µP Automatische registeruitlezing

D. de Boer

Bij het ontwikkelen van programma's op de KIM 1 kunnen we gebruik maken van de single step mogelijkheid die de KIM heeft. Na elke druk op de 'GO' toets wordt er een instructie uitgevoerd.
Op het display staat dan het adres van de volgende uit te voeren instructie. Na elke stap kunnen we de inhoud van de verschillende registers controleren. Een nadeel is, dat voor elk register een adres ingevoerd moet worden, waardoor het controleren wat omslachtig wordt. In dit artikel wordt een programma besproken, dat er voor zorgt dat tijdens het indrukken van de 'GO' toets de inhoud van 3 geheugenplaatsen naar keuze, tegelijkertijd op het display verschijnen. Bij het loslaten van de toets verschijnt dan weer het adres van de volgende uit te voeren instructie.

De werking van de single-step

Bij de single-step op de KIM wordt gebruik gemaakt van de NMI (non maskable interrupt). Voor een goede werking is het daarom nodig het adres te definiëren waar de microprocessor na een interrupt heen moet springen. Voor de single-step (en de ST-toets) is dit adres 1C00. Op dit adres begint het monitor-programma met het kopiëren van de inwendige registers in het werkgeheugen. Hierdoor kunnen we de inhoud van deze registers controleren en eventueel veranderen. Na een druk op 'GO' wordt de (eventueel veranderde) inhoud van de registers weer terug gezet en vervolgt de microprocessor zijn oorspronkelijke programma. Wanneer het single-step schakelaartje omgezet wordt, wordt de 'SYNC' uitgang verbonden met de 'NMI' ingang van de microprocessor. De 'SYNC' uitgang geeft aan het begin van elke instructie een puls af, zodat tijdens elke instructie een interrupt ontstaat.

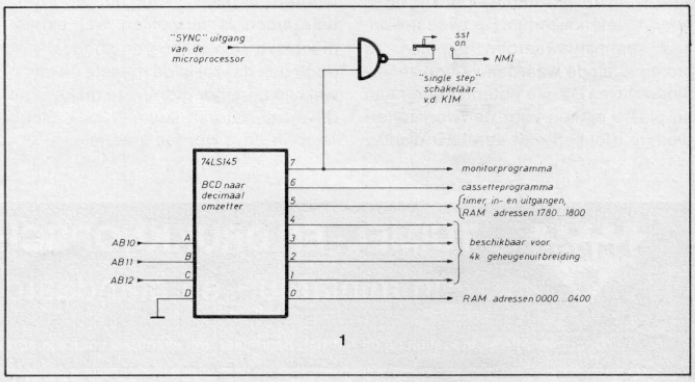
Wanneer we een programma starten met het single-step schakelaartje aan, dan zal tijdens de eerste instructie een interrupt ontstaan. Hierdoor springt de microprocessor naar het monitorprogramma. Pas wanneer we weer op 'GO' drukken wordt met de tweede instructie van het programma begonnen. Tijdens deze tweede instructie ontstaat echter weer een interrupt, waardoor weer naar het monitorprogramma wordt gesprongen. Op deze wijze wordt bereikt dat slechts één instructie tegelijk wordt uitgevoerd. Om een goede single-step werking te verkrijgen is echter nog iets nodig. In het bovenstaande verhaal zijn we er van uitgegaan dat de interrupts uitsluitend ontstaan bij de instructies van het te testen programma. In werkelijkheid ontstaat er (als hier geen maatregelen tegen genomen zouden zijn) tijdens elke instructie een interrupt, dus ook bij de instructies van het monitorprogramma. Het gevolg zou zijn dat na het eerste interrupt naar adres 1C00 gesprongen wordt, waar de instructie 'STA' staat.

De microprocessor gaat deze instructie uitvoeren, maar tijdens deze instructie ontstaat er weer een interrupt. Het gevolg is dat de microprocessor weer naar adres 1C00 springt, met weer een interrupt. Hierdoor zal het display donker blijven en de stack wordt door het grote aantal interrupten volledig vol geschreven. Daarom moet op een of andere manier worden voorkomen dat er interrupten gegenereerd worden als het monitorprogramma loopt. In afb. 1 is te zien hoe dit probleem bij de KIM 1 is opgelost.

Op de KIM 1 bevindt zich een decodering voor de eerste 8 K adresruimte. Deze decodering wordt gerealiseerd met de 74LS145, een BCD naar decimaal omzetter. Met uitgang 0 wordt de 1K RAM van de KIM geselecteerd, uitgangen 1, 2, 3 en 4 zijn beschikbaar voor 4K geheugenuitbreiding. Met uitgang 5 wordt het RAM gedeelte van de twee 6530 IC's (met timer en in- en uitgangen) geselecteerd.

Uitgang 6 selecteert het 1K cassetteprogramma, dat vast in ROM staat. Uitgang 7 ten slotte selecteert het monitorprogramma. Dit wil zeggen dat uitgang 7 laag wordt wanneer de microprocessor een adres in de 'monitor ruimte' selecteert. Dit signaal wordt nu

1 De werking van de single-step van de KIM. Wanneer het monitorprogramma is geselecteerd, kunnen er geen interrupten ontstaan.



1

samen met het 'SYNC' signaal aan een NAND-poort toegevoerd. Zolang uitgang 7 hoog is (en dus het monitorprogramma niet geselecteerd is) én het single-stepschakelaartje gesloten is, zullen de pulsen van de 'SYNC' uitgang interrupten veroorzaken. Wanneer het monitorprogramma geselecteerd wordt (na elk interrupt) zal uitgang 7 (afb. 1) laag worden, waardoor de uitgang van de NAND onvoorwaardelijk hoog blijft. Er ontstaan geen interrupten meer en de microprocessor kan nu rustig het monitorprogramma uitvoeren.
Hiermee is meteen het verschijnsel verklaard dat de subroutines van het monitorprogramma niet in single-step kunnen worden doorlopen. (Deze subroutines worden ook in zelf geschreven programma's nog wel eens aangeroepen.) Wanneer we in single-step naar zo'n subroutine springen, zal het eerste interrupt pas weer ontstaan tijdens de eerste instructie ná de subroutine. Hierdoor zal het programma pas weer worden onderbroken bij de tweede instructie na die subroutine. Ten onrechte wordt dan wel eens gedacht dat de microprocessor naar een verkeerd adres terug springt, wat beslist niet het geval is.

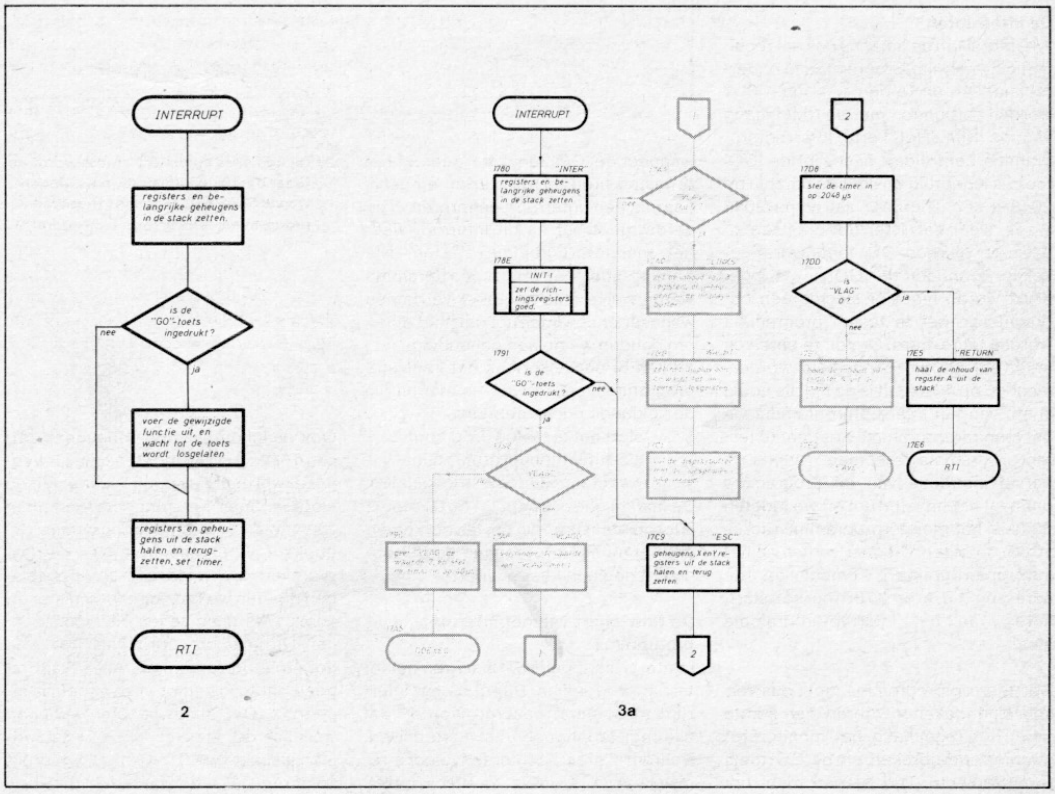
Hoe kan de single-step gewijzigd worden?

Nu we weten hoe de single-step gerealiseerd is, kunnen we gaan denken hoe we de werking kunnen wijzigen. We willen hierbij beslist niet gaan solderen en wijzigen op de print van de KIM. Ten eerste omdat dan het gevaar bestaat dat de KIM voorgoed sneuvelt, ten tweede om te voorkomen dat er verschillende KIM versies komen. Het alternatief is het maken van een tweede single-step, waarbij de interruptpulsen met een speciaal programma worden opgewekt. Dit programma moet dan niet groter zijn dan 103 bytes, zodat het volledig past in het weinig gebruikte RAM geheugen op de adressen 1780 . . . 17E6.

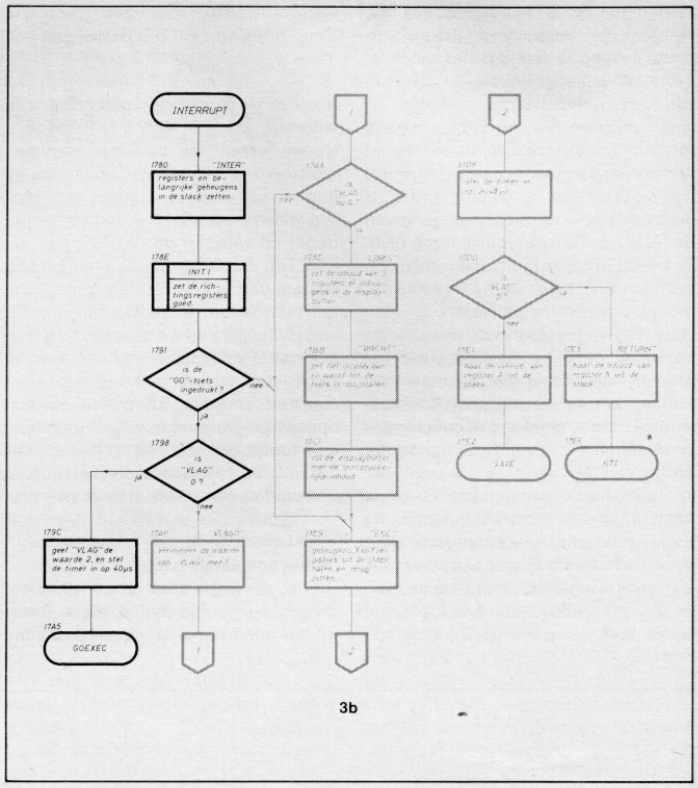
De tweede single-step

Omdat de single-step op de KIM zich niet zo eenvoudig laat wijzigen gaan we dus een tweede single-step maken. Normaal zal, als we op 'GO' drukken, een programma geheel uitgevoerd worden. (Het single-stepschakelaartje staat nu uit.) Als we nu met een druk op de 'GO' toets ook de timer van de KIM starten, kunnen we zorgen dat er tijdens de eerste instructie van het programma een interrupt ontstaat. Op deze manier zal er dan maar één instructie uitgevoerd worden. We zitten echter met een klein probleem: het monitorprogramma, dat decodeert welke toets is ingedrukt en de bijbehorende opdracht uitvoert, start bij de toets 'GO' geen timer!!

2 Het principe om de functie van een bepaalde toets (hier GO) te wijzigen.
3a Zolang de 'GO'-toets niet is ingedrukt, wordt er door het interruptprogramma geen actie ondernomen.



Het is dus zaak de functie van de 'GO' toets te wijzigen. Hoe dit gedaan kan worden zien we in afb. 2. We zorgen dat het monitorprogramma regelmatig wordt onderbroken d.m.v. een interrupt. Dit korte interruptprogramma controleert dan snel of de 'GO' toets is ingedrukt. Wanneer dit niet het geval is, gaan we gewoon terug naar het monitorprogramma. Hierbij zorgt de timer dat regelmatig interrupten worden opgewekt. Zodra de 'GO' toets is ingedrukt, zal (volgens afb. 2) het interruptprogramma de gewijzigde functie uitvoeren. Natuurlijk mag het monitorprogramma geen actie ondernemen bij het indrukken van 'GO'. Daarom zal de microprocessor het interruptprogramma pas mogen verlaten zodra de toets is losgelaten. Het monitorprogramma zal nu geen toetsdruk meer constateren, waardoor eventuele eerdere waarnemingen teniet gedaan worden. Wel moeten we zorgen dat de interrupten elkaar snel genoeg opvolgen, want als het monitorprogramma eenmaal bezig is de 'GO' opdracht uit te voeren, dan zal hij hier ook na het interrupt mee doorgaan.



3b

3b Bij een druk op de 'GO'-toets wordt de timer op 40 µs ingesteld, en springen we via 'GOEXEC' naar het te testen programma. 'VLAG' was 0, en wordt 2.

De interrupten

We gebruiken de timer nu dus om regelmatig het monitorprogramma te onderbreken. Om dit te bereiken moet PB7 worden verbonden met de NMI ingang van de KIM. Het korte interruptprogramma controleert alleen of de 'GO' toets is ingedrukt en stelt de timer in op 2048 µ sec. Hierdoor zal er na 2048 µ sec. weer een interrupt ontstaan zo dat weer naar de 'GO' toets wordt gekeken. Wanneer de 'GO' toets ingedrukt wordt, moet er slechts één opdracht van het te testen programma worden uitgevoerd. Om dit te bereiken moet ook de timer ingesteld worden, maar nu op 40 µs. Dit is de tijd die nodig is om van het interruptprogramma via het monitorprogramma naar het te testen programma te komen. Tijdens de eerste instructie van dit programma ontstaat er een interrupt en we moeten nu naar het monitorprogramma (adres 1C00) springen. Dit gaat echter via het interruptprogramma, omdat op de adressen 17FA en 17FB nu het startadres van het interruptprogramma staat.

Het interruptprogramma moet dus verschillende taken uitvoeren. Ten eerste moet het regelmatig het monitorprogramma onderbreken, om de 'GO' toets te controleren. Ten tweede moet het wanneer de 'GO' toets is ingedrukt het te testen programma starten en gelijk daarna een interrupt genereren. Ten derde moet het na dit interrupt weer het monitorprogramma starten. Natuurlijk moet het interruptprogramma weten welke taak het moet verrichten wanneer er een interrupt ontstaat. Daarom houden we op een geheugenplaats bij (00F6) bij welke taak het interruptprogramma moet uitvoeren. Hierbij zijn de volgende codes gebruikt:

0 . . . . start het te testen programma
2 . . . . start het monitorprogramma
1 . . . . wacht tot GO-toets is losgelaten.

De laatste code (wacht tot de GO-toets is losgelaten) is nodig om te voorkomen dat het monitorprogramma de GO-opdracht nog een keer uitvoert.

De flow-chart van het interruptprogramma

In afb. 3 staat de flow-chart van het interruptprogramma. Direct na het interrupt moet eerst gezorgd worden dat alle registerinhouden bewaard blijven. Daarom worden achtereenvolgens registers A, X en Y in de stack gezet. Ook de inhoud van de geheugenplaatsen 1740 . . . 1743 moeten tijdelijk weggezet worden. Deze geheugenplaatsen worden door het monitorprogramma zowel als het interruptprogramma gebruikt voor het toetsenbord en display. Wanneer alle gegevens zijn opgeslagen moeten we het richtingsregister op adres 1741 goed zetten. Hiervoor springen we naar de subroutine 'INIT' van het monitorprogramma. Alles staat nu goed om te kijken of er een toets is ingedrukt. Om dit te bepalen springen we naar de subroutine 'GETKEY'. Alleen wanneer de GO-toets is ingedrukt krijgt register A de waarde $ 13.

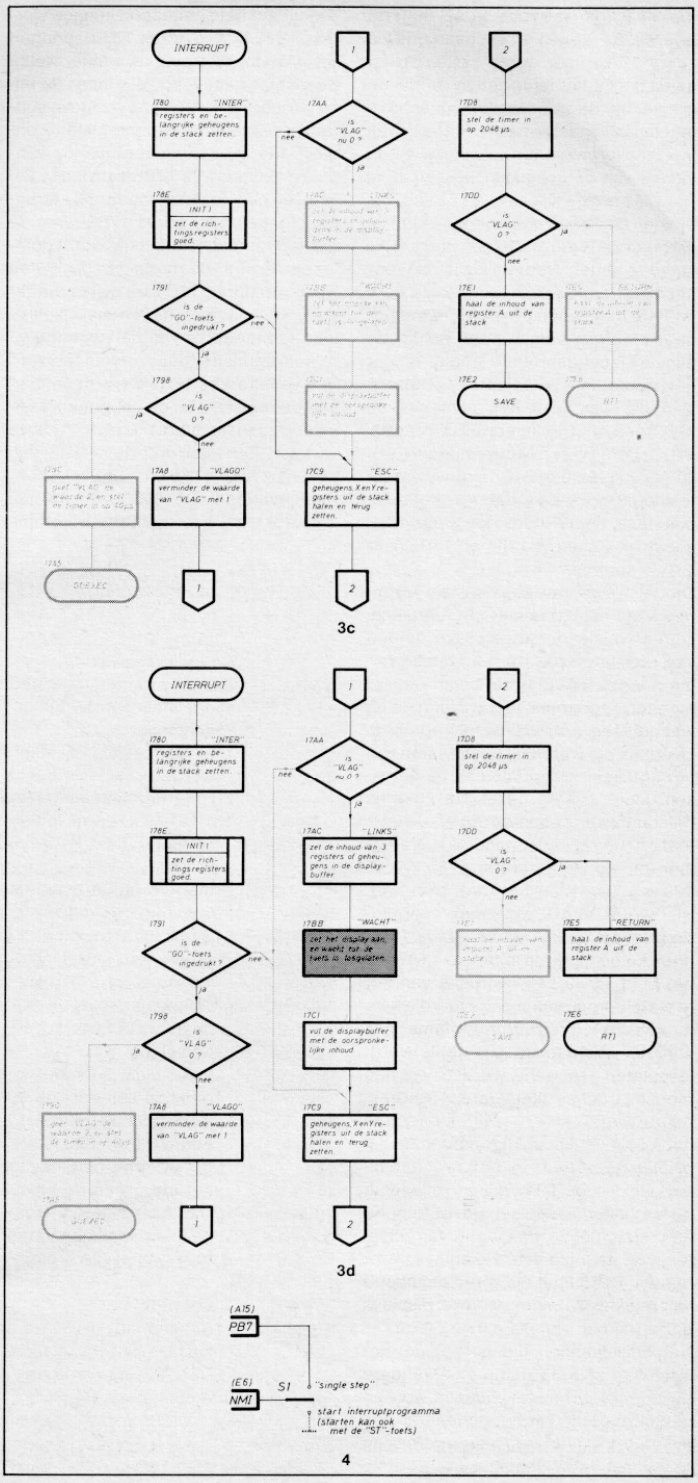
Als de GO-toets niet is ingedrukt (afb. 3a) worden de registers weer met hun oorspronkelijke inhoud gevuld, de timer wordt ingesteld en via RTI gaan we weer verder met het monitorprogramma, alsof er niets is gebeurd.

De timer zorgt er dan voor dat na 2,048 ms weer een interrupt ontstaat. De inhoud van 'VLAG' (adres 00F6) heeft nu de waarde 0. Zodra de GO-toets ingedrukt wordt (afb. 3b), gaan we via adressen 1794 en 1798 naar adres 179C. Hier wordt 'VLAG' op 2 gezet, zodat bij een volgend interrupt niet opnieuw deze weg wordt ingeslagen.

Vervolgens wordt de timer op $28 µs ingesteld (40 µs decimaal) en via JMP-instructie komt de microprocessor bij 'GOEXEC' adres IDC8. Het programmadeel 'GOEXEC' zorgt dat de registers van de microprocessor weer gevuld worden met de waarden die bij het te testen programma horen. Deze waarden waren eerder door het monitorprogramma op de adressen 00EF . . . 00F5 gezet. Ook de stackpointer wordt weer in de goede stand gezet. Via een RTI springt de microprocessor nu naar het te testen programma. Meteen zorgt de timer voor een interrupt en het interruptprogramma wordt opnieuw gestart. Nu heeft 'VLAG' echter de waarde '2'.

Het interruptprogramma loopt nu als volgt (afb. 3c). Via adressen 1780, 178E, 1791, 1794, 1798 (de GO-toets is nog steeds ingedrukt) naar 17A8 ('VLAG' = 2). Op dit adres wordt de inhoud van 'VLAG' met 1 verminderd, en wordt dus '01'.

Dat we dit doen heeft een aantal voordelen; ten eerste heeft 'VLAG' nu met een korte instructie zijn nieuwe waarde gekregen, ten tweede kunnen we straks



3c Na de eerste instructie van het te testen programma springen we weer naar het interruptprogramma. Nu gaan we naar 'SAVE', waar de inwendige registers van de microprocessor worden gesaved. 'VLAG' was 2, en wordt 1.

3d De laatste cyclus. Als alle registers gesaved zijn, wordt het monitorprogramma weer onderbroken. Nu blijft het interruptprogramma wachten tot de 'GO'-toets is losgelaten. 'VLAG' was 1, en wordt 0.

4 De aansluiting van de nieuwe single-stepschakelaar.

met dezelfde instructie 'VLAG' nog de waarde '01' geven en we gaan dus via adres 17AA naar adres 17C9. Op dit adres begint het terug zetten van de registers en de geheugeninhouden. We moeten uiteindelijk naar het begin van het monitorprogramma springen en de inhoud van de geheugenplaatsen 1740 . . . 1743 is eigenlijk niet van belang. We moeten dit deel echter doorlopen om de stackpointer weer in de oorspronkelijke stand te zetten. Natuurlijk zijn hiervoor ook andere mogelijkheden, zoals 'LDX en TXS' of gewoon 10xPLA. De methode die we hier volgen heeft echter de minste geheugenruimte nodig, omdat het programma vanaf adres 17C9 toch al nodig was voor het geval dat de 'GO'-toets niet is ingedrukt. Op adres 17DD controleren we de inhoud van 'VLAG' om te bepalen hoe we het interruptprogramma moeten verlaten. In ons geval heeft 'VLAG' de waarde 01, zodat we via adres 17E1 en 17E2 naar 'SAVE' springen.

Omdat we bij het bepalen van de inhoud van 'VLAG' register A gebruiken, zetten we de oorspronkelijke inhoud van register A pas op het laatste moment terug. 'SAVE' is het begin van het monitorprogramma, hier worden de inwendige registers van de microprocessor zoals die waren bij het verlaten van het te testen programma op de adressen 00EF . . . 00F5 gezet. Na 2048 µs ontstaat weer een interrupt en de GO-toets is nog steeds ingedrukt. We komen nu via de adressen 1780, 178E, 1791, 1794, 1798 bij 17A8 (afb. 3d). Hier wordt 'VLAG' weer met 1 verminderd, zodat de inhoud nu '00' is. Hierdoor komen we via 17AA naar 17AC. We hebben nu één instructie van het te testen programma uitgevoerd en we mogen het interruptprogramma pas verlaten wanneer de 'GO'-toets wordt losgelaten. (Anders worden er nog meer instructies uitgevoerd.) De adresruimte van 17AC . . . 17C0 is nu vrij te gebruiken om bepaalde waarden op het display te zetten. In het programma hebben we de inhoud van register A op de linker 2 displays gezet (inhoud 00F3 naar 00FB), de inhoud van register X op de middelste 2 displays gezet (inhoud 00F5 naar 00FA) en de inhoud van register Y op de rechter 2 displays gezet (inhoud van 00F4 naar 00F9).

Natuurlijk kunnen tijdens het controleren van een programma ook andere belangrijke geheugenplaatsen worden gecontroleerd. Om het geheel flexibel te houden is geen zero page addressing toegepast, zodat gemakkelijk de adressen uit het hele geheugen kunnen worden ingetypt. Op adres 17BB springen we naar de display subroutine, welke de displays activeert. Wanneer de microprocessor terugkomt van deze subroutine is de inhoud van register A ongelijk aan '00' zolang er een toets is ingedrukt. Op adres 17BE wordt dus zolang er een toets is ingedrukt, teruggesprongen naar adres 17BB waar de display subroutine opnieuw wordt doorlopen. Zodra de toets is losgelaten wordt op adres 17C1 de displaybuffer weer gevuld met de oorspronkelijke inhoud (het adres van de volgende uit te voeren instructie). Vervolgens worden vanaf adres 17C9 weer de inhoud van de registers en de geheugenplaatsen teruggezet. 'VLAG' is nu '00' en we verlaten het interruptprogramma dan ook via 17DD, 17E5 en 17E6.

Samenvattend: De 'GO'-toets is niet ingedrukt en het monitorprogramma loopt, zodat het startadres van het te testen programma op het display staat. Regelmatig wordt het monitorprogramma onderbroken door het interruptprogramma. De geheugenplaats 'VLAG' heeft de waarde '00' (afb. 3a). Zodra de 'GO'-toets wordt ingedrukt zal na het eerstvolgende interrupt het interruptprogramma 'VLAG' op '02' zetten en onafhankelijk van de plaats waar het monitorprogramma werd afgebroken naar 'GOEXEC' adres 1DC8 van het monitorprogramma springen (afb. 3b). Dit stukje van het monitorprogramma start het te testen programma. Bij de eerste instructie van dit programma ontstaat een interrupt, en we gaan weer naar het interruptprogramma. Nu wordt de 'VLAG' op '01' gezet en het monitorprogramma wordt bij 'SAVE' gestart (afb. 3c). Bij het volgende interrupt geeft het interruptprogramma de geheugenplaats 'VLAG' de waarde '00'

Gebruiksaanwijzing

Om het programma te gebruiken moeten we de volgende handelingen verrichten.

a. eenmalig:

1 Er moet een schakelaar tussen PB7 en NMI gesoldeerd worden volgens afb. 4.

2 Het programma moet ingetypt worden op de adressen 1780 . . . 17E6, of, indien deze geheugenruimte voor een ander programma gebruikt is op een willekeurige andere plaats in het geheugen (natuurlijk mag het programma niet de gereserveerde ruimten van het geheugen overlappen).

3 De NMI vector moet ingevoerd worden. Wanneer het programma wordt ingetypt vanaf adres 1780, zetten we '80' op adres 17FA, en '17' op adres 17FB.

4 Geheugenplaats 'VLAG' moet goed gezet worden ('00' op adres 00F6).

b. bediening:

Schakelaar S1 moet in de stand 'Normaal' staan. We drukken een keer op 'ST' en het programma is gestart. Zolang S1 op 'normaal' staat merken we hier niets van. Zodra S1 op 'Single step' gezet wordt zal bij elke druk op 'GO' een instructie uitgevoerd worden, terwijl tijdens het indrukken de inhoud van resp. register A, X en Y zichtbaar worden. Ook andere registers en/of geheugenplaatsen kunnen met een kleine verandering zichtbaar gemaakt worden.

Let op!!

Bij gebruik van deze tweede 'single-step' moet het single-stepschakelaartje van de 'KIM' uit staan!!

De 'ST' toets werkt niet, stoppen kan alleen met de 'RS' toets.

en wacht tot de GO-toets wordt losgelaten. In deze tijd kunnen willekeurige geheugenplaatsen zichtbaar gemaakt worden (afb. 3d). Wanneer de 'GO'-toets wordt losgelaten, wordt het monitorprogramma vervolgd waar het onderbroken was. Vanaf dit moment start het hele proces weer opnieuw.

Het starten van de interrupten

Het interruptprogramma wekt steeds zelf de interrupten op. Eénmaal moeten de interrupten echter beginnen. Op het eerste gezicht zou je zeggen dat een druk op de 'ST'-toets voldoende is, met 'ST' geven we immers een NMI. Een druk op 'ST' zorgt inderdaad dat naar het interruptprogramma wordt gesprongen en dus dat de timer gestart wordt. Als echter na 2048 µs de timer een interrupt geeft, is de 'ST'-toets nog steeds ingedrukt en dus de NMI nog laag. Hierdoor zal het interrupt van de timer verloren gaan en wordt niet opnieuw naar het interruptprogramma gesprongen. Daarom moet schakelaar S1 (afb. 4) de nieuwe single-stepschakelaar op 'uit' staan. Wanneer we nu op 'ST' drukken zal inderdaad het interruptprogramma de timer starten, zodat PB7 laag wordt. Wanneer we nu éérst de ST-toets loslaten en daarna S1 op single-step zetten, zal er weer een interrupt ontstaan (PB7 was laag). Met dit interrupt zal weer het interruptprogramma gestart worden, zodat van nu af de interrupten automatisch gegenereerd worden. (Het proces kan ook gestart worden door alleen adres 170E in te typen.)

Verdere mogelijkheden

Door veranderingen tussen de geheugenplaatsen 17AC en 17C9 kunnen we b.v. om de sec. automatisch een programmastop laten uitvoeren. Ook zou het mogelijk zijn het programma stopadres te controleren en alleen bij bepaalde geheugenplaatsen te stoppen. (Dit is handig bij loops.) Een andere handige functie is een '−' toets, die de tegenovergestelde functie verricht van de '+' toets.

We zouden hiervoor de 'PC' toets kunnen gebruiken. Al deze functies gelijk lijkt aantrekkelijk, maar zou voor de basisversie van de KIM te veel geheugen gebruiken. Het interruptprogramma is immers een hulpprogramma voor het ontwikkelen van veel grotere programma's in het overblijvende geheugen. Niettemin kan het interessant zijn deze mogelijkheden eens te proberen te realiseren.

Lijst 1

1780	48	INTER	PHA-*impl*		
1781	8A		TXA-*impl*		
1782	48		PHA-*impl*		Zet de inhoud van de
1783	98		TYA-*impl*		registers in de stack.
1784	48		PHA-*impl*		
1785	A2 04		LDX-*imm*	$04	
1787	BD 3F 17	SAVE	LDA-*abs, X*	$173F	
178A	48		PHA-*impl*		De inhoud van 4
178B	CA		DEX-*impl*		geheugens in de stack.
178C	D0 F9		BNE-*rel*	SAVE	
178E	20 8C 1E		ISR-*abs*	INIT1	richtingreg. goed zetten.
1791	20 6A 1F		ISR-*abs*	GETKEY	Is 'GO' ingedrukt?
1794	C9 13		CMP-*imm*	$13	Zo nee, naar ESC.
1796	D0 31		BNE-*rel*	ESC	
1798	A5 F6		LDA-*Z page*	VLAG	Indien VLAG=0
179A	D0 0C		BNE-*rel*	VLAG0	naar VLAG0
179C	A9 02		LDA-*imm*	$02	VLAG op 2
179E	85 F6		STA-*Z page*	VLAG	
17A0	A9 28		LDA-*imm*	$28	Timer op 40 µs
17A2	8D 0C 17		STA-*abs*	TIM1T	(28 hex).
17A5	4C C8 1D		JMP-*abs*	GOEXEC	Naar monitor.
17A8	C6 F6	VLAG0	DEC-*Z page*	VLAG	Verminder VLAG,
17AA	D0 1D		BNE-*rel*	ESC	bij 0 naar ESC.
17AC	AD **F3 00**	LINKS	LDA-*abs*	ACC	Reg. A op de 2
17AF	85 FB		STA-*Z page*	POINTH	linker displays.
17B1	AD **F5 00**	MIDDEN	LDA-*abs*	XREG	Reg. X op de 2
17B4	85 FA		STA-*Z page*	POINTL	middelste displays.
17B6	AD **F4 00**	RECHTS	LDA-*abs*	YREG	Reg. Y op de 2
17B9	85 F9		STA-*Z page*	INH	rechter displays.
17BB	20 1F 1F	WACHT	JSR-*abs*	SCANDS	Display aan, en
17BE	D0 FB		BNE-*rel*	WACHT	wacht tot GO
17C0	EA		NOP-*impl*		is losgelaten.
17C1	A5 EF		LDA-*Z page*	PCL	
17C3	85 FA		STA-*Z page*	POINTL	Vul de displaybuffer
17C5	A5 F0		LDA-*Z page*	PCH	met de oorspr. inhoud.
17C7	85 FB		STA-*Z page*	POINTH	
17C9	A2 00	ESC	LDX-*imm*	$00	
17CB	68	TERUG	PLA-*impl*		
17CC	9D 40 17		STA-*abs, X*	$1740	Inhoud van 4 geheugens
17CF	E8		DEX-*impl*		uit de stack.
17D0	E0 04		CPX-*imm*	$04	
17D2	D0 F7		BNE-*rel*	TERUG	
17D4	68		PLA-*impl*		
17D5	A8		TAY-*impl*		Inhoud van X en Y
17D6	68		PLA-*impl*		uit de stack.
17D7	AA		TAX-*impl*		
17D8	A9 02		LDA-*imm*	$02	Timer op 2048 µs.
17DA	8D 0F 17		STA-*abs*	TIMKT	
17DD	A5 F6		LDA-*Z page*	VLAG	Indien VLAG=0
17DF	F0 04		BEQ-*rel*	RETURN	naar RTI.
17E1	68		PLA-*impl*		A uit de stack
17E2	4C 00 1C		JMP-*abs*	SAVE	naar monitor.
17E5	68	RETURN	PLA-*impl*		A uit de stack naar het
17E6	40		RTI-*impl*		onderbroken programma.

Het interruptprogramma zoals hier gepubliceerd kan op elke plek in het geheugen worden gezet, mits het startadres op 17FA en 17FB wordt gezet.

Lijst 1. Het interruptprogramma. De vetgedrukte getallen zijn de adressen van de zichtbaar gemaakte geheugenplaatsen en kunnen naar behoefte worden gewijzigd.