University of California at Berkeley

College of Engineering

Department of Electrical Engineering and Computer Science

EECS 150 Mike Lowey

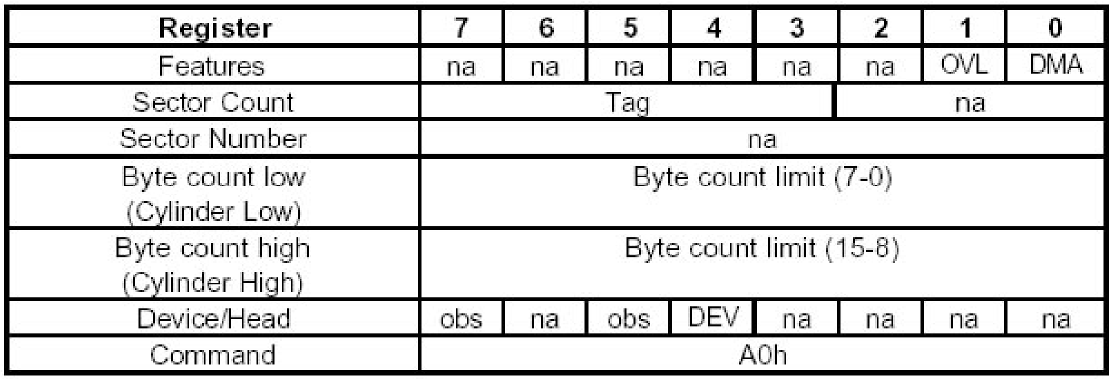
Fall 2001

Project Checkpoint 2
CD-ROM Drive Control

<u>ATA/ATAPI CD-ROM Interface Specifications</u>:

In this stage of the project we will be implementing a small subset of the ATAPI CD-ROM interface commands. By using just 3 of the available CD-ROM commands we will be able to duplicate most of the controls on a normal CD player.

Most hard-disk drives and CD-ROM drives are controlled by a standard known as ATA or the newer version ATAPI. ATAPI added a feature known as *command packets* to the older ATA standard (the –PI in ATAPI stands for Packet Interface). In the ATA interface, commands are sent to the drive using a system of *command blocks*. These blocks consist of 7 bytes of data being written to the drive in a particular order. Using this system the host device will write set-up parameters in to 6 registers and then write the command to be executed in to the command register. The command block structure, register addresses, and functions are in the pictures below.



Register	7	6	5	4	3	2	1	0
Features	na	na	na	na	na	na	OVL	DMA
Sector Count	Tag					na		
Sector Number	na							
Byte count low (Cylinder Low)	Byte count limit (7-0)							
Byte count high (Cylinder High)	Byte count limit (15-8)							
Device/Head	obs	na	obs	DEV	na	na	na	na
Command	A0h							

Table F.4 – Register functions and selection addresses except PACKET and SERVICE commands

Addresses					Functions	
CS0-	**CS1-**	**DA2**	**DA1**	**DA0**	**Read (DIOR-)**	**Write (DIOW-)**
N	N	x	x	x	Released	Not used
					Control block registers	
N	A	N	x	x	Released	Not used
N	A	A	N	x	Released	Not used
N	A	A	A	N	Alternate Status	Device Control
N	A	A	A	A	Obsolete(see note)	Not used
					Command block registers	
A	N	N	N	N	Data	Data
A	N	N	N	A	Error	Features
A	N	N	A	N	Sector Count	Sector Count
A	N	N	A	A	Sector Number	Sector Number
A	N	A	N	N	Cylinder Low	Cylinder Low
A	N	A	N	A	Cylinder High	Cylinder High
A	N	A	A	N	Device/Head	Device/Head
A	N	A	A	A	Status	Command
A	A	x	x	x	Released	Not used

Key:
A = signal asserted N = signal negated x = don't care
NOTE – This register is obsolete. It is recommended that a device not respond to a read of this address.

The registers are addressed using the signals CS0-, CS1-, DA2, DA1, and DA0. These signals are each specific wires on the 40-pin bus. CS0- and CS1- are active low, so in the table above they are zero when asserted. The complete list of the pins is below:

Table A.3 – 40-pin I/O connector interface signals

Signal name	Connector contact	Conductor		Connector contact	Signal name
RESET-	1	1	2	2	Ground
DD7	3	3	4	4	DD8
DD6	5	5	6	6	DD9
DD5	7	7	8	8	DD10
DD4	9	9	10	10	DD11
DD3	11	11	12	12	DD12
DD2	13	13	14	14	DD13
DD1	15	15	16	16	DD14
DD0	17	17	18	18	DD15
Ground	19	19	20	20	(keypin)
DMARQ	21	21	22	22	Ground
DIOW-:STOP	23	23	24	24	Ground
DIOR-:HDMARDY-:HSTROBE	25	25	26	26	Ground
IORDY:DDMARDY-:DSTROBE	27	27	28	28	CSEL
DMACK-	29	29	30	30	Ground
INTRQ	31	31	32	32	Obsolete (see note)
DA1	33	33	34	34	PDIAG-:CBLID-
DA0	35	35	36	36	DA2
CS0-	37	37	38	38	CS1-
DASP-	39	39	40	40	Ground

NOTE – Pin 32 was defined as IOCS16 in ATA-2, ANSI X3.279-1996.

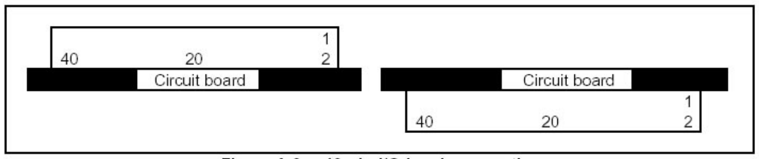
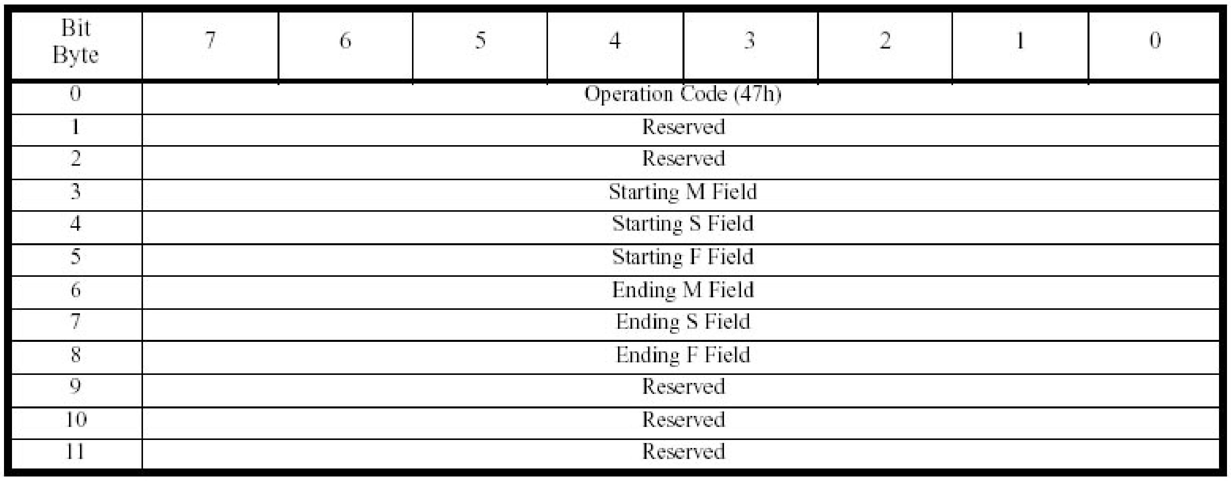


Figure A.3 – 40-pin I/O header mounting

In ATA data and command blocks are written to the registers using the lower 8 of the 16 data wires. The signals that control reading and writing to and from the registers are called DIOW- (write) and DIOR- (read). Both of these signals are active low as well. To write data to a register you should assert the address (CS0-, CS1-, DA0, DA1, DA2) , assert the data, and pull DIOR- low at the same time. Remember that some of the signals are active low. The only command block that we will use is the called PACKET. This command was added to the ATA standard to allow for the addition of a new set of commands. The structure for the PACKET command block is pictured above. Its command value is A0h. I have noticed that A0h will execute without having to set up any registers beforehand. You can just write the single command.

When the PACKET command block is written the drive knows that the next command will be an ATAPI *command packet.* The structure of command packets is different than command blocks.

Table 76 - PLAY AUDIO MSF Command



Bit Byte	7	6	5	4	3	2	1	0
0	Operation Code (47h)							
1	Reserved							
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	Reserved							
10	Reserved							
11	Reserved							

Command packets are 12 bytes long and contain the command code in the first set of data written. In ATAPI reading and writing happen using all 16 data bits. The difference can be seen on the ATAPI register map below. This means that the bytes are written two at a time to the device. The lower numbered byte will be in the low order 8 bits and the higher numbered byte will be in the upper 8 bits. In ATAPI CS0- and CS1- are called CS1FX and CS3FX. The register map is read exactly the same way. Reading and writing data is also handled the same way as ATA except that the data now flows 16 bits at a time. Submitting the packet above would take 6 writes to the *ATA command* register (see the register map below). I think that they try to make the names as confusing as possible…

6.7 ATAPI Register Map (Packet Command)

Logic conventions are: A = signal asserted, N = signal negated, x = does not matter which it is

Table 10 - I/O Port Functions/Selection Addresses (Compatibility Model)

Addresses					Functions	
CS1FX	CS3FX	DA2	DA1	DA0	Read (DIOR-)	Write (DIOW-)
					Control Block Registers	
N	A	0	0	0	Floppy A Status	Unused
N	A	0	0	1	Floppy B Status	Unused
N	A	0	1	0	Unused	Floppy Digital Output Register
N	A	0	1	1	Floppy ID / Tape Control	RESERVED
N	A	1	0	0	Floppy Controller Status	RESERVED
N	A	1	0	1	Floppy Data Register	
N	A	1	1	0	Alternate ATAPI Status	Device Control
N	A	1	1	1	*Note*[1]	Not Used
					Command Block Registers	
A	N	0	0	0	Data	
A	N	0	0	1	ATAPI Error Register	ATAPI Features
A	N	0	1	0	ATAPI Interrupt Reason Register	Unused
A	N	0	1	1	Reserved For SAM TAG Byte	
A	N	1	0	0	ATAPI Byte Count Register (bits 0-7)	
A	N	1	0	1	ATAPI Byte Count Register (bits 8-15)	
A	N	1	1	0	Drive Select	
A	N	1	1	1	ATAPI Status	ATA Command

1. This register is obsolete. It is recommended that a device not respond to a read of this address. If a device does respond, it ***shall*** not drive the DDF signal.

To execute an ATAPI CD-ROM command packet we first write the ATA PACKET command block (A0h), wait, and then write the ATAPI command packet. The timing of these operations is very important. You must wait around 5µsec between writing A0h and beginning the CD-ROM command packet. The duration of each write or read is also very important. Writing out at our maximum clock speed of 16MHz gives us 62.5nsec / write. Anything slower than this won't work so be careful to write to the bus as fast as possible. As the sequence of commands is written , you must leave a space between each write for the command to be interpreted correctly. I have been using 080000h (DIOR-, DIOW-, CS0-, CS1-, DA0, DA1, DA2, DD[15:0]) and it makes a good no-op. To wait for the 5µsec I write about 75 no-ops to the bus and then start the packet. I am sure that there are better ways to do it, but that is a start. This is a good time to start to consider the amount of redundant data in your design and begin to think about how to compress it.

The ATAPI packet example above is the PLAY AUDIO MSF command. This is one of the commands that we will be using for the project. Its structure is very intuitive. The MSF format is a refreshingly simple way to address locations on the disk:

M = minutes

S = seconds

F = frame

NOTE: The term "frame" is used in two different ways in the CD-ROM media standard. The intended meaning can only be determined from the context. Whenever possible, this description replaces the larger data unit with the more familiar term sector. The primary exception to this policy is the use of frame when referring to the MSF address. In the MSF context, one frame (F field unit) equals one sector. On a typical two channel CD-DA media, each frame (F field unit) is played in 1/75th of a second.

To play the section of music on the disk from 15-minutes to 20-minutes time you would write the following series of data words to the command register:

0047	-	first 2 bytes – reserved and play op-code
1500	-	starting M field and reserved
0000	-	starting F field and starting S field
0020	-	ending S field and ending M field
0000	-	reserved and ending F field

Reserved fields should get zeros.

For this checkpoint we are going to give you an I/O block that has all of the pins set up correctly for you. It is a good idea to have all of your pins instantiated in one macro to keep track of them. It also makes it easier to find pins to use for debugging.

This is a list of the pins that you will need to wire for the 40-pin connector:

Xilinx Pin#	cd pin#	signal name
3	1	reset
4	3	DD7
5	4	DD8
6	5	DD6
7	6	DD9
8	7	DD5
9	8	DD10
10	9	DD4
14	10	DD11
18	11	DD3

29	12	DD12
51	13	DD2
61	14	DD13
62	15	DD1
68	16	DD14
69	17	DD0
56	18	DD15
77	21	DMARQ
50	23	DIOW-
35	25	DIOR-
36	27	IORDY
37	28	CSEL
38	29	DMACK-
39	31	INTRQ
40	32	RESERVED
44	34	PDIAG-
45	35	DA0
46	36	DA2
47	37	CS0-
48	38	CS1-
49	39	DASP-
40	33	DA1

Please wire the 40-pin connector as close to the end of the board with the power connectors as practical. You will need the space in the middle for more chips and it would be a problem to wire wrap around the IDE cable. You will be provided with a Chekpoint2.bit file that will test your wiring.

Checkpoint Overview:

There are four main tasks for CP#2. The first will be to wire wrap the IDE cable connector and test it with the .bit file provided. The second will be to implement a series of commands that will give us CD player functionality. The third will be to implement an approximation of a working table of contents (TOC) for your player to use. The fourth will be to replace your random time function from CP#1 with something that actually counts time.

IMPLEMENTING COMMMANDS:

You will be implementing the following functions in the CD player:

PLAY	- Plays CD from beginning to end
NEXT TRACK	- Plays the next track
PREVIOUS TRACK	- Plays the previous track
STOP	- Terminates playing and resets current track to zero
PAUSE / RESUME	- Pauses or resumes playback at current time
EJECT	- Ejects disk from player

All of these functions can be implemented by using one of three built in commands. The commands that we will be using are:

- PLAY AUDIO MSF (47h)
- START / STOP UNIT (1Bh)
- PAUSE / RESUME (4Bh)

PLAY AUDIO MSF can be used to implement PLAY, NEXT TRACK, and PREVIOUS TRACK. This is because it allows us to easily specify in minutes and seconds the place to begin playback. We will be creating a primitive TOC that will contain the start times of the tracks, so jumping from track to track should be easy.

START / STOP UNIT can be used to stop playback, eject the disk, or load a disk. We will be using it to do the first two.

PAUSE / RESUME will do exactly what it says.

TABLE OF CONTENTS:

In this section you will create a TOC for a CD. This is a bit of a kludge, but it makes other functions possible until someone figures out how to use the READ TOC (43h) command (extra credit?). Groups should pick a CD and manually create an array of its track starting times in minutes and seconds. You will also need to keep a record of you current track number and update it appropriately in response to the various commands.

TIME:

In this section you will make the digits that appear on the LCD represent actual time. They should display the elapsed time since the beginning of the current track. This is a simple case. It should reset to zero and begin counting (in minutes and seconds) each time a new track if played. It should stop counting and resume in the same place when playback is paused and resumed. It should reset to zero when playback is stopped. The elapsed time does not need to reset if the player plays through one track and begins another. The output should refresh at least once a second. It doesn't need to be 100% exact either. Incrementing every 0.953674 seconds would be fine (16M/2^{24}).

10.8.7 PAUSE/RESUME Command

The PAUSE/RESUME command requests that the device stop or start an audio play operation. This command is used with PLAY AUDIO commands that are currently executing.

Table 70 - PAUSE/RESUME Command

Bit / Byte	7	6	5	4	3	2	1	0
0	Operation Code (4Bh)							
1	Reserved							
2	Reserved							
3	Reserved							
4	Reserved							
5	Reserved							
6	Reserved							
7	Reserved							
8	Reserved							Resume
9	Reserved							
10	Reserved							
11	Reserved							

A Resume bit of zero causes the drive to enter the hold track state with the audio output muted after the current block is played. A Resume bit of one causes the drive to release the pause/scan and begin play at the block following the last block played/scanned.

If an audio play operation cannot be resumed and the resume bit is one, the command is terminated with CHECK CONDITION status. If the resume bit is zero and an audio play operation cannot be paused, (no audio play operation has been requested, or the requested audio play operation has been completed), the command is terminated with CHECK CONDITION status. See *"Figure 15 - Stop Play/Play Audio/Audio Scan/Pause/Resume Sequencing"* on page 196 for additional information.

It ***shall not*** be considered an error to request a PAUSE when a pause is already in effect or to request a RESUME when a play operation is in progress.

Table 71 - Recommended Sense Key, ASC and ASCQ for Pause/Resume Command Errors

Sense Key	ASC	ASCQ	Description of Error
05	20		INVALID COMMAND OPERATION CODE
05	24		INVALID FIELD IN COMMAND PACKET
06	28		NOT READY TO READY TRANSITION
06	29		POWER ON, RESET OR BUS DEVICE RESET OCCURRED
02	04	00	LOGICAL DRIVE NOT READY - CAUSE NOT REPORTABLE
02	04	01	LOGICAL DRIVE NOT READY - IN PROGRESS OF BECOMING READY
02	04	02	LOGICAL DRIVE NOT READY - INITIALIZING COMMAND REQUIRED
02	04	03	LOGICAL DRIVE NOT READY - MANUAL INTERVENTION REQUIRED
02	3A		MEDIUM NOT PRESENT
0B	B9		PLAY OPERATION ABORTED

10.8.9 PLAY AUDIO MSF Command

The PLAY AUDIO MSF command requests that the ATAPI CD-ROM Drive begin an audio playback operation. The command function and the output of audio signals ***shall*** be as specified by the settings of the mode parameters including the SOTC bit described on page 111.

Table 76 - PLAY AUDIO MSF Command

Bit / Byte	7	6	5	4	3	2	1	0
0	Operation Code (47h)							
1	Reserved							
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	Reserved							
10	Reserved							
11	Reserved							

This command responds with immediate status, allowing overlapped commands. This command ***shall*** set the DSC bit upon command completion. See also *"10.4 Immediate Command Processing Considerations"* on page 90.

The Starting M field, the Starting S field, and the Starting F field specify the absolute MSF address at which the audio play operation ***shall*** begin. The Ending M field, the Ending S field, and the Ending F field specify the absolute MSF address where the audio play operation ***shall*** end. All contiguous audio sectors between the starting and the ending MSF address ***shall*** be played.

If the Starting Minutes, Seconds and Frame Fields are set to FFh, the Starting address is taken from the Current Optical Head location. This allows the Audio Ending address to be changed without interrupting the current playback operation.

A Starting MSF address equal to an ending MSF address causes no audio play operation to occur. This ***shall not*** be considered an error. If the Starting MSF address is greater than the Ending MSF address, the command ***shall*** be terminated with CHECK CONDITION status. The sense key ***shall*** be set to ILLEGAL REQUEST.

If the starting address is not found, if the address is not within an audio track, or if a not ready condition exists, the command ***shall*** be terminated with CHECK CONDITION status.

See *"10.8.8.1 Play Audio with Immediate Packet Commands"* on page 126 for information on overlapped commands during an Audio Playback.

10.8.25 START/STOP UNIT Command

The START/STOP UNIT command requests that the ATAPI CD-ROM Drive enable or disable media access operations.

Table 154 - START/STOP UNIT Command

Bit Byte	7	6	5	4	3	2	1	0
0	Operation code (1Bh)							
1	Reserved							Immed
2	Reserved							
3	Reserved							
4	Reserved						LoEj	Start
5								
6	Reserved							
7	Reserved							
8	Reserved							
9	Reserved							
10	Reserved							
11	Reserved							

An immediate (Immed) bit of one indicates that status ***shall*** be returned as soon as the Command Packet has been validated. An Immed bit of zero indicates that status ***shall*** be returned after the operation is completed.

A start bit of one requests the Device be made ready for use. A start bit of zero requests that the Device be stopped (media cannot be accessed by the Host Computer).

Table 155 - Start/Stop and Eject Operations

LoEj	Start	Operation to be Performed
0	0	Stop the Disc
0	1	Start the Disc and read the TOC
1	0	Eject the Disc if possible (See *"Table 84 - Actions for Lock / Unlock / Eject"* on page 136)
1	1	Load the Disc (Close Tray)

Any attempt to Eject or Load a Disc when the Drive does not support that capability ***shall*** result in an error condition being reported to the Host (Sense key 05 ILLEGAL REQUEST, Sense Code 24 INVALID FIELD IN COMMAND PACKET.)

A load eject (LoEj) bit of zero requests that no action be taken regarding loading or ejecting the medium. A LoEj bit of one requests that the medium be unloaded if the start bit is zero. A LoEj bit of one requests that the medium be loaded if the start bit is one.

When the Loading Mechanism Type is a Changer utilizing individual disc change capability (4h), the Eject operation ***shall*** only eject the disc that is currently in the Play Position. If the Loading Mechanism is a changer utilizing a Cartridge (5h), then the Cartridge ***shall*** only be ejected when no media is in the play position.

Name________________________________ Name___________________________________

Project Checkpoint 2

Checkoff Sheet

Design

Design time counter before lab	___________________(5%)
FSMs designed before lab	___________________(5%)

Testing

Play works	___________________(10%)
Next Track works	___________________(10%)
Prev Track works	___________________(10%)
Pause works	___________________(10%)
Resume works	___________________(10%)
Stop works	___________________(10%)
Eject works	___________________(10%)
TOC works	___________________(10%)
Time works	___________________(10%)

Turned in on time (2 weeks)	___________________**(100%)**
Turned in one week late (3 weeks)	___________________**(50%)**
Turned in one week early (1 week)	___________________**(120%)**