7. ATA INTERFACE

This laboratory work presents several versions of the ATA interface for disk drives and highlights the enhancements made by these versions to the original ATA interface. In addition, the laboratory work describes sector addressing, the transfer modes of the interface, the interface registers, the list of main commands, and command examples.

7.1. Overview of the ATA Interface

ATA (*AT Attachment*) is the most used interface for connecting disk drives to personal computers. The "*AT Attachment*" name originates from the fact that the interface was originally designed to connect a disk drive directly to the bus of an IBM PC/AT (*Advanced Technology*) computer, bus called ISA (*Industry Standard Architecture*) or AT. ATA is a 16-bit parallel interface. A serial version of this interface, called *Serial ATA* (SATA), has been introduced in 2000 and is used in computer systems starting from 2002.

The ATA interface is also called IDE, for the name of the first disk drives with this interface. The name IDE (*Integrated Drive Electronics*) refers to disk drives which have the controller built into the drive and not on a separate board, as with earlier interfaces. The assembly composed of the drive and controller is connected to one of the connectors on the motherboard of the computer.

The first disk drives that used the ATA interface have been produced in 1986 by Control Data Corporation (CDC), Western Digital (WD), and Compaq, which also established the signal assignment to the ATA connector pins. To eliminate incompatibilities and problems related to interfacing ATA drives to systems based on the ISA or EISA (*Extended ISA*) buses, in 1988 the *CAM* (*Common Access Method*) committee of the *ANSI* organization was formed. This committee developed the first version of the CAM ATA interface standard, and a draft version of this standard has been published in 1989.

Later on, the specifications of the parallel ATA interface have been developed and updated by an independent group that represented the major computer and disk drive manufacturers, the T13 Technical Committee (www.t13.org), which was part of the *InterNational Committee on Information Technology Standards* (*INCITS*). The standards developed by this committee have been approved and published by the *American National Standards Institute* (*ANSI*). The same committee was in charge with updating the ATAPI (*AT Attachment Packet Interface*) standards; this interface enables to connect optical disc drives through the same physical interface as ATA, but using a different logical protocol. Starting with the fourth version of the ATA standard, the ATAPI interface specifications have been included into the ATA standard. The last version of the ATA standard is *AT Attachment 8*, which has been released in 2008. For developing and updating the specifications of the SATA standard, a separate working group was formed, the *Serial ATA Workgroup* (www.serialata.org).

The ATA interface enables to connect in series two disk drives to an interface connector placed on the motherboard through a cable with three connectors: one for connecting to the motherboard, and two for connecting to the disk drives. Out of the two drives, one is the primary (master) drive, and the other is the secondary (slave) drive. Each disk drive has its own built-in controller, but the two drives use the same bus. For a correct operation, a single controller must respond to a command at a given time. Usually, this is ensured by properly positioning some jumpers on the two drives.

Personal computers contain two ATA interfaces integrated within the chipset, which allow connecting a number of up to four disk drives.

7.2. Evolution of the ATA Standards

From the introduction of the original version of the ATA standard, as the technology of the ATA interfaces in the industry improved, several versions of the ATA standards have been developed, which included the improvements appeared earlier into their specifications. Each version of the ATA standard is backward-compatible with the previous versions. This means that an older disk drive can be used with an ATA interface complying with a newer version of the standard. In general, the newer versions of the ATA standards can be considered as extensions of the previous versions.

Next, the main features of the various versions of the ATA standards are described.

ATA (ATA-1)

The original version of the ATA standard was officially approved by *ANSI* in 1994. This version, as the later versions, specifies a parallel interconnection which originates from the 16-bit ISA (AT) bus. The standard eliminated various incompatibility problems between the first generations of ATA/IDE disk drives, especially when two disk drives from different manufacturers were connected to the same ATA interface.

The original ATA standard defines the following features of the ATA interface:

- 40-pin or 44-pin connectors;
- A single ATA channel, which can be shared by two disk drives, configured as a master drive and a slave drive;
- The PIO (*Programmed Input/Output*) transfer modes 0, 1, and 2, with different timing characteristics and transfer rates;
- The single-word DMA (*Direct Memory Access*) transfer modes 0, 1, and 2;
- The multi-word DMA transfer mode 0;
- CHS (*Cylinder*, *Head*, *Sector*) addressing, which specifies the number of cylinder, head, and sector on the disk drive.

Although the original version of the ATA standard supported a maximum theoretical drive capacity of 128 GB in binary (137 GB in decimal[1]), the standard did not specify how to eliminate the 504 MB (528 MB in decimal) capacity barrier caused by the INT 13h programming interface of the BIOS program, because, at the time, no disk drives larger than 504 MB existed.

ATA-2

ATA-2 represents an extension of the standard for the original ATA interface, extension developed as a result of the technological improvements of disk drives and the increased demand for storage capacity. Published in 1996, the standard maintains compatibility with the original ATA interface and improves it, without the need to change the installed drives or the existing software.

The main improvements introduced by the ATA-2 standard are the following:

- Faster PIO transfer modes (PIO 3 and 4);
- Faster DMA transfer modes (DMA multiword 1 and 2);

[1] 1 GB in binary (also denoted by GiB) equals 2^{30} (1,073,741,824) bytes, while 1 GB in decimal equals 10^9 (1,000,000,000) bytes.

- Additional commands that allow block transfers (multiple words) to increase performance;
- Disk drives that support, optionally, *logical block addressing* (LBA) and BIOS programming interfaces that perform CHS parameter translation, in order to increase the addressable capacity of the disk drives up to 7.88 GB (8.46 GB in decimal);
- Improved *Identify Device* command, which enables the disk drive to report additional information required for "*Plug and Play*" systems and for compatibility with later revisions of the standard.

The ATA-2 standard was known under various names that represented marketing terms used by various companies, and not real standards. Thus, Seagate and Quantum used the names Fast ATA and Fast ATA-2 to refer to different portions of the ATA-2 standard. For instance, Fast ATA included support for PIO mode 3 and multiword DMA mode 1 transfers, while Fast ATA-2 included in addition support for PIO mode 4 and multiword DMA mode 2 transfers. Both variants supported block transfers and LBA logical addressing.

Western Digital used the name EIDE (*Enhanced* IDE) for its proposed extension of the ATA-2 standard. The main improvements were the introduction of an additional ATA channel, which uses a different interrupt and different addresses, and the possibility to connect optical discs or tape drives. Out of the enhanced transfer modes specified by the ATA-2 standard, EIDE included support for the PIO mode 3 or mode 4 transfers and for the multiword DMA mode 1 transfer.

ATA-3

The ATA-3 standard, published in 1997, is a minor revision of the ATA-2 standard. This revision did not define new high-performance transfer modes. The main changes introduced by the ATA-3 standard are the following:

- Eliminating single-word DMA transfer protocols;
- Adding S.M.A.R.T. (*Self-Monitoring, Analysis, and Reporting Technology*) support for prediction of disk drive failures;
- Adding a mode that allows protection of the data stored on disk drives with a password;
- Specifying the LBA addressing mode as being mandatory (this mode was optional in the ATA-2 standard);
- Recommendations for source and receiver bus termination in order to increase reliability in high-speed transfer modes.

S.M.A.R.T., initially developed by IBM, allows the operating system to monitor the operational parameters of a disk drive with the aim to detect its performance degradation. This degradation may progressively aggravate, finally leading to drive failure and data loss. By using this technology it is possible to predict the drive failure and to notify the user so that it can copy the data from the drive to another storage device in order to prevent data loss. However, S.M.A.R.T. does not allow predicting sudden failure of a disk drive.

ATA/ATAPI-4

Published in 1998, the ATA/ATAPI-4 standard introduced important changes to the previous version ATA-3. First, it included the ATAPI (ATA *Packet Interface*) protocol, which allows connecting devices such as optical disc drives to an ATA channel. Attaching these peripherals to the ATA interface was already possible before the ATA/ATAPI-4 version of the standard, but ATAPI was a separately published standard. A new command was added to the ATA command set, called *Packet*, which allows sending a data structure known as command packet to an ATAPI peripheral. The command set recognized by ATAPI peripherals is different

than that used by the ATA interface, and it is derived from the command set of the SCSI interface. The reason is that the ATA command set and register set are not adequate for some commands specific to optical drives.

The second important change introduced by the ATA/ATAPI-4 standard was the addition of a new transfer protocol called Ultra-ATA or Ultra-DMA (UDMA), in which data transfers take place on both edges of the clock signal. There are several Ultra-DMA transfer modes, out of which the ATA/ATAPI-4 specifications include modes 0, 1, and 2. For instance, mode 2 Ultra-DMA allows a maximum transfer rate of 33.3 MB/s, and for that reason this mode is also called Ultra-ATA/33 or UDMA/33. The possibility to use a particular mode is conditioned by the disk drive, by the chipset on the motherboard, and by the operating system or BIOS.

The main enhancements introduced by the ATA/ATAPI-4 standard are the following:

- Including the *Packet* command and the corresponding protocol for sending the ATAPI commands;
- Adding the Ultra-ATA protocol and modes 0, 1, and 2 that use this protocol, the maximum transfer rates reaching 33.3 MB/s;
- Increasing the data integrity by using a cyclic redundancy check (CRC);
- Defining an optional 80-conductor cable (out of which 40 are ground wires), which allows to increase noise immunity;
- Support for using a *Compact Flash* adapter for portable computers;
- Support for command overlapping (a new command can be sent before execution of previous commands completed) by implementing queues for storing the commands by the ATA and ATAPI peripherals.

ATA/ATAPI-5

This version of the ATA standard was approved in 2000. The main changes introduced by this version are the following:

- Adding the Ultra-DMA modes 3 and 4; mode 4 allows a maximum transfer rate of 66 MB/s (this mode is also called Ultra-ATA/66 or UDMA/66);
- Use of the 80-conductor cable is mandatory for operation in the UDMA/66 mode;
- Automatic detection of 40-conductor and 80-conductor cables;
- UDMA modes faster than UDMA/33 are enabled only if an 80-conductor cable is detected.

The UDMA/66 mode allows doubling the maximum transfer rate of the interface by reducing signal setup times and increasing the clock frequency. The higher clock frequency increases the possibility of interferences between signals when using the 40-conductor ATA cable. To eliminate interference and noise, the standard specifies that the use of the 80-conductor cable, defined as optional in the ATA/ATAPI-4 version, is mandatory for transfer modes starting with UDMA/66. This cable can be used with older disk drives as well, because it contains the same 40-pin connectors.

ATA/ATAPI-6

The development of this version of the ATA standard began in 2000 and the official standard was published in 2002. The main enhancements or changes compared to the previous version are the following:

- Adding the Ultra-DMA mode 5, which allows a maximum transfer rate of 100 MB/s (this mode is also called Ultra-ATA/100 or UDMA/100);

- Increasing the size of logical addresses from 28 bits to 48 bits;
- In the 48-bit LBA addressing, the size allocated for the number of sectors transferred by a single command increased from 8 bits (256 sectors or 128 KB) to 16 bits (65,536 sectors or 32 MB), which allows a more efficient transfer of large files;
- Disk drives must use LBA addressing and CHS addressing is declared obsolete;
- Adding new commands for *Audio/Visual* (AV) applications.

By extending the size of logical addresses to 48 bits, the number of addressable sectors increased from 2^{28} to 2^{48} (281,474,976,710,656 sectors). Thus, the maximum addressable capacity of disk drives increased significantly, from 128 GB (137 GB in decimal) to 128 PB (petabytes) or 144,115,188 GB in decimal. This extension became necessary because disks with capacity of over 137 GB appeared during 2001, but they were originally available only with the SCSI interface, which did not have the same limitation as the ATA interface.

ATA/ATAPI-7

This version of the ATA was published in 2005. The main enhancement is the addition of Ultra-DMA mode 6 (Ultra-ATA/133 or UDMA/133), which allows a maximum transfer rate of 133 MB/s. This version of the standard also includes the specifications of the SATA-150 serial interface, with a maximum transfer rate of 1.5 Gbits/s (150 MB/s).

AT Attachment 8

This is the last version of the ATA standard, published in 2008. For the parallel ATA interface, no enhanced transfer modes are introduced. For the SATA serial interface, this version includes the specifications of the SATA-300 interface, with a maximum transfer rate of 3 Gbits/s (300 MB/s).

7.3. Sector Addressing on ATA Disk Drives

There are two main methods to address sectors on an ATA disk drive. The first method is CHS addressing, in which three components are specified to address a sector: cylinder (track) number, read/write head number, and sector number. The second method is LBA addressing, in which a single logical address that is specific to the sector to be addressed is indicated. Starting with the ATA/ATAPI-6 version of the ATA/ATAPI interface, disk drives must use the LBA addressing.

Note

- Both CHS and LBA addresses represent logical addresses of the sectors. The disk drive will perform the conversion of the logical address into a physical address through a *translation* operation, which is specific to the drive.

CHS addressing was conceived based on the physical parameters of a disk drive, although a CHS address represents a logical address of the sector. Such an address is composed of three fields: cylinder number, head number, and sector number. Cylinders are numbered from 0 up to the maximum number allowed by the current translation mode, but the maximum number could not exceed 65,535. Heads are numbered from 0 to the maximum number allowed by the current translation mode, but the maximum number could not exceed 15. Sectors are numbered from 1 to the maximum number allowed by the current translation mode, but the maximum number could not exceed 255. These maximum values have been chosen somehow arbitrarily when the CHS addressing method has been developed, considering that they are enough for the disk drives at that time and for those of future generations.

When reading a disk drive sequentially in CHS mode, the process starts with cylinder 0, head 0, and sector 1. Next, all the remaining sectors on the same track are read; then, head 1 is selected and all the sectors on that track are read. Reading from cylinder 0 continues by

selecting the remaining heads, until the last one. Then, the next cylinder is selected and the process is repeated.

With LBA addressing, each sector on the disk drive is assigned a unique logical address. The logical sectors of the drive are allocated linearly; the first sector addressed in LBA mode (sector 0) is the same as the first logical sector addressed in CHS mode (cylinder 0, head 0, and sector 1). The allocation continues up to the last physical sector. Regardless of the current CHS translation mode, the LBA address of a given logical sector does not change. The following equation can be used for converting the CHS parameters into an LBA address:

$$LBA = ((C \times heads\_per\_cylinder + H) \times sectors\_per\_track) + S - 1$$

where *C*, *H*, and *S* represent the cylinder number, head number, and sector number, respectively, while *heads_per_cylinder* and *sectors_per_track* represent the values for the current translation mode.

7.4. Data Transfer Modes

The ATA/ATAPI interface specifications define two categories of data transfers: programmed transfers (PIO – *Programmed Input/Output*) and direct memory access transfers (DMA – *Direct Memory Access*). For each category, several transfer modes are defined, and each mode is characterized by a particular cycle time. These cycle times determine the maximum transfer rates that can be achieved.

7.4.1. PIO Transfer Modes

PIO transfer modes are less efficient, because for each word transferred the processor has to execute a program sequence. There are five PIO transfer modes, numbered 0 through 4. Depending on the protocol used, there are two types of PIO transfers: with no acknowledgement and with acknowledgement.

In the case of PIO transfers *with no acknowledgement*, the ATA/ATAPI interface does not have the confirmation that the computer's processor can accept the data from the disk drive. To minimize the risk of losing data when the processor is busy with other activities during the transfer of a data block, these transfers are executed at a low speed, irrespective of the capabilities of the computer. PIO modes 0, 1, and 2 use transfers with no acknowledgement.

In the case of PIO transfers *with acknowledgement*, the *IORDY* interface control signal is used. If necessary, the disk drive may assert this signal to extend the transfer cycle time and to delay the interface. Without using this signal, the transfer may be incorrect in the fast PIO modes. PIO modes 3 and 4 use transfers with acknowledgement.

In the slowest PIO transfer mode (mode 0), the cycle time cannot exceed 600 ns. In a single cycle, 16 bits (2 bytes) are transferred. Hence, in a second $2/600 \cdot 10^9$ bytes are transferred, and the maximum theoretical transfer rate is 3.33 MB/s. Table 7.1 presents the PIO modes and the maximum transfer rates allowed by these modes.

Table 7.1. PIO transfer modes of the ATA/ATAPI interface.

Mode	Cycle Time (ns)	Transfer Rate (MB/s)	Standard
PIO 0	600	3.33	ATA
PIO 1	383	5.22	ATA
PIO 2	240	8.33	ATA
PIO 3	180	11.11	ATA-2, IORDY required
PIO 4	120	16.67	ATA-2, IORDY required

The first three modes (0, 1, and 2) are also present in the original ATA standard. PIO modes 3 and 4 are specific to the ATA-2 and later standards. These modes use the *IORDY* signal for controlling the transfer.

To increase efficiency, block PIO transfers are used, which are initiated with the *Read/Write Multiple* commands. By using these commands, the number of interrupt requests to the host computer is decreased.

When interrogating a disk drive controller with the *Identify Device* command, it also returns information about the PIO and DMA modes supported. For example, bits 7..0 of word 64 indicates the advanced PIO modes that can be used. If bit 0 of this word is set to 1, the drive supports PIO mode 3, and if bit 1 is set to 1, the drive supports PIO mode 4. Bits 7..2 are reserved.

7.4.2. DMA Transfer Modes

Data transfers that are initiated through DMA commands, such as *Read DMA* and *Write DMA*, differ from PIO transfers in two aspects:

- Data transfers are performed via a DMA channel;
- A single interrupt request is generated at command completion.

Transfers performed through direct memory access are much more efficient than PIO transfers, because the processor is released from the burden to execute a program sequence for each word transferred. In addition, the processor can perform other operations while the data are transferred directly between the disk drive and main memory.

The ATA/ATAPI interface allows two types of DMA transfers: single-word and multiword. With single-word transfers, the computer initiates a transfer, selects the word to be transferred, and then the drive controller transfers that word. These operations must be repeated for each of the following words. These transfers have low efficiency, and for this reason they are no longer used.

The single-word DMA transfers are presented in Table 7.2.

Table 7.2. Single-word DMA transfers of the ATA/ATAPI interface.

Mode	Cycle Time (ns)	Transfer Rate (MB/s)	Standard
Single-word DMA 0	960	2.08	ATA
Single-word DMA 1	480	4.17	ATA
Single-word DMA 2	240	8.33	ATA

Note

- Single-word DMA transfers were removed from the ATA-3 and later standards.

Multiword DMA transfers enable to achieve higher performance. After the computer initiates a transfer, it selects the first word and the last word of the block to be transferred, and then the drive controller transfers all the block words. Table 7.3 presents the multiword DMA transfers.

Table 7.3. Multiword DMA transfers of the ATA/ATAPI interface.

Mode	Cycle Time (ns)	Transfer Rate (MB/s)	Standard
Multiword DMA 0	480	4.17	ATA
Multiword DMA 1	150	13.33	ATA-2
Multiword DMA 2	120	16.67	ATA-2

Depending on the control of the transfer operations, there are two types of DMA transfers: normal and bus-mastering. Normal transfers are performed by the system DMA controller. DMA transfers of the ATA/ATAPI interface are bus-mastering transfers; they are performed by the interface logic, which takes control of the bus and performs the transfer.

Word 63 of the data block returned by the *Identify Device* command indicates the multiword DMA transfer modes that are supported by the disk drive and the mode selected. Bits 0, 1, and 2 of this word indicate by value 1 that mode 0, 1, and 2 is supported, respectively. Bits 8, 9, and 10 of word 63 indicate by value 1 that mode 0, 1, and 2 is selected, respectively.

Starting with the ATA/ATAPI-4 version of the ATA standard, higher-performance DMA transfer modes have been introduced, called Ultra-DMA or Ultra-ATA. In these modes, data are transferred on both edges (rising and falling) of the clock signal used to control the data bus, so that the transfer rate is doubled. In addition, in the higher-performance variants of the Ultra-DMA modes, the frequency of the clock signal is higher compared to that of multi-word DMA modes.

To increase the reliability of transfers in Ultra-DMA modes, synchronous transfers are used instead of asynchronous transfers. The equipment that sends the data (the computer on writing, the disk drive on reading) generates the clock signal and synchronizes the data transfers with the clock signal. Since a single equipment controls both the clock signal and the data lines, the synchronization of the transfers is more precise.

The Ultra-DMA modes that have been introduced by the various versions of the ATA/ATAPI standard are presented in Table 7.4.

Table 7.4. Ultra-DMA transfer modes of the ATA/ATAPI interface.

Mode	Cycle Time (ns)	Transfer Rate (MB/s)	Standard
Ultra-DMA 0	240	16.67	ATA/ATAPI-4
Ultra-DMA 1	160	25	ATA/ATAPI-4
Ultra-DMA 2	120	33.33	ATA/ATAPI-4
Ultra-DMA 3	90	44.44	ATA/ATAPI-5
Ultra-DMA 4	60	66.67	ATA/ATAPI-5
Ultra-DMA 5	40	100	ATA/ATAPI-6
Ultra-DMA 6	30	133	ATA/ATAPI-7

Word 88 of the data block returned by the *Identify Device* command indicates the Ultra-DMA modes that are supported by the disk drive and the mode selected. Bits 0..6 of this word indicate by value 1 that the mode with the number corresponding to the particular bit position is supported (mode 0 for bit 0, up to mode 6 for bit 6). Bits 8..14 of word 88 indicate by value 1 that the mode with the number corresponding to the particular bit position minus 8 is selected (mode 0 for bit 8, up to mode 6 for bit 14).

7.5. Serial ATA Interface

7.5.1. Overview

Since the release of the first draft version of the ATA standard in 1989, the ATA interface has been continuously improved, so that its speed increased more than 25 times compared to the speed of the original version. Nevertheless, further improvement of the parallel ATA interface performance is difficult because of the problems specific to a parallel interface, such as the electromagnetic interference between signals or the difficulty of signal timing. The solution to these problems consists in using a serial interface, whose performance could be improved in a much simple manner by increasing the frequency of the clock signal.

In 2000, Intel and several manufacturers of disk drives (APT Technologies, Dell, IBM, Maxtor, Quantum, and Seagate Technology) started the development of a serial ATA interface, referred to as *Serial* ATA (SATA). For developing the specification of this interface, the *Serial ATA Working Group* has been formed. The first version (1.0) of the SATA specification has been released in 2001. For improving this specification, in 2002 the *Serial ATA II Working Group* has been formed, and it developed version 2.0 of the SATA specification. Later on, for updating the SATA specification and for promoting this interface, a new industrial association has been set up, called *Serial ATA International Organization* (*SATA-IO*). Information about the activity of this organization is available at https://sata-io.org. Version 3.0 of the SATA specification has been published in 2009, and version 3.1 in 2011.

Version 3.2 of the SATA standard has been released in 2013. This version contains the specification of the SATA Express interface, which enables to connect one or two disk drives with SATA interface or one disk drive with PCI Express interface. The PCI Express interface may use one or two PCI Express lanes. The increased speed of the PCI Express bus

allows optimizing the performance of *Solid-State Drives* (SSDs) and *Solid-State Hybrid Drives* (SSHDs), which incorporate NAND flash memories into magnetic disk drives. Version 3.2 also includes the specification of a card referred to as M.2, with small size (width of 22 mm, length of 30, 42, 60, 80, or 110 mm), which may be used as a solid-state drive for tablets or Ultrabook computers. In addition, version 3.2 contains the microSSD specification, which eliminates the connector from a solid-state drive, enabling to implement such a drive as a single chip placed directly to the motherboard.

Version 3.3 of the SATA standard, released in 2016, includes the specification of SMR (*Shingled Magnetic Recording*) technology, which enables to increase disk drive capacities by about 25%. With this technology, when a new track is written, it will partially overlap a previously written track, so that the previous track will be narrower and track density will increase. This approach was chosen because recording magnetic heads cannot be made as narrow as reading heads due to physical limitations. Version 3.3 also includes a feature called *Power Disable*, which enables the host computer to power-down a SATA disk drive. Version 3.4 of the SATA standard has been released in June 2018, and version 3.5 in July 2020. The current version of the SATA standard is 3.5a, released in March 2021.

Although the serial ATA interface differs physically from the parallel ATA interface by the cable and connector used, the two interfaces are software-compatible between them. Therefore, the existing BIOS programs, operating systems, and software that use the parallel ATA interface can be used without any changes with the serial ATA interface. Also, this interface allows connecting the existing ATA and ATAPI peripheral devices, including CD-ROM, CD-RW, and DVD drives, and other devices that can be connected to the parallel ATA interface.

Figure 7.1 shows the logo of the SATA interface.

Figure 7.1. Serial ATA interface logo.

While the parallel ATA interface transfers the data via a 16-bit channel, the serial ATA interface uses only two unidirectional serial channels, one for transmit and one for receive. Even though the data are sent serially bit by bit, the frequency of the clock signal used for the transfer is much higher than that of the parallel ATA interface. For instance, with the UDMA/133 transfer mode of the parallel ATA interface, the clock frequency is 33 MHz, and two words (four bytes) are transferred in each clock cycle, which results in a maximum transfer rate of 133 MB/s. With the first version of the SATA interface, the clock frequency is 1500 MHz (1.5 GHz). Because a byte is encoded on 10 bits, the transfer rate corresponding to this frequency is 150 MB/s, which is 12% higher than that of the last version of the parallel ATA interface. Whereas doubling the transfer rate for the parallel ATA interface in the near future is not probable, the transfer rate of the SATA-600 interface increased four times compared to the initial version, since this interface is available with a clock frequency of 6 GHz and a maximum transfer rate of 600 MB/s.

Table 7.5 presents the types of SATA interfaces.

Table 7.5. Types of SATA interfaces.

Interface Type	Frequency (GHz)	Transfer Rate
SATA-150	1.5	1.5 Gbits/s, 150 MB/s
SATA-300	3	3 Gbits /s, 300 MB/s
SATA-600	6	6 Gbits /s, 600 MB/s
SATA Express	8	2x8 Gbits/s, 2 GB/s

With the SATA interface, data bits are represented on the transmission line using the NRZ (*No Return to Zero*) coding. With this coding, a bit is represented by a change in the

electrical voltage of the line, and not by a certain voltage level. Two voltage levels are used, and for each bit of 1 in the data stream there is a transition from the current voltage level to the other level. The voltage level then remains unchanged until the next bit of 1, without returning to zero voltage.

The SATA interface uses the 8b/10b encoding for the data sent along the serial line, through which each data byte is represented by a particular combination of 10 bits. This encoding was originally developed by IBM in the early 1980's for high-speed data communications. The same encoding is used by many high-performance serial communication interfaces, including *Gigabit Ethernet*, *Fibre Channel*, *IEEE 1394*, and others. One of the purposes of the 8b/10b encoding is to ensure that there are never more than four 0 bits (or 1 bits) transmitted consecutively. This is actually a variant of RLL (*Run Length Limited*) encoding, which uses two parameters to define the encoding form. These parameters are the minimum number (the *run length*) and the maximum number (the *run limit*) of identical consecutive bits in each encoded byte. The variant used is RLL 0,4, where 0 represents the minimum number and 4 represents the maximum number of identical consecutive bits in each byte.

The 8b/10b encoding also ensures that there are never more than six or less than four bits of 0 (or bits of 1) in a single encoded byte. Because bits of 0 and 1 are represented on the transmission line as voltage changes on the line, the previous constraint ensures that the spacing between the voltage transitions on the line are balanced. This way a more balanced load on the circuits is achieved, increasing reliability.

The signals are sent on two pairs of wires, differentially. One pair represents the transmit channel, and the other pair represents the receive channel. Low voltage levels are used, of 0.25 V. The signals of a channel are differential in the sense that, if on one wire of the channel the voltage level is 0.25 V, on the other wire the voltage level is –0.25V. At every instant, the voltages on the two wires of a channel are opposite, and the voltage difference between the two wires is 0.5 V. This voltage difference is not affected by noise or other external perturbations, which represents an important advantage of differential signaling.

The SATA interface uses a point-to-point connection. Therefore, unlike with the parallel ATA interface, a single device is connected to each SATA port. This way, there are no daisy-chained devices and no settings are required to designate the master and the slave device. To connect several devices, multiple SATA ports are needed. Usually, motherboards are equipped with four SATA ports, two primary ports and two secondary ports.

The main advantages of the SATA interface compared to the parallel ATA interface are high speed, small connector and cable sizes, increased cable length, and the capability to connect and disconnect the devices without removing the supply voltage (hot plug). In addition, the SATA standards describe the possibility to connect several devices to the same SATA port by using a port extender. This way, the connections are simplified and the SATA ports on the motherboard are used more efficiently, because they may ensure a transfer rate that is enough for several devices. Furthermore, it is possible to set up more complex storage systems, such as RAID (*Redundant Array of Independent Disks*) disk arrays, especially if a PCI Express bus is available, which could provide the required transfer rate. Due to these advantages, the SATA interface gradually replaced the parallel ATA interface.

7.5.2. Connectors and Cables

The SATA interface requires a data connector and a power connector. In many cases, combo connectors are used, which contain both the data connector and the power connector. The data connector contains seven pins and it has small size, as its width is 14 mm. Four pins are used for the transmit and receive differential channels, and three pins are used for ground connections. By using multiple ground connections, which separate the two differential data channels, the electrical interferences between these channels are reduced.

Table 7.6. SATA interface data connector pins.

Pin	Signal	Description
S1	GND	Ground
S2	A+	Host transmit +
S3	A–	Host transmit –
S4	GND	Ground
S5	B–	Host receive –
S6	B+	Host receive +
S7	GND	Ground

Table 7.6 presents the SATA interface data signals and their assignment to the data connector pins. All of the ground pins are longer than the other pins, so that they will make the connection before the signal pins. This enables to connect and disconnect the SATA disk drives without switching off the computer.

The power connector may have a standard size or a smaller size (mini-SATA or micro-SATA). The standard power connector contains 15 pins and it supplies the voltages of 5 V, 12 V, and 3.3 V (although the 3.3 V voltage is used by very few disk drives). Each voltage is supplied by three pins in parallel to reduce the impedance and increase the current. Each pin can supply a current of 1.5 A, so that the connector can supply a current of up to 4.5 A for each of the three voltages. The power connector contains five ground pins instead of a single ground pin, which provide a low-impedance ground connection. One pin of the power connector can be used for staggered spin-up of the disk drives and/or to indicate drive activity. If this pin is connected to the ground at the connector, the drive motor spins up as soon as power is applied. If left unconnected, the drive waits until it receives a command. This prevents several drives to spin up simultaneously, which might overload the power supply. The pin is pulled low by the drive when it executes a command, so it can be connected to an LED to indicate drive activity.

Figure 7.2 illustrates a standard combo SATA connector. The 7-pin data connector is on the left, and the 15-pin power connector is on the right.

Figure 7.2. Standard combo SATA connector of a disk drive.

A mini-SATA (mSATA or *slimline*) combo connector is intended for smaller devices, such as notebook optical drives. In this combo connector, the data connector is identical to the data connector of the standard version, while the power connector is reduced to six pins. One pin is provided to signal device presence, two pins supply in parallel 5 V, two pins are used for ground connection, and one pin is used for diagnostic during manufacturing. Passive adapters exist to convert between a standard SATA connector and a mini-SATA connector. Figure 7.3 illustrates the connectors of a mini-SATA cable.

Figure 7.3. Mini-SATA combo connectors of a SATA cable.

A micro-SATA (uSATA) combo connector is intended for 1.8 inch (46 mm) disk drives. The data connector of this combo connector is similar in appearance to the standard data connector, but is slightly thinner. The power connector contains nine pins, including two pins for the supply voltage of 3.3 V, two pins for the supply voltage of 5 V, and two ground pins. Two pins are defined as vendor specific. Figure 7.4 illustrates the connectors of a micro-SATA cable.

The maximum length of the SATA cable is 1 m, unlike the parallel ATA interface cable, for which the maximum length is only 0.45 m. A separate cable is needed to connect each device to a SATA port of the motherboard.

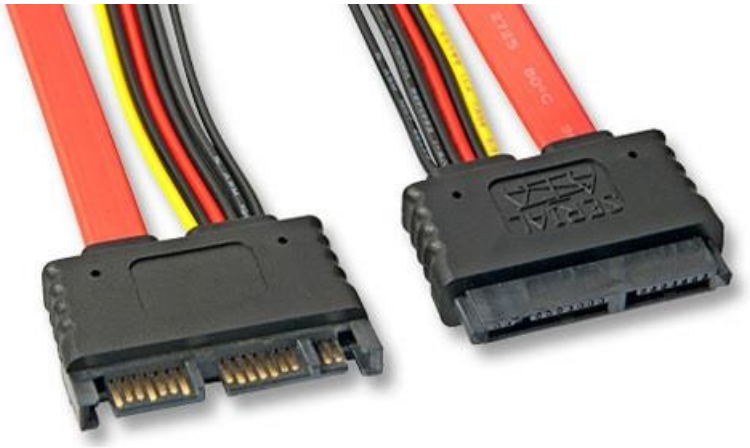


Figure 7.4. Micro-SATA combo connectors of a SATA cable.

7.5.3. Differences between the SATA and SAS Interfaces

The main differences between the SATA and SAS (*Serial Attached* SCSI) interfaces are the following:

- SATA disk drives are identified by their port number connected to the host adapter, while SAS devices are identified by their SAS address or *World Wide Name*.
- Unlike the SATA protocol, the SAS protocol supports multiple initiators in a SAS domain.
- The SATA interface only supports magnetic disk drives and optical drives. The SAS interface also supports other types of devices, such as scanners and printers. However, these devices usually have other interfaces than SAS, such as USB, IEEE 1394, Ethernet, or Wi-Fi.
- The SATA interface uses lower voltage levels (0.4 – 0.6 V) than the SAS interface (0.8 – 1.6 V).
- Because of the higher signaling voltages used by the SAS interface, longer cables (up to 8 m) can be used with this interface, compared to the SATA interface for which the maximum cable length can be 1 m.
- A difference between a SATA drive connector and a SAS drive connector is that the latter has a second set of pins underside. These additional pins are provided for the second port of the drive. SATA drives, which are single-ported, do not have these pins.
- SATA drives are less expensive than SAS drives and they usually have higher capacities than that of SAS drives.

7.6. ATA/ATAPI Interface Registers

Communication with disk drive controllers is achieved via I/O registers. Unlike with other interfaces, in which only the selected controller receives commands from the computer, with the ATA/ATAPI interface the contents of registers are sent to both drives and their embedded controllers. The computer makes the distinction between the two drives through the DEV bit of the Device register. If the DEV bit is 0, drive 0 (master) is selected, otherwise drive 1 (slave) is selected. If there is a single drive, this should be configured as master.

Table 7.7 presents the ATA/ATAPI interface registers and their offsets relative to the base address of command block registers and the base address of control block registers.

Table 7.7. ATA/ATAPI interface registers.

Offset	ATA		ATAPI	
	Command Block Registers		Command Block Registers	
	When Read	When Written	When Read	When Written
0	Data	Data	Data	Data
1	Error	Features	Error	Features
2	Sector Count	Sector Count	Interrupt Reason	Sector Count
3	LBA Low	LBA Low	LBA Low	LBA Low
4	LBA Mid	LBA Mid	Byte Count Low	Byte Count Low
5	LBA High	LBA High	Byte Count High	Byte Count High
6	Device	Device	Device	Device
7	Status	Command	Status	Command
	Control Block Registers		Control Block Registers	
6	Alternate Status	Device Control	Alternate Status	Device Control

Data are transferred in parallel (on 16 bits) between the computer memory and the disk drive buffer under the control of commands sent previously from the computer. Data read from the medium are stored in the drive buffer, to be transferred to the computer, and data transferred from the computer memory are stored in the drive buffer, to be written to the medium. If two drives are daisy-chained, the commands are sent to both drives and, except for the device diagnostic command, only the selected drive will execute the command.

When the ATA interface implements the 48-bit LBA addressing, defined starting with the ATA/ATAPI-6 version of the standard, the following registers operate as two byte deep FIFO memories: the Features register, the Sector Count register, and the LBA address registers (Low, Mid, and High). Each time one of these registers is written, the new content written is placed into the "most recently written" location, and the previous content is moved to the "previous content" location. For instance, when a command code that uses the 48-bit LBA addressing, such as the *Read Sector(s) Ext* command, is written to the Command register, the address used by this command and the sector count are indicated in Table 7.8.

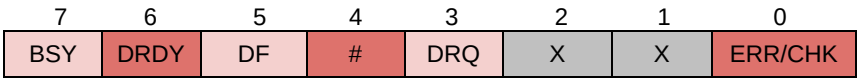
Table 7.8. The LBA address and sector count when using 48-bit LBA addressing.

Register	"Most Recently Written" Location	"Previous Content" Location
LBA Low	LBA address, bits 7..0	LBA address, bits 31..24
LBA Mid	LBA address, bits 15..8	LBA address, bits 39..32
LBA High	LBA address, bits 23..16	LBA address, bits 47..40
Sector Count	Sector count, bits 7..0	Sector count, bits 15..8

The host computer may read the "previous content" location of the Features register, Sector Count register, and LBA address registers by setting to 1 bit 7 (HOB – *High Order Bit*) of the Device Control register and then reading the desired register. If the HOB bit is 0, when reading one of the registers mentioned, the "most recently written" location is read. Writing is always performed to the "most recently written" location, regardless of the state of HOB bit of the Device Control register.

7.6.1. Status Register

This register contains the current status of the drive. If the BSY bit is 0, the other bits of the register contain valid information; otherwise the other bits do not contain valid information. If this register is read by the host computer during a pending interrupt, the interrupt condition is cleared.



- Bit 7 (BSY – *Busy*) is set to 1 whenever the disk drive has control of the Command Block registers. If the BSY bit is 1, a write to any Command Block register by the host computer will be ignored by the drive. The BSY bit is cleared to 0 by the drive at command completion and after setting the DRQ status bit to 1 to indicate the device is ready to transfer data.
- Bit 6 (DRDY – *Device Ready*) is set to 1 to indicate that the disk drive accepts commands. If the DRDY bit is 0, the drive will accept and attempt to execute the *Device Reset* and *Execute Device Diagnostic* commands. Other commands will not be accepted, and the drive will set the ABRT bit in the Error register and the ERR/CHK bit in the Status register, before resetting the BSY bit to indicate completion of the command.
- Bit 5 (DF – *Device Fault*) indicates by value 1 that a device fault has been detected. A device fault is any event that prevents the device from completing a command, event which is not the result of an error described in the Error register.
- Bit 4 is command specific.
- Bit 3 (DRQ – *Data Request*) indicates by value 1 that the disk drive is ready to transfer data between the host computer and the drive. After the computer writes a command code to the Command register, the drive sets the BSY bit or the DRQ bit to 1 until command completion.
- Bits 2..1 are undefined.
- Bit 0 (ERR/CHK – *Error*/*Check*) is defined as ERR for all commands except for the *Packet* and *Service* commands, for which this bit is defined as CHK. The ERR bit indicates by value 1 that an error occurred during execution of the previous command. The bits in the Error register contain additional information about the error. The CHK bit indicates by value 1 that an exception condition occurred.

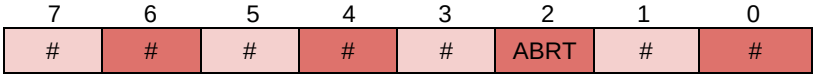
7.6.2. Data Register

This is a 16-bit register and is used for reading or writing the data during data transfers. This register shall be accessed for data transfers in PIO mode only when the DRQ bit of the Status register is set to 1.

7.6.3. Error Register

This register contains the status of the last command executed by the disk drive or a diagnostic code. At completion of any command except the *Execute Device Diagnostic* and *Device Reset* commands, the contents of this register are valid when the BSY and DRQ bits of the Status register are cleared to 0 and the ERR/CHK bit in the same register is set to 1. At completion of an *Execute Device Diagnostic* or *Device Reset* command and after a hardware or software reset, this register contains a diagnostic code.

Except for bit 2 (ABRT), the meaning of other bits of the Error register varies depending on the command that has been executed.



- Bit 2 (ABRT – *Command Aborted*) indicates by value 1 that the requested command has been aborted because the command code or a command parameter is invalid, the command is not implemented, or some other error has occurred.

7.6.4. Features Register

This register can be used to set various features of the interface, e.g., to validate or invalidate the cache memory through the *Set Features* command. The register shall be written only when the BSY and DRQ bits of the Status register are both 0. The content of this register becomes a command parameter when the command code is written into the Command register.

The structure of this register is command specific. For commands that use 48-bit LBA addressing, the Features register operates as a two byte deep FIFO memory.

7.6.5. Sector Count / Interrupt Reason Register

For the ATA interface, this register is called the Sector Count register. For the ATAPI interface, this register is called the Interrupt Reason register. The register shall be written only when the BSY and DRQ bits of the Status register are both 0. The contents of this register are valid only when the BSY and DRQ bits of the Status register are both 0. The contents of this register become a command parameter when the command code is written into the Command register.

The structure of this register is command specific. In general, this register is written with the number of data sectors to be transferred in a read or write operation between the host computer and the disk drive. For some commands, this register has a function different than a count register. For media access commands, this register contains the value 0 at command completion if no errors were indicated in the Status register. When errors occurred, this register contains the number of sectors that should be transferred in order to complete the operation.

For read and write commands that use 28-bit LBA addressing, if the Sector Count register contains the value 0x00, this specifies a number of 256 sectors to be transferred. For commands that use 48-bit LBA addressing, this register operates as a two byte deep FIFO memory. For these commands, if the register contains the value 0x0000, this specifies a number of 65,536 sectors to be transferred.

7.6.6. LBA Low Register

This register allows writing part of the sector address for read and write commands that use LBA addressing. The register shall be written only when the BSY and DRQ bits of the Status register are both 0. The contents of this register are valid only when the BSY and DRQ bits of the Status register are both 0. The contents of this register become a command parameter when the command code is written into the Command register.

For commands that use 28-bit LBA addressing, the LBA Low register should be written with bits 7..0 of the LBA address. For commands that use 48-bit LBA addressing, the LBA Low register operates as a two byte deep FIFO memory, as described in Section 7.6.

7.6.7. LBA Mid / Byte Count Low Register

For the ATA interface, this register is called LBA Mid register. For the ATAPI interface, this register is called the Byte Count Low register. The register shall be written only when the BSY and DRQ bits of the Status register are both 0. The contents of this register are valid only when the BSY and DRQ bits of the Status register are both 0. The contents of the register become a command parameter when the command code is written into the Command register.

For commands that use 28-bit LBA addressing, the LBA Mid register should be written with bits 15..8 of the LBA address. For commands that use 48-bit LBA addressing, the LBA Mid register operates as a two byte deep FIFO memory.

7.6.8. LBA High / Byte Count High Register

For the ATA interface, this register is called the LBA High register. For the ATAPI interface, this register is called the Byte Count High register. The register shall be written only when the BSY and DRQ bits of the Status register are both 0. The contents of this register are valid only when the BSY and DRQ bits of the Status register are both 0. The contents of the register become a command parameter when the command code is written into the Command register.

For commands that use 28-bit LBA addressing, the LBA High register should be written with bits 23..16 of the LBA address. For commands that use 48-bit LBA addressing, the LBA High register operates as a two byte deep FIFO memory.

7.6.9. Device Register

This register is used for selecting the disk drive. The register shall be written only when the BSY and DRQ bits of the Status register are both 0. The contents of this register are valid only when the BSY bit of the Status register is 0. Except the DEV bit, all other bits of this register become a command parameter when the command code is written into the Command register.

7	6	5	4	3	2	1	0
X	LBA	X	DEV	#	#	#	#

- Bit 7 and bit 5 are undefined.
- Bit 6 (LBA) selects the sector addressing mode. Some commands require to set this bit to 1 to select LBA addressing. If this bit is cleared to 0, the CHS addressing is selected; this addressing has been used only up to the ATA/ATAPI-5 version of the standard.
- Bits 3..0 are command specific.
- Bit 4 (DEV – *Device Select*) selects by value 0 the drive 0, and by value 1 the drive 1.

7.6.10. Command Register

This register contains the command code to be sent to the disk drive. Command execution begins immediately after the command code is written into the Command register. The contents of the Command Block registers become parameters of the command when this register is written. Writing this register clears any pending interrupt condition.

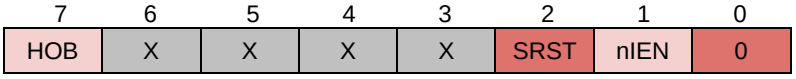
Except for the *Device Reset* command, this register shall be written only when the BSY and DRQ bits of the Status register are both 0.

7.6.11. Alternate Status Register

This register contains the same information as the Status register. The only difference is that reading the Alternate Status register does not imply an interrupt acknowledgement or clearing of the interrupt condition.

7.6.12. Device Control Register

This register allows the host computer to perform a software reset of the disk drives and to enable or disable the assertion of the *INTRQ* interrupt signal by the selected drive. When the Device Control register is written, both drives respond to the write regardless of which drive is selected.



- Bit 7 (HOB – *High Order Byte*) is defined only when the 48-bit LBA addressing is implemented. If this bit is set to 1, reading of the Features register, the Sector Count register, and the LBA address registers is performed from the "previous content" location. If the HOB bit is set to 0, reading is performed from the "most recently written" location. Writing to any Command Block register has the effect of resetting the HOB bit to 0.
- Bits 6..3 are reserved.
- Bit 2 (SRST – *Software Reset*) is the software reset bit of the disk drives. If there are two daisy-chained drives, by setting this bit to 1 both drives are reset.
- Bit 1 (nIEN – *INTRQ Enable*) enables by value 0 the assertion of the *INTRQ* interrupt request signal by the disk drive.
- Bit 0 shall be cleared to 0.

7.7. Protocols Used for ATA Commands

The ATA/ATAPI standards define the protocols used for data transfers between the host computer and the disk drive and the read/write cycle times. This section presents example protocols used for executing ATA commands: the protocol for commands that do not transfer data, the protocol for input in PIO mode, and the protocol for the *Execute Device Diagnostic* command.

7.7.1. ATA Protocol for Non-Data Commands

This protocol is used for commands such as *Check Power Mode*, *Flush Cache*, *Read Native Max Address Ext*, *Read Verify Sector(s) Ext*, *SMART Enable Operations*, or *SMART Return Status*. Assuming that interrupt generation is not enabled, the ATA protocol for non-data commands is the following:

1. Software reads the Status register and waits until the BSY and DRQ bits become 0. If these bits do not become 0 in a certain time (e.g., 1 second), software may assume that the drive does not respond or it does not exist and should abandon the execution of the protocol.
2. Software clears the DEV bit of the Device register to 0 if drive 0 should be selected, or sets it to 1 if drive 1 should be selected.
3. Software initializes the registers with the command parameters. These registers may include the Features register, the Sector Count register, and the LBA registers. The registers that should be initialized depend on the specific command. Example commands are described in Sections 7.8.2-7.8.4.
4. Software writes the command code to the Command register. The drive will start executing the requested command.
5. Software reads the Status register and waits until the BSY bit becomes 0, which indicates that command execution is completed. If the BSY bit does not become 0 in a certain time (e.g., 1 second), software should abandon the execution of the protocol.
6. Software checks the ERR/CHK bit in the Status register. If this bit is 1, the command completed with an error; otherwise, the command completed successfully.

7.7.2. ATA Protocol for Input in PIO Mode

This protocol is used for commands such as *Identify Device*, *Identify Packet Device*, *Read Buffer*, *Read Sector(s) Ext*, or *SMART Read Log*. Assuming that interrupt generation is not enabled, the ATA protocol for an input operation in PIO mode is the following:

1. Software reads the Status register and waits until the BSY and DRQ bits become 0. If these bits do not become 0 in a certain time (e.g., 1 second), software may assume that the drive does not respond or it does not exist and should abandon the execution of the protocol.
2. Software clears the DEV bit of the Device register to 0 if drive 0 should be selected, or sets it to 1 if drive 1 should be selected.
3. Software initializes the registers with the command parameters. These registers may include the Features register, the Sector Count register, and the LBA registers. The registers that should be initialized depend on the specific command. Example commands are described in Sections 7.8.2-7.8.4.
4. Software writes the command code to the Command register. The drive will prepare the requested data for transfer to the host computer.
5. Software reads the Status register and waits until the BSY bit becomes 0. If the BSY bit does not become 0 in a certain time (e.g., 1 second), software should abandon the execution of the protocol.
6. Software checks the DRQ bit of the Status register. If this bit is 0, the drive has completed the command with an error. In this case, the protocol is completed; software may read the Error register for more information about the error occurred. If the DRQ bit is 1, software continues with Step 7.
7. Software transfers a data block by reading the Data register word by word. The number of words that should be transferred depends on the particular command. For instance, the *Identify Device* and *Identify Packet Device* commands require to transfer a number of 256 data words.
8. If all data blocks for the particular command have been transferred, the command completed successfully. Otherwise (if there are more data blocks to be transferred), software continues with Step 9.
9. Software waits for a time corresponding to a PIO transfer cycle. For instance, it may read the Alternate Status register, ignoring the result.
10. Software continues with the transfer of a new data block, from Step 5.

7.7.3. ATA Protocol for *Execute Device Diagnostic* Command

Assuming that interrupt generation is not enabled, the ATA protocol for the *Execute Device Diagnostic* command is the following:

1. Software reads the Status register and waits until the BSY and DRQ bits become 0. If these bits do not become 0 in a certain time (e.g., 1 second), software may assume that the drive does not respond or it does not exist and should abandon the execution of the protocol.
2. Software clears the DEV bit of the Device register to 0.
3. Software writes the command code to the Command register. The drive (or both drives, 0 and 1, if present) will begin performing the self-diagnostic testing.
4. Software waits for at least 2 ms.

5. Software reads the Status register and waits until the BSY bit becomes 0, which indicates that command execution is completed. If the BSY bit does not become 0 in a time of 6 seconds, software should abandon the execution of the protocol.
6. Software checks the results of the command execution. The Error register contains a diagnostic code. The LBA registers and the Sector Count register contain a signature that can be used to identify ATA and ATAPI drives.

Note

- The diagnostic codes returned by the *Execute Device Diagnostic* command are listed in Table 7.10, and the signatures specific to ATA and ATAPI drives are listed in Table 7.11.

7.8. ATA Commands

7.8.1. ATA Command List

Table 7.9 presents a list of the main ATA commands and the registers that should be loaded with the command parameters. The meaning of the registers is described next.

FR: Features register;
SCR: Sector Count register;
LBA: LBA registers;
DR: Device register.

V indicates a valid parameter for the corresponding register. For the Device register (DR), V indicates the use of both the DEV bit for the drive number and bits 27..24 of the LBA address, while D indicates that only the parameter for the drive number is valid.

Table 7.9. List of ATA commands.

Command	Code	FR	SCR	LBA	DR
Check Power Mode	0xE5				D
Device Configuration Freeze Lock	0xB1	0xC1			D
Device Configuration Identify	0xB1	0xC2			D
Device Configuration Restore	0xB1	0xC0			D
Device Configuration Set	0xB1	0xC3			D
Device Reset	0x08				D
Download Microcode	0x92	V	V	V	V
Execute Device Diagnostic	0x90				
Flush Cache	0xE7				D
Flush Cache Ext	0xEA				D
Identify Device	0xEC				D
Identify Packet Device	0xA1				D
Idle	0xE3	V			D
Idle Immediate	0xE1				D
NOP	0x00	V			D
Packet	0xA0	V	V	V	D
Read Buffer	0xE4				D
Read DMA	0xC8		V	V	V
Read DMA Ext	0x25		V	V	D
Read DMA Queued	0xC7	V	V	V	V
Read DMA Queued Ext	0x26	V	V	V	D
Read Multiple	0xC4		V	V	V
Read Multiple Ext	0x29		V	V	D
Read Native Max Address	0xF8				D
Read Native Max Address Ext	0x27				D
Read Sector(s)	0x20		V	V	V
Read Sector(s) Ext	0x24		V	V	D
Read Verify Sector(s)	0x40		V	V	V
Read Verify Sector(s) Ext	0x42		V	V	D
Security Disable Password	0xF6				D
Security Erase Prepare	0xF3				D
Security Erase Unit	0xF4				D

Command	Code	FR	SCR	LBA	DR
Security Freeze Lock	0xF5				D
Security Set Password	0xF1				D
Security Unlock	0xF2				D
Service	0xA2				D
Set Features	0xEF	V	V	V	D
Set Max Address	0xF9			V	V
Set Max Address Ext	0x37			V	D
Set Multiple Mode	0xC6		V		D
Sleep	0xE6				D
SMART Disable Operations	0xB0	0xD9		V	D
SMART Enable Operations	0xB0	0xD8		V	D
SMART Execute Off-Line	0xB0	0xD4		V	D
SMART Read Log	0xB0	0xD5	V	V	D
SMART Return Status	0xB0	0xDA		V	D
SMART Write Log	0xB0	0xD6	V	V	D
Standby	0xE2		V		D
Standby Immediate	0xE0				D
Write Buffer	0xE8				D
Write DMA	0xCA		V	V	V
Write DMA Ext	0x35		V	V	D
Write DMA Queued	0xCC	V	V	V	V
Write DMA Queued Ext	0x36	V	V	V	D
Write Log Ext	0x3F		V	V	D
Write Multiple	0xC5		V	V	V
Write Multiple Ext	0x39		V	V	D
Write Sector(s)	0x30		V	V	V
Write Sector(s) Ext	0x34		V	V	D

7.8.2. *Execute Device Diagnostic* Command

This command causes the drives to perform the internal diagnostic tests. If present, both drives connected to an ATA channel will execute the command regardless of which device is selected. The protocol for this command is described in Section 7.7.3. No registers have to be initialized before writing the command code into the Command register.

After command execution, the ERR bit of the Status register will be cleared to 0. The Error register will contain an 8-bit diagnostic code. The meaning of the diagnostic codes is presented in Table 7.10. Codes other than 0x01 and 0x81 may indicate additional information about a drive's failure.

Table 7.10. Diagnostic codes returned by the *Execute Device Diagnostic* command.

Diagnostic Code	Description
0x01	Drive 0 passed, Drive 1 passed or not present
0x00, 0x02-0x7F	Drive 0 failed, Drive 1 passed or not present
0x81	Drive 0 passed, Drive 1 failed
0x80, 0x82-0xFF	Drive 0 failed, Drive 1 failed

The Sector Count, LBA Low, LBA Mid, and LBA High registers will contain a signature that is specific to ATA drives. When the *Execute Device Diagnostic* command is sent to an ATAPI drive, the drive will also return a signature in the corresponding registers of the ATAPI interface, signature that is specific to ATAPI drives. Table 7.11 shows the signatures returned by ATA drives and ATAPI drives.

Table 7.11. Signatures of ATA and ATAPI drives.

ATA Register	ATA Signature	ATAPI Register	ATAPI Signature
Sector Count	0x01	Interrupt Reason	0x01
LBA Low	0x01	LBA Low	0x01
LBA Mid	0x00	Byte Count Low	0x14
LBA High	0x00	Byte Count High	0xEB

7.8.3. *Identify Device* Command

This command allows the host computer to receive the parameters of the disk drive. When receives this command, the drive prepares a block of 256 words with information about the drive: operating parameters, manufacturer, model, revision number, serial number, etc. The host computer may transfer the data block by reading successively the Data register. The protocol for this command is the protocol for input in PIO mode (Section 7.7.2). Except for the DEV bit that should be cleared or set in the Device register, no other registers have to be initialized before writing the command code into the Command register.

ATA drives will not report an error after executing this command. ATAPI drives will set the ABRT bit in the Error register and will place the signature of ATAPI drives in the Interrupt Reason, LBA Low, Byte Count Low, and Byte Count High registers (Table 7.11).

Table 7.12 presents the meaning of part of the 16-bit words that are returned by the *Identify Device* command. Some parameters of the disk drive are defined as strings of ASCII characters. Each word contains two ASCII characters: the first character is contained in the most significant byte of the word, and the second character is contained in the least significant byte of the word. Some parameters are defined as two or four consecutive words. For these parameters, the least significant part is contained in the first word.

Table 7.12. Meaning of words returned by the *Identify Device* command.

Word	Meaning
0	General configuration information
1	Number of logical cylinders in the default CHS translation
3	Number of logical heads in the default CHS translation
6	Number of logical sectors per track in the default CHS translation
10-19	Serial number (20 ASCII characters)
23-26	Firmware revision (8 ASCII characters)
27-46	Model number (40 ASCII characters)
54	Number of logical cylinders in the current CHS translation
55	Number of current logical heads in the current CHS translation
56	Number of current logical sectors per track in the current CHS translation
57-58	Capacity in sectors in the current CHS translation
60-61	Total number of addressable sectors (28-bit LBA addressing)
100-103	Total number of addressable sectors (48-bit LBA addressing)

7.8.4. *Read Native Max Address Ext* Command

This command is implemented by disk drives that support 48-bit LBA addressing. The command returns the native maximum LBA address of the disk drive. This address is the highest address accepted by the drive in the factory default condition. For this command, the protocol for non-data commands should be used (Section 7.7.1). Before writing the command code into the Command register, the LBA bit in the Device register should be set to 1 to specify LBA addressing, and the DEV bit in the same register should be cleared or set to specify the selected drive.

The 48-bit native maximum address is returned in the LBA Low, LBA Mid, and LBA High registers. When 48-bit LBA addressing is used, the LBA registers operate as two byte deep FIFO memories. The two bytes of each LBA register are selected individually by clearing to 0 or setting to 1 the HOB bit of the Device Control register. Table 7.13 shows the contents of the LBA registers after the execution of this command.

Table 7.13. Contents of LBA registers after executing the *Read Native Max Address Ext* command.

Register		Contents
LBA Low	HOB = 0	Native max LBA address (7..0)
	HOB = 1	Native max LBA address (31..24)
LBA Mid	HOB = 0	Native max LBA address (15..8)
	HOB = 1	Native max LBA address (39..32)
LBA High	HOB = 0	Native max LBA address (23..16)
	HOB = 1	Native max LBA address (47..40)

If this command is not implemented, the drive will set to 1 the ABRT bit in the Error register. In this case, the ERR bit of the Status register will also be set to 1.

7.8.5. *SMART Return Status* Command

This command allows the host computer to receive the S.M.A.R.T. reliability status of the disk drive. For this command, the protocol for non-data commands should be used (Section 7.7.1). Before writing the command code into the Command register, the DEV bit in the Device register should be cleared or set to specify the selected drive. In addition, the following registers should be initialized: the Features register should be set to 0xDA; the LBA Mid register should be set to 0x4F; the LBA High register should be set to 0xC2.

For drives that implement the S.M.A.R.T. technology, each manufacturer defines a set of attributes or operational parameters of the drive, and sets threshold values beyond which these attributes do not pass under normal operation. Examples of attributes are: count of reallocated sectors (when a bad sector is found, its contents are transferred to a spare sector); number of data blocks with uncorrectable errors; count of attempts to reach the nominal spin speed of the drive (if a first attempt was unsuccessful). A threshold exceeded condition means that at least one attribute passed the threshold value. As response to the *SMART Return Status* command, the drive will indicate whether it has detected a threshold exceeded condition.

If the drive has not detected a threshold exceeded condition, it sets the LBA Mid register to 0x4F and the LBA High register to 0xC2. If the drive has detected a threshold exceeded condition, it sets the LBA Mid register to 0xF4 and the LBA High register to 0x2C. If the drive does not support this command, if S.M.A.R.T. operation is disabled, or if the input register values are invalid, the drive will set to 1 the ABRT bit in the Error register. In this case, the ERR bit of the Status register will also be set to 1.

7.9. Intel SATA Controllers

The *Platform Controller Hub* (PCH) component of current Intel chipsets contains two SATA controllers, both on PCIe bus 0. The first controller represents the PCIe device 31, function 2, and the second controller represents the PCIe device 31, function 5 (the device and function number may depend on the chipset). Depending on the system configuration, only the first controller may be enabled, or both controllers may be enabled. The controllers interact with the disk drives through a register interface that is equivalent to that of a traditional ATA (IDE) host adapter. The two controllers support up to six SATA ports. Each port can be independently enabled or disabled and has a separate DMA controller.

The features of the SATA controllers depend on the chipset series. For instance, the SATA controllers of the Intel 8 Series chipset used in the laboratory computers support transfer rates of 6 Gbits/s, 48-bit LBA addressing, and three modes of operation: ATA (IDE), AHCI (*Advanced Host Controller Interface*), and RAID (*Redundant Array of Independent Disks*).

For the ATA (IDE) mode of operation, each SATA controller contains a set of registers that hold copies of the ATA interface registers. The SATA controllers emulate the behavior of Command Block registers, Control Block registers, PIO data transfers, DMA data transfers, and interrupts.

The SATA controllers provide hardware support for the AHCI programming interface, which has been developed by Intel and other companies. This programming interface defines memory structures for performing transactions between a SATA controller and a software driver, and enables advanced performance and features. Examples of advanced features enabled by the AHCI programming interface are no master/slave designation for SATA drives (each drive is treated as a master drive), hardware assisted native command queuing (the drive may reorder commands in order to increase the efficiency of data transfers), and hot-plug support (disk drives can be connected and disconnected without prior notification to the system).

The PCI configuration registers of the Intel SATA controllers include the PCI configuration header registers and a number of PCI device-specific registers. These registers can be accessed using either the PCI-compatible configuration mechanism or the PCIe enhanced configuration mechanism. The PCI configuration header registers are listed in Table 7.14. The offset indicated in the table is relative to the base address of the configuration space allocated for a specific controller (device 31, function 2 for the first SATA controller, or device 31, function 5 for the second SATA controller).

Table 7.14. The PCI configuration header registers of Intel SATA controllers.

Offset	Mnemonic	Register Name	Size (Bits)
0x00	VID	Vendor Identification	16
0x02	DID	Device Identification	16
0x04	PCICMD	PCI Command	16
0x06	PCISTS	PCI Status	16
0x08	RID	Revision Identification	8
0x09	PI	Programming Interface	8
0x0A	SCC	Sub-Class Code	8
0x0B	BCC	Base Class Code	8
0x0D	PMLT	Primary Master Latency Timer	8
0x0E	HTYPE	Header Type	8
0x10	PCMD_BAR	Primary Command Block Base Address	32
0x14	PCNL_BAR	Primary Control Block Base Address	32
0x18	SCMD_BAR	Secondary Command Block Base Address	32
0x1C	SCNL_BAR	Secondary Control Block Base Address	32
0x20	BAR	Legacy Bus Master Base Address	32
0x24	ABAR/SIDPBA	AHCI Base Address / SATA Index Data Pair Base Address	32
0x2C	SVID	Subsystem Vendor Identification	16
0x2E	SID	Subsystem Identification	16
0x34	CAP	Capabilities Pointer	8
0x3C	INT_LN	Interrupt Line	8
0x3D	INT_PN	Interrupt Pin	8

The PCI configuration header registers of the SATA controllers have the same functions as the general PCI configuration header registers described in the laboratory work *PCI Express Bus*. The PCMD_BAR register contains on bit positions 15..3 the base address of the I/O space allocated to the Command Block registers of the primary ATA channel, and the SCMD_BAR register contains on bit positions 15..3 the base address of the I/O space allocated to the Command Block registers of the secondary ATA channel. The PCNL_BAR register contains on bit positions 15..2 the base address of the I/O space allocated to the Control Block registers of the primary ATA channel, and the SCNL_BAR register contains on bit positions 15..2 the base address of the I/O space allocated to the Control Block registers of the secondary ATA channel.

7.10. Applications

7.10.1. Answer the following questions:

a. What is the aim of the S.M.A.R.T. technology?

b. What are the enhancements introduced by the ATA/ATAPI-6 version of the ATA standard?

c. What are the advantages of the serial ATA interface compared to the parallel ATA interface?

d. What are the enhancements introduced by the 3.2 and 3.3 versions of the SATA standard?

7.10.2. Create a *Windows* application for determining the base addresses of the I/O registers for the first SATA controller of the computer. As model for the *Windows* application,

use the AppScroll-e application available on the laboratory web page in the AppScroll-e.zip archive. Perform the following operations to create the application project:

1. In the *Visual Studio 2022* programming environment, create a new empty *Windows Desktop* project with the *Windows Desktop Wizard*.
2. Verify that the active solution platform is set to x64.
3. Change the *Character Set* project property to *Not Set*.
4. Copy to the project folder the files contained in the AppScroll-e.zip archive and add to the project these files.
5. Copy to the project folder the Hw.h and Hw64.lib files from a previously created project. Copy to the project folder the PciBaseAddressUEFI-e.cpp and ATA-ATAPI-e.h files, available on the laboratory page in the ATA-ATAPI-e.zip archive.
6. Add to the project the Hw.h, ATA-ATAPI-e.h, and PciBaseAddressUEFI-e.cpp files.
7. Specify the Hw64.lib file as an additional dependency for the linker.
8. Open the AppScroll-e.cpp source file and add an `#include` directive to include the ATA-ATAPI-e.h header file. Declare `PciBaseAddressUEFI()` as a function that has no parameters and returns a `DWORD64` value.
9. Select *Build* → *Build Solution* and make sure that the application builds correctly.

In the AppScroll-e.cpp source file, first call the `PciBaseAddressUEFI()` function to determine the base address of the PCIe extended configuration space and store the base address in a global variable. If the function returns 0, the base address cannot be successfully determined, and in this case the application should return with an error code. Then, write a function that returns in a double-word (`DWORD`) the base address of the Command Block registers and the base address of the Control Block registers for a SATA controller of the computer. The function has four input parameters; the first three parameters are the bus number, device number, and PCIe function number of the SATA controller. The last parameter specifies the ATA channel for which the base addresses should be returned: if this parameter is 0, the function should return the base addresses for the primary ATA channel (channel 0) of the SATA controller, and if the parameter is 1, it should return the base addresses for the secondary ATA channel (channel 1) of the SATA controller. In this function, call the function that returns a pointer to a PCIe function's configuration header, written for Application 2.7.2 of the laboratory work *PCI Express Bus*, and access the base address registers through this pointer. The function returns a double word containing the base address of the Command Block registers in the low word and the base address of the Control Block registers in the high word.

Notes

- In the PCMD_BAR and SCMD_BAR registers, the base addresses of the Command Block registers are in the low words of these registers; bits 2..0 of these words should be cleared to 0 before returning the base addresses.
- In the PCNL_BAR and SCNL_BAR registers, the base addresses of the Control Block registers are in the low words of these registers; bits 1..0 of these words should be cleared to 0 before returning the base addresses.

After writing the function, include a call to this function in the `AppScroll()` function to determine the base addresses for the primary ATA channel of the first SATA controller, and another call to determine the base addresses for the secondary ATA channel of the same controller. For each channel, display the base address of the Command Block registers and the base address of the Control Block registers.

7.10.3. In the AppScroll-e.cpp source file, write a function for sending the *Execute Device Diagnostic* command to a disk drive. The input parameter of the function is the base

address (of type WORD) of the Command Block registers for an ATA channel; the function does not return any value. This command is described in Section 7.8.2, and the protocol for this command is described in Section 7.7.3. The function displays messages if the drive does not respond (if the BSY and DRQ bits do not become 0 in a time of around 1 s) or the command execution does not complete in a time of 6 s. If the command completes successfully, the function reads the Error register and displays the diagnostic code returned by the drive. In addition, the function reads the registers containing the signature of the drive (Table 7.11) and displays whether the drive is an ATAPI drive.

After writing the function, include a call to this function in the AppScroll() function using as parameter the base address of the Command Block registers for the primary ATA channel, and then another call using as parameter the base address of the Command Block registers for the secondary ATA channel.

Notes

- Define symbolically the offsets of the registers used in the function and the bitmasks needed for these registers with #define directives at the beginning of the source file.
- The address of a register is formed by adding its offset to the base address transmitted as parameter to the function.

7.10.4. Extend Application 7.10.3 by writing a function for sending the *Read Native Max Address Ext* command to a disk drive. The input parameters of the function are the following: the base address (of type WORD) of the Command Block registers for the ATA channel the drive is connected to; the base address (of type WORD) of the Control Block registers for the ATA channel the drive is connected to; the drive number (0 or 1). The function does not return any value. This command is described in Section 7.8.4, and the protocol for this command is described in Section 7.7.1. The function displays messages if the drive does not respond (if the BSY and DRQ bits do not become 0 in a time of around 1 s) or the command execution does not complete in a time of around 1 s. If the command completes successfully, the function clears to 0 the HOB bit of the Device Control register, reads the contents of the LBA registers, and stores them in local variables. Then the function sets to 1 the HOB bit of the Device Control register, reads again the contents of the LBA registers, and stores them in other local variables. Next, the function determines and displays the maximum LBA address of the disk drive using the contents of the LBA registers stored previously; the contents of the LBA registers after the command execution are presented in Table 7.13. Finally, the function computes and displays the maximum capacity in GB of the disk drive, assuming a sector size of 512 bytes.

After writing the function, include a call to this function in the AppScroll() function using as parameters the base addresses of the Command Block registers and Control Block registers for the primary ATA channel, and the drive number 0.

7.10.5. Extend Application 7.10.2 by writing a function that sends the *Identify Device* command to a disk drive. The input parameters of the function are the base address (of type WORD) of the Command Block registers for the ATA channel the drive is connected to, and the drive number (0 or 1). The function does not return any value. This command is described in Section 7.8.3, and the protocol for this command is described in Section 7.7.2, with a single data block of 256 words to be transferred. The function displays messages if the drive does not respond (if the BSY and DRQ bits do not become 0 in a time of around 1 s) or the command execution does not complete in a time of around 1 s. When the command completes successfully, the function displays the following information about the disk drive: model number, serial number, firmware revision, total number of addressable sectors with 28-bit LBA addressing, and total number of addressable sectors with 48-bit LBA addressing. Next, using the total number of addressable sectors and assuming that a sector contains 512 bytes, the function computes and displays the capacity in GB of the disk drive for 28-bit LBA addressing and for 48-bit LBA addressing.

After writing the function, include a call to this function in the `AppScroll()` function using as parameters the base addresses of the Command Block registers for the primary ATA channel and the drive number 0.

7.10.6. Extend the function written for Application 7.10.5 by displaying the following additional information about the disk drive:

- Maximum number of sectors that can be transferred per interrupt when executing the *Read/Write Multiple* commands (word 47, bits 7..0) and the current setting for the number of sectors transferred per interrupt (word 59, bits 7..0);
- Support for multiword DMA mode 2 (word 63, the meaning of the bits is described in Section 7.4.2);
- Support for PIO mode 4 (bit 1 of word 64 is set to 1 if this mode is supported);
- Minimum cycle time in PIO transfer modes when using the *IORDY* signal (the value of word 68 represents the minimum cycle time in nanoseconds);
- Version of the ATA standard the disk drive is compatible with (if one of the bits 4, 5, 6, and 7 of word 80 is set to 1, the drive is compliant with the ATA/ATAPI-4, ATA/ATAPI-5, ATA/ATAPI-6, and ATA/ATAPI-7 version, respectively);
- Support for 48-bit LBA addressing (bit 10 of word 83 is set to 1 if 48-bit addressing is supported);
- Support for Ultra-DMA mode 6 and the Ultra-DMA mode selected (word 88, the meaning of the bits is described in Section 7.4.2).

Bibliography

[1] American National Standards Institute, Inc., “Information Technology - AT Attachment 8 - ATA/ATAPI Architecture Model (ATA8-AAM)”, T13/1700-D Revision 3, 2006.

[2] American National Standards Institute, Inc., “Information Technology - AT Attachment 8 - ATA/ATAPI Command Set (ATA8-ACS)”, T13/1699-D Revision 4a, 2007.

[3] American National Standards Institute, Inc., “Information Technology - AT Attachment 8 - ATA/ATAPI Parallel Transport (ATA8-APT)”, T13/1698-D Revision 2, 2007.

[4] American National Standards Institute, Inc., “Information Technology - AT Attachment 8 - ATA/ATAPI Serial Transport (ATA8-AST)”, T13/1697-D Revision 1, 2007.

[5] Intel Corporation, “Intel 8 Series/C220 Series Chipset Family Platform Controller Hub (PCH)”, Datasheet, May 2014.

[6] Serial ATA International Organization, “Fast Just Got Faster: SATA 6 Gb/s”, 2009.

[7] Serial ATA International Organization, “SATA-IO Introduces New Standard for Embedded SSDs”, 2011.

[8] Serial ATA International Organization, “SATA-IO Unveils Revision 3.2 Specification”, 2013.

[9] Serial ATA International Organization, “What Is SATA Express?”, 2015.

[10] Wikimedia Foundation, Inc., “SATA”, 2024, https://en.wikipedia.org/wiki/SATA.

[11] Wikimedia Foundation, Inc., “SATA Express”, 2024, https://en.wikipedia.org/wiki/SATA_Express.