Roland Löhr

Das VIA 6522 -

ein intelligenter Interfacebaustein

Was ist Interfacing?

In blockmäßiger Beschreibung besteht ein Mikroprozessor aus Zentraleinheit (CPU) mit Taktgenerator, Speichern und Eingabe/Ausgabe (E/A oder I/O). Als Festwertspeicher für Programme, Konstante usw. dienen dabei ROMs, PROMs, EPROMs, Zwischenergebnisse werden in RAMs als Schreib-/Lesespeicher abgelegt. Allein mit CPU und Speichern könnte ein Prozessor zwar arbeiten, er könnte aber von der Umwelt keine Signale und Anweisungen empfangen und auch keinerlei Ergebnisse und Steuerungen dorthin abgeben. Daher der Bedarf für Schnittstellen, Interfaces zur Ein- und Ausgabe. Schnittstellen mögen dabei zu anderen elektronischen Systemen z. B. zur Prozeßsteuerung bestehen oder zu Geräten, die der Verständigung mit dem Menschen dienen, wie Tastaturen, Anzeigen, Drucker. Für das Interfacing wurde eine Vielzahl von Bausteinen geschaffen, die oft nur für eine bestimmte Anwendung ausgelegt sind, man denke an Tastaturenkoder oder Video-Controller.

Die Prozessorbausteine CPU, Speicher und E/A werden untereinander durch drei Leiterbahnsysteme verbunden, den Adreßbus, den Datenbus und den Steuerbus (s. *Bild 1*).

Die Busse werden von der Zentraleinheit regiert. Auf dem im allgemeinen 16 bit breiten Adreßbus sendet sie Adressen zur gezielten Ansprache einzelner Speicherzellen, der Datenbus (8 bit) transportiert Informationen in zwei Richtungen, entweder zur CPU oder von der CPU in die Speicher. Der Kontrollbus schließlich führt Signale zur Synchronisation aller Bauteile und für die Bestimmung, ob es sich um einen Schreib- oder Lesezyklus der Zentraleinheit handelt.

Für die sehr verbreitete 65XX-Prozessorfamilie, repräsentiert durch die Systeme AIM-65, PC-100, PET, SYM-1 und KIM-1, besprechen wir den außerordentlich leistungsfähigen Versatile Interface Adapter (VIA) 6522 *(Bild 2)*. Er wird von Rockwell, MOS-Technology und Synertek hergestellt und ist auf den vier erstgenannten Geräten z. T. mehrfach enthalten, um die Systemhardware zu bedienen. Aber er steht dem Benutzer dort auch für eigene Anwendungen zur Verfügung (Application Connector, bzw. User Port).

Dieser Interfacebaustein eignet ebenso für andere Prozessoren, vor allem für die sehr verwandte 68XX-Serie von Motorola. Er weist große Ähnlichkeiten mit dem 6820/6520-PIA auf (Peripheral Interface Adapter) und auch mit dem IC 6530 im KIM-1. Das VIA vermag aber wesentlich mehr als die eben genannten. – Unsere Darstellung soll die bei den Herstellern und ihren Distributoren erhältlichen Datenblätter nicht ersetzen, sondern die vielen Möglichkeiten des Interfacing mit dem VIA in einer Übersicht aufzeichnen.

In der blockmäßigen Vereinfachung gemäß *Bild 2* können wir das VIA als einen mit der CPU über Adreß-, Daten- und Steuerbus verbundenen Interfacebaustein mit 16 Registeradressen bezeichnen, der zur Verbindung mit der Umwelt 2 Ports mit je 8 E/A-Pins und 4 Pins für Steuerfunktionen trägt. Insgesamt stehen damit 20 E/A-Leitungen zur Verfügung.

Verbindung zur CPU und Adressierung

Im Konzept der 65er-Familie werden die Register der Interfacebausteine von der CPU wie ganz normale Speicherzellen gelesen und beschrieben. Daher der volle Anschluß an den Datenbus und an die Steuerleitungen R/W und Φ2. Um Pins zu sparen, werden nicht alle 16 Adreßleitungen auf den Chip gelegt, sondern nur 4 als Register Select (RS0-RS3). Das reicht zur Ansprache der 16 Maschinenregister aus. Und über eine externe Decodierlogik wird aus den Signalen der höheren Adreßbusleitungen ein Chip-Select gebildet (CS1, CS2).

Je nach dem bei der Decodierung betriebenen Hardwareaufwand mag sich dabei für mehrere Interfacebausteine eine lückenlose Aneinanderreihung von Registeradressen ergeben oder auch nicht. Im letzteren Fall mag ein Baustein wegen zu grober Decodierung die Adressen einer ganzen Page oder gar eines ganzen KByte belegen. Für eine spätere Expansion des Systems im vorhandenen Adressenraum kann das hinderlich sein.

Zur Verbindung mit dem Steuerbus gehört auch die Interrupt-Leitung IRQ. Damit ist nicht gesagt, daß das VIA im Interrupt betrieben wird. Vielmehr: Es darf einen Interrupt auslösen, wenn der Programmierer das für gewisse Ereignisse vorgesehen hat.

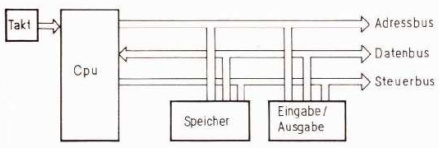


Bild 1. Blockschaltbild eines einfachen Mikrocomputers

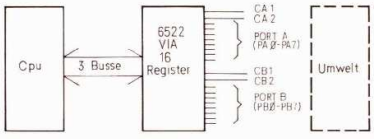


Bild 2. Der Mikroprozessor kann über das VIA 6522 mit der Außenwelt kommunizieren

Funktionskontrolle durch 16 Register

Die modernen Interfacebausteine enthalten ein feines Netz digitaler Schaltungen. Sie können nach dem Willen des Anwenders in ihrer Funktion programmiert werden, indem er bei Programmbeginn gewisse Bitmuster in die Steuerregister des Interfacebausteines hineinschreibt. Dadurch werden die benötigten digitalen Funktionen auf dem Baustein *softwaremäßig* durchgeschaltet. Seitens der CPU erfolgt diese Intitialisierung mit STA-Befehlen (speichere Akku). Mit der Zahl der jeweiligen Register steigt im allgemeinen auch die Vielseitigkeit eines Peripheriebausteines. Dem VIA sind 16 Registeradressen zugeordnet, in Wirklichkeit ist eine noch größere Zahl an Steuerungen im Einsatz.

Auf die Register des VIA kommen wir noch im einzelnen zu sprechen. Wir gliedern sie zunächst schematisch wie folgt:

Zwei Datenrichtungsregister, DDRA und DDRB, die die Funktionen der Ports bitweise (pinweise) für Eingabe oder Ausgabe festlegen.

Zwei Ports mit Ausgaberegister ORA, bzw. ORB, Leseregister IRA (IRB) und einschaltbarem Lese-Latch.

Zwei 16 bit breite Timer, die zahlreiche Funktionen ausüben können, wie Impulsweitenmessung, Ereigniszählung, Erzeugung von einmaligen oder stetigen Rechteckimpulsen an PB7 und Kontrolle seriellen Datenverkehrs.

Schieberegister SR für die serielle Ein- und Ausgabe an Pin CB2.

Hilfsregister ACR, das die Funktionen der Timer, der Pins PB7 und PB6 und die Vorspeicherung einkommender Signale an den Latches der Ports A und B reguliert.

Steuerregister PCR zur Funktionskontrolle der Pins CA1, CA2, CB1 und CB2 für die Reaktion auf Impulsflanken und für den Handshake-Verkehr mit Peripherieeinheiten.

Interrupt-Anzeigeregister IFR. Die Interruptkontrolle findet in 3 Stufen statt. Im IFR können die möglichen Interrupt-Quellen (nämlich die 4 Steuerleitungen, die beiden Timer bei Nulldurchgang und das Schieberegister nach Abarbeitung) lediglich ein Flag setzen. Bit 7 ist logisch ODER auf die nachfolgenden Flags gesetzt und wird ‚1' sobald nur ein Flag ‚1' wird. Das ermöglicht eine schnelle Abfrage mit dem BIT-Befehl.

Interrupt Enable Register IER. Für jedes der vorgenannten Flags (Interruptquellen) kann durch Setzen seines Bits in diesem Register bestimmt werden, ob es einen Interrupt auslösen darf. Wenn ja, dann bringt das auslösende Ereignis das Ausgangssignal IRQ des Chips auf ‚0' (dritte Stufe).

In der *Tabelle* sind für den AIM-65 die Adressen des dem Anwender voll zur Verfügung stehenden ‚User VIA' und des für Systemzwecke eingesetzten VIA aufgeführt, für den PET die Adressen des gemischt genutzten VIA. Die Hälfte der Leitungen ist hier für das System genutzt, die 8 Leitungen des Port A sowie CA1 und CB2 sind auf den „User Port" geführt.

Tabelle: VIA-Registeradressen beim AIM 65 und beim PET2001

AIM Anwender-VIA hexadezimal	AIM System-VIA hexadezimal	PET2001 System- und Anwender-VIA hexadezimal	bzw. dezimal	Register
A000	A800	E840	59456	ORB Ausgaberegister Port B
A001	A801	E841	59457	ORA Ausgaberegister Port A (mit Handshake)
A002	A802	E842	59458	DDRB Datenrichtunsregister für Port B
A003	A803	E843	59459	DDRA Datenrichtungsregister für Port A
				Timer 1:
A004	A804	E844	59460	T1C-L Lesen Zähler ‚low', Schreiben Vorspeicher ‚low'
A005	A805	E845	59461	T1C-H Lesen und Schreiben Zähler ‚high'
A006	A806	E846	59462	T1L-L Lesen oder Schreiben in Vorspeicher ‚low'
A007	A807	E847	59463	T1L-H Lesen oder Schreiben in Vorspeicher ‚high'
				Timer 2:
A008	A808	E848	59464	T2C-L Lesen Zähler ‚low', Schreiben Vorspeicher ‚low'
A009	A809	E849	59465	T2C-H Lesen und Schreiben Zähler ‚high'
A00A	A80A	E84A	59466	SR Schieberegister
A00B	A80B	E84B	59467	ACR Hilfsregister
A00C	A80C	E84C	59468	PCR Steuerregister
A00D	A80D	E84D	59469	IFR Interrupt-Anzeigeregister
A00E	A80E	E84E	59470	IER Interrupt Enable Register
A00F	A80F	E84F	59471	ORA Ausgaberegister für Port A (ohne Einfluß auf Handshakesignal)

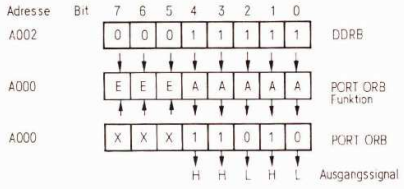


Bild 3. Ein Port besteht aus acht Leitungen; ob es sich dabei um Ein- oder Ausgänge handelt, wird per Software mit dem Datenrichtungs-Register DDRB (hier Port B) festgelegt

Die Ports als E/A-Leitungen

Die wohl einfachste Nutzung des VIA besteht darin, an den Ports Signale zu senden oder zu empfangen. Ein Port muß dabei keineswegs nur als Eingang oder nur als Ausgang betrieben werden, er kann beide Funktionen gemischt wahrnehmen, weil jeder Pin individuell geschaltet werden kann. Dazu schreibt man zunächst ein Bitmuster in das Datenrichtungsregister DDRA bzw. DDRB. Eine ‚1' schaltet den dazugehörigen Pin auf Ausgang, eine ‚0' auf Eingang. Das eigentliche Ausgangssignal wird erst danach durch Einschreiben in das Ausgangsregister ORA (ORB) erzeugt. Für das Beispiel gemäß *Bild 3*, das die Pins PB7-5 als Eingang schaltet und PB4-0 als Ausgang, programmiert man:

```
LDA #$ 1F    FESTLEGUNG
             DER PINS FÜR E/A
STA DDRB     IM DATEN-
             RICHTUNGSREGISTER
LDA #$1A     ERWÜNSCHTE AUSGANGS-
             SIGNALE „HIGH" UND „LOW"
STA ORB      IN DEN PORT
```

Über die TTL-Kompatibilität und Belastbarkeit der Pins orientiere man sich in den Datenblättern. Daß PB7 als Output für Timer-kontrollierte Impulse

Sonderfunktionen übernehmen und daß PB6 für Ereigniszählung dienen kann, wurde bereits erwähnt und findet weitere eigene Darstellung.

Während die Dinge für das Schreiben in die Peripherie übersichtlich und einfach liegen, sind für das Lesen einige Besonderheiten zu beachten, mit deutlichen Unterschieden für die A- und B-Seite des VIA. Zunächst ist zu erwähnen, daß jede Seite neben dem Output-Register (ORA oder ORB) zwei weitere Schaltungen aufweist, das Leseregister (Input Register IRA bzw. IRB) und das Lese-Latch. Was in diesem Leseregister steht und was damit auf den Datenbus gelangt, hängt nicht nur vom anstehenden Signal, sondern auch von der Funktion ab, die den Steuerleitungen zugewiesen wurde. Vorgreifend ist darauf hinzuweisen, daß die Steuerleitungen CA1 und CB1 durch Einschreiben in das Steuerleitungsregister PCR auf das Erkennen aktiver Übergänge geschaltet werden können (aufsteigende oder abfallende Impulsflanke) und daß in Abhängigkeit davon verschiedene Vorgänge im Chip auslösbar sind, unter anderem auch ein Handshake an dem korrespondierenden Pin CA2 bzw. CB2 als Ausgang (Handshake Control).

Und dann gibt es da noch die Latching-Kontrolle, von den beiden niederwertigsten Bit des Hilfsregisters ACR kontrolliert und unabhängig für die A- und für die B-Seite ausübbar. Das bedeutet: Man liest den Port so, wie er sich im Moment des Lesens befindet (das ACR-Bit ist in Normalstellung auf ‚0') oder so, wie er sich vor dem aktiven Übergang am CA1- bzw. CB1-Pin befunden hatte (ACR-Bit ‚1').

Damit nicht genug der Besonderheiten. Es sind ja Fälle denkbar, in denen ein Ausgangspin unter der zu treibenden Last unter die Spannungsschwelle absinkt, die mit ‚high' erkannt wird, obwohl er mit ‚high' beschrieben wurde. Im Leseregister B lesen wir in diesen Fällen einen Mix aus Pin-Ausgangs-Sollsignalen und tatsächlichen Eingangspegeln, solange das von CB1 gesetzte Interrupt-Flag ‚high' ist. Sobald dieses Flag gelöscht ist, liest man die tatsächlichen Pegel an den Pins, ‚low' und ‚high'.

Nachzutragen ist: Für den Prozessor haben die Leseregister IRA, bzw. IRB diesselben Adressen wie die Ausgangsports. Sofern man nicht im Handshake-Betrieb arbeitet (s.u.), kann man die A-Seite gleichmäßig in den Adressen A001 und A00F auslesen. Das Latching eröffnet die Möglichkeit, die Eingangskombination nach dem aktiven Übergang mit derjenigen zuvor z. B. mit logisch EOR zu vergleichen. Dabei werden die Veränderungen erkannt.

Das nachfolgende Beispiel initialisiert wie folgt: CB1 reagiert auf aufsteigende Impulsflanke, Port B ist als Eingang geschaltet mit Latching der Signale.

```
LDA #$02    ERMÖGLICHE LATCHING
            AN DER B-SEITE
STA ACR     HILFSREGISTER
LDA #$10    CB1 AUF AUFSTEIGENDE
            FLANKE TRIGGERN
STA PCR     STEUERREGISTER
LDA #$00    PORT B ALS INPUT
STA DDRB    DATENRICHTUNGS-
            REGISTER
```

Bei dieser Initialisierung wird ein aktiver Übergang an CB1 das Interrupt-Flag dieses Pins setzen und zugleich auch logisch ODER das Bit 7 im Interrupt-Anzeigeregister. Im weiteren Programmverlauf kann man daher die Interruptbedingung wie folgt abfragen:

```
    LDA #$10  MASKE
W BIT IFR     DERBIT-BEFEHL VOLLZIEHT
              LOGISCH AND MIT DER
              MASKE
    BEQ W     WARTESCHLEIFE
```

Die häufig gebrauchten Techniken der Portbedienung dürften weitgehend bekannt sein. Z. B. Abfrage, ob sich irgendein Eingangssignal geändert hat:

```
    LDA PORT  ALTE SIGNALKOMBINATION
W CMP         WARTE, BIS SICH ETWAS
  PORT        ÄNDERT
    BEQ W     WARTESCHLEIFE
```

An Ausgangsports lassen sich – unabhängig von den besonderen Fähigkeiten des VIA – Rechteckimpulse durch Flippen des niedrigsten Bits wie folgt erzeugen:

```
INC PORT    BIT 0 AUF ‚1' SETZEN,
            WENN ES ZUVOR ‚0' WAR
DEC PORT    UND WIEDER AUF ‚0'
```

Für Tastaturabfragen u. ä. ist es geläufig, eine Rasterung von Gitterpunkten in der Weise vorzunehmen, daß man eine logische „0" durch einen Sendeport shiftet (ASL oder LSR) und an einem Empfangsport abfragt, in welcher Bitposition die „0" empfangen wurde. Man nimmt die „0", weil unbeschaltete offene Leiterbahnen „1" senden.

Durch Beschaltung eines Ports mit z. B. 2 Stück 4 zu 16 Decodern/Multiplexern kann die Zahl der möglichen Verbindungen zur Außenwelt ggfs. vervielfacht werden. Es sind natürlich auch andere Beschaltungen für das Multiplexing möglich, wie z. B. mit dem 4016 oder 74 157. Mit einem Steuerbit könnte man 2 x 7 Datenbits umschalten.

Impulsflanken an den Eingangs-Steuerleitungen

Die vier Pins CA1, CA2, CB1 und CB2 können im Steuerleitungsregister PCR einzeln so programmiert werden, daß sie als Eingangsleitungen und in Reaktion auf steigende oder fallende Impulsflanken ihr korrespondierendes Interrupt-Flag im Interrupt-Anzeigeregister auf „1" setzen. Ob daraus nur eine Abfragemoglichkeit oder ein tatsächlicher Interrupt wird, hängt von der Programmierung des Interrupt-Enable-Registers IER ab. Wie in *Bild* 4 dargestellt, bestimmen die Bitpositionen 6, 4, 2 und 0 im PCR für CB2, CB1, CA2 bzw. CA1, ob die Reaktion bei fallender (= 0) oder bei steigender (= 1) Impulsflanke erfolgt.

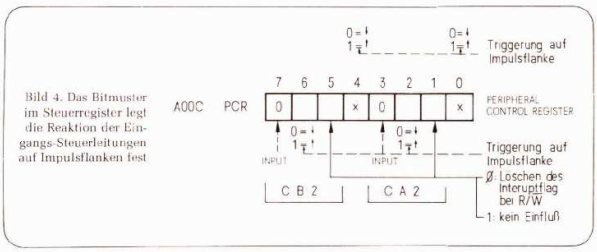


Bild 4. Das Bitmuster im Steuerregister legt die Reaktion der Eingangs-Steuerleitungen auf Impulsflanken fest

CA1 und CB1 können bei aktivem Übergang am Eingang nur ihr Interrupt-Flag setzen. CA2 und CB2 werden durch Bit 3 = ‚0' im PCR bzw. Bit7 = ‚0' als Eingänge geschaltet. Und die Bitposition 1 bzw. 5 bestimmt, ob das Interrupt-Flag durch Lesen oder Schreiben des Ports automatisch gelöscht wird (‚0': automatisch, ‚1': kein Einfluß).

Beispiel: CB2 soll das Input bei aufsteigender Impulsflanke Interrupt auslösen. Ein Lesen/Schreiben des Port B soll das Interrupt-Flag automatisch löschen:

```
LDA #$40    CB2-KONTROLLE
STA PCR     STEUERLEITUNGS-
            REGISTER
LDA #$88    INTERRUPTFREIGABE FÜR
            CB2. WEGEN BIT7 =
            ‚1' S. U.
STA IER
```

Die Steuerleitungen als Ausgang und Handshaking

Handshaking ist beim asynchronen Verkehr zwischen Prozessor und Peripherieeinheit der Signalaustausch zum Zwecke der zuverlässigen Synchronisation von Eingabe oder Ausgabe über die Ports. Englisch spricht man von ‚talker' und ‚listener' für die betroffenen Einheiten. Vom Talker wird ein Signal ‚data ready' und vom Listener eines ‚data taken' erwartet. Beim 6522 VIA stehen für diesen Signalaustausch die bereits genannten 4 Steuerleitungen zur Verfügung.

CA1 und CB1 dienen als Input für die Erkennung aktiver Übergänge. Sie setzen Interrupt-Flags. Ist eine sendende Peripherieeinheit angeschlossen, so bedeutet ihr Signal an den Eingangspins ‚ready to transmit to processor'. Handelt es sich dagegen um eine Ausgabeeinheit, so signalisiert sie dem Prozessor auf diesen Leitungen ‚ready to receive' oder ‚data taken'.

Im Handshake-Verkehr dienen dann CA2 oder CB2 als Ausgabeleitungen, um einer sendenden Einheit mitzuteilen ‚data taken' oder um einer Ausgabeeinheit anzukündigen ‚data ready for transmission'.

CA1 (CB1) ‚data taken' gemeldet wird. Erst jetzt wird das Interrupt-Flag gesetzt, damit der Prozessor neue Daten am Port bereitstellen kann.

Beispiel: Am Port A sollen Daten im Interrupt-Betrieb empfangen werden. Die sendende Einheit meldet mit aufsteigender Flanke an CA1 ‚data ready' und will einen kurzen Impuls über CA2 für ‚data taken' erhalten:

```
LDA #$0B   CA1 AUF AUFSTEI-
           GENDE FLANKE,
           CA2 AUF PULSAUSGABE
STA PCR    STEUERLEITUNGSREGISTER
LDA #$82   ERMÖGLICHE INTERRUPT
           FÜR CA1
STA IER
```

Unabhängig von solchem Handshake-Betrieb können CA2 und CB2 unter Kontrolle des Programmes auch auf definierte Pegel ‚high' oder ‚low' gesetzt werden (manual output mode) um z. B. äußere Multiplexer umzusteuern.

Die Vielseitigkeit der Ausgangssteuerung läßt für die Bedienung wohl kaum Wünsche offen, zumal noch eine Ergänzung hinzukommt: Im handshake mode an Port A geht CA2 auf ‚low', wenn man den Port in Adresse A001 liest. Wenn neue Daten noch nicht bereitstehen, könnte das in zeitlicher Überschneidung zu Mißverständnissen

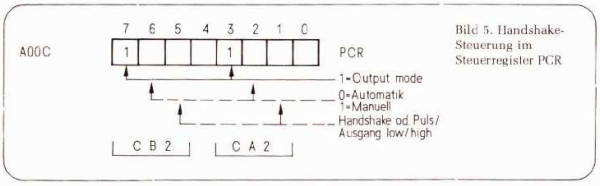


Bild 5. Handshake-Steuerung im Steuerregister PCR

CA2 und CB2 werden als Ausgänge geschaltet, indem man in Bitposition 3 bzw. 7 des PCR eine ‚1' einschreibt (*Bild 5*). Für jede Seite des VIA gibt es 4 Möglichkeiten zur Erzeugung des Ausgangssignales. Die eleganteste finden wir nur auf der A-Seite, das automatische Handshaking: Sobald man (bei der Eingabe in den Prozessor) in der Prüf- oder Interruptroutine den Port A liest, geht CA2 in automatischer Beantwortung auf ‚low'. Das bedeutet ‚data taken'. Im handshake mode geht CA2 erst dann wieder auf ‚high', wenn die Peripherie durch aktiven Übergang an CA1 meldet ‚new data ready'. Daneben gibt es auch eine Betriebsart, in der CA2 oder CB2 mit einem kurzen Impuls quittieren.

Ähnlich sinnvoll liegen die Verhältnisse bei der Ausgabe an die Peripherie: Ein Beschreiben des Ports A oder B setzt CA2 bzw. CB2 auf ‚low'. Damit wird ‚data ready for transmission' signalisiert. Das Ausgangssignal verharrt auf diesem Pegel, bis von draußen über führen. Es wurde daher für Zwischenprüfungen die Adresse A00F geschaffen, deren Einlesung oder Beschreibung die Handshake-Steuerung nicht verändert. Damit ist auch die Option eröffnet, den Port gemischt als Ein- und Ausgang zu fahren. Die Steuerung entsprechender Pins eines Ausganges oder die Prüfung eines zweiten Einganges beeinflußt dann nicht die ebenfalls angeschlossene Handshake-Einheit.

Die Timer/Zähler

Das VIA trägt zwei Timer, T1 und T2, von je 16 bit Breite. Anders als bei den Timern z. B. des 6532 RIOT und des 6530 RRIOT (im KIM) arbeiten sie nicht mit einstellbaren Teil- oder Zählgeschwindigkeiten (z. B.: 8, : 1024), sondern zählen die 2 Byte breite Vorgabe mit jedem Maschinentakt um 1 herunter. Beide Timer setzen bei ihrem Nulldurchgang ein eigenes Interrupt-Flag im Interrupt-Anzeigeregister (IFR) und können auch wahlweise einen Interrupt auslösen, und zwar unter Kontrolle des im Interrupt Enable Register (IER) hinterlegten Bitmusters.

Den Timern ist gemeinsam, daß sie für die CPU definierte Zeit- und Verzögerungsintervalle erzeugen können. Ihre eigentliche Leistungsfähigkeit geht weit darüber hinaus. Timer T1 kann man etwa als Erzeuger von vielfältigen Rechteckimpulsen an Pin PB7 charakterisieren, T2 als Intervalltimer, Zähler und Kontrolleur für das serielle Schieberegister. Die Sonderfunktionen der Timer und der Pins PB7 und PB6 werden dabei vom Hilfsregister ACR kontrolliert.

Den Timern ist weiterhin die Zuordnung von Vorspeicher-Latches gemeinsam. T1 hat Latches für beide Bytes, T2 nur ein Latch für das niederwertige Byte.

Der Timer T1

Diesem Timer sind 4 Registeradressen zugeordnet, bei dem Schreiben und Lesen teilweise jedoch eine unterschiedliche Bedeutung haben (*Bild 6*).

Zunächst ist festzuhalten, daß man in den niedrigen Zählerteil nicht direkt hineinschreiben kann, sondern nur in seinen Vorspeicher T1L-L. Das Laden dieses Zählerteils aus dem Vorspeicher erfolgt erst in dem Augenblick, da man in das hohe Zählerbyte, Adresse A005, hineinschreibt. Damit beginnt der ‚count down' gleichzeitig an der gesamten 16-bit-Vorgabe. Nach Nulldurchgang des Timers wird sein Interrupt-Flag gesetzt. Alle hier besprochenen Funktionen des Timers 1 werden durch die Bitposition 7 und 6 im Hilfsregister ACR bestimmt.

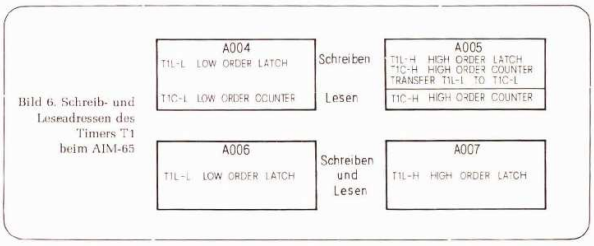


Bild 6. Schreib- und Leseadressen des Timers T1 beim AIM-65

Ein Beispiel für einfache Zeitverzögerung (one shot) von 160 µs, der Timer durchläuft nur einmal die Zählervorgabe. Interrupt wird durch Schreiben in das IER zugelassen:

```
LDA #$80    EINFACHES TIME OUT
STA ACR     HILFSREGISTER
LDA #$C0    INTERRUPTMÖGLICHKEIT
            FÜR T1
STA IER     INTERRUPT ENABLE
            REGISTER
LDA #$AO    TIMERVORGABE LOW,
            160 µSEC
STA T1L-L   INS LATCH
LDA #$00    VORGABE HIGH
STA T1C-H   STARTE TIMER DURCH
            SCHREIBEN IN DEN
            ZÄHLER (A005)
```

Nach seinem Nulldurchgang wird der Timer weiter mit dem Systemtakt 02 heruntergezählt, so daß man ggfs. die Zeit seit dem Nulldurchgang erfassen kann. Das Interrupt-Flag wird dadurch gelöscht, daß man den Timer in T1C-L (A004) liest oder in T1C-H (A005) neu beschreibt.

Neben diesem einfachen time-out gibt es weitere drei Betriebsweisen. Da ist zunächst ‚free running' zu erwähnen, Bit 6 im ACR ist gesetzt: Nach jedem time-out wird der Zähler automatisch mit dem Inhalt der Vorspeicher T1L-L und T1L-H (A006, A007) nachgeladen. Das Interrupt-Flag muß man dann aber per Programm zurücksetzen, indem man z. B. T1C-L liest. Damit sind wir auf eine weitere Besonderheit gestoßen: Der niedrige Zählerteil kann jederzeit gelesen werden, geladen wird er aber immer aus dem Vorspeicher-Latch. Der hohe Zähleranteil kann direkt gelesen und beschrieben werden, er wird automatisch aus dem Latch T1L-H nachgeladen, wenn mit ACR6 = ‚1' ‚free running mode' gesetzt ist. Und damit wird auch gleich der Nutzen der Vorspeicher ersichtlich:

Bei ‚free running' kann man den Timer irgendwann zwischenzeitlich mit der Zeitkonstanten für die nächste Phase vorladen, während er noch an der alten Phase herunterzählt. Nimmt man gar keine Veränderung der Werte in den Vorspeichern vor, so werden die Zeitabschnitte gleich lang, und es liegt eine Form der Selbstverwaltung im VIA vor. In nachfolgendem Beispiel nehmen wir einmal an, der Interruptvektor sei bereits geladen, und die Zeitabschnitte sollten alle gleich sein. Wir programmieren dann:

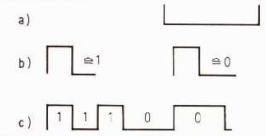


Bild 7. Impulserzeugung mit dem Timer T1 am Anschluß PB7: a) Einzel-Impuls (one-shot), b) Codierung von 1 oder 0 mit Startimpuls, c) Wechselphasencodierung

```
LDA #$40    FREE RUNNING
STA ACR     HILFSREGSITER
LDA #$XX    LADE NIEDRIGEN
            VORSPEICHER
STA T1L-L   INS LATCH
LDA #$YY    VORGABE FÜR DAS
            HÖHERE BYTE
STA T1C-H   STARTE TIMER UND LADE
            ZUGLEICH LATCH T1L-H
```

In der Interrupt-Routine setzen wir das Interrupt-Flag durch LDA T1C-L zurück.
Die beiden vorgenannten Betriebsweisen, die nur das Interrupt-Flag beeinflussen, werden ergänzt durch das ‚PB7-Enable' (Bit 7 im ACR = ‚1'). PB7 dient jetzt als Ausgang für Rechteckimpulse, die von T1 generiert werden (*Bild 7*).

Der Timer T2

Diesem Timer sind zwei Registeradressen zugeordnet, die wie beim T1 beim Lesen und Schreiben eine unterschiedliche Bedeutung für das niedrige Byte haben (*Bild 8*).

Bild 8. Schreib- und Leseadressen des Timers T2 beim AIM-65

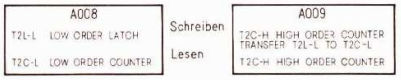
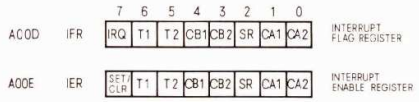


Bild 9. Funktionen des Interrupt-Flag- und des Interrupt-Enable-Registers; die Adressen gelten wieder für den AIM-65



Auch hier kann der untere Zählerteil nicht direkt beschrieben werden, er wird in dem Moment aus seinem Vorspeicher T2L-L geladen, wo man in das obere Zählerbyte T2C-H einschreibt. T2 arbeitet im ‚one shot mode' als Intervalltimer, wenn Bit 5 im ACR = ‚0'. Bei seinem Nulldurchgang setzt er einmalig sein Interrupt-Flag und zählt dann weiter von ‚FFFF' herunter. Das Interrupt-Flag wird durch Lesen von T2C-L oder neues Beschreiben von T2C-H gelöscht.

Setzt man Bit 5 im ACR auf „1", so fungiert T2 als „Count-Down-Zähler" für an PB6 angelegte abfallende Impulsflanken. Der Zähler ist mit einer Vorgabe zu laden. Das Interrupt-Flag erscheint, wenn die Vorgabe mit Nulldurchgang des Zählers verbraucht ist.

T2 kann unter gewissen Bedingungen auch als Taktgenerator für das Schieberegister SR eingesetzt werden (s. u.), dann ist allerdings nur der untere Zählerteil in Aktion.

Das Schieberegister

Es dient dem seriellen Datenverkehr über Pin CB2 als Ein- oder Ausgabe. Intern besteht eine Verkoppelung mit einem Modulo-8-Zähler. Auf dem VIA sind für die Eingabe drei und für die Ausgabe vier Betriebsarten vorgesehen. Bei der seriellen Datenübertragung kommt es wesentlich auf ein definiertes Timing zwischen Sender und Empfänger an. Pin CB1 übernimmt dabei neue Funktionen, die vom ACR gesteuert werden. Man kann dort den Takt einer externen Quelle anlegen (z. B. des Senders) oder einen intern erzeugten Takt für einen Empfänger über CB1 aussenden.

Als Takt an CB1 stehen zur Verfügung: a.) die Systemuhr 02, deren effektiver Takt mit 0,5 MHz arbeitet, b.) die Zeitvorgabe im niedrigen Teil des Timers 2, c.) z. B. Clockimpulse, die man im Timer 1 erzeugt und aus PB7 auf CB1 führt.

Hinsichtlich des Taktes bei serieller Datenübertragung beachte man in den Diagrammen des 6522-Datenblattes sehr genau, wann mit aufsteigender und wann mit abfallender Impulsflanke Daten übernommen werden. Im übrigen kann CB2 als Ausgabe für den Ruhezustand im PCR sowohl für ‚high' wie auf ‚low' programmiert werden.

Interruptkontrolle

In einigen Beispielen dieses Artikels wiesen wir schon auf die 3 Stufen des Interrupts hin. Den möglichen Interruptquellen des VIA (Steuerleitungen, Timer und Schieberegister) sind im Interrupt-Anzeigeregister IFR Flags zugeordnet. Sobald ein Flag auf ‚1' geht, wird dort auch Bit 7 logisch ODER auf ‚1' gesetzt. Das ermöglicht dem Prozessor mit dem BIT-Befehl eine schnelle Abfrage, ob ein Interrupt vom betreffenden Chip ausging. Wenn ja, dann kann in diesem IFR durch logische oder Verschiebebefehle die Interruptquelle im einzelnen schnell ausgemacht werden (*Bild 9*).

Eine hier angezeigte Interruptbedingung muß nicht notwendig zum Verlassen des im Vordergrund laufenden Maschinenprogramms führen. Welche Bedingungen – oder ob gar keine – Interrupt auslösen dürfen, wird durch Einschreiben in das IER, das Interrupt Enable Register festgelegt.

Literatur

65xx MICRO MAG, Heft 7/79: Das VIA 6522.
Rockwell International, Doc. No. 29650 N40: Versatile Interface Adapter (VIA).

Stichworte zum Inhalt

VIA 6522, Timer, Interrupt, Schieberegister, Handshaking, Interface, I/O, E/A.

der Operationscodes enthält, und zwar in der Reihenfolge ADL, ADH. Der Tabellenplatz ergibt sich durch Multiplizieren des Operationscodes mit 2; so steht etwa die Adresse des Unterprogramms für den Operationscode 04 an der Tabellenadresse 0008. Die Adressentabelle kann maximal 256 Byte, also eine „Page" umfassen, so daß die Operationscodes 00...7F möglich sind (das höchstwertige Bit dient ja der Trace-Betriebsart, s. o.). Bei der hier gewählten Speicherbelegung ist der Tabellenraum natürlich eingeschränkt, da in der „Zero Page" für das Interpreterprogramm und Daten schon einige Adressen belegt sind. In einem 6502-System mit erweitertem Speicher kann die Tabelle aber woanders hingelegt werden, ebenso braucht der Interpreter nicht in der Zero-Page stehen.

Ein Beispiel zur Adressentabelle: Will man dafür sorgen, daß beim Operationscode 04 ein Unterprogramm an der Adresse 1E5A angesprungen wird (beim KIM-1 kann man damit ein ASCII-Zeichen vom Terminal holen), so muß man in die Tabelle bei 0008 die Daten 5A und bei 0009 1E hineinschreiben.

Der Programmzähler

Wie jeder Mikroprozessor, so braucht auch unser „virtueller", simulierter Prozessor einen Programmzähler. Er wird von zwei aufeinanderfolgenden Zero-Page-Adressen dargestellt, nämlich 00EF und 00F0. Der Interpreter hat den Programmzähler bereits um 1 inkrementiert, wenn das Unterprogramm zur Abarbeitung des Operationscodes angesprungen wird. Bei Mehr-Byte-Befehlen kann so das dem Operationscode folgende Byte z. B. als Datenbyte ausgewertet werden. Das jeweilige Unterprogramm muß aber selbst dafür sorgen, daß – wenn der decodierte Befehl mehr als 1 Byte umfaßte – der Programmzähler zusätzlich inkrementiert wird. Bei 1-Byte-Befehlen braucht man sich um den Programmzähler nicht weiter zu kümmern.

```
0080 98       TYA         DEBUG-
0081 48       PHA         ROUTINE
0082 8A       TXA
0083 48       PHA         X UND Y
0084 A2 F0    LDX #F0     WERDEN
0086 8A       TXA         GERETTET
0087 48       PHA
0088 A5 EF    LDA EF      ANZEIGE
008A 85 FA    STA FA      VON ADR.
008C A5 F0    LDA F0      UND OP.-
008E 85 FB    STA FB      CODE
0090 20 19 1F JSR 1F19
0093 D0 FB    BNE 0090    HALT BEI
0095 68       PLA         GEDR.KIM-
0096 AA       TAX         TASTE
0097 CA       DEX
0098 D0 EC    BNE 0086    VERZOEG.-
009A 68       PLA         SCHLEIFE
009B AA       TAX         (ZEIT:
009C 68       PLA         0085)
009D A8       TAY
009E 60       RTS
```

Bild 2. Debugging-Routine für den KIM-1. Bei jedem Befehl wird während des Programmablaufes auf dem KIM-Display kurz die Adresse und der dort stehende Operationscode dargestellt

Stack-Verwendung

Um Platz in der Zero-Page und im RAM ab 0200 zu sparen, wurde ausgiebiger Gebrauch von der Möglichkeit gemacht, Zwischenwerte und Registerinhalte auf dem Stack zu retten. Das Interpreter-Programm selbst verändert weder den Akku noch das X-Register, das Y-Register und den Status. Dadurch ist es möglich, daß ein Operationscode dem nächsten Variablenwerte in einem der Register übergibt. Auch bedingte Sprungbefehle lassen sich leicht implementieren, da ja das Statusregister zwischen den einzelnen Operationscodes des Anwenderprogramms und damit zwischen den Interpreter-Unterprogrammen nicht verändert wird.

Jetzt wird's individuell

Außer dem Interpreter-Steuerprogramm können die Operationscodes des virtuellen Prozessors, die Unterprogramme für ihre Abarbeitung und damit auch die Unterprogramm-Adressentabelle individuell optimal an einen bestimmten Verwendungszweck angepaßt werden. Trotzdem sind einige Überlegungen hinsichtlich einer günstigen Verwendung der Operationscodes angebracht.

Zunächst einmal muß man sich über Sinn und Grenzen eines solchen Mikroprogramm-Interpreters klar werden. Die erreichbare Geschwindigkeit liegt deutlich unter der eines „nackten" Prozessors, da ja auch das Interpreter-Steuerprogramm Zeit benötigt. Hinsichtlich der Geschwindigkeit kann der virtuelle Prozessor daher umso eher mit dem Mutterprozessor konkurrieren, je länger die Unterprogramme für die einzelnen Operationscodes sind, je komfortabler also die Befehle der simulierten CPU sind. Dann ist nämlich der Zeitbedarf des Steuerprogramms im Vergleich zu dem der Unterprogramme relativ gering.

Es ist aber ohnehin nicht der Zweck des Interpreters, eine möglichst hohe Verarbeitungsgeschwindigkeit zu erzielen. Vielmehr soll die Programmierung bei Problemstellungen, die nicht allzu kritisch in bezug auf die Ausführungszeit der Befehle sind, durch möglichst komfortable Befehle vereinfacht werden. Wie das in der Praxis aussieht, sei hier zum Abschluß an einem einfachen Beispiel gezeigt. Es ist für den Betrieb des Mikrocomputers KIM-1 an einem ASCII-Terminal ausgelegt, eignet sich nach geringfügigen Änderungen aber auch z. B. für den SYM-1 oder AIM-65 bzw. PC-100.

```
0000 4F 1C    00=SPRUNG ZUM KIM
0002 A0 00    01=RESTART BEI 0200
0004 2F 1E    02=CRLF AUSGEBEN
0006 70 00    03=SPACE AUSGEBEN
0008 5A 1E    04=ASCII-CHR. HOLEN
000A 3B 1E    05=BYTE AUSGEBEN

0070 48       PHA         SPACE
0071 20 9E 1E JSR 1E9E    AUSGEBEN,
0074 68       PLA         AKKU
0075 60       RTS         RETTEN

0200 82 84 83 85 81  (DEBUG)
0200 02 04 03 05 01  (NORMAL)
```

Bild 3. Ein einfaches Anwendungsbeispiel. Mit nur wenigen 1-Byte-Befehlen läßt sich ein ASCII-Zeichen in sein Hex-Äquivalent umrechnen. Die Adressen der Monitor-Unterprogramme beziehen sich wieder auf den KIM-1