8. COMPACT DISCS. ATA PACKET INTERFACE

In the first part, this laboratory work presents the compact disc physical medium, data recording and encoding on the medium, error correction levels, sector format, and disc organization. In the second part, the laboratory work presents an overview of the ATA Packet Interface (ATAPI), the interface registers, command protocols, a command list for CD/DVD drives, and example commands.

8.1. Compact Disc Physical Medium

A compact disc has a diameter of 12 cm and a thickness of about 1.2 mm. It is made from a transparent polycarbonate substrate, the data being recorded as variable-sized microscopic cavities called *pits*, placed between surfaces called *lands*. The entire disc is covered with a thin reflective aluminum coating, and then a plastic protective layer (Figure 8.1).

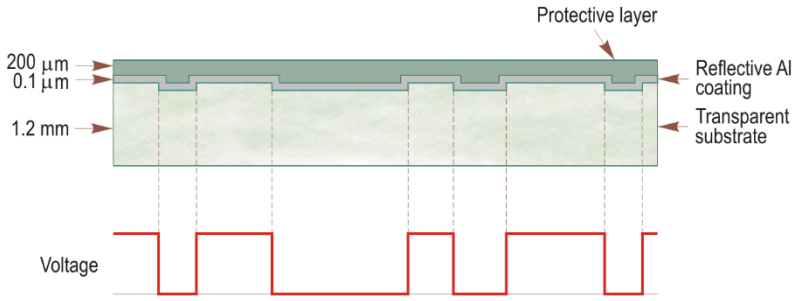

Figure 8.1. Layers of a compact disc and the signal generated by the photoelectric cell.

Reading the information from the disc is based on the different degree a laser beam is reflected by pits and lands. Lands have higher reflectivity than pits. The reflected beam is detected by a photoelectric cell, which generates a trend of electrical pulses.

Data are recorded on a single spiral, starting at the center of the disc. The distance between two consecutive tracks of the spiral is about 1.6 µm, and this is equivalent to a density of about 16,000 tracks per inch (TPI). Each bit of information requires a length of 300 nm on the spiral track. The pits have a depth of 0.12 µm and a width of 0.6 µm (Figure 8.2). The length of pits and lands ranges between 0.9 µm and 3.3 µm.

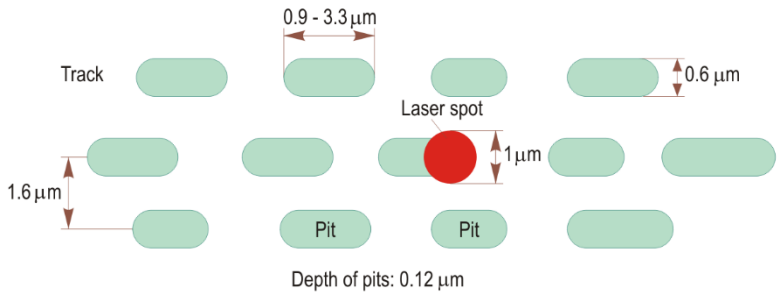

Figure 8.2. Sizes of pits and distance between the tracks of a compact disc.

8.2. Compact Disc Data Organization and Encoding

There are several levels of data organization on the compact discs, with each level built upon the previous one.

- At the lowest level, data are recorded as *pits* and *lands* on the disc. They are actually encoded by several methods to provide high recording density and reliable data recovery.
- At the next level, data are organized into sectors and tracks.
- The *High Sierra* specifications, which have been adopted later as the ISO 9660 standard, define a file system based on the disc sectors and tracks.

8.2.1. Data Recording and Encoding on the Medium

At the lowest level, the disc data are encoded to optimize the analog-to-digital conversion process. Goals of the data encoding at this level include the following:

1. *High information density*. This requires an encoding that makes the best possible use of the high, but limited, resolution of the laser beam and read optical assembly.
2. *Minimum inter-symbol interference*. This requires making the minimum run length, i.e. the minimum number of consecutive 0 bits or 1 bits, as large as possible.
3. *Avoiding a separate timing track*. For this, the data should be encoded so as to allow the clock signal to be regenerated from the data signal. Therefore, it is required to limit the maximum run length of the data so that data transitions will regenerate the clock signal.

A straightforward encoding would be to represent bits of 0 as lands and bits of 1 as pits. However, such an encoding does not meet neither goal (1), nor goals (2) and (3). For example, the integer 0 expressed as 32 bits of zero would have too long a run length and would not satisfy goal (3). The reason for this requirement is technological in nature. If, for instance, on a region there is a long stream of 0 bits (or 1 bits), the read assembly should have an extremely precise clock that switches exactly at the interval corresponding to the bit length (300 nm), in order for the next bit to be read correctly. This is not possible in the current stage of the technology.

For this reason, the length of pits and lands has been limited. By this limitation, passing from one state to another will occur frequently enough so that the clock precision will be sufficient. The clock is used to number the bits represented in a pit or on a land, depending on their length.

The bits are recorded on the medium using the NRZI (*Non Return to Zero Inverted*) coding. A bit of 1 is recorded as a state change of the medium, specifically, by a transition from a pit to a land (or conversely). A pit-to-land transition corresponds to a transition from an area with low reflectivity to an area with high reflectivity (the start of a new land). A land-to-pit transition corresponds to a transition from an area with high reflectivity to an area with low reflectivity (the start of a new pit). A bit of 0 is recorded as the absence of state change for the medium, which indicates the continuation of a land or pit.

Before recording, the data bits are modulated (encoded) in order to transform arbitrary binary sequences into sequences that have some desirable properties. For the recording medium used by optical discs, a convenient property is that the recorded sequences contain neither very short nor very long runs of successive zeros or ones. Very short runs lead to high frequencies of the signal generated by reading the information from the disc, which can cause errors in the bit detection module. Very long runs lead to timing inaccuracies of the PLL (*Phase-Locked Loop*) circuit that regenerates the clock signal required for synchronization and which adjusts the clock signal to each data signal transition.

The modulation method used consists of representing a data byte through 14 bits and it is referred to as *Eight-to-Fourteen Modulation* (EFM) coding. There are many more combinations of 14 bits (16,384) than there are combinations of 8 bits (256). To encode the 8-bit

combinations, 256 combinations of 14 bits were chosen that satisfy the requirement to avoid very short and very long runs of successive zeros. A table is used to make the association between the 256 valid combinations for a byte and the 14-bit sequences. For some decimal values, this association is illustrated in Table 8.1.

Table 8.1. Example encodings of 8-bit values through 14-bit sequences (EFM coding).

Decimal	8 Bits	14 Bits
0	0000 0000	01 0010 0010 0000
1	0000 0001	10 0001 0000 0000
2	0000 0010	10 0100 0010 0000
3	0000 0011	10 0010 0010 0000
4	0000 0100	01 0001 0000 0000
5	0000 0101	00 0001 0001 0000
6	0000 0110	00 0100 0010 0000
7	0000 0111	00 1001 0000 0000
8	0000 1000	01 0010 0100 0000
9	0000 1001	10 0000 0100 0000
10	0000 1010	10 0100 0100 0000

To avoid having two successive 1 bits between two adjacent EFM code sequences, after each sequence of 14 bits three merging bits are inserted, which are not taken into account when the data are read. Consequently, for representing a data byte 17 channel bits are required: 14 EFM bits and three merging bits.

The EFM codes satisfy the requirements of an RLL (*Run-Length Limited*) code. Run-length represents the number of identical bits (1s or 0s) for which the signal does not have transitions. In general, an RLL code is specified as (*d*, *k*) RLL. The two parameters, *d* and *k*, which specify an RLL code represent, respectively, the minimum and maximum number of 0 bits between two consecutive 1 bits. RLL coding terminology assumes NRZI coding, so that 1 bits indicate transitions and 0 bits indicate no transition; therefore, only runs of zero bits are counted. The RLL code that was used for choosing the EFM codes is (2, 10) RLL. As a consequence, between two 1 bits there will be at least 2 and at most 10 zero bits. As another consequence, the minimum length of pits and lands is 3 bits and their maximum length is 11 bits.

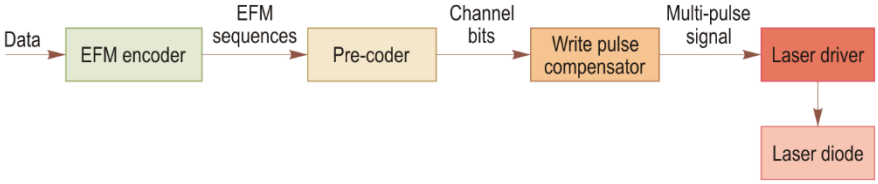

Figure 8.3. Structure of a recording system for compact discs.

For recording data on compact discs, each data byte is first encoded into an EFM code by an EFM encoder module (Figure 8.3). Then, the resulting binary sequences are transformed with a pre-coder module into the so-called channel bits. The channel bits are the actual bits that are written to the disc. A channel bit of 1 is associated with one physical state of the medium (e.g., with a pit) and a channel bit of 0 is associated with the other physical state (e.g., with a land). The channel bits are then sent to a write pulse compensator where they are modulated into a multi-pulse signal. Finally, the pulse waveform is sent to a laser driver circuit, which modulates the power of a laser beam to record the pits to the disc.

Binary Data (0x070A)	0000 0111		0000 1010	
EFM Sequences + Merging Bits	00 1001 0000 0000	100	10 0100 0100 0000	000
Channel Bits	00 1110 0000 0000	111	00 0111 1000 0000	000

Figure 8.4. Example EFM encoding for two data bytes and the corresponding stream of channel bits.

Figure 8.4 illustrates an example EFM encoding for two data bytes and the channel bits generated from the EFM sequences and the merging bits inserted. The stream of channel

bits can be partitioned into runs, where each run consists of a number of consecutive identical bits.

To avoid the need for a separate clock track, periodic synchronization is necessary. For this reason, data are broken up into blocks called *frames*, each beginning with a synchronization header. The frame is the basic synchronization unit. The structure of a frame is presented in Figure 8.5.

Synchronization	Control	Data (L samples)	CIRC	Data (R samples)	CIRC
27 bits	1 byte	12 bytes	4 bytes	12 bytes	4 bytes

Figure 8.5. Structure of a compact disc frame.

Except for the synchronization bits, all of the other fields in a frame are encoded through the EFM method. A frame consists of the following:

- A synchronization header: 27 bits;
- A control byte: 14 + 3 = 17 bits;
- 24 data bytes, divided in two fields: 2 × 12 × (14 + 3) = 408 bits;
- 8 bytes for error detection and correction code (CIRC – *Cross Interleaved Reed-Solomon Code*), divided in two fields: 2 × 4 × (14 + 3) = 136 bits.

In total, to represent the 24 bytes of a frame (192 bits) 588 channel bits are used.

For audio discs, each frame contains 6 digitized 16-bit samples of each audio channel. They are indicated in Figure 8.5 as *L samples* and *R samples*. These audio channels are usually the left and right components of a stereo pair. Each frame takes approximately 136.05 μs to play. This gives a sampling rate of 44.1 KHz for each channel.

8.2.2. First Level of Error Correction

Data errors can be due to dust, finger marks, production defects, or disc damage. A significant characteristic of these errors is that they often occur in long bursts.

The error detection and correction system used within the frames is called *Cross Interleaved Reed-Solomon Code* (CIRC), by the name of the mathematicians Reed and Solomon. This system is integrated at hardware level in the disc drives, and it has two components. The "*cross interleave*" component breaks up the long error bursts into many short errors, and the "*Reed-Solomon*" component provides the error correction.

Two *Reed-Solomon* decoders are used. As each frame is read from the disc, it is first decoded from 14 channel bits (the three merging bits are ignored) into data bytes. Then, the bytes from each frame (24 data bytes and 8 error correction bytes) are passed to the first *Reed-Solomon* decoder, which uses 4 of the error correction bytes and is able to correct one byte in error out of the 32. If there are no uncorrectable errors, the data are simply passed along. If there are such errors, the data are marked as being in error at this stage of decoding.

The 24 data bytes and the 4 remaining error correction bytes are then passed through *unequal delays* before sending them to the second *Reed-Solomon* decoder. These unequal delays result in an *interleaving* of the data that spreads long error bursts among several different passes through the second decoder. The delays are chosen such that error bursts up to 450 bytes long can be completely corrected. The second *Reed-Solomon* decoder uses the last 4 error correction bytes to correct any remaining errors in the 24 data bytes. After correcting the errors in the 24 data bytes by the second *Reed-Solomon* decoder, the correct byte order is restored.

If a larger area is destroyed, for audio discs data recovery can be achieved through interpolation. For example, if a string of values is recorded on the disc, e.g., 12, 16, and 24, and the middle value cannot be read (due to impurities or surface damage), this value can be inferred by interpolation as being 18, which is the average of values 12 and 24. Although the interpolated value is not exactly the correct one, the differences at the acoustic level would not be perceptible.

8.2.3. Sector Format

A number of 98 frames make a physical unit referred to as a *sector*. Therefore, a sector contains 98 control bytes, 24×98 = 2352 data bytes and 8×98 = 784 error detection and correction bytes.

The basic technology used for CD-ROM data discs is the same as for audio discs. However, each sector must be directly accessed, and for this purpose a header is provided for each sector. Also, the data must have a higher degree of integrity, and for this reason a larger number of error detection and correction bytes are allocated.

For direct access to each sector, synchronization bytes and a header containing the sector address are used. The 12 synchronization bytes have predefined values: the first and the last byte are 0, and the 10 bytes in the middle are 0xFF. This sequence allows the disc drive to identify the beginning of a new sector.

In the structure of a sector, after the synchronization bytes the sector header follows. This consists of 4 fields, and each of them occupies a byte:

- M field of the absolute address;
- S field of the absolute address;
- F field of the absolute address;
- Data mode field.

The absolute address of a sector is coded in MSF form (*Minute*, *Second*, *Frame*). The address is divided into three fields, and each field contains two BCD digits. The M field contains the most significant part of the absolute address (the minutes), and has values between 00 and 99. The S field contains the seconds and has values between 00 and 59. The F field contains the less significant part of the absolute address (the frame number), with values between 00 and 74. The MSF address of the beginning of data tracks must be accurate, but for audio tracks this address has a tolerance of ± 75 sectors.

The data mode field indicates the type of information in the sector (data or audio/video), as well as the audio encoding mode.

There are two modes of data organization on CD-ROM discs. In Mode 1, provided for data tracks (programs), after the synchronization bytes and the sector header 2048 bytes of data follow. Then there is an auxiliary data field, which contains the *Error Detection Code* (EDC) and the *Error Correction Code* (ECC). Between the two codes there are 8 bytes of 0. Because 75 sectors are required for one second, the total capacity of a 74-minute CD-ROM disc is 2048 ∗ 75 ∗ 74 ∗ 60 = 681,984,000 bytes or 650.39 MB. The structure of a sector in Mode 1 is presented in Figure 8.6.

Synchronization	Header	Data	EDC	0	ECC
12 bytes	4 bytes	2048 bytes	4 bytes	8 bytes	276 bytes

Figure 8.6. Sector structure of CD-ROM discs in Mode 1.

In Mode 2, provided for less critical applications (audio and video), the auxiliary field is also employed for the user data, instead of the error detection and correction codes. This is because, usually, it is not necessary to ensure a high integrity for the audio and video information. In this case, a sector contains 2048 + 288 = 2336 data bytes. The structure of a sector in Mode 2 is presented in Figure 8.7.

Synchronization	Header	Data
12 bytes	4 bytes	2336 bytes

Figure 8.7. Sector structure of CD-ROM discs in Mode 2.

On the same disc there may be data, as well as audio tracks, video images, etc. Such discs are *mixed mode discs*, which comply with the ISO/IEC 10149 standard. With these discs, the first track contains data (in Mode 1), and the other tracks contain audio or video

information (in Mode 2). The problem that can occur in this case is that some CD players do not automatically skip over the first track, and the possibility exists to damage the audio equipment if the first track is played with a CD audio player at high volume.

A common CD-ROM drive, which is able to read mixed mode discs cannot read data at the same time with the audio playback, because the data and audio information are located on different tracks. For instance, such a drive does not allow a synchronized playback of sound and images. To solve this problem, a new format has been added to the specifications of the CD-I (CD *Interactive*) and CD-ROM/XA (CD-ROM *eXtended Architecture*) discs. This format has been obtained by splitting sector Mode 2 into two forms: Form 1 and Form 2.

Form 1 is used for numeric data and is similar to Mode 1 of CD-ROM discs, but there is an additional identifier in the sector header (Figure 8.8).

Synchronization	Header	Sub-header	Data	EDC	ECC
12 bytes	4 bytes	8 bytes	2048 bytes	4 bytes	276 bytes

Figure 8.8. Sector structure of CD-ROM/XA discs in Mode 2, Form 1.

The 8 bytes of 0 within the CD-ROM sector, located between the EDC and ECC fields, are placed immediately after the header bytes, forming a *sub-header* that is used to identify the sector type (of Form 1 or Form 2).

Form 2 is used for audio and video information, and is similar to Mode 2 of CD-ROM discs. The 276 bytes of the ECC field are used for storing data, so that a sector contains 2324 bytes of usable data (Figure 8.9).

Synchronization	Header	Sub-header	Data	EDC
12 bytes	4 bytes	8 bytes	2324 bytes	4 bytes

Figure 8.9. Sector structure of CD-ROM/XA discs in Mode 2, Form 2.

The s*ub-header* contains information such as the file number, channel number, a sub-mode byte and control information.

- The *file number* can range between 0 and 255; 0 indicates a file that will be read continuously (non-interleaved).
- The *channel number* is provided to allow a separate or combined read of several sectors and their selection in real time. Channels between 0 and 15 are reserved for audio data. For video data and numeric data the channels between 16 and 31 are used.
- The *sub-mode* byte allows to identify the sector type (data or audio/video) and to store some parameters.
- The *control information* refers to the recording mode (mono/stereo) and to the sound level.

The two types of sectors, data and audio/video, can be placed alternately on a single track in Mode 2. The data are written in Form 1, and the audio/video information is written in Form 2. This interleaving technique allows the synchronized playback of images and audio.

8.2.4. Second Level of Error Correction

For audio discs, an uncorrected error at the frame level results in a noise, often imperceptible, during listening. For data discs, an uncorrected error may cause the disc to become unusable. The 10^{-9} error rate ensured by the CIRC method is not acceptable for data discs. To improve this rate, for CD-ROM discs a second level of error detection and error correction (EDC/ECC) is provided for data recorded in Mode 1.

For the second level of error correction 280 additional bytes are allocated, 4 bytes for error detection (EDC) and 276 bytes for error correction (ECC). The error correction technique used is called *Layered Error Correction* (L-EC); the error rate ensured is 10^{-12}.

The error detection code is a *Cyclic Redundancy Check* (CRC). It occupies the first four bytes of the auxiliary data field and provides a very high probability that uncorrected errors will be detected. The error correction code is based on the same principle as the first level of error correction, in that interleaving and *Reed-Solomon* coding are used. It occupies the final 276 bytes of the auxiliary data field.

8.2.5. Sub-Channel Information Formats

Each bit of a control byte is identified by a letter: *P*, *Q*, *R*, *S*, *T*, *U*, *V*, and *W*. Bits with the same rank in all of the control bytes (and which have assigned the same letter) make a *sub-channel*. Thus, all bits that occupy the first position in all the 98 control bytes make *sub-channel P*. The bits of this sub-channel are used separately and can be employed for audio muting control or as audio track separator. The bits in the second position make *sub-channel Q*. The last 6 bits are combined into *sub-channel R-W*. This sub-channel is used by several variants of CD digital audio discs to store information such as the text of songs or some graphical data.

Sub-channel *Q* has a more complex structure. For a sector, the sub-channel *Q* bits are organized into an information block. This block consists of 98 bits, one bit from each frame of the sector. The structure of this block is presented in Table 8.2.

Table 8.2. Sub-channel *Q* information block.

Field	Meaning
S0, S1	2 bits of sub-channel synchronization
CONTROL	4 bits to define the type of information within the current track: x0xx = audio track 01xx = data track 11xx = reserved xx0x = digital copy prohibited xx1x = digital copy permitted
ADR	4 bits to define the format used by the DATA-Q field
DATA-Q	72 bits of data
CRC	16 bits for the CRC of CONTROL, ADR, and DATA-Q fields

There are three possible formats for the DATA-Q field. These are described next.

Format 1

The first format must be present in at least 9 of 10 consecutive sectors. Two different structures of this format are possible. For the *lead-in* area, the structure of the DATA-Q field is illustrated in Figure 8.10.

ADR	DATA-Q								
0001	TNO	POINT	MIN	SEC	FRAME	ZERO	PMIN	PSEC	PFRAME

Figure 8.10. Format 1 of the DATA-Q field for the *lead-in* area.

For the other tracks (data, audio, and *lead-out* area) the structure of the DATA-Q field is illustrated in Figure 8.11.

ADR	DATA-Q								
0001	TNO	INDEX	MIN	SEC	FRAME	ZERO	AMIN	ASEC	AFRAME

Figure 8.11. Format 1 of the DATA-Q field for data tracks, audio tracks, and the *lead-out* area.

TNO (*Track Number*) represents the track number and is expressed by two BCD digits.

INDEX (*Index to* TNO) represents the subdivision within the TNO track. Value 00 indicates a pause or *pre-gap* area. In the *lead-out* area the index value is 01. Within an audio

track, the first value of the index is 01 and this value can be incremented only by 1. In a data track the index has the value 01.

The MIN, SEC, FRAME fields contain the running time relative to the beginning of the track in minutes, seconds, and frames, each expressed as two BCD digits (the relative address of the sector coded in MSF form). In the pause and *pre-gap* areas, the logical addresses are negative and decrease in absolute value, reaching the value 0 at the end of the particular area. In other areas, including the *lead-in* and *lead-out* areas, the addresses increase. One second is subdivided into 75 frames.

The ZERO field contains the value 00.

The AMIN, ASEC, AFRAME fields contain the absolute address of the sector in MSF form.

The POINT, PMIN, PSEC, PFRAME fields contain the table of contents during the *lead-in* area (TNO = 0). This table of contents is continuously repeated in the *lead-in* area. In each table of contents, the individual fields are repeated three times.

The value given by PMIN, PSEC, and PFRAME represents the starting point of the track pointed to by POINT, as an absolute value on the time scale, with a precision of ± 1 second. The starting position of a track is the first position with the new value of the track and the index different than 00.

- If POINT = 0xA0, PMIN indicates the track number for the first piece of audio on the disc; PSEC and PFRAME are 0.
- If POINT = 0xA1, PMIN indicates the number of the last track on the disc; PSEC and PFRAME are 0.
- If POINT = 0xA2, PMIN, PSEC and PFRAME contain the starting point of the *lead-out* area.

Format 2

The second format of the DATA-Q field is optional, and it has the structure of Figure 8.12.

ADR	DATA-Q														
0010	N1	N2	N3	N4	N5	N6	N7	N8	N9	N10	N11	N12	N13	ZERO	AFRAME

Figure 8.12. Format 2 of the DATA-Q field.

N1..N13 represents the disc *catalog number* expressed in 13 BCD digits, according to the uniform product code values (UPC/EAN bar coding). These values are controlled by the *Uniform Product Code Council* and the *European Article Number Council*. The catalog number does not change on a disc. If no catalog number has been encoded, N1..N13 contain digits of 0.

The ZERO field contains 12 bits of 0.

The AFRAME field has the same meaning as in the first format (two BCD digits running from 00 to 74). In the *lead-in* area (TNO = 00), all the 8 bits of this field are 0.

Format 3

The third format of the DATA-Q field is also optional, and is used to assign a unique number to an audio track by means of the *International Standard Recording Code* (ISRC). The ISRC code recorded on the medium is defined in Figure 8.13. In the *lead-in* and *lead-out* area, this format is not present on the disc. The ISRC code changes immediately after the track number (TNO) has changed.

ADR																
0011	I1	I2	I3	I4	I5	0	0	I6	I7	I8	I9	I10	I11	I12	ZERO	AFRAME

Figure 8.13. Format 3 of the DATA-Q field (ISRC code).

The I1 - I12 fields define the ISRC code, as follows:

I1..I2: Country code;
I3..I5: Owner code;
I6..I7: Year of recording;
I8..I12: Serial number of the recording.

Each character in the I1..I5 fields is represented through a 6-bit coding. The I6..I12 characters are coded in BCD. The ZERO field contains 4 bits of 0. The AFRAME field has the same meaning as in formats 1 and 2.

8.2.6. Disc Organization

A *track* may be viewed as a partition of the disc address space. A CD may contain from 1 to 99 tracks, numbered consecutively. However, the first information track may have a number greater than 1. Tracks have a minimum length of 300 sectors. All the sectors of a track are required to be of the same type (to contain either data or audio/video information), and be recorded in the same mode. Each change in the type of information on the disc requires a change in track number. A disc containing both data and audio/video information will have at least two tracks, one for data and one for audio/video.

Between tracks encoded with different types of information *transition areas* must exist. In addition, transition areas may be used at the beginning or end of any track. For audio tracks the transition areas are called *pause* areas. For data tracks, transition areas are called *pre-gap* and *post-gap* areas. The IEC 908 and ISO/IEC 10149 standards specify minimum time durations for these areas. Maximum time durations are not specified. Transition areas are formatted and the logical address continues to increment through transition areas.

A CD is composed of three areas: the *lead-in* area, the data area, and the *lead-out* area. The *lead-in* area of the disc is designated track zero, but it is not addressable through the drive command set. The sub-channel *Q* of this area contains the *Table Of Contents* (TOC) of the disc. The *lead-out* area is designated track 0xAA, but it is not addressable through the drive command set.

The table of contents gives the absolute address of the first information sector of each track. Control information for each track is also given in the TOC (audio/data, method of audio encoding, etc.). However, the TOC does not distinguish between the different modes of data tracks (e. g. Mode 1 vs. Mode 2). The absolute address of a sector is coded in MSF (*Minute*, *Second*, *Frame*) form and indicates the time from the beginning of the disc in minutes, seconds, and frames.

An *index* is a partition of a track. *Pre-gap* areas are encoded with an index value of zero. Pause areas at the beginning of audio tracks are also encoded with an index value of zero. The first information sector of a track has an index value of 1. Consecutive values up to 99 are permitted. Index information is not stored in the TOC.

The sub-channel information which is part of each sector (more precisely, in sub-channel *Q*) includes the relative address of the sector, giving the distance from the first information sector of the track. This value is negative in the *pre-gap* areas and increases to 0 in these areas.

8.3. Overview of the ATA Packet Interface

The aim of developing the ATA Packet Interface (ATAPI) was to provide a more extendable and more general interface than the ATA interface. ATAPI devices must support both the master and slave modes of operation. These devices use the same signals that are specified in the ATA interface standards.

Although for attaching an optical disc drive through the ATAPI interface the same ATA physical interface is used, the logical interface is different, and it supports additional capabilities. The devices connected to the ATA interface use eight registers to communicate with the computer, and those registers contain the command and all parameters needed for an

operation. However, these registers are not enough to pass all the information needed for controlling various peripheral types. To eliminate this disadvantage, the commands for ATAPI devices are sent as structures referred to as *command packets*. These commands supplement the existing ATA commands.

A new command called PACKET has been added, which allows a command packet to be sent to an ATAPI device. The packet is sent by writing multiple times to the data register. This technique reduces the number of register addresses needed. Although all the commands for ATAPI devices could be sent via this mode, these devices must also accept some existing ATA commands, as well as the main ATA command protocols, so that there will be minimal changes needed to the existing drivers. This minimal set of ATA commands is reduced compared to the minimal set defined for the ATA interface, but is sufficient for normal operation.

On the other hand, ATAPI devices will not respond to the IDENTIFY DEVICE or READ commands of the ATA interface, which allows the BIOS program and older drivers to ignore these devices and not confuse data from them with data from ATA drives. Commands that are not supported will not be executed, and an interrupt request will be generated to signal the completion of the command with an error status (*ABORTED*).

To identify an ATAPI device, the IDENTIFY PACKET DEVICE command is provided; the information returned by this command is used by the low-level drivers to perform hardware configuration of the ATA interface. This information has a structure similar to that of the information returned by the IDENTIFY DEVICE command of the ATA interface, for compatibility reasons. Since the information returned by the INQUIRY command of the SCSI interface cannot be returned by the IDENTIFY PACKET DEVICE command of the ATA Packet Interface, the INQUIRY command is recognized by the ATA Packet Interface to be used by the higher-level drivers.

Although the ATA Packet Interface uses packet structures defined in the SCSI standards, most of the features of the SCSI protocol are not used. The major differences from the SCSI interface are the following:

- An ATAPI peripheral has the function of a slave device during an operation, as opposed to the master function of a SCSI peripheral.
- There are no phases of operation, and no messages are sent.
- Disconnect / reconnect operations are not possible.
- Command packets are 12 bytes in length, as opposed to the SCSI interface, where packets can be 6, 10, 12, or 16 bytes in length. However, 16-byte packets are also defined for compatibility with future devices. The size of packets accepted by a device is defined in word 0 of the information returned by the IDENTIFY PACKET DEVICE command.

To configure the ATAPI devices, the MODE SELECT and MODE SENSE commands of the SCSI standards are used. The combination of the SCSI MODE SELECT command and the ATA SET FEATURES command ensures the functionality necessary and is compatible with most existing BIOS programs and operating systems' drivers.

8.4. ATA Packet Interface Registers

Communication with peripheral devices is achieved via I/O registers. Except for the data register, which is 16 bits wide, all the other registers of the ATA Packet Interface are accessed through 8-bit I/O operations.

Table 8.3 presents the registers of the ATA Packet interface. These registers are the same as the registers of the ATA interface, although the names of some registers are different. These registers are described in the laboratory work *ATA Interface*, Sections 7.6.1-7.6.12.

Table 8.3. ATA Packet Interface registers.

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

8.5. ATA Packet Interface Command Execution

The control of an ATAPI device may be performed using one of two methods, via the regular ATA commands, or via the ATAPI PACKET command. For both methods, the devices attached to the interface are programmed by the host computer to execute commands and return status to the host computer at command completion. When two devices are daisy-chained on the interface, the commands are sent in parallel to both devices, but only the device selected by the DEV bit of the Device register will execute the command.

To issue the PACKET command, the normal rules and protocol of the ATA interface are used, but after the command has been issued, a new set of rules applies:

1. The command to be executed by the device will be sent as a packet via the Data register and not via the other registers of the interface.
2. The command parameters are sent through the command packet as well as via the registers of the interface.
3. A byte count is used to determine the amount of data that the host computer must transfer at each DRQ interrupt.
4. To indicate that the DMA transfer mode will be used, the Features register is used instead of using different operation codes.
5. The final status is returned to the host computer as a new interrupt after the last datum has been transferred, rather than along with the last block of data.

These new rules apply until the status that indicates command completion has been read by the host computer. After this, the register definitions and the protocol used will be those defined by the ATA standard.

The PACKET command is issued as any regular ATA command, by initializing the interface registers, setting the drive selection bit in the Device register, and writing the command byte to the Command register. With regular ATA commands, the DRQ bit in the Status register is set (and, optionally, an interrupt is generated) to indicate that the command parameters can be transferred to/from the device. With the PACKET command, the first setting of the DRQ bit indicates that the command packet data must be sent to the device. After sending the packet, the command proceeds as a regular ATA command.

If, while checking the BSY bit, the device remains in a state where it cannot accept a command for more than 5 seconds, the host computer must reset the device.

Since the length of data transferred in PIO mode to and from an ATAPI device is controlled by the host computer, the capability to transfer a variable number of bytes has been created by using the Byte Count registers. The device indicates to the host computer the number of bytes that should be transferred on each DRQ interrupt. Before the data transfer, the host computer must read the Byte Count Low and Byte Count High registers and take into account the length requested. The ATAPI device and the host computer will have their own byte counts and will transfer data until those counts reach zero. For some commands, such as

MODE SENSE, the host computer does not know the amount of data that will be transferred and must rely on the byte count supplied by the device to transfer the correct number of bytes.

By using the Byte Count registers, the host computer is also capable to indicate to the device the maximum number of bytes it can receive in a single DRQ packet or the preferred packet size. For all commands that require data to be transferred, the host computer must load the Byte Count registers with the desired length before issuing the PACKET command. This length is used by the ATAPI device as the maximum size for each PIO or DMA data packet. The device may transfer packets smaller than the size indicated by the host computer.

If the device requests a larger number of bytes to be transferred than required by the command protocol, the host computer must extend the data sent to the device and receive the extra data when reading data from the device. The device is not responsible for extending the data, only transferring the number of bytes specified by the host computer.

Some ATAPI commands are immediate. For these commands, the device returns the command completion status immediately, continuing the actual execution of the command. The following should be taken into account:

- If a new ATAPI command is issued after an immediate command that reports completion before the actual completion (SEEK, PLAY AUDIO, etc.), the new command is stored by the device in the command queue.
- When a new ATA command is written to the Command register before a command has completed, the executing command will stop execution, the new command will be aborted, and the ABRT (*Aborted Command*) bit will be set in the Error register.
- New ATAPI commands received while a previous ATAPI command is still executing will cause both commands to be aborted, and the CHK (*Check Condition*) bit will be set in the Status register.

There are devices that support command overlapping to improve system performance. These devices release the bus before completing an executing command, allowing the bus to be used by the other device.

8.6. Protocols Used for ATAPI Commands

8.6.1. ATAPI Protocol for Non-Data Commands

This protocol is used for commands such as SEEK, PLAY AUDIO, or START/STOP UNIT, whose execution involves no data transfer. Assuming that interrupt generation is not enabled, from the host computer's viewpoint, the 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, the software should abandon the execution of the protocol.
2. Software initializes the Device register. In the Device register, the DEV bit should be reset to 0 for accessing drive 0, or it should be set to 1 for accessing drive 1.
3. Software writes the code of the PACKET command to the Command register.
4. Software reads the Status register and waits until the BSY bit becomes 0.
5. Software reads the Status register and checks the DRQ bit. If the DRQ bit is 0, the command has not been recognized by the drive, and the execution of the protocol is completed. If the DRQ bit is set, software continues with Step 6.
6. Software writes the command packet to the Data register, one word at a time. If the command packet is defined as an array of bytes, two consecutive bytes should be packed in a word before writing the word to the Data register. The first byte should

be placed in the low byte of the word, and the second byte should be placed in the high byte of the word.

7. Software waits for a time corresponding to a PIO transfer cycle. For instance, it may read the Alternate Status register, ignoring the read result.
8. 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, the software should abandon the execution of the protocol.
9. Software reads the Status register and waits until the DRQ bit becomes 0, when the command execution is completed.

8.6.2. ATAPI Protocol for Input in PIO Mode

This protocol is used by commands such as INQUIRY or READ. Execution includes the transfer of some unknown number of data bytes from the device to the host computer. Assuming that interrupt generation is not enabled, from the host computer's viewpoint, the protocol for input 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, the software should abandon the execution of the protocol.
2. Software initializes the Device register, the Byte Count Low register, and the Byte Count High register. In the Device register, the DEV bit should be reset to 0 for accessing drive 0, or it should be set to 1 for accessing drive 1. The Byte Count registers should be loaded with the maximum number of bytes that the drive should transfer with each assertion of the DRQ bit. For the commands used in this laboratory work, these registers should be loaded with the total number of bytes that should be transferred for the particular command that will be issued.
3. Software writes the code of the PACKET command to the Command register.
4. Software reads the Status register and waits until the BSY bit becomes 0.
5. Software reads the Status register and checks the DRQ bit. If the DRQ bit is 0, the command has not been recognized by the drive, and the execution of the protocol is completed. If the DRQ bit is set, software continues with Step 6.
6. Software writes the command packet to the Data register, one word at a time. If the command packet is defined as an array of bytes, two consecutive bytes should be packed in a word before writing the word to the Data register. The first byte should be placed in the low byte of the word, and the second byte should be placed in the high byte of the word.
7. Software waits for a time corresponding to a PIO transfer cycle. For instance, it may read the Alternate Status register, ignoring the read result.
8. 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, the software should abandon the execution of the protocol.
9. Software reads the Status register and checks the DRQ bit. If the DRQ bit is 0, the drive has completed the command with an error, and the execution of the protocol is completed. If the DRQ bit is set, software continues with Step 10.
10. Software reads the Byte Count Low and Byte Count High registers and initializes a variable of type `WORD` with the contents of these registers. Software transfers a data block in a loop, by reading the Data register and storing it in a buffer, using the variable initialized previously as the loop count. If the buffer is defined as an array of bytes, the word read from the Data register should be stored in two consecutive bytes

of the buffer. The low byte of the word should be stored in the first byte, and the high byte of the word should be stored in the second byte.

11. Software waits for a time corresponding to a PIO transfer cycle. For instance, it may read the Alternate Status register, ignoring the read result.
12. 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, the software should abandon the execution of the protocol.
13. Software reads the Status register and if the DRQ bit is 1, proceeds with the transfer of a new data block (Step 10). If the DRQ bit is 0, command execution is completed.

8.7. ATAPI Commands

8.7.1. Identification of ATAPI Devices

When ATAPI devices are powered on, a hardware or software reset is performed, or the ATA DEVICE RESET and EXECUTE DEVICE DIAGNOSTIC commands are executed, these devices load into some registers of the interface a signature that allows to identify ATAPI devices. Moreover, the ATA IDENTIFY DEVICE command will not be executed by ATAPI devices; these devices will set the ABRT bit in the Error register and will load some of the interface registers with the same signature as when executing the DEVICE RESET command.

The signature that identifies ATAPI devices and the ATAPI registers this signature is loaded into are shown in Table 8.4 for the parallel ATA Packet Interface and for the serial ATA Packet Interface.

Table 8.4. The characteristic signature of ATAPI devices.

ATAPI Register	Parallel ATA Packet Interface	Serial ATA Packet Interface
Interrupt Reason	0x01	0x01
LBA Low	0x01	0x01
Byte Count Low	0x14	0x69
Byte Count High	0xEB	0x96

To detect whether a particular device attached to an ATA interface is an ATAPI device, the IDENTIFY DEVICE command could be issued to that device, checking whether this command is recognized by the device. If the command execution is abandoned with the *ABORTED* error status, indicated by setting the ABRT bit in the Error register, the Byte Count Low and Byte Count High registers contain the characteristic signature of ATAPI devices. If these registers contain the values shown in Table 8.4, the device is an ATAPI device, and it supports the PACKET command. Next, the IDENTIFY PACKET DEVICE command (code 0xA1) could be issued to find out detailed information about that ATAPI device, using the same protocol as that for executing the IDENTIFY DEVICE command.

To issue the IDENTIFY DEVICE command, the ATA protocol for input in PIO mode can be used. This protocol is described in the laboratory work *ATA Interface*, Section 7.7.2. For clarity, the operations required for issuing and executing this command are presented below, supplemented with the operations required to detect command abortion by an ATAPI device.

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, the software should abandon the execution of the operation.
2. Software sets the DEV bit in the Device register depending on the drive the command is issued to (DEV = 0 for drive 0, DEV = 1 for drive 1).
3. Software writes the IDENTIFY DEVICE command code to the Command register.
4. Software reads the Status register and waits until the BSY bit becomes 0.

5. Software reads the Status register and checks the DRQ bit. If the DRQ bit is set, meaning that the drive is an ATA drive because it has recognized the command, software proceeds with Step 6. If the DRQ bit is 0, the command has not been recognized by the drive. In this case, the software checks whether the ABRT bit in the Error register is set.
   - If the ABRT bit is set, the software reads the contents of the Byte Count Low and Byte Count High registers. If these registers contain the values shown in Table 8.4, the drive is identified as an ATAPI drive; otherwise, the drive is not an ATAPI drive. In both cases, the sequence of operations is completed.
   - If the ABRT bit is 0, there is no drive present with the number specified in the Device register, and the sequence of operations is completed.
6. Software transfers 256 data words by reading repeatedly the Data register. When all the 256 words have been read, the sequence of operations is completed.

8.7.2. IDENTIFY PACKET DEVICE Command

The IDENTIFY PACKET DEVICE command is similar to the IDENTIFY DEVICE command of the ATA interface, but it is only recognized by ATAPI drives. After receiving this command, the drive prepares a block of 256 words with information about the drive: 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 ATA protocol for input in PIO mode (described in the laboratory work *ATA Interface*, 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 to the Command register.

ATAPI drives will not report an error after executing this command. ATA drives will set the ABRT bit in the Error register.

Table 8.5 presents the meaning of some of the 16-bit words that are returned by the IDENTIFY PACKET DEVICE command. Some parameters of the 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.

Table 8.5. Meaning of words returned by the IDENTIFY PACKET DEVICE command.

Word	Meaning
0	General configuration information
10-19	Serial number (20 ASCII characters)
23-26	Firmware revision (8 ASCII characters)
27-46	Model number (40 ASCII characters)

Bits 1..0 of word 0 indicate the length of the ATAPI command packets recognized by the drive. If these bits are 00, the length of the command packets is 12 bytes, and if these bits are 01, the length of the command packets is 16 bytes; other values are reserved.

8.7.3. ATAPI Command Packet Structure

An ATAPI command is sent to an ATAPI device through a packet. For some commands, the packet is followed by a list of parameters that is sent after generating an interrupt as the result of sending the command packet.

In general, ATAPI command packets are 12 bytes long. Figure 8.14 presents the typical structure of an ATAPI command packet. The fields have the same meaning as the fields of command descriptor blocks of the SCSI interface. The first byte of the packet always contains an *operation code*, representing the code of the command to be executed by the device. The *logical block address* within a logical unit starts with block zero and must be contiguous up to the last logical block of the logical unit. The size of logical block addresses is 32 bits. The *transfer length* specifies the length of data that must be transferred, either in number of blocks or in number of

bytes, depending on the command. The *allocation length* specifies the maximum number of bytes allocated by the host computer for the data sent by the device. This length is used to limit the number of bytes returned by the device.

The last byte of the command packet contains a *control code*. For regular commands, this control code must be set to 0. Reserved fields must be set to 0.

Bit Byte	7	6	5	4	3	2	1	0
0	Operation Code							
1	Reserved							
2	MSB			Logical Block Address				
3								
4								
5								LSB
6	MSB			Transfer Length, Allocation Length				
7								
8								
9								LSB
10	Reserved							
11	Control Code							

Figure 8.14. Typical structure of a 12-byte ATAPI command packet.

8.7.4. ATAPI Command List for CD/DVD Drives

The ATAPI commands for CD/DVD drives are derived from the SCSI command set. Except for the MSF addressing mode, the interface uses logical addressing for all the data blocks. Each device can be interrogated to determine the number of blocks it contains.

Table 8.6 presents the list of ATAPI commands for CD/DVD drives.

Table 8.6. ATAPI commands for CD/DVD drives.

Command	Operation Code	Type
BLANK	0xA1	E
CLOSE TRACK/SESSION	0x5B	R
FORMAT UNIT	0x04	E
INQUIRY	0x12	M
LOAD/UNLOAD CD	0xA6	C
MECHANISM STATUS	0xBD	M
MODE SELECT (10)	0x55	M
MODE SENSE (10)	0x5A	M
PAUSE/RESUME	0x4B	A
PLAY AUDIO (10)	0x45	A
PLAY AUDIO (12)	0xA5	A
PLAY AUDIO MSF	0x47	A
PLAY CD	0xBC	O
PREVENT/ALLOW MEDIUM REMOVAL	0x1E	M
READ (10)	0x28	M
READ (12)	0xA8	M
READ BUFFER CAPACITY	0x5C	O
READ CD	0xBE	M
READ CD MSF	0xB9	M
READ CD RECORDED CAPACITY	0x25	M
READ DISC INFORMATION	0x51	R
READ HEADER	0x44	M
READ MASTER CUE	0x59	O
READ SUB-CHANNEL	0x42	M
READ TOC/PMA/ATIP	0x43	M
READ TRACK INFORMATION	0x52	R
REPAIR TRACK	0x58	O
REQUEST SENSE	0x03	M
RESERVE TRACK	0x53	R

Command	Operation Code	Type
SCAN	0xBA	A
SEEK	0x2B	M
SEND CUE SHEET	0x5D	O
SEND OPC INFORMATION	0x54	O
SET CD SPEED	0xBB	R
START/STOP UNIT	0x1B	M
STOP PLAY/SCAN	0x4E	M
SYNCHRONIZE CACHE	0x35	R
TEST UNIT READY	0x00	M
WRITE (10)	0x2A	R
WRITE (12)	0xAA	R
M: Mandatory command O: Optional command A: Mandatory command for audio drives R: Mandatory command for CD-R/RW drives E: Mandatory command for CD-RW drives C: Mandatory command for disc changers		

8.7.5. PLAY AUDIO MSF Command

The PLAY AUDIO MSF command requests a drive containing an audio CD disc to begin an audio playback operation. This is an immediate command, allowing command overlapping. At actual completion of the operation, the drive sets the SERV bit (bit 4) in the Status register. The protocol used is the ATAPI protocol for non-data commands.

The structure of the command packet is shown in Figure 8.15.

Bit Byte	7	6	5	4	3	2	1	0
0	Operation Code (0x47)							
1..2	Reserved							
3	Starting M Field							
4	Starting S Field							
5	Starting F Field							
6	Ending M Field							
7	Ending S Field							
8	Ending F Field							
9	Control Code							
10..11	Reserved							

Figure 8.15. Structure of the PLAY AUDIO MSF command packet.

The starting M, S, and F fields specify the absolute MSF address where the operation shall begin, and the ending M, S, and F fields specify the absolute MSF address where the operation shall end. If the starting M, S, and F fields are set to 0xFF, the starting address will be the current location of the optical head. This allows the ending address to be changed without interrupting the current operation.

If the starting MSF address is equal to the ending MSF address, no operation is executed. If the starting MSF address is greater than the ending MSF address, the command will be terminated with the *CHECK CONDITION* status, and the ERR/CHK bit of the Status register will be set.

8.7.6. READ (12) Command

The READ (12) command requests the CD/DVD drive to transfer data from the drive to the host computer. The command packet has the typical structure of a 12-byte command packet (Figure 8.14). Bytes 6..9 contain the transfer length. The protocol used is the ATAPI protocol for input in PIO mode.

The transfer length specifies the number of contiguous logical blocks that shall be transferred. Although the CD/DVD drive may return different type of information, this com-

mand will only transfer the data part within the sector. This data field is always 2048 bytes in length for sectors in Mode 1 and Mode 2, Form 1, which are the only sector types allowed. For other types of sectors, the ILI (*Illegal Length Indication*) bit (bit 0) will be set in the Error register if an attempt is made to read them with the READ command.

8.7.7. READ CD RECORDED CAPACITY Command

The READ CD RECORDED CAPACITY command allows the host computer to request information regarding the recorded capacity of a CD/DVD. The protocol used for this command is the ATAPI protocol for input in PIO mode. The command returns the address of the last logical block, based on the table of contents data. If the last track is an audio track, the returned value may be inexact, because there is a tolerance of ± 75 frames when addressing audio data, according to the specifications of the medium. Therefore, the last block may be at a distance of ± 75 frames to the actual end of the track. For CD/DVD drives, this implementation allows a faster response.

The structure of the command packet is shown in Figure 8.16. The Reserved and Control fields should be set to 0.

Bit Byte	7	6	5	4	3	2	1	0
0	Operation Code (0x25)							
1	Reserved							
. . .								
8								
9	Control							
10..11	Reserved							

Figure 8.16. Structure of the READ CD RECORDED CAPACITY command packet.

The data returned by this command have the structure shown in Figure 8.17. The command reports a block length of 2048 bytes.

Bit Byte	7	6	5	4	3	2	1	0
0..3	Last Logical Block Address (Byte 0 - MSB)							
4..7	Block Length in Bytes (Byte 4 - MSB)							

Figure 8.17. Format of data returned by the READ CD RECORDED CAPACITY command.

8.7.8. READ TOC/PMA/ATIP Command

The READ TOC/PMA/ATIP command requests the CD/DVD drive to transfer data from the *Table of Contents* (TOC), the *Program Memory Area* (PMA), or the *Absolute Time in Pre-Groove* (ATIP) area. PMA is an additional area present on CD-R/DVD-R and CD-RW/DVD-RW discs, which contains the track numbers for the recorded titles and their start and end positions. The ATIP area contains information such as the absolute start time for the *lead-in* and *lead-out* areas in MSF form, the recommended laser power for writing, and the lowest and highest usable recording speed. The structure of the command packet is shown in Figure 8.18. The protocol used for this command is the ATAPI protocol for input in PIO mode.

The allocation length indicates the maximum number of bytes that shall be returned by this command. To specify the type of data that shall be returned, the four least significant bits of byte 2 (Format) are used. For multi-session discs and/or *Kodak Photo* CDs, the format 0001b may be used. For drives that do not support multi-session discs, the first session number must be equal to last session number from the TOC information returned. The definition of the Format field is the following:

0000b For this format, the Track/Session Number field specifies the number of the first track for which the data shall be returned. If this value is 0, the table of contents will start with the first track data. For multi-session discs, the command will return the table of

contents data for all sessions. The data are returned in increasing order of the track number.

Bit / Byte	7	6	5	4	3	2	1	0
0	Operation Code (0x43)							
1	Reserved						MSF	Reserv.
2	Reserved				Format			
3..5	Reserved							
6	Track/Session Number (Hex)							
7	MSB	Allocation Length						
8	(Byte 7 - MSB)							LSB
9	Control Code							
10..11	Reserved							

Figure 8.18. Structure of the READ TOC/PMA/ATIP command packet.

0001b This format is intended for multi-session discs. The format returns the first complete session number, last complete session number, and last complete session starting address. In this format, the Track/Session Number field should be set to 0x00.

0010b Specifies to return the sub-channel *Q* data in the *lead-in* (TOC) area starting from the session number specified in the Track/Session Number field.

0011b Specifies to return the sub-channel *Q* data in the PMA area. The Track/Session Number field should be set to 0x00.

0100b Specifies to return the ATIP data. The Track/Session Number field should be set to 0x00.

The data returned for format 0000b have the structure shown in Figure 8.19. The data block returned contains a header of four bytes followed by zero or more track descriptors.

Bit / Byte	7	6	5	4	3	2	1	0
0..1	TOC Data Length (Byte 0 - MSB)							
2	First Track Number (Hex)							
3	Last Track Number (Hex)							
Track Descriptor(s)								
0	Reserved							
1	ADR				CONTROL			
2	Track Number (Hex)							
3	Reserved							
4..7	Logical Block Address (Byte 4 - MSB)							

Figure 8.19. Data returned by the READ TOC/PMA/ATIP command for format 0000b.

TOC Data Length specifies the length in bytes of the following TOC data. This length does not include the TOC data length field itself. The First Track Number field contains the first track number in the table of contents of the first complete session. The Last Track Number field contains the last track number in the table of contents of the last complete session (before the *lead-out* area). Valid track numbers are running between 01 and 99 (0x63). The first track may have any valid number. The ADR field indicates the type of information encoded in the *Q* sub-channel of the block where this entry in the table of contents was found:

0x0: Information about *Q* sub-channel information encoding not supplied.

0x1: *Q* sub-channel encodes current position information (track, index, absolute address, relative address).

0x2: *Q* sub-channel encodes media catalog number.

0x3: *Q* sub-channel represents the *International Standard Recording Code* (ISRC).

0x4-0xF: Reserved values.

The CONTROL field indicates the attributes of the track (data or audio, uninterrupted or incremental recording, digital copy prohibited or permitted). The Track Number field indicates the track number for which the data in the track descriptor are valid. A track number of 0xAA indicates that the track descriptor refers to the *lead-out* area. The Logical Block Address contains the number of the first block with information for that particular track. If the MSF bit in the command packet is 0, the Logical Block Address field contains a logical address, and if the MSF bit is 1 this field contains an MSF address. The M, S, and F fields of this address are located in bytes 1, 2, and 3 of the absolute address, respectively (byte 0 is reserved).

The data returned for format 0001b have the structure shown in Figure 8.20.

Bit Byte	7	6	5	4	3	2	1	0
0..1	TOC Data Length (Byte 0 - MSB)							
2	First Complete Session Number (Hex)							
3	Last Complete Session Number (Hex)							
Track Descriptor								
0	Reserved							
1	ADR				CONTROL			
2	First Track Number in Last Complete Session (Hex)							
3	Reserved							
4..7	Logical Block Address of First Track in Last Session (Byte 4 - MSB)							

Figure 8.20. Data returned by the READ TOC/PMA/ATIP command for format 0001b.

The first complete session number is set to 1. For single-session discs, or if the drive does not support multi-session discs, the last complete session number is set to 1.

For format 0010b, the data returned have the structure shown in Figure 8.21.

Bit Byte	7	6	5	4	3	2	1	0
0..1	TOC Data Length (Byte 0 - MSB)							
2	First Complete Session Number (Hex)							
3	Last Complete Session Number (Hex)							
Track Descriptor(s)								
0	Session Number (Hex)							
1	ADR				CONTROL			
2	TNO							
3	POINT							
4	MIN							
5	SEC							
6	FRAME							
7	ZERO							
8	PMIN							
9	PSEC							
10	PFRAME							

Figure 8.21. Data returned by the READ TOC/PMA/ATIP command for format 0010b.

The meaning of bytes 2..10 is the same as that described in Section 8.2.5 for the DATA-Q field of sub-channel *Q*. The data returned for multi-session discs are arranged in increasing order of the session number. Within a session, the data are arranged in the following order of the POINT field value: 0xA0, 0xA1, 0xA2, track numbers, 0xB0, 0xB1, 0xB2, 0xB3, 0xB4, 0xC0, and 0xC1.

8.7.9. START/STOP UNIT Command

The START/STOP UNIT command allows to enable or disable access to the media by the CD/DVD drive and to place the drive in a low-power mode. The protocol used is the ATAPI protocol for non-data commands. The structure of the command packet is shown in Figure 8.22.

Bit Byte	7	6	5	4	3	2	1	0
0	Operation Code (0x1B)							
1	Reserved							IMED
2..3	Reserved							
4	Mode				Reserved		LOEJ	START
5	Control Code							
6..11	Reserved							

Figure 8.22. Structure of the START/STOP UNIT command packet.

If the IMED bit is 1, the status will be returned immediately after the command packet has been sent and validated. If the IMED bit is 0, the status will be returned after completing the operation. The Mode field allows selecting an operating mode with low-power consumption. If this field contains a value different than 0x0, the LOEJ and START will be ignored. Table 8.7 indicates some usual values for the Mode field.

Table 8.7. Usual values of the Mode field in the START/STOP UNIT command packet.

Mode	Meaning
0x0	The operating mode remains unchanged
0x1	Select the active mode
0x2	Select the inactive mode
0x3	Select the "*Standby*" mode
0x5	Select the "*Sleep*" mode

If the START bit is 1, the use of the drive is enabled, and if the START bit is 0, the use of the drive is disabled (the media cannot be accessed by the host computer).

If the LOEJ (*Load / Eject*) bit is 0, no actions will be performed to load or unload the media. If the LOEJ bit is 1, the command requests media unload if the START bit is 0, or media load if the START bit is 1.

8.7.10. STOP PLAY/SCAN Command

The STOP PLAY/SCAN command stops execution of CD audio commands. The protocol used is the ATAPI protocol for non-data commands. The structure of the command packet is shown in Figure 8.23.

Bit Byte	7	6	5	4	3	2	1	0
0	Operation Code (0x4E)							
1..8	Reserved							
9	Control Code							
10..11	Reserved							

Figure 8.23. Structure of the STOP PLAY/SCAN command packet.

8.8. Applications

8.8.1. Answer the following questions:

a. What are the steps required for recording data on compact discs?

b. How the error correction within the frames is performed?

c. How data and audio/video information can be read simultaneously from CD-ROM/XA and CD-I discs?

d. What do the sub-channels of compact discs mean and what are they used for?

8.8.2. Open the project created for the laboratory work *ATA Interface* and extend it by writing a function that sends the IDENTIFY PACKET DEVICE command to an ATAPI 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 8.7.2; the protocol used for this command is the same as that used for the IDENTIFY DEVICE command of the ATA interface. If the command completes successfully, the function displays the following information about the ATAPI drive: model number, serial number, firmware revision, and length of the ATAPI command packets.

After writing the function, include a call to this function in the `AppScroll()` function, using as parameters the base address of the Command Block registers for the primary ATA channel, and drive number 1. Repeat the function call for the secondary ATA channel with drive number 1.

8.8.3. Open the project created for the laboratory work *ATA Interface* and extend it by writing a function that sends the START/STOP UNIT command to an ATAPI drive in order to unload the media from the 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. The command is described in Section 8.7.9. First, define the command packet as an array of 12 bytes and initialize each byte to 0 using the `memset()` function. Next, initialize byte 0 of the command packet with the code of the SCSI START/STOP UNIT command. In byte 4 of the command packet, set the LOEJ bit to 1, and the other bits to 0 (Figure 8.22). Then, implement the ATAPI protocol for non-data commands, as described in Section 8.6.1.

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 drive number 1. Repeat the function call for the secondary ATA channel with drive number 1.

8.8.4. Continue Application 8.8.2 by writing a function that sends the READ CD RECORDED CAPACITY command to an ATAPI 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. The command is described in Section 8.7.7. First, define the command packet as an array of 12 bytes, and the data returned by the command as an array of eight bytes. Initialize each byte of the command packet to 0 using the `memset()` function. Next, initialize byte 0 of the command packet with the code of the READ CD RECORDED CAPACITY command. Then, implement the ATAPI protocol for input in PIO mode, as described in Section 8.6.2. If the command completes successfully, define variables of type `DWORD` for the last logical block address and for the block length. Initialize these variables with the corresponding fields of the data returned by the command, considering the correct order of the bytes (the last logical block address should be incremented to get the number of logical blocks). Finally, display the address of the last logical block, the block length, and the recorded capacity of the disc in MB.

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 drive number 1. Repeat the function call for the secondary ATA channel with drive number 1. Insert a data CD or DVD into the drive and verify the operation of the function by comparing the recorded capacity displayed with the capacity shown by the operating system.

8.8.5. Continue Application 8.8.4 by writing a function that sends the READ TOC/PMA/ATIP command to an ATAPI drive in order to read the table of contents of an audio CD. The input parameters of the function are the same as the parameters of the function written for Application 8.8.4. The function does not return any value. The command is described in Section 8.7.8. First, define the command packet as an array of 12 bytes, and the data returned by the command as an array of 256 bytes. Initialize each byte of the command packet to 0 using

the `memset()` function. Next, initialize byte 0 of the command packet with the code of the SCSI READ TOC/PMA/ATIP command. In byte 1 of the command packet, set the MSF bit to 1 and the other bits to 0 (Figure 8.18). Specify the format 0000b for the data that should be returned by the drive by leaving byte 2 of the command packet unchanged (the bytes of the command packet have been initialized to 0). Initialize the field for the allocation length of the command packet with the size of the buffer allocated for the data returned by the command (byte 7 should be initialized with the high-order byte of the size, and byte 8 with the low-order byte of the size). Next, implement the ATAPI protocol for input in PIO mode, as described in Section 8.6.2. If the command completes successfully, display the number of the first track and the number of the last track of the audio CD (Figure 8.19). Finally, for each audio track, display the beginning minute, second, and frame of the track (the M, S, and F fields of the logical block address).

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 drive number 1. Repeat the function call for the secondary ATA channel with drive number 1. Insert an audio CD into the drive and verify whether the information displayed is correct based on the track lengths printed on the disc label.

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] InterNational Committee for Information Technology Standards, “Information Technology – MultiMedia Command Set – 6 (MMC-6)”, Project T10/BSR INCITS 468, Revision 02g, December 11, 2009.

[7] InterNational Committee for Information Technology Standards, “Information Technology – SCSI Block Commands – 4 (SBC-4)”, Project T10/BSR INCITS 506, Revision 22, September 15, 2020.

[8] InterNational Committee for Information Technology Standards, “Information Technology – SCSI Primary Commands – 6 (SPC-6)”, Project T10/BSR INCITS 566, Revision 5, March 8, 2021.

[9] Rosch, W. L., *Hardware Bible*, Sixth Edition, Que Publishing, 2003.