```
181 A152 E1              POP   H
182 A153 CA63A1          JZ    DONE      indien 0 dan einde
183 A156 D1              POP   D
184 A157 1C              INR   E         adresteller+1
185 A158 7B              MOV   A,E       laatste adres van blok?
186 A159 FE00            CPI   0
187 A15B C23AA1          JNZ   NXTBY      volgende byte
188 A15E D1              POP   D
189 A15F 1C              INR   E         blokkenteller +1
190 A160 C333A1          JMP   NXTBLC    volgende blok
191 A163 115022    DONE  LXI   D,:2250   deselekteren EPROM
192 A166 CDC8D8          CALL  RICOUT
193 A169 21BFA2          LXI   H,MES11   pointer message 11
194 A16C CD2FED          CALL  :ED2F     printen message 11
195 A16F CD83A0          CALL  READ      verifieren EPROM in-
196 A172 2ACBA2          LHLD  LENGTE    houd byte per byte
197 A175 EB              XCHG
198 A176 2AC8A2          LHLD  SRCBUF
199 A179 0100A3          LXI   B,BUFFER
200 A17C 0A        NXTVER LDAX  B
201 A17D BE              CMP   M
202 A17E C298A1          JNZ   ERR
203 A181 23              INX   H
204 A182 03              INX   B
205 A183 1B              DCX   D
206 A184 7B              MOV   A,E
207 A185 B2              ORA   D
208 A186 C27CA1          JNZ   NXTVER
209 A189 115022          LXI   D,:2250
210 A18C CDC8D8          CALL  RICOUT
211 A18F 217BA2          LXI   H,MES6
212 A192 CDC3A1          CALL  PNT$
213 A195 C361A0          JMP   PROMPT
214 A198 E5        ERR   PUSH  H
215 A199 219BA2          LXI   H,MES9    pointer message 9
216 A19C CD3AED          CALL  :ED3A
217 A19F CD2FED          CALL  :ED2F     printen message 9
218 A1A2 E1              POP   H
219 A1A3 CD18ED          CALL  :ED18     printen adres
220 A1A6 C361A0          JMP   PROMPT
221                      *
222                      * bevelentabel
223                      *
224 A1A9 00        CMDTAB DATA  00
225 A1AA 50               DATA  'P'
226 A1AB EFA0             DBL   PGMCMD
227 A1AD 52               DATA  'R'
228 A1AE 7DA0             DBL   RDCMD
229 A1B0 54               DATA  'T'
230 A1B1 BAA0             DBL   TSTCMD
231 A1B3 55               DATA  'U'
232 A1B4 42EA             DBL   :EA42
233 A1B6 FF               DATA  :FF
234 A1B7 CD3AED    ERROR  CALL  :ED3A
235 A1BA 0E3F             MVI   C,'?'
236 A1BC 2A5B00           LHLD  :5B
237 A1BF F9               SPHL
238 A1C0 C361A0           JMP   PROMPT
239 A1C3 CD3AED    PNT$   CALL  :ED3A
240 A1C6 CD2FED           CALL  :ED2F
241 A1C9 CD3AED           CALL  :ED3A
242 A1CC C9               RET
243 A1CD 3ACDA2    IMPULS LDA   ROMTYP    programmeerimp  2716
```

```
244 A1D0 FE10            CPI   16
245 A1D2 C2E5A1          JNZ   IMP2
246 A1D5 110923          LXI   D,:2309
247 A1D8 CDC8D8          CALL  RICOUT
248 A1DB CDF5A1          CALL  DELAY
249 A1DE 110823          LXI   D,:2308
250 A1E1 CDC8D8          CALL  RICOUT
251 A1E4 C9              RET
252 A1E5 110823    IMP2  LXI   D,:2308   programmeerimp 2732
253 A1E8 CDC8D8          CALL  RICOUT
254 A1EB CDF5A1          CALL  DELAY
255 A1EE 110923          LXI   D,:2309
256 A1F1 CDC8D8          CALL  RICOUT
257 A1F4 C9              RET
258 A1F5 E5        DELAY PUSH  H         lengte impuls
259 A1F6 217008          LXI   H,:0870
260 A1F9 2B        LUS   DCX   H
261 A1FA 7D              MOV   A,L
262 A1FB B4              ORA   H
263 A1FC C2F9A1          JNZ   LUS
264 A1FF E1              POP   H
265 A200 C9              RET
266 A201 0C        MES1  DATA  :0C
267 A202 50524F          ASC   'PROM PROGRAMMER'
268 A211 00              DATA  00
269 A212 574849    MES2  ASC   'WHICH TYPE OF EPROM'
270 A225 0D              DATA  :0D
271 A226 282020          ASC   '(  2716=2  2732=4  )'
272 A23A 00              DATA  00
273 A23B 494E53    MES3  ASC   'INSERT EPROM AND TYPE SPACE'
274 A256 00              DATA  00
275 A257 455052    MES4  ASC   'EPROM NOT EMPTY'
276 A266 00              DATA  00
277 A267 4E4F20    MES5  ASC   'NO MATCH AT ADDRESS'
278 A27A 00              DATA  00
279 A27B 4E4F20    MES6  ASC   'NO ERRORS'
280 A284 00              DATA  00
281 A285 524541    MES7  ASC   'READY FOR PROGRAMMIMG'
282 A29A 00              DATA  00
283 A29B 202020    MES9  ASC   '   ERROR AT ADDRESS  '
284 A2B0 00              DATA  00
285 A2B1 50524F    MES10 ASC   'PROGRAMMING  '
286 A2BE 00              DATA  00
287 A2BF 444F4E45  MES11 ASC   'DONE'
288 A2C3 00              DATA  00
289 A2C4           BUFLEN RES  2
290 A2C6           LENGTH RES  2
291 A2C8           SRCBUF RES  2
292 A2CA           NMBLK  RES  1
293 A2CB           LENGTE RES  2
294 A2CD           ROMTYP RES  1
295 A2CE           CONST  RES  1
296                       ORG  :A300
297 A300           BUFFER RES  :1000,00
298 B300                  END
```

```
****************************
* S Y M B O L   T A B L E *
****************************

AIN    2390   AOUT   2380   BUFFER A300   BUFLEN A2C4
CMD    A06F   CMDTAB A1A9   CONST  A2CE   DELAY  A1F5
DONE   A163   EPR16  A123   EPR32  A128   ERR    A198
```

```
ERROR  A1B7   ERRTYP A03F   GETCMD A069   GO     A12A
IMP2   A1E5   IMPULS A1CD   INIT   A000   INSERT A053
LENGTE A2CB   LENGTH A2C6   LUS    A1F9   MES1   A201
MES10  A2B1   MES11  A2BF   MES2   A212   MES3   A23B
MES4   A257   MES5   A267   MES6   A27B   MES7   A285
MES9   A29B   MONIT  EA42   NEMP   A0E2   NEXT   A09C
NMBLK  A2CA   NXTBLC A133   NXTBLK A095   NXTBY  A13A
NXTTST A0CB   NXTVER A17C   PGMCMD A0EF   PNT$   A1C3
PROMPT A061   RDCMD  A07D   READ   A083   RICIN  D8E0
RICOUT D8C8   ROMTYP A2CD   SELCRD A047   SRCBUF A2C8
TEST   A0C0   TSTCMD A0BA   TYPE   A00C   TYPE16 A02E
TYPE32 A01D   WAITSP A059
```

EPROM programmeerapparaat voor 2716 en 2732

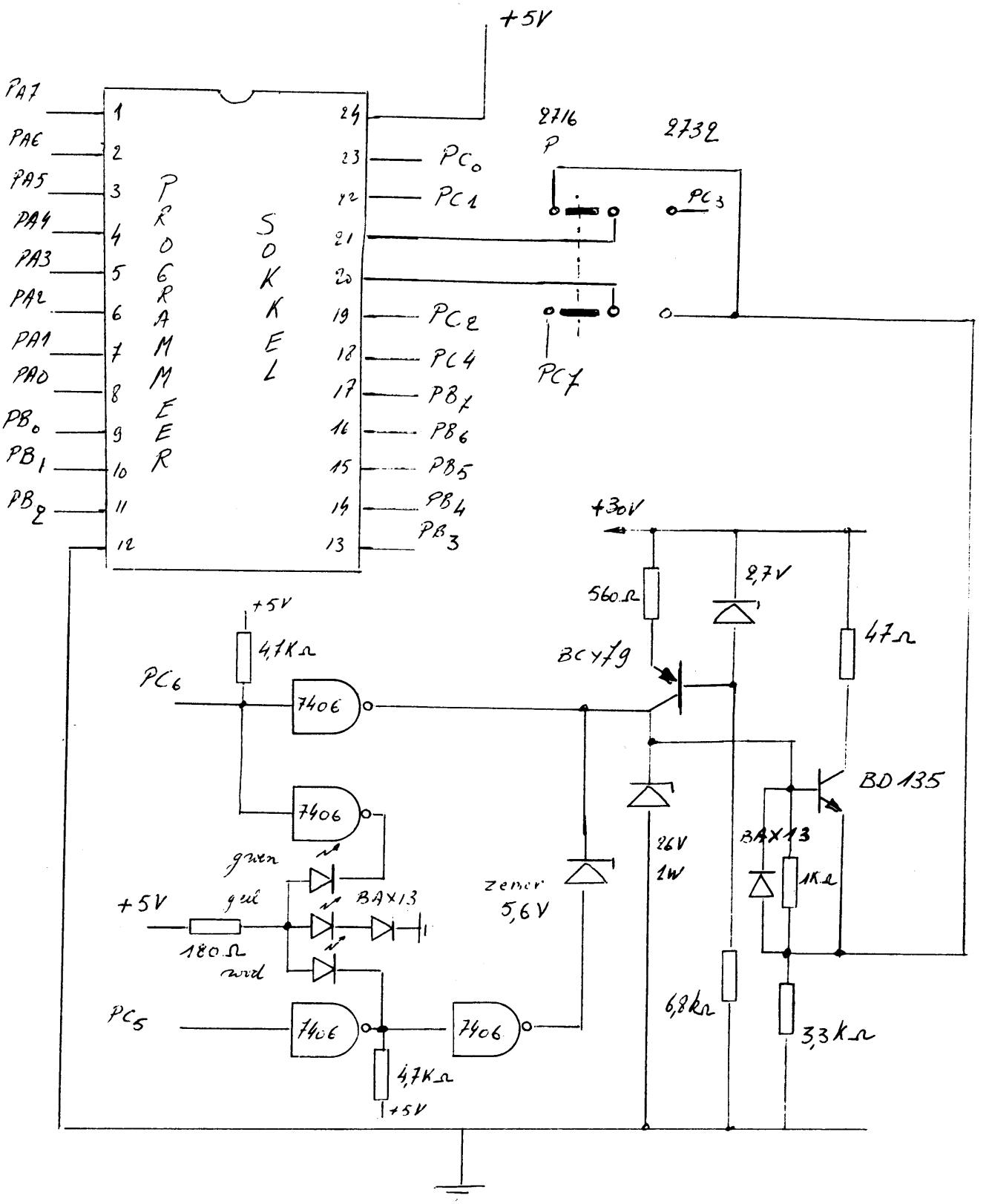
Het programmeerapparaat kan gemonteerd worden op het WIRE-WRAP veld van een universele PC DCE BUS INTERFACEKAART, en kan dus parallel met een snelle cassette en alle eventuele verdere DCE kaarten gebruikt worden. Er kunnen zowel 2K als 4K EPROMS gebruikt worden. Het programma herkent een aantal bevelen, die gegeven worden door het intikken van een karakter, al dan niet gevolgd door het aangeven van een of meerdere adressen.
Na het starten van het programma meldt het systeem zich met
PROM PROGRAMMER.

WHICH TYPE OF EPROM
(2716=2 2732=4).

Na het intikken van 2 of 4 alnaargelang men met een 2716 of 2732 wenst te werken, meldt het programma zich terug met
INSERT EPROM AND TYPE SPACE
Nu kan u zonder gevaar de EPROM in de programmeersokkel steken (groene LED brandt) en daarna een SPACE intikken.
Het systeem meldt zich met de prompt en wacht nu op het intikken van een bevel.
Het is normaal dat men eerst wenst te testen of de EPROM leeg is of behoorlijk gewist is. Dit kan gebeuren door het ingeven van het bevel (T).
Het programma leest (gele LED brandt) de ganse EPROM en verifieert of inderdaad in elke geheugenplaats FF staat. Is zulks het geval dan komt de melding
READY FOR PROGRAMMING
In het tegenovergestelde geval komt de melding
EPROM NOT EMPTY
Wenst men nu over te gaan tot de programmering van de EPROM dan kan dit gebeuren met het bevel (P). Dit bevel dient,om volledig te zijn,gevolgd te worden door drie hexadecimale ad-

cont. on p.411

SCHEMA EPROM PROGRAMMER



$PA_0 \div PA_7$, $PB_0 \div PB_7$, $PC_0 \div PC_7$ zijn de overeenkomstige klemmen van de 8255 op de DCE interfacekaart.

program identification

title : MAZE GAME

author : M. DIERCKX

purpose : try to move the object through the maze

comment :

```
1     MODE 0
2     REM M.DIERCKX
3     REM IMP INT
10    REM het maken van een doolhof
20    MODE 2:COLORG 0 8 8 0
80    FOR I=0 TO XMAX STEP 10:DRAW I,0 I,YMAX 21:NEXT
90    FOR I=0 TO YMAX STEP 10:DRAW 0,I XMAX,I 21:NEXT
130   EIND=RND(30)+60
135   X1=0:Y1=0
140   FOR I=1 TO EIND
150   IF INT(RND(2))=0 THEN FLAG=1:X=RND(XMAX):DRAW X1,Y1 X,Y1 0:X1=X
160   IF FLAG=0 THEN Y=RND(YMAX):DRAW X1,Y1 X1,Y 0:Y1=Y
165   FLAG=0
170   NEXT I
175   DRAW X1,Y1 X1,YMAX 0
176   DRAW X1,YMAX XMAX,YMAX 0
180   COLORG 0 8 12 14
190   X=0:Y=0
191   DOT X,Y 0
192   X2=RND(XMAX):Y2=RND(YMAX):IF SCRN(X2,Y2)=0 THEN DOT X2,Y2 14
195   IF X=XMAX AND Y=YMAX GOTO 2000
196   ON RND(4)+1 GOTO 220,240,260
200   XI=X+1:IF XI>XMAX GOTO 195
210   IF SCRN(XI,Y)=0 THEN DOT XI,Y 12:GOSUB 1000:DOT X,Y 0:X=XI:GOTO 200
215   GOTO 195
220   YI=Y+1:IF YI>YMAX GOTO 195
230   IF SCRN(X,YI)=0 THEN DOT X,YI 12:GOSUB 1000:DOT X,Y 0:Y=YI:GOTO 220
235   GOTO 195
240   XI=X-1:IF XI<0 GOTO 195
250   IF SCRN(XI,Y)=0 THEN DOT XI,Y 12:GOSUB 1000:DOT X,Y 0:X=XI:GOTO 240
255   GOTO 195
260   YI=Y-1:IF YI<0 GOTO 195
270   IF SCRN(X,YI)=0 THEN DOT X,YI 12:GOSUB 1000:DOT X,Y 0:Y=YI:GOTO 260
280   GOTO 195
1000  WAIT TIME 2:SEC=SEC+1:Q=GETC:IF Q=0 THEN RETURN
1010  IF Q=19 THEN X2I=X2+1:IF X2I<=XMAX THEN IF SCRN(X2I,Y2)=0 THEN DOT X2I,Y2
14:DOT X2,Y2 0:X2=X2I:RETURN
1020  IF Q=18 THEN X2I=X2-1:IF X2I>=0 THEN IF SCRN(X2I,Y2)=0 THEN DOT X2I,Y2 14:
DOT X2,Y2 0:X2=X2I:RETURN
1030  IF Q=16 THEN Y2I=Y2+1:IF Y2I<=YMAX THEN IF SCRN(X2,Y2I)=0 THEN DOT X2,Y2I
14:DOT X2,Y2 0:Y2=Y2I:RETURN
1040  IF Q=17 THEN Y2I=Y2-1:IF Y2I>=0 THEN IF SCRN(X2,Y2I)=0 THEN DOT X2,Y2I 14:
DOT X2,Y2 0:Y2=Y2I:RETURN
1050  RETURN
2000  MODE 0
2010  CURSOR 10,10:PRINT " u hebt er ";SEC/70;"  seconden over gedaan"
2020  Q=GETC:IF Q=0 GOTO 2020
2030  GOTO 10
```

VIDEO RAM TABLE


```
001                ******************************************
002                *                                        *
003                *        Video RAM table generator       *
004                *                                        *
005                *          by Salvatore PENNISI          *
006                * Via Mario Borsa 63 - 00159 ROMA (ITALY) *
007                ******************************************
008                *                                        *
009                *   This program generates a table with  *
010                * the addresses of the lines of the video *
011                * RAM in any MODE.                       *
012                *   It is particularly useful when the   *
013                * speed is very important   (games,      *
014                * graphics,...).                         *
015                *   In fact, having the y-coordinate,    *
016                * it is possible to know the address of  *
017                * line without time consuming            *
018                * calculations.                          *
019                *                                        *
020                ******************************************
021                *                                        *
022                *         How to use this program        *
023                *  1) Load this program                  *
024                *     UT                                 *
025                *     R                                  *
026                *     Z3                                 *
027                *     G300                               *
028                *  2) Write in HEX the start address of  *
029                *     video RAM table to merge after     *
030                *     in your ASSEMBLER program          *
031                *  3) Select the MODE                    *
032                *  OK. The table is created, save it     *
033                *      on tape by the W command          *
034                ******************************************
035                * N.B. For example the video RAM table   *
036                *      in MODE 5 (512 bytes) is :        *
037                *      Start address                     *
038                *      6649 66A3 66FD 6757 ... BF95 BFEF *
039                ******************************************
040                * INDIRIZZI ROM
041                *
042                RSTART  EQU   :C80C
043                USTART  EQU   :E009
044                GETFRC  EQU   :D6BE     INPUT DA TASTIERA
045                PMSG    EQU   :DAD4     PRINT DI UN MESSAGGIO
046                OUTC    EQU   :DD60     OUTPUT DI UN CARATTERE
047                ALFNUM  EQU   :DE09     CONTROLLO 0-9 A-Z
048                MODE0   EQU   :FF
049                *
050                * COSTANTI MODE 0
051                *
052                INCR0   EQU   :86
053                VB0     EQU   :B35F     :B3E5-:86
054                COUNT0  EQU   :18
055                *
056                * COSTANTI MODE 1/2
057                *
058                INCR1   EQU   :18
059                VB1     EQU   :B9D7     :B9EF-:18
060                COUNT1  EQU   :41
061                *
```

```
062                     * COSTANTI MODE 3/4

PAGE 02

063                     *
064                     INCR3   EQU   :2E
065                     VB3     EQU   :A893     :A8C1-:2E
066                     COUNT3  EQU   :82
067                     *
068                     * COSTANTI MODE 5/6
069                     *
070                     INCR5   EQU   :5A
071                     VB5     EQU   :65EF     :6649-:5A
072                     COUNT5  EQU   :FF       CONTATORE RIGHE
073                     *
074                     * COSTANTI ASCII
075                     *
076                     FF      EQU   :0C       FORMFEED
077                     CR      EQU   :0D       CARRIAGE RETURN
078                     DELC    EQU   :08       DEL CHAR
079                     CAR0    EQU   :30       ZERO
080                     CAR1    EQU   :31       UNO
081                     CAR3    EQU   :33       TRE
082                     CAR5    EQU   :35       CINQUE
083                     *
084                     CARG    EQU   :47       G
085                     *
086                             ORG   :300
087                     *
088                     * IMPOSTA MODE 0
089                     *
090 0300 3EFF           START   MVI   A,MODE0
091 0302 EF                     RST   5
092 0303 18                     DATA  :18
093                     *
094                     * PRINT CHR$(12)
095                     *
096 0304 217C04                 LXI   H,CHR12$
097 0307 CDD4DA                 CALL  PMSG
098                     *
099                     * PRINT MESS0
100                     *
101 030A 217E04                 LXI   H,MESS0
102 030D CDD4DA                 CALL  PMSG
103                     *
104                     * PRINT MESS1
105                     *
106 0310 21A104         ME1     LXI   H,MESS1
107 0313 CDD4DA                 CALL  PMSG
108 0316 CDBED6         GETC    CALL  GETFRC
109 0319 CA1603                 JZ    GETC
110 031C FE08                   CPI   DELC
111 031E CA1003                 JZ    ME1
112 0321 CD09DE                 CALL  ALFNUM
113 0324 D21003                 JNC   ME1
114 0327 FE47                   CPI   CARG
115 0329 DA2F03                 JC    OK
116 032C C31003                 JMP   ME1
117 032F CD60DD         OK      CALL  OUTC
118 0332 323805                 STA   IND
119 0335 CDBED6         GETC1   CALL  GETFRC
120 0338 CA3503                 JZ    GETC1
121 033B FE08                   CPI   DELC
122 033D CA1003                 JZ    ME1
123 0340 CD09DE                 CALL  ALFNUM
```

```
124 0343 D21003              JNC    ME1


PAGE 03

125 0346 FE47                CPI    CARG
126 0348 DA4E03              JC     OK1
127 034B C31003              JMP    ME1
128 034E CD60DD      OK1     CALL   OUTC
129 0351 323905              STA    IND+1
130 0354 CDBED6      GETC2   CALL   GETFRC
131 0357 CA5403              JZ     GETC2
132 035A FE08                CPI    DELC
133 035C CA1003              JZ     ME1
134 035F CD09DE              CALL   ALFNUM
135 0362 D21003              JNC    ME1
136 0365 FE47                CPI    CARG
137 0367 DA6D03              JC     OK2
138 036A C31003              JMP    ME1
139 036D CD60DD      OK2     CALL   OUTC
140 0370 323A05              STA    IND+2
141 0373 CDBED6      GETC3   CALL   GETFRC
142 0376 CA7303              JZ     GETC3
143 0379 FE08                CPI    DELC
144 037B CA1003              JZ     ME1
145 037E CD09DE              CALL   ALFNUM
146 0381 D21003              JNC    ME1
147 0384 FE47                CPI    CARG
148 0386 DA8C03              JC     OK3
149 0389 C31003              JMP    ME1
150 038C CD60DD      OK3     CALL   OUTC
151 038F 323B05              STA    IND+3
152                  *
153                  * TEST CARRIAGE RETURN
154                  *
155 0392 CDBED6      CRETUR  CALL   GETFRC
156 0395 CA9203              JZ     CRETUR
157 0398 FE08                CPI    DELC
158 039A CA1003              JZ     ME1
159 039D FE0D                CPI    CR
160 039F CAA503              JZ     CONV
161 03A2 C39203              JMP    CRETUR
162                  *
163                  * CONVERSIONE IND DA  ASCII IN HEX
164                  *
165                  *  PRIMO E SECONDO BYE
166                  *
167 03A5 213805      CONV    LXI    H,IND
168 03A8 7E                  MOV    A,M
169 03A9 FE40                CPI    :40
170 03AB DAB003              JC     ASCZ
171 03AE D607                SUI    :07
172 03B0 D630        ASCZ    SUI    CAR0
173 03B2 07                  RLC
174 03B3 07                  RLC
175 03B4 07                  RLC
176 03B5 07                  RLC
177 03B6 47                  MOV    B,A
178 03B7 23                  INX    H
179 03B8 7E                  MOV    A,M
180 03B9 FE40                CPI    :40
181 03BB DAC003              JC     ASCZ1
182 03BE D607                SUI    :07
183 03C0 D630        ASCZ1   SUI    CAR0
184 03C2 80                  ADD    B
185 03C3 323D05              STA    INDH+1
```

```
186                     *


PAGE 04

187                     * TERZO E QUARTO BYTE
188                     *
189 03C6 23                     INX   H
190 03C7 7E                     MOV   A,M
191 03C8 FE40                   CPI   :40
192 03CA DACF03                 JC    ASCZ2
193 03CD D607                   SUI   :07
194 03CF D630           ASCZ2   SUI   CAR0
195 03D1 07                     RLC
196 03D2 07                     RLC
197 03D3 07                     RLC
198 03D4 07                     RLC
199 03D5 47                     MOV   B,A
200 03D6 23                     INX   H
201 03D7 7E                     MOV   A,M
202 03D8 FE40                   CPI   :40
203 03DA DADF03                 JC    ASCZ3
204 03DD D607                   SUI   :07
205 03DF D630           ASCZ3   SUI   CAR0
206 03E1 80                     ADD   B
207 03E2 323C05                 STA   INDH
208                     *
209                     * CONTROLLO INDH MAGGIORE #055F
210                     *
211 03E5 3A3D05                 LDA   INDH+1
212 03E8 FE05                   CPI   :05
213 03EA DA1003                 JC    ME1
214 03ED C2F803                 JNZ   ME2
215 03F0 3A3C05                 LDA   INDH
216 03F3 FE5F                   CPI   :5F
217 03F5 DA1003                 JC    ME1
218                     *
219                     * SCELTA MODE
220                     *
221 03F8 21E304         ME2     LXI   H,MESS2
222 03FB CDD4DA                 CALL  PMSG
223 03FE CDBED6         GETC4   CALL  GETFRC
224 0401 CAFE03                 JZ    GETC4
225 0404 FE08                   CPI   DELC
226 0406 CAF803                 JZ    ME2
227 0409 FE30                   CPI   CAR0
228 040B CA2004                 JZ    M0
229 040E FE31                   CPI   CAR1
230 0410 CA2D04                 JZ    M1
231 0413 FE33                   CPI   CAR3
232 0415 CA3A04                 JZ    M3
233 0418 FE35                   CPI   CAR5
234 041A CA4704                 JZ    M5
235 041D C3FE03                 JMP   GETC4
236                     *
237                     * INIZIALIZZA MODE 0
238                     *
239 0420 CD60DD         M0      CALL  OUTC
240 0423 1618                   MVI   D,COUNT0
241 0425 1E86                   MVI   E,INCR0
242 0427 015FB3                 LXI   B,VB0
243 042A C35404                 JMP   INIZIO
244                     *
245                     * INIZIALIZZA MODE 1/2
246                     *
247 042D CD60DD         M1      CALL  OUTC
```

```
248 0430 1641               MVI   D,COUNT1


PAGE 05

249 0432 1E18               MVI   E,INCR1
250 0434 01D7B9             LXI   B,VB1
251 0437 C35404             JMP   INIZIO
252                     *
253                     * INIZIALIZZA MODE 3/4
254                     *
255                     *
256 043A CD60DD     M3      CALL  OUTC
257 043D 1682               MVI   D,COUNT3
258 043F 1E2E               MVI   E,INCR3
259 0441 0193A8             LXI   B,VB3
260 0444 C35404             JMP   INIZIO
261                     * INIZIALIZZA MODE 5/6
262                     *
263 0447 CD60DD     M5      CALL  OUTC
264 044A 16FF               MVI   D,COUNT5
265 044C 1E5A               MVI   E,INCR5
266 044E 01EF65             LXI   B,VB5
267 0451 C35404             JMP   INIZIO
268                     *
269                     * CREAZIONE TABELLA IND. VIDEO
270                     *
271 0454 2A3C05     INIZIO  LHLD  INDH
272                     *
273 0457 79         LOOP    MOV   A,C
274 0458 83                 ADD   E
275 0459 4F                 MOV   C,A
276 045A 78                 MOV   A,B
277 045B CE00               ACI   :00
278 045D 47                 MOV   B,A
279 045E 70                 MOV   M,B
280 045F 23                 INX   H
281 0460 71                 MOV   M,C
282 0461 23                 INX   H
283 0462 15                 DCR   D
284 0463 C25704             JNZ   LOOP
285                     *
286                     * CALCOLO FUORI LOOP
287                     *
288 0466 79                 MOV   A,C
289 0467 83                 ADD   E
290 0468 4F                 MOV   C,A
291 0469 78                 MOV   A,B
292 046A CE00               ACI   :00
293 046C 47                 MOV   B,A
294 046D 70                 MOV   M,B
295 046E 23                 INX   H
296 046F 71                 MOV   M,C
297                     *
298                     * PRINT FINE
299                     *
300 0470 212B05     FINE    LXI   H,ENDMES
301 0473 CDD4DA             CALL  PMSG
302 0476 3100F9             LXI   SP,:F900
303 0479 C309E0             JMP   USTART
304                     *
305                     * MESSAGGIO CHR$(12)
306                     *
307 047C 0C         CHR12$  DATA  FF
308 047D 00                 DATA  0
309                     *
```

```
310                     * MESSAGGI

PAGE 06

311                     *
312 047E 0D0D           MESS0   DATA  CR,CR
313 0480 202020                 ASC   '      Video RAM table generator'
314 049F 0D                     DATA  CR
315 04A0 00                     DATA  0
316 04A1 0D0D           MESS1   DATA  CR,CR
317 04A3 537461                 ASC   'Start addrress of table'
318 04BA 20696E                 ASC   ' in HEX'
319 04C1 0D                     DATA  CR
320 04C2 286164                 ASC   '(address greater then #055F)   #'
321 04E2 00                     DATA  0
322 04E3 0D0D           MESS2   DATA  CR,CR
323 04E5 30203D                 ASC   '0 = MODE 0'
324 04EF 0D                     DATA  CR
325 04F0 31203D                 ASC   '1 = MODE 1/2'
326 04FC 0D                     DATA  CR
327 04FD 33203D                 ASC   '3 = MODE 3/4'
328 0509 0D                     DATA  CR
329 050A 35203D                 ASC   '5 = MODE 5/6'
330 0516 0D0D                   DATA  CR,CR
331 0518 53656C                 ASC   'Select the MODE
332 052A 00                     DATA  0
333 052B 0D0D           ENDMES  DATA  CR,CR
334 052D 2A2A20                 ASC   '** DONE **'
335 0537 00                     DATA  0
336                     *
337                     * CAMPI DI LAVORO
338                     *
339 0538                IND     RES   4,0
340 053C                INDH    RES   2,0
341 053E                        END

****************************
* S Y M B O L   T A B L E *
****************************

ALFNUM DE09   ASCZ   03B0   ASCZ1  03C0   ASCZ2  03CF
ASCZ3  03DF   CAR0   0030   CAR1   0031   CAR3   0033
CAR5   0035   CARG   0047   CHR12$ 047C   CONV   03A5
COUNT0 0018   COUNT1 0041   COUNT3 0082   COUNT5 00FF
CR     000D   CRETUR 0392   DELC   0008   ENDMES 052B
FF     000C   FINE   0470   GETC   0316   GETC1  0335
GETC2  0354   GETC3  0373   GETC4  03FE   GETFRC D6BE
INCR0  0086   INCR1  0018   INCR3  002E   INCR5  005A
IND    0538   INDH   053C   INIZIO 0454   LOOP   0457
M0     0420   M1     042D   M3     043A   M5     0447
ME1    0310   ME2    03F8   MESS0  047E   MESS1  04A1
MESS2  04E3   MODE0  00FF   OK     032F   OK1    034E
OK2    036D   OK3    038C   OUTC   DD60   PMSG   DAD4
RSTART C80C   START  0300   USTART E009   VB0    B35F
VB1    B9D7   VB3    A893   VB5    65EF
```

NEW CHARS ON GP100

```
PAGE 01    MODERNISED CHARS FOR SEIKOSHA GP100

002                     *
003                             ORG     :2EC
004                     PPICMD  EQU     :FE03
005                     PPIP~A  EQU     :FE00
006                     PPIP~B  EQU     :FE01
007                     PPIP~C  EQU     :FE02
008 02EC 800000         CHRP$   DATA    :80,0,0,0,0,0 < CHAR. DATA WORD
009 02F2 0D             LSTCHR  DATA    :D
010 02F3 00             ACTCHR  DATA    :0
011 02F4 0F             PRMODE  DATA    :F              < ACTUAL PRINTER MODE
012 02F5 100000         PRPOS$  DATA    :10,0,0         < LEFT SPACE POSITION
013                     *
014                             ORG     :2FF
015 02FF 0E             LEFTSP  DATA    14              NUMBER OF SPACES LEFT-SIDE
016                     *
017 0300 E5             INIT    PUSH    H               INIT PARALLEL PRINT-SERV
018 0301 F5                     PUSH    PSW
019 0302 3EA0                   MVI     A,:A0
020 0304 3203FE                 STA     PPICMD          DCE 8255 COMMAND MODE 1
021 0307 3EFF                   MVI     A,:FF           .  Port A : DATA OUT
022 0309 3201FE                 STA     PPIP~B          .  Port B : DEV ADDR OUT
023 030C 21DD02                 LXI     H,:02DD         .  Port C : CONTROL I/O
024 030F 36C3                   MVI     M,:C3
025 0311 23                     INX     H
026 0312 3640                   MVI     M,:40
027 0314 23                     INX     H               ADJUST DOUTC
028 0315 3603                   MVI     M,:03           SYSTEM JUMP #340
029                     *
030 0317 3AFF02                 LDA     LEFTSP          INIT INSERT LEFT-SPACE
031 031A 3D                     DCR     A               NUMBER OF SPACES LEFT
032 031B 27                     DAA                     DECIMAL ADJUST
033 031C 6F                     MOV     L,A
034 031D E6F0                   ANI     :F0
035 031F 0F                     RRC
036 0320 0F                     RRC
037 0321 0F                     RRC
038 0322 0F                     RRC
039 0323 C630                   ADI     :30
040 0325 32F602                 STA     PRPOS$+1        TENTHS IN ASCII
041 0328 7D                     MOV     A,L
042 0329 E60F                   ANI     :0F
043 032B C630                   ADI     :30
044 032D 32F702                 STA     PRPOS$+2        SINGLES IN ASCII
045 0330 F1                     POP     PSW
046 0331 E1                     POP     H
047 0332 C9                     RET                     END OF INIT-ROUTINE
048                     *
049                     *
050                                                     ENTRY : CHR IN ACCU
051                             ORG     :340
052 0340 E5             START   PUSH    H               START OF DOUTC-PROCESSING
053 0341 D5                     PUSH    D
054 0342 C5                     PUSH    B
```

```
055 0343 F5                PUSH  PSW
056 0344 32F302            STA   ACTCHR
057 0347 3AF202            LDA   LSTCHR
058 034A FE0D              CPI   :D
059 034C CAC803            JZ    CARRET
060 034F 3AF302      NOCRRT LDA   ACTCHR
061 0352 FE61              CPI   'a'          TEST CHR IN RANGE
062 0354 DA9D03            JC    UPCASE
063 0357 FE7B              CPI   :7B          'z'+1
064 0359 F29D03            JP    UPCASE
065 035C 3AF402            LDA   PRMODE
066 035F FE08              CPI   :8
067 0361 C4FA03            CNZ   PRMD=F
068                  *
069 0364 3AF302            LDA   ACTCHR
070 0367 210004            LXI   H,TABLE
071 036A D661              SUI   'a'          COMPUTE INDEX
072 036C 5F                MOV   E,A          IN NEW CHR TABLE
073 036D 010500            LXI   B,:0005
074 0370 CA7803      NEXTCH JZ    INXSET
075 0373 09                DAD   B
076 0374 1D                DCR   E
077 0375 C37003            JMP   NEXTCH       EXIT : HL=TABLE-CHAR-POINTER
078 0378 11ED02      INXSET LXI   D,CHRP$+1
079                  *
080 037B 7E          NEXTBT MOV   A,M
081 037C C680              ADI   :80
082 037E EB                XCHG               COPY TABLE-CHR$ TO CHRP$
083 037F 77                MOV   M,A
084 0380 EB                XCHG
085 0381 23                INX   H
086 0382 13                INX   D
087 0383 0D                DCR   C
088 0384 C27B03            JNZ   NEXTBT
089                  *
090 0387 21EC02            LXI   H,CHRP$
091 038A 0E06              MVI   C,6
092 038C CDBE03            CALL  PRPAP$
093                  *
094 038F 3AF302      RETURN LDA   ACTCHR
095 0392 CDAE03            CALL  PRSCRN
096 0395 32F202            STA   LSTCHR
097 0398 F1                POP   PSW
098 0399 C1                POP   B
099 039A D1                POP   D
100 039B E1                POP   H
101 039C C9                RET                END OF DOUTC-PROCESSING
102                  *
103                  *
104 039D 3AF402      UPCASE LDA   PRMODE
105 03A0 FE0F              CPI   :F
106 03A2 C4EC03            CNZ   PRMD=8
107 03A5 3AF302            LDA   ACTCHR
```

```
108 03A8 CDB103     PRTUPC  CALL   PRPAPC
109 03AB C38F03             JMP    RETURN
110                 *
111 03AE EF         PRSCRN  RST    5          ENTRY : Accu = act char
112 03AF 03                 DATA   3
113 03B0 C9                 RET
114                 *
115 03B1 F5         PRPAPC  PUSH   PSW        ENTRY : Accu
116 03B2 3A02FE     WTACKP  LDA    PPIP~C
117 03B5 A7                 ANA    A
118 03B6 F2B203             JP     WTACKP
119 03B9 F1                 POP    PSW
120 03BA 3200FE             STA    PPIP~A
121 03BD C9                 RET               EXIT : Accu
122                 *
123 03BE 7E         PRPAP$  MOV    A,M        ENTRY : HL=strpntr,C=nrbytes
124 03BF CDB103             CALL   PRPAPC
125 03C2 0D                 DCR    C
126 03C3 C8                 RZ
127 03C4 23                 INX    H
128 03C5 C3BE03             JMP    PRPAP$
129                 *
130 03C8 3AF302     CARRET  LDA    ACTCHR
131 03CB FE0D               CPI    :D
132 03CD CAA803             JZ     PRTUPC
133 03D0 3AFF02             LDA    LEFTSP
134 03D3 A7                 ANA    A
135 03D4 CA4F03             JZ     NOCRRT     JUMP IF NO LEFTSPACE
136 03D7 FE01               CPI    :1
137 03D9 CAE403             JZ     ONESPC
138 03DC 21F502             LXI    H,PRPOS$
139 03DF 0E03               MVI    C,3
140 03E1 CDBE03             CALL   PRPAP$
141 03E4 3E20       ONESPC  MVI    A,:20
142 03E6 CDB103             CALL   PRPAPC
143 03E9 C34F03             JMP    NOCRRT
144                 *
145 03EC 3E80       PRMD=8  MVI    A,:80      HAMMER-SPC AT LOW-UP TRANS.
146 03EE CDB103             CALL   PRPAPC
147 03F1 3E0F               MVI    A,:F
148 03F3 32F402     SVPRMD  STA    PRMODE
149 03F6 CDB103             CALL   PRPAPC
150 03F9 C9                 RET
151                 *
152 03FA 3E08       PRMD=F  MVI    A,:8
153 03FC C3F303             JMP    SVPRMD
154                 *
155                                            TABLE WITH NEW CHARACTER-SET
156                                            normal width, normal hight
157                                            no descenders
158                 *
159                         ORG    :400
160 0400 205454     TABLE   DATA   :20,:54,:54,:54,:78 >  a
```

```
161 0405 7F4444            DATA  :7F,:44,:44,:44,:38 >  b
162 040A 384444            DATA  :38,:44,:44,:44,:44 >  c
163 040F 384444            DATA  :38,:44,:44,:44,:7F >  d
164 0414 385454            DATA  :38,:54,:54,:54,:58 >  e
165 0419 00447E            DATA  :00,:44,:7E,:45,:01 >  f
166 041E 085454            DATA  :08,:54,:54,:54,:3C >  g
167 0423 7F0404            DATA  :7F,:04,:04,:04,:78 >  h
168 0428 00447D            DATA  :00,:44,:7D,:40,:00 >  i
169 042D 404044            DATA  :40,:40,:44,:3D,:00 >  j
170 0432 7F1028            DATA  :7F,:10,:28,:44,:44 >  k
171 0437 00417F            DATA  :00,:41,:7F,:40,:00 >  l
172 043C 7C047C            DATA  :7C,:04,:7C,:04,:78 >  m
173 0441 7C0404            DATA  :7C,:04,:04,:04,:78 >  n
174 0446 384444            DATA  :38,:44,:44,:44,:38 >  o
175 044B 7C1414            DATA  :7C,:14,:14,:14,:08 >  p
176 0450 081414            DATA  :08,:14,:14,:14,:7C >  q
177 0455 447C48            DATA  :44,:7C,:48,:04,:04 >  r
178 045A 485454            DATA  :48,:54,:54,:54,:24 >  s
179 045F 043F44            DATA  :04,:3F,:44,:40,:00 >  t
180 0464 3C4040            DATA  :3C,:40,:40,:40,:7C >  u
181 0469 1C2040            DATA  :1C,:20,:40,:20,:1C >  v
182 046E 3C4078            DATA  :3C,:40,:78,:40,:3C >  w
183 0473 442810            DATA  :44,:28,:10,:28,:44 >  x
184 0478 0C5050            DATA  :0C,:50,:50,:50,:7C >  y
185 047D 446454            DATA  :44,:64,:54,:4C,:44 >  z
186 0482 7F7F7F            DATA  :7F,:7F,:7F,:7F,:7F > res
187 0487                   END
```

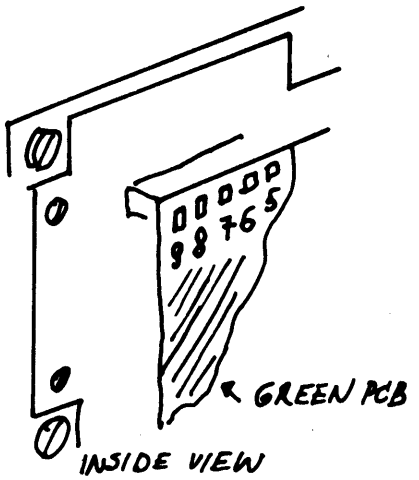
```
**************************
* S Y M B O L   T A B L E *
**************************
```

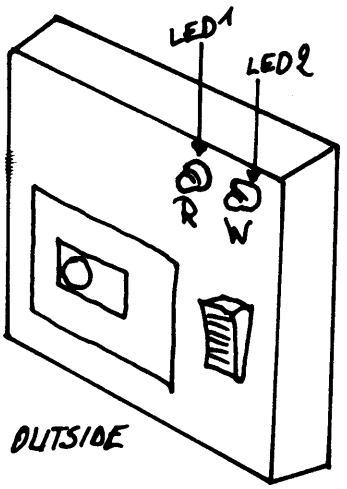
ACTCHR	02F3	CARRET	03C8	CHRP$	02EC	INIT	0300
INXSET	0378	LEFTSP	02FF	LSTCHR	02F2	NEXTBT	037B
NEXTCH	0370	NOCRRT	034F	ONESPC	03E4	PPICMD	FE03
PPIP~A	FE00	PPIP~B	FE01	PPIP~C	FE02	PRMD=8	03EC
PRMD=F	03FA	PRMODE	02F4	PRPAP$	03BE	PRPAPC	03B1
PRPOS$	02F5	PRSCRN	03AE	PRTUPC	03A8	RETURN	038F
START	0340	SVPRMD	03F3	TABLE	0400	UPCASE	039D
WTACKP	03B2						

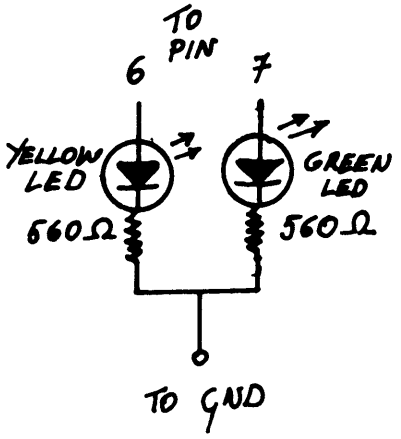
```
    ]#P.
02EC 00 00 00 00
02F0 00 00 0D 00 0F 10 00 00

02FF 0E
0300 E5 F5 3E A0 32 03 FE 3E FF 32 01 FE 21 DD 02 36
0310 C3 23 36 40 23 36 03 3A FF 02 3D 27 6F E6 F0 0F
0320 0F 0F 0F C6 30 32 F6 02 7D E6 0F C6 30 32 F7 02
0330 F1 E1 C9

0340 E5 D5 C5 F5 32 F3 02 3A F2 02 FE 0D CA C8 03 3A
0350 F3 02 FE 61 DA 9D 03 FE 7B F2 9D 03 3A F4 02 FE
0360 08 C4 FA 03 3A F3 02 21 00 04 D6 61 5F 01 05 00
0370 CA 78 03 09 1D C3 70 03 11 ED 02 7E C6 80 EB 77
0380 EB 23 13 0D C2 7B 03 21 EC 02 0E 06 CD BE 03 3A
0390 F3 02 CD AE 03 32 F2 02 F1 C1 D1 E1 C9 3A F4 02
03A0 FE 0F C4 EC 03 3A F3 02 CD B1 03 C3 8F 03 EF 03
03B0 C9 F5 3A 02 FE A7 F2 B2 03 F1 32 00 FE C9 7E

03BF CD
03C0 B1 03 0D C8 23 C3 BE 03 3A F3 02 FE 0D CA A8 03
03D0 3A FF 02 A7 CA 4F 03 FE 01 CA E4 03 21 F5 02 0E
03E0 03 CD BE 03 3E 20 CD B1 03 C3 4F 03 3E 80 CD B1
03F0 03 3E 0F 32 F4 02 CD B1 03 C9 3E 08 C3 F3 03

0400 20 54 54 54 78 7F 44 44 44 38 38 44 44 44 44 38
0410 44 44 44 7F 38 54 54 54 58 00 44 7E 45 01 08 54
0420 54 54 3C 7F 04 04 04 78 00 44 7D 40 00 40 40 44
0430 3D 00 7F 10 28 44 44 00 41 7F 40 00 7C 04 7C 04
0440 78 7C 04 04 04 78 38 44 44 44 38 7C 14 14 14 08
0450 08 14 14 14 7C 44 7C 48 04 04 48 54 54 54 24 04
0460 3F 44 40 00 3C 40 40 40 7C 1C 20 40 20 1C 3C 40
0470 78 40 3C 44 28 10 28 44 0C 50 50 50 7C

047D 44 64 54
0480 4C 44 7F 7F 7F 7F 7F
```

DCR TAPE DIRECTION

Using the DCR after a CHECK command, it is sometimes difficult to guess if the tape is still running forward or if it is already being rewound... but there is a very simple way to abtain a tape direction monitor on the DCR, using only two small resistors, two LED's and a small knife...







Referring to the drawing, pins 6 & 7 of a connector inside the DCR bear +5V, when the engine is winding or rewinding the tape. An easy to solder ground connection can be found on the interface board. (the one with socketed IC's): the ground track is the largest one, near a bolt. Moreover, it is very easy to take out the black DCR-front panel, unscrewing four little screws (the smallest ones) from inside the DCR: so here the sharp knife becomes use-ful to make two little holes in the plastic panel, the tiny and cheap pressure LED mountings perfectly match the colour of the front panel... The two 560 Ω resistors tie the LED cathodes to ground; so three thin isolated wires are sufficient for the connections. Be careful not to interfere with the microcassette-detection microswitch inside the front panel.

Paolo Siccardo - Savona - Italy

B. 8-BIT SYSTEMEN

HOOFDSTUK III : ARCHITECTUUR VAN 8-BITS MICROPROCESSOREN

We geven een korte samenvatting van de meest voorkomende microprocessoren en noemen dit de 'identiteitsfiche'.
Van de reeds eerder vernoemde industriële types geven we daarenboven nog een gedetailleerde beschrijving.

3.1. Identiteitsfiche van de microprocessor 8080 van INTEL

- TECHNOLOGIE : NMOS dynamisch

- KLOKFREQUENTIE : maximum 2MHz (8080) ; 3,13 MHz (A1 versie) ; 2,63 MHz (A2 versie).
  minimum 500 kHz

- KENMERKEN : voedingsspanningen : $\pm$ 5V en +12V
  verbruik : 1W
  klok : tweefasig verkregen met afzonderlijke schakeling (8224)
  databus : 8 bits
  adresbus : 16 bits
  controlesignalen : 5 bits afgeleid van de statussignalen die tijdens de eerste klokperiode beschikbaar zijn op de databus. Dit kan gebeuren door de 8228 (systeem en bus controller).
  interruptmogelijkheden : 1 niveau met 8 vaste sprongadressen.
  stapel (*stack*) : in het werkgeheugen (RAM) op een willekeurige plaats.
  instructieset : 78 basisinstructies
  logische niveaus : TTL compatibel
  uitvoeringstijden : van 2 tot 9µs (8080), van 1,3 tot 5,8µs (8080-A1) en van 1,5 tot 6,8µs (8080-A2)
  adresseringsmogelijkheden : impliciet, direct, indirect, immediate.

- INWENDIGE REGISTERS :

Accumulator	Flags
B	C
D	E
H	L
stapelwijzer (SP)	
programmateller (PC)	

- VEREENVOUDIGD TIJDSDIAGRAMMA (fig. 3.1)

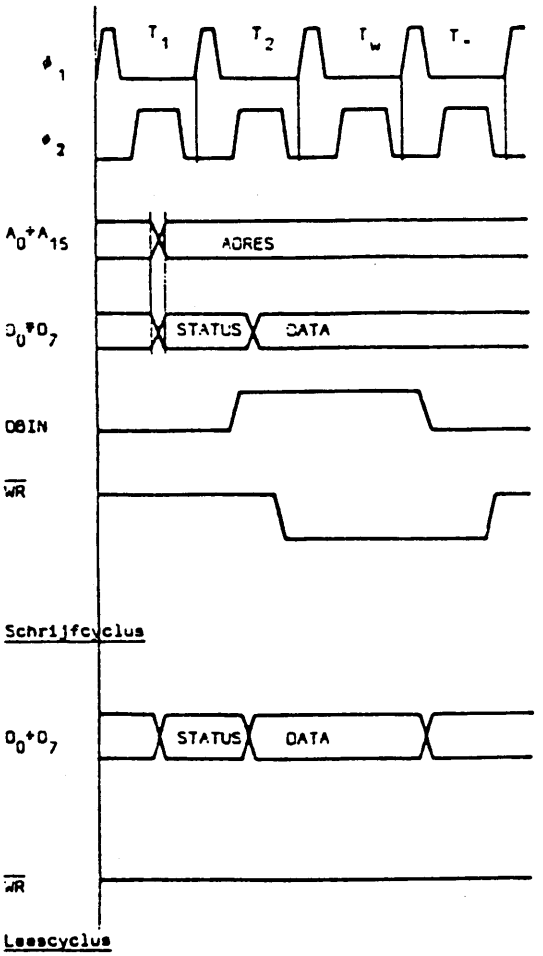


fig. 3.1

3.2. Architectuur van de microprocessor 8080 van INTEL

De microprocessor 8080 van INTEL is een dynamische 8-bits processor uitgevoerd in een N-kanaal MOS met één bus architectuur. Het inwendige schema wordt gegeven in fig. 3.2 en bestaat uit 5 delen :

- de 3-state buffers voor de uitwendige data- en adresbus
- de logische en rekenkundige eenheid met hulpregisters (ALU)
- het instructieregister (IR) met de instructiedecoder
- het registerveld met de adresseringslogica
- het controlegedeelte

3.2.1. De 3-state buffers voor data en adressen

Door deze buffers in de hoge impedantietoestand te plaatsen, kan de microprocessor zich isoleren zowel van de uitwendige gegevensbus als van de adresbus.

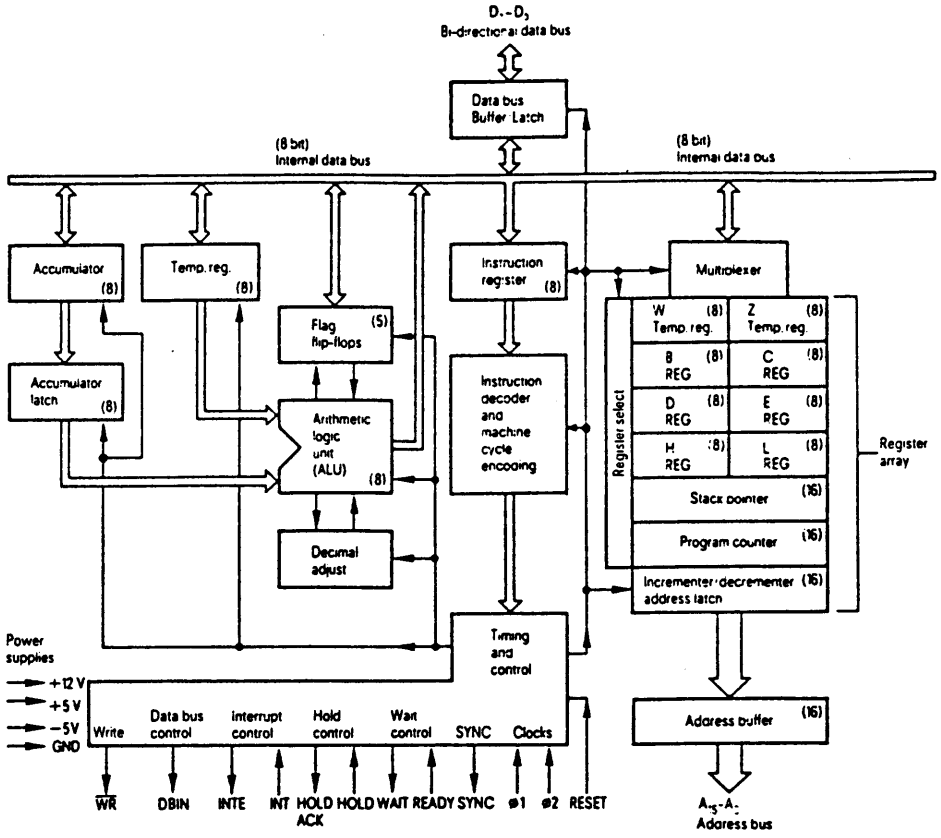


fig. 3.2

De bewerkingen die dan gebeuren in het inwendige van de microprocessor zijn onzichtbaar voor de omgeving. Het transfert van informatie in de microprocessor gebeurt via de interne databus waarop alle inwendige delen aangesloten zijn.

3.2.2. De logische en rekenkundige eenheid (ALU)

Dit gedeelte bevat :

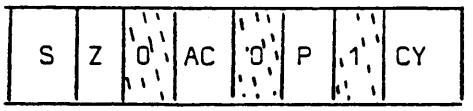
- een 8-bits accumulator (A)
- een 8-bits accumulator buffer/geheugen (*latch*)
- een 8-bits voorlopig register (TMP)
- de logische en rekenkundige eenheid zelf
- een 5 bits toestandsregister (vlag register)
- een register voor decimale correctie van de accumulatorinhoud bij het verwerken van BCD getallen.

Alle bewerkingen zoals rekenkundige bewerkingen en de logische operaties EN, OF, EXOF die door de ALU uitgevoerd worden, gebeuren met de inhoud van de accumulator en het TMP register.
Het resultaat van de bewerking wordt terug ingeschreven in de accumulator.
Om rondlopen van de informatie te vermijden is de accumulator voorzien van een latch die bij het begin van de operatie de inhoud van de accumulator ontvangt en onmiddellijk daarna nog slechts toegankelijk is voor uitlezing.

Het register voor decimale correctie kan de 8 bits binaire accumulator inhoud omzetten in 2 tetraden in de BCD code.
Het vlagregister, ook wel register van de toestandenbits genoemd, memoriseert bepaalde merkwaardige eigenschappen van het resultaat van een logische of rekenkundige operatie, uitgevoerd door de ALU. Deze eigenschappen worden gesymboliseerd door een bit in het toestandenregister op 1 te zetten of terug te zetten op 0.
De toestanden die in het register opgenomen worden zijn :

- het al (1) of niet (0) (*zero*=z) zijn van de accumulatorinhoud
- het negatief (1) of positief (0) zijn van het teken (*sign*=S) van het resultaat.
- het bestaan (1) of niet bestaan (0) van een overdrachtbit (*carry*=C of CY)
- het bestaan (1) of niet bestaan (0) van een hulpoverdracht (*auxiliary carry*=AC) tussen de eerste en tweede tetrade.
- een even (1) of oneven (0) pariteit (*parity*=P) van het resultaat.

Deze 5 bits hebben een vaste plaats in het 8-bits toestandenregister, wat meteen betekent dat 3 vaste bits in dit register ingeschreven zijn.



3.2.3. Het instructieregister met de decoder

Het instructieregister (IR) ontvangt de uit te voeren instructie uit het programmageheugen (ROM) via de databusbuffer.
Deze instructie wordt dan in de instructiedecoder gedecodeerd om uit te maken wat dient uitgevoerd te worden. Deze decodering gebeurt in een inwendige ROM. Het resultaat van deze decodering zet uiteindelijk een signaal op de controlebus. Dit signaal maakt de gewenste onderlinge verbinding van de verschillende schakelingen van het systeem.
De controlesignalen kunnen zijn :

MEMR *Memory read* wanneer de instructie gepaard gaat met een lezing uit het geheugen hetzij ROM, hetzij RAM.

MEMW *Memory write* wanneer de instructie een inschrijving in het geheugen RAM veroorzaakt.

I/OR *Input/output read* wanneer een van de ingangskanalen gelezen wordt.

I/OW *Input/output write* wanneer in een van de uitgangskanalen geschreven wordt.

INTA *Interrupt acknowledgement* (kennisgeving van ontvangst van een onderbrekingsaanvraag) wanneer het een onderbreking betreft van een interruptprogramma dat in uitvoering is (zie verder).

De controlesignalen zijn niet zonder meer op de klemmen van de microprocessor aanwezig maar als een 8 bits statuswoord op de databus, afwisselend met de gegevens. Dit veronderstelt dat de inhoud van de databus gemultiplexeerd is tussen de gegevens (data) en statussignalen waaruit de controlesignalen opgebouwd worden. Met behulp van een speciale bouwsteen (bv. 8228) wordt deze informatie op de databus omgezet in de 5 controlesignalen (zie verder).

3.2.4. Het registerveld met de adresseringslogica

Het registerveld (*scratch pad*, kladblok) is in feite een inwendig geheugen met beperkte afmetingen waarin de CPU rechtstreeks kan schrijven of lezen door aangepaste instructies. Het voordeel van dit inwendig geheugen is de snelheid van afwikkeling van de operaties. Alles gebeurt binnen de CPU en er moet niet steeds gelezen of geschreven worden in het uitwendig geheugen. Het registerveld bestaat uit 8 registers van 8 bits, die kunnen samengevoegd worden tot drie 16 bits registers. Het zijn de registers B, C, D, E, H en L die eveneens kunnen aangesproken worden als registerparen B, D en H.
Twee bijkomende registers W en Z worden alleen gebruikt voor de inwendige verwerking van sommige instructies door de microprocessor; toegang tot deze registers via een programma is onmogelijk.
We merken op dat de registers H en L een dubbele functie hebben. Enerzijds kunnen ze dienst doen als inwendig geheugen, zoals hoger aangegeven. Anderzijds wordt het HL registerpaar gebruikt als adresseringspointer en bevatten H en L respectievelijk de hoogste byte (*high*) en de laagste byte (*low*) van een geheugenadres bij indirecte adressering.
Naast deze registers van algemeen nut zijn er nog 2 registerparen die een vaste functie hebben in verband met adresseringslogica : het PC register (*program counter*) of de instructieteller. Dit registerpaar bevat steeds het adres van de volgende uit te voeren instructie. Het wordt door een RESET op 0 gezet, d.w.z. na RESET is het adres 0000H ingeschreven in het PC register. De eerstvolgende instructie die zal uitgevoerd worden bevindt zich dus op adres 0000H van het programmageheugen. Het registerpaar SP (*stack pointer*) of stapelwijzer bevat het adres van de laatste geheugencel die gebruikt is in het stapelgeheugen (*stack*).
Het beginpunt van dit stapelgeheugen wordt vastgelegd in het programma en begint gewoonlijk bij het hoogste physische RAM adres. Inschrijving of lezing in de *stack* gebeurt steeds per 2 bytes. Bij inschrijving in het stapelgeheugen telt de stapelwijzer af. Bij lezing uit het stapelgeheugen daarentegen telt hij op.

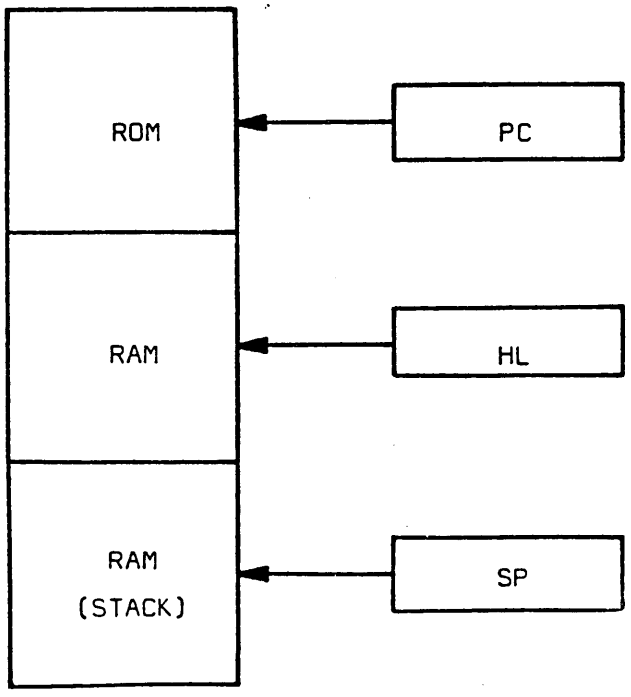


fig. 3.3

De *stack* wordt door de microprocessor gebruikt, o.a. om met terugkeeradres in te schrijven wanneer er gewerkt wordt met subroutines of om de inhoud van die registers veilig te stellen die nodig zijn om een hoofdprogramma te kunnen verderzetten na een onderbreking (*interrupt*). Resumerend kunnen we zeggen dat in het registerveld 3 adresseringswijzers aanwezig zijn (fig. 3.3):

- de programmateller (*program counter*) wijst het adres aan van een instructie in het ROM geheugen.
- het HL registerpaar wordt gebruikt als adreswijzer (*pointer*) in het RAM geheugen.
- de stapelwijzer (*stack pointer*) is de adreswijzer van het speciaal gedeelte in het RAM geheugen dat de stapel of *stack* wordt genoemd.

3.2.5. Het controle gedeelte

Het controlegedeelte is verantwoordelijk voor de besturing en de controle van de centrale verwerkingseenheid.
De ingangscontrolesignalen zijn :

RESET — Het signaal waarmee de microprocessor op nul gezet wordt zodat de uitvoering van een programma kan beginnen. Het RESET signaal zet de instructieteller (PC) op nul, terwijl de andere registers niet beïnvloed worden. RESET is aktief met niveau hoog en moet een duur hebben van minstens 3 klokperiodes.

$Ø_1$ en $Ø_2$ — Deze twee signalen vormen een tweefasenklok voor de tijdsbesturing van de CPU met een amplitude van 8V, opgewekt door een afzonderlijke bouwsteen 8224. Het spanningsverloop van deze kloksignalen wordt gegeven in fig. 3.4.

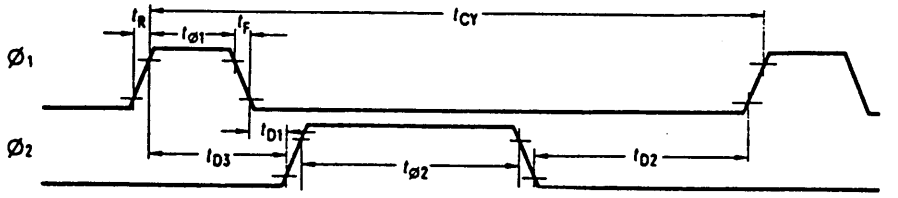


fig. 3.4

INT — Interrupt is het signaal dat aan de CPU meedeelt wanneer een van de periferie apparaten het hoofdprogramma wenst te onderbreken. Het is dus een aanvraag tot onderbreking.

READY — Is het signaal dat aan de CPU meedeelt dat het (traag) geheugen of het (trage) periferie apparaat nog niet klaar is met de uitvoering van de gevraagde cyclus. De CPU wacht en kan niet verder werken voor het READY signaal op hoog komt. READY is aktief met niveau laag.

HOLD — Is eveneens een vraag tot stoppen van de kativiteiten van de CPU met dit verschil dat nu zowel adres als gegevensbuffers in de hoge impedantietoestand worden gezet (3-state) en de CPU zich isoleert van dit bussysteem zodat een andere processor bv. direkte toegang tot het geheugen DMA) de besturing van de bussen van het systeem kan overnemen. HOLD is aktief met een hoog niveau.

De signalen die vertrekken van het controlegedeelte zijn :

$\overline{WR}$ Het signaal dat aangeeft op de operatie die volgt een schrijf- of een leesopdracht is.
Niveau laag geeft schrijven.
Niveau hoog geeft lezen.

DBIN (*data bus in*) Door dit signaal geeft de processor te kennen dat gegevens ingelezen kunnen worden via de bidrirectionele databus (niveau hoog) of uitgelezen worden (niveau laag). DBIN is eveneens laag bij het uitsturen op de databus van de status-signalen die dan verder gecodeerd dienen te worden om de controlebussignalen te vormen van de 5 bits controlebus.

HLDA (*hold acknowledgement*) is het signaal dat de ontvangst van een HOLD aanvraag bevestigt met een hoog niveau.

SYNC Dit synchronisatiesignaal geeft het begin aan van elke nieuwe machinecyclus en synchroniseert de afwikkeling van de verschillende onderdelen van een instructie, in functie van de tijd. Dit signaal wordt hoog na de opgaande flank van $Ø_2$ tijdens de eerste klokperiode tot de volgende opgaande flank van $Ø_2$. (zie fig. 3.4).

WAIT Met WAIT geeft de CPU te kennen dat hij wacht op het READY signaal dat kan uitgegeven worden door een traag geheugen of een traag periferie apparaat.
WAIT is aktief voor niveau hoog.

Figuur 3.5 geeft de bezetting van de pennen van het microprocessor-IC. Het is een 40 pennen IC waarop al de hoger vermelde signalen terug te vinden zijn, nl. :

- 16 adressignalen
- 8 gegevenssignalen
- 12 signalen van het besturings- en controlesysteem

Tevens zijn er 4 pennen beschikbaar voor de 3 voedingsspanningen waarvan de 8080 gebruik maakt en de massa.

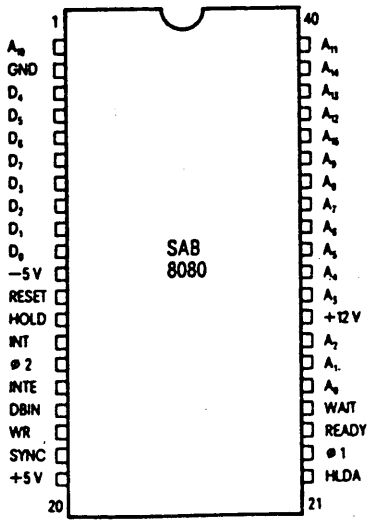


fig. 3.5

RGB-VIDEO

****
* ADAPTEUR DAI/VIDEO MONOCHROME *
****

D'apres Radio-Plans de juillet 1983 No 428 p 20

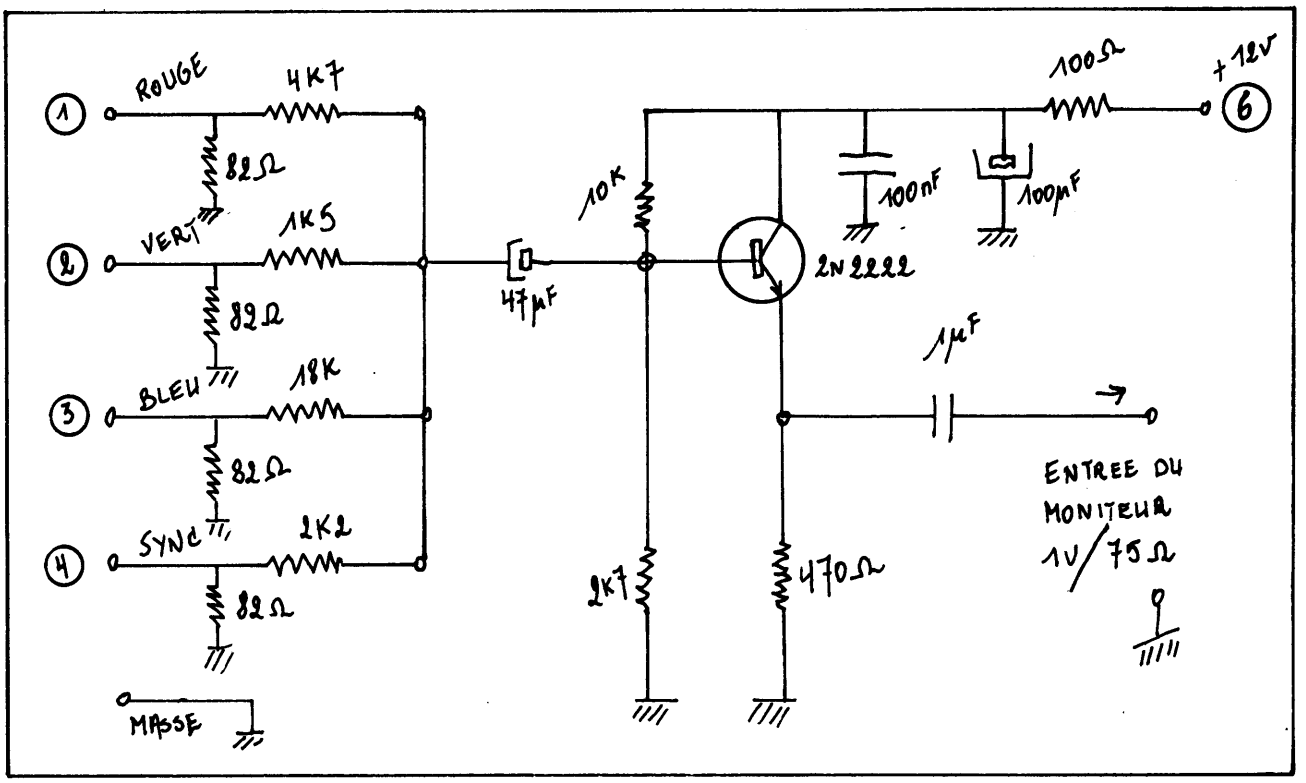
Alain Mariatte

Pour utiliser un ordinateur a sorties Peritel (RGB) comme le DAI avec un moniteur monochrome necessitant un signal composite de 1V crete sur 75 Ohms (moniteur qui equipe la majorite des ordinateurs de table), il faut faire la somme des signaux que le DAI sort sur la prise "VIDEO".
Cette somme doit respecter la relation:

$$Y = 0.3\ R + 0.59\ V + 0.11\ B$$

Le circuit suivant, tres simple et tres largement inspire de l'article cite en reference, assure ce travail et permet de se passer momentanement de l'encombrant tele couleur. On y perd bien-entendu la couleur et le son, mais ce n'est pas grave pour bien des applications (et on gagne en facilite de transport: on fait actuellement des moniteurs vraiment legers).

Les numeros des broches sont ceux de la prise DIN sortie PERITEL (RGB) du DAI. Bien entendu, la sortie son '5' n'est pas utilise ici.



6. DE TICC

Vraag 5: TICC is een afkorting van .....

De tweede I/O-module waarover de DCE beschikt is de TICC (Timer Interrupt and Communications Controller).
De TICC wordt gevormd door de TMS 5501 van Texas Instruments.
De TICC bestaat uit de volgende delen.
a. Een 8-bits input-poort.
b. Een 8-bits output-poort.
c. Een serial input- en output.
d. Een aantal timers.
e. Een interrupt controller.

In fig. 6 is een gedetailleerd blokschema van de TICC weergegeven. Tussen haakjes is aangegeven welk adres elk register en elke timer heeft.
(Om de tekening overzichtelijk te houden zijn de selectie-lijnen van de adres decoder en de besturingsbus niet getekend.)

7. PARALLEL INPUT/OUTPUT

De TICC beschikt over een 8 bits input-poort en een 8-bits output-poort via welke de data van of naar de CPU wordt getransporteerd.
De modes van deze poorten kunnen, in tegenstelling tot de GIC-poorten, niet veranderd worden.

Vraag 6: Met de instructie STA 9807H brengen we de
..... naar .....

Willen we b.v. de inhoud van de accumulator naar de output-poort met adres 9807_{16} brengen, dan gebruiken we de instructie STA 9807H.
Met de instructie LDA 9801H halen we de informatie op de input-poort naar de accumulator.

Opmerking:

Een opgaande flank op b_7 van de input-poort genereert tevens een interrupt request. Deze komt binnen op b_7 van het interrupt register (zie paragraaf 10). Deze interrupt noemen we de auxiliary interrupt.

8. SERIAL INPUT/OUTPUT

Voor de communicatie met b.v. een TTY of een display beschikt de TICC over een serial input en een serial output.

Willen we een 8-bitswoord in serie (dus achter elkaar) overzenden, dan brengen we dit woord naar het transmitter register (to transmit = zenden), die ervoor zorgt dat de bits achter elkaar de lijn worden opgestuurd.

Antw.5: Timer, Interrupt and Communications Controller.
Antw.6: inhoud van de accumulator; output-poort.

Het transmitter register zorgt zelf voor het opwekken van de start- en stop-bits (fig. 7).

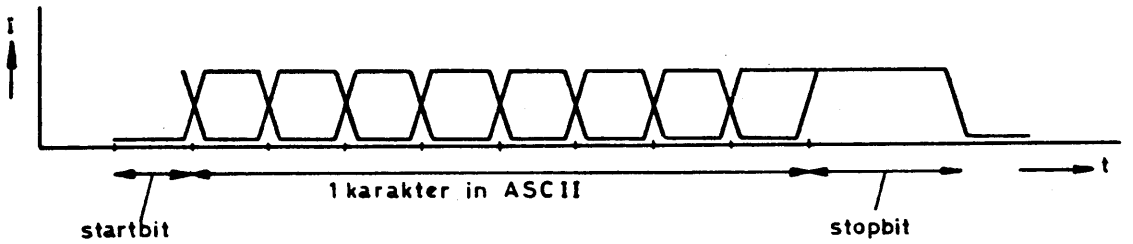


fig.7

Vraag 7: De baud rate is het aantal ..... per .....

De baud rate (= aantal bits/sec) waarmee dit overzenden gebeurt, moeten we van te voren aanpassen aan de baud-rate van het randapparaat.

b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0
stop-bit select	9600 Baud	4800 Baud	2400 Baud	1200 Baud	300 Baud	150 Baud	110 Baud

fig.8

In fig. 8 is het baud-rate register weergegeven. Door één bepaalde bit 1 te maken, kunnen we de baud-rate waarmee de TICC data ontvangt en overzendt instellen.
Door b_7 1 of 0 te maken kunnen we bepalen of het transmitter-register bij het overzenden van een karakter 1 resp. 2 stopbits genereert.

Vraag 8 : Willen we de baud rate instellen op 110 Baud en het transmitter register 2 stopbits laten genereren, dan moeten we naar adres $.....{}_{16}$ de waarde $.....{}_{16}$ sturen.

Voor een baud-rate van b.v. 110 moeten we b_0 van het transmitter register 1 maken en de bits b_6 t/m b_1 0.
Vereist het randapparaat b.v. 2 stopbits, dan dient b_7 0 te zijn.
In dit geval moeten we het baud-rate register (adres 9805) vullen met $00000001_2 = 01_{16}$.

Wanneer het transmitter register alle 8 bits van een karakter en de start- en stopbits de lijn op heeft gezonden, geeft het een interrupt request om aan te geven, dat de CPU nieuwe data mag zenden.

Op dezelfde manier geeft het receiver register (to receive = ontvangen) een interrupt request wanneer een 8 bits woord en de start- en stop-bits zijn ontvangen. De CPU kan de data dan overnemen.

Vraag 9 : De CPU brengt data van het receiver register naar de accumulator m.b.v. de instructie .....

De CPU doet dit m.b.v. een instructie die data overbrengt van geheugenwoord met adres 9800_{16} (het receiver register) naar de accumulator. Dit is b.v. de instructie LDA 9800H.

Antw.7: bits per seconde. Antw.8: 9805_{16}; 01_{16}. Antw.9: LDA 9800. 20-01

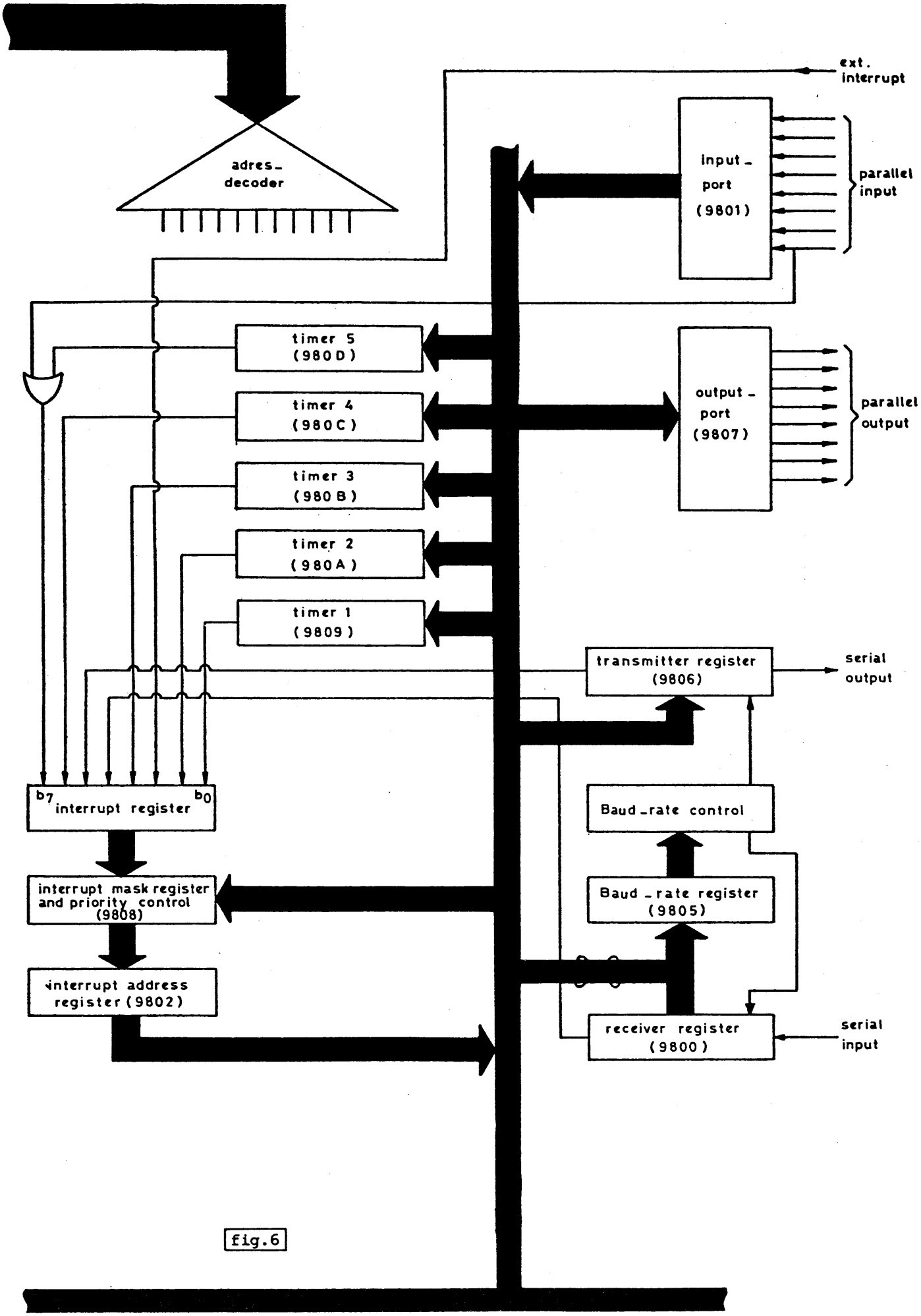

ext.
interrupt
adres_
decoder
input_
port
(9801)
parallel
input
timer 5
(980D)
timer 4
(980C)
timer 3
(980B)
timer 2
(980A)
timer 1
(9809)
output_
port
(9807)
parallel
output
transmitter register
(9806)
serial
output
b7 interrupt register b0
Baud_rate control
interrupt mask register
and priority control
(9808)
Baud_rate register
(9805)
interrupt address
register (9802)
receiver register
(9800)
serial
input

fig.6

Opmerking: Het voordeel van deze "interrupt-methode" is, dat de CPU verder kan gaan met de uitvoering van het hoofdprogramma, terwijl het transmitter register of het receiver register bezig is de data de lijn op te zenden of te ontvangen.

SAMENVATTING 2

4. TICC betekent Timer, Interrupt and Communications Controller. De TICC beschikt over
   a. een 8-bits input-poort en een 8-bits output-poort,
   b. een serial in- en output,
   c. 5 timers,
   d. een interrupt controller.
5. Wanneer we gebruik maken van de serial in- en output van de TICC, moeten we van te voren de Baud-rate instellen, door één van de bits van het Baud-rate register 1 te maken. Tevens moeten we aangeven of een karakter gevolgd wordt door 1 of door 2 stopbits.
6. Het in serie overzenden en ontvangen van een karakter geschiedt op basis van interrupt I/O. Wanneer een karakter is overgezonden of ontvangen, zendt de TICC een interrupt request naar de CPU, zodat de CPU nieuwe data naar de TICC kan zenden of data vanuit de TICC kan ophalen.

cont. from p.386

ressen elk gescheiden door een space (cfr het Display bevel uit de utility). Het eerste adres is het beginadres van de geheugenzone die in EPROM dient gebracht te worden (begin sourcebuffer), het tweede adres is het eindadres van deze geheugenzone (eindadres sourcebuffer), het derde adres ten slotte is het beginadres waar de data dienen ondergebracht te worden in de EPROM.

Hiervoor dient altijd het beginadres van een blok van 256 byte (pagina) genomen te worden. Gedurende de programmatie brandt de rode LED. Na het voltooien van de programmatie, test het programma of alles foutloos is gebeurd en meldt zich met

PROGRAMMING DONE

gevolgd door de melding NO ERRORS of

ERROR AT ADDRESS xxxx .

xxxx staat voor het hexadecimale adres waar de eerste fout gelezen is.

Het is eveneens mogelijk de inhoud van een EPROM te lezen. Het bevel is (R) 'READ' en de geheugenruimte beginnende bij het hexadecimale adres A300 wordt dan gevuld met de inhoud van de EPROM.

Wenst men terug te keren naar UTILITY dan kan dit gebeuren door het bevel (U).

2 EPROM's mogen slechts in de sokkel gestoken of uit de sokkel gehaald worden, als de groene LED brandt.

```
10    REM ================================================
11    REM titel           KERSTNACHT
12    REM datum           82-12-20
13    REM (c)             Herman MOEYS - 1982
14    REM ================================================
20    REM INITIALIZATIE ----------------------------------
21    POKE #75,32:PRINT CHR$(12):COLORT 0 0 0 0:COLORG 0 0 0 0:MODE 5A
:POKE #744B,#6A:CLEAR 10000:RESTORE
22    DIM KL(5.0),XL(24.0),YL(24.0),XS(3.0,99.0),KS(15.0),T$(1.0)
23    FOR I=0 TO 5:READ T:KL(I)=T:NEXT:FOR I=0 TO 15:READ T:KS(I)=T:NE
XT
24    DATA 1,2,3,10,12,14,8,1,2,3,4,15,6,7,0,9,10,11,12,13,14,15
25    MS=30:XM=XMAX+1:YM=YMAX+1:XC=118+INT(RND(13.0))*8.0
26    T$(0.0)="          PRETTIGE  KERSTDAGEN          "
27    T$(1.0)="           VOORSPOEDIG   1984           "
30    REM KERSTBOOM HALEN --------------------------------
31    CURSOR 0,1:PRINT "     een kerstboompje halen  ....":COLORT 0 10
 0 0
32    YC=171:BB!=20.0:EB!=2.0:TB=24:GOSUB 100
33    YC=147:BB!=35.0:EB!=10.0:TB=26:GOSUB 100
34    YC=117:BB!=55.0:EB!=17.0:TB=34:GOSUB 100
35    YC=79:BB!=80.0:EB!=27.0:TB=44:GOSUB 100
36    YC=35:BB!=110.0:EB!=40.0:TB=52:GOSUB 100
37    FOR Y=45 TO 0 STEP -1:DRAW XC-10,Y XC+10,Y 6:DOT XC-11,Y 15:DOT
XC+11,Y 15:NEXT
40    REM LAMPJES HANGEN ---------------------------------
41    T=0
42    COLORT 0 0 0 0:CURSOR 0,1:PRINT "     enkele lampjes ophangen  .
..    ":WAIT TIME 30:COLORT 0 10 0 0
43    X=RND(XM):Y=RND(YM):IF SCRN(X,Y)<>5 GOTO 43
44    XL(T)=X:YL(T)=Y:L2=T MOD 6:GOSUB 200
45    T=T+1
46    IF T<25 GOTO 43
50    GOTO 1000
100   REM [S] KERSTBOOM TEKENEN --------------------------
101   TAU!=TB/LOG(BB!/EB!)
102   FOR T=TB TO 0 STEP -1
103   X=BB!*EXP(-T/TAU!)-2.0
104   DRAW XC-X,YC+T XC+X,YC+T 5
105   DOT XC-X,YC+T 15
106   DOT XC+X,YC+T 15
107   NEXT
108   RETURN
200   REM [S] LAMPJES KLEUREN ----------------------------
201   K=KL(L2)
202   FILL X-3,Y-3 X+3,Y+3 K
203   FILL X-1,Y-4 X+1,Y+4 K
204   K=INT(K/8.0)*15.0
205   DRAW X-4,Y X+4,Y K
206   DRAW X-4,Y-1 X+4,Y-1 15-K
207   DRAW X-4,Y+1 X+4,Y+1 15-K
208   RETURN
1000  REM KERSTNACHT -------------------------------------
1100  REM KNIPPERENDE STER -------------------------------
1110  FOR L1=0 TO 5
1120  COLORT 0 0 0 0:YC=200:K=KL(L1)
1130  FOR I=0 TO K STEP K
1131  DRAW XC-10,YC XC+10,YC I
1132  DRAW XC-7,YC-7 XC+7,YC+7 I
1133  DRAW XC-7,YC+7 XC+7,YC-7 I
1134  DRAW XC,YC+10 XC,YC-10 I
1135  DRAW XC-7,YC-3 XC+7,YC+3 I
1136  DRAW XC-3,YC-7 XC+3,YC+7 I
1137  DRAW XC-3,YC+7 XC+3,YC-7 I
```

```
1138  DRAW XC-7,YC+3 XC+7,YC-3 I
1139  NEXT
1140  CURSOR 0,1:PRINT T$(L1 MOD 2.0):COLORT 0 10 0 0
1200  REM KNIPPERENDE LAMPJES ------------------------
1210  FOR L2=0 TO 5
1220  T=RND(25.0):X=XL(T):Y=YL(T):GOSUB 200
1300  REM SNEEUWVLOKJES ------------------------------
1310  FOR L3=0 TO 5
1320  X=RND(XM):Y=RND(YM)
1321  K=KS(SCRN(X,Y))
1322  HS=XS(X/100.0,X MOD 100.0)
1323  IF HS<MS-30.0 THEN HS=MS-30
1330  DOT X,Y K
1331  DOT X,HS 15
1340  XS(X/100.0,X MOD 100.0)=HS+1
1350  IF HS<MS GOTO 2000
1360  FOR I=0 TO XM-1
1361  IF SCRN(I,HS)=8 THEN DOT I,HS 15
1362  NEXT
1370  DRAW 0,HS-30 XM-1,HS-30 15
1380  MS=MS+1
1390  IF MS=212 GOTO 10
2000  NEXT:NEXT:NEXT:GOTO 1000
```

DELETE

HOE WIS JE EEN GEDEELTE VAN EEN PROGRAMMA UIT?

Het komt nogal vaak voor dat men een gedeelte van een programma dat overbodig geworden is wil verwijderen, of dat men een onderdeel uit een groot programma wil afzonderen. Sommige BASIC-dialecten gebruiken daarvoor een speciaal bevel (bvb. DELETE in TRS 80-basic), maar in DAI-basic is dit niet voorzien.

Toch zijn er een aantal mogelijkheden om iets uit te wissen:

1. Tik gewoon de nummers van de te wissen regels opnieuw in. Deze methode is omslachtig als je grote gedeelten van een programma wil verwijderen.

2.-Zet het programmadeel dat je wil bewaren in de EDIT-buffer
   -Wis de tekstbuffer
   -Copieer de EDIT-buffer in de tekstbuffer.

We willen bijvoorbeeld een subroutine bewaren die op regel 1ØØØ tot 2ØØØ staat, en de rest van het programma vernietigen. Dit kan op volgende manier:

```
*CLEAR 3ØØØ
*EDIT 1ØØØ-2ØØØ
 ⟨BREAK⟩ ⟨BREAK⟩
*NEW
*POKE 3Ø9,2
```

Deze methode is vooral geschikt om programmaonderdelen uit een groot programma af te zonderen (bvb. een subroutine die men in een ander programma wil gebruiken). Ze kan enkel gebruikt worden als het te bewaren stuk een aaneensluitend geheel vormt.

3. De "Monte Carlo-methode": als je zeer snel na het indrukken van de RETURN- toets achter 'EDIT1ØØØ-2ØØØ" 2X⟨BREAK⟩ duwt is het mogelijk dat er een gedeelte van het programma tussen 1ØØØ en 2ØØØ verdwenen is. De rest kan je dan met methode 1 verwijderen. Deze methode vindt enkel aanhangers bij personen die het genoegen willen smaken sneller te zijn dan de computer en hoort dus eerder thuis in het hoofdstuk "Games&Strategy" dan in het hoofdstuk "Serieus programmeren" Af te raden!

4. Om bvb. regels 7Ø - 12Ø te verwijderen doe je (in command mode): *EDIT 7Ø-12Ø :EDIT n (n= willekeurig nummer)

Op het scherm verschijnen dan regels 7Ø-12Ø in EDIT-mode. Doe dan ⟨BREAK⟩ ⟨SPATIE⟩ . Regel n verschijnt op het scherm; doe ⟨BREAK⟩ ⟨BREAK⟩
Deze methode blijkt feilloos te werken, al kan ik niet verklaren hoe.

Ik heb ook getracht deze "truuk" in te bouwen in een basic-programma, maar dan blijkt het niet in alle gevallen te werken. (met behulp van het programma van de heer Dufour uit Dainamic 16, p.200)

Als je grote gedeelten uit een programma verwijderd hebt is het aan te raden de symbol table weer op te poetsen. Daar staan immers nog steeds de variabelen in die uit het programma verdwenen zijn.

```
*CLEAR xxxxx      (voldoende om het ganse programma te bevatten)
*EDIT
⟨BREAK⟩ ⟨BREAK⟩
*NEW
*POKE 3Ø9,2
```

J o s V a n d e b e r g h

---

p.s. don't forget your subscription

1984

JANUARY						
SUN	MON	TUE	WED	THU	FRI	SAT
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31				

FEBRUARY						
SUN	MON	TUE	WED	THU	FRI	SAT
			1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29			

MARCH						
SUN	MON	TUE	WED	THU	FRI	SAT
				1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	31

APRIL						
SUN	MON	TUE	WED	THU	FRI	SAT
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30					

MAY						
SUN	MON	TUE	WED	THU	FRI	SAT
		1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30	31		

JUNE						
SUN	MON	TUE	WED	THU	FRI	SAT
					1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30

JULY						
SUN	MON	TUE	WED	THU	FRI	SAT
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31				

AUGUST						
SUN	MON	TUE	WED	THU	FRI	SAT
			1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30	31	

SEPTEMBER						
SUN	MON	TUE	WED	THU	FRI	SAT
						1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30						

OCTOBER						
SUN	MON	TUE	WED	THU	FRI	SAT
	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30	31			

NOVEMBER						
SUN	MON	TUE	WED	THU	FRI	SAT
				1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	

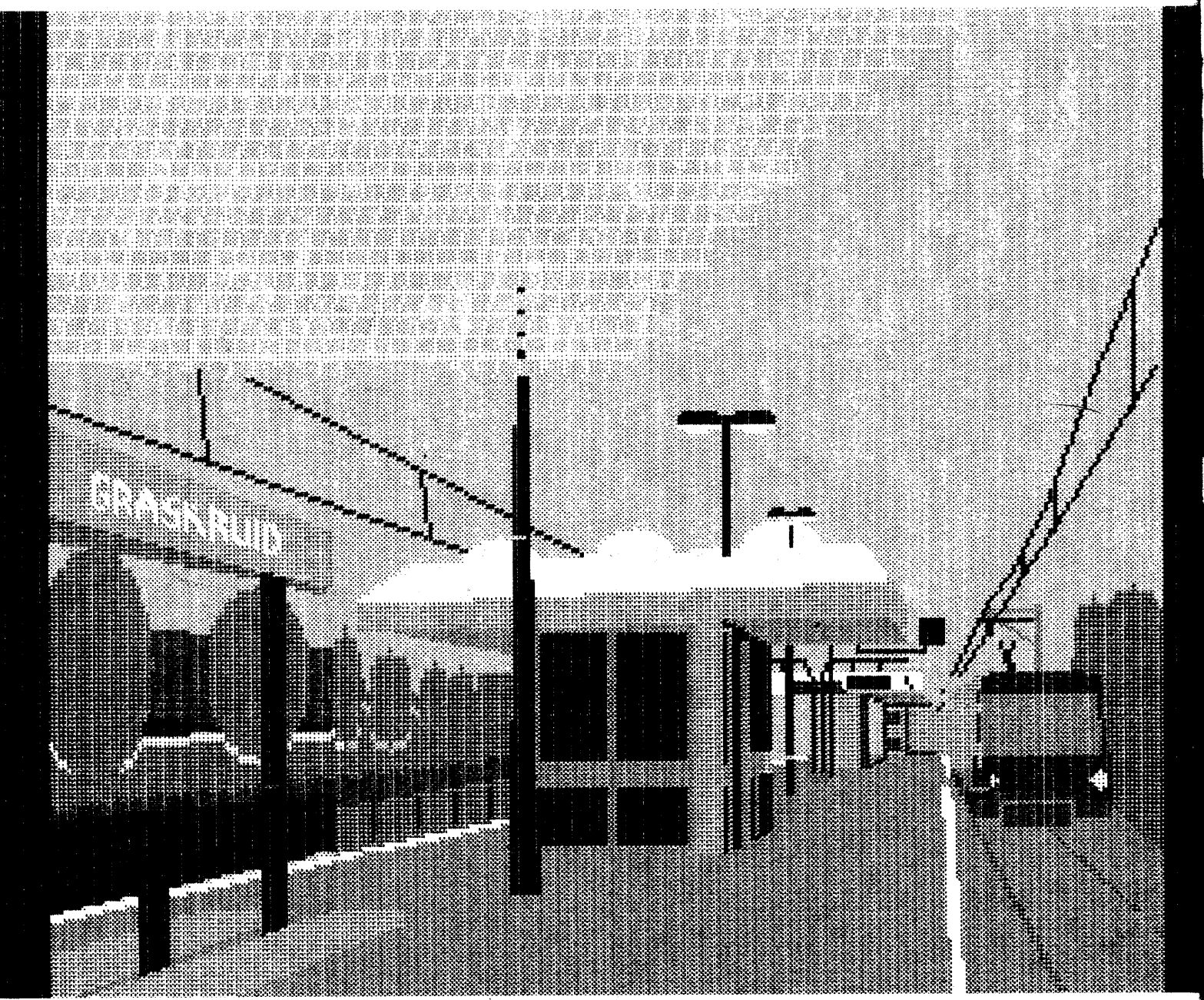
DECEMBER						
SUN	MON	TUE	WED	THU	FRI	SAT
						1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30	31					

Morrison

4th international DAInamic meeting on saturday 21 april in Tongelsbos, Westerlo.

DAI NAMIC

DAI graphics



Jeroen Overvoorden

SEND YOUR DRAWINGS TO DAINAMIC EDITOR