J. Neerhof
05730 - 5,886

PRACTICUM-HANDLEIDING DIGITALE REALISATIE en COMPUTERTECHNIEK 1

- Veel commando's kunnen ook verkort worden gegeven door middel van control-codes. Deze staan in de menus vermeld, waarbij bv. ^K^B betekent: *control-K*, *control-B*, waarmee het begin van een block wordt gemarkeerd.
- De belangrijkste commando's in de diverse submenus zijn:

Block Verplaatsen, kopieren, verwijderen en tussenvoegen van aaneengesloten stukken tekst. Van belang is vooral het commando *Read* om de file IOSIM.ASM met de IO-macros in te voegen.

Search Zoeken en vervangen van tekst. Geef als optie *UN*, wat betekent dat er niet wordt gelet op hoofdletters of kleine letters.

Go to Positioneren van de cursor in de tekst.

Text format Instellen van diverse opties. *Auto indent ON* betekent dat nieuwe regels beginnen op dezelfde positie als de voorgaande regel. Tabs staan standaard ingesteld op elke 10e positie, wat voor assembly gebruikelijk is.

Window Opdelen van het scherm in twee delen, waarin twee delen van dezelfde file geedit kunnen worden.

File Manipuleren van files op schijf. De belangrijkste commando's zijn *Open*, *Save* en *Quit*. *Open* gebeurt automatisch bij het starten van !ED, *Save* gebeurt automatisch bij *Quit*.

4.4 De vertaler (assembler): MAC

Nadat een programma via !ED is ingetypt moet het worden geassembleerd, als we uitgaan van een programma in assembleertaal. In CP/M is daartoe de assembler MAC standaard beschikbaar. Deze vertaalt de assembleertaal in de 8080\8085 machinetaal. Naast die vertaling kan bovendien een listing geproduceerd worden die de oorspronkelijke invoer en de vertaling bevat.

Een assembleertaal programma bestaat uit regels met statements. De opbouw van zo'n statement ziet er als volgt uit:

```
|-----------------------------------------------|
| label: | opcode | operanden |   ; commentaar   |
|-----------------------------------------------|
```

```
BEGIN:    MVI     C,LIST    ;PRINT HET KARAKTER
```

ELK VAN DEZE REGELS MOET WORDEN AFGESLOTEN MET EEN RETURN!
Niet alle "velden" (b.v. label) hoeven in ieder statement aanwezig te zijn. De betekenis van ieder veld wordt nu in het kort weergeven.

Label

Ook een label hoeft niet te worden gebruikt, maar is wel toegestaan. Als een label wordt gebruikt dient deze te bestaan uit maximaal 16 karakters, of cijfers (zgn. alphanumerike karakters), waarbij het eerste karakter een letter moet zijn. Een label mag (hoeft niet) door een : (dubbele punt) te worden gevolgd. Verder worden alle letters opgevat als hoofdletters,

ook al zijn ze ingetypt als kleine letters. Dus AAP is identiek aan Aap maar ook aan aap. Deze maatregelen zijn genomen om zo goed mogelijk vergelijkbaar te blijven met andere, veel gebruikte assemblers.
Een label wordt gewoonlijk gebruikt voor de identificatie van sprongadressen, adressen van data en ter vergroting van de leesbaarheid van het programma. Kies daarom zo mogelijk een duidelijke naam voor een label!
MAC zal tijdens de vertaling de labelnaam vervangen door een waarde, die verder in de vertaling zal worden gebruikt.

Bewerking (Opcode)

De bewerking is een 8080\8085 instructie (volgens de 8080\8085 mnemonics, ofwel naamgeving) of een assembleer aanwijzing (Engels: assembler directive). De 8080\8085 instruktie geeft direct de bewerking weer die door de 8080\8085 kan worden uitgevoerd. Voorbeelden: ANI, CALL. De assembler vertaalt deze naam naar de bijbehorende code in machine taal (dus in bits). Voor de assembler instructies van de 8080\8085 is een apart overzicht aanwezig.

Operand

Het operandveld kan geen, een of twee operanden bevatten die moeten worden gescheiden door komma's. Iedere operand bestaat uit een expressie (ofwel uitdrukking) van labels en/of constanten. Het verdient aanbeveling uit te gaan van leesbare namen en expressies die een direct begrijpbare betekenis hebben.
De operand kan bestaan uit een getalwaarde die decimaal of hexadecimaal wordt opgegeven. Bij de hexadecimale weergave moet de waarde door een H worden gevolgd en altijd met een van de getallen 0 t/m 9 beginnen (bijv. 0FH). Appendix B toont de omzetting van hexadecimale naar decimale en binaire getallen.
Bepaalde te gebruiken namen hebben in het operandveld een gereserveerde betekenis. Dit slaat op de door de 8080\8085 gebruikte registers, zoals A, B, C, SP, enz.
Ook kunnen strings in het operandveld staan, die zijn opgebouwd uit ASCII karakters. Zo'n string moet tussen aanhalingstekens staan ter herkenning, bijv. 'FOUTMELDING'. Een string is dus gewoon een stuk tekst.
Er mag een beperkt aantal arithmetische en logische bewerkingen in de operand voorkomen. De belangrijksten daarvan zijn +, -, NOT, AND en OR.
Voorbeelden van operanden zijn: ANTW, BYTE, 15+ANTW, 15H-ANTW, ANTW-10, ANTW AND BYTE.
ASM zal deze waarden uitrekenen en in de vertaling gebruiken.
In de afgebeelde programma's zien we voorbeelden van operanden.
In regels die uitsluitend uit commentaar bestaan zullen de operanden ontbreken. Dus ook operanden hoeven niet perse aanwezig te zijn; de bewerking bepaalt of er geen, een of twee operanden nodig zijn.

Commentaar

Het laatste veld dat in een statement voor MAC kan voorkomen is het commentaarveld. Dit commentaarveld moet met een ; beginnen. De tekst achter de ; wordt door de assembler genegeerd. Voor de programmeur is de tekst wel van belang, omdat hij op deze wijze toe kan lichten wat hij in een programma precies bedoelt. Het commentaar moet dan ook zo worden gegeven dat het programma leesbaar en begrijpelijk is.
Geheel lege regels worden als commentaar opgevat. Er behoeft dan geen ; in te staan. Geheel lege regels kunnen dienen als afscheiding tussen delen van het programma.

Assembler directives

Een assembler directive geeft aanwijzingen voor de assembler zelf. Zo'n directive komt dus niet direct vertaald weer te voorschijn.

De belangrijkste toegestane directives zijn:

ORG — geeft het begin adres van de vertaalde code
```
formaat:    label   ORG  expressie
voorbeeld:  START:  ORG  100H
```

END — beeindigt het te vertalen programma
```
formaat:    label   END  expressie
                De expressie geeft het start adres aan
voorbeeld:  EINDE:  END  100H
```

EQU — geeft een waarde aan een symbolische naam
```
formaat:    label   EQU  expressie
voorbeeld:  CR      EQU  0DH
```

DB — definieert data bytes of data strings
```
formaat:    label   DB   b1,b2....
voorbeeld:  DATA:   DB   30H,10H,20H
            STRING: DB   'ERROR-CODE',CR,LF,0
```

DW — definieert data woorden
```
formaat :   label   DW   w1,w2,w3....
voorbeeld:          DW   'aa','BB'
```

DS — reserveert geheugen ruimte
```
formaat:    label   DS   expressie
voorbeeld:  BUFFER: DS   10
```

nb. o er zijn in MAC meer directives mogelijk (b.v. voor gebruik macro's), zie hiervoor MAC manual.
o een constante moet altijd beginnen met een numerieke waarde.

vb.: DB FFH
levert een foutmelding, juist is: DB 0FFH

Er kunnen ook regels zonder bewerking worden aangeboden aan MAC. Veelal gaat het dan om regels die uitsluitend uit commentaar bestaan.

RL = Record Length
aantal bytes in data

Aanroep en werking van MAC

De aanroep van MAC bestaat uit: MAC file-naam. De invoer-file van MAC moet als eerste naam file-naam hebben en als file-type ASM. Als de invoer-file bijvoorbeeld VOORB.ASM heet dan moet de aanroep zijn:

MAC VOORB

Het programma MAC produceert als resultaat twee files: een met het file-type HEX en een met het file-type PRN. De file met het type HEX bevat het naar machinetaal vertaalde programma volgens het door Intel gedefinieerde hex-formaat, dat alom wordt gebruikt. De indeling daarvan ziet er als volgt uit:

| : | RL | ADDRESS | CODE | DATA | CHECKSUM |

Alle (event. HEX) karakters staan in ASCII-code
RL = Record Length = aantal bytes in "DATA"
ADDRESS = Geheugen adres van eerste byte
CODE = 00 voor data-record, 01 voor laatste (data-loze) record
DATA = data-bytes
CHECKSUM = (Hexadecimale) som van alle bytes achter de ":" t/m het DATA-veld

De file met het file-type PRN bevat een listing die de vertaling bevat en de oorspronkelijke code plus eventuele foutmeldingen. Bijvoorbeeld:

```
R4444 00 LABEL:  MOV     A,Q      ;  DATA UIT NIET BESTAAND REG.
```

Foutmeldingen worden niet alleen in de file gezet maar ook op het beeldscherm weergegeven. Dus als gevolg van de aanroep:

MAC VOORB

ontstaan twee files: VOORB.HEX en VOORB.PRN.

MAC zal tijdens de vertaling controleren of aan alle taalregels van de assembler is voldaan (syntax check). Vindt MAC fouten dan geeft hij dit in de PRN file eventueel en op het beeldscherm aan. De gebruiker zal dan eerst via de wordprocessor zijn invoer-file (met het file-type ASM) in orde moeten brengen en opnieuw MAC aanroepen totdat geen fouten meer worden gesignaleerd.

In het aangehaalde voorbeeld haalt MAC de invoer-file van de default-schijf. Deze is op het scherm aangegeven door CP/M voor het > teken, bijv. Z80 B>. De HEX file en de PRN file worden op de default-schijf geplaatst. Het is ook mogelijk tijdens de aanroep aan te geven van welke schijf de invoer-file moet worden gehaald, en op welke schijf of schijven de beide uitvoer-files moeten worden gezet. Dan moeten achter de aanroep nog een punt en drie letters worden geplaatst. De eerste letter geeft de schijf van de invoer-file .ASM aan, de tweede van de .HEX file en de derde van de .PRN file. Als letter mag men

```
;#################################################################
;          DEFINITION OF THE MACROS 'INP' AND 'OUTP'
;#################################################################

INP     MACRO   ?INPRT
        PUSH    B               ;SAVE ALL REGISTERS
        PUSH    D
        PUSH    H
        PUSH    PSW
        LXI     H,?INPN         ;INPUTPORTNR. SAVE ADDR.-->HL
        MVI     M,?INPRT        ;SAVE INPUT PORTNR.
        CALL    ?INPUT          ;EXECUTE 'IN' SIM. ROUTINE
        MOV     B,A             ;SAVE ACCUMULATOR
        POP     PSW             ;RESTORE STATUS
        MOV     A,B             ;RESTORE ACCUMULATOR
        POP     H               ;RESTORE 'HL','DE','BC'
        POP     D
        POP     B
        ENDM

;#################################################################

OUTP    MACRO   ?OUTPRT
        PUSH    PSW             ;SAVE ALL REGISTERS
        PUSH    B
        PUSH    D
        PUSH    H
        MVI     B,?OUTPRT       ;PORTNR TO REG. B
        CALL    ?OUTPUT         ;EXECUTE 'OUT' SIM. ROUTINE
        POP     H               ;RESTORE ALL REGISTERS
        POP     D
        POP     B
        POP     PSW
        ENDM

;#################################################################

; INITIATING OF 'PC' AND JUMPING TO THE USER-PROGRAM

        ORG     0100H           ;SET 'PC' TO 0100H
        JMP     ?USER           ;JUMP TO USER PROGRAM


;#################################################################

;       'INPUT' AND 'OUTPUT' ROUTINES
;       THESE ROUTINES PERFORM THE ACTUAL I/O SIMULATION
```

```
;#################################################################

; THE ROUTINE 'OUTPUT' PERFORMS THE ACTUAL 'OUT'-OPCODE
;  SIMULATION

?OUTPUT:LXI     D,?TX1          ;PRINT 'OUT '
        CALL    ?PRSTR

        MOV     C,A             ;SAVE CONTENTS OF A IN C
        MOV     A,B             ;PORTNR TO ACCU
        CALL    ?PRBYTE         ;PRINT PORTNR

        LXI     D,?TX2          ;PRINT ': '
        CALL    ?PRSTR

        MOV     A,C             ;PRINT ACCU
        CALL    ?PRBYTE

        LXI     D,?TX3          ;WRITELN OUTPUT
        CALL    ?PRSTR
        RET

?TX1:   DB      ?CR,?LF,'OUT ','$'
?TX2:   DB      ': ','$'
?TX3:   DB      ?CR,?LF,'$'

;#################################################################

; INPUT ROUTINE: PERFORMS THE ACTUAL 'IN'-OPCODE SIMULATION

?INPUT: LXI     D,?TX4          ;TYPE MESSAGE FOR
        CALL    ?PRSTR          ; INPUT REQUEST

        LDA     ?INPN           ;INPUT PORTNR. TO ACCU
        CALL    ?PRBYTE         ;TYPE INPUT PORTNR.

        LXI     D,?TX2          ;PRINT ': '
        CALL    ?PRSTR

        LXI     D,?BUFFER       ;BUFFER ADDRESS IN 'DE'
        CALL    ?R2KAR          ;READ TWO CHARACTERS
        JC      ?INPUT          ;ERROR:RESTART ROUTINE

        CALL    ?ASCBYT         ;CONVERSION FROM HEX TO BYTE
        JC      ?INPUT          ;ERROR:RESTART ROUTINE

        RET

?TX4:   DB      ?CR,?LF,'IN  ','$'
?INPN:  DS      1               ;INPUT PORTNR. SAVE ADDRESS
```

```
;#################################################################
;                    I/O SERVICE ROUTINES
;#################################################################

;CONSTANT DEFINITION

BDOS    EQU     5H              ;CP/M IO-ROUTINE STARTADDRESS
?CR     EQU     0DH             ;CARRIAGE RETURN
?LF     EQU     0AH             ;LINE FEED


;################################################################

; SUBROUTINE PRSTR SENDS A MESSAGE TO THE SCREEN
;  THE ADDRESS OF THE MESSAGE SHOULD BE
;  LOADED IN REGISTER-PAIR 'DE'

?CONOUT EQU     9               ;BDOS FUNCTION CODE

?PRSTR: PUSH    PSW             ;SAVE ALL REGISTERS
        PUSH    B
        PUSH    D
        PUSH    H
        MVI     C,?CONOUT       ;PRINT STRING WITH
        CALL    BDOS            ; CP/M ROUTINE
        POP     H
        POP     D
        POP     B               ;RESTORE ALL REGISTERS
        POP     PSW

        RET                     ;RETURN

;################################################################

; SUBROUTINE R2KAR READS TWO CHARACTERS INTO A BUFFER
;  THE ADDRESS OF THE BUFFER SOULD BE
;  LOADED IN THE REGISTER-PAIR 'DE'.
;  IN CASE OF AN ERROR THE CARRY FLAG (=ERROR FLAG) WILL
;  BE SET.

?CONSTR EQU     10              ;BDOS FUNCTION CODE

?R2KAR: PUSH    PSW             ;SAVE ALL REGISTERS
        PUSH    B
        PUSH    D
        PUSH    H

        MVI     C,?CONSTR       ;READ STRING OF TWO
        CALL    BDOS            ; CHARS. WITH CP/M ROUTINE

        LDA     ?BUFFER+1       ;IF ACTUAL STRINGLENGTH = 2
        CPI     2               ; THEN GOTO OK
        JZ      ?OK             ; ELSE ERROR

        LXI     D,?MER2         ;PRINT ERROR MESSAGE
        CALL    ?PRSTR

        POP     H               ;RESTORE ALL REGISTERS
```

```
        POP     D
        POP     B
        POP     PSW
        STC                     ;SET ERROR FLAG;
        RET

?OK:    POP     H               ;RESTORE ALL REGISTERS
        POP     D
        POP     B
        POP     PSW
        STC                     ;CLEAR ERROR FLAG
        CMC                     ;  (=CARRY FLAG)
        RET

?BUFFER:DB      4               ;MAXIMUM STRING LENGTH
        DS      1               ;ACTUAL STRING LENGTH
        DS      4               ;STRING BUFFER

?MER2:  DB      ?CR,?LF
        DB      'ERROR: WRONG NUMBER OF CHARACTERS'
        DB      ?CR,?LF,'$'

;###############################################################

; PRBYTE PRINTS THE HEXADECIMAL VALUE OF THE ACCUMULATOR
;  ON THE CONSOLE TERMINAL

?PRBYTE:CALL    ?OUTHL          ;CONVERT AND PRINT LEFT NIBBLE
        CALL    ?OUTHR          ;CONVERT AND PRINT RIGHT NIBBLE
        RET

?OUTHL: PUSH    PSW             ;SAVE ACCU (AND STATUS)
        ANI     0F0H            ;CLEAR RIGHT NIBBLE OF ACCU
        RRC                     ;MOVE LEFT NIBBLE OF ACCU
        RRC                     ;      TO RIGHT NIBBLE
        RRC
        RRC
        CALL    ?OUTHR          ;PRINT RIGHT NIBBLE
        POP     PSW             ;RESTORE ACCU (AND STATUS)
        RET

?OUTHR: PUSH    PSW             ;SAVE ALL REGISTERS
        PUSH    B
        PUSH    D
        PUSH    H
        ANI     0FH             ;CLEAR LEFT NIBBLE
        ADI     '0'             ;ASCII
        CPI     3AH             ;CORRECTION NECESSARY ?
        JM      ?TYP            ;NO
        ADI     7               ;YES:CORRECTION FOR A TO F

?TYP:   MOV     E,A             ;MOVE ASCII CHARACTER TO REG. E
        MVI     C,2             ;FUNCTIONCODE TO REG. C
        CALL    BDOS            ;CALL CP/M IO-ROUTINE
        POP     H               ;RESTORE ALL REGISTERS
        POP     D
        POP     B
        POP     PSW
        RET
```

```
;#################################################################

; SUBROUTINE ASCBYT CONVERTS THE TWO HEXADECIMAL
;  ASCII CHARACTERS PLACED IN THE BUFFER TO A BYTE,
;  WHICH WILL BE PLACED IN THE ACCUMULATOR.
;  IN CASE OF AN ERROR THE CARRY FLAG (=ERROR FLAG)
;  WILL BE SET.

?ASCBYT:PUSH    B               ;SAVE REGISTERS 'BC'
        LDA     ?BUFFER+2       ;CONVERT FIRST HEX
        CALL    ?ASCNIB         ; TO NIBBLE
        JC      ?ER1            ;ERROR: RETURN

        MOV     B,A             ;FIRST NIBBLE IN 'B'
        LDA     ?BUFFER+3       ;CONVERT SECOND HEX
        CALL    ?ASCNIB         ; TO NIBBLE
        JC      ?ER1            ;ERROR: RETURN

        MOV     C,A             ;SECOND NIBBLE IN 'C'
        MOV     A,B             ;  PLACE THE FIRST
        RLC                     ;  NIBBLE IN THE 4
        RLC                     ;  MOST SIGNIFICANT
        RLC                     ;  BITS OF THE
        RLC                     ;  ACCUMULATOR
        ANI     0F0H            ;CLEAR THE OTHER BITS
        ADD     C               ;  THE SECOND NIBBLE
                                ;  IN THE 4 LEAST
                                ;  SIGN. BITS
        STC                     ;RESET ERROR FLAG
        CMC                     ; (=CARRY FLAG)

?ER1:   POP     B               ;RESTORE REGISTERS 'BC'

        RET

;#################################################################

; SUBROUTINE ASCNIB CONVERTS A HEXADECIMAL ASCII CHARACTER
;  PLACED IN THE ACCUMULATOR TO ITS BINARY VALUE
;  IN CASE OF AN ERROR THE CARRY WILL BE SET

?ASCNIB:SUI     '0'             ;SUBTRACT OFFSET OF ASCII
        JM      ?ER2            ;NO HEX CHARACTER
        CPI     10              ;TEST ON A FIGURE
        JM      ?READY
        SBI     'A'-'9'-1       ;SUBTRACT OFFSET FOR LETTERS
        CPI     10
        JM      ?ER2            ;NO HEX CHARACTER
        CPI     16              ;TEST ON A CHARACTER
        JM      ?READY

?ER2:   LXI     D,?MER1         ;ERROR:NOT HEXADECIMAL
        CALL    ?PRSTR          ;PRINT MESSAGE
        STC                     ;SET ERROR FLAG
        RET

?READY: STC                     ;RESET ERROR FLAG
        CMC                     ; (CARRY-FLAG)
        RET
```

```
?MER1:  DB      ?CR,?LF,'ERROR: NO HEXADECIMAL CHARACTERS'
        DB      ?CR,?LF,'$'
?USER:
$+PRINT
;#################################################################
```

4.7 Van PC naar Microtrainer

Na het assembleren en testen van het programma op de PC, moet het programma ook op de Microtrainer worden getest. Daartoe moet de .HEX-file van de diskette naar een microcassette worden gekopieerd. Daarvoor is op het practicum een speciaal systeem aanwezig, bestaande uit PC, Exidy Sorcerer en microcassette recorder. Een handleiding voor het gebruik van dit speciale systeem is op het practicum aanwezig.

4.8 Voorbeeld op de PC

In deze paragraaf laten we zien wat u tijdens het samenstellen t/m het op cassette laden van een programma te zien kunt krijgen op uw beeldscherm. De cursieve tekst is door de gebruiker ingetypt; enige uitleg staat steeds tussen accolades.

(Aanzetten van de PC)

(Na diverse meldingen verschijnt de prompt Z80 B>)

Z80 B>*!ED VOORB.ASM*

{U komt nu in de editor waarin de source file wordt samengesteld. (Zie de gebruikers handleiding) De voorbeeld file ziet er als volgt uit:}

```
;*************************************************************
;
; VOORBEELD PROGRAMMA VOOR DE PRACTICUM HANDLEIDING
;
; DIT PROGRAMMA TOONT EEN KNIPPEREND GETAL IN HET
; ADRESVELD VAN EEN SDK-85
;
;*************************************************************

UPDAD   EQU     0363H   ; SDK-ROUT. UPDATE ADDRESS
OUTPT   EQU     02B7H   ; SDK-ROUT. OUTPUT TO DISPLAY
DELAY   EQU     05F1H   ; SDK-ROUT. DELAY
CONST   EQU     8085H   ; CONSTANTE DIE GETOOND WORDT
TIME1   EQU     04000H  ; TIJDCONSTANTE
TIME2   EQU     02800H  ; TIJDCONSTANTE
STACK   EQU     020C2H  ; TOP OF STACK
BLANK   EQU     15H     ; CONSTANTE VOOR LEEG DISPLAY
```

```
          ORG       0C000H    ; PROGRAMMA START VOOR MICROTR.

START:    LXI       SP,STACK  ; ZET STACKPOINTER
LOOP:     LXI       D,CONST   ; LAAD PARAMETER VOOR UPDAD
          CALL      UPDAD     ; TOON IN ADRESVELD
          LXI       D,TIME1   ; LAAD EERSTE TIJDCONSTANTE
          CALL      DELAY     ; VERTRAAG ...MSEC
          LXI       H,BUFFER  ; LAAD CODE VOOR OUTPT
          MVI       A,0       ; SELECTEER ADRESVELD
          MVI       B,0       ; GEEN DECIMALE PUNT
          CALL      OUTPT     ; MAAK ADRESVELD LEEG
          LXI       D,TIME2   ; LAAD TWEEDE TIJDCONSTANTE
          CALL      DELAY     ; EN VERTRAAG WEER
          JMP       LOOP      ; DOE DIT TOT ST.JUTTEMIS

BUFFER    DB        BLANK,BLANK,BLANK,BLANK

          END                 ; EINDE PROGRAMMA
```

{Nadat deze file weggeschreven is op disk onder de naam "voorb", wordt de editor verlaten met het commando File Quit. We komen nu weer in CP/M:}

```
Z80 B>MAC voorb
CP/M ASSEMBLER - VER 1.4
C02D
000H USE FACTOR
END OF ASSEMBLY
```

{Als het programma fouten bevat krijgt u hier tevens een melding van.
De assembler heeft nu de .HEX- en de .PRN- (ofwel resp. de object- en de list-) file gecreeerd.}

```
Z80 B>!PR VOORB.PRN { print de listing op de printer }
```

{De listing kan eventueel ook op het scherm worden bekeken.}

```
Z80 B>TYPE VOORB.PRN {Het commentaar kan de max. regelbreedte
overschrijden}
```

```
                    ;***********************************************
                    ;
                    ; VOORBEELD PROGRAMMA VOOR DE PRACTICUM HANDLEI
                    ;
                    ; DIT PROGRAMMA TOONT EEN KNIPPEREND GETAL IN H
                    ; ADRESVELD VAN EEN SDK-85
                    ;
                    ;***********************************************

  0363 =            UPDAD     EQU       0363H     ; SDK-ROUT. UPDAT
  02B7 =            OUTPT     EQU       02B7H     ; SDK-ROUT. OUTPU
  05F1 =            DELAY     EQU       05F1H     ; SDK-ROUT. DELAY
  8085 =            CONST     EQU       8085H     ; CONSTANTE DIE G
```

```
4000 =          TIME1     EQU     04000H    ; TIJDCONSTANTE
2800 =          TIME2     EQU     02800H    ; TIJDCONSTANTE
20C2 =          STACK     EQU     020C2H    ; TOP OF STACK
0015 =          BLANK     EQU     15H       ; CONSTANTE VOOR

C000                      ORG     0C000H    ; PROGRAMMA START

C000 31C220     START:    LXI     SP,STACK  ; ZET STACKPOINTE
C003 118580     LOOP:     LXI     D,CONST   ; LAAD PARAMETER
C006 CD6303               CALL    UPDAD     ; TOON IN ADRESVE
C009 110040               LXI     D,TIME1   ; LAAD EERSTE TIJ
C00C CDF105               CALL    DELAY     ; VERTRAAG ...MSE
C00F 2122C0               LXI     H,BUFFER  ; LAAD CODE VOOR
C012 3E00                 MVI     A,0       ; SELECTEER ADRES
C014 0600                 MVI     B,0       ; GEEN DECIMALE P
C016 CDB702               CALL    OUTPT     ; MAAK ADRESVELD
C019 110028               LXI     D,TIME2   ; LAAD TWEEDE TIJ
C01C CDF105               CALL    DELAY     ; EN VERTRAAG WEE
C01F C303C0               JMP     LOOP      ; DOE DIT TOT ST.

C022 15151515   BUFFER    DB      BLANK,BLANK,BLANK,BLANK

C026                      END               ; EINDE PROGRAMMA
```

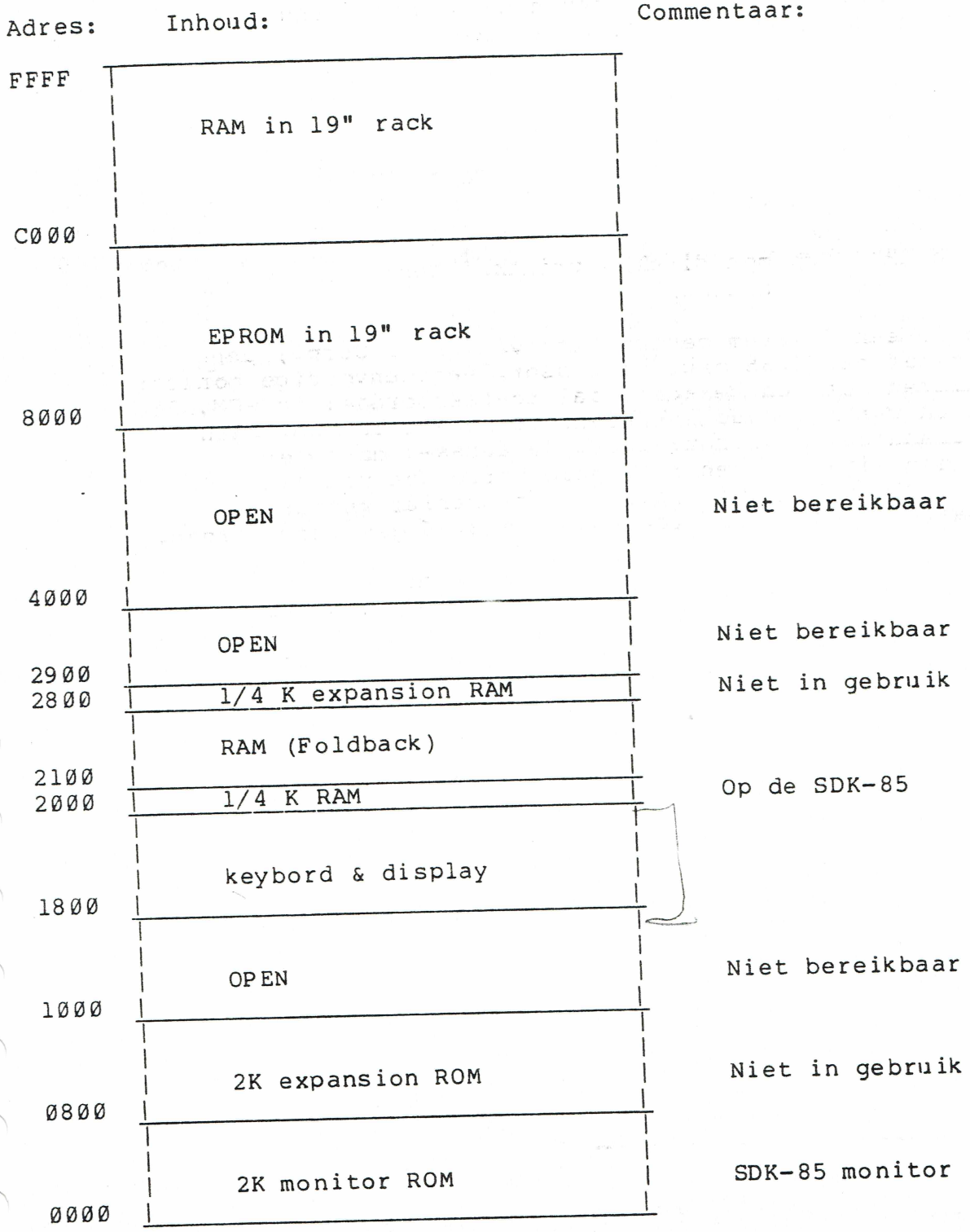
{Ook de .HEX-file kan zonodig op het scherm worden bekeken.}

```
Z80 B>TYPE VOORB.HEX

:10C0000031C220118580CD6303110040CDF105219F
:10C0100022C03E000600CDB702110028CDF105C3B5
:06C0200003C01515151503
:0000000000
```

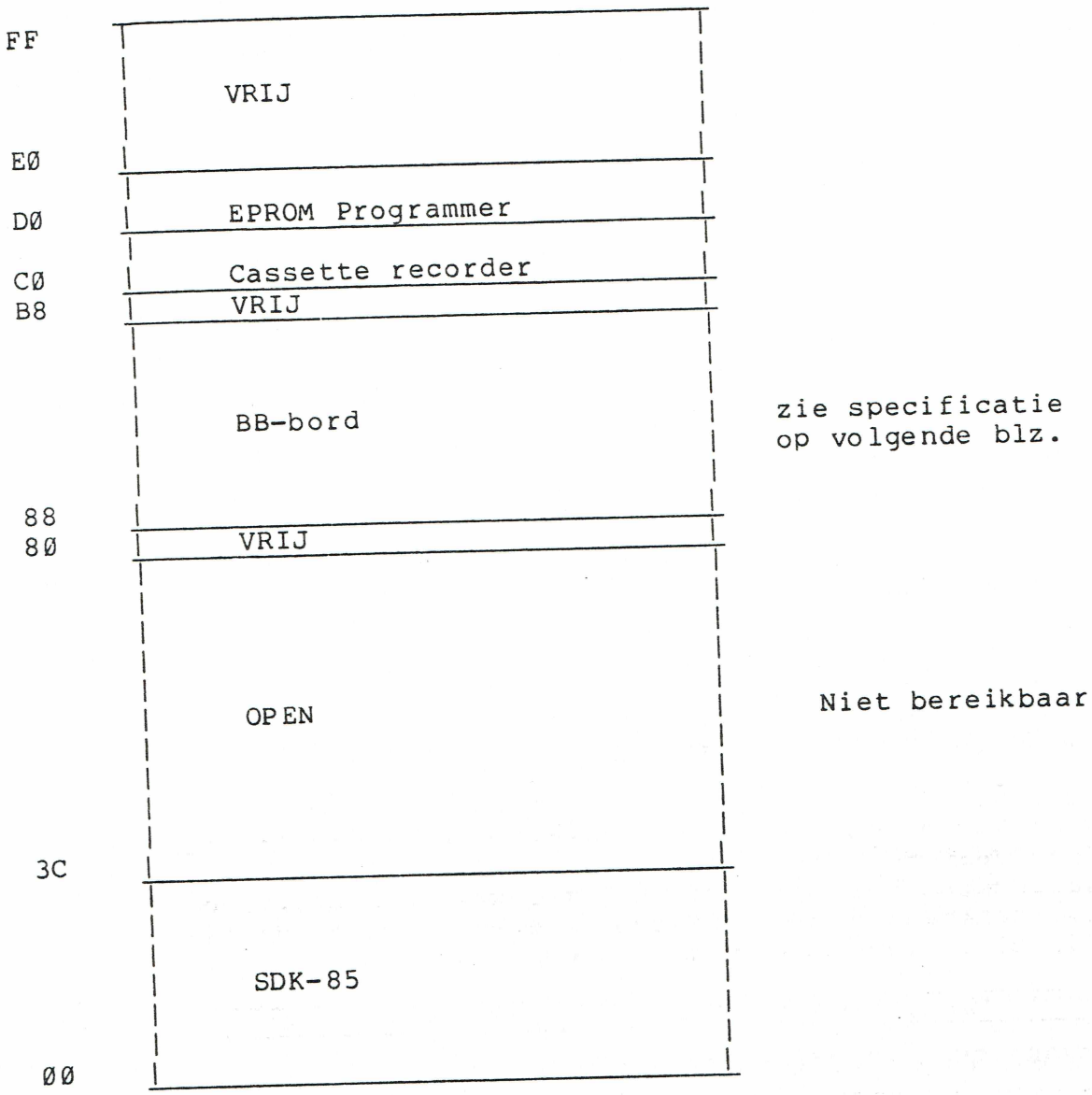
De memory- en I/O-map van de Microtrainer zien er als volgt uit:

MEMORY MAP MICROTRAINER.



De geheugen adressen 0000 t/m 7FFF zijn alleen toegankelijk op het SDK-85 bord.

I/O MAP MICROTRAINER.



De I/O adressen 00 t/m 7F zijn alleen toegankelijk op het SDK-85 bord.

5.2 De SDK-85

De SDK-85 bevat een 8085 microprocessor, een ("HEX"-) toetsenbord, een LED display met zes cijfers, een eenvoudige monitor in ROM en een klein RAM gebied. Met het toetsenbord kunnen we de monitor functies oproepen en op het LED display kunnen enkele meldingen verschijnen. Het RAM gebied wordt van adres 20C2 t/m 20FF gebruikt voor opslag van monitor gegevens, van adres 2000 tot 20C2 kan het gebied bijv. gebruikt worden als stack voor uw programma's (initieer de stackpointer op 20C2 m.b.v. LXI SP,20C2H).
Met het toetsenbord kunnen we de volgende functies laten uitvoeren:

RESET Hardware reset van de 8085; de monitor wordt gestart en er verschijnt '- 80 85' op het display. N.B! deze funktie nooit gebruiken met een cassette in de cassette recorder.

SUBST MEM <adres> NEXT (<data>) NEXT (<data>).....EXEC.
Met SUBSTitude MEMory kunt u de inhoud van een willekeurige geheugenplaats bekijken en eventueel veranderen. Na NEXT verschijnt het betreffende adres in het adres-veld van het display, en de bijbehorende inhoud in het data-veld. Als u de inhoud wilt veranderen, kunt u de nieuwe inhoud direct intoetsen. De inhoud van de geheugenplaats wordt echter pas veranderd als u NEXT of EXEC indrukt. Na NEXT verschijnen het volgende adres en data, met EXEC wordt het commando beeindigd.

EXAM REG <reg> NEXT (<data>) NEXT ..... EXEC
Met EXAMine REGisters kunt u, op dezelfde manier als bij SUBST MEM, de inhoud van registers bekijken of veranderen. De registers A t/m E krijgt u m.b.v. de overeenkomstige HEX-toetsen te zien; F toont de flags (de bits staan hex gecodeerd op het display:

| SIGN | ZERO | - | Aux Carry | - | Parity | - | Carry |)

en met de klein gedrukte letters op de toetsen krijgt u resp. te zien:

I = interrupt masker (M=1 betekent gemaskeerd):

| - | - | - | - | Int Enable | M 7.5 | M 6.5 | M 5.5 |

H = register H
L = register L
SPH = hoogste byte van de Stack Pointer
SPL = laagste byte van de Stack Pointer
PCH = hoogste byte van de Program Counter
PCL = laagste byte van de Program Counter

GO (<adres>) EXEC. Start de executie vanaf het adres dat het display toont na het indrukken van de EXEC toets. Het startadres kan daarvoor nog gewijzigd worden, door een nieuw in te toetsen. Zorg ervoor dat de stackpointer een goede waarde heeft (bijv. 20C2H of ergens in het RAM gebied boven C000H)!!

SINGLE STEP (<adres>) NEXT NEXT .... EXEC. Na het intoetsen van NEXT wordt steeds een instructie uitgevoerd. Het adresveld toont steeds het adres van de volgende uit te voeren instructie en het dataveld toont de code van de zich op dat adres bevindende instructie. Na het intoetsen van EXEC wordt geen instructie meer uitgevoerd en de single step-mode wordt verlaten. U kunt dan bijv. weer geheugen- en register-inhouden opvragen.

VECTor INTerrupt. Verzorgt een RST 7.5 interrupt. De monitor springt naar locatie 3C waar zich een jumpinstructie naar adres 20CE bevindt. De adressen 20CE t/m 20D0 kunt u zo voor eigen doeleinden gebruiken (bijv. een sprong naar een ander programma).
N.B. Voor een goede werking moeten de interrupts enabled zijn (EI instructie) en mag RST 7.5 niet gemaskeerd zijn (SIM instructie of I-register veranderen). Daar de RST 7.5 een hardware interrupt is, heeft de monitor geen kans het display te veranderen.

Via een serie-interface kaart in het 19-inch rack is het ook mogelijk om een soortgelijke SDK-monitor via een terminal te bedienen. Zie hiervoor par. 5.5.6.

Bruikbare monitor routines.

De SDK-85 monitor bevat een aantal voor het practicum bruikbare subroutines, waarvan hier een beschrijving volgt.

Adres:	Naam:	Omschrijving:
0363	UPDAD	UPDate ADdress. Toont de inhoud van het DE registerpaar in het adresveld. Alle registers en vlaggen kunnen veranderd zijn na aanroep van deze routine.
036E	UPDDT	UPDate DaTa. Toont de inhoud van het A register in het dataveld. Alle registers en vlaggen kunnen veranderd zijn na aanroep van deze routine.
02E7	RDKBD	ReaD KeyBoarD. Deze routine wacht tot een karakter is ingetoetst. De waarde van het karakter wordt in het A register gezet. De registers A, H en L en de vlaggen kunnen veranderd zijn na aanroep van deze routine. Voor een goede werking moet de interrupt 5.5 niet gemaskeerd zijn (m.b.v. SIM-instructie).
05F1	DELAY	time DELAY. De 16 bits inhoud van registerpaar DE wordt tot 0 teruggeteld. De registers A, D en E en de vlaggen kunnen veranderd zijn na aanroep van deze routine.
02B7	OUTPT	OUTPuT characters to display. Deze routine toont "karakters" op het display, gebruik makend van de parameters in de registers A,

B, H en L.
Register A = Ø -> adresveld
= 1 -> dataveld
Register B = Ø -> geen decimale punt
= 1 -> decimale punt
Registers HL = pointer naar geheugenplaats waar zich de code voor het karakter bevindt.

Kar	Code	Kar	Code	Kar.	Code
Ø	ØØ	8	Ø8	H	1Ø
1	Ø1	9	Ø9	L	11
2	Ø2	A	ØA	P	12
3	Ø3	b	ØB	I	13
4	Ø4	C	ØC	r	14
5	Ø5	d	ØD	S	Ø5
6	Ø6	E	ØE	blank	15
7	Ø7	F	ØF		

Extra Dokumentatie voor Microtrainerdisplays

(zie ook pag. 5-6 SDK-85 handleiding).

Alvorens een karakter te kunnen displayen, moeten eerst één of meer modewoorden naar de Keyboad Display Controller (Intel 8279) gestuurd worden, die zich op RAM-adres 19ØØ H bevindt.
De volgende modewoorden kunnen van belang zijn bij het praktikum (zie eventueel Intel Component Data Catalog).

Display/Keyboard modewoord:

Ø	Ø	Ø	D	D	K	K	K

DD (= display mode):
ØØ 8 displays left entry (zie verderop)
1Ø 8 displays right entry (zie verderop)
KKK (= keyboard mode):
voor het praktikum voldoet KKK = ØØØ

Write Display modewoord:

1	Ø	Ø	AI	A	A	A	A

AI = Auto Increment, indien AI = 1, wordt AAAA automatisch met 1 verhoogd bij iedere volgende lees- of schrijfoperatie;

AAAA = display RAM-plaats: ØØØØ is adresdisplay 1
ØØØ1 " " 2
ØØ1Ø " " 3
ØØ11 " " 4
Ø1ØØ " datadisplay 1
Ø1Ø1 " " 2

NB. Via het display/keyboard modewoord is bepaald dat 8 displays aangestuurd kunnen worden (via de bitjes DD). Er zijn er maar 6!
Twee displays zijn dus virtueel, namelijk AAA = Ø11Ø en AAAA = Ø111.
Dit heeft konsekwenties als AI = 1 (zie voorbeelden verderop).

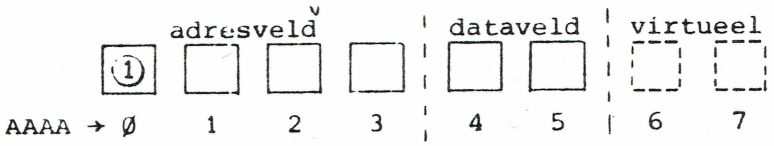
Read Display modewoord:

Ø	1	1	AI	A	A	A	A

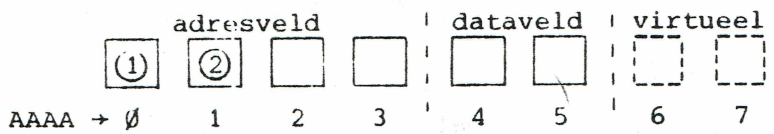
AI en AAAA als bij Write Display modewoord.

Voorbeelden:

1. Left entry auto increment.
   a. Stuur ØØH naar adres 19ØØH (kan achterwege blijven): hiermee wordt left entry bepaald.
   b. Stuur dan bijvoorbeeld 9ØH naar adres 19ØØH: hiermee selekteer je AAAA = ØH, autoincrement, write mode.
   c. Stuur nu karakter 1 naar adres 18ØØ:



   d. Stuur nu karakter 2 naar adres 18ØØ:



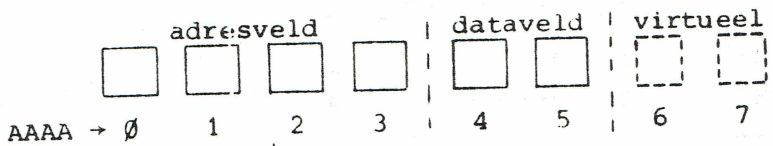
enzovoorts.
Het 9e karakter overschrijft het 1e karakter in het 1e adresdisplay, het 10e het het 2e enz.

N.B. Het 7e en 8e karakter worden dus naar een virtueel display gestuurd en zijn dus niet zichtbaar. Als het 7e karakter het 1e moet overschrijven, moet de software dus na 6 schrijfoperaties eerst weer 9Ø naar adres 19ØØH sturen.

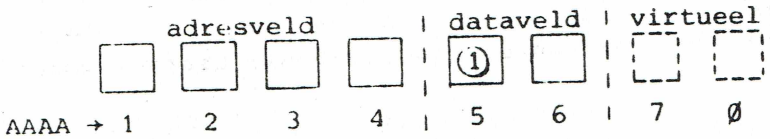
2. Right entry auto increment.

a. Stuur 1ØH naar adres 19ØØH; hiermee wordt right entry bepaald.

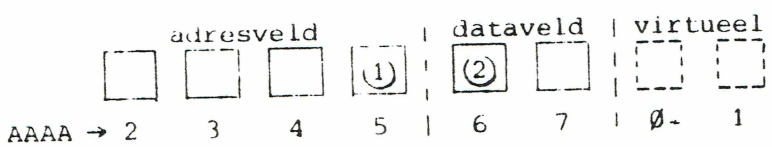
b. Stuur dan bijvoorbeeld 95H naar adres 19ØØH: hiermee selekteer je het tweede data display, auto increment, write mode. Nu is startkonfiguratie van de displays:



c. Stuur nu karakter 1 naar adres 18ØØH:

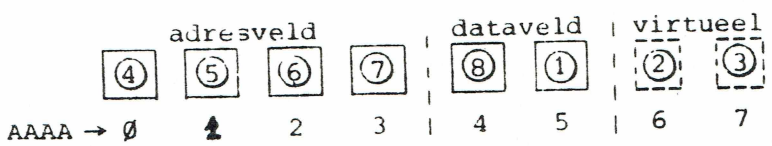


d. Stuur karakter 2 naar adres 18ØØH:

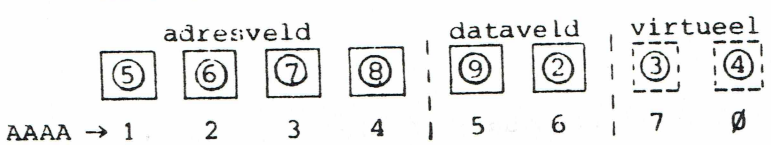


enzovoorts.

Na 8 karakters is de situatie:



Na 9 karakters is de situatie:



NB. Let erop dat in de right entry mode AAAA niet meer éénduidig verbonden is met één display!

5.3 BB-bord.

Het BB-bord is een print met een groot aantal I/O mogelijkheden, zoals serie I/O, parallelle I/O, timing, A/D en D/A converters, luidspreker, motor, toerenteller, lichtopnemer met lampje, schakelaars met L.E.D.'s, alsmede een experimenteerbordje. De I/O map vindt u in par. 5.1. In deze paragraaf komen de diverse onderdelen een voor een aan de orde; ze is summier, maar met behulp van het schema van het BB-bord (Appendix H) zullen de nodig zaken duidelijk worden.
De EPROM op het bord kunt u niet gebruiken.

Aangeraden wordt bij de volgende paragrafen de schema's te raadplegen.

5.3.1 PPI 2655.

De "Programmable Peripheral Interface" 2655 is een chip die enerzijds met de databus van een microprocessor verbonden is en anderzijds drie uitgangspoorten van 8 bits parallel bevat, die d.m.v. een aantal instructies op diverse manieren gebruikt kunnen worden voor I/O toepassingen. Deze PPI lijkt veel op de Intel 8255A PPI(is daar zelfs pin-compatible mee), maar is niet software compatible met deze chip. Het voornaamste verschil is de timer, die zich in de 2655 bevindt, waardoor pin 36 een andere betekenis heeft gekregen en de instelling van het mode-woord anders is (door middel van het modewoord wordt de 2655 verteld wat deze moet doen).

De adressen van PA, PB, PC en het modewoord zijn:

PA: 88H
PB: 89H
PC: 8AH
modewoord: 8BH

Voor een goede werking kan de 2655 op het BB-bord bijvoorbeeld ingesteld zijn met mode-woord 80H. (naar poort 8BH is dan 84H gestuurd) Poort A is dan in bidirectionele mode; poort A is verbonden met het proefbord (zie 5.3.6) en met de 15-polige connector (zie 5.3.7).
Poort B is in "static I/O-mode"; via poort B zijn aangesloten: (In 5.3.5 wordt uitgelegd hoe u de switches SW0 t/m SW5 kunt besturen)

- B7 niet verbonden
- B6(=C0) INPUT LDR-circuit (lichtgevoelige weerstand) of (SW0) RPM-circuit ("snelheidsmeter" van het motortje).
- B5 INPUT PCI klok of (SW1) de halve systeemklok (1,5 MHz).
- B4 OUTPUT reset voor LDR- en RPM-flipflop.
- B3 INPUT altijd een "1".
- B2 niet verbonden
- B1 niet verbonden
- B0 OUTPUT lampje bij de LDR

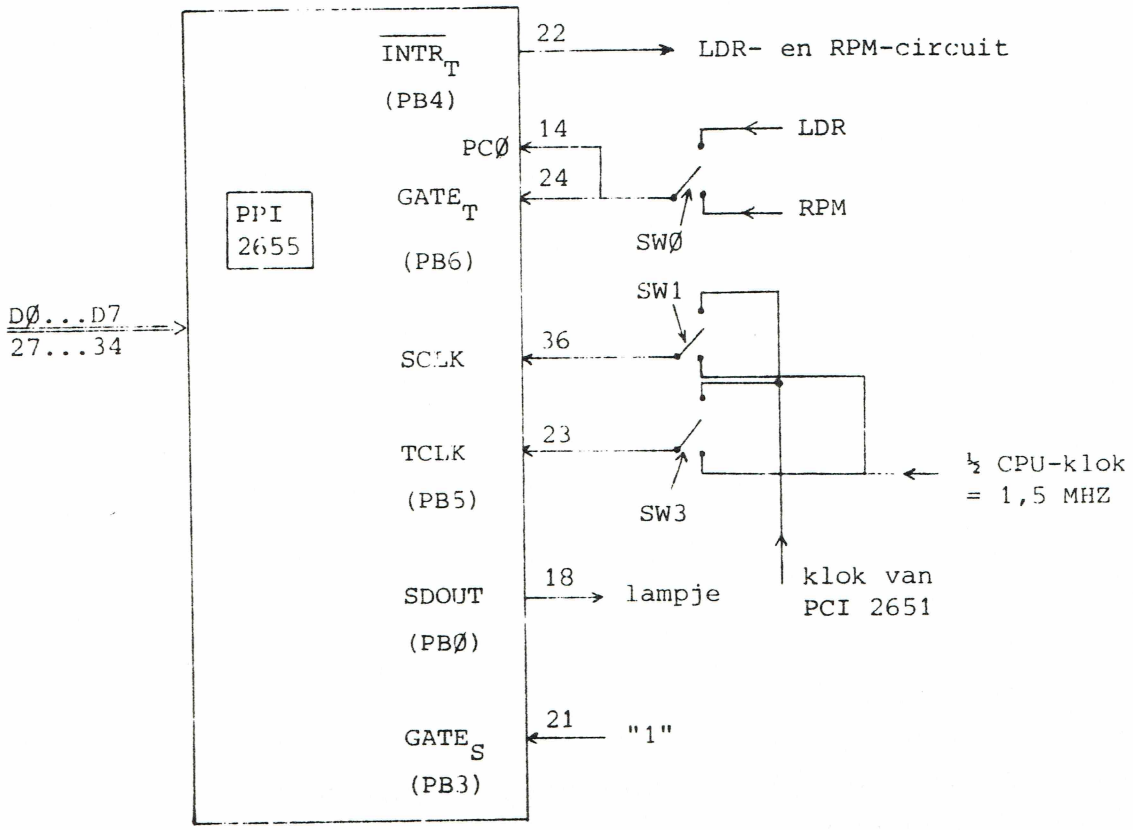
Poort C is ook in "static I/O-mode"; via poort C zijn aangesloten:

- C7 OUTPUT (met vertraagde positieve flank) proefbord en 15-polige connector (zie 5.3.6 en 5.3.7).
- C6 INPUT proefbord en 15-polige connector (zie 5.3.6 en 5.3.7)
- C5 OUTPUT idem
- C4 INPUT idem
- C3 niet verbonden
- C2 INPUT PCI klok.
- C1 INPUT busy-lijn van A/D converter (laag actief).
- C0(=B6) INPUT LDR-circuit of (SW0) RPM-circuit ("snelheidsmeter" van het motortje).

Bij gebruik van het LDR-circuit zal na een reset van de LDR flipflop het LDR-circuit een "1" afgeven en na verloop van tijd weer een "0". De tijd dat het LDR-circuit een "1" geeft is een maat voor de lichtintensiteit, die de LDR op dat moment opvangt.

De PPI 2655 heeft een eigenaardigheid, die de programmeur parten kan spelen. Wil men in "static I/O" iets uit poort A, B of C lezen, dan dient men eerst een "1" naar elk inputbit van de betreffende poort te schrijven. Wanneer dat vergeten wordt, dan wordt altijd een "0" gelezen, ongeacht de werkelijke waarde op de poort.

Extra documentatie voor de PPI 2655



DØ...D7 = databits

$\overline{INTR_T}$ = PB4 = Interrupt Request van de timer van de PPI 2655; als de interrupt enabled is (via $INTE_T$), gaat $\overline{INTR_T}$ laag als de teller nul bereikt; deze gaat hoog als een nieuwe waarde geladen wordt in één van beide bytes van de teller (LSB of MSB) óf wanneer de timer mode geprogrammeerd wordt.

$GATE_T$ = PCØ = PB6 = Kontrolebit voor de teller; als $GATE_T$ = "1" is de teller enabled, is $GATE_T$ "Ø" dan is de teller disabled (respectievelijk aan- en afgeschakeld).

TCLK = PB5 = als de teller telt, verlaagt de timer de inhoud van de teller met 1 op elke neergaande flank van dit kloksignaal.

SCLK = kloksignaal voor het versturen (en ontvangen) van data in seriële vorm. Op elke opgaande flank wordt 1 bit verstuurd (of ontvangen).

SDOUT = PBØ = seriële datauitgang.

$GATE_S$ = PB3 = kontrolebit voor seriële dataoverdracht: in dit geval altijd "1" en als input gebruikt, zodat seriële datauitvoer via SDOUT altijd mogelijk is.

SWØ, SW1, SW3 - schakelaars die met het programma in de juiste stand geplaatst moeten worden.

In het onderstaande worden de basishandelingen beschreven voor gebruik van de timer en/of seriële mode.

Programmeren van de timer/seriële mode.

Ervan uitgaande dat poort A van de PPI 2655 in de bidirektionele mode geprogrammeerd wordt zijn er de volgende mogelijkheden:

A4H naar poort 8BH (moderegister): dan is poort B in seriële output mode gedefiniëerd.

B4H naar poort 8BH (moderegister): dan is poort B in seriële output + timer mode gedefiniëerd.

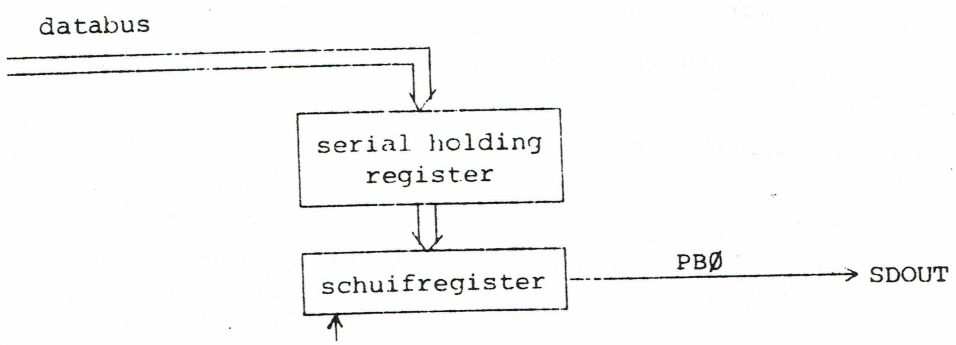
Selektrie van timer of seriële funkties.

Bij gebruik van de seriële output + timer mode (B4H naar poort 8BH) kan data aangeboden worden aan / of gelezen worden uit het serial holding register respectievelijk het timer register.

Van groot belang is, dat bij het programmeren van deze mode onder andere de volgende neveneffekten optreden.

- de databus is in eerste instantie verbonden met het serial holding register;
- $\overline{INTR}_T$ wordt "1";
- bij eerstvolgend gebruik van de timer zal eerst het least significant byte (LSB) van het timerregister aan de beurt zijn, dan het most significant byte (MSB). Ter toelichting: de teller heeft een 2-bytes register (dus 16 bits). (Zie voor gebruik van de timer verderop).

Na het versturen van B4H naar poort 8BH kan dus direkt seriëel data verstuurd worden via SDOUT (= PBØ). Schematisch is de situatie als volgt:



Data komt in het serial holding register door deze naar poort B van de PPI te sturen. Het byte in dit register wordt doorgeschreven naar het schuifregister.
Bij elke opgaande flank van SLCK wordt 1 bit van het byte verstuurd. Zodra 8 bits (= 1 byte) aldus verstuurd zijn, wordt de nieuwe waarde uit het serial holding register in het schuifregister geplaatst en seriëel verzonden. Indien geen nieuwe waarde in het serial holding register is geplaatst wordt de oude waarde opnieuw verstuurd.
Wil men toegang krijgen tot het timer register, dan moet het zogenaamde Serial/Timer Status and Control Register aangepast worden (STSR).
Het STSR is bereikbaar via het moderegister: na B4H naar het moderegister gestuurd te hebben kan STSR bereikt worden door het volgende byte naar het moderegister te sturen:

Ø	Ø	1	1	X	X	X	Y

geeft toegang tot STSR

XXX = ØØØ = bit Ø = STØ
XXX = ØØ1 = bit 1 = ST1
enz. tot en met:
XXX = 111 = bit 7 = ST7
y = Ø = reset (bit XXX van STSR naar "Ø")
y = 1 = set (bit XXX van STSR naar "1")

De definities van de bits STØ t/m ST7 is als volgt:
STØ = Access Control (AC); indien STØ = "1" is het Timer register geselekteerd, indien STØ = "Ø" is het serial holding register geselekteerd.
ST1 = $INTR_S$ (let op: PB1 = $\overline{INTR_S}$!)
Interrupt request van seriële mode: is verder niet van belang, want PB1 is niet aangesloten.
ST2 = IBF_S/OBE_S (let op: PB2 = $\overline{IBF_S}$ of $\overline{OBE_S}$)
Input buffer full/Output buffer empty van seriële mode; niet van belang, want PB2 is niet aangesloten.
ST3 = $INTE_S$: interrupt seriële mode enabled.
Als $INTE_S$ = "1" (enabled) wordt $INTR_S$ ook 1, waarmee in principe interrupt te plegen valt.
ST4 = $INTR_T$: interrupt request van de timer.
(let op: PB4 = $\overline{INTR_T}$)
NB; de waarde van ST4 kan pas vrijelijk via een programma gewijzigd worden, nadat $INTE_T$ (=ST6) op "1" is gezet! Zie voor verdere eigenschappen onder

de eerste figuur van deze extra informatie.

ST5 = niet gebruikt, altijd "Ø".

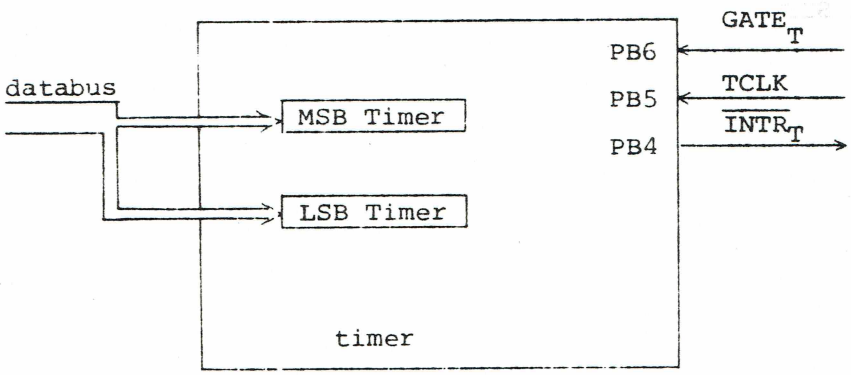
ST6 = $INTE_T$: interrupt enable van de timer.
Zie voor funktie hierboven bij ST4.
ST6 = "1" : interrupt enabled.
ST6 = "Ø" : interrupt disabled.

ST7 = Upper/Lower select : als ST7 = "Ø" wordt LSB van het timeregister geselekteerd, als ST7 = "1" het MSB (LSB zijn de bits Ø t/m 7, MSB zijn de bits 8 t/m 15).

De preciese gevolgen van het programmeren van poort B van de PPI in de serial output + timer mode zijn:
- STØ, ST4, ST7 worden gereset op "Ø".
- ST2 wordt gezet op "1".
- ST3 en ST6 worden niet gewijzigd.

Het tellen van de timer

De situatie van de timer is als volgt:



Zie voor definiëring van de verschillende symbolen het voorgaande.
Na via STØ de timer te hebben geselekteerd, kan het 16-bits timer register via poort B gelezen en beschreven worden. Via ST7 wordt bepaald of LSB of MSB aan de beurt is: na elke lees- of schrijfoperatie verandert ST7 automatisch van waarde, zodat beide bytes (LSB en MSB) toegankelijk zijn door achtereenvolgende lees- en/of schrijfoperaties.
De timer begint te tellen, zodra MSB is geladen.
De timer stopt met tellen zodra:
- de timer mode wordt geprogrammeerd òf
- $GATE_T$ "Ø" wordt.

Als de timer telt, en de interrupt is enabled (d.w.z. $INTE_T$ = "1"), wordt $\overline{INTR}_T$ laag zodra de teller tot nul heeft afgeteld. De teller gaat dan door met tellen! $\overline{INTR}_T$ blijft laag tot LSB of MSB gewijzigd wordt door het programma òf de timer mode wordt geherprogrammeerd: dan wordt $\overline{INTR}_T$ weer "1".
Zoals vermeld telt de timer 1 af van de inhoud van de teller op elke neergaande flank van TCLK (PB5).

5.3.2 PCI 2651

De "Programmable Communication Interface" 2651 is een chip die seriele communicatie kan verzorgen, d.w.z. dat data die parallel over de databus worden aangeboden, door de PCI in een seriele datastroom wordt omgezet en met een van te voren ingestelde baudrate aan de uitgangspen zal verschijnen; andersom kan ook een in serie aangeboden datastroom worden omgezet naar parallelle data en op de databus worden aangeboden. De 2651 lijkt veel op de Intel 8251A (en is daar in de meeste toepassingen pin-compatible mee). Ook hier is de software zijde echter niet compatible. Het voornaamste verschil is hier, dat de 2651 een interne baudrate generator bevat, waarmee in software de baudrate ingesteld kan worden. De inkomende (via een opto-coupler) en uitgaande seriele datalijnen zijn verbonden met twee van de 5-polige DIN connectoren (zie 5.3.7).

De Poorten van de PCI 2651.

a. Poortadressen

90H : bij schrijfkommando (OUT) → Transmit Data Holding Register.
bij leeskomando (IN) → Receive Data Holding Register.

91H : bij schrijfkommando (OUT) →
1e maal geeft toegang tot SYN1 Register,
2e " " " " SYN2 "
3e " " " " DLE "
4e " " " " SYN1 "
enzovoorts.
bij leeskommando (IN) → Status Register.

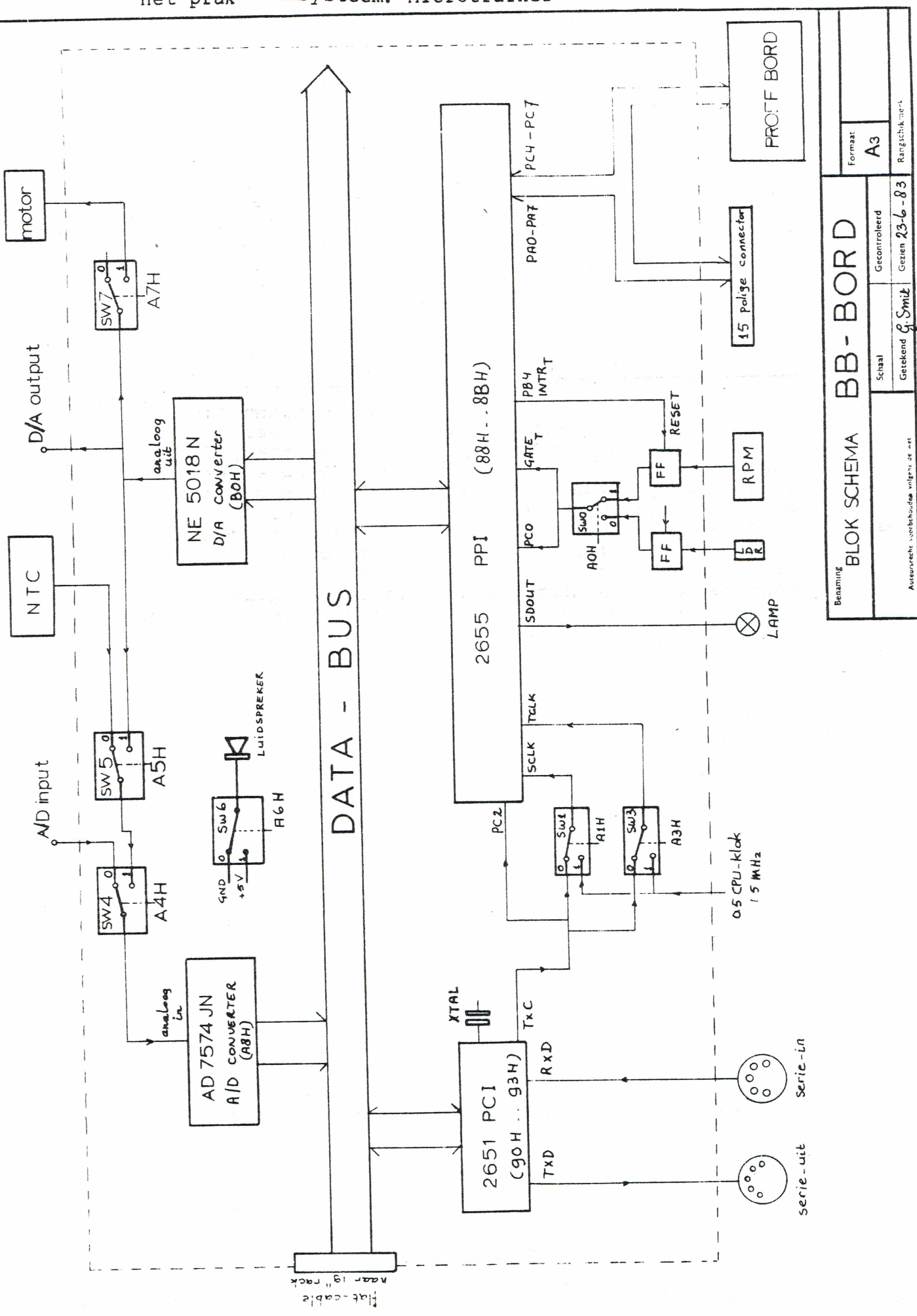
92H : Mode Register 1 óf 2;
1e maal lezen óf schrijven geeft toegang tot Mode Register 1
2e " " " " " " " Mode Register 2
3e " " " " " " " Mode Register 1
enzovoorts.

93H : Command Register.

N.B. Bij een RESET van het gehele systeem én bij elke leesoperatie (IN) van het Command Register gaan de interne pointers van de poorten 91H en 92H terug naar het SYN1 Register, respectievelijk Mode Register 1.

b. Betekenis van de registers.

- Transmitter Data Holding Register (THR) = het register waarin de CPU een karakter kan plaatsen, dat door de PCI seriëel verstuurd moet worden (poort 90H).
- Receiver Data Holding Register (RHR) = het register waarin de PCI de seriëel ontvangen karakters plaatst, die vervolgens door de CPU uitgelezen kunnen worden (poort 90H).
- SYN1 Register = een register dat alleen in de synchrone mode (zie punt 3.2) gebruikt wordt en waarin de CPU een karakter (5-8 bits) plaatst dat gebruikt wordt ter synchronisatie van verzending en ontvangst van seriëel


motor
D/A output
SW7
A7H
analoog uit
NE 5018 N
D/A converter
(B0H)
NTC
A/D input
SW5
A5H
SW4
A4H
SW6
A6H
LUIDSPREKER
GND
+5V
analoog in
AD 7574 JN
A/D CONVERTER
(A8H)
DATA - BUS
2655 PPI (88H..8BH)
PC4-PC7
PA0-PA7
PB4 INTR
GATE T
PC0
SDOUT
TCLK
SCLK
PC2
SW0
A0H
FF
RESET
RPM
DR
LAMP
SW1
A1H
SW3
A3H
05 CPU-klok
1.5 MHz
PROTF BORD
15 polige connector
XTAL
TxC
RxD
TxD
2651 PCI
(90H .. 93H)
serie-in
serie-uit
flat-cable
naar 19" rack
Benaming
BLOK SCHEMA
BB-BORD
Schaal
Getekend G. Smit
Gecontroleerd
Gezien 23-6-83
Formaat
A3
Rangschikking
Auteursrecht voorbehouden volgens de wet

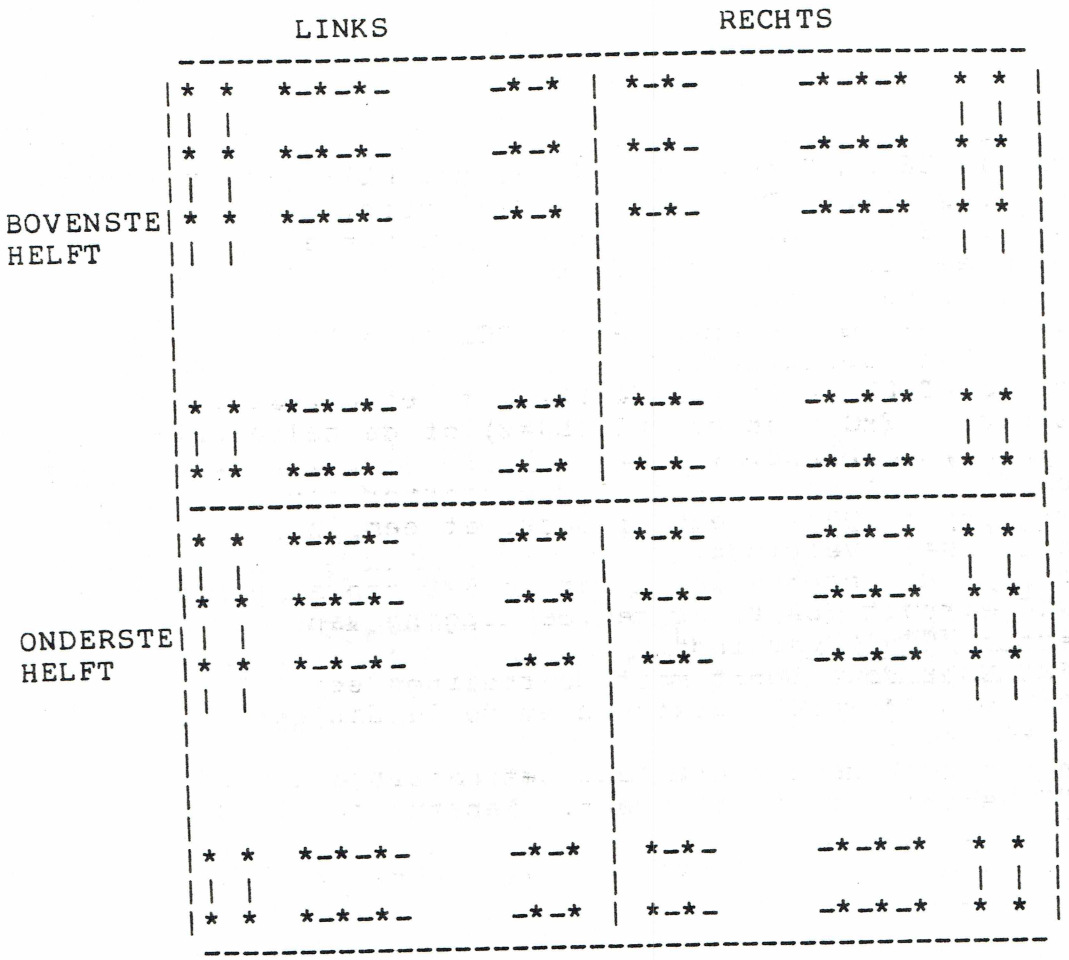
SWØ (AØH) Switch die B6 en CØ van de PPI met het LDR-circuit (DØ=Ø) of met het RPM-circuit (DØ=1) verbindt.
SW1 (A1H) Switch die SCLK van de PPI met het uitgaande kloksignaal (TxC) van de PCI (DØ=Ø) of de halve CPU-klok (DØ=1) verbindt.
SW2 (A2H) Switch die de PGC tezamen met de PCI (DØ=Ø) of met de de PPI (DØ=1) selecteert.
SW3 (A3H) Switch die TCLK (=B5) van de PPI met het uitgaande kloksignaal (TxC) van de PCI (DØ=Ø) of de halve CPU-klok (DØ=1) verbindt.
SW4 (A4H) Switch die de ingang van de A/D converter aan de externe input (DØ=Ø) legt of deze met een intern signaal (DØ=1) verbindt.
SW5 (A5H) Switch die de interne input van de A/D converter met het NTC-circuit (DØ=Ø) of met de uitgang van de D/A converter (DØ=1) verbindt.
luidspreker (A6H) Naar deze poort moet beurtelings een "Ø" en een "1" via DØ worden gestuurd om de luidspreker te laten klinken.
motor (A7H) Een "1" naar deze poort laat het motortje lopen, een "Ø" stopt het motorje weer. (Aansturing via DØ)

Dus door ØØH naar AØH te sturen wordt B6 van de PPI met het LDR-circuit verbonden.

5.3.6 Proefbord.

Op het witte proefbord kunt u zelf circuits aanleggen en uittesten. Van veel soorten componenten (I.C.'s, weerstanden, etc.) kunt u de aansluitpennen zo in de gaatjes van het proefbord steken. De gaatjes zijn op de volgende manier met elkaar verbonden: Van de twee linker en de twee rechter kolommen zijn de gaatjes in de bovenste en de gaatjes in de onderste helft van het bord met elkaar verbonden. In de overige kolommen zijn de gaatjes steeds per 5 horizontaal met elkaar verbonden.

Proefbord:



Verder kunt u nog gebruik maken van een aantal, op het BB-bord aanwezige, signalen. Deze zijn verbonden met de bovenste 16 rijen van 5 gaatjes links en rechts van het midden. (A = poort A van de PPI; C = poort C van de PPI; S = schakelaar)

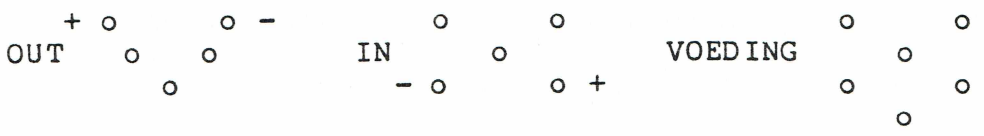
C4	C6
C7	C5
AØ	A7
A1	A6
A2	A5
A3	A4
S7	LED7
:	:
:	:
SØ	LEDØ
GND	GND
+5V	+5V

5.3.7 Connectoren.

Het BB-bord bevat 5 connectoren:

- 64-polige flatcable connector, waar de adres-, data- en control-signalen door gaan naar de interface kaart in het 19"-rack. Hiermee wordt de verbinding met het SDK-bord en het bussysteem gerealiseerd.

- drie 5-polige DIN connectoren voor resp. voedingsspanning, seriele ingang en seriele uitgang. Configuratie (vooraanzicht):



- 15-polige Cannon connector met de volgende pin-configuratie:

pen1= A4 A5 A6 A7 C5 C7 C6 C4 =pen8

pen9= A3 A2 A1 AØ NV NV NV =pen15

NV = niet verbonden.
Merk op dat deze signalen zich ook op het proefbord bevinden.

5.4 Het 19" rack.

In deze paragraaf wordt iets verteld over wat zich in het 19" rack bevindt, maar het is niet de bedoeling dat u de zaak open maakt en de kaarten eruit haalt!
Het 19" rack is een rack met een aantal connectoren als 'backplane' die pensgewijs met elkaar zijn doorverbonden. In het rack zijn een aantal printkaarten gestoken, waarvan het dus in principe niet uitmaakt waar ze zitten.
Voor het practicum zijn de volgende printkaarten beschikbaar:

- RAM kaart met maximaal 16 Kbytes statische RAM. Deze kaart wordt als geheugen gebruikt om uw programma's in te laden en om eventuele data in op te slaan.

- EPROM kaart met maximaal 16 Kbytes EPROM (Intel 2716) In deze EPROMs bevindt zich de programmatuur om de cassette recorder, de EPROM programmer, enz. aan te sturen (zie paragraaf 5.5). De kaart bevat 8 IC-voetjes (een 2716 bevat 2K x 8 bits), die niet allemaal worden gebruikt; een ervan is verbonden met het frontpaneel, zodat u daarin uw eigen geprogrammeerde EPROM kunt steken. Let erop dat het startadres van uw programma overeenkomt met het "adres" van dit voetje.

- Cassette recorder interface kaart.

- EPROM programmer. Deze kaart bevat de timing en sturing voor het programmeren van EPROMs m.b.v. het daarvoor op de EPROM kaart aanwezige programma.

- SDK-85 interface kaart. Deze kaart zorgt voor de busaanpassing en -buffering van de signalen van de SDK-85, zodat de kaarten in het 19" rack gestuurd kunnen worden door de 8085 op de SDK-85.

- BB-bord interface kaart. Deze kaart zorgt voor de busaanpassing en -buffering van de signalen van het BB-bord, zodat dit via het 19" rack ook door de 8085 aangestuurd kan worden.

In principe kunnen zich ook de volgende kaarten in het 19-rack bevinden:

- Serie interface kaart. Een kaart met twee Intel 8251 USART's. De twee 25-polige connectoren zijn als "modem-zijde" aangesloten. De baudrate kan voor elk van de aansluitingen tussen 110 en 9600 worden ingesteld.

5.5 Aanwezige software.

In het 19" rack staat op adres 8000H de programmatuur in PROM voor de aansturing van de cassette recorder: de CASS-monitor.

Mogelijkheden van de CASS-monitor

De mogelijkheden zijn vrijwel gelijk aan een gewone cassetterecorder. Men kan er mee opnemen (store), weergeven (load), vooruitspoelen (forward), terugspoelen (rewind) en stoppen (stop). Naast deze "gewone" functies zijn er nog de functies wissen van de band (erase) en tellen van het aantal files op de band (count).

De cassettes.

De cassettes zijn aan twee kanten bespeelbaar. Ze zijn te beschermen tegen abusievelijk wissen door het betreffende zwarte dopje te verwijderen.
Het verdient aanbeveling om de cassette pas na het opstarten van de CASS-monitor in de cassetterecorder te doen en deze voor het beeindigen van de CASS-monitor weer te verwijderen. De CASS-monitor zal u helpen dit goede gebruik aan te leren. Mocht u er geen behoefte aan hebben om deze regels te volgen dan loopt u het risico informatie te verliezen.

Opstarten van de CASS-monitor

Zet eerst de stackpointer op adres 20C2H:

"EXAM REG" "SPH" 2 0 "NEXT" C 2 "EXEC"

Onder de SDK-monitor wordt de CASS-monitor opgestart door het volgende in te typen.

"GO" 8 0 0 0 "EXEC"

Vervolgens begint er in het display 'CASS' te knipperen. Door er nu een cassette in te doen houdt het knipperen op en is de CASS-monitor opgestart.

De toetsen en hun functies onder de CASS-monitor

Bijzondere toetsen.

"RESET"	"RESET" zorgt voor een hardware reset van de SDK. Men wordt geadviseerd deze toets NOOIT te gebruiken onder de CASS-monitor om beschadiging van de informatie op de band te voorkomen.
"VECT INTR"	"VECT INTR" is als stop gedefinieerd. Stop stopt onmiddelijk de actie en keert terug naar de CASS-monitor. Het neveneffect van deze toets is per commando beschreven.
"EXEC"	"EXEC" geldt als afsluiter van adressen. Deze toets vertelt de CASS-monitor dat er een adres is ingevoerd.
Overigen	De overige toetsen hebben geen betekenis.

De CASS-monitor interpreteert deze toetsen als NOP-instructies.

Cijfertoetsen.

"1"	Rewind
"2"	Load
"3"	Forward
"7"	Count
"A"	Erase
"B"	Store
"E"	End
Overigen	No operation

De bovenstaande functies kunnen altijd gestart worden als er 'CASS' in het display staat. Gaat er iets fout bij een commando dan verschijnt er knipperend ' Err' in het display met daarachter een getal. In de aan het eind staande tabel van foutmeldingen kunt u opzoeken wat de foutmelding inhoudt.

Count (toets 7)

Count telt het aantal files op de band en laat dit in het display zien.

DISPLAY	Wat er gebeurt en wat u moet doen
' E' '7 '	Eerst wordt de band teruggespoeld, daarna wordt de band gelezen om het aantal files te tellen.
' ' 'XY'	Terwijl de band wordt teruggespoeld staat er in het display het aantal files (XY) in decimale vorm. Vervolgens wordt de band een klein stukje vooruitgespoeld om het eerste stukje over te slaan.

Stop heeft geen nadelige effecten voor de informatie op de band.

End (toets E)

End beeindigt de CASS-monitor en keert terug naar de SDK-monitor.

DISPLAY	Wat er gebeurt en wat u moet doen
' E' 'E ' en 'CASS' 'E ' afwisselend.	De CASS-monitor wacht tot u de cassette verwijderd hebt.
'- 8Ø' '85'	U bent weer bij de SDK-monitor terug.

Stop heeft geen nadelige effecten voor de informatie op de band.

Erase (toets A)

Erase wist zoveel van de band, zodat deze door de CASS-monitor als leeg beschouwd wordt.

DISPLAY	Wat er gebeurt en wat u moet doen
'ErA ' ' '	De CASS-monitor vraagt om een bevestiging. "1" betekent ja en elke andere toets nee.
Na "1":	
' E' 'A '	De band wordt teruggespoeld, vervolgens gewist, daarna weer teruggespoeld en tenslotte een klein stukje vooruit.

Stop TIJDENS het wissen laat de band in een ongedefinieerde toestand, verder heeft stop VOOR het wissen tot gevolg dat de band NIET gewist wordt, stop NA het wissen heeft geen enkele invloed, want de band is al leeg.

Forward (toets 3)

Forward spoelt de band naar het eind.

DISPLAY	Wat gebeurt er en wat u moet doen
' E' '3 '	De band wordt vooruitgespoeld en aan het eind een stukje terug.

Stop heeft geen nadelige effecten voor de informatie op de band.

Load (toets 2)

Load laadt een file van cassette.

DISPLAY	Wat gebeurt er en wat u moet
'FILE' ' '	Load wacht op het nummer van het file, dat geladen moet worden. U tikt het nummer in in twee cijfers (decimaal, file 1 als Ø1).
' ' 'OF'	Load wacht nu op de offset voor het laden. Normaal is Ø, zodat Ø "EXEC" (d.i. de afsluiter) voldoet. Een eventuele andere offset moet zelf berekend worden. Dit moet hexadecimaal modulo 1ØØØØ gebeuren. 1 is ØØØ1 en -1 is FFFF (d.i. 1ØØØØ - 1).
' E' '2 '	Eerst wordt de band teruggespoeld, vervolgens wordt het file opgezocht en ingelezen. Tenslotte wordt de band teruggespoeld en weer een klein stukje vooruit.

Stop heeft geen nadelige effecten voor de informatie op de band. Alleen als er na het inlezen op stop gedrukt wordt staat het gevraagde file compleet in het geheugen.

No operation (toetsen Ø, 4, 5, 6, 8, 9, C, D, F, "SINGLE STEP",

"GO", "SUBST MEM", "EXAM REG", "NEXT" en "EXEC")
Het indrukken van een van deze toetsen heeft geen effect. U zult het display even zien doven.

DISPLAY	Wat er gebeurt en wat u moet doen
' ' ' '	Niets!

Stop heeft geen nadelige gevolgen voor de informatie op de band.

Rewind (toets 1)
Rewind spoelt de band terug.

DISPLAY	Wat er gebeurt en wat u moet doen
' E' '1 '	De band wordt teruggespoeld en vervolgens een klein stukje vooruit.

Stop heeft geen nadelige effecten voor de informatie op de band.

Store (toets B)
Store schrijft de inhoud van een stuk geheugen op de band.

DISPLAY	Wat er gebeurt en wat u moet doen
' ' 'bE'	Store wacht op het adres van de eerste byte (af te sluiten door "EXEC").
' ' 'Ed'	Store wacht op het adres van de laatste byte (af te sluiten door "EXEC").
' E' 'b '	Eerst wordt de band terug-gespoeld, vervolgens vooruit om het eind van de informatie op de band te vinden. Daarna wordt een stukje teruggespoeld en vervolgens wordt de inhoud van het opgegeven geheugengedeelte op band geschreven.
'FILE' 'XY'	Het opgegeven gedeelte is als het XY-de file op de band geschreven. Door bij load dit nummer op te geven kan het gedeelte weer in het geheugen geschreven worden.

Stop VOOR het schrijven heeft geen gevolgen voor de informatie op de band. Stop TIJDENS het schrijven laat de band in een ongedefinieerde toestand. Stop NA het schrijven heeft geen nadelige effecten voor de informatie op de band, het opgegeven geheugengedeelte staat er goed op.

BELANGRIJK
Na goed notie van de foutmelding genomen te hebben kunt u door op "VECT INTR" te drukken weer terug naar

de CASS-monitor.

Tabel van foutmeldingen.

Foutnummer	Verklaring
1	De cassette is er te vroeg in gedaan, dit kan informatie verlies tot gevolg hebben.
2	Frame error. (Stop of start-bits ontbreken). Probeer nog een keer te lezen. Gaat dit weer fout, waarschuw dan een student-assistent.
3	Er is geen cassette aanwezig, stop er een in.
4	De cassette mag niet beschreven worden.
5	Het begin of het eind van de cassette is bereikt. De huidige actie is mislukt en de informatie op de band is mogelijk niet in orde. (Stuk van een file ontbreekt o.i.d).
6	De rest van de band is leeg. (Als dit bij load optreedt, dan wil dat zeggen dat het gevraagde file niet bestaat).
7	Checksum error. Probeer nog een keer te lezen. Gaat dit weer fout, waarschuw dan een student-assistent.
8	Format error. (Informatie op de band staat niet in Intel-formaat). Probeer nog een keer te lezen. Gaat dit weer fout, waarschuw dan een student-assistent.
9	Overrun error. (Er zijn een of meer bytes verloren gegaan tijdens het lezen). Probeer nog een keer te lezen. Gaat dit weer fout, waarschuw dan een student-assistent.

HEXADECIMAL CONVERSION TABLE

HEX	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	00	000
0	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	0	0
1	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	256	4096
2	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	512	8192
3	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	768	12288
4	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	1024	16384
5	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	1280	20480
6	96	97	98	99	100	101	102	103	104	105	106	107	108	109	110	111	1536	24576
7	112	113	114	115	116	117	118	119	120	121	122	123	124	125	126	127	1792	28672
8	128	129	130	131	132	133	134	135	136	137	138	139	140	141	142	143	2048	32768
9	144	145	146	147	148	149	150	151	152	153	154	155	156	157	158	159	2304	36864
A	160	161	162	163	164	165	166	167	168	169	170	171	172	173	174	175	2560	40960
B	176	177	178	179	180	181	182	183	184	185	186	187	188	189	190	191	2816	45056
C	192	193	194	195	196	197	198	199	200	201	202	203	204	205	206	207	3072	49152
D	208	209	210	211	212	213	214	215	216	217	218	219	220	221	222	223	3328	53248
E	224	225	226	227	228	229	230	231	232	233	234	235	236	237	238	239	3584	57344
F	240	241	242	243	244	245	246	247	248	249	250	251	252	253	254	255	3840	61440

5		4		3		2		1		0	
HEX	DEC	HEX	DEC	HEX	DEC	HEX	DEC	HEX	DEC	HEX	DEC
0	0	0	0	0	0	0	0	0	0	0	0
1	1,048,576	1	65,536	1	4,096	1	256	1	16	1	1
2	2,097,152	2	131,072	2	8,192	2	512	2	32	2	2
3	3,145,728	3	196,608	3	12,288	3	768	3	48	3	3
4	4,194,304	4	262,144	4	16,384	4	1,024	4	64	4	4
5	5,242,880	5	327,680	5	20,480	5	1,280	5	80	5	5
6	6,291,456	6	393.216	6	24,576	6	1,536	6	96	6	6
7	7,340,032	7	458,752	7	28,672	7	1,792	7	112	7	7
8	8,388,608	8	524,288	8	32,768	8	2,048	8	128	8	8
9	9,437,184	9	589,824	9	36,864	9	2,304	9	144	9	9
A	10,485,760	A	655,360	A	40,960	A	2,560	A	160	A	10
B	11,534,336	B	720,896	B	45,056	B	2,816	B	176	B	11
C	12,582,912	C	786,432	C	49,152	C	3,072	C	192	C	12
D	13,631,488	D	851,968	D	53,248	D	3,328	D	208	D	13
E	14,680,064	E	917,504	E	57,344	E	3,584	E	224	E	14
F	15,728,640	F	983,040	F	61,440	F	3,840	F	240	F	15

ASCII CONVERSION TABLE

HEX	MSD	0	1	2	3	4	5	6	7
LSD	BITS	000	001	010	011	100	101	110	111
0	0000	NUL	DLE	SPACE	0	@	P	–	p
1	0001	SOH	DC1	!	1	A	Q	a	q
2	0010	STX	DC2	"	2	B	R	b	r
3	0011	ETX	DC3	#	3	C	S	c	s
4	0100	EOT	DC4	$	4	D	T	d	t
5	0101	ENQ	NAK	%	5	E	U	e	u
6	0110	ACK	SYN	&	6	F	V	f	v
7	0111	BEL	ETB	'	7	G	W	g	w
8	1000	BS	CAN	(	8	H	X	h	x
9	1001	HT	EM	)	9	I	Y	i	y
A	1010	LF	SUB	*	:	J	Z	j	z
B	1011	VT	ESC	+	;	K	[	k	{
C	1100	FF	FS	,	<	L	\	l	|
D	1101	CR	GS	-	=	M	]	m	}
E	1110	SO	RS	.	>	N	^	n	~
F	1111	SI	US	/	?	O	←	o	DEL

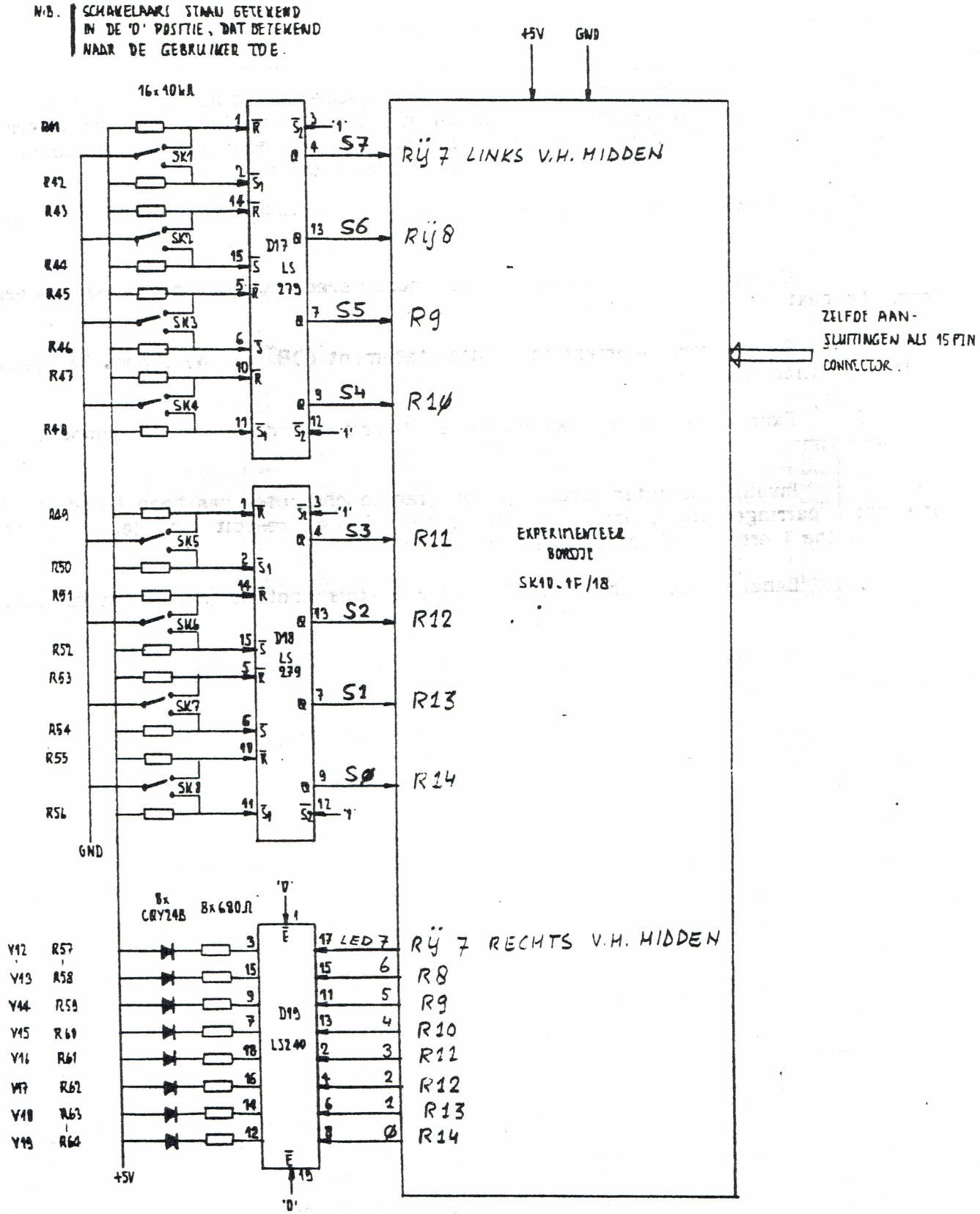
THE ASCII SYMBOLS

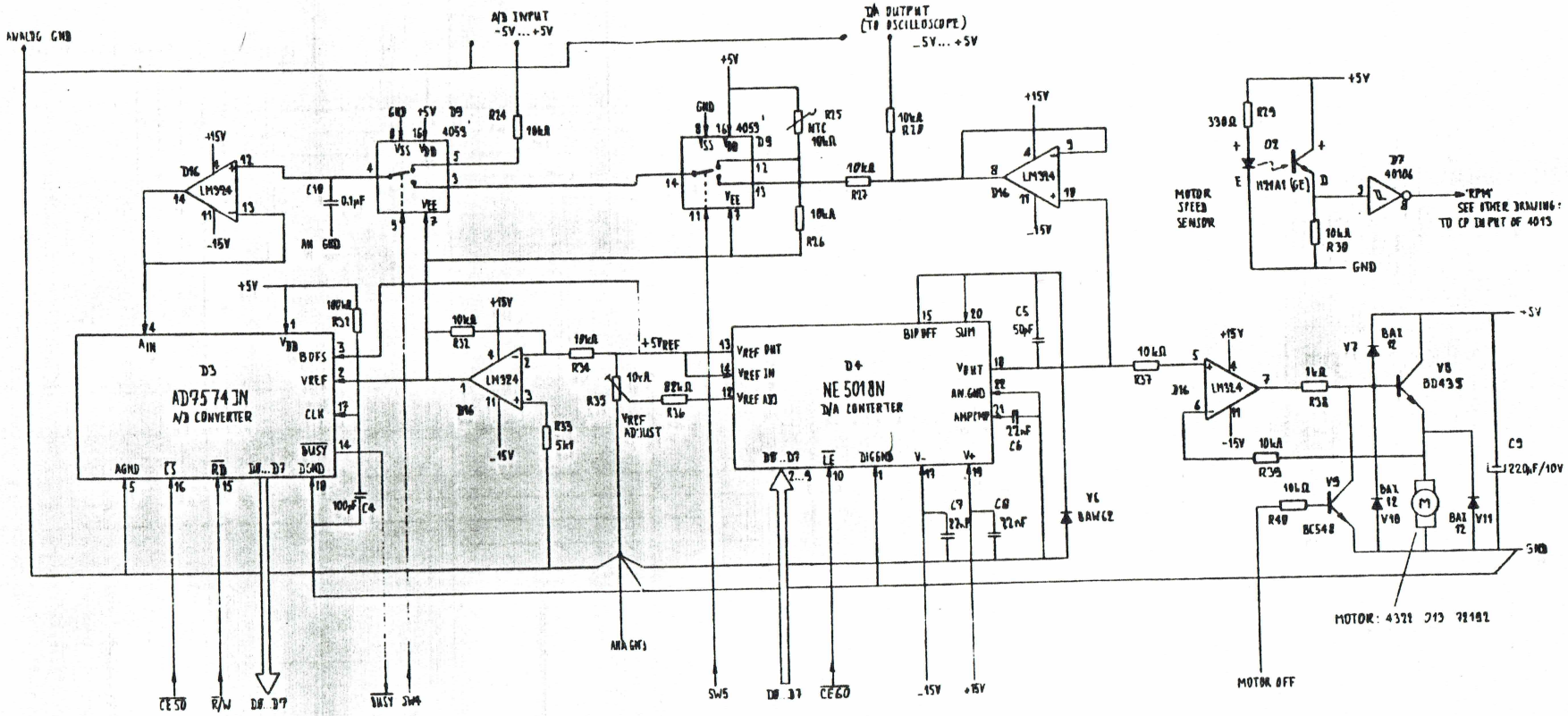
NUL – Null
SOH – Start of Heading
STX – Start of Text
ETX – End of Text
EOT – End of Transmission
ENQ – Enquiry
ACK – Acknowledge
BEL – Bell
BS – Backspace
HT – Horizontal Tabulation
LF – Line Feed
VT – Vertical Tabulation
FF – Form Feed
CR – Carriage Return
SO – Shift Out
SI – Shift In
DLE – Data Link Escape
DC – Device Control
NAK – Negative Acknowledge
SYN – Synchronous Idle
ETB – End of Transmission Block
CAN – Cancel
EM – End of Medium
SUB – Substitute
ESC – Escape
FS – File Separator
GS – Group Separator
RS – Record Separator
US – Unit Separator
SP – Space (Blank)
DEL – Delete

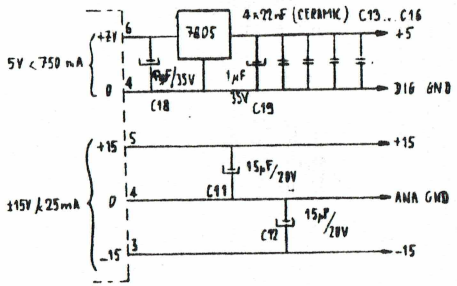
Appendix D. STANDAARD BUS-AANSLUITINGEN: H-64 BUS, VOOR 19"-RACKS.

(afgeleid van Huizink's (MI) 32-polige bus)

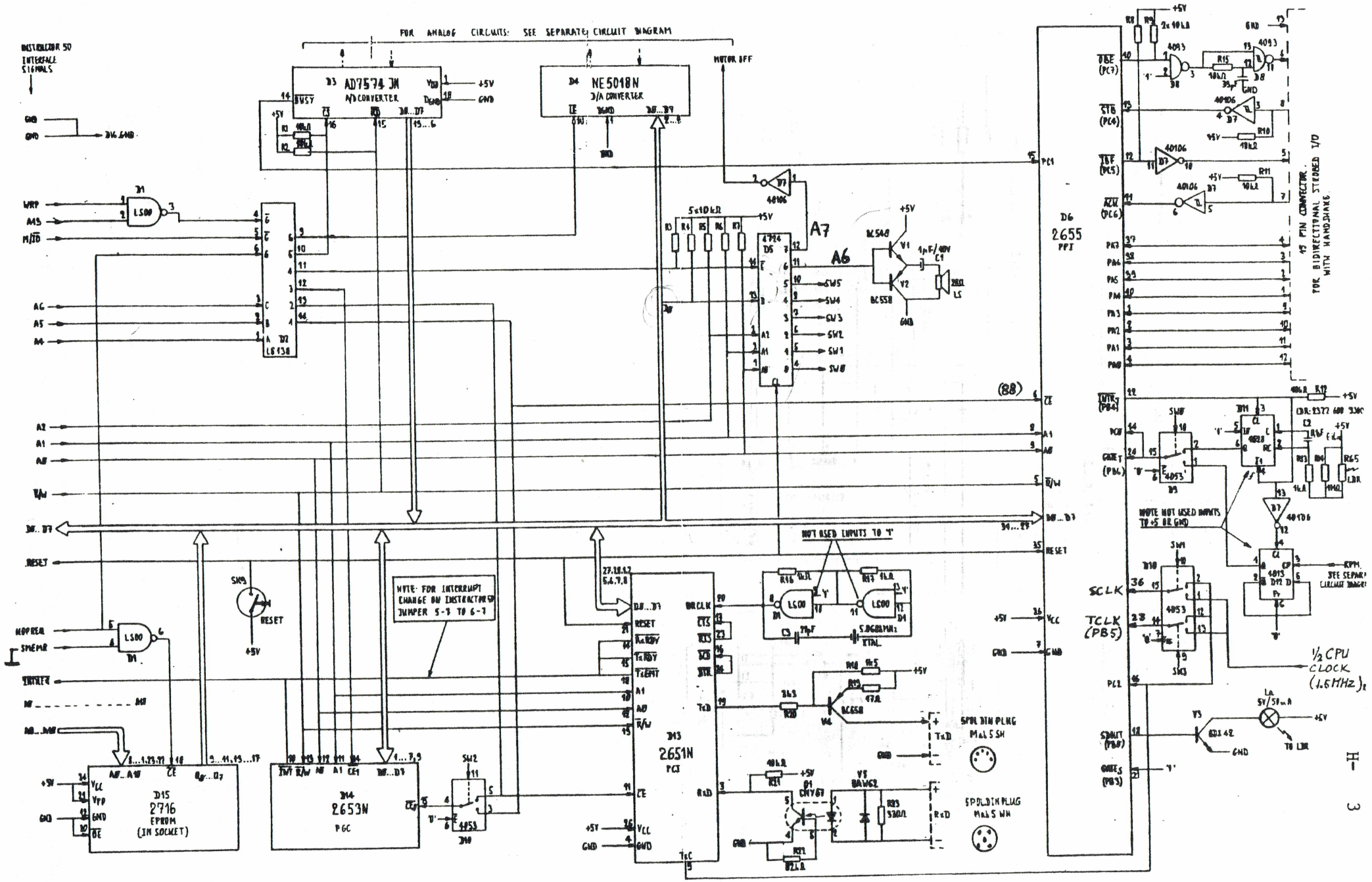
PEN		PEN	
1a	A0	1b	RDY
2a		2b	A1
3a	A2	3b	- WR
4a		4b	A3
5a	A4	5b	- RD
6a	S1	6b	A5
7a	A6	7b	IO/- M
8a	S0	8b	A7
9a	A8	9b	RFSH
10a	ALE	10b	A9
11a	- MemW	11b	
12a	HLDA	12b	- MemR
13a	CLK	13b	
14a	HOLD	14b	A14
15a	A15	15b	
16a	- INTA	16b	A10
17a	A11	17b	
18a	INTR	18b	A12
19a	A13	19b	
20a	RST 6.5	20b	- I/O R
21a	- I/O W	21b	
22a		22b	DB0
23a	DB1	23b	
24a	- RESET IN	24b	DB2
25a	DB3	25b	
26a		26b	DB4
27a	DB5	27b	
28a	-12V	28b	DB6
29a	DB7	29b	
30a	+12V	30b	RESET
31a	+5V	31b	+5V
32a	GND	32b	GND


N.B. SCHAKELAARS STAAN GETEKEND IN DE 'O' POSITIE, DAT BETEKEND NAAR DE GEBRUIKER TOE.
+5V
GND
16x10kΩ
R41
R42
R43
R44
R45
R46
R47
R48
SK1
SK2
SK3
SK4
D17 LS 279
S7
S6
S5
S4
RIJ 7 LINKS V.H. MIDDEN
RIJ 8
R9
R10
R49
R50
R51
R52
R53
R54
R55
R56
SK5
SK6
SK7
SK8
D18 LS 939
S3
S2
S1
SØ
R11
R12
R13
R14
GND
EXPERIMENTEER BORDJE
SK10-1F/18
ZELFDE AANSLUITINGEN ALS 15 PIN CONNECTOR.
8x CQY24B
8x680Ω
V12 R57
V13 R58
V14 R59
V15 R60
V16 R61
V17 R62
V18 R63
V19 R64
D19 LS240
LED 7
RIJ 7 RECHTS V.H. HIDDEN
R8
R9
R10
R11
R12
R13
R14
+5V





74LS00	1	2651	1
74LS138	1	2653	1
HEF 4013	1	2655	1
HEF 4053	2	NE5018	1
HEF 4081	1	AD7574	1
HEF 4528	1	LM324	1
HEF 4716	1	27-16	1
HEF 40106	1		


FOR ANALOG CIRCUITS: SEE SEPARATE CIRCUIT DIAGRAM
D3 AD7574 JN
A/D CONVERTER
D4 NE5018N
D/A CONVERTER
MOTOR OFF
D6 2655 PPI
25 PIN CONNECTOR FOR BIDIRECTIONAL STEREO I/O WITH HANDSHAKE
A7
A6
D15 2716 EPROM (IN SOCKET)
D14 2653N PGC
D13 2651N PCI
NOT USED INPUTS TO "1"
NOTE: FOR INTERRUPT CHANGE ON INSTRUCTION JUMPER 5-3 TO 6-7
RESET
HOPPER
SHEMER
SCLK
TCLK (PB5)
½ CPU CLOCK (1.5MHz)
SPOL DIN PLUG MAL 5 SH
SPOL DIN PLUG MAL 5 WH
TxD
RxD
TO LDR