ATA & ATAPI [DISK & CD-ROM DRIVE] ASSEMBLY PROGRAMMING

DIRECT I/O [TR-DOS project - CENTRAL.COM - P2002.COM i/o drafts]

ataid.html   ataping.zip   ataid.zip   atapi.zip

```
; *****************************************************************************
;
; ATAPINQ.ASM [ ATA & ATAPI device I/O code draft - ATAPI INQUIRY Command ]
; Copyright (C) 2002  Erdogan TAN  [ 20/11/2002 ]
; (Based on ATAID.ASM by Erdogan Tan & ATAPI Specification SFF-8020i Rev. 2.6)
;
; *****************************************************************************

; ATA/IDE Command Register Block [ AT Task File ]
IdeCmdReg_R_Data      equ 0  ; Data Register
IdeCmdReg_W_Data      equ 0  ; Data Register
IdeCmdReg_R_Error     equ 1  ; Error Register
IdeCmdReg_W_Feature   equ 1  ; Feature Register
IdeCmdReg_R_SectCount equ 2  ; Sector Count Register
IdeCmdReg_W_SectCount equ 2  ; Sector Count Register
IdeCmdReg_R_Sector    equ 3  ; Sector Number or LBA Bits 0-7
IdeCmdReg_W_Sector    equ 3  ; Sector Number or LBA Bits 0-7
IdeCmdReg_R_Cylinder0 equ 4  ; Cylinder Bits 0-7 or LBA Bits 8-15
IdeCmdReg_W_Cylinder0 equ 4  ; Cylinder Bits 0-7 or LBA Bits 8-15
IdeCmdReg_R_Cylinder1 equ 5  ; Cylinder Bits 8-15 or LBA Bits 16-23
IdeCmdReg_W_Cylinder1 equ 5  ; Cylinder Bits 8-15 or LBA Bits 16-23
IdeCmdReg_R_DriveHead equ 6  ; Drive & Head Bits or LBA Bits 24-27
IdeCmdReg_W_DriveHead equ 6  ; Drive & Head Bits or LBA Bits 24-27
IdeCmdReg_R_Status    equ 7  ; Status Register
IdeCmdReg_W_Command   equ 7  ; Command Register

; IDE Status Register Bits
IdeCmdReg_R_Status_BSY  equ 80h ; Bit 7
IdeCmdReg_R_Status_DRDY equ 40h ; Bit 6
IdeCmdReg_R_Status_DWF  equ 20h ; Bit 5
IdeCmdReg_R_Status_DSC  equ 10h ; Bit 4
IdeCmdReg_R_Status_DRQ  equ 08h ; Bit 3
IdeCmdReg_R_Status_CORR equ 04h ; Bit 2
IdeCmdReg_R_Status_IDX  equ 02h ; Bit 1
IdeCmdReg_R_Status_ERR  equ 01h ; Bit 0

; [ ATA Commands ]

; ATA PACKET INTERFACE Command
ATAPI_PKT_COMMAND equ 0A0h ; Mandatory
; ATAPI_IDENTIFY_DRIVE equ 0A1h ; Mandatory
; ATAPI_SOFT_RESET equ 08h ; Mandatory
; ATAPI_SERVICE equ A2h    ; Optional

; [ ATAPI Pkt Commands - as a parameter of ATA Command A0h ]
ATAPI_INQUIRY equ 12h ; Operation Code

; ATAPI INQUIRY DATA FORMAT
inquiry_peripheral_device_type equ 1Fh  ; Bit 0 to 4 of Byte 0
; Reserved Bits = Bit 5,6,7 of Byte 0
inquiry_removable              equ 80h  ; Bit 7 of Byte 1 (RMB bit)
inquiry_ANSI_version           equ 07h  ; Bit 0 to 2 of Byte 2
inquiry_ECMA_version           equ 38h  ; Bit 3 to 5 of Byte 2
inquiry_ISO_version            equ 0C0h ; Bit 6 & 7 of Byte 2
inquiry_response_data_format   equ 0Fh  ; Bit 0 to 3 of Byte 3
inquiry_ATAPI_version          equ 0F0h ; Bit 4 to 7 of Byte 3
; Additional Lenght (Bytes) = Byte 4  (Number of bytes following Byte 4)
; Reserved Bytes = Byte 5,6,7
; Vendor Identification = Byte 8 to 15
; Product Identification = Byte 16 to 31
; Product Revision Level = Byte 32 to 35
; Vendor Specific = Byte 36 to 55
; Reserved Bytes = Byte 56 to 95
; Vendor Specific Parameters = Byte 96 to n

; Parameter Offset values Of INQUIRY Data
; addr_inq_peripheral_device_type equ 0
offset_inq_removable        equ 1
offset_inq_standard_ver     equ 2
offset_inq_atapi_response   equ 3
offset_inq_additional       equ 4
offset_inq_vendor_id        equ 8
offset_inq_product_id       equ 16
offset_inq_product_rev      equ 32
offset_inq_vendor_spec      equ 36

; PERIPHERAL DEVICE TYPES
ptype_direct_access_device  equ 00h ; (e.g. magnetic disk)
; ptype_reserved_1 equal 01h
; ptype_reserved_2 equal 02h
; ptype_reserved_3 equal 03h
; ptype_reserved_4 equal 04h
ptype_cdrom_device          equ 05h
; ptype_reserved_6 equal 06h
ptype_optical_memory_device equ 07h
; Reserved device types = 08h to 1Eh
ptype_unknown               equ 1Fh

 Present segment Para 'code'

                assume CS:Present, DS:Present, ES:Present, SS:Present

;±±±±±±±±±±±±±±±±±±±±±±±±±±±±±±±±±±±±±±±±±±±±±±±±±±±±±±±±±±±±±±±±±±±±±±±±±±±±±±±
;±
```

```
;±                PROCEDURE proc_start
;±
;±±±±±±±±±±±±±±±±±±±±±±±±±±±±±±±±±±±±±±±±±±±±±±±±±±±±±±±±±±±±±±±±±±±±±±±±±±±±±±

proc_start        proc    far

                  org 100h
start:
                  push ds
                  pop es

                  mov byte ptr [Command], ATAPI_PKT_COMMAND

loc_inq_atapi_drive:

                  call proc_clear_screen

                  mov si, offset INQ_Table_Header
                  call proc_printmsg

                  mov word ptr [Port], 1F0h
                  mov byte ptr [Drive], 0
                  mov word ptr [INQ_T_Port], "F1"  ; 1F0h
                  mov byte ptr [INQ_T_Drive], "0"
                  call proc_atapi_inquiry

                  jc short pass_1F0_0

                  mov si, offset Msg_PressAnyKey
                  call proc_printmsg

                  xor ah, ah
                  int 16h

pass_1F0_0:
                  mov byte ptr [Drive], 10h ; Drive 1
                  mov byte ptr [INQ_T_Drive], "1"
                  call proc_atapi_inquiry

                  jc short pass_1F0_1

                  mov si, offset Msg_PressAnyKey
                  call proc_printmsg

                  xor ah, ah
                  int 16h

pass_1F0_1:
                  mov word ptr [Port], 170h
                  mov byte ptr [Drive], 0
                  mov word ptr [INQ_T_Port], "71"  ; 170h
```

```
                mov byte ptr [INQ_T_Drive], "0"
                call proc_atapi_inquiry

                jc short pass_170_0

                mov si, offset Msg_PressAnyKey
                call proc_printmsg

                xor ah, ah
                int 16h

pass_170_0:
                mov byte ptr [Drive], 10h ; Drive 1
                mov byte ptr [INQ_T_Drive], "1"
                call proc_atapi_inquiry

loc_terminate:
                int 20h

proc_start      endp

proc_atapi_inquiry proc near

                mov dx, ideCmdReg_R_Status
                add dx, word ptr [Port]

                mov cx, 0FFFFh
loc_read_status_reg_1:
                in al, dx
                and al, ideCmdReg_R_Status_BSY
                jz short loc_write_ide_command_1
                loop loc_read_status_reg_1

                jmp short loc_device_is_busy

loc_write_ide_command_1:
                mov dx, ideCmdReg_W_DriveHead
                add dx, word ptr [Port]
                mov al, byte ptr [Drive]
                or al, 0EFh ; Select Drive via Bit 4
                out dx, al
                mov cx, 0FFFFh
                mov dx, ideCmdReg_R_Status
                add dx, word ptr [Port]
loc_read_status_reg_2:
                in al, dx
                and al, 80h  ; BSY
                jz short loc_write_ide_command_2
                loop loc_read_status_reg_2

                jmp short loc_device_is_busy
```

```
loc_write_ide_command_2:
                mov dx, ideCmdReg_W_Command
                add dx, word ptr [Port]
                mov al, byte ptr [Command]
                out dx, al
                mov cx, 0FFFFh
                mov dx, ideCmdReg_R_Status
                add dx, word ptr [Port]
loc_read_status_reg_3:
                in al, dx
                test al, 80h ; BSY bit
                jnz short pass_drq_err_check_1
                test al, 01h ; ERR bit
                jnz short loc_ata_ide_io_error
                test al, 08h  ; DRQ bit
                jnz short loc_write_command_packet_1
pass_drq_err_check_1:
                loop loc_read_status_reg_3
                jmp short loc_device_is_busy

loc_write_command_packet_1:
                mov dx, ideCmdReg_R_Data
                add dx, word ptr [Port]
                mov si, offset Command_Packet_Buffer
                mov cx, 6
loc_write_command_packet_1a:
                lodsw
                out dx, ax
                loop loc_write_command_packet_1a

                mov cx, 0FFFFh
                mov dx, ideCmdReg_R_Status
                add dx, word ptr [Port]
loc_read_status_reg_4:
                in al, dx
                test al, 80h ; BSY bit
                jnz short pass_drq_err_check_2
                test al, 01h ; ERR bit
                jnz short loc_ata_ide_io_error
                test al, 08h  ; DRQ bit
                jnz short loc_read_data_reg_1a
pass_drq_err_check_2:
                loop loc_read_status_reg_4
loc_device_is_busy:
              ; mov si, Offset Device_is_busy
              ; call proc_printmsg

                stc

                retn
```

```
loc_read_data_reg_1a:
                mov dx, ideCmdReg_R_Data
                add dx, word ptr [Port]
                mov cx, 48
                mov di, offset Inquiry_Data_Buffer
                push di
loc_read_data_reg_1b:
                in ax, dx
                stosw
                loop loc_read_data_reg_1b
                pop si

                mov al, byte ptr [SI]  ; Peripheral Device Type at Offset 0.
                and al, inquiry_peripheral_device_type
                call proc_hex ; AL= Input, AX= Output as HEX num characters.
                mov word ptr [INQ_T_PDT], ax
                mov al, byte ptr [SI][offset_inq_removable] ; at Offset 1.
                and al, inquiry_removable
                jz short pass_RMB_Yes
                mov word ptr [INQ_T_RMB], "EY"
                mov byte ptr [INQ_T_RMB]+2, "S"
                jmp short pass_RMB_No

                ; This procedure is located here for "Short Jump"
loc_ata_ide_io_error:
              ; mov si, offset IO_Error
              ; call proc_printmsg

                stc

                retn

pass_RMB_Yes:
                mov word ptr [INQ_T_RMB], "ON"
                mov byte ptr [INQ_T_RMB]+2, 20h
pass_RMB_No:
                mov al, byte ptr [SI][offset_inq_standard_ver] ; at Offset 2.
                push ax
                and al, inquiry_ANSI_version
                add al, 30h
                mov byte ptr [INQ_T_ANSI_V], al
                pop ax
                push ax
                and al, inquiry_ECMA_version
                shr al, 1
                shr al, 1
                shr al, 1
                add al, 30h
                mov byte ptr [INQ_T_ECMA_V], al
                pop ax
                and al, inquiry_ISO_version
                shr al, 1
                shr al, 1
                shr al, 1
                shr al, 1
                shr al, 1
                shr al, 1
                add al, 30h
                mov byte ptr [INQ_T_ISO_V], al
                mov al, byte ptr [SI][offset_inq_atapi_response] ; Offset 3.
                push ax
                and al, inquiry_response_data_format
                add al,30h
                mov byte ptr [INQ_T_RDF], al
                pop ax
                and al, inquiry_ATAPI_version
                shr al, 1
                shr al, 1
                shr al, 1
                shr al, 1
                add al, 30h
                mov byte ptr [INQ_T_ATAPI_V], al

                mov al, byte ptr [SI][offset_inq_additional] ; at Offset 4.
                mov byte ptr [INQ_ADDL_Value], al
                call proc_hex ; AL= Input, AX= Output as HEX num characters.
                mov word ptr [INQ_T_ADDL], ax

                push si
                add si, offset_inq_vendor_id
                mov di, offset INQ_T_VENDOR_ID
                mov cx, 4
                rep movsw
                pop si
                push si
                add si, offset_inq_product_id
                mov di, offset INQ_T_PRODUCT_ID
                mov cx, 8
                rep movsw
                pop si
                push si
                add si, offset_inq_product_rev
                mov di, offset INQ_T_PRODUCT_REV
                movsw
                movsw
                pop si

                cmp byte ptr [INQ_ADDL_Value], 1Fh ; more than 31 bytes ?
                jna short loc_print_INQ_Data_Table
                add si, offset_inq_vendor_spec
                mov di, offset INQ_T_VENDOR_SPEC
                mov cx, 10
                rep movsw
                mov byte ptr [INQ_T_VS_Data_Ext], "V"
loc_print_INQ_Data_Table:
                mov si, offset INQ_Data_Table
                call proc_printmsg

                mov byte ptr [INQ_T_VS_Data_Ext], 0

                clc

                retn

proc_atapi_inquiry endp

proc_clear_screen proc near

                mov ah, 0Fh
                int 10h
                mov ah, 0
                int 10h

                retn

proc_clear_screen endp

proc_printmsg   proc near
loc_print:
                lodsb                           ; Load byte at DS:SI to AL
                and     AL,AL
                je      short loc_return        ; If AL = 00h then return
                mov     AH,0Eh
                mov     BX,07h
                int     10h                     ; BIOS Service func ( ah ) = 0Eh
                                                ; Write char as TTY
                                                ;AL-char BH-page BL-color
                jmp     short loc_print
loc_return:
                retn

proc_printmsg   endp

;'''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''';
; From binary (byte) to hexadecimal (character) converter    ;
;                                                             ;
; input -> AL = byte (binary number) to be converted         ;
; output -> AH = First character of hexadecimal number       ;
; output -> AL = Second character of hexadecimal number      ;
;                                                             ;
; (c) Erdogan TAN  1998 - 1999                                ;
;.............................................................;

; 1998
```

```
proc_hex        proc    near

                db 0D4h,10h                     ; Undocumented inst. AAM
                                                ; AH = AL / 10h
                                                ; AL = AL MOD 10h
                or AX,'00'                      ; Make it ZERO (ASCII) based

                xchg AH,AL

; 1999
                cmp AL,'9'
                jna short pass_cc_al
                add AL,7
pass_cc_al:
                cmp AH,'9'
                jna short pass_cc_ah
                add AH,7
pass_cc_ah:

; 1998
                retn

proc_hex        endp

Command:        db 0
Port:           dw 0
Drive:          db 0

; ATAPI INQUIRY Command Parameters (Input - Command packet)
Command_Packet_Buffer:
db 12h ; Operation Code
db 3 dup(0) ; Byte 1 to 3 are Reserved
db 60h ; BYTE 4 - Allocation Lenght = 96 bytes
db 7 dup(0) ; Byte 5 to 11 are Reserved

; ATAPI INQUIRY DATA Buffer
Inquiry_Data_Buffer:
db 96 dup(20h)

Msg_PressAnyKey:
                db 0Dh, 0Ah
                db "Press any key to continue ..."
                db 0Dh, 0Ah, 0

INQ_Table_Header:
                db 7
                db 0Dh, 0Ah
                db "ATAPI INQUIRY COMMAND OUTPUT  [ (c) Erdogan Tan 2002 ]"
                db 0Dh, 0Ah, 0
INQ_Data_Table:
                db 0Dh, 0Ah
                db "I/O Port                    : "
INQ_T_Port:     db "1F0h"
                db 0Dh, 0Ah
                db "Drive                       : "
INQ_T_Drive:    db "0"
                db 0Dh, 0Ah
                db 0Dh, 0Ah
                db "Peripheral Device Type      : "
INQ_T_PDT:
                db "00h  [ CD-ROM = 05h ]"
                db 0Dh, 0Ah
                db "Medium is Removable         : "
INQ_T_RMB:
                db "YES"
                db 0Dh,0Ah
                db "ANSI Version                : "
INQ_T_ANSI_V:
                db "0"
                db 0Dh,0Ah
                db "ECMA Version                : "
INQ_T_ECMA_V:
                db "0"
                db 0Dh,0Ah
                db "ISO Version                 : "
INQ_T_ISO_V:
                db "0"
                db 0Dh,0Ah
                db "Response Data Format        : "
INQ_T_RDF:
                db "0"
                db 0Dh, 0Ah
                db "Atapi Version               : "
INQ_T_ATAPI_V:
                db "0"
                db 0Dh,0Ah
                db "Additional Lenght           : "
INQ_T_ADDL:
                db "00h bytes"
                db 0Dh, 0Ah
                db "Vendor Identification       : "
INQ_T_VENDOR_ID:
                db 8 Dup(20h)
                db 0Dh, 0Ah
                db "Product Identification      : "
INQ_T_PRODUCT_ID:
                db 16 Dup(20h)
                db 0Dh, 0Ah
                db "Product Revision Level      : "
INQ_T_PRODUCT_REV:
                db 4 Dup(20h)
                db 0Dh, 0Ah
INQ_T_VS_Data_Ext:
```

```
                db 0 ; Will be replaced with "V" if Additional Bytes > 1Fh.
                db "endor Specific             : "
INQ_T_VENDOR_SPEC:
                db 20 Dup(20h)
                db 0Dh, 0Ah
                db 0Dh, 0Ah
end_of_table:
                db 0Dh, 0Ah,0
INQ_ADDL_Value:
                dw 0

; Drive_Is_Not_Ready:
              ; db "Drive is not ready !"
              ; db 0Dh, 0Ah, 0
; Device_Is_Busy:
              ; db "Device is busy !"
              ; db 0Dh, 0Ah, 0
; IO_Error:
              ; db "IO Error !"
              ; db 0Dh, 0Ah,0

Present         ends

                end  start
```

index.html specs.html trdos.html

Tarkan - KUZU KUZU