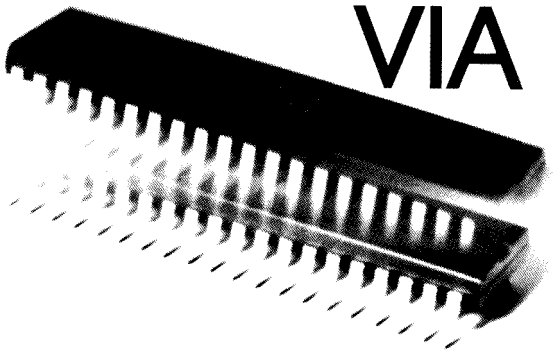
Junior-computer



VIA 6522

Uitgave:
Uitgeversmaatschappij Elektuur B.V.
Postbus 75 . 6190 AB Beek (L)

Rond uw Junior Computer af en haal het uiterste eruit! Dit VIA-boek helpt u daarbij.
U maakt kennis met de beide bidirektionele I/O-poorten van deze veelzijdige interface-adapter. Ook de werking van de beide 16 bits timer/counters wordt uitvoerig uit de doeken gedaan. Met het geïntegreerde schuifregister kan men op eenvoudige wijze parallelle data omzetten in seriële data en omgekeerd.
Bovendien zijn in dit boek de "hand shake"-mogelijkheden van de VIA op overzichtelijke wijze samengevat. Talrijke figuren en tabellen maken de werking en de programmering van de VIA begrijpelijk.
Een nuttige aanvulling op uw Junior Computer boekenserie!

Uitgeversmaatschappij Elektuur B.V.
Postbus 75 6190 AB Beek (L)

ISBN 90-70160-29-3

voorwoord

Een microprocessor-systeem (huis-computer, programmeerbare schakelklok, speel-computer, enz.) is voor de gebruiker een min of meer kompakt apparaat waarvan de toegankelijkheid niet groter is dan door de fabrikant mogelijk is gemaakt. De kommunicatie vindt plaats via een aantal in- en uitgangen, bestaande uit één of meer interface-schakelingen met een opmerkelijk veelzijdige opzet. Deze PIA, VIA, PIOT, enz. werken in wezen allemaal op eenzelfde wijze, ofschoon ze voor verschillende doeleinden gedacht zijn. Ze kunnen worden vergeleken met een sluis waarmee het kontakt tussen "binnen" en "buiten" mogelijk wordt gemaakt.
Dit rijk geïllustreerde boek is geheel gewijd aan één van die schakelingen, de veel gebruikte VIA 6522, waarvan alle eigenschappen zullen worden besproken. Van het adresseren van de poorten en de invloed van de elektrische belasting op de logische nivo's tot en met interrupt-procedures met of zonder "hand shake" en de beide interval-timers: alle funkties worden uitgebreid behandeld.
Dit boek is niet alleen geschikt voor de gebruikers van de Junior Computer, maar ook voor alle andere bezitters van een systeem met één of meer 6522's. Het stelt hen in staat om in zeer korte tijd uitgebreide kennis te verwerven van dit IC dat souplesse paart aan doelmatigheid.

INHOUD

1. Veelzijdig

De Versatile Interface Adapter (VIA), type 6522, is een veelzijdige bouwsteen ten behoeve van de interface (koppeling) van een computer met de buitenwereld. Hij herbergt een aantal registers, waarvan we de funktie en de werking zullen leren kennen. Er zijn de volgende groepen registers:

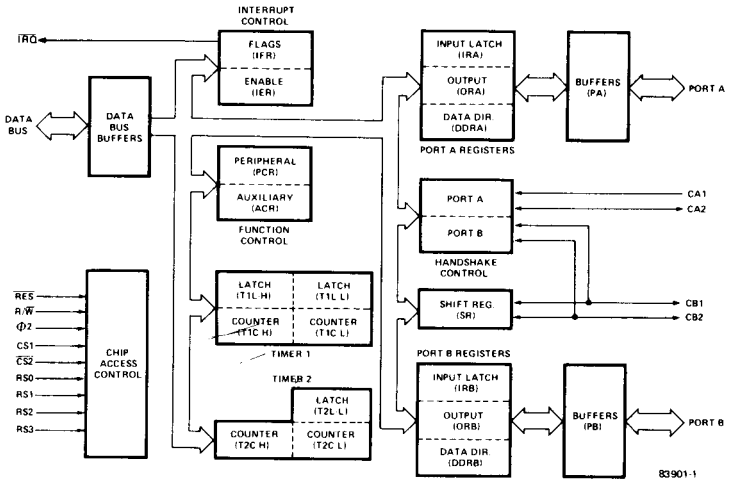
- twee achtbits poorten, A en B, die bidirektioneel zijn, dus als ingang of als uitgang kunnen worden geprogrammeerd;
- twee zestienbits timers/tellers;
- een achtbits I/O-schuifregister;
- vier zogenaamde handshake-leidingen, waarmee het dataverkeer van en naar aangesloten randapparaten kan worden geregeld;
- de in- of uitgaande data kan met behulp van een handshake-signaal in een tussenregister (latch) worden geschreven.

De voedingsspanning van de 6522 bedraagt 5 volt. Er is een 1 MHz-versie en een 2 MHz-versie (A). De beide poorten zijn TTL-kompatibel; poort A is bovendien ook CMOS-kompatibel.
De 6522 is zeer flexibel en kan worden toegepast in kombinatie met de microprocessoren 65XX, 68XX en ook met de zestienbits processor 68000. Die toepassingen zijn: het realiseren van zeer komplexe digitale signalen of nauwkeurige tijdsintervallen ten behoeve van tellers of klokken (timers/tellers); omzetting van seriële naar parallelle data en omgekeerd (schuifregister); opslag van in- of uitgaande data in het tussenregister; sturing van het dataverkeer in een multi-processorsysteem.
De sturing van randapparatuur (bijvoorbeeld een printer of een display) wordt in eerste instantie geregeld via de twee poorten. Elke poortlijn kan afzonderlijk als ingang of als uitgang worden geprogrammeerd. Bij de 6522 is het zelfs mogelijk om diverse poortlijnen direkt door de timer te laten sturen; het realiseren van blokgolfvormige spanningen met een programmeerbare frekwentie of "duty cycle" (de in % uitgedrukte verhouding tussen de "hoog-tijd" en de periode = "hoog-tijd" + "laag-tijd") is betrekkelijk eenvoudig. Ook het omgekeerde kan: één timer kan worden gestuurd uit poortlijnen en dus als teller worden gebruikt.

Ten behoeve van de programmering van de 6522 is voorzien in een viertal hulpregisters (we noemen ze bij hun oorspronkelijke Engelse naam):
- Interrupt Flag Register (IFR);
- Interrupt Enable Register (IER);
- Peripheral Control Register (PCR);
- Auxiliary Control Register (ACR).

In figuur 1 treft u het blokschema van de 6522 aan.

1

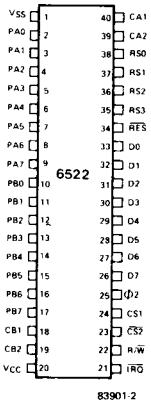


Figuur 1. Het blokschema van de VIA 6522.

2. De aansluitingen van de 6522

In figuur 2 is te zien welke funktie bij welke aansluiting (genummerd 1 . . . . 40) hoort. Alle aansluitingen zullen nu worden beschreven.

2



Figuur 2. De aansluitingen van de VIA 6522.

a. $\overline{RES}$ (Reset) (pen 34)
Via de reset-ingang worden alle registers binnen de 6522 gereset: alle bits zijn dan Ø. De tussenregisters en de tellers van Timer 1 en Timer 2 blijven echter onbeïnvloed. Alle poortlijnen zijn na het resetten als ingang geprogrammeerd. Alle interrupt-mogelijkheden ($\overline{IRQ}$) zijn geblokkeerd.
De reset-ingang is aktief indien zij Ø is (vandaar $\overline{RES}$ en niet RES).

b. Φ2 (klokingang) (pen 25)
Deze ingang is verbonden met de systeemklok Φ2 van 65XX-processors, danwel het Enable-signaal E van 68XX-processoren of van de 68000. Dit hartslag-signaal aktiveert alle vormen van dataverkeer tussen de VIA en de processor.

c. R/$\overline{W}$ (Read/$\overline{Write}$) (pen 22)
Het logische nivo op deze ingang bepaalt in welke richting dataverkeer tussen de VIA en de processor op een bepaald moment plaatsvindt. In de Ø-toestand leest de processor data uit een bepaald geselekteerd VIA-register; in de 1-toestand schrijft de processor data in een bepaald geselekteerd VIA-register. Vóór het lezen of schrijven moeten de beide Chip Select-ingangen CS1 en $\overline{CS2}$ worden geaktiveerd.

d. DØ . . . D7 (databus) (pennen 33 . . . 26)
Via de databus loopt het dataverkeer van en naar de systeemprocessor. Bidirektioneel dus. Gedurende het lezen van data wordt data uit een geselekteerd VIA-register op de databus gezet. Gedurende het schrijven van data gedragen de databusleidingen zich als acht hoogohmige ingangen; de data op de databus wordt in een geselekteerd VIA-register geschreven. Indien de Chip Select-ingangen CS1 en $\overline{CS2}$ niet aktief zijn is de VIA van de databus geïsoleerd, ontkoppeld.

3 6522-Register-File

Junior	Register	RS Coding				Register	Description	
Address	Number	RS3	RS2	RS1	RS0	Desig.	Write	Read
180Ø	0	0	0	0	0	ORB/IRB	Output Register "B"	Input Register "B"
180Ø1	1	0	0	0	1	ORA/IRA	Output Register "A"	Input Register "A"
180Ø2	2	0	0	1	0	DDRB	Data Direction Register "B"	
180Ø3	3	0	0	1	1	DDRA	Data Direction Register "A"	
180Ø4	4	0	1	0	0	T1C-L	T1 Low-Order Latches	T1 Low-Order Counter
180Ø5	5	0	1	0	1	T1C-H	T1 High-Order Counter	
180Ø6	6	0	1	1	0	T1L-L	T1 Low-Order Latches	
180Ø7	7	0	1	1	1	T1L-H	T1 High-Order Latches	
180Ø8	8	1	0	0	0	T2C-L	T2 Low-Order Latches	T2 Low-Order Counter
180Ø9	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"	

Figuur 3. Een overzicht van de 16 VIA-registers, met adresgegevens.

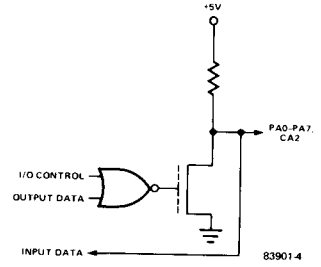
e. CS1 & $\overline{CS2}$ (Chip Select) (pennen 24 en 25)
Deze twee ingangen zijn doorgaans via een adresdekoder aangesloten op de adresbus van het processorsysteem. De VIA is ingeschakeld zodra CS1 1 is èn $\overline{CS2}$ Ø.

f. RSØ . . . RS3 (Registerselektie) (pennen 38 . . . 35)
Gewoonlijk zijn deze vier leidingen doorverbonden met de adreslijnen AØ . . . A3 (bij de 68000 A1 . . . A4). Met deze leidingen worden de 16 software-relevante VIA-registers geselekteerd door de systeemprocessor. In figuur 3 staat de register-selektietabel voor de 6522, gekombineerd met het geheugenplaatje (memory map) voor de Junior-Computer.

g. $\overline{IRQ}$ (Interrupt ReQuest-uitgang) (pen 21)
Deze uitgang is – samen met andere $\overline{IRQ}$-uitgangen – aangesloten op de $\overline{IRQ}$-lijn van het computersysteem. In het geval van een 68000 moet de $\overline{IRQ}$-lijn via een prioriteitskoder (bijvoorbeeld 74LS348) worden aangesloten.
In rust is de $\overline{IRQ}$-uitgang van de VIA 1. Dat wordt Ø zodra een interruptvlag in het register IFR èn de bijbehorende interrupt-enable-vlag in het register IER 1 zijn.
De $\overline{IRQ}$-uitgang is als open-kollektoruitgang uitgevoerd. Dit vanwege de al genoemde mogelijkheid tot het aan elkaar knopen van verschillende $\overline{IRQ}$-uitgangen.

h. PAØ . . . PA7 (Poort A) (pennen 2 . . . 9)
Alle acht leidingen van poort A kunnen afzonderlijk als in- of uitgang worden geprogrammeerd. De richting van een poortlijn (dus: ingang of uitgang) wordt bepaald door de waarde van een overeenkomstig bit van datarichtingsregister A (DDRA). Het logisch nivo van een poortlijn komt overeen met de waarde van een overeenkomstig bit van het datauitgangsregister (ORA). Het schrijven naar een als uitgang geprogrammeerde poortlijn geschiedt via een flipflop. Dat betekent dat de data ook nog na het schrijven op die poortlijn aanwezig is.
Bij een als ingang geprogrammeerde poortlijn ligt de mogelijkheid om te lezen voor de hand. Dat lezen kan op twee manieren plaatsvinden:
– lees de momentane (op het tijdstip dat R/$\overline{W}$ 1 is) waarde. Een spanning, hoger dan 2,4 volt wordt als 1 gelezen, een spanning, lager dan 2,4 volt als Ø;

4



Figuur 4. De opbouw van een A-poortlijn.

– lees het ingangsnivo dat tijdens een nivo-overgang van het signaal CA1 in een tussenregister is opgeslagen.

Hoe er wordt gelezen bepaalt de inhoud van het register PCR. Daarover later meer.

In figuur 4 treft u een schema aan van een poortlijn A. De weerstand naar +5 volt bedraagt ca. 7 kilo-ohm.

i. CA1 en CA2 (stuurlijnen poort A) (pennen 40 en 39)

Deze leidingen hebben een dubbele funktie. Ze kunnen fungeren als interrupt-ingangen, via welke een interruptvlag in het register IFR wordt geset, en wel via een nivoverandering van CA1 of CA2. Of het setten van een interruptvlag óók tot een interrupt ($\overline{IRQ}$-lijn laag) leidt hangt af van de toestand van de overeenkomende interrupt-enable-vlag in het register IER.

In de handshake-mode werkt CA1 als handshake-ingang, CA2 als handshake-uitgang. Omdat CA2, afhankelijk van de funktie, ingang of uitgang is, is de opbouw van CA2 identiek aan die van een A-poortlijn (figuur 4).

Als uitgang kan CA2 één standaard-TTL-belasting sturen.

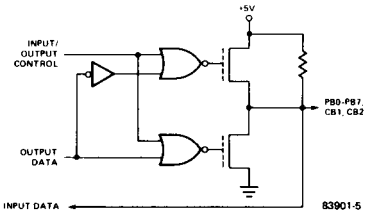
j. PB0 . . . PB7 (poort B) (pennen 10 . . . 17)

Het verhaal is identiek aan dat van poort A (h) met dien verstande dat het nu gaat om DDRB en DRB, in plaats van DDRA en DRA. De struktuur van een B-poortlijn (figuur 5) wijkt af van de struktuur van een A-poortlijn (figuur 4). Een als ingang geprogrammeerde B-poortlijn is niet hoogohmig, maar gedraagt zich als een TTL-ingang. Als uitgang kunnen de B-poortlijnen één TTL-standaardbelasting sturen (stroomopname in laagtoestand), danwel een stroom van 1 mA leveren (zij het ten koste van een daling van de uitgangsspanning tot 1,5 volt). Via poort B kunnen dus NPN- of PNP-darlington-transistoren direkt worden gestuurd.

Door de registers DDRB en ACR op de juiste manier te programmeren is het mogelijk om de uitgang van timer 1 door te verbinden met poortlijn PB7. Op PB7 zijn dus komplexe, jitter-vrije en software-gestuurde digitale signalen te realiseren.

Ook PB6 heeft desgewenst (programmering van DDRB en ACR) een speciale funktie, namelijk de funktie van een met timer 2 verbonden ingang. Het is dus mogelijk om timer 2 als pulsteller te gebruiken.

5



Figuur 5. De opbouw van een B-poortlijn.

k. CB1 en CB2 (stuurlijnen poort B) (pennen 18 en 19)
Het verhaal is deels identiek aan hoofdstukje i; kwa sturingsmogelijkheden is CB2 identiek aan een B-poortlijn (figuur 5).

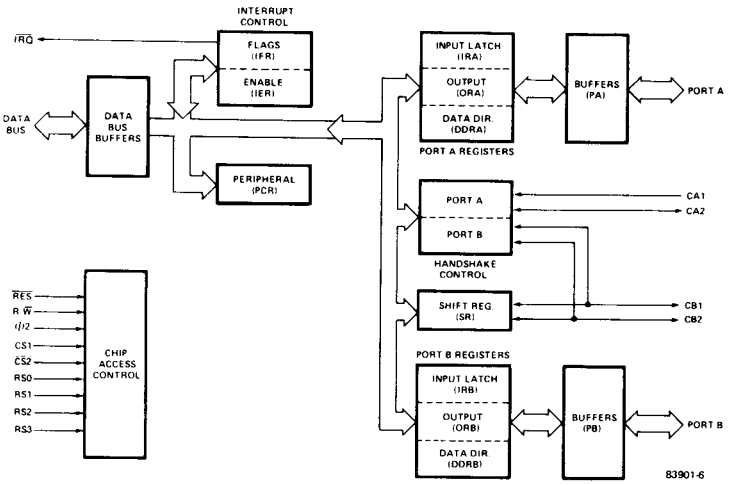
Echter: CB1 en CB2 hebben extra mogelijkheden ten opzichte van CA1 en CA2. Door de juiste software-dressuur van het register ACR kunnen CB1 en CB2 met het seriële schuifregister worden doorverbonden.
Via CB2 loopt een datastroom naar buiten (omzetting van de parallelle data van het schuifregister in seriële data), of komt een datastroom binnen (die in het schuifregister als parallelle data beschikbaar is).
Voor de omzetting van serieel naar parallel of omgekeerd zijn (schuif-) klokpulsen nodig. Die kunnen van interne oorsprong zijn of via CB1 worden toegevoerd.
Zie verder hoofdstuk 8.

l. +5 volt (V_{CC}) (pen 20) en massa (V_{SS}) (pen 1)

3. De werking van de poorten A en B

Aan poort A is het data-richtingsregister DDRA toegevoegd, aan poort B het data-richtingsregister DDRB. De bits in zo'n richtingsregister bepalen of de bijbehorende poortlijn als uitgang of als ingang is geprogrammeerd. Is een bit logisch nul, dan is de bijbehorende poortlijn ingang; voor een funktie als uitgang moet men ervoor zorgen dat het overeenkomstige bit in het overeenkomstige data-richtingsregister logisch één is.
Indien een poortlijn als uitgang is geprogrammeerd, komt er op die uitgang data te staan die overeen komt met de waarde van het korresponderende bit in het bijbehorende uitgangsregister (ORA voor poort A, ORB voor poort B). Indien niet alle lijnen van een poort als uitgang zijn geprogram-

6



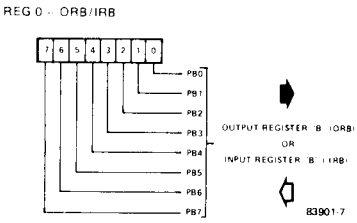
Figuur 6. Poort A, poort B en de handshake-leidingen.

meerd worden de als ingang geprogrammeerde poortlijnen niet beïnvloed door de overeenkomstige data van het uitgangsregister.
Bij het lezen van als ingang geprogrammeerde poortlijnen zijn er twee mogelijkheden, afhankelijk van de inhoud van het register PCR:

1. De processor leest het ingangsregister IRA (poort A) of IRB (poort B). De inhoud van ACR is zodanig dat het niet mogelijk is om "ingangsdata te bevriezen" (input latching disabled, zie figuur 16). Dat betekent dat data op als ingang geprogrammeerde poortlijnen niet tijdens een nivoverandering van CB1 of CA1 in IRA of IRB wordt opgeslagen, maar dat de momentane ingangsdata op de poortlijnen wordt gelezen.
2. Ook nu wordt IRA of IRB gelezen. De inhoud van ACR is zó geprogrammeerd dat ingangsdata kan worden "bevroren" (input latching enabled). Nu wordt de inhoud van IRA of IRB gelezen, te weten de inhoud die is doorgegeven vanuit de poortlijnen op het tijdstip van een nivoovergang van CA1 of CB1.

Zoals al opgemerkt kan de processor óók een als uitgang geprogrammeerde poortlijn lezen. Is er bijvoorbeeld (onlangs) een één op een uitgang geschreven, dan wordt er ook een één gelezen. Hetzelfde geldt voor een nul.
Echter: Indien men een als uitgang geprogrammeerde poortlijn A zodanig elektrisch belast dat de uitgangsspanning tot onder 2,4 volt daalt (na het schrijven van een één in die poortlijn), zal een nul worden gelezen. Dit geldt voor poort A, niet voor poort B. Poort B is namelijk sterker gebufferd; een daling van de uitgangsspanning heeft geen invloed op het leesgedrag. Men dient goed te letten op dit verschil tussen de poorten A en B als het gaat om het maskeren van bits in IRA en IRB, in het geval van "zwaar" belaste poortlijnen.

7

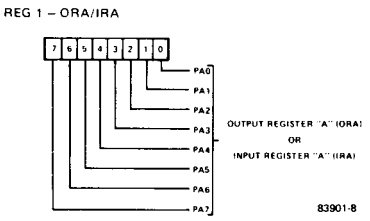


Keuze ingang/uitgang	schrijven	lezen
DDRB = 1: overeenkomstige poortlijn is uitgang	CPU schrijft een Ø of 1 op de poortlijn (write ORB)	CPU leest ORB, **niet** het nivo op de lijn!
DDRB = Ø: overeenkomstige poortlijn is ingang (met bevriezing van ingangsdata)	CPU schrijft in ORB. Echter geen invloed op ingang	CPU leest nivo op de poortlijn
DDRB = Ø: overeenkomstige poortlijn is ingang (zonder bevriezing van ingangsdata)		CPU leest IRB (= data tijdens de laatste CB1-nivoverandering

Figuur 7. Poort B, met ingangsregister IRB en uitgangsregister ORB.

In de figuren 7 . . . 9 is de samenhang van de poortlijnen met de ingangs- en uitgangsregisters, alsmede de data-richtingsregisters, geïllustreerd.

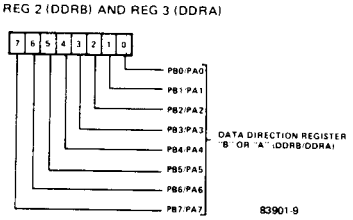
8



keuze ingang/uitgang	schrijven	lezen
DDRA = 1: overeenkomstige poortlijn is uitgang (zonder bevriezing van ingangsdata)	CPU schrijft een Ø of 1 op de poortlijn (write ORA)	CPU leest spanning op poortlijn (spanning afhankelijk van belasting)
DDRA = 1: overeenkomstige poortlijn is uitgang (met bevriezing van ingangsdata)		CPU leest IRA (= data tijdens de laatste CA1-nivoverandering)
DDRA = Ø: overeenkomstige poortlijn is uitgang (zonder bevriezing van ingangsdata)	CPU schrijft in ORA. Echter geen invloed op ingang	CPU leest poortlijn
DDRA = Ø: overeenkomstige poortlijn is uitgang (met bevriezing van ingangsdata)		CPU leest IRA (= data tijdens de laatste CA1-nivoverandering)

Figuur 8. Poort A, met ingangsregister IRA en uitgangsregister ORA.

9



Figuur 9. De inhoud van de data-richtingsregisters DDRA en DDRB bepaalt welke poortlijnen ingang (bit = Ø) zijn en welke uitgang (bit = 1).

4. Handshaking via de VIA

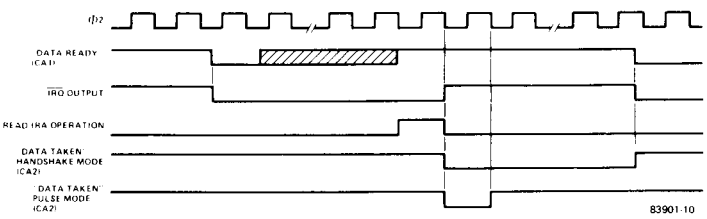
Door bemiddeling van de VIA is het mogelijk om via de poortlijnen A en B allerlei randapparaten, zoals een printer, een cassette-recorder of een tweede computersysteem op het systeem waar de VIA deel van uitmaakt

aan te sluiten. Deze randapparatuur verwerkt data doorgaans veel minder snel dan de computer. Door middel van "handshake"-signalen (geen gouden maar een digitale handdruk) deelt een randapparaat de computer mede dat nieuwe data (via de poortlijnen) kan worden ontvangen, of deelt de computer aan een randapparaat mede dat hij graag data wil verzenden. Ten behoeve van dit vraag-en-antwoord-spelletje is voorzien in de stuurlijnen CA1 en CA2 voor poort A en CB1 en CB2 voor poort B.
Er is kwa handshake-mogelijkheden een duidelijk verschil tussen de poorten A en B. Onder gebruikmaking van CA1 en CA2 is via poort A tweerichtingsverkeer mogelijk (lezen of schrijven, dus ontvangen of verzenden van data). Met CB1 en CB2 is handshake-sturing slechts dan mogelijk indien de processor in poort B schrijft.

a. Read-handshake
Hiervan is sprake indien de processor via de poortlijnen A van een randapparaat ontvangen data inleest, onder supervisie van CA1 en CA2. Het randapparaat verstuurt data naar poort A en deelt de processor via CA1 of CA2 mee dat het om geldige data gaat. Dit laatste geschiedt met behulp van het signaal DATA READY. Dit signaal maakt een bepaalde interruptvlag van het register IFR één (zie figuur 30). Indien het korresponderende bit in het register IER eveneens één is, wordt de $\overline{IRQ}$-lijn van de VIA nul. Het is dus mogelijk om via het DATA READY-signaal een interruptroutine af te wikkelen. Indien echter het overeenkomende bit in IER nul is, is een interrupt onmogelijk. De processor moet dan door het lezen van het register IFR vaststellen op welke van de handshake-leiding sprake is van voor verzending (door een randapparaat) gereed staande data. Deze manier van werken heet "polling".
Nadat de processor de data gelezen heeft deelt hij het randapparaat via de tweede handshake-leiding mede dat nieuwe data naar de poortlijnen kan worden verzonden.
Samenvattend:
* het randapparaat verzendt data en vertelt de computer via CA1 dat het om geldige data gaat;
* de processor leest de aangeboden geldige data in. Nadat dit gebeurd is verzendt de computer via CA2 een DATA TAKEN-signaal naar het randapparaat, ten teken dat, wat de computer betreft nieuwe data kan worden ontvangen;
* het randapparaat verzendt nieuwe data en vertelt dat via CA1. Een en ander gaat net zo lang door totdat alle data verzonden is.

Een automatische handshake-sturing behoort tot de mogelijkheden: CA1 is aangesloten op de DATA READY-lijn, CA2 op de DATA TAKEN-lijn. De vlag in het register IFR die daardoor één wordt kan men gebruiken voor het aktiveren en afwikkelen van een interruptroutine, of men leest IFR (polling). Het signaal DATA TAKEN (CA2) kan naar keuze (programmering PCR) worden geaktiveerd in de vorm van een snelle nivoverandering danwel als statische nivoverandering. In het laatste geval wordt het signaal DATA TAKEN automatisch gereset zodra een nieuw DATA READY-signaal binnenkomt. In figuur 10 vindt u een "pulsplaatje" van read-handshaking op poort A. Zoals al eerder opgemerkt: dit kan alleen maar met poort A, niet met poort B!

10

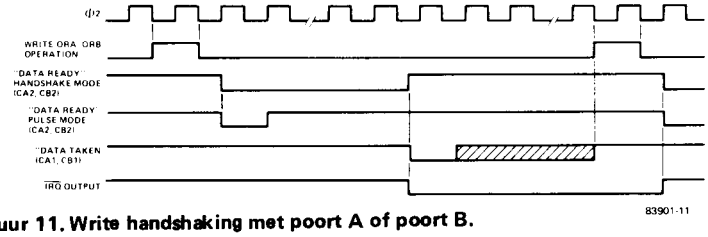


Figuur 10. Read handshaking met poort A. Is niet mogelijk met poort B!

b. Write-handshake

Lijkt erg veel op read-handshake. Nu verzorgt echter de VIA (onder invloed van de processor) het signaal DATA READY, en niet het randapparaat. Nu sluit het randapparaat het inlezen van door de computer verzonden data af met het DATA TAKEN-signaal. Deze vorm van handshaking is voor beide poorten A en B mogelijk. Als handshake-uitgangen doen CA2 of CB2 dienst; ze verzorgen het signaal DATA READY, hetzij als dynamische nivoverandering (pulsflank), hetzij als statische nivoverandering. Dat laatste is een kwestie van de juiste programmering van PCR.
Als handshake-ingangen doen CA1 en CB1 dienst; ze ontvangen het signaal DATA TAKEN van het randapparaat. Zodra de data door het randapparaat is gelezen wordt een vlag van IFR één. Dit kan òf de afwikkeling van een interruptroutine, òf het lezen van IFR (polling) tot gevolg hebben. Indien er is gekozen voor de statische vorm van het uit te zenden signaal DATA READY (CA2 of CB2), wordt de DATA READY-lijn automatisch gereset zodra een nieuw DATA TAKEN-signaal binnenkomt. In figuur 11 ziet u de gang van zaken tijdens write-handshaking.

11



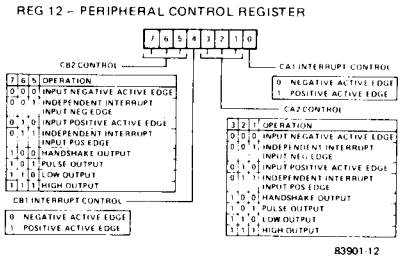
Figuur 11. Write handshaking met poort A of poort B.

Indien read- of write-handshaking via een interruptroutine wordt afgehandeld moet de interruptroutine worden afgesloten (d.w.z. direkt vóór RTI) met een instruktie die ervoor zorgt dat de $\overline{IRQ}$-lijn weer één wordt (rusttoestand), en wel via het schrijven van data in het register IFR.

5. Het register PCR

Er zijn nogal wat mogelijkheden om handshake-sturing te realiseren. Door het schrijven van data in het register PCR maakt men een keuze uit die mogelijkheden. De onderste vier bits (0 . . . 3) van PCR betreffen CA1 en CA2, de bovenste vier (4 . . . 7) betreffen CB1 en CB2. Zie ook figuur 12.

12



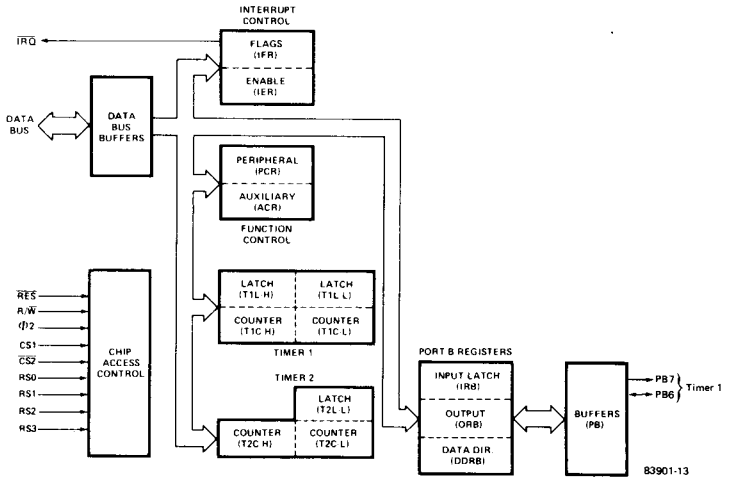
Figuur 12. Het hulpregister PCR.

Indien bit Ø van PCR één is reageert CA1 op positieve flanken (Ø/1-overgangen); is bit Ø nul, dan reageert CA1 op negatieve flanken (1/Ø-overgangen). Indien bit 3 nul is werkt CA2 als ingang; is dit bit één, dan is CA2 uitgang. Met de bits 1 en 2 zijn vier verschillende situaties voor CA2, als ingang of als uitgang, programmeerbaar. Figuur 12 biedt meer informatie.
Voor CB1 en CB2 geldt hetzelfde verhaal met dien verstande dat de bits 4 . . . 7 kwa funktie overeenkomen met de bits Ø . . . 3.

6. Timer 1

De intervaltimer 1 bestaat uit twee achtbits tussenregisters (latches) en een 16-bits teller-register. De tussenregisters doen dienst als tussengeheugen voor het teller-register. Nadat het teller-register van timer 1 is geladen begint het aftellen, in het ritme van de Φ2-klokpuls. Zodra de inhoud van het teller-register nul is geworden wordt timer-interruptvlag 1, dus geset. Na de eerstvolgende klokpuls komt de inhoud van het teller-register overeen met het getal −1. Indien de overeenkomstige enable-vlag in het register

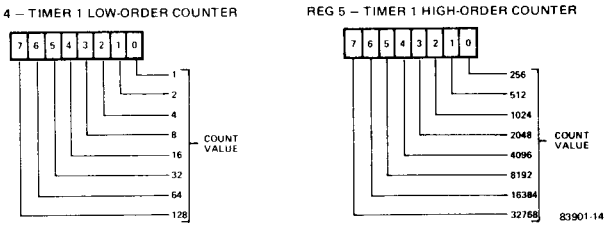
13



Figuur 13. De twee intervaltimers 1 en 2.

IER is geset, wordt de $\overline{IRQ}$-leiding nul; de CPU is dan in staat om een interruptroutine af te werken. Via het register ACR (zie figuur 16) moet men de werkwijze van timer 1 programmeren. Indien de inhoud van ACR het mogelijk maakt dat timer 1 opnieuw aftelt, wordt na elke time out, dus elke keer dat de inhoud van het teller-register nul is, de inhoud van de twee tussenregisters in het teller-register gekopieerd. Ook is het mogelijk om na elke time out het logisch nivo op poortlijn PB7 om te keren, dus een één wordt een nul en omgekeerd. Op PB7 staat dan een symmetrische blokspanning, waarvan de frekwentie wordt bepaald door de inhoud van de twee tussenregisters en door de klokfrekwentie.

14



Write: schrijf 8 bits in het tussenregister (laag), waarvan inhoud naar het teller-register (laag) gaat zodra in teller-register (hoog) is geschreven

Read: CPU leest teller-register (laag). Interruptvlag gereset

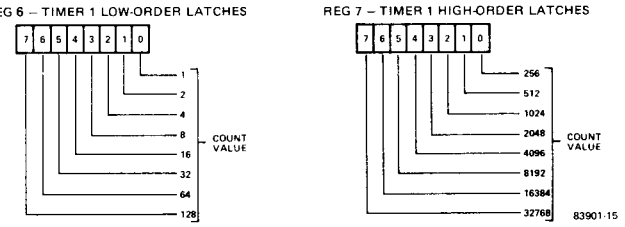
Write: schrijf 8 bits in het teller-register (hoog) en tussenregister (hoog). Inhoud tussenregisters → teller-register. Daarna start aftellen en resetten interruptvlag

Read: CPU leest teller-register (hoog). Interruptvlag wordt **niet** gereset.

Door het lezen van het teller-register (van timer 1) kan de computer vaststellen hoe lang het nog duurt voordat een time out optreedt, danwel hoeveel tijd er sinds de time out is verstreken.

Figuur 14. Lezen en schrijven rond het teller-register van timer 1.

15



Write: Schrijf 8 bits in het tussenregister (laag), waarvan inhoud naar het teller-(REG 4).

Read: CPU leest het tussenregister (laag). Interruptvlag van timer 1 blijft onbeïnvloed

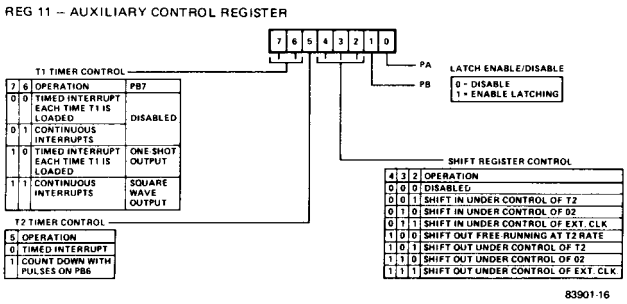
Write: Schrijf 8 bits in het teller-register (hoog) en tussenregister (hoog). Inhoud laag) wordt niet in teller-register gekopieerd.

Read: CPU leest het tussenregister (hoog). Het aftellen en de interruptvlag van timer 1 blijft onbeïnvloed

Figuur 15. Lezen en schrijven rond de tussenregisters van timer 1.

In de figuren 14 en 15 ziet u de registers van timer 1. Ook ziet u hoe het lezen van en schrijven in een register in zijn werk gaat. In en bij figuur 16 is te zien hoe de bits 6 en 7 van het register ACR de werkwijze van timer 1 beïnvloeden. Er zijn dus vier verschillende gebruiksmogelijkheden voor timer 1. Bij het schrijven in de registers van timer 1 moet men op het volgende letten:

16



b0, b1 — Maak het bevriezen van ingangsdata op de poorten A (B) mogelijk. In het geval van bevriezen van ingangsdata gaat deze data naar IRA (IRB) tijdens een nivoverandering van CA1 (CB1).

b4, b3, b2 — sturing van het schuifregister

0 0 0 — geen schuiven

0 0 1 — inschuiven onder supervisie van timer 2

0 1 0 — inschuiven onder supervisie van de Φ2-systeemklok

0 1 1 — inschuiven onder supervisie van een externe klok

1 0 0 — uitschuiven onder supervisie van timer 2 (free running)

1 0 1 — uitschuiven onder supervisie van timer 2

1 1 0 — uitschuiven onder supervisie van de Φ2-systeemklok

1 1 1 — uitschuiven onder supervisie van een externe klok

b5 — werkwijze van timer 2

0 — interrupt na time out van timer 2

1 — terugtellen met impulsen op PB6

b7, b6 — werkwijze van timer 1

0 0 — interrupt na time out van timer 1. Geen invloed op PB7.

0 1 — periodieke interrupts: na elke time out opnieuw aftellen. Geen invloed op PB7.

1 0 — interrupt na time out van timer 1. PB7 werkt als monoflop-uitgang (one shot).

1 1 — periodieke interrupts: na elke time out opnieuw aftellen. Blokspanning op PB7.

Figuur 16. Het hulpregister ACR.

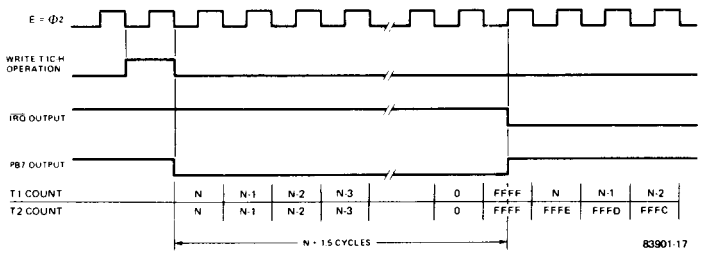
Data, bestemd voor de onderste helft van het teller-register moet in het bijbehorende tussenregister worden geschreven. De data-overgang van het tussenregister naar de onderste helft van het teller-register vindt pas plaats nadat er data in de bovenste helft van het teller-register (alsmede in het bijbehorende tussenregister) is geschreven. Op datzelfde moment begint het aftellen.

a. Timer 1 one shot mode

In deze toestand komt na iedere time out een interrupt tot stand. Uiteraard moet er eerst data in het teller-register zijn geschreven, anders krijg je geen time out. Verder is timer 1 via ACR zo te programmeren dat gedurende het aftellen een negatieve puls op PB7 ontstaat.
Om een eenmalige interrupt via timer 1 te realiseren moeten de bits 6 en 7 van ACR nul zijn. Dan moet de processor een offset schrijven in hetzij de onderste helft van het teller-register (low order counter) hetzij het bijbehorende tussenregister (low order latch). Na dit schrijven schrijft de processor een offset direkt in de bovenste helft van het teller-register (high order counter). Deze offset wordt ook in het bijbehorende tussenregister (high order latch) gezet en het aftellen begint, in het ritme en met de snelheid van de Φ2-klokpulsen. Zodra de inhoud van het teller-register nul is wordt de interruptvlag van timer 1 (in het register IFR) geset.
Indien PB7 tijdens het aftellen nul moet zijn moet bit 7 van ACR één zijn; PB7 wordt direkt na het starten van het aftellen nul en direkt na de time out weer één.
De timer 1-vlag van ACR kan men op twee manieren weer nul maken:

1. de processor schrijft een offset in de bovenste helft van het teller-register. Dit levert opnieuw aftellen op;
2. de processor leest de bovenste helft van het teller-register. Het aftellen wordt niet beïnvloed.

17



Figuur 17. De werking van timer 1 als "software-monoflop" (one shot mode).

In figuur 17 ziet men het pulsdiagram bij de one shot mode. Poortlijn PB7 blijft gedurende N + 1½ klokpulsen nul; N is een hexadecimaal getal 0000 . . . FFFF. Men ziet dat de IRQ-leiding nul wordt op hetzelfde tijdstip dat PB7 weer één wordt.

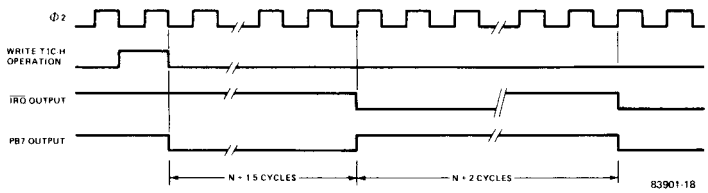
b. Timer 1 free run mode

Het belangrijkste voordeel van timer 1 is dat men er een zeer gaaf en jitter-

vrij bloksignaal (op PB7) mee kan realiseren.
In de free run mode van timer 1 wordt de timer 1-interruptvlag één en het logisch nivo op PB7 omgekeerd zodra de inhoud van het teller-register nul is geworden. In tegenstelling tot de one shot mode gaat timer 1 niet door met het aftellen, maar begint opnieuw met aftellen nadat de inhoud van de twee tussenregisters in het teller-register is gekopieerd.
Na de eerste time out van timer 1 blijft de bijbehorende interruptvlag in het register IFR één. Resetten kan op drie manieren:
1. de processor schrijft een één in het register IFR (zie hoofdstuk 9);
2. de processor leest de onderste helft van het teller-register;
3. de processor schrijft nieuwe data in de bovenste helft van het teller-register.

Voor beide timers van de VIA geldt dat vóór een time out via een schrijfoperatie in het teller-register opnieuw met aftellen kan worden begonnen. Het is echter ook mogelijk om vóór een time out in de tussenregisters te schrijven. De nieuwe aftelsituatie treedt dan pas op na de time out, dus na afwikkeling van de lopende zaken.
Indien de processor onder invloed van de software telkens na een time out de timer 1-interruptvlag reset, kan deze vlag na de volgende time out weer worden geset. Via een interrupt- of polling-programma kan de processor na elke time out nieuwe data schrijven in de tussenregisters van timer 1. De duty cycle van het signaal PB7 kan men op deze manier willekeurig variëren. Op PB7 zijn zeer komplexe blokvormige signalen te realiseren, die door software zijn bepaald. Bij een klokfrekwentie van 1 MHz zijn signalen mogelijk met een frekwentie die varieert van nul tot 250 kHz.
Figuur 18 toont het pulsplaatje dat hoort bij deze werkwijze van timer 1.

18



Figuur 18. Voor het realiseren van een blokvormige spanning met een programmeerbare frekwentie moet timer 1 in de free running mode geprogrammeerd zijn. Let op! Indien PB7 als uitgang van timer 1 wordt gebruikt moeten de bits 7 van DDRB en ACR één zijn. Verder moet PB7 als uitgang zijn geprogrammeerd.

7. Timer 2

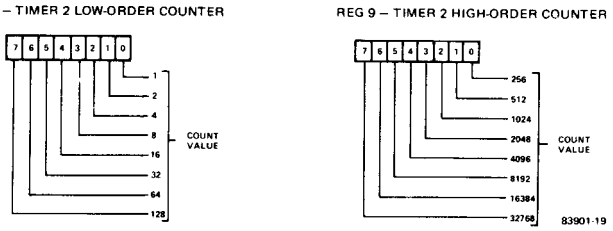
Belangrijk verschil ten opzichte van timer 1: er is geen tussenregister voor de bovenste acht bits van het teller-register. De free running mode is derhalve niet mogelijk voor timer 2. Via programmering van ACR (bit 5, zie figuur 16) is timer 2 op twee manieren te gebruiken:
1. als interval-timer in de one shot mode;
2. als afteller, gestuurd door negatieve flanken op PB6.

Ook het lezen en schrijven verschilt van timer 1:

* in het onderste tussenregister van timer 2 kan uitsluitend data worden geschreven (write only);
* de onderste helft van het teller-register kan uitsluitend worden gelezen (read only);
* de bovenste helft van het teller-register kan worden gelezen en beschreven (read/write).

Het teller-register van timer 2 is eveneens 16 bits breed en telt eveneens af in het ritme van de Φ2-klokpulsen. In figuur 19 is te zien wat er gebeurt als de registers van timer 2 worden gelezen of beschreven.

19



Write: Schrijf 8 bits in het tussenregister (laag)

Read: CPU leest teller-register (laag). Interruptvlag van timer 2 wordt gereset

Write: Schrijf 8 bits in het teller-register (hoog). Inhoud tussenregister (laag) gaat naar teller-register (laag). Start aftellen. Interruptvlag van timer 2 gereset

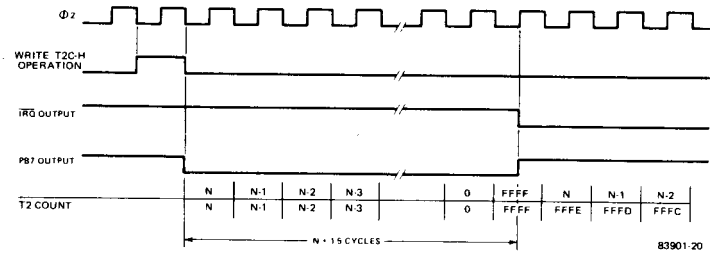
Read: CPU leest teller-register (hoog). Interruptvlag van timer 2 wordt **niet** gereset.

Figuur 19. Lezen en schrijven rond het teller-register van timer 2.

a. Timer 2 one shot mode

In de one shot mode werkt timer 2 net zo als timer 1. Na elke time out wordt de bijbehorende interruptvlag geset, dus één. Indien de korresponderende enable-vlag in het register IER eveneens één is, wordt de $\overline{IRQ}$-lijn aktief, dus nul; via polling kan de processor dan IFR lezen.
Na de time out (inhoud teller-register $0000) gaat timer 2 door met aftellen, dus $FFFF, $FFFE, $FFFD, enzovoorts. Via het lezen van de onderste helft van het teller-register kan de computer dus aan de weet

20



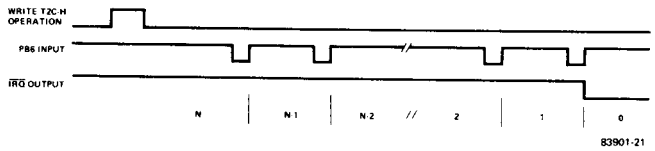
Figuur 20. De werking van timer 2 in de one shot mode.

komen hoeveel tijd er sinds de time out is verstreken. Net als bij timer 1 is het met timer 2 mogelijk om de afteltijd willekeurig te verlengen of te verkorten. Daartoe moet de processor voortdurend tijdens een aftelperiode de inhoud van het onderste tussengeheugen en die van de bovenste helft van het teller-register van nieuwe data voorzien. Bij het lezen en schrijven moet men erop letten dat de interruptvlag van timer 2 door het lezen (van het lage tussenregister) of schrijven (in de bovenste helft van het teller-register) nul wordt. De werking van timer 2 in de one shot mode is te zien in figuur 20.

b. Timer 2 pulse counting mode

In dit geval telt een bepaald aantal negatieve impulsen op de poortlijn PB6. Daartoe moet eerst een 16-bits getal in het teller-register van timer 2 worden geschreven. Na het schrijven begint het aftellen van timer 2: bij elke negatieve impuls op PB6 wordt het getal met één verlaagd. Zodra de inhoud van het teller-register $\$\emptyset\emptyset\emptyset\emptyset$ is wordt de bijbehorende interruptvlag één. Indien het overeenkomende enable-bit in het register IER eveneens één is, wordt de $\overline{IRQ}$-leiding aktief, dus nul. De processor wikkelt dan een interruptroutine af. Wil men opnieuw aftellen, dan moet men timer 2 opnieuw laden. In dat geval wordt de interruptvlag weer gereset. In figuur 21 is de gang van zaken geïllustreerd.

21



Figuur 21. De werking van timer 2 in de pulse counting mode: als pulsteller.

Indien de interruptvlag voor timer 2 gereset is en indien de processor de onderste helft van het teller-register 2 leest, kan worden vastgesteld hoeveel negatieve impulsen er nog nodig zijn (op PB6) om de inhoud van het teller-register $\$\emptyset\emptyset\emptyset\emptyset$ te maken. Indien de interruptvlag voor timer 2 geset is en indien de processor het teller-register 2 leest, kan worden vastgesteld hoeveel negatieve impulsen zijn opgetreden sinds de time out. Immers: het aftellen gaat gewoon door: $\$\emptyset\emptyset\emptyset\emptyset$, $FFFF, $FFFE, $FFFD, $FFFC...

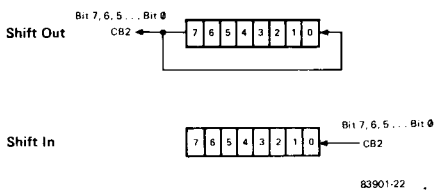
8. De VIA-schuifregisters

Met de VIA is het mogelijk om van een randapparaat ontvangen seriële data in parallelle data om te zetten of om naar een randapparaat te verzenden parallelle data om te zetten in seriële data. Het schuiven kan naar keuze onder invloed van interne of externe (schuif-)klokpulsen plaatsvinden. Externe klokpulsen worden via CB1 toegevoerd. Het is ook mogelijk om de klokpulsen door timer 2 te laten leveren; deze klokpulsen staan dan op CB1 ter beschikking. Het is dus mogelijk om hiermee allerlei externe schakelingen te sturen. De verschillende werkwijzen als schuifregister hangen af van de inhoud van het register ACR (de bits 2, 3 en 4). Zie hiervoor figuur 16. In figuur 22 is te zien in welke volgorde de diverse bits van een seriële

datastroom worden ontvangen of verzonden. Bij het verzenden is de volgorde: bit 7 . . . bit Ø. Verder wordt het laatst verzonden bit naar bitpositie Ø van het schuifregister geroteerd. Bij binnenkomst (via CB2) van bits is de schuifvolgorde: bit Ø . . . bit 7. Het schuifregister verwerkt willekeurige 8-bits woorden zonder de voor ASCII-data gebruikelijke start- en stopbits. Echter: start- en stopbits zijn betrekkelijk eenvoudig te realiseren met een klein programmaatje.
Het schuifregister heeft acht verschillende gebruiksmogelijkheden:

Nul ACR b4b3b2 = ØØØ
Het schuifregister is uitgeschakeld, d.w.z. er wordt niet geschoven. De processor schrijft data in of leest data uit het schuifregister. De interruptvlag van het schuifregister is altijd gereset, dus nul. De handshake-leidingen CB1 en CB2 worden gestuurd uit het register PCR.

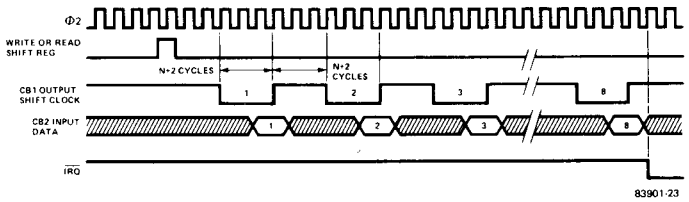
22



Figuur 22. Zo wordt data het schuifregister in- of uitgeschoven.

Eén ACR b4b3b2 = ØØ1
Onder supervisie van timer 2 wordt data het schuifregister ingeschoven. Alleen de onderste acht bits van het teller-register en het bijbehorende tussenregister worden gebruikt. Op de CB1-lijn staat het schuifkloksignaal ter beschikking.
Schuiven vindt plaats na het schrijven in of lezen van het schuifregister. Na acht schuifklokpulsen is het schuifregister vol; de interruptvlag van het schuifregister wordt dan geset, dus één. De schuifklokfrekwentie hangt samen met de systeem-klokfrekwentie (Φ2) en de inhoud van het (lage) tussenregister van timer 2. In figuur 23 is te zien hoe de baudrate ermee samenhangt. Het getal N staat in digitale vorm in het tussenregister.

23

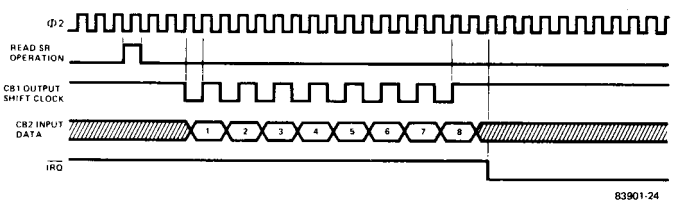


Figuur 23. Gebruiksmogelijkheid één van het schuifregister: inschuiven onder supervisie van timer 2.

Twee ACR b4b3b2 = Ø1Ø
Onder supervisie van de Φ2-klokpulsen wordt data het schuifregister inge-

schoven. Op CB1 zijn de schuifklokpulsen (=Φ2) beschikbaar. Schuiven vindt plaats na het schrijven in of lezen van het schuifregister. Na acht schuifklokpulsen is het schuifregister vol; de interruptvlag van het schuifregister wordt geset. Verder wordt dan de versturing van klokpulsen naar CB1 beëindigd. Zie ook figuur 24.

24

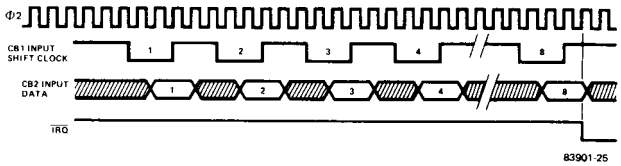


Figuur 24. Gebruiksmogelijkheid twee van het schuifregister: inschuiven onder supervisie van de Φ2-systeemklok.

Drie ACR b4b3b2 = Ø11

Onder supervisie van externe klokpulsen wordt data het schuifregister ingeschoven. Het externe kloksignaal moet via CB1 worden toegevoerd. Schuiven vindt plaats na het lezen van of schrijven in het schuifregister. Na acht schuifklokpulsen is het register vol; de bijbehorende interruptvlag wordt geset. Deze vlag wordt weer nul na het lezen van of schrijven in het schuifregister. Zie verder figuur 25.

25

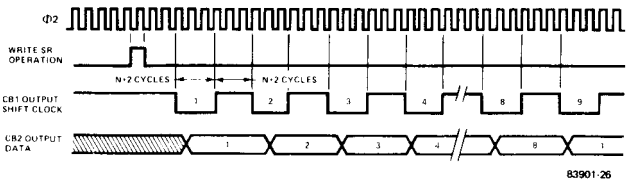


Figuur 25. Gebruiksmogelijkheid drie van het schuifregister: inschuiven onder supervisie van een externe klok.

Vier ACR b4b3b2 = 1ØØ

Onder supervisie van timer 2 wordt de inhoud van het schuifregister via CB2 uitgeschoven. Zoals in figuur 26 is te zien is de schuifsnelheid afhankelijk van de inhoud van het (lage) tussenregister van timer 2. De inhoud is de digitale versie van het getal N. De schuifklokpulsen zijn aanwezig op

26

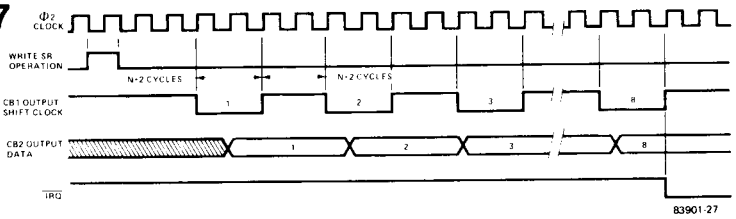


Figuur 26. Gebruiksmogelijkheid vier van het schuifregister: uitschuiven onder supervisie van timer 2.

CB1; men kan er externe schakelingen mee sturen. Na acht schuifklokpulsen gaat het schuiven gewoon door. De weggeschoven data gaat niet verloren, want tijdens het schuiven wordt bit 7 naar bit Ø geroteerd. Dit is de "free running mode" van het schuifregister. De bijbehorende interruptvlag blijft onbeïnvloed.

Vijf ACR b4b3b2 = 1Ø1
Onder supervisie van timer 2 wordt de inhoud van het schuifregister via CB2 uitgeschoven. Zoals in figuur 27 is te zien is de schuifsnelheid afhankelijk van de inhoud van het (lage) tussenregister van timer 2. Die inhoud is de digitale versie van het getal N. De schuifklokpulsen zijn aanwezig op CB1; men kan er externe schakelingen mee sturen. Nadat alle acht bits zijn uitgeschoven stopt het schuiven; de bijbehorende interruptvlag wordt geset. Op CB2 staat een logisch nivo dat gelijk is aan het laatst uitgeschoven bit.

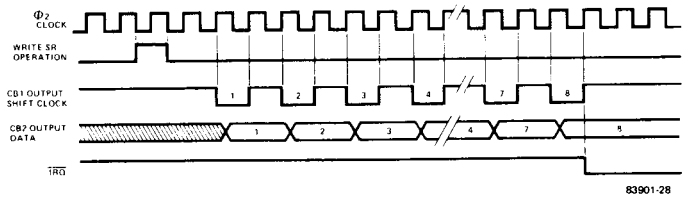
27



Figuur 27. Gebruiksmogelijkheid vijf van het schuifregister: uitschuiven onder supervisie van timer 2. Na acht schuifoperaties stopt het schuiven.

Zes ACR b4b3b2 = 11Ø
Onder supervisie van de systeemklok (Φ2) wordt de inhoud van het schuifregister via CB2 uitgeschoven. De Φ2-klokpulsen zijn aanwezig op CB1; men kan er externe schakelingen mee sturen. Na het schuiven van acht bits stopt het schuiven. Verder wordt de bijbehorende interruptvlag geset. Zie ook figuur 28.

28

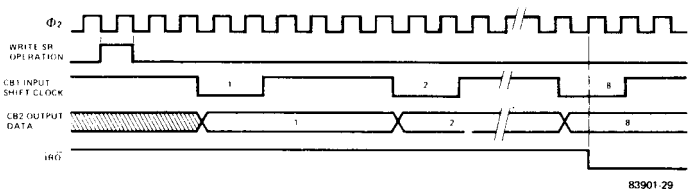


Figuur 28. Gebruiksmogelijkheid zes van het schuifregister: uitschuiven onder supervisie van de Φ2-systeemklok.

Zeven ACR b4b3b2 = 111
Onder supervisie van externe klokpulsen wordt de inhoud van het schuifregister via CB2 uitgeschoven. Het externe kloksignaal moet via CB1 wor-

den toegevoerd. Na het schuiven van acht databits wordt de bijbehorende interruptvlag geset. Het schuiven gaat echter gewoon door. Resetten van de interruptvlag is mogelijk door het lezen van of schrijven in het schuifregister. Na de eerstvolgende acht schuifoperaties wordt de interruptvlag weer geset. Telkens na het één zijn van de interruptvlag kan er door de processor nieuwe data in het schuifregister worden geschreven. In figuur 29 is de gang van zaken geschetst.

29

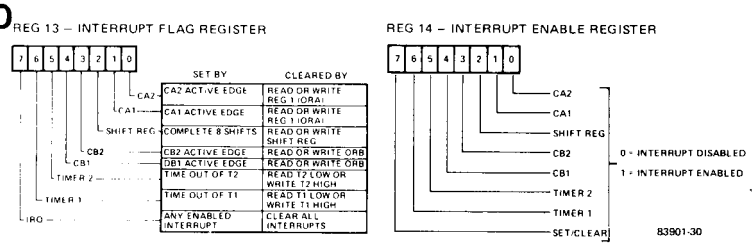


Figuur 29. Gebruiksmogelijkheid zeven van het schuifregister: uitschuiven onder supervisie van een externe klok.

9. De registers IFR en IER

Deze registers (zie figuur 30) hebben te maken met diverse interrupts van diverse oorsprong. Het interruptvlagregister (IFR) kan door de processor worden gelezen, maar ook beschreven. Een te resetten vlag wordt gereset door een schrijfoperatie, waarbij het desbetreffende databit nul is.

30



Figuur 30. De registers IFR en IER. Elke gesette vlag van IFR kan de $\overline{IRQ}$-leiding aktiveren, vooropgesteld dat het overeenkomstige bit van IER één is.

Bit Ø is de CA2-vlag. Deze vlag wordt geset indien er een negatieve of positieve nivoverandering op CA2 optreedt, waarbij CA2 via het register PCR als ingang is geprogrammeerd. Via PCR wordt eveneens bepaald of er op een negatieve danwel positieve nivoverandering wordt gereageerd. Bij bit Ø van het interrupt-vlagregister hoort bit Ø van het register IER (interrupt enable). Indien het enable-bit één (geset) is is het aktiveren van de $\overline{IRQ}$-leiding mogelijk (na het setten van de CA2-vlag). De CA2-vlag kan men resetten door te schrijven in of te lezen uit het uitgangsregister ORA. De $\overline{IRQ}$-leiding wordt dan eveneens weer op rustnivo (één) gebracht, vooropgesteld dat alle andere vlaggen van IFR gereset zijn. Indien men CA2 als onafhankelijke interruptingang heeft geprogrammeerd (independent interrupt, zie figuur 12) is het onmogelijk om de CA2-vlag door lezen

of schrijven te resetten. In dat geval moet men daartoe door schrijven in IFR CA2 één maken.

Bit 1 is de CA1-vlag. Deze vlag wordt geset indien er een negatieve of positieve nivoverandering op CA1 optreedt, waarbij CA1 via het register PCR als ingang is geprogrammeerd. Via PCR wordt eveneens bepaald of er op een negatieve danwel positieve nivoverandering wordt gereageerd. Bij dit bit hoort bit 1 van het interrupt-enable-register IER. Indien dit enable-bit één is en indien de CA1-vlag eveneens geset is wordt de $\overline{IRQ}$-lijn aktief. De CA1-vlag kan men resetten door te schrijven in of te lezen uit het uitgangsregister ORA. De $\overline{IRQ}$-leiding wordt dan eveneens weer op rustnivo (één) gebracht, vooropgesteld dat alle andere vlaggen van IFR gereset zijn.

Bit 2 van IFR is de schuifregistervlag. Deze vlag wordt geset zodra alle acht bits het schuifregister uitgeschoven zijn (met uitzondering van gebruiksmogelijkheid vier, zie hoofdstuk 8). Deze vlag wordt gereset door data in het schuifregister te schrijven of eruit te lezen.

Bit 3 is de CB2-vlag, bit 4 de CB1-vlag. Het verhaal is identiek aan dat van de vlaggen CA1 en CA2.

Bit 5 is de timer 2-vlag. Deze vlag wordt geset na een time out van timer 2. Indien de processor na de time out de onderste helft van teller-register 2, of indien de processor schrijft in de bovenste helft van teller-register 2, dan wordt deze vlag gereset. De $\overline{IRQ}$-leiding wordt dan eveneens weer op rustnivo (één) gebracht, vooropgesteld dat alle andere vlaggen van IFR gereset zijn. Bij deze vlag hoort bit 5 van het interrupt-vlagregister IFR. Indien dit laatste bit evenals de timer 2-vlag geset is wordt de $\overline{IRQ}$-lijn aktief.

Bit 6 is de timer 1-vlag. Hierbij hoort bit 6 van het register IER. Het verhaal is verder gelijk aan dat voor timer 2.

Bit 7 van IFR is géén interruptvlag! Dit bit laat zien of de $\overline{IRQ}$-lijn aktief is (nul; dan is bit 7 één) danwel in rust (één; dan is bit 7 nul). De waarde van bit 7 is het resultaat van de volgende logische funktie:
B7 = IFR6xIER6+IFR5xIER5+IFR4xIER4+IFR3xIER3+IFR2xIER2+IFR1xIER1+IFRØxIERØ, waarbij "x" de AND-funktie voorstelt en "+" de OR-funktie. Bit 7 is nul te krijgen door IER of IFR te resetten.

Bit 7 van het register IER heeft een bijzondere funktie. Indien de processor via schrijven dit bit Ø maakt worden automatisch de bits 6 . . . Ø van dit register gereset. Bij het (door de processor) lezen van IER is bit 7 altijd één. Bij het setten van bits in IER moet men data in IER schrijven, waarbij niet alleen de overeenkomstige bits, maar ook bit 7 één moet zijn.

Samenvatting IFR & IER. De bits van het register signaleren een nivoverandering op de ingangen CA1, CA2, CB1 en CB2, of een time out voor één of beide timers, of het afsluiten van acht opeenvolgende schuifopera-

ties. De bits van het bijbehorende register IER bepalen of een gesette interruptvlag ook nog de aktivering van de $\overline{IRQ}$-lijn tot gevolg heeft, dus of er al dan niet een interrupt moet komen. Indien er meer dan één vlag van het interrupt-vlaggenregister is geset is het voor de processor mogelijk om dit register te lezen; een programma kan vaststellen welke interrupts zijn opgetreden en deze interrupts vervolgens in een bepaalde volgorde (die samenhangt met de prioriteit) afwikkelen. Het is echter óók mogelijk om alle interrupts te verbieden. De $\overline{IRQ}$-leiding is dan altijd in rust. In dat geval kan de processor via polling de diverse bits van IFR onderzoeken en afhandelen, desgewenst eveneens in een bepaalde volgorde.