

Seminar Einchipcomputer
Ansteuerung von Festplatten über IDE

Christopher Israel
`cisrael@uni-koblenz.de`

Juli 2007

Inhaltsverzeichnis

1	Einleitung	2
1.1	Allgemeines	2
1.2	Warum IDE?	2
2	Funktionsweise	3
2.1	Übertragungsmodi	3
2.2	Adressierung	4
2.3	Anschlussbelegung	5
2.4	Register	6
2.5	Kommandos	8
2.6	Beispiel: Sektor lesen	8
3	Probleme und Lösungen	9
3.1	Problem: IDE belegt zu viele Pins des Mikrocontrollers	9
3.2	Problem: IDE benötigt zuviel Speicher	10
4	Quellen	11

1 Einleitung

1.1 Allgemeines

Ein großer Teil der verfügbaren Festplatten wird über die IDE-Schnittstelle angesteuert.

Die IDE-Schnittstelle wurde 1984 von den Firmen Imprimis und Western Digital im Auftrag von Compaq Computer entwickelt und 1989 erstmals als Standard ausgeschrieben. Durch diese Entwicklung wurden die damals vorherrschenden, größtenteils „dummen“ Platten ohne integrierte Logik abgelöst, die durch separate Controllersteckkarten an den Computer angeschlossen wurden.

Bei IDE-Laufwerken ist die Logik direkt in der Festplatte integriert, daher auch die Bezeichnung IDE - „Integrated Device Electronics“. Es gibt auch andere Bezeichnungen für dieselbe Schnittstelle, wie ATA, E-IDE („Enhanced IDE“), Fast-ATA oder Ultra-ATA.

ATA steht für „Advanced Technology Attachment“ und diese Bezeichnung kommt daher, dass die IDE-Schnittstelle eine vereinfachte Form des ISA-Busses des IBM PC/AT darstellt. E-IDE, Fast-ATA und Ultra-ATA sind dahingegen Marketing-Begriffe, die von den Festplattenherstellern eingeführt wurden, um ihre Produkte von denen der anderen Festplattenherstellern abzugrenzen. Der ATA-Standard wird vom T13-Komitee gepflegt und weiterentwickelt und hat mit ATA-8 bereits acht Versionen durchlaufen:

Version	Erscheinungsjahr	Features
ATA-1	1989	erster Standard
ATA-2	1994	28-Bit LBA, höhere Geschwindigkeiten
ATA-3	1996	SMART, Sicherheitsfeatures
ATA-4	1997	ATAPI, Ultra-DMA
ATA-5	1999	80-adriges Kabel
ATA-6	2000	48-Bit LBA
ATA-7	2001	SATA
ATA-8	2005	

1.2 Warum IDE?

Der große Vorteil von Festplatten gegenüber Speicherkarten, EEPROM oder dem integrierten Speicher des Mikrocontrollers ist die Datenmenge, die auf einer Festplatte gespeichert werden kann.

Mögliche Verwendungen für die Anbindung einer Festplatte an einen Mikrocontroller sind z. B.:

- MP3-Player
- AVR-Programmer
- Diktiergerät
- „Image-Tank“ (Gerät, das den Inhalt von Speicherkarten auf eine Festplatte kopiert)

Ein weiterer Vorteil der IDE-Schnittstelle ist, dass über sie auch CompactFlash-Karten angesteuert werden können. Diese haben nämlich einen Kompatibilitätsmodus, in dem sie sich wie eine IDE-Festplatte verhalten.

2 Funktionsweise

Eine IDE-Festplatte wird über eine Reihe von Registern angesprochen. Man wählt das Register, das beschrieben oder ausgelesen werden soll, über 5 Adresspins aus und gibt dann einen Impuls auf die Lese- oder Schreibleitung der Festplatte. Während dieses Impulses liegt der Registerinhalt an den Datenpins an (wenn die Leseleitung verwendet wurde), beziehungsweise das Register übernimmt den Wert, der während des Impulses an den Datenpins anliegt (wenn die Schreibleitung verwendet wurde).

2.1 Übertragungsmodi

Es gibt verschiedene Modi zur Übertragung der Daten zwischen Host und Festplatte:

- PIO (Programmed IO)
- DMA (Direct Memory Access)
- Ultra DMA

Bei PIO (Programmed IO) muss der Host den Transfer jedes einzelnen Datenwortes separat initiieren. Dadurch wird allerdings die CPU bei jedem Festplattenzugriff stark ausgebremst, da sie auf die langsamere Festplatte warten muss und somit blockiert ist. Um diese Problematik zu umgehen, wurden die DMA-Modi eingeführt, bei denen die Daten von einem DMA-Controller ohne Zutun der CPU zwischen Festplatte und Arbeitsspeicher übertragen werden können. Es gibt Single-Word DMA, wobei jeweils ein einzelnes Datenwort übertragen wird, und Multi-Word DMA, bei dem mehrere Datenworte hintereinander gesendet werden können ohne erneut Instruktionen austauschen zu müssen. Multi-Word DMA ist daher bei größeren Datenübertragungen aufgrund des geringeren Overheads schneller als PIO oder Single-Word DMA.

Ultra DMA ist eine schnellere Variante von DMA, die allerdings einige Einschränkungen hat. So müssen bei Ultra DMA-Transfers 80-adrige Kabel verwendet werden und es werden CRC16-Prüfsummen ausgetauscht. Diese Einschränkungen sind nötig, da bei den hohen Transferraten von Ultra DMA Übersprechungen und Übertragungsfehler auftreten können.

Für die Ansteuerung einer Festplatte durch einen Mikrocontroller ist allerdings nur PIO interessant, denn:

- die Maximalgeschwindigkeit von PIO ist mit 16 MB/s für AVRs nicht erreichbar
- bei gleicher Maximalgeschwindigkeit ist DMA nur komplizierter und belegt zusätzliche Pins am μC
- U-DMA ist noch komplizierter als DMA, da zusätzlich CRC-Prüfsummen gebildet werden müssen

2.2 Adressierung

Bei IDE-Festplatten sind zwei Arten der Adressierung möglich:

- CHS (Cylinder Head Sector) und
- LBA (Logical Block Addressing)

Sowohl bei CHS als auch bei LBA werden Blöcke adressiert, die sich aus 256 16-Bit Datenwörtern, also insgesamt 512 Byte, zusammensetzen.

CHS-Adressen spiegeln die tatsächliche Aufteilung der Festplatte in Zylinder, Leseköpfe und Sektoren wieder. Sie sind jedoch veraltet, da der maximal adressierbare Speicher durch die kurzen CHS-Adressen (insgesamt 24 Bit) auf 8 GB beschränkt ist. Zudem haben bei neueren Festplatten die äußeren Spuren mehr Sektoren als die inneren Spuren, während bei der CHS-Adressierung davon ausgegangen wird, dass alle Spuren dieselbe Anzahl von Sektoren haben.

Bei LBA ist dieses Problem nicht mehr vorhanden, da es nur noch eine Adresse gibt. Die Festplatte kümmert sich dabei um die Umrechnung zwischen logischer Adresse und physikalischer Position. Auch bei LBA gibt es allerdings eine Grenze des maximal adressierbaren Speichers, die durch die Adresslänge bestimmt wird: Bei der älteren Variante von LBA, die im Jahr 1994 mit der 2. Version der ATA-Spezifikation eingeführt wurde, ist die Adresslänge auf 28 Bit beschränkt. Damit können 268.435.456 Blöcke, also 128 GB adressiert werden.

Seit dem Jahr 2000 ist es mit der 6. Version der ATA-Spezifikation allerdings auch möglich, 48 Bit lange LBA-Adressen zu verwenden, wodurch 281.474.976.710.656 Blöcke adressiert werden können, was 128 Petabytes* entspricht.

* 1 Petabyte entspricht 1024 Terabytes

2.3 Anschlussbelegung

/RESET	1	2	GND
DB7	3	4	DB8
DB6	5	6	DB9
DB5	7	8	DB10
DB4	9	10	DB11
DB3	11	12	DB12
DB2	13	14	DB13
DB1	15	16	DB14
DB0	17	18	DB15
GND	19	20	KEYPIN
DMAREQ	21	22	GND
/DIOW	23	24	GND
/DIOR	25	26	GND
IORDY	27	28	CSEL
/DMAACK	29	30	GND
INTRQ	31	32	/IOCS16
ADDR1	33	34	/PDIAG
ADDR0	35	36	ADDR2
/CS0	37	38	/CS1
/DASP	39	40	GND

- Die Adressleitungen /CS0, /CS1 und ADDR0-2 dienen zur Auswahl des Registers
- Die Leitungen DB0-15 stellen den 16-Bit Datenbus dar
- /DIOW und /DIOR sind die Leitungen, durch die ein Schreib-/Lesezugriff auf ein Register eingeleitet wird
- INTRQ ist eine optionale Interruptleitung
- Mit IORDY informiert die Festplatte den Host darüber, dass Daten gelesen werden können
- Mit /IOCS16 informiert die Festplatte den Host darüber, dass ein 16-Bit Transfer ansteht
- DMAREQ und /DMAACK werden bei DMA-Zugriffen verwendet
- An /DASP kann eine LED zur Aktivitätsanzeige angeschlossen werden
- /PDIAG („Passed Diagnostics“) dient zur Kommunikation von zwei Festplatten miteinander und soll nicht mit dem Host verbunden werden
- CSEL wird verwendet um Cable Select zu aktivieren/deaktivieren
- RESET verursacht einen Hardware-Reset der Festplatte*

*Es ist auch möglich, mit einem Kommando einen Software-Reset durchzuführen. Dieser unterscheidet sich dadurch vom Hardware-Reset, dass nur die aktive Platte davon betroffen ist, während bei einem Hardware-Reset beide Festplatten einen Reset ausführen.

2.4 Register

Um im PIO-Modus auf eine Festplatte zuzugreifen, schreibt und liest man in den Registern der Festplatte.

Es gibt insgesamt 13 Register, man benötigt allerdings nur die folgenden:

- Command Register
- Sector Count Register
- Data Register
- Status Register
- Error Register
- LBA Low
- LBA Mid
- LBA High
- Device Register

Bis auf das Data Register sind alle Register 8 Bit breit. Das Data Register hat eine Breite von 16 Bit. Daher wird im Normalfall auch die /IOCS16-Leitung der Festplatte nicht benötigt, da allein durch die Auswahl des Registers bereits bekannt ist, ob ein 8-Bit oder 16-Bit Transfer ansteht.

Das Command Register ist die Schnittstelle, um Befehle an die Festplatte zu senden. Um eine Aktion durchzuführen, schreibt man den passenden Befehlscode für die Aktion in das Command Register und die Festplatte wird die Aktion ausführen.

Die Liste der Befehlscodes kann man in der ATA-Spezifikation nachlesen.

Beim Auslesen oder Beschreiben von Sektoren gibt das Sector Count Register die Anzahl der Sektoren an, die ausgelesen, bzw. beschrieben werden sollen.

Wenn bei der Ausführung eines Kommandos Daten gelesen oder geschrieben werden sollen, so geschieht dies über das Data Register.

Das Status Register enthält Informationen über den aktuellen Zustand der Festplatte, während das Error-Register im Falle eines Fehlers Informationen über den aufgetretenen Fehler beinhaltet.

Um beim Lesen oder Schreiben der Festplatte einen Sektor auszuwählen, schreibt man dessen Adresse in das LBA Low, LBA Mid und LBA High Register sowie in das Device Register. Dabei enthält das LBA Low Register das niedrigstwertige Byte der 28 Bit langen Adresse und das Device Register enthält das höchstwertige Nibble der Adresse.

Das Device Register wird außerdem dazu verwendet, die Festplatte auszuwählen, auf die zugegriffen wird*.

*Wenn zwei Festplatten angeschlossen sind, liest üblicherweise nur eine Festplatte am Bus mit. Wenn jedoch das Device Register beschrieben wird, lesen beide Festplatten gleichzeitig.

Status Register

Das Statusregister enthält folgende Flags:

Flag	Bedeutung
BSY („Busy“)	Gerät ist beschäftigt
DRDY („Device Ready“)	Gerät ist bereit, Kommandos auszuführen
DF („Device Fault“)	Ein Fehler ist aufgetreten, der nicht im Error Register beschrieben wird
DRQ („Data Request“)	Gerät ist bereit zur Datenübertragung
ERR („Error“)	Ein Fehler ist aufgetreten, der im Error Register beschrieben wird

Wichtig sind hierbei vor allen Dingen die Flags BSY, DRDY und DRQ.

BSY sollte immer geprüft werden, bevor ein Register beschrieben oder ausgelesen wird, da die Daten des Registers ungültig sind, während das BSY-Flag gesetzt ist. Dies trifft auch auf die anderen Bits des Status-Register zu.

Man sollte nur dann versuchen, Befehle in das Command Register zu schreiben, wenn das DRDY-Flag gesetzt ist und somit die Festplatte in der Lage ist, den Befehl auszuführen.

Das DRQ-Flag ist dann gesetzt, wenn die Festplatte bereit ist, Daten über das Data Register zu übertragen, beispielsweise wenn das Kommando zum Auslesen eines Sektors ausgeführt wurde.

Das DF-Flag gibt an, dass ein Fehler bei der Abarbeitung eines Kommandos aufgetreten ist, allerdings wird dieser Fehler nicht weiter beschrieben.

Das ERR-Flag dahingegen wird zwar ebenfalls gesetzt, wenn ein Fehler aufgetreten ist, zusätzlich gibt es auch an, dass das Error-Register einen gültigen Fehlercode enthält. Die Informationen, die das Error-Register enthält, hängen allerdings davon ab, bei welchem Kommando ein Fehler aufgetreten ist.

Register beschreiben/auslesen

Um ein Register zu beschreiben geht man folgendermaßen vor:

- Register über Adressleitungen auswählen
- zu schreibendes Datenwort an Datenbus anlegen
- /DIOW auf Low ziehen
- ca. 1 μs warten*
- /DIOW wieder auf High ziehen

*Die durch die ATA-Spezifikation vorgeschriebenen Timings sind tatsächlich wesentlich schneller, so ist beispielsweise die Zeit für einen kompletten Lese/Schreibzyklus mit 120ns angegeben (PIO Mode 4). Diese Timings sind jedoch nur die Mindestzeiten, das heißt dass einerseits die tatsächlichen Timings von Festplatte zu Festplatte variieren und andererseits der Datentransfer beliebig lange dauern darf. Um die tatsächlichen Timings einer Festplatte zu erfahren, muss man das IDENTIFY DEVICE Kommando ausführen.

Die Vorgehensweise um ein Register auszulesen ist ähnlich:

- Register über Adressleitungen auswählen
- /DIOR auf Low ziehen
- ca. 1 μs warten
- Datenwort vom Datenbus lesen
- /DIOR wieder auf High ziehen

Zusätzlich sollte vor jedem Lese- oder Schreibzugriff auf ein Register geprüft werden, ob das BSY-Bit im Status Register gesetzt ist, da sonst alle gelesenen oder geschriebenen Daten ungültig sind.

2.5 Kommandos

Es gibt eine große Anzahl von Kommandos, die von ATA-Laufwerken ausgeführt werden können. Um auf einer Festplatte lesen und schreiben zu können, reichen allerdings folgende Kommandos:

Kommando	Funktion
IDENTIFY DEVICE	Identifikationscode auslesen*
SET FEATURES	Einstellungen ändern†
READ SECTOR(S)	Sektor(en) lesen
WRITE SECTOR(S)	Sektor(en) schreiben

2.6 Beispiel: Sektor lesen

Um einen Sektor der Festplatte auszulesen, geht man wie folgt vor:

- LBA Low, Mid, High Register, sowie Device Register mit Adresse des Sektors füllen
- 1 in Sector Count Register schreiben‡
- READ SECTOR(S)-Kommando in Command Register schreiben
- warten, bis BSY-Flag im Status Register nicht mehr gesetzt und DRQ-Flag gesetzt ist
- 256 mal das Data Register auslesen
- überprüfen, ob ERR-Flag im Status Register gesetzt ist

Um einen Sektor der Festplatte zu beschreiben, geht man genauso vor, nur benutzt man das WRITE SECTOR(S)-Kommando anstelle des READ SECTOR(S)-Kommandos und man schreibt auf das Data Register anstatt es auszulesen.

*Die im 512 Byte langen Identifikationscode übermittelten Informationen enthalten unter Anderem die möglichen Übertragungsmodi, die Anzahl der Sektoren, die genauen Timings, und Informationen über die weiteren Fähigkeiten des Laufwerks

†Die Einstellungen, die geändert werden können, sind zum Beispiel, welcher Übertragungsmodus verwendet werden soll, ob der Cache-Speicher der Festplatte aktiviert sein soll, oder Einstellungen des Power-Managements

‡sorgt dafür, dass genau ein Sektor gelesen wird

3 Probleme und Lösungen

3.1 Problem: IDE belegt zu viele Pins des Mikrocontrollers

Die Minimalbelegung, die man benötigt, wenn man eine IDE-Festplatte ansteuern will, besteht aus insgesamt 24 Pins:

- 16 Datenleitungen
- 5 Adressleitungen (/CS0, /CS1 und ADDR0-2)
- Schreib- und Leseleitung (/DIOW und /DIOR)
- RESET-Pin

Da der AVR im 40-Pin DIP Package vier 8-Bit Ports haben, bleibt beim direkten Anschluss der IDE-Pins an den Mikrocontroller nur ein 8-Bit Port für andere Anwendungen frei.

Lösung: Pins doppelt verwenden

Die Datenleitungen der Festplatte sind hochohmig geschaltet, während kein Datentransfer in Gange ist.

Daher lassen sich die Datenleitungen doppelt belegen. Hierbei bieten sich vor allem Geräte an, die ebenfalls einen parallelen Datenbus benutzen.

Vorteile:

- zwei weitere 8-Bit Ports werden verfügbar

Nachteile:

- Datentransfer mit der Festplatte kann den an doppelt belegten Datenleitungen Probleme verursachen, wenn die anderen Geräte, die mit den Datenleitungen verbunden sind, nicht ebenfalls die entsprechenden Pins hochohmig schalten.

Lösung: Datenleitungen über Schieberegister ansteuern

Dadurch, dass die Datenübertragung beliebig lange dauern kann, ist es möglich die Datenleitungen nicht direkt, sondern über Schieberegister anzusteuern.

Vorteile:

- ungefähr zwei weitere 8-Bit Ports werden verfügbar

Nachteile:

- Das SPI des Mikrocontrollers wird belegt
- Die Datenübertragung nimmt mehr Zeit in Anspruch
- Schieberegister werden benötigt

Lösung: Separater IDE-Controller

Man könnte auch einen weiteren Mikrocontroller als IDE-Controller verwenden.

Vorteile:

- Es werden fast keine Pins am Haupt-Mikrocontroller belegt
- Das Speicherproblem wird auf diese Weise ebenfalls gelöst

Nachteile:

- Man benötigt einen weiteren Mikrocontroller
- Die Programmierung ist insgesamt komplizierter, da man ein Kommunikationsprotokoll zwischen beiden Mikrocontrollern ausarbeiten muss.

Lösung: CompactFlash-Karte verwenden

CompactFlash-Karten unterstützen im Gegensatz zu „richtigen“ Festplatten auch 8-Bit PIO Transfers. Dabei werden nur die unteren 8 Bit des 16-Bit Datenbusses verwendet. Um den 8-Bit Modus zu nutzen, muss man ihn mit dem SET FEATURES Kommando aktivieren.

Vorteile:

- Ein weiterer 8-Bit Port wird verfügbar

Nachteile:

- Man benötigt einen CF zu IDE Adapter

3.2 Problem: IDE benötigt zuviel Speicher

Es können beim Zugriff auf die Festplatte nur ganze Sektoren gelesen oder geschrieben werden. Wenn man also auf einzelne Bytes in einem Sektor zugreifen will, muss man den Sektor im Speicher des Mikrocontrollers zwischenspeichern. Da z. B. der ATmega 16 einen Speicher von 1024 Bytes hat und ein Sektor einer Festplatte aus 512 Bytes besteht, bleiben für andere Anwendungen nur noch 512 Bytes an Speicher übrig

Lösung: Externer Speicher

Man kann einen externen SRAM anschließen.

Vorteile:

- Durch den externen SRAM steht genügend Speicher zur Verfügung

Nachteile:

- Man benötigt externen SRAM
- Die Steuerleitungen des RAMs belegen weitere Pins des Mikrocontrollers

Lösung: Separater IDE-Controller

Man könnte auch hier einen weiteren Mikrocontroller als IDE-Controller verwenden.

Vorteile:

- Der zweite Mikrocontroller hat einen eigenen Speicher
- Das Pinproblem wird auf diese Weise ebenfalls gelöst

Nachteile:

- Man benötigt einen weiteren Mikrocontroller
- Die Programmierung ist insgesamt komplizierter

4 Quellen

- Wikipedia
- <http://www.circuitcellar.com/library/print/0103/Eady150/>
Fred Eady: „Construct an ATA Hard Drive Controller“
- <http://www.pcguides.com/ref/hdd/if/ide/index.htm>
The PC Guide: „Overview and History of the IDE/ATA Interface“
- <http://www.ata-atapi.com/>
Allgemeine Informationen über IDE/ATA
- <http://www.t10.org/t13>
Website des T13-Konsortiums, das den ATA-Standard entwickelt hat
- <http://www.gaby.de/gide/IDE-TCJ.txt>
Tilmann Reh: „Connecting IDE Drives“