Support Group Application Note
Number: 051
Issue: 1
Author:

Acorn

Master 512:
Mouse Driver Demo Program

This program demonstrates how to interface to the Acorn Master 512 mouse.

Applicable Hardware : BBC Master 512

Related Application Notes:

Support Group
Acorn Computers Limited
Acorn House
Vision Park
Histon
Cambridge CB4 4AE

```
;This program claims the mouse, setting up an interrupt routine to note the mouse information as it comes
;in.  The main program prints out the current state of the mouse continually until both mouse buttons are
;depressed.  We then release the mouse and exit gracefully.

;  To produce a .EXE file use DRI tools
;
;  RASM86.EXE MOUSE
;  LINK86.EXE MOUSE
;

Cr                              equ     13
Lf                              equ     10

Bdos                            equ     21h

Print_string                    equ     9
Exit                            equ     4c00h

Mouse_button_exit               equ     5       ; both buttons down

OEM_Mouse                       equ     134
Claim_Mouse                     equ     0
Release_Mouse                   equ     1

XIOS_ENTRY                      equ     dword ptr .0028h
dos_plus_RLR                    equ     word ptr .4eh

cseg
        mov     dx, offset msg_header
        mov     ah, Print_string
        int     Bdos                            ; say hello

        call    init_xios_calls                 ;initialist the XIOS routine
        call    install_mouse                   ; install mouse interrupt handler
print_loop:
        call    display_info                    ;display state of the mouse
        cmp     cs: mouse_buttons, Mouse_button_exit
        jne     print_loop                      ; should we exit?

        call    remove_mouse                    ; remove mouse interrupt handler
        mov     ax, Exit                        ; and exit the program
        int     Bdos

;
;  This interrupt routine is called by Dos Plus with the current state of the mouse.  Save this information and
;  return with a RETF, all registers should be preserved.

i_mouse:
;-----------
; On entry      AX = mouse buttons
;               CX = mouse X coordinates
```

```
;               DX = mouse Y coordinates
;
; Note that you cannot make DOS system calls from within an interrupt safely, so the information is simply
; stored for display later
;
                mov             cs: mouse_buttons, ax
                mov             cs: mouse_x_loc, cx
                mov             cs: mouse_y_loc, dx
                retf

mouse_buttons           dw              0
mouse_x_loc             dw              0
mouse_y_loc             dw              0


display info:
;--------------
;
; Print mouse status in the form
; "X = nnnn Y = nnnn Buttons = nnnn"
;
                mov             dx, offset msg_x_coord
                mov             ah, Print_string
                int             Bdos
                mov             ax, cs: mouse_x_loc
                call            display_word

                mov             dx, offset msg_y-coord
                mov             ah, Print_string
                int             Bdos
                mov             ax, cs: mouse_y_loc
                call            display_word

                mov             dx, offset msg_buttons
                mov             ah, Print_string
                int             Bdos
                mov             ax, cs: mouse_buttons
                call            display_word

                mov             dx, offset msg_return
                mov             ah, Print_string
                int             Bdos
                ret


display_word:
;----------------
;
; Display the contents of AX in as a Hex word of the form "nnnn"
;
                push            ax
```

```
                mov     al, ah
                call    display_byte
                pop     ax
display_byte:
                push    ax
                mov     cl, 4
                shr     al, cl
                call    display_nibble
                pop     ax
display_nibble:
                and     al, 0fh
                mov     bx, offset bin_to_ascii
                xlat    al
                mov     dl, al
                mov     ah, 2
                int     21h
                ret


ini5_xios_calls:
;-----------------
                int     0feh                    ; get @sysdat address
                mov     sysdat, ax              ; and save for later
                ret


xios:
;-----
                push    ds                      ; we must first
                mov     ds, sysdat              ; point DS at the SYStem DATa
                push    es                        area
                mov     es, DOS_PLUS_rlr        ; point ES at User Data Area
                                                ; which is in Ready List Root
                callf   XIOS_ENTRY              ; and we can then call the XIOS
                pop     es
                pop     ds
                ret


install_mouse:
;----------------
                pushf                           ; turn off interrupts
                cli

                mov     ax, OEM_Mouse
                mov     cl, Clain_Mouse         ; take over the mouse
                mov     bx, cs                  ; BX:DX = address of our interrupt routine
                mov     dx, offset i_mouse
                call    xios

                popf                            ; interrupts OK now
```

```
                ret


remove_mouse:
;-----------------
                pushf                                   ; no interrupts
                cli
                mov             ax, OEM_Mouse
                mov             cx, Release_Mouse       ; release mouse
                call            xios
                popf
                ret


dseg

msg_header              db              'Mouse Demo Program', cr, lf, '$'
msg_x_coord             db              '  X = $ '
msg_y_coord             db              '  Y = $ '
msg_buttons             db              '  Buttons = $ '
msg_return              db              Cr, '$'

bin_to_ascii            db              '0123456789ABCDEF'

sysdat                  dw              0

sseg

                        rw              200             ; some stack for the program

end
```