ASSEMBLY
PROGRAMMING
8080/8085
en INTERFACING

AP map 3.

INTERFACING-2.

Interfacing-2

1. INLEIDING

In de les "Interfacing-1" is ingegaan op geprogrammeerde I/O en interrupt I/O. In de les "Programmeerbare chips" is het verschijnsel strobed I/O aan de orde gekomen. Strobed I/O kan op basis van zowel geprogrammeerde I/O als interrupt I/O plaatsvinden.

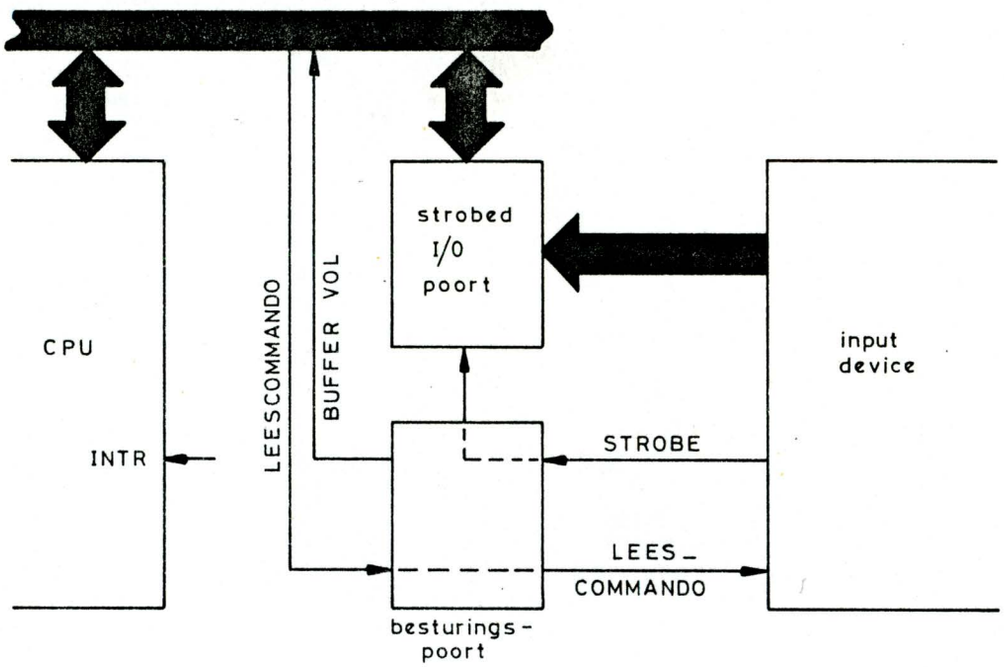


fig.1

In fig.1 is geprogrammeerde strobed input weergegeven. Dit verloopt als volgt:

1. De CPU geeft d.m.v. een output-instructie een actief leescommando. D.m.v. input-instructies wordt steeds het BUFFER VOL-signaal getest.
2. M.b.v. een STROBE-impuls klokt het input-apparaat data in de input-buffer. De besturingspoort maakt het BUFFER VOL-signaal actief. Dit zal even later door de CPU worden gedetecteerd.
3. D.m.v. een input-instructie haalt de CPU data uit de input-buffer. De input-actie is dan ten einde.

Fig.1 is eenvoudig te veranderen in een systeem met strobed I/O op interrupt basis. De BUFFER VOL-lijn moet dan met een interrupt-ingang van de CPU worden verbonden. Het BUFFER VOL-signaal behoeft nu niet d.m.v. input-instructies via de databus naar de CPU te worden getransporteerd. Na het geven van een leescommando kan de CPU doorgaan met de uitvoering van het hoofdprogramma. Wanneer het input-apparaat data in de input-buffer heeft geklokt, wordt de programma-uitvoering onderbroken door een BUFFER VOL-signaal. Deze functioneert nu nl. als interrupt request.

Door toevoeging van deze strobed I/O, kunnen we de geprogrammeerde I/O-methoden nu onderverdelen in drie groepen, nl.
a. geprogrammeerde I/O zonder acknowledge,
b. geprogrammeerde strobed I/O,
c. geprogrammeerde I/O met handshake.

Geprogrammeerde I/O zonder acknowledge wordt ook wel aangeduid met directe geprogrammeerde I/O of met simple I/O (= eenvoudige I/O).

In deze les behandelen we de toepassing van geprogrammeerde I/O zonder acknowledge, bij het aftasten van een toetsenbord en het aansturen van 7-segment displays. Aansluitend behandelen we de keyboard display controller 8279. Deze programmeerbare chip kan nl. de communicatie tussen CPU, toetsenbord en 7-segment displays besturen.

SAMENVATTING 1

1. Strobed I/O kan op basis van zowel geprogrammeerde I/O als interrupt I/O plaatsvinden.

2. Bij geprogrammeerde strobed I/O moet de CPU na het geven van een schrijf- resp. leescommando de BUFFER VOL-lijn continu testen.

3. Bij strobed I/O op basis van interrupt veroorzaakt een actief BUFFER VOL-signaal een interrupt request. De CPU kan na het geven van een schrijf- resp. leescommando, doorgaan met de uitvoering van het hoofdprogramma.

4. Geprogrammeerde I/O is te verdelen in
   a. I/O zonder acknowledge,
   b. strobed I/O,
   c. I/O met handshake.

2. DIRECTE GEPROGRAMMEERDE I/O

Directe geprogrammeerde I/O leent zich alleen goed voor I/O devices, waarbij sprake is van eenvoudige aan/uit-functies. Een goed voorbeeld van deze vorm van interfacing zijn de schakelaars en de LED's op de uitbreidingsprint van de SDK 85.

Het aansluitschema voor een systeem met directe geprogrammeerde I/O is in fig.2 weergegeven. Het I/O-proces verloopt geheel zonder het opwekken en testen van statussignalen. Er wordt niet getest of een output device data kan ontvangen of dat een input device relevante (= geldige) data klaar heeft.

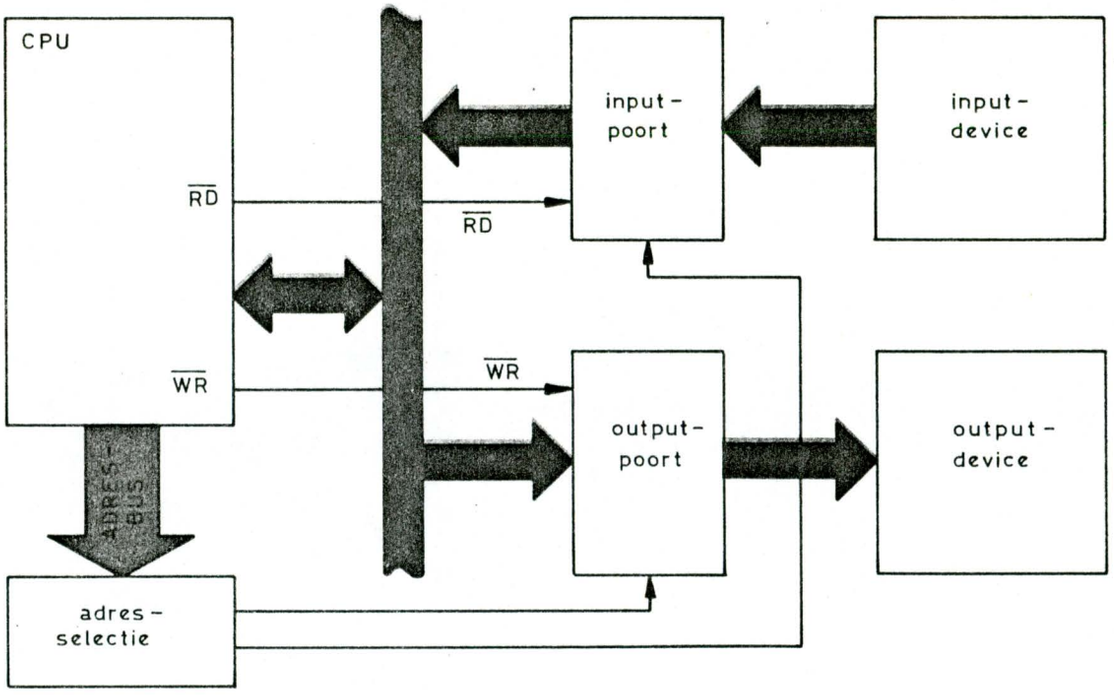


fig.2

Populair gezegd: de data wordt plompverloren op de output-lijnen geplaatst of van de input-lijnen ingelezen. Dit betekent, dat we ons bij de keuze van een bepaald I/O device en directe geprogrammeerde I/O af moeten vragen of dit device ook werkelijk geschikt is voor deze I/O-methode.

Het zal u duidelijk zijn, dat b.v. een ponsbandponser of een floppy disk een totaal andere vorm van interfacing vereisen.

Directe geprogrammeerde I/O is wel bijzonder geschikt voor b.v. het uitlezen van schakelaars en het aansturen van lampen, relais, motoren, kleppen, enz.

We zullen nu een voorbeeld uitwerken van directe geprogrammeerde I/O in een toepassing met een hexadecimaal toetsenbord.

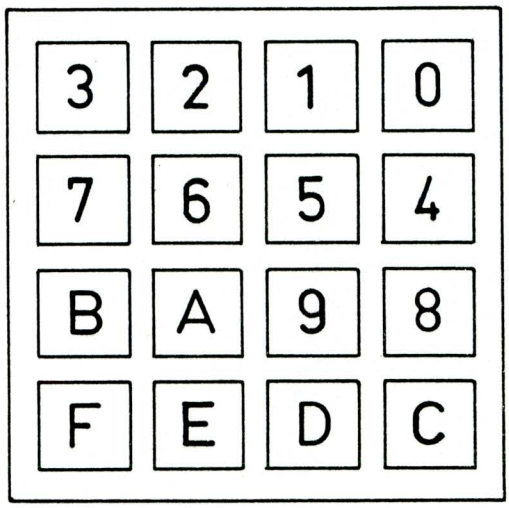


fig.3

Het hexadecimale toetsenbord bestaat uit 16 toetsen. Elke toets is een maakcontact, d.w.z. dat het contact wordt gesloten als de toets is ingedrukt.

We willen nu onder programmabesturing laten bepalen of er een toets, en zo ja welke toets, is ingedrukt. Een oplossing voor dit probleem zou kunnen zijn, dat voor elke toets een input-lijn wordt gebruikt.

Vraag 1: Er zijn dan ..... input-poorten nodig. Voor een keyboard met 48 toetsen zouden dan ..... input-poorten zijn vereist.

Voor een hexadecimaal toetsenbord zijn dan 16 input-lijnen, dus 2 input-poorten, nodig. (In deze les gaan we uit van een 8-bits microcomputer.)

Voor een toetsenbord met 48 toetsen (een ASCII-keyboard heeft er vaak nog meer !) zouden dan 48 input-lijnen, dus 6 input-poorten, nodig zijn. Het gebruik van een input-lijn voor elke toets is dus niet de meest efficiënte methode. Daarom wordt vaak in dit soort applicaties het z.g. scannen van het toetsenbord toegepast. Hierop wordt in de volgende paragraaf nader ingegaan.

SAMENVATTING 2

5. Bij directe geprogrammeerde I/O of simple I/O worden geen statussignalen getest.

6. Directe geprogrammeerde I/O is bijzonder geschikt voor het testen van schakelaars, kleppen, relais, enz. en het aansturen van lampen, LED's, relais, enz.

Antw.1: 2; 6.

3. SCANNED KEYBOARD

De contacten van een toetsenbord worden meestal als een matrix met elkaar verbonden. D.w.z. ze worden verdeeld in rijen en kolommen en zodanig met elkaar verbonden, dat elke toets deel uit maakt van slechts één rij en één kolom. Voor het hexadecimale toetsenbord uit fig.3 kunnen de contacten volgens de matrix in fig.4 worden verbonden.

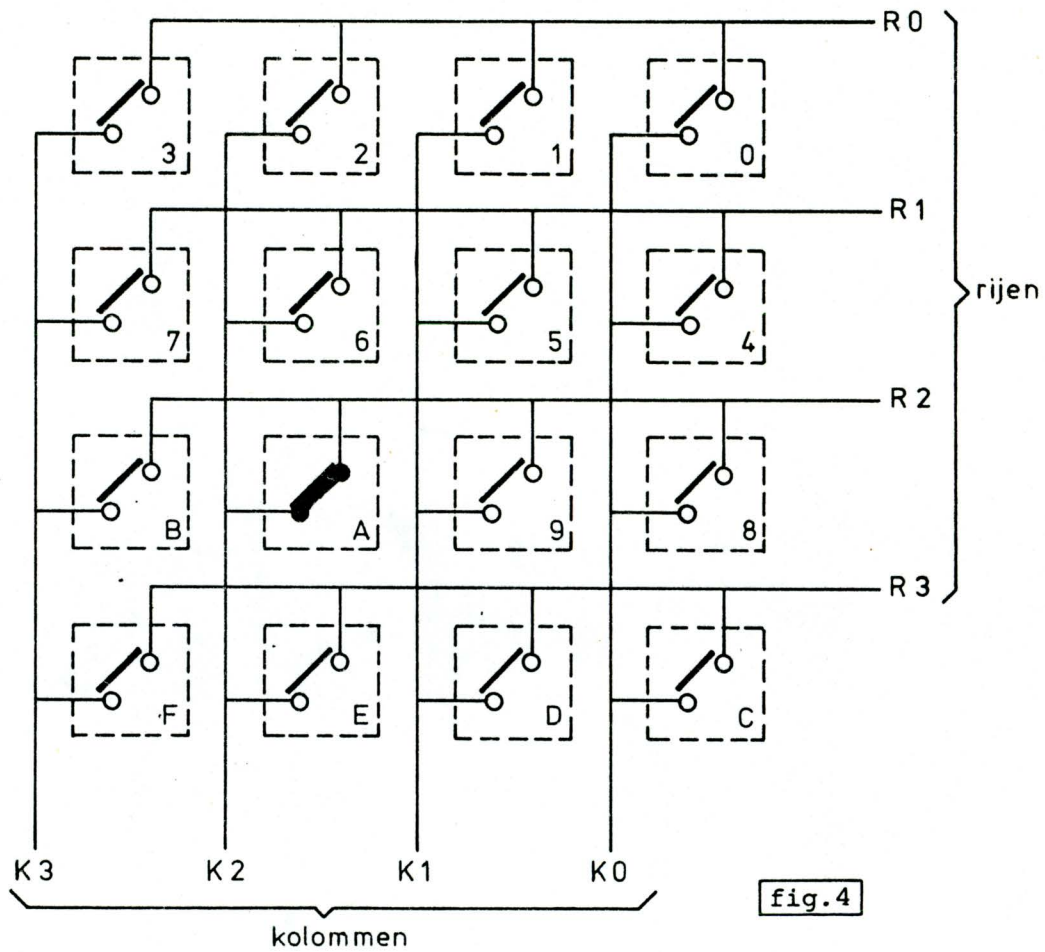


fig.4

Vraag 2: In fig.4 wordt de toets ..... ingedrukt.

Vooral in schema's met grote toetsenborden, levert de tekenwijze van fig.4 veel werk op. Daarom wordt in de praktijk een eenvoudiger tekenwijze toegepast.
In fig.5 is dit gedaan voor het hexadecimale toetsenbord. Duidelijk is te zien, dat ook nu alleen toets A is ingedrukt.

In fig.5 is tevens aangegeven, hoe dit toetsenbord op b_0 t/m b_3 van een output-poort (de rijen) en op b_0 t/m b_3 van een input-poort (de kolommen) is aangesloten.

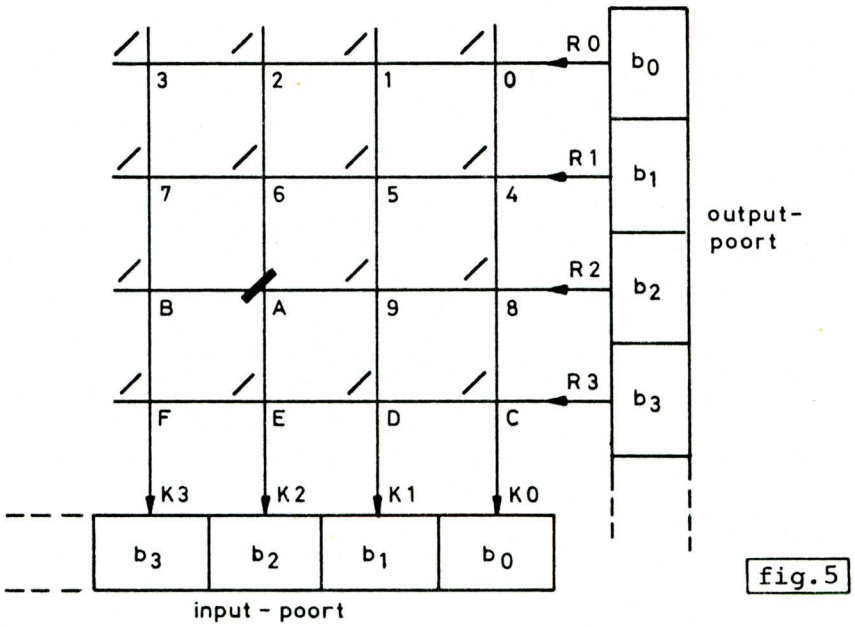


fig.5

Het scannen (= aftasten) van het toetsenbord gebeurt nu door via de output-poort op één van de rijlijnen een 1 te zetten. De overige 3 lijnen moeten dan 0 zijn. De toestanden van de kolomlijnen kunnen nu via de input-poort worden ingelezen. We gaan er voorlopig van uit, dat een open kolomlijn, dus een kolom waarin geen enkele toets is ingedrukt, als een 0 door de input-poort wordt gedetecteerd.

Als na het inlezen van de kolomlijnen b_3 t/m b_0 van de input-poort alle 0 zijn, betekent dit dat geen van de vier contacten in de met 1 aangestuurde rijlijn is gesloten.
We kunnen nu op een volgende rijlijn een 1 plaatsen (de vorige moet dan weer 0 worden) en opnieuw de kolomlijnen testen.

Op deze wijze wordt achtereenvolgens op elk der rijlijnen een 1 geplaatst. Dit wordt wel het principe van de "lopende 1" genoemd.
In de figuren 6a t/m 6d is weergegeven wat achtereenvolgens de output- en input-combinaties zijn, als het toetsenbord zo systematisch wordt afgetast gedurende de tijd dat toets A is ingedrukt.

Bij het indrukken van toets A wordt een doorverbinding tot stand gebracht tussen rij 2 en kolom 2. Bij scannen van de rijen 0, 1 en 3 is er geen enkele verbinding met de kolommen. In de figuren 6a, 6b en 6d wordt dan ook steeds de combinatie 0000 ingelezen. Bij het scannen van rij 2 zal er een 1 via het met pijltjes aangegeven pad (fig.6c) op de input-poort terechtkomen.
Welke toets is ingedrukt, kan nu software-matig worden afgeleid uit de rij- en kolom-informatie.

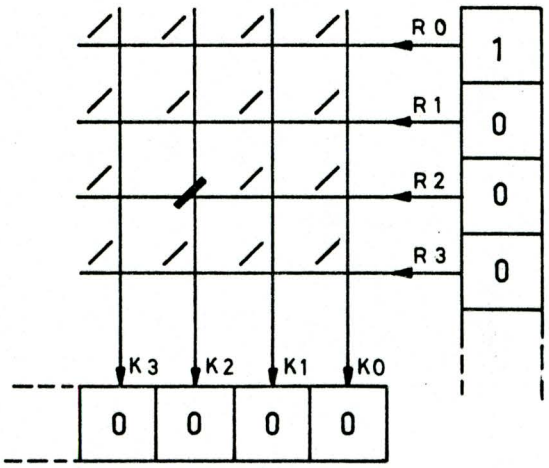


fig.6a

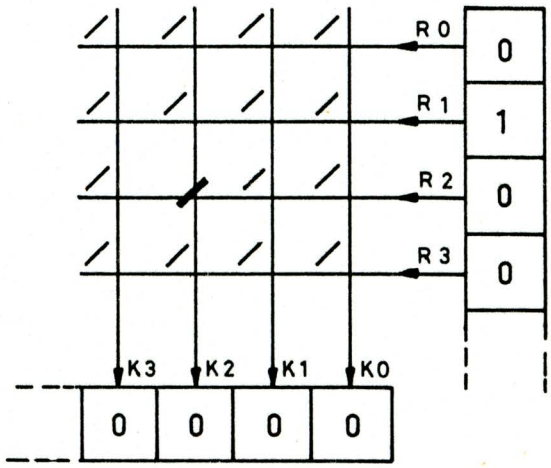


fig.6b

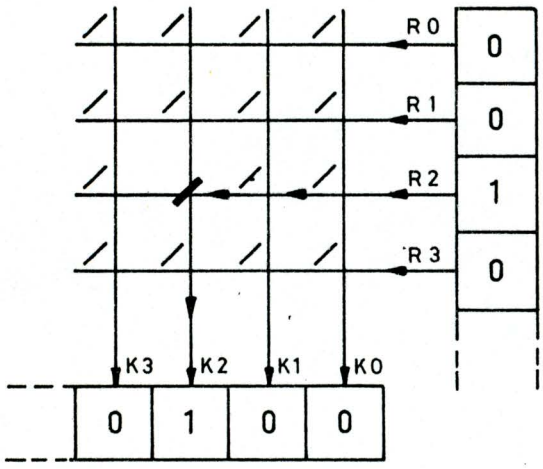


fig.6c

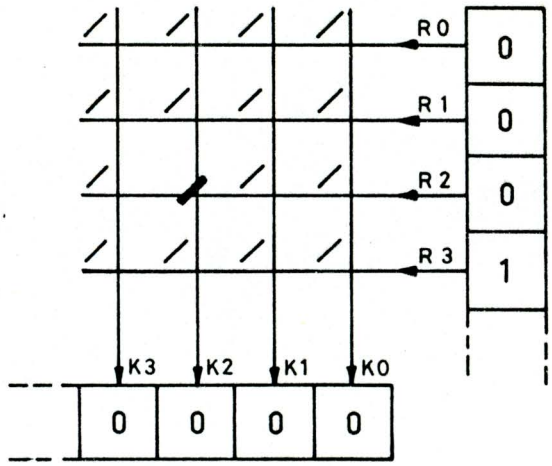


fig.6d

Vraag 3: Op deze wijze kunnen we m.b.v. 2 I/O-poorten maximaal ..... toetsen scannen.

Als we van beide poorten alle 8 bits gebruiken, kunnen we zo een toetsenbord met 8 x 8 = 64 toetsen aftasten.

Er zijn in dit soort applicaties nog twee problemen, nl.
1. Hoe realiseren we het principe van de "lopende 1" ?
2. Hoe combineren we de rij- en de kolom-informatie van een ingedrukte toets tot een gemakkelijk te bewerken en te herkennen code ?

We gaan bij het oplossen van deze problemen weer uit van het hexadecimale toetsenbord, dat volgens fig.7 op het microprocessorsysteem is aangesloten.

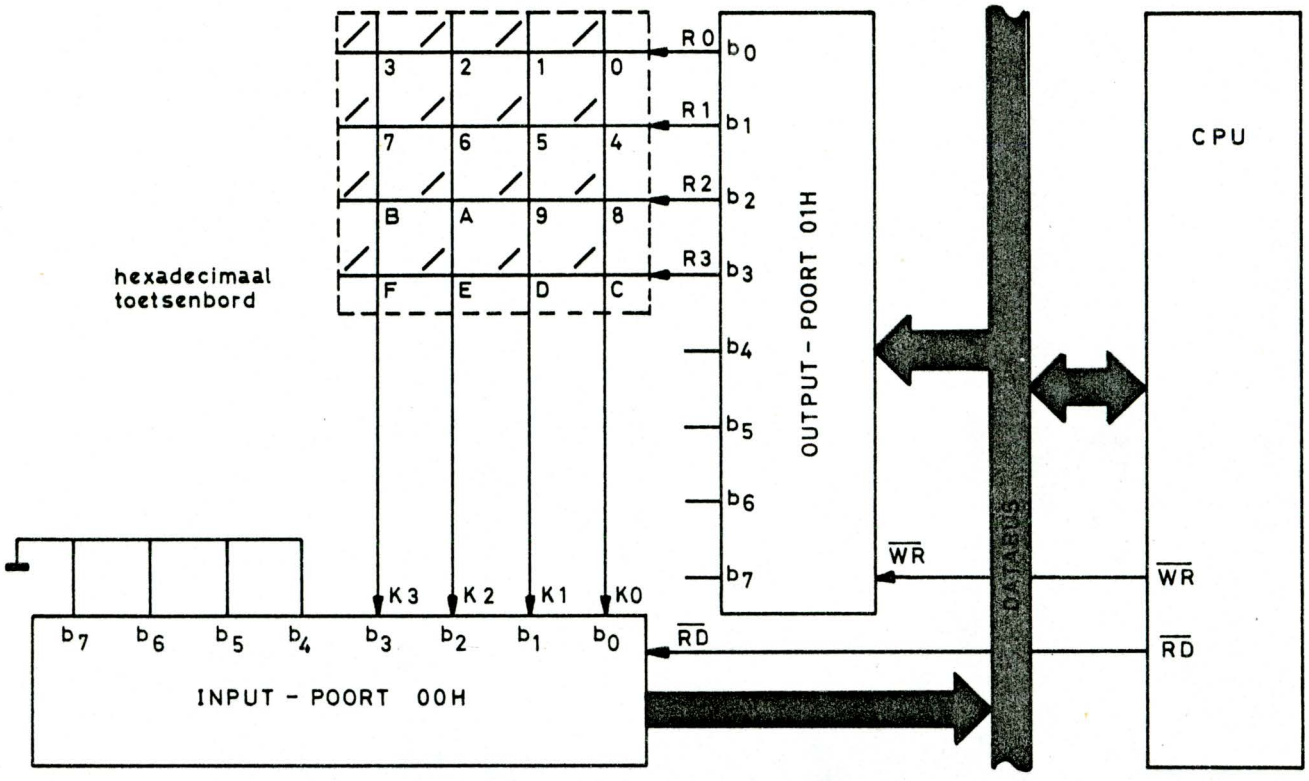


fig.7

Van elk van de I/O-poorten worden slechts 4 bits gebruikt, omdat we in dit geval slechts 16 toetsen hebben. Bij input-poort 00_{16} zijn de 4 meest significante bits aan 0 gelegd. Van output-poort 01_{16} zijn de 4 meest significante bits niet aangesloten. Als CPU wordt een 8085 gebruikt.

Vraag 4: Hiervoor maken we gebruik van optel-/schuif-/roteer-instructies.

We maken gebruik van roteerinstructies, omdat we hiermee een bepaalde bit in de accumulator kunnen laten rondgaan. Stel, dat we de accumulator vullen met 00010001_2.

Vraag 5: Na het uitvoeren van 4 RLC-instructies is de inhoud van de accumulator .....$_2$.

In tabel 1 is weergegeven wat de inhoud van de accumulator is na het resp. 1, 2, 3 en 4 maal uitvoeren van een RLC-instructie.

Antw.4: roteer-. Antw.5: 00010001.

	accumulator-inhoud
beginwaarde	00010001_2
na 1 maal RLC	00100010_2
na 2 maal RLC	01000100_2
na 3 maal RLC	10001000_2
na 4 maal RLC	00010001_2

Tabel 1

Omdat we slechts geïnteresseerd zijn in b_0 t/m b_3 (i.v.m. output-poort 01_{16}), is het niet belangrijk wat de overige 4 bits bevatten. Als we b_4 t/m b_7 gelijk maken aan b_0 t/m b_3, dan zal na 4 maal roteren de accumulator-inhoud weer gelijk zijn aan de beginwaarde. En dat is precies wat we wilden bereiken bij het scannen van de rijen. Na rij 3 moeten we weer terug naar rij 0. Het principe van de "lopende 1" is dan te bereiken met het programma in fig.8.

```
      MVI   A,00010001B
SCAN: OUT   01H
      RLC
      JMP   SCAN
```

fig.8

Vraag 6: Door dit programma verschijnt er achtereenvolgens op de 4 rijlijnen ....$_2$, ...$_2$, ...$_2$, ...$_2$, ...$_2$, enz.

Door uitvoering van dit programma bereiken we de "lopende 1" op b_0 t/m b_3 van de output-poort. Dit programma op zich is natuurlijk vrij zinloos. Er wordt alleen maar een scan-patroon gegenereerd, maar nergens vindt een test plaats, of er soms een toets is ingedrukt. Er moet dus nog een input-operatie en een test worden tussengevoegd.

Vraag 7: Als er een toets is ingedrukt, is de via input-poort 00_{16} ingevoerde kolominformatie wel/niet gelijk aan nul.

De kolominformatie bevat een 1 als er een toets is ingedrukt. Als na de input-instructie IN 00H blijkt, dat de inhoud van de accumulator 0 is, is er dus in de afgetaste rij geen enkele toets ingedrukt. We moeten dus na de input-instructie de accumulator-inhoud op 0 testen. Dit kan b.v. door het bekijken van de zero status flag na het uitvoeren van de instructie ANA A. (Dit had echter ook ORA A, CPI 00H, ADI 00H of SUI 00H mogen zijn. Ga dit voor uzelf na.)

Door het toevoegen van deze test, gaat het programma van fig.8 over in dat van fig.9.

```
        MVI   A,00010001B    ;BEGINWAARDE
SCAN:   OUT   01H            ;SCAN RIJ
        MOV   B,A            ;BERG RIJ-INFO OP
        IN    00H            ;VOER KOLOMINFO IN
        ANA   A              ;TEST KOLOMINFO
        JNZ   KEY            ;SPRING BIJ INGEDRUKTE TOETS
        MOV   A,B            ;HAAL RIJ-INFO TERUG
        RLC                  ;LOPENDE 1
        JMP   SCAN           ;VOLGENDE RIJ
KEY:    ...   ....           ;BEWERK RIJ- EN
        ...   ....           ; KOLOMINFO
```

fig.9

De instructies MOV B,A en MOV A,B zijn nodig om de rij-informatie veilig te stellen tijdens het invoeren en testen van de kolominformatie. Deze moeten immers i.v.m. de I/O mapped I/O beide via de accumulator worden uit- resp. ingevoerd. Als er geen CPU-registers beschikbaar zijn (dit programma kan nl. deel uit maken van een groter geheel), dan gebruiken we natuurlijk PUSH PSW en POP PSW.

Als het programma van fig.9 een ingedrukte toets heeft gedetecteerd, wordt naar de instructie met de label "KEY" gesprongen.

Vraag 8: De rij-informatie bevindt zich dan in register .....
. De kolominformatie staat in register .....

In register B staat dan de rij-informatie. Deze was hierin immers vlak voor de test veiliggesteld.
De kolominformatie bevindt zich nog in de accumulator, omdat de instructie ANA A wel de status flags activeert, maar niet de accumulator-inhoud wijzigt.

Op het adres met de symbolische naam "KEY" beginnen de instructies, die de rij- en kolominformatie bewerken om b.v. de binaire code voor een ingedrukte toets te bepalen.

Vraag 9: Als de toets D is ingedrukt, dan heeft register B de inhoud $....._2$.
In de accumulator staat dan $....._2$.

Als b.v. de toets D is ingedrukt, dan is de rij-informatie 1000_2 (rij 3) en de kolominformatie 0010_2 (kolom 1).
In register B staat dan 10001000_2, omdat de rij-informatie zowel in b_0 t/m b_3 als in b_4 t/m b_7 is opgeslagen.
De accumulator-inhoud is dan 00000010_2, want b_4 t/m b_7 van de input-poort zijn met 0 verbonden.

In tabel 2 is voor elke ingedrukte toets de rij-informatie, de kolominformatie en de bijbehorende binaire waarde vermeld.

Antw.8: B; A. Antw.9: 10001000; 00000010.

ingedrukte toets	rij-info	kolom-info	binaire waarde
0	0001	0001	00000000
1	0001	0010	00000001
2	0001	0100	00000010
3	0001	1000	00000011
4	0010	0001	00000100
5	0010	0010	00000101
6	0010	0100	00000110
7	0010	1000	00000111
8	0100	0001	00001000
9	0100	0010	00001001
A	0100	0100	00001010
B	0100	1000	00001011
C	1000	0001	00001100
D	1000	0010	00001101
E	1000	0100	00001110
F	1000	1000	00001111

Tabel 2

Op adres "KEY" in fig.9 moet een programmadeel beginnen, dat uit de rij- en de kolominformatie de binaire waarde voor een ingedrukte toets berekent en in de accumulator plaatst.

SCHRIJF NU ZELF HET GEWENSTE PROGRAMMADEEL EN TEST DIT OP DE SDK 85.
BESTUDEER DAARNA ONZE OPLOSSING IN FIG.10.

```
        MVI     A,11H     ;BEGINWAARDE
SCAN:   OUT     01H       ;SCAN RIJ
        MOV     B,A       ;BERG RIJ-INFO OP
        IN      00H       ;VOER KOLOMINFO IN
        ANA     A         ;TEST KOLOMINFO
        JNZ     KEY       ;SPRING BIJ INGEDRUKTE TOETS
        MOV     A,B       ;HAAL RIJ-INFO TERUG
        RLC               ;LOPENDE 1
        JMP     SCAN      ;VOLGENDE RIJ
KEY:    MVI     C,00H     ;(C) = 0000B
        RRC               ;KOLOM 0 ?
        JC      RIJ       ;SPRING ALS KOLOM 0 = 1
        INR     C         ;(C) = 0001B
        RRC               ;KOLOM 1 ?
        JC      RIJ       ;SPRING ALS KOLOM 1 = 1
        INR     C         ;(C) = 0010B
        RRC               ;KOLOM 2 ?
        JC      RIJ       ;SPRING ALS KOLOM 2 = 1
        INR     C         ;KOLOM 3 = 1:(C) = 0011B
RIJ:    MOV     A,B       ;RIJ-INFO NAAR ACCU
        RRC               ;RIJ 0 ?
        MVI     B,00H     ;(B) = 0000B
        JC      BIN       ;SPRING ALS RIJ 0 = 1
        RRC               ;RIJ 1 ?
        MVI     B,04H     ;(B) = 0100 B
        JC      BIN       ;SPRING ALS RIJ 1 = 1
        RRC               ;RIJ 2 ?
        MVI     B,08H     ;(B) = 1000B
        JC      BIN       ;SPRING ALS RIJ 2 = 1
        MVI     B,0CH     ;RIJ 3 = 1:(B) = 1100B
BIN:    MOV     A,B       ;PLAATS BINAIRE
        ADD     C         ; WAARDE IN ACCU
```

fig.10

In fig.10 is het programmadeel van fig.9 uitgebreid met instructies, die uit de kolominformatie (accumulator) en de rij-informatie (register B) de binaire waarde voor een ingedrukte toets berekenen en in de accumulator plaatsen.

Deze oplossing berust op het volgende principe:
a. Eerst wordt de kolominformatie omgezet in een binaire waarde 0000; 0001; 0010 of 0011.
b. Dan wordt de rij-informatie omgezet in een binaire waarde 0000; 0100; 1000 of 1100.
c. Daarna worden beide binaire waarden bij elkaar opgeteld.

U kunt dit programmadeel testen, door het achtereenvolgens vanaf het symbolische adres "KEY" te vertalen, in het RAM-geheugen van de SDK 85 te plaatsen en te starten.
Voor het starten vult u m.b.v. "EXAM REG" de accumulator en register B met de gewenste kolom- resp. rij-informatie.
Na afloop van het programma kunt u dan (m.b.v. tabel 2) de accumulatorinhoud controleren.

SAMENVATTING 3

7. Een scanned keyboard bestaat uit schakelaars, die in rijen en kolommen (matrix) met elkaar zijn verbonden.

8. Om een toetsenbord te kunnen aftasten, moet op de rijlijnen een "lopende 1" worden gerealiseerd.

9. Door het combineren van de rij-informatie en de kolominformatie is te bepalen of er een toets is ingedrukt en zo ja, welke toets dit is.

4. MULTIPLEXED DISPLAYS

Een goed voorbeeld van directe geprogrammeerde output is het aansturen van 7-segment displays. Evenals bij het toetsenbord, wordt in dat geval gebruik gemaakt van scannen. Bij het aansturen van displays, spreken we echter meestal van multiplexen.

In fig.11 is zo'n 7-segment display weergegeven.
Hoewel het een 7-segment display wordt genoemd, zijn er in feite 8 segmenten, want de decimale punt (dp) telt ook mee.

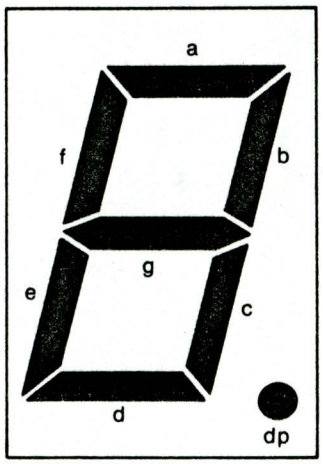


fig.11

In fig.12a is de interne opbouw van een 7-segment display weergegeven.
Elk segment wordt gevormd door een lichtgevende diode (LED).
De anodes van deze 8 LED's zijn met elkaar verbonden.
Deze uitvoering heet common anode (= gemeenschappelijke anode).

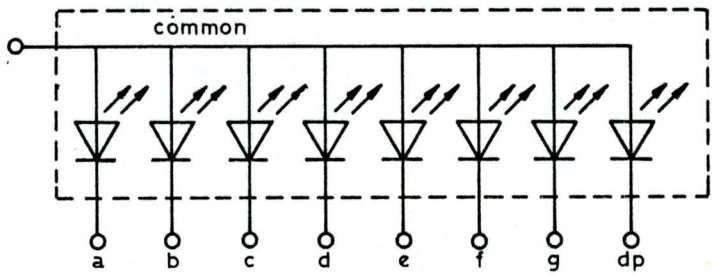


fig.12a

De common is met een positieve voedingsspanning verbonden. Als nu een van de katodes met een lagere spanning, b.v. 0 V, wordt verbonden, dan loopt er een stroom door de betreffende LED. Deze zal dan oplichten. Door het aansturen van meer katodes kunnen we karakters op het display laten oplichten. Stel dat we op de aansluitpunten a t/m dp in fig.12a de combinatie 00001101 aanbieden.

Vraag 10: Op het display wordt dan het cijfer ..... weergegeven.

Door deze combinatie blijven de segmenten e en f en de decimale punt gedoofd. De overige segmenten lichten op en vormen het cijfer 3.

In tabel 3 is aangegeven welke combinaties we voor de hexadecimale symbolen 0 t/m F aan het display van fig.12a moeten aanbieden.

symbool	a	b	c	d	e	f	g
0	0	0	0	0	0	0	1
1	1	0	0	1	1	1	1
2	0	0	1	0	0	1	0
3	0	0	0	0	1	1	0
4	1	0	0	1	1	0	0
5	0	1	0	0	1	0	0
6	0	1	0	0	0	0	0
7	0	0	0	1	1	1	1
8	0	0	0	0	0	0	0
9	0	0	0	1	1	0	0
A	0	0	0	1	0	0	0
B	1	1	0	0	0	0	0
C	0	1	1	0	0	0	1
D	1	0	0	0	0	1	0
E	0	1	1	0	0	0	0
F	0	1	1	1	0	0	0

Tabel 3

Opmerking:

Het is ook mogelijk om de katodes van de segmenten met elkaar te verbinden (fig.12b). Men spreekt dan van een common cathode display. Daarbij is de common met een lage spanning verbonden en moeten de segmenten met een hoge spanning worden aangestuurd om ze te laten oplichten.

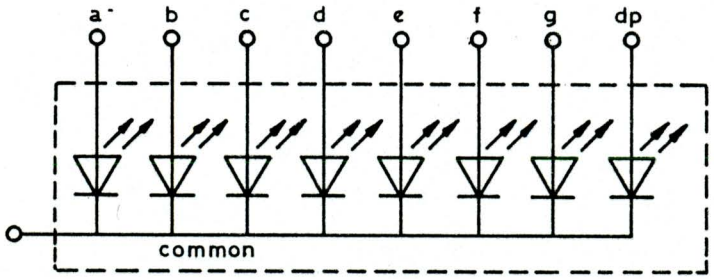


fig.12b

In de rest van deze les werken we steeds met common anode displays volgens fig.12a.

Vraag 11: Een 7-segment display heeft ..... aansluitpunten.

Een 7-segment common anode heeft 9 aansluitpunten, nl. één voor de common anode en 8 voor de katodes. Het is echter niet gebruikelijk om per 7-segment display 9 output-lijnen te reserveren.

 Antw.11: 9.

Vraag 12: Voor 8 displays zouden dan ..... output-poorten nodig zijn.

Om b.v. 8 displays aan te sluiten, zouden dan 72 output-lijnen, dus 9 output-poorten, noodzakelijk zijn.

In fig.13 is weergegeven, hoe de displays meestal worden aangesloten. De aansluitingen van de segmenten van de 8 displays zijn alle verbonden en wel zo, dat alle segmenten a, alle segmenten b, enz., op één lijn van output-poort B zijn aangesloten. In elke output-lijn moet wel een driver (= stroomversterker) worden opgenomen, omdat de output-poort de benodigde stroom niet kan leveren.
De commons zijn elk, ook weer via een driver, op een lijn van output-poort A aangesloten.
Om een bepaald display aan te sturen, moet nu via poort A een 1 op de betreffende common worden geplaatst en via poort B de gewenste 7-segment-code aan de segmenten worden afgegeven.
In tabel 4 is van elke output-lijn de functie vermeld.

POORT A	FUNCTIE
b_7	common display 7
b_6	common display 6
b_5	common display 5
b_4	common display 4
b_3	common display 3
b_2	common display 2
b_1	common display 1
b_0	common display 0

POORT B	FUNCTIE
b_7	segment d
b_6	segment c
b_5	segment b
b_4	segment a
b_3	decimale punt
b_2	segment g
b_1	segment f
b_0	segment e

Tabel 4

Stel, dat we op display 4 het cijfer 7 willen weergeven. De decimale punt moet uit zijn.

Vraag 13: Via poort A moet dan de combinatie $\ldots\ldots_2$ worden uitgevoerd.
Poort B moet dan de combinatie $\ldots\ldots_2$ afgeven.

We moeten dan zorgen, dat alleen de common van display 4 met een 1 wordt aangestuurd. Dit bereiken we door via poort A de combinatie 00010000_2 uit te voeren. D.m.v. poort B moeten dan de segmenten a(= b_4), b(= b_5) en c(= b_6) met 0 worden aangestuurd. Poort B moet dan de combinatie 10001111_2 afgeven. Deze 7-segment-code wordt wel aan alle 8 displays tegelijk aangeboden, maar omdat alleen op de common van display 4 een 1 staat, zullen ook alleen van dit display segmenten kunnen oplichten.

In tabel 5 zijn de combinaties vermeld, die we via poort B moeten uitvoeren om de hexadecimale symbolen 0 t/m F op een (door poort A bepaald) display op te laten lichten.

symbool	binair d	c	b	a	dp	g	f	e	hexadecimaal
0	0	0	0	0	1	1	0	0	0C
1	1	0	0	1	1	1	1	1	9F
2	0	1	0	0	1	0	1	0	4A
3	0	0	0	0	1	0	1	1	0B
4	1	0	0	1	1	0	0	1	99
5	0	0	1	0	1	0	0	1	29
6	0	0	1	0	1	0	0	0	28
7	1	0	0	0	1	1	1	1	8F
8	0	0	0	0	1	0	0	0	08
9	0	0	0	0	1	0	0	1	09
A	1	0	0	0	1	0	0	0	88
b	0	0	1	1	1	0	0	0	38
C	0	1	1	0	1	1	0	0	6C
d	0	0	0	1	1	0	1	0	1A
E	0	1	1	0	1	0	0	0	68
F	1	1	1	0	1	0	0	0	E8

Tabel 5

In tabel 5 is er steeds van uit gegaan, dat de decimale punt gedoofd is. Moet deze wel branden, dan moet op poort B gelden $b_3 = 0$. (Door de volgorde waarin de segmenten op poort B zijn aangesloten, zijn de combinaties in tabel 5 niet gelijk aan die in tabel 3.)

Stel, we willen op de displays in fig.13 de tekst "EOd 1980." weergeven. We moeten de 8 displays dan achtereenvolgens aansturen. Steeds wanneer via poort A een bepaald display wordt geselecteerd (= de common wordt aangestuurd), moet via poort B de 7-segment-code voor het op dit display weer te geven karakter worden uitgevoerd. In tabel 6 is vermeld welke combinaties achtereenvolgens via de poorten A en B moeten worden uitgevoerd.

Poort A	Poort B	
00000001	01101000	"E" op display 0
00000010	00001100	"O" op display 1
00000100	00011010	"d" op display 2
00001000	11111111	display 3 gedoofd
00010000	10011111	"1" op display 4
00100000	00001001	"9" op display 5
01000000	00001000	"8" op display 6
10000000	00000100	"0." op display 7

Tabel 6

M.b.v. de "lopende 1" worden de displays één voor één aangestuurd. Na het aansturen van display 7 moet weer opnieuw worden begonnen met display 0. Als we er nu voor zorgen, dat elk display gedurende ca. 0,5 ms wordt aangestuurd, dan zal door de traagheid van ons oog de indruk ontstaan, dat de displays tegelijkertijd oplichten. Een voorwaarde is wel, dat dit proces continu wordt voortgezet.

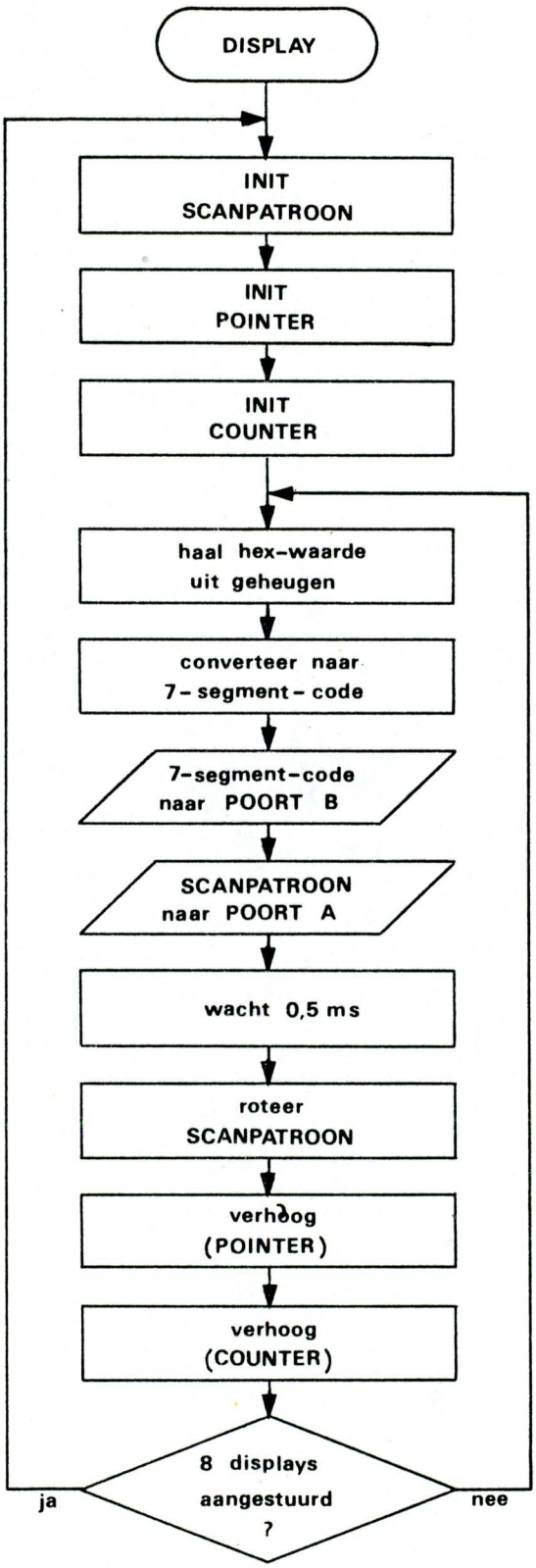

DISPLAY
INIT SCANPATROON
INIT POINTER
INIT COUNTER
haal hex-waarde uit geheugen
converteer naar 7-segment-code
7-segment-code naar POORT B
SCANPATROON naar POORT A
wacht 0,5 ms
roteer SCANPATROON
verhoog (POINTER)
verhoog (COUNTER)
8 displays aangestuurd ?
ja
nee

fig.14

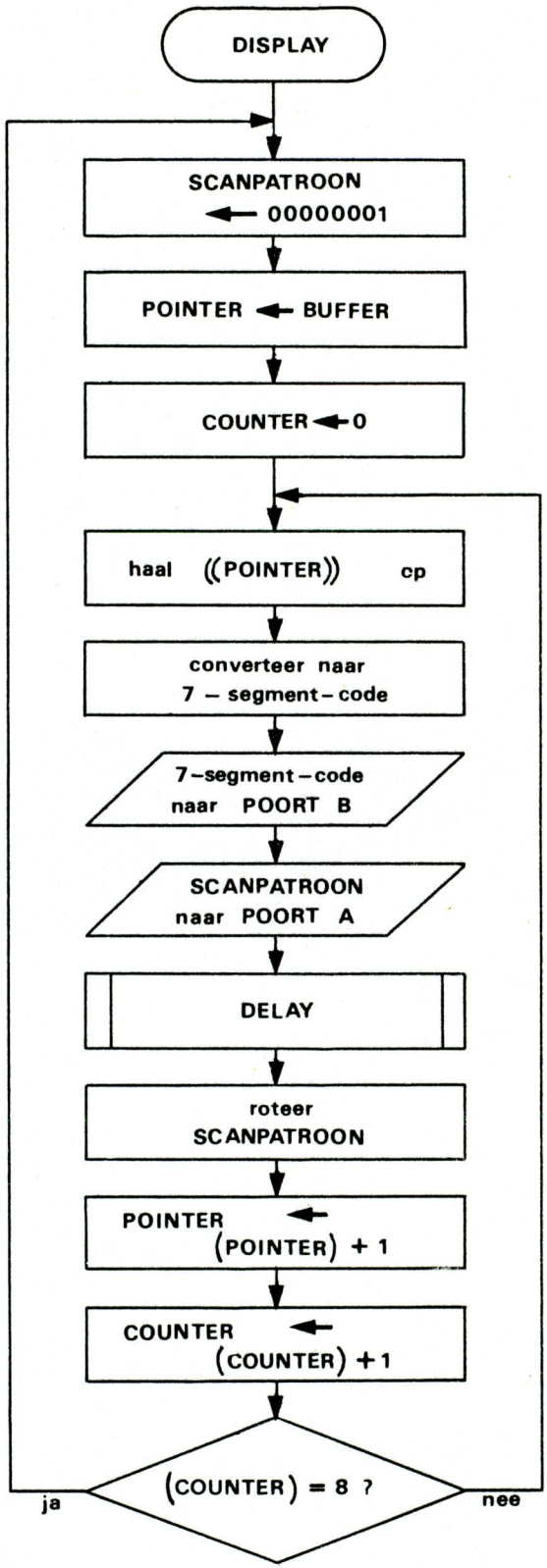

DISPLAY
SCANPATROON ← 00000001
POINTER ← BUFFER
COUNTER ← 0
haal ((POINTER)) cp
converteer naar 7 – segment – code
7-segment-code naar POORT B
SCANPATROON naar POORT A
DELAY
roteer SCANPATROON
POINTER ← (POINTER) + 1
COUNTER ← (COUNTER) + 1
(COUNTER) = 8 ?
ja
nee

fig.15

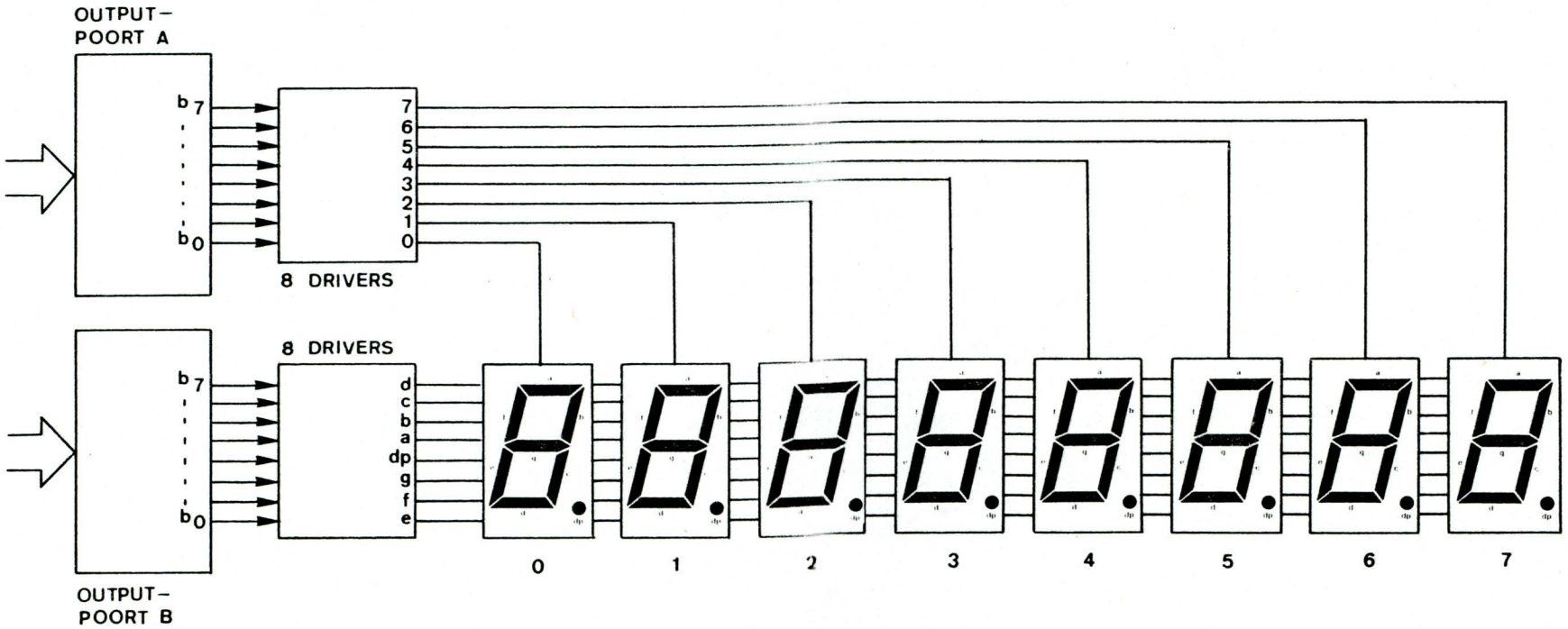

OUTPUT-
POORT A
b 7
b 0
8 DRIVERS
7
6
5
4
3
2
1
0
8 DRIVERS
b 7
b 0
d
c
b
a
dp
g
f
e
0
1
2
3
4
5
6
7
OUTPUT-
POORT B

fig.13

```
                        ORG     2800H
                PRTA    EQU     21H         ;POORT A : OUTPUT
                                            ;VOOR SCANPATROON
                PRTB    EQU     22H         ;POORT B : OUTPUT
                                            ;VOOR 7-SEGMENT-CODE
                BUFF    EQU     2880H       ;BUFFER : EERSTE ADRES
                                            ;VAN HEX-WAARDEN
2800  3E 03    DISPL:  MVI     A,03H       ;INIT I/O-MODULE
2802  D3 20            OUT     20H         ;POORT 21H = OUTPUT
2804  32 FF 20        STA     20FFH       ;POORT 22H = OUTPUT
2807  31 C2 20        LXI     SP,20C2H
280A  0E 01    INIT:   MVI     C,01H       ;INIT SCANPATROON
280C  11 80 28        LXI     D,BUFF      ;INIT POINTER
280F  06 00           MVI     B,00H       ;INIT COUNTER
2811  1A        NEXT:   LDAX    D           ;HAAL HEX-WAARDE OP
2812  21 39 28        LXI     H,TABEL     ;BEREKEN
2815  85              ADD     L           ; ADRES
2816  6F              MOV     L,A         ;  VAN DE
2817  D2 1B 28        JNC     UIT         ;   7-SEGMENT-
281A  24              INR     H           ;    CODE
281B  7E        UIT:    MOV     A,M         ;VOER 7-SEGMENT-
281C  D3 22           OUT     PRTB        ; CODE UIT
281E  79              MOV     A,C         ;VOER SCAN-
281F  D3 21           OUT     PRTA        ; PATROON UIT
2821  CD 32 28        CALL    DELAY       ;WACHT CA  0.5 MS
2824  79              MOV     A,C         ;ROTEER
2825  07              RLC                 ; SCAN-
2826  4F              MOV     C,A         ;  PATROON
2827  13              INX     D           ;VERHOOG POINTER
2828  04              INR     B           ;VERHOOG COUNTER
2829  3E 08           MVI     A,08H       ;ZIJN ER 8 DISPLAYS
282B  B8              CMP     B           ; AANGESTUURD ?
282C  C2 11 28        JNZ     NEXT        ;NEE : VOLGEND DISPLAY
282F  C3 0A 28        JMP     INIT        ;JA : BEGIN OPNIEUW
2832  3E 64    DELAY:  MVI     A,64H       ;DEZE SUBROUTINE
2834  3D      LUS:    DCR     A           ; GENEREERT EEN
2835  C2 34 28        JNZ     LUS         ;  WACHTTIJD
2838  C9              RET                 ;   VAN CA 0.5 MS
2839  0C        TABEL:  DB      0CH         ;0
283A  9F                DB      9FH         ;1  **********
283B  4A                DB      4AH         ;2  *
283C  0B                DB      0BH         ;3  * TABEL MET
283D  99                DB      99H         ;4  * 7-SEGMENT
283E  29                DB      29H         ;5  *   -CODES
283F  28                DB      28H         ;6  *
2840  8F                DB      8FH         ;7  **********
2841  08                DB      08H         ;8
2842  09                DB      09H         ;9
2843  88                DB      88H         ;A
2844  38                DB      38H         ;B
2845  6C                DB      6CH         ;C
2846  1A                DB      1AH         ;D
2847  68                DB      68H         ;E
2848  E8                DB      E8H         ;F
                        END
```

fig.16

We zullen voor dit multiplexen van 8 displays het stroomdiagram tekenen. We gaan er hierbij van uit, dat in het geheugen 8 locaties zijn gereserveerd, waarin de op de displays weer te geven hexadecimale waarden zijn opgeslagen. Het eerste adres van deze 8 geheugenwoorden geven we de symbolische naam BUFFER. Om aan te wijzen, welke van deze 8 geheugeninhouden naar de displays moet worden gezonden, gebruiken we een adresaanwijzer met de symbolische naam POINTER.
Om aan te geven, welk van de displays moet worden aangestuurd, creëren we een display-teller met de symbolische naam COUNTER.

Fig.14 en fig.15 zijn het algemeen, resp. gedetailleerd stroomdiagram voor dit probleem. U moet nu zonder meer in staat zijn, deze stroomdiagrammen te analyseren en te doorzien. Beide stroomdiagrammen zijn als oneindige lus uitgevoerd. Het aansturen van de displays moet nl. continu doorgaan.

Fig.15 is eenvoudig om te zetten in een programma. Er blijft alleen het probleem op welke wijze we een hexadecimale waarde naar de juiste 7-segment-code converteren. Een mogelijke oplossing is de volgende.
We plaatsen de 7-segment-codes in volgorde in 16 geheugenwoorden.
Het eerste van de hiervoor gereserveerde adressen noemen we TABEL.

Vraag 14: De 7-segment-code voor 9_{16} bevindt zich dan op adres .....

In tabel 7 is weergegeven hoe de 7-segment-codes in het geheugen zijn opgeslagen.

hexadec. waarde	adres	inhoud (7-segment-code)
0	TABEL	0C
1	TABEL + 1	9F
2	TABEL + 2	4A
3	TABEL + 3	0B
4	TABEL + 4	99
5	TABEL + 5	29
6	TABEL + 6	28
7	TABEL + 7	8F
8	TABEL + 8	08
9	TABEL + 9	09
A	TABEL + A	88
B	TABEL + B	38
C	TABEL + C	6C
D	TABEL + D	1A
E	TABEL + E	68
F	TABEL + F	E8

Tabel 7

Het adres van de gewenste 7-segment-code is nu eenvoudig te berekenen door de hexadecimale waarde bij TABEL op te tellen.

ZET NU ZELF FIG.15 OM IN EEN PROGRAMMA. BESTUDEER DAARNA ONZE OPLOSSING IN FIG.16.

Fig.16 is het programma, zoals dit na vertaling door de assembler wordt afgedrukt. Hierin is de assembler directive DB (= define byte) gebruikt. Deze directive zorgt ervoor, dat hetgeen in de operand-kolom staat, niet als label of adres wordt vertaald. De operand wordt als 8-bits getal in het objectprogramma geplaatst.

We hebben zo een programma verkregen, dat met directe geprogrammeerde output 8 7-segment displays aanstuurt.
Een groot nadeel is echter, dat de CPU continu met dit programma bezig moet zijn, omdat anders de displays weer zullen doven. Er is dus geen tijd meer voor de uitvoering van een programma, waarin b.v. resultaten ontstaan.
Dit nadeel is op te heffen, door een tweede CPU in het systeem op te nemen. Deze processor dient dan het scannen van de displays (en liefst ook van het toetsenbord) te verzorgen.
In de SDK 85 is zo'n I/O-processor aanwezig. Dit is nl. de keyboard display controller 8279.

SAMENVATTING 4

10. 7-segment displays worden vaak gemultiplexed, d.w.z. één voor één aangestuurd. Door de traagheid van ons oog lijkt het of alle displays continu worden aangestuurd.

11. Bij multiplexed displays zijn de overeenkomstige segment-aansluitingen van alle displays doorverbonden.
Een display wordt dan op de common aangestuurd.

12. Het voordeel van het multiplexen van 7-segment displays is het besparen van output-lijnen.

13. Het nadeel van scannen van toetsenbord en displays m.b.v. directe geprogrammeerde I/O is het feit, dat dit proces continu moet worden voortgezet. De CPU kan zich dan niet bezig houden met de uitvoering van een hoofdprogramma.

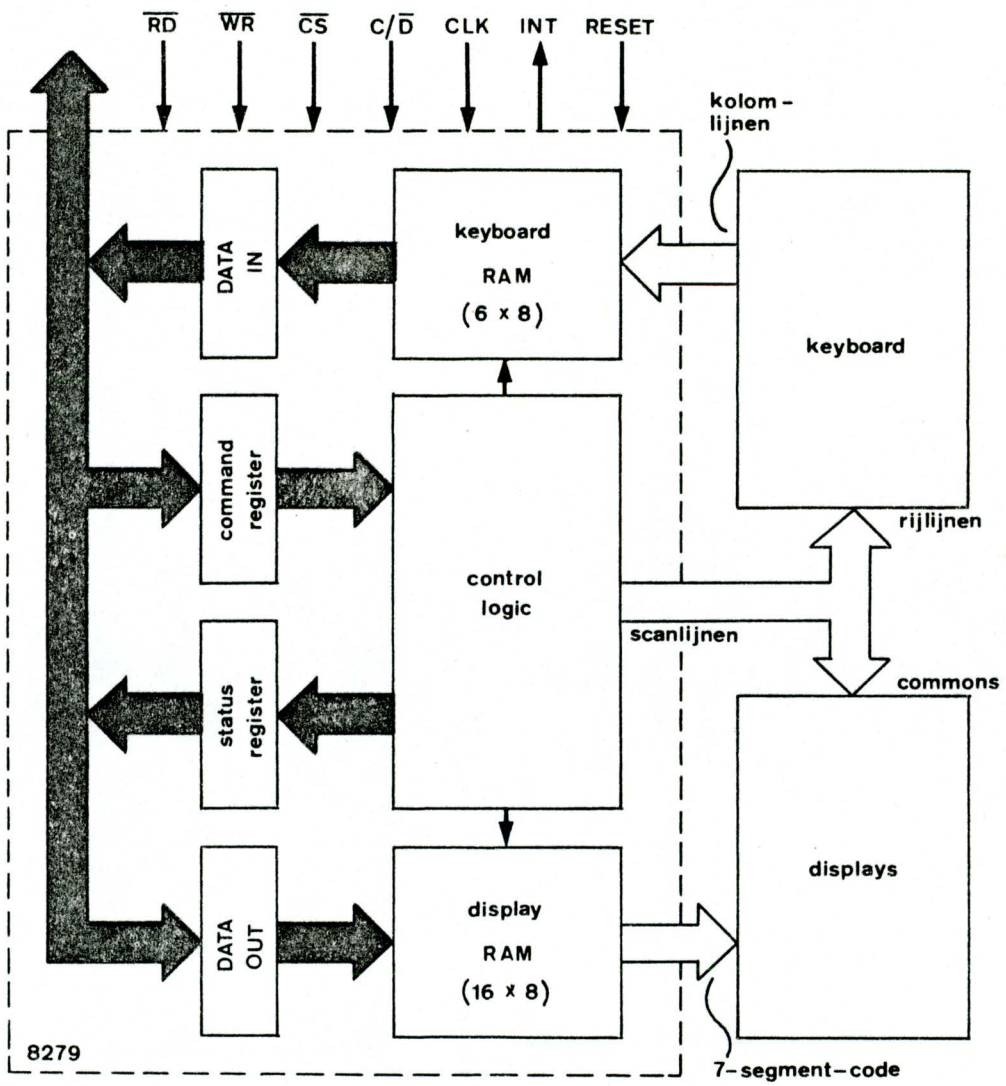

RD
WR
CS
C/D
CLK
INT
RESET
kolom-
lijnen
DATA
IN
keyboard
RAM
(6 x 8)
keyboard
command
register
control
logic
rijlijnen
scanlijnen
commons
status
register
displays
DATA
OUT
display
RAM
(16 x 8)
8279
7-segment-code

fig.17

5. DE KEYBOARD DISPLAY CONTROLLER 8279

De 8279 is een universele programmeerbare keyboard display controller. Terwijl de CPU bezig is met de programma-uitvoering, kan de 8279 geheel zelfstandig een toetsenbord aftasten en tegelijkertijd maximaal 16 7-segment displays aansturen. Hiertoe moet van te voren wel aan de 8279 zijn meegedeeld, op welke wijze dit dient te geschieden.
De 8279 kent voor het scannen van zowel het toetsenbord als de displays een aantal mogelijkheden, waaruit de gewenste gekozen kan worden door het vullen van een command register. De 8279 is dus een programmeerbare chip.

In fig.17 is het blokschema van de 8279 weergegeven. Zowel de commando's voor als de data van en naar deze chip worden via de databus getransporteerd.

Vraag 15: Als $C/\overline{D} = 0$, wordt via de databus data/een commando verzonden.

Met het $C/\overline{D}$-signaal wordt steeds aangegeven of het data of een commando betreft, nl.:
$C/\overline{D} = 1$: commando
$C/\overline{D} = 0$: data

Hoewel de 8279 een zeer uitgebreide chip is, zijn er geen aansluitingen voor de adresbus aanwezig. Met de signalen $\overline{RD}$, $\overline{WR}$, $\overline{CS}$ en $C/\overline{D}$ is elke voor de gebruiker belangrijke locatie te adresseren (zie tabel 8).

$\overline{CS}$	$C/\overline{D}$	$\overline{RD}$ of $\overline{WR}$	actie van de 8279
1	X	X	-
0	0	$\overline{RD}$	(DATA IN) naar databus
0	0	$\overline{WR}$	(databus) naar DATA OUT
0	1	$\overline{RD}$	(statusregister) naar databus
0	1	$\overline{WR}$	(databus) naar command register
- = geen actie X = don't care			

Tabel 8

We zullen nu de belangrijkste blokken uit fig.17 kort toelichten.
Deze beschrijving bevat echter alleen de meest gebruikte toepassingsmogelijkheden van de 8279. Voor een meer gedetailleerde beschrijving van deze chip, verwijzen we naar de databooks.
Nu u zover met deze cursus bent gevorderd, moet u in staat zijn om de (Engelstalige) beschrijvingen te begrijpen, die door een fabrikant bij (of over) een programmeerbare chip worden geleverd.

SAMENVATTING 5

14. De 8279 is een programmeerbare keyboard display controller, die toetsenbord en displays kan scannen, terwijl de CPU bezig is met de uitvoering van een programma. De 8279 en de CPU werken dus parallel.

15. De 8279 heeft geen echte adres-ingangen. De door de CPU adresseerbare locaties worden d.m.v. $\overline{RD}$, $\overline{WR}$, $\overline{CS}$ en C/$\overline{D}$ geselecteerd.

16. De 8279 bevat
    a. Control logic
    b. Keyboard RAM
    c. Display RAM
    d. Command register
    e. Statusregister
    f. DATA IN-register
    g. DATA OUT-register

a. Control logic

De control logic bestuurt de gehele gang van zaken binnen de 8279. Hoe deze control logic is opgebouwd, is voor de programmeur niet van belang. Wel belangrijk is het feit dat de control logic 4 scan-lijnen heeft. Hiermee worden zowel het toetsenbord als de 7-segment displays gescanned. Er wordt op deze 4 scan-lijnen binair geteld.
Om het principe van de "lopende 1" te verkrijgen, moeten nog 3-naar-8 decoders (b.v. 8205) of 4-naar-16 decoders worden aangebracht.
Hierop wordt verderop in de les ingegaan.

b. Keyboard RAM

De keyboard RAM heeft een capaciteit van 8 bytes. Het vullen en uitlezen van deze 8 geheugenwoorden gaat volgens FIFO (= first in first out). We zullen dit verduidelijken aan de hand van een voorbeeld.
Er wordt achtereenvolgens op de hexadecimale toetsen A, 3 en F gedrukt.
Na het indrukken van toets 3 leest de CPU het DATA IN-register uit.
De 8279 is in dit geval zo geprogrammeerd, dat DATA IN steeds dezelfde inhoud heeft als de eerste RAM-locatie (adres 000_{16}).
Na het indrukken van de toets F leest de CPU DATA IN nog tweemaal uit.

In fig.18 is aangegeven, wat in dit geval achtereenvolgens de inhouden van accumulator, DATA IN en de keyboard RAM zijn.

	ACCU	DATA IN	keyboard 000	001	RAM_Locaties 010	011	100	101
begin inhouden	—	—	—	—	—	—	—	—
na indrukken toets A	—	0A	0A	—	—	—	—	—
na indrukken toets 3	—	0A	0A	03	—	—	—	—
na read DATA IN	0A	03	03	—	—	—	—	—
na indrukken toets F	—	03	03	0F	—	—	—	—
na read DATA IN	03	0F	0F	—	—	—	—	—
na read DATA IN	0F	—	—	—	—	—	—	—

— = geen gedefinieerde inhoud

fig.18

Het is duidelijk te zien, dat de code voor een ingedrukte toets steeds rechts van (achter) de voorgaande codes in het RAM-geheugen wordt geplaatst. Steeds wanneer de CPU via DATA IN een code ophaalt, worden de RAM-inhouden 1 plaats meer naar links geschoven. Het is dus zo onmogelijk, dat de code voor een ingedrukte toets tweemaal als nieuwe input data wordt ingelezen.

Vraag 16: De 8279 kan 7/8/9 codes voor ingedrukte toetsen opslaan.

Op deze wijze kan de 8279 8 ingedrukte toetsen verwerken en opslaan, terwijl de CPU doorgaat met de programma-uitvoering. Als er daarna een 9e toets wordt ingedrukt, wordt hiervoor geen code in de RAM opgeslagen. Er wordt wel een foutmelding in het statusregister geplaatst (zie paragraaf 5d).

SAMENVATTING 6

17. De control logic bestuurt de gehele gang van zaken binnen de 8279 en plaatst een telpatroon op de 4 scan-lijnen.

18. De keyboard RAM bestaat uit 8 bytes, waarin de codes voor 8 achtereenvolgens ingedrukte toetsen kunnen worden opgeslagen.

19. De keyboard RAM werkt volgens het FIFO-principe. Dit houdt in, dat hetgeen er het eerst is ingescheven, ook het eerst wordt uitgelezen.

20. De CPU kan een locatie van de keyboard RAM uitlezen, door het DATA IN-register te adresseren.

c. Display RAM

De display RAM heeft een capaciteit van 16 bytes. In deze display RAM worden de 7-segment-codes voor maximaal 16 7-segment displays opgeslagen (fig.19).

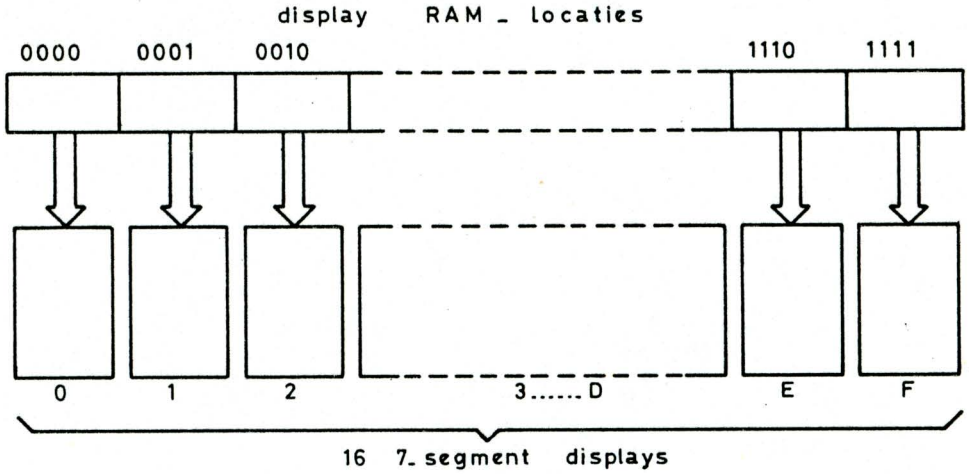


fig.19

Het vullen van de RAM-geheugenwoorden gebeurt vanuit het DATA OUT-register.

Vraag 17: Om DATA OUT te vullen moet de CPU een $\overline{RD}/\overline{WR}$-impuls afgeven en $C/\overline{D}$ moet dan 0/1 zijn.

De CPU kan DATA OUT vullen als $C/\overline{D}$ 0 is. Door een $\overline{WR}$-impuls wordt dan de op de databus aanwezige bitcombinatie in DATA OUT geplaatst.
Als we van te voren een correct commando WRITE DISPLAY RAM hebben gegeven (zie paragraaf 6g), dan plaatst de 8279 de inhoud van DATA OUT in de juiste RAM-locatie en geeft deze op het gewenste display weer.

23-12 Antw.17: $\overline{WR}$; 0.

d. Statusregister

Het statusregister (fig.20) bevat status-informatie en foutmeldingen van de keyboard RAM en de display RAM

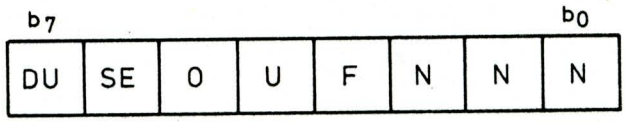


fig.20

NNN (b_0, b_1, b_2) geven binair aan hoeveel codes er in de keyboard RAM zijn opgeslagen.

F (FIFO FULL = keyboard RAM vol) is 1 als alle 8 locaties in de keyboard RAM zijn gevuld.

Vraag 18: De maximale inhoud van FNNN is .....$_2$.

Omdat de keyboard RAM maximaal 8 codes kan bevatten, is de maximale waarde in FNNN dus 1000_2.

Vraag 19: Als de CPU DATA IN uitleest wordt NNN met 1 verhoogd/verlaagd.
Als er een toets wordt ingedrukt, wordt NNN met 1 verhoogd/verlaagd.

De inhoud van NNN wordt automatisch aangepast, wanneer de CPU een nieuwe code uit de keyboard RAM ophaalt of wanneer er een nieuwe toets wordt ingedrukt en de bijbehorende code in de keyboard RAM wordt geplaatst.

U (= underrun error) wordt 1, als de CPU DATA IN uitleest, op een moment dat er nog geen nieuwe codes in de keyboard RAM zijn opgeslagen.

Vraag 20: U kan alleen 1 worden, als FNNN de waarde.....$_2$ bevat.

O (= overrun error) wordt 1, als er een toets wordt ingedrukt, op het moment dat er al 8 codes in de keyboard RAM zijn opgeslagen.

Vraag 21: O kan alleen 1 worden als FNNN de waarde .....$_2$ bevat.

SE (sensor matrix/error mode) wordt alleen toegepast als er i.p.v. een normaal toetsenbord een elektronische schakeling is aangesloten. Hierop gaan we in deze les niet in. De toestand van b_6 van het statusregister is in ons geval steeds 0.

Antw.18: 1000. Antw.19: verlaagd; verhoogd. Antw.20: 0000.
Antw.21: 1000.

DU (display unavailable = display RAM niet beschikbaar) is 1 gedurende de tijd, dat we de display RAM niet mogen vullen. Dit kan gebeuren als de 8279 bezig is met de initialisatie van de display RAM (zie paragraaf 6c). De CPU kan door het testen van DU bepalen wanneer er weer nieuwe data naar de display RAM mag worden gezonden.

Vraag 22: DATA OUT mag worden geselecteerd als DU 0/1 is.

SAMENVATTING 7

21. De display RAM bestaat uit 16 bytes, waarin de 7-segment-codes t.b.v. maximaal 16 displays dienen te worden opgeslagen.

22. De CPU kan een bepaalde locatie in de display RAM vullen door het DATA OUT-register te adresseren.
Van te voren moet dan wel het juiste commando write display RAM in het command register zijn geplaatst.

23. In het statusregister wordt informatie opgeslagen over
    a. het aantal gevulde locaties in de keyboard RAM.
    b. het wel of niet vol zijn van de keyboard RAM.
    c. foutmeldingen voór te veel vullen resp. te vaak uitlezen van de keyboard RAM.
    d. het wel of niet beschikbaar zijn van de display RAM.

Antw.22: 0.

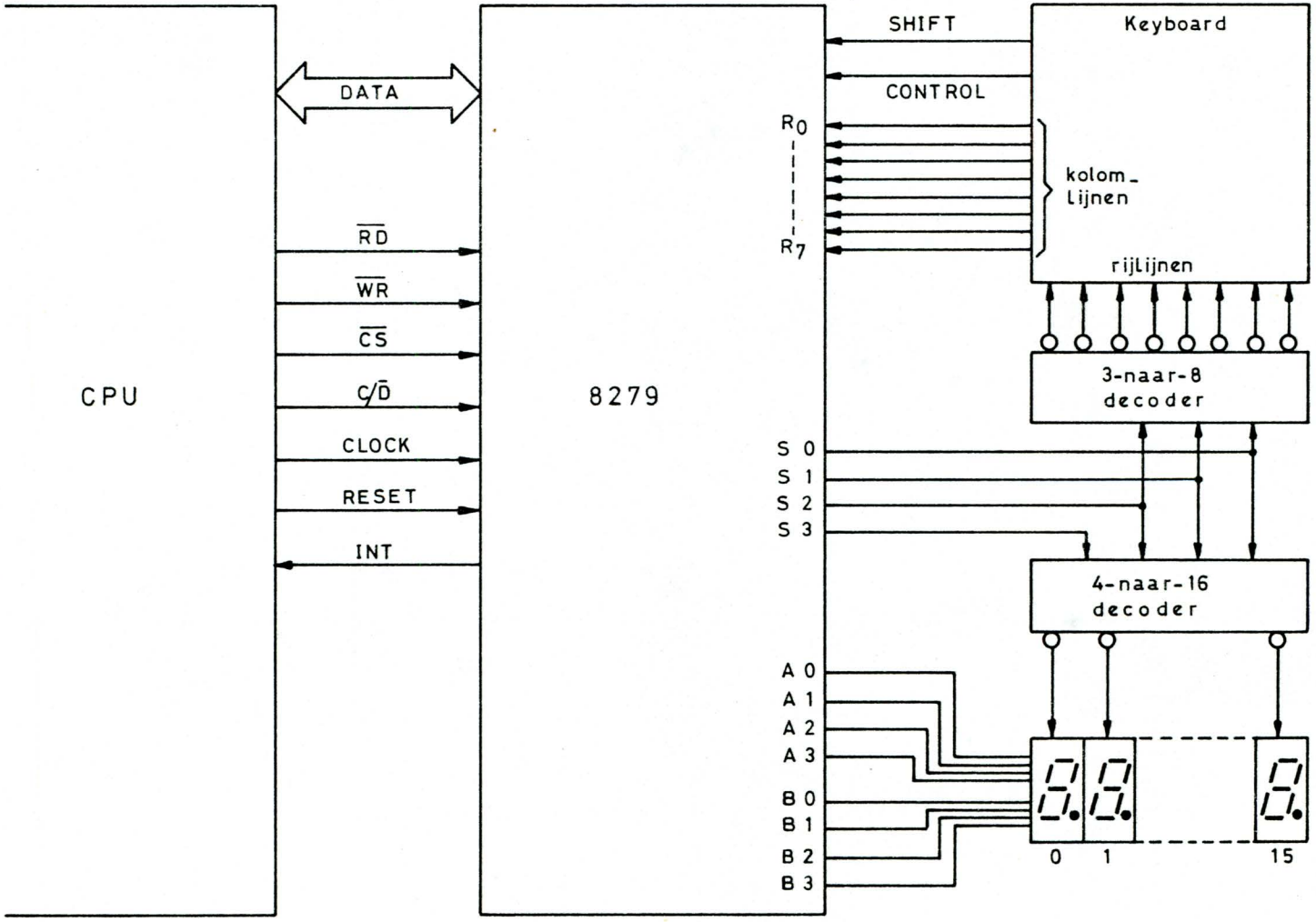

CPU
DATA
RD
WR
CS
C/D
CLOCK
RESET
INT
8279
SHIFT
CONTROL
R0
R7
Keyboard
kolom_
lijnen
rijlijnen
3-naar-8
decoder
S 0
S 1
S 2
S 3
4-naar-16
decoder
A 0
A 1
A 2
A 3
B 0
B 1
B 2
B 3
0
1
15

fig.22

6. COMMAND REGISTER (8279)

In fig.17 is één 8-bits command register getekend. In feite zijn er nog 8 5-bits command registers aanwezig (fig.21). De 8279 heeft nl. 8 verschillende functies, die gelijktijdig kunnen worden uitgevoerd.
Voor elk van deze functies is een 5-bits command register aanwezig.

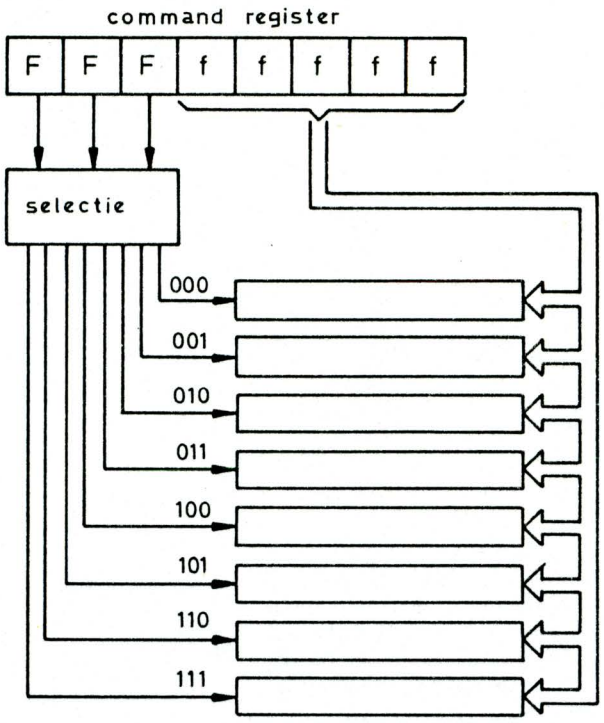


fig.21

De algemene vorm van een opdracht voor de 8279 is FFFfffff.
FFF selecteert de functie, dus het type commando.
Met fffff wordt voor elke afzonderlijke functie aangegeven, hoe deze dient te worden uitgevoerd. Voor elke functie zijn nl. een aantal mogelijkheden. Deze fffff wordt, afhankelijk van FFF, in het juiste 5-bits command register geplaatst. Hiermee is bereikt dat een bepaalde functie gewijzigd wordt, zonder dat dit invloed heeft op de overige functies.

De 8 mogelijke commando's zijn in drie groepen te verdelen, nl.
1. Algemene initialisatie 8279:
   a. 000fffff, keyboard/display mode
   b. 001fffff, program clock
   c. 110fffff, clear
2. Input via keyboard:
   d. 010fffff, read keyboard RAM
   e. 111fffff, end interrupt/error mode
3. Output via 7-segment displays
   f. 011fffff, read display RAM
   g. 100fffff, write display RAM
   h. 101fffff, write inhibit/blanking.

We zullen deze commando's nu achtereenvolgens toelichten.
Bij de volgende beschrijving dient u steeds fig.22 te raadplegen.
In deze figuur zijn de noodzakelijke buffers t.b.v. de 7-segment displays niet weergegeven.

a. Keyboard/display mode set

Dit commando dient om de 8279 mee te delen, hoe het toetsenbord en de 7-segment displays moeten worden gescanned.

0	0	0	D	D	K	K	S

fig.23a

S (= b_0) geeft aan of er sprake is van decoded scan (S = 1) of encoded scan (S = 0). Decoded scan wil zeggen, dat op de 4 scan-lijnen (S0 t/m S3) achtereenvolgens de combinaties 1000, 0100, 0010, 0001, 1000, enz. worden geplaatst. Op deze 4 lijnen verschijnt dus direct de "lopende 1".

Vraag 23: Er kunnen dan maximaal 4/8/16 7-segment displays worden aangesloten.

Bij decoded scan zijn er maar 4 rijlijnen voor het toetsenbord beschikbaar. Er kunnen ook slechts 4 displays worden aangesloten. Een voordeel is echter, dat de 3-naar-8- en de 4-naar-16-decoders uit fig.22 kunnen worden weggelaten. Decoded scan wordt dan ook toegepast voor kleine toetsenborden en afleeseenheden.

Bij encoded scan ontstaat er op de scan-lijnen een binair telpatroon. Om de "lopende 1" te verkrijgen, zijn decoders nodig.
Voor de 8 rijlijnen van het toetsenbord wordt een 3-naar-8-decoder aangesloten op S0, S1 en S2 (S3 mag voor het toetsenbord niet worden gebruikt). Voor het aansturen van de commons van de 7-segment displays is een 4-naar-16-decoder vereist. Als er niet meer dan 8 displays aanwezig zijn, kan worden volstaan met een 3-naar-8-decoder, aangesloten op S0, S1 en S2.

Met S wordt dus voor zowel het toetsenbord als de 7-segment displays de keuze tussen encoded en decoded scan gemaakt.

Met KK (b_1 en b_2) wordt aangegeven wat voor type toetsenbord is aangesloten. Dit behoeft nl. niet beslist een matrix van schakelaars (zoals in fig.4) te zijn. Mogelijk is b.v. een volkomen uitgecodeerd ASCII-toetsenbord. De 8279 behoeft dan geen rijen en kolommen te scannen, maar kan direct wanneer er een toets is ingedrukt, de op R0 t/m R7 aangeboden ASCII-code in de keyboard RAM plaatsen.

In deze les gaan we er steeds van uit, dat het aangesloten toetsenbord een matrix van schakelaars is, die rij voor rij gescanned dient te worden. Hiervoor moet KK de waarde 00 bevatten.

DD (b_3 en b_4) bepaalt de display mode, d.w.z. het aantal 7-segment displays en de volgorde waarin de weer te geven data hierop wordt weergegeven.
In tabel 9 zijn de mogelijkheden voor de display mode vermeld.

DD	display mode
00	8 displays, left entry
01	16 displays, left entry
10	8 displays, right entry
11	16 displays, right entry

Tabel 9

We kunnen dus kiezen of er 8 of 16 displays moeten worden aangestuurd. Dit is van belang bij WRITE DISPLAY, als auto increment is geprogrammeerd. (Dit is reeds in de les "Systeemeigenschappen hardware" behandeld.) Stel we hebben gekozen voor 8 displays. Als er dan data naar het 8e display is gestuurd, zal de volgende data weer op het eerste display verschijnen, i.p.v. het niet bestaande 9e display.

De begrippen left entry (= links binnenkomend) en right entry (= rechts binnenkomend) zullen we verduidelijken aan de hand van een voorbeeld. In fig.24a geldt DD = 00. In fig.24b is DD = 10. In beide gevallen worden dus maximaal 8 displays aangestuurd.

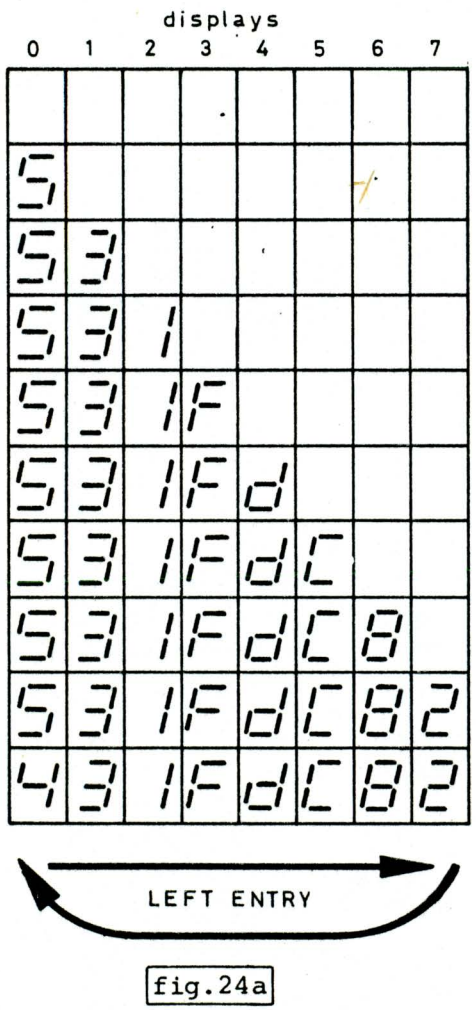


fig.24a

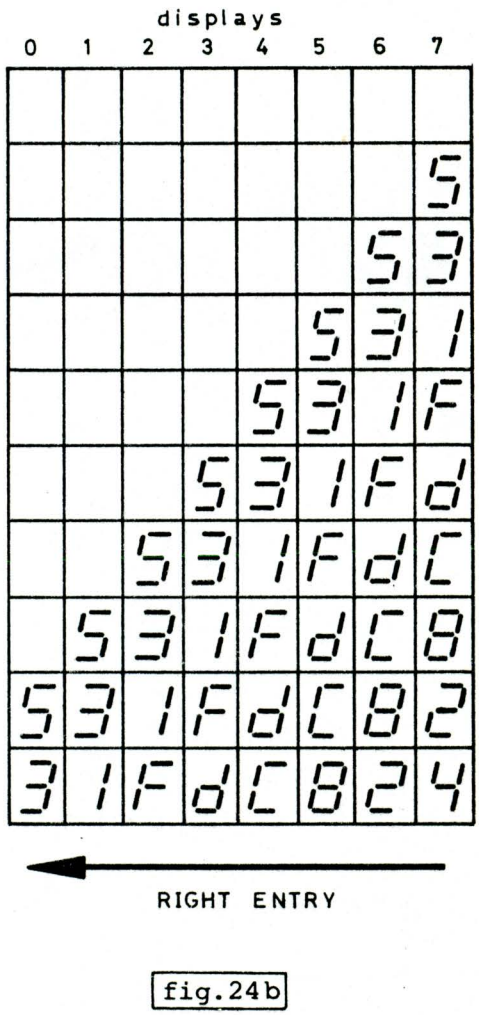


fig.24b

We gaan nu voor beide gevallen bekijken, wat er op de displays wordt weergegeven als achtereenvolgens de 7-segment-codes voor de cijfers 5, 3, 1, F, D, C, 8, 2 en 4 naar de 8279 worden gestuurd. We nemen aan dat er sprake is van auto increment.

In fig.24a is left entry weergegeven. Er wordt aan de linkerkant begonnen. Elk volgend cijfer wordt rechts naast het voorgaande geplaatst. Als zo 8 displays van data zijn voorzien, wordt het 9e cijfer weer op display 0 weergegeven. Het cijfer dat hier stond, gaat dan verloren.

In fig.24b is right entry weergegeven. Er wordt aan de rechterkant begonnen. Elk volgend cijfer wordt op het meest rechtse display geplaatst, terwijl de voorgaande cijfers een plaats naar links schuiven. Door het weergeven van het 9e cijfer gaat het cijfer dat op display 0 stond, verloren.

Beide mogelijkheden kunnen ook voor 16 displays worden toegepast. Dan zal pas door het weergeven van het 17e cijfer de oorspronkelijke data van display 0 verloren gaan.

Vraag 24: Om de 8279 in fig.22 voor left entry te programmeren, moet het command register worden gevuld met $\ldots\ldots_2$.

Als we de 8279 in fig.22 voor left entry willen programmeren, dan moet in het command register (fig.23a) gelden:

FFF = 000, keyboard/display mode
DD = 01 , 16 displays, left entry
KK = 00 , normaal toetsenbord
S = 0 , encoded scan

Het command register moet dan worden gevuld met $00001000_2 = 08_{16}$.

SAMENVATTING 8

24. Voor elk van de 8 mogelijke functies van de 8279 is een afzonderlijk 5-bits command register aanwezig.

25. De eerste 3 bits van een commando geven aan op welke functie dit commando betrekking heeft.

26. D.m.v. het commando keyboard/display mode set wordt aangegeven op welke wijze de 8279 de displays en het toetsenbord moet scannen. In dit commando zitten tevens de maximale aantallen displays en rijlijnen van het toetsenbord besloten.

Antw.24: 00001000.

b. Program clock

De 8279 heeft voor het scannen van de displays, het scannen van het toetsenbord en het opheffen van de contactdender van de schakelaars (= debounce) een interne klokfrequentie nodig van ca. 100 kHz.
Deze frequentie moet worden afgeleid uit het op de CLK-ingang van de 8279 aangeboden kloksignaal.

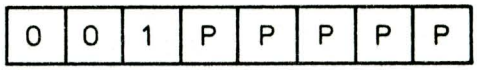


fig.23b

Omdat in elk systeem weer een andere klokfrequentie aanwezig is, is in de 8279 een voorziening aangebracht, om uit frequenties van 200 kHz - 3 MHz een klokfrequentie van ca. 100 kHz af te leiden. Er is in de 8279 nl. een programmeerbare frequentiedeler opgenomen. De deelfactor hiervan wordt geprogrammeerd met de binaire waarde van de 5 bits PPPPP in het commando program clock.

Stel, dat in een 8080-systeem een klokfrequentie van 2 MHz aanwezig is.

Vraag 25: Het commando program clock moet dan zijn $.....{}_2 = ...{}_{16}$.

De deelfactor moet dan de waarde 20 hebben. De 5 bits PPPPP moeten dan de combinatie 10100_2 ($= 20_{10}$) vormen. Het commnando program clock luidt dan $00110100_2 = 34_{16}$.

Vraag 26: De maximale waarde van PPPPP is $.....{}_{10}$.

De bits PPPPP kunnen maximaal de waarde $11111_2 = 31_{10}$ vormen. De 8279 (zonder extra hardware) is dus geschikt voor systemen met klokfrequenties tot ca. 31 x 100 kHz = 3,1 MHz. Als we een externe frequentiedeler toepassen, kunnen we hoger gaan. I.v.m. de access-tijd van de 8279 is de hoogste grens echter ca. 4 MHz.

c. Clear

D.m.v. het commando clear kan de keyboard RAM en/of de display RAM met een bepaalde inhoud worden gevuld.

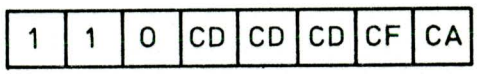


fig.23c

D.m.v. de 3 bits CD wordt de displays RAM geïnitialiseerd. De 16 RAM-locaties worden door dit commando alle met een waarde volgens tabel 10 gevuld.

CD (3x)	inhouden van de 16 RAM-locaties
0XX	blijven ongewijzigd
10X	16 maal 00_{16}
110	16 maal 20_{16}
111	16 maal FF_{16}
X = don't care	

Tabel 10

Antw.25: 00110100; 34. Antw.26: 31.

Het uitvoeren van dit commando duurt ca. 160 µs. De CPU mag na het geven van dit commando gedurende deze 160 µs geen data in de display RAM plaatsen.
Gedurende de uitvoering van het commando clear geldt DU = 1 (b_7 van het statusregister). Zie ook paragraaf 5d.

Met CF (= clear FIFO) kan de gehele keyboard RAM met 0 worden gevuld (CF = 1). Als CF = 0 dan heeft de uitvoering van het commando clear geen invloed op de keyboard RAM.

Als CA (= clear all) 1 is, worden zowel de keyboard RAM als de display RAM geïnitialiseerd, d.w.z. de gehele keyboard RAM wordt met 0 gevuld en de display RAM wordt gevuld met de door b_2 en b_3 (CD) bepaalde waarde. Als CA = 1 dan hebben b_1 (CF) en b_4 (CD) geen invloed.
Als CA = 0 dan is de uitvoering van clear afhankelijk van CD en CF.

d. Read keyboard RAM

Als dit commando is gegeven, dan zal de eerst volgende keer, dat het DATA IN-register wordt aangesproken, de door AAA aangegeven locatie in de keyboard RAM worden uitgelezen.

fig.23d

Er zijn nu 2 mogelijkheden:
1. AI = 0 (auto increment, b_4). Na het uitlezen van de betreffende RAM-locatie blijft de inhoud van AAA ongewijzigd.
2. AI = 1. Na het uitlezen van de betreffende RAM-locatie wordt de inhoud van AAA met 1 verhoogd.

Vraag 27: Als een normaal scanned keyboard is aangesloten, dan moet altijd gelden AI = 0/1 en AAA = $\ldots\ldots_2$.

Als we een normaal scanned keyboard hebben aangesloten en de 8279 hiervoor hebben geprogrammeerd, dan moet in het commando read keyboard RAM altijd gelden AI = 0 en AAA = 000_2. Immers, in deze keyboard mode wordt de inhoud van de laagste RAM-locatie steeds in het DATA IN-register geplaatst (zie paragraaf 5b).

SAMENVATTING 9

27. Het commando program clock dient om de interne frequentiedeler zodanig te programmeren, dat uit de systeemklok een frequentie van ca. 100 kHz wordt afgeleid. Deze 100 kHz is nodig voor de debounce en het multiplexen van de 7-segment displays.
28. D.m.v. het commando clear kan de keyboard RAM en/of de display RAM worden geïnitialiseerd.
29. D.m.v. het commando read keyboard RAM wordt de inhoud van de geadresseerde RAM-locatie in het DATA IN-register geplaatst.

Antw.27: 0; 000.

e. End interrupt/error mode set

Dit commando wordt alleen gegeven als er niet een normaal scanned keyboard op de 8279 is aangesloten.
Op dit commando gaan we in deze les niet in.

fig.23e

f. Read display RAM

Als direct na het geven van dit commando het DATA IN-register wordt geselecteerd, dan kan de inhoud van de door AAAA aangegeven RAM-locatie worden uitgelezen. Afhankelijk van de toestand van de auto increment bit (b4) wordt daarna de inhoud van AAAA niet veranderd (AI = 0) of met 1 verhoogd (AI = 1). Door dit commando is het dus mogelijk, om de naar een bepaald display gezonden 7-segment-code terug te lezen.

fig.23f

g. Write display RAM

Dit commando dient om de locaties in de display RAM met de gewenste 7-segment-codes te vullen. De beschrijving van dit commando is al in paragraaf 9 van de les "Systeemeigenschappen hardware" gegeven.
Lees deze zonodig nog eens door.

fig.23g

Opmerking:

De bits AAAA in fig. 23f en 23g beïnvloeden elkaar niet. Ze zijn immers in afzonderlijke 5-bits command registers opgeslagen (zie fig.21).

h. Display write inhibit/blanking

De 8 lijnen, die met alle segmenten van de displays zijn verbonden, zijn verdeeld in 2 groepen van 4 lijnen, nl. A0 t/m A3 en B0 t/m B3.
Dit commando biedt de mogelijkheid om beide groepen afzonderlijk te besturen. In onze toepassingen moeten IWA en IWB resp. BLA en BLB steeds gelijk zijn, omdat we alle segmenten van een display op dezelfde wijze willen behandelen.

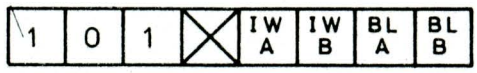


fig.23h

Als IWA en IWB beide 1 zijn, dan heeft het commando write display RAM geen invloed op de inhoud van de door AAAA geadresseerde RAM-locatie. IW betekent nl. write inhibit (= schrijven verboden). Dit is van belang in die gevallen, waarin we de inhoud van een locatie binnen de display RAM, dus de op een bepaald display weergegeven data, niet willen laten wijzigen, ongeacht de door de CPU afgegeven 7-segment-code.

Als BLA en BLB beide 1 zijn, dan heeft het commando write display RAM geen invloed op het door AAAA geadresseerde display. Het betreffende display zal dan nl. gedoofd zijn, ongeacht de inhoud van de bijbehorende locatie in de display RAM. BL is een afkorting van blanking (= doven). Dit wordt toegepast als we een bepaald display willen doven, ongeacht de 7-segment-code, die de CPU hiervoor naar de display RAM stuurt.

SAMENVATTING 10

30. D.m.v. het commando write display RAM wordt de data, die de CPU in het DATA OUT-register plaatst, overgebracht naar de gewenste locatie in de display RAM.

31. D.m.v. het commando read display RAM wordt de inhoud van de geadresseerde locatie binnen de display RAM naar het DATA IN-register overgebracht.

32. D.m.v. het commando display write inhibit/blanking kan
    a. een bepaalde RAM-locatie ongewijzigd blijven, ongeacht de data, die de CPU hierin wil plaatsen.
    b. een bepaald display worden gedoofd, ongeacht de inhoud van de bijbehorende display RAM-locatie.

7. CODERING VAN TOETSEN

In deze paragraaf bespreken we hoe we, door het juist aansluiten van een scanned keyboard, direct de gewenste code voor een ingedrukte toets in de keyboard RAM kunnen laten plaatsen.

We gaan weer even uit van fig.22. Op het moment dat er een ingedrukte toets wordt gedetecteerd, ontstaat in de 8279 de volgende code:

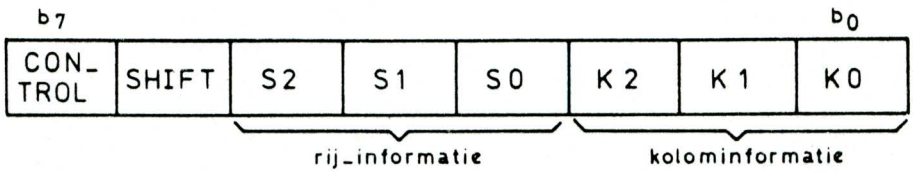


fig.25

b_7 en b_6 bevatten de toestand van de control- en de shift-lijn.

b_5, b_4 en b_3 geven de toestand van de drie scan-lijnen weer, zoals deze is bij het detecteren van een ingedrukte toets.

b_2, b_1 en b_0 bevatten de kolom-informatie, d.w.z. het binaire nummer van de kolom, waarin de ingedrukte toets zich bevindt.
Deze 3-bits informatie wordt afgeleid uit de toestanden van de 8 kolomlijnen R0 t/m R7 (tabel 11).
Laat u zich door de aanduidingen R0 t/m R7 niet in verwarring brengen. Het zijn wel degelijk kolomlijnen. De ontwerper van de 8279 heeft deze met R aangeduid (R is een afkorting van retourlijn).

Kolomlijnen (retourlijnen)								Kolom-informatie		
R_7	R_6	R_5	R_4	R_3	R_2	R_1	R_0	K2	K1	K0
0	0	0	0	0	0	0	0	X	X	X
0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	1	0	0	0	1
0	0	0	0	0	1	0	0	0	1	0
0	0	0	0	1	0	0	0	0	1	1
0	0	0	1	0	0	0	0	1	0	0
0	0	1	0	0	0	0	0	1	0	1
0	1	0	0	0	0	0	0	1	1	0
1	0	0	0	0	0	0	0	1	1	1
X = don't care										

Tabel 11

In fig.26 is weergegeven hoe een hexadecimaal toetsenbord op de 8279 is aangesloten.

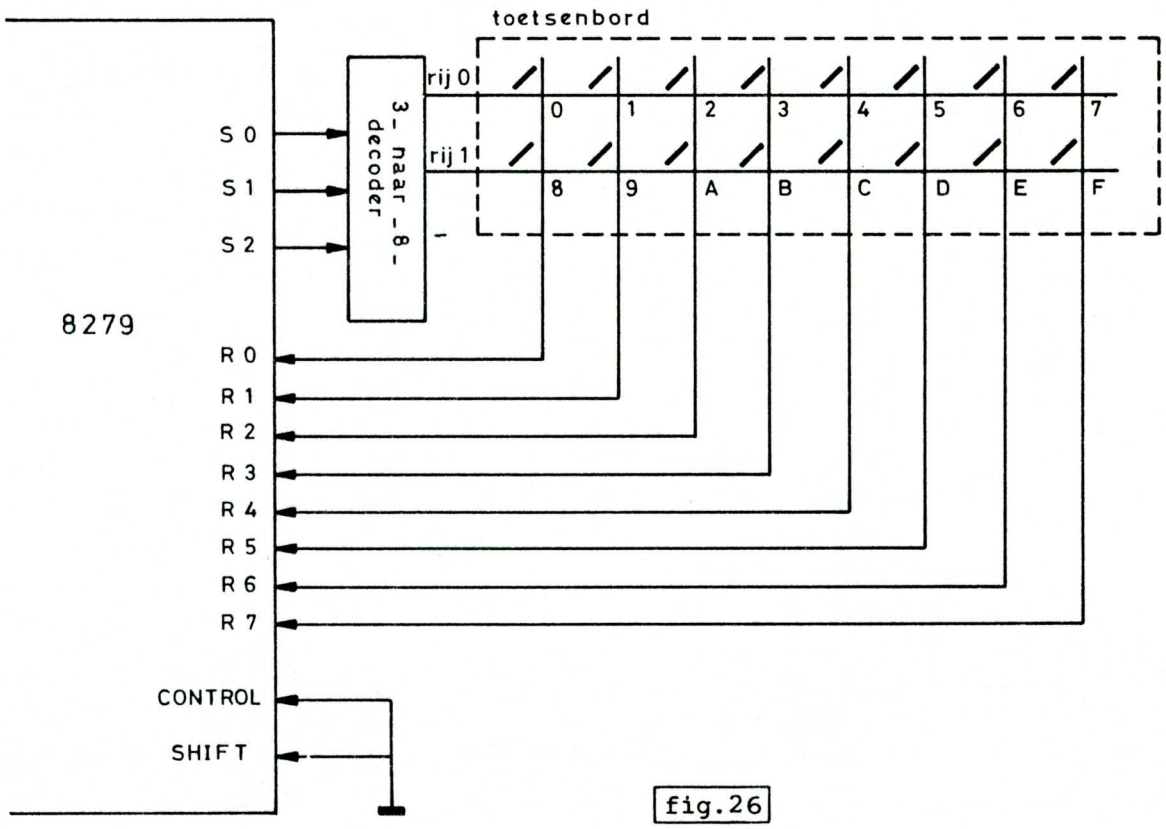


fig.26

Het toetsenbord is nu verdeeld in 2 rijen en 8 kolommen. Van de 3-naar-8 decoder worden alleen rij 0 en rij 1 gebruikt. Control en shift worden niet gebruikt. Deze lijnen zijn met 0 verbonden.

Stel nu dat toets B wordt ingedrukt.

Vraag 28: Bij het detecteren van deze toets vormen S2, S1 en S0 de combinatie $\ldots\ldots_2$. Op de retourlijnen R7 t/m R0 wordt de combinatie $\ldots\ldots_2$ ontvangen. De 8279 plaatst dan de combinatie $\ldots\ldots_2$ = $\ldots\ldots_{16}$ in de keyboard RAM.

Als dit detecteerd wordt, vormen S2, S1 en S0 de combinatie 001_2, want toets B bevindt zich in rij 1. Op retourlijn R3 staat dan een 1.
De lijnen R7 t/m R0 vormen dan de combinatie 00001000_2. Dit wordt in de 8279 omgezet in de combinatie 011 (zie tabel 11).
Control en shift zijn 0. De 8279 plaatst dan in de eerst volgende vrije locatie in de keyboard RAM de combinatie $00001011_2 = 0B_{16}$.

Ook voor alle andere ingedrukte toetsen wordt direct de bijbehorende binaire waarde in de keyboard RAM geplaatst (ga dit voor uzelf na).

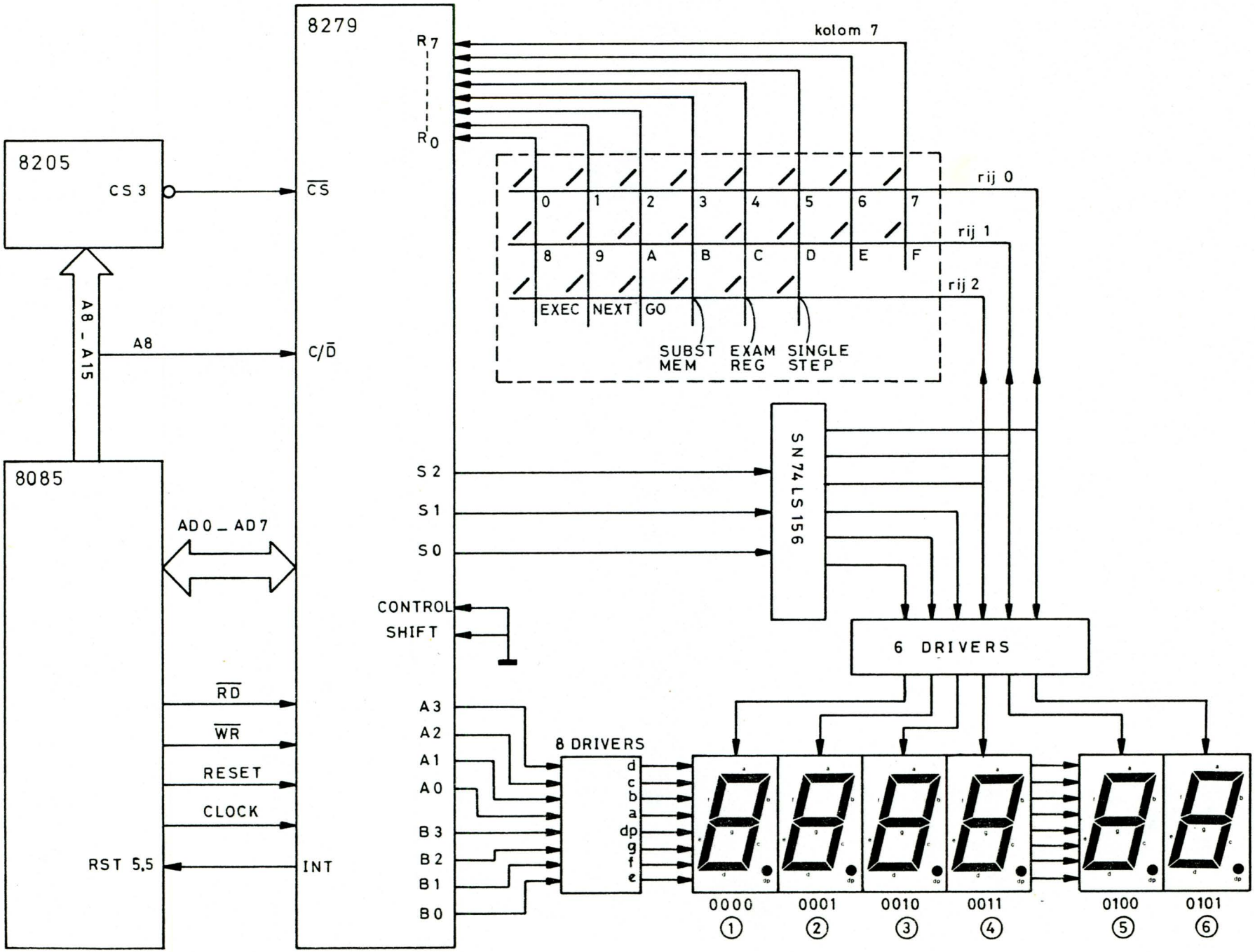

8279
R7
R0
kolom 7
8205
CS 3
CS
rij 0
rij 1
rij 2
0 1 2 3 4 5 6 7
8 9 A B C D E F
EXEC NEXT GO
SUBST MEM
EXAM REG
SINGLE STEP
A8 - A15
A8
C/D
8085
S2
S1
S0
SN74LS156
AD0 _ AD7
CONTROL
SHIFT
6 DRIVERS
RD
WR
RESET
CLOCK
A3
A2
A1
A0
B3
B2
B1
B0
8 DRIVERS
d
c
b
a
dp
g
f
e
RST 5,5
INT
0000
0001
0010
0011
0100
0101
1
2
3
4
5
6

fig.27

Door de schakelaars op een andere wijze met elkaar te verbinden, of op andere rijlijnen aan te sluiten, kunnen we ook andere codes in de keyboard RAM laten plaatsen.

Opmerking:

Wanneer er een of meer locaties in de keyboard RAM zijn gevuld, dan wordt de INT-uitgang van de 8279 actief (1). Steeds wanneer een RAM-locatie wordt uitgelezen, wordt de INT-lijn 0. Als daarna blijkt dat er nog meer data in de keyboard RAM aanwezig is, wordt de INT-lijn direct weer 1.

8. DE 8279 IN DE SDK 85

In fig.27 is weergegeven hoe de 8279 in de SDK 85 is aangesloten.
Om in dit systeem de 8279 op de gewenste wijze te initialiseren, zijn echter geen commando's nodig. Door een impuls op de RESET-ingang wordt de 8279 nl. geprogrammeerd voor een bepaalde toepassing.
Aangezien deze toepassing volkomen overeenstemt met de eigenschappen van de SDK 85 behoeven we de initialisatie niet meer d.m.v. nieuwe commando's te wijzigen.

In de MPC-opgaven wordt op het gebruik van de 8279 nader ingegaan.

Opmerking:

Omdat voor zowel het toetsenbord als de 7-segment displays in de SDK 85 maar 3 scan-lijnen behoeven te worden gebruikt, is er een gemeenschappelijke 3-naar-8-decoder (type SN74LS156) toegepast.

SAMENVATTING 11

33. De 8279 stelt de code voor een ingedrukte toets samen uit de toestanden van de control- en shift-lijnen, de rij-informatie en de kolominformatie.

34. De rij-informatie bestaat uit de toestanden van de 3 scan-lijnen S2, S1 en S0.
De kolominformatie wordt afgeleid uit de op de retourlijnen ontvangen bitcombinatie.

35. Door het correct aansluiten van de schakelaars, kunnen we voor elke ingedrukte toets een gewenste code in de keyboard RAM laten opslaan.

INTERFACING-3.

Interfacing-3

1. INLEIDING

In deze les bespreken we een methode voor het plegen van asynchrone serial I/O volgens de z.g. Kansas City Standard. We doen dit aan de hand van een voorbeeld met een cassetterecorder interface. Dit had echter elk willekeurig randapparaat mogen zijn, dat d.m.v. asynchrone serial I/O met de computer communiceert.

Het probleem bij asynchrone serial I/O ligt nl. niet in het omzetten van digitale serie-informatie naar een signaal volgens de Kansas City Standard. Hiervoor bestaan zeer eenvoudige interface-schakelingen. Het punt waar het hierbij om draait, is het omzetten van de (8 bits) parallel-informatie in serie-informatie (en omgekeerd).

In deze les gaan we de serie-parallel-omzetting en de parallel-serie-omzetting software-matig realiseren.

We beginnen met serial I/O via parallel I/O-poorten. Daarna bekijken we welke wijzigingen dienen te worden aangebracht om deze I/O via de SID- en SOD-lijnen van de 8085 microprocessor te laten verlopen.

We beperken ons hierbij tot het programma dat nodig is voor het zenden resp. ontvangen van één 8-bits karakter. Voor het transport van een heel blok data moet dit programma dan een groot aantal malen worden uitgevoerd. Om u in staat te stellen de in deze les beschreven programma's te testen, gaan we steeds uit van de eigenschappen en mogelijkheden van de SDK 85.

Uit deze les zal u duidelijk worden, dat het probleem bij interfacing niet alleen in de hardware (de interface-schakeling) ligt, maar vooral in de software.

2. BESCHRIJVING VAN DE INTERFACE

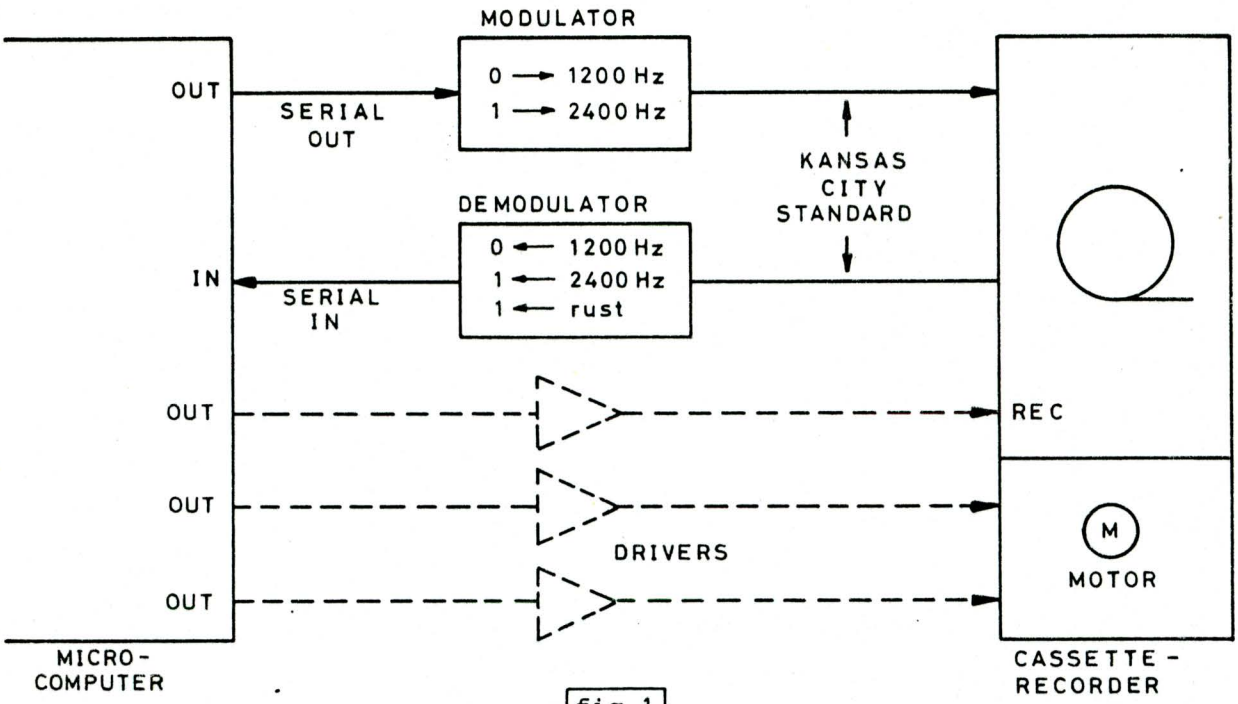


fig.1

In fig.1 is het hardware-gedeelte van de interface weergegeven.

Het SERIAL OUT-signaal, dat door de computer wordt afgegeven, wordt door de modulator omgezet in een signaal volgens de Kansas City Standard. D.w.z. dat een 0 wordt omgezet in een wisselspanning met een frequentie van 1200 Hz. Een door de microcomputer afgegeven 1 wordt omgezet in een 2400 Hz-wisselspanning (zie fig.2). Deze frequenties kunnen zonder meer op een cassette worden opgenomen.

Wanneer het datatransport in omgekeerde richting verloopt, dan worden de van de cassetterecorder ontvangen wisselspanningen door de demodulator omgezet in 0 (bij 1200 Hz) en 1 (bij 2400 Hz).

Als de cassetterecorder stil staat, of als er een leeg stuk magneetband wordt afgespeeld, ontvangt de demodulator geen signaal (rusttoestand). De demodulator plaatst dan continu een 1 op de SERIAL IN-lijn.

De meeste audio-cassetterecorders kunnen alleen worden bediend d.m.v. de op het apparaat aanwezige toetsen. Wanneer u zo'n cassetterecorder als extern geheugen toepast, dan zult u deze steeds met de hand moeten bedienen.

Enkele duurdere typen audio-cassetterecorders en de speciaal voor dit doel gefabriceerde datarecorders, hebben een mogelijkheid tot afstands-

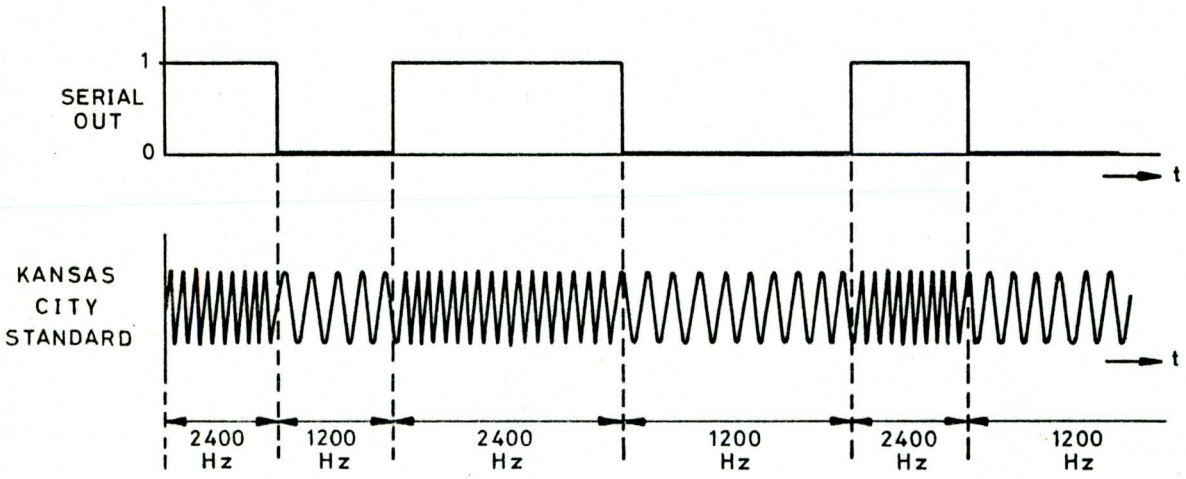


fig.2

bediening (remote control).
De cassetterecorder kan in dat geval software-matig vanuit de computer worden bestuurd. Dit gebeurt dan door het gestippelde gedeelte van fig.1.

3. BESCHRIJVING VAN HET SERIE-KARAKTER

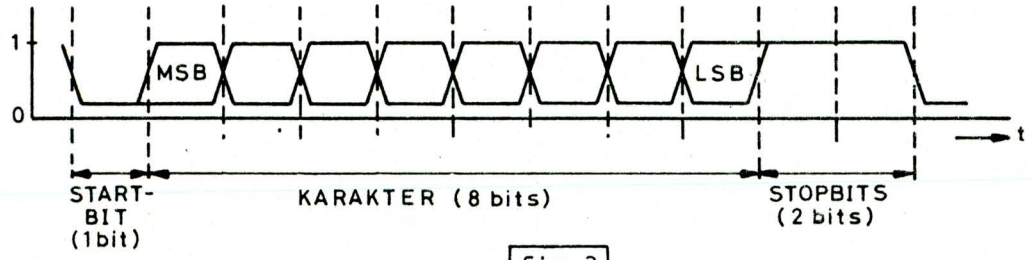


fig.3

Bij asynchrone serial I/O wordt een serie-karakter ingesloten door start- en stopbits. In deze les gaan we uit van 8-bits karakters, die met 1 startbit beginnen en met 2 stopbits eindigen.

Een startbit is 0, de stopbits zijn 1. Wanneer er geen karakter wordt verzonden (rusttoestand), is de lijn 1.

Vraag 1: Het serie-karakter van fig.3 wordt wel/niet direct gevolgd door een volgend karakter.

In fig.3 treedt na de 2 stopbits een 0 op. Dit moet dus een startbit van een volgend karakter zijn.

De transmissiesnelheid (baud rate), die we in deze les aanhouden, is 110 baud (bits/s).

Vraag 2: Dit komt overeen met ..... karakters per seconde.

Aangezien een serie-karakter uit 11 bits (zie fig.3) bestaat, komt 110 baud overeen met 10 karakters per seconde.

Van een karakter, dat naar de cassetterecorder wordt weggeschreven, wordt eerst de MSB (b_7) verzonden, dan b_6, b_5, enz. en als laatste de LSB (b_0).

Vraag 3: Van een karakter, dat van de cassetterecorder wordt ontvangen, komt eerst de MSB/LSB binnen.

De cassetterecorder geeft de bits van een karakter in dezelfde volgorde af, dus na de startbit volgt eerst de MSB.

Antw.1: wel. Antw.2: 10 . Antw.3: MSB.

4. PARALLEL-SERIE-OMZETTING

Wanneer er een karakter naar de cassetterecorder moet worden gezonden, moet er achtereenvolgens plaatsvinden:

a. Er wordt een startbit (0) op de SERIAL OUT-lijn geplaatst.
b. Dan worden de afzonderlijke bits van het karakter na elkaar op de SERIAL OUT-lijn geplaatst.
c. Dan worden er twee stopbits (beide 1) op de SERIAL OUT-lijn geplaatst.

Dit is schematisch in fig.4 weergegeven.
Hierin hebben we een willekeurig karakter ($10100110_2 = A6_{16}$) gekozen.

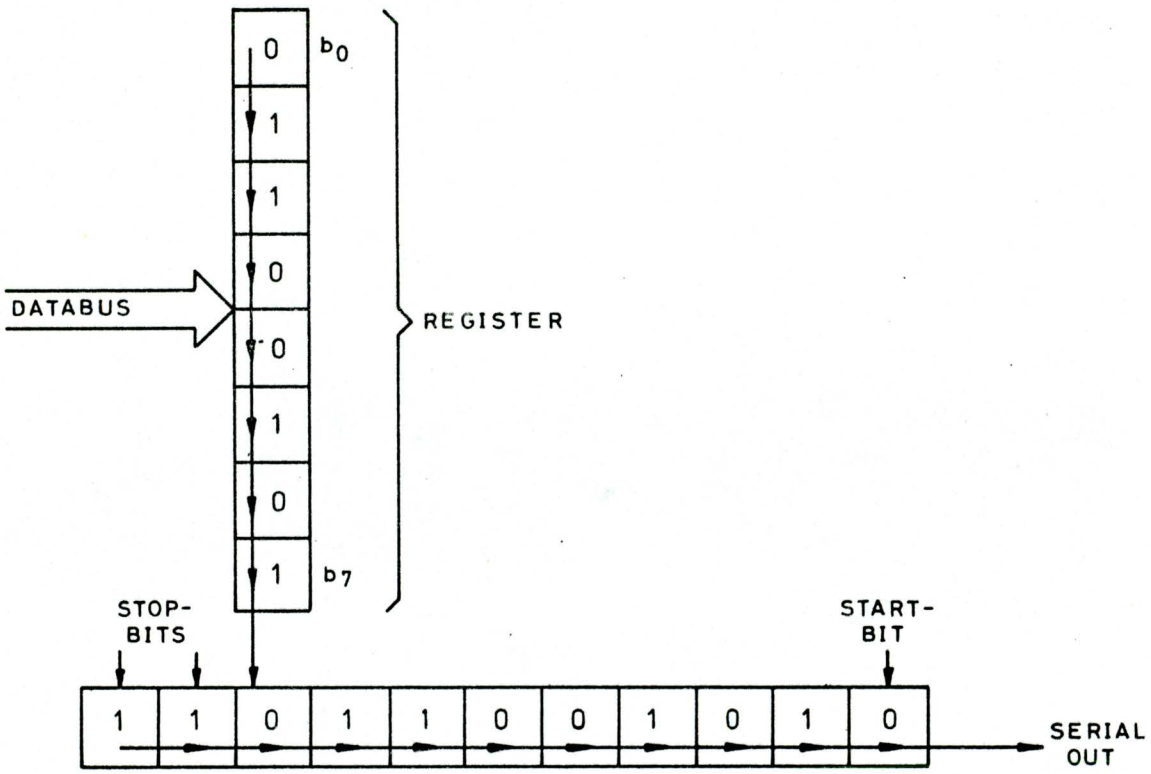


fig.4

Vraag 4: In fig.4 moet elke bit gedurende ca. 1/9,1/10 ms op de SERIAL OUT-lijn aanwezig zijn.

De baud rate is 110 baud = 110 bits/s.
Voor 1 bit is dan $\frac{1}{110}$ s = $\frac{1000}{110}$ ms = ca. 9,1 ms gereserveerd.
Dit houdt in, dat na het plaatsen van een bit op de SERIAL OUT-lijn er ca. 9,1 ms moet worden gewacht, voordat de volgende bit op deze lijn mag worden geplaatst.

Antw.4: 9,1.

Fig.5 is het algemeen stroomdiagram voor het overzenden van een volledig karakter.
Dit stroomdiagram heeft de naam "SOUK" (serial out-karakter) gekregen.

In fig.5 vallen de volgende feiten op:

a. Als een stopbit op de SERIAL OUT-lijn is geplaatst, wordt er 18,2 ms gewacht. Er worden eigenlijk niet twee stopbits van 9,1 ms uitgevoerd, maar slechts één stopbit met dubbele bitlengte. Dit komt natuurlijk op hetzelfde neer.
b. Het laatste blok is een return-opdracht. "SOUK" is dus een subroutine. Dit is gedaan omdat we altijd een blok data naar de cassetterecorder wegschrijven. Voor elk over te zenden karakter kunnen we dan eenvoudig "SOUK" aanroepen. De taak van het hoofdprogramma is dan te bepalen hoevaak "SOUK" dient te worden aangeroepen.

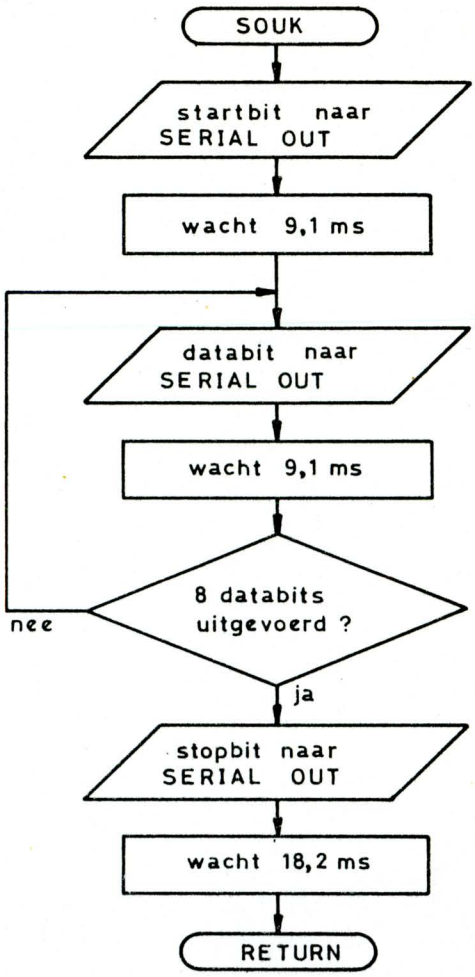


fig.5

5. SERIE-PARALLEL-OMZETTING

Wanneer de cassetterecorder op weergeven is geschakeld, worden door de demodulator énen en nullen op de SERIAL IN-lijn geplaatst.
Aangezien we met asynchrone serial I/O werken, is het de taak van de microcomputer om deze reeks énen en nullen in complete karakters om te zetten.
Om één karakter in te lezen, moet achtereenvolgens

a. een startbit worden gedetecteerd.
b. de 8 volgende ontvangen bits tot een 8-bits parallel-karakter worden samengesteld.
c. twee stopbits worden gedetecteerd.

Dit is schematisch in fig.6 weergegeven.

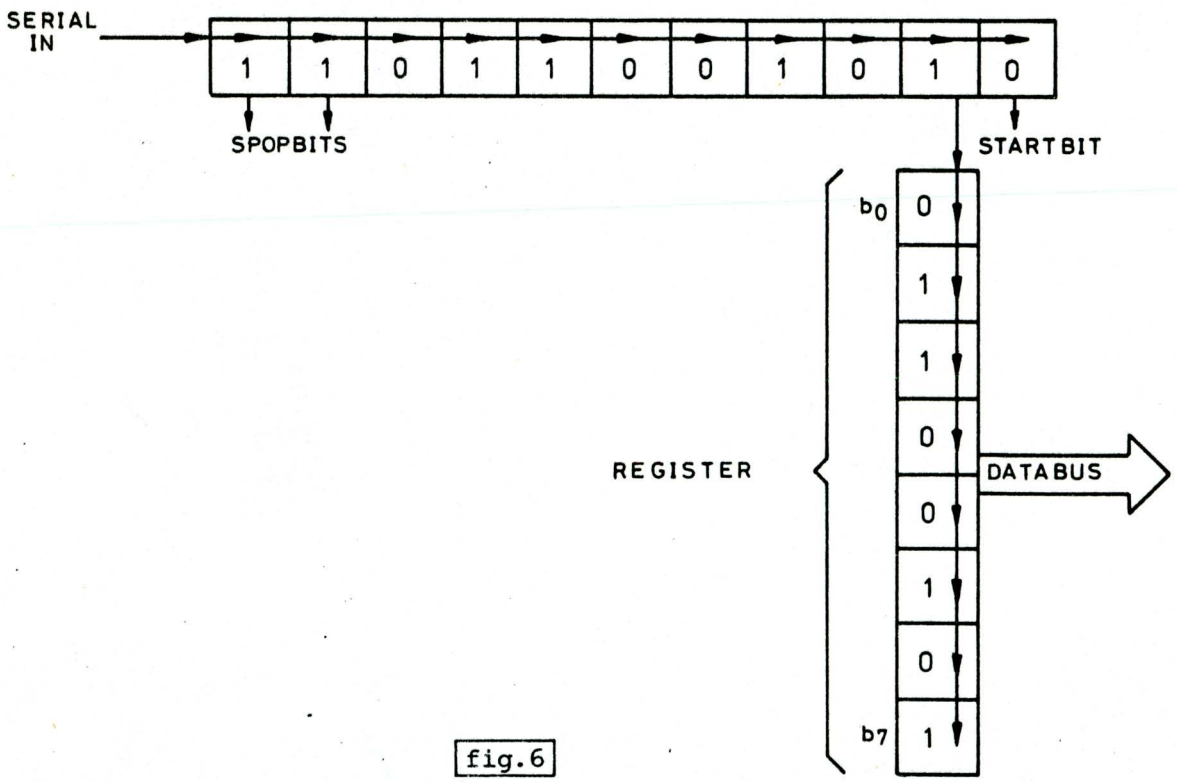


fig.6

In fig.7 is weergegeven op welke momenten de toestand van de SERIAL IN-lijn dan moet worden getest.

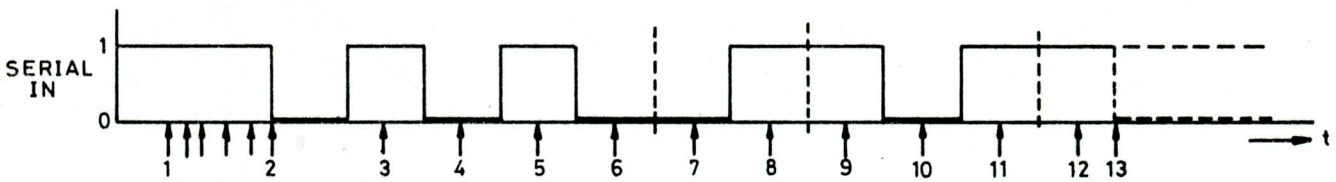


fig.7

Van t=1 tot t=2 wordt de SERIAL IN-lijn getest. Deze lijn is steeds 1, dus er wordt geen startbit gedetecteerd en de lijn moet opnieuw worden getest.

Op t=2 wordt het begin van een startbit gedetecteerd. De SERIAL IN-lijn is dan nl. 0 geworden. Daarna kan de eerste databit worden binnengehaald.

Vraag 5: Tussen t=2 en t=3 ligt minimaal 4,55/9,1/13,65 ms.

Wanneer het begin van een startbit is gedetecteerd, moet minimaal 9,1 ms worden gewacht. Dit is immers de lengte van de startbit. Dan mag de SERIAL IN-lijn pas weer worden getest om de toestand van de eerste databit vast te stellen.

Stel, dat we na t=2 de minimale tijd van 9,1 ms wachten. Dan valt t=3 precies na het begin van de eerste databit, dus direct na het eind van de startbit. Theoretisch is dit volkomen correct.

Praktisch gezien is dit echter niet een ideale oplossing. Als de startbit nl. 10 µs langer duurt dan was berekend, dan zal op t=3 i.p.v. 1 een 0 worden gedetecteerd. In dit geval is er al direct een foutieve waarde voor de eerste databit ingevoerd. Deze fout zal zich verder voortzetten bij het detecteren van alle volgende databits.

Deze fout is te wijten aan een afwijking van 10 µs. Dit is slechts een afwijking van ca. 0,1 % t.o.v. de berekende 9,1 ms. Een afwijking in deze grootte-orde is beslist niet denkbeeldig. Oorzaken hiervan kunnen nl. zijn:

a. Variaties in de bandsnelheid van de cassetterecorder tijdens het opnemen of weergeven van de data. Bij een niet al te slechte recorder liggen deze variaties tussen 0,1 en 0,5 %.
b. Het niet helemaal exact berekend hebben van de wachtlussen in de programma's voor de serial input en output. Deze afwijkingen zijn echter te verwaarlozen t.o.v. de onder punt a genoemde.

I.v.m. de genoemde afwijkingen, laten we t=3 precies in het midden van de eerste databit vallen.

Vraag 6: Tussen t=2 en t=3 ligt dan ..... ms.

Na het detecteren van het begin van de startbit moet dan 9,1 + 4,55 = 13,65 ms worden gewacht, voordat de toestand van de eerste databit kan worden getest.

Vraag 7: Tussen t=3 en t=4 ligt dan ..... ms.

Om dezelfde reden worden ook de toestanden van de volgende 7 databits "in het midden" getest. De tijdstippen t=3 t/m t=10 liggen dus steeds 9,1 ms uit elkaar.

Vraag 8: Na t=10 moet ..... ms worden gewacht.

Als de laatste van de 8 databits is gedetecteerd, moet weer 9,1 ms worden gewacht, voordat de eerste stopbit (weer in het midden) kan worden gedetecteerd. Tussen t=10 en t=11 ligt dus ook 9,1 ms. Hetzelfde geldt voor de tijd tussen t=11 en t=12.

Vraag 9: Op t=11 en t=12 moet een 0/1 worden gedetecteerd.

Op t=11 en op t=12 moeten stopbits worden gedetecteerd. Deze zijn 1. Als er op een van deze tijdstippen een 0 wordt gedetecteerd, dan moet er een foutmelding worden gegeven.

Antw.6: 13,65. Antw.7: 9,1. Antw.8: 9,1. Antw.9: 1.

Vraag 10: Na t=12 moet er wel/niet 4,55 ms worden gewacht.

Tussen t=12 en t=13 ligt 4,55 ms. Dit is nl. de halve bit-lengte. Het is echter niet raadzaam om na t=12 eerst 4,55 ms te wachten, voordat het begin van een startbit van een eventueel volgend karakter kan worden gedetecteerd. I.v.m. de mogelijke afwijkingen, kan het nl. voorkomen, dat de negatieve flank op de SERIAL IN-lijn binnen deze 4,55 ms valt.
De juiste methode is die, waarbij direct na t=12 opnieuw wordt begonnen met het detecteren van een startbit van een eventueel volgend serie-karakter.
In de tijd tussen t=12 en t=13 gebeurt dan hetzelfde als in de tijd tussen t=1 en t=2.

Het voordeel van deze methode is het feit dat bij elk karakter het werkelijke begin van de startbit wordt gedetecteerd. Wanneer b.v. door variaties in de bandsnelheid de timing van de cassetterecorder af gaat wijken van de timing van de microcomputer, dan zal de afwijking die ontstaat bij het invoeren van een serie-karakter geen gevolgen hebben voor het detecteren van een volgend karakter. M.a.w. op deze wijze worden de microcomputer en cassetterecorder aan het begin van elk karakter opnieuw gesynchroniseerd.

Uit de voorgaande beschouwing volgt nu het algemeen stroomdiagram "SINK" (= serial in karakter) van fig.8.
Evenals "SOUK" is "SINK" als subroutine opgezet.

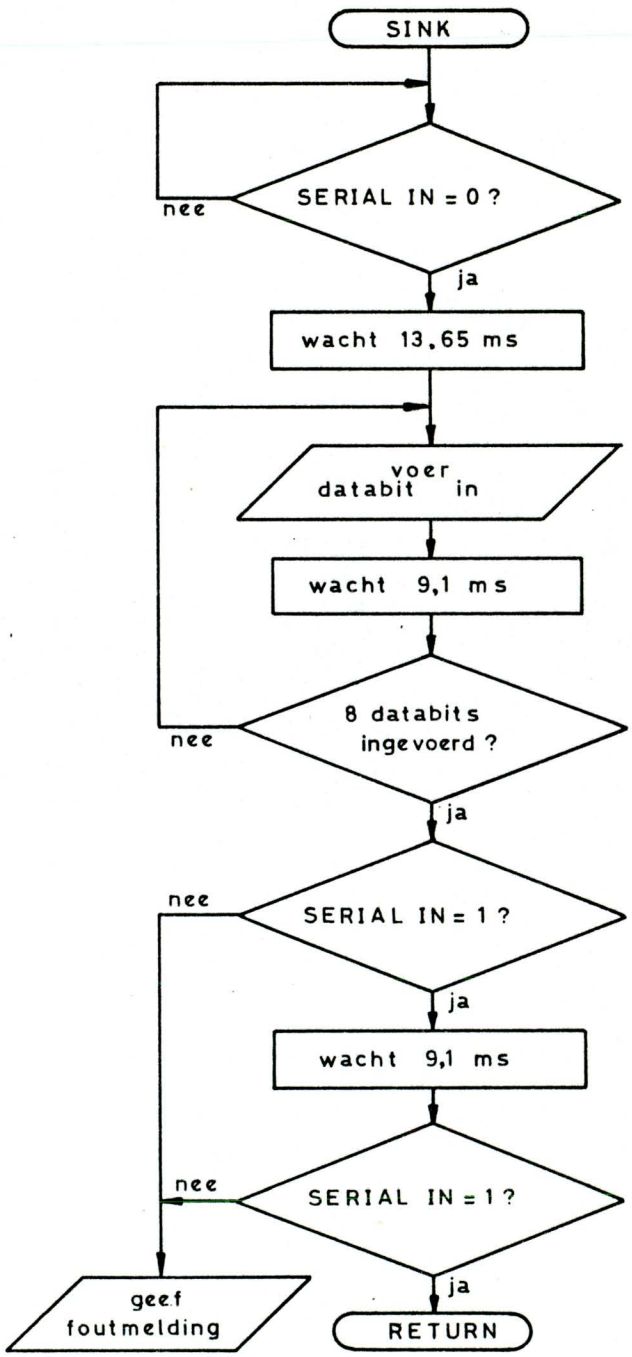


fig.8

In fig.8 is tevens aangegeven, dat er een foutmelding moet worden gegeven, als er niet 2 stopbits worden gedetecteerd. Wat er na zo'n foutmelding verder moet gebeuren, hangt af van de eisen van de gebruiker en de opbouw van het systeem. Een mogelijkheid bij het toepassen van een volledig door de computer bestuurde cassetterecorder is, dat na een foutmelding de band wordt teruggespoeld om het betreffende blok data opnieuw in te lezen.

 Antw.10: niet.

6. TE GEBRUIKEN I/O-LIJNEN

In de voorgaande paragrafen zijn de algemene stroomdiagrammen voor "SOUK" en "SINK" ontwikkeld. Voordat we hieruit gedetailleerde stroomdiagrammen en programma's kunnen afleiden, moeten eerst enkele machinegerichte details worden vastgelegd. Bovendien moeten we onderzoeken welke nauwkeurigheid voor de berekende tijden dient te worden aangehouden. Dit laatste wordt in paragraaf 7 behandeld.

We gaan nu verder uit van de microcomputer SDK 85 met een klokfrequentie van 3 MHz.

Voor de SERIAL OUT-lijn kunnen we een lijn van een output-poort of de SOD-lijn van de CPU nemen. In paragraaf 8 bespreken we een programma, waarin b_3 van output-poort 21_{16} als SERIAL OUT wordt toegepast.
In paragraaf 11 bekijken we, welke wijzigingen moeten worden aangebracht, als hiervoor de SOD-lijn wordt gebruikt.

Ook voor de SERIAL IN-lijn zijn er meer mogelijkheden.
In paragraaf 9 wordt b_5 van input-poort 00_{16} als SERIAL IN toegepast.
In paragraaf 12 wordt dan beschreven, welke wijzigingen noodzakelijk zijn, als we de SID-lijn gebruiken.

De keuze van de poort- en bitnummers was vrij willekeurig.
We hebben ons alleen laten leiden door het feit, dat in deze cursus poort 00_{16} steeds als input-poort en poort 21_{16} steeds als output-poort is toegepast. U kunt hiervoor ook twee andere I/O-lijnen kiezen, b.v. beide aan één I/O-poort van de 8355 (in deze chip is nl. voor elke I/O-lijn afzonderlijk aan te geven of deze als input- of als output-lijn moet fungeren).

7. NAUWKEURIGHEID

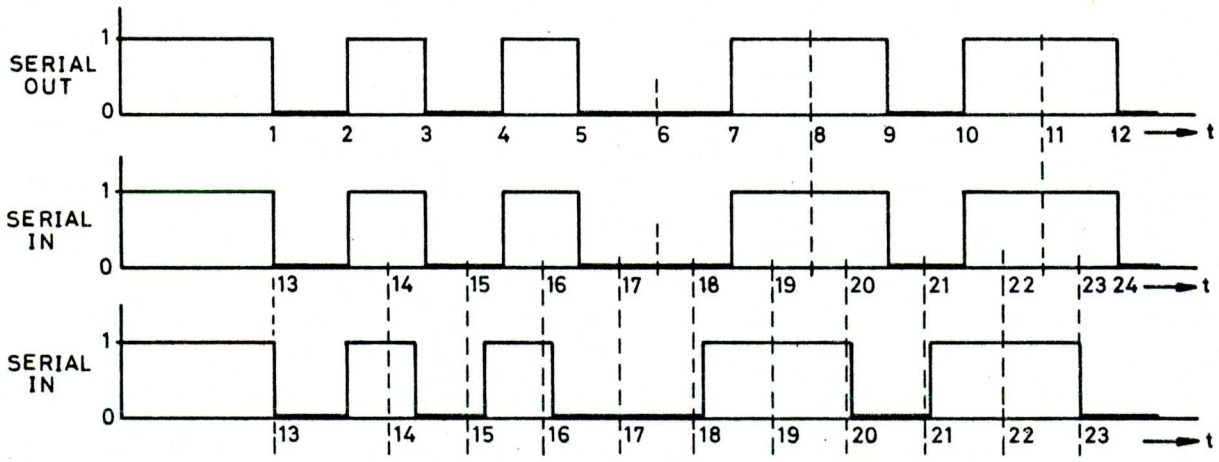


fig.9

In fig.9 zijn de golfvormen op de SERIAL OUT- en SERIAL IN-lijn weergegeven voor het naar de cassetterecorder wegschrijven en naderhand weer teruglezen van het karakter 10100110_2.

De bovenste golfvorm geldt voor het geval dat het karakter met exact de juiste snelheid (110 baud) naar de cassetterecorder wordt gestuurd. t=1 t/m t=11 zijn de tijdstippen, waarop de microcomputer een nieuwe bit op de SERIAL OUT-lijn plaatst. Op t=12 begint de startbit van een volgend karakter.

De middelste golfvorm vindt veel later plaats omdat deze betrekking heeft op het naderhand inlezen van een karakter dat eerder uitgevoerd is. Voor de overzichtelijkheid is deze recht onder de voorgaande getekend.
In deze golfvorm is weergegeven, dat het karakter precies zo wordt ontvangen als het werd weggeschreven. Op t=13 wordt de startbit gedetecteerd. De tijdstippen t=14 t/m t=23 vallen exact in het midden van de ontvangen bits. U ziet, dat er hier niets fout gaat.

In de onderste golfvorm loopt de band van de cassetterecorder te snel. Het ontvangen karakter zal dan ook met een hogere snelheid binnenkomen. De tijdstippen t=14 t/m t=23 liggen nog even ver uit elkaar als in de middelste golfvorm. Dit is logisch, omdat deze tijdstippen in beide gevallen door hetzelfde programma worden bepaald.
De onderste golfvorm geldt voor de maximaal toelaatbare afwijking. Het tijdstip t=23 valt nog net voor het einde van de tweede stopbit.

Vraag 11: De maximale afwijking in de onderste golfvorm is 0,25/0,5/1 maal de lengte van een bit. Dit komt overeen met ca. 4,5/9/10 %.

In het ideale geval zou het einde van de tweede stopbit op t=24 vallen. In de onderste golfvorm gebeurt dit al op t=23, dus een halve bitlengte eerder.
Op een totaal van 11 bits is dit $\frac{0,5}{11}$ = ca. 4,5 %.

Dit houdt niet in, dat de maximale variatie in de bandsnelheid van de cassetterecorder ook 4,5 % mag zijn. Stel b.v. dat bij het opnemen van een karakter de cassetterecorder 1 % te snel en bij het weergeven 1 % te langzaam loopt.

Vraag 12: De afwijking tussen de signalen op de SERIAL OUT- en SERIAL IN-lijn is dan ca. 0,5/1/2 %.

De afwijking is dan ongeveer 2 x 1 % = 2 %. Als dus de maximaal toelaatbare afwijking van het SERIAL IN-signaal t.o.v. het SERIAL OUT-signaal 4,5 % is, dan mag de bandsnelheid niet meer dan 2,25 % variëren.

Nu is het niet zo, dat de afwijkingen alleen door de cassetterecorder worden veroorzaakt.
Ook de modulator en de demodulator (fig.1) zijn verantwoordelijk voor afwijkingen.
Ze bezitten nl. een zekere traagheid, d.w.z. dat het uitgangssignaal vertraagd reageert op het ingangssignaal.
In fig.10 zijn de gevolgen hiervan weergegeven.

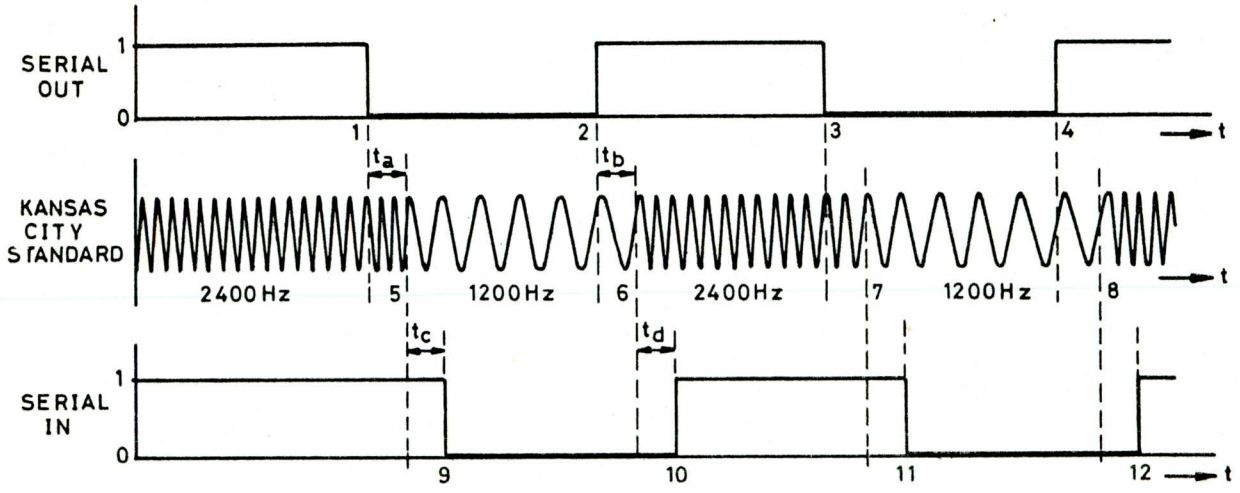


fig.10

In de bovenste golfvorm van fig.10 is het signaal op de SERIAL OUT-lijn, dus het ingangssignaal van de modulator, weergegeven.

De middelste golfvorm geeft het signaal volgens Kansas City Standard weer. Dit door de modulator afgegeven signaal is iets vertraagd t.o.v. het SERIAL OUT-signaal. M.a.w. t=5 valt wat later dan t=1 en t=6 valt later dan t=2. Deze vertragingstijden noemen we t_a resp. t_b.

Als we de middelste golfvorm nu beschouwen als het ingangssignaal voor de demodulator, dan zal door de traagheid hiervan, ook het signaal op de SERIAL IN-lijn weer vertraagd doorkomen. De vertragingstijden noemen we t_c en t_d.

Als nu geldt $t_a = t_b$ en $t_c = t_d$ dan is er geen enkel probleem.

Vraag 13: De tijd tussen t=1 en t=2 is dan wel/niet gelijk aan de tijd tussen t=9 en t=10.

Aangezien de tijd tussen t=1 en t=9 exact gelijk is aan die tussen t=2 en t=10 (immers $t_a + t_c = t_b + t_d$), geldt dus dat de startbit in de onderste en bovenste golfvormen precies even lang zijn. Dit geldt ook voor alle hierna volgende bits, zodat het signaal op de SERIAL IN-lijn gelijk is aan dat op de SERIAL OUT-lijn. M.a.w. er wordt exact hetzelfde serie-karakter ontvangen, als dat er eerder werd uitgezonden.

Het bovenstaande geldt echter alleen als $t_a = t_b$ en $t_c = t_d$.
De modulator moet dan even snel reageren op een overgang van 1 naar 0 (t_a) als op een overgang van 0 naar 1 (t_b) op de ingang. Tevens moet dan de demodulator even snel reageren op een overgang van 2400 Hz naar 1200 Hz (t_c) als op een overgang van 1200 Hz naar 2400 Hz (t_d) op de ingang.
In de praktijk is dit echter vaak niet het geval. Stel b.v. dat t_c groter is dan t_d.

Antw.13: wel.

Vraag 14: Een 0 op de SERIAL IN-lijn duurt dan korter/langer dan een 1 op deze lijn.

Omdat de demodulator trager van 1 naar 0 schakelt dan omgekeerd, zal in fig.10 de tijd tussen t=9 en t=10 korter zijn, dan die tussen t=10 en t=11. Algemeen kunnen we dan zeggen dat een 0 op de SERIAL IN-lijn korter duurt dan een 1.

De afwijking, die op deze wijze door de modulator en demodulator wordt veroorzaakt, ligt meestal in de grootte-orde van 1 à 2 %.

De totale afwijking, die door de cassetterecorder, modulator en demodulator wordt veroorzaakt, mag maximaal 4,5 % zijn.
Als de modulator en de demodulator hiervan 2 % veroorzaken, dan mag de maximale variatie in de bandsnelheid van de cassetterecorder 1,25 % (de helft van 2,5 %) bedragen. Aan deze eis wordt door veel "normale" cassetterecorders wel voldaan. Behalve natuurlijk tijdens het starten en stoppen van de band. Tussen twee blokken moeten we dus een ruime blokspatie (block gap) aanhouden, b.v. 3 seconden.

Uit deze paragraaf kunnen we concluderen, dat het toepassen van een normale cassetterecorder, een standaard modulator en een standaard demodulator geen bezwaar is.

8. SERIAL OUTPUT VIA EEN OUTPUT-POORT

In deze paragraaf zullen we een subroutine ontwikkelen, die een 8-bits karakter via b_3 van output-poort 21_{16} volgens serial output uitvoert. Om deze subroutine te onderscheiden van die uit paragraaf 11, noemen we deze subroutine "SOUKA". Het algemeen stroomdiagram is al in fig.5 weergegeven.

Voordat we hieruit een gedetailleerd stroomdiagram kunnen ontwikkelen, moeten er nog een aantal deelproblemen worden opgelost.

a. Waar bevindt zich het uit te voeren karakter ?
   Omdat we een blok data naar de cassetterecorder zenden, staan alle karakters in het geheugen (in de CPU is hiervoor natuurlijk geen plaats). We kiezen nu de oplossing, waarbij het adres van het op een bepaald moment uit te voeren karakter zich in registerpaar H,L bevindt. Door het hoofdprogramma moet dan steeds voor het aanroepen van "SOUKA" het juiste adres in H,L worden gezet.

 Antw.14: korter.

b. Hoe wordt het karakter in 8 afzonderlijke bits gesplitst ?
We kiezen b.v. de oplossing van fig.11.
Het karakter wordt in de accumulator steeds 1 plaats linksom geroteerd.
De 8 databits worden dus achtereenvolgens in de carry status flag opgeslagen.
Afhankelijk van de toestand van deze status flag wordt de combinatie BIT0 (b_3 = 0) resp. BIT1 (b_3 = 1) uitgevoerd.
Omdat zowel bij het roteren als bij de uitvoer de accumulator is betrokken, moeten we na het roteren de accumulatorinhoud steeds veilig stellen.
Dit doen we b.v. in register B.
Om te kunnen testen of er op deze wijze 8 databits zijn uitgevoerd, gebruiken we register C als teller.

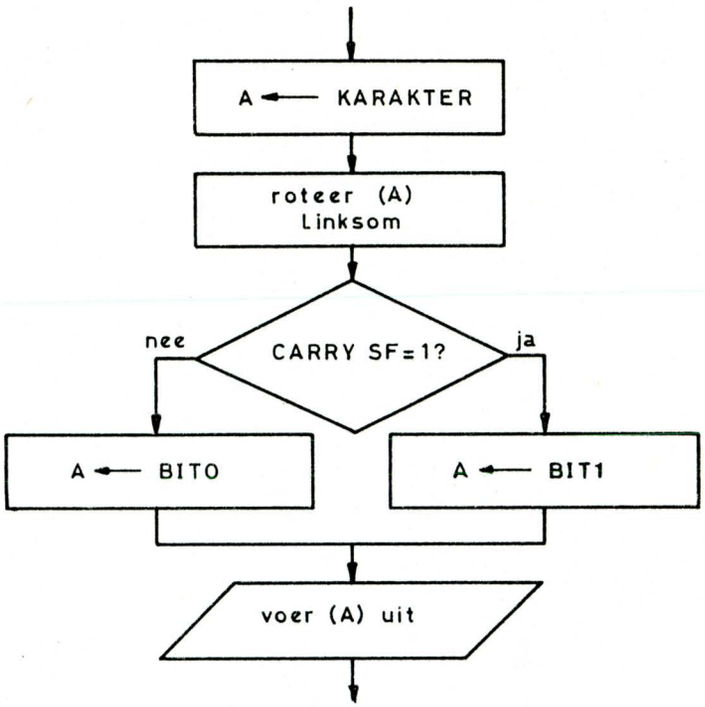


fig.11

c. Hoe realiseren we de wachttijden van 9,1 resp. 18,2 ms ?
We nemen b.v. de wachtlus van fig.12.

```
WACHT: MVI A,FFH
LUS1:  DCR A
       JNZ LUS1
```

fig.12

Vraag 15: Deze wachtlus duurt ca. 1,2/9,1/11,9 ms. (Ga uit van een klokfrequentie van 3 MHz.)

De wachtlus duurt dan maximaal 7 + 255 x (4 + 10) - 3 = 3574 states.
Bij een klokfrequentie van 3 MHz is dit 1191,3 µs = 1,19 ms.
Dit is niet genoeg. Een oplossing is b.v. de accumulator met een dusdanige waarde te vullen, dat de lus 0,91 ms duurt.

Vraag 16: Om een wachttijd van 18,2 ms te realiseren, moet de totale lus dan 1/2/20 maal worden uitgevoerd.

Door de gehele lus dan 10 of 20 maal uit te laten voeren, kunnen we wachttijden van 9,1 resp. 18,2 ms realiseren.

Vraag 17: Om de lus exact 0,91 ms te laten duren, moet de accumulator worden gevuld met .....$_{10}$.

Antw.15: 1,2. Antw.16: 20. Antw.17: 195.

0,91 ms = 910 µs = 2730 states. Om deze tijd te realiseren, moet dan gelden

$$7 + N \times (4 + 10) - 3 = 2730$$
$$14\,N = 2726$$
$$N = 194{,}71$$

De accumulator zou dan met de decimale waarde 195 moeten worden gevuld. De lus moet echter minder dan 0,91 ms duren, omdat het uitvoeren van de overige instructies ook een zekere tijd in beslag neemt.
We kiezen voor N b.v. de waarde 190_{10}.
Daar deze lus op verscheidene plaatsen in "SOUKA" (en waarschijnlijk ook in "SINK") moet worden uitgevoerd, brengen we deze onder in een subroutine "WACHT".
In register D geven we aan hoe vaak deze subroutine moet worden aangeroepen. Dit is in fig.13 weergegeven.

```
        .                       WACHT: MVI A,190D
        .                       LUS1:  DCR A
    MVI  D,....                        JNZ LUS1
    CALL WACHT                         DCR D
        .                              JNZ WACHT
        .                              RET
```

fig.13

Met deze toelichting is het algemeen stroomdiagram van fig.5 om te zetten in een gedetailleerd stroomdiagram (fig.14) voor "SOUKA".

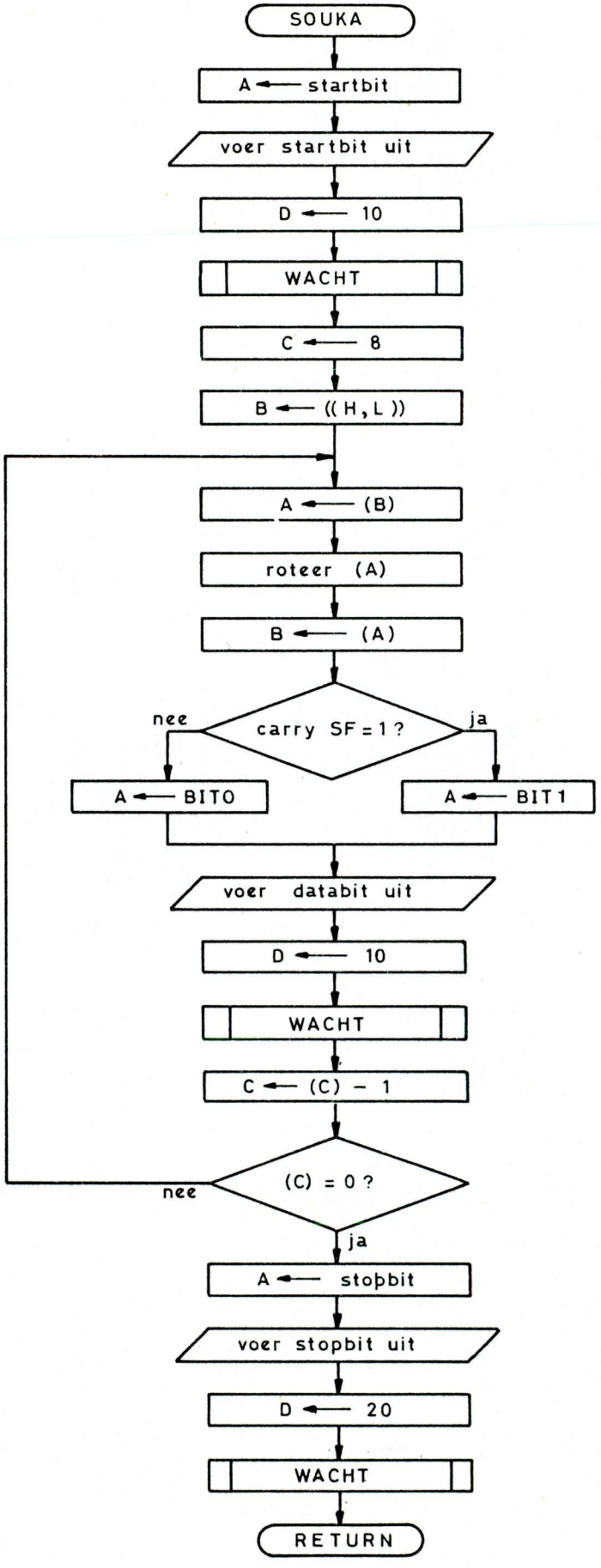

SOUKA
A ← startbit
voer startbit uit
D ← 10
WACHT
C ← 8
B ← ((H,L))
A ← (B)
roteer (A)
B ← (A)
carry SF = 1 ?
nee
ja
A ← BIT0
A ← BIT1
voer databit uit
D ← 10
WACHT
C ← (C) − 1
(C) = 0 ?
nee
ja
A ← stopbit
voer stopbit uit
D ← 20
WACHT
RETURN

fig.14

9. SERIAL INPUT VIA EEN INPUT-POORT

In deze paragraaf zullen we een subroutine ontwikkelen, die een 8-bits karakter via b_5 van input-poort 00_{16} volgens serial input invoert. Om deze subroutine van die uit paragraaf 12 te kunnen onderscheiden, noemen we deze subroutine "SINKA". Het algemeen stroomdiagram is al in fig.8 weergegeven.

Voordat we hieruit een gedetailleerd stroomdiagram kunnen ontwikkelen, moeten er nog een aantal deelproblemen worden opgelost.

a. Waar moet de ingevoerde data worden geplaatst ?
Omdat de data als een blok wordt ingevoerd, moeten alle ingevoerde karakters in het centrale geheugen worden geplaatst. We kiezen voor de oplossing, waarbij registerpaar H,L het adres van een op een bepaald moment in te voeren karakter bevat.
In het hoofdprogramma moet de inhoud van H,L dan steeds voor het aanroepen van "SINKA" worden aangepast.

b. Hoe worden de 8 in serie ontvangen databits gecombineerd tot een 8-bits parallel-karakter ?
Een mogelijke oplossing is die in fig. 15.
De laatst ontvangen databit wordt bij de voorgaande bits opgeteld. Daarna wordt het resultaat 1 plaats naar links geschoven.
Als deze behandeling 8 maal is uitgevoerd, dan is het eindresultaat een 8-bits karakter, waarvan de MSB op de meest linkse plaats staat.
De MSB is immers de eerste databit die van de cassette-recorder wordt ontvangen.

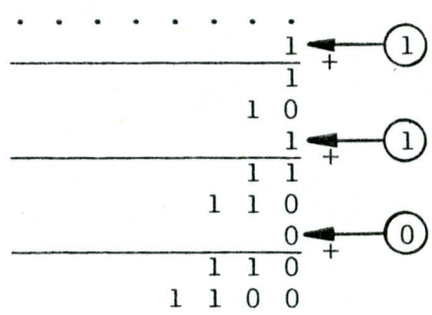


fig.15

c. Hoe realiseren we de wachttijden van 13,65 resp. 9,1 ms ?
Hiervoor gebruiken we weer de subroutine "WACHT" (0,91 ms) uit de vorige paragraaf.
In register D geven we weer aan hoe vaak de lus in deze subroutine moet worden uitgevoerd.

> Vraag 18: Om een wachttijd van 13,65 ms te realiseren, moet register D worden gevuld met ..... $_{10}$.

Met deze toelichtingen kunnen we het algemeen stroomdiagram van fig.8 omzetten in een gedetailleerd stroomdiagram (fig.16) voor "SINKA".
In dit stroomdiagram is aangegeven dat, indien er niet 2 stopbits worden gedetecteerd, er naar een programmadeel met de naam "ERROR" wordt gesprongen. Dit programmadeel moet een foutmelding geven. In dit geval springen we naar adres 0215_{16} in de monitor van de SDK 85. Hier begint nl. een reeks instructies, die de tekst "Err" op de 7-segment displays weergeven.
U moet nu echter in staat zijn, om zelf het programmadeel "ERROR" te schrijven. Dit kan zeer eenvoudig worden verwezenlijkt door een commando "write display RAM", gevolgd door de juiste 7-segment-codes, naar de 8279 te sturen.

Antw.18: 15.

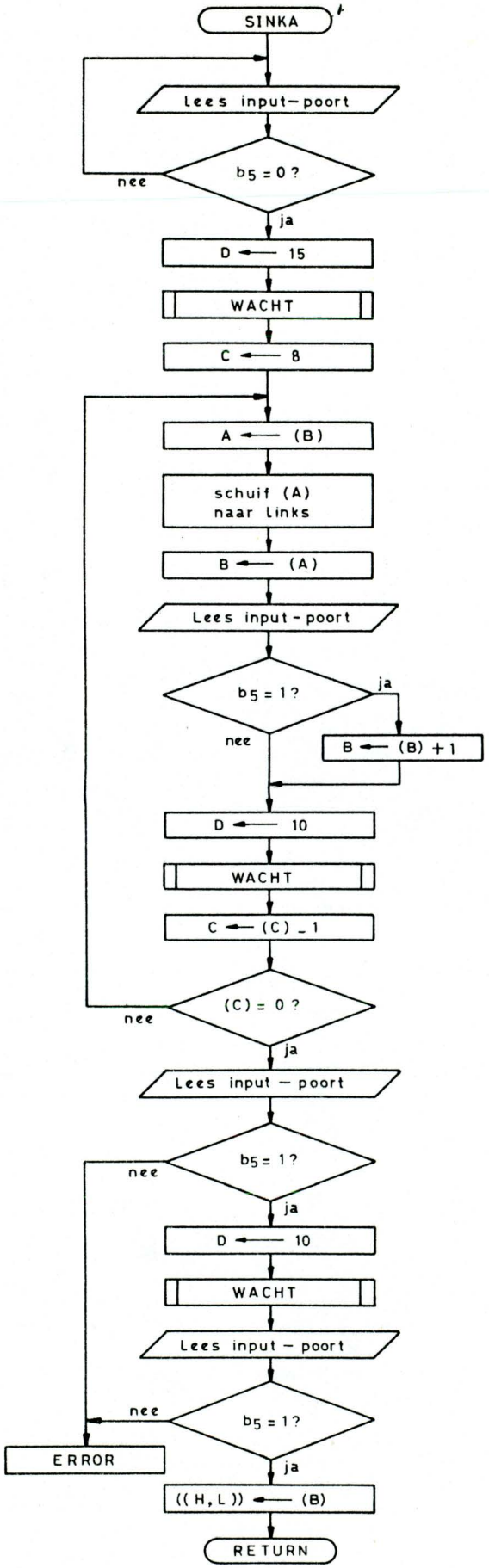

SINKA
Lees input-poort
b5 = 0 ?
nee
ja
D ← 15
WACHT
C ← 8
A ← (B)
schuif (A)
naar links
B ← (A)
Lees input - poort
b5 = 1 ?
ja
nee
B ← (B) + 1
D ← 10
WACHT
C ← (C) _ 1
(C) = 0 ?
nee
ja
Lees input – poort
b5 = 1 ?
nee
ja
D ← 10
WACHT
Lees input – poort
b5 = 1 ?
nee
ja
ERROR
((H,L)) ← (B)
RETURN

fig.16

10. PROGRAMMA'S

Vanuit de gedetailleerde stroomdiagrammen van fig.14 en fig.16 zijn nu direct de subroutines "SOUKA" en "SINKA" te schrijven.
Om deze te kunnen testen, moet er tevens een hoofdprogramma worden ontwikkeld, van waaruit de subroutines kunnen worden aangeroepen.
Het stroomdiagram voor dit programma is in fig.17 weergegeven.
Hierin wordt achtereenvolgens

a. het microcomputersysteem geïnitialiseerd.
b. een volgens serial input ontvangen blok data in het geheugen geplaatst (te beginnen op adres 2800_{16}).
c. dit blok data volgens serial output weer uitgevoerd.
d. een blokspatie van ca. 2 seconden (continu 1) gegenereerd.

In fig.18 zijn het hoofdprogramma en de subroutines "SOUKA", "SINKA" en "WACHT" weergegeven. U kunt dit programma nu op de SDK 85 testen.
Evenals in de les "Interfacing-1" is gedaan, kunt u op de uitbreidingsprint het randapparaat, in dit geval de cassetterecorder, simuleren.
Er treedt nu echter een probleem op, omdat we niet op handshake-basis met de computer communiceren. De CPU gaat uit van een baud rate van 110 bits/s. Dit is voor ons te snel om LED's af te lezen of schakelaars in een bepaalde stand te zetten.
Er zijn nu bij het testen drie mogelijkheden:

1. U verandert de subroutine "WACHT" zodanig, dat deze niet in stappen van 0,91 ms maar in stappen van ca. 10 s is te programmeren. U werkt dan met een baud rate van 1 bit per 10 s.
2. U test het programma m.b.v. "SINGLE STEP". Denk er dan aan, dat u niet in de subroutine "WACHT" terecht komt. Het zou u nl. uren kosten om deze eenmaal in single step uit te voeren. U kunt b.v. elke instructie CALL WACHT vervangen door 3 NOP-instructies.
3. U kunt breakpoints aanbrengen. Deze voorziening is in het programma van fig.18 aangebracht, nl. een RST1-instructie aan het eind van subroutine "WACHT". Zie voor de beschrijving van het werken met breakpoints de les "Systeemeigenschappen software".

Vraag 19: De grootte van het blok wordt bepaald door de waarde met de symbolische naam ".....".

Het aantal karakters van het ingevoerde en uit te voeren blok wordt in fig.18 met de symbolische naam "SIZE" aangeduid.
Door de EQU-directive heeft "SIZE" de waarde 128_{10} gekregen.
Als u dit te veel tijd kost tijdens het testen, dan kunt u deze waarde verkleinen (adressen 2015_{16} en 2022_{16}).

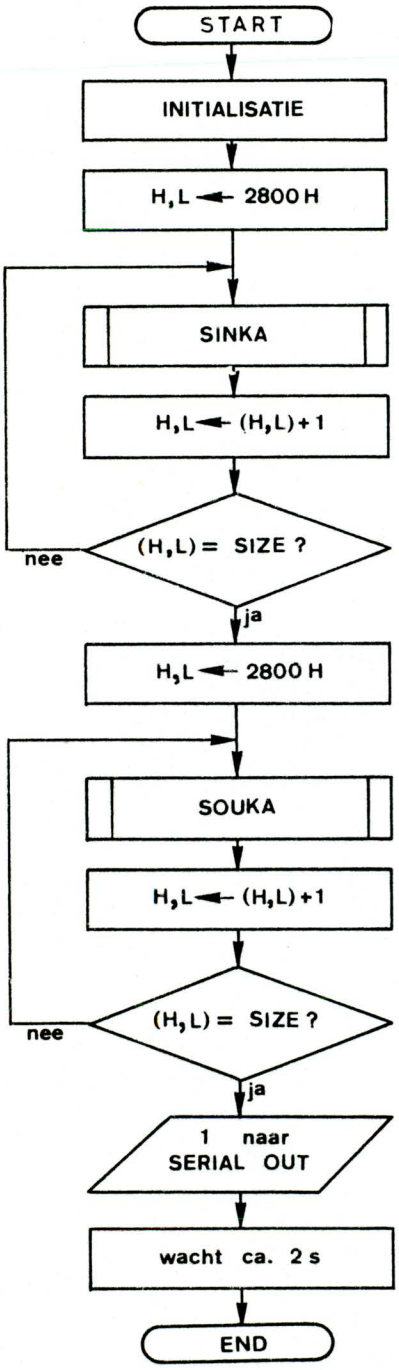


fig.17

```
                        ORG   2000H
                START   EQU   00H        ;STARTBIT=0
                STOP    EQU   08H        ;STOPBIT=1
                BIT0    EQU   00H        ;DATABIT=0
                BIT1    EQU   08H        ;DATABIT=1
                SEROU   EQU   21H        ;SERIAL OUTPUT
                SERIN   EQU   00H        ;SERIAL INPUT
                MSKIN   EQU   20H        ;UNMASK SERIAL IN
                ERROR   EQU   0215H      ;MONITOR SDK 85
                SIZE    EQU   128D       ;BLOK=128 KARAKTERS
2000  3E 03             MVI   A,03H
2002  D3 20             OUT   20H        ;POORT 21H=OUTPUT
2004  32 FF 20          STA   20FFH
2007  AF                XRA   A
2008  D3 02             OUT   02H        ;POORT 00H=INPUT
200A  31 C2 20          LXI   SP,20C2H
200D  21 00 28          LXI   H,2800H    ;BEGINADRES BLOK
2010  CD 62 20    KARIN: CALL  SINKA      ;SERIE-KARAKTER IN
2013  23                INX   H
2014  3E 80             MVI   A,SIZE     ;LAATSTE
2016  BD                CMP   L          ;  KARAKTER?
2017  C2 10 20          JNZ   KARIN      ;NEE: SPRING TERUG
201A  21 00 28          LXI   H,2800H    ;BEGINADRES BLOK
201D  CD 37 20    KAROU: CALL  SOUKA     ;SERIE-KARAKTER UIT
2020  23                INX   H
2021  3E 80             MVI   A,SIZE     ;LAATSTE
2023  BD                CMP   L          ;  KARAKTER?
2024  C2 1D 20          JNZ   KAROU      ;NEE: SPRING TERUG
2027  06 03             MVI   B,03H      ;  **********************
2029  11 FF FF  GAP1:   LXI   D,FFFFH    ;  *
202C  1B        GAP2:   DCX   D          ;  * DEZE INSTRUCTIES
202D  7A                MOV   A,D        ;  * GENEREREN EEN
202E  B3                ORA   E          ;  * BLOKSPATIE VAN
202F  C2 2C 20          JNZ   GAP2       ;  * CA 2 SECONDEN
2032  05                DCR   B          ;  *
2033  C2 29 20          JNZ   GAP1       ;  **********************
2036  76                HLT
2037  3E 00     SOUKA: MVI   A,START    ;STARTBIT NAAR
2039  D3 21            OUT   SEROU.     ;  SERIAL OUT
203B  16 0A             MVI   D,10D      ;WACHT
203D  CD 99 20          CALL  WACHT      ;  9,1 MS
2040  0E 08             MVI   C,8D       ;TELLER DATABITS
2042  46                MOV   B,M        ;HAAL KARAKTER OP
2043  78        DATOU: MOV   A,B        ;TEST
2044  07                RLC              ; VOLGENDE
2045  47                MOV   B,A        ;  DATABIT
2046  3E 00             MVI   A,BIT0
2048  D2 4E 20          JNC   UIT        ;DATABIT
204B  3E 08             MVI   A,BIT1     ; NAAR
204D  D3 21       UIT:  OUT   SEROU      ;  SERIAL OUT
204F  16 0A             MVI   D,10D      ;WACHT
2051  CD 99 20          CALL  WACHT      ;  9,1 MS
2054  0D                DCR   C          ;8 DATABITS?
2055  C2 43 20          JNZ   DATOU      ;NEE: VOLGENDE BIT
2058  3E 08             MVI   A,STOP     ;STOPBIT NAAR
205A  D3 21             OUT   SEROU      ;  SERIAL OUT
205C  16 14             MVI   D,20D      ;WACHT
205E  CD 99 20          CALL  WACHT      ;  18,2 MS
2061  C9                RET
2062  DB 00     SINKA: IN    SERIN      ;LEES SERIAL IN
2064  E6 20            ANI   MSKIN      ;TEST B5
2066  C2 62 20         JNZ   SINKA      ;WACHT OP STARTBIT
2069  16 0F            MVI   D,15D      ;WACHT
206B  CD 99 20         CALL  WACHT      ;  13,65 MS
206E  0E 08            MVI   C,8D       ;TELLER DATABITS
2070  78        DATIN: MOV   A,B        ;SCHUIF KARAKTER
2071  87               ADD   A          ; 1 PLAATS
2072  47               MOV   B,A        ;  NAAR LINKS
2073  DB 00            IN    SERIN      ;LEES SERIAL IN
2075  E6 20            ANI   MSKIN      ;TEST B5
2077  CA 7B 20         JZ    WAIT       ;DATABIT=0
207A  04               INR   B          ;DATABIT=1
207B  16 0A       WAIT: MVI   D,10D      ;WACHT
207D  CD 99 20          CALL  WACHT      ;  9,1 MS
2080  0D                DCR   C          ;8 DATABITS?
2081  C2 70 20          JNZ   DATIN      ;NEE: VOLGENDE BIT
2084  DB 00             IN    SERIN      ;LEES SERIAL IN
2086  E6 20             ANI   MSKIN      ;TEST STOPBIT
2088  CA 15 02          JZ    ERROR      ;SPRING ALS STOPBIT=0
208B  16 0A             MVI   D,10D      ;WACHT
208D  CD 99 20          CALL  WACHT      ;  9,1 MS
2090  DB 00             IN    SERIN      ;LEES SERIAL IN
2092  E6 20             ANI   MSKIN      ;TEST STOPBIT
2094  CA 15 02          JZ    ERROR      ;SPRING ALS STOPBIT=0
2097  70                MOV   M,B        ;BERG KARAKTER OP
2098  C9                RET
2099  3E BE     WACHT: MVI   A,190D     ;  **********************
209B  3D          LUS1:  DCR   A          ;  * DEZE SUBROUTINE
209C  C2 9B 20           JNZ   LUS1       ;  * GENEREERT EEN
209F  15                 DCR   D          ;  * WACHTTIJD VAN
20A0  C2 99 20           JNZ   WACHT      ;  * (D) x 0,91 MS
20A3  CF                 RST   1          ;BREAKPOINT
20A4  C9                 RET              ;  **********************
                         END
```

fig.18

11. SERIAL OUTPUT VIA DE SOD-LIJN

In deze paragraaf gaan we bekijken welke wijzigingen in "SOUKA" moeten worden aangebracht, om een serie-karakter via de SOD-lijn van de 8085 uit te voeren. De nieuw te ontwikkelen subroutine noemen we "SOUKB". We kunnen weer uitgaan van het algemeen stroomdiagram van fig.5. Het enige verschil tussen "SOUKA" en "SOUKB" is, dat nu niet b_3 van de accumulator d.m.v. een output-instructie, maar b_7 d.m.v. een SIM-instructie wordt uitgevoerd.

We gebruiken weer register B, om de inhoud van de accumulator veilig te stellen, tijdens het uitvoeren van de overige instructies in de subroutine. Om een databit uit te voeren, moet de inhoud van register B naar de accumulator worden overgebracht. D.m.v. een SIM-instructie wordt een databit (b_7) op de SOD-lijn geplaatst. Dan moet de inhoud van de accumulator één plaats naar links worden geschoven of geroteerd. De volgende databit moet immers nu in b_7 worden geplaatst. Het uitvoeren van de 8 databits zou dan verlopen volgens de in fig.19 weergegeven methode. De blokken ① t/m ④ kunnen dan worden vervangen door de instructies van fig.20.

```
MOV   A,B
SIM
RLC
MOV   B,A
```

fig.20

Vraag 20: De instructies van fig.20 hebben wel/niet het gewenste resultaat.

Bij het uitvoeren van een SIM-instructie moet in de accumulator gelden:
b_6 = 1, om de SOD-lijn te beïnvloeden (zie paragraaf 5c van de les "Systeemeigenschappen hardware").
b_3 = 0, om de interrupt masks niet te veranderen (zie paragraaf 9a van de les "Interrupt").
Fig.20 moet dan worden gewijzigd in fig.21. Hierin dient de instructie ORI ENSOD om b_6 1 te maken en ANI DISIM om b_3 0 te maken. De extra instructie MOV A,B is nodig, omdat ORI en ANI de accumulatorinhoud wijzigen.

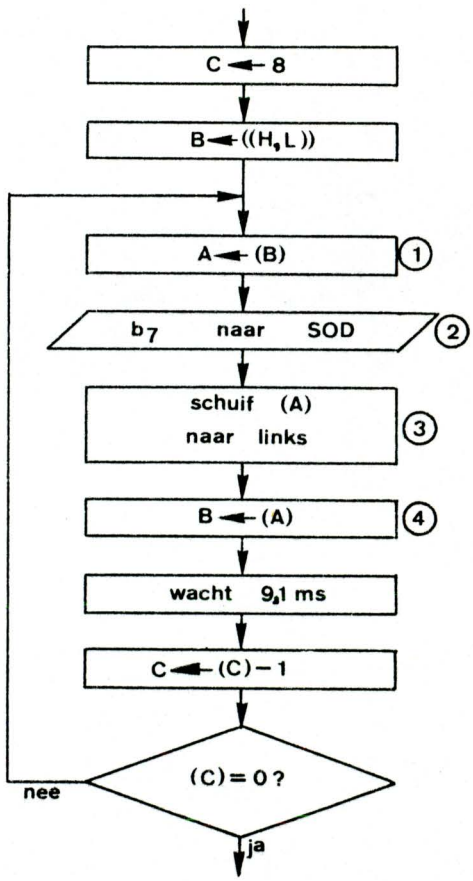


fig.19

Antw.20: niet.

```
                        ORG   2000H
                START   EQU   40H       ;STARTBIT=0
                STOP    EQU   COH       ;STOPBIT=1
                MSKIN   EQU   80H       ;UNMASK SID
                ENSOD   EQU   40H       ;ENABLE SOD
                DISIM   EQU   COH       ;DISABLE INT.MASKS
                ERROR   EQU   0215H     ;MONITOR SDK 85
                SIZE    EQU   128D      ;BLOK=128 KARAKTERS
2000  31 C2 20          LXI   SP,20C2H
2003  21 00 28          LXI   H,2800H   ;BEGINADRES BLOK
2006  CD 53 20  KARIN:  CALL  SINKB     ;SERIE-KARAKTER IN
2009  23                INX   H
200A  3E 80             MVI   A,SIZE    ;LAATSTE
200C  BD                CMP   L         ;  KARAKTER?
200D  C2 06 20          JNZ   KARIN     ;NEE: SPRING TERUG
2010  21 00 28          LXI   H,2800H   ;BEGINADRES BLOK
2013  CD 2D 20  KAROU:  CALL  SOUKB     ;SERIE-KARAKTER UIT
2016  23                INX   H
2017  3E 80             MVI   A,SIZE    ;LAATSTE
2019  BD                CMP   L         ;  KARAKTER?
201A  C2 13 20          JNZ   KAROU     ;NEE: SPRING TERUG
201D  06 03             MVI   B,03H     ; **********************
201F  11 FF FF  GAP1:   LXI   D,FFFFH   ; *
2022  1B        GAP2:  DCX   D         ; * DEZE INSTRUCTIES
2023  7A               MOV   A,D       ; * GENEREREN EEN
2024  B3               ORA   E         ; * BLOKSPATIE VAN
2025  C2 22 20         JNZ   GAP2      ; * CA 2 SECONDEN
2028  05               DCR   B         ; *
2029  C2 1F 20         JNZ   GAP1      ; **********************
202C  76               HLT
202D  3E 40     SOUKB: MVI   A,START   ;STARTBIT
202F  30               SIM             ;  NAAR SOD
2030  16 0A             MVI   D,10D     ;WACHT
2032  CD 86 20          CALL  WACHT     ;  9,1 MS
2035  0E 08             MVI   C,8D      ;TELLER DATABITS
2037  46               MOV   B,M       ;HAAL KARAKTER OP
2038  78      DATOU: MOV   A,B
2039  F6 40          ORI   ENSOD     ;DATABIT
203B  E6 C0          ANI   DISIM     ;  NAAR SOD
203D  30             SIM
203E  78             MOV   A,B       ;SCHUIF KARAKTER
203F  07             RLC             ; 1 PLAATS
2040  47             MOV   B,A       ;  NAAR LINKS
2041  16 0A          MVI   D,10D     ;WACHT
2043  CD 86 20       CALL  WACHT     ;  9,1 MS
2046  0D             DCR   C         ;8 DATABITS?
2047  C2 38 20       JNZ   DATOU     ;NEE: VOLGENDE BIT
204A  3E C0          MVI   A,STOP    ;STOPBIT
204C  30             SIM             ;  NAAR SOD
204D  16 14          MVI   D,20D     ;WACHT
204F  CD 86 20       CALL  WACHT     ;  18,2 MS
2052  C9            RET
2053  20      SINKB: RIM             ;LEES SID
2054  E6 80          ANI   MSKIN     ;TEST B7
2056  C2 53 20       JNZ   SINKB     ;WACHT OP STARTBIT
2059  16 0F          MVI   D,15D     ;WACHT
205B  CD 86 20       CALL  WACHT     ;  13,65 MS
205E  0E 08          MVI   C,8D      ;TELLER DATABITS
2060  78      DATIN: MOV   A,B       ;SCHUIF KARAKTER
2061  87             ADD   A         ; 1 PLAATS
2062  47             MOV   B,A       ;  NAAR LINKS
2063  20             RIM             ;TEST
2064  E6 80          ANI   MSKIN     ;  DATABIT
2066  CA 6A 20       JZ    WAIT      ;DATABIT=0
2069  04             INR   B         ;DATABIT=1
206A  16 0A     WAIT:  MVI   D,10D     ;WACHT
206C  CD 86 20       CALL  WACHT     ;  9,1 MS
206F  0D             DCR   C         ;8 DATABITS?
2070  C2 60 20       JNZ   DATIN     ;NEE: VOLGENDE BIT
2073  20             RIM             ;TEST EERSTE
2074  E6 80          ANI   MSKIN     ;  STOPBIT
2076  CA 15 02       JZ    ERROR     ;SPRING ALS STOPBIT=0
2079  16 0A          MVI   D,10D     ;WACHT
207B  CD 86 20       CALL  WACHT     ;  9,1 MS
207E  20             RIM             ;TEST TWEEDE
207F  E6 80          ANI   MSKIN     ;  STOPBIT
2081  CA 15 02       JZ    ERROR     ;SPRING ALS STOPBIT=0
2084  70             MOV   M,B       ;BERG KARAKTER OP
2085  C9             RET
2086  3E BE     WACHT: MVI   A,190D    ; ******************
2088  3D        LUS1:  DCR   A         ; * DEZE SUBROUTINE
2089  C2 88 20       JNZ   LUS1      ; * GENEREERT EEN
208C  15             DCR   D         ; * WACHTTIJD VAN
208D  C2 86 20       JNZ   WACHT     ; * (D) x 0,91 MS
2090  CF             RST   1         ;BREAKPOINT
2092  C9             RET             ; ******************
                     END
```

fig.22

```
MOV   A,B
ORI   ENSOD   ; B6 = 1
ANI   DISIM   ; B3 = 0
SIM
MOV   A,B
RLC
MOV   B,A
```

fig.21

Het totale programma is samen met "SINKB" in fig.22 weergegeven.

12. SERIAL INPUT VIA DE SID-LIJN

In deze paragraaf gaan we bekijken welke wijzigingen in "SINKA" moeten worden aangebracht, om een serie-karakter via de SID-lijn van de 8085 in te voeren. De nieuw te ontwikkelen subroutine noemen we "SINKB". Het enige verschil tussen "SINKA" en "SINKB" is, dat een bit niet in b_5 maar in b_7 van de accumulator wordt ontvangen. De wijzigingen, die we in "SINKA" moeten aanbrengen, zijn:

a. De input-instructies worden RIM-instructies.
b. De waarde voor MSKIN moet zodanig worden gewijzigd, dat b_7 wordt getest.

Het programma is dan dat in fig.22.

13. TESTEN VAN HET PROGRAMMA

In fig.22 zijn een hoofdprogramma en de subroutines "SOUKB", "SINKB" en "WACHT" weergegeven. Het programma lijkt bijzonder veel op dat van fig.18. Er zijn echter enkele kleine verschillen.

a. De waarde, die in de accumulator moet worden geplaatst om een startbit of een stopbit uit te kunnen voeren zijn gewijzigd.
b. De input- en output-instructies zijn vervangen door RIM- resp. SIM-instructies.
c. De waarde van "MSKIN" is gewijzigd, om b_7 van de accumulator te kunnen testen.

In de subroutine "WACHT" is weer een breakpoint opgenomen, om het programma vertraagd te kunnen testen. Voordat u dit programma in het RAM-geheugen van de SDK 85 kunt plaatsen en testen, zult u echter enkele veranderingen in de hardware van het systeem moeten aanbrengen. In de SDK 85 zijn de SID- en SOD-lijnen nl. vast verbonden met 0 (massa) resp. de TTY-interface-schakeling. Dit is blokschematisch weergegeven in fig.23.

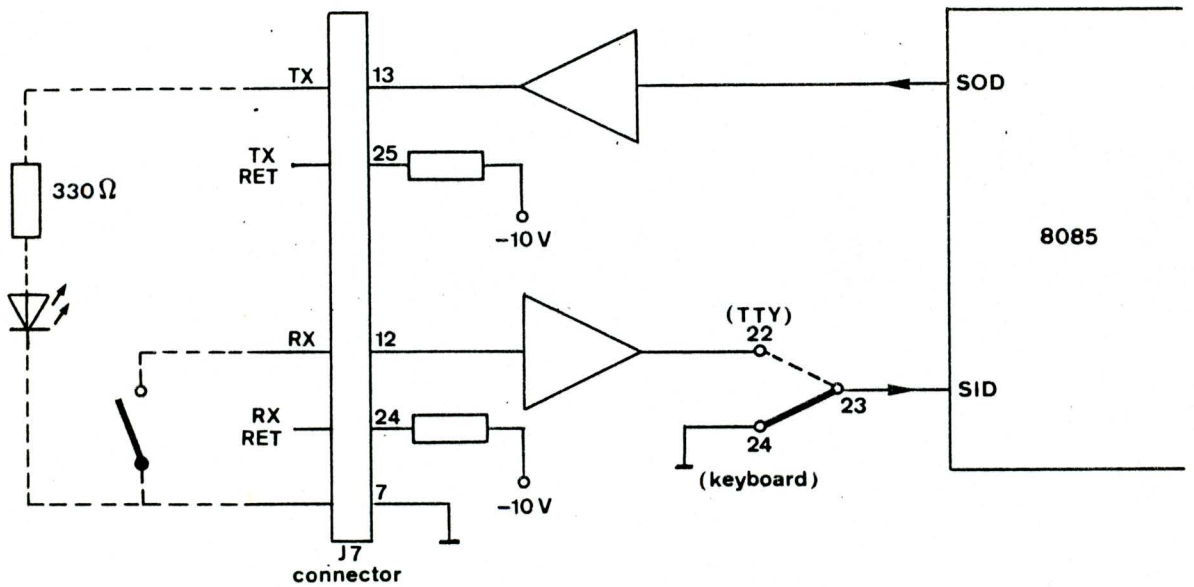


fig.23

De SID-lijn is met 0 verbonden. Dit is noodzakelijk om de bootstrap in de monitor te laten bepalen, dat er geen teletype is aangesloten (zie paragraaf 2 van de les "Systeemeigenschappen software").

De TTY-interface-schakeling van de SDK 85 bestaat uit 2 (inverterende) buffers, in de zendlijn (TX) en de ontvanglijn (RX). Bovendien zijn nog twee retourlijnen aanwezig, die via weerstanden met -10V zijn verbonden. Om de benodigde lijnstromen voor een TTY te verkrijgen, is bij gebruik van een TTY n.l. een extra negatieve voedingsspanning noodzakelijk.

In fig.23 is tevens weergegeven welke wijzigingen moeten worden aangebracht (gestippeld deel) om het programma van fig.22 te kunnen testen. Als u uw SDK 85 wilt aanpassen om de cassetterecorder te simuleren, dan dient u onderstaande punten zorgvuldig door te lezen en eventuele opdrachten uit te voeren.

a. De TTY-interface-schakeling bevindt zich rechtsboven op de printplaat van de SDK 85.
b. Hier bevindt zich tevens een aantal genummerde aansluitpunten. (connector J7). Voor u zijn nu van belang:
   punt 7: massa
   punt 12: RX
   punt 13: TX
   punt 24: RX RET
   punt 25: TX RET
c. De retourlijnen (punten 24 en 25) gebruiken we in dit geval niet. We hebben dan ook geen -10V voedingsspanning nodig.
d. Soldeer tussen RX en massa een schakelaar (enkelpolig maakcontact).
e. Soldeer tussen TX en massa een LED met een voorschakelweerstand van 330Ω.
f. Rechts naast de 8279 op de printplaat van de SDK 85 bevinden zich de aansluitpunten 22, 23 en 24.
g. Verwijder de draadverbinding tussen de punten 23 en 24.
h. Verbind nu de punten 22 en 23 met elkaar.

Als u de opdrachten uit de punten d, e, g en h hebt uigevoerd, dan heeft de nieuw ontstane hardware de volgende eigenschappen:

SOD = 0 : LED brandt
SOD = 1 : LED is uit
Schakelaar gesloten: SID = 1
Schakelaar geopend : SID = 0

Deze eigenschappen zijn een gevolg van het feit, dat de buffers in de TTY-interface inverterend zijn.

U kunt nu het programma van fig.22 in het RAM-geheugen plaatsen en daarna testen.

Vraag 21: Tijdens het invoeren van het programma moet de schakelaar in de RX-lijn geopend/gesloten zijn.

Denk er aan, dat wanneer de monitor werkzaam is, SID = 0 moet zijn. De schakelaar in de RX-lijn moet dus geopend zijn. Pas als u het programma hebt gestart met "GO"; 2; 0; 0; 0;"EXEC" mag u de schakelaar gebruiken om de toestand op de SID-lijn te veranderen.

 Antw.21: geopend.

14. CONCLUSIES

Uit deze les kunnen we de volgende conclusies trekken:

a. Het accent bij de ontwikkeling van interface tussen microcomputer en cassetterecorder valt voornamelijk op de software. Dit verschijnsel treedt eigenlijk altijd op bij de interfacing van randapparaten. Voor de meeste interface-problemen is de benodigde hardware als integrated circuit te kopen, b.v. in de vorm van een programmeerbare chip. De noodzakelijke software dient u in de meeste gevallen zelf te schrijven.

b. Het programma van fig.22 is korter dan het programma van fig.18. Een van de redenen hiervoor is het feit, dat in fig.22 de initialisatie van de I/O-module is weggelaten. Om een duidelijk beeld te krijgen van de besparing aan geheugenruimte bij het gebruik van de SID- en SOD-lijnen van de CPU, moeten we de subroutines met elkaar vergelijken.

Vraag 22: "SOUKA" beslaat .... bytes.
"SOUKB" beslaat .... bytes.
"SINKA" beslaat .... bytes.
"SINKB" beslaat .... bytes.

Uit het tellen van de aantallen bytes van de subroutines blijkt dat "SOUKB" 5 bytes korter is dan "SOUKA". "SINKB" is 4 bytes korter dan "SINKA". Er blijkt dus dat het gebruik van de SID- en SOD-lijnen van de 8085 voor serial I/O niet tot belangrijke besparing van geheugenruimte leidt.

De SID- en SOD-lijnen bieden wel een voordeel in 8085-systemen, waarin er slechts één randapparaat met serial I/O werkt. We behoeven dan n.l. niet 2 lijnen van de 8-bits parallel-poorten uit de I/O-module te reserveren, waardoor een "onvolledige" parallel-poort zou overblijven. Dit voordeel vervalt natuurlijk als er meer randapparaten volgens serial I/O zijn aangesloten.

c. Door de lage baud rate en de hiermee samenhangende lange wachttijden kost serial I/O op basis van geprogrammeerde I/O vrij veel tijd. In deze les was dit geen bezwaar omdat we het hoofdprogramma alleen gebruikten om de subroutines aan te roepen. In de praktijk zal behalve de I/O ook een aantal bewerkingen uitgevoerd moeten worden. Dan is voor deze lage baud rate een interface met serial I/O op interrupt-basis een meer efficiënte oplossing.

VERKEERSLICHTENREGELING.

Verkeerslichtenregeling

1. INLEIDING

In deze les gaan we een verkeersafhankelijke verkeerslichtenregeling ontwikkelen. De term verkeersafhankelijk wil zeggen, dat de verkeerslichten worden bediend, terwijl steeds rekening wordt gehouden met het verkeersaanbod.
De verkeerslichtenregeling wordt gerealiseerd m.b.v. een microcomputer. Dit houdt in, dat het aanwezig zijn van voertuigen in de vorm van nullen en énen aan de computer moet worden aangeboden en dat de door de computer afgegeven énen en nullen moeten worden omgezet in brandende rode, groene en gele lampen. Er is dus een interface tussen de computer en het kruispunt noodzakelijk.

Evenals in voorgaande lessen, gaan we de probleemanalyse en het ontwikkelen van een algemeen stroomdiagram zodanig uitvoeren, dat de gevonden oplossingsmethode voor ieder type microcomputer geldt.

Bij het ontwikkelen van het gedetailleerd stroomdiagram moeten we rekening houden met de eigenschappen van de te gebruiken microcomputer. In deze les gaan we weer uit van de SDK 85.

Als u gebruik wilt maken van een andere computer, dan moet u in de laatste fase van het ontwikkelingsproces de eigenschappen van die computer verwerken. U volgt dan wel de in deze les beschreven methode, maar de machinegerichte details zult u aan de door u gekozen microcomputer moeten aanpassen.

Deze les is te beschouwen als een vervolg op de les "Verkeerlichtenregeling" uit de cursus "Microprocessors/Microcomputers".
We hebben nu echter te maken met meer verkeersstromen. Er zullen dus ook meer detectoren en verkeerslichten een rol spelen.

2. PROBLEEMOMSCHRIJVING

a. Hardware-beschrijving

Gevraagd wordt om m.b.v. detectoren en verkeerslichten het verkeer op de T-kruising van fig.1 te regelen.

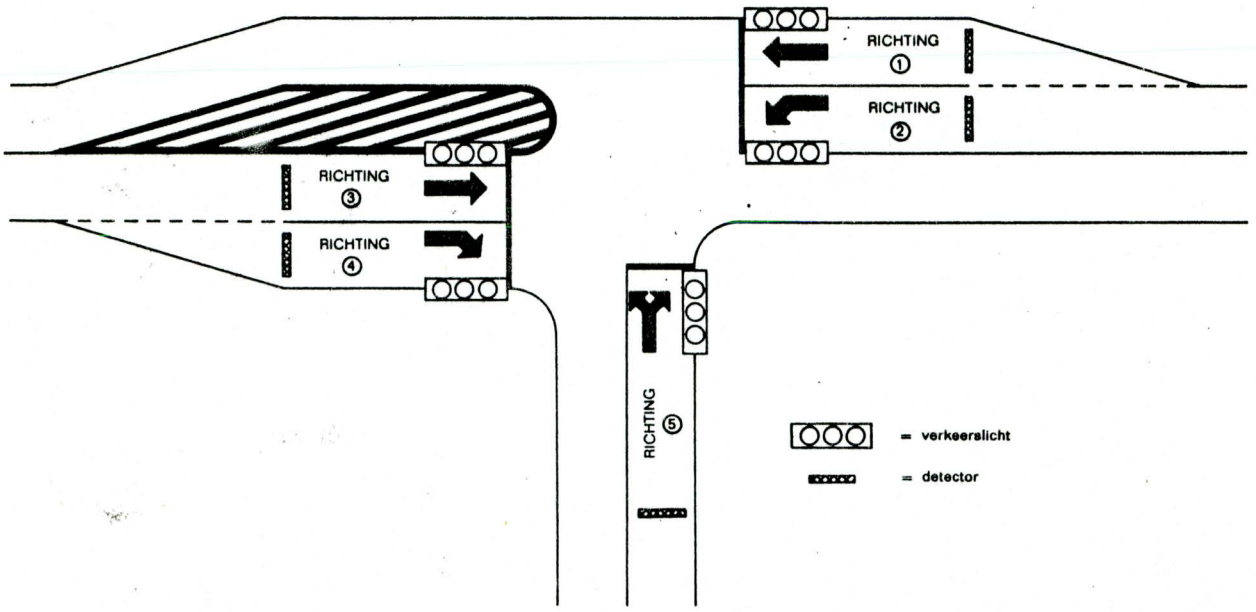


fig.1

De interface-schakelingen tussen microcomputer, detectoren en verkeerslichten zijn in fig.2 gegeven.

De 5 detectoren zijn in het wegdek aangebracht. Elke detector geeft een 0 af als er een voertuig passeert. De detector-flipflops (DET FF) zijn elk opgebouwd uit 2 NEN-poorten (fig.3).

Een flipflop wordt dan geset door een 0 op de S-ingang. Dus als er een voertuig door de detector wordt gedetecteerd, wordt de bijbehorende detector-flipflop geset (Q = 1). Een detector-flipflop wordt gereset (Q = 0), als de microcomputer een 0 op de desbetreffende R-ingang plaatst. (Deze 0 op de R-ingang mag natuurlijk maar kortstondig aanwezig zijn, anders kan de detector-flipflop niet meer worden geset.)

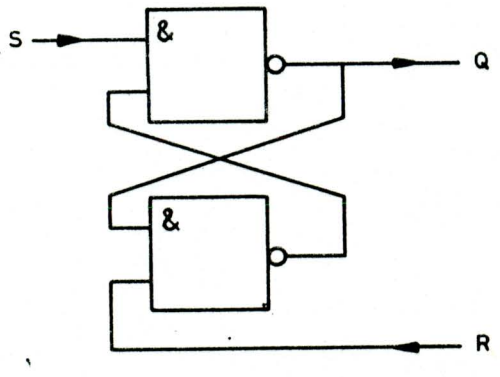


fig.3

Elk van de 15 buffers is zo opgebouwd, dat een door de microcomputer afgegeven 1 de bijbehorende lamp doet branden.

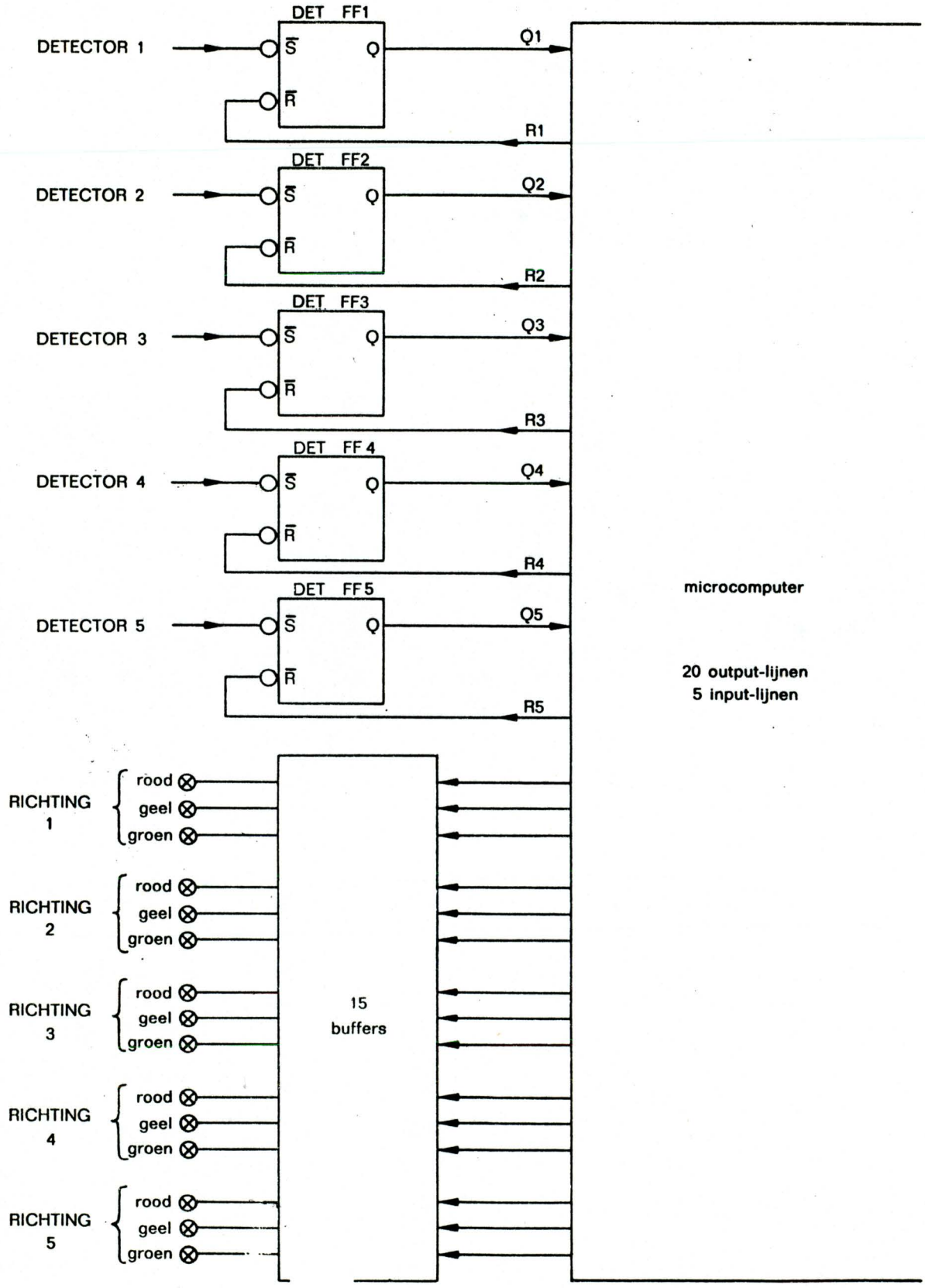

DET FF1
DETECTOR 1
S̄
R̄
Q
Q1
R1
DET FF2
DETECTOR 2
Q2
R2
DET FF3
DETECTOR 3
Q3
R3
DET FF4
DETECTOR 4
Q4
R4
DET FF5
DETECTOR 5
Q5
R5
microcomputer
20 output-lijnen
5 input-lijnen
RICHTING 1
RICHTING 2
RICHTING 3
RICHTING 4
RICHTING 5
rood
geel
groen
15
buffers

fig.2

b. Verkeerstechnische eisen

Verkeerstellingen hebben uitgewezen, dat het verkeer in de richtingen 1 en 3 het drukst is. Dit is dan ook de hoofdverkeersstroom. Het verkeer van de zijweg (richting 5) is het minst druk. Het aantal voertuigen, dat de richtingen 2 en 4 neemt, zit hier tussen in.

We hebben dus te maken met 5 verkeersstromen (richtingen 1 t/m 5). In tabel 1 is voor elke richting aangegeven welke voorwaarden voor de overige richtingen gelden, als de betreffende richting groen licht heeft. Deze voorwaarden zijn zo opgesteld, dat de op een bepaald moment in beweging zijnde verkeersstromen elkaar niet kruisen.

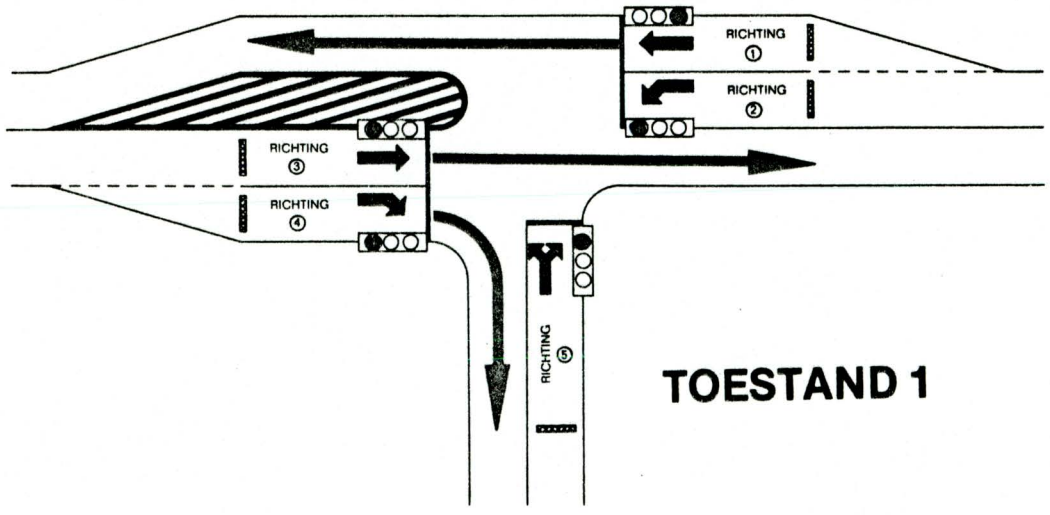
groen voor	dan moet gelden rood voor de richtingen	dan mag gelden groen voor de richtingen
richting 1	2 en 5	3 en 4
richting 2	3, 4 en 5	1
richting 3	2 en 5	1 en 4
richting 4	2 en 5	1 en 3
richting 5	1, 2 en 3	4

Tabel 1

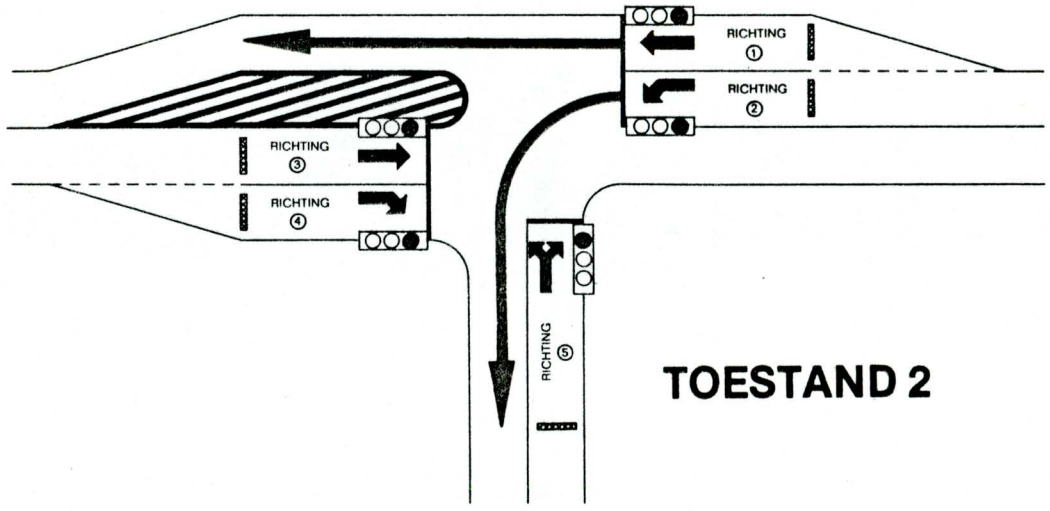
Uit tabel 1 blijkt, dat er in feite maar 3 combinaties van brandende rode en groene verkeerslichten overblijven.
Deze drie toestanden zijn in tabel 2 en fig.4 weergegeven.

toestand	groen voor de richtingen	rood voor de richtingen
1	1, 3 en 4	2 en 5
2	1 en 2	3, 4 en 5
3	4 en 5	1, 2 en 3

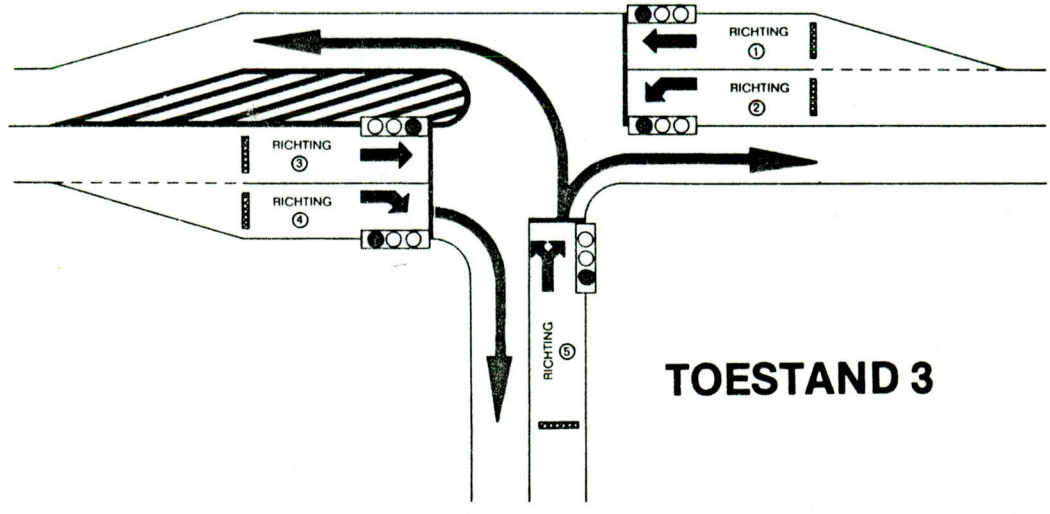
Tabel 2



TOESTAND 1



TOESTAND 2



TOESTAND 3

fig.4

Elektronica opleidingen Dirksen

Gevraagd wordt een programma te schrijven, waardoor de microcomputer de verkeerslichten op de volgende wijze aanstuurt.

a. Toestand 1 is de voorkeurstoestand. Immers hierin heeft de hoofdverkeersstroom (richting 1 en 3) groen licht. Deze toestand moet gelden als het programma is gestart, als er in de richtingen 1, 3 en 4 verkeer wordt gedetecteerd of als er helemaal geen voertuigen worden gedetecteerd.
b. Toestand 1 moet minimaal 10 seconden duren. Blijkt, dat tijdens deze 10 seconden nog voertuigen voor de richtingen 1, 3 of 4 zijn gedetecteerd, dan moet toestand 1 nog eens 20 seconden extra gehandhaafd blijven.
c. Nadat toestand 1 10 resp. 30 seconden aanwezig is geweest, moet worden onderzocht of er condities aanwezig zijn om naar toestand 2 ($Q_2 = 1$) of naar toestand 3 ($Q_5 = 1$) over te gaan. Zijn deze condities niet aanwezig, dan moet toestand 1 blijven bestaan, totdat ze wel aanwezig zijn.
d. Toestand 2 wordt bereikt na een detectie voor richting 2.
Deze toestand duurt 15 seconden. Na deze 15 seconden moet naar toestand 1 worden teruggekeerd.
e. Toestand 3 wordt bereikt na een detectie voor richting 5 en duurt 10 seconden.
Na deze 10 seconden moet naar toestand 1 worden teruggekeerd.
f. Bij het overgaan van de ene toestand in de andere mogen de verkeerslichten niet direct van groen op rood springen. Er moet steeds een bepaalde "geeltijd" worden ingelast. Deze geeltijd moet steeds 5 seconden duren.

PROBEER NU ZELF EEN PROBLEEMANALYSE UIT TE VOEREN EN EEN ALGEMEEN STROOMDIAGRAM OP TE STELLEN.
BESTUDEER DAARNA ONZE OPLOSSING IN DE VOLGENDE PARAGRAFEN.

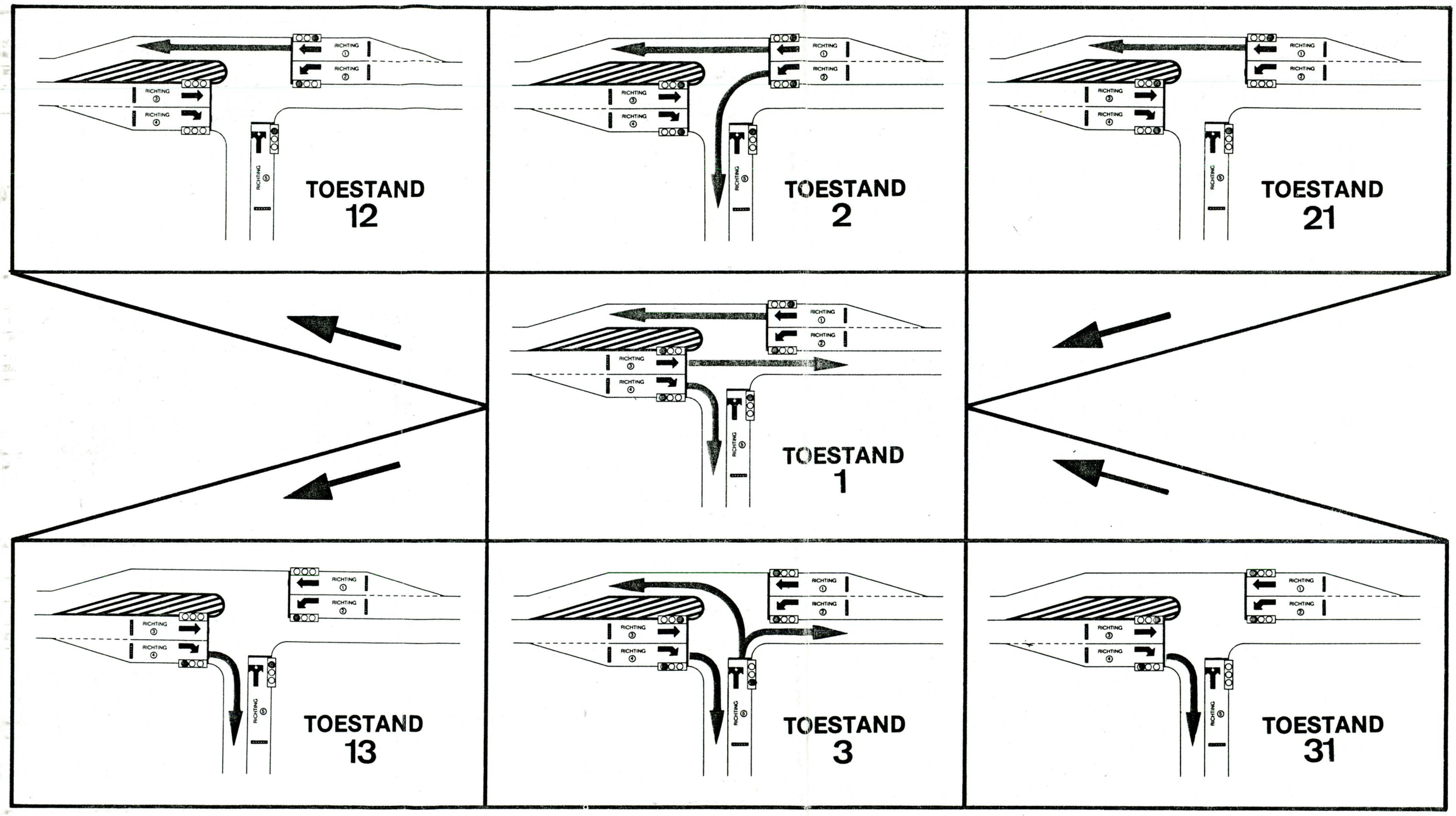

RICHTING 1
RICHTING 2
RICHTING 3
RICHTING 4
RICHTING 5
TOESTAND 12
TOESTAND 2
TOESTAND 21
TOESTAND 1
TOESTAND 13
TOESTAND 3
TOESTAND 31

fig.7

3. PROBLEEMANALYSE

Evenals in de les "Verkeerslichtenregeling" van de cursus "Microprocessors/Microcomputers" is gedaan, tekenen we eerst een toestandendiagram (fig.5).

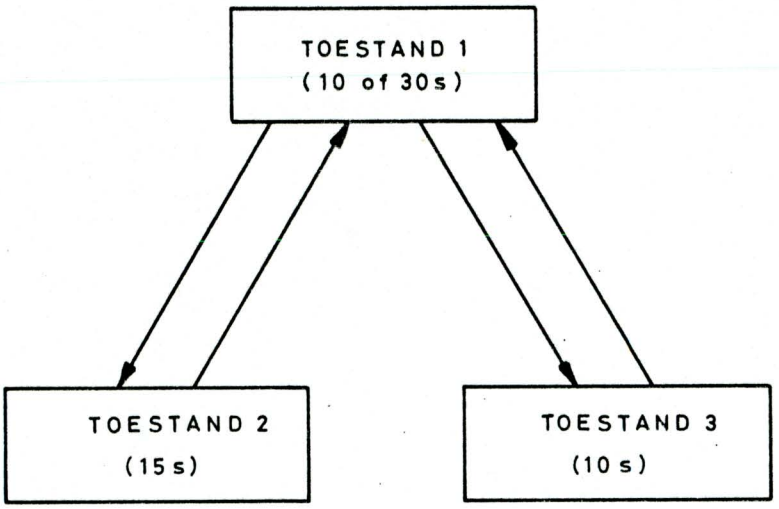


fig.5

In fig.5 bevinden zich geen rechtstreekse pijlen tussen de toestanden 2 en 3. Dit stemt overeen met de probleemomschrijving.
Wanneer toestand 2 of toestand 3 is beëindigd, moet eerst naar toestand 1 worden teruggekeerd.

De toestanden 1 en 2, resp. 1 en 3, mogen echter niet direct in elkaar overgaan. Er moeten immers nog geeltijden worden tussengevoegd.
Het toestandendiagram van fig.5 gaat dan over in dat van fig.6.

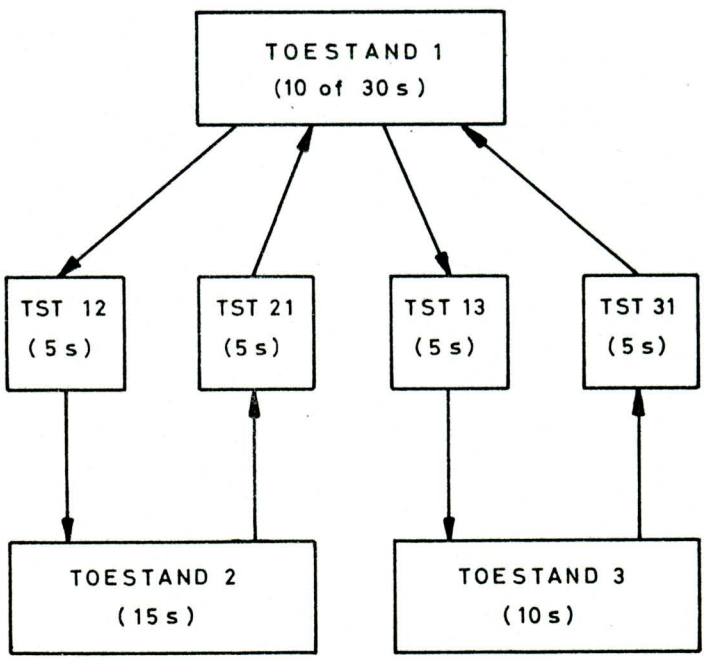


fig.6

Deze "tussentoestanden" duiden we aan met toestand 12, toestand 21, toestand 13 en toestand 31.
Deze nummers zijn niet willekeurig gekozen. B.v. toestand 12 treedt op, wanneer toestand 1 over moet gaan in toestand 2.

In fig.7 is voor elk van de 7 toestanden aangegeven, welke lampen moeten branden.

De te ontwikkelen oplossingsmethode is dan vrij eenvoudig. Na het starten van het programma moet eerst gedurende 10 of 30 seconden toestand 1 heersen. Dan kunnen de detectoren worden getest.

a. Als detector 2 een voertuig heeft gedetecteerd, dan moet naar toestand 2 worden overgegaan. Na 15 seconden moet worden teruggekeerd naar toestand 1.
b. Als detector 5 een voertuig heeft gedetecteerd, dan moet naar toestand 3 worden gesprongen. Na 10 seconden moet worden teruggekeerd naar toestand 1.

Deze gang van zaken is in fig.8 in een stroomdiagram weergegeven.

Vraag 1: Het testen van de detector-flipflops gebeurt in fig.8 wel/niet volkomen correct.

In fig.8 zit echter nog een principiële fout. Stel nl. dat in het spitsuur door alle detectoren continu voertuigen worden gedetecteerd.

Vraag 2: Toestand 1/2/3 wordt dan nooit bereikt.

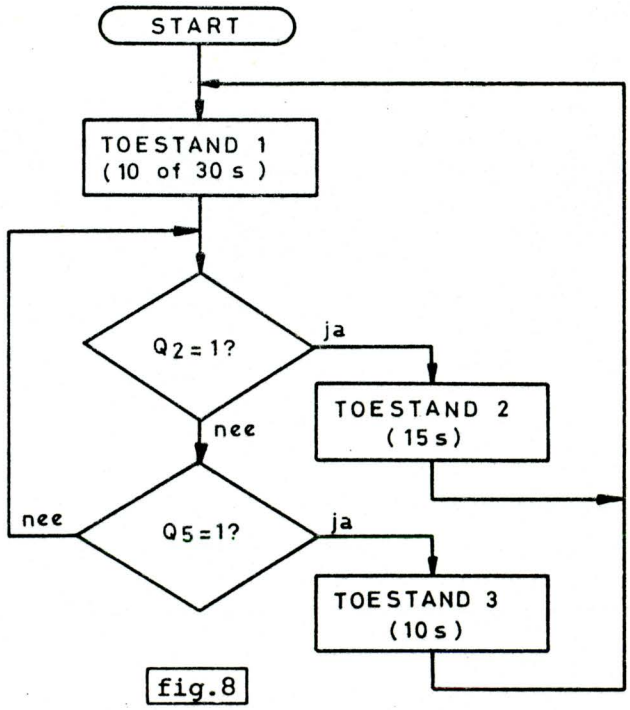


fig.8

Als toestand 1 30 seconden heeft geduurd, wordt eerst de detector-flipflop van richting 2 getest. Er wordt dan naar toestand 2 overgegaan. Deze toestand blijft 15 seconden gehandhaafd. Dan wordt naar toestand 1 teruggekeerd.
Vervolgens wordt na 30 seconden weer eerst de detector-flipflop van richting 2 getest en er wordt weer naar toestand 2 overgegaan. In dit geval wordt toestand 3 dus nooit bereikt.

We moeten er dus voor zorgen, dat als toestand 1 vanuit toestand 2 is bereikt, na 10 resp. 30 seconden eerst Q_5 wordt getest. Omgekeerd moet er dan ook gelden, dat als toestand 1 vanuit toestand 3 is bereikt, na 10 resp. 30 seconden eerst Q_2 wordt getest.

Antw.1: niet. Antw.2: 3.

Een mogelijke oplossing hiervoor is in fig.9a getekend.
In fig.9b is dezelfde oplossingsmethode weergegeven, maar nu zijn tevens de toestanden 12, 21, 13 en 31 aangebracht.

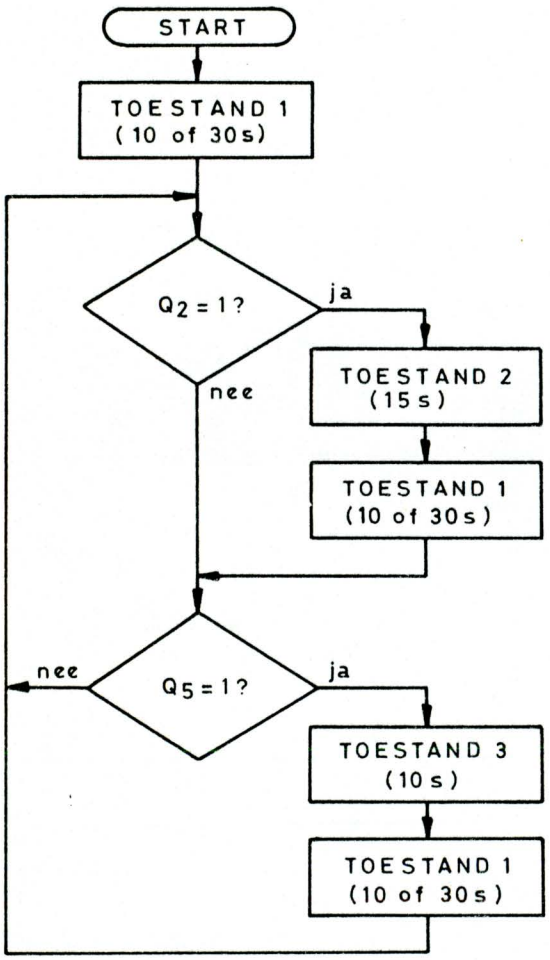


fig.9a

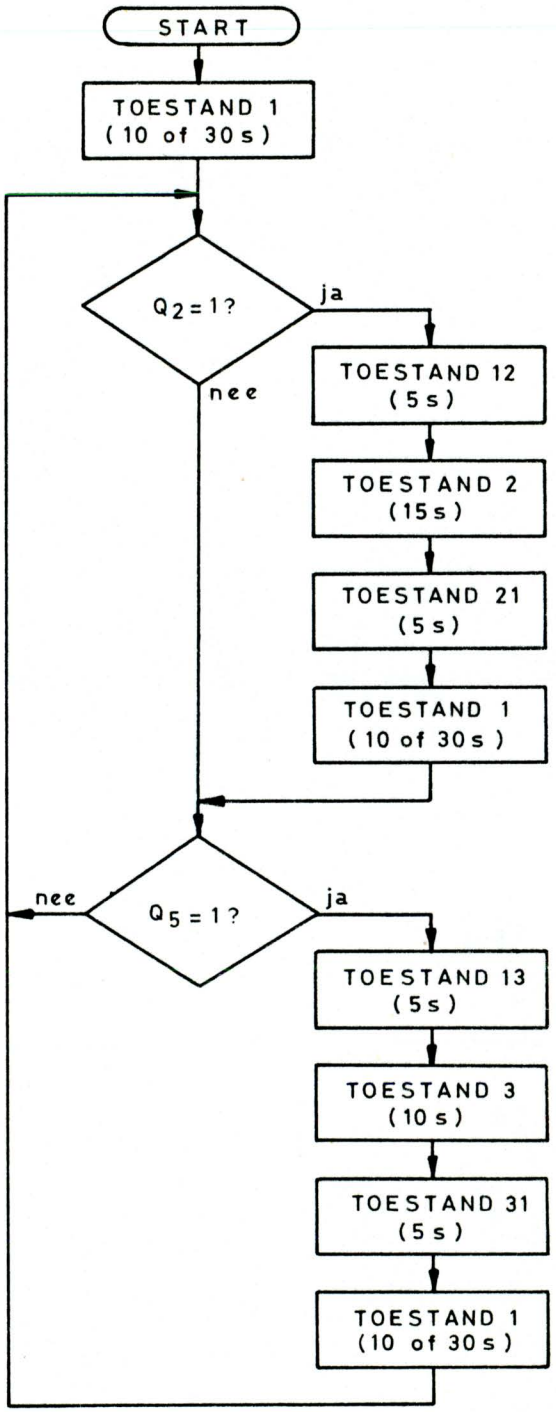
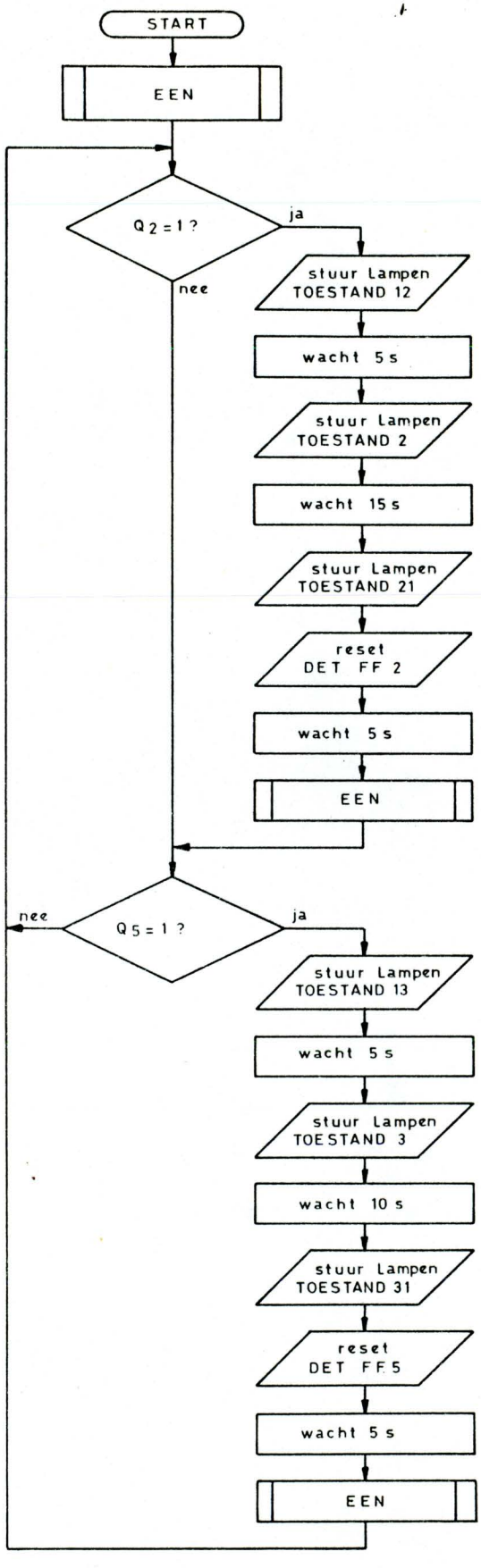


fig.9b


START
EEN
Q2 = 1 ?
ja
nee
stuur Lampen TOESTAND 12
wacht 5 s
stuur Lampen TOESTAND 2
wacht 15 s
stuur Lampen TOESTAND 21
reset DET FF 2
wacht 5 s
EEN
Q5 = 1 ?
nee
ja
stuur Lampen TOESTAND 13
wacht 5 s
stuur Lampen TOESTAND 3
wacht 10 s
stuur Lampen TOESTAND 31
reset DET FF 5
wacht 5 s
EEN

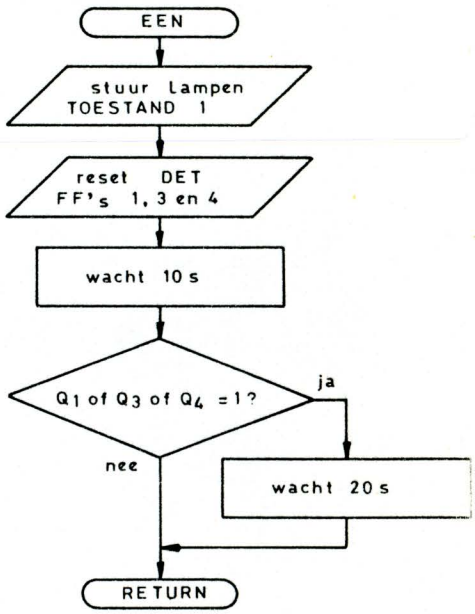

EEN
stuur Lampen TOESTAND 1
reset DET FF's 1, 3 en 4
wacht 10 s
Q1 of Q3 of Q4 = 1 ?
ja
nee
wacht 20 s
RETURN

fig.12

4. ALGEMEEN STROOMDIAGRAM

We hebben in feite het algemeen stroomdiagram al getekend (fig.9b). Hierin moet nog worden aangegeven:

a. op welke wijze wordt bepaald of toestand 1 10 of 30 seconden moet duren,
b. op welke wijze de detector-flipflops moeten worden gereset.

Vraag 3: Toestand 1 duurt 10 seconden als er wel/geen detectie voor de richtingen 1, 3 en 4 heeft plaatsgevonden.

Toestand 1 duurt minimaal 10 seconden.
Alleen als blijkt, dat tijdens deze 10 seconden nog voertuigen in de richtingen 1, 3 en 4 zijn gedetecteerd, dan moet toestand 1 nog 20 seconden extra worden gehandhaafd.
Het stroomdiagram voor het testen van Q_1, Q_3 en Q_4 is in fig.10 weergegeven.

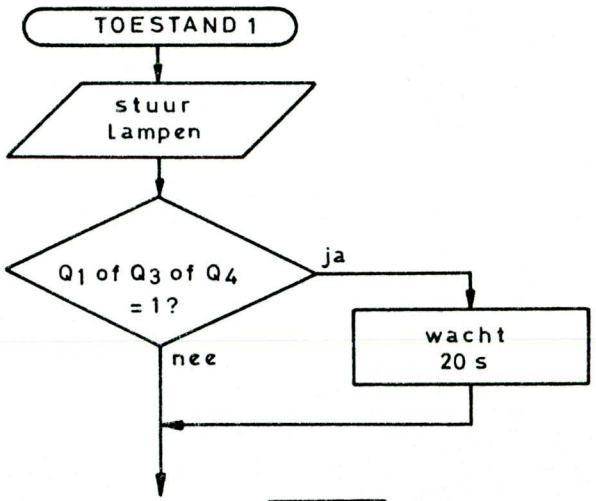


fig.10

Vraag 4: Het resetten van de detector flipflops 1, 3 en 4 moet plaatsvinden tussen de blokken ① en ② / ② en ③ / ③ en ④ van fig. 10.

In blok ③ wordt getest, of tijdens de 10 seconden (blok ②) nog voertuigen zijn gedetecteerd. Dit houdt in, dat direct aan het begin van deze 10 seconden, dus direct voor blok ②, de detector-flipflops 1, 3 en 4 moeten worden gereset.
Het stroomdiagram van fig.10 gaat dan over in dat van fig.11.
Merk op, dat dit stroomdiagram wordt afgesloten met een RETURN-opdracht.
Toestand 1 is dus in een subroutine ondergebracht. Dit is gedaan, omdat in fig.9b drie maal een blok met toestand 1 voorkomt.
Deze subroutine noemen we b.v. "EEN".

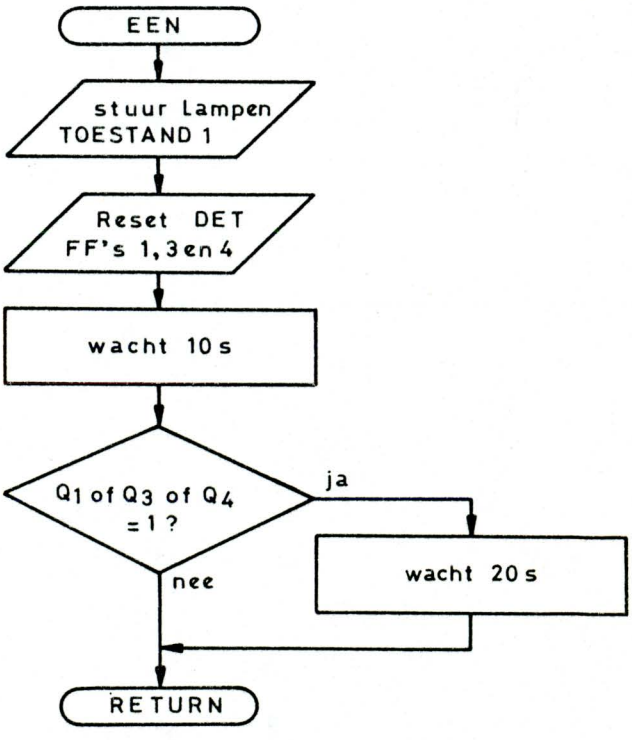


fig.11

Antw.3: geen. Antw.4: ① en ②.

Op dezelfde wijze als in fig.11 is gedaan, moeten we de overige toestanden gedetailleerder beschrijven.

Vraag 5: DET FF2 moet worden gereset, als het licht voor richting 2 op groen/geel/rood springt.

Als richting 2 groen licht heeft, is de toestand van DET FF2 niet belangrijk. Immers, de voertuigen, die de detector passeren, mogen t.g.v. het groene licht doorrijden. Pas als het licht voor richting 2 op geel of rood staat, dan moet de detector-flipflop worden geset, als er een voertuig passeert.
Dit voertuig moet dan nl. wachten, totdat het licht weer op groen springt. DET FF2 moet dus worden gereset, als het licht voor richting 2 op geel springt. Evenzo moet DET FF5 worden gereset, direct nadat richting 5 geel licht heeft gekregen.

Uit fig.9b, fig.11 en bovenstaande toelichting volgt het algemeen stroomdiagram van fig.12.

5. GEDETAILLEERD STROOMDIAGRAM

Bij het tekenen van het gedetailleerd stroomdiagram moeten we uitgaan van de eigenschappen van de toegepaste computer. In ons geval is dat de SDK 85. Wanneer u een ander type computer wenst toe te passen, dan zult u alle in deze paragraaf genoemde machinegerichte details moeten aanpassen. De methode om tot een gedetailleerd stroomdiagram te komen, blijft echter gelijk.

Vraag 6: Voor de verkeerslichtenregeling zijn ..... input-lijnen en ..... output-lijnen vereist.

De interface-schakeling tussen kruispunt en microcomputer vereist de volgende I/O-lijnen:
a. 15 output-lijnen voor het aansturen van de lampen.
b. 5 output-lijnen voor het resetten van de detector-flipflops.
c. 5 input-lijnen, waarop de Q-uitgangen van de detector-flipflops zijn aangesloten.

Vraag 7: Er zijn dan .... 8-bits input-poort(en) en .... 8-bits output-poort(en) nodig.

Voor de 20 output-lijnen moeten we de beschikking hebben over 3 output-poorten. Deze noemen we voorlopig OUPP1, OUPP2 en OUPP3. De 5 input-lijnen vereisen een input-poort, die we met INPP aanduiden.

We moeten nu bepalen welke functie elke I/O-lijn heeft. We maken b.v. de keuze van tabel 3. (U mag hiervan natuurlijk weer afwijken.)

OUPP1	
b_0	rode lamp, richting 1
b_1	gele lamp, richting 1
b_2	groene lamp, richting 1
b_3	rode lamp, richting 2
b_4	gele lamp, richting 2
b_5	groene lamp, richting 2
b_6	rode lamp, richting 3
b_7	gele lamp, richting 3

OUPP3	
b_0	-
b_1	R_1
b_2	R_2
b_3	R_3
b_4	R_4
b_5	R_5
b_6	-
b_7	-

OUPP2	
b_0	groene lamp, richting 3
b_1	rode lamp, richting 4
b_2	gele lamp, richting 4
b_3	groene lamp, richting 4
b_4	rode lamp, richting 5
b_5	gele lamp, richting 5
b_6	groene lamp, richting 5
b_7	-

INPP	
b_0	-
b_1	Q_1
b_2	Q_2
b_3	Q_3
b_4	Q_4
b_5	Q_5
b_6	-
b_7	-

Tabel 3

Antw.6: 5; 20. Antw.7: 1; 3.

Vraag 8: Voor toestand 1 moet via OUPP1 de waarde $\ldots\ldots_{16}$ en via OUPP2 de waarde $\ldots\ldots_{16}$ of $\ldots\ldots_{16}$ worden uitgevoerd.

Voor toestand 1 moet gelden dat de richtingen 1, 3 en 4 groen licht en de richtingen 2 en 5 rood licht hebben.
Via OUPP1 moet dan de combinatie $00001100_2 = 0C_{16}$ worden uitgevoerd.
Via OUPP2 moet dan de combinatie $00011001_2 = 19_{16}$ of $10011001_2 = 99_{16}$ worden uitgevoerd. b_7 van OUPP2 mag zowel 1 als 0 zijn, omdat hierop geen buffer voor een verkeerslicht is aangesloten. In de rest van deze les maken we de niet gebruikte output-lijnen steeds 0.
Tabel 4 geeft dan de combinaties, die bij elk der toestanden via OUPP1 en OUPP2 moeten worden uitgevoerd. Ga deze voor uzelf na.

toestand	OUPP2	OUPP1
TST1	$00011001_2 = 19_{16}$	$00001100_2 = 0C_{16}$
TST2	$00010010_2 = 12_{16}$	$01100100_2 = 64_{16}$
TST3	$01001000_2 = 48_{16}$	$01001001_2 = 49_{16}$
TST12	$00010100_2 = 14_{16}$	$10001100_2 = 8C_{16}$
TST13	$00011000_2 = 18_{16}$	$10001010_2 = 8A_{16}$
TST21	$00010010_2 = 12_{16}$	$01010100_2 = 54_{16}$
TST31	$00101000_2 = 28_{16}$	$01001001_2 = 49_{16}$

Tabel 4

Vraag 9: Om DET FF2 te resetten, moet via OUPP3 de waarde $\ldots\ldots_{16}$ worden uitgevoerd.

Een detector-flipflop wordt gereset door kortstondig een 0 op de betreffende reset-ingang te plaatsen. Als we alleen DET FF2 willen resetten, moet R_2 dus kortstondig 0 zijn. De overige reset-ingangen moeten 1 blijven. Via OUPP3 voeren we dan de waarde $00111010_2 = 3A_{16}$ uit.

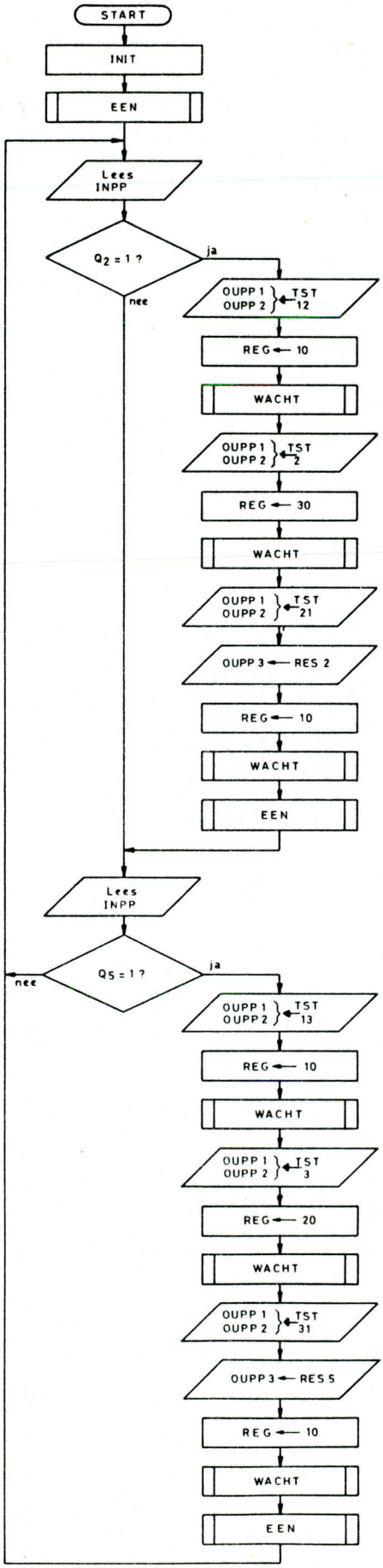
In tabel 5 zijn de combinaties weergegeven, die we nodig hebben om de detector-flipflops te resetten.

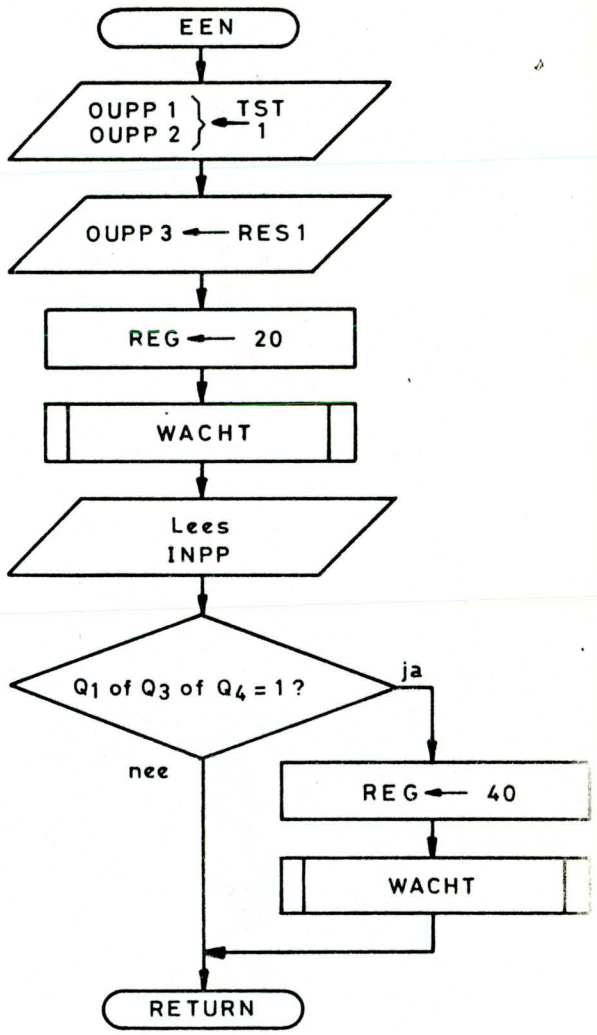
naam	waarde	functie
RES1	$00100100_2 = 24_{16}$	reset DET FF's 1, 3 en 4
RES2	$00111010_2 = 3A_{16}$	reset DET FF2
RES5	$00011110_2 = 1E_{16}$	reset DET FF5

Tabel 5

Het testen van de Q-uitgangen van de detector flipflops is ook geen probleem. Dit doen we door het maskeren van bits m.b.v. de EN-functie.

Het enige punt in fig.12 wat dan nog overblijft, is het creëren van de verschillende wachttijden (5 s, 10 s, 15 s en 20 s).


START
INIT
EEN
Lees INPP
Q2 = 1 ?
ja
nee
OUPP 1 OUPP 2 ← TST 12
REG ← 10
WACHT
OUPP 1 OUPP 2 ← TST 2
REG ← 30
WACHT
OUPP 1 OUPP 2 ← TST 21
OUPP 3 ← RES 2
REG ← 10
WACHT
EEN
Lees INPP
Q5 = 1 ?
ja
nee
OUPP 1 OUPP 2 ← TST 13
REG ← 10
WACHT
OUPP 1 OUPP 2 ← TST 3
REG ← 20
WACHT
OUPP 1 OUPP 2 ← TST 31
OUPP 3 ← RES 5
REG ← 10
WACHT
EEN


EEN
OUPP 1 OUPP 2 ← TST 1
OUPP 3 ← RES 1
REG ← 20
WACHT
Lees INPP
Q1 of Q3 of Q4 = 1 ?
ja
nee
REG ← 40
WACHT
RETURN

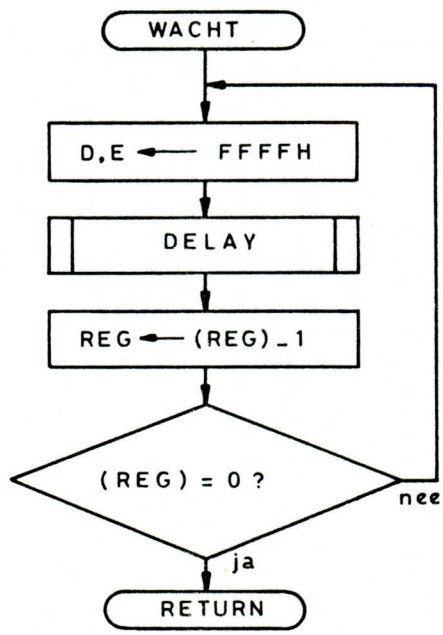

WACHT
D,E ← FFFFH
DELAY
REG ← (REG) _ 1
(REG) = 0 ?
nee
ja
RETURN

fig.15

Vraag 10: Hiervoor gebruiken we de monitor-subroutine ".....".
De maximale wachttijd van deze subroutine is ca.
1 ms/0,5 s/1 s/5 s.

Hiervoor kunnen we gebruik maken van de monitor subroutine "DELAY". Als we voor het aanroepen van deze subroutine registerpaar D,E met $FFFF_{16}$ vullen, dan neemt de uitvoering van deze subroutine iets meer dan 0,5 seconde in beslag. Om een wachttijd van b.v. 15 s te bereiken, moet "DELAY" dan 30 maal worden aangeroepen.

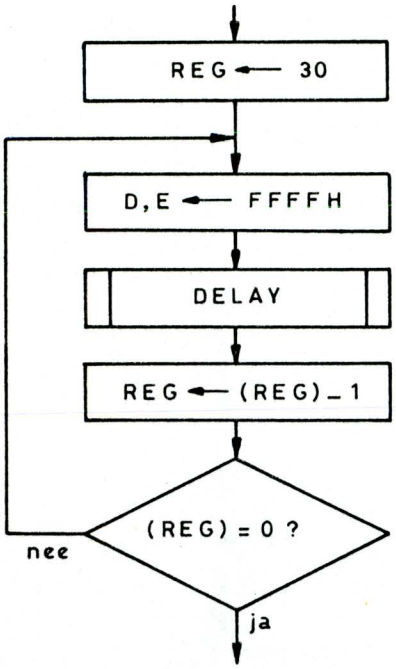


fig.13

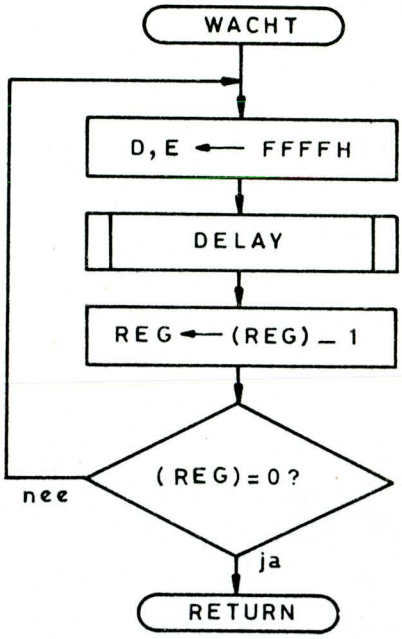


fig.14a

Dit doen we natuurlijk m.b.v. een programmalus. We vullen een register met 30. Elke keer wanneer "DELAY" is uitgevoerd, verlagen we de inhoud van dit register met 1, totdat de inhoud nul is geworden (fig.13).

In fig.12 komen 8 blokken voor, waarin een wachttijd dient te worden gerealiseerd. Dit houdt in, dat als we de methode van fig.13 kiezen, deze programmalus ook 8 maal in het programma moet worden opgenomen. Een oplossing, die geheugenruimte spaart, is in fig.14 gegeven. De programmalus is in een subroutine, b.v. "WACHT", ondergebracht (fig. 14a). Deze subroutine wordt dan aangeroepen d.m.v. de 2 opdrachten van fig. 14b.

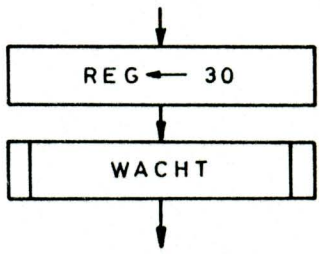


fig.14b

Antw.10: DELAY; 0,5 s.

Vraag 11: Voor een wachttijd van 5 s moet "REG" worden gevuld met de waarde $.....10 = ...16$.

Als we een wachttijd van 5 s willen bereiken, dan moet "DELAY" 10 maal worden aangeroepen. We moeten "REG" dan vullen met de waarde $10_{10} = 0A_{16}$.

Opmerking: Registerpaar D,E wordt gevuld met $FFFF_{16}$. "DELAY" duurt dan iets langer dan 0,5 seconde. Ook de instructies uit de programmalus (fig.14a) en die uit het hoofdprogramma nemen een zekere tijd in beslag. De timing speelt echter niet een echt nauwkeurige rol. Het maakt n.l. niet zoveel uit, of b.v. een verkeerslicht 10 s of 10,5 s op rood staat. Wanneer tijdens het testen van het programma blijkt, dat de tijden erg veel afwijken van de gestelde tijden, dan kunnen we altijd nog de waarde $FFFF_{16}$ aanpassen.

Met behulp van de toelichting in deze paragraaf kunnen we nu het algemeen stroomdiagram van fig.12 omzetten in het gedetailleerd stroomdiagram van fig.15.

Antw.11: 10; 0A.

```
                        ORG     2800H
                TST1    EQU     190CH
                TST2    EQU     1264H
                TST3    EQU     4849H
                TST12   EQU     148CH
                TST13   EQU     188AH
                TST21   EQU     1254H
                TST31   EQU     2849H
                RES1    EQU     24H
                RES2    EQU     3AH
                RES5    EQU     1EH
                UMSK1   EQU     1AH
                UMSK2   EQU     04H
                UMSK5   EQU     20H
                INPP    EQU     00H
                OUPP1   EQU     21H
                OUPP2   EQU     22H
                OUPP3   EQU     01H
                DELAY   EQU     05F1H
2800  31 C2 20  INIT:   LXI     SP,20C2H  ;INITIALISATIE
2803  F3                DI
2804  3E 03             MVI     A,03H
2806  D3 20             OUT     20H       ;POORT 21H = OUTPUT
2808  32 FF 20          STA     20FFH     ;POORT 22H = OUTPUT
280B  AF                XRA     A
280C  D3 02             OUT     02H       ;POORT 00H = INPUT
280E  2F                CMA
280F  D3 03             OUT     03H       ;POORT 01H = OUTPUT
2811  CD 7D 28          CALL    EEN
2814  DB 00    TEST2:    IN      INPP
2816  E6 04             ANI     UMSK2     ;UNMASK Q2
2818  CA 47 28          JZ      TEST5     ;SPRING ALS Q2 = 0
281B  21 8C 14          LXI     H,TST12   ;TOESTAND 12
281E  CD A6 28          CALL    LICHT     ;  STUUR LAMPEN
2821  06 0A             MVI     B,10D     ;WACHT
2823  CD 9B 28          CALL    WACHT     ;  5 SECONDEN
2826  21 64 12          LXI     H,TST2    ;TOESTAND 2
2829  CD A6 28          CALL    LICHT     ;  STUUR LAMPEN
282C  06 1E             MVI     B,30D     ;WACHT
282E  CD 9B 28          CALL    WACHT     ;  15 SECONDEN
2831  21 54 12          LXI     H,TST21   ;TOESTAND 21
2834  CD A6 28          CALL    LICHT     ;  STUUR LAMPEN
2837  3E 3A             MVI     A,RES2    ;RESET
2839  D3 01             OUT     OUPP3     ;  DET FF2
283B  3E 3E             MVI     A,3EH
283D  D3 01             OUT     OUPP3
283F  06 0A             MVI     B,10Ø     ;WACHT
2841  CD 9B 28          CALL    WACHT     ;  5 SECONDEN
2844  CD 7D 28          CALL    EEN       ;TOESTAND 1
2847  DB 00    TEST5:    IN      INPP
2849  E6 20             ANI     UMSK5     ;UNMASK Q5
284B  CA 14 28          JZ      TEST2     ;SPRING ALS Q5 = 0
284E  21 8A 18          LXI     H,TST13   ;TOESTAND 13
2851  CD A6 28          CALL    LICHT     ;  STUUR LAMPEN
2854  06 0A             MVI     B,10D     ;WACHT
2856  CD 9B 28          CALL    WACHT     ;  5 SECONDEN
2859  21 49 48          LXI     H,TST3    ;TOESTAND 3
285C  CD A6 28          CALL    LICHT     ;  STUUR LAMPEN
285F  06 14             MVI     B,20D     ;WACHT
2861  CD 9B 28          CALL    WACHT     ;  10 SECONDEN
2864  21 49 28          LXI     H,TST31   ;TOESTAND 31
2867  CD A6 28          CALL    LICHT     ;  STUUR LAMPEN
286A  3E 1E             MVI     A,RES5    ;RESET
286C  D3 01             OUT     OUPP3     ;  DET FF5
286E  3E 3E             MVI     A,3EH
2870  D3 01             OUT     OUPP3
2872  06 0A             MVI     B,10D     ;WACHT
2874  CD 9B 28          CALL    WACHT     ;  5 SECONDEN
2877  CD 7D 28          CALL    EEN       ;TOESTAND 1
287A  C3 14 28          JMP     TEST2
287D  21 0C 19  EEN:      LXI     H,TST1    ;TOESTAND 1
2880  CD A6 28          CALL    LICHT     ;  STUUR LAMPEN
2883  3E 24             MVI     A,RES1    ;RESET DET FF'S
2885  D3 01             OUT     OUPP3     ;  1, 3 EN 4
2887  3E 3E             MVI     A,3EH
2889  D3 01             OUT     OUPP3
288B  06 14             MVI     B,20D     ;WACHT
288D  CD 9B 28          CALL    WACHT     ;  10 SECONDEN
2890  DB 00             IN      INPP
2892  E6 1A             ANI     UMSK1     ;UNMASK Q1, Q3, Q4
2894  C8                RZ
2895  06 28             MVI     B,40D     ;WACHT
2897  CD 9B 28          CALL    WACHT     ;  20 SECONDEN
289A  C9                RET
289B  11 FF FF  WACHT:    LXI     D,FFFFH   ;WACHT
289E  CD F1 05          CALL    DELAY     ;  0.5 SECONDE
28A1  05                DCR     B         ;(REG)-1
28A2  C2 9B 28          JNZ     WACHT     ;SPRING ALS (REG)≠0
28A5  C9                RET
28A6  7D        LICHT:    MOV     A,L       ;STUUR
28A7  D3 21             OUT     OUPP1     ; LAMPEN
28A9  7C                MOV     A,H       ;  VANUIT
28AA  D3 22             OUT     OUPP2     ;   H EN L
28AC  C9                RET
                        END
```

fig.16

6. PROGRAMMA

Voordat we uit fig.15 een programma kunnen schrijven, moeten we nog aandacht besteden aan de initialisatie en de gebruikte symbolische namen.

Vraag 12: Er zijn totaal ..... I/O-poorten nodig.

In totaal zijn er 4 I/O-poorten vereist. Hiervan moeten er 3 als output-poort fungeren. Tot nu toe hebben we de 4 I/O-poorten van de SDK 85 steeds gesplitst in 2 input- en 2 output-poorten.
We moeten een van de poorten, die steeds als input-poort werkte, nu tot output-poort omvormen. We maken b.v. de keuze van tabel 6.

naam	SDK-poort
OUPP1	poort 21_{16}
OUPP2	poort 22_{16}
OUPP3	poort 01_{16}
INPP	poort 00_{16}

Tabel 6

Vraag 13: Het data direction register voor poort 01_{16} moet worden gevuld met $\ldots\ldots_{16}$.

Omdat poort 01_{16} nu als output-poort dient te werken, moet het data direction register worden gevuld met $11111111_2 = FF_{16}$.

Vraag 14: De interrupt controller moet wel/niet worden geïnitialiseerd.

Omdat in dit programma geen sprake van interrupts is, behoeven we de interrupt controller niet te initialiseren. Voor alle zekerheid kunnen we aan het begin van het programma een DI-instructie opnemen om eventueel toch optredende interrupt requests te laten negeren.

Van alle gebruikte symbolische namen, behoeven we alleen "REG" nog om te zetten in een register- of een geheugenadres.

Vraag 15: Voor "REG" kiezen we bij voorkeur niet een van de registers ....., ... en .....

Subroutine "DELAY" beïnvloedt de inhouden van de accumulator en registerpaar D,E. Door de rest van het programma wordt geen van de overige registers gereserveerd. Voor "REG" kiezen we dan b.v. register B.

We kunnen nu uit fig.15 een programma schrijven.

PROBEER HET EERST ZELF. TEST UW PROGRAMMA OP DE SDK 85.
BESTUDEER DAARNA ONZE OPLOSSING IN FIG.16.

Antw.12: 4. Antw.13: FF. Antw.14: niet. Antw.15: A; D; E.

Let in dit programma vooral op het toepassen van de subroutine "LICHT" en de symbolische namen "UMSK1", "UMSK2" en "UMSK5".

Een belangrijk verschijnsel in dit programma is, dat we in de instructies MVI B,10D, MVI B,20D enz. decimale waarden vermelden. Als dit programma door een assembler zou worden vertaald, dan worden deze decimale waarden in binaire getallen omgezet.

In de assembly-taal voor 8080- en 8085-microprocessoren wordt een decimaal getal aangegeven, door achter de waarde de letter D te plaatsen. Op dezelfde wijze geven we een hexadecimaal getal met H en een binair getal met B aan. Zo zullen b.v. onderstaande 3 instructies alle hetzelfde resultaat opleveren:

```
MVI B,14H
MVI B,20D
MVI B,00010100B
```

Opmerking: Door 8080- en 8085-assemblers wordt een getal, dat niet eindigt op een van de letters B, D of H, als een decimaal getal beschouwd.

7. TESTEN OP DE SDK 85

Om dit programma volledig te kunnen testen, dient u de interface uit fig.2 te bouwen en op de SDK 85 aan te sluiten.

Voor elk van de 15 buffers kunt u één van de schakelingen uit fig.16 gebruiken. I.p.v. de LED's en voorschakelweerstanden kunt u natuurlijk ook gloeilampjes toepassen. Denk dan om de maximale stroom.

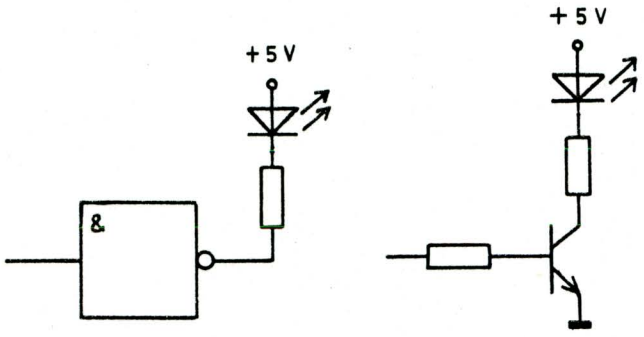


fig.16

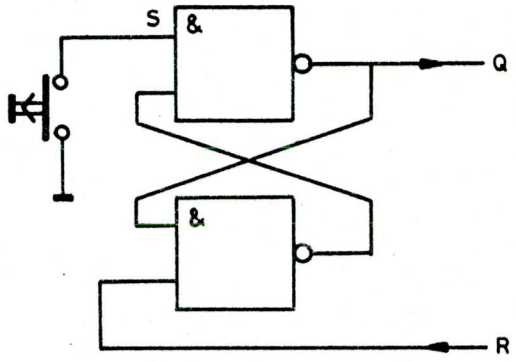


fig.17

De detectoren kunt u dan simuleren door druktoetsen op de S-ingangen van de 5 detector-flipflops aan te sluiten (fig.17).

Als u deze hardware niet wilt bouwen, dan kunt u deze voor een groot deel simuleren op de uitbreidingsprint van de SDK 85. De 15 verkeerslichten worden dan voorgesteld door LED's. Leg voor het gemak een papiertje naast de LED's waarop voor elke LED is aangegeven, welk verkeerslicht hierbij behoort (fig.18).

De detectors met de detector-flipflops worden gesimuleerd met de schakelaars van poort 00_{16}. Om een voertuig te laten detecteren, moet u de betreffende schakelaar in de 1-stand zetten. U moet er dan zelf aan denken, dat wanneer de betreffende richting groen licht heeft gekregen, de schakelaar weer in de 0-stand wordt gezet. De door de computer afgegeven reset-signalen hebben n.l. nu geen enkele invloed op de Q-lijnen.

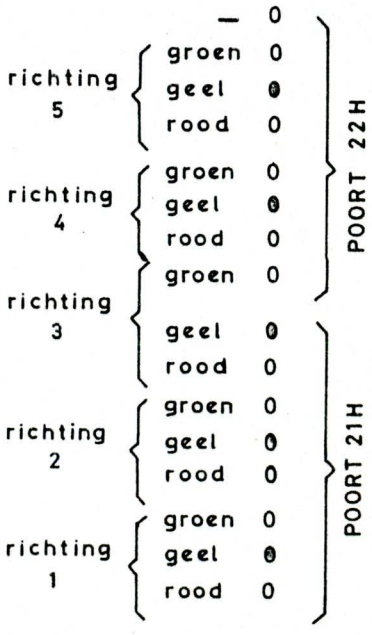


fig.18

8. WIJZIGEN VAN HET PROGRAMMA

Als u het programma van fig.16 of uw eigen programma heeft getest, dan kunt u nog wijzigingen aanbrengen. We doen hiervoor enkele suggesties.

a. De subroutine "WACHT" maakt gebruik van de monitor-subroutine "DELAY". Wanneer we het ontwikkelde programma in een andere microcomputer plaatsen, dan is "DELAY" hierin waarschijnlijk niet aanwezig. "WACHT" moet nu zo gewijzigd worden, dat de wachttijd van 0,5 seconde m.b.v. een programmalus wordt gerealiseerd.

b. Als aan het eind van de groentijd voor toestand 2 een voertuig wordt gedetecteerd, dan wordt in het programma van fig.16 toch naar toestand 21 overgegaan. In de praktijk bevinden de detectoren zich enkele tientallen tot honderden meters vóór de verkeerslichten. De kans is dus groot, dat een voertuig, dat 1 seconde voor het eind van toestand 2 detector 2 passeert, voor geel of rood licht moet stoppen.
Wijzig het programma zodanig, dat als in de laatste 2 seconden van toestand 2 een voertuig de detector passeert, toestand 2 eenmalig met 4 seconden wordt verlengd.
Fig.19 geeft hiervoor de oplossingsmethode.

c. Dezelfde wijziging als in punt b, maar nu voor toestand 3 en detector 5.

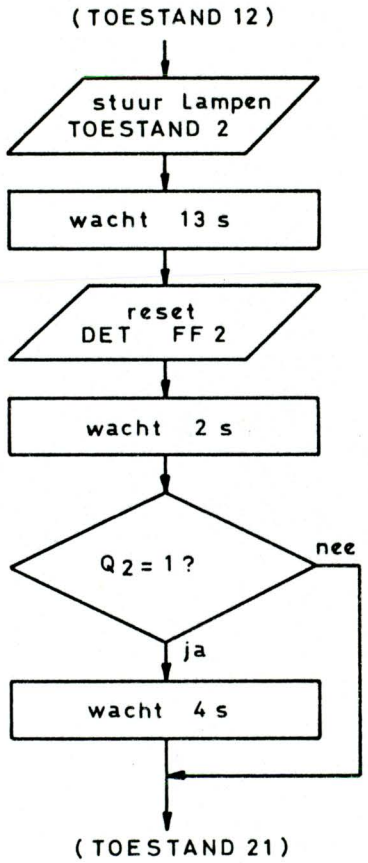


fig.19

DATALOGER.

Datalogger

1. INLEIDING

Toen de mensheid het zich kon veroorloven om zich niet meer alleen bezig te houden met het voorzien in de primaire levensbehoeften, is zij zich op haar omgeving gaan richten.
In het begin werden alleen door de natuur veroorzaakte veranderingen gemeten, zoals hoeveelheid neerslag, temperatuurwisselingen, enz.
Door de resultaten van deze waarnemingen over een lange tijd te beschouwen is men tot de indeling van de jaargetijden en later tot kalenders gekomen.
Naderhand was het mogelijk om aan door de mens gestarte processen te meten, b.v. snelheidsveranderingen van rijdende voertuigen, de tijd die nodig is om een liter water te koken, de levensduur van gloeilampen enz.

Naast het vastleggen van de meetresultaten, probeerde men door bestudering hiervan de toekomst ermee te voorspellen. M.a.w. uit de gegeven situatie werd afgeleid, wat er enige tijd later zou gebeuren.

Als men op deze wijze het toekomstige gedrag van een proces wil bepalen, dan moet men gedurende een bepaalde tijd zeer nauwkeurig de meetresultaten en de tijdstippen van de metingen vastleggen. Bovendien is het nodig, dat de tussen de metingen liggende tijden contstant zijn. De grafieken, waarin de resultaten worden weergegeven, zijn dan n.l. overzichtelijker en gemakkelijker op te stellen. Uit het verloop van de grafieken kan men dan door extrapolatie met enige zekerheid de toekomstige veranderingen in een proces voorspellen.
Onder extrapolatie verstaan we het "doortrekken" van de lijnen, zoals we denken, dat deze in de toekomst zullen verlopen. Een voorbeeld hiervan is in fig.1 weergegeven. Tussen t=1 en t=2 zijn een aantal metingen verricht. De resultaten zijn in de grafiek gelegd. De gestippelde lijnen geven aan, hoe volgens ons deze lijnen door moeten lopen.

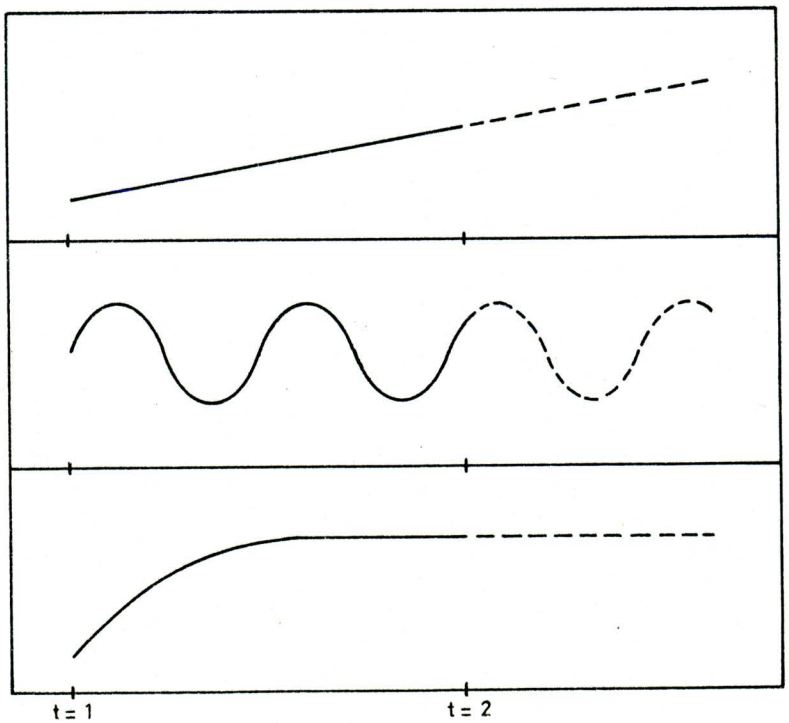


fig.1

Elektronica
opleidingen
Dirksen

Het verzamelen en noteren van al die gegevens gebeurde oorspronkelijk met de hand. Omdat de processen veel sneller gingen verlopen en men sneller en meer verschillende gegevens verwerkte, werd dat ondoenlijk. Er werden apparaten ontwikkeld, die in hoog tempo meetresultaten konden binnenhalen, opslaan en desgewenst weergeven. Dit zijn de z.g. dataloggers (to log = noteren). Deze apparaten moeten variabelen (temperatuur, druk enz.) van een proces omzetten in b.v. een evenredige mechanische beweging (de bestuurbare inktpen van een XY-recorder) of een elektrische spanning, zodat ze kunnen worden opgeslagen op magneetband.
Op deze wijze kan men zeer veel gegevens automatisch laten vastleggen. Het probleem is nu hoe we zo'n grote hoeveelheid gegevens verwerken tot relevante informatie. We zijn n.l. meestal niet geïnteresseerd in de gegevens zelf, maar in hieruit afgeleide grootheden, zoals minimum, maximum en gemiddelde waarden. Dit verwerken van gegevens kan men automatisch laten verrichten door b.v. een microprocessorsysteem. Een voorwaarde daarbij is dat de datalogger de gegevens in binaire vorm afgeeft.

Een volgende stap is het onderbrengen van de gegevensverwerkende microcomputer in de datalogger. Het is dan voor de hand liggend, dat ook de invoer, het eventueel sorteren en de opslag van de meetresultaten door dezelfde microprocessor wordt bestuurd. Op deze wijze worden door de "intelligente" datalogger direct de ontvangen gegevens bewerkt tot relevante informatie. Dit heeft als voordeel dat er veel geheugenruimte kan worden bespaart (zie ook de volgende paragraaf).

Het systeem moet dan zodanig zijn opgebouwd, dat de gebruiker gemakkelijk informatie kan opvragen. We kunnen zelfs nog een stap verder gaan, door een aantal intelligente dataloggers via een interface op een centrale computer aan te sluiten. Een voorbeeld hiervan is het landelijk meetnet voor luchtverontreiniging. Op verschillende plaatsen in Nederland staan intelligente dataloggers, de z.g. snuffelpalen, die elk de hoeveelheid in de lucht meten. Uit de meetresultaten worden maximale, minimale en gemiddelde percentages berekend. Een centrale computer vraagt regelmatig van alle dataloggers informatie op. Hieruit wordt dan in de centrale computer een totaaloverzicht opgesteld.

In deze les zullen we zo'n intelligente datalogger realiseren.
Om u in staat te stellen het uiteindelijke programma te testen, gaan we in de laatste fase van het ontwikkelingsproces weer uit van de SDK 85.

2. INTELLIGENTE DATALOGGER

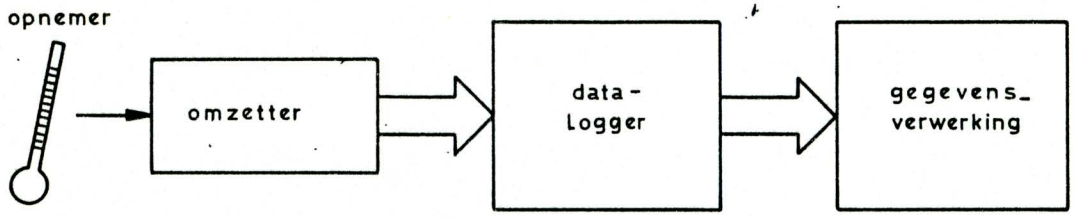


fig.2

In fig.2 is een conventionele datalogger weergegeven, waarin elke minuut een meetwaarde van de omzetter wordt opgeslagen. Stel nu dat gedurende 24 uur steeds over een periode van 1 uur de minimale, de maximale en de gemiddelde temperatuur moet worden bepaald. De datalogger in fig.2 moet, omdat er geen intelligentie in aanwezig is, alle meetwaarden over een periode van 24 uur opslaan.

Vraag 1: Hiertoe moet geheugenruimte voor ..... getallen worden gereserveerd.

Aangezien er elke minuut een nieuwe meetwaarde wordt opgeslagen, moet de datalogger beschikken over een stuk geheugen voor 24 x 60 = 1440 getallen. Pas aan het eind van die 24 uur kunnen de opgeslagen gegevens worden verwerkt.

In fig.3 vindt dezelfde gang van zaken m.b.v. een intelligente datalogger plaats. Deze verwerkt direct de ingevoerde meetwaarde.

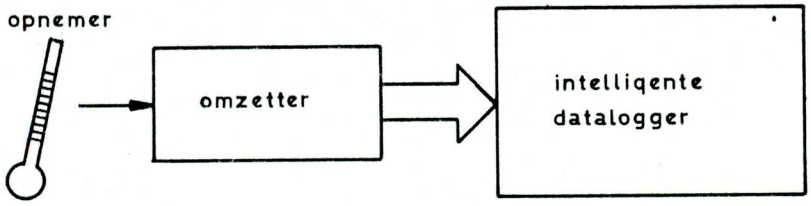


fig.3

Vraag 2: Nu moet geheugenruimte voor ..... getallen worden gereserveerd.

Aangezien elke meetwaarde direct wordt verwerkt, behoeven alleen nog maar de maximale, minimale en gemiddelde waarde van elk uur te worden opgeslagen. Hiervoor zijn 24 x 3 = 72 geheugenlocaties voldoende. Na 24 uur kan door de gebruiker (of een andere computer) deze relevante informatie worden opgevraagd.

Antw.1: 1440. Antw.2: 72.

3. PROBLEEMOMSCHRIJVING

In een zekere stad wil men de doorstroming van het verkeer gaan optimaliseren. Hiertoe moeten eerst op een aantal, verkeerstechnisch gezien, belangrijke plaatsen verkeerstellingen worden verricht. Bij de meetpunten moeten intelligente dataloggers worden opgesteld. Alle dataloggers zijn verbonden met een centrale minicomputer, die van tijd tot tijd de in de dataloggers opgeslagen informatie opvraagt. Ook als m.b.v. de resultaten van de verkeerstellingen een nieuw verkeersplan is ontwikkeld en gerealiseerd, dan blijven de intelligente dataloggers werkzaam. De centrale minicomputer zal dan n.l. worden gebruikt om het aansturen van de verkeerslichten op de afzonderlijke kruispunten te coördineren. Hiertoe moet de minicomputer beschikken over het verkeersaanbod op een aantal plaatsen in het te regelen gebied.
Gevraagd wordt nu zo'n door een microprocessor bestuurde intelligente datalogger te ontwikkelen. Het blokschema hiervoor is in fig.4 weergegeven.

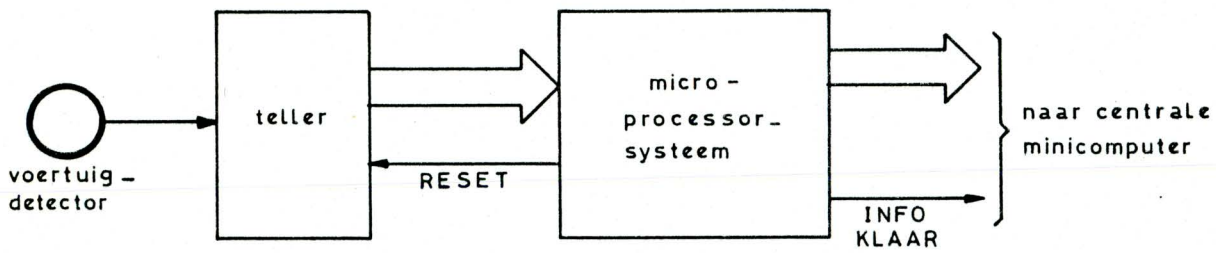


fig.4

De in het wegdek aangebrachte detector geeft een korte impuls af, op het moment, dat een voertuig de detector passeert. Deze impulsen worden toegevoerd aan de klokingang van een 8-bits binaire teller. Deze teller houdt dus het aantal gedetecteerde voertuigen bij. Het microprocessorsysteem leest regelmatig de tellerinhoud uit en geeft direct daarna een reset-impuls aan de teller af, zodat deze weer opnieuw zal beginnen met het tellen van voertuigen.

De eisen, die aan het microprocessorsysteem worden gesteld zijn:

a. Elke minuut moet de tellerinhoud worden uitgelezen. Direct na het uitlezen moet de teller worden gereset.

b. Gedurende een uur moet elke ingevoerde waarde worden opgeteld bij het totaal van de voorgaande waarden. Na een uur is het resultaat van deze optellingen dus gelijk aan het aantal in dat uur gedetecteerde voertuigen.

c. Per uur wordt het maximum aantal per minuut gedetecteerde voertuigen bepaald, m.a.w. de hoogste tellerinhoud, die in dat uur is ingelezen, moet worden opgeslagen. Hierbij moet tevens worden opgeslagen in welke minuut dit aantal was gedetecteerd.

d. Idem als punt c, maar nu voor wat betreft het minimum aantal per minuut gedetecteerde voertuigen.

e. De in b, c en d genoemde handelingen worden voor elk heel uur van de dag (= 24 uren) herhaald. Het gevolg is, dat na 24 uur een tabel in het geheugen is opgeslagen, die 72 relevante gegevens bevat (fig.5).

f. Om precies 24.00 uur = 0.00 uur moet de INFO KLAAR-lijn actief worden gemaakt. Direct daarna moeten alle getallen in de volgorde waarin ze in het geheugen zijn opgeslagen (fig.6), via een output-poort worden uitgevoerd. Dit uitvoeren moet zo snel mogelijk plaatsvinden. De minicomputer werkt n.l. veel sneller dan het microprocessorsysteem en kan op deze wijze informatie van een groot aantal van deze intelligente dataloggers tegelijkertijd invoeren en opslaan. Nadat alle waarden zijn uitgevoerd moet het actieve signaal op de INFO KLAAR-lijn worden weggenomen.

De tabel in het geheugen moet na 24 uur de waarden van fig.5. bevatten.

MAX 0	MMAX 0
MAX 1	MMAX 1
MAX 2	MMAX 2
MAX 22	MMAX 22
MAX 23	MMAX 23

MIN 0	MMIN 0
MIN 1	MMIN 1
MIN 2	MMIN 2
MIN 22	MMIN 22
MIN 23	MMIN 23

TOTAAL 0
TOTAAL 1
TOTAAL 2
TOTAAL 22
TOTAAL 23

fig.5

De 24 uren van een etmaal zijn genummerd van 0 t/m 23. Zo valt de tijd van 0.00 t/m 0.59 uur in UUR0. UUR21 loopt van 21.00 t/m 21.59 uur. In fig.5 is MAX2 de maximale waarde, die in UUR2 is gemeten. MMAX2 is dan de minuut in welke deze maximale waarde is gemeten. Stel dat b.v. de inhouden van MAX5 en MMAX5 19_{10} resp. 37_{10} zijn. Dan wil dat zeggen dat in UUR5, dus tussen 5.00 en 5.59 uur, het grootste verkeersaanbod van 19 voertuigen is gemeten in de minuut, die begon op 5.37 uur. Als er in de tabel geldt, dat de inhouden van MIN1 en MMIN1 3_{10} resp. 48_{10} zijn, dan geeft dat aan, dat tussen 1.00 en 1.59 uur het verkeersaanbod in de minuut, die begon op 1.48 uur het kleinst was. In deze minuut zijn slechts 3 voertuigen gedetecteerd.

In het meest rechtse deel van fig.5 zijn de totale aantallen gedetecteerde voertuigen per uur opgeslagen. De centrale minicomputer kan hieruit eenvoudig gemiddelde waarden berekenen, door de totale aantallen door 60 te delen.

De gemeten waarden en bijbehorende aantallen minuten staan natuurlijk niet naast elkaar maar achter elkaar in het geheugen. Dit is in fig.6 weergegeven.

MAX 0
MMAX 0
MAX 1
MMAX 1
MAX 23
MMAX 23
MINO
MMINO
MIN 23
MMIN 23
TOTAAL 0
TOTAAL 1
TOTAAL 23

fig.6

Vraag 3: Fig.6 beslaat ..... 8-bits geheugenwoorden.

De waarden MAX0 t/m MAX23 bestaan elk uit 1 byte. Immers dit zijn steeds de grootste van een aantal ingevoerde 8-bits meetwaarden. De bijbehorende MMAX0 t/m MMAX23 zijn elk ook 1 byte groot. Hierin wordt n.l. een aantal minuten opgeslagen, dat van 0_{10} t/m 59_{10} kan zijn. Evenzo beslaan MIN0 t/m MIN23 en MMIN0 t/m MMIN23 elk 1 byte. Voor de 24 totaalwaarden moeten 48 bytes worden gereserveerd. Ze ontstaan immers door het optellen van 60 8-bits meetwaarden. Voor resultaat moeten dan 2 bytes beschikbaar zijn.

De voor fig.6 noodzakelijke geheugenruimte moet dus uit 144 8-bits geheugenwoorden bestaan.

Antw.3: 144.

Enkele aanvullende eisen zijn:

a. Alle in de tabel opgeslagen waarden voor MAX0 t/m MAX23, MIN0 t/m MIN23 en TOTAAL0 t/m TOTAAL23 zijn binaire waarden.
b. De minuten MMAX en MMIN moeten in BCD-code worden opgeslagen.
c. Voor de signalen RESET en INFO KLAAR geldt: 0 = actief.

VOER NU ZELF EEN PROBLEEMANALYSE UIT EN ONTWIKKEL DAN VIA STROOM-DIAGRAMMEN HET PROGRAMMA.
BEDENK EEN METHODE OM DIT PROGRAMMA TE TESTEN EN VOER DEZE TEST DAN UIT. BESTUDEER DAARNA ONZE OPLOSSING.

4. PROBLEEMANALYSE

Uit de probleemomschrijving volgt, dat de werkzaamheden van de datalogger uit 3 in elkaar opgenomen cycli bestaan. Dit is in fig.7 weergegeven.

De dagcyclus bestaat uit handelingen, die eenmaal per dag moeten worden verricht. Dit is dus het verzenden van de informatie naar de centrale minicomputer.

Binnen deze dagcyclus bevindt zich een uurcyclus, die 24 maal per dag moet worden doorlopen. Dit is dus het in de tabel opslaan van de in dat uur bepaalde maximale, minimale en totale waarden.

En binnen de uurcyclus bevindt zich weer een minuutcyclus, die 60 maal per uur moet worden uitgevoerd. In de minuutcylcus wordt een nieuwe meetwaarde ingevoerd en verwerkt.

Fig. 7 lijkt veel op het algemeen stroomdiagram uit de les "Digitale klok". In feite is de datalogger een uitbreiding van de in die les ontwikkelde klok. Behalve het bijhouden van de tijd, moet er nu een aantal meetwaarden worden ingevoerd en verwerkt.

Het ligt voor de hand ook nu weer gebruik te maken van een urenteller UURTL en een minutenteller MINTL. Als we deze voorzieningen in fig.7 aanbrengen, dan ontstaat het stroomdiagram van fig.8.

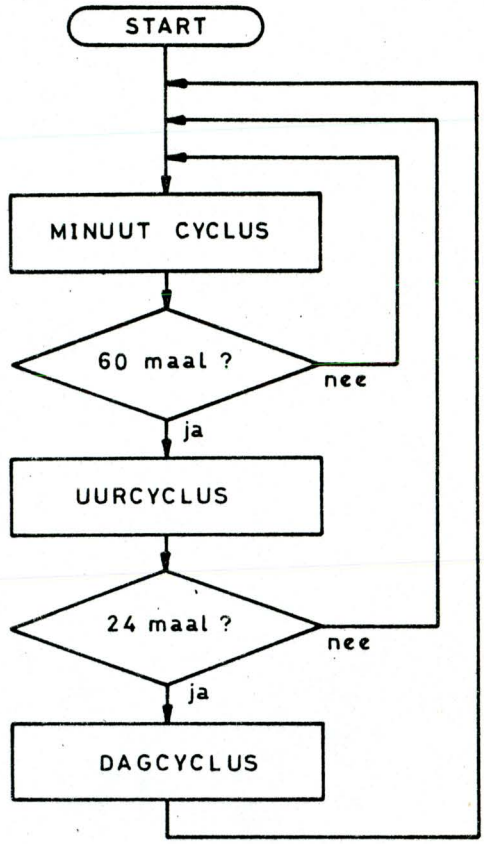


fig.7

In fig.8 is een blok t.b.v. de initialisatie van het microprocessorsysteem aangebracht. Een initialisatie is in vrijwel elk programma nodig, vooral wanneer we werken met computers, die één of meer programmeerbare chips bevatten.

Verder valt in fig.8 op, dat de uit te voeren instructies van de 3 genoemde cycli in subroutines zijn ondergebracht. Fig.8 is in feite het algemeen stroomdiagram voor het hoofdprogramma. We behoeven ons nu alleen nog bezig te houden met de drie afzonderlijke subroutines "MINCY", "UURCY" en "DAGCY". Dit wordt in de volgende paragrafen besproken.

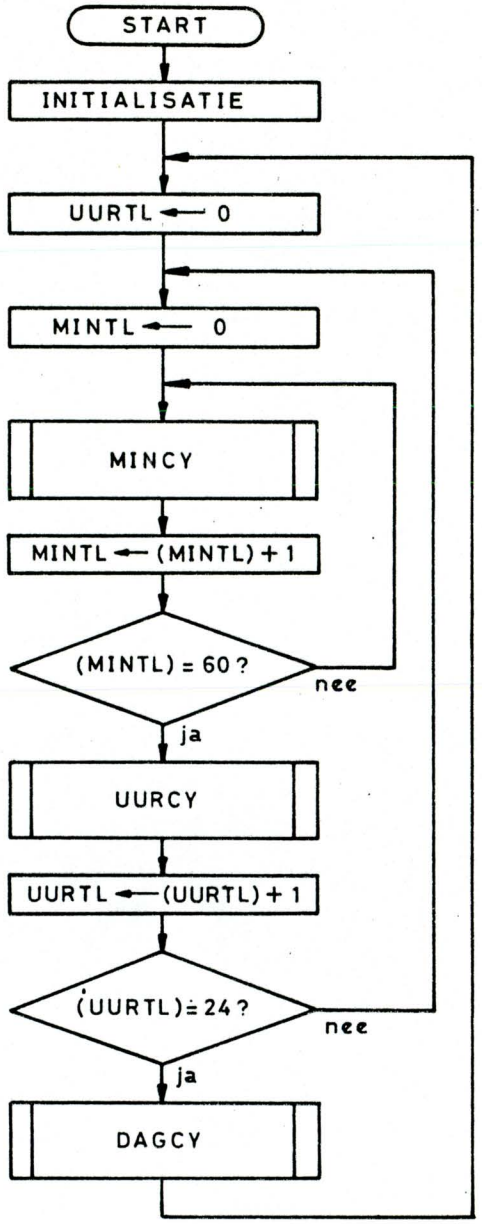


fig.8

5. SUBROUTINE "MINCY"

De subroutine "MINCY" heeft de volgende taken:

a. Elke minuut moet een nieuwe meetwaarde worden ingevoerd, d.w.z. dat de 8-bits binaire teller moet worden uitgelezen.
b. Direct na het uitlezen van de teller moet deze worden gereset.
c. Elke nieuwe meetwaarde moet worden vergeleken met die waarde, die tot dan als grootste waarde was ingevoerd. Als de nieuwe meetwaarde groter is, dan moet deze meetwaarde op de plaats van de grootste waarde worden opgeslagen. De inhoud van de minutenteller geeft dan aan in welke minuut deze maximale waarde was gemeten. Als de nieuwe meetwaarde niet groter is, dan blijft de tot dan gevonden maximale waarde ongewijzigd.
d. Idem als punt c, maar nu voor de minimale waarde.
e. Elke nieuwe meetwaarde moet worden opgeteld bij het totaal van alle voorgaande meetwaarden die in het betreffende uur al zijn ingevoerd.

Een belangrijk punt is de wijze, waarop steeds het begin van een minuut wordt aangegeven. M.a.w. hoe wordt het moment bepaald, waarop de teller moet worden uitgelezen? We maken natuurlijk weer gebruik van een timer (zie de les "Digitale klok"), die steeds na dezelfde tijd een interrupt request afgeeft. In de bijbehorende interrupt service routine moet dan een z.g. seinbit worden geset, als er een minuut is verstreken. Door subroutine "MINCY" dient deze seinbit dan continu te worden getest. Als er een 1 wordt gedetecteerd, dan moet de teller worden uitgelezen en de seinbit moet worden gereset, zodat een minuut later dezelfde actie opnieuw kan worden gestart.

Opmerking: Het toepassen van zo'n seinbit wordt vaak aangeduid met de term semaphore (= seinpaal).
De seinbit is te beschouwen als een overdracht van het ene programmadeel aan het andere (in ons geval van de timer interrupt service routine aan de subroutine).

Uit deze beschrijving kunnen we direct een stroomdiagram opstellen (fig.9).

Er blijft alleen het probleem op welke wijze de adressen voor MAX, MMAX, MIN, MMIN en TOTAAL worden bepaald. We hebben in fig.9 de volgnummers 0 t/m 23 achterwege gelaten. Deze zijn n.l. per uur verschillend.

Vraag 4: Het volgnummer is opgeslagen in .....

De inhoud van UURTL geeft steeds het betreffende uur, dus het volgnummer, aan. Als we nu het beginadres van de tabel (fig.6) weten, dan kunnen we m.b.v. de inhoud van UURTL elk gewenst adres binnen de tabel bepalen.

We geven het eerste adres binnen de tabel de symbolische naam TABEL.

Vraag 5: MAX0 staat dan op adres .....
MNI0 staat dan op adres .....
TOTAAL0 staat op adres .....

MAX0 bevindt zich op de eerste plaats in de tabel (fig.6), dus op adres TABEL. Voor MAX0 t/m MMAX23 zijn 48 geheugenwoorden gereserveerd. MIN0 staat dus op adres TABEL+48. TOTAAL0 staat dan op adres TABEL+96.

Vraag 6: MAX7 staat op adres .....
MIN8 staat op adres .....
TOTAAL15 staat op adres .....

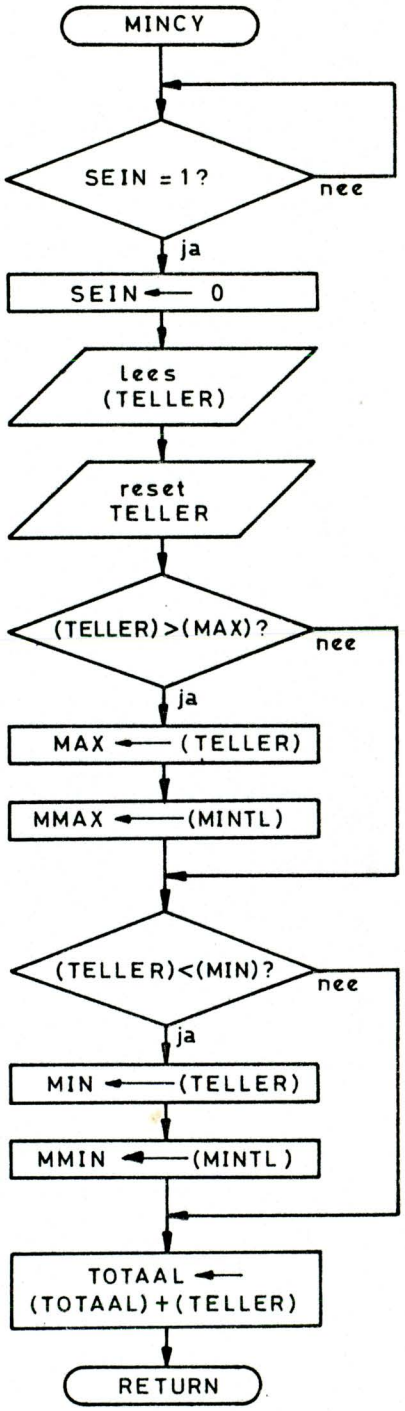


fig.9

Voor elke maximale waarde in de tabel zijn 2 geheugenwoorden gereserveerd, n.l. één voor MAX en één voor MMAX. Voor MAX0 t/m MMAX6 zijn dan 14 geheugenwoorden nodig. MAX7 is dus op adres TABEL+14 opgeslagen. Op dezelfde wijze kunnen we berekenen, dat MIN8 op adres TABEL+48+16= TABEL+64 staat. TOTAAL15 bevindt zich dan op adres TABEL+96+30= TABEL+126.

Antw.4: UURTL. Antw.5: TABEL; TABEL+48; TABEL+96.
Antw.6: TABEL+14; TABEL+64; TABEL+126.

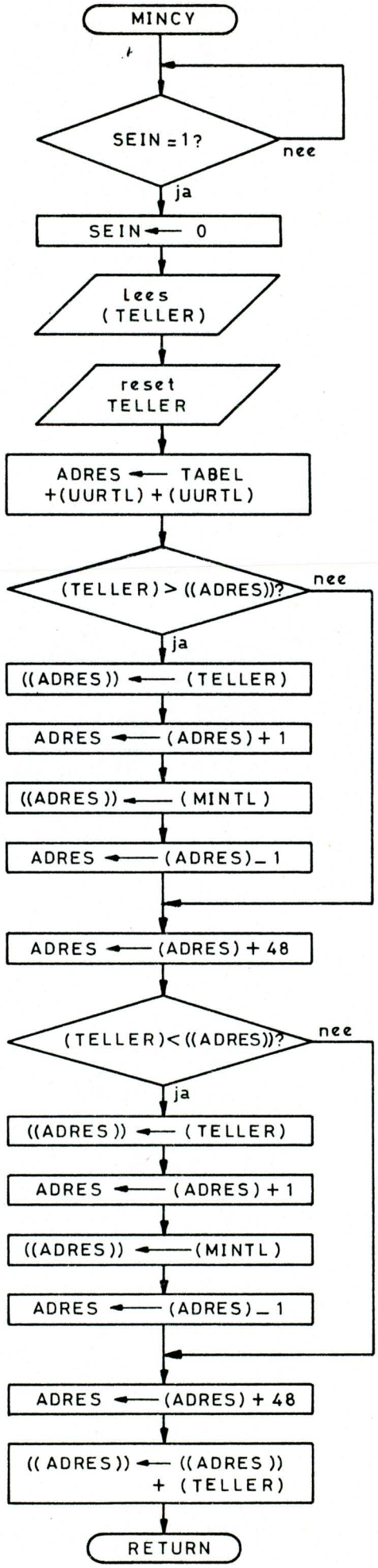

MINCY
SEIN = 1?
nee
ja
SEIN ← 0
lees
(TELLER)
reset
TELLER
ADRES ← TABEL
+(UURTL) + (UURTL)
(TELLER) > ((ADRES))?
nee
ja
((ADRES)) ← (TELLER)
ADRES ← (ADRES) + 1
((ADRES)) ← (MINTL)
ADRES ← (ADRES)_1
ADRES ← (ADRES) + 48
(TELLER) < ((ADRES))?
nee
ja
((ADRES)) ← (TELLER)
ADRES ← (ADRES) + 1
((ADRES)) ← (MINTL)
ADRES ← (ADRES)_1
ADRES ← (ADRES) + 48
((ADRES)) ← ((ADRES))
+ (TELLER)
RETURN

fig.10

Vraag 7: Om het adres van b.v. MAX13 te berekenen moet bij adres TABEL 1/2/4 maal de inhoud van UURTL worden opgeteld.

We kunnen nu eenvoudig het adres voor b.v. MAX13 berekenen, door bij TABEL 2 maal de inhoud van UURTL op te tellen. Immers gedurende de tijd dat MAX13 een rol speelt, dus van 13.00 - 13.59 uur, is de inhoud van de urenteller gelijk aan 13. Het adres van MAX13 is TABEL+13+13 = TABEL+26.

Vraag 8: Om het adres van TOTAAL13 te berekenen moet de inhoud van UURTL 2 maal worden opgeteld bij adres .....

Op dezelfde wijze kunnen we de adressen voor de minimale en de totaalwaarden berekenen. We tellen dan 2 maal de inhoud van UURTL op bij TABEL+48 resp. TABEL+96. Het adres van b.v. TOTAAL13 is dan TABEL+96+(UURTL)+(UURTL) = TABEL+96+26 = TABEL+122.

Als we deze berekeningen voor het bepalen van het juiste adres binnen de tabel in fig.9 aanbrengen dan ontstaat het gedetailleerd stroomdiagram van fig.10. Hierin is het gewenste adres steeds opgeslagen in een locatie met de symbolische naam ADRES. Ga voor uzelf na hoe de inhoud van ADRES in de subroutine steeds wordt aangepast, om MAX, MMAX, MIN, MMIN en TOTAAL op de juiste wijze te kunnen adresseren.

Antw.7: 2. Antw.8: TABEL+96.

6. SUBROUTINE "UURCY"

De taak van subroutine "UURCY" is het in de tabel opslaan van de in een bepaald uur gevonden maximale, minimale en totale waarden.

Vraag 9: Hiervoor moeten wel/niet instructies worden opgenomen.

Aangezien de tabel al rechtstreeks door subroutine "MINCY" wordt gevuld (m.b.v. de inhoud van ADRES), behoeven we in "UURCY" hiervoor geen instructies meer op te nemen. De inhoud van UURTL wordt in het hoofdprogramma (fig.8) steeds aangepast. M.a.w. de subroutine "UURCY" kan komen te vervallen.

7. SUBROUTINE "DAGCY"

Subroutine "DAGCY" dient om op 24.00 uur = 0.00 uur zo snel mogelijk alle in de tabel opgeslagen waarden uit te voeren. Daartoe moet achtereenvolgens plaatsvinden:

a. De INFO KLAAR-lijn moet actief worden gemaakt.
b. Alle waarden moeten direct na elkaar worden uitgevoerd.
c. Het actieve INFO KLAAR-signaal moet worden weggenomen.

Vraag 10: De tabel bestaat uit ..... bytes.

Uit deze beschrijving is direct het gedetailleerd stroomdiagram voor "DAGCY" op te stellen (fig.11). Hierin maken we weer gebruik van de adresaanwijzer ADRES, die achtereenvolgens de adressen TABEL t/m TABEL+144 bevat.

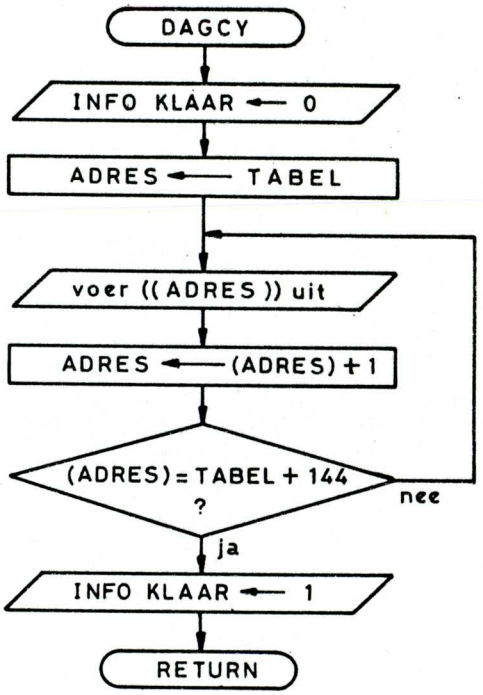


fig.11

Antw.9: niet. Antw.10: 144.

8. TIMER INTERRUPT SERVICE ROUTINE

De bedoeling van de interrupt service routine voor de timer is het setten van de seinbit. Dit moet exact na elke minuut gebeuren. Evenals in de les "Digitale klok" krijgen we te maken met het feit, dat de maximale tijd, gedurende welke de timer kan tellen, iets meer dan 2,5 ms is. We gaan ervan uit, dat we de timer voor exact 2,5 ms programmeren.

Vraag 11: Per minuut worden dan ..... interrupt requests van de timer ontvangen.

Per minuut geeft de timer dan 60 x 400 = 24000 interrupt requests af. Pas na de 24000e moet de seinbit worden geset. We moeten dus weer een teller voor het aantal ontvangen interrupt requests toepassen. Dit is in het gedetailleerd stroomdiagram (fig.12) gedaan in de vorm van IRTL.

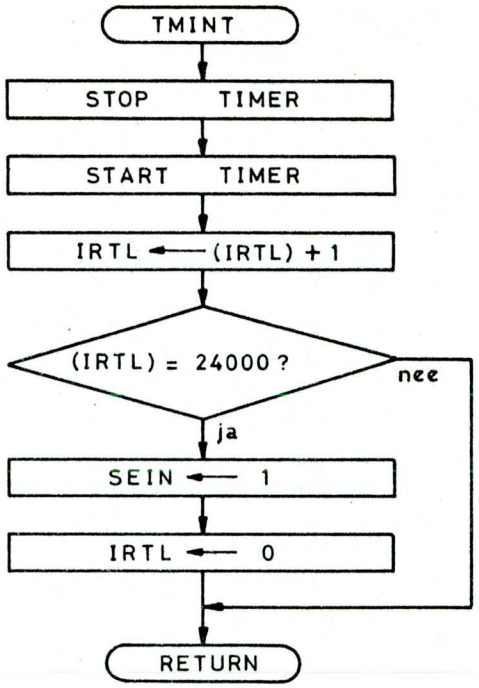


fig.12

In de les "Digitale klok" is al berekend met welke waarden de timer moet worden gevuld, om de tijd van 2,5 ms te realiseren. We nemen weer dezelfde waarden, n.l. TIHI = 79_{16} en TILO = $F4_{16}$.

Antw.11: 24000.

9. INITIALISATIES EN TOEWIJZINGEN

Voordat we de stroomdiagrammen voor het hoofdprogramma, "MINCY", "DAGCY" en "TMINT" in een programma kunnen omzetten zullen we nog een aantal punten moeten regelen n.l.
a. de initialisatie van het systeem,
b. het vastleggen van beginwaarden van variabelen,
c. het toewijzen van definitieve adressen aan de gebruikte symbolische namen.

We zullen deze punten uitvoeren, waarbij we weer uitgaan van de eigenschappen van de SDK 85.

a. Initialisatie van het systeem

Aan het begin van het programma moeten we in ieder geval de volgende zaken initialiseren:
de stackpointer
de I/O-module
de timer
de teller (resetten)
het INFO KLAAR-signaal (rusttoestand)
de interrupt controller (in de 8085)

Vraag 12: We hebben minimaal ..... input-poort(en) en
..... output-poort(en) nodig.

Voor het uitlezen van de 8-bits teller is een input-poort nodig. Deze noemen we INPP. Via output-poort OUPP1 wordt 8-bits data naar de centrale minicomputer verzonden. Bovendien zijn nog 2 lijnen van een output-poort (OUPP2) nodig voor de RESET- en INFO KLAAR-signalen.

b. Beginwaarden van variabelen

In het algemeen stroomdiagram voor subroutine "MINCY" bevinden zich enkele variabelen, die een gedefinieerde waarde moeten hebben, ook als "MINCY" voor de eerste keer in elk uur wordt aangeroepen. De minutenteller wordt dan wel met 0 gevuld (fig.8), maar er zijn nog enkele variabelen, die elk uur opnieuw moeten worden geïnitialiseerd.

Vraag 13: Dit zijn de variabelen
(.....), (.....) en (.....).

De in elk uur als eerste ingevoerde meetwaarde wordt vergeleken met (MAX) en (MIN) en bij (TOTAAL) opgeteld. We moeten dan wel zorgen, dat hier beginwaarden worden opgeslagen.

Vraag 14: Deze beginwaarden moeten dan zijn:
(MAX) = ...16.
(MIN) = ...16.
(TOTAAL) = ...16.

We moeten dan zorgen, dat de inhoud van MAX nul is, zodat de eerste meetwaarde direct als de tot dan gevonden maximale waarde wordt genomen.

Antw.12: 1; 2. Antw.13: MAX; MIN; TOTAAL.
Antw.14: 00; FF; 00.

Evenzo moet gelden dat (MIN) = FF_{16}. De eerste meetwaarden van dat uur wordt dan tevens als minimale waarde opgeslagen.
TOTAAL moet dan worden gevuld met 0, zodat wordt voorkomen, dat de eerste meetwaarde bij een van de vorige dagcylcus afkomstig resultaat wordt opgeteld.
Omdat "MINCY" deze variabelen direct vanuit de tabel ophaalt, moeten we de beginwaarden ook in deze tabel opslaan. Dit doen we weer m.b.v. de inhoud van de adresaanwijzer ADRES, die we per uur uit de inhoud van UURTL moeten berekenen.
Het stroomdiagram voor het hoofdprogramma (fig.8) gaat dan over in dat van fig.13. Merk op, dat overeenkomstig paragraaf 6 de subroutine "UURTL" is weggelaten.

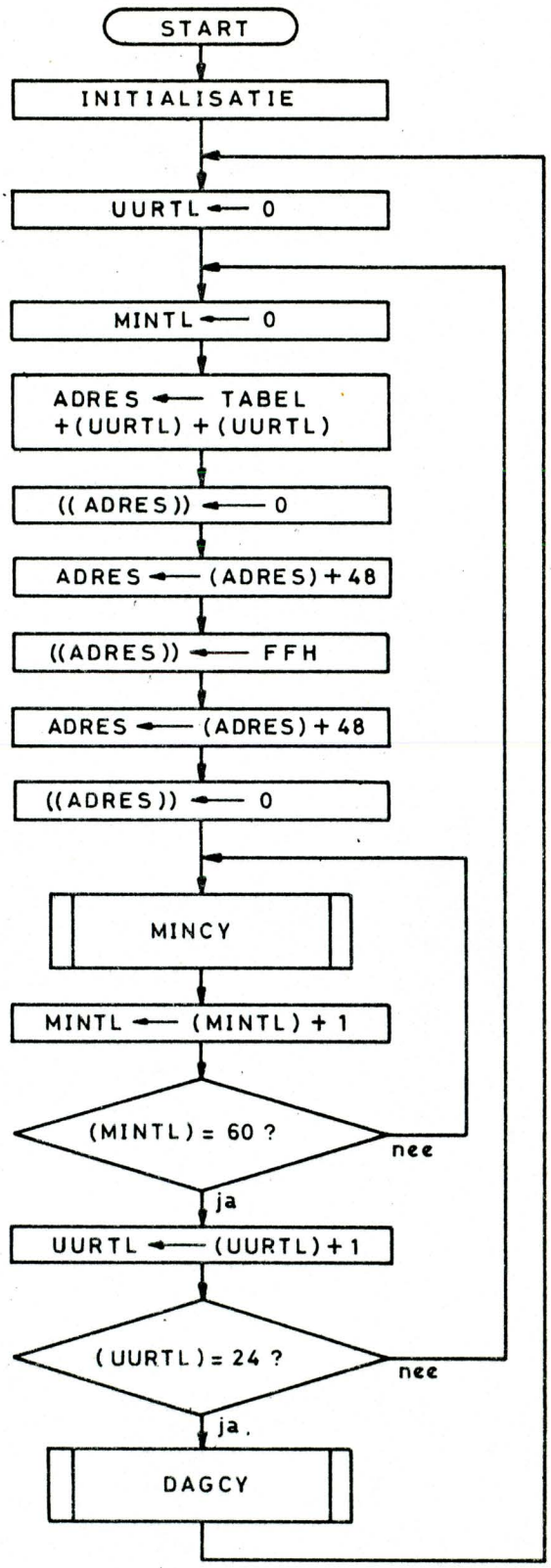


fig.13

c. Symbolische namen

In de stroomdiagrammen van fig.10, fig.11, fig. 12 en fig.13 (voor het gemak zijn deze nog eens op een uitvouwblad bij elkaar geplaatst) komen een aantal symbolische namen voor, waarbij we nog definitieve registeren/of geheugenadressen moeten vaststellen. We zouden al deze locaties graag in de CPU-registers onderbrengen. Er zijn n.l. veel meer instructies voor het bewerken en transporteren van registerinhouden. Hiervoor komen echter teveel symbolische namen voor. Bovendien zijn de CPU-registers waarschijnlijk alle betrokken bij de bewerkingen met 16-bits operanden. Denk b.v. aan het vergelijken van (IRTL) met 24000_{10} en het steeds berekenen van (ADRES).

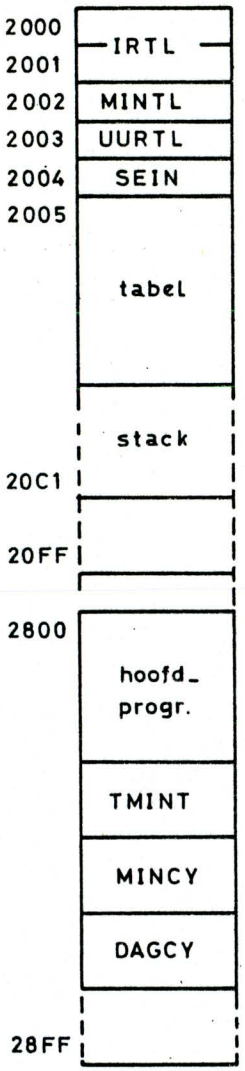


fig.14

Alleen voor de adresaanwijzer ADRES reserveren we registerpaar H,L. Dit is gedaan omdat (ADRES) relatief vaak moet worden gebruikt en gewijzigd. Voor de overige symbolische namen reserveren we geheugenadressen. We kiezen b.v. een memory-map volgens fig.14.

Behalve deze symbolische namen, moeten we nog enkele zaken vastleggen. We hebben voor SEIN een heel geheugenwoord gereserveerd. Voor de seinbit is echter maar 1 bit vereist. Hiervoor nemen we b.v. b_7. De I/O-poorten kiezen we als volgt:

INPP = poort 00_{16}
OUPP1 = poort 21_{16}
OUPP2 = poort 22_{16}: b_0 = RESET
b_1 = INFO KLAAR

We kunnen nu fig.10, fig.11, fig.12 en fig.13 omzetten in een programma.

SCHRIJF NU ZELF EEN PROGRAMMA EN TEST DIT OP DE SDK 85.
BESTUDEER DAARNA ONZE OPLOSSING IN FIG.15.

10. PROGRAMMA EN TESTEN

Fig.15 is het door ons ontwikkelde programma. Let hierin vooral op

a. het gebruik van de waarde IREND. Deze waarde is in een EQU-directive geplaatst, zodat de assembler voor ons de omzetting van 24000_{10} in een hexadecimaal getal ($5CD0_{16}$) uitvoert.
b. het gebruik van de symbolische naam TBEND. In de bijbehorende EQU-directive wordt hieraan de waarde TABEL+144 toegekend. Tijdens de uitvoering van deze EQU-directive moet de assembler dus een optelling uitvoeren. Veel assemblers zijn hiertoe in staat. Een voorwaarde is dan wel, dat op het moment, dat deze berekening moet worden uitgevoerd, de op te tellen getallen aan de assembler bekend zijn. In dit geval telt de assembler de waarden 2005_{16} en $144_{10} = 90_{16}$ bij elkaar op. Het resultaat 2095_{16} wordt aan TBEND toegekend.
c. het feit dat de inhoud van MINTL in BCD-code is opgeslagen. Dit is n.l. één van de eisen uit de probleemomschrijving.
d. de wijze waarop in de instructies op de adressen $282D_{16}$ t/m 2839_{16} en $287A_{16}$ t/m 2886_{16} rekening wordt gehouden met het optreden van een eventuele carry.
e. de wijze waarop door de instructies op de adressen $28EF_{16}$ t/m $28F8_{16}$ twee 16-bits getallen met elkaar worden vergeleken.

Waarschijnlijk zal dit programma afwijken van uw oplossing. U zult uw programma moeten testen. Met de schakelaars van input-poort 00_{16} kunt u meetwaarden aanbieden. Op de LED's van de output-poorten kunt u de uitgevoerde data en de signalen RESET en INFO KLAAR controleren. Er treden nu echter twee problemen op:

a. Om een volledige dagcyclus te kunnen testen, zult u 24 uur lang meetwaarden op poort 00_{16} moeten aanbieden, voordat de inhoud van de tabel wordt uitgevoerd. Om deze tijd te bekorten kunt u de instructies CPI 60H (adres 2850_{16}) en CPI 24D (adres 2858_{16}) b.v. wijzigen in b.v. CPI 04H resp. CPI 03D. U maakt dan als het ware een nieuwe tijdrekening: 1 dag = 3 uren; 1 uur = 4 minuten. De test duurt dan nog slechts 12 minuten. Denk er aan, dat u tevens de waarde van TBEND en de 2 instructies LXI B,0048D aanpast.

b. Aan het eind van een dagcyclus wordt de inhoud van de tabel met een zo hoge snelheid uitgevoerd, dat u deze niet op de LED's van poort 21_{16} kunt aflezen. Oplossingen voor dit probleem zijn het aanbrengen van breakpoints in de subroutine "DAGCY" of het vervangen van "DAGCY" door een subroutine, die de opeenvolgende tabelinhouden in een laag tempo na elkaar op de 7-segment displays weergeeft.

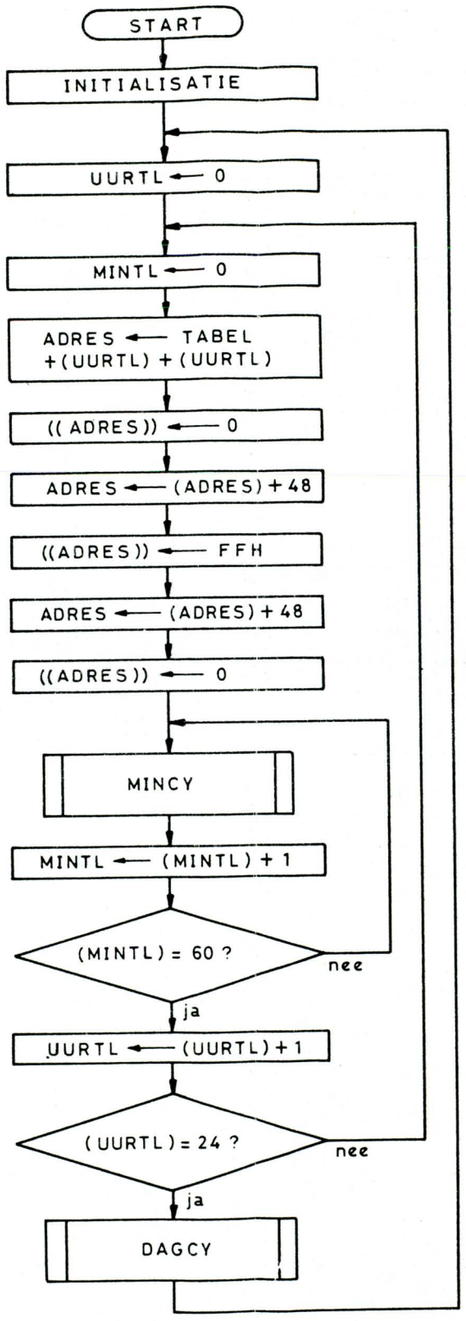


fig.13

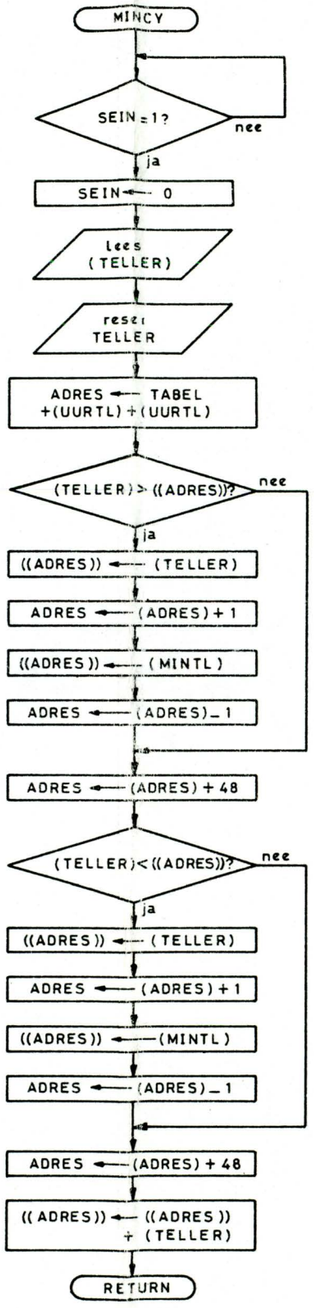


fig.10

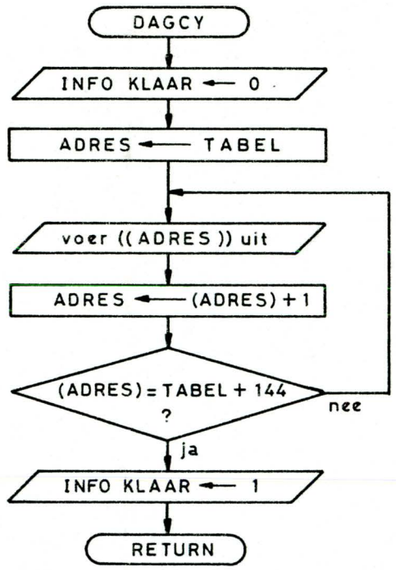


fig.11

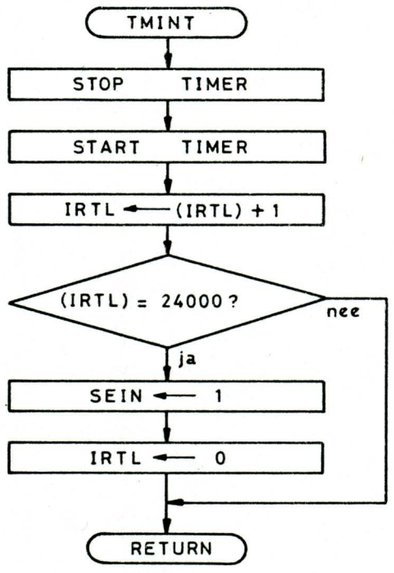


fig.12

```
                        ORG  2800H
                IRTL    EQU  2000H     ;INT.REQUEST TELLER
                IREND   EQU  24000D    ;EINDWAARDE IRTL
                MINTL   EQU  2002H     ;MINUTENTELLER
                UURTL   EQU  2003H     ;URENTELLER
                SEIN    EQU  2004H     ;ADRES SEINBIT
                SEIN0   EQU  00H       ;SEINBIT=0
                SEIN1   EQU  80H       ;SEINBIT=1
                TABEL   EQU  2005H     ;BEGINADRES TABEL
                TBEND   EQU  TABEL+144D ; EINDWAARDE TABEL
                START   EQU  C0H       ;START TIMER COMMAND
                STOP    EQU  40H       ;STOP TIMER COMMAND
                TIHI    EQU  79H       ;CLR HIGH ORDER BYTE
                TILO    EQU  F4H       ;CLR LOW ORDER BYTE
                EN6.5   EQU  1DH       ;ENABLE RST6.5
                INPP    EQU  00H       ;INPUT TELLER
                OUPP1   EQU  21H       ;OUTPUT MINICOMPUTER
                OUPP2   EQU  22H       ;OUTPUT RESET+INFO KLAAR
                RESET   EQU  02H       ;RESET TELLER ACTIEF
                INFKL   EQU  01H       ;INFO KLAAR ACTIEF
                RUST    EQU  03H       ;RUSTTOESTAND VOOR OUPP2
2800  3E 03             MVI  A,03H     ;INITIALISATIE
2802  D3 20             OUT  20H       ;POORT 21H=OUTPUT
2804  32 FF 20          STA  20FFH     ;POORT 22H=OUTPUT
2807  AF                XRA  A
2808  D3 02             OUT  02H       ;POORT 00H=INPUT
280A  3E 1D             MVI  A,EN6.5   ;INIT
280C  30                SIM            ;  INTERRUPT
280D  FB                EI             ;    CONTROLLER
280E  3E 79             MVI  A,TIHI    ;INIT
2810  D3 2D             OUT  2DH       ;  COUNT
2812  3E F4             MVI  A,TILO    ;    LENGTH
2814  D3 2C             OUT  2CH       ;      REGISTER
2816  3E C0             MVI  A,START   ;START
2818  D3 28             OUT  28H       ;  TIMER
281A  3E 02             MVI  A,RESET   ;INIT
281C  D3 22             OUT  OUPP2     ;  TELLER
281E  3E 03             MVI  A,RUST    ;    EN INFO
2820  D3 22             OUT  OUPP2     ;      KLAAR
2822  31 C2 20          LXI  SP,20C2H  ;INIT STACKPOINTER
2825  AF          CYDAG: XRA  A
2826  32 03 20          STA  UURTL     ;INIT URENTELLER
2829  AF          CYUUR: XRA  A
282A  32 02 20          STA  MINTL     ;INIT MINUTENTELLER
282D  21 05 20          LXI  H,TABEL   ;  ********************
2830  3A 03 20          LDA  UURTL     ;  *
2833  87                ADD  A         ;  *  BEREKEN ADRES
2834  85                ADD  L         ;  *  (H,L)=TABEL+
2835  6F                MOV  L,A       ;  *  (UURTL)+(UURTL)
2836  D2 3A 28          JNC  INMAX     ;  *
2839  24                INR  H         ;  ********************
283A  36 00       INMAX: MVI  M,00H     ;INIT MAXIMALE WAARDE
283C  01 30 00          LXI  B,0048D
283F  09                DAD  B
2840  36 FF             MVI  M,FFH     ;INIT MINIMALE WAARDE
2842  09                DAD  B
2843  36 00             MVI  M,00H     ;INIT TOTAALWAARDE
2845  CD 63 28  CYMIN:  CALL MINCY     ;VERWERK 1 MEETWAARDE
2848  21 02 20          LXI  H,MINTL   ;  ********************
284B  7E                MOV  A,M       ;  *
284C  C6 01             ADI  01H       ;  *  VERHOOG (MINTL)
284E  27                DAA            ;  *
284F  77                MOV  M,A       ;  ********************
2850  FE 60             CPI  60H       ;60 MEETWAARDEN VERWERKT?
2852  C2 45 28          JNZ  CYMIN     ;NEE: VOLGENDE MEETWAARDE
2855  23                INX  H         ;  ********************
2856  34                INR  M         ;  *  VERHOOG (UURTL)
2857  7E                MOV  A,M       ;  ********************
2858  FE 18             CPI  24D       ;24 UURCYCLI DOORLOPEN?
285A  C2 29 28          JNZ  CYUUR     ;NEE: VOLGENDE UURCYCLUS
285D  CD E1 28          CALL DAGCY     ;JA: INFO NAAR MINICOMPUTER
2860  C3 25 28          JMP  CYDAG     ;VOLGENDE DAGCYCLUS
```

fig.15

```
2863  3A 04 20  MINCY:  LDA  SEIN      ;HAAL SEINBIT OP
2866  07                RLC            ;SEINBIT=1?
2867  D2 63 28          JNC  MINCY     ;NEE: TEST OPNIEUW
286A  3E 00             MVI  A,SEIN0   ;JA: MAAK
286C  32 04 20          STA  SEIN      ;    SEINBIT 0
286F  DB 00             IN   INPP      ;LEES TELLER UIT
2871  5F                MOV  E,A       ;BERG (TELLER) OP
2872  3E 02             MVI  A,RESET   ;  ********************
2874  D3 22             OUT  OUPP2     ;  *
2876  3E 03             MVI  A,RUST    ;  *  RESET TELLER
2878  D3 22             OUT  OUPP2     ;  *
287A  21 05 20          LXI  H,TABEL   ;  ********************
287D  3A 03 20          LDA  UURTL     ;  *
2880  87                ADD  A         ;  *  BEREKEN ADRES:
2881  85                ADD  L         ;  *  (H,L)=TABEL+
2882  6F                MOV  L,A       ;  *  (UURTL)+(UURTL)
2883  D2 87 28          JNC  CPMAX     ;  *
2886  24                INR  H         ;  ********************
2887  7E          CPMAX: MOV  A,M       ;IS (TELLER) NIEUWE
2888  BB                CMP  E         ;  MAXIMALE WAARDE?
2889  D2 93 28          JNC  CPMIN     ;NEE: VERGELIJK MINIMUM
288C  73                MOV  M,E       ;JA: (TELLER) NAAR TABEL
288D  23                INX  H         ;BIJBEHORENDE
288E  3A 02 20          LDA  MINTL     ;  MINUUT
2891  77                MOV  M,A       ;    NAAR
2892  2B                DCX  H         ;      TABEL
2893  01 30 00  CPMIN:  LXI  B,0048D   ;BEREKEN ADRES VAN
2896  09                DAD  B         ;  MINIMALE WAARDE
2897  7E                MOV  A,M       ;IS (TELLER) NIEUWE
2898  BB                CMP  E         ;  MINIMALE WAARDE?
2899  DA A6 28          JC   TOTAL     ;NEE: GA (TELLER)
289C  C2 A6 28          JZ   TOTAL     ;      OPTELLEN
289F  73                MOV  M,E       ;JA: (TELLER) NAAR TABEL
28A0  23                INX  H         ;BIJBEHORENDE
28A1  3A 02 20          LDA  MINTL     ;  MINUUT
28A4  77                MOV  M,A       ;    NAAR
28A5  2B                DCX  H         ;      TABEL
28A6  09          TOTAL: DAD  B         ;  *****************
28A7  7E                MOV  A,M       ;  *  TEL (TELLER)
28A8  83                ADD  E         ;  *  OP BIJ HET
28A9  77                MOV  M,A       ;  *  TOTAAL VAN DE
28AA  D0                RNC            ;  *  VOORGAANDE
28AB  23                INX  H         ;  *  MEETWAARDEN
28AC  34                INR  M         ;  *****************
28AD  C9                RET
28AE  F5          TMINT: PUSH PSW
28AF  E5                PUSH H
28B0  3E 40             MVI  A,STOP    ;STOP
28B2  D3 28             OUT  28H       ;  TIMER
28B4  3E C0             MVI  A,START   ;START
28B6  D3 28             OUT  28H       ;  TIMER
28B8  21 00 20          LXI  H,IRTL    ;  **********
28BB  34                INR  M         ;  *
28BC  C2 C2 28          JNZ  CPIRT     ;  *  VERHOOG
28BF  23                INX  H         ;  *    (IRTL)
28C0  34                INR  M         ;  *
28C1  2B                DCX  H         ;  **********
28C2  C5          CPIRT: PUSH B
28C3  01 C0 5D          LXI  B,IREND   ;VERGELIJK LOW ORDER
28C6  7E                MOV  A,M       ;  BYTES VAN IRTL
28C7  B9                CMP  C         ;    EN EINDWAARDE
28C8  C2 DC 28          JNZ  TERUG     ;SPRING BIJ ONGELIJK
28CB  23                INX  H         ;VERGELIJK HIGH ORDER
28CC  7E                MOV  A,M       ;  BYTES VAN IRTL
28CD  B8                CMP  B         ;    EN EINDWAARDE
28CE  C2 DC 28          JNZ  TERUG     ;SPRING BIJ ONGELIJK
28D1  3E 80             MVI  A,SEIN1   ;24000 INT.REQUESTS
28D3  32 04 20          STA  SEIN      ;  ONTVANGEN:
28D6  21 00 00          LXI  H,0000H   ;    SET SEINBIT EN
28D9  22 00 20          SHLD IRTL      ;      VUL IRTL MET 0
28DC  C1          TERUG: POP  B
28DD  E1                POP  H
28DE  F1                POP  PSW
28DF  FB                EI
28E0  C9                RET
28E1  3E 01       DAGCY: MVI  A,INFKL   ;MAAK INFO
28E3  D3 22             OUT  OUPP2     ;  KLAAR ACTIEF
28E5  21 05 20          LXI  H,TABEL   ;(H,L)=BEGINADRES TABEL
28E8  01 95 20          LXI  B,TBEND   ;(B,C)=EINDWAARDE TABEL
28EB  7E          NEXT: MOV  A,M       ;VOER 1
28EC  D3 21             OUT  OUPP1     ;  BYTE UIT
28EE  23                INX  H         ;VERHOOG ADRES
28EF  79                MOV  A,C       ;VERGELIJK LOW
28F0  BD                CMP  L         ;  ORDER BYTES
28F1  C2 EB 28          JNZ  NEXT      ;SPRING BIJ ONGELIJK
28F4  78                MOV  A,B       ;VERGELIJK HIGH
28F5  BC                CMP  H         ;  ORDER BYTES
28F6  C2 EB 28          JNZ  NEXT      ;SPRING BIJ ONGELIJK
28F9  3E 03             MVI  A,RUST    ;ALLE 144 BYTES UITGEVOERD:
28FB  D3 22             OUT  OUPP2     ;  MAAK INFO KLAAR 1
28FD  C9                RET
                        ORG  20C8H     ;(20CEH)
20C8  C3 AE 28          JMP  TMINT
                        END
```

A/D-CONVERTER.

A/D-converter

1. INLEIDING

In enkele van de voorgaande lessen zijn toepassingen van microcomputers in procesbesturingen beschreven. Elke applicatie betrof echter digitale procesbesturing. D.w.z. dat zowel de input als de output data door combinaties van énen en nullen werd gevormd.

In veel processen worden echter analoge signalen afgegeven.
Onder een analoog signaal verstaan we een signaal, dat tussen een gegeven maximum en een gegeven minimum iedere willekeurige waarde kan hebben.

Via het bussysteem van de microcomputer kan echter alleen digitale informatie worden getransporteerd. Voordat we een analoog signaal via een input-poort kunnen invoeren, zullen we dit signaal eerst in een digitale bitcombinatie moeten omzetten. Hiertoe is een z.g. A/D-converter (= analoog-naar-digitaal-omzetter) vereist. Deze A/D-converter moet dan tussen de analoge uitgang van het proces en de input-poort van de microcomputer worden aangebracht. De A/D-converter is dus te beschouwen als een interface tussen het analoge proces en de digitale microcomputer (fig.1).

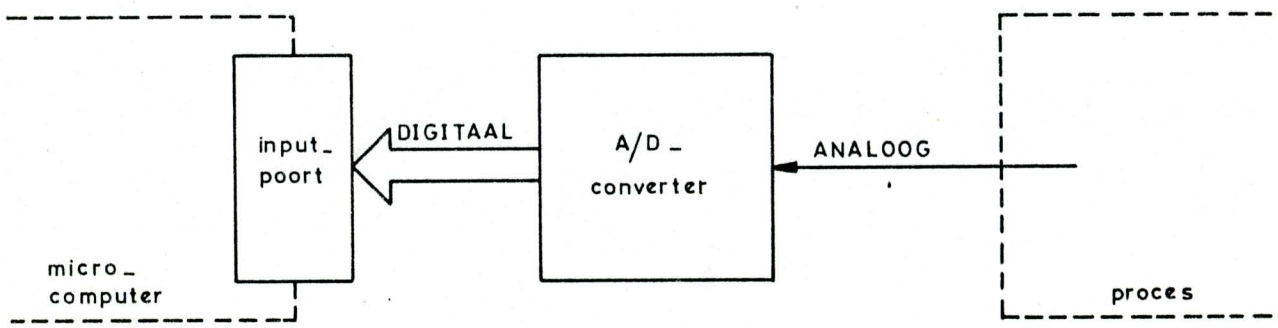


fig.1

De A/D-converter in fig.1 is een zelfstandig opererend blok hardware met eigen besturingslogica. Vooral wanneer er een grote nauwkeurigheid is vereist, is er een complexe en dus dure schakeling noodzakelijk.
Het ligt daarom voor de hand, dat in microprocessorsystemen, waarin de CPU niet continu bezig behoeft te zijn met de uitvoering van het feitelijke programma, een deel van de A/D-conversie software-matig kan worden gerealiseerd. In dit soort gevallen kan de hardware van de A/D-omzetter eenvoudig blijven. Hiervan wordt in deze les een voorbeeld beschreven.

2. DE OMZETTING VAN ANALOOG NAAR DIGITAAL

De meest elementaire vorm van een A/D-omzetter is de z.g. schmitt-trigger of drempeldetector (fig.2a). Het gedrag van deze schakeling is in fig.2b weergegeven.

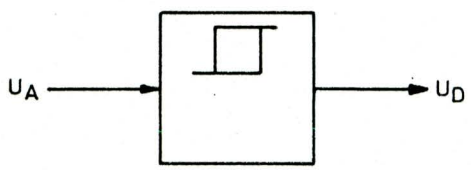


fig. 2a

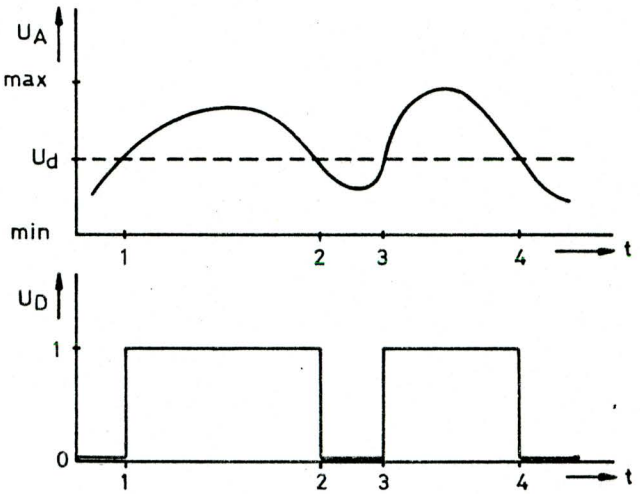


fig.2b

U_A is een analoge spanning, die elke willekeurige waarde tussen een minimum en een maximum kan hebben. Als U_A lager is dan de door de interne opbouw van de schmitt-trigger bepaalde drempelspanning U_d, dan is de uitgang 0. Als U_A groter is dan U_d, dan is de uitgang 1.
De uitgangsspanning U_D van de schmitt-trigger is dus een (binair) digitaal signaal. De schmitt-trigger is een 1-bits A/D-converter.

Vraag 1: Met het uitgangssignaal U_D kunnen we ..... verschillende getallen weergeven.

Op de uitgang van de schmitt-trigger kunnen maar 2 toestanden optreden. Met het uitgangssignaal U_D kunnen we dus ook slechts 2 getallen weergeven, nl. 0 en 1. In het binaire talstelsel kunnen we door gebruik te maken van een aantal bits, meer getallen voorstellen.
Evenzo is het mogelijk een aantal schmitt-triggers te combineren, zodat op de gezamenlijke uitgangen meer getallen kunnen worden weergegeven. Fig.3a is hiervan een voorbeeld.

Vraag 2: De A/D-converter in fig.3a kan .... verschillende getallen weergeven.

De A/D-converter in fig.3a bestaat uit twee schmitt-triggers, waarvan de drempelspanningen 1 V resp. 2 V bedragen. In fig.3b is weergegeven welke bitcombinaties op de uitgangen D0 en D1 kunnen optreden. We hebben het gedrag van de schakeling tevens in tabel 1 beschreven.

U_A	D1	D0
<1 V	0	0
1 V - 2 V	0	1
>2 V	1	1

Tabel 1

Antw.1: 2. Antw.3: 3.

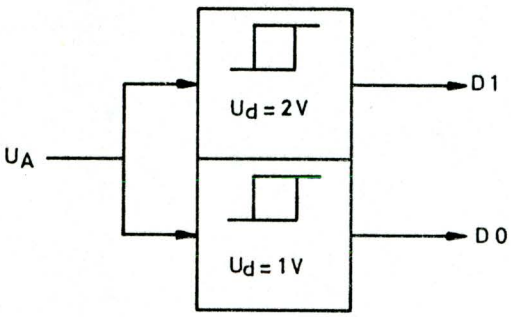


fig.3a

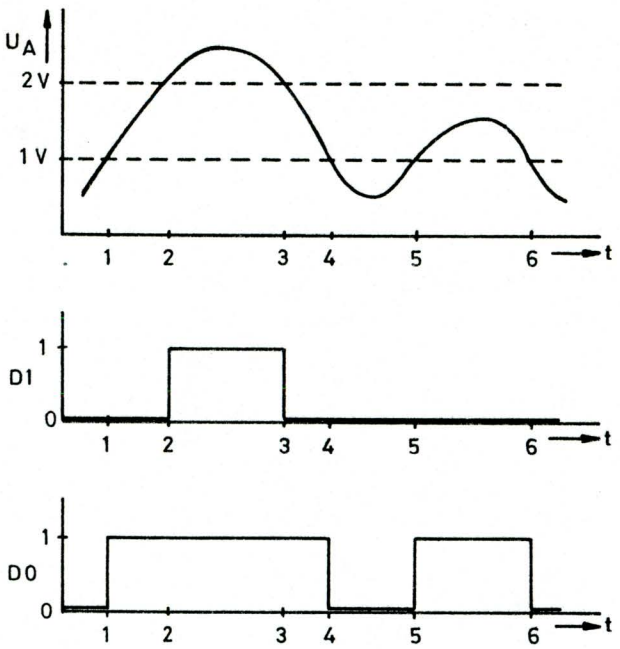


fig.3b

Uit deze tabel blijkt dat er 3 mogelijke bitcombinaties afgegeven kunnen worden. Er kunnen dus 3 verschillende getallen worden voorgesteld.

Vraag 3: Een A/D-converter, bestaande uit 4 schmitt-triggers, kan maximaal ..... verschillende bitcombinaties afgeven.

In fig.4 is een A/D-converter met 4 schmitt-triggers weergegeven. Uit de bijbehorende tabel 2 blijkt, dat er 5 verschillende getallen kunnen worden voorgesteld.

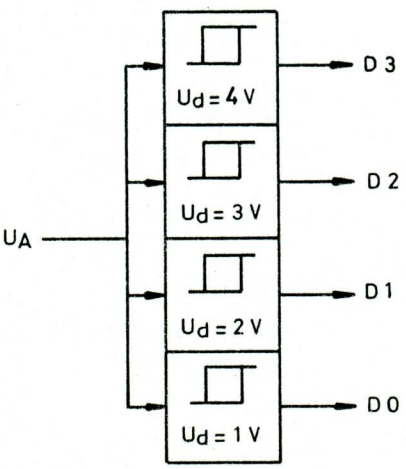


fig.4

U_A	D3	D2	D1	D0
<1 V	0	0	0	0
1 V - 2 V	0	0	0	1
2 V - 3 V	0	0	1	1
3 V - 4 V	0	1	1	1
>4 V	1	1	1	1

Tabel 2

Antw.3: 5.

U_A (V)	INPP $b_7 \ldots b_0$	OUPP $b_3 \ldots b_0$	getalswaarde
<0,5	00000000	0000	0
0,5 - 1	00000001	0001	1
1 - 1,5	00000011	0010	2
1,5 - 2	00000111	0011	3
2 - 2,5	00001111	0100	4
2,5 - 3	00011111	0101	5
3 - 3,5	00111111	0110	6
3,5 - 4	01111111	0111	7
>4	11111111	1000	8

Tabel 3

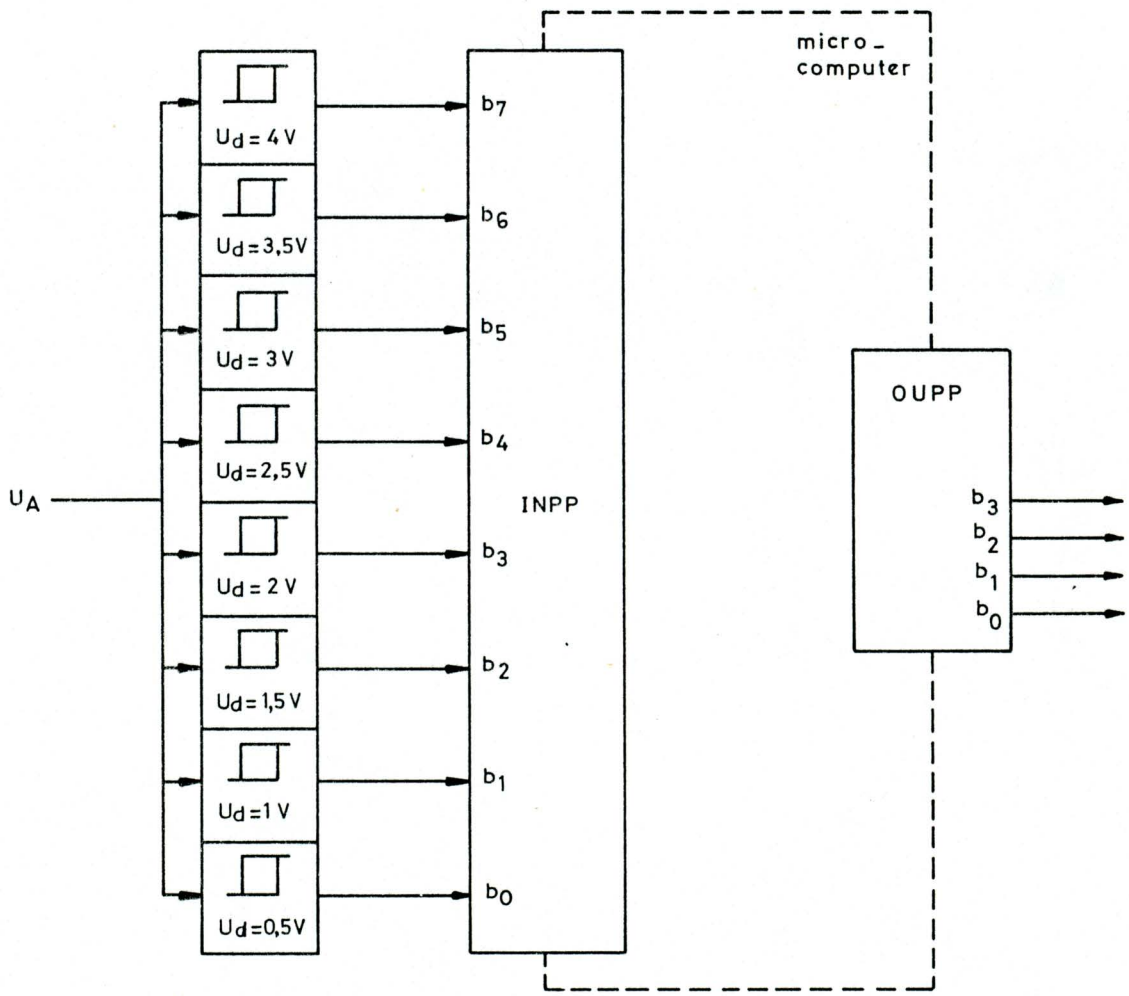


fig.5

In deze les gaan we uit van een A/D-converter, die 8 schmitt-triggers bevat (fig.5). Hiermee zijn dus 9 verschillende getallen voor te stellen. Tabel 3 geeft het verband tussen in- en uitgangssignalen van de A/D-converter aan. Bovendien is vermeld welke code de microcomputer hieruit moet vormen.
Fig.6 geeft een voorbeeld van een mogelijke analoge ingangsspanning en de bijbehorende binaire waarden, die de microcomputer in dit geval af moet geven op de output-poort.
Op de tijdstippen t=1 t/m t=10 worden de op de input-poorten aangeboden waarden ingelezen. Deze tijdstippen worden vaak bemonsteringstijden genoemd.

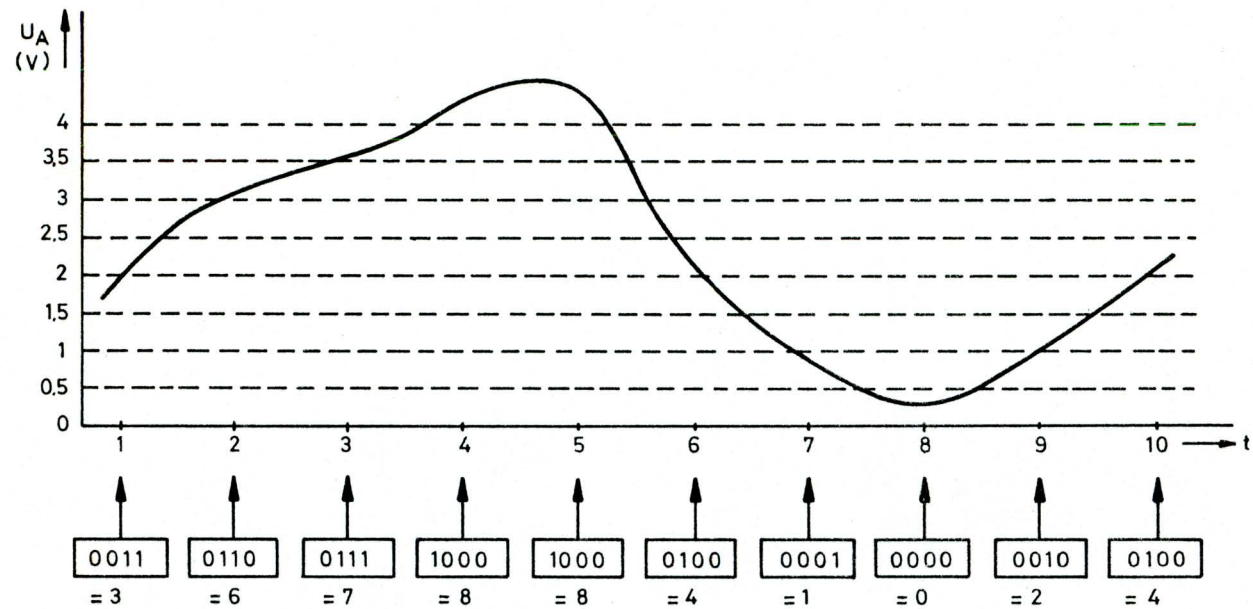


fig.6

In fig.7a is weergegeven welke decimale waarden dan via de output-poort worden uitgevoerd. In fig.7b is dit zelfde gedaan, maar nu voor het geval, dat de bemonsteringstijden dichter bij elkaar liggen.

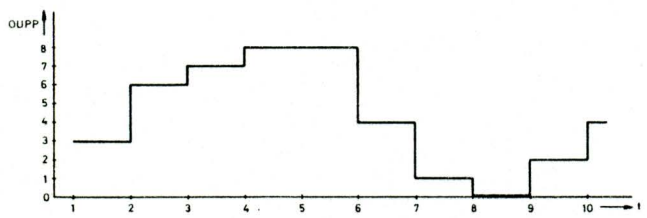


fig.7a

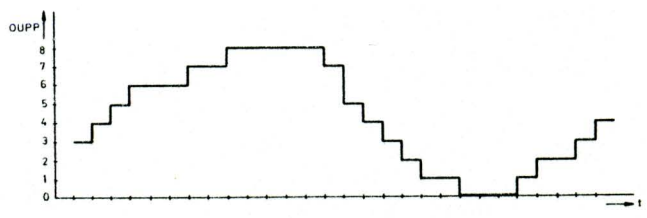


fig.7b

Uit fig.7a en fig.7b blijkt, dat des te kleiner de tijd tussen de bemonsteringstijden is, des te nauwkeuriger de decimale waarde op de output-poort zal zijn. Een afwijking tussen het analoge signaal en de uit-

gevoerde decimale waarde blijft er echter altijd bestaan. Dit komt, doordat de A/D-converter de analoge spanning steeds "afrondt" op een lager niveau, nl. de hoogste drempelspanning, die nog kleiner is dan de analoge spanning. B.v. 2,9 V wordt afgerond tot 2,5 V. Als we deze afwijking willen verkleinen, dan moeten we in de A/D-converter meer schmitt-triggers opnemen.
In fig.5 kunnen we b.v. extra schmitt-triggers met drempelspanningen van 0,25 V, 0,75 V, 1,25 V, enz. aanbrengen. De A/D-converter krijgt dan ook meer uitgangen, zodat de microcomputer over meer input-lijnen moet beschikken.

3. PROBLEEMOMSCHRIJVING

Gevraagd wordt een programma te ontwikkelen, waardoor het systeem van fig.5 de in tabel 3 beschreven uitgangscombinaties afgeeft.
De taak van dit programma is dus het omzetten van de uit de schmitt-triggers afkomstige bitcombinaties in binaire waarden.

De bemonsteringstijden moeten 1 ms uit elkaar liggen. b_4 t/m b_7 van de output-poort moeten 0 zijn.

SCHRIJF EERST ZELF EEN PROGRAMMA EN TEST DIT OP UW SDK 85.
BESTUDEER DAARNA ONZE OPLOSSING.

4. PROBLEEMANALYSE

Door het te ontwikkelen programma moet eenmaal per milliseconde:
a. de op de input-poort aangeboden bitcombinatie worden ingelezen;
b. deze bitcombinatie in de juiste binaire code worden omgezet;
c. deze code via de output-poort worden uitgevoerd.

Dit is direct in een algemeen stroomdiagram weergegeven (fig.8).
We maken weer gebruik van een semaphore.
Een hardware timer set elke miliseconde een seinbit. Deze seinbit wordt continu getest, totdat een 1 aangeeft, dat er nieuwe data moet worden ingelezen.
In fig.8 is het programmadeel, waarin de feitelijke code-omzetting plaatsvindt, als subroutine uitgevoerd. Dit is gedaan, omdat er, voor wat betreft de toe te passen algorithme een aantal mogelijkheden zijn.

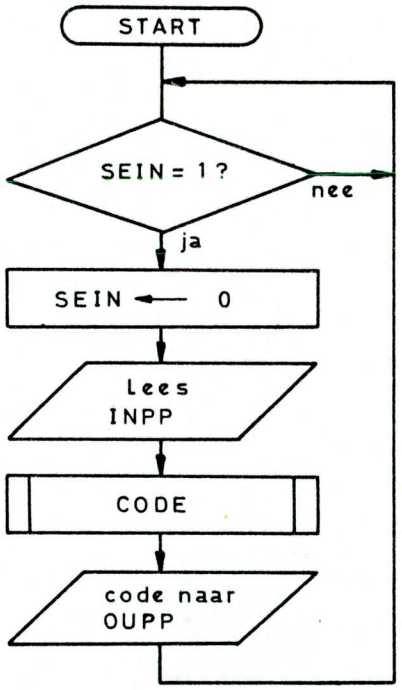


fig.8

Mogelijke algorithmen voor "CODE" zijn:
a. het tellen van de énen in de ontvangen bitcombinatie. Als dit aantal énen binair wordt weergegeven, ontstaat direct de gewenste output-code.
b. het gebruik van een tabel.
   Uit de ontvangen bitcombinatie wordt rechtstreeks een adres afgeleid. Op dit adres bevindt zich de gewenste output-code.
c. de z.g. trial and error-methode.
   We vullen een register met de waarde 00001111_2 en vergelijken dit met de aangeboden bitcombinatie. Zijn beide niet gelijk, dan gaan we systematisch énen in het register bijplaatsen of er uit halen, totdat de inhoud van het register en de aangeboden bitcombinatie gelijk zijn.

In de volgende paragrafen gaan we elk van deze algorithmen omzetten in een subroutine "CODE". De eisen, die aan elke oplossing worden gesteld zijn:
a. de via INPP ingevoerde bitcombinatie bevindt zich in de accumulator.
b. de via OUPP uit te voeren binaire code moet in de accumulator worden geplaatst.
c. de inhouden van de overige CPU-registers moeten bij het terugkeren naar het hoofdprogramma gelijk zijn aan die bij het aanroepen van "CODE".

ONTWIKKEL NU EERST ZELF DE DRIE GEVRAAGDE SUBROUTINES. BESTUDEER DAARNA ONZE OPLOSSINGEN.

5. HET TELLEN VAN DE ÉNEN

Het tellen van het aantal énen in de ontvangen bitcombinatie kan plaatsvinden volgens de in fig.9 weergegeven methode.
De inhoud van de accumulator wordt steeds geroteerd, waarna de carry status flag wordt getest. Als deze 1 is, wordt de inhoud van een teller met 1 verhoogd.
Wanneer dit 8 maal is gebeurd, bevat de teller het aantal énen, dat in de ontvangen bitcombinatie aanwezig was.

Vraag 4: In fig.9 mag alleen linksom/alleen rechtsom/zowel linksom als rechtsom worden geroteerd.

In fig.9 is het niet belangrijk of er linksom of rechtsom wordt geroteerd. Elk van de 8 bits van de ontvangen bitcombinatie wordt immers een keer in de carry status flag geplaatst.

In fig.9 is een extra teller nodig, om te kunnen testen of er al 8 maal is geroteerd. Deze extra teller kunnen we achterwege laten, als we de methode van fig.10 toepassen. Hierin wordt de inhoud van de accumulator steeds rechtsom geroteerd, totdat een 0 in de carry status flag wordt gedetecteerd. Dan zijn alle énen geteld. De énen staan immers naast elkaar en zoveel mogelijk aan de "rechterkant" (zie tabel 3).

In de methode van fig.10 kan echter een fout optreden. Stel, dat de via INPP ingevoerde bitcombinatie uit 8 énen bestaat.

Vraag 5: Er wordt dan wel/niet uit de subroutine "CODE" teruggekeerd naar het hoofdprogramma.

Er zal dan na elke rotatie een 1 in de carry status flag worden gedetecteerd. Na het testen van deze status flag zal dus steeds in de richting "ja" verder worden gegaan. De subroutine wordt dan nooit meer verlaten.

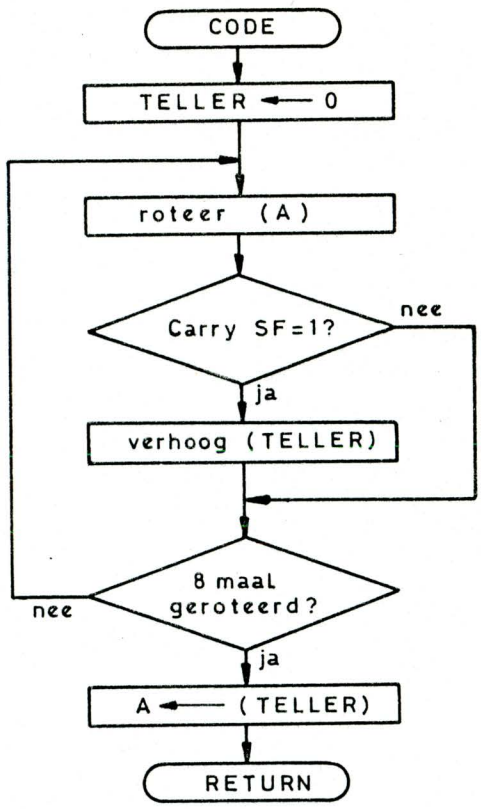


fig.9

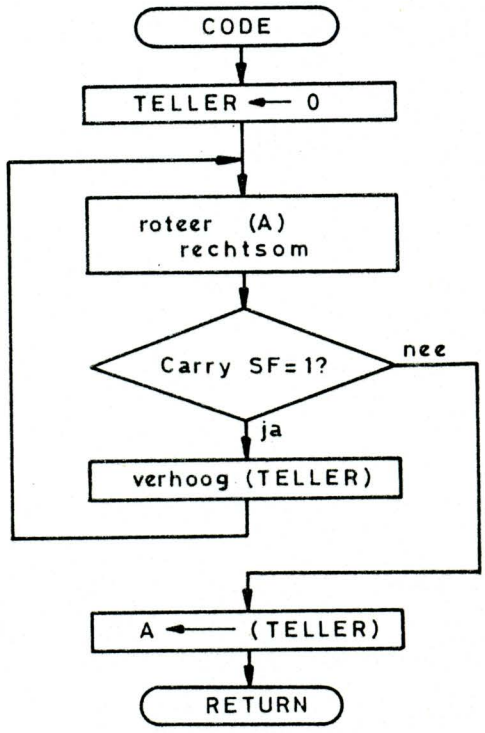


fig.10

Antw.4: zowel linksom als rechtsom. Antw.5: niet.

Een oplossing hiervoor is, dat niet alleen de accumulatorinhoud, maar tevens de carry status flag in de rotatiekring wordt opgenomen, dus het realiseren van een 9-bits rotatie. Een voorwaarde is dan wel, dat aan het begin van de subroutine "CODE" de carry status flag is gereset. Dan ontstaat het stroomdiagram van fig. 11.

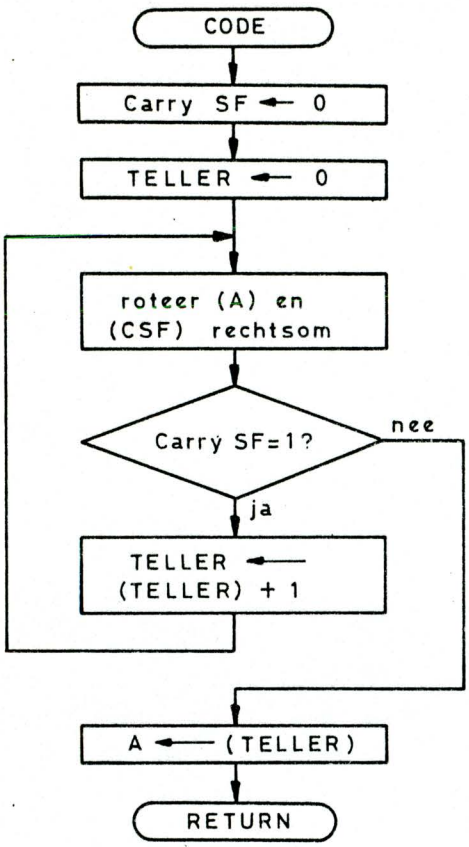


fig.11

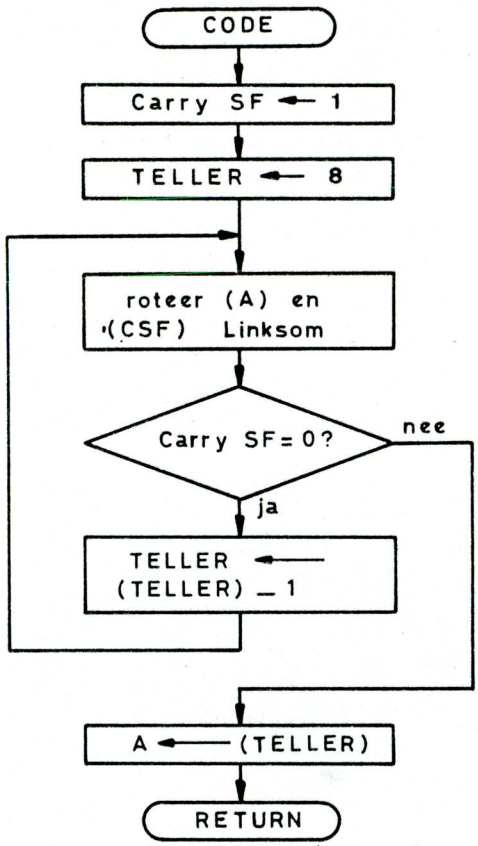


fig.12

Een variant op deze oplossing is die in fig.12. Hierin wordt niet het aantal énen, maar het aantal nullen geteld.
Ga deze methode voor uzelf na.

6. HET TOEPASSEN VAN EEN TABEL

Bij deze methode staan alle mogelijke uit te voeren binaire waarden in een tabel opgeslagen.
Uit de ingevoerde bitcombinatie wordt een adres afgeleid, dat aangeeft op welke plaats in de tabel zich de uit te voeren waarde bevindt.

In fig.13 is hiervoor een oplossingsmethode weergegeven.
De ingevoerde bitcombinatie vormt de low order byte van het adres in de tabel.

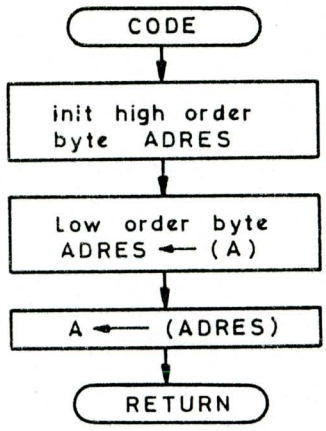


fig.13

Vraag 6: De benodigde tabel beslaat ..... bytes.
Hiervan worden er slechts ..... gebruikt.

Aangezien de minimale en maximale waarde van de low order adresbyte resp. 00_{16} en FF_{16} zijn, moeten voor de tabel 256 geheugenwoorden worden gereserveerd. Hiervan worden er echter maar 9 daadwerkelijk gebruikt, om er de binaire waarden voor 0 t/m 8 in op te slaan. Deze methode heeft dus als nadeel, dat er bijzonder veel geheugenruimte nodig is. Het grote voordeel is het feit, dat de verwerkingstijd veel korter is dan de oplossing uit de vorige paragraaf. (Vergelijk fig.13 met fig.11 en fig.12.)

Antw.6: 256; 9.

7. DE TRIAL AND ERROR-METHODE

De trial and error-methode (trial = poging) komt er op neer, dat de microcomputer systematisch alle mogelijke bitcombinaties genereert en deze steeds vergelijkt met de ontvangen bitcombinatie.

In fig.14 is hiervoor een oplossingsmethode weergegeven. De ingevoerde bitcombinatie wordt vanuit de accumulator naar een register overgebracht. Dit is gedaan, omdat de accumulator waarschijnlijk nodig is bij het vormen van de mogelijke bitcombinaties. Deze worden met de inhoud van het bovengenoemde register vergeleken, totdat ze gelijk zijn. Als tijdens het doorlopen van de programmalus de inhoud van de teller op de juiste wijze wordt verhoogd, dan zal aan het eind van de subroutine de gewenste binaire waarde zich in deze teller bevinden.

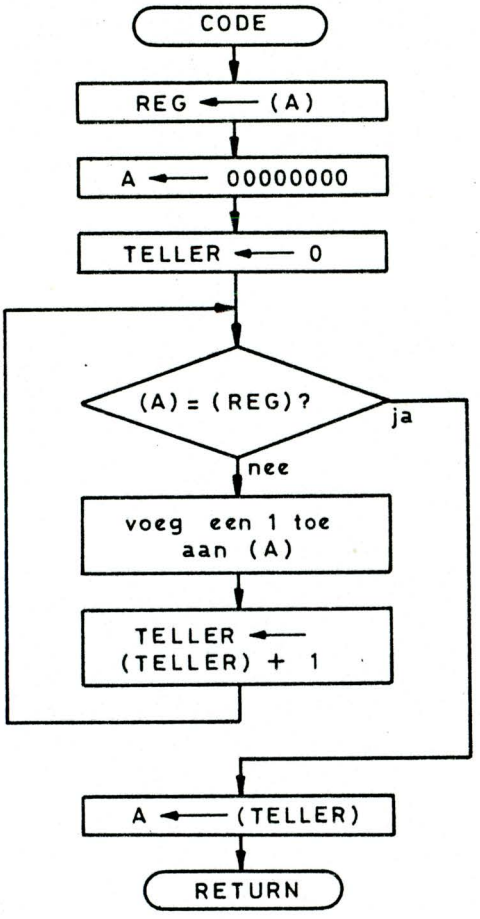


fig.14

Vraag 7: De programmalus in fig.14 wordt maximaal ..... maal doorlopen.

Als de ingevoerde bitcombinatie 11111111_2 is, dan moet er 8 maal een 1 aan de accumulator-inhoud worden toegevoegd, voordat de eindwaarde is bereikt. De programmalus moet dan 8 maal worden uitgevoerd.

We kunnen de maximale uitvoeringstijd verkorten, door niet te beginnen met de combinatie 00000000_2 maar met 00001111_2. Uitgaande van deze combinatie wordt dan de inhoud van de accumulator met 1 verhoogd of met 1 verlaagd, totdat de accumulatorinhoud en de ontvangen bitcombinatie gelijk zijn. Fig.15 is hiervoor het stroomdiagram.

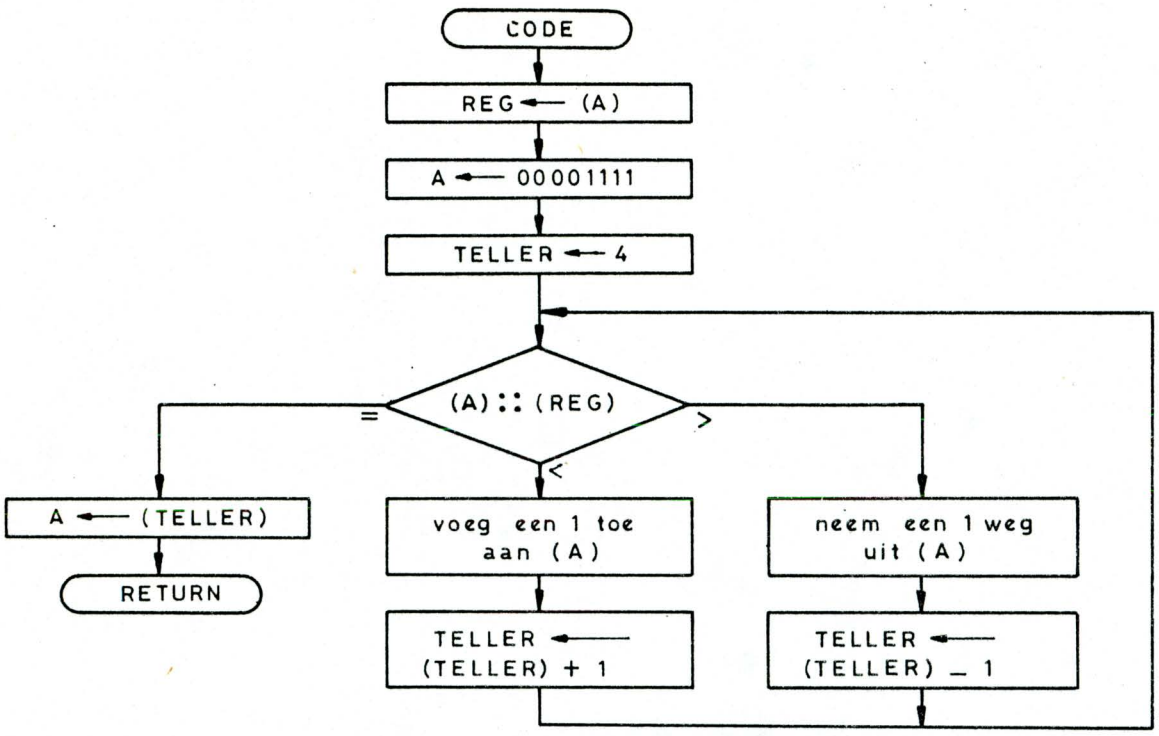


fig.15

Opmerking:

De vier punten in het testblok in fig.15 geven aan, dat het een vergelijking van 2 getallen betreft.

Vraag 8: In fig.15 wordt de programmalus maximaal .... maal doorlopen.

Om de accumulatorinhoud van 00001111_2 in 11111111_2 óf in 00000000_2 te veranderen, behoeft nu maximaal 4 maal een 1 in de accumulator te worden toegevoegd of weggenomen. Dit geeft t.o.v. de vorige oplossing een relatief grote tijdwinst.

8. PROGRAMMA

Uit het stroomdiagram van fig.8 is rechtstreeks een hoofdprogramma te schrijven.
De interrupt service routine voor de hardware timer is zeer eenvoudig. De timer in de SDK 85 kan nl. de tijd van 1 ms d.m.v. één maal starten realiseren.
In de interrupt service routine behoeft dan alleen de timer gestopt en weer gestart te worden.
Tevens moet de seinbit worden geset (fig.16).

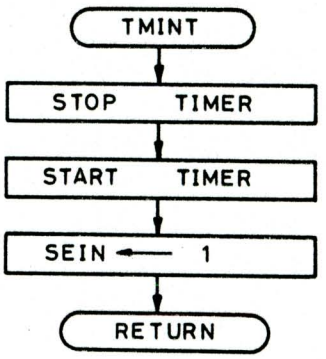


fig.16

Voor de subroutine "CODE" kunnen we één van de oplossingen uit fig.9 t/m fig.15 kiezen.

Voor wat betreft de symbolische namen, maken we, uitgaande van de SDK 85, de volgende keuze. (U mag natuurlijk hiervan weer afwijken.)

INPP = input-poort 00_{16}
OUPP = output-poort 21_{16}
SEIN = b_0 van geheugenwoord $20C1_{16}$
REG = register B
TELLER = register C
ADRES = opgeslagen in registerpaar H,L

We kunnen nu het programma schrijven. Ons resultaat is dat in fig.17. Hierin is
fig.17a: het hoofdprogramma en de interrupt service routine.
fig.17b: subroutine "CODE" volgens fig.9.
fig.17c: subroutine "CODE" volgens fig.11.
fig.17d: subroutine "CODE" volgens fig.12.
fig.17e: subroutine "CODE" volgens fig.13.
fig.17f: subroutine "CODE" volgens fig.14.
fig.17g: subroutine "CODE" volgens fig.15.

Om een compleet programma te verkrijgen, moet u één van de subroutines direct achter fig.17a plaatsen. Welke subroutine u kiest, is afhankelijk van de gestelde eisen.

U kunt het programma nu eenvoudig testen, door bitcombinaties op poort 00_{16} aan te bieden

9. CONCLUSIES EN OPMERKINGEN

De in deze les beschreven A/D-converter bestaat uit zowel hardware als software. De snelheid van het totale systeem hangt voornamelijk af van de software. In ons geval kunnen we b.v. door het kiezen van de juiste subroutine "CODE" bereiken, dat niet 1 maal maar b.v. 10 maal per milliseconde een analoge spanning in een binaire waarde wordt omgezet.
In de MPC-opgaven wordt dieper op het verschil in snelheid van de beschreven subroutines ingegaan.

Het is echter niet zo, dat alleen de snelheid bepalend is voor de keuze van een oplossingsmethode. B.v. van de subroutine in fig.17e is de uitvoeringstijd veel korter dan die van de overige subroutines. Om deze snelheid te bereiken, moeten we 256 bytes voor een tabel reserveren. Ook de benodigde geheugenruimte kan dus een rol spelen bij de keuze. Als deze 256 bytes niet beschikbaar zijn, dan is de oplossing van fig. 17e niet te gebruiken.

Een ander aspect van de in deze les behandelde A/D-converter is de nauwkeurigheid. Aan het eind van paragraaf 2 is al beschreven, dat we de nauwkeurigheid kunnen verbeteren door meer schmitt-triggers toe te passen. Als we b.v. een analoge spanning in een 8-bits binaire waarde willen uitdrukken, dan zijn er 255 schmitt-triggers nodig.
Behalve het feit, dat er dan ook 255 input-lijnen vereist zijn, is er nog een bezwaar. Het steeds inlezen van 8 input-poorten en het werken met 255-bits getallen kost bijzonder veel verwerkingstijd. Het systeem wordt dan zeer traag. (Vooral als er meer analoge spanningen moeten worden verwerkt.)

In dit soort gevallen past men meestal de hardware-configuratie van fig.18 toe.

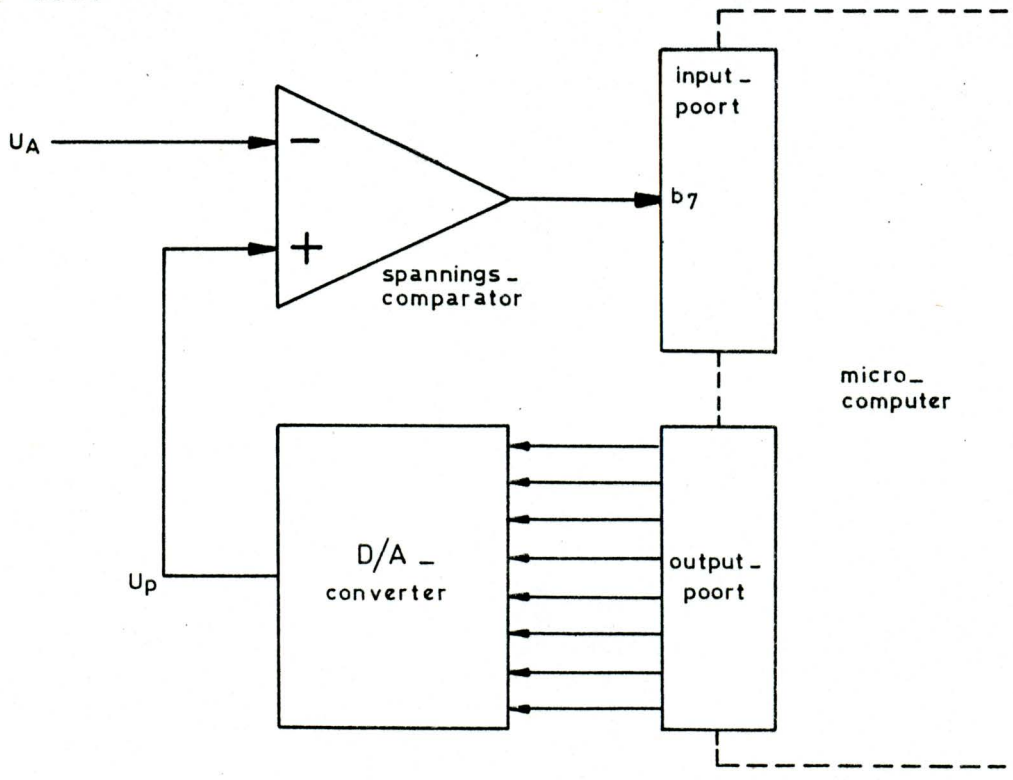


fig.18

Hierin wordt een D/A-converter (digitaal-naar-analoog-omzetter) toegepast. Deze D/A-converter zet de van de output-poort ontvangen 8-bits binaire waarden om in een analoge spanning U_p.
Deze spanning wordt vergeleken met de analoge ingangsspanning U_A.
Dit vergelijken gebeurt d.m.v. een spanningscomparator. In dit geval is hiervoor een OpAmp (= operationele versterker) gebruikt. De comparator gedraagt zich als volgt:

$U_p < U_A$: uitgang is 0.
$U_p > U_A$: uitgang is 1.
$U_p = U_A$: uitgang is 1.

Fig.19 geeft globaal weer, hoe het bijbehorende programma kan zijn opgebouwd.
Via de 8 bits van de output-poort wordt de inhoud van een binaire (software) teller uitgevoerd.
Deze teller wordt steeds met 1 verhoogd, totdat de spanningscomparator een 1 afgeeft.
Op dat moment is U_p net iets hoger geworden dan U_A.
In de teller staat dan de bijbehorende binaire waarde.

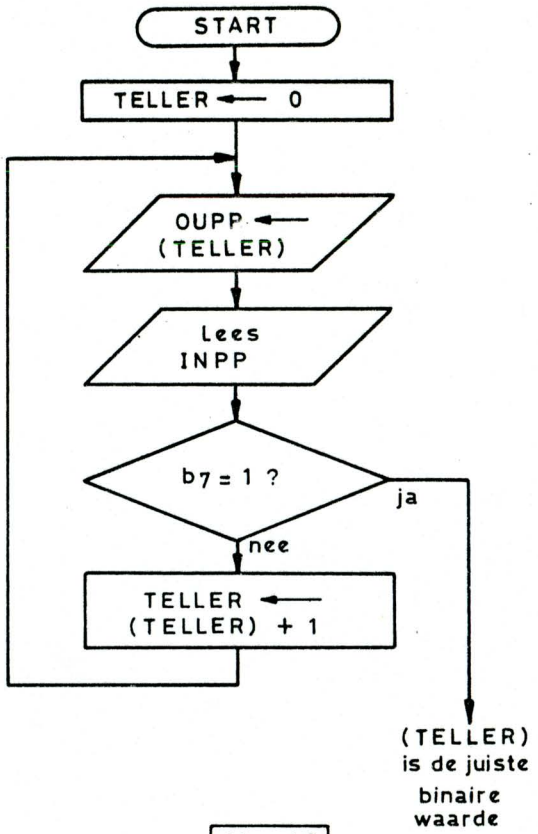


fig.19

De minimale tellerstand is 00000000_2, de maximale 11111111_2.
Het kan dus voorkomen, dat de programmalus in fig.19 256 maal moet worden doorlopen. (Dit gebeurt als U_A de maximale waarde heeft.)
Dit kan bijzonder veel tijd in beslag nemen. Een veel snellere oplossing is het gebruik van de trial and error-methode van fig.20.

We zullen dit stroomdiagram toelichten aan de hand van een voorbeeld.
In tabel 4 is het verband tussen de analoge spanningen en de bijbehorende binaire waarden weergegeven.

U_A en U_p (V)	binaire waarde	dec. waarde
0	00000000	0
0,01	00000001	1
0,02	00000010	2
0,03	00000011	3
⋮	⋮	⋮
1,36	10001000	136
1,37	10001001	137
1,38	10001010	138
⋮	⋮	⋮
2,54	11111110	254
2,55	11111111	255

Tabel 4

Stel, dat de spanning U_A 1,37 V is. Als het programma wordt gestart, worden een register REG en een teller beide met 128_{10} gevuld. (Voor het gemak geven we de inhouden decimaal weer. In de microcomputer zijn ze natuurlijk binair opgeslagen.)

Dan wordt de programmalus voor de eerste maal uitgevoerd. De inhoud van REG wordt door 2 gedeeld. (REG) is nu 64_{10}.
Dan wordt (TELLER) = 128_{10} uitgevoerd. Hierdoor wordt U_p 1,28 V.
Aangezien U_p kleiner is dan U_A (deze was immers 1,37 V), geldt $b_7 = 0$. De inhoud van de teller wordt dus met 64_{10} verhoogd. (TELLER) wordt nu 192_{10}.

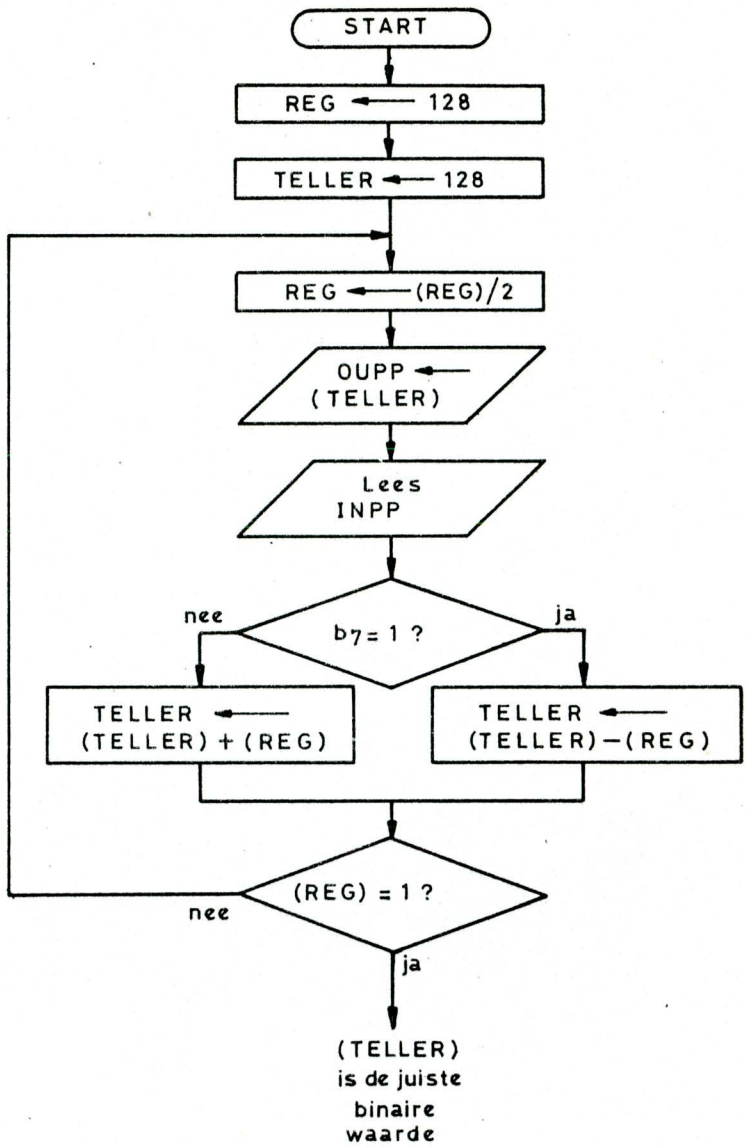


fig.20

De lus wordt nu voor de tweede maal doorlopen. (REG) wordt gehalveerd tot 32_{10}. Door het uitvoeren van (TELLER) wordt U_p 1,92 V, dus groter dan U_A. b_7 wordt dan 1 en van de tellerinhoud wordt 32_{10} afgetrokken. (TELLER) wordt nu 160_{10} en de lus zal voor de derde keer worden doorlopen.

Deze gang van zaken is schematisch in tabel 5 weergegeven.
De programmalus wordt 7 maal uitgevoerd. Dan wordt gedetecteerd dat (REG) = 1 en de programmalus wordt verlaten. In de teller staat dan de juiste waarde $137_{10} = 10001001_2$.

```
                        ORG   2000H
                INPP    EQU   00H       ;INPUT BITCOMBINATIE
                OUPP    EQU   21H       ;OUTPUT BINAIRE WAARDE
                SEIN    EQU   20C1H     ;ADRES SEINBIT
                SEIN0   EQU   00H       ;SEINBIT = 0
                SEIN1   EQU   01H       ;SEINBIT = 1
                TIHI    EQU   4BH       ;CLR HIGH ORDER BYTE
                TILO    EQU   72H       ;CLR LOW ORDER BYTE
                START   EQU   C0H       ;START TIMER COMMAND
                STOP    EQU   40H       ;STOP TIMER COMMAND
                EN6.5   EQU   1DH       ;ENABLE RST6.5
2000  3E 03             MVI   A,03H     ;INITIALISATIE
2002  D3 20             OUT   20H
2004  32 FF 20          STA   20FFH     ;POORT 21H = OUTPUT
2007  AF               XRA   A
2008  D3 02             OUT   02H       ;POORT 00H = INPUT
200A  31 C1 20          LXI   SP,20C1H  ;INIT STACKPOINTER -
200D  3E 1D             MVI   A,EN6.5   ;INIT
200F  30                SIM             ;  INTERRUPT
2010  FB                EI              ;    CONTROLLER
2011  3E 4B             MVI   A,TIHI    ;INIT
2013  D3 2D             OUT   2DH       ;  COUNT
2015  3E 72             MVI   A,TILO    ;    LENGTH
2017  D3 2C             OUT   2CH       ;      REGISTER
2019  3E 00             MVI   A,SEIN0   ;INIT
201B  32 C1 20          STA   SEIN      ;  SEINBIT
201E  3E C0             MVI   A,START   ;START
2020  D3 28             OUT   28H       ;  TIMER
2022  3A C1 20  TEST:    LDA   SEIN      ;TEST
2025  0F                RRC             ;  SEINBIT
2026  D2 22 20          JNC   TEST      ;SPRING ALS SEINBIT ≠ 1
2029  3E 00             MVI   A,SEIN0   ;RESET
202B  32 C1 20          STA   SEIN      ;  SEINBIT
202E  DB 00             IN    INPP      ;LEES BITCOMBINATIE
2030  CD 60 20          CALL  CODE      ;BEPAAL BINAIRE WAARDE
2033  D3 21             OUT   OUPP      ;VOER BINAIRE WAARDE UIT
2035  C3 22 20          JMP   TEST
2038  F5        TMINT:  PUSH  PSW
2039  3E 40             MVI   A,STOP    ;STOP
203B  D3 28             OUT   28H       ;  TIMER
203D  3E C0             MVI   A,START   ;START
203F  D3 28             OUT   28H       ;  TIMER
2041  3E 01             MVI   A,SEIN1   ;SET
2043  32 C1 20          STA   SEIN      ;  SEINBIT
2046  F1                POP   PSW
2047  FB                EI
2048  C9                RET
                        ORG   20C8H     ;{20CEH}
20C8  C3 38 20          JMP   TMINT
```

fig.17a

```
                        ORG   2060H     ;FIG.9
2060  C5        CODE:   PUSH  B
2061  0E 00             MVI   C,00H     ;INIT TELLER
2063  06 08             MVI   B,08H     ;INIT HULPTELLER
2065  0F        COD1:   RRC             ;BIT NAAR CARRY SF
2066  D2 6A 20          JNC   COD2      ;SPRING ALS BIT ≠ 1
2069  0C                INR   C         ;VERHOOG (TELLER)
206A  05        COD2:   DCR   B         ;8 MAAL GEROTEERD?
206B  C2 65 20          JNZ   COD1      ; NEE: TEST VOLGENDE BIT
206E  79                MOV   A,C       ; JA: (TELLER) NAAR ACCU
206F  C1                POP   B
2070  C9                RET
                        END
```

fig.17b

```
                        ORG   2060H     ;FIG.11
2060  C5        CODE:   PUSH  B
2061  A7                ANA   A         ;RESET CARRY SF
2062  0E 00             MVI   C,00      ;INIT TELLER
2064  1F        COD1:   RAR             ;ROTEER (A) EN (CSF)
2065  D2 6C 20          JNC   COD2      ;SPRING ALS CARRY SF ≠ 1
2068  0C                INR   C         ;VERHOOG (TELLER)
2069  C3 64 20          JMP   COD1      ;TEST VOLGENDE BIT
206C  79        COD2:   MOV   A,C       ;(TELLER) NAAR ACCU ,
206D  C1                POP   B
206E  C9                RET
                        END
```

fig.17c

```
                        ORG   2060H     ;FIG.12
2060  C5        CODE:   PUSH  B
2061  37                STC             ;SET CARRY SF
2062  0E 08             MVI   C,08H     ;INIT TELLER
2064  17        COD1:   RAL             ;ROTEER (A) EN (CSF)
2065  DA 6C 20          JC    COD2      ;SPRING ALS CARRY SF ≠ 0
2068  0D                DCR   C         ;VERLAAG (TELLER)
2069  C3 64 20          JMP   COD1      ;TEST VOLGENDE BIT
206C  79        COD2:   MOV   A,C       ;(TELLER) NAAR ACCU
206D  C1                POP   B
206E  C9                RET
                        END
```

fig.17d

```
                        ORG   2060H     ;FIG.13
2060  E5        CODE:   PUSH  H
2061  26 28             MVI   H,28H     ;INIT HIGH ORDER ADRES-BYTE
2063  6F                MOV   L,A       ;HAAL LOW ORDER ADRES-BYTE
2064  7E                MOV   A,M       ;HAAL BINAIRE WAARDE UIT TABEL
2065  E1                POP   H
2066  C9                RET
                        ORG   2800H     ;PLAATS TABEL MET 9
2800  00                DB    00H       ;BINAIRE WAARDEN IN
2801  01                DB    01H       ;HET GEHEUGEN
                        ORG   2803H
2803  02                DB    02H
                        ORG   2807H
2807  03                DB    03H
                        ORG   280FH
280F  04                DB    04H
                        ORG   281FH
281F  05                DB    05H
                        ORG   283FH
283F  06                DB    06H
                        ORG   287FH
287F  07                DB    07H
                        ORG   28FFH
28FF  08                DB    08H
                        END
```

fig.17e

```
                        ORG   2060H     ;FIG.14
2060  C5        CODE:   PUSH  B
2061  47                MOV   B,A       ;BITCOMBINATIE NAAR REG
2062  AF                XRA   A         ;INIT ACCU
2063  4F                MOV   C,A       ;INIT TELLER
2064  B8        COD1:   CMP   B         ;(A)=(REG)?
2065  CA 6E 20          JZ    COD2      ; JA:SPRING
2068  87                ADD   A         ; NEE: VOEG EEN 1
2069  3C                INR   A         ;      TOE AAN (ACCU)
206A  0C                INR   C         ;VERHOOG (TELLER)
206B  C3 64 20          JMP   COD1      ;VERGELIJK OPNIEUW
206E  79        COD2:   MOV   A,C       ;(TELLER) NAAR ACCU
206F  C1                POP   B
2070  C9                RET
                        END
```

fig.17f

```
                        ORG   2060H     ;FIG.15
2060  C5        CODE:   PUSH  B
2061  47                MOV   B,A       ;BITCOMBINATIE NAAR REG
2062  3E 0F             MVI   A,0FH     ;INIT ACCU
2064  0E 04             MVI   C,04H     ;INIT TELLER
2066  B8        COD1:   CMP   B         ;
2067  CA 79 20          JZ    COD3      ;SPRING ALS (A)=(REG)
206A  D2 73 20          JNC   COD2      ;SPRING ALS (A)>(REG)
206D  87                ADD   A         ;(A)<(REG): VOEG EEN 1
206E  3C                INR   A         ;           TOE AAN (ACCU)
206F  0C                INR   C         ;  VERHOOG (TELLER)
2070  C3 66 20          JMP   COD1
2073  3D        COD2:   DCR   A         ;(A)>(REG): NEEM EEN 1
2074  0F                RRC             ;           WEG UIT (ACCU)
2075  0D                DCR   C         ;  VERLAAG (TELLER)
2076  C3 66 20          JMP   COD1
2079  79        COD3:   MOV   A,C       ;(TELLER) NAAR ACCU
207A  C1                POP   B
207B  C9                RET
                        END
```

fig.17g

Programma-lus	(REG)	U_p(V)	b_7	(TELLER)	(REG) = 1 ?
ervoor	128	-	-	128	-
1e maal	64	1,28	0	192	nee
2e maal	32	1,92	1	160	nee
3e maal	16	1,60	1	144	nee
4e maal	8	1,44	1	136	nee
5e maal	4	1,36	0	140	nee
6e maal	2	1,40	1	138	nee
7e maal	1	1,38	1	137	ja

Tabel 5

In dit geval biedt de trial and error-methode duidelijk de voorkeur boven de oplossing van fig.18.

LIFTSTURING.

Liftsturing

1. INLEIDING

In deze les wordt een microcomputertoepassing besproken, waarin een lift moet worden bestuurd. Deze vorm van procesbesturing speelt zich af in een gebouw met een begane grond en 3 verdiepingen (etages). Voor het gemak noemen we de begane grond in deze les vaak etage 0.

Evenals in voorgaande lessen zijn de eerste fasen van het ontwikkelingsproces (probleemanalyse en algemeen stroomdiagram) niet machine-gericht. Echter bij het opstellen van een gedetailleerd stroomdiagram en het schrijven van een programma moeten we rekening houden met de eigenschappen van een bepaald type microcomputer. In deze les zullen we dan weer uitgaan van de SDK 85.
Probeer steeds zoveel mogelijk zelf te doen. Test uw programma op uw microcomputer. Als uw programma niet aan de gestelde eisen voldoet, moet u in uw eigen programma correcties aanbrengen. Bestudeer pas onze oplossing, als die van u volkomen correct functioneert.
Wanneer u alle voorgaande lessen op de juiste wijze hebt doorgenomen, moet u in staat zijn een probleem, zoals in deze les is gegeven, in een foutloos programma om te zetten.

Mocht u halverwege het ontwikkelingsproces toch vastlopen, leg dan deze les terzijde en begin een of twee dagen later opnieuw.
Lees in die tussentijd b.v. enkele van de voorgaande lessen op uw gemak door. Daarna zal het oplossen van dit probleem u zeker lukken.

Een methode die we in enkele voorgaande lessen hebben toegepast, is het splitsen van het totale probleem in een aantal deelproblemen.
Voor elk deelprobleem wordt dan een deelprogramma ontwikkeld.
Pas aan het eind van het ontwikkelingsproces wordt een hoofdprogramma geschreven, waarin de deelprogramma's in de juiste volgorde worden geplaatst.
Deze methode zullen we in de les zoveel mogelijk aanhouden.

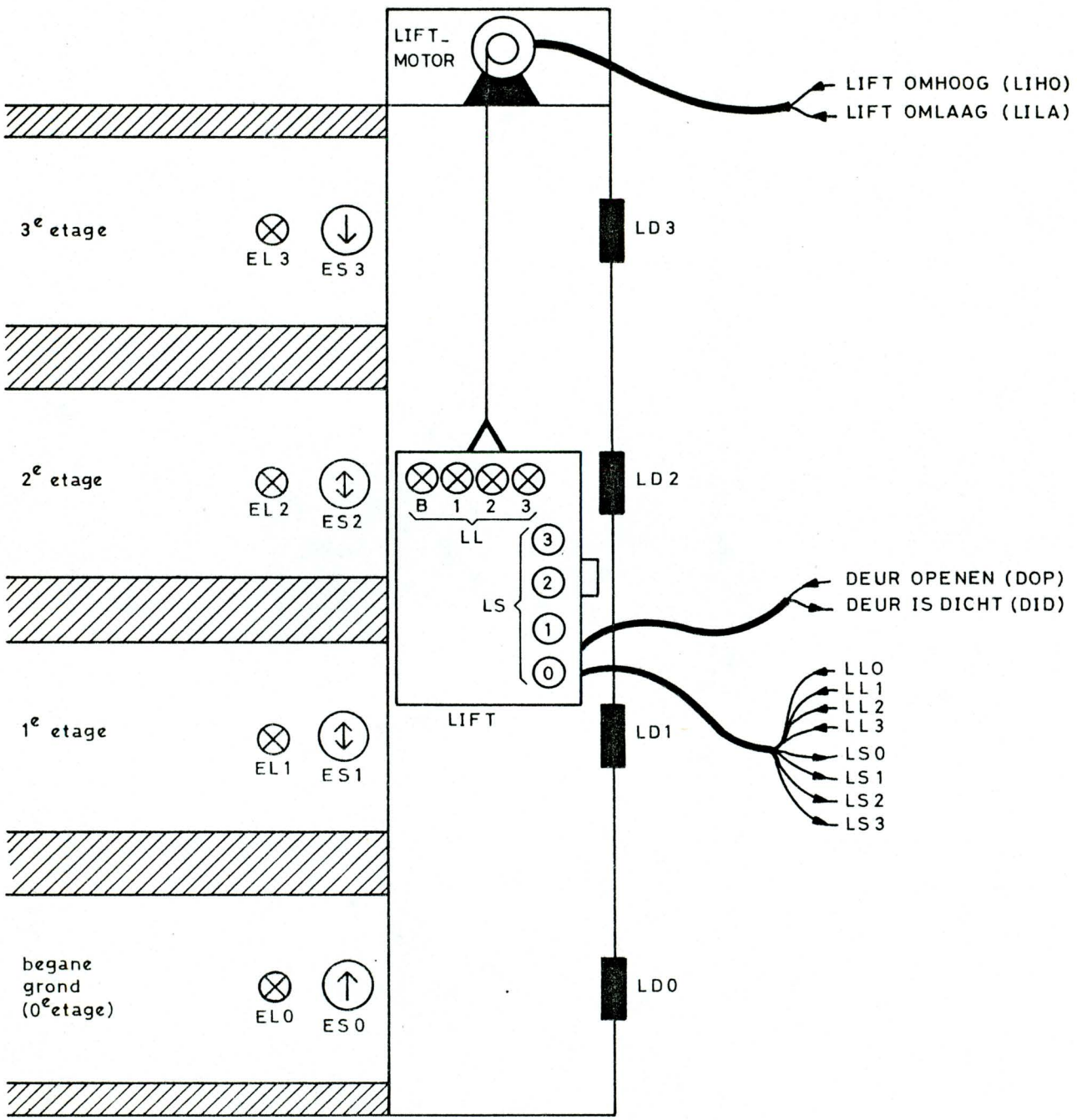

LIFT_
MOTOR
LIFT OMHOOG (LIHO)
LIFT OMLAAG (LILA)
3e etage
EL 3
ES 3
LD 3
2e etage
EL 2
ES 2
LD 2
B 1 2 3
LL
LS
3
2
1
0
DEUR OPENEN (DOP)
DEUR IS DICHT (DID)
LIFT
1e etage
EL 1
ES 1
LD 1
LL0
LL 1
LL 2
LL 3
LS 0
LS 1
LS 2
LS 3
begane
grond
(0e etage)
EL 0
ES 0
LD 0

fig.1

2. PROBLEEMOMSCHRIJVING

a. De liftinstallatie

Fig. 1 geeft een schematisch overzicht van de liftinstallatie. Deze bevindt zich in een gebouw met de etages 0 t/m 3. Op elke etage bevindt zich een etageschakelaar (ES0 t/m ES3). D.m.v. deze druktoetsen kunnen we de lift naar de betreffende etage laten komen. Wanneer een aanvraag is geaccepteerd (d.w.z. door de microcomputer in het geheugen is opgeslagen) gaat de bijbehorende etagelamp (EL0 t/m EL3) branden. In de lift zelf bevinden zich vier liftlampen (LL0 t/m LL3) en vier liftschakelaars LS0 t/m LS3. M.b.v. de liftschakelaars kunnen we de gewenste bestemming aangeven. Op de liftlampen is steeds af te lezen waar de lift zich bevindt.

Voor wat betreft het besturen van de liftdeur zijn er twee signalen aanwezig. De microcomputer kan d.m.v. het signaal DOP opdracht geven de liftdeur te openen of te sluiten. Het signaal DID dient om de microcomputer mee te delen, dat de liftdeur wel of niet dicht is. In de lift zelf bevindt zich een schakeling, die voorkomt dat de liftdeur wordt gesloten, als de deuropening nog niet vrij is (in de deurpost zit b.v. een elektronisch oog). Deze schakeling voorkomt tevens het openen van de deur, op een moment dat de lift nog niet geheel tot stilstand is gekomen.

M.b.v. de signalen LIHO en LILA kan de microcomputer de liftmotor aansturen. Als geen van beide signalen actief is, dan stopt de lift geleidelijk.

In de liftschacht zijn vier liftdetectoren (LD0 t/m LD3) aangebracht. Deze detectoren geven een actief signaal af, gedurende de tijd, dat het zwarte blokje (aan de rechterkant van de lift gemonteerd) zich geheel of gedeeltelijk voor de detector bevindt.

b. Starten en stoppen van de lift

De lift mag niet abrupt starten en stoppen. Dit moet geleidelijk gebeuren. Om dit te realiseren behoeven we geen voorzieningen in de microcomputer aan te brengen. De liftmotor is n.l. zo geconstrueerd, dat na het wegnemen van een actief LIHO- of LILA-signaal de lift geleidelijk tot stilstand komt. Na het wegnemen van een actief signaal zal de lift dus nog een bepaalde afstand afleggen.
De hoogte en de afmetingen van de liftdetectoren zijn zo gekozen, dat de lift na het afleggen van deze remweg op exact de juiste positie t.o.v. de etage stilstaat. We zullen dit verduidelijken aan de hand van een voorbeeld.

Stel, dat de lift bezig is van de eerste naar de tweede etage te stijgen. Daar moet de lift stoppen. In het timing-diagram van fig.2 is weergegeven wat er dan achtereenvolgens gebeurt.

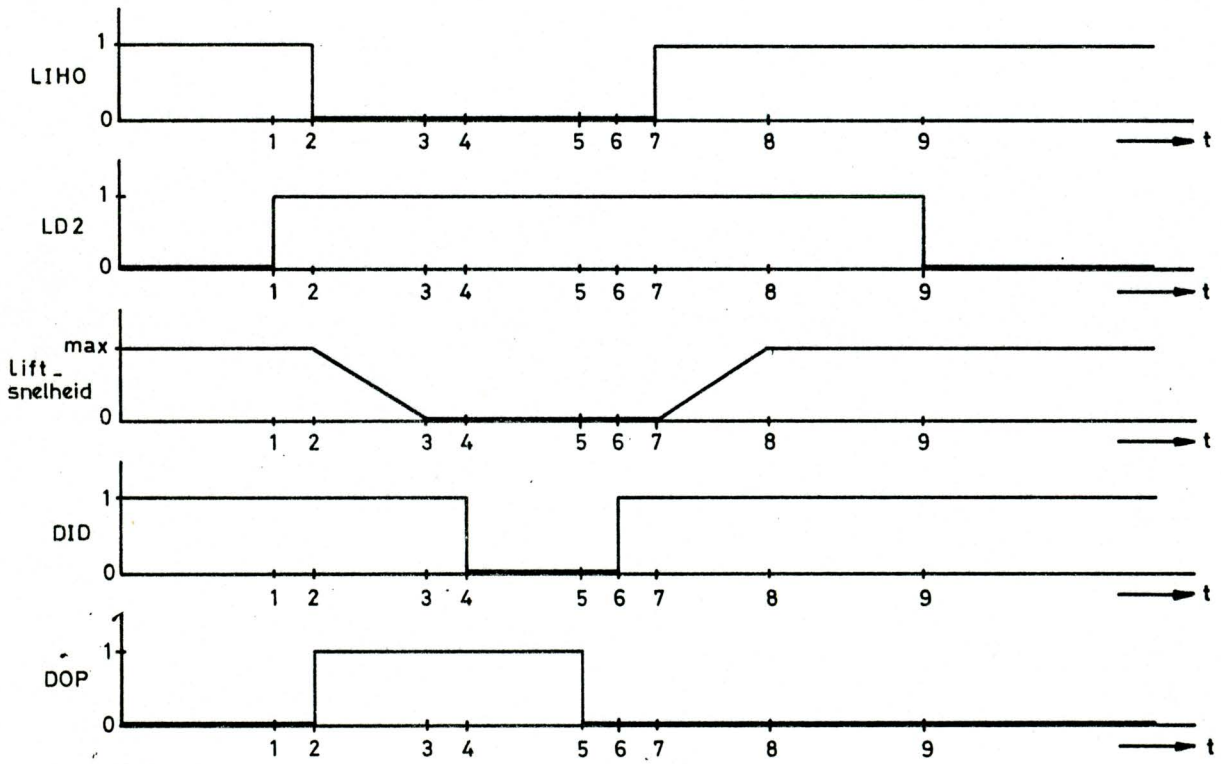


fig.2

De lift stijgt met maximale snelheid, LIHO is dus actief. Op t=1 is de lift ter hoogte van LD2. Deze geeft dan een actief signaal af. Dit wordt door de microcomputer gedetecteerd. Deze maakt op t=2 het LIHO-signaal 0. (Het tijdsverschil tussen t=1 en t=2 wordt veroorzaakt door de tijd, die nodig is voor het uitvoeren van de instructies in de microcomputer).

Merk op, dat op t=2 de microcomputer d.m.v. een actief DOP-signaal aangeeft, dat de liftdeur geopend dient te worden. Aangezien de lift nog niet tot stilstand is gekomen, zal hierop nog niet worden gereageerd door de elektronische deurschakeling.

Op t=3 is de lift tot stilstand gekomen. De deurschakeling zal dan beginnen met het openen van de liftdeur. Op t=4 wordt gemeld, dat de liftdeur niet meer dicht is (dit wil niet zeggen, dat de deur al volledig is geopend).

Na een (softwarematig bepaalde) tijd maakt de microcomputer het DOP-signaal weer 0, om aan te geven dat de liftdeur moet worden gesloten. Dit gebeurt op t=5.

Op t=6 meldt de schakeling, dat de liftdeur dicht is. Dit wordt door de microcomputer gedetecteerd, waarna op t=7 LIHO weer actief wordt gemaakt om de lift naar de derde etage te laten stijgen. Door de constructie van de liftmotor komt de lift geleidelijk op gang. Op t=8 wordt de maximale snelheid bereikt.

Op t=9 is de lift buiten het bereik van de LD2. Deze geeft dan weer een 0 af. De in fig.2 weergegeven procedure herhaalt zich op de derde etage zodra LD3 actief wordt, met dat verschil dat de lift daarna alleen door een actief LILA-signaal weer in beweging mag worden gezet.

c. Gestelde eisen

Gevraagd wordt een programma te ontwikkelen, dat aan de volgende eisen voldoet:

1. Direct na het starten van het programma moet de lift naar de begane grond worden gestuurd. Alle eventueel aanwezige aanvragen moeten worden gereset. Daarna mogen pas nieuwe aanvragen in ontvangst worden genomen (=accepteren) en mag de lift naar de gewenste etage worden gestuurd (=honoreren).
2. Eenmaal per 2 ms moeten de liftschakelaars en de etageschakelaars worden afgetast. Alle gedetecteerde aanvragen moeten dan worden geaccepteerd.
3. Alle aanvragen dienen in volgorde van binnenkomst te worden gehonoreerd. Hierop is echter een uitzondering mogelijk. (Zie de opmerking aan het eind van deze paragraaf). Er wordt wat dit betreft een verschil gemaakt tussen aanvragen, die van de etages afkomstig zijn en die vanuit de lift. Deze laatste moeten zoveel mogelijk eerst worden afgewerkt, anders zou het voor kunnen komen, dat er een tijd lang geen mensen uit de lift stappen, omdat er alleen etage-aanvragen worden gehonoreerd.
4. De tijd gedurende welke de lift op een bepaalde etage blijft staan is minimaal 5 seconden. (In fig.2 moet tussen t=4 en t=5 dus een tijdverschil van 5 seconden liggen).
5. Wanneer de lift op een bepaalde etage is gestopt, dan moeten de aanvragen voor deze etage worden gereset. (Deze aanvragen zijn n.l. op dat moment gehonoreerd).
6. Alle lijnen zijn gebufferd, d.w.z. dat de microcomputer de deur, de liftmotor en de lampen direct kan aansturen. De door de liftinstallatie afgegeven enen en nullen kunnen rechtstreeks via input-poorten worden ingelezen.

7. Voor het aansturen van lampen geldt: 1= lamp brandt
   0= lamp uit
   De schakelaars geven alle een 0 af, op het moment dat ze worden ingedrukt.
8. Voor de signalen LILA en LIHO geldt:
   1= actief
   0= niet actief.
   Deze signalen mogen natuurlijk niet tegelijkertijd actief zijn.
9. Voor het DOP-signaal geldt:
   1= deur is open
   0= deur sluiten
10. Voor het DID-signaal geldt:
   1= deur is dicht
   0= deur is niet dicht.
11. Een liftdetector geeft een 1 af, als de lift zich binnen het bereik van de betreffende detector bevindt.
12. Als ergeen aanvraag is te honoreren, moet de lift op de laatst bereikte etage blijven wachten op een nieuwe aanvraag. De liftdeur staat dan continu open.
13. We gaan ervan uit, dat op een bepaald moment niet meer dan een van de liftschakelaars is ingedrukt. Hetzelfde geldt voor de etageschakelaars.

Opmerking bij punt 3:
Van het in volgorde van binnenkomst honoreren van de aanvragen moet worden afgeweken, in die gevallen waarin de lift een etage passeert, waarvoor een aanvraag is ontvangen.
Stel b.v. dat de lift bezig is met het dalen van de derde etage naar de begane grond. Als er nu op de eerste etage een aanvraag wordt ingediend. Moet de lift hier onderweg stoppen.Dit mag natuurlijk alleen gebeuren als de aanvraag wordt ontvangen, voordat LD1 een actief signaal afgeeft.
De lift moet namelijk wel op de juiste manier kunnen worden afgeremd.

PROBEER NU EERST ZELF TOT EEN PROGRAMMA TE KOMEN. BESTUDEER DAARNA ONZE OPLOSSING.

3. PROBLEEMANALYSE

Uit de probleemomschrijving volgt, dat de taak van de microcomputer in twee hoofdfuncties is te splitsen, n.l.

a. Het accepteren van aanvragen. Dit is dus het in volgorde opslaan van alle ontvangen aanvragen.
b. Het honoreren van aanvragen. Dit is het besturen van de lift, zodat deze achtereenvolgens alle aangevraagde etages bereikt.

Vraag 1: Het aftasten van de schakelaars gebeurt wel/niet op vaste tijden.

Daar het accepteren van eventuele aanvragen, dus het aftasten van de schakelaars,elke 2 ms plaats moet vinden, ligt het voor de hand hiervoor een hardware timer te gebruiken. In de bijbehorende interrupt service routine vindt dan het accepteren van de ontvangen aanvragen plaats. Door de instructies van het hoofdprogramma worden dan de geaccepteerde aanvragen gehonoreerd. Deze scheiding van beide taken is in fig.3 weergegeven.

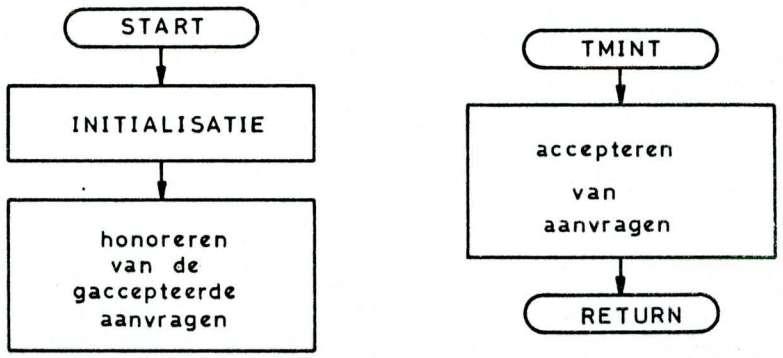


fig.3

In de volgende paragrafen zullen we de afzonderlijke taken in gedetailleerde stroomdiagrammen uitdrukken.

4. ACCEPTEREN VAN AANVRAGEN

Eenmaal per 2 ms moeten de lift- en etage-schakelaars worden afgetast. Blijkt dat er een of meer schakelaars zijn ingedrukt, dan moet(en) deze aanvraag resp. aanvragen bij de overige nog niet gehonoreerde aanvragen worden opgeslagen. In een hiervoor gereserveerd geheugendeel ontstaat dus een tabel van alle nog te honoreren aanvragen. Fig.4 is het algemeen stroomdiagram voor de timer interrupt service routine.

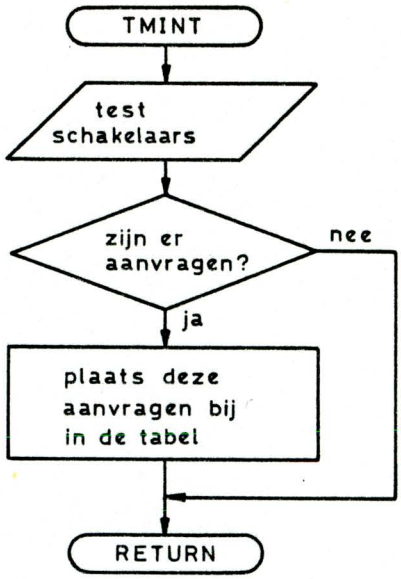


fig.4

De tabel wordt regelmatig door het hoofdprogramma geraadpleegd bij het bepalen van de liftbestemming. Daarom worden er enkele eisen aan deze tabel gesteld. Uit de tabel moet n.l. direct kunnen worden bepaald:
a. in welke volgorde de aanvragen zijn ontvangen,
b. welke aanvragen vanuit de lift en welke d.m.v. de etageschakelaars zijn ingediend,
c. welke aanvraag als eerst volgende moet worden gehonoreerd,
d. of er voor een bepaalde etage een aanvraag is ontvangen.

Om een duidelijk verschil tussen aanvragen vanuit de lift en die van de etages te maken, hebben we gekozen voor twee tabellen.

Vraag 2: De tabel voor de liftschakelaars moet maximaal ..... aanvragen kunnen bevatten. De tabel voor de etageschakelaars moet maximaal ..... aanvragen kunnen bevatten.

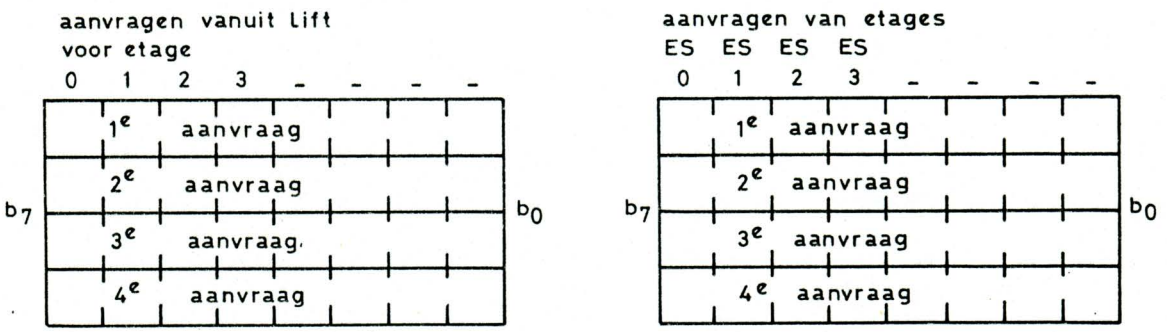


fig.5

Antw.2: 4; 4.

In fig. 5 zijn deze twee tabellen weergegeven. De tabel voor de vanuit de lift ingediende aanvragen kan maximaal 4 verschillende aanvragen bevatten. Dit is voldoende omdat er vier etages zijn. Twee aanvragen voor dezelfde etage kunnen n.l. als één aanvraag worden beschouwd. Om dezelfde reden kan de tabel voor de etageschakelaars ook maximaal 4 aanvragen bevatten.

We hebben in feite aan alle aan de aanvraagtabel gestelde eisen voldaan. Echter om nu te testen of er voor een bepaalde etage een aanvraag is ingediend, zouden nu maximaal 8 geheugeninhouden moeten worden getest. Om deze test te verkorten kunnen we de beide afzonderlijke tabellen uitbreiden met een geheugenwoord, waarin alle aanvragen als het ware zijn samengevat. Dit is in fig. 6 weergegeven.

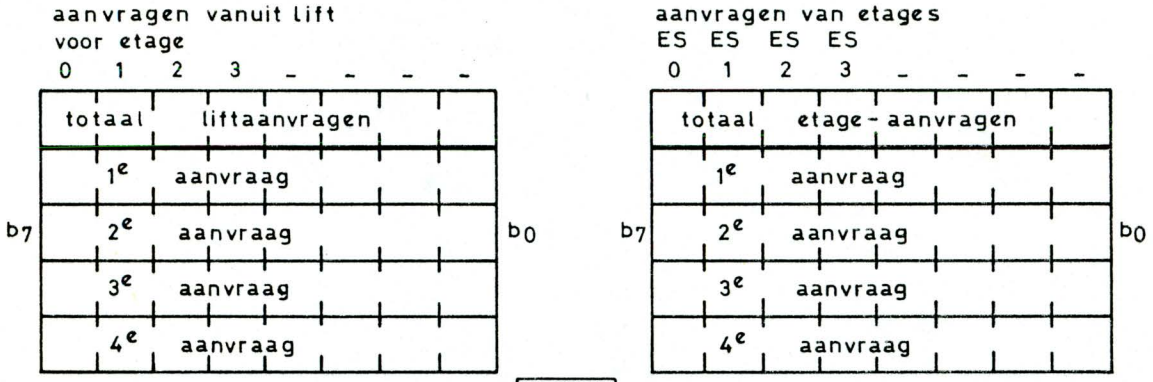


fig.6

We zullen dit verduidelijken met een voorbeeld. Stel b.v. dat achtereenvolgens de etageschakelaars ES1, ES3 en ES0 worden ingedrukt. We gaan ervan uit, dat de niet gebruikte bits in de tabel alle 0 zijn.

Vraag 3: De inhoud van het eerste geheugenwoord in de tabel voor de etage-aanvragen wordt dan .....$_2$.

De inhoud van de tabel voor de etage-aanvragen is dan zoals in fig.7 is weergegeven.

	ES 0	ES 1	ES 2	ES 3	-	-	-	-	
	1	1	0	1	0	0	0	0	
1e	0	1	0	0	0	0	0	0	
2e	0	0	0	1	0	0	0	0	b0
3e	1	0	0	0	0	0	0	0	
-	0	0	0	0	0	0	0	0	

fig.7

Antw.3: 11010000.

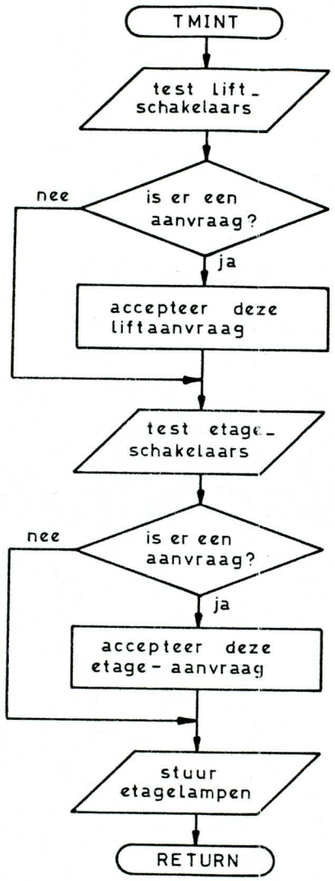


fig.9

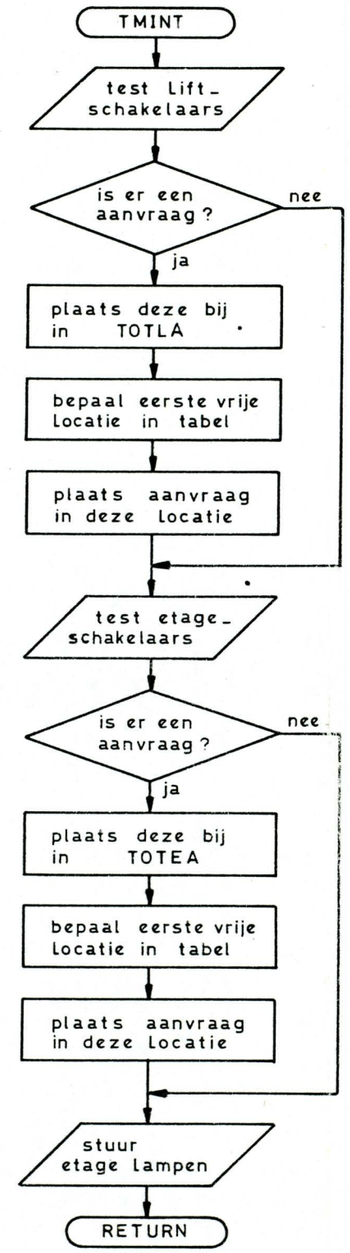
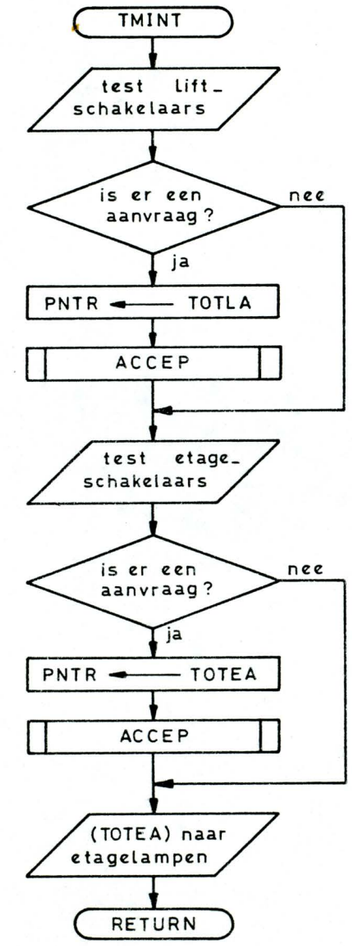


fig.10



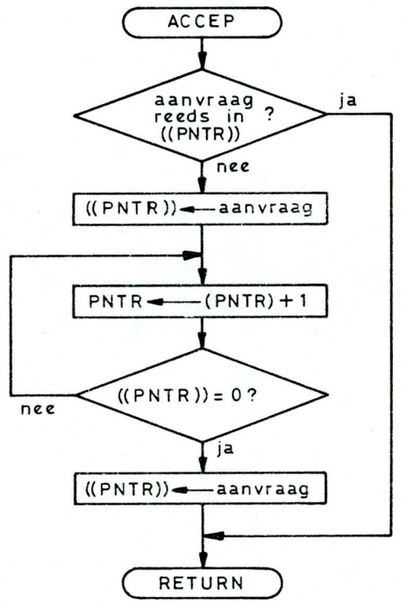


fig.11

De taak van de interrupt service routine is dus het opslaan van alle ontvangen aanvragen in de tabel. Voor het gemak hebben we in fig.8 deze tabel nog eens weergegeven maar nu voorzien van symbolische namen.

"TOTLA" = totaal liftaanvragen.
"TABLA" = eerste adres in de tabel voor de liftaanvragen.
"TOTEA" = totaal etage-aanvragen.
"TABEA" = eerste adres in de tabel voor de etage-aanvragen.

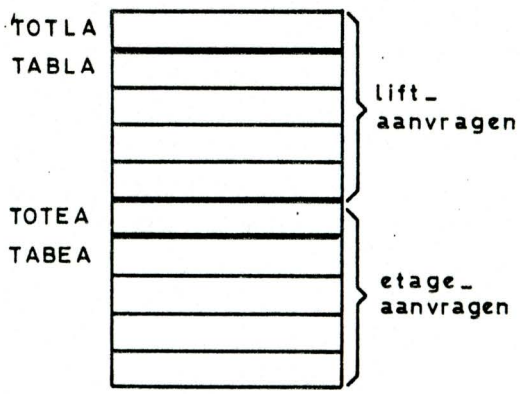


fig.8

In fig.9 is globaal aangegeven wat er achtereenvolgens in de interrupt service routine moet gebeuren. In fig.10 is het accepteren van een aanvraag gedetailleerder weergegeven. U ziet, dat er een grote overeenkomst is tussen het accepteren van een liftaanvraag en het accepteren van een etage-aanvraag. Het enige verschil zit in de beginadressen (TOTLA resp. TOTEA) van de te vullen tabellen.

In fig.11 is voor het accepteren van beide aanvragen één subroutine "ACCEP" gebruikt. Voor het aanroepen van deze subroutine wordt dan een geheugen-pointer PNTR gevuld met het beginadres van de betreffende tabel.

In de subroutine wordt eerst getest of deze aanvraag misschien al eerder is geaccepteerd. Is dit niet het geval, dan wordt bepaald wat de eerstvolgende vrije locatie in de tabel is. Is deze gevonden, dan wordt hierin de aanvraag opgeslagen.

In fig.11 is tevens aangegeven, dat het aansturen van de etagelampen gebeurt door de inhoud van TOTEA uit te voeren. In TOTEA staan n.l. alle geaccepteerde etage-aanvragen opgeslagen.

Voordat we dit gedetailleerd stroomdiagram in een programma gaan omzetten, zullen we eerst een oplossingsmethode voor het hoofdprogamma ontwikkelen.

5. HONOREREN VAN AANVRAGEN

Als er een aanvraag is geaccepteerd, dus in de tabel opgeslagen, dan moet deze worden gehonoreerd.
In fig.12 is weergegeven hoe het honoreren van één aanvraag ongeveer dient te verlopen.
Als er één of meer aanvragen in de tabel staan, dan moet de lift in de juiste richting in beweging worden gezet. Dan moet er worden gewacht, totdat één van de liftdetectoren een actief signaal afgeeft.

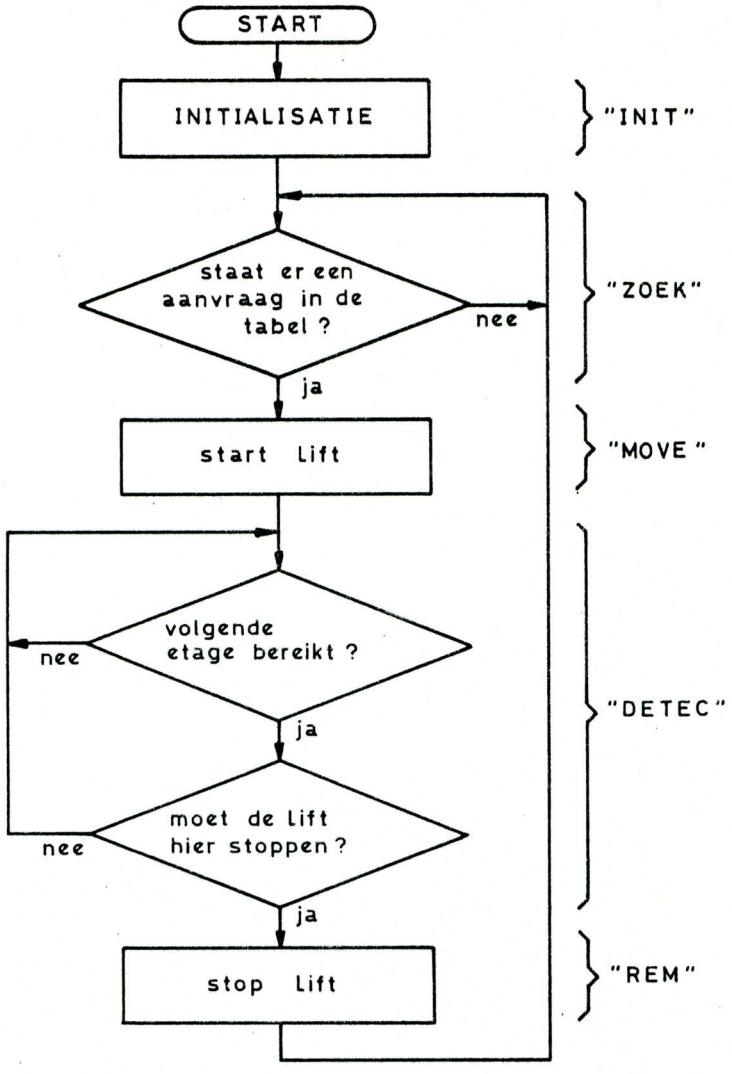


fig.12

In de tabel wordt dan gecontroleerd of er voor de betreffende etage een aanvraag was ontvangen. Zo niet, dan kan de lift doorgaan. Is er wel een aanvraag voor de betreffende etage, dan moet de lift worden gestopt. In de tabel moet deze aanvraag dan natuurlijk worden gereset, want deze is nu gehonoreerd.
Voor het gemak splitsen we fig.12 in een aantal programmadelen, die we afzonderlijk gaan ontwikkelen. Door het naderhand samenvoegen van deze programmadelen ontstaat het totale hoofdprogramma. De deelprogramma's die wij gekozen hebben zijn:

"INIT": dit is de initialisatie. Deze beschrijven we pas na de overige programmadelen.
"ZOEK": hierin wordt de eerstvolgende te accepteren aanvraag opgezocht.
"MOVE": hierin wordt de lift in beweging gezet.
"DETEC": hierin wordt getest of de lift al een volgende etage heeft bereikt, en zoja, of de lift hier moet stoppen.
"REM": hierin wordt de lift op de juiste wijze tot stilstand gebracht.

a. Programmadeel "ZOEK"

In dit programmadeel moet uit de inhoud van de tabel de eerstvolgende te honoreren aanvraag worden bepaald. Dit kan b.v. volgens de in fig.13 weergegeven methode. Er wordt eerst getest of er een lift-aanvraag is. Zoniet dan kan een eventuele etage-aanvraag worden gehonoreerd.
Wanneer er in een van de beide gevallen een aanvraag blijkt te zijn opgeslagen, dan bevindt deze zich in de tabel, op de locatie direct na TOTLA resp. TOTEA. Deze aanvraag moet naar de CPU worden overgebracht, zodat in het programmadeel "MOVE" bepaald kan worden of de lift dient te dalen of te stijgen.
In fig.14 is een gedetailleerd stroomdiagram voor "ZOEK" weergegeven.
Hierin is weer gebruik gemaakt van de geheugen-pointer PNTR, om locaties binnen de tabel te adresseren.
De gevonden aanvraag, die als eerste moet worden gehonoreerd, wordt in de CPU in een register REG opgeslagen.
Daarna moet met programmadeel "MOVE" verder worden gegaan.

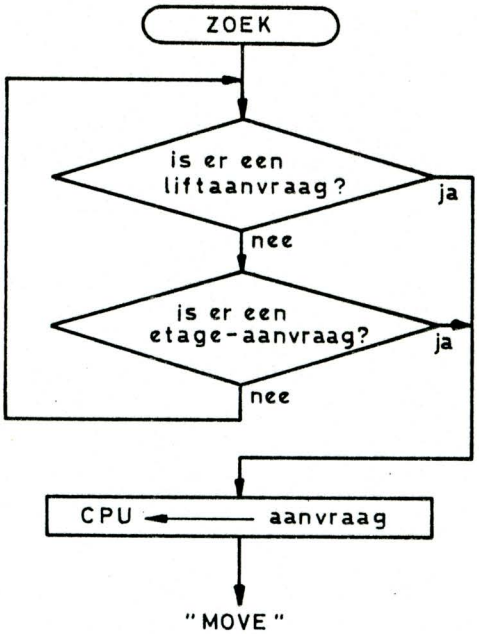


fig.13

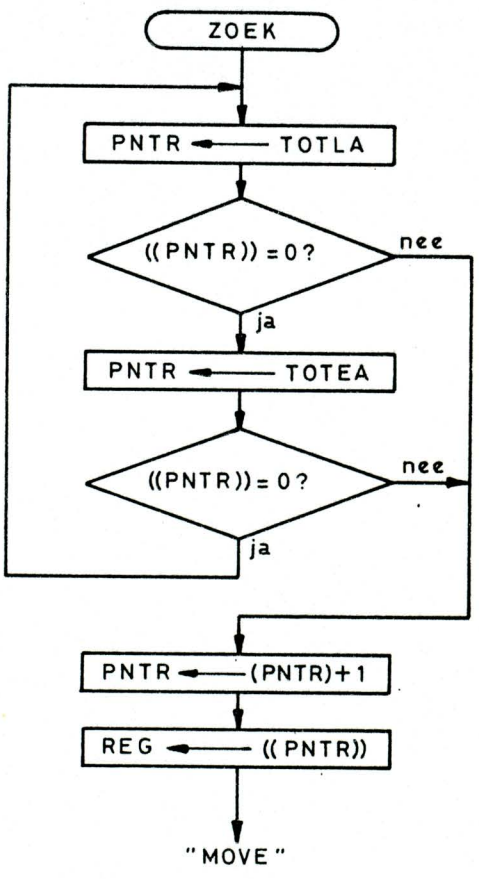


fig.14

b. Programmadeel "MOVE"

In dit programmadeel moet worden bepaald in welke richting de lift zich moet bewegen om de aanvraag te honoreren. Bovendien moet er worden getest of de lift wel naar een andere etage moet om de aanvraag te honoreren. Het kan immers voorkomen dat iemand een etageschakelaar indrukt, op het moment dat de lift al op de betreffende etage staat of vlak voordat de lift op die etage tot stilstand is gekomen. In dat geval moet de aanvraag in de tabel worden gereset en d.m.v. programmadeel "ZOEK" moet de volgende te honoreren aanvraag worden bepaald. In fig.15 is aangegeven, dat het resetten van een aanvraag gebeurt d.m.v. de subroutine "RESET".
Deze subroutine moeten we straks dus nog ontwikkelen.
Als voor het honoreren van de aanvraag de lift wel naar een andere etage moet worden gestuurd, dan moet natuurlijk eerst de liftdeur worden gesloten. Daarna wordt bepaald of de lift moet stijgen of dalen en LIHO resp. LILA wordt actief gemaakt. Daarna moet met programmadeel "DETEC" verder worden gegaan, om te bepalen of de eerstvolgende etage al is bereikt.

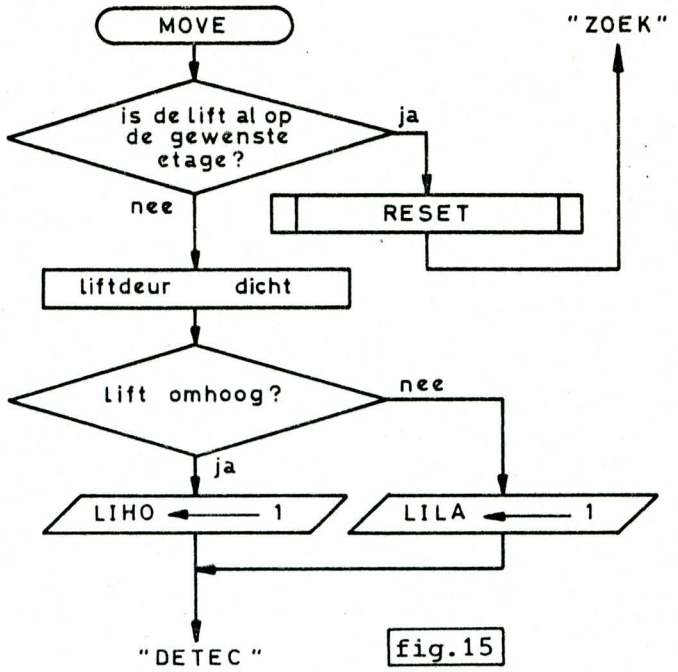


fig.15

Fig.16 is het gedetailleerd stroomdiagram voor "MOVE".
Hierin wordt, om te bepalen of de lift op deze etage moet blijven staan, moet stijgen of moet dalen, steeds de door de liftdetectoren afgegeven bitcombinatie ingevoerd. De accumulatorinhoud geeft dan aan, op welke etage de lift zich bevindt. Deze inhoudt wordt dan vergeleken met de inhoud van REG. Hierin is immers door programmadeel "ZOEK" de bestemming van de lift opgeslagen.

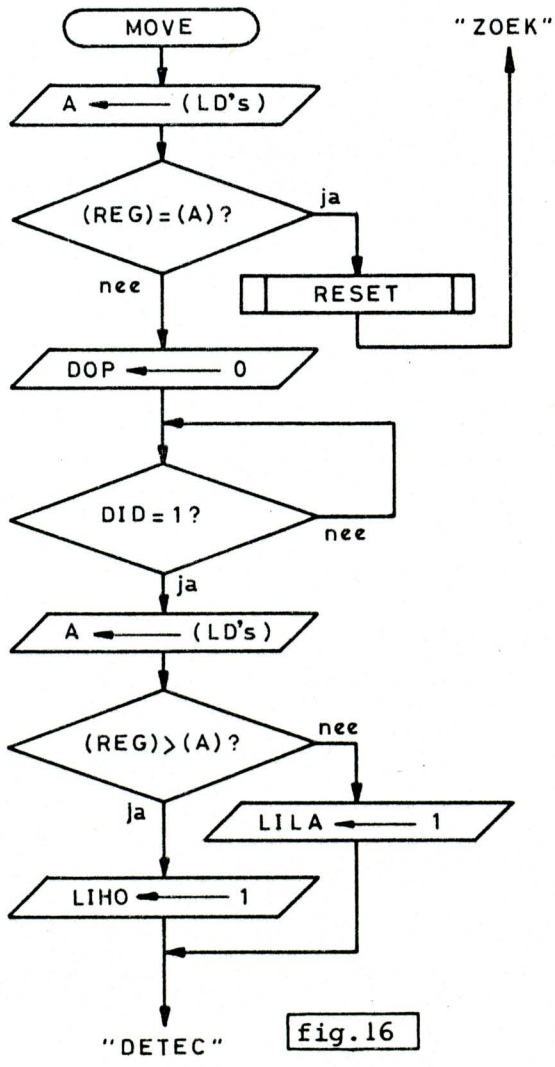


fig.16

c. Programmadeel "DETEC"

Fig.17 is het algemeen stroomdiagram voor programmadeel "DETEC". Dit is rechtstreeks uit fig.12 overgenomen.

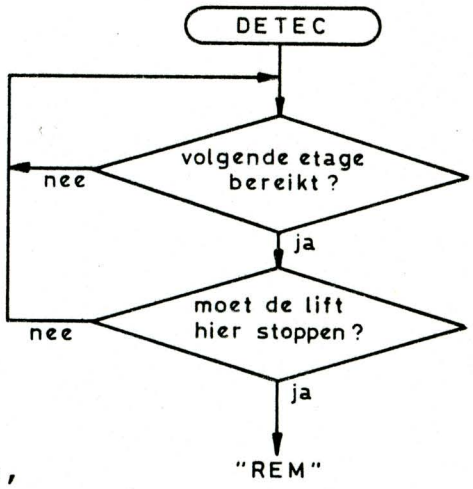


fig.17

Om te bepalen of een volgende etage is bereikt, moet worden getest of één van de liftdetectoren een actief signaal afgeeft. We moeten er dan wel zeker van zijn, dat een gedetecteerd actief signaal niet afkomstig is van de liftdetector op de etage, wanneer de lift net is gestart.
Het duurt immers nog even, voordat de lift buiten het bereik van deze liftdetector is (zie fig.2). Er moet dus eerst worden gewacht, totdat geen der liftdetectoren actief is.
Als daarna een actief detectorsignaal wordt ontvangen, weten we zeker, dat dit van een volgende etage afkomstig is.
Deze procedure wordt d.m.v. de eerste 2 programmalussen in fig.18 gerealiseerd. Als er dan een actief signaal van een volgende etage is ontvangen wordt, weer m.b.v.de geheugen-pointer PNTR, getest of er voor de betreffende etage een aanvraag in TOTLA of TOTEA is opgeslagen.
Zoja, dan moet de lift door programmadeel "REM" worden gestopt.
Behoeft de lift op de betreffende etage niet te stoppen, dan moet naar het begin van "DETEC" worden teruggesprongen.

In fig.18 is tevens aangegeven, dat de liftlampen dienen te worden aangestuurd als de lift een nieuwe etage heeft bereikt.

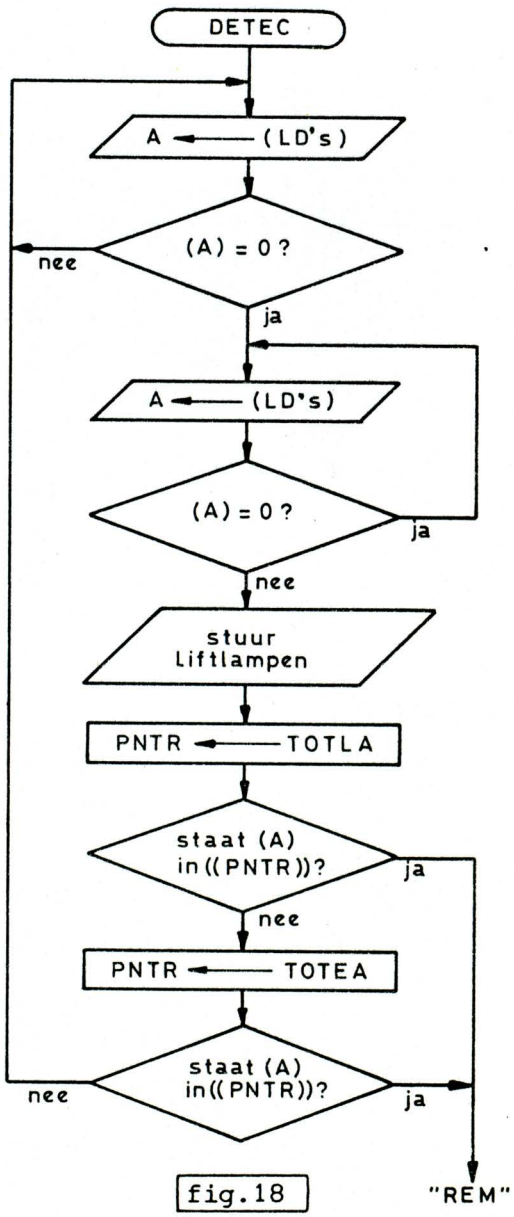


fig.18

d. Programmadeel "REM"

Door dit programmadeel moet de lift op de bereikte etage worden gestopt. D.w.z. dat het actieve signaal (LIHO of LILA) voor de liftmotor moet worden weggenomen (fig.19).
De aanvraag voor deze etage is nu gehonoreerd en kan dus worden gereset. Daarna moet 5 seconden worden gewacht, voordat teruggesprongen wordt naar programmadeel "ZOEK" (zie punt 4 van de probleemomschrijving).

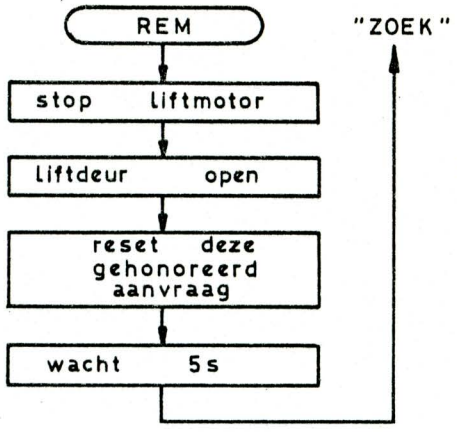


fig.19

In fig.20 is het gedetailleerd stroomdiagram voor "REM" weergegeven.
Hierin wordt, overeenkomstig de beschrijving bij fig.2, direct na het wegnemen van LIHO resp. LILA een actief DOP-signaal aan de liftdeur afgegeven. Door de elektronische deurschakeling wordt de liftdeur echter niet geopend, voordat de lift tot stilstand is gekomen.

Voor het resetten van de gehonoreerde aanvraag maken we gebruik van de subroutine "RESET", die ook vanuit programmadeel "MOVE" wordt aangeroepen. De wachttijd van 5 seconden wordt d.m.v. subroutine "WACHT" gerealiseerd. Behalve programmadeel "INIT" moeten er dus nog twee subroutines worden ontwikkeld.

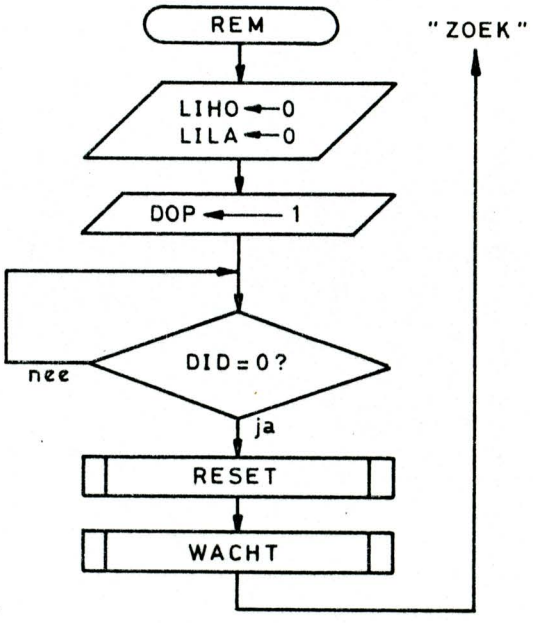


fig. 20

e. Subroutine "RESET"

De taak van subroutine "RESET" is het resetten van de aanvragen voor de etage waarop de lift zich bevindt.
Zoals in fig.21 is weergegeven kunnen dit zowel liftaanvragen als etage-aanvragen zijn.
De methode voor het resetten van beide typen aanvragen is dezelfde, echter het beginadres van de betreffende tabel (TOTLA resp. TOTEA) is niet steeds gelijk.

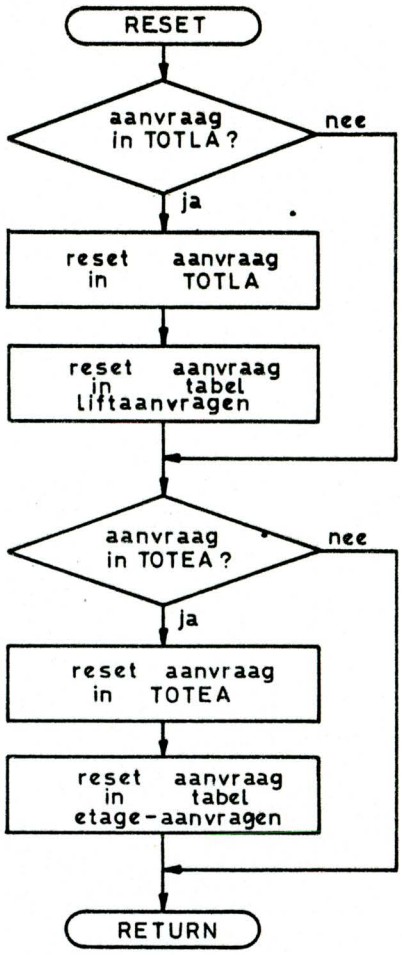


fig.21

We hebben daarom gekozen voor de methode van fig. 22a. Om te bepalen voor welke etage een aanvraag moet worden gereset (dit is de etage waarop de lift zich tijdens het aanroepen van "RESET" bevindt), worden de uitgangssignalen van de liftdetectoren ingelezen en in een register met de symbolische naam BUFF (=buffer) opgeslagen.
Dan wordt, met gebruikmaking van de geheugenpointer PNTR een eventuele liftaanvraag gereset.
Hetzelfde gebeurt daarna voor een eventuele etageaanvraag.

Het feitelijke resetten vindt dan plaats in de subroutine "RESTB" (=reset tabel) die tweemaal vanuit "RESET" wordt aangeroepen. Er is dus sprake van nesten. Het gedetailleerd stroomdiagram voor "RESTB" is in fig.22b weergegeven. Probeer dit stroomdiagram nu zelf te analyseren.

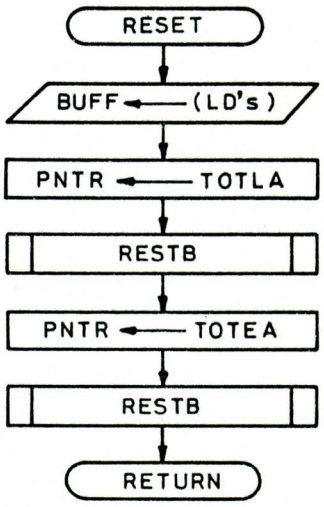


fig.22a

Er wordt eerst getest of de te resetten aanvraag zich bevindt in de locatie, waarin het totaal aan liftaanvragen resp. etage-aanvragen is opgeslagen. Is dit niet het geval, dan moet worden teruggekeerd. Er is dan immers geen aanvraag in de betreffende tabel te resetten.

Moet er wel een aanvraag worden gereset, dan moet dit gebeuren in zowel TOTLA resp. TOTEA als in de daarop volgende tabel. Daarom moet de betreffende aanvraag in de tabel worden opgezocht en worden vervangen door 0. Eventueel daarachter opgeslagen aanvragen moeten dan 1 plaats "naar voren" worden opgeschoven.
Dit gebeurt door de laatste vier blokken van fig.22b.
Let hierin vooral op het laatste blok van deze programmalus. Dit blok geeft aan, dat de inhoud van de locatie, waarvan het adres zich in PNTR bevindt, wordt overgebracht naar die locatie, waarvan het adres wordt gevormd door de inhoud van PNTR verminderd met 1.
De inhoud van PNTR is na het uitvoeren van deze bewerking ongewijzigd. Dit is b.v. te realiseren door het gebruik van een tweede geheugenpointer, waarvan de inhoud 1 lager is dan die van PNTR.
Een andere oplossing is die in fig.23. Hierin wordt de inhoud van PNTR tijdelijk met 1 verlaagd. Echter, voordat teruggesprongen wordt naar het begin van de lus, krijgt PNTR weer de oorspronkelijke inhoud.

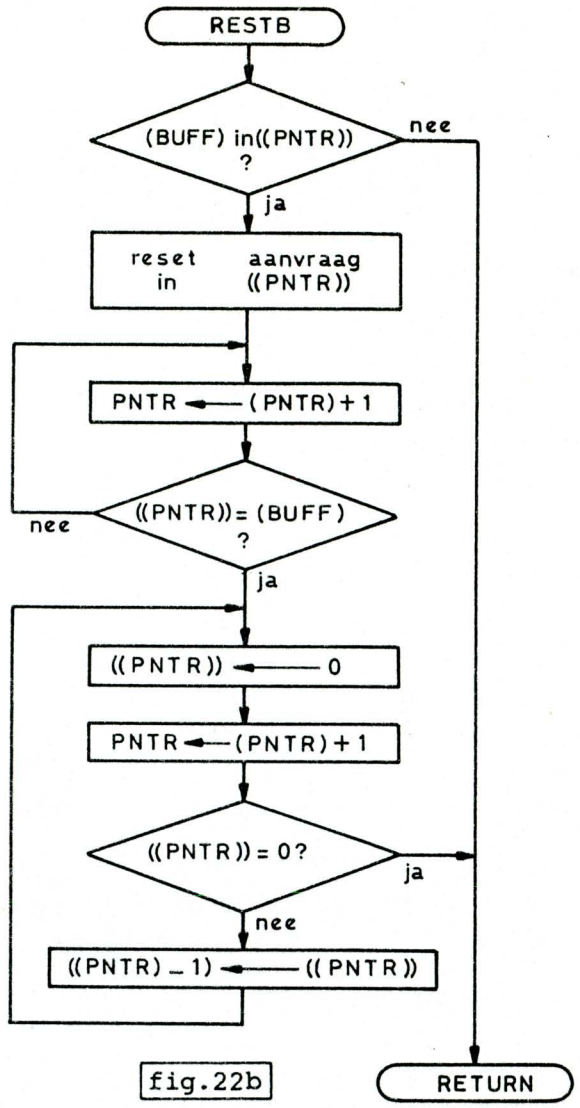


fig.22b

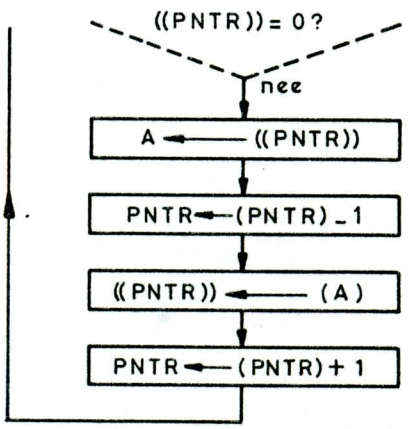


fig.23

Voor de oplossing van fig.22b kan er echter één situatie ontstaan, waarbij deze oplossing niet voldoet. Stel n.l. dat er 4 liftaanvragen (het maximale aantal) en b.v. 2 etage-aanvragen zijn geaccepteerd. De tabelinhouden zijn dan b.v. zoals in fig.24a is weergegeven. Als de lift de eerste etage heeft bereikt wordt subroutine "RESET" aangeroepen.

	0 1 2 3 _ _ _ _
TOTLA	1 1 1 1 0 0 0 0
TABLA	0 0 1 0 0 0 0 0
TABLA+1	0 0 0 1 0 0 0 0
TABLA+2	0 1 0 0 0 0 0 0
TABLA+3	1 0 0 0 0 0 0 0
TOTEA	1 0 1 0 0 0 0 0
TABEA	1 0 0 0 0 0 0 0
TABEA+1	0 0 1 0 0 0 0 0
TABEA+2	0 0 0 0 0 0 0 0
TABEA+3	0 0 0 0 0 0 0 0

fig.24a

	0 1 2 3 _ _ _ _
TOTLA	1 0 1 1 0 0 0 0
TABLA	0 0 1 0 0 0 0 0
TABLA+1	0 0 0 1 0 0 0 0
TABLA+2	0 1 0 0 0 0 0 0
TABLA+3	1 0 0 0 0 0 0 0
TOTEA	1 0 1 0 0 0 0 0
TABEA	1 0 0 0 0 0 0 0
TABEA+1	0 0 1 0 0 0 0 0
TABEA+2	0 0 0 0 0 0 0 0
TABEA+3	0 0 0 0 0 0 0 0

fig.24b

	0 1 2 3 _ _ _ _
TOTLA	1 0 1 1 0 0 0 0
TABLA	0 0 1 0 0 0 0 0
TABLA+1	0 0 0 1 0 0 0 0
TABLA+2	0 0 0 0 0 0 0 0
TABLA+3	1 0 0 0 0 0 0 0
TOTEA	1 0 1 0 0 0 0 0
TABEA	1 0 0 0 0 0 0 0
TABEA+1	0 0 1 0 0 0 0 0
TABEA+2	0 0 0 0 0 0 0 0
TABEA+3	0 0 0 0 0 0 0 0

fig.24c

	0 1 2 3 _ _ _ _
TOTLA	1 0 1 1 0 0 0 0
TABLA	0 0 1 0 0 0 0 0
TABLA+1	0 0 0 1 0 0 0 0
TABLA+2	1 0 0 0 0 0 0 0
TABLA+3	1 0 1 0 0 0 0 0
TOTEA	1 0 0 0 0 0 0 0
TABEA	0 0 1 0 0 0 0 0
TABEA+1	0 0 0 0 0 0 0 0
TABEA+2	0 0 0 0 0 0 0 0
TABEA+3	0 0 0 0 0 0 0 0

fig.24d

Eerst worden dan de liftdetectoren getest. Omdat alleen LD1 actief is, krijgt BUFF de inhoud 01000000_2.
De geheugen-pointer wordt gevuld met het adres van TOTLA en subroutine "RESTB" wordt aangeroepen. Er blijkt dan, dat in TOTLA een aanvraag voor de eerste etage staat.
Deze wordt dan gereset (fig.24b).
Daarna wordt getest op welk adres in de tabel deze aanvraag zich bevindt. In dit geval blijkt dat adres TABLA+2 te zijn. Deze aanvraag wordt dan gereset (fig.24c).

Dan moeten de aanvragen, die hierna in de tabel voor de liftaanvragen zijn opgeslagen, 1 plaats worden opgeschoven. Door de oplossing van fig.22b worden echter de inhouden van alle locaties opgeschoven, totdat een inhoud 0 wordt gedetecteerd (laatste testblok van fig.22b). Doordat TOTEA direct na TABLA+3 staat, wordt in dit geval tevens de gehele tabel voor de etage-aanvragen meegeschoven (fig.24d).
Deze fout wordt veroorzaakt door het feit, dat aan het einde van de tabel voor de liftaanvragen, dus na adres TABLA+3, zich niet een locatie met inhoud 0 bevindt.
Immers in alle 4 locaties was een aanvraag opgeslagen en in de hieropvolgende locatie, dus TOTEA, staat het totaal aan etage-aanvragen.

Deze fout is op te heffen, door tussen TABLA+3 en TOTEA een locatie te reserveren, waarvan de inhoud altijd 0 is (fig.25). Hetzelfde moeten we doen voor de locatie, die na TABEA+3 komt, zodat ook het einde van de tabel voor de etage-aanvragen kan worden gedetecteerd.

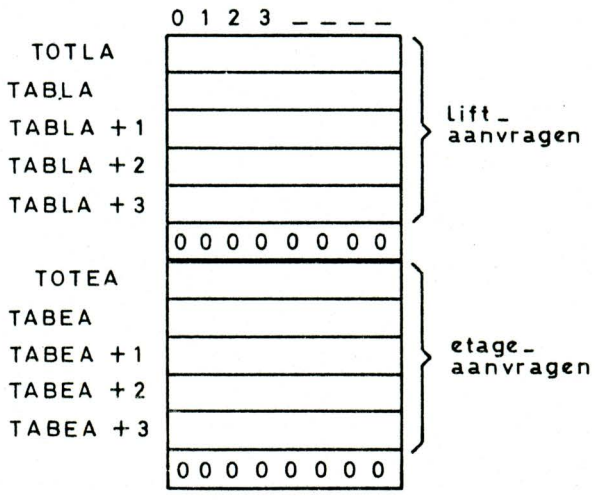


fig.25

f. Subroutine "WACHT"

Het doel van deze subroutine is het genereren van een wachttijd van 5 seconden. Meestal realiseren we een software-wachtlus door de inhoud van een register of registerpaar zolang met 1 te verlagen, totdat deze 0 is geworden. Met een registerpaar is op deze wijze een maximale tijd van ca. 0,5s te bereiken (zie b.v. de monitor-subroutine "DELAY" van de SDK 85). Om 5s te verkrijgen moet deze lus 10 maal worden uitgevoerd. Zo ontstaat het gedetailleerd stroomdiagram van fig.26. Voor de hierin gebruikte registers mag u natuurlijk andere kiezen.

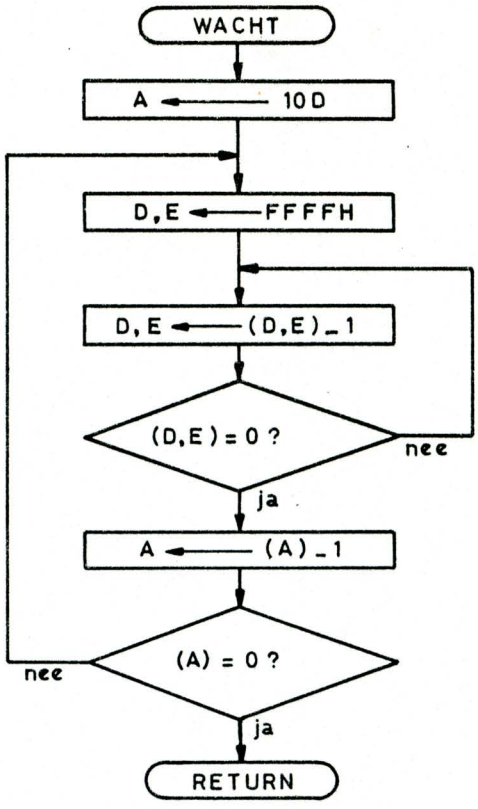


fig.26

```
                        ORG   2800H
                TOTLA   EQU   2000H      ;TOTAAL LIFTAANVRAGEN
                TOTEA   EQU   2006H      ;TOTAAL ETAGE-AANVRAGEN
                HOOG    EQU   80H        ;LIHO ACTIEF
                LAAG    EQU   40H        ;LILA ACTIEF
                LSTOP   EQU   01H        ;STOP LIFT EN OPEN DEUR
                SLLD    EQU   00H        ;SLUIT LIFTDEUR
                TILO    EQU   04H        ;LOW ORDER BYTE CLR
                TIHI    EQU   6FH        ;HIGH ORDER BYTE CLR
                STOP    EQU   40H        ;STOP TIMER COMMAND
                START   EQU   C0H        ;START TIMER COMMAND
                EN6.5   EQU   1DH        ;ENABLE TIMER INTERRUPT
2800  3E 03    INIT:   MVI   A,03H
2802  D3 20            OUT   20H        ;POORT 21H = OUTPUT
2804  32 FF 20        STA   20FFH      ;POORT 22H = OUTPUT
2807  AF              XRA   A
2808  D3 02            OUT   02H        ;POORT 00H = INPUT
280A  D3 03            OUT   03H        ;POORT 01H = INPUT
280C  31 C2 20         LXI   SP,20C2H   ;INIT STACKPOINTER
280F  3E 04            MVI   A,TILO     ;INIT
2811  D3 2C            OUT   2CH        ;  COUNT
2813  3E 6F            MVI   A,TIHI     ;     LENGTH
2815  D3 2D            OUT   2DH        ;       REGISTER
2817  3E 1D            MVI   A,EN6.5    ;INIT
2819  30              SIM              ;  INT. MASKS
281A  DB 01            IN    01H
281C  E6 80            ANI   80H        ;LIFT OP ETAGE 0?
281E  C2 36 28         JNZ   OPEN       ;SPRING BIJ JA
2821  AF              XRA   A          ;SLUIT
2822  D3 22            OUT   22H        ;  LIFTDEUR
2824  DB 01    DICHT: IN    01H        ;WACHT TOT
2826  E6 01            ANI   01H        ;  LIFT DEUR IS
2828  CA 24 28         JZ    DICHT      ;     GESLOTEN
282B  3E 40            MVI   A,LAAG     ;START
282D  D3 22            OUT   22H        ;  LIFT
282F  DB 01    LDNUL: IN    01H        ;WACHT TOT
2831  E6 80            ANI   80H        ;  ETAGE 0
2833  CA 2F 28         JZ    LDNUL      ;     IS BEREIKT
2836  3E 01    OPEN:  MVI   A,LSTOP    ;STOP LIFT EN
2838  D3 22            OUT   22H        ;  OPEN LIFTDEUR
283A  3E 0C            MVI   A,12D      ;RESET TABEL
283C  21 00 20         LXI   H,TOTLA
283F  36 00    TBRES: MVI   M,00H
2841  23              INX   H
2842  3D              DCR   A
2843  C2 3F 28         JNZ   TBRES
2846  3E 80            MVI   A,80H      ;STUUR LIFTLAMPEN
2848  D3 21            OUT   21H        ;  EN ETAGELAMPEN
284A  FB              EI
284B  21 00 20  ZOEK:  LXI   H,TOTLA    ;IS ER EEN
284E  3E 00            MVI   A,00H      ;  GEACCEPTEERDE
2850  B6              ORA   M          ;     LIFTAANVRAAG?
2851  C2 5B 28         JNZ   FOUND      ;SPRING BIJ JA
2854  21 06 20         LXI   H,TOTEA    ;IS ER EEN
2857  B6              ORA   M          ;  ETAGE-AANVRAAG?
2858  C2 4B 28         JZ    ZOEK       ;SPRING BIJ NEE
285B  23        FOUND: INX   H          ;AANVRAAG
285C  46              MOV   B,M        ;  NAAR REG
285D  DB 01    MOVE:  IN    01H        ;TEST
285F  E6 F0            ANI   F0H        ;  LIFTDETECTOREN
2861  B8              CMP   B          ;(REG)=(A)?
2862  C2 6B 28         JNZ   CLOSE      ;NEE: START LIFT
2865  CD BE 28         CALL  RESET      ;JA: RESET DEZE AANVRAAG
2868  C3 4B 28         JMP   ZOEK       ;     EN ZOEK DE VOLGENDE
286B  3E 00    CLOSE: MVI   A,SLLD     ;SLUIT DE
286D  D3 22            OUT   22H        ;  LIFTDEUR
286F  DB 01    NOCLO: IN    01H        ;WACHT TOT
2871  E6 01            ANI   01H        ;  LIFTDEUR IS
2873  CA 6F 28         JZ    NOCLO      ;     GESLOTEN
2876  DB 01            IN    01H        ;TEST LIFT-
2878  E6 F0            ANI   F0H        ;  DETECTOREN
287A  B8              CMP   B          ;(REG)>(A)?
287B  3E 80            MVI   A,HOOG     ;LIHO ACTIEF
287D  DA 82 28         JC    STYG       ;SPRING ALS (REG)>(A)
2880  3E 40            MVI   A,LAAG     ;LILA ACTIEF
2882  D3 22    STYG:  OUT   22H        ;START LIFT
2884  DB 01    DETEC: IN    01H        ;WACHT TOT
2886  E6 F0            ANI   F0H        ;  ETAGE IS
2888  C2 84 28         JNZ   DETEC      ;     VERLATEN
288B  DB 01    DET1:  IN    01H        ;WACHT TOT
288D  E6 F0            ANI   F0H        ;  VOLGENDE ETAGE
288F  CA 8B 28         JZ    DET1       ;     IS BEREIKT
2892  57              MOV   D,A        ;STUUR
2893  DB 21            IN    21H        ; LIFTLAMPEN
2895  E6 0F            ANI   0FH        ;  MAAR LAAT
2897  82              ADD   D          ;    ETAGELAMPEN
2898  D3 21            OUT   21H        ;     ONGEWIJZIGD
289A  21 00 20         LXI   H,TOTLA    ;IS ER EEN
289D  7A              MOV   A,D        ;  LIFTAANVRAAG
289E  A6              ANA   M          ;    VOOR DEZE ETAGE?
289F  C2 AA 28         JNZ   REM        ;  SPRING BIJ JA
28A2  21 06 20         LXI   H,TOTEA    ;IS ER EEN
28A5  7A              MOV   A,D        ;  ETAGE-AANVRAAG?
28A6  A6              ANA   M          ;     VOOR DEZE ETAGE?
28A7  CA 84 28         JZ    DETEC      ;  SPRING BIJ NEE
```

```
28AA  3E 01     REM:   MVI  A,LSTOP   ;STOP LIFT EN
28AC  D3 22            OUT  22H       ;  OPEN LIFTDEUR
28AE  DB 01     NOPEN: IN   01H       ;WACHT TOT
28B0  E6 01            ANI  01H       ;  LIFTDEUR IS
28B2  C2 AE 28         JNZ  NOPEN     ;     GEOPEND
28B5  CD BE 28         CALL RESET     ;RESET AANVRAAG
28B8  CD EF 28         CALL WACHT     ;WACHT 5 SECONDEN
28BB  C3 4B 28         JMP  ZOEK
28BE  DB 01     RESET: IN   01H       ;TEST
28C0  E6 F0            ANI  F0H       ;  LIFTDETECTOREN
28C2  4F               MOV  C,A
28C3  21 00 20         LXI  H,TOTLA   ;RESET
28C6  CD D0 28         CALL RESTB     ;  LIFTAANVRAAG
28C9  21 06 20         LXI  H,TOTEA   ;RESET
28CC  CD D0 28         CALL RESTB     ;  ETAGE-AANVRAAG
28CF  C9               RET
28D0  F3        RESTB: DI
28D1  79               MOV  A,C       ;STAAT DEZE AANVRAAG
28D2  A6               ANA  M         ;  IN HET TOTAAL?
28D3  CA ED 28         JZ   TERUG     ;SPRING BIJ NEE
28D6  7E               MOV  A,M       ;RESET DEZE
28D7  A9               XRA  C         ;  AANVRAAG
28D8  77               MOV  M,A       ;     IN HET
28D9  79               MOV  A,C       ;       TOTAAL
28DA  23        NEXT:  INX  H         ;ZOEK DEZE
28DB  BE               CMP  M         ;  AANVRAAG IN
28DC  C2 DA 28         JNZ  NEXT      ;     DE TABEL
28DF  AF        SHIFT: XRA  A         ;RESET DEZE
28E0  77               MOV  M,A       ;  AANVRAAG IN
28E1  23               INX  H         ;     DE TABEL
28E2  BE               CMP  M
28E3  CA ED 28         JZ   TERUG
28E6  7E               MOV  A,M       ;SCHUIF ALLE
28E7  2B               DCX  H         ;  VOLGENDE
28E8  77               MOV  M,A       ;     AANVRAGEN
28E9  23               INX  H         ;       VOORUIT
28EA  C3 DF 28         JMP  SHIFT
28ED  FB        TERUG: EI
28EE  C9               RET
28EF  06 0A     WACHT: MVI  B,10D     ;DEZE SUBROUTINE
28F1  11 FF FF  LUS1:  LXI  D,FFFFH   ;GENEREERT EEN
28F4  1B        LUS2:  DCX  D         ;WACHTTIJD VAN
28F5  7A               MOV  A,D       ;CA. 5 SECONDEN
28F6  B3               ORA  E
28F7  C2 F4 28         JNZ  LUS2
28FA  05               DCR  B
28FB  C2 F1 28         JNZ  LUS1
28FE  C9               RET
                       ORG  2010H
2010  F5        TMINT: PUSH PSW
2011  E5               PUSH H
2012  3E 40            MVI  A,STOP    ;STOP
2014  D3 28            OUT  28H       ;  TIMER
2016  3E C0            MVI  A,START   ;START
2018  D3 28            OUT  28H       ;  TIMER
201A  DB 00            IN   00H       ;IS ER EEN
201C  E6 F0            ANI  F0H       ;  LIFTAANVRAAG?
201E  CA 27 20         JZ   ETSCH     ;SPRING BIJ NEE
2021  21 00 20         LXI  H,TOTLA   ;PLAATS LIFTAANVRAAG
2024  CD 4D 20         CALL ACCEP     ;  IN DE TABEL
2027  DB 00     ETSCH: IN   00H
2029  07               RLC
202A  07               RLC            ;IS ER EEN
202B  07               RLC            ;  ETAGE-AANVRAAG?
202C  07               RLC
202D  E6 F0            ANI  F0H
202F  CA 38 20         JZ   ETLMP     ;SPRING BIJ NEE
2032  21 06 20         LXI  H,TOTEA   ;PLAATS ETAGE-AANVRAAG
2035  CD 4D 20         CALL ACCEP     ;  IN DE TABEL
2038  3A 06 20 ETLMP: LDA  TOTEA
203B  0F               RRC
203C  0F               RRC
203D  0F               RRC            ;STUUR
203E  0F               RRC            ; ETAGELAMPEN
203F  E6 0F            ANI  0FH       ;  MAAR LAAT
2041  6F               MOV  L,A       ;   LIFTLAMPEN
2042  DB 21            IN   21H       ;    ONGEWIJZIGD
2044  E6 F0            ANI  F0H
2046  85               ADD  L
2047  D3 21            OUT  21H
2049  E1               POP  H
204A  F1               POP  PSW
204B  FB               EI
204C  C9               RET
204D  C5        ACCEP: PUSH B
204E  47               MOV  B,A       ;STAAT AANVRAAG
204F  A6               ANA  M         ;  REEDS IN TABEL?
2050  C2 5D 20         JNZ  REEDS     ;SPRING BIJ JA
2053  78               MOV  A,B       ;PLAATS
2054  B6               ORA  M         ;  AANVRAAG
2055  77               MOV  M,A       ;     IN TOTAAL
2056  23        BEZET: INX  H         ;ZOEK
2057  7E               MOV  A,M       ;  EERSTVOLGENDE
2058  A7               ANA  A         ;     VRIJE PLAATS
2059  C2 56 20         JNZ  BEZET     ;       IN DE TABEL
205C  70               MOV  M,B       ;SLA DE AANVRAAG HIER OP
205D  C1        REEDS: POP  B
205E  C9               RET
                       ORG  20C8H     ;{20CEH}
20C8  C3 10 20         JMP  TMINT
                       END
```

Programmadeel "INIT"

De initialisatie dient om
1. het microcomputersysteem op de juiste wijze te programmeren,
2. een aantal variabelen, die in het programma voorkomen, een beginwaarde te geven.

De initialisatie van de microcomputer heeft evenals in voorgaande lessen betrekking op de I/O-module, de timer, de stackpointer en de interrupt controller.

Voor het programma is als eis gesteld, dat direct na het starten de lift naar de begane grond moet worden gestuurd (uitgangssituatie) en dat alle eventuele lift- en etage-aanvragen worden gereset.

Fig.27 is het stroomdiagram voor dit tweede deel van de initialisatie. Als de lift zich niet op de begane grond bevindt, wordt deze daar eerst naar toe gestuurd. Vervolgens wordt de gehele aanvragentabel met 0 gevuld.

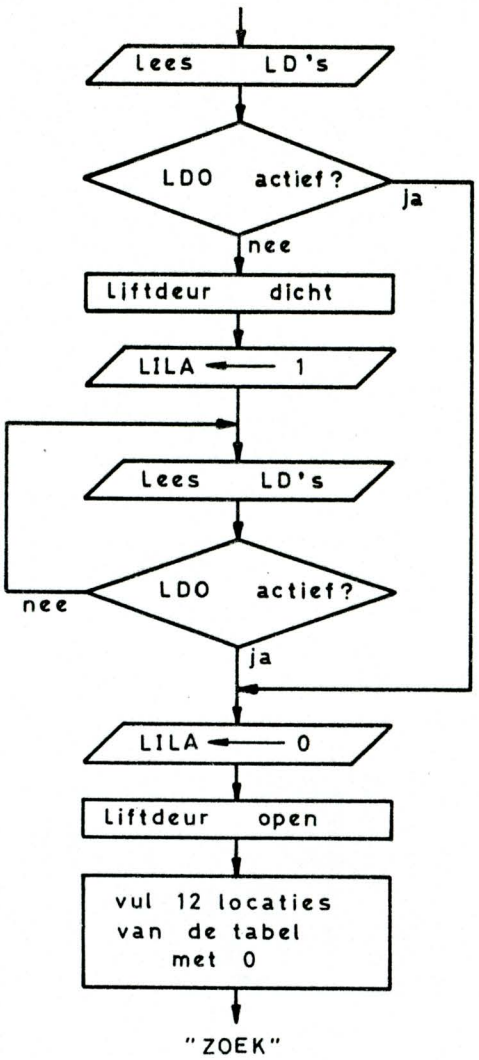


fig.27

6. PROGRAMMA

Uit de stroomdiagrammen van de figuren 11, 12, 14, 16, 18, 20, 22 26 en 27 kunnen we nu een programma schrijven, mits er enkele machinegerichte details worden vastgelegd.
Uitgaande van de eigenschappen van de SDK 85 maken we b.v. de volgende keus. De functies van de I/O-lijnen zijn in fig. 28 weergegeven.

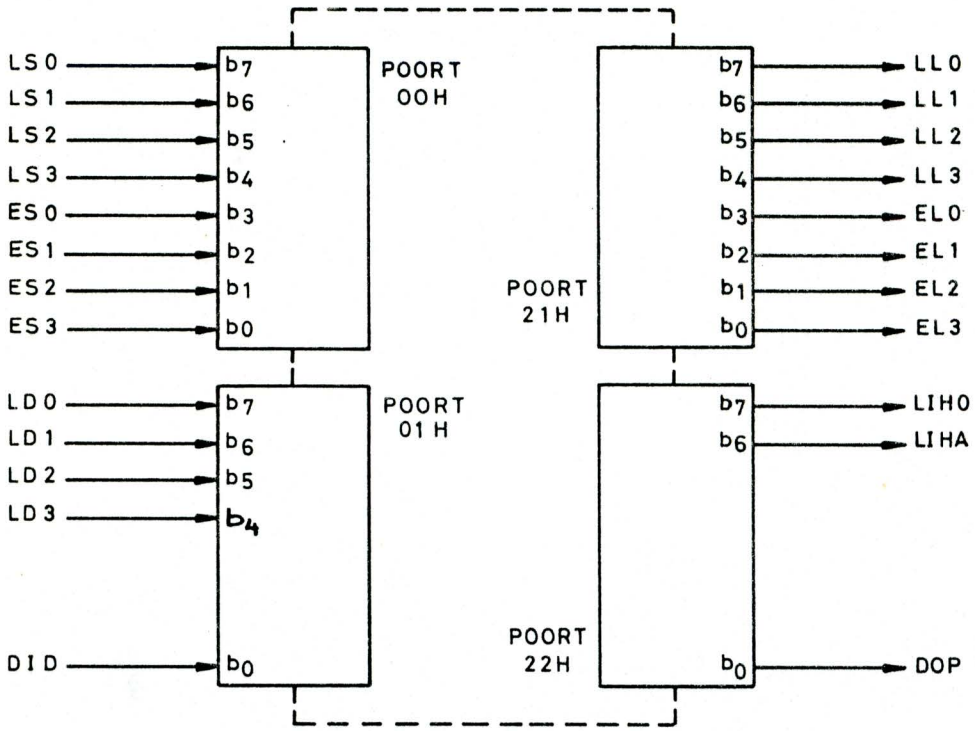


fig.28

Merk op dat zowel de liftlampen als de etagelampen op één outputpoort zijn aangesloten. We zullen dan moeten zorgen dat als de liftlampen worden aangestuurd, de naar de etagelampen gezonden bitcombinatie niet wordt gewijzigd (en omgekeerd).
Hierbij kunnen we gebruik maken van het feit, dat data, die via een output-poort van de 8155 is uitgevoerd, weer m.b.v. een input-instructie is in te lezen (zie paragraaf 11c van de les "Programmeerbare chips").

Voor de in stroomdiagrammen optredende symbolische namen kiezen we
TOTLA = 2000_{16} (hiermee liggen de overige adressen in de tabel ook vast).
PNTR = registerpaar H,L.
REG = register B.
BUFF = register C.

Het hoofdprogramma begint op adres 2800_{16} in de expansion RAM.

Het programma wordt dan zoals in fig.29 is weergegeven. Let in dit programma vooral op

a. het gebruik van DI- en EI-instructies in de subroutine "RESTB". Dit is gedaan om te voorkomen dat de interrupt service routine "TMINT" de tabelinhouden gaat wijzigen, op het moment dat de tabel d.m.v. "RESTB" wordt gereorganiseerd.
b. de wijze, waarop met AND- en OR-functies een aanvraag aan de inhoud van TOTLA resp. TOTEA wordt toegevoegd (de aanvragen die er al stonden mogen natuurlijk niet verloren gaan).
c. het feit, dat in subroutine "WACHT" niet de accumulator (zie fig.26) maar register B met 10_{10} wordt gevuld.
   Dit is noodzakelijk, omdat de accumulator is betrokken bij het testen van de inhoud van registerpaar D,E.

Bij het testen van uw programma en dat in fig.29, kunt u de liftinstallatie simuleren m.b.v. de LED's en de schakelaars op de uitbreidingsprint van de SDK 85.