Note: This IBM PC BIOS API call is for DOS/Win16 personality only. Use Family API for portability.

Note: osFree Macro Library provides macros for most of functions

2022/03/13 05:41 · prokushev · 0 Comments

Int 13H, AH=02H

Version

IBM 5150 and higher

Brief

Read sector

Family API

DosRead

Input

- AH = 02h
- AL = number of sectors to read (must be nonzero)
- CH = low eight bits of cylinder number
- CL = sector number 1-63 (bits 0-5) high two bits of cylinder (bits 6-7, hard disk only)
- DH = head number
- DL = drive number (bit 7 set for hard disk)
- ES:BX → data buffer

Return

- CF set on error
  - if AH = 11h (corrected ECC error), AL = burst length
- CF clear if successful
- AH = status (see status)
- AL = number of sectors transferred (only valid if CF set for some BIOSes)

Macro

```
INCLUDE BIOS.INC

@DskRead  0, 0, 0, 1, 1, offset buf
```

Notes

- errors on a floppy may be due to the motor failing to spin up quickly enough; the read should be retried at least three times, resetting the disk with AH=00h between attempts
- most BIOSes support “multitrack” reads, where the value in AL exceeds the number of sectors remaining on the track, in which case any additional sectors are read beginning at sector 1 on the following head in the same cylinder; the MSDOS CONFIG.SYS command MULTITRACK (or the Novell DOS DEBLOCK=) can be used to force DOS to split disk accesses which would wrap across a track boundary into two separate calls
- the IBM AT BIOS and many other BIOSes use only the low four bits of DH (head number) since the WD-1003 controller which is the standard AT controller (and the controller that IDE emulates) only supports 16 heads
- AWARD AT BIOS and AMI 386sx BIOS have been extended to handle more than 1024 cylinders by placing bits 10 and 11 of the cylinder number into bits 6 and 7 of DH
- under Windows95, a volume must be locked (see INT 21/AX=440Dh/CX=084Bh) in order to perform direct accesses such as INT 13h reads and writes
- all versions of MS-DOS (including MS-DOS 7 [Windows 95]) have a bug which prevents booting on hard disks with 256 heads (FFh), so many modern BIOSes provide mappings with at most 255 (FEh) heads
- some cache drivers flush their buffers when detecting that DOS is bypassed by directly issuing INT 13h from applications. A dummy read can be used as one of several methods to force cache flushing for unknown caches (e.g. before rebooting).
- When reading from floppies, some AMI BIOSes (around 1990-1991) trash the byte following the data buffer, if it is not arranged to an even memory boundary. A workaround is to either make the buffer word aligned (which may also help to speed up things), or to add a dummy byte after the buffer.
- MS-DOS may leave interrupts disabled on return from this function.
- Apparently some BIOSes or intercepting resident software have bugs that may destroy DX on return or not properly set the Carry flag. At least some Microsoft software frames calls to this function with PUSH DX, STC, INT 13h, STI, POP DX.
- on the original IBM AT BIOS (1984/01/10) this function does not disable interrupts for harddisks (DL >= 80h). On these machines the MS-DOS/PC DOS IO.SYS/IBMBIO.COM installs a special filter to bypass the buggy code in the ROM (see CALL F000h:211Eh)

Note

Text based on Ralf Brown Interrupt List Release 61

IBM PC BIOS API	
Video I/O	**INT 10H**: 00H, 01H, 02H, 03H, 05H, 06H, 07H, 08H, 09H, 0AH, 0BH, 0CH, 0DH, 0EH, 0FH
Hardware info	INT 11H, INT 12H
Serial I/O	**INT 14H**: 00H, 01H, 02H, 03H
Tape I/O	**INT 15H**: 00H, 01H, 02H, 03H
Keyboard I/O	**INT 16H**: 00H, 01H, 02H
Printer I/O	**INT 17H**: 00H, 01H, 02H
Disk I/O	**INT 13H**: 00H, 01H, 02H, 03H, 04H, 05H
Date and Time	**INT 1AH**: 00H, 01H

osFree Macro Library	
Video I/O	@SetMode @SetCurSz @SetCurPos @GetCur @SetPage @ScrollUp @ScrollDn @Scroll @GetChAtr @PutChAtr @PutCh @SetPalet @SetColor @SetDot @GetDot @WrtTTY @VideoState @GetMode @GetDisplay @GetVideoState @GetEGAInfo @Cls
Hardware info	@Equipment @MemSize
Serial I/O	@AuxInit @AuxSendChar @AuxRecieveChar @AuxStatus
Tape I/O	@TapeOn @TapeOff @TapeRead @TapeWrite
Keyboard I/O	@KbdStatus @CharIn @CharPeek
Printer I/O	@PrnPrint @PrnInit @PrnStatus
Disk I/O	@DskReset @DskStatus @DskRead @DskWrite @DskVerify @DskFormat
Date and Time	@SetTime @GetTime
Mouse	@MouInit @MouShowPointer @MouStatus @MouSetPos @MouSetMickey @MouRegion
Memory manager	@ModBlok SET_BLOCK

2022/10/04 14:28 · prokushev · 0 Comments
2022/03/13 05:54 · prokushev · 0 Comments

Family API		
DOS	Process Manager	DosBeep DosExit DosSleep DosExecPgm
	File Manager	DosChDir DosChgFilePtr DosClose DosDelete DosDupHandle DosMkDir DosMove DosQCurDir DosQCurDisk DosSetFileMode DosOpen DosQFileInfo DosRead DosQFileMode DosQFSInfo DosQVerify DosRmDir DosSelectDisk DosFindClose DosFindFirst DosFindNext DosSetFileInfo DosSetVerify DosWrite DosFileLocks DosSetFHandState DosNewSize DosBufReset DosQFHandState DosSetFSinfo DosShutdown
	Memory Manager	DosFreeSeg DosSubAlloc DosSubFree DosSubSet DosAllocHuge DosAllocSeg DosReallocHuge DosReallocSeg DosGetHugeShift DosCreateCSAlias
	NLS	DosCaseMap DosGetCtryInfo DosGetDBCSEv DosSetCtryCode DosGetCollate DosGetMessage DosInsMessage DosPutMessage
	Date and Time	DosSetDateTime DosGetDateTime
	Devices	DosDevConfig DosDevIOCtl DosDevIOCtl2
	Signals	DosHoldSignal DosSetSigHandler
	Misc	BadDynLink DosGetEnv DosGetMachineMode DosGetVersion DosError DosErrClass DosSetVec
KBD		KbdCharIn KbdFlushBuffer KbdGetStatus KbdSetStatus KbdStringIn KbdPeek

Family API	
VIO	VioGetBuf VioGetConfig VioGetCurPos VioGetCurType VioGetPhysBuf VioReadCellStr VioReadCharStr VioScrollUp VioScrollDn VioScrollLf VioScrollRt VioScrUnLock VioSetCurPos VioSetCurType VioSetMode VioGetMode VioShowBuf VioWrtCellStr VioWrtCharStr VioWrtCharStrAtt VioWrtNAttr VioWrtNCell VioWrtNChar VioWrtTTY VioScrLock VioPopUp
Tools	BIND
Modules	DOSCALLS.DLL VIOCALLS.DLL KBDCALLS.DLL MSG.DLL
Libraries	API.LIB OS2386.LIB FAPI.LIB DOSCALLS.LIB SUBCALLS.LIB

2018/08/25 15:05 · prokushev · 0 Comments