DCT

Instituut voor Perceptie Onderzoek
Postbus 513, 5600 MB Eindhoven

28.01.1991

Rapport no. 778

DCT: Software voor de besturing van DECtalk in een VAX/VMS cluster-omgeving

H.P.J. Diesveld

Inhoudsopgave

fm 33

DCT: Software voor de besturing van DECtalk in een VAX/VMS clusteromgeving

H.P.J. Diesveld

1 Samenvatting

Tijdens een stage van 3 weken in het IPO is software ontwikkeld voor het besturen van DECtalk, een systeem waarmee tekst kan worden omgezet in (amerikaans) engels spraakgeluid. Het systeem is gebaseerd op allofoonsynthese door regels (Klatt). Met de ontwikkelde software kan DECtalk in een VAX-VMS-clusteromgeving worden bestuurd.

De gebruiker logt via een willekeurige terminal in op één van de vier IPO machines (NL200, VAXLAB, VAXDAC of IPOMV4) die deel uitmaken van het VAX-cluster. Met het commando DCT wordt engelse tekst, vanuit een disk-file of rechtstreeks ingetikt op het toetsenbord, via Ethernet verzonden naar de IPOMV4, die als host voor DECtalk functioneert. DECtalk spreekt de tekst vervolgens in real time uit en deze synthetische spraak kan dan op kanaal 7 van het audio-FM kabelnet worden beluisterd.

Verslag van een stage uitgevoerd van 3 t/m 21 december 1990 in de groep "Horen en Spraak" van het IPO o.l.v. L.L.M. Vogten.

2 Inleiding

In het Instituut voor Perceptie Onderzoek (IPO) doet men binnen de groep 'Horen en Spraak' onderzoek naar o.a. spraaksynthese per computer. In dit kader is in het IPO software ontwikkeld waarmee nederlandse, duitse en engelse teksten ten gehore kunnen worden gebracht. Digital heeft in 1984 een systeem, DECtalk genaamd, op de markt gebracht voor spraaksynthese. Dat is, volgens kenners, het beste systeem dat (voor amerikaans engels) op het moment verkrijgbaar is. Voor het IPO-onderzoek aan tekst-naar-spraak wil men DECtalk op dezelfde wijze kunnen gebruiken als het eigen IPO-systeem. Dit houdt in dat men, na inloggen op een van de centrale VAX computers, vanaf elke willekeurige terminal de mogelijkheid moet hebben om een tekstbestand (of een op de terminal ingetikte regel tekst) naar DECtalk te versturen en vervolgens die spraak via het audio-FM-net te beluisteren.

Tijdens mijn stage van 3 tot en met 21 december 1990 heb ik de software ontwikkeld om DECtalk op deze wijze te besturen. Daarbij is gebruik gemaakt van de door Philippe Allain en Leo Vogten ontwikkelde LIO-software voor het sturen van spraak samples naar de D/A converters van de VAXDAC, de VAXLAB en de IPOMV4.

Ik wil hier mijn begeleider Leo Vogten bedanken voor de tijd die hij heeft vrijgemaakt gedurende vooral het tweede gedeelte van mijn stage. Tevens dank ik Philippe Allain voor het wegwijs maken in de reeds bestaande LIO-software. Bij beiden vond ik een altijd gewillig oor als ik een van mijn vele vragen stelde. Een groot voordeel van de door hen geschreven LIO-software is dat er slechts geringe aanpassingen nodig waren om de DECtalk software te kunnen laten draaien. Dit ondanks het feit dat de LIO-software in eerste instantie was ontworpen om grote hoeveelheden spraaksamples tussen de afzonderlijke computers uit te wisselen; niet om ASCII teksten te transporteren.

Ik heb, ondanks het feit dat de stage slechts drie weken kon duren, de werksfeer binnen het IPO als heel erg prettig ervaren en ik zal er nog vaak met plezier aan terugdenken.

3 Opdracht

In de groep 'Horen en Spraak' van het Instituut voor Perceptie Onderzoek vindt onderzoek plaats met behulp van VAX computers (VAX8530, VAXLAB, microVAX-II systemen en terminalservers) die onderling een cluster vormen en via Ethernet toegankelijk zijn vanuit een groot aantal terminals. Bij dat spraakonderzoek wordt o.m. gebruik gemaakt van een tweetal systemen waarmee tekst kan worden omgezet in synthetische spraak: DS en DECtalk. Het eerste, DS (Difoon-Synthese), is een software-systeem dat bestaat uit Pascal- en Fortran- modules die binnen het IPO zijn ontwikkeld en gebruik maakt van difoonconcatenatie (van Rijnsoever, 1988). Daarmee kunnen nederlandse, engelse en duitse teksten hoorbaar worden gemaakt vanuit een terminal of vanuit tekstfiles op disk. Het tweede systeem, DECtalk, is in hardware uitgevoerd en kan engelse teksten (amerikaans) uitspreken door middel van allofoonsynthese via regels (Digital, 1985a). DECtalk kan als standalone systeem (off line) worden gebruikt om tekst, ingetikt op een direct aangesloten terminal, uit te spreken. Daarbij ontbreekt echter de mogelijkheid om die tekst te editen. In het spraakonderzoek moet DECtalk op een andere manier worden gebruikt, n.l. on-line, waarbij de communicatie via een host computer verloopt, in ons geval een microVAX-II. Daarvoor moet nog speciale software worden ontwikkeld en die ontwikkeling vormt het hoofdonderdeel van deze stage.

Een belangrijke stap in het proces van tekst naar kunstmatige spraak is de omzetting van de gewone ASCII tekst (incl. afkortingen, cijfers enz.) in een fonetische representatie. DECtalk kan de bij synthese gebruikte foneemreeks zichtbaar maken en uitgeven. De DECtalk grafeem-foneemomzetter is dus in principe beschikbaar voor de toepassing in bijv. het IPO tekst-naar- spraaksysteem DS. "Inbouwen" van deze DECtalk grafeem-foneemomzetter in DS vormt het tweede onderdeel van de stage.

Samenvattend bestaat de stage-opdracht dus uit de volgende delen:

1. Ontwikkeling van software om DECtalk online via een microVAX-II (deel uitmakend van het IPO-VAX-cluster) teksten uit te laten spreken, die, na inloggen op een willekeurige machine in het cluster, hetzij op een willekeurige terminal worden ingetikt, dan wel afkomstig zijn van ASCII-tekst files die op disk staan.

2. Voor zover de tijd het toelaat: ontwikkelen van een module voor grafeem-foneem-omzetting, in te bouwen in het bestaande IPO tekst-naar-spraak systeem DS, gebruik makend van de grafeem-foneem-omzetter in DECtalk.

4 Korte beschrijving van DECtalk

DECtalk is een systeem dat, geheel automatisch, tekst omzet in amerikaans engels spraakgeluid. Het berust op allofoonsynthese door regels, in hoofdzaak ontwikkeld door Klatt (Allen et al, 1987). Invoer voor het systeem is een ASCII-tekst, die eerst wordt omgezet in een "verrijkte" foneemrepresentatie. Daarmee worden de parameters bestuurd van een digitaal "bron-filter" model ("formant synthesizer") voor spraaksynthese.

4.1 Tekstinvoer

Om te worden uitgesproken moet de aan DECtalk toegevoerde tekst worden afgesloten met een punt, vraag- of uitroepteken. Ontbreekt zo'n "terminator", dan zal de tekst pas na een time-out van 5 seconden ten gehore worden gebracht.

Behalve gewone tekst zijn er nog twee manieren om DECtalk te besturen: door foneem-spelling en door commando's. Beide moeten tussen vierkante haken in de tekst worden aangegeven.

De volgende foneemsymbolen (max. twee letters per foneem) kunnen worden gespecificeerd (zie ook tabel C1 van (Digital, 1985a)):

Vowels:

fAther	aa	bAt	ae	About	ax	bAR	ar	bOAt	ow
bOUght	ao	bIt	ih	kissEs	ix	bORe	or	bOUt	aw
bEAt	iy	bUt	ah			bEAR	er	bIte	ay
lUte	uw	bEt	eh			bEER	ir	bAke	ey
bIRd	rr	bOOk	uh			pOOR	ur	bOY	oy
								cUte	yu

Consonants:

bottLE	el	Bet	b	Met	m	Test	t
ransOM	em	Fin	f	Net	n	THin	th
buttON	en	Guess	g	siNG	nx	Vest	v
		Head	hx	Pet	p	Wet	w
CHin	ch	Gin	jh	Red	r	Yet	y
Debt	d	Ken	k	Sit	s	Zoo	z
THis	dh	Let	l	SHin	sh	leiSure	zh

Van de commando's die in de ingevoerde tekst kunnen worden aangebracht zullen we de belangrijkste in de volgende sectie noemen; voor details wordt verwezen naar (Digital, 1985a, hoofdstuk 5).

Voor een 70-tal woorden kan een alternatieve uitspraak worden gekozen door in de tekst direct vóór het betreffende woord een ronde sluithaak te plaatsen. Zo wordt bijv. bij intikken van het woord

```
abstract
```

het woordaccent op de eerste syllabe gelegd maar bij

op de tweede. Andere woorden waarvoor een alternatieve uitspraak kan worden aangegeven zijn opgesomd in (Digital, 1985a, tabel B1). Verder is er nog de mogelijkheid om een gebruikers-lexicon aan te leggen van "eigen" woorden, afkortingen en acroniemen, met hun foneemrepresentatie.

Tenslotte kunnen duur en toonhoogte van individuele fonemen worden veranderd. Daartoe moet in de tekst het foneemsymbool worden gevolgd door twee getallen tussen hoekhaken < en > en gescheiden door een komma. Het eerste getal specificeert de duur in ms van het foneem, het tweede de toonhoogte in Hz (tussen 50 en 300). Toonhoogte-getallen lager dan 50 hebben een speciale betekenis: waarden van 1 t/m 37 representeren de tonen C2 t/m C5 (64 t/m 512 Hz), voorzien van "vibrato". Daarmee kunnen dan melodietjes gespecificeerd worden. Verdere details staan in (Digital, 1985a, pag 54 e.v.).

4.2 De belangrijkste commando's voor DECtalk

De volgende commando's kunnen in de tekst worden gebruikt:

```
M.b.t. spreeksnelheid en pauzes (gegeven zijn de default waarden):

[:ra 180]    "rate of speaking" in woorden per minuut (120-300)
[+]          "paragraph pause"
[:cp 160]    "comma pause" in ms (0-32767)
[:pp 640]    "period pause" in ms (0-32767)
[_]          "silence" phoneme

M.b.t. prosodie:

[/] [\] [/\]          pitch rise, fall, both
["] ['] [`]           emphatic, primary secondary lexical stress
[-] [*] [#]           syllable, morpheme, compound boundary
[ ] <return> <tab>    word boundary
[(] [)]               begin, end of relative clause
[,]                   end of clause  (same as written comma)
[.] [?] [!]           end of normal, question, exclam. sentence
[+]                   new paragraph

Verder is het mogelijk uit een aantal synthese-stemmen te kiezen:

commando  stemnaam            omschrijving

[:nb]     Beautiful Betty     standaard vrouwelijke stem
[:nr]     Rough Rita          diepe vrouwelijke stem
[:nu]     Uppity Ursula       lichte vrouwelijke stem
[:nw]     Whispering Wendy    fluisterende vrouwelijke stem

[:nk]     Kit the Kid         kinderstem
```

```
[:np]     Perfect Paul        standaard mannelijke stem
[:nh]     Huge Harry          diepe mannelijke stem
[:nf]     Frail Frank         stem van oudere man
[:nd]     Doctor Dennis       fluisterende mannelijke stem

[:nv]     Variable Val        zelf in te stellen stem
```

Alle tekst na zo'n commando wordt uitgesproken door die nieuwe stem. Het is dus raadzaam de tekst steeds te laten beginnen met een specificatie van de gewenste stem, anders wordt die van de vorige DECtalk-gebruiker toegepast.

Bovenstaande stemmen kunnen gemodificeerd worden door na het commando voor de stemkeuze het commando [:dv] te gebruiken, gevolgd door een specificatie van de te wijzigen stemparameter. Daarbij mogen meerdere commando's en parameters binnen een set haken worden geplaatst:

```
[:ap 120]     "average pitch" in Hz (50-300)
[:pr 100]     "pitch range" in % (0-250), monotoon is dus 0
[:sex 0]      "sex" (0 of f: vrouw, 1 of m: man)
[:hs 100]     "head size" in % (75-150)
[:br 0]       "breathiness" in dB (0-60)
[:sm 20]      "smoothness" in dB (0-24)
[:la 0]       "laryngealization" in % (0-100)
[:fo 100]     "forte" in % (0-100)
[:g1 60]      "gain 1" in dB (0-80) (reductie van "squawk")
[:g2 60]      "gain 2" in dB (0-80) (reductie van "squawk")

               etc.
```

Voor een complete lijst van commando's en nadere details wordt verwezen naar (Digital 1985a, pag 64 e.v.). In appendix A is een voorbeeld gegeven van tekst waarin een aantal van de hier genoemde commando's is opgenomen.

5 Handleiding voor het gebruikersprogramma DCT

DCT is het programma dat de communicatie verzorgt tussen de gebruiker, ingelogd op een van de vier hostcomputers die deel uitmaken van het IPO-VAX- cluster en DECtalk, aangesloten op de IPOMV4. Het is in staat om teksten, uit diskfiles of direct na intikken op de terminal, naar DECtalk te versturen. Om de uitgesproken zinnen te beluisteren moet worden afgestemd op kanaal 7 van het audio FM-net, waaraan de audio-uitgang van DECtalk is aangesloten. De syntax voor de aanroep van DCT is:

```
$ DCT tekstfilenaam
```

Indien men een filenaam opgeeft zal DCT controleren of dit een correcte file is en zal vervolgens de inhoud oversturen naar DECtalk. Tijdens de spraakuitgifte vanuit een file drukt DCT de tekst af op het scherm. De uitgifte kan voortijdig worden gestopt door een willekeurige toets in te drukken. Het kan dan nog wel even duren voordat DECtalk uitgepraat is omdat de de inhoud van buffer die "onderweg is" afgemaakt wordt. Indien men bij de aanroep van DCT geen filenaam opgeeft wordt erom gevraagd.

Als ook daarop geen filenaam wordt ingetikt gaat DCT over in "terminal mode" en kunnen teksten vanaf het toetsenbord worden ingetikt. Die worden dan uitgesproken als ze eindigen op ? ! of . Voor verdere details over de in te tikken tekst en commando's voor DECtalk wordt naar het vorige hoofdstuk verwezen. "Helpinformatie" kan worden opgevraagd door

```
/h
```

in te tikken; na intikken van

```
/l
```

wordt een lijst van foneemsymbolen op het scherm weergegeven Ook bij tekstinvoer via het toetsenbord van de terminal is het mogelijk spraakuitgifte voortijdig te stoppen, door tijdens het spreken een willekeurig toets in de drukken. Als bij de prompt om tekst een lege regel (alleen CR) wordt gegeven, wordt het programma DCT verlaten[1].

In appendix B zijn de DECtalk instellingen opgesomd die voor het programma DCT van belang (kunnen) zijn.

[1] Na afloop van de stage zijn door Vogten nog enkele aanvullingen in DCT aangebracht (niet vermeld in de originele listing van Appendix C) zodat DCT voor de gebruikers compatibel is met de overige LVS-programma's.

6 Beschrijving van de ontwikkelde programmatuur

Er zijn in drie modules ontwikkeld ten behoeve van de besturing van DECtalk:

`DCT.PAS`

Dit is het "clusterwide" gebruikersprogramma dat op iedere machine van het IPO-VAX-cluster is te gebruiken. Het stuurt een tekstbestand, afkomstig van een willekeurige disk of van het toetsenbord, via Ethernet naar de IPOMV4, alwaar het continu draaiend proces SPEECH_7 de data ontvangt en via de de procedure DECTALKOUT doorstuurt naar DECtalk, die ze uitpsreekt. DCT.PAS maakt gebruik van een fortranprocedure, SPRDECTALK.FOR, die de interface vormt tussen DCT.PAS en de LIO-procedure SPEAK$PROCEDURE. Deze laatste is de "algemene" LIO-routine die "clusterwide" de spraakuitgifte verzorgt voor alle D-A converters van VAXLAB, VAXDAC en IPOMV4.

`DECTALKOUT.PAS`

Dit is de module die, in het continu draaiend proces SPEECH_7 op de IPOMV4, in feite de communicatie tussen IPOMV4 en DECtalk verzorgt. Ze ontvangt in een "inputbuffer" data die, vanuit een willekeurige machine van het IPO-VAX-cluster, afkomstig zijn van de routine SPEAK$PROCEDURE en stuurt de vervolgens door naar DECtalk.

Als argumenten heeft DECTALKOUT.PAS behalve het genoemde "inputbuffer" nog een tweede buffer, het "outputbuffer", waarmee in principe data teruggestuurd kunnen worden naar de gebruiker. Dit is echter nog niet in gebruik.

DECTALKOUT.PAS zorgt er ook voor dat bij opstarten van het proces SPEECH_7 op de IPOMV4, de benodigde eventflags en de terminallijnverbindingen TXA0 en TXA1 met DECtalk worden geassigneerd.

`DECTALK.PAS`

Dit is een experimenteel programma waarmee DECtalk direct bestuurd kan worden via de terminallijn TXA0. Een voorwaarde voor het gebruik van DECTALK.PAS is dat men is ingelogd op de IPOMV4 en de terminallijn TXA0 aldaar geassigneerd kan worden. DECTALK.PAS gebruikt een tekstbestand dat aanwezig moet zijn in de file genaamd DCTBEST.TXT. Het programma is door mij gebruikt om de besturing van DECtalk te oefenen, alvorens me te wagen aan moeilijker onderdelen van de opdracht. Het kan (alleen na inloggen op de IPOMV4) DECtalk besturen zonder gebruikmaking van de "clusterwide" LIO-software.

Onderstaande fig. 1 geeft schematisch het verband weer tussen de verschillende hardware elementen hierbovengenoemd. In fig. 2 is de onderlinge relatie tussen de genoemde software modules geschetst.

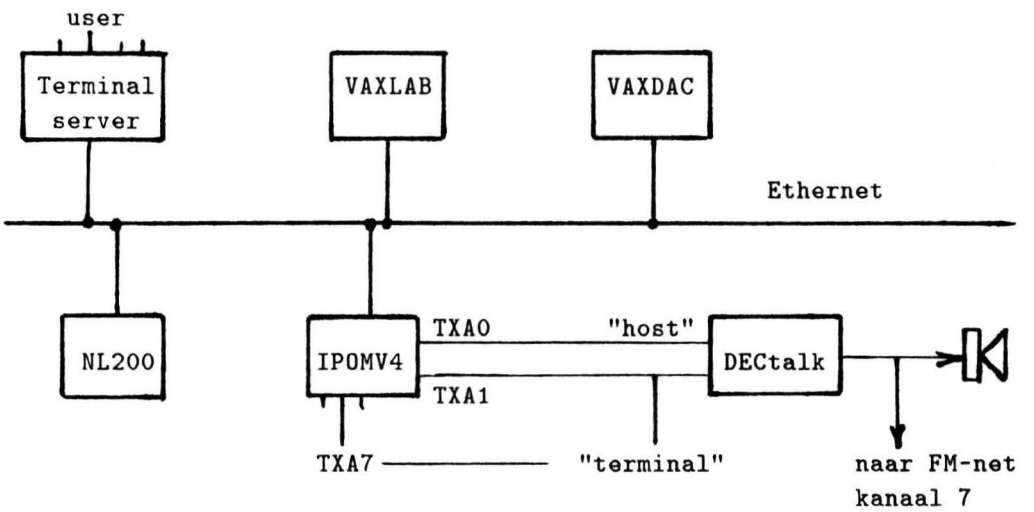

Fig. 1. Hardware-configuratie van de VAX-clusteromgeving waarin DECtalk is opgenomen.

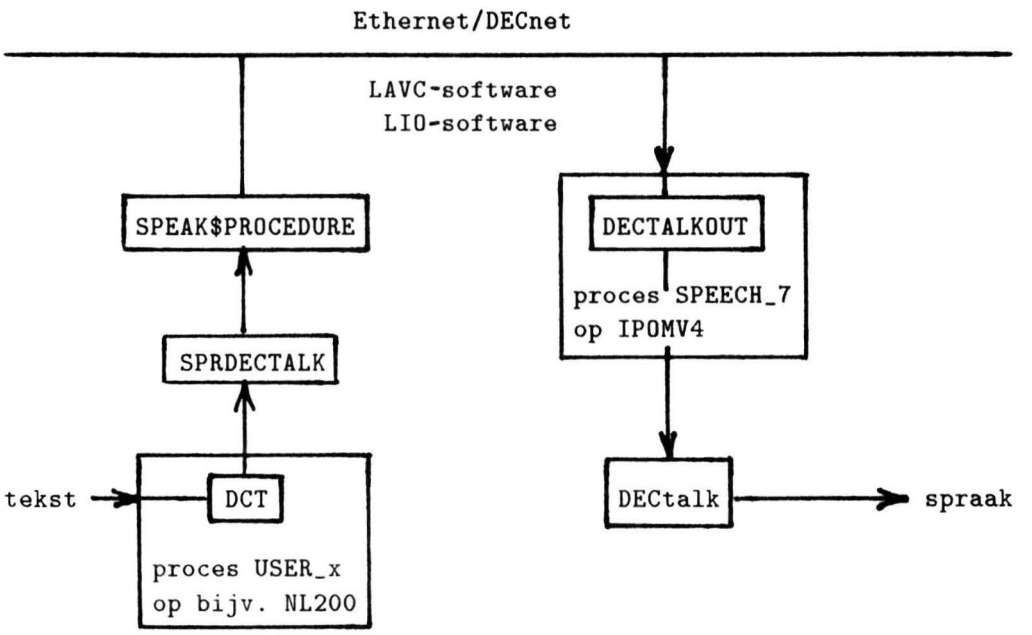

Fig. 2. De modules die bij het besturen van DECtalk in een VAX-VMS-clusteromgeving worden gebruikt.

6.1 DCT.PAS

DCT.PAS bevat de tekst van het gebruikersprogramma DCT; van een viertal procedures ervan wordt hierna een korte beschrijving gegeven.

6.1.1 De procedures

```
procedure BestandNaarBuffer (var F:text; var BUFFER:bigbuftype);
```

Deze procedure leest tekst vanuit file F in BUFFER. Hierbij wordt na elke ingelezen regel een extra carriage return (ASCII 13) toegevoegd, omdat die anders tijdens de verzending van de bufferinhoud automatisch worden toegevoegd. Dit gebeurt dan meestal op de verkeerde plaats, met als gevolg dat soms woorden verkeerd worden uitgesproken. Er wordt telkens gekeken of de maximale bufferlengte (de constante MAXBUFLEN) van 16 kilobytes wordt overschreden. Zo ja, dan wordt daar een mededeling over gedaan en wordt er gestopt met inlezen. BUFFER bevat in dit geval uiteraard niet het gehele bestand.

```
procedure ParseBuffer (var BUFFER:bigbuftype; var ZIN:buffertype);
```

Deze procedure haalt van de tekst in BUFFER een voorstuk af dat eindigt op een zogenaamde delimiter, een van de karakters ".", ",", "!" en "?". Het voorstuk wordt in de variabele ZIN geplaatst. Tevens wordt ervoor gezorgd dat de lengte van de varying ZIN altijd even is. Dit is nodig omdat bij aanroep van SPEAK$PROCEDURE het aantal over te sturen spraaksamples moet worden opgegeven en een "sample" twee karakters kan bevatten. Ergo moet in DCT de halve lengte van de over te sturen string worden opgegeven. De stringlengte wordt even gemaakt door, zo nodig, de string aan te vullen met een extra spatie. Aan het eind van elke zin wordt nog een carriage return toegevoegd om de eerdergenoemde ongewenste automatische toevoeging te vermijden.

```
procedure Schrijf (ZIN:buffertype);
```

In deze procedure wordt er een zin zoals afgesplitst door ParseBuffer op de terminal weergegeven in een leesbare layout. Daarbij wordt aan de af te drukken CR (carriage return)-karakters een Line Feed toegevoegd met een writeln en zo de overgang naar het begin van de volgende regel geforceerd.

```
procedure Trim (var ZIN:buffertype);
```

Deze procedure verwijdert eventueel aanwezige overbodige spaties aan het begin en het eind van een zin.

6.1.2 Tegengekomen problemen

Het grootste probleem dat ik gedurende het testen van de oorspronkelijke versie van DCT ben tegengekomen is dat er nog geen data correct teruggestuurd kunnen worden van DECtalk (in on-line mode) naar de IPOMV4 c.q. de gebruiker elders ingelogd in het cluster. Binnen de (te) korte stageperiode heb ik helaas niet de tijd gehad om dit probleem aan te pakken. Het bleek ook niet mogelijk om "clusterwide" met DCT escape sequences naar DECtalk te sturen. Er is ergens onderweg in de communicatie

waarschijnlijk een programma dat deze sequences probeert te interpreteren, terwijl dat niet mag. Welk programma (of misschien een systeemroutine?) dat is heb ik niet precies kunnen achterhalen, maar ik vermoed dat het hier SPEAK$KBINT betreft (stond in de verder zeer cryptische foutmelding). Het probleem treedt onder andere op als je een DT_STOP sequence (Digital, 1985b) stuurt. Dit resulteert na het verzenden van een aantal regels in een foutmelding.

Ook het feit dat er soms CR's werden toegevoegd reken ik tot een van de problemen. Dit probleem is dus omzeild door "op tijd" CR's toe te voegen.

6.2 DECTALKOUT.PAS

DECTALKOUT.PAS is een module die slechts procedures bevat. De belangrijkste daarvan is DECTALKOUT. Alle andere procedures zijn lokaal ten opzichte van DECTALKOUT. Deze module wordt gebruikt in het (continu draaiende) proces SPEECH_7 op de IPOMV4, waaraan DECtalk gekoppeld is.

```
procedure DECTALKOUT (var SPKPAR: [readonly] speak$parameters;
                      DEV_CHAN: [readonly] int2;
                      SPR_UIT: [readonly,volatile] boolean);
```

Deze procedure verzorgt de communicatie tussen DECtalk en IPOMV4. Als de procedure voor de eerste keer (d.w.z. na opstarten van het proces SPEECH_7 op de IPOMV4) wordt aangeroepen, worden er eventflags gereserveerd voor de QIOW-operaties en communicatielijnen geassigneerd met DECtalk. Vervolgens wordt DECtalk in de toestand na power-up gezet. Daarna herhaalt de procedure telkens de cyclus van het ontvangen van data op DEV_CHAN en het vervolgens doorsturen naar DECtalk.

Communicatie in de omgekeerde richting, dus ontvangst van data van DECtalk en vervolgens het doorsturen daarvan, terug naar DCT, is nog niet operationeel. Dat deel is in de programma-tekst tussen commentaarhaken gezet zodat men hiermee eventueel kan experimenteren.

6.2.1 De procedures

```
procedure GeefFoutMelding (cv:integer);
```

Deze procedure geeft de bij een bepaalde foutcode behorende booschap weer in een logfile die hoort bij het proces SPEECH_7 dat de spraakuitgifte via DECtalk op de IPOMV4 regelt.

```
procedure AssigneerLijn (var OK:boolean);
```

Hierin wordt een lijn geassigneerd met DECtalk. Indien dit gelukt is is OK true, anders is false.

```
procedure VerstuurData (DATA:buffertype; ZENDKANAAL:integer);
```

Deze procedure stuurt de in de variabele DATA staande data over de door ZENDKANAAL gespecificeerde terminallijn door naar DECtalk en wacht tot de operatie is afgerond. Als er iets mis is gegaan dan wordt er een foutmelding gegeven.

```
procedure OntvangData (var DATA:buffertype; ONTVANGSTKANAAL:integer);
```

In deze procedure wordt er van de door ONTVANGSTKANAAL gespecificeerde terminallijn data gelezen en in DATA geplaatst, totdat er een terminator zoals gespecificeerd in termin is gelezen of er een timeout heeft plaatsgevonden.

```
procedure InitDECtalk;
```

Deze procedure zorgt ervoor dat DECtalk na power-up in de juiste positie wordt gezet.

6.2.2 Tegengekomen problemen

Het grootste probleem dat ik heb moeten oplossen was dat er door een QIOW-operatie blokken data (karakters) van ten hoogste 1996 bytes mogen worden verstuurd. Grotere blokken leveren een systeem-foutmelding. Vermoedelijk wordt dit veroorzaakt door het feit dat de IPOMV4-terminallijn TXA0 (voor DECtalk) voor gebruik met een "VT100 device" staat geschakeld en dat om een of andere reden hierbij geen blokken groter dan 1996 bytes mogen worden verstuurd. Dit "probleem" is in de praktijk opgelost door gebruik te maken van kleinere blokken bij de verzending van data.

Een ander probleem was de discrepantie tussen de declaratie van een FORTRAN logical en een PASCAL boolean. De betreffende FORTRAN logical is ter grootte van 4 bytes gedeclareerd terwijl een PASCAL boolean maar 1 byte groot is. Omdat ik een 1 op 1 vertaling had gedaan waren alle data in de variabele spkpar 3 bytes verschoven en leverde zodoende verminkte gegevens op. Dit heb ik opgelost door bij de vertaling voor een FORTRAN logical een PASCAL integer te nemen. Een "nettere" oplossing zou zijn om in de bestaande LIO-software de betreffende FORTRAN logical te declareren met logical*1.

6.2.3 Tips en hints voor verdere ontwikkeling

Het is zeer belangrijk om ervoor te zorgen dat de records one_speech_channel en speak$parameters overeenstemmen met de structuren zoals gedefinieerd in LIO_DECLARE_COM.FOR van de LIO-software. Mocht men ooit belang hebben bij het versturen van blokken groter dan 1996 bytes dan kan men proberen TXA0 anders te laten schakelen, misschien als een "real time device" zoals bij de D/A converters op de VAXLAB en VAXDAC.

6.3 DECTALK.PAS

Het hoofdprogramma van DECTALK.PAS is nagenoeg hetzelfde als dat van DCT.PAS. Het enige verschil bestaat hierin dat de uit te spreken tekst nu direct van IPOMV4 naar DECtalk wordt gestuurd en niet via de LIO-software en Ethernet. Voorwaarde is uiteraard dat men is ingelogd op de IPOMV4. De gebruikte procedures zijn reeds beschreven bij de bespreking van DCT.PAS.

6.3.1 De tegengekomen problemen

Het belangrijkste probleem dat ik ben tegengekomen is dat het ook bij directe besturing van DECtalk vanuit de IPOMV4 nog niet mogelijk is gebleken om tevens data terug te sturen van DECtalk naar de IPOMV4. Pogingen hiertoe leiden tot verminkingen van de door IPOMV4 verzonden data (niet van de ontvangen data voor zover ik kon nagaan). Het versturen gaat een poosje goed maar op een gegeven moment gaat het mis en blijft het mis gaan. DECtalk gaat dan uiteraard vreemde dingen uitspreken en stukken tekst overslaan. Het lijkt erop dat er door DECtalk meer data ontvangen wordt dan er in het buffer kunnen worden opgeslagen. Een oplossing heb ik niet kunnen vinden.

7 Conclusies

1. Het belangrijkste deel van de stage-opdracht zoals omschreven in hoofdstuk 2 is met succes voltooid. Met het ontwikkelde programma DCT kunnen gebruikers, ingelogd op een van de vier IPO VAX-cluster machines, DECtalk besturen, zowel direkt vanaf het toetsenbord als vanuit een (te specificeren) tekstfile op disk.

2. Het tweede deel van de opdracht, "inbouwen" van de DECtalk grafeem-foneem-omzetting in het bestaande IPO programma DS voor spraaksynthese via difonen, is nog niet gereed. Bij "offline" ingestelde DECtalk verschijnt wel de foneemrepresentatie via de DECtalk-terminallijn op de (locale) terminal. Maar bij "online" bedrijf via het programma DECTALKOUT, draaiend op de IPOMV4, kan het (door DECtalk) ontvangen van tekst via IPOMV4-lijn TXA0 nog niet worden gecombineerd met het (door DECtalk) versturen van de foneemsymbolen via lijn TXA1 naar de IPOMV4. Daarbij treden nog problemen op. Pas als die zijn opgelost kan geprobeerd worden om, na verdere aanpassing van de bestaande LIO-software, de foneemrepresentatie via het DECnet "clusterwide" terug te sturen naar de gebruiker. En pas als dat lukt, kan in het programma DS de (real-time) grafeem-foneem-omzetting van DECtalk worden gebruikt.

8 Referenties

Allen, J., Hunnicutt, M.S., Klatt, D., Armstrong, R.C. and Pisoni, D.B. (1987) *From Text to Speech: The MITalk system*, Cambridge University Press, Cambridge.

Digital (1985a) Dectalk documentatie DTC01, Owner's Manual EK-DTC-OM-003.

Digital (1985b) Dectalk documentatie DTC01, Programmer's Reference Manual EK-DTC-RM-003.

Rijnsoever van, P.A. (1988) "A multilingual text-to-speech system", IPO Ann. Prog. Rep. 23, p 34-40.

9 Appendix A. Tekstvoorbeeld voor DECtalk

```
[:np].
Hello, I am Perfect Paul of the family of ["]DEC talkers. We live
here with 10 persons. Let me introduce the others.
[+]. First here, in this room we have Betty.
[+ :nb].
Hi, I'm beatiful Betty, and I love being beautiful. Would you hear my
contribution to the international vocal contest in sertogen[']bos
this year? Listen. [lay<1000,40>][lao<2000,30>][nay<1000,37>].
[:np].
Let's go! Living next to ["]her is Harry. Be careful with him because
he can become very aggressive.
[+ :nh].
Hi, how are you doing? I am huge Harry and I am also a singer. Read
my lips! [ay<1000,10>][uh<2000,30>][iy<2000,15>].
[+ :np].
On this side of the hall we have Frank. Better be silent, because
Frank is a bit ill at the moment.
[+ :nf].
Good evening. My name is frail Frank. Nice to meet you. Please close
the door firmly when you go.
[+ :np].
Upst[']airs live Rita and Dennis and their son Kit.
[+ :nr].
Well, hello mister! Would you like to [/]marry me?
[:nd].
Oh! Stop it Rita. I've had enough of it!
[:nr].
Sorry dear Dennis. Won't happen again.
[:nd].
Ok. Here is my son Kit. Kit, say hello to the mister!
[:nk].
Hello mister. Do you have some [/]candy for me?
[+ :np].
Allright. Let's go to the basement. The basement is entirely owned by
Wendy. Wendy has some voice problems, so she can only whisper.
[:nw].
Hello. I am whispering Wendy. Meet my robot Charles.
[:np :dv ap 180 pr 0].
Hello. I am a robot, as you will understand. I take care of Wendy.
Most of the time, I speak for her, because, as you've heard, she
can't speak very well. Too much smoking, and smoking is bad for your
health.
[+ :np].
Ok. Now you've met all people, I will show you out. Bye bye.
See you ur[']ound.
```

10 Appendix B. De DECtalk instellingen

DECtalk kan zich bevinden in een van drie standen:

- de SETUP stand

Deze stand is alleen te bereiken vanaf een terminal die direct is aangesloten op de terminaluitgang van DECtalk. Door op deze terminal de BREAK toets in te drukken, kan men de diverse settings van DECtalk, indien gewenst, veranderen. De betekenis van deze settings staat in detail beschreven in (Digital, 1985a, pag 23 e.v.).

Voor DCT, het "clusterwide" besturingsprogramma voor DECtalk, is het van belang dat de settings als volgt staan:

```
SET LOG TEXT OFF
        PHONEME OFF
        RAWHOST OFF
        INHOST OFF
        OUTHOST OFF
        ERROR OFF
        TRACE OFF
        DEBUG OFF

SET HOST SPEED 9600
         FORMAT NONE
         SPEAK ON
         MODEM OFF

SET LOCAL SPEED 9600
          FORMAT NONE
          HOST OFF
          SPEAK OFF
          EDITED OFF
          HARDCOPY OFF
          SPOKENSETUP OFF
          FILTER OFF

SET MODE SQUARE ON
         ASKY OFF
         MINUS OFF

SET MASK OFF
SET INTERRUPT OFF
```

Na power-up heeft DECtalk altijd deze settings (geprogrammeerd in user non-volatile RAM).

Het is mogelijk om de acties van de host en DECtalk te bestuderen door een terminal rechtstreeks op de terminaluitgang van DECtalk aan te sluiten en vervolgens een of meerdere SET LOG settings te veranderen. Men kan dan op die locale terminal zien wat er over en weer gebeurt. Als men in de SETUP modus verkeert

kunnen de settings worden opgeslagen in het niet-vluchtig geheugen, zodat DECtalk iedere keer bij het opstarten deze settings gebruikt. Dit kan door het commando SAVE te geven. Blijft dit commando achterwege dan zullen, na het uitschakelen van DECtalk (of na het opstarten van het proces SPEECH_7 op de IPOMV4, de host waaraan DECtalk vast is verbonden) de oude settings weer gelden. Men kan de SETUP modus verlaten door het commando EXIT te geven.

- de ON-LINE stand

Dit is de stand waarin DECtalk altijd (na power-up) opstart. In deze mode kan men DECtalk besturen vanuit de host, in ons geval de IPOMV4, door ASCII tekst te sturen en/of commando's te geven via escape sequences in die tekst. Deze sequences staan gedocumenteerd in (Digital, 1985b). Daarbij zijn er ook sequences die antwoord terug behoren te geven maar het is mij (met het programma DECTALK.PAS, locaal draaiend op de IPOMV4) nog niet gelukt deze antwoorden inderdaad correct te verkrijgen. Ik vermoed dat deze sequences ergens in de communicatieverbinding worden afgevangen door een of ander systeemprogramma.

- de OFF-LINE stand

Dit is de stand die gebruikt moet worden om DECtalk via een terminal aan te sturen. Men kan DECtalk in deze stand zetten door in de SETUP modus het commando OFFLINE te geven.

11 Appendix C. De source teksten

```
(*****************************************************************************)
(*                                                                           *)
(* Copyright (c) 1990  Instituut voor Perceptie-Onderzoek                    *)
(*                     Eindhoven                                             *)
(*                     The Netherlands                                       *)
(*                                                                           *)
(* Dit programma verzorgt de communicatie tussen de gebruikers van DECtalk   *)
(* en het proces SPRDECTALK                                                  *)
(*                                                                           *)
(* Geschreven door Harald Diesveld, gedurende zijn stage                     *)
(* van 3-12-1990 tot en met 21-12-1990                                       *)
(*                                                                           *)
(*****************************************************************************)

[inherit ('sys$library:starlet','LVSENV')]

program DCT(input,output);

const MAXBUFLEN = 1024;      {Maximale lengte per keer te sturen met $QIOW     }
      BIGBUFLEN = 16384;     {Maximale lengte in totaal te sturen door DCT     }
      CR        = CHR(13);   {ASCII Carriage Return                            }
      LF        = CHR(10);   {ASCII Line Feed                                  }
      ESC       = CHR(27);   {ASCII Escape                                     }
      VERSIE    = '1.0';     {Versienummer van DCT                             }

type  string     = varying [82] of char;
      buffertype = varying [MAXBUFLEN] of char;
      bigbuftype = varying [BIGBUFLEN] of char;

var   klaar                : boolean;
      fnaam                : varying [64] of char;
      dclfname             : filename;  {eventuele filenaam van DCL commandline}
      zin,uitbuf           : buffertype;
      inbuf                : bigbuftype;
      kara                 : int1;      {bevat ASCII-waarde van ingetoets karak.}
      lzin,luitbuf         : integer;   {lengte zin en *maximale* lengte uitbuf }
      ntlout               : integer;   {lengte van de van terugontvangen tekst }
      f                    : text;
      viadcl               : boolean;
      dummy1,dummy2,dummy3 : filename;


procedure BestandNaarBuffer(var f:text;var buffer:bigbuftype);
{pre : fnaam is de naam van een bestaand bestand wat gebruikt kan worden
 post: fnaam is in buffer gezet
}

var  regel : string;
     klaar : boolean;

begin
   reset(f);
   buffer:='';
   klaar:=false;
   while (not eof(f)) and (not klaar) do
   begin
      readln(f,regel);
      if buffer.length + regel.length < BIGBUFLEN
      then buffer:=buffer + regel + CR {voeg CR toe na iedere regel}
      else begin
              klaar:=true;
              writeln;
              writeln('File too long. Input truncated.');
              writeln
           end
```

```
   end;
end;{BestandNaarBuffer}

procedure ParseBuffer(var buffer:bigbuftype;var zin:buffertype);

const  DELIMITERS = ['.' , ',' , '!' , '?'];

var  index      : integer;
     voorwaarde : boolean;

begin
   index:=1;
   zin:='';
   if index < buffer.length
   then voorwaarde:=not(buffer[index] in DELIMITERS) or
                    (buffer[index + 1] in DELIMITERS)
   else voorwaarde:=false;
   {voorwaarde zorgt ervoor dat er wordt doorgelezen totdat er een
    delimiter is gelezen waarachter geen andere delimiter staat
   }
   while (index < buffer.length) and voorwaarde do
   begin
      zin:=zin + buffer[index];
      index:=index + 1;
      if index < buffer.length
      then voorwaarde:=not(buffer[index] in DELIMITERS) or
                       (buffer[index + 1] in DELIMITERS)
      else voorwaarde:=false;
   end;
   zin:=zin + buffer[index] + CR;
   buffer:=substr(buffer,index + 1,buffer.length - index);
                                                  {verwijder zin van de buffer}
   if length(zin) mod 2 = 1 then zin:=zin + ' '   {maak zin met even lengte   }
end;{ParseBuffer}

procedure Schrijf(zin:buffertype);

var  i : integer;

procedure Trim(var zin:buffertype);
{pre : true
 post: zin begint en eindigt niet met een spatie
}

begin
   while zin[1] = ' ' do zin:=substr(zin,2,zin.length - 1);
   while zin[zin.length] = ' ' do zin:=substr(zin,1,zin.length - 1);
end;{Trim}

begin
   Trim(zin);
   for i:=1 to zin.length do
      if zin[i] <> CR
      then write(zin[i])
      else writeln
end;{Schrijf}


{begin van body van DCT}

begin
   writeln;
   writeln('DCT version ',VERSIE);
   writeln('Developed by Harald Diesveld. (c) 1990 IPO.');
   writeln('-----------------------------------------');
```

```
   writeln('DECtalk will speak on channel 7.');
   writeln('-----------------------------------------');
   writeln;
   klaar:=false;
   kara:=0;
   repeat
      viadcl:=DCLFLN(dclfname,dummy1,dummy2,dummy3); {is filename meegegeven?}
      if viadcl
      then fnaam:=dclfname
      else begin
             writeln;
             write('_File : ');
             readln(fnaam);
             writeln;
             writeln
           end;
      if fnaam.length <> 0
      then begin
             open(f,file_name:=fnaam,history:=readonly,
                    access_method:=sequential,error:=continue);
             if status(f) <> 0
             then begin
                    case status(f) of
                        3 : write('FILE NOT FOUND : ');
                        4 : write('INVALID FILE NAME : ');
                      116 : write('FILE IS NOT A TEXT FILE : ');
                      119 : write('WRITEONLY FILE : ');
                      otherwise write('ERROR OPENING FILE : ')
                    end;
                    writeln(fnaam)
                  end
             else begin
                    BestandNaarBuffer(f,inbuf);
                    while (inbuf.length <> 0) and (kara = 0) do
                    begin
                       ParseBuffer(inbuf,zin); {splits zin af van buffer}
                       Schrijf(zin);           {zet zin op beeldscherm  }
                       writeln;
                       lzin:=zin.length div 2; {vanwege speak$procedure}
                       uitbuf:='';
                       luitbuf:=MAXBUFLEN;
                       ntlout:=0;
                       SPRDECTALK(zin.body,lzin,uitbuf.body,luitbuf,
                                  ntlout,kara);
                       {indien kara <> 0 dan is er een toets gedrukt en moet
                        er worden gestopt}
                    end;
                    close(f);
                    klaar:=true
                  end
           end
      else begin
             writeln('Type the text to be spoken :');
             writeln;
             inbuf:='';
             repeat
                 zin:='';
                 readln(zin);
                 if zin.length <> 0 then inbuf:=inbuf + CR + zin
             until zin.length = 0;
             while (inbuf.length <> 0) and (kara = 0) do
             begin
                ParseBuffer(inbuf,zin);  {splits zin af van buffer}
                Schrijf(zin);            {zet zin op beeldscherm  }
                writeln;
```

```
                lzin:=zin.length div 2;  {vanwege speak$procedure }
                uitbuf:='';
                luitbuf:=MAXBUFLEN;
                ntlout:=0;
                SPRDECTALK(zin.body,lzin,uitbuf.body,luitbuf,
                           ntlout,kara);
             end;
             klaar:=true;
          end
   until klaar
end.{DCT}
```

```
! Audio output of TEXT via DECKTALK.


! INBUF[*]   in,b    buffer that contains the characters to be sent
! NTLIN      in,I4   number of characters to be sent
! OUTBUF[*]  out,b   buffer where answer from DECTALK will be written
! LOUTBUF    in,I4   number of characters that can be stored in OUTBUF
! NTLOUT     out,I4  number of characters written to OUTBUFFER by routine
! KARA  out,b      byte to return ASCII code of character eventually typed
!                  to stop speech before the end; returns 0 if no character typed


        subroutine SPRDECTALK (inbuf, ntlin, outbuf, loutbuf, ntlout, kara)

        implicit none
        byte      inbuf(*), outbuf(*)
        integer*4 ntlin, ntlout, itype, ichn, loutbuf,j
        byte      kara
        real      sfr, filter
        logical   stereo, kbint, dsp_spk

        ichn = 7                    ! at  IPO, DECTALK is on channel 7
        stereo = .false.            ! dummy
        sfr = 10000.                        ! must be > 1000 for speak$procedure
        filter = 6.                         ! dummy
        itype = 1                   ! > 0: text file,  no N-file
        kbint  = .true.                     ! keyboard interrupt allowed
        dsp_spk = .true.            ! display message "speaking"

        call speak$procedure (0, 0, inbuf,
        1        ntlin, sfr, itype, ichn, stereo, filter, kbint, dsp_spk, kara)

        end
```

```
(*************************************************************************)
(*                                                                       *)
(* Copyright (c) 1990  Instituut voor Perceptie Onderzoek                *)
(*                     Eindhoven                                         *)
(*                     The Netherlands                                   *)
(*                                                                       *)
(* Deze module verzorgt de communicatie met DECtalk                      *)
(*                                                                       *)
(* Geschreven door Harald Diesveld, gedurende zijn stage                 *)
(* van 3-12-1990 tot en met 21-12-1990                                   *)
(*                                                                       *)
(*************************************************************************)

[inherit ('LVSENV','sys$library:starlet')]

module DECTALKOUT(input,output);

const ANTWTIJD  = 4;          {Wacht 4 seconden op antwoord                }
      MAXBUFLEN = 1024;       {Maximale lengte per keer te sturen met $QIOW }
      MAXPOGING = 100;        {Maximaal # pogingen om te alloceren         }
      CR        = CHR(13);    {ASCII Carriage Return                       }
      ESC       = CHR(27);    {ASCII Escape                                }


      {DECtalk escape sequences}
      RIS         = ESC + 'c';
      DT_STOP     = ESC + 'P0;10z' + ESC + '\';

type  ioblk     = record
                    io_stat  : int2;     {i/o blok, status}
                    count    : int2;     {telt buffer tot terminator}
                    dev_info : integer   {informatie over device}
                  end;
      buffertype = varying [MAXBUFLEN] of char;


{ BELANGRIJK!!!!! DE NU KOMENDE TYPEDEFINITIES MOETEN OVEREENSTEMMEN MET DE    }
{ STRUCTUUR ZOALS GEDECLAREERD IN LIO_DECLARE_COM.FOR                          }

      par3     = packed array [1..3] of char;
      par6     = packed array [1..6] of char;
      par8     = packed array [1..8] of char;
      par12    = packed array [1..12] of char;
      par20    = packed array [1..20] of char;

      one_speech_channel =
         record
            channel_number : integer; {the number in the whole system       }
            node           : par6;    {where it is installed                }
            task           : par12;   {declared by control process          }
            device         : par8;    {physical name in computer            }
            device_aux     : par8;    {name of clock for example            }
            system         : par3;    {'LIO', 'DSC', ...                    }
            stereo_channel : boolean; {True or False                        }
            dachan_max     : integer; {D/A chan.: 2 on LIO, 2 on DSC        }
            dachan_used    : integer; {1 for mono, 2 for stereo             }
            dachan_list_16 : integer; {0, 1 (LIO); 8, 9 (DSC)               }
            resolution     : integer; {12 or 16-bit samples                 }
            associated_chan: integer; {in case 2 dachan & not stereo        }
            maxsfr         : real;    {maximum sampling frequency           }
            filter         : real;    {0 if variable, else value            }
            filter_factor  : real;    {0.48 means 0.48*sfr if filter=0      }
            filter_IEEE_dev: par8;    {device through which code is sent    }
            filter_IEEE_adr: integer; {adress of filter on IEEE bus         }
            filter_chan    : integer; {which channel on filter itself       }
```

```
            general_info    : par20     {anything you need to specify          }
         end;

         speak$parameters =
            record
               version      : integer; {must be incremented with new version }
               ntlsmpls     : integer;
               sfreq        : real;
               itype        : integer;
               stereo       : integer; {was oorspronkelijk een boolean       }
               filter       : real;
               sp           : one_speech_channel
            end;


[global] procedure DECTALKOUT(var spkpar   : [readonly] speak$parameters;
                              var dev_chan : [readonly] int2;
                              var spr_uit  : [readonly,volatile] boolean);


var  kanaal1      : [static] int2;      {kanaalnummer van DECtalk                }
     kanaal2      : [static] int2;      {kanaalnummer van de DECtalk terminal    }
     ok           : [static] boolean:=true; {allocatie succesvol verlopen?       }
     poging       : integer;            {hoeveelste allocatiepoging?             }
     buffer       : buffertype;         {input/output buffer                     }
     eerste_keer  : [static] boolean:=true;       {is het allereerste aanroep? }
     ef1,ef2      : [static] integer;             {eventflags voor $QIOW         }

     [external] procedure lib$get_ef
                    (%ref eflags : integer);external;

procedure GeefFoutmelding(cv:integer);

begin
   printmess('ERROR : ',cv);
   writeln
end;{GeefFoutmelding}

procedure AssigneerLijn(var ok:boolean);
{pre : er zijn nog geen kanalen geassigneerd
 post: (ok & (er zijn kanalen geassigneerd voor DECtalk en de terminal)) or
       not ok
}

var  sysi : integer; {returncode voor $ASSIGN aanroep}

begin
   sysi:=$ASSIGN(spkpar.sp.device,kanaal1);
{   if kanaal1 <> 0
   then sysi:=$ASSIGN(spkpar.sp.device_aux,kanaal2);
}   ok:=(kanaal1 <> 0) {and (kanaal2 <> 0)};
   if kanaal1 = 0
   then writeln('There is no line to assign to DECtalk.');
{   if kanaal2 = 0
   then writeln('There is no line to assign to the DECtalk terminal.')}
end;{AssigneerLijn}

procedure DeassigneerLijn;
{pre : true
 post: de twee geassigneerde lijnen zijn gedeassigneerd
}

var sysi : integer;  {algemene returncode van $DASSGN operatie}

begin
```

```
   sysi:=$DASSGN(kanaal1);
   sysi:=$DASSGN(kanaal2)
end;{DeassigneerLijn}

procedure VerstuurData(data:buffertype;zendkanaal:integer);
{pre : data bevat de te versturen data
 post: (odd(VerstuurData) & data verstuurd over kanaal1) or even(VerstuurData)
}

var iowblk : ioblk;     {returncodes van de I/O operatie       }
    sysi   : integer;   {algemene returncode van $QIOW operatie}

begin
   sysi:=$QIOW( efn  := ef1,
                chan := zendkanaal,
                func := IO$_WRITEVBLK,
                iosb := iowblk,
                p1   := data.body,
                p2   := data.length);
   if odd(sysi) then sysi:=iowblk.io_stat;
   if not odd(sysi)
   then begin
           writeln('ERROR OCCURRED IN PROCEDURE VERSTUURDATA : ',sysi);
           GeefFoutmelding(sysi)
        end
end;{VerstuurData}

procedure OntvangData (var data:buffertype;ontvangstkanaal:integer);
{pre : true
 post: (odd(OntvangData) & data is zojuist ontvangen) or
       (even(OntvangData)  & er is geen data binnengekomen)
}

var  iorblk : ioblk;                    {returncodes van de I/O operatie}
     termin : array [1..2] of integer; {definitie van de terminator    }
     sysi   : integer;                  {algemene returncode van $QIOW operatie }

begin
   termin[1]:=0;          {definitie van de terminator}
   termin[2]:= %X'2000';  {definitie van de terminator}
   data:='';
   sysi:=$QIOW( efn  := ef2,
                chan := ontvangstkanaal,
                func := IO$_READVBLK + IO$M_TIMED, {doe read met timeout}
                iosb := iorblk,
                p1   := data.body,
                p2   := MAXBUFLEN,
                p3   := ANTWTIJD,
                p4   := %ref termin );
   if odd(sysi)
   then begin
           sysi:=iorblk.io_stat;
           data.length:=iorblk.count
        end;
   if not odd(sysi) and (sysi<>SS$_TIMEOUT)
   then begin
           writeln('ERROR OCCURRED IN PROCEDURE ONTVANGDATA : ',sysi);
           GeefFoutmelding(sysi)
        end
end;{OntvangData}

procedure InitDECtalk;
{pre : true
 post: DECtalk is geinitialiseerd
}
```

```
begin
   VerstuurData(DT_STOP,kanaal1);    {stop met spreken       }
   VerstuurData(RIS,kanaal1)         {Return to Initial State}
end;{InitDECtalk}



{begin van body van DECTALKOUT}

begin
   if eerste_keer then begin   {moet maar een keer gebeuren, bij eerste aanroep}
                          lib$get_ef(ef1);     {reserveer ef1 als flag         }
                          lib$get_ef(ef2);     {reserveer ef2 als flag         }
                          poging:=0;
                          ok:=false;
                          while (not ok) and (poging <= MaxPoging) do
                          begin
                             DeassigneerLijn;
                             poging:=poging + 1;
                             AssigneerLijn(ok)
                          end;
                          if ok then InitDECtalk;{Initialiseer DECtalk         }
                          eerste_keer:=false   {nu geen flags meer reserveren  }
                       end;
   if ok then begin
                 OntvangData(buffer,dev_chan);  {Lees data in van dev_chan    }
                 VerstuurData(buffer,kanaal1);  {Stuur data door naar DECtalk}
     {            OntvangData(buffer,kanaal2);       }
     {            VerstuurData(buffer,dev_chan);     }
             end
        else writeln('FATAL ERROR OCCURRED : LINE(S) COULD NOT BE ALLOCATED.')
end;{DECTALKOUT}

end.{module DECTALKOUT}
```

```
(*****************************************************************************)
(*                                                                           *)
(* Copyright (c) 1990  Instituut voor Perceptie-Onderzoek                    *)
(*                     Eindhoven                                             *)
(*                     The Netherlands                                       *)
(*                                                                           *)
(* Dit programma is gebruikt om de I/O operaties te testen die uiteindelijk  *)
(* zijn gebruikt in de module DECTALKOUT.PAS                                 *)
(*                                                                           *)
(* Geschreven door Harald Diesveld, gedurende zijn stage                     *)
(* van 3-12-1990 tot en met 21-12-1990                                       *)
(*                                                                           *)
(*****************************************************************************)

[inherit ('LVSENV','sys$library:starlet')]

program DECtalk(input,output);

const MAXBUFLEN = 1024;      {Maximale lengte per keer te sturen met $QIOW     }
      BIGBUFLEN = 16384;     {Maximale lengte in totaal te sturen door DECTALK }
      CR        = CHR(13);   {ASCII Carriage Return                            }
      LF        = CHR(10);   {ASCII Line Feed                                  }
      ESC       = CHR(27);   {ASCII Escape                                     }
      VERSIE    = '1.0';     {Versienummer van DECTALK                         }
      ANTWTIJD  = 4;         {Wacht 4 seconden op antwoord van DECtalk         }
      MAXPOGING = 100;       {Maximaal aantal pogingen om te assigneren        }

      DECtalk         = 'TXA0:';   {Terminalnummer van DECtalk HOST ingang     }
      DECtalkTerminal = 'TXA1:';   {Terminalnummer van de DECtalk TERMINAL uitg}

      {DECtalk escape sequences}
      DSR         = ESC + '[n';
      RIS         = ESC + 'c';
      DT_LOG      = ESC + 'P0;81;' + CHR(10) + 'z' + ESC + '\ ';
      DT_STOP     = ESC + 'P0;10z' + ESC + '\';
      DT_TERMINAL = ESC + 'P0;82;' + CHR(0) + 'z' + ESC + '\ ';

type  ioblk     = record
                    io_stat  : int2;     {i/o blok, status}
                    count    : int2;     {telt buffer tot terminator}
                    dev_info : integer   {informatie over device}
                  end;
      string    = varying [82] of char;
      buffertype = varying [MAXBUFLEN] of char;
      bigbuftype = varying [BIGBUFLEN] of char;


var   kanaal1              : int2;        {kanaalnummer van DECtalk             }
      kanaal2              : int2;        {kanaalnummer van de DECtalk terminal }
      ok                   : boolean;     {allocatie succesvol verlopen?        }
      poging               : integer;     {hoeveelste allocatiepoging?          }
      eerstekeer           : boolean;     {is het de allereerste aanroep?       }
      klaar                : boolean;
      fnaam,fnaam2         : varying [64] of char;
      dclfname             : filename;
      zin,uitbuf           : buffertype;
      inbuf                : bigbuftype;
      kara                 : int1;
      f                    : text;
      viadcl               : boolean;
      dummy1,dummy2,dummy3 : filename;

procedure GeefFoutmelding(cv:integer);

begin
```

```
   printmess('ERROR : ',cv);
   writeln
end;{GeefFoutmelding}

procedure AssigneerLijn(var ok:boolean);
{pre : er zijn nog geen kanalen geassigneerd
 post: (ok & (er zijn lijnen geassigneerd voor DECtalk en de terminal)) or
       not ok
}

var  sysi : integer;   {returncode voor $ASSIGN aanroep}

begin
   sysi:=$ASSIGN(DECtalk,kanaal1);
  { if kanaal1 <> 0
   then sysi:=$ASSIGN(DECtalkTerminal,kanaal2);}
   ok:=(kanaal1<>0) {and (kanaal2<>0)};
   if kanaal1 = 0
   then writeln('There is no line to assign to DECtalk.');
 {  if kanaal2 = 0
   then writeln('There is no line to assign to the DECtalk terminal.')}
end;{AssigneerLijn}

procedure DeassigneerLijn;
{pre : er zijn twee lijnen geassigneerd voor DECtalk en zijn terminal
 post: de twee geassigneerde lijnen zijn gedeassigneerd
}

var sysi : integer;  {algemene returncode voor $DASSGN aanroep}

begin
   sysi:=$DASSGN(kanaal1);
   sysi:=$DASSGN(kanaal2);
end;{DeassigneerLijn}

procedure VerstuurData(data:buffertype);
{pre : data bevat de te versturen data
 post: (odd(sysi) & data verstuurd over kanaal1) or even(sysi)
}

var iowblk : ioblk;      {returncodes van de I/O operatie        }
    sysi   : integer;    {algemene returncode voor $QIOW operatie}

begin
   sysi:=$QIOW( chan := kanaal1,
                func := IO$_WRITEVBLK,
                iosb := iowblk,
                p1   := data.body,
                p2   := data.length);
   if odd(sysi) then sysi:=iowblk.io_stat;
   if not odd(sysi)
   then begin
          writeln('ERROR OCCURRED IN PROCEDURE VERSTUURDATA : ',sysi);
          GeefFoutmelding(sysi);
        end;
end;{VerstuurData}

procedure OntvangData (var antw:buffertype;ontvangstkanaal:integer);
{pre : true
 post: (odd(sysi) & antw is zojuist ontvangen) or
       (even(sysi)  & er is geen antwoord binnengekomen)
}

var  iorblk : ioblk;                    {returncodes van de I/O operatie       }
     termin : array [1..2] of integer; {definitie van de terminator           }
```

```
     sysi   : integer;                {algemene returncode voor $QIOW operatie}

begin
   termin[1]:=0;          {definitie van de terminator}
   termin[2]:= %X'2000';  {definitie van de terminator}
   antw:='';
   sysi:=$QIOW( chan := ontvangstkanaal,
                func := IO$_READVBLK + IO$M_TIMED, {doe read met timeout}
                iosb := iorblk,
                p1   := antw.body,
                p2   := MAXBUFLEN,
                p3   := ANTWTIJD,
                p4   := %ref termin );
   if odd(sysi)  then    sysi:=iorblk.io_stat;
   antw.length:=iorblk.count;
   if not odd(sysi) and (sysi<>SS$_TIMEOUT)
   then begin
           writeln('ERROR OCCURRED IN PROCEDURE ONTVANGDATA : ',sysi);
           GeefFoutmelding(sysi)
        end
end;{OntvangData}

procedure BestandNaarBuffer(var f:text;var buffer:bigbuftype);
{pre : f is een geopend bestand
 post: de inhoud van f is in buffer gezet
}

var  regel : string;
     klaar : boolean;

begin
   reset(f);
   buffer:='';
   klaar:=false;
   while (not eof(f)) and (not klaar) do
   begin
      readln(f,regel);
      if buffer.length + regel.length < BIGBUFLEN
      then buffer:=buffer + regel + CR  {voeg na iedere regel CR toe}
      else klaar:=true
   end;
end;{BestandNaarBuffer}

procedure ParseBuffer(var buffer:bigbuftype;var zin:buffertype);

const  DELIMITERS = ['.' , ',' , '!' , '?'];

var  index      : integer;
     voorwaarde : boolean;

begin
   index:=1;
   zin:='';
   if index < buffer.length
   then voorwaarde:=not(buffer[index] in DELIMITERS) or
                    (buffer[index + 1] in DELIMITERS)
   {voorwaarde zorgt ervoor dat er wordt doorgelezen totdat er een delimiter
    is gelezen waarachter geen delimiter staat}
   else voorwaarde:=false;
   while (index < buffer.length) and voorwaarde do
   begin
      zin:=zin + buffer[index];
      index:=index + 1;
      if index < buffer.length
      then voorwaarde:=not(buffer[index] in DELIMITERS) or
```

```
                        (buffer[index + 1] in DELIMITERS)
      else voorwaarde:=false;
   end;
   zin:=zin + buffer[index] + CR;
   buffer:=substr(buffer,index + 1,buffer.length - index)
end;{ParseBuffer}

procedure InitDECtalk;
{pre : true
 post: DECtalk is geinitialiseerd
}

begin
{   VerstuurData(DT_LOG);     {Zet LOG PHONEME ON       }
   VerstuurData(DT_STOP)     {Stop met spreken         }
end;{InitDECtalk}



{begin van body van DECTALK}

begin
   writeln;
   writeln('DECTALK version ',VERSIE);
   writeln('Developed by Harald Diesveld. (c) 1990 IPO.');
   writeln('-----------------------------------------');
   writeln('DECtalk will speak on channel 7.');
   writeln('-----------------------------------------');
   writeln;
   poging:=0;
   ok:=false;
   while (not ok) and (poging <= MAXPOGING) do
   begin
      DeassigneerLijn;
      poging:=poging + 1;
      AssigneerLijn(ok);
   end;
   if ok
   then begin
           writeln;
           klaar:=false;
           repeat
              viadcl:=DCLFLN(dclfname,dummy1,dummy2,dummy3);
                   {als viadcl dan bevat dclfname een filenaam die is
                    meegegeven vanuit DCL }
              if viadcl
              then fnaam:=dclfname
              else begin
                      writeln;
                      write(' File : ');
                      readln(fnaam);
                      writeln;
                      writeln
                   end;
              if fnaam.length <> 0
              then begin
                      open(f,file_name:=fnaam,history:=readonly,
                             access_method:=sequential,error:=continue);
                      if status(f) <> 0
                      then begin
                              case status(f) of
                                  3 : write('FILE NOT FOUND : ');
                                  4 : write('INVALID FILE NAME : ');
                                116 : write('FILE IS NOT A TEXT FILE : ');
                                119 : write('WRITEONLY FILE : ');
```

```
                              otherwise write('ERROR OPENING FILE : ')
                            end;
                            writeln(fnaam)
                         end
                    else begin
                            BestandNaarBuffer(f,inbuf);
                            while (inbuf.length <> 0) do
                            begin
                               ParseBuffer(inbuf,zin);
                               writeln(zin);
                               VerstuurData(zin);
{                               OntvangData(uitbuf,kanaal2);}
{                               writeln(uitbuf)}
                            end;
                            close(f);
                            klaar:=true
                         end
               end
            else begin
                    writeln('Type the text to be spoken :');
                    writeln;
                    inbuf:='';
                    repeat
                        zin:='';
                        readln(zin);
                        if zin.length <> 0 then inbuf:=inbuf + CR + zin
                    until zin.length = 0;
                    while (inbuf.length <> 0) do
                    begin
                       ParseBuffer(inbuf,zin);
                       writeln(zin);
                       VerstuurData(zin);
{                      OntvangData(uitbuf,kanaal2);}
{                       writeln(uitbuf)}
                    end;
                    klaar:=true;
                 end
          until klaar;
          DeassigneerLijn
       end
   else writeln('FATAL ERROR OCCURRED : LINE(S) COULD NOT BE ALLOCATED.')
end.
```