i386disk.c

```
00001 /*
00002  *  FreeLoader
00003  *  Copyright (C) 1998-2003  Brian Palmer  <brianp@sginet.com>
00004  *
00005  *  This program is free software; you can redistribute it and/or modify
00006  *  it under the terms of the GNU General Public License as published by
00007  *  the Free Software Foundation; either version 2 of the License, or
00008  *  (at your option) any later version.
00009  *
00010  *  This program is distributed in the hope that it will be useful,
00011  *  but WITHOUT ANY WARRANTY; without even the implied warranty of
00012  *  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00013  *  GNU General Public License for more details.
00014  *
00015  *  You should have received a copy of the GNU General Public License along
00016  *  with this program; if not, write to the Free Software Foundation, Inc.,
00017  *  51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA.
00018  */
00019
00020 #include <freeldr.h>
00021
00022 #define NDEBUG
00023 #include <debug.h>
00024
00025 DBG_DEFAULT_CHANNEL(DISK);
00026
00028 // FUNCTIONS
00030
00031 BOOLEAN DiskResetController(UCHAR DriveNumber)
00032 {
00033     REGS    RegsIn;
00034     REGS    RegsOut;
00035
00036     WARN("DiskResetController(0x%x) DISK OPERATION FAILED -- RESETTING CONTROLLER\n", DriveNumber);
00037
00038     // BIOS Int 13h, function 0 - Reset disk system
00039     // AH = 00h
00040     // DL = drive (if bit 7 is set both hard disks and floppy disks reset)
00041     // Return:
00042     // AH = status
00043     // CF clear if successful
00044     // CF set on error
00045     RegsIn.b.ah = 0x00;
00046     RegsIn.b.dl = DriveNumber;
00047
00048     // Reset the disk controller
00049     Int386(0x13, &RegsIn, &RegsOut);
00050
00051     return INT386_SUCCESS(RegsOut);
00052 }
00053
00054 BOOLEAN DiskInt13ExtensionsSupported(UCHAR DriveNumber)
00055 {
00056     static UCHAR    LastDriveNumber = 0xff;
00057     static BOOLEAN    LastSupported;
00058     REGS    RegsIn;
00059     REGS    RegsOut;
00060
00061     TRACE("PcDiskInt13ExtensionsSupported()\n");
00062
00063     if (DriveNumber == LastDriveNumber)
00064     {
00065         TRACE("Using cached value %s for drive 0x%x\n", LastSupported ? "TRUE" : "FALSE", DriveNumber);
00066         return LastSupported;
00067     }
00068
00069     // Some BIOSes report that extended disk access functions are not supported
00070     // when booting from a CD (e.g. Phoenix BIOS v6.00PG and Insyde BIOS shipping
00071     // with Intel Macs). Therefore we just return TRUE if we're booting from a CD -
00072     // we can assume that all El Torito capable BIOSes support INT 13 extensions.
00073     // We simply detect whether we're booting from CD by checking whether the drive
00074     // number is >= 0x8A. It's 0x90 on the Insyde BIOS, and 0x9F on most other BIOSes.
00075     if (DriveNumber >= 0x8A)
00076     {
00077         LastSupported = TRUE;
00078         return TRUE;
00079     }
00080
```

```
00081     LastDriveNumber = DriveNumber;
00082
00083     // IBM/MS INT 13 Extensions - INSTALLATION CHECK
00084     // AH = 41h
00085     // BX = 55AAh
00086     // DL = drive (80h-FFh)
00087     // Return:
00088     // CF set on error (extensions not supported)
00089     // AH = 01h (invalid function)
00090     // CF clear if successful
00091     // BX = AA55h if installed
00092     // AH = major version of extensions
00093     // 01h = 1.x
00094     // 20h = 2.0 / EDD-1.0
00095     // 21h = 2.1 / EDD-1.1
00096     // 30h = EDD-3.0
00097     // AL = internal use
00098     // CX = API subset support bitmap
00099     // DH = extension version (v2.0+ ??? -- not present in 1.x)
00100     //
00101     // Bitfields for IBM/MS INT 13 Extensions API support bitmap
00102     // Bit 0, extended disk access functions (AH=42h-44h,47h,48h) supported
00103     // Bit 1, removable drive controller functions (AH=45h,46h,48h,49h,INT 15/AH=52h) supported
00104     // Bit 2, enhanced disk drive (EDD) functions (AH=48h,AH=4Eh) supported
00105     //        extended drive parameter table is valid
00106     // Bits 3-15 reserved
00107     RegsIn.b.ah = 0x41;
00108     RegsIn.w.bx = 0x55AA;
00109     RegsIn.b.dl = DriveNumber;
00110
00111     // Reset the disk controller
00112     Int386(0x13, &RegsIn, &RegsOut);
00113
00114     if (!INT386_SUCCESS(RegsOut))
00115     {
00116         // CF set on error (extensions not supported)
00117         LastSupported = FALSE;
00118         return FALSE;
00119     }
00120
00121     if (RegsOut.w.bx != 0xAA55)
00122     {
00123         // BX = AA55h if installed
00124         LastSupported = FALSE;
00125         return FALSE;
00126     }
00127
00128     if (!(RegsOut.w.cx & 0x0001))
00129     {
00130         // CX = API subset support bitmap
00131         // Bit 0, extended disk access functions (AH=42h-44h,47h,48h) supported
00132         DbgPrint("Suspicious API subset support bitmap 0x%x on device 0x%lx\n", RegsOut.w.cx, DriveNumber);
00133         LastSupported = FALSE;
00134         return FALSE;
00135     }
00136
00137     LastSupported = TRUE;
00138     return TRUE;
00139 }
00140
00141 VOID DiskStopFloppyMotor(VOID)
00142 {
00143     WRITE_PORT_UCHAR((PUCHAR)0x3F2, 0);
00144 }
00145
00146 BOOLEAN DiskGetExtendedDriveParameters(UCHAR DriveNumber, PVOID Buffer, USHORT BufferSize)
00147 {
00148     REGS    RegsIn;
00149     REGS    RegsOut;
00150     PUSHORT    Ptr = (PUSHORT)(BIOSCALLBUFFER);
00151
00152     TRACE("DiskGetExtendedDriveParameters()\n");
00153
00154     if (!DiskInt13ExtensionsSupported(DriveNumber))
00155             return FALSE;
00156
```

```
00157     // Initialize transfer buffer
00158     *Ptr = BufferSize;
00159
00160     // BIOS Int 13h, function 48h - Get drive parameters
00161     // AH = 48h
00162     // DL = drive (bit 7 set for hard disk)
00163     // DS:SI = result buffer
00164     // Return:
00165     // CF set on error
00166     // AH = status (07h)
00167     // CF clear if successful
00168     // AH = 00h
00169     // DS:SI -> result buffer
00170     RegsIn.b.ah = 0x48;
00171     RegsIn.b.dl = DriveNumber;
00172     RegsIn.x.ds = BIOSCALLBUFSEGMENT;    // DS:SI -> result buffer
00173     RegsIn.w.si = BIOSCALLBUFOFFSET;
00174
00175     // Get drive parameters
00176     Int386(0x13, &RegsIn, &RegsOut);
00177
00178     if (!INT386_SUCCESS(RegsOut))
00179     {
00180         return FALSE;
00181     }
00182
00183     memcpy(Buffer, Ptr, BufferSize);
00184
00185 #if DBG
00186     TRACE("size of buffer:                          %x\n", Ptr[0]);
00187     TRACE("information flags:                       %x\n", Ptr[1]);
00188     TRACE("number of physical cylinders on drive:   %u\n", *(PULONG)&Ptr[2]);
00189     TRACE("number of physical heads on drive:       %u\n", *(PULONG)&Ptr[4]);
00190     TRACE("number of physical sectors per track:    %u\n", *(PULONG)&Ptr[6]);
00191     TRACE("total number of sectors on drive:        %I64u\n", *(unsigned long long*)&Ptr[8]);
00192     TRACE("bytes per sector:                        %u\n", Ptr[12]);
00193     if (Ptr[0] >= 0x1e)
00194     {
00195         TRACE("EED configuration parameters:            %x:%x\n", Ptr[13], Ptr[14]);
00196         if (Ptr[13] != 0xffff && Ptr[14] != 0xffff)
00197         {
00198            PUCHAR SpecPtr = (PUCHAR)(ULONG_PTR)((Ptr[13] << 4) + Ptr[14]);
00199            TRACE("SpecPtr:                                 %x\n", SpecPtr);
00200            TRACE("physical I/O port base address:          %x\n", *(PUSHORT)&SpecPtr[0]);
00201            TRACE("disk-drive control port address:         %x\n", *(PUSHORT)&SpecPtr[2]);
00202            TRACE("drive flags:                             %x\n", SpecPtr[4]);
00203            TRACE("proprietary information:                 %x\n", SpecPtr[5]);
00204            TRACE("IRQ for drive:                           %u\n", SpecPtr[6]);
00205            TRACE("sector count for multi-sector transfers: %u\n", SpecPtr[7]);
00206            TRACE("DMA control:                             %x\n", SpecPtr[8]);
00207            TRACE("programmed I/O control:                  %x\n", SpecPtr[9]);
00208            TRACE("drive options:                           %x\n", *(PUSHORT)&SpecPtr[10]);
00209         }
00210     }
00211     if (Ptr[0] >= 0x42)
00212     {
00213         TRACE("signature:                               %x\n", Ptr[15]);
00214     }
00215 #endif
00216
00217     return TRUE;
00218 }
00219
00220 /* EOF */
```