Heinrich Heine
Universität Düsseldorf

UNIVERSITÄT DÜSSELDORF
Mathematisch-Naturwissenschaftliche Fakultät
Institut für Informatik

Prof. Dr. Michael Schöttner

Programmable Interrupt Controller

Ergänzung zum 3. Aufgabenblatt zum Modul „Betriebssystem-Entwicklung“

Der Programmierbare Interrupt Controller (PIC)

Der PIC 8259A ist ein Chip (oder Teil eines Chips), der wie der Prozessor selbst auf dem Motherboard jedes PCs zu finden ist. Seine Aufgabe besteht in der Koordination der Unterbrechungsanforderungen der verschiedenen Geräte.

Dazu besitzt der PIC 8 Eingänge IRQ0 bis IRQ7, an die jeweils der Interruptausgang eines Gerätes angeschlossen werden kann. Beispiele für diese Geräte sind der Timer, die Tastatur, die seriellen Schnittstellen (Maus) und der Festplattencontroller. Die Nummer des IRQ-Pins gibt gleichzeitig die Priorität der Unterbrechung an. Das Gerät, das an IRQ0 angeschlossen ist, wird mit höchster Priorität behandelt, das an IRQ7 mit der niedrigsten.

Der PIC hat mehrere Ausgänge, durch die er mit der CPU verbunden ist. Unter anderem ist der Pin INT des PICs direkt an den Pin INTR der CPU angeschlossen, ebenso sind die bei PIC und CPU gleichermaßen mit INTA bezeichneten Pins gekoppelt. Die Pins D0 bis D7 des PICs sind mit den acht niederwertigsten Leiterbahnen des Datenbusses verbunden.

Wenn der PIC feststellt, dass auf mindestens einem seiner Eingänge IRQ0 bis IRQ7 eine Interruptanforderung anliegt und dem PIC nicht mitgeteilt wurde, dass er die entsprechende Anforderung ignorieren soll, so gibt er die Interruptanforderung über seinen Ausgang INTR an die CPU weiter. Diese wird die Anforderung mit Hilfe der Leitung INTA quittieren und durch ein zweites Signal auf INTA nach der Nummer des Interrupts fragen. Der PIC antwortet daraufhin, indem er die Nummer des Interrupts (gleich der Nummer der Eingangsleitung + einem Offset) auf den Datenbus legt. Die CPU ist nun in der Lage, die Anfangsadresse der Unterbrechungsbehandlungsroutine zu finden und die Unterbrechungsbehandlung zu starten.

Damit der PIC sich merken kann, von welchen Geräten er Interrupt-Anforderungen erhalten hat, welche er ignorieren soll und welche gerade von der CPU behandelt werden, besitzt er drei Register, die jeweils 8 Bit breit sind:

Interrupt Request Register (IRR)

Hier speichert der PIC, auf welchen der IRQ-Leitungen Interruptanforderungen signalisiert wurden. Dadurch braucht das betreffende Gerät nur eine kurze Änderung des Pegels auf der Leitung auszulösen.

Interrupt Service Register (ISR)

Liegen die Interruptanforderungen mehrerer Geräte gleichzeitig an, so soll der PIC zunächst nur die wichtigste weiterleiten. Dazu wird die Unterbrechungsanforderung der CPU über die Leitung INT angezeigt. Sobald diese daraufhin mit zwei INTA Signalen reagiert hat, wird nicht nur die Nummer des Interrupts auf den Datenleitungen D0-D7 ausgegeben, sondern auch das entsprechende Bit im ISR gesetzt. Gleichzeitig wird das Bit im IRR gelöscht. Da das Bit im ISR gesetzt bleibt, bis es von der Unterbrechungsbehandlungsroutine mit Hilfe eines speziellen Befehls explizit gelöscht wird, kann der PIC beim Eintreffen einer weiteren Unterbrechung leicht erkennen, ob diese noch wichtiger als die gerade bearbeitete ist. In diesem Fall wird er dies der CPU durch ein erneutes Signal auf der INT-Leitung mitteilen.

Interrupt Mask Register (IMR)

Mit Hilfe dieses Registers ist es möglich, Interrupts selektiv zu unterdrücken. Ein gesetztes Bit im IMR sorgt dafür, dass der PIC Interrupts des entsprechenden Gerätes ignoriert.

Um mehr als 8 Interruptquellen unterscheiden zu können, enthalten moderne PCs (das heißt, alle ab dem XT) zwei PICs, die hintereinander geschaltet sind. Der INT-Ausgang des Slave-PICs hat also keine direkte Verbindung zu dem INTR-Eingang der CPU, sondern wird an einen der IRQ-Eingänge (IRQ2) des Master-PICs angeschlossen. Außerdem stehen Master und Slave über drei weitere Leitungen in Verbindung, über die der Master dem Slave mitteilen kann, wann dieser die Nummer des von ihm signalisierten Interrupts auf den Datenbus legen soll.

Software

Ignorieren eines Interrupts

CPU-seitig kann dafür gesorgt werden, dass das laufende Programm nicht durch Interrupts unterbrochen wird. Dazu wird mit der Assembleranweisung `cli` das Interrupt-Bit im EFLAGS-Register gelöscht. Der Prozessor wird auf Signale seiner INTR Leitung nun nicht mehr reagieren. Da der PIC einen Interrupt frühestens dann weitergibt, wenn die CPU auf den vorhergehenden Interrupt reagiert hat, wird der PIC keine Interrupts weiterleiten, solange die CPU Interrupts ignoriert.

Dieser Vorgang kann mit dem Befehl `sti` wieder rückgängig gemacht werden.

Interrupts können auch selektiv unterdrückt werden. Dazu muss der PIC programmiert werden. Das geht wie üblich mit `in`- und `out`-Befehlen.

Software zur Interruptbehandlung

Wenn ein Interrupt ankommt, der sowohl seitens des PICs als auch seitens der CPU zugelassen wurde, dann verzweigt der Prozessor automatisch zur Interruptbehandlungsroutine.

Deren Adresse wird einer Interruptdeskriptortabelle (IDT) entnommen, wobei die von dem PIC auf den Datenbus gelegte Nummer des Interrupts als Index dient. Beim 8086-Prozessor war die Lage der Interruptdeskriptortabelle noch fest von der Hardware vorgegeben, beim 80386 wird ihr Anfang und ihre Größe durch das IDT-Register beschrieben.

Die Interruptdeskriptortabelle kann maximal 256 Interruptgate-Deskriptoren enthalten, von denen es drei verschiedene Typen gibt:

Task-Gate

> Der Interruptgate-Deskriptor zeigt auf einen Task, einen hardwaremäßig unterstützten Prozess. Wenn der entsprechende Interrupt eintritt, führt der Prozessor automatisch einen Taskwechsel zu dem angegebenen Interrupt-Task durch.

Interrupt-Gate

> Der Interruptgate-Deskriptor zeigt auf eine Prozedur, die als Interruptbehandlungsroutine ohne vorherigen Taskwechsel aufgerufen wird.

Trap-Gate

> Der Trapgate-Deskriptor zeigt auf eine Prozedur, die als Trapbehandlungsroutine ohne vorherigen Taskwechsel aufgerufen wird.

Bevor der Prozessor infolge eines Interrupts oder Traps die angegebene Behandlungsroutine aufruft, legt er den aktuellen Inhalt des EFLAGS-Registers auf den Stack ab. Dies ermöglicht es ihm, nun das Interrupt-Enable-Flag im EFLAGS-Register zu löschen und auf diese Weise die geschachtelte Behandlung weiterer Interrupts zu verhindern. Wie bei einem normalen Funktionsaufruf wird dann noch die Rücksprungadresse (Inhalt von Code-Segment und Instruction Pointer) auf dem Stack gesichert, bevor die Behandlungsroutine begonnen wird. Bei manchen Exceptions legt der Prozessor zusätzlich einen Fehlercode auf dem Stack ab.

Wurde die Unterbrechung durch einen Interrupt ausgelöst, so besteht eine Aufgabe der Interruptbehandlungsroutine darin, dem PIC mitzuteilen, dass der Interrupt behandelt wurde. Anderenfalls wird der PIC nämlich keine weiteren Interrupts desselben Gerätes weiterleiten. Das Senden des Interrupt-Acknowledge-Signals erfolgt wieder mit einem (bzw. bei einer Kaskadierung der PICs mit zwei) `out` Befehl(en) an den Port des PICs.

Mit dem Befehl `iret` wird die Unterbrechungsbehandlung abgeschlossen. Der Prozessor holt die Rücksprungadresse vom Stack, stellt den Inhalt des EFLAGS-Registers wieder her und kehrt zu der unterbrochenen Funktion zurück. Dadurch, dass auch die EFLAGS wieder hergestellt werden, werden spätestens jetzt die Interrupts CPU-seitig wieder zugelassen.

Zugriff auf die PICs über Ports

Jedem der beiden PICs in einem PC sind zwei Ports im I/O-Adressraum zugeordnet, die wie folgt belegt sind:

Port (IRQ0-7/IRQ8-15)	Lesedaten	Schreibdaten
0x20/0xa0	IRR,ISR,Int.Vektor	ICW1,OCW2,OCW3
0x21/0xa1	IMR	ICW2,ICW3,ICW4,OCW1

Die Bezeichnungen *ICW* und *OCW* stehend dabei für *Initialization Control Word* und *Operation Control Word*. Relevant sind nur die Operationsbefehlsworte OCW1 und OCW2:

- OCW1 ist dabei schlicht ein Platzhalter für die Maske der gesperrten Interrupts. Diese Maske wird direkt in das Interrupt Mask Register (IMR, 0x21/0xa1) übertragen; ein gesetztes Bit X sorgt dafür, dass Interrupt X gesperrt wird.
- OCW2 wird verwendet um dem PIC einen EOI-Befehl zu senden, damit dieser erfährt, dass der Interrupt bearbeitet wird und das gesetzte Bit im ISR löscht. Andernfalls würde der Interrupt bei "nächster Gelegenheit" (d.h., wenn die Interruptbearbeitung wieder zugelassen wird) nochmals gemeldet. **Beim HHUos wird der PIC jedoch für "automatic EOI (AEOI)" konfiguriert, wodurch das Bit im ISR vor dem Verzweigen in den Interrupt-Handler automatisch gelöscht wird. Das Senden von EOIs bleibt so also erspart.**

Die Initialisierungsbefehlsworte ICW1-ICW4 dienen der Initialisierung des Chips, wobei unter anderen der AEOI-Modus festgelegt wird. All dies erfolgt bei HHUos innerhalb des Startup-Codes (startup.asm), so dass hier nichts gemacht werden muss.

Weiterführende Informationen

Intel, "8259A - Programmable Interrupt Controller"

Unter hier: http://wiki.osdev.org/8259_PIC