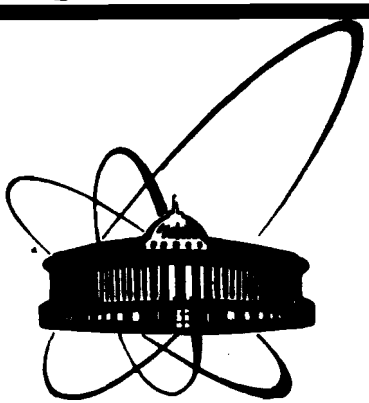


сообщения
объединенного
института
ядерных
исследований
дубна

Р11-89-872

М.В.Чижов

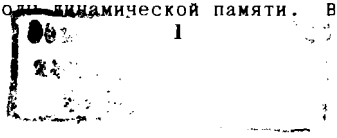
НОВЫЙ ГРАФИЧЕСКИЙ РЕЖИМ CGA АДАПТЕРА

1989

Стандартный цветной графический адаптер фирмы IBM даёт возможность реализовать режимы цветной графики среднего разрешения (320*200 точек с четырьмя цветами) и черно-белую графику высокого разрешения (640*200 точек). Однако для некоторых приложений желательно иметь цветную графику большего разрешения, чем предлагает стандартный адаптер. Существующие в настоящее время новые видеоадаптеры типа EGA, VGA и другие отличаются большей ценой и требуют использования специальных дорогих дисплеев. Небольшие изменения в стандартной схеме с добавлением пяти микросхем (D2, D4, D11, D40', D41') позволяют решить эту проблему. Модифицированная таким образом видеокарта помимо основных режимов работы также поддерживает цветную графику высокого разрешения (640*200 точек с четырьмя цветами). Разработанная схема представлена в приложении I.

Первое, что необходимо сделать для этого, это надо увеличить буфер видеопамяти с 16 Кбайт до 32 Кбайт. Однако некоторые программисты, не знакомые с принципиальной схемой видеокарты, могут тут же возразить, что DEBUG и другие программы показывают наличие 32 Кбайт видеопамяти. Эта ошибка возникает, конечно, не по вине сервисных программ. Просто адресный бит A14 не используется при выработке сигнала выборки, и адреса B8XXX и BCXXX отождествляются. Из этой ошибки можно извлечь и выгоду. Не изменяя схемы выборки, мы легко можем расширить память.

Память расширяется добавлением двух микросхем TMS4416 (D40', D41') с организацией 16 Кбит * 4. Таким образом, образуются два банка видеопамяти по 16 Кбайт с общими шинами адреса, данных и управления, за исключением сигнала "CAS" в последней. Этот сигнал необходим для подачи данных на выходы динамической памяти. В нашем случае он служит



для выбора одного из двух банков памяти. Центральный процессор выбирает их с помощью адресного разряда A14. Видеоконтроллер (Motorola 6845) также имеет возможность сканировать оба банка этой видеопамяти и в зависимости от режима работы отражать её содержимое на экране. Сигналы выборки банков памяти "CAS0" и "CAS1" вырабатывает микросхема D2 (КР531ИД14).

Второе, это необходимо выбрать адрес порта и сделать дополнительный однобитный регистр для переключения адаптера на новый режим работы. Естественно использовать адрес существующего порта 3D8 переключения режимов работы и расширить этот шестибитный порт до семи значащих бит. Таким образом, в зависимости от значения седьмого бита порта 3D8 будет реализовываться новый или стандартные режимы работы адаптера. В качестве однобитного регистра порта мы используем один из триггеров микросхемы D11 (К555ТМ2), который защёлкивается сигналом выбора порта "SEL1". При инициализации адаптера устанавливается нулевое значение этого регистра, что соответствует стандартным режимам работы.

И, наконец, третье, это надо осуществить необходимые коммутации. Выход триггера управляет мультиплексором К531КП11П (микросхема D4) через вход 1. Для сканирования видеопамяти контроллером при стандартном режиме работы старший адресный бит A13 подаётся с выхода 2 микросхемы D10 на вход 17 микросхемы D42. В текстовом режиме или в режиме обычной графики в качестве этого бита выбирается выходной сигнал контроллера "MA12" или "RA0", соответственно. Для осуществления нового режима работы необходимо, чтобы в графическом режиме адресный бит A13 определялся выходным сигналом "MA12", а не "RA0". Поэтому при появлении высокого потенциала на входе 1 мультиплексора он подключает выход 16 контроллера D32 ко входу 17 микросхемы D42.

Мультиплексор делает ещё одну коммутацию, связанную с выбором банков памяти контроллером в стандартных и новом режимах работы. В стандартном режиме расширение памяти используется просто как

дополнительная память, доступная видеоконтроллеру, и сигнал выбора банков памяти определяется старшим адресным битом контроллера "MA13" (в стандартном адаптере этот выход не используется). При переходе к новому графическому режиму мультиплексор подаёт на вход микросхемы выбора банка памяти D2 сигнал "RA0". Заметим, что этот момент является самым существенным для понимания работы контроллера в новом режиме.

Программирование контроллера на новый режим можно легко осуществить, если принять во внимание последнее замечание. Действительно, в стандартных графических режимах область памяти видеоконтроллера делится на две области по 8 Кбайт каждая: при младших адресах содержатся байты чётных линий экрана, а при старших адресах содержатся байты нечётных линий. Тот же принцип сохраняется и при распределении памяти в новом графическом режиме. Только теперь мы имеем две области по 16 Кбайт каждая. Выбор байтов чётных или нечётных линий определяется сигналом "RA0". В новом графическом режиме адрес начала первой области совпадает с прежней границей видеопамяти B8000, а начало второй области отстоит от первой на 16 Кбайт и имеет адрес BC000. В графике высокого разрешения имеется 128 тысяч точек (640 * 200), причем для создания четырёхцветного изображения на каждую точку требуется два бита памяти. Таким образом, нулевая линия начинается с адреса B8000, первая - с BC000, вторая - с B80A0 и т.д. Для правильной работы контроллера необходимо записать во внутренние регистры R0 - R15 следующие значения:

71, 50, 5A, 0A, 7F, 06, 64, 70, 02, 01, 06, 07, 00, 00, 00, 00.

Обращение к этим регистрам происходит через два порта. Чтобы загрузить какой-нибудь из регистров, надо сначала записать в порт 3D4 номер этого регистра, а затем послать в порт 3D5 нужное значение. Переключение на новый режим работы осуществляется через порт 3D8. Туда нужно записать число 6B. Пример такого программирования содержится в приложении II.

Помимо нового графического режима расширение памяти можно использовать и в стандартных режимах работы как дополнительную память,

содержание которой при необходимости можно показать на экране. Для этого достаточно изменить содержание внутренних регистров контроллера R12/R13 (начало памяти) и R14/R15 (позиция курсора) на соответствующие значения. Так, в текстовом 80-колоночном режиме мы будем иметь 8 страниц вместо 4, а в 40-колоночном - 16 страниц. В стандартных графических режимах два изображения, записанные в разных банках памяти, также можно рассматривать как две различные страницы.

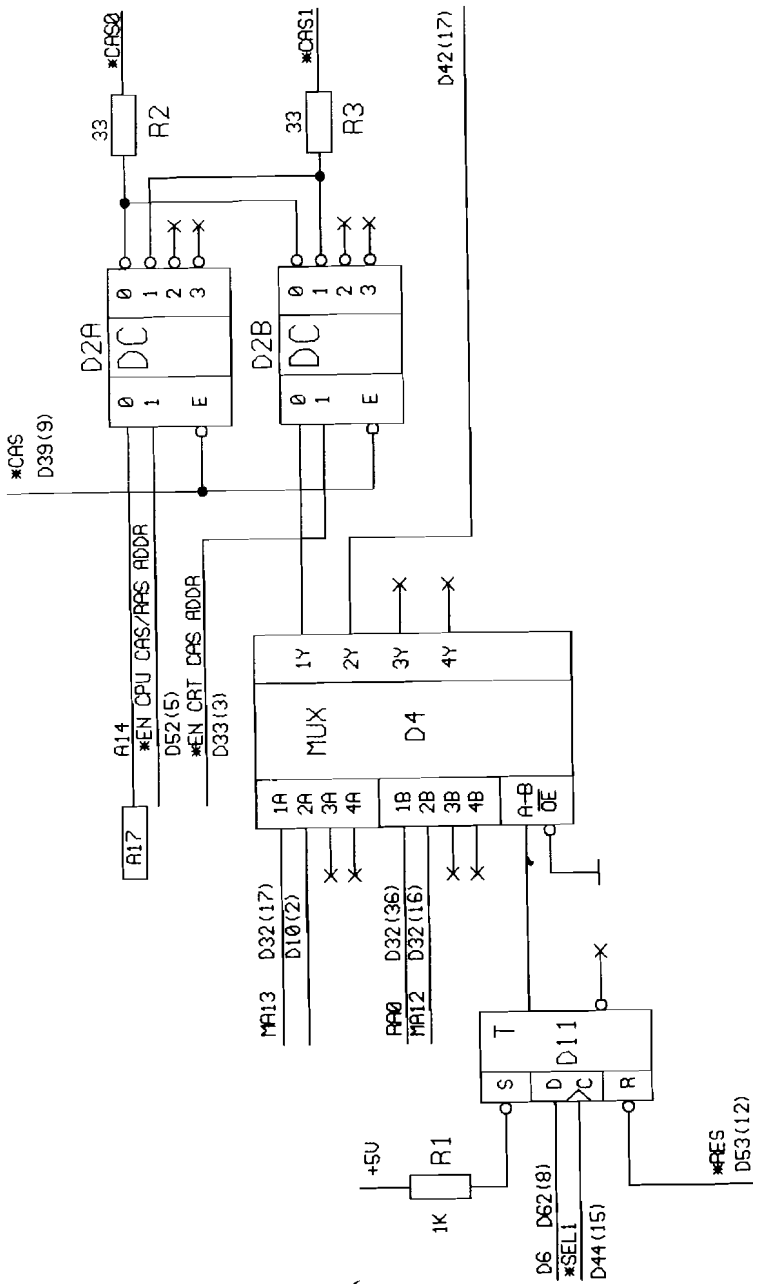
Стандартный BIOS для CGA адаптера, естественно, не поддерживает новый графический режим. Поэтому для этой цели было разработано программное обеспечение, которое позволяет работать в этом режиме стандартным образом через обращение к функциям BIOSa. Данное программное обеспечение реализовано в виде резидентной программы, работающей под операционной системой DOS. Текст программы представлен в приложении III. Для компьютеров "Правец-16" с модифицированным BIOSом (отсутствует EPROM 2764 с начальным адресом F4000) существует видоизменённый вариант данной программы для реализации этой функции, записанной в постоянную память компьютера, так как стандарт IBM PC-XT позволяет при загрузке автоматически инсталлировать новые драйверы.

В качестве конкретного использования нового цветного графического режима высокого разрешения был написан драйвер для системы PCAD. Для этого был определён протокол взаимодействия между системой PCAD и драйвером вывода информации на дисплей. В результате создана программа на языке ассемблер для IBM PC-XT/AT, которая может быть модифицирована и использована как драйвер для любого известного адаптера. Ввиду слишком большого её объёма текст программы в данной работе не представлен. При работе со стандартным драйвером для CGA адаптера, распространяемым с системой PCAD, поддерживается четырёхцветная графика среднего разрешения (320*200 точек). Новый драйвер позволяет в два раза повысить разрешающую способность по горизонтали, сохраняя при этом возможность выбора четырёх цветов из двух палитр. Заметим, что по-прежнему используются стандартные

устройства: обычный цветной дисплей и модифицированный CGA адаптер. Для видеоадаптера B440 написан также PCAD-драйвер, который поддерживает режим цветной графики 640*400 точек.

Автор выражает благодарность начальнику группы ремонта малых ЭВМ Карамышеву В.А. за предоставленную возможность работать на оборудовании группы, а также за плодотворные дискуссии и консультации.

ПРИЛОЖЕНИЕ I



ПРИЛОЖЕНИЕ II

```
;                       ПРИЛОЖЕНИЕ II
code    SEGMENT
        org     100h
        ASSUME  cs:code,ds:code
start:  JMP     a

par1    db      38h,28h,2dh,0ah,7fh,06h,64h,70h,02h,01h,06h,07h,0,0,0,0

par2    db      71h,50h,5ah,0ah,7fh,06h,64h,70h,02h,01h,06h,07h,0,0,0,0

figure  db      'fig.bin',0

a:      MOV     ax,3d00h
        MOV     dx,OFFSET figure
        INT     21h                     ;open file
        MOV     bx,ax                   ;save handle
        MOV     ah,3fh
        MOV     cx,4000h                ;number of byte for read
        MOV     dx,OFFSET pattern       ;DX points to buffer
        INT     21h                     ;read file

        MOV     ax,0b800h
        MOV     es,ax                   ;load ES with video area

        MOV     ah,11
        MOV     bx,100h
        INT     10h                     ;choose palette

; Display figure in medium-resolution mode.

        MOV     si,OFFSET par1          ;SI points to parameter table
                                        ; for medium-resolution mode
        MOV     al,2ah                  ; and AL mode value
        CALL    set                     ;set the mode
        MOV     di,0                    ;DI points to start address
        MOV     si,OFFSET pattern       ;SI points to figure
        MOV     cx,2000h                ;number of words
rep     MOVsw                           ;move figure in video area

        MOV     ah,0                    ;wait for key strike
        INT     16h

; Display figure in high-resolution mode.

        MOV     si,OFFSET par2          ;SI points to parameter table
                                        ; for high-resolution mode
        MOV     al,06bh                 ; and AL mode value
        CALL    set                     ;set the mode
        MOV     di,0                    ;DI points to start address
        MOV     si,OFFSET pattern       ;SI points to figure
        MOV     cx,64h                  ;number of odd lines
        CALL    fill                    ;make figure
        MOV     di,04000h               ;DI points to high address
                                        ;  of video memory
        MOV     si,OFFSET pattern[2000h];SI points to even lines
        MOV     cx,64h                  ;number of even lines
        CALL    fill                    ;make figure

        MOV     ah,0                    ;wait for key strike
        INT     16h
```

```
        MOV     ax,3                    ;return to text mode
        INT     10h

        MOV     ah,4ch                  ;terminate program
        INT     21h

set     PROC    NEAR
        PUSH    ax                      ;save mode value
        sub     ax,ax                   ;initialize index
r:      MOV     dx,03d4h
        out     dx,al                   ;load index register
        PUSH    ax
        lodsb                           ;get data
        INC     dx
        out     dx,al                   ;load data register
        POP     ax
        INC     ax
        cmp     al,10h                  ;is last register?
        jnz     r                       ;no, then repeat
        POP     ax                      ;restore mode value
        MOV     dx,3d8h
        out     dx,al                   ;load mode register
        ret
set     ENDp

fill    PROC    NEAR
l:      PUSH    cx                      ;save line number
        MOV     cx,28h                  ;number of words in line
                                        ; in medium-resolution mode
        PUSH    si                      ;SI points to start of line
rep     MOVsw
        POP     si                      ;restore SI
        MOV     cx,28h
rep     MOVsw                           ;repeat line
        POP     cx                      ;restore line number
        loop    l
        ret
fill    ENDp

pattern LABEL   BYTE

code    ENDs
        END     start
```



```
;                                       ПРИЛОЖЕНИЕ III
        name    newvideo_asm
;-----------------------------------------
; This routines provide the CRT interface for
; color graphics mode 8 with 640*200 PELs .
;-----------------------------------------
new_int         equ     0Ah
disable_video   equ     25h
enable_video    equ     6bh

ABS0    SEGMENT AT 0
        ORG     1FH*4
EXT_PTR LABEL   DWORD
ABS0    ENDS

DATA    SEGMENT AT 40H
        ORG     10H
EQUIP_FLAG      DW      ?
        ORG     49H
CRT_MODE        DB      ?
CRT_COLS        DW      ?
CRT_LEN         DW      ?
CRT_START       DW      ?
CURSOR_POSN     DW      8 DUP(?)
CURSOR_MODE     DW      ?
ACTIVE_PAGE     DB      ?
ADDR_6845       DW      ?
CRT_MODE_SET    DB      ?
CRT_PALETTE     DB      ?
DATA    ENDS

VIDEO_RAM       SEGMENT AT      0B800H
REGEN   LABEL   BYTE
REGENW  LABEL   WORD
        DB      8000H DUP(?)
VIDEO_RAM       ENDS

code    segment
        assume  cs:code,ds:data,es:video_ram
        org     100h
start:  JMP     init

M1      LABEL   WORD                    ;table of routines within
        dw      OFFSET set_mode         ; video i/o
        dw      OFFSET exit
        dw      OFFSET exit
        dw      OFFSET exit
        dw      OFFSET read_lpen
        dw      OFFSET exit
        dw      OFFSET graphics_up
        dw      OFFSET graphics_down
        dw      OFFSET graphics_read
        dw      OFFSET graphics_write
        dw      OFFSET graphics_write
        dw      OFFSET exit
        dw      OFFSET write_dot
        dw      OFFSET read_dot
        dw      OFFSET exit
        dw      OFFSET exit
M1L     EQU     $-M1
```

```
video_io        PROC    NEAR
        sti
        PUSH    ds
        CALL    dds
        or      ah,ah
        jnz     skip
        MOV     crt_mode,al
skip:   cmp     crt_mode,new_int
        jne     return
        cld
        push    es
        push    dx
        push    cx
        push    bx
        push    si
        push    di
        push    ax
        mov     al,ah
        xor     ah,ah
        shl     ax,1
        mov     si,ax                   ;put into SI for branch
        cmp     ax,M1L                  ;test for within range
        jb      M2
        pop     ax
        jmp     VIDEO_RETURN            ;do nothing if not in range
M2:
        mov     ax,0B800h               ;segment for video memory
        mov     es,ax
        pop     ax
        mov     di,EQUIP_FLAG           ;get equipment setting
        and     di,00030h
        cmp     di,030h                 ;is setting for bw card?
        je      exit                    ;yes, then redirect in BIOS

        jmp     word ptr cs:[si+OFFSET M1]

;---- redirect to oldvideo

exit:   POP     di
        POP     si
        POP     bx
        POP     cx
        POP     dx
        POP     es
return: POP     ds
        db      0eah                    ;jmp old_int10
old_INT10       dw      2 dup(?)
video_io        ENDp

CRT_CHAR_GEN    LABEL   DWORD           ;pointer to character generator
        dw      OFFSET table_char       ; for graphics mode
        dw      ?

table_char      LABEL   BYTE
 db   0,  0,  0,  0,  0,  0,  0,  0
 db 126,129,165,129,189,153,129,126
 db 126,255,219,255,195,231,255,126
 db 108,254,254,254,124, 56, 16,  0
 db  16, 56,124,254,124, 56, 16,  0
 db  56,124, 56,254,254,124, 56,124
 db  16, 16, 56,124,254,124, 56,124
 db   0,  0, 24, 60, 60, 24,  0,  0
```



```
db 255,255,231,195,195,231,255,255
db   0, 60,102, 66, 66,102, 60,  0
db 255,195,153,189,189,153,195,255
db  15,  7, 15,125,204,204,204,120
db  60,102,102,102, 60, 24,126, 24
db  63, 51, 63, 48, 48,112,240,224
db 127, 99,127, 99, 99,103,230,192
db 153, 90, 60,231,231, 60, 90,153
db 128,224,248,254,248,224,128,  0
db   2, 14, 62,254, 62, 14,  2,  0
db  24, 60,126, 24, 24,126, 60, 24
db 102,102,102,102,102,  0,102,  0
db 127,219,219,123, 27, 27, 27,  0
db  62, 99, 56,108,108, 56,204,120
db   0,  0,  0,  0,126,126,126,  0
db  24, 60,126, 24,126, 60, 24,255
db  24, 60,126, 24, 24, 24, 24,  0
db  24, 24, 24, 24,126, 60, 24,  0
db   0, 24, 12,254, 12, 24,  0,  0
db   0, 48, 96,254, 96, 48,  0,  0
db   0,  0,192,192,192,254,  0,  0
db   0, 36,102,255,102, 36,  0,  0
db   0, 24, 60,126,255,255,  0,  0
db   0,255,255,126, 60, 24,  0,  0
db   0,  0,  0,  0,  0,  0,  0,  0
db  48,120,120, 48, 48,  0, 48,  0
db 108,108,108,  0,  0,  0,  0,  0
db 108,108,254,108,254,108,108,  0
db  48,124,192,120, 12,248, 48,  0
db   0,198,204, 24, 48,102,198,  0
db  56,108, 56,118,220,204,118,  0
db  96, 96,192,  0,  0,  0,  0,  0
db  24, 48, 96, 96, 96, 48, 24,  0
db  96, 48, 24, 24, 24, 48, 96,  0
db   0,102, 60,255, 60,102,  0,  0
db   0, 48, 48,252, 48, 48,  0,  0
db   0,  0,  0,  0,  0, 48, 48, 96
db   0,  0,  0,252,  0,  0,  0,  0
db   0,  0,  0,  0,  0, 48, 48,  0
db   6, 12, 24, 48, 96,192,128,  0
db 124,198,206,222,246,230,124,  0
db  48,112, 48, 48, 48, 48,252,  0
db 120,204, 12, 56, 96,204,252,  0
db 120,204, 12, 56, 12,204,120,  0
db  28, 60,108,204,254, 12, 30,  0
db 252,192,248, 12, 12,204,120,  0
db  56, 96,192,248,204,204,120,  0
db 252,204, 12, 24, 48, 48, 48,  0
db 120,204,204,120,204,204,120,  0
db 120,204,204,124, 12, 24,112,  0
db   0, 48, 48,  0,  0, 48, 48,  0
db   0, 48, 48,  0,  0, 48, 48, 96
db  24, 48, 96,192, 96, 48, 24,  0
db   0,  0,252,  0,  0,252,  0,  0
db  96, 48, 24, 12, 24, 48, 96,  0
db 120,204, 12, 24, 48,  0, 48,  0
db 124,198,222,222,222,192,120,  0
db  48,120,204,204,252,204,204,  0
db 252,102,102,124,102,102,252,  0
db  60,102,192,192,192,102, 60,  0
db 248,108,102,102,102,108,248,  0
db 254, 98,104,120,104, 98,254,  0
```

```
db 254, 98,104,120,104, 96,240,  0
db  60,102,192,192,206,102, 62,  0
db 204,204,204,252,204,204,204,  0
db 120, 48, 48, 48, 48, 48,120,  0
db  30, 12, 12, 12,204,204,120,  0
db 230,102,108,120,108,102,230,  0
db 240, 96, 96, 96, 98,102,254,  0
db 198,238,254,254,214,198,198,  0
db 198,230,246,222,206,198,198,  0
db  56,108,198,198,198,108, 56,  0
db 252,102,102,124, 96, 96,240,  0
db 120,204,204,204,220,120, 28,  0
db 252,102,102,124,108,102,230,  0
db 120,204,224,112, 28,204,120,  0
db 252,180, 48, 48, 48, 48,120,  0
db 204,204,204,204,204,204,252,  0
db 204,204,204,204,204,120, 48,  0
db 198,198,198,214,254,238,198,  0
db 198,198,108, 56, 56,108,198,  0
db 204,204,204,120, 48, 48,120,  0
db 254,198,140, 24, 50,102,254,  0
db 120, 96, 96, 96, 96, 96,120,  0
db 192, 96, 48, 24, 12,  6,  2,  0
db 120, 24, 24, 24, 24, 24,120,  0
db  16, 56,108,198,  0,  0,  0,  0
db   0,  0,  0,  0,  0,  0,  0,255
db  48, 48, 24,  0,  0,  0,  0,  0
db   0,  0,120, 12,124,204,118,  0
db 224, 96, 96,124,102,102,220,  0
db   0,  0,120,204,192,204,120,  0
db  28, 12, 12,124,204,204,118,  0
db   0,  0,120,204,252,192,120,  0
db  56,108, 96,240, 96, 96,240,  0
db   0,  0,118,204,204,124, 12,248
db 224, 96,108,118,102,102,230,  0
db  48,  0,112, 48, 48, 48,120,  0
db  12,  0, 12, 12, 12,204,204,120
db 224, 96,102,108,120,108,230,  0
db 112, 48, 48, 48, 48, 48,120,  0
db   0,  0,204,254,254,214,198,  0
db   0,  0,248,204,204,204,204,  0
db   0,  0,120,204,204,204,120,  0
db   0,  0,220,102,102,124, 96,240
db   0,  0,118,204,204,124, 12, 30
db   0,  0,220,118,102, 96,240,  0
db   0,  0,124,192,120, 12,248,  0
db  16, 48,124, 48, 48, 52, 24,  0
db   0,  0,204,204,204,204,118,  0
db   0,  0,204,204,204,120, 48,  0
db   0,  0,198,214,254,254,108,  0
db   0,  0,198,108, 56,108,198,  0
db   0,  0,204,204,204,124, 12,248
db   0,  0,252,152, 48,100,252,  0
db  28, 48, 48,224, 48, 48, 28,  0
db  24, 24, 24,  0, 24, 24, 24,  0
db 224, 48, 48, 28, 48, 48,224,  0
db 118,220,  0,  0,  0,  0,  0,  0
db   0, 16, 56,108,198,198,254,  0

video_parms LABEL   BYTE
```



```
        db      71h,50h,5ah,0ah,7fh,06h,64h,70h,02h,01h,06h,07h,0,0,0,0

SET_MODE        PROC    NEAR
        mov     dx,003D4h
        mov     ADDR_6845,dx            ;save address of base
        push    dx
        add     dx,004h
        mov     al,disable_video
        out     dx,al                   ;reset video
        pop     dx
        MOV     bx,OFFSET video_parms   ;get pointer to video parms
        MOV     CX,10H                  ;number of parameters
        xor     ah,ah

;---- loop through table,outputting reg address, then value from table

M10:
        mov     al,ah
        out     dx,al
        inc     dx
        inc     ah
        mov     al,cs:[bx]
        out     dx,al
        inc     bx
        dec     dx
        loop    M10

;---- fill regen area with blank

        xor     di,di
        mov     CRT_START,di
        mov     ACTIVE_PAGE,000h
        mov     cx,04000h
        xor     ax,ax
rep     stosw

;---- enable video and correct port setting

        mov     CURSOR_MODE,00607h
        mov     dx,ADDR_6845
        add     dx,4
        mov     al,enable_video
        out     dx,al                   ;enable video
        mov     CRT_MODE_SET,al

;---- determine number of columns and buffer length

        mov     CRT_COLS,80
        mov     CRT_LEN,8000h

;---- set cursor positions

        mov     cx,00008h
        mov     di,OFFSET cursor_posn
        push    ds
        pop     es
        xor     ax,ax
rep     stosw

;---- set up overscan register

        inc     dx
```

```
        mov     al,030h
        out     dx,al
        mov     CRT_PALETTE,al

;---- normal return from newvideo

VIDEO_RETURN:
        pop     di
        pop     si
        pop     bx
        pop     cx
        pop     dx
        pop     es
        pop     ds
        iret
SET_MODE        ENDP

READ_DOT        PROC    NEAR
        call    R3
        mov     al,es:[si]
        and     al,ah
        shl     al,cl
        mov     cl,dh
        rol     al,cl
        jmp     VIDEO_RETURN
READ_DOT        ENDP

WRITE_DOT       PROC    NEAR
        push    ax
        push    ax
        call    R3
        shr     al,cl
        and     al,ah
        mov     cl,es:[si]
        pop     bx
        test    bl,080h
        jne     R2
        not     ah
        and     cl,ah
        or      al,cl
R1:
        mov     es:[si],al
        pop     ax
        jmp     VIDEO_RETURN
R2:
        xor     al,cl
        jmp     short   R1
WRITE_DOT       ENDP

; This subroutine determines the regen byte location.

R3      PROC    NEAR
        push    bx
        push    ax
        mov     al,80
        push    dx
        and     dl,0FEh
        mul     dl
        pop     dx
        test    dl,001h
        je      R4
        add     ax,04000h
```



```
R4:
        mov     si,ax
        pop     ax
        mov     dx,cx
        mov     bx,002C0h
        mov     cx,00302h
        and     ch,dl
        shr     dx,cl
        add     si,dx
        mov     dh,bh
        sub     cl,cl
R6:
        ror     al,1
        add     cl,ch
        dec     bh
        jne     R6
        mov     ah,bl
        shr     ah,cl
        pop     bx
        ret
R3      ENDP

GRAPHICS_UP     PROC    NEAR
        mov     bl,al
        mov     ax,cx
        call    GRAPH_POSN
        mov     di,ax
        sub     dx,cx
        add     dx,00101h
        shl     dh,1
        shl     dh,1
        shl     dl,1
        shl     di,1
        push    es
        pop     ds
        sub     ch,ch
        shl     bl,1
        shl     bl,1
        je      R11
        mov     al,bl
        mov     ah,0A0h
        mul     ah
        mov     si,di
        add     si,ax
        mov     ah,dh
        sub     ah,bl
R8:
        call    R17
        sub     si,4000H-160
        sub     di,4000H-160
        dec     ah
        jne     R8
R9:
        mov     al,bh
R10:
        call    R18
        sub     di,4000H-160
        dec     bl
        jne     R10
        jmp     VIDEO_RETURN
R11:
        mov     bl,dh
```

```
        jmp     short   R9
GRAPHICS_UP     ENDP

GRAPHICS_DOWN   PROC    NEAR
        std
        mov     bl,al
        mov     ax,dx
        call    GRAPH_POSN
        mov     di,ax
        sub     dx,cx
        add     dx,00101h
        shl     dh,1
        shl     dh,1
        shl     dl,1
        shl     di,1
        inc     di
        push    es
        pop     ds
        sub     ch,ch
        add     di,160*3
        shl     bl,1
        shl     bl,1
        je      R16
        mov     al,bl
        mov     ah,0A0h
        mul     ah
        mov     si,di
        sub     si,ax
        mov     ah,dh
        sub     ah,bl
R13:
        call    R17
        sub     si,040A0h
        sub     di,040A0h
        dec     ah
        jne     R13
R14:
        mov     al,bh
R15:
        call    R18
        sub     di,040A0h
        dec     bl
        jne     R15
        cld
        jmp     VIDEO_RETURN
R16:
        mov     bl,dh
        jmp     short   R14
GRAPHICS_DOWN   ENDP

;---- routine to move one row of information

R17     PROC    NEAR
        mov     cl,dl
        push    si
        push    di
rep     movsb
        pop     di
        pop     si
        add     si,04000h
        add     di,04000h
        push    si
```



```
        push    di
        mov     cl,dl
rep     movsb
        pop     di
        pop     si
        ret
R17     ENDP

;---- clear a single row

R18     PROC    NEAR
        mov     cl,dl
        push    di
rep     stosb
        pop     di
        add     di,04000h
        push    di
        mov     cl,dl
rep     stosb
        pop     di
        ret
R18     ENDP

GRAPHICS_WRITE  PROC    NEAR
        mov     ah,000h
        push    ax
        call    S26
        mov     di,ax
        pop     ax
        cmp     al,080h
        jnb     S1
        LDS     SI,CRT_CHAR_GEN
        jmp     short   S2
S1:
        sub     al,080h
        sub     si,si
        ASSUME  DS:ABS0
        mov     ds,si
        lds     si,EXT_PTR
        ASSUME  DS:DATA
S2:
        shl     ax,1
        shl     ax,1
        shl     ax,1
        add     si,ax
        mov     dl,bl
        shl     di,1
        call    S19
S8:
        push    di
        push    si
        mov     dh,004h
S9:
        lodsb
        call    S21
        and     ax,bx
        xchg    ah,al
        test    dl,080h
        je      S10
        xor     ax,es:[di]
S10:
        mov     es:[di],ax
```

```
        lodsb
        call    S21
        and     ax,bx
        xchg    ah,al
        test    dl,080h
        je      S11
        xor     ax,es:[di+4000H]
S11:
        mov     es:[di+4000H],ax
        add     di,0A0h
        dec     dh
        jne     S9
        pop     si
        pop     di
        inc     di
        inc     di
        loop    S8
        jmp     VIDEO_RETURN
GRAPHICS_WRITE  ENDP

GRAPHICS_READ   PROC    NEAR
        call    S26
        mov     si,ax
        sub     sp,008h
        mov     bp,sp
        push    es
        pop     ds
        shl     si,1
        mov     dh,004h
S14:
        call    S23
        add     si,04000h
        call    S23
        sub     si,4000H-160
        dec     dh
        jne     S14
        LES     DI,CRT_CHAR_GEN
        sub     bp,008h
        mov     si,bp
        cld
        mov     al,000h
S16:
        push    ss
        pop     ds
        mov     dx,00080h
S17:
        push    si
        push    di
        mov     cx,00008h
rep     cmpsb
        pop     di
        pop     si
        je      S18
        inc     al
        add     di,008h
        dec     dx
        jne     S17
        cmp     al,000h
        je      S18
        sub     ax,ax
        mov     ds,ax
        ASSUME  DS:ABS0
```



```
        les     di,EXT_PTR
        mov     ax,es
        or      ax,di
        je      S18
        mov     al,080h
        jmp     short   S16
        ASSUME  DS:DATA
S18:
        add     sp,008h
        jmp     VIDEO_RETURN
GRAPHICS_READ   ENDP

; This routine expands the low 2 bits in BL to BX.

S19     PROC    NEAR
        and     bl,003h
        mov     al,bl
        push    cx
        mov     cx,00003h
S20:
        shl     al,1
        shl     al,1
        or      bl,al
        loop    S20
        mov     bh,bl
        pop     cx
        ret
S19     ENDP

; This routine doubles all of the AL bits.

S21     PROC    NEAR
        push    dx
        push    cx
        push    bx
        sub     dx,dx
        mov     cx,00001h
S22:
        mov     bx,ax
        and     bx,cx
        or      dx,bx
        shl     ax,1
        shl     cx,1
        mov     bx,ax
        and     bx,cx
        or      dx,bx
        shl     cx,1
        jnb     S22
        mov     ax,dx
        pop     bx
        pop     cx
        pop     dx
        ret
S21     ENDP

; This routine will take 2 bytes from the regen buffer,
; compare against the current foreground color, and place
; the corresponding on/off bit pattern into the current
; position in save area.

S23     PROC    NEAR
        mov     ah,[si]
```

```
        mov     al,[si+001h]
        mov     cx,0C000h
        mov     dl,000h
S24:
        test    ax,cx
        clc
        je      S25
        stc
S25:
        rcl     dl,1
        shr     cx,1
        shr     cx,1
        jnb     S24
        mov     [bp],dl
        inc     bp
        ret
S23     ENDP

; This routine takes the cursor position contained
; in the memory location, and convert it into an
; offset into the regen buffer.

S26     PROC    NEAR
        mov     ax,CURSOR_POSN
GRAPH_POSN:
        push    bx
        mov     bx,ax
        mov     al,ah
        mul     BYTE PTR CRT_COLS
        shl     ax,1
        shl     ax,1
        sub     bh,bh
        add     ax,bx
        pop     bx
        ret
S26     ENDP

READ_LPEN       PROC    NEAR
        mov     ah,000h
        mov     dx,ADDR_6845
        add     dx,006h
        in      al,dx
        test    al,004h
        jne     V6
        test    al,002h
        jne     V7A
        jmp     V7
V7A:
        mov     ah,010h
        mov     dx,ADDR_6845
        mov     al,ah
        out     dx,al
        inc     dx
        in      al,dx
        mov     ch,al
        dec     dx
        inc     ah
        mov     al,ah
        out     dx,al
        inc     dx
        in      al,dx
        mov     ah,ch
```



```
        sub     ax,4
        mov     bx,CRT_START
        shr     bx,1
        sub     ax,bx
        jns     V2
        sub     ax,ax
V2:
        mov     dl,050h
        div     dl
        mov     ch,al
        add     ch,ch
        mov     bl,ah
        sub     bh,bh
        mov     cl,3
        shl     bx,cl
        mov     dl,ah
        mov     dh,al
        shr     dh,1
        shr     dh,1
        mov     ah,001h
V6:
        push    dx
        mov     dx,ADDR_6845
        add     dx,007h
        out     dx,al
        pop     dx
V7:
        pop     di
        pop     si
        pop     ds
        pop     ds
        pop     ds
        pop     ds
        pop     es
        iret
READ_LPEN       ENDP

DDS     PROC    NEAR
        PUSH    AX
        MOV     AX,DATA
        MOV     DS,AX
        POP     AX
        RET
DDS     ENDP


 msg    db      13,10,'BIOS extension for graphics mode 0Ah is installed.'
        db      13,10,'(c) Copyright M.Chizhov 1988.',13,10,'$'

;--------------------------------------------------------------
; Initial procedure.
;--------------------
        ASSUME  DS:CODE
INIT:

;---- initialization for BIOS extension

        MOV WORD ptr crt_char_gen[2],ds
        MOV     AX,3510H
        INT     21H
        MOV     OLD_INT10,BX
        MOV     OLD_INT10[2],ES
```

```
        MOV     AX,2510H
        MOV     DX,OFFSET VIDEO_IO
        INT     21H

        MOV     dx,OFFSET msg           ;print message
        MOV     ah,9
        INT     21h

        INT     27h

code    ends

        end     START
```

Рукопись поступила в издательский отдел
27 декабря 1989 года.

НЕТ ЛИ ПРОБЕЛОВ В ВАШЕЙ БИБЛИОТЕКЕ?

Вы можете получить по почте перечисленные ниже книги, если они не были заказаны ранее.

Д13-84-63	Труды XI Международного симпознума по ядерной электронике. Братислава, Чехословакия, 1983.	4 р. 50 к.
Д2-84-366	Труды 7 Международного совещания по проблемам квантовой теории поля. Алушта, 1984.	4 р. 30 к.
Д1,2-84-599	Труды VII Международного семинара по проблемам физики высоких энергий. Дубна, 1984.	5 р. 50 к.
Д17-84-850	Труды III Международного симпозиума по избранным проблемам статистической механики. Дубна, 1984. (2 тома)	7 р. 75 к.
Д11-85-791	Труды Международного совещания по аналитическим вычислениям на ЭВМ и их применению в теоретической физике. Дубна, 1985.	4 р. 00 к.
Д13-85-793	Труды XII Международного симпозиума по ядерной электроннке. Дубна, 1985.	4 р. 80 к.
Д4-85-851	Труды Международной школы по структуре ядра. Алушта, 1985.	3 р. 75 к.
Д3,4,17-86-747	Труды V Международной школы по нейтронной физике Алушта, 1986.	4 р. 50 к.
—	Труды IX Всесоюзного совещания по ускорителям заряженных частиц. Дубна, 1984. (2 тома)	13 р. 50 к.
Д1,2-86-668	Труды VIII Международного семинара по проблемам физики высоких энергий. Дубна, 1986. (2 тома)	7 р. 35 к.
Д9-87-105	Труды X Всесоюзиого совещания по ускорителям заряженных частиц. Дубна, 1986. (2 тома)	13 р. 45 к.
Д7-87-68	Труды Международной школы-семинара по физике тяжелых ионов. Дубна, 1986.	7 р. 10 к.
Д2-87-123	Труды Совещания "Ренормгруппа - 86". Дубна, 1986.	4 р. 45 к.
Д4-87-692	Труды Международного совещания по теории малочастичных и кварк-адронных систем. Дубна, 1987.	4 р. 30 к.
Д2-87-798	Труды VIII Международного совещания по проблемам квантовой теории поля. Алушта, 1987.	3 р. 55 к.
Д14-87-799	Труды II Международного симпозиума по проблемам взаимодействия мюонов и пионов с веществом. Дубна, 1987	4 р. 20 к.
Д17-88-95	Труды IV Международного симпозиума по избранным проблемам статистической механики. Дубна, 1987.	5 р. 20 к.