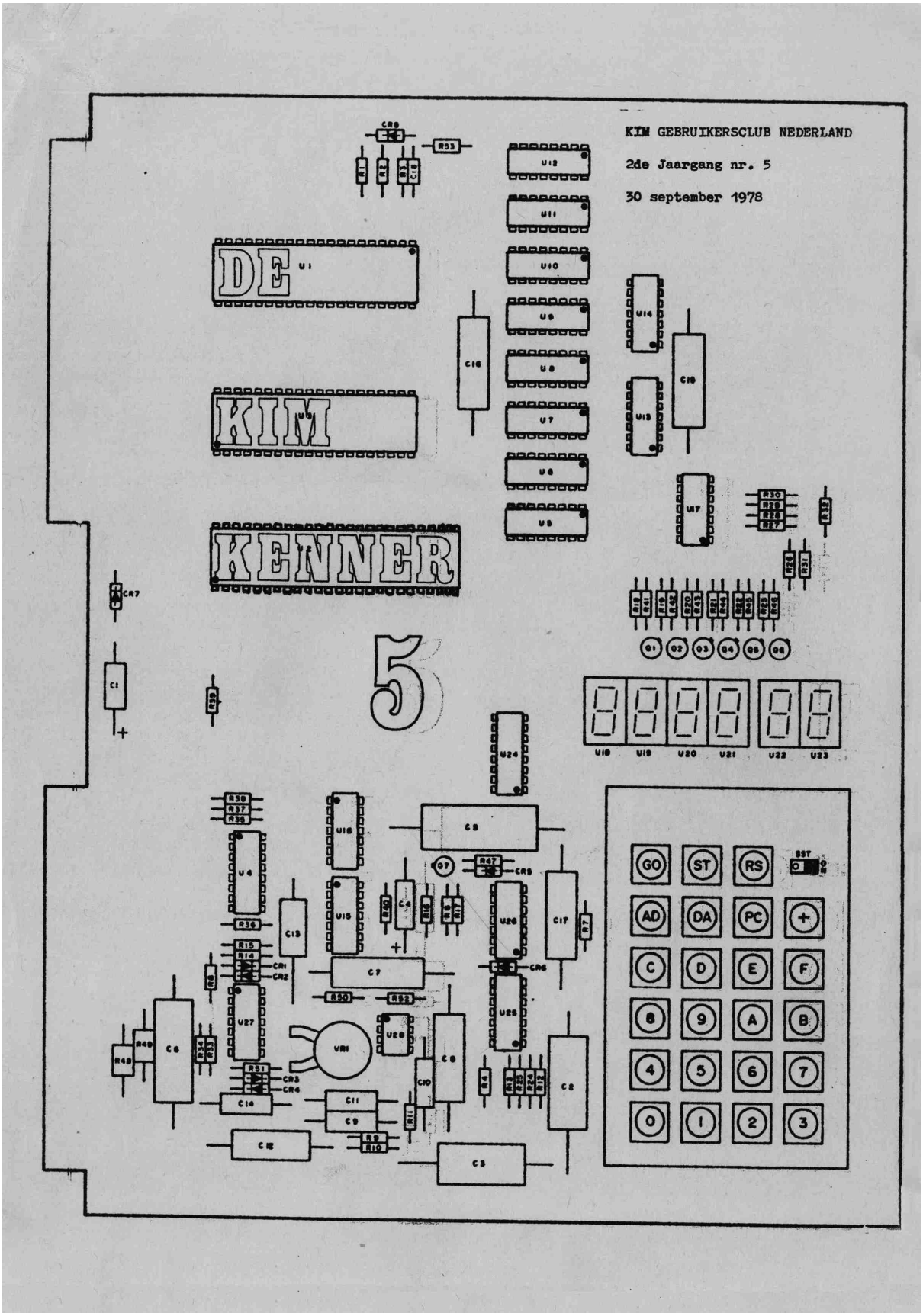

KIM GEBRUIKERSCLUB NEDERLAND
2de Jaargang nr. 5
30 september 1978
DE
KIM
KENNER
5
GO
ST
RS
AD
DA
PC
+
C
D
E
F
8
9
A
B
4
5
6
7
0
1
2
3

KIM

MICROCOMPUTERS. WAT DOEN ZE ERMEE ?

Vorige week werd ik door een verslaggever van een krant gebeld met de vraag: Kunt u mij het adres geven van mensen, die zelf een computer hebben en daar interessante dingen mee doen, zoals automatisch de zonnegordijnen open en dicht doen ?

Dit heeft mij weer eens aan het denken gezet. Wat doen wij toch eigenlijk met onze micro's ? Als ik bij anderen ga kijken en tracht te ontdekken wat ermee gedaan wordt, blijkt die vraag het volgende antwoord op te leveren: Voorzover het zichtbare toepassingen betreft, is er slechts een enkeling, die iets met zijn hobbycomputer "doet". Iedereen is wel erg druk met het hebben en krijgen van geweldige ideeen of het programmeren van mooie programma's, het filosoferen over modulair programmeren, subroutines, interrupt, S-100 bus of andere bussen, EPROM's programmeren en nog een eindeloze serie van computeronderwerpen.

Toepassingen echter die gezien kunnen worden zijn er niet of nauwelijks. Dit geldt zelfs niet alleen voor amateurgebruikers, maar evengoed voor de industrie. Een enkeling heeft een of meer processen, die door een micro bestuurd worden. De meeste zijn er mee bezig, zoals dat heet.

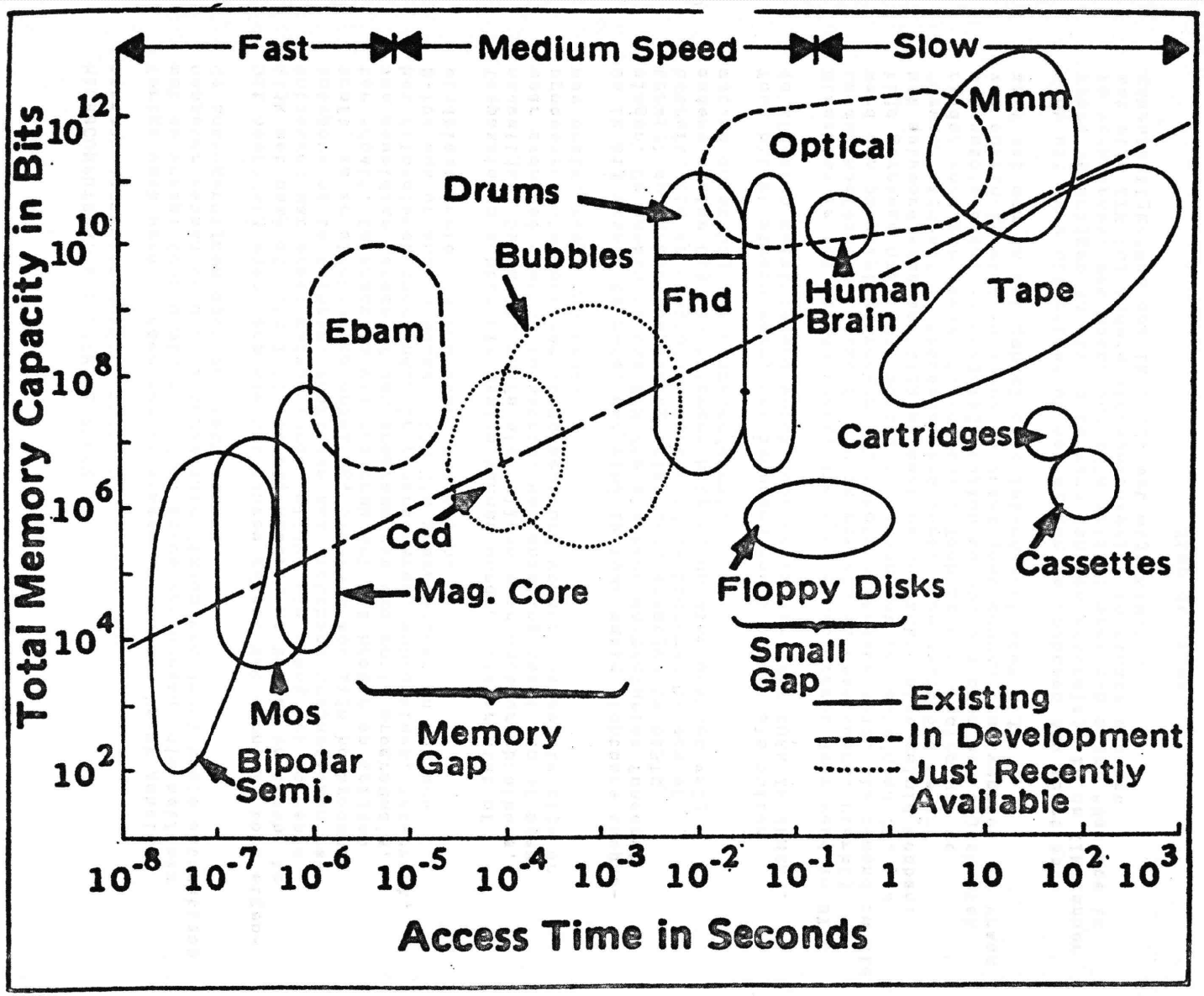
Ga ik bij mezelf te rade, dan vind ik twee matig zichtbare toepassingen. De eerste is een KIM die als terminalbesturing fungeert terwijl de tweede toepassing enige programmatjes in BASIC betreft, die of spelletjes spelen of de kinderen op een of andere manier bij hun huiswerk helpen. Op zich dus een vrij matige oogst bij een computerfanaat.

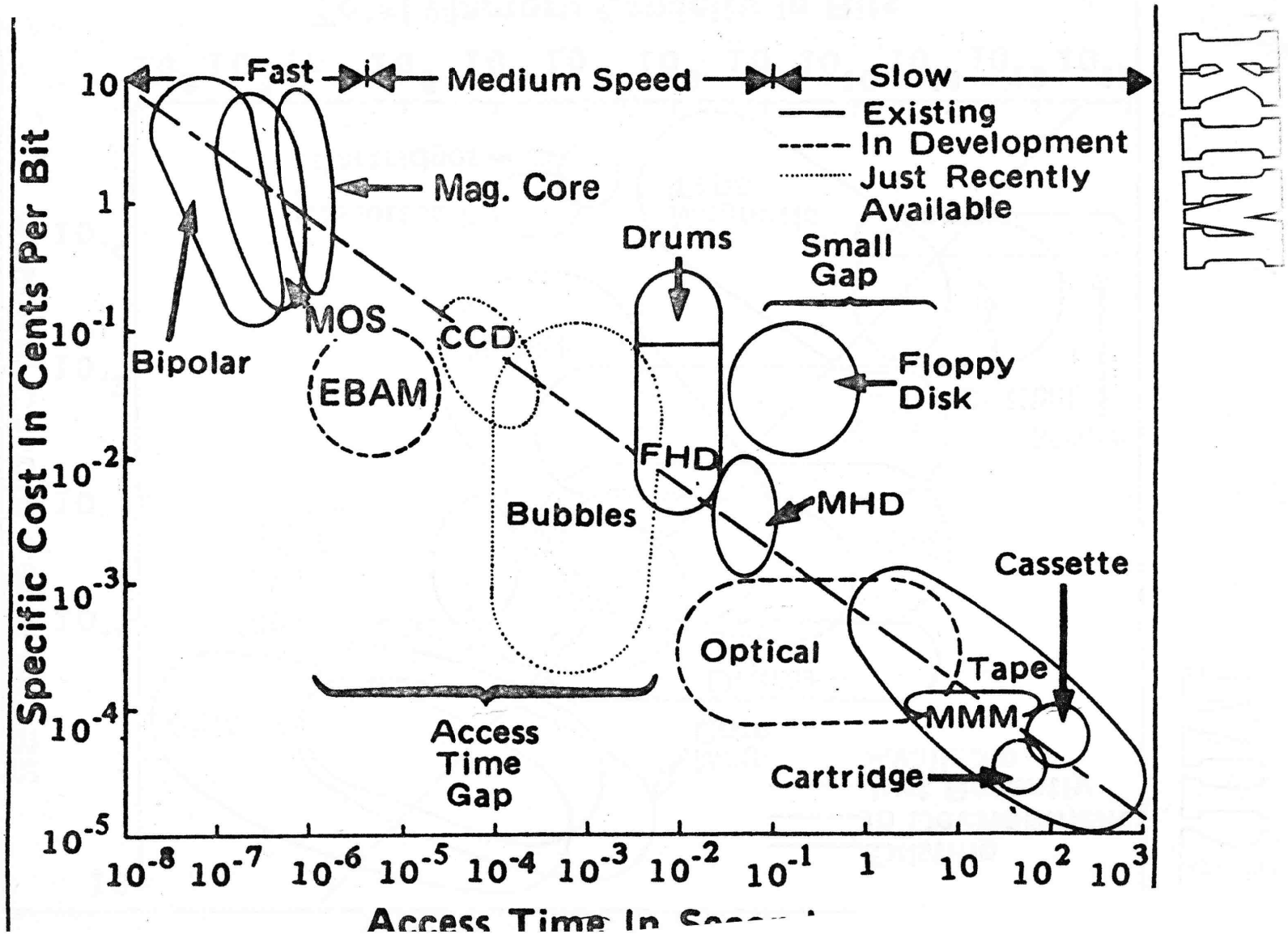
Toch blijkt steeds weer, dat iedereen zowel binnen als buiten de KIM-club geintrigeerd wordt door de vraag: "WAT DOEN ZE ERMEE".

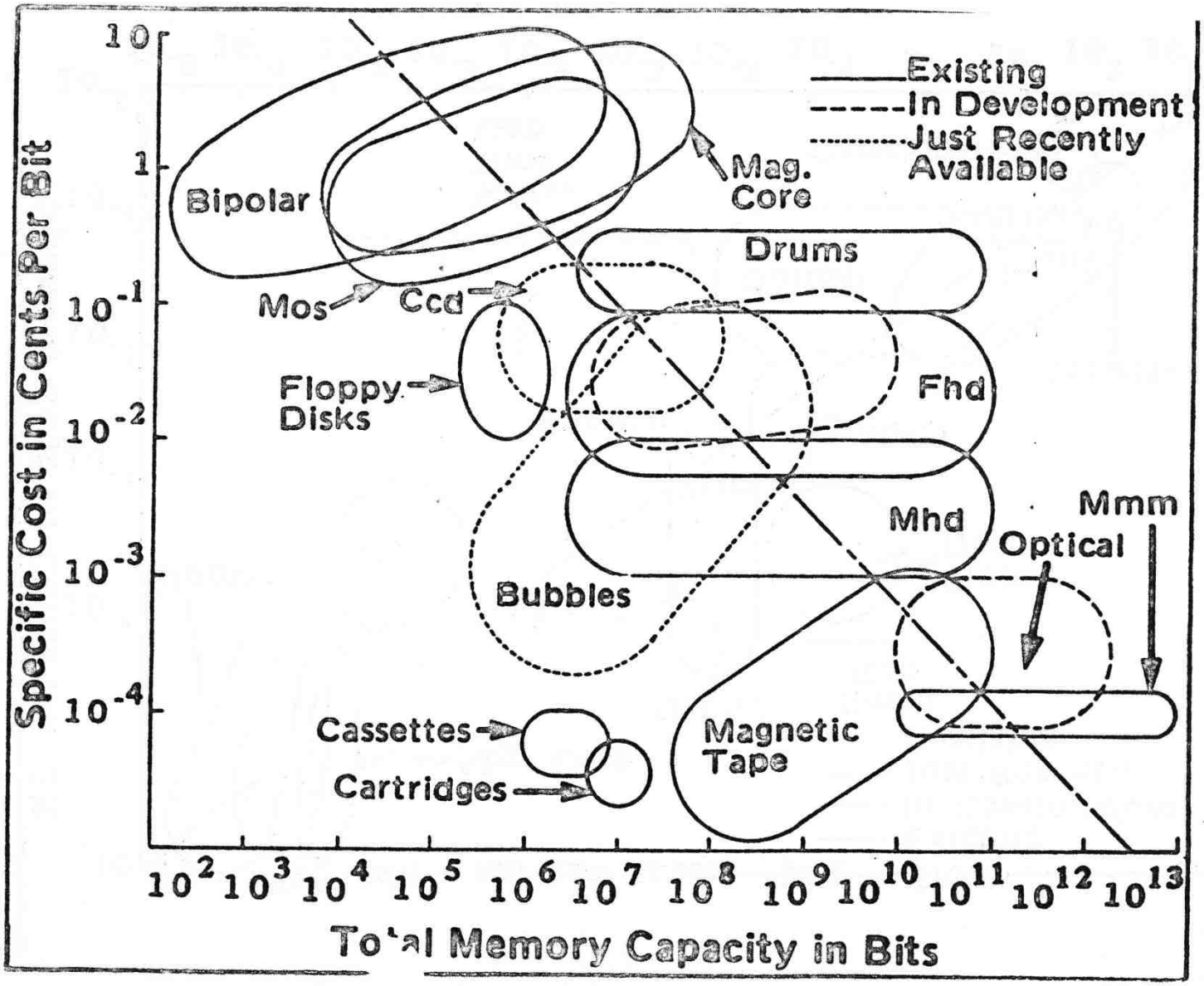
Misschien is het wel de vraag, die niet deugt. Vorig jaar werd in BYTE in een hoofdartikel dezelfde vraag gesteld en beantwoord. Hierbij werd een parallel getrokken net auto's omstreeks 1910. Als iemand in die tijd gevraagd had: "Wat moeten de mensen met een auto doen ?", was het antwoord even moeilijk geweest en zou zeker niet geluid hebben: "Gewoon rijden." of "Boodschappen doen. Soms zelfs 500 meter rijden voor een pakje sigaretten.". Toch is dit precies wat we met auto's doen. Vermoedelijk zullen we dus pas over een tijdsbestek van enkele tientallen jaren de vraag goed kunnen beantwoorden. De vraag zelf zal dan in het geheel niet interessant meer zijn.

Toch wil ik u uitnodigen om met elkaar te proberen een soort antwoord te krijgen en dit in de vorm van een artikeltje in de KIM-kenner te verwoorden. Het motto van zo'n artikel moet dan zijn: "Wat doe ik met mijn KIM (of andere microcomputer)." in plaats van het gebruikelijke"Hoe doe ik ..... met mijn KIM."

Siep de Vries


Fast
Medium Speed
Slow
Total Memory Capacity in Bits
10^{12}
10^{10}
10^{8}
10^{6}
10^{4}
10^{2}
Mmm
Optical
Drums
Bubbles
Ebam
Fhd
Human Brain
Tape
Cartridges
Ccd
Mag. Core
Floppy Disks
Cassettes
Small Gap
Mos
Bipolar Semi.
Memory Gap
Existing
In Development
Just Recently Available
10^{-8} 10^{-7} 10^{-6} 10^{-5} 10^{-4} 10^{-3} 10^{-2} 10^{-1} 1 10 10^{2} 10^{3}
Access Time in Seconds


Fast
Medium Speed
Slow
Existing
In Development
Just Recently Available
Mag. Core
Drums
Small Gap
Bipolar
MOS
CCD
EBAM
Floppy Disk
FHD
MHD
Bubbles
Cassette
Optical
Tape
MMM
Cartridge
Access Time Gap
Specific Cost In Cents Per Bit
10
1
10^{-1}
10^{-2}
10^{-3}
10^{-4}
10^{-5}
10^{-8} 10^{-7} 10^{-6} 10^{-5} 10^{-4} 10^{-3} 10^{-2} 10^{-1} 1 10 10^{2} 10^{3}


Existing
In Development
Just Recently Available
Bipolar
Mag. Core
Drums
Mos
Ccd
Floppy Disks
Fhd
Mhd
Mmm
Optical
Bubbles
Cassettes
Cartridges
Magnetic Tape
Specific Cost in Cents Per Bit
10
1
10-1
10-2
10-3
10-4
10² 10³ 10⁴ 10⁵ 10⁶ 10⁷ 10⁸ 10⁹ 10¹⁰ 10¹¹ 10¹² 10¹³
Total Memory Capacity in Bits

KIM GEBRUIKERS CLUB NEDERLAND
HARDWARE LIBRARY

Enige interessante trucjes met de KIM.

Zodra de teletype in- en uitgang een ander doel dienen dan de KIM-monitor via een teletype of display met de gebruiker te laten communiceren, willen er nog wel eens enige probleempjes optreden.

Deze komen erg vaak voort vanuit het feit, dat de hardware van de teletype een z.g. halfduplex interface is. Dit wil zeggen, dat ieder signaal, dat op de keyboard ingang komt, ook naar de printer uitgang gestuurd wordt.

Veronderstel nu, dat een programma eerst een character wil lezen en pas daarna besluiten om iets uit te printen (of niet). Dit is op zich onmogelijk, maar er is wel een praktische benadering die een plezierig resultaat geeft.

Voordat het lezen van een teken start, moet dan een 0 naar het teletype outputbit gestuurd worden. Deze nul wordt weer 1 gemaakt als het te lezen teken binnen is. Het is dan voor het printende apparaat net alsof er een rubout gestuurd is, aangezien in de teletypecircuits een 0 overheerst.

Als het printende apparaat een display is zal er meestal niets op het scherm geprint worden. Op deze manier is een "semi full duplex"- verbinding te realiseren.

Een tweede probleem doet zich voor bij apparaten, waarvan de lezer door de computer met tekens bediend kan worden. (CTRL/Q en CTRL/S voor TAPE-ØN en TAPE-ØFF)

Het teken TAPE-ØN gaat uitstekend, maar het teken TAPE-ØFF moet a.h.w. tussen de binnenkomende tekens ingewurmd worden. (alle binnenkomende tekens worden automatisch geëchoed)

In zo'n situatie is het mogelijk om de ingang "doof" te maken, zodat er ook niets meer uitgaat. Dit kan gebeuren door bit 5 van $ 1742 1 te maken. Dit blokkeert de teletype ingang.

Hierna kan dan het TAPE-ØFF-teken gestuurd worden. Onvermijdelijk is, dat enige input-tekens verloren gaan.

ALWEER EEN TAAL OP DE KIM

Ongeveer 12 jaar geleden maakte ik kennis met een hogere computertaal, uitgebracht door DEC, op een PDP-8. Deze taal heette FØCAL en was bedoeld als een conversationeel hulpmiddel voor mensen, die met weinig inspanning op het gebied van programmeren, programmaatjes moesten schrijven voor wetenschappelijke berekeningen, analyse van data, statistiek en administratie.

Wie schetst mijn verbazing, toen enige weken geleden Anton MÜller bij mij kwam en zei: "Ik heb bij Aresco in Amerika voor $ 75 een FØCAL-compiler gekocht. Zullen we die eens proberen?

Uiteraard werd het programma in de KIM gedraaid en groot was de verrassing toen dit inderdaad een vrijwel volledige copie was van de FØCAL-versie op de PDP-8!!!!!!

Bij gebruik is het eerste, dat opvalt aan dit systeem de grote "gebruikersvriendelijkheid". Dit is vooral dankzij het commando MØDIFY, dat de gebruiker in staat stelt zonder veel moeite binnen een programmaregel verbeteringen en wijzigingen aan te brengen.

Iets dat niet zo verbazingwekkend meer is tegenwoordig, is dat FØCAL alle normale rekenkundige functies zoals: - optellen, aftrekken, vermenigvuldigen, delen, machtsverheffen, absolute waarde, geheel getal, random number en afronding kanken -, evenals formules met haakjes. (getallen 9 cijfers nauwkeurig).

Eveneens vriendelijk is de mogelijkheid om een programma te tracen, d.w.z. ieder statement dat uitgevoerd wordt, wordt ook uitgeprint.

Verdere faciliteiten die FØCAL biedt, zijn:

- Stringhandling. Maximale grootte van een string is 250 characters.
- Arrays. Variabelen die uit een rijtje getallen bestaan in plaats van uit één getal.
- Volledige gebruikersbesturing van papierindeling bij output.
- Bij het uitvoeren van foute statements wordt het statement dat niet thuis te brengen is, uitgeprint met een pijltje onder de plaats, waar de fout mogelijkerwijs zit.
- Mogelijkheid om de echo, die normaal gesproken van het toetsenbord altijd naar de printer doorkomt op de KIM, te onderdrukken. Dit is werkelijk een zeer gave softwareтruc.
- Uitstekende documentatie. De documentatie bestaat uit een beschrijving van de taal en zijn faciliteiten plus een listing.

Vooral opmerkelijk is de flexibele opbouw van het geheel. Op diverse plaatsen is bijvoorbeeld ruimte opengelaten voor patches, die een gebruiker erin zou willen maken. De schrijver noemt dit: Space for hackers. Dit betekent, dat een gebruiker nieuwe statements kan bedenken en nieuwe functies.

De input-output dient wel een aparte waardering!!!!!!

KIM GEBRUIKERS CLUB NEDERLAND
HARDWARE LIBRARY

Het is mogelijk om aan het systeem nieuwe apparatuur met software toe te voegen door de subroutine adressen in een tabel erbij te zetten. Ter illustratie hiervan het volgende:

Mijn KIM heeft een papertapereader, papertapepunch en printer, die volgens eigen inzichten aangesloten zijn.
Er was ongeveer 5 minuten werk nodig om te zorgen, dat FØCAL-programma's hiervan gebruik kunnen maken!!!!!

Rest mij nog te vertellen, dat het pakket ongeveer 7 K ruimte in beslag neemt en dat een systeem van 16 K de gebruiker in staat stelt programma's van redelijke omvang te schrijven.

Mocht u besluiten tot de aanschaf van dit pakket over te gaan, dan zult u hier erg veel plezier aan beleven!!!!!!!!!!!!!!!!!!!!

Siep de Vries.

TE KOOP aangeboden: Electrische IBM schrijfmachine ƒ 400,-
Programmeerbare calculator ƒ 100,-
19 inch rek in ruil voor onderdelen.
Tel.: 078 - 71607

TE KOOP aangeboden: Video display kaarten + boekje RTTY-TV-Display
Set van 4 printkaarten op europa formaat, bevattende ca 40 IC's, karaktergenerator, geheugen, kristal-oscillator enz. ƒ 125,00
Tel: 072 - 12 66 52

TE KOOP aangeboden: Kleinbeeld MONITOR merk Sanyo, beeldgrootte 9 inch
Is voor demonstratiedoeleinden gebruikt. Te bevragen bij Visser Assembling Electronics B V te Alkmaar. Tel: 072 - 12 66 52

HARDWARE LIBRARY

KIM HINTS

Since you and your KIM-1 are relative strangers, we'd like to help you get better acquainted. The material in this pamphlet will answer questions that are frequently asked by a new KIM-1 user.

ANSWERS TO POPULAR KIM SYSTEM QUESTIONS

1. IS IT POSSIBLE TO OUTPUT DIGITS OTHER THAN HEX TO THE 6 OUTPUT LED'S?

Since the 6502 is doing all segment decode and multiplex, it is possible to display data other than hex on a 7-segment readout. A pseudo alphabet has been developed and is displayed in the 7-segment display of the KIM in a scrolling manner.

2. WHEN HANDLING THE BOARD, WOULD THE STATIC HAZARD BE RELIEVED IF ALL EDGE CONNECTORS WERE SHORTED TOGETHER?

The static problems are not as serious once the devices are installed in the P.C. board. Just be sure to use grounded tools and to discharge yourself to ground before touching KIM or the connected circuits.

3. WHAT TYPE OF LED READOUT IS USED ON KIM-1 FOR U18, etc?GENERAL COMMON ANODE OR CATHODE?

USE MAN-72 Type displays, available from many manufacturers. General common anodes should work, although you may find intensity differences between them.

4. WHERE CAN I GET MORE 44-PIN EDGE CONNECTORS FOR KIM?

The connector is a standard part – you can order a Vector No. R644 from most electronic supply houses. The connector is also carried by most Radio Shack stores as Part No. 276-548.

5. ARE THERE ANY INTERFACES OR PROM PROGRAMMERS AVAILABLE WITH KIM TO PROGRAM EPROMs OR TO DUPLICATE PROMs?

No, not yet.

6. IS THERE AN I/O EXPANSION BOARD AVAILABLE?

Not yet . . . soon, we hope.

7. IS THERE A BOARD AVAILABLE TO MAKE USE OF MEMORY ADDRESSES 0400-13FF?

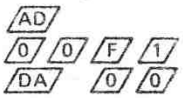
Check the **"Kilobaud"** article (issue #4, April 1, 1977, page 74) entitled "KIM Memory Expansion."

8. HOW DO I SET UP MY KIM FOR AUDIO CASSETTE RECORDING AND PLAYBACK?

A number of KIM-1 customers have reported difficulty in achieving correct results for the sample problem shown in Sec. 2.4 of the KIM-1 User Manual. In addition, some customers have experienced problems in recording or playback of audio cassettes. (Sec. 2.5 of the KIM-1 User Manual). In all cases, the problems have been traced to a single cause: the inadvertent setting of the DECIMAL MODE.

The 6502 Microprocessor Array used in the KIM-1 system is capable of operating in either binary or decimal arithmetic mode. The programmer must be certain that the mode is selected correctly for the program to be executed. Since the system may be in either mode after initial power-on, a specific action is required to insure the selection of the correct mode.

Specifically, the results predicted for the sample problem (Sec. 2.4) are based on the assumption that the system is operating in the *binary* arithmetic mode. To insure that this is the case, insert the following key sequence prior to the key operations shown at the bottom of Page 11 of the KIM-1 User Manual.



This sequence resets the decimal mode flag in the Status Register prior to the execution of the sample program.

The same key sequence may be inserted prior to the key operations shown on pages 14 and 15 for audio cassette recording and playback. These operations will not be performed correctly if the decimal mode is in effect.

In general, whenever a program is to be executed in response to the GO key, the programmer should insure that the correct arithmetic mode has been set in the status register (00F1) prior to program execution.

9. HOW DO I SOLVE AUDIO CASSETTE INTERFACE PROBLEMS?

A. Insure that memory location 00F1 has been set to a value of 00 before recording or playing back the tape. This is the source of 90% of all cassette problems.

B. Mis-adjustment of the variable resistor (VR1) in the cassette circuitry is almost *never* a problem. Any setting near the center of its rotation will work fine.

C. Make sure that +12V is connected during playback. NOTE: +12V is not required for recording, so a lack of +12V will result in good recording but no playback.

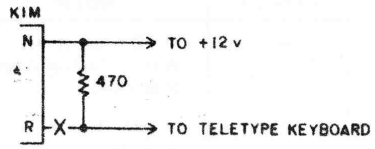
D. If the display frequently relights showing FFFF, the fault is probably in the tape unit itself – not the KIM. Using poor quality cassettes is usually to blame. Some cassette recorders have such poor power filtering circuits that they will work fine on batteries, but will not work with an AC adapter because of hum induced during record or playback. Tapes should always be rewound before removal from the machine, as a fingerprint on the tape will result in errors on playback.

E. Make sure that only a single ground line is run from the KIM ground to the barrel of the microphone input of the cassette recorder. Leave the barrel of the earphone output ungrounded. The shield around the line to the earphone should be attached to ground on KIM.

F. Problems of playing a tape recorded on one KIM system back on another system or a different cassette player can usually be solved by adjusting the head adjustment screw on the new cassette recorder. Play back a cassette recorded on the old deck on the new machine and adjust the head screw on the new machine for maximum volume. This adjustment is especially critical when using the SuperTape program.

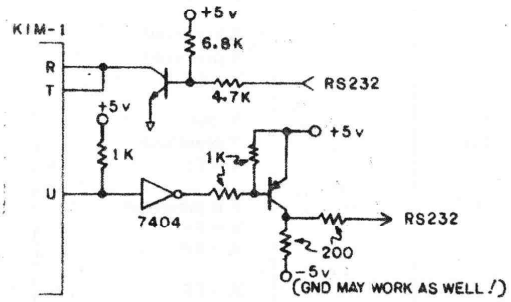
10. HOW DO I SOLVE TELETYPE PROBLEMS?

A. The most common problem is that the system does not respond to a reset-rubout sequence with a model 33 Teletype. This can be fixed by removing the wire connected to pin R on the KIM application connector, connecting a 470 ohm resistor to that wire, and connecting the other end of the resistor to the +12V supply at pin N.



B. No information is available on connecting other Teletype models (14, 28, 32) to KIM.

C. Schematics for interfacing KIM to an RS232C port are in the April, 1976 "Byte" magazine and in the first issue of the KIM user notes. (Reproduced below):



D. Other common sources of Teletype problems are a short circuit in C5 or a burned-out Q7. Signal tracing with a 'scope should reveal these problems.

11. HOW DO I SOLVE PAPER TAPE PROBLEMS?

A. KIM-1's having a date code in 1975 on the 6502 will not read paper tape correctly. These CPU's will be replaced by MOS without charge. Tom Pittman's TINY BASIC will not work on these machines either. The problem occurs because early versions of the processor did not set the zero flag correctly on TXA, TYA, TAX, or TAY instructions.

B. When using a Texas Instruments Silent 700 data terminal equipped with digital cassettes or other higher-speed paper tape devices, a Q (paper tape dump) may be performed at any speed acceptable to the data terminal, but playback (through the L command) must be at 10 cps.

12. WHAT DO I DO ABOUT OTHER PROBLEMS?

A. If the RESET on KIM causes only a single digit or segment to light on the display, the KIM must be returned for repair.

B. When in doubt, check all power supply voltages *on the KIM board,* not at the power supply terminals.

C. When software works strangely or erratically, decimal/binary mode problems may be involved.

D. There is an error in the KIM Resident Assembler manual regarding the addresses for the symbol table vectors. The vector locations are DF, E0, E1, E2. The text is incorrect, the example is correct.

E. Problems with KIM-2/3's which fail the memory test program can almost always be traced to excessive cable length between the KIM-1 and the KIM-2/3. Any cable should be 6" in length or less.

13. WHAT ARE THE KIM SYSTEM POWER SUPPLY REQUIREMENTS?

KIM 1 – Microcomputer Board:

Recommended: 1.2A +5V ±5%
100 mA +12V ±5%

The actual power measured ranges 700 mA to 1A at +5V and the schematic indicating 3A at transformer is incorrect.

KIM 3A–8K RAM Memory Board:

Recommended: +5V, 3A

Average consumption calculated is about 2.4A. Board has +5V regulator accepting unregulated +8 to +10V DC.

KIM 4 – Mother Board:

Consumption about 200mA. Board has +5V regulator accepting unregulated +8 to +10V DC and +12V regulator accepting unregulated +15V DC to support both KIM1 and KIM 4. KIM 4 has 6 slots for memory expansion with KIM2 and KIM3 and hence a total power supply requirement is a cumulative value dependent on KIM-System configuration.

14. WHAT SOFTWARE IS AVAILABLE?

The following software is available for use with the KIM-1 and/or other 6502-based systems:

KIM

GEBRUIKERS CLUB NEDERLAND

HARDWARE LIBRARY

1. Tiny BASIC – runs in 2K. $5 for paper tape from:
   Tom Pittman
   Box 23189
   San Jose, California 95153
2. Many games and other information in the KIM-1 User Group Newsletter, $5 for 6 issues:
   Eric Rehnke
   109 Centre Avenue
   W. Norriton, PA 19401
3. An excellent Chess playing program which runs in 1K. $10
   MICRO CHESS
   27 Firstbroke Rd.
   Toronto, CANADA M 4E 2L2
4. A good group of games plus an intermediate-level language called PLEASE for KIM-1 – $15 from:
   THE COMPUTERIST
   Post Office Box 3
   S. Chelmsford, MA 01824
5. The 6502 Program Exchange
   2920 Moana
   Reno, NV 89509
6. Micro Software Specialists
   2024 Washington Street
   Commerce, TX 75428
7. KIMATH, a complete floating-point math package including both source and object code is available from MOS Technology for $15.
8. A 4K version of FOCAL, a BASIC-like interpreter, and a 6K Resident assemble/text Editor, both with source listings and object code on KIM cassette or paper tape are available from:
   ARESCO
   314 Second Ave.
   Haddon Heights, NJ 08035
   The FOCAL is $50 and the assembler/Editor is $70. A complete information package is $2.
9. An 8K version of BASIC for KIM is available for $99 from:
   Johnson Computing
   123 W. Washington St.
   Medina, Ohio 44256
   (215) 725-4568
10. "FIRST BOOK OF KIM" is a collection of games, utility programs, hints and kinks, etc. (180 pgs). $9.00 plus 50¢ postage from:
    ORB
    P.O. Box 311
    Argonne, ILL 60439

INTERVAL TIMER OPERATION

1. OPERATION

a. Loading the timer

The divide rate and interrupt option enable/disable are programmed by decoding the least significant address bits.

KIM SUBROUTINES

CALL	ADDRESS	ACTION	ARG.	RESULT	NOTES
JSR AK	1EFE	Check for key depressed	–	A	A = 0 = Key down A≠0 = No Key down X & Y lost
JSR GETKEY	1F6A	Get key from keyboard	–	A	A>15 illegal or no key
JSR SCANS	1F1F	Display F9, FA, FB	F9, FA, FB	–	A, X, Y are lost
JSR GETCH	1E5A	Put character from TTY in A	–	A	X preserved Y = FF
JSR PRTBYT	1E3B	Prints A as 2 Hex Char.	A	–	A preserved X preserved Y = FF
JSR PRTPNT	1E1E	Prints Contents of FB & FA on TTY	FB, FA		A lost X preserved Y = FF
JSR OUTCH	1EA0	Print ASCII char in A on TTY	A	–	X is preserved Y = FF A = FF
JSR OUTSP	1E9E	Print a space	–	–	A = FF X preserved Y = FF

The starting count for the timer is determined by the value written to that address.

Writing to Address	Sets Divide Ratio To	Interrupt Capability Is
1704	1	Disabled
1705	8	Disabled
1706	64	Disabled
1707	1024	Disabled
170C	1	Enabled
170D	8	Enabled
170E	64	Enabled
170F	1024	Enabled

b. **Determining the timer status**

After timing has begun, reading address location 1707 will provide the timer status. If the counter has passed the count of zero, bit 7 will be set to 1, otherwise, bit 7 (and all other bits in location 1707) will be zero. This allows a program to "watch" location 1707 and determine when the timer has timed out. Note that reading 1707 provides an entirely different function from writing the same location.

c. **Reading the count in the timer**

If the timer has not counted past zero, reading location 1706 will provide the current timer count and disable the interrupt option; reading location 170E will provide the current timer count and enable the interrupt option. Thus the interrupt option can be changed while the timer is counting down. Note that you read 1706 or 170E regardless of which location (1704-0F) was written to start the timer.

If the timer has counted past zero, reading either memory location 1706 or 170E will restore the divide ratio to its previously programmed value, disable the interrupt option and leave the timer with its current count.

d. **Using the interrupt option**

In order to use the interrupt option described above, line PB7 (application connector, pin 15) should be connected to either the $\overline{\text{IRQ}}$ (Expansion Connector, pin 4) or $\overline{\text{NMI}}$ (Expansion Connector, pin 6) pin depending on the desired interrupt function. PB7 should be programmed as an *input* line (it's normal state after a RESET).

NOTE

If the programmer desires to use PB7 as a normal I/O line, the programmer is responsible for disabling the timer interrupt option (by writing or reading address 1706) so that it does not interfere with normal operation of PB7. Also, PB7 was designed to be wire-ORed with other possible interrupt sources; if this is not desired, a 5.1K resistor should be used as a pull-up from PB7 to +5v. (The pull-up should NOT be used if PB7 is connected to NMI or IRQ.)

2. CAPABILITIES

The KIM Interval Timer allows the user to specify a preset count and a clock divide rate by writing to a memory location. As soon as the write occurs, counting at the specified rate begins. The timer counts down at the clock frequency divided by the divide rate. The current timer count may be read at any time. At the user's option the timer may be programmed to generate an interrupt when the counter counts down past zero. When a count of zero is passed, the divide rate is automatically set to 1 and the counter continues to count down at the clock rate starting at a count of FF (-1 in two's complement arithmetic). This allows the user to determine how many clock cycles have passed since the timer reached a count of zero. Since the counter never stops, continued counting down will reach 00 again then FF, and the count will continue.

3. INTERVAL TIMER AND KEYBOARD OPERATION

The following three programs show the use of the interval timer, keyboard and seven segment displays in user programs.

The first program loads a value of 50 in the timer and waits for it to time out, repeats the process and then increments the count in the display register (00FA and 00FB) and calls the display subroutine SCANS. The process then repeats.

The second program performs the same function as the first, but uses the timer to provide interrupts, rather than watching the timer status register (1707). Thus this program is constantly cycling through the display program SCANS except when the timer generates an interrupt. When an interrupt occurs the interrupt service routine (starting at location 010C) resets the timer, increments the display register and returns to the display program. Note that the LED display is brighter when using this program because most of the computer's time is spent displaying rather than watching the timer.

The third example program demonstrates the use of the keyboard and display. Any key depressed will appear in the rightmost digit of the display and will be shifted to the left with each successive keyboard entry.

Notice that the SCANS routine not only displays the contents of 00F9, 00FA and 00FB but also returns with the Z flag set to Ø if a key is currently depressed. The GETKEY routine is then called to determine which key has been depressed. Since the SCANS subroutine takes several milliseconds, a call to this routine can be used to "waste time" and let any keybounce stop.

INTERVAL TIMER

LOC CODE DEFINITION OF COMMONLY USED LOCATIONS

```
DA      =$1700          DATA REG A
DDA     =$1701          DATA DIREC REG A
DB      =$1702          DATA REG B
DDB     =$1703          DATA DIREC REG B

                TIMERS (WRITE TIME TO)

C1D     =$1704          DIV BY 1    DISABLE INT
C8D     =$1705          DIV BY 8    DISABLE INT
C64D    =$1706          DIV BY 64   DISABLE INT
C1024D  =$1707          DIV BY 1024 DISABLE INT

C1E     =$170C          DIV BY 1    ENABLE INT
C8E     =$170D          DIV BY 8    ENABLE INT
C64E    =$170E          DIV BY 64   ENABLE INT
C1024E  =$170F          DIV BY 1024 ENABLE INT

TRD     =$1706          READ TIME DISABLE INT
SR      =$1707          READ INT STAT
TRE     =$170E          READ TIME ENABLE INT
```

WHEN THE INTERRUPT STATUS IS READ THE INTERRUPT IS NEITHER DISABLED OR ENABLED. BIT 7 IS A ONE IF TIME OUT HAS OCCURRED. BIT 7 IS ZERO IF TIME OUT HAS NOT OCCURRED. BITS 0-6 ARE ALL ZERO

WHEN THE TIMER TIMES OUT THE DIVIDER IS SET TO A DIV BY ONE AND THE TIMER CONTINUES TO COUNT AT CLOCK RATE

WHEN THE TIMER IS READ THE DIVIDER IS RESTORED TO ITS ORIGINAL VALUE AND THE INTERRUPT IS RESET

```
SCANS   =$1F1F          EXTERNAL SUBROUTINES
INCPT   =$1F63
GETKEY  =$1F6A
```

TO USE INTERRUPT PB7 MUST BE EXTERNALLY WIRED TO IRQ

Program 1

THIS EXAMPLE DOES NOT USE INTERRUPTS – THE DISPLAY WILL DIM AS A RESULT OF SLOW SCANNING

```
                COUNT   =2              COUNT DOWN 2 TIMES
                DELAY   =50             EACH DELAY 50 CYCLES
0000                    *=$0000         ORG AT 0
0000  A2 02     START1  LDX  =COUNT
0002  A9 32             LDA  =DELAY
0004  8D 06 17  AGAIN   STA  C64D       DIV BY 64 DISABLE INT
0007  2C 07 17  WAIT    BIT  SR         READ STATUS DISABLE INT
000A  10 FB             BPL  WAIT       BIT 7 = 1 TIME OUT COMPLETE
000C  CA                DEX
000D  D0 F5             BNE  AGAIN      LOOP ON COUNT
000F  20 63 1F          JSR  INCPT      MONITOR UTIL INC FA,FB
0012  20 1F 1F          JSR  SCANS      MONITOR UTIL DISP F9,FA,FB
0015  4C 00 00          JMP  START1
```

INTERVAL TIMER (Continued)

CARD = LOC CODE CARD

Program 2

THIS EXAMPLE USES INT
WIRE PB7 TO IRQ EXTERNALLY

```
0018                        *=$0100          ORG AT HEX 100
0100  58          START 2  CLI              CLEAR INT MASK
0101  A9 FF                 LDA  =$FF
0103  8D 0F 17              STA  C1024E      THIS ENABLES TMR INT
                                             FIRST TIME
0106  20 1F 1F    DISP     JSR  SCANS       THIS IS AN ENDLESS LOOP THAT
0109  4C 06 01             JMP  DISP        DISPLAYS CONTENTS OF F9,FA,FB
```

INTERRUPT SERVICE ROUTINE

```
010C  A9 FF       INTSVC   LDA  =$FF        SET DISPLAY TO 255 CPS PR INT
010E  8D 0F 17             STA  C1024F
0111  20 63 1F             JSR  INCPT
0114  40                   RTI

0115                       *=$17FE          ORG AT IRQ VECTOR
17FE  0C 01       IRQT     .WORD INTSVC     SET = TO INT SERVICE RTN
```

Program 3

THIS EXAMPLE DESCRIBES USE OF
KEYBOARD AND DISPLAY

```
1800                       *=$0200
                  INH      =$F9             LSD'S
                  PTL      =$FA             THESE 3 BYTES ARE DISPLAY BVF
                  PTH      =$FB             MSD'S
0200  58          START 3  CLI
0201  D8                   CLD
0202  20 1F 1F             JSR  SCANS       IF KEY IS DEPRESSED WAIT FOR
0205  D0 F9                BNE  START3      ITS RELEASE
0207  20 1F 1F    DISP1    JSR  SCANS       WAIT FOR KEY DEPRESSED
020A  F0 FB                BEQ  DISP1       WHEN DEPRESSED GO TO VALIDATION
020C  20 1F 1F    VALIDT   JSR  SCANS       THIS USED AS DEBOUNCE
020F  20 6A 1F             JSR  GETKEY      MONITOR UTIL WHICH GETS KEY VAL
0212  C9 15                CMP  =$15        IF MPU IN DEC MODE THEN GET KEY
0214  10 EA                BPL  START3      GETS DECIMAL VALUE A=10
0216  2A                   ROL  A           LEFT JUSTIFY KEY VALUE
0217  2A                   ROL  A
0218  2A                   ROL  A
0219  2A                   ROL  A
021A  A0 04                LDY  =4          SET UP LOOP COUNT=4
021C  2A          V1       ROL  A
021D  26 F9                ROL  INH         SHIFT ALL DIGITS 1 PLACE LEFT
021F  26 FA                ROL  PTL
0221  26 FB                ROL  PTH
0223  88                   DEY
0224  D0 F6                BNE  V1          DO THIS ONE BIT AT A TIME
0226  4C 00 02             JMP  START3      FOR 4 BITS
                           .END
```

END OF MOS/TECHNOLOGY 650X ASSEMBLY VERSION 4
NUMBER OF ERRORS = 0, NUMBER OF WARNINGS = 0

DE MICRO-SOFT BASIC

Sinds een aantal maanden gebruik ik met veel plezier de grote micro-soft basic, zowel voor de voorbereiding van mijn experimenten en de statistische verwerking van de resultaten, als voor de "real-time" besturing van de experimenten. Het enige nadere van deze basic is, dat er nauwelijks enige informatie over verstrekt wordt zodat het een "black box" voor de gebruiker is. Omdat ik méér wilde dan de basic gewoon te gebruiken, was mijn eerste grote basic-programma een intelligente disassembler. Bij deze disassembler kun je voor zover bekend de namen van variabelen en subroutineadressen van tevoren opgeven. In de eerste Pass worden de niet gedefinieerde variabelen en subroutineadressen dan van een symbolische naam voorzien, terwijl in de tweede Paas een printing gemaakt wordt die erg veel op een source-listing lijkt. Dankzij deze source-listing was ik in staat om een aantal problemen die ik met de basic had, op te lossen.

Het spreekt vanzelf, dat ik gaarne met anderen van gedachten wissel over dingen die zij ondekt hebben. Met een aantal mensen in Eindhoven ben ik dan ook bezig om de Pet-basic verder uit te pluizen, om hem geschikt te maken voor mijn KIM-systeem.

Problem 1: You wrote a basic program, telling your basic you don't want sin,cos,atn. After that, you decide to use a sinus in your program. Loading the basic-program again, you tell it: Yes, I want sin,cos,atn, and then load the sourcetext by means of the LOAD-command. Making a LIST your program seems to be loaded incorrect, why? Asking basic: "PRINT SIN(1)" your basic is blown, why?

Answer: Micro-soft tried to make basic programs "relocatable" by writing the program on tape with a tape-identifier FF. Reloading the tape they ask the KIM cassette load program to load a program, tape identifier FF and therefore to put the content of the tape at the adres specified in 17F5-17F8 (KIM-user manual, ch. 4.2 using the tape recorder, loading data from audio tape point 8) .
From the flow-diagram it is clear why this does not work. The first question in the tape load program is: Are the tape-identifier at 17F9 and the tape the same? Because this is true if both are FF, the tape wil be loaded at the adress specified on tape, and <u>nót</u> at the adress specified at 17F5,17F6. Changing the tape identifier on tape

into any value not equal FF results in loading of the tape at the adress specified at 17F5,17F6, and you can fix this by changing at adress 2743 the code A9 FF by A9 00 (9 digit version)

Problem 2 You want to use hypertape, a high speed paper tape, or ....
How?

Answer The dump-routine is called at 275C :JMP 1800 (4C 00 18). You are allowed to change this into: JSR "hypertape", returnig to the basic with RTS. The begin and end adresses of the text buffer are available at 17f5 to 17f8.
The load routine is called at 27A6: JMP 1873 (4C 73 18). Change this into JSR "tape load", you have to specify at 17ED and 17EE the end of the text loaded (that is: the adress of the last byte loaded + 1), or change at 27B8-27BD: LDX $17Ed, LDY $17EE to accomodate basic needs.

Problem 3 You own a video terminal and want to change the "rubout"(5f) into "back space". Even if you found where to fix this, it does not work.

Answer Change the "getline" routine at 2426, and you never will have problems with the rubout code.

```
2420  ca        br2420   dex
2421  10 05              bpl br2428

2423  20 bf 29  br2423   jsr crlf
2426  a2 00     getlin   ldx# $00
2428  20 56 24  br2428   jsr getch from kim
242b  c9 07              cmp# $07          ;bell is a valid char.
242d  f0 14              beq  br2443

242f  c9 0d              cmp# $0d          ;carriage return?
2431  f0 20              beq  br2453

2433  c9 08              cmp# $08          ;rubout?
2435  f0 e9              beq  br2420       ;yes, then skip previous char

2437  c9 7d              cmp# $7d          ;char≥7d, then skip it
2439  b0 ed              bcs  br2428

243b  c9 40              cmp# $40          :cancel line?
243d  f0 e4              beq  br2423

243f  c9 20              cmp# $20          ;char<20, then skip it
2441  90 e5              bcc  br2428
```

(lines to be changed are underlined)

Problem 4 You want to include data in your basic program, maybe even change this data in runtime and want more information on how text is stored in the microsoft basic.

Answer Text in the textbuffer is stored in code. Each line of a basic program results in one line of code. The organisation of this line of code is:

1. The line starts with two bytes containing the memory adress of the next line in the buffer
2. Then two bytes with the (hex) line number
3. A code of one byte for each command or ASCII code (but not equal 0). A byte ≥80 hex represents a command, for instance 83 is equivalent with "DATA", 8E with "REM" .(83 and 8F in the PET). Numbers in your program are stored by means of their ASCII code.
4. A line is terminated by the byte 00
5. The number of bytes of one line may not exceed FF hex, resulting in max. 250 bytes real code and 5 bytes overhead.
6. The text buffer starts with the byte 00, followed by the program lines
7. The text buffer is terminated with the bytes 00,00; the adress of the first 00 is noted at 7A,7B in the Zero Page (Pointer to start of simple variable table).

Problem 5 You wrote your own monitor, after hitting reset, restarting the basic at 0000 4C you cannot RUN programs any longer.

Answer Entering the basic with a stack pointer FD (my monitor) in stead of FF (KIM monitor) results in an error on the stack if the RUN or CLEAR statments are executed.

Problem 6 Is the Micro-soft basic interruptable?

Answer Maybe Yes, I wrote an interrupt driven Telex output routine, and it works until now.

Problem 7 Your basic program is short, you donot use many variables and get the message :" out of memory". (try for instance the program: 10 k = k + 1: print k: gosub 10, after number 26 you get the message "out of memory". Microsoft told you: Gosub nesting is limited only by available memory, did they lie?

Answer On each gosub basic pushes the "return adress" on the stack, just like a JSR in machine language. The same is true for each for-statement Because the stack on the 6500 is limited, the number of gosubs and for-statements is also limited to 26 gosubs or 10 for-statements that are nested. It is a pitty that microsoft didnot seperate this "out of memory" from the "out of memory" of the text/variables buffer.

Problem 8 Is the basic promable?

Answer Yes, but you better rewrite the inialisation part of the basic, starting at $4065. By the way, I think it is not wise to put ROM at the adress 2000-3FFF because most programs are writen in that area. I use a memory-protect on my system and hypertape, so loading the basic is done quickly, and it is not destroiable just as ROM.

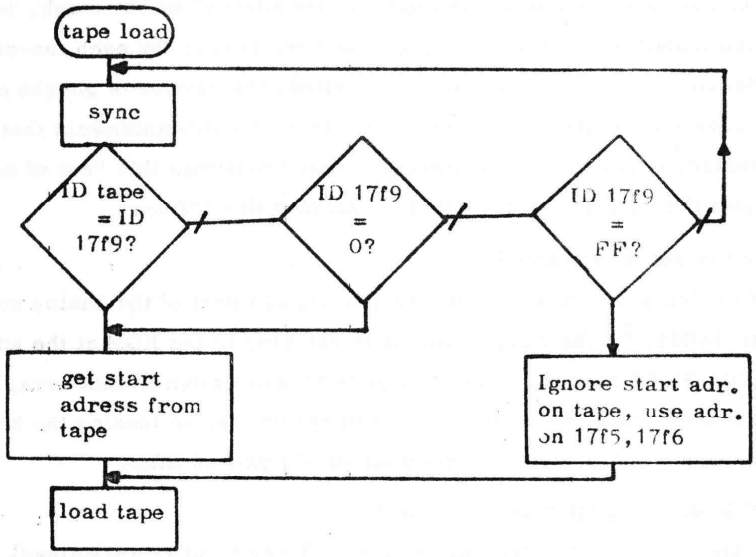
Problem 9 What does the GET statement do?

Answer I wish I know. Try the next program: (User input is underlined)

```
10 GET H$: PRINT "H$="; H$
20 INPUT H$: PRINT "H$="; H$
RUN
X ?? How strange, isn't it?
H$ = How strange, isn't it?
? How strange, isn't it?
EXTRA IGNORED
H$ = How strange
OK
```

Using GET H$ basic asks for input by the keyboard without telling it (using INPUT H$ basic first output a question mark). The first letter typed in however is skiped, answered with two questionmarks, and everything typed thereafter is put into H$, inclusi comma's. By the way, the input call for the GET statement is at 2AE5.

U.O.Schröder
Echternachlaan 161
Eindhoven



flow chart of audio tape load KIM-monitor

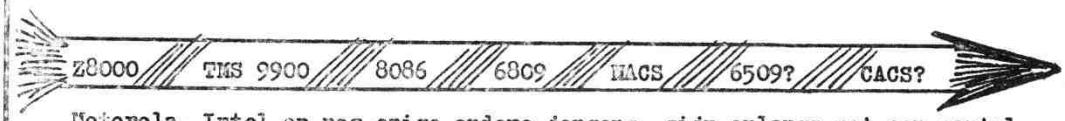
VACATURE GEZOCHT

Ben TH ingenieur, in 1975 afgestudeerd op de afdeling meten en regelen met als onderwerp: de ontwikkeling en realisatie van een onbloedige bloedvat diktemeter op basis van ultrageluid. Daarna was ik op het Instituut voor perceptie onderzoek te Eindhoven werkzaam op het gebied van lees-onderzoek. Voor dit onderzoek verzorgde ik o.a. de hard- en software voor de 6500 microprocessor die mijn experimenten bestuurd. Behalve met digitale technieken kan ik uitstekend overweg met analoge schakelingen voor kleine signalen en/of hoge frekwenties. Met ingang van 1 januari 1979 (of eerder) zoek ik een werkkring als electronicus, zo mogelijk in een medisch/biologische sfeer. Ik stel het zeer op prijs als U mij attent zou willen maken op een passende vacature. U.O.Schröder, Echternachlaan 161, Eindhoven.

KIM

Waar blijft de MCS 6509 ?	Nummer:
	Blad: 1 van 3

MCS 6509 ?



Motorola, Intel en nog enige andere jongens, zijn onlangs met een aantal (naar mijn mening) interessante processoren op de markt gekomen, waarvan, naar ik hoop, misschien het resultaat zal zijn dat ik binnenkort mijn 6502 op de KIM kan vervangen door een 6509 of misschien wel door een CACS (Commodore Advanced Computer System).

Waar gaat het om? Wel, het goede nieuws is dat de chip-fabrikanten hun bestaande 8-bits microcomputer mogelijkheden hebben uitgebreid in 16-bits units door verbetering - en niet vernieuwing - van het bouwkundig concept en de instruktie sets van hun 8-bits processoren. Een paar voorbeelden van deze nieuwe richting zijn de 16-bit Intel H-MOS 8086 en de Motorola twee-chip 6802/6846 en de een-chip 2 MHz 6809. Deze 16-bits microcomputers voeren de complete set instrukties van hun voorgangers uit, plus een krachtige nieuwe set van 16-bit instruktics.

Motorola heeft een voorzichtigere benadering gekozen dan Intel. De 6809 is een tussenstap tussen de zeven-chips 6800 serie en de komende 16-bits MACS (Motorola Advanced Computer System) reeks van VLSI H-MOS high-level language chips.

De volledige 16-bit instruktie set en de 24-bits adresserings ruimte van de MACS zal de 6809 ver overtreffen en geeft een mogelijkheid voor 16 Mega bytes geheugen.

Met zijn 16-bits registers, interne bus en geheugen stacks, geeft de 40-pins 6809 een betere doorvoersnelheid voor interrupt-gedreven toepassingen. De chip voert programma's uit met 50% geheugen besparing en heeft een clock op de chip met crystal en een ready-status pen voor langzamere geheugens.

De 6809 heeft 18 adresserings modes en drie prioriteit-interrupts; de 6800, heeft slechts 6 adresserings modes en één interrupt.

Mogelijkheden die niet op de 6800 aanwezig zijn maar wel op de 6809, zijn: een nieuwe adresserings techniek; nieuw index register, stack pointer en dubbele accumulator; dubbele interne bussen; hardware vermenigvuldigings mogelijkheid; gestructureerde high level language en positie-onafhankelijke coding. Er zijn een paar nieuwe hardware signalen voor de 6809 nodig, maar verder zijn alle 6800 signalen aanwezig op de 6809 en zelfs op dezelfde pennen. Naast de "hardware" stack heeft de 6809 ook een "User" stack. De 6800 had één index register (X), de 6809 heeft er vier: X en Y en de beide stack pointers zijn óók index registers.

Een van de upward-compatability doelstellingen was in staat te zijn bestaande 6800 sourcecoding te kunnen draaien op de 6809, hetgeen mogelijk is met Motorola's "in-house" 6800 naar 6809 cross-assembler.

Datum ingang:	Vervangt:	d.d.:	Ref.:
15 september 1978	-	-	Anton Müller

KIM

Waar blijft de MCS 6509 ?	Nummer:
	Blad: 2 van 3

Motorola was geinteresseerd in de performance van de 6809 op realistische programma segmenten, speciaal die gerelateerd aan door high level language compilers geproduceerde coding. Enige totaal performance uitslagen van een aantal benchmarks worden als volgt samengesteld:

GENORMALISEERDE PERFORMANCE UITSLAGEN

	2MHz 6809	4MHz Z80	1MHz 6800	2MHz 8080
INSTRUKTIES	1.00	1.56	1.72	2.30
BYTES	1.00	1.31	1.52	1.80
TIJD	1.00	2.24	5.34	7.32

(LAGERE SCORES ZIJN BETER)

Nog even iets over de adresseringsmogelijkheden van de 6809. Deze zijn: direct, extended, immediate, relative branches, inherent, long relative branches, 16 variaties van indexed adressering, program counter relative en extended indirecte adressering.
De vergrote indexed adresserings mogelijkheden zijn: auto (post) increment, auto (pre) decrement, indexing met 0,5,8 of 16-bit twee-complement offsets en indexing met een accumulator als offset. Bij al deze modes is dan ook nog indirecte adressering mogelijk. Elk van de vier index registers kan worden gebruikt als basis register voor de indexed adresserings mode.

Naast de normaal te verwachten 16-bit operations zijn er nog een aantal die de moeite van het vermelden waard zijn: LEA - load effective address into pointer register, die toestaat 0, 5, 8 of 16-bit "immediate" waarden op te tellen bij een van de vier registers. LEA kan ook worden gebruikt voor het adresseren van tabellen door de PC-relative mode van adressering te gebruiken. LEA kan ook worden gebruikt om 8 of 16-bits berekende waardes in de accumulators op te tellen bij elk van de vier pointer registers. De enige reden dat ik op deze instruktie zo diep inga is, dat Motorola bij hun vooronderzoek voor de benchmark het meest gebruik hebben gemaakt van de LEA, welke de meest gebruikte van de nieuwe instrukties is. LEA geeft een mogelijkheid om adressen van parameters door te geven aan subroutines. Andere instrukties die ik belangrijk vindt zijn: TFR waarmee elk van de vier registers kunnen worden overgebracht naar een ander register, iets waarbij ik in mijn verhaal over structured programming bij het IF-THEN-ELSE voorbeeld in de fout ging door een TYX instruktie te schrijven die niet bestond. Gelukkig had ik echter een virtuele macro-faciliteit zodat de oplossing simple was: PHA,TYA,TAX,PLA.
Wat ook handige instrukties zijn, zijn: PSHS,PSHU waarmee je in een instruktie alle registers op de hardwarestack of userstack kunt pushen en uiteraard de korresponderende pull's en last but not least de EXG voor het exchangen (uitwisselen) van twee registers.

Wat betreft de multiply: dit is een unsigned! 8-bit by 8-bit multiply met een 16-bit produkt. Ik denk dat hij meer is voor het doen van adresberekeningen dan voor het normale rekenwerk.

Twee extra software interrupts (SWI2 & SWI3) zijn toegevoegd; SWI2 wordt nimmer gebruikt door systeem software en is dus beschikbaar voor de gebruiker.

Datum ingang:	Vervangt:	d.d.:	Ref.:
15 september 1978	-	-	Anton Müller

KIM

WAAR BLIJFT DE MCS 6509 ?	Nummer:
	Blad: 3 van 3

High speed synchronisatie tussen hardware en software is verbeterd op de 6809 door de toevoeging van de SYNC instruktie. Deze instruktie is gelijk aan de wait for interrupt, met dat verschil, dat de registers niet worden gestacked en dat de processor naar de volgende instruktie gaat wanneer een gemaskeerde interrupt opkomt. Door het disabelen van de interrupt en de software in een korte loop te zetten waarin de SYNC is opgenomen is het mogelijk om high speed synchronisatie met hardware devices tot stand te brengen.

Andere subtiele verbeteringen zijn aangebracht in de 6809, zoals het reduceren van het aantal cycles benodigd voor een instruktie in vergelijking met de 6800. Dus bevat de 6809 niet alleen krachtige nieuwe instrukties, maar lopen vele van de bestaande 6800 instrukties sneller, hetgeen ze naar ik vermoed hebben afgekeken van de 650X.

Wat ik niet aardig van ze vindt is dat ze bij hun benchmark de 8080 hebben genomen in plaats van de 8086 en de Z80 in plaats van de Z8000 en de ontbrekende TMS 9900.

Over de 8086 van Intel wil ik kort zijn aangezien die voor ons KIMMERS minder interressant is.

- Direkte adresserings mogelijkheid tot 1 Mega bytes geheugen. Omdat 2^{16} gelijk is aan 64K, is het geheugen onderverdeeld in 64K blokken. Elk blok start op een adres dat deelbaar is door 16. Ook kan de chip 64K besturen voor I/O poorten.
- Assembler language compatible met de 8080/8085.
- 14 Woord, bij 16-bit register set met symetrische operaties.
- 24 Operand adresserings modes
- Bit, byte, woord en block operaties
- 8 en 16 bit signed en unsigned rekenkundige instrukties, zowel binair als decimaal, met inbegrip van vermenigvuldigen en delen. Iets waar de 6809 nog aan kan tippen, maar wat de 6509 wellicht zal hebben, omdat die er nog niet is.
- 5 MHz clock rate
- MULTIBUS™ Compatible system interface

U ziet, de mogelijkheden zijn nog lang niet uitgeput. Binnenkort komen de 64K-bit RAM chips op de markt, of zijn ze er al? Wat voorlopig, wat betreft de miniaturisering van geheugens, het einde van de computersnelheid zal zijn, zijn de Josephson schakelingen van IBM, die een extreem snelle schakeltijd hebben van tussen de 100 en 50 picoseconden en een toegangstijd van 7 nanoseconden. Met deze Josephson schakelingen zijn ze momenteel druk aan het experimenteren in de researchlaboratoria. Het uiteindelijke resultaat van deze experimenten zal vermoedelijk zijn een chip met 1 megabyte geheugen met een toegangstijd van ongeveer 15 nanoseconden. Waar ik ook op zit te wachten is een IBM/370 compatible chip (gewoon 40 pins), met VM/370 in ROM. Niet lachen, want er is al een minifabrikant die een compatible 370 mini op de markt heeft gebracht, dus waarom geen 370 op een chip?

Datum ingang:	Vervangt:	d.d.:	Ref.:
15 september 1978	-	-	Anton Müller

EPROM PROGRAMMERINGS SYSTEEM		Nummer:
BESCHRIJVING		Blad: 1 van 9

Het komt nog wel eens voor, dat je een zelf gemaakt programma eigenlijk niet weer iedere keer vanaf de cassette wil inlezen, maar dat je eigenlijk liever het programma in ROM zou willen hebben. Nu zijn er wel bedrijven die ROM's voor je willen programmeren, maar dat is vaak duur en bovendien kun je nooit meer iets in dat programma veranderen, omdat het er "vast" ingebrand zit.

Om aan dit euvel te ontkomen is de EPROM uitgevonden, de Erasable Programmable Read-Only Memory. Erasble, dus je kunt hem weer uitwissen.

Om hem te kunnen programmeren heb je een programmer nodig, waarvan je in dit artikel het ontwerp vind. Om de programmer vanuit de KIM te sturen, heb je een stuur-programma nodig. Ook dat staat verderop in dit artikel.

Eerst iets over de programmer. In het schema zijn niet de schema's opgenomen voor de vier benodigde voedingsspanningen. (-5V, +5V, +12V en +26V).

Om te programmeren moet je eerst op de adreslijnen het adres aangeven, dan op de datalijnen de data voor dat adres, de CS/WE-pin moet hoog zijn en vervolgens moet gedurende ongeveer een halve milli-sekonde de program-pin +26V zijn. Het kiezen van het adres gebeurt door een hardware-tellertje. Dat bespaart een dure PIA en het geeft de mogelijkheid om ook 2K-EPROM's te programmeren. De data wordt aangevoerd op de A-kant van de PIA, de verdere stuurbits bevinden zich aan de B-kant van de PIA.

Voor de veiligheid is de programmeerpuls begrensd tot maximaal 1 milli-sekonde. Bovendien is een program-not-enable ingebouwd, die er voor zorgt, dat alleen dan een programmeerpuls wordt gegeven, als het programmeer-bit 1 is en het program-not-enable-bit 0 is. Voor de rest lijkt me de zaak voor zichzelf sprekend. Voor iemand met een beetje ervaring in het bouwen van elektronische schakelingen, is het bouwen van de programmer in een weekend gebeurd.

Het stuurprogramma heb ik EPROM genoemd. Het maakt gebruik van een PIA. Als je op de KIM gebruik wilt maken van de vrije 6530, zule je de subroutines PAIN, PACUT en PBOUT moeten aanpassen en verder overal de volgende wijzigingen moeten aanbrengen:

PIAA $8003 wordt SAD $1700
PIAB $8001 wordt SBD $1702

Het in- of output maken gebeurt dan d.m.v. het programmeren van PADD $1701 en PBDD $1703.

Het programma EPROM heeft eigenlijk drie zogenaamde entry points. Dat is ten eerste het begin van het programma. Daar zit het programmeer gedeelte. Het programmeert de EPROM en wel alle 1024 adressen en doet dat 200 keer achter elkaar. Pas dan is de EPROM goed geprogrammeerd. Het programma dat je in de EPROM wilt wegschrijven moet ergens in RAM staan. Je moet er wel voor zorgen

Datum ingang:	Vervangt:	d.d.:	Ref.:
15 september 1978	-	-	Wicher Pattje

EPROM PROGRAMMERINGS SYSTEEM		Nummer:
BESCHRIJVING (vervolg)		Blad: 2 van 9

dat alle adressen van je programma dat straks in de EPROM komt berekend zijn op de adressen die de EPROM straks in je geheugen zal gaan innemen. Uiteraard kun je dit vóórkomen door geen absolute adresseringen te gebruiken. Ook bij het initialiseren van vectoren ($17FA, $17FB en $17FE, $17FF) respektievelijk voor NMI en voor IRQ of BRK, moet je je goed realiseren, dat de adressen van je oorspronkelijke programma in RAM en van je uiteindelijke programma in EPROM niet overeenkomen. Het begin van je RAM-blok schrijf je in SAL, SAH ($17F5, $17F6). Daarna is $0200 GO genoeg om je programma te laten lopen. 200 keer 1024 adressen programmeren duurt ongeveer 2 minuten. Daarna begint het programma de EPROM weer uit te lezen en de data te vergelijken met de data in RAM. Adressen die eventueel fout in de EPROM zijn gekomen worden uitgeprint op een teletype als je die hebt. Het programma komt daarna terug in de KIM-monitor. (Beide andere entry points doen dat ook).

Het tweede entry point is er voor het overschrijven van de EPROM naar RAM. (DUMP). Het kopieert de volledige inhoud van de EPROM naar RAM, te beginnen op een adres aangegeven door SAL, SAH. Op die manier kun je gemakkelijk dingen in EPROM veranderen. Je kopieert de EPROM, verandert in RAM, je neemt een nieuwe EPROM (of je wist eerst de oude) en programmeert daarin het vernieuwde programma.

Het derde entry point (DUMPFF) kijkt of een EPROM wel helemaal goed is gewist. (Dat is: of er overal wel $FF staat). Ook dit deel van het programma kopieert eerst weer de hele EPROM naar RAM, te beginnen bij een adres aangegeven door SAL, SAH en kijkt daarna of alle RAM plaatsen wel $FF zijn. Dit heb ik gedaan opdat mensen zonder teletype of een soortgelijk geval toch vrij gemakkelijk kunnen nagaan of hun EPROM wel helemaal goed is gewist. Ze hebben immers de kopie van de EPROM tot hun beschikking. In mijn eigen versie worden de niet-FF adressen geprint met daarachter de "verkeerde" data van de EPROM.

Het wissen van de EPROM moet gebeuren met ultra-violet licht. Ik gebruik daar zelf een kwik-lampje voor, maar het kan ook met een niet-gecoate TL-buis of met een hoogte zon. Wel oppassen dat je nooit je ogen bloot stelt aan ultra-violet licht!!! Overigens wis ik voor iedereen gratis EPROM's, als de portokosten voor terugsturen maar wel van tevoren worden voldaan.

Adres: Wicher Pattje
Nylandsingel 10
9631 RK ALUARD

Datum ingang:	Vervangt:	d.d.:	Ref.:
15 september 1978	-	-	Wicher Pattje

2708 EPROM PROGRAMMER		Nummer:
HARDWARE		Blad: 3 van 9

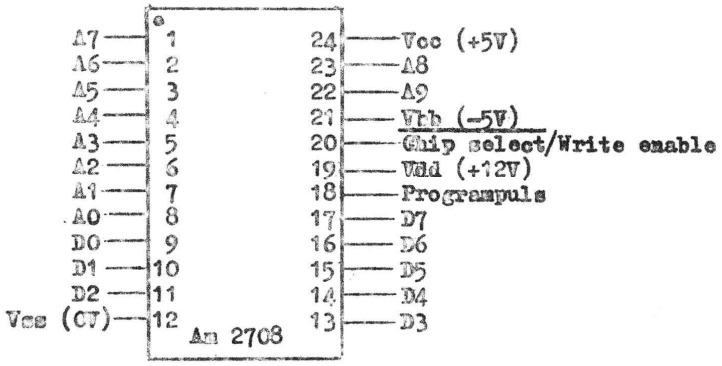


Fig. 1 Aansluitingen EPROM

Adressen A0 tot en met A9 0 of +5 volt. Positieve logica. Adres dient minstens 10 microseconden voor de programmapuls aanwezig te zijn.

Data D0 tot en met D7 0 of +5 volt. Positieve logica. Data dient minstens 10 microseconden voor de programmapuls aanwezig te zijn.

Chip select/Write enable. Indien 0 volt dan wordt de ROM uitgelesen, anders +5 volt. Bij programmering een +12 volt, dient minstens 10 microseconden voor de programmapuls te gebeuren.

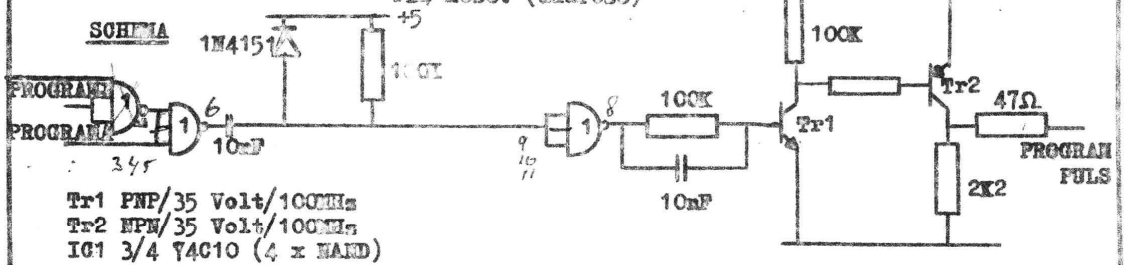
Programmapuls van 0 naar +25 volt en weer terug. Deze puls mag nooit langer dan 1 ms duren. Voordat een nieuwe puls wordt aangeboden dienen eerst alle andere adressen gepulst te zijn. Voor programmering is 100 msec. voor de som van alle programmapulsen op een adres voldoende. De programmeringstijd is dus altijd minstens gelijk aan 100 x 1K = 100 x 1024 = 102400 msec, dus minstens 100 sec.

PIA - Aansluitingen.

De aansluitingen op de PIA zijn als volgt:

PA0 t/m PA7 aan D0 t/m D7
PB0 Programm
PB1 Clock address
PB2 Programb (program enable)
PB3 Write enable
PB4 Reset (address)

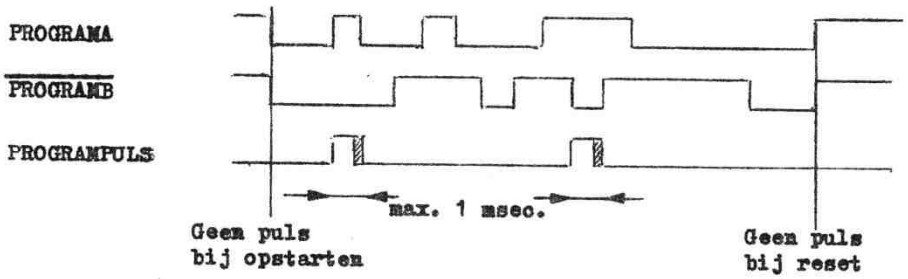
SCHEMA

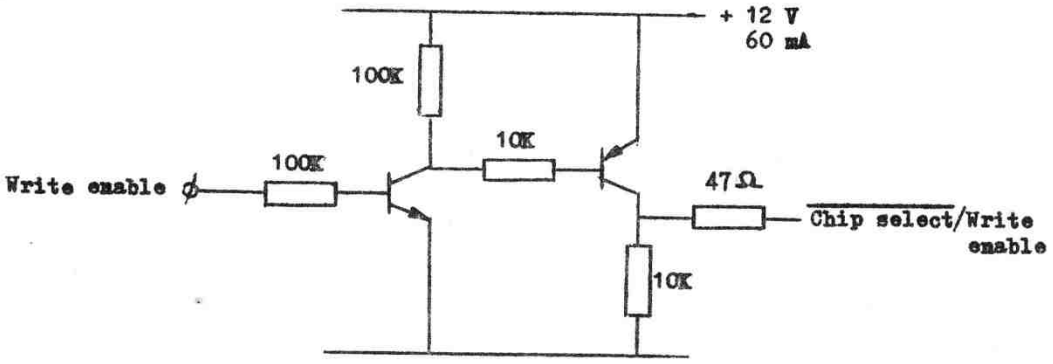


Tr1 PNP/35 Volt/100MHz
Tr2 NPN/35 Volt/100MHz
IC1 3/4 74C10 (4 x NAND)

Datum ingang:	Vervangt:	d.d.:	Ref.:
15 september 1978	-	-	W.R. Pattje

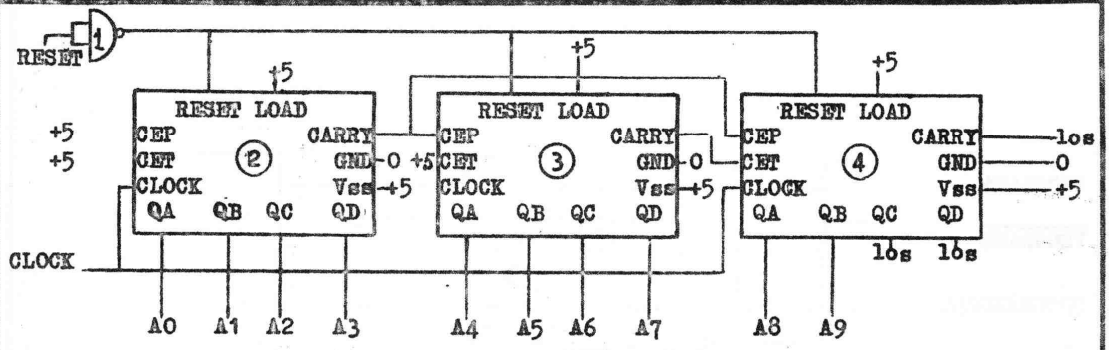
2708 EPROM PROGRAMMER		Nummer:
HARDWARE (vervolg)		Blad: 4 van 9



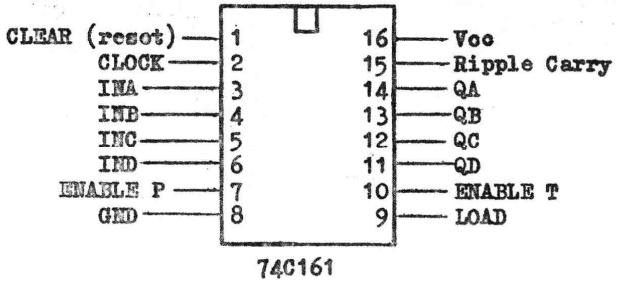


Datum ingang:	Vervangt:	d.d.:	Ref.:
15 september 1978	-	-	W.R. Pattje

2708 EPROM PROGRAMMER	Nummer:
HARDWARE (vervolg)	Blad: 5 van 9



IC1 1/4 74C10 (4 x NAND)
IC2 74C161 (16 deler)
IC3 74C161 (16 deler)
IC4 74C161 (16 deler)



Datum ingang:	Vervangt:	d.d.:	Ref.:
15 september 1978	-	-	W.R. Pattje

EPROM PROGRAMMERINGS SYSTEEM (SOFTWARE)	Nummer:
EPROM KIM SOFTWARE LIBRARY 65XX-1.0 PAGE 01	Blad: 6 van 9

```
0010: 035B              PL      *       $0000
0020: 035B              PH      *       $0001   RAM ADRES-POINTER
0030: 035B              TELA    *       $0002
0040: 035B              TELB    *       $0003   1K-TELLERS
0050: 035B              TELC    *       $0004   200-KEER TELLER
0060: 035B              SAL     *       $17F5
0070: 035B              SAH     *       $17F6   BEGIN 1K RAM-BLOK
0080: 035B              CRB     *       $8000   CONTROLE REGISTER B
0090: 035B              PIAB    *       $8001   DATA/DIRECTIE-REG B
0100: 035B              CRA     *       $8002   CONTROLE REGISTER A
0110: 035B              PIAA    *       $8003   DATA/DIRECTIE-REG A
0120: 035B              START   *       $1C00   KIM-ENTRYPOINT
0130: 035B              PRTBYT  *       $1E3B   KIM PRINT BYTE SUBR
0140: 035B              OUTSP   *       $1E9E   KIM PRINT SPATIE SUBR
0150: 0200              EPROM   ORG     $0200
0160: 0200 D8                   CLD             VOOR ALLE ZEKERHEID
0170: 0201 20 9E 02             JSR     PAOUT   MAAK PIAA EN PIAB OUTPUTS
0180: 0204 20 AB 02             JSR     PBOUT
0190: 0207 A9 00                LDAIM   $00
0200: 0209 85 04                STA     TELC    CLEAR 200 KEER TELLER
0210: 020B 20 CC 02     EPROMA  JSR     BEGIN   INITIALISEER
0220: 020E 20 0C 03     EPROMB  JSR     SETDAT  ZET DATA VAN PL,PH IN EPROM
0230: 0211 20 E5 02             JSR     INCADR  VERHOOG EPROM-ADRES
0240: 0214 20 6A 02             JSR     INCK    VERHOOG PL,PH EN TEST OP 1024 KB
0250: 0217 90 F5                BCC     EPROMB  CY=0, DAN NOG GEEN 1024 ADRESSEN
0260: 0219 E6 04        EPROMC  INC     TELC
0270: 021B A5 04                LDA     TELC
0280: 021D C9 B8                CMPIM   $B8     AL 200 KEER GEHAD?
0290: 021F D0 EA                BNE     EPROMA  NEE, DAN TERUG
0300: 0221 20 90 02     EPROMD  JSR     PAIN    LEES EPROM WEER UIT
0310: 0224 20 CC 02             JSR     BEGIN   INITIALISEER
0320: 0227 20 24 03     EPROME  JSR     GETDAT  HAAL DATA UIT EPROM
0330: 022A 20 E5 02             JSR     INCADR  VERHOOG EPROM-ADRES
0340: 022D 20 6A 02             JSR     INCK    VERHOOG PL,PH EN TEST OP 1024 KB
0350: 0230 90 F5                BCC     EPROME  CY=0, DAN NOG GEEN 1024 ADRESSEN
0360: 0232 4C 00 1C     TERUG   JMP     START   SPRING ERUIT
0370:
0380: 0235 20 44 02     DUMP    JSR     DUMPDA  DUMP DE EPROM IN RAM
0390: 0238 4C 32 02             JMP     TERUG   EN SPRING ERUIT
0400:
0410: 023B 20 44 02     DUMPFF  JSR     DUMPDA  DUMP EPROM IN RAM
0420: 023E 20 79 02             JSR     TESTFF  KIJK OF OVERAL $FF IN STAAT
0430: 0241 4C 32 02             JMP     TERUG   SPRING ERUIT
0440:
0450: 0244 D8           DUMPDA  CLD             VOOR DE ZEKERHEID
0460: 0245 20 90 02             JSR     PAIN    MAAK PIAA INPUTS
0470: 0248 20 AB 02             JSR     PBOUT   MAAK PIAB OUTPUTS
0480: 024B 20 CC 02             JSR     BEGIN   INITIALISEER
0490: 024E 20 5A 02     DUMPA   JSR     ROMRAM  DUMP EPROMADR IN RAMADRES
0500: 0251 20 E5 02             JSR     INCADR  VERHOOG EPROM-ADRES
```

Datum ingang:	Vervangt:	d.d.:	Ref.:
15 september 1978	-	-	Wicher Pattje

EPROM PROGRAMMERINGS SYSTEEM (SOFTWARE)	Nummer:
EPROM KIM SOFTWARE LIBRARY 65XX-1.0 PAGE 02	Blad: 7 van

```
0510: 0254 20 6A 02          JSR   INCK    VERHOOG PL,PH EN TEST OP 1024 KB
0520: 0257 90 F5             BCC   DUMPA   CY=0, DAN NOG GEEN 1024 ADRESSEN
0530: 0259 60                RTS           KLAAR
0540:
0550: 025A A0 00     ROMRAM  LDYIM $00
0560: 025C A9 F7             LDAIM $F7
0570: 025E 2D 01 80          AND   PIAB
0580: 0261 8D 01 80          STA   PIAB    CS/WE IS NU LAAG
0590: 0264 AD 03 80          LDA   PIAA    HAAL DATA UIT EPROM
0600: 0267 91 00             STAIY PL      NAAR RAM-LOKATIE
0610: 0269 60                RTS
0620:
0630: 026A 20 54 03  INCK    JSR   INCP    VERHOOG PL,PH
0640: 026D E6 02             INC   TELA
0650: 026F 18                CLC
0660: 0270 D0 06             BNE   INKA    VERHOOG 1024 KEER TELLERS
0670: 0272 E6 03             INC   TELB
0680: 0274 A5 03             LDA   TELB
0690: 0276 C9 04             CMPIM $04     ALS CY=1, DAN NU 1024 ADRESSEN
0700: 0278 60        INKA    RTS           ALS CY=0, DAN NOG NIET KLAAR
0710:
0720: 0279 20 CC 02  TESTFF  JSR   BEGIN   SAL,SAH NAAR PL,PH
0730: 027C A0 00     TESTA   LDYIM $00
0740: 027E B1 00             LDAIY PL      HAAL RAM-LOKATIE
0750: 0280 C9 FF             CMPIM $FF     IS $FF? ?
0760: 0282 F0 03             BEQ   TESTB   ZO JA, GA DOOR
0770: 0284 20 3A 03          JSR   PRINT   PRINT FOUTE RAM ADRES
0780: 0287 20 E5 02  TESTB   JSR   INCADR  VERHOOG EPROM-ADRES
0790: 028A 20 6A 02          JSR   INCK    VERHOOG PL,PH EN TEST OP 1024 KB
0800: 028D 90 ED             BCC   TESTA   ALS CY=0, DAN NOG NIET KLAAR
0810: 028F 60                RTS
0010:
0020: 0290 A9 00     PAIN    LDAIM $00
0030: 0292 8D 02 80          STA   CRA     KIES DIREKTIE-REG A
0040: 0295 8D 03 80          STA   PIAA    ALLEMAAL INPUTS
0050: 0298 A9 04     PAINA   LDAIM $04
0060: 029A 8D 02 80          STA   CRA     KIES DATAREG A
0070: 029D 60                RTS
0080:
0090: 029E A9 00     PAOUT   LDAIM $00
0100: 02A0 8D 02 80          STA   CRA     KIES DIREKTIE-REG A
0110: 02A3 A9 FF             LDAIM $FF
0120: 02A5 8D 03 80          STA   PIAA    ALLEMAAL OUTPUTS
0130: 02A8 4C 98 02          JMP   PAINA
0140:
0150: 02AB A9 00     PBOUT   LDAIM $00
0160: 02AD 8D 00 80          STA   CRB     KIES DIREKTIE-REG B
0170: 02B0 A9 FF             LDAIM $FF
0180: 02B2 8D 01 80          STA   PIAB    ALLEMAAL OUTPUTS
0190: 02B5 A9 04             LDAIM $04
```

Datum ingang:	Vervangt:	d.d.:	Ref.:
15 september 1978	-	-	Wicher Pattje

EPROM PROGRAMMERINGS SYSTEEM (SOFTWARE)	Nummer:
EPROM KIM SOFTWARE LIBRARY 65XX-1.0 PAGE 03	Blad: 8 van 9

```
0200: 02B7 8D 00 80          STA   CRB     KIES DATA-REG B
0210: 02BA 60                RTS
0220:
0230: 02BB A9 10      RESADR LDAIM $10
0240: 02BD 0D 01 80          ORA   PIAB
0250: 02C0 8D 01 80          STA   PIAB    RESET-BIT HOOG
0260: 02C3 A9 EF             LDAIM $EF
0270: 02C5 2D 01 80          AND   PIAB
0280: 02C8 8D 01 80          STA   PIAB    RESET-BIT LAAG
0290: 02CB 60                RTS           ADRES IN EPROM OP NUL
0300:
0310: 02CC AD F5 17  BEGIN   LDA   SAL     SAL,SAH NAAR PL,PH
0320: 02CF 85 00             STA   PL
0330: 02D1 AD F6 17          LDA   SAH
0340: 02D4 85 01             STA   PH
0350: 02D6 A9 00             LDAIM $00
0360: 02D8 85 02             STA   TELA
0370: 02DA 85 03             STA   TELB    RESET TELLERS VOOR 1K
0380: 02DC A9 04             LDAIM $04
0390: 02DE 8D 01 80          STA   PIAB    PROGRAM NOT ENABLE = HOOG!!
0400: 02E1 20 BB 02          JSR   RESADR  RESET ADRES-COUNTER
0410: 02E4 60                RTS
0420:
0430: 02E5 A9 02      INCADR LDAIM $02
0440: 02E7 0D 01 80          ORA   PIAB
0450: 02EA 8D 01 80          STA   PIAB    ADRES-COUNTER-BIT HOOG
0460: 02ED A9 FD             LDAIM $FD
0470: 02EF 2D 01 80          AND   PIAB
0480: 02F2 8D 01 80          STA   PIAB    ADRES-COUNTER-BIT LAAG
0490: 02F5 60                RTS           ADRES IN EPROM NU MET 1 VERHOOGD
0500:
0510: 02F6 A9 FB      PULS   LDAIM $FB
0520: 02F8 2D 01 80          AND   PIAB
0530: 02FB 8D 01 80          STA   PIAB    PROGRAM PULS BEGINT
0540: 02FE A2 58             LDXIM $58
0550: 0300 CA         PULSA  DEX
0560: 0301 D0 FD             BNE   PULSA   WACHT 500 MICRO-SEC
0570: 0303 A9 04             LDAIM $04
0580: 0305 0D 01 80          ORA   PIAB
0590: 0308 8D 01 80          STA   PIAB    EINDE PROGRAM PULS
0600: 030B 60                RTS
0610:
0620: 030C A0 00      SETDAT LDYIM $00
0630: 030E B1 00             LDAIY PL      HAAL DATA VAN PL,PH
0640: 0310 8D 03 80          STA   PIAA    NAAR PIA
0650: 0313 A9 09             LDAIM $09
0660: 0315 0D 01 80          ORA   PIAB
0670: 0318 8D 01 80          STA   PIAB    CS/WE EN PROGRAM HOOG!
0680: 031B A2 02             LDXIM $02
0690: 031D CA         WAIT   DEX
```

Datum ingang:	Vervangt:	d.d.:	Ref.:
15 september 1978	-	-	Wicher Pattje

EPROM PROGRAMMERINGS SYSTEEM (SOFTWARE)	Nummer:
EPROM KIM SOFTWARE LIBRARY 65XX-1.0 PAGE 04	Blad: 9 van 9

```
0700: 031E D0 FD          BNE   WAIT   WACHT 10 MICRO-SEC
0710: 0320 20 F6 02       JSR   PULS   SCHRIJF DATA IN EPROM
0720: 0323 60             RTS
0730:
0740: 0324 A0 00    GETDAT LDYIM $00
0750: 0326 B1 00           LDAIY PL     HAAL DATA VAN PL,PH
0760: 0328 AA              TAX          NAAR X-REGISTER
0770: 0329 A9 F7           LDAIM $F7
0780: 032B 2D 01 80        AND   PIAB
0790: 032E 8D 01 80        STA   PIAB   CS/WE LAAG
0800: 0331 EC 03 80        CPX   PIAA
0810: 0334 F0 03           BEQ   BACK
0820: 0336 20 3A 03        JSR   PRINT  PRINT "FOUTE" RAM-ADR
0830: 0339 60       BACK   RTS
0840:
0850: 033A A5 01    PRINT  LDA   PH     PRINT RAM-ADR DAT FOUT
0860: 033C 20 3B 1E        JSR   PRTBYT IN DE EPROM IS GEKOMEN
0870: 033F A5 00           LDA   PL
0880: 0341 20 3B 1E        JSR   PRTBYT
0890: 0344 20 9E 1E        JSR   OUTSP
0900: 0347 AD 03 80        LDA   PIAA
0910: 034A 20 3B 1E        JSR   PRTBYT
0920: 034D 20 9E 1E        JSR   OUTSP
0930: 0350 20 9E 1E        JSR   OUTSP
0940: 0353 60              RTS
0950:
0960: 0354 E6 00    INCP   INC   PL
0970: 0356 D0 02           BNE   INA
0980: 0358 E6 01           INC   PH
0990: 035A 60       INA    RTS
```

Datum ingang:	Vervangt:	d.d.:	Ref.:
15 september 1978	-	-	Wicher Pattje

STRUCTURED PROGRAMMING		Nummer:
Deel: 1	Definities en overzicht	Blad: 1 van 12

Structured Programming (1)

Enige definities en een overzicht. (19)

Een van de eerste zaken die besproken moeten worden zijn: "Wat structured programming is en wat de doelstellingen ervan zijn". Er zijn ontzettend veel definities, waarvan David Gries in zijn brief aan de Communications of the ACM (November 1974), een opsomming geeft.

Hieronder volgt een uittreksel uit die brief, die een grote verscheidenheid aan definities omtrent structured programming bevat.

"Structured Programming Definities".

De verdedigende instelling omtrent structured programming komt gedeeltelijk voor uit het feit dat mensen er niet van houden dat hun verteld wordt dat zij niet weten hoe zij hun werkzaamheden efficient kunnen doen. Ik moet ook toegeven dat een deel van deze instelling komt van de manier waarop de literatuur is geschreven en van het feit dat structured programming nooit volledig is gedefiniëerd. Uit hun verband gehaald en verkeerd belicht, zien sommige definities er in de literatuur gewoon stom uit (bijv. structured programming is het programmeren zonder GOTO's) en sommigen lijken op loze kreten waar niemand iets mee van doen wil hebben en ik kan mij voorstellen waarom sommigen kijken alsof ze water zien branden wanneer zij ze lezen. Als voorbeelden, heeft Denning (16) ontdekt dat de volgende punten de meeste indrukken weergeven die mensen over structured programming hebben:

1. Het is een terugkeer naar "common sense".
2. Het is de algemene methode waarmee onze leidende programmeurs programmeren.
3. Het is programmeren zonder het gebruik van GOTO statements.
4. Het is het proces van het besturen van het aantal interakties tussen een gegeven lokale taak en zijn omgeving zo dat het aantal interakties : enige lineaire funktie van enige parameter of parameter van de taak is.
5. Het is top-down programmeren.

Ik heb ook de volgende kommentaren in de literatuur gevonden:

6. De structured programming theorie houdt zich bezig met het converteren van arbitrair grote en complexe flowcharts in standaard vormen zodat zij kunnen worden voorgesteld door iteratie en nesting van een klein aantal basis en standaard besturings logika strukturen (deze zijn gewoonlijk sequencing, alternation en iteration)(10).
7. Structured programming is een manier voor het organiseren en coderen van programma's, die de programma's gemakkelijker begrijpbaar en modificeerbaar maken. (11)

Datum ingang:	Vervangt:	d.d.:	Ref.:
14 september 1978	-	-	Anton Müller

STRUCTURED PROGRAMMING		Nummer:
Deel: 1	Struktuurtheorie/afspraken	Blad: 2 van 12

8. Het doel van structured programming is komplexiteit te besturen door theorie en discipline. (12)

9. Structured programming moet niet gekarakteriseerd worden door de afwezigheid van GOTO's, maar bij de aanwezigheid van struktuur. (12)

10. Een hoofd funktie van het strukturceren van een programma is het mogelijk maken van een korrektheids proef. (3)

11. Een fundamenteel concept van structured programming is een proef van korrektheid. (13)

12. Structured programming staat verifikatie van de korrektheid toe van alle stappen in het ontwerp proces en leidt dus automatisch tot een self-verklarende en self-verdedigende programmerings stijl. (14)

13. Structured programming is geen "Panacea", het bestaat werkelijk uit een formele notatie voor ordelijk denken (een kenmerk dat niet algemeen aanwezig is bij programmeurs). Het is een discipline die moet worden aangeleerd en continue moet worden versterkt door bewuste inspanning. Het is de moeite waard. (15)

Bij elkaar genomen, geven zij een goed algemeen overzicht van het onderwerp.

Struktuur theorie en afspraken. (17)

Structured programming is gebaseerd op de wiskundig bewezen Struktuur Theorie die inhoudt dat elk "goed" programma (een programma met één ingang en één uitgang) gelijk is aan een programma dat als besturings strukturen alleen bevat:

- Volgorden van twee of meer handelingen (move, add, ...)
- Voorwaardelijke sprong naar een van twee handelingen en terugkeer (IF y THEN b ELSE c)
- Herhaling van een handeling zolang een voorwaarde waar is (DO WHILE)

Aanvullende besturings logika strukturen.

Hoewel alle programma's geschreven kunnen worden met slechts gebruikmaking van bovengenoemde drie basis strukturen, is het soms handig er nog een paar meer te gebruiken:

- Herhaling van een handeling totdat een voorwaarde waar is (DO UNTIL)
- Selecteer een handeling uit vele, gebaseerd op het aftesten van de waarde van een variabele (CASE)

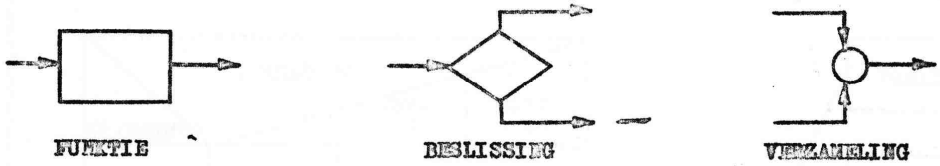
De selektie van de te gebruiken besturings strukturen zal in zekere mate afhankelijk zijn van de te gebruiken programmeer taal. Zo is het bijvoorbeeld in 6502 Assembler Language soms niet te vermijden om de DO UNTIL struktuur te omseilen, zonder je in allerlei bochten te wringen. Bij High Level Languages (HLL) is de DO UNTIL beslist niet nodig. De CASE is alleen maar handig om in bepaalde gevallen nested-IF's te voorkomen, doch is niet beslist noodzakelijk.

Alvorens verder te gaan zal ik nu eerst wat plaatjes laten zien van de behandelde strukturen, zowel in stroomdiagram symbolen als in Nassi/Schneiderman diagrammen (18).

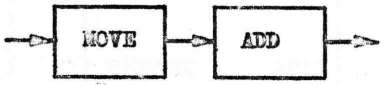
Datum ingang:	Vervangt:	d.d.:	Ref.:
14 september 1978	-	-	Anton Müller

STRUCTURED PROGRAMMING		Nummer:
Deel: 1	Flowcharts/iteration graphs	Blad: 3 van 12

We kunnen een programma weergeven in de vorm van een stroomdiagram. Een stroomdiagram is een richting-aangevend diagram dat de loop beschrijft van de besturings uitvoer van het programma. Voor ons doel zullen we stroomdiagrammen beschouwen als drie types knooppunten, genaamd funktie knooppunt, beslissings knooppunt en verzamelings knooppunt. De bovenste en onderste lijnen van beslissings knooppunten worden altijd verondersteld respektievelijk waar en niet-waar te zijn.

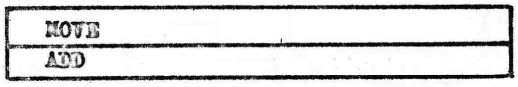


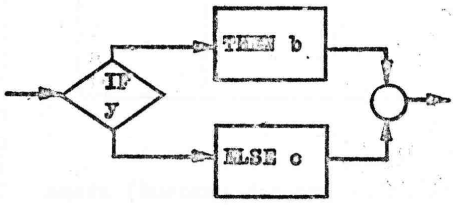
Stroomdiagram



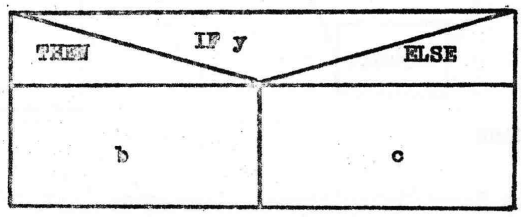
SEQUENCE

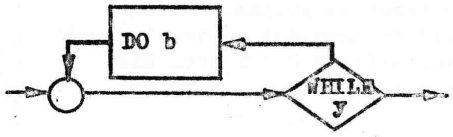
Nassi/Schneiderman-diagram



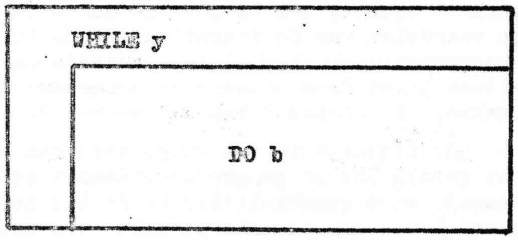


IFTHENELSE



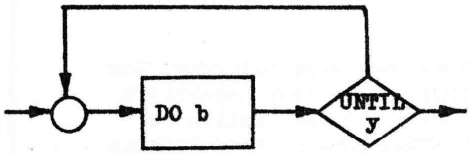


DOWHILE

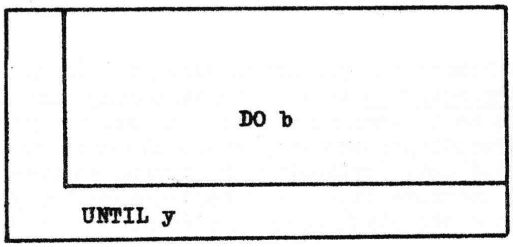


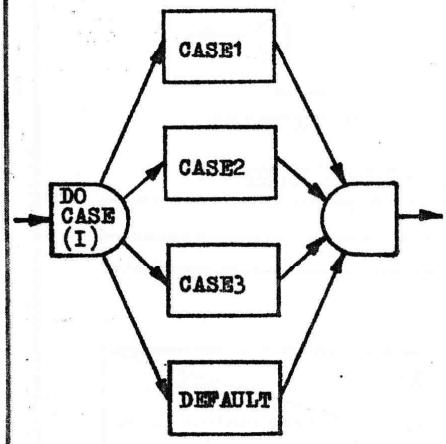
Datum ingang:	Vervangt:	d.d.:	Ref.:
14 september 1978	-	-	Anton Müller

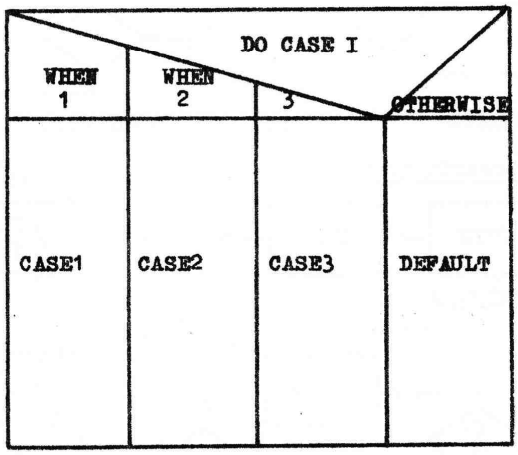
STRUCTURED PROGRAMMING		Nummer:
Deel: 1	Flowcharts/iteration graphs	Blad: 4 van 12



DOUNTIL







CASE

De Nassi/Schneiderman diagrammen (ook wel "iteration graphs" genoemd) staan uitvoerig beschreven in de ACM SIGPLAN Notices, vol. 8, no. 8, pagina's 12 t/m 26, van Augustus 1973. Bij alle voorbeelden die ik verder in mijn verhaal opneem, zal ik gebruik maken van deze iteration graphs. Een van de voordelen van de iteration graphs is, dat zij dwingen tot gestructureerd programmeren. Ik beveel deze methode dan ook ten zeerste bij U aan. Niet alleen helpt deze notatie de programmeur om op een ordelijke manier te denken, zij forceert hem ook om het te doen.

De afwezigheid van enige vorm van de GOTO of branch statement heeft tot gevolg dat de programmeur zonder ze moet werken: een taak die in toenemende mate gemakkelijker wordt bij het praktisch doen. Programmeurs die eerst leren programma's te ontwerpen met deze symbolen, zullen nooit de slechte gewoontes krijgen die andere stroomdiagram-notatie-systemen toestaan.

Omdat er niet meer dan vijftien tot twintig symbolen op een enkel blaadje A4 kunnen worden getekend, moet de programmeur zijn programma op een bepaalde manier in moten hakken. Hoe? Daar kom ik in een volgend deel "Programma ontwerp methode" op terug.

Datum ingang:	Vervangt:	d.d.:	Ref.:
14 september 1978	-	-	Anton Müller

STRUCTURED PROGRAMMING		Nummer:
Deel: 1	Keuze van de programmeertaal	Blad: 5 van 12

De ontwikkeling van een methodologie is noodzakelijk om complexe problemen aan te kunnen. Het oplossen van een probleem, dat resulteert in de ontwikkeling van uitvoerbare programma's, is een complex probleem. Om deze reden is het belangrijk dat een methodologie wordt ontwikkeld die helpt programma ontwikkelings problemen op een systematische manier op te lossen. We moeten ons zelf beperken met programma's te werken, die we volledig begrijpen en intellectueel onder controle hebben.

De methode om complexiteit te besturen is het programma zodanig te structureren dat het sequentieel kan worden gelezen, in kleine segmenten, waarbij elk segment op zich weer sequentieel is georganiseerd en alle besturingspaden zichtbaar zijn.

Teneinde bovenstaande methodologie toe te kunnen passen, moet de taal waarmee wij programmeren bepaalde faciliteiten bieden. De basis besturingsstrukturen IFTHENELSE en DOWHILE moeten minimaal aanwezig zijn, terwijl DOUNTIL en CASE wenselijk doch niet beslist noodzakelijk zijn.

Voor de KIM zijn er zulke talen, onder andere XPLO, PASCAL, FOCAL en BASIC. Last but not least, zou ik ook nog als mogelijkheid willen noemen Assembler Language, doch de Assembler moet dan het liefst voorzieningen hebben om met behulp van MACRO's gecondition eerd coding te genereren.

Zuiver genomen zijn deze macro's er alleen maar om het de programmeur gemakkelijker te maken en te voorkomen dat hijzelf compare en branch instrukties moet schrijven. Hebben we deze macro-faciliteit niet (en die hebben we nog niet) dan kunnen we toch wel gestruktureerde programma's in Assembler Language schrijven, waarbij we net doen alsof we wèl een macro-faciliteit als hiervoor omschreven hebben. Na het schrijven van het programma in symbolische taal met IFTHENELSE, DOUNTIL, DOWHILE en CASE macro's, gaan we deze macro's met de hand expanderen in normale symbolische instrukties, volgens een gestandaardiseerde methode, waarna we dit geheel aan de Assembler aanbieden, of zo we die ook niet tot onze beschikking hebben, het geheel handmatig omzetten in machine taal.

De keuze van de te gebruiken programmeertaal is geheel afhankelijk van onze mogelijkheden, de mogelijkheden van de taal, de mogelijkheden van onze computer en de mogelijkheden van ons budget.

XPLO is een subset van PL/1, een uitstekende taal om gestruktureerd in te programmeren, doch aan de compiler op zich mankeert nogal het een en ander, zelfs zoveel dat de leverancier ARESCO het pakket teruggegeven heeft aan de auteur, omdat hij er niet voldoende support op kan geven. Desondanks onderzoeken Siep de Vries en ik momenteel XPLO, teneinde er een bruikbaar pakket van te maken.

PASCAL is een block-gestruktureerde taal, uitstekend geschikt om gestruktureerd in te programmeren. PASCAL voor de 6502 is echter pas volgend jaar beschikbaar.

Blijven over FOCAL en BASIC. Ondanks het feit dat dit aardige taaltjes zijn, heb ik er iets tegen, maar dat zal wel aan mij liggen. Als U wilt kunt U er mee gestruktureerd programmeren, waarbij FOCAL dan nog beter geschikt is als BASIC.

STRUCTURED PROGRAMMING		Nummer:
Deel: 1	Voorbeeld van DOUNTIL	Blad: 6 van 12

In de voorbeelden die ik verder in mijn verhaal opneem, zal ik gebruik maken van Assembler Language, MICRO ADE formaat (20), voor de KIM/6502, er vanuitgaande dat deze een MACRO-faciliteit zoals hiervoor omschreven heeft, die hij dus duidelijk niet heeft. Ik ben geen voorstander van Assembler Language voor gestruktureerd programmeren, maar het is momenteel de enige mogelijkheid. Bovendien ben ik van mening dat 99% van onze leden deze taal gebruikt.

De structured programming macro's die ik gebruik zijn afgeleid uit een brochure van IBM (21), waarvan ik de expansie van de macro's heb aangepast aan het KIM/MICRO-ADE-formaat.

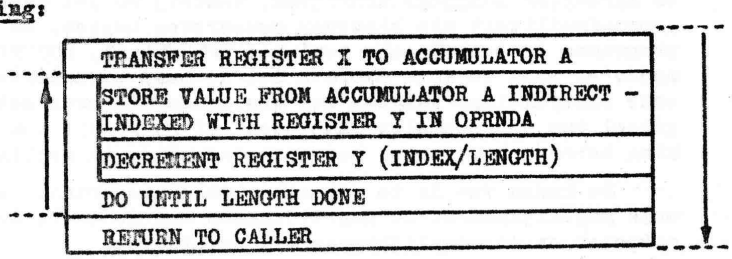
De belangrijkste reden waarom er (nog) geen macro-faciliteit voor de KIM Assembler Language is, is naar mijn mening het ontbreken van een betaalbaar direkt toegankelijk massageheugen (bijv. floppy disk) met een daarop gezet operating systeem. Hoewel er onlangs een floppy disk drive met interface en operating system voor de KIM (FODS van HDE) op de markt is verschenen, zal het gezien de prijs ($ 1995,-) nog wel enige tijd duren voor we ons dit soort zaken kunnen permiteren.

Na al deze theorie wil ik dit deel nu besluiten met een paar programma voorbeelden, die bewust eenvoudig zijn gehouden.

Voorbeeld 1: Gegeven: Een waarde in register X, een lengte van 1 t/m 256 in register Y (00 = 1, FF = 256), beginadres op locaties OPRNDA = 0000 (low) en OPRNDA +01 = 0001 (high).

Gevraagd Ontwerp een programma voor het initialiseren van een stuk geheugen met behulp van bovenstaande gegevens.

Oplossing:



Gestruktureerde coding:

```
SRCLRM TXA                          TRANSFER X TO A
       DOUNTIL ((Y),EQ,DONE,IM)     DOUNTIL LENGTH DONE
          STAIY OPRNDA              STORE VALUE
          DEY                       DECR INDEX/LENGTH
       ENDDO
       RTS                          RETURN TO CALLER
```

Datum ingang:	Vervangt:	d.d.:	Ref.:
14 september 1978	-	-	Anton Müller

Gegomereerde coding:

```
                        ORG   $0200
0200 8A         SRCLRM  TXA                     TRANSFER X TO A
                        DOUNTIL ((Y),EQ,DONE,IM) DOUNTIL LENGTH DONE
+0201 D0 06               BNE LOOO1B            UNCONDITIONAL BRANCH TO
+0203 F0 04               BEQ LOOO1B            EXECUTION LOGIC
+0205 C0 FF     LOOO1A    CPYIM DONE            DO UNTIL CONDITION
+0207 F0 07               BEQ LOOO1X            IS TRUE
+0209           LOOO1B    *
 0209 91 00               STAIY OPRNDA          STORE VALUE
 020B 88                  DEY                   DECR INDEX/LENGTH
                        ENDDO
+020C D0 F7               BNE LOOO1A            UNCONDITIONAL JUMP BACK TO
+020E F0 F5               BEQ LOOO1A            CONDITION TEST
+0210           LOOO1X    *                     DO EXIT POINT
 0210 60                 RTS                   RETURN TO CALLER
```

Bij deze uitgewerkte voorbeelden horen mog de volgende definities:

```
OPRNDA * $0000
DONE   * $FF
```

De statements met een + ervoor zijn de expansie van de macro-instruktie s.

Zonder de structured programming macro's zouden we het gevraagde programma als volgt coderen:

```
                ORG   $0200
0200 8A         SRCLRM TXA
0201 91 00      LOOO1A STAIY OPRNDA
0203 88                DEY
0204 C0 FF             CPYIM DONE
0206 D0 F9             BNE   LOOO1A
0208 60                RTS
```

U ziet, het gestruktureerde programma kost dus 8 bytes meer in dit voorbeeld, hetgeen U zich mijnsinziens niet druk over hoeft te maken. De prijs van een byte is slechts vijf cent vandaag de dag en we kunnen er ruim 60.000 van aan onze KIM knopen en dat is heel wat.

Even terugkomen op het voorbeeld. Ik gebruik hier de DOUNTIL omdat het schoonmaken van een geheugenplaats minimaal één keer moet gebeuren en maximaal 256 keer. De DOWHILE biedt hier geen uitkomst, tenzij we een foef toepassen en ervan uitgaan dat de DOUNTIL niet bestaat. Het programma komt er dan als volgt uit te zien:

```
                SRCLRM TXA                      TRANSFER X TO A
                       STAIY OPRNDA             STORE VALUE
                       DEY                      DECREMENT INDEX/LENGTH
                       DOWHILE ((Y),NE,DONE,IM) DOWHILE LENGTH NOT DONE
                          STAIY OPRNDA          STORE VALUE
                          DEY                   DECREMENT INDEX/LENGTH
                       ENDDO
                       RTS                      RETURN TO CALLER
```

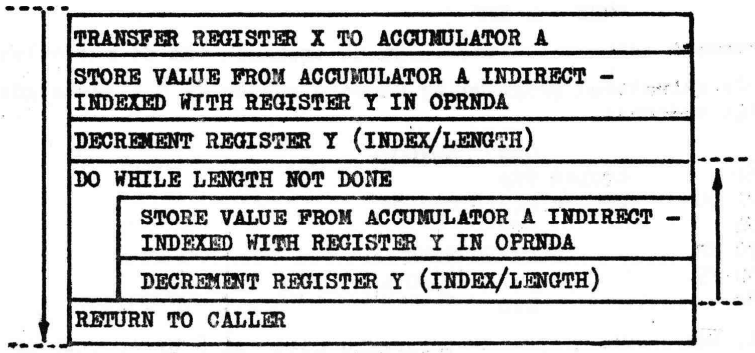
STRUCTURED PROGRAMMING		Nummer:
Deel: 1	Voorbeeld van DOWHILE	Blad: 8 van 12

Even kijken hoe de expansie hiervan is:

```
                        ORG   $0200
0200 8A       SRCLRM    TXA                          TRANSFER X TO A
0201 91 00              STAIY OPRNDA                 STORE VALUE
0203 88                 DEY                          DECREMENT INDEX/LENGTH
                        DOWHILE ((Y),NE,DONE,IM)     DOWHILE LENGTH NOT DONE
+0204 C0 FF   L0001A      CPYIM DONE                 DO WHILE CONDITION
+0206 F0 07               BEQ L0001X                 IS TRUE
 0208 91 00               STAIY OPRNDA               STORE VALUE
 020A 88                  DEY                        DECREMENT INDEX/LENGTH
                        ENDDO
+020B D0 F7               BNE L0001A                 UNCONDITIONAL JUMP BACK TO
+020D F0 F5               BEQ L0001A                 CONDITION TEST
+020F          L0001X     #                          DO EXIT POINT
 020F 60                 RTS                          RETURN TO CALLER
```

Zo kan het dus ook. Scheelt zelfs 66n byte met het voorgaande, maar daar zouden we ons niet druk over maken.

De iteration graph van Nassi en Schneiderman voor dit voorbeeld ziet er als volgt uit:



U ziet, het werkt. We kennen nu al strukturen, de DOUNTIL en de DOWHILE, rest ons nog de IF-THEN-ELSE en de CASE.

Dan nu een voorbeeld dat o.a. de IF-THEN-ELSE gebruikt.

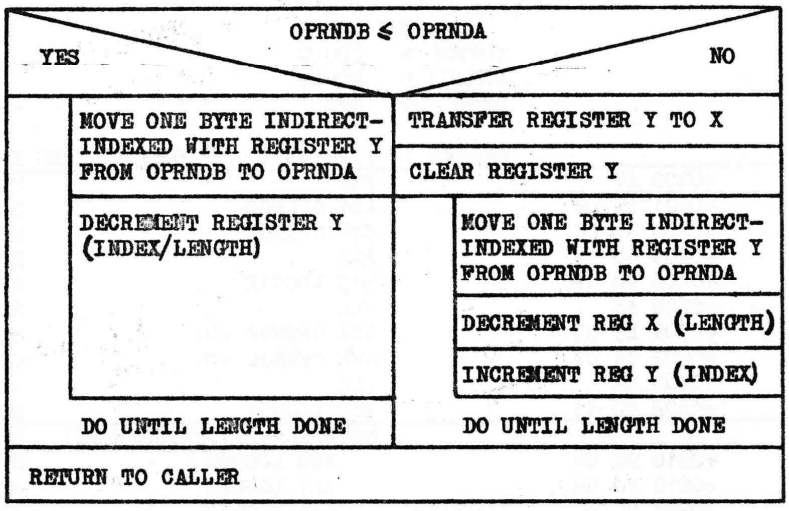
Voorbeeld 2: Gegeven: Een lengte van 1 t/m 256 in register Y (00 = 1, FF = 256), TO-adres in locaties OPRNDA = 0000 (low) en OPRNDA +01 = 0001 (high), FROM-adres in locaties OPRNDB = 0002 (low) en OPRNDB +01 = 0003 (high).

Gevraagd: Ontwerp een programma voor het "moven" van een stuk geheugen met behulp van bovenstaande gegevens, rekening houdend met het overlappen van de operands.

Datum ingang:	Vervangt:	d.d.:	Ref.:
14 september 1978	-	-	Anton Müller

STRUCTURED PROGRAMMING		Nummer:
Deel: 1	Voorbeeld van IFTHENELSE	Blad: 9 van 12

Oplossing:



Gestruktureerde coding:

```
SRMOVE IF (OPRNDB(2),LE,OPRNDA),THEN    IF OPERANDS OVERLAP THEN
       DOUNTIL ((Y),EQ,DONE,IM)          MOVE B TO A FROM HIGHORDER
          LDAIY OPRNDB                   ADDRESS
          STAIY OPRNDA
          DEY                            DECREMENT INDEX/LENGTH
       ENDDO
     ELSE                                ELSE
       TYX                               TRANSFER REGISTER Y TO X
       LDYIM $00                         CLEAR REGISTER Y (INDEX)
       DOUNTIL ((X),EQ,DONE,IM)          MOVE B TO A FROM LOWORDER
          LDAIY OPRNDB                   ADDRESS
          STAIY OPRNDA
          DEX                            DECREMENT LENGTH
          INY                            INCREMENT INDEX
       ENDDO
     ENDIF
     RTS                                 RETURN TO CALLER
```

Datum ingang:	Vervangt:	d.d.:	Ref.:
14 september 1978	-	-	Anton Müller

STRUCTURED PROGRAMMING		Nummer:
Deel: 1	Voorbeeld IFTHENELSE (vervolg)	Blad: 10 van 12

Gegenereerde coding:

```
                ORG $0200
OPRNDA          *   $0000
OPRNDB          *   $0002
DONE            *   $FF

                SRMOVE IF (OPRNDB(2),LE,OPRNDA),THEN IF OPERANDS OVERLAP THEN
+0200 48                PHA                     SAVE ACCUMULATOR
+0201 A5 02             LDA OPRNDB              COMPARE FIRST VALUE
+0203 C5 00             CMP OPRNDA              WITH SECOND VALUE
+0205 68                PLA                     RESTORE ACCUMULATOR
+0206 90 19             BCC L0001X              FIRST VALUE   SECOND VALUE
+0208 48                PHA                     SAVE ACCUMULATOR
+0209 A5 03             LDA OPRNDB +01          COMPARE FIRST VALUE
+020B C5 01             CMP OPRNDA +01          WITH SECOND VALUE
+020D 68                PLA                     RESTORE ACCUMULATOR
+020E 90 11             BCC L0001X              FIRST VALUE   SECOND VALUE
                        DOUNTIL ((Y),EQ,DONE,IM) MOVE B TO A FROM HIGHORDER
+0210 D0 06                BNE L0002B           UNCONDITIONAL BRANCH TO
+0212 F0 04                BEQ L0002B           EXECUTION LOGIC
+0214 C0 FF      L0002A    CPYIM DONE           DO UNTIL CONDITION
+0216 F0 09                BEQ L0002X           IS TRUE
+0218            L0002B    *
 0218 B1 02                LDAIY OPRNDB         ADDRESS
 021A 91 00                STAIY OPRNDA
 021C 88                   DEY                  DECREMENT INDEX/LENGTH
                        ENDDO
+021D D0 F5                BNE L0002A           UNCONDITIONAL JUMP BACK TO
+021F F0 F3                BEQ L0002A           CONDITION TEST
+0221            L0002X    *                    DO EXIT POINT
                     ELSE                       ELSE
+0221            L0001X    *                    ELSE LOGIC
                        TYX                     TRANSFER REGISTER Y TO X
+0221 48                PHA                     SAVE ACCUMULATOR
+0222 98                TYA                     TRANSFER REGISTER Y TO A
+0223 AA                TAX                     TRANSFER REGISTER A TO X
+0224 68                PLA                     RESTORE ACCUMULATOR
 0225 A0 00             LDYIM $00               CLEAR REGISTER Y (INDEX)
                        DOUNTIL ((X),EQ,DONE,IM) MOVE B TO A FROM LOWORDER
+0227 D0 06                BNE L0006B           UNCONDITIONAL BRANCH TO
+0229 F0 04                BEQ L0006B           EXECUTION LOGIC
+022B E0 FF      L0006A    CPXIM DONE           DO UNTIL CONDITION
+022D F0 0A                BEQ L0006X           IS TRUE
+022F            L0006B    *
 022F B1 02                LDAIY OPRNDB         ADDRESS
 0231 91 00                STAIY OPRNDA
 0233 CA                   DEX                  DECREMENT LENGTH
 0234 C8                   INY                  INCREMENT INDEX
```

Datum ingang:	Vervangt:	d.d.:	Ref.:
14 september 1978	-	-	Anton Müller

```
                                ENDDO
+0235 D0 F4                     BNE L0006A      UNCONDITIONAL JUMP BACK TO
+0237 F0 F2                     BEQ L0006A      CONDITION TEST
+0239              L0006X       =               DO EXIT POINT
                                ENDIF
 0239 60                        RTS             RETURN TO CALLER
```

U ziet, het werkt nog steeds. Overigens vindt ik wel dat de programma listings met de geëxpandeerde coding niet zo duidelijk meer te lezen zijn. Ik raadt U dan ook aan er niet al te veel naar te kijken en U meer te concentreren op de Nassi/Schneiderman diagrammen en de gestruktureerde (ongeëxpandeerde) coding. Met de gegenereerde coding heb ik alleen maar willen laten zien dat het inderdaad ook werkt in de uiteindelijke machinetaal.

Rest ons nog een voorbeeldje van de CASE struktuur, doch dat bewaren we tot de volgende keer. Uiteindelijk kunt U voorlopig de CASE ondervangen met de IF macro.

Gaarne ontvang ik Uw reakties op dit verhaal. In de eerste plaats omdat ik wil weten of ik gehoor vindt. In de tweede plaats omdat zaken die voor mij vanzelfsprekend zijn, voor sommigen van U misschien niet zo vanzelfsprekend zijn. Em, als U alles snapt, het met dit verhaal helemaal niet, of niet helemaal eens bent.

Afhankelijk van de reakties ga ik in KIM KENNER 6 en volgenden verder met dit onderwerp.

Copyright © 1978 by Anton Müller, Sinjeur Semeynsstr 78 I, Amsterdam, The Netherlands.

No part of this article may be reproduced in any form, by print, photoprint, microfilm or any other means without written permission from the author. Niets uit deze uitgave mag worden verveelvoudigd en/of openbaar gemaakt door middel van druk, fotocopie, microfilm of op welke andere wijze ook, zonder voorafgaande schriftelijke toestemming van de schrijver.

GEBRUIKERS CLUB NEDERLAND

SOFTWARE LIBRARY

STRUCTURED PROGRAMMING		Nummer:
Deel: 1	Referenties/literatuuropgave	Blad: 12 van 12

Referenties:

1. Conway, R en Gries, D. "An introduction to Programming: A Structured Approach". Winthrop, Cambridge, Mass., 1973.
2. Dahl, O.S., Dijkstra, E.W. en Hoare, C.A.R. "Structured Programming". Academic Press, New York, 1972.
3. Dijkstra, E.W. Notes on structured programming. In (2).
4. Gries, D. Describing an algorithm by Hopcroft. "Acta Informatica 2" (1973), 97-100.
5. Hetzel, W.C. "Program Test Methods". Prentice Hall, Englewood Cliffs, N.J., 1973.
6. Polya, G. "How to solve it". Princeton Press, 1971.
7. Wirth, N. "Program development by stepwise refinement". Comm. ACM 14 (Apr. 1971), 221-227.
8. Wirth, N. "Systematic Programming: An Introduction". Prentice Hall, Englewood Cliffs, N.J., 1973.
9. Dijkstra, E.W. "Goto statement considered harmful". Comm. ACM 11 (Mar. 1968), 147-148.
10. Mills, H.D. "Chief programmer team operations". IBM Technical Report FSC 71-5108, 1971.
11. Donaldson, J. "Structured programming". Datamation (Dec. 1973), 53.
12. Mills, H.D. "Mathematical foundations of structured programming". IBM Technical Report FSC 72-6012, 1972.
13. Karp, R. Datamation (Mar. 1974), 158.
14. Bauer, F.L. "A course of three lectures on a philosophy of programming", Oct. 1973.
15. Butterworth, D. Datamation (Mar. 1974), 158.
16. Denning, P.J. ACM SIGPLAN Notices, Oct. 1973.
17. Improved Programming Technologies: Management Overview. IBM publication nr. GE 19-5036-0 (Jan. 1975).
18. Nassi, I. en Schneiderman, B. "Flowchart techniques for structured programming". ACM SIGPLAN Notices, vol. 8, no. 8, (Aug. 1973), 12-26.
19. Basili, V. "Structured programming: foundations". Structured programming tutorial, IEEE Catalog No. 75CH1049-6, IEEE Computer Society, 5855 Naples Plaza, Suite 301, Long Beach, California 90803.
20. Micro ADE for the 6502, Assembler Disassembler Editor, by Peter R. Jennings Micro-Ware Ltd. 27 Firstbrooke Road, Toronto, Ontario, Canada M4E 2L2, (1977).
21. Structured Programming Macros, Field developped program nr: 5798-CLF Program description/operations manual, IBM publication nr: SB21-1958-0, (1976).

Datum ingang:	Vervangt:	d.d.:	Ref.:
14 september 1978	-	-	Anton Müller

MODIFICATIONS AND EXTENSIONS TO MICRO-ADE		Nummer:
Introduction		Blad: 1

```
**************************************************
*                                                *
*   MODIFICATIONS AND EXTENSIONS TO MICRO-ADE    *
*                                                *
*    EDITOR, ASSEMBLER AND DIS-ASSEMBLER FOR     *
*                                                *
*            6502-MICRO-PROCESSORS               *
*                                                *
**************************************************
```

COPYRIGHT, 1978 ALL RIGHTS RESERVED.
TOM OFFRINGA LEIDSCHENDAM HOLLAND.

****

MICRO-ADE IS A COMPLETE PACKAGE, WRITTEN BY PETER R.JENNINGS (MICROWARE LTD., ONTARIO, CANADA). IT OPERATES IN CONJUNCTION WITH THE KIM-1 MONITOR IN A 8K-RAM EXTENSION TO THIS MICROCOMPUTER.

ALTHOUGH MICRO-ADE WILL OPERATE WITH ONE CASSETTE-RECORDER FOR FILE HANDLING, IT IS SUPPLIED WITH START/STOP CONTROL FOR TWO OF THEM.

ALL I/O SOFTWARE IS ASSEMBLED AT THE END OF THE PACKAGE AND WELL DOCUMENTED IN THE MANUAL; SEPARATELY A GOOD LISTING CAN BE BOUGHT FROM MICROWARE LTD.

SIEP DE VRIES TESTED MICRO-ADE'S FIRST APPEARENCE IN HOLLAND AND WAS VERY SATISFIED WITH ITS DOCUMENTATION, ITS OPERATION AND ITS COMMAND-DECODER. HE JUST FOUND ONE BUG, CHANGED ITS MEMORY-BACK-UP FROM CASSETTE INTO PAPERTAPE AND MODIFIED THE PRINT-OUT OF THE ASSEMBLER.

AFTER NEARLY ONE MONTH'S EXPERIMENTATION WITH ONE CASSETTE RECORDER AND MICRO-ADE, I CAME TO THE CONCLUSION, THAT IT WOULD BE WORTH WHILE TO ARRANGE FOR A SECOND RECORDER AND INSTALL THE START/STOP CONTROLS IN THEM.

THESE START/STOP CONTROLS COSTED ME ANOTHER WEEK, BUT WORKED THEN TO SATISFACTION.
MICRO-ADE STILL PROVED TO BE A POWERFUL, HOWEVER SLOW ACTING OPERATING SYSTEM FOR SOFTWARE DEVELOPMENT. THE TIME CONSUMING ELEMENTS WERE MY MIND, MY ACCURACY AND THE CASSETTE-SYSTEM.
IN THAT ORDER !

Datum ingang:	Vervangt:	d.d.:	Ref.:
September, 19, 1978	-	-	Tom Offringa

MODIFICATIONS AND EXTENSIONS TO MICRO-ADE		Nummer:
Introduction (cont'd)		Blad: 2

MY DOUBTS ABOUT THE RELIABILITY OF THE CASSETTES AND KIM'S PHASE-LOCKED LOOP INPUT CHANGED GRADUALLY INTO THE CLASSIFICATION, " NEARLY PROFESSIONAL". UNDER CERTAIN RESTRICTIONS.

THE MODIFICATIONS AND EXTENSIONS BELOW CAN BE SEEN AS THE RESULT OF THE GROWING PATIENCE WITH MYSELF AND THE HUMAN LUST FOR PERFECTION.

YET, STILL SOME IMPROVEMENTS ARE POSSIBLE.

FOR INSTANCE:

* MICRO-ADE DOESN'T WARN YOU WHEN YOU FORGET TO DEFINE AN ARGUMENT FOR MULTIPLE-BYTES-INSTRUCTIONS.

* YOU HAVE TO CHECK THE OBJECT CODE, IN PARTICULAR THE RELATIVE BRANCHES AT THE END OF YOUR PROGRAM, EVEN WHEN GENERATED WITHOUT ERROR MESSAGES.

* WORKING WITH CASSETTES ONLY ONE ORG-STATEMENT CAN BE USED IN A FILE.

NEVERTHELESS, I FEEL RATHER SATISFIED WITH MY MACHINE TODAY AND FULLY RECOMMEND MICRO-ADE TO USERS OF THE KIM-MICROCOMPUTER.

THE MODIFICATIONS AND EXTENSIONS ARE PRESENTED IN A "STEP-BY-STEP" METHOD: YOU CAN TRY THEM ONE BY ONE AND THEN DECIDE WHICH YOU WANT TO IMPLEMENT ON YOUR MICRO-ADE.

THE PRESENTATION IS IN THE ORDER:

1. TWO BUGS IN MICRO-ADE
2. MEMORY ALLOCATION FOR TESTING SOFTWARE
3. IMPROVEMENTS (DESIGN SIEP DE VRIES, HOLLAND)
4. COMPRESSED OUTPUT ON YOUR DISPLAY
5. RESET OF LINE-NUMBERS AT COLD-START
6. SOURCE ADDITION FROM CASSETTE TO BUFFER
7. PROTECTION OF THE SOURCE-BUFFER
8. LISTINGS WITHOUT LINE-NUMBERS

Datum ingang:	Vervangt:	d.d.:	Ref.:
September,19 1978	-	-	Tom Offringa

MODIFICATIONS AND EXTENSIONS TO MICRO-ADE		Nummer:
BUGS IN MICRO-ADE AND..	MEMORY ALLOCATION	Blad: 3

****

TWO BUGS IN MICRO-ADE

****

A. IN PROCESSING THE = STATEMENT,THE ASSEMBLER ALWAYS OUTPUTS THE HIGH ORDER PART OF AN ADDRESS.

-MAKE THE FOLLOWING PATCH:
-LOCATION £2AF9 = 49 INTO 48

B. IN PRODUCING OBJECT CODE ON CASSETE,AN EXTRA BYTE IS SAVED.. THIS BYTE MIGHT OVERWRITE AN EXISTING PROGRAM.

-MAKE THE FOLLOWING PATCH:
-LOCATION £26C9 = 01 INTO 00

****
MEMORY ALLOCATIONS FOR TESTING SOFTWARE
****

IF YOU HAVE ANOTHER 4K RAM INSTALLED ON YOUR MACHINE DIRECTLY BEHIND THE STANDARD RAM OF KIM-1, IT IS WORTH WHILE TO USE YOUR MEMORY AS FOLLOWS:

--
- 0070 TO 00DF PAGE ZERO LOCATIONS
- 0200 TO 11FF SOFTWARE DEVELOPMENT (4K)
-1200 TO 13FF OBJECTCODE FROM MICROADE

- 2000 TO 30FF MICRO-ADE AND EXTENSIONS
- 3100 TO 35FF SYMBOL TABLE
- 3600 TO 3FFF SOURCE CODE FOR THE ASSEMBLER

IT IS NOW POSSIBLE TO DEVELOP 4K OF SOFTWARE, PATCHING AND EXTENDING IT BY MICRO-ADE AND DEBUG IT BY THE KIM MONITOR,WITHOUT BOTHERING ABOUT THE COMMON USE OF MEMORY..

THE MEMORY ALLOCATION TABLE OF MICRO-ADE SHOULD BE:
-CHANGE FROM LOCATION £2EA3 = 35 36 40 30 36 02
INTO 35 36 40 31 36 12

@

Datum ingang:	Vervangt:	d.d.:	Ref.:
September,19 1978	-	-	Tom Offringa

MODIFICATIONS AND EXTENSIONS TO MICRO-ADE		Nummer:
IMPROVEMENTS OF	SIEP DE VRIES	Blad: 4

IMPROVEMENTS OF SIEP DE VRIES, LIMMEN HOLLAND.
WESTVRIES COMPUTER CONSULTING
**********:

VERY PRACTICALLY,,SIEP WANTED TO USE ALL OF THE
STANDARD FORMAT PAPER. HE CHANGED THE HEADER
OUTPUT OF THE ASSEMBLER AND THE PAGELENGTH.BESIDES
HE EXTENDED MICRO-ADE WITH A ROUTINE TO GET A PROPER
OUTPUT ON THE * STATEMENT WITHIN THE LISTING.

- MAKE THE FOLLOWING PATCHES:
- FROM LOC. £29FD = A9 0C 20 8D 27
  INTO EA EA 20 87 27
- LOC.. £2A2E = 03 INTO 01 JUST ONE LINE)
- LOC. £2A36 = C8 INTO BE(FOR A4=FORMAT).
  OR:
  OR: INTO CE (FOR KIMKENNER FORMAT)
- LOC. £2A5E = A5 3E 20
  INTO: 4C 00 30 (JMP PRINTIT)
- THEN EXTEND MICRO ADE WITH:

```
3000            PRINTI ORG    £3000

          3E 00 PCHI   *      £003E  HIGH PC
          47 00 OP     *      £0047  OPCODE

          80 27 HEXPR  *      £2780  PRINT A BYTE
          8B 27 OUTSP  *      £278B  PRINT A SPACE
          63 2A BACK   *      £2A63  RETURN TO ASS.
          83 2A PRBUF  *      £2A83  RETURN TO ASS.

3000 A5 47                    LDA   OP
3002 C9 FA                    CMPIM £FA     TEST FOR *
3004 F0 08                    BEQ           TISTAR
3006 A5 3E                    LDA   PCHI    NO,ACT NORMAL
3008 20 80 27                 JSR   HEXPR
300B 4C 63 2A                 JMP   BACK
300E A0 08      TISTAR        LDYIM £08     EIGHT SPACES
3010 20 8B 27                 JSR   OUTSP
3013 88                       DEY
3014 D0 FA                    BNE   TISTAR +02
3016 A5 48                    LDA   OP       +01 LOW ORDER BYTE
3018 20 80 27                JSR   HEXPR
301B 20 8B 27                 JSR   OUTSP
301E A5 49                    LDA   OP
3020 20 80 27                 JSR   HEXPR
3023 20 8B 27                 JSR OUTSP
3026 4C 63 2A                 JMP   BACK
```

Datum ingang:	Vervangt:	d.d.:	Ref.:
September, 19 1978	-	-	Tom Offringa

MODIFICATIONS AND EXTENSIONS TO MICRO-ADE		Nummer:
COMPRESSED OUTPUT AND..	RESET OF LINE-NUMBERS	Blad: 5

```
**************************************************

        COMPRESSED OUTPUT ON THE DISPLAY

**************************************************

WHEN USING A 64 * 16 DISPLAY, ONE IS ASTONISHED TO
SEE MUCH OF THE DISPLAY'S AREA LOST TO EXTRA RETURN
LINEFEED'S AFTER A COMMAND.
I SCRUTINIZED MICRO-ADE FOR THIS DUBIOUS OUTPUT
AND MADE THE FOLLOWING PATCHES:

- LOCATION £20F8 =     31     INTO     34
- LOCATION £2175 =     31     INTO     34
- LOCATION £261C = 20 87 27 INTO EA EA EA
- LOCATION £271E =     31     INTO     34
- LOCATION £2C0A =     0D     INTO     20
- LOCATION £2C12 =     0D     INTO     20
- LOCATION £2C19 =     0D     INTO     20
- LOCATION £2C22 =     0D     INTO     20
- LOCATION £2C55 =     0D     INTO     10   !!!
- LOCATION £2C68 =     0D     INTO     20
- LOCATION £2C6D =     0D     INTO     10   !!!
- LOCATION £2EAC = 20 EE 2D INTO 20 E7 2D

- AND THEN THE LAST ONE:
- FROM LOCATION £2C73 = 0D 43 4C 45 41 52
-                  INTO   43 4C 45 41 52 0D

**************************************************

      RESET OF LINE-NUMBERS AT A COLD START

**************************************************

ORIGINALLY MICO-ADE'S FIRST LINE-NUMBER AT A COLD
START IS 0000:.  THIS SHOULD BE  0010:.

- FIRST MAKE THE FOLLOWING PATCH:
- LOCATION £202C = A0 69 20 INTO 4C 29 30

- THEN EXTEND MICRO-ADE WITH
```

LNSTRT TOM OFFRINGA LEIDSCHENDAM PAGE 01

```
0010:
0020: 3029            LNSTRT ORG   £3029
0030:
0040:       57 21     NUMB   *     £2157
0050:       A0 27     MESSY  *     £27A0
0060:
0070: 3029 A0 69            LDYIM £69
0080: 302B 20 A0 27         JSR   MESSY
0090: 302E 4C 54 21         JMP   NUMB   -33
```

Datum ingang:	Vervangt:	d.d.:	Ref.:
September, 19 1978	-	-	Tom Offringa

MODIFICATIONS AND EXTENSIONS TO MICRO-ADE	Nummer:
SOURCE ADDITION FROM CASSETTE TO THE BUFFER	Blad: 6

```
          ************************************************

                SOURCE ADDITION FROM CASSETTE TO THE BUFFER

          ************************************************


CASSAD      TOM OFFRINGA LEIDSCHENDAM     PAGE 01

0010: 22D3              CASSAD ORG   £22D3
0020:
0030:                   ****************************
0040:                   ADD A FILE TO EXISTING BUFFER
0050:
0060:                   COMMAND:    G 00
0070:                   ****************************
0080:
0090:                   - WHEN EDITING, THE SOURCE BUFFER NORMALLY IS
0100:                   - REPLACED BY THE NEW FILE FROM YOUR CASSETTE.
0110:                   - THIS EXTENSION ENABLES ADDITION TO THE SOURCE
0120:                   - UNTIL THE VERY END OF THE BUFFER.
0130:                   - HOWEVER, ALL LINENUMBERS WILL BE RENUMBERED !!!
0140:
0150:           10 00   BLO     *       £0010
0160:           11 00   BHI     *       £0011
0170:           1A 00   LOPAR   *       £001A
0180:           62 00   ID      *       £0062
0190:           F3 00   TMP     *       £00F3
0200:
0210:           42 17   SBD     *       £1742
0220:           E7 17   CHKL    *       £17E7
0230:           E8 17   CHKH    *       £17E8
0240:           EC 17   VEB     *       £17EC
0250:           32 19   INTVEB  *       £1932
0260:           4C 19   CHKT    *       £194C
0270:           EA 19   INCVSB  *       £19EA
0280:           F3 19   RDBYT   *       £19F3
0290:           24 1A   RDCHT   *       £1A24
0300:           41 1A   RDBIT   *       £1A41
0310:           8C 1E   INIT    *       £1E8C
0320:           57 21   NUMB    *       £2157
0330:           E6 23   DECBUF  *       £23E6
0340:           96 24   FNDND   *       £2496
0350:           2F 24   STORE   *       £242F
0360:           CD 2D   HEXOUT  *       £2DCD
0370:           E7 2D   CRLF    *       £2DE7
0380:           94 23   PACKT   *       £2394
0390:
0400: 22D3 A5 1B        GETYY   LDA     LOPAR   +01  MODIFICATION OF
0410: 22D5 C5 62                CMP     ID              ( GET FILE )
0420: 22D7 B0 F7                BCS     GETYY   -03
0430: 22D9 4C 54 21             JMP     NUMB    -03 RENUMBER LINES ANYWAY ID=41
```

Datum ingang:	Vervangt:	d.d.:	Ref.:
September, 19 1978	-	-	Tom Offringa

SOURCE ADDITION (CONTINUED 1)

****

TOM OFFRINGA LEIDSCHENDAM PAGE 02

```
0010: 2EA9                      ORG   £2EA9
0020:
0030: 2EA9 20 CD 2D  ERID       JSR   HEXOUT
0040: 2EAC 20 E7 2D             JSR   CRLF    MINOR CHANGE FOR PRESENTATION
0050: 2EAF A5 62     CREAD      LDA   ID
0060: 2EB1 C9 00                CMPIM £00
0070: 2EB3 D0 0B                BNE   CRE
0080: 2EB5 20 96 24             JSR   FNDND   RESET BLO/BHI
0090: 2EB8 A5 10                LDA   BLO
0100: 2EBA 38                   SEC
0110: 2EBB E9 04                SBCIM £04      4* DECBUF
0120: 2EBD 20 EB 23             JSR   DECBUF +05
0130: 2EC0 AD 02 17  CRE        LDA   £1702   START CASSETTE
0140: 2EC3 29 FB                ANDIM £FB
0150: 2EC5 8D 02 17             STA   £1702
0160: 2EC8 A9 7F                LDAIM £7F     DISPLAY
0170: 2ECA 8D 41 17             STA   £1741
0180: 2ECD 20 32 19             JSR   INTVEB  RESET CHECKSUM
0190: 2ED0 A9 13                LDAIM £13     INPUT CASSETTE
0200: 2ED2 8D 42 17             STA   £1742
0210: 2ED5 20 41 1A  SYNC       JSR   RDBIT
0220: 2ED8 46 F3                LSR   TMP
0230: 2EDA 05 F3                ORA   TMP
0240: 2EDC 85 F3                STA   TMP
0250: 2EDE 8D 40 17             STA   £1740    DISPLAY
0260: 2EE1 C9 16     TST        CMPIM £16
0270: 2EE3 D0 F0                BNE   SYNC
0280: 2EE5 20 24 1A             JSR   RDCHT
0290: 2EE8 8D 40 17             STA   £1740
0300: 2EEB C9 2A                CMPIM £2A      START OF FILE
0310: 2EED D0 F2                BNE   TST
0320: 2EEF A6 62                LDX   ID
0330: 2EF1 E0 00                CPXIM £00
0340: 2EF3 F0 1D                BEQ   NOID
0350: 2EF5 20 F3 19             JSR   RDBYT    READ FILE WITH ID
0360: 2EF8 C5 62                CMP   ID
0370: 2EFA D0 AD                BNE   ERID
0380: 2EFC 20 F3 19             JSR   RDBYT
0390: 2EFF 20 4C 19             JSR   CHKT
0400: 2F02 85 10                STA   BLO
0410: 2F04 20 F3 19             JSR   RDBYT
0420: 2F07 20 4C 19             JSR   CHKT
0430: 2F0A 85 11                STA   BHI
```

MODIFICATIONS AND EXTENSIONS TO MICRO-ADE		Nummer:
SOURCE ADDITION (Cont'd)		Blad: 8

SOURCE ADDITION (CONTINUED 2)

****

TOM OFFRINGA LEIDSCHENDAM PAGE 03

```
0440: 2F0C 20 E6 23          JSR   DECB'JF
0450: 2F0F 4C 23 2F          JMP   LOADIT
0460: 2F12 86 1B     NOID    STX   LOPAR  +01 RESET 2ND ARG. AND
0470: 2F14 20 F3 19          JSR   RDBYT      ADD JUST ONE FILE
0480: 2F17 20 F3 19          JSR   RDBYT
0490: 2F1A 20 4C 19          JSR   CHKT
0500: 2F1D 20 F3 19          JSR   RDBYT
0510: 2F20 20 4C 19          JSR   CHKT
0520: 2F23 A2 02     LOADIT  LDXIM £02
0530: 2F25 20 24 1A  READIT  JSR   RDCHT
0540: 2F28 C9 2F             CMPIM '/
0550: 2F2A F0 06             BEQ   ENDRDS
0560: 2F2C 20 94 2E          JSR   PACKT
0570: 2F2F 4C 31 30          JMP   £3031
0580: 2F32 4C 3F 30  ENDRDS  JMP   ENDRD   ID=42

0010: 3031                   ORG   £3031
0020:
0030: 3031 D0 29             BNE   SYNCS
0040: 3033 CA                DEX
0050: 3034 D0 29             BNE   READS
0060: 3036 20 4C 19          JSR   CHKT
0070: 3039 20 2F 24          JSR   STORE   AND CHECK FOR OVERFLOW
0080: 303C 4C 23 2F          JMP   LOADIT
0090: 303F 20 F3 19  ENDRD   JSR   RDBYT   END OF FILE
0100: 3042 CD E7 17          CMP   CHKL
0110: 3045 D0 15             BNE   SYNCS
0120: 3047 20 F3 19          JSR   RDBYT
0130: 304A CD E8 17          CMP   CHKH
0140: 304D D0 0D             BNE   SYNCS
0150: 304F AD 02 17          LDA   £1702   TURN OFF CASSETTE
0160: 3052 09 04             ORAIM £04
0170: 3054 8D 02 17          STA   £1702
0180: 3057 E6 62             INC   ID
0190: 3059 4C 8C 1E          JMP   INIT    STOP DISPLAY
0200: 305C 4C D5 2E  SYNCS   JMP   SYNC
0210: 305F 4C 25 2F  READS   JMP   READIT  ID=
```

Datum ingang:	Vervangt:	d.d.:	Ref.:
September, 19 1978	-	-	Tom Offringa

KIM

GEBRUIKERS CLUB NEDERLAND

SOFTWARE LIBRARY

MODIFICATIONS AND EXTENSIONS TO MICRO-ADE	Nummer:
PROTECTION SOURCEBUFFER	Blad: 9

```
BFPROT      TOM OFFRINGA LEIDSCHENDAM     PAGE 01

0010: 2437            BFPROT ORG   £2437
0020:
0030:
0040:                 ************************
0050:                 PROTECTION FOR TEXTBUFFER
0060:                  EXTENSION  TO MICRO-ADE
0070:                 ************************
0080:
0090:                 - MICRO-ADE PROVIDES NO PROTECTION ON ITS BUFFER.
0100:                 - ONLY ERROR MESSAGE ***<3B> WILL SIGNAL AN ATTEMPT
0110:                 - TO STORE SOURCE-CODED DATA BEYOND THIS BUFFER.
0120:
0130:                 - THIS EXTENSION PROVIDES A SAFE BUFFER  PROTECTION
0140:                 - FOR THE 'G 00' COMMAND AND IS ALSO EFFECTIVE WHEN
0150:                 - EDITING FROM YOUR KEYBOARD.
0160:                 - HOWEVER: ERROR MESSAGE WILL BE   ***<85>  AND
0170:                 -        : YOU MIGHT BE FORCED TO DEBUG THE END
0180:                 -          OF THE TEXTBUFFER WHEN EDITING !!!
0190:                 -        : FIRST: FILL 3 LOCATIONS AFTER LAST '0D'
0200:                 -               : WITH THE ALPHA-TERMINATOR   '40'
0210:                 -        : THEN : APPLY THE 'N' COMMAND.
0220:
0230:          10 00  BLO    *      £0010
0240:          11 00  BHI    *      £0011
0250:
0260:          8C 1E  INIT   *      £1E8C
0270:
0280: 2437 20 62 30           JSR    PROTCT
0290:
0300:                  ID=44

0010: 3062                    ORG    £3062
0020:
0030: 3062 F0 01      PROTCT  BEQ    NO-OKS  OKS = LABEL IN MICRO-ADE
0040: 3064 60                 RTS
0050: 3065 88         NO-OKS  DEY            SOURCE -01
0060: 3066 84 11              STY    BHI
0070: 3068 A0 FD              LDYIM  £FD      FORCE 3* £40 AT BUFFER-END
0080: 306A 84 10              STY    BLO
0090: 306C A9 40              LDAIM  £40
0100: 306E A2 03              LDXIM  £03
0110: 3070 A0 00              LDYIM  £00
0120: 3072 91 10              STAIY  BLO
0130: 3074 C8                 INY
0140: 3075 CA                 DEX
0150: 3076 D0 FA              BNE    NO-OKS +0D
0160: 3078 AD 02 17           LDA    £1702
0170: 307B 09 04              ORAIM  £04      STOP CASSETTE
0180: 307D 8D 02 17           STA    £1702
0190: 3080 20 8C 1E           JSR    INIT     STOP DISPLAY
0200: 3083 00                 BRK             ***<85> ID=
```

Datum ingang:	Vervangt:	d.d.:	Ref.:
September, 19 1978	-	-	Tom Offringa

MODIFICATIONS AND EXTENSIONS TO MICRO-ADE	Nummer:
LISTINGS WITHOUT LINE-NUMBERS	Blad: 10

```
*****************************************************

                LISTINGS WITHOUT LINE-NUMBERS

*****************************************************
TEXT        TOM OFFRINGA LEIDSCHENDAM     PAGE 01

3084                TEXT    ORG    £3084

                    ************************
                    LIST WITHOUT LINENUMBERS
                     EXTENSION TO MICRO-ADE
                    ************************

                    - THIS IS JUST A MODIFICATION FOR SOMEONE WHO
                    - LIKES TO USE HIS COMPUTER FOR WRITING LETTERS.

                    - MAKE THE FOLLOWING PATCHES:

                    -  LOCATION £20F4 = 20 67 23 INTO 20 84 30
                    -  LOCATION £238A = 20 C5 2D INTO 4C 99 30

                    - THAN EXTEND MICRO-ADE WITH:

          4D 00   PRFLAG *      £004D
          00 01   BUFFER *      £0100
          67 23   LIST   *      £2367
          8A 23   PRINT  *      £238A
          C5 2D   NOUT   *      £2DC5
          EE 2D   OUTSP  *      £2DEE

3084 A5 4D        LISTX  LDA    PRFLAG
3086 C9 54               CMPIM 'T
3088 F0 07               BEQ    LISTY
308A AD 01 01            LDA    BUFFER +01  LOOK FOR 2ND. LETTER
308D C9 54               CMPIM 'T        IS IT 'T' ?
308F F0 04               BEQ    LISTY  +04
3091 20 67 23     LISTY  JSR    LIST
3094 60                  RTS
3095 85 4D               STA    PRFLAG
3097 F0 F8               BEQ    LISTY

3099 A5 4D        PRNTX  LDA    PRFLAG
309B C9 54               CMPIM 'T
309D F0 06               BEQ    PRNTY
309F 20 C5 2D            JSR    NOUT    OUTPUT NUMBER
30A2 4C 8D 23            JMP    PRINT  +03
30A5 A2 06        PRNTY  LDXIM £06    6 SPACES
30A7 20 EE 2D            JSR    OUTSP
30AA CA                  DEX
30AB D0 FA               BNE    PRNTY  +02
30AD 4C 95 23            JMP    PRINT  +03 ID=
```

Datum ingang:	Vervangt:	d.d.:	Ref.:
September, 19 1978	-	-	Tom Offringa

THE RESULT IS SHOWN IN THE END OF THIS STORY.

IT IS NOW POSSIBLE TO USE YOUR MICRO-ADE AS A TEXTEDITOR.

JUST GIVE A TWO-LETTER COMMAND 'LT' WITH A RETURN
AND YOU WILL NOT SEE ANY LINENUMBER ON YOUR PRINTER.
ONLY THE TERMINATING SIGN AT THE END OF YOUR LETTER
WILL SHOW MICRO-ADE'S WORK.

IT IS POSSIBLE TO GET THE NUMBERS BACK BY RESETTING THE
PRINTFLAG OF THE ASSEMBLER THROUGH AN INEFFECTIVE PASS 2:

```
COMMANDS:     'X2636' RETURN
PASS 2 PRINT?  'YES' RETURN
SAVE ID=             RETURN
ID=                  RETURN
```

NOW THE ORIGINAL LISTING WILL BE AVAILLABLE AGAIN.

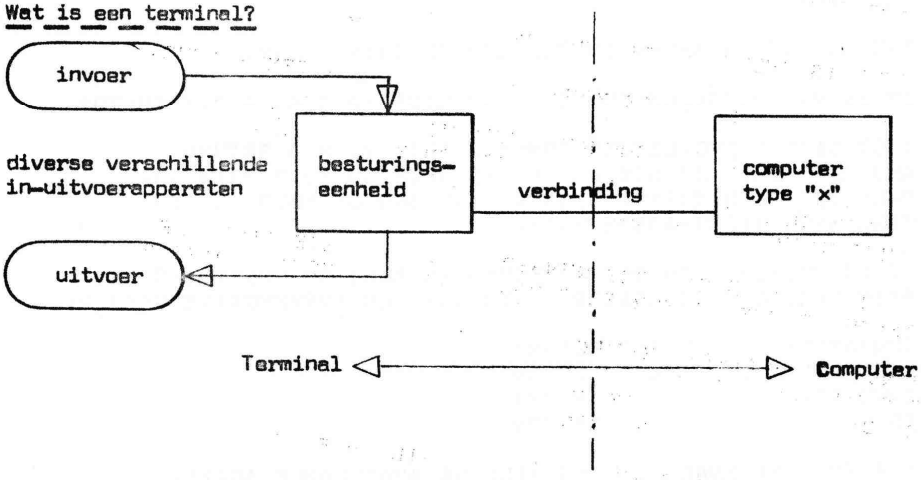
I HOPE, YOU ENJOYED ALL THESE EXTENSIONS.

```
3000 A5 47 C9 FA F0 08 A5 3E 20 80 27 4C 63 2A A0 08 .G.....>...L....
3010 20 8B 27 88 D0 FA A5 48 20 80 27 20 8B 27 A5 49 .......H.......I
3020 20 80 27 20 8B 27 4C 83 2A A0 69 20 A0 27 4C 54 ......L.......LT
3030 21 D0 29 CA D0 29 20 4C 19 20 2F 24 4C 23 2F 20 .......L....L...
3040 F3 19 CD E7 17 D0 15 20 F3 19 CD E8 17 D0 0D AD ................
3050 02 17 09 04 8D 02 17 E6 62 4C 8C 1E 4C D5 2E 4C ........L..L..L
3060 25 2F F0 01 60 88 84 11 A0 FD 84 10 A9 40 A2 03 ............@...
3070 A0 00 91 10 C8 CA D0 FA AD 02 17 09 04 8D 02 17 ................
3080 20 8C 1E 00 A5 4D C9 54 F0 07 AD 01 01 C9 54 F0 .....M.T......T.
3090 04 20 67 23 60 85 4D F0 F8 A5 4D C9 54 F0 06 20 ......M...M.T...
30A0 C5 2D 4C 8D 23 A2 06 20 EE 2D CA D0 FA 4C 95 23 ..L.....L...L..
```

```
**************************************************
*                                                *
*   MODIFICATIONS AND EXTENSIONS TO MICRO-ADE    *
*                                                *
*        LEIDSCHENDAM          15 SEPT. 1978     *
*                                                *
**************************************************
```

HET GEBRUIK VAN DE KIM ALS BESTURING VAN EEN TERMINAL.

Wat is een terminal?



Terminal is een plaats voor een computeroperator, dus iemand die de computer opdracht geeft om dingen en taken te verrichten.

Welke faciliteiten heeft een terminal normaal gesproken?

1. Een apparaat waarmee de operator kan communiceren met computer "x". Bijv. een display met toetsenbord.
2. Een apparaat om grotere hoeveelheden data in de computer te brengen. Bijv. magneetband, ponsband.
3. Een apparaat om grotere hoeveelheden data vanuit de computer op te slaan bijv. magneetband, ponsband.
4. Een apparaat om gegevens uit de computer leesbaar naar buiten te brengen zoals een teletype.

Hoe ziet de verbinding tussen computer en besturingseenheid eruit?

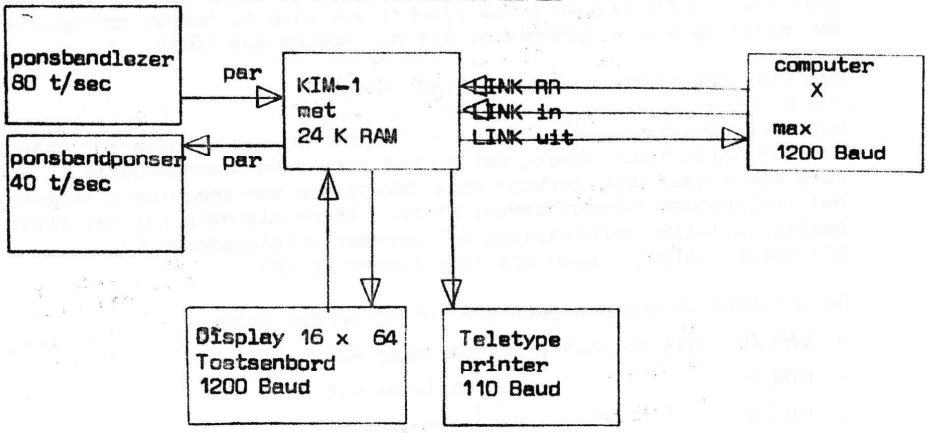
Aangezien de terminal geschikt moet zijn om aan vele verschillende computers aangesloten te kunnen worden, dient de koppelingshardware zo universeel mogelijk te zijn. Bijna alle computers ter wereld, hoe groot of klein, hebben een aansluiting voor een teletype.
De koppeling dient dus een seriesignaal te zijn over een fullduplexkanaal.

Wat is de functie van de besturingseenheid?

De besturingseenheid zorgt enerzijds voor de verschillende aansluiting van de verschillende periferals, die erop aangesloten zijn.De verschillende apparaten hebben verschillende transmissiesnelheden.

Anderzijds kan over het communicatiekanaal slechts één teken per tijdseenheid overgeseind worden. De besturingseenheid bepaalt, waar dit teken vandaan komt en waar het heen gaat.

Welke configuratie is voor IØKIM gebruikt?



- Centrale besturingseenheid is een KIM-1 met 24 K RAM, waarin het programma (± 3 K) en buffers voor snelheidsaanpassing zich bevinden.
- Communicatiemiddel voor de operator is een beeldbuisstation met beeldscherm van 16 regels bij 64 tekens per regel. Dit werkt op 1200 Baud en is aangesloten op de KIM teletype in- en uitgang.
- Data-invoer gebeurt via een ponsbandlezer, die maximaal 80 tekens per seconde kan lezen. De lezer kan per teken gestart en gestopt worden.
- Data-uitvoer gebeurt door middel van een ponsbandponser, die maximaal 40 tekens per seconde kan ponsen.
- Alle hardcopy uitvoer gebeurt op een teletype, waarvan alleen de printer gebruikt wordt. De snelheid bedraagt 10 tekens per seconde.
- De link naar de andere computer is een 20 MA serie in- en uitgang. (full-duplex) Behalve een receive en een transmit signaal kan hier ook nog een z.g. Reader Run signaal aangesloten worden. Dit is het signaal, waarmee een computer de ponsbandlezer van een erop aangesloten teletype kan bedienen. Als computer X dit signaal heeft, kan het aangesloten worden. Zo niet, dan wordt de ingang opengelaten.

Interessante bijkomstigheid van de hardware is, dat ponsbandlezer, ponsbandponser, teletype en computerlink gezamenlijk bediend worden via de applicatie-PIA van de KIM. De bits van de A-kant zijn gemultiplexed, terwijl de B-kant dient om een kanaal te selecteren.

Met welk doel is IØKIM gemaakt?

De KIM zoals beschreven word gebruikt als programmaontwikkelingssysteem voor 6502-programma's. Met zijn ponsbandapparatuur was dit een redelijk comfortabel systeem voor editing, assembleren en testen.
Bij ons bedrijf komen regelmatig andere types computers binnen, die dan geprogrammeerd moeten worden.

Frustrerend is dan om óf terug te moeten vallen op volledig werken met een teletype op 110 Baud, óf in het gunstigste geval als de interface aanwezig is, alles te moeten omschakelen, andere pluggen aan te zetten e.d. Veel comfortabeler is om slechts een plug te hoeven aankoppelen en dan alles te kunnen gebruiken. Dit nu gebeurt met IØKIM.

Hoe ziet het protocol over de link eruit?

Het eenvoudigste is wat genoemd zou kunnen worden transparant gebruik. Hierbij wordt ieder teken, dat op het toetsenbord aangeslagen is naar computer x gestuurd, terwijl alle tekens die van computer x afkomen, op het beeldscherm terecht komen. Hierbij is de situatie dus net alsof het beeldbuisstation rechtstreeks op computer x aangesloten is.
Dit geldt, zolang x geen speciale tekens stuurt.

De speciale tekens die gestuurd kunnen worden zijn:

- CTRL/Q Dit betekent: Zet de lezer aan
- CTRL/S Zet de lezer uit
- CTRL/R Zet de pons aan
- CTRL/T Zet de pons uit.

Deze tekens fungeren a.h.w. als schakelaar en x kan met behulp van deze tekens omschakelen van toetsenbord naar ponsbandlezer en van beeldscherm naar ponsbandponser.
X bepaalt dus, <u>waar</u> zijn invoer vandaan komt en <u>waar</u> zijn uitvoer heengaat.

Een alternatieve mogelijkheid voor het bedienen van de lezer bestaat uit het signaal <u>Reader Run</u>, dat sommige computers afgeven.

- Reader Run stroomvoerend betekent dan invoer vanaf toetsenbord.
- Reader Run stroomloos betekent invoer van de ponsbandlezer.

Deze procedure is het z.g. teletypeprotocol zoals op diverse timesharing-systemen in gebruik is.

Welke functie staan de operator ter beschikking?

De operator kan commando's aan het terminalsysteem geven, die in <u>2 categorieën</u> uiteenvallen.
<u>De eerste categorie</u> omvat commando's, die identiek zijn aan de bedieningsfuncties van x. Deze zijn dus voor het geval x geen teletype-protocol heeft.
<u>De tweede categorie</u> omvat die commando's, die het leven van de operator eenvoudiger maken. De functies, die hiertoe aanwezig zijn, zijn de volgende:

- Het beeldscherm in "Page-mode" gebruiken, dus na iedere 16 regels wordt niet meer op het scherm geschreven, totdat er een toets van het toetsenbord ingedrukt wordt. Deze toets geldt dan niet als te verzenden data.
  Alle data, die voor het beeldscherm bedoeld is, wordt in een buffer opgeslagen zolang er geen toets ingedrukt is.

- De hardcopyprinter aanzetten. Alles wat op het beeldscherm geschreven wordt, komt dan ook op de printer terecht.
- De hardcopyprint onderdrukken. Alles wat geprint moet worden, komt wel in een buffer terecht, maar het wordt nog niet geprint. Dit heeft men als bijv. het inktlint vastloopt o.i.d.
- De status van de terminal opvragen. Hierbij wordt de stand van alle "schakelaars" op het beeldscherm (en nooit op de printer) uitgeschreven en de hoeveelheid RAM, die nog vrij is.

Welke commando' zijn er?

Alle commando's vallen in de reeks van de control-characters. Dit zijn de tekens, die gevormd worden door de CTRL-toets samen met een andere toets aan te slaan.

- CTRL/R Dit bedient de software reader-run schakelaar. Als de softwareschakelaar en het hardwaresignaal ongelijk zijn, loopt de ponsbandlezer. CTRL/R zet de schakelaar aan als hij uit staat en uit als hij aan staat.
- CTRL/P Dit flipt een softwareschakelaar om, die als hij aan staat, alle data die voor het beeldscherm bedoeld is, naar de bandponser stuurt.
- CTRL/D Dit flipt een softwareschakelaar om, die bepaalt of het beeldscherm in Page-mode staat of continu.
- CTRL/E De schakelaar hierachter bepaalt of een teken, dat op het toetsenbord aangeslagen wordt, door de terminalsoftware ook op het scherm komt of niet.
- CTRL/H Zet de hardcopyprinter aan/uit.
- CTRL/Ø Onderdruk alle output van de printerbuffer naar de printer. Dit blijft gelden, tot weer CTRL/Ø gedrukt wordt.

Hoe wordt het RAM-geheugen als buffer gebruikt?

Aangezien meerere buffers gebruikt worden om snelheidsverschillen te overbruggen en aangezien niet te voorspellen valt, hoe de grootteverhouding van de diverse buffers ligt, wordt geheugenruimte programmatisch beheerd en uitgedeeld op aanvraag. De maximale buffergrootte is dus voor alle buffers de maximale hoeveelheid RAM die er is.

De beschikbare RAM wordt uitgedeeld naar behoefte in stukken van 1 page. (256 bytes) Als een uitgedeeld stuk, dat dus bij een bepaalde buffer gaat horen, niet meer nodig is, wordt het teruggegeven.
De buffers zijn gestructureerd als z.g. "FIFØ-lists".
(FIFØ = First IN, First Out)

Wat doet IØKIM als er meer bufferruimte nodig is, dan er beschikbaar is?

In deze afschuwelijke situatie wordt overgeschakeld op een nood-programma, dat op het beeldscherm meedeelt, dat de situatie optreedt en dan de buffers die data voor ponser en teletype bevatten leegmaakt. Hierna begint alles opnieuw. Er zijn echter voorzorgen getroffen om te verhinderen, dat deze situatie optreedt.

Als er deze z.g. "Bufferoverflow" optreedt, komt dit, omdat X teveel data gestuurd heeft. In de meeste gevallen zal dit zijn, omdat X teveel inputdata gehad heeft. M.a.w. als we X geen input meer geven, zal op een gegeven ogenblik de output vanzelf ophouden. Bij IØKIM wordt dan ook geen data meer geaccepteerd vanaf ponsbandlezer en toetsenbord, zodra de hoeveelheid vrije ruimte beneden een bepaald minimum (10 %) gedaald is. In de praktijk blijkt dit goed te voldoen.

Hoe is IØKIM in staat om de diverse apparaten gelijktijdig aan te drijven?

Alle randapparatuur wordt aangedreven per interrupt.
De interrupt wordt verzorgd door de timer van de applicatie-PIA.
Iedere 206 microsecondes komt een timerinterrupt. Dit representeert een kwart van de tijd voor één bit van de seriële interface. De bittijd is dus 824 microsecondes, wat betekent, dat de hoogst mogelijke transmissiesnelheid 1200 Baud bedraagt.

De hierbij optredende fout is ongeveer 1 %, wat bij asynchrone transmissie geen enkel probleem geeft. Zie voor details "Timing en flowsheet of asynchronous receiver".
Lagere transmissiesnelheden kunnen toegepast worden door het aantal klokcycli per bit te verhogen.

1200 Baud = 4 cycli
600 Baud = 8 cycli
300 Baud = 16 cycli
150 Baud = 32 cycli
110 Baud = 44 cycli

Een asynchrone transmitter werkt volgens hetzelfde principe. De transmitter is eenvoudiger, omdat er geen testen inzitten. Het teken wordt eenvoudig iedere keer uitgeschoven.

De ponsbandlezer en ponser zullen hier niet nader behandeld worden, omdat ze vrij specifiek zijn i.v.m. foutdedektie.

Hoe is de KIM bezet bij gebruik als IØKIM?

In feite is de KIM 100 % bezet als het interrupt programma evenlang duurt als de tijd tussen 2 opeenvolgende interrupts.
Dit betekent dan, dat de interruptafhandeling alle tijd opslokt en er geen tijd meer beschikbaar is voor het hoofdprogramma.
Zowel metingen als berekeningen hebben uitgewezen, dat in deze toepassing de KIM bij 1200 Baud nog onder 100% interruptbezetting blijft.
Als volgt:

Algemene interruptafhandeling	50 microsecondes
2 X asynchrone receiver	80 microsecondes
3 X asynchrone transmitter	120 microsecondes

Dit brengt het totaal op 250 microsecondes iedere 206 microsecondes. Dit gaat niet. Daarom worden de transmitters in de tijd verschoven bediend. Het totaal komt dan op 170 microsecondes per 206 microseconds = 82 % bezetting.

Voor het hoofdprogramma is dan 20 % tijd over. Dit heeft hetzelfde effect alsof de klok van de processor op 200 Khz i.p.v. 1 Mhz loopt.

Wat doet het hoofdprogramma?

Terwijl het interruptprogramma zorgt voor de afhandeling van één teken, knoopt het hoofdprogramma a.h.w. de touwtjes aan elkaar.

Dus : - Kijk of er een teken van het toetsenbord is.
- Als er één is, bekijk of het een controlcharacter is zoals ↑H.
- Is het een controlcharacter, doe dan wat gedaan moet worden. Anders wordt het in een buffer gestopt.

Het hoofdprogramma zorgt dus voor bufferen van tekens en weer uitsturen, als dat kan.

Wat zijn de opvallendste eigenschappen van een dergelijke terminal?

- Ten eerste de "gespoolde" hardcopy. Dit wil zeggen, dat er tekens binnenkomen op 1200 Baud, naar het display gaan op 1200 Baud, in de teletypebuffer gestopt worden en van daaruit geprint worden op 110 Baud.
  Als dus bijv. op computer X geassembleerd wordt, kan alvast met testen of iets dergelijks begonnen worden, terwijl de listing nog bezig is geprint te worden. Een buffer van 20 K blijkt in de praktijk voor listings van 10 à 15 bladzijden ruimschoots voldoende te zijn.
- Ten tweede het onderdrukken van de printout. Dit blijkt het meeste nut te hebben als de telefoon gaat, terwijl de teletype lawaai staat te maken. Bij het onderdrukken gaat dan alles normaal door, behalve de teletype, die tijdelijk uitgezet wordt.

Zijn er nog speciale voorzieningen voor buffering aangebracht?

De grootste buffer zal in het algemeen nodig zijn voor het langzaamste apparaat. Dit is de teletype. De data voor de teletype wordt dan ook "gepackt". Dit wil zeggen, dat als hetzelfde teken 2 maal of meer achter elkaar voorkomt, niet ieder teken afzonderlijk opgeslagen wordt, maar een telling van het aantal malen en het teken.
Dit bewijst vooral zijn nut bij listings, waar altijd lange series spaties in voorkomen.

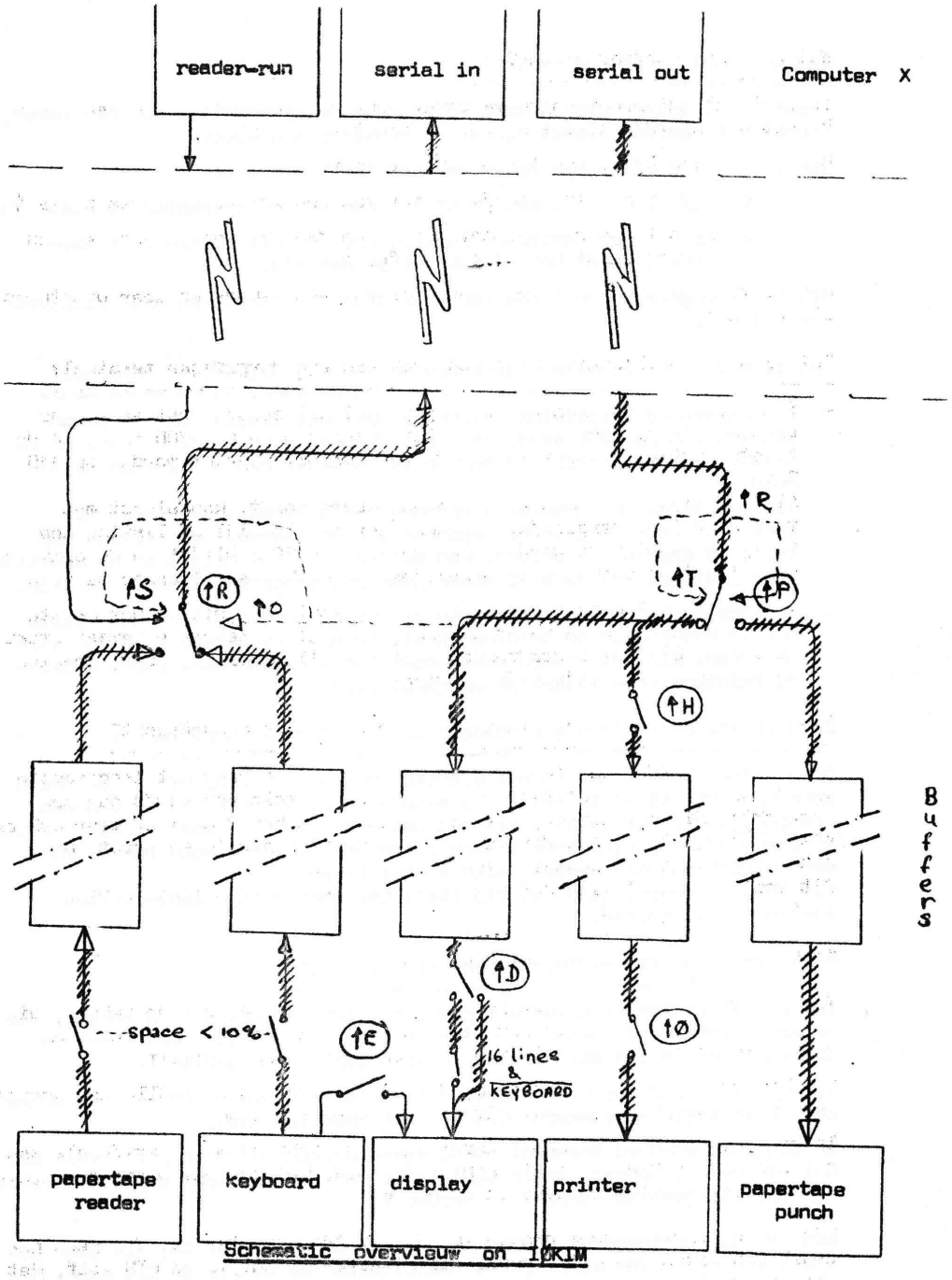
Welke verdere uitbreidingsplannen zijn er nog?

Ten eerste zullen er computers zijn, die erop staan, dat de tekens, die ernaar toegestuurd worden een bepaalde pariteit hebben. Voor diverse computers komen dan ook diverse versies zoals even pariteit.

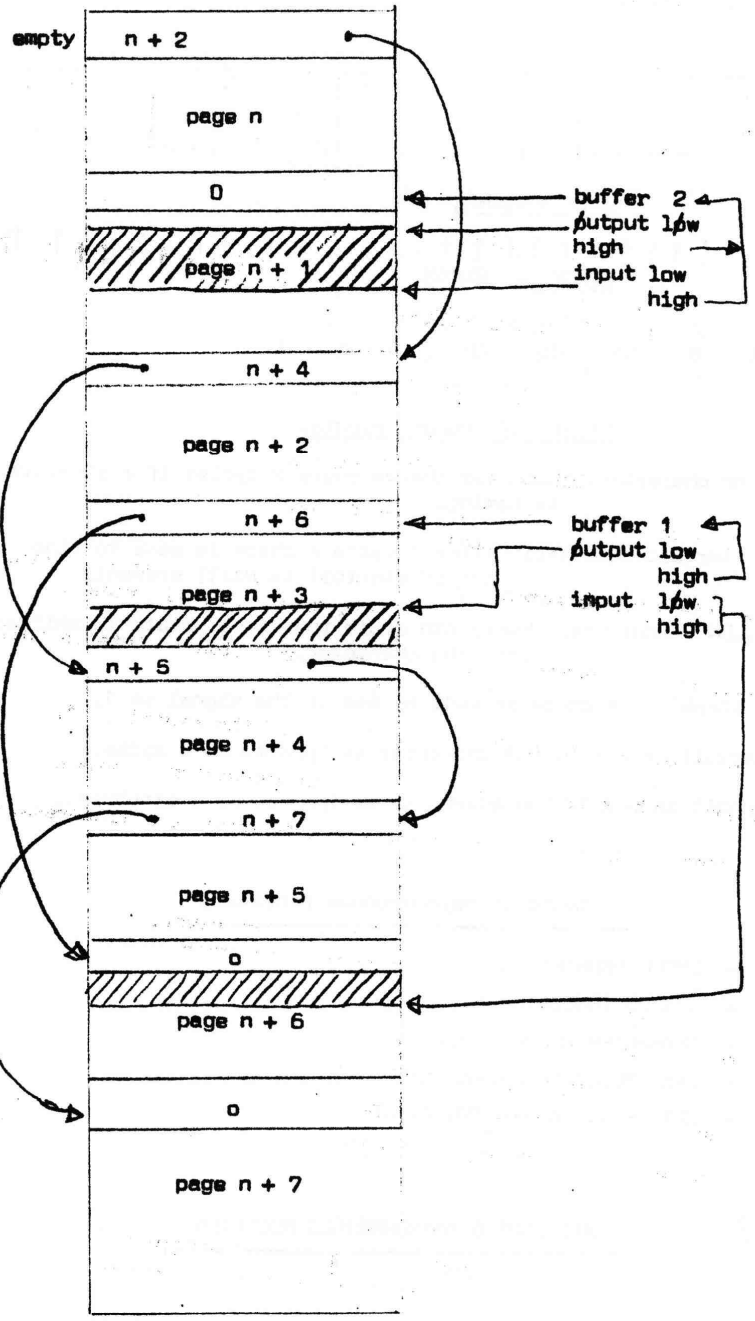
Verder wordt nog voor computers, die een andere dan de ASCII-code voeren, conversie ingeprogrammeerd. Dit zal via tabellen gaan.

In een iets verdere toekomst wordt eraan gedacht om meer periferals aan het systeem te hangen, zoals schijfgeheugen. Het systeem wordt dan a.h.w. een operatingsystem verpakt in hardware.

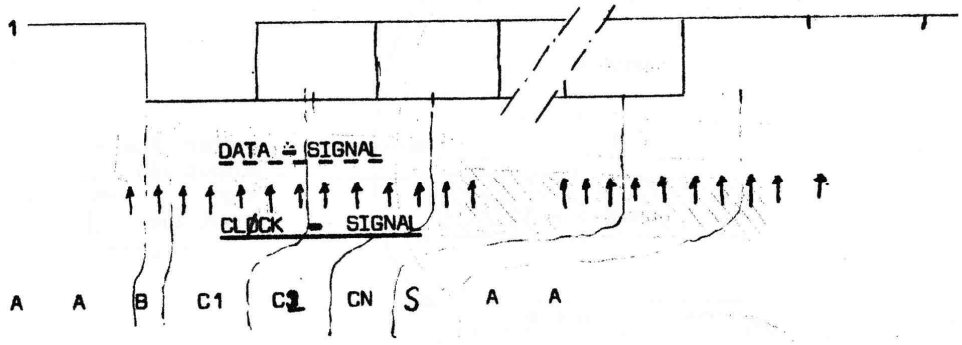
Eén van de frustrerende dingen is, dat er één computer is, die geen gebruik kan maken van alle fraaie faciliteiten en dat is de KIM zelf. Het plan bestaat dan ook om het systeem zodanig om te schrijven, dat niet alleen een extern, via hardware aangesloten computer IØKIM kan gebruiken, maar ook een softwaregekoppeld programma. Hiertoe moet dan met meer dan 2 niveau's van executie gewerkt worden.



Schematic overview on 1ØKIM



Example of bufferallocation



FAZES of receiverlogic.

A = no character. Receiver checks every 2 cycles if a startbit is coming.

B = startbit detected. After 1 cycle a check is made to find out if startbit is still present.

C1.....CN = databit. Every 4th cycle the signal state is shifted into the character.

S = stopbit. A check is made to see if the signal is 1.

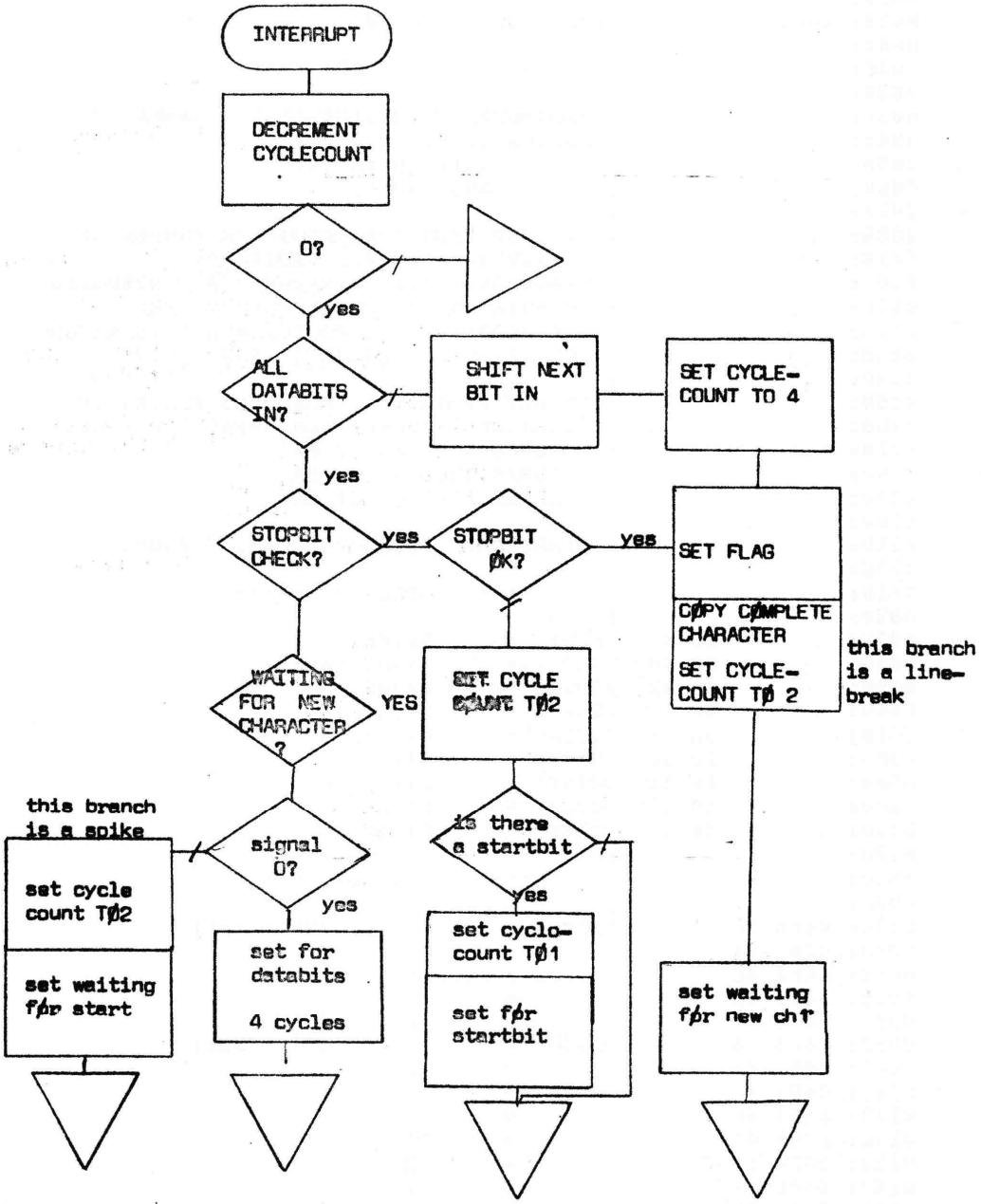
If startbit is not 0, the character is ignored as a spike.

If stopbit is not 1, the character is ignored as a linebreak.

Timing of asynchronous receiver.

CNT = CYCLE COUNTER
STAT = STATUS INDICATOR
CHR = CHARACTER BEING RECEIVED
CHAR = LAST RECEIVED CHARACTER
FLAG = SET IF NEW CHARACTER IN

ATTRIBUTES ASYNCHRONOUS RECEIVER



FLOWSHEET ASYNCHRØNØUS RECEIVER

COMPGM WESTVRIES COMPUTER CONSULTING PAGE 01

```
0010:                   ;      *****  FILE 01  *****
0020:                   ;
0030: 0400              PRMCOM ORG    $0400
0040:                   ;
0010:                   ;      *****  FILE 02  *****
0020:                   ;
0030:                   ; PROGMOVER -COMMUNICATIE-PROGRAMMA
0040:                   ; AUTHOR S. T. WOLDRINGH
0050:                   ;        KLIEVERINK 619
0060:                   ;        AMSTERDAM.
0070:                   ;
0080:                   ; VIA HET PROGRAMMA PROG-COM KUNNEN DE
0090:                   ;  GEGEVENS VOOR HET PROGRAMMA
0100:                   ;  PROGMOVER (ZIE KIMKENNER-4) INGEVOERD
0110:                   ;  WORDEN. DE GEGEVENS WORDEN DEELS
0120:                   ;  GEVALIDEERD, BY FOUTEN WORDT NOGMAALS
0130:                   ;  OM DEZELFDE GEGEVENS GEVRAAGD.
0140:                   ;
0150:                   ; OP HET PROGRAMMA PROGMOVER MOETEN DE
0160:                   ;  VOLGENDE PATCHES AANGEBRACHT WORDEN:
0170:                   ;   ADRES 03DA WORDT 60
0180:                   ;   ADRES 03DB WORDT EA
0190:                   ;   ADRES 03DC WORDT EA
0200:                   ;
0210:                   ; STARTADRES PROGMOVERCOMM. : 0600.
0220:                   ;
0010:                   ;      *****  FILE 03  *****
0020:                   ;
0030:          00 00    INPUT  *      $0000
0040:          0C 00    HULP   *      INPUT  +0C
0050:          00 02    PROGMV *      $0200
0060:          2F 1E    CRLF   *      $1E2F
0070:          5A 1E    GETCH  *      $1E5A
0080:          9D 1F    GETBYT *      $1F9D
0090:          9E 1E    OUTSP  *      $1E9E
0100:          A0 1E    OUTCH  *      $1EA0
0110:          4F 1C    START  *      $1C4F
0120:                   ;
0010:                   ;      *****  FILE 04  *****
0020:                   ;
0030: 0400 48           REG001 =      'H      (05 = $05)
0040: 0401 41                  =      'A
0050: 0402 4C                  =      'L
0060: 0403 4C                  =      'L
0070: 0404 4F                  =      'O
0080: 0405 54           REG002 =      'T      (32 = $20)
0090: 0406 49                  =      'I
0100: 0407 4B                  =      'K
0110: 0408 20                  =      '
0120: 0409 47                  =      'G
0130: 040A 45                  =      'E
0140: 040B 47                  =      'G
0150: 040C 45                  =      'E
0160: 040D 56                  =      'V
0170: 040E 45                  =      'E
0180: 040F 4E                  =      'N
```

```
0190: 0410 53                   =       'S
0200: 0411 20                   =       '
0210: 0412 49                   =       'I
0220: 0413 4E                   =       'N
0230: 0414 20                   =       '
0240: 0415 56                   =       'V
0250: 0416 4F                   =       'O
0260: 0417 4F                   =       'O
0270: 0418 52                   =       'R
0280: 0419 20                   =       '
0290: 041A 4D                   =       'M
0300: 041B 4F                   =       'O
0310: 041C 56                   =       'V
0320: 041D 45                   =       'E
0330: 041E 4E                   =       'N
0340: 041F 20                   =       '
0350: 0420 50                   =       'P
0360: 0421 52                   =       'R
0370: 0422 4F                   =       'O
0380: 0423 47                   =       'G
0390: 0424 2E                   =       '.
0400: 0425 49           REG003  =       'I      (40 = $28)
0410: 0426 53                   =       'S
0420: 0427 20                   =       '
0430: 0428 48                   =       'H
0440: 0429 45                   =       'E
0450:                   ;
0010:                   ;       *****  FILE 05  *****
0020:                   ;
0030: 042A 54                   =       'T
0040: 042B 20                   =       '
0050: 042C 50                   =       'P
0060: 042D 52                   =       'R
0070: 042E 4F                   =       'O
0080: 042F 47                   =       'G
0090: 0430 52                   =       'R
0100: 0431 41                   =       'A
0110: 0432 4D                   =       'M
0120: 0433 4D                   =       'M
0130: 0434 41                   =       'A
0140: 0435 20                   =       '
0150: 0436 52                   =       'R
0160: 0437 45                   =       'E
0170: 0438 45                   =       'E
0180: 0439 44                   =       'D
0190: 043A 53                   =       'S
0200: 043B 20                   =       '
0210: 043C 56                   =       'V
0220: 043D 45                   =       'E
0230: 043E 52                   =       'R
0240: 043F 50                   =       'P
0250: 0440 4C                   =       'L
0260: 0441 41                   =       'A
0270: 0442 41                   =       'A
0280: 0443 54                   =       'T
0290: 0444 53                   =       'S
```

```
0300: 0445 54                =      'T
0310: 0446 20                =      '
0320: 0447 28                =      '(
0330: 0448 59                =      'Y
0340: 0449 2F                =      '/
0350: 044A 4E                =      'N
0360: 044B 29                =      ')
0370: 044C 3F                =      '?
0380: 044D 4F          REG004 =      'O      (02 = $02)
0390: 044E 4B                =      'K
0400: 044F 54          REG005 =      'T      (22 = $16)
0410: 0450 49                =      'I
0420: 0451 4B                =      'K
0430: 0452 20                =      '
0440:                  ;
0010:                  ;       ***** FILE 06  *****
0020:                  ;
0030: 0453 44                =      'D
0040: 0454 45                =      'E
0050: 0455 20                =      '
0060: 0456 56                =      'V
0070: 0457 4F                =      'O
0080: 0458 4C                =      'L
0090: 0459 47                =      'G
0100: 045A 45                =      'E
0110: 045B 4E                =      'N
0120: 045C 44                =      'D
0130: 045D 45                =      'E
0140: 045E 20                =      '
0150: 045F 47                =      'G
0160: 0460 45                =      'E
0170: 0461 47                =      'G
0180: 0462 20                =      '
0190: 0463 49                =      'I
0200: 0464 4E                =      'N
0210: 0465 41          REG006 =      'A      (47 = $2F)
0220: 0466 4C                =      'L
0230: 0467 53                =      'S
0240: 0468 4F                =      'O
0250: 0469 46                =      'F
0260: 046A 20                =      '
0270: 046B 48                =      'H
0280: 046C 45                =      'E
0290: 046D 54                =      'T
0300: 046E 20                =      '
0310: 046F 50                =      'P
0320: 0470 52                =      'R
0330: 0471 4F                =      'O
0340: 0472 47                =      'G
0350: 0473 52                =      'R
0360: 0474 41                =      'A
0370: 0475 4D                =      'M
0380: 0476 4D                =      'M
0390: 0477 41                =      'A
0400: 0478 20                =      '
0410: 0479 4E                =      'N
```

PRMCOM WESTVRIES COMPUTER CONSULTING PAGE 04

```
0420: 047A 4F                =      'O
0430: 047B 47                =      'G
0440: 047C 20                =      '
0450:                ;
0010:                ;       *****  FILE 07  *****
0020:                ;
0030: 047D 56                =      'V
0040: 047E 45                =      'E
0050: 047F 52                =      'R
0060: 0480 50                =      'P
0070: 0481 4C                =      'L
0080: 0482 41                =      'A
0090: 0483 41                =      'A
0100: 0484 54                =      'T
0110: 0485 53                =      'S
0120: 0486 54                =      'T
0130: 0487 20                =      '
0140: 0488 4D                =      'M
0150: 0489 4F                =      'O
0160: 048A 45                =      'E
0170: 048B 53                =      'S
0180: 048C 54                =      'T
0190: 048D 20                =      '
0200: 048E 57                =      'W
0210: 048F 4F                =      'O
0220: 0490 52                =      'R
0230: 0491 44                =      'D
0240: 0492 45                =      'E
0250: 0493 4E                =      'N
0260: 0494 4F          REG007 =      'O      (39 = $27)
0270: 0495 50                =      'P
0280: 0496 4C                =      'L
0290: 0497 4F                =      'O
0300: 0498 50                =      'P
0310: 0499 45                =      'E
0320: 049A 4E                =      'N
0330: 049B 44                =      'D
0340: 049C 20                =      '
0350: 049D 4F                =      'O
0360: 049E 46                =      'F
0370: 049F 20                =      '
0380: 04A0 41                =      'A
0390: 04A1 46                =      'F
0400: 04A2 4C                =      'L
0410: 04A3 4F                =      'O
0420: 04A4 50                =      'P
0430: 04A5 45                =      'E
0440: 04A6 4E                =      'N
0450: 04A7 44                =      'D
0460: 04A8 20                =      '
0470:                ;
0010:                ;       *****  FILE 08  *****
0020:                ;
0030: 04A9 56                =      'V
0040: 04AA 45                =      'E
0050: 04AB 52                =      'R
```

```
0060: 04AC 50                =      'P
0070: 04AD 4C                =      'L
0080: 04AE 41                =      'A
0090: 04AF 41                =      'A
0100: 04B0 54                =      'T
0110: 04B1 53                =      'S
0120: 04B2 45                =      'E
0130: 04B3 4E                =      'N
0140: 04B4 20                =      '
0150: 04B5 28                =      '(
0160: 04B6 4F                =      'O
0170: 04B7 2F                =      '/
0180: 04B8 41                =      'A
0190: 04B9 29                =      ')
0200: 04BA 3F                =      '?
0210: 04BB 4C        REG008  =      'L      (26 = $1A)
0220: 04BC 41                =      'A
0230: 04BD 41                =      'A
0240: 04BE 47                =      'G
0250: 04BF 53                =      'S
0260: 04C0 54                =      'T
0270: 04C1 45                =      'E
0280: 04C2 20                =      '
0290: 04C3 4F                =      'O
0300: 04C4 4E                =      'N
0310: 04C5 54                =      'T
0320: 04C6 56                =      'V
0330: 04C7 41                =      'A
0340: 04C8 4E                =      'N
0350: 04C9 47                =      'G
0360: 04CA 45                =      'E
0370: 04CB 4E                =      'N
0380: 04CC 44                =      'D
0390: 04CD 45                =      'E
0400: 04CE 20                =      '
0410: 04CF 41                =      'A
0420: 04D0 44                =      'D
0430: 04D1 52                =      'R
0440: 04D2 45                =      'E
0450:                ;
0010:                ;      *****  FILE 09  *****
0020:                ;
0030: 04D3 53                =      'S
0040: 04D4 3F                =      '?
0050: 04D5 4C        REG009  =      'L      (20 = $14)
0060: 04D6 41                =      'A
0070: 04D7 41                =      'A
0080: 04D8 47                =      'G
0090: 04D9 53                =      'S
0100: 04DA 54                =      'T
0110: 04DB 45                =      'E
0120: 04DC 20                =      '
0130: 04DD 41                =      'A
0140: 04DE 44                =      'D
0150: 04DF 52                =      'R
0160: 04E0 45                =      'E
```

```
0170: 04E1 53                =     'S
0180: 04E2 20                =     '
0190: 04E3 50                =     'P
0200: 04E4 52                =     'R
0210: 04E5 4F                =     'O
0220: 04E6 47                =     'G
0230: 04E7 20                =     '
0240: 04E8 3F                =     '?
0250: 04E9 48          REG010 =     'H      (23 = $17)
0260: 04EA 4F                =     'O
0270: 04EB 4F                =     'O
0280: 04EC 47                =     'G
0290: 04ED 53                =     'S
0300: 04EE 54                =     'T
0310: 04EF 45                =     'E
0320: 04F0 20                =     '
0330: 04F1 2B                =     '+
0340: 04F2 20                =     '
0350: 04F3 31                =     '1
0360: 04F4 20                =     '
0370: 04F5 41                =     'A
0380: 04F6 44                =     'D
0390: 04F7 52                =     'R
0400: 04F8 45                =     'E
0410: 04F9 53                =     'S
0420: 04FA 20                =     '
0430: 04FB 50                =     'P
0440:                  ;
0010:                  ;     ***** FILE 10  *****
0020:                  ;
0030: 04FC 52                =     'R
0040: 04FD 4F                =     'O
0050: 04FE 47                =     'G
0060: 04FF 3F                =     '?
0070: 0500 47          REG011 =     'G      (16 = $10)
0080: 0501 4F                =     'O
0090: 0502 4F                =     'O
0100: 0503 44                =     'D
0110: 0504 42                =     'B
0120: 0505 59                =     'Y
0130: 0506 45                =     'E
0140: 0507 20                =     '
0150: 0508 46                =     'F
0160: 0509 4F                =     'O
0170: 050A 52                =     'R
0180: 050B 20                =     '
0190: 050C 4E                =     'N
0200: 050D 4F                =     'O
0210: 050E 57                =     'W
0220: 050F 21                =     '!
0230: 0510 4D          REG012 =     'M      (49 = $31)
0240: 0511 4F                =     'O
0250: 0512 45                =     'E
0260: 0513 54                =     'T
0270: 0514 45                =     'E
0280: 0515 4E                =     'N
```

```
0290: 0516 20                =      '
0300: 0517 44                =      'D
0310: 0518 45                =      'E
0320: 0519 20                =      '
0330: 051A 41                =      'A
0340: 051B 42                =      'B
0350: 051C 53                =      'S
0360:                ;
0010:                ;      *****  FILE 11  *****
0020:                ;
0030: 051D 20                =      '
0040: 051E 49                =      'I
0050: 051F 4E                =      'N
0060: 0520 53                =      'S
0070: 0521 54                =      'T
0080: 0522 52                =      'R
0090: 0523 55                =      'U
0100: 0524 43                =      'C
0110: 0525 54                =      'T
0120: 0526 49                =      'I
0130: 0527 45                =      'E
0140: 0528 53                =      'S
0150: 0529 20                =      '
0160: 052A 41                =      'A
0170: 052B 41                =      'A
0180: 052C 4E                =      'N
0190: 052D 47                =      'G
0200: 052E 45                =      'E
0210: 052F 50                =      'P
0220: 0530 41                =      'A
0230: 0531 53                =      'S
0240: 0532 54                =      'T
0250: 0533 20                =      '
0260: 0534 57                =      'W
0270: 0535 4F                =      'O
0280: 0536 52                =      'R
0290: 0537 44                =      'D
0300: 0538 45                =      'E
0310: 0539 4E                =      'N
0320: 053A 20                =      '
0330: 053B 28                =      '(
0340: 053C 59                =      'Y
0350: 053D 2F                =      '/
0360: 053E 4E                =      'N
0370: 053F 29                =      ')
0380: 0540 3F                =      '?
0390: 0541 41          REG013 =      'A      (26 = $1A)
0400: 0542 44                =      'D
0410: 0543 52                =      'R
0420: 0544 45                =      'E
0430: 0545 53                =      'S
0440: 0546 20                =      '
0450:                ;
0010:                ;      *****  FILE 12  *****
0020:                ;
0030: 0547 45                =      'E
```

PRMCOM WESTVRIES COMPUTER CONSULTING PAGE 08

```
0040: 0548 45                =      'E
0050: 0549 52                =      'R
0060: 054A 53                =      'S
0070: 054B 54                =      'T
0080: 054C 45                =      'E
0090: 054D 20                =      '
0100: 054E 49                =      'I
0110: 054F 4E                =      'N
0120: 0550 53                =      'S
0130: 0551 54                =      'T
0140: 0552 52                =      'R
0150: 0553 55                =      'U
0160: 0554 43                =      'C
0170: 0555 54                =      'T
0180: 0556 49                =      'I
0190: 0557 45                =      'E
0200: 0558 20                =      '
0210: 0559 3F                =      '?
0220: 055A 20                =      '
0230: 055B 4C        REG014  =      'L      (35 = $23)
0240: 055C 41                =      'A
0250: 055D 41                =      'A
0260: 055E 54                =      'T
0270: 055F 53                =      'S
0280: 0560 54                =      'T
0290: 0561 45                =      'E
0300: 0562 20                =      '
0310: 0563 2B                =      '+
0320: 0564 20                =      '
0330: 0565 31                =      '1
0340: 0566 20                =      '
0350: 0567 41                =      'A
0360: 0568 44                =      'D
0370: 0569 52                =      'R
0380: 056A 45                =      'E
0390: 056B 53                =      'S
0400: 056C 20                =      '
0410: 056D 56                =      'V
0420: 056E 41                =      'A
0430: 056F 4E                =      'N
0440: 0570 20                =      '
0450:                ;
0010:                ;       ***** FILE 13  *****
0020:                ;
0030: 0571 49                =      'I
0040: 0572 4E                =      'N
0050: 0573 53                =      'S
0060: 0574 54                =      'T
0070: 0575 52                =      'R
0080: 0576 55                =      'U
0090: 0577 43                =      'C
0100: 0578 54                =      'T
0110: 0579 49                =      'I
0120: 057A 45                =      'E
0130: 057B 53                =      'S
0140: 057C 20                =      '
```

PRMCOM WESTVRIES COMPUTER CONSULTING PAGE 09

```
0150: 057D 3F              =      '?
0160: 057E 49        REG015 =      'I      (37 = $25)
0170: 057F 4D              =      'M
0180: 0580 4D              =      'M
0190: 0581 45              =      'E
0200: 0582 44              =      'D
0210: 0583 49              =      'I
0220: 0584 41              =      'A
0230: 0585 54              =      'T
0240: 0586 45              =      'E
0250: 0587 20              =      '
0260: 0588 49              =      'I
0270: 0589 4E              =      'N
0280: 058A 53              =      'S
0290: 058B 54              =      'T
0300: 058C 52              =      'R
0310: 058D 55              =      'U
0320: 058E 43              =      'C
0330: 058F 54              =      'T
0340: 0590 49              =      'I
0350: 0591 45              =      'E
0360: 0592 53              =      'S
0370: 0593 20              =      '
0380: 0594 50              =      'P
0390: 0595 52              =      'R
0400: 0596 49              =      'I
0410: 0597 4E              =      'N
0420: 0598 54              =      'T
0430: 0599 45              =      'E
0440: 059A 4E              =      'N
0450:                ;
0010:                ;     ***** FILE 14 *****
0020:                ;
0030: 059B 20              =      '
0040: 059C 28              =      '(
0050: 059D 59              =      'Y
0060: 059E 2F              =      '/
0070: 059F 4E              =      'N
0080: 05A0 29              =      ')
0090: 05A1 20              =      '
0100: 05A2 3F              =      '?
0110: 05A3 46        REG016 =      'F      (14 = $0E)
0120: 05A4 4F              =      'O
0130: 05A5 55              =      'U
0140: 05A6 54              =      'T
0150: 05A7 45              =      'E
0160: 05A8 4E              =      'N
0170: 05A9 20              =      '
0180: 05AA 49              =      'I
0190: 05AB 4E              =      'N
0200: 05AC 20              =      '
0210: 05AD 47              =      'G
0220: 05AE 45              =      'E
0230: 05AF 47              =      'G
0240: 05B0 21              =      '!
0250: 05B1 4F        REG017 =      'O      (14 = $0E)
```

```
0260: 05B2 50              =     'P
0270: 05B3 4E              =     'N
0280: 05B4 49              =     'I
0290: 05B5 45              =     'E
0300: 05B6 55              =     'U
0310: 05B7 57              =     'W
0320: 05B8 20              =     '
0330: 05B9 28              =     '(
0340: 05BA 59              =     'Y
0350: 05BB 2F              =     '/
0360: 05BC 4E              =     'N
0370: 05BD 29              =     ')
0380: 05BE 3F              =     '?
0390: 05BF 53        REG018 =     'S
0400: 05C0 54               =     'T
0410: 05C1 41               =     'A
0420: 05C2 52               =     'R
0430: 05C3 54               =     'T
0440: 05C4 20               =     '
0450:                  ;
0010:                  ;     *****  FILE 15  *****
0020:                  ;
0030: 05C5 50        REG019 =     'P      (16 = $10)
0040: 05C6 52               =     'R
0050: 05C7 4F               =     'O
0060: 05C8 47               =     'G
0070: 05C9 4D               =     'M
0080: 05CA 4F               =     'O
0090: 05CB 56               =     'V
0100: 05CC 45               =     'E
0110: 05CD 52               =     'R
0120: 05CE 20               =     '
0130: 05CF 47               =     'G
0140: 05D0 45               =     'E
0150: 05D1 52               =     'R
0160: 05D2 45               =     'E
0170: 05D3 45               =     'E
0180: 05D4 44               =     'D
0190: 05D5 4E        REG020 =     'N      (15 = $0F)
0200: 05D6 4F               =     'O
0210: 05D7 47               =     'G
0220: 05D8 20               =     '
0230: 05D9 4D               =     'M
0240: 05DA 45               =     'E
0250: 05DB 45               =     'E
0260: 05DC 52               =     'R
0270: 05DD 20               =     '
0280: 05DE 28               =     '(
0290: 05DF 59               =     'Y
0300: 05E0 2F               =     '/
0310: 05E1 4E               =     'N
0320: 05E2 29               =     ')
0330: 05E3 3F               =     '?
0340: 05E4 54        REG021 =     'T      (18 = $12)
0350: 05E5 45               =     'E
0360: 05E6 52               =     'R
```

PRMCOM WESTVRIES COMPUTER CONSULTING PAGE 11

```
0370: 05E7 55                =      'U
0380: 05E8 47                =      'G
0390: 05E9 20                =      '
0400: 05EA 4E                =      'N
0410: 05EB 41                =      'A
0420: 05EC 41                =      'A
0430: 05ED 52                =      'R
0440: 05EE 20                =      '
0450: 05EF 4D                =      'M
0460: 05F0 4F                =      'O
0470: 05F1 4E                =      'N
0480: 05F2 49                =      'I
0490: 05F3 54                =      'T
0500: 05F4 4F                =      'O
0510: 05F5 52                =      'R
0520:                        ;
0010:                        ;     *****  FILE 16  *****
0020:                        ;
0030: 0600                   COMPGM ORG    $0600
0040:                        ;
0010:                        ;     *****  FILE 17  *****
0020:                        ;
0030: 0600 A2 0B             PGMCM0 LDXIM  $0B
0040: 0602 A9 00                    LDAIM  $00
0050: 0604 95 00             PGMCM1 STAZX  INPUT
0060: 0606 CA                       DEX
0070: 0607 10 FB                    BPL    PGMCM1
0080: 0609 20 2F 1E                 JSR    CRLF
0090: 060C 20 2F 1E                 JSR    CRLF
0100: 060F A0 05                    LDYIM  $05
0110: 0611 A2 00                    LDXIM  REG001 HALLO
0120: 0613 20 C0 07                 JSR    PRT1
0130: 0616 20 2F 1E                 JSR    CRLF
0140: 0619 A0 20                    LDYIM  $20
0150: 061B A2 05                    LDXIM  REG002 TIK GEG IN
0160: 061D 20 C0 07                 JSR    PRT1
0170: 0620 20 2F 1E                 JSR    CRLF
0180: 0623 20 2F 1E                 JSR    CRLF
0190: 0626 A0 28             PGMCM2 LDYIM  $28
0200: 0628 A2 25                    LDXIM  REG003 PROG REEDS VERPL
0210: 062A 20 C0 07                 JSR    PRT1
0220: 062D 20 5A 1E                 JSR    GETCH
0230: 0630 85 0C                    STAZ   HULP
0240: 0632 20 2F 1E                 JSR    CRLF
0250: 0635 A5 0C                    LDAZ   HULP
0260: 0637 C9 4E                    CMPIM  'N
0270: 0639 F0 28                    BEQ    PGMCM3
0280: 063B C9 59                    CMPIM  'Y
0290: 063D D0 E7                    BNE    PGMCM2
0300: 063F A0 02                    LDYIM  $02
0310: 0641 A2 4D                    LDXIM  REG004 OK
0320: 0643 20 C0 07                 JSR    PRT1
0330: 0646 20 2F 1E                 JSR    CRLF
0340: 0649 A0 16                    LDYIM  $16
0350: 064B A2 4F                    LDXIM  REG005 TIK VOLG GEG IN
0360: 064D 20 C0 07                 JSR    PRT1
```

COMPGM WESTVRIES COMPUTER CONSULTING PAGE 12

```
0370: 0650 20 2F 1E          JSR   CRLF
0380: 0653 A0 2F             LDYIM $2F
0390: 0655 A2 65             LDXIM REG006 ALSOF ETC ETC
0400: 0657 20 C0 07          JSR   PRT1
0410: 065A 20 2F 1E          JSR   CRLF
0420: 065D 20 2F 1E          JSR   CRLF
0430: 0660 4C 81 06          JMP   PGMCM5
0440: 0663 A0 27       PGMCM3 LDYIM $27
0450:                        ;
0010:                        ;     *****  FILE 18  *****
0020:                        ;
0030: 0665 A2 04             LDXIM REG007 OPL OF AFL
0040: 0667 20 C0 07          JSR   PRT1
0050: 066A 20 5A 1E          JSR   GETCH
0060: 066D 85 0C             STAZ  HULP
0070: 066F 20 2F 1E          JSR   CRLF
0080: 0672 A2 01             LDXIM $01
0090: 0674 A5 0C             LDAZ  HULP
0100: 0676 C9 4F             CMPIM 'O
0110: 0678 F0 05             BEQ   PGMCM4
0120: 067A C9 41             CMPIM 'A
0130: 067C D0 E5             BNE   PGMCM3
0140: 067E E8                INX
0150: 067F 86 00       PGMCM4 STXZ  INPUT
0160: 0681 A0 1A       PGMCM5 LDYIM $1A
0170: 0683 A2 BB             LDXIM REG008 LAAGST ONTV ADR
0180: 0685 20 C0 07          JSR   PRT1
0190: 0688 20 9D 1F          JSR   GETBYT
0200: 068B 85 02             STAZ  INPUT  +02
0210: 068D 20 9D 1F          JSR   GETBYT
0220: 0690 85 01             STAZ  INPUT  +01
0230: 0692 20 2F 1E          JSR   CRLF
0240: 0695 A0 14             LDYIM $14
0250: 0697 A2 D5             LDXIM REG009 LAAGST ADR PROG
0260: 0699 20 C0 07          JSR   PRT1
0270: 069C 20 9D 1F          JSR   GETBYT
0280: 069F 85 04             STAZ  INPUT  +04
0290: 06A1 20 9D 1F          JSR   GETBYT
0300: 06A4 85 03             STAZ  INPUT  +03
0310: 06A6 20 2F 1E          JSR   CRLF
0320: 06A9 A0 17             LDYIM $17
0330: 06AB A2 E9             LDXIM REG010 HOOGSTE + 1 ADR
0340: 06AD 20 C0 07          JSR   PRT1
0350: 06B0 20 9D 1F          JSR   GETBYT
0360: 06B3 85 06             STAZ  INPUT  +06
0370: 06B5 20 9D 1F          JSR   GETBYT
0380: 06B8 85 05             STAZ  INPUT  +05
0390: 06BA 20 2F 1E          JSR   CRLF
0400: 06BD 20 2F 1E          JSR   CRLF
0410: 06C0 A0 31       PGMCM6 LDYIM $31
0420: 06C2 A2 10             LDXIM REG012 ABS AANPASSEN?
0430: 06C4 20 D1 07          JSR   PRT2
0440:                        ;
0010:                        ;     *****  FILE 19  *****
0020:                        ;
0030: 06C7 20 5A 1E          JSR   GETCH
```

COMPGM WESTVRIES COMPUTER CONSULTING PAGE 13

```
0040: 06CA 85 0C            STAZ  HULP
0050: 06CC 20 2F 1E         JSR   CRLF
0060: 06CF A5 0C            LDAZ  HULP
0070: 06D1 C9 59            CMPIM 'Y
0080: 06D3 F0 11            BEQ   PGMCM7
0090: 06D5 C9 4E            CMPIM 'N
0100: 06D7 D0 E7            BNE   PGMCM6
0110: 06D9 A0 02            LDYIM $02
0120: 06DB A2 4D            LDXIM REG004 OK
0130: 06DD 20 C0 07         JSR   PRT1
0140: 06E0 20 2F 1E         JSR   CRLF
0150: 06E3 4C 2B 07         JMP   PGMCM9
0160: 06E6 A0 1A     PGMCM7 LDYIM $1A
0170: 06E8 A2 41            LDXIM REG013 ADRES EERSTE INSTR
0180: 06EA 20 D1 07         JSR   PRT2
0190: 06ED 20 9D 1F         JSR   GETBYT
0200: 06F0 85 08            STAZ  INPUT  +08
0210: 06F2 20 9D 1F         JSR   GETBYT
0220: 06F5 85 07            STAZ  INPUT  +07
0230: 06F7 20 2F 1E         JSR   CRLF
0240: 06FA A0 23            LDYIM $23
0250: 06FC A2 5B            LDXIM REG014 LAATSTE + 1
0260: 06FE 20 D1 07         JSR   PRT2
0270: 0701 20 9D 1F         JSR   GETBYT
0280: 0704 85 0A            STAZ  INPUT  +0A
0290: 0706 20 9D 1F         JSR   GETBYT
0300: 0709 85 09            STAZ  INPUT  +09
0310: 070B 20 2F 1E         JSR   CRLF
0320: 070E A0 25     PGMCM8 LDYIM $25
0330: 0710 A2 7E            LDXIM REG015 IMM PRINTEN
0340: 0712 20 D1 07         JSR   PRT2
0350: 0715 20 5A 1E         JSR   GETCH
0360: 0718 85 0C            STAZ  HULP
0370: 071A 20 2F 1E         JSR   CRLF
0380: 071D A5 0C            LDAZ  HULP
0390: 071F C9 4E            CMPIM 'N
0400: 0721 F0 08            BEQ   PGMCM9
0410: 0723 C9 59            CMPIM 'Y
0420: 0725 D0 E7            BNE   PGMCM8
0430: 0727 A9 01            LDAIM $01
0440: 0729 85 0B            STAZ  INPUT  +0B
0450: 072B A0 0F     PGMCM9 LDYIM $0F
0460:                       ;
0010:                       ;     *****  FILE 20  *****
0020:                       ;
0030: 072D A2 BF            LDXIM REG018 START PROGMOVER
0040: 072F 20 D1 07         JSR   PRT2
0050: 0732 20 2F 1E         JSR   CRLF
0060: 0735 20 2F 1E         JSR   CRLF
0070: 0738 20 00 02         JSR   PROGMV
0080: 073B 86 0C            STXZ  HULP
0090: 073D 20 2F 1E         JSR   CRLF
0100: 0740 A6 0C            LDXZ  HULP
0110: 0742 E0 11            CPXIM $11
0120: 0744 D0 26            BNE   PGMC11
0130: 0746 A0 0E            LDYIM $0E
```

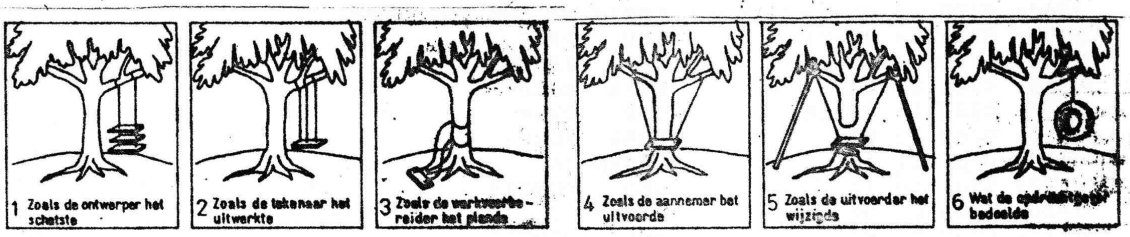
```
0140: 0748 A2 A3           LDXIM REG016 FOUTEN IN GEG
0150: 074A 20 D1 07        JSR   PRT2
0160: 074D 20 2F 1E        JSR   CRLF
0170: 0750 A0 0E     PGMC10 LDYIM $0E
0180: 0752 A2 B1           LDXIM REG017 OPNIEUW?
0190: 0754 20 D1 07        JSR   PRT2
0200: 0757 20 5A 1E        JSR   GETCH
0210: 075A 85 0C           STAZ  HULP
0220: 075C 20 2F 1E        JSR   CRLF
0230: 075F A5 0C           LDAZ  HULP
0240: 0761 C9 4E           CMPIM 'N
0250: 0763 F0 37           BEQ   PGMC13
0260: 0765 C9 59           CMPIM 'Y
0270: 0767 D0 E7           BNE   PGMC10
0280: 0769 4C 00 06        JMP   PGMCM0
0290: 076C A0 10     PGMC11 LDYIM $10
0300: 076E A2 C5           LDXIM REG019 PROGMOVER GEREED
0310: 0770 20 D1 07        JSR   PRT2
0320: 0773 20 2F 1E        JSR   CRLF
0330: 0776 A0 0F     PGMC12 LDYIM $0F
0340: 0778 A2 D5           LDXIM REG020 NOG MEER?
0350: 077A 20 D1 07        JSR   PRT2
0360: 077D 20 5A 1E        JSR   GETCH
0370: 0780 85 0C           STAZ  HULP
0380: 0782 20 2F 1E        JSR   CRLF
0390: 0785 A5 0C           LDAZ  HULP
0400: 0787 C9 4E           CMPIM 'N
0410: 0789 F0 11           BEQ   PGMC13
0420: 078B C9 59           CMPIM 'Y
0430: 078D D0 E7           BNE   PGMC12
0440: 078F A0 02           LDYIM $02
0450: 0791 A2 4D           LDXIM REG004 OK
0460: 0793 20 C0 07        JSR   PRT1
0470:                ;
0010:                ;     *****  FILE 21  *****
0020:                ;
0030: 0796 20 2F 1E        JSR   CRLF
0040: 0799 4C 00 06        JMP   PGMCM0
0050: 079C A0 02     PGMC13 LDYIM $02
0060: 079E A2 4D           LDXIM REG004 OK
0070: 07A0 20 C0 07        JSR   PRT1
0080: 07A3 20 2F 1E        JSR   CRLF
0090: 07A6 A0 12           LDYIM $12
0100: 07A8 A2 E4           LDXIM REG021
0110: 07AA 20 D1 07        JSR   PRT2
0120: 07AD 20 2F 1E        JSR   CRLF
0130: 07B0 A0 10           LDYIM $10
0140: 07B2 A2 00           LDXIM REG011 BYE
0150: 07B4 20 D1 07        JSR   PRT2
0160: 07B7 20 2F 1E        JSR   CRLF
0170: 07BA 20 2F 1E        JSR   CRLF
0180: 07BD 4C 4F 1C        JMP   START
0190:                ;
0200:                ;
0210: 07C0 84 0C     PRT1   STYZ  HULP
0220: 07C2 BD 00 04  PRT1A  LDAAX REG001
```

```
COMPGM    WESTVRIES COMPUTER CONSULTING PAGE 15

0230: 07C5 20 A0 1E          JSR   OUTCH
0240: 07C8 E8                INX
0250: 07C9 C6 0C             DECZ  HULP
0260: 07CB D0 F5             BNE   PRT1A
0270: 07CD 20 9E 1E          JSR   OUTSP
0280: 07D0 60                RTS
0290:                   ;
0300:                   ;
0310: 07D1 84 0C     PRT2    STYZ  HULP
0320: 07D3 BD 00 05  PRT2A   LDAAX REG011
0330: 07D6 20 A0 1E          JSR   OUTCH
0340: 07D9 E8                INX
0350: 07DA C6 0C             DECZ  HULP
0360: 07DC D0 F5             BNE   PRT2A
0370: 07DE 20 9E 1E          JSR   OUTSP
0380: 07E1 60                RTS
0390:                   ;
```

-

Als u begrijpt wat ik bedoel



T___T

```
SYMBOL TABLE 5000 512C
COMPGM 0600    CRLF   1E2F    GETBYT 1F9D    GETCH  1E5A
HULP   000C    INPUT  0000    OUTCH  1EA0    OUTSP  1E9E
PGMCMP 0600    PGMCMQ 0604    PGMCMR 0626    PGMCMS 0663
PGMCMT 067F    PGMCMU 0681    PGMCMV 06C0    PGMCMW 06E6
PGMCMX 070E    PGMCMY 072B    PGMCQP 0750    PGMCQQ 076C
PGMCQR 0776    PGMCQS 079C    PRMCOM 0400    PROGMV 0200
PRTQ   07C0    PRTQA  07C2    PRTR   07D1    PRTRA  07D3
REGPPQ 0400    REGPPR 0405    REGPPS 0425    REGPPT 044D
REGPPU 044F    REGPPV 0465    REGPPW 0494    REGPPX 04BB
REGPPY 04D5    REGPQP 04E9    REGPQQ 0500    REGPQR 0510
REGPQS 0541    REGPQT 055B    REGPQU 057E    REGPQV 05A3
REGPQW 05B1    REGPQX 05BF    REGPQY 05C5    REGPRP 05D5
REGPRQ 05E4    START  1C4F
```

-

KIM GEBRUIKERS CLUB NEDERLAND
HARDWARE LIBRARY

De Telex: een goedkope Teletype

Deel 2

In het eerste deel van deze serie van twee is beschreven, hoe een telex aangesloten kan worden op de KIM en zijn een paar testprogrammaatjes gegeven. Hier volgt een beschrijving van een monitorprogramma voor de Telex met zoveel kommentaar, dat U het na enige studie zelf kan uitbreiden of delen weglaten. Het programma is bijna ½K lang, inklusief alle subroutines.

Baudot versus ASCII

De KIM heeft een outputsubroutine OUTCH op geheugenplaats 1EAØ, die we vorige keer al gebruikt hebben. Die doet het volgende:

- eerst wordt op de serie-uitgang voor de Teletype of Telex een startbit gezet (een nul);
- vervolgens worden de 8 bits vanuit het A-register naar rechts uitgeklokt (ASCII code heeft 8 bits);
- tenslotte worden nog twee stopbits weggeschreven (2 enen).

Het totale patroon is dus

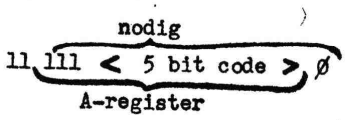
11 < 8 bit code > Ø,

waarbij de bits van rechts naar links worden weggeschreven.

Dat klopt niet helemaal met de Baudot-code. Die heeft een patroon nodig, dat er als volgt uitziet:

11 < 5 bit code > Ø.

We halen nu de volgende truc uit: we vullen de 5-bits code in het A-register links met 3 één-bits aan en schrijven dat weg. We krijgen dan het volgende:



Het effekt is, dat we 3 énen teveel wegschrijven. De Telex interpreteert die énen als een pauze, waarin niets gedaan wordt, zodat het resultaat is, dat de Telex niet op volle snelheid schrijft, maar met 73% daarvan. Bij het inlezen van karakters van de Telex naar de KIM gebeurt hetzelfde. Alleen als U Uw best doet zo snel mogelijk op één toets te hameren zult U weleens een fout gelezen karakter vinden, maar bij normaal gebruik gaat alles prima.

Letters versus cijfers

Een bepaalde Baudot-code kan twee betekenissen hebben, afhankelijk van de modo: letters of cijfers. Dat is nodig, omdat de code maar uit 5 bits bestaat. Dat betekent, dat de cijfers- en letterstoetsen nogal vaak aangeslagen zullen worden. Om nu te voorkomen dat we al die karakters moeten bewaren, doen we het volgende.
De gemodificeerde subroutine GETCH (zie later) haalt een karakter van de Telex op in het A-register, dat er dan als volgt uitziet:

Ø 11 < 5 bit code >

Als het karakter nu een cijfer is, maken we het hoogste bit een één, als het een letter is, blijft het een nul.
Verder spreken we voor het gemak af, dat intern in onze KIM we een Baudot-karakter maar zullen opslaan als een 7-bit code, waarvan de hoogste twee bits altijd énen zijn, plus het achtste cijfer/letter bit.

KIM subroutines

De volgende Kim subroutines worden gebruikt:

OUTCH (1EA0)	: print het karakter in A, dat er uit moet zien als 111 < 5 bit code > ; vernietigt A en Y
INITS (1E88)	: initialiseert de KIM: vernietigt X
OPEN (1F6C)	: laadt POINT uit input buffer; A := INH
INCPT (1F63)	: POINT := POINT +1
SAVEL (1C05)	: redt alle registers behalve A; springt naar KIM-monitor
GOEXEC (1DC8)	: herstelt registers; start programma vanaf POINT

POINT is een pointer (wijzer) naar de geheugenplaats, waar we mee bezig zijn.
INH is de inhoud (data) op die plaats.

Extra subroutines

De meeste subroutines, die nu volgen, zijn (geringe) wijzigingen van Kim subroutines. We bespreken ze stuk voor stuk. Maar eerst hebben we een tabel nodig van alle codes voor de hexadecimale cijfers.

```
0300   F6  F7  F3  E1      TABLE: ....; tabel van hex cijfers
       EA  FØ  F5  E7                 ; van Ø tot en met F
       E6  F8  63  79
       6E  69  61  6D
```

Het hoogste bit is een één als het een cijfer is, anders een nul. De volgende twee bits zijn altijd één.
Let op! De meeste subroutines vernietigen de inhoud van A en Y!

Subroutine HEXTA print het hexadecimale getal in de lage 4 bits van A als Telex code. Eerst wordt gekontroleerd of de mode klopt; zo nee, dan wordt de Telex eerst in de juiste mode gezet.

```
0310  29 ØF          HEXTA: AND #ØF      ; laagste 4 bit
      A8                   TAY
      B9 ØØ Ø3             LDA TABLE, Y ; haal karakter op
0316  48           OUTCHØ: PHA          ; bewaar
      C9 8Ø                CMP #8Ø
      1Ø ØF                BPL CIJFER
031B  A5 EE                LDA MODE2    ; het is een letter
      FØ 18                BEQ CONT
      A9 FF        LETTER: LDA #FF      ; zet telex goed
0321  2Ø AØ 1E             JSR OUTCH
      A9 ØØ                LDA #ØØ
      85 EE                STA MODE2    ; pas mode aan
      FØ ØD                BEQ CONT
      A5 EE        CIJFER: LDA MODE2    ; het is een cijfer
      3Ø Ø9                BMI CONT
      A9 FB                LDA #FB      ; zet telex goed
0330  2Ø AØ 1E             JSR OUTCH
      A9 8Ø                LDA #8Ø
      85 EE                STA MODE2    ; pas mode aan
      68           CONT  : PLA          ; haal karakter weer op
      Ø9 EØ                ORA #EØ      ; zet hoogste bit
      4C AØ 1E             JMP OUTCH    ; print
```

De volgende subroutine, PRBYT, gebruikt HEXTA tweemaal en print dus het A-register als twee hexadecimale cijfers.

```
033D  85 FC        PRBYT:  STA TEMP    ; redt A
      4A                   LSR A       ; neem hoogste 4 bit
      4A                   LSR A
      4A                   LSR A
      4A                   LSR A
0343  20 10 03             JSR HEXTA   ; print karakter
      A5 FC                LDA TEMP    ; herstel A
      20 10 03             JSR HEXTA   ; print karakter
      A5 FC                LDA TEMP    ; herstel A
034D  60                   RTS
```

Subroutine CRLF zorgt voor overgang naar een nieuwe regel. Ten behoeve van oude, niet meer zo goede telexen, wordt daarna nog een extra pauze ingelast om de wagen van de telex helemaal tot stilstand te laten komen.

```
034E  A9 E8        CRLF:   LDA #E8     ; terug wagen
      20 A0 1E             JSR OUTCH
0353  A9 E2                LDA #E2     ; nieuwe regel
      20 A0 1E             JSR OUTCH
      A9 E0                LDA #E0     ; pauze
      20 A0 1E             JSR OUTCH
      60                   RTS
```

De volgende subroutine laat de bel van de telex eenmaal klinken.

```
035E  A9 FB        BELL:   LDA #FB     ; bel
      4C 16 03             JMP OUTCH 0
```

Subroutine GETCH haalt een karakter van de telex op in het A-register. Het komt daarin te staan in de vorm 011 < 5 bit code >.

```
0363  86 FD        GETCH:  STX TMPX    ; bewaar X
      A2 08                LDX #08     ; 8 bits
      2C 40 17     GET1:   BIT SAD     ; wacht zonodig
      30 FB                BMI GET1
0369  4C 67 1E             JMP GET0
```

Als de vorige subroutine een karakter opgehaald heeft, kan subroutine PACK daarvan een hexadecimaal cijfer maken. Bovendien wordt dit dan in het input buffer geschoven. A wordt nul als alles goed gegaan is. Was de code niet van een hex cijfer, dan is in A alleen het juiste mode bit geplaatst.

```
036F  05 EE          PACK: ORA MODE2     ; voeg mode bit toe
      A0 0F                LDY #0F
      D9 00 03       LOOP: CMP TABLE, Y  ; zoek in tabel
      F0 04                BEQ EXIT1
      88                   DEY
      10 F8                BPL LOOP
      60                   RTS           ; niet in tabel
      98             EXIT1:TYA           ; Y is tabel nummer
      2A                   ROL A
      2A                   ROL A
      2A                   ROL A
      2A                   ROL A
      A0 04                LDY #04
0383  2A             CONT: ROL A         ; schuif naar IN-buffer
      26 F8                ROL INL
      26 F9                ROL INH
      88                   DEY
      D0 F8                BNE CONT
      A9 00                LDA #00       ; A=0 als korrekt
038D  60                   RTS
```

Subroutine OUTSP print een spatie.

```
038E  A9 E4          OUTSP:LDA #E4       ; spatie
      4C A0 1E             JMP OUTCH
```

Subroutine PRINT A print het adres POINT als 4 hex-cijfers, gevolgd door een spatie.

```
0393  A5 FB          PRINTA :LDA POINT H ; hoge byte
      20 3D 03               JSR PRTBYT
      A5 FA                  LDA POINT L ; lage byte
      20 3D 03               JSR PRTBYT
      20 8E 03               JSR OUTSP   ; spatie
      60                     RTS
```

PRINTD print de data in adreslokatie POINT als twee hex cijfers, gevolgd door een spatie.

```
03A1  A0 00          PRINTD:LDY #00
      B1 FA                 LDA(POINTL),Y; haal data op
      20 3D 03              JSR PRTBYT    ; print
      20 8E 03              JSR OUTSP     ; spatie
      60                    RTS
```

Tenslotte is subroutine CHECKM nodig, die test of een ontvangen telex-karakter een cijfer of lettercode is. Is dat zo, dan wordt de mode juist gezet en A wordt nul. Is het een ander karakter, dan doet de subroutine niets.

```
03AC  C9 7B          CHECKM:CMP #7B      ; cijfercode?
      D0 07                 BNE CHECK2
      A9 80                 LDA #80
      85 EE                 STA MODE2     ; zet cijfermode
      A9 00                 LDA #00       ; A:=0
      60                    RTS
03B7  C9 7F          CHECK2:CMP #7F       ; lettercode?
      D0 04                 BNE EXIT
      A9 00                 LDA #00       ; A:=0
      85 EE                 STA MODE2     ; zet lettermode
      60             EXIT  :RTS
```

Het Telex Monitor Programma

Het hoofdprogramma heeft de volgende funkties:

1) initialiseer de telex timing (zie vorige artikel hoe U aan de juiste getallen voor Uw telex komt);
2) initialiseer de stack pointer;
3) zorg ervoor, dat een BRK instruktie terugspringt naar de telex-monitor; daarbij is nodig dat de IRQ vector adres 0212 bevat in plaats van de gebruikelijke 1C00 (zet op 17FE 12 en op 17FF 02);
4) zorg ervoor, dat met een JSR vanuit het programma de telex-monitor bereikbaar is. Wilt U vanuit het programma terug naar de telex, neem dan als laatste instruktie van Uw programma:

```
20 17 02        JSR SAVEØ
```

5) aksepteer kommando's van de telex en voer die uit. Een lijst van mogelijke kommando's volgt na de programmatekst.

Het programma dat begint op geheugenplaats 0200, volgt hieronder.

```
0200  A9 82        INITX: LDA #82     ; telex timing
      8D F2 17            STA CNTL30
      A9 05               LDA #05
      8D F3 17            STA CNTH30
020A  A2 FF        STCKP: LDX #FF     ; initialiseer stack
      9A                  TXS
      86 F2               STX SPUSER
020F  4C 24 02            JMP START
0212  85 F3        SAVE:  STA ACC     ; BRK ENTRY
      68                  PLA
      85 F1               STA PREG   ; bewaar A
0217  68           SAVEØ: PLA         ; JSR ENTRY
      85 FA               STA POINTL ; bewaar adres
      68                  PLA
      85 FB               STA POINTH
      84 F4               STY YREG   ; bewaar Y
      86 F5               STX XREG   ; bewaar X
      BA                  TSX
      86 F2               STX SPUSER ; bewaar stack pointer
0224  20 88 1E    START: JSR INITS   ; initialiseer KIM
      A9 60               LDA #60    ; blank
      20 16 03            JSR OUTCHØ ; zet letter mode
```

```
022C  4C 32 02          JMP RTRN2
022F  20 63 1F    RTRN: JSR INCPT    ; POINT := POINT + 1
0232  20 4E 03    RTRN2:JSR CRLF     ; nieuwe regel
0235  20 93 03    ADDOUT:JSR PRINTA  ; print adres
0238  20 A1 03    DATOUT:JSR PRINTD  ; print data
023B  20 5E 03    BELLS :JSR BELL    ; bel
023E  A9 00       CLEAR:LDA #00      ; input buffer :=0
      85 F3             STA INL
      85 F9             STA INH
0244  20 63 03    READ: JSR GETCH    ; lees karakter in
      20 AC 03          JSR CHECKM   ; letter of cijferkode?
      F0 F8             BEQ READ
      20 6F 03          JSR PACK     ; hex cijfer?
      F0 F3             BEQ READ
0251  29 7F       SCAN: AND #7F      ; reset mode bit
0253  C9 64       L2:   CMP #64      ; spatie?
      D0 05             BNE L3
      20 CC 1F          JSR OPEN     ; POINT : = IN
      B0 DC             BCS DATOUT
025C  C9 68       L3:   CMP #68      ; CR?
      F0 CF             BEQ RTRN
      C9 62       L4:   CMP #62      ; LF?
      D0 10             BNE L5
      A5 FA             LDA POINTL
      D0 02             BNE X1
      C6 FB             DEC POINTH
      C6 FA       X1:   DEC POINTL
      A9 E8             LDA #E8      ; CR
      20 A0 1E          JSR OUTCH
      4C 35 02          JMP ADDOUT
0274  05 EE       L5:   ORA MODE2    ; zet juiste mode
      C9 66             CMP #66      ; I?
      D0 05             BNE L6
      85 F3             STA ACC
      20 05 1C          JSR SAVE1    ; naar KIM
```

```
027F  C9 FC        L6:  CMP #FC      ; punt?
      D0 09             BNE L7
      A0 00             LDY #00
      A5 F8             LDA INL     ; haal data op
      91 FA             STA(POINTL),Y;en zet ze weg
      4C 2F 02          JMP RTRN
028C  C9 7A        L7:  CMP #7A      ; G?
      D0 03             BNE L8
      4C C8 1D          JMP 1DC8    ; GOEXEC
0293  C9 EB        L8:  CMP#EB       ; bel?
      D0 06             BNE L9
      20 4E 03          JSR CRLF
      4C 3B 02          JMP BELLS   ; wacht op nieuw kommando
029D  C9 74        L9:  CMP #74      ; H?
      D0 1A             BNE L10
      A5 FA             LDA POINTL
      29 F0             AND #F0
      85 FA             STA POINTL
      20 4E 03          JSR CRLF    ; nieuwe regel
      20 93 03          JSR PRINTA  ; print adres
      A2 0F             LDX #0F     ; zestien maal
02AF  20 A1 03     X2:  JSR PRINTD  ; print data
      20 63 1F          JSR INCPT
      CA                DEX
      10 F7             BPL X2
      4C 3B 02          JMP BELLS
02BB  C9 6A        L10: CMP #73      ; W?
      D0 17             BNE L11
      20 4E 03          JSR CRLF    ; print
02C2  A0 00        X3:  LDY #00      ; karakter buffer
      B1 FA             LDA(POINTL),Y
      C9 EB             CMP #EB     ; bel?
      D0 03             BNE X4
      4C 2F 02          JMP RTRN2   ; ja!
02CD  20 16 03     X4:  JSR OUTCHØ  ; print karakter
      20 63 1F          JSR INCPT
      4C C2 02          JMP X3
```

```
02D6  C9 73        L11:  CMP #6A      ; R?
      D0 1E              BNE L12
      20 4E 03           JSR CRLF     ; nieuwe regel
02DB  20 63 03     X5:   JSR GETCH    ; lees karakter
      20 AC 03           JSR CHECKM   ; cijfer of letter?
      F0 F8              BEQ X5
      05 EE              ORA MODE2    ; mode bit
      A0 00              LDY #00
      91 FA              STA(POINTL),Y: berg op in karakterbuffer
      C9 EB              CMP #EB      ; was het de bel?
      D0 03              BNE X6
      4C 32 02           JMP RTRN2    ; ja!
02F2  20 63 1F     X6:   JSR INCPT
      4C DD 02           JMP X5
02F8  4C 00 02     L12:  JMP INITX    ; ongeldig!
```

Kommando's

spatie:	maakt adres geldig; print data op dat adres
punt:	zet data weg; print volgend adres en data
terug wagen:	print volgend adres en data
nieuwe regel:	print vorig adres en data
I:	springt terug naar KIM
G:	start programma vanaf ingetikt adres
bel:	nooduitgang; wacht op nieuw kommando (eerst adres intikken)
H:	hex dump; print 16 bytes
W:	print karakter buffer vanaf ingetikt adres totdat karakter = bel; komt terug met adres van bel-karakter
R:	leest karakters van telex en schrijft ze in het karakter buffer tot een bel-karakter wordt ingetikt.

Net als bij de KIM worden foute adressen en data hersteld door opnieuw intikken Van adressen worden de laatste 4, van data de laatste 2 hexadecimale cijfers genomen, b.v.

1253 1256 AC 4356. (respons van KIM is onderstreept)
adres data

Op geheugenplaats 1256, waar AC stond, wordt 56 geplaatst.

Het klinken van de bel geeft aan dat een nieuw kommando wordt verwacht.

Een paar voorbeelden

Start het programma met het toetsenbord van de KIM door 0200G0.
Hieronder een voorbeeld van de werking van sommige kommando's. De response van de KIM is onderstreept. Het begint zo:

0200 A9 H ↵

A9 82 8D F2 17 A9 05 8D F3 17 A2 FF 9A 86 F2 4C TW
0210 24 NR
020F 4C bel
0000 XX 12. ↵
0001 yy 34. ↵
0002 zz G (start het programma, dat hier begint)

TW = terug wagen
NR = nieuwe regel
↵ = TW + NR
bel = bel

Tenslotte

Het gegeven monitorprogramma voor de telex kan met meer gemak het toetsenbord van de KIM vervangen, op één uitzondering na: de reset. De RST-toets van de KIM blijft nodig om een "wild" programma te beëindigen.
Het programma is ook erg flexibel: U kunt naar behoefte kommando's weglaten of nieuwe toevoegen; de beschreven subroutines maken dat erg eenvoudig.
Heeft u b.v. geen behoefte aan de R en W kommando's dan laat U eenvoudig alles tussen adressen 02BB en 02F8 weg.
Veel sukses!

J.A. Blom
Tarantostraat 48
Eindhoven