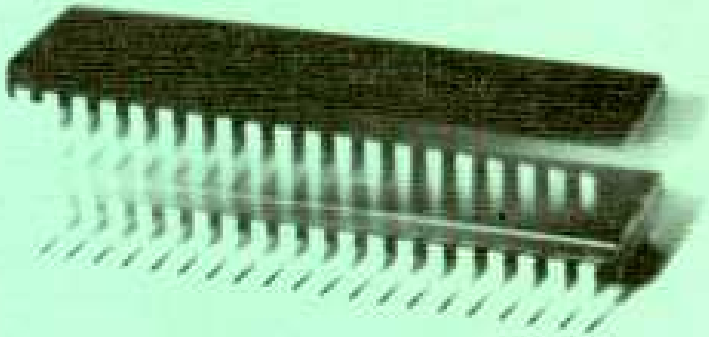
Der einfache Einstieg in die faszinierende Computertechnik

JUNIOR COMPUTER

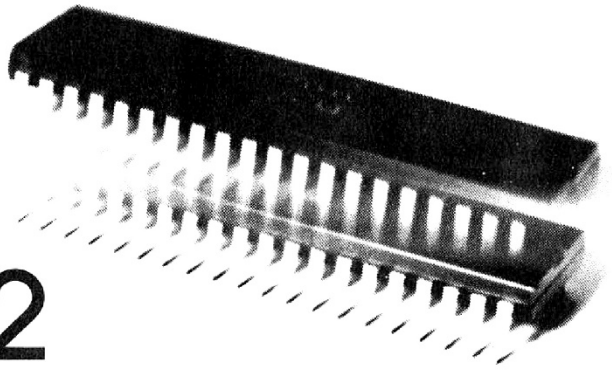


VIA 6522

Elektor Verlag

Junior-Computer

VIA 6522



ISBN 3-921608-34-1

Elektor Verlag GmbH, 5133 Gangelt 1

2. unveränderte Auflage November 1983

Vorwort

Die Mikroprozessor-Systeme, die man zur Zeit kaufen kann, sind kompakte und komplizierte Geräte; je professioneller sie sind, desto geringer ist die Chance, sie wirklich zu durchschauen. Um Daten übertragen zu können, braucht der Anwender eine Ein-/Ausgabe-Einheit, die aus einer oder mehreren Schnittstellen besteht. Diese vielseitigen PIAs, VIAs und RIOTs arbeiten im Prinzip alle gleich, obwohl sie für unterschiedliche Zwecke entwickelt wurden. Ihre Funktionsweise könnte man mit der einer Schleuse vergleichen, mit deren Hilfe ein Schiff von einem Wasserpegel auf einen höheren oder niedrigeren Pegel transportiert wird.

Dieses Buch beschäftigt sich ausschließlich mit dem sehr gängigen VIA 6522. Eigenschaften und Funktionsweise dieser "Schnittstelle" werden ausführlich beschrieben: Die Adressierung der Ports, der Einfluß von Lasten auf die logischen Pegel und die manuell oder automatisch durchgeführten, komplizierten Interrupt-Prozeduren, sowie die beiden bemerkenswerten Zeitgeber. Daher wendet sich dieses Buch nicht nur an die Anwender des Junior-Computers, sondern auch an die Besitzer eines Systems mit einem 6522. Sie lernen den VIA kennen und mit ihm umzugehen.

Dieses Buch von der Elektor-Redaktion verstehen wir als eine nützliche und notwendige Ergänzung zu den bereits veröffentlichten vier Junior-Büchern.

Inhalt

Das VIA 6522 ist ein Peripheriebaustein. VIA ist die Abkürzung von "Versatile Interface Adapter", was vielseitiger Interface-Adapter bedeutet. Dieser Interface-Adapter hat eine Anzahl von Registern an Bord, deren Funktion und Arbeitsweise wir nach und nach kennenlernen werden. Die Register des 6522 lassen sich zu folgenden Gruppen zusammenfassen:

- Zwei 8-bit-bidirektionale-Input/Output-Ports.
- Zwei programmierbare 16-bit-Timer bzw. Counter.
- Ein serielles Shift-In/Out-Register, das 8 bit breit ist.
- Vier Handshake-Leitungen, über die sich der Datenaustausch zwischen dem 6522 und den angeschlossenen Geräten leicht steuern läßt.
- Die Input/Output Daten lassen sich per Handshake in einem Latch abspeichern.

Der 6522 arbeitet mit einer Versorgungsspannung von 5 Volt und ist in einer 1-MHz- und einer 2-MHz-(A-Typ)Version erhältlich. Die I/O-Leitungen sind TTL-kompatibel. Port A ist darüber hinaus auch noch CMOS-kompatibel.
Das VIA 6522 ist ein sehr flexibler Input/Output-Baustein, der direkt an die Mikroprozessoren 65XX, 68XX oder an den 16-bit-Prozessor MC68000 angeschlossen werden kann. Dieser Interface-Baustein enthält eine Anzahl interner Register, wie zwei 16-bit-Timer, ein Schieberegister und zwei I/O-Ports. Mit den Timern lassen sich komplexe Pulsfolgen oder auch quarzgenaue Zeitintervalle für Zähler oder Uhrenanwendungen erzeugen. Mit dem Schieberegister lassen sich parallele Daten in serielle Daten umwandeln oder umgekehrt. Die Input/Output-Daten auf den Portleitungen lassen sich in einem Latch abspeichern. Das VIA 6522 ist außerdem vorzüglich für den Datenaustausch in einem Multiprozessorsystem geeignet.
Periphere Geräte, wie Drucker, Displays etc. werden in erster Linie von zwei 8 bit breiten, bidirektionalen Ports gesteuert. Jede Portleitung kann sowohl als Eingang, wie als Ausgang programmiert werden. Beim 6522 lassen sich sogar diverse I/O-Leitungen direkt vom Timer ansteuern. Somit lassen sich auf einfache Weise rechteckförmige Spannungen in Frequenz, Periodendauer und Tastverhältnis (Duty Cycle) programmieren. Ein Timer läßt sich aber auch über I/O-Leitungen ansteuern und somit als Zähler verwenden. Um die Programmierung des 6522-VIA zu vereinfachen, sind diverse Register auf dem Chip vorhanden: Ein Interrupt-Flag-Register,

ein Interrupt-Enable-Register und zwei Steuer-Register, mit denen sich diverse Steuerfunktionen programmieren lassen. Die Steuer-Register heißen Peripheral-Control-Register und Auxiliary-Control-Register. Bild 1 zeigt das Blockschaltbild des 6522. Die einzelnen Funktionsgruppen sind deutlich erkennbar.

1

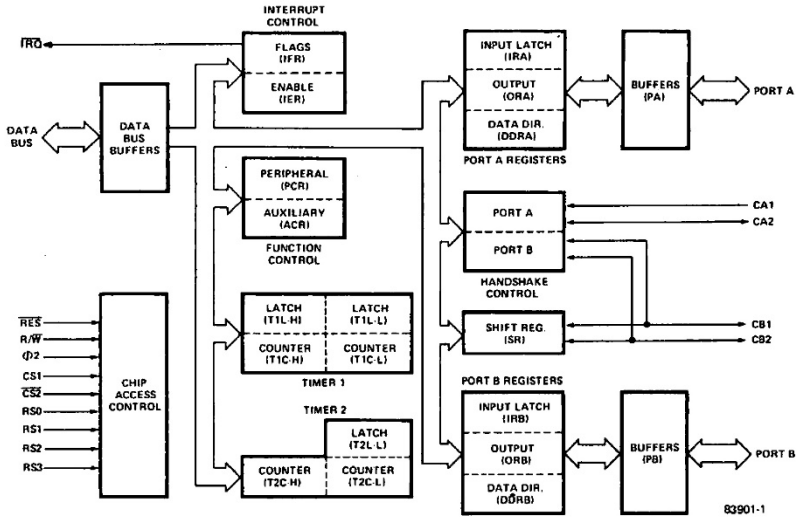


Bild 1. Das Blockschaltbild des VIA.

Pin-Beschreibung des 6522

Bild 2 zeigt die Anschlußbelegung des 6522. Dieses IC hat ein 40-Pin-Gehäuse. Alle Pins werden jetzt im einzelnen beschrieben.

2

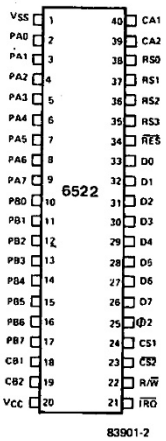


Bild 2. Die Pinkonfiguration des VIA.

RES (Reset)
Über den Reset-*Eingang* werden alle internen Register des 6522 auf Null gesetzt. Die Latches und Counters von Timer 1 und Timer 2 werden vom Reset-Eingang nicht beeinflußt. Die peripheren Interface-Leitungen werden über den Reset-Eingang in den Input-Status gesetzt und alle Timer, Handshake und Schieberegister-Interrupts werden verboten. Der 6522 kann also keinen Interrupt auslösen. Der Reset-Eingang ist bei logisch Null aktiv.

Φ2 (Takt-Eingang)
Der Takt-Eingang Φ2 ist an Φ2 von 65XX-Prozessoren oder an den Enable (E) von 68XX-Prozessoren angeschlossen. Auch der MC68000 hat eine Enable-Leitung. Somit kann der 6522 direkt an diesen 16-bit-Prozessor angeschlossen werden. Über den Φ2-Eingang des 6522 werden alle Datentransfers zwischen dem VIA und der CPU getriggert.

R/W (Read/Write-Eingang)
Die Richtung von Datentransfers zwischen dem 6522 und dem Systemprozessor wird über die R/W-Leitung gesteuert. Ist die R/W-Leitung logisch Eins, dann liest der Systemprozessor Daten aus einem selektierten 6522-Register. Ist die R/W-Leitung logisch Null, dann schreibt der Prozessor in ein selektiertes 6522-Register. Bevor in ein 6522-Register geschrieben oder aus einem Register Daten gelesen werden können, müssen über einen externen Adreßdekoder die Chip-Selekt-Leitungen aktiviert werden.

DBØ. . . DB7 (8 bidirektionale Datenbus-Leitungen)
Über die 8 bidirektionalen Datenbus-Leitungen erfolgt der Datenaustausch zwischen dem VIA und dem Systemprozessor. Während eines Read-Zyklus setzt der 6522 den Inhalt eines selektierten Registers auf die acht Datenbus-Leitungen. Der Registerinhalt wird dann in die CPU gelesen. Bei einem Schreib-Zyklus werden die acht Datenbus-Leitungen als hochohmige Eingänge geschaltet. Die Daten werden dann vom Prozessor über den Datenbus in ein internes VIA-Register übertragen. Wenn der 6522 nicht über die Chip-Selekt-Leitungen aktiviert ist, dann sind die acht Datenbus-Leitungen hochohmig (tri-state).

CS1, CS2 (Chip-Selekt-Leitungen)
Die beiden Chip-Selekt-Leitungen sind normalerweise über einen Adreßdekoder mit dem Adreßbus des Mikroprozessors verbunden. Das VIA wird aktiviert, wenn CS1 logisch Eins und CS2 logisch Null ist.

RSØ. . . RS3 (Register-Selekt-Leitungen)
Die vier Register-Selekt-Eingänge des 6522 sind normalerweise mit den Adreßleitungen AØ. . . A3 des Systemprozessors verbunden (A1 . . . A4 beim MC68000). Über diese Selekt-Leitungen kann der Systemprozessor eines von 16 VIA-Registern anwählen. Bild 3 zeigt, welches Bitmuster auf

den vier Register-Selekt-Leitungen liegen muß, um ein bestimmtes Register zu adressieren.

3 **6522-Register-File**

Junior Address	Register Number	RS Coding RS3	RS2	RS1	RS0	Register Desig.	Description Write	Read
1800	0	0	0	0	0	ORB/IRB	Output Register "B"	Input Register "B"
1801	1	0	0	0	1	ORA/IRA	Output Register "A"	Input Register "A"
1802	2	0	0	1	0	DDRB	Data Direction Register "B"	
1803	3	0	0	1	1	DDRA	Data Direction Register "A"	
1804	4	0	1	0	0	T1C-L	T1 Low-Order Latches	T1 Low-Order Counter
1805	5	0	1	0	1	T1C-H	T1 High-Order Counter	
1806	6	0	1	1	0	T1L-L	T1 Low-Order Latches	
1807	7	0	1	1	1	T1L-H	T1 High-Order Latches	
1808	8	1	0	0	0	T2C-L	T2 Low-Order Latches	T2 Low-Order Counter
1809	9	1	0	0	1	T2C-H	T2 High-Order Counter	
180A	10	1	0	1	0	SR	Shift Register	
180B	11	1	0	1	1	ACR	Auxiliary Control Register	
180C	12	1	1	0	0	PCR	Peripheral Control Register	
180D	13	1	1	0	1	IFR	Interrupt Flag Register	
180E	14	1	1	1	0	IER	Interrupt Enable Register	
180F	15	1	1	1	1	ORA/IRA	Same as Reg 1 Except No "Handshake"	

Bild 3. Die Register-File des 6522.

IRQ (Interrupt-Request-Ausgang)
Die Interrupt-Request-Leitung des VIA ist mit der IRQ-Leitung des Systemprozessors verbunden. Ist ein MC68000 der Systemprozessor, dann muß die IRQ-Leitung des 6522 über einen Prioritäts-Enkoder (z.B. 74LS348) angeschlossen werden.
Die Interrupt-Request-Leitung des VIA ist im Ruhezustand oder nach einem Reset logisch Eins. IRQ geht nach logisch Null, wenn im Interrupt-Flag-Register des VIA (Bild 30) eine Interrupt-Flag gesetzt ist und das korrespondierende Interrupt-Enable-Flag im Interrupt-Enable-Register gesetzt ist. Der IRQ-Ausgang des VIA ist ein offener Kollektorausgang, der mit den IRQ-Ausgängen anderer Peripheriebausteine zusammengeschaltet werden darf.

PA0. . . PA7 (Die Port-A-Leitungen des VIA: Eingänge/Ausgänge)
Port A des VIA besteht aus acht Leitungen, die sowohl Eingänge wie auch Ausgänge sein können. Ob die Port-Leitungen als Eingang oder als Ausgang arbeiten, bestimmt ein Bit im Data-Direction-Register (DDRA). Die Polarität einer Ausgangsleitung wird vom korrespondierenden Bit des Data-Output-Registers (ORA) bestimmt. Wenn der Prozessor auf eine Ausgangsleitung des Ports ein Bit schreibt, dann schreibt er nicht auf die Ausgangsleitung selbst, sondern in ein Flipflop. Somit bleibt die Information auch nach der Schreibeoperation auf der Port-Leitung erhalten. Der Prozessor kann aber auch das Bitmuster auf der Port-Leitung lesen, wenn die Port-Leitungen über das Data-Direction-Register zu Eingängen erklärt sind. Beim Lesen von Daten von den Port-Leitungen sind zwei Betriebsarten möglich:

1. Der Prozessor liest das momentane Bitmuster auf den Port-Leitungen (logisch Eins > 2,4 Volt; logisch Null < 2,4 V).
2. Das Bitmuster auf den Port-Leitungen wird auf eine Flanke auf der CA1-Leitung in einem internen Latch abgespeichert. Der Prozessor liest in dieser Betriebsart nicht direkt das Bitmuster auf den Port-Leitungen, sondern den Ausgang des Latches. Welche von beiden Modes angewählt ist, läßt sich über das periphere Control-Register programmieren (PCR). Doch später mehr darüber. Das Schaltbild der Port-A-Leitungen und der Handshake-Leitung CA2 zeigt Bild 4. Aus diesem Bild ist ersichtlich, daß die Leitungen von Port A über einen Widerstand (ca. 7 kΩ) nach 5 Volt gezogen werden.

4

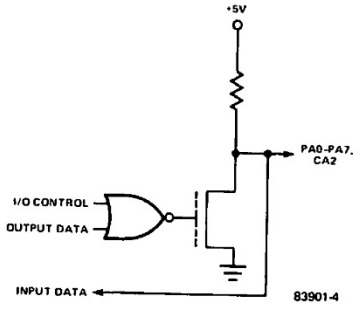


Bild 4. Das Schaltbild für eine Port-A-Leitung.

CA1, CA2 (Port-A-Control-Lines)

Die Leitungen CA1 und CA2 arbeiten entweder als Interrupt-Eingangsleitungen oder als Handshake-Ausgangsleitungen. Jede Leitung steuert ein internes Interrupt-Flag im Interrupt-Flag-Register. Eine Flanke an einer CA-Leitung setzt das entsprechende Interrupt-Flag. Ob ein gesetztes Interrupt-Flag einen Interrupt Request über die IRQ-Leitung des VIA auslösen darf (IRQ = logisch Null), entscheidet das korrespondierende Interrupt-Enable-Bit im Interrupt-Enable-Register (Bild 30). Die Leitung CA1 kann nur als Eingang arbeiten, dagegen kann die Leitung CA2 sowohl als Eingang wie auch als Ausgang programmiert werden. Im Output-Mode kann CA2 eine Standard-TTL-Last treiben.

PBØ. . . PB7 (Die Port-B-Leitungen des VIA: Eingänge/Ausgänge)

Die Leitungen PBØ. . . PB7 lassen sich wie die Port-A-Leitungen als Eingänge oder Ausgänge programmieren. Es gilt dasselbe, was für die Port-A-Leitungen gesagt wurde. Einige kleine Unterschiede sind aber dennoch zu beachten: Die Port-B-Leitungen sind als Eingänge nicht hochohmig, sondern stellen eine TTL-Last dar. Im Output-Mode können diese Leitungen eine TTL-Last treiben. Darüber hinaus kann jede Port-B-Leitung an eine Last einen Strom von 1 mA abgeben. Die Spannung an der Leitung fällt dann aber auf 1,5 Volt zusammen (Port-B-Leitung als Spannungs- oder Stromquelle). Somit können die Port-B-Leitungen direkt Darlington-Transistoren ansteuern. Auch das Schaltbild einer Port-B-Leitung weicht von dem einer Port-A-Leitung etwas ab. Bild 5 zeigt die Schaltung für die Port-B-Leitungen PBØ. . . PB7. Die Leitungen PB7 und PB6 arbeiten nicht

nur als Port-Leitungen, sondern können auch als Steuerleitungen für die internen Timer verwendet werden. An PB7 läßt sich durch Programmieren des Data-Direction-Registers (DDRB) und des Auxiliary-Control-Registers (ACR) als Timer-1-Ausgang schalten. In dieser Betriebsart lassen sich an PB7 komplexe, jitterfreie Pulsmuster unter Programmsteuerung erzeugen.

5

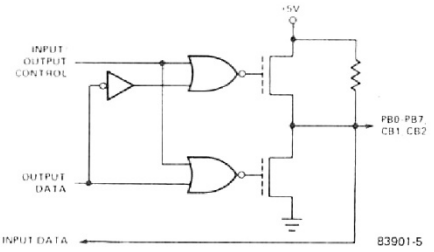


Bild 5. Das Schaltbild für eine Port-B-Leitung.

PB6 läßt sich durch Programmieren des Data-Direction-Registers (DDRB) und des Auxiliary-Control-Registers (ACR) als Eingang definieren und mit dem Timer 2 des 6522 verbinden. Timer 2 kann somit als Pulszähler verwendet werden. Später werden wir noch genau diese Betriebsarten von Timer 1 und Timer 2 erklären.

CB1, CB2 (Port-B-Control-Lines)

Die Leitungen CB1 und CB2 arbeiten wie die Leitungen CA1 und CA2 als Interrupt-Eingangsleitungen oder als Handshake-Ausgangsleitungen. CB1 und CB2 können nicht wie die Port-B-Leitungen PBØ . . . PB7 zur direkten Steuerung von Darlington-Transistoren verwendet werden. CB1 und CB2 stellen als Eingänge eine TTL-Last dar und können als Ausgänge eine TTL-Last treiben.
Darüber hinaus lassen sich die Leitungen CB1 und CB2 durch Programmierung des Auxiliary-Control-Registers mit dem Schieberegister des VIA verbinden. Über die Leitung CB2 läßt sich unter Programmsteuerung das 8 bit breite Wort im Schieberegister herausschieben. Es ist auch möglich, über CB2 ein externes, serielles Signal in das Schieberegister hineinzuschieben. Auf einfache Weise läßt sich somit eine Parallel/Seriell-Konversion oder eine Seriell/Parallel-Konversion von Daten durchführen.
Die Schiebeimpulse für das Schieberegister lassen sich auf verschiedene Weise erzeugen, wie später gezeigt wird. Wird das Taktsignal für das Schieberegister extern erzeugt, dann werden diese Pulse über CB1 dem Schieberegister zugeführt.

Arbeitsweise von Port A und Port B

Port A und Port B haben ein Data-Direction-Register DDRA und DDRB. Das Bitmuster im Data-Direction-Register bestimmt, ob eine Port-Leitung als Eingang oder Ausgang arbeitet. Eine Null in einer bestimmten Bitposition im Data-Direction-Register programmiert die korrespondierende Port-Leitung als Eingang. Eine Eins in einer bestimmten Bitposition im Data-Direction-Register programmiert die korrespondierende Port-Leitung als Ausgang.

Ist eine bestimmte Port-Leitung als Ausgang programmiert, dann arbeitet diese Port-Leitung mit dem Output-Register des VIA zusammen. Die Namen für die Output-Register für Port A und Port B sind ORA und ORB. Eine Eins in einer bestimmten Bitposition des Output-Registers steuert die korrespondierende Port-Leitung auf logisch Eins. Eine Null in einer bestimmten Bitposition des Output-Registers steuert die korrespondierende Port-Leitung auf logisch Null. Es dürfen auch Daten auf solche Bits im Output-Register geschrieben werden, die als Eingänge programmiert sind. In diesem Fall werden nur diejenigen Port-Leitungen beeinflußt, die Ausgänge sind, nicht jedoch die Eingangsleitungen. Beim Lesen von Port-Leitungen, die als Eingänge programmiert sind, sind zwei Betriebsarten möglich. Welche von beiden Betriebsarten gewählt ist, bestimmt das Bitmuster im Peripheral-Control-Register (PCR). Das Bitmuster im Peripheral-Control-Register wird durch eine Schreiboperation in dieses Register programmiert:

1. Der Prozessor liest eines der Input-Register (IRA oder IRB). Durch das Bitmuster im Peripheral-Control-Register ist "Input Latching Disabled". Das heißt, die Daten auf den zu Eingängen erklärten Port-Leitungen werden *nicht* durch eine Pulsflanke an CB1 oder CA1 in das Input-Register IRB oder IRA (siehe Bild 6) übernommen bzw. gespeichert. Die CPU liest in dieser Betriebsart direkt das Bitmuster auf den Port-Leitungen. Ist die Spannung auf der Port-Leitung > 2,4 Volt, dann liest die CPU eine logische Eins.
2. Der Prozessor liest wieder eines der Input-Register IRA oder IRB. Durch das Bitmuster im Peripheral-Control-Register ist "Input Latching Enabled". Das heißt, der Prozessor liest nicht mehr die Port-Leitungen selbst, sondern den Ausgang des Input-Registers. Der Ausgang des Input-Registers reflektiert das Bitmuster, das auf den Port-Leitungen zum Zeitpunkt der Pulsflanke an CA1 oder CB1 anlag.

6

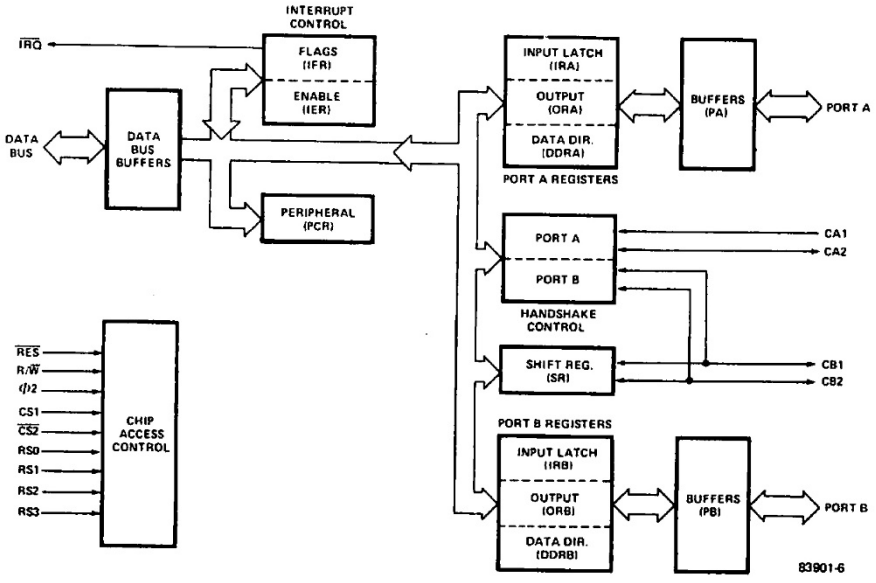
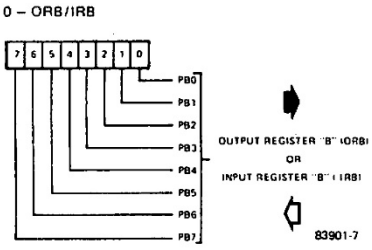


Bild 6. Port A, Port B und die Handshake-Leitungen des VIA.

Wie bereits erwähnt, lassen sich auch Port-Leitungen, die zu Ausgängen erklärt sind, vom Prozessor lesen. Wurde auf die Port-Leitung beispielsweise eine Eins geschrieben, dann wird der Prozessor bei einer Leseoperation eine Eins lesen. Dasselbe gilt, wenn eine Null auf eine Ausgangs-Port-Leitung geschrieben wurde.
Werden aber die Port-Leitungen, die als Ausgänge erklärt wurden, stark belastet, dann gibt es wieder Unterschiede zwischen Port A und Port B. Liest der Prozessor eine stark belastete Port-A-Ausgangsleitung, auf die zuvor eine logische Eins geschrieben wurde, dann bricht wegen der Last die Ausgangsspannung zusammen. Der Prozessor liest dann den momentanen Pegel auf der Port-Leitung. Obwohl eine Eins auf der Port-Leitung steht, liest der Prozessor eine Null von der belasteten Port-A-Leitung, falls die Spannung auf weniger als 2,4 Volt zusammenbricht.
Liest der Prozessor eine stark belastete Port-B-Ausgangsleitung, dann kann die Spannung auf dieser Leitung ebenfalls zusammenbrechen. Da die Pufferstruktur der Port-B-Leitungen von der der Struktur der Port-A-Leitungen abweicht, wird der Prozessor immer die richtigen Nullen oder Einsen von einer Port-B-Ausgangsleitung lesen. Dieser Unterschied ist zu berücksichtigen, wenn Maskierungsoperationen auf die Input-Register IRA und IRB bei belasteten Port-Leitungen ausgeführt werden.
Die Bilder 7 . . . 9 zeigen nochmals in einer kurzen Zusammenstellung, wie die Register IRA, IRB sowie ORA, ORB mit den beiden Data-Direction-Registern DDRA und DDRB zusammenarbeiten.

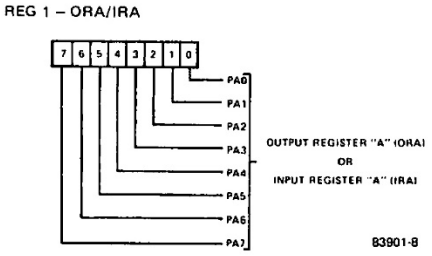
7



Output/Input Selektion	Write	Read
DDRB=1: korrespondierender PB-Pin ist Output.	Die CPU schreibt eine Null oder Eins auf den PB-Pin. (Write ORB)	Die CPU liest das Output-Register Bit in ORB. Die Spannung am Pin (Last!) hat keinen Einfluß.
DDRB=Ø: korrespondierender PB-Pin ist Input (Input Latching Disabled).	Die CPU schreibt ins Output-Register ORB; aber kein Einfluß auf den Input-Pin.	Die CPU liest die Eingangsspannung auf dem PB-Pin.
DDRB=Ø: korrespondierender PB-Pin ist Input (Input Latching Enabled).		CPU liest Bit im IRB. Dieses Bit ist der Zustand auf der PB-Leitung zum Zeitpunkt, als CB1 aktiv war (Flanke an CB1).

Bild 7. Die Steuerung des Port B.

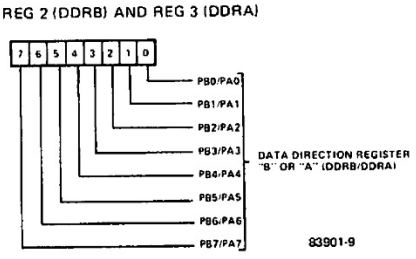
8



Output/Input Selektion	Write	Read
DDRA=1: korrespondierender PA-Pin ist Output (Input Latching Disabled).	Die CPU schreibt eine Null oder Eins auf den PA-Pin (Write ORA)	Die CPU liest das Spannungspotential am PA-Pin. Die Spannung am PA-Pin ist lastabhängig.
DDRA=1: Korrespondierender PA-Pin ist Output (Input Latching Enabled).		CPU liest Bit im IRA. Dieses Bit ist der Zustand auf der PA-Leitung zum Zeitpunkt, als CA1 aktiv war (Flanke an CA1).
DDRA=Ø: korrespondierender PA-Pin ist Input (Input Latching Disabled).	Die CPU schreibt ins Output Register ORB; aber kein Einfluß auf den Input-Pin.	Die CPU liest die Eingangsspannung auf dem PA-Pin.
DDRA=Ø: korrespondierender PA-Pin ist Input (Input Latching Enabled).		Die CPU liest Bit im IRA. Dieses Bit ist der Zustand auf der PA-Leitung zum Zeitpunkt, als CA1 aktiv war (Flanke an CA1).

Bild 8. Die Steuerung des Port A.

9



Eine Null an einer bestimmten Bitposition im Data-Direction-Register A oder B (DDRA, DDRB) erklärt die korrespondierende I/O-Leitung als Eingang.
Eine Eins an einer bestimmten Bitposition im Data-Direction-Register A oder B (DDRA, DDRB) erklärt die korrespondierende I/O-Leitung als Ausgang.

Bild 9. Das Data-Direction-Register von Port A und Port B. Mit diesen Registern läßt sich programmieren, ob eine Port-Leitung als Eingang oder Ausgang programmiert wird.

Die Handshake-Steuerung mit dem 6522-VIA

Über die Port-Leitungen Port A und Port B lassen sich an den VIA periphere Geräte wie ein Drucker, ein Kassettenrekorder oder ein zweites Computersystem anschließen. Da periphere Geräte in der Regel viel langsamer arbeiten als der Computer, an den sie angeschlossen sind, ist eine Handshake-Steuerung erforderlich. Mit Handshake können die angeschlossenen Geräte dem Computer mitteilen, ob sie bereit sind, über die Port-Leitungen Daten zu empfangen. Andererseits kann der Computer mit Handshake peripheren Geräten mitteilen, daß er Daten über die Port-Leitungen senden möchte. Für die Handshake-Steuerung sind beim 6522 die Handshake-Steuerleitungen CA1, CA2 für die Port-Leitungen von Port A und CB1, CB2 für die Port-Leitungen von Port B vorgesehen.
Die Handshake-Leitungen CA1 und CA2 lassen sich für die Handshake-Steuerung verwenden, wenn der Prozessor von den Port-A-Leitungen Daten liest oder auf die Port-Leitungen Daten schreibt. Etwas anders arbeiten die Handshake-Leitungen CB1 und CB2. Diese Leitungen können eine Handshake-Steuerung nur dann durchführen, wenn der Prozessor auf die Port-B-Leitungen Daten schreibt.

Read Handshake

Liest der Prozessor über die Port-Leitungen Daten von einem peripheren Gerät und führt diese Leseoperation unter Handshake-Steuerung durch, dann spricht man von "Read Handshaking". In dieser Betriebsweise setzt das periphere Gerät Daten auf die Port-Leitungen von Port A oder Port B und teilt dem Prozessor über eine der Handshake-Leitungen CA1 oder CA2 mit, daß gültige Daten auf den Port-Leitungen anliegen. Das Signal, das vom peripheren Gerät auf die Handshake-Leitungen gegeben wird, heißt "Data Ready"-Signal. Das Data-Ready-Signal setzt ein Flag im Interrupt-Flag-Register IFR des VIA (siehe Bild 30). Ist das korrespondierende Bit im Interrupt-Enable-Register des VIA gesetzt, dann geht die Interruptleitung des VIA auf logisch Null. Das Data-Ready-Signal kann somit eine Interruptsequenz im Computer hervorrufen. Ist aber das korrespondierende Bit im Interrupt-Enable-Register zurückgesetzt, dann kann das VIA keinen Interrupt auslösen, und der Prozessor muß durch Lesen des Interrupt-Flag-Registers feststellen, an welcher Handshake-Leitung ein Data-Ready-Signal von einem peripheren Gerät anliegt. Diese Betriebsweise nennt man "Polling".
Nachdem der Prozessor die Daten von den Port-Leitungen gelesen hat, teilt er dem peripheren Gerät über die zweite Handshake-Leitung mit, daß neue Daten auf die Port-Leitungen gesetzt werden können. Zusammenfassend läßt sich eine Read-Handshake-Steuerung wie folgt beschreiben:

* Das periphere Gerät setzt Daten auf die Port-Leitungen und teilt dem Prozessor über CA1 mit, daß gültige Daten auf den Port-Leitungen sind.
* Im Interrupt- oder Polling-Mode liest der Prozessor die Daten des angeschlossenen peripheren Gerätes von den Port-Leitungen. Nachdem die Port-Leitungen gelesen sind, kann das periphere Gerät neue Daten auf diese Leitungen geben. Dazu sendet der Prozessor über die CA2-Handshake-Leitung ein "Data Taken"-Steuersignal und teilt somit dem peripheren Gerät mit, daß die Port-Leitungen für neue Daten frei sind.

* Das periphere Gerät setzt neue Daten auf die Port-Leitungen und gibt ein Data-Ready-Signal auf die Handshake-Leitung CA1. Dieser Vorgang wiederholt sich, bis alle Daten vom peripheren Gerät in den Computer gelesen sind.

Über VIA-Port A ist sogar bei Read-Handshaking eine automatische Handshake-Steuerung möglich. Der CA1-Input-Pin übernimmt das Data-Ready-Signal vom abgeschlossenen peripheren Gerät und der CA2-Output sendet das Data-Taken-Signal. Das setzt in dieser Mode ein Flag im Interrupt-Flag-Register (IFR) (Bild 30). Dieses Flag kann einen Interrupt auslösen oder kann über Polling gelesen werden. Das Data-Taken-Signal an CA2 kann als Puls- oder Signal-Pegel programmiert werden. Ist das Data-Taken-Signal als Signal-Pegel programmiert (programmierbar mit dem Peripheral-Control-Register: PCR), dann wird es beim Eintreffen des nächsten Data-Ready-Signals automatisch zurückgesetzt. Das Zeitdiagramm für Read-Handshaking auf Port A zeigt Bild 10. Read-Handshaking auf Port B ist nicht möglich!

10

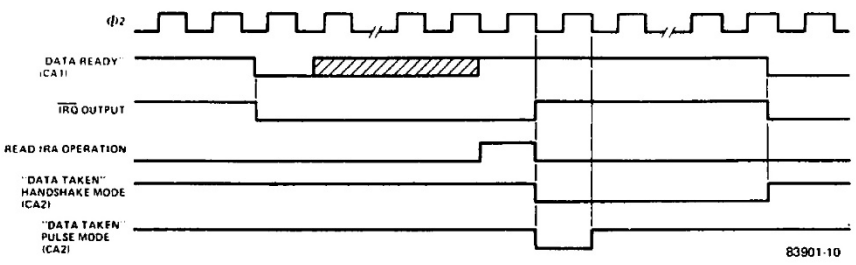


Bild 10. Read-Handshaking auf Port A. Diese Betriebsart ist auf Port B nicht möglich.

Write Handshake

Eine Write-Handshake-Sequenz ist der Read-Handshake-Sequenz sehr ähnlich. Der Unterschied besteht darin, daß das VIA unter Programmsteuerung der CPU das Data-Ready-Signal erzeugt und *nicht* das angeschlossene periphere Gerät. Hat das an die Port-Leitungen angeschlossene Gerät das Data-Ready-Signal empfangen, dann weiß es, daß gültige Daten auf den Port-Leitungen des VIA anliegen. Das periphere Gerät liest die Daten von den Port-Leitungen und antwortet mit einem Data-Taken-Signal. Write-Handshaking ist auf beiden Ports des 6522 möglich. CA2 oder CB2 arbeiten als Handshake-Ausgänge und erzeugen die Data-Ready-Signale für das periphere Gerät. Die Data-Ready-Signale lassen sich wieder als Puls oder Signal-Pegel programmieren (über das Peripheral-Control-Register: PCR).

Die Handshake-Leitungen CA1 und CB1 arbeiten bei Write-Handshaking als Eingänge und übernehmen vom peripheren Gerät das Data-Taken-Signal. Das Data-Taken-Signal des peripheren Gerätes setzt ein Flag im Interrupt-Flag-Register (IFR). Dieses Flag kann einen Interrupt auslösen oder kann im Polling-Mode vom System-Prozessor gelesen werden. Ist das Data-Ready-Signal an CA2 oder CB2 als Signal-Pegel programmiert, dann wird Data-Ready beim Eintreffen des nächsten Data-Taken-Signales auto-

matisch zurückgesetzt. Bild 11 zeigt das Zeitdiagramm für Write-Handshaking.

11

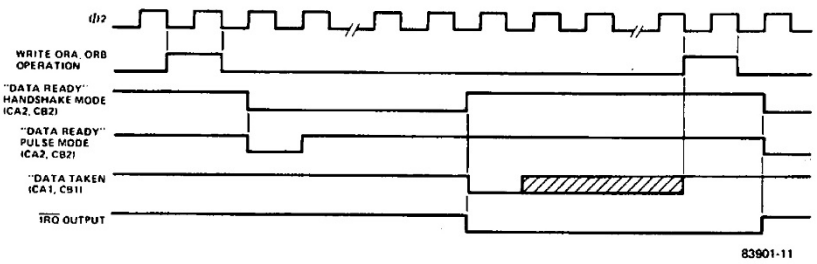


Bild 11. Write-Handshaking ist möglich auf Port A und Port B.

Wird Read/Write-Handshaking unter Interrupt-Steuerung vom Systemprozessor abgehandelt, dann muß am Ende der Interrupt-Routine die IRQ-Leitung des 6522 zurückgesetzt werden. Das entsprechende Interrupt-Flag läßt sich durch Schreiben einer Eins auf das Interrupt-Flag im Interrupt-Flag-Register zurücksetzen (siehe Beschreibung des IRF).

Das Peripheral-Control-Register (PCR)

Das VIA 6522 hat mehrere Möglichkeiten für eine Handshake-Steuerung. Ob Read- oder Write-Handshaking, die Art und Weise der Handshake-Steuerung wird immer über das Peripheral-Control-Register programmiert.

12

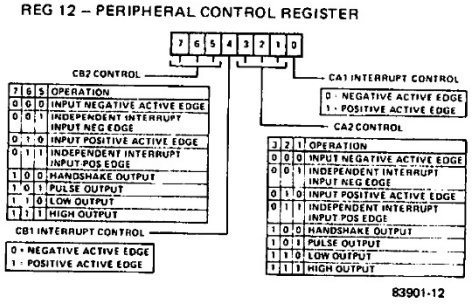


Bild 12. Das Peripheral-Control-Register.

Das Peripheral-Control-Register ist in zwei Hälften aufgeteilt (Bild 12). Mit den unteren vier Bits (Bit Ø . . . Bit 3) läßt sich die Funktionsweise der Handshake-Leitungen CA1 und CA2 programmieren. Ist Bit Ø logisch Null, dann reagiert CA1 auf negative Flanken; ist Bit Ø logisch Eins, dann reagiert CA1 auf positive Flanken. Mit Bit 1 . . . Bit 3 des PRC läßt sich die Funktionsweise von CA2 programmieren. Wenn Bit 3 Null ist, dann arbeitet CA2 als Eingang; ist Bit 3 Eins, dann arbeitet CA2 als Ausgang. Mit Bit 1 und Bit 2 lassen sich vier verschiedene Betriebszustände für CA2 als Input oder als Output programmieren. Bild 12 zeigt ausführlich, welches Bitmuster für welche Betriebsart zuständig ist.
Die oberen vier Bits des Peripheral-Control-Registers steuern die Funktions-

weise der Handshake-Leitungen CB1 und CB2. Bit 4 ist für die Steuerung von CB1 zuständig, während Bit 5 . . . Bit 7 für die Steuerung von CB2 zuständig sind. Für Bit 4 . . . Bit 7 gilt dasselbe, was zuvor über Bit 1 . . . Bit 3 gesagt wurde.

Timer 1

Der Intervall-Timer 1 besteht aus zwei 8-bit-Latches und einem 16-bit-Counter-Register. Die beiden Latches werden als Zwischenspeicher für die Daten verwendet, die in das 16-bit-Counter-Register geschrieben werden. Ist das Counter-Register von Timer 1 geladen, dann zählt der Timer mit Φ2-Taktsignalfrequenz rückwärts. Timer 1 zählt rückwärts, bis sein Inhalt Null ist. Das Timer 1 Interrupt-Flag wird in diesem Moment gesetzt.

13

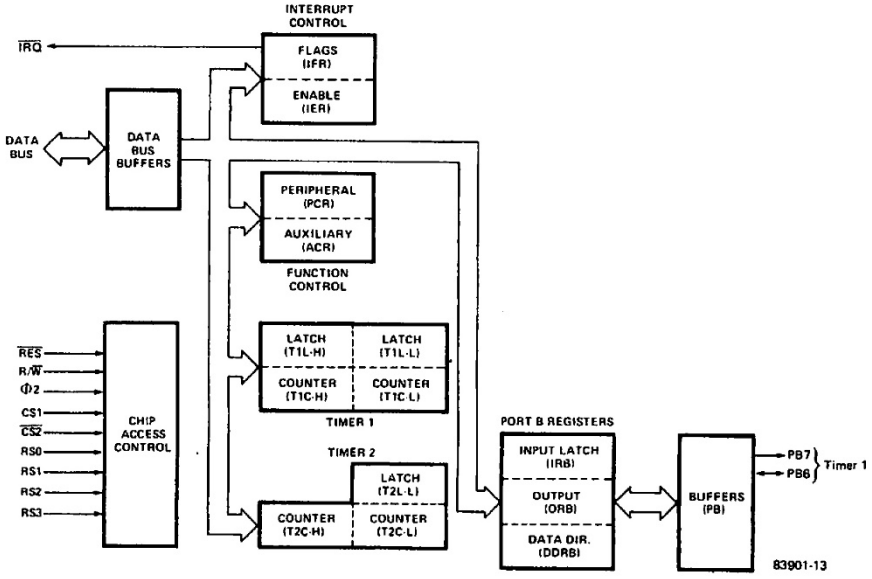
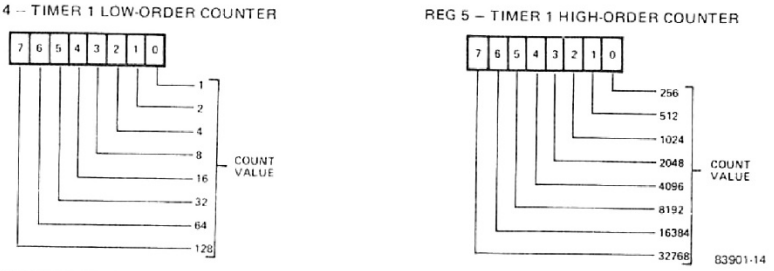


Bild 13. Die Timer im 6522-VIA.

Beim nächsten Φ2-Puls ist der Inhalt von Timer 1 minus Eins. Ist das korrespondierende Interrupt-Enable-Flag im Interrupt-Enable-Register gesetzt, dann geht die Leitung IRQ nach logisch Null und die CPU kann eine Interrupt-Sequenz einleiten. Ist der Timer einmal abgelaufen, dann löst er keine weiteren Interrupts mehr aus, da die IRQ-Leitung logisch Null bleibt. Die Arbeitsweise von Timer 1 läßt sich über das Auxiliary-Control-Register (ACR, siehe Bild 16) programmieren. Ist es dem Timer 1 über das Auxiliary-Control-Register erlaubt, sich selbst wieder zu starten, dann werden automatisch nach jedem Time-Out (der Inhalt des Counter-Registers von Timer 1 ist Null) die Daten in den beiden 8-bit-Latches in das 16-bit-Counter-Register übernommen. Timer 1 beginnt dann wieder rückwärts zu zählen, bis der nächste Time-Out erreicht ist. Es ist auch möglich, die Arbeitsweise von Timer 1 über das Auxiliary-Control-Register so zu programmieren, daß die Spannung an PB7-Pin bei jedem Time-Out invertiert wird. Am PB7-Pin entsteht somit eine Rechteckspannung, deren Periode

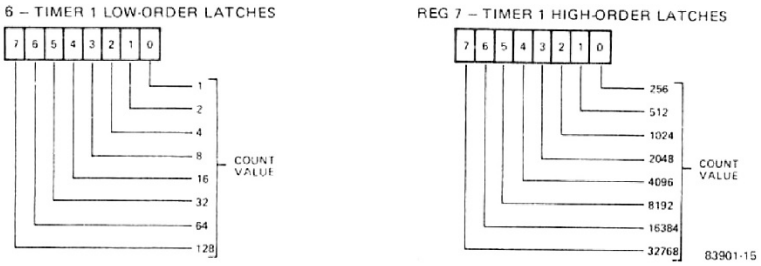
14



Write: Schreibe 8 Bits in Timer-1-Low-Order-Latch. Der Inhalt des Latch wird in den Counter kopiert, sobald der High-Order-Counter (REG 5) geladen wird.	**Write:** Schreibe 8 Bits in Timer-1-High-Order-Counter und Timer-1-High-Order-Latch. Bei dieser Schreiboperation werden Timer-1-Low-Order-Latch und Timer 1-High-Order-Latch in die beiden Counter-Register von Timer 1 übertragen. Der Count-Down von Timer 1 beginnt und das Timer-1-Interrupt-Flag wird zurückgesetzt.
Read: Die CPU liest Timer-1-Low-Order-Counter. Das Timer-1-Interrupt-Flag wird zurückgesetzt.	Die CPU liest Timer-1-High-Order-Counter. Das Timer-1-Interrupt-Flag wird **nicht** zurückgesetzt.
Durch Lesen der Counter-Register von Timer 1 kann der Computer feststellen, wieviel Zeit bis zum "Time-Out" (Counter-Register = FFFF = –1; Timer-1-Interrupt-Flag zurückgesetzt) zur Verfügung steht, oder wieviel Zeit seit dem "Time-Out" vergangen ist (Counter-Register = FFFF = –1; Timer-1-Interrupt-Flag gesetzt).	

Bild 14. Lese- und Schreibeoperationen auf die Counter-Register von Timer 1.

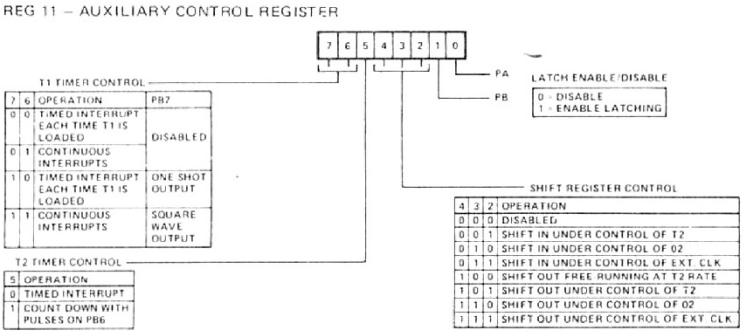
15



Write: Schreibe 8 Bits in Timer-1-Low-Order-Latch. Kein Unterschied zwischen Write REG 4 und Write REG 6	**Write:** Schreibe 8 Bits in Timer-1-High-Order-Latch. Der Inhalt des Latch wird **nicht** nach Timer-1-High-Order-Counter übertragen. Auch Timer-1-Low-Order-Latch wird **nicht** nach Timer-1-Low-Order-Counter übertragen. Der Timer wird nicht gestartet und das Timer-1-Interrupt-Flag wird **nicht** beeinflußt.
Read: Die CPU liest Timer-1-Low-Order-Latch. Das Timer-1-Interrupt-Flag wird durch diese Leseoperation **nicht** beeinflußt.	Die CPU liest Timer-1-High-Order-Latch. Timer-1-Countdown und das Timer-1-Interrupt-Flag werden durch diese Leseoperation nicht beeinflußt.

Bild 15. Lese- und Schreiboperationen auf den Latch-Registern von Timer 1.

16



Erklärungen:

Bit Ø und Bit 1	Erlaube/Verbiete das Zwischenspeichern von Daten auf den Port-A- und Port-B-Leitungen. Ist das Zwischenspeichern von Daten auf den Port-Leitungen erlaubt, dann werden die Daten durch eine Flanke an CA1 für Port A und durch eine Flanke an CB1 für Port B in den Input-Latch der Ports übernommen.
Bit 4, 3, 2	Steuerung des Schieberegisters im VIA.
Ø Ø Ø	Schalte das Schieberegister aus.
Ø Ø 1	Schiebe Daten in das Schieberegister. Timer 2 steuert das Schieberegister im VIA (Timer-2-Takt).
Ø 1 Ø	Schiebe Daten in das Schieberegister. Systemtakt Ø2 oder Enable (E) steuert das Schieberegister. Shift-In-Baudrate = Systemtakt.
Ø 1 1	Schiebe Daten in das Schieberegister. Das Schieberegister wird von einem externen Taktgenerator (z.B. 555) gesteuert.
1 Ø Ø	Schiebe Daten aus dem Schieberegister. Timer 2 steuert das Schieberegister im VIA (Timer-2-Free-Running-Mode).
1 Ø 1	Schiebe Daten aus dem Schieberegister. Timer 2 steuert das Schieberegister im VIA (Timer-2-Takt).
1 1 Ø	Schiebe Daten aus den Schieberegister. Systemtakt Ø2 oder Enable (E) steuert das Schieberegister. Shift-Out-Baudrate = Systemtakt.
1 1 1	Schiebe Daten aus dem Schieberegister. Das Schieberegister wird von einem externen Taktgenerator (z.B. 555) gesteuert.
Bit 5	**Steuerung der Arbeitsweise von Timer 2**
Ø	Erzeuge einen Interrupt, sobald der Timer 2 abgelaufen ist.
1	Zähle rückwärts mit Impulsen auf der Portleitung PB6.
Bit 7, 6	**Steuerung der Arbeitsweise von Timer 1**
Ø Ø	Erzeuge einen Interrupt, sobald der Timer 1 abgelaufen ist. Die Portleitung PB7 wird **nicht** beeinflußt.
Ø 1	Der Timer lädt sich selbst nach jedem Time-Out und erzeugt somit kontinuierlich Interrupts. Die Port-Leitung PB7 wird **nicht** beeinflußt.
1 Ø	Erzeuge einen Interrupt, sobald der Timer 1 abgelaufen ist. Die Port-Leitung PB7 arbeitet als "One-Shot"- oder Monoflop-Ausgang.
1 1	Der Timer 1 lädt sich selbst nach jedem Time-Out und erzeugt somit kontinuierliche Interrupts. An der Port-Leitung PB7 entsteht eine rechteckförmige Spannung. Timer 2 läßt sich als "freilaufender Oszillator" verwenden.

Bild 16. Das Auxiliary-Control-Register im VIA.

vom Inhalt der beiden Timer-1-Latches und der Systemfrequenz Φ2 bestimmt wird.
Die Register von Timer 1 werden in Bild 14 und Bild 15 gezeigt. Auch das Lesen und Schreiben aus den/in die Timer-1-Register(n) ist in diesen Bildern erklärt. Bild 16 zeigt das Auxiliary-Control-Register und wie sich die Arbeitsweise von Timer 1 über Bit 7 und Bit 6 programmieren läßt. Vier unterschiedliche Betriebsarten sind somit für Timer 1 möglich. Bei Schreibeoperationen in die Timer-1-Register ist folgende Eigenart von Timer 1 zu beachten:
Der Prozessor kann nicht direkt in den Low-Order-Counter von Timer 1 schreiben. Stattdessen schreibt er immer in Timer-1-Low-Order-Latch. Der Low-Order-Counter von Timer 1 wird immer mit dem Inhalt von Low-Order-Latch geladen, sobald der Prozessor in High-Order-Counter von Timer 1 Daten schreibt; auch der Timer-Countdown wird dadurch gestartet. Bleibt während eines Programmes der Inhalt von Timer-1-Low-Order-Latch unverändert, dann muß für einen neuen Timerstart nur das Register Timer-1-High-Order-Counter neu beschrieben werden, nicht jedoch Timer-1-Low-Order-Latch oder Low-Order-Counter.

Timer-1-One-Shot-Mode
Ist der Timer 1 im One-Shot-Mode programmiert, dann erzeugt er einen Interrupt nach jedem Time-Out. Der Timer muß natürlich zuvor geladen worden sein, um einen Time-Out erzeugen zu können (Time-Out: Der Inhalt des Counters ist minus Eins). Darüber hinaus läßt sich die Arbeitsweise von Timer 1 über das Auxiliary-Control-Register so programmieren, daß für die Zeitdauer des Countdown ein negativer Impuls an PB7 erzeugt wird.
Um einen einzelnen Interrupt mit Timer 1 zu erzeugen, müssen Bit 7 und Bit 6 im Auxiliary-Control-Register Null sein. Dann muß der Prozessor entweder in das Register Timer 1-Low-Order-Counter oder Timer 1-Low-Order-Latch ein Offset schreiben. Ist diese Schreibeoperation erledigt, dann schreibt der Prozessor in das VIA-Register Timer 1-High-Order-Counter ein weiteres Offset. Dieses High-Order-Offset wird in das Register Timer-1-High-Order-Latch übertragen und startet den Countdown. Der Countdown startet bei der nächsten negativen Φ2-Flanke, die der Schreiboperation folgt. Timer 1 zählt jetzt mit der Φ2-Frequenz rückwärts bis zum Time-Out. Ist schließlich in Counter-Register Timer 1 der Wert Null erreicht, dann wird das Timer-1-Flag im Interrupt-Flag-Register gesetzt.
Soll PB7 während des Countdown logisch Null sein, dann muß im Auxiliary-Control-Register Bit 7 logisch Eins sein. PB7 wird logisch Null, sobald ein Offset in Timer-1-High-Order-Counter geschrieben wird. Hat schließlich Timer 1 auf Null zurückgezählt, dann wird PB7 automatisch auf logisch Eins zurückgesetzt.
Die Timer-1-Interrupt-Flag im Interrupt-Flag-Register läßt sich auf zwei Arten zurücksetzen:

1. Der Prozessor schreibt ein Offset in das Register Timer-1-High-Order-Counter. Ein neuer Countdown wird durch diese Schreiboperation gestartet.
2. Der Prozessor liest das Register Timer-1-High-Order-Counter. Der Countdown wird von dieser Leseoperation nicht beeinflußt.

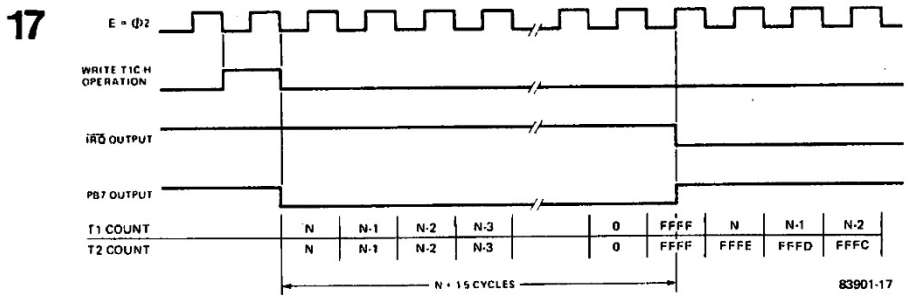


Bild 17. So arbeitet Timer 1 im One-Shot-Mode oder als "Software Monoflop".

Das Zeitdiagramm für Timer 1 One-Shot-Mode zeigt Bild 17. Die Leitung PB7 bleibt für N + 1,5 Zyklen logisch Null. N ist eine Hexzahl im Bereich 0000 . . . FFFF. Deutlich ist aus diesem Diagramm ersichtlich, daß die Leitung $\overline{IRQ}$ zu dem Zeitpunkt logisch Null (aktiv) wird, wenn PB7 wieder auf Eins zurückgesetzt wird.

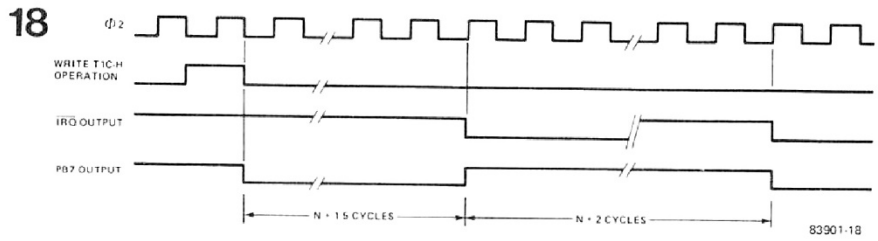
Timer-1-Free-Run-Mode

Der Hauptvorteil von Timer 1 ist, daß ein kontinuierliches, jitterfreies Rechtecksignal erzeugt werden kann. Dieses Rechtecksignal steht am Port B (Pin PB7) zur Verfügung. Die Periodendauer des Rechtecksignals wird auch nicht von Prozessor-Leseoperationen oder sonstigen Zeiten, wie zum Beispiel der Zeit, die vergeht, bis der Prozessor auf einen Timer-Interrupt antworten kann, beeinflußt. Daß Timer 1 ein jitterfreies Rechtecksignal erzeugen kann, liegt an den High-Order- und Low-Order-Latches, von denen die Counter-Register nach einem Time-Out automatisch nachgeladen werden.
Arbeitet Timer 1 im Free-Running-Mode, dann wird das Timer-1-Interrupt-Flag gesetzt und das Signal an PB7 invertiert, wenn Timer 1 auf Null zurückgezählt hat. Im Gegensatz zum One-Shot-Mode zählt Timer 1 nicht nur mit der Frequenz Φ2 nach unten, sondern kopiert den Inhalt des Low-Order-Latch und des High-Order-Latch in das 16 bit breite Counter-Register. Nach dem ersten Time-Out von Timer 1 bleibt das Interrupt-Flag im Interrupt-Enable-Register gesetzt. Ein Rücksetzen des Timer-1-Interrupt-Flag kann auf mehrere Arten geschehen:

1. Der Prozessor schreibt eine Eins auf das gesetzte Timer-1-Flag im Interrupt-Flag-Register (wird später erklärt).
2. Der Prozessor liest das Register Timer-1-Low-Order-Counter.
3. Der Prozessor schreibt einen neuen Wert in das Register Timer-1-High-Order-Counter.

Beide Intervall-Timer des VIA lassen sich aufs neue setzen, bevor ein Time-Out erreicht ist. Eine Schreiboperation in die Timer-Register vor dem Time-Out wird immer wieder den Time-Out-Zeutpunkt "in die Länge ziehen". Timer 1 wird auf diese Art arbeiten, wenn der Prozessor in das Register Timer-1-High-Order-Counter schreibt. Schreibt der Prozessor aber nur in die Latches von Timer 1, so kann der Prozessor während des Countdowns auf den Timer 1 zugreifen, ohne jedoch den gerade laufen-

den Countdown zu beeinflussen. Die Daten, die während eines Countdown in die Latches von Timer 1 geschrieben werden, beeinflussen erst den folgenden Countdown!



Timer-1-Free-Running-Mode

Achtung! Wenn die Leitung PB7 als Timer 1-Output verwendet wird, dann müssen Bit 7 im DDRB **und** Bit 7 im Auxiliary-Control-Register logisch Eins sein. PB7 muß für diese Betriebsart als Ausgang erklärt sein.

Bild 18. Timer-1-Free-Running-Mode wird benötigt, wenn Timer 1 kontinuierlich eine rechteckförmige Spannung mit einer programmierbaren Frequenz erzeugen soll.

Setzt der Prozessor unter Programmsteuerung nach jedem Time-Out das Timer-1-Interrupt-Flag zurück, so kann der Timer 1 dieses Flag beim nächsten Time-Out wieder setzen. Mit einer Polling- oder Interrupt-Sequenz kann somit der Prozessor nach jedem Time-Out die Latches von Timer 1 aufs Neue beschreiben. Der Duty-Cycle (Tastverhältnis) der rechteckförmigen Spannung an PB7 läßt sich somit beliebig variieren. Komplexe rechteckförmige Spannungen lassen sich mit Timer 1 auf einfache Weise an PB7 unter Programmsteuerung erzeugen. Frequenzen von Null bis ca. 250 kHz lassen sich bei einem 1-MHz-Φ2-Signal quarzgenau erzeugen! Bild 18 zeigt das Zeitdiagramm für Timer 1 im Free-Running-Mode.

Timer 2

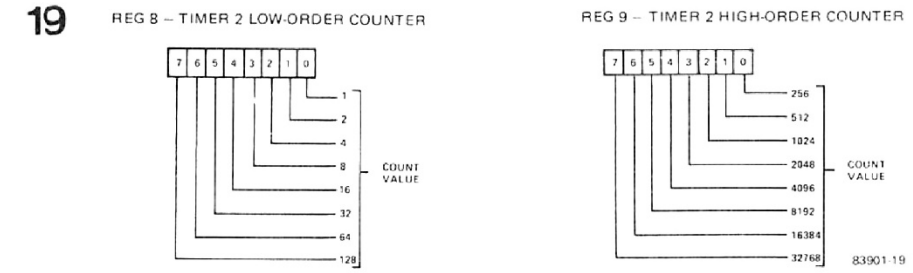
Timer 2 unterscheidet sich von Timer 1 dadurch, daß kein High-Order-Latch vorhanden ist. Deshalb ist ein Free-Running-Mode nicht möglich. Zwei Betriebsarten lassen sich über das Auxiliary-Control-Register für Timer 2 programmieren:

1. Intervall-Timer im One-Shot-Mode
2. Rückwärtszähler für negative Impulse am PB6-Eingang.

Auch die Lese- und Schreiboperationen der Timer-2-Register unterscheiden sich von Timer 1:

* Timer 2-Low-Order-Latch ist ein Register, in das nur Daten geschrieben werden können (Write-only).
* Timer 2-Low-Order-Counter ist ein Register, aus dem nur Daten gelesen werden können (Read-only).
* Timer 2-High-Order-Counter ist ein Register, in das Daten geschrieben werden können, oder dessen Inhalt der Prozessor lesen kann (Read/Write).

Das Counter-Register von Timer 2 ist 16 bit breit und zählt mit der Φ2-Frequenz rückwärts.
Bild 19 zeigt ausführlich, was geschieht, wenn die Register des Timer 2 gelesen oder mit Daten beschrieben werden.



Write: Schreibe 8 Bits in Timer-2-Low-Order-Latch.

Write: Schreibe 8 Bits in Timer-2-High-Order-Counter. Bei dieser Schreiboperation wird Timer-2-Low-Order-Latch in Timer-2-Low-Order-Counter kopiert. Der Count-Down von Timer 2 wird gestartet und das Timer-2-Interrupt-Flag wird zürückgesetzt.

Read: Die CPU liest Timer-2-Low-Order-Counter. Das Timer-2-Interrupt-Flag wird zurückgesetzt.

Read: Die CPU liest Timer-2-High-Order-Counter. Das Timer-2-Interrupt-Flag wird **nicht** zurückgesetzt.

Bild 19. Lese- und Schreiboperationen auf den Counter-Registern von Timer 2.

Timer-2-One-Shot-Mode

Im One-Shot-Mode arbeitet Timer 2 ähnlich wie Timer 1. In dieser Betriebsart wird das Timer-2-Interrupt-Flag bei jedem Time-Out gesetzt. Ist das korrespondierende Flag im Interrupt-Enable-Register gesetzt, dann geht die IRQ-Leitung des VIA auf logisch Null, und die CPU kann die Interruptverarbeitung einleiten. Ist aber das korrespondierende Interrupt-Enable-Flag zurückgesetzt, dann kann der Prozessor über Polling das Interrupt-Flag von Timer 2 abfragen.
Bei den folgenden Pulsen nach dem Time-Out (Time Out = 0000) beginnt der Timer 2 weiterhin rückwärts zu zählen und sein Inhalt ist FFFF, FFFE, FFFD . . . Der Computer kann somit durch Lesen der Register Timer-2-Low-Order-Counter und Timer-2-High-Order-Counter feststellen, wieviel Zeit seit dem Time-Out verstrichen ist. Wie bei Timer 1 läßt sich auch bei Timer 2 der Zeitpunkt des Time-Out beliebig verlängern oder verkürzen. Der Prozessor muß dazu während einer Countdown-Periode Timer-2-Low-Order-Latch und Timer-2-High-Order-Counter mit neuen Daten beschreiben. Bei diesen Lese/Schreibeoperationen ist zu beachten, daß das Timer-2-Interrupt-Flag durch Lesen des Registers Timer-2-Low-Order-Latch und durch Schreiben in das Register Timer-2-High-Order-Counter zurückgesetzt wird. Das Zeitdiagramm für Timer 2 im One-Shot-Mode zeigt Bild 20.

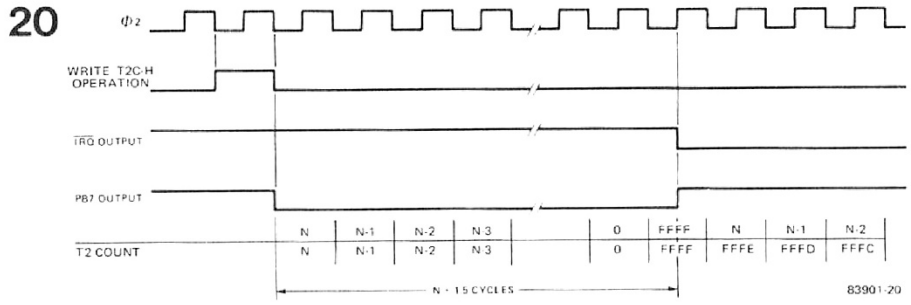


Bild 20. So arbeitet Timer 2 im One-Shot-Mode oder als "Software Monoflop".

Timer-2-Pulse-Counting-Mode

Timer 2 kann auch über das Auxiliary-Control-Register so programmiert werden, daß er im Pulse-Counting-Mode arbeitet. In dieser Betriebsart zählt Timer 2 eine bestimmte Anzahl negativer Pulse am PB6-Pin. Dazu muß zuerst eine 16-bit-Zahl in die Timer 2-Register geladen werden. Dazu müssen Daten in Timer-2-Low-Order-Latch und Timer-2-High-Order-Counter geschrieben werden. Die Schreiboperation in Timer-2-High-Order-Counter initialisiert den Countdown von Timer 2. Bei jedem negativen Puls auf der Leitung PB6 dekrementiert Timer 2 um Eins. Ist der Inhalt des Counter-Registers 0000, dann wird das Timer-2-Interrupt-Flag gesetzt. Ist das korrespondierende Interrupt-Enable-Flag gesetzt, dann geht die IRQ-Leitung des VIA auf logisch Null und der Prozessor kann eine Interrupt-Sequenz einleiten. Soll ein neuer Countdown mit negativen Pulsen an PB6 initialisiert werden, dann muß der Prozessor-Timer zwei aufs neue laden. Auch das Interrupt-Flag wird dadurch zurückgesetzt. Bild 21 zeigt das Zeitdiagramm für Timer 2 im Pulse-Counting-Mode.

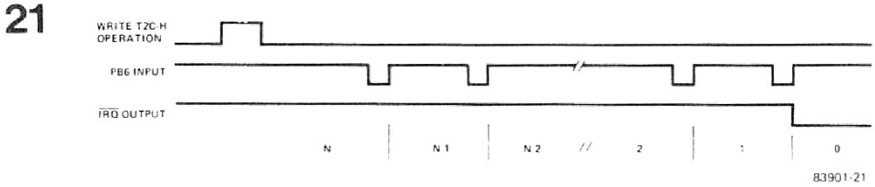


Bild 21. Timer-2-Puls-Counting-Mode. In dieser Betriebsart läßt sich Timer 2 als Impulszähler verwenden.

Ist das Timer-2-Interrupt-Flag zurückgesetzt und liest der Prozessor das Timer-2-Counter-Register, dann kann festgestellt werden, wieviele Pulse an PB6 noch nötig sind, bis das Counter-Register von Timer 2 Null ist. Ist das Timer-2-Interrupt-Flag gesetzt und liest der Prozessor das Timer 2-Counter-Register, dann kann festgestellt werden, wieviele Pulse an PB6 eingetroffen sind, bevor Timer 2 auf Null gezählt hat. Hat Timer 2 einmal auf Null gezählt, dann ist die Zählfolge bei jedem weiteren negativen Puls an PB6: 0000, FFFF, FFFE, FFFD . . .

Das Schieberegister im VIA

Das Schieberegister ermöglicht einen seriellen Datentransfer in das VIA

und aus dem VIA. Die seriellen Daten werden über den CB2-Pin empfangen oder gesendet. Das Taktsignal für das Schieberegister läßt sich intern oder extern erzeugen. Wird das Taktsignal für das Schieberegister extern erzeugt, dann muß es über den CB1-Pin dem Schieberegister im VIA zugeführt werden. Es ist auch möglich, daß Timer 2 die Taktsignale für das Schieberegister erzeugt. In dieser Betriebsart lassen sich die intern erzeugten Taktsignale am Pin CB1 abnehmen. Weitere externe Schaltungen lassen sich somit direkt mit dem Taktsignal des VIA-Schieberegisters steuern. Die diversen Betriebsarten des Schieberegisters lassen sich über das Auxiliary-Control-Register programmieren. Bild 16 zeigt, wie sich mit dem Auxiliary-Control-Register die diversen Betriebsarten anwählen lassen.

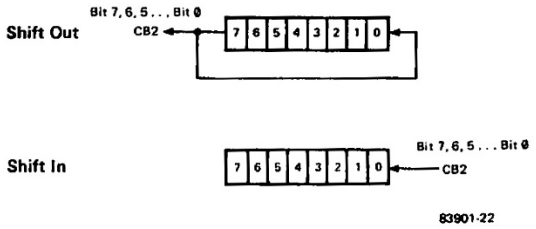


Bild 22. So werden die Daten in das Schieberegister und aus dem Schieberegister geschoben.

Aus Bild 22 geht hervor, in welcher Reihenfolge die einzelnen Bits aus dem Schieberegister geschoben werden und in welcher Reihenfolge das Schieberegister die einzelnen Bits des seriellen Datenstromes empfängt. Werden Daten aus dem Schieberegister geschoben, dann werden die Bits in der Reihenfolge Bit 7, Bit 6, Bit 5 . . . Bit Ø über die Leitung CB2 übertragen. Gleichzeitig wird das zuletzt herausgeschobene Bit in Bit Ø des Schieberegisters abgelegt. Werden Bits über CB2 in das Schieberegister geschoben, dann wird das Schieberegister der Reihenfolge Bit Ø, Bit 1, Bit 2 . . . Bit 7 aufgefüllt. Das Schieberegister überträgt beliebige 8-bit-Worte ohne Start- und Stoppbits, wie das bei der Übertragung von ASCII-Zeichen üblich ist. Die Start- und Stoppbits lassen sich jedoch über ein kleines Programm leicht hinzufügen.
Das Schieberegister hat 8 Betriebsarten:
Betriebsart Null: ACR Bit 4,3,2 = ØØØ
In dieser Betriebsart ist das Schieberegister abgeschaltet. Der Prozessor kann in das Schieberegister schreiben oder das Schieberegister lesen. Das Interrupt-Flag des Schieberegisters ist immer zurückgesetzt. Die Handshake-Leitungen CB1 und CB2 stehen unter der Steuerung des Peripheral-Control-Registers.
Betriebsart Eins: ACR Bit 4,3,2 = ØØ1 (Shift In)
Die Daten werden unter Steuerung des Timer 2 in das Schieberegister hineingeschoben. Es werden nur die niederwertigen acht Bits von Timer-2-Low-Order-Latch und Timer-2-Low-Order-Counter verwendet. Das Taktsignal für das Schieberegister kann am CB1-Pin abgenommen werden.
Die Schiebeoperation wird durch Schreiben in das Schieberegister oder Lesen aus dem Schieberegister initialisiert. Nach acht Schiebepulsen ist das Schieberegister voll, und das Interrupt-Flag des Schiebregisters wird gesetzt.

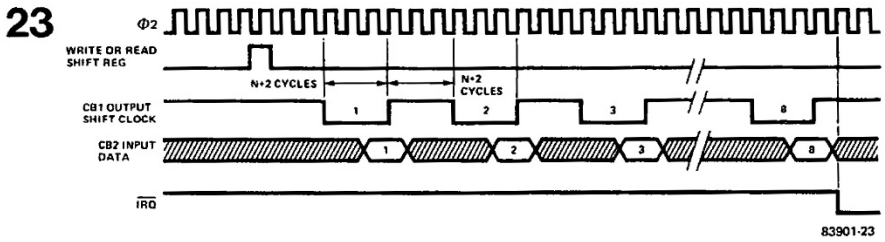


Bild 23. Schieberegister in Betriebsart Eins. Die Daten werden seriell in das Schieberegister unter Steuerung Timer 2 geschoben.

Die Frequenz des Taktsignales für das Schieberegister hängt vom Taktsignal Φ2 und vom Inhalt des Low-Order-Latch von Timer 2 ab. Bild 23 zeigt, wie sich die "Baudrate" berechnet: N = Zahl im Low-Order-Latch.
Betriebsart Zwei: ACR Bit 4,3,2 = Ø1Ø (Shift In)
Die Daten werden unter Steuerung des Systemtaktes Φ2 in das Schieberegister hineingeschoben. An CB1 liegen wieder die Schiebepulse für das Schieberegister an. Timer 2 arbeitet als unabhängiger Intervall-Timer und hat keinen Einfluß auf das Schieberegister.
Die Schiebeoperation wird durch Schreiben in das Schieberegister oder Lesen aus dem Schieberegister initialisiert. Nach acht Φ2-Pulsen ist das Schieberegister leer, und das Interrupt-Flag des Schieberegisters wird gesetzt. Auch die Schiebepulse am CB1-Pin werden gesperrt. Bild 24 zeigt diese Betriebsart.

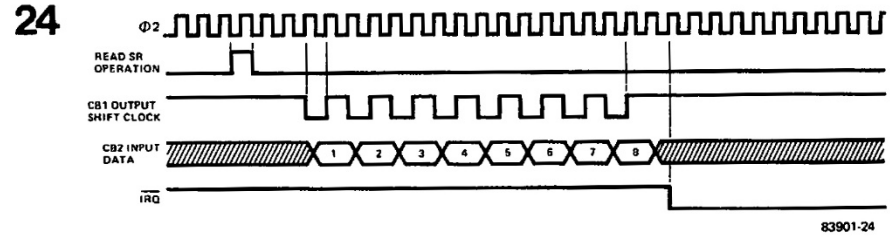


Bild 24. Schieberegister in Betriebsart Zwei: Die Daten werden seriell in das Schieberegister unter Steuerung des Systemtaktes Φ2 geschoben.

Betriebsart Drei: ACR Bit 4,3,2 = Ø11 (Shift In)
Die Daten werden unter Steuerung eines externen Taktsignales in das Schieberegister hineingeschoben. Dem CB1-Pin wird das externe Taktsignal für das Schieberegister zugeführt.

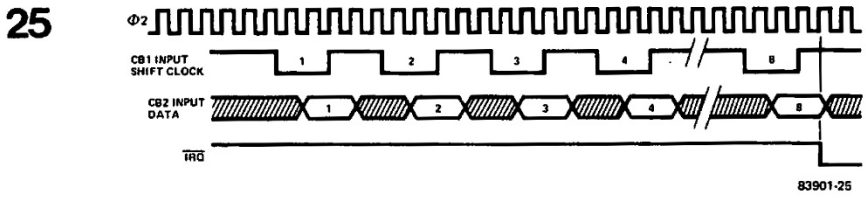


Bild 25. Schieberegister in Betriebsart Drei: Die Daten werden seriell in das Schieberegister unter Steuerung eines externen Taktsignales geschoben.

Die Schiebeoperation wird durch Schreiben in das Schieberegister oder Lesen aus dem Schieberegister initialisiert. Nach acht Schiebepulsen ist das Schieberegister voll und das Interrupt-Flag des Schieberegisters wird gesetzt. Eine Lese- oder Schreiboperation auf dem Schieberegister setzt das Interrupt-Flag wieder zurück. Bild 25 zeigt das Zeitdiagramm für diese Betriebsart.

Betriebsart Vier: ACR Bit 4,3,2 = 1ØØ (Shift Out)

In dieser Betriebsart wird der Inhalt des Schieberegisters unter Steuerung von Timer 2 über den CB2-Pin herausgeschoben. Wie Bild 26 zeigt, ist die Schiebegeschwindigkeit vom Inhalt des Low-Order-Latch von Timer 2 abhängig: N = Zahl in Low-Order-Latch von Timer 2.

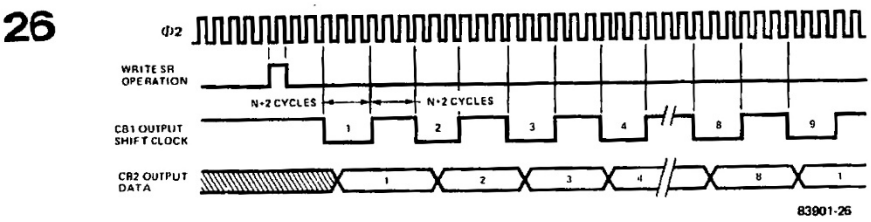


Bild 26. Schieberegister in Betriebsart Vier: Die Daten werden seriell aus dem Schieberegister unter Steuerung von Timer 2 geschoben.

Die Schiebepulse stehen am CB1-Pin zur Steuerung externer Logik zur Verfügung. Sind alle acht Bits des Schieberegisters herausgeschoben, dann wird der Inhalt des Schieberegisters kontinuierlich herausgeschoben. Das Bitmuster im Schieberegister geht nicht verloren, da beim Herausschieben von Daten Bit 7 in Bit Ø abgelegt wird. In diesem Schieberegister-Free-Running-Mode wird niemals das Interrupt-Flag des Schieberegisters gesetzt oder zurückgesetzt.

Betriebsart Fünf: ACR Bit 4,3,2 = 1Ø1 (Shift Out)

In dieser Betriebsart wird der Inhalt des Schieberegisters unter Steuerung von Timer 2 über den CB2-Pin herausgeschoben. Wie Bild 27 zeigt, ist die Schiebegeschwindigkeit vom Inhalt des Low-Order-Latch von Timer 2 abhängig: N = Zahl im Low-Order-Latch von Timer 2.

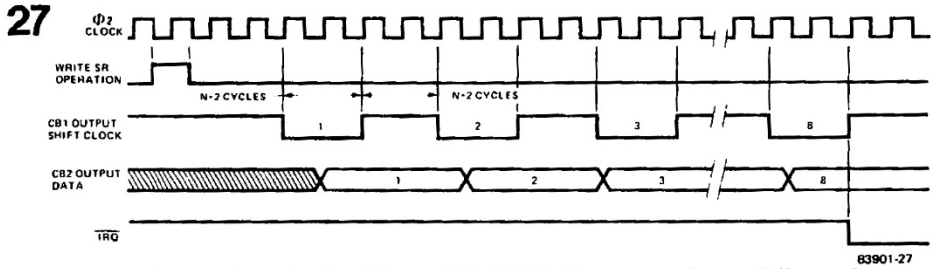


Bild 27. Schieberegister in Betriebsart Fünf: Die Daten werden seriell aus dem Schieberegister unter Steuerung von Timer 2 geschoben. Sind acht Bits aus dem Schieberegister geschoben, dann wird der Schiebevorgang automatisch beendet.

Die Schiebepulse stehen am CB1-Pin zur Steuerung der externen Logik zur Verfügung. Sind alle 8 Bits aus dem Schieberegister herausgeschoben, dann

hört der Schiebevorgang auf, und das Interrupt-Flag des Schieberegisters wird gesetzt. Am CB2-Pin liegt der Datenpegel des letzten Datenbits (Null oder Eins) an.

Betriebsart Sechs: ACR Bit 4,3,2 = 11Ø (Shift Out)

In dieser Betriebsart wird der Inhalt des Schieberegisters unter Steuerung des Systemtaktes Φ2 über den CB2-Pin herausgeschoben. Die Schiebepulse stehen am CB1-Pin zur Steuerung externer Logik zur Verfügung. Sind alle acht Bits aus dem Schieberegister herausgeschoben, dann hört der Schiebevorgang auf, und das Interrupt-Flag des Schieberegisters wird gesetzt. Am CB2-Pin liegt der Datenpegel des letzten Datenbits (Null oder Eins) an (Bild 28).

28

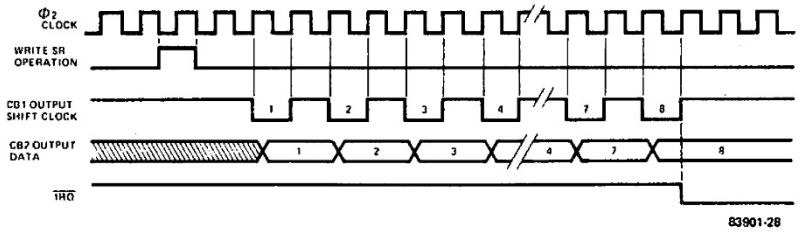


Bild 28. Schieberegister in Betriebsart Sechs: Die Daten werden seriell aus dem Schieberegister unter Steuerung des Systemtaktes Φ2 geschoben.

Betriebsart Sieben: ACR Bit 4,3,2 = 111 (Shift Out)

In dieser Betriebsart wird der Inhalt des Schieberegisters unter Steuerung eines externen Taktsignales über den CB2-Pin herausgeschoben. Das externe Taktsignal für das Schieberegister wird über den CB1-Pin zugeführt. Sind acht Bits aus dem Schieberegister herausgeschoben, dann wird das Interrupt-Flag des Schieberegisters gesetzt. Der Schiebevorgang wird jedoch nicht beendet, sondern setzt sich bei jedem externen Taktsignal fort. Eine Schreib- oder Leseoperation auf dem Schieberegister setzt das Interrupt-Flag des Schieberegisters zurück. Nach weiteren acht Schiebeimpulsen wird das Interrupt-Flag wieder gesetzt. Immer wenn das Interrupt-Flag gesetzt ist, kann der Prozessor das Schieberegister aufs neue laden. Bild 29 zeigt hierfür das Zeitdiagramm.

29

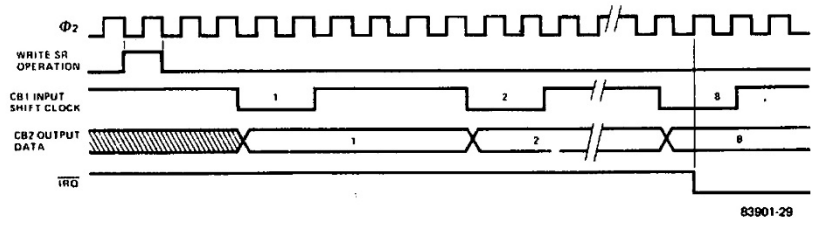


Bild 29. Schieberegister in Betriebsart Sieben: Die Daten werden seriell aus dem Schieberegister unter Steuerung des externen Taktsignales geschoben.

Interrupt-Flag- und Interrupt-Enable-Register

Bild 30 zeigt das Interrupt-Flag- und das Interrupt-Enable-Register. Das

Interrupt-Flag-Register kann der Prozessor lesen oder auch beschreiben. Jedes Flag im Interrupt-Flag-Register kann der Prozessor zurücksetzen, wenn er eine logische Eins "auf das zurückzusetzende Flag" schreibt.
Bit Ø ist das CA2-Flag. Dieses Flag wird gesetzt, wenn CA2 über das Peripheral-Control-Register als Eingang erklärt ist und eine positive oder negative Flanke am CA2-Eingang entsteht. Ob das CA2-Flag auf die positive oder negative Flanke gesetzt wird, entscheidet die Programmierung des Peripheral-Control-Registers. Bit Ø im Interrupt-Enable-Register ist das korrespondierende Interrupt-Enable-Bit. Ist dieses Bit gesetzt, dann geht die IRQ-Leitung des VIA auf logisch Null, wenn das CA2-Flag gesetzt wird. Das CA2-Flag läßt sich zurücksetzen, wenn der Prozessor das Output-Register A liest oder beschreibt. Auch die IRQ-Leitung wird zurückgesetzt, wenn kein weiteres Flag im Interrupt-Flag-Register gesetzt ist. Ist CA2 als "Independent Interrupt Input" (siehe Bild 12) erklärt, dann wird eine Lese- oder Schreiboperation auf das Output-Register A das CA2-Interrupt-Flag nicht zurücksetzen. Das CA2-Interrupt-Flag kann in dieser Betriebsart zurückgesetzt werden, wenn der Prozessor eine Eins auf das CA2-Flag im Interrupt-Flag-Register schreibt.

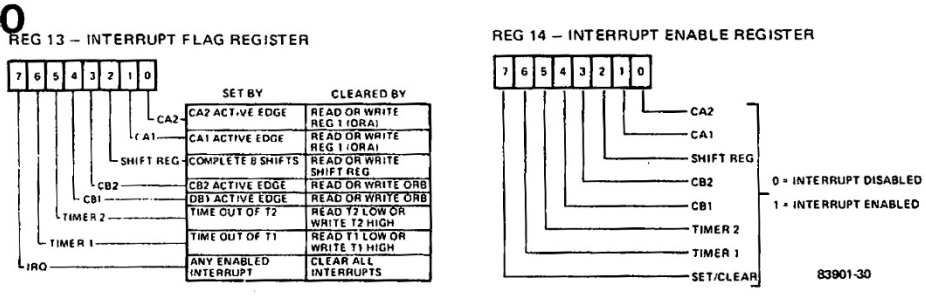


Bild 30. Das Interrupt-Flag und das Interrupt-Enable-Register des VIA. Jedes gesetzte Interrupt-Flag im Interrupt-Flag-Register kann die IRQ-Leitung des VIA aktivieren. Ob ein gesetztes Interrupt-Flag die IRQ-Leitung aktivieren darf oder nicht, entscheidet das korrespondierende Interrupt-Enable-Flag im Interrupt-Enable-Register.

Bit 1 ist das CA1-Flag. Dieses Flag wird gesetzt, wenn CA1 über das Peripheral-Control-Register als Eingang erklärt ist und eine positive oder negative Flanke am CA1-Eingang entsteht. Ob das CA1-Flag auf eine positive oder negative Flanke gesetzt wird, entscheidet die Programmierung des Peripheral-Control-Registers (siehe Bild 12). Bit 1 im Interrupt-Enable-Register ist das korrespondierende Interrupt-Enable-Bit. Ist dieses Bit gesetzt, dann geht die IRQ-Leitung des VIA auf logisch Null, wenn das CA1-Flag gesetzt wird. Das CA1-Flag wird immer zurückgesetzt, wenn der Prozessor das Output-Register A liest oder beschreibt. Auch die IRQ-Leitung wird zurückgesetzt, wenn kein weiteres Flag im Interrupt-Enable-Register gesetzt ist.
Bit 2 ist das Schieberegister-Flag. Dieses Flag wird gesetzt, wenn alle acht Bits aus dem Schieberegister herausgeschoben sind und das Schieberegister nicht im "Free-Running Mode" arbeitet. Das Schieberegister-Flag wird zurückgesetzt, wenn der Prozessor das Schieberegister liest oder mit Daten beschreibt.

Bit 3 und Bit 4 sind das CB2- und CB1-Flag. Diese beiden Flags werden gesetzt und zurückgesetzt, wie bei den Flags CA1 und CA2 beschrieben.
Bit 5 ist das Timer-2-Flag. Dieses Flag wird gesetzt, wenn Timer 2 einen Time-Out hat. Liest der Prozessor nach einem Time-Out Timer-2-Low-Order-Counter oder schreibt der Prozessor in Timer-2-High-Order-Counter, dann wird das Timer-2-Interrupt-Flag zurückgesetzt. Auch die IRQ-Leitung wird zurückgesetzt, wenn kein weiteres Bit im Interrupt-Flag-Register gesetzt ist. Bit 5 im Interrupt-Enable-Register ist das korrespondierende Interrupt-Enable-Bit. Ist dieses Bit gesetzt, dann geht die IRQ-Leitung des VIA auf logisch Null, wenn das Timer-2-Flag gesetzt wird.
Bit 6 ist das Timer-1-Flag. Dieses Flag wird gesetzt und zurückgesetzt, wie bei Bit 5 (Timer-2-Flag) beschrieben. Bit 6 im Interrupt-Enable-Register ist das korrespondierende Interrupt-Enable-Bit für Timer 1.
Bit 7 ist kein Interrupt-Flag! Bit 7 im Interrupt-Flag-Register zeigt an, ob die IRQ-Leitung des VIA logisch Null (= aktiv) ist. Ist die IRQ-Leitung logisch Null, dann ist Bit 7 logisch Eins. Mit Boolescher Algebra läßt sich Bit 7 so definieren:

B7 = IFR6 x IER6 + IFR5 x IER5 + IFR4 x IER4 + IFR3 x IER3 + IFR2 x IER2 + IFR1 x IER1 + IFRØ x IERØ.

Dabei ist x = logisch AND und + = logisch OR.
Bit 7 im Interrupt-Flag-Register läßt sich nicht zurücksetzen, wenn eine Eins auf dieses Bit geschrieben wird. Bit 7 kann nur zurückgesetzt werden (und auch die IRQ-Leitung des VIA), wenn Bit 6 . . . Bit Ø im Interrupt-Flag-Register zurückgesetzt werden *oder* alle Bits im Interrupt-Enable-Register Null sind.
Bit 7 im Interrupt-Enable-Register hat eine besondere Steuerfunktion. Schreibt der Prozessor auf Bit 7 eine Null, dann werden Bit 6 . . . Bit Ø im Interrupt-Enable-Register zurückgesetzt. Jedes gesetzte Flag im Interrupt-Flag-Register aktiviert die IRQ-Leitung des VIA. Liest der Prozessor das Interrupt-Enable-Register, dann ist Bit 7 immer Eins, und die übrigen Flags zeigen an, ob sie gesetzt oder zurückgesetzt sind.

Beliebige Bits im Interrupt-Enable-Register lassen sich setzen, wenn der Prozessor auf das zu setzende Bit und auf Bit 7 im Interrupt-Enable-Register eine Eins schreibt.

Zusammenfassung für das Interrupt-Flag- und Interrupt-Enable-Register:
Die Bits im Interrupt-Flag-Register signalisieren, ob an den CA1-, CA2- und CB1- und CB2-Eingängen eine aktive Flanke eingetroffen ist, ob einer der beiden Intervall-Timer oder beide Timer zusammen einen Time-Out hatten, oder ob das Schieberegister acht Bits herausgeschoben hat. Die Flags im Interrupt-Enable-Register entscheiden, ob ein aktives Flag im Interrupt-Flag-Register über die Interrupt-Leitung des VIA einen Interrupt auslösen darf oder nicht. Sind mehrere Flags im Interrupt-Flag-Register gleichzeitig gesetzt, dann kann der Prozessor das Interrupt-Flag-Register lesen. Ein Programm kann die Priorität der einzelnen Interrupt-Flags festsetzen und die einzelnen Interrupts abhandeln. Es ist jedoch auch möglich, alle Interrupts zu verbieten. Die einzelnen Interrupt-Flags im Interrupt-Flag-Register werden in dieser Betriebsart wie beschrieben gesetzt; die IRQ-Leitung des VIA bleibt jedoch immer logisch Eins (inaktiv). Der

Prozessor kann im Polling-Mode das Interrupt-Flag-Register lesen und dann die einzelnen Interrupt-Flags "bearbeiten". Auch hier kann mit einem Programm die Priorität der einzelnen Flags festgesetzt werden.