Автор Mac Buster

1.2. Доступные видеорежимы

В нашем распоряжении **два графических** режима.
Один из них имеет разрешение **320** точек по горизонтали и **256** точек по вертикали.
В этом режиме мы можем каждую точку окрашивать в любой из **256 цветов**, т.е. одной точке соответствует один байт видеопамяти.
Таким образом экран в этом режиме требует **320x256=81920** байт.
Второй режим имеет **640** точек по горизонтали, **256** точек по вертикали, где каждая точка может быть окрашена в один из **16 цветов**, т.е. отводится только **4 бита на точку**, а значит каждый байт содержит информацию сразу о двух точках
(младший полубайт об одной точке, старший - о другой).
В каждом режиме мы можем использовать **две идентичные видеостраницы** (нумеруемых **0** и 1), и четыре палитры (к сожалению **только одна палитра** может быть использована в текущий момент).
В терминах "Спринтера" блоки имеющие **320** или **640** точек по горизонтали и одну точку по вертикали называются строками.

1.3. Структура видеопамяти

В настоящее время объем видеопамяти "**Спринтера**" составляет **256 килобайт**.
Возможно Вы обратили внимание на то, что числа количества строк (их **256**) и объёма видеопамяти (**256 килобайт**) одинаковы.
Это не простое совпадение, дело в том, что на каждую строку отводится по одному килобайту видеопамяти, причем вне зависимости от того, сколько точек по горизонтали содержит строка данного режима. Первые **320 байт** строки (с номерами **0…319**) определяют цвета точек для первой видеостраницы, следующие **320 байт** (с номерами **320-639**) — то же для второй видеостраницы, остальные байты (с номерами **640…1023**) режиме, а так же для хранения данных палитры (будет описана позже).
Структура описателя экрана мне пока неясна полностью, поэтому я в данном руководстве не буду её рассматривать.

1.4. Как работать с видеопамятью "Спринтера"

Для простоты я сначала покажу вам как надо записывать данные в одну строку видеорежима с разрешением **320x256**.
Что бы это сделать, нам требуется произвести следующие действия:

- установить видеорежим;
- установить страницу видеопамяти в адресное пространство **Z80**;
- выбрать строку видеопамяти;
- записать данные по необходимому смещению от начала строки.

Видеорежим будем устанавливать с помощью функции "**SETVMOD**" набора функций системы "**Estex**".
Перед её вызовом необходимо в регистр **A** загрузить номер видеорежима (для режима **320x256** это **0x81**, для **640x256** это **0x82**), затем в регистр **B** загрузить номер видеостраницы (**0** для первой страницы и **1** для второй страницы), а в регистр **C** поместить номер функции "**SETVMOD**" системы "**Estex**" — **0x50**.
После чего мы вызываем функцию с помощью инструкции **RST 0x10**.
Полностью это должно выглядеть примерно так:

```
LD   A,0x81 ; мы собираемся  использовать  режим 320x256
LD   B,0x00 ; нам нужна первая  страница
LD   C,0x50 ; номер функции "SETVMOD"
RST 0x10 ;   установить видеорежим
```

Не помешает проверить была ли выполнена наша функция без ошибок.
Если в процессе выполнения произошла ошибка, то после возвращения из подпрограммы вызываемой нами с помощью **RST 0x10**, будет установлен флаг переноса **C**, а значит мы не сможем работать с видеопамятью.
Нам надо както обрабатывать такую ситуацию.
Предположим, что у нас где-то есть процедура "**VMError**" для такого случая, выводящая на экран строку "**Err — Unable to set videomode!**", тогда сразу после **RST 0x10** следует поставить проверку установлен ли флаг **C**.

```
JR C,VMError ; произошла ошибка?
```

Кроме того, стоит заметить, что раздельная загрузка регистров **B** и **C** была сделана только для наглядности, и в следующих примерах я буду использовать одну инструкцию загрузки значения в регистровую пару **BC** — "**LD BC**, 0xNNNN".
То есть предыдущий фрагмент кода будет выглядеть следующим образом:

```
LD   A,0x81      ; мы собираемся  использовать режим  320x256
LD   BC,0x0050 ; первая страница,номер  функции "SETVMOD"
RST 0x10         ; установить видеорежим
JR    C,VMError  ; произошла ошибка?
```

Теперь надо установить видеостраницу в адресное пространство **Z80**. Видеопамять разбита на **16- килобайтные** страницы (по **16** строк на страницу) с номерами **0x50...0x5F**, которые могут быть подключены так же, как это делается с обычной оперативной памятью.
Причем следует иметь в виду, что номера страниц видеопамяти так же определяют режим доступа к видеоданным (обычный; без изменения данных в основном **ОЗУ**; с так называемым "прозрачным" цветом;).

Сейчас мы ограничимся самым простым для понимания обычным режимом.
Будем подключать видеопамять в страницу **3** начинающуюся с адреса **0xC000** (или **49152** в десятичной системе счисления).
Это значит, что адрес первой по счету строки в видеостранице будет начинаться с этого адреса (**0xC000**).
В данном примере будем использовать прямую запись в порт, однако, хочу предупредить, что поступать так в приложениях, предназначенных для распространения, не рекомендуется (лучше использовать для этой цели специально отведенные функции "**Estex**" или **BIOS**).
Все, что нам надо сделать — это предварительно сосчитать и сохранить данные из порта, предназначенного для указания номера страницы, подключаемой в третье окно — **0xE2**, а затем записать в этот порт значение **0x50** (номер страницы отведенной под видеопамять).

```
IN    A,(0xE2) ;            считываем текущее  значение
LD    (OldWin3Page),A ;  сохраняем его  в памяти, чтобы
                          ;  вернуть при выходе
LD    A,0x50 ;              номер страницы  видеопамяти
OUT (0xE2),A ;              записываем новое  значение
```

Теперь в адресное пространство подключена страница видеопамяти, и мы уже можем записывать в нее данные.
Но перед этим нам ещё требуется указать в какую именно строку надо записывать данные, для чего в порт **0x89** надо ввести номер строки (**0...255**).

```
LD    A,0x10 ; выбираем  семнадцатую  строку
OUT (0x89),A ; записываем  номер строки  в порт
```

После этого с адреса **0xC000** у нас располагаются данные видеопамяти относящиеся к семнадцатой строке, с адреса **0xC400** — к восемнадцатой, с **0xC800** — к девятнадцатой, и так далее, до тридцать первой, с шагом в один килобайт.
Записав по адресу **0xC000** какоенибудь число, мы изменим цвет самой первой слева точки семнадцатой сверху строки на экране.

```
LD    A,0x12 ; выбираем цвет
LD    (0xC000),A ; записываем  в видеопамять
```

Мы выполнили все, что планировали, осталось только произвести какую-либо задержку, чтобы появилась возможность увидеть результат нашего “титанического труда”.
Проще всего использовать для этого функцию “**WAITKEY**” с номером **0x30**, которая ждет нажатия любой алфавитно-цифровой клавиши.
Делается это следующим образом:

```
LD    C,0x30 ; загружаем номер  функции  “WAITKEY”
RST  0x10 ; вызываем “Estex”
```

Восстанавливаем старое значение в использованной нами странице памяти и устанавливаем текстовый режим:

```
LD    A,(OldWin3Page);
OUT (0xE2),A ;
LD    A,0x03 ; текстовый режим
LD    BC,0x0050 ;
RST   0x10 ;
```

Теперь нам надо вернуться в систему или вызвавшую нас программу.
Для этого существует функция “**Estex**” под названием “**Exit**”, имеющая номер **0x41**.
Предварительно в регистр **B** следует поместить код ошибки, либо **0**, если ее не было.
Мы будем считать, что никаких ошибок не было:

```
Exit LD   BC,0x0041 ;

     RST  0x10 ;
```

Вот и почти вся наша программа.
Осталась только процедура вывода строки о невозможности установки видеорежима и последующим переходом на метку “**Exit**”.

```
VMError     LD    HL,ErrMessage ;
            LD    BC,0x005C ;
            RST  0x10 ;
            JR    Exit ;13
ErrMessage DB “Err — Unable to set videomode!”,0x0D, 0x0A,0x00
```

Первая часть руководства окончена.
Вы научились подключать видеопамять, выбирать строку и выводить на экран точку.
Этого вполне достаточно для начала.
Настоятельно рекомендую поэкспериментировать с выводом, например, сделать так, чтобы точка постоянно меняла цвет :)
Ниже приведен полностью работающий, готовый к ассемблированию с помощью **Z80asm 1.5**, пример нашей программы:

```
                ORG 0x7E00 ;
                DB  “EXE” ;
                DB  0x00 ;
                DW  0x0200,0x0000 ;
                DW  0x0000 ;
                DB  0x00,0x00 ;
                DB  0x00,0x00 ;
                DW  0x0000 ;
                DW  0x8000 ;
                DW  EntryPoint ;
                DW  0xBFFF ;
times 0x1EA     DB  0x00 ;
EntryPoint      LD  A,0x81
                LD  BC,0x0050
                RST 0x10
                JR  C,VMError
                IN  A,(0xE2)
                LD  (OldWin3Page),A
                LD  A,0x50
                OUT (0xE2),A
                LD  A,0x10
                OUT (0x89),A
                LD  A,0x12
                LD  (0xC000),A
                LD  C,0x30
                RST 0x10
                LD  A,(OldWin3Page)
                OUT (0xE2),A
                LD  A,0x03
                LD  BC,0x0050
                RST 0x10
Exit            LD  BC,0x0041
                RST 0x10
VMError         LD  HL,ErrMessage
                LD  BC,0x005C
                RST 0x10
                JR  Exit
ErrMessage      DB  “Err — Unable to setvideomode!”, 0x0D, 0x0A,0x00
OldWin3Page     DB  0x00
```

Заголовок исполняемого файла для «Спринтера»

Решив попробовать свои силы в программировании для «Спринтера»,
вы можете воспользоваться ассемблером на самом «Спринтере» (например, **OrgAsm**),
либо любым подходящим вам кроссассемблером.
Спринтеровские ассемблеры умеют автоматически создавать исполняемые файлы.
Однако, если вы выбрали второй вариант, то для создания исполняемого файла вам
необходимо добавить к своей программе заголовок, содержащий в себе служебную
информацию, указывающую операционной системе как и куда загружать
исполняемый код.
Вид заголовка показан на рисунке (префикс **0x** означает запись в шестнадцатеричной системе
счисления).
Обычно код программы размещают начиная с адреса **0x8100**, и первая строка в заголовке
указывает начальный адрес компиляции – **0x7F00**, т.е.
0x8100–0x0200 (общая длина заголовка **512** байт).
После этого следуют три латинские литеры «**EXE**»,
служащие для определения типа файла.
За ними указывается номер версии исполняемого файла (сейчас используется нулевая версия).
Далее располагается информация о смещении кода вашей программы от начала исполняемого
файла, т.е. фактически указывается длина заголовка (**512** байт).

Если ваш ассемблер не поддерживает тип «двойное слово», можно заменить эту строку на:
DEFW 0x0000,0x0200
Теперь надо указать длину начального загрузчика, если он есть (либо **0** если его нет).
В том случае, если ваша программа разбита на независимые модули (или секции кода и данных),
вы можете организовать программу таким образом, чтобы при запуске она самостоятельно
принимала решение о том, что и как загружать в память и запускать, выделив часть кода
в начальный загрузчик.
Затем идут **6 байт**, зарезервированных под дальнейшее развитие формата исполняемого файла.
Обратите внимание, что для соблюдения совместимости с последующими форматами следует
всегда заполнять зарезервированные байты нулевыми значениями.
После зарезервированных байт указывается начальный адрес загрузки кода вашей программы.
За ним указывается адрес ее запуска, т.е. значение, которое будет занесено в регистр **PC**.
Сразу после него идет адрес вершины стека (для загрузки регистра **SP**), устанавливаемый
при запуске вашей программы, которое не рекомендуется менять.
И в конце заголовка идут **490** зарезервированных байт.
Далее с метки **START** начинается код вашей программы.

```
        ORG 0x7F00
        DEFW 0x5845 ; EXE Signature
        DEFB 0x45 ; Reserved (EXE type)
        DEFB 0x00 ; Version of EXE file
        DEFD 0x00000200 ; Code offset
        DEFW 0x0000 ; Primary loader size or 0
        DEFD 0x00000000 ; Reserved
        DEFW 0x0000 ; Reserved
        DEFW START ; Loading address
        DEFW START ; Starting address (register PC)
        DEFW 0x0BFFFh ; Stack address (register SP)
        DEFS 0x1EAh ; Reserved 1
START первая команда вашей программы
```