Uitbreiding van de pocketstem en ondersteuningsprogrammatuur

Instituut voor Perceptie Onderzoek
Postbus 513, 5600 MB EINDHOVEN

FLCW/flcw
01.12.1988

Rapport no. 679

Uitbreiding van de pocketstem en ondersteuningsprogrammatuur

F.L.C. Winthagen

FACULTEIT DER ELEKTROTECHNIEK
TECHNISCHE UNIVERSITEIT
EINDHOVEN
VAKGROEP MEDISCHE ELEKTROTECHNIEK EME

UITBREIDING VAN DE POCKETSTEM EN ONDERSTEUNINGSPROGRAMMATUUR.

door F.L.C. Winthagen

Verslag van het stagewerk
uitgevoerd van 1-9-'87 tot 1-3-'88.

Onder begeleiding van ir. R.P. Waterham,
Mede begeleid door ir. W.H. Leliveld
H.J.M. Ossevoort
ir. R.W.M. Mathijssen

1 Samenvatting.

Dit stageverslag handelt over een uitbreiding van de Pocketstem.
De Pocketstem is een hulpmiddel voor spraakgehandicapten; het kan 28 van te voren ingeprogrammeerde zinnen uitspreken.

Allereerst zijn een aantal wetenschappelijke tijdschriften uit de periode van 1982 tot 1987 onderzocht op relevante artikelen.

Door toepassing van een andere spraakchip in de Pocketstem kan de spreeksnelheid in kleine stappen gevarieerd worden. Deze snelheidsvariaties worden in dit verslag besproken. Tevens heeft hierover een evaluatie plaats gevonden, die ook besproken wordt.

Verder is de Pocketstem uitgerust met het alfabet en spelalfabet. De hard- en software wijzigingen die hiervoor nodig zijn komen in dit verslag aan de orde. Op een APPLE IIe computer is reeds BASIC programmatuur ontwikkeld waarmee een gebruiker een EPROM kan programmeren met 28 door hem gekozen zinnen. Deze programmatuur wordt uitgebreid. Nu kan ook de data van alfabet en spelalfabet automatisch in een EPROM geprogrammeerd worden.

De hoofdconclusie is dat de gewijzigde Pocketstem goed functioneert.
Het alfabet-onderdeel zal nog aan een praktijk evaluatie onderworpen moeten worden. De APPLE programmatuur bleek ook tijdens het gebruik goed te functioneren.

Inhoud

2 Inleiding.

In het kader van het project "Ergonomische communicatie apparatuur ten behoeve van gehandicapten" wordt in een interfaculteitswerkgroep van de Technische Universiteit Eindhoven en het Instituut voor Perceptie Onderzoek o.a gewerkt aan communicatie hulpmiddelen voor spraakgehandicapten.
In het kader hiervan werd de "Compacte Spraakhulp I" (CSH I) ontwikkeld. Dit is een klein draagbaar apparaat, dat twintig van te voren ingeprogrammeerde zinnen kan reproduceren. Een zin kan geselecteerd worden door het indrukken van de bijbehorende toets. Vervolgens verschijnt de tekst van de zin op het L.C.D. display. De zin kan nu ten gehore gebracht worden door te drukken op de "spreek" toets.
Uit de praktijkevaluatie van de CSH I zijn een aantal eisen naar voren gekomen voor een tweede model, de CSH II. Dit apparaat wordt "Pocketstem" [1] genoemd. De vier grootste verschillen tussen de Pocketstem en zijn voorganger zijn:

- de Pocketstem is veel kleiner en lichter dan de CSH I,
- de Pocketstem heeft geen "spreek" knop; een zin wordt meteen uitgesproken als hij geselecteerd wordt,
- de Pocketstem heeft een bestand van 28 zinnen; de CSH I 60 zinnen (3x20),
- de Pocketstem heeft geen display; op een inlegkaart zijn symbolen geplakt die de betekenis van de zin weergeven.

De CSH I [14] en de Pocketstem [1] zijn beide uitgerust met een MEA8000 spraakchip van Philips. Inmiddels is er door Philips een opvolger van deze spraakchip geïntroduceerd; de PCF8200.
De Pocketstem is inmiddels ook uitgerust met deze PCF8200 spraakchip, hetgeen resulteert in een betere geluidskwaliteit en de mogelijkheid om een vrouwenstem te creëren. Met de MEA spraakchip was het niet mogelijk om een vrouwenstem te creëren. Verschil van deze nieuwe spraakchip is ook dat de spraaksnelheid beter gevarieerd kan worden dan bij de MEA8000. Bij de CSH I is, als een zin niet verstaan of begrepen wordt, de mogelijkheid aanwezig om die zin dan met een andere zinsmelodie uit te spreken (ook wel tweede intonatie genoemd). Dit heeft tot gevolg dat er twee spraakdatablokken van elk 14kbyte in het geheugen zijn. Bij de PCF8200 is gebleken

dat per zin meer geheugenruimte nodig is. De kans bestaat dus dat de beschikbare 14kbyte overschreden wordt. Om dat te voorkomen wordt de tweede intonatie weggelaten waardoor een hoeveelheid geheugenruimte vrijkomt, die voor andere doeleinden gebruikt kan worden.
De mogelijkheid van het variëren van de spraaksnelheid wordt nu toegepast in een versie van de Pocketstem en zal zijn nut moeten bewijzen in de praktijk. Als eerste onderdeel van mijn stage vermeld ik het literatuuronderzoek. In het kader van het bibliotheekpraktikum, en daar ik het interessant vond meer te weten te komen van dit onderwerp, heb ik literatuur vanaf 1982 tot 1987 doorzocht naar artikelen die te maken hebben met communicatiehulpmiddelen voor gehandicapten. De resultaten hiervan staan in hoofdstuk 3.
Het tweede onderdeel van mijn stage, die in dit verslag beschreven wordt, is het realiseren van verschillende snelheden van uitspreken bij de PCF8200 Pocketstem. In hoofdstuk 4 wordt beschreven welke software wijzigingen in het besturingsprogramma dit met zich mee brengt. Verder wordt deze versie van de Pocketstem in de praktijk getest op een aantal proefpersonen.
Het laatste onderdeel van mijn stage heeft betrekking op de uitdrukkingsbeperking van de Pocketstem. Het is met de Pocketstem mogelijk 28 zinnen uit te spreken, hetgeen in de praktijk beperkingen met zich mee kan brengen. Een poging om de gebruiker meer uitdrukkingsvrijheid te geven is om de Pocketstem naast de set van 28 zinnen uit te rusten met het alfabet, zodat de gebruiker door het spellen van woorden ook iets duidelijk kan maken. Dit is nu mogelijk geworden daar nu geheugenruimte vrij is gekomen door het weglaten van de tweede intonatie. De praktijk zal uitwijzen of het nuttig is. Verder moet dit alfabet met de bestaande APPLE programmatuur (Applesoft Basic) [2,3] in een EPROM geprogrammeerd worden. De hard- en software wijzigingen en uitbreidingen die hiervoor nodig waren zijn beschreven in hoofdstuk 5.

3 Literatuuronderzoek.

In dit hoofdstuk wordt het literatuuronderzoek beschreven, dat gedaan is in het kader van het bibliotheekpraktikum. Het resultaat hiervan is een aantal artikelen uit wetenschappelijke tijdschriften uit de periode van 1982 tot 1987.
Uit de meeste relevante artikelen, gepubliceerd in de periode '82 tot '87, blijkt dat de eisen die aan een communicatie hulpmiddel voor gehandicapten gesteld worden zeer hoog zijn. Waar een niet-gehandicapte met een standaard toetsenbord kan volstaan moet voor een gehandicapte een oplossing gevonden worden, die veelal niet geschikt is voor een grote groep [6,7], daar iedere gehandicapte door zijn of haar handicap specifieke eisen stelt aan een hulpmiddel.
De opmars van de computer is ook hier te bespeuren, omdat de computer gehandicapten kan helpen onafhankelijk te worden van anderen en zich (met computer) beter te kunnen uitdrukken. Op een APPLE computer is een systeem ontwikkeld, waarmee de gebruiker met het toetsenbord de beschikking heeft over 5000 woorden, opgeslagen op hard-disk, die hij kan combineren tot zinnen. Vervolgens kan hij ze laten uitspreken.[10]
Een ander artikel beschrijft de situatie van een gehandicapte die alleen zijn rechterduim kan bewegen. Voor die patiënt is een systeem ontworpen dat bestuurd wordt door twee micro-switches, die de patiënt met zijn duim kan bedienen. Deze micro-switches zijn via een interface aangesloten op een P.C., welke de morse codes afkomstig van de twee schakelaars decodeert in woorden, en deze vervolgens op een scherm weergeeft.[11]
Een hulpmiddel dat op een totaal ander principe gebaseerd is, is het volgende: Met behulp van electrodes die op nek en hoofd geplaatst worden, kunnen electrische signalen opgevangen worden van nek- en hoofdspieren, waarmee spraak gegenereerd kan worden.[12]
"Minspeak" is de naam van een codeersysteem voor een toetsenbord met een gering aantal toetsen. Alle toetsen van dit toetsenbord zijn voorzien van symbolen. Een gehandicapte kan met een Minspeak toetsenbord met een gering aantal aanslagen een zin laten uitspreken. Bij de "Light- of Touchtalker" (een communicatiehulpmiddel dat gebruik maakt van een Minspeak toetsenbord) werd gepoogd een aantal belangrijke punten bij communicatie te realiseren, met name de snelheid waarmee de gehandicapte kan communiceren. In de praktijk is gebleken dat het een goed hulpmiddel is, dat ook bij een groot vocabulaire goed functioneert.[4]
"SpeechPAC" is een soortgelijk hulpmiddel waar naast de snelheid ook gelet

is op de aansluitmogelijkheden op andere apparaten. Het apparaat kan namelijk probleemloos op andere (APPLE en Franklin) computers worden aangesloten.[5]
Voor een gedetailleerdere uiteenzetting van deze onderwerpen verwijs ik naar de literatuurlijst, die achterin het verslag is opgenomen.

4 Snelheidsvariaties in de uitspraak.

4.1 Inleiding.

De beide spraakchips (MEA8000 en PCF8200) werken beide volgens hetzelfde principe ("bron-filter model") dat uitvoerig beschreven staat in [15]: (zie fig. 4.1)

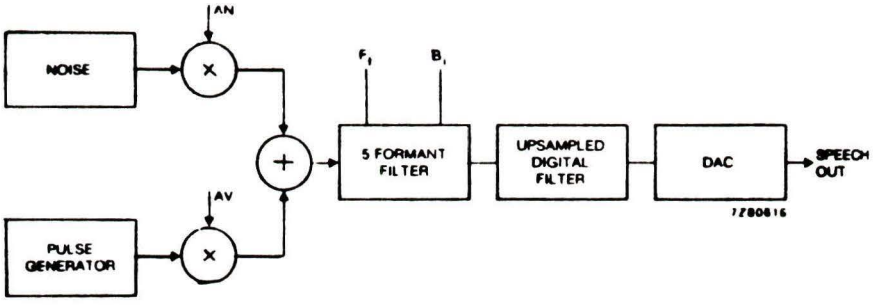


Fig. 4.1: Principe van de spraakchips.

Beide spraakchips hebben twee bronnen met daarachter een set van filters. Door nu filterparameters in te stellen kan het mond-keelkanaal nagebootst worden. Of een bepaalde klank stemhebbend of stemloos is wordt bepaald door de bron. De informatie van de filters en de bron is opgeslagen in een blok van 5 bytes (frame) bij de PCF, 4 bytes bij de MEA. Elke frame verzorgt spraak gedurende een bepaalde tijd. Voordat de PCF echter spraakdata kan ontvangen moet hij eerst een "DAC amplitude factor" (luidsterkte van de spraak), een "start pitch" (start toonhoogte) en een "Command write" (spreeksnelheid en geslacht van de stem) ontvangen hebben.
De spreeksnelheid kan nu op twee manieren veranderd worden:

- Elk frame bevat twee bits FD1 en FD0 (FD=frame duration). Door in elk frame nu deze bits te wijzigen kan de spreeksnelheid een andere waarde krijgen.
- Het "Command write" bevat twee bits FS1 en FS0 (FS=framespeed). Door deze twee bits te wijzigen kan de spreeksnelheid ook een andere waarde krijgen.

Als deze gewijzigde snelheid bijvoorbeeld hoger is, dan zou de gebruiker deze mogelijkheid kunnen gebruiken om een verzoek wat meer kracht bij te zetten. Een lagere (gewijzigde) snelheid zou de gebruiker bijvoorbeeld kunnen hanteren om iets wat niet goed is verstaan te verduidelijken. Door het vervangen van de MEA8000 spraakchip door de PCF8200, wordt het mogelijk gemaakt ook kleine snelheidsvariaties in de uitspraak te creëren. Bij de MEA8000 bestond als enige snelheidsvariant de mogelijkheid om de normale snelheid te verdubbelen of te halveren (afhankelijk van waar je begint).
In de praktijk blijkt dan dat de zinnen zodanig misvormd worden, dat ze in de meeste gevallen niet meer verstaanbaar zijn. Van deze mogelijkheid is dan ook geen gebruik gemaakt. De snelheidsvariaties van de beide spraakchips zijn weergegeven in tabel 4.1, waarbij 100% de "normale" snelheid (standaard frame duur) weergeeft en de overige snelheden aan deze gerelateerd worden.

Tabel 4.1: Spraaksnelheden van MEA8000 en PCF8200.

MEA8000	PCF8200	PCF8200	PCF8200	PCF8200	
50%	145%	123%	100%	72%	FD
100%	72%	61%	50%	36%	FD
200%	49%	41%	33%	24%	FD
400%	29%	25%	20%	15%	FD
	FS	FS	FS	FS	

Uit deze tabel blijkt duidelijk dat met de PCF8200 spraakchip veel meer mogelijkheden bestaan tot snelheidsvariaties dan met de MEA8000.
De snelheidsvariaties worden als volgt gerealiseerd:
De eerste keer dat een zin wordt uitgesproken is de snelheid "normaal" (100%); druk je daarna nog eens op **dezelfde** knop, dan zal de zin in de andere snelheid ten gehore gebracht worden.

4.2 Software wijzigingen.

4.2.1 Evaluatie versie.

Het data formaat van de PCF8200 is als volgt opgebouwd [15]:

- de eerste twee bytes die naar de PCF verstuurd worden zijn de "DAC amplitude factor" en een "start pitch byte" die hier verder niet van belang zijn,
- de volgende byte die naar de PCF gestuurd wordt is een vaste Command write byte". Dit is een teken voor de PCF dat de volgende byte een "Control byte" is,
- deze volgende byte (control byte) geeft informatie over o.a. de stem (mannelijke of vrouwelijke) en de spraaksnelheid,
- de volgende bytes worden in blokken van vijf naar de PCF verstuurd en een dergelijk blok (frame) bevat informatie over de spraak gedurende een bepaalde tijd.

De control byte (die informatie geeft over stem en snelheid) is als volgt ingedeeld: (zie fig. 4.2)

0	0	stop	M/F	0	0	FS1	FS0
7	6	5	4	3	2	1	0

Fig. 4.2: De indeling van de control byte.

De bits 2, 3, 6 en 7 van deze byte zijn nul. Bit 5 (stop) is voor dit geval niet van belang (bit 5 is alleen van belang op het einde van een uit te spreken zin). De bits 1 en 0 bevatten de informatie over de spraaksnelheid. De spraaksnelheid als functie van FS1 en FS0 is weergegeven in tabel 4.2. (FS=framespeed)

Tabel 4.2: Spraaksnelheid als functie van FS1 en FS0.

FS1	FS0	Spraaksnelheid	Tijdsduur per frame (msec)
0	0	100%	12.8
0	1	123%	10.4
1	0	145%	8.8
1	1	73%	17.6

In deze tabel geeft 100% weer de "normale" snelheid aan.
De spraaksnelheid kan nu eenvoudig gewijzigd worden door de waarden van FS1 en FS0 te wijzigen. Bij de normale snelheid zijn FS1 en FS0 dus beide nul. Om de snelheid te veranderen moet FS1 en/of FS0 één gemaakt worden.
Een tweede mogelijkheid om de spraaksnelheid te veranderen, is de volgende: In elke blok van 5 bytes is informatie opgeslagen over de tijdsduur die elke frame duurt. Deze informatie is opgeslagen in de twee bits FD1 en FD0 (FD= Frame duration). De spraaksnelheid kan dus ook veranderd worden door in elk blok van vijf bytes, die naar de PCF verstuurd worden, FD1 en/of FD0 te veranderen. Bij de normale snelheid zijn FD1 en FD0 beide nul. Deze methode wordt wel toegepast bij één van de vier modellen bij de eerste evaluatie, doch zal daarna niet meer worden gebruikt.
De snelheid moet veranderd worden, als een zin voor de tweede keer achter elkaar uitgesproken wordt (de derde keer is de snelheid weer normaal, de vierde keer weer veranderd enz.).
In een flowchart ziet het geheel er uit als weergegeven in figuur 4.3.
In de procedure "chr7" [1] van het PCF besturingsprogramma wordt een ingedrukte toets vergeleken met de vorige. Zijn deze twee gelijk dan wordt het bit "secout" geset. Dit bit geeft aan dat een toets voor de tweede keer is ingedrukt; de spreeksnelheid wordt nu gewijzigd. Dit bit is het nulde bit van het "device status word". Het "device status word" is een byte in het RAM geheugen, waarin een aantal bits die informatie bevatten over o.a. spreeksnelheid. Was "secout" van de vorige uitspraak nog hoog, en is de nu ingedrukte toets weer hetzelfde (de toets is dus nu voor de derde keer ingedrukt), dan wordt secout gereset en de snelheid weer normaal.
In de procedure "sssl" [1] van het PCF besturingsprogramma wordt de control byte (in het programma ook wel command byte genoemd) naar de PCF gestuurd. Afhankelijk van de waarde van secout wordt de command byte onveranderd gelaten of veranderd. Voor de gewijzigde versies van de procedures "chr7" en "sssl" verwijs ik naar de bijlagen 1 en 2.

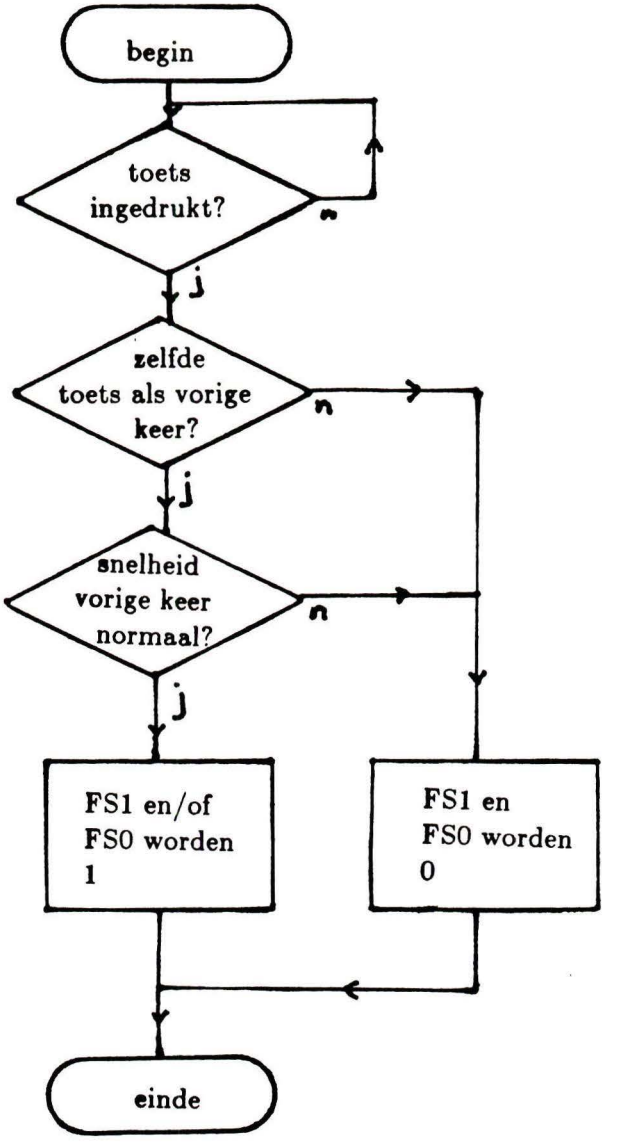


Fig. 4.3: Flowchart.

4.2.2 Evaluatie.

In het voorgaande is besproken hoe een zin de tweede keer met een andere snelheid kan worden uitgesproken. In deze paragraaf wordt uitgezocht door middel van evaluatie, welke tweede snelheid als de beste ervaren wordt.
Er worden vier varianten getest. Bij elk van deze vier is de snelheid waarmee een zin de **eerste** keer wordt uitgesproken hetzelfde. De frameduur bij de eerste uitspraak is dus bij alle vier de varianten 12.8 msec. Bij variant 1 is de tweede uitspraak langzamer dan de eerste uitspraak (frameduur 17.6 msec). Bij variant 2 is de tweede uitspraak nog langzamer (frameduur 20.8 msec). Bij variant 3 is de tweede uitspraak sneller dan de eerste uitspraak (frameduur 10.4 msec). Bij variant 4 is de tweede uitspraak nog sneller dan bij variant 3 (frameduur 8.8 msec).
Bij de evaluatie is als volgt te werk gegaan :
De proefpersoon krijgt iedere keer twee varianten te horen, met drie verschillende zinnen en kiest bij elke zin welke variant hij het best vindt (als algemene indruk). Dit wordt herhaald totdat alle combinaties van varianten gehoord zijn. In tabel 4.3 zijn horizontaal de proefpersonen (PP) uitgezet, verticaal de combinaties van varianten. In het veld staat die variant die de proefpersoon het best vindt.

Tabel 4.3: Resultaten van de evaluatie.

	PP1	PP2	PP3	PP4	PP5	PP6	PP7	PP8	PP9	PP10
(1,2)	1	1	1	2	1	1	1	1	1	1
(1,3)	3	3	3	3	3	3	3	3	3	3
(1,4)	1	1	1	4	4	1	1	4	1	1
(2,3)	3	3	2	3	3	3	3	3	3	3
(2,4)	4	4	2	2	4	4	2	4	4	2
(3,4)	3	3	3	3	3	3	3	3	3	3

Uit tabel 4.3 blijkt het volgende:

- versie 1 is 16 keer gekozen,
- versie 2 is 6 keer gekozen,
- versie 3 is 29 keer gekozen,
- versie 4 is 9 keer gekozen.

Hieruit kan geconcludeerd worden dat de derde variant als beste ervaren wordt gevolgd door de eerste variant. Uit de evaluatie is ook gebleken dat de varianten twee en vier als slecht ervaren worden. De eerste variant (langzame) zorgt soms voor verheldering als een zin slecht verstaan of niet begrepen is.
In de uiteindelijke versie is gekozen voor variant drie (tweede keer iets sneller). Om echter ook te voorzien in de gevallen dat een zin niet verstaan wordt (de eerste keer niet en de tweede keer ook niet), wordt nu ook een derde uitspraak ingebouwd, die langzamer is dan de eerste uitspraak.
Resumerend: Als je drie keer achter elkaar op één en dezelfde knop drukt, dan krijg je dezelfde zin de eerste keer normaal te horen (100%); de tweede keer sneller dan de eerste keer (123%); de derde keer langzamer dan de eerste keer (73%).

4.3 Uiteindelijke versie.

4.3.1 Wijzigingen.

In de uiteindelijke versie moet de microprocessor dus drie situaties onthouden. Hier zijn dus minimaal twee bits voor nodig: "secout" en "thrdout". Het "device status word" (DSW) ziet er nu als volgt uit (zie tabel 4.4):

Tabel 4.4: Het DSW.

bit	naam	omschrijving
0	secout	wordt geset bij tweede uitspraak
1	thrdout	wordt geset bij derde uitspraak

De bits 2 tot en met 7 van het DSW zijn weggelaten, daar ze hier niet van belang zijn. [1]
In de procedure "chr7" wordt niet alleen gekeken of de ingedrukte toets gelijk is aan de vorige, maar ook welke bits geset zijn (secout en thrdout).
In de procedure "sssl" wordt nu afhankelijk van welk bit geset is (secout, thrdout of geen van beide) een bepaald command byte naar de PCF gestuurd.
In de procedure "init", waar het DSW gereset wordt (wanneer een "power-up keyboard" signaal gegenereerd wordt, of wanneer de Pocketstem aangezet wordt) moet de informatie die opgeslagen is in de twee bits secout en thrdout bewaard blijven.

De flowchart van het geheel is weergegeven in fig. 4.4 .

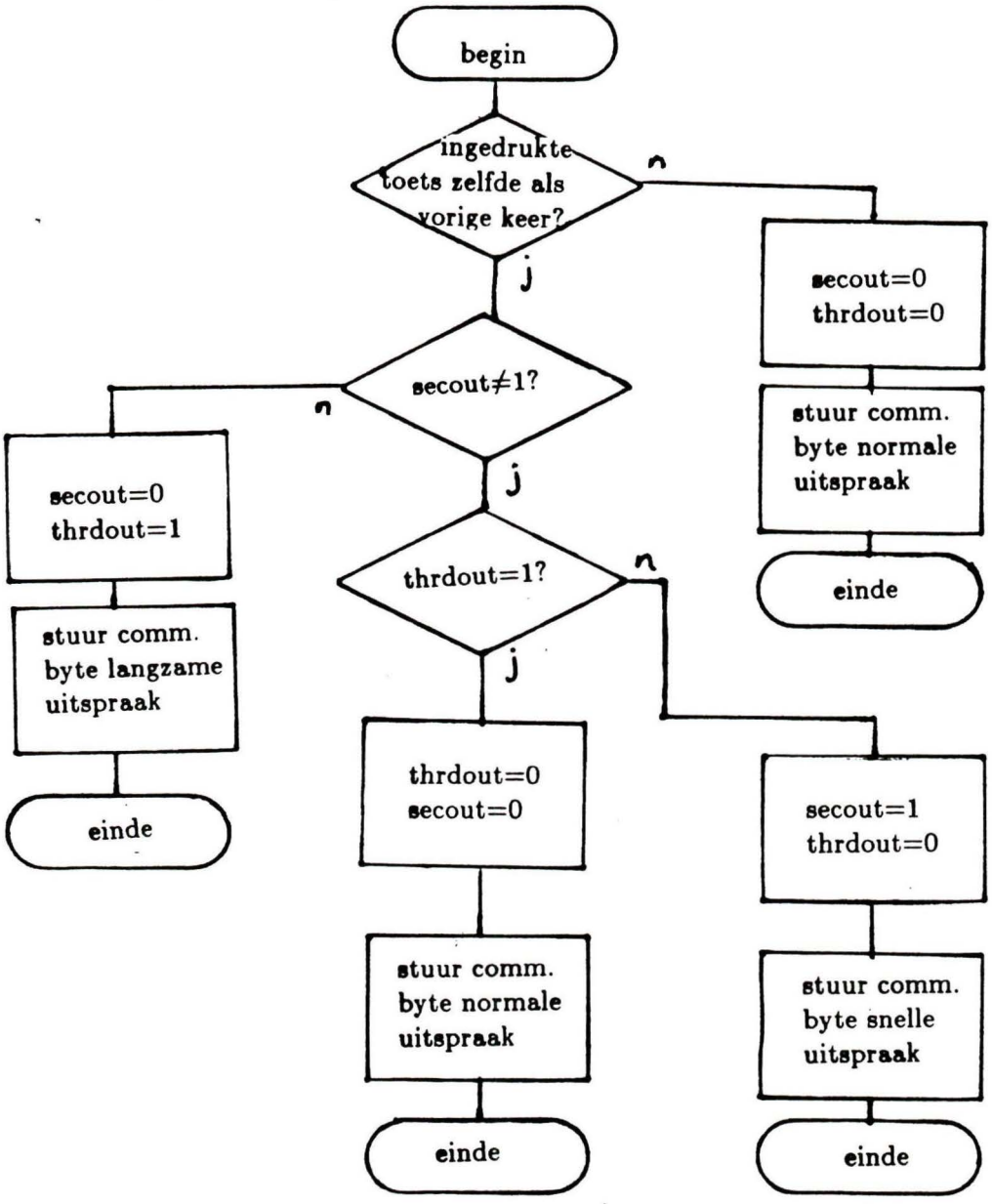


Fig. 4.4: Flowchart van de uiteindelijke versie.

Voor de assembler programma's van de uiteindelijke versie verwijs ik naar de bijlagen 3,4 en 5.

4.3.2 Evaluatie.

De hier beschreven evaluatie heeft betrekking op de uiteindelijke versie (3 snelheden). Het doel van deze evaluatie is om na te gaan of de derde uitspraak ooit voor verheldering zorgt. Derhalve worden bij de evaluatie zinnen gebruikt die niet voor de hand liggend zijn (bijv.: Ik wil in het vlinderbad). Bij zulke zinnen zou het namelijk best mogelijk zijn dat de zin pas verstaan wordt als hij langzaam uitgesproken wordt.
De evaluatie wordt nu als volgt uitgevoerd:
De proefpersoon krijgt vijf zinnen te horen. Elke zin krijgt hij drie keer achter elkaar te horen in de respectievelijke snelheden normaal, snel en langzaam. Bij elke van deze drie snelheden bij elk van de tien zinnen moet hij aangeven of hij de zin verstaan heeft. De evaluatie resultaten zijn weergegeven in tabel 4.5 . In de tabel staan horizontaal de proefpersonen (PP), verticaal de zinnummers en in het veld staat of de proefpersoon de zin met de betreffende snelheid verstaan heeft (J=ja, N=nee).

Tabel 4.5: Evaluatieresultaten.

	PP1 123	PP2 123	PP3 123	PP4 123	PP5 123	PP6 123	PP7 123	PP8 123	PP9 123
1	NNJ	NNJ	NNJ	NNN	NNN	NNN	NNJ	NNJ	NNJ
2	JJJ	NNJ	JJJ	JJJ	JJJ	JJJ	JJJ	JJJ	JJJ
3	JJJ	NNJ	JJJ	JJJ	JJJ	JJJ	JJJ	NNJ	JJJ
4	JJJ	NJJ	NNJ	NNJ	NNN	NNJ	NNJ	NNJ	NNN
5	JJJ	NJJ	NNJ	NNN	JJJ	NNJ	NNJ	NNJ	NNJ

Uit de tabel blijkt dat van de relevante testen (Nxx) de derde versie in 70% van de gevallen voor verheldering zorgt (van de relevante testen is 70% NNJ en 30% NNN of NJJ).
Er mag dan ook geconcludeerd worden dat de uiteindelijke versie een zinvolle uitbreiding is van de oorspronkelijke versie.

5 Uitbreiding Pocketstem met alfabet.

5.1 Inleiding.

Door het weglaten van de tweede intonatie, zoals die bij de MEA8000 versie van de Pocketstem voorkwam, is een hoeveelheid geheugenruimte vrijgekomen in het EPROM geheugen (van het type 27C256). De beschikbare 32 kbyte werden bij de MEA als volgt gebruikt:

- 2 kbyte besturingsprogramma,
- 14 kbyte spraakdatablok 1,
- 14 kbyte spraakdatablok 2,
- 2 kbyte tekstblok (ten behoeve van evaluatiedoeleinden).

Het spraakdatablok 2 is nu bij de PCF8200 versie van de Pocketstem weggelaten. Hierdoor is 14 kbyte geheugen vrijgekomen. Deze 14 kbyte zal nu worden gebruikt om spraakdata van het alfabet en het spelalfabet in op te slaan.

5.2 Hardware wijzigingen.

Om naast de zinnen ook gebruik te maken van het alfabet (en spelalfabet), moet de gebruiker zijn keuze doorgeven aan de microprocessor. Randvoorwaarde hierbij is dat het geheel gebruikersvriendelijk blijft.
Na uitgebreid overleg is besloten tot het volgende:
Met behulp van een drie-standen-schakelaar (zie fig. 5.1) kan de gebruiker aan de microprocessor doorgeven waarvan hij gebruik wil maken:

- de zinnenset,
- het alfabet,
- het spelalfabet.

Deze schakelaar zit aan dezelfde kant van de Pocketstem als de aan-uit schakelaar. In de middenstand spreekt de Pocketstem de zinnen uit,

zoals voorheen (dus ook met snelheidsvariaties). In de bovenstand staat de Pocketstem in de "alfabet mode" en is met elke toets een letter van het alfabet verbonden. Een druk op een toets resulteert dus in het uitspreken van de betreffende letter zonder snelheidsvariaties. (a,b,c,...)
Bij het spelalfabet geldt hetzelfde, alleen wordt nu bij elke letter het bijbehorende spelwoord uitgesproken. (anna, bernard, cornelis,...)
Het alfabet heeft 26 letters, terwijl het toetsenbord 28 toetsen heeft. De 2 toetsen die zodoende overblijven, worden gebruikt voor de woorden "ja" en "nee" in zowel alfabet als spelalfabet mode.
De indeling van de alfabet letters over de toetsen is gebeurd als weergegeven in fig. 5.1 .

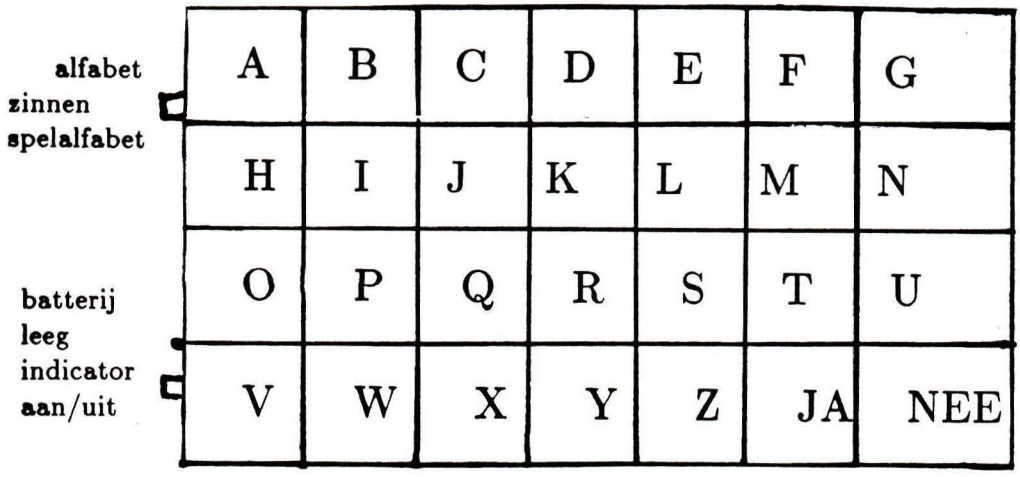


Fig. 5.1:Indeling van de toetsen.

Het toetsenbord is met zeven ingangslijnen (via een latch) en met vier uitgangslijnen (via een buffer) op de databus aangesloten ("7x4 matrix"). Door nu op de zeven ingangslijnen achtereenvolgens spanning te zetten en van de vier uitgangslijnen te kijken welke hoog wordt, kan achterhaald worden welke toets was ingedrukt.
De databus bevat acht lijnen, waarvan er zeven verbonden met het toetsenbord. De achtste lijn kan nu gebruikt worden om te kijken in welke stand de alfabetschakelaar staat. De schakeling waarmee dit gebeurt is gegeven in bijlage 6 en in fig. 5.2. De achtste lijn is dus nu ingangslijn van de

schakelaar. De schakelaar heeft twee uitgangslijnen, die ook weer met met de databus verbonden zijn, op dezelfde manier als de vier uitgangslijnen van het toetsenbord (daar waren nog vier lijnen over). In feite wordt de schakelaar dus op dezelfde manier aangesloten en gescand als het toetsenbord:
Er wordt door de microprocessor een signaal op de middenaftakking van de schakelaar gezet. Als de schakelaar in de middenstand staat (zinnen) dan zal géén van de uitgangslijnen hoog worden (zie bijlage 6). Als de schakelaar in één van de twee andere standen staat, zal één van de twee uitgangslijnen hoog worden, afhankelijk van de stand van de schakelaar. Dit signaal gaat terug naar de microprocessor, waar het wordt verwerkt. Om de stand van de schakelaar te bekijken is een subroutine geschreven ("scansw"), die is opgenomen in bijlage 7. Hierover meer in de volgende paragraaf.

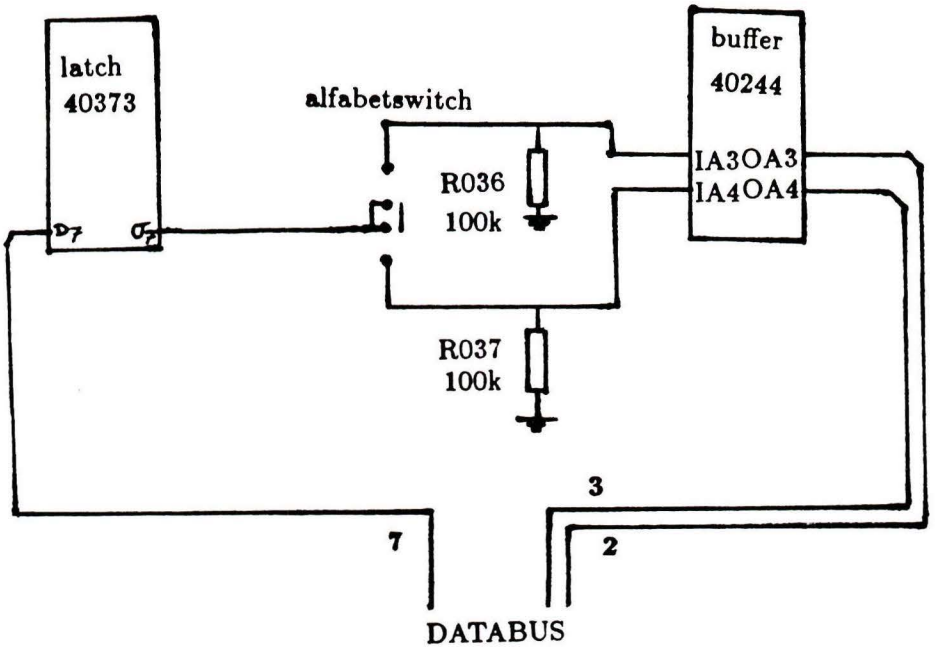


Fig. 5.2: De alfabetschakeling.

5.3 De assembler software.

Om de stand van de schakelaar te bekijken is een subroutine geschreven. Van deze routine ("scansw") waarvan de listing staat in bijlage 7, is de flowchart gegeven in fig. 5.3 . De gewijzigde routine "scan" staat in bijlage 8. In de routine wordt het "alfabet status word" (alf) gebruikt: dit is een byte in het RAM geheugen, die verder nergens anders voor wordt gebruikt. De twee minst significante bits van deze byte zijn de "alfabetflag" en de "spelalfabetflag". Deze bits geven aan of de schakelaar in de alfabet stand respectievelijk de spelalfabetstand staat. De overige bits van het Alfabet Status Word worden niet gebruikt. (zie tabel 5.1)
Deze twee flags kunnen in de subroutine "scansw" geset worden. Scansw wordt als eerste instructie aangeroepen in de subroutine "scan". Als een toets ingedrukt wordt, en een power-up keyboard signaal wordt gegenereerd [1], dan wordt "scan" aangeroepen om te kijken welke toets was ingedrukt. In "scan" wordt dus ook meteen de stand van de schakelaar bekeken (scansw). In de procedure "ssas", wordt de datapointer naar de beginplaats van een datablok gezet. Afhankelijk van welk bit in het Alfabet Status Word geset is, wordt de datapointer naar het begin van het betreffende blok gezet. De geheugenindeling hierbij is weergegeven in fig. 5.4, waarbij de geheugenruimte van 2 kbyte (0800H) tot 18176 byte (4700H) gebruikt kan worden voor spraakdata van de zinnen.

Tabel 5.1: Het alfabet status word.

bit	naam	omschrijving
0	spalfa	spelalfabetbit
1	alfa	alfabetbit

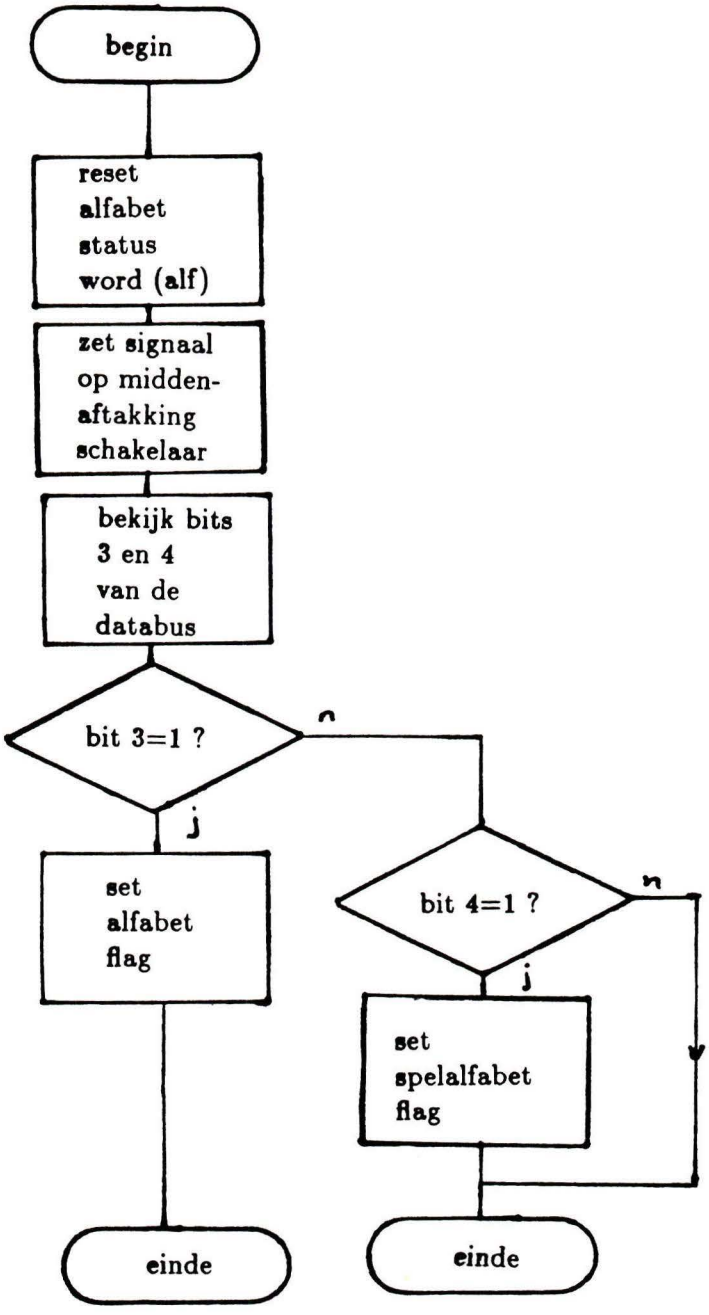


Fig. 5.3: Flowchart van de subroutine "scansw".

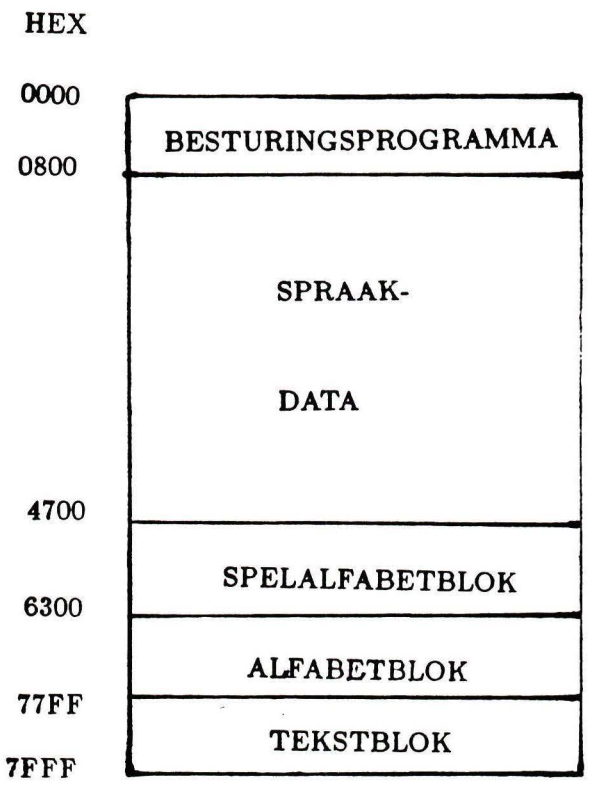


Fig. 5.4: Geheugenindeling van de EPROM.

Als bijvoorbeeld de alfabet flag geset is, zal de datapointer in "ssas" de waarde 6300H krijgen, wat het beginadres is van het alfabetblok (hexadecimaal). De rest (de data overbrengen naar de PCF) gaat hetzelfde als bij de zinnen en daarvoor wordt verwezen naar [1].

5.4 De APPLE software.

Door Tan [2] en Legdeur [3] is een programmapakket in BASIC ontwikkeld voor een APPLE IIe computer, waarmee de gebruiker een EPROM voor de Pocketstem kan programmeren met een door hem gekozen zinnenset. Met dit programmapakket is het mogelijk om een EPROM geheugen te vullen met het besturingsprogramma, de spraakdata en het tekstblok (zie fig. 5.4). In dit hoofdstuk staat beschreven hoe het programmapakket geschikt wordt

gemaakt om tevens het spelalfabet- en alfabetblok automatisch in de EPROM te bakken. Om dit te realiseren moeten allereerst de spraakdata van het alfabet en spelalfabet gegenereerd worden. Hierin zijn de volgende handelingen te onderscheiden:

- het opnemen van het gesproken alfabet en spelalfabet met een bandrecorder,
- het analyseren en in PCF parameters omzetten van deze boodschappen op de IPO-VAX 8530 computer met behulp van het LVS software pakket [13],
- het overhalen van de spraakdata naar de APPLE IIe computer,
- het programmeren van een EPROM met behulp van een EPROM programmer, welke aangesloten is op de APPLE computer.

Deze manier is voor de gebruiker nogal omslachtig. Derhalve is de APPLE programmatuur van Tan [2] en Legdeur [3] zodanig gewijzigd, dat het programmeren van alfabet- en spelalfabetblok (hierna alfabetblok genoemd) ook als optie ingebouwd.
De gewijzigde versie is ook weer menu-gestuurd. De procedure "bak-eprompcf" is gewijzigd, de overige zijn hetzelfde gebleven. Het stuk listing van deze procedure "bak-eprompcf" dat gewijzigd is, is gegeven in bijlage 9.
Bij het opstarten van de gewijzigde systeemschijf wordt nu het volgende menu geboden: (voor de ongewijzigde versie verwijs ik naar [2,3])

"Wat wil je doen?"
1. EPROM bestand editten;
2. EPROM bakken;
3. Stoppen.

De mogelijkheden 1 en 3 zijn ongewijzigd gebleven en daarvoor wordt verwezen naar Tan [2] en Legdeur [3].
Bij mogelijkheid 2 wordt gevraagd (na invoering van het type EPROM en beantwoording van de vraag of automatisch gebakken moet worden):

Alles bakken? (Y/N)

1. Bij het intypen van "Y" komt na een bevestiging op het beeldscherm te staan:

Te bakken: Frank's programma
Blok 1
Alfabetblok
Tekstblok

Waarbij

- Frank's programma het PCF besturingsprogramma is dat in dit verslag besproken is; het realiseert de verschillende spraaksnelheden en het is geschikt voor het uitspreken van alfabet en spelalfabet.
- Blok 1 de spraakdata van de 28 gekozen zinnen bevat.
- Alfabetblok de spraakdata van alfabet en spelalfabetblok bevat.
- Tekstblok de tekst van de 28 gekozen zinnen bevat.

Achtereenvolgens worden nu de 4 bovenstaande blokken in de EPROM gebakken, hetgeen op het scherm te zien is aan de mededeling:

Bezig met:

Achter deze mededeling staat één van de 4 te programmeren blokken. (Frank's programma, Blok 1, Alfabetblok en Tekstblok).

2. Bij het intypen van "N" wordt daarna gevraagd:

-Alfabetblok? (Y/N)

- Wanneer "Y" ingetypt wordt, dan wordt Frank's programma en het alfabetblok gebakken. Het programma gaat verder met de vraag:

Blok 1? (Y/N)

- Wanneer "N" wordt ingevoerd, dan gaat het programma verder met de vraag:

Ronald's PCF programma? (Y/N)

Waarbij Ronald's PCF programma het ongewijzigde Pocketstem programma is (zonder snelheidsvariaties en zonder alfabetmogelijkheid). Het programma gaat verder met de vraag:

Blok 1? (Y/N)

-Blok 1? (Y/N)

Wanneer deze vraag met "Y" beantwoord wordt, dan wordt het blok met 28 geselecteerde zinnen gebakken; bij antwoord "N" wordt het niet gebakken. In beide gevallen krijgt de gebruiker nog de vraag:

Tekstblok? (Y/N)

De afhandeling hiervan gaat analoog als bij Blok 1.

Het geheel is weergegeven in de vorm van flowchart in fig. 5.5 .
Na deze vragen wordt door de computer nog om een bevestiging gevraagd, waarna de computer begint met het bakken van de gewenste blokken.

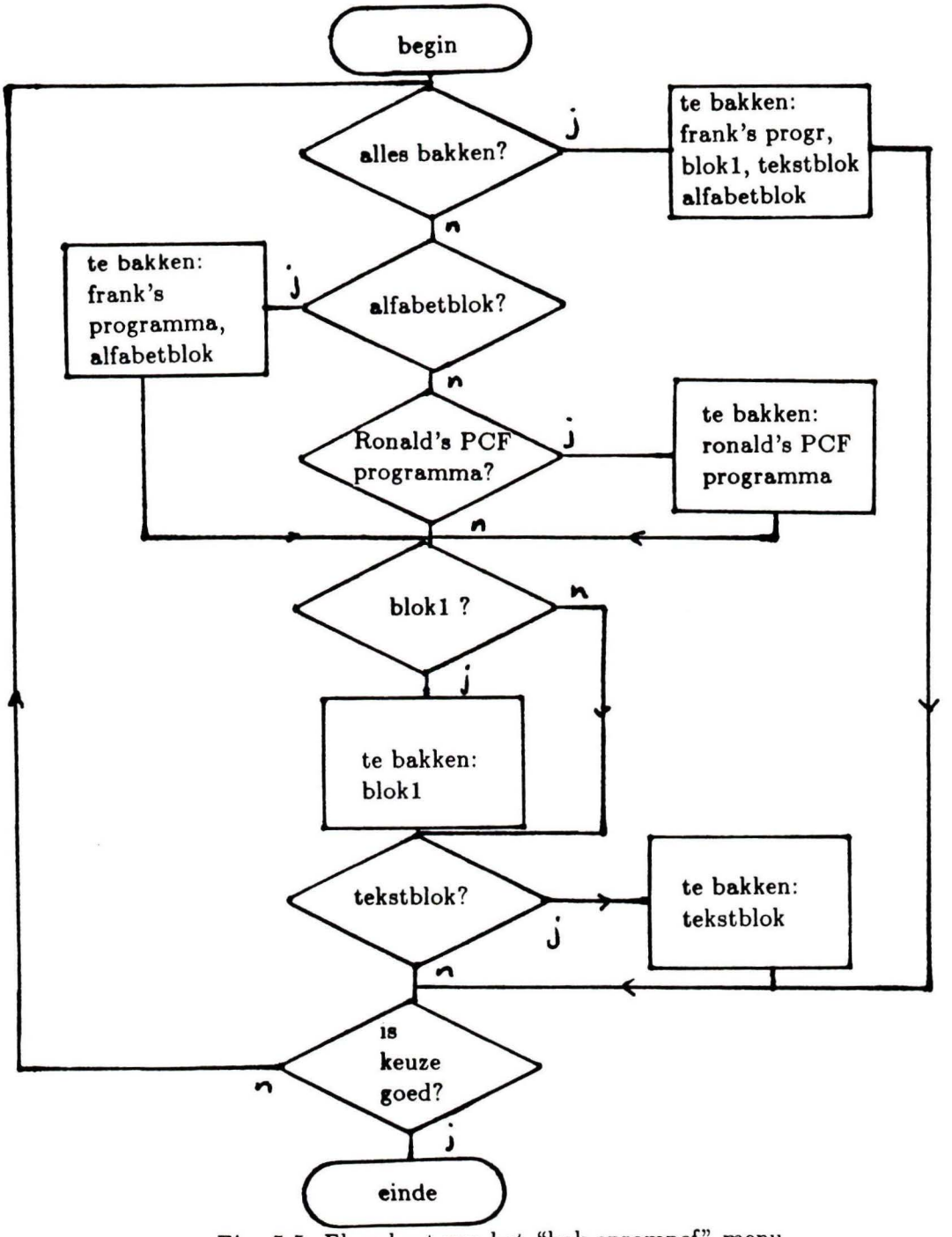


Fig. 5.5: Flowchart van het "bak-eprompcf" menu.

6 Conclusies.

Met betrekking tot het literatuuronderzoek kan gezegd worden dat een aantal relevante artikelen gevonden zijn. Uit deze artikelen blijkt dat er ook hulpmiddelen bestaan voor spraakgehandicapten die zwaar lichamelijk gehandicapt zijn. Verder is daarvan een specifieke oplossing besproken. Ook is een spraakhulpmiddel aan de orde gekomen waarbij met name gelet is op de snelheid waarmee de gebruiker ermee kan communiceren.

Wat betreft de spreeksnelheidsvariaties van de Pocketstem kan het volgende opgemerkt worden:
De PCF8200 spraakchip biedt inderdaad betere mogelijkheden tot snelheidsvariaties dan de MEA8000. Echter de aanvankelijke gedachte om één snelheidsvariatie in te bouwen is uitgelopen in twee variaties. Dit bleek in de evaluatie in sommige gevallen ook nodig, daar anders sommige zinnen niet goed verstaan of begrepen werden.

De Pocketstem is nu ook uitgerust met alfabet en spelalfabet. Technisch gezien werkt het alfabet (en spelalfabet) prima.
Of het ook gebruikersvriendelijk is, zal een evaluatie moeten uitwijzen. Het gewijzigde software pakket op de APPLE IIe computer werkt zoals voorheen volgens menusturing. Tijdens gebruik is gebleken dat het naar behoren functioneert.

Literatuurlijst.

1. *Waterham R.P., Verhoeven M.W.C.*: Technische beschrijving van de Pocketstem; IPO-rapport 606, Eindhoven, 1987.

2. *Tan T.S.G.*: Ondersteuningsprogrammatuur voor de Pocketstem; Stagerapport vakgroep EME Technische Universiteit Eindhoven, 1987.

3. *Legdeur G.J.A.A.*: Aanpassing van en documentatie voor programma's ten behoeve van de Pocketstem; IPO-rapport 624,Eindhoven, 1987.

4. *Baker B*: Minspeak; Byte, vol.7, no.9; p.186-202, USA,1982.

5. *Gordon D, Zabo D*: Technical solutions for people with communication impairments; Proceedings of the international congress on technology and technology exchange, p42-43, 1984, Pittsburg USA.

6. *Newell A.F.*: Speech communication technology- lessons from the disabled; Electronics and power, vol.32, no.9; p.661-664, 1986.

7. *Johnson E.L.*: A new approach to keyboard design for the handicapped; Proceedings on the third annual workshop on computers and the handicapped, p.1-2, USA, 1984.

8. *Mason S.D., Tanaka N.K., Ming gon Jon Lian*: Computer assisted speech synthesis for severly disabled individuals; Computers in the schools, vol.3, no. 3-4, p.131-140, USA, 1986.

9. *Mariani L.*: Technical aids for the handicapped: recent results, problems and perspectives; Elettrotechnica, vol. 71, no.11, p. 1007-1029, Italy, 1984.

10. *AFIPS conference proceedings*: Computer speech for people with cerebral palsy; vol. 50, p.663-664, USA, 1981.

11. *Shannon D.A., Staewen W.S., Miller J.T., Cohen B.S.*: Morse code controlled aid for the non-vocal quadriplegic; Medical Instrumentation, vol. 15, no. 5, p. 341-343, USA, 1981.

12. *Morse M.S., Obrien E.M.*: Research summary of a scheme to ascertain the availability of speech information in the myoelectric signals of neck and head muscles using surface electrodes; Computers in biology and medicine, vol. 16, no.6, p. 399-410, 1986.

13. *Vogten L.L.M.*: LVS Speech processing programs on IPO-VAX 11/780, IPO handleiding 67, Eindhoven, 1985.

14. *Waterham R.P.*:Technische beschrijving van de Compacte Spraakhulp I (CSH I), IPO rapport 525, Eindhoven, 1986.

15. *Have; ten M*: System specification voice synthesizer PCF8200 (M4790), Philips laboratory report DPE 85105, 1985.

Bijlage 1.

```
!*****************************************************************************
!
.sect   _chr7
.define chr7
.extern incrr7,decrr7
!
chr7:   mov     r1,7            !load @r7 in accu
        mov     a,@r1           !done
        lcall   decrr7          !check with previous @r7
        mov     r1,7            !
        mov     0,@r1           !previous sentence in r0
        cjne    a,0,1f          !if not equal then return
        lcall   incrr7          !restore r7
        mov     r1,7            !reload @r7 in accu
        mov     a,@r1           !done
        jb      secout,2f       !id
        setb    secout          !second pronunciation
        sjmp    3f              !skip next instruction
2:      clr     secout          !id
3:      mov     @r1,a           !store in @r7
        ret
1:      lcall   incrr7          !restore r7
        clr     secout          !id
        ret
!
!*****************************************************************************
```

Bijlage 2.

```
!*******************************************************************************
!
.sect   _sssl
.define sssl
!
sssl:   setb    rs0             !routine works in reg bank 1
        mov     dph,r5          !fill data pointer
        mov     dpl,r4          !filled with address of sent. length
        inc     dptr            !first get lower order byte
        clr     a               !
        movc    a,@a+dptr       !lo sentence length in accu
        add     a,r4            !result is lo end address
        mov     r6,a            !store in r6
!
!attention decr dpl can go over border of dph !!!action!!!
!it is because the pcf is 5 byte orientated!!
!
        mov     a,dpl           !check if dpl is zero
        jnz     1f              !if accu not zero then no action
        dec     dph             !low dp will underflow, that is corrected!!
1:      dec     dpl             !now get high order
        clr     a               !
        movc    a,@a+dptr       !ho sent. length in accu
        addc    a,r5            !result is ho end address (with lo overflow)
        mov     r7,a            !store in r7
        inc     dptr            !dptr on address of.....
!****************** here an extra inc of data pointer !!!!!!!!
        inc     dptr            !dptr on address of.....
        inc     dptr            !dptr on address of.....
        clr     a               !command byte goes to r0 of bank 2
        movc    a,@a+dptr       !first command byte from 4th byte 1st frame
        jb      secout,2f       !if secout high then speak fast
5:      anl     a,#0xf0         !normal speech command byte
        sjmp    3f              !skip next instruction
2:      orl     a,#0x02         !fast speech command byte
3:      setb    rs1             !select reg bank 2
        clr     rs0             !select reg bank 2
        mov     r0,a            !acc to r0 of bank 2
        clr     rs1             !select reg bank 1
        setb    rs0             !select reg bank 1
        inc     dptr            !...pitchstart
        lcall   cilis0          !check if length is 0
        clr     rs0             !select reg. bank 0
        ret
!
!*******************************************************************************
```

Bijlage 3.

```
!*****************************************************************************
!
.sect   _chr7
.define chr7
.extern incrr7,decrr7
!
chr7:   mov     r1,7            !load @r7 in accu
        mov     a,@r1           !done
        lcall   decrr7          !check with previous @r7
        mov     r1,7            !
        mov     0,@r1           !previous sentence in r0
        cjne    a,0,1f          !if not equal then return
        lcall   incrr7          !restore r7
        mov     r1,7            !reload @r7 in accu
        mov     a,@r1           !done
        jb      thrdout,4f      !check which pronunciation speed
        jb      secout,2f       !id
        setb    secout          !second pronunciation
        sjmp    3f              !skip next instruction
2:      setb    thrdout         !third pronunciation
        clr     secout          !id
3:      mov     @r1,a           !store in @r7
        ret
1:      lcall   incrr7          !restore r7
4:      clr     thrdout         !first pronunciation speed
        clr     secout          !id
        ret
!
!*****************************************************************************
```

Bijlage 4.

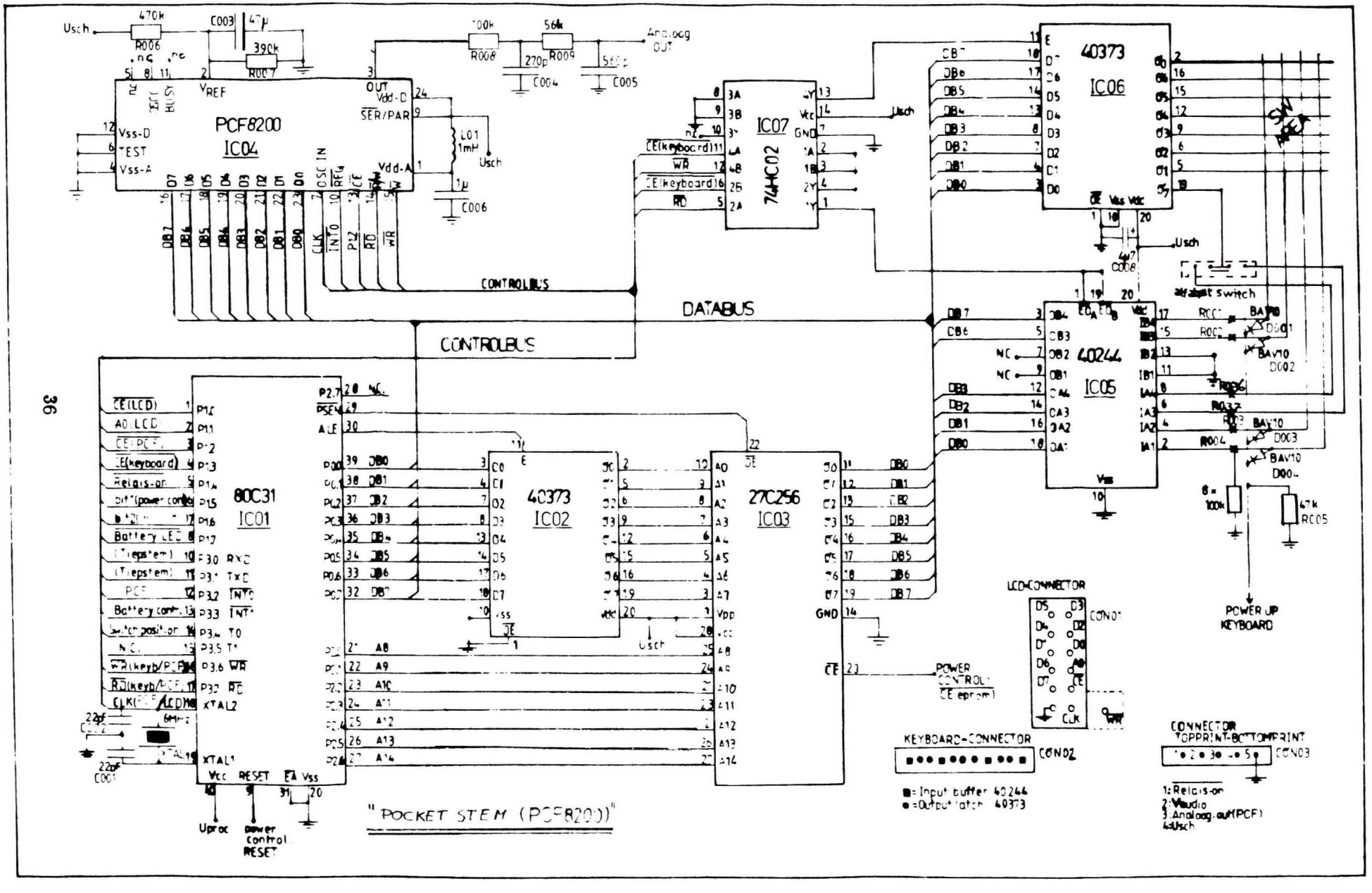
```
!*******************************************************************************
!
.sect   _sssl
.define sssl
!
sssl:   setb    rs0             !routine works in reg bank 1
        mov     dph,r5          !fill data pointer
        mov     dpl,r4          !filled with address of sent. length
        inc     dptr            !first get lower order byte
        clr     a               !
        movc    a,@a+dptr       !lo sentence length in accu
        add     a,r4            !result is lo end address
        mov     r6,a            !store in r6
!
!attention decr dpl can go over border of dph !!!action!!!
!it is because the pcf is 5 byte orientated!!
!
        mov     a,dpl           !check if dpl is zero
        jnz     1f              !if accu not zero then no action
        dec     dph             !low dp will underflow, that is corrected!!
1:      dec     dpl             !now get high order
        clr     a               !
        movc    a,@a+dptr       !ho sent. length in accu
        addc    a,r5            !result is ho end address (with lo overflow)
        mov     r7,a            !store in r7
        inc     dptr            !dptr on address of.....
!****************** here an extra inc of data pointer !!!!!!!!
        inc     dptr            !dptr on address of.....
        inc     dptr            !dptr on address of.....
        clr     a               !command byte goes to r0 of bank 2
        movc    a,@a+dptr       !first command bety from 4th byte 1st frame
        jbc     alfa,5f         !if alfa is high then no different speed
        jbc     spalfa,5f       !id if spalfa is high
        jb      secout,2f       !if secout high then speak fast
        jb      thrdout,4f      !if thrdout high then speak slow
5:      anl     a,#0xf0         !normal speech command byte
        sjmp    3f              !skip next instruction
2:      orl     a,#0x02         !fast speech command byte
        sjmp    3f              !skip next instruction
4:      orl     a,#0x03         !slow speech command byte
3:      setb    rs1             !select reg bank 2
        clr     rs0             !select reg bank 2
        mov     r0,a            !acc to r0 of bank 2
        clr     rs1             !select reg bank 1
        setb    rs0             !select reg bank 1
        inc     dptr            !...pitchstart
        lcall   cilis0          !check if length is 0
        clr     rs0             !select reg. bank 0
        ret
```

Bijlage 5.

```
!*****************************************************************************
!
.sect   _init
.define _init
.extern main
!
init:   mov     sp,#0x27        !set stack pointer on ram address 40, stack
        mov     a,r7            !check if r7 is within limits (24-31)
        clr     c               !always clear before substraction
        subb    a,#24           !borrow if a<24
        jnc     1f              !if carry then correct r7
        clr     c               !carry was set
        mov     r7,#24          !speakbuffer pointer on ram address 24
        mov     r6,#24          !speakbuffer pointer on ram address 24
        sjmp    2f              !finish initialisation
1:      mov     a,r7            !same for a>31
        subb    a,#32           !if carry then right values
        jc      3f              !
        mov     r7,#24          !speakbuffer pointer on ram address 24
        mov     r6,#24          !speakbuffer pointer on ram address 24
        sjmp    2f              !finish initialisation
3:      clr     c               !carry was set
        mov     6,7             !copy r7 to r6
2:      anl     dsw,#3          !reset DSW, except secout and thrdout
        setb    pt1             !timer 1 high priority
        setb    ea              !enable all interrupts
        ajmp    main            !jump to main program
!
!*****************************************************************************
```

Bijlage 6.

Schema van de hardware.


PCF8200
IC04
80C31
IC01
40373
IC02
27C256
IC03
74HC02
IC07
40373
IC06
40244
IC05
CONTROLBUS
DATABUS
CONTROLBUS
"POCKET STEM (PCF8200)"
KEYBOARD-CONNECTOR
LCD-CONNECTOR
CONNECTOR TOPPRINT-BOTTOMPRINT
■ = Input buffer 40244
● = Output latch 40373
1: Relais-on
2: Vaudio
3: Analoog-out(PCF)
4: Usch

Bijlage 7.

```
!*****************************************************************************
!
.sect _scansw
.define scansw
!
scansw:  anl  alf,#0            !reset alfabet status word
         clr  keyen             !enable keyboard
         mov  a,#0x80           !send 1 to alfabetswitch
         movx @r0,a             !id
         movx a,@r0             !input switch position
         setb keyen             !disable keyboard
         anl  a,#0x0c           !check bits 3 and 4
         clr  c                 !reset carry
         rrc  a                 !check third bit
         rrc  a                 !id
         rrc  a                 !id
         jc   1f                !if third bit 1 then set alfa
         rrc  a                 !id
         jc   2f                !if fourth bit 1 then set spalfa
         sjmp 3f                !no alfabetspeech, return to 'scan'
1:       setb alfa              !set alfabet flag
         clr  c                 !reset carry
         sjmp 3f                !return to scan
2:       setb spalfa            !set spelalfabet flag
         clr  c                 !reset carry
3:       ret                    !return to scan
!
!
```

Bijlage 8.

```
!*****************************************************************************
!
.sect   _scan
.define scan
.extern calsno,incrr7,conspe,chr7,scansw
!
scan:   lcall   scansw          !check position alfabetswitch
        mov     r0,#250         !scan 250 times to prevent 'dender'
        mov     r1,#0           !reset r0
again:  clr     keyen           !enable keyboard
        mov     a,#0x7f         !send 1's to keyboard
        movx    @r0,a           !send them to imaginairy address (keyboard)
        movx    a,@r0           !read keyboard
        setb    keyen           !disable keyboard
        anl     a,#0xc3         !be sure that only bits 7,6 and 1,0 are good
        cjne    a,1,5f          !compare accu with reg 1
        sjmp    4f              !if equal then go on
5:      mov     r1,a            !save value in r1
        mov     r0,#250         !reset r0 on 250
4:      djnz    r0,again        !we want 250 times the same scan result!
        jnz     0f              !if accu 250 times equals zero then out
        clr     key             !no key action anymore
        lcall   conspe          !routine to control speane bit
        ret
0:      mov     a,#1            !now find which key it is
        mov     r0,a            !copy accu to r0
3:      clr     keyen           !enable keyboard
        movx    @r0,a           !write accu to keyboard (im. address)
        movx    a,@r0           !read keyboard line
        setb    keyen           !disable keyboard
        anl     a,#0xc3         !be sure that only bits 7,6 and 1,0 are good
        jnz     1f              !jump if key is detected
        mov     a,r0            !copy r0 to accu
        rlc     a               !go to next scan line
        jc      2f              !if carry then end of scan
        mov     r0,a            !copy accu to r0 (backup)
        sjmp    3b              !try next line
2:      clr     c               !reset carry
        clr     key             !no key action after all
        lcall   conspe          !routine to control speane bit
        ret
1:      jnb     key,6f          !go on if no key action was present
!*****************************************************************************
!a second key can only be detected if the first has been released in time.
!*****************************************************************************
        lcall   conspe          !routine to control speane bit
        ret
6:      setb    key             !now indicate key action
        lcall   calsno          !
        mov     1,7             !move content of r7 to r1
        mov     @r1,3           !move r3 to @r7
        lcall   chr7            !
!*****************************************************************************
!here action to determine whether sec. intonation is wanted!!!!!
!*****************************************************************************
        lcall   incrr7          !routine to incr speakpointer
        lcall   conspe          !routine to control speane bit
        ret
!
```

Bijlage 9.

```
1000  REM  BAK-EPROMPOF PROGRAMM
    A
1010 AN -  PEEK (778)
1020  LET HRAM - 32767
1030  LET LRAM - 16384
1040  HIMEM: LRAM - 1
1050  LET D$ -  CHR$ (4):BELL$ -
     CHR$ (7):BS$ -  CHR$ (8):CR
    $ -  CHR$ (13)
1060  LET TØP% - 34
1070  LET TYP$ - ""
1080  POKE TØP%,Ø: HOME : INVERSE
    : PRINT "BAKKENPOF": NORMAL
    : PRINT "APPARAAT ";AN: VTAB
    7: POKE TØP%,8: HOME
1090  PRINT "AUTOMATISCH BAKKEN
    ?(Y/N): ";: GET YN$: PRINT Y
    N$
1100  IF YN$ - "Y" THEN  VTAB 1:
     HTAB 10: PRINT ": AUTOMATIS
    CH": VTAB 7: HOME : GOTO 113
    Ø
1110  IF YN$ - "N" GOTO 2030
1120  PRINT BELL$: PRINT "FOUTIE
    F ANTWOORD!!!": GOTO 1050
1125  REM  AUTOMATISCH BAKKEN
1130 CV - 1
1140  GOTO 2340
1145  IF TYP$ - "" GOTO 2340
1150  VTAB 7: HOME
1155  LET FF$ - "N":RP$ - "N":B1
    $ - "N":BA1$ - "N":TB$ - "N"
    :ALF$ - "N":SPALF$ - "N"
1160  PRINT
1170  PRINT "ALLES BAKKEN ?(Y/N)
    : ";: GET YN$: PRINT YN$
1180  IF YN$ - "Y" THEN FF$ - "Y
    ":B1$ - "Y":ALF$ - "Y":SPALF
    $ - "Y":TB$ - "Y": GOTO 1330
```

```
1180  IF YN$ = "N" GOTO 1202
1200  PRINT BELL$: PRINT "FOUTIE
    F ANTWOORD!!!": GOTO 1170
1202  PRINT : PRINT "ALFABETBLOK
    ?(Y/N):";: GET ALF$: PRINT A
    LF$
1204  IF ALF$ = "Y" OR ALF$ = "N
    " GOTO 1208
1206  PRINT BELL$: PRINT "FOUTIE
    F ANTWOORD!!!": GOTO 1202
1208  IF ALF$ = "Y" THEN FF$ = "
    Y":SPALF$ = "Y": GOTO 1240
1210  PRINT : PRINT "RONALD'S FO
    F PROGRAMMA?(Y/N): ";: GET R
    P$: PRINT RP$
1220  IF RP$ = "Y" OR RP$ = "N" GOTO
    1240
1230  PRINT BELL$: PRINT "FOUTIE
    F ANTWOORD!!!": GOTO 1210
1240  PRINT : PRINT "BLOK1 ?(Y/N
    ): ";: GET B1$: PRINT B1$
1250  IF B1$ = "Y" OR B1$ = "N" GOTO
    1300
1260  PRINT BELL$: PRINT "FOUTIE
    F ANTWOORD!!!": GOTO 1240
1300  PRINT : PRINT "TEKSTBLOK ?
    (Y/N): ";: GET TB$: PRINT TB
    $
1310  IF TB$ = "Y" OR TB$ = "N" GOTO
    1330
1320  PRINT BELL$: PRINT "FOUTIE
    F ANTWOORD!!!": GOTO 1300
1330  VTAB 3: HTAB 1: PRINT  SPC(
    35): PRINT  SPC( 35): PRINT
     SPC( 35): PRINT  SPC( 35):V
    T - 2: VTAB 3: HTAB 1: PRINT
    "TE BAKKEN:"
1340  IF RP$ = "Y" THEN VT = VT +
    1: VTAB VT: HTAB 11: PRINT "
    RONALD'S FOF PROGRAMMA"
1345  IF FP$ = "Y" THEN VT = VT +
    1: VTAB VT: HTAB 11: PRINT "
    FRANK'S PROGRAMMA"
1350  IF B1$ = "Y" THEN VT = VT +
    1: VTAB VT: HTAB 11: PRINT "
    BLOK1"
```

```
1360  IF ALF$ = "Y" THEN VT = VT
     + 1: VTAB VT: HTAB 11: PRINT
    "ALFABET/SPELALFABETBLOK"
1370  IF TB$ = "Y" THEN VT = VT +
    1: VTAB VT: HTAB 11: PRINT "
    TEKSTBLOK"
1380  VTAB 7: HOME
1390  PRINT
1400  PRINT "IS BOVENSTAANDE KEU
    ZE GOED ?(Y/N): ";: GET YN$:
     PRINT YN$
1410  IF YN$ = "Y" GOTO 1440
1420  IF YN$ = "N" GOTO 1150
1430  PRINT BELL$: PRINT "FOUTIE
    F ANTWOORD!!!": GOTO 1400
1440  HOME : VTAB 12
1450  FLASH : PRINT "ER WORDT AU
    TOMATISCH GEBAKKEN!": NORMAL

1460  PRINT "MOCHT ER IETS MIS G
    AAN,"
1470  PRINT "DAN HOOR JE EEN BEE
    P."
1480  PRINT "JE ZULT DAN IETS FO
    UT HEBBEN GEDAAN!"
1490  PRINT "IK HOOP, DAT DE KOF
    FIE (";: INVERSE : PRINT "OF
     IETS ANDERS";: NORMAL : PRINT
    ");"
1500  PRINT "SMAAKT."
1510  VTAB 20: HTAB 1: PRINT "BE
    ZIG MET: "
1520  IF RP$ = "N" GOTO 1561
1530  VTAB 20: HTAB 12: FLASH : PRINT
    "RONALD'S POF PROGRAMMA": NORMAL

1540  LET OV = 2:A$ = "RONALDPOF
    "
1550  LET A1 = 0:H1 = 0:L1 = 0
1560  GOSUB 1660
1561  IF FP$ = "N" GOTO 1570
1562  VTAB 20: HTAB 12: FLASH : PRINT
    "FRANK'S PROGRAMMA": NORMAL
```

```
1563  LET A$ = "FRANK":A1 = 0:H1
    = 0:L1 = 0
1564  GOSUB 1880
1570  IF B1$ = "N" GOTO 1620
1580  VTAB 20: HTAB 12: FLASH : PRINT
    "BLOK1";: NORMAL : PRINT  SPC(
    20)
1590  LET CV = 3:A$ = "BLOK1"
1600  LET A1 = 2048:H1 = 8:L1 =
    0
1610  GOSUB 1880
1620  LET A1 = A1 + LB
1630  IF A1 < 16384 THEN  GOTO 1
    662
1635 BA1$ = "Y"
1640  LET CV = 4:A$ = "BLOK1A"
1650 H1 =  INT (A1 / 256)
1660 L1 = A1 - (H1 * 256)
1661  GOSUB 1880
1662  IF ALF$ = "N" GOTO 1668
1663  VTAB 20: HTAB 12: FLASH : PRINT
    "ALFABETBLOK";: NORMAL : PRINT
     SPC( 14)
1664  LET A$ = "ALFABET":A1 = 25
    344:H1 = 99:L1 = 0
1665  GOSUB 1880
1666  IF SPALF$ = "N" GOTO 1670
1667  VTAB 20: HTAB 12: FLASH : PRINT
    "SPELALFABETBLOK";: NORMAL :
     PRINT  SPC( 7)
1668  LET A$ = "SPELALFABET":A1 =
    18176:H1 = 71:L1 = 0
1669  GOSUB 1880
1670  IF TB$ = "N" GOTO 1720
1680  VTAB 20: HTAB 12: FLASH : PRINT
    "TEKSTBLOK";: NORMAL : PRINT
     SPC( 16)
1690  LET CV = 5:A$ = "TEKSTBLOK
    "
1700  LET A1 = 30720:H1 = 120:L1
    = 0
1710  GOSUB 1880
```