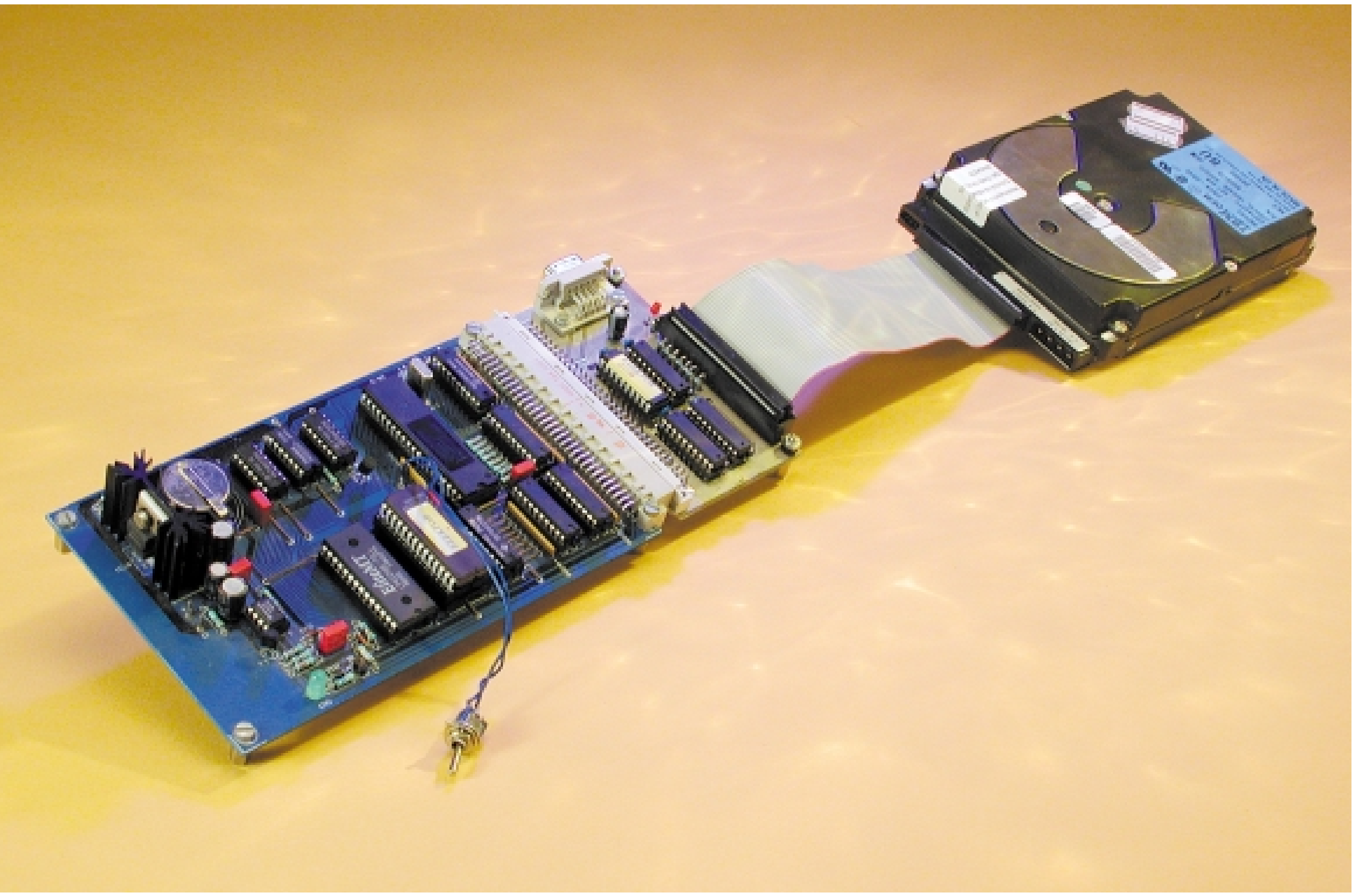
IDE Hard Disk Interface for 8-bit Controllers

controller seeks attachment with a hard disk…

Design by S. Schwark

Although 8-bit operation is included in the IDE interface, most hard disk drives do not support this mode. If you nonetheless want to control such drives with an 8-bit microcontroller, you will find the interface circuit described here to be essential.



Hard disk drives as mass storage devices are an indispensable part of modern computers. A few years ago, they were large and expensive devices, but presently (and up until the next generation) they are available with enormous capacities and small physical dimensions at quite reasonable prices. Most contemporary computers use hard disk drives with IDE interfaces. Since this interface (which is also referred to as ATA) comes from the PC world and thus works with a 16-bit operating system, it is possible to use it directly with 8-bit controllers, albeit with some problems. Although the interface can be switched to the 8-bit operating mode, this does not always work in practice, since many hard disk drives do not support this option. We are thus forced to provide a 16-bit data bus in the microcontroller circuit. This is precisely the task of the interface circuit described here. It has been developed to allow hard disk drives to be controlled by the 8032 Single Biard Computer, which was originally described in articles published in *Elektor Electronics* in 1991. Due to its relatively uncomplicated design, it can also be used relatively easily with other microprocessors.

The Integrated Disk Electronic Interface (IDE) was developed in 1984 by Compaq. The objective was to integrate the hard disk controller (which up to then had been a rather large expansion card) into the disk drive. The PC expansion card could then be made much simpler, since all that was needed was a connection to the system bus. The IDE interface is in principle nothing more than an AT slot, reduced to the leads required by the hard disk drive. This is why this interface can also be properly called ATA (AT Attachment).

A PC communicates with a hard disk drive via addresses 1F0h–1F7h and 3F0h–3F7h. There are also two CS (Card Select) leads, which select individual regions in the hard disk drive. The actual register selection takes place using three address lines. In the interface described here, different addresses are used. Both register banks lie in the address range C000h–C00Fh. The individual registers and their functions are described below under 'Software'.

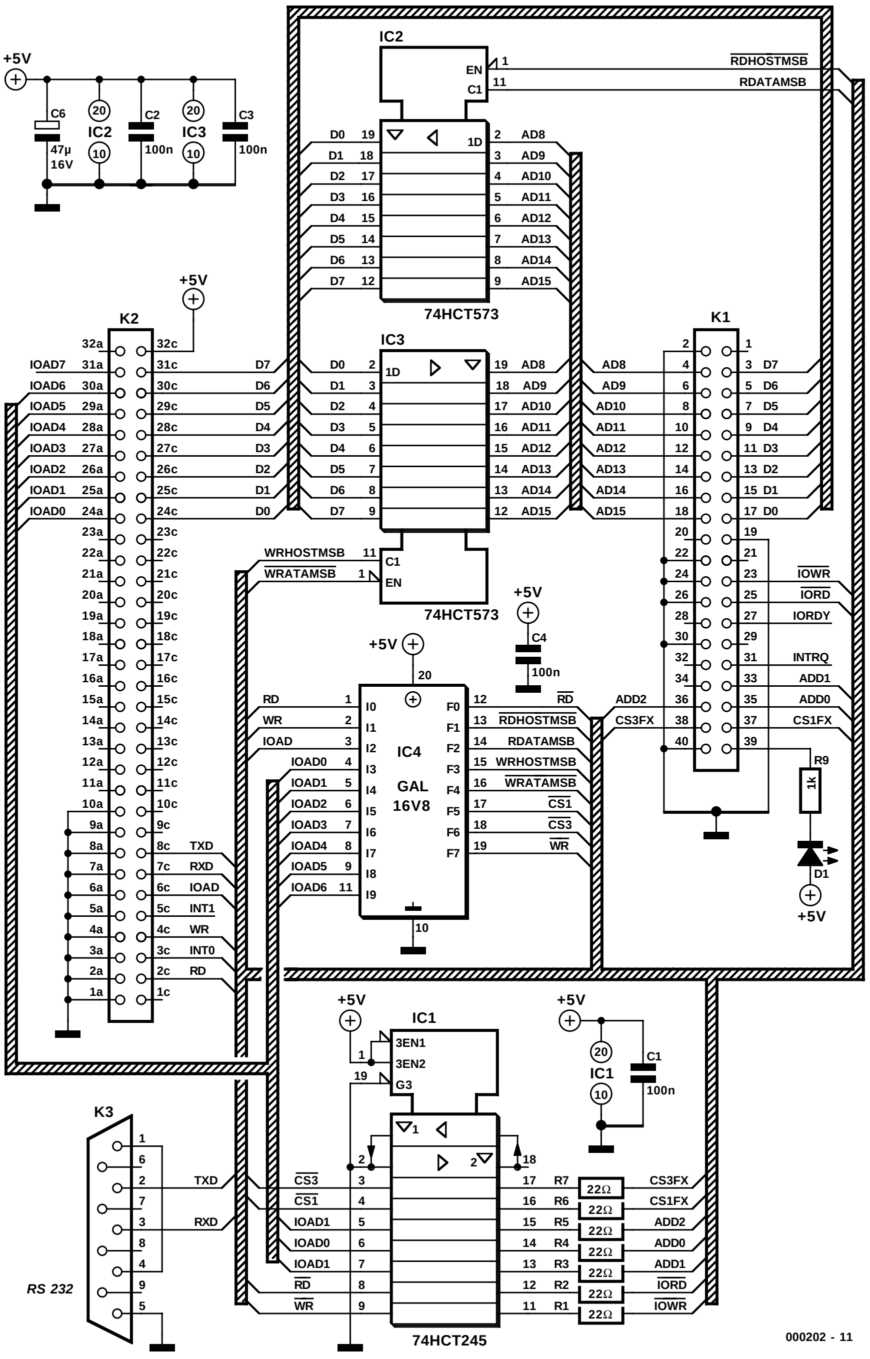


Figure 1. A GAL IC forms the heart of the interface circuit.

The ATA command set is used to communicate with the hard disk. This defines the codes that initiate the various functions (see **Table 1**). The commands for the hard disk are written to the command register at address 1F7h (corresponding to C00Fh). The individual commands are defined in the ATA command set specification. However, there are also vendor-specific variations, so some drives do not recognise some of the commands.

Since its original development, the ATA interface has achieved a considerable range of distribution, and today it is used for hard disks and CD-ROM drives in most PCs. Since the original specification was not suitable for devices such as CD-ROM drives, an extension was introduced a few years ago. The extended interface, which is called ATA Packet Interface (ATAPI), fully matches the ATA interface as far as the hardware is con-

COMPONENTS LIST

Resistors:
R1-R7 = 22Ω
R8 = 1kΩ

Capacitors:
C1-C4 = 100nF
C5 = 47µF 16V

Semiconductors:
D1 = high-efficiency LED, red
IC1 = 74HCT245
IC2,IC3 = 74HCT573
IC4 = GAL 16V8 (programmed, order code **000202-31**)

Miscellaneous:
K1 = 40-way boxheader with polarisation
K2 = 64-way DIN41612-C socket (female), angled pins, PCB mount
PCB, order code **000202-1**
Disk, order code **000202-11** (example program in C and GAL listing)

cerned. The transfer modes and registers are the same. While most normal ATA commands consist of only a command code that is written to the command register, ATAPI employs a 12-byte command packet. This structure of this data packet is strongly based on the SCSI interface, so it is understandable that IDE CD-ROM drives and streaming tape drives (ZIP) are actually SCSI devices that conduct their communications via a sort of translation step.

Hardware

Four ICs suffice for the hardware of the interface circuit. The heart of the circuit, as shown in **Figure 1**, is a GAL16V8 (IC4) that is used to drive latches IC2 and IC3. The source code for programming the GAL can be downloaded free of charge from the Free Downloads section on the *Elektor Electronics* Internet site, and it is also available on diskette from Readers Services (number **000202-11**). The GAL generates some of the control signals needed for the ATA interface, such as the two card select signals CS1Fx and CS3Fx. These two signals select the two register banks that are available in the hard disk drive. The designations of the two leads come from the PC world, where these registers are located at addresses 1F0h–1F7h and 3F0h–3F7h, respectively. The GAL of the interface circuit uses the address region C000h–C00Fh, into which all the registers are mapped. Individual registers are selected using address lines A0–A2. The individual registers are listed below under 'Software'.

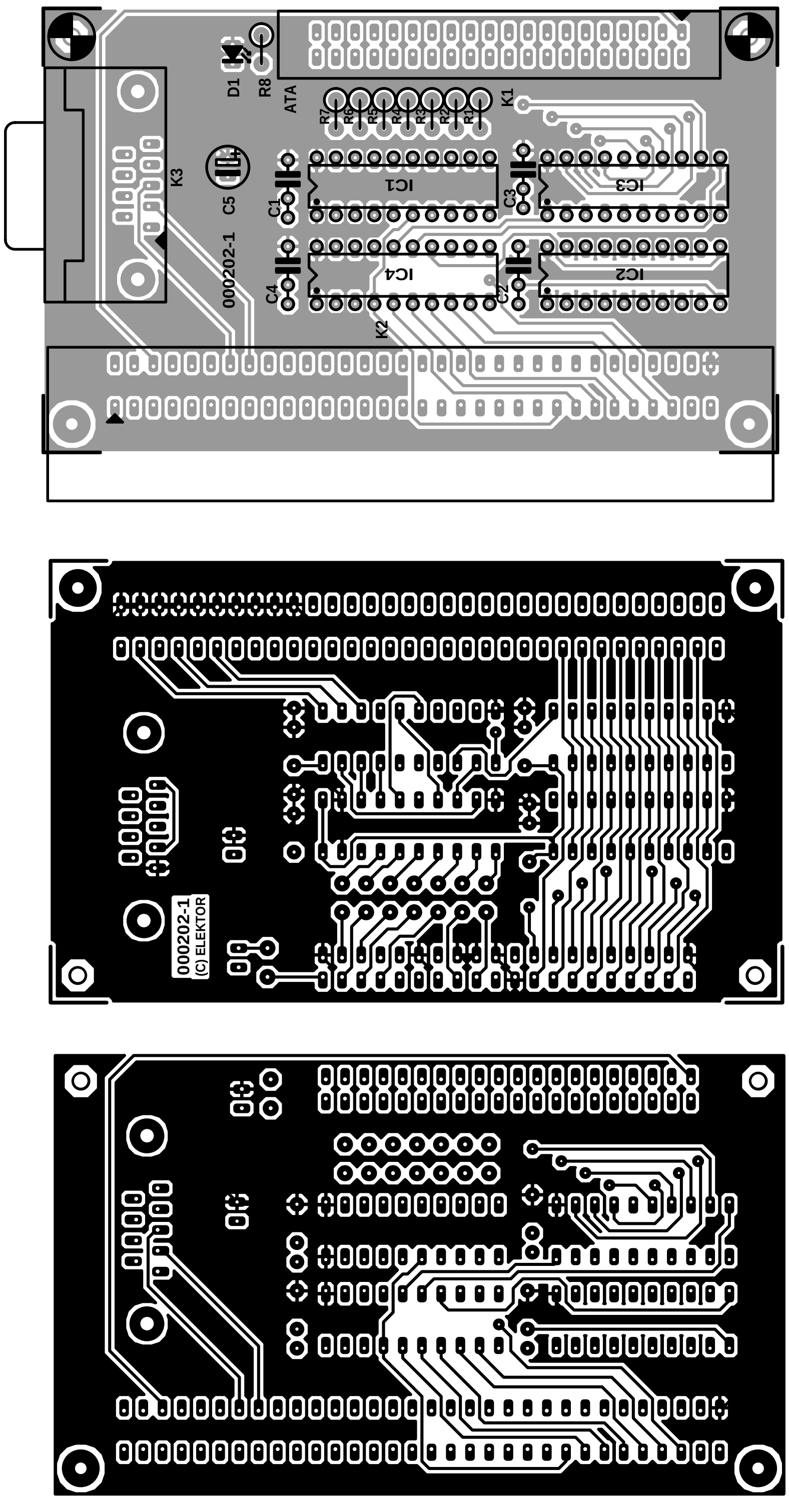


Figure 2. The track layout and component layout for the small interface circuit board.

The GAL also generates the two signals /WR and /RD by inverting the controller signals RD and WR. The remaining outputs control IC2 and IC3. IC2 is used for temporary storage of the MSB when the processor reads a data word from the bus. Whenever the host sends a read command via the ATA interface, IC2 loads the MSB from data lines D8–D15 into its memory. If the processor then reads address C000h, IC2 passes the most recently read

Table 1. ATA commands

Command	Code	FR	SC	SN	CY	DH
CHECK POWER MODE	E5h		x			D
EXECUTE DEVICE DIAGNOSTIC	90h					D
IDENTIFY DEVICE	Ech					D
IDLE	E3h		x			D
IDLE IMMEDIATE	E1h					D
INITIALIZE DEVICE PARAMETERS	91h		x			x
READ MULTIPLE	C4h		x	x	x	x
READ SECTOR(S)	20h 21h		x	x	x	x
READ VERIFY SECTOR(S)	40h 41h		x	x	x	x
SEEK	70h			x	x	x
SET FEATURES	Efh	x				D
SET MULTIPLE MODE	C6h		x			D
SLEEP	E6h					D
STANDBY	E2h		x			D
STANDBY IMMEDIATE	E0h					D
WRITE DMA	CAh CBh		x	x	x	x
WRITE MULTIPLE	C5h		x	x	x	x
WRITE SECTOR(S)	30h 31h		x	x	x	x

SC = Sector Counter C00Ah
SN = Sector Number register C00Bh
CY = Cylinder registers C00Ch and C00Dh
DH = Head Number register C00Eh
Note: 'D' means that only the device parameters in the register are significant.

MSB to the processor via the data bus. IC3 performs the same task when a write command is issued via the ATA interface. The host writes the relevant MSB to address C000h, which causes IC3 to transfer this byte into its memory. If the hard disk wants to read the data, the IC puts the MSB on data lines D8–D15. The final IC, IC1, is used to buffer the control signals that link IC4 and the processor to the ATA interface.
In addition to the 40-pin ATA connector (K1) and the 64-pin DIN4162-C connector for the link to the microcontroller board, there is a 9-pin Sub-D connector on the circuit board. The serial lines of the controller board are looped through this connector. Capacitors C1–C5 are the usual decoupling capacitors, and LED D1 indicates disk activity (the drive pulls this lead to ground when the disk is active). No distinction between master and slave drives is possible, since both drive signals use the same pin.

Table 2. Partition table entries

Offset address	Contents
00h	partition status (00h = inactive, 80h = boot partition)
01h	read/write head for the start of the partition
02h	sector and cylinder for the start of the partition
04h	partition type (e.g. 04h for DOS 16-bit FAT)
05h	read/write head for the end of the partition
06h	sector and cylinder for the end of the partition
08h	offset from the partition sector to the first sector of the partition (boot sector), in sectors
0Ch	number of sectors in this partition

Assembling the interface circuit board (**Figure 2**) should not present any problems. Sockets may be provided for the ICs. When soldering in the components, pay attention to the polarity of C6 and the LED, and naturally also to the orientation of the ICs.

Software

In order to drive the ATA interface directly from a microcontroller, you naturally need not only the necessary hardware, but also suitable software for writing the individual commands to the disk drive registers, which are described below. The Compuboard communicates with the ATA interface via the addresses that are listed after the register names. Some of the registers have different meanings if the disk drive is operated in LBA (Logical Block Addressing) mode. In this case, instead of addressing the sectors using cylinder, head and sector numbers, the individual disk sectors are numbered in a linear sequence. LBA was introduced to eliminate the ATA interface's capacity limit of 504 MB (you may also see 528 MB, but this is calculated using 1000 bytes for a kilobyte). The first sector has an address of '0'. With a 504-MB disk, the address of the final physical sector is '1,065,456'. Using this type of addressing, the capacity limit can be extended to 7.8 GB. If you want to use a hard disk that has a capacity of more than 504 GB, you thus have to operate the drive in LBA mode.

Data register C000h

Data bits 8–15 are read and written via this register.

Data register C008h

Data bits 0–7 are transferred via this register. The data are valid when the DRQ bit is set in the Status register.

Error Bits / Feature C009h

With a read access, the error bits are returned in this register. These bits can be used to determine the nature of any error that has occurred. With a write access, parameters are passed via this register (for example, cache on/off).

Sector Counter C00Ah

This register contains the number of sectors to be processed in the following drive access. Its meaning is the same for write and read accesses.

Start Sector C00Bh

The start sector for the subsequent read or write access can be written to this register. In LBA mode, LBA bits 0–7 are located here.

Cylinder LSB C00Ch

The lower eight bits of the cylinder address are located here. In LBA mode, this register contains LBA bits 8–15.

Cylinder MSB C00Dh

This contains either the upper eight bits of the cylinder address or LBA bits 16–23.

Drive / Head Number C00Eh

Bits 0–3 contain the binary-coded head number. In Normal mode, there can be up to 16 heads. Bit 4 distinguishes between the Master and Slave drives (1 = Slave, 0 = Master). Bits 5–7 contains the number of bytes per sector. A fixed value of 512 bytes/sector is entered here. In LBA mode, LBA bits 24–27 are located here.

Status / Command register C00Fh

The ATA commands are written to this register. For read accesses, this register is the Status register. The current status of the drive can be recognised from the individual bits:

Bit 0 (ERR): indicates an error. The nature of the error can be determined from the error bits at address 1F1h.
Bit 1 (IDX): index pulse (vendor-specific).
Bit 2 (CORR): indicates that the data being transferred have been corrected using ECC.
Bit 3 (DRQ): the device is ready for data transfer.
Bit 4 (DSC): the search command has been executed.
Bit 5 (DF): a write error has been detected.
Bit 6 (DRDY): drive ready (following switch-on, for example).
Bit 7 (BUSY): the drive is busy executing a command.

2nd Status register / Device Control C006h

A read access here yields the same result as an access to the first Status register, except that the state of the interrupt line is not changed. The interrupt can be activated by setting bit 1 to ‘0’ with a write access. A software reset of the disk drive can be initiated by setting bit 2 to ‘1’. Bits 3–7 are not used, and bit 0 must always be ‘0’.

Active Address C007h

This is a read-only register. Bits 0 and 1 distinguish between Master and Slave drives. The ones-complement value of the enabled head is located in bits 2–5. Drive write activity is indicated by a ‘0’ in bit 6. Bit 7 indicates to the PC that the diskette has been removed from the floppy disk drive. This bit is not used for our purposes.

Table 3. ATA interface connector pin assignments (K1)

Pin	Signal	Description
2,19,22,24,26,30,40	GND	ground
1	RES	reset signal from the computer
17	D0	D0–D12 form a bi-directional data bus between the hard disk drive and the computer. The data, status and control information are transferred via this bus. If the drive is not selected, these leads are switched to a high-impedance state.
15	D1	
13	D2	
11	D3	
9	D4	
7	D5	
5	D6	
3	D7	
4	D8	
6	D9	
8	D10	
10	D11	
12	D12	
14	D13	
16	D14	
18	D15	
20	Keying pin, cut off to provide protection against connecting the cable the wrong way around.	
21	/IOCHRDY	The computer must wait until this signal is inactive before accessing the drive.
23	/IOWR	write signal for I/O port addresses
25	/IORD	read signal for I/O port addresses
27	/IOCHRDY	same as pin 21
28	ALE	Address Latch Enable
29	frei	reserved for future extensions
31	IRQ14	interrupt request to the computer
32	IO16	indicator for 16-bit data transfers
34	/PDIAG	Passed Diagnostics: the Slave drive reports to the Master that the internal diagnostics have been completed.
33	A1	A0–A2 are address lines from the computer, used to select the individual registers
35	A0	
36	A2	
37	/CS0	addresses 1F0h through 1F7h
38	/CS1	addresses 3F0h through 3F7h
39	/ACT	Can be used to drive an LED to indicate drive activity.

In order to make programming somewhat easier, the author has written a small C program that demonstrates how a drive can be controlled. This program is also available via Internet and on the diskette. The program ATA.C first initialises the serial interface, since later on messages will have to be output via this interface. Next, an interface reset is triggered. If the drive then reports itself to be ready, the program sends an IDENTIFY DEVICE command to the hard disk drive. The hard disk then starts to output its internal data. The individual data items specify the model, version, serial number, number of heads and cylinders and so on. The data sent by the hard disk drive are output via the serial interface. Next, the first data block of the boot sector is read and also output. Here it is important to always read out the full 512 bytes, since otherwise problems can occur later on with read processes. This small example program provides a good indication of how easy it is to communicate with an IDE hard disk drive. There are also several MCS-51 assembler routines available, which you can incorporate into your own programs.

Hard disk organisation

In order to be able to find data that you have written to a hard disk, you naturally need some sort of structure. The first level of the hard disk structure is called 'low level formatting'. This consists of defining tracks on the surface of the disk, with multiple sectors per track. Each sector can hold 512 bytes. Following low level formatting, the hard disk is like an enormous diskette. This state would be adequate for a few applications. For example, if you only want to store data for an extended period, you could just write each sector in turn with 512-byte data packets. Normally, however, this sort of sequential utilisation of the disk is not a satisfactory solution.

A normal PC user does not have to be concerned with the exact operation of the hard disk drive. This task is looked after by the device drivers of the operating system or the BIOS. However, if you want to be able to access a disk using software that you have written yourself, there's no getting around the internal structure of the disk.

A hard disk contains one or more partitions. These partitions, which are also called 'volumes', are distinct, precisely demarcated regions of the hard disk. Their purpose is to make available different hard disk regions the can be managed by different operating systems.

After executing the memory test, running the test for various basic functions of the motherboard and initialising the graphics card, the BIOS loads the first sector of the hard disk drive (head 0, cylinder 0, sector 1), which is the partition sector. If the BIOS finds the code sequence '55AAh' in the final two bytes of the 512 sector bytes, it recognises the sector as executable code and starts program execution at address 0000:7000 with the first byte of the sector. The job of the code stored at this address is to detect the active partition and to execute the contents of the boot sector available in that partition. The routine can determine where to find this sector from the contents of the partition table (which normally starts at offset 01BEh in the partition sector). This table contains a 16-byte entry for each partition. The routine can determine which partition is the active boot partition from the first byte of the table entry, which is '80h' for an active partition and '00h' for an inactive partition. If no partition is active, or if more than one partition is active, the routine terminates execution.

The meaning of the partition table entries is listed in **Table 2**. Although these entries are not mandatory, they are present for practically every operating system. Of course, the operating system cannot do very much with the partition alone. This region must first be formatted. This amounts to arranging the physical sectors into logical regions (clusters), which the operating system in question can use coherently. The administration of these clusters depends on the operating system that is used. DOS uses the File Allocation Table (FAT). Each entry in this table designates one of these clusters. It contains information such as which file this cluster belongs to and which cluster contains the following portion of the file. You will have refer to the documentation for the operating system in question to find out how the FAT is used for reading and writing files, since it is not possible to deal with this subject within the confines of this article.

The content of byte 04h indicates the following to the operating system:

Byte 04h	Operating System
00h	unused partition
01h	DOS 12-bit FAT
04h	DOS 16-bit FAT
05h	extended partition
08h	OS/2
0Bh	FAT 32
41h	PowerPC boot
51h	Novell
64–69h	Novell
81h	Linux
83h	Linux swap
A5h	FreeBSD
A9h	NetBSD

Table 4. Memory structure of the ATA interface

Address	Read	Write
C000h	D8...D15	
C006h	alternate status	device control
C007h	drive address	
C008h	Data	
C009h	error	Feature
C00Ah	sector count	
C00Bh	sector number	
C00Ch	cylinder low	
C00Dh	cylinder high	
C00Eh	drive/head	
C00Fh	staus	Command
Status register		
D7	BUSY	device busy
D6	DRDY	device ready
D5	DF	devive fault
D4	DSC	device seek complete
D3	DRQ	data request
D2	CORR	correction
D1	IDX	vendor specific
D0	ERR	error
Device control register		
D7...D23		reseved
D2	SPST	soft reset
D1	NIEN	interrupt enable (active low)
D0	0	always 0

The capabilities of this interface circuit extend beyond just connecting a hard disk drive. Since the ATAPI CD-ROM interface employs the same register model, the circuit presented here can also be used for a CD-ROM drive. For these ATAPI devices, there is a series of special commands that can be used to activate the normal playback functions in a fairly straightforward manner. Maybe you can find a new use for that old 2x-speed CD-ROM in your junk box?

(000202-1)

You can find Internet addresses for information related to this subject at the Internet site of the T13 committee, which is responsible for the specification of the ATA standard:

http://www.t13.org