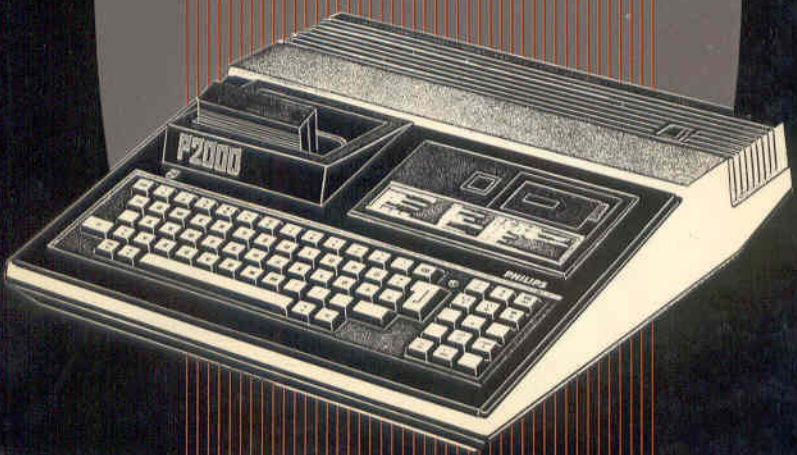
PHILIPS

P2000

PHILIPS

GEBRUIKSAANWIJZING
P2000T MET
P2305 BASIC NL

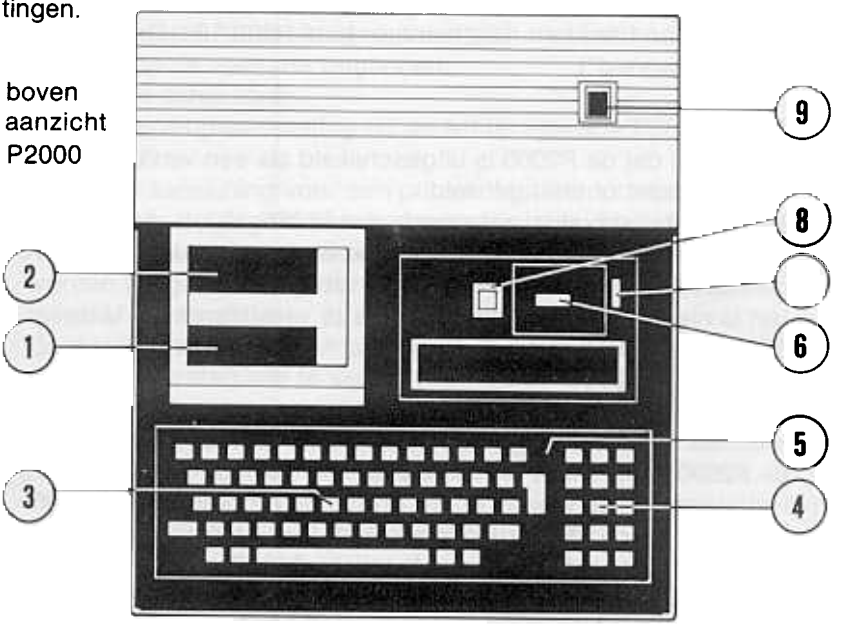


INHOUD

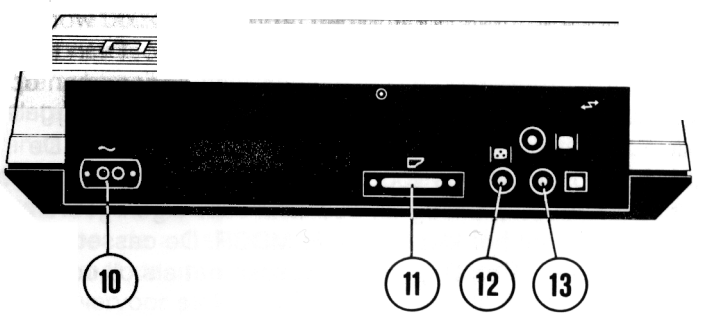
1 PLAATSEN VAN DE P2000

De P2000 bestaat uit een kast die er ongeveer uitziet als een draagbare schrijfmachine. Om er mee te werken is het nodig hem op een TV-toestel aan te sluiten. Op het scherm van dit TV-toestel verschijnt dan de informatie. Dit kan een zwart-wit-TV of een kleuren-TV zijn. Alleen in het laatste geval hebt u profijt van de kleurmogelijkheden van de P2000.
Zoek een plaatsje voor de P2000 op een tafeltje in de buurt van het TV-toestel zodanig dat u gemakkelijk op het scherm kunt kijken als u achter het toetsenbord zit.
Op de P2000 vindt u de volgende bedieningsorganen en aansluitingen.

boven aanzicht P2000



achteraanzicht P2000



Links boven het toetsenbord zit een schuifdeksel met daar onder twee sleuven.

①In sleuf 1 behoort een insteekdoos of -module waarin een programma is vastgelegd. De „Basic Interpreter 16K" is zo'n insteekmodule. Op de doos staat het cijfer 1 ten teken dat deze module in sleuf 1 behoort. Met verschillende modules kan de P2000 geheel verschillende taken uitvoeren. Met de „Basic Interpreter 16K" in sleuf 1 kan de P2000 worden aangesproken in de eenvoudig te leren programmeertaal BASIC.

②Sleuf 2 is bestemd voor een insteekmodule met verbindingen naar buiten. Via deze verbindingen kan men de P2000 allerlei dingen laten besturen. (zie hiervoor hoofdstuk 18: „De P2000 en de buitenwereld")

Belangrijk!

* Zorg altijd dat de P2000 is uitgeschakeld als een van de modules wordt ingezet of eruit gehaald.
* Zorg dat het schuifdeksel steeds zo ver mogelijk is dicht geschoven. Zo wordt voorkomen dat er stof en stukjes metaal in de sleuven vallen.
* Het is niet nodig de modules steeds te verwijderen als U de P2000 even niet gebruikt.

③Het toetsenbord komt overeen met dat van een schrijfmachine. Hiermee geeft u uw eigen informatie door aan het inwendige van de P2000.

④Het kleine toetsenbordje is erg prettig bij programma's, waarin veel cijfers moeten worden ingevoerd. Ook bevindt zich op het kleine toetsenbord een aantal functietoetsen.

⑤Het rode indicatielampje geeft aan of de P2000 in bedrijf is.

⑥De Mini Digitale Cassette Recorder (MDCR) kan een klein model cassette bevatten waarop gegevens en programma's uit de P2000 kunnen worden opgenomen. Zodra de P2000 wordt uitgeschakeld „vergeet" hij namelijk alles. Daarom moet van te voren het ingetikte programma of bestand worden opgenomen op zo'n kleine cassette. Later kan alles dan weer worden „teruggelezen"! Bij de P2000 zijn twee van deze minicassettes verpakt. De BASIC-demo cassette is voorbespeeld met enkele programma's, de andere cassette is leeg en bestemd voor eigen gebruik.

⑦Dit schuifje opent het klepje van de MDCR. De cassettes kunnen aan twee zijden, A en B, gebruikt worden net als bij compactcassettes.

Belangrijk!

* Het is onverstandig om de P2000 uit te schakelen wanneer zich nog een cassette in de recorder bevindt. Eerst even het klepje laten openspringen.
* Zorg dat het klepje gesloten is als de P2000 niet gebruikt wordt. Dit voorkomt dat er stof in het mechaniek van de recorder valt.

⑧De zwarte rechthoekige knop gemerkt R is de RESET-knop. Als hierop gedrukt wordt vergeet de P2000 alles en begint weer van voren af aan. Met de Basic Interpreter 16K in sleuf ① en een minicassette met programma's in de MDCR wordt na het drukken op de RESET-knop automatisch met het eerste programma op de cassette begonnen.

⑨De aan/uit-schakelaar.

⑩De netspanningsaansluiting op de achterzijde. De P2000 werkt op 220 volt.

⑪Dit is een aansluiting voor een printer. Met een printer aangesloten op de P2000 is het mogelijk gegevens of programma's op papier af te drukken. Er kunnen diverse merken printers worden aangesloten op deze RS 232-C connector. Raadpleeg hiervoor Uw handelaar.
Op deze connector zijn in plaats van een printer ook diverse andere apparaten aan te sluiten (zie hoofdstuk 18).

⑫Een aansluiting voor een monitor. Dit kan zowel een zwart-wit monitor als een RGB kleurenmonitor zijn.

⑬De aansluiting voor een TV-toestel, zowel zwart-wit als kleur, op TV-kanaal 35.

Aansluiten van de P2000

Plaats de P2000 in de nabijheid van het TV-toestel. De Basic Interpreter 16K bevindt zich niet in sleuf ① en er zit nog geen cassette in de MDCR. Bij de P2000 bevinden zich twee aansluitsnoeren. Het ene is het netsnoer, nodig om de microcomputer aan te sluiten op het stopcontact. Plaats het ene einde van dit snoer in de aansluiting ⑩ op de achterzijde van de P2000. Steek het andere einde in het stopcontact, kies hiervoor als het enigszins mogelijk is een type met randaarde. Als nu de aan/uit-schakelaar ⑨ wordt ingedrukt gaat het indicatielampje ⑤ op het toetsenbord aan.
Het andere snoer is bestemd om het TV-toestel aan te sluiten. Hiermee moet de aansluitbus ⑬ achter op de P2000 worden verbonden met de antenne aansluiting van de TV. Oude typen TV's gebruiken hiervoor stekertjes in plaats van de moderne ronde

aansluitbus. In dit geval is een aanpassingsdoosje nodig, soms is het aansluitsnoer van de centrale antenne hiervoor te gebruiken. Zoek nu met de afstemming van de TV het beeld van de P2000 op. Dit is te vinden op de UHF, ongeveer op TV-kanaal 35. Wat u moet zien is een zwart scherm met daarop de tekst:

P H I L P S

MICROCOMPUTER

P2000

In lichtblauw, of lichtgrijs als u een zwart-wit TV gebruikt. Zet de afstemming van de P2000 niet op het hoogste voorkeurskanaal. Dit is speciaal bestemd voor een videorecorder en de P2000-beelden zullen hierop minder stabiel zijn. Raadpleeg eventueel de gebruiksaanwijzing van het TV-toestel voor het afstemmen en het bewaren van de afstemming.

Insteken van de Basic module
Schakel de P2000 uit door op de schakelaar ⑨ te drukken. Het TV beeld zal nu verdwijnen. Open het schuifdeksel en plaats de doos „Basic Interpreter 16K" nu in sleuf ① zodanig dat de tekst op de bovenkant leesbaar is. Met lichte druk en enig bewegen „pakt" de connector en is de doos er niet meer zonder trekken uit te halen. Schuif het dekseltje dan weer zo ver mogelijk dicht.
Schakel de P2000 weer in met de aan/uit-schakelaar ⑨. Uit de luidspreker van de TV klinkt een pieptoontje en na enkele ogenblikken staat op het scherm de tekst:

PHILIPS CASSETTE BASIC
Versie 1.1 NL
14966 bytes vrij
Ok

Uw P2000 is nu in staat BASIC-programma's te verwerken.
In hoofdstuk 2 staat hoe u nu de programma's van de BASIC-demo cassette kunt gebruiken.

Andere beeldschermen aansluiten
De ingebouwde UHF modulator geeft een goed beeld op een TV-toestel. Er kan echter een behoorlijke verbetering in scherpte

van de beelden worden verkregen bij gebruik van een monitor in plaats van een TV-toestel. Een monitor is een beeldweergever zonder ontvangstmogelijkheid. Vergelijk dit maar met een versterker met luidsprekerboxen waarmee zonder tuner geen radio te ontvangen is. Monitoren worden aangesloten op de DIN-aansluiting ⑫. Het is heel goed mogelijk zowel een TV-toestel als een monitor tegelijkertijd te gebruiken.

RGB monitoren
Het beste resultaat geeft een z.g. RGB-kleurenmonitor. Hierbij worden de stuursignalen voor de drie basiskleuren rood, groen en blauw via afzonderlijke draden van de P2000 naar de monitor geleid. Het beeld is dan scherper en zuiverder van kleur.
Moderne TV monitoren (en een enkel TV-toestel) hebben een RGB ingang op een z.g. SCART AV-plug. Met een speciale kabel kan de SCART-plug worden verbonden met de DIN-aansluiting ⑫ op de P2000. Raadpleeg uw handelaar voor het benodigde verbindingssnoer.

Zwart-wit monitoren
Onder het typenummer V7001 is een kleine Philips zwart-wit monitor verkrijgbaar speciaal geschikt voor de P2000. De monitor wordt aangesloten op de DIN-aansluiting ⑫. Kleuren worden omgezet in grijstinten en de scherpte is aanzienlijk beter dan bij een TV. Hierbij wordt tevens de verbinding gemaakt met de in deze monitor ingebouwde geluidsversterker.

2. BASIC-DEMO CASSETTE

Bij de P2000 behoort een BASIC-demo cassette waarop een aantal BASIC programma's staat, bedoeld om de mogelijkheden van deze microcomputer te laten zien.

Cassette inlezen
Zet de P2000 in BASIC zoals in hoofdstuk 1 is beschreven. Open nu het klepje van het cassetterecordertje en plaats de BASIC-demo cassette met zijde A boven. Doe dit zodanig dat de tekst leesbaar is; alleen in dat geval kunt u het klepje sluiten.
Sluit dit klepje en druk op de RESET-knop ⑧. De tekst verdwijnt even van het scherm, komt weer terug en de cassette gaat automatisch lopen. Na enig heen en weer zoeken op de cassette verschijnt een z.g. keuze-menu op het scherm.

BASIC-DEMO cassette zijde A:

1 Kleurendemonstratie
2 Sluit in
3 U hangt (woord raden)
4 Adressenboek

>GEEF NUMMER VAN GEWENST PROGRAMMA<

Door een van de toetsen 1, 2, 3 of 4 in te drukken kunt u een keuze maken tussen de vier programma's op deze zijde. Als u het eens bent met de keuze drukt u op de ENTER-toets, dat is de grote toets rechts op het grote toetsenbord met het teken
De cassette gaat nu spoelen en de P2000 haalt het gevraagde programma van de cassette.

Dit zijn de programma's op zijde A
Kleuren demonstratie geeft een indruk van de grafische mogelijkheden en kleur van de P2000. Kiest u in dit programma voor „ophouden" dan komt u weer terug bij het keuze-menu.

Sluit in is een bordspel, ook wel Othello of Reversie genaamd. U kunt dit spelen tegen de P2000 in twee moeilijkheidsgraden.

U hangt is een woordraadspel. Speciaal kinderen zijn hier dol op.

Adressenboek is een adressenbestand in BASIC waarin u kunt zoeken op elk woord of deel van een woord. U moet iets afweten van het gebruik van de Editor van BASIC om zelf gegevens te kunnen inbrengen (zie hoofdstuk 11).
Als u dit programma wilt gaan gebruiken is het nodig het te kopiëren op een lege cassette. Hoe dat moet vindt u verderop in deze handleiding.

De andere zijde
Keer de BASIC-demo cassette om en druk op RESET. U krijgt dan het keuze-menu:

1 Huishoudboekje
2 Huishoudboekje uitleg
3 Viditel

U kunt nu weer uw keuze maken.

Dit zijn de programma's op zijde B
Huishoudboekje is een programma om de inkomsten en uitgaven van het huishoudgeld te administreren. Het is hierbij ook nodig het programma te kopiëren naar een lege cassette (zijde). Dit kan het programma zelf doen. Hoe alles in zijn werk gaat vindt u in het programma: **Huishoudboekje uitleg.**

Het programma **Viditel** maakt van uw P2000 een volwaardige Viditel terminal. Het enige dat u hierbij extra nodig hebt is de aansluiting op de telefoon (de modem). In het programma staat hoe één en ander moet worden aangesloten op de connector ⑪ voor de printer.

Fouten die kunnen optreden.

Cassette fout I	:	het bandje is gebroken.
Leesfout	:	de informatie op het bandje is mogelijk beschadigd. Het programma is niet foutloos gelezen. Probeer nog eens.
Geen cassette	:	het klepje is opengesprongen of u hebt de cassette er uit gehaald. Cassette terugplaatsen en op RESET drukken.
Niet gevonden	:	het gevraagde programma staat niet op deze cassette(zijde). Plaats de juiste cassette en druk op RESET.

Geen stopje : er moet iets op de cassette worden opgenomen terwijl die daartegen beschermd is. Plaats een zwart stopje in de linker bovenhoek van de minicassette of plaats een andere cassette.

Mocht u andere foutmeldingen tegenkomen, dan kunt u de verklaring hiervan vinden in tabel 4 van deze handleiding.

Copiëren van programma's

In hoofdstuk 10 wordt het laden en bewaren van programma's en bestanden behandeld. Wellicht zult u, vóórdat u dit hoofdstuk hebt gelezen, een programma van een ander willen overnemen. U handelt dan als volgt:

- Schakel de P2000 in.
- Plaats de cassette, waarop het programma dat u wilt copiëren bevindt, in de cassetterecorder.
- Sluit het klepje.
- Tik in
  CLOAD ***"naam programma"***
  en sluit af met de ENTER-toets ↵ . ***naam programma*** dient de naam te zijn waaronder het programma op de cassette moet worden opgezocht.
- U hoort nu het cassetterecordertje lopen. Wacht rustig tot op het beeldscherm
  Ok
  verschijnt.
- Verwijder de cassette en plaats uw eigen cassette.
- Tik nu
  CSAVE"*naam*" en sluit af met de ENTER-toets.
  naam hoeft niet gelijk te zijn aan de oorspronkelijke naam van het programma. U tikt de naam in, waaronder u het programma wilt bewaren.
- U hoort de cassetterecorder weer lopen. Wacht rustig tot de P2000 de letters
  Ok
  op het beeldscherm afdrukt.

Het programma staat nu op uw cassette en met de instructie
RUN"*naam*"
kunt u het later weer in uw P2000 laden en laten uitvoeren.

Opmerkingen: Sommige programma's bestaan uit meerdere delen, bijv. een programma en een gegevensbestand. De copieerprocedure voor dit soort programma's is ingewikkelder en wordt hier niet behandeld.
Er zijn programma's die niet meer te copiëren zijn als ze zijn ge-RUN-d. Houdt u zich strikt aan bovengenoemd schema en „bekijk" het ingeladen programma niet eerst even met RUN.

3. INLEIDING

In deze handleiding wordt een aantal onderdelen van de P2000 besproken en wordt de bediening van de computer uitgelegd. Er is niet naar gestreefd om alle details van het programmeren te beschrijven. Op den duur zult u merken, dat er meer mogelijk is met de P2000 dan in dit boekje beschreven is. In de verschillende hoofdstukken zal een aantal „vaktermen" worden gebruikt, die we in deze inleiding kort zullen verklaren.

De BASIC interpreter
De BASIC interpreter maakt het mogelijk de P2000 te programmeren in de programmeertaal BASIC. Een programma in BASIC bestaat uit instructies die ingetikt kunnen worden als gemakkelijk te herkennen codewoorden. De BASIC interpreter herkent deze codewoorden en „vertaalt" ze (vandaar de naam „interpreter") in een aantal handelingen die het beoogde effect hebben en voert deze uit. Staat er b.v. ergens in een programma het woord PRINT, dan zal de BASIC interpreter dit herkennen als de opdracht „Stuur alle tekst en getallen die nu volgen in leesbare vorm naar het beeldscherm". De interpreter beschikt hiervoor over een aantal routines, die automatisch worden opgezocht en uitgevoerd.
De interpreter is verpakt in een insteekdoos, die in sleuf 1 geplaatst moet worden. Ten teken, dat de P2000 BASIC opdrachten zal herkennen verschijnt op het scherm de tekst: PHILIPS CASSETTE BASIC (zie ook hoofdstuk 1: „Plaatsen van de P2000").

Toetsenbord
Aan de P2000 kunnen opdrachten gegeven worden d.m.v. het toetsenbord. Hoe dit toetsenbord bediend moet worden en welke speciale functies sommige toetsen vertegenwoordigen, staat beschreven in hoofdstuk 4: „Het toetsenbord".

Processor en geheugen
In de P2000 bevindt zich een zgn. microprocessor (Z80), de centrale rekeneenheid. Bij het programmeren in BASIC hoeven we van deze processor niets te weten. De microprocessor verzorgt het hele transport en de bewerking van alle gegevens, zoals dat door de BASIC interpreter wordt opgedragen.
Behalve de microprocessor bevat de P2000 een zgn. geheugen. In dit geheugen worden alle programma-instructies die uitgevoerd moeten worden bewaard, evenals alle resultaten van berekeningen etc. Een deel van dit geheugen heeft de P2000 zelf nodig voor de

„interne administratie". Voor het programma en de gegevensopslag zijn in een standaard P2000 14966 geheugenplaatsen vrij ter beschikking.
Om een geheugenplaats te kunnen terugvinden zijn alle geheugenplaatsen in de P2000 genummerd. Het nummer van de geheugenplaats heet in computertermen: het adres. In elke geheugenplaats, op elk adres dus, kan een getal geschreven worden van 0 t.m. 255. Via een coderingstabel (zie tabel 1, ASCII-waarden en tabel 6, BASIC-tokens) zijn alle letters, cijfers, leestekens en BASIC instructies genummerd en kunnen door de computer worden herkend. Het getal op één geheugenplaats noemen we een byte.

Cassetterecorder
Als de P2000 uitgeschakeld wordt zijn alle gegevens en programma's die in het geheugen stonden, gewist. Programma's en gegevensbestanden die we willen bewaren om later weer te gebruiken dienen dus uit het geheugen van de P2000 te worden „overgeschreven" naar een ander opslagmiddel. Bij de P2000 is dit een mini-cassette. Het overhevelen van de informatie naar de cassette gebeurt met de ingebouwde cassetterecorder. Via deze recorder kunnen gegevens en programma's uit het geheugen van de P2000 naar de band worden geschreven en ook weer van de band in het geheugen van de P2000 worden gelezen. De bediening van de cassetterecorder en het inlezen en wegschrijven van gegevens wordt beschreven in de hoofdstukken 9 en 10.

Tot slot van deze inleiding nog een paar opmerkingen over de hoofdstukken 19 t.m. 22. In deze hoofdstukken wordt besproken wat de BASIC interpreter in de P2000 kan en mag.
Hier gaat het om de precisie van getallen, waarheidstabellen bij logische operaties etc. Deze hoofdstukken zijn meer bedoeld als „naslagwerk", dan om BASIC te leren. Speciaal geldt dit voor hoofdstuk 22, dat een alfabetische lijst bevat van alle BASIC-instructies en BASIC-functies.

In een aantal tabellen aan het eind van deze handleiding wordt nog wat nuttige informatie samengevat, die verspreid in deze handleiding voorkomt.

4. TOETSENBORD

De P2000 beschikt over een groot toetsenbord, dat veel weg heeft van het toetsenbord van een normale schrijfmachine en een klein toetsenbord met cijfers en functies. Een plattegrond van het toetsenbord staat achterin deze handleiding.

Het grote toetsenbord
Enkele toetsen van het grote toetsenbord hebben een speciale functie.
Allereerst is dat de toets gemerkt ↵ , die we de ENTER-toets noemen (in de Engelse literatuur vaak Return-key genoemd). Met het indrukken van deze toets beëindigen we de invoer van opdrachten in de directe stand, de invoer van gegevens tijdens de uitvoering van een programma met INPUT en LINEINPUT en ook het intikken of wijzigen van een programmaregel in BASIC. Met de ENTER-toets geven we aan dat de P2000 iets moet doen met de ingetikte gegevens. Tik bijv. eens
PRINT 5+2 en sluit af met de ENTER-toets
7
Ok

De P2000 rekent dan de optelling 5+2 uit en geeft het resultaat op het scherm.
Het afsluiten van elke regel of instructie dient altijd met de ENTER-toets te gebeuren. In het vervolg van deze handleiding zal dit niet steeds worden vermeld.

De toetsen links en rechts van de spatiebalk noemen we cursor toetsen. Tijdens invoer van gegevens of programmaregels kan met deze toetsen de cursor* (positie aanwijzer) worden verplaatst, b.v. om fouten in de invoer te verbeteren alvorens invoer wordt beëindigd met ENTER. Soms hoort u, uit de luidspreker van de TV, een piepje bij het indrukken. Dit gebeurt als u de cursor probeert te verplaatsen buiten het reeds getikte deel van de regel.
De twee toetsen gemerkt ⇧ dienen om de bovenste rijen karakters, die op de toetsen zijn aangegeven te kunnen tikken. We noemen deze de SHIFT-toetsen. De SHIFT-toetsen dienen ook, zoals bij een schrijfmachine, om hoofdletters te typen i.p.v. kleine letters. Na het inschakelen van de P2000 of na het indrukken van de RESET-toets verschijnen bij het typen kleine letters op het scherm.

* Over de cursor, het witte vlakje dat aangeeft waar een ingetikte letter verschijnt, komen we te spreken in hoofdstuk 6.

De P2000 kan als volgt op alleen hoofdletters worden overgeschakeld: Druk een SHIFT-toets in en houd deze ingedrukt. Druk even op de toets gemerkt TAB. Laat de SHIFT-toets los.
Intypen van letters laat nu hoofdletters op het scherm verschijnen. Indrukken van de SHIFT is nu alleen werkzaam voor het intikken van de bovenste karakters op de toetsen. Terugschakelen naar kleine letters gaat op dezelfde manier (SHIFT-TAB).

De SHIFT-toets kan worden „vastgezet" door indrukken van de SHIFT-vergrendel toets. De P2000 reageert nu alsof de SHIFT-toets voortdurend ingedrukt wordt gehouden. Bovendien gaat het schrijven op het beeldscherm nu veel langzamer. Door één van de SHIFT-toetsen eenmaal in te drukken wordt de SHIFT-vergrendeling weer losgemaakt.
Verder is het nog van belang te weten:
SHIFT-@ zet op het scherm het teken ↑, dat gebruikt wordt voor machtsverheffen.
Rechtsboven op het grote toetsenbord is een toets gemerkt ⌫ , de **correctie-toets.** Indrukken van deze toets wist het voorgaande karakter en zet de cursor een plaats terug. Indrukken van deze toets te zamen met SHIFT wist het karakter onder de cursor.
De toets hiernaast, met de keine accentjes, geeft het symbool ¼ en met SHIFT het symbool ¾ op het scherm. Deze symbolen hebben echter niet de betekenis één kwart, resp. drie kwart en worden door de P2000 niet geaccepteerd in berekeningen. Deze toets kan bijv. worden gebruikt voor het intikken van bepaalde grafische tekens (zie Tabel 3). De toets hieronder, met de vierkante haken, toont pijltjes naar rechts en naar links op het scherm. SHIFT# geeft een wit blokje.

Het kleine toetsenbord
Op het kleine toetsenbord vinden we nog eens alle cijfers. Voor het snel ingeven van getallen is het handiger om het kleine toetsenbord te gebruiken. De komma, rechts-onder, is de decimale komma, die door de P2000 als een decimale punt op het scherm wordt gezet. Wordt tevens de SHIFT-toets ingedrukt, dan bevat het kleine toetsenbord een aantal functie-toetsen.

DEF	wijzigen van de laatst-gebruikte programma-regel (zie hoofdstuk 11)
☞	afdrukken van het scherm op papier via een printer

STOP	-	onderbreken van een lopend programma of LIST-ing
ZOEK	-	tonen van de inhoud van een cassetteband (zie hoofdstuk 9)
SHIFT-5	-	deel-PRINT aanzetten en vervolgens LIST-en vanaf het begin van het programmeren (zie hoofdstuk 5)
START	-	start uitvoering van het programma in het geheugen
INL OPN ⊙ ↩	}	niet in bedrijf bij BASIC
⊡	-	na verkregen toestemming wissen van cassetteband (zie hoofdstuk 9)
M	-	deel-PRINT aanzetten (zie bijv. hoofdstuk 5)
	-	teken voor de integer deling (÷)
-	-	aftrekteken
x	-	vermenigvuldigteken
+	-	optelteken
⊠	-	wis het beeldscherm (zie hoofdstuk 6)
⊢⊣	-	wis de tot zover ingetikte programmaregel (zie hoofdstuk 6)

RESET-knop.
Naast de cassetterecorder bevindt zich een verzonken toets, gemerkt R. Na het indrukken van deze toets is het gehele geheugen van de P2000 gewist. Alle gegevens en programma's in het geheugen gaan verloren (niet de programma's en gegevens op de cassetteband).
Wees dus uiterst voorzichtig met de RESET-toets.

ASCII-waarden
Op de plattegrond van de toetsenborden zijn bij de meeste toetsen twee getallen aangegeven. Dit zijn de zgn. ASCII-waarden van het teken, dat op de toets is aangegeven (het bovenste cijfer is de waarde met indrukken van SHIFT, het onderste cijfer is de waarde zonder indrukken van SHIFT).
Deze ASCII-waarde wordt, bij toetsindruk, in de P2000 opgeslagen en bij afdrukken op het scherm of via de printer weer omgezet in het bijbehorende symbool.

In een programma-regel kan de ASCII-waarde van een ingedrukte toets worden uitgelezen met de instructie INP("").
Voorbeeld:

10 X = INP("")

Bij INP("") wacht het programma tot een toets wordt ingedrukt en voegt de betreffende ASCII-waarde aan de variabele X toe.

5. PROGRAMMEREN IN BASIC

De BASIC-interpreter kan een groot aantal instructies en opdrachten uitvoeren. Deze zijn uitvoerig beschreven in hoofdstuk 22 (BASIC-instructies en -functies).
De P2000 kan zich in twee standen bevinden : de directe stand en de programma-stand.* In de directe stand kan een instructie worden ingetikt en de P2000 voert deze instructie direct uit. Om aan te geven dat het intikken van een instructie beëindigd is wordt de ENTER-toets ingedrukt.
Voorbeeld:

```
PRINT 2+3
 5
 Ok
```

Na het uitvoeren van de instructie komt de P2000 terug in de directe stand en is gereed voor de volgende instructie.
Wordt een instructie ingevoerd voorafgegaan door een nummer, het regelnummer, dan ziet de P2000 dit als een programmaregel. De instructie wordt dan niet direct uitgevoerd maar met het regelnummer opgeslagen in het geheugen. Na het invoeren van een aantal programmaregels kan de toets START worden ingedrukt (SHIFT-3 op het rechter toetsenbord). De P2000 zal dan alle ingevoerde instructies afwerken in volgorde van de regelnummers. I.p.v. de START-toets kan ook het woord RUN worden ingetikt, afgesloten door de ENTER-toets. De instructies hoeven dus niet in volgorde te worden ingegeven. De stand waarin de programmaregels worden uitgevoerd, noemen we de programma-stand.
Voorbeeld:

```
20 PRINT A+B
10 A=2
15 B=3
RUN
 5
 Ok
```

*) De directe stand heet in het Engels Direct mode of ook wel Command-level.

Na het uitvoeren van het programma komt de P2000 terug in de directe stand.
In het algemeen kan de uitvoering van een programma onderbroken worden door de STOP-toets (SHIFT-STOP) enkele malen in te drukken.
De P2000 komt terug in de directe stand en meldt waar het programma werd onderbroken.
Voorbeeld:

```
10 A=2
15 B=3
20 INPUT C
30 PRINT A+B+C
START-toets
?
STOP-toets
 Stop in 20
 Ok
```

Niet alle instructies zijn mogelijk in de directe stand terwijl andere instructies niet werken in de programma-stand. Dit is in de lijst van instructies (hoofdstuk 22) aangegeven.

Opmaak van een programmaregel
Een programmaregel heeft de volgende opmaak:

***regelnummer* BASIC-instructies**
Hierin is ***regelnummer*** het regelnummer, een getal van 0 t.m. 65529. Het regelnummer geeft aan in welke volgorde de instructies door de interpreter moeten worden uitgevoerd. Bovendien wordt er naar verwezen bij spronginstructies (GOTO, GOSUB, enz.) en het wordt ook gebruikt bij het wijzigen van het programma m.b.v. de EDIT-opdracht.
Het is mogelijk om op een regel meer opdrachten te plaatsen; de afzonderlijke opdrachten moeten in dat geval worden gescheiden met een dubbele punt (:). Bedenk wel dat de maximale lengte van een programmaregel 255 tekens is en dat elke programmaregel moet worden afgesloten door op de ENTER-toets te drukken.

LIST-en van een programma
Om een overzicht te krijgen van het programma zoals dat in het P2000 geheugen aanwezig is, dient de opdracht LIST.

Het intikken van het woord LIST, afgesloten met ENTER-toets, heeft tot gevolg, dat het gehele programma in volgorde van de regelnummers op het beeldscherm wordt getoond. Is het programma langer dan 24 schermregels, dan schuiven de bovenste regels van het scherm af, terwijl er aan de onderzijde nieuwe regels bijkomen.
Door het indrukken van een SHIFT-toets kan het afdrukken worden vertraagd en met SHIFT-STOP worden onderbroken.
Wordt na SHIFT-STOP een toets ingedrukt, dan wordt het LIST-en hervat. Door nogmaals SHIFT-STOP in te drukken wordt het LIST-en beëindigd.
De BASIC van de P2000 kent een zgn. „rustige LIST". Deze wordt ingeschakeld door de toets SHIFT-5 of M (SHIFT-9) op het kleine toetsenbord in te drukken.
Wordt de rustige LIST ingeschakeld met de **M**-toets dan wordt gewacht tot een LIST-commando wordt gegeven, bijv. LIST of LIST *regelnummer1-regelnummer2.*
De SHIFT-5-toets zet de rustige LIST aan en begint het programma te LIST-en bij de eerste programmaregel.
De rustige LIST onderbreekt de LIST-ing als het beeldscherm vol is. Is het eind van het LIST-en nog niet bereikt, dan verschijnt linksonder op het scherm de mededeling:

meer

Indrukken van de ENTER-toets laat de LIST één schermvol doorschuiven. De wis-regeltoets ⏮ maakt eerst het scherm schoon en toont daarna het volgende scherm vol programma. Is een printer aangesloten dan bewerkstelligt de printer-toets 🖝 het afdrukken van het ge-LIST-e scherm. SHIFT-STOP beëindigt het LIST-en, maar laat de „rustige LIST" aanstaan. Dit blijkt als later weer LIST wordt ingetikt.
Met de CODE-toets wordt de „rustige LIST" uitgeschakeld. Dit kan alleen als het woord **meer** op het scherm staat. Was het LIST-en nog niet voltooid, dan wordt de LIST-ing zonder onderbreken afgemaakt.
Elke andere toets voegt één regel aan de LIST-ing toe. De inhoud van het scherm schuift hierbij één regel naar boven en de bovenste schermregel verdwijnt.
Voor het wijzigen van programmaregels verwijzen we naar hoofdstuk 11.

6. BEELDSCHERM

Het normale beeldscherm van de P2000 beslaat 24 regels tekst van elk 40 karakters. Bij het aanzetten van de P2000 staat links op het scherm een wit blokje, de cursor. Bij het indrukken van een toets verschijnt op deze plaats de vertolking van de ingedrukte toets en schuift de cursor een plaats naar rechts. Aan het eind van een regel springt de cursor automatisch naar het begin van de volgende regel. Stond de cursor op de onderste regel, dan schuift de inhoud van het scherm automatisch een regel naar boven.
Teksten en getallen kunnen op het scherm zichtbaar worden gemaakt met het commando PRINT.

Direct
PRINT "2+5 = ";2+5
2+5 = 7

In programma
100 A=2:B=5
110 PRINT A+B
RUN (of START-toets)
7
Ok

Besturing van de cursor in de PRINT opdracht
Wil men een tekst of getal op een bepaalde plaats afdrukken bijv. op regel Y en te beginnen in kolom X, dan kan dat met de opdracht

PRINT CHR$(4)CHR$(Y)CHR$(X) ***gevolgd door de af te drukken tekst of getallen***
Alleen als X en Y „redelijke" waarden zijn wordt deze instructie uitgevoerd, anders wordt de PRINT opdracht uitgevoerd vanaf de plaats waar de cursor staat. Alle opdrachten, die met een PRINT-instructie worden uitgevoerd komen op achter elkaar te staan (als er plaats voor is. Anders wordt op de volgende regel verder gegaan).
Voorbeeld:

PRINT "Opdracht" A% "is uitgevoerd op" D;M;J
(De variabelen D M en J worden gescheiden door een ; om aan te geven dat het om verschillende variabelen gaat, anders ziet de P2000 de variabele DMJ).*

* Het begrip variabele wordt behandeld in hoofdstuk 19.

Na de PRINT opdracht gaat de cursor naar de volgende regel, behalve als de PRINT-opdracht door een ; is afgesloten.

```
10 PRINT "Opdracht nr";
20 PRINT A%;
30 PRINT "is uitgevoerd"
```

Om ruimte tussen de afgedrukte getallen te zetten kan gebruik gemaakt worden van de PRINT functies **TAB(*n*)** en **SPC(*m*)**. TAB(*n*) zet de cursor op de *n*-de kolom, SPC(*m*) geeft *m* spaties tussen de afgedrukte grootheden.

```
10 B% = 5: A% = 100
100 PRINT B% TAB(10) "Opdracht" A% SPC(5) "is klaar"
RUN
 5          Opdracht  100       is klaar
 Ok
```

Een PRINT-opdracht van meer getallen of teksten gescheiden door **komma's** zet elk volgende getal 14 kolommen verder.

```
10 A% = 100: B% = 5
100 PRINT A%, B%
RUN
 100            5
 Ok
```

Wissen

Het hele scherm wordt gewist en de cursor wordt links bovenaan gezet met de opdracht **PRINT CHR$(12)**.Om een gedeelte van het scherm te wissen bestaat een aantal instructies. Bij de meeste hiervan is van belang het zgn. „cursorpunt". Dit „cursorpunt" hoeft niet gelijk te zijn dan de plaats van de cursor op dit moment. Meestal is het cursorpunt het begin van een regel, d.w.z. de plaats waar de cursor kwam te staan na de vorige PRINT-instructie. Met PRINTCHR$(4)CHR$(Y)CHR$(X) kunnen we de cursor verplaatsen zonder het cursorpunt te wijzigen. Om het cursorpunt gelijk te maken aan de huidige plaats van de cursor geldt de instructie **PRINT CHR$(6);**.

Om een deel van het scherm te wissen zijn de volgende instructies mogelijk:*

CHR$(15) wis vanaf de plaats waar de cursor staat tot aan het cursorpunt en zet cursor op cursorpunt.

CHR$(21) wis vanaf de huidige cursorplaats tot het eind van de schermregel. Cursor verplaatst niet.

CHR$(22) wis vanaf de huidige cursorplaats tot het eind van het scherm. Cursor verplaatst niet.

CHR$(11) wis het karakter vóór de cursor en zet cursor een plaats terug.

Venster

Het is mogelijk slechts een gedeelte van het scherm te gebruiken voor de PRINT opdrachten. Dit is nuttig als men een vaste omlijsting van het beeldscherm wil aanbrengen. Het definiëren van een venster gaat als volgt in een programma:

100 PRINT CHR$(2): REM zet cursor uit
110 POKE &H60AD,*LK*: REM *LK* is linker kolom van het venster
120 POKE &H60AE,*BR*: REM *BR* is bovenste regel van het venster
130 POKE &H60AF,*H*: REM *H* is hoogte van het venster
140 POKE &H60B0,*B*: REM *B* is breedte van het venster
150 PRINT CHR$(1): REM zet cursor aan

Om de cursor binnen het venster te plaatsen kan daarna een PRINTCHR$(4)CHR$(1)CHR$(1) instructie worden gegeven.

Let op: deze tellingen lopen vanaf nul. De linker kolom van het scherm is kolom 0, evenzo de bovenste regel van het scherm. Voor een breedte van 10 kolommen moet ***B*** = 9 worden gezet.
Alle PRINT-stuurcommando's, bijv.
PRINTCHR$(4)CHR$(Y)CHR$(X)
werken nu vanaf de linker bovenhoek van het venster. Alles wat buiten het venster staat blijft onaangetast, ook door het wiscommando PRINT CHR$(12). Dit commando wist nu alleen het venster. Om het hele scherm, inclusief het gedefinieerde venster te wissen gebruiken we **PRINT CHR$(28)**,het venster wordt daarbij weer op het volle beeldscherm, 40 breed en 24 hoog, gezet en gewist.

* Deze instructies dienen in een PRINT-opdracht voor te komen.

Cursor besturingcommando's
De reeds genoemde ÇHR$(4)CHR$(X)CHR$(Y) zet de cursor op de regel X en kolom Y.
Er zijn nog meer instructies mogelijk (als deel van een PRINT-opdracht).

CHR$(13) zet de cursor voor aan de regel.

CHR$(10) zet de cursor één regel omlaag. Staat de cursor op de onderste regel van het venster, dan schuift de inhoud van het venster één regel op.

PRINT combinatie van PRINTCHR$(13); en PRINTCHR$(10)

CHR$(29) als de cursor voor aan de regel staat doet deze instructie niets, anders gelijk aan CHR$(13) + CHR$(10).

CHR$(20)CHR$(X) zet de cursor op kolom X t.o.v. de linker kantlijn van het venster.

CHR$(9) de cursor gaat vooruit en wist tot de volgende tabulator-positie. Tabulator-posities zijn de kolommen 1, 9, 17 etc.
Deze posities zijn niet te wijzigen.

CHR$(2) zet de cursor uit.

CHR$(1) zet de cursor aan.

CHR$(8) zet de cursor een plaats naar links. Stond de cursor in kolom 0 van het venster, dan springt deze naar de regel erboven, geheel rechts in het venster.

CHR$(16) zet de cursor een plaats naar links, echter niet verder dan de rand van het venster.

CHR$(17) zet de cursor een regel omhoog, niet verder dan de rand van het venster.

CHR$(18) zet de cursor een regel omlaag, niet verder dan de rand van het venster.

CHR$(19) zet de cursor een plaats naar rechts, niet verder dan de rand van het venster.

CHR$(31) zet de cursor terug op het „cursorpunt".

CHR$(6) definiëer de huidige plaats van de cursor als cursorpunt.

„Tweede" videoscherm
De P2000 beschikt over een videoscherm-geheugen, dat bestaat uit 24 regels van 80 kolommen. Het rechter deel van dit scherm is normaal niet zichtbaar. Op dit rechter deel van het videoscherm kan worden geschreven indien de breedte van het venster zó groot gemaakt wordt, dat het tweede videoscherm binnen het venster valt.
De instructie

POKE &H60B0, *getal*
laat toe op het gehele videoscherm te schrijven, mits ***getal*** een zo grote waarde heeft, dat de rechterkant van het scherm wordt overschreden, bijv. POKE &H60B0,79,
Om dit deel van het scherm zichtbaar te maken geldt de instructie

OUT 48, *getal*
Hierdoor schuift de inhoud van het videoscherm ***getal*** posities naar links en wordt getoond wat er op (een deel) van het rechter scherm is geschreven.
In de directe stand van de P2000 is altijd het linker videoscherm zichtbaar. De OUT 48, *getal* instructie werkt daarom alleen in een programma.

Snelheid van printen op het scherm
De snelheid, waarmee BASIC op het scherm schrijft wordt bepaald door het getal, dat op geheugen adres &H60A6 staat. Normaal staat hier een 0 (hoogste snelheid). Wanneer de SHIFT toets wordt ingedrukt schakelt de P2000 over op een andere snelheid, die bepaald wordt door de geheugenplaats &H60A5. Normaal staat hier 32 en geeft het indrukken van de SHIFT toets een vertraging in de schrijfsnelheid.
Door op resp. &H60A6 of &H60A5 een andere waarde te POKE-n kan schrijfsnelheid zonder en met SHIFT-toets groter of kleiner worden gemaakt.

Printen van de scherminhoud op de printer
Door indrukken van de toets ☞ (SHIFT 00) resp. in het programma door PRINT CHR$(5); wordt de inhoud van het scherm naar de printer-uitgang gevoerd.
Is geen printer aangesloten, dan volgt de foutmelding.
Geen printer
in de programma-stand gevolgd door **in *regelnummer***

POKE 25088,225 ⇒ HERSTEL 195
" 25089,201 198

F INP(&H70)AND =1 THEN OUT 0

7. KLEUR

Normaal worden de letters, tekens en cijfers op het beeldscherm afgedrukt in wit op zwarte achtergrond. (Foutmeldingen en andere „boodschappen" van de P2000 verschijnen wél in kleur). Door het afdrukken van een zgn, „kleur omschakelkarakter" verschijnt de rest van de regel op het scherm in kleur.
Deze kleuromschakelkarakters, die altijd in een PRINT-opdracht worden opgenomen, zijn:

CHR$(129) rood
CHR$(130) groen
CHR$(131) geel
CHR$(132) blauw
CHR$(133) magenta (paars)
CHR$(134) cyaan (lichtblauw)
CHR$(135) wit

Voorbeeld:
PRINT CHR$(130) "Dit is groen" CHR$(134) "en dit is licht-blauw".

Kleuromschakelkarakters kosten één positie op het scherm.
Om gekleurde letters op een gekleurde achtergrond te krijgen dient CHR$(157):
PRINT CHR$(*kleur achtergrond*) CHR$(157) CHR$(*kleur letters*) "Tekst".
Voorbeeld:

PRINT CHR$(129)CHR$(157)CHR$(131) "Gele letters op een rode achtergrond"

Terugschakelen naar zwarte achtergrond gaat met de opdracht **CHR$(156).**
Verdere mogelijkheden om tekst op een bijzondere manier op het scherm te zetten zijn:

CHR$(136) laat de rest van de regel knipperen
CHR$(137) schakelt knipperen weer uit
CHR$(141) schakelt op lettertekens van dubbele hoogte
CHR$(140) schakelt op lettertekens van normale hoogte

Opmerking:
Wordt in een programma vaak met kleuren gewerkt, dan kan het gemakkelijk zijn de kleuromschakel karakters één keer te definieren.
Voorbeelden:

```
10 RO$ = CHR$(129): GE$ = CHR$(131): RG$ = CHR$(131)
+ CHR$(157) + CHR$(129)
100 PRINT RO$ "Rode letters" RG$ "Rood op geel" CHR$ (156)
GE$ "Gele letters"
```

```
10 DEFFN RG$(T$) = CHR$(131) + CHR$(157) + CHR$(129)
+ T$ + CHR$(156)
100 A$ = "Rood op geel": PRINT FNRG$(A$)
```

8. GRAFISCHE MOGELIJKHEDEN

De P2000 kan figuren op het beeldscherm weergeven. Hiervoor staan 64 grafische karakters ten dienste, die elk opgebouwd zijn uit een rechthoekig blokje, dat in 6 kleine blokjes is verdeeld.

De kleine blokjes kunnen op zwart worden gezet (resp. de gekozen achtergrondkleur) dan wel in kleur worden weergegeven.
Voor iedere combinatie van lichte en donkere blokjes is één karakter beschikbaar (zie tabel 3).
Om op grafische tekens om te schakelen wordt een omschakelkarakter gegeven en wel:

CHR$(145) rood
CHR$(146) groen
CHR$(147) geel
CHR$(148) blauw
CHR$(149) magenta (paars)
CHR$(150) cyaan (lichtblauw)
CHR$(151) wit

Net als bij kleuromschakelkarakters is een grafisch omschakelkarakter werkzaam tot het eind van de regel op het scherm, tenzij een ander schakelkarakter wordt gegeven.
Geven we het PRINT commando
PRINT CHR$(147) "0123456789"
dan verschijnt een patroon van gele blokjes op het scherm
PRINT CHR$(150) ",,,,,,,,,,"
geeft een lichtblauwe horizontale lijn.
In tabel 3 zijn de grafische karakters opgenomen met hun ASCII-waarde en het overeenkomende alfanumerieke karakter.

Met de hoofdletters en enkele leestekens corresponderen geen grafische karakters. Hoofdletters en grafische tekens zijn dus gemakkelijk te combineren.

PRINT CHR$(147) "IS DE PRINTERTOETS"

verschijnt dus als
IS DE PRINTERTOETS

Grafische schakelkarakters nemen, net als schakelkarakters voor gekleurde letters (hoofdstuk 7), één positie op het scherm in beslag.

CHR$(135) schakelt terug naar witte letters evenals het begin van een nieuwe schermregel.

CHR$(154) CHR$ ***(grafisch)*** maakt wat kleinere grafische blokjes, die niet aaneengesloten op het scherm verschijnen (separated graphics).

CHR$(158) geeft in de rest van de regel i.p.v. een spatie op de omschakelplaats een herhaling van het laatst afgedrukte grafische karakter.
Hiermee zijn aaneengesloten kleuromschakelingen te maken.

CHR$(159) maakt CHR$(158) weer ongedaan.

→ CHR$(153 SCHAKELT (154) UIT!

9. DE CASSETTERECORDER

De cassetterecorder dient om programma's en gegevensbestanden uit het geheugen van de P2000 naar de cassette te schrijven en van de cassette in het geheugen van de P2000 te lezen. Met de ZOEK-toets is het mogelijk op het scherm de namen te tonen van de programma's en gegevensbestanden die op een cassette zijn opgeslagen. De inhoud van de cassette kan worden gewist met de toets ⊡ .
Programma's en gegevens kunnen op de cassette worden weggeschreven onder een naam van maximaal 16 karakters.
Allereerst iets over de te gebruiken cassettes. Voor de P2000 bestaan geteste minicassettes, type 52/NC/21.
Uit ervaring blijkt, dat bij deze cassettes erg weinig storingen optreden. Er bestaan ook cassettes, die bv. passen in een dicteerapparaat, waarvan de buitenmaat gelijk is aan die van de P2000-cassettes. Vaak is het mogelijk op deze, goedkopere, cassettes programma's en gegevensbestanden weg te schrijven en terug te lezen. Deze cassettes zijn echter niet bedoeld voor computergebruik (de samenstelling van de band is anders) en de kans op storingen ligt dan ook aanzienlijk hoger.
Ook bij het gebruik van de geteste P2000 cassettes kan het wel eens voorkomen dat een programma of gegevensbestand niet meer terug te lezen valt.
Oorzaken hiervan kunnen zijn:

- Men heeft de cassettehouder geopend terwijl de P2000 bezig was gegevens weg te schrijven. Hierdoor is het opnemen halverwege afgebroken en wordt het programma niet meer als correct herkend.
- De BASIC module is uit de P2000 getrokken, terwijl de P2000 ingeschakeld was en de cassette zich nog in het recordertje bevond. Soms wordt dan een klein stukje band gewist waardoor programma's of gegevens op de band verloren zijn gegaan.
- De band is per ongeluk gewist.
- Er is een ander programma met dezelfde beginletter over een bestaand programma heen geschreven.
- Er zat een fout in de band of een storing in het apparaat.

In professionele computercentra worden voortdurend kopieën (BACK UP'S) gemaakt van alle programmatuur en gegevensbestanden. Het is raadzaam van uw programma's en gegevensbestanden een kopie te maken op een ander bandje en deze kopiëen regelmatig bij te werken met de nieuwste versies.

De informatie (programma's en bestanden) wordt op de cassetteband geschreven in „blokken" van 1024 bytes. Aan iedere kant van een cassette kunnen 42 blokken worden weggeschreven. Als het laatste blok van een programma of bestand niet geheel gevuld is, wordt de rest van dit blok niet gebruikt voor een ander programma.

Wissen van de band
Voordat de inhoud van een band gewist kan worden, dient de cassette voorzien te zijn van een stopje in de linker bovenhoek en in de recorder geplaatst te worden. Zonder zo'n stopje is de cassette beschermd tegen wissen, en er kan ook niet op geschreven worden.

Door op de toets ⊡ (SHIFT 7) te drukken verschijnt op het scherm de vraag:

Cassette wissen ? (J/N)
Is geen cassette aanwezig, dan kan de wis-opdracht niet worden uitgevoerd en de P2000 meldt:

Geen cassette

Is de cassette beveiligd tegen overschrijven, doordat het stopje links boven is verwijderd, dan meldt de P2000 dit met

Geen stopje

Is wel een cassette met stopje geplaatst, en wordt de J of de Y (Ja of Yes) toets ingedrukt, dan bevestigt de P2000 de opdracht, door het woord **Ja**
achter de vraag af te drukken. De cassette spoelt terug tot het begin, wist een stukje band en de P2000 meldt:
Ok
Alle programma's en bestanden op deze kant van de band zijn nu onbereikbaar geworden. De band wordt gezien als „schoon".
Wordt de vraag:

Cassette wissen (J/N)

OM TE WISSEN VANAF BEPAALD PROGRAMMA OF BESTAND
CLOAD LAATSTE PROGR. DAT MOET BLIJVEN STAAN EN
TOETS DAN IN: DEFUSR=&H18:?USR(1.1)

met een andere toets dan J of Y beantwoord, dan ziet de P2000 dit als:

Nee

bevestigt dit „Nee" en komt terug in de directe stand.

N.B. Zolang de cassette nog teruggespoeld wordt kan het wissen worden voorkomen door de cassettehouder te openen.

Inhoud van de cassette

Door op de toets ZOEK (SHIFT-1) te drukken wordt de inhoud van de cassette op het scherm getoond en wel in de volgende opmaak:

Inhoud			
Probeersel 5 okt	**BAS B**	**6**	**5950**
Januari	**INT B**	**2**	**1036**
Kasboek	**SNG B**	**1**	**324**

Eerst de naam, waaronder het programma of het gegevensbestand is weggeschreven, daarna het soort gegevens, dat onder deze naam is opgeslagen, waarbij geldt:

BAS	BASIC programma
INT	Integer array
SNG	Array in enkel precisie
DBL	Array in dubbele precisie
FAM	Gegevensbestand voor familiegeheugen

De letter B achter de type-aanduiding betekent, dat de BASIC insteekmodule gebruikt moet worden.
Het getal hierachter vermeldt het aantal blokken (van 1024 bytes) dat het programma of het gegevensbestand op de cassette in beslag neemt en het laatste getal geeft het aantal bytes dat het bestand of programma groot is.
Het zoeken op de band mag worden gestopt door het openen van het klepje van de cassetterecorder. De P2000 meldt dan:

Geen cassette

10. LADEN EN BEWAREN VAN PROGRAMMA'S EN GEGEVENSBESTANDEN

Laden van programma's.
Een programma op de cassette kan in het geheugen van de P2000 geladen worden door het intypen van:

CLOAD ***"programma naam"***
De cassetterecorder spoelt de band terug naar het begin en zoekt het gevraagde programma op. Hierbij wordt alleen gekeken naar de eerste letter van de ingetikte programmanaam en van de programmanamen, die op de band voorkomen. "Philips" en "P2000" worden dus als gelijke namen gezien, "Philips" en "p2000" niet vanwege het verschil tussen hoofdletter en kleine letter aan het begin.
Komt het gevraagde programma niet voor op de band, dan meldt de P2000

Niet gevonden

en komt terug in de directe stand.
Is het programma wel gevonden, dan meldt de P2000
Ok
Voorbeeld:
CLOAD "sorteer"
Ok
Om de P2000 het programma vervolgens te laten uitvoeren wordt ingetikt

RUN

(of de toets, gemerkt START (SHIFT-3) op het kleine toetsenbordje). Het programma wordt dan in uitvoering genomen.

De commando's **CLOAD** en **RUN** kunnen gecombineerd worden door in te tikken:

RUN ***"programmanaam"***

Het gevraagde programma wordt opgezocht en direct in uitvoering genomen.
Het commando **RUN""** of **RUN"** zorgt voor het laden en starten van het ***eerste*** programma op de band. Ditzelfde gebeurt ook na het indrukken van de RESET toets.

Om het **eerste** programma van de band in de P2000 te laden, kan gebruik gemaakt worden van de opdracht

CLOAD

Belangrijk !

* Bij het zoeken naar programma's of gegevensbestanden wordt niet gekeken naar het soort gegevens dat op de band voorkomt. Begint de band met een gegevensbestand, bv. een integer array, dan zal het commando RUN"" het eerste bestand op de band opzoeken, dit inladen en als programma behandelen. In het algemeen zal de uitvoering van zo'n programma snel beeindigd worden met **Syntax error** (zie hoofdstuk 12 en tabel 4 over programma-fouten).

Bewaren van programma's

Een programma in het geheugen van de P2000 kan bewaard worden op de cassette door het commando

CSAVE *"programmanaam"*

waarin ***programmanaam*** uit maximaal 16 karakters mag bestaan. Als geen cassette in de recorder aanwezig is, volgt de melding

Geen cassette

De aanwezige cassette dient in de linker bovenhoek te zijn voorzien van het reeds genoemde „schrijfstopje", anders volgt de melding

Geen stopje

Door deze foutmeldingen is het programma in het geheugen niet beschadigd. Als de cassette is geplaatst en het stopje is aangebracht, kan opnieuw de opdracht CSAVE *"programmanaam"* worden gegeven.
Na intikken van **CSAVE *"programmanaam"*** spoelt de cassette terug naar het begin van de band en zoekt van daar, of er al een programma of gegevensbestand met de gevraagde beginletter op de band voorkomt. Is dit het geval, dan worden naam en type van het programma getoond en wordt toestemming tot overschrijven gevraagd.

Voorbeeld

CSAVE "probeer 1"
probeersel 5 okt BAS B 6 5950
Hier overheen? (J/N)

Wordt J of Y ingetoetst, dan bevestigt de P2000 dit met **Ja** en het oude programma **"probeersel 5 okt"** wordt vervangen door **probeer 1**.
Nu kan het voorkomen, dat het nieuwe programma, **probeer 1** zoveel langer is dan **probeersel**, dat het niet in de beschikbare 6 blokken past. Was **probeersel** het laatste programma op de band, dan wordt gewoon een stukje band meer gebruikt en het programma met beginletter **p** heeft voortaan 7 of meer blokken. Als dit niet mogelijk is, omdat er achter **probeersel** andere programma's of gegevensbestanden zijn weggeschreven, dan kan men besluiten deze informatie te overschrijven. De P2000 vraagt dan:

Mag de rest weg? (J/N)

Als deze vraag met J of Y wordt beantwoord, zijn de bestanden en programma's, die na het programma **p** op de band stonden, uitgewist.

Inlezen en wegschrijven van gegevensbestanden
Gegevensbestanden op de cassette kunnen in het geheugen van de P2000 worden gelezen door het commando **CLOAD***. Van dit commando bestaan enkele variaties:

CLOAD* A laadt het array A van de cassette in het geheugen van de P2000. Het array A moet eerst ge-DIM-d zijn, anders volgt de fout
Illegal function call.

Is onvoldoende geheugenruimte gereserveerd dan meldt de P2000,

Out of memory

Uiteraard dient de juiste cassette in de recorder geplaatst te zijn. Anders volgen meldingen als **Geen cassette** of **Niet gevonden**.

Een tweede commando om een bestand in de P2000 te laden is:

CLOAD* A@ ***"naam"***

Het bestand, dat op de cassette bekend is onder ***naam*** wordt opgezocht en in het geheugen geladen op de plaats van het array **A**. Dit commando geeft de mogelijkheid verschillende bestanden die op de band onder verschillende namen staan, na elkaar onder dezelfde naam in het geheugen op te slaan en te bewerken.

Voor het wegschrijven van gegevensbestanden naar de cassette kent de P2000 ook twee instructies.

CSAVE* A en

CSAVE* A@ ***"naam"***
Het eerste commando schrijft array A naar de band onder de naam A, met type-aanduiding SNG, het tweede commando schrijft array A naar de band onder ***naam***, met type-aanduiding SNG. Eventueel kan de naam, waaronder het array wordt weggeschreven worden aangevuld.
Voorbeeld:·

CSAVE* A @ "Kas t.m. " + "16/4/83"

CSAVE* A @ "Kas t.m. " + DA$
waar **DA$** bijv. een van te voren ingegeven datum is. Voorwaarde is ook hier dat ***naam*** uit maximaal 16 karakters bestaat. Is ***naam*** langer, dan wordt het teveel genegeerd.

Alle meldingen, zoals vermeld bij het bewaren van programma's zoals:

Geen cassette
Geen stopje
Hier overheen? (J/N)
Mag de rest weg? (J/N)

Verschijnen ook op het scherm bij het bewaren van gegevens-bestanden als daartoe aanleiding is.

Opmerkingen

1. Alle opdrachten mogen zowel in HOOFDLETTERS als in kleine letters worden ingegeven.
2. In de directe stand dienen alle opdrachten te worden afgesloten door een ENTER.
3. In de programma-stand dienen alle opdrachten in een programmaregel te zijn opgenomen.
4. Als de instructies voor het wegschrijven van programma's en bestanden in een programmaregel voorkomen, wordt aangenomen, dat de programmeur maatregelen heeft genomen tegen ongewenst overschrijven van bestanden op de band.
   In de programma-stand wordt normaal geen toestemming gevraagd oude bestanden of programma's te mogen overschrijven.
5. Eventuele foutmeldingen als "geen cassette", "geen stopje" en "niet gevonden" worden in de programma-stand gevolgd door het regelnummer, waar de fout optrad, bijv.

   Geen cassette in 25

   Na plaatsing van de cassette kan met de instructie CONT de programma-uitvoering worden voortgezet.
6. Alvorens de P2000 een programma of gegevensbestand gaat opzoeken, wordt eerst de band tot aan het begin teruggespoeld. Als men zeker weet, dat het programma of bestand verderop op de band staat, kan het terugspoelen worden onderdrukt door voor de CLOAD opdracht de instructie:

   POKE &H60AC, 1

   te geven.
   Door deze instructie wordt aan de P2000 het sein gegeven: „Gereed met terugspoelen", en er wordt alleen voorwaarts gespoeld. Het gevaar is echter aanwezig, dat een programma of bestand niet gevonden wordt of gedeeltelijk wordt geladen. Deze POKE-instructie dient voor elke CLOAD-instructie te worden herhaald. Zo niet, dan spoelt de band weer terug naar het begin.
   Het commando CSAVE spoelt de band altijd terug naar het begin.

KOPPELEN: 1) CLOAD 1e PROG
2) ? PEEK (&H6405)+256*PEEK(&H6406)-2 → GETAL X
3) POKE &H625C, X - 256 * INT(X/256) : POKE &H625D, INT(X/256)
4) CLOAD 2e PROG

11. WIJZIGEN VAN PROGRAMMAREGELS

Door het commando **EDIT *regelnummer*** schakelt de P2000 in de EDIT-stand en toont de programmaregel met *regelnummer* op het scherm. De cursor staat op het eerste cijfer van het regelnummer. De Editor van de BASIC interpreter kan in drie standen staan:

1. De uitgangs-stand
2. De wijzig-stand
3. De invoeg-stand

In **alle standen** kan worden gebruikt

De cursortoetsen. Deze verplaatsen de cursor naar rechts, links, boven, beneden, naar het begin of het eind van de programmaregel of naar het begin van de schermregel.
De correctietoets. Deze wist het karakter vóór de cursor en sluit de tekst aan.
De SHIFT-correctietoets. Deze wist het karakter onder de cursor en sluit de tekst aan.
De ENTER-toets. Hiermede wordt de EDIT-stand verlaten en de veranderde programmaregel wordt in het programma geplaatst.
De STOP-toets. Hiermede wordt de EDIT-stand verlaten en de oorspronkelijke programmaregel wordt weer in het programma geplaatst.
De wis-regel-toets. Deze doet hetzelfde als de STOP-toets.
De SHIFT → toets. Deze zet de cursor op het volgende gezochte karakter (zie bij S).

In de **uitgangs-stand** kan bovendien worden gebruikt:

C schakelt naar de wijzig-stand.
I schakelt naar de invoeg-stand.
H wist de programmaregel vanaf de cursor en schakelt naar de invoeg-stand.
X zet de cursor aan het eind van de programmaregel en schakelt naar de invoeg-stand.
S zoekt het karakter in de programmaregel, dat na het indrukken van S wordt ingegeven. (zoals boven vermeld, kan met SHIFT → het volgende gezochte karakter worden gevonden).
K als S, maar alle karakters die gepasseerd worden onderweg, worden gewist.
N beëindigt de EDIT-stand, zoals dat met de ENTER-toets zou gebeuren en zet de volgende programmaregel in de EDIT-stand.

Opmerking:
Het laatste zoek-karakter wordt bewaard, ook na het verlaten van de EDIT-stand en zelfs na RUN-nen van het programma. Met de SHIFT → toets kan, bij een volgende keer EDIT-en, naar hetzelfde karakter worden gezocht. Zolang geen zoek-karakter is ingegeven, is het zoek-karakter de dubbele punt (:).

In de **wijzig-stand** (C) wordt het karakter onder de cursor vervangen door een karakter, dat op het toetsenbord wordt ingetikt. Met de CODE-toets wordt de wijzig-stand verlaten.

In de **invoeg-stand** (I) worden op het toetsenbord ingetikte karakters tussen de tekst gevoegd op de plaats van de cursor. De rest van de tekst schuift op. De invoeg-stand wordt verlaten door het indrukken van de CODE-toets.

Het is mogelijk ook het regelnummer van de programmaregel te wijzigen. Dit heeft tot gevolg, dat deze programmaregel nu onder twee regelnummers in het programma voorkomt. De ongewijzigde programmaregel staat in het programma onder het „oude" regelnummer. De gewijzigde programmaregel staat in het programma onder het „nieuwe" regelnummer.

Met de opdracht EDIT. of door het indrukken van de DEF-toets (SHIFT-0) wordt de EDIT-stand ingeschakeld en de laatst ge-LIST-e of ge-EDIT-e programmaregel getoond.

Wees voorzichtig met het invoegen van PRINT-opdrachten als vraagteken in lange programmaregels. Bij LIST-en en EDIT-en wordt dit door de interpreter omgezet in het woord PRINT, dat 4 karakters langer is.
Het is mogelijk dat hierdoor de programmaregel, inclusief regelnummer, langer wordt dan 255 karakters, waardoor het eind verloren gaat.

12. PROGRAMMAFOUTEN

Als de P2000 tijdens de uitvoering van een programma een onduidelijkheid tegenkomt of een handeling moet uitvoeren, die verboden is (bijv. delen door 0) dan wordt de programma uitvoering onderbroken. In het algemeen schakelt de P2000 in de directe stand en toont een foutmelding op het scherm met het regelnummer waar de fout optrad.
Voorbeeld:

Syntax error in 25
Illegal function call in 1000

De waarden van alle variabelen zijn nog in het geheugen aanwezig en kunnen in de directe stand worden opgevraagd om te onderzoeken wat de oorzaak van de fout was.
Voorbeeld:

PRINT A%
FOR I% = 0 TO 50: PRINT Z(I%): NEXT

In het geval van een Syntax error („spelfout") schakelt de P2000 in de EDIT-stand en toont de regel waarin de fout voorkomt op het scherm. Als u nu de regel verbetert, of de EDIT-stand verlaat met ENTER, zijn alle variabelen gewist. Als u in het geval van een syntax error de variabelen wilt onderzoeken dan dient u de EDIT-stand te verlaten door het indrukken van de STOP-toets of de wis-regel-toets. De waarden van de variabelen blijven dan behouden. U kunt de gewenste regel daarna weer op het scherm laten afdrukken met LIST. zonder de variabelen aan te tasten. In de directe stand kunt u een array of het programma nog naar de cassette schrijven (met een van de CSAVE instructies). U kunt het programma herstarten met GOTO *regelnummer* (CONT werkt alleen als de fout hersteld kan worden zonder het programma te wijzigen, bijv. door het herstellen van een cassette fout).

Door de instructie

ON ERROR GOTO *regelnummer*

In het programma op te nemen kunt u het programma acties laten ondernemen bij optredende fouten. Als een fout in het programma optreedt wordt naar het aangegeven ***regelnummer*** gesprongen,

waar een stukje programma moet staan dat de fout onderzoekt en afhandelt. Daarna kan men terugspringen naar het hoofdprogramma met de instructie RESUME, RESUME NEXT, RESUME *regelnummer* of RESUME 0 (zie ook hoofdstuk 22), of de programma uitvoering beëindigen.

ERR en ERL
Bij een optredende fout geeft het aanroepen van de functie ERR het nummer van de fout (zie tabel 4) en de functie ERL het regelnummer waarin de fout optrad.
Voorbeeld:

```
10 ON ERROR GOTO 1000
 .
 .
100 C =A/B
110 . . .
 .
 .
1000 IF ERR = 11 AND ERL = 100 THEN PRINT "Door nul delen
kan niet": B = B + 0.1: RESUME
1010 IF ERR = 6 THEN PRINT "C wordt te groot": RESUME
NEXT
```

De foutafhandeling in regel 1000 vindt plaats als B = 0. B wordt 0.1 groter gemaakt en de deling wordt opnieuw uitgevoerd.
Als het resultaat van de deling groter zou worden dan de maximum toegestane waarde van C, wordt een waarschuwing gegeven en het programma vervolgt met de instructie volgend op de deling (regel 110).

De foutnummers met de omschrijving zijn gegeven in tabel 4.

STOP-toets
Een bijzondere "fout" mag hier nog even worden genoemd, nl. als gevolg van het indrukken van de SHIFT-STOP-toets. Hierdoor wordt foutnummer 64 gegenereerd en normaal wordt hierdoor de uitvoering van een lopend programma beëindigd. Met een foutafhandelingsroutine kan men dit voorkomen.

Voorbeeld:

```
10 ON ERROR GOTO 1000
 .
 .
1000 IF ERR = 64 THEN PRINT "Wilt U echt stoppen? J/N"
1010 A$ = CHR$(INP("")): IFA$ = "J" OR A$ = "j" THEN
END ELSE RESUME
```

„Eigen fouten"
De mogelijkheid is aanwezig zelf fouten te genereren. Dit kunnen dus fouten zijn die de P2000 tijdens de uitvoering vindt, maar ook situaties, die door de programmeur verboden zijn. Een ingegeven naam begint bij voorbeeld niet met een hoofdletter. Hiervoor heeft de programmeur de keus uit alle niet-gebruikte foutnummers uit de lijst van tabel 4. (Zie ook de instructie ERROR in hoofstuk 22).
Voorbeeld:

```
10 ON ERROR GOTO 1000
 .
 .
100 LINE INPUT "Geef naam"; A$
110 IF ASC (A$) < 65 OR ASC(A$) > 90 THEN ERROR 82

 .
 .
1000 IF ERR = 82 THEN PRINT "Naam zonder hoofdletter"
RESUME 100
```

Het foutnummer achter ERROR mag eventueel het resultaat van een berekening zijn.
Voorbeeld:

```
ERROR 30 + I%.
```

ON ERROR GOTO wordt uitgeschakeld door de instructie

```
ON ERROR GOTO 0
```

of door de instructies **CLEAR**, **NEW**, **CLOAD**, of door het wijzigen van een programmaregel.
Als in de directe stand een fout gevonden wordt, wordt als regelnummer (ERL) opgegeven 65535.

13. GELUID

De luidspreker van de aangesloten TV is te activeren via de output-poort 80.
Met de instructie:

OUT 80,1: OUT 80,0

kan in BASIC een tik uit de luidspreker worden geprogrammeerd. Als deze instructies vele malen achter elkaar worden herhaald, wordt een lage toon geproduceerd. BASIC is niet snel genoeg om bij directe aansturing van de luidspreker via output-poort 80 hogere tonen te genereren. Dit kan wel met behulp van de software toon-generator.

PRINT CHR$(7) geeft een korte „piep" toon.
De volgende opdracht kan gebruikt worden voor het genereren van een aantal fluitjes van instelbare toonhoogte en duur.

PRINT CHR$(23)CHR$(*T*)CHR$(*N1*)CHR$(*N2*).....CHR$(0)

Hierbij
Opent **CHR$(23)** de „muziekstring"
Bepaalt ***T*** de lengte van de tonen, waarbij de lengte per toon = ***T***∗4 msec en
geven ***N1***, ***N2***, etc. de toonhoogte aan volgens:
toonhoogte (= frequentie) = 34.7/***N*** kHz.
CHR$(0) sluit de „muziekstring" af.
Voorbeelden:

Een zuiver octaaf wordt verkregen door twee tonen met verhouding 2 op 1
PRINTCHR$(23)CHR$(20)CHR$(90)CHR$(45)CHR$(0)

Voor een zuivere kwint is een verhouding 3 op 2 nodig
PRINTCHR$(23)CHR$(20)CHR$(90)CHR$(60)CHR$(0)

Een zuivere terts heeft 5 op 4
PRINTCHR$(23)CHR$(20)CHR$(90)CHR$(72)CHR$(0)

Een zuivere toonladder wordt verkregen door
PRINTCHR$(23)CHR$(20)CHR$(180)CHR$(160)CHR$(144)CHR$(135)
CHR$(120)CHR$(108)CHR$(96)CHR$(90)CHR$(0)

Herkent U deze?
PRINTCHR$(23)CHR$(20)CHR$(216)CHR$(108)CHR$(54)
CHR$(27)CHR$(18)CHR$(0)

Belangrijk
Als de toonroutine niet wordt afgesloten met CHR$(0), worden alle volgende te PRINT-en tekens als toon geïnterpreteerd. Dit blijft zo totdat een CHR$(0) wordt gepasseerd of tot de P2000 in de directe stand komt. Foutmeldingen en **Ok** worden ook als tonen ge-"PRINT"

$T * 4$ MSEC

N FREQ $= 34.7/N$ kHz

14. DE PRINTER

Een printer kan op de P2000 worden aangesloten via de RS 232-C uitgang (zie hoofdstuk 1). Er is echter meer nodig dan een passende connector om een printer correct werkend aan te sluiten.

De seinsnelheid
De ASCII tekens worden bit voor bit naar de printer gestuurd. De snelheid waarmee deze bits elkaar opvolgen moet hetzelfde zijn ingesteld in de P2000 en de printer. In de P2000 is deze seinsnelheid door de programmeur in te stellen, zie hiervoor het einde van dit hoofdstuk.
De standaard seinsnelheid van de P2000 is 1200 bits/sec, of te wel 1200 Baud.
Wanneer echter in de connector van het snoer naar de printer achterop de P2000 pen 2 doorverbonden is met pen 6, dan wordt de standaard seinsnelheid 300 Baud. Het is dus verstandig een printer te gebruiken, die òf met 1200 Baud werkt òf een die met 300 Baud werkt. In een printer is de seinsnelheid meestal door middel van schakelaars of stekkertjes in te stellen.

Printer Ready
Een printer is een traag apparaat. De P2000 kan tekst veel sneller sturen dan een printer die op het papier krijgt. Daarom is er een terugmeld-verbinding om het sturen van informatie naar de printer tijdelijk te stoppen, zodat niets verloren gaat. Dit werkt via pen 20 van de P2000. Alleen zolang de printer een positieve spanning van meer dan 1,5 Volt op pen 20 zet stuurt de P2000 informatie via pen 3. Veel printers gebruiken eveneens pen 20 voor dit Printer Ready signaal. Ook komt men wel printers tegen die pen 4, pen 8 of pen 25 hiervoor in gebruik hebben. In dit geval is een aangepast snoer nodig.

Letterwiel - matrixprinters
Letterwielprinters hebben de mogelijkheid meer dan één letter op dezelfde plaats af te drukken. Van deze mogelijkheid wordt gebruik gemaakt bij het printen van accenten boven letters. Matrixprinters hebben deze mogelijkheid zelden.
De P2000 kan beide soorten printers bedienen. Het printer stuurprogramma kan worden „verteld" welk type printer in gebruik is door middel van een doorverbindingsstekkertje. Dit vindt U in een dop die in de bodem van de P2000 zit. Onder deze dop bevinden zich twee pennetjes met daarop het schuifstekkertje. Als dit verwijderd wordt is de P2000 ingesteld voor een letterwielprinter. In dit

geval worden pijlen naar links en rechts geprint in plaats van de tekens < en > door deze tekens een streepje - te zetten.
Voor het gebruik bij BASIC is het verstandig de doorverbinding altijd te laten zitten.

Onderstaande printeropdrachten zullen door de meeste printers worden begrepen en juist uitgevoerd.
BASIC stuurt karakters, tekst en getallen naar de printer m.b.v. de instructie
LPRINT
Voorbeeld:

LPRINT "Volgende getallen naar de printer" A% B%

Na het commando LPRINT wordt op een nieuwe regel begonnen, tenzij (net als bij de PRINT-routines voor het beeldscherm) de LPRINT-instructie afgesloten wordt met ;. De uitvoering van de volgende LPRINT-instructie begint dan achter de vorige.
Als de te LPRINT-en tekst langer is dan de toegestane regellengte, wordt bij het bereiken van het eind van de regel op de volgende regel overgegaan. De regellengte wordt bepaald door het getal op adres &H60AB en dit kan met een POKE-instructie worden veranderd.
POKE &H60AB,40 zet de regellengte op 40 karakters.
De positie van de printkop (het kolomnummer) kan worden opgevraagd met
PRINTLPOS(X)

Het aantal afgedrukte regels op de pagina wordt bijgehouden op geheugenplaats &H60A1, het maximaal per pagina af te drukken aantal regels staat op &H60AA.
Is de pagina „vol", dus is het maximale aantal regels afgedrukt, dan wordt automatisch een aantal lege regels afgedrukt. Dit dient om te voorkomen, dat op de perforatie van het papier wordt geprint. Het aantal lege regels tussen de pagina's wordt bepaald door het getal op geheugenplaats &H60A9. Bij het opstarten wordt het aantal regels per pagina gezet op 66 en het aantal blanco regels tussen de pagina's op 6. De totale lengte van een pagina is dus 72 regels.

Samengevat		**startwaarde**
Kolomteller	&H6253	0
Aantal kar./regel	&H60AB	80
Regelteller	&H60A1	0
Aantal reg./pag.	&H60AA	66
Aantal lege regels	&H60A9	6

Deze waarden zijn te wijzigen, bij voorbeeld met POKE-instructies.

Stuurkarakters
Stuurkarakters kunnen invloed hebben op de kolomteller, maar worden niet altijd afgedrukt.

Enkele belangrijke stuurkarakters in een LPRINT-opdracht zijn

CHR$(8)
Dit „terug"-commando wordt door de meeste printers niet uitgevoerd. Als de printer het wel kan, wordt de printkop een kolom terug gezet, maar de kolomteller wordt verhoogd. Daardoor kan het voorkomen, dat te vroeg gesignaleerd wordt dat de regel vol is. Dit is te verhelpen door zelf bij CHR$(8) de kolomteller van te voren 2 posities terug te zetten. Bijv. met
POKE&H6253, PEEK(&H6253)-2 (mits de waarde van de kolomteller niet 0 is).

CHR$(9)
Bij dit „tabulatie"-commando wordt door de P2000 een aantal spaties naar de printer gestuurd (minimaal 1, maximaal 8) om het volgende karakter op de volgende tabulatiestop af te drukken. De tabulatiestops zijn de kolommen 1, 9, 17 etc.

CHR$(10)
Het papier wordt 1 regel opgeschoven. Hierbij wordt ook de regelteller (&H60A1) één verhoogd. Wordt het maximale aantal regels per pagina overschreden, dan wordt overgegaan op een nieuwe pagina.

CHR$(12)
Er worden zoveel lege regels naar de printer gestuurd, dat bovenaan een nieuwe pagina wordt begonnen. Het aantal lege regels wordt bepaald door de stand van de regelteller.

CHR$(13)
Zet de printkop terug op het begin van de huidige regel en zet de kolomteller op 1.

Opmerking: voor het afdrukken van het beeldscherm op de printer wordt gebruik gemaakt van de opdracht PRINT CHR$(5) of de SHIFT-00-toets.

Als veel beeldscherminhouden op de printer afgedrukt moeten worden is het praktisch om de pagina-lengte (&H60AA) op 24 te zetten en het aantal af te drukken lege regels (&H60A9) op 12. Zo passen twee volle schermen op één vel papier.

Instellen van de seinsnelheid
De normale seinsnelheid is 1200 bit/sec. Deze snelheid is te wijzigen doorop de geheugenplaats &H6016 een getal te plaatsen met een POKE instructie. De seinsnelheid PS wordt gegeven door de volgende formule:
PS = 2400/(PEEK(&H6016) + 1)

POKE-n we op &H6016 de volgende getallen, dan zal de seinsnelheid zijn:

0	2400 bit/sec
1	1200 bit/sec
7	300 bit/sec
21	110 bit/sec

Controle op de printer
Als geen printer is aangesloten en de P2000 krijgt opdracht om tekst of getallen naar de printer te sturen, wordt de programma uitvoering onderbroken en verschijnt de foutmelding.

Geen printer in *regelnummer*

Deze fout kan worden opgevangen door een ON ERROR GOTO regelnummer in het programma op te nemen. Het nummer van deze fout is 80 (zie tabel 4).

15. GEHEUGEN INDELING EN OPSLAG VAN PROGRAMMA EN VARIABELEN

Het geheugen van de P2000T is als volgt ingedeeld:

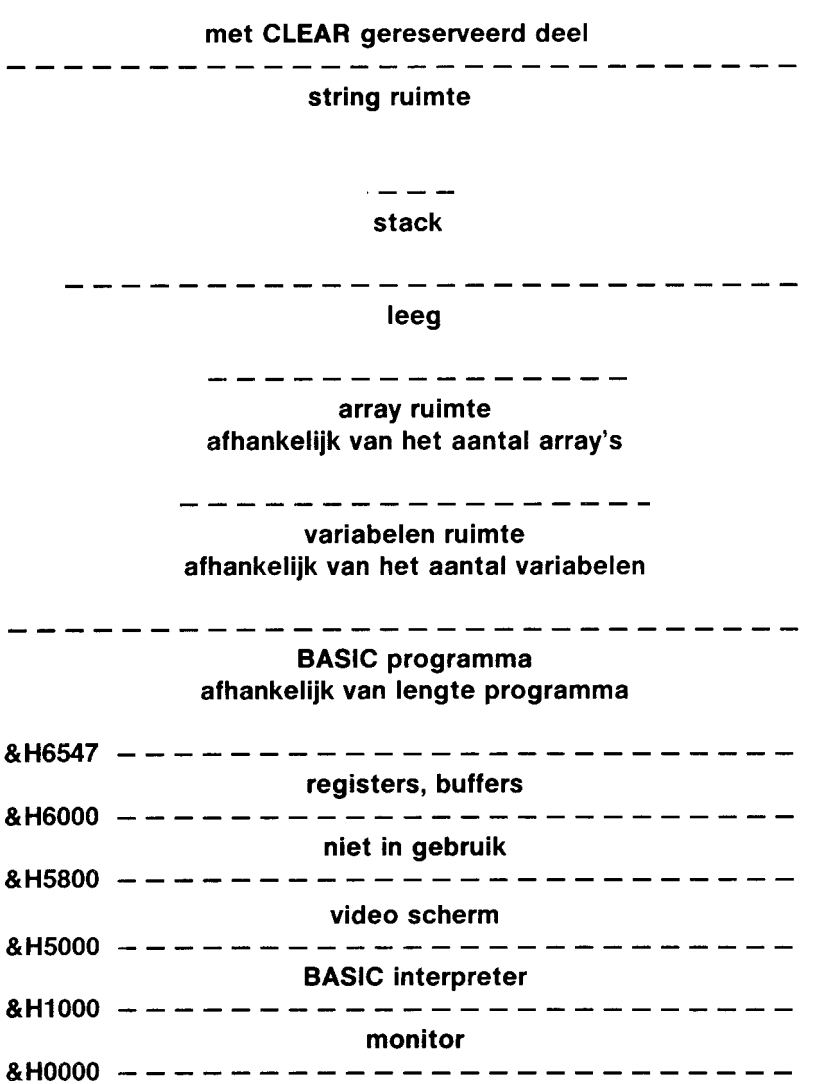
&H0000-&H0FFF Monitor in ROM
&H1000-&H4FFF Plug-in ROM module in sleuf 1
&H5000-&H57FF Video geheugen
&H5800-&H5FFF Gereserveerd voor Videoschakel bits
(niet aanwezig op T model)
&H6000-&H9FFF 16 K RAM standaard
&HA000-&HDFFF Geheugen uitbreiding tot 32 K
&HE000-&HFFFF Geheugen uitbreiding tot 48 K

Gebruik van het RAM-geheugen
Het gedeelte van &H6000 tot &H61FF wordt door de monitor gebruikt als registerruimte en voor buffer blokken. Vanaf &H6200 gebruikt de Philips Cassette BASIC een gedeelte voor zijn eigen administratie en als input buffer.
Het BASIC programma begint normaal op &H6547. Vanaf dit adres staat een ketting van programma regels in het geheugen.
Elke regel is als volgt opgebouwd:
2 bytes - RAM adres van het begin van de volgende regel
(de waarde hiervan is 1e byte + $256*2^e$ byte)
2 bytes - regelnummer (1^e byte + $256*2^e$ byte)
Hierna volgt de BASIC-programmaregel. Alle BASIC-instructies zijn gecodeerd in een enkele byte (zie hiervoor tabel 6 - BASIC tokens). De rest (getallen, teksten) is in ASCII-waarden opgeslagen. Het einde van de programmaregel wordt aangegeven door een byte met waarde 0. Daarna komt de volgende regel.
Na de laatste programmaregel komen nog 2 bytes ter waarde 0. Het programma eindigt dus met 3 bytes gelijk aan 0. Alle BASIC acties zoals LIST en RUN eindigen als dit vervolgadres 0 wordt bereikt.

Onmiddellijk na het BASIC programma begint de variabelen-ruimte. Alle variabelen die bij het RUN-nen van het programma nodig zijn, worden in volgorde van opkomst achter het BASIC-programma geplaatst. Na de laatste gewone variabele volgt de array-ruimte. Op het moment dat een array wordt ge-DIM-d wordt in de array-ruimte, een stuk geheugen gereserveerd voor de elementen van dit array. Dit heeft tot gevolg dat een ge-DIM-d array een stukje moet opschuiven als er na de DIM-instructie nog een nieuwe variabele moet worden aangemaakt (zie ook hoofdstuk 16). Na de array-ruimte volgt een leeg stuk geheugen.

Schematisch samengevat

top geheugen



Kijken we vanaf de bovenkant van het geheugen, dus vanaf het hoogste geheugen-adres, dan vinden we eerst een gereserveerd stuk dat niet door BASIC kan worden gebruikt. De grootte van dit stuk kan door de programmeur worden gekozen m.b.v. een CLEAR-opdracht.
Daaronder bevindt zich de string-ruimte. Hier worden alle nieuwe strings geplaatst, zoals strings, die met INPUT of LINEINPUT zijn verkregen en strings die door string operaties zijn ontstaan. Ook de grootte van de string-ruimte kan met CLEAR worden ingesteld.
Onder de string-ruimte wordt een stukje geheugen gebruikt voor de zgn. stapel (stack). Hier bewaart de BASIC interpreter een aantal gegevens die tijdens de uitvoering van het programma belangrjk zijn, zoals terug-spring adressen van GOSUB en FOR...NEXT routines.

Variabelen
Variabelen worden in de variabelen-ruimte in bijna leesbare vorm bewaard. De algemene opmaak is:

1 byte	type variabele, hier vindt men voor een integer variabele = 2 string variabele = 3 enkele-precisie = 4 dubbele-precisie = 8
2 bytes	naam van de variabele in ASCII
2-8 bytes	de waarde

Integer variabelen
De waarde van integer variabelen wordt opgeslagen in 2 bytes te beginnen met het laagste byte. Als voorbeeld, de variabele AZ%, die de waarde 721 heeft, staat in het geheugen als:

type	naam		waarde	
2	65	90	209	2

Voor negatieve getallen geldt dat het hoogste byte minstens 128 is Dit betekent dat -3000 in feite geschreven wordt als 62536.

Enkele en dubbele-precisie variabelen
De waarde van de variabele AZ worden als volgt opgeslagen

type	naam		waarde			
4	65	90	m3	m2	m1	ex
			drie bytes mantissa			1 byte exponent

De drie bytes van de mantissa vormen een getal M, wat als volgt wordt samengesteld:
Het hoogste bit van m1 is een tekenbit. Is dit bit = 1 dan is M negatief. De absolute waarde van M wordt nu bepaald door de reeks:

$$M = 0.5 + b_1*2^{-2} + b_2*2^{-3} + b_3*2^{-4} + \quad b_{23}*2^{-23}$$

waarin b_1 t/m b_7 de resterende bits zijn van m1 en b8 t/m b23 de bits van m2 en m3. De uiteindelijke waarde van het getal wordt dan

$$M * 2^{\text{exponent byte } -128}$$

Voorbeeld:

m3	m2	m1	ex	
00000000	00000000	00000000	10000010 =	$0.5*2^2 = 2$
00000000	00000000	01010000	10000010 =	$(0.5 + 2^{-2} + 2^{-4})$ $*2^1 = 1.625$

De precisie van een enkele-precisie waarde wordt dus bepaald door 23 bits en is dus $2^{-23} = 10^{-7}$. Er zijn dus 7 cijfers significant. Dubbele-precisie waarden worden in feite net zo opgeslagen als enkele-precisie waarden, echter omdat voor dubbele-precisie waarden 8 bytes beschikbaar zijn, wordt de mantissa uit 55 bits gevormd. (De exponent wordt door eéń byte weergegeven.) De laatste toevoeging aan een dubbel-precisie getal is dus

$$2^{-55} = 3.10^{-17}$$

hetgeen betekent dat 16 cijfers significant zijn.

String variabelen

De string variabelen AZ$ worden als volgt opgeslagen:

type	naam			
3	65	90		a1 a2
			lengte van de string	adres van het eerste karakter van de string

Na het type en de naam volgen 3 bytes die samen de zgn. string descriptor vormen. Het eerste daarvan bevat de lengte van de string, het tweede en derde byte vormen het adres van het eerste karakter van de string waarbij dit *adres* = a2 * 256 + a1. Staat de

string in het programma, bijv. als:

10 ST$ = "Dit is een string"

dan „wijst" de string descriptor naar het adres in programmaregel 10, waar de **D** van **Dit** staat; de lengte van de string staat in het eerste byte van de string descriptor als 17.
Wordt een string AZ$ naar de string-ruimte overgebracht (bijv. door de instructie AZ$ = "ge" + "weest"), dan zal de stringlengte als 7 genoteerd staan en de laatste twee bytes van de string descriptor zullen een adres in de string-ruimte bevatten. De strings worden in volgorde van binnenkomst in de string-ruimte geplaatst met het laatste karakter van de string op het hoogste adres. Is het hoogste adres van de P2000 &H9FFF (16 K uitvoering), dan staat bovenstaande string AZ$ als volgt in de string-ruimte:

&H9FF9	9FFA	9FFB	9FFC	9FFD	9FFE	9FFF
g	e	w	e	e	s	t

Wordt aan AZ$ een andere tekst toegekend, dan blijft de oude tekst in de string-ruimte staan. Er wordt echter niet meer naar verwezen. Raakt de string-ruimte vol dan gaat de P2000 oude strings overschrijven. Hiertoe wordt de string-ruimte gereorganiseerd. Alle nog in gebruik zijnde strings worden aangeschoven, zodat de nog vrije ruimte aaneensluitend in de string-ruimte staat. Dit kan in het programma worden geactiveerd met:

A = FRE("")
A geeft dan de, na „opruiming" overgebleven, stringruimte.

Arrays
De opslag van arrays in de array-ruimte doet sterk denken aan de opslag van variabelen, behalve dat aan de array-elementen enkele gegevens over het array vooraf gaan. In het algemeen is de opmaak van een array als volgt

1e byte	soort array, waarbij weer voor een integer array = 2 string array = 3 enkele-precisie = 4 dubbele-precisie = 8

2e/3e byte	naam van het array in ASCII
4e/5e byte	het aantal bytes dat het array lang is vanaf hier (dat is de lengte van het datablok + 2* het aantal dimensies + 1)
6e byte	aantal dimensies
7e/8e byte	lengte van de laatste dimensie
9e/10e byte	lengte van de voorlaatste dimensie etc.

1ste byte van het 0e element van het array.

Voorbeelden

a. Integer array van 1 dimensie; ZA% (100)
b. Enkel-precisie array, 1 dimensie; ZA (100)
c. Integer array 2 dimensies; ZA% (25,100)

byte nr.	a	b	c	type
1	2	4	2	
2	90	90	90	naam ZA
3	65	65	65	
4	205	151	137	lengte
5	0	1	20	
6	1	1	2	aantal dimensies
7	101	101	26	aantal elementen per dimensie
8	0	0	0	
9	ZA%(0)	ZA(0)	101	
10			0	
11	ZA%(1)		ZA%(0,0)	
12				

16. VERSNELLEN EN VERKORTEN VAN PROGRAMMA'S

Hieronder volgt een aantal suggesties om de uitvoering van programma's te versnellen.

1. Als een programma vaak d.m.v. een **GOTO *regelnummer*** of **GOSUB *regelnummer*** naar ***regelnummer*** springt, heeft het zin deze regels in het begin van het programma te plaatsen.
   Bij het uitvoeren van GOTO of GOSUB naar een lager regelnummer begint BASIC bij de eerste programmaregel te zoeken. Is dit niet de gevraagde programmaregel, dan springt BASIC naar de volgende programmaregel, enzovoort, tot de programmaregel ***regelnummer*** gevonden is. Wordt naar een hoger regelnummer gesprongen, dan begint het zoeken naar ***regelnummer*** op de volgende regel.
   Om deze reden is het ook wenselijk DATA regels en REM regels achteraan in het programma te plaatsen.
2. Gebruik variabelen, die vaak voorkomen als eerste in het programma. Bij uitvoering van de instructie **A% = 1** begint BASIC aan het begin van de variabelen-ruimte te zoeken naar de variabele A%. Als A% als eerste een waarde heeft gekregen duurt het zoeken kort.
3. Gebruik indien mogelijk integer variabelen. Integer-rekenen gaat sneller.
4. Gebruik korte expressies in **IF *expressie* THEN** opdrachten, die vaak voorkomen omdat de hele ***expressie*** getest wordt.

   50 IF A=1 OR A=2 OR A=3 OR A=5 OR A=7 THEN 100
   kost beduidend meer tijd dan:
   50 IF A=1 THEN 100
   52 IF A=2 THEN 100
   54 IF A=3 THEN 100
   56 IF A=5 THEN 100
   58 IF A=7 THEN 100

5. Gebruik integer variabelen als teller in een FOR-NEXT lus en laat de variabele-naam achter NEXT weg, als dit kan
   FOR A =1 TO 1000: NEXT A duurt bijna 3x zo lang als
   FOR A%=1 TO 1000: NEXT
6. Dimensioneer arrays zo laat mogelijk. Als een array ge-DIM-d is wordt de ruimte voor dit array gereserveerd direct achter de variabelen-ruimte. Komt er dan later nog een variabele bij, dan moet het hele array wat opgeschoven worden. Speciaal bij veel en grote array's kan dit lang duren.

7. Gebruik variabelen in plaats van getallen. Een getal in een programmaregel wordt geconverteerd naar een aantal bytes alvorens de operatie kan worden uitgevoerd, bij voorbeeld:

   10 OM = 2x3.1415926536*ST
   20 OP = 3.145926536 * ST * ST
   kost meer tijd dan:

   10 PI# = 3.1415592653
   20 OM = 2 * PI# * ST
   30 OP = PI# * ST * ST

Als een programma te lang is geworden en, met de nodige variabelen en stringruimte, niet meer in het geheugen past, kunnen de volgende tips dienen om het programma te verkleinen:

1. Verwijder alle REM's. Denk erom dat hierdoor geen programmaregels waar naartoe gesprongen wordt (met GOTO of GOSUB) mogen verdwijnen.
2. Verwijder variabele-aanduidingen zoals %, !, #, $. Definieer typen met DEFINT, DEFDBL, DEFSTR. Gebruik integer variabelen waar mogelijk. Deze nemen slechts 5 bytes in beslag in de variabelen-ruimte.
3. Verwijder alle onnodige spaties, punt-komma's, string aanhalingstekens, GOTO's, dubbele punten enz.
   20 IF C < > 0 THEN GOTO 100 : ELSE PRINT A(I);B(J); "Tekst"
   mag ook geschreven worden als:

   20 IFCTHEN100ELSEPRINTA(I)B(J)"Tekst"
   waarbij de 2e regel 16 bytes korter is!

4. Gebruik subroutines of FN voor bewerkingen, die zeer vaak voorkomen. Moet er bijv. vaak op het scherm geschreven worden, definieer dan eerst een cursor routine:

   10 DEFFN P$(X,Y) = CHR$(4) + CHR$(Y) + CHR$(X)
   100 PRINT FNP$(10,12)

   Dit kost minder ruimte dan vaak

   PRINT CHR$(4)CHR$(12)CHR$(10)

5. Plaats meerdere opdrachten op een programmaregel, gescheiden door een:

   10 A=1:B=2:C=3
   is 8 bytes korter dan:
   10 A=1
   20 B=2
   30 C=3

   Bovendien gaat de uitvoering van de opdrachten sneller.
6. Gebruik de 0-elementen van arrays.
   Ook als U korte arrays nodig hebt (kleiner dan 11 elementen), DIM deze arrays dan. Bij aanroep van een onge-DIM-d array wordt automatisch de ruimte voor 11 elementen gereserveerd. Voor een onge-DIM-d 3-dimensionaal array wordt zelfs ruimte voor 1331 elementen gereserveerd. (De DIM-instructie van een extra array kost minimaal 5 bytes).
7. Kies lage nummers voor programmaregels, waar U vaak, vanuit verschillende plaatsen uit het programma naar toe springt.
   GOTO 25000 is 4 bytes langer dan **GOTO 5**
8. In programma's met veel tekst kan een veel voorkomend woord als string worden gedefinieerd.

   10 P$="programma"
   100 PRINT"De"P$"'s voor de "P$"beoordeling".

9. Gebruik zo weinig mogelijk variabelen. Stel in het algemeen „klad-variabelen" in. Gebruik ook de variabelen die bij DEFFN zijn aangemaakt.

17. ENKELE OPMERKINGEN OVER HET PROGRAMMEREN IN MACHINETAAL

Bij het programmeren in machinetaal worden de instructies niet, zoals in BASIC, door een leesbaar codewoord ingegeven, dat door de interpreter in een aantal instructies wordt omgezet. Machinetaal programma's bestaan uit machinecodes, die de microprocessor direct aansturen. Een voordeel van machinetaal programma's is, dat ze vele malen sneller zijn dan programma's in BASIC. Een moeilijkheid is dat alle instructies één voor één gegeven moeten worden, terwijl in BASIC een aantal standaard-routines aanwezig is. Het maken van een machinetaal programma kost veel meer tijd en moeite dan een overeenkomstig programma in BASIC. In het algemeen zal men dus die delen van een programma waar de tijd geen rol speelt, in BASIC maken en stukken die snel moeten werken in machinetaal (bijv. sorteerroutines voor grote bestanden). Gelukkig is het op een eenvoudige wijze mogelijk machinetaal programma's te koppelen aan een BASIC-programma.

Koppelen machinetaal aan BASIC
Alvorens een machinetaal programma in het geheugen te plaatsen, is het verstandig een stuk geheugen hiervoor te reserveren zodat het machinetaal programma niet door BASIC overschreven kan worden. Meestal kiest men hiervoor de bovenkant van het geheugen. Dit kan voor BASIC ontoegankelijk worden gemaakt d.m.v. een CLEAR opdracht, bijv.
CLEAR 50, &H9000

Deze opdracht zet de stringruimte op 50 bytes grootte, en verschuift de bovengrens van het geheugen naar &H9000, zodat het bovenste deel (vanaf &H9001) bij de normale programmauitvoering niet wordt gebruikt. Vervolgens kunnen in dit deel machinetaalcodes ~~té~~ worden geplaatst. Deze codes worden door de fabrikant van de Z80 microprocessor opgegeven. In het BASIC-programma kunnen we deze hexadecimale codes in DATA regels opnemen, bijv.:

1000 DATA *code, code, code, code, code,* STOP
Met behulp van READ en POKE opdrachten worden de codes nu naar de beschermde ruimte overgebracht.

```
100 RESTORE 1000:P = &H9001
110 READ A$:IFA$ = "STOP"THEN 150 ELSE POKE P,
VAL("&H" + A$):P = P + 1:GOTO 110
150 REM Voortzetting programma
```

Nu moeten we het machinetaal programma nog een nummer geven; hiervoor staan de nummers 0 t/m 9 ter beschikking. We dienen nog te vermelden op welk geheugen adres dit programma begint. Kiezen we voor het nummer bijv. 5 en begint, zoals in het bovengenoemde voorbeeld het programma op &H9001, dan gaat de definitie van het machinetaal programma met
150 DEF USR5 = &H9001

Om van het BASIC-programma naar het machinetaal programma te springen bestaan de instructies

X = USR5(Y) en **PRINT USR5(Y)**
Bij het passeren van deze instructie verlaat de interpreter het BASIC-programma, springt naar geheugenplaats &H9001, voert het daar staande machinetaal programma uit en springt terug naar de volgende opdracht in het BASIC programma. Een fout in het machinetaal programma heeft meestal tot gevolg dat de P2000 „vast blijft zitten" en alleen nog via de RESET-toets in BASIC terug kan komen. Het is daarom verstandig, alvorens bij het testen een machinetaal programma aan te roepen, eerst het geheel te CSAVE-en.

Nog iets over de USR-functie
De USR-functie ziet er als volgt uit: **USR *cijfer (argument)*** waarbij ***cijfer*** een cijfer is tussen 0 en 9. Het argument is een numerieke of stringexpressie. ***cijfer*** geeft aan welke machinetaalroutine wordt aangeroepen en correspondeert met het cijfer dat in de DEF USR instructie voor die routine is gebruikt. Wordt ***cijfer*** weggelaten dan vult de interpreter zelf de waarde 0 in. Het adres dat in de DEF USR instructie is opgegeven, is het startadres van de machinetaal routine.

Wanneer de USR-functie begint aan het stukje machinetaal bevat de in de microprocessor van de P2000 aanwezige accumulator (register A) een waarde die correspondeert met het type argument dat was opgegeven. In register A kunnen de volgende waarden staan:

Waarde in A	Type argument
2	2-byte integer (2-complement)
3	string
4	enkel-precisie
8	dubbel-precisie

Indien ***argument*** een getal is (integer of floating point) dan geldt dat de inhoud van registerpaar HL een adres bevat van maximaal acht geheugenlocaties die samen de zgn. Floating Point Accumulator (FAC) vormen. In die Floating Point Accumulator is dan het argument opgeslagen.
Er zijn nu drie mogelijkheden:

- ***argument*** is een integer. De waarde van de integer is opgeslagen in &H650D en &H650E. In HL staat &H650D.

- ***argument*** is een enkel-precisie getal. De vier bytes van ***argument*** zijn opgeslagen van &H650D - &H6510. In HL staat &H650D.

- ***argument*** is een dubbel-precisie getal. De acht bytes van ***argument*** zijn te vinden in &H6509 - &H6510. In HL staat &H6509.

Is ***argument*** een string, dan bevat HL het getal &H650D. Op &H650D en &H650E staat nu het adres van de stringdescriptor van ***argument***. Dit adres zou men ook vinden met de BASIC-instructie VARPTR(A$) (zie hoofdstuk 22).

Het resultaat van de uitvoering van de machinetaal routine kan als volgt naar BASIC worden doorgegeven.

- Zet (in de machinetaal routine) het type van de constante in de type-indicator &H63B6. Zet de waarde in de Floating Point Accumulator, zoals hier boven is aangegeven.
- Beëindig de machinetaal routine.
  Het resultaat wordt nu toegevoegd aan **X**, resp. op het beeldscherm afgedrukt, afhankelijk van de aanroep van de USR-functie.

H605C H2000 H5FFF

1) A=PEEK(24668): CLEAR 50,(2*A+(A=3))*8192+24575

2) CLEAR 50,40959+8192*PEEK(24668)

3) CLEAR 50,&H5FFF+&H4000*PEEK(&H605C)+&H2000*(3=PEEK(&H605C))

4) CLEAR 50,(2*PEEK(H605C)+(PEEK(H605C)=3))*H2000+H....

→ A1=A-...., TO A=READ..: POKE A1,VAL("&H"+..): NEXT: DEFUSR=A+

18. DE P2000 EN DE BUITENWERELD

De Z80 microprocessor werkt met ingangs- en uitgangspoorten. Dit is een 256-tal adressen bedoeld voor de verbinding met de buitenwereld. Zoals geheugen-adressen toegankelijk zijn vanuit BASIC met **K = PEEK(*n*)** en **POKE *n, K*** zijn de ingangs- en uitgangspoorten toegankelijk met **K = INP(*m*)** en **OUT *m, K***. Beide getallen ***m*** en ***K*** liggen tussen 0 en 255 (&H00 en &HFF).

In gebruik zijnde poorten
De volgende input-output poorten worden door de 16K P2000T intern gebruikt:

&H00-&H09 input poorten van toetsenbord
&H10-&H1F output poort naar cassette en printer
&H20-&H2F input poort van cassette en printer
&H30-&H3F output poort horizontale verschuiving videoscherm
&H50-&H5F output poort van luidspreker
&H70-&H7F output poort voor DISC stuursignaal

Behalve bij de 10 poorten voor het toetsenbord is het zo dat de 16 poortadressen per blok hetzelfde doen. Dus &H14 doet hetzelfde als &H10 en &H1F.
Er is een uitbreidingskaart voor de P2000 met daarop meer geheugen en een aansluitmogelijkheid voor floppy disc. Op deze kaart zijn de volgende input-output poorten in gebruik:

&H80-&H87 input en output communicatie interface
&H88-&H8B output voor interrupt controller-timer
&H8C-&H93 floppy disc controller
&H94-&H97 geheugenbank schakelaar (E000-FFFF)
OUT &H94, 0 geeft bank 0
OUT &H94, 1 geeft bank 1

Vrije poorten
Voor een 16 K P2000 zijn vrij voor input en output de poorten:
&H40-&H4F
&H60-&H6F
&H80-&HFF
Voor een P2000 met de uitbreidingskaart geldt dat slechts vrij zijn voor input en output de poorten:
&H40-&H4F
&H60-&H6F

Alleen output-poort
De in de P2000 ingebouwde output poorten worden ook aangesproken wanneer er input gevraagd wordt van hetzelfde poortadres. Men moet dus geen eigengemaakte input-interface maken voor een poort die de P2000 zelf gebruikt voor output. Omgekeerd is het zo dat de ingebouwde input poorten alleen op een input-instructie reageren. Men kan dus een eigen interface die alleen voor output bedoeld is zo maken dat hij werkt op door de P2000 intern in gebruik zijnde input adressen.
Voor alleen output komt er dan extra beschikbaar:
&H00-&H0F
&H20-&H2F
en voor de uitgebreide versie:
&H98-&HFF

Input poorten &H00-&H09
Op de input poorten &H00-&H09 zit het toetsenbord. Alle toetsen hebben een bit in één van de bytes die op de poorten 0-9 te vinden zijn. Als er geen toets is ingedrukt zijn alle bits hoog, het getal dat met A=INP(0)...A=INP(9) binnen komt is gelijk aan 255 (=&HFF). Het indrukken van een toets maakt op één van de poorten één bit nul. Voor U dit gaat proberen moet U de normale keyboard input routine van BASIC uitzetten met OUT &H10,0 en dan met INP(0)...INP(9) naar de toetsen te kijken. Tik hiervoor dit programma in:

```
10 PRINT CHR$(12)
20 OUT&H10,&H00
30 GOSUB 100
40 OUT&H10,&H40
50 PRINT
60 GOSUB 100
70 FOR I=0 TO 1000:NEXT
80 GOTO 10
100 FOR I= 0 TO 9
110 PRINT "POORTNR" I;HEX$(INP(I))
120 NEXT
130 RETURN
RUN
```

en probeer eens wat toetsen.

In regel 20 wordt het toetsenbord uitgezet zodat alle toetsen afzonderlijk zijn te zien. In regel 40 wordt het toetsenbord weer aangezet. Alle input-poorten 0-9 staan nu parallel. Dit is te gebruiken om te zien of er een toets is ingedrukt.

Output poort &H10-&H1F. Output naar cassette en printer
Bit 7 = printer data out. Printer connector pen 3.
Bit 6 = keyboard enable. Een 0 blokkeert het toetsenbord.
Bit 3 = forward cassette. Transport vooruit.
Bit 2 = reverse cassette. Transport achteruit.
Bit 1 = write command cassette. Opnemen.
Bit 0 = data naar cassette.

Let op!
De interrupt afhandelings-routine zet iedere 20 msec poort &H10 op &H40. Pogingen om de cassette vanuit BASIC te besturen met de interrupt in werking worden hierdoor gedwarsboomd.

Input poort &H20-&H2F. Input van cassette en printer
Bit 7 = read data van cassette.
Bit 6 = cassette read clock. Kloksignaal van de band.
Bit 5 = begin-einde van het band signaal.
Bit 4 = cassette in het recordertje.
Bit 3 = write enable (het zwarte stopje).
Bit 2 = stekkertje dat het soort printer aangeeft, zie hoofdstuk 14.
Bit 1 = printer ready. Printer connector pen 20.
Bit 0 = data van printer. Printer connector pen 2.

Output poort &H30-&H3F. Output van het videoscherm
Bit 7 = blanking van het scherm. Een 1 hier maakt het scherm zwart, maar wist niet de informatie.
Bit 6-0 = horizontale verplaatsing op het scherm.

Output poort &H50-&H5F.
Output naar de luidspreker van TV of de luidspreker aangesloten op het middenpennetje van de RGB monitor plug.
Bit 0 = luidspreker.
Rest niet in gebruik.

De printer connector
Het gebruik van de printer connector is de eenvoudigste manier om een verbinding met de P2000 tot stand te brengen. Deze is wel is waar bedoeld voor seriële data overdracht, maar in wezen zijn er twee ingangen en een uitgang, die vrij gebruikt kunnen worden.

De gebruikte electrische spanningen zijn:
+ 10 Volt voor een logische 0
– 10 Volt voor een logische 1

De maximaal te leveren stroom per uitgang is 10 mA, de ingangsweerstand is meer dan 10 kOhm. Er wordt een 0 doorgegeven als de ingangsspanning hoger is dan + 1,5 Volt.

De 25 polige connector heeft de volgende aansluitingen:

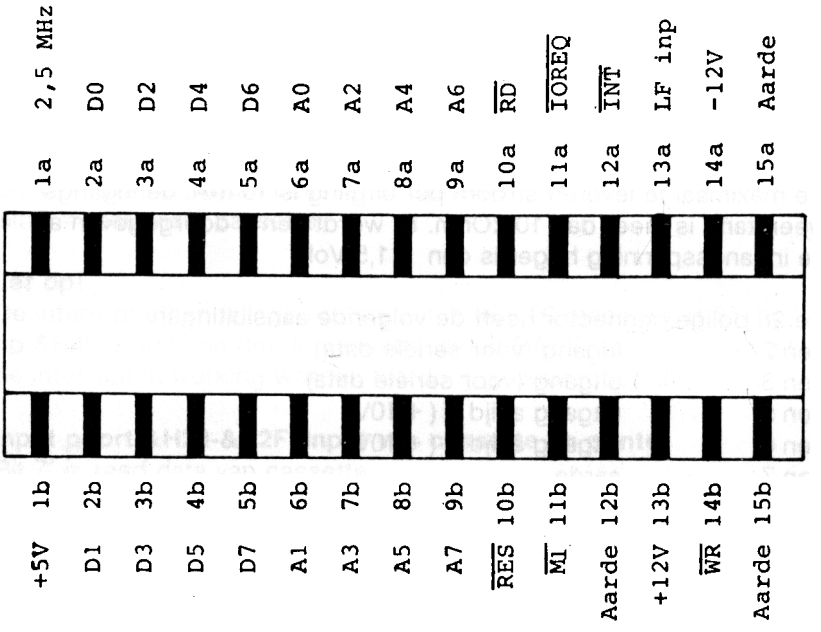
pen 2	ingang (voor seriële data)
pen 3	uitgang (voor seriële data)
pen 5	uitgang altijd 0 (+ 10V)
pen 6	uitgang altijd 0 (+ 10V)
pen 7	aarde
pen 20	ingang (voor Printer Ready)

De twee uitgangen 5 en 6 mogen gebruikt worden als voeding voor zuinige schakelingen.
De twee uitgangen komen in het programma uit op input-poort &H20, de enige te besturen uitgang (pen 3) is een bit in output-poort&H10.

Voor gebruik als seriële aansluiting voor de printer is in de monitor-ROM een stukje machinetaal aanwezig om informatie bytes te versturen zolang ingangspen 20 een positieve spanning krijgt toegevoerd. Er is geen standaard routine aanwezig om seriële data op pen 2 te ontrafelen.

De connector van sleuf 2

De aansluitingen van de connector voor de input/output module zijn als volgt



De connector is getekend zoals U erop kijkt als U voor het toetsenbord van de P2000 zit.

Betekenis van de aanduidingen

D0 - D7	=	Data-bus (niet gebufferd)
A0 - A7	=	Adres-bus (niet gebufferd)
$\overline{\text{RES}}$	=	RESET, laag als op de resetknop gedrukt wordt
$\overline{\text{IOREQ}}$	=	I/O request, laag voor input of output
$\overline{\text{RD}}$	=	Read, laag als er gelezen wordt (input)
$\overline{\text{WR}}$	=	Write, laag als er geschreven wordt (output)
$\overline{\text{INT}}$	=	Interrupt, laag maken geeft een interrupt
$\overline{\text{M1}}$	=	Machine cyclus 1 in uitvoering (haal instructie)
LF inp	=	Laagfrequent input, hier kan de I/O-module geluid naar de luidspreker (TV) sturen.

19. CONSTANTEN, GETALLEN, VARIABELEN EN ARRAYS

Constanten
Tijdens de programma uitvoering werkt de BASIC interpreter met constanten. Twee soorten constanten kunnen worden gebruikt: **string-constanten** en **numerieke constanten.**

Een string-constante is een reeks van maximaal 255 alfanumerieke karakters en leestekens ingesloten tussen aanhalingstekens (" "), zoals bijv.
"HALLO"
"FL 98.21"
"Personal Computers"

Numerieke constanten zijn positieve of negatieve getallen. Er zijn 3 soorten numerieke constanten in BASIC mogelijk:

1. **Gehele getallen (integers)**

   a. Decimale getallen.
   Getallen zonder punt in het bereik -32768 t.m. 32767

   b. Hexadecimale getallen.
   Dit zijn integer getallen uitgedrukt in het 16-tallig stelsel. Ter onderscheiding van decimale getallen wordt de waarde vooraf gegaan door &H.
   Het bereik van deze getallen loopt van &H0000 tot &HFFFF.

   De hexadecimale getallen van 0 tot &H7FFF hebben een waarde van 0 tot 32767 en de hexadecimale getallen van &H8000 tot &HFFFF hebben een waarde van -32768 tot -1

   c. Octale getallen.
   Dit zijn gehele getallen in het 8-tallig stelsel. Ter onderscheiding voorafgegaan door & of &O (letter O). Het bereik van deze getallen loopt van &O000000 - &O177777.

   Octale getallen van &O0 tot &O077777 zijn positief, van &O100000 tot &O177777 zijn negatief.

2. **Enkele-precisie** getallen. Deze getallen kunnen een waarde hebben van -10^{38} t.m. 10^{37}. Als het getal in 7 cijfers kan worden weergegeven, wordt het afgedrukt als getal met decimale punt. (Floating point of drijvende-komma notatie). Grotere en kleinere getallen worden afgedrukt in de zgn. wetenschappelijke notatie met de letter E voor exponent. De precisie is nooit groter dan 7 cijfers.
   Voorbeelden:

7.325	2.3598E-7 (0.00000023598)
0.82765	6.723E9 (6723000000)
27.986	

3. **Dubbele-precisie getallen.** Het bereik van deze getallen is gelijk aan dat van enkele-precisie getallen. De precisie is echter 16 cijfers. Ter onderscheiding van enkele-precisie getallen worden de exponent met de letter D aangegeven.
   Voorbeeld:

7.3257926137243	6.25D17

Variabelen
Een variabele is een grootheid waarvan de waarde kan veranderen. De waarde van een variabele kan in het programma expliciet worden aangegeven of kan ook het resultaat van een berekening (expressie) of functies zijn.

Namen van variabelen.
De naam van een variabele mag in BASIC elke lengte hebben, echter alleen de eerste twee karakters worden door de interpreter als herkenning gebruikt; de variabele-namen ALFA en ALLE, bij voorbeeld, zijn voor de interpreter gelijk en duiden dus op dezelfde variabele.
Een variabele-naam *moet* beginnen met een letter; daarachter zijn alfanumerieke karakters toegestaan. Als laatste karakter kunnen enkele speciale tekens volgen die het soort variabele aanduiden (%, !, # en $) (Engels: type declaration characters).
Variabele-namen, ingegeven met kleine letters worden door de BASIC interpreter omgezet in hoofdletters.
Een variabele-naam mag geen gereserveerd woord zijn, evenmin mag een gereserveerd woord deel uitmaken van een variabele-naam. Tot de gereserveerde woorden in BASIC horen alle instructies en functienamen.

Ook operatoren, zoals +, -, *, ÷, / zijn in een variabele-naam niet toegestaan.
Aan een numerieke variabele kan als waarde een numerieke constante worden toegekend; aan string-variabelen een string waarde.
De waarde van numerieke variabelen kan van het type integer, enkele-precisie of dubbele-precisie zijn.
Behalve door een type-aanduidingsteken kan men het type variabele laten bepalen door de BASIC instructies DEFINT, DEFSTR, DEFDBL, DEFSNG (hoofdstuk 22). Wordt géén type-aanduidingsteken of type definitie gebruikt, dan wordt de variabele als enkele-precisie behandeld.
De volgende type-aanduidingstekens zijn in gebruik;

Teken	**Type variabele**
%	**Integer**
	Enkele-precisie variabele
#	**Dubbele-precisie variabele**
$	**String**

Enkele voorbeelden van variabele-namen:

toegestaan	**niet toegestaan**	
PIET$	FRED# (bevat gereserveerd woord FRE)	
PI#	DEF	DEF, LET, TO en STOP zijn gereserveerde woorden
DELTA!	LETTER	
	TO%	
NN$	STOP$	
A	7A%	beginnen niet met een letter
XY	#6	

Array's
Een array of getalrij of getaltabel is een reeks variabelen die in het programma met dezelfde variabele-naam worden aangeduid.
De afzonderlijke elementen uit de rij of tabel worden aangeduid met een aantal volgnummers (of indices) achter de variabele-naam.
Een array heeft drie dimensies als een element van dat array door drie indices wordt aangegeven.
Zo duidt V(10) een element aan in een 1-dimensionale getalrij; S(1,4) is een element in een 2-dimensionale rij, enz.
Het maximale aantal dimensies is 255; het maximale aantal elementen per dimensie is 32767 (maar een array met zoveel elementen past niet in het geheugen van de P2000).

Denk eraan dat de index 0 ook een geldige waarde is; deze duidt op het eerste element uit een array. V(0), is dan ook het 1e element van het array V.
Indien de index-waarden van een array in het programma of als resultaat van een expressie worden aangegeven met enkele of dubbele precisie, worden deze waarden afgerond tot integers (gehele getallen).
Elementen van array's kunnen zowel integer, enkele-precisie, dubbele-precisie als string waarden hebben; dit hangt af van het type array.

A2%(4)	rij van 5 integers	AZ$(12)	rij van 13 strings
V!(3,2)	vierkant van 4x3 enkele-precisie getallen	KL(19,2,1)	
ZX(2*3)	rij van 7 enkele-precisie getallen	AX%(5*A,B-3)	

Type conversie
De BASIC interpreter is in staat om een getal van een bepaald type om te zetten in een ander type. Hierbij gelden de volgende regels:

1. Wanneer een numerieke waarde van een bepaald type wordt toegekend aan een numerieke variabele van een ander type, zal die waarde worden opgeslagen volgens het type van de variabele. Hierbij maakt de BASIC-interpreter gebruik van de conversiefuncties CINT, CDBL, CSNG (zie hoofdstuk 22).
   (Wanneer een numerieke waarde wordt toegekend aan een string-variabele of andersom, geeft de interpreter een **Type mismatch** fout).
   Voorbeeld:
   Als aan de integer variabele A% de waarde 23.42 wordt gegeven zet de interpreter dit om naar 23, terwijl 23.67 naar 24 wordt afgerond. Omgekeerd kan een integer getal zonder meer aan een enkele-precisie variabele worden toegekend.

2. Tijdens de uitvoering van een expressie worden beide operanden (te bewerken gegevens) bij iedere operatie omgezet naar dezelfde graad van precisie, d.i. de in die operatie hoogst

voorkomende precisie of naar het type dat voor de operatie vereist is (zoals integer voor ÷ en MOD). Ook het resultaat van de operatie zal aanvankelijk die precisie hebben.

```
10 D# = 6#/7
20 PRINT D#
RUN
 .8571428571428571
 Ok
```

Deze berekening wordt uitgevoerd in dubbele precisie. Ook het resultaat in D# is weergegeven in dubbele precisie.

```
10 D = 6#/7
20 PRINT D
RUN
 .857143
 Ok
```

De berekening is uitgevoerd in dubbele precisie en het resultaat is toegekend aan D, een enkele precisie variabele. Er vindt dus een afronding plaats tot enkele precisie.

3. Wanneer een enkel of dubbel-precisie getal wordt omgezet in een integer, wordt het getal afgerond.

```
10 C% = 55.88
20 PRINT C%
RUN
 56
 Ok
```

Conversie naar integer kan een **Overflow** fout opleveren als de waarde buiten het bereik van integer waarden valt.

4. Wanneer aan een dubbele-precisie variabele een enkel-precisie of integer getal wordt toegekend, zullen alleen de eerste 7 cijfers van het resultaat betekenis hebben. De absolute waarde van het verschil tussen de afgedrukte dubbele-precisie en de originele enkele-precisie getallen zal daardoor altijd kleiner zijn dan 6.3E-8 maal het originele enkele-precisie getal.
Voorbeeld:

```
10 A = 2.04
20 B # = A
30 PRINT A; B#
RUN
 2.04   2.039999961853027
 Ok
```

20. EXPRESSIES, OPERATOREN EN FUNCTIES

Een expressie kan een constante of variabele zijn, maar ook een combinatie van constanten, variabelen en operatoren. We spreken van een stringexpressie als het resultaat een string is, en van een numerieke expressieals het resultaat een getal is.
Een operator geeft een rekenkundige of logische bewerking aan. De operatoren die de BASIC interpreter kent zijn als volgt in te delen:
- rekenkundige operatoren
- relationele operatoren
- logische operatoren
- functie-operatoren

De **rekenkundige** operatoren zijn, in volgorde van prioriteit:

Operator	Bewerking	Voorbeeld
↑ (SHIFT@)	machtsverheffen	X↑Y*
-	teken veranderen	-X
*/	vermenigvuldigen en	X*Y
	floating point delen	X/Y
÷	integer delen	X÷Y
MOD	modulo (rest v.e. deling)	X MOD Y
+-	optellen en	X+Y
	aftrekken	X-Y

Om deze volgorde van uitvoering te veranderen kunnen haakjes worden gebruikt; bewerkingen tussen haakjes worden het eerst uitgevoerd. Komen binnen de haakjes meer bewerkingen voor, dan geldt weer bovenstaande prioriteit.

Bij integer delen worden 2 getallen op elkaar gedeeld en de cijfers „achter de komma" worden weggelaten. De operanden worden afgerond tot integers (en moeten dan ook tussen -32768 en +32767 liggen), voordat de deling wordt uitgevoerd. Het resultaat van de deling wordt eveneens afgerond tot een integer.

*) Op het scherm en op de printer wordt het machtsverheffingsteken afgedrukt als ↑. Via het toetsenbord moet dit worden ingegeven met het teken boven de @ (SHIFT @) en niet met de cursortoets links van de spatiebalk. Het teken voor integer-deling bevindt zich linksboven op het kleine toetsenbord en is aangegeven met :

Voorbeeld:
$10 \div 4 = 2$
$25.68 \div 6.99 = 3$ (25.68 afgerond = 26, integer-gedeeld door 6.99, afgerond = 7, geeft als resultaat = 3).

Integer delen volgt in prioriteit direct na vermenigvuldigen en floating point delen.
Ook bij **modulo** berekeningen (berekenen van de „rest" bij een deling) moeten de operanden liggen tussen -32768 en +32767.
Voorbeeld:

10 MOD 4	= 2	(10/4 = 2 met als rest 2)
25.68 MOD 6.99	= 5	(26/7 = 3 met als rest 5)

Modulo berekeningen volgen in prioriteit direct na integer delen.
Wanneer in een berekening een „deling-door-nul" voorkomt geeft de interpreter de foutmelding **Division by zero** en stopt de uitvoering van het programma. Hetzelfde geldt wanneer het getal 0 tot een negatieve macht wordt verheven.
Het resultaat van een machtsverheffing voor een negatief getal tot een niet-gehele exponent kan wiskundig niet in een enkele of dubbele-precisie waarde worden uitgedrukt bijv. $(-3)\uparrow 2.35$. In dit geval meldt de P2000 **Illegal function call.**

Relationele Operatoren
Relationele operatoren worden gebruikt om twee waarden met elkaar te vergelijken. Het resultaat van zo'n vergelijking is „waar" of „onwaar", hetgeen in getalwaarden wordt uitgedrukt als resp. „waar" ongelijk aan 0 (-1) en „niet waar" gelijk aan 0. Het resultaat kan worden gebruikt om een beslissing te nemen ten aanzien van het verdere verloop van de programma- uitvoering. De BASIC-interpreter kent de volgende relationele operatoren:

Operator	Onderzochte relatie	Expressie
=	gelijk	X = Y
>< of <>	ongelijk	X >< Y
<	kleiner dan	X < Y
>	groter dan	X < Y
<= of =<	kleiner dan of gelijk aan	X ⩽ Y
>= of =>	groter dan of gelijk aan	X ⩾ Y

(Het = teken wordt ook gebruikt om een waarde toe te kennen aan een variabele).

Wanneer in een expressie zowel rekenkundige als relationele operatoren voorkomen, worden de rekenkundige bewerkingen eerst uitgevoerd; deze hebben een hogere prioriteit, bijv.
X + Y < (T - 1)/Z
is waar indien de waarde van X plus Y kleiner is dan de waarde van T minus 1, gedeeld door Z.
Andere voorbeelden van dergelijke combinaties zijn:

IF SIN(X) >< 0 GOTO 1000
IF I MOD J >= 0 THEN K=K+1

Logische operatoren
Logische operatoren voeren bewerkingen uit volgens de "booleaanse" regels. Het resultaat van een booleaanse berekening is een getal dat "waar" (ongelijk aan nul) of "onwaar" (nul) is. In een expressie worden logische bewerkingen uitgevoerd **na** rekenkundige en vergelijkende bewerkingen.
De volgende logische operatoren zijn in BASIC beschikbaar: AND, OR, XOR, NOT, IMP en EQV.

Waarheidstabel

X	Y	X AND Y	X OR Y	X XOR Y	NOT X	X IMP Y	X EQV Y
0	0	0	0	0	1	1	1
0	1	0	1	1	1	1	0
1	0	0	1	1	0	0	0
1	1	1	1	0	0	1	1

Voorbeeld

D > 200 AND F = 4 is ***waar*** als zowel **D > 200** als **F = 4**

I = 10 OR K < 0 is ***waar*** als of **= 10** en/of **K < 0**

NOT P is ***waar*** als **P *onwaar*** is. (dus ongelijk aan 0)

Logische operatoren zetten hun operanden eerst om in 16-bits binaire getallen (volgens de 2-complement methode). Het getalbereik hierbij is van -32768 ... +32767. Indien een operand buiten dit bereik valt, volgt een foutmelding.

De aangegeven logische bewerking wordt bitsgewijs uitgevoerd, d.w.z. elk bit van het resultaat wordt bepaald door de overeenkomstige bits van beide operanden. Logische operatoren kunnen dan ook worden gebruikt om bytes te testen op een bepaald voorkomend bitpatroon. De AND-operator kan bij voorbeeld worden gebruikt om bepaalde bits in een byte (8-bit woord) te "maskeren". De OR-operator kan worden gebruikt om twee bytes a.h.w. te mengen teneinde een bepaalde binaire waarde te verkrijgen. De volgende voorbeelden dienen om de werking van logische operatoren verder te verduidelijken:

(63 AND 16) = 16 — 63 = binair 111111 en 16 = binair 10000. 63 AND 16 is dus 100000 AND geeft de bits, die én in 63 én in 16 voorkomen.

(15 AND 14) = 14 — 15 = binair 1111 en 14 = binair 1110. 1111 AND 1110 = 1110 = 14

(-1 AND 8) = 8 — -1 = binair 1111111111111111 en 8 = binair 1000, of eigenlijk 0000000000001000

(4 OR 2) = 6 — 4 = binair 100 en 2 = binair 10. 100 OR 010 = 110 = 6 (de bits die óf in 4 óf in 2 óf in beide voorkomen)

(10 OR 10) = 10 — 10 = binair 1010. 1010 OR 1010 = 1010

-1 OR -2) = 1 — -1 = binair 1111111111111111 en -2 = binair 1111111111111110

(NOT X) = -(X + 1) — De 2-complement waarde van een getal is gelijk aan het bit-complement (alle bits inverteren), plus 1

(4 IMP 2) = -5 — 4 = 0000000000000100
2 = 0000000000000010
1111111111111011 = -5

(8 XOR 10) = 2 — 8 = 1000 10 = 1010
8 X0R10 = 0010 = 2 (bits die in 8 óf in 10 maar niet in beide voorkomen).

(12 EQV 10) = -7

12 = 0000000000000001100
10 = 0000000000000001010
1111111111111111001 = -7
(alle bits, die in 12 en 10 gelijk zijn)

Functies

In een expressie kunnen functies worden gebruikt om een bewerking uit te voeren. De interpreter kent de volgende functies voor getallen (zie hoofdstuk 22 voor een nadere beschrijving).

functie	**omschrijving**	**resultaat**
ABS(X)	absolute waarde	enkele-precisie
ASC(X$)	ASCII-waarde van 1e karakter van X$	integer
ATN(X)	boogtangens	enkele-precisie
EXP(X)	e-macht	enkele-precisie
INT(X)	integer waarde	integer
LOG(X)	natuurlijke logarithme	enkele-precisie
RND(X)	random getal tussen 0 en 1	enkele-precisie
SGN(X)	teken van X (1,0 of -1)	integer
SIN(X)	sinus van hoek in radialen	enkele-precisie
SQR(X)	vierkantswortel	enkele-precisie
TAN(X)	tangens van hoek in radialen	enkele-precisie
VAL(X$)	"getalwaarde" van string X$	enkele-precisie
LEN(X$)	aantal karakters van string X$	integer

21. STRINGBEWERKINGEN EN STRINGFUNCTIES

Strings mogen worden samengevoegd door gebruik te maken van het + teken.

```
10 A$ = "FILE" : B$ = "NAAM"
20 C$ = A$ + B$ : PRINT C$
30 PRINT "NIEUWE " + A$ + B$
RUN
FILENAAM
NIEUWE FILENAAM
 Ok
```

Ook is het mogelijk om strings met elkaar te *vergelijken.* Hiervoor worden dezelfde relationele operatoren gebruikt die op numerieke waarden van toepassing zijn:
=, >, <, = >, > =, < =, = <, < >, > <.
Stringvergelijkingen worden uitgevoerd door één voor één de ASCII-waarden (zie tabel 1) van de afzonderlijke tekens in de strings met elkaar te vergelijken.
Indien alle ASCII-waarden gelijk zijn, zijn beide strings gelijk. Verschillen de ASCII-waarden dan geldt een string met kleinere ASCII-waarden als kleiner dan een string met grotere ASCII-waarden. Hoofdletters worden dus "kleiner" gezien dan kleine letters. Denk er aan dat ook spaties bij de vergelijking worden betrokken.
Bij strings van ongelijke lengte waarbij het begin van de langere string gelijk is aan de hele kortere string, wordt de kortste string beschouwd als de kleinste.

```
"AA" < "AB"
"FILENAAM" = "FILENAAM"
"X&" > "X#"
"kg" > "Kg"
"JAN" < "JANSEN"
B$ < "9/12/80" als B$ = "80/12/09"
```

Stringvergelijkingen kunnen dus worden gebruikt om de waarde van een string te onderzoeken, om strings te alfabetiseren, enz. Alle strings in stringvergelijkingen moeten zijn gevat in aanhalingstekens.

Ook string-variabelen kunnen worden vergeleken:

```
10 A$ = "AA" : B$ = "BB"
20 IF A$ < B$ THEN PRINT "Kleiner"
RUN
Kleiner
 Ok
```

Stringfuncties

De interpreter heeft de volgende functies beschikbaar voor strings.

CHR$ (I)	string van 1 karakter lengte met ASCII-waarde I
HEX$ (X)	hexadecimale string-representatie van getal X (eventueel afgerond)
INSTR(I,X$,Y$)	zoekt vanaf het I-e karakter in X$ of de string Y$ in X$ voorkomt
LEFT$(X$,I)	linker I karakters van string X$
MID$(X$,I,J)	deel van X$, te beginnen bij het I-de karakter, en J karakters lang
OCT$(X)	octale representatie van getal X
RIGHT$(X$,I)	rechter I karakters van string X$
SPACE$(I)	string van I spaties
STR$(X)	getal X als string
STRING$(I,X$)	string, I karakters elk gelijk aan (het eerste karakter van) X$

Alle functies worden uitgebreid besproken in hoofdstuk 22.

22. BASIC-INSTRUCTIES EN FUNCTIES

In dit hoofdstuk worden alle instructies en functies beschreven die de BASIC interpreter kent. Instructies en functies staan door elkaar in alfabetische volgorde.
Elke beschrijving bestaat uit de volgende delen:

Tekst instructie:	Toont de mogelijke opbouw en schrijfwijzen van de instructie of functie
Korte beschrijving:	Geeft aan voor welke doeleinden de instructie kan worden gebruikt
Opmerkingen:	In detail wordt aangegeven hoe de instructie moet worden gebruikt.
Voorbeeld:	Toont m.b.v. eenvoudige programma's het gebruik van de instructie of functie.

1. Tekst die in hoofdletters is aangegeven moet precies zo worden ingevoerd, bijv. **GOTO**
2. Cursief gedrukte delen moeten worden vervangen door een uitdrukking van de gevraagde soort, bijv. **GOTO *regelnummer*** wordt **GOTO 100**
3. Alle leestekens moeten zoals aangegeven worden gebruikt.
4. Waar is aangegeven ***getal*** of ***string*** mag ook een resp. numerieke of stringexpressie of variabele worden ingevuld. Waar zulks niet is toegestaan wordt dit vermeld.
5. Als geen foutafhandelingsroutine aangeroepen wordt, worden fouten op het beeldscherm afgedrukt als
   foutomschrijving **in** ***regelnummer***

ABS ***(getal)***

Functie. Berekent de absolute waarde van ***getal***

```
PRINT ABS(7*(-5))
 35
 Ok

PRINT ABS(345)
 345
 Ok
```

ASC ***(string)***

Functie. Geeft de numerieke waarde van de ASCII-waarde van het eerste karakter van ***string***
Is de lengte van ***string*** gelijk aan nul dan volgt de fout
Illegal function call

```
10 X$ = "TEST"
20 PRINT ASC(X$)
RUN
 84
 Ok
```

ATN ***(getal)***

Functie boogtangens. Berekent de hoek, uitgedrukt in radialen, waarvan de tangens gelijk is aan ***getal.***
Het resultaat zal liggen tussen -pi/2 en +pi/2 waarbij pi = 3.141592

```
10 INPUT X
20 PRINT ATN(X)
RUN
? 3
 1.24905
 Ok
```

Opmerking: Met de ATN(X)-functie kunnen ARCSIN,ARCCOS etc. worden berekend.

AUTO
AUTO ***regelnummer***
AUTO ***regelnummer, toename***
AUTO, ***toename***

Opdracht tot het automatisch genereren van een volgend regelnummer na elke ENTER tijdens het intikken van een programma.
De AUTO-instructie begint het nummeren van de regels bij ***regelnummer*** en verhoogt na iedere ENTER het regelnummer met ***toename***.
Is ***regelnummer*** niet ingevuld maar ***toename*** wel, dan begint AUTO bij 0 te nummeren.
Worden ***regelnummer*** en ***toename*** niet opgegeven dan begint AUTO te nummeren bij 10 en hoogt met 10 op.
Als het regelnummer op het scherm verschijnt kan de BASIC programmaregel worden ingetikt.
Indien AUTO een regelnummer genereert dat reeds in het geheugen van de computer aanwezig is, wordt achter dat regelnummer een ster * afgedrukt om de programmeur te waarschuwen.
Wordt direct achter de * op ENTER gedrukt dan wordt de bestaande programmaregel gewist uit het geheugen en genereert AUTO een volgend regelnummer. Wordt na de * tekst ingetypt en afgesloten met ENTER, dan wordt de reeds in het geheugen aanwezige regel vervangen door de nieuwe regel.
Wil men de bestaande regel niet wissen of overschrijven dan moet de AUTO-stand worden beëindigd.
AUTO wordt beeindigd door de SHIFT-STOP toets in te drukken. Gebeurt dit tijdens het invoeren van een programmaregel, dan wordt deze regel niet in het geheugen opgeslagen.
Na de SHIFT-STOP schakelt de BASIC-interpreter de P2000 in de directe stand.

AUTO	Genereert de regelnummers 10, 20, 30, 40 enz.
AUTO 250	Genereert de regelnummers 250, 260, 270, enz.
AUTO 100,50	Genereert de regelnummers 100, 150, 200, enz.
AUTO, 2	Genereert regelnummers 0, 2, 4, enz.

Auto is alleen nuttig tijdens het intypen van een programma, dus in directe stand.

CDBL *(getal)*

Zet ***getal*** om in een dubbele-precisie waarde.

```
10A# = 454.7600158429368
20 B = A#
30 PRINT A#;B: PRINT CDBL(B)
RUN
 454.7600158429368  454.76
 454.760009675625
 Ok
```

De toegevoegde decimalen zijn niet betrouwbaar. De functie CDBL wordt door de BASIC interpreter automatisch aangeroepen bij het toekennen van een enkele-precisie waarde aan een dubbele-precisie variabele.

CHR$*(getal)*

Functie. Levert een string van één karakter lengte af met ASCII-waarde ***getal***. ***getal*** wordt omgezet naar integer en mag 0 t.m. 255 zijn.

CHR$ is vooral belangrijk in PRINT-opdrachten, waarin deze functie de cursor bestuurt, kleur- en grafische omschakelkarakters afdrukt, etc.
Voorbeelden zijn te vinden in de hoofdstukken over beeldscherm, kleur, grafische mogelijkheden, geluid en printer.
Zie ook tabel 1.

CINT *(getal)*

Functie. Geeft integer waarde van ***getal*** door afronding.

```
PRINT CINT(45.67)
 46
 Ok
```

Zie ook de functies FIX en INT.
Bij de behandeling van de functie INT worden CINT, INT en FIX vergeleken.

CLEAR
CLEAR ***getal***
CLEAR ***getal 1, getal 2***

Wissen van alle variabelen in het geheugen, reserveren van geheugenruimte voor strings en vastleggen van de hoogst beschikbare geheugenplaats voor BASIC.
getal of ***getal 1*** geeft de hoeveelheid geheugenruimte aan die wordt gereserveerd als stringruimte. ***getal 2*** geeft het adres van de laatste geheugenplaats die in het RAM geheugen ter beschikking is van de BASIC-interpreter.
De ruimte boven de aangegeven geheugenplaats (de hogere adressen) kan vrij worden gebruikt, bijv. voor het opbergen van machinetaal-routines.
Tenzij anders aangegeven met ***getal*** of ***getal 1*** is de totale ruimte voor strings gelijk aan 50 geheugenplaatsen, tenzij reeds eerder een CLEAR commando was gegeven. Dan blijft de „oude" waarde gehandhaafd, dit geldt ook voor ***getal 2***.
Indien de BASIC-interpreter onvoldoende ruimte beschikbaar heeft voor de opslag van strings, volgt de fout **Out of string space**.
Is er geen geheugenruimte meer aanwezig voor de opslag van variabelen, volgt de fout **Out of memory**.

CLEAR	Wist alle variabelen
CLEAR 2000	Wist alle variabelen en zet de stringruimte op 2000 karakters
CLEAR 2000,&H9000	Wist alle variabelen, zet de string-ruimte op 2000 karakters en de BASIC grens op &H9000 zodat de adressen van &H9001 en hoger vrij zijn voor ander gebruik.

Opmerkingen: CLEAR maakt ON ERROR GOTO ongedaan, RESTORE wordt RESTORE 0, alle variabelen zonder type aanduiding, die vanaf hier worden gebruikt zijn enkele-precisie variabelen en van GOSUB en FOR ... NEXT lussen zijn de terugsprong-adressen gewist. CLEAR kan dus niet in een subroutine worden gebruikt.
Als ***getal 2*** groter is dan de totaal beschikbare geheugenruimte RESET de computer zichzelf.

CLOAD
CLOAD *string*

Zoeken op de cassette naar het programma ***string*** en dit programma in het geheugen laden.
Alleen het eerste karakter van ***string*** is significant.
Nadat een CLOAD-opdracht is uitgevoerd komt de BASIC interpreter altijd terug in de directe stand. Indien het programma dat men wenst in te lezen niet op de cassetteband aanwezig is, volgt de fout **Niet gevonden**.
Als ***string*** wordt weggelaten wordt het eerste programma of bestand geladen dat op de cassette voorkomt.

CLOAD	Laadt het eerste programma van de cassette.
CLOAD "Viewdata"	Laadt programma "V".
CLOAD "V"	Idem
CLOAD A$	Laadt het programma van de naam die aan de variabele A$ is gegeven.
CLOAD RIGHT$ (D$,1)	Laadt het programma, waarvan de beginletter gelijk is aan de laatste letter van de waarde van D$

Als de gegevens op de band geen programma vormen, maar bijv. een array, dan wordt dit array als programma gezien en geladen.
Bij de uitvoering treedt dan vrijwel altijd een foutmelding op.
Is het te laden programma groter dan de beschikbare geheugenruimte, dan wordt de fout **Out of memory** gegeven.
Dit kan ook gebeuren als het RUN-nen van een voorgaand programma d.m.v. een CLEAR commando de voor BASIC beschikbare geheugenruimte heeft verkleind. In dat geval kan men proberen of het programma geladen kan worden nadat eerst op de RESET-toets gedrukt is.
Is geen cassette aanwezig of is het klepje niet gesloten, dan verschijnt de foutmelding **Geen cassette**.
Voordat een programma van de cassette band wordt ingelezen voert CLOAD eerst een NEW-opdracht uit, waardoor het geheugen wordt gewist.

CLOAD* ***arraynaam***
CLOAD* ***arraynaam @ string***

Zoeken op de cassette naar het numerieke array ***arraynaam*** of ***string*** en dit in het geheugen laden, op de plaats van het array ***arraynaam***.
CLOAD* ***arraynaam*** zoekt het array waarvan de eerste letter overeenkomt met ***arraynaam*** op de band en laadt dit in de voor dit array gereserveerde ruimte.
CLOAD* ***arraynaam @ string*** zoekt op de band het bestand met de beginletter van ***string*** en laadt dit in het geheugen op de plaats gereserveerd voor ***arraynaam***.

CLOAD* A% Zoekt en laadt het array met naam **A....** in het array **A%**.
CLOAD* A%@ "Test" Laadt het bestand met naam **T** in array **A%.**

```
100 D$ ="Januari"
110 CLOAD* M@D$
```

Laadt in dit geval de file **J.....** van de band en zet deze in het enkele precisie array M.
Als voor het array, dat gevuld moet worden niet van te voren een DIM instructie is gegeven, of dit op een andere manier bekend is wordt de fout **Illegal function call** gegeven.
Is niet voldoende ruimte gereserveerd d.m.v. een DIM, dan volgt de fout **Out of memory**.
De fout **Geen cassette**, wordt veroorzaakt als geen cassette aanwezig is, of het klepje niet gesloten.

CONT

Opdracht tot het voortzetten van de programma-uitvoering nadat de SHIFT-STOP toets is ingedrukt, nadat een STOP of END instructie is uitgevoerd, of na een ERROR. De programma-uitvoering gaat verder op het punt waar de onderbreking optrad.
Was dit het geval na de hulpstring van een INPUT of LINEINPUT, dat wordt de uitvoering voortgezet door eerst nogmaals die hulpstring op het scherm af te drukken.
CONT wordt meestal gebruikt in combinatie met de STOP-instructie of het indrukken van de STOP-toets bij het testen

van een programma. Is de programma-uitvoering gestopt, dan kunnen de waarden die de gebruikte variabelen op dat moment hebben, worden onderzocht m.b.v. PRINT-opdrachten in de directe stand, bij voorbeeld PRINT A,B om de waarden van de variabelen A en B te onderzoeken of d.m.v. FOR I = 1 TO 100: PRINT A(I): NEXT voor het bekijken van de waarden van een array. Behalve met CONT kan de programma-uitvoering ook worden voortgezet m.b.v. de GOTO-opdracht in de directe stand.
Na het optreden van fouten, kan soms geen gebruik worden gemaakt van CONT, omdat CONT opnieuw de instructie uitvoert waar de fout optrad, zodat de foutmelding na CONT opnieuw verschijnt.
Dit geldt bijv. niet voor cassette fouten, zoals **Geen cassette**. Door een cassette te plaatsen kan deze fout worden hersteld. CONT kan niet worden gebruikt wanneer na het optreden van de programma-onderbreking veranderingen in het programma zijn aangebracht. Wordt in dat geval toch een CONT-opdracht gegeven, dan verschijnt de foutmelding **Can't continue**.
Waarden van variabelen daarentegen, mogen in de directe stand worden gewijzigd alvorens de CONT-opdracht te geven.
CONT is alleen werkzaam in de directe stand.

✳

CSAVE✳ *arraynaam*
CSAVE✳ *arraynaam* @ *string*

Schrijven van een array uit het geheugen naar de cassetteband.

CSAVE✳ *arraynaam* schrijft het opgegeven array naar de band. De elementen van een meer-dimensionale array worden zodanig opgeslagen, dat de eerste index het snelst varieert.
CSAVE✳ *arraynaam* @ *string*
Schrijft het array, dat zich in het geheugen onder ***arraynaam*** bevindt op de band onder de naam ***string*** waarbij ***string*** maximaal 16 karakters lang mag zijn.

CSAVE✳ A%	Schrijft array **A%** als **A** met type INT.
CSAVE✳ A@"Test"	Schrijft array **A** weg onder de naam **Test** met type SNG

✳ cos (getal)
functie - geeft cosinus v/o hoek 'getal' waarby 'getal' is uitgedrukt in radialen
- print cos (1.5)
.0.707371

10 D$ = "JAN"
20 CSAVE* A@"Uitgaven " + D$
Schrijft array **A** weg onder de naam **Uitgaven JAN**

Als ***string*** een CHR$(0) bevat worden de eerste drie letters hierna gezien als type-aanduiding (evt. aangevuld met spaties)

CSAVE* A% @ "Plaatje" + CHR$(0) + "PEP"
Schrijft array **A%** weg onder de naam **"Plaatje"** met type **"PEP"**.

Opmerking: Het type van het bestand op de band is gelijk aan het type van het array (integer, enkele of dubbele precisie). String-arrays kunnen niet worden weggeschreven op deze manier.
Als geen cassette aanwezig is veroorzaakt dit de fout:
Geen cassette
De cassette dient in de linker bovenhoek van een zwart stopje te zijn voorzien. Anders verschijnt de fout:
Geen stopje
Wordt CSAVE* in de directe stand gebruikt en staat er op de band al een bestand met dezelfde beginletter dan wordt naam en type van dit array getoond met de vraag
Hier overheen? (J/N).
Wordt met J of Y geantwoord, dan wordt het „oude" array op de band overschreven. In de programmastand wordt deze vraag niet gesteld (zie echter tabel 2).
Is het nieuwe bestand groter dan het oude, dan kan de vraag **Mag de rest weg ? (J/N)** verschijnen (ook in de programma-stand).

CSAVE string
Bewaren van een programma op de cassetteband.
CSAVE ***string*** schrijft het programma, dat in het geheugen staat naar de band onder de naam ***string***. ***string*** mag maximaal 16 karakters lang zijn. Het type van de gegevens is BAS.

CSAVE "sorteer"

10 L$ = "per 1/1"
20 CSAVE "leden " + L$

Schrijft het programma naar de band onder de naam **leden per 1/1**.

Zie voor foutmeldingen betreffende cassette en stopje en voor de waarschuwing **Hier overheen? (J/N)** de instructie CSAVE*

CSNG (getal).

Functie. ***getal*** wordt omgezet naar een enkele-precisie waarde. De BASIC interpreter gebruikt deze functie voor het toekennen van een integer of dubbele-precisie waarde aan een enkele-precisie variabele.

```
10 A# = 975.3421#
20 PRINT A#; CSNG (A#)
RUN
 975.3421  975.342
 Ok
```

DATA *lijst van constanten*

In het programma plaatsen van een lijst van numerieke waarden of strings.
De opdracht READ kent de waarde van deze constanten toe aan variabelen.

DATA instructies worden door de BASIC-interpreter overgeslagen in de programma-uitvoering. Ze worden pas gebruikt na een READ-opdracht. DATA-regels mogen op willekeurige plaatsen in het programma staan. Een DATA-lijst mag zoveel waarden bevatten als er op een programmaregel kunnen. De getallen of strings worden gescheiden door een komma (,). De DATA-lijst kan beëindigd worden door een dubbele punt (:). In een programma mag een willekeurig aantal DATA-regels worden opgenomen. De DATA-regels worden door READ-instructies in volgorde van regelnummer afgewerkt. De getallen en strings in alle DATA-regels mogen worden beschouwd als één lijst, ongeacht de plaats of het aantal DATA-regels.
lijst van constanten mag constanten bevatten van elk in BASIC toegestaan type, d.w.z. strings, integers, enkele en dubbele-precisie in floating point en wetenschappelijke notatie. Expressies zijn niet toegestaan, ze worden automatisch als strings gezien. Strings en getallen mogen door elkaar in een DATA-lijst voorkomen, uiteraard gescheiden door een komma.

Als in een string komma's, dubbele punten en/of spaties aan het begin voorkomen, moet de string tussen aanhalingstekens (") geplaatst worden.
Het type variabele dat in de READ-instructie is aangegeven moet overeenstemmen met het type constante dat in de DATA-lijst wordt gelezen. Is dat niet het geval dan volgt een **Syntax error** fout met het regelnummer van de DATA-lijst.
Wanneer het programma tracht om constanten te lezen wanneer alle constanten reeds zijn gelezen, volgt de fout **Out of data**.
Een reeds afgewerkte DATA-lijst kan opnieuw worden gebruikt door toepassen van de RESTORE-instructie.

```
10 DATA 2,3,Jan," woont in",3.14E4,Eindhoven,Nederland
20 READ A%, B%
50 READ NA$, T$, FL
80 READ PL$, LA$
```

Een DATA-lijst mag in een programma-regel voorkomen.

```
100 RESTORE 100: DATA 10,20,30: FOR I% = 1 TO 3: READ
A: NEXT: PRINT
```

DEF FN ***naam*** = ***functie definitie***
DEF FN ***naam (parameters)*** = ***functie definitie***

Definiëren en benoemen van een, door de gebruiker opgestelde, functie.

naam moet een geldige variabele naam zijn. Met deze naam, voorafgegaan door FN, wordt de functie aangeduid, en het type aangegeven. ***parameters*** bestaat uit die variabele namen in de functie definitie die, wanneer de functie wordt aangeroepen, moeten worden vervangen door de waarden van de expressie in de aanroep. De parameters worden van elkaar gescheiden door een komma.
functie definitie is een expressie waarmee de eigenlijke functie wordt aangegeven. De lengte van deze expressie is beperkt tot één programma regel. Namen van variabelen die in de functie voorkomen dienen uitsluitend om die functie te definiëren; ze hebben geen invloed op de programma-variabelen met dezelfde naam. Een in de functie vermelde variabele hoeft niet noodzakelijk in de lijst van parameters voor te komen. Is dit niet het geval dan wordt de variabele in de functie

vervangen door de waarde die de gelijknamige variabele in programma op het moment van aanroep van de functie heeft. Zowel numerieke als string-functies kunnen worden gedefinieerd. Bij aanroep van de functie dienen de parameters van hetzelfde type te zijn als bij de definitie, anders volgt een **Type mismatch** fout. Als een functie met parameters is gedefinieerd moet de aanroep evenveel parameters bevatten, anders volgt **Syntax error**. Voordat een functie mag worden aangeroepen, moet eerst een DEF FN instructie zijn uitgevoerd, waarin die functie is gedefinieerd. Gebeurt dit niet, dan volgt een **Undefined user function** fout. DEF FN kan niet in de directe stand worden gebruikt.
In de definitie van een functie mag een andere gedefinieerde functie voorkomen,

```
DEF FN EX(X) = EXP(X)
DEF FN Y(X) = FN EX(X) + FN EX(-X)
```

Ook in de aanroep mag de parameter een gedefinieerde functie zijn.

```
PRINT FN EX(FN EX(1.5)).
```

```
410 DEF FNAB (X,Y)=X↑3/Y↑2
420 T = FNAB(I,J)
```

In regel 410 wordt de functie FN AB gedefinieerd. In regel 420 wordt deze functie aangeroepen, waarbij voor X en Y de waarde van de variabelen I en J worden ingevuld.

```
10 DEF FN PR$(VE,HO)=CHR$(4)+CHR$(VE)+CHR$(HO)
100 PRINT FN PR$(5,10) "cursor hier"
```

PRINT FN PR$(5,10) verplaatst eerst de cursor naar regel 5, kolom 10 en drukt daar de tekst **cursor hier** af.

DEFINT ***bereik(en) van letters***
DEFSNG ***bereik(en) van letters***
DEFDBL ***bereik(en) van letters***
DEFSTR ***bereik(en) van letters***

Aangeven van de typen variabelen: integer, enkele-precisie, dubbele-precisie of string.

Deze instructies geven het type aan van de variabelen waarvan de eerste letter is vermeld. Echter, een type-aanduidingskarakter bij een variabele (%, #, ! of $) heeft tijdens uitvoering altijd voorrang op een DEF- instructie. Indien in een programma geen DEF-instructies voorkomen gaat de BASIC-interpreter er van uit dat alle variabelen zonder type-aanduidingskarakter enkele-precisie variabelen zijn. In een programma mogen op verschillende plaatsen DEF-type instructies voorkomen. De laatst-uitgevoerde geldt.

10 DEFDBL L-P Alle variabelen zonder type aanduiding die beginnen met de letter L, M, N, O of P zijn dubbele-precisie variabelen.

10 DEFSTR A Alle variabelen zonder type aanduiding die beginnen met de letter A zijn string variabelen.

10 DEFINT I-N,W-Z Alle variabelen zonder type aanduiding die beginnen met de letter I, J, K, L, M, N, W, X, Y of Z zijn integer variabelen.

10 DEFSNG C,K Alle variabelen zonder type aanduiding die met de letter C of K beginnen zijn enkele-precisie variabelen.

DEF USR *cijfer* = *adres*

Aangeven van het startadres van een subroutine in machinetaal.

cijfer mag elk cijfer zijn van 0 t.m 9. Het cijfer correspondeert met het nummer van de subroutine, die wordt aangeroepen. Wordt ***cijfer*** weggelaten, dan gaat de BASIC interpreter uit van USR0

adres is het eigenlijke startadres van de subroutine.

In een BASIC-programma mag een willekeurig aantal DEF USR instructies voorkomen, teneinde een startadres van steeds een andere subroutine aan te geven. Het aantal mogelijke machinetaal subroutines dat met een BASIC-programma op deze manier kan worden gekoppeld is dus eveneens onbeperkt, hoewel er hoogstens 10 tegelijk gedefinieerd kunnen zijn.

200 DEF USR4 = &H9800

Het startadres van subroutine 4 is &H9800.
Startadressen van machinetaal-routines worden niet gewist door NEW, CLEAR, CLOAD.
Zie ook hoofdstuk 17: Opmerkingen over het programmeren in machinetaal.

DELETE ***regelnummer***
DELETE ***regelnummer 1-regelnummer 2***

Verwijderen van één of meer programmaregels.

Nadat de DELETE opdracht is uitgevoerd komt de interpreter terug in de directe stand. Indien ***regelnummer*** niet bestaat, volgt een **Illegal function call** fout en is er niets gewist.

DELETE 40 wist regel 40.
DELETE 40-100 wist de regels 40 t/m 100.
DELETE -40 wist alle regels t/m 40.
DELETE 40-. wist alle regels van 40 t/m de huidige regel (dit is de regel waar het laatst iets in veranderd is, ge-LIST is, of waarin een fout is opgetreden).
DELETE .-40 wist alle regels van de huidige regel t/m 40.

Een enkele regel kan ook uitgewist worden door alleen het regelnummer in te typen en daarna op de ENTER toets te drukken.
DELETE voert automatisch een CLEAR-opdracht uit.

DIM ***arraynaam (getallen), arraynaam (getallen),...***

Aangeven van de maximum waarden van de indices bij arrays. Bovendien wordt een overeenkomstige hoeveelheid geheugenruimte gereserveerd. ***getallen*** bestaat uit één of meer getallen gescheiden door een komma.
Wanneer een arrayelement wordt gebruikt zonder dat tevoren met een DIM- instructie is aangegeven uit hoeveel elementen het array zal bestaan, gaat de BASIC interpreter ervan uit dat de hoogste index 10 is, dus dat het array uit 11 elementen bestaat (0 t.m 10). Als een array element gebruikt wordt met een index die groter is dan de maximale aangegeven index, of kleiner is dan 0 dan volgt een **Subscript out of range** fout. De kleinst mogelijke waarde van een index is altijd 0. De DIM-instructie kent aan alle elementen van de aangegeven arrays de waarde 0 toe en geeft alle elementen van string-arrays de lengte 0.

10 DIM A(20) aangegeven wordt dat de getallenrij uit maximaal 21 elementen (0 t/m 20) zal bestaan.

```
20 FOR I=0 TO 20
30 READ A(I)
40 NEXT I
50 DATA .....
```

```
10 A=10 : B=8
20 DIM C%(A*8), Q(A*B,A+B)
```

Herdimensionering van arrays kan alleen plaatsvinden, nadat die arrays met een ERASE opdracht zijn gewist, anders volgt de fout **Redimensioned array** .

EDIT ***regelnummer***
EDIT.

Wijzigen van programmaregel ***regelnummer,*** resp. van de laatst ge-EDIT-e of ge-LIST-e programmaregel. Voor een bespreking van de EDIT instructie wordt verwezen naar hoofdstuk 11 van deze handleiding.

Samenvatting.

Tijdens EDIT zijn 3 standen mogelijk.
1. De uitgang-stand.
2. De wijzig-stand. Hierin kan tekst worden overschreven.
3. De invoeg-stand. Hierin kan tekst worden tussengevoegd.

In **alle standen** kunnen worden gebruikt.
Cursor-pijlen: beweeg de cursor door de programmaregel.
ENTER-toets: beëindig de EDIT-stand en bewaar de gewijzigde programmaregel.
STOP-toets: beëindig de EDIT-stand en herstel de programma regel in de toestand van vóór het EDIT-en.
wis-regel-toets: als de STOP-toets.
CORRECTIE-toets: wis karakter voor de cursor en sluit tekst aan.
SHIFT-correctietoets: wis karakter onder de cursor en sluit tekst aan.
SHIFT→: zoek volgende "zoek"-karakter.

Alléén in de **uitgang-stand** werken.
C schakel naar de wijzig-stand (change)
I schakel naar de invoeg-stand
H wis programmaregel vanaf cursor en ga in invoeg-stand
X zet cursor aan het eind van de programmaregel en ga in invoeg-stand
S zoek het karakter dat na S wordt ingetikt en plaats de cursor op dat karakter
K als S en wis onderweg
N beëindig de EDIT-stand, en zet de volgende programmaregel in de EDIT-stand

In de **wijzig-stand** komen letters, cijfers en tekens op de plaats van de cursor in de plaats van het teken dat daar staat.
CODE-toets : terug naar uitgang-stand

In de **invoeg-stand** worden letters, cijfers, tekens ingevoegd in de tekst op de plaats van de cursor. Vervolg van de tekst schuift op.
CODE-toets : terug naar uitgang-stand.

EDIT. zet de laatst ge-EDIT-e of ge-LIST-e programmaregel in de EDIT-stand. Dit zelfde gebeurt door het indrukken van de DEF-toets (SHIFT-0).
Het laatst-ingegeven "zoek" karakter blijft bewaard, óók na het verlaten van de EDIT-stand en na RUN. Er kan dus steeds weer met SHIFT→ verder worden gezocht in alle programma-regels.
Na EDIT komt de P2000 in de directe stand.

END

Beëindigen van de programma-uitvoering en terugkeren naar de directe stand. END-opdrachten mogen op elke willekeurige plaats in het programma worden opgenomen teneinde de programma-uitvoering op die plaats te beëindigen. Na uitvoeren van de END-opdracht verschijnt **Ok** op het scherm.

520 IF K=1000 THEN END ELSE GOTO 20

Als in het programma na de END-opdracht nog andere instructies volgen, kan men het programma vervolgen met CONT.

```
10 K=0
20 K=K+1:IF K=5 THEN END
30 PRINTK;:GOTO 20
RUN
 1 2 3 4
 Ok
CONT
 5 6 7 8 9
```

ERASE lijst van arraynamen

Verwijderen van arrays uit het geheugen.
Nadat een array is verwijderd, mag opnieuw een array onder dezelfde array-naam worden gedimensioneerd. Ook mag de vrijgekomen geheugenruimte voor andere doeleinden worden gebruikt.
Wordt een poging gedaan om een array opnieuw te dimensioneren zonder dat dit array eerst is gewist m.b.v. de ERASE-instructie dan geeft dit een fout **Redimensioned array**

```
100 DIM A(200),B(5,10)
450 ERASE A,B        wis de arrays A en B.
460 DIM B(99)        dimensioneer array B
```

Wordt getracht een array te wissen, dat niet gedimensioneerd was, dan volgt de fout: **Illegal function call**.

ERL en **ERR**

ERL: functie die het regelnummer geeft waarin een fout is opgetreden.
ERR: functie die de foutcode geeft na het optreden van een fout.
Zie verder bij ERROR.

ERROR ***getal***

Het simuleren van een fout in het programma, met een zelf te bepalen foutcode.
De waarde van ***getal*** wordt omgezet naar integer en moet groter zijn dan 0 en kleiner dan 255. Indien de waarde van ***getal*** gelijk is aan een foutcode die al door de BASIC interpreter wordt gebruikt (zie tabel 4), zal ERROR het optreden van die fout simuleren.

Om zelf een foutcode te definieren moet men een waarde gebruiken die groter is dan de grootst voorkomende BASIC-foutcode; nu groter dan 80. Aanbevolen wordt om zo hoog mogelijke waarden te gebruiken, dit i.v.m. een eventuele uitbreiding van de BASIC-foutcodes. De door de gebruiker te definieren foutcodes kunnen van pas komen bij foutafhandelingsroutines.
Indien een ERROR instructie een code aangeeft waarbij geen foutmelding behoort, dan zal de boodschap **Unprintable error in *regelnummer*** verschijnen.
Indien een ERROR instructie wordt uitgevoerd waarbij geen foutafhandelingsroutine behoort wordt de foutmelding op het beeldscherm weergegeven en stopt de programma-uitvoering.
Bij de foutafhandelingsroutines kan gebruik worden gemaakt van twee speciale functies: ERR en ERL. De waarde van de functie ERR is gelijk aan de code van de opgetreden fout, terwijl de functie ERL het regelnummer geeft waarin die fout optrad.
Indien de opdracht, waarin de fout optrad, werd uitgevoerd in de directe stand, zal ERL de waarde 65535 hebben.
Let er op dat ERL en ERR gereserveerde woorden zijn en dus niet mogen voorkomen in variabele-namen.

```
10 S=0
20 T=5
30 ERROR S+T
40 END
RUN
 String too long in 30
```

In de directe stand:

```
ERROR 5
String too long
```

```
110 ON ERROR GOTO 400
120 INPUT"Hoeveel zet u in";B
130 IF B>5000 THEN ERROR 210
350 END
400 IF ERR=210 THEN PRINT"De limiet is f 5000.-"
410 IF ERL=130 THEN RESUME 120
```

Zie ook ON ERROR GOTO en hoofdstuk 12: Programmafouten.

EXP (getal)

Functie: Het grondtal van de natuurlijke logarithme (e), verheven tot de macht ***getal***.
getal moet kleiner dan of gelijk zijn aan 87.3365. Is dit niet het geval dan veroorzaakt dit een **Overflow** fout en wordt de programma- uitvoering onderbroken.

```
10 X = 5
20 PRINT EXP (X-1)
RUN
 54.5982
 Ok
```

FIX (X)

Functie: Levert het deel van een getal af, dat zich vóór de decimale punt bevindt. De functie FIX is gelijk aan:
SGN (X) * INT (ABS (X)).
Het verschil tussen de functie FIX en INT is dat INT als resultaat bij een negatief argument altijd het eerstvolgende kleinere gehele getal heeft. (zie bij INT).

```
PRINT FIX (58.75)
 58
 Ok

PRINT FIX (-58.25), INT (-58.25)
-58                 -59
Ok
```

FOR ***variabele*** = ***begin*** **TO** ***eind: instructies*****: NEXT** ***variabele***
FOR ***variabele*** = ***begin*** **TO** ***eind*** **STEP** ***toename*****:** ***instructies*** **NEXT** ***variabele***

Een opgegeven aantal malen uitvoeren van een aantal instructies. ***begin, eind*** en ***toename*** zijn numerieke expressies, ***variabele*** wordt gebruikt als een teller. ***begin*** is de beginwaarde van de teller; ***eind*** is de eindwaarde.
variabele kan van het integer of enkele-precisie type zijn.
De instructies die volgen op de FOR-opdracht worden uitgevoerd tot de NEXT-opdracht is bereikt. Dan wordt de teller verhoogd met ***toename*** , de waarde achter STEP. Nu wordt gecontroleerd of de waarde van de teller groter is geworden dan ***eind.*** Is dit niet het geval, dan wordt teruggesprongen naar

de instructie, die volgt op de FOR-opdracht en herhaalt het hele gebeuren zich. Is de waarde van de teller groter dan ***eind***,dan gaat de programma-uitvoering verder bij de instructie, die volgt op de NEXT opdracht.
Indien **STEP** ***toename*** niet is aangegeven wordt de teller steeds verhoogd met de waarde 1.
Is de waarde van ***toename*** negatief, dan moet ***eind*** uiteraard kleiner zijn dan ***begin.*** De teller wordt dan na elke rondgang door de FOR...NEXT lus verlaagd, totdat de waarde van de teller kleiner is geworden dan de eindwaarde.
In alle gevallen worden de instructies tussen de FOR- en NEXT-opdracht minstens éénmaal uitgevoerd.
FOR...NEXT lussen mogen worden „genest", d.w.z. in elkaar vervat. In een FOR...NEXT lus mag dus een tweede FOR-NEXT lus voorkomen, binnen die lus weer een derde, enz. Het maximum aantal FOR...NEXT lussen is bepaald door de ruimte in het geheugen van de computer. Wél moeten in deze gevallen alle lussen een andere variabele-naam voor de teller gebruiken.
Wordt achter NEXT geen variabele-naam vermeld, dan gaat de BASIC interpreter ervan uit dat deze NEXT-opdracht hoort bij de laatst-uitgevoerde en nog niet door NEXT afgesloten FOR-opdracht.
Als de interpreter een NEXT-opdracht moet uitvoeren en is de bijbehorende FOR-opdracht nog niet gepasseerd, dan volgt een **Next without For** fout en wordt de programma-uitvoering beëindigd.

```
10 K=10
20 FOR I=1 TO K STEP 2
30 PRINT I;
40 K=K+10
50 PRINT K
60 NEXT
RUN
 1  20
 3  30
 5  40
 7  50
 9  60
 Ok
```

De waarde van K op het moment dat de FOR-opdracht voor het eerst gepasseerd wordt, is bepalend voor het aantal keren dat de lus wordt doorlopen.

```
10 J=0
20 FOR I=1 TO J
30 PRINT I
40 NEXT I
RUN
1
 Ok
```

De eindwaarde is weliswaar kleiner dan de beginwaarde, maar toch wordt de instructie in de lus éénmaal uitgevoerd. Pas bij de NEXT-opdracht immers wordt de waarde van de teller onderzocht.

```
10 I=5
20 FOR I=1 TO I+5
30 PRINT I;
40 NEXT
RUN
 1  2  3  4  5  6
 Ok
```

In dit voorbeeld wordt de programmalus 6 maal uitgevoerd. De eindwaarde van de teller wordt namelijk pas berekend nadat de beginwaarde is vastgesteld.

```
10 FOR I=1 TO 5
20 FOR J=I TO 5 * I STEP 10
30 PRINT I+J;
40 NEXT J,I
RUN
 2  3  4  6  16  8  18  10  20  30
 Ok
```

Let op! In regel 40 wordt eerst de J-lus beëindigd en daarna de I-lus.

FRE ***(getal)***
FRE ***(string)***

Functie. **FRE** ***(getal)*** geeft aantal geheugen plaatsen dat nog vrij is voor de opslag van het programma en variabelen. **FRE**

(string) geeft het aantal geheugenplaatsen dat nog vrij is voor de opslag van strings. ***getal*** en ***string*** hebben geen betekenis, alleen het type is van belang.

Na aanzetten of reset van een 16K P-2000.

```
PRINT FRE (0)
 14966
 Ok

PRINT FRE ("")
 50
 Ok
```

De aanroep van FRE ("") heeft bovendien tot gevolg, dat de stringruimte wordt „gereorganiseerd". Strings, die niet meer aan een string variabele zijn toegewezen worden door FRE ("") uit het geheugen verwijderd. Als aan een string-variabele meermalen een waarde wordt toegekend, blijft de „oude" string nl. in de stringruimte staan. Pas als de stringruimte vol is gelopen of na de opdracht FRE ("") worden de strings, die niet meer in gebruik zijn verwijderd en de andere strings aangeschoven.

```
10 LINE INPUT A$:
20 LINE INPUT A$:
RUN
aap
noot
 Ok
```

Heeft tot gevolg dat A$ gelijk is aan **noot**. De string **aap** blijft echter staan tot er ruimtegebrek komt of tot FRE ("") wordt uitgevoerd. Bij grote en volle stringruimte kan FRE ("") dan ook wel seconden duren, speciaal bij veel korte strings.

GOSUB ***regelnummer***
RETURN

Door de GOSUB *regelnummer* opdracht springt de interpreter direct naar dit regelnummer en vervolgt de programma-uitvoering daar. De opdracht RETURN veroorzaakt een terugkeer naar de eerste opdracht na GOSUB.

regelnummer is het nummer van de eerste programmaregel van de subroutine. Een subroutine mag een willekeurig aantal malen worden aangeroepen vanuit het hoofdprogramma. Ook mag een subroutine vanuit een andere subroutine worden aangeroepen. Dit „nesten" van subroutines wordt slechts beperkt door de beschikbare vrije geheugenruimte.
De RETURN-opdracht zorgt ervoor dat wordt teruggesprongen naar de instructie die volgt op de GOSUB-opdracht, die de sprong naar de subroutine heeft veroorzaakt. Het is toegestaan dat in één subroutine meer dan één RETURN-opdracht is opgenomen.
Als in de subroutine FOR-NEXT lussen geopend worden, dan worden deze door RETURN afgesloten.
Subroutines mogen op elke willekeurige plaats in het programma worden geplaatst. Om het ongewenst binnenlopen van een subroutine te voorkomen, doet U er goed aan om vóór de subroutine een STOP, END of GOTO opdracht op te nemen.
Indien ***regelnummer*** niet bestaat volgt een **Undefined line-number** fout.
Wordt, bijv. met een GOTO opdracht, naar een regel binnen een subroutine gesprongen dan volgt, bij het passeren van de RETURN-opdracht de fout **RETURN without GOSUB**.
Het verlaten van een subroutine via een GOTO, dus zonder de RETURN te passeren moet in het algemeen afgeraden worden. Het programma wordt onoverzichtelijk en een RETURN, te veel in het programma wordt gezien als de RETURN van de subroutine die al verlaten is.

```
10 GOSUB 100
20 PRINT"Terugkeer van subroutine"
30 END
100 REM subroutine begint hier
110 PRINT "Subroutine ";
120 PRINT "in ";
130 PRINT "uitvoering"
140 RETURN
RUN
Subroutine in uitvoering.
Terugkeer van subroutine.
 Ok
```

GOTO *regelnummer*

De programma-uitvoering in oplopende regelnummervolgorde wordt vervolgd na een sprong-opdracht naar ***regelnummer***. De opdracht op ***regelnummer***, en de daarop volgende opdrachten worden uitgevoerd. Indien ***regelnummer*** niet bestaat, geeft dit een **Undefined line number** fout.

```
5 R=1
10 R=2*R
20 PRINT"Straal=";R,
30 A=3.14 * R 2
40 PRINT"Oppervlak=";A
70 GOTO 10
RUN
Straal=  2   Oppervlak=   12.56
Straal=  4   Oppervlak=   50.24
Straal=  8   Oppervlak=  200.96
Straal=16   Oppervlak=  803.84
Etc.
```

In de directe stand kan men rechtstreeks naar een regel van een reeds in het geheugen staand programma springen. Het kan echter gebeuren, dat het programma hierdoor anders loopt, omdat aan bepaalde condities niet is voldaan.

HEX$*(getal)*

Functie: Levert een string waarvan de karakters de waarde van ***getal*** voorstellen in het zestien-tallig stelsel (hexadecimaal) ***getal*** wordt afgerond naar integer voordat de HEX$-functie wordt uitgevoerd, en moet liggen tussen -32768 en 65535.

```
10 INPUT X
20 A$ = HEX$(X)
30 PRINT X "decimaal is " A$ " hexadecimaal"
40 GOTO 10
RUN
? 32
 32 decimaal is 20 hexadecimaal
? 63
 63 decimaal is 3F hexadecimaal
 Ok
```

IF ***expressie*** **THEN** ***regelnummer***
IF ***expressie*** **THEN** ***instructies***
IF ***expressie*** **GOTO** ***regelnummer***
IF ***expressie*** **THEN** ***instructies*** **ELSE** ***instructies***
IF ***expressie*** **THEN** ***regelnummer*** **ELSE** ***regelnummer***

Afhankelijk van de waarde van ***expressie*** beslissen over de verdere loop van de programma-uitvoering.

Indien het resultaat van ***expressie*** "waar" is (ongelijk aan nul)* worden de opdrachten achter THEN uitgevoerd, t.m. de laatste opdracht voor ELSE (indien aanwezig).
Is het resultaat van ***expressie*** "niet waar" (gelijk aan nul) dan worden uitsluitend de opdrachten achter ELSE uitgevoerd. Is geen ELSE aanwezig, dan wordt verder gegaan met de volgende programmaregel.
Wordt direct na THEN of ELSE naar een regelnummer verwezen, dan mag GOTO worden weggelaten.
IF...THEN...ELSE constructies mogen worden genest, d.w.z. achter THEN en ELSE mag een volgende IF...THEN...ELSE constructie worden geplaatst.

IF X > Y THEN IF X > = 0 THEN PRINT "groter en positief" ELSE PRINT "groter en negatief" ELSE PRINT "kleiner"

Bevat een dergelijke programmaregel een ongelijk aantal THEN en ELSE dan gaat de interpreter ervan uit dat elke ELSE hoort bij de dichts bijzijnde THEN waarvan nog geen ELSE is uitgevoerd.

IF A = B THEN IF B = C THEN PRINT"A = C" ELSE PRINT"A < > C"

Bij uitvoering van deze opdrachten zal A < > C niet worden afgedrukt als A ongelijk is aan B.
Opmerking: IF....THEN ELSE werkt ook in de directe stand.
Wanneer U een expressie gebruikt om een waarde te onderzoeken die het resultaat is van enkele-of dubbele-precisie berekeningen, moet U er aan denken dat het resultaat van deze berekeningen niet helemaal exact is. De test moet daarom

* Bij het berekenen van ***expressie*** geldt de normale volgorde van prioriteiten van operaties; **eerst** rekenkundige operaties (in volgorde van prioriteit) **dan** relationele operaties en **dan** logische operaties. In de P2000 BASIC is "waar" gelijk aan -1.

worden uitgevoerd t.o.v. het bereik waarbinnen de nauwkeurigheid van die waarde mag varieren. Om bijvoorbeeld de variabele A te vergelijken met 1.0, gebruikt u: IF ABS(A-1.0) <1.0E-6 THEN
Deze expressie is alleen dan waar als A gelijk is aan 1.0 met een nauwkeurigheid van 1.0E-06.

```
10 FOR I=-1 TO 1
20 IF I THEN PRINT "I<>0" : GOTO40 : REM sprong naar 40 als I ongelijk 0
30 PRINT"      "I=";I
40 NEXT
50 END
RUN
I<>0
   I=0
I<>0
 Ok
```

```
10 FOR I= 10 TO 20
20 IF (I<18)*(I>12) THEN A$="GOED" : GOTO 50
30 PRINT I; "FOUT"
40 GOTO 60
50 PRINT I;"       ";A$
60 NEXT
70 END
RUN
 10 FOUT
 11 FOUT
 12 FOUT
 13       GOED
 14       GOED
 15       GOED
 16       GOED
 17       GOED
 18 FOUT
 19 FOUT
 20 FOUT
 Ok
```

1) IF A=45 THEN A=46 → A=A-(A=45)

2) IF A>0 AND A<5 THEN B=1 ELSE B=2

B=2+(A>0 AND A<5)

INP ***(getal)***

Functie. Leest de 8-bit waarde af die wordt aangeboden op ingangspoort nummer ***getal***. ***getal*** moet in het bereik 0..255 liggen en wordt m.b.v. CINT naar integer geconverteerd. Vgl. ook de OUT-instructie, die een 8-bit waarde naar een uitgangspoort zendt.

```
100 A=INP (32)
```

INP("")

Een bijzondere uitvoering van de INP-functie is INP("") of INP *(string)*. Deze geeft namelijk de ASCII-waarde van een ingedrukte toets op het toetsenbord.
De programmaregel:

```
10,A=INP("")
```

heeft tot gevolg, dat de P2000 wacht op het indrukken van een toets. Daarna wordt aan A de ASCII-waarde van de ingedrukte toets toegekend (zie ook Hoofdstuk 4. Toetsenbord).

INPUT ***lijst van variabelen***
INPUT ***"hulpstring"; lijst van variabelen***

Invoer van gegevens via het toetsenbord gedurende de uitvoering van het programma.

INPUT drukt altijd een vraagteken af. Indien achter de INPUT instructie een ***hulpstring*** is meegegeven, wordt deze vóór het vraagteken afgedrukt. Daarna wacht het programma tot de gegevens zijn ingevoerd.

```
10 INPUT "Geef aantal", A%
20 PRINT A%
RUN
Geef aantal? 2
 2
 Ok
```

Worden meer gegevens gevraagd dan kunnen die achter elkaar, gescheiden door komma's, worden ingetikt.

```
10 INPUT A%, B$
? 2.5, getal
 Ok
```

Gebeurt dit niet, dan wordt na het intikken van het eerste gegeven afgedrukt,
??

```
10 INPUT A%, B$
RUN
? 5
?? getal
```

(**5** en **getal** zijn hier door de gebruiker ingetikt als antwoorden op **?** en **??**).
hulpstring mag geen string expressie of string variabele zijn.

De variabelen mogen van elk type zijn te weten integer, enkele- en dubbele-precisie, string en ook array elementen. Stemt het type van de ingevoerde constante niet overeen met het type van de corresponderende variabele en is geen type conversie mogelijk dan volgt een foutmelding **Fout, opnieuw** en mag opnieuw een waarde worden ingetikt.
INPUT kan niet in de directe stand worden gebruikt.
De invoer van gegevens moet met een ENTER worden afgesloten. Wordt alleen een ENTER gegeven, dan krijgt de variabele niet de waarde nul, maar behoudt de waarde, die de variabele op dat moment had.
Als de cursor in het programma is uitgezet, wordt deze bij het uitvoeren van INPUT weer aangezet. De cursor toetsen kunnen worden gebruikt tijdens het invoeren van gegevens.
Voordat ENTER wordt gegeven kan het ingetikte gegeven worden gewijzigd, zoals in de invoeg-stand van de EDIT-stand.
Komen in ingevoerde gegevens één of meer komma's voor dan wordt het gegeven als meerdere gegevens opgevat. Is het aantal gegevens hierdoor groter dan het gevraagde aantal, dan verschijnt de foutmelding **Komma teveel**, de overtollige invoer wordt genegeerd en het programma gaat verder.
Voor strings is de instructie LINE INPUT aan te bevelen, of de ingevoerde strings moeten tussen aanhalingstekens worden geplaatst.

INSTR*(string 1, string2)*
INSTR*(getal, string 1,string 2)*

Functie. Geeft de positie in ***string1*** waar ***string2***, voor de eerste maal, begint. Is de waarde van ***getal*** groter dan 1, dan begint de zoekprocedure op positie ***getal*** in ***string1***; ***getal*** moet binnen het bereik 0...255 liggen. Is ***getal*** groter dan het aantal karakters van ***string1***, heeft ***string1*** een lengte nul of komt ***string2*** niet voor in ***string1*** dan is het resultaat van de INSTR-functie gelijk aan nul. Heeft ***string2*** de lengte 0, dan is het resultaat van INSTR gelijk aan ***getal***, of als ***getal*** niet is opgegeven aan 1.

```
10 X$ = "ABCDCBA"
20 Y$ = "B"
30 PRINT INSTR(X$,Y$); INSTR (4,X$,Y$)
RUN
 2     6
 Ok
```

INT *(getal)*

Functie. Levert de grootste integer, kleiner dan of gelijk aan ***getal***.

```
PRINT INT (99.89)
 99
 Ok

PRINT INT (-12.11)
-13
 Ok
```

Vergelijk met de functies FIX en CINT, die ook als resultaat een integer waarde afleveren.

X=	-1.8	-1.2	1.2	1.8
INT(X)	-2	-2	1	1
FIX(X)	-1	-1	1	1
CINT(X)	-2	-1	1	2

LEFT$ ***(string, getal)***

Functie, die een string aflevert bestaande uit de linker ***getal*** karakters van de string ***string. getal*** moet binnen het bereik 0...255 liggen. Is ***getal*** groter dan het aantal karakters van ***string*** dan is het resultaat van de LEFT$-functie de hele string. Is ***getal***=0, dan is het resultaat een lege string, is ***getal*** negatief of groter dan 255 dan volgt een **Illegal function call** fout.

```
10 A$ = "BASIC-NL"
20 B$ = LEFT$(X$,5)
30 PRINT B$
40 B$ = LEFT$(X$,100)
50 PRINT B$
RUN
BASIC
BASIC-NL
 Ok
```

LEN ***(string)***

Functie. Geeft het aantal karakters waaruit de string ***string*** bestaat. (Ook niet printbare karakters en spaties worden meegeteld).

```
10 X$ = "PHILIPS P2000"
20 PRINT LEN (X$)
RUN
 13
 Ok
```

LET ***variabele*** = ***getal*** of ***string***

Het toekennen van de waarde van een expressie aan een variabele.
Het woord LET mag worden weggelaten.

```
110 LET D = 12
120 LET E = 5*3
130 LET F = 345 - 67
140 LET SUM = D + E + F
150 LET A$ = CHR$(12)
```

òf

```
110 D = 12
120 E = 5*3
130 F = 345 - 67
140 SUM = D + E + F
150 A$ = CHR$(12)
```

LINE INPUT ***string variabele***
LINE INPUT ***"hulpstring"; string variabele***

Het invoeren via het toetsenbord van maximaal 254 karakters, inclusief komma's, aanhalingstekens enz., en het toekennen van deze tekst aan een stringvariabele.

hulpstring mag geen string expressie of variabele zijn. Op het beeldscherm wordt ***hulpstring*** weergegeven, voordat tekstinvoer wordt geaccepteerd. Er wordt niet, zoals bij INPUT, een vraagteken afgedrukt.

De invoer van tekst kan worden afgebroken door op de STOP-toets te drukken. De interpreter komt dan in de directe stand. Als de STOP-toets afgevangen is met ON ERROR GOTO wordt de voortzetting van het programma uiteraard geheel door de foutafhandelingsroutine bepaald. De opdracht CONT start de LINE INPUT opdracht opnieuw en drukt ***hulpstring*** opnieuw af. Wordt alleen de ENTER toets ingedrukt, dan wordt aan ***stringvariabele*** een lege string toegekend; dit in tegenstelling tot INPUT, waar de laatste waarde van de variabele behouden blijft. LINE INPUT kan niet in de directe stand worden gebruikt. Tijdens de invoer kunnen de cursor toetsen gebruikt worden om de tekst te wijzigen, als bij EDIT in de Invoeg stand.

```
10 LINE INPUT "geef naam"; NA$
```

LIST
LIST regelnummer
LIST regelnummer 1 - regelnummer 2
LIST regelnummer-
LIST - regelnummer
LIST.

LIST. - regelnummer
LIST regelnummer-.
LIST.-
LIST-.

Op het beeldscherm afdrukken van een deel van de tekst van het programma dat zich op dat moment in het geheugen van de computer bevindt.
Na uitvoering van de LIST-opdracht keert de interpreter altijd terug naar de directe stand. De weergave (LIST-ing) van het programma op het scherm kan worden tegengehouden door de STOP-toets in te drukken; nogmaals SHIFT-STOP beëindigt het LIST-en, elke andere toets laat het LIST-en verder gaan.
De volgende varianten op de LIST opdracht zijn toegestaan:
LIST
of
LIST -
toont alle in het geheugen aanwezige programmaregels op het scherm.
LIST *regelnummer*
toont alleen de programmaregel met het aangegeven regelnummer.
LIST *regelnummer* -
toont de in het geheugen aanwezige programmaregels, te beginnen bij het aangegeven regelnummer tot het einde van het programma.
LIST - *regelnummer*
toont alle in het geheugen aanwezige programmaregels, vanaf het begin tot en met het aangegeven ***regelnummer***.
LIST *regelnummer1* - *regelnummer2*
toont de programmaregels vanaf ***regelnummer 1*** tot en met ***regelnummer 2.***
LIST.
toont huidige regel (regel, die het laatste ge-LIST of ge-EDIT is).

"Rustige" listing
Om een programma met kleine stukken tegelijk te LIST-en kent de P2000 BASIC een "rustige LIST". Hierbij wordt gebruik gemaakt van de „deel-PRINT-routine". Deze wordt ingeschakeld door eerst de M (SHIFT 9)-toets op het kleine toetsenbord in te drukken. Tikt men daarna één van de LIST-

opdrachten (bijv. LIST1000-), dan verschijnt het programma op het scherm. Echter, als het scherm vol is loopt de LIST-ing niet verder, waardoor het boven gedeelte van het scherm zou verdwijnen, maar het LIST-en stopt. Als de LIST-ing nog niet voltooid is verschijnt het woord "meer" in de linker onderhoek van het scherm. Indrukken van de ENTER-toets toont het volgende scherm vol van de LIST-ing. De wisregel-toets wist eerst het scherm en geeft daarna het volgende deel van de LIST-ing. Met de printer-toets (SHIFT 00) kan men met behulp van een printer een afdruk op papier maken van het getoonde deel van het programma. Elke andere toetsindruk voegt eéń schermregel toe, waarbij een schermregel aan de bovenkant van het beeldscherm verdwijnt.
De "deel-PRINT-routine" kan worden uitgeschakeld door het indrukken van de CODE-toets, op het moment dat het woord "meer" te zien is.
Het indrukken van de SHIFT-5-toets op het kleine toetsenbord heeft tot gevolg dat de rustige LIST-ing wordt ingeschakeld en dat het programma ge-LIST wordt vanaf het begin. De toets verenigt dus de wis scherm-toets de M-toets en de opdracht LIST
De rustige LIST-ing wordt uitgeschakeld bij RUN, maar blijft actief bij RUN *regelnummer*. Dit kan tot gevolg hebben dat tijdens de uitvoering van het programma het afdrukken van gegevens op het scherm ongewild wordt onderbroken en dat linksonder het woord "meer" verschijnt.

LLIST

Het op de printer afdrukken van het programma, of een deel van het programma dat zich op dat moment in het geheugen bevindt.
Na uitvoering van de LLIST opdracht keert de interpreter automatisch terug naar de directe stand. De varianten op de LLIST opdracht zijn dezelfde als die op LIST. Voor instellen van de printsnelheid, de regelbreedte e.d. van de printer zie hoofdstuk 14: de printer.

LOG *(getal)*

Functie. Levert de waarde van de natuurlijke logarithme van ***getal***. Het grondtal van deze logarithme is e; ***getal*** moet groter zijn dan nul, anders volgt een **Illegal function call**.

```
PRINT LOG(45/7)
 1.86075
 Ok
```

LOG(X)/LOG(10) geeft de logarithme op basis van grondtal 10.

LPOS ***(getal)***

Functie: Geeft de positie op de printerregel van het volgende karakter dat afgedrukt wordt. Vergelijk ook POS.

```
100 IF LPOS (0) = 60 THEN LPRINT CHR$(13);
```

LPRINT

Op de printer afdrukken van getallen en teksten.

Zie PRINT en hoofdstuk 14.

MID$ ***(string,getal1,getal2)***
MID$ ***(string,getal)***

MID$ kan op 2 manieren gebruikt worden:

1. Als functie om een deel van ***string*** eruit te lichten.
2. Als opdracht om een deel van ***string*** door een andere tekst te vervangen.

1. Vanaf het ***getal1***-de karakter van de ***string*** worden ***getal2*** karakters eruit gelicht. Is ***getal2*** niet opgegeven dan worden alle karakters vanaf het ***getal1*** overgenomen.
2. Vanaf het ***getal1***-de karakter van ***string*** worden ***getal2*** karakters vervangen. ***string*** mag hierdoor niet van lengte veranderen, m.a.w. de vervangende string moet een gelijk aantal karakters hebben als het deel dat vervangen wordt. Is dit niet het geval,dan worden zoveel karakters vervangen als mogelijk is. Ook hier geldt, dat als ***getal2*** niet wordt opgegeven de vervanging t.m. het laatste karakter van ***string*** plaats vindt. Telling van de karakters in de string begint bij 1.

1
```
10 A$ = MID$("Kees programmeert", 6,7)
20 PRINT A$
RUN
program
 Ok
```

2.
```
10 A$ = "Laren (NB)"
20 MID$(A$,8) = "NH"
```

```
30 PRINT A$
RUN
Laren (NH)
 Ok
```

Opmerking: met de eerste MID$-instructie kunnen delen van strings met elkaar worden vergeleken.
IF MID$ (A$,2,3) = MID$ (B$,12,3) THEN.....

NEW

Wissen van het BASIC programma dat zich op dat moment in het geheugen van de P2000 bevindt, alsmede het verwijderen van alle variabelen.

De NEW opdracht wordt meestal gegeven in de directe stand. Als NEW in een programmaregel voorkomt wist het programma zichzelf. NEW tast de met CLEAR gemaakte reserveringen niet aan. Eventuele aanwezige machinetaal routines blijven aanwezig en oproepbaar. Hierdoor kan tijdens het laden of RUN-nen van een volgend programma **Out of memory** optreden. NEW wordt automatisch uitgevoerd bij CLOAD.

OCT$ ***(getal)***

Functie. Geeft een string waarvan de karakters ***getal*** weergeven in het achttallig stelsel (octaal). Voordat de functie OCT$ wordt uitgevoerd, wordt ***getal*** omgezet naar integer.

```
PRINT OCT$(24)
30
 Ok
```

ON ERROR GOTO ***regelnummer***

Het opvangen van fouten tijdens de programmauitvoering en het aangeven van het regelnummer van de eerste programmaregel van de foutafhandelingsroutine. De foutopvanging wordt uitgeschakeld m.b.v. de instructie ON ERROR GOTO 0 en door het uitvoeren van CLEAR, NEW en CLOAD.
Als ***regelnummer*** niet bestaat, volgt een **Undefined line number** fout.

Wanneer de foutafhandeling in werking is na het uitvoeren van de ON ERROR GOTO opdracht, zal bij elke fout die optreedt (ook fouten in de directe stand) een sprong volgen naar het in

ON ERROR GOTO aangegeven ***regelnummer,*** waar de foutafhandelingsroutine begint.
Fouten die na uitvoering van ON ERROR GOTO 0 optreden, veroorzaken de "normale" foutmeldingen op het scherm van de interpreter en doen de uitvoering van het programma stoppen. Komt in de foutafhandelingsroutine zèlf een ON ERROR GOTO 0 instructie voor, dan stopt het programma eveneens en wordt voor de meeste fouten de "normale" foutmelding gegeven. De foutafhande vlingsroutine wordt beëindigd door de opdracht RESUME.
ON ERROR GOTO heeft alleen zin als in het programma geen ***Syntax error*** meer voorkomt. In een programma kunnen meer ON ERROR GOTO opdrachten voorkomen. Alleen de laatstuitgevoerde telt.

```
10 ON ERROR GOTO 100
20 INPUT A$
 -
 -
 -
100 IF ERR=64 AND ERL=20 THEN RESUME ELSE ON
ERROR GOTO 0
```

Het indrukken van de SHIFT-STOP-toets geeft foutnummer 64 (ERR=64).
Als bij het wachten op INPUT in regel 20 de STOP-toets wordt ingedrukt laat de foutafhandelings routine via RESUME terugspringen naar regel 20 en kan het programma hier niet met de STOP-toets worden beëindigd.
Elders in het programma heeft het indrukken van de STOP-toets tot gevolg, dat de uitvoering van het programma wel wordt beeindigd. (Zie ook onder ERROR).

Het afvangen van de STOP-toets met ON ERROR GOTO kan soms problemen opleveren als de STOP-toets wordt ingedrukt als het programma een FOR...NEXT lus aan het uitvoeren is of bij een GOSUB. De informatie over de laatst uitgevoerde opdracht kan verloren gaan. In deze gevallen kan dus niet met RESUME direct in het programma worden teruggesprongen, maar moet de foutafhandelingsroutine andere acties ondernemen.

```
10 ON ERROR GOTO 1000
100 A=B/C
110..
150

1000 IF ERR=11 THEN A=1E6: RESUME NEXT
1010 IF ERR=6 THEN A=1E5: RESUME 150
```

Als getracht wordt door 0 te delen (bijv. in regel 100) wordt A gelijk gemaakt aan 10^6 en het programma wordt vervolgd met de volgende regel (bijv. regel 110). Wordt A groter dan de maximaal mogelijke waarde, (maar niet doordat door 0 gedeeld wordt) dan wordt A gelijk gemaakt aan 10^5 en het programma vervolgt op regel 150.

ON *getal* GOTO *lijst van regelnummers*
ON *getal* GOSUB *lijst van regelnummers*

Uitvoeren van een sprong naar één van de aangegeven regelnummers, afhankelijk van de waarde van ***getal.*** De regelnummers in de lijst worden gescheiden door een komma.

De waarde van ***getal*** bepaalt welke van de in de lijst aangegeven regelnummers voor de sprong zal worden gebruikt.
Is die waarde 1, dan wordt het eerste regelnummer uit de lijst gebruikt; is de waarde 2, dan wordt het tweede regelnummer gebruikt, enz.
Is de waarde van ***getal*** niet geheel, dan vindt eerst afronding plaats.
Bestaat een regelnummer niet, dan volgt een **Undefined iine number** fout.
Indien de waarde van ***getal*** gelijk is aan nul, of groter dan het aantal in de lijst vermelde regelnummers, gaat de programma-uitvoering verder met de eerstvolgende opdracht in het programma. Is de waarde van ***getal*** echter kleiner dan nul of groter dan 255, dan volgt een **Illegal function call** fout.

```
ON L GOTO 150, 300, 380, 390
```

Bij L=1 volgt een sprong naar regel 150, is L=2 dan wordt naar regel 300 gesprongen enz.

Deze instructie vervangt in feite de volgende opdrachten

IF L = 1 THEN 150
IF L = 2 THEN 300
IF L = 3 THEN 380
IF L = 4 THEN 390

10 ON SGN(A) + 2 GOSUB 100, 200, 300
Als A < 0 dan wordt subroutine 100 uitgevoerd, is A = 0 dan subroutine 200 en als A > 0 dan subroutine 300.

OUT *getal1, getal2*
Zenden (schrijven) van een byte naar een uitgangspoort (output port) van de P2000.
getal1 en ***getal2*** worden omgezet naar integers in het bereik 0...255.
getal1 is het nummer van de uitgangspoort; ***getal2*** is het getal dat naar de poort wordt gezonden.

10 OUT 32, 100
De waarde 100 wordt naar uitgangspoort 32 gezonden.

Een veel voorkomende OUT-instructie is OUT 48,J. Deze verschuift het beeldscherm zodanig, dat de J-de kolom de eerst-zichtbare is. OUT 48,40 toont het „rechterdeel" van het beeldscherm. OUT 48,0 zet het „linkerdeel" van het beeldscherm voor.
In de directe stand wordt altijd het „linkerdeel" van het scherm getoond.

PEEK *(adres)*
Levert inhoud van de geheugenplaats ***adres***.
adres moet in het bereik -32768...65535 liggen.

100 A = PEEK (&H5000): REM A = inhoud hexadecimaal adres 5000

PRINT PEEK(532): REM Inhoud geheugenplaats 532.

PEEK("")

Een bijzondere uitvoering van de PEEK instructie is: PEEK("").
Met deze opdracht kan gekeken worden of een toets op het toetsenbord is ingedrukt. PEEK("") geeft de waarde 0 af als geen toets was ingedrukt. Na indrukken van een toets geeft PEEK("") de waarde 1.
PEEK("") wordt weer op 0 gezet door A=INP("") of door POKE &H600C,0

10 IF PEEK("") = 0 THEN 10

POKE *adres, getal*

Schrijven van ***getal*** in geheugenplaats ***adres***

De waarde van ***adres*** mag varieren van -32768 tot en met 65535; echter, alleen de waarden die overeen komen met een RAM geheugenplaats zijn zinvol. De waarde van ***getal*** mag van 0 tot en met 255 zijn. Is hieraan niet voldaan dan volgt een **Illegal function call** fout. Eventueel vindt eerst een conversie naar integer getallen plaats, zowel van ***getal*** als van ***adres***.

POKE en PEEK zijn nuttig voor efficiente gegevensopslag, het laden van machinetaal subroutines, het doorgeven van parameters van en naar zulke subroutines etc.

10 POKE &H5000, &HFF zet getal 255 (hexadecimaal &HFF) op geheugenplaats &H5000

10 POKE &H5000,65
10 POKE &H5000,PEEK(&H5000)+128 leest met PEEK (&H5000) de waarde van geheugenplaats &H5000, telt er 128 bij op en schrijft het resultaat weer terug.

Waarschuwing: POKE is een gevaarlijke instructie. Zonder enige controle door het systeem wordt de inhoud van een geheugenplaats gewijzigd. De geringste fout (verkeerd adres, verkeerde waarde, verkeerde plaats in het programma) kan desastreuse gevolgen hebben voor het programma of de resultaten van de uitvoering.

POS ***(getal)***

De functie POS geeft als resultaat het aantal karakters dat door de laatste PRINT-opdrachten zonder ; is afgedrukt. Zolang niet gebruik gemaakt is van CHR$ cursor besturing is de kolom van de cursor (POS(0)-1))/breedte +1
getal is een "dummy argument".

IF POS (X) > 30 THEN PRINT

PRINT ***lijst van expressies***

Op het beeldscherm tonen van getallen en teksten.

Indien ***lijst van expressie*** wordt weggelaten, d.w.z. de opdracht bestaat alleen uit het woord PRINT, wordt de cursor op het begin van de volgende regel geplaatst. Eventueel schuift de inhoud van het scherm een regel naar boven. Zijn achter PRINT één of meer ***expressies*** aangegeven, dan worden de waarden van die expressies afgedrukt. In ***lijst van expressies*** mogen zowel numerieke als string expressies en variabelen worden opgenomen.

De positie op het beeldscherm waar de waarde van de expressie wordt weergegeven, hangt af van de wijze waarop de expressies achter PRINT van elkaar worden gescheiden. Gebeurt dit d.m.v. een komma, dan wordt de volgende waarde afgedrukt aan het begin van het volgende veld van 14 posities. Op het normale beeldschermvenster van 40 karakters breed is ruimte voor 2 van deze velden.
Wordt als scheidingsteken een punt-komma (;) gebruikt dan wordt een volgende waarde direct achter de vorige weergegeven. Als geen verwarring mogelijk is, mag de ; worden weggelaten, bijv.
PRINT "sinus" SIN(X) "cosinus" COS(X).
Wordt achter de laatste expressie in de lijst ook een komma of een punt-komma geplaatst, dan worden de waarden van de expressies in een volgende PRINT-opdracht, indien mogelijk, op dezelfde regel weergegeven. Staat achter de laatste expressie geen komma of punt-komma staat, dan wordt na uitvoering van de PRINT-opdracht de cursor op het begin van de volgende regel geplaatst.

Indien de af te drukken waarden meer ruimte innemen dan de breedte van het venster toelaat, wordt, wanneer de regel vol is, verder gegaan op de volgende regel.
Achter alle afgedrukte getallen wordt een spatie afgedrukt. Bovendien wordt een positief getal voorafgegaan door een spatie; een negatief getal wordt voorafgegaan door een minteken (-) in plaats van een spatie.
Enkele-precisie getallen die in de niet-exponentiele vorm kunnen worden weergegeven m.b.v. 6 of minder cijfers worden in die vorm afgedrukt. Bijvoorbeeld: 1E-6 wordt weergegeven als .000001 terwijl 1E-7 wordt weergegeven als 1E-7. Voor dubbele-precisie getallen geldt in feite hetzelfde, maar er worden 16 of minder cijfers weergegeven, bijv.

1D-16 wordt weergegeven als .0000000000000001
terwijl
1D-17 wordt weergegeven als 1D-17.
In plaats van het woord PRINT mag bij het intypen ook een vraagteken (?) worden gebruikt.

```
10 X=5
20 PRINT X+5, X-5, X*(-5), X 5
30 END
RUN
 10            0
-25           3155
 Ok
```

In bovenstaand voorbeeld zorgen komma's tussen de expressies ervoor dat alle getallen op dezelfde regel worden afgedrukt, steeds beginnend in het volgende veld.

```
10 INPUT "getal";x
20 PRINT X "in het kwadraat is" X 2 "en";
30 PRINT X;"tot de macht 3 is" X 3
40 PRINT
50 GOTO 10
RUN
getal? 9
 9 in het kwadraat is 81 en 9 tot de macht 3 is 729

getal? 21
 21 in het kwadraat is 441 en 21 tot de macht 3 is 9261
```

In dit voorbeeld zorgt de punt-komma aan het eind van regel 20 ervoor dat de waarden van de PRINT-opdracht in regel 30 achter de waarden van de PRINT-opdracht in regel 20 worden weergegeven. Regel 40 zorgt ervoor dat steeds een regel wordt overgeslagen voordat een nieuwe waarde voor X wordt binnengehaald.

```
10 FOR X=1 TO 5
20 J=J+5
30 K=K+10
40 ?J;K;
50 NEXT X
RUN
 5   10   10   20   15   30   20   40   25   50
 30   60   35   70   40   80   45   30
 Ok
```

De punt-komma's in regel 40 zorgen ervoor dat alle waarden direct achter elkaar worden weergegeven. Achter elk afgedrukt getal wordt een spatie afgedrukt en positieve getallen worden voorafgegaan door een spatie; hier staat bij negatieve getallen het min-teken.
In regel 40 wordt een vraagteken gebruikt in plaats van het woord PRINT. Als het programma wordt ge-LIST blijkt de interpreter het vraagteken door het woord PRINT te hebben vervangen.
Voor meer informatie over de mogelijkheden van de PRINT-instructie wordt verwezen naar hoofdstuk 6

READ ***lijst van variabelen***

Het lezen van de waarden in DATA-lijsten en het toekennen ervan aan variabelen.

Een READ-opdracht moet altijd worden gebruikt in combinatie met één of meer DATA-lijsten. De constanten (numerieke of string waarden), vermeld in de DATA-lijsten, worden achtereenvolgens toegekend aan de variabelen in de READ-opdracht.
Het type variabele (numeriek of string) in de READ-opdracht moet uiteraard overeenstemmen met het type (numeriek of string) van de constante in de DATA-lijst. Is dit niet het geval dan volgt een **Syntax error** met als regelnummer de DATA-regel. Numerieke data worden, zo nodig, als string opgevat.

Een enkele READ-opdracht mag waarden betrekken uit meerdere DATA-regels; ook is het mogelijk dat meerdere READ-opdrachten waarden betrekken uit slechts eéń DATA-regel. Overschrijdt het aantal variabelen in de READ-opdrachten het aantal DATA-waarden, dan volgt een **Out of DATA** fout. Is het aantal variabelen in de READ-opdracht kleiner dan het aantal DATA-waarden, dan zal een volgende READ-opdracht de eerste, nog niet gelezen DATA-waarde lezen.
Om met het lezen van de waarden uit een DATA-regel weer bij de eerste waarde te beginnen, moet de RESTORE-opdracht worden gebruikt. Na CLEAR en RUN gebeurt dit automatisch.

```
80 FOR I=1 TO 10
90 READ A(I)
100 NEXT I

1000 DATA 3.08, 5.19, 3.12, 3.98, 4.24
1010 DATA 5.08, 5.55, 4.00, 3.16, 3.37
```

De waarden die in de DATA-regels 1000 en 1010 zijn vermeld worden achtereenvolgens ingelezen in het array A(I).
Na uitvoering van programmaregels 80 tot 100 is de waarde A(1)=3.08, A(2)=5.19, enz.

```
30 READ PL$,PR$,V
30 DATA "Eindhoven, ",Noord Brabant,34784
40 PRINT V: PRINT PL$ PR$
RUN
 34784
Eindhoven, Noord Brabant
 Ok
```

Dit programma leest de strings en de numerieke waarde in regel 30 en drukt ze af. De string "Eindhoven," is tussen aanhalingstekens geplaatst, omdat de komma achter Eindhoven bij de string hoort en niet als scheidingsteken met de volgende DATA-waarde moet worden opgevat.

REM ***toelichting***

Plaatsen van toelichting bij het programma.

Het gedeelte van de programma-regel na een REM instructie wordt door de interpreter niet uitgevoerd, wel wordt het bij het afdrukken (LIST-en) van het programma weergegeven.

Vanuit een GOTO of GOSUB opdracht mag direct naar een REM regel worden gesprongen; de werkelijke uitvoering van het programma wordt dan voortgezet op de eerstvolgende programmaregel. Een REM-instructie kan niet worden afgesloten met een dubbele punt (:).
In plaats van REM mag ook een accent '(SHIFT-7) worden gebruikt.

```
10 REM Bereken gemiddelde versnelling
20 FOR I=1 TO 20
30 SOM=SOM+V(I)
40 NEXT I

20 FOR I=1 TO 20:REM Bereken gemiddelde versnelling
30 SOM=SOM+V(I)
40 NEXT I
50 GOSUB 1000
60 REM Afgelegde weg
1000 REM Bereken snelheid
 .
 .
1150 RETURN
```

RESTORE
RESTORE ***regelnummer***

Opnieuw lezen van de waarden in DATA-regels.

Nadat de RESTORE-opdracht zonder ***regelnummer*** is uitgevoerd, zal de eerstvolgende READ-opdracht de eerste waarde in de eerste DATA-regel lezen.
Is ***regelnummer*** aangegeven, dan zal de eerstvolgende READ-opdracht de eerste waarde lezen van de DATA-regel met ***regelnummer*** of, als dit geen DATA-regel is, de eerst volgende DATA-regel.

```
10 READ A,B,C
20 PRINT A,B,C
30 RESTORE
40 READ D,E,F
50 PRINT D;E;F
60 DATA 57,68,79
```

```
RUN
 57              68
 79
 57   68   79
 Ok

10 RESTORE 1000
20 READ A,B,C
30 PRINT A,B,C
40 IFA*B >100 THEN RESTORE 4000 ELSE RESTORE 2000
50 GOTO 20
1000 DATA 1,2,3
2000 DATA 10,20,30
4000 DATA 1000,2000,3000
RUN
 1               2
 3               10
 20              30
 1000            2000
 3000            1000
 2000            3000
```

RESUME
RESUME 0
RESUME NEXT
RESUME ***regelnummer***

Voortzetten van de programma-uitvoering, wanneer een fout is opgetreden die door een ON ERROR GOTO *regelnummer* is opgevangen en nadat de foutafhandelingsroutine is uitgevoerd.

Welke van de vier uitvoeringsvormen wordt gebruikt is afhankelijk van de plaats waar het programma moet worden voortgezet:

RESUME Programma-uitvoering wordt voortgezet bij de
RESUME 0 opdracht waarin de fout voorkwam.

RESUME NEXT Programma-uitvoering wordt voortgezet bij de opdracht, volgend op die waarin de fout voorkwam.

RESUME ***regelnummer***

Programma-uitvoering wordt voortgezet bij de eerste opdracht op het aangegeven regelnummer.

Een RESUME-opdracht veroorzaakt een **Resume without Error in *regelnummer*** foutmelding, wanneer RESUME moet worden uitgevoerd zonder dat een fout was opgetreden. Zie ook ON ERROR GOTO.

```
10 ON ERROR GOTO 900
 .
 .
900 PRINT"TOETS ANDERE WAARDE IN":RESUME 80
```

RIGHT$ *(string,getal)*

Functie. Levert een string bestaande uit de rechter ***getal*** karakters van de string ***string***.

Is ***getal*** groter dan of gelijk aan het aantal karakters van ***string,*** dan is het resultaat van de RIGHT$-functie de hele string. Is ***getal*** $=0$ dan is het resultaat een lege string. Bij negatieve waarden van ***getal*** of waarden groter dan 255 volgt een **Illegal Function Call**.

```
10 A$ = "BASIC-NL"
20 PRINT RIGHT$(A$,2)
30 PRINT RIGHT$(A$,100)
RUN
NL
BASIC-NL
 Ok
```

RND *(getal)*

Functie. Levert een willekeurige (random) waarde tussen 0 en 1.
Is ***getal*** <0, dan wordt bij herhaald uitvoeren van de RND-functie steeds dezelfde waarde gegenereerd.
Is ***getal*** >0 dan wordt de volgende waarde uit de reeks gegenereerd.
Is ***getal*** $=0$ dan wordt de laatst gegenereerde waarde herhaald.

```
10 X=1
20 FOR I=1 TO 5
30 PRINT INT (RND (X) * 100);
40 NEXT I
```

```
RUN
 49   67   98   73   78
 Ok

10 X=0
20 FOR I=1 TO 5
30 PRINT INT (RND (X) * 100);
40 NEXT I
RUN
 30   30   30   30   30
 Ok
```
(Steeds weer na iedere run).

RUN ***"programmanaam"***
RUN ***string***
RUN"
RUN
RUN ***regelnummer***

Wissen van alle variabelen en starten van het programma.

Deze RUN-opdrachten kunnen zowel vanuit het programma worden gegeven als vanuit de directe stand.
RUN ***"programmanaam"*** zoekt op de band een programma op waarvan de eerste letter gelijk is aan die van ***programmanaam***, laadt dit in het geheugen en neemt het in uitvoering.
RUN ***string*** doet hetzelfde, met dit verschil, dat ***string*** apart kan worden opgegeven of berekend.
RUN" of **RUN""** zoekt het eerste programma op de band op, laadt het en neemt het in uitvoering. Als op het begin van de band een array staat, wordt dit door de P2000 niet als zodanig herkend. Het array wordt als programma ingeladen en in uitvoering genomen, hetgeen in het algemeen snel tot een foutmelding leidt.
RUN start het programma dat op dat moment in het geheugen staat.
RUN ***regelnummer*** start het programma in het geheugen te beginnen bij het aangegeven regelnummer.

In plaats van met RUN kan men vanuit de directe stand het programma ook starten door het indrukken van de START-toets.

```
10 IF I=0 THEN RUN"sorteer" ELSE RUN"scramble"

10 IF I=0 THEN A$="sorteer" ELSE A$="test"

20 RUN A$

10 RUN CHR$(64+I)

RUN 200

100 IF K=3 THEN RUN"
```

Opmerking: Bij het inschakelen van de P2000 óf na het indrukken van de RESET-toets wordt, als een cassette in de recorder is geplaatst, automatisch een RUN" opdracht gegeven.

SGN ***(getal)***

Functie. Geeft het teken van ***getal***.
Als X < 0 dan is SGN (X) = -1
X = dan is 0 SGN (X) = 0
X > dan is 0 SGN (X) = 1

SIN***(getal)***

Functie, geeft de sinus van de hoek ***getal,*** met ***getal*** uitgedrukt in radialen.

```
PRINT SIN (1.5)
 .997495
 Ok
```

SPACE$ ***(getal)***

Functie. Levert een string bestaande uit ***getal*** spaties. ***getal*** wordt eerst afgerond tot een integer waarde en moet dan in het bereik 0...255 liggen.

```
10 FOR I=1 TO 5
20 X$ = SPACES (T)
30 PRINT X$; I
40 NEXT
```

```
RUN
 1
  2
   3
    4
     5
 Ok
```

Zie ook de functie SPC en de functie STRING$(I,J).

SPC*(getal)*

Geeft ***getal*** spaties weer op het beeldscherm of de printer. De SPC-functie mag alleen worden gebruikt in een PRINT of LPRINT-instructie.
getal moet in het bereik 0 t.m. 255 liggen.

```
PRINT "Naam" SPC (20) "Woonplaats"
Naam                 Woonplaats
 Ok
```

Zie ook de SPACE$-functie.

SQR*(getal)*

Functie. Vierkantswortel uit ***getal***. ***getal*** moet groter dan of gelijk zijn aan nul, zoniet, dan volgt een **Illegal function call** fout.

```
10 FOR X=10 TO 25 STEP 5
20 PRINT X, SQR(X)
30 NEXT X
RUN
 10          3.16228
 15          3.87298
 20          4.47214
 25          5
 Ok
```

STOP

Onderbreken van de programma-uitvoering en terugkeren naar de directe stand.

Tussen elke twee instructies in het programma mag een STOP-opdracht worden opgenomen, teneinde de uitvoering

van het programma op die plaats te onderbreken. De interpreter ziet de STOP-opdracht als een programmafout (foutnr. 64). Als geen ON ERROR GOTO werkzaam is wordt uitvoering van het programma onderbroken met de melding **Stop in *regelnummer.*** Als wel een ON ERROR GOTO is gepasseerd kan de STOP-instructie d.m.v. een foutafhandeling verwerkt worden.
Als na uitvoering van de STOP-opdracht teruggekeerd is naar de directe stand kan de programma-uitvoering worden voortgezet door CONT in te typen.
De STOP-instructie wordt net zo behandeld als het indrukken van de SHIFT-STOP-toets op het rechter toetsenbord.

STR$*(getal)*

Functie. Geeft het ***getal,*** weer als een string. Als ***getal*** $>=0$ is het eerste karakter van deze string een spatie; voor waarden van ***getal*** <0 is het eerste karakter een min-teken.

```
10 INPUT X
20 Y$ = STR$ (X)
30 Z = LEN (Y$)
40 PRINT Z
50 GOTO 10
RUN
? 12
 3
? -10
 3
```

STRING$*(getal 1,getal 2)*
STRING$*(getal 1,string)*

Functie. Levert een string van ***getal 1*** gelijke karakters, alle met de ASCII-waarde ***getal 2*** respectievelijk alle gelijk aan het eerste karakter van ***string***.

```
10 Z$ = STRING$ (10,45)
20 PRINT Z$ "Jaaroverzicht" Z$
RUN
----------Jaaroverzicht-------
Ok
```

```
10 X$ = "X1" : PRINT STRING$(6,X$)
RUN
 XXXXXX

 PRINT STRING$(4,65)
 AAAA
```

Als ***getal 2*** = 32 of ***string*** = " " dan is STRING$ equivalent met SPACE$.
STRING$ kan gebruikt worden om "niet intikbare tekens" in een string te zetten.

```
A$ = STRING$(10,124)
A$ = STRING$(10,7)
```
(10xpiep)

SWAP ***variabele 1, variabele 2***

Verwisselen van de waarden van twee variabelen.
De SWAP-opdracht kan worden toegepast voor elk type variabele (integer, enkele-precisie, dubbele-precisie en string).
Uiteraard moeten de beide variabelen van hetzelfde type zijn, zoniet dan volgt een **Type mismatch**.

```
10 A$ + "EEN":B$ = "ALLEN"C$ = " VOOR "
20 PRINT A$;C$;B$
30 SWAP A$,B$
40 PRINT A$;C$;B$
RUN
EEN VOOR ALLEN
ALLEN VOOR EEN
 Ok
```

```
10 DIM A$(100)
100 FOR I = 1 TO 100
110 IF A$(I-1) > A$(I) THEN SWAP A$(I-1), A$(I)
120 NEXT
```
Dit programma zet de elementen van het array A$ op alfabetische volgorde.

TAB*(getal)*

De weergave van de spaties op beeldscherm of printer, tot aan positie ***getal.*** De meest linkse positie is positie 1; de meest rechtse positie is gelijk aan de breedte van de printer c.q. beeldschermvenster, minus 1.

Is ***getal*** groter dan de breedte van het venster, dan wordt eerst een regelopvoer gegeven ***getal*** wordt met de breedte van het scherm verminderd en TAB wordt opnieuw uitgevoerd. Is ***getal*** >255 dan volgt een **Illegal function call**.TAB kan alleen worden gebruikt in combinatie met PRINT en LPRINT.

```
10 PRINT "Naam" TAB(25) "Bedrag":PRINT
20 READ A$,B$
30 PRINT A$ TAB(25) B$
40 DATA "H.J. Willemsen","f 245.46"
RUN
Naam                      Bedrag

H.J. Willemsen            f 245.46
 Ok
```

Als TAB gebruikt wordt in combinatie met een cursor besturingsopdracht (CHR$(4), zie hoofdstuk 6) dan kan tekst op het scherm worden gewist omdat TAB een aantal spaties afdrukt.

```
10 PRINT CHR$(4)CHR$(5)CHR$(1)"Bedrag"
20 PRINT CHR$(4)CHR$(5)CHR$(1)TAB(8)X
```

TAB wist hier het eerder afgedrukte woord **Bedrag.**

TAN*(getal)*

Functie. Geeft de tangens van de hoek ***getal,*** waarbij getal uitgedrukt is in radialen. Is het resultaat van de TAN-functie groter dan de grootste waarde die de BASIC kan verwerken, dan volgt een **Division by zero** fout.

```
10 Y=Q*TAN (X)/2
```

TRON
TROFF

Volgen van de uitvoering van de programmaregels.

De TRON- en TROFF-opdrachten zijn bedoeld als hulp bij het opsporen van fouten in een programma. Wanneer de TRON-opdracht is uitgevoerd (in de directe of in de programma stand) worden de regelnummers van alle programmaregels, wanneer ze aan de beurt zijn om uitgevoerd te worden, op het beeldscherm weergegeven. Deze regelnummers verschijnen tussen pijltjes

De TRON-opdracht kan worden opgeheven met TROFF.

```
TRON
 Ok
10 K=10
20 FOR J=1 TO 2
30 L=K+10
40 PRINT J;K;L
50 K=K+10
60 NEXT
70 END
RUN
←10→←20→←30→←40→ 1  10  20
←50→←60→←30→←40→ 2  20  30
←50→←60→←70→
 Ok
TROFF
 Ok
```

USING *opmaak string; lijst van expressies*

Op het beeldscherm weergeven van strings en getallen volgens een aan te geven opmaak, waarbij ***opmaakstring*** de opmaak bepaalt.

lijst van expressies bevat de getallen of strings waarvan de waarden moeten worden weergegeven op het beeldscherm. De expressies in de lijst worden van elkaar gescheiden door komma's of punt-komma's (;).
USING kan alleen gebruikt worden in een PRINT of LPRINT-opdracht. Per PRINT of LPRINT-opdracht mag maar één USING voorkomen.

De opmaakstring bestaat uit een tekst die ge-PRINT wordt. Bepaalde karakters in de opmaakstring geven aan waar en hoe de expressies afgedrukt worden (opmaakveld). Deze karakters zijn ! spatie # ÷ + - $ ↑.
Voor ieder opmaakveld moet tenminste één expressie af te drukken zijn. Zijn er méér expressies dan opmaakvelden, dan wordt de opmaakstring opnieuw gebruikt.

Allereerst een voorbeeld van een PRINT USING-opdracht:

PRINT USING"De P2000 heeft ## kilobyte ÷ ÷"; 16; "RAM"
De P2000 heeft 16 kilobyte RAM
Bij het uitvoeren van deze opdracht wordt in het opmaakveld ## het af te drukken getal 16 geplaatst en in het opmaakveld ÷ ÷ de string RAM.

Wordt de USING opdracht gebruikt voor de weergave van strings, dan zijn 2 verschillende opmaakvelden mogelijk:

"!" geeft aan dat alleen het eerste karakter van de string moet worden weergegeven.

"÷ n spaties ÷"
geeft aan dat n + 2 karakters van de string moeten worden weergegeven. Bevinden zich tussen de integer-delingtekens geen spaties dan worden 2 karakters weergegeven.
Is de string langer dan n + 2 karakters dan worden de resterende karakters niet afgedrukt. Is de string korter dan n + 2 karakters dan wordt het verschil toegevoegd in de vorm van spaties achter de string.

```
10 A$ = "KIJK":B$ = "UIT"
20 PRINT USING "!";A$;B$
30 PRINT USING "÷     ÷";A$;B$
40 PRINT USING "÷      ÷";A$;B$;"!!"
RUN
KU
KIJKUIT
KIJK UIT   !!
```

Wordt de PRINT USING opdracht gebruikt voor de weergave van getallen dan kunnen de volgende opmaakvelden worden aangegeven:

\# Dit teken wordt gebruikt om de plaats van elk cijfer van een getal aan te geven. Is een getal opgebouwd uit een kleiner aantal cijfers dan met het aantal #-tekens wordt aangegeven, dan wordt dat getal rechts-lijnend afgedrukt, d.w.z. er worden extra spaties voorgevoegd.
Op één plaats in dit veld mag een punt worden geplaatst.

Geeft het opmaakveld aan dat een cijfer vooraf moet gaan aan een decimale punt, dan wordt één cijfer altijd afgedrukt, dus eventueel als een nul. Indien nodig worden getallen afgerond.

```
PRINT USING "##.##"; 78
78
PRINT USING "###.##";987.648
987.65
PRINT USING "##.##   ";10.2;5.3;66.789;.234
10.20   5.30   66.79   0.23
```

In dit laatste voorbeeld werden in de opmaakstring drie spaties opgenomen om de getallen bij de uitvoer van elkaar te scheiden.

+ Een plus-teken aan het begin of eind van het opmaakveld wordt gebruikt om voor resp. achter het weer te geven getal het teken van dat getal (plus of min) af te drukken.
Een min-teken na de laatste # van het opmaakveld zorgt ervoor dat negatieve getallen worden gevolgd door een min-teken, in plaats van het min-teken voor het getal.

```
PRINT USING "+##.##      ";68.95;2.4;55.6;-.9
+68,95     +2.40     +55.60     -0.90
PRINT USING "##.##-      "; 68.95;+22.45;-7.01
68.95  22.45  7.01-
```

****** Een dubbele ster (**) aan het begin van het opmaakveld zorgt ervoor dat de spaties aan het begin van het weergaveveld worden vervangen door één of meer sterren.
De ** aanduiding neemt bovendien twee karakterposities in beslag.

```
PRINT USING "**##.#      ";12.39;-0.9;1765.1
**12.4     **-0.9     1765.1
```

$$ Twee dollartekens aan het begin van het opmaakveld zorgen ervoor dat direct voor het af te drukken getal een dollarteken wordt weergegeven. Bovendien neemt de $$ aanduiding twee karakterposities in beslag, waarvan er één wordt gebruikt voor het $ teken.

Deze opmaak aanduiding kan niet worden gebruikt in combinatie met de exponentiële weergave van getallen.
Ook negatieve getallen kunnen niet worden weergegeven, tenzij het min-teken achter het betreffende getal wordt weergegeven.

PRINT USING "$$###.##";456.78
$ 456.78

**$ Deze aanduiding combineert de twee bovengenoemde. Spaties vóór een getal worden vervangen door * tekens en voor het getal wordt een $ teken afgedrukt.
**$ neemt drie karakterposities in beslag, waarvan er één het $ teken is.

PRINT USING "$#.##";2.34**
****$2.34**

Een komma, links van de decimale punt in het opmaakveld zorgt ervoor dat vóór elk derde cijfer links van de decimale punt een komma wordt afgedrukt. Een komma rechts van de decimale punt werkt als „veld-scheider". Deze komma wordt afgedrukt en de #-jes die nog volgen vormen een nieuw opmaakveld. De komma neemt één karakterpositie in. De komma kan niet worden gebruikt in combinatie met de exponentiele (wetenschappelijke) notatie van getallen.

PRINT USING ",####.##";1234.5
1,234.50

↑ Vier exponent tekens (omhoog wijzende pijltjes) (SHIFT @) mogen achter de cijfer-aanduidingskarakters (#) worden geplaatst om aan te geven dat het getal volgens de exponentiële notatie moet worden weergegeven.
De vier pijlen zorgen er tevens voor dat ruimte wordt gereserveerd om E+XX weer te geven. Tenzij een aan het getal voorafgaand plus-teken of een achter het cijfer weer te geven plus of min-teken wordt gespecificeerd, zal vòòr het af te drukken getal één positie worden gereserveerd voor een spatie of een min-teken.

```
PRINT USING "##.##↑↑↑↑";234.56
 2.35E+02

PRINT USING ".####↑↑↑↑";888888
 .8889E+06

PRINT USING "+.##   ↑↑";123
+.12E+03
```

Bestaat het af te drukken getal uit meer cijfers dan aangegeven posities, dan wordt een procent-teken (%) voor dat getal afgedrukt. Het procent-teken wordt ook voor een getal afgedrukt wanneer het aantal printposities t.g.v. een afronding is overschreden.

```
PRINT USING "##.##";111.22
%111.22

PRINT USING ".##";.999
%1.00
```

Indien het aangegeven aantal cijfers groter is dan 24, volgt een **Illegal function call**.

Het is mogelijk de opmaakstring apart te definieren. Dit is handig als op meer plaatsen in het programma volgens dezelfde USING moet worden ge-PRINT.

```
10 A$="###.##"
20 PRINT USING A$;X,Y,Z
```

Tussen PRINT en USING mag een tekst of een andere PRINT-opdracht worden geplaatst.

```
10 PRINT CHR$(4)CHR$(10)CHR$(12)"Besteed bedrag"
USING A$;X,Y,Z
```

USR ***cijfer (getal*** of ***string)***

Aanroepen van een subroutine in machinetaal met het argument ***getal*** of ***string***. ***cijfer*** moet liggen tussen 0 en 9, corresponderend met het cijfer dat is opgegeven in de DEF USR instructie. Als ***cijfer*** wordt weggelaten, wordt aangenomen dat ***cijfer*** = 0.

```
40 B = T*SIN (Y)
50 C = USR0(B/2)
60 D = USR1(0)
```

Zie ook hoofdstuk 17: Programeren in machinetaal.

VAL*(string)*

Functie. Geeft de numerieke waarde van ***string***. Is het eerste karakter van ***string***, dat geen spatie is, ongelijk aan +, -, & of aan een cijfer dan is het resultaat van de VAL-functie gelijk aan nul.

```
100 LINE INPUT "Kies 1, 2 of 3"; A$
110 ON VAL(A$) GOTO 1000, 2000, 3000
```

VARPTR*(variabele-naam)*

Het geheugenadres van het eerste databyte van de waarde, van de variabele aangegeven met ***variabele-naam***. Voor uitvoering van deze functie moet aan ***variabele-naam*** een waarde zijn toegekend. Is dit nog niet gebeurd, dan volgt een **Illegal function call** fout.
Elk type variabele (integer, floating point, enz.) mag worden gebruikt; het resultaat is een integer tussen -32768 en +32767. De negatieve getallen verwijzen naar het geheugen gebied &H8000 tot &HFFFF. Door 65536 bij het negatieve getal op te tellen vindt men het adres in positieve notatie.
De VARPTR-functie wordt vaak gebruikt om het adres te vinden van een variabele of van het begin van een array, zodat deze waarde dan kan worden gebruikt in bijvoorbeeld een machinetaal routine.

Vóórdat de VARPTR-functie wordt gebruikt voor het vinden van het beginadres van een arrayelement, moeten alle in het programma voorkomende „gewone" variabelen van een waarde zijn voorzien. Dit, omdat array's verschuiven als nieuwe „gewone" variabelen worden toegevoegd, door ze voor het eerst een waarde te geven.

```
10 X = VARPTR(Y)
20 X = VARPTR(Z%(0))
```

WAIT *poortnr, getal 1*
WAIT *poortnr, getal 1, getal 2*

getal 1 en ***getal 2*** moeten van het integer type zijn.

Het tijdelijk stopzetten van de programa-uitvoering, afhankelijk van de op ingangs ***poortnr*** aangeboden data.

De programma-uitvoering wordt stopgezet totdat op de ingangspoort een aan te geven bitpatroon aanwezig is.
Het aangeboden getal op de ingangspoort wordt ingelezen en volgens XOR-functie bewerkt met ***getal 2***.Dan wordt het resultaat volgens de AND-functie bewerkt met ***getal 1***. Is dit resultaat gelijk aan nul, dan wordt opnieuw van de ingangspoort een getal ingelezen en worden deze bewerkingen opnieuw uitgevoerd. Pas wanneer het resultaat van de XOR- en AND-bewerking ongelijk aan nul is, wordt de uitvoering van het programma voortgezet (bij de opdracht die volgt op de WAIT-instructie). Indien ***getal 2*** wordt weggelaten, wordt aangenomen dat deze waarde nul is.

In feite is WAIT een compacte uitvoering van de instruktie
100 IF INP (*poortnr*) XOR *getal2* AND *getal1* = 0 THEN 100

10 WAIT &H20, &H10, &HFF
Als er geen cassette in de recorder zit, blijft het programma wachten.

20 WAIT &H20, &H10
Het programma wacht tot het klepje van de recorder geopend wordt. (Zie hoofdstuk 18 over de functies van poort &H20).

Waarschuwing: De WAIT-opdracht kan de interpreter in een toestand brengen waaruit "geen ontsnappen mogelijk is", namelijk, wanneer de ingangswaarde XOR ***getal2*** AND ***getal1*** altijd gelijk aan nul blijft. In dat geval moet de computer opnieuw m.b.v. de RESET-toets worden gestart en is het programma dat in het geheugen aanwezig was verloren.

De STOP-toets werkt niet als het programma in de WAIT staat.

Tabel 1.

ASCII TABEL VOOR DE P2000

De volgende tabel geeft het resultaat van een instructie
PRINTCHR$*(ASCII-waarde);*

0 einde toonstring
1 cursor aan
2 cursor uit
3 zet linker helft van beeldgeheugen op scherm (OUT 48,0) en CHR$(29)
4 positioneer cursor. Te volgen door CHR$*(regel)*CHR$*(kolom)*
5 scherm naar printer
6 definieer plaats cursor als nieuw cursor punt
7 piep
8 cursor naar links met overloop op vorige regel, tenzij boven aan venster
9 horizontale TAB
10 cursor omlaag of scherm opvoer
11 wis karakter links van cursor
12 wis venster, zet cursor en cursorpunt links boven in venster
13 cursor naar linker kantlijn venster
14 zet deel-PRINT aan
15 wis van plaats cursor tot cursorpunt
16 cursor 1 stap naar links
17 cursor 1 stap omhoog
18 cursor 1 stap omlaag
19 cursor 1 stap naar rechts
} tot randen van het venster
20 zet cursor op kolom *k*. Te volgen door CHR$(*k*)
21 wis tot einde schermregel
22 wis tot einde venster
23 begin van toonstring
24 cursor 1 stap naar rechts met overloop
28 zet venster op 24 regels, 40 kolommen en wis venster
29 als cursor niet links in het venster staat dan CHR$(13) + CHR$(10)
30 zet deel-PRINT uit
31 zet cursor op cursorpunt

De volgende ASCII waarden hebben tot gevolg het afdrukken van het erachter vermelde karakter.

32	spatie	48	0	64	@	80	P	96	–	112	p
33	!	49	1	65	A	81	Q	97	a	113	q
34	"	50	2	66	B	82	R	98	b	114	r
35	#	51	3	67	C	83	S	99	c	115	s
36	$	52	4	68	D	84	T	100	d	116	t
37	%	53	5	69	E	85	U	101	e	117	u
38	&	54	6	70	F	86	V	102	f	118	v
39	'	55	7	71	G	87	W	103	g	119	w
40	(	56	8	72	H	88	X	104	h	120	x
41	)	57	9	73	I	89	Y	105	i	121	y
42	*	58	:	74	J	90	Z	106	j	122	z
43	+	59	;	75	K	91	←	107	k	123	¼
44	,	60	<	76	L	92	÷	108	l	124	‖
45	-	61	=	77	M	93	→	109	m	125	¾
46	.	62	>	78	N	94	↑	110	n	126	½
47	/	63	?	79	O	95	£	111	o	127	■

De volgende ASCII waarden worden afgegeven door een toets indruk van het kleine toetsenbord.

3	STOP	programma uitvoering stopt in programma stand
128	ZOEK	geeft in directe stand inhoud cassette
129	⊡	wist cassette in directe stand (na verkregen toestemming)
	DEF	zet P2000 in EDIT stand en toont laatst ge-LIST-e of ge-EDIT-e regel
	START	schakelt van directe stand naar programma stand en neemt programma in geheugen in uitvoering
	SHIFT 5	zet deel-PRINT-routine aan en begint te LIST-en vanaf begin programma
14	M	zet deel-PRINT-routine aan.

Andere toetsen

16	←	cursor stap naar links	
17		cursor stap omhoog	
		cursor stap omlaag	
19	→	cursor 1 stap naar rechts	
8		correctie toets	
11		SHIFT correctie toets	
27		CODE	
9		TAB	

26	
29	←|
31	
30	SHIFT
25	SHIFT TAB
12	⌧ wis schern
15	⏭ wis regel

Tabel 2.

ENKELE NUTTIGE ADRESSEN

In deze tabel worden de geheugen adressen samengevat die van belang zijn voor het programmeren in BASIC. De inhoud van deze adressen kan met POKE opdrachten worden gewijzgd. Er is een niet-verwaarloosbare kans dat een POKE op een van deze adressen desastreuse gevolgen heeft voor het programma. Alle volgende adressen zijn hexadecimaal.

6000-600B - toetsenbord input buffer

600C - teller toetsenbord input buffer

6010-6011 - klok, loopt van 0 tot plm. 20 minuten met stapjes van 20 msec. Klok uit te lezen door PEEK(&H6010) + 256*PEEK(&H6011)

6016 - baudsnelheid printer

605C - beschikbare geheugen-ruimte

waarde	RAM geheugen
1	16 K
2	32 K
3	48 K

60A1 - aantal reeds op deze pagina afgedrukte regels op de printer

60A5 - print snelheid naar scherm met SHIFT ingedrukt

60A6 - print snelheid naar scherm (0 is hoogste snelheid) zonder SHIFT ingedrukt

60A9 - aantal lege regels tussen twee pagina's op de printer (opstart = 6)

60AA - maximum aantal regels per printer pagina (opstart =66)

60AB - regelbreedte printer (opstart =80)

60AC - als hier 1 staat spoelt de cassette niet eerst terug naar het begin bij CLOAD en CLOAD*. Staat hier 128 dan, wordt in de programma stand wel toestemming gevraagd bestanden te overschrijven

60AD - linker kolom venster beeldscherm (0 = linker kolom scherm)

60AE - bovenste regel venster beeldscherm

60AF - hoogte venster beeldscherm

60B0 - breedte venster beeldscherm

60B3 - horizontale positie cursor

60B4 - verticale positie cursor
60B6 - letter type, even voor hoofdletters, oneven voor kleine letters
6253 - kolomteller printer
625C-625D adres begin BASIC programma (normaal &H6547)
63B6 - Type-indicator FAC
63F8-63F9 - regelnummer van laatst ge-EDIT-e of ge-LIST-e regel
6405-6406 - adres begin variabelen ruimte
6407-6408 - adres begin array ruimte
6409-640A - adres begin vrije ruimte
6509-6510 - Floating Point Accumulator

600F 0 = NO H# H# H LOCK

066-6067 E 5

60A0 NVOER FE/254 = 255 KAR

60A1: 0 = HALF
1 = 0
2 = VDA A - BLOKJE
* 3 = " "
4 = TRANSP.

60B1-60B2 ADRES CRSR H5000 + P(H60B3) + 80 * P(H60B4)

60BE 0 = GEEN TAB # 9 = TAB

623E 625 TABEL U R S CH 28 9C

63B8-63B9 CLEAR

HEADER:
* 6030-6031 TRANSFER
* 6032-6033 TOT. LENG
* 6034-6035 ZINVOL
* 6036-603D N
603E-6040 EXTENSION
6041 FILE TYPE
6042 IDENT. TOETSBORD
6043-6044 START SA
6045-6046 LAAD SA
6047-604E N
604F BLOKNR

Tabel 3.

Grafische tekens
In deze tabel wordt weergegeven:
Het alfa-numerieke karakter
De ASCII-waarde
Het overeenkomende grafische teken

Karakter	ASCII	Grafisch	Karakter	ASCII	Grafisch
Spatie	32		–	96	
!	33		a	97	
"	34		b	98	
£	95		c	99	
$	36		d	100	
%	37		e	101	
&	38		f	102	
	39		g	103	
	40		h	104	
)	41		i	105	
*	42		j	106	
+	43		k	107	
	44			108	
	45		m	109	
	46		n	110	
/	47		o	111	
0	48		p	112	
1	49		q	113	
2	50		r	114	
3	51		s	115	
4	52		t	116	
5	53		u	117	
6	54		v	118	
7	55		w	119	
8	56		x	120	
9	57		y	121	
	58		z	122	
,	59		¼	123	
<	60		ll	124	
=	61		¾	125	
>	62		÷	92	
?	63		■	127	

Opmerking: Door bij alle ASCII waarden 128 op te tellen wordt het teken „geinverteerd" m.a.w. het teken verschijnt in de achtergrondkleur en de achtergrond verschijnt in de kleur van het teken. Het teken behorende bij " is natuurlijk niet in een string in te geven; dat kan alleen met CHR$(34). Voor de te gebruiken omschakel karakters zie hoofstuk 8; Grafische mogelijkheden.
Er is een onregelmatigheid in de ASCII waarden bij £ en ÷. Dit is om de compatibiliteit voor teksten met andere computers optimaal te doen zijn.

Tabel 4.

FOUT CODES EN FOUTMELDINGEN

Na het constateren van een fout geeft de functie ERR het nummer van de geconstateerde fout.

1 **NEXT without FOR**
Een variabele in een NEXT-instructie correspondeert niet met een reeds eerder uitgevoerde FOR of er is een NEXT gevonden zonder dat een FOR gepasseerd was.

2 **Syntax error**
In een programmaregel komen één of meer onjuiste karakters voor, bij voorbeeld spelfouten in opdrachten.

3 **RETURN without GOSUB**
Een RETURN-opdracht is niet vooraf gegaan door een GOSUB opdracht waarmee de betreffende subroutine zou moeten worden aangeroepen.

4 **Out of DATA**
Een READ opdracht wordt uitgevoerd, terwijl in het programma geen DATA-regels met nog ongelezen DATA-waarden meer aanwezig zijn.

5 **Illegal function call**
Een parameter die buiten het toegestane bereik valt vormt het argument van een numerieke of stringfunctie. Een Illegal function call fout kan gegeven worden als:
- een negatieve of te grote index wordt gebruikt bij een array - een LOG functie moet worden uitgevoerd met het argument gelijk aan of kleiner dan nul
-een negatief getal tot een niet-gehele macht wordt verheven
-een SQR functie wordt uitgevoerd met als argument een getal kleiner dan nul
- een floating point getal voorkomt met én een negatieve mantissa én een niet-gehele exponent
- een USR functie wordt aangeroepen terwijl het startadres nog niet is opgegeven
- een ongeldig argument voorkomt in één van de functies MID$, LEFT$, RIGHT$, INP, OUT, WAIT, PEEK, POKE, TAB, SPC, STRING$, SPACE$, INSTR, ON ... GOTO, DELETE etc.

6 **Overflow**
Het resultaat van een berekening is groter dan de grootste waarde die de BASIC interpreter aan het gebruikte type variabele kan toekennen. Is het resultaat kleiner dan het kleinste absolute getal dat kan worden weergegeven (underflow) dan wordt als resultaat de waarde 0 toegekend zonder dat dit wordt gemeld.

7 **Out of memory**
Een programma is te lang, heeft te veel geneste FOR ... NEXT lussen of GOSUB's, te veel variabelen, of een van cassette in te laden array is groter dan het array is ge-DIM-d.

8 **Undefined line number**
In een GOTO, GOSUB, IF...THEN...ELSE, RUN ***regelnummer***, RESTORE, EDIT, ON ERROR GOTO of RESUME opdracht is een niet bestaand regelnummer opgegeven

9 **Subscript out of range**
Een array-element wordt gebruikt met een index die groter is dan de omvang van dat array, of een verkeerd aantal indices wordt opgegeven.

10 **Redimensioned array**
Voor hetzelfde array worden twee DIM opdrachten in het programma opgenomen, dan wel een DIM opdracht wordt uitgevoerd nadat één of meer elementen van dat array (met de opstart dimensie gelijk aan 10) al een waarde hebben gekregen

Division by zero
In een expressie komt een deling door nul voor, dan wel de waarde nul wordt tot een negatieve macht verheven.

12 **Illegal direct**
Een opdracht wordt gegeven, die in de directe stand ongeldig is, terwijl de P2000 in de directe stand staat (b.v.INPUT)

13 **Type mismatch**
Aan een string variabele wordt een numerieke waarde toegekend, of een functie, die een string argument verwacht, heeft als argument een numerieke waarde of omgekeerd.

14 **Out of string space**
De string variabelen nemen meer stringruimte in beslag, dan voor dit doel was gereserveerd. Tenzij de stringruimte expliciet wordt opgegeven m.b.v. een CLEAR-instructie worden door de interpreter 50 plaatsen gereserveerd voor de opslag van strings.

15 **String too long**
Er wordt een poging gedaan om een string te creëren die uit meer dan 255 karakters bestaat.

16 **String formula too complex**
Een stringexpressie is te lang of te ingewikkeld. De expressie moet worden verdeeld in een aantal kortere expressies.

17 **Can't continue**
Er wordt een poging gedaan om, m.b.v. de CONT opdracht een programma voort te zetten dat:
- veranderingen heeft ondergaan na een onderbreking in de uitvoering, of
- niet bestaat

18 **Undefined user function**
Een functie wordt aangeroepen voordat de functie-definitie DEF FN is gegeven.

19 **No RESUME**
Een foutafhandelingsroutine wordt aangeroepen, maar bevat geen RESUME-instructie.

20 **RESUME without ERROR**
Een RESUME-instructie wordt uitgevoerd, zonder dat de foutafhandelingsroutine is aangeroepen.

21 **Unprintable error**
Voor de opgetreden fout is geen specifieke foutmelding aanwezig. Dit wordt gewoonlijk veroorzaakt door uitvoering van een ERROR instructie met een niet in BASIC gedefinieerde foutcode.

22 **Missing operand**
Een expressie bevat een operator (+, -, *, LOG, enz.) zonder dat er een operand (= te bewerken gegeven) op volgt.

64 **Stop**
De gebruiker heeft de uitvoering van het programma onderbroken door het indrukken van de SHIFT-STOP toets of door een STOP-instructie in het programma.

65 **Geen cassette**
Bij CSAVE, CLOAD, RUN, ZOEK of WIS was geen cassette aanwezig in de recorder.

66 **Cassette fout B**
Begin van de tape (komt in BASIC niet voor)

67 **Leesfout**
Bij het lezen van cassette is een fout opgetreden.

68 **Cassette fout D**
Checksom fout in het start merk

69 **Cassette fout E**
Einde van de tape (cassette vol)

70 **Cassette fout F**
Cassette vol maar het hele bestand weggeschreven

71 **Geen stopje**
Bij CSAVE of wis was de cassette niet voorzien van een schrijfstopje (zie hoofdstuk 9)

73 **Cassette fout I**
Terugspoeltijd voorbij. Wellicht is de band gebroken.

77 **Niet gevonden**
Bij CLOAD bleek het gevraagde bestand of programma niet op de cassette te staan.

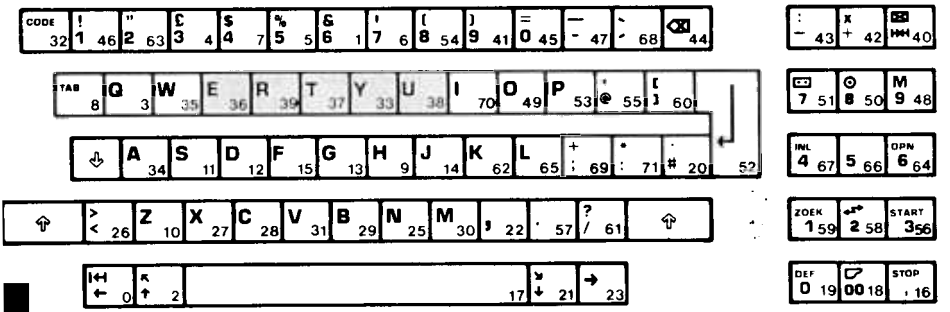
80 **Geen printer**
Een printer-routine wordt aangeroepen, terwijl geen printer beschikbaar is.

Opmerkingen: De fouten 64-80 kunnen niet worden afgedrukt met ERROR*foutnummer*
Als geen ON ERROR GOTO actief is worden boven genoemde foutmeldingen afgedrukt op het beeldscherm gevolgd door **in *regelnummer***, waarbij ***regelnummer*** het regelnummer is, waar de fout optrad.

Tabel 5.

TOETSCODE

Het toetsenbord van de P2000 geeft niet direct een ASCII-waarde af. In feite genereert een toets indruk eerst een toetscode, die in de interpreter wordt omgezet in de bijbehorende ASCII-waarde. Een overzicht van de toetscodes geeft de onderstaande figuur:



De toetscode met SHIFT ingedrukt is 72 groter dan de toetscode zonder SHIFT.

De procedure is nu als volgt

Na het indrukken van een toets wordt de toetscode, TC, in de toetsenbord input-buffer geplaatst. Deze staat op de geheugenplaatsen &H6000-&H600B, terwijl het aantal aanwezige toetscodes in de buffer geteld wordt in &H600C. De toetscode wordt "vertaald" naar de ASCII-waarde in de vertaaltabel in de interpreter en wel als volgt:

AS = PEEK (&H1814 + TC)

In de praktijk van het programmeren in BASIC zult U de toetscode zelden gebruiken. De instructie A = INP("") levert direct de ASCII-waarde van de laatst-ingedrukte toets.

Toetscodes kunnen gebruikt worden om de niet-aangesloten toetsen een functie te geven. Deze toetsen geven allen de ASCII-waarde 7 af, hoewel de toetscodes verschillend zijn. Ook voor het bereiken van zeer bijzondere effecten moet soms van de toetscode gebruik worden gemaakt.

```
100 POKE &H600C,0
110 IF PEEK ("") = 0 THEN 110 ELSE TC = PEEK(&H6000): POKE
&H600C,0
```

Tabel 6.

BASIC TOKENS

BASIC instructies worden in het geheugen van de P2000 in gecodeerde vorm opgeslagen. Elke instructie heeft daartoe een code van 1 byte ("token"). Bij LIST en EDIT wordt de bij de code behorende tekst op het scherm getoond. Hieronder volgt een alfabetische lijst van BASIC-instructies met de waarde van de code.

WOORD	*CODE*	*HEX*	*WOORD*	*CODE*	*HEX*
*	191	BF	DIM	133	85
+	189	BD	EDIT	156	9C
-	190	BE	ELSE	146	92
/	192	C0	END	128	80
<	203	CB	EQV	197	C5
=	202	CA	ERASE	150	96
>	201	C9	ERL	182	B6
÷	200	C8	ERR	183	B7
↑	193	C1	ERROR	157	9D
ABS	206	CE	EXP	214	D6
AND	194	C2	FIX	223	DF
ASC	229	E5	FN	177	B1
ATN	218	DA	FOR	129	81
AUTO	170	AA	FRE	207	CF
CDBL	222	DE	GOSUB	140	8C
CHR$	230	E6	GOTO	136	88
CINT	220	DC	HEX$	226	E2
CLEAR	171	AB	IF	138	8A
CLOAD	172	AC	IMP	198	C6
CONSOLE	144*	90	INP	208	D0
CONT	166	A6	INPUT	132	84
COS	215	D7	INSTR	185	B9
CSAVE	173	AD	INT	205	CD
CSNG	221	DD	LEFT$	232	E8
DATA	131	83	LEN	244	E0
DEF	163	A3	LET	135	87
DEFDBL	154	9A	LINE	155	9B
DEFINT	152	98	LIST	167	A7
DEFSNG	153	99	LLIST	168	A8
DEFSTR	151	97	LOG	213	D5
DELETE	169	A9	LPOS	209	D1

LPRINT	162	A0	SIN	216	D8
MID$	234	EA	SPACE$	231	E7
MOD	199	C7	SPC(	178	B2
NEW	174	AE	SQR	211	D3
NEXT	130	82	STEP	188	BC
NOT	187	BB	STOP	143	8F
OCT$	225	E1	STR$	227	E3
ON	160	9E	STRING$	184	B8
OR	195	C3	SWAP	149	95
OUT	159	9D	TAB(	175	AF
PEEK	219	DB	TAN	217	D9
POKE	164	A4	THEN	186	BA
POS	210	D2	TO	176	B0
PRINT	165	A5	TROFF	148	94
READ	134	86	TRON	147	93
REM	142	8E	USING	179	B3
RESTORE	139	8B	USR	181	B5
RESUME	158	9E	VAL	228	E4
RETURN	141	8D	VARPTR	180	B4
RIGHT$	233	E9	WAIT	161	9F
RND	212	D4	WIDTH	145*	91
RUN	137	89	XOR	196	C4
SGN	204	CC			

CONSOLE en WIDTH hebben geen betekenis in de cassette-BASIC.

REGISTER

Ruimte voor aantekeningen.

Het toetsenboord van de P2000 met de ASCII-waarden, die de toetsen afgeven mèt (boven) en zonder (onder) de SHIFT-toets ingedrukt.