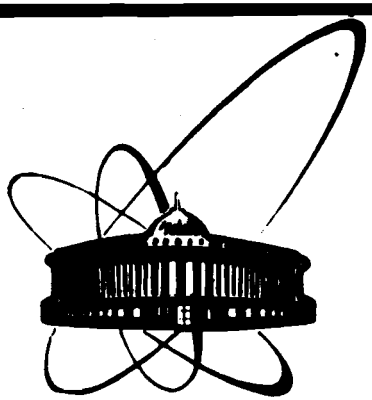

объединенный
институт
ядерных
исследований
дубна

Ч 596

Р11-88-264

М.В.Чижов

КУРСОР ДЛЯ ГРАФИЧЕСКОГО РЕЖИМА

Направлено в журнал "PC Magazine"

1988

Цветной графический адаптер фирмы IBM имеет два основных режима работы: текстовый и графический. В графическом режиме доступна любая точка экрана, поэтому некоторые экранные редакторы используют именно этот режим. Однако им приходится создавать видимый курсор, чтобы отмечать его положение на экране, так как в графическом режиме такой курсор не создается автоматически. Ситуация резко изменяется, если сервисная программа, подобная COMMAND.COM в MS-DOS, DEBUG, не обеспечивает этого. В таком случае использование расширенных редакторских возможностей становится трудной задачей.

Предположим, что надо убрать (или вставить) символ в командную строку. Без видимого курсора эта элементарная операция превращается в нелегкое испытание. Сначала следует подсчитать число позиций до символа, затем подвести невидимый (!) курсор на это место, сделать исправление и вернуться назад. Поэтому часто лучше набрать команду еще раз без ошибки, чем пользоваться расширенными редакторскими возможностями. Программа GRAPHCUR сделает курсор видимым и упростит редактирование в графическом режиме.

Здесь приведены программы на ассемблере (GRAPHCUR.ASM) и на БЕЙСИКе для создания исполнимого файла GRAPHCUR.COM . GRAPHCUR.ASM даст возможность более детально проследить логику программы и модифицировать её, но для этого потребуется Макро Ассемблер и следующие команды:

```
MASM GRAPHCUR;
LINK GRAPHCUR;
EXE2BIN GRAPHCUR GRAPHCUR.COM
```

Для ввода командной строки большинство программ используют сервисную функцию 10. При этом видимый курсор в графическом режиме так же необходим, как и в текстовом. Программа GRAPHCUR при ее вызове помещается резидентно в динамическую память, и каждый раз, когда вызывается сервисная функция 10, программа создает видимый курсор. Для этой цели используется прерывание 8. Подпрограмма NEW08 перехватывает прерывание 8 и проверяет, активна ли сервисная функция 10. Если это так, видимый курсор появляется на экране.

Использование прерывания 8 накладывает жесткое ограничение на скорость выполнения подпрограммы NEW08. Это ограничение будет руководящим принципом в дальнейшем для выбора метода получения текущей позиции курсора, его формы и метода записи на экран. Поэтому мы будем определять текущую позицию курсора, непосредственно читая ячейку памяти в области данных BIOS по абсолютному адресу 0040:0050h вместо вызова соответствующей сервисной функции.

Чтобы записать курсор на экран без использования сервисной функции, необходимо подсчитать его смещение в байтах от начала видеопамяти (адрес 0B8000h). Это позволит записать курсор непосредственно в видеопамять, что намного быстрее вызова соответствующего сервиса. Более того, вычисление смещения делается с помощью двоичных сдвигов вместо обычного умножения. Младший байт слова в ячейке памяти по адресу 0040:0050h содержит номер колонки текущей позиции курсора на экране, а старший - номер строки. Каждая строка текста в графическом режиме состоит из 8 линий. Колонка (или ширина символа) в режиме высокого разрешения соответствует одному байту в видеопамяти, а в режиме среднего разрешения - двум байтам или слову.

Информация в видеопамяти содержится в двух блоках. Один из них, при младших адресах, содержит байты четных линий, а другой, при старших адресах, - байты нечетных линий. Наш курсор будет всегда появляться точно под текущим рядом, т.е. в первой четной линии следующего. Это соответствует его записи в первый блок видеопамяти. Так как символы в графическом режиме представлены в виде блоков 8*8 точек, то, чтобы указать их положение на экране, курсор должен иметь длину в 8 точек, что соответствует байту в видеопамяти в режиме высокого разрешения и слову в режиме среднего разрешения. Такой выбор позволяет избежать исчезновения курсора на зоне символов. Даже, если набрано больше двух рядов, и надо сделать исправление в предыдущем, то возможна лишь небольшая интерференция, но не полное его исчезновение. Так, если бы он находился в нижней линии текущего ряда, тогда возможно полное его совпадение с символом подчеркивания.

Однако, имея 200 линий на экране, что соответствует 25 рядам, нам не хватает линии, чтобы отметить курсором положение символов в самом нижнем ряду. Действительно, для нижнего ряда уже не существует "первой линии следующего". Но это не беда, если мы имеем такой совершенный периферийный адаптер на базе микропроцессора Motorola

Объединенный институт ядерных исследований БИБЛИОТЕКА

6845. Он содержит 18 внутренних регистров, которые контролируют сканирование луча дисплея. Обращение к этим регистрам происходит только через два порта. Чтобы загрузить какой-нибудь из 18 регистров, сначала необходимо послать в порт 3D4h указатель на этот регистр (от 0 до 17), затем в порт 3D5h послать необходимую информацию. Число линий, показываемых на экране, записано во внутреннем регистре 6. При включении компьютера во время инициализации, туда заносится число 100, что соответствует 200 линиям на экране. Последовательно читая первый блок видеопамяти, затем второй, за два раза контроллер восстанавливает весь экран. Если мы запишем в регистр 6 число 101, то получим необходимую нам линию.

Поговорим теперь о методе записи курсора на экран. Мы могли бы поступить стандартно: сначала спасти то место экрана, куда хотим записать курсор, затем произвести запись. При изменении положения курсора необходимо восстановить прежнее содержимое памяти, и повторить процедуру сохранения и записи для нового положения. Однако вспомним о временных ограничениях и будем использовать как можно меньше медленных операций записи/чтения памяти.

"Запись" курсора на экран производит макро PRTCUR, которая видоизменяется для различных режимов разрешения. Она же производит и "стирание" курсора, используя логическое тождество NOT (NOT X) = X, тем самым восстанавливая прежнее содержимое памяти.

Эта программа может быть использована только для цветного графического адаптера. Она позволит легко редактировать командную линию в графическом режиме, так же как и в текстовом.

ПРИЛОЖЕНИЕ

Текст программы GRAPHCUR.COM на ассемблере.

```
;=============================================================================
; GRAPHCUR.COM for the IBM Personal Computer
; with Color/Graphics Monitor Adapter
; 1988 by Michail Chizhov

prtcur macro   disp
disp   db      ? , 15h          ;op code: not BYTE/WORD ptr ds:[di]
       ENDM
```



```
code   SEGMENT
       ASSUME cs:code
       org     100h                    ;for .COM file
start: JMP     initialize              ;go to initialize

program db     'GRAPHCUR  1.0'
        db     'Program by M.Chizhov  (c) 1988',1AH

msg     db     10,13,'Cursor for the graphics mode is installed.'
        db     10,13,'$'
        EVEN
discur dw      1ffeh                   ;initial value for
                                       ; cursor displacement
 flag   db     0                       ;flag for function 10 of INT 21h

;-----------------------------------------------------------------------
; The new entry for interrupt 21h.
;--------------------------------
 new21: PUSH    ax                     ;save AX register
        cmp     ah,10                  ;is it function 10?
        je      cont1                  ;yes, then continue
 exit:  POP     ax                     ;else restore AX
        db      0eah                   ; and exit through FAR JMP old21
 old21  dw      2 dup(?)

 cont1: sti                            ;enable interrupts
        PUSH    bx                     ;save BX
        MOV     ah,15                  ;get CRT mode
        INT     10h
        POP     bx                     ;restore BX
        cmp     al,3                   ;alphanumeric or graphics mode?
        jle     exit                   ;if alphanumeric,then exit
        test    al,2                   ;medium-resolution mode?
        MOV     cs:shift,0e0d0h        ;initilize op code: shl al,1
        MOV     al,0f7h                ; for 320 PELs in line
        jz      skip                   ;yes, then skip reinitialization
        MOV     cs:shift,9090h         ;else rewrite initialize code
        MOV     al,0f6h                ; for 640 PELs in line
```

```
skip:   MOV     cs:e0,al                ;initialize code for the
        MOV     cs:e1,al                ; corresponding resolution mode
        MOV     cs:e2,al

; Set 202 lines in graphics mode.

        PUSH    dx                      ;save DX
        cli                             ;disable interrupts
        MOV     dx,3d4h                 ;Index Register address of 6845
        MOV     al,6                    ;byte for port
        out     dx,al                   ;select register no. 6
        INC     dx                      ;Data Register address of 6845
        MOV     al,65h                  ;byte for port
        out     dx,al                   ;set graphics mode with 202 lines
        sti                             ;enable interrupts
        POP     dx                      ;restore registers
        POP     ax

        cli                             ;disable interrupts
        MOV     cs:flag,0ffh            ;activate cursor routine
        PUSHF                           ;recall INT 21h
        CALL DWORD ptr old21
        MOV     cs:flag,0               ;disactivate cursor routine

; Clear cursor before exit.

        PUSH    di                      ;save registers
        PUSH    ds
        MOV     di,0b800h               ;load DS with video data area
        MOV     ds,di
        MOV     di,cs:discur            ;point to cursor position
        prtcur  e0                      ;clear cursor
        MOV     cs:discur,1ffeh  ;initialize value for cursor displacement
        MOV     cs:poscur,0ffffh ;initialize value for cursor position
        POP     ds                      ;restore registers
        POP     di
        iret                            ;return
;-------------------------------------------------------------------
; New entry for interrupt 8.
```



```
;-------------------------
 new08: cmp     cs:flag,0       ;is function 10 of INT 21h requested?
        jne     cont2           ;yes, then continue
 exi08: db      0eah            ;no, then exit through FAR JMP old08
 old08  dw      2 dup(?)

 cont2: PUSH    ax              ;save registers
        PUSH    ds

; Get cursor position, if it has changed convert it into offset in bytes.

        MOV     ax,40h
        MOV     ds,ax           ;load DS with BIOS data area
        MOV     ax,ds:[50h]     ;get current cursor position
        db      3dh             ;op code: cmp ax,poscur
 poscur dw      0ffffh          ;initial value for cursor position
        je      ex08            ;no jump if position has changed
        mov     cs:poscur,ax    ;save new cursor position
        INC     ah              ;adjust next row
 shift  LABEL   WORD
        nop                     ;room for op code  shl al,1
        nop                     ; for medium-resolution mode
        PUSH    di              ;save pointer DI
        MOV     di,ax           ;save 256*AH+AL in DI
        xor     al,al           ;clear AL
        shr     ax,1            ;divide AX by 4
        shr     ax,1
        add     di,ax           ;DI points to displacement in bytes

; Print cursor.

        MOV     ax,0b800h       ;yes, then load DS with video data area
        MOV     ds,ax

        prtcur  e1              ;print cursor at new pos
        xchg    di,cs:discur    ;save new cursor displacement
        prtcur  e2              ;clear cursor at old position
        POP     di              ;restore pointer DI
 ex08:  POP     ds              ;save registers
```

```
        POP     ax
        JMP     exi08

;-------------------------------------------------------
; Initial procedure.
;--------------------

        ASSUME ds:code
initialize:
        MOV     ah,35h                  ;get the vector for INT 21h
        MOV     al,21h
        INT     21h
        MOV     old21,bx                ; and save it
        MOV     old21[2],es
        MOV     dx,OFFSET new21         ;set the new vector for INT 21h
        MOV     ah,25h
        MOV     al,21h
        INT     21h

        MOV     ah,35h                  ;get the vector for INT 8
        MOV     al,8
        INT     21h
        MOV     old08,bx                ; and save it
        MOV     old08[2],es
        MOV     dx,OFFSET new08         ;set the new vector for INT 8
        MOV     ah,25h
        MOV     al,8
        INT     21h

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

        MOV     dx,OFFSET initialize    ;terminate and stay resident
        INT     27h
code    ENDS
        END     start
;-------------------------------------------------------------------
```


Текст программы на Бейсике для создания GRAPHCUR.COM.

```
100 REM -- BASIC PROGRAM TO CREATE GRAPHCUR.COM
110 OPEN "GRAPHCUR.COM" AS #1 LEN = 1
120 FIELD #1,1 AS A$
130 CHECKSUM = 0
140 FOR I = 1 TO  43
150    LINESUM = 0
155    PRINT  ".";
160    FOR J = 1 TO 8
170       READ BYTE
180       CHECKSUM = CHECKSUM + BYTE
190       LINESUM = LINESUM + BYTE
200       IF (BYTE < 256) THEN LSET A$ = CHR$(BYTE)
210       PUT #1
220    NEXT J
230    READ LINECHECK
240    IF LINECHECK <> LINESUM THEN PRINT "Error in Line";280 + 10 * I
250 NEXT I
260 CLOSE
270 IF CHECKSUM =  33770! THEN PRINT "Successful Comletion!" : END
280 PRINT "COM file is not valid!" : END
290 DATA   233,    26,     1,    71,    82,    65,    80,    72,   630
300 DATA    67,    85,    82,    32,    32,    49,    46,    48,   441
310 DATA    80,   114,   111,   103,   114,    97,   109,    32,   760
320 DATA    98,   121,    32,    77,    46,    67,   104,   105,   650
330 DATA   122,   104,   111,   118,    32,    32,    40,    99,   658
340 DATA    41,    32,    49,    57,    56,    56,    26,    10,   327
350 DATA    13,    67,   117,   114,   115,   111,   114,    32,   683
360 DATA   102,   111,   114,    32,   116,   104,   101,    32,   712
370 DATA   103,   114,    97,   112,   104,   105,    99,   115,   849
380 DATA    32,   109,   111,   100,   101,    32,   105,   115,   705
390 DATA    32,   105,   110,   115,   116,    97,   108,   108,   791
400 DATA   101,   100,    46,    10,    13,    36,   254,    31,   591
410 DATA     0,    80,   128,   252,    10,   116,     6,    88,   680
420 DATA   234,     0,     0,     0,     0,   251,    83,   180,   748
430 DATA    15,   205,    16,    91,    60,     3,   126,   239,   755
440 DATA   168,     2,    46,   199,     6,   253,     1,   208,   883
```

```
450 DATA  224,  176,  247,  116,    9,   46,  199,    6, 1023
460 DATA  253,    1,  144,  144,  176,  246,   46,  162, 1172
470 DATA  200,    1,   46,  162,   15,    2,   46,  162,  634
480 DATA   22,    2,   82,  250,  186,  212,    3,  176,  933
490 DATA    6,  238,   66,  176,  101,  238,  251,   90, 1166
500 DATA   88,  250,   46,  198,    6,   96,    1,  255,  940
510 DATA  156,   46,  255,   30,  105,    1,   46,  198,  837
520 DATA    6,   96,    1,    0,   87,   30,  191,    0,  411
530 DATA  184,  142,  223,   46,  139,   62,   94,    1,  891
540 DATA    0,   21,   46,  199,    6,   94,    1,  254,  621
550 DATA   31,   46,  199,    6,  243,    1,  255,  255, 1036
560 DATA   31,   95,  207,   46,  128,   62,   96,    1,  666
570 DATA    0,  117,    5,  234,    0,    0,    0,    0,  356
580 DATA   80,   30,  184,   64,    0,  142,  216,  161,  877
590 DATA   80,    0,   61,  255,  255,  116,   34,   46,  847
600 DATA  163,  243,    1,  254,  196,  144,  144,   87, 1232
610 DATA  139,  248,   50,  192,  209,  232,  209,  232, 1511
620 DATA    3,  248,  184,    0,  184,  142,  216,    0,  977
630 DATA   21,   46,  135,   62,   94,    1,    0,   21,  380
640 DATA   95,   31,   88,  235,  198,  180,   53,  176, 1056
650 DATA   33,  205,   33,  137,   30,  105,    1,  140,  684
660 DATA    6,  107,    1,  186,   97,    1,  180,   37,  615
670 DATA  176,   33,  205,   33,  180,   53,  176,    8,  864
680 DATA  205,   33,  137,   30,  228,    1,  140,    6,  780
690 DATA  230,    1,  186,  219,    1,  180,   37,  176, 1030
700 DATA    8,  205,   33,  186,   47,    1,  180,    9,  669
710 DATA  205,   33,  186,   29,    2,  205,   39,    0,  699
```

Рукопись поступила в издательский отдел
21 апреля 1988 года

Чижов М.В. P1-88-264

Курсор для графического режима

Рассмотрены режимы работы графического адаптера персонального компьютера "Правец-16". На примере создания курсора в графическом режиме показаны возможности использования сервисных функций системного программного обеспечения: ввод строки, функции часов, сохранения пользовательской программы в памяти после завершения ее работы. Обсуждаются методы использования видеопамяти для непосредственной записи на экран.

Работа выполнена в Лаборатории теоретической физики ОИЯИ.

Препринт Объединенного института ядерных исследований. Дубна 1988

Перевод автора

Chizhov M.V. P1-88-264

A Cursor for the Graphics Mode

The "Pravetz-16" PC graphics adapter modes of operation are considered. Using the creation of a cursor in the graphics mode as an example, we demonstrate the possibilities of utilizing some system service functions: buffered keyboard input, clock function, making up a program resident after its termination. The methods of writing directly into the video memory are discussed.

The investigation has been performed at the Laboratory of Theoretical Physics, JINR.

Preprint of the Joint Institute for Nuclear Research. Dubna 1988