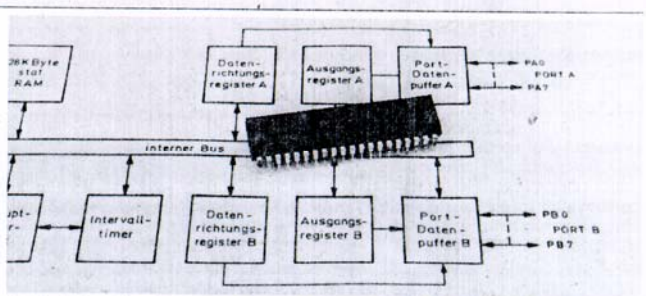
RIOT 6532: Einsatz und Programmierung

Christian Persson



Jeder Computer benötigt Ein-/Ausgabeleitungen, um mit der Umwelt in Datenverkehr treten zu können, und er braucht einen Arbeitsspeicher, um beispielsweise Variablen und Rücksprungadressen aufbewahren zu können. Bei unserem Minimal-Computer CEPAC-65, der in dieser Ausgabe vorgestellt wird, übernimmt ein IC beide Funktionen: Der RIOT-Baustein 6532 stellt sowohl zwei 8-Bit-Ports als auch 128 Bytes statisches RAM zur Verfügung. Außerdem enthält das IC einen programmierbaren Countdown-Timer und einen Flankendetektor. In diesem Beitrag sollen die vielen Einsatzmöglichkeiten des RIOT-Chips, den man in 6502-Systemen oft einsetzt, anhand von Beispielen erläutert werden.

Das statische RAM läßt sich in einem Minimal-Computer wie dem CEPAC-65 besonders problemlos verwenden: Während man in 6502-Systemen sonst durch eine zusätzliche Logik ein RAM-R/$\overline{W}$-Signal zu bilden hat, ist dies beim Einsatz des RIOT nicht erforderlich. Die CPUs der 65er-Familie verwenden den Datenbus nur in der zweiten Hälfte eines jeden Taktzyklus. Das vom Prozessor gebildete Taktsignal Φ2 zeigt an, daß die Daten auf dem Bus gültig sind. Zur Bildung des Signals RAM-R/$\overline{W}$ werden gewöhnlich Φ2 und R/$\overline{W}$ durch Gatter verknüpft. Beim Einsatz des 6532 kann man auf die Gatterschaltung verzichten, weil der Chip mit der Prozessorleitung Φ2 verbunden ist und die Verknüpfung intern ausführt.

16 I/O-Leitungen

Die beiden I/O-Ports werden üblicherweise mit A und B bezeichnet. Sie bestehen aus jeweils acht Leitungen, die einzeln wahlweise als Ein- oder Ausgänge 'deklariert' werden können. Zu diesem Zweck sind den beiden Ports 'Datenrichtungsregister' zugeordnet. Diese werden PADD und PBDD genannt ('Port A/B Data Direction'). Jedes Bit der Datenworte in den Datenrichtungsregistern korrespondiert mit einer Portleitung (Bild 1): Eine '0' macht die Portleitung zu einem Eingang, eine '1' deklariert den Anschluß als Ausgang. Eine bestimmte Konfiguration von Ein-/Ausgabeleitungen läßt sich also sehr einfach dadurch herstellen, daß man die entsprechenden Daten in die Richtungsregister schreibt.

Die in Bild 2 dargestellte Konfiguration ließe sich also durch folgende Befehlssequenz einstellen:

```
LDA #3F
STA PADD
```

Den beiden Ports sind außerdem die Datenregister PAD und PBD ('Port A/B Data') zugeordnet. Diese beiden Register dienen als Zwischenspeicher (Latch) für Daten, die an den Ports ausgegeben werden sollen. Portleitungen, die als Eingänge deklariert sind, werden durch die interne Logik von den Datenregistern getrennt.

Die CPU kann in die Datenrichtungsregister und in die Datenregister schreiben, als handele es sich um gewöhnliche RAM-Zellen. (Bekanntlich sind in 6502-Systemen die I/O-Ports 'memory-mapped'; sie liegen in demselben Adreßraum wie der Speicher.) Um alle als Ausgänge deklarierten Leitungen von Port A auf log. 0 zu legen, könnte man zum Beispiel diese Befehlsfolge ausführen lassen:

```
LDA #00
STA PAD
```

Als Ergebnis wird sich an den Portschlüssen PA0...PA5 der log.-0-Pegel einstellen, wenn der Port entsprechend Bild 2 konfiguriert ist. Der Pegel an den als Eingänge deklarierten Leitungen PA6 und PA7 wird natürlich nicht beeinflußt, obwohl auch die Bits 6 und 7 im Datenregister PAD auf 0 gesetzt worden sind.

Nehmen wir an, die beiden höchstwertigen Anschlüsse würden von außen auf log. 1 gelegt (offene Eingänge liegen ebenfalls auf 1). Liest die CPU nun den Inhalt des Registers PAD (zum Beispiel mit LDA PAD), so übernimmt sie das Datenwort 11 00 00 00, entsprechend C0h. Die Bits 0...5 entsprechen denen im Datenregi-

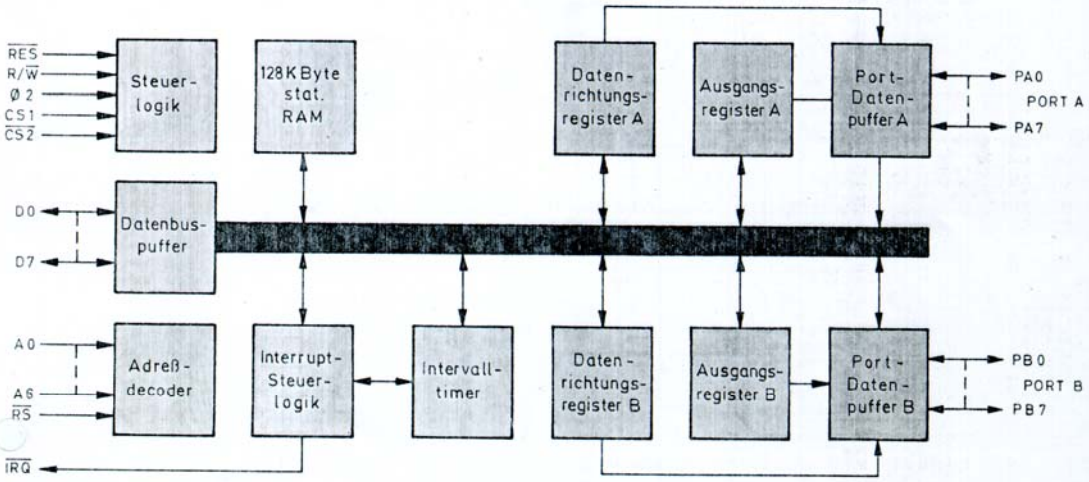


Bild 1. Innere Struktur der 6532

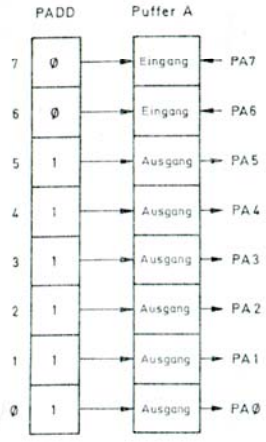


Bild 2. Deklarieren von Ein- und Ausgängen durch Datenrichtungsregister

ster, die Bits 6 und 7 repräsentieren den Pegel an den beiden als Eingänge definierten Leitungen. Beim Lesevorgang werden die als Eingänge deklarierten Leitungen direkt auf den Datenbus geschaltet, die betreffenden Bitwerte geben wieder den momentanen Pegel des anliegenden Signals.

Elektrisch verhalten sich die Portleitungen ähnlich wie LS-TTL-Anschlüsse: Pegel zwischen 0 und 0,8 Volt werden als log. 0 erkannt, Pegel zwischen 2 und 5 Volt gelten als log. 1; der Bereich dazwischen ist 'nicht erlaubt'. Ein als Eingang geschalteter Portanschluß stellt eine LS-TTL-Last dar. Die als Ausgang verwendete Portleitung kann eine LS-TTL-Last treiben. Port B besitzt im Unterschied zu Port A spezielle Treiber an den Ausgängen und kann einen Quellstrom von 3 mA bei 1,5 V liefern; er ist damit zum direkten Anschluß von Darlington-Leistungstreibern geeignet. Die Ausgangsbelastung wirkt übrigens auf das Datenregister zurück: Bei einem Lesevorgang kann der Pegel nicht korrekt erkannt werden, wenn die Spannung am Ausgang infolge Belastung im Bereich zwischen 0,4 und 2,4 Volt liegt.

Das folgende kleine Programm erzeugt ein Rechteck-Signal an einer der Portleitungen. Dies geschieht, indem der Pegel in regelmäßigen Abständen zwischen 0 und 1 umgeschaltet wird. Die Variable TIME bestimmt die Dauer der Wartezeit zwischen den Schaltvorgängen und damit die Frequenz. Vorausgesetzt wird bei diesem Programm, daß die betreffende Portleitung zuvor als Ausgang deklariert worden ist.

```
SQUARE: LDA PBD
        EOR #20        ;Bit 5 invertieren
        STA PBD
        LDX TIME
LOOP:   DEX
        BNE LOOP       ;Zeit abwarten
        BEQ SQUARE     ;wieder invertieren
```

Die drei Instruktionen zu Beginn zeigen, wie man es bewerkstelligt, daß sich der Pegel an nur einer Ausgangsleitung ändert, wenn der an den übrigen nicht bekannt ist. Mit dieser Instruktionsfolge sollte man immer dann arbeiten, wenn an dem betreffenden Port mehrere Geräte angeschlossen sein könnten.

Hier ein simples Programm zur Überwachung des Pegels an einem Eingang:

```
POLLIN: LDA PAD
        AND #80        ;Bits Ø...6
                        maskieren
        BNE POLLIN     ;Bit 7 nicht Ø:
                        wiederholen
```

Es wird das sogenannte Polling-Verfahren angewandt: Die CPU fragt in einer Programmschleife so lange den Eingang ab, bis sich der betreffende Bitwert von 1 auf 0 ändert. Der BIT-Befehl des 6502 erlaubt eine Vereinfachung des Programms, wenn es sich um eine der höchstwertigen Portleitungen handelt. Man verwendet deshalb bevorzugt diese

Leitungen für die Abfrage von Statusanzeigen bei Peripheriegeräten:

```
POLLIN: BIT PAD
        BMI POLLIN  ;Warten,
                    bis Bit 7 = 0
```

(für die Überwachung von PA7)

```
POLLIN: BIT PAD
        BVS POLLIN  ;Warten,
                    bis Bit = 0
```

(für die Überwachung von PA6)

Flankendetektor

Tritt an der überwachten Leitung ein sehr kurzer Impuls auf, so kann es allerdings vorkommen, daß der CPU dies entgeht, weil zum Zeitpunkt der Abfrage schon der vorherige Pegel wiederhergestellt ist. Der RIOT-Baustein bietet jedoch mit dem Flankendetektor eine Möglichkeit an, dieses Problem auszuschalten. Der Detektor erkennt — je nach Programmierung — eine negative oder positive Flanke an der Leitung PA7. Dies wird durch Setzen einer Flag in einem Statusregister des RIOT-Chips angezeigt. Das Register heißt RDFLAG; die sogenannte PA7-Flag ist das Bit 6. Sein Wert kann im Polling-Verfahren durch eine BIT-Instruktion abgefragt werden; er beeinflußt die Overflow-Flag im Prozessor-Statusregister:

```
POLLF:  BIT RDFLAG
        BVC POLLF   ;Warten,
                    bis Bit 6 = 1
```

Ein wesentlicher Nachteil des Polling-Verfahrens besteht darin, daß es die CPU blockiert: Sie kann nur dann innerhalb kürzester Frist auf eine Statusänderung reagieren, wenn sie innerhalb der Abfrageschleife keine anderen Programmschritte ausführt. Es gibt aber auch für dieses Problem eine Lösung: Der Flankendetektor läßt sich so programmieren, daß bei einer Pegeländerung an der Leitung PA7 ein Interrupt-Request (IRQ) erfolgt. Die Interrupt-Anforderung wird gleichzeitig mit dem Setzen der PA7-Flag abgegeben, indem der RIOT die IRQ-Leitung der CPU auf log. 0 zieht.

Der Flankendetektor wird programmiert, indem die CPU beliebige Daten in eines der Steuerregister schreibt. Vier solcher Register sind vorhanden:

EDETA: negative Flanke detektieren
EDETB: positive Flanke detektieren
EDETC: negative Flanke detektieren und IRQ auslösen
EDETD: positive Flanke detektieren und IRQ auslösen

Countdown

Vor allem bei Steuerungsaufgaben spielen Zeitabläufe eine wichtige Rolle. Der Einsatz von Verzögerungsschleifen, in denen sich der Prozessor mit einer Anzahl sinnloser Instruktionen aufhält, ist aus mancherlei Gründen nachteilig, vor allem dann, wenn die Rechenzeit dringend benötigt wird. Der Countdown-Timer des RIOT 6532 ermöglicht es, den Computer sinnvoller einzusetzen als zum Abzählen von Zeiteinheiten. Er läuft im Systemtakt unabhängig von der CPU und meldet nach Ablauf einer vorher programmierten Zeitspanne den 'Time Out'.

Dieses Ergebnis wird ebenfalls durch das Setzen einer Statusflag angezeigt. Bit 7 des Registers RDFLAG ist das Timer-Statusbit. Ähnlich wie bei Anwendung des Flankendetektors kann der RIOT-Baustein aber auch so programmiert werden, daß er gleichzeitig mit dem Setzen der Timer-Flag einen IRQ auslöst.

Der Timer ist ein 8-Bit-Zähler, so daß maximal 255 Zeiteinheiten vorprogrammiert werden können. Zusätzlich hat der Programmierer die Wahl zwischen vier verschieden langen Zeiteinheiten, nämlich 1xT, 8xT, 64xT und 1024xT, wobei T für die Dauer eines Taktzyklus' steht. In einem Standard-6502-System beträgt die Taktfrequenz gewöhnlich 1 MHz, T ist also 1 µs. Der Multiplikationsfaktor ergibt sich aus der Wahl des Registers, in das die CPU die gewünschte Anzahl von Zeiteinheiten bis zum Time Out schreibt. Die Wahl des Registers bestimmt auch, ob der RIOT nach dem Countdown einen Interrupt auslöst. Dies sind die Timer-Register und ihre Funktionen:

CNTA: 1xT
CNTB: 8xT
CNTC: 64xT
CNTD: 1024xT
CNTE: 1xT, IRQ
CNTF: 8xT, IRQ
CNTG: 64xT, IRQ
CNTH: 1024xT, IRQ

Ein Timer-Start könnte beispielsweise folgendermaßen ablaufen:

```
LDA #10
STA CNTG
```

Mit dem Schreiben des sogenannten 'Timer-Offsets' in das Register CNTG wird die Timer-Flag im Register RDFLAG zurückgesetzt. Alle 64 µs nach diesem Vorgang zählt der RIOT eine Zeiteinheit ab. Beim Zählerstand 00 ist der Time Out erreicht. Einen Taktzyklus später ist die Timer-Flag gesetzt, und die IRQ-Leitung wird auf log. 0 gezogen. Die CPU erhält damit einen Interrupt Request, der allerdings nur dann zur Ausführung einer Interrupt-Routine führt, wenn die I-Flag im Prozessor-Statusregister zurückgesetzt ist.

128 Byte RAM	xx00...xx7F
PAD	xx80
PADD	xx81
PBD	xx82
PBDD	xx83
RDTDIS	xxD4
RDFLAG	xxD5
RDTEN	xxDC
EDETA	xxE4
EDETB	xxE5
EDETC	xxE6
EDETD	xxE7
CNTA	xxF4
CNTB	xxF5
CNTC	xxF6
CNTD	xxF7
CNTE	xxFC
CNTF	xxFD
CNTG	xxFE
CNTH	xxFF

Tabelle 1. Typische Adressenlage der RIOT-Register

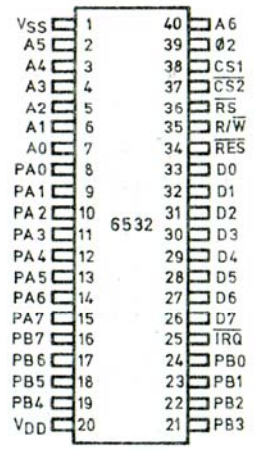


Bild 3. Pinbelegung des RIOT

Während der Timer-Laufzeit kann die CPU die verbleibende Anzahl von Zeiteinheiten in einem der Register RDTEN oder RDTDIS lesen. Liest die CPU das Register RDTDIS, so wird damit ein IRQ untersagt. Liest sie das Register RDTEN, so wird damit der IRQ erlaubt — unabhängig von der vorherigen Betriebsart.

Um die Timer-Flag in Polling Mode zu testen, eignet sich wiederum eine BIT-Instruktion:

```
TFTEST: BIT RDFLAG  ;Timer-Flag gesetzt?
        BPL TFTEST  ;Nein: Warten
```

Das folgende Programm dient dazu, die Länge eines Impulses zu messen:

```
START:  SEC
        LDA #FF
WAIT:   BIT PAD
        BMI START   ;Warte auf
                    L-Impuls an PA7
        STA CNTx    ;Starte Timer
WAITB:  BIT PAD
        BPL PAD     ;Warte auf H-Pegel
        SBC RDTDIS
        STA LENGTH  ;Länge ablegen
```

Ein wichtiger Hinweis für die Anwendung der Interrupt-Technik: Während der Ausführung einer Interrupt-Routine muß die Ursache des Interrupt beseitigt werden, sonst bleibt die IRQ-Leitung auf L-Pegel, und bei der Rückkehr wird sogleich die nächste Unterbrechung ausgelöst. Der RIOT gibt die IRQ-Leitung frei, wenn die CPU den Timer neu startet (wahlweise für IRQ beim nächsten Time Out) oder RDTDIS liest. Wurde der Interrupt durch den Flankendetektor ausgelöst, so wird durch Lesen von RDFLAG die IRQ-Leitung freigegeben; gleichzeitig wird die PA7-Flag zurückgesetzt. Es empfiehlt sich, den Flankendetektor auch beim erstmaligen Einsatz in einem Programm durch Lesen von RDFLAG zu initialisieren.

Adressenlage

Welche effektiven Speicheradressen die verschiedenen RIOT-Register haben, hängt natürlich vom Anschluß des Bausteins ab. Der RIOT besitzt sieben Adreßleitungen, die in der Regel mit den niederwertigen Prozessor-Adreßleitungen verbunden werden. Verwendet man die Adreßleitung A7 — wie beim CEPAC 65 — zur Ansteuerung des Pins 36 ($\overline{\text{RAM-Select}}$), dann ergibt sich die in Tabelle 1 aufgeführte Adressenlage. Dabei belegt der RIOT insgesamt eine Speicherseite (256 Byte); die untere Hälfte davon enthält das RAM, die obere die Steuer- und Portregister. Auf welcher Speicherseite der Baustein liegt, hängt wiederum von der übrigen Adreßdekodierung des Systems ab, aus der das Chip-Select-Signal gebildet wird. □