox-blaster
- акселератор операций пересылки и заполнения;
- шина ISA-8.

Теперь более подробно по пунктам.

Процессор.
В режиме расширенной графики (будем называть этот режим – режим Спринтера или SpMode) процессор работает в турбо режиме на частоте 21мгц. Турбо режим снимается по клавише F12. При переключении в режим Спектрума частота сбрасывается до 3.5мгц. При этом, можно использовать клавишу F12 для включения турбирования до 7мгц (данный параметр задаётся отдельно при переключении или через 128е меню в настройках).
Важной особенностью процессора является то, что у него имеются встроенные порты и периферия. Соответственно, часть портов может конфликтовать с уже существующими портами Спектрума. Например,

внутренний порт **0x1f** конфликтует с существующим портом на Спектруме. Чтобы избежать проблем, в SpMode используется порт **0x0f**, а в режиме Спектрума обращение к данному порту перехватывается и перенаправляется на порт 0x0f. Полный список внутренних устройств вы можете найти в описании на процессор. Тут я приведу список внутренних портов:
0x10 - 0x1F, 0xEE, 0xEF, 0xF0, 0xF1, 0xF4

RAM.
Полный поддерживаемый объём оперативной памяти равен 4096кб. ОЗУ представлено в виде модуля **SIMM72**. *Поддерживаются только модули **FPM***. С модулями EDO плата не запускается. Можно использовать модули от 4мб до 16мб (по некоторым данным и больше), но при использовании модулей более 4мб всегда используется только 4мб от общего объёма модуля.
ОЗУ разбито на страницы по 16кб. Всего 256 страниц. Каждая страница может быть подключена в любое окно процессора: адресные пространства 0 – 0x3fff, 0x4000 – 0x7fff, 0x8000 – 0xbfff, 0xc000 – 0xffff.
Для подключения страницы в нужное окно используются соответствующие порты, каждый из которых отвечает за подключение к одному окну. Порты:
cpu_w0 = 0x82
cpu_w1 = 0xa2
cpu_w2 = 0xc2
cpu_w3 = 0xe2
Каждый из указанных портов является двунаправленным, т.е. работает на чтение и на запись.
В режиме Спектрума можно использовать либо полный объём озу (режим zx sprinter) либо в режиме Пентагона использовать от 129кб до 512кб по стандарту Пентагона + 256кб по стандарту Скорпиона. При этом данные порты являются теневыми в SpMode. В этом режиме эти порты являются системными и используются для переключения некоторых режимов.
Важно помнить – страница с номером 64 (0x40) содержит страницу с дешифратором. Поэтому, крайне не рекомендуется производить какие-либо манипуляции в этой странице, если вы точно не хотите изменить дешифрацию каких-то отдельных портов. Неосознанное изменение этой страницы может привести к полному зависанию машины.

Fast-RAM.
На плате присутствует микросхема статической памяти, которая когда-то давно, в 90е годы (и немного в начале 2000х) на пентагонах использовалась как кэш-память или, как её ещё называли – shadow ram. Полный объём установленный на Спринтере – 256кб. Однако, на данный момент доступно к использованию только 16кб. Включается по стандарту пентагона ***– in a,(0xfb)*** для включения и ***in a,(0x7b)*** для отключения. Включается fast-ram в 0е окно процессора (в адреса 0 – 0x3fff). При использовании этой памяти процессор работает без задержек. Небольшой тест, написанный Иваном, показывает, что в области включения fast-ram процессор работает на полной скорости.

ROM.
Полный объём ПЗУ составляет 256кб. Его содержимое: биос, с минимальным набором подпрограмм и функций для работы с устройствами, программа предварительной настройки компьютера – setup, драйвера устройств (низкоуровневые), конфигурация, заливаемая в fpga. Приведу структуру пзу по адресам/страницам:

страница ПЗУ	*что зашито*
1	*I/O и Setup (0000h...3FFFh)*
2	*чистая*
3	*чистая*
4	*чистая*
5	*чистая*
6	*чистая*
7	*чистая*
8	*чистая*
9	*Биос (20000h...23FFFh)*
10	*чистая*
11	*чистая*
12	*чистая*
13..16	*Прошивка ППЛМ (30000h...3E84Fh)*

Графические режимы.
Основными графическими режимами для SpMode являются 320p (320x256 8bpp), 640p (640x256 4bpp) и TXT (80x32). При переключении в режим Спектрума основным становится zx std, однако другие режимы так же могут быть доступны, если использовать, так называемую конфигурацию zx sprinter. Для режимов 320p, 640p и TXT доступна палитра в 16млн цветов. Палитра задаётся как 32х битные значения в формате BGRi, где i = ignored. Т.е. младшие 8 бит игнорируются. Для управления координатой Y можно использовать порт **port_y (0x89).** Для каждого из режимов предусмотрено по 2 экрана – 0й и 1й, один из которых является теневым, пока отображается другой (по аналогии с экраном в 128м Спектруме). Для переключения между экранами используется порт **rgmod (0xc9).** Более подробно об устройстве экрана (на примере 320p) смотрите в соответствующем разделе.

FDD.
В качестве контроллера fdd используется, как это уже давно принято на отечественных клонах Спектрума - *Кр1818ВГ93*. Контроллер может работать как с дискетами DD (двойная плотность, объём дискет до 720кб или в отдельных случаях до 800кб), так и с дискетами HD (высокая плотность, объём до 1.44мб для дискет 3.5 дюйма). Режимы плотности можно менять программно. Однако высокая плотность доступна только тогда, когда процессор работает на частоте 21мгц. Доступ к контроллеру возможен из любого места адресного пространства процессора. Для включения этого доступа нужно использовать **системный порт (0x3c)**:

```
ld a,0x1c
out (SysPort),a
```

Отключение доступа:

```
ld a,4
out (SysPort),a
```

HDD.
Про контроллер жёстких дисков даже не знаю что сказать. Обычный контролер. Примерно такой же как и nemo-ide. Есть возможность использовать команды типа inir и outir. Доступ к портам возможен в любом месте адресного пространства процессора.
Полный список портов hdd:

```
SP_1F7W     EQU 0X4153          ;W РЕГИСТР КОМАНД
SP_1F7R     EQU 0X4053          ;R  РЕГИСТР СОСТОЯНИЯ
```

```
SP_1F6W     EQU 0X4152      ;W CHS-НОМЕР ГОЛОВЫ И УСТР/LBA АДРЕС 24-27
SP_1F6R     EQU 0X4052      ;R  CHS-НОМЕР ГОЛОВЫ И УСТР/LBA АДРЕС 24-27
SP_1F5W     EQU 0X0155      ;W CHS-ЦИЛИНДР 8-15/LBA АДРЕС 16-23
SP_1F5R     EQU 0X0055      ;R  CHS-ЦИЛИНДР 8-15/LBA АДРЕС 16-23
SP_1F4W     EQU 0X0154      ;W CHS-ЦИЛИНДР 0-7/LBA АДРЕС 8-15
SP_1F4R     EQU 0X0054      ;R  CHS-ЦИЛИНДР 0-7/LBA АДРЕС 8-15
SP_1F3W     EQU 0X0153      ;W CHS-НОМЕР СЕКТОРА/LBA АДРЕС 0-7
SP_1F3R     EQU 0X0053      ;R  CHS-НОМЕР СЕКТОРА/LBA АДРЕС 0-7
SP_1F2W     EQU 0X0152      ;W СЧЕТЧИК СЕКТОРОВ
SP_1F2R     EQU 0X0052      ;R  СЧЕТЧИК СЕКТОРОВ
SP_1F1W     EQU 0X0151      ;W ПОРТ СВОЙСТВ
SP_1F1R     EQU 0X0051      ;R  ПОРТ ОШИБОК
SP_1F0W     EQU 0X0150      ;W ПОРТ ДАННЫХ МЛАДШИЕ 8 БИТ
SP_1F0R     EQU 0X0050      ;R  ПОРТ ДАННЫХ МЛАДШИЕ 8 БИТ
```

ISA8.

На мой взгляд, совершенно лишняя и не нужная вещь в рамках данного компьютера (и вообще для любого 8 бит компьютера). Возможно, преимущества тут есть перед zxbus, но на данный момент найти устройства для этой шины очень и очень сложно. Возможно, чуть позже опишу работу с этой шиной с программной точки зрения (точнее сказать, процитирую автора из его документации, т.к. сам я с устройствами на этой шине не работал).

Клавиатура.

В качестве клавиатуры можно использовать любую доступную PS/2 клавиатуру (в любом комп.магазине любого города таких клавиатур целые вагоны). Клавиатура подключена через внутренний порт процессора. В отличии от других отечественных клонов, на Спринтере контроллер клавиатуры работает не с битами полуряда, а со скан кодами (аналогично пц).

Мышь.

В качестве мышки можно использовать любую доступную мышь, подключаемую на serial port. СОгасно некоторым обрывочным данным, реализована эмуляция kempston mouse, но как оно включается и какую мышку куда подключать, я так и не смог понять.

Звук.

В качестве звуковых устройств доступны – AY-3-8910 и covox поддерживающий несколько режимов работы. При этом, в железном виде AY на плате отсутствует. Он реализован через эмуляцию на ahdl, являющейся частью ядра Спринтера. Многие могут подумать – «фи, эмуляция…». Но я уверяю вас, эмуляция тут очень и очень качественная. Далеко не каждый сможет отличить железный AY от эмулируемого на Спринтере.
Для covox`a есть несколько режимов работы. Первый – режим обычного covox`a по схеме пентагона. Второй – режим covox-blaster`a с наличием fifo. В этом режиме так же доступны различные варианты – моно, стерео, 8бит, 16бит, частоты от 8кгц до 110кгц (на самом деле нет смысла использовать частоты выше 44кгц). Fifo является зацикленным.

Акселератор.

Почти во всех конфигурациях есть возможность использовать акселератор для пересылки и/или заполнения области памяти. **ВАЖНО – перед началом использования акселератора, запретите прерывания!!!**
Акселератор включается путём перехвата команд процессора. Акселератор пересылает данные максимальным размером транзакции в 256 байт. Ожидать конец транзакции не требуется.

Дополнительная информация.

В заключение данного раздела я бы хотел сказать несколько слов о питании для платы и о её форм-факторе.

Изначально в качестве форм-фактора был выбран baby-AT. В качестве источника питания должен был выступать AT блок питания. Но с течением времени данный форм-фактор устарел, и его перестали поддерживать. Сегодня невозможно найти в магазинах (или где-то ещё в свободном виде) корпуса и блоки питания AT, а современные корпуса не поддерживают установку плат в формате AT. Однако есть решение для подключения современных блоков питания ATx. На просторах площадки Aliexpress есть множество продовцов, которые с радостью вам продадут (по довольно не большой цене) переходник AT-ATx. Качество очень хорошее. Пример такого переходника можно найти тут:
https://ru.aliexpress.com/item/20Pin-ATX-to-2-Port-6Pin-AT-PSU-Converter-Power-Cable-Cord-20cm-for-286-386/32323305712.html?spm=2114.13010608.0.88.B7EXKQ

С корпусами всё немного сложнее – тут каждый сам себе изобретатель велосипеда. Я использовал корпус mini-ITX от inWin, в который установил заранее выпиленную переходную пластину из пластика, в которой рассверлил отверстия для ATx и для платы Спринтера. В этом случае я потерял возможность использования слотов ISA8, т.к. плата теперь лежит высоко и устройства которые (в теории) можно было бы использовать, не влезают. Собственно, корпус сам по себе не высокий (формат desktop`a, т.е. лежачий корпус).

3. **Программирование**

Начну с того, что все программы для Спринтера лучше всего писать под его ДОС. Некоторые вещи быстрее и/или удобнее выполнять именно под ДОС, чем вручную или запускать в режиме Спектрума. У каждого исполняемого файла (EXE) есть свой заголовок:

```
                org 0x8100-512
start_addr:
                DB 'EXE'                ; EXE ID
                DB 0            ; EXE ВЕРСИЯ
                DW 0x200        ; 512, МЛ. СМЕЩ. КОДА
                DW 0            ; СТ. СМЕЩ. КОДА
                DW 0            ; END-BEG, ПЕРВИЧНЫЙ ЗАГРУЗЧИК
                DW 0,0          ; РЕЗЕРВ0
                DW 0            ;
                DW begin        ; АДРЕС ЗАГРУЗКИ КОДА
                DW begin        ; АДРЕС ПЕРЕДАЧИ УПРАВЛЕНИЯ
                DW 0xBFFF       ; АДРЕС СТЕКА
                DS 490          ; РЕЗЕРВ1
```

Вам важно запомнить, что такой (подобный) вид заголовка годится для небольших проектов/программ, размер которых не превышает ~32кб. Весь заголовок имеет размер 512 байт. РЕЗЕРВ1 добивает заголовок до конца и уже на 513м байте начинается код программы. Однако, ДОС позволяет запускать большие программы, не дробя их на куски. Примером такого проекта является, игра TITD (или его инсталлер, который имеет размер 1мб). Чтобы ДОС мог грузить большие программы, в заголовке требуется поменять несколько моментов:

```
                org 0x8100-0x16
EXEHeader:
                db "EXE"
                db 0
                dw LoaderStart -EXEHeader       ;Размер заголовка
                dw 0
                dw LoaderEnd - LoaderStart      ;Размер загрузчика
                dw 0,0,0
                dw LoaderStart                  ; Адрес загрузки (загрузчика)
                dw LoaderStart                  ; Адрес передачи управления
                dw LoaderStart -1               ; Стэк.

LoaderStart
                Тело загрузчика…
```

Как можно заметить, в заголовке теперь отсутствует ds 490, а так же поменялись некоторые поля в заголовке. Тут, в общем-то, всё более или менее понятно и логично – размер заголовка, размер загрузчика, адрес загрузки, передачи и стэк. Там где было ds 490, теперь должен быть код загрузчика, который догрузит остальные части программы, раскидает их по страницам памяти (если нужно) и в конце сделать jp mainStart.
Для себя я пока делаю для большинства программ вот так:
Выкинул заголовок (по первому варианту) в отдельный файл head.inc, в теле основной программы (например, main.asm) делаю так:

```
                device zxspectrum128
```

```
                include "..\..\include\dss.inc"
                include "..\..\include\head.inc"

begin:          бла бла бла...код программы


code_end:
                savebin "progname.exe", code_start,code_end- code_start
```

Содержимое head.inc

```
                org 0x8100-512
code_start:
                db "EXE"
                db 0
                dw 0x200
                dw 0
                dw 0
                dw 0
                dw 0
                dw 0
                dw begin
                dw begin
                dw 0xbfff
                ds 490
```

Сборка осуществляется через батник:

```
@echo off

if EXIST progname.exe (
        del progname.exe
)
if EXIST progname.lst (
        del progname.lst
)

..\..\asm\sjasmplus.exe main.asm --lst= progname.lst
if errorlevel 1 goto ERR
echo Ok!
goto END

:ERR
del progname.exe
pause
echo ошибки компиляции...
:END
```

Для работы с ДОСом есть отдельный мануал, когда-то написанный Иваном. В нём приведён список системных вызовов и аргументы к ним (с редкими комментариями). В целом, по этой документации можно понять всё. Там описаны вызовы для работы с файлами, памятью, печать

символов, переключение в графические режимы и т.д. Тут я дам несколько комментариев к некоторым вызовам.

Первое. Все номера всех вызовов в ДОСе всегда передаются через регистр “**c**”.
Второе. В отличие от ТРДОС, тут не стоит забирать какую угодно страницу памяти. Это может привести к непредсказуемым последствиям. Например, когда запускается дос и следом консоль, консоль получает для себя некий кусок памяти (1 или 2 страницы). Какие это страницы – не известно. Запустили Flex Navigator – аналогично – страницы будут для него выделены динамически. Заранее не известно, какие это страницы. Если вы будете принудительно забирать под себя страницы памяти, не «информируя» об этом систему, то вы рискуете либо, затереть данные от коммандера, консоли, доса или ещё какой программы либо, другая программа затрёт ваши данные, если они должны сохраняться в памяти. Для выделения памяти у ДОСа есть соответствующий вызов – getmem (0x3d). Дос может обрабатывать запрос сразу в несколько страниц. Запросив, к примеру, 8 страниц ДОС возвращает вам в регистре «**A»** идентификатор. Этот идентификатор необходимо передать в биос, указав в регистровой паре HL адрес сохранения таблички с выделенными страницами. Нужно помнить, что эта табличка должна быть на 1 байт больше, чем количество запрашиваемых страниц. Последний байт в этой табличке будет равен 0xFF как признак конца этой таблички. Вы можете поэкспериментировать с этим вызовом и проверить результаты либо в отладчике эмулятора либо, в отладчике demon.exe на реальной машине (demon.exe, прочтите документацию на этот отладчик). Пример запроса 8 страниц памяти

```
                ld c,getmem
                ld b,8
                rst 0x10                ;на выходе A содержит идентификатор выделенных страниц
                ld hl,pages
                ld c,0xc5
                rst 8
                …

Pages:          ds 9            ;9й байт будет иметь значение 0xff после rst 8, а первый байт можно
                                ;использовать как идентификатор
```

Чтобы подключить нужную страницу потом, можно использовать вариант:

```
                pop hl          ;восстановили pages
                ld a,(hl)
                out (cpu_w3),a  ;подключили страницу в 3е окно проца.
```

И так далее и тому подобное. Изобретайте свои конструкции, проверяйте и эксперементируйте.

Теперь хочу прокомментировать работу с акселератором. **Перед тем, как использовать акселератор, запретите прерывание!!!**
Акселератор включается путём перехвата команд процессора. Есть основной набор команд, которые перехватываются и распознаются ядром как обращение к акселератору: LD A,A; LD B,B; LD C,C; LD D,D; LD E,E; LD H,H, LD L,L.

Назначение команд следующее:

LD B,B - выключить акселератор.

LD D,D - включить акселератор в режим приема байта размера блока
далее следует команда типа LD A,cnt, где cnt и будет новым
размером блока. Если размер блока был установлен ранее,
его можно не устанавливать.

LD C,C - Операция Fill - заполнение одним байтом. Последующая команда типа LD (HL),A приведет к заполнению указанного ранее количества байт значением A

LD E,E - Операция Fill для графического экрана - заполнение вертикальных линий.

LD H,H - rezerved

LD L,L - копирование блока. Последующая команда типа LD A,(HL) приведет к заполнению ОЗУ акселератора данными из адреса (HL), а команда типа LD (DE),A приведет к перезаписи данных из ОЗУ акселератора в основное или видео-ОЗУ.

LD A,A - копирование блока для графического экрана подобна команде LD L,L, но работает с вертикальными линиями экрана.

Самый простой пример по переброске куска данных при помощи акселератора:

```
        di
        ld d,d
        ld a,0
        ld l,l
        ld a,(hl)           ;в этот момент данные залетают из озу во вн.память fpga
        ld (de),a           ;а тут наоборот данные залетают в озу.
        ld b,b
```

Как можно заметить, в примере и в описании сказано, что размер транзакции может быть задан, якобы, только через конструкцию вида ld a,cnt. На самом деле это не так. Bo1x, вместо регистра «A» можно использовать любой другой по вашему усмотрению. Bo2x, применятся, могут и иные конструкции, вроде pop af или подобные.

Отдельной темой является работа с графическими режимами. Тут я дам несколько комментариев по работе с режимом 320x256x8, работа в режиме 640x256x4 выглядит примерно таким же.
Для того чтобы включить режим можно использовать функционал который уже есть в ДОСе:

```
        ld c,setvmode       ;0x50
        ld a,_320p          ;0x81
        rst 0x10
        ld a,norm_scr       ;0x50
        out (cpu_w3),a      ;cpu_w3 = 0xe2
```

После выполнения этих строк будет включён режим 320x256. Далее можно отчистить экран от «случайного» мусора (или от того, что там было изображено ранее):

```
        ld (sp_2+1),sp
        ld sp,0xc040+640
        ld b,160
        di
loop_cls:   ld d,d
        ld e,0
        ld b,b              ;*
        ld d,e
        ld e,e
        push de
        push de
        ld b,b
        djnz loop_cls
sp_2:       ld sp,0
```

Обратите внимание на комментарий со звёздочкой. В этом месте стоит команда отключения акселератора.

Автор (Иван) рекомендует в своих мануалах всегда делать отключение акселератора между сменами режимов работы акселератора. На самом деле это не совсем так. Делать это нужно только в тех случаях, когда, например, следующая команда процессора предполагает какую-то манипуляцию с регистрами или числами и при этом эти данные не должны попасть в акселератор. В приведённом выше примере отключение сделано потому, что выполнение команды ld d,e изменило бы размер транзакции для акселератора. Вы можете проверить подобные варианты сами. Однако проверить это можно только на реальном железе, т.к. эмулятор подобные комбинации не поддерживает (только стандартные, описанные в документации от Ивана, но, как я уже говорил ранее, таких комбинаций чуть больше, чем в мануале). В качестве эмулятора можно использовать одну из последних версий ZX MAK2.

Включив режим 320р и зачистив экран нужно (если это требуется) задать палитру. Прочитать о том, как выглядит палитра на низком уровне (в каких адресах расположена, какие байты куда кидать и т.д.) вы можете в документации от Ивана (Sp2000_man.pdf). Тут я опишу вариант с использование биоса.

```
ld hl,pallete          ; данные палитры:  список цветов по четыре байта B,G,R,Y
ld e,beg_color         ; начальный цвет
ld d,num_colors        ; количество устанавливаемых цветов
ld b,pal_mask          ; маска при установке палитры. Для нормального
                       ; режима должна быть FF
Ld c,0xa4              ; номер функции
ld a, page_pal         ; номер палитры 0..15 значения от 8 до 15 резервные
rst 8
```

В общем случае, если задавать 256 цветов, то вся процедура выглядит вот так:

```
ld hl,pallete
ld de,0
ld bc,0xffa4
ld a,0
rst 8
```

Насколько я помню, лучше всего эту операцию выполнить дважды, первый раз для 0го экрана, второй раз для 1го. Ссылка на принадлежность к экрану задаётся через регистр «A».

В процессе вывода спрайтов/тайлов и иных «видов» изображений вам может потребоваться переключение с одного экрана на другой. Для этого нужно использовать порт RGMOD (0xc9). Обновление экрана происходит в момент, когда луч дорисовал экран до конца и переместился на начало, на первую строку. Последовательность «ei:halt» приводит к тому, что экран обновляется. Если запретить прерывания и пытаться постоянно выводить данные на экран, то скорей всего вы ничего не увидите (за редким случаем можно видеть данные рывками). Экраны 0й и 1й расположены на одной строке с разницей по адресам в 320 байт. Т.е. включив экран, скажем, в 3е окно (на адрес 0xc000), 0я координата по X для 0го экрана будет на адресе 0xc000, а координата по X для 1го экрана будет по адресу X0+320, т.е. на адресе 0xc140. Чтобы перемещаться по координате Y, можно использовать один из двух способов. Первый (на мой взгляд, медленный и не удобный), выполнять инкремент адреса экрана на 1024 байта (полный размер одной строки, всего в одной странице будет 16 строк). Второй вариант, использовать порт **port_y (0x89).** Данный порт работает на чтение и на запись. Важное замечание по данному порту: когда вы производите чтение или запись данных в адреса 0x4000 – 0x5aff (экран Спектрума), не обходимо выставить значение порта port_y в значение 0xc0 (192) или любое другое значение, больше 191. Более подробная информация об этой особенности можно найти в документации от Ивана (Sp2000_man.pdf).

Как уже ранее говорилось, акселератор может пересылать данные или производить заполнение байтом некой области в озу или на экране. Все режимы акселератора могут забрасывать данные на экран. Например, используя команду копирования (ld l,l), можно переместить данные из озу на экран. В этом случае данные на экране будут отображены «линейно», т.е. горизонтально. При этом не забывайте, что требуется произвести: изменение координаты Y и смещение по «изображению». Если использовать режимы акселератора именно для графического экрана, то тут схема будет выглядеть примерно так: Берём данные линейно, командой ld l,l, а на экран отправляем командой копирования для графического экрана – ld a,a. В данном случае данный будет из озу выгребаться «линейно», а на экран будут записываться данные уже вертикальными линиями (а это уже не совсем линейно, а на каждую строку, пропуская каждые 1024 байт). ПО этой причине, спрайт или иное изображение не может храниться в обычном виде, его нужно развернуть на 90гр вправо. При этом нужно помнить, что при использовании таких команд, как ld a,a и ld e,e автоматически производят увеличение координаты Y!

Если у вас есть какие-либо вопросы, вы можете задать их:
На форуме: http://zx-pk.ru/threads/10983-sprinter-vtoroe-prishestvie.html
Написав мне на почту: sayman_nsk@bk.ru
Исходники и прочая информация доступна на git: https://github.com/SaymanNsk/Sprinter200x

01.09.2016
Мирошниченко Александр aka Sayman